

CRANFIELD UNIVERSITY
School of Aerospace, Transport & Manufacturing

PhD

Academic Year 2018 - 2019

David Barry

**Using flight data in Bayesian networks and other methods to
quantify airline operational risks**

Supervisor: Dr. Simon Place

September 2019

© Cranfield University 2019. All rights reserved. No part of this
publication may be reproduced without the written permission of the
copyright owner.

Abstract

The risk assessment methods used in airline operations are usually qualitative rather than quantitative, despite the routine collection of vast amounts of safety data through programmes such as flight data monitoring (FDM). The overall objective of this research is to exploit airborne recorded flight data to provide enhanced operational safety knowledge and quantitative risk assessments. Runway veer-off at landing, accounting for over 10% of air transport incidents and accidents, is used as an example risk.

Literature on FDM, risk assessment and veer-off accidents is reviewed, leading to the identification of three potential areas for further examination: variability in operational parameters as a measure of risk; measures of workload derived from flight data as a measure of risk; and Bayesian networks.

Methods relating to variability and workload are briefly explored and preliminary results are presented, before the main methods of the thesis relating to Bayesian networks are introduced.

The literature shows that Bayesian networks are a suitable method for quantifying risk and a causal network for lateral deviation at landing is developed based on accident investigation data. Flight data from over 300,000 flights is used to provide empirical probabilities for causal factors and data for some causal factors is modelled to estimate the probabilities of extreme events.

As an alternative to predefining the Bayesian network structure from accident data, a series of networks are learnt from flight data and an assessment is made of the performance of different learning algorithms, such as Bayesian Search and Greedy Thick Thinning.

Finally, a network with parameters and structure learnt from flight data is adapted to incorporate causal knowledge from accident data, and the performance of the resulting “combined” network is assessed.

All three types of network were able to make use of flight data to calculate relative probabilities of a lateral deviation event, given different scenarios of causal factors present, and for different airports, however the “combined” approach is preferred due to the relative ease of running scenarios for different airports and the avoidance of the lengthy process of modelling data for causal factor nodes.

The preferred method provides airlines with a practicable way to use their existing flight data to quantify operational risks. The resulting quantitative risk assessments could be used to provide pilots with enhanced pre-flight briefings and provide airlines with up-to-date risk information of operations to different airports, and enhanced safety oversight.

Recommendations for future work are to use the method to quantify other risks and trial the method in an operational environment with a different dataset.

Acknowledgements

I am hugely grateful to the Safety & Accident Investigation Centre at Cranfield University and Prof. Graham Braithwaite for giving me the opportunity to do this PhD, and of course to my supervisor Dr Simon Place.

I would like to thank the Review Team, Dr Athanasios Kolios and Dr Vassilios Pachidis, for their guidance and advice at each annual review.

Dr Colin Pilbeam, Dr Jim Nixon, Dr Rebecca Charles and Dr Matt Greaves have been very kind, supportive and great sources of practical advice on how to research and write, and Dr Ken McNaught of the Centre for Simulation & Analytics has been a much-needed sounding-board on Bayesian networks. Thank you to all five of you.

I must take this opportunity to thank my fabulous parents, Dreena and Norman, for providing a solid launchpad and steadfast support throughout my education and career.

Finally, all my love and thanks to my dear wife Tracy and our lovely daughter Orla..... You have been so wonderfully patient and understanding.

Table of contents

Abstract	2
Acknowledgements	4
Table of contents	5
List of figures	9
List of tables	13
Glossary	15
1 Introduction	19
2 Literature review	25
2.1 <i>Flight data monitoring</i>	25
2.1.1 Regulatory framework	25
2.1.2 The foundations of FDM – recorded flight data.....	26
2.1.3 FDM process and analysis.....	29
2.1.4 Research based on flight data and FDM	33
2.2 <i>Risk methods</i>	39
2.2.1 Risk assessment and risk management methods used in airline operations and maintenance	39
2.2.2 Risk quantification methods in aviation	48
2.2.3 Bayes’ theorem and Bayesian networks.....	51
2.2.4 Summary.....	60
2.3 <i>Runway veer-offs.....</i>	61
2.3.2 Summary.....	64
3 Theory	65
3.1 <i>Exploration of methodologies with potential application to risk assessment using flight data</i> 65	
3.1.1 Variability in performance.....	65
3.1.2 Workload and risk.....	66
3.2 <i>Bayesian networks and risk assessment</i>	67

3.2.1	Determining BN structure.....	67
3.2.2	Modelling nodes in a Bayesian network.....	68
3.2.3	Propagation of evidence through a Bayesian network	75
3.2.4	Bayesian network software.....	75
4	Methodology	77
4.1	<i>Flight data used in this research</i>	<i>77</i>
4.2	<i>FDM results data</i>	<i>78</i>
4.3	<i>Variability as an indicator of risk</i>	<i>79</i>
4.3.1	Process capability analysis using flight data.....	83
4.3.2	Variability in basic flight parameters.....	84
4.4	<i>Workload as an indicator of risk.....</i>	<i>88</i>
4.4.1	Identifying workload.....	88
4.4.2	Comparison between number of FDM events and recorded actions.....	91
4.5	<i>Summary of variability and workload methods.....</i>	<i>95</i>
4.6	<i>Bayesian network derived from accident report causal factors</i>	<i>96</i>
4.6.1	Causal factors identifiable from flight data	97
4.6.2	Proposed BN structure for lateral runway excursion.....	101
4.6.3	Deriving node priors from recorded flight data	102
4.6.4	Node modelling for Bayesian network.....	106
4.7	<i>Bayesian network learnt from flight data.....</i>	<i>125</i>
4.8	<i>Data preparation for learning Bayesian network structure.....</i>	<i>126</i>
4.8.1	Learning network algorithms in GeNIe.....	128
4.8.2	Discretisation of variables	129
5	Results.....	130
5.1.1	Scenarios.....	130
5.1.2	Comparing risk between airports.....	131
5.2	<i>Validation</i>	<i>133</i>
5.2.1	High-risk flights and LATGsd.....	134

5.2.2	Software comparison	136
5.3	<i>Further validation</i>	136
5.4	<i>Summary of causal network results</i>	136
5.5	<i>Comparison of results from learning algorithms</i>	138
5.5.1	Bayesian search	138
5.5.2	PC learning.....	148
5.5.3	Greedy Thick Thinning.....	149
5.5.4	Tree Augmented Naïve Bayes.....	152
5.6	<i>Comparison of network performance</i>	152
5.7	<i>Sensitivity analysis</i>	154
5.7.1	Sensitivity analysis of Bayesian Search network.....	155
5.7.2	Sensitivity analysis of PC learning network.....	155
5.7.3	Sensitivity analysis of Greedy Thick Thinning network	155
5.7.4	Sensitivity analysis of Tree Augmented Naïve Bayes network.....	156
5.7.5	Sensitivity analysis summary	156
5.8	<i>Learning parameters for causal network</i>	157
5.9	<i>Validation of learnt networks</i>	159
5.9.1	Bayesian Search network validation.....	159
5.9.2	Greedy Thick Thinning network validation	161
	Tree Augmented Naïve Bayes network.....	162
5.9.3		162
5.9.4	Causal network with learnt parameters validation	163
5.9.5	Greedy Thick Thinning on two-level discretised data without background knowledge	164
5.9.6	Greedy Thick Thinning on two-level discretised data with background knowledge	167
5.9.7	Summary of validation results	167
6	Discussion	169

6.1	<i>Variability</i>	170
6.2	<i>Workload</i>	171
6.3	<i>Bayesian networks</i>	172
6.3.1	The problem of validation.....	172
6.3.2	Network structure – learnt or applied?.....	174
6.3.3	Combining applied and learnt networks	180
6.3.4	The recommended approach.....	182
6.3.5	Dealing with the consequences	182
6.3.6	Applicability	185
6.3.7	Hybrid Bayesian networks	186
6.3.8	Summary of Bayesian network discussion.....	187
7	Conclusion	189
7.1	<i>Future work</i>	191
	References	193
	Appendix 1 – R code for converting CSV flight data files into R Objects	205
	Appendix 2 – R code for SPC and variability parameter extraction	209
	Appendix 3 – Examples of flight parameter variability during approach	216
	Appendix 4 – R code variability feature extraction from flight data	225
	Appendix 5 – Calculations of height above airfield (HAAL) and flight path angle (FPA).	228
	Appendix 6 – R code for workload extraction from flight data	229
	Appendix 7 – R code for runway excursion feature extraction from flight data	234
	Appendix 8 – R code for comparison of crosswind calculations	244
	Appendix 9 – R code for addition of runway data to existing dataset	247
	Appendix 10 – R code for crosswind modelling	252
	Appendix 11 – R code used for airspeed modelling	254
	Appendix 12 – R code use for glide slope modelling	257
	Appendix 13 – R code use for normal acceleration at landing modelling	259
	Appendix 14 – Example of R code for modelling a causal factor at a specific airport	262

List of figures

Figure 1: Example simulated distribution of maximum difference between actual and target airspeed below 500ft.	21
Figure 2: Typical 5x5 risk matrix. Source: ICAO Safety Management Manual, 3rd Edition.....	42
Figure 3. The bow tie (from Strategic Risk, November 2005)	42
Figure 4. Simple FTA example	46
Figure 5. Simple event tree.....	47
Figure 6: Simple Bayesian network.....	51
Figure 7: BN with probabilities	52
Figure 8: BN with updated probabilities	53
Figure 9: Updated BN.....	53
Figure 10. Simple SPC chart.....	66
Figure 11: Modelling operational data	71
Figure 12: Single parent and single child	72
Figure 13: Two parents and single child.....	73
Figure 14. Height above airfield (ft) vs distance from touchdown (NM) for 889 flights	82
Figure 15. Process Capability Analysis chart for height above runway from 1.5 to 1.4NM.....	83
Figure 16: Different methods for examining variability	85
Figure 17: Density plots of Total number of actions for approaches with and without FDM events	91
Figure 18: Screenshot from R showing number of flights from each group by arrival airfield.	92
Figure 19: Updated density plot of actions following changes to parameters.....	93
Figure 20: Density plots from refined data. The red plot is approaches that had no FDM events....	94
Figure 21: BEST difference of means for Total Number of Actions for flights with and without FDM events	94
Figure 22: Bayesian network for lateral runway excursion	101
Figure 23: Basic process for node modelling	107
Figure 24: Example Cullen & Frey graph from fitdistrplus	108
Figure 25: Empirical cumulative distribution function for observed absolute crosswind.	110
Figure 26: Cullen & Frey graph for crosswind	111

Figure 28: fitdistrplus graphical output for fitted gamma distribution using matching quantile estimation with revised probabilities.....	112
Figure 28: Histogram for CAS - Vapp at 50ft	115
Figure 29: fitdistrplus graphical output for Gumbel distribution	116
Figure 30: Histogram of glideslope deviation at 150ft with Johnson distribution in red.....	117
Figure 31: Q-Q plot for modelled glideslope data.....	118
Figure 32: Histogram of observed normal acceleration at landing (g)	119
Figure 33: Cullen & Frey graph for normal acceleration at touchdown.....	120
Figure 34: Gamma distribution fit plot for normal acceleration at touchdown.....	120
Figure 35: Second attempt at gamma distribution using extreme upper quantile	121
Figure 36: fitdistrplus output for Gumbel model	122
Figure 37: Track of CRJ900 landing at Turku Airport, 25th October 2017.....	126
Figure 38: Comparison of density distributions for lateral acceleration in calm conditions and crosswind conditions.....	134
Figure 39: BEST difference of means in LATGsd between high-risk and normal-risk groups	135
Figure 40: Learnt network using Bayesian Search algorithm	139
Figure 41: Bayesian Search learnt network, sample size = 500	142
Figure 42: Bayesian Search learnt network, sample size = 5000	143
Figure 43: Bayesian Search network learnt from data discretised into 8 intervals	144
Figure 44: Bayesian estimation of difference of means for asymmetric thrust with and without gusting winds.....	145
Figure 45: Bayesian Search network using 16 interval discretisation.....	146
Figure 46: Bayesian search learnt network using 8 interval hierarchical discretisation	147
Figure 47: Bayesian search network from 4 interval hierarchical discretisation	148
Figure 48: Network learnt using PC algorithm.....	149
Figure 49: Network learnt using Greedy Thick Thinning algorithm.....	150
Figure 50: Network learnt using Greedy Thick Thinning and 8 interval discretised data	151
Figure 51: Network learnt using Tree Augmented Naive Bayes algorithm	152
Figure 52: Sensitivity analysis of Bayesian Search learnt network.....	155
Figure 53: Sensitivity analysis of Greedy Thick Thinning network.....	156

Figure 54: Structure for causal network with learnt parameters.....	157
Figure 55: Sensitivity analysis for network with learnt parameters	159
Figure 56: ROC curve for Bayesian Search network validation	161
Figure 57: ROC curve for Greedy Thick Thinning network validation	162
Figure 58: ROC curve for Tree Augmented Naive Bayes network validation.....	163
Figure 59: ROC curve for causal network with learnt parameters validation.....	164
Figure 60: Greedy Thick Thinning learnt network from two-level discretised data without any background knowledge.....	165
Figure 61: ROC curve for Greedy Thick Thinning network learnt from two-level discretised data	166
Figure 62: Greedy Thick Thinning learnt network from two-level discretised data with background knowledge.....	167
Figure 63: Learnt network structure modified to include missing causal arcs.....	181
Figure 64: Runway at Trabzon airport, Turkey. By Ercan Karakas - https://www.jetphotos.com/photo/6012865 , GFDL, https://commons.wikimedia.org/w/index.php?curid=20306566	183
Figure 65: Runway at Funchal airport, Madeira. By Koshelyev - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6207088	184
Figure 66: Runway at Palm Springs International airport, USA. Photograph by D. Ramey Logan, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=39528001	184
Figure 67. Computed airspeed (CAS) vs distance to touchdown (NM)	216
Figure 68. CAS delta from target vs distance to touchdown (NM).....	217
Figure 69. Groundspeed vs distance to touchdown (NM)	217
Figure 70. Lateral acceleration (g) vs distance to touchdown (NM)	218
Figure 71. Longitudinal acceleration (g) vs distance to touchdown (NM)	218
Figure 72. Normal acceleration (g) vs distance to touchdown (NM).....	219
Figure 73. Magnetic heading delta to runway heading (degrees) vs distance to touchdown (NM)	219
Figure 74. Left engine N1 (%) vs distance to touchdown (NM).....	220
Figure 75. Right engine N1 (%) vs distance to touchdown (NM).....	220
Figure 76. Pitch angle (degrees) vs distance from touchdown (NM)	221
Figure 77. Roll angle (degrees) vs distance from touchdown (NM)	221

Figure 78. Radio altitude (ft) vs distance from touchdown (NM)	222
Figure 79. Rudder position (degrees) vs distance from touchdown (NM)	222
Figure 80. Left seat side-stick pitch (degrees) vs distance from touchdown (NM)	223
Figure 81. Right seat side-stick pitch (degrees) vs distance from touchdown	223
Figure 82. Left seat side-stick roll (degrees) vs distance from touchdown (NM)	224
Figure 83. Right seat side-stick roll (degrees) vs distance from touchdown (NM)	224

List of tables

Table 1. Safety risk probability table from ICAO SMM 3 rd Edition	40
Table 2. Safety risk probability table from ICAO SMM 1 st Edition.....	41
Table 3: NPT for P(U).....	52
Table 4: NPT for landing accident (L)	52
Table 5: Lateral runway excursion causal factors, derived from (Post, 2015).....	63
Table 6: Node probability table for approach speed high	72
Table 7: Node probability table for approach stability, given speed	73
Table 8: Node probability table for second parent	73
Table 9: Node probability table for child with two parent nodes	73
Table 10: GMD rankings for Phase 3b.....	87
Table 11: Flight data parameters to monitor workload.....	88
Table 12: Mean frequency of state changes	89
Table 13: Top seven approaches with most actions	90
Table 14: Summary statistics for approaches with and without FDM events.....	91
Table 15: Updated Table 14	93
Table 16: Causal factors identifiable from flight data	98
Table 17: Correlation matrix for wind calculation methods	103
Table 18: NPT for heavy precipitation	110
Table 19: NPT for runway wet.....	110
Table 20: NPT for crosswind.....	112
Table 21: NPT for Inaccurate information	113
Table 22: NPTs for technical issues	113
Table 23: NPT for gusting wind.....	113
Table 24: Frequency table for time with asymmetric thrust.....	114
Table 25: NPT for asymmetric thrust at landing.....	114
Table 26: NPT for airspeed vs target >20kt.....	116
Table 27: NPT for glideslope.....	118
Table 28: NPT for Unstable approach.....	119

Table 29: NPT for hard landing.....	122
Table 30: METAR visibility reports	123
Table 31: NPT for poor visibility	123
Table 32: Effect on probability of risk factors	131
Table 33: BN probabilities for lateral deviation at different airports.....	132
Table 34: Temporal tiers for background knowledge	138
Table 35: Strength of influence between nodes, , sample size = 50.....	141
Table 36: Strength of influence between nodes, sample size = 500	143
Table 37: Summary of effect of Bayesian Search sample size on learnt network.....	144
Table 38: Strength of influence for 8 interval discretised network	145
Table 39: Strength of influence of arcs from Greedy Thick Thinning algorithm	151
Table 40: Tables showing probabilities of high LATGsd, RUDsd and HDGdev, for different airports, derived by different learning algorithms	153
Table 41: Ranking of specific airport probabilities of lateral deviation calculated from different networks	154
Table 42: Causal network with learnt parameters airport ranking.....	158
Table 43: Bayesian Search network validation results	160
Table 44: Validation confusion matrix for Bayesian Search network	160
Table 45: Greedy Thick Thinning network validation results	161
Table 46: Validation confusion matrix for Greedy Thick Thinning network.....	162
Table 47: Tree Augmented Naive Bayes network validation results.....	162
Table 48: Validation confusion matrix for Tree Augmented Naive Bayes network.....	163
Table 49: Causal network with learnt parameters network validation results.....	163
Table 50: Validation confusion matrix for causal network with learnt parameters.....	164
Table 51: Validation accuracy results of Greedy Thick Thinning network LATGsd node with two- level discretisation	166
Table 52: Confusion matrix for Greedy Thick Thinning network with two-level discretisation ..	166
Table 53: Summary of validation results.....	168
Table 54: noisy-or parameters for combined network	181
Table 55: Comparison of lateral deviation probabilities calculated from wholly causal and combined causal/learnt networks	181

Glossary

AFDX	Avionics full-duplex switched ethernet
ARINC	Aeronautical Radio Inc
ARMS	Airline risk management solutions
<i>arrid</i>	Airport ID
<i>asymTHR</i>	Duration of asymmetric thrust from touchdown -5 seconds to the end of the landing roll (seconds)
ATC	Air traffic control
BEA	British European Airways
BHD	Belfast City airport
Bit (computing)	Binary digit
BN	Bayesian network
CAA	Civil Aviation Authority
CAP	Civil aviation publication
CAS	Computed airspeed (kt)
CATS	Causal modelling of Air Transport System
CFIT	Controlled flight into terrain
CI	Credible interval
CSV	Comma separated values
DAG	Directed acyclic graphs
DAR	Digital aircraft conditioning monitoring system recorder
DLM	Dalaman airport, Turkey
DME	Moscow Domodedovo airport
DMU	Data management unit
EASA	European Aviation Safety Agency
ECAM	Electronic centralised aircraft monitor
EDI	Edinburgh airport
EFIS	Electronic flight instrumentation system
EGPWS	Enhanced ground proximity warning system
EOFDM	European operators flight data monitoring forum
ERC	Event risk classification
ETA	Event tree analysis
FAA	Federal Aviation Administration
FDIMU	Flight data integrated management unit
FDM	Flight data monitoring
FDR	Flight data recorder

FMEA	Failure mode and effects analysis
FOQA	Flight operations quality assurance
FORAS	Flight operations risk assessment system
FPA	Flight path angle (degrees)
FTA	Fault tree analysis
GeNie	Bayesian network software
GMD	Gini's mean difference
GMM	Gaussian mixture models
GPS	Global positioning system
GRN	Gated recurrent networks
<i>GSlope150</i>	Glideslope deviation at 150ft (dots)
GTT	Greedy thick thinning
HCL	Hybrid causal logic (FAA)
<i>HDGdev</i>	Heading deviation during landing roll (degrees)
HFACS	Human factors analysis and classification system
Hz	Hertz
IATA	International Air Transport Association
ICAO	International Civil Aviation Organisation
IRP	Integrated risk picture (Eurocontrol)
kg	Kilogrammes
km	Kilometres
kt	Knots
<i>LATGsd</i>	Standard deviation of lateral acceleration during landing roll
LBA	Leeds Bradford airport
LHR	London Heathrow airport
LOCI	Loss of control inflight
LOOCV	Leave one out cross validation
LSTM	Long short-term memory
<i>maxG</i>	Maximum recorded normal acceleration during landing (g)
MCTOM	Maximum certified take-off mass (kg)
METAR	Aviation routine weather report
<i>metVizkm</i>	Visibility, derived from METAR reports (km)
<i>metXgust</i>	Gusting winds, derived from METAR reports (true/false)
MKAD	Multiple kernel anomaly detection
N1/N2/N3	Shaft speeds in a jet engine (%)
NASA	National Aeronautics and Space Administration
NASA-TLX	NASA task load index

NCE	Nice Côte d'Azur International airport
NM	Nautical miles
NOTAM	Notice to airmen
NPA	Non-precision approach
NPT	Node probability table
P1/P2	Pilot 1/Pilot 2
$P(A CF)$	Probability of an accident, given the presence of a causal factor
$P(A)$	Probability of an accident
PB	Push button
$P(CF A)$	Probability of a causal factor, given an accident has occurred
$P(CF)$	Probability of a causal factor occurring
PCMCIA	Personal Computer Memory Card International Association
PRA	Probabilistic risk assessment
QAR	Quick access recorder
<i>rLandXwind</i>	Recorded crosswind component at landing (kt)
ROC	Receiver operating characteristic
<i>RUDsd</i>	Standard deviation of rudder deflection during landing roll
RNN	Recurrent neural networks
<i>runWet</i>	Runway wet, derived from METAR reports
SAG	Safety Action Group
SESMA	Special Event Search and Measurement Analysis
SIRA	Safety issue risk assessment
SME	Subject matter expert
SMM	Safety management manual
SOP	Standard operating procedure
<i>spdDelta50</i>	Difference from target airspeed at 50ft radio height
SPC	Statistical process control
SRB	Safety Review Board
SVM	Support vector machines
TAN	Tree augmented naïve Bayes
TB	Terabytes
TD	Touchdown
TCAS	Traffic collision avoidance system
TRK/FPA	Track/Flight path angle (Airbus flight director mode)
Vapp	Target approach speed (kt)
VHF1/2/3	Radio transmission parameters
Word (computing)	Unit of data used by a processor. Commonly 12 bits for flight data.

wps

words per second

1 Introduction

This research is born of frustration. With 22 years of experience working in aviation safety, including 11 as a safety manager of a large airline, the author has often found himself frustrated at the limited use of quantitative methods to assess operational risk. In other areas of aviation, such as airworthiness and design, quantitative methods, such as Fault Tree Analysis (FTA), are routinely used, however this is not the case in the operations domain. Some methods give the allusion of being quantitative, but usually they are actually qualitative (e.g. risk matrices, bow tie diagrams) (Hubbard, 2012), as discussed later in the Literature Review. This can be dangerous, as it provides airline practitioners and management with false comfort, believing risk has been assessed with scientific rigour, when in fact the result has relied upon subjective judgement.

Wholly qualitative assessment would be perfectly acceptable if sources of quantitative data were not available, however airlines have rich sources of data such as safety reports, audit findings, investigation findings and, significantly, recorded flight data.

Flight data recording has been around since the earliest days of manned flight, with Charles and the Robert brothers taking manual recordings of pressure and temperature during their early balloon ascents (Ilčev, 2018) in the 18th century. Later, it is reported that the Wright Brothers had instruments to make basic recordings during the first powered flights (National Air and Space Museum, 2015), but it was later in the 1950's that the first crash protected flight data recorders (FDR) as we know them today first appeared.

Some early FDRs recorded data by scratching foil as it was wound between two rollers, while others used points of light to record traces onto photographic paper. These recorders were difficult and time-consuming to transcribe. Later versions recorded data digitally onto steel wire, which provided excellent fire resilience properties, however they were not suited to regular or routine downloading and had relatively low data density. Steel wire recorders were gradually replaced by magnetic tape, which allowed higher data recording rates, but were also not well suited to routine downloading and were usually only downloaded after an incident or for maintenance.

Today most FDRs use solid-state memory which has very good survivability properties and allows higher data recording rates (e.g. > 1,024 words per second). Due to the absence of moving parts and media, they can be downloaded regularly (i.e. every 24 hours), however usually FDRs are located in the tail of the aircraft where they have the most impact protection and this can make them relatively inaccessible. Today it is common for airlines to equip their fleets with additional recorders which are more easily accessed.

In the early 1970's, British European Airways (BEA) had to prove the safety of the new auto-land technology of its Hawker Siddeley Trident fleet. Recorded flight data would be used to do this, by analysing parameters from each landing, however routine downloads from the FDR would not have been practicable, so a new type of recorder was fitted; the quick access recorder (QAR). The QAR recorded similar data to the FDR, however the data could be downloaded much faster, allowing routine regular downloads to be made. Parameters from the QAR, such as vertical speed, airspeed and flightpath deviations, could be analysed post-landing to see if any "special events" (Anon., 1971) had occurred. Such was the value of this data, that it was soon used to monitor not just automatic landings, but entire flights for these "special events". This was the birth of flight data monitoring (FDM) as we know it today. To this day, British Airways' FDM programme is known internally as SESMA – Special Event Search and Measurement Analysis.

Other large airlines started investing in their own FDM programmes when they realised what a valuable source of safety information it provided. The investment was not trivial; software alone would typically cost hundreds of thousands of pounds, QARs would need to be fitted to aircraft, engineers tasked with routine data downloading and departments set-up to validate and analyse the resulting data. During the 1990s, FDM was no longer the preserve of large flag carrier airlines and smaller airlines were establishing their own programmes. Eventually, on the 1st January 2005, ICAO made FDM a mandatory practice for all commercial air transport operations of aircraft with a maximum take-off weight in excess of 27,000kg.

Modern aircraft, such as the Airbus A350 and Boeing 787, record thousands of parameters on their QARs. The parameter set, known as the data frame, consists of measurements and condition status from a wide variety of sensors and systems, such as:

- Flight control positions
- Control surface positions
- Aircraft attitude
- Airspeed, altitude, vertical speed
- Geographic position
- Engine thrust, temperatures, pressures and vibrations
- Autopilot modes and selections
- Accelerations (normal, longitudinal and lateral)

Most parameters in the data frame are recorded at 1Hz, however some that are very dynamic (e.g. normal acceleration) may be recorded at 8Hz or more. Others which are less dynamic, may be recorded at rates of ¼ Hz or lower.

A typical 1-hour flight on a modern aircraft recording 1,500 parameters could easily result in over five million data points; scale this up to an airline operating two hundred aircraft over the course of a year, and over 200 billion data points (*5 million data points x 200 aircraft x 6 flights per day x 365 days*) would be generated.

Given this large amount of data available for analysis, it is somewhat surprising that flight data analysis methods today are not much different to the methods used in the 1970s. “Special events” remain the mainstay of the analysis (Civil Aviation Authority, 2013). These events are algorithms coded into the flight data replay software which find exceedances of flight manual limits and deviations from standard operating procedures. Appendix B of CAP739 (Civil Aviation Authority, 2013) provides a typical list of these events.

In addition to special events, it is quite common now for the replay software to take “measurement” snapshots during flights. Typical measurements might be:

- Height at gear extension
- Maximum pitch attitude at take off
- Groundspeed at touchdown
- Maximum rate of descent below 1000ft

These measurements allow normal operations to be analysed, not just the exceptional special events.

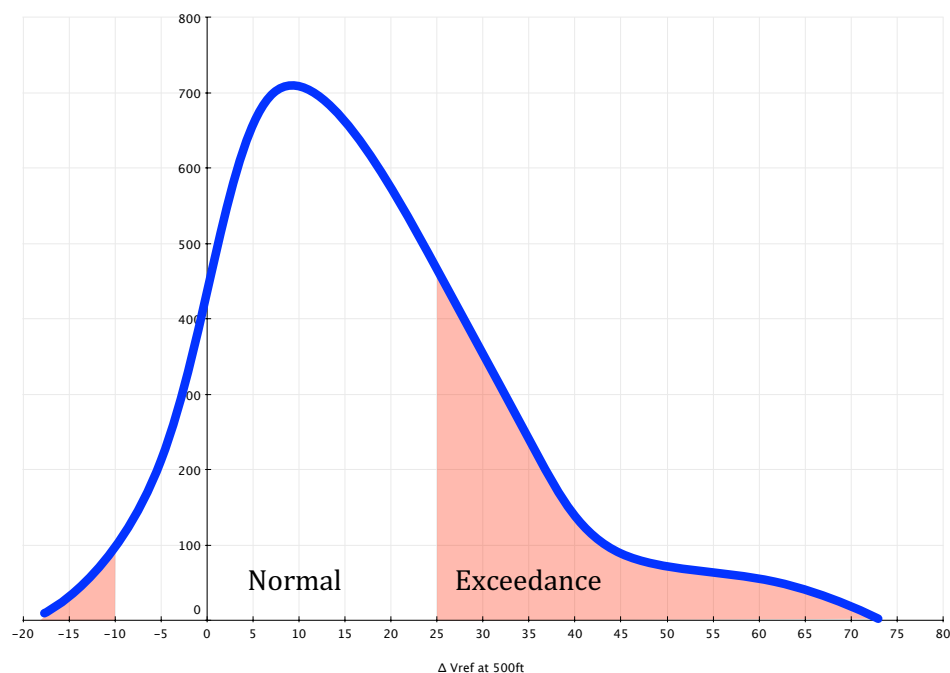


Figure 1: Example simulated distribution of maximum difference between actual and target airspeed below 500ft.

Figure 1 shows a simulated example distribution of maximum difference between actual and target airspeed below 500ft. Traditional exceedance or “special” events would trigger for the extremes, in this case -10kt and +25kt; the data from normal operations would be ignored, but is now commonly used for measurement analysis.

Once the data has been replayed and exceedances found, the events and measurements are stored in a database with some additional contextual information and used to produce safety performance indicators for input into the airline’s safety management system.

At the core of safety management systems is risk assessment and management and in 2011 the UK CAA, working with industry, published a list of the seven most significant risks facing the industry (Civil Aviation Authority, 2011):

- Loss of control
- Runway overrun or excursion
- Controlled flight into terrain
- Runway incursion and ground collision
- Airborne conflict
- Ground handling operations
- Airborne and post-crash fire

Many of the individual “Special events” and measurements from FDM can be mapped on to some of these risks as precursors. For example, a high angle of bank event may be mapped to the loss of control risk. Usually the individual events have a basic risk weighting and commonly labelled low, medium or high, depending on the magnitude of exceedance. An airline can therefore study the prevalence of “high” risk weighted events over time, or at different airports, however this method does little to help an airline quantify actual risk or be able to answer questions such as:

- What is our risk of a runway veer-off?
- At which airport are we most likely to experience a runway veer-off incident?

Answers to questions like these are much more helpful in managing risk than simple precursor trends, however providing answers to these questions can be complex, even with the abundant data available today. The overarching aim of this research is to develop a method which can make use of recorded flight data to routinely quantify risk and provide answers to questions like these.

As described at the beginning, a goal of this research is to make better use of flight data to provide additional and better quantitative risk assessments, and various methods have been identified during the Literature Review. These methods have been investigated to test whether they have the potential to be used to enhance existing FDM and provide new risk information.

The literature (Fenton and Neil, 2007) indicates that Bayesian methods are being used for risk assessment, albeit not commonly in aviation, and Bayesian networks specifically can exploit mixtures of objective information (such as flight data) and subjective information, where no hard data is available. A method is developed to create a Bayesian network which makes use of flight data to quantify risk and the resulting method is demonstrated using one of the seven significant risks: runway excursion. This risk is chosen because runway excursion incidents and accidents occur relatively frequently, accounting for around a quarter of all incidents and accidents in air transport (Australian Transport Safety Bureau, 2008). The literature shows that a large proportion (over 40%) of these are landing veer-off events (Flight Safety Foundation, 2009a), hence that risk is used to demonstrate the method.

To achieve the aim of this research, it is broken down into four objectives, as follows:

1. Appraise the literature on risk quantification methods, flight data monitoring and runway veer-off risk
2. Evaluate novel measures of risk using airborne recorded flight data and FDM
3. Develop a Bayesian network which can make use of flight data to quantify risk in an airline operational domain
4. Demonstrate how the Bayesian network can be used with a large data set (>300,000 flights) to quantify risk, using runway veer-off risk as an example

Following on from this introduction, the thesis appraises the literature on the three topics and highlights areas where they overlap. In Chapter 3, the background theory to methods which have the potential to make use of flight data to generate new risk knowledge are described. While the Literature Review did not find any reference to the application of these methods in airline operations, the methods are introduced with descriptive literature to support the rationale for their potential use. This is followed by an introduction to the theory on Bayesian networks.

Chapter 4 describes the Methodology, beginning with descriptions of the flight data available to this research. It continues with descriptions of methods based on variability and workload, and a brief summary of the results of these methods. The Chapter continues by describing in detail the steps required to build Bayesian networks which make use of flight data.

Results from Bayesian networks where the structure has been derived from accident causal factors are described in Chapter 5. Due to the absence of veer-off events in the dataset, a proxy

measure of lateral deviation is used to help validate the results. The chapter continues by detailing the results from networks where the structure has been learnt from flight data, rather than derived from accident causal factors.

In Chapter 6, the results are discussed and the rationale for the recommended approach is made.

The Conclusions are stated in Chapter 7, which advocates the method's practical application and includes recommendations for future work.

2 Literature review

2.1 Flight data monitoring

The previous chapter gave an overview of the origins of FDM and a basic introduction to flight data, however it is worth looking at both of these areas in more detail to help understand the potential and limitations of flight data and FDM. This chapter will also evaluate previous research focussing on FDM.

2.1.1 Regulatory framework

By the time FDM became an ICAO mandatory practice in 2005, many operators and airlines already had mature programmes in place. Accordingly, the regulations which were introduced, reflected current practice and so many airlines did not have to make changes to existing programmes. However, those operators without existing FDM programmes had to invest in FDM hardware and software, or opt to outsource their programmes to service providers.

The ICAO regulations, enshrined in Annex 6 (International Civil Aviation Organisation, 2010), state that:

3.3.5 Recommendation. An operator of an aeroplane of a certificated take-off mass in excess of 20 000 kg should establish and maintain a flight data analysis programme as part of its safety management system.

3.3.6 An operator of an aeroplane of a maximum certificated take-off mass in excess of 27 000 kg shall establish and maintain a flight data analysis programme as part of its safety management system.

To provide context, this means that aircraft such as the BAe Jetstream 41, Saab 340 and ATR42-500 fall outside the recommendation. It is, however, a *recommended practice* for types such as the Embraer ERJ 145, ATR 72-600 and Bombardier CRJ200, whereas it is *mandatory* for Bombardier Q400, Embraer 170, Avro RJ-85 and larger aircraft.

The ICAO requirements are reflected in European Commission Regulation No 859/2008 (European Aviation Safety Agency, 2008), specifically in OPS 1.037 which states:

4. a flight data monitoring programme for those aeroplanes in excess of 27 000 kg MCTOM. Flight data monitoring (FDM) is the pro-active use of digital flight data from routine operations to improve aviation safety. The flight data monitoring programme shall be non-punitive and contain adequate safeguards to protect the source(s) of the data;

In the “*Acceptable Means of Compliance and Guidance Material*” to the regulation (European Aviation Safety Agency, 2014), the overall intentions of FDM with regard to risk are stated:

Part ORO.AOC.130

An FDM programme should allow an operator to:

- (1) identify areas of operational risk and quantify current safety margins;*
- (2) identify and quantify operational risks by highlighting occurrences of non-standard, unusual or unsafe circumstances;*

These two points are especially interesting, because they require operators to quantify operational risks. FDM is very good at “...*highlighting occurrences of non-standard, unusual or unsafe circumstances*”, however difficulty arises when trying to translate information on those occurrences to a measure of risk. For example, how does the airline analyst translate the typical 59 events listed in CAP 739 (Civil Aviation Authority, 2013) (e.g. Approach speed high below 500 ft AAL, High normal acceleration at landing) to the risks identified in the CAA “Significant Seven” (Civil Aviation Authority, 2011)? It is that *translation* which is the novelty of the work described in this thesis.

The ICAO and EU regulations then filter down to national level and in the UK are embodied in the “Air Navigation Order” (Civil Aviation Authority, 2009) Article 94. In addition, the UK CAA has produced extensive guidance material in CAP 739 (Civil Aviation Authority, 2013), providing aircraft operators with nearly 200 pages of detail of how FDM programmes can be implemented. It is a useful resource for airlines, analysts and safety managers embarking on FDM.

The ICAO regulations are reflected at a national level around the globe, however the US FAA is a notable exception. In the US, Flight Operations Quality Assurance (FOQA) as it’s called there, is a voluntary safety programme and is described comprehensively in FAA Advisory Circular 120-82 (Federal Aviation Administration, 2004). Despite being voluntary, it has been widely adopted with many large US carriers having implemented FOQA well before the ICAO regulations were established in 2005.

2.1.2 The foundations of FDM – recorded flight data

There are essentially four necessary components or functions required to produce airborne recorded flight data. They are:

1. Sensors and avionics which measure and calculate parameters relating to flight and the operation of the aircraft.

2. Data buses for moving the data around.
3. Acquisition units for sampling data from the buses and packaging data for output to data recorders.
4. Data recorders for storing data for later retrieval or transmission.

A modern aircraft needs huge amounts of information to operate. Flight control computers need to know the aircraft's attitude, speed, altitude, accelerations, configuration and a multitude of other parameters; flight warning computers (e.g. EGPWS, TCAS) need to know the spatial positioning of the aircraft; and of course, the pilots need to have access to all of this information too. How the sensors and different avionics systems communicate is defined, for many aircraft types, in ARINC (Aeronautical Radio Inc.) standards 429 and 629, however recent aircraft, such as the B787 and A350 use an ethernet based system known as Avionics Full-Duplex Switched Ethernet (AFDX) (McHale, 2005).

Other ARINC standards, 717 and 767 (Nahapetian, 2014; Sudolsky, 2009), define the format for the data stream output from the acquisition unit to the recorders.

The data used later in this research was recorded onboard Airbus A320 series aircraft, conforming to ARINC 429 and 717 standards.

2.1.2.1 Description of flight data

While the overall structure of recorded flight data will comply with one of the standards, there is variation in how the data is recorded, even on aircraft from the same manufacturer. The main drivers for these differences are:

- Aircraft age (e.g. an A320 which came of the production line in 1995 will record data differently to an aircraft produced today),
- Engine type. Aircraft manufacturers sometimes offer a choice of engine types for an aircraft and this can lead to different parameters being recorded (e.g. some engines have 3 shafts and record N1, N2 and N3 shaft speeds, rather than just N1 and N2)
- Avionics fit. At purchasing, different avionics systems may be selected by the operator ("buyer furnished equipment") and systems such as the data acquisition unit will affect the parameters recorded.

These differences result in variation in the recorded output format, which is known as the *data frame*.

2.1.2.1.1 The data frame

ARINC 717 data frames have roots in an era when data storage was costly and low-density by today's standards. The main considerations were getting as much information on to the

recorder media, typically magnetic tape (Bureau d'Enquêtes et d'Analyses pour la Sécurité de l'Aviation Civile, 2005), as efficiently as possible. The data recorded onto tape was digitalized and it was envisaged that 12 bits would be sufficient to represent most recorded values, hence the use of 12-bit words in the ARINC 717 standard. For example, if all bits in a word are set to 1, the resulting binary number 111111111111 is equivalent to a decimal number of 4095, allowing parameters such as airspeed, radio height and heading to be recorded in a single word. For larger values, such as pressure altitude, the resolution can be made coarse so as to fit into the 12-bit word, or part of another word can be used in addition.

The words are arranged into four subframes, which make up the entire data frame. The recording of the data frame repeats on a four second cycle and each subframe represents one second's worth of data. Subframes usually consist of 64, 128, 256, 512, 1024 etc words, with the most recent aircraft recording 2048 words in every subframe, equivalent to 2048 words per second. The size of the subframe therefore describes both the rate of data recording and the amount of data. It follows that larger subframes result in larger overall data frames, allowing more parameters to be recorded.

A typical data frame consists a mixture of digitised analogue measurements (e.g. airspeed, altitude) and discrete parameters (e.g. landing gear up or down, seatbelt sign on or off). The analogue measures will use several bits in a word, if not the whole word or more, whereas discrete parameters may only use a single bit. Some discrete parameters, for example flap lever position, may require several bits to record the lever position (e.g. flap 0, 1, 2, 3 & FULL would require 3 bits to record the five possible states). It is important to note that the data frame size, expressed in words per second, does not describe exactly how many parameters are recorded, but does provide an indication. The data used in this research was recorded at 128wps, with the data frame containing 370 parameters; a data frame recorded at 512wps could record well over one thousand parameters.

The benefit of larger data frames is not just more recorded parameters; the additional bits available mean that parameters can be recorded at higher frequencies and with better resolution. For most parameters, a recording frequency of once per subframe (1Hz) is sufficient, however where parameters are very dynamic in nature (e.g. accelerations, pitch, roll) they are recorded at higher frequencies. Normal acceleration, for instance, is usually recorded (and required to be recorded on the FDR) at 8Hz or more. Likewise pitch attitude is usually recorded at 4Hz. This is achieved by the parameter having multiple, equally spaced interval locations in the subframe. For example, in the 128wps data used in this research, pitch attitude is recorded in every subframe in words 22, 54, 86 and 118.

Parameters which are less dynamic and change more slowly, such as aircraft weight, may be recorded less frequently at, say, 0.25Hz (once in each data frame cycle). When data frames were smaller and space was very limited, it was not uncommon for some parameters be recorded even less frequently, through using a super-frame. The super frame used several bits as a counter, and the counter indicated which parameters were located in a particular word, so for example, when the counter was at 1, word X could contain the calendar month, when it was at 2, word X might contain engine serial number and so on. Note that the word could still contain multiple parameters. The larger data frames today mean that this complication can usually be avoided.

The digitized recording is a stream of bits, made up of these 12-bit words. To allow for later decoding, synchronisation words are recorded once every second at the start of each subframe. The specific bit pattern for the synchronisation words is defined in the ARINC standard. The synchronisation words allow software to later decode the recording and convert the bitstream into parameters in engineering units and discrete parameters with the correct sense i.e. up/down, true/false, on/off etc.

When decoding the recorded data stream, the software will look for the subframe 1 synchronisation word, then move ahead to find the next and so on until the start of each subframe is found. The synchronisation word is in the first recorded 12-bit word, so once all the synchronisation words are found, the software will be able to locate the bit positions of all the other parameters in the data frame. A map, or data frame layout, is provided by aircraft and avionic manufacturers which specifies the location for each parameter e.g. the position of the gear up or down parameter could be described as “every subframe, word 21, bit 3”. Once the data has been converted into meaningful parameters, FDM software can be used to help interpret that data and generate safety information from it.

2.1.3 FDM process and analysis

2.1.3.1 FDM process

The FDM process varies from airline to airline, depending on a variety of factors such as fleet size, aircraft types, equipment fit and so on. There are however common fundamentals.

2.1.3.1.1 Data recording and recovery

One of the primary tasks for a flight data recording system is to send the flight data, as described in 2.1.2.1, to the crash protected flight data recorder (FDR) for accident investigation purposes. FDRs can be downloaded and the data used for FDM, however they are only required to record the last 25 hours of data (International Civil Aviation Organisation, 2010), with older data being

overwritten. It is more common for the acquisition unit to output data also to a quick access recorder (QAR), which is a unit with removeable storage media, in an accessible location such as the avionics bay. The QAR might record an identical copy of the data sent to the FDR, but on some aircraft the QAR data frame can be different to the FDR data frame, albeit with a lot of commonality. On some aircraft types, it is possible for the operator to modify the data frame to record additional parameters, or parameters at a higher resolution or frequency.

The storage media will typically have a large enough capacity to record at least several weeks of data, but in order to ensure the data is processed in a timely manner, the media is usually retrieved and replaced as part of an aircraft's daily check, or on a weekly basis. It is important that the media is retrieved regularly because FDM can help identify exceedances of flight manual limits (e.g. flap speeds) which may require maintenance action.

The data retrieval can present logistical problems, particularly if an operator has a large fleet with several bases, so an alternative solution exists whereby data can be transmitted automatically after each flight, via mobile telephone or wifi networks. These "wireless" systems can be very efficient, resulting in rapid data recovery and very high (e.g. > 99%) data capture rates.

2.1.3.1.2 FDM data replay and validation

Once the data is received, it is *replayed* through FDM software where it is converted from the raw binary stream into meaningful parameters. The replay can often be started automatically when the FDM software detects that new data has arrived, or started manually by a safety analyst.

Flight data is usually very reliable and of good quality. Magnetic tape-based recorders and those with spinning disks have largely been replaced by solid state media, meaning that issues of vibration and dirt on read/write heads is now avoided. However, even though the data quality is usually very good, problems can still arise, resulting in potentially misleading information in safety databases. Problems can arise as a result of:

- Faulty sensors onboard aircraft.
- Incorrectly coded data frame within acquisition unit.
- Errors in manufacturer's data frame specification
- Errors in FDM software vendor's code

2.1.3.2 FDM analysis

With FDM having roots in the 1970s, it's not surprising that there are several FDM software programs and services available to aircraft operators today. Some have been developed in-house by large airlines (e.g. British Airways' Flight Data Analysis), others from avionics manufacturers (e.g. Airfase from Teledyne) and others that have grown independently as the use of FDM has spread (e.g. Aerobytes). Some operators elect to outsource the data processing to FDM service providers (e.g. Flight Data Services, Teledyne), freeing the operator from infrastructure issues such as data warehousing and perhaps some labour-intensive tasks, such as data validation.

Having converted the raw flight data into meaningful parameters, the next task of the FDM program or service is to search for exceedances of parameter limits and take snapshots or measurements of certain parameters from every flight.

2.1.3.2.1 Exceedance detection

A typical FDM system will search for one hundred or more exceedances of predefined limits or *event* algorithms (Civil Aviation Authority, 2013). These *events* can be quite simple in definition, for example, if airspeed is greater than 200kt and aircraft is within 2 minutes of landing, flag it as an *event*. Others can require more complicated algorithms, such as deep landing events where recorded position may lack the necessary resolution for an accurate estimate, or low buffet margin where aircraft are being operated in the region of "coffin corner"¹.

These exceedance events are often based on airline standard operating procedures or aircraft flight manual limitations and the limits can be set and adjusted by the operator. Typically, there are two or three event levels (e.g. low, medium or high severity) and the operator can set a threshold and time-over-threshold necessary to trigger the event. This is a basic automated risk assessment of the event, with a "high severity" perhaps indicating a need for further investigation or action. An example might be *high normal acceleration on landing*; the operator might set an informational level event (*low*) at say 1.7g, a *medium* level event requiring further investigation at say 2g, and a *high* level event requiring a maintenance inspection at 2.5g.

This basic risk assessment might be seen by some operators, and perhaps regulators, as satisfying the EU-OPS requirement, described in 2.1.1, that FDM should "*identify and quantify operational risks by highlighting occurrences of non-standard, unusual or unsafe circumstances*". However it seems like quite a leap to describe that basic assessment as "*quantifying operational risk*"; for instance, how can the operator translate events such as *high normal acceleration on*

¹ Coffin corner is the term describing a region of the flight envelope where if the aircraft slows or accelerates it will stall.

landing into a quantified risk of a runway veer-off? This research corrects this disparity by making the translation from flight data measures to a quantified measure of a recurring operational risk.

2.1.3.2.2 *Measurement analysis*

The most basic form of FDM is exceedance detection which highlights the extremes of operational parameters. The weakness of this approach is that it focusses on exceptions, so oversight of what is normal in the operation is limited. To overcome this, *measures* can be taken for every flight, not just those that have exceeded some operational limit. For example, rather than just highlighting the *late* extension of landing gear during an approach, the height of extension of landing gear for *every* approach is stored. This allows the operator to monitor and assess if the distribution of these measures is moving or changing in an undesirable way. It might be that the mean or median is moving towards a less safe condition, or the level of dispersion in an operational measure is increasing, when the intention is to standardise.

Measurements from every flight make quantifying operational risk a real possibility, assuming the measures map onto operational risks. For example, unstable approaches are a causal factor in runway veer-off accidents (Post, 2015) and *approach speed* is an indicator of unstable approaches. By taking measures of approach speed from every flight, the likelihood of excessive speed can be determined, which in turn helps quantify the operational risk of runway veer-off. It could be argued that exceedance detection could serve the same function, however only data exceeding a threshold is retained, and that threshold cannot be changed without reprocessing all the data. Also, knowing the full empirical distribution of a measure allows that measure to be modelled and used in risk analysis tools, such as Bayesian networks, even if the empirical data is sparse e.g. where extreme values are possible.

2.1.3.2.3 *Safety trends*

Exceedance detection can help identify safety occurrences that have otherwise been unreported and undiscovered, however the main use of this data, and measurement analysis data, is to develop long-term safety trend information and safety performance indicators. These trends are often presented as exceedance event rates over time, usually by a monthly period, and are presented at company safety meetings such as safety action groups (SAG) and safety review boards (SRB). A typical set of trend data presented at these meetings might consist of the following:

- Overall FDM exceedance event rate over time
- Individual exceedance event rates over time
- Top 10 event rate lists

- Airport rankings of where most exceedance events are occurring.

Some trends are used as safety performance indicators and can be loosely aligned with operational risks, with some interpretation, and provide an assessment of safety performance. However, it can be difficult to definitively conclude that the trend relates to a specific risk, such as loss of control in flight or runway veer-off.

Some operators try to link trends to operational safety issues (Aghdassi, 2018), while others are actively trying or aspiring to map FDM trends onto operational risks (Adikari, 2016). EasyJet has mapped FDM events onto operational risks and uses a form of statistical process control to produce alerts when a trend has breached a limit e.g. one standard deviation from the previous twelve-month average (Henchie, 2013).

The ARMS methodology (see 2.2.1.3) is in use in some airlines to produce subjective “event risk classification” scores for FDM events and then trend these scores over time, a method which some might claim is quantifying risk. However scores like these, ultimately based on subjective judgement, can be weak and inconsistent (Hubbard, 2012).

2.1.4 Research based on flight data and FDM

Considering the length of time flight data and flight data monitoring has been around, there is surprisingly little published research on the topic. This is perhaps a symptom of the data being regarded as sensitive and most FDM programmes have restrictions on the use and divulgence of FDM data. Prior to FDM becoming a mandatory practice, airlines would have to negotiate the introduction of FDM with pilot union groups who were afraid that the data could be misused. Usually the negotiations resulted in quite strict protocols allowing only a limited number of “approved” individuals to have full access to identifiable² data. Despite FDM becoming a mandatory practice in 2005, this practice persists, as reflected in the CAP 739 guidance (Civil Aviation Authority, 2013), making it difficult for researchers to get access to data. Hopefully this will change over time and the huge amounts of data can be made more available, allowing deeper mining of the safety information within.

Fortunately, some researchers have been able to acquire data and use it to find new methods for gaining safety insights. Several studies have focussed on addressing an inherent weakness in FDM; the reliance on predefined events and measures to detect anomalies. FDM has traditionally relied on exceedance events being defined in response to safety incidents or anticipated issues and this has left gaps in the oversight it provides. A good example is the

² i.e. where flight number and flight date are visible, providing the possibility of pilot identification

incident which occurred during a landing at Gibraltar in 2002 (Air Accidents Investigation Branch, 2014). Gibraltar can be a fairly challenging airport to operate into, with a relatively short runway and aircraft can experience turbulence off the nearby terrain during final approach. On the 22nd May 2002 a Boeing 757 landed at Gibraltar and before the nosewheel touched down, the commander applied full nose-down elevator resulting in an excessive de-rotation. This caused the nosewheel to impact the ground hard and the aircraft was significantly damaged. The subsequent investigation found, through using historical flight data, that the commander had developed a habit of applying nosewheel down input during landings, against the advice of the aircraft type's training manual. During the incident landing, the commander applied the input earlier than usual, resulting in the hard impact and damage. Had there been a FDM exceedance event in place to monitor excessive de-rotation, the operator may have been aware of the developing technique and could have intervened. Soon after publication of the report, FDM software and service providers implemented de-rotation at landing events into their systems.

The example shows how reliant FDM is on previous incidents; where a previous incident has not occurred, or no one has had the foresight to envisage a type of incident, it is unlikely that a FDM event will exist. This has led researchers to explore ways to avoid the predefinition of events and find ways to automatically detect anomalies in data.

Mugtussidis (2000) ran into the flight data access problem, having just four flights available. However, knowing the limitations of flight data, Mugtussidis proposed a sensible approach to flight data anomaly detection by limiting the analysis to twenty or so features rather than every parameter recorded. A Bayesian classifier was then proposed, which would aim to identify which flight a particular data point originated. If this was possible, it would be concluded that the data point was unusual. Unfortunately the lack of data available severely restricted opportunity to validate the approach.

Smart (2011) worked with a FDM service provider to gain access to a large dataset. Smart compared the performance of three one-class classifiers: Support Vector Machine (SVM), Mixture of Gaussian, and K-means. It was found that the SVM performed best in classifying approaches to landing as abnormal, and was also able to quantify the abnormality of the approach. Smart found that the SVM outperformed traditional event-based FDM. In common with Mugtussidis, Smart reduced the dimensionality of the flight data by selecting a limited set of features.

Li et al. (2011) used principle component analysis to reduce the dimensionality of data from 365 Boeing B777 flights, before using a density-based clustering algorithm to identify anomalous flights. The algorithm, called DBSCAN, does not require an estimate of the number of

clusters present in advance and is not sensitive to the shape of clusters. The results were promising, with abnormalities in flights identified such as low altitude on approach with high power, late alignment with runway, speed high. It is likely that traditional FDM exceedance monitoring would have also detected these issues, however the strength of this approach is that it did not require specific algorithms to be predefined for each anomaly found.

Clustering methods in flight data anomaly detection were further developed by Das et al. (2012) and the closely related work of Li et al. (2015), who developed an algorithm called cluster-based anomaly detection (ClusterAD) and compared it alongside multiple kernel anomaly detection (MKAD) and traditional FDM exceedance detection. The researchers had access to a good-sized dataset consisting over 25,000 flights and took a subset of available recorded parameters for the period of 6NM on the approach to touchdown for comparison. The researchers found overlap in the anomalies detected by the three methods and each method identified anomalies which the others did not. This suggests that the clustering methods should be used alongside exceedance detection rather than supersede it. An interesting difference in the approaches of exceedance detection and the clustering methods was highlighted by Das et al. Neither of the clustering methods detected short-term transgressions or deviations, whereas the exceedance detection did. Exceedance detection algorithms often have simple logic: if airspeed is greater than X, and altitude less than Y, for duration Z, then event is true. If this deviation is corrected, an exceedance event is still triggered, even if the remainder of the flight is normal. Clustering methods are able to find longer-term anomalies, but because, for short-term deviations, the amount of normal values outweighs the number of anomalous values, no anomaly is found. This again highlights how clustering could be used to complement, rather than replace, exceedance detection.

The work of Li et al. (2015) was further developed in Li et al. (2016), where Gaussian Mixture Model based clustering was used to detect anomalous flights. Gaussian Mixture Models (GMM) are often used to represent subpopulations within a larger population and therefore can be applied to aircraft flight data. Within a number of flights, there will be variations in how the aircraft is flown, depending on the conditions present. For example, an approach to landing will be flown differently in marginal visibility conditions (i.e. a precision approach), compared to one flown in good visibility (i.e. a visual approach). The approaches will differ in terms of timings of configuration changes, heading relative to runway, height relative to runway and so on. A GMM can represent these differences as different types of approaches within the overall population of flights. This allows the potential to identify anomalous subpopulations and anomalous flights that do not belong to any population. Li et al. transformed the flight data into a number of vectors, each relating to a flight parameter, of equal length and normalised the

values. Imagining this as a large table with each column representing a specific flight parameter from a specific flight, each row will represent the population of all parameters from all flights at a particular time or distance from, say touchdown. This row vector can then be used in cluster analysis. Li et al. carried out the analysis on 10,528 Airbus A320 flights and used four domain subject experts to validate the results. They concluded that the approach was able to identify flights with elevated risk and outperform existing methods for detecting “known unsafe events”. The authors acknowledge that further work is required to determine performance for detecting unknown issues.

Other research focussing on anomaly detection has included Gorinevsky, Matthews and Martin (2012), Mendes (2012) and Nanduri and Sherry (2016), all of whom took different approaches. Gorinevsky, Matthews and Martin (2012) used the residuals from a regression model to feed a multivariate statistical process control in order to identify anomalies. The regression model was trained using flight performance coefficients derived from FDM data. Care had to be taken so as not to include anomalies in the model, as would happen if the entire dataset was used for training. However, if only a small subset of the dataset was used for training, the normal variability of day-to-day aircraft operations would lead to spurious anomalies. A novel three-level regression approach was taken to address those issues. Also worth noting from Gorinevsky, Matthews and Martin is the highly efficient data processing they were able to achieve. Their method scanned for anomalies in 500,000 flights in just 10 hours. The anomalies found were generally related to individual flight data parameters, such as discrepancies in control surface position and sensor problems, rather than operational safety issues. Nonetheless, the anomalies detected could potentially result in safety issues and are therefore valuable knowledge.

Similarly to Smart (2011), Mendes (2012) used support vector machines (SVM) to highlight anomalies in aircraft autoland performance using recorded flight data. Principle component analysis was used on a training set of known good autolands, and then a SVM used to classify the performance of the autoland. Mendes’ approach was able to filter 111 abnormal autolands from a set of 629, thus greatly reducing the manual effort required to investigate autoland performance.

The anomaly detection methods described so far have relied on some form of dimension reduction to make the method viable, however Nanduri and Sherry (2016) avoid this problem by using recurrent neural networks (RNN).

Standard artificial neural networks are inspired by biological nervous systems, with nodes (neurons) passing signals over connections. Each neuron can process the information sent to it, and provide an output based on the input. Typically, there are multiple layers of neurons, with

the input and output layers being separated by hidden layers. Artificial neural networks are very good at machine learning tasks and are widely used in fields such as image recognition and medical diagnosis. Their ability to “learn” without prior knowledge makes them extremely useful and they can be highly accurate, however the resulting classification decisions can be difficult, if not impossible, to trace back due to the opacity of the network. This is a problem for applications such as credit applications, where decisions can be appealed, however for classification of normal vs anomalous in the safety domain, it is usually not a problem, as the classifier is often just a means of directing an analyst to investigate potentially anomalous data.

Recurrent neural networks, as used by Nanduri and Sherry, allow the output from hidden layer neurons to be fed back as inputs to other neurons, thus allowing the use of past history, which is desirable when using time series data such as recorded flight data. They modified the typical RNN architecture by making use of long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014). Nanduri and Sherry claim that recurrent neural networks using LSTM and GRU do not share the limitations of MKAD or ClusterAD as described in Li et al. (2015) because “...*they [RNN] are capable of handling multivariate sequential input data without any modifications and treat it as time series data*”, hence potentially ideal for recorded flight data. Unfortunately, despite the existence in the United States of the Distributed FOQA Archive (Chidester, 2007) containing some 2 million flights, restrictions on the archive’s use prevented the researchers using it, so they had to simulate data for 500 flights (Nanduri and Sherry, 2016b).

Anomalies were introduced into the simulated data and the anomaly detection performance of MKAD, LSTM and GRU were compared, finding that LSTM and GRU RNNs significantly outperformed MKAD.

While much of the literature on the use of FDM data focusses on anomaly detection, some researchers have used it to address specific operational problems. Haverdings and Chan (2009) used quick access recorder (QAR) data to study low-level windshear and turbulence, both significant aviation hazards, for aircraft operating at Hong Kong International Airport. Wang, Ren and Wu (2018) also made use of QAR data to look at how pilots’ landing techniques effected landing safety. They used an analysis of variance on 293 flights to determine the characteristics of landings, and then two regression models to examine correlations between pilot operations and landing outcome. Unsurprisingly it was concluded that pilot inputs did effect touchdown distance and vertical acceleration at touchdown, but possibly of more value were their findings around pitch and thrust control technique i.e. a positive and steady pitch up input, coupled with a relative slow reduction in thrust lead to better landing outcomes.

Finding new methods to help detect anomalies is valuable research because it helps identify hazards which may otherwise remain hidden and unknown. Likewise for other research looking at specific operational problems. Hazard identification is central to safety management, and even with the millions of hours of operational experience accumulated by the World's airlines, new hazards are still being identified thanks to anomaly detection and other methods. However, the hazards causing aircraft accidents today are usually well known and documented by individual airlines and the industry in general. At an industry level, the risk these hazards represent is quite easy to quantify by looking at accident statistics and operational data, but at an individual airline level this risk quantification is much more difficult because in most cases, the airline has not experienced a loss. The airline, therefore, must use accident precursor (i.e. causal factor) data to try to quantify the level of risk present in the operation and it would seem that FDM data would be an obvious source. Indeed, FDM is very good at quantifying the prevalence of those causal factors or precursors in the operation, but how is that prevalence translated into a risk? How does landing fast with a crosswind effect the risk of a landing veer-off for example? At which airports do those risk factors combine to regularly increase the risk of a runway veer-off? As it stands, FDM is not being used to help provide this risk quantification and there is sparse to non-existent literature on the topic. Fortunately though, there is literature on methods that could potentially make use of FDM data.

2.2 Risk methods

Fruhen et al (2013) in their paper titled “Chronic unease for safety managers: a conceptualisation” neatly summarised five attributes of the safety manager:

- Pessimism
- Propensity to worry
- Vigilance
- Requisite imagination
- Flexible thinking

These attributes resonated with the author, who spent several years working as a safety manager in an airline. In particular, “Propensity to worry” sounded very familiar. Safety managers have two major things to worry about. The first is whether they have robust processes in place for identifying hazards. Are there hazards lurking that haven’t been identified? Has every stone been turned over? Has the wealth of data available through flight data monitoring (FDM), safety reporting and auditing been fully analysed to squeeze as much information from it as possible? The second worry is whether those hazards, which have been identified, have been subjected to a rigorous risk assessment process to ensure they are prioritised correctly and limited resources can be targeted most efficiently.

It was the latter that often worried the author most. Whilst hazard identification in airlines isn’t always perfect, the industry does have the advantage of generating large quantities of data through FDM and usually mature safety reporting programmes, so if these initiatives are in place, the safety manager can take some comfort. However, this wealth of data is not widely used to try to quantify risks in any way. The vast majority risk assessments are qualitative, perhaps unsurprisingly, given the regulatory framework of airline operational risk.

2.2.1 Risk assessment and risk management methods used in airline operations and maintenance

2.2.1.1 Risk matrices

The ICAO Safety Management Manual (International Civil Aviation Organisation, 2013) is the basis on which most safety management practices in airlines are founded. The guidance it contains is adopted by national regulators and used by inspectors to determine the quality and efficacy of an airline’s safety management system (SMS).

LIKELIHOOD	MEANING	VALUE
FREQUENT	Likely to occur many times (has occurred frequently)	5
OCCASIONAL	Likely to occur sometimes (has occurred infrequently)	4
REMOTE	Unlikely to occur, but possible (has occurred rarely)	3
IMPROBABLE	Very unlikely to occur (not known to have occurred)	2
EXTREMELY IMPROBABLE	Almost inconceivable that the event will occur	1

Table 1. Safety risk probability table from ICAO SMM 3rd Edition

Estimating risk probability is fundamental to risk assessment and Table 1 shows the ICAO guidance for assessing risk probabilities. Note the qualitative definitions for “Likelihood”. These definitions are used by safety analysts in airlines to rank risks and thereby prioritise resources towards mitigating those risks. In a large airline, the analyst will have the advantage of having plenty of historical data to call upon while reviewing a safety incident and trying to determine the risk likelihood, but problems arise due to the highly subjective nature of the likelihood meanings. Where, for example, can the line be drawn between “has occurred infrequently” and “has occurred rarely”? Hubbard (2009) explains how bias and overconfidence can affect how risks are rated using terms like these, leading to poor risk assessment.

Another problem in the large airline scenario, is that it is necessary to have several analysts performing the categorisation and it can be difficult to maintain consistency between individuals. This can be helped to some extent by holding regular calibration meetings between the analysts, where they will discuss incidents and risks and agree how they should be categorised. Ideally this would be the process for every risk identified, however it would just not be practicable given the amount of data passing through a typical large airline safety department.

The problem is worse for smaller airlines who do not have vast amounts of historical data. Compare a large low-cost airline with say 400 aircraft, and a smaller airline operating 5 aircraft. It would take the smaller airline 80 years to gain the operational experience and generate the safety data the larger airline achieved in one year. The result is that for a given risk, the analyst in the small airline might choose “not known to have occurred” whereas the analyst in the large airline might choose “has occurred frequently”.

LIKELIHOOD	MEANING	QUANTITATIVE DEFINITION
FREQUENT	May occur once or several times during operational life.	1 to 10^{-3} per flight hour
REASONABLY PROBABLE	May occur once during total operational life of one system.	10^{-3} to 10^{-5} per flight hour
REMOTE	Unlikely to occur during the total operational life of each system but may occur several times when considering several systems of the same type.	10^{-5} to 10^{-7} per flight hour
EXTREMELY REMOTE	Unlikely to occur when considering several systems of the same type, but nevertheless has to be considered as being possible.	10^{-7} to 10^{-9} per flight hour
EXTREMELY IMPROBABLE	Should virtually never occur in the whole fleet life.	$< 10^{-9}$ per flight hour

Table 2. Safety risk probability table from ICAO SMM 1st Edition

The first edition of the Safety Management Manual (International Civil Aviation Organisation, 2006) did refer to quantifying risk probabilities in the table shown in Table 2. Of course, it would be impossible for ICAO to provide similar quantitative definitions for all sizes of airline operation, however at least it gave operators some idea of how to quantify likelihood and even perhaps produce their own definitions.

Note in Table 1 that a “Value” is given. This value is combined, through a risk matrix (Figure 2), with another value for the potential severity of an undesirable outcome, to provide a risk index. This may lead some safety managers to believe they are actually quantifying risk, because often a numerical value is the result. However, as the index is based solely on qualitative assessments, it is impossible to produce a truly quantitative result.

Risk probability		Risk severity				
		Catastrophic A	Hazardous B	Major C	Minor D	Negligible E
Frequent	5	5A	5B	5C	5D	5E
Occasional	4	4A	4B	4C	4D	4E
Remote	3	3A	3B	3C	3D	3E
Improbable	2	2A	2B	2C	2D	2E
Extremely improbable	1	1A	1B	1C	1D	1E

Figure 2: Typical 5x5 risk matrix. Source: ICAO Safety Management Manual, 3rd Edition

Hubbard (2009) asserts that this type of risk assessment could actually be worse than doing no risk assessment at all, because it can give management undue reassurance that something robust is being done. Anthony Cox (2008) provided a thorough examination of the weaknesses of risk matrices and the resulting risk assessments and cautions that “*Risk matrices do not necessarily support good (e.g. better than random) risk management decisions...*”.

2.2.1.2 Bow tie diagrams

Bow tie diagrams are a useful way to communicate risk and they have been adopted by some airlines as part of their risk management processes. An example is shown in Figure 3 below.

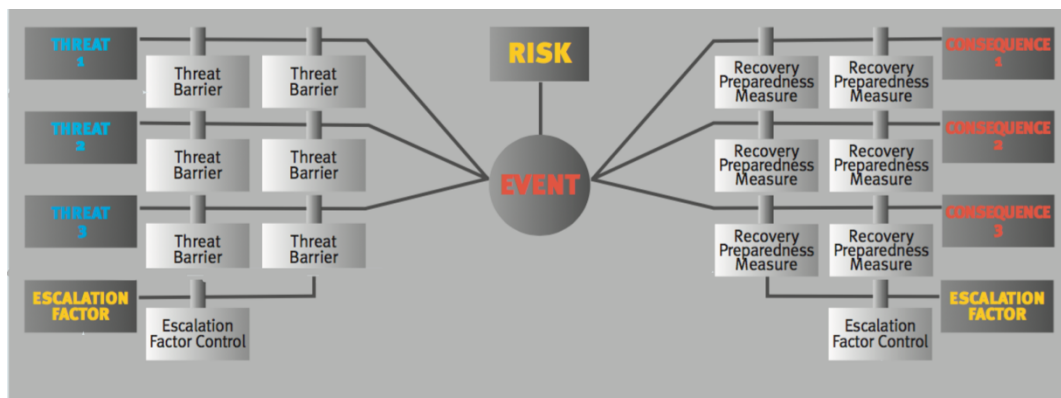


Figure 3. The bow tie (from Strategic Risk, November 2005)

The concept is fairly simple and follows the principles of layered security made popular by the “Swiss-cheese” (Reason, 1997) accident causation model. At the centre of the bow tie is a *hazard* (labelled as a risk³ in Figure 3) connected to an *event*. This event is the undesirable situation

³ There is some variation in terminology used in literature related to bow ties, however the concept remains the same.

occurring due to the release of the hazard. Note that it is usual for the event to be a situation which is recoverable i.e. it is an event that has the potential to cause loss or disaster, but no loss has actually occurred at that stage.

On the far left of the bow tie are *threats* which can cause the hazard to be released into an event. To the right of the threats are *barriers* or *defences* to stop the threat releasing the hazard. Similarly to the Swiss-cheese model, these barriers may have different degrees of effectiveness. To the far right are *consequences*; these are the potential outcomes if the undesired event develops, and represent some form of loss (e.g. injury, equipment damage, reputation damage). To the left of the consequences are recovery measures i.e. further barriers or controls to try to prevent or minimise the undesirable consequences, given that the event has occurred. An example might be a runway overrun area to minimise aircraft damage in the event of an excursion, or an emergency response plan.

The barriers on both sides of the bow tie could be degraded in certain circumstances (e.g. a power-cut might render a warning system useless) and these are called *escalation factors*.

A simple example to help explain bow tie terminology is an aeroplane flying into an airport surrounded by terrain. A *hazard* is the terrain and the undesirable *event* could be unintended close proximity to the terrain. *Threats* which could lead to the unintended proximity could be poor weather with reduced visibility or poor vectoring by air traffic control. *Barriers* against these threats could be a procedure not to descend below a safe altitude or the prohibition of approaches to the airport if the weather is not good. An *escalation factor* might be a lack of pilot awareness in these policies and procedures. The potential *consequences* are quite obvious; controlled flight into terrain, loss of life, aircraft destroyed. An example of a *recovery measure* in this case would be a ground proximity warning system, alerting the pilots to their proximity with the terrain.

The UK CAA has produced a series of bow tie diagrams based on their “Significant Seven” risks (see CAA n.d.) and they are now widely used in airlines to help manage risk. The diagrams are an excellent way of communicating how risks are being managed and therefore are a valuable tool for demonstrating risk management to a regulator, for example, during an audit. Bow ties can be thought of as a Fault Tree and Event Tree (see below), coupled together to represent the left and right-hand sides respectively. As such, it is possible to use quantitative information to produce bow ties, however those produced in airline operational environments are usually

qualitative with little, if any, information relating to probabilities, even where data exists (e.g. FDM) that could provide this information.

2.2.1.3 ARMS methodology

The Airline Risk Management Solutions (ARMS) working group has produced a methodology (Aviation Risk Management Solutions, 2010) which, like Bow ties, is based on barrier principles of safety management. The methodology comprises three main elements. Firstly, new individual safety events are classified – the “Event Risk Classification” (ERC). This is “...*a quick initial estimate on the risk inherent in the event*” which results in a subjective numerical value being assigned to the risk “...*which can be used in quantitative risk analysis*”. It could be argued that the use of “quantitative” here is misleading; a numerical value is used, but it could just as well be a character, A, B or C. The values are later summed to produce a cumulative ERC score, but this could as easily be counts of A, B or C, hence it is not strictly *quantitative* as it depends on an initial qualitative assessment.

The second element is the analysis of a safety event database, comprising ERC-scored events, to produce safety trends and identify Safety Issues.

The third element is the “Safety Issue Risk Assessment” (SIRA). The Safety Issues identified are risk assessed and assigned a risk value. The intention then is that the risk assessment then leads to safety improvement actions. Safety Assessments can also be carried out using similar methods in advance of conducting a new activity.

The SIRA risk assessments are “*calculated*” using the following factors, quoted from Aviation Risk Management Solutions (2010):

- *Frequency/probability of the so-called Triggering Event*
- *Effectiveness of the Avoidance Barriers*
- *Effectiveness of the Recovery Barriers*
- *Severity of the (most probable) accident outcome*

The methodology is in use at several large airlines (e.g. easyJet) who were instrumental in its development and one of the objectives was to keep it “...*easy to use and not create an unreasonable workload*”, which it admirably achieves. However, while the attempt to quantify barrier effectiveness and the probabilities of “*triggering events*” is to be welcomed, the method still relies on some subjective assessment and the language used may mislead the user into believing the method is more quantitative in nature than it really is. Also, while the method has been broadly accepted by the operational community, literature and research on its effectiveness is sparse to non-existent.

Other areas of the aviation industry are well used to quantifying risk. In design and airworthiness, it is common to use techniques that do not rely solely on qualitative analysis. Three of these techniques are briefly described below.

2.2.1.4 Failure mode and effects analysis (FMEA)

FMEA is a tool to analyse failures in hardware or processes and examine their potential effects. Each component or process may have multiple failure modes and associated effects, and these are risk assessed in terms of probability, severity and opportunity for detection. Wherever possible, data is used to quantify these items, so for example, reliability data may be used for a component failure and historical data for how often or how quickly a human operator might detect the failure. The basic FMEA process (Mikulak, McDermott and Beauregard, 2008) is:

1. Review the process
2. Brainstorm potential failure modes
3. List potential effects of each failure mode
4. Assign severity rating for each effect
5. Assign a probability rating for each failure mode
6. Assign a detection rating for each failure mode and/or effect
7. Calculate a risk priority number for each effect. This is the product of the ratings in 4, 5 and 6, so if each of those was rated on a scale of 1 to 10, the risk priority number would be between 1 and 1,000.
8. Prioritise the failure modes for action
9. Mitigate or eliminate the high-risk failure modes
10. Calculate the residual risk priority number once elimination or mitigation has been carried out

[Emphasis added]

Note that 4, 5 and 6 may be based on real-world data where available, hence an FMEA analysis can quantify risk and go beyond a purely qualitative analysis. However, whilst FMEA has found uses outside of strictly engineering problems, such as medicine (Institute for Safe Medication Practices, 2001; Senders, 2004) and project management (Carbone and Tippet, 2004), it still is not widely used in operational risk management domains. This is possibly because FMEA is ideally conducted at the design stage (Mikulak, McDermott and Beauregard, 2008), or for existing processes. This is echoed by Palady (1998) who states that FMEA is a tool for developing new designs and processes.

FMEA is not widely used routinely in airline operations and maintenance, however Hu and Hsiao (2016) did make use of the method to risk assess quality standards in airline service and Barkat, Karan and Pradeep (2017) proposed an on-board automated FMEA to diagnose defects in real-time to suggest actions to airline maintenance personnel.

2.2.1.5 Fault tree analysis (FTA)

FTA is a safety engineering tool and form of probabilistic risk assessment, which uses Boolean logic gates to analyse potential failures. Whereas a FMEA takes a failure as a starting point for the analysis, FTA takes an undesired outcome, sometimes called a “top-event”, as the starting point. FTA then works backwards from the outcome to identify failures which could lead to the outcome. These failures are usually mapped out with logic gates. For (a very simple) example, a top-event might be burnt toast. This could be caused by the operator of the toaster using a too high setting and a probability could be assigned to this. It could also be caused by the operator using a too low setting, putting the toast back down, AND then becoming distracted. Probabilities for this combination could also be estimated. Burnt toast could be caused by the first OR second scenario.

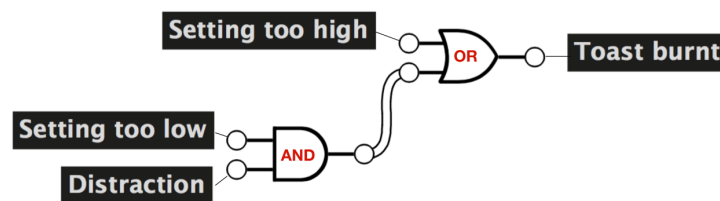


Figure 4. Simple FTA example

According to Stamatelatos and Caraballo (2002), the fault tree is a “....qualitative model that can be evaluated quantitatively and often is”. By assigning probabilities, either real or estimated, to each of the failures which contribute to the top-event, an overall probability and most likely pathway to the top-event can be determined. Fault tree analysis is a powerful methodology, so much so that organisations such as NASA encourage managers and support staffs to be familiar with its application (Stamatelatos et al., 2002).

FTA is seldom routinely used by airline safety departments, however some studies have found it to have potential in both operations (Lu, Wetmore and Smith, 2005) and in maintenance (Chen, Ren and Bil, 2014).

2.2.1.6 Event Tree Analysis (ETA)

ETA is similar to FTA in that it considers an initiating or “top” event, but instead of working towards failures that could contribute to the initiating event, ETA works towards potential consequences of the initiating event. Figure 5 shows a simple example:

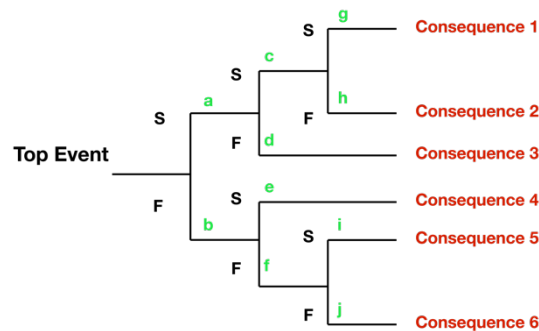


Figure 5. Simple event tree

In the diagram, “S” denotes success and “F” failure. Each node can be thought of as a barrier or mitigation to prevent the initiating event from propagating towards an undesirable consequence. Probabilities can be attached to the success or failure at each node, either through estimation or by observations, and a resulting probability calculated. For example, the probability of Consequence 1 occurring would be:

$$P(C_1) = P(a) * P(c) * P(g)$$

Likewise for Consequence 3:

$$P(C_3) = P(a) * P(d)$$

A bow tie diagram is essentially a fault tree on the left-hand side, and an event tree on the right-hand side, so it could be argued that FTA and ETA are in routine use in airlines. However ETA is rarely used in its own right in an operations context, but it has been shown to be a useful tool in modelling the operational consequences of aircraft failures (Ahmadi and Soderholm, 2008).

2.2.1.7 Limitations

The analysis techniques briefly described above are useful tools for considering accident pathways and assessing risk, however they do have some limitations. Nivolianitou et al (2004), in their paper on accident analysis, found that ETA and FTA were both “average” in coping with dependencies between events. This is echoed by Khakzad et al (2011) who state that Fault Trees assume events are independent, which is not always the case. Another limitation is that both ETA and FTA deal with binary outcomes and cannot easily cope with multi-state variable inputs, which are often necessary in risk analysis.

Further to the limitations above, none of the analysis techniques mentioned so far can efficiently deal with uncertainty (Khakzad et al., 2011). Quite often in risk analysis, data is not available for some events of the scenarios being considered. For example, when analysing the risk of a runway excursion, an airline may have ample data from FDM or safety reporting to help establish probabilities of high speed or deep landings, however other factors which are perhaps not recorded by FDM nor reported, may need to be estimated by subject matter experts. Ideally these estimates should be updated when data becomes available, so that the risk assessment is dynamic. Bayesian networks can help overcome these limitations and are growing in popularity in risk analysis (Weber et al., 2012).

2.2.2 Risk quantification methods in aviation

The methods described in 2.2.1 are in fairly common use in airline operations and airworthiness, however there is also a body of academic literature relating to risk quantification applied to specific aviation problems. Many of these relate to models aimed at addressing risk scenarios or accident causes and employ various methodologies to attempt to quantify risk.

Ayra et al. (2015) used quantitative methods to address the specific problem in airline operations of unintentional emergency slide deployment. When an aircraft departs and starts to taxi, the cabin crew arm the escape slides so that they deploy automatically when a door is opened. On arrival, the slides must be disarmed to ensure they do not inflate when the doors are opened. Slides deploying along jetty walkways or dropping onto ramp personnel can obviously result in serious injury. Ayra used a logistic regression with Bayesian methods to analyse the risk of inadvertent slide deployment and forecast the associated financial costs. By taking this quantitative and objective approach, safety managers were able to persuade key stakeholders of the need to adapt standard operating procedures, ultimately resulting in a significant reduction in the number of inadvertent slide deployments. It is possible that purely qualitative analysis would not have been as persuasive.

Quantitative methods have also been employed for higher-consequence risks, such as mid-air collision. Mori (2014) modelled empirical data (an approach similar to that taken later in this research) using a Laplace distribution to refine a collision risk model for aircraft on oceanic flight plans. Mori's quantitative treatment of the risk suggested that the existing model was overly conservative and savings could be made by adopting a refined model. Also modelling empirical data, but this time from experiments in flight simulators, Knecht (1997) developed a risk metric for mid-air collision risk, based on time-to-contact. In this case, least-squares regression and the Levenberg-Marquardt algorithm were used for data fitting.

Logistic regression was again central to an interesting study using incident data and pilots' age and experience to produce a model (McFadden, 2003) to compare, between airlines, the probability of pilot-error accidents. McFadden notes however, that logistic regression can be problematic when there are frequent non-events in the data. This is especially problematic when modelling data in safety-critical industries, such as aviation, where accidents are thankfully relatively rare.

Shyur et al. (2012) use a different regression method for a very similar problem. In order to estimate the probability of safety events, Shyur et al. again used safety report data, but coupled with regulatory oversight data (e.g. audit findings) as inputs into a proportional hazards model. A proportional hazards model (Cox, 1972) is a type of survival model, where multiple variables, both quantitative and categorical, can be related to the time until an event occurs. It is usually applied to a medical setting, where the effect of several variables (e.g. patient age, sex, smoker?) are assessed to provide a probability of an outcome event (e.g. cancer, heart disease), however it is easy to see the potential of transferring it to an aviation risk setting, as Shyur et al. did.

Flight Operations Risk Assessment System (FORAS)(Hadjimichael and McCarthy, 2004) is another methodology which attempts to quantify aviation risk. FORAS is a hierarchical risk model, where risks are broken down into constituent parts and represented as a series of connected parent and child nodes. Each parent has no more than three children in order to minimise complexity. Hadjimichael (2009) provides an example of "*Work amount risk*" which is decomposed into two factors, "*Number of sectors in previous duty period*" and "*Duty hours*". Categories (e.g. low, medium, high) for each of the factors are then elicited from subject matter experts (SME) to derive an overall "quantitative" score for each risk through fuzzy membership functions. These scores are then combined to produce a relative risk index for a flight; the goal being to provide a pre-flight indication of flights which could represent a higher risk.

The system has two significant strengths. The first is that it formally captures the knowledge available in the organisation from SMEs on safety risks and allows that information to be re-used as necessary. The second is that it provides risk information in a dynamically preventive way by providing a pre-flight risk index. The authors are careful to point out that it is not intended to be a "go/no-go" tool, but providing pilots with guidance that the flight they are about to operate has an elevated risk, could potentially allow mitigations to be enforced by being more prepared (e.g. through more comprehensive briefings, taking more fuel etc). This is an appealing feature and is developed later in this research.

However, whilst it is claimed that FORAS is quantitative, the authors also state that it is non-probabilistic; it could be argued that any system which relies solely on the subjective opinion of SMEs to provide "quantitative" scores, is more qualitative rather than quantitative in nature,

like the ARMS methodology described earlier. Another issue is that the only validation provided in the literature, is one based again on SME opinion, however the authors do propose that FDM results could be used to validate the method, which would “ideally” correlate with FORAS results.

Roelen et al. (2011) highlights three initiatives to develop quantitative risk models which use multiple methods. Roelen argues that the necessity for multiple methods is born out of the recognition that the ubiquitous linear models (Reason, 1997) do not reflect the complexity of the aviation system (Dekker, 2004; Leveson, 2004). Three initiatives highlighted by Roelen are:

- Federal Aviation Authority (FAA) Hybrid Causal Logic (HCL) model
- Eurocontrol Integrated Risk Picture (IRP)
- Netherlands Ministry of Transport Causal modelling of Air Transportation System (CATS)

The three models have similarities in that they use combinations of fault trees and event trees to model the more certain aspects of the risk, and then use influence diagrams and Bayesian networks (see 2.2.3 Bayes’ theorem and Bayesian networks) to model less certain factors, such as those relating to humans and organisations. As will be discussed later, a strength of Bayesian networks is their ability to model uncertainty.

Roelen points out that to be successful, these models need good exposure data to feed them and states:

“In-flight recorded data is the most accurate and detailed data, suited to quantify large pieces of the risk model.”

In addition to being used to identify risk control measures, the CATS model has also been used during post-accident analysis (Ale et al., 2010) to draw conclusions from accidents which traditional investigation methods may not have identified. By using historical data, and data gathered from the accident investigation, parameters from the model can be adjusted and ultimately the probability of an accident calculated under different scenarios. It is easy to see how this would be a powerful tool for looking at *what-ifs* and identify the relative strength of different risk mitigations.

The use of a hybrid model using BN, ETA and FTA is possibly an overcomplication as both ETA and FTA can be represented, arguable more coherently, by a BN (Fenton and Neil, 2013).

2.2.3 Bayes' theorem and Bayesian networks

Bayesian statistics, named after the English statistician Thomas Bayes (1701-1761), is a branch in the field of statistics which is growing in popularity. Central to Bayesian statistics is Bayes' theorem which is a relatively simple statement describing the probability of an event. The rise in popularity of Bayesian statistics is in part due to technological advances in computing which has allowed complex probability calculations to be made on everyday computers, however it is also due in part to the intuitiveness of the theorem. The theorem is essentially an expression of updating our belief in light of new evidence, which is something humans do intuitively. The theorem is expressed as:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \quad (1)$$

If we say that $P(A)$ is the probability a runway veer-off, and $P(B)$ is the probability of a causal factor, the equation is simply stating:

The probability of a runway veer-off, given the causal factor, is equal to the probability of the causal factor, given a veer-off has occurred, multiplied by the probability of a veer-off, divided by the probability of the causal factor occurring.

Bayesian networks (BN) are directed acyclic graphs which use probabilistic inference, including Bayes' theorem, to update probabilities in light of available evidence. They often have a relatively simple causal structure which can be readily understood, despite the underlying mathematics being quite complex for the non-specialist.

A BN consists of nodes connected by edges (also called arcs) which represent conditional dependencies between nodes and a simple example is shown below in Figure 6.

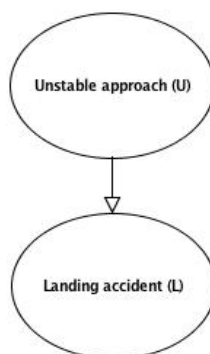


Figure 6: Simple Bayesian network

This simple example depicts a causal BN for a landing accident (L), conditional on the causal factor unstable approach (U). L is often referred to as a *child* or *leaf* node, and U as a *parent* or *root* node.

Every node in a BN has probabilities associated with it and these are shown in node probability tables (NPT). For this example, let's say that the probability of an unstable approach is 1 in a 100 ($P(U) = 0.01$). The NPT for $P(U)$ is shown simply as:

Unstable approach		Yes	0.01
		No	0.99

Table 3: NPT for $P(U)$

Let's assume that from landing accident statistics we know that the probability of a landing accident occurring given an unstable approach is 2%. We can now say that the probability of a landing accident, given an unstable approach has occurred, is 0.02 or $P(L(true) | U(true)) = 0.02$. Let's also say that the chance of a landing accident without an unstable approach ($P(L(true) | U(false))$) is 1 in 10,000 or $1e-5$.

The NPT for landing accident (L) is therefore:

		Unstable approach	
		No	Yes
Landing accident	No	0.99999	0.98
	Yes	1e-5	0.02

Table 4: NPT for landing accident (L)

The probability for a landing accident ($P(L = \text{yes})$) can be found using the equation of marginalization (see Chapter 4 of Fenton and Neil, 2012):

$$P(L = \text{yes}) = P(L = \text{yes} | U = \text{yes}).P(U = \text{yes}) + P(L = \text{yes} | U = \text{no}).P(U = \text{no}) \quad (2)$$

Which gives:

$$P(L = \text{yes}) = 0.02 * 0.01 + 1e^{-5} * 0.99$$

$$P(L = \text{yes}) = 2.099e^{-4}$$

The BN in Figure 6 can now be shown with the probabilities as follows:

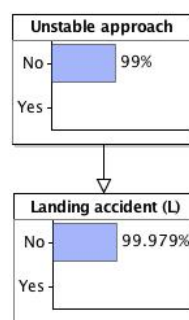


Figure 7: BN with probabilities

A quality of Bayesian networks which makes them so popular is the ability to run different scenarios and query the network. For example, let's say that a particular approach is definitely unstable i.e. $P(U) = 1$. After entering this evidence into the BN, the probabilities are updated as follows:

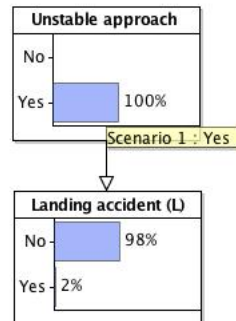


Figure 8: BN with updated probabilities

This is as expected because it is the probability we entered into Table 4, however the BN allows evidence to be entered anywhere in the network and the conditional probabilities for other nodes will be updated accordingly. For example, if we knew that a landing accident had occurred, $P(L = \text{yes})$ could be entered into the landing accident node and the BN would be updated with a new $P(U = \text{yes}/L = \text{yes})$ of 95.283%:

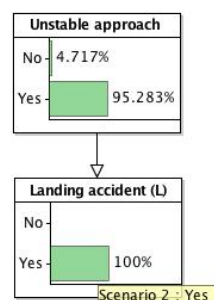


Figure 9: Updated BN

The example here is simplistic, however Bayesian networks can be much more complex and have many nodes, allowing complex risk scenarios to be modelled.

2.2.3.1 Bayesian networks in risk assessment

The flexibility and power of Bayesian networks in calculating probabilities when there is uncertainty and where data may be sparse, has made them a popular tool for conducting risk assessments across many diverse domains. In some domains they are quickly becoming the standard for risk assessment (Pasman and Rogers, 2015).

Traditional probabilistic risk assessments (PRA) are good at modelling failure scenarios in hardware systems where there is a binary output (e.g. fail, not fail), however they lack the flexibility of BNs in being able to deal with non-binary causal factors and the typical uncertainty which arises in risk assessments involving ageing of components, human error or

environmental conditions (Groth and Swiler, 2013). In their comprehensive work on BN and risk assessment, Fenton and Neil (2012) explain that to model operational risk it is necessary to consider not only “hard” systems (e.g. mechanical) but “soft” systems too, such as people, processes and culture. Bayesian networks allow the interface between these hard and soft systems to be modelled and allow the inevitably arising uncertainties to be taken into account. These properties make BNs very versatile and appropriate for modelling risks across domains as diverse as financial investments, law, and systems reliability in nuclear power stations (Fenton and Neil, 2013).

This flexibility and versatility is echoed by Siu and Kelly (1998) in their tutorial on Bayesian parameter estimation. Whilst not specifically dealing with Bayesian *networks*, the tutorial provides a good summary of Bayesian methods and how empirical data can be used alongside subjective sources in probabilistic risk assessments.

Bayesian networks have also been used to assess natural hazards risk (Straub, 2005). Straub also emphasises the flexibility of Bayesian networks in the example of rock-fall hazards, where the traditional method struggled to cope with conflicting evidence. Straub found that a Bayesian network could readily cope with these conflicts and also deal with an absence of data, when for example, some expected evidence had not yet been observed. The amalgamation of different types of evidence from varied sources (e.g. hard and soft) is a strength of Bayesian networks which consistently arises in the literature.

Trucco et al. (2008) employ a Bayesian network to model human and organisational risk factors in a maritime setting. The influence of various organisations on the future operation of a high-speed craft was modelled using a Bayesian network, the output of which was used as an input to a fault tree. This hybrid approach was used so that expert judgement could be captured by the network and fed into the more objective, data driven fault tree. The question arises as to why a hybrid was required at all; it would have been entirely possible to model the whole system using a Bayesian network with Boolean operators at each of the non-primary event nodes (Fenton and Neil, 2013). Fenton and Neil list several reasons why fault tree analysis should actually be performed by Bayesian networks, among them being the capability to expand on purely Boolean states and the ability to query the network in a diagnostic way. That is, if a failure is known to have occurred, that information can be entered into the failure node of the Bayesian network and the likelihoods of different failure modes is then traceable through the network. It is quite possible that the advantages of Bayesian networks may in time lead to a reduction in the use of traditional fault tree analysis.

2.2.3.2 Dynamic risk assessments and Bayesian networks

Imagine a scenario where two pilots arrive at crew dispatch ahead of their day of rostered flying. Amongst their usual briefing material of flight plans and weather charts there is a risk report highlighting that one of the airports they are due to fly to today has a predicted higher than normal risk for runway veer-off. The report states that due to forecast crosswinds and rain, coupled with the airline's FDM history of unstable approaches at the airport, the risk of a runway veer-off for this flight is predicted to be 350% above the baseline risk for the airline. The pilots can now use this information to make operational decisions about the flight. In addition to general heightened awareness, the pilots might decide that the more experienced of the two will fly that particular landing, they will configure the aircraft for landing earlier to avoid an unstable approach, and they will be careful to ensure they are aligned with the runway centreline at touchdown.

Such dynamic risk briefing material is seldom, if ever, available to pilots today. Risk assessments in airlines are usually static decision aids which are infrequently updated, but with the abundant data available in airlines today it should be possible to update the risk assessments frequently, or in other words, make them "living" risk assessments (Goble and Bier, 2013).

Bolsover (2015) advocates the use of Bayesian networks to provide real-time quantitative risk assessment, highlighting amongst several advantages, the speed at which reassessments can be made given new data. Bolsover concludes that Bayesian networks are capable of addressing the risks portrayed by bow tie diagrams, not merely parts of the bow tie, and this, coupled with the speed of reassessment, makes them well-suited to operational risk assessment.

Kanes et al. (2017) point out that while bow ties have improved the management of hazards, they "*...are static in nature and do not fully take into account the dynamic nature of risk*", a criticism they also level at quantitative risk methods such as event trees and fault trees.

This is a problem in the airline operations domain. Risk assessments are usually carried out prior to the start-up of a new route and then annually or bi-annually reviewed. The airline operations manual will contain details of the airports operated to, and a risk classification (e.g. category A, B or C). Category A are relatively benign airports, whereas restrictions may be in place for operation to Category B and C airports. Restrictions may take the form of only experienced pilots being allowed to fly approaches there, or there might be a requirement for the weather to be good, otherwise a diversion will be necessary. In the most severe Category C airports, pilots may be required to practice the approach in a simulator before operating to the airport, or there might be a stipulation that only training captains can fly the approach. Due to

the infrequent review schedule, these risk classifications may not always reflect the latest operational safety knowledge.

Kalantarnia, Khan and Hawboldt (2009) echo the concerns of Kanes et al. and highlight that during the operation of a process, near misses and incidents will occur. In an airline it would be expected that near-misses and other accident pre-cursors would be identified by FDM and safety reporting. This information should be fed into the airline's safety management system via safety meetings such as safety action groups (SAG) and safety review boards (SRB). However, it is the translation of that safety information into a quantitative risk assessment that is problematic. Rates and numbers of occurrences of incidents or FDM events at airfields is interesting information, but how does it map onto the risks which generate the airport classification? In other words, how can the decision makers at the SAG and SRB meetings determine if any of the information presented results in an airport having an increased risk of, for example, a runway veer-off? One of the aims of this research is to make that link clearer, traceable and more dynamic. The vision is to enable almost real-time updating of quantitative risk assessments allowing decision makers to know, using the latest operational data available, where the biggest risks lie.

Kalantarnia, Khan and Hawboldt (2010) used accident precursor data to provide a quantitative risk assessment based on a Bayesian network, and while the work is focussed on the processes industries, an aircraft arriving at an airport could easily be interpreted as a process, with performance and the level of risk changing over time. They highlighted that the accuracy of the method is highly dependent on the quality of the near-miss data available, which could be problematic if the source was purely voluntary safety reporting. Safety reporting is of course a very valuable source of safety data, however Paltrinieri et al. (2014) found that it is prone to biases and subjectivity. Fortunately, in aviation FDM exists and provides a source of objective data, thus avoiding some of these problems.

Yeo et al. (2016) also made use of accident precursor data to develop a Bayesian network for the assessment of risk on a gas offloading process. They identified different potential accident scenarios and associated causal factors from research conducted into previous accidents. The causal factors were incorporated into the Bayesian network with probabilities assigned to each. In such a network, the structure of the Bayesian network is usually quite obvious, as the links between causal nodes and consequence nodes reflect the influence each causal factor has, as determined from accident investigation reports. With the completed Bayesian network, Yeo et al. were able to run different what-if scenarios and calculate probabilities for different consequences, demonstrating another powerful advantage over other, static forms of quantitative risk assessment.

Dynamic Bayesian networks are different to dynamic risk assessment using Bayesian networks; with the latter, the underlying network remains constant, but probabilities are updated when new data becomes available. Dynamic Bayesian networks can change over time, with the network reflecting temporal changes and the effect of state changes, perhaps derived from time series data (Fenton and Neil, 2013). A dynamic Bayesian network can consist of standard nodes found in any Bayesian network, plus temporal nodes which will change over time. Dynamic Bayesian networks are more complex than standard Bayesian networks and their use is most common in systems reliability applications, speech recognition and bioinformatics, rather than quantitative risk assessment.

2.2.3.3 Bayesian networks in aviation risk assessment

Much of the literature on Bayesian networks in risk management relates to the process industries (Kalantarnia, Khan and Hawboldt, 2009; Meel and Seider, 2006; Villa et al., 2016; Yeo et al., 2016; Zarei et al., 2017) and environmental issues (Uusitalo, 2007), however they have not been widely adopted in the aviation industry and there is little research on their use in this domain.

Perhaps a reason for this is that aviation is an ultra-safe industry (Amalberti, 2001) with catastrophic accident rates occurring in the order of 1 in every 10 million flights. This is of course a fantastic achievement, however it presents challenges to an industry keen to reduce the rate even further. The problem can be illustrated by considering some of the research already described in 2.1.4. Several of these studies were characterised by having limited access to data, with one study having to generate simulated data for 500 flights, despite a database of 2 million flights existing, albeit unavailable for research due to the perceived sensitivity of the data. Even with 2 million flights available, it would be unlikely that the dataset would contain an accident flight. This makes typical machine learning classification methods more difficult, because there is no bad-outcome data set from which to learn from. The methods have to rely on softer indicators and precursors and relatively minor outcomes to try to infer when the conditions for an accident are getting closer. As a result, there is a higher level of uncertainty with any output from risk models. In this setting, Bayesian networks are an ideal tool as they can accept uncertainty in their inputs and reflect uncertainty in the output.

Brooker (2011) agrees that risk assessment in aviation today is a great challenge and states that Bayesian networks “...are conceptually sensible models for aviation risk assessment”. However he raises a very good point in that probabilities derived from risk assessments in an ultra-safe industry are going to be very difficult to validate. To illustrate this, consider a small to medium sized airline with a fleet of 40 short-haul aircraft. In a month the fleet will operate around 8,000 flights. Against the backdrop of the accident rate being of the order of 1 in 10 million, it means

that the airline might experience a major loss in every 100 years. Fortunately the risk analyst does not have the luxury of regularly occurring catastrophes which would help validate the latest risk estimation. In this context, a number calculated indicating such a rare occurrence is of little use to an individual airline; it could be argued that 1 in 10 million is such a low rate that it is not worth bothering about. However, no sensible airline would countenance such an approach. What would make more sense to the individual airline looking to direct resources efficiently, would be to look at relative probabilities and scenarios. It is answers to questions such as *where are we most likely to have a runway veer-off* or *what effect does landing from an unstable approach have on our risk of a runway veer-off*, that are more useful, however existing methods cannot provide quantitative answers; one of the aims of this research is to describe a method which can.

In common with other literature on the topic, the focus of Brooker (2011) is on eliciting opinion from subject matter experts to provide conditional probabilities. Ding and Ru (2008) use experts to help build a Bayesian network to assess risk levels in airlines and highlight again the ability of Bayesian networks to handle uncertainty in data. However they also claim that their method avoids "*subjectivity of the experts*"; it might be possible to control for some expert bias and subjectivity, but eliminating it altogether is probably impossible. Generally, eliciting expert opinion is used where hard data is unavailable. If a method could truly avoid subjectivity it would obviate the need for hard data and there are few, if any, studies which have chosen expert opinion over hard data when it is available. Another issue for Ding and Ru (2008) is that they have not validated the probabilities they have derived, perhaps not surprisingly given the difficulties already discussed.

Luxhøj and Coit (2006) also state the rarity of accidents in the aviation industry and acknowledge the criticality of reducing low accident rates even further. Their approach in modelling low probability, high consequence events using Bayesian networks involved the following steps:

1. Select and analyse real accidents cases
2. Identify the accident causal factors
3. Construct an influence diagram to depict the causal factor interactions
4. Build the Bayesian network
5. Insert prevention barriers (interventions)
6. Evaluate the effectiveness of the barriers i.e. compare the relative risk of barriers present or absent

The Bayesian networks produced from the six steps utilise quantitative data where available and otherwise through "*knowledge engineering sessions*" with domain experts. The causal

factors were derived from accident reports using the Human Factors Analysis and Classification System (HFACS) (Wiegmann and Shappell, 2003) to try to ensure consistency. The subject matter experts' opinion is used to derive conditional probabilities if no data is available. Rather than producing a one-off number for accident probability, the models are used to establish relative risks, where a baseline is set and then interventions introduced. By running different scenarios with different interventions, the effectiveness of the interventions can be estimated and this is the approach taken later in this research. Again, validation of the method would be difficult to achieve, but at least Bayesian networks can allow uncertainty to be reflected in their output.

In related work, Bareither and Luxhøj (2007) focus on uncertainty and sensitivity in aviation risk assessment Bayesian networks. The authors acknowledge the propensity of experts to introduce bias and error, so use cognitive measures and tactics to minimise them during the elicitation process. The model structure is evaluated using sensitivity analysis. Sensitivity analysis allows the sources of uncertainty in a model's output to be identified in the model's inputs and allows the identification of the most important inputs, in this case, the accident precursors. Once again, the problems arising from the often necessary reliance on subject matter expert opinion is highlighted.

Like Luxhøj and Coit (2006), Chen and Huang (2018) make use of historical accident reports to help define a Bayesian network to evaluate flight crew performance. Accident reports were used to identify causal factor nodes, which are then arranged into an influence diagram using subject matter expert opinion. In an effort to achieve a reliable consensus of expert opinion, the Delphi Technique (see Dalkey and Helmer, 1963) was used, again highlighting some of the difficulties and mitigations necessary when relying on experts. Unfortunately, the use of experts is unavoidable when data is sparse, and Bayesian networks are an excellent method for accommodating the resulting uncertainty. However, the airline industry is supposedly data rich, with FDM and various other systems generating vast quantities of data every flight. The challenge is to make use of that data to produce risk assessments which minimise the use of subjective opinion.

2.2.4 Summary

The literature reviewed in this chapter demonstrates that a range of risk assessment methods exist, with some commonly deployed in aviation. However there is a gap in the use of quantitative risk methods in the domain of airline operations, where some methods which claim to be quantitative are actually more qualitative in nature. Qualitative methods dominate, despite the availability of data which could inform quantitative analyses.

It is possible that some form of workload analysis or statistical process control could make use of the data available, but it is Bayesian networks which show the most promise, with their ability to utilise different types of data, accommodate uncertainty in beliefs and easily provide probabilities for different scenarios. It seems that Bayesian networks are an ideal candidate to fill the gap of quantitative risk assessment in airline operations.

2.3 Runway veer-offs

Despite a general downward trend in aviation accidents (Airbus S.A.S., 2017), runway excursions persist and account for around a quarter of all incidents and accidents in air transport (Australian Transport Safety Bureau, 2008). Around 80% of runway excursions occur during landing, and 53% of those involve a veer-off (Flight Safety Foundation, 2009a), that is, a lateral rather than longitudinal excursion. Hence, this research focusses on lateral runway excursions (veer-offs), however the methods described later could be applied to other common aviation risks, such as longitudinal excursions, loss of control in flight (LOCI) and controlled flight into terrain (CFIT).

2.3.1.1 Recent runway veer-off events

Fortunately most runway veer-off events do not result in damage, death or injury, however some do and the potential is certainly there. Even without damage or injury, the incidents can still be very costly, requiring runways to be closed and aircraft to be recovered, not to mention the reputational damage caused by media pictures showing aircraft sinking into grass.

The following descriptions, extracted from avherald.com (NOMIS SOFT Datenverarbeitung Gesellschaft mit beschränkter Haftung, no date), are of recent runway veer-off events and highlight some of the common characteristics.

2.3.1.1.1 *Thai Boeing 747-400 at Bangkok, October 2018*

Aircraft landed in heavy rain and gusting winds, veered left off runway and stopped on soft ground. The aircraft received minor damage and no passengers were injured.

2.3.1.1.2 *Polar Cargo Boeing 747-400 freighter at Cincinnati, October 2018*

Aircraft touched down in strong gusting winds, veered right and became stuck in soft ground. No injuries to crew and aircraft was undamaged.

2.3.1.1.3 *Lion Air Boeing 737-800 at Gorontalo, April 2018*

Aircraft landed at night with heavy rain on the approach. Aircraft veered left off the runway, the nose gear collapsed and the aircraft was substantially damaged. No injuries to passengers or crew.

2.3.1.1.4 *Pegasus Boeing 737-800 at Trabzon, January 2018*

Aircraft landed at night on a wet runway. Aircraft veered to the left off the runway and slid down a slope and came to a stop in soft ground. There were no injuries, however the aircraft suffered substantial damage. Incident is under investigation.

2.3.1.1.5 Wind Rose Embraer 145 at Zaporozhye, September 2017

Aircraft landed with a tailwind on a contaminated runway in low visibility. Crew elected to land with flap 22 instead of flap 45, which would have been more appropriate in the conditions.

Aircraft damaged, no injuries.

2.3.1.1.6 LATAM Brasil Airbus A320 at Brasilia, November 2017

Landed in rain and temporarily veered off the runway, causing minor damage.

2.3.1.1.7 Cebu Pacific Airbus A320 at Iloilo, October 2017

Aircraft veered right off the runway during landing, suffering substantial damage. Under investigation.

2.3.1.1.8 Vietnam Airbus A321 at Da Nang, July 2017

Aircraft touched down to the right of the centreline with the first officer flying. The aircraft continued to veer right and temporarily left the runway. Weather at the time was gusting crosswind and rain. The aircraft remained on the ground for nearly two months.

2.3.1.1.9 Vietnam Airbus A321 at Hanoi, July 2017

Aircraft landed hard (2.78g) in gusting crosswind and rain, temporarily leaving the runway.

2.3.1.1.10 Peruvian Boeing 737-300 at Jauja, March 2017

Aircraft landed hard, causing gear to collapse and subsequent runway excursion. Aircraft caught fire and 29 people injured.

2.3.1.1.11 Air Canada Airbus A320 at Toronto, February 2017

Aircraft landed in low visibility and light rain, temporarily leaving the runway during the landing roll

2.3.1.2 Common causal factors

Unfortunately the list of incidents and accidents above is by no means exhaustive, not even for 2017/18. Whilst most of these events have resulted in relatively few injuries, the potential for a catastrophic outcome is obvious, and that is why the topic has received widespread attention in projects such as Future Sky Safety (Future Sky Safety Consortium, no date), European Operator's FDM forum (European Operators Flight Data Monitoring Working Group B, 2017) and organisations such as the Flight Safety Foundation (Flight Safety Foundation, 2009a).

A work package of the Future Sky Safety project was to identify risk factors associated with lateral runway excursions (Post, 2015). The research examined runway veer-off incidents and

accidents between 1994 and 2014. Causal factors were determined from accident and incident reports and the results are shown in Table 5 below.

<i>Causal factor</i>	<i>Probability (%) of causal factor in veer-off incidents (P(CF veer-off))</i>
<i>Crew performance inaccurate</i>	55
<i>Wet/Contaminated runway</i>	25
<i>Crosswind</i>	23
<i>Inaccurate info to crew</i>	22
<i>Technical issue: Landing gear</i>	17
<i>Gust</i>	11
<i>Technical issue: NW steering system</i>	11
<i>Asymmetric thrust</i>	11
<i>Unstable Approach</i>	8
<i>Hard landing</i>	7
<i>Deteriorating/poor visibility</i>	7
<i>Heavy precipitation</i>	6
<i>Aquaplaning</i>	5
<i>Technical issue: Hydraulics</i>	5
<i>Maintenance issue</i>	5
<i>Technical issue: Braking system</i>	4
<i>Technical issue: Thrust reverse system</i>	3
<i>Technical issue: Engine control</i>	3
<i>Technical issue: Electrical power</i>	3
<i>Tailwind</i>	2
<i>Technical issue: Propeller control</i>	1
<i>Runway lack of centerline lights</i>	1
<i>Technical issue: Elevator control</i>	1
<i>Technical issue: Fuel imbalance</i>	1
<i>Technical issue: Rudder control</i>	1
<i>ATC performance inaccurate</i>	1
<i>Technical issue: ILS</i>	1
<i>Technical issue: Engine</i>	1
<i>Technical issue: Autopilot system</i>	1
<i>Technical issue: Engine fire</i>	1
<i>Collision with animal</i>	1

Table 5: Lateral runway excursion causal factors, derived from Post (2015).

These causal factors relate to incidents to jet and turboprop aircraft with a maximum take-off weight greater than 5,700kg. Post (2015) is the most thorough review of veer-off occurrences available in the literature and was one of the first steps in the EU-funded Future Sky Safety project, part of which was focussed on runway veer-off risk. Not all runway veer-off events

during the period were reviewed by Post, however 103 occurrences were selected which had detailed narrative available from occurrence reports or investigation reports. Note that the percentage figures relate to whether the causal factor was present, but do not provide any indication of the magnitude of the causal factor. It is reasonable to assume that professional accident investigators would not cite a causal factor if it was present in an insignificant magnitude.

Most of the other literature has focussed on longitudinal runway excursions (Trucco, De Ambroggi and Chiara Leva, 2015; Wagner and Barker, 2014; Zwirgmaier and Straub, 2015), however Moretti et al. (2018) recently published a paper on the quantitative risk assessment of veer-off occurrences. Their analysis looked at 305 veer-off events occurring between 1980 and 2016 and examined where the aircraft came to a stop, following the veer-off. This information was used to define the lateral and longitudinal position probability distribution curves which could then be used for risk assessment of veer-off consequences at different airports. The focus was to provide airport operators (rather than aircraft operators) with a risk assessment tool to risk assess the physical characteristics of specific runways and their potential veer-off areas, however it is possible that aircraft operators could make good use of the probability distributions for their own risk assessments, particularly if operating to unforgiving airports such as Trabzon in Turkey (see Figure 64).

2.3.2 Summary

Runway veer-offs during landing are a persistent problem in the aviation industry. Due to the number of subsequent investigations, the causal factors for veer-off accidents are well known, however despite this awareness, veer-off accidents continue to happen. It is evident that new interventions are needed to reduce the accident rate.

3 Theory

3.1 Exploration of methodologies with potential application to risk assessment using flight data

Many methodologies exist which might have the potential to be applied to the domain of risk assessment. For example, in the domain of quality management, variability in process output is used as an indicator of the quality of the process (Kane, 1986). It is not a giant leap of imagination to apply the concept of process quality to that of safety of an activity in a safety critical system. That is, could the variability in performance of a safety critical activity be used as a risk measure?

Similarly, there is a wealth of literature relating to human workload and the relationship to errors and performance degradation (Ahmed et al., 2011; Hwang et al., 2008; Looije and Mioch, 2011; Sheridan, 2008; Di Stasi et al., 2016). If workload can be determined from recorded flight data, then perhaps it could be developed into a risk measure by aggregating results by airport, for example. Whether variability and task load could be used as risk measures is explored later.

3.1.1 Variability in performance

A typical flight from A to B could be considered to be a process, just like manufacturing an item or producing chemicals in a chemical plant. Airlines have standard operating procedures (SOP) and aircraft manufacturers provide flight manuals, both of which define how flights should be conducted. Of course, not every flight follows a single template. Flights between different airports may operate at different altitudes and speeds, and they will certainly follow different tracks. However, some parts of a flight should be conducted in virtually the same way every time. For example, the actions taken by pilots during a take-off will normally follow a standard sequence and flight parameters will be similar to other take-offs. Likewise for the final approach and landing.

Typically, airline SOPs define how a final approach and landing should be flown. Pilots are expected to fly the approach within set flight path, speed, thrust and configuration limits. These are often referred to as stable approach criteria. Naturally there will be variation in some of these parameters due to external influences, such as weather, however the author speculated that as these phases of flight have an optimum, could variation from that optimum be used as a measure of risk?

Statistical process control (SPC) is a methodology born out of quality management, which seeks to continually improve a process by reducing its variability (Oakland, 2012) and to detect variation to an undesirable state as early as possible. SPC acknowledges that there will be a natural variation in any process (Wheeler, 1993) due to common (or routine) causes, however there may be underlying variation due to “special” causes, and it is these exceptional causes that it seeks to highlight. The aim of SPC is to determine whether a process is in control or out of control.

Using approaches and landings as an example, SPC would expect variation due to common causes such as weather or aircraft weight; these variations occur all the time, but usually occur within a normal range.

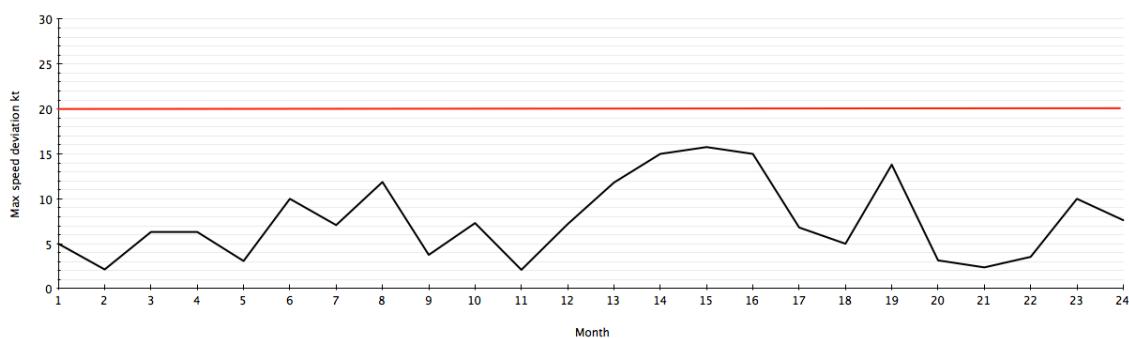


Figure 10. Simple SPC chart

For a simple example, Figure 10 shows the fictitious maximum deviation from a target approach speed for a given month over a 24-month period. Based on historical noisy data, SPC allows limits (example given in red) to be calculated for what can be expected as normal or due to common causes. Should performance deviate outside those limits, SPC infers that there is an underlying exceptional cause.

Applying this to aircraft flight data to ascertain whether a risk measure could be produced is explored further in 4.3.

3.1.2 Workload and risk

It seems intuitive that with increasing workload, an individual would be more likely to commit errors and mistakes. In 1988, NASA developed the popular “NASA-TLX” task load index to help subjectively rate perceived workload for a given task (Hart and Staveland, 1988), and thereby aim to optimise task design and subsequently improve task performance, with error reduction amongst potential improvements. NASA-TLX has been widely used since its inception (Hart, 2006), with many studies aiming to reduce errors through its use, for example, in medicine.

Yurko et al. (2010) using NASA-TLX, found that increased workload “*during task performance may increase fatigue and facilitate errors*”. However there is some debate whether a subjective high rating of workload, as provided by NASA-TLX, actually reflects any negative effect on performance (Tubbs-Cooley et al., 2018).

The subjective nature of workload self-reporting has led researchers to try to identify more objective measures. Di Stasi et al. (2016) used “gaze entropy” to evaluate workload in surgeons during surgery and similarly, Ahlstrom & Friedman-Berg (2006) also used eye movements, but in air traffic controllers, to assess cognitive workload. Less invasively, Porterfield (1997) used communication timings⁴ to assess the workload of air traffic controllers.

A question arises from the above literature, as to whether an objective measure of pilot workload could be identified. More specifically, could flight data be used to assess pilot workload and, moreover, could that measure be correlated with risks at different airports? This question is investigated in 4.4.1.

3.2 Bayesian networks and risk assessment

3.2.1 Determining BN structure

Sometimes when defining a new BN, the structure can be obvious i.e. the conditional dependencies between the nodes can be intuitive or even well described in literature. There are however, occasions when the relationship between variables is not obvious and the structure must be derived from other means. Fortunately, methods exist whereby the BN structure can be derived or *learnt* from observed data (Buntine, 1996). These methods, which often use mathematical optimisation techniques such as hill-climbing, can be extremely computationally expensive, particularly in complex BNs where multiple variations of dependencies between nodes must be tried and in most cases eventually discarded. Friedman, Nachman and Peér (1999) tackled this issue by limiting the number of candidate dependencies a particular variable could have, and this resulted a method well-suited to learning BN structure from very large datasets. Usually learning a BN structure in this way requires large datasets, with thousands of observations (Tonda et al., 2012), however it is also possible to blend this kind of learning, using smaller datasets, with existing knowledge of the problem (Heckerman, Geiger and Chickering, 1995).

⁴ The frequency and duration of radio communications between aircraft and controller.

It is worth noting that for many problems, a single best BN structure does not exist, and there may be several candidates to choose from once the learning process has completed.

The problem of BNs which relate to safety incident and risk scenarios can be the opposite of the problems mentioned above. Here, observational data might be very sparse, particular in ultra-safe industries, whereas the link between variables, and hence the BN structure, is already well known. These BNs are known as causal Bayesian networks, where incident causal factors are represented as nodes which lead to an outcome or event node. Note that causal factors can be dependent on other factors, so they may well have parent nodes. The causal links between variables and the outcome can be elicited from expert domain knowledge (Nadkarni and Shenoy, 2004), or in the case of this research, detailed examination and analysis of accident reports (Post, 2015). Other studies have taken a similar approach. Chen and Pollino (2012) used work carried out and published by other researchers to help them define their Bayesian network, modelling an environmental system.

3.2.2 Modelling nodes in a Bayesian network

The nodes in a Bayesian network must represent the probabilities and conditional probabilities of the variables in the model, and there are various ways in which these probabilities can be represented. Probably the simplest are Boolean nodes, where a node can take on only two states, usually true or false. The probabilities of the two states are complementary and must sum to 1 as they represent every possible outcome. Quite often, Boolean nodes are sufficient for every node in the network, and while the nodes are quite simple, networks consisting of solely Boolean nodes are still capable of modelling complex problems.

Nodes can also represent ordinal variables (e.g. states might be high/medium/low), or categorical (Fenton and Neil, 2013) e.g. different symptoms of a disease, aircraft type and so on. Each state is assigned a probability and the sum of all of the probabilities must equal 1. By expanding the number of states in this way, the conditional probability tables for nodes with multiple parents can become very large and complex. Bayesian networks comprised of Boolean, ordinal and categorical nodes are general described as discrete networks and they are the most common type found in literature (Scutari and Denis, 2014).

Gaussian Bayesian networks allow continuous probability distributions to be used for each node, which makes them especially useful when wishing to represent uncertainty around a variable. A practical limitation is that every variable must be represented by a Gaussian distribution, however there is no theoretical reason why networks cannot consist of mixed discrete and continuous variables with any kind of distribution (Scutari and Denis, 2014).

Networks with mixed variables are known as hybrid Bayesian networks and consist of discrete and continuous variables. The continuous distributions must be discretised before inference can be made from the network (Fenton and Neil, 2013; Langseth et al., 2009). This means that whichever distribution is used (e.g. beta, gamma), the probability distribution must be split into “bins”. At its simplest, imagine a normal distribution with mean = 0 and standard deviation = 1; the bins could be less than -3 to -2, -2 to -1, -1 to zero and so on. A probability for each bin can be calculated and used in the node probability table. Discretizing the distribution this way is known as static discretization. Static discretization, however, can lead to a loss of accuracy and defining optimal bin sizes can be difficult (Fenton and Neil, 2013).

Fortunately efficient dynamic discretisation algorithms now exist which allow better accuracy, overcoming the problems of static discretisation, see Fenton and Neil (2012) & Scutari and Denis (2015) for descriptions of discretisation.

Whichever form of discretisation is necessary, Chen and Pollino (2012) recommend using the fewest number of states necessary to adequately define the node probabilities.

3.2.2.1 Bayesian networks and empirical data

Earlier, the problems of relying solely on subject matter expert opinion to provide *a priori* probabilities for variable nodes were described, however sometimes objective, hard data is available, for example through FDM.

Sometimes the causal dependencies between variables in a network may be vague or unknown and in these cases the overall structure of the network can potentially be learned from empirical data or a combination of data and expert opinion (Heckerman, Geiger and Chickering, 1995). “*Potentially*” is a key word in the last sentence, as learning network structure from data is far from trivial and it is likely that the result will be several candidate structures (Scutari, 2017). Often, the resulting candidate structures will require some verification from subject matter experts, and there can also be problems due to the presence of latent variables which can “*induce correlations between observed variables that do not correspond to causal relations between them*” (Koller and Friedman, 2009).

In other cases the causal link between variables may be well-understood by experts and this knowledge can be used to help define the network structure (Nadkarni and Shenoy, 2004).

In the field of risk assessment, the causal links between variables are often well known and documented in accident and incident reports. This makes defining the structure of a Bayesian network much easier in most cases, as there is a logical and intuitive flow between the variables. In the case of a typical airline risk, for example runway veer-off, the causal factors are well-understood from accident reports (Post, 2015) and it is quite easy to determine how these

causal factors can be linked to a veer-off outcome. Where a risk is less well known and there is limited information available from accident reports, probabilistic approaches are available (Squillante et al., 2018) that might help produce a risk assessment.

Once the structure of a Bayesian network is determined, it must be decided how best to represent the probabilities for each node. In the context of an airline risk assessment, there is likely to be a mixture of discrete and continuous variables which might suit a hybrid Bayesian network. However, when looking at causal factors for accidents, it is often a simple case of stating whether the causal factor was present or not. This would oversimplify causal factors such as crosswinds, which are often present in normal operations, but it would be wrong to state that an accident causal factor was present with an insignificant 1 knot crosswind. Here, the boundary between normal and abnormal must be defined, or at least estimated, and fortunately in aviation that boundary is often stated in aircraft flight limitations or airline operations manuals.

For example, an aircraft may have a maximum demonstrated crosswind specified in the aircraft flight manual and an airline may have a lower value maximum crosswind component for a First Officer's landing stated in the operations manual. Likewise for operational issues such as unstable approaches; there is an allowable envelope for parameters such as airspeed and vertical speed (Moriarty and Jarvis, 2014).

This means in many cases for risk assessment where empirical data is available, Boolean nodes can be specified with probabilities i.e. the probability of causal factor being present and/or above a threshold. An example for runway veer-off might be a causal factor node for crosswind component greater than 20 knots. If the crosswind component for every landing is known, any instance less than 20 knots is considered to have the causal factor not present, but above 20 knots it is. This is a form of static discretization described earlier, and transforms the empirical data distribution into Boolean true/false states.

Unfortunately a problem arises when using empirical data in this way to define probabilities. In most risk assessments it is necessary to try to quantify the risk of extreme events and therefore necessary to quantify the probability of the prevalence of potentially extreme causal factors. Even with a very large dataset, observations of extreme occurrences are likely to be limited, hence the empirical probability distribution will be flawed for modelling the extreme values of causal factor variables. This is shown in Figure 11 below.

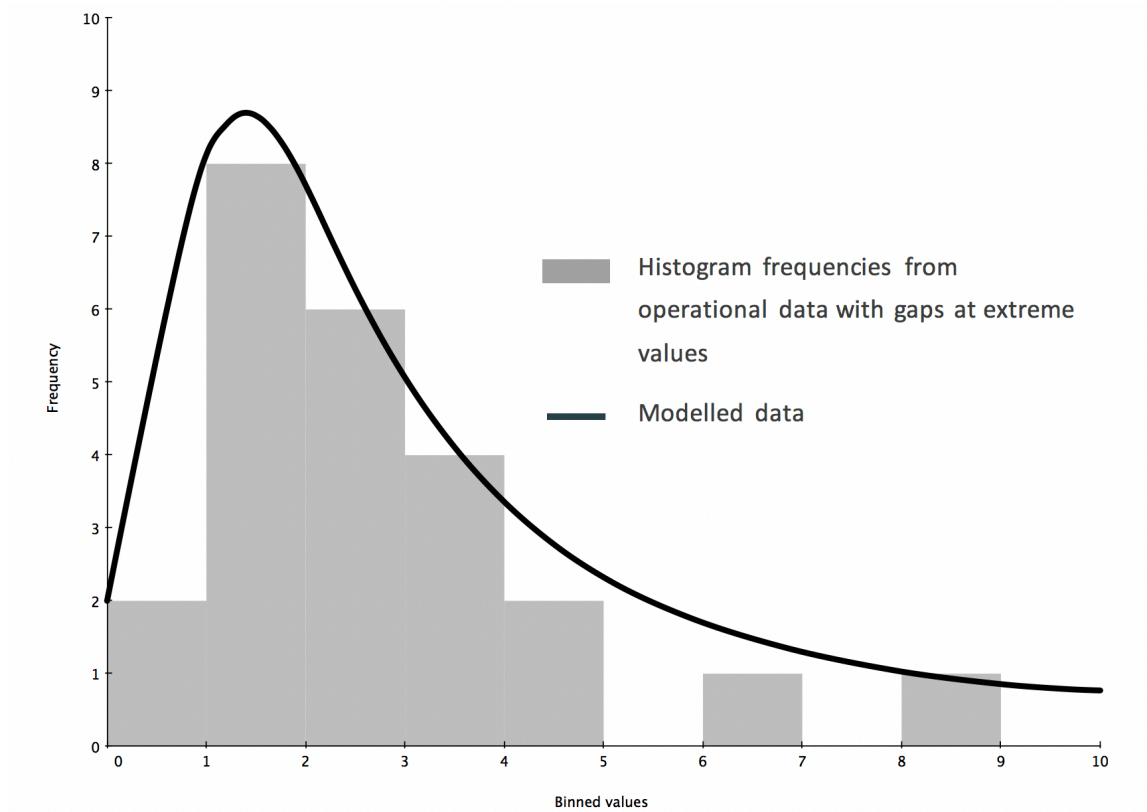


Figure 11: Modelling operational data

The histogram in Figure 11 shows gaps in the observed values above 5. In reality this usually does not mean that the probability of a value between 5 and 6 is impossible, whereas a value between 6 and 7 is possible. It is just that those values have not been observed in the available data. As a result, the empirical probability distribution will understate the probabilities of the more extreme values (Siu and Kelly, 1998), hence it is necessary to model the empirical data, with the emphasis of modelling the extremes as accurately as possible (Das and Dey, 2016). In fact, if Boolean nodes with a probability of a value exceeding a threshold are being used, the fit of the model to the extreme values is critical, whereas the fit to the “normal” values is unimportant, as long as the overall probability split between normal and abnormal is good (Hills, 1975).

Any model fitted to empirical data will have some uncertainty with respect to the fit, especially in the extremes where observations are sparse. Das and Dey (2016) quantify this uncertainty by sampling with replacement from extreme values in the empirical data to produce a bootstrap dataset from 2000 iterations. From this dataset, the 95% confidence intervals are calculated for the parameter of interest.

Not all distributions are suited to modelling extreme values. For example, the tails of normal distributions are often too thin to model extremes properly (Coleman, 2002), so alternative distributions with fatter tails are required. Examples of fatter tailed distributions are Weibull,

lognormal, gamma and exponential distributions (Fenton and Neil, 2013). There are also distributions which specifically address extreme values, such as the Generalised Pareto Distribution and Generalised Extreme Value distribution (Coleman, 2002).

The methods which have been described in this chapter are fundamental to later work in this research.

3.2.2.2 Nodes with many parents

A problem can arise in Bayesian networks when a node has multiple parent nodes. This is often the case in risk assessment when causal factor parent nodes tend to point towards a sole consequence/accident child node. Consider the simple Bayesian network in Figure 12 with one parent node and one child node.

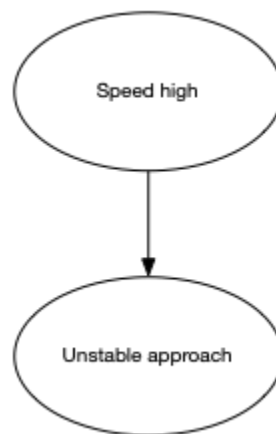


Figure 12: Single parent and single child

The probability table for the parent Boolean node is simply two probability values for yes or no as shown in Table 6.

Speed > 20kt faster than target Yes 0.0006

No	0.9994
----	--------

Table 6: Node probability table for approach speed high

In reality, there can be multiple reasons why an approach is unstable, but for the sake of this example it will be assumed that high speed is the only possible causal factor. Therefore the node probability table for the unstable approach node is now a 2 by 2 table (Table 7). The table is effectively stating that if the speed is fast, the approach must be *unstable*, and if the speed is not fast it must be *stable*.

<i>Speed >20kt faster than target</i>	No	Yes
<i>Unstable</i>	0	1
<i>Not unstable</i>	1	0

Table 7: Node probability table for approach stability, given speed

Suppose that a new causal factor is discovered for unstable approaches: glideslope deviation. The glideslope is the vertical guidance provided to pilots for precision approaches and normally results in a 3 degree flight path angle during the final approach. It is presented to pilots in units of *dots* and a 1 dot deviation is usually specified as the maximum deviation allowable for a stable approach. The new Bayesian network is now shown in Figure 13.

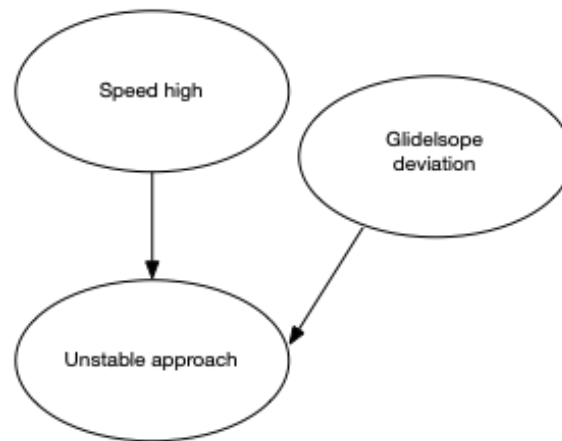


Figure 13: Two parents and single child

Similarly to the *Speed high* node, the node probability table for *Glideslope deviation* has just two values (Table 8).

Glideslope > 1 dot Yes 0.0121

No 0.9879

Table 8: Node probability table for second parent

Now the *Unstable approach* node has two parents. The approach is unstable if either the two parent nodes are true, i.e. speed is high and/or glideslope deviation > 1 dot. The approach can only be stable if both are false (Table 9).

<i>Speed >20kt</i>	No		Yes	
<i>Glide > 1dot</i>	No	Yes	No	Yes
<i>Unstable</i>	0	1	1	1
<i>Not unstable</i>	1	0	0	0

Table 9: Node probability table for child with two parent nodes

The node probability table for *Unstable approach* now has 8 values to reflect each possible combination. If another parent node was added, the table would have 16 entries and so on, increasing to 2^n , where n is the number of parent nodes. The node probability table for a node with 10 parents would consist over one thousand values, making it prohibitively time consuming to complete. Fortunately there is a function which can greatly simplify the specification of node probabilities in these cases: the *noisy-or* function.

3.2.2.2.1 The noisy-or function

Table 9 gave *TRUE* (i.e. approach is unstable) if either *Glideslope* OR *Speed* were also *TRUE*; this is the logical OR function. In a Bayesian network, the OR function at a Boolean node will produce *TRUE* if any of the parent Boolean nodes are *TRUE*. The *noisy-or* function is different, in that it allows some uncertainty as to whether a *TRUE* will be produced if the inputs are all *TRUE*. For the unstable approach example, in reality there may be causes of unstable approaches other than *Speed* and *Glideslope*, in which case it could not be said that the approach is not unstable if both *Speed* and *Glideslope* are *FALSE*. The other, perhaps unknown, causes can be dealt with using a leak value which is explained below. *Noisy-or* also has the benefit a greatly simplifying the specification of probabilities for nodes which have many parents.

Instead of 2^n values to specify the conditional probabilities of the child node, *noisy-or* allows the probabilities to be specified with just $n+1$ parameters. There is a caveat: each parent node must be capable of occurring independently of other parents and each variable must not be capable of reducing the effect of another variable (Pearl, 1988a), or in other words, there must be causal independence between the parents.

To specify a *noisy-or* function, each causal factor is assigned a value between 0 and 1 to reflect the probability that the consequence will happen if the causal factor has occurred. Looking back at Figure 13 as an example (and ignoring the tables which followed), if there was a 75% chance that an approach would be unstable if the speed was high, the speed high causal factor would be assigned a value of 0.75. Likewise for glideslope deviation; if it results in an unstable approach 60% of the time, it would be assigned a value of 0.6. In *noisy-or*, a *leak value* can also be defined. The leak value represents the probability of the consequence occurring in the absence of the specified causal factors (Fenton and Neil, 2013), for example, there may be other causal factors for unstable approaches which have not been encountered and are unknown.

For the simple example in Figure 13, just three values are required to specify the unstable approach node probability, 0.75, 0.6 and the leak value, making them a very efficient way of specifying probabilities for nodes with a *noisy-or* relationship.

Darwiche (2009) and Fenton and Neil (2012) provide full mathematical definitions of *noisy-or*.

When modelling most accident scenarios, it is usually the case that causal factors can occur independently and therefore do not break the requirements of *noisy-or*. However, if the causes are in some way dependent, a *recursive noisy-or* function can be used (Ashrafi, Davoudpour and Khodakarami, 2017; Lemmer and Gossink, 2004).

3.2.3 Propagation of evidence through a Bayesian network

New information, or evidence, can be added to any node(s) of a Bayesian network which is then propagated through the network providing updated conditional probabilities. This feature is extremely useful for running *what-if* scenarios in risk assessment and has led to the rise in popularity of Bayesian networks for producing quantitative risk assessments. The mechanism for the propagation of new evidence is quite complex and it was not until the 1980s that computationally efficient algorithms first appeared (Fenton and Neil, 2013) from authors such as Pearl (1988a) and Spiegelhalter et al. (1993).

Propagation works through a process of message passing in a tree structure, known as polytree (Darwiche, 2009) or junction tree (Fenton and Neil, 2013) algorithms and a full explanation of the algorithms can be found in Darwiche (2009), Fenton and Neil (2012) and Pearl (1988a).

Fortunately the analyst building a Bayesian network does not need to fully understand the complexities of propagation. Today there are various Bayesian network software programs which incorporate the propagation algorithms, amongst them AgenaRisk, GeNIe, HUGIN, and Netica.

3.2.4 Bayesian network software

The increasing popularity of Bayesian methods has led to the availability of a range of software allowing the user to create and query Bayesian networks. Here is a selection:

AgenaRisk – AgenaRisk is a multiplatform “design and execution environment” (Agena Ltd, 2016) for Bayesian networks, allowing discrete, continuous and hybrid networks to be modelled using a number of built-in probability distributions. The excellent “*Risk Assessment and Decision Analysis with Bayesian Networks*” (Fenton and Neil, 2013) came with a “Lite” version of the software which the book makes use of throughout.

Unfortunately that version is now defunct and the developer has blocked access to the program, meaning it will no longer run on machines where it is already installed. The user is now directed to a newer version which is extremely expensive.

BayesiaLab – BaysiaLab by Baysia is another Bayesian network environment for producing networks and the focus is on providing a user interface to make the experience as user-

friendly as possible for the non-programmer. BayesiaLab allows networks to be learned from data and provides a comprehensive suite of tools for knowledge modelling and machine learning. At \$700 for academic use for one year, it is priced similarly to AgenaRisk.

bnlearn – bnlearn (Scutari, 2010) is an R package which allows the construction and querying of discrete and Gaussian Bayesian networks. It also has a very helpful accompanying book “*Bayesian Networks: With Examples in R*” (Scutari and Denis, 2014) which provides the user with an overview of the topic area and how the methods can be applied in R. Despite being a very capable package, at the time of writing it does not support noisy-or/noisy-max functions which in practice limits the use of the software to networks where nodes do not have a large number of parents. It is however free to use.

GeNIe – GeNIe is a Bayesian network environment from the developer BayesFusion LLC. It is a comprehensive tool for building and learning Bayesian networks and has an intuitive user interface. It can model discrete, continuous and hybrid networks and supports noisy-max functions. It is cross-compatible with other Bayes software and whilst it is native to Windows, it works seamlessly on macOS and Linux via Wine. It is free to use for academic and research purposes and has an excellent user guide.

Hugin - Hugin is popular Bayesian network analysis software aimed at helping decision makers across a wide range of industries. It is cross-platform and has a clean, uncluttered user interface with plenty of literature to support users. Academic licenses are available.

Netica – Netica by Norsys Software Corporation who claim it to be “the world’s most widely used Bayesian network development software” (Norsys Software Corp., no date). Netica can learn network structures from data while coping with missing values and supports noisy-or functions. At the time of writing, it is available to academic users for \$285.

pgmpy – pgmpy is a Python library for creating and using Bayesian networks. It is fully featured, supporting discrete and continuous distributions, discretising and noisy-or. It is open source and can be found on GitHub. Like bnlearn, it does require the user to have some programming knowledge.

The list above is by no means exhaustive. For the academic user who has little programming experience, GeNIe offers wide ranging capabilities with an intuitive user interface and it is free to use for academic purposes.

4 Methodology

4.1 Flight data used in this research

This chapter describes the flight data used in this research. This is necessary because flight data can vary considerably from one aircraft type to another, and also between similar types due to differences in avionics fit, engine fit and even aircraft age.

The flight data used in this research was recorded on Airbus A319, A320 and A321 which were delivered to the operator over a ten-year period. The aircraft used the same engine variant (IAE V2500), however older aircraft had SAGEM data management units (DMU) fitted, whereas newer aircraft were fitted with Teledyne flight data integrated management units (FDIMU). Some of the older aircraft were retrofitted with the newer Teledyne FDIMU during the course of their service. Using the Digital Aircraft Conditioning Monitoring System Recorder (DAR) functionality, which allows airlines to create bespoke data frames, all aircraft were configured to record a near identical data frame consisting around 370 recorded parameters.

The data was recorded onto PCMCIA card media in the DAR recorders.

The data was recorded at 128 words per second in ARINC 717 format and converted to engineering units using Aerobytes FDM software (Aerobytes Ltd, 2014). Aerobytes presents and exports tabulated data with 8 rows per second to accommodate the normal acceleration parameter, recorded at 8Hz. Aerobytes has built-in functionality to export entire flights (i.e. all parameters) to CSV files. It is a time-consuming process, taking one minute or more per flight, but it allows the data to be converted into a format useful for researchers. One minute per flight for over 300,000 flights is equivalent to over 200 days of continuous processing, so multiple instances of Aerobytes were used on a multi-core server to expedite the process.

The flight data in CSV format amounted to around 4TB, with each flight being approximately 15 to 30MB, making storage and analysis inefficient. It was therefore decided to convert the CSV files into R Object files (.rds) which reduced the size of flight files by around 95%.

Any duplicate flights were removed and the quality of each flight assessed by checking for:

- Correct date in the filename. The filename created for the flight in Aerobytes may contain erroneous date values due to operator input errors at the time of first replay, which were corrected using the recorded flight date.

- Correct aircraft registration in the filename. It was found that occasionally the filename attributed the flight to the wrong aircraft, so this was corrected by using the recorded aircraft registration.
- Non-continuous frame counter values. An incremental data frame counter is recorded and this was checked for discontinuities within each flight. Discontinuities could be produced through data losses or errors during the splitting of the recorded data file into individual flights. Any flights exhibiting frame counter discontinuities were discarded.
- Very short flights. Flights which had a total duration of less than 10 minutes were discarded as this can be indicative of data loss and corruption issues. In the rare instance of there being any real short flights in this group, they were also discarded as the short duration would not represent normal operational flights i.e. the flights would have been maintenance test flights or training circuits.
- If no take-off or landing point could be determined, the flight was discarded.
- Any flight with a maximum recorded pressure altitude of less than 200ft was discarded.

In addition to the quality checks, the flight data was cleaned to remove characters from numerical values (e.g. Aerobytes sometimes labels numerical values with “*”).

The R code used to perform the quality checks and data cleaning can be found in the appendices.

The recorded flights were from aircraft flying on a mostly European network between 2003 and 2011, for a single airline and standard operating procedures were common across the aircraft types.

4.2 FDM results data

In addition to the recorded flight data, the results of the FDM analysis derived from the recorded data was also available to this research. The FDM results data was contained within a database listing each flight and details of any exceedance events (see 2.1.3.2.1 Exceedance detection) which were found. Each event and flight can be associated with a month/year of occurrence, an aircraft type and departure/arrival airfields. This database was used by the airline to produce safety trend information for a safety management system. Details of flights and events were verified and validated by a dedicated flight data analyst while the data was collected (2004 – 2011).

This dataset is used later to provide a validation benchmark for new methods.

4.3 Variability as an indicator of risk

Following on from the literature reviewed in 3.1.1, operational variability of some flight parameters might provide a measure useful in assessing and quantifying risk. This was explored further by examining a selection of recorded parameters from 3NM to just beyond touchdown. On a 3-degree approach profile, 3NM is approximately equivalent to 1,000ft in height terms. Many airlines have standard operating procedures (SOP) relating to the approach phase of flight, which require the aircraft to be established on the approach and meet criteria relating to speed, altitude, flight path, configuration and engine thrust. Usually the requirement stipulates that these criteria are met by 1,000ft (International Air Transport Association (IATA), 2016), hence the choice of a 3NM window.

In preparation for parameter extraction and possible future extractions, each of the R object (.rds) flight files, described in 4.1, was split into the following flight phases:

- Shutdown:
 - Both engine N1 values < 20%
- Taxi:
 - Either engine N1 between 20% and 65%,
 - aircraft on ground (all squat switches = ground),
 - groundspeed < 20kt
- Takeoff:
 - both engines N1 > 65%,
 - gear down,
 - height above take off runway <100ft
 - groundspeed > 15kt
 - flaps extended (i.e. not clean configuration)
- Initial climb:
 - positive rate of climb (derived from recorded pressure altitude)
 - height above take off runway >100ft
 - aircraft not on ground
 - airspeed > 90kt
 - flaps extended
- Climb:
 - positive rate of climb
 - flaps retracted
 - landing gear retracted
 - aircraft not on ground

- Cruise:
 - no significant vertical speed
 - landing gear retracted
 - flaps retracted
 - pressure altitude > 15,000ft
- Descent:
 - positive rate of descent
 - flaps retracted
 - landing gear retracted
 - aircraft not on ground
- Approach:
 - positive rate of descent
 - flaps extended
 - aircraft not on ground
 - height above landing runway >200ft
- Landing:
 - positive rate of descent
 - flaps extended
 - landing gear extended
 - aircraft not on ground
 - height above landing runway <200ft
- Landing rollout:
 - From touchdown to 12 seconds after touchdown.

The flight phases above were defined by the author and based on previous experience in specifying FDM algorithms.

Whilst the data frame from the A320 series which generated this data consists of 370 parameters, it was necessary to derive some additional parameters for this analysis. They were:

- Height above landing runway
 - Recorded pressure altitude minus the landing runway pressure altitude (calculated as mean of values of pressure altitude from touchdown plus 5s to 10s)
- Precise point of touch down
 - Coarse: change of squat switch from air to ground
 - Precise: search coarse period -5s to + 5s for increase in normal acceleration > 0.1g
- Time to touchdown

- Distance to touch down
 - Integrated groundspeed from touchdown point back
- Runway heading
 - Mean recorded magnetic heading during last 5 seconds of landing roll
- Difference of airspeed to target airspeed
 - Difference between recorded airspeed and recorded target approach speed
- Difference of heading to runway heading

The following parameters were then extracted from the 1,000ft above runway point to the end of the landing roll-out:

- Pitch
- Roll
- Airspeed
- Groundspeed
- Height above landing elevation
- Distance to touchdown
- Radio altitude
- Engine thrust, N1 left and right
- Side-stick positions in pitch and roll for left and right seat
- Rudder position
- Normal acceleration
- Longitudinal acceleration
- Lateral acceleration
- Glideslope deviation (in dots)
- Difference of airspeed to target airspeed
- Difference of heading to runway heading

(The extraction R code can be found in Appendix 2.)

Plots were made of distance to touchdown from 3NM (X axis) against the other parameters listed above. Distance was chosen as a common axis rather than height because distance is a more fixed quantity for an approach⁵ and therefore allows better benchmarking of different approaches.

Initially it was hoped that plots could be made for thousands of flights, however it was only possible to plot just under 900 flights at a time. Each of the plots, shown in Figure 14 and

⁵ Approaches are typically flown using a 3° descent profile, so there is a target height for any given distance from touchdown. However, the actual aircraft height can vary around the target profile.

Appendix 3, comprises 889 approaches, equivalent to around 750,000 data points; plotting more took too much processing time and it was felt that over 800 flights were sufficient for this exploratory work.

The density of the grey in each of the plots represents the number of data points, with black equivalent to 16 overlaid points. Using the “Height above airfield” plot (Figure 14) as an example, the X-axis shows distance to touchdown from 3NM (“-3”) to touchdown (“0”) and the end of the landing roll. The Y-axis is the extracted variable, in this case showing a range from around -100ft to 1,300ft. As one would expect, the spread of values becomes more concentrated as the approach nears the touchdown point, however it is interesting to note the spread at 3NM, ranging from around 800ft to 1,200ft.

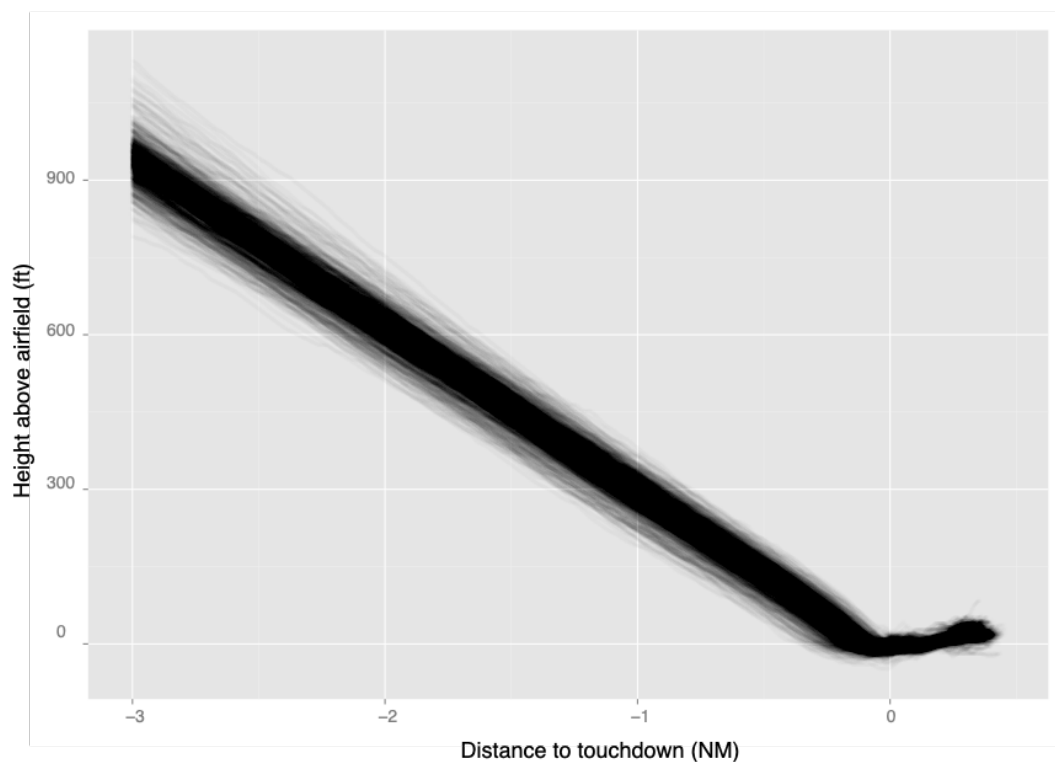


Figure 14. Height above airfield (ft) vs distance from touchdown (NM) for 889 flights

Outliers are visible as faint grey lines or points. Further plots in Appendix 3 show the variability of other key approach features.

It is the “spread” at any point in the approach that could potentially be a useful measure of safety performance. To investigate this further, statistical process capability analysis tools were used.

4.3.1 Process capability analysis using flight data

The data for each of the plots was split into 0.1NM distance bins and a process capability chart produced for each bin. The R Package “Quality Control Charts” (Scrucca, 2004) was used. Figure 15 shows an example chart, in this case for height above runway from 1.5 to 1.4NM.

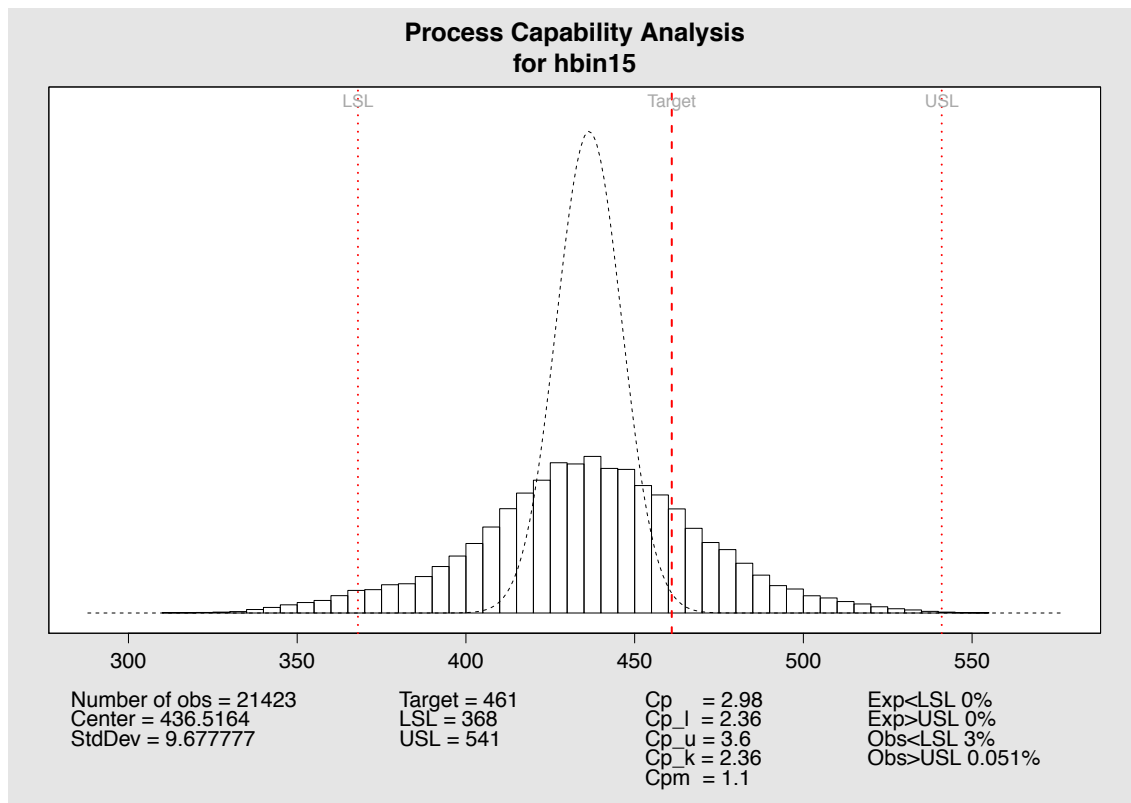


Figure 15. Process Capability Analysis chart for height above runway from 1.5 to 1.4NM

The number of data points used is 21,423 and they appear to be normally distributed, with a mean of around 436ft.

The target (461ft) is the height representing zero three-degree glideslope deviation at 1.45NM (midpoint for the bin).

The lower (“LSL”) and upper (“USL”) limits are calculated from a 1 dot deviation (i.e. ± 0.4 degrees (Airbus, 2009)) on the glideslope (assuming a 3-degree glideslope).

Note that the distribution is shifted to the left from the target. This is due to the distances and the 3° flight path being calculated from the touchdown point, however for the majority of the approaches the aircraft would be tracking a 3° flight path to the glideslope aerial, but will flare at around 30ft and land some distance beyond it, hence the distribution appears to show low flight paths. Future work would have to resolve this discrepancy.

Potentially the standard deviation of the distribution could be used as a comparative measure that may be useful for monitoring operational variability. As previously mentioned, there is a

prescribed way for approaches to be flown, so a measure of the variability of this process might be a useful safety performance indicator. Comparisons could be made for different time periods, or for different airports.

This method is aligned with variation risk management (Thornton, 1999), which is usually applied to manufacturing processes and products, however it is an interesting concept which could make use of flight data to monitor operational safety in airlines. A possible avenue for future work could be to use the features shown in Figure 14 and Appendix 3 and use a measure of variation in the feature at a particular distance (e.g. 1NM from landing) over time to provide a safety performance indicator. This would be relatively easy to implement, especially if the operator is using some form of measurement analysis which extracts specific features from every flight. Assessing the temporal change in the distribution of those parameters could help identify emerging trends, but it could be difficult to use the method to quantify common risk types, such as CFIT or runway excursion as the features do not necessarily map onto those risk types.

The potential for using variability in flight data parameters as a risk measure is explored further in the following section.

4.3.2 Variability in basic flight parameters

Two strategies for investigating variability were adopted, referred to here as Phase (a) and Phase (b). Phase (a) took samples of height and speed from each flight at a series of time intervals. These samples were then combined with those from other flights and the variability amongst them analysed.

Phase (b) took a slightly different strategy in that the variability of three parameters - height, speed and flight path angle – was calculated for each flight and then these measures of variability were combined with those of other flights.

Figure 16 shows the difference between the two phases. Phase (a) takes a lateral cross section of multiple flights at a given time point, whereas Phase (b) compares multiple flights by taking longitudinal, temporal measures from parameters.

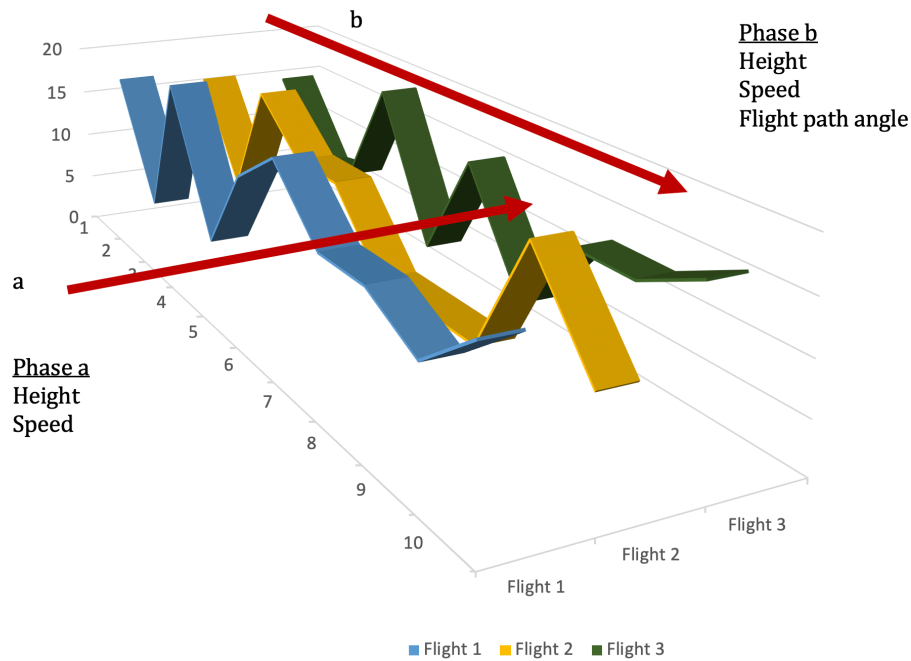


Figure 16: Different methods for examining variability

4.3.2.1 Phase (a)

To explore whether variability in basic recorded flight parameters could be a potential indicator of risk, a comparison of heights and speeds during approaches to particular airports versus FDM event rates (see 4.2) was made. The height above airfield and speed were extracted from each flight at 30, 60, 120, 180, 240, 300, 420 and 600 seconds from touchdown. The R code for extraction is provided in Appendix 4.

Gini's mean difference (GMD) is a measure of variability which can be used for non-normal distributions. GMD was chosen for this analysis over variance because it was not certain whether the parameters would be normally distributed and it has been argued (Yitzhaki, 2003) that it can be more informative than the variance.

Initially a quick comparison was made of the extracted speeds for three airports, London Heathrow (LHR), Nice (NCE) and Leeds Bradford (LBA). The GMD for the speeds for each airport was calculated and compared to the FDM ranking of airports. The GMD for LHR speeds showed more variability than NCE, placing LHR higher in any subsequent ranking. This strongly contradicted traditional FDM results, which consistently showed NCE as having a much higher event rate than other airports. Overall the results for all three airports had no relation to traditional FDM results and therefore it was considered that the variability of approach speeds would not be a good risk indicator. The focus therefore turned to height.

The height at 120 seconds was the first parameter to be examined in detail. For each of 25 airports the GMD was calculated and the airports ranked accordingly, with the airport with the

highest GMD ranked as #1. This ranking was then compared to the FDM event rate ranking and no significant correlation was found (Spearman 0.25).

120 seconds to landing is a point which is fairly well established on the approach, corresponding to height above the airfield of around 1,300ft on a 3° glide path, therefore the variability in heights earlier in the approach was examined. The GMD in height at 300 seconds to touchdown was calculated for each airport and the GMDs ranked, with the highest GMD ranked as #1. Again, only a weak correlation was found (Spearman 0.485). This was repeated for the height at 600 seconds from touchdown and the correlation was even weaker (Spearman 0.327).

It was therefore concluded that the variability in basic flight parameters does not correlate with FDM results and therefore variability in these parameters is unlikely to be a good risk indicator.

4.3.2.2 Phase (b)

Phase (a) extracted values from each flight and then calculated the variability of these values across flights to different airports; Phase (b) instead calculated the variability of three parameters from each flight and these measures of variability were aggregated by airport. The three parameters were height above airfield, airspeed and flight path angle. The calculations of height above airfield and flight path angle are provided in Appendix 4.

For each flight, each parameter was extracted over a 3 minute period. The time period was 3 minutes 10 seconds before touchdown to 10 seconds before touch down. The Gini's Mean Difference (GMD) was then calculated for each parameter/period. The GMD for each parameter from each flight was then aggregated by airport, creating a vector of GMD's for each airport. The mean GMD for each airport was then calculated, and the means then ranked with the highest scoring airport ranked as #1. This ranking was then compared with the FDM ranking as shown in the table below:

AIRPORT	FDM rank	FPA GMD mean	FPA GMD rank	Height GMD mean	Height GMD rank	CAS GMD mean	CAS GMD rank
EMA	1	1.065213	3	632.3	24	18.11	12
EFL	2	1.136	1	714	8	10.72	25
VRN	3	0.8100279	8	742.7	4	20.51	2
NCE	4	0.8558118	5	656.2	22	13.28	24
LBA	5	0.7840928	11	720.7	7	18.68	9
NAP	6	0.5104	25	834.9	1	17.79	13
DUB	7	0.793582	10	697.6	16	18.8	7
JER	8	1.131692	2	624	25	14.75	22
ALC	9	0.535	23	701.6	14	16.51	19
IBZ	10	0.6181636	16	678.7	21	14.61	23
PMI	11	0.6037	18	731.5	5	18.51	10
MME	12	0.9176062	4	635.9	23	15.73	20
BRU	13	0.8089087	9	709.5	10	21.57	1
INV	14	0.6832587	13	687.3	20	14.82	21
MAN	15	0.6155	17	714	9	19.83	5
AMS	16	0.8284441	7	707.6	11	18.71	8
VCE	17	0.5825	22	763.4	3	19.89	3
ABZ	18	0.6965698	12	694.4	17	17.55	14
EDI	19	0.6427	15	690.8	18	16.88	17
AGP	20	0.5326	24	782.8	2	17.47	15
GLA	21	0.6445	14	703.7	13	16.97	16
LHR	22	0.5853041	21	728.4	6	18.28	11
BHD	23	0.8532302	6	690.4	19	16.74	18
CDG	24	0.6	19	704.1	12	19.68	6
HAI	25	0.5940665	20	698.1	15	19.86	4

Table 10: GMD rankings for Phase 3b

The Spearman correlation coefficients for the rankings were as follows:

FDM rank and FPA GMD rank: 0.46

FDM rank and Height GMD rank: -0.08

FDM rank and CAS GMD rank: -0.22

It was therefore concluded that the methodology employed in this phase was unlikely to produce useful risk indicators.

4.4 Workload as an indicator of risk

Unstable approaches are a causal factor in runway excursions (Post, 2015) and the author has conducted many incident investigations to try to understand why they happen and what preventative measures could be put in place. A common theme emanating from interviews with pilots during these investigations, was that the approaches had been hurried and characterised by a high workload. This, and previous research described in 3.1.2 led to the hypothesis that workload could be an indicator useful in helping to quantify risk.

4.4.1 Identifying workload

Flight data can provide an indication of workload, however it cannot provide a complete picture due to the limitations of what is recorded. Flight data cannot, for example, show whether a pilot is calculating landing performance, or providing instructional guidance to an inexperienced colleague. However, given those limitations, it still might be possible that flight data can be used as a relative indicator of workload.

Therefore, flight data parameters relating to pilot actions were selected from the Airbus A319 series data frame (see Appendix 5 for extraction code). Parameters were selected which showed a positive pilot action (rather than automated action), for example, selection of the landing gear down, selection of flap, a change in selected heading and so on. The full list is provided in the table below.

<i>Anti ice PB1 on</i>	<i>Selected altitude</i>
<i>Autobrake low/med/high armed</i>	<i>Selected decision height</i>
<i>Autopilot 1 engaged</i>	<i>Selected EFIS mode</i>
<i>Autopilot 2 engaged</i>	<i>Selected EFIS range (P1 & P2)</i>
<i>Autothrust engaged</i>	<i>Selected flight path</i>
<i>Baro setting P1/P2</i>	<i>Selected heading</i>
<i>Cockpit call</i>	<i>Selected track</i>
<i>ECAM page selected</i>	<i>Selected vertical speed</i>
<i>Fasten seatbelts lamp on</i>	<i>Speed brake command</i>
<i>Flap angle</i>	<i>TRK/FPA select</i>
<i>Flight director 1 engaged</i>	<i>VHF1 keyed</i>
<i>Flight director 2 engaged</i>	<i>VHF2 keyed</i>
<i>Ground spoilers armed</i>	<i>VHF3 keyed</i>
<i>Landing gear</i>	

Table 11: Flight data parameters to monitor workload

Side-stick and throttle lever positions were deliberately omitted; some approaches may be flown manually and the constant variation in these parameters would have led to manually flown approaches being highlighted as high workload approaches. Of course this may be true, however it would have prevented comparison with normally flown approaches, so parameters were chosen which could indicate high workload whatever type of approach was being flown.

The last 180 seconds of each flight prior to touchdown were analysed. To provide context, at 180 seconds from landing an A319 will be established on the approach, but possibly not fully configured for landing. The aircraft's height will usually be around 2,000ft and it will have approximately six or seven nautical miles to run before touchdown.

During each of the 180 seconds, a check was made to see whether any of the parameters in Table 11 above had changed state. If they had, a count of one was added to a counter for that particular one-second period. For example, if at 100 seconds from touch down, the autopilot was disengaged, the radio was keyed and flap extended, a count of 3 would be allocated to the 100th second. The result was a table, with a row for each flight and 180 columns representing the last 3 minutes of each flight. The count of state changes was listed in each column. This table was then summarised to show how many of the 180 seconds had one state change, how many had two state changes, how many had three and so on.

<i>State changes</i>	<i>Mean frequency</i>
x0	138.804
x1	23.204
x2	15.800
x3	1.993
x4	0.188
x5	0.010
x6	0.001

Table 12: Mean frequency of state changes

Table 12 shows that on average, during 139 seconds there were no state changes, during 23 there was one, during 16 there were two and so on. Six was the maximum number of state changes identified within any single second.

Overall there were 150 intervals which had five state changes and very few for six, hence five was chosen for further scrutiny because it represented the more extreme cases and had a relatively large number of cases.

For each airport, the number of occasions five state changes (x5) occurred was divided by the number of arrivals to that airfield, to provide a ratio. The airports were then ranked by this ratio, so the airport with the highest number of x5 occurrences per arrival was ranked #1.

(Rankings of the variables was used to aid comparison and also reduce the potential for spurious correlations caused by the normalisation (i.e. rate per n flights) of FDM result data.)

Similarly, the FDM event rate (derived from 4.2 *FDM results data*), expressed as events per 1000 arrivals, for each airport was ranked, so the airport with the highest event rate was ranked #1. These rankings were then examined for any correlation and none was found (Spearman -0.042), which was expected as the analysis was crude and one would not necessarily expect instantaneous examples of multiple actions to be a robust indicator. After all, a busy approach may have actions spread more evenly throughout the time-period.

To examine this further, a new ranking was created, which ranked the airports by total number of actions identified. Whilst not perfect, given the limitations of flight data in determining true workload, it would provide an indicator of approaches which contained a relatively high number of state changes. This was also compared to the FDM event rate ranking and again, no correlation was found (Spearman -0.098).

A further exploratory analysis was conducted whereby the approaches were ranked by the total number of actions recorded. The top seven were then examined to identify reasons for the high number of actions.

<i>Flight ID</i>	<i>No. of actions</i>	<i>Notes</i>
190899	188	Multiple FDM events during a non-precision approach (NPA)
198598	182	Flap placard speed exceedance and overweight landing
193902	180	NPA following a go-around
193610	179	NPA with no FDM events
231846	177	NPA following a go-around
196319	173	NPA with high approach speed FDM event
202679	172	NPA with late configuration change and high speed FDM events

Table 13: Top seven approaches with most actions

Precision approaches are much more common in the dataset of A319 approaches, as most of the approaches are made to airports where precision guidance is available, so it is interesting that in all but one of the flights in the table above, the approaches are non-precision. Like most FDM software, the two FDM programs at this particular airline were unable to differentiate between types of approach, hence it is not possible to provide here any data on NPA frequency, however it appears that the number of recorded actions might be a useful indicator. Likewise, the seven approaches above are all slightly unusual in that they have FDM events and/or are NPAs, therefore it is possible that recorded actions could be useful in highlighting atypical approaches.

4.4.2 Comparison between number of FDM events and recorded actions

As an alternative to comparing rankings, a comparison was made of the number of events triggered on each flight versus the total number of actions. No correlation was found (Pearson 0.03). The flights were then split into two groups: those with FDM events and those without. The table below shows basic summary statistics for the two groups.

	Mean number of actions	Median number of actions	Minimum no. of actions	Maximum no. of actions
Flights with FDM events	63.5	43	1	188
Flights with no FDM events	61.0	37	0	182

Table 14: Summary statistics for approaches with and without FDM events

A comparison of the density plots for the two groups is provided below.

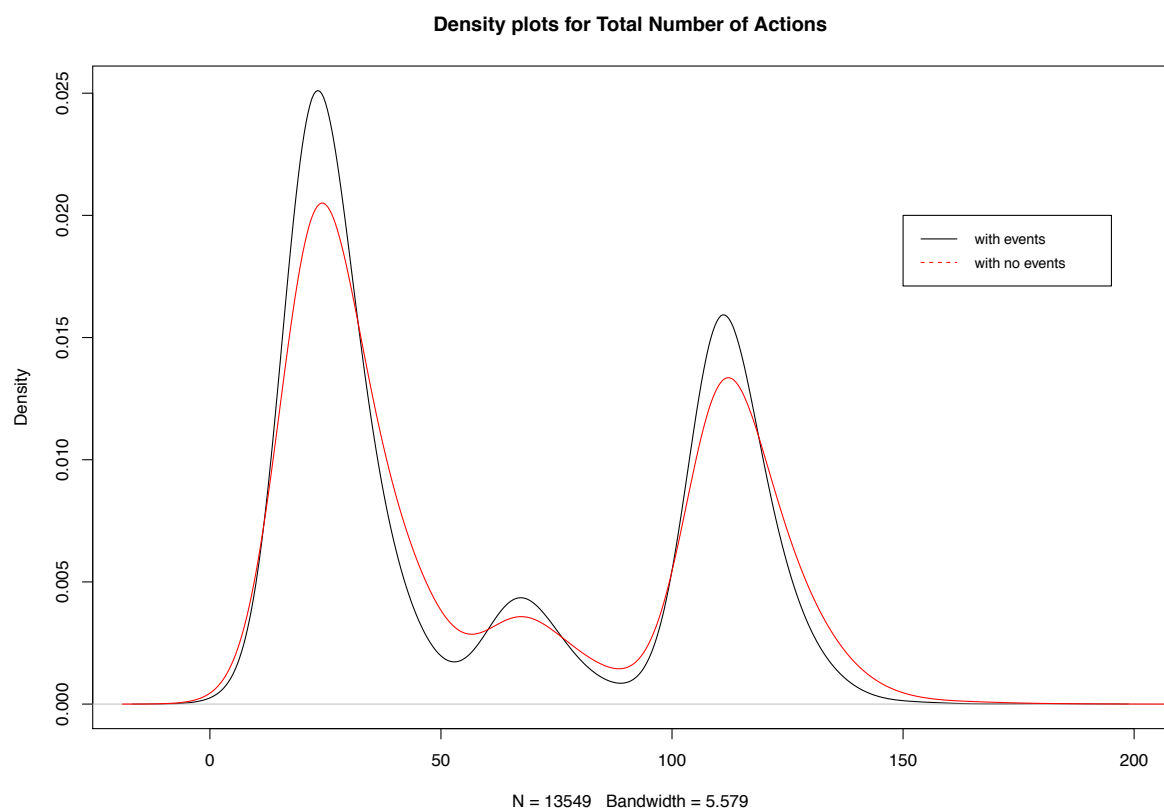


Figure 17: Density plots of Total number of actions for approaches with and without FDM events

The density plots show there is little difference between the number of recorded actions for the two groupings, however the bi-modal distribution is interesting and worth investigating further.

4.4.2.1 Further investigation into bi-modal distribution

The data was put into three groups, A, B and C, with Group A having 0 to 50 actions, B having 50 to 100 and C having 100 to 150.

Data from a sample of flights from each group was visually inspected in Aerobytes and no distinguishing features were apparent. Each group was then aggregated by arrival airport to see if any particular airport was over-represented in any group. As can be seen in the screenshot in Figure 18, the number of arrivals at each airport corresponded with the ratio sizes of each group. For example, LHR arrivals were split into each group (A, B & C) by the ratios 0.568/0.098/0.335. For ABZ the ratios were 0.545/0.105/0.350. Each airport followed similar ratios, indicating that no particular airport was over-represented in a group.

```
> table(GroupA$Arrive)
ABZ AGP ALA ALC AMM AMS BCN BHD BHX BRU CDG DKR DUB EDI EFL EMA ESB FAO FCO GLA GYD HAJ IBZ IKA INV JER KMD LBA LGW LHR
487 19 1 33 1 986 7 185 2 715 93 1 64 235 13 7 2 2 2 780 1 17 7 1 123 122 1 504 24 5198
LYS MAN MME NAP NCE NCL PIK PMI REU SDV SOF STN SVX SZG TRN VCE VRN XXX
22 446 105 80 74 15 6 145 3 2 2 1 2 10 4 119 15 1

> table(GroupB$Arrive)
ABZ AGP ALC AMS BCN BFS BHD BRU CDG DUB EDI EFL EMA ESB FCO GLA HAJ IKA INV JER KTT LBA LGW LHR LYS MAN MME NAP NCE NCL PIK PMI REU SOF SZG TBS VCE
94 6 9 252 1 1 54 148 14 15 59 3 1 2 1 152 7 2 27 33 1 86 8 898 9 100 27 13 17 4 1 30 4 2 3 1 28
VRN
6

> table(GroupC$Arrive)
ABZ AGP ALA ALC AMS ATH BCN BHD BRU CDG DAM DUB EDI EFL EMA FAO GLA HAJ IBZ INV JER LBA LGW LHR LPL LYS MAN MME NAP NCE
313 21 1 22 714 1 3 120 503 65 2 58 202 6 1 1 425 19 4 88 101 246 21 3064 1 20 346 110 64 60
NCL PMI REU RVN SDV SOF STN SZG TRN VCE VRN
12 127 9 2 1 5 1 3 8 111 13
```

Figure 18: Screenshot from R showing number of flights from each group by arrival airfield.

The investigation then started to look at which action parameters were present in each group. Through a slow process of elimination, it was found that a subset of aircraft had two parameters (Captain's barometer setting and First Officer's barometer setting) which alternated values every other second. This contributed to the action count seen in Group C i.e. the third hump in the distribution. This demonstrates how the vagaries of recorded flight data can sometimes cause problems for researchers.

It was also found that flap angle was used as an action parameter instead of flap lever position. When a stage of flap is selected it takes time for the flap to reach the commanded setting, and therefore each second in which the flap angle changed was being counted as an action. Not all approaches had flap changes, so those that did had additional actions counted and this contributed to Group B, the middle hump.

Both barometer parameters were removed and the flap angle parameter changed to flap lever position. The resulting distribution of actions is shown below.

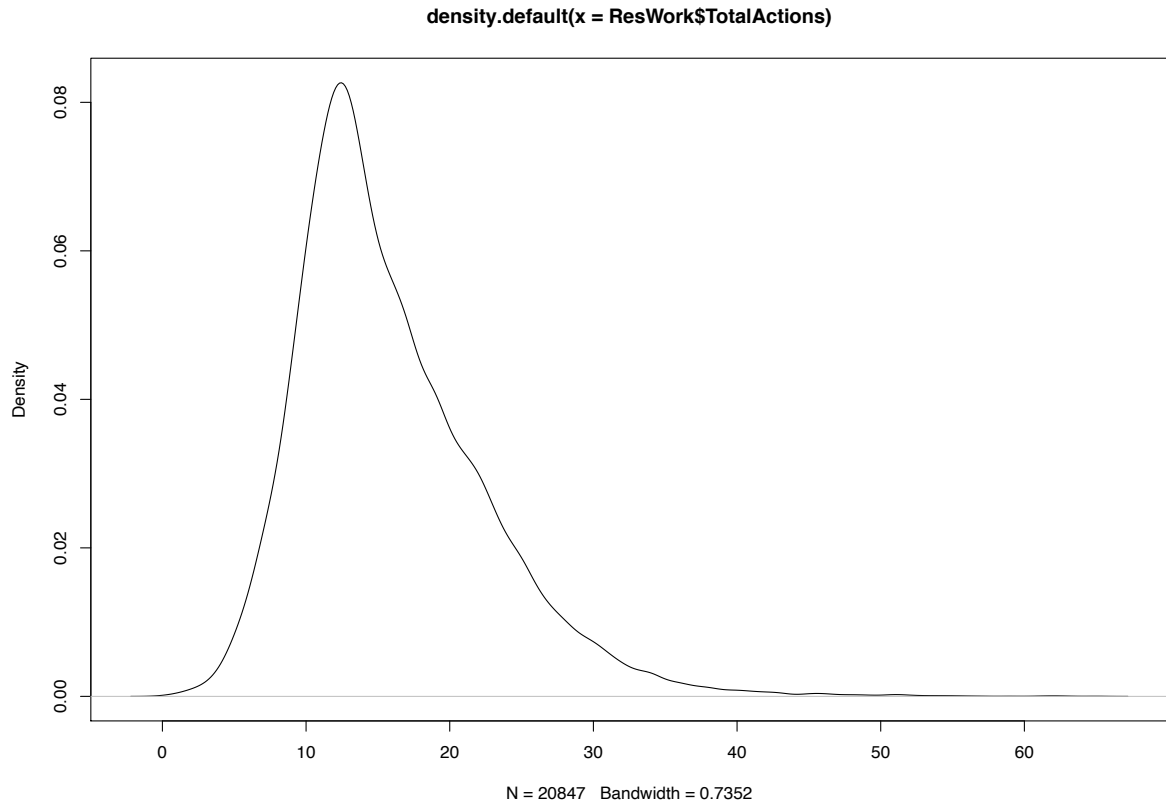


Figure 19: Updated density plot of actions following changes to parameters.

Once the erroneous parameters had been removed a comparison was made of the number of events triggered on each flight versus the total number of actions. No correlation was found (Pearson 0.11).

	<i>Mean number of actions</i>	<i>Median number of actions</i>	<i>Minimum no. of actions</i>	<i>Maximum no. of actions</i>
<i>Flights with FDM events</i>	16.7	15	1	65
<i>Flights with no FDM events</i>	15.4	14	0	62

Table 15: Updated Table 14

In light of the refined data, Figure 17 was reproduced.

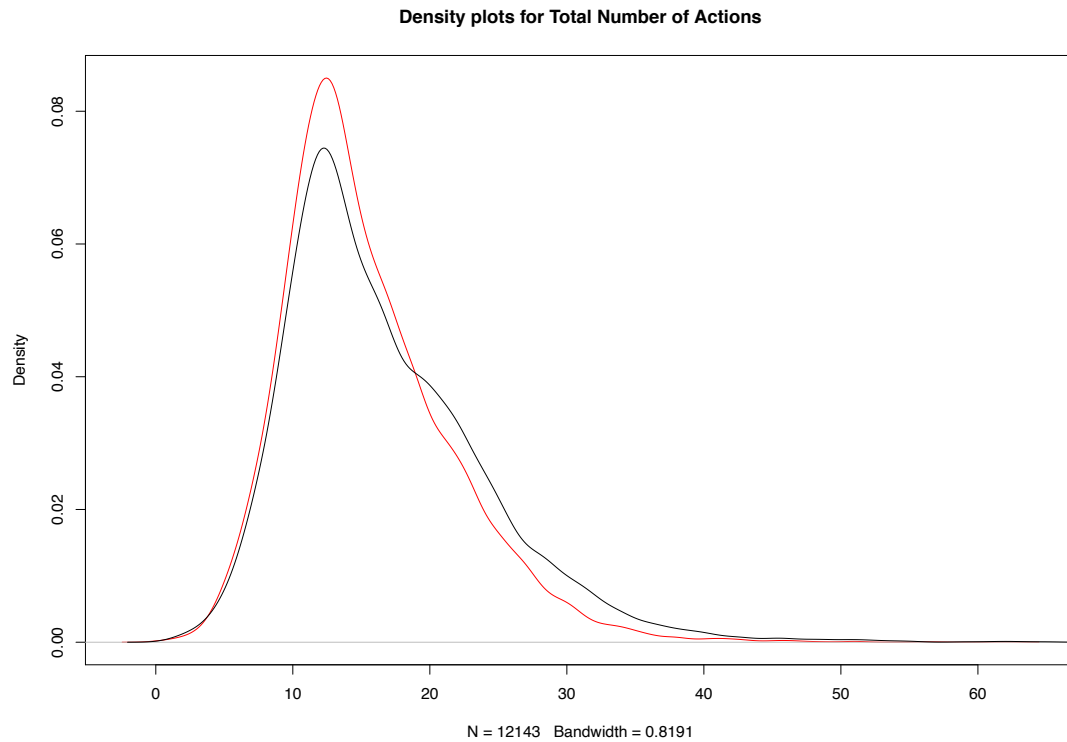


Figure 20: Density plots from refined data. The red plot is approaches that had no FDM events.

Despite the refinement of the data, once again the density plots show there is little difference between the number of recorded actions for the two groupings.

The *BEST* package (Kruschke and Meredith, 2018) in R provides a Bayesian alternative to a t-test and was used to test the difference of means of the Total Number of Actions for the two groups (flights with, μ_2 , and without, μ_1 , FDM events). See Kruschke (2013) for a description and examples of Bayesian estimation.

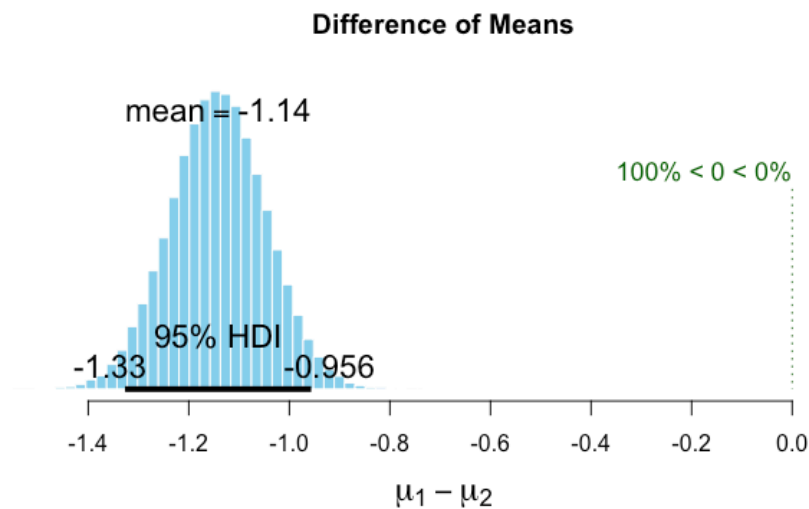


Figure 21: BEST difference of means for Total Number of Actions for flights with and without FDM events

Figure 21 shows the distribution of credible values of the differences of the means, along with a point estimate of the most likely difference (-1.14) and the probability that the difference is less than zero (100%). There is a small difference between the means of the two groups, with the FDM event group having approximately one more action per flight than the no event group.

4.5 Summary of variability and workload methods

The methods described in this chapter so far have used recorded flight data to investigate whether they can supplement existing FDM analyses and help quantify risk. Process capability analysis could be applied to flight operations for flight phases that could be treated as a process. For example, for take-off and landing there are defined procedures which an airline would expect pilots to follow, and the goal is to make these procedures as standardised as possible, with as little variation as possible. Measuring this variation over time could be used as a safety performance indicator and in risk assessment.

This chapter has also investigated using pilot workload derived solely from recorded flight data, as a possible risk measure. Comparison of the results with those from traditional FDM analysis found very little correlation. It is acknowledged that the methods were tested to see if they could supplement existing FDM methods, and it could be argued that there would be no point using a new method if it resulted in the same knowledge discovery as existing methods i.e. a perfect correlation would not be expected. However, to validate the methods, further work is needed to determine if higher workload and higher variability result in higher levels of risk.

The rest of this chapter is dedicated to examining Bayesian methods for quantifying risk, specifically Bayesian networks.

4.6 Bayesian network derived from accident report causal factors

The literature described in 2.2.3 *Bayes' theorem and Bayesian networks*, showed that Bayesian networks are becoming ever more popular for quantitative risk assessment. It is also apparent that Bayesian networks lend themselves well to modelling typical aviation risks, such as loss of control in flight, controlled flight into terrain and runway excursions. This chapter aims to produce a Bayesian network for a specific aviation risk, runway veer-off, and use recorded flight data to provide informative priors for runway veer-off causal factors.

To recap, Bayes theorem for an accident can be stated as:

$$P(A|CF) = \frac{P(CF|A) P(A)}{P(CF)} \quad (3)$$

Where:

$P(A|CF)$ is the probability of an accident, given the presence of a causal factor

$P(CF|A)$ is the probability of a causal factor, given an accident has occurred

$P(A)$ is the probability of an accident

$P(CF)$ is the probability of a causal factor occurring

The work of Post (2015) provided the probabilities of causal factors in runway veer-off accidents which were derived from the analysis of runway veer-off accident reports. In terms of Bayes theorem, these probabilities provide the probability of a causal factor, given an accident, i.e. $P(CF|A)$. As mentioned in the Literature Review, Post's analysis of runway veer-off events is the most thorough to date and therefore a suitable source for these probabilities.

$P(A)$ (i.e. the probability of a runway veer-off) can be found in aviation accident statistics such as those published by the International Air Transport Association (2017) and the Flight Safety Foundation (2009).

The prevalence of causal factors, i.e. $P(CF)$ is quite challenging to define, particularly for those causal factors which are related to the operation of the aircraft, rather than environmental factors such as weather. The following section will describe how some of the causal factors might be identifiable from recorded flight data.

If $P(CF)$ is available, $P(A|CF)$ can easily be calculated using Bayes theorem, thus providing a probability of a veer-off event for a single causal factor. However, causal factors do not always occur singularly. For example, gusting winds, crosswinds and wet runways have been identified

as causal factors of runway veer-offs and it is of course possible that all three factors could be present during a landing. The challenge, therefore, is to combine the causal factors into various scenarios and assess the probabilities of a runway veer-off incident, which is something Bayesian networks are ideally suited for.

4.6.1 Causal factors identifiable from flight data

Table 16 is a reproduction of Table 5 with a third column, titled “*Identifiable?*” added to show which of the causal factors could be directly identified through flight data monitoring and thereby aid the quantification of runway veer-off risk. This column is mostly influenced by the availability of parameters recorded on board aircraft, however it is possible to combine recorded flight data with data from other sources. For example, by associating flights with the externally observed actual weather conditions (METAR reports), it is possible to get data about the likely runway condition and visibility conditions at the time of the landing.

<i>Causal factor</i>	<i>Probability (%) of causal factor in veer-off incidents ($P(CF veer-off)$)</i>	<i>Identifiable?</i>	<i>Note</i>
<i>Crew performance inaccurate</i>	55	Partly	
<i>Wet/Contaminated runway</i>	25	Partly	From METAR
<i>Crosswind</i>	23	Yes	
<i>Inaccurate info to crew</i>	22	Partly	From METAR
<i>Technical issue: Landing gear</i>	17	No	
<i>Gust</i>	11	Partly	From METAR
<i>Technical issue: NW steering system</i>	11	No	
<i>Asymmetric thrust</i>	11	Yes	
<i>Unstable Approach</i>	8	Yes	
<i>Hard landing</i>	7	Yes	
<i>Deteriorating/poor visibility</i>	7	Partly	From METAR
<i>Heavy precipitation</i>	6	Partly	From METAR
<i>Aquaplaning</i>	5	No	
<i>Technical issue: Hydraulics</i>	5	No	
<i>Maintenance issue</i>	5	No	
<i>Technical issue: Braking system</i>	4	No	
<i>Technical issue: Thrust reverse system</i>	3	No	
<i>Technical issue: Engine control</i>	3	No	
<i>Technical issue: Electrical power</i>	3	No	
<i>Tailwind</i>	2	Yes	
<i>Technical issue: Propeller control</i>	1	N/A	
<i>Runway lack of centerline lights</i>	1	No	
<i>Technical issue: Elevator control</i>	1	No	

<i>Technical issue: Fuel imbalance</i>	1	No
<i>Technical issue: Rudder control</i>	1	No
<i>ATC performance inaccurate</i>	1	No
<i>Technical issue: ILS</i>	1	No
<i>Technical issue: Engine</i>	1	No
<i>Technical issue: Autopilot system</i>	1	No
<i>Technical issue: Engine fire</i>	1	No
<i>Collision with animal</i>	1	No

Table 16: Causal factors identifiable from flight data

To keep the Bayesian network relatively simple, only causal factors present in more than 5% of incidents will be included as nodes. Most of the remaining causal factors relate to technical issues, so if maintenance records and data were available, these causal factors could easily be added to the Bayesian network. For now, the focus is on using flight data to provide prior probabilities for each node and all but one causal factor (tailwind) which can be identified from flight data occurred in more than 5% of incidents. Note that the remaining causal factors will not be ignored as they will be rolled into a leak factor for a *noisy-or* function (see 3.2.2.2.1).

The following sections describe how these causal factors can be derived.

4.6.1.1 Crew performance inaccurate

Post (2015) describes this as “....a wide range of crew handling from (in rare events) crew errors to (in most cases) non-optimal response to a special situation.” This might be non-compliance with standard operating procedures and application of non-optimal technique. Recorded flight data will have many parameters related to the functioning of the aircraft and from these it is possible to assess crew performance for some, but not all tasks, hence this factor is labelled as “partially” identifiable. Flight data generally has good coverage of parameters relating to thrust handling, steering and braking, but identifying changes in those parameters as “inappropriate” is difficult and perhaps a topic for future research.

As such, it was decided to omit this causal factor from this research because it mostly relates to pilots’ actions being inappropriate for the conditions they encountered, whereas the focus of this research is to model those conditions, rather than the response to them. In future, this causal factor could be added to the network if data on the prevalence of inappropriate actions by pilots became available. A potential source of this data could be some form of historical measure of inappropriate actions by individual pilots or another measure based on pilots’ training records, however this data is sensitive and may not be available for this type of analysis. Without this data, it is not possible to differentiate between individual pilots’ propensity to act inappropriately, so pilots have to be treated as an homogenous group.

Therefore, as a causal factor, the probability would be static and would add nothing to the Bayesian network except more complexity.

4.6.1.2 Wet/Contaminated runway

The runway condition isn't directly recorded on the aircraft, however it might be possible to infer it from deceleration and braking parameters. A simpler method however, is to lookup the actual weather observed at the airport at the time of the landing. This is available through historic METAR reports.

4.6.1.3 Crosswind & tailwind

Available from wind direction and wind speed parameters recorded on board, and from METAR reports.

4.6.1.4 Inaccurate information to crew

This cannot be determined from flight data, however discrepancies between aircraft recorded and METAR reported wind speeds and directions can be quantified.

4.6.1.5 Technical issues (various) and maintenance errors

The list of causal factors contains a number of technical failures and whilst it might be possible to infer the presence of a failure from flight data, failure data is not recorded directly. However, it could be possible in future to combine flight data with maintenance records for post-flight analysis.

4.6.1.6 Asymmetric thrust

This can be determined directly from flight data by comparing the thrust or N1 speed of each engine.

4.6.1.7 Unstable approach

Most airlines have standard operating procedures where approaches to landing have to fulfil a set criteria, otherwise the approach should be abandoned. Typically the criteria has limits relating to speed, flightpath, thrust setting, roll angle and configuration. Flight data can be used to determine if an approach was stable.

4.6.1.8 Hard landing

Normal acceleration values are usually recorded at 8Hz providing a good indicator of hard landings.

4.6.1.9 Deteriorating or poor visibility

Similarly to Wet/Contaminated runway, this can be derived from METAR reports.

4.6.1.10 Heavy precipitation

From METAR reports it is usually possible to determine whether heavy precipitation was occurring when the observation was made, however it does not necessarily mean that heavy precipitation was encountered during the landing.

4.6.2 Proposed BN structure for lateral runway excursion

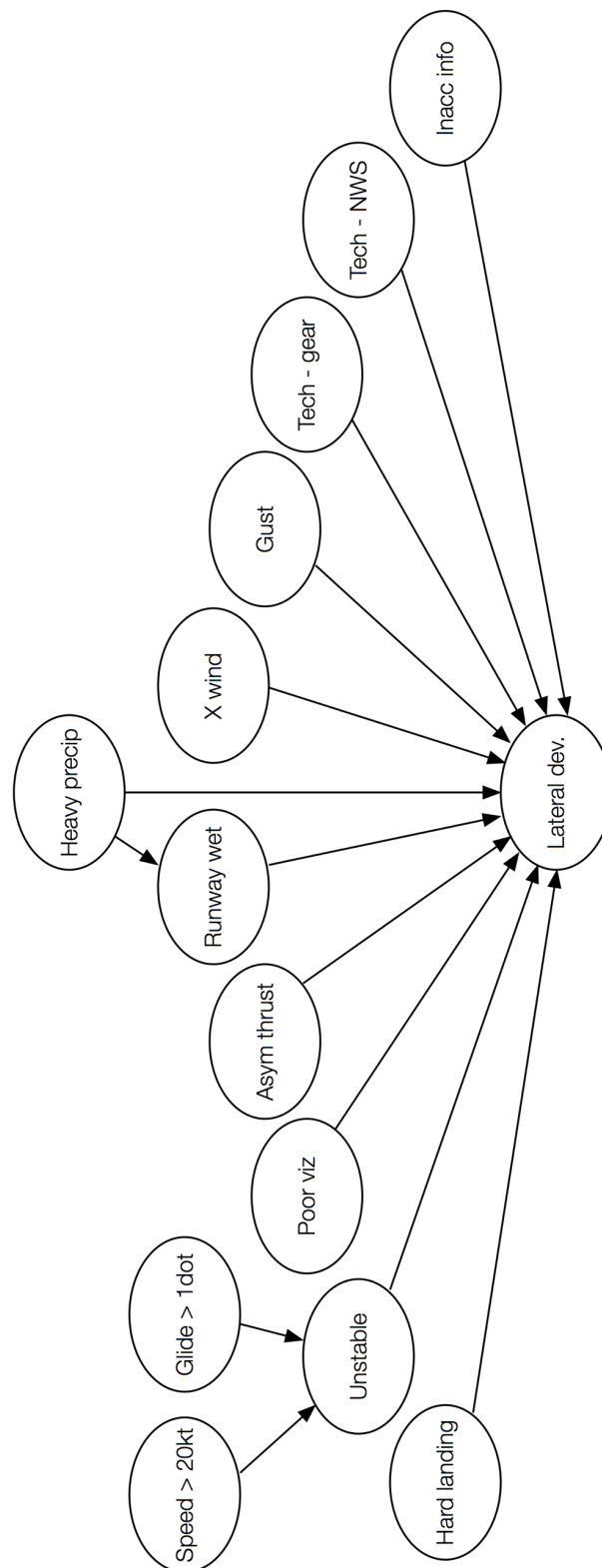


Figure 22: Bayesian network for lateral runway excursion

Figure 22 shows a causal Bayesian network for a lateral runway veer-off event. Most of the BN is simply the causal factor nodes linking directly to the lateral deviation node, however the *Unstable* and *Runway wet* nodes also have parents.

Unstable approaches can be the result of exceedances in a number of parameters, such as speed, flightpath, rate of descent, roll angle, thrust setting and aircraft configuration. FDM is very good at identifying these exceedances so it would be possible to add these as parents to the unstable approach node. They are omitted here however for the sake of simplicity, so the two most common exceedances in unstable approaches have been used.

In most cases, if heavy precipitation is present at or near an airport, it is likely that the runway will be wet, hence the causal links between those nodes. However, there may be a small chance that the heavy precipitation is very localised on the late stages of the approach to landing (obscuring visibility), but the runway remains dry, hence the direct link to the lateral deviation node.

In keeping with the advice of Chen and Pollino (2012) and Marcot et al. (2006), the network is kept as simple as possible, however if new data sources become available or new causal factors are identified, they can be relatively easily added to the network.

Once the structure of the network is complete, the next stage is to define the prior probabilities for each node.

4.6.3 Deriving node priors from recorded flight data

In order to model the node probabilities for each causal factor (as described in 4.6.1), various features were extracted from recorded flight data from 310,448 flights.

4.6.3.1 Extraction of parameters from flight data

Each flight was loaded into R and documentary data, such as the date, arrival and departure airports were extracted from the filename created by Aerobytes. Numeric parameters were then linearly interpolated to 8Hz where necessary and character parameters cleaned to remove spaces and punctuation. For character parameters which are all recorded at less than 8Hz, the last observation was carried forward to create 8Hz data.

This section describes the extracted parameters. See Appendix 7 for R code used to extract data.

4.6.3.1.1 Basic parameters

The squat switch positions for each of the three landing gears was then used to determine when the aircraft was *on the ground* and a coarse take-off point i.e. squat switches transition from *ground* to *air*. A coarse landing point was determined in the same way and refined using normal

acceleration values to create a *precise touchdown point*. The *end of the landing roll* was defined as the point of precise touchdown plus 12 seconds.

The *runway heading* was derived from the mean magnetic heading during the 5 seconds before the end of the landing roll.

4.6.3.1.2 Wind speed and direction

The recorded wind speed and direction was extracted at 5 seconds prior to touchdown and the crosswind component calculated. This simple snapshot value was deemed sufficient for this analysis, however three other methods were considered:

- V1. Calculate crosswind component from recorded wind direction and speed at 5 seconds before touchdown (TD-5)
- V2. Derive mean crosswind TD-25 to TD-5 from other recorded parameters (see NLR wind reconstruction concept (van der Geest, 2018)) i.e.
 - a. True air temperature
 - b. Mach
 - c. Drift angle
 - d. Groundspeed
- V3. Calculate mean crosswind component TD-25 to TD-5 from recorded wind direction and speed
- V4. Calculate mean crosswind component TD-21 to TD-1 from recorded wind direction and speed

Each method was used to compute the crosswind component of 10,000 landings. The correlation matrix for the four methods is shown here:

	V1	V2	V3	V4
V1	-	0.9624343	0.9585477	0.9691531
V2	0.9624343	-	0.9964235	0.9963572
V3	0.9585477	0.9964235	-	0.9972284
V4	0.9691531	0.9963572	0.9972284	-

Table 17: Correlation matrix for wind calculation methods

All methods are highly correlated. The least correlated methods, V1 and V3, have a mean of their absolute differences of 1.69kt, with 95.2% of the mean differences being < 5kt.

Comparing V2 and V3, the mean of their absolute differences is 0.54kt and 99.98% of the mean differences are <5kt. 98.1% are less than 2kt.

The simplest (V1) method was chosen for this analysis due to the small differences. (The R code for this analysis can be found in Appendix 8.)

4.6.3.1.3 *Other parameters*

A further set of parameters was extracted, as described below (see Appendix 7 for extraction code). Not all of these parameters are used in the lateral runway excursion BN described previously in 3.2, however due to the lengthy processing of the large number of flights, additional parameters were extracted in case they could be used in later analysis, or to extend the BN to longitudinal excursions in future.

Asymmetric thrust

Between touchdown -5s and the end of the landing roll, the duration of any period where the difference between N1L and N1R > 10% was extracted.

Actual vs target airspeed at 50ft radio height

The difference between the recorded airspeed at 50ft radio height and the recorded target approach speed was extracted.

Maximum normal acceleration at landing

The maximum normal acceleration value from touchdown -5s to touchdown +5s was extracted.

Heading deviation at landing

The difference between the recorded magnetic heading at touchdown and the runway heading was extracted and expressed as an absolute value.

Rudder deflection

Between touchdown and the end of the landing roll, the maximum, minimum, mean, median and standard deviation values for rudder deflection were extracted. Note negative is left deflection.

Nose wheel steering

Between touchdown and the end of the landing roll, the maximum, minimum, mean, median and standard deviation values for nose wheel steering were extracted. Note negative is left deflection.

Glideslope deviation

The glideslope deviation (where available) at 150ft RALT and 50ft RALT were extracted.

Localiser deviation

Between touchdown and the end of the landing roll, the maximum, minimum, mean, median and standard deviation values for localiser deviation were extracted. Note negative is left deviation.

Lateral acceleration

Between touchdown and the end of the landing roll, the maximum, minimum, mean, median and standard deviation values for lateral acceleration were extracted.

Longitudinal acceleration

Between touchdown and the end of the landing roll, the maximum, minimum, mean, median and standard deviation values for longitudinal acceleration were extracted. In addition, snapshot values were taken at touchdown +3s, +5s, +7s and +10s were taken.

Time to spoiler deployment

The time in seconds from touchdown to first spoiler deployment was extracted.

Time to reverse thrust deployment

The time in seconds from touchdown to first reverser deployment was extracted.

Maximum N1

The maximum value for N1 for each engine from touchdown to the end of the landing roll was extracted.

Autobrake setting

The selected autobrake setting at touchdown was extracted.

Time to first brake pedal input

The time in seconds from touchdown to first brake pedal input was extracted.

Total brake pedal input

The sum of each brake pedal input from touchdown to the end of the landing roll was extracted. This value was extracted to provide a comparative indicator of overall braking effort. In addition, the mean and maximum values for the same period, for each pedal were extracted.

Idle thrust

If the N1 values on both engines at touchdown were <50%, idle thrust was extracted as TRUE.

Pitch and roll

The pitch and roll angles at touchdown were extracted.

Airspeed and groundspeed

The airspeed and ground speed at touchdown were extracted.

Flap

The flap angle at touchdown was extracted.

4.6.3.2 Derived parameter cleaning

Each of the derived data parameters listed above was summarised using the *Hmisc* package (Harrell Jr, with contributions from Charles Dupont and many others., 2017) in R. The summary data was used to identify and validate outliers and, where necessary, invalid values were removed and not used in the analysis. Where there was any doubt regarding validity, the flight data from the originating flight was inspected.

4.6.3.3 Addition of METAR data

The *weatherData* (Narasimhan, 2017) package in R was used to match landings to historical METAR observations. The observation closest to the landing time was used for wind and visibility. If either the observation closest to the landing time or the one before had a precipitation event, the runway was deemed to be wet.

4.6.3.4 Addition of runway data

The flight data used for this research consists of landings on runways with similar width characteristics and similar elevations, so runway characteristics are not considered to be critical for this work, however it is acknowledged that runway characteristics would be critical for any risk assessment of longitudinal excursion or if operations were conducted to airports with narrow runways. Therefore runway characteristics were added to the existing dataset for each landing. Runway data was obtained from <http://ourairports.com/data/> (Megginson, no date).

See Appendix 9 for R code for addition of runway data to dataset.

4.6.4 Node modelling for Bayesian network

As previously discussed in 3.2.2, it is often the more extreme values of a probability distribution which are of interest when assessing risk.

Even though empirical frequencies were derived from a large data set, in excess of 300,000 flights, in some cases there was still inadequate information to properly estimate the probabilities of the more extreme values. This section details how the empirical data related to each node was modelled. The overall approach is shown in Figure 23.

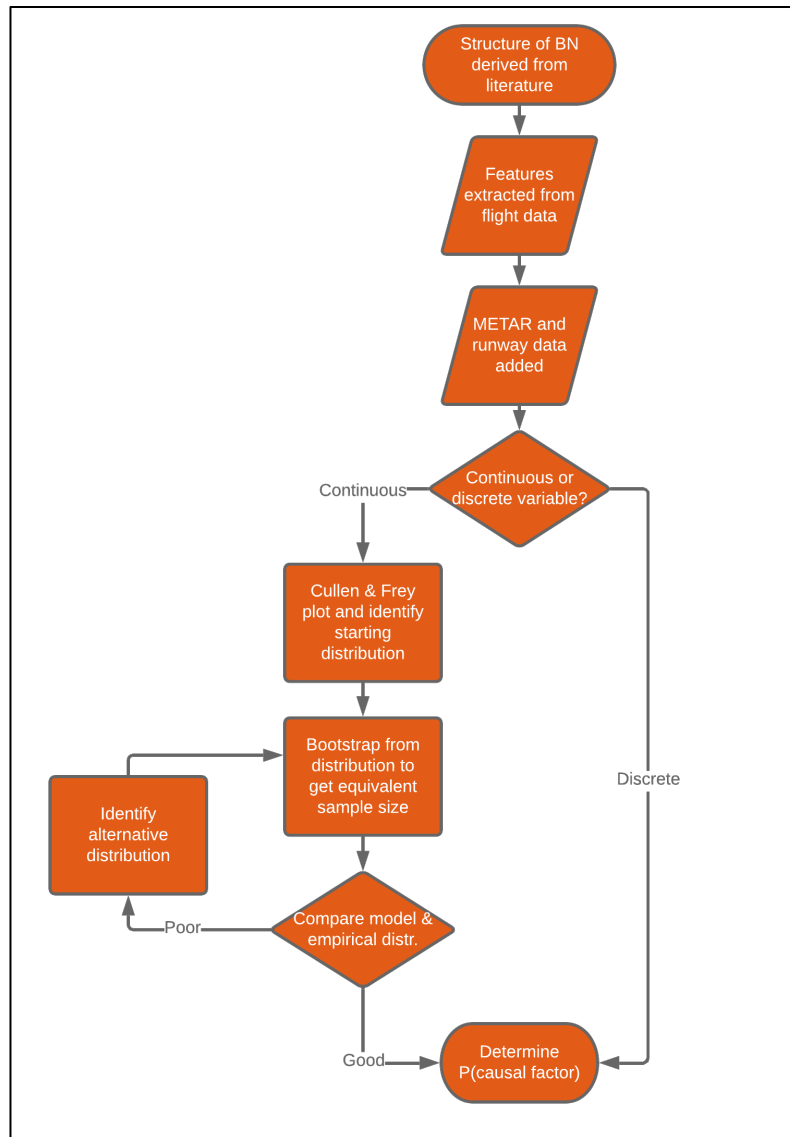


Figure 23: Basic process for node modelling

4.6.4.1 General approach to modelling continuous variables

A consistent approach was taken to model the continuous variables with R markdown scripts produced each time to ensure the modelling could be repeated. See Appendix 8 for an example of an R markdown script.

Firstly, the clean dataset was loaded into R and the variable of interest extracted. A histogram and/or an empirical cumulative distribution was plotted to see an overview of the data. For each variable there would likely be a number of candidate distributions which could be used to model the data, so a Cullen and Frey graph (Cullen and Frey, 1999) was plotted to identify likely candidates.

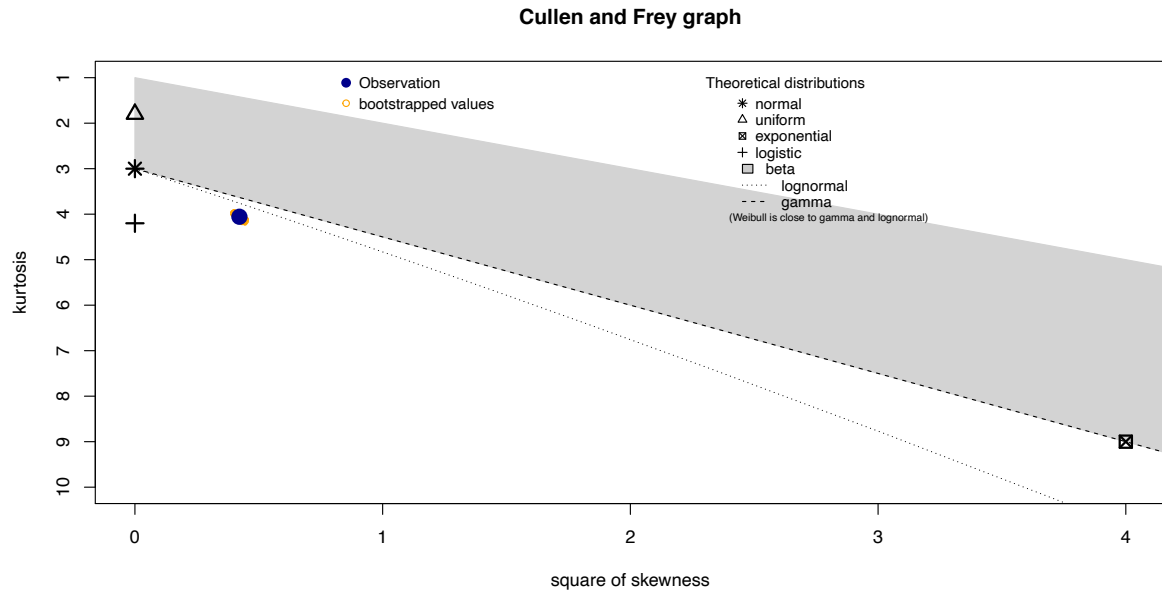


Figure 24: Example Cullen & Frey graph from *fitdistrplus*

Cullen and Frey graphs are skewness-kurtosis plots with common distributions plotted either as singular points (e.g. normal, exponential) or as lines (e.g. gamma) and areas (e.g. beta). The skewness and kurtosis of the empirical distribution is plotted onto the graph and this can help steer the analyst towards an appropriate distribution. Bootstrapping can be used to represent any uncertainty in kurtosis and skewness values. The example in Figure 24 suggest that a lognormal or gamma distribution might be a good starting point.

Using the *fitdist* function from the *fitdistrplus* package (Delignette-Muller and Dutang, 2015), the likely candidate distribution was fitted to the empirical data, initially using the maximum goodness of fit method (with the Cramer-von Mises distance). The parameters for the distribution were then used to create a simulated dataset with the same number of samples as the empirical dataset, so the two datasets could be compared. In risk assessment, most interest is in the extremes, so a threshold was chosen in the empirical data which represented the more extreme values. Where possible, the threshold was based on an aircraft limitation or operational standard operating procedure. If neither of these were available, a threshold value which represented unusual observations, often around less than 10% of the population was chosen.

The number of empirical occurrences exceeding that threshold were compared with that of the simulated dataset, with the expectation that a similar number of exceedances should be seen. This was repeated 1,000 times (i.e. simulated dataset of 310,448 samples from the model distribution with the number over the threshold counted) and a 95% credible interval was used to compare with the empirical values over the threshold. If the simulated data exceedance observations were not aligned to the real exceedance observations, the method was repeated

using a different distribution fitting method. It was found that quantile matching estimation produced the best results for fitting distributions to the more extreme values, especially when weighting was given to the extreme quantiles, as would be expected. In some cases, the extreme quantiles were matched at the expense of the bulk of the distribution, however this is an acceptable compromise when the extreme values are of primary interest (Das and Dey, 2016), as is the case usually in risk assessment.

The method for each individual causal factor is described in 4.6.4.3.

4.6.4.2 General approach to modelling discrete variables

Several of the causal factor probabilities (e.g. runway wet, gusting wind, heavy precipitation), especially those not derived from flight data, are expressed through simple true or false, present or not present, Boolean statements. In these cases, it was not necessary to model the empirical distribution and simple Boolean probabilities were used for each node probability table.

4.6.4.3 Modelling specific variables

4.6.4.3.1 Crew performance inaccurate

This causal factor encompasses a potentially wide range of inaccurate actions by pilots. Many of these, but probably not all, could be identified through flight data, however it is not the purpose of this work to define algorithms for measuring or identifying inaccurate crew performance. It was decided to omit this causal factor from the BN for the reasons described in 4.6.1.1.

4.6.4.3.2 Wet runway and heavy precipitation

Runway state is not directly available from recorded flight data, however it can be estimated from METAR reports. METAR reports were added to each of the flights used in this study, which were mostly operated in western Europe, hence the probability derived may not be representative of a global probability.

METARs from these flights indicated a probability of landing on a wet runway to be 9.98%.

Another causal factor identified from accident reports was the presence of heavy precipitation. Whilst heavy precipitation cannot be identified from flight data, it should be reported via METARs. To establish a probability for the presence of heavy precipitation, 12 years of London Heathrow (LHR) METARs were used and the probabilities shown in Table 18 derived. LHR was chosen as an exemplar, being the most common airport in the dataset and the METAR information routinely differentiates heavy precipitation from other precipitation, which was not always the case for other airports in the data set.

Heavy precip. Yes 0.0017

No 0.9983

Table 18: NPT for heavy precipitation

The BN was structured so that the wet runway node has heavy precipitation as a parent node. This structure reflects the causal link between the nodes i.e. if heavy precipitation is present, it's likely that the runway will be wet. However, there is a possibility that heavy precipitation could be encountered during an approach, but the runway could be dry, hence the estimated 1% probability for heavy precipitation with runway dry in Table 19.

Heavy precip.	No	Yes
Wet	0.0998	0.99
Not wet	0.9002	0.01

Table 19: NPT for runway wet

4.6.4.3.3 Crosswind

The empirical cumulative distribution function for the observed absolute crosswind is shown in Figure 25.

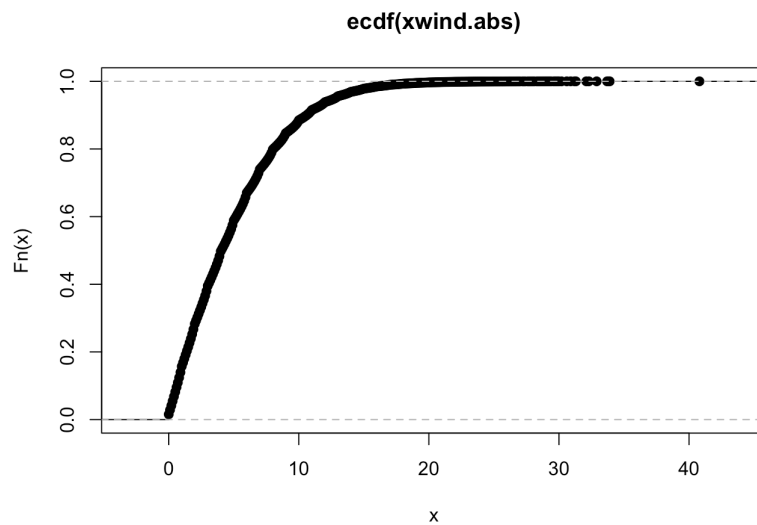


Figure 25: Empirical cumulative distribution function for observed absolute crosswind.

The summary statistics for the observed data are:

```
min: 0    max: 40.8
median: 4.1
mean: 5.019634
estimated sd: 4.018592
estimated skewness: 1.125867
estimated kurtosis: 4.436616
```

The R (R Core Team, 2017) package *fitdistrplus* (Delignette-Muller and Dutang, 2015) was used to produce a Cullen & Frey graph for the observed data, which indicated that a gamma distribution might be a candidate for the model (Figure 26).

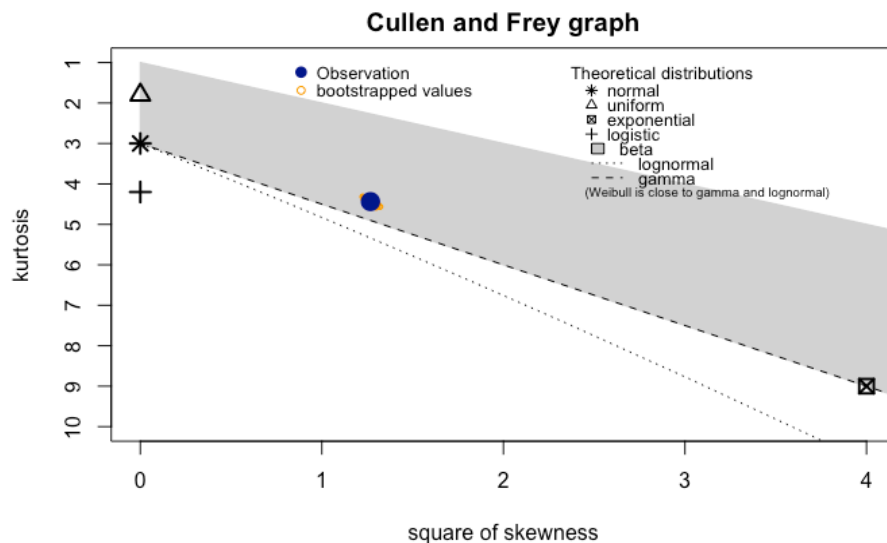


Figure 26: Cullen & Frey graph for crosswind

Initially, maximum goodness of fit (Cramer-von Mises distance) was used to fit the gamma distribution, however this resulted in poorly modelled crosswind quantiles greater than 15kt, estimating between 130 to 180 observations (95% credible interval) above 35kt in comparison to a single observation in the observed data. Quantile matching estimation was then used to refine the model fit, using the extreme upper quantile probability for crosswind less than 25kt (0.999549). Initially the lower quantile used was the mirror of the upper quantile (i.e. 1-0.999549) and this gave between 2 to 11 values greater than 35kt (95% CI).

The extreme quantiles resulted in a poorly modelled central distribution, and while this central region is of less importance in risk assessment, it is worth attempting to model it better where possible, hence a new lower quantile of 0.5 was used with the existing upper quantile (0.999549). This resulted in 0 to 8 simulated values (95% CI) greater than 35kt, which is considered acceptable.

The best resulting parameters for the gamma distribution were therefore determined as:

Parameters (gamma):

	estimate
shape	1.9682606
rate	0.4017006

Figure 27 below is the graphical output from *fitdistrplus* showing the model fit.

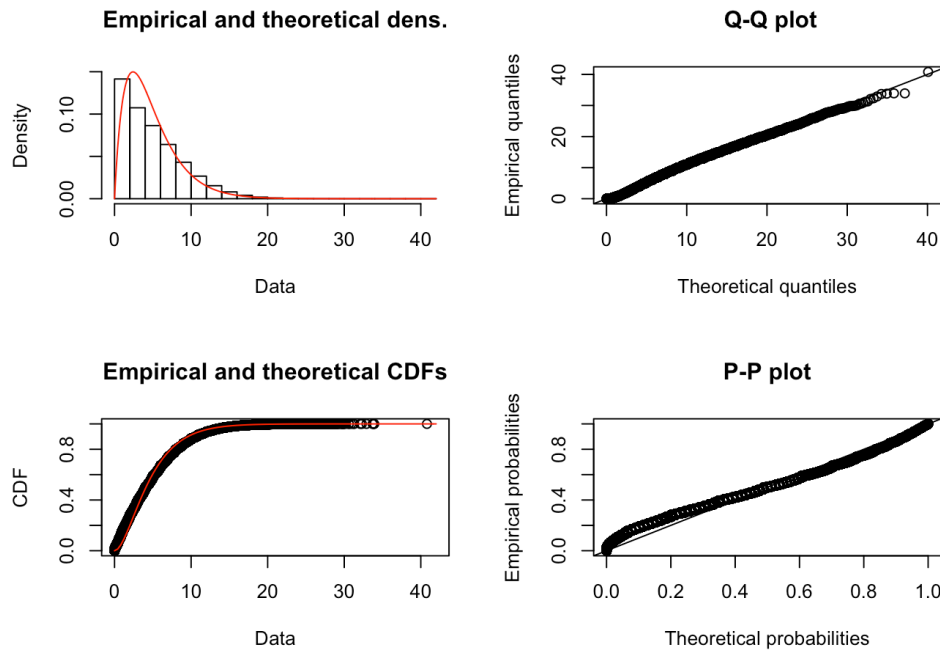


Figure 27: *fitdistrplus* graphical output for fitted gamma distribution using matching quantile estimation with revised probabilities

The model provides a probability of crosswinds > 10kt, 20kt and 33kt (max demonstrated) as 0.0869, 0.00277 and 0.0000232 respectively. For the purposes of the BN, the probability for 10kt crosswind will be used, which is the maximum allowable crosswind for contaminated runways. While the threshold choice is somewhat arbitrary, 10kt seems reasonable given the airline’s standard procedures state that 22kt is the absolute limit for co-pilot landings i.e. 10kt is not insignificant, but also not very extreme.

Crosswind Yes 0.0869

No	0.9131
----	--------

Table 20: NPT for crosswind

See Appendix 10 for R code used to model the crosswind data.

4.6.4.3.4 Inaccurate information to crew

Similarly to 4.6.4.3.1 Crew performance inaccurate, whether pilots have been given incorrect information cannot be fully determined from flight data and it would be extremely difficult to model all potential sources of inaccurate information available to pilots. However, it is possible to compare METAR wind observations with the wind recorded on board the aircraft for each landing. This comparison showed that 2.248% of METAR crosswind components were >10kt different to that recorded on board the aircraft.

Inaccurate information *Yes* 0.0225

No	0.9775
----	--------

Table 21: NPT for Inaccurate information

4.6.4.3.5 *Technical issues: landing gear and nosewheel steering*

Some technical issues might be evident in flight data, however the best source of this information would be an operator's own defect data. For the purposes of this BN, an estimated probability of 1 in 100,000 is used for landing gear and nose wheel steering defects.

Tech: Gear *Yes* 1e-5

No	0.99999
----	---------

Tech: NWS *Yes* 1e-5

No	0.99999
----	---------

Table 22: NPTs for technical issues

These probabilities, based on estimates, will influence the overall probability of lateral deviation derived from the Bayesian network, but they will have no effect on the probability of lateral deviation at individual airports, as calculated later, because they are not airport dependent. However, if the Bayesian network was used to compare different aircraft types, or perhaps different maintenance bases, these probabilities could be changed to reflect those differences and these technical issue nodes will be more useful.

4.6.4.3.6 *Gusting wind*

The METAR observations can be used for this node. From the empirical data (Western Europe operation), gusting conditions were reported for 2.777% of landings.

Gust *Yes* 0.0278

No	0.9722
----	--------

Table 23: NPT for gusting wind

4.6.4.3.7 Asymmetric thrust

Table 24 shows the frequency of periods of asymmetric thrust identified in the observed data.

<i>Time (s)</i>	<i>Frequency</i>
0	308786
1	684
2	656
3	209
4	46
5	20
6	18
7	10
8	5
9	3
10	1
11	2
12	0
13	0
14	1
15	1
16	0
17 or more	6

Table 24: Frequency table for time with asymmetric thrust

The period of measurement is 5 seconds before touchdown to the end of the landing rollout. The six landings which had 17 seconds or more of asymmetric thrust are landings with engine technical issues (i.e. shutdown or partially shutdown).

This data is difficult to model as it is highly skewed and truncated at 17 seconds, so rather than attempt to model it, an empirical probability will be used. A one or two second period of asymmetry may not be significant, however a period of five or more seconds seems more problematic. From Table 24, the probability of a period of five or more seconds is 0.000216.

Asym thrust Yes 0.0002

No	0.9998
----	--------

Table 25: NPT for asymmetric thrust at landing

4.6.4.3.8 Unstable approach

Fast and high are the two most commonly cited causal factors for unstable approaches in landing veer-off events, hence they are used as the parent nodes for *Unstable approach*. Other unstable approach causal factors could be added in future as required.

Speed

A histogram for recorded CAS – Vapp at 50ft is shown in Figure 28.

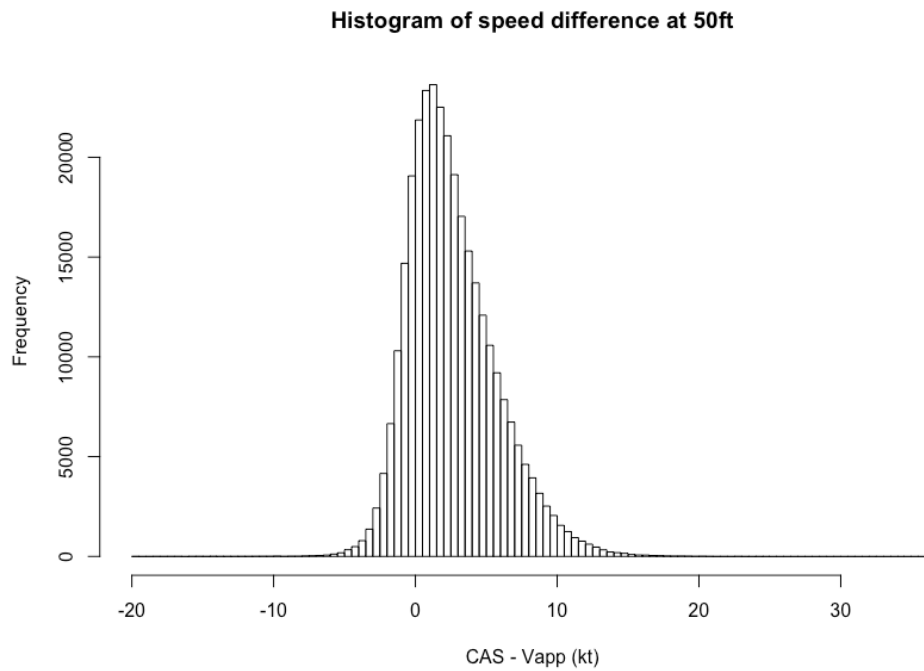


Figure 28: Histogram for CAS - Vapp at 50ft

The summary statistics for the observed data are:

```
min: -19.9   max: 36.45
median: 2.1125
mean: 2.615654
estimated sd: 3.117802
estimated skewness: 0.7312505
estimated kurtosis: 4.460254
```

The shape of the histogram suggests a Gumbel distribution and once again, the *fitdistrplus* package in R was used to find the distribution parameters:

```
Parameters (Gumbel):
      estimate
location 1.238794
scale    2.505402
```

The associated graphical output is shown below.

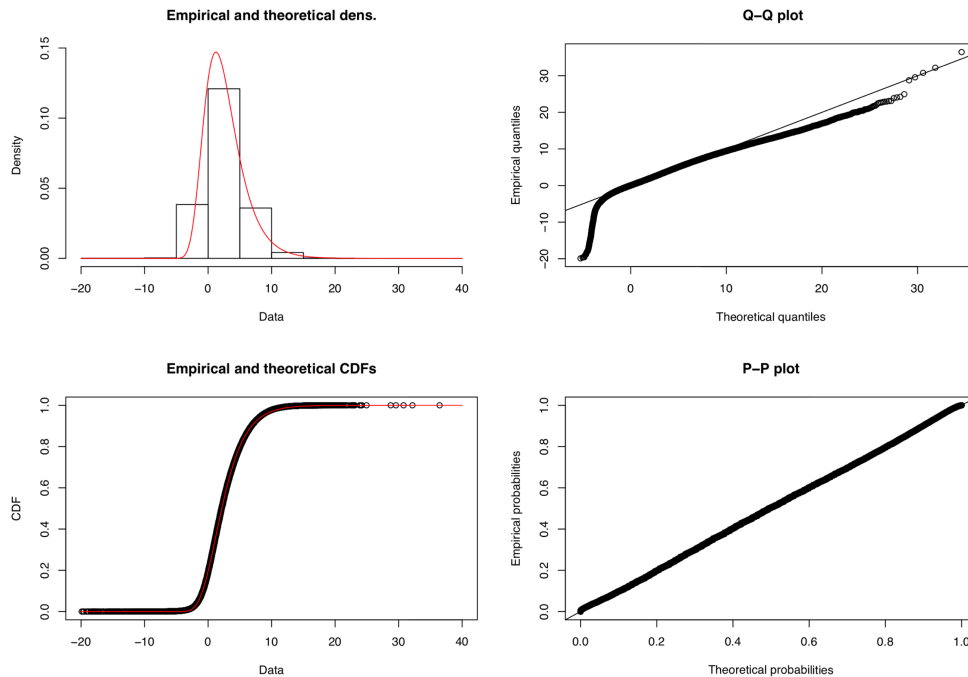


Figure 29: fitdistrplus graphical output for Gumbel distribution

The distribution models the extreme values fairly well. From the accidents analysed as part of Post (2015), high, rather than low, speed is more commonly associated with veer-off events, so the deviation for low quantiles is acceptable.

Comparing this model to the empirical data, the model would predict 15 to 33 values >25kt (derived by bootstrapping model distribution, 1000 iterations, 95% credible interval), whereas the actual value was 5. The model estimate is a little high, however that is acceptable for purposes of risk assessment, where a slight overestimate is prudent and more preferable than an underestimate.

Being fast by 20kt is generally considered to be unstable (Moriarty and Jarvis, 2014), hence the NPT for airspeed > 20kt fast is given below.

Speed > 20kt fast Yes 0.0006

No	0.9994
----	--------

Table 26: NPT for airspeed vs target >20kt

See Appendix 11 for R code used to model speed.

Glideslope

A histogram of the observed data for glideslope deviation at 150ft is shown in Figure 30 below.

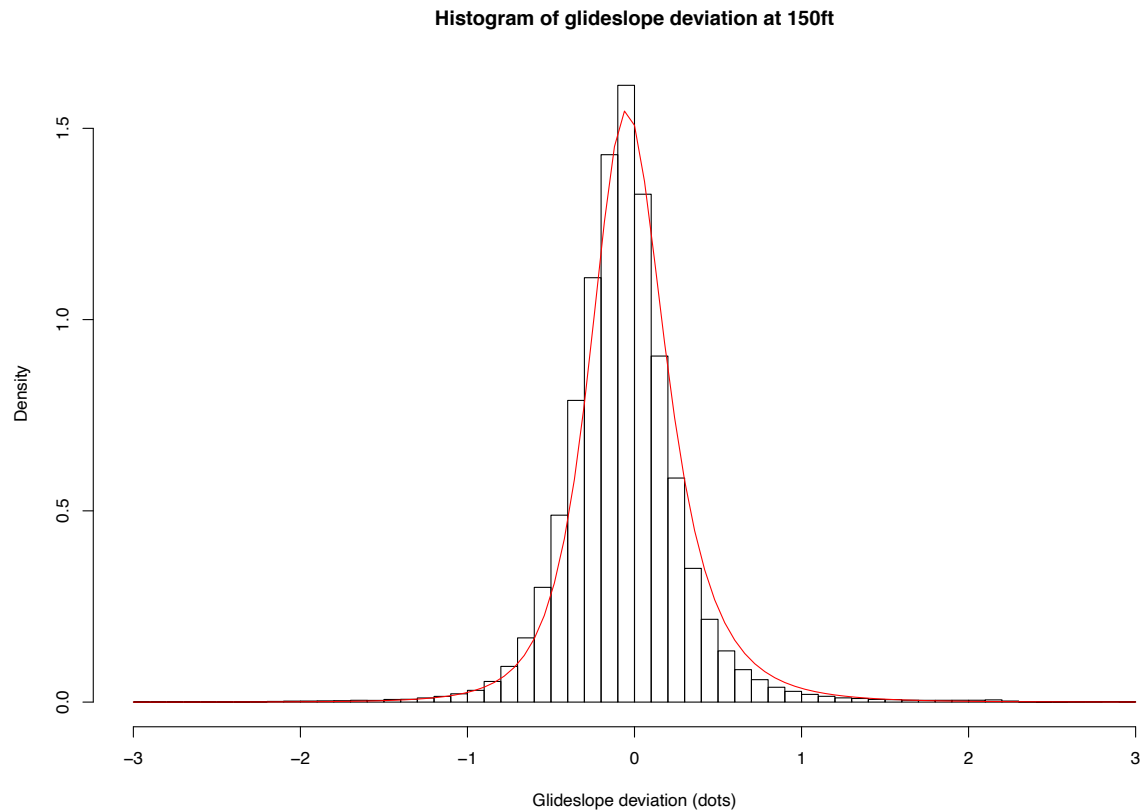


Figure 30: Histogram of glideslope deviation at 150ft with Johnson distribution in red

The summary statistics for the observed data are:

```
min: -2.97  max: 3
median: -0.05
mean: -0.04493792
estimated sd: 0.3603182
estimated skewness: 0.6726488
estimated kurtosis: 10.91812
```

Although the data appears to be normally distributed, it is characterised by high kurtosis and thin tails, making the normal distribution inappropriate. The Johnson distributions (Johnson, 1949) are transformations of the normal distribution and can offer greater flexibility in modelling empirical distributions than many of the more standard distributions (Cullen and Frey, 1999).

Using the *SuppDists* package (Wheeler, 2016) in R, a Johnson distribution was used to model the data, with parameters:

Parameters (Johnson [*JohnsonFit(data, moment = "find")*]):

```

gamma    -0.204612
delta    1.295692
xi        -0.08031886
lambda    0.3312998
type      SU

```

The Q-Q plot in Figure 31 shows that data sampled from the model distribution is acceptably similar to the observed data.

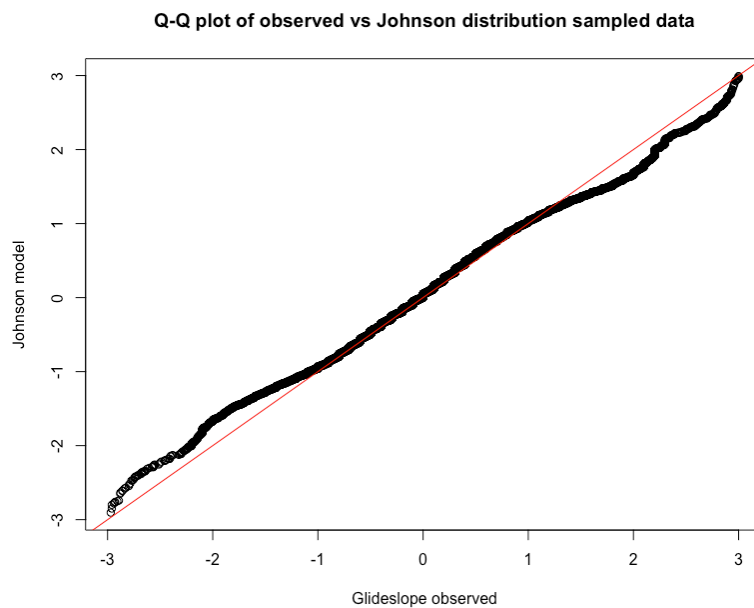


Figure 31: Q-Q plot for modelled glideslope data

Generally, > 1 dot high on the glideslope is considered unstable (Flight Safety Foundation, 2009c). Using the modelled probability for glideslope > 1 dot, the NPT is given below.

Glideslope > 1 dot Yes 0.0121

No 0.9879

Table 27: NPT for glideslope

See Appendix 12 for R code used to model glideslope.

Unstable approach NPT

The approach is unstable if either or both *Glideslope* and *Speed* are positive, therefore the NPT for *Unstable approach* is as follows:

<i>Speed >20kt</i>	<i>No</i>		<i>Yes</i>	
<i>Glide > 1dot</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>
<i>Unstable</i>	0	1	1	1
<i>Not unstable</i>	1	0	0	0

Table 28: NPT for Unstable approach

4.6.4.3.9 Hard landing

Figure 32 shows a histogram of the observed values of normal acceleration at landing.

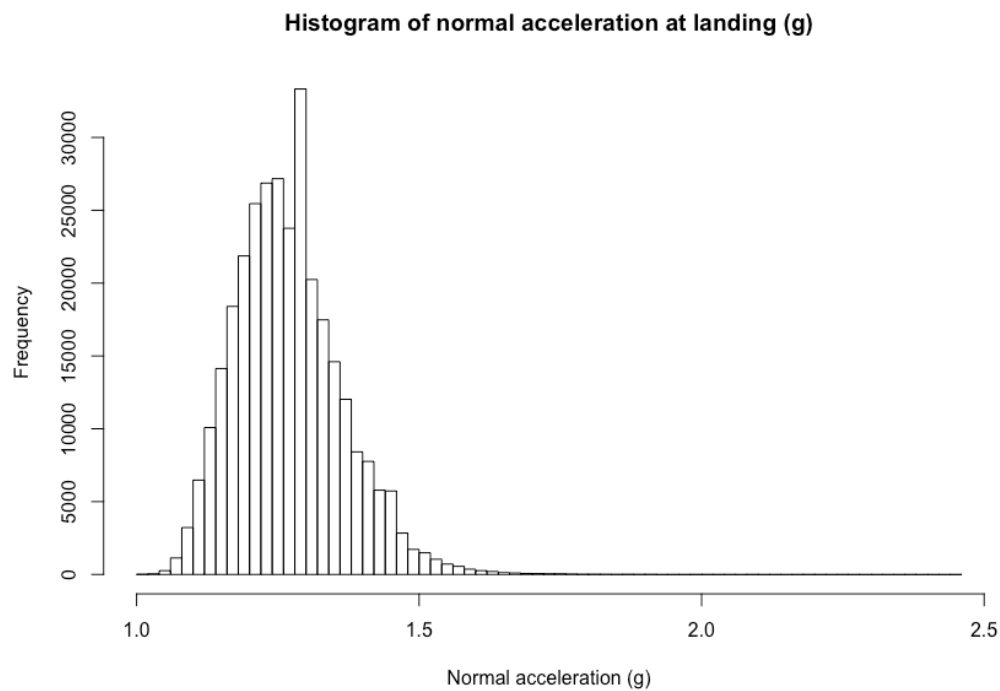


Figure 32: Histogram of observed normal acceleration at landing (g)

The summary statistics for the observed data are:

```

min:  1    max:  2.45
median: 1.27
mean:  1.272884
estimated sd: 0.0952111
estimated skewness: 0.7551227
estimated kurtosis: 4.620904

```

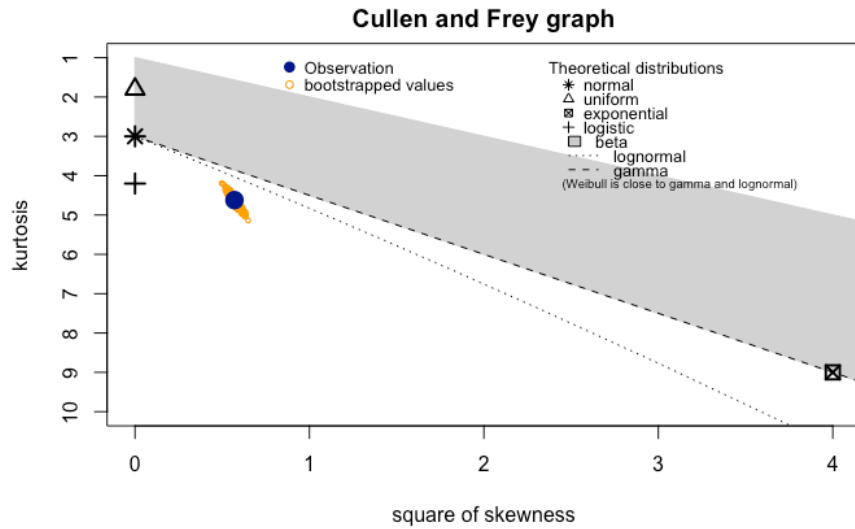


Figure 33: Cullen & Frey graph for normal acceleration at touchdown

The Cullen & Frey graph (Figure 33) suggests that a lognormal or gamma distribution might be candidates, so the gamma distribution was tried first using quantile matching estimation, with quantile probabilities of 0.3 and 0.99 used. The tails of the gamma distribution were found to be too thin and therefore underestimated the frequency of the more extreme values (Figure 34).

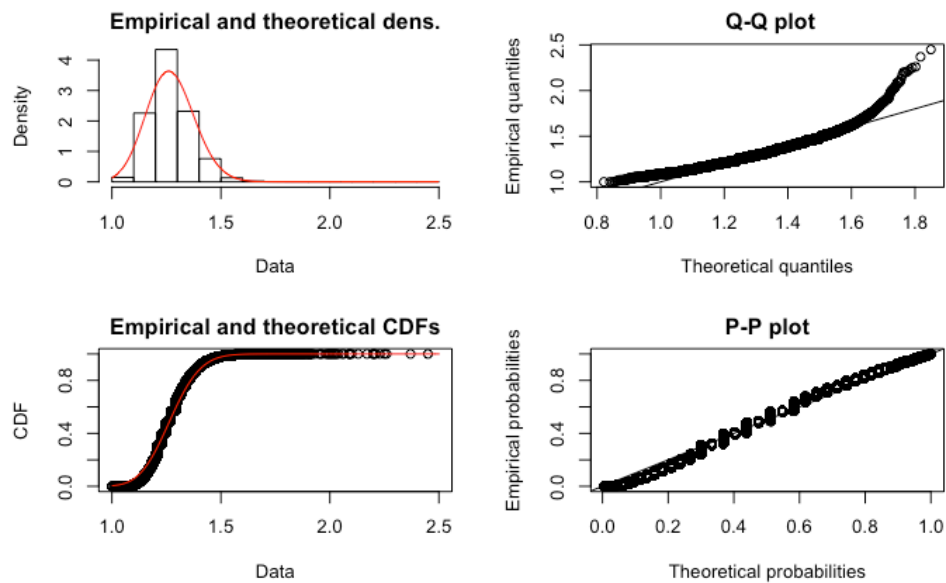


Figure 34: Gamma distribution fit plot for normal acceleration at touchdown

Modifying the quantile probabilities to give more weighting to the most extreme values, the quantile probability representing $>2.1g$ at landing (0.9999646) was tried and found to be a poor fit Figure 35.

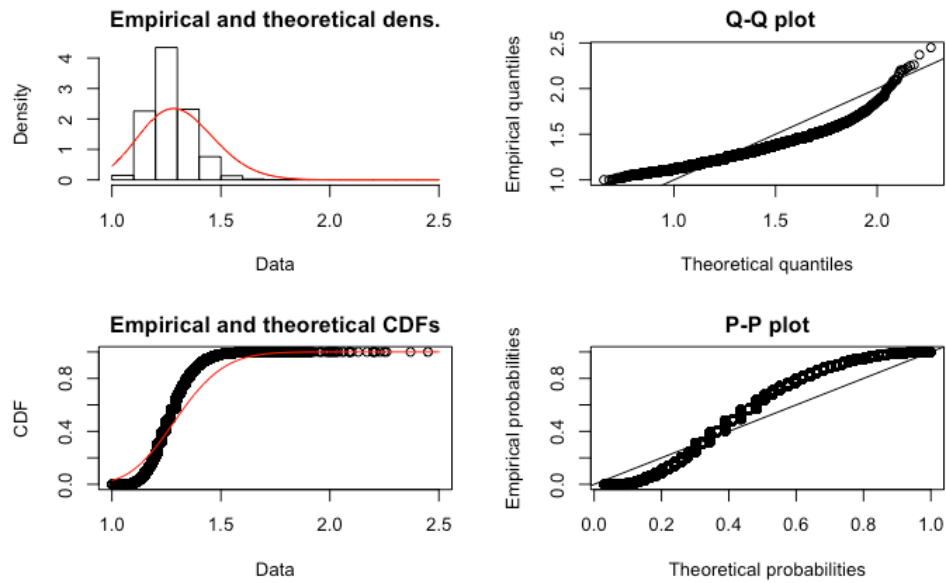


Figure 35: Second attempt at gamma distribution using extreme upper quantile

A lognormal distribution fitted the upper quantiles poorly, similarly to the gamma distribution in Figure 34.

A Gumbel distribution was then tried using *fitdistrplus* and quantile matching (quantiles at 1.3 and 2.1g), which resulted in parameters:

```
Parameters (Gumbel):
      estimate
location 1.225838
scale    0.085324
```

The resulting output is shown below.

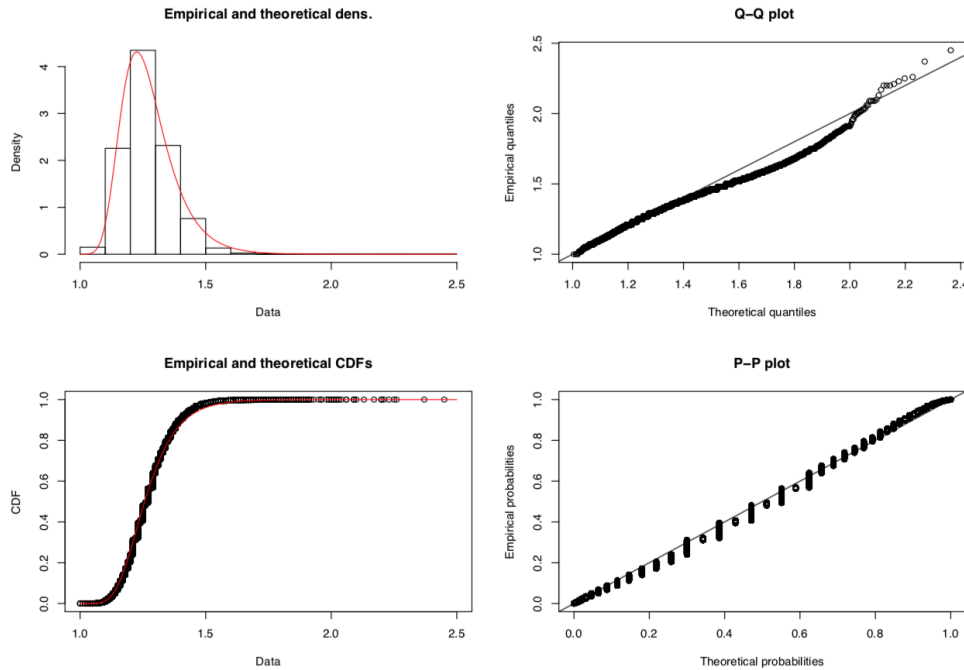


Figure 36: fitdistrplus output for Gumbel model

The Gumbel distribution is also known as the extreme value distribution and is often used to model maximum values (Cullen and Frey, 1999). The observations here, normal acceleration at landing, are the maximum recorded values of normal acceleration at landing, so perhaps it is no surprise that the Gumbel distribution provides a good fit.

In the observed data there were 11 landings $>2.1g$; the model estimates between 5 and 18 (95% CI). $2.1g$ is a suitable threshold because this was the value the operator used as an “alert” level event threshold in their FDM program for the A320 series fleet i.e. values exceeding $2.1g$ would require further investigation to determine whether maintenance checks should be carried out.

Based on the model, the NPT for hard landing is as follows:

Hard landing $> 2.1g$ Yes $3.5526e-5$

No	0.99996
----	---------

Table 29: NPT for hard landing

See Appendix 13 for R code used for modelling normal acceleration at landing.

4.6.4.3.10 Poor visibility

METAR visibility reports were available for 306,017 flights. The visibility is reported in meters or kilometres, with visibilities of 5km or more usually expressed in whole kilometres. Table 30 summarises the frequency of reported visibility distances.

<i>METAR viz (km)</i>	<i>Frequency</i>
<1	1732
1 to 2	1717
2 to 3	3321
3 to 4	2,999
4 to 5	4,724
5	5,524
6	7,153
7	7,135
8	10,150
9	7,469
10 or more	254,093
<i>Total</i>	306,017

Table 30: METAR visibility reports

ICAO Annex 2 *Rules of the Air* (International Civil Aviation Organisation, 2005) states that visual flight rules (VFR) can apply down to a visibility of 5km, so taking this as the threshold for poor visibility, the probability of visibility <5km is 4.74% (i.e. 14,493/306,017).

Poor viz Yes 0.0474

No	0.9526
----	--------

Table 31: NPT for poor visibility

4.6.4.3.11 Lateral deviation (veer-off)

IATA's 2016 report (International Air Transport Association (IATA), 2017) states a runway excursion rate of 0.3 per million flights. According to the Flight Safety Foundation (Flight Safety Foundation, 2009a), 79% of excursions are during landing and 53% of these are veer-off events. Multiplying these values together gives the runway veer-off at landing rate, 1.2561e-7 and this can be used with the frequencies of causal factors to represent the node probabilities for lateral deviation.

In Bayesian networks where a child node has many parents, the NPTs can become large very quickly. For the network described here, all of the nodes are Boolean, having just two states (yes/no, true/false), however even so, the number of entries in the NPT for *Lateral Deviation* would be 2^{11} (2048) because there are eleven parent nodes. Fortunately, there is an alternative to laboriously entering 2048 values into a NPT: the *noisy-or* function.

The *noisy-or* function reduces the number of entries to just n , rather than 2^n , so 11 in this case. The values for the *noisy-or* have been calculated by multiplying the veer-off at landing rate (1.2561e-7) by the frequencies of the relevant causal factors shown in Table 5.

Table 5 only includes causal factors which have a frequency of > 5%, so it is necessary to add a *leak factor* (see 3.2.2.2.1) to the *noisy-or* expression to account for the other causal factors not modelled in the Bayesian network. The *leak factor* has been calculated by multiplying the combined frequency of the remaining (i.e. <5% frequency) causal factors with the veer-off at landing rate ($1.2561e-7$).

AgenaRisk (Agena Ltd, 2016) is a software program commonly used to create and run Bayesian networks and has an inbuilt *noisy-or* function.

The *noisy-or* expression entered into AgenaRisk for the Lateral Deviation node is:

```
noisyor(unstable, 1.979e-06, landg, 2.475e-04, viz, 1.855e-07, asym, 3.140e-05, wet, 3.398e-07,
xwind, 3.614e-07, gust, 4.970e-07, tech_gear, 2.387e-03, tech_nws, 1.507e-03, heavy_p, 4.433e-06,
innac_info, 1.396e-06, 2.085024e-08 )
```

Pearl (1988) fully describes *noisy-or*.

4.7 Bayesian network learnt from flight data

The methods described in chapter 4.6 and chapter 5 rely heavily on research conducted into previous accidents. Accident investigation reports are usually a reliable source of information on causal factors as they are normally produced by government bodies, following international protocols, and investigators are usually well-trained experts in their field. However, the quality of reports is not entirely consistent across different investigation bodies and there can be significant differences in the methodologies and models used in the investigation process. Whilst efforts are made to avoid bias, it is inevitable that bias will occur in investigations due, amongst other things, to the organisational culture of the investigating body and the previous experiences of the investigators involved. The question therefore arises, as to whether a Bayesian network predicated entirely on flight data (and some meteorological observations) could yield useful results, whilst avoiding any biases which may be present in the causal factor literature. This chapter explores this further.

To be of practical use, any resulting network would have to fulfil the two most important functions of the network previously described:

- Provide relative probabilities for different scenarios (e.g. crosswind and fast approach, poor visibility and wet runway etc)
- Provide relative probabilities for different airports

A problem with causal networks based on investigation reports, is that the causal factors cited in do not necessarily have equal strength in terms of leading to the accident outcome. For example, in the previous network it is quite obvious how a crosswind could contribute to lateral deviation, however the contribution of “inaccurate information to crew” is more abstract. Inaccurate information will not *cause* a lateral deviation, but its presence could contribute to the pilots not being prepared for the presence of another, more acute causal factor. This interaction between causal factors is not explored in the causal factor literature; instead simple counts of the presence of individual causal factors are provided.

A possible solution to these problems is to take a more naïve approach and not assume the network structure is simply all the causal factors directed towards the outcome. Instead, the data itself may be able to inform the structure of the network. Methods to discover network structure from data have been around since the early 1990s (Buntine, 1991; Cooper and Herskovits, 1992; Spiegelhalter et al., 1993), and generally the most popular methods use constraint-based or score-based algorithms. Constraint-based algorithms use conditional independence tests to discover which variables should be connected and those which should

not. Examples are the PC, Grow-shrink and Incremental Association algorithms. Score-based algorithms attempt to find the network which has the highest probability, given the data. Examples are hill-climbing and simulated annealing. A proper and full description of constraint-based and score-based algorithms can be found in Darwiche (2009) and Scutari and Denis (2015).

4.8 Data preparation for learning Bayesian network structure

From the 310,448 flights available, a total of 106 features were extracted for each flight. The majority of these features have not been used, however they were extracted in case they could be of later use in future research. All of the data and all of the features could theoretically be used to learn a Bayesian network structure, however the processing time would probably make that impractical and the resulting network would likely be overly complex and difficult to interpret. It is possible that dependencies would be drawn between completely unrelated variables due to the size of the dataset. A more sensible approach is to use a subset of those features, choosing those which are believed to be directly related to runway veer-off.

It was established in the previous chapter that the standard deviation of lateral acceleration could be used as a substitute for actual lateral deviation in the absence of any better parameters. However, there is an issue with this variable. In a scenario where an aircraft is landing on a slippery runway, lateral accelerations might be minimised due to the lack of grip between the aircraft tyres and runway surface. The pilot could be making inputs to correct a deviation, but the aircraft could, in some circumstances, not fully respond to those inputs, hence in some, albeit fairly rare cases, the standard deviation of lateral acceleration (*LATGs_d*) might underrepresent lateral deviation.

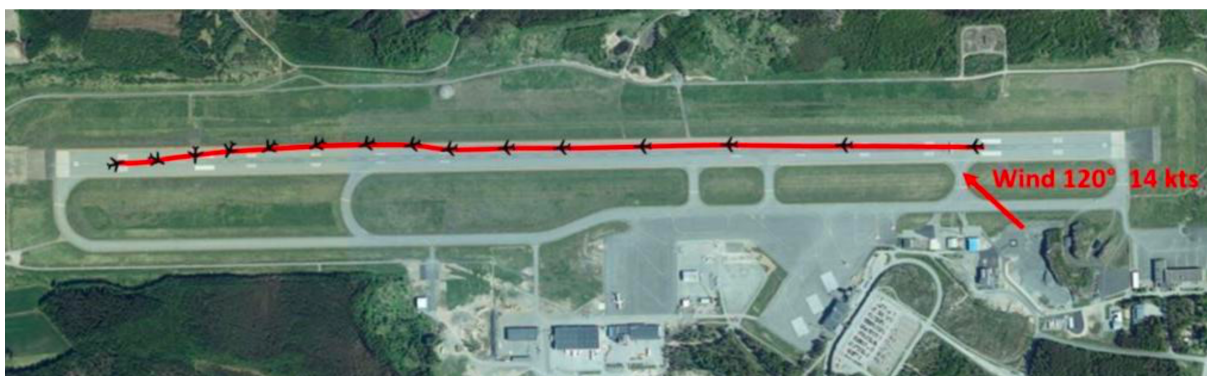


Figure 37: Track of CRJ900 landing at Turku Airport, 25th October 2017

An example is shown in Figure 37. The diagram shows the reconstructed track and heading of an aircraft during the landing roll on a contaminated runway (Safety Investigation Authority

Finland, 2018). The aircraft touched down hard (1.95g) and faster than usual (151kt) with a 10kt tailwind component and a slight crosswind. The pilots made large rudder inputs to try to correct the track of the aircraft, however due to a loss of grip the aircraft continued towards the edge of the runway, despite the aircraft's heading being towards the centreline. In this scenario, there would not be large changes to *LATGsd* due to the lack of grip, however there were large, full deflection rudder inputs. Therefore an additional feature will be included in the learning dataset; the standard deviation of rudder deflection during the landing roll.

In total 12 features were selected from the dataset to explore whether a meaningful and useful Bayesian network could be learnt from the data. They were:

- Recorded crosswind component at landing – *rLandXwind*
- Runway wet, derived from METAR reports – *runWet*
- Difference from target airspeed at 50ft radio height (indicative of an unstable approach) – *spdDelta50*
- Maximum recorded normal acceleration during landing – *maxG*
- Heading deviation during landing roll – *HDGdev*
- Glideslope deviation at 150ft (indicative of an unstable approach) – *GSlope150*
- Standard deviation of lateral acceleration during the landing roll – *LATGsd*
- Duration of asymmetric thrust from touchdown -5 seconds to the end of the landing roll – *asymTHR*
- Gusting winds, derived from METAR reports – *metXgust*
- Visibility, derived from METAR reports – *metVizkm*
- Airport ID – *arrid*
- Standard deviation of rudder deflection during landing roll – *RUDsd*

In preparation for network learning, the original crosswind component (wind from left = negative, wind from right = positive) was converted to absolute values because only the magnitude of the crosswind is of importance to this research. The direction could become important if the consequences of a lateral deviation were to be considered in the network later, especially if, for example, landings to Funchal, Madeira or Trabzon, Turkey were being assessed. Both of these airports have sea to one side of the runway.

The gust variable was transformed into a categorical true or false, rather than a component crosswind gust. Gusting winds are only reported if the gust exceeds the mean reported wind

speed from the previous 10 minutes by 10kt or more. If this condition is not met, no gust is reported, which results in a missing value in the dataset. Therefore, missing values were transformed to “False” and any value of gust changed to “True”.

Once these changes had been made, any incomplete records were omitted from the dataset, thus avoiding potential problems with network learning algorithms which cannot cope with missing values. The resulting dataset of 287,039 (92%) from an original 310,448 flights was then output from R to a CSV file, ready for import into GeNIe.

4.8.1 Learning network algorithms in GeNIe

GeNIe has six built-in algorithms for learning Bayesian networks from data. They are:

- **Bayesian Search:** This algorithm is a hill climbing procedure with random restarts, introduced by Cooper and Herskovits (1992).
- **PC:** This algorithm uses “*independences observed in the data to infer the structure that has generated them*” (BayesFusion LLC, 2019) and was introduced by Spirtes, Glymour and Scheines (1993). The PC algorithm is compatible with continuous data as long as the data conforms approximately to the Normal distribution.
- **Greedy Thick Thinning:** Described by Cheng, Bell and Liu (1997), the algorithm is based on the Bayesian Search approach, and repeatedly adds arcs (thickening) between nodes and then removes them (thinning) to maximise the marginal likelihood of the observed data in each phase. The algorithm stops when no arc addition or removal results in an increase in the likelihood.
- **Tree Augmented Naïve Bayes:** Also based on the Bayesian Search algorithm and uses a class variable which is the parent of all other variables as a starting point. It then adds connections between other variables where possible dependencies are found. A full description can be found in Friedman, Geiger and Goldszmidt (1997). The algorithm “...imposes [a] limit of only one additional parent of every feature variable (additional to the class variable, which is a parent of every feature variable)” (BayesFusion LLC, 2019).
- **Augmented Naïve Bayes:** Similar to Tree Augmented Naïve Bayes, however the algorithm imposes no limit on the number of connections entering each feature variable. The algorithm is described in full in Friedman, Geiger and Goldszmidt (1997).
- **Naïve Bayes:** This is a simple structure where it is assumed that the only connections are directed arcs from a class variable to the feature variables, with no other connections between nodes.

4.8.2 Discretisation of variables

Only the PC algorithm is capable of learning from continuous variables and none of them is capable of mixing discrete and continuous variables. The 12 features extracted from the dataset are a mixture of continuous and discrete variables, therefore the continuous variables must be discretised. There are various methods for discretising variables, some of which are focussed on ensuring extreme values are properly represented for risk assessment purposes (Zwirgmaier and Straub, 2016). In GeNIe, three methods are implemented:

- *Uniform widths* splits the variable data into equally sized intervals. If the data is not distributed uniformly, the intervals will have different counts of observations.
- *Uniform counts* splits the range of the variable at intervals so that each bin has the same number of observations.
- *Hierarchical* is an unsupervised method related to clustering, details of which can be found in the GeNIe documentation (BayesFusion LLC, 2019).

A powerful feature of GeNIe is that the number of bins and their boundaries can be easily modified by the user.

From the 12 variables in the dataset, 8 are continuous and require discretising. They are:

- *rLandXwind* – discretised using Uniform Counts into four bins
 - *speedDelta50* - discretised using Uniform Counts into four bins
 - *maxG*- discretised using Uniform Counts into four bins
 - *HDGdev*- discretised using Uniform Counts into four bins
 - *GSlope150* - discretised using Uniform Counts into four bins initially, however due to the high kurtosis of the empirical distribution, the threshold for the highest bin led to the inclusion of observations which were not significantly different from the mean. Therefore the threshold was manually moved to the right to 0.225 to mirror the lower bin threshold (-0.225). Even after this change, the bin still contained a good number of observations (43,553).
 - *LATGsd* - discretised using Uniform Counts into four bins
 - *metVizkm* – due to the high skewness of this variable, Uniform Counts resulted in the data being divided into just two bins. Uniform Widths was used instead, splitting the data into four bins with thresholds at 2.5km, 5km and 7.5km.
- RUDsd* - discretised using Uniform Counts into four bins

5 Results

The Bayesian network structure was created in AgenaRisk and the probabilities derived above were entered into the NPTs.

Based on the modelled data from over 300,000 flights and the background global rate of veer-off events ($1.2561\text{E-}07$), the network calculated an overall probability of a veer-off, for the subject airline's operation, of $2.2740\text{E-}07$, which is nearly twice the worldwide rate of runway veer-offs. A single probability value like this is of little use on its own and it is very difficult, if not impossible, to validate. It might be that the estimated probabilities for technical problems are too high, or that the threshold for what constitutes, for example, a hard landing, are set too low, both of which would inflate the veer-off probability. However, the baseline figure is not entirely useless, as it can be used as a safety performance indicator to track performance over time. The baseline will change as the probability of the causal factors change, so for example, if data from the airline's FDM programme suggests unstable approaches are becoming more prevalent and there are more defects occurring to landing gear, the resulting baseline will change. This change will be quantifiable and based on real and objective data and is one of the contributions of this research.

The actual numerical value of the baseline is not of critical importance, but ideally it should be of the right order of magnitude to ensure the analyst still has oversight of the actual magnitude of the risk. For example, if a set of conditions result in a 100% increase for a risk with a baseline probability of $1\text{E-}08$, the resulting risk is still extremely unlikely at $2\text{E-}08$.

The real utility of this Bayesian network is in allowing relative probabilities of different scenarios to be compared.

5.1.1 Scenarios

A powerful feature of Bayesian networks is that a risk probability can be calculated for different scenarios. Any node in the network can be set to *TRUE* or *FALSE* to indicate that the causal factor is definitely present or not present. Several scenarios were created in the network by setting causal factor probabilities to 1 i.e. true or present. These calculations can be made for single causal factors, or combinations of causal factors.

Table 32 below shows results from seven different scenarios.

Factors present	P(Lateral Deviation)	Increase
<i>Baseline probability, no evidence added</i>	2.27E-07	
<i>Wet</i>	5.99E-07	x 2.64
<i>Wet + Xwind</i>	9.29E-07	x 4.09
<i>Wet + Xwind + Gust</i>	1.41E-06	x 6.21
<i>Unstable</i>	2.18E-06	x 9.60
<i>Unstable + Xwind</i>	2.51E-06	x 11.05
<i>Unstable + Xwind + Gust + Wet</i>	3.37E-06	x 14.80

Table 32: Effect on probability of risk factors

The top row of the table shows the probability for lateral deviation, based on the prevailing causal factor probabilities, modelled and determined from flight data and other sources. The remaining rows reflect the lateral deviation probability once evidence has been added to the network e.g. “Wet” runway set to *TRUE*.

As can be seen from the table, landing from an unstable approach increases the probability of a veer-off event nearly ten-fold, albeit that the risk is still remote. Likewise, landing on a wet runway from an unstable approach in a gusty crosswind represents a near fifteen-fold increase in lateral deviation probability.

Being able to run different scenarios in this way allows an airline to make risk-based operational decisions and also allows the quantification of benefit in eliminating undesirable practices, such as landing from unstable approaches. This is not possible using current methods of airline risk assessment and is a contribution of this work.

Further benefits can be derived from tailoring the BN to specific airports.

5.1.2 Comparing risk between airports

A fundamental aspect of an airline’s overall risk picture is understanding which airports present the most significant risk. This has implications for pilot training, airfield briefings and emergency response preparedness.

To demonstrate how Bayesian networks can differentiate the risk at different airports, five airports were chosen for comparison:

- Belfast City (BHD)
- Dalaman (DLM)
- Moscow Domodedovo (DME)
- Edinburgh (EDI)
- London Heathrow (LHR)

These airports were chosen because flight data from a good number of flights (at least 600) was available for each of them, and also because the airports have different features and characteristics. BHD has a relatively short runway; DLM has generally fine weather and visual approaches are common; DME is a major airport and can experience low temperatures and contaminated runways; EDI is a medium sized airport with sometimes windy weather; and LHR is a major, busy airport, where nearly all landings are from precision approaches to long runways.

5.1.2.1 Modelling nodes for specific airports

The process described in 4.6.4 was repeated for each of the five airports, thereby providing the Boolean probabilities for each Bayesian network node, for each airport. Appendix 13 provides an example of the R code used for modelling *Speed* at BHD. The five networks were then used to calculate the overall baseline probability of lateral deviation and the results are shown below.

<i>Airport</i>	<i>BN probability</i>
<i>DME</i>	2.11E-07
<i>BHD</i>	2.27E-07
<i>EDI</i>	2.27E-07
<i>LHR</i>	2.27E-07
<i>DLM</i>	2.28E-07

Table 33: BN probabilities for lateral deviation at different airports

The values are very similar, as expected, with Dalaman marginally having the highest probability, a reflection of it being at the top of the list of airports where inaccurate wind information is provided (see 4.6.4.3.4), and for unstable approaches. The difference in probabilities between the top and bottom of the table is around 8%. It is worth remembering that these values are influenced not only by weather and runway conditions at these airports, but also the airline's safety performance with regard to unstable approaches at each airport.

Being able to rank airports using a quantitative risk assessment of a common accident category is not possible using current methods, so the method described here is a contribution of this work.

5.2 Validation

Estimating and validating probabilities for rare events can be extremely difficult (Hubbard, 2012; Taleb, 2007), particularly when, as is the case here, the data available does not contain examples of the event being estimated. Some research has relied on subject matter experts to review the resulting model output (Bareither and Luxhøj, 2007), however it would always be preferable to validate the model using some form of related, quantifiable measure if available.

None of the 300,000 plus landings in the dataset actually resulted in a veer-off event, therefore another measure of excessive lateral deviation must be identified to validate the method.

The obvious measure would be localiser deviation. The localiser provides lateral guidance to pilots during the approach and landing phase and the deviation from the correct track is presented in units of “dots”. Three dots is usually the maximum displayed deviation. The deviation is based on an angular displacement, with the localiser transmitter located at the far end of the runway to provide guidance all the way to landing and during the landing roll in low visibility operations. Because it is an angular displacement, the actual deviation distance represented by 1 dot reduces as the aircraft approaches the transmitter.

Through a combination of localiser deviation and knowing the distance from the localiser transmitter, basic geometry can be used to calculate the lateral deviation from the runway centreline. The lateral deviation could then be used to identify which landings experienced excessive deviation, thereby helping to validate the probabilities calculated by the BNs.

However, not every runway has a localiser, and, in the data available, the localiser deviation values are too coarse (resolution 0.1 dot) and recorded too infrequently (0.5Hz on two channels, giving 1Hz combined) to be of use. Another obvious measure of deviation would be GPS location, however the resolution (3 decimal places) of the recorded GPS position is also far too coarse to be used. Three decimal places is roughly 100m resolution, which is insufficient given that a standard runway width is 45m.

Instead, the standard deviation of lateral acceleration ($LATGsd$) (recorded at 4Hz) for the 12 second period following touchdown is used. Whilst not a direct measure of deviation, the standard deviation of lateral acceleration is an indicator of the range of acceleration values and will probably, in most cases, indicate landings during which deviations have occurred, requiring corrective directional changes.

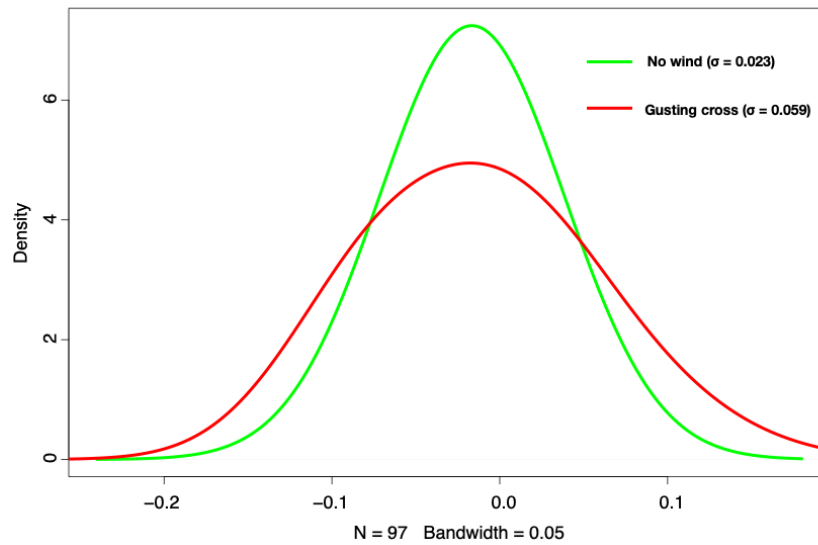


Figure 38: Comparison of density distributions for lateral acceleration in calm conditions and crosswind conditions

To demonstrate the effect of crosswind on lateral acceleration, Figure 38 shows the density distributions for lateral acceleration during the landing rollout for two landings. The green distribution is a landing in calm wind conditions (standard deviation = 0.023) and the red distribution is a landing in crosswind conditions (standard deviation = 0.059).

It could be argued that lateral accelerations do not equate to deviation, for instance, a pilot could be applying abrupt short inputs resulting in short duration accelerations in either direction, but the resulting track of the aircraft could remain close to the centreline. However, these short abrupt inputs would probably be indicative of control difficulties during the landing roll and therefore lateral acceleration would still be an indicator of potential lateral deviation.

5.2.1 High-risk flights and LATGsd

The Bayesian network calculates higher probabilities of runway veer-off if causal factors are present. It should follow that flights with causal factors present experience more lateral deviation. In other words, the hypothesis is:

Flights which have an elevated BN calculated probability for veer-off, have higher LATGsd.

To establish if this is true, the data for the year 2011 (23,726) flights was filtered so that only flights arriving at BHD, DME, DLM EDI and LHR remained, leaving 13,407 flights, which is a sufficient number for validation and can be processed quite quickly. Flights having at least one veer-off causal factor present were then split from the rest of the flights, leaving two groups:

- BN High-risk flights (n = 1915) i.e. one or more causal factors present

- BN Low-risk flights ($n = 11,492$) i.e. no causal factors present

The *BEST* package (Kruschke and Meredith, 2018) in R provides a Bayesian alternative to a t -test and was used to test the difference of means of *LATGsd* between the two groups. See Kruschke (2013) for a description and examples of Bayesian estimation.

The output is shown in Figure 39 below.

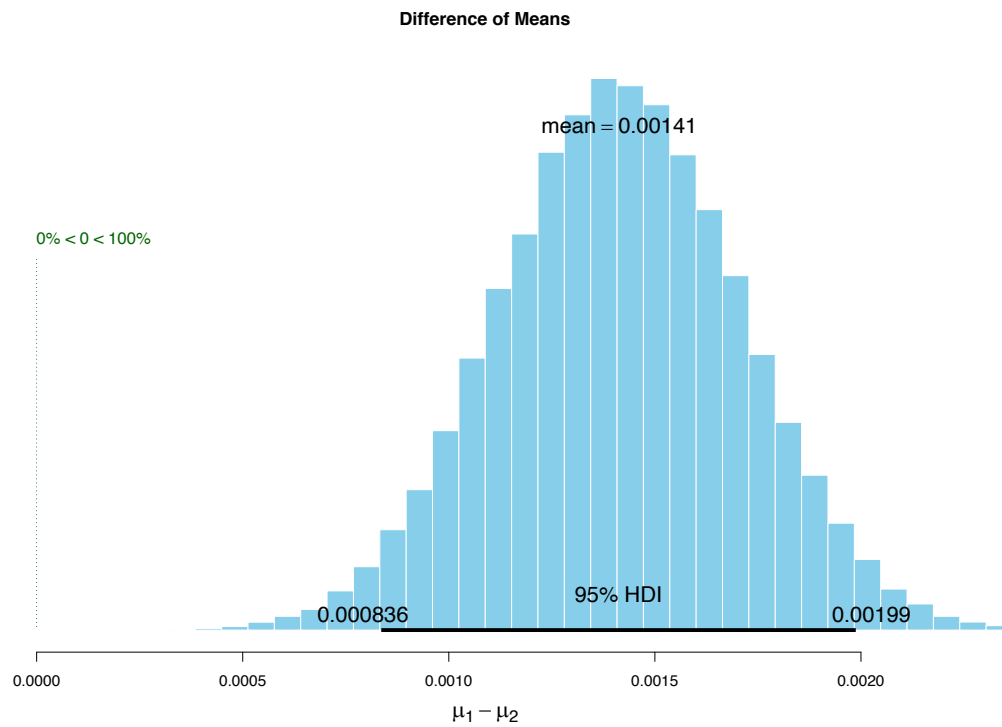


Figure 39: BEST difference of means in *LATGsd* between high-risk and normal-risk groups

The output provides more detailed information than a t test, showing the distribution of credible values for the difference between the means and the probability that the difference between the means is less than some threshold value.

The magnitude of *LATGsd* values is usually around 0.03 and Figure 39 provides a mean point estimate of 0.00141 difference between the means, which is quite small. However, it shows that the 95% highest density interval for the difference between the means of the two groups, ranges from 0.000836 to 0.00199 and the probability of the difference of the means being zero is 0%, shown here as “0% < 0 < 100%” i.e. the probability that the difference of means is more than zero, is 100%. It seems therefore, that the *high-risk* flights identified by the BN, do have larger *LATGsd* thereby confirming the expected relationship between the network calculated probabilities and the lateral deviation measure, *LATGsd*. In other words, there is evidence to support the stated hypothesis.

5.2.2 Software comparison

The same Bayesian network was also created in GeNIe (BayesFusion LLC, 2019) and the results were the same as those derived from AgenaRisk. This provides some confidence that the method is providing consistent results regardless of differences in setting up Bayesian networks in different software tools.

5.3 Further validation

The validation approach used above is encouraging, however it would be desirable to validate further through:

- Using data from other airlines from different regions around the world, particularly in areas where environmental causal factors are more prevalent. The BN would be expected to generate higher probability values of lateral deviation in these areas.
- By developing a better measure of lateral deviation. As previously mentioned, this could be achieved by using data from aircraft which record localiser deviation and/or GPS location with higher resolution.

5.4 Summary of causal network results

This section showed how flight data can be used in a causal Bayesian network to provide informative priors. Modelling the empirical distributions derived from flight data allows the probability of extreme values to be properly represented. This is necessary even for large datasets, as used here, due to the relative sparsity of extreme values compared to the main population. It has been shown that gamma, Gumbel and Johnson distributions are useful candidates when modelling flight data features due to their relatively thick tails and versatility.

The causal network was based on research of runway veer-off accidents and this led to a relatively simple network structure where most nodes were directly linked to the outcome i.e. lateral deviation.

In addition to nodes based on flight data, other nodes made use of reported weather information (e.g. runway wet, gusting winds) which had been combined with flight data observations of each landing. Other nodes were introduced to represent causal factors for which no data was available, but could in practice be made available (e.g. technical issues). The probabilities in these nodes were static and therefore had no effect on any results, however they could be used to derive deviation probabilities in different scenarios with technical failures.

The resulting network was queried by looking at different scenarios to see how evidence would affect the probability of a lateral deviation. This allowed the influence of combinations of causal factors to be quantified; something which is not possible with the current state-of-the-art tools used in flight data monitoring.

The structure of the network was retained, but the parameters of the modelled distributions altered to reflect data for different airports. Five new networks were developed for five individual airports, allowing comparison of lateral deviation probability to be made. Validating rare event probabilities is always difficult, however attempts were made to validate as far as possible by using the standard deviation of lateral acceleration. Whilst not an ideal measure of lateral deviation, in the absence of actual deviation measures in the flight data it provided a good correlation with the presence of causal factors likely to produce deviation.

It was found that there is evidence to support the hypothesis *“Flights which have an elevated BN calculated probability for veer-off, have higher LATGsd”* following a Bayesian difference of means test.

Further validation would be desirable, however the method shows that a Bayesian network using flight data derived informative priors, makes quantifying and assessing the relative probability of an operational risk possible for different combinations of causal factors and for different airports. This is currently not possible with flight data monitoring methods in use today.

5.5 Comparison of results from learning algorithms

The following sections present a number of Bayesian networks produced in GeNIe. The reader will notice several icons attached to nodes, which give a visual indication of their status. The icons are as follows:



The node has had changes made to it, but results (beliefs) have not been updated. This is the initial state of nodes.



The node has been updated and values are available.



The node has evidence entered.



The node is set as a target node e.g. for sensitivity analysis.



The node has been used to calculate the parameters for sensitivity analysis.

A fuller description can be found in the GeNIe user guide (BayesFusion LLC, 2019).

5.5.1 Bayesian search

Once the data has been discretised, it is ready for the different learning algorithms. The Bayesian search algorithm allows background knowledge to be specified prior to network structure learning. It allows the temporal relationship between variables to be specified, the result being that arcs from later to earlier variables are prohibited.

Temporal tier 1	Temporal tier 2	Temporal tier 3	Temporal tier 4	Temporal tier 5
<i>arrID</i>	<i>metXgust</i>	<i>GSlope150</i>	<i>maxG</i>	<i>LATGsd</i>
	<i>rLandXwind</i>	<i>asymTHR</i>		<i>RUDsd</i>
	<i>metVizkm</i>	<i>spdDelta50</i>		<i>HDGdev</i>
	<i>runWet</i>			

Table 34: Temporal tiers for background knowledge

Table 34 shows the assignment of variables to each temporal tier. The assignment was based on the general order of occurrence of each variable i.e. the approach to the specific airport (*arrID*) comes first, followed by the environmental conditions at that airport, followed by features of the final approach, followed by a feature of the landing, followed by features from the landing rollout.

5.5.1.1 Bayesian Search: sample size = 50

GeNIe allows for a number of algorithm parameters to be changed. Most were left at their default values, however “Iterations” was set to 100. This represents the number of restarts of the algorithm allowed and the developers of GeNIe suggest making this as large as practicable in terms of processing time. The other default parameter which was changed was “Max Time (seconds)” which is the maximum time the algorithm will run. Due to the large dataset, this limit was set to 600s as a starting point. The sample size, which “...represents the inertia of the current parameters when introducing new data” (BayesFusion LLC, 2019), was left at the default value of 50.

The learning algorithm was executed and the resulting network is shown in Figure 40.

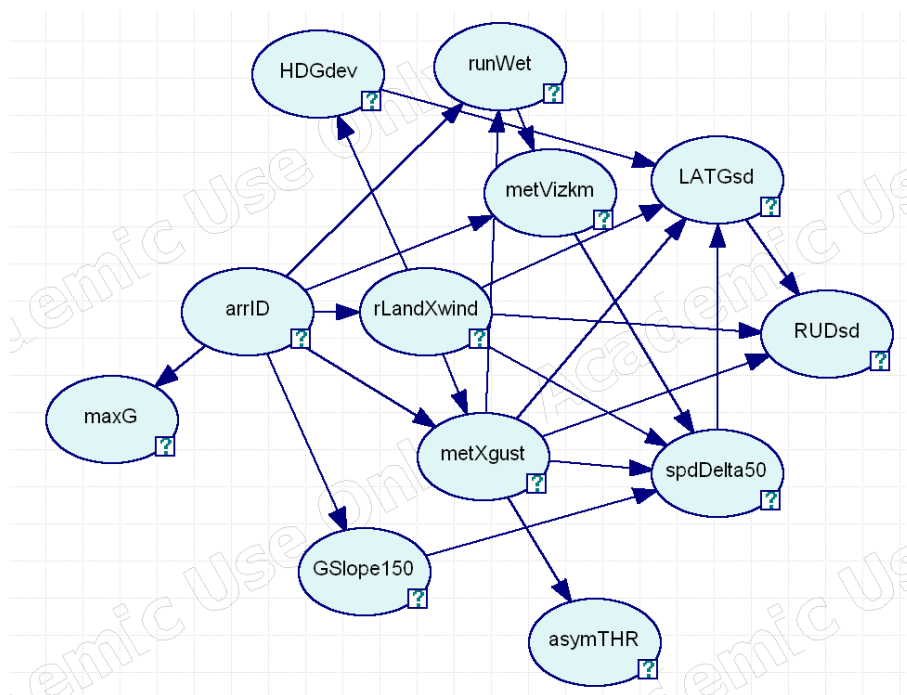


Figure 40: Learnt network using Bayesian Search algorithm

The accompanying metrics reported by GeNIe were:

Best score in iteration 52: -3.50026e+06

EM Log Likelihood⁶: -3.49206e+06

These metrics can be used to compare the performance of the network in representing the data i.e. the learning parameters can be altered and the resulting networks compared. It is interesting to note that neither *maxG* nor *asymTHR* have any effect on any other parameters (i.e.

⁶ EM Log Likelihood is a metric reported by GeNIe based on the expectation-maximisation algorithm. It can be used to compare networks learnt with different parameters. Higher values indicate that the network is more consistent with data it was learnt from.

they have no children nodes) according to the Bayesian Search algorithm. It appears that *maxG* is influenced by the arrival airport (*arrID*), which is reasonable, however *asymTHR* has *metXgust* (gusting crosswind present) as a parent which does not make much sense in causal terms given that asymmetric thrust should not be a technique used on this aircraft type in crosswind conditions.

Another feature of this network is that it introduces an arc between *metVizkm* and *spdDelta50*. The effect of visibility on speed might not be immediately apparent, but it can be seen by introducing evidence at the visibility node and seeing how the speed probabilities change. Setting evidence at the *metVizkm* node to the lowest visibility, the probability of *spdDelta50* being high reduces. This would be expected, because approaches in low visibility are more often flown with the autopilot and autothrust engaged, usually resulting in less speed deviation from the target. Also, in the author's experience, pilots are less likely to fly unstable approaches in poor visibility.

There is an arc to *metVizkm* from *runWet*, which in turn has an arc from *metXgust*, which could indicate that poor weather, such as gusty and wet conditions are influencing the *spdDelta50* node. Approaches are flown faster in gusty conditions and more variability around the target speed would be expected.

Elsewhere, the learnt network follows a mostly logical and causal flow:

- Weather related nodes are dependent on the arrival airport – some airports are likely to experience more inclement weather than others.
- Approach speed has gusts and crosswinds as parents as expected, and also glideslope deviation. Approaches which are high are quite often fast too.
- Heading deviation has an arc to lateral acceleration standard deviation, which in turn has an arc to rudder deflection standard deviation.

Table 35 shows the strength of influence between the nodes, sorted by the average distance “...between the various conditional probability distributions over the child node conditional on the states of the parent node” (BayesFusion LLC, 2019):

Parent	Child	Average	Maximum
<i>runWet</i>	<i>metVizkm</i>	0.812903	0.967996
<i>LATGsd</i>	<i>RUDsd</i>	0.80508	0.96409
<i>arrID</i>	<i>GSlope150</i>	0.744525	0.976706
<i>arrID</i>	<i>rLandXwind</i>	0.723703	0.975084
<i>metXgust</i>	<i>runWet</i>	0.711269	0.964884
<i>arrID</i>	<i>maxG</i>	0.700233	0.972003
<i>metXgust</i>	<i>spdDelta50</i>	0.698689	0.948308
<i>rLandXwind</i>	<i>RUDsd</i>	0.691147	0.932093
<i>rLandXwind</i>	<i>LATGsd</i>	0.657746	0.970307
<i>rLandXwind</i>	<i>HDGdev</i>	0.64943	0.882569
<i>metXgust</i>	<i>LATGsd</i>	0.638561	0.945366
<i>arrID</i>	<i>metVizkm</i>	0.615769	0.988769
<i>HDGdev</i>	<i>LATGsd</i>	0.60204	0.958112
<i>arrID</i>	<i>metXgust</i>	0.543608	0.98166
<i>rLandXwind</i>	<i>spdDelta50</i>	0.508641	0.964764
<i>metVizkm</i>	<i>spdDelta50</i>	0.505338	0.944718
<i>metXgust</i>	<i>RUDsd</i>	0.436036	0.731308
<i>GSlope150</i>	<i>spdDelta50</i>	0.408371	0.948664
<i>arrID</i>	<i>runWet</i>	0.375158	0.981662
<i>spdDelta50</i>	<i>LATGsd</i>	0.369542	0.914436
<i>rLandXwind</i>	<i>metXgust</i>	0.274417	0.94873
<i>metXgust</i>	<i>asymTHR</i>	0.0442262	0.0442262

Table 35: Strength of influence between nodes, , sample size = 50

Note: The table is presented here to show the relative strengths of the arcs based on the J-Divergence distance as advocated by Koiter (2006); for an explanation of the meaning of the actual values, the reader is directed to Koiter (2006).

The arc between *metXgust* and *asymTHR* is weak, as expected. Also as expected, *runWet* to *metVizkm* and *LATGsd* to *RUDsd* are strong influences. Fairly strong influences are also shown between *rLandXwind* and *RUDsd*, *LATGsd* and *HDGdev* which is intuitive.

5.5.1.2 Bayesian Search: sample size = 500

To investigate the effect of sample size, the network learning was carried out again on the same dataset, with the same GeNIe parameters except that the sample size was set to 500. The resulting network is shown in Figure 41.

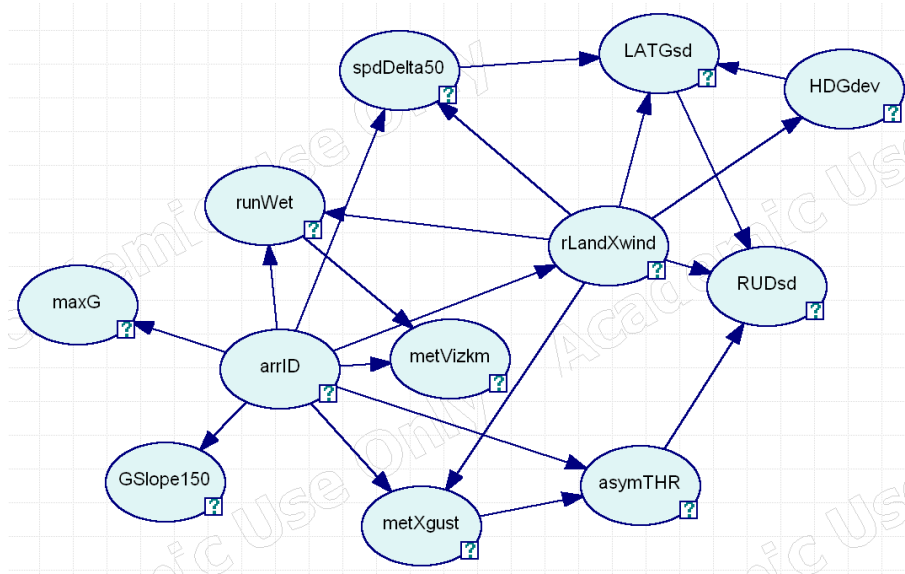


Figure 41: Bayesian Search learnt network, sample size = 500

The accompanying metrics reported by GeNIe were:

Best score in iteration 92: -3.49836e+06

EM Log Likelihood: -3.48774e+06

The resulting network is slightly simpler than the previous example, with 20 arcs compared to 22. The EM Log Likelihood score is also marginally better (closer to zero is an improvement). The arc between *Gslope50* and *spdDelta50* has now gone, but a new arc between *asymTHR* and *RUDsd* has been introduced. The arc between *metVizkm* and *spdDelta50* has also gone. Overall, this network seems to be a slight improvement over the previous example.

Calculating the strengths of the arcs gives:

Parent	Child	Average	Maximum
<i>metXgust</i>	<i>asymTHR</i>	0.873009	0.985523
<i>runWet</i>	<i>metVizkm</i>	0.812903	0.967996
<i>arrID</i>	<i>GSlope150</i>	0.744525	0.976706
<i>arrID</i>	<i>rLandXwind</i>	0.723703	0.975084
<i>arrID</i>	<i>maxG</i>	0.700233	0.972003
<i>arrID</i>	<i>spdDelta50</i>	0.674521	0.976247
<i>rLandXwind</i>	<i>HDGdev</i>	0.64943	0.882569
<i>arrID</i>	<i>metVizkm</i>	0.615769	0.988769
<i>asymTHR</i>	<i>RUDsd</i>	0.590711	0.950648
<i>rLandXwind</i>	<i>spdDelta50</i>	0.582155	0.973223
<i>rLandXwind</i>	<i>LATGsd</i>	0.578374	0.91625
<i>arrID</i>	<i>metXgust</i>	0.543608	0.98166
<i>HDGdev</i>	<i>LATGsd</i>	0.536829	0.905571

<i>arrID</i>	<i>asymTHR</i>	0.52684	0.984946
<i>LATGsd</i>	<i>RUDsd</i>	0.504357	0.974388
<i>rLandXwind</i>	<i>RUDsd</i>	0.482741	0.955017
<i>rLandXwind</i>	<i>runWet</i>	0.298224	0.943395
<i>rLandXwind</i>	<i>metXgust</i>	0.274417	0.94873
<i>spdDelta50</i>	<i>LATGsd</i>	0.171547	0.541262
<i>arrID</i>	<i>runWet</i>	nan	0.977907

Table 36: Strength of influence between nodes, sample size = 500

It is not apparent from GeNIe why the average strength for *arrID* to *runWet* has been calculated as “nan” (not a number), however its positioning towards the bottom of the table would be expected, similar to its position in the previous network, and other distance measures in GeNIe (e.g. Euclidean) also suggest a weak influence. The *metXgust* and *asymTHR* pairing at the top of the list does not make much sense for the reasons described earlier.

Overall the network is slightly simpler with a marginally better log likelihood, however the changes to the strength of influence of the arcs leads to a conclusion that it is not a significant improvement.

5.5.1.3 Bayesian Search: sample size = 5000

Further increasing the sample size to 5000 yields the following network.

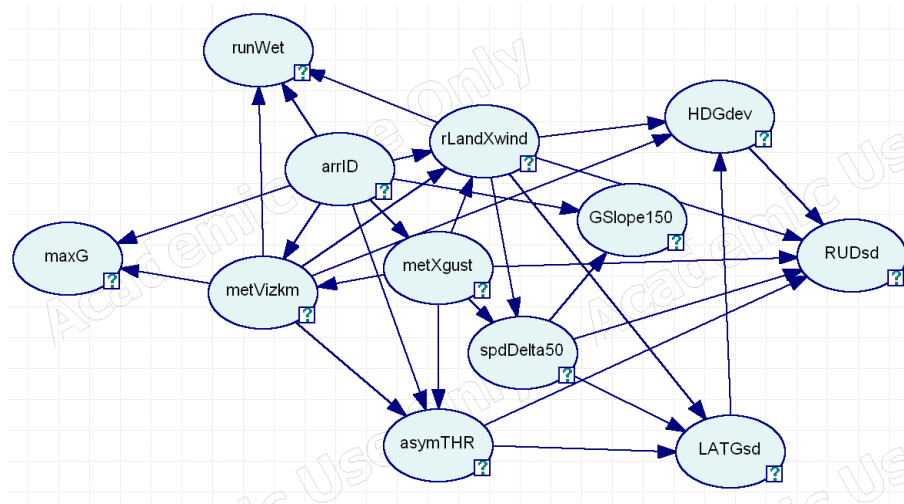


Figure 42: Bayesian Search learnt network, sample size = 5000

The network has increased in complexity, with 30 arcs and the log likelihood has reduced to -3.51105e+06, so increasing the sample size has not improved the network. It appears that increasing the sample size leads to overfitting of the network.

5.5.1.4 Summary of effect of sample size

Sample sizes of 250, 375, 750 and 1000 were also tried and the results are summarised in Table 37.

SAMPLE SIZE	ARCS	EM LOG LIKELIHOOD
50	22	-3.49206e+06
250	21	-3.48748e+06
375	20	-3.48794e+06
500	20	-3.48774e+06
750	21	-3.48729e+06
1000	22	-3.48708e+06
5000	30	-3.51105e+06

Table 37: Summary of effect of Bayesian Search sample size on learnt network

Mid-table there is little variation and the sample size changes and the number of arcs is at a minimum around 375 to 500, so a sample size of 500 seems like a reasonable compromise.

5.5.1.5 Effect of discretisation

5.5.1.5.1 Number of intervals

The networks above were all based on data discretised into 4 intervals. Increasing the number of intervals might improve the accuracy of the network. To investigate this, the data was discretised following the schema described in 4.8.2, however 8 intervals were used instead of 4. The Bayesian Search parameters remained the same and a sample of size of 500 used. The resulting network is shown in Figure 43.

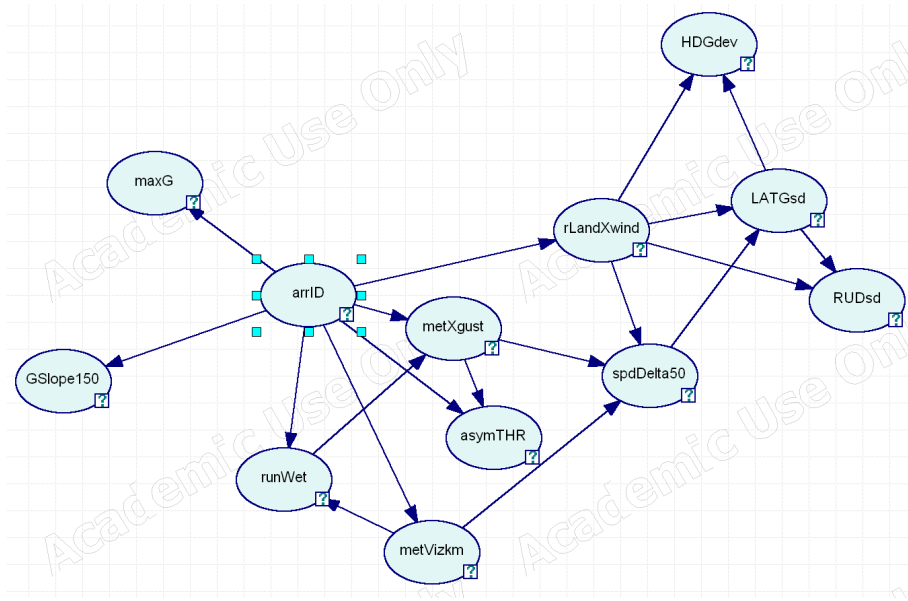


Figure 43: Bayesian Search network learnt from data discretised into 8 intervals

This network has an EM Log Likelihood of -4.9046e+06, however it cannot be compared to the score from previous networks because it is based on different data, due to the increase in discretisation intervals. It has 19 arcs and the strength of influences is shown in Table 38.

Parent	Child	Average	Maximum
<i>metXgust</i>	<i>asymTHR</i>	0.873009	0.985523
<i>arrID</i>	<i>GSlope150</i>	0.851714	0.982471

<i>metXgust</i>	<i>spdDelta50</i>	0.829722	0.94543
<i>arrID</i>	<i>rLandXwind</i>	0.827902	0.979706
<i>arrID</i>	<i>maxG</i>	0.817899	0.978445
<i>LATGsd</i>	<i>RUDsd</i>	0.737006	0.972833
<i>arrID</i>	<i>metVizkm</i>	0.715941	0.991118
<i>rLandXwind</i>	<i>RUDsd</i>	0.643741	0.955427
<i>rLandXwind</i>	<i>LATGsd</i>	0.637396	0.94787
<i>rLandXwind</i>	<i>spdDelta50</i>	0.600107	0.973422
<i>metVizkm</i>	<i>spdDelta50</i>	0.586778	0.971511
<i>LATGsd</i>	<i>HDGdev</i>	0.574023	0.93387
<i>runWet</i>	<i>metXgust</i>	0.571688	0.959138
<i>arrID</i>	<i>asymTHR</i>	0.52684	0.984946
<i>rLandXwind</i>	<i>HDGdev</i>	0.481459	0.881846
<i>arrID</i>	<i>metXgust</i>	0.481021	0.978483
<i>metVizkm</i>	<i>runWet</i>	0.380193	0.982284
<i>spdDelta50</i>	<i>LATGsd</i>	0.155812	0.543446
<i>arrID</i>	<i>runWet</i>	-nan(ind)	0.978178

Table 38: Strength of influence for 8 interval discretised network

The network retains many of the features of the previous networks, with *maxG* and *GSlope150* being redundant in influencing the lateral deviation nodes (*HDGdev*, *LATGsd*, *RUDsd*). The highest strength of influence is again the *metXgust* to *asymTHR* arc. The persistence of this arc at the top of the table deserves further exploration, so the *asymTHR* observations were split into two groups; *metXgust* present and *metXgust* not present. Bayesian estimation (Kruschke, 2013) was then used to estimate the difference in means between the two groups, as shown in Figure 44.

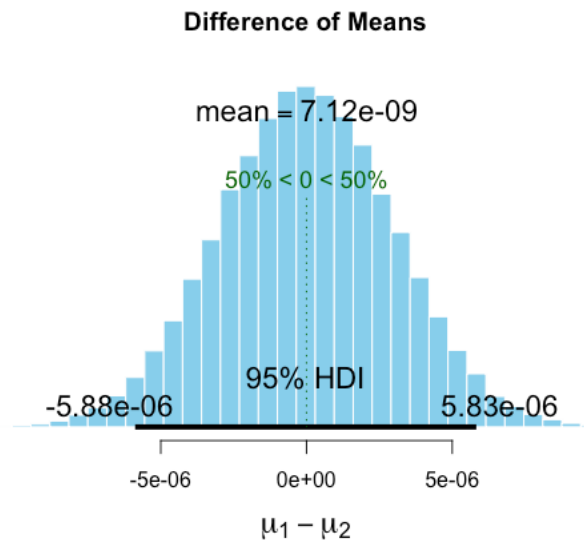


Figure 44: Bayesian estimation of difference of means for asymmetric thrust with and without gusting winds

There is no significance difference between the means and there is no logical causal link, therefore it would make sense to remove the arc.

Overall, it appears that increasing the number of intervals during discretisation has resulted in a slightly more refined model. To explore this further, the data was discretised into 16 intervals, yielding the following network (same Bayesian search parameters as before):

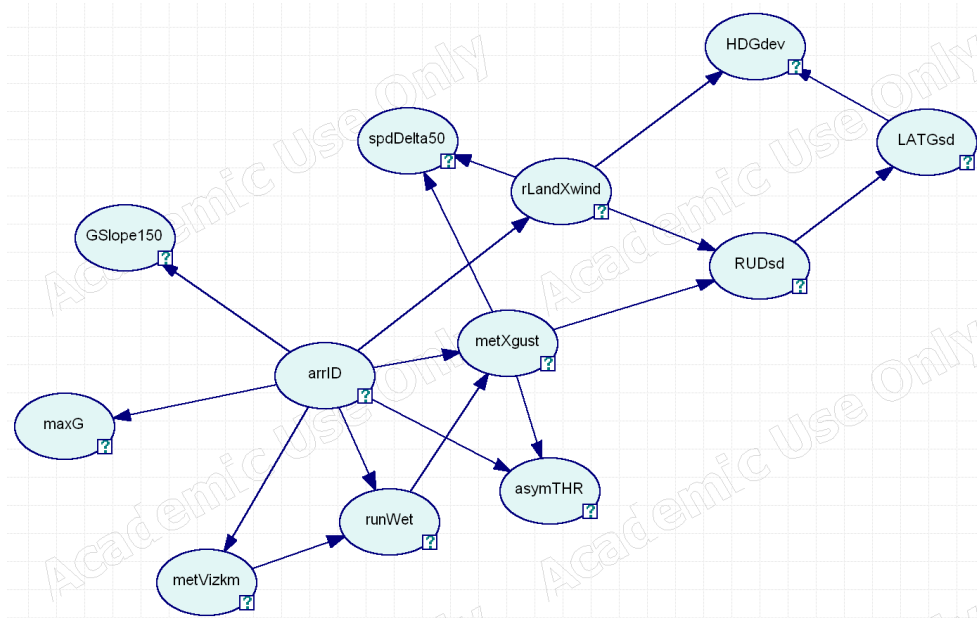


Figure 45: Bayesian Search network using 16 interval discretisation

Note that *metVizkm* was discretised into 8 equal width intervals, as before, because 16 intervals results in unrealistic counts due to the way the visibility is reported.

The new network in Figure 45 has just 17 arcs and lacks the connections between visibility and speed and crosswinds and lateral acceleration, which seems to be a retrograde step, despite the reduced complexity.

5.5.1.5.2 Discretisation method

The networks in the previous section used data discretised into equal count intervals (apart from *metVizkm* which used equal widths). GeNIe also allows hierarchical discretisation using the following algorithm:

- Input: N =# of records, K =# of desired bins
1. Let k denote the running number of bins, initialized to $k=N$ (each record starts in its own cluster)
 2. If $k=K$ quit, else set $k=k-1$ by combining the two bins whose mean value has the smallest separation
 3. Repeat 2

(BayesFusion LLC, 2019)

Hierarchical discretisation results in the more extreme values having dedicated intervals, rather than being lumped together (and therefore possibly lost), with unextreme observations.

However, it also results in some intervals having very few observations, even in a large dataset.

The data was discretised into 8 intervals using hierarchical discretisation and the resulting network is shown in Figure 46.

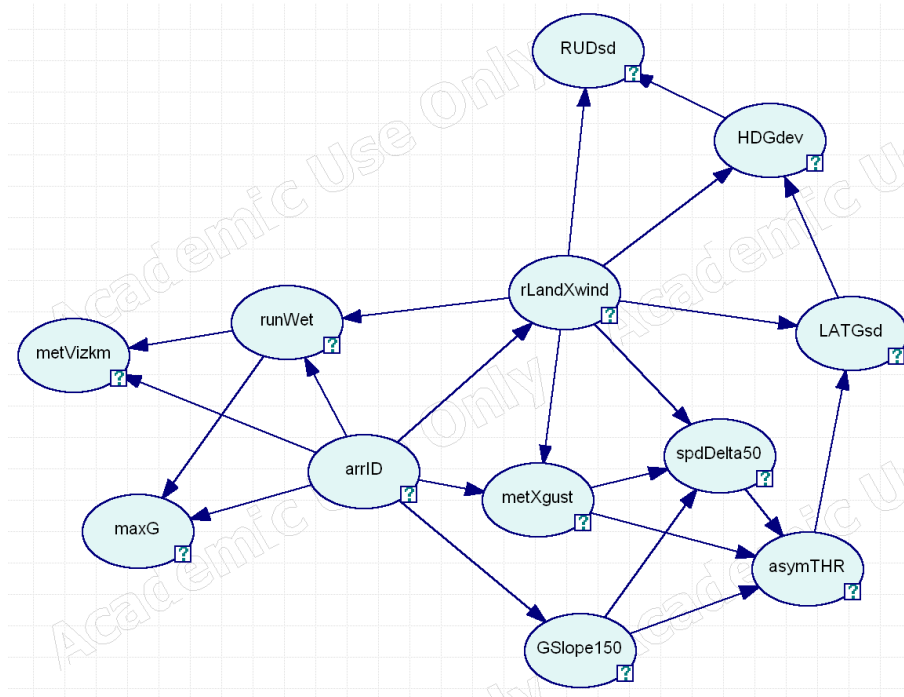


Figure 46: Bayesian search learnt network using 8 interval hierarchical discretisation

This version of the network has 22 arcs and it appears that there may be an element of overfitting e.g. *GSlope150* to *asymTHR*. Also, the lack of an arc between *metXgust* and any of the lateral deviation related nodes, is questionable.

This was repeated using 4 interval hierarchical discretisation and the overfitting is more obvious with 30 arcs in the network (Figure 47).

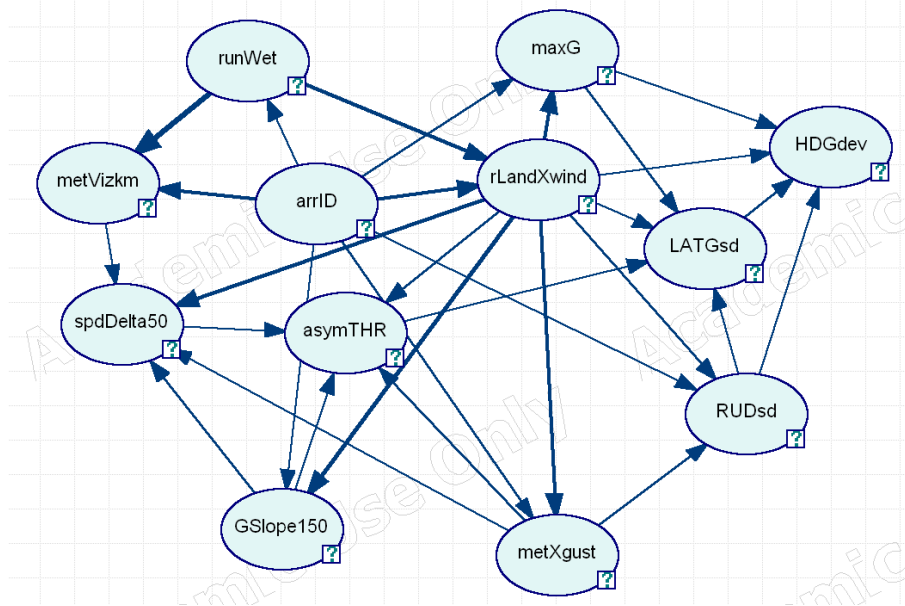


Figure 47: Bayesian search network from 4 interval hierarchical discretisation

5.5.1.6 Bayesian Search summary

Bayesian Search is a popular algorithm for learning Bayesian networks and it has produced several candidate networks from the observed flight data. The resulting number of arcs, and thus the overall complexity of the model, is dependent on the number of samples used.

Likewise, the EM Log Likelihood score is also influenced by the number of samples and it seems that, in this case, somewhere between 300 and 500 samples would be a good compromise, generating less complex networks, but with good likelihood scores when compared to very low and very high sample sizes.

Increasing the intervals of discretisation oversimplified the model, resulting in the loss of important arcs, and hierarchical discretisation appears to result in some overfitting.

The Bayesian search using 4 interval equal count discretisation and 500 samples, shown in Figure 41, appears to be the best network generated by this algorithm and it could be further enhanced with expert input, by removing questionable arcs and possibly introducing one or two others.

5.5.2 PC learning

The PC learning algorithm is compatible with continuous variables, however they should be reasonably aligned with a Normal distribution. As this is not the case for the observed flight data, and due to there being a mixture of continuous and discrete variables, the continuous variables were discretised again using 4 intervals of equal counts. The same background

knowledge was used and a significance threshold of 0.01 set. The resulting network is shown in Figure 48.

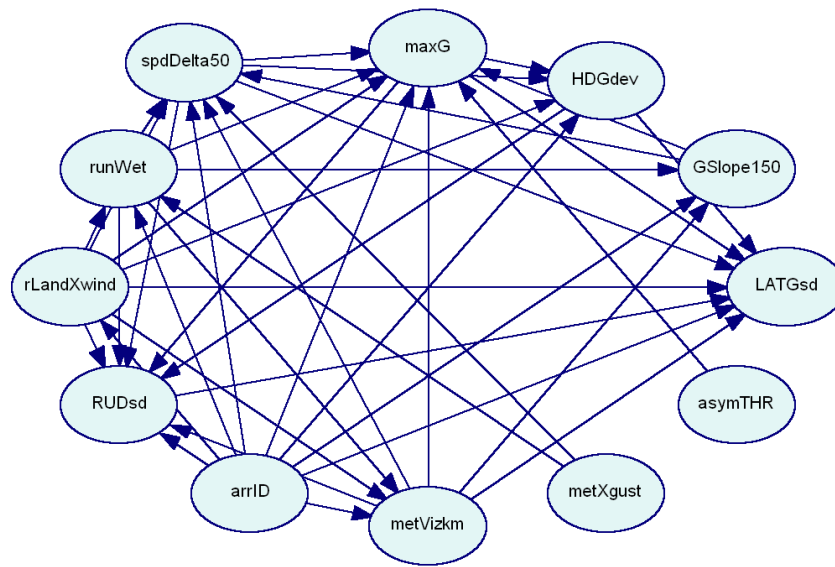


Figure 48: Network learnt using PC algorithm

The network is very complex with many nodes having multiple parents. Compared to the Bayesian Search algorithm, which produced one or two parents for *maxG*, the PC learning algorithm results in it having 7 parents. *LATGsd* and *RUDsd* also have 7 parents. Due to the apparent overfitting and complexity of the network, further investigation of the algorithm was not pursued.

5.5.3 Greedy Thick Thinning

Greedy Thick Thinning is closely related to the Bayesian Search algorithm. Using the same background knowledge as before, the network learnt by the algorithm was as follows.

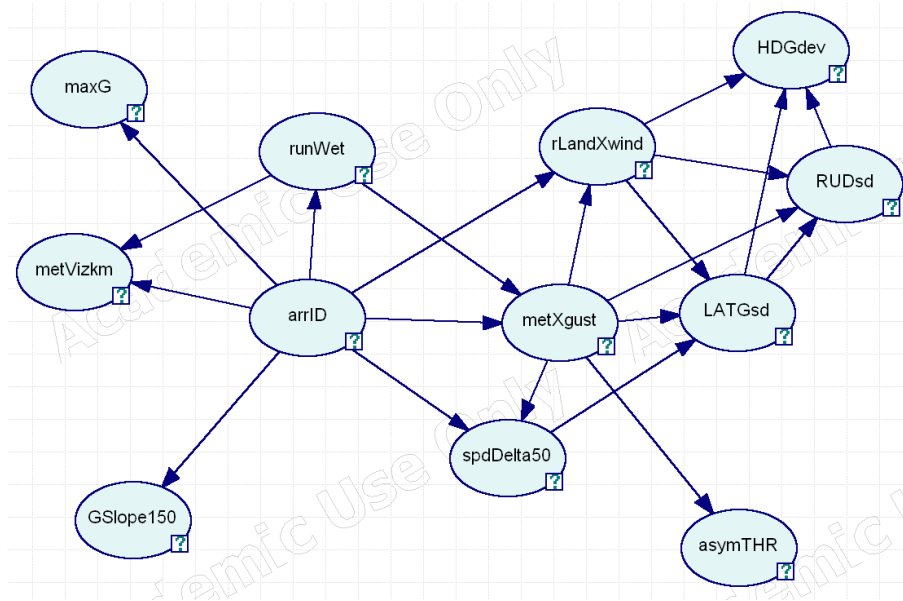


Figure 49: Network learnt using Greedy Thick Thinning algorithm

Figure 49 shows a fairly simple network, compared with that from PC learning, and as expected is similar to the networks learnt by Bayesian Search. In total there are 21 arcs and there are no real surprises in the connections, which mostly follow an intuitive causal logic. Table 39 shows the strength of influence of the arcs.

Parent	Child	Average	Maximum
runWet	metVizkm	0.812903	0.967996
LATGsd	RUDsd	0.80508	0.96409
arrID	GSlope150	0.744525	0.976706
metXgust	rLandXwind	0.740155	0.95184
metXgust	spdDelta50	0.739413	0.959102
rLandXwind	LATGsd	0.710408	0.938986
arrID	maxG	0.700233	0.972003
rLandXwind	RUDsd	0.691147	0.932093
arrID	metVizkm	0.615769	0.988769
arrID	rLandXwind	0.586804	0.976488
LATGsd	HDGdev	0.580175	0.915271
arrID	spdDelta50	0.578471	0.975654
metXgust	LATGsd	0.575704	0.757023
runWet	metXgust	0.571688	0.959138
arrID	metXgust	0.481021	0.978483
rLandXwind	HDGdev	0.465797	0.866012
metXgust	RUDsd	0.436036	0.731308
arrID	runWet	0.397705	0.981662
spdDelta50	LATGsd	0.286439	0.740489

<i>RUDsd</i>	<i>HDGdev</i>	0.129406	0.686489
<i>metXgust</i>	<i>asymTHR</i>	0.0442262	0.0442262

Table 39: Strength of influence of arcs from Greedy Thick Thinning algorithm

This network also has an arc between *metXgust* and *asymTHR*, however it has a very weak influence compared to the other arcs.

The Greedy Thick Thinning algorithm in GeNIe has only two parameters which can be changed; maximum number of parents and background knowledge. The maximum number of parents was set to 8, the default which was set for all the other networks learnt, and the background knowledge used was the same as for the other networks. To see how the algorithm is affected by different discretisation, 8 interval data was used which resulted in the following network.

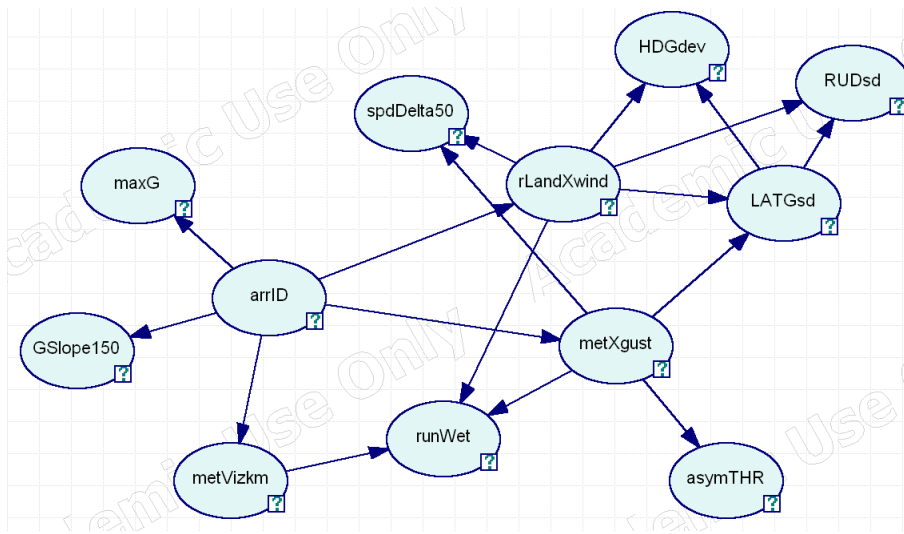


Figure 50: Network learnt using Greedy Thick Thinning and 8 interval discretised data

The network becomes slightly more simplified, reducing to just 17 arcs, however the arc between *metXgust* and *rLandXwind* is lost and *spdDelta50* no longer influences any of the deviation related parameters (i.e. *LATGsd*, *RUDsd* and *HDGdev*). It seems therefore that it is not an improvement over the previous network which was learnt from 4 interval discretised data. Hierarchical 4 interval discretisation was also tried and results were similar to those found with Bayesian Search using the same data. Some overfitting was evident and the network was not an overall improvement on equal count discretisation.

Overall, Greedy Thick Thinning appears to work well with this dataset, discretised into 4 equal count intervals. The resulting network has a mostly logical causal flow and is not overly complex, however it does suffer from several nodes which are children only e.g. *maxG* and *GSlope150*. This means they do not influence the lateral deviation nodes.

5.5.4 Tree Augmented Naïve Bayes

The remaining learning algorithms available in GeNIe are Tree Augmented Naïve Bayes, Augmented Naïve Bayes and Naïve Bayes. All of these algorithms require a Class Variable to be declared prior to learning and this variable will be a parent of all other variables. As such, it is probably not the best choice for causal networks, however the Tree Augmented algorithm was tried, with *LATGsd* as the class variable as this is the main variable of interest. The resulting network is shown below.

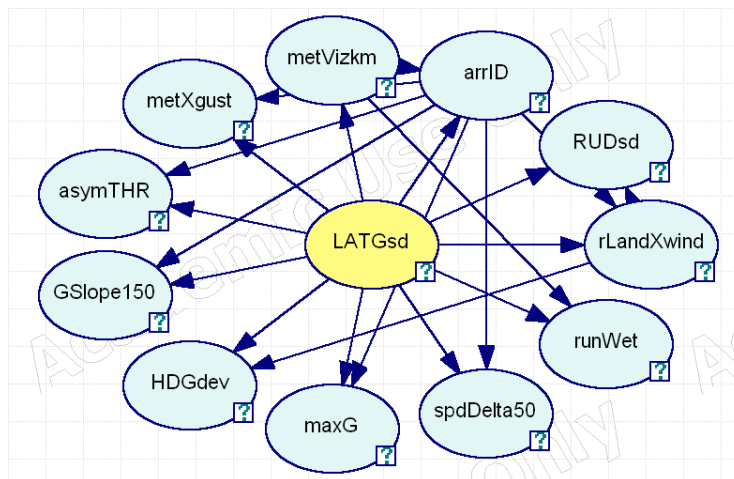


Figure 51: Network learnt using Tree Augmented Naive Bayes algorithm

The expected arcs are in the network, however there are other arcs which do not follow a causal logic. It could be used as a starting point and the network modified using expert knowledge, however it would be very difficult to achieve a properly causal flow without creating cycles in the network.

5.6 Comparison of network performance

For any given problem, and any given set of data, in most cases there is unlikely to be a single network which is “best”; more likely is that there will be several candidate networks which will model the problem sufficiently. In the previous sections, several networks have been produced through different methods, but using an identical dataset. This section will examine the performance of the networks and compare how they would rank the five airports used earlier, in relation to likelihood of lateral deviation.

The networks which will be compared are:

1. Bayesian Search, 500 samples
2. PC learning

3. Greedy Thick Thinning (GTT)
4. Tree Augmented Naïve Bayes (TAN)

All the networks will use the dataset discretised by equal counts into four intervals, and where applicable use the same background knowledge.

The specific airport is introduced as evidence at the *arrID* node and the probability of the most severe cases of *LATGsd*, *RUDsd* and *HDGdev* are calculated. The following tables show the results.

Airport - BHD	<i>LATGsd</i>	<i>RUDsd</i>	<i>HDGdev</i>
Bayesian Search	0.251	0.253	0.271
PC learning	0.286	0.293	0.235
GTT	0.254	0.256	0.272
TAN	0.290	0.298	0.235

Airport - DLM	<i>LATGsd</i>	<i>RUDsd</i>	<i>HDGdev</i>
Bayesian Search	0.143	0.122	0.171
PC learning	0.149	0.115	0.137
GTT	0.142	0.121	0.165
TAN	0.103	0.078	0.131

Airport - DME	<i>LATGsd</i>	<i>RUDsd</i>	<i>HDGdev</i>
Bayesian Search	0.219	0.213	0.247
PC learning	0.236	0.195	0.214
GTT	0.220	0.215	0.246
TAN	0.232	0.181	0.213

Airport - EDI	<i>LATGsd</i>	<i>RUDsd</i>	<i>HDGdev</i>
Bayesian Search	0.184	0.163	0.200
PC learning	0.174	0.130	0.165
GTT	0.186	0.166	0.201
TAN	0.169	0.122	0.165

Airport - LHR	<i>LATGsd</i>	<i>RUDsd</i>	<i>HDGdev</i>
Bayesian Search	0.273	0.277	0.290
PC learning	0.293	0.295	0.302
GTT	0.270	0.275	0.290
TAN	0.292	0.294	0.302

Table 40: Tables showing probabilities of high *LATGsd*, *RUDsd* and *HDGdev*, for different airports, derived by different learning algorithms

The Bayesian Search and Greedy Thick Thinning learnt networks produce very similar probabilities for the lateral deviation nodes, unsurprisingly given how closely related the algorithms are. Table 41 shows how each network would rank the airports used in the earlier causal network.

	Causal network	Bayesian Search	PC Learning	GTT	TAN
1	DME	LHR	LHR	LHR	LHR
2	BHD	BHD	BHD	BHD	BHD
3	EDI	DME	DME	DME	DME
4	LHR	EDI	EDI	EDI	EDI
5	DLM	DLM	DLM	DLM	DLM

Table 41: Ranking of specific airport probabilities of lateral deviation calculated from different networks

All of the learnt networks rank the airports in the same order and there is some commonality with ranking produced by the earlier causal network (see 5.1.2). DLM is ranked lowest for lateral deviation probability across all networks, and likewise BHD is ranked 2nd. The major difference in rankings is how the learnt networks ranked LHR compared to the causal network.

5.7 Sensitivity analysis

Sensitivity analysis can be used to tune parameters of a Bayesian network to provide a desired output, so for example in this network, sensitivity analysis could be used to determine which parameter needs to be changed, and by how much, to result in an X% drop in veer-off probability (Darwiche, 2009). It is also a method often used to evaluate the output of Bayesian networks, especially when there might be some uncertainty over the model, particularly the structure, or when it has been necessary to elicit expert opinion to help build the network.

Sensitivity analysis shows which nodes have the most influence on any particular target node, and if it is shown that a node has unexpectedly high influence there may be a problem with the definition of that node, or perhaps the structure of the network. That is not to say that all nodes should have equal influence; in the case of a causal Bayesian network it would be expected that the most prevalent causes would have the most influence.

There are several methods for sensitivity analysis (Castillo, Gutierrez and Hadi, 1997; Jensen, Aldenryd and Jensen, 1995), some of them extremely complex, but for this analysis the built-in function in GeNIe will be used, which follows the work of Kjærulff and van der Gaag (2000). The method highlights nodes for which a small change in a parameter results in a large change in the posterior probability of a target node.

5.7.1 Sensitivity analysis of Bayesian Search network

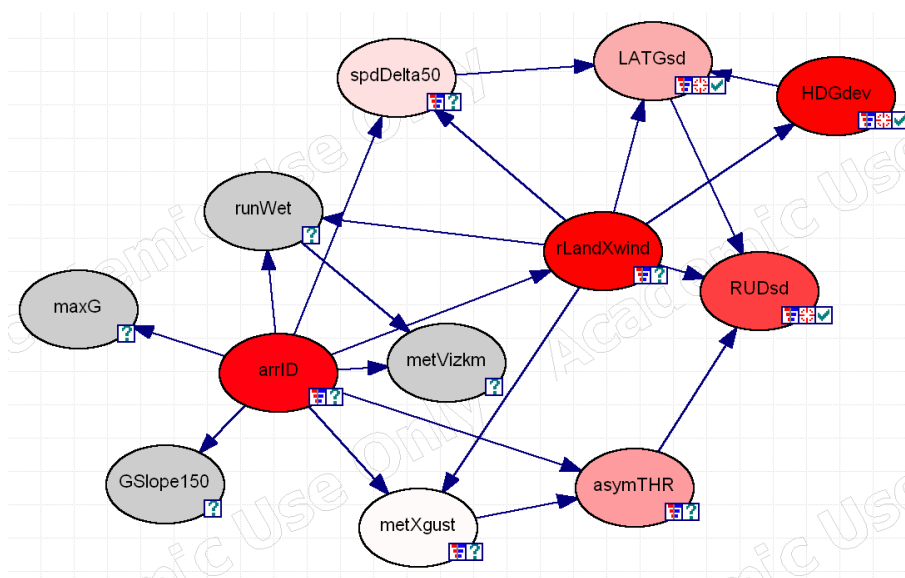


Figure 52: Sensitivity analysis of Bayesian Search learnt network

The shading of the nodes in Figure 52 depicts how sensitive the target nodes are to changes in parameters in the shaded variable node. Red nodes are the most important for the calculation of posterior probabilities in the target nodes, pink nodes less important and grey nodes have no effect.

The sensitivity analysis of the Bayesian Search learnt network confirms that there are several nodes which are not used in the calculation of the posterior probabilities of the lateral deviation target nodes (i.e. *LATGsd*, *RUDsd* and *HDGdev*). This is apparent from the structure of the network as those nodes have no route to the target nodes. The target nodes are most sensitive to *arrID* and *rLandXwind*. A concern in this network is that *metXgust* has little influence over the lateral deviation nodes.

5.7.2 Sensitivity analysis of PC learning network

It was not possible to run sensitivity analysis on the PC learning network because it resulted in an error in GeNIe. This could be due to the network having several nodes with a large number of parents, which results in large node probability tables, and the overall complexity of the table.

5.7.3 Sensitivity analysis of Greedy Thick Thinning network

The sensitivity analysis of the Greedy Thick Thinning network shows similar results to the Bayesian Search sensitivity analysis, however this time the deviation nodes are also sensitive to changes in *metXgust*.

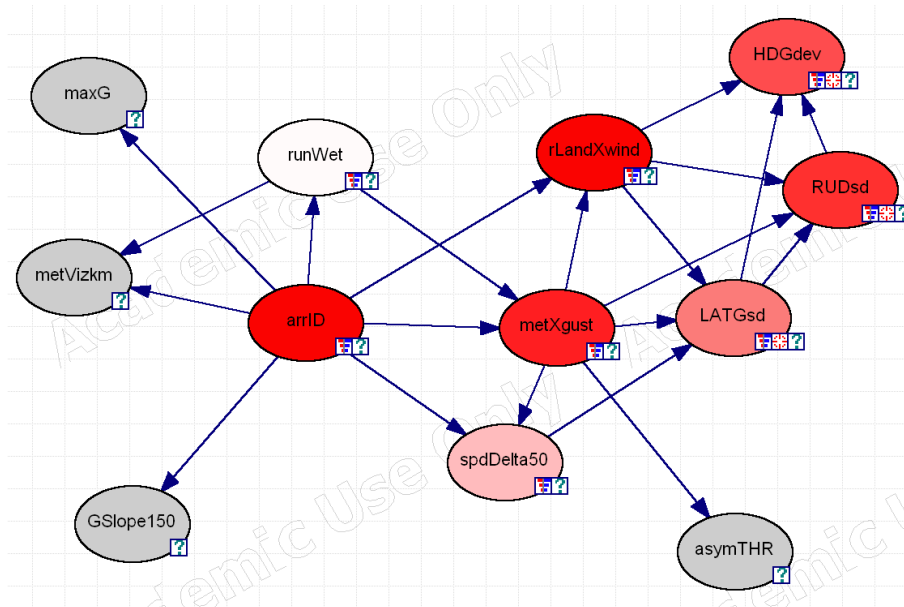


Figure 53: Sensitivity analysis of Greedy Thick Thinning network

There are no real surprises in the results of the analysis, with crosswinds being the leading factor in changes to the lateral deviation probabilities.

5.7.4 Sensitivity analysis of Tree Augmented Naïve Bayes network

LATGsd is a parent for all the other nodes in the TAN network and therefore it influences these other nodes, rather than being influenced by them. Setting it as the target node for sensitivity analysis will not yield any useful information.

5.7.5 Sensitivity analysis summary

Sensitivity analysis is usually used to fine tune parameters of a Bayesian network, however here it has been used to identify peculiarities of learnt networks. It is a useful tool for highlighting relationships between variables which should exist, but are absent, or should not exist and are present. It highlights the relative strength of influence each variable has on a target variable, which can be coupled with background knowledge (e.g. from a SME) to help assess the validity of the network.

5.8 Learning parameters for causal network

In Chapter 4.6 a Bayesian network was defined based on causal factors determined from accident investigation reports. The structure was determined simply by the causal relationships of the variables and the outcome. The parameters for the network nodes were derived manually from empirical data through a mixture of modelling continuous variables and calculating probabilities of discrete variables. So far in this chapter, both the structure and parameters of the network have been learnt from the observed data using different, popular algorithms.

Another approach is to impose the network structure and learn the parameters for the nodes from data. In other words, the network is defined manually using the causal relationship between the variables, and the probability tables for each node are learnt from data.

One of the advantages of the learnt networks over the manually defined network, was the ease in which comparisons could be made between different airports. The manually defined network required the empirical distributions for each variable to be modelled for each airport, whereas the airport was included as a variable in the learnt networks and therefore running comparative scenarios was easy. The network shown in Figure 54 seeks to address this problem by including the airport (*arrID*) as a parent for each of the causal variables.

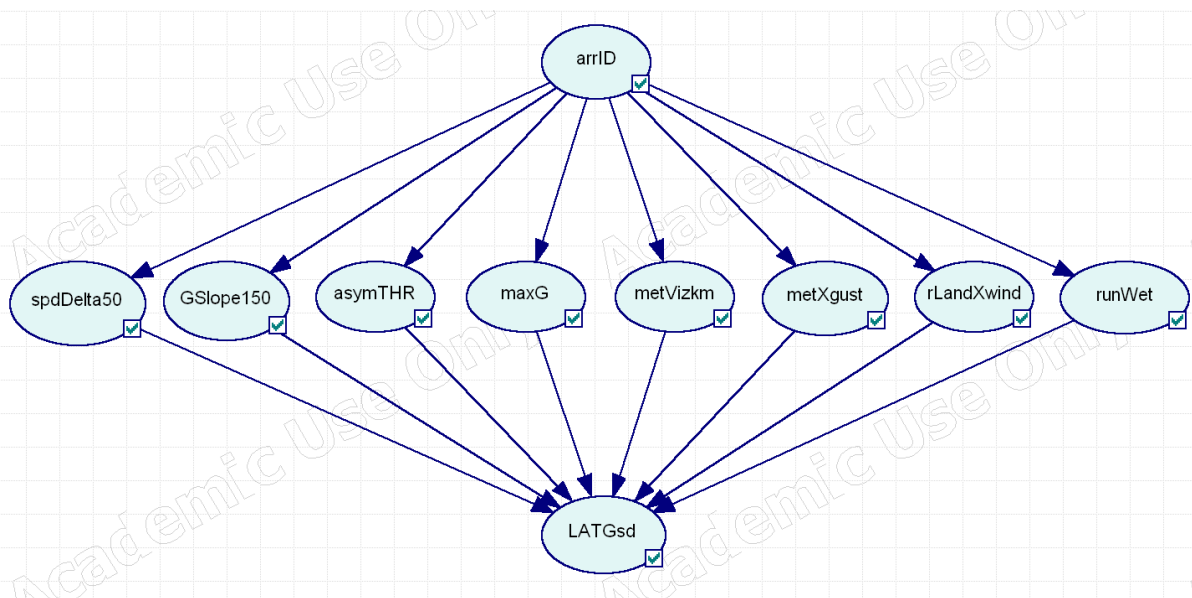


Figure 54: Structure for causal network with learnt parameters

The network differs from the manual causal network derived in chapter 4.6 because only the causal factors which can be directly derived from the observed data are included. This is necessary for the parameter learning to work. It also differs from the previously learnt networks by having a single indicator of lateral deviation (*LATGsd*); *RUDsd* and *HDGdev* have been omitted to reduce the complexity of the network. Had those variables been included, there

would have been two additional nodes, each having 8 parent nodes, and the resulting network would be slow to use.

A subset of the entire dataset relating to the five airports (BHD, DLM, DME, EDI and LHR) was extracted for the parameter learning and discretised according to the schema described in 4.8.2. GeNIe allows node states and data variables to be matched and to make this easier, the node states were amended to be the same as the discretised values. The EM algorithm (Dempster, Laird and Rubin, 1977; Lauritzen, 1995) which GeNIe uses, can deal with missing values, however incomplete records had already been removed from the dataset in R. The algorithm starts with uniform distributions for all parameters across the network, thereby disregarding any parameters already present (BayesFusion LLC, 2019). The parameter learning was executed and the following probabilities of lateral deviation at five airports was calculated from the network.

	Airport	Probability
1	LHR	0.255
2	BHD	0.246
3	DME	0.212
4	EDI	0.179
5	DLM	0.144

Table 42: Causal network with learnt parameters airport ranking

The resulting network ranks the airports in the same order as all of the learnt networks, as shown in Table 42.

Performing sensitivity analysis on the network shows that the posterior probabilities for *LATGsd* are most sensitive to *rLandXwind* and *metXgust*, and least sensitive to *GSlope150* and *maxG* (see Figure 55).

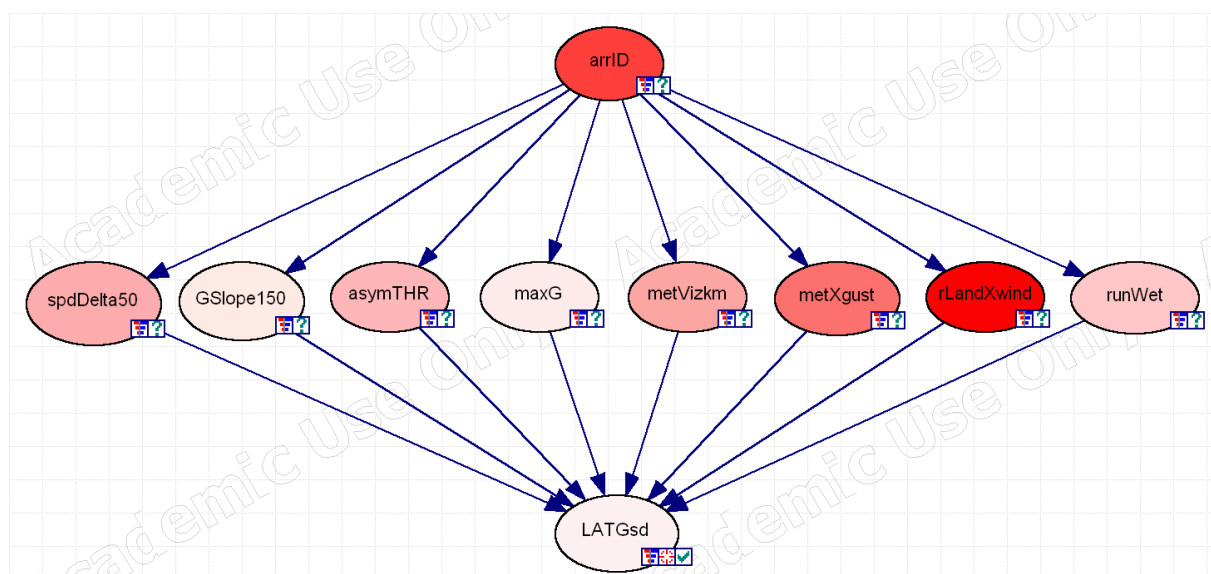


Figure 55: Sensitivity analysis for network with learnt parameters

This network is appealing because it has an intuitive structure and scenarios for different airports can be calculated easily. Likewise there are no surprises in the sensitivity analysis, with the lateral deviation parameter *LATGsd* being most sensitive to changes in crosswinds, as would be intuitively expected. However, these characteristics do not say much about how the network actually performs. Given an input set of conditions, how good is the network at predicting the outcome conditions? Fortunately, when a network or its parameters have been learnt from data, it is possible to validate how well the network performs and this will be described next.

5.9 Validation of learnt networks

Cross-validation is often used to validate learnt networks and parameters and there are various methods available (e.g. leave one out (LOOCV), repeated random sub-sampling). A popular method, which is implemented in GeNIe, is k-fold cross validation. In k-fold cross-validation, the dataset is split into k number of subsets, one of which is used as the validation set, and the remaining $k - 1$ subsets used for training. The cross-validation process is repeated k times, using each validation subset once. There are different opinions about the optimal value of k ; too low a value can result in biased results, whereas high values can result in excessive computing time. A value of $k = 10$ is generally regarded as sensible (James et al., 2013; Kuhn and Johnson, 2013).

The following sections show the results of cross-validation carried out on the networks learnt so far. They are:

- Bayesian Search network with sample size = 500 as described in 5.5.1.2
- Greedy Thick Thinning network as described in 5.5.3
- Tree Augmented Naïve Bayes network described in 5.5.4
- Causal network with learnt parameters described in 5.8

K-fold cross-validation was carried out using $k = 10$, with *LATGsd* set as the class node in GeNIe and the *Uniformize* EM option selected. The validation dataset was the same as that used to learn the network, discretised as per 4.8.2.

5.9.1 Bayesian Search network validation

Table 43 shows the validation results for the Bayesian Search learnt network.

Overall	0.478729 (137414/287039)
s1_below_0	0.67232 (48245/71759)
s2_0_0	0.291119 (20891/71761)
s3_0_0	0.255947 (18367/71761)
s4_0_up	0.695546 (49911/71758)

Table 43: Bayesian Search network validation results

The table shows how good the network is at representing observations in each of the four discretisation groupings. The last row labelled *s4_0_up*, represents the highest level of standard deviation of lateral acceleration (*LATGsd*) i.e. the 25% highest observations.

The overall accuracy of the network at predicting *LATGsd* was just under 48%, however it did much better at predicting the top 25% of occurrences (labelled *s4_0_up*) with an accuracy of nearly 70%.

		Predicted			
		S1_below_0	S2_0_0	S3_0_0	S4_0_up
Actual	S1_below_0	48245	14925	6120	2469
	S2_0_0	25791	20891	14605	10474
	S3_0_0	11114	16732	18367	25548
	S4_0_up	2517	6829	12501	49911

Table 44: Validation confusion matrix for Bayesian Search network

Table 44 shows the validation confusion matrix for the Bayesian Search network. The matrix shows correct predictions along the diagonal in bold, and errors in the other cells. Using the lower left cell as an example, the table is showing that 2,517 observations were incorrectly classified as having the lowest probability of lateral deviation, when they should have been in the top level, *s4_0_up*.

Receiver operating characteristic (ROC) curves are used to show the ability of a model in classifying binary systems, for example in classifying correct versus false predictions. The ROC curve in Figure 56 shows good performance for the Bayesian Search network in predicting the highest *LATGsd* values. The grey diagonal represents random performance and, informally, the closer the green curve is to the top left corner, the better the performance. The area under the curve (AUC) is around 0.84; perfect performance would result in AUC = 1. A full explanation of ROC curves can be found in Carvalho et al. (2014) and Fawcett (2006).

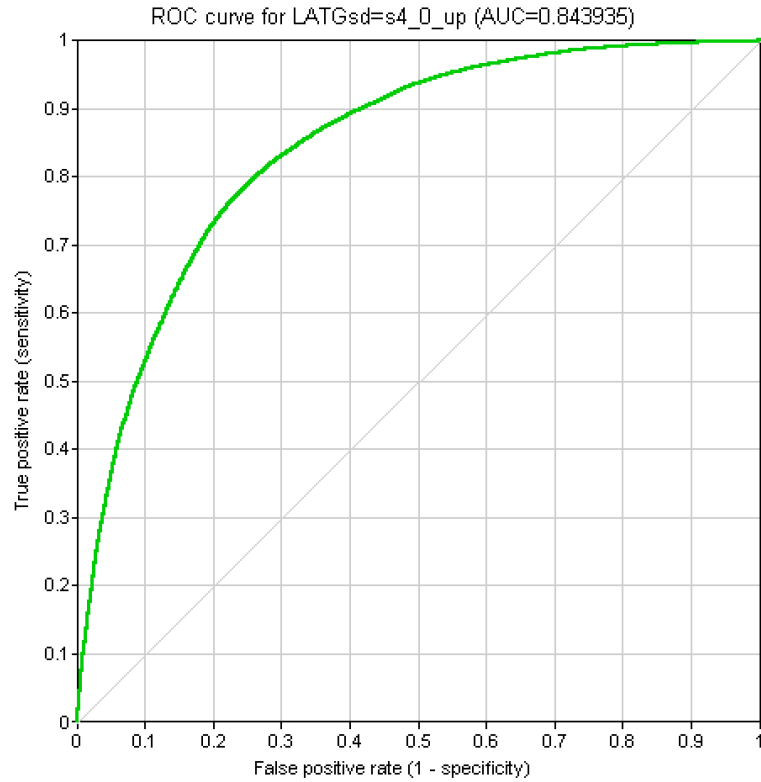


Figure 56: ROC curve for Bayesian Search network validation

5.9.2 Greedy Thick Thinning network validation

Validation of the Greedy Thick Thinning network shows very similar performance to that of the Bayesian Search network, with overall accuracy of 48% and accuracy for the highest interval of *LATGsd* of 70%. The confusion matrix and ROC curve are also very similar.

Overall	0.479297 (137577/287039)
s1_below_0	0.650692 (46693/71759)
s2_0_0	0.311381 (22345/71761)
s3_0_0	0.25741 (18472/71761)
s4_0_up	0.69772 (50067/71758)

Table 45: Greedy Thick Thinning network validation results

		Predicted			
		S1_below_0	S2_0_0	S3_0_0	S4_0_up
Actual	S1_below_0	46693	16499	6050	2517
	S2_0_0	24319	22345	14531	10566
	S3_0_0	10195	17468	18472	25626
	S4_0_up	2248	6978	12465	50067

Table 46: Validation confusion matrix for Greedy Thick Thinning network

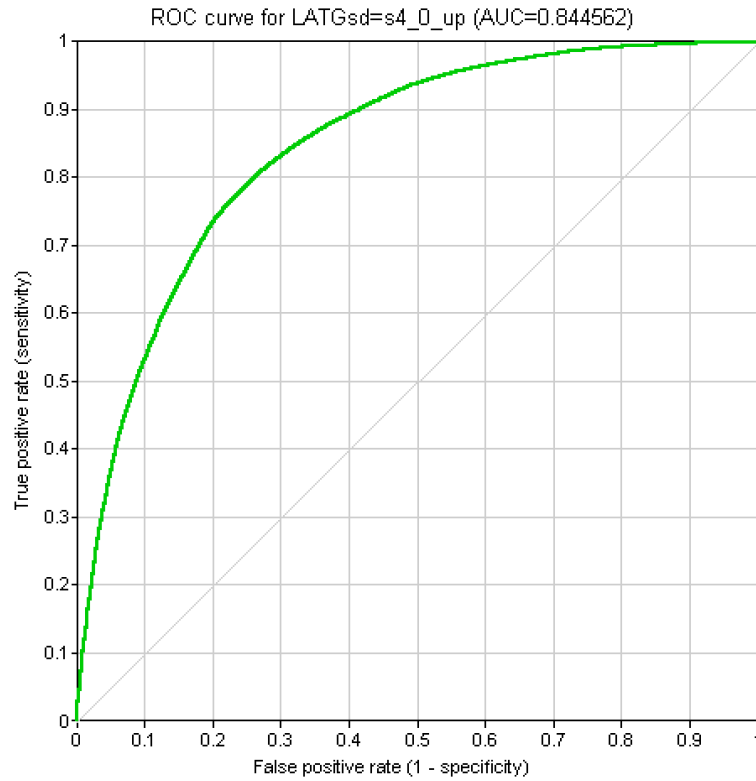


Figure 57: ROC curve for Greedy Thick Thinning network validation

5.9.3 Tree Augmented Naïve Bayes network

The validation results show similar overall performance to the previous two networks, with a slight drop in the highest *LATGsd* interval to 67%.

Overall	0.477036 (136928/287039)
s1_below_0	0.655737 (47055/71759)
s2_0_0	0.297237 (21330/71761)
s3_0_0	0.287273 (20615/71761)
s4_0_up	0.667912 (47928/71758)

Table 47: Tree Augmented Naive Bayes network validation results

		Predicted			
		S1_below_0	S2_0_0	S3_0_0	S4_0_up
Actual	S1_below_0	47055	15495	6850	2359
	S2_0_0	24890	21330	15810	9731
	S3_0_0	10802	16697	20615	23647
	S4_0_up	2475	6796	14559	47928

Table 48: Validation confusion matrix for Tree Augmented Naive Bayes network

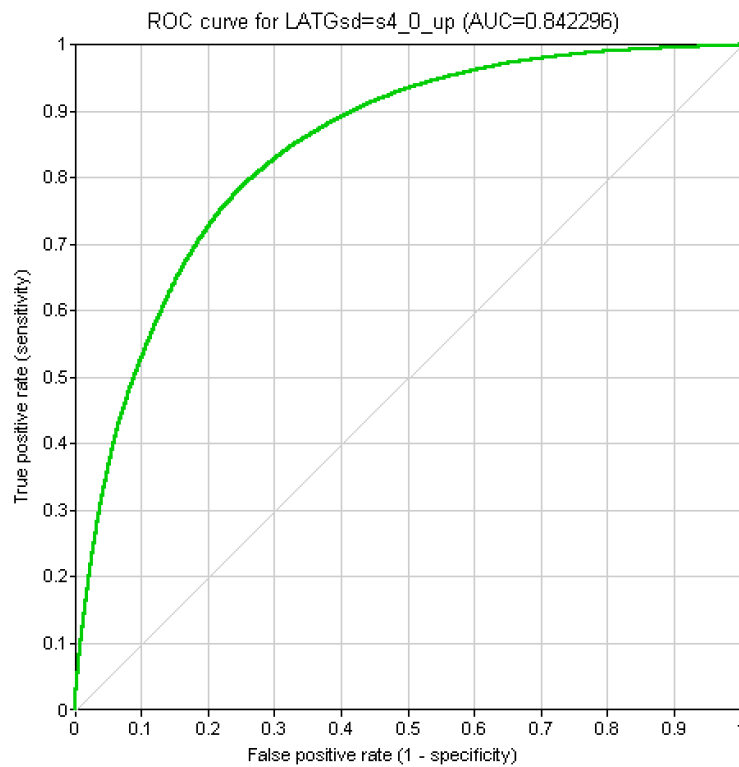


Figure 58: ROC curve for Tree Augmented Naive Bayes network validation

5.9.4 Causal network with learnt parameters validation

The causal network does not perform as well as the other networks, with overall accuracy at 39% and for the highest *LATGsd* interval only 53%.

Overall	0.388135 (111410/287039)
s1_below_0	0.635976 (45637/71759)
s2_0_0	0.144619 (10378/71761)
s3_0_0	0.244854 (17571/71761)
s4_0_up	0.527105 (37824/71758)

Table 49: Causal network with learnt parameters network validation results

		Predicted			
		S1_below_0	S2_0_0	S3_0_0	S4_0_up
Actual	S1_below_0	45637	9647	10699	5776
	S2_0_0	33276	10378	15477	12630
	S3_0_0	23260	9228	17571	21702
	S4_0_up	12346	6207	15381	37824

Table 50: Validation confusion matrix for causal network with learnt parameters

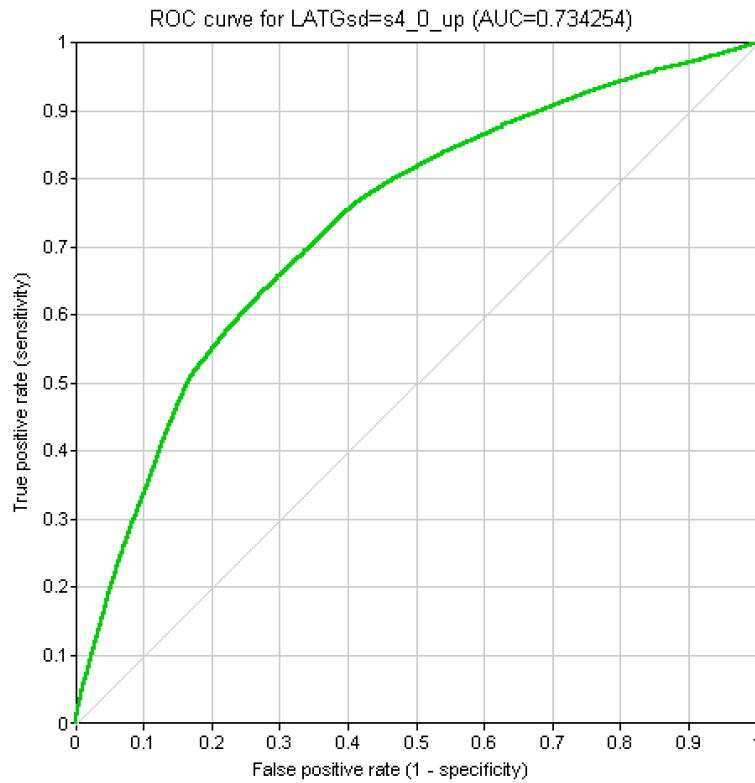


Figure 59: ROC curve for causal network with learnt parameters validation

5.9.5 Greedy Thick Thinning on two-level discretised data without background knowledge

The causal factor network from Chapter 4.6 used Boolean nodes for the causal nodes, so for the sake of completeness and consistency, another network was learnt using the Greedy Thick Thinning algorithm (which has proven to be a good candidate) and data discretised into two levels. The discretisation thresholds were chosen to mirror the probabilities derived in 4.6.4.3, e.g. *speed* discretised into <20kt and >20kt, *maxG* into <2.1g and >2.1g. The network was learnt

first without using background knowledge and then repeated with background knowledge. The resulting network had some arcs which were causally inconsistent so these were amended as follows:

- Removed *RUDsd* to *metVizkm*
- Removed *rLandXwind* to *runWet*
- Reversed *arrID* to *RUDsd*
- Reversed *RUDsd* to *metXgust*
- Added *rLandXwind* to *metXgust*

The network is shown in Figure 60.

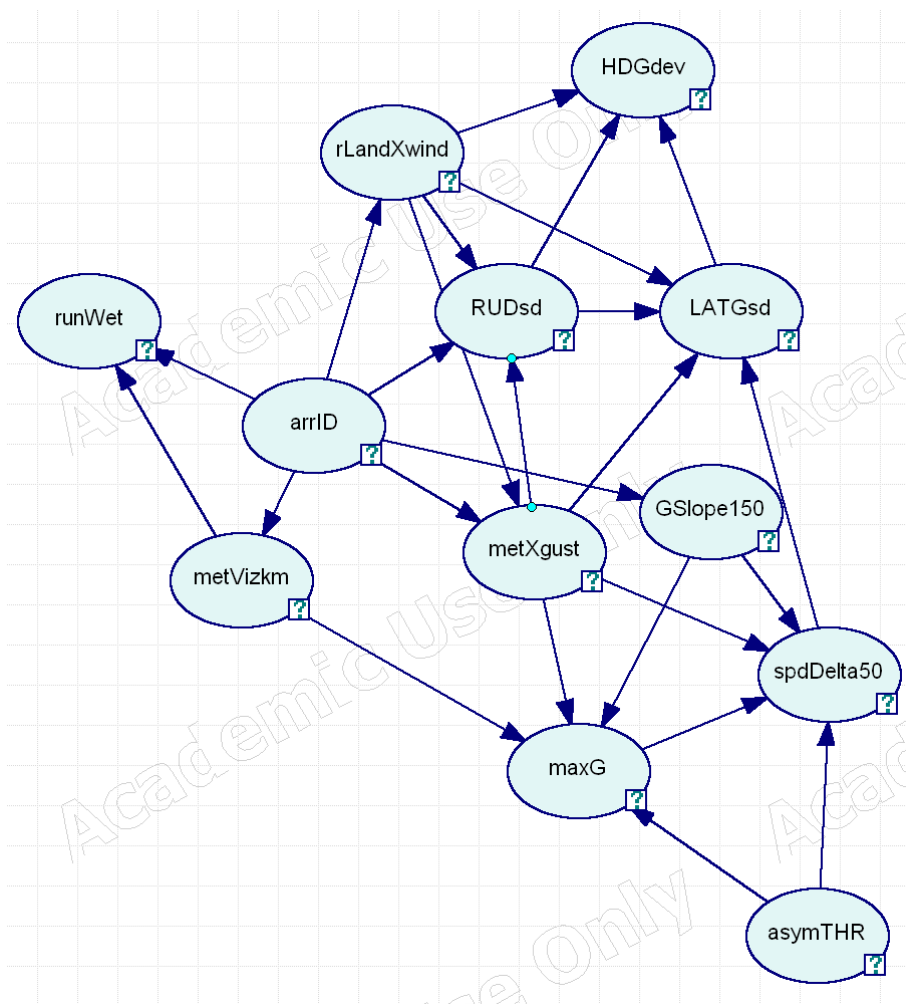


Figure 60: Greedy Thick Thinning learnt network from two-level discretised data without any background knowledge

Table 51 shows that the overall accuracy of the network is very good with nearly 91% of instances predicted. However, this is a result of the large difference in sample sizes between the discrete categories i.e. around 90% of values lie in the lower *LATGsd* discretisation window.

Looking at the prediction score for the higher *LATGsd* category shows poor performance with just 17% accuracy. This is further highlighted in the confusion matrix (Table 52).

Overall	0.906159 (260103/287039)
s1_below_0	0.987398 (255190/258447)
s2_0_up	0.171831 (4913/28592)

Table 51: Validation accuracy results of Greedy Thick Thinning network *LATGsd* node with two-level discretisation

		Predicted	
		S1_below_0	S2_0_up
Act.	S1_below_0	255190	3257
	S2_0_up	23679	4913

Table 52: Confusion matrix for Greedy Thick Thinning network with two-level discretisation

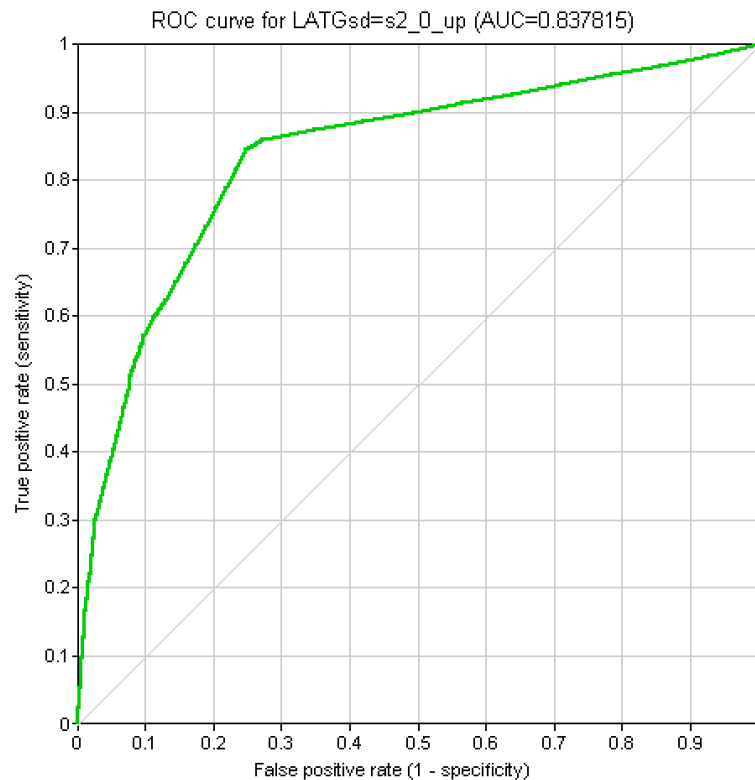


Figure 61: ROC curve for Greedy Thick Thinning network learnt from two-level discretised data

The ROC curve for the network is shown in Figure 61 and is somewhat deceptive in showing good performance. If the overall accuracy of the network was of most importance, the performance would indeed be very good, however as prediction of the more extreme values is of most importance here, the performance is actually quite poor. This highlights that the interpretation of ROC curves is sensitive to context and care must be taken not to use them in isolation.

5.9.6 Greedy Thick Thinning on two-level discretised data with background knowledge

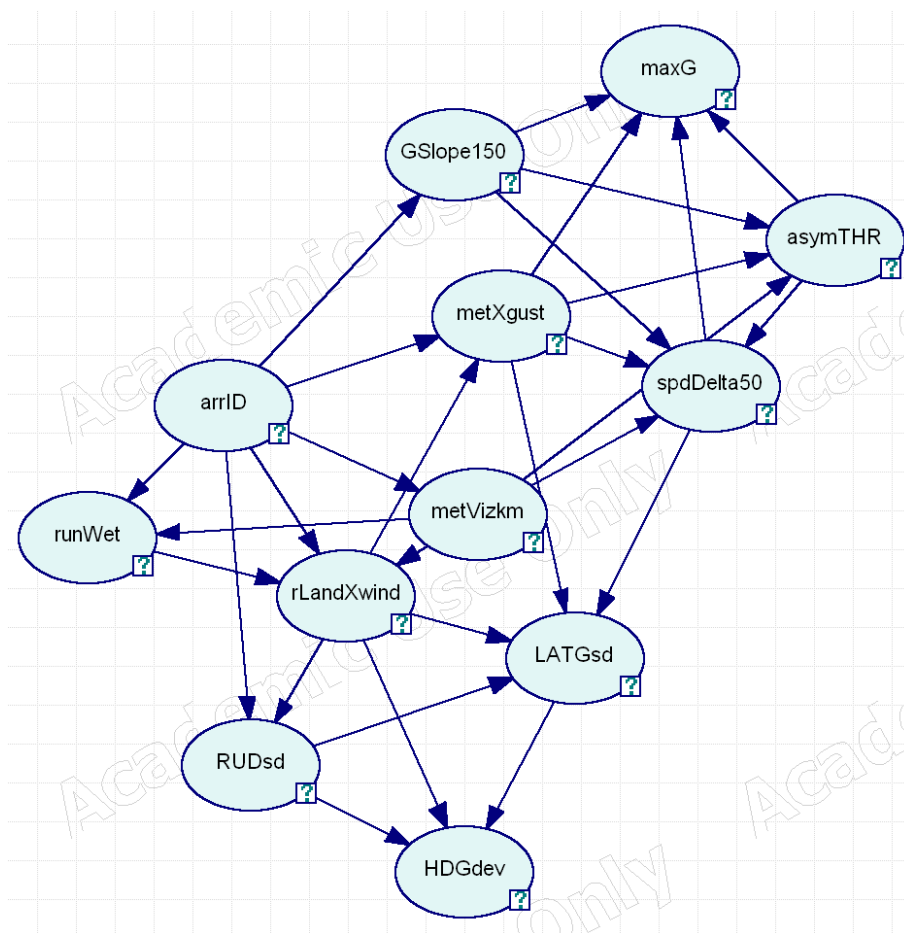


Figure 62: Greedy Thick Thinning learnt network from two-level discretised data with background knowledge

Again, for the sake of completeness and comparison, a network was learnt from data using Greedy Thick Thinning, but this time background information was made available. This resulted in the network above and avoided the necessity to remove or amend spurious arcs, however the overall network performed no better with identical validation scores to the previous network.

5.9.7 Summary of validation results

The causal network with learnt parameters performed relatively poorly, meaning it is not good at predicting the lateral deviation measure, *LATGsd*. The network structure was imposed, based on accident data causal factors, and just the parameters of the nodes were learnt. It seems that better accuracy can be attained by allowing the parameters and the structure to be learnt from data. As Table 53 shows, there is very little difference between these networks.

Network	High <i>LATGsd</i> accuracy (%)	ROC AUC
Bayesian Search	69.6	0.844
Greedy Thick Thinning	69.7	0.845
TAN Bayes	66.8	0.842
Causal with learnt params	52.7	0.734
GTT two-level discretised	17.2	0.838

Table 53: Summary of validation results

The Bayesian Search, Greedy Thick Thinning and Tree Augmented Naïve Bayes, performed similarly, however the Greedy Thick Thinning is marginally the best. The sensitivity analysis of the Bayesian Search network showed that *LATGsd* was not very sensitive to gusting crosswinds (*metXgust*), which is counter-intuitive. This problem did not arise in the Greedy Thick Thinning network and therefore for this reason and its overall performance, it is the preferred method for learning.

6 Discussion

There are two main issues at the heart of the rationale for this research:

1. That the airline industry, particularly in the operations domain, does not truly quantify risks
2. That the vast amounts of data recorded routinely during flight operations is not fully exploited to help quantify risks

The literature includes methods which purportedly quantify risk, such as ARMS, however there is a subjective element at the core which leads to qualitative assessment in the end. In many ways this is not surprising as traditional FDM exceedance monitoring does not provide results which readily map onto the typical risks an airline faces, hence it makes it difficult to provide robust probabilities for a quantitative risk assessment. If the hundreds or thousands of parameters recorded during every second of every flight could be exploited to yield information more readily adapted to quantifiable risk assessment, flight data monitoring would take a leap forward in utility and airline operations would benefit from quantitative risk assessments of the hazards the industry faces.

Early on in the research, several potential methods for quantifying risk using flight data were explored, amongst them was looking into the variability of processes. In principle, treating stages of a flight from A to B as a process is valid because in airline operations there is usually a prescribed and optimum way of carrying out certain parts or phases of flights, such as the take-off, approach to landing, the landing itself and abnormal situations such as go-arounds. Standard operating procedures and training manuals dictate how these manoeuvres should be carried out. For example, during an Airbus A320 take-off, the thrust levers will be advanced to the appropriate gate, the sidestick will be held forward until a particular speed is reached, the rotation (at 3 degrees per second) will be initiated at the defined rotation speed (V_r) and the landing gear will be selected up when a positive rate of climb is achieved. Of course, this is simplified, but it demonstrates that there is a specific order in which actions occur, and that targets exist for optimal performance (e.g. rotation rate). This order and these targets are defined by aircraft manufacturers and airlines and are essentially stating “this is how this aeroplane should be operated”. In other words, there is a quest for standardisation, hence the existence of “standard operating procedures”.

6.1 Variability

The traditional methods of flight data monitoring, which have been around since the 1970s, use exceedances above or below a threshold to monitor compliance with these standards, but there is little, if any, attention paid to the variability in these processes. It could be argued that variability is allowable, as long as exceedance thresholds are not crossed, and that is right to some extent. With human beings in control of the process, which is itself subject to multiple external influences (e.g. weather), variability is inevitable. Looking at the variability of a single take-off will not provide much in the way of safety information, but looking at the variability of features from all take-offs over months or years, might yield safety information relating to changes in performance.

Methods traditionally associated with quality control could be applied to airline operations quite easily, particularly as many existing FDM software tools allow measures to be captured from every flight. Applying methods such as statistical process control to this data would be relatively straightforward and once the variation in an operational process can be quantified, the reasons for that variability can be investigated. It can be said that we are in control of a process if we can show that the variations arise from random or common causes (Oakland, 2012).

Adopting these methods would require a change in mindset, given that the industry has been looking at FDM data in much the same way for several decades. However there should not be much resistance if the new methods are used to compliment, rather than replace, existing methods. A set of standard variability features could be compiled, in much the same way exceedance events are in traditional FDM, and the standard deviation of each feature could be reported as a safety performance indicator. For example, one such feature could be deviation from target approach speed at 1NM from touchdown. The overall change in standard deviation could be tracked as a time series and the standard deviation for individual airports or runways could be presented. Any anomalies identified could be targeted by the airline's training department, who could focus on specific techniques to try to reduce variability. If the right data safeguards were in place, and if the pilot workforce were agreeable, variation between individual pilots could also be monitored, and potentially bespoke training or practice offered.

Adopting these methods would be technically easy, but a challenge would be in creating a meaningful set of measures for which more variability would be undesirable. Through the involvement of subject matter experts, these challenges could be overcome and FDM would have an additional toolset with which to extract more safety information from existing data.

A further, and more difficult challenge is that of validation. It has been demonstrated that ranking airports by variability of features during approach to landing did not correlate well with FDM threshold exceedance methods. It was posited that some similarity in rankings would be expected i.e. more variability would lead to more threshold exceedances. That could well be true, however the effect could be more subtle and, importantly, as the whole point of the method would be to enhance knowledge discovery rather than to just replicate results from existing methods, it is therefore recommended that the weak correlations should not lead to the dismissal of the method. Whether the method will actually lead to enhanced safety knowledge discovery will only be determined in practical use. If the safety management system within an airline benefits from additional insight gleaned from variability measures, the method clearly has value, and due to the relative simplicity in adopting the method, it would seem sensible for a trial to be carried out.

6.2 Workload

Unstable approaches are often cited as a causal factor in landing accidents and during investigations⁷ into unstable approaches, pilots often describe the approach as having been “busy”. This leads to a hypothesis that workload could be a useful safety indicator, possibly highlighting approaches to airports where safety margins are tighter or where there is less resilience to deal with the unusual or unexpected. Routine monitoring of actual workload is impracticable, however it can be inferred to some extent from recorded flight data. Much depends on the availability of flight data parameters which can be definitively linked to an action carried out by a pilot, such as movement of a lever or a switch selection. Counting these actions during the final stages of the approach does correlate with “busy” approaches”; it was shown that the approaches with the highest number of actions were non-precision approaches, a type of approach that does require pilots to do more. However, flight data can only count a subset of tasks and actions actually carried out. For instance, the reading of a checklist or the briefing for the approach would not be recorded in flight data. Nevertheless, the knowledge that approaches into a specific airport are busier than any other would be useful in understanding the wider risk picture.

⁷ Investigations carried out by the author as part of his duties while employed at an airline.

6.3 Bayesian networks

The literature on quantitative risk assessment quickly leads the reader to Bayesian networks and works such as Fenton and Neil (2012) whose book takes a practical approach to Bayesian risk assessment, which the practitioner can relate to everyday risk problems. There are other methods that could be used, such as artificial neural networks, but the transparency, flexibility and intuitiveness of Bayesian networks makes them a very appealing solution to risk assessment, especially when there is uncertainty involved. A quality of Bayesian networks is the ability to model problems where there are both known and unknown quantities. Known quantities can be derived from data and can be the basis prior probabilities in the Bayesian network. Unknown quantities can be estimated and the uncertainty surrounding them can be reflected using prior probability distributions. These quantities can also be elicited from subject matter experts and again, the uncertainty surrounding estimates can be reflected in the model. Once more data or evidence is available, the network can be updated resulting in an updated set of outputs across the network. These qualities make Bayesian networks quite intuitive as they act much like human reasoning: our beliefs are updated in light of new evidence.

Bayesian networks are based on Bayes' Theorem (see Eq. 1). The accident statistic literature (e.g. investigation reports and work such as Post (2015)) has quite comprehensive data on the probability of a particular accident, such as a veer-off ($P(VO)$), and investigation reports usually summarise the causal factors involved in any particular accident ($P(CF/VO)$). However these two pieces of information on their own are of limited use to the risk analyst. The risk analyst really wants to know that, given the presence of an accident causal factor, what is the probability of an accident ($P(VO/CF)$). With this information, decisions can be made regarding the avoidance of those causal factors or improved mitigations against them. Fortunately that information can be calculated with Bayes Theorem, but only if another missing piece of information is available: the prevalence of, or exposure to, the causal factor $P(CF)$.

Deriving $P(CF)$ can be a real stumbling block in risk assessment and often relies on the elicitation of expert opinion and can therefore be subject to the biases that entails. Core to this research is using airborne recorded flight data, FDM data in other words, to estimate $P(CF)$ and thereby allow the risk analyst to quantify $P(VO/CF)$.

6.3.1 The problem of validation

If the causal factors relating to an outcome are known, it is usually quite easy to define a Bayesian network structure as the cause-consequence relationship is usually intuitive and the causes are often well documented and understood. This means that the resulting model is

readily interpreted by those not involved in its construction and those without prior experience of Bayesian networks. Datasets, such as those from FDM, can be used to define probabilities of the causal factors occurring, but while the probability of the consequence is known at say, an industry level, it may not be known at an individual airline level. Thanks to the ultra-reliability of airline operations, data for the modelled consequence will probably be unavailable. In other words, even with a dataset of millions of flights, due to the rarity of events such as runway veer-off (one of the more common accident types at $1.2561\text{e-}07$ per flight), the dataset is unlikely to contain even one example of the consequence. This presents a problem in terms of validating the model. Deriving a number from a Bayesian network for the probability of an undesirable consequence is all well and good, but is it any good? How can others be persuaded that this probability should be used to help safety-critical decision making or direct resources which are in-demand?

It might be assumed that validating causal networks is trivial, given that their structures and variables are often easily understood, however this is not the case. The lack of consequence data means that a proxy for the consequence needs to be used. The proxy could be a symptom of getting close to the consequence or an unusual deviation towards it, and in some cases the proxy might be easily available. For example, the research here has concentrated on runway veer-off and encountered the problem of no runway veer-offs being available in the dataset. An obvious measure that could be used as a proxy to veer-off would be deviation from the runway centreline. If a high resolution (say 2m) GPS position was recorded in the dataset, validation would be more straightforward. However, no such data was available, necessitating the identification of some other, more obscure proxy. Perhaps this is just a peculiarity of runway veer-off events? Unfortunately not; take runway longitudinal excursions for example. Identification of the touchdown point is relatively easy and it is possible to integrate the aircraft's groundspeed to calculate the distance of the ground roll, but once again, without an accurate position it is very difficult, if not impossible, to accurately determine how far down the runway the aircraft touched down.

This research has highlighted that flight data recordings which include accurate position parameters (latitude and longitude) recorded at 1Hz or more with good resolution, would be beneficial to FDM for routine monitoring purposes, but also for validation of new, more complex methods.

Another potential proxy was localiser deviation. The localiser provides lateral guidance to pilots during precision approaches and deviation from the runway centreline could potentially be calculated trigonometrically. However, not all runways in the dataset are equipped with

localisers and once again, the recorded resolution of the parameter is inadequate to provide any useful level of accuracy. Thus an alternative proxy had to be identified.

A parameter which is recorded at a useful resolution and at a good rate (4Hz) is lateral acceleration (relative to the aircraft's longitudinal axis). As was seen in 5.2.1, landings with a higher range of lateral acceleration values during the landing roll, does correlate with landings carried out when veer-off causal factors are present, hence the standard deviation of lateral acceleration was used as a proxy for lateral deviation. There are two important caveats to note:

1. Consider an extreme example of a landing made well off the centreline, close to the runway edge. If the aircraft tracked along the edge of the runway for the entire landing roll-out, there would be very little lateral acceleration despite being very close to a veer-off event. This demonstrates a limitation of the lateral acceleration proxy. In reality, it would be likely that the pilot would make prompt attempts to recover the centreline, thereby producing lateral accelerations.
2. Landings on slippery, contaminated runways might result in lower lateral accelerations because of the lack of grip available. Lateral accelerations might be produced predominantly by aerodynamic forces, which in turn might lead to a reduced standard deviation of lateral acceleration, despite a relatively high risk of veer-off due to the contamination.

These two potential issues are rare scenarios in the dataset used in this research, however they highlight the need for a better measure of lateral deviation, both for validation and parameter learning, especially for airlines operating to airports where contaminated runways are common.

6.3.2 Network structure – learnt or applied?

In the Literature, it is often the case that the researcher has data available with which to model a Bayesian network, or there is little or no data available and the network structure and parameters must be elicited from experts. This research had both a large dataset available and the causal factors for the example accident scenario, runway veer-off, were well documented and available, thus comparisons could be made.

The comparisons were made in terms of the causal logic of the resulting networks, the ranking of probabilities derived from the network, sensitivity analyses and a series of validation measures. More theoretical analyses of network performance can be found in the Literature (e.g. Heckerman, Geiger and Chickering (1995)).

There is no doubt that the network structure derived from the accident report literature (Post, 2015), herein referred to as the *applied causal network*⁸, is easy to interpret, as it is essentially a series of causal factors nodes all pointing towards the consequence node, however, herein lies a problem. With so many parent nodes, the consequence node suffers from having an unwieldy node probability table, in this case over 2,000 cells due to the 11 parent nodes. For practicality, this necessitates the use of a *noisy-or* function (Pearl, 1988b). For the causal network modelled here, the *noisy-or* expression which was entered into the lateral deviation node definition in AgenaRisk, contained a factor to bring the probabilities of the causal factors into the same order of magnitude as the worldwide runway veer-off rate (1.2561e-07). In other words, the weighting values for each causal factor in the *noisy-or* expression were calculated by multiplying the causal factor frequency percentage by the worldwide veer-off rate. This meant that the resulting probabilities for the lateral deviation node were scaled similarly to the global rate. There is a weakness here, in that the global rate is perhaps not representative of the rate within the subject airline. It might be that the subject airline operates in a region of the world where there are fewer runway veer-off events due to environmental factors or pilot training, thus the overall probability of lateral deviation should be lower. Ideally, the runway veer-off rate for the subject airline would be used, but runway veer-offs are rare events and no rate data is likely to exist. Using a rate of zero makes no sense, as it would result in a zero probability of veer-off, which is unrealistic. As an alternative, it might be possible to generate regional statistics for runway veer-off, rather than relying on the global picture. A problem with this is that airline operations often traverse regions, so a global statistic is probably the best available in many cases.

Using this multiplication factor (i.e. the background veer-off rate) produces probabilities which are in the right order of magnitude, and thus act as a reminder of the true likelihood of these occurrences, but beyond that, is there any point in including the worldwide rate? For most safety management applications, the airline would be most interested in where they are most likely to experience a veer-off and how their performance at mitigating veer-offs is changing over time. These are relative measures which do not actually require the calculated probability to be in the right order of magnitude.

A potential issue with the use of *noisy-or* functions is that they require the causal factors to be independent, or more specifically, no one causal factor should diminish the influence of another (Pearl, 1988a). In the case of this research, each causal factor *can* occur independently, so this condition is satisfied. Note that “*can*” is not the same as “always does”; heavy rain will nearly

⁸ The term “applied causal network” is used here to describe the network for which a previously known structure and parameters have been applied to the network, rather than learnt from data.

always result in a wet runway, for example, and they often will occur together, however runways remain wet after rain has passed, and a heavy rain shower can be localised to some areas of a large airport. But even in this example, neither causal factor is acting in a way to diminish the effect of the other, therefore the condition of independence for *noisy-or* holds. If this were not the case, a *recursive noisy-or* function (Anand and Downs, 2008; Ashrafi, Davoudpour and Khodakarami, 2017; Lemmer and Gossink, 2004) could be used. While *noisy-or* functions can be found in some Bayesian network software tools, *recursive noisy-or* are still not common.

Applied network structures, whether based on expert knowledge or causal factor literature, might not be the best solution for modelling the data they aim to represent. This was the case here, with the adapted causal factor network scoring badly in validation (see 5.9.4) when compared with learnt networks. The structure of this network was based on the causal factor literature, however the parameters for the nodes were learnt from data. It is interesting to explore why this network performed relatively badly compared to those networks where both the structure *and* the parameters were learnt from data.

For both types of network, data was available to learn the node parameters, which suggests that the difference lies in the structure. The literature on *veer-off* causal factors is comprehensive in identifying which causal factors are most common, but it does not explore the interaction of those causal factors. It was necessary to assume that all causal factors can occur independently to satisfy the requirements of the *noisy-or* function used in the causal network, which is true, they can. However, in reality they often occur together e.g. low visibility often accompanies precipitation. The learnt networks can reflect these interactions in their structure and indeed it is obvious that the wholly learnt networks are more complex than the theoretical *applied causal network*.

None of this is intended to say that the *applied causal network* is less valid than the wholly learnt networks, it is just that they are better at representing real-world data, the data they were learnt from, which is probably not surprising!

6.3.2.1 Comparison of results

The applied causal factor network ranked the five airports for lateral deviation probability as follows:

1. Dalaman (DLM)
- =2. Belfast City (BHD)
- =2. Edinburgh (EDI)
- =2. London Heathrow (LHR)

3. Moscow (DME)

All of the learnt networks, including the causal network with learnt parameters, ranked the airports as:

1. London Heathrow (LHR)
2. Belfast City (BHD)
3. Moscow (DME)
4. Edinburgh (EDI)
5. Dalaman (DLM)

There is little similarity and there is a significant difference in the position of DLM. This could suggest some sensitivity in the learning algorithms to the number of cases available. The number of landings at each airport was:

1. LHR – 144,849
2. BHD – 17,487
3. EDI – 17,038
4. DME – 2,375
5. DLM – 622

The number of cases correlates well with the learnt network rankings, although DME and EDI have switched places. It could be that the higher probability of lateral deviation, as found in the causal network, has outweighed the number of cases effect and pushed DME up the table, however this is conjecture. To investigate further, the network and parameter learning could be restricted to a dataset which contained equal numbers of cases (i.e. landings) at each airport.

Another variable affecting the performance of learnt networks is how the data was discretised. Eventually it is likely that more and more Bayesian network software tools will support truly continuous distributions to define nodes, but at the moment most only support a limited set of probability distributions and/or require the distributions to be discretised before calculations can be made. It was seen that a higher level of discretisation appeared to result in a kind of overfitting of the networks to the data, characterised by nodes connected by dubious arcs to other nodes. Reducing the discretisation to just two levels, similar to the way the *applied causal network* nodes were dealt with, resulted in a poor performing network, according to the validation.

The question arises, which is best, applied or learnt? Both methods have advantages and disadvantages and both have difficulties in terms of the validation of results.

6.3.2.2 Advantages of the applied causal network

The structure of the *applied causal network* is relatively simple and reflects the causal relationship between causal factor and outcome. This means it can be easily understood and because the structure is based on past accidents, it is more easily defensible. Another important advantage is that new causal factors or escalation factors can be added to the network relatively easily once they are identified, without any reliance on data being available. This is reflected in the network created here in 4.6.2, which has nodes for which no data was available for the learnt networks.

Another important advantage is that the network makes use of all the causal factors. For several of the learnt networks, some causal factors had no child nodes, thus not contributing to the probability of the lateral deviation. For example, the Greedy Thick Thinning network in 5.5.3 had *maxG* and *asymTHR* as nodes without children. Given that both of these are documented causal factors in some veer-off accidents, the omission of arcs to lateral deviation nodes is problematic. The reason why this occurs is quite simply that the dataset does not contain instances of where *asymTHR*, for example, has influenced other variables. That does not mean that it could not in future, a fact demonstrated by the presence of asymmetric thrust as a veer-off causal factor in the accident literature. A solution to this could be to force arcs between known causal factor and consequence nodes, which is discussed in 6.3.3.

The method of modelling real data to provide probabilities of extreme events, as in 4.6.4, is time consuming, however it does mean that the network reflects, as much as is possible, the potential full range of values of causal factor variables, and not just those seen in the available dataset.

6.3.2.3 Disadvantages of the applied causal network

One of the most fundamental disadvantages is that this type of network relies on well documented and researched causal factor data. Runway veer-offs are one of the most common aircraft accidents, so it's not surprising that literature exists, however if we were interested in risk assessing a less common accident type, it might be difficult to identify causal factors. Alternatively, some causal factors may be documented, but they might be nowhere near as complete as those for the more common types.

Another disadvantage is that they are difficult to validate, although this can be said of any methodology which attempts to risk assess rare events. Here, a proxy for lateral deviation had to be used in place of veer-off events which are fortunately relative rare. The proxy provided a limited level of validation, however this could be improved if a better measure of lateral deviation was available in the recorded flight data. Note that this issue does not negate the

overall method, as the lack of a direct proxy for veer-off accidents, is not necessarily the case for other accident types.

Probably the biggest disadvantage of all, related to one of the advantages, is the time-consuming process of modelling probability distributions for the causal factor nodes. It means that if the analyst is trying to compare different airports, probability distributions for each node for each airport need to be modelled. It is possible that this process could be automated by software, with the probability distributions being continuously updated as new data is received. Even if carried out manually, this research found the process to be faster for subsequent airports as only the parameters of the distribution, and not the underlying distribution, needed to be changed.

6.3.2.4 Advantages of learnt networks

Learnt networks can avoid the problems of having to model nodes for different airports by including the airport in the learning dataset. The airport is included as a node and the parameters for all nodes are learnt with the airport as a variable. Therefore, to calculate probabilities relating to specific airports, a particular airport can be entered as evidence in the airport node, which greatly improves the speed of running scenarios for different locations.

Learnt networks also have the advantage of potentially identifying nuanced interactions between causal factors which may not be readily apparent from the accident literature. It could even be the case that there is potential for interactions unknown to investigators to be identified. They are also much less time-consuming to construct, as algorithms learn the structure and parameters of the network from the data and the subject matter expert can then adapt arcs as necessary. There are several choices to be made, in terms of the algorithm to use and the relevant parameters, but as has been shown, several of the tested algorithms gave very similar results. Validation is also easier, as algorithms such as K-folds cross validation exist which can provide metrics of how well the network describes the data. This kind of validation is extremely important for networks where the underlying logic of how the nodes are connected is unknown, but for causal networks it is perhaps not so critical as the relationships are better understood.

6.3.2.5 Disadvantages of learnt networks

Probably the biggest disadvantage of learnt networks is that they are tailored to the dataset available. This means that extreme values, which could potentially occur, are not accounted for in the probabilities calculated in learnt nodes, because the dataset has not contained those extremes. Say for example, a dataset contains three landings at 2.0g, one at 2.2g and one at 2.3g. The samples at the tail end of the empirical distributions are sparse, but there is clearly a chance

that 2.1g landings could occur and there will be some diminishing probability that landings for values greater than 2.3g will occur. These probabilities will be underrepresented in the learnt network as they were not seen in the dataset, however the causal network, with modelled extremes, can represent them.

6.3.3 Combining applied and learnt networks

Given the advantages and disadvantages of the network types, could the methods be combined to produce a better network? Potentially this could provide a more nuanced network which reflects some of the interactions between causal factors, but at the same time ensures the arcs are in place to represent known causal links.

To test this, the Greedy Thick Thinning network from 5.5.3 was adapted to maintain the arcs learnt from data and add the causal arcs known from literature. The changes made were:

- Add:
 - *GSlope150* -> *spdDelta50* to ensure the known correlation between being high on the approach and fast is reflected.
 - *asymTHR* -> *LATGsd* to ensure the asymmetric thrust causal factor has influence
 - *maxG* -> *LATGsd* to ensure the landing acceleration causal factor has influence
 - *runWet* -> *LATGsd* to ensure the wet runway causal factor has influence
 - *metVizkm* -> *LATGsd* to ensure the low visibility causal factor has influence
- Remove:
 - *runWet* -> *rLandXwind* as this link is not causal
 - *metXgust* -> *asymTHR* as this link is not causal
 - *runWet* -> *metXgust* as this link is not causal
- Reverse:
 - *metXgust* -> *rLandXwind* as this seems the most appropriate representation; a crosswind exists, then crosswind gusts exists. The opposite order would not make sense.

The resulting network is shown below:

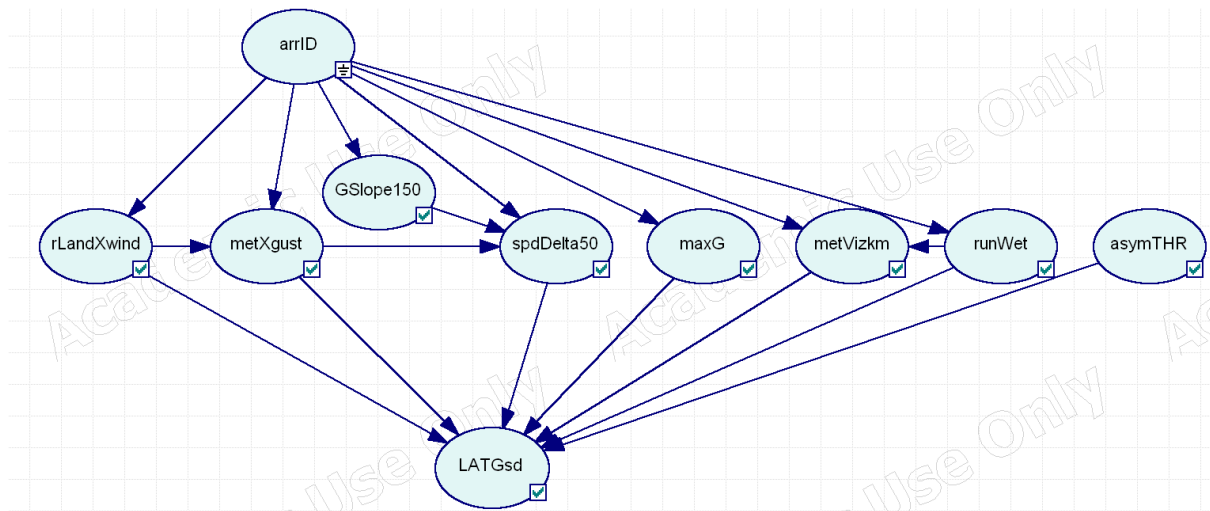


Figure 63: Learnt network structure modified to include missing causal arcs

The network has some additional arcs compared to the causal network in 5.8 and note that all causal factors influence the consequence node *LATGsd*. The parameters were learnt from data discretised into two bins, with thresholds selected to be the same as for the causal network in 4.6.4.3, so that comparisons could be made with that network. The node probability table which had been learnt, was replaced with a noisy-or function with the following parameters:

Parent	<i>rLandXwind</i>	<i>runWet</i>	<i>metXgust</i>	<i>asymTHR</i>	<i>spdDelta50</i>	<i>maxG</i>	<i>metVizkm</i>	LEAK
State	>10kt	Wet	Gusting	> 5s	> 20kt	> 21.g	< 5km	
Deviation	3.614e-07	3.398e-07	4.97e-07	3.14e-05	1.979e-06	2.475e-04	1.855e-07	2.085e-08
Normal	1 - Deviation							

Table 54: noisy-or parameters for combined network

The parameter values were the same as those used in the noisy-or expression in 4.6.4.3.11.

It is not possible to validate this network using the dataset, because the *LATGsd* node no longer contains learnt parameters, so validation is limited to the methods described in 5.2. Table 55 shows a comparison of the probabilities calculated from the wholly causal and the causal/learnt networks.

Airport ranking causal network	BN probability from causal network	BN probability from causal/learnt network
<i>DLM</i>	2.28E-07	3.11E-07
<i>BHD</i>	2.27E-07	1.68E-07
<i>EDI</i>	2.27E-07	1.30E-07
<i>LHR</i>	2.27E-07	1.23E-07
<i>DME</i>	2.11E-07	1.91E-07

Table 55: Comparison of lateral deviation probabilities calculated from wholly causal and combined causal/learnt networks

The ranking of airports for the causal/learnt network is:

1. DLM
2. DME
3. BHD
4. EDI
5. LHR

The probabilities are generally lower, apart from DLM which is increased and stays at the top of the ranking. Comparing the validity of these two networks is difficult, particularly as they do not have identical nodes. The wholly causal network has a node for *inaccurate information to the crew*, which is based solely on the difference between reported and actual winds at airports, and is the only additional node which differentiates between airports. It is likely that the presence of that node leads to some differences in rankings between airports as, for example, LHR has a higher empirical probability of inaccurate information than EDI. This should be investigated further in future work, with the *inaccurate information to crew data* added to the learning dataset.

It is also possible that higher probabilities of extremes, modelled in the wholly causal network, but not in the combined, could also lead to differences.

6.3.4 The recommended approach

The combined applied/learnt mixture approach produces similar results to the applied, wholly causal network and both should be validated further if better lateral deviation parameters become available. The ease of specifying a particular airport in the combined network is an attractive advantage and learning from data avoids the lengthy process of modelling each node.

However the process of modelling nodes for the wholly causal network will provide better probabilities for extreme events, which are not present in the learning data.

With these two points in mind, the recommended approach depends on the available dataset and the parameters available within that dataset. If the dataset is large (i.e. several hundred thousands of flights) and there are good parameters to measure the deviation of interest (e.g. high resolution position information), then learnt networks would be recommended. If the dataset is small, a causal network with modelled nodes will be more appropriate.

6.3.5 Dealing with the consequences

The general topic of this thesis is risk assessment. A recent definition of risk is that it is “*the effect of uncertainty on objectives*” (International Organization for Standardization, 2018), which, for operational risk assessment purposes, maybe over-generalises. The more traditional

definitions are that risk is a combination of the probability of an event occurring and the severity of the consequences of that event (International Civil Aviation Organisation, 2013). This can be enhanced by adding a third element: exposure. Looking at an individual flight, the probability and severity are sufficient to define the risk, but if a broader risk picture over several flights is needed, exposure is important. For example, flying an approach into an airfield with significant terrain might be high risk, but at an operational level the risk would be low if only one flight per year was operated there i.e. exposure to the hazard is low.

These definitions raise two important points with regard to the methods which have been described. The first is that the overall exposure to hazards has not been accounted for. This is fairly easily resolved by simply multiplying the base probability rate by the exposure. So for example, Moscow DME has a calculated prior probability of veer-off of $2.11\text{E-}07$; if there are 1,000 flights there per year, the adjusted probability is $2.11\text{E-}04$ per year. Some Bayesian network software tools allow quite advanced arithmetical and conditional statements to be included in node definitions, so it might be possible to include exposure data in the network itself, particularly for the learnt or combined networks which have an airport node.

The second is that the severity of potential consequences have not been factored into the output of the networks. This is important because the consequences of a runway veer-off are not the same at every airport.



Figure 64: Runway at Trabzon airport, Turkey. By Ercan Karakas - <https://www.jetphotos.com/photo/6012865>, GFDL, <https://commons.wikimedia.org/w/index.php?curid=20306566>



Figure 65: Runway at Funchal airport, Madeira. By Koshelyev - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=6207088>



Figure 66: Runway at Palm Springs International airport, USA. Photograph by D. Ramey Logan, CC BY 4.0, <https://commons.wikimedia.org/w/index.php?curid=39528001>

The three photographs in Figure 64, Figure 65 and Figure 66 show how the consequences of a veer-off might be very different. A veer-off at Trabzon (top) or Funchal (middle) could have catastrophic consequences due to the proximity of sharp drops towards the sea, whereas Palm Springs is relatively benign, with large concrete areas either side of the runway. That is not to say that a veer-off at Palm Spring could not be catastrophic; a veer-off could result in a collision with a taxiing aircraft, for example.

ICAO defines the international standards and recommendations for aerodromes and runways in Annex 14 to the Convention on International Civil Aviation (International Civil Aviation Organisation, 2018). The runway and runway shoulder width characteristics are defined as recommendations. Runway strips, which include the runway and stop-ways, have a stated minimum width of 70m or 140m depending on aircraft size, but this requirement is stated as “*wherever practicable*”, thus there is no regulatory guarantee that the airport the airline is flying to is compliant with ICAO standards and recommendations.

Thanks to the flexibility of Bayesian networks, dealing with the differences in runway characteristics and the potential consequences of a veer-off can be modelled using additional nodes which are conditional on the lateral deviation having occurred. These additional nodes can be thought of as an Event Tree Analysis (Fenton and Neil, 2013), where a hazardous event has occurred (in this case, a lateral deviation), which then leads to potential consequence nodes which are conditional on the lateral deviation.

Adding consequence information to the network would be useful for comparing the relative risk between airports at a systemic level, but it could be problematic because it would be difficult to define the conditional probabilities for the consequence, however they could be elicited from experts.

6.3.6 Applicability

6.3.6.1 Aircraft

The methods described in this thesis have made use of flight data recorded onboard Airbus A320 series aircraft, but data from almost any aircraft capable of recording flight data could be used. Since 2005, flight data monitoring has been a mandatory practice for operators with aircraft which have a maximum take-off weight of more than 27,000kg, which means that all these aircraft must have some means of recording flight data, and hence these methods could be applied. The quality of the recorded data frames can vary, as seen here with the lack of an adequate direct measure of lateral deviation, but even the aircraft used in this research, some of which entered service over 20 years ago, still record a comprehensive parameter set. It can be expected that more recent types, (e.g. Boeing B787, Airbus A350, Airbus A320neo) will have a more comprehensive dataset and perhaps better measures of lateral deviation. Groups such as the European Operators Flight Data Monitoring forum (EOFDM) (European Union Aviation Safety Agency, no date) continue to try to influence aircraft manufacturers in providing more comprehensive parameter sets suitable for FDM, so hopefully research such as this can benefit from them in future.

6.3.6.2 Availability of data

This research had flight data from over 300,000 flights available for use, accumulated from 10 years of operation of a donor airline. While some operators will have datasets available which will dwarf the set used here, others may have much less data available. However, even with a limited dataset, an operator could still use these methods by making use of the informative node priors derived in this work, and then use their own data to update the priors as time passes and more data is accumulated. Regardless of size, every operator with a FDM programme will know

the rate of unstable approaches and hard landing events, and historical meteorological information is now widely available, so it would be possible for even small operators to use a Bayesian network to quantify runway veer-off risk.

6.3.6.3 Risk

The Bayesian network methods have used runway veer-off as a case study risk for two main reasons. The first is that veer-off accidents are amongst the most common today and the rate of occurrence shows no sign of abating. The second is that thanks to the prevalence of veer-off events, there is a body of literature and research projects (Future Sky Safety Consortium, no date) which can be used to identify the causal factors for inclusion in the Bayesian network. However this does not mean that the methods cannot be used to quantify the risks of other accident types. Indeed, the data has already been extracted, as part of this work, to allow the application of the methods to longitudinal runway excursions, a closely related risk to runway veer-off. The risk of other accident types could also be modelled and assessed as long as there are flight data parameters available relating to the accident causal factors. FDM is already used to monitor precursors to several accident types, such as loss of control in-flight (LOCI) and controlled flight into terrain (CFIT), and many of the causal factors of these accident types are well understood, so it should be possible to create a causal Bayesian network which uses flight data for these risks too.

6.3.7 Hybrid Bayesian networks

The Bayesian networks derived in this research have all consisted of discrete nodes, where the variable is often described as a Boolean or binary probability. Some nodes for the learnt networks had four or more discrete levels, depending on how the learning data was discretised. As previously described in the literature review, hybrid Bayesian networks can contain both discrete and continuous variables and are an excellent way of representing uncertainty in inference. A question may have arisen as to why, having made the effort to model causal factors as continuous variables, hybrid Bayesian networks were not pursued here. After all, several causal factors were modelled as gamma distributions; a distribution available for use in several Bayesian network software tools.

Indeed, it would have been fairly straightforward to include continuous variables as some of the causal factors. Most of the distributions used in the modelling process are available in Bayesian network software, and those which are not (e.g. Johnson) could perhaps have been approximated using an alternative distribution, however the inclusion of continuous variables would have made little difference to the subsequent inference. At the moment most, if not all,

Bayesian network software relies on continuous variables being discretised before any inference can be made from the network, and because the goal of this work is to produce networks for quantitative risk assessment, the interest in continuous variables lies mainly in the extremes. In addition, aircraft operations are characterised by having a normal envelope within which to operate, so for example, crosswinds are a known veer-off causal factor and are, to some extent, present on nearly every flight. It is not until the crosswind exceeds a certain magnitude that it becomes significant. Hence, even with continuous variables, the interest is in when that variable exceeds a threshold and becomes significant, effectively discretising the continuous distribution into two bins, below the threshold, and above it. These thresholds and their rationale are described in 4.6.4.

In summary, hybrid networks would be worth pursuing in future work to see if they could yield richer risk information than the simpler discrete networks used here, however at the moment it is hard to see what value the extra complexity would provide above that provided by the simpler discrete causal networks used here.

6.3.8 Summary of Bayesian network discussion

The discussion in this section started with the difficulties of validation and highlighted the problems arising when necessary flight parameters are not recorded at sufficient quality. It is hoped that new aircraft will have higher quality parameter sets which researchers, aircraft operators and ultimately flight safety, can benefit from.

Networks having a structure which reflects the straightforward causal relationship between variables (often well documented in accident reports) and parameters (i.e. causal factor prevalence probabilities) derived from flight data (Figure 22) can, through modelling, make use of estimates of extremes. However this type of network is heavily dependent on data for which it might be difficult to quality assure, such as causal factor prevalence in accidents derived from investigation reports.

Learning Bayesian networks from recorded flight data is possible and has the potential to produce networks which reflect the nuanced interactions between variables, however there is a significant dependency on having a large learning dataset and even then, the probabilities of extreme events may be under-represented.

If a network is to be learnt from flight data, Greedy Thick Thinning (Figure 49) is a good candidate algorithm. This algorithm produced a network with logical causal relationships, especially when provided with background knowledge of the temporal relationship between variables. The resulting network can be modified as necessary to remove or add arcs,

particularly if the accident literature indicates causal relationships not present in the learning dataset.

Given a large dataset from which to learn from, the overall preference is for a learnt network, due to ease of use (e.g. ability to quickly derive probabilities for different airports) and the avoidance of dependencies on other data sources.

The consequences of a hazardous event occurring must be considered in any risk assessment, and could be included in a causal network similarly to an event tree, following on from the hazardous event.

7 Conclusion

As a former safety manager of an airline, responsible for the development of the company safety management system, the author was frustrated at the lack of available methods which could turn the large amounts of flight data routinely collected into quantitative risk analyses. Trends of FDM threshold exceedance events would be presented to the Board with statements such as “...high speed on approach below 500ft has increased by 20% this quarter, while pitch rate high at take-off has reduced slightly, but is still in the top ten events”. That’s all very interesting, but how does it relate to the overall risk picture of the airline? How do details of those threshold exceedances map onto the big significant risks well known to the industry? Seemingly simple questions such as “where are we most likely to experience a runway veer-off” were (and still are) impossible to answer objectively, despite the vast amounts of data available, and so the idea for this thesis was born.

At the beginning of the Literature Review, regulations relating to flight data monitoring were stated:

“An FDM programme should allow an operator to:

(1) identify areas of operational risk and quantify current safety margins;

*(2) identify and **quantify operational risks** by highlighting occurrences of non-standard, unusual or unsafe circumstances;”* (European Aviation Safety Agency, 2014) [Emphasis added]

A contribution of this thesis is that it describes a method by which recorded flight data can be exploited to allow the quantitative analysis of an operational risk. Using the methods described, will allow aircraft operators, at both an organisational and individual level, to make better informed risk decisions, and thus ultimately improve aviation safety.

Airlines and other aircraft operators have relied on subjective risk assessments of airports to categorise them as A, B or C (European Aviation Safety Agency, 2014). Category A airports are those thought to represent less risk and have fewer apparent hazards. Operations to these airports are largely unrestricted and pilots do not require any special training to operate there, relatively inexperienced pilots are allowed to conduct landings and so on. At the other end of the scale, category C airports may require pilots to have flown the approach with a training captain or practiced the approach in a simulator. It might be specified that landings can only be conducted by captains. Category B airports are somewhere in-between. The methods described in this thesis will aid this categorisation, and future reviews of categorisations, by highlighting potentially unknown areas of risk.

The categorisation process typically involves a senior pilot, or a group of senior pilots, who will examine airport characteristics, such as runway dimensions, local terrain and the complexity of procedures. A method which highlights, based on operational evidence, that the airport is a high risk for runway veer-off, for example, will surely be a welcome compliment to the categorisation process and particularly the category review process. The output of the Bayesian network, fed with operational evidence, might highlight the influence or presence of accident causal factors which had been missed by the review pilots.

The contribution also lies in how the additional risk knowledge derived from these Bayesian networks can enhance an organisation's safety management system. By providing ongoing and routinely updated quantitative risk information, will allow the operator to maintain oversight of risks which had, hitherto, been impossible to quantify. For example, knowing that a particular airport is high on the list for veer-off risk, prompts the operator to identify reasons why and allows that risk to be communicated and addressed through pilot training, pilot briefings and standard operating procedures.

Earlier the reader was asked to imagine pilots arriving in a briefing room ahead of a day's flying. In the briefing pack, amongst the weather and NOTAMs is an additional briefing note: "CAUTION: [the airport you are operating to today] is currently classified as having a high risk of runway veer-off." This caution could be generated by software, from a Bayesian network fed with up-to-date FDM results data, aircraft defect data and local weather forecasts, to predict that the specific airport has a higher veer-off risk *today*. The contribution here is the pilots have heightened risk awareness and can take steps to mitigate those risks. They might decide to carry a little more fuel because a go-around might be more likely, given the heightened importance of flying a stable approach. They might decide that the captain, being more experienced, should carry out the landing on that particular sector. Or they might decide to refer to their operations department to advise that an aircraft swap is carried out, as the scheduled aircraft has a thrust reverser defect (which could lead to control difficulties on the landing roll-out). The list is not exhaustive; the point is that at an organisational and individual pilot level, there is knowledge and awareness that did not previously exist of a risk.

Aviation is a very safe industry, especially in airline operations, however it is not perfect. To push towards perfection, new methods must be explored and implemented. This thesis addresses two significant areas which have not kept pace with contemporary methods found in other industries, namely:

1. Data analysis methods used in flight data monitoring have not changed significantly in decades. This thesis introduces new methods which can complement existing methods

and exploit the richness of modern recorded flight data, to provide better risk information.

2. Risk assessments in airline operations are mostly qualitative, despite an abundance of data. This thesis describes a method which can provide quantitative risk analyses, allowing safety and risk managers to finally answer questions like “*Where are we most likely to experience a runway veer-off event*”.

7.1 Future work

The Bayesian methods for quantifying lateral veer-off risk should be validated further. Flight data with aircraft position recorded at a high enough resolution to allow lateral deviation of 3 or 4m to be determined, would allow the validation to use a direct measure of deviation instead of the proxy (lateral acceleration) used here. If such data became available, it would also be possible to repeat the learnt Bayesian networks using the actual lateral deviation as the target node. It is hoped that bodies such as the EOFDM Forum will continue to push aircraft and avionic manufactures to provide data at the resolutions and sampling rates useful to researchers in airlines and academia so that future safety improvements can be made.

This research showed that Bayesian networks using aircraft recorded flight data allowed the risk of a lateral deviation (veer-off) at landing to be quantified. A close relation to runway veer-off risk is runway overrun and during the data processing carried out for this research, additional features were extracted for later use in an overrun Bayesian network. Additional features included measures of braking, use of reverse thrust and runway length, which should be useful in creating the network. Due to the existence of this data, and the similar work carried out for veer-off, this is a good opportunity for using the method to quantify another risk.

The methods described are not limited to runway excursion risk; they could readily be applied to other risks for which flight data may yield information on the prevalence of causal factors, such as loss of control inflight or controlled flight into terrain.

A further recommendation for future work is to repeat the methods using datasets from other aircraft types and airlines to discover if the features modelled here (e.g. maximum g at landing, unstable approaches) can be modelled using the same distributions. Likewise, datasets from different regions of the world, where perhaps some causal factors are more common and of higher intensity, would allow distributions of features such as crosswind to be modelled and compared to the models here.

However, the ultimate goal, as previously discussed, is to establish Bayesian networks, fed with flight data and other operational data, in airlines so that the currently elusive answers to questions such as “what is our risk of a runway excursion?” can be answered quantitatively.

References

- Adikari, D. (2016) 'FDM within the TUI safety system', *Flight Data Monitoring Short Course*, Cranfield University. Cranfield, Beds.
- Aerobytes Ltd (2014) *Aerobytes FDM v3*. Available at: <https://www.aerobytes.co.uk> (Accessed: 23 October 2018).
- Agena Ltd (2016) *AgenaRisk 7.0*. Available at: www.agenarisk.com (Accessed: 11 September 2018).
- Aghdassi, N. (2018) 'FDM within the Netjets safety system', *Flight Data Monitoring Short Course*, Cranfield University. Cranfield, Beds.
- Ahlstrom, U. and Friedman-Berg, F.J. (2006) 'Using eye movement activity as a correlate of cognitive workload', *International Journal of Industrial Ergonomics*, 36(7), pp. 623–636.
- Ahmadi, A. and Soderholm, P. (2008) 'Assessment of Operational Consequences of Aircraft Failures: Using Event Tree Analysis', *2008 IEEE Aerospace Conference*. Big Sky, MT: IEEE, pp. 1–14.
- Ahmed, A., Chandra, S., Herasevich, V., Gajic, O. and Pickering, B.W. (2011) 'The effect of two different electronic health record user interfaces on intensive care provider task load, errors of cognition, and performance*', *Critical Care Medicine*, 39(7), pp. 1626–1634.
- Air Accidents Investigation Branch (2014) *Boeing 757-2T7, G-MONC, 22 May 2002 - GOV.UK, AAIB Bulletin* Available at: <https://www.gov.uk/aaib-reports/boeing-757-2t7-g-monc-22-may-2002> (Accessed: 19 December 2018).
- Airbus (2009) *Flight Crew Operating Manual A319/A320/A321 - Systems*.43 (bmi).
- Airbus S.A.S. (2017) *A Statistical Analysis of Commercial Aviation Accidents 1958-2016*.
- Ale, B.J.M., Bellamy, L.J., Cooper, J., Ababei, D., Kurowicka, D., Morales, O. and Spouge, J. (2010) 'Analysis of the crash of TK 1951 using CATS', *Reliability Engineering and System Safety*, 95(5), pp. 469–477.
- Amalberti, R. (2001) 'The paradoxes of almost totally safe transportation systems', *Safety Science*, 37(2–3), pp. 109–126.
- Anand, V. and Downs, S.M. (2008) 'Probabilistic asthma case finding: a noisy or reformulation.', *AMIA ... Annual Symposium proceedings. AMIA Symposium*. Washington DC, pp. 6–10.
- Anon. (1971) 'The Advance Towards Category Three Automatic Landings', *Aircraft Engineering and Aerospace Technology*, 43(2), pp. 28–30.
- Ashrafi, M., Davoudpour, H. and Khodakarami, V. (2017) 'A Bayesian Network to Ease Knowledge Acquisition of Causal Dependence in CREAM: Application of Recursive Noisy-OR Gates', *Quality and Reliability Engineering International*, 33(3), pp. 479–491.
- Australian Transport Safety Bureau (2008) 'Runway Excursions', *Aviation Research and Analysis Report*, 018(1), pp. 1–103.
- Aviation Risk Management Solutions (2010) *ARMS Methodology for Operational Risk Assessment*

in *Aviation Organisations*. Available at: <http://www.skybrary.aero/bookshelf/books/1141.pdf> (Accessed: 20 December 2015).

Ayra, E.S., Insua, D.R., Castellanos, M.E. and Larbi, L. (2015) 'Risk Analysis for Unintentional Slide Deployment During Airline Operations', *Risk Analysis*, 35(9), pp. 1652–1662.

Bareither, C. and Luxhøj, J.T. (2007) 'Uncertainty and sensitivity analysis in Bayesian Belief Networks: applications to aviation safety risk assessment', *International Journal of Industrial and Systems Engineering*, 2, pp. 137–158.

Barkat, M., Karan, V. and N, P. (2017) 'A Method of Reporting and Prioritizing Faults for Aircraft Downtime Reduction', *SAE Technical Paper 2017-01-2125*. SAE International.

BayesFusion LLC (2019) *GeNIe Modeler 2.3.1*.

Bolsover, A. (2015) 'Real-time risk assessment and decision support', *Process Safety Progress*, 34(2), pp. 183–190.

Brooker, P. (2011) 'Experts, Bayesian Belief Networks, rare events and aviation risk estimates', *Safety Science*, 49(8–9), pp. 1142–1155.

Buntine, W. (1996) 'A guide to the literature on learning probabilistic networks from data', *IEEE Transactions on Knowledge and Data Engineering*, 8(2), pp. 195–210.

Buntine, W.L. (1991) 'Theory Refinement on Bayesian Networks', D'Ambrosio, B. and Smets, P. (eds.) *Proceedings of the Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-91)*. University of California at Los Angeles, Los Angeles, CA, USA, Vol.UAI1991, pp. 52–60.

Bureau d'Enquêtes et d'Analyses pour la Sécurité de l'Aviation Civile (2005) *Flight Data Recorder Read-out: Technical and Regulatory Aspects*. Available at: https://www.bea.aero/uploads/tx_scalaetudessecurite/use.of.fdr_01.pdf (Accessed: 31 July 2018).

Carbone, T.A. and Tippet, D.D. (2004) 'Project Risk Management Using the Project Risk FMEA', *Engineering Management Journal*, 16(4), pp. 28–35.

Castillo, E., Gutierrez, J.M. and Hadi, A.S. (1997) 'Sensitivity analysis in discrete Bayesian networks', *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 27(4), pp. 412–423.

Chen, S.H. and Pollino, C.A. (2012) 'Good practice in Bayesian network modelling', *Environmental Modelling & Software*, 37, pp. 134–145.

Chen, W. and Huang, S. (2018) 'Evaluating Flight Crew Performance by a Bayesian Network Model', *Entropy*, 20(3), pp. 1–14.

Chen, X., Ren, H. and Bil, C. (2014) 'Fault tree analysis for composite structural damages', *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 228(9), pp. 1466–1474.

Cheng, J., Bell, D.A. and Liu, W. (1997) 'An algorithm for Bayesian belief network construction from data', *Proceedings of AI & STAT'97*, pp. 1–8.

Chidester, T.R. (2007) *Voluntary Aviation Safety Process : Preliminary Audit of Distributed*

FOQA and ASAP Archives Against Industry Statement of Requirements *DOT/FAA/AM-07/7*.

Cho, K., van Merriënboer, B., Bahdanau, D. and Bengio, Y. (2014) 'On the Properties of Neural Machine Translation: Encoder-Decoder Approaches', Wu, D., Carpuat, M., Carreras, X. and Vecchi, E. M. (eds.) *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Vol. abs/1409.1, pp. 1–9.

Civil Aviation Authority (2013) *CAP739 - Flight Data Monitoring*. Available at: <http://publicapps.caa.co.uk/docs/33/CAP739.pdf> (Accessed: 18 February 2018).

Civil Aviation Authority (2011) *CAA 'Significant Seven' Task Force Reports*. Available at: https://publicapps.caa.co.uk/docs/33/2011_03.pdf (Accessed: 13 March 2017).

Civil Aviation Authority (2009) *Air Navigation Order 2009*. 2009 No. 3015.

Civil Aviation Authority (no date) *UK CAA Bowtie guidance*. Available at: <http://www.caa.co.uk/Safety-Initiatives-and-Resources/Working-with-industry/Bowtie/> (Accessed: 16 July 2018).

Coleman, R. (2002) 'Modelling Extremes', *International Statistical Workshop*. Seoul, pp. 1–14.

Cooper, G.F. and Herskovits, E. (1992) 'A Bayesian Method for the Induction of Probabilistic Networks from Data', *Machine Learning*, 9(4), pp. 309–347.

Cox, D.R. (1972) 'Regression Models and Life-Tables', *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2), pp. 187–220.

Cox, L.A. (2008) 'What's Wrong with Risk Matrices?', *Risk Analysis*, 28(2), pp. 497–512.

Cullen, A. and Frey, C. (1999) *Probabilistic Techniques in Exposure Assessment*. New York: Plenum Press.

Dalkey, N. and Helmer, O. (1963) 'An Experimental Application of the DELPHI Method to the Use of Experts', *Management Science*, 9(3), pp. 458–467.

Darwiche, A. (2009) *Modeling and Reasoning with Bayesian Networks*. Cambridge: Cambridge University Press.

Das, K.P. and Dey, A.K. (2016) 'Quantifying the risk of extreme aviation accidents', *Physica A: Statistical Mechanics and its Applications*, 463, pp. 345–355.

Das, S., Li, L., Srivastava, A. and Hansman, R.J. (2012) 'Comparison of Algorithms for Anomaly Detection in Flight Recorder Data of Airline Operations', *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics.

Dekker, S.W.A. (2004) 'Why We Need New Accident Models', *Human Factors and Aerospace Safety*, 4(1), pp. 1–18.

Delignette-Muller, M.L. and Dutang, C. (2015) '{fitdistrplus}: An {R} Package for Fitting Distributions', *Journal of Statistical Software*, 64(4), pp. 1–34.

Dempster, A.P.P., Laird, N.M. and Rubin, D.B. (1977) 'Maximum Likelihood from Incomplete Data via the EM Algorithm', *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1),

pp. 1–38.

Ding, S. and Ru, Y. (2008) 'Flight risk assessment to airlines using Bayesian belief networks and fuzzy comprehensive evaluation', *2008 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE, pp. 1280–1284.

European Aviation Safety Agency (2008) *COMMISSION REGULATION (EC) No 859/2008. Annex 3*.

European Aviation Safety Agency (2014) *Acceptable Means of Compliance (AMC) and Guidance Material (GM) to part -ORO*.

European Operators Flight Data Monitoring Working Group B (2017) *Guidance for the Implementation of FDM Precursors*. Available at: https://www.easa.europa.eu/system/files/dfu/214908_Study_WGB_Precursors_2Publish_20170717.pdf (Accessed: 21 January 2018).

European Union Aviation Safety Agency (no date) *European Operators Flight Data Monitoring (EOFDM) forum / EASA*. Available at: <https://www.easa.europa.eu/easa-and-you/safety-management/safety-promotion/european-operators-flight-data-monitoring-eofdm-forum> (Accessed: 17 March 2019).

Fawcett, T. (2006) 'An introduction to ROC analysis', *Pattern Recognition Letters*, 27(8), pp. 861–874.

Federal Aviation Administration (2004) *AC 120-82 - Flight Operational Quality Assurance*. Available at: https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_120-82.pdf (Accessed: 12 December 2017).

Fenton, N. and Neil, M. (2007) *Managing Risk in the Modern World: Applications of Bayesian Networks*. London Mathematical Society.

Fenton, N. and Neil, M. (2013) *Risk Assessment and Decision Analysis with Bayesian Networks*. 1st edn. CRC Press.

Flight Safety Foundation (2009a) 'Reducing the Risk of Runway Excursions', *Report of the Runway Safety Initiative*, , p. 235.

Flight Safety Foundation (2009b) 'Runway Excursions', *FSF ALAR Briefing Note 8.1*, , pp. 1–4.

Flight Safety Foundation (2009c) *Stabilized Approach*. Available at: <https://www.skybrary.aero/bookshelf/books/864.pdf> (Accessed: 21 June 2018).

Friedman, N., Geiger, D. and Goldszmidt, M. (1997) 'Bayesian Network Classifiers', *Machine Learning*, 29(2–3), pp. 131–163.

Friedman, N., Nachman, I. and Pe'er, D. (1999) 'Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm', Laskey, K. and Prade, H. (eds.) *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco (CA): Morgan Kaufmann Publishers Inc, pp. 206–215.

Fruhen, L.S., Flin, R. and McLeod, R. (2014) 'Chronic unease for safety in managers: a conceptualisation', *Journal of Risk Research*, 17(8), pp. 969–979.

Future Sky Safety Consortium (no date) *Project #3: Specific solutions for runway excursion accidents*, *Future Sky Safety Project #3 web page* Available at: <https://www.futuresky->

safety.eu/project-3/ (Accessed: 25 August 2018).

van der Geest, P. (NLR) (2018) *Dissemination material from second Future Sky Safety Internal Workshop*, *Future Sky Safety* Available at: https://www.futuresky-safety.eu/wp-content/uploads/2018/03/FSS_P2_DBL_D2.13_v2.0.pdf (Accessed: 28 February 2018).

Goble, R. and Bier, V.M. (2013) 'Risk assessment can be a game-changing information technology--but too often it isn't.', *Risk Analysis*, 33(11), pp. 1942–51.

Gorinevsky, D., Matthews, B. and Martin, R. (2012) 'Aircraft anomaly detection using performance models trained on fleet data', Chawla, N. V and Srivastava, A. N. (eds.) *2012 Conference on Intelligent Data Understanding*. IEEE, pp. 17–23.

Groth, K.M. and Swiler, L.P. (2013) 'Use of Limited Data to Construct Bayesian Networks for Probabilistic Risk Assessment', *Sandia Report*, (SAND2013-2019) Albuquerque, New Mexico, pp. 1–32.

Hadjimichael, M. (2009) 'A fuzzy expert system for aviation risk assessment', *Expert Systems with Applications*, 36(3), pp. 6512–6519.

Hadjimichael, M. and McCarthy, J. (2004) 'Using a Fuzzy Expert System to Represent Risk: FORAS', *SAE Technical Paper*, (2004-01-3143)

Harrell Jr, F.E., with contributions from Charles Dupont and many others. (2017) *Hmisc: Harrell Miscellaneous*. Available at: <https://cran.r-project.org/package=Hmisc> (Accessed: 30 June 2018).

Hart, S.G. (2006) 'Nasa-Task Load Index (NASA-TLX); 20 Years Later', *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9), pp. 904–908.

Hart, S.G. and Staveland, L.E. (1988) 'Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research', in *Advances in Psychology*, pp. 139–183.

Haverdings, H. and Chan, P.W. (2009) 'Quick Access Recorder (QAR) Data Analysis Software for Windshear and Turbulence Studies', *1st AIAA Atmospheric and Space Environments Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics, pp. 22–25.

Heckerman, D., Geiger, D. and Chickering, D.M. (1995) 'Learning Bayesian Networks: The Combination of Knowledge and Statistical Data', *Machine Learning*, 20, pp. 197–243.

Henchie, S. (2013) 'FDM within the easyJet safety system', *Flight Data Monitoring Short Course*, Cranfield University. Cranfield, Beds.

Hills, B. (1975) 'A Simple General Approach to Inference About the Tail of a Distribution', *The Annals of Statistics*, 3(5), pp. 1163–1174.

Hochreiter, S. and Schmidhuber, J. (1997) 'Long Short-Term Memory', *Neural Computation*, 9(8), pp. 1735–1780.

Hu, K.-C. and Hsiao, M.-W. (2016) 'Quality risk assessment model for airline services concerning Taiwanese airlines', *Journal of Air Transport Management*, 53, pp. 177–185.

Hubbard, D.W. (2012) *The Failure of Risk Management*. 1st edn. Hubbard, D. W. (ed.) Hoboken, NJ, USA: John Wiley & Sons, Inc.

- Hwang, S.-L., Yau, Y.-J., Lin, Y.-T., Chen, J.-H., Huang, T.-H., Yenn, T.-C. and Hsu, C.-C. (2008) 'Predicting work performance in nuclear power plants', *Safety Science*, 46(7), pp. 1115–1124.
- Ilčev, S.D. (2018) *Global Satellite Meteorological Observation (GSMO) Theory*. Volume 1. Cham: Springer International Publishing.
- Institute for Safe Medication Practices (2001) *Failure mode and effects analysis can help guide error-prevention efforts.*, *ISMP Safety Alert* Available at: <https://www.ismp-canada.org/fmea.htm> (Accessed: 29 October 2018).
- International Air Transport Association (IATA) (2016) *Unstable Approaches Risk Mitigation Policies, Procedures and Best Practices 2nd Edition*. Available at: <https://www.iata.org/whatwedo/safety/runway-safety/Documents/unstable-approaches-2016-2nd-edition.pdf> (Accessed: 21 June 2018).
- International Air Transport Association (IATA) (2017) *Safety Report 2016*. Available at: <https://skybrary.aero/bookshelf/books/3875.pdf> (Accessed: 3 April 2018).
- International Civil Aviation Organisation (2010) *ICAO Annex 6 - Operation of Aircraft*. 9th edn. International Civil Aviation Organisation.
- International Civil Aviation Organisation (2013) *Safety Management Manual (SMM)*. 3rd edn. International Civil Aviation Organisation.
- International Civil Aviation Organisation (2006) *Safety Management Manual (SMM)*. 1st edn. International Civil Aviation Organisation.
- International Civil Aviation Organisation (2005) *Annex 2 to the Convention on International Civil Aviation - Rules of the Air*. 10th edn. International Civil Aviation Organisation.
- International Civil Aviation Organisation (2018) *ICAO Annex 14 Aerodromes*. 8th edn. International Civil Aviation Organisation.
- International Organization for Standardization (2018) *ISO 31000 Risk management - Principles and guidelines International Organization for Standardization*.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013) *An Introduction to Statistical Learning*. New York, NY: Springer New York, Springer Texts in Statistics.
- Jensen, F. V., Aldenryd, S.H. and Jensen, K.B. (1995) 'Sensitivity analysis in Bayesian networks', in *Symbolic and Quantitative Approaches*, pp. 243–250.
- Johnson, N.L. (1949) 'Systems of frequency curves generated by methods of translation', *Biometrika*, 36(1–2), pp. 149–176.
- Kalantarnia, M., Khan, F. and Hawboldt, K. (2009) 'Dynamic risk assessment using failure assessment and Bayesian theory', *Journal of Loss Prevention in the Process Industries*, 22(5), pp. 600–606.
- Kalantarnia, M., Khan, F. and Hawboldt, K. (2010) 'Modelling of BP Texas City refinery accident using dynamic risk assessment approach', *Process Safety and Environmental Protection*, 88(3), pp. 191–199.
- Kane, V.E. (1986) 'Process Capability Indices', *Journal of Quality Technology*, 18, pp. 41–52.

- Kanes, R., Ramirez Marengo, M.C., Abdel-Moati, H., Cranefield, J. and Véhot, L. (2017) 'Developing a framework for dynamic risk assessment using Bayesian networks and reliability data', *Journal of Loss Prevention in the Process Industries*, 50, pp. 142–153.
- Khakzad, N., Khan, F. and Amyotte, P. (2011) 'Safety analysis in process facilities: Comparison of fault tree and Bayesian network approaches', *Reliability Engineering & System Safety*, 96(8), pp. 925–932.
- Kjærulff, U. and van der Gaag, L. (2000) 'Making Sensitivity Analysis Computationally Efficient', Boutilier, C. and Goldszmidt, M. (eds.) *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc, pp. 317–325.
- Knecht, W.R. (1997) 'Developing a Probabilistic Metric of Midair Collision Risk', *Transportation Research Record: Journal of the Transportation Research Board*, 1567(1), pp. 26–32.
- Koiter, J.R. (2006) *Visualizing Inference in Bayesian Networks*, MSc Thesis Available at: <http://www.kbs.twi.tudelft.nl/docs/MSc/2006/JRKoiter/thesis.pdf> (Accessed: 6 February 2019).
- Koller, D. and Friedman, N. (2009) *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series)*. MIT Press.
- Kruschke, J.K. (2013) 'Bayesian estimation supersedes the t test.', *Journal of Experimental Psychology: General*, 142(2), pp. 573–603.
- Kruschke, J.K. and Meredith, M. (2018) *BEST: Bayesian Estimation Supersedes the t-Test*. Available at: <https://cran.r-project.org/package=BEST> (Accessed: 21 November 2018).
- Kuhn, M. and Johnson, K. (2013) *Applied Predictive Modeling*. New York, NY: Springer New York.
- Langseth, H., Nielsen, T.D., Rumí, R. and Salmerón, A. (2009) 'Inference in hybrid Bayesian networks', *Reliability Engineering & System Safety*, 94(10), pp. 1499–1509.
- Lauritzen, S.L. (1995) 'The EM algorithm for graphical association models with missing data', *Computational Statistics & Data Analysis*, 19(2), pp. 191–201.
- Lemmer, J.F. and Gossink, D.E. (2004) 'Recursive Noisy OR—A Rule for Estimating Complex Probabilistic Interactions', *2252 IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 34(6), pp. 2252–2261.
- Leveson, N. (2004) 'A new accident model for engineering safer systems', *Safety Science*, 42(4), pp. 237–270.
- Li, L., Das, S., John Hansman, R., Palacios, R. and Srivastava, A.N. (2015) 'Analysis of Flight Data Using Clustering Techniques for Detecting Abnormal Operations', *Journal of Aerospace Information Systems*, 12(9), pp. 587–598.
- Li, L., Gariel, M., Hansman, R.J. and Palacios, R. (2011) 'Anomaly detection in onboard-recorded flight data using cluster analysis', Burkle, M. and Herndon, A. (eds.) *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*. Seattle, Washington: IEEE, pp. 1–31.
- Li, L., Hansman, R.J., Palacios, R. and Welsch, R. (2016) 'Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring', *Transportation Research Part C: Emerging Technologies*, 64(Supplement C), pp. 45–57.

- Looije, R. and Mioch, T. (2011) 'The effect of task load on the occurrence of cognitive lockup in a high-fidelity flight simulator', Dittmar, A. and Forbrig, P. (eds.) *Proceedings of the 29th Annual European Conference on Cognitive Ergonomics - ECCE '11*. New York, New York, USA: ACM Press, p. 19.
- Lu, C., Wetmore, M. and Smith, J. (2005) 'Advocating System Safety Concept in Preventing Airline Accidents', *46th Annual Research & Policy Forum.*, pp. 1–20.
- Luxhoj, J.T. and Coit, D.W. (2006) 'Modeling low probability/high consequence events: an aviation safety risk model', *Annual Reliability and Maintainability Symposium, 2006*. IEEE, pp. 215–221.
- Marcot, B.G., Steventon, J.D., Sutherland, G.D. and McCann, R.K. (2006) 'Guidelines for developing and updating Bayesian belief networks applied to ecological modeling and conservation', *Canadian Journal of Forest Research*, 36(12), pp. 3063–3074.
- McFadden, K.L. (2003) 'Risk models for analyzing pilot-error at US airlines: a comparative safety study', *Computers & Industrial Engineering*, 44(4), pp. 581–593.
- McHale, J. (2005) AFDX technology to improve communications on Boeing 787, *Magazine of Transformation in Electronic and Optical Technology, Military and Aerospace Electronics*, 16(4).
- Meel, A. and Seider, W.D. (2006) 'Plant-specific dynamic failure assessment using Bayesian theory', *Chemical Engineering Science*, 61(21), pp. 7036–7056.
- Meggison, D. (no date) *Open data @ OurAirports*. Available at: <http://ourairports.com/data/> (Accessed: 19 August 2017).
- Mendes, H. (2012) *Study of Mathematical Algorithms to Identify Abnormal Patterns in Aircraft Flight Data*. Available at: <https://fenix.tecnico.ulisboa.pt/downloadFile/395144227066/ResumoAlargadoHM12Jun2012.pdf> (Accessed: 13 December 2017).
- Mikulak, R., McDermott, R. and Beauregard, M. (2008) *The Basics of FMEA*. 2nd edn. Boca Raton, FL: CRC Press.
- Moretti, L., Di Mascio, P., Nichele, S. and Cokorilo, O. (2018) 'Runway veer-off accidents: Quantitative risk assessment and risk reduction measures', *Safety Science*, 104, pp. 157–163.
- Mori, R. (2014) 'Refined collision risk model for oceanic flight under longitudinal distance-based separation in ADS-C environment', *Journal of Navigation*, 67(5), pp. 845–868.
- Moriarty, D. and Jarvis, S. (2014) 'A systems perspective on the unstable approach in commercial aviation', *Reliability Engineering & System Safety*, 131, pp. 197–202.
- Mugtussidis, I.B. (2000) *Flight Data Processing Techniques to Identify Unusual Events.*, *Doctoral Dissertation* Available at: <http://hdl.handle.net/10919/28095> (Accessed: 29 April 2015).
- Nadkarni, S. and Shenoy, P.P. (2004) 'A causal mapping approach to constructing Bayesian networks', *Decision Support Systems*, 38(2), pp. 259–281.
- Nahapetian, A. (2014) A Comparison of ARINC 717 and ARINC 767 Flight Data Recording, *Aviation Maintenance*,
- Nanduri, A. and Sherry, L. (2016a) 'Anomaly detection in aircraft data using Recurrent Neural

Networks (RNN)', *2016 Integrated Communications Navigation and Surveillance (ICNS)*. Herndon, VA: IEEE, pp. 5C2-1-5C2-8.

Nanduri, A. and Sherry, L. (2016b) 'Generating Flight Operations Quality Assurance (foqa) data from the X-Plane Simulation', *2016 Integrated Communications Navigation and Surveillance (ICNS)*. Herndon, VA: IEEE, pp. 5C1-1-5C1-9.

Narasimhan, R. (2017) *weatherData: Get Weather Data from the Web*. Available at: <https://cran.r-project.org/package=weatherData> (Accessed: 1 August 2018).

National Air and Space Museum (2015) *The Wright Brothers, Inventing a Flying Machine, Flight Controls*. Available at: <https://airandspace.si.edu/exhibitions/wright-brothers/online/fly/1903/flightcontrol.cfm> (Accessed: 8 February 2018).

Nivolianitou, Z.S., Leopoulos, V.N. and Konstantinidou, M. (2004) 'Comparison of techniques for accident scenario analysis in hazardous systems', *Journal of Loss Prevention in the Process Industries*, 17(6), pp. 467–475.

NOMIS SOFT Datenverarbeitung Gesellschaft mit beschränkter Haftung (no date) *The Aviation Herald*, *The Aviation Herald* Available at: <http://avherald.com> (Accessed: 21 January 2018).

Norsys Software Corp. (no date) *Norsys Software Corp. - Bayes Net Software*. Available at: <https://www.norsys.com/index.html> (Accessed: 3 February 2019).

Oakland, J.S. (2012) *Statistical Process Control*. 5th edn. Oxford: Butterworth–Heinemann.

Palady, P. (1998) *Failure Modes and Effects Analysis: Author's Edition*. Michigan: Pal Publications.

Paltrinieri, N., Khan, F., Amyotte, P. and Cozzani, V. (2014) 'Dynamic approach to risk management: Application to the Hoeganaes metal dust accidents', *Process Safety and Environmental Protection*, 92(6), pp. 669–679.

Pasman, H. and Rogers, W. (2015) 'The bumpy road to better risk control: A Tour d'Horizon of new concepts and ideas', *Journal of Loss Prevention in the Process Industries*, 35, pp. 366–376.

Pearl, J. (1988a) 'Belief Updating by Network Propagation', in Pearl, J. (ed.) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Burlington, MA: Morgan Kaufmann Publishers Inc, pp. 143–237.

Pearl, J. (1988b) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Burlington, MA: Morgan Kaufmann Publishers Inc.

Porterfield, D.H. (1997) 'Evaluating Controller Communication Time as a Measure of Workload', *The International Journal of Aviation Psychology*, 7(2), pp. 171–182.

Post, J.A. (2015) *Identification and analysis of veer- off risk factors in accidents / incidents*, *Future Sky Safety* Available at: https://www.futuresky-safety.eu/wp-content/uploads/2016/01/FSS_P3_NLR_D3.4_v2.0.pdf (Accessed: 14 November 2017).

R Core Team (2017) *R: A language and environment for statistical computing*, *R Foundation for Statistical Computing* Available at: <https://www.r-project.org/> (Accessed: 28 August 2018).

Reason, J. (1997) *Managing the Risks of Organizational Accidents*. 1st edn. London: Ashgate.

Roelen, A.L.C., Lin, P.H. and Hale, A.R. (2011) 'Accident models and organisational factors in air

transport: The need for multi-method models', *Safety Science*, 49(1), pp. 5–10.

Safety Investigation Authority Finland (2018) *L2017-05 Loss of Directional Control during Landing Roll at Turku Airport on October 25, 2017., Investigation report* Available at: <https://www.turvallisuustutkinta.fi/en/index/tutkintaselostukset/ilmailuonnettomuuksientutkinta/tutkintaselostuksetvuosittain/2017/l2017-05liikennelentokoneenhallinnanmenetysslaskukiidossaturunlentoasemalla25.10.2017.html> (Accessed: 4 February 2019).

Scrucca, L. (2004) qcc: an R package for quality control charting and statistical process control, *R News - The Newsletter of the R Project*, 4/1, pp. 11–17.

Scutari, M. (2017) *Understanding Bayesian Networks with Examples in R*. Available at: <http://www.bnlearn.com/about/teaching/slides-bnshort.pdf> (Accessed: 11 March 2018).

Scutari, M. (2010) *bnlearn: Bayesian network structure learning*. Available at: <https://cran.r-project.org/web/packages/bnlearn/bnlearn.pdf> (Accessed: 5 August 2018).

Scutari, M. and Denis, J. (2014) *Bayesian networks - With examples in R*. 1st edn. Boca Raton, FL: CRC Press.

Senders, J.W. (2004) 'FMEA and RCA: the mantras; of modern risk management', *Quality and Safety in Health Care*, 13(4), pp. 249–250.

Sheridan, T.B. (2008) 'Risk, Human Error, and System Resilience: Fundamental Ideas', *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(3), pp. 418–426.

Shyur, H.-J., Keng, H., I, I.-K. and Huang, C.-L. (2012) 'Using extended hazard regression model to assess the probability of aviation event', *Applied Mathematics and Computation*, 218(21), pp. 10647–10655.

Siu, N.O. and Kelly, D.L. (1998) 'Bayesian parameter estimation in probabilistic risk assessment', *Reliability Engineering & System Safety*, 62(1–2), pp. 89–116.

Smart, E. (2011) *Detecting Abnormalities in Aircraft Flight Data and Ranking their Impact on the Flight*. Available at: <https://researchportal.port.ac.uk/portal/files/6061688/thesis.pdf> (Accessed: 13 December 2017).

Spiegelhalter, D.J., Dawid, A.P., Lauritzen, S.L. and Cowell, R.G. (1993) 'Bayesian Analysis in Expert Systems', *Statistical Science*, 8(3), pp. 219–247.

Spirtes, P., Glymour, C. and Scheines, R. (1993) *Causation, Prediction, and Search*. 1st edn. Springer New York.

Squillante, R., Santos Fo, D.J., Maruyama, N., Junqueira, F., Moscato, L.A., Nakamoto, F.Y., Miyagi, P.E. and Okamoto, J. (2018) 'Modeling accident scenarios from databases with missing data: A probabilistic approach for safety-related systems design', *Safety Science*, 104, pp. 119–134.

Stamatelatos, M., Vesely, W., Dugan, J., Fragola, J., Minarich III, J. and Railsback, J. (2002) *Fault Tree Handbook with Aerospace Applications*. 1.1. Washington DC: NASA.

Di Stasi, L.L., Diaz-Piedra, C., Rieiro, H., Sánchez Carrión, J.M., Martín Berrido, M., Olivares, G. and Catena, A. (2016) 'Gaze entropy reflects surgical task load', *Surgical Endoscopy*, 30(11), pp. 5034–5043.

- Straub, D. (2005) 'Natural hazards risk assessment using Bayesian networks', Augusti, G., Schuëller, G. I. and Ciampoli, M. (eds.) *9th International Conference on Structural Safety and Reliability ICOSSAR 05 Rome Italy*. Rotterdam: Millpress.
- Streiner, D.L. and Cairney, J. (2007) 'What's under the ROC? An Introduction to Receiver Operating Characteristics Curves', *The Canadian Journal of Psychiatry*, 52(2), pp. 121–128.
- Sudolsky, M.D. (2009) 'ARINC 573/717, 767 and 647A: The Logical Choice for Maintenance Recording And IVHM Interface Control or Frame Updates', *Annual Conference of the Prognostics and Health Management Society*. San Diego: PHM Society.
- Taleb, N.N. (2007) *The A Priori Problem of Observed Probabilities*. Available at: <http://www.foolledbyrandomness.com/central.pdf> (Accessed: 22 January 2019).
- Thornton, A.C. (1999) 'Variation Risk Management Using Modeling and Simulation', *Journal of Mechanical Design*, 121(2), p. 297.
- Tonda, A.P., Lutton, E., Reuillon, R., Squillero, G. and Wuillemin, P.-H. (2012) 'Bayesian Network Structure Learning from Limited Datasets through Graph Evolution', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 254–265.
- Trucco, P., De Ambroggi, M. and Chiara Leva, M. (2015) 'Topological risk mapping of runway overruns: A probabilistic approach', *Reliability Engineering & System Safety*, 142, pp. 433–443.
- Trucco, P., Cagno, E., Ruggeri, F. and Grande, O. (2008) 'A Bayesian Belief Network modelling of organisational factors in risk analysis: A case study in maritime transportation', *Reliability Engineering & System Safety*, 93(6), pp. 845–856.
- Tubbs-Cooley, H.L., Mara, C.A., Carle, A.C. and Gurses, A.P. (2018) 'The NASA Task Load Index as a measure of overall workload among neonatal, paediatric and adult intensive care nurses', *Intensive and Critical Care Nursing*, 46, pp. 64–69.
- Uusitalo, L. (2007) 'Advantages and challenges of Bayesian networks in environmental modelling', *Ecological Modelling*, 203(3–4), pp. 312–318.
- Villa, V., Paltrinieri, N., Khan, F. and Cozzani, V. (2016) 'Towards dynamic risk analysis: A review of the risk assessment approach and its limitations in the chemical process industry', *Safety Science*, 89, pp. 77–93.
- Wagner, D.C.S. and Barker, K. (2014) 'Statistical methods for modeling the risk of runway excursions', *Journal of Risk Research*, 17(7), pp. 885–901.
- Wang, L., Ren, Y. and Wu, C. (2018) 'Effects of flare operation on landing safety: A study based on ANOVA of real flight data', *Safety Science*, 102, pp. 14–25.
- Weber, P., Medina-Oliva, G., Simon, C. and Iung, B. (2012) 'Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas', *Engineering Applications of Artificial Intelligence*, 25(4), pp. 671–682.
- Wheeler, B. (2016) *SuppDists: Supplementary Distributions*. Available at: <https://cran.r-project.org/package=SuppDists> (Accessed: 21 June 2018).
- Wheeler, D.J. (1993) *Understanding Variation: The Key to Managing Chaos*. 1st edn. Knoxville, TN: SPC Press.

Wiegmann, D. and Shappell, S. (2003) *A Human Error Approach to Aviation Accident Analysis: The Human Factors Analysis and Classification System*. 1st edn. Ashgate.

Yeo, C., Bhandari, J., Abbassi, R., Garaniya, V., Chai, S. and Shomali, B. (2016) 'Dynamic risk analysis of offloading process in floating liquefied natural gas (FLNG) platform using Bayesian Network', *Journal of Loss Prevention in the Process Industries*, 41, pp. 259–269.

Yitzhaki, S. (2003) 'Gini's Mean difference: A superior measure of variability for non-normal distributions', *Metron*, 61(2), pp. 285–316.

Yurko, Y.Y., Scerbo, M.W., Prabhu, A.S., Acker, C.E. and Stefanidis, D. (2010) 'Higher Mental Workload is Associated With Poorer Laparoscopic Performance as Measured by the NASA-TLX Tool', *Simulation in Healthcare: The Journal of the Society for Simulation in Healthcare*, 5(5), pp. 267–271.

Zarei, E., Azadeh, A., Aliabadi, M.M. and Mohammadfam, I. (2017) 'Dynamic safety risk modeling of process systems using bayesian network', *Process Safety Progress*, 36(4), pp. 399–407.

Zwirglmaier, K. and Straub, D. (2015) 'Discretization of structural reliability problems : an application to runway overrun', Haukaas, T. (ed.) *International Conference on Applications of Statistics and Probability in Civil Engineering (ICASP) (12th : 2015)*. Vancouver: University of British Columbia Library.

Zwirglmaier, K. and Straub, D. (2016) 'A discretization procedure for rare events in Bayesian networks', *Reliability Engineering & System Safety*, 153(Supplement C), pp. 96–109.

Appendix 1 – R code for converting CSV flight data files into R Objects

```
1. # Converttempv4.R
2. # v1 Convert batches of CSVs into R format
3. # v2 includes correcting date from Date_Correct_RDS_v01.R
4. # v3 fixes end of month problem. Also removed acreg extraction from path
5. # for when converting one aircraft only i.e. acreg is known.
6. # v4 Added reg correct from Reg_Correct_RDS_v03
7.
8.
9. library(zoo) ### for na.locf
10. library(data.table)
11. library(Hmisc) ### for %nin%
12.
13. setwd("E:/Rworking/convertR")
14.
15. CSVlocation <- "E:/CSV/2002/A321"
16. saveDirName <- "E:/Rflights/RDSxDupe/2002/A321_2002"
17. realyear <- "2002" ##for filename correction
18.
19. ## Load already processed flight IDs for duplicate check
20. MflightID <- list.files("E:/Rflights/RDSxDupe", recursive = TRUE, full.names = TRUE)
21. MflightID <- strsplit(MflightID, ".rd")
22. y <- sapply(MflightID[1], length)
23. MflightID <- matrix(unlist(MflightID), ncol = y, byrow = TRUE)
24. MflightID <- MflightID[,1]
25.
26. MflightID <- strsplit(MflightID, "/")
27. z <- sapply(MflightID[1], length)
28. MflightID <- matrix(unlist(MflightID), ncol=z, byrow=TRUE)
29. MflightID <- MflightID[,z]
30.
31.
32. ### Get file path list in folder
33. newCSVfull <- list.files(CSVlocation, recursive = TRUE, full.names = TRUE)
34.
35. CSVid <- strsplit(newCSVfull, "/")
36. x <- sapply(CSVid[1], length)
37. CSVid <- matrix(unlist(CSVid), ncol=x, byrow=TRUE)
38. CSVid <- CSVid[,x]
39. newCSVfull <- newCSVfull[which(CSVid %nin% MflightID)]
40. newCSV <- CSVid[CSVid %nin% MflightID]
41.
42. flight <- character()
43. errortype <- character()
44.
45. errors <- data.frame(flight = character(0), errortype = character(0), stringsAsFactors
= FALSE)
46.
47. colnames(errors) <- c("Flight", "Error")
48.
49. #pb <- winProgressBar("Converting files", "", min = 0, max = 1, initial = 0, width =
300)
50.
51. cat(paste("Converting", length(newCSV), "files."))
52.
53. for(i in 1:length(newCSV)) {
54.   # Progress
55.   cat(paste(".", i, "."))
56.
57.   ## flight ID for output
58.   flight <- newCSV[i]
59.
60.   ## Check if file has been seen before
61.   dupe <- is.element(flight, MflightID)
62.   if (dupe == TRUE) {
63.     errortype <- "Duplicate"
64.     errors[nrow(errors)+1,] <- c(flight, errortype)
65.     next}
66.
67.   MflightID <- c(MflightID, flight) # add this flight to Master flight list
68.
```

```

69. # Read in data and make good column names
70.
71. tempflt <- try(fread(newCSVfull[i], stringsAsFactors = FALSE, sep = ",", header =
TRUE))
72.
73. if (length(tempflt) < 2) {
74.   errortype <- "fread ERROR"
75.   errors[nrow(errors)+1,] <- c(flight, errortype)
76.
77.   next} ## try above returns object of length [1] if fread fails
78.
79. colnames(tempflt) <- make.names(colnames(tempflt))
80.
81. # Quality check
82.
83. endoffile <- length(tempflt$Computed.Airspeed)
84. frame <- as.numeric(gsub("\\*", "", tempflt$Frame.Counter))
85. frame <- na.locf(frame, na.rm = FALSE)
86.
87. frame4095 <- which(frame==4095)
88. frameadj <- frame
89.
90. if (length(frame4095) == 32) {
91.   framezero1 <- frame4095[32]+1
92.   frameadj[framezero1:endoffile] <- frameadj[framezero1:endoffile]+4096
93. }
94.
95. if (length(frame4095) == 64) {
96.   framezero1 <- frame4095[32]+1
97.   framezero2 <- frame4095[64]+1
98.   frameadj[framezero1:framezero2-1] <- frameadj[framezero1:framezero2-1]+4096
99.   frameadj[framezero2:endoffile] <- frameadj[framezero2:endoffile]+8192 # check
this. +1 to framezero2?
100. }
101.
102. framep32 <- c(NA + 1:32, frameadj[1:(length(frameadj) - 32)]) # instead of for-
loop, creates offset vector
103. framechk <- ((frameadj - 1) == (framep32)) # checks continuity of frame number
104. framechkrun <- max(rle(framechk) [[TRUE]]) # looks for maximum run of continuity
105. #rleref <- which(rle(framechk) [[TRUE]] == framechkrun) # looks for start of max
run
106. rlecols <- rle(framechk)[1]
107. rlecols <- unlist(rlecols)
108. names(rlecols) = NULL
109. rleref <- which(rlecols == max(rlecols))
110. endframe <- sum(rlecols[1:rleref])
111. startframe <- endframe - framechkrun
112.
113. #endframe <- startframe + framechkrun
114.
115. # if run is less than 10 minutes, skip this flight
116. if (framechkrun < 4800) {
117.
118.   errortype <- "TOO SHORT"
119.   errors[nrow(errors)+1,] <- c(flight, errortype)
120.
121.   next
122. }
123.
124. # remove data that does not have continuous frame count
125. tempflt <- tempflt[startframe:endframe]
126.
127. # fix date in file name
128. realday <- tempflt$GMT.date..Day.
129. realday <- as.numeric(realday[!(realday=="")])
130. realday <- realday[!is.na(realday)]
131. commonday <- as.numeric(names(which.max(table(realday))))
132. firstday <- as.numeric(realday[1])
133.
134. ifelse((commonday==(firstday+1)) || (firstday > 27 && commonday==1),
135.   realday <- firstday, realday <- commonday)
136.
137. realday <- realday[1]
138.
139. realmonth <- tempflt$GMT.date..Month.
140. realmonth <- as.numeric(realmonth[!(realmonth=="")])
141. realmonth <- realmonth[!is.na(realmonth)]

```

```

142. commonmonth <- as.numeric(names(which.max(table(realmonth))))
143. firstmonth <- as.numeric(realmonth[1])
144.
145. ifelse(commonmonth==(firstmonth +1), realmonth <- firstmonth, realmonth <-
commonmonth)
146.
147. if ((firstmonth==12 & commonmonth==1)) {
148.   (realmonth <- firstmonth)}
149.
150. realmonth <- realmonth[1]
151.
152. # Pad day and months with zeroes if necessary
153. if (realday < 10) {
154.   realday <- paste("0", realday, sep = "")
155. }
156.
157. if (realmonth < 10) {
158.   realmonth <- paste("0", realmonth, sep = "")
159. }
160.
161. # paste date elements together
162. realdate <- paste("dt=", realyear, realmonth, realday, sep = "")
163.
164. flight <- gsub("dt=.....", realdate, flight)
165.
166. # Correct registration (based on Reg_Correct_RDS_v03)
167. fileReg <- strsplit(flight, "ac=")
168. y <- sapply(fileReg[1], length)
169. fileReg <- matrix(unlist(fileReg), ncol = y, byrow = TRUE)
170. fileReg <- fileReg[,2]
171.
172. fileReg <- strsplit(fileReg, " dep=")
173. z <- sapply(fileReg[1], length)
174. fileReg <- matrix(unlist(fileReg), ncol=z, byrow=TRUE)
175. fileReg <- fileReg[,1]
176.
177. realReg <- tempflt$A.C.Tail.Number
178. realReg <- (realReg[!(realReg=="")])
179. realReg <- names(which.max(table(realReg)))
180. realReg <- gsub("[/*]", "", realReg)
181. realReg <- gsub("[/.]", "", realReg)
182.
183. if (fileReg != realReg) {
184.
185.   realReg <- paste("ac=", realReg, sep = "")
186.
187.   if (nchar(fileReg) == 6){
188.     flight <- gsub("ac=.....", realReg, flight)
189.   }
190.
191.   if (nchar(fileReg) == 7){
192.     flight <- gsub("ac=.....", realReg, flight)
193.   }
194.
195.   if (nchar(fileReg) == 8){
196.     flight <- gsub("ac=.....", realReg, flight)
197.   }
198.
199.   if (nchar(fileReg) == 9){
200.     flight <- gsub("ac=.....", realReg, flight)
201.   }
202.
203.   if (nchar(fileReg) == 10){
204.     flight <- gsub("ac=.....", realReg, flight)
205.   }
206. }
207.
208.
209. # Save rds
210. saveFilename <- paste(flight, "rds", sep = ".")
211. savePath <- paste(saveDirName, saveFilename, sep = "/")
212. saveRDS(tempflt, file = savePath)
213.
214. # Rename CSV
215. # Get a/c reg for path first
216. #   aereg <- strsplit(newCSV[i], "ac=")
217. #   v <- sapply(aereg[1], length)

```

```

218. #   acreg <- matrix(unlist(acreg), ncol = v, byrow = TRUE)
219. #   acreg <- acreg[,2]
220. #   acreg <- strsplit(acreg, " ")
221. #   u <- sapply(acreg[1], length)
222. #   acreg <- matrix(unlist(acreg), ncol = u, byrow = TRUE)
223. #   acreg <- acreg[,1]
224.
225.   saveCSVfull <- paste(CSVlocation, flight, sep = "/")
226.
227.   file.rename(newCSVfull[i], saveCSVfull)
228.
229.
230.   rm("dupe", "endframe", "endoffile", "errortype", "flight", "frame",
231.      "frame4095", "frameadj", "framechk", "framechkrun", "framep32", "framezero1",
232.      "framezero2",
233.      "rlecols", "rleref", "saveFilename", "savePath", "startframe",
234.      "tempflt", "x", "commonday", "commonmonth", "day", "dayi", "firstday",
235.      "firstmonth",
236.      "flight", "newID", "realdate", "realday", "realmonth", "v",
237.      "w", "y", "yearpath", "z", "saveCSVfull", "saveCSVname", "acreg", "u",
238.      "realReg",
239.      "fileReg")
239. # saveRDS(MflightID, file = "MflightID.rds")

```

Appendix 2 – R code for SPC and variability parameter extraction

```
1. # AppVar_v5.R
2. # v1 based on PhExtAB319v18
3. # v2 changed from 1000ft to TD to 3nm to TD.
4. # v3 changed merge to rbind for output tables
5. # v4 change to make use of Rdata files instead of CSV
6. # v5 added & (onground == FALSE) to landlog due to occ descent on TO roll
7.
8. library(zoo) ### for na.locf
9. library(data.table)
10. library(Hmisc) ### for %nin%
11. library(caTools)
12.
13. setwd("~/CloudStation/DB_Work/R/Variance")
14.
15. ## Load already processed flight IDs for duplicate check
16. VflightID <-
  readRDS("/Users/davidbarry/CloudStation/DB_Work/R/Variance/VflightID.rds")
17.
18. ### Get file path list in folder
19. newRDSfull <- list.files("/Volumes/SlimDrive/TestData/Rflights/test", recursive =
  TRUE, full.names = TRUE)
20.
21. RDSid <- strsplit(newRDSfull, "/")
22. x <- sapply(RDSid[1], length)
23. RDSid <- matrix(unlist(RDSid), ncol = x, byrow = TRUE)
24. RDSid <- RDSid[,x]
25. newRDSfull <- newRDSfull[which(RDSid %nin% VflightID)]
26. newRDS <- RDSid[RDSid %nin% VflightID]
27.
28. ## Create empty output tables
29. flight <- character()
30. errortype <- character()
31.
32. ifelse(file.exists("MoutPitch.rds") == TRUE, load("MoutPitch.rds"),
33.        MoutPitch <- data.frame(xFlight = character(0), xDist = numeric(0), xPITCH =
  numeric(0), stringsAsFactors = FALSE))
34.
35. ifelse(file.exists("MoutRoll.rds") == TRUE, load("MoutRoll.rds"),
36.        MoutRoll <- data.frame(xFlight = character(0), xDist = numeric(0), xROLL =
  numeric(0), stringsAsFactors = FALSE))
37.
38. ifelse(file.exists("MoutCAS.rds") == TRUE, load("MoutCAS.rds"),
39.        MoutCAS <- data.frame(xFlight = character(0), xDist = numeric(0), xCAS =
  numeric(0), stringsAsFactors = FALSE))
40.
41. ifelse(file.exists("MoutCASdd.rds") == TRUE, load("MoutCASd.rds"),
42.        MoutCASd <- data.frame(xFlight = character(0), xDist = numeric(0), xCASd =
  numeric(0), stringsAsFactors = FALSE))
43.
44. ifelse(file.exists("MoutGS.rds") == TRUE, load("MoutGS.rds"),
45.        MoutGS <- data.frame(xFlight = character(0), xDist = numeric(0), xGS =
  numeric(0), stringsAsFactors = FALSE))
46.
47. ifelse(file.exists("MoutHaLD.rds") == TRUE, load("MoutHaLD.rds"),
48.        MoutHaLD <- data.frame(xFlight = character(0), xDist = numeric(0), xHaLD =
  numeric(0), stringsAsFactors = FALSE))
49.
50. ifelse(file.exists("MoutRALTL1.rds") == TRUE, load("MoutRALTL1.rds"),
51.        MoutRALTL1 <- data.frame(xFlight = character(0), xDist = numeric(0), xRALTL1 =
  numeric(0), stringsAsFactors = FALSE))
52.
53. ifelse(file.exists("MoutN11.rds") == TRUE, load("MoutN11.rds"),
54.        MoutN11 <- data.frame(xFlight = character(0), xDist = numeric(0), xN11 =
  numeric(0), stringsAsFactors = FALSE))
55.
56. ifelse(file.exists("MoutN12.rds") == TRUE, load("MoutN12.rds"),
```

```

57.     MoutN12 <- data.frame(xFlight = character(0), xDist = numeric(0), xN12 =
numeric(0), stringsAsFactors = FALSE))
58.
59. ifelse(file.exists("MoutSSPP1.rds") == TRUE, load("MoutSSPP1.rds"),
60.     MoutSSPP1 <- data.frame(xFlight = character(0), xDist = numeric(0), xSSPP1 =
numeric(0), stringsAsFactors = FALSE))
61.
62. ifelse(file.exists("MoutSSRP1.rds") == TRUE, load("MoutSSRP1.rds"),
63.     MoutSSRP1 <- data.frame(xFlight = character(0), xDist = numeric(0), xSSRP1 =
numeric(0), stringsAsFactors = FALSE))
64.
65. ifelse(file.exists("MoutSSPP2.rds") == TRUE, load("MoutSSPP2.rds"),
66.     MoutSSPP2 <- data.frame(xFlight = character(0), xDist = numeric(0), xSSPP2 =
numeric(0), stringsAsFactors = FALSE))
67.
68. ifelse(file.exists("MoutSSRP2.rds") == TRUE, load("MoutSSRP2.rds"),
69.     MoutSSRP2 <- data.frame(xFlight = character(0), xDist = numeric(0), xSSRP2 =
numeric(0), stringsAsFactors = FALSE))
70.
71. ifelse(file.exists("MoutRUDPOS.rds") == TRUE, load("MoutRUDPOS.rds"),
72.     MoutRUDPOS <- data.frame(xFlight = character(0), xDist = numeric(0), xRUDPOS =
numeric(0), stringsAsFactors = FALSE))
73.
74. ifelse(file.exists("MoutNMLA.rds") == TRUE, load("MoutNMLA.rds"),
75.     MoutNMLA <- data.frame(xFlight = character(0), xDist = numeric(0), xNMLA =
numeric(0), stringsAsFactors = FALSE))
76.
77. ifelse(file.exists("MoutLONACC.rds") == TRUE, load("MoutLONACC.rds"),
78.     MoutLONACC <- data.frame(xFlight = character(0), xDist = numeric(0), xLONACC =
numeric(0), stringsAsFactors = FALSE))
79.
80. ifelse(file.exists("MoutLATACC.rds") == TRUE, load("MoutLATACC.rds"),
81.     MoutLATACC <- data.frame(xFlight = character(0), xDist = numeric(0), xLATACC =
numeric(0), stringsAsFactors = FALSE))
82.
83. ifelse(file.exists("MoutMHDGd.rds") == TRUE, load("MoutMHDGd.rds"),
84.     MoutMHDGd <- data.frame(xFlight = character(0), xDist = numeric(0), xMHDGd =
numeric(0), stringsAsFactors = FALSE))
85.
86. errors <- data.frame(flight = character(0), errortype = character(0), stringsAsFactors
= FALSE)
87. colnames(errors) <- c("Flight", "Error")
88.
89. for (i in 1:length(newRDS)) {
90.     # Progress
91.     cat(paste(".", i, "."))
92.
93.     ## flight ID for output
94.     flight <- newRDS[i]
95.
96.     ## Create new variables
97.     xFlight <- character()
98.     xDist <- numeric()
99.     xPITCH <- numeric()
100.    xROLL <- numeric()
101.    xCAS <- numeric()
102.    xCASd <- numeric()
103.    xGS <- numeric()
104.    xHaLD <- numeric()
105.    xRALT1 <- numeric()
106.    xN11 <- numeric()
107.    xN12 <- numeric()
108.    xSSPP1 <- numeric()
109.    xSSRP1 <- numeric()
110.    xSSPP2 <- numeric()
111.    xSSRP2 <- numeric()
112.    xRUDPOS <- numeric()
113.    xNMLA <- numeric()
114.    xLONACC <- numeric()
115.    xLATACC <- numeric()
116.    xMHDGd <- numeric()
117.
118.    ## Check if file has been seen before
119.    dupe <- is.element(flight, VflightID)
120.    if (dupe == TRUE) {
121.        errortype <- "Duplicate"
122.        errors[nrow(errors) + 1,] <- c(flight, errortype)

```

```

123.     next}
124.
125. VflightID <- c(VflightID, flight) # add this flight to Master flight list
126.
127. # Read in data and make good column names
128.
129. tempflt <- readRDS(newRDSfull[i])
130.
131.
132. ## Clean up used numerical params to remove "***". Makes param numeric, fills blanks
with NA then backfills
133. ## NA with last value
134. CAS <- as.numeric(tempflt$Computed.Airspeed)
135. CAS <- na.approx(CAS, na.rm = FALSE)
136. N11 <- as.numeric(tempflt$N1.Actual.Eng.1)
137. N11 <- na.approx(N11, na.rm = FALSE)
138. N12 <- as.numeric(tempflt$N1.Actual.Eng.2)
139. N12 <- na.approx(N12, na.rm = FALSE)
140. GS <- as.numeric(tempflt$Ground.Speed)
141. GS <- na.approx(GS, na.rm = FALSE)
142. RALT1 <- as.numeric(tempflt$Radio.Altitude.1)
143. RALT1 <- na.approx(RALT1, na.rm = FALSE)
144. ALT <- as.numeric(tempflt$Altitude..1013.25mB.)
145. ALT <- na.approx(ALT, na.rm = FALSE)
146. # Check if ALT good
147. if (max(ALT, na.rm = TRUE) < 200) {
148.   errortype <- "Poor ALT"
149.   errors[nrow(errors) + 1,] <- c(flight, errortype)
150.   next
151. }
152. FLAPANG <- as.numeric(tempflt$Flap.Angle)
153. FLAPANG <- na.approx(FLAPANG, na.rm = FALSE)
154. PITCH <- as.numeric(tempflt$Pitch.Angle)
155. PITCH <- na.approx(PITCH, na.rm = FALSE)
156. ROLL <- as.numeric(tempflt$Roll.Angle)
157. ROLL <- na.approx(ROLL, na.rm = FALSE)
158. NMLA <- as.numeric(tempflt$Normal.Acceleration)
159. APPSPD <- as.numeric(tempflt$Approach.Speed.Target)
160. APPSPD <- na.approx(APPSPD, na.rm = FALSE)
161. LATACC <- as.numeric(tempflt$Lateral.Acceleration)
162. LATACC <- na.approx(LATACC, na.rm = FALSE)
163. LONACC <- as.numeric(tempflt$Longitudinal.Acceleration)
164. LONACC <- na.approx(LONACC, na.rm = FALSE)
165. MHDG <- as.numeric(tempflt$Magnetic.Heading)
166. MHDG <- na.approx(MHDG, na.rm = FALSE)
167. RUDPOS <- as.numeric(tempflt$Rudder.Position)
168. RUDPOS <- na.approx(RUDPOS, na.rm = FALSE)
169. SSPP1 <- as.numeric(tempflt$Sidestick.Pos.Capt.Pitch)
170. SSPP1 <- na.approx(SSPP1, na.rm = FALSE)
171. SSRP1 <- as.numeric(tempflt$Sidestick.Pos.Capt.Roll)
172. SSRP1 <- na.approx(SSRP1, na.rm = FALSE)
173. SSPP2 <- as.numeric(tempflt$Sidestick.Pos.F.O.Pitch)
174. SSPP2 <- na.approx(SSPP2, na.rm = FALSE)
175. SSRP2 <- as.numeric(tempflt$Sidestick.Pos.F.O.Roll)
176. SSRP2 <- na.approx(SSRP2, na.rm = FALSE)
177. GSDEV1 <- as.numeric(tempflt$Glideslope.Deviation.1)
178. GSDEV1 <- na.approx(GSDEV1, na.rm = FALSE)
179.
180.
181. ## Clean up used character params to remove "***" and backfill blanks.
182. gear <- gsub("[^[:alnum:]]//'", "", tempflt$Gear.Selector.Down)
183. gear[gear == ""] <- NA ### fills blanks with NA
184. gear <- na.locf(gear, na.rm = FALSE) ### Backfills NA with last value
185.
186. SQ_L <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...LH)
187. SQ_L[SQ_L == ""] <- NA
188. SQ_L <- na.locf(SQ_L, na.rm = FALSE)
189.
190. SQ_N <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...Nose)
191. SQ_N[SQ_N == ""] <- NA
192. SQ_N <- na.locf(SQ_N, na.rm = FALSE)
193.
194. SQ_R <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...RH)
195. SQ_R[SQ_R == ""] <- NA
196. SQ_R <- na.locf(SQ_R, na.rm = FALSE)
197.
198.

```

```

199.  ## Gear is down.
200.  geardn <- (gear == "Down 1")
201.
202.  ## Shutdown
203.  shutdownlog <- (N11 < 20 & N12 < 20)
204.
205.  ## Aircraft is on ground
206.  onground <- (SQ_L == "Gnd 1" | SQ_N == "Gnd 1" | SQ_R == "Gnd 1")
207.  OGcnt <- (sum(onground, na.rm = TRUE))/8
208.
209.  ## Set up empties
210.  descending <- logical(length(CAS))
211.  climbing <- logical(length(CAS))
212.
213.  ## Climbing or descending?
214.
215.  ALTp8 <- c(NA + 1:8, ALT[1:(length(ALT) - 8)])
216.  ALTp16 <- c(NA + 1:16, ALT[1:(length(ALT) - 16)])
217.  ALTp24 <- c(NA + 1:24, ALT[1:(length(ALT) - 24)])
218.  ALTp32 <- c(NA + 1:32, ALT[1:(length(ALT) - 32)])
219.  ALTp40 <- c(NA + 1:40, ALT[1:(length(ALT) - 40)])
220.
221.  descending <- ((ALT < ALTp8) & (ALTp8 < ALTp16) & (ALTp16 < ALTp24) & (ALTp24 <
  ALTp32) & (ALTp32 < ALTp40) & (ALT < ALTp40 - 42))
222.  climbing <- ((ALT > ALTp8) & (ALTp8 > ALTp16) & (ALTp16 > ALTp24) & (ALTp24 >
  ALTp32) & (ALTp32 > ALTp40) & (ALT > ALTp40 + 42))
223.
224.  ## Take off point
225.  ongroundp1 <- c(NA + 1:16, onground[1:(length(onground) - 16)]) # creates vector
  shifting onground 2 secs down
226.  ongroundp1 <- as.logical(ongroundp1)
227.
228.  TOsearch <- logical(length(CAS))
229.  TOsearch <- (onground == FALSE) & (ongroundp1 == TRUE) & (geardn == TRUE) & (GS >
  60)
230.
231.  TakePoint <- which(TOsearch == TRUE)
232.
233.  if (length(TakePoint) == 0) {
234.
235.    errortype <- "NO T/O"
236.    errors[nrow(errors) + 1,] <- c(flight, errortype)
237.
238.    next
239.  }
240.
241.  TakePoint <- TakePoint[1]
242.
243.  ## Landing Point
244.  LDsearch <- logical(length(CAS))
245.  LDsearch <- (onground == TRUE) & (ongroundp1 == FALSE) & (geardn == TRUE) & (GS >
  60) & (ALT > -1000) &
246.  (ALT < 12000) & (GS < 250)
247.
248.  LandPoint <- which(LDsearch == TRUE)
249.  if (length(LandPoint) == 0) {
250.
251.    errortype <- "NO LANDING"
252.    errors[nrow(errors) + 1,] <- c(flight, errortype)
253.
254.    next
255.  }
256.
257.  LandPoint <- LandPoint[length(LandPoint)]
258.
259.  ## Height above TO runway
260.  TOelev <- mean(ALT[TakePoint - 40], ALT[TakePoint - 48], ALT[TakePoint - 56],
  ALT[TakePoint - 64], ALT[TakePoint - 72], ALT[TakePoint - 80])
261.  HaTO <- (ALT - TOelev)
262.
263.  ## Height above LD runway
264.  LDelev <- round(mean(ALT[LandPoint + 40], ALT[LandPoint + 48], ALT[LandPoint +
  56], ALT[LandPoint + 64], ALT[LandPoint + 72],
265.  ALT[LandPoint + 80]))
266.  HaLD <- (ALT - LDelev)
267.
268.  ## Taxi. Note not out/in yet

```



```

269.   taxipwr <- (N11 < 65 & N11 >= 20) |
270.     (N12 < 65 & N12 >= 20) ## does OR cause overlap with shutdown?
271.   taxilog <- (onground == TRUE & taxipwr == TRUE & GS < 30)
272.
273.   ## Take off
274.   takeofflog <- (HaTO < 100) & (N11 >= 65) & (N12 >= 65) &
275.     (geardn == TRUE) & (GS >= 15) & (FLAPANG != "0")
276.
277.   ## Initial climb
278.   inclimblog <- (climbing == TRUE) & (HaTO >= 100) & (onground == FALSE) &
279.     (CAS > 90) & (FLAPANG != "0")
280.
281.   ## Climb
282.   climblog <- (climbing == TRUE) & (FLAPANG == "0") & (geardn == FALSE) &
283.     (onground == FALSE)
284.
285.   ## Cruise
286.   cruiselog <- (climbing == FALSE) & (descending == FALSE) & (FLAPANG == "0") &
287.     (geardn == FALSE) & (ALT > 15000)
288.
289.   ## Descent
290.   descentlog <- (descending == TRUE) & (FLAPANG == "0") &
291.     (onground == FALSE) & (geardn == FALSE)
292.
293.   ## Approach
294.   approachlog <- (descending == TRUE) & (FLAPANG != "0") &
295.     (onground == FALSE) & (HaLD > 200)
296.
297.   ## Landing
298.   landinglog <- (descending == TRUE) & (FLAPANG != "0") & (HaLD <= 200) &
299.     (geardn == TRUE) & (onground == FALSE) ### will end on touchdown due to
    descending
300.
301.
302.   ## Current phase
303.   phaseNow <- character(length(CAS))
304.
305.   phaseNow[(shutdownlog == "TRUE")] <- "SHUTDOWN"
306.   phaseNow[(taxilog == "TRUE")] <- "TAXI"
307.   phaseNow[(takeofflog == "TRUE")] <- "TAKEOFF"
308.   phaseNow[(inclimblog == "TRUE")] <- "INCLIMB"
309.   phaseNow[(climblog == "TRUE")] <- "CLIMB"
310.   phaseNow[(cruiselog == "TRUE")] <- "CRUISE"
311.   phaseNow[(descentlog == "TRUE")] <- "DESCENT"
312.   phaseNow[(approachlog == "TRUE")] <- "APPROACH"
313.   phaseNow[(landinglog == "TRUE")] <- "LANDING"
314.
315.
316.   ## convert level segments in Approach from "" to APPROACH (is this a fudge? Will
    do so for all phases)
317.   phaseNow[phaseNow == ""] <- NA ### fills blanks with NA
318.   phaseNow <- na.locf(phaseNow, na.rm = FALSE) ### Backfills NA with last value
319.
320.   topDes <- which(descentlog == TRUE)
321.   topDes <- topDes[1]
322.
323.   HaLD[1:topDes] <- NA
324.
325.   HaLD3000 <- which.min(abs(HaLD - 3000)) # made HaLD NA pre topDes to avoid 1000ft
    on climb
326.
327.   # Precise point of TD
328.
329.   NMLApl <- c(NA, NMLA[1:(length(NMLA) - 1)])
330.
331.   precTD <- which(((NMLA[(LandPoint - 40):(LandPoint + 40)]) - (NMLApl[(LandPoint -
    40):(LandPoint + 40)])) > 0.1)
332.   ifelse((length(precTD) == 0), (precTD <- LandPoint), (precTD <- precTD[1] + 1))
333.
334.   if (precTD != LandPoint) {precTD <- (LandPoint - 40 + precTD)}
335.
336.   time2TD <- c((precTD - 1):(precTD - (length(CAS))))
337.   time2TD <- time2TD * -1
338.
339.   # Distance
340.
341.   Distance <- integer(length(CAS))

```

```

342. endRoll <- precTD + 120
343. threek2roll <- HaLD3000:endRoll
344.
345. Distance[threek2roll] <- cumsum(GS[threek2roll]/28800)
346.
347. DistanceTD <- integer(length(CAS))
348. DistanceTD[threek2roll] <- Distance[threek2roll] - Distance[precTD]
349.
350. Distance3m <- which.min(abs(DistanceTD + 3))
351.
352. Dist3m <- Distance3m:endRoll
353.
354. # Runway heading
355.
356. rwyHDG <- round(mean(MHDG[(endRoll - 40):endRoll]))
357.
358. # Parameter extracts for 1000ft to endRoll
359. xPITCH <- PITCH[Dist3m]
360. xROLL <- ROLL[Dist3m]
361. xCAS <- CAS[Dist3m]
362. xGS <- GS[Dist3m]
363. xHaLD <- HaLD[Dist3m]
364. xDist <- DistanceTD[Dist3m]
365. xRALT1 <- RALT1[Dist3m]
366. xN11 <- N11[Dist3m]
367. xN12 <- N12[Dist3m]
368. xSSPP1 <- SSPP1[Dist3m]
369. xSSRP1 <- SSRP1[Dist3m]
370. xSSPP2 <- SSPP2[Dist3m]
371. xSSRP2 <- SSRP2[Dist3m]
372. xRUDPOS <- RUDPOS[Dist3m]
373. xNMLA <- NMLA[Dist3m]
374. xLONACC <- LONACC[Dist3m]
375. xLATACC <- LATACC[Dist3m]
376. xGSDEV1 <- GSDEV1[Dist3m]
377.
378. # Delta params
379. xCASd <- xCAS - APPSPD[Dist3m]
380. xMHDGd <- MHDG[Dist3m] - rwyHDG
381.
382. # Outputs
383. xFlight[1:length(xDist)] <- flight
384.
385. outPitch <- data.frame(xFlight, xDist, xPITCH, stringsAsFactors = FALSE)
386. MoutPitch <- rbind(MoutPitch, outPitch)
387.
388. outRoll <- data.frame(xFlight, xDist, xROLL, stringsAsFactors = FALSE)
389. MoutRoll <- rbind(MoutRoll, outRoll)
390.
391. outCAS <- data.frame(xFlight, xDist, xCAS, stringsAsFactors = FALSE)
392. MoutCAS <- rbind(MoutCAS, outCAS)
393.
394. outGS <- data.frame(xFlight, xDist, xGS, stringsAsFactors = FALSE)
395. MoutGS <- rbind(MoutGS, outGS)
396.
397. outHaLD <- data.frame(xFlight, xDist, xHaLD, stringsAsFactors = FALSE)
398. MoutHaLD <- rbind(MoutHaLD, outHaLD)
399.
400. outRALT1 <- data.frame(xFlight, xDist, xRALT1, stringsAsFactors = FALSE)
401. MoutRALT1 <- rbind(MoutRALT1, outRALT1)
402.
403. outN11 <- data.frame(xFlight, xDist, xN11, stringsAsFactors = FALSE)
404. MoutN11 <- rbind(MoutN11, outN11)
405.
406. outN12 <- data.frame(xFlight, xDist, xN12, stringsAsFactors = FALSE)
407. MoutN12 <- rbind(MoutN12, outN12)
408.
409. outSSPP1 <- data.frame(xFlight, xDist, xSSPP1, stringsAsFactors = FALSE)
410. MoutSSPP1 <- rbind(MoutSSPP1, outSSPP1)
411.
412. outSSRP1 <- data.frame(xFlight, xDist, xSSRP1, stringsAsFactors = FALSE)
413. MoutSSRP1 <- rbind(MoutSSRP1, outSSRP1)
414.
415. outSSPP2 <- data.frame(xFlight, xDist, xSSPP2, stringsAsFactors = FALSE)
416. MoutSSPP2 <- rbind(MoutSSPP2, outSSPP2)
417.
418. outSSRP2 <- data.frame(xFlight, xDist, xSSRP2, stringsAsFactors = FALSE)

```

```

419.   MoutSSRP2 <- rbind(MoutSSRP2, outSSRP2)
420.
421.   outRUDPOS <- data.frame(xFlight, xDist, xRUDPOS, stringsAsFactors = FALSE)
422.   MoutRUDPOS <- rbind(MoutRUDPOS, outRUDPOS)
423.
424.   outNMLA <- data.frame(xFlight, xDist, xNMLA, stringsAsFactors = FALSE)
425.   MoutNMLA <- rbind(MoutNMLA, outNMLA)
426.
427.   outLONACC <- data.frame(xFlight, xDist, xLONACC, stringsAsFactors = FALSE)
428.   MoutLONACC <- rbind(MoutLONACC, outLONACC)
429.
430.   outLATACC <- data.frame(xFlight, xDist, xLATACC, stringsAsFactors = FALSE)
431.   MoutLATACC <- rbind(MoutLATACC, outLATACC)
432.
433.   outCASd <- data.frame(xFlight, xDist, xCASd, stringsAsFactors = FALSE)
434.   MoutCASd <- rbind(MoutCASd, outCASd)
435.
436.   outMHDGd <- data.frame(xFlight, xDist, xMHDGd, stringsAsFactors = FALSE)
437.   MoutMHDGd <- rbind(MoutMHDGd, outMHDGd)
438.
439.   # remove("ALT",      "ALTp16",    "ALTp24",    "ALTp32",    "ALTp40",    "ALTp8",
"approachlog", "APPSPD",    "CAS", "climbing",
440.   #       "climblog", "cruiselog", "descending", "descentlog", "Distance",
"DistanceTD", "Distance3m", "Dist3m", "dupe", "endframe", "endoffile",
441.   #       "endRoll", "errortype", "FLAPANG", "flight", "frame",
"frame4095", "frameadj", "framechk", "framechkrun",
442.   #       "framep32", "framezero1", "framezero2", "gear", "geardn", "GS",
"HaLD", "HaLD3000", "HaTO", "i",
443.   #       "inclimblog", "landinglog", "LandPoint", "LATACC", "LDelev",
"LDsearch", "LONACC", "MHDG", "N11", "N12",
444.   #       "NMLA", "NMLAp1", "OGcnt", "threek2roll", "onground",
"ongroundp1", "outPitch", "outCAS", "outCASd", "outGS",
445.   #       "outHaLD", "outLATACC", "outLONACC", "outMHDGd", "outN11",
"outN12", "outNMLA", "outRALt1",
446.   #       "outRoll", "outRUDPOS", "outSSPP1", "outSSPP2", "outSSRP1",
"outSSRP2", "phaseNow", "PITCH",
447.   #       "precTD", "RALt1", "rlecols", "rleref", "ROLL",
"RUDPOS", "rwyHDG", "shutdownlog", "SQ_L", "SQ_N",
448.   #       "SQ_R", "SSPP1", "SSPP2", "SSRP1", "SSRP2",
"startframe", "takeofflog", "TakePoint", "taxilog", "taxipwr",
449.   #       "tempflt", "time2TD", "TOelev", "topDes", "TOsearch", "x",
"xCAS", "xCASd", "xDist",
450.   #       "xFlight", "xGS", "xHaLD", "xLATACC", "xLONACC",
"xMHDGd", "xN11", "xN12", "xNMLA", "xPITCH",
451.   #       "xRALt1", "xROLL", "xRUDPOS", "XRUDPOS", "xSSPP1",
"xSSPP2", "xSSRP1", "xSSRP2")
452. }
453.
454. #save(MoutPitch, file = "MoutPitch.rds")
455. #save(VflightID, file = "VflightID.rds")

```

Appendix 3 – Examples of flight parameter variability during approach

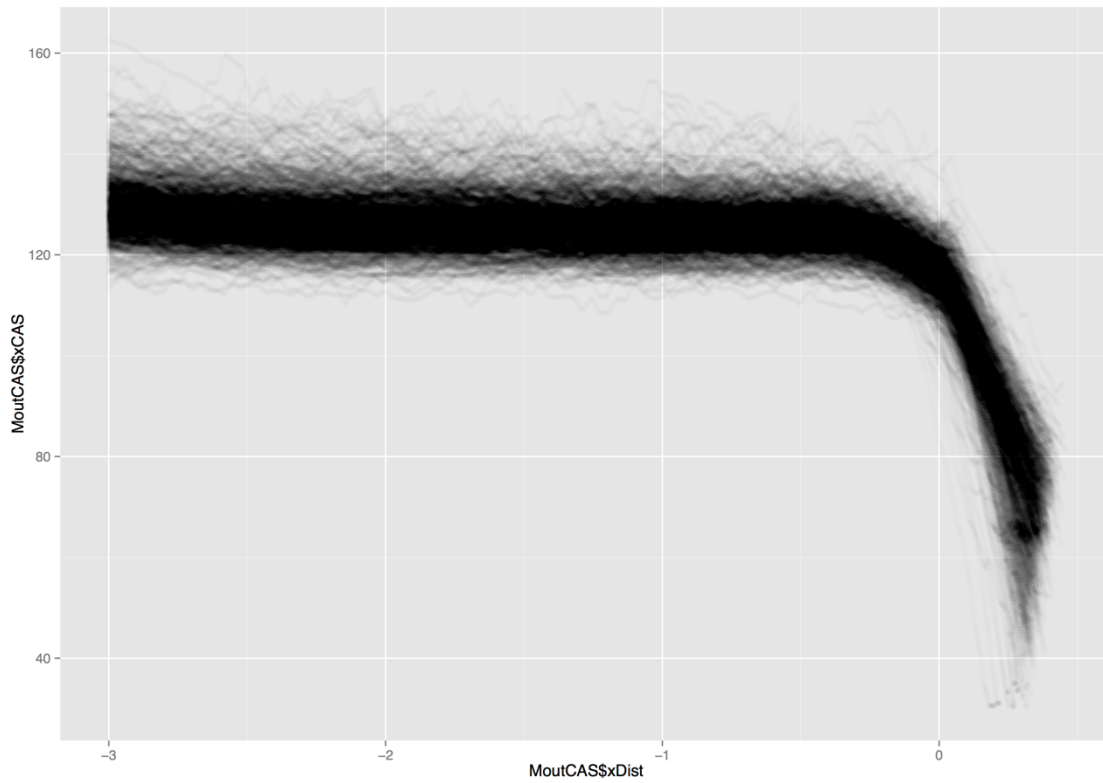


Figure 67. Computed airspeed (CAS) vs distance to touchdown (NM)

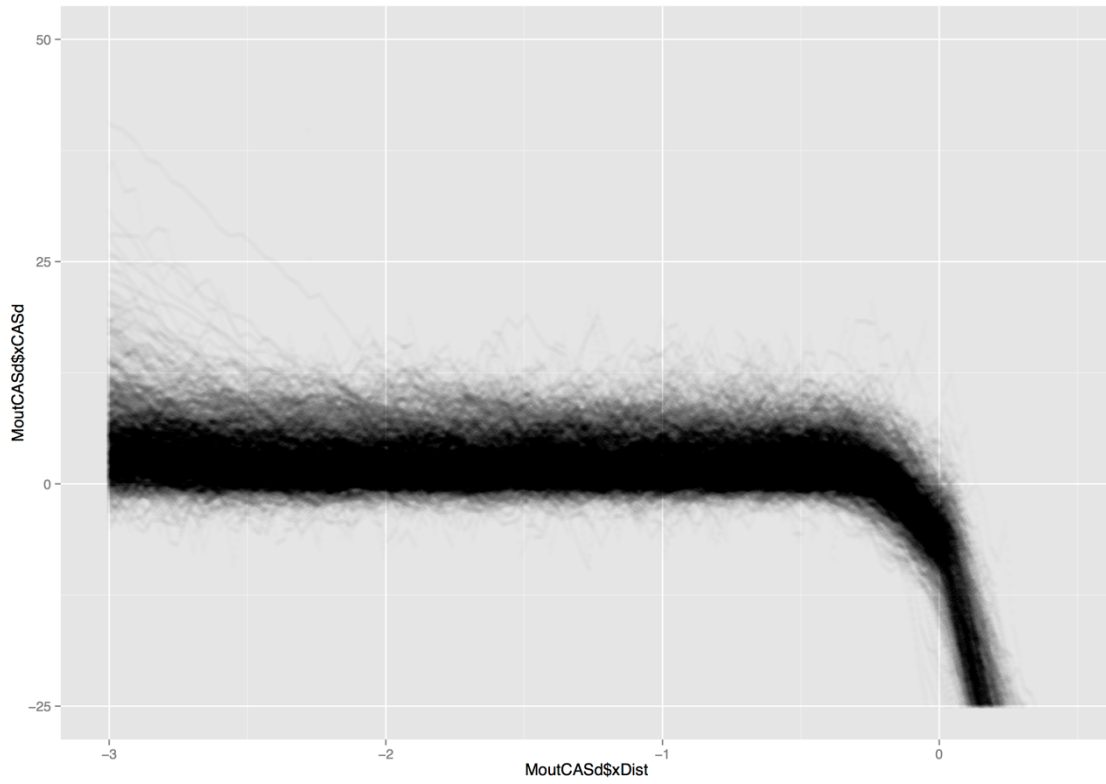


Figure 68. CAS delta from target vs distance to touchdown (NM)

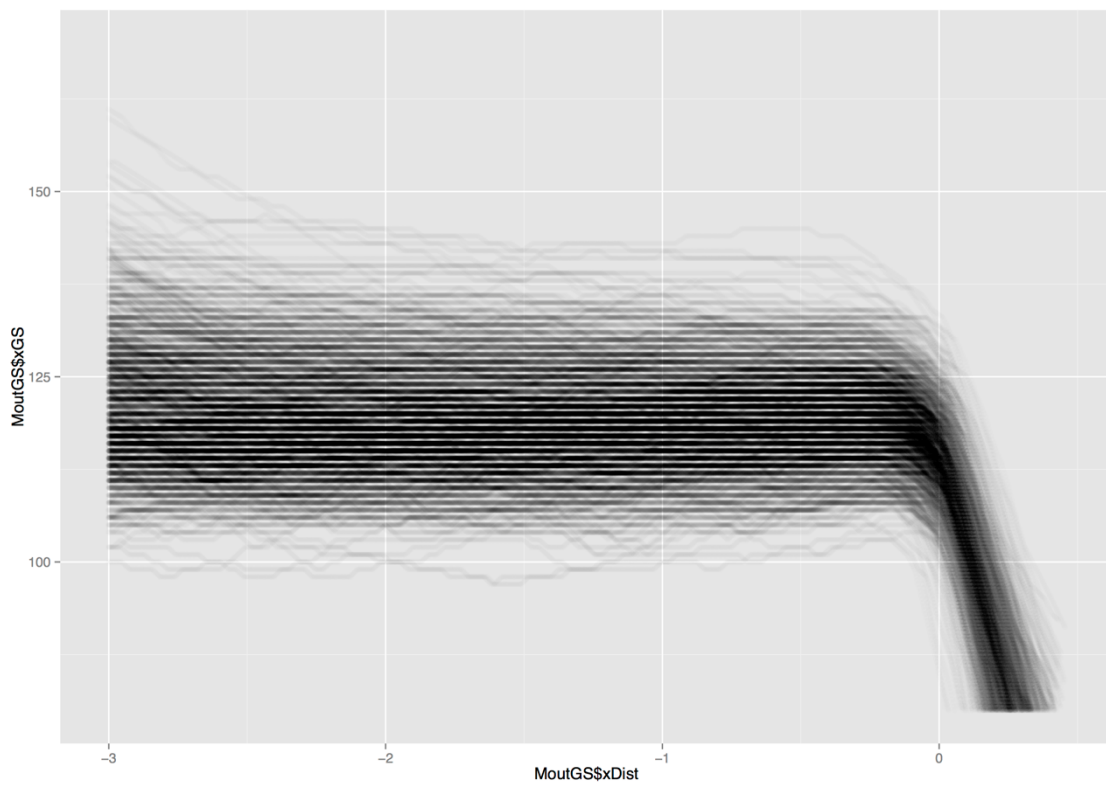


Figure 69. Groundspeed vs distance to touchdown (NM)

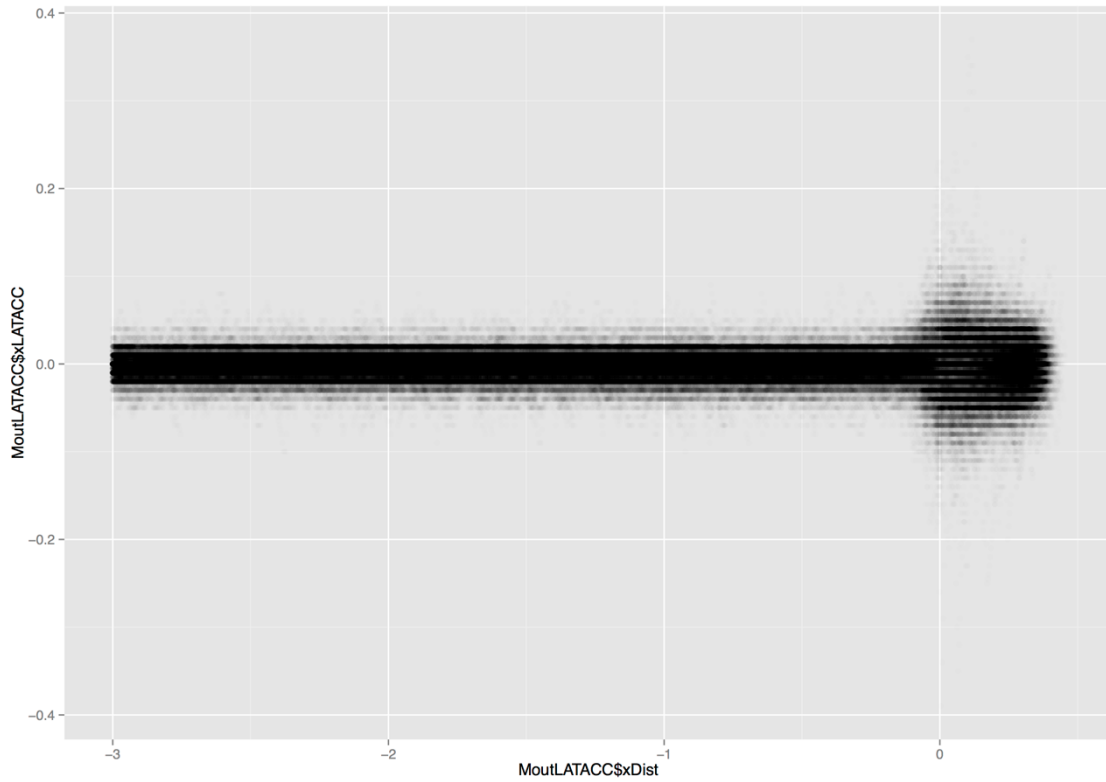


Figure 70. Lateral acceleration (g) vs distance to touchdown (NM)

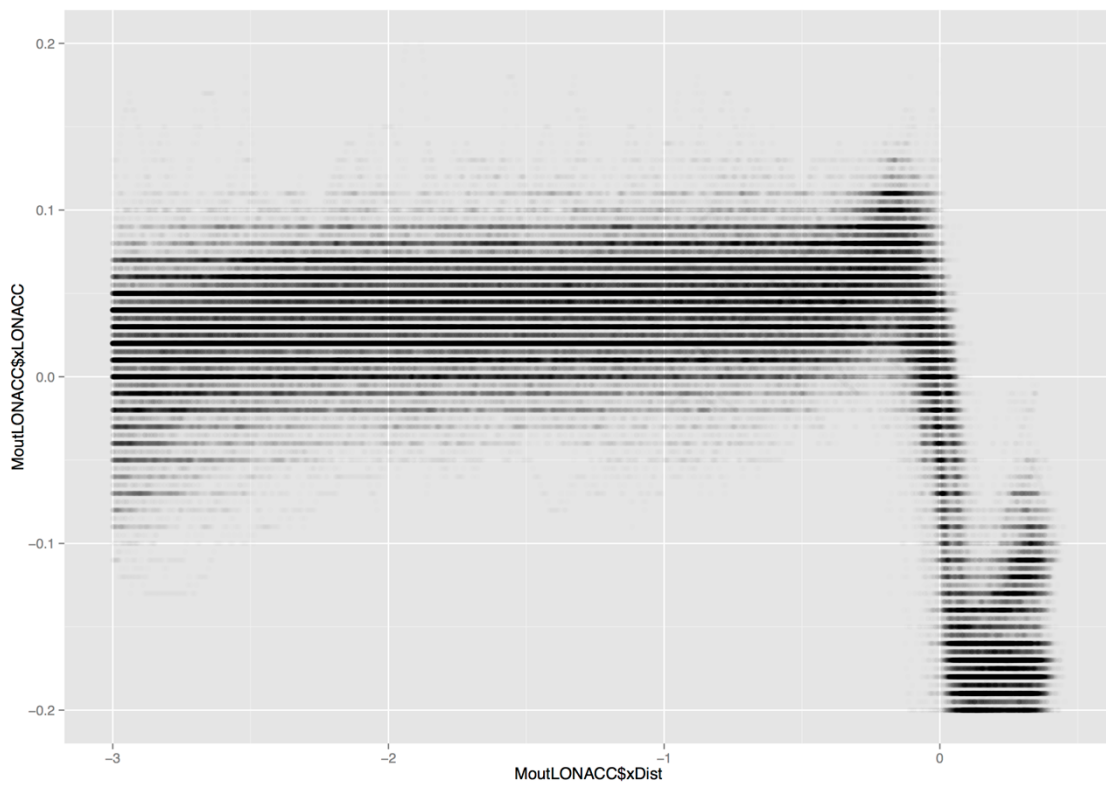


Figure 71. Longitudinal acceleration (g) vs distance to touchdown (NM)

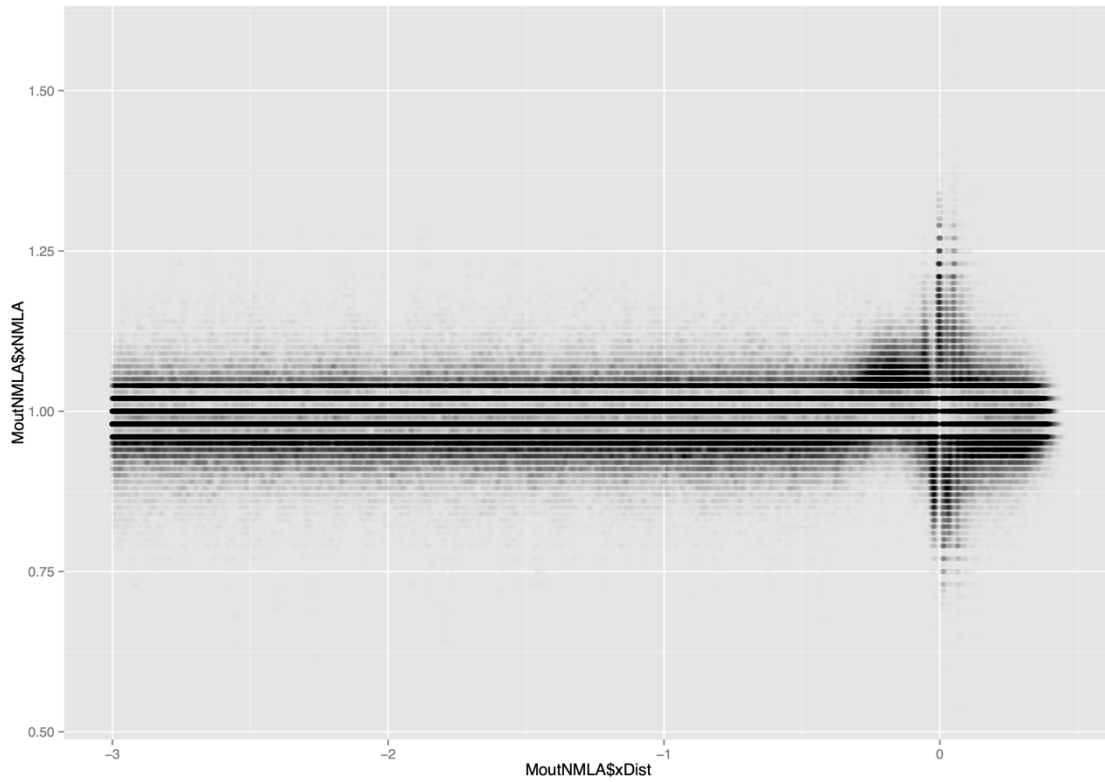


Figure 72. Normal acceleration (g) vs distance to touchdown (NM)

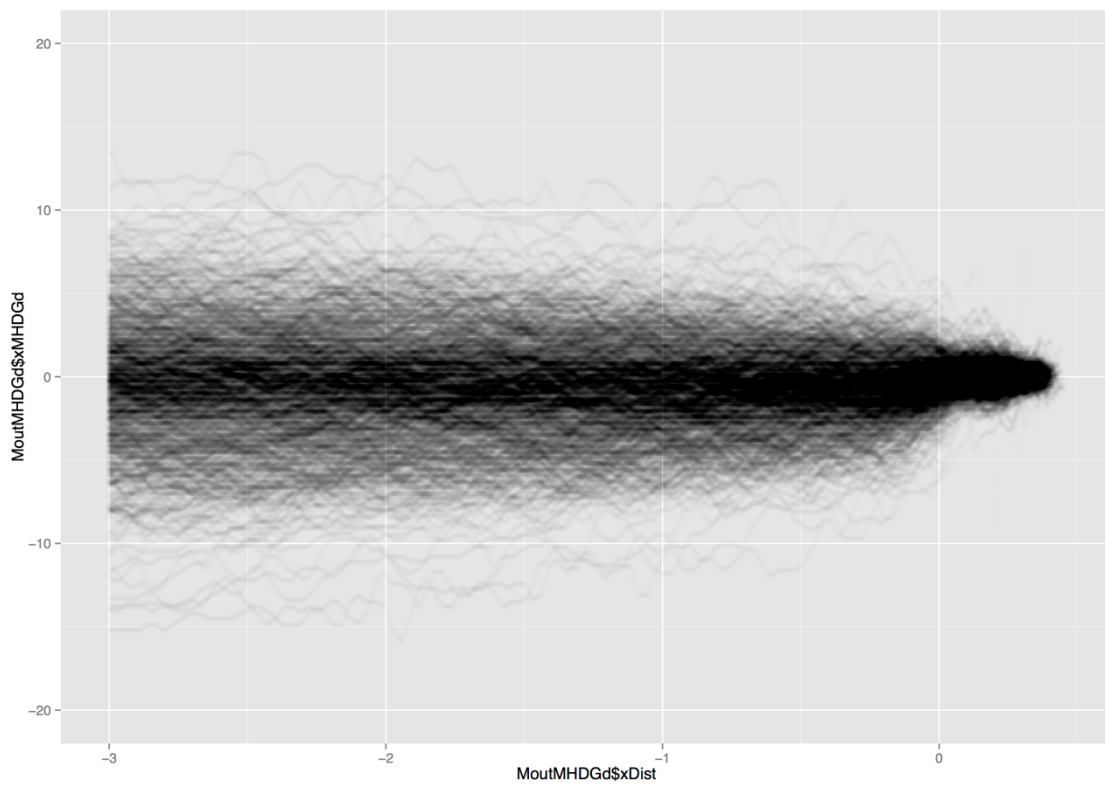


Figure 73. Magnetic heading delta to runway heading (degrees) vs distance to touchdown (NM)

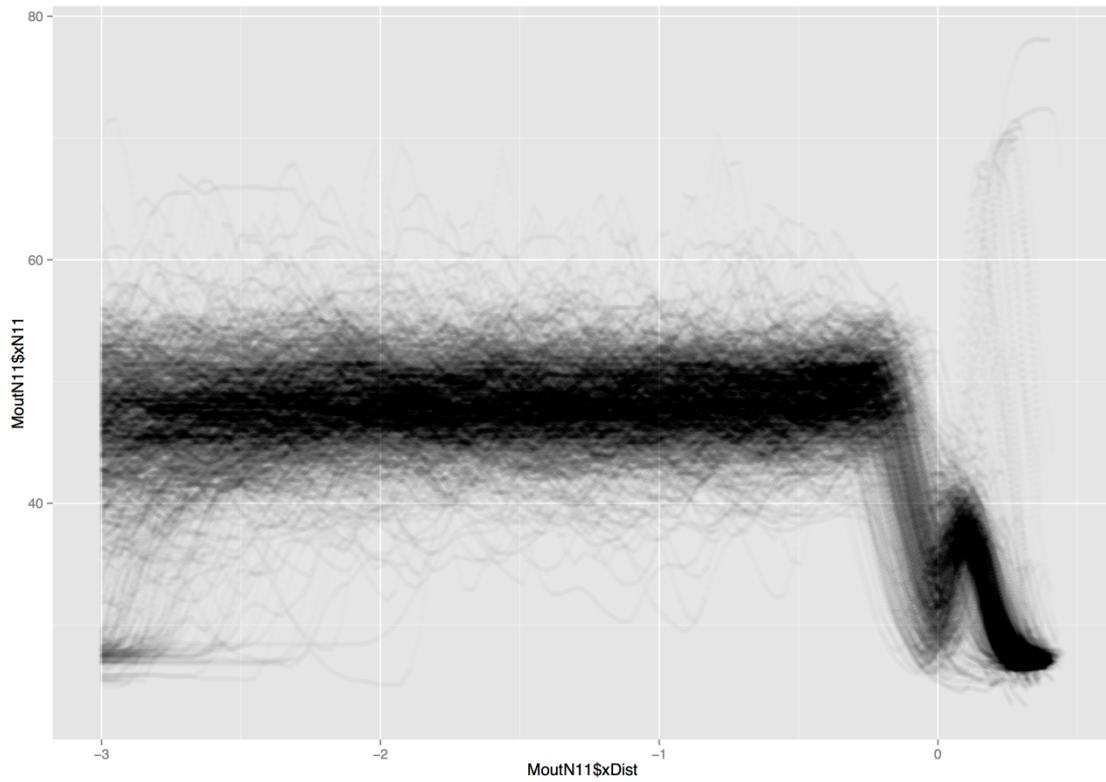


Figure 74. Left engine N1 (%) vs distance to touchdown (NM)

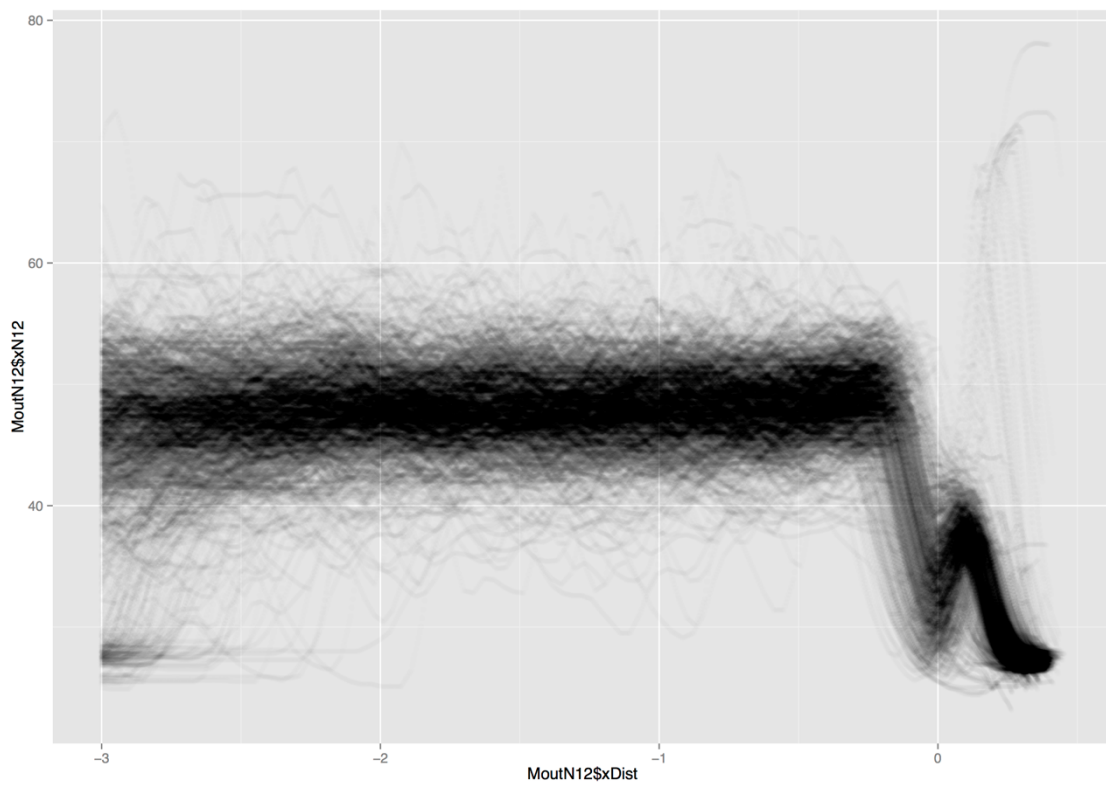


Figure 75. Right engine N1 (%) vs distance to touchdown (NM)

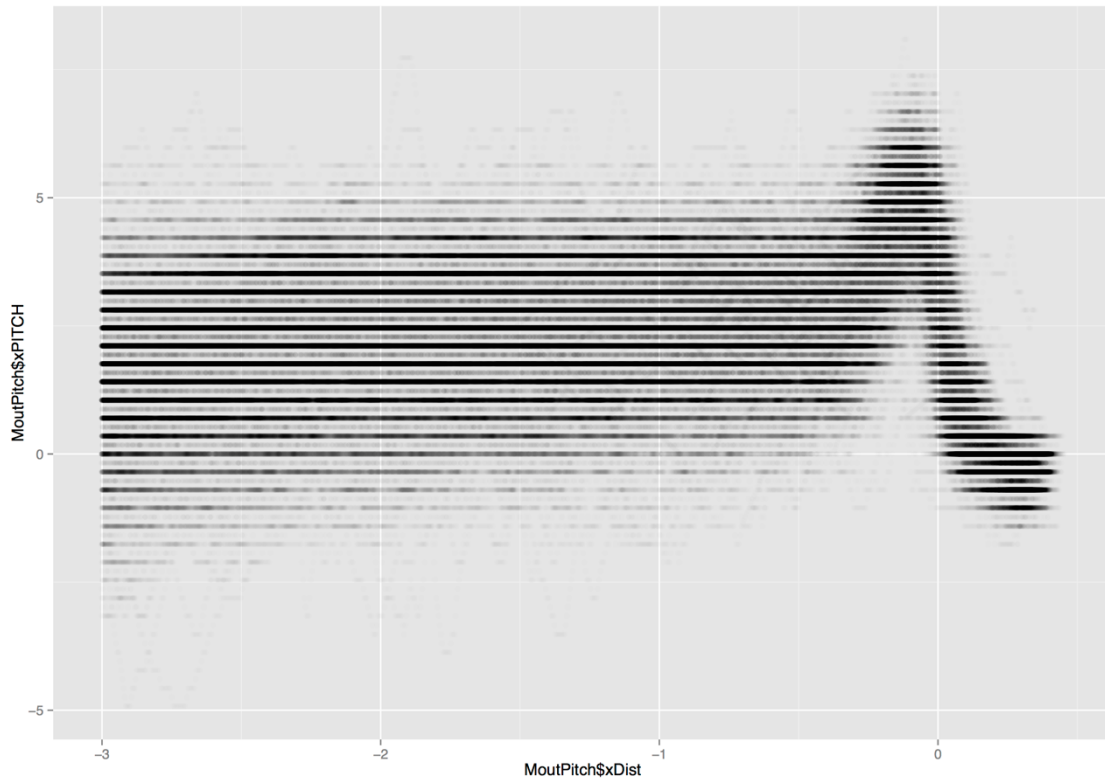


Figure 76. Pitch angle (degrees) vs distance from touchdown (NM)

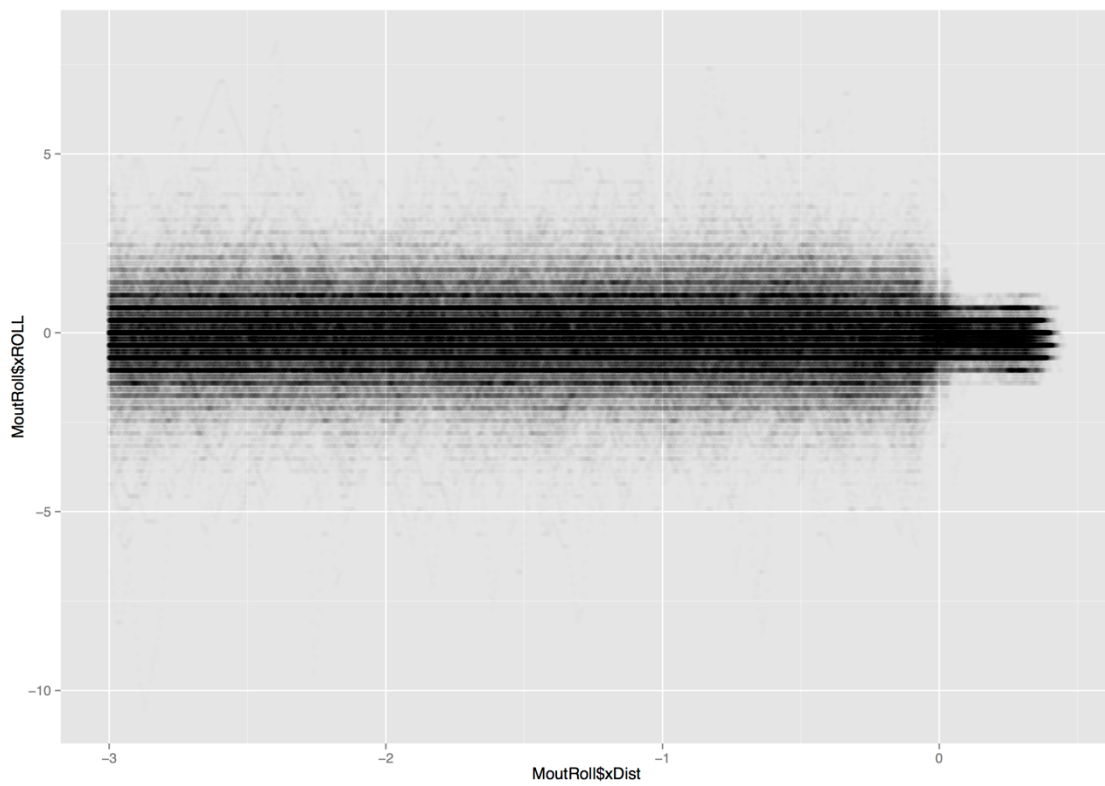


Figure 77. Roll angle (degrees) vs distance from touchdown (NM)

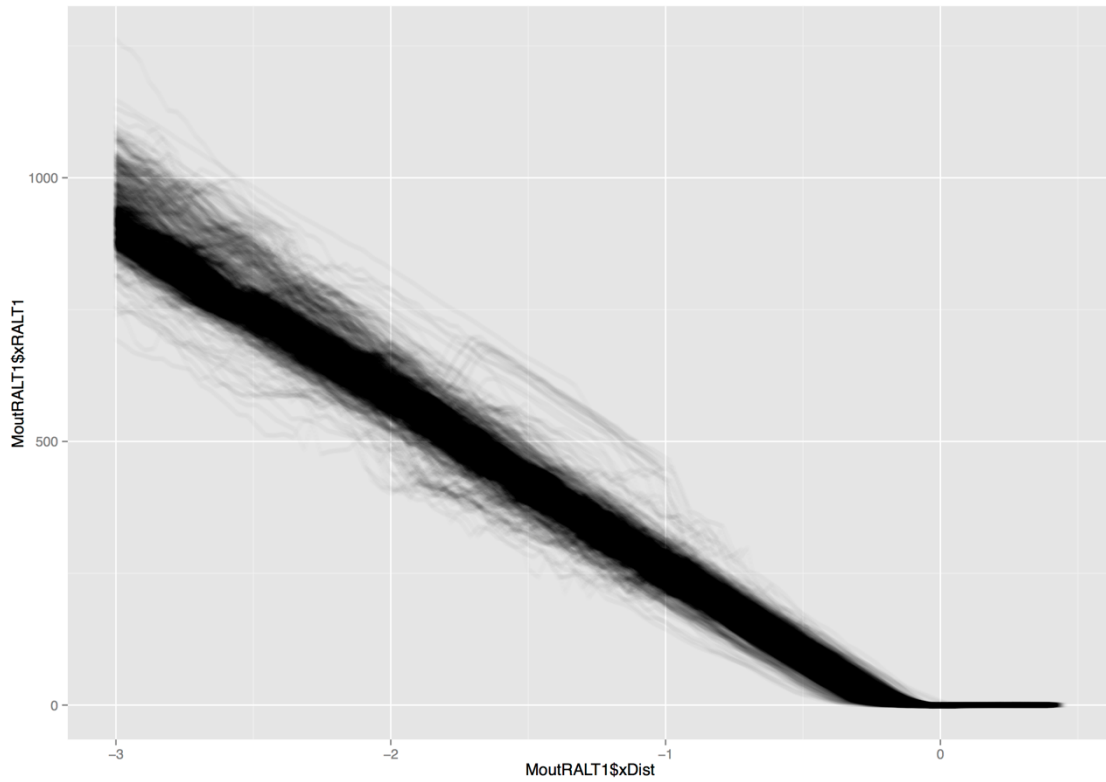


Figure 78. Radio altitude (ft) vs distance from touchdown (NM)

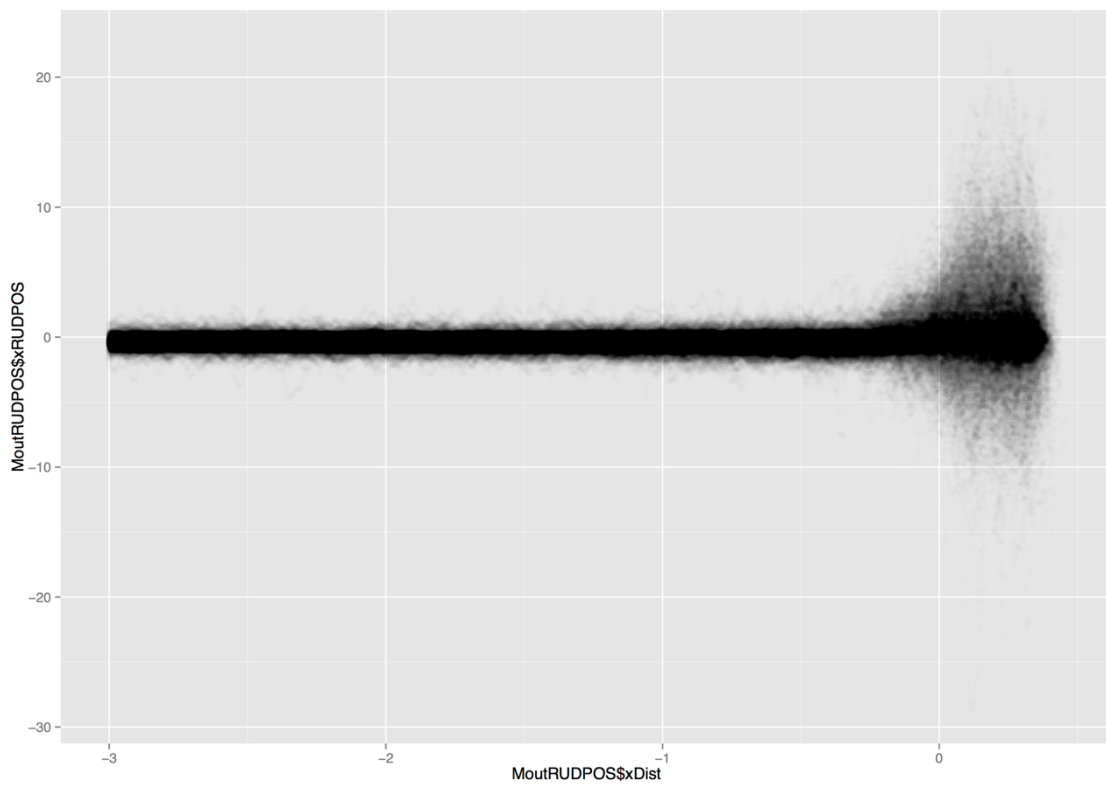


Figure 79. Rudder position (degrees) vs distance from touchdown (NM)

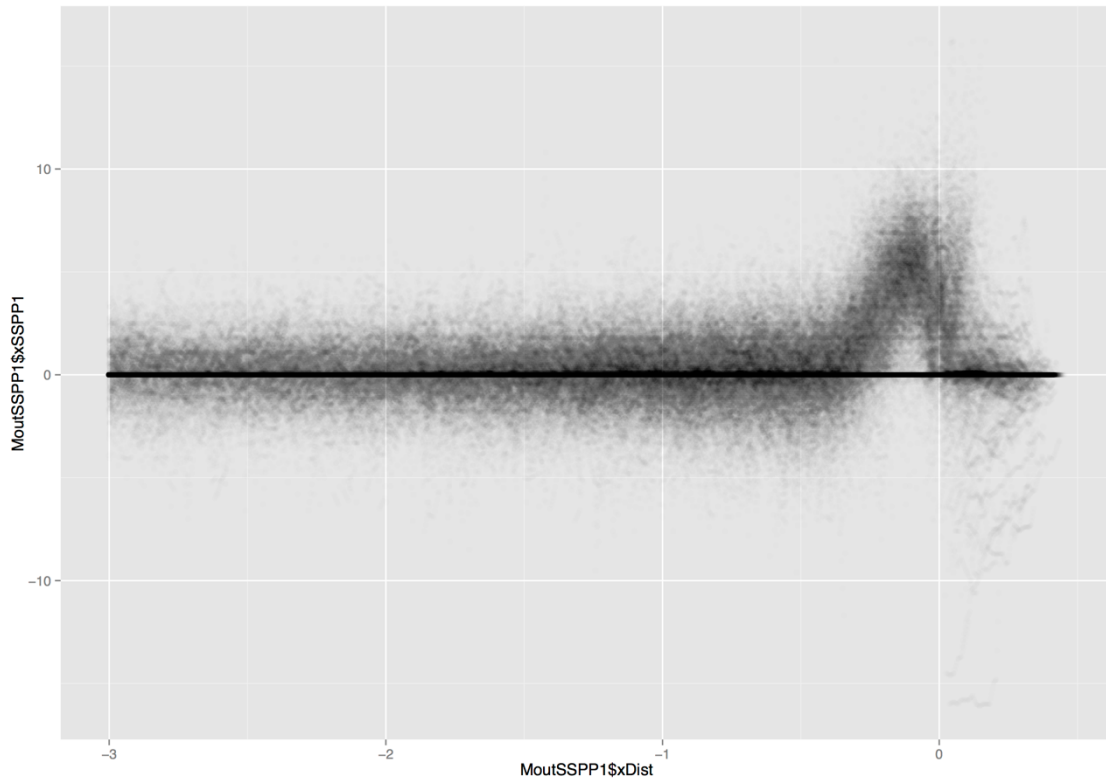


Figure 80. Left seat side-stick pitch (degrees) vs distance from touchdown (NM)

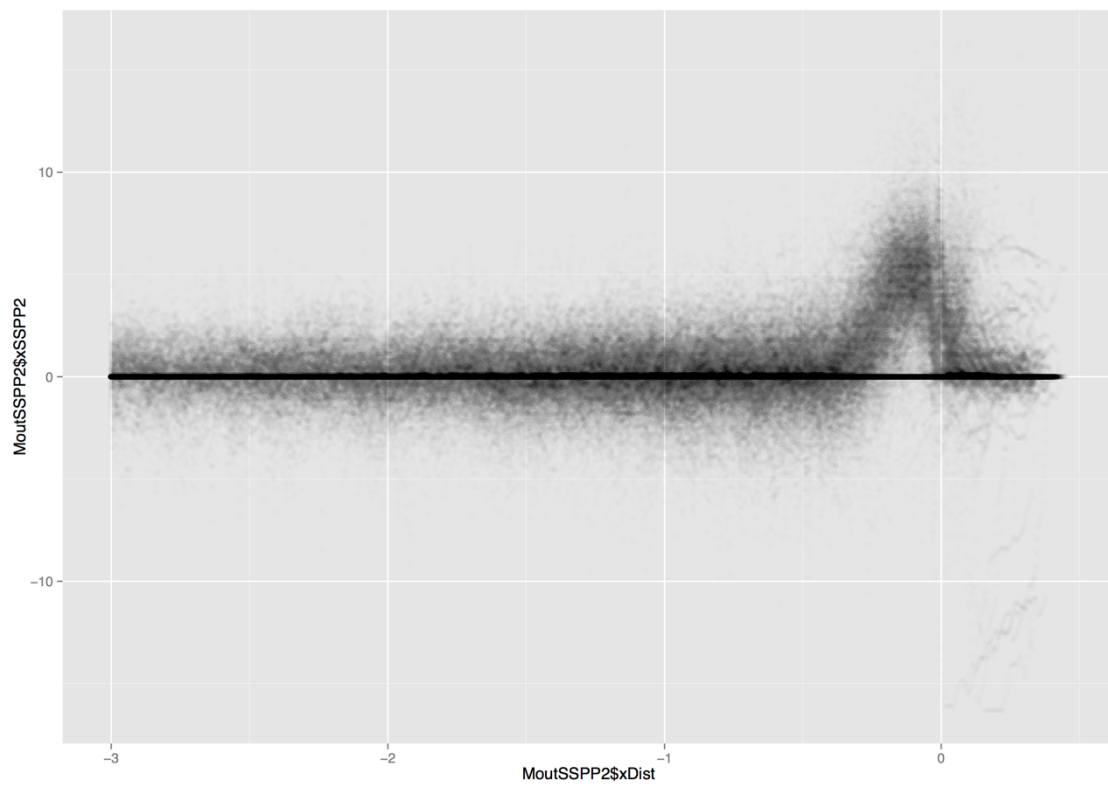


Figure 81. Right seat side-stick pitch (degrees) vs distance from touchdown

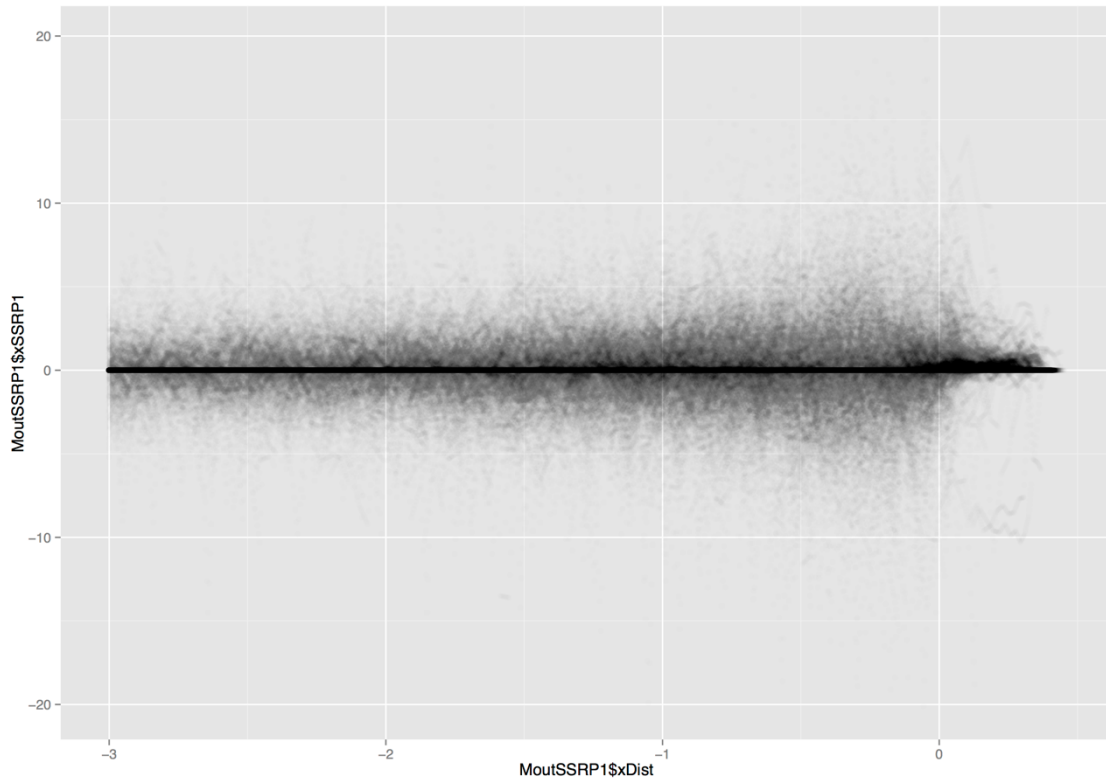


Figure 82. Left seat side-stick roll (degrees) vs distance from touchdown (NM)

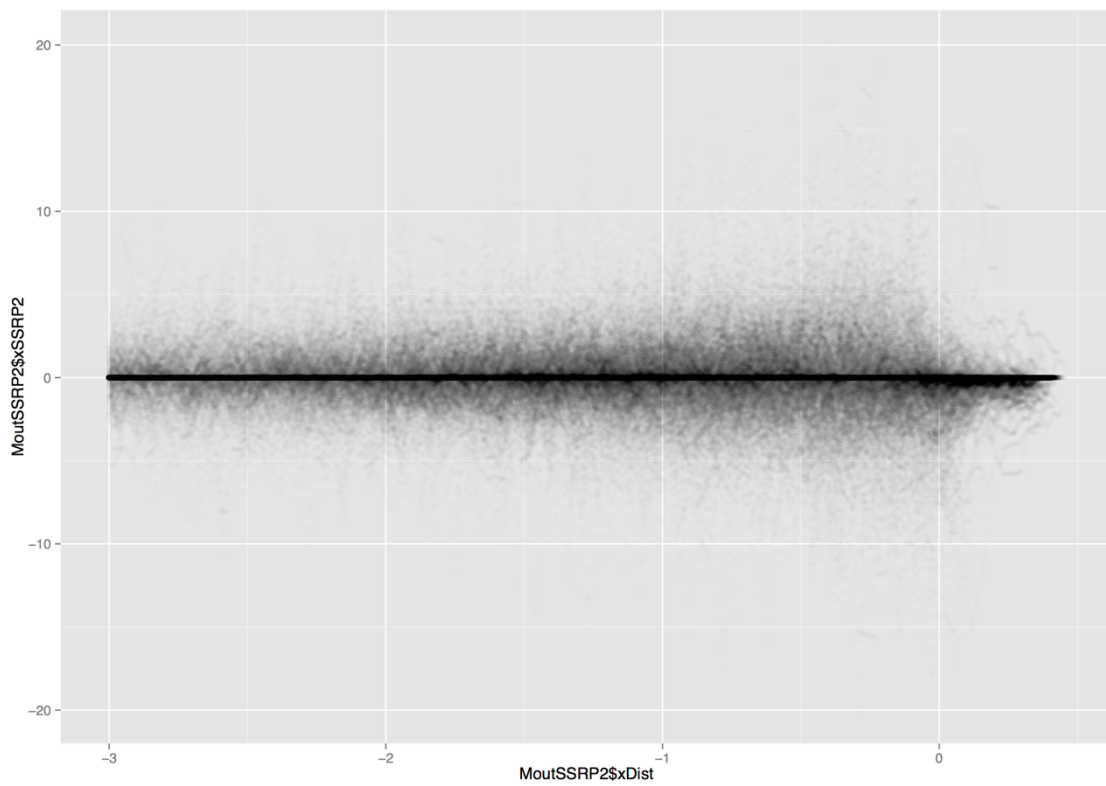


Figure 83. Right seat side-stick roll (degrees) vs distance from touchdown (NM)

Appendix 4 – R code variability feature extraction from flight data

```
1. # variance_v01.R
2. # WVariance project height and speed capture at set minutes from touch down.
3. # v1 based on HeightSpd_V01.R from Workload folder
4.
5.
6. library(zoo) ### for na.locf
7. library(data.table)
8. library(Hmisc) ### for %nin%
9. library(caTools)
10. library(gdata) ### for keep
11. library(progress) ### for progress bar
12.
13. setwd("~/CloudStation/DB_Work/R/Variance/V2")
14.
15. ### Get file path list in folder
16. newRDSfull <- list.files("/Volumes/SlimDrive/Rflights/2007/A319_2007", recursive =
  TRUE, full.names = TRUE)
17.
18. RDSid <- strsplit(newRDSfull, "/")
19. x <- sapply(RDSid[1], length)
20. RDSid <- matrix(unlist(RDSid), ncol = x, byrow = TRUE)
21. RDSid <- RDSid[,x]
22. newRDS <- RDSid
23.
24. numflights <- length(newRDS)
25.
26. flight <- character()
27. errortype <- character()
28.
29. errors <- data.frame(flight = character(0), errortype = character(0), stringsAsFactors
  = FALSE)
30. colnames(errors) <- c("Flight", "Error")
31.
32. ResHS <- data.frame(flight = character(0), arrID = character(0),
33.   h30 = numeric(0), s30 = numeric(0),
34.   h60 = numeric(0), s60 = numeric(0),
35.   h120 = numeric(0), s120 = numeric(0),
36.   h180 = numeric(0), s180 = numeric(0),
37.   h240 = numeric(0), s240 = numeric(0),
38.   h300 = numeric(0), s300 = numeric(0),
39.   h420 = numeric(0), s420 = numeric(0),
40.   h600 = numeric(0), s600 = numeric(0),
41.   stringsAsFactors = FALSE)
42.
43. #ResHS <- ResHS[-1,]
44.
45. # Progress bar
46. pb <- progress_bar$new(total = length(newRDS))
47.
48. for (i in 1:length(newRDS)) {
49.   # Progress
50.   pb$tick()
51.   Sys.sleep(1 / 100)
52.
53.   ## flight ID for output
54.   flight <- newRDS[i]
55.
56.   ## Get arrival airport
57.   arrID <- strsplit(flight, "arr=")
58.   w <- sapply(arrID[1], length)
59.   arrID <- matrix(unlist(arrID), ncol = w, byrow = TRUE)
60.   arrID <- arrID[,2]
61.   arrID <- strsplit(arrID, " dep=")
62.   arrID <- matrix(unlist(arrID), ncol = 2, byrow = TRUE)
63.   arrID <- arrID[,1]
64.
65.   # Read in data
```

```

66. tempflt <- readRDS(newRDSfull[i])
67.
68. # Make data 1Hz by removing CAS NA samples
69. CAS <- as.numeric(tempflt$Computed.Airspeed)
70. blankRows <- is.na(CAS)
71. blankRowsFalse <- which(blankRows == FALSE)
72. tempflt <- tempflt[blankRowsFalse]
73.
74. CAS <- as.numeric(tempflt$Computed.Airspeed) # replace 8Hz version of CAS
75.
76. fltLength <- length(CAS)
77.
78. ## Clean up used numerical params to remove "*". Makes param numeric
79. N11 <- as.numeric(tempflt$N1.Actual.Eng.1)
80. N12 <- as.numeric(tempflt$N1.Actual.Eng.2)
81. GS <- as.numeric(tempflt$Ground.Speed)
82. RALT1 <- as.numeric(tempflt$Radio.Altitude.1)
83. ALT <- as.numeric(tempflt$Altitude..1013.25mB.)
84.
85. # Check if ALT good
86. if (max(ALT, na.rm = TRUE) < 200) {
87.   errortype <- "Poor ALT"
88.   errors[nrow(errors) + 1,] <- c(flight, errortype)
89.   next
90. }
91.
92. FLAPANG <- as.numeric(tempflt$Flap.Angle)
93. FLAPANG <- na.approx(FLAPANG, na.rm = FALSE)
94.
95. ## Clean up used character params to remove "*" and backfill blanks.
96. gear <- gsub("[^[:alnum:]]//'", "", tempflt$Gear.Selector.Down)
97. gear[gear == ""] <- NA ### fills blanks with NA
98. gear <- na.locf(gear, na.rm = FALSE) ### Backfills NA with last value
99.
100. SQ_L <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...LH)
101. SQ_L[SQ_L == ""] <- NA
102. SQ_L <- na.locf(SQ_L, na.rm = FALSE)
103.
104. SQ_N <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...Nose)
105. SQ_N[SQ_N == ""] <- NA
106. SQ_N <- na.locf(SQ_N, na.rm = FALSE)
107.
108. SQ_R <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...RH)
109. SQ_R[SQ_R == ""] <- NA
110. SQ_R <- na.locf(SQ_R, na.rm = FALSE)
111.
112. ## Gear is down.
113. geardn <- (gear == "Down 1")
114.
115. ## Aircraft is on ground
116. onground <- (SQ_L == "Gnd 1" & SQ_N == "Gnd 1") | (SQ_R == "Gnd 1" & SQ_N == "Gnd
117. 1") |
118. (SQ_L == "Gnd 1" & SQ_R == "Gnd 1")
119. ongroundp1 <- c(NA + 1:2, onground[1:(length(onground) - 2)]) # creates vector
    shifting onground 2 secs down
120. ongroundp1 <- as.logical(ongroundp1)
121.
122. LDsearch <- logical(fltLength)
123. LDsearch <- (onground == TRUE) & (ongroundp1 == FALSE) & (geardn == TRUE) & (GS >
    60) & (ALT > -1000) &
124. (ALT < 12000) & (GS < 250) & (tempflt$FMA.Longitudinal.Mode != "FMA: Pitch T/O
    (16)")
125.
126. LandPoint <- which(LDsearch == TRUE)
127. if (length(LandPoint) == 0) {
128.   errortype <- "NO LANDING"
129.   errors[nrow(errors) + 1,] <- c(flight, errortype)
130.
131.   next
132. }
133.
134. LandPoint <- LandPoint[length(LandPoint)]
135.
136. ##### Height Speed code #####
137. ## Height above LD runway

```

```

139. LDelev <- round(mean(ALT[LandPoint + 5], ALT[LandPoint + 6], ALT[LandPoint + 7],
    ALT[LandPoint + 8], ALT[LandPoint + 9],
140.                      ALT[LandPoint + 10]))
141. HaLD <- (ALT - LDelev)
142.
143. h30 <- HaLD[(LandPoint - 30)]
144. s30 <- CAS[(LandPoint - 30)]
145. h60 <- HaLD[(LandPoint - 60)]
146. s60 <- CAS[(LandPoint - 60)]
147. h120 <- HaLD[(LandPoint - 120)]
148. s120 <- CAS[(LandPoint - 120)]
149. h180 <- HaLD[(LandPoint - 180)]
150. s180 <- CAS[(LandPoint - 180)]
151. h240 <- HaLD[(LandPoint - 240)]
152. s240 <- CAS[(LandPoint - 240)]
153. h300 <- HaLD[(LandPoint - 300)]
154. s300 <- CAS[(LandPoint - 300)]
155. h420 <- HaLD[(LandPoint - 420)]
156. s420 <- CAS[(LandPoint - 420)]
157. h600 <- HaLD[(LandPoint - 600)]
158. s600 <- CAS[(LandPoint - 600)]
159.
160. ResHS[nrow(ResHS)+1,] <- c(flight, arrID, h30, s30, h60, s60, h120, s120, h180,
    s180, h240, s240, h300, s300,
161.                          h420, s420, h600, s600)
162.
163. keep(errors, ResHS, newRDS, newRDSfull, pb, sure = TRUE)
164.
165. }

```

Appendix 5 – Calculations of height above airfield (HAAL) and flight path angle (FPA)

Height above airfield

Landing airfield elevation (LDelev) was calculated by taking the mean of recorded pressure altitude (ALT) over the period 10 to 15 seconds after touchdown (LandPoint). The height above airfield (HaLD) was calculated by subtracting LDelev from ALT.

The R code is given below:

```
## Height above LD runway

LDelev <- round(mean(ALT[LandPoint + 10], ALT[LandPoint + 11], ALT[LandPoint + 12],
  ALT[LandPoint + 13], ALT[LandPoint + 14],
  ALT[LandPoint + 15]))

HaLD <- (ALT - LDelev)
```

Flight path angle

Flight path angle (instantFPAdeg) was calculated by taking the inverse tan of the change in HaLD divided by distance covered. It was then converted from radians to degrees. The R code is given below:

```
##Flight path angle

HaLDm1 <- c(NA, HaLD[1:(length(HaLD) - 1)])

instantFPAdeg <- (atan((HaLD - HaLDm1) / (GS * 1.68781))) * 180 / pi
```


Appendix 6 – R code for workload extraction from flight data

```
1. # Workload_v10.R
2. # v1 based on AppVar_v4.R (Approach variance work)
3. # v2 200ft to 40kt ground speed only (v1 took 10.8s per flight, now 3.6s)
4. # v3 added & (onground == FALSE) to landlog due to occ descent on TO roll
5. #   changed onground to onground <- (SQ_L == "Gnd 1" & SQ_N == "Gnd 1") |
6. #     (SQ_R == "Gnd 1" & SQ_N == "Gnd 1") |(SQ_L == "Gnd 1" & SQ_R == "Gnd 1")
7. #   due to squat switches sticking
8. #   added (tempflt$FMA.Longitudinal.Mode != "FMA: Pitch T/O (16)")
9. #   to landlog due to occ descent on TO roll
10. # v4 Extract arrival airport and create new column in output. Added landFLAPANG.
11. # v5 Changed startLand to work back from touchdown to 200ft HAAL because go arounds
    were
12. #   causing long landing phases.
13. # v6 Trying last 90 seconds of approach (to 10 sec before td. Added many more
    work/action parameters.
14. # v7 Traying 180sec. Changed progress bar.
15. # v8 commented out duplicate selected flight path
16. # v9 removed commenetd out params in prep for commenting out params in order to
    identify
17. #   bi-modal distribution causes.
18. # v10 tidied up params. Removed Baro params which alternated. Changed flap position to
    flap lever.
19. #   made Landgear 180s
20.
21. library(zoo) ### for na.locf
22. library(data.table)
23. library(Hmisc) ### for %nin%
24. library(caTools)
25. library(gdata) ### for keep
26. library(progress) ### for progress bar
27.
28. setwd("/Users/DJB/CloudStation/DB_Work/PhD/Year 3/R/Bi_mode_investigation")
29.
30. ### Get file path list in folder
31. newRDSfull <- list.files("/Volumes/DB/Rflights/2007/A319_2007", recursive = TRUE,
    full.names = TRUE)
32.
33. RDSid <- strsplit(newRDSfull, "/")
34. x <- sapply(RDSid[1], length)
35. RDSid <- matrix(unlist(RDSid), ncol = x, byrow = TRUE)
36. RDSid <- RDSid[,x]
37. newRDS <- RDSid
38.
39. flight <- character()
40. errortype <- character()
41.
42. errors <- data.frame(flight = character(0), errortype = character(0), stringsAsFactors
    = FALSE)
43. colnames(errors) <- c("Flight", "Error")
44.
45. cntHead <- paste("x", 0:20, sep = "")
46. ResWork <- as.data.frame(matrix(rep(0, 3 + length(cntHead)), nrow = 1))
47. names(ResWork) <- c("Flight", "Length", "Arrive", cntHead)
48. ResWork <- ResWork[-1,]
49.
50. # Progress bar
51. pb <- progress_bar$new(total = length(newRDS))
52.
53. for (i in 1:length(newRDS)) {
54.   # Progress
55.   pb$tick()
56.   Sys.sleep(1 / 100)
57.
58.   ## flight ID for output
59.   flight <- newRDS[i]
60.
61.   ## Get arrival airport
```

```

62. arrID <- strsplit(flight, "arr=")
63. w <- sapply(arrID[1], length)
64. arrID <- matrix(unlist(arrID), ncol = w, byrow = TRUE)
65. arrID <- arrID[,2]
66. arrID <- strsplit(arrID, " dep=")
67. arrID <- matrix(unlist(arrID), ncol = 2, byrow = TRUE)
68. arrID <- arrID[,1]
69.
70. # Read in data
71. tempflt <- readRDS(newRDSfull[i])
72.
73. # Make data 1Hz by removing CAS NA samples
74. CAS <- as.numeric(tempflt$Computed.Airspeed)
75. blankRows <- is.na(CAS)
76. blankRowsFalse <- which(blankRows == FALSE)
77. tempflt <- tempflt[blankRowsFalse]
78.
79. CAS <- as.numeric(tempflt$Computed.Airspeed) # replace 8Hz version of CAS
80.
81. fltLength <- length(CAS)
82.
83. ## Clean up used numerical params to remove "*". Makes param numeric
84. N11 <- as.numeric(tempflt$N1.Actual.Eng.1)
85. N12 <- as.numeric(tempflt$N1.Actual.Eng.2)
86. GS <- as.numeric(tempflt$Ground.Speed)
87. RALT1 <- as.numeric(tempflt$Radio.Altitude.1)
88. ALT <- as.numeric(tempflt$Altitude..1013.25mB.)
89.
90. # Check if ALT good
91. if (max(ALT, na.rm = TRUE) < 200) {
92.   errortype <- "Poor ALT"
93.   errors[nrow(errors) + 1,] <- c(flight, errortype)
94.   next
95. }
96.
97. FLAPANG <- as.numeric(tempflt$Flap.Angle)
98. FLAPANG <- na.approx(FLAPANG, na.rm = FALSE)
99.
100. ## Clean up used character params to remove "*" and backfill blanks.
101. gear <- gsub("[^[:alnum:]]+", "", tempflt$Gear.Selector.Down)
102. gear[gear == ""] <- NA ### fills blanks with NA
103. gear <- na.locf(gear, na.rm = FALSE) ### Backfills NA with last value
104.
105. SQ_L <- gsub("[^[:alnum:]]+", "", tempflt$Land.Gear.Squat.Switch...LH)
106. SQ_L[SQ_L == ""] <- NA
107. SQ_L <- na.locf(SQ_L, na.rm = FALSE)
108.
109. SQ_N <- gsub("[^[:alnum:]]+", "", tempflt$Land.Gear.Squat.Switch...Nose)
110. SQ_N[SQ_N == ""] <- NA
111. SQ_N <- na.locf(SQ_N, na.rm = FALSE)
112.
113. SQ_R <- gsub("[^[:alnum:]]+", "", tempflt$Land.Gear.Squat.Switch...RH)
114. SQ_R[SQ_R == ""] <- NA
115. SQ_R <- na.locf(SQ_R, na.rm = FALSE)
116.
117. ## Gear is down.
118. geardn <- (gear == "Down 1")
119.
120. ## Aircraft is on ground
121. onground <- (SQ_L == "Gnd 1" & SQ_N == "Gnd 1") | (SQ_R == "Gnd 1" & SQ_N == "Gnd
122. 1") |
123.   (SQ_L == "Gnd 1" & SQ_R == "Gnd 1")
124. ongroundp1 <- c(NA + 1:2, onground[1:(length(onground) - 2)]) # creates vector
125. ongroundp1 <- as.logical(ongroundp1)
126.
127. LDsearch <- logical(fltLength)
128. LDsearch <- (onground == TRUE) & (ongroundp1 == FALSE) & (geardn == TRUE) & (GS >
129. 60) & (ALT > -1000) &
130.   (ALT < 12000) & (GS < 250) & (tempflt$FMA.Longitudinal.Mode != "FMA: Pitch T/O
131. (16)")
132. LandPoint <- which(LDsearch == TRUE)
133. if (length(LandPoint) == 0) {
134.   errortype <- "NO LANDING"

```

```

135.     errors[nrow(errors) + 1,] <- c(flight, errortype)
136.
137.     next
138. }
139.
140.     LandPoint <- LandPoint[length(LandPoint)]
141.
142.     ##### Workload specific code #####
143.     landData <- tempflt[(LandPoint - 189):(LandPoint - 10)] # creates 180s window for
data collection
144.     landLength <- length(landData$Computed.Airspeed)
145.
146.     ## Create flap angle param for landing data. Was 0.5Hz
147.     landFLAPANG <- as.numeric(landData$Flap.Angle)
148.     landFLAPANG <- na.approx(landFLAPANG, na.rm = FALSE)
149.     ifelse(is.na(landFLAPANG[1]), landFLAPANG[1] <- landFLAPANG[2], landFLAPANG)
150.     ifelse(is.na(landFLAPANG[90]), landFLAPANG[90] <- landFLAPANG[89], landFLAPANG)
151.
152.     ## Create Eng Ant Ice param for landing data. Was 0.5Hz
153.     landAntIce1 <- gsub("[^[:alnum:]]//'", "", landData$Eng.1.Anti.Ice.P.B.On)
154.     landAntIce1[landAntIce1 == ""] <- NA
155.     landAntIce1 <- na.locf(landAntIce1, na.rm = FALSE)
156.     ifelse(is.na(landAntIce1[1]), landAntIce1[1] <- landAntIce1[2], landAntIce1)
157.     ifelse(is.na(landAntIce1[90]), landAntIce1[90] <- landAntIce1[89], landAntIce1)
158.
159.     ## Make landing gear landData param
160.     landGear <- gear[(LandPoint - 189):(LandPoint - 10)]
161.
162.     ## EFIS mode Capt
163.     EFISmodeSel <- gsub("[^[:alnum:]]//'", "", landData$EFIS.Mode.Sel...Capt)
164.     EFISmodeSel[EFISmodeSel == ""] <- NA
165.     EFISmodeSel <- na.locf(EFISmodeSel, na.rm = FALSE)
166.     ifelse(is.na(EFISmodeSel[2]), EFISmodeSel[2] <- EFISmodeSel[3], EFISmodeSel)
167.     ifelse(is.na(EFISmodeSel[1]), EFISmodeSel[1] <- EFISmodeSel[2], EFISmodeSel)
168.     ifelse(is.na(EFISmodeSel[89]), EFISmodeSel[89] <- EFISmodeSel[88], EFISmodeSel)
169.     ifelse(is.na(EFISmodeSel[90]), EFISmodeSel[90] <- EFISmodeSel[89], EFISmodeSel)
170.
171.     ## EFIS ranges
172.     EFISrangeCapt <- gsub("[^[:alnum:]]//'", "",
landData$EFIS.Scale.Sel..Range....Capt)
173.     EFISrangeCapt[EFISrangeCapt == ""] <- NA
174.     EFISrangeCapt <- na.locf(EFISrangeCapt, na.rm = FALSE)
175.     ifelse(is.na(EFISrangeCapt[3]), EFISrangeCapt[3] <- EFISrangeCapt[4], EFISrangeCapt)
176.     ifelse(is.na(EFISrangeCapt[2]), EFISrangeCapt[2] <- EFISrangeCapt[3], EFISrangeCapt)
177.     ifelse(is.na(EFISrangeCapt[1]), EFISrangeCapt[1] <- EFISrangeCapt[2], EFISrangeCapt)
178.     ifelse(is.na(EFISrangeCapt[88]), EFISrangeCapt[88] <- EFISrangeCapt[87],
EFISrangeCapt)
179.     ifelse(is.na(EFISrangeCapt[89]), EFISrangeCapt[89] <- EFISrangeCapt[88],
EFISrangeCapt)
180.     ifelse(is.na(EFISrangeCapt[90]), EFISrangeCapt[90] <- EFISrangeCapt[89],
EFISrangeCapt)
181.
182.     EFISrangeFO <- gsub("[^[:alnum:]]//'", "", landData$EFIS.Scale.Sel..Range....F.O)
183.     EFISrangeFO[EFISrangeFO == ""] <- NA
184.     EFISrangeFO <- na.locf(EFISrangeFO, na.rm = FALSE)
185.     ifelse(is.na(EFISrangeFO[3]), EFISrangeFO[3] <- EFISrangeFO[4], EFISrangeFO)
186.     ifelse(is.na(EFISrangeFO[2]), EFISrangeFO[2] <- EFISrangeFO[3], EFISrangeFO)
187.     ifelse(is.na(EFISrangeFO[1]), EFISrangeFO[1] <- EFISrangeFO[2], EFISrangeFO)
188.     ifelse(is.na(EFISrangeFO[88]), EFISrangeFO[88] <- EFISrangeFO[87], EFISrangeFO)
189.     ifelse(is.na(EFISrangeFO[89]), EFISrangeFO[89] <- EFISrangeFO[88], EFISrangeFO)
190.     ifelse(is.na(EFISrangeFO[90]), EFISrangeFO[90] <- EFISrangeFO[89], EFISrangeFO)
191.
192.     ## Page selected
193.     pageSel <- gsub("[^[:alnum:]]//'", "", landData$Page.Selected)
194.     pageSel[pageSel == ""] <- NA
195.     pageSel <- na.locf(pageSel, na.rm = FALSE)
196.     ifelse(is.na(pageSel[3]), pageSel[3] <- pageSel[4], pageSel)
197.     ifelse(is.na(pageSel[2]), pageSel[2] <- pageSel[3], pageSel)
198.     ifelse(is.na(pageSel[1]), pageSel[1] <- pageSel[2], pageSel)
199.     ifelse(is.na(pageSel[88]), pageSel[88] <- pageSel[87], pageSel)
200.     ifelse(is.na(pageSel[89]), pageSel[89] <- pageSel[88], pageSel)
201.     ifelse(is.na(pageSel[90]), pageSel[90] <- pageSel[89], pageSel)
202.
203.     ## Baro settings
204.     BaroCapt <- gsub("[^[:alnum:]]//'", "", landData$Selected.Baro.Setting.Capt)
205.     BaroCapt[BaroCapt == ""] <- NA
206.     BaroCapt <- na.locf(BaroCapt, na.rm = FALSE)

```

```

207. ifelse(is.na(BaroCapt[3]), BaroCapt[3] <- BaroCapt[4], BaroCapt)
208. ifelse(is.na(BaroCapt[2]), BaroCapt[2] <- BaroCapt[3], BaroCapt)
209. ifelse(is.na(BaroCapt[1]), BaroCapt[1] <- BaroCapt[2], BaroCapt)
210. ifelse(is.na(BaroCapt[88]), BaroCapt[88] <- BaroCapt[87], BaroCapt)
211. ifelse(is.na(BaroCapt[89]), BaroCapt[89] <- BaroCapt[88], BaroCapt)
212. ifelse(is.na(BaroCapt[90]), BaroCapt[90] <- BaroCapt[89], BaroCapt)
213.
214. BaroFO <- gsub("[^[:alnum:]]//'", "", landData$Selected.Baro.Setting.F.O)
215. BaroFO[BaroFO == ""] <- NA
216. BaroFO <- na.locf(BaroFO, na.rm = FALSE)
217. ifelse(is.na(BaroFO[3]), BaroFO[3] <- BaroFO[4], BaroFO)
218. ifelse(is.na(BaroFO[2]), BaroFO[2] <- BaroFO[3], BaroFO)
219. ifelse(is.na(BaroFO[1]), BaroFO[1] <- BaroFO[2], BaroFO)
220. ifelse(is.na(BaroFO[88]), BaroFO[88] <- BaroFO[87], BaroFO)
221. ifelse(is.na(BaroFO[89]), BaroFO[89] <- BaroFO[88], BaroFO)
222. ifelse(is.na(BaroFO[90]), BaroFO[90] <- BaroFO[89], BaroFO)
223.
224. ## Selected decision
225. SelDec <- gsub("[^[:alnum:]]//'", "", landData$Selected.Decision.Height)
226. SelDec[SelDec == ""] <- NA
227. SelDec <- na.locf(SelDec, na.rm = FALSE)
228. ifelse(is.na(SelDec[3]), SelDec[3] <- SelDec[4], SelDec)
229. ifelse(is.na(SelDec[2]), SelDec[2] <- SelDec[3], SelDec)
230. ifelse(is.na(SelDec[1]), SelDec[1] <- SelDec[2], SelDec)
231. ifelse(is.na(SelDec[88]), SelDec[88] <- SelDec[87], SelDec)
232. ifelse(is.na(SelDec[89]), SelDec[89] <- SelDec[88], SelDec)
233. ifelse(is.na(SelDec[90]), SelDec[90] <- SelDec[89], SelDec)
234.
235. ## Speed brake command
236. SpdBrkCmd <- gsub("[^[:alnum:]]//'", "", landData$Speed.Brake.Command)
237. SpdBrkCmd[SpdBrkCmd == ""] <- NA
238. SpdBrkCmd <- na.locf(SpdBrkCmd, na.rm = FALSE)
239. ifelse(is.na(SpdBrkCmd[3]), SpdBrkCmd[3] <- SpdBrkCmd[4], SpdBrkCmd)
240. ifelse(is.na(SpdBrkCmd[2]), SpdBrkCmd[2] <- SpdBrkCmd[3], SpdBrkCmd)
241. ifelse(is.na(SpdBrkCmd[1]), SpdBrkCmd[1] <- SpdBrkCmd[2], SpdBrkCmd)
242. ifelse(is.na(SpdBrkCmd[88]), SpdBrkCmd[88] <- SpdBrkCmd[87], SpdBrkCmd)
243. ifelse(is.na(SpdBrkCmd[89]), SpdBrkCmd[89] <- SpdBrkCmd[88], SpdBrkCmd)
244. ifelse(is.na(SpdBrkCmd[90]), SpdBrkCmd[90] <- SpdBrkCmd[89], SpdBrkCmd)
245.
246.
247. work <- numeric(landLength)
248.
249. for(j in 1:length(work)) {
250.   ifelse ((landData$Flap.Lever.Pos[j+1] != landData$Flap.Lever.Pos[j]), work[j] <-
work[j] +1, work)
251.   ifelse ((landData$VHF1.Keyed[j] == "EMIT (0)"), work[j] <- work[j] +1, work)
252.   ifelse ((landData$VHF2.Keyed[j] == "EMIT (0)"), work[j] <- work[j] +1, work)
253.   ifelse ((landData$VHF3.Keyed[j] == "EMIT (0)"), work[j] <- work[j] +1, work)
254.   ifelse ((landData$A.P.1.Engaged[j+1] != landData$A.P.1.Engaged[j]), work[j] <-
work[j] +1, work)
255.   ifelse ((landData$A.P.2.Engaged[j+1] != landData$A.P.2.Engaged[j]), work[j] <-
work[j] +1, work)
256.   ifelse ((landAntIcel[j+1] != landAntIcel[j]), work[j] <- work[j] +1, work)
257.   ifelse ((landData$Selected.Altitude[j+1] != landData$Selected.Altitude[j]),
work[j] <- work[j] +1, work)
258.   ifelse ((landData$ATS.Engaged[j+1] != landData$ATS.Engaged[j]), work[j] <- work[j]
+1, work)
259.   ifelse ((landData$COCKPIT.CALL.SYS.2[j]) == "Yes (0)", work[j] <- work[j] +1,
work)
260.   ifelse ((landData$F.D.1.Engaged[j+1] != landData$F.D.1.Engaged[j]), work[j] <-
work[j] +1, work)
261.   ifelse ((landData$F.D.2.Engaged[j+1] != landData$F.D.2.Engaged[j]), work[j] <-
work[j] +1, work)
262.   ifelse ((landData$FASTEN.SEAT.BELTS.LAMP.ON[j+1] !=
landData$FASTEN.SEAT.BELTS.LAMP.ON[j]), work[j] <- work[j] +1, work)
263.   ifelse ((landGear[j+1] != landGear[j]), work[j] <- work[j] +1, work)
264.   ifelse ((landData$Selected Heading[j+1] != landData$Selected Heading[j]), work[j]
<- work[j] +1, work)
265.   ifelse ((landData$Deceleration.Low.Armed[j+1] !=
landData$Deceleration.Low.Armed[j]), work[j] <- work[j] +1, work)
266.   ifelse ((landData$Deceleration.Medium.Armed[j+1] !=
landData$Deceleration.Medium.Armed[j]), work[j] <- work[j] +1, work)
267.
268.   ifelse ((EFISmodeSel[j+1] != EFISmodeSel[j]), work[j] <- work[j] +1, work)
269.   ifelse ((EFISrangeCapt[j+1] != EFISrangeCapt[j]), work[j] <- work[j] +1, work)
270.   ifelse ((EFISrangeFO[j+1] != EFISrangeFO[j]), work[j] <- work[j] +1, work)

```

```

271.   ifelse ((landData$Ground.Spoiler.Armed[j+1] != landData$Ground.Spoiler.Armed[j]),
work[j] <- work[j] +1, work)
272.   ifelse ((pageSel[j+1] != pageSel[j]), work[j] <- work[j] +1, work)
273.   ifelse ((landData$Selected.Flight.Path[j+1] != landData$Selected.Flight.Path[j]),
work[j] <- work[j] +1, work)
274.   ifelse ((SelDec[j+1] != SelDec[j]), work[j] <- work[j] +1, work)
275.   ifelse ((landData$Selected.Track[j+1] != landData$Selected.Track[j]), work[j] <-
work[j] +1, work)
276.   ifelse ((landData$Selected.Vertical.Speed[j+1] !=
landData$Selected.Vertical.Speed[j]), work[j] <- work[j] +1, work)
277.   ifelse ((SpdBrkCmd[j+1] != SpdBrkCmd[j]), work[j] <- work[j] +1, work)
278.   ifelse ((landData$TRK...FPA.SELECT[j+1] != landData$TRK...FPA.SELECT[j]), work[j]
<- work[j] +1, work)
279.
280. }
281.
282. newWork <- table(work)
283. newWork <- as.data.frame(newWork)
284. newWork <- t(newWork)
285. newWork <- as.numeric(newWork[2,])
286. length(newWork) <- 21
287.
288. ResWork[nrow(ResWork)+1,] <- c(flight, landLength, arrID, newWork)
289.
290. keep(errors, ResWork, newRDS, newRDSfull, pb, sure = TRUE)
291.
292. }

```

Appendix 7 – R code for runway excursion feature extraction from flight data

```

1. # RwyExc_PhD_v08.R
2. # v1
3. # v2 added lower weight check > 20k
4. # v3 Changed Brkpedt to capture Inf and make NA. Added poor date error
5. # v4 Fixed GS and LOC error. Fixed GRWEIGHT error
6. # v5 refined precTD to look for earlier of NMLA or Squat
7. # v6 corrected RALT150 and RALT50 error. They were counting back the wrong number
8. #   from precTD. Needed to add RALT50 <- 240 - RALT50. This means Spddelta50 was
   #   wrong before
9. #   and Glidslope 150
10. # v7 added RALT1 < 50 to LDsearch. Made onground any two squat switches to avoid rare
   #   stuck squat.
11. # v8 added new error "Landing c/o" for when flight ends during landing roll
12.
13.
14. codeVer <- "v8"
15.
16. library(zoo) ### for na.locf
17. library(data.table)
18. library(Hmisc) ### for %nin%
19. library(caTools)
20. library(weatherData)
21.
22. setwd("~/CloudStation/DB_Work/R/RWYEXC")
23.
24. ## Load already processed flight IDs for duplicate check
25. VflightID <- readRDS("VflightID.rds")
26.
27. ### Get file path list in folder
28. newRDSfull <- list.files("/Users/DJB/Desktop/test", recursive = TRUE, full.names =
   TRUE)
29.
30. RDSid <- strsplit(newRDSfull, "/")
31. x <- sapply(RDSid[1], length)
32. RDSid <- matrix(unlist(RDSid), ncol = x, byrow = TRUE)
33. RDSid <- RDSid[,x]
34. newRDSfull <- newRDSfull[which(RDSid %nin% VflightID)]
35. newRDS <- RDSid[RDSid %nin% VflightID]
36.
37. ## Create empty output tables
38. flight <- character()
39. errortype <- character()
40.
41. ifelse(file.exists("RexOut.rds") == TRUE, RexOut <- readRDS("RexOut.rds"),
42.   RexOut <- data.frame(flight = character(0), Fleet = character(0), TDtimeChar =
   character(0),
43.     arrID = character(0), TDweight = numeric(0),
44.     rwyHDG = numeric(0), recWindDir = numeric(0), recWindSpd =
   numeric(0),
45.     rLandHDwind = numeric(0), rLandXwind = numeric(0),
   metWindD = numeric(0),
46.     metWindS = numeric(0), methDwind = numeric(0),
47.     metXwind = numeric(0), methDgust = numeric(0), metXgust =
   numeric(0),
48.     metVizkm = numeric(0), runWet = logical(0), asymTHR =
   numeric(0), spdDelta50 = numeric(0),
49.     maxG = numeric(0), HDGdev = numeric(0), RUDmax =
   numeric(0), RUDmin = numeric(0),
50.     RUDmed = numeric(0), RUDmean = numeric(0), RUDsd =
   numeric(0),
51.     NWSmax = numeric(0), NWSmin = numeric(0), NWSmed =
   numeric(0),
52.     NWSmean = numeric(0), NWSsd = numeric(0), GSlope50 =
   numeric(0), GSlope150 = numeric(0),
53.     LOCmax = numeric(0), LOCmin = numeric(0), LOCmed =
   numeric(0),
54.     LOCmean = numeric(0), LOCsd = numeric(0),
55.     LATGmax = numeric(0), LATGmin = numeric(0), LATGmed =
   numeric(0),

```

```

56.         LATGmean = numeric(0), LATGsd = numeric(0), LONGmax =
   numeric(0), LONGmin = numeric(0),
57.         LONGmed = numeric(0), LONGmean = numeric(0), LONGsd =
   numeric(0), LONGp3 = numeric(0),
58.         LONGp5 = numeric(0), LONGp7 = numeric(0), LONGp10 =
   numeric(0), SPOILt = numeric(0),
59.         REVt = numeric(0), maxN1L = character(0), maxN1R =
   character(0), autoBRK = character(0), BRKPEDt = numeric(0),
60.         LBrPedSum = numeric(0), LBrPedMean = numeric(0),
61.         LBrPedMax = numeric(0),
62.         RBrPedSum = numeric(0), RBrPedMean = numeric(0), RBrPedMax
   = numeric(0), TDidle = logical(0),
63.         pitchTD = numeric(0), rollTD = numeric(0),
64.         TDGspd = numeric(0), TDCAS = numeric(0), TDflap =
   numeric(0), codeVer = character(0),
65.         stringsAsFactors = FALSE))
66.
67. errors <- data.frame(flight = character(0), errortype = character(0), stringsAsFactors
   = FALSE)
68. colnames(errors) <- c("Flight", "Error")
69.
70. for (i in 1:length(newRDS)) {
71.   # Progress
72.   cat(paste(".", i, "."))
73.
74.   ## flight ID for output
75.   flight <- newRDS[i]
76.
77.   ## Get arrival airport
78.   arrID <- strsplit(flight, "arr=")
79.   w <- sapply(arrID[1], length)
80.   arrID <- matrix(unlist(arrID), ncol = w, byrow = TRUE)
81.   arrID <- arrID[,2]
82.   arrID <- strsplit(arrID, " dep=")
83.   arrID <- matrix(unlist(arrID), ncol = 2, byrow = TRUE)
84.   arrID <- arrID[,1]
85.
86.   # Get arrival date
87.   flightdate <- strsplit(flight, "dt=")
88.   v <- sapply(flightdate[1], length)
89.   flightdate <- matrix(unlist(flightdate), ncol = v, byrow = TRUE)
90.   flightdate <- flightdate[,2]
91.   flightdate <- strsplit(flightdate, ".csv")
92.   v <- sapply(flightdate[1], length)
93.   flightdate <- matrix(unlist(flightdate), ncol = v, byrow = TRUE)
94.   flightdate <- flightdate[,1]
95.   flightdate <- as.Date(flightdate, "%Y%m%d")
96.
97.   if (is.na(flightdate)) {
98.     errortype <- "Poor date"
99.     errors[nrow(errors) + 1,] <- c(flight, errortype)
100.    next
101.  }
102.
103.  ## Check if file has been seen before
104.  # dupe <- is.element(flight, VflightID)
105.  # if (dupe == TRUE) {
106.  #   errortype <- "Duplicate"
107.  #   errors[nrow(errors) + 1,] <- c(flight, errortype)
108.  #   next}
109.
110.  VflightID <- c(VflightID, flight) # add this flight to Master flight list
111.
112.  # Read in data
113.  tempflt <- readRDS(newRDSfull[i])
114.
115.  ## Clean up used numerical params to remove "". Makes param numeric, fills blanks
   with NA then backfills
116.  ## NA with last value
117.  CAS <- as.numeric(tempflt$Computed.Airspeed)
118.  CAS <- na.approx(CAS, na.rm = FALSE)
119.  N11 <- as.numeric(tempflt$N1.Actual.Eng.1)
120.  N11 <- na.approx(N11, na.rm = FALSE)
121.  N12 <- as.numeric(tempflt$N1.Actual.Eng.2)
122.  N12 <- na.approx(N12, na.rm = FALSE)
123.  GS <- as.numeric(tempflt$Ground.Speed)
124.  GS <- na.approx(GS, na.rm = FALSE)

```

```

125. RALT1 <- as.numeric(tempflt$Radio.Altitude.1)
126. RALT1 <- na.approx(RALT1, na.rm = FALSE)
127. ALT <- as.numeric(tempflt$Altitude..1013.25mB.)
128. ALT <- na.approx(ALT, na.rm = FALSE)
129. # Check if ALT good
130. if (max(ALT, na.rm = TRUE) < 200) {
131.   errortype <- "Poor ALT"
132.   errors[nrow(errors) + 1,] <- c(flight, errortype)
133.   next
134. }
135. FLAPANG <- as.numeric(tempflt$Flap.Angle)
136. FLAPANG <- na.approx(FLAPANG, na.rm = FALSE)
137. PITCH <- as.numeric(tempflt$Pitch.Angle)
138. PITCH <- na.approx(PITCH, na.rm = FALSE)
139. ROLL <- as.numeric(tempflt$Roll.Angle)
140. ROLL <- na.approx(ROLL, na.rm = FALSE)
141.
142. APPSPD <- as.numeric(tempflt$Approach.Speed.Target)
143. APPSPD <- na.approx(APPSPD, na.rm = FALSE)
144. LATACC <- as.numeric(tempflt$Lateral.Acceleration)
145. LATACC <- na.approx(LATACC, na.rm = FALSE)
146. LONACC <- as.numeric(tempflt$Longitudinal.Acceleration)
147. LONACC <- na.approx(LONACC, na.rm = FALSE)
148. MHDG <- as.numeric(tempflt$Magnetic Heading)
149. MHDG <- na.approx(MHDG, na.rm = FALSE)
150. THDG <- as.numeric(tempflt$True Heading)
151. THDG <- na.approx(THDG, na.rm = FALSE)
152. RUDPOS <- as.numeric(tempflt$Rudder.Position)
153. RUDPOS <- na.approx(RUDPOS, na.rm = FALSE)
154. SSPP1 <- as.numeric(tempflt$Sidestick.Pos.Capt.Pitch)
155. SSPP1 <- na.approx(SSPP1, na.rm = FALSE)
156. SSRP1 <- as.numeric(tempflt$Sidestick.Pos.Capt.Roll)
157. SSRP1 <- na.approx(SSRP1, na.rm = FALSE)
158. SSPP2 <- as.numeric(tempflt$Sidestick.Pos.F.O.Pitch)
159. SSPP2 <- na.approx(SSPP2, na.rm = FALSE)
160. SSRP2 <- as.numeric(tempflt$Sidestick.Pos.F.O.Roll)
161. SSRP2 <- na.approx(SSRP2, na.rm = FALSE)
162.
163.
164.
165. NWS <- as.numeric(tempflt$Nose.Wheel.Steering.Angle)
166. NWS <- na.approx(NWS, na.rm = FALSE)
167. BRKPED_L <- as.numeric(tempflt$Brake.Pedal.Angle.L)
168. BRKPED_L <- na.approx(BRKPED_L, na.rm = FALSE)
169. BRKPED_R <- as.numeric(tempflt$Brake.Pedal.Angle.R)
170. BRKPED_R <- na.approx(BRKPED_R, na.rm = FALSE)
171. THRANG_1 <- as.numeric(tempflt$Throttle.Angle.Eng.1)
172. THRANG_1 <- na.approx(THRANG_1, na.rm = FALSE)
173. THRANG_2 <- as.numeric(tempflt$Throttle.Angle.Eng.2)
174. THRANG_2 <- na.approx(THRANG_2, na.rm = FALSE)
175. WINDDIR <- as.numeric(tempflt$Wind.Direction.True)
176. WINDDIR <- na.locf(WINDDIR, na.rm = FALSE)
177. WINDSPD <- as.numeric(tempflt$Wind.Speed)
178. WINDSPD <- na.locf(WINDSPD, na.rm = FALSE)
179. GRWEIGHT <- as.numeric(tempflt$Actual.Gross.Weight)
180. GRWEIGHT <- na.locf(GRWEIGHT, na.rm = FALSE)
181.
182. ## Normal Acc: at 8Hz but contains *. Need to remove *
183. NMLA <- gsub("[/*]", "", tempflt$Normal.Acceleration)
184. NMLA <- as.numeric(NMLA)
185.
186. ## Clean up used character params to remove "*" and backfill blanks.
187. gear <- gsub("[^[:alnum:]]//'", "", tempflt$Gear.Selector.Down)
188. gear[gear == ""] <- NA ### fills blanks with NA
189. gear <- na.locf(gear, na.rm = FALSE) ### Backfills NA with last value
190.
191. SQ_L <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...LH)
192. SQ_L[SQ_L == ""] <- NA
193. SQ_L <- na.locf(SQ_L, na.rm = FALSE)
194.
195. SQ_N <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...Nose)
196. SQ_N[SQ_N == ""] <- NA
197. SQ_N <- na.locf(SQ_N, na.rm = FALSE)
198.
199. SQ_R <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...RH)
200. SQ_R[SQ_R == ""] <- NA
201. SQ_R <- na.locf(SQ_R, na.rm = FALSE)

```



```

202.
203.   UTChh <- gsub("[^[:alnum:]]//'", "", tempflt$UTC..Hours.)
204.   UTChh[UTChh == ""] <- NA
205.   UTChh <- na.locf(UTChh, na.rm = FALSE)
206.   UTChh <- c(UTChh[9:(length(UTChh))], NA + 1:8) # aligns hours with minutes. Need
    to check works with both frames.
207.
208.   UTCmm <- gsub("[^[:alnum:]]//'", "", tempflt$UTC..Minutes.)
209.   UTCmm[UTCmm == ""] <- NA
210.   UTCmm <- na.locf(UTCmm, na.rm = FALSE)
211.
212.   # FLAPLVR <- tempflt$Flap.Lever.Pos
213.   # FLAPLVR[FLAPLVR == ""] <- NA
214.   # FLAPLVR <- na.locf(FLAPLVR, na.rm = FALSE) # not reliable
215.
216.   ACTYPE <- gsub("[/*]", "", tempflt$A.C.Type)
217.   ACTYPE[ACTYPE == ""] <- NA
218.   ACTYPE <- na.locf(ACTYPE, na.rm = FALSE)
219.
220.   ## Gear is down.
221.   geardn <- (gear == "Down 1")
222.
223.   ## Shutdown
224.   #shutdownlog <- (N11 < 20 & N12 < 20)
225.
226.   ## Aircraft is on ground
227.   onground <- (SQ_L == "Gnd 1" & SQ_N == "Gnd 1") | (SQ_R == "Gnd 1" & SQ_N == "Gnd
    1") |
228.   (SQ_L == "Gnd 1" & SQ_R == "Gnd 1")
229.
230.   ## Set up empties
231.   # descending <- logical(length(CAS))
232.   # climbing <- logical(length(CAS))
233.
234.   ## Climbing or descending?
235.   # ALTp8 <- c(NA + 1:8, ALT[1:(length(ALT) - 8)])
236.   # ALTp16 <- c(NA + 1:16, ALT[1:(length(ALT) - 16)])
237.   # ALTp24 <- c(NA + 1:24, ALT[1:(length(ALT) - 24)])
238.   # ALTp32 <- c(NA + 1:32, ALT[1:(length(ALT) - 32)])
239.   # ALTp40 <- c(NA + 1:40, ALT[1:(length(ALT) - 40)])
240.   #
241.   # descending <- ((ALT < ALTp8) & (ALTp8 < ALTp16) & (ALTp16 < ALTp24) & (ALTp24 <
    ALTp32) & (ALTp32 < ALTp40) & (ALT < ALTp40 - 42))
242.   # climbing <- ((ALT > ALTp8) & (ALTp8 > ALTp16) & (ALTp16 > ALTp24) & (ALTp24 >
    ALTp32) & (ALTp32 > ALTp40) & (ALT > ALTp40 + 42))
243.
244.   ## Take off point
245.   ongroundp1 <- c(NA + 1:16, onground[1:(length(onground) - 16)]) # creates vector
    shifting onground 2 secs down
246.   ongroundp1 <- as.logical(ongroundp1)
247.
248.   TOsearch <- logical(length(CAS))
249.   TOsearch <- (onground == FALSE) & (ongroundp1 == TRUE) & (geardn == TRUE) & (GS >
    60)
250.
251.   TakePoint <- which(TOsearch == TRUE)
252.
253.   if (length(TakePoint) == 0) {
254.
255.     errortype <- "NO T/O"
256.     errors[nrow(errors) + 1,] <- c(flight, errortype)
257.
258.     next
259.   }
260.
261.   TakePoint <- TakePoint[1]
262.
263.   ## Landing Point
264.   LDsearch <- logical(length(CAS))
265.   LDsearch <- (onground == TRUE) & (ongroundp1 == FALSE) & (geardn == TRUE) & (GS >
    60) & (ALT > -1000) &
266.   (ALT < 12000) & (GS < 250) & (RALT1 < 50)
267.
268.   LandPoint <- which(LDsearch == TRUE)
269.   if (length(LandPoint) == 0) {
270.
271.     errortype <- "NO LANDING"

```

```

272.     errors[nrow(errors) + 1,] <- c(flight, errortype)
273.
274.     next
275. }
276.
277. LandPoint <- LandPoint[length(LandPoint)]
278.
279. ## Height above TO runway
280. # TOelev <- mean(ALT[TakePoint - 40], ALT[TakePoint - 48], ALT[TakePoint - 56],
ALT[TakePoint - 64], ALT[TakePoint - 72], ALT[TakePoint - 80])
281. # HaTO <- (ALT - TOelev)
282.
283. ## Height above LD runway
284. # LDelev <- round(mean(ALT[LandPoint + 40], ALT[LandPoint + 48], ALT[LandPoint +
56], ALT[LandPoint + 64], ALT[LandPoint + 72],
285. #                      ALT[LandPoint + 80]))
286. # HaLD <- (ALT - LDelev)
287. #
288. # Precise point of TD
289. NMLAp1 <- c(NA, NMLA[1:(length(NMLA) - 1)])
290.
291. precTD <- which(((NMLA[(LandPoint - 40):(LandPoint + 40)]) - (NMLAp1[(LandPoint -
40):(LandPoint + 40)])) > 0.1)
292. ifelse((length(precTD) == 0), (precTD <- LandPoint), (precTD <- precTD[1] + 1))
293.
294. if (precTD != LandPoint) {precTD <- (LandPoint - 40 + precTD)} # landpoint - 40
due window for looking starts at
295. # landpoint - 40 in precTD above
296.
297. # Check to see if onground shows TRUE before precTD
298. precTDog <- which(onground[(precTD - 40):precTD] == TRUE)
299. if (length(precTDog) > 0) {precTD <- precTD - (40 - precTDog[1])}
300. }
301.
302.
303.
304. # time2TD <- c((precTD - 1):(precTD - (length(CAS))))
305. # time2TD <- time2TD * -1
306.
307. endRoll <- precTD + 96
308.
309. if (endRoll > length(CAS)) {
310.
311.     errortype <- "Landing c/o"
312.     errors[nrow(errors) + 1,] <- c(flight, errortype)
313.
314.     next
315. }
316.
317. # Runway heading
318. rwyHDG <- round(mean(MHDG[(endRoll - 40):endRoll]))
319.
320. ##### Weather #####
321. # Recorded wind 5 secs before touchdown
322. recWindSpd <- WINDSPD[precTD - 40]
323. recWindDir <- WINDDIR[precTD - 40]
324.
325. # Recorded wind components
326. recXwind <- WINDSPD * sin(2*pi*(WINDDIR - THDG)/360) # convert to radians!
327. recHDwind <- WINDSPD * cos(2*pi*(WINDDIR - THDG)/360)
328. rLandHDwind <- round(recHDwind[precTD - 40], digits = 1)
329. rLandXwind <- round(recXwind[precTD - 40], digits = 1)
330.
331. # Get weather
332. arrWX <- try(getDetailedWeather(arrID, flightdate, opt_all_columns = T))
333. if (length(arrWX) < 2) {arrWX <- NA} # 2 catches 'try' errors
334. if (is.null(ncol(arrWX))) {arrWX <- as.data.frame(rwyHDG)} # catches null and
leaves next check intact
335. if (ncol(arrWX) != 15) {arrWX <- NA}
336.
337. # get touchdown time
338. TDtime <- paste(UTChh[precTD], UTCmm[precTD], "00", sep = ":")
339. TDtime <- paste(flightdate, TDtime, sep = " ")
340. TDtime <- as.POSIXct(TDtime)
341. TDtimeChar <- as.character(TDtime)
342.
343. # remove any rows that are all NA

```

```

344.   if (!is.na(arrWX)) {arrWX <- arrWX[rowSums(is.na(arrWX)) != ncol(arrWX),]
345.   colnames(arrWX)[2] <- "TimeLocal"
346.
347.   wxTime <- which.min(abs(TDtime - arrWX$Time)) # find closest METAR to TDtime
348.
349.   landWx <- arrWX[wxTime,] # Get closest row.
350.
351.   # METAR winds
352.   metWindD <- as.numeric(landWx$WindDirDegrees)
353.   if (landWx$Wind_Direction == "Variable") {metWindD <- NA}
354.   metWindS <- round(as.numeric(landWx$Wind_SpeedKm_h) * 0.539957) # convert to kt
355.   metXwind <- round(metWindS * sin(2*pi*(metWindD - rwyHDG)/360))
356.   metHDwind <- round(metWindS * cos(2*pi*(metWindD - rwyHDG)/360))
357.
358.   # METAR gusts
359.   metGustS <- round(as.numeric(landWx$Gust_SpeedKm_h) * 0.539957) # convert to kt
360.   metXgust <- round(metGustS * sin(2*pi*(metWindD - rwyHDG)/360))
361.   metHDgust <- round(metGustS * cos(2*pi*(metWindD - rwyHDG)/360))
362.
363.   # METAR runway condition. Looks at METAR closest to landing and one before
364.   if (!is.na(arrWX$Events[wxTime])) {
365.     ifelse(arrWX$Events[wxTime] == "Rain"
366.           | arrWX$Events[wxTime] == "Rain-Snow"
367.           | arrWX$Events[wxTime] == "Snow"
368.           | arrWX$Events[wxTime] == "Thunderstorm"
369.           | arrWX$Events[wxTime] == "Rain-Thunderstorm",
370.           runWet <- TRUE, runWet <- FALSE)
371.   }
372.
373.   if (!is.na(arrWX$Events[wxTime])) & wxTime > 1 {
374.     ifelse(arrWX$Events[wxTime - 1] == "Rain"
375.           | arrWX$Events[wxTime - 1] == "Rain-Snow"
376.           | arrWX$Events[wxTime - 1] == "Snow"
377.           | arrWX$Events[wxTime - 1] == "Thunderstorm"
378.           | arrWX$Events[wxTime - 1] == "Rain-Thunderstorm",
379.           runWet <- TRUE, runWet <- FALSE)
380.   }
381.
382.   if (wxTime > 1) {
383.     if (is.na(arrWX$Events[wxTime]) & is.na(arrWX$Events[wxTime - 1])) {runWet <-
FALSE}
384.   }
385.
386.   # wxTime > 1 to account for when wxTime = 1
387.   if (wxTime == 1) {
388.     if (is.na(arrWX$Events[wxTime])) {runWet <- FALSE}
389.   }
390.   # METAR viz
391.   metVizkm <- landWx$VisibilityKm
392. }
393.
394. # If arrWX is NA then set weatherData derived variables to NA
395. if (is.na(arrWX)) {metHDwind <- NA}
396. if (is.na(arrWX)) {metXwind <- NA}
397. if (is.na(arrWX)) {metXgust <- NA}
398. if (is.na(arrWX)) {metHDgust <- NA}
399. if (is.na(arrWX)) {runWet <- NA}
400. if (is.na(arrWX)) {metWindS <- NA}
401. if (is.na(arrWX)) {metWindD <- NA}
402. if (is.na(arrWX)) {metVizkm <- NA}
403.
404.
405. ##### Asymmetric thrust #####
406. # use non-interpolated N1.
407. asymN1L <- as.numeric(tempflt$N1.Actual.Eng.1[(precTD - 40):endRoll])
408. asymN1R <- as.numeric(tempflt$N1.Actual.Eng.2[(precTD - 40):endRoll])
409. THRsplitt <- abs(asymN1L - asymN1R)
410. asymTHR <- which(THRsplitt > 10)
411. asymTHR <- length(asymTHR)
412.
413. maxN1L <- max(N11[precTD:endRoll])
414. maxN1R <- max(N12[precTD:endRoll])
415.
416. ##### Unstable app - using speed at 50ft only #####
417. RALT50 <- which.min(abs(RALT1[(precTD - 240):precTD] - 50))
418. RALT50 <- 240 - RALT50 # precTD-240.....RALT50.....precTD
419. #####don't want ^#####want ^#####

```

```

420.   RALT150 <- which.min(abs(RALT1[(precTD - 240):precTD] - 150)) # used for
      Glideslope at 150ft
421.   RALT150 <- 240 - RALT150
422.   spd50 <- CAS[precTD - RALT150]
423.   spdDelta50 <- spd50 - (APPSPD[precTD - RALT150])
424.   if (APPSPD[precTD - RALT150] == 0) {spdDelta50 <- NA}
425.
426.   ##### Landing G #####
427.   landG <- NMLA[(precTD - 40):(precTD + 40)]
428.   maxG <- which.max(landG)
429.   maxG <- landG[maxG]
430.
431.   ##### Heading deviation #####
432.   tdHDG <- MHDG[precTD]
433.   HDGdev <- abs(tdHDG - rwyHDG)
434.   HDGdev <- round(HDGdev, digits = 1)
435.   if (HDGdev > 180) {HDGdev <- (360 - HDGdev)}
436.
437.   # add in distribution, SD, mean, max, min for lat g, long g plus params from RWY
      EXC WG B
438.
439.   ##### Rudder deflection #####
440.   landRUDPOS <- RUDPOS[precTD:endRoll]
441.   sumRUDPOS <- summary(landRUDPOS)
442.   RUDsd <- sd(landRUDPOS)
443.   RUDmin <- as.numeric(sumRUDPOS[1])
444.   RUDmax <- as.numeric(sumRUDPOS[6])
445.   RUDmean <- as.numeric(sumRUDPOS[4])
446.   RUDmed <- as.numeric(sumRUDPOS[3])
447.
448.   ##### lat G #####
449.   landLATG <- LATACC[precTD:endRoll]
450.   sumlandLATG <- summary(landLATG)
451.   LATGsd <- sd(landLATG)
452.   LATGmin <- as.numeric(sumlandLATG[1])
453.   LATGmax <- as.numeric(sumlandLATG[6])
454.   LATGmean <- as.numeric(sumlandLATG[4])
455.   LATGmed <- as.numeric(sumlandLATG[3])
456.
457.   ##### lon G #####
458.   landLONG <- LONACC[precTD:endRoll]
459.   sumlandLONG <- summary(landLONG)
460.   LONGsd <- sd(landLONG)
461.   LONGmin <- as.numeric(sumlandLONG[1])
462.   LONGmax <- as.numeric(sumlandLONG[6])
463.   LONGmean <- as.numeric(sumlandLONG[4])
464.   LONGmed <- as.numeric(sumlandLONG[3])
465.   LONGp3 <- LONACC[precTD + 24]
466.   LONGp5 <- LONACC[precTD + 40]
467.   LONGp7 <- LONACC[precTD + 56]
468.   LONGp10 <- LONACC[precTD + 80]
469.
470.   ##### Nose wheel steering #####
471.   landNWS <- NWS[precTD:endRoll]
472.   sumlandNWS <- summary(landNWS)
473.   NWSsd <- sd(landNWS)
474.   NWSmin <- as.numeric(sumlandNWS[1])
475.   NWSmax <- as.numeric(sumlandNWS[6])
476.   NWSmean <- as.numeric(sumlandNWS[4])
477.   NWSmed <- as.numeric(sumlandNWS[3])
478.
479.   # "time-to" params
480.
481.   ##### Spoilers #####
482.   SPOILarm <- tempflt$Ground.Spoiler.Armed[(precTD - 40):precTD]
483.   SPOILarm <- gsub("[/*]", "", SPOILarm)
484.   SPOILarm <- which(SPOILarm == "Armed (1)")
485.   SPOILarm <- length(SPOILarm) > 0
486.
487.   LSPOIL <- tempflt$Spoiler.Pos.LH.2[precTD:endRoll]
488.   LSPOIL <- gsub("[/*]", "", LSPOIL)
489.   LSPOIL <- as.numeric(LSPOIL)
490.   LSPOILt <- min(which(LSPOIL > 0))
491.   if (LSPOILt == Inf) {
492.     LSPOILt <- NA
493.   }
494.

```

```

495.   RSPOIL <- tempflt$Spoiler.Pos.RH.2[precTD:endRoll]
496.   RSPOIL <- gsub("[/*]", "", RSPOIL)
497.   RSPOIL <- as.numeric(RSPOIL)
498.   RSPOILt <- min(which(RSPOIL > 0))
499.   if (RSPOILt == Inf) {
500.     RSPOILt <- NA
501.   }
502.
503.   SPOILt <- (min(c(RSPOILt, LSPoILt)))
504.   ifelse(SPOILt == 0, SPOILt <- NA, SPOILt <- SPOILt/8)
505.
506.   ##### Rev thrust #####
507.   REVL <- tempflt$Reverser.Deployed.Eng.1[precTD:endRoll]
508.   REVL <- gsub("[/*]", "", REVL)
509.   REVLt <- min(which(REVL == "DEPLOY (1)"))
510.   if (REVLt == Inf) {
511.     REVLt <- NA
512.   }
513.
514.   REVR <- tempflt$Reverser.Deployed.Eng.2[precTD:endRoll]
515.   REVR <- gsub("[/*]", "", REVR)
516.   REVRt <- min(which(REVR == "DEPLOY (1)"))
517.   if (REVRt == Inf) {
518.     REVRt <- NA
519.   }
520.
521.   REVt <- (min(c(REVRt, REVLt)))
522.   ifelse(REVt == 0, REVt <- NA, REVt <- REVt/8) # in seconds
523.
524.   ##### Brake px #####
525.   # not recorded!
526.
527.   ##### Brake pedals #####
528.   landBRKPED_R <- BRKPED_R[precTD:endRoll]
529.   landBRKPED_L <- BRKPED_L[precTD:endRoll]
530.   landBRKPED_Rt <- min(which(landBRKPED_R > 0))
531.   if (landBRKPED_Rt == Inf) {
532.     landBRKPED_Rt <- NA
533.   }
534.   landBRKPED_Lt <- min(which(landBRKPED_L > 0))
535.   if (landBRKPED_Lt == Inf) {
536.     landBRKPED_Lt <- NA
537.   }
538.
539.   if (is.na(landBRKPED_Lt)) {landBRKPED_Lt <- landBRKPED_Rt}
540.   if (is.na(landBRKPED_Rt)) {landBRKPED_Rt <- landBRKPED_Lt}
541.
542.   BRKPEDt <- (min(c(landBRKPED_Lt, landBRKPED_Rt)))
543.   ifelse(BRKPEDt == 0, BRKPEDt <- NA, BRKPEDt <- BRKPEDt/8) # in seconds
544.
545.   LBrPedsummary <- summary(landBRKPED_L)
546.   RBrPedsummary <- summary(landBRKPED_R)
547.
548.   LBrPedMax <- LBrPedsummary[6]
549.   RBrPedMax <- RBrPedsummary[6]
550.
551.   LBrPedMean <- LBrPedsummary[4]
552.   RBrPedMean <- RBrPedsummary[4]
553.
554.   LBrPedSum <- sum(landBRKPED_L)
555.   RBrPedSum <- sum(landBRKPED_R)
556.
557.   ##### Auto brake armed #####
558.   LOWBRK1arm <- tempflt$Deceleration.Low.Armed[(precTD - 40):precTD]
559.   LOWBRK1arm <- gsub("[/*]", "", LOWBRK1arm)
560.   LOWBRK1arm <- which(LOWBRK1arm == "Armed (1)")
561.   LOWBRK1arm <- length(LOWBRK1arm) > 0
562.
563.   LOWBRK2arm <- tempflt$Deceleration.Low.Armed..LO_ARMED.2.[(precTD - 40):precTD]
564.   LOWBRK2arm <- gsub("[/*]", "", LOWBRK2arm)
565.   LOWBRK2arm <- which(LOWBRK2arm == "Armed (1)")
566.   LOWBRK2arm <- length(LOWBRK2arm) > 0
567.
568.   MEDBRKarm <- tempflt$Deceleration.Medium.Armed[(precTD - 40):precTD]
569.   MEDBRKarm <- gsub("[/*]", "", MEDBRKarm)
570.   MEDBRKarm <- which(MEDBRKarm == "Armed (1)")
571.   MEDBRKarm <- length(MEDBRKarm) > 0

```

```

572.
573. MAXBRKarm <- tempflt$Deceleration.Maximum.Armed[(precTD - 40):precTD]
574. MAXBRKarm <- gsub("[/*]", "", MAXBRKarm)
575. MAXBRKarm <- which(MAXBRKarm == "Armed (1)")
576. MAXBRKarm <- length(MAXBRKarm) > 0
577.
578. autoBRK <- character()
579. if (LOWBRK1arm == TRUE) {autoBRK <- "LOW"}
580. if (LOWBRK2arm == TRUE) {autoBRK <- "LOW"}
581. if (MEDBRKarm == TRUE) {autoBRK <- "MED"}
582. if (MAXBRKarm == TRUE) {autoBRK <- "MAX"}
583.
584. if (length(autoBRK) == 0) {autoBRK <- "NA"}
585.
586. # Idle at touchdown?
587. TDidle <- logical()
588. TDidle <- ((N11[precTD] < 50) & (N12[precTD] < 50))
589.
590. #Pitch and Roll at TD
591. pitchTD <- PITCH[precTD]
592. rollTD <- ROLL[precTD]
593.
594. # speeds at TD
595. TDGspd <- round(GS[precTD])
596. TDCAS <- round(CAS[precTD])
597.
598. # GSDev at 50ft and 150ft
599. GSDEVC <- tempflt$Glideslope.Deviation..combined.
600. GSDEVC <- gsub("FU ", "-", GSDEVC)
601. GSDEVC <- gsub("FD ", "", GSDEVC)
602. GSDEVC <- as.numeric(GSDEVC)
603. ifelse(length(GSDEVC) > (length(which(is.na(GSDEVC))) + 100), GSDEVC <-
na.approx(GSDEVC, na.rm = FALSE), GSDEVC <- NA)
604. # in case GS is all NA or there are few NA samples. Hence plus 100.
605.
606. LOCDEVC <- tempflt$Localizer.Deviation..combined.
607. LOCDEVC <- gsub("FL ", "-", LOCDEVC)
608. LOCDEVC <- gsub("FR ", "", LOCDEVC)
609. LOCDEVC <- as.numeric(LOCDEVC)
610. ifelse(length(LOCDEVC) > (length(which(is.na(LOCDEVC))) + 100), LOCDEVC <-
na.approx(LOCDEVC, na.rm = FALSE), LOCDEVC <- NA)
611.
612.
613. GSlope50 <- GSDEVC[precTD - RALT50]
614. GSlope150 <- GSDEVC[precTD - RALT150]
615.
616. # Localiser
617. landLOC <- LOCDEVC[precTD:endRoll]
618. sumlandLOC <- summary(landLOC)
619. LOCsd <- sd(landLOC)
620. LOCmin <- as.numeric(sumlandLOC[1])
621. LOCmax <- as.numeric(sumlandLOC[6])
622. LOCmean <- as.numeric(sumlandLOC[4])
623. LOCmed <- as.numeric(sumlandLOC[3])
624.
625. # Weight
626. TDweight <- GRWEIGHT[precTD]
627. if (TDweight < 20000) {TDweight <- 100001} # can't be <- NA because next line
would fail
628. if (TDweight > 100000) {TDweight <- NA}
629.
630.
631. # Flap
632. TDflap <- FLAPANG[precTD]
633.
634. # A/C type
635. Fleet <- ACTYPE[precTD]
636.
637.
638. RexOut <- rbind(RexOut, data.frame(flight, Fleet, TDtimeChar, arrID, TDweight,
rwyHDG, recWindDir, recWindSpd,
639.                                     rLandHDwind, rLandXwind,
640.                                     metWindD, metWindS, metHDwind, metXwind,
metHDgust, metXgust, metVizkm, runWet, asymTHR,
641.                                     spdDelta50, maxG, HDGdev, RUDmax, RUDmin,
RUDmed, RUDmean, RUDsd,

```

```

642.                                     NWSmax, NWSmin, NWSmed, NWSmean, NWSsd,
643.     GSlope150,                                     GSlope50,  LOCmax, LOCmin, LOCmed, LOCmean,
644.     LOCsd,                                     LATGmax, LATGmin, LATGmed, LATGmean, LATGsd,
645.     LONGmax, LONGmin, LONGmed,                 LONGmean, LONGsd, LONGp3, LONGp5, LONGp7,
646.     LONGp10,                                     SPOILt, REvt, maxN1L, maxN1R, autoBRK, BRKPEDt,
647.     LBrPedSum, LBrPedMean, LBrPedMax,          RBrPedSum, RBrPedMean, RBrPedMax, TDidle,
648.     pitchTD, rollTD, TDGspd, TDCAS,            TDflap, codeVer, stringsAsFactors = FALSE))
649.
650.
651. rm("ACTYPE", "ALT", "ALTp16", "ALTp24", "ALTp32",
652.    "ALTp40", "ALTp8", "APPSPD", "asymN1L", "asymN1R", "asymTHR",
653.    "arrID", "arrWX", "asymN1L", "asymN1R", "asymTHR",
654.    "autoBRK", "BRKPED_L", "BRKPED_R", "climbing", "descending", "endRoll",
655.    "BRKPEDt", "CAS", "climbing", "descending", "endRoll",
656.    "errortype", "FLAPANG", "flightdate", "gear", "geardn",
657.    "Fleet", "flight", "flightdate", "gear", "geardn",
658.    "GRWEIGHT", "GS", "GSDEVC", "HaTO", "HDGdev",
659.    "GSlope50", "GSlope150", "HaLD", "HDGdev",
660.    "landBRKPED_L", "landBRKPED_Lt", "landBRKPED_R",
661.    "landBRKPED_Rt", "landG", "landLATG", "landLOC", "landLONG",
662.    "landNWS", "LandPoint", "landRUDPOS", "LATGmean", "LATGmed",
663.    "landWx", "LATACC", "LATGmax", "LATGmin", "LATGsd",
664.    "LDsearch", "LOCDEVC", "LOCmax", "LOCmean", "LOCmed",
665.    "LOCmin", "LOCsd", "LONACC", "LONGmean", "LONGmed",
666.    "LONGmax", "LONGmin", "LONGp10", "LONGp3", "LONGp5", "LONGp7",
667.    "LONGsd", "LOWBRK1arm", "LOWBRK2arm", "LSPOIL", "LSPOILt",
668.    "MAXBRKarm", "maxG", "MEDBRKarm", "metGustS", "metHDwind",
669.    "metGustS", "metHDgust", "metHDwind", "metVizkm", "metWindD",
670.    "metWinds", "metXgust", "metXwind", "MHDG", "N11", "N12",
671.    "MHDG", "N11", "N12", "NMLA", "NMLAp1",
672.    "NWS", "NWSmax", "NWSmean", "NWSmed", "NWSmin", "NWSsd",
673.    "onground", "ongroundp1", "PITCH", "RALT1", "RALT50", "RALT150", "RDSid",
674.    "pitchTD", "precTD", "RALT1", "RALT50", "RALT150", "RDSid",
675.    "recHDwind", "recWindDir", "recWindSpd", "REVL", "REVLt",
676.    "recXwind", "REVL", "REVLt", "REVR", "REVRt",
677.    "REvt", "rLandHDwind", "rLandHDwind", "RSPOIL", "RSPOILt",
678.    "rLandXwind", "ROLL", "rollTD", "RUDmax", "RUDmean", "RUDmed",
679.    "RUDmin", "RUDPOS", "RUDsd", "runWet", "rwyHDG",
680.    "spd50", "spdDelta50", "SPOILarm", "SQ_L", "SQ_N", "SQ_R",
681.    "SPOILt", "SSRP1", "SSRP2", "SSRP2", "SSRP2", "SSRP2",
682.    "SSRP2", "SSRP1", "SSRP2", "SSRP2", "SSRP2", "SSRP2",
683.    "sumlandLATG", "sumlandLOC", "sumlandLONG", "sumlandNWS", "sumRUDPOS",
684.    "TakePoint", "TDCAS", "TDflap", "TDGspd", "tdHDG", "TDidle",
685.    "TDGspd", "tdHDG", "TDidle", "TDtime", "TDtimeChar",
686.    "TDweight", "tempflt", "THDG", "THRANG_1", "THRANG_2", "THRsplit",
687.    "THRANG_1", "THRANG_2", "THRsplit", "time2TD", "TOelev",
688.    "TOsearch", "UTChh", "UTCmm", "v", "w", "WINDDIR", "WINDSPD",
689.    "v", "w", "WINDDIR", "WINDSPD", "wxTime",
690.    "x", "LBrPedsummary", "RBrPedsummary",
691.    "LBrPedMax", "RBrPedMax", "LBrPedMean", "RBrPedMean", "LBrPedSum", "RBrPedSum",
692.    "maxN1L", "maxN1R", "precTDOG" )
693. }
694.
695. # save
696. saveRDS(RexOut, file = "RexOut.rds")
697. saveRDS(VflightID, file = "VflightID.rds")

```

Appendix 8 – R code for comparison of crosswind calculations

```
1. # Recon_wind_v02.R
2. # Add Peter van der Geest's reconstructed surface winds to ReXout results previously
   obtained.
3. # v1 based on basic elements of RwyExc_v08.R
4. # v2 add average wind 20s
5.
6. PvdGCodeVer <- "v2"
7. library(zoo) ### for na.locf
8. library(data.table)
9. library(Hmisc) ### for %nin%
10. library(caTools)
11. library(gdata)
12. library(dplyr)
13.
14. ## For MAC
15. # setwd("~/CloudStation/DB_Work/R/RWYEXC/Recon_wind")
16. # flight.index <- readRDS("/Volumes/E/Rflights/flights_index_MAC_24APR17.rds")
17. # rexout <- readRDS("~/CloudStation/DB_Work/R/RWYEXC/REXC_clean/RexOut04_11_v07.rds")
18.
19. ## For DELL
20. setwd("E:/Rworking/Recon_wind")
21. flight.index <- readRDS("E:/Rflights/flights_index_E_24APR17.rds")
22. rexout <- readRDS("E:/Rworking/RWYEXC/RexOut04_11_v07.rds")
23.
24. rexout <- sample_n(rexout, 10000)
25. Vcross <- numeric(length(rexout$flight))
26. Vhead <- numeric(length(rexout$flight))
27. rLandHDwind25 <- numeric(length(rexout$flight))
28. rLandXwind25 <- numeric(length(rexout$flight))
29. rLandHDwind21 <- numeric(length(rexout$flight))
30. rLandXwind21 <- numeric(length(rexout$flight))
31.
32. rexout <- cbind(rexout, Vcross, Vhead, rLandHDwind25, rLandXwind25, rLandHDwind21,
   rLandXwind21)
33.
34. for (i in 1:length(rexout$flight)) {
35.   cat(paste(".", i, "."))
36.
37. path <- flight.index$fileName[like(flight.index$fileName, rexout$flight[i])]
38. if (length(path) == 0) {
39.   rexout$Vcross[i] <- NA
40.   rexout$Vhead[i] <- NA
41.   next}
42.
43. tempflt <- readRDS(path)
44.
45. CAS <- as.numeric(tempflt$Computed.Airspeed)
46. CAS <- na.approx(CAS, na.rm = FALSE)
47. N11 <- as.numeric(tempflt$N1.Actual.Eng.1)
48. N11 <- na.approx(N11, na.rm = FALSE)
49. N12 <- as.numeric(tempflt$N1.Actual.Eng.2)
50. N12 <- na.approx(N12, na.rm = FALSE)
51. GS <- as.numeric(tempflt$Ground.Speed)
52. GS <- na.approx(GS, na.rm = FALSE)
53. RALT1 <- as.numeric(tempflt$Radio.Altitude.1)
54. RALT1 <- na.approx(RALT1, na.rm = FALSE)
55. ALT <- as.numeric(tempflt$Altitude..1013.25mB.)
56. ALT <- na.approx(ALT, na.rm = FALSE)
57. MHDG <- as.numeric(tempflt$Magnetic.Heading)
58. MHDG <- na.approx(MHDG, na.rm = FALSE)
59. THDG <- as.numeric(tempflt$True.Heading)
60. THDG <- na.approx(THDG, na.rm = FALSE)
61. True.Air.Temp <- as.numeric(tempflt$True.Air.Temp)
62. True.Air.Temp <- na.approx(True.Air.Temp, na.rm = FALSE)
63. True.Air.Temp <- True.Air.Temp + 273.15 # to Kelvin
64. MACH <- as.numeric(tempflt$Mach.Number)
65. MACH <- na.approx(MACH, na.rm = FALSE)
```



```

66. DRIFT <- as.numeric(tempflt$Drift.Angle)
67. DRIFT <- na.approx(DRIFT, na.rm = FALSE)
68. WINDDIR <- as.numeric(tempflt$Wind.Direction.True)
69. WINDDIR <- na.locf(WINDDIR, na.rm = FALSE)
70. WINDSPD <- as.numeric(tempflt$Wind.Speed)
71. WINDSPD <- na.locf(WINDSPD, na.rm = FALSE)
72.
73. ## Normal Acc: at 8Hz but contains *. Need to remove *
74. NMLA <- gsub("[/*]", "", tempflt$Normal.Acceleration)
75. NMLA <- as.numeric(NMLA)
76.
77. ## Clean up used character params to remove "*" and backfill blanks.
78. gear <- gsub("[^[:alnum:]]//'", "", tempflt$Gear.Selector.Down)
79. gear[gear == ""] <- NA ### fills blanks with NA
80. gear <- na.locf(gear, na.rm = FALSE) ### Backfills NA with last value
81.
82. SQ_L <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...LH)
83. SQ_L[SQ_L == ""] <- NA
84. SQ_L <- na.locf(SQ_L, na.rm = FALSE)
85.
86. SQ_N <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...Nose)
87. SQ_N[SQ_N == ""] <- NA
88. SQ_N <- na.locf(SQ_N, na.rm = FALSE)
89.
90. SQ_R <- gsub("[^[:alnum:]]//'", "", tempflt$Land.Gear.Squat.Switch...RH)
91. SQ_R[SQ_R == ""] <- NA
92. SQ_R <- na.locf(SQ_R, na.rm = FALSE)
93.
94. ## Gear is down.
95. geardn <- (gear == "Down 1")
96.
97. ## Aircraft is on ground
98. onground <- (SQ_L == "Gnd 1" & SQ_N == "Gnd 1") | (SQ_R == "Gnd 1" & SQ_N == "Gnd 1")
99. | (SQ_L == "Gnd 1" & SQ_R == "Gnd 1")
100.
101. ## Take off point
102. ongroundp1 <- c(NA + 1:16, onground[1:(length(onground) - 16)]) # creates vector
    shifting onground 2 secs down
103. ongroundp1 <- as.logical(ongroundp1)
104.
105. T0search <- logical(length(CAS))
106. T0search <- (onground == FALSE) & (ongroundp1 == TRUE) & (geardn == TRUE) & (GS >
    60)
107.
108. TakePoint <- which(T0search == TRUE)
109.
110. TakePoint <- TakePoint[1]
111.
112. ## Landing Point
113. LDsearch <- logical(length(CAS))
114. LDsearch <- (onground == TRUE) & (ongroundp1 == FALSE) & (geardn == TRUE) & (GS >
    60) & (ALT > -1000) &
115. (ALT < 12000) & (GS < 250) & (RALT1 < 50)
116.
117. LandPoint <- which(LDsearch == TRUE)
118. LandPoint <- LandPoint[length(LandPoint)]
119.
120. # Precise point of TD
121. NMLAp1 <- c(NA, NMLA[1:(length(NMLA) - 1)])
122.
123. precTD <- which(((NMLA[(LandPoint - 40):(LandPoint + 40)]) - (NMLAp1[(LandPoint -
    40):(LandPoint + 40)])) > 0.1)
124. ifelse((length(precTD) == 0), (precTD <- LandPoint), (precTD <- precTD[1] + 1))
125.
126. if (precTD != LandPoint) {precTD <- (LandPoint - 40 + precTD)} # landpoint - 40 due
    window for looking starts at
127. # landpoint - 40 in precTD above
128.
129. # Check to see if onground shows TRUE before precTD
130. precTDOG <- which(onground[(precTD - 40):precTD] == TRUE)
131. if (length(precTDOG) > 0) {precTD <- precTD - (40 - precTDOG[1])}
132. }
133.
134. # True air speed calculation
135. Static.Air.Temp <- True.Air.Temp / (1 + (0.2 * MACH^2))
136. gascon <- 287.053

```

```

137. TASms <- MACH * sqrt(1.4 * gascon * Static.Air.Temp)
138. TAS <- TASms/0.514444
139.
140. # Wind calculation
141. DRIFTm5 <- DRIFT[(precTD - 200):(precTD - 40)]
142. TASm5 <- TAS[(precTD - 200):(precTD - 40)]
143. if(TASm5 == 0){
144.   rexout$Vcross[i] <- NA
145.   rexout$Vhead[i] <- NA
146.   next
147. }
148. GSm5 <- GS[(precTD - 200):(precTD - 40)]
149. rexout$Vcross[i] <- round(mean((TASm5 * sin((2*pi*DRIFTm5)/360))*(-1)), digits = 1)
# converts to radians
150. rexout$Vhead[i] <- round(mean(TASm5 * cos((2*pi*DRIFTm5)/360) - GSm5), digits = 1)
151.
152. ## Average winds 25-5s from TD and 21s-1s before TD
153. # Recorded wind components
154. recXwind <- WINDSPD * sin(2*pi*(WINDDIR - THDG)/360) # convert to radians!
155. recHDwind <- WINDSPD * cos(2*pi*(WINDDIR - THDG)/360)
156. rexout$rLandHDwind25[i] <- round(mean(recHDwind[(precTD - 200):(precTD - 40)]),
  digits = 1)
157. rexout$rLandXwind25[i] <- round(mean(recXwind[(precTD - 200):(precTD - 40)]), digits
  = 1)
158. rexout$rLandHDwind21[i] <- round(mean(recHDwind[(precTD - 168):(precTD - 8)]),
  digits = 1)
159. rexout$rLandXwind21[i] <- round(mean(recXwind[(precTD - 168):(precTD - 8)]), digits
  = 1)
160.
161. keep(rexout, flight.index, i, sure = TRUE)
162.
163. }
164.
165. # plot(sample.rexout$rLandXwind, sample.rexout$Vcross)
166. # abline(coef=c(0,1))
167.
168. rexout.cor <- as.data.frame(cbind(rexout$rLandXwind, rexout$Vcross,
  rexout$rLandXwind25, rexout$rLandXwind21))
169. cor(rexout.cor, use = "complete.obs")

```

Appendix 9 – R code for addition of runway data to existing dataset

```
1. ---
2. title: "Runway_data"
3. author: "David Barry"
4. date: "19/08/2017"
5. output: word_document
6. ---
7.
8. ```{r setup, include=FALSE}
9. knitr::opts_chunk$set(echo = TRUE)
10. ```
11.
12. ## Addition of runway data to REX results
13. This document describes how runway data was added to the working dataset.
14. ## Data source
15. Open data for airport runways is available from http://ourairports.com/data/.runways.csv 19AUG2017 version was downloaded.
16. Open data for IATA and ICAO airport codes is available from http://data.okfn.org/data/core/airport-codes. "airport-codes.csv" downloaded 19AUG2017.
17.
18.
19. ```{r}
20. library(data.table)
21. library(tidyverse)
22. library(dplyr)
23. library(gdata)
24. library(Hmisc)
25. runways <- fread("/Users/DJB/CloudStation/DB_Work/R/PhD/RWYExc_Bayes/runways.csv",
26. stringsAsFactors = FALSE, sep = ",",
27. header = TRUE, fill = TRUE)
28. airport_codes <- fread("https://raw.githubusercontent.com/datasets/airport-codes/master/data/airport-codes.csv",
29. stringsAsFactors = FALSE, sep = ",", header = TRUE, fill =
30. TRUE)
31.
32.
33.
34. ## Combining the working data with the runway codes....
35. ```{r}
36. colnames(airport_codes)[11] <- "arrID"
37. combined <- rex
38. combined <- merge(combined, airport_codes, by = "arrID", all.x = TRUE)
39.
40.
41.
42. Three extra rows are added, 1 for JRS and 2 for MUC, due to different entries in
43. airport codes table for closed airports.
44. These are removed:
45. ```{r}
46. combined <- combined[-124599,] ##Check whether indices have changed
47. combined <- combined[-297566,]
48. combined <- combined[-297568,]
49.
50.
51.
52. Remove rows with no arrival ID:
53. ```{r}
54. i <- which(combined$arrID == "___")
55. combined <- combined[-i,]
56.
57.
58.
59.
60. Now need to add runway info to dataframe. Airports have multiple runways e.g. 27L/27R.
61. Using runway headings it is possible to differentiate, say, 27 and 09, however it will
62. not be possible to differentiate L & R.
63. Average?
64. Take shortest?
65. Runways table contains one entry per physical runway i.e. 27/09 is one row.
66. Need to extract landing runway ID from flight ID.
67.
68.
```

```

59. ```{r}
60. arrRWY <- strsplit(combined$flight, "arr=")
61. y <- sapply(arrRWY[1], length)
62. arrRWY <- matrix(unlist(arrRWY), ncol = y, byrow = TRUE)
63. arrRWY <- arrRWY[,3]
64.
65. arrRWY <- strsplit(arrRWY, " fn=")
66. z <- sapply(arrRWY[1], length)
67. arrRWY <- matrix(unlist(arrRWY), ncol=z, byrow=TRUE)
68. arrRWY <- arrRWY[,1]
69. library(stringr)
70. arrRWY <- str_sub(arrRWY, 4)
71. combined$arrRWY <- arrRWY
72. combined$arrRWY <- trimws(combined$arrRWY)
73.
74. ```
75. Now need to match combined df to runways df....
76.
77. ```{r}
78. a <- merge(combined, runways, by.x = c("airport_ident", "arrRWY"), by.y =
  c("airport_ident", "le_ident"), all.x = FALSE, all.y = FALSE)
79. b <- merge(combined, runways, by.x = c("airport_ident", "arrRWY"), by.y =
  c("airport_ident", "he_ident"), all.x = FALSE, all.y = FALSE)
80. c <- bind_rows(a, b)
81. comData <- c
82. ```
83. Note that around 9000 rows were not matched and are thus discarded. <<<<<<<
  INVESTIGATE THIS compare rex with comData
84. Missing 9000 rows are due to changes in runway designators due to magnetic variation.
85. Cleaning these up.... (Note only 10 or more arrivals have been corrected)
86. ```{r}
87. i <- rex$flight[rex$flight %nin% comData$flight]
88. arrRWY <- strsplit(i, "arr=")
89. y <- sapply(arrRWY[1], length)
90. arrRWY <- matrix(unlist(arrRWY), ncol = y, byrow = TRUE)
91. View(arrRWY)
92. arrRWY <- arrRWY[,3]
93. arrRWY <- strsplit(arrRWY, " fn=")
94. z <- sapply(arrRWY[1], length)
95. arrRWY <- matrix(unlist(arrRWY), ncol=z, byrow=TRUE)
96. View(arrRWY)
97. arrRWY <- arrRWY[,1]
98. library(stringr)
99. arrRWY <- str_sub(arrRWY, 4)
100. arrID <- strsplit(i, "arr=")
101. y <- sapply(arrID[1], length)
102. arrID <- matrix(unlist(arrID), ncol = y, byrow = TRUE)
103. arrID <- arrID[,2]
104. arrID <- strsplit(arrID, " dep=")
105. z <- sapply(arrID[1], length)
106.
107. arrID <- matrix(unlist(arrID), ncol=z, byrow=TRUE)
108. i <- which(combined$arrID == "BRU" & combined$arrRWY == "02")
109. combined$arrRWY[i] <- "01"
110. i <- which(combined$arrID == "BRU" & combined$arrRWY == "20")
111. combined$arrRWY[i] <- "19"
112. which(combined$arrID == "ALA")
113. combined[6082,]
114. i <- which(combined$arrID == "ALA" & combined$arrRWY == "05R")
115. combined$arrRWY[i] <- "05"
116. i <- which(combined$arrID == "ALA" & combined$arrRWY == "23L")
117. combined$arrRWY[i] <- "23"
118. j <- which(combined$arrID == "ALA")
119. combined$arrRWY[j]
120. i <- which(combined$arrID == "ALA" & combined$arrRWY == "23R")
121. combined$arrRWY[i] <- "23"
122. i <- which(combined$arrID == "ALA" & combined$arrRWY == "05L")
123. combined$arrRWY[i] <- "05"
124. combined$arrRWY[j]
125. j <- which(combined$arrID == "ANK")
126. combined$arrRWY[j]
127.
128. combined[j,]
129. j <- which(combined$arrID == "DLM")
130. combined$arrRWY[j]
131. combined[60039,]
132. i <- which(combined$arrID == "DLM" & combined$arrRWY == "01R")

```

```

133. combined$arrRWY[i] <- "01"
134. i <- which(combined$arrID == "DLM" & combined$arrRWY == "19R")
135. combined$arrRWY[i] <- "19"
136. j <- which(combined$arrID == "FUE")
137. combined$arrRWY[j]
138. i <- which(combined$arrID == "FUE" & combined$arrRWY == "01R")
139. combined$arrRWY[i] <- "01"
140. i <- which(combined$arrID == "FUE" & combined$arrRWY == "19L")
141. combined$arrRWY[i] <- "19"
142. j <- which(combined$arrID == "GYD")
143. combined$arrRWY[j]
144. i <- which(combined$arrID == "GYD" & combined$arrRWY == "18")
145. combined$arrRWY[i] <- "17"
146. i <- which(combined$arrID == "GYD" & combined$arrRWY == "36")
147. combined$arrRWY[i] <- "35"
148.
149. table(combined$arrRWY[j])
150. j <- which(combined$arrID == "INV")
151. combined$arrRWY[j]
152. combined$ident[122643]
153. i <- which(combined$arrID == "INV" & combined$arrRWY == "06")
154. combined$arrRWY[i] <- "05"
155. i <- which(combined$arrID == "INV" & combined$arrRWY == "24")
156. combined$arrRWY[i] <- "23"
157. j <- which(combined$arrID == "KGS")
158. combined$ident[124782]
159. combined$arrRWY[j]
160. i <- which(combined$arrID == "KGS" & combined$arrRWY == "15")
161. combined$arrRWY[i] <- "14"
162. i <- which(combined$arrID == "KGS" & combined$arrRWY == "33")
163. combined$arrRWY[i] <- "32"
164. j <- which(combined$arrID == "KMD")
165. table(combined$arrRWY[j])
166. combined[125063,]
167. combined$flight[j]
168. table(combined$rrwyHDG[j])
169. combined$arrID[j] <- "ADD"
170. i <- which(combined$arrID == "AMM" & combined$rrwyHDG > 70 & combined$rrwyHDG < 74)
171. i <- which(combined$arrID == "ADD" & combined$rrwyHDG > 70 & combined$rrwyHDG < 74)
172. combined$ident[j] <- "HAAB"
173. combined$arrRWY[i] <- "07L"
174. i <- which(combined$arrID == "ADD" & combined$rrwyHDG > 249)
175. combined$arrRWY[i] <- "25R"
176. j <- which(combined$arrID == "LCY")
177. combined$flight[j]
178. combined$ident[j] <- "EGLL"
179. combined$arrID[j] <- "LHR"
180. combined$arrRWY[j] <- "27L"
181. j <- which(combined$arrID == "LIN")
182. table(combined$rrwyHDG[j])
183. table(combined$arrRWY[j])
184. i <- which(combined$arrID == "LIN" & combined$arrRWY == "36R")
185. combined$arrRWY[i] <- "36"
186. i <- which(combined$arrID == "LIN" & combined$arrRWY == "18L")
187. combined$arrRWY[i] <- "18"
188. table(combined$arrRWY[j])
189. j <- which(combined$arrID == "LYS")
190. table(combined$arrRWY[j])
191. combined$ident[278789]
192. i <- which(combined$arrID == "LYS" & combined$arrRWY == "18L")
193. combined$arrRWY[i] <- "17L"
194. i <- which(combined$arrID == "LYS" & combined$arrRWY == "18R")
195. combined$arrRWY[i] <- "17R"
196. i <- which(combined$arrID == "LYS" & combined$arrRWY == "36L")
197. combined$arrRWY[i] <- "36L"
198. combined$arrRWY[i] <- "35L"
199. i <- which(combined$arrID == "LYS" & combined$arrRWY == "36R")
200. combined$arrRWY[i] <- "35R"
201. table(combined$arrRWY[j])
202. j <- which(combined$arrID == "MAD")
203. table(combined$arrRWY[j])
204. combined$ident[278989]
205. i <- which(combined$arrID == "MAD" & combined$arrRWY == "33L")
206. combined$arrRWY[i] <- "32L"
207. i <- which(combined$arrID == "MAD" & combined$arrRWY == "33R")
208. combined$arrRWY[i] <- "32R"
209. j <- which(combined$arrID == "MAN")

```

```

210. table(combined$arrRWY[j])
211. i <- which(combined$arrID == "MAN" & combined$arrRWY == "___")
212. combined$arrRWY[i] <- "23R"
213. i <- which(combined$arrID == "MAN" & combined$arrRWY == "06L")
214. combined$arrRWY[i] <- "05L"
215. i <- which(combined$arrID == "MAN" & combined$arrRWY == "06R")
216. combined$arrRWY[i] <- "05R"
217. i <- which(combined$arrID == "MAN" & combined$arrRWY == "24L")
218. combined$arrRWY[i] <- "23L"
219. i <- which(combined$arrID == "MAN" & combined$arrRWY == "24R")
220. combined$arrRWY[i] <- "23R"
221. j <- which(combined$arrID == "MLH")
222. table(combined$arrRWY[j])
223. combined$ident[294445]
224. combined$ident[j] <- "LFSB"
225. i <- which(combined$arrID == "MLH" & combined$arrRWY == "16")
226. combined$arrRWY[i] <- "15"
227. i <- which(combined$arrID == "MLH" & combined$arrRWY == "26")
228. combined$rrwyHDG[i]
229. j <- which(combined$arrID == "PIK")
230. table(combined$arrRWY[j])
231. combined$ident[302722]
232. i <- which(combined$arrID == "PIK" & combined$arrRWY == "13")
233. combined$arrRWY[i] <- "12"
234. i <- which(combined$arrID == "PIK" & combined$arrRWY == "30")
235. i <- which(combined$arrID == "PIK" & combined$arrRWY == "31")
236. combined$arrRWY[i] <- "30"
237. j <- which(combined$arrID == "SDV")
238. table(combined$arrRWY[j])
239. table(combined$rrwyHDG[j])
240. combined$ident[j] <- "LLBG"
241. combined$arrID[j] <- "TLV"
242. i <- which(combined$arrID[j] == "TLV" & combined$rrwyHDG[j] > 115 &
combined$rrwyHDG[j] < 120)
243. j <- which(combined$arrID == "TLV")
244. table(combined$rrwyHDG[j])
245. i <- which(combined$arrID == "TLV" & combined$rrwyHDG > 115 & combined$rrwyHDG < 120)
246. combined$arrRWY[i] <- "12"
247. i <- which(combined$arrID == "TLV" & combined$rrwyHDG > 253 & combined$rrwyHDG < 257)
248. combined$arrRWY[i] <- "26"
249. i <- which(combined$arrID == "TLV" & combined$rrwyHDG > 295 & combined$rrwyHDG < 300)
250. combined$arrRWY[i] <- "30"
251. j <- which(combined$arrID == "SZG")
252. table(combined$rrwyHDG[j])
253. table(combined$arrRWY[j])
254. i <- which(combined$arrID == "SZG" & combined$arrRWY == "16")
255. combined$arrRWY[i] <- "15"
256. i <- which(combined$arrID == "SZG" & combined$arrRWY == "34")
257. combined$arrRWY[i] <- "33"
258. j <- which(combined$arrID == "TFS")
259. table(combined$arrRWY[j])
260. combined$ident[307640]
261. i <- which(combined$arrID == "TFS" & combined$arrRWY == "08")
262. combined$arrRWY[i] <- "07"
263. i <- which(combined$arrID == "TFS" & combined$arrRWY == "26")
264. combined$arrRWY[i] <- "26"
265. j <- which(combined$arrID == "TOJ")
266. table(combined$arrRWY[j])
267. table(combined$rrwyHDG[j])
268. combined$arrRWY[j] <- "32L"
269. combined$ident[j] <- "LEMD"
270. combined$arrID[j] <- "MAD"
271. j <- which(combined$arrID == "WIE")
272. table(combined$arrRWY[j])
273. combined$flight[313901]
274. combined$arrID[j] <- "FRA"
275. combined$ident[j] <- "EDDF"
276. combined$arrRWY[j] <- "07L"
277.
278.
279. Now merging combined with runways again:
280. ```{r}
281. a <- merge(combined, runways, by.x = c("ident", "arrRWY"), by.y = c("airport_ident",
"le_ident"), all.x = FALSE, all.y = FALSE)
282. b <- merge(combined, runways, by.x = c("ident", "arrRWY"), by.y = c("airport_ident",
"he_ident"), all.x = FALSE, all.y = FALSE)
283. c <- bind_rows(a, b)

```

```
284. comData <- c
285. ```
286. How many flights are now missing?
287. ```{r}
288. length(rex$flight[rex$flight %nin% comData$flight])
289. ```
290. Previously nearly 9000.
291.
292. Tidy up environment....
293. ```{r}
294. gdata::keep(rex, comData, sure = TRUE)
295. saveRDS(comData, file = "rex_plus_rwys.rds")
296. ```
297.
298.
299.
300. Note that the `echo = FALSE` parameter was added to the code chunk to prevent
    printing of the R code that generated the plot.
```

Appendix 10 – R code for crosswind modelling

```
1. ---
2. title: "Xwind_model"
3. author: "David Barry"
4. date: "21/05/2018"
5. output: html_document
6. ---
7. Airport: ALL X2011
8. ```{r setup, include=FALSE}
9. knitr::opts_chunk$set(echo = TRUE)
10. ```
11.
12. ## Modelling cross wind for BN node
13.
14. This document describes attempts made to model a BN node from observational data.
15. This document follows from Tailwind.Rmd
16.
17. ```{r, message=FALSE, warning=FALSE}
18. library(fitdistrplus)
19. library(gdata) #for *keep*
20. setwd("~/Documents/DB_Work/R/PhD/RWYExc_Bayes/BN_build/Nodes_complete/NodesX2011/ALL")
21. ```
22.
23. ## Load data
24. Loading the REX data which has been combined with runway data (see "Runway_data.Rmd").
25. ```{r}
26. comData <-
27.   readRDS("~/Documents/DB_Work/R/PhD/RWYExc_Bayes/BN_build/Nodes_complete/NodesX2011/com
28.   data_X2011_14MAY18.rds")
29.
30. Get crosswind data and plot absolute values as empirical cdf.
31. ```{r}
32. xwind <- comData$rLandXwind
33. xwind.abs <- abs(xwind)
34. xwind.abs.ecdf <- ecdf(xwind.abs)
35. plot(xwind.abs.ecdf)
36.
37.
38. Now plot Cullen and Frey graph to help choose a distribution for the model.
39. ```{r}
40. descdist(xwind.abs, boot = 1000)
41.
42.
43. A gamma distribution might be suitable. Firstly, using maximum goodness of fit method:
44. ```{r}
45. xwind.gamma <- fitdist(xwind.abs, "gamma", method = "mge")
46. xwind.gamma
47. plot(xwind.gamma)
48.
49.
50. Now creating the model distribution and comparing probabilities of crosswind > 35kt:
51. ```{r}
52. model.gamma <- rgamma(286770, 1.2573797, 0.2350315)
53. ecdf(model.gamma)(35)
54.
55.
56. There is one value greater than 35kt in the observed data. This model would predict
57.   around 130 to 180 observations (95% CI - see below) i.e. (1 - result from above) *
58.   286770. Note that this value will change each time due to the random sampling from the
59.   distribution.
60. ```{r}
61. result <- data.frame(above35 = numeric(0))
62.
63. for(i in 1:1000) {
64.   model.gamma.test <- rgamma(286770, 1.2573797, 0.2350315)
65.   ecdf35 <- ecdf(model.gamma.test)(35)
66.   above35 <- (1 - ecdf35)*286770
67.   result <- rbind(result, above35)
68. }
```



```

67. colnames(result) <- "samples"
68. quantile(result$samples, c(0.025, 0.975))
69. ```
70.
71. The model quantile for one value in 310,448 observations is around 49kt to 63kt (95%
    CI):
72. ```{r}
73. result2 <- data.frame(q.model.gamma = numeric(0))
74. for(i in 1:1000) {
75.   model.gamma <- rgamma(286770, 1.2573797, 0.2350315)
76.   q.model.gamma <- quantile(model.gamma, 0.9999968)
77.   result2 <- rbind(result2, q.model.gamma)
78. }
79. colnames(result2) <- "samples"
80. quantile(result2$samples, c(0.025, 0.975))
81. ```
82. This suggests that the model tails are too fat, so alternative methods will be tried.
    Firstly, matching quantile estimation:
83. ```{r}
84. xwind.gamma.qme.1 <- fitdist(xwind.abs, "gamma", method = "qme", probs = c((1-
    0.999549), 0.999549))
85. xwind.gamma.qme.1
86. plot(xwind.gamma.qme.1)
87. ```
88.
89. The probs arguments were chosen as the being the extreme quantiles >25kt, with the
    lower and upper ends mirrored.
90. ```{r}
91. length(xwind.abs[xwind.abs > 35])
92. ```
93.
94.
95. This model would produce around 2 to 11 (95% CI) values > 35kt:
96. ```{r}
97. result3 <- data.frame(model.gamma.qme.1 = numeric(0))
98. for(i in 1:1000) {
99.   model.gamma.qme.1 <- rgamma(310448, 1.0808519, 0.3168794)
100.   ecdf.model.gamma.qme.1 <- (1 - ecdf(model.gamma.qme.1)(35)) * 310448
101.   result3 <- rbind(result3, ecdf.model.gamma.qme.1)
102. }
103. colnames(result3) <- "samples"
104. quantile(result3$samples, c(0.025, 0.975))
105. ```
106.
107. Using the extreme quantiles has resulted in the middle of the distribution being
    poorly modeled, so trying again with 0.5 quantile:
108. ```{r}
109. xwind.gamma.qme.2 <- fitdist(xwind.abs, "gamma", method = "qme", probs = c(0.5,
    0.999549))
110. xwind.gamma.qme.2
111. plot(xwind.gamma.qme.2)
112. ```
113.
114.
115. ```{r}
116. result4 <- data.frame(model.gamma.qme.3 = numeric(0))
117. for(i in 1:1000) {
118.   model.gamma.qme.3 <- rgamma(286770, 1.9377020, 0.3942837)
119.   ecdf.model.gamma.qme.3 <- (1 - ecdf(model.gamma.qme.3)(25)) * 286770
120.   result4 <- rbind(result4, ecdf.model.gamma.qme.3)
121. }
122. colnames(result4) <- "samples"
123. quantile(result4$samples, c(0.025, 0.975))
124. ```
125.
126. Which estimates between 1 and 8 (95% CI) values > 35kt. Actual 1
127. And
128. Which estimates between 121 and 167 (95% CI) values > 25kt. Actual 138

```

Appendix 11 – R code used for airspeed modelling

```
1. ---
2. title: "CAS_delta"
3. author: "David Barry"
4. date: "24/11/2017"
5. output:
6.   html_document:
7.     df_print: paged
8. ---
9.
10. ```{r setup, include=FALSE}
11. knitr::opts_chunk$set(echo = TRUE)
12. ```
13.
14. ## Modelling difference between actual and target airspeed at 50ft for BN node
15.
16. This document describes attempts made to model a BN node from observational data.
17. This document follows from Tailwind_model.Rmd
18.
19. ```{r libraries, echo=FALSE}
20. library(fitdistrplus)
21. library(gdata) #for *keep*
22. library(EnvStats) # for evd distribution (and others)
23. ```
24.
25. ## Load data
26. Loading the REX data which has been combined with runway data (see "Runway_data.Rmd").
27. ```{r, echo=FALSE}
28. comData <-
29.   readRDS("~/Documents/DB_Work/R/PhD/RWYExc_Bayes/Clean_data/comData02OCT17.rds")
30. Get CAS Delta data and plot.
31. ```{r, echo=FALSE}
32. cas_delta <- comData$spdDelta50
33. cas_delta <- cas_delta[complete.cases(cas_delta)]
34. i <- which(cas_delta > 0)
35. cas_delta_pos <- cas_delta[i]
36. hist(cas_delta_pos)
37. cas_delta.ecdf <- ecdf(cas_delta)
38. ```
39.
40. ```{r, echo=FALSE}
41. cas_delta_pos <- cas_delta_pos[complete.cases(cas_delta_pos)]
42. descdist(cas_delta_pos, boot = 1000)
43. ```
44.
45.
46.
47. Trying a Gamma distribution first. Note that the interest is primarily in the positive
48. values (i.e. aircraft too fast), therefore the negative values will be ignored.
49. ```{r, echo=FALSE}
50. cas_delta_gamma_1 <- fitdist(cas_delta_pos, "gamma", method = "mge")
51. cas_delta_gamma_1
52. plot(cas_delta_gamma_1)
53. ```
54.
55. This shows that the quantiles between 15 and 30 are not very well modelled, therefore
56. matching quantile estimation will be used. First, trying the quantiles at 20 and 35:
57. ```{r}
58. ecdf(cas_delta_pos)(20)
59. ecdf(cas_delta_pos)(35)
60. ```
61. ```{r}
62. cas_delta_gamma_2 <- fitdist(cas_delta_pos, "gamma", method = "gme", probs =
63.   c(0.999832, 0.999996))
64. cas_delta_gamma_2
65. plot(cas_delta_gamma_2)
66. ```
67. Values above 15kt appear to be modelled quite well, however it might be possible to
68. model the lower values better.
```

```

67.
68. ```{r}
69. ecdf(cas_delta_pos)(10)
70. ecdf(cas_delta_pos)(30)
71. ```
72.
73. ```{r}
74. cas_delta_gamma_3 <- fitdist(cas_delta_pos, "gamma", method = "gme", probs =
  c(0.97265, 0.999988))
75. cas_delta_gamma_3
76. plot(cas_delta_gamma_3)
77. ```
78. This looks like a fairly good compromise, so the next step is to test how well it
  models the higher values.
79.
80. ```{r}
81. result.3 <- data.frame(model.gamma.3 = numeric(0))
82. for(i in 1:1000) {
83.   model.gamma.3 <- rgamma(250018, shape = 1.2297603, rate = 0.4061938) # 250018 is +ve
     values only
84.   ecdf.model.gamma.3 <- (1 -ecdf(model.gamma.3)(20)) * 250018
85.   result.3 <- rbind(result.3, ecdf.model.gamma.3)
86. }
87. colnames(result.3) <- "samples"
88. quantile(result.3$samples, c(0.025, 0.975))
89. ```
90. From the empirical data, the actual number of occurrences > 20kt was 42, therefore the
  model is over-estimating the extreme values.
91. Trying quantiles at 10 and 25 instead:
92.
93. ```{r}
94. ecdf(cas_delta_pos)(10)
95. ecdf(cas_delta_pos)(25)
96. ```
97.
98. ```{r}
99. cas_delta_gamma_4 <- fitdist(cas_delta_pos, "gamma", method = "gme", probs =
  c(0.97265, 0.99998))
100. cas_delta_gamma_4
101. plot(cas_delta_gamma_4)
102. ```
103. The model fits extremely well, apart from the top 5 extreme values. Before any
  further work is undertaken, a visual check of these 5 data points will be used to
  check they are real values.
104. id=321310
105. Flap 3 approach, no AT, speed kept high above target, possibly deliberately due to
  Flap 3.
106. id=175183
107. Using selected speed rather than Managed. Target speed fluctuating between 0 and
  145kt. Approach wasn't fast.
108. id=273449
109. Recorded Vapp 124kt; Selected 158kt used for approach, hence fast.
110. id=199350
111. Fast and unstabilised.
112. id=342817
113. Not fast. Target fluctuating as in id=175183.
114.
115. Three out of the five extreme values were real, therefore it is worth modelling the
  extreme empirical values with some accuracy.
116.
117. ```{r}
118. ecdf(cas_delta_pos)(17)
119. ecdf(cas_delta_pos)(30)
120. ```
121.
122. ```{r}
123. cas_delta_gamma_5 <- fitdist(cas_delta_pos, "gamma", method = "gme", probs =
  c(0.999344, 0.999988))
124. cas_delta_gamma_5
125. plot(cas_delta_gamma_5)
126. ```
127. This seems good for values > 10kt. Checking performance:
128.
129. ```{r}
130. result.5 <- data.frame(model.gamma.5 = numeric(0))
131. for(i in 1:1000) {

```

```

132. model.gamma.5 <- rgamma(250018, shape = 0.2880172, rate = 0.2903899) # 250018 is
+ve values only
133. ecdf.model.gamma.5 <- (1 -ecdf(model.gamma.5)(20)) * 250018
134. result.5 <- rbind(result.5, ecdf.model.gamma.5)
135. }
136. colnames(result.5) <- "samples"
137. quantile(result.5$samples, c(0.025, 0.975))
138. ```
139. The model predicts between 47 and 78 values > 20kt (95% CI); the actual value was
42. This is probably close enough. Now checking more extreme values:
140. ```{r}
141. result.6 <- data.frame(model.gamma.6 = numeric(0))
142. for(i in 1:1000) {
143. model.gamma.6 <- rgamma(250018, shape = 0.2880172, rate = 0.2903899) # 250018 is
+ve values only
144. ecdf.model.gamma.6 <- (1 -ecdf(model.gamma.6)(25)) * 250018
145. result.6 <- rbind(result.6, ecdf.model.gamma.6)
146. }
147. colnames(result.6) <- "samples"
148. quantile(result.6$samples, c(0.025, 0.975))
149. ```
150. The model predicts between 6 and 20 (CI 95%) values > 25kt. The actual was 5. This
is considered acceptable as it slightly over estimates in the direction of greater
caution, that is, it does not underestimate extreme values.
151.
152. 10MAY18
153. However! This poorly models the lower values, which are still important i.e. the
range 10-20 is still important.
154. Try evd on -ve and +ve values:
155.
156. ```{r}
157. cas_delta.gumbel.1 <- fitdist(cas_delta, "evd", method = "mge", start =
list(location = 1, scale = 1), gof = "CvM")
158. plot(cas_delta.gumbel.1)
159. cas_delta.gumbel.1
160.
161. ```
162.
163. ```{r}
164. result.7 <- data.frame(model.evd.1 = numeric(0))
165. for(i in 1:1000) {
166. model.evd.1 <- revd(310173, location = 1.235965, scale = 2.501428) # 310173 is
complete cases
167. ecdf.model.evd.1 <- (1 -ecdf(model.evd.1)(25)) * 310173
168. result.7 <- rbind(result.7, ecdf.model.evd.1)
169. }
170. colnames(result.7) <- "samples"
171. quantile(result.7$samples, c(0.025, 0.975))
172. ```
173.
174. The EVD models around 15 to 33 values > 25kt, which is slightly high, but
acceptable.
175.
176. Now getting probabilities for intervals <10, 10-20, >20:
177.
178. ```{r}
179. pevd(c(10,20), location = 1.235965, scale = 2.501428)
180. 0.9994478 - 0.9703597
181. 1- pevd(c(20), location = 1.235965, scale = 2.501428)
182.
183. ```

```

Appendix 12 – R code use for glide slope modelling

```
1. ---
2. title: "Glideslope"
3. author: "David Barry"
4. date: "16/10/2017"
5. output: html_document
6. ---
7.
8. ```{r setup, include=FALSE}
9. knitr::opts_chunk$set(echo = TRUE)
10. ```
11.
12. ## Modelling Glideslope deviation at 150ft for BN node
13.
14. This document describes attempts made to model a BN node from observational data.
15. This document follows from Landing_G.Rmd
16.
17. ```{r libraries}
18. library(fitdistrplus)
19. library(gdata) #for *keep*
20. library(truncnorm)
21. setwd("~/CloudStation/DB_Work/R/PhD/RWYExc_Bayes/BN_build/Nodes_complete/ALL/Glideslop
22. e")
23.
24. ## Load data
25. Loading the REX data which has been combined with runway data (see "Runway_data.Rmd").
26. ```{r}
27. comData <-
28.   readRDS("~/Documents/DB_Work/R/PhD/RWYExc_Bayes/Clean_data/comData02OCT17.rds")
29.
30. Get glideslope at 150ft data and plot.
31. ```{r}
32. GS150 <- comData$GSlope150
33. hist(GS150)
34. GS150.ecdf <- ecdf(GS150)
35.
36. ```{r}
37. plot(density(GS150, na.rm = TRUE, bw = 0.05))
38.
39. Plotting Cullen and Frey:
40.
41. ```{r}
42.
43. GS150.comp <- GS150[complete.cases(GS150)]
44. GS150.comp <- GS150.comp[GS150.comp < 2]
45. GS150.comp <- GS150.comp[GS150.comp > -2]#remove spurious values
46. descdist(GS150.comp, boot = 1000)
47.
48.
49.
50. This shows that there is no obvious candidate for the distribution. However, most
    airlines, including the source of the empirical data, have a standard operating
    procedure which states that approached must be within one dot glideslope indication.
    As above glideslope is the main risk precursor for runway excursions (i.e. high on
    profile, landing deep), it is sufficient to use the probability of > 1 dot from the
    empirical density function.
51. The probability of a > 1 dot high on the glideslope at 150ft is 0.008669788:
52. ```{r}
53. 1 - ecdf(GS150.comp)(1)
54.
55.
56. GS150 <- comData$GSlope150
57. hist(GS150)
58. GS150.comp <- GS150[complete.cases(GS150)]
59. GS150.comp <- GS150.comp[GS150.comp < 2] #remove spurious values
60. descdist(GS150.comp, boot = 1000)
61. plot(density(GS150.comp))
62. GS150.comp <- GS150.comp[GS150.comp > -2]#remove spurious values
63. plot(density(GS150.comp))
64. descdist(GS150.comp, boot = 1000)
```

```

65. descdist(GS150.comp, boot = 1000, discrete = TRUE)
66. library("truncdist",
  lib.loc="/Library/Frameworks/R.framework/Versions/3.5/Resources/library")
67. library("truncnorm",
  lib.loc="/Library/Frameworks/R.framework/Versions/3.5/Resources/library")
68. GS150.tnorm <- fitdist(GS150.comp, "truncnorm", a = -2, b = 2)
69. GS150.tnorm <- fitdist(GS150.comp, "truncnorm")
70. GS150.tnorm <- fitdist(GS150.comp, "truncnorm", start = 0)
71. GS150.tnorm <- fitdist(GS150.comp, "truncnorm", start = 0, method = "mge")
72. GS150.tnorm <- fitdist(GS150.comp, "truncnorm", start = 0, method = "mme")
73. GS150.tnorm <- fitdist(GS150.comp, "norm")
74. GS150.tnorm
75. plot(GS150.tnorm)
76. gs150.gamma.1 <- fitdist(GS150.comp, "gamma", method = "qme", probs = c(0.01, 0.99))
77. ??students
78. ??students
79. gs150.t.1 <- fitdist(GS150.comp, "t")
80. gs150.t.1 <- fitdist(GS150.comp, "t", start = 0.1)
81. gs150.t.1
82. plot(gs150.t.1)
83. gs150.t.1 <- fitdist(GS150.comp, "cauchy", start = 0.1)
84. gs150.t.1
85. plot(gs150.t.1)
86. ??laplace
87. install.packages("rmutil")
88. library("rmutil",
  lib.loc="/Library/Frameworks/R.framework/Versions/3.5/Resources/library")
89. gs150.t.1 <- fitdist(GS150.comp, "laplace")
90. gs150.t.1 <- fitdist(GS150.comp, "laplace", start = 0)
91. gs150.t.1
92. plot(gs150.t.1)
93. ??Logistic
94. gs150.t.1 <- fitdist(GS150.comp, "logis", start = 0)
95. gs150.t.1
96. plot(gs150.t.1)
97. ??glogis
98. gs150.t.1 <- fitdist(GS150.comp, "glogis", start = 0.1)
99. plot(gs150.t.1)
100. gs150.t.1
101. GS.gumbel.1 <- fitdist(GS150.comp, "evd", method = "mge", start = list(location = 1,
  scale = 1), gof = "CvM")
102. plot(GS.gumbel.1)
103. GS.gumbel.1 <- fitdist(GS150.comp, "evd", method = "mge", start = list(location = 0,
  scale = 1), gof = "CvM")
104. GS.gumbel.1 <- fitdist(GS150.comp, "evd", method = "mge", start = list(location =
  0.1, scale = 1), gof = "CvM")
105. gs150.t.1 <- fitdist(GS150.comp, "skewlaplace", start = 0)
106. gs150.t.1
107. plot(gs150.t.1)
108. GS150.tnorm
109. hist(rtruncnorm(300000, a = -2, b = 2, mean = -0.04825448, sd = 0.33853687))
110.
111. ```{r}
112. library(SuppDists)
113. parms <- JohnsonFit(GS150.comp, moment = "find")
114. hist(GS150.comp, freq=FALSE)
115. plot(function(x)dJohnson(x,parms),-3,3,add=TRUE, col = "red")
116. result.1 <- data.frame(model.johnson.1 = numeric(0))
117. for(i in 1:1000) {
118.   model.johnson.1 <- rJohnson(300530, parms) # 250018 is +ve values only
119.   ecdf.model.johnson.1 <- (1 -ecdf(model.johnson.1)(-1)) * 300530
120.   result.1 <- rbind(result.1, ecdf.model.johnson.1)
121. }
122. colnames(result.1) <- "samples"
123. quantile(result.1$samples, c(0.025, 0.975))
124.

```

Appendix 13 – R code use for normal acceleration at landing modelling

```
1. ---
2. title: "Landing_G"
3. author: "David Barry"
4. date: "15/10/2017"
5. output:
6.   html_document: default
7.   word_document: default
8. ---
9.
10. ```{r setup, include=FALSE}
11. knitr::opts_chunk$set(echo = TRUE)
12. ```
13.
14. ## Modelling landing normal acceleration (NMLA) for BN node
15.
16. This document describes attempts made to model a BN node from observational data.
17. This document follows from Tailwind.Rmd
18.
19. ```{r libraries}
20. library(fitdistrplus)
21. library(gdata) #for *keep*
22. library(EnvStats) # for evd distribution (and others)
23. ```
24.
25. ## Load data
26. Loading the REX data which has been combined with runway data (see "Runway_data.Rmd").
27. ```{r}
28. comData <-
29.   readRDS("~/CloudStation/DB_Work/R/PhD/RWYExc_Bayes/Clean_data/comData02OCT17.rds")
30. ```
31. Get landing NMLA data and plot.
32. ```{r}
33. land.G <- comData$maxG
34. hist(land.G)
35. land.G.ecdf <- ecdf(land.G)
36. ```
37. Now plot Cullen and Frey graph to help choose a distribution for the model.
38. ```{r}
39. descdist(land.G, boot = 1000)
40. ```
41. The tailwind and crosswind modelling showed similar Cullend and Frey graphs and a
42. gamma distribution was used to model those variables. Trying gamma with quantile
43. estimation method first:
44. ```{r}
45. land.G.gamma.1 <- fitdist(land.G, "gamma", method = "qme", probs = c(0.01, 0.99))
46. land.G.gamma.1
47. plot(land.G.gamma.1)
48. ```
49. The model underestimates the mid-values, so 0.3 and 0.99 will be used for the next
50. quantiles:
51. ```{r}
52. land.G.gamma.2 <- fitdist(land.G, "gamma", method = "qme", probs = c(0.3, 0.99))
53. land.G.gamma.2
54. plot(land.G.gamma.2)
55. ```
56. How does this work in practice?
57. Eleven landings had G > 2.1:
58. ```{r}
59. length(which(land.G > 2.1))
60. ```
61. How does the model compare?
62. ```{r}
63. result.1 <- data.frame(model.gamma.1 = numeric(0))
64. for(i in 1:1000) {
65.   model.gamma.1 <- rgamma(310448, shape = 133.0696, rate = 104.7832)
66.   ecdf.model.gamma.1 <- (1 - ecdf(model.gamma.1)(2.1)) * 310448
67. }
```

```

64. result.1 <- rbind(result.1, ecdf.model.gamma.1)
65. }
66. colnames(result.1) <- "samples"
67. quantile(result.1$samples, c(0.025, 0.975))
68. ```
69. The tail is too thin. What is the quantile for > 2.1g?
70. ```{r}
71. ecdf(land.G)(2.1)
72. ```
73. Using the quantile from above:
74. ```{r}
75. land.G.gamma.3 <- fitdist(land.G, "gamma", method = "qme", probs = c(0.3, 0.9999646))
76. land.G.gamma.3
77. plot(land.G.gamma.3)
78. ```
79.
80. Try with a different ditribution. Lognormal?
81. ```{r}
82. land.G.lnorm <- fitdist(land.G, "lnorm")
83. plot(land.G.lnorm)
84. land.G.lnorm
85. ```
86.
87. ```{r}
88. result.2 <- data.frame(model.lnormal.1 = numeric(0))
89. for(i in 1:1000) {
90. model.lnormal.1 <- rlnorm(310448, meanlog = 0.23856197, sdlog = 0.07336574)
91. ecdf.model.lnormal.1 <- (1 - ecdf(model.lnormal.1)(2.1)) * 310448
92. result.2 <- rbind(result.2, ecdf.model.lnormal.1)
93. }
94. colnames(result.2) <- "samples"
95. quantile(result.2$samples, c(0.025, 0.975))
96. ```
97.
98. Will try Weibull as an alternative.
99. ```{r}
100. land.G.Weibull <- fitdist(land.G, "weibull")
101. plot(land.G.Weibull)
102.
103.
104. Not good!
105.
106. ```{r}
107. plot(density(land.G, bw = 0.02))
108. land.G.ext <- land.G[which(land.G > 1.6)]
109. ```
110.
111.
112. ```{r}
113. land.G.ext.gamma.1 <- fitdist(land.G.ext, "gamma", method = "qme", probs = c(0.6,
0.99))
114. land.G.ext.gamma.1
115. plot(land.G.ext.gamma.1)
116. ```
117. The above doesn't model the low values very well, however it does an acceptable job
with the extreme values, giving between 7 and 20 values greater than 2.1g, compared
with 11 in the empirical data.
118.
119. ```{r}
120. result.3 <- data.frame(model.gamma.2 = numeric(0))
121. for(i in 1:1000) {
122. model.gamma.2 <- rgamma(1034, shape = 68.42273, rate = 42.03663) # 1034 is > 1.6g
123. ecdf.model.gamma.2 <- (1 - ecdf(model.gamma.2)(2.1)) * 1034
124. result.3 <- rbind(result.3, ecdf.model.gamma.2)
125. }
126. colnames(result.3) <- "samples"
127. quantile(result.3$samples, c(0.025, 0.975))
128. ```
129.
130.
131. In the empirical data there are 1034 values > 1.6g, so the probability of landing g
< 1.6g is 99.67% i.e.
132. ```{r}
133. ecdf(land.G)(1.6) # 1- (1034/310448)
134. ```
135. If the landing is > 1.6g, the probability can be approximated with the gamma
distribution with shape = 68.42273 and rate = 42.03663.

```



```

136.
137. 08MAY18
138. However.... this model can't be applied in Agena, so need to find alternative model.
139.
140. ```{r}
141. land.G.gumbel.1 <- fitdist(land.G, "evd", start = list(location = 1, scale = 2),
142.   lower = c(0, 0))
143. # note lower above from : https://stats.stackexchange.com/questions/158163/why-
144.   does-this-data-throw-an-error-in-r-fitdistr
145. land.G.gumbel.1
146. plot(land.G.gumbel.1)
147.
148. Looks OK. Testing:
149.
150. ```{r}
151. result.gumbel.1 <- data.frame(model.gumbel.1 = numeric(0))
152. for(i in 1:1000) {
153.   model.gumbel.1 <- revd(310448, location = 1.22823757, scale = 0.08085452)
154.   ecdf.model.gumbel.1 <- (1 - ecdf(model.gumbel.1)(2.1)) * 310448
155.   result.gumbel.1 <- rbind(result.gumbel.1, ecdf.model.gumbel.1)
156. }
157. colnames(result.gumbel.1) <- "samples"
158. quantile(result.gumbel.1$samples, c(0.025, 0.975))
159.
160. Trying with quantile matching:
161.
162. ```{r}
163. land.G.gumbel.2 <- fitdist(land.G, "evd", method = "qme", probs = c(0.3, 0.9999646),
164.   start = list(location = 1, scale = 1), lower = c(0, 0)) # upper quantile for 2.1g
165. # note lower above from : https://stats.stackexchange.com/questions/158163/why-
166.   does-this-data-throw-an-error-in-r-fitdistr
167. land.G.gumbel.2
168. plot(land.G.gumbel.2)
169.
170.
171.
172. ```{r}
173. result.gumbel.2 <- data.frame(model.gumbel.2 = numeric(0))
174. for(i in 1:1000) {
175.   model.gumbel.2 <- revd(310448, location = 1.225838, scale = 0.085324)
176.   ecdf.model.gumbel.2 <- (1 - ecdf(model.gumbel.2)(2.1)) * 310448
177.   result.gumbel.2 <- rbind(result.gumbel.2, ecdf.model.gumbel.2)
178. }
179. colnames(result.gumbel.2) <- "samples"
180. quantile(result.gumbel.2$samples, c(0.025, 0.975))
181.
182. That looks good. Trying others....
183.
184. ```{r}
185. land.G.chi.1 <- fitdist(land.G, "chi", start = list(df = 3), lower = 0)
186. land.G.chi.1
187. plot(land.G.chi.1)
188.
189. Dreadful.
190.
191. ```{r}
192. land.G.tnorm.1 <- fitdist(land.G, "normTrunc", start = list(mean = 1.3, sd = 0.2))
193. land.G.tnorm.1
194. plot(land.G.tnorm.1)
195.
196. Poor. See Gumbel/evd.
197.
198. ```{r}
199. pevd(c(1.6, 2.0), location = 1.225838, scale = 0.085324)
200. 0.9998853 - 0.9876168
201. 1 - pevd(2.0, location = 1.225838, scale = 0.085324)
202.

```

Appendix 14 – Example of R code for modelling a causal factor at a specific airport

```
1. ---
2. title: "CAS_delta"
3. author: "David Barry"
4. date: "21/05/2018"
5. output:
6.   html_document:
7.     df_print: paged
8. ---
9.
10. ```{r setup, include=FALSE}
11. knitr::opts_chunk$set(echo = TRUE)
12. ```
13.
14. ## Modelling difference between actual and target airspeed at 50ft for BN node. BHD
   airport.
15.
16. This document describes attempts made to model a BN node from observational data.
17. This document follows from Tailwind_model.Rmd
18.
19. ```{r libraries, echo=FALSE}
20. library(fitdistrplus)
21. library(gdata) #for *keep*
22. library(EnvStats) # for evd distribution (and others)
23. setwd("~/Documents/DB_Work/R/PhD/RWYExc_Bayes/BN_build/Nodes_complete/NodesX2011/BHD")
24. ```
25.
26. ## Load data
27. Loading the REX data which has been combined with runway data (see "Runway_data.Rmd").
28. ```{r, echo=FALSE}
29. comData <-
   readRDS("~/Documents/DB_Work/R/PhD/RWYExc_Bayes/BN_build/Nodes_complete/NodesX2011/com
   data_X2011_14MAY18.rds")
30. comData <- comData[comData$arrID == "BHD",]
31. ```
32. Get CAS Delta data and plot.
33. ```{r, echo=FALSE}
34. cas_delta <- comData$spdDelta50
35. i <- which(cas_delta > 0)
36. cas_delta_pos <- cas_delta[i]
37. hist(cas_delta_pos)
38. cas_delta.ecdf <- ecdf(cas_delta)
39. ```
40.
41. ```{r, echo=FALSE}
42. cas_delta_pos <- cas_delta_pos[complete.cases(cas_delta_pos)]
43. cas_delta <- cas_delta[complete.cases(cas_delta)]
44. descdist(cas_delta_pos, boot = 1000)
45.
46. ```
47.
48.
49. 10MAY18
50. However! This poorly models the lower values, which are still important i.e. the range
   10-20 is still important.
51. Try evd on -ve and +ve values:
52.
53. ```{r}
54. cas_delta.gumbel.1 <- fitdist(cas_delta, "evd", method = "mge", start = list(location
   = 1, scale = 1), gof = "AD2R")
55. plot(cas_delta.gumbel.1)
56. cas_delta.gumbel.1
57. ```
58.
59.
60. ```{r}
61. result.7 <- data.frame(model.evd.1 = numeric(0))
62. for(i in 1:1000) {
```

```

63. model.evd.1 <- revd(16246, location = 1.387150, scale = 2.400736) # 16246 is complete
    cases
64. ecdf.model.evd.1 <- (1 - ecdf(model.evd.1)(20)) * 16246
65. result.7 <- rbind(result.7, ecdf.model.evd.1)
66. }
67. colnames(result.7) <- "samples"
68. quantile(result.7$samples, c(0.025, 0.975))
69. ``
70.
71. ``{r}
72. length(cas_delta[cas_delta>20])
73. ``
74.
75. The EVD models around 3 to 12 values > 20kt, which is slightly high, but acceptable.
76.
77. Now getting probabilities for intervals <10(a), 10-20(b), >20(c):
78.
79. ``{r}
80. a <- pevd(c(10,20), location = 1.296405, scale = 2.49085)
81. b <- a[2] - a[1]
82.
83. c <- 1- pevd(c(20), location = 1.296405, scale = 2.49085)
84.
85. a <- 1 - b + c
86.
87. a
88. b
89. c
90.
91.
92. ``

```