Prediction of Animal Strike on US Commercial Flights

Final Paper for the CEU MSc in Business Analytics program

Gábor Horváth

June 2017

CENTRAL EUROPEAN UNIVERSITY

1 Introduction

The structure of the document follows the Cross Industry Standard Process for Data Mining (CRISP-DM) process model, which is a non-proprietary, documented, and freely available data mining model (Shearer 2000). Whenever the model sections can be matched to (and can fulfill) the requirements stated by CEU for the Final Paper I'm using the appropriate section identified by the CRIPS-DM model. Please keep in mind that the model supports the full end-to-end process of a data mining project, but the project does not require the use of all the model elements.

2 Business Understanding

2.1 Determine Business Objectives

2.1.1 Business Objectives

There are two main objectives what the project is aiming to complete.

1. Create a statistical analysis to identify those reasons (based on the data available), which are determining the the risk of an animal strike for an airport.
2. Create a prediction model, which can be used to predict the risk of an animal strike for a given flight.

The result of the statistical analysis could be used in the completion of the model building and evaluation the recommended order of the completion is the order of the objectives stated above.

2.1.2 Business Success Criteria

- Identification of features determining the risk potential of an airport
- Working model for animal strike prediction

2.2 Assess Situation

2.2.1 Inventory of Resources

- Flight Data
- Animal Strike Data
- R
- Buckets

2.2.2 Requirements, Assumptions, and Constraints

- Additional Requirements:
  - No additional requirements identified on top of the requirements already stated in this document.
- Assumptions
  - No initial assumptions made.
- Constraints
  - No initial hard constraints identified.

2.2.3 Risks and Contingencies

- Risks
  - No initial risks identified
- Contingencies
  - No initial contingencies identified

2.2.4 Terminology

The project is using different terminologies from the different domains. The terms/definitions used will not be marked or explained in details, if based on the context the reader can easily identify the domain of the particular term. In case there are uncertainties about a term (and it's not explained in the paper), the following sources can be used for the definitions:

- Aviation:
  - Aviation Terms / Directory
  - Aviation Glossary
  - Aviation Glossaries
- Data Mining
  - Data Mining Glossary
  - Data Mining - Terminologies
  - Data Mining and Predictive Analytics Glossary
- Data Science / Big Data
  - Data Science Glossary
  - Analytics and Big Data Glossary
  - Data Science Glossary

2.2.5 Costs and Benefits

This is a one-man project, no significant cost is expected. Main benefit is to put to and almost end-to-end scenario the topics covered during the courses and discovering bits and bolts of the techniques for creating the project.

2.3 Determine Data Mining Goals

2.3.1 Data Mining Goals

- Understand, Analyse, Clean and Merge the source data correctly
- Create the required attributes
- Generate the required records (if applicable)

2.3.2 Data Mining Success Criteria

- Identification of featured determining the risk potential of an airport
- Working model for animal strike prediction

2.4 Produce Project Plan

2.4.1 Project Plan

The project is managed in an agile way, where all the tasks, requirements, issues, solutions, and ideas are kept in a project at buckets.

2.4.2 Initial Assessment of Tools and Techniques

- Programming language:
  - R: https://www.r-project.org/
- IDE for the programming language:
  - RStudio: https://www.rstudio.com/
- Documentation is created using:
  - knitr: https://yihui.name/knitr/
  - MiKTeX: https://miktex.org/
  - ReporteRs: https://cran.r-project.org/web/packages/ReporteRs/index.html
- Data visualization:
  - ggplot2: http://ggplot2.org/

- Data manipulation:
  - access2csv: https://github.com/AccelerationNet/access2csv
  - dtplyr: https://cran.r-project.org/web/packages/dtplyr/index.html
- Project plan / task management:
  - Buckets: https://www.buckets.co/
- Source code repository:
  - GitHub: https://github.com/

Note: The list above do not contain the list of all the tools and packages used to create the project, but the full list will be provided in the source code.

3 Data Understanding

3.1 Collect Initial Data

3.1.1 Initial Data Collection Report

There have been multiple data sources acquired in the initial phase of the project. These sources are the following:

3.1.1.1 Federal Aviation Administration

- Data source: Wildlife Strike Database
- The FAA provides the database as a compressed Microsoft Access file.
- The database version used is Version 2016.4-P (as of 24-10-2016).
- The database contains 180,177 Strike Reports from 1-1-1990 through 30-4-2016.
- The compressed file size is 44,730,852 bytes.
- The uncompressed Microsoft Access database file size is 193,495,040 bytes.
- The extracted tables are:
  - STRIKE_REPORTS (1990-1999) - 30082 rows - CSV size is 21,523,668 bytes
  - STRIKE_REPORTS (2000-2009) - 69960 rows - CSV size is 51,833,820 bytes.
  - STRIKE_REPORTS (2010-Current) - 70577 rows - CSV size is 53,973,874 bytes.
  - STRIKE_REPORTS_BASH (1990-Current).csv - 8046 rows - CSV size is 5,412,394 bytes.

3.1.1.2 United States Department of Transportation

- Data source: Bureau of Transportation Statistics - Flight performance
- The BTS provides the database as separate compressed CSV files. One file contains data of one month.
- The datestamp of the first CSV file available is 1-1-1987.
- The datestamp of the first data available is 1-10-1987.
- The datestamp of the last data acquired from BTS in the project is 31-12-2016.
- The number of files is 360.
  - Compressed size of the files is 6,196,385,360 bytes.
  - Uncompressed size of the files is 71,146,030,010 bytes.
- The download speed of the public access to these files seems to be limited, which needs to be taken into account in case of reproducing the results.

3.1.1.3 Federal Aviation Administration

- Data source: Airport Data & Contact Information
- The FAA provides the database as a tabulator separated csv file.
- The database used is as current as of 25-05-2017.
- The database used contains the details of 19,601 airport facilities.
- The file size is 10,490,580 bytes.

3.2 Describe Data

3.2.1 Data Description Report

The data sources have the following column explanations, which is attached to the downloaded files or can be downloaded separately, by the data provider agencies.

3.2.1.1 Animal Strike Data

Column name	Explanation of Column Name and Codes
INDEX NR	Individual record number
OPID	Airline operator code
OPERATOR	A three letter International Civil Aviation Organization code for aircraft operators. (BUS = business, PVT = private aircraft other than business, GOV = government aircraft, MIL - military aircraft.)
ATYPE	Aircraft
AMA	International Civil Aviation Organization code for Aircraft Make
AMO	International Civil Aviation Organization code for Aircraft Model
EMA	Engine Make Code (see Engine Codes tab below)
EMO	Engine Model Code (see Engine Codes tab below)
AC_CLASS	Type of aircraft (see Aircraft Type tab below)
AC_MASS	1 = 2,250 kg or less: 2 = ,2251-5700 kg: 3 = 5,701-27,000 kg: 4 = 27,001-272,000 kg: 5 = above 272,000 kg
NUM_ENGS	Number of engines
TYPE_ENG	Type of power A = reciprocating engine (piston): B = Turbojet: C = Turboprop: D = Turbofan: E = None (glider): F = Turboshaft (helicopter): Y = Other
ENG_1_POS	Where engine # 1 is mounted on aircraft (see Engine Position tab below)
ENG_2_POS	Where engine # 2 is mounted on aircraft (see Engine Position tab below)
ENG_3_POS	Where engine # 3 is mounted on aircraft (see Engine Position tab below)
ENG_4_POS	Where engine # 4 is mounted on aircraft (see Engine Position tab below)
REG	Aircraft registration
FLT	Flight number
REMAINS_COLLECTED	Indicates if bird or wildlife remains were found and collected
REMAINS_SENT	Indicates if remains were sent to the Smithsonian Institution for identification
INCIDENT_DATE	Date strike occurred
INCIDENT_MONTH	Month strike occurred
INCIDENT_YEAR	Year strike occurred
TIME_OF_DAY	Light conditions
TIME	Hour and minute in local time
AIRPORT_ID	International Civil Aviation Organization airport identifier for location of strike whether it was on or off airport
AIRPORT	Name of airport
STATE	State
FAAREGION	FAA Region where airport is located
ENROUTE	If strike did not occur on approach, climb, landing roll, taxi or take-off, aircraft was enroute. This shows location.
RUNWAY	Runway
LOCATION	Various information about aircraft location if enroute or airport where strike evidence was found. Some locations show the two airports for the flight departure and arrival if pilot was unaware of the strike.
HEIGHT	Feet Above Ground Level
SPEED	Knots (indicated air speed)
DISTANCE	Miles from airport
PHASE_OF_FLT	Phase of flight during which strike occurred

Column name	Explanation of Column Name and Codes
DAMAGE	Blank - Unknown; M = minor - When the aircraft can be rendered airworthy by simple repairs or replacements and an extensive inspection is not necessary.; M? = uncertain level - The aircraft was damaged, but details as to the extent of the damage are lacking.; S = substantial - When the aircraft incurs damage or structural failure which adversely affects the structure strength, performance or flight characteristics of the aircraft and which would normally require major repair or replacement of the affected component.; D = Destroyed - When the damage sustained makes it inadvisable to restore the aircraft to an airworthy condition.
STR_RAD	Struck radome
DAM_RAD	Damaged radome
STR_WINDSHLD	Struck windshield
DAM_WINDSHLD	Damaged windshield
STR_NOSE	Struck nose
DAM_NOSE	Damaged nose
STR_ENG1	Struck Engine 1
DAM_ENG1	Damaged Engine 1
STR_ENG2	Struck Engine 2
DAM_ENG2	Damaged Engine 2
STR_ENG3	Struck Engine 3
DAM_ENG3	Damaged Engine 3
STR_ENG4	Struck Engine 4
DAM_ENG4	Damaged Engine 4
INGESTED	Engine ingested the bird/ animal
STR_PROP	Struck Propeller
DAM_PROP	Damaged Propeller
STR_WING_ROT	Struck Wing or Rotor
DAM_WING_ROT	Damaged Wing or Rotor
STR_FUSE	Struck Fuselage
DAM_FUSE	Damaged Fuselage
STR_LG	Struck Landing Gear
DAM_LG	Damaged Landing Gear
STR_TAIL	Struck Tail
DAM_TAIL	Damaged Tail
STR_LGHTS	Struck Lights
DAM_LGHTS	Damaged Lights
STR_OTHER	Struck Other than parts shown above
DAM_OTHER	Damaged Other than parts shown above
OTHER_SPECIFY	What part was struck other than those listed above
EFFECT	Effect on flight
EFFECT_OTHER	Effect on flight other than those listed on the form
SKY	Type of cloud cover, if any
PRECIP	Precipitation
SPECIES_ID	International Civil Aviation Organization code for type of bird or other wildlife
SPECIES	Common name for bird or other wildlife
BIRDS_SEEN	Number of birds/wildlife seen by pilot
BIRDS_STRUCK	Number of birds/wildlife struck
SIZE	Size of bird as reported by pilot is a relative scale. Entry should reflect the perceived size as opposed to a scientifically determined value. If more than one species was struck, larger bird is entered.
WARNED	Pilot warned of birds/wildlife

Column name	Explanation of Column Name and Codes
COMMENTS	As entered by database manager. Can include name of aircraft owner, types of reports received, updates, etc.
REMARKS	Most of remarks are from the form but some are data entry notes and are usually in parentheses.
AOS	Time aircraft was out of service in hours. If unknown, it is blank.
COST_REPAIRS	Estimated cost of repairs of replacement in dollars (USD)
COST_OTHER	Estimated other costs, other than those in previous field in dollars (USD). May include loss of revenue, hotel expenses due to flight cancellation, costs of fuel dumped, etc.
COST_REPAIRS_INFL_ADJ	Costs adjusted for inflation
COST_OTHER_INFL_ADJ	Other cost adjusted for inflation
REPORTED_NAME	Name(s) of person(s) filing report
REPORTED_TITLE	Title(s) of person(s) filing report
REPORTED_DATE	Date report was written
SOURCE	Type of report. Note: for multiple types of reports this will be indicated as Multiple. See “Comments” field for details
PERSON	Only one selection allowed. For multiple reports, see field “Reported Title”
NR_INJURIES	Number of people injured
NR_FATALITIES	Number of human fatalities
LUPDATE	Last time record was updated
TRANSFER	Unused field at this time
INDICATED_DAMAGE	Indicates whether or not aircraft was damaged

3.2.1.2 Flight Data

Column name	Explanation of Column Name and Codes
Year	Year
Quarter	Quarter (1-4)
Month	Month
DayofMonth	Day of Month
DayOfWeek	Day of Week
FlightDate	Flight Date (yyyymmdd)
UniqueCarrier	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.
AirlineID	An identification number assigned by US DOT to identify a unique airline (carrier). A unique airline (carrier) is defined as one holding and reporting under the same DOT certificate regardless of its Code, Name, or holding company/corporation.
Carrier	Code assigned by IATA and commonly used to identify a carrier. As the same code may have been assigned to different carriers over time, the code is not always unique. For analysis, use the Unique Carrier Code.
TailNum	Tail Number
FlightNum	Flight Number
OriginAirportID	Origin Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
OriginAirportSeqID	Origin Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.

Column name	Explanation of Column Name and Codes
OriginCityMarketID	Origin Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
Origin	Origin Airport
OriginCityName	Origin Airport, City Name
OriginState	Origin Airport, State Code
OriginStateFips	Origin Airport, State Fips
OriginStateName	Origin Airport, State Name
OriginWac	Origin Airport, World Area Code
DestAirportID	Destination Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
DestAirportSeqID	Destination Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.
DestCityMarketID	Destination Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
Dest	Destination Airport
DestCityName	Destination Airport, City Name
DestState	Destination Airport, State Code
DestStateFips	Destination Airport, State Fips
DestStateName	Destination Airport, State Name
DestWac	Destination Airport, World Area Code
CRSDepTime	CRS Departure Time (local time: hhmm)
DepTime	Actual Departure Time (local time: hhmm)
DepDelay	Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
DepDelayMinutes	Difference in minutes between scheduled and actual departure time. Early departures set to 0.
DepDel15	Departure Delay Indicator, 15 Minutes or More (1=Yes)
DepartureDelayGroups	Departure Delay intervals, every (15 minutes from <-15 to >180)
DepTimeBlk	CRS Departure Time Block, Hourly Intervals
TaxiOut	Taxi Out Time, in Minutes
WheelsOff	Wheels Off Time (local time: hhmm)
WheelsOn	Wheels On Time (local time: hhmm)
TaxiIn	Taxi In Time, in Minutes
CRSArrTime	CRS Arrival Time (local time: hhmm)
ArrTime	Actual Arrival Time (local time: hhmm)
ArrDelay	Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
ArrDelayMinutes	Difference in minutes between scheduled and actual arrival time. Early arrivals set to 0.
ArrDel15	Arrival Delay Indicator, 15 Minutes or More (1=Yes)
ArrivalDelayGroups	Arrival Delay intervals, every (15-minutes from <-15 to >180)
ArrTimeBlk	CRS Arrival Time Block, Hourly Intervals
Cancelled	Cancelled Flight Indicator (1=Yes)
CancellationCode	Specifies The Reason For Cancellation
Diverted	Diverted Flight Indicator (1=Yes)
CRSElapsedTime	CRS Elapsed Time of Flight, in Minutes
ActualElapsedTime	Elapsed Time of Flight, in Minutes
AirTime	Flight Time, in Minutes

Column name	Explanation of Column Name and Codes
Flights	Number of Flights
Distance	Distance between airports (miles)
DistanceGroup	Distance Intervals, every 250 Miles, for Flight Segment
CarrierDelay	Carrier Delay, in Minutes
WeatherDelay	Weather Delay, in Minutes
NASDelay	National Air System Delay, in Minutes
SecurityDelay	Security Delay, in Minutes
LateAircraftDelay	Late Aircraft Delay, in Minutes
FirstDepTime	First Gate Departure Time at Origin Airport
TotalAddGTime	Total Ground Time Away from Gate for Gate Return or Cancelled Flight
LongestAddGTime	Longest Time Away from Gate for Gate Return or Cancelled Flight
DivAirportLandings	Number of Diverted Airport Landings
DivReachedDest	Diverted Flight Reaching Scheduled Destination Indicator (1=Yes)
DivActualElapsedTime	Elapsed Time of Diverted Flight Reaching Scheduled Destination, in Minutes. The ActualElapsedTime column remains NULL for all diverted flights.
DivArrDelay	Difference in minutes between scheduled and actual arrival time for a diverted flight reaching scheduled destination. The ArrDelay column remains NULL for all diverted flights.
DivDistance	Distance between scheduled destination and final diverted airport (miles). Value will be 0 for diverted flight reaching scheduled destination.
Div1Airport	Diverted Airport Code1
Div1AirportID	Airport ID of Diverted Airport 1. Airport ID is a Unique Key for an Airport
Div1AirportSeqID	Airport Sequence ID of Diverted Airport 1. Unique Key for Time Specific Information for an Airport
Div1WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code1
Div1TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code1
Div1LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code1
Div1WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code1
Div1TailNum	Aircraft Tail Number for Diverted Airport Code1
Div2Airport	Diverted Airport Code2
Div2AirportID	Airport ID of Diverted Airport 2. Airport ID is a Unique Key for an Airport
Div2AirportSeqID	Airport Sequence ID of Diverted Airport 2. Unique Key for Time Specific Information for an Airport
Div2WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code2
Div2TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code2
Div2LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code2
Div2WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code2
Div2TailNum	Aircraft Tail Number for Diverted Airport Code2
Div3Airport	Diverted Airport Code3
Div3AirportID	Airport ID of Diverted Airport 3. Airport ID is a Unique Key for an Airport
Div3AirportSeqID	Airport Sequence ID of Diverted Airport 3. Unique Key for Time Specific Information for an Airport
Div3WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code3
Div3TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code3
Div3LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code3
Div3WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code3
Div3TailNum	Aircraft Tail Number for Diverted Airport Code3
Div4Airport	Diverted Airport Code4
Div4AirportID	Airport ID of Diverted Airport 4. Airport ID is a Unique Key for an Airport
Div4AirportSeqID	Airport Sequence ID of Diverted Airport 4. Unique Key for Time Specific Information for an Airport
Div4WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code4

Column name	Explanation of Column Name and Codes
Div4TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code4
Div4LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code4
Div4WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code4
Div4TailNum	Aircraft Tail Number for Diverted Airport Code4
Div5Airport	Diverted Airport Code5
Div5AirportID	Airport ID of Diverted Airport 5. Airport ID is a Unique Key for an Airport
Div5AirportSeqID	Airport Sequence ID of Diverted Airport 5. Unique Key for Time Specific Information for an Airport
Div5WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code5
Div5TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code5
Div5LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code5
Div5WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code5
Div5TailNum	Aircraft Tail Number for Diverted Airport Code5

3.2.1.3 Airport Data

Column name	Explanation of Column Name and Codes
SiteNumber	Landing facility site number - a unique identifying number which, together with the landing facility type code, forms the key to the airport record. (ex. 04508.*A)
Type	Landing facility type. (ex. Airport, Balloonport, Seaplane Base, Gliderport, Heliport, Stolport, Ultralight)
LocationID	Location identifier unique 3-4 character alphanumeric identifier assigned to the landing facility. (ex. 'ORD' for Chicago O'Hare)
EffectiveDate	Information effective date (mm/dd/yyyy). This date coincides with the 56-day charting and publication cycle date.
Region	FAA region code. (ex. AAL - Alaska, ACE - Central, AEA - Eastern, AGL - Great Lakes, AIN - International, ANE - New England, ANM - Northwest Mountain, ASO - Southern, ASW - Southwest, AWP - Western-Pacific)
DistrictOffice	FAA district or field office code. (ex. CHI)
State	Associated state post office code standard two letter abbreviation for u.s. states and territories. (ex. IL, PR, CQ)
StateName	Associated state name. (ex. Illinois)
County	Associated county (or parish) name. (ex. Cook)
CountyState	Associated county's state (post office code) state where the associated county is located; may not be the same as the associated city's state code. (ex. IL)
City	Associated city name. (ex. Chicago)
FacilityName	Official facility name. (ex. Chicago O'Hare Intl)
Ownership	Airport ownership type. (ex. PU - publicly owned, PR - privately owned, MA - air force owned, MN - navy owned, MR - army owned)
Use	Facility use. (ex. PU - open to the public, PR - private)
Owner	Facility owner's name.
OwnerAddress	Owner's address.
OwnerCSZ	Owner's city, state and zip code.
OwnerPhone	Owner's phone number. (data formats: nnn-nnn-nnnn (area code + phone number), 1-nnn-nnnn (dial 1-800 then number), 8-nnn-nnnn (dial 800 then number)
Manager	Facility manager's name.
ManagerAddress	Manager's address.

Column name	Explanation of Column Name and Codes
ManagerCSZ	Manager’s city, state and zip code.
ManagerPhone	Manager’s phone number. (data formats: nnn-nnn-nnnn (area code + phone number), 1-nnn-nnnn (dial 1-800 then number), 8-nnn-nnnn (dial 800 then number)
ARPLatitude	Airport reference point latitude (formatted).
ARPLatitudeS	Airport reference point latitude (seconds).
ARPLongitude	Airport reference point longitude (formatted).
ARPLongitudeS	Airport reference point longitude (seconds).
ARPMethod	Airport reference point determination method. (ex. E - estimated, S - surveyed)
ARPElevation	Airport elevation (nearest foot MSL). Elevation is measured at the highest point on the centerline of the usable landing surface. (ex. 1200; -10 for 10 feet below sea level)
ARPElevationMethod	Airport elevation determination method. (ex. E - estimated, S - surveyed)
MagneticVariation	Magnetic variation and direction magnetic variation to nearest degree. (ex. 03W)
MagneticVariationYear	Magnetic variation epoch year. (ex. 1985)
TrafficPatternAltitude	Traffic pattern altitude (whole feet AGL). (ex. 1000)
ChartName	Aeronautical sectional chart on which facility appears. (ex. Washington)
DistanceFromCBD	Distance from central business district of the associated city to the airport (nearest nautical mile - ex. 08).
DirectionFromCBD	Direction of airport from central business district of associated city (nearest 1/8 compass point - ex. NE).
LandAreaCoveredByAirport	Amount of land owned by the airport in acres.
BoundaryARTCCID	Boundary ARTCC Identifier. The boundary ARTCC is the FAA air route traffic control center within whose published boundaries the airport lies. It may not be the controlling ARTCC for the airport if a letter of agreement exists between the boundary ARTCC and another ARTCC. (ex. ZDC for Washington ARTCC)
BoundaryARTCCComputerID	Boundary ARTCC (FAA) computer identifier. (ex. ZCW for Washington ARTCC)
BoundaryARTCCName	Boundary ARTCC name. (ex. Washington)
ResponsibleARTCCID	Responsible ARTCC identifier the responsible ARTCC is the FAA air route traffic control center who has assumed control over the airport through a letter of agreement with the boundary ARTCC. (ex. ZDC for Washington ARTCC)
ResponsibleARTCCComputerID	Responsible ARTCC (FAA) computer identifier. (ex. ZCW for Washington ARTCC)
ResponsibleARTCCName	Responsible ARTCC name. (ex. Washington)
TieInFSS	Tie-in FSS physically located on facility. (ex. Y - tie-in FSS is on the airport, n - tie-in FSS is not on the airport)
TieInFSSID	Tie-in flight service station (FSS) identifier. (ex. DCA for Washington FSS)
TieInFSSName	Tie-in FSS name. (ex. Washington)
AirportToFSSPhoneNumber	Local phone number from airport to FSS for adminstrative services

Column name	Explanation of Column Name and Codes
TieInFSSTollFreeNumber	Toll free phone number from airport to FSS for pilot briefing services the data describes the type of toll-free communications and the number to dial. The data formats and their meanings are: 1-nnn-nnnn, dial 1-800- then nnn-nnnn; 8-nnn-nnnn, dial 800 then nnn-nnnn; e-nnnnnnnn, enterprise number dial 0 & ask for enterprise nnnnnnnn; lcnnn-nnnn, local call - dial nnn-nnnn; dl, direct line telephone at the airport - no dialing required; z-nnnnnnnn, zenith number - dial 0 and ask for zenith nnnnnnnn; w-nnnnnnnn, dial 0 and ask for wx nnnnnnnn; c-nnnnnnnn, dial 0 and ask for commerce nnnnnnnn; ld-nnnnnnnn, long distance call - dial (area code) then nnnnnnn; lt-nnnnnnnn, long distal call dial 1-nnnnnnn; 1-wx-brief, dial 1-800-wx-brief; 8-wx-brief, dial 800-wx-brief
AlternateFSSID	Alternate FSS identifier provides the identifier of a full-time flight service station that assumes responsibility for the airport during the off hours of a part-time primary FSS. (ex. ‘DCA’ for Washington FSS)
AlternateFSSName	Alternate FSS name. (ex. ‘Washington’ for Washington FSS)
AlternateFSSTollFreeNumber	Toll free phone number from airport to FSS for pilot briefing services the data describes the type of toll-free communications and the number to dial. The data formats and their meanings are: 1-nnn-nnnn, dial 1-800- then nnn-nnnn; 8-nnn-nnnn, dial 800 then nnn-nnnn; e-nnnnnnnn, enterprise number dial 0 & ask for enterprise nnnnnnnn; lcnnn-nnnn, local call - dial nnn-nnnn; dl, direct line telephone at the airport - no dialing required; z-nnnnnnnn, zenith number - dial 0 and ask for zenith nnnnnnnn; w-nnnnnnnn, dial 0 and ask for wx nnnnnnnn; c-nnnnnnnn, dial 0 and ask for commerce nnnnnnnn; ld-nnnnnnnn, long distance call - dial (area code) then nnnnnnn; lt-nnnnnnnn, long distal call dial 1-nnnnnnn; 1-wx-brief, dial 1-800-wx-brief; 8-wx-brief, dial 800-wx-brief.
NOTAMFacilityID	Identifier of the facility responsible for issuing notices to airmen (NOTAMS) and weather information for the airport. (ex. ORD)
NOTAMService	Availability of NOTAM ‘d’ service at airport. (ex. Y - yes, N - no)
ActivationDate	Airport activation date (mm/yyyy). Provides the month and year that the facility was added to the NFDC airport database. Note: this information is only available for those facilities opened since 1981. (ex. 06/1981)
AirportStatusCode	Airport status code: CI - closed indefinitely; CP - closed permanently; O - operational
CertificationTypeDate	Airport certification type and date. Format is the class code (‘I’, ‘II’, ‘III’ or ‘IV’) followed by a one characther code A, B, C, D, E, or L, followed by a one character code S or U, followed by the month and year of certification. (ex. ‘I A S 07/1980’, ‘I C S 01/1983’ or ‘I A U 09/1983’). Codes A, B, C, D, and E are for airports having a full certificate under CFR Part 139, and receiving scheduled air carrier service from carriers certificated by the Civil Aeronautics Board. The A, B, C, D, and E identify the aircraft rescue and firefighting index for the airport. Code L is for airports having limited certification under CFR Part 139. Code S is for Airports receiving scheduled air carrier service from carriers certificated by the Civil Aeronautics Board. Code U is for airports not receiving this scheduled service.

Column name	Explanation of Column Name and Codes
FederalAgreements	NPIAS/Federal Agreement Code. A combination of 1 to 7 codes that indicate the type of federal agreements existing at the airport. (ex. NGH). N - national plan of integrated airport systems (NPIAS); B - installation of navigational facilities on privately owned airports under F&E program; G - grant agreements under FAAP/ADAP/AIP; H - compliance with accessibility to the handicapped; P - surplus property agreement under Public Law 289; R - surplus property agreement under Regulation 16-WAA; S - conveyance under section 16, Federal Airport Act of 1946 or Section 23, Airport and Airway Development Act of 1970; V - advance planning agreement under FAAP; X - obligations assumed by transfer; Y - assurances pursuant to Title VI, Civil Rights Act of 1964; Z - conveyance under Section 303(C), Federal Aviation Act of 1958; 1 - grant agreement has expired, however, agreement remains in effect for this facility as long as it is public use.
AirspaceDetermination	Airport airspace analysis determination. (ex. CONDL (conditional), NOT ANALYZED, NO OBJECTION, OBJECTIONABLE)
CustomsAirportOfEntry	Facility has been designated by the U.S. Treasury as an international airport of entry for customs (ex. Y - yes, N - no)
CustomsLandingRights	Facility has been designated by the U.S. Treasury as a customs landing rights airport (ex. Y - yes, N - no)
MilitaryJointUse	Facility has military/civil joint use agreement that allows civil operations at a military airport or military operations at a civil airport (ex. Y - yes, N - no)
MilitaryLandingRights	Airport has entered into an agreement that grants landing rights to the military (ex. Y - yes, N - no)
InspectionMethod	Airport inspection method. (ex. F - federal, S - state, C - contractor, 1 - 5010-1 public use mail out program, 2 - 5010-2 private use mail out program)
InspectionGroup	Agency/group performing physical inspection (ex. F - faa airports field personnel, s - state aeronautical personnel, c - private contract personnel, n - owner)
LastInspectionDate	Last physical inspection date (mmddyyyy)
LastOwnerInformationDate	Last date information request was completed by facility owner or manager (mmddyyyy)
FuelTypes	Fuel types available for public use at the airport. There can be up to 8 occurrences of a fixed 5 character field (ex. 80___100__100LL115__). 80 - grade 80 gasoline (red), 100 - grade 100 gasoline (green), 100LL - grade 100LL gasoline (low lead blue), 115 - grade 115 gasoline, A - jet A - kerosene, freeze point -40C, A1 - jet A-1 - kerosene, freeze point -50C, A1+ - jet A-1 - kerosene, with icing inhibitor freeze point -50C, B - jet B - wide-cut turbine fuel, freeze point -50C, B+ - jet B - wide-cut turbine fuel with icing inhibitor, freeze point -50C, MOGAS - automotive gasoline.
AirframeRepair	Airframe repair service availability/type. (ex. MAJOR, MINOR, NONE)
PowerPlantRepair	Power plant (engine) repair availability/type. (ex. MAJOR, MINOR, NONE)
BottledOxygenType	Type of bottled oxygen available (value represents high and/or low pressure replacement bottle). (ex. HIGH, LOW, HIGH/LOW, NONE)
BulkOxygenType	Type of bulk oxygen available (value represents high and/or low pressure cylinders). (ex. HIGH, LOW, HIGH/LOW, NONE)
LightingSchedule	Airport lighting schedule value is the beginning-ending times (local time) that lights are operated. Format can be 1900-2300, DUSK-0100, ALL, DUSK-DAWN, NONE, etc.

Column name	Explanation of Column Name and Codes
BeaconSchedule	Beacon lighting schedule value is the beginning-ending times (local time) that the rotating airport beacon light is operated. Value can be "SS-SR" (indicating sunset-sunrise), blank, or "SEE RMK", indicating that the details are in a facility remark data entry.
ATCT	Air traffic control tower located on airport. (ex. Y - yes, N - no)
UNICOMFrequencies	Unicom frequencies available at the airport there can be up to 6 occurrences of a fixed 7 character field. (ex. 122.700 or 122.700122.800 or NONE)
CTAFFrequency	Common traffic advisory frequency. (CTAF) (ex. 122.800)
SegmentedCircle	Segmented circle airport marker system on the airport. (ex. Y - yes, N - no, none)
BeaconColor	Lens color of operable beacon located on the airport. (ex. CG - clear-green (lighted land airport); CY - clear-yellow (lighted seaplane base); CGY - clear-green-yellow (heliport); SCG - split-clear-green (lighted military airport); C - clear (unlighted la
NonCommercialLandingFee	Landing fee charged to non-commercial users of airport. (ex. Y - yes, N - no)
MedicalUse	Landing facility is used for medical purposes. (ex. Y - yes, N - no)
SingleEngineGA	Number of single engine general aviation aircraft.
MultiEngineGA	Number of multi engine general aviation aircraft.
JetEngineGA	Number of jet engine general aviation aircraft.
HelicoptersGA	Number of general aviation helicopter.
GlidersOperational	Number of operational gliders.
MilitaryOperational	Number operational military aircraft (includingg helicopters).
Ultralights	Number of ultralight aircraft.
OperationsCommercial	Commercial services. Scheduled operations by cab-certificated carriers or intrastate carriers.
OperationsCommuter	Commuter services. Scheduled commuter and cargo carriers.
OperationsAirTaxi	Air taxi. Air taxi operators carrying passengers, mail, or mail for revenue.
OperationsGALocal	General aviation local operations. Those operating in the local traffic pattern or within a 20-mile radius of the airport.
OperationsGAItin	General aviation itinerant operations. Those general aviation operations (excluding commuter or air taxi) not qualifying as local.
OperationsMilitary	Military aircraft operations.
OperationsDate	12-month ending date on which annual operations data in above six field is based (mm/dd/yyyy).
AirportPositionSource	Airport position source.
AirportPositionSourceDate	Airport position source date (mm/dd/yyyy).
AirportElevationSource	Airport elevation source.
AirportElevationSourceDate	Airport elevation source date (mm/dd/yyyy).
ContractFuelAvailable	Contract fuel available. (ex. Y - yes, N - no)
TransientStorage	Transient storage. (ex. Y - yes, N - no, none)
OtherServices	Other services. (ex. Y - yes, N - no, none)
WindIndicator	Wind direction indicator. (ex. Y - yes, N - no, none)
IcaoIdentifier	International coding for airport.

3.3 Explore Data

3.3.1 Data Exploration Report

Keeping the length of this section reasonable, the exploration report shown here contains the data from 1990. The report for the rest of the data is in the appendix of the final document.

3.3.1.1 Animal Strike Data

The first summary table shows the number of distinct items for each year regarding the Airline operators, Aircraft, Aircraft types, Aircraft mass types, and Engine types, which have been reported as being affected in an animal strike.

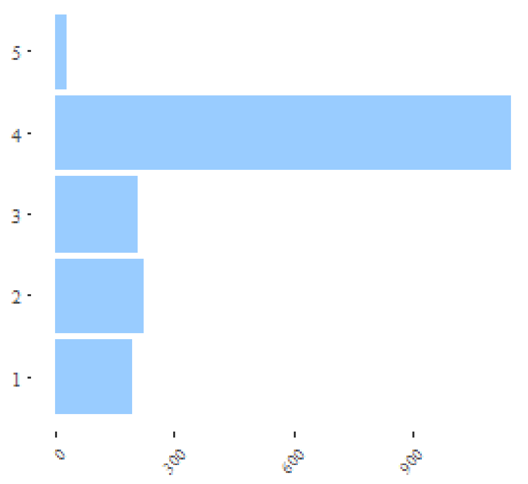
Year	# of reports	Operators	Aircraft	Aircraft type	Aircraft mass type	Engine type
1990	1847	316	329	4	5	9

The second summary table shows the number of distinct items for each year regarding the Time of day, Airports, States, Phase of flight, weather conditions (Sky and Precipitation), and the flag for showing if the pilot has been warned or not about birds / wildlife in the reports.

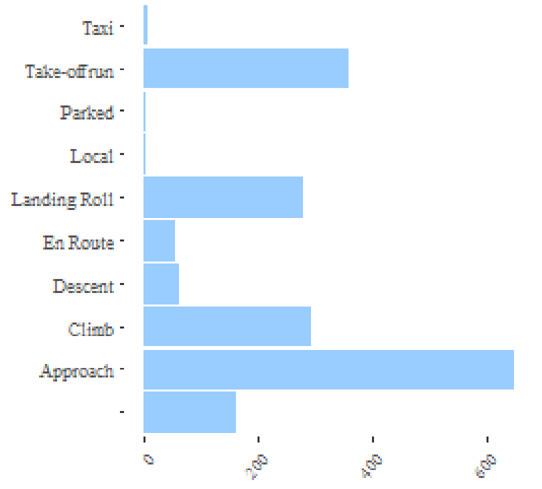
Year	Time of day	Airports	States	Phase of flight	Sky	Precipitation	Warned
1990	5	1175	61	12	7	8	4

The following graphs show the distributions of some of the selected distinct items summarized in the tables above.

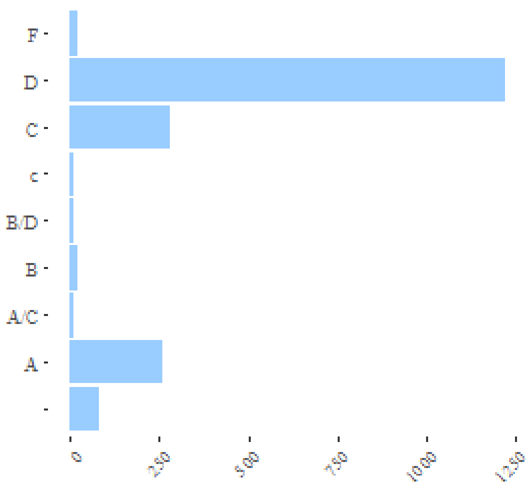
Data distribution of aircraft mass type in 1990



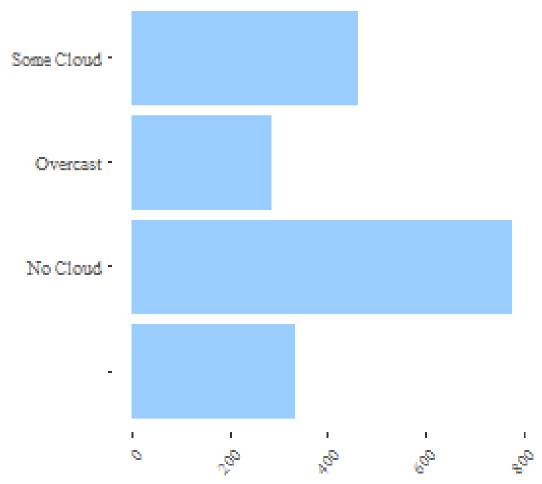
Data distribution of flight phase in 1990



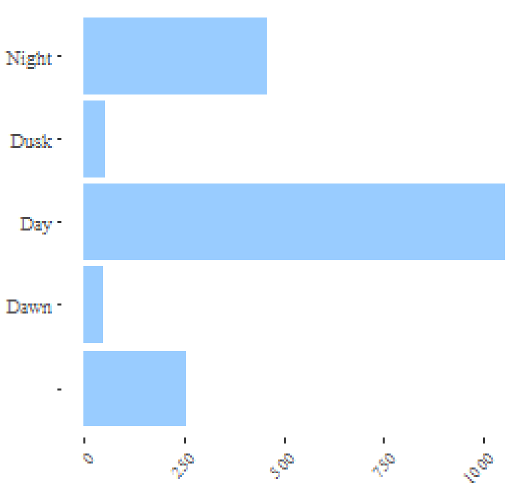
Data distribution of engine type in 1990



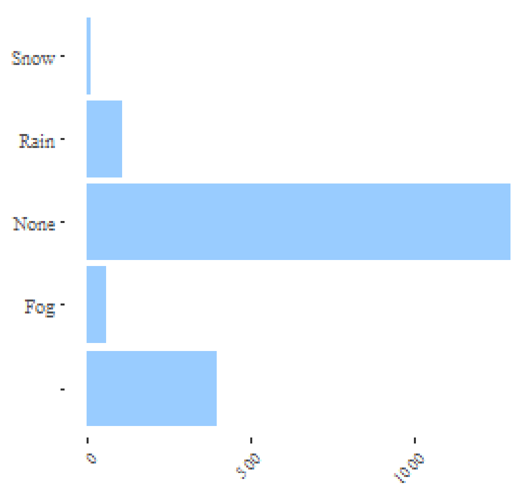
Data distribution of sky condition in 1990



Data distribution of time of day in 1990



Data distribution of precipitation in 1990



3.3.1.2 Flight Data

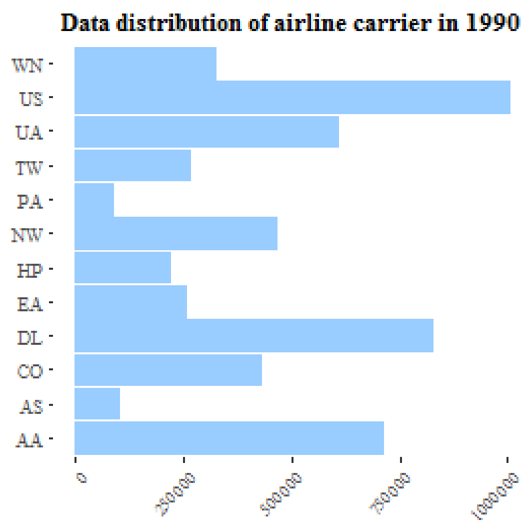
The first summary table shows the number of distinct items for each year regarding the number of records, the carriers, and the origin and the destination airports.

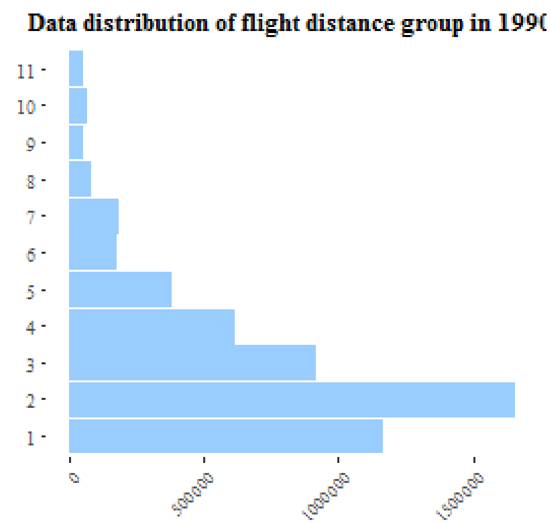
Year	# of flights	# of carriers	Origin airports	Origin states	Destination airports	Destination states
1990	5270893	12	235	53	236	53

The second summary table shows the number of distinct items for each year the departure time group and distance between the airports.

Year	Departure time block	Distance group
1990	19	11

The following graphs show the distributions of some of the selected distinct items summarized in the tables above.





3.3.1.3 Airport Data

The airport data is used only as a reference data source, therefore no data exploration needs to be executed.

3.4 Verify Data Quality

3.4.1 Data Quality Report

3.4.1.1 Animal Strike Data

The data set provided by the Federal Aviation Administration is a data set based on voluntary strike reporting from airlines, airports, pilots, and other sources. Therefore the quality of the data enormously depends on the goodwill of the reporting source and even with the best intentions there are several quality issues which needs to be addressed later in the project.

- Mixed use of uppercase and lowercase letters/codes
- Mixed use of codes (e.g.: engine type is defined as “A/C”)
- Number of States in the data set is above the actual number of states of the U.S.

The Federal Aviation Administration provides code books for some of the data details in the strike reports. Based on these code books the records with the following values can be removed from the data set, as they are not relevant for the goals of the project.

Column name	Value	Reason for removal
OPID	“PVT”	Record is related of a strike to a privately owned aircraft, not to an aircraft operated by a commercial airline.
OPID	“BUS”	Record is related of a strike to a business aircraft, not to an aircraft operated by a commercial airline.
OPID	“GOV”	Record is related of a strike to a government aircraft, not to an aircraft operated by a commercial airline.
OPID	“MIL”	Record is related of a strike to a military aircraft, not to an aircraft operated by a commercial airline.
OPID	“UNKC”	Record is related of a strike to an aircraft of an unknown commercial operator. Without this information identification of the flight can’t be done correctly.
OPID	“UNK”	Record is related of a strike to an aircraft of an unknown operator. Without this information identification of the flight can’t be done correctly.
AC_CLASS	“B”	Value stands for helicopter.
AC_CLASS	“C”	Value stands for glider.
AC_CLASS	“D”	Value stands for balloon.
AC_CLASS	“F”	Value stands for dirigible.
AC_CLASS	“I”	Value stands for gyroplane.
AC_CLASS	“J”	Value stands for ultralight.
AC_CLASS	“Y”	Value stands for other.
AC_CLASS	“Z”	Value stands for unknown.
AC_CLASS	“”	Value is empty.
TYPE_ENG	“E”	Value stands for none (glider).
TYPE_ENG	“F”	Value stands for turboshaft (helicopter).
TYPE_ENG	“”	Value is empty.

The strike report itself contains a great deal of details, which can be used in different projects, but for my purposes the following details have to be removed to concentrate on those information, which I expect to be the cause and not the effect of the strike. The following details needs to be removed from the data set in a later stage.

Column name	Explanation of Column Name and Codes
AMA	International Civil Aviation Organization code for Aircraft Make
AMO	International Civil Aviation Organization code for Aircraft Model
EMA	Engine Make Code
EMO	Engine Model Code
NUM_ENGS	Number of engines
ENG_1_POS	Where engine # 1 is mounted on aircraft
ENG_2_POS	Where engine # 2 is mounted on aircraft
ENG_3_POS	Where engine # 3 is mounted on aircraft
ENG_4_POS	Where engine # 4 is mounted on aircraft
REMAINS_COLLECTED	Indicates if bird or wildlife remains were found and collected
REMAINS_SENT	Indicates if remains were sent to the Smithsonian Institution for identification
LOCATION	Various information about aircraft location if enroute or airport where strike evidence was found. Some locations show the two airports for the flight departure and arrival if pilot was unaware of the strike.
DAMAGE	Amount of the damage.
STR_RAD	Struck radome
DAM_RAD	Damaged radome
STR_WINDSHLD	Struck windshield
DAM_WINDSHLD	Damaged windshield
STR_NOSE	Struck nose
DAM_NOSE	Damaged nose
STR_ENG1	Struck Engine 1
DAM_ENG1	Damaged Engine 1
STR_ENG2	Struck Engine 2
DAM_ENG2	Damaged Engine 2
STR_ENG3	Struck Engine 3
DAM_ENG3	Damaged Engine 3
STR_ENG4	Struck Engine 4
DAM_ENG4	Damaged Engine 4
INGESTED	Engine ingested the bird/ animal
STR_PROP	Struck Propeller
DAM_PROP	Damaged Propeller
STR_WING_ROT	Struck Wing or Rotor
DAM_WING_ROT	Damaged Wing or Rotor
STR_FUSE	Struck Fuselage
DAM_FUSE	Damaged Fuselage
STR_LG	Struck Landing Gear
DAM_LG	Damaged Landing Gear
STR_TAIL	Struck Tail
DAM_TAIL	Damaged Tail
STR_LGHTS	Struck Lights
DAM_LGHTS	Damaged Lights
STR_OTHER	Struck Other than parts shown above
DAM_OTHER	Damaged Other than parts shown above
OTHER_SPECIFY	What part was struck other than those listed above
EFFECT	Effect on flight
EFFECT_OTHER	Effect on flight other than those listed on the form
SPECIES_ID	International Civil Aviation Organization code for type of bird or other wildlife
SPECIES	Common name for bird or other wildlife
BIRDS_SEEN	Number of birds/wildlife seen by pilot

Column name	Explanation of Column Name and Codes
BIRDS_STRUCK	Number of birds/wildlife struck
SIZE	Size of bird as reported by pilot is a relative scale. Entry should reflect the perceived size as opposed to a scientifically determined value. If more than one species was struck, larger bird is entered.
COMMENTS	As entered by database manager. Can include name of aircraft owner, types of reports received, updates, etc.
REMARKS	Most of remarks are from the form but some are data entry notes and are usually in parentheses.
AOS	Time aircraft was out of service in hours. If unknown, it is blank.
COST_REPAIRS	Estimated cost of repairs of replacement in dollars (USD)
COST_OTHER	Estimated other costs, other than those in previous field in dollars (USD). May include loss of revenue, hotel expenses due to flight cancellation, costs of fuel dumped, etc.
COST_REPAIRS_INFL_ADJ	Costs adjusted for inflation
COST_OTHER_INFL_ADJ	Other cost adjusted for inflation
REPORTED_NAME	Name(s) of person(s) filing report
REPORTED_TITLE	Title(s) of person(s) filing report
REPORTED_DATE	Date report was written
SOURCE	Type of report. Note: for multiple types of reports this will be indicated as Multiple. See “Comments” field for details
PERSON	Only one selection allowed. For multiple reports, see field “Reported Title”
NR_INJURIES	Number of people injured
NR_FATALITIES	Number of human fatalities
LUPDATE	Last time record was updated
TRANSFER	Unused field at this time
INDICATED_DAMAGE	Indicates whether or not aircraft was damaged

3.4.1.2 Flight Data

The data set provided by the United States Department of Transportation is a data set based on the timetable and the actual flight information collected by various systems. Therefore the quality of the data is significantly better than the data from the Federal Aviation Administration Animal Strike Database, but there are still some possible quality issues which needs to be addressed later in the project after further investigation. These issues include:

- Number of States in the data set is above the actual number of states of the U.S.

The data in the Federal Aviation Administration Animal Strike Database is available only until 30-4-2016, so the flight data needs to be adjusted accordingly.

Similarly to the Federal Aviation Administration Animal Strike Database, the flight performance data set contains a great deal of details as well, which can be used in different projects, but for my purposes the following details have to be removed to concentrate on those information, which I expect to be the cause and not the effect of the strike. The following details needs to be removed from the data set in a later stage.

Column name	Explanation of Column Name and Codes
AirlineID	An identification number assigned by US DOT to identify a unique airline (carrier). A unique airline (carrier) is defined as one holding and reporting under the same DOT certificate regardless of its Code, Name, or holding company/corporation.
TailNum	Tail Number
OriginAirportID	Origin Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.

Column name	Explanation of Column Name and Codes
OriginAirportSeqID	Origin Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.
OriginCityMarketID	Origin Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
OriginStateFips	Origin Airport, State Fips
OriginWac	Origin Airport, World Area Code
DestAirportID	Destination Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
DestAirportSeqID	Destination Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.
DestCityMarketID	Destination Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
DestStateFips	Destination Airport, State Fips
DestWac	Destination Airport, World Area Code
CRSDepTime	CRS Departure Time (local time: hhmm)
DepTime	Actual Departure Time (local time: hhmm)
DepDelay	Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
DepDelayMinutes	Difference in minutes between scheduled and actual departure time. Early departures set to 0.
DepDel15	Departure Delay Indicator, 15 Minutes or More (1=Yes)
DepartureDelayGroups	Departure Delay intervals, every (15 minutes from <-15 to >180)
TaxiOut	Taxi Out Time, in Minutes
WheelsOff	Wheels Off Time (local time: hhmm)
WheelsOn	Wheels On Time (local time: hhmm)
TaxiIn	Taxi In Time, in Minutes
ArrTime	Actual Arrival Time (local time: hhmm)
ArrDelay	Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
ArrDelayMinutes	Difference in minutes between scheduled and actual arrival time. Early arrivals set to 0.
ArrDel15	Arrival Delay Indicator, 15 Minutes or More (1=Yes)
ArrivalDelayGroups	Arrival Delay intervals, every (15-minutes from <-15 to >180)
Cancelled	Cancelled Flight Indicator (1=Yes)
CancellationCode	Specifies The Reason For Cancellation
Diverted	Diverted Flight Indicator (1=Yes)
ActualElapsedTime	Elapsed Time of Flight, in Minutes
AirTime	Flight Time, in Minutes
Flights	Number of Flights
CarrierDelay	Carrier Delay, in Minutes
WeatherDelay	Weather Delay, in Minutes
NASDelay	National Air System Delay, in Minutes
SecurityDelay	Security Delay, in Minutes
LateAircraftDelay	Late Aircraft Delay, in Minutes
FirstDepTime	First Gate Departure Time at Origin Airport
TotalAddGTime	Total Ground Time Away from Gate for Gate Return or Cancelled Flight
LongestAddGTime	Longest Time Away from Gate for Gate Return or Cancelled Flight

Column name	Explanation of Column Name and Codes
DivAirportLandings	Number of Diverted Airport Landings
DivReachedDest	Diverted Flight Reaching Scheduled Destination Indicator (1=Yes)
DivActualElapsedTime	Elapsed Time of Diverted Flight Reaching Scheduled Destination, in Minutes. The ActualElapsedTime column remains NULL for all diverted flights.
DivArrDelay	Difference in minutes between scheduled and actual arrival time for a diverted flight reaching scheduled destination. The ArrDelay column remains NULL for all diverted flights.
DivDistance	Distance between scheduled destination and final diverted airport (miles). Value will be 0 for diverted flight reaching scheduled destination.
Div1Airport	Diverted Airport Code1
Div1AirportID	Airport ID of Diverted Airport 1. Airport ID is a Unique Key for an Airport
Div1AirportSeqID	Airport Sequence ID of Diverted Airport 1. Unique Key for Time Specific Information for an Airport
Div1WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code1
Div1TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code1
Div1LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code1
Div1WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code1
Div1TailNum	Aircraft Tail Number for Diverted Airport Code1
Div2Airport	Diverted Airport Code2
Div2AirportID	Airport ID of Diverted Airport 2. Airport ID is a Unique Key for an Airport
Div2AirportSeqID	Airport Sequence ID of Diverted Airport 2. Unique Key for Time Specific Information for an Airport
Div2WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code2
Div2TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code2
Div2LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code2
Div2WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code2
Div2TailNum	Aircraft Tail Number for Diverted Airport Code2
Div3Airport	Diverted Airport Code3
Div3AirportID	Airport ID of Diverted Airport 3. Airport ID is a Unique Key for an Airport
Div3AirportSeqID	Airport Sequence ID of Diverted Airport 3. Unique Key for Time Specific Information for an Airport
Div3WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code3
Div3TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code3
Div3LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code3
Div3WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code3
Div3TailNum	Aircraft Tail Number for Diverted Airport Code3
Div4Airport	Diverted Airport Code4
Div4AirportID	Airport ID of Diverted Airport 4. Airport ID is a Unique Key for an Airport
Div4AirportSeqID	Airport Sequence ID of Diverted Airport 4. Unique Key for Time Specific Information for an Airport
Div4WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code4
Div4TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code4
Div4LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code4
Div4WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code4
Div4TailNum	Aircraft Tail Number for Diverted Airport Code4
Div5Airport	Diverted Airport Code5
Div5AirportID	Airport ID of Diverted Airport 5. Airport ID is a Unique Key for an Airport
Div5AirportSeqID	Airport Sequence ID of Diverted Airport 5. Unique Key for Time Specific Information for an Airport
Div5WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code5
Div5TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code5
Div5LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code5

Column name	Explanation of Column Name and Codes
Div5WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code5
Div5TailNum	Aircraft Tail Number for Diverted Airport Code5

3.4.1.3 Airport Data

The data set provided by the Federal Aviation Administration is a data set created based on the Airport Master Record (5010-*) forms. The following list of issues needs to be corrected in a later stage of the project:

- The LocationID have an apostrophe as the first character, which should be removed for further processing.
- Number of States in the data set is above the actual number of states of the U.S.

The Federal Aviation Administration provides code books for some of the data details in the airport data. Based on these code books the records with the following values can be removed from the data set, as they are not relevant for the goals of the project.

Column name	Value	Reason for removal
TYPE	"BALLOONPORT"	Record is indicating a balloon port, not an airport.
TYPE	"GLIDERPORT"	Record is indicating a glider port, not an airport.
TYPE	"HELIPORT"	Record is indicating a helicopter port, not an airport.
TYPE	"SEAPLANE BASE"	Record is indicating a port for seaplanes, not an airport.
TYPE	"ULTRALIGHT"	Record is indicating a port for ultralight airplanes, not an airport.

Similarly to the previous data sets this data set contains a great deal of details as well, which can be used in different projects, but for our purposes the following details have to be removed.

Column name	Explanation of Column Name and Codes
SiteNumber	Landing facility site number - a unique identifying number which, together with the landing facility type code, forms the key to the airport record. (ex. 04508.*A)
EffectiveDate	Information effective date (mm/dd/yyyy). This date coincides with the 56-day charting and publication cycle date.
DistrictOffice	FAA district or field office code. (ex. CHI)
County	Associated county (or parish) name. (ex. Cook)
CountyState	Associated county's state (post office code) state where the associated county is located; may not be the same as the associated city's state code. (ex. IL)
Ownership	Airport ownership type. (ex. PU - publicly owned, PR - privately owned, MA - air force owned, MN - navy owned, MR - army owned)
Use	Facility use. (ex. PU - open to the public, PR - private)
Owner	Facility owner's name.
OwnerAddress	Owner's address.
OwnerCSZ	Owner's city, state and zip code.
OwnerPhone	Owner's phone number. (data formats: nnn-nnn-nnnn (area code + phone number), 1-nnn-nnnn (dial 1-800 then number), 8-nnn-nnnn (dial 800 then number)
Manager	Facility manager's name.
ManagerAddress	Manager's address.
ManagerCSZ	Manager's city, state and zip code.

Column name	Explanation of Column Name and Codes
ManagerPhone	Manager’s phone number. (data formats: nnn-nnn-nnnn (area code + phone number), 1-nnn-nnnn (dial 1-800 then number), 8-nnn-nnnn (dial 800 then number)
ARPMethod	Airport reference point determination method. (ex. E - estimated, S - surveyed)
ARPElevationMethod	Airport elevation determination method. (ex. E - estimated, S - surveyed)
MagneticVariation	Magnetic variation and direction magnetic variation to nearest degree. (ex. 03W)
MagneticVariationYear	Magnetic variation epoch year. (ex. 1985)
TrafficPatternAltitude	Traffic pattern altitude (whole feet AGL). (ex. 1000)
ChartName	Aeronautical sectional chart on which facility appears. (ex. Washington)
DistanceFromCBD	Distance from central business district of the associated city to the airport (nearest nautical mile - ex. 08).
DirectionFromCBD	Direction of airport from central business district of associated city (nearest 1/8 compass point - ex. NE).
BoundaryARTCCID	Boundary ARTCC Identifier. The boundary ARTCC is the FAA air route traffic control center within whose published boundaries the airport lies. It may not be the controlling ARTCC for the airport if a letter of agreement exists between the boundary ARTCC and another ARTCC. (ex. ZDC for Washington ARTCC)
BoundaryARTCCComputerID	Boundary ARTCC (FAA) computer identifier. (ex. ZCW for Washington ARTCC)
BoundaryARTCCName	Boundary ARTCC name. (ex. Washington)
ResponsibleARTCCID	Responsible ARTCC identifier the responsible ARTCC is the FAA air route traffic control center who has assumed control over the airport through a letter of agreement with the boundary ARTCC. (ex. ZDC for Washington ARTCC)
ResponsibleARTCCComputerID	Responsible ARTCC (FAA) computer identifier. (ex. ZCW for Washington ARTCC)
ResponsibleARTCCName	Responsible ARTCC name. (ex. Washington)
TieInFSS	Tie-in FSS physically located on facility. (ex. Y - tie-in FSS is on the airport, n - tie-in FSS is not on the airport)
TieInFSSID	Tie-in flight service station (FSS) identifier. (ex. DCA for Washington FSS)
TieInFSSName	Tie-in FSS name. (ex. Washington)
AirportToFSSPhoneNumber	Local phone number from airport to FSS for adminstrative services
TieInFSSTollFreeNumber	Toll free phone number from airport to FSS for pilot briefing services the data describes the type of toll-free communications and the number to dial. The data formats and their meanings are: 1-nnn-nnnn, dial 1-800- then nnn-nnnn; 8-nnn-nnnn, dial 800 then nnn-nnnn; e-nnnnnnnn, enterprise number dial 0 & ask for enterprise nnnnnnnn; lcnnn-nnnn, local call - dial nnn-nnnn; dl, direct line telephone at the airport - no dialing required; z-nnnnnnnn, zenith number - dial 0 and ask for zenith nnnnnnnn; w-nnnnnnnn, dial 0 and ask for wx nnnnnnnn; c-nnnnnnnn, dial 0 and ask for commerce nnnnnnnn; ld-nnnnnnnn, long distance call - dial (area code) then nnnnnnn; lt-nnnnnnnn, long distal call dial 1-nnnnnnn; 1-wx-brief, dial 1-800-wx-brief; 8-wx-brief, dial 800-wx-brief
AlternateFSSID	Alternate FSS identifier provides the identifier of a full-time flight service station that assumes responsibility for the airport during the off hours of a part-time primary FSS. (ex. ‘DCA’ for Washington FSS)
AlternateFSSName	Alternate FSS name. (ex. ‘Washington’ for Washington FSS)

Column name	Explanation of Column Name and Codes
AlternateFSSTollFreeNumber	Toll free phone number from airport to FSS for pilot briefing services the data describes the type of toll-free communications and the number to dial. The data formats and their meanings are: 1-nnn-nnnn, dial 1-800- then nnn-nnnn; 8-nnn-nnnn, dial 800 then nnn-nnnn; e-nnnnnnnn, enterprise number dial 0 & ask for enterprise nnnnnnnn; lcnnn-nnnn, local call - dial nnn-nnnn; dl, direct line telephone at the airport - no dialing required; z-nnnnnnnn, zenith number - dial 0 and ask for zenith nnnnnnnn; w-nnnnnnnn, dial 0 and ask for wx nnnnnnnn; c-nnnnnnnn, dial 0 and ask for commerce nnnnnnnn; ld-nnnnnnnn, long distance call - dial (area code) then nnnnnnn; lt-nnnnnnnn, long distal call dial 1-nnnnnnn; 1-wx-brief, dial 1-800-wx-brief; 8-wx-brief, dial 800-wx-brief.
NOTAMFacilityID	Identifier of the facility responsible for issuing notices to airmen (NOTAMS) and weather information for the airport. (ex. ORD)
NOTAMService	Availability of NOTAM 'd' service at airport. (ex. Y - yes, N - no)
ActivationDate	Airport activation date (mm/yyyy). Provides the month and year that the facility was added to the NFDC airport database. Note: this information is only available for those facilities opened since 1981. (ex. 06/1981)
CertificationTypeDate	Airport certification type and date. Format is the class code ('I', 'II', 'III' or 'IV') followed by a one characther code A, B, C, D, E, or L, followed by a one character code S or U, followed by the month and year of certification. (ex. 'I A S 07/1980', 'I C S 01/1983' or 'I A U 09/1983'). Codes A, B, C, D, and E are for airports having a full certificate under CFR Part 139, and receiving scheduled air carrier service from carriers certificated by the Civil Aeronautics Board. The A, B, C, D, and E identify the aircraft rescue and firefighting index for the airport. Code L is for airports having limited certification under CFR Part 139. Code S is for Airports receiving scheduled air carrier service from carriers certificated by the Civil Aeronautics Board. Code U is for airports not receiving this scheduled service.
FederalAgreements	NPIAS/Federal Agreement Code. A combination of 1 to 7 codes that indicate the type of federal agreements existing at the airport. (ex. NGH). N - national plan of integrated airport systems (NPIAS); B - installation of navigational facilities on privately owned airports under F&E program; G - grant agreements under FAAP/ADAP/AIP; H - compliance with accessibility to the handicapped; P - surplus property agreement under Public Law 289; R - surplus property agreement under Regulation 16-WAA; S - conveyance under section 16, Federal Airport Act of 1946 or Section 23, Airport and Airway Development Act of 1970; V - advance planning agreement under FAAP; X - obligations assumed by transfer; Y - assurances pursuant to Title VI, Civil Rights Act of 1964; Z - conveyance under Section 303(C), Federal Aviation Act of 1958; 1 - grant agreement has expired, however, agreement remains in effect for this facility as long as it is public use.
AirspaceDetermination	Airport airspace analysis determination. (ex. CONDL (conditional), NOT ANALYZED, NO OBJECTION, OBJECTIONABLE)
CustomsAirportOfEntry	Facility has been designated by the U.S. Treasury as an international airport of entry for customs (ex. Y - yes, N - no)
CustomsLandingRights	Facility has been designated by the U.S. Treasury as a customs landing rights airport (ex. Y - yes, N - no)
MilitaryJointUse	Facility has military/civil joint use agreement that allows civil operations at a military airport or military operations at a civil airport (ex. Y - yes, N - no)

Column name	Explanation of Column Name and Codes
MilitaryLandingRights	Airport has entered into an agreement that grants landing rights to the military (ex. Y - yes, N - no)
InspectionMethod	Airport inspection method. (ex. F - federal, S - state, C - contractor, 1 - 5010-1 public use mail out program, 2 - 5010-2 private use mail out program)
InspectionGroup	Agency/group performing physical inspection (ex. F - faa airports field personnel, s - state aeronautical personnel, c - private contract personnel, n - owner)
LastInspectionDate	Last physical inspection date (mmddyyyy)
LastOwnerInformationDate	Last date information request was completed by facility owner or manager (mmddyyyy)
FuelTypes	Fuel types available for public use at the airport. There can be up to 8 occurrences of a fixed 5 character field (ex. 80___100__100LL115__). 80 - grade 80 gasoline (red), 100 - grade 100 gasoline (green), 100LL - grade 100LL gasoline (low lead blue), 115 - grade 115 gasoline, A - jet A - kerosene, freeze point -40C, A1 - jet A-1 - kerosene, freeze point -50C, A1+ - jet A-1 - kerosene, with icing inhibitor freeze point -50C, B - jet B - wide-cut turbine fuel, freeze point -50C, B+ - jet B - wide-cut turbine fuel with icing inhibitor, freeze point -50C, MOGAS - automotive gasoline.
AirframeRepair	Airframe repair service availability/type. (ex. MAJOR, MINOR, NONE)
PowerPlantRepair	Power plant (engine) repair availability/type. (ex. MAJOR, MINOR, NONE)
BottledOxygenType	Type of bottled oxygen available (value represents high and/or low pressure replacement bottle). (ex. HIGH, LOW, HIGH/LOW, NONE)
BulkOxygenType	Type of bulk oxygen available (value represents high and/or low pressure cylinders). (ex. HIGH, LOW, HIGH/LOW, NONE)
LightingSchedule	Airport lighting schedule value is the beginning-ending times (local time) that lights are operated. Format can be 1900-2300, DUSK-0100, ALL, DUSK-DAWN, NONE, etc.
BeaconSchedule	Beacon lighting schedule value is the beginning-ending times (local time) that the rotating airport beacon light is operated. Value can be “SS-SR” (indicating sunset-sunrise), blank, or “SEE RMK”, indicating that the details are in a facility remark data entry.
ATCT	Air traffic control tower located on airport. (ex. Y - yes, N - no)
UNICOMFrequencies	Unicom frequencies available at the airport there can be up to 6 occurrences of a fixed 7 character field. (ex. 122.700 or 122.700122.800 or NONE)
CTAFFrequency	Common traffic advisory frequency. (CTAF) (ex. 122.800)
SegmentedCircle	Segmented circle airport marker system on the airport. (ex. Y - yes, N - no, none)
BeaconColor	Lens color of operable beacon located on the airport. (ex. CG - clear-green (lighted land airport); CY - clear-yellow (lighted seaplane base); CGY - clear-green-yellow (heliport); SCG - split-clear-green (lighted military airport); C - clear (unlighted la
NonCommercialLandingFee	Landing fee charged to non-commercial users of airport. (ex. Y - yes, N - no)
MedicalUse	Landing facility is used for medical purposes. (ex. Y - yes, N - no)
SingleEngineGA	Number of single engine general aviation aircraft.
MultiEngineGA	Number of multi engine general aviation aircraft.
JetEngineGA	Number of jet engine general aviation aircraft.
HelicoptersGA	Number of general aviation helicopter.
GlidersOperational	Number of operational gliders.

Column name	Explanation of Column Name and Codes
MilitaryOperational	Number operational military aircraft (includingg helicopters).
Ultralights	Number of ultralight aircraft.
OperationsCommercial	Commercial services. Scheduled operations by cab-certificated carriers or intrastate carriers.
OperationsCommuter	Commuter services. Scheduled commuter and cargo carriers.
OperationsAirTaxi	Air taxi. Air taxi operators carrying passengers, mail, or mail for revenue.
OperationsGALocal	General aviation local operations. Those operating in the local traffic pattern or within a 20-mile radius of the airport.
OperationsGAItin	General aviation itinerant operations. Those general aviation operations (excluding commuter or air taxi) not qualifying as local.
OperationsMilitary	Military aircraft operations.
OperationsDate	12-month ending date on which annual operations data in above six field is based (mm/dd/yyyy).
AirportPositionSource	Airport position source.
AirportPositionSourceDate	Airport position source date (mm/dd/yyyy).
AirportElevationSource	Airport elevation source.
AirportElevationSourceDate	Airport elevation source date (mm/dd/yyyy).
ContractFuelAvailable	Contract fuel available. (ex. Y - yes, N - no)
TransientStorage	Transient storage. (ex. Y - yes, N - no, none)
OtherServices	Other services. (ex. Y - yes, N - no, none)
WindIndicator	Wind direction indicator. (ex. Y - yes, N - no, none)

4 Data Preparation

4.1 Data Set

4.1.1 Data Set Description

The resolution of the issues found during the data quality verification includes the reducing of several details originally provided by the Federal Aviation Administration and the United States Department of Transportation agencies. This section describes the resulted data sets.

4.1.1.1 Animal Strike Data

Column name	Explanation of Column Name and Codes
INDEX NR	Individual record number
OPID	Airline operator code
OPERATOR	A three letter International Civil Aviation Organization code for aircraft operators. (BUS = business, PVT = private aircraft other than business, GOV = government aircraft, MIL - military aircraft.)
ATYPE	Aircraft
AC_CLASS	Type of aircraft (see Aircraft Type tab below)
AC_MASS	1 = 2,250 kg or less: 2 = ,2251-5700 kg: 3 = 5,701-27,000 kg: 4 = 27,001-272,000 kg: 5 = above 272,000 kg
TYPE_ENG	Type of power A = reciprocating engine (piston): B = Turbojet: C = Turboprop: D = Turbofan: E = None (glider): F = Turboshaft (helicopter): Y = Other
REG	Aircraft registration
FLT	Flight number
INCIDENT_DATE	Date strike occurred
INCIDENT_MONTH	Month strike occurred
INCIDENT_YEAR	Year strike occurred
TIME_OF_DAY	Light conditions
TIME	Hour and minute in local time
AIRPORT_ID	International Civil Aviation Organization airport identifier for location of strike whether it was on or off airport
AIRPORT	Name of airport
STATE	State
FAAREGION	FAA Region where airport is located
ENROUTE	If strike did not occur on approach, climb, landing roll, taxi or take-off, aircraft was enroute. This shows location.
RUNWAY	Runway
HEIGHT	Feet Above Ground Level
SPEED	Knots (indicated air speed)
DISTANCE	Miles from airport
PHASE_OF_FLT	Phase of flight during which strike occurred
SKY	Type of cloud cover, if any
PRECIP	Precipitation
WARNED	Pilot warned of birds/wildlife

The number of details (columns) for each strike report has been reduces from 94 to 27.

4.1.1.2 Flight Data

Column name	Explanation of Column Name and Codes
Year	Year
Quarter	Quarter (1-4)
Month	Month
DayofMonth	Day of Month
DayOfWeek	Day of Week
FlightDate	Flight Date (yyyymmdd)
Carrier	Code assigned by IATA and commonly used to identify a carrier. As the same code may have been assigned to different carriers over time, the code is not always unique. For analysis, use the Unique Carrier Code.
UniqueCarrier	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.
FlightNum	Flight Number
Origin	Origin Airport
OriginCityName	Origin Airport, City Name
OriginState	Origin Airport, State Code
OriginStateName	Origin Airport, State Name
Dest	Destination Airport
DestCityName	Destination Airport, City Name
DestState	Destination Airport, State Code
DestStateName	Destination Airport, State Name
CRSDepTime	CRS Departure Time (local time: hhmm)
DepTimeBlk	CRS Departure Time Block, Hourly Intervals
CRSArrTime	CRS Arrival Time (local time: hhmm)
CRSElapsedTime	CRS Elapsed Time of Flight, in Minutes
Distance	Distance between airports (miles)
DistanceGroup	Distance Intervals, every 250 Miles, for Flight Segment

The number of details (columns) for each flight performance record has been reduces from 110 to 23.

4.1.1.3 Aiport Data

Column name	Explanation of Column Name and Codes
Type	Landing facility type. (ex. Airport, Balloonport, Seaplane Base, Gliderport, Heliport, Stolport, Ultralight)
LocationID	Location identifier unique 3-4 character alphanumeric identifier assigned to the landing facility. (ex. 'ORD' for Chicago O'Hare)
Region	FAA region code. (ex. AAL - Alaska, ACE - Central, AEA - Eastern, AGL - Great Lakes, AIN - International, ANE - New England, ANM - Northwest Mountain, ASO - Southern, ASW - Southwest, AWP - Western-Pacific)
State	Associated state post office code standard two letter abbreviation for u.s. states and territories. (ex. IL, PR, CQ)
StateName	Associated state name. (ex. Illinois)
City	Associated city name. (ex. Chicago)
FacilityName	Official facility name. (ex. Chicago O'Hare Intl)
ARPLatitude	Airport reference point latitude (formatted).
ARPLatitudeS	Airport reference point latitude (seconds).
ARPLongitude	Airport reference point longitude (formatted).
ARPLongitudeS	Airport reference point longitude (seconds).

Column name	Explanation of Column Name and Codes
ARPElevation	Airport elevation (nearest foot MSL). Elevation is measured at the highest point on the centerline of the usable landing surface. (ex. 1200; -10 for 10 feet below sea level)
LandAreaCoveredByAirport	Amount of land owned by the airport in acres.
AirportStatusCode	Airport status code: CI - closed indefinitely; CP - closed permanently; O - operational
IcaoIdentifier	International coding for airport.

The number of details (columns) for each airport record has been reduces from 102 to 8.

4.2 Select Data

4.2.1 Rationale for Inclusion / Exclusion

The resolution of the issues found during the data quality verification includes the exclusion of certain records from the data sets originally provided by the Federal Aviation Administration and the United States Department of Transportation agencies. This section provides the summary of the changes on the data sets.

4.2.1.1 Animal Strike Data

The following columns are impacted by the selection criteria described in the data quality verification section:

- OPID
- AC_CLASS
- TYPE_ENG

Additionally the number of States in the data set is above the actual number of states of the U.S., so the data needs to be reduced to contain only the following states:

Abbreviation	Name	Abbreviation	Name
AL	Alabama	MT	Montana
AK	Alaska	NE	Nebraska
AZ	Arizona	NV	Nevada
AR	Arkansas	NH	New Hampshire
CA	California	NJ	New Jersey
CO	Colorado	NM	New Mexico
CT	Connecticut	NY	New York
DE	Delaware	NC	North Carolina
FL	Florida	ND	North Dakota
GA	Georgia	OH	Ohio
HI	Hawaii	OK	Oklahoma
ID	Idaho	OR	Oregon
IL	Illinois	PA	Pennsylvania
IN	Indiana	RI	Rhode Island
IA	Iowa	SC	South Carolina
KS	Kansas	SD	South Dakota
KY	Kentucky	TN	Tennessee
LA	Louisiana	TX	Texas
ME	Maine	UT	Utah
MD	Maryland	VT	Vermont
MA	Massachusetts	VA	Virginia

Abbreviation	Name	Abbreviation	Name
MI	Michigan	WA	Washington
MN	Minnesota	WV	West Virginia
MS	Mississippi	WI	Wisconsin
MO	Missouri	WY	Wyoming

4.2.1.2 Flight Data

The data in the Federal Aviation Administration Animal Strike Database is available only until 30-4-2016, so the flight data needs to be adjusted accordingly.

Additionally the number of States in the data set is above the actual number of states of the U.S., so the data needs to be reduced to contain only the following states:

Abbreviation	Name	Abbreviation	Name
AL	Alabama	MT	Montana
AK	Alaska	NE	Nebraska
AZ	Arizona	NV	Nevada
AR	Arkansas	NH	New Hampshire
CA	California	NJ	New Jersey
CO	Colorado	NM	New Mexico
CT	Connecticut	NY	New York
DE	Delaware	NC	North Carolina
FL	Florida	ND	North Dakota
GA	Georgia	OH	Ohio
HI	Hawaii	OK	Oklahoma
ID	Idaho	OR	Oregon
IL	Illinois	PA	Pennsylvania
IN	Indiana	RI	Rhode Island
IA	Iowa	SC	South Carolina
KS	Kansas	SD	South Dakota
KY	Kentucky	TN	Tennessee
LA	Louisiana	TX	Texas
ME	Maine	UT	Utah
MD	Maryland	VT	Vermont
MA	Massachusetts	VA	Virginia
MI	Michigan	WA	Washington
MN	Minnesota	WV	West Virginia
MS	Mississippi	WI	Wisconsin
MO	Missouri	WY	Wyoming

4.2.1.3 Airport Data

The data set contains the data of the currently operational and closed airports along with other aviation facilities (e.g.: balloon port), while I am interested of the data of only those airports which are operational, so the data of the closed airports and other type of aviation facilities needs to be removed.

The number of States in the data set is above the actual number of states of the U.S., so the data needs to be reduced to contain only the following states:

Abbreviation	Name	Abbreviation	Name
AL	Alabama	MT	Montana

Abbreviation	Name	Abbreviation	Name
AK	Alaska	NE	Nebraska
AZ	Arizona	NV	Nevada
AR	Arkansas	NH	New Hampshire
CA	California	NJ	New Jersey
CO	Colorado	NM	New Mexico
CT	Connecticut	NY	New York
DE	Delaware	NC	North Carolina
FL	Florida	ND	North Dakota
GA	Georgia	OH	Ohio
HI	Hawaii	OK	Oklahoma
ID	Idaho	OR	Oregon
IL	Illinois	PA	Pennsylvania
IN	Indiana	RI	Rhode Island
IA	Iowa	SC	South Carolina
KS	Kansas	SD	South Dakota
KY	Kentucky	TN	Tennessee
LA	Louisiana	TX	Texas
ME	Maine	UT	Utah
MD	Maryland	VT	Vermont
MA	Massachusetts	VA	Virginia
MI	Michigan	WA	Washington
MN	Minnesota	WV	West Virginia
MS	Mississippi	WI	Wisconsin
MO	Missouri	WY	Wyoming

4.3 Clean Data

4.3.1 Data Cleaning Report

The resolution of the issues found during the data quality verification includes the exclusion of certain records from the data sets originally provided by the Federal Aviation Administration and the United States Department of Transportation agencies. This section provides the summary of the changes on the data sets.

4.3.1.1 Animal Strike Data

The data quality verification identified that the data provided by the Federal Aviation Administration contains the following problems impacting the indicated columns:

- Mixed use of uppercase and lowercase letters/codes
  - TYPE_ENG
  - TIME_OF_DAY
  - PHASE_OF_FLT
  - SKY
  - PRECIP
  - WARNED
- Mixed use of codes (e.g.: engine type is defined as "A/C")
  - TYPE_ENG
  - SKY

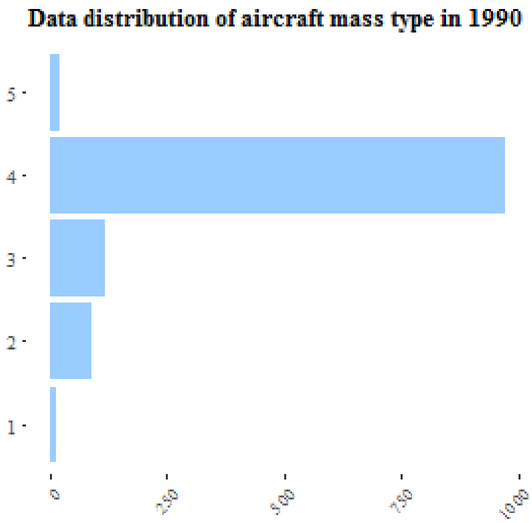
The first summary table shows the number of distinct items for each year regarding the Airline operators, Aircraft, Aircraft types, Aircraft mass types, and Engine types, which have been reported as being affected in an animal strike after the selection and cleanup tasks.

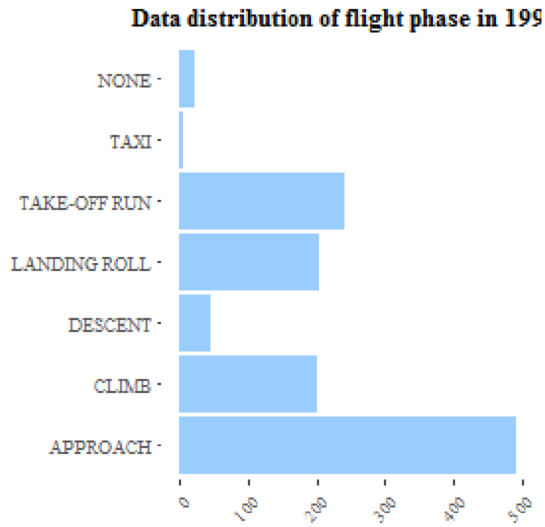
Year	# of reports	Operators	Aircraft	Aircraft type	Aircraft mass type	Engine type
1990	1190	94	79	1	5	4

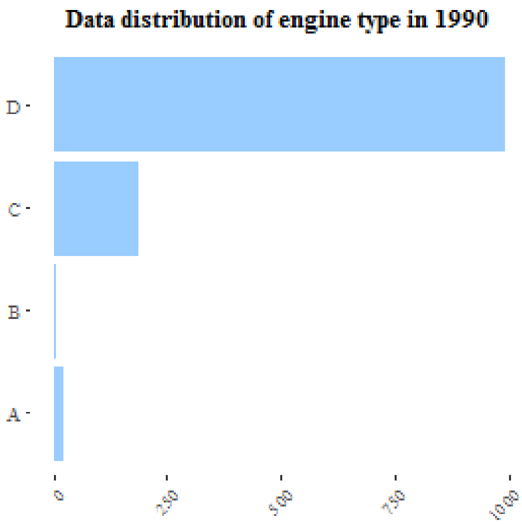
The second summary table shows the number of distinct items for each year regarding the Time of day, Airports, States, Phase of flight, weather conditions (Sky and Precipitation), and the flag for showing if the pilot has been warned or not about birds / wildlife in the reports after the selection and cleanup tasks.

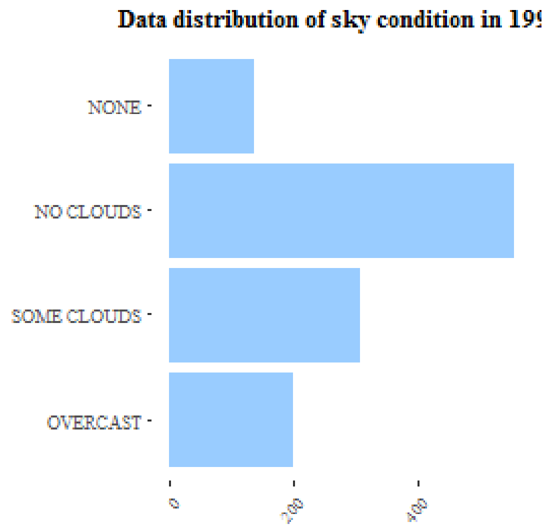
Year	Time of day	Airports	States	Phase of flight	Sky	Precipitation	Warned
1990	5	208	49	7	4	4	3

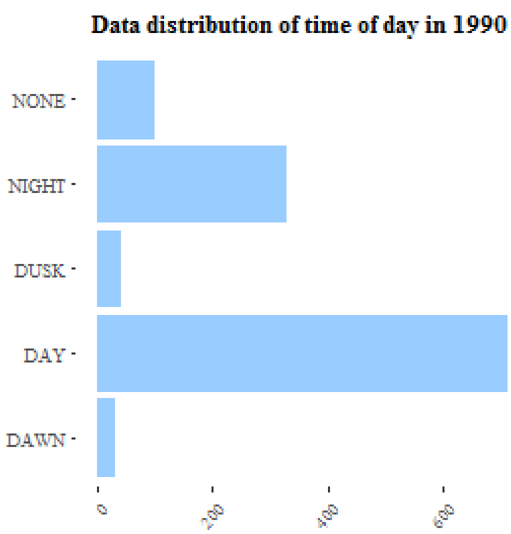
The following graphs show the distributions of some of the selected distinct items summarized in the tables above.

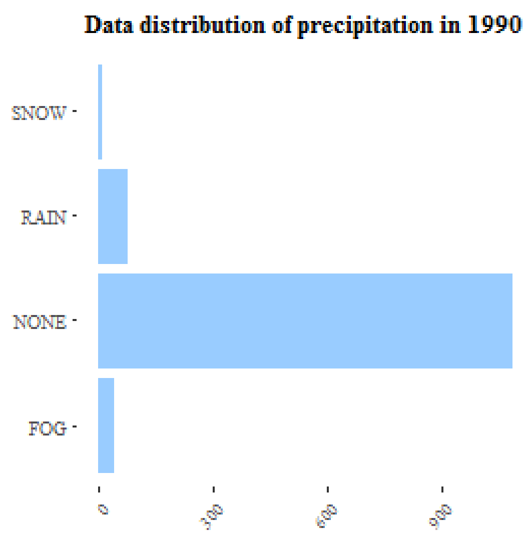

Data distribution of aircraft mass type in 1990
5
4
3
2
1
0
250
500
750
1000


Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
100
200
300
400
500


Data distribution of engine type in 1990
D
C
B
A
0
250
500
750
1000


Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400


Data distribution of time of day in 1990
NONE
NIGHT
DUSK
DAY
DAWN
0
200
400
600


Data distribution of precipitation in 1990
SNOW
RAIN
NONE
FOG
0
300
600
900

4.3.1.2 Flight Data

I did not identify any data quality issues - which have not been corrected in the previous steps - with the data provided by the United States Department of Transportation during the data exploration and data quality verification exercises.

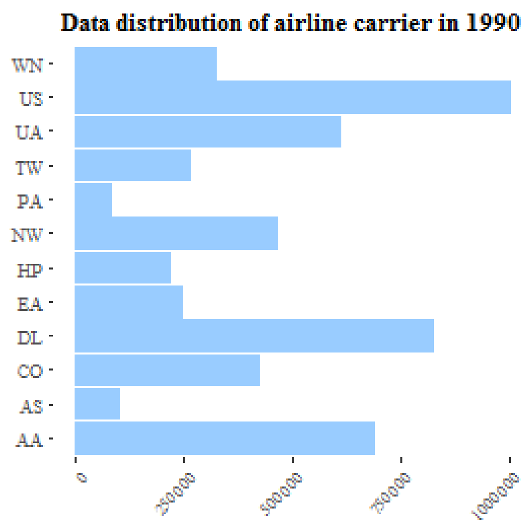
The first summary table shows the number of distinct items for each year regarding the number of records, the carriers, and the origin and the destination airports after the selection and cleanup tasks.

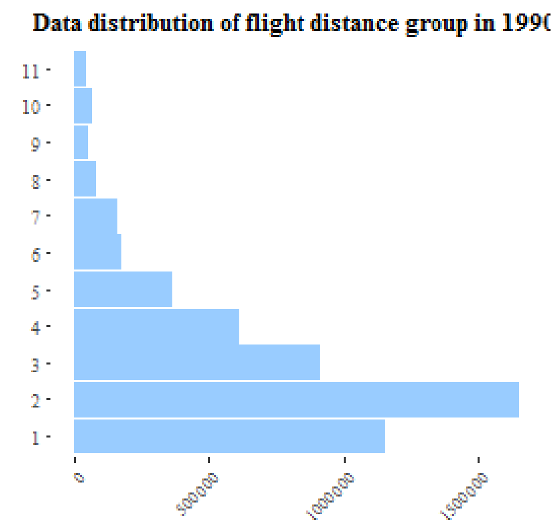
Year	# of flights	# of carriers	Origin airports	Origin states	Destination airports	Destination states
1990	5220743	12	226	49	227	49

The second summary table shows the number of distinct items for each year the departure time group and distance between the airports after the selection and cleanup tasks.

Year	Departure time block	Distance group
1990	19	11

The following graphs show the distributions of some of the selected distinct items summarized in the tables above.





4.3.1.3 Airport Data

I did not identify any data quality issues - which have not been corrected in the previous steps - with the data provided by the Federal Aviation Administration during the data exploration and data quality verification exercises.

4.4 Construct Data

4.4.1 Derived Attributes

Taking into account that the animal strikes might be related to the amount of the traffic being generated by the airports and some conditions (like the minimum, maximum and average distance) of the flights initiated and terminated from the airports the following supporting attributes will be created for each airport:

- Average number of originated flights
- Average number of departed flights
- Longest flight originated from the airport
- Longest flight departed to the airport
- Shortest flight originated from the airport
- Shortest flight departed to the airport
- Average distance of the flights originated from the airport
- Average distance of the flights departed to the airport

Another set of attributes, which needs to be taken into account is based on the animal strike data. The following supporting attribute(s) will be created for each airport:

- Number of strikes at the airport

4.4.2 Generated Records

The information described by the data records and the massive amount of data provided by the Federal Aviation Administration and the United States Department of Transportation does not require to generate additional records at this stage of the project. It might still happen on the other hand that during the model building it will be required to generate more records (or reduce the number of records), but this task will be performed at the model creation stage based on the preliminary evaluation of the model.

4.5 Integrate Data

4.5.1 Merged Data

The business objectives are defining two main goals for the project, namely:

1. Create a statistical analysis to identify those reasons (based on the data available), which are determining the the risk of an animal strike for an airport.
2. Create a prediction model, which can be used to predict the risk of an animal strike for a given flight.

Realizing these objectives I need to merge the data sets based on two very different set of criteria, while I have to take into account the impact of the merge on the final results.

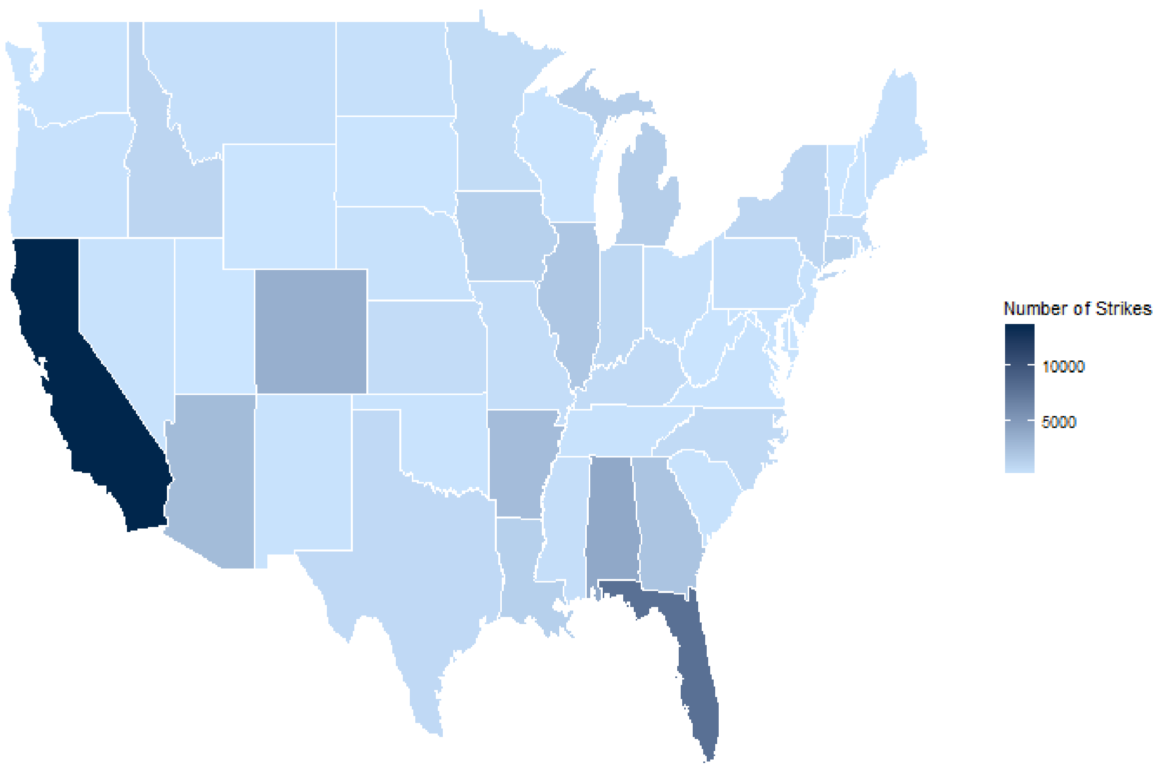
Criteria set for merging the data regarding the first business goal:

- Base data set to be used is the Federal Aviation Administration Airport Data & Contact Information
- Use airports having flight data available (both as an origin and as a destination airport) in the data set obtained from the United States Department of Transportation
- Enrich the data with the animal strike data acquired from the Federal Aviation Administration Wildlife Strike Database

There are multiple different ways of uniquely identifying an airport. In the US one option is to use the Federal Aviation Administration Location ID, while another way is to use the international airport code of the International Civil Aviation Organization (ICAO). The data sets are using a mix of these identifiers.

Note: The airports might have identification code from the International Air Transport Association (IATA) and/or the International Civil Aviation Organization (ICAO) and/or Federal Aviation Administration (FAA LID). Some airports have identification codes from all these organizations, some does not. The data acquired from the agencies are using codes from two of the identification code types.

Based on the merged data I can visualize the distribution of the animal strikes in the map of the US.



Criteria set for merging the data regarding the second business goal:

- Base data set to be used is the available in the data set obtained from the United States Department of Transportation
- Marking the flight data with the animal strike information should be based on the following list of criteria:
  - airline / carrier
  - incident date
  - airport and state
  - flight number

Notes:

- The Federal Aviation Administration Wildlife Strike Database is based on the geographical location of the strike. The database contains the strike records regardless of the airline / carrier, meaning that it contain the strike data of international flights as well.
- The flight performance data from the United States Department of Transportation contains the data only from the major airlines.

The strict criteria for integrating the two main data sets resulted the following actual number of identified strikes in the flight performance data:

Number of flight records	Number of striked records	Percentage
155,432,729	28,740	0.01849%

4.6 Format Data

4.6.1 Reformatted Data

The data provided by the Federal Aviation Administration and the United States Department of Transportation did already contain several restrictions about the data format and during the selection, cleanup and integration exercises more data formatting has been applied, therefore no additional data formatting is required at this stage of the project.

5 Modeling

5.1 Select Modeling Technique for the first model

5.1.1 Modeling Technique

My goal is to create a statistical model to analyse a continuous variable as the outcome variable (called by statisticians as Y, the response variable, the dependent variable) using multiple predictors (called by statisticians as X, independent variable, explanatory variable). Having the outcome variable as a continuous variable, I'm going to use a linear regression model for the analysis.

The data sets I have available are simplifying the situation as they do not contain several variables (like the distance to national parks, the flight routes of the birds, actions taken by the authorities of the different airports, etc.), which might have significant impact on the model results. These variables are called confounder variables and can be defined as follows:

"A confounder is a third variable that biases (increases or decreases) the association we are interested in. The confounder is always associated with both the response and the predictor." (Gergely Daróczi, Renáta Németh, and Gergely Tóth 2015)

5.1.2 Modeling Assumptions

Besides the assumptions taken with the standard estimation techniques of the linear regression model I did not take any other modeling related assumption.

These assumptions taken are the following:

1. The outcome variable is a continuous variable
2. The residuals are statistically independent
3. There is a relationship between the outcome variable and each predictor variable
4. The outcome variable has a normal distribution
5. The variance of the outcome variable is fixed regardless of the predictor variables

The assumptions above are based on the definitions and the descriptions written in the "Mastering Data Analysis with R" book by (Gergely Daróczi, Renáta Németh, and Gergely Tóth 2015).

5.2 Generate Test Design for the first model

5.2.1 Test Design

The first model I am building is not a prediction model, instead it's a regression model for analising the data. Therefore I'm not going to create a separate train and test data set, as it has not meaning in this context.

5.3 Build Model for the first model

5.3.1 Parameter Settings

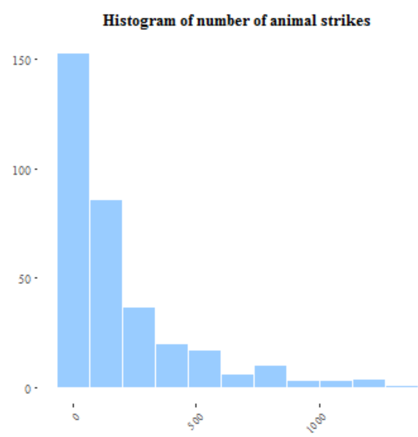
The R language provides quite a lot of possibilities for setting different parameters for the linear regression models. Keeping the modelling as simple as possible I'm using the default settings for the linear model fitting.

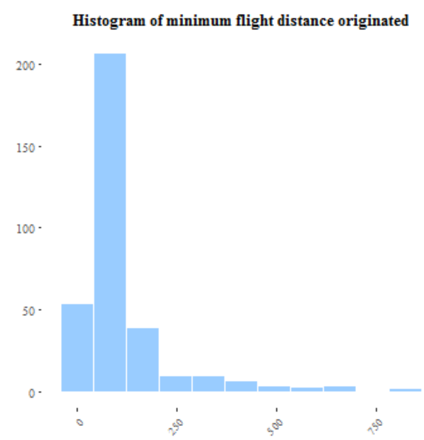
5.3.2 Models

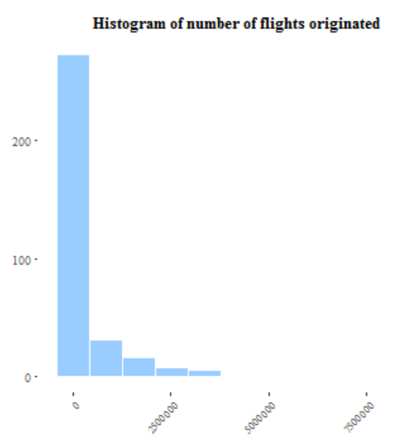
Before building any model I checked the type and in the relevant cases the distribution of the variables (for the outcome and for the predictor variables as well). The following table shows the type of the variables I've used in the model building:

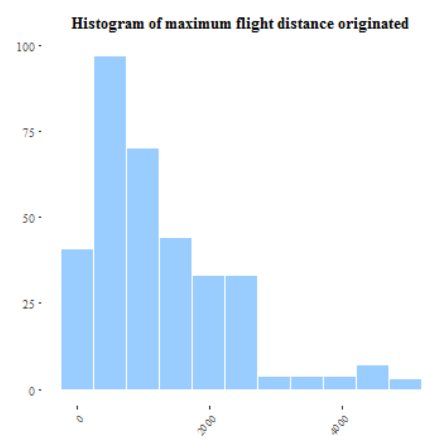
Variable name	Type of the variable	Modelling relevance	Check distribution
StrikeNo	Number	Outcome	Yes
Region	Categorical	Predictor	No
State	Categorical	Predictor	No
OriginCount	Integer	Predictor	Yes
OriginMaxDistance	Number	Predictor	Yes
OriginMinDistance	Number	Predictor	Yes
OriginAvgDistance	Number	Predictor	Yes
DestinationCount	Integer	Predictor	Yes
DestinationMaxDistance	Number	Predictor	Yes
DestinationMinDistance	Number	Predictor	Yes
DestinationAvgDistance	Number	Predictor	Yes
ARPElevation	Integer	Predictor	Yes
LandAreaCoveredByAirport	Number	Predictor	Yes

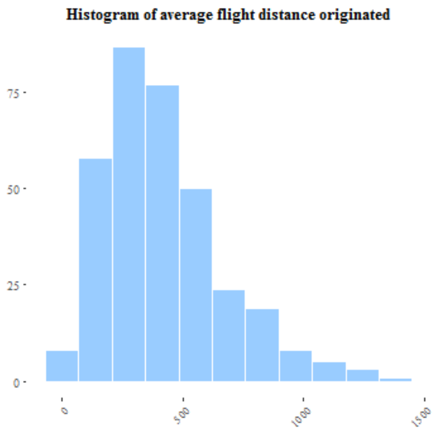
Checking the histograms of the relevant fields I saw that none of the fields have a normal distribution, instead all of them are left-skewed.

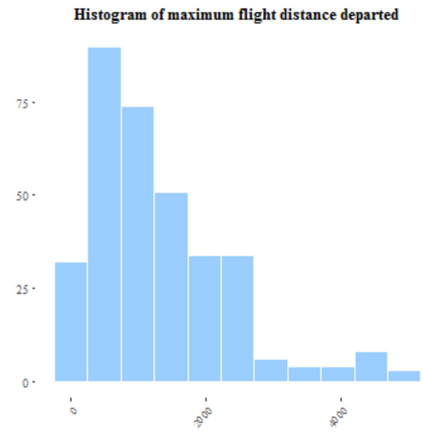


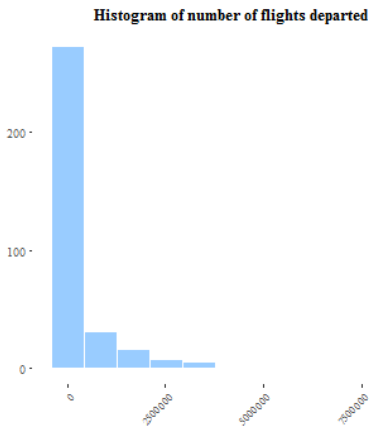


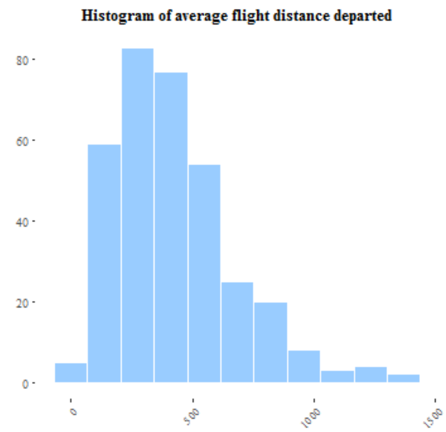


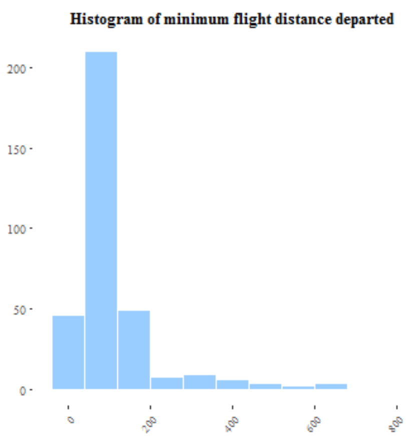


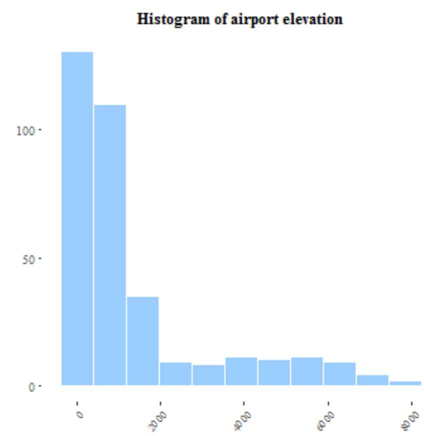

Histogram of average flight distance originated
0
25
50
75
0
500
1000
1500

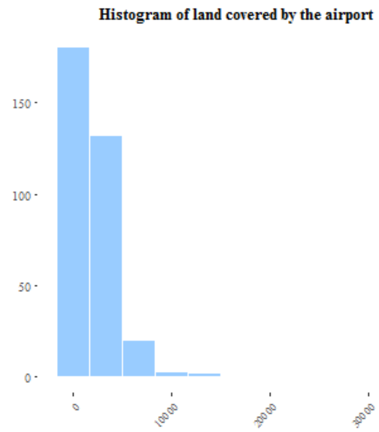

Histogram of maximum flight distance departed
0
25
50
75
0
2000
4000


Histogram of number of flights departed
0
100
200
0
2500000
5000000
7500000

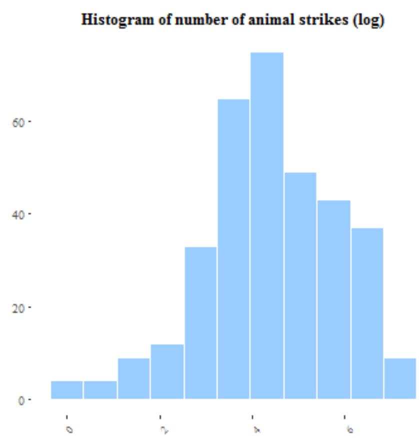

Histogram of average flight distance departed
0
20
40
60
80
0
500
1000
1500

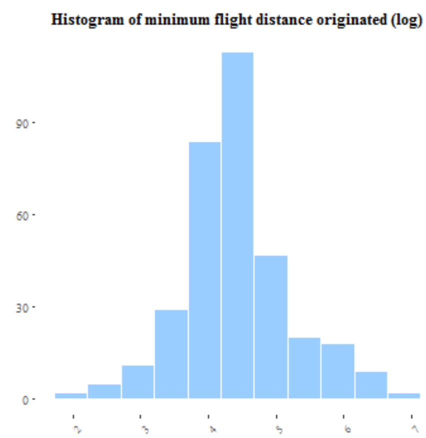

Histogram of minimum flight distance departed
0
50
100
150
200
0
200
400
600
800

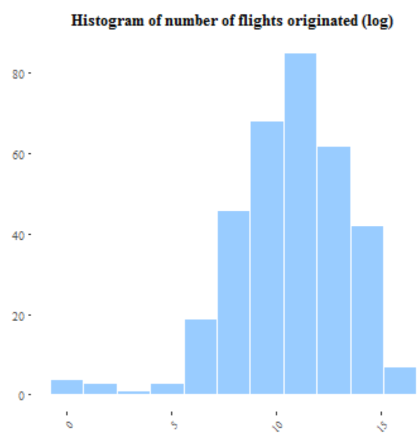

Histogram of airport elevation
0
50
100
0
2000
4000
6000
8000

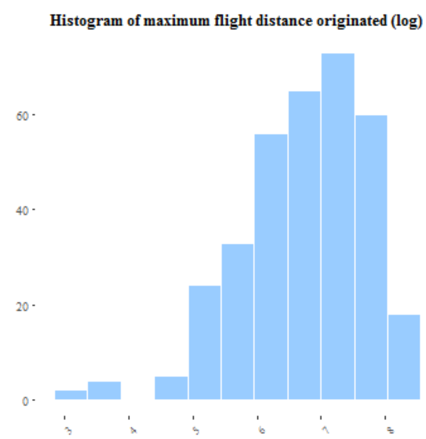


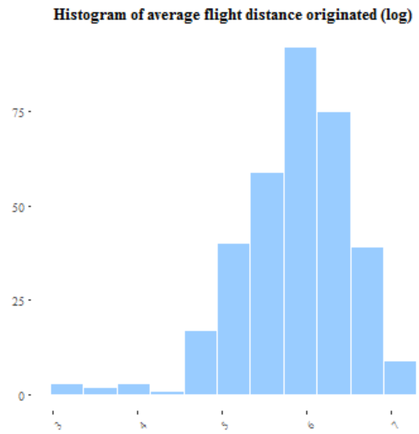
Resolving this issue can be done using the log of these fields, which will normalize the distributions as it is shown below.

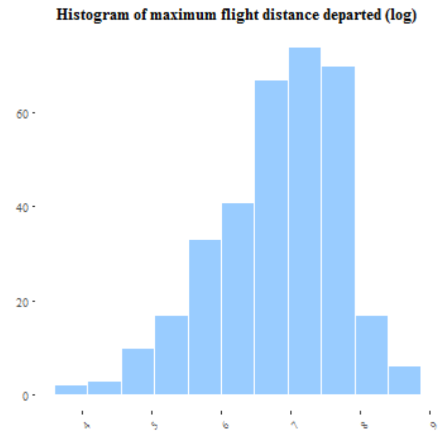


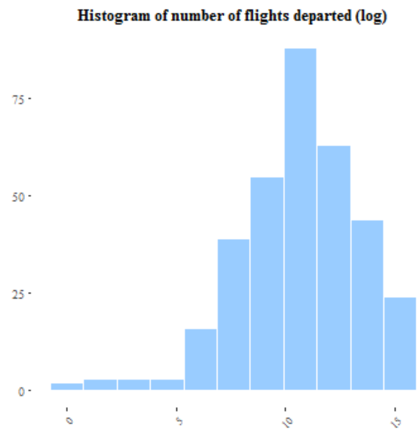


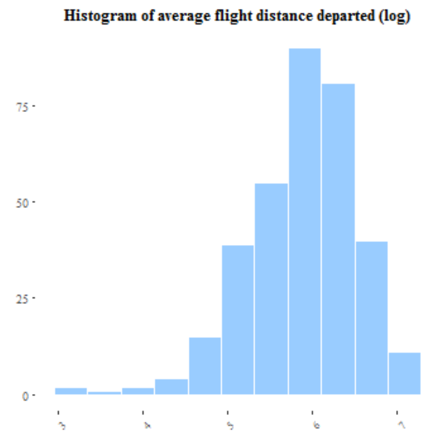


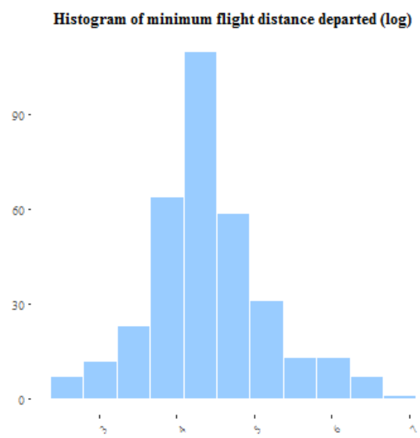


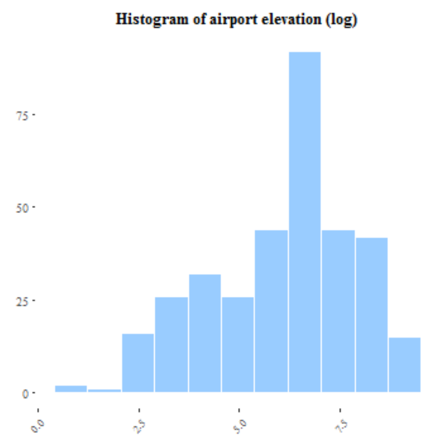

Histogram of average flight distance originated (log)

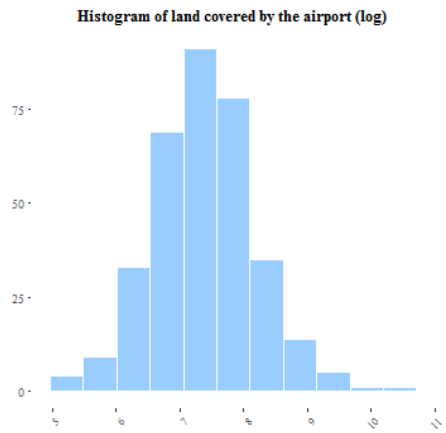

Histogram of maximum flight distance departed (log)


Histogram of number of flights departed (log)


Histogram of average flight distance departed (log)


Histogram of minimum flight distance departed (log)


Histogram of airport elevation (log)



Taking into account several combinations I've built the following models.

Model name	Predictors
Model01-01	OriginCount, DestinationCount
Model01-02	OriginCount, OriginMaxDistance, OriginMinDistance, OriginAvgDistance, DestinationCount, DestinationMaxDistance, DestinationMinDistance, DestinationAvgDistance
Model01-03	ARPElevation, LandAreaCoveredByAirport
Model01-04	Region
Model01-05	Region, State
Model01-06	State, OriginCount, DestinationCount, ARPElevation

5.3.3 Model Description

The first model (**Model01-01**) is built based on the assumption that the traffic generated by the airport has a significant impact on the number of animal strike on the given airport.

The second model (**Model01-02**) is an extension of the first model, since it contains data about the flights of the airport using the assumption that a longer distance requires a bigger airplane as well.

The third model (**Model01-03**) is taking into account only airport related details, namely the elevation of the airport above the see level and the land occupied by the airport.

The fourth model (**Model01-04**) is based on only the FAA regions and with the fifth model (**Model01-05**) they cover the possibilities from the airport location point of view (up to a reasonable level).

The sixth and last model (**Model01-06**) is combining the most predictors being evaluated as most significant ones from the previous models.

5.4 Assess Model for the first model

5.4.1 Model Assessment

All the linear regression models will be assessed using the following criteria:

Criteria	Description
Residuals	The expectation is that the residuals have a normal distribution, with a Median value close to 0.
Significance markings	A very low p-value is indicated as “***“, while a” ” is considered a high p-value. A lower p-value indicates that it’s more unlikely that there is no relationship between the outcome and the predictor variable.
Standard error of the coefficient estimate	this value is measuring the variability in the coefficient estimate. The value is relative to the coefficient estimate, in general lower is better.
p-value	The predictor variable probability of being not relevant. Lower value is better, since it means that the predictor is relevant.
R-squared	It’s the numeric representation of how well is the model fitting the data. In general higher is better, indicating a good correlation, but this does not mean causation.

5.4.1.1 Model01-01 Assessment

```
Call:
lm(formula = StrikeNo ~ OriginCount + DestinationCount, data = modelDataLog)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.0135 -0.9630 -0.0252  1.0351  2.5590 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)        1.8013     0.2956   6.093 3.03e-09 ***
OriginCount        0.4591     0.5409   0.849    0.397    
DestinationCount  -0.2147     0.5506  -0.390    0.697    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.269 on 337 degrees of freedom
Multiple R-squared:  0.2302,	Adjusted R-squared:  0.2256 
F-statistic: 50.38 on 2 and 337 DF,  p-value: < 2.2e-16
```

While the residuals seem to have a quite normal distribution none of the predictors is significant for the outcome. This model is not good, the remaining criteria do not need to be evaluated.

5.4.1.2 Model01-02 Assessment

```
Call:
lm(formula = StrikeNo ~ OriginCount + OriginMaxDistance + OriginMinDistance + 
    OriginAvgDistance + DestinationCount + DestinationMaxDistance + 
    DestinationMinDistance + DestinationAvgDistance, data = modelData)

Residuals:
    Min      1Q  Median      3Q     Max 
-467.68 -143.38  -69.86   57.14  969.11 
```

```
Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)            104.098565  28.474573   3.656 0.000298 ***
OriginCount              0.012548   0.005781   2.171 0.030675 *  
OriginMaxDistance        0.065088   0.054281   1.199 0.231354    
OriginMinDistance       -0.236468   0.192657  -1.227 0.220545    
OriginAvgDistance        0.197944   0.327524   0.604 0.546015    
DestinationCount        -0.012545   0.005773  -2.173 0.030474 *  
DestinationMaxDistance  -0.035235   0.050652  -0.696 0.487154    
DestinationMinDistance  -0.195390   0.201544  -0.969 0.333020    
DestinationAvgDistance   0.034932   0.323663   0.108 0.914118    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 234.2 on 331 degrees of freedom
Multiple R-squared:  0.1613,    Adjusted R-squared:  0.1411 
F-statistic:  7.96 on 8 and 331 DF,  p-value: 8.156e-10
```

In the second model the residuals are not distributed normally and even though the predictor significance got a little bit better, the model is still not acceptable. Just like in the previous case, the remaining criteria do not need to be evaluated.

5.4.1.3 Model01-03 Assessment

```
Call:
lm(formula = StrikeNo ~ ARPElevation + LandAreaCoveredByAirport, 
    data = modelData)

Residuals:
    Min      1Q  Median      3Q     Max 
-362.40 -152.38  -97.36   46.66 1086.43 

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)    
(Intercept)              188.383932  19.959855   9.438  < 2e-16 ***
ARPElevation              -0.022871   0.007570  -3.021  0.00271 ** 
LandAreaCoveredByAirport   0.013593   0.004987   2.726  0.00675 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 247.6 on 337 degrees of freedom
Multiple R-squared:  0.04548,   Adjusted R-squared:  0.03982 
F-statistic: 8.029 on 2 and 337 DF,  p-value: 0.0003923
```

While this model has a much better predictor significance, the distribution of the residuals can't be accepted, making the model rejected.

5.4.1.4 Model01-04 Assessment

```
Call:
lm(formula = StrikeNo ~ Region, data = modelData)

Residuals:
    Min      1Q  Median      3Q     Max 
```

```
-431.98  -96.98  -48.45   61.53  930.08

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   443.91      65.81   6.745 6.84e-11 ***
RegionACE    -306.51      81.94  -3.741 0.000216 ***
RegionAEA    -388.41      74.73  -5.197 3.55e-07 ***
RegionAGL    -343.08      71.59  -4.792 2.50e-06 ***
RegionANE    -249.16      91.11  -2.735 0.006583 **
RegionANM    -333.07      72.69  -4.582 6.54e-06 ***
RegionASO    -175.99      71.50  -2.461 0.014351 *
RegionASW    -329.54      73.26  -4.498 9.50e-06 ***
RegionAWP      13.07      73.93   0.177 0.859816
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 218.3 on 331 degrees of freedom
Multiple R-squared:  0.2712,    Adjusted R-squared:  0.2536
F-statistic:  15.4 on 8 and 331 DF,  p-value: < 2.2e-16
```

The forth is the first model where the predictor significance seems to be really good, along with an acceptable level of r-squared values, but I still have to reject the model because of the distribution of the residuals.

5.4.1.5 Model01-05 Assessment

```
Call:
lm(formula = StrikeNo ~ Region + State, data = modelData)

Residuals:
    Min      1Q  Median      3Q     Max
-643.00  -31.87   -0.66   28.84  890.09

Coefficients: (8 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  443.909     51.990   8.538 7.78e-16 ***
RegionACE   -368.909    100.678  -3.664 0.000295 ***
RegionAEA   -425.909    100.678  -4.230 3.13e-05 ***
RegionAGL   -424.534     80.122  -5.299 2.31e-07 ***
RegionANE   -415.909    180.098  -2.309 0.021627 *
RegionANM   -434.338     83.369  -5.210 3.59e-07 ***
RegionASO   -397.109     93.002  -4.270 2.65e-05 ***
RegionASW   -415.909     61.678  -6.743 8.37e-11 ***
RegionAWP   -362.909    112.311  -3.231 0.001374 **
StateAL      760.200    109.055   6.971 2.13e-11 ***
StateAR      648.750     92.381   7.023 1.55e-11 ***
StateAZ      600.750    131.696   4.562 7.50e-06 ***
StateCA      409.286    104.751   3.907 0.000116 ***
StateCO      344.429     84.975   4.053 6.49e-05 ***
StateCT      580.500    211.184   2.749 0.006357 **
StateDE       67.000    192.784   0.348 0.728438
StateFL      388.367     87.168   4.455 1.20e-05 ***
StateGA      224.575     98.301   2.285 0.023060 *
StateHI      275.429    118.989   2.315 0.021325 *
```

```
StateIA      193.200    115.670   1.670 0.095946 .
StateID      163.095     95.932   1.700 0.090181 .
StateIL      230.125     86.215   2.669 0.008032 **
StateIN      200.375    105.592   1.898 0.058737 .
StateKS       41.800    115.670   0.361 0.718084
StateKY      117.450    115.670   1.015 0.310768
StateLA      174.143     73.135   2.381 0.017905 *
StateMA       91.200    188.889   0.483 0.629585
StateMD      150.000    192.784   0.778 0.437159
StateME      137.500    211.184   0.651 0.515503
StateMI       83.092     75.490   1.101 0.271940
StateMN       60.125     86.215   0.697 0.486123
StateMO       12.167    111.304   0.109 0.913032
StateMS       17.343    100.965   0.172 0.863738
StateMT       50.000     92.168   0.542 0.587900
StateNC       33.644     96.177   0.350 0.726730
StateND       27.250     86.215   0.316 0.752178
StateNE           NA         NA      NA       NA
StateNH       87.000    243.854   0.357 0.721523
StateNJ       55.333    131.696   0.420 0.674681
StateNM       17.400     83.950   0.207 0.835948
StateNV           NA         NA      NA       NA
StateNY       47.667     97.032   0.491 0.623624
StateOH       52.825     98.301   0.537 0.591416
StateOK       24.000    104.938   0.229 0.819258
StateOR       29.429     92.168   0.319 0.749735
StatePA       39.143    108.077   0.362 0.717484
StateRI       22.000    243.854   0.090 0.928176
StateSC        1.200    109.055   0.011 0.991228
StateSD        8.875    105.592   0.084 0.933075
StateTN           NA         NA      NA       NA
StateTX           NA         NA      NA       NA
StateUT       -3.000     92.168  -0.033 0.974056
StateVA        7.571    108.077   0.070 0.944197
StateVT           NA         NA      NA       NA
StateWA       17.595     95.932   0.183 0.854601
StateWI           NA         NA      NA       NA
StateWV           NA         NA      NA       NA
StateWY           NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 172.4 on 290 degrees of freedom
Multiple R-squared:  0.6016,    Adjusted R-squared:  0.5342
F-statistic: 8.935 on 49 and 290 DF,  p-value: < 2.2e-16
```

The fifth model is showing the possibility of a higher significance of the airport location with a quite good residual distribution, but from interpretation point of you it’s a bit misleading as well, since the regions and the states are overlapping, making the predictors incorrect from the domain point of view.

5.4.1.6 Model01-06 Assessment

```
Call:
```

```
lm(formula = StrikeNo ~ State + OriginCount + DestinationCount + 
    ARPElevation, data = modelData)

Residuals:
    Min      1Q  Median      3Q     Max 
-630.04  -42.36   -0.23   38.09  866.93 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)       4.424e+02  4.890e+01   9.048  < 2e-16 ***
StateAL           3.631e+02  8.753e+01   4.148 4.42e-05 ***
StateAR           2.406e+02  9.483e+01   2.537 0.011708 *  
StateAZ           3.182e+02  1.077e+02   2.953 0.003403 ** 
StateCA           1.794e+01  5.844e+01   0.307 0.759111    
StateCO           9.015e+01  1.004e+02   0.898 0.369788    
StateCT           1.322e+02  1.248e+02   1.060 0.290065    
StateDE          -3.545e+02  1.693e+02  -2.094 0.037131 *  
StateFL          -4.329e+01  6.232e+01  -0.695 0.487861    
StateGA          -2.363e+02  7.649e+01  -3.089 0.002203 ** 
StateHI          -9.153e+01  7.842e+01  -1.167 0.244147    
StateIA          -1.551e+02  8.801e+01  -1.762 0.079052 .  
StateID          -1.573e+02  9.273e+01  -1.696 0.090912 .  
StateIL          -1.782e+02  7.719e+01  -2.308 0.021713 *  
StateIN          -2.263e+02  9.500e+01  -2.382 0.017862 *  
StateKS          -2.794e+02  8.925e+01  -3.130 0.001927 ** 
StateKY          -2.999e+02  9.521e+01  -3.150 0.001805 ** 
StateLA          -2.569e+02  7.841e+01  -3.276 0.001181 ** 
StateMA          -3.549e+02  8.768e+01  -4.047 6.66e-05 ***
StateMD          -4.106e+02  1.711e+02  -2.399 0.017066 *  
StateME          -2.828e+02  1.246e+02  -2.269 0.023994 *  
StateMI          -3.238e+02  6.486e+01  -4.991 1.04e-06 ***
StateMN          -3.369e+02  7.656e+01  -4.401 1.52e-05 ***
StateMO          -3.561e+02  8.309e+01  -4.286 2.48e-05 ***
StateMS          -3.731e+02  7.838e+01  -4.760 3.08e-06 ***
StateMT          -2.510e+02  9.285e+01  -2.703 0.007286 ** 
StateNC          -3.634e+02  7.318e+01  -4.967 1.17e-06 ***
StateND          -3.501e+02  7.719e+01  -4.535 8.47e-06 ***
StateNE          -3.186e+02  9.716e+01  -3.279 0.001171 ** 
StateNH          -3.382e+02  1.693e+02  -1.998 0.046702 *  
StateNJ          -4.218e+02  1.060e+02  -3.979 8.78e-05 ***
StateNM          -2.619e+02  1.035e+02  -2.531 0.011922 *  
StateNV          -3.222e+02  1.159e+02  -2.780 0.005787 ** 
StateNY          -3.920e+02  6.476e+01  -6.053 4.43e-09 ***
StateOH          -3.821e+02  8.823e+01  -4.331 2.05e-05 ***
StateOK          -3.974e+02  1.064e+02  -3.736 0.000226 ***
StateOR          -3.787e+02  7.964e+01  -4.756 3.13e-06 ***
StatePA          -4.001e+02  7.896e+01  -5.067 7.23e-07 ***
StateRI          -4.375e+02  1.695e+02  -2.581 0.010344 *  
StateSC          -4.003e+02  8.747e+01  -4.577 7.04e-06 ***
StateSD          -3.601e+02  9.690e+01  -3.717 0.000243 ***
StateTN          -3.953e+02  8.802e+01  -4.491 1.03e-05 ***
StateTX          -4.058e+02  5.912e+01  -6.864 4.15e-11 ***
StateUT          -3.153e+02  9.354e+01  -3.371 0.000851 ***
StateVA          -4.102e+02  7.859e+01  -5.219 3.46e-07 ***
```

```
StateVT          -4.159e+02  1.693e+02  -2.456 0.014640 *
StateWA          -4.177e+02  8.305e+01  -5.029 8.70e-07 ***
StateWI          -4.065e+02  7.599e+01  -5.349 1.81e-07 ***
StateWV          -3.856e+02  9.566e+01  -4.031 7.13e-05 ***
StateWY          -2.523e+02  1.030e+02  -2.450 0.014878 *
OriginCount       1.518e-02  4.111e-03   3.692 0.000266 ***
DestinationCount -1.512e-02  4.106e-03  -3.684 0.000275 ***
ARPElevation     -3.073e-02  1.149e-02  -2.674 0.007933 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 162.1 on 287 degrees of freedom
Multiple R-squared:  0.6516,    Adjusted R-squared:  0.5885
F-statistic: 10.32 on 52 and 287 DF,  p-value: < 2.2e-16
```

I consider the sixth and final model the best logistic regression model, as it has a quite normal residual distribution, very high significance for most of the predictors (some of the predictors are showing a very strong relation to the outcome) with good standard error for the predictors and very high r-squared values.

Taking into account that the model is a log-log model including categorical predictors (meaning that I used the natural log of the outcome and the continous predictor variables along with categorical predictors), the interpretation would go for the traffic originated from the airport as a one percent change in the originated traffic would be expected to have a 0.01518 percent change in the number of strikes is, compared to the number of strikes of Alaska.

5.4.2 Revised Parameter Settings

Keeping the modelling task as simple as possible I did not change the parameter settings for the models, instead I've used different predictor combinations to get the best possible statistical model from the data available.

5.5 Select Modeling Technique for the second model

5.5.1 Modeling Technique

Selecting the modelling technique I had to take into account two main restrictions and the business goal of the model. The first restriction was about the variety of the data I had, which was more of a needle in a haystack like data, instead of a nicely balanced data from the outcome point of view. The second restriction was about the volume of the data which made it impossible with the available resources to build the model in one single run. Additionally the main business goal was to build a prediction mode, so I had to select a model type fitting all these restrictions.

The modelling techniques I selected is the Gradient Boosting Machine (GBM) provided as one of the supervised algorithm in the H2O platform. The GBM technique can be used for both regression and classification models using forward learning ensemble method by building regression trees for all the predictors in the training data set. The H2O platform implementation is having a huge variety of parameters and can be easily used for a wide variety of model building exercises.

5.5.2 Modeling Assumptions

I did not take any modelling related assumption.

5.6 Generate Test Design for the second model

5.6.1 Test Design

The most efficient way of assessing the model would be to use a subset of the data set which was not used for training and for testing the model during the model building. Therefore I've split the data set to three sub-sets, containing 60% of the records for the training, 20% of the records for validating the model during the model building and 20% of the records for scoring the built model.

5.7 Build Model for the second model

5.7.1 Parameter Settings

The following parameters have been set for the GBM model:

Parameter name	Description	Initial	Final
x	Specify the columns to use as the predictor variables.	X	X
y	Specify the column to use as the outcome variable.	X	X
training_frame	Specify the data set used to build the model.	X	X
validation_frame	Specify the data set used to evaluate the accuracy of the model.	X	X
ntrees	Specify the number of trees to build.		X
learn_rate	Specify the learning rate.		X
max_depth	Specify the maximum tree depth.		X
stopping_rounds	Stops training when the option selected for stopping_metric doesn't improve for the specified number of training rounds, based on a simple moving average.		X
stopping_metric	Specify the metric to use for early stopping.		X
stopping_tolerance	Specify the relative tolerance for the metric-based stopping to stop training if the improvement is less than this value.		X
score_each_iteration	Specify whether to score during each iteration of the model training.		X
seed	Specify the random number generator (RNG) seed for algorithm components dependent on randomization.		X

Stopping parameters are defined to avoid unnecessary iterations without significant precision gain.

5.7.2 Models

I've built two model using different parameters (see the used parameters in the table above), first an initial model so that I could do a quick evaluation on the data and the model statistics, and secondly a final model using more parameters fine tuned to create a more precise result.

Based on the initial results the models have been built using different predictors, to keep the importance of the predictors in the final model in a reasonable level. The variables used in the models are the following:

Variable name	Type of the variable	Modelling relevance	Initial	Final
Year	Integer	Predictor	X	
Quarter	Integer	Predictor	X	X
Month	Integer	Predictor	X	X
DayofMonth	Integer	Predictor	X	X

Variable name	Type of the variable	Modelling relevance	Initial	Final
DayOfWeek	Integer	Predictor	X	X
FlightDate	Factor	Predictor	X	
UniqueCarrier	String	Predictor	X	X
FlightNum	Integer	Predictor	X	X
Origin	Factor	Predictor	X	X
Dest	Factor	Predictor	X	X
DepTimeBlk	Factor	Predictor	X	X
ArrTimeBlk	Factor	Predictor	X	X
CRSElapsedTime	Number	Predictor	X	X
Distance	Number	Predictor	X	X
DistanceGroup	Factor	Predictor	X	X
strikeFlag	Factor	Outcome	X	X

5.7.3 Model Description

To check the performance of the model parameters I've built the first model on the data from the year 1990. In this data sub-set I had 5,220,743 records, with only 9 records with strike data.

The evaluation of the first model indicated that the most probable reason behind the wrong performance is the lack of data variety with the huge difference in the data volume. There are multiple ways to overcome on these kind of situations. These techniques usually try to balance the outcome variable variety with reducing the number of records and/or generating additional records. Reducing the number of records works well in a situation when the variety and the volume of the data allows the reducing without significant impact on the results, while generating additional records can be used when the volume of the data is too small.

The situation of the data I'm working with requires to use both the reducing and generating/boosting, since the volume of records without strike is overwhelmingly bigger than the records with strikes. So for the second model I've modified the training data to take only the 10% of the non-strike records and add each strike record three times into the data set.

The data manipulation have resulted the following number of data records:

Number of flight records	Number of striked records	Percentage
15,447,169	85,353	0.5525%

5.8 Assess Model for the second model

5.8.1 Model Assessment

5.8.1.1 Initial model

The initial model (using almost only default settings) provided the following performance metrics:

5.8.1.1.1 Reported on training data.

The model performance metrics are the following on the train data set.

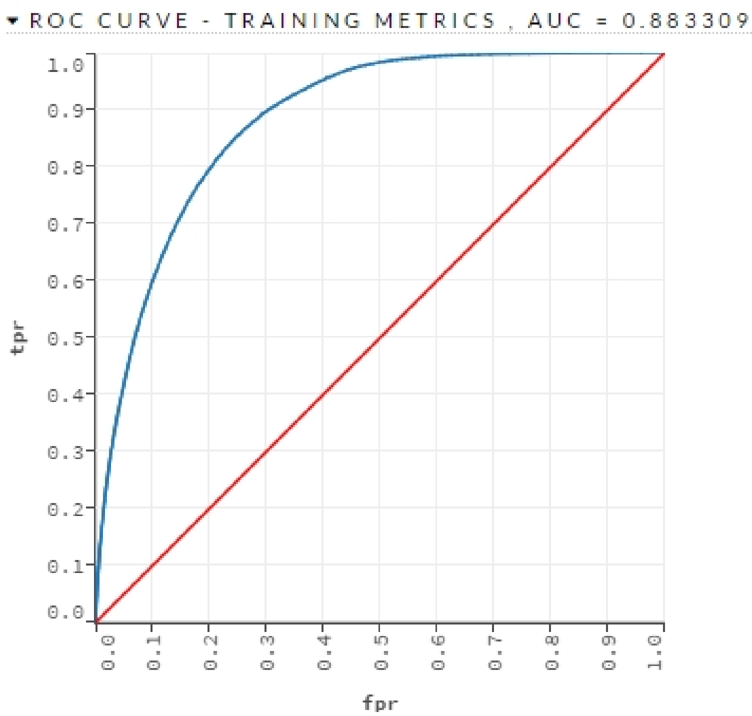
Metric name	Metric value
MSE	0.005354946
RMSE	0.0731775
LogLoss	0.0278349

Metric name	Metric value
Mean Per-Class Error	0.4181091
AUC	0.8833093
Gini	0.7666185

Confusion Matrix

	0	1	Error
0	9215132	109455	0.011738
1	42713	9093	0.824480
Totals	9257845	118548	0.016229

Receiver Operating Characteristic (ROC) curve



5.8.1.1.2 Reported on validation data.

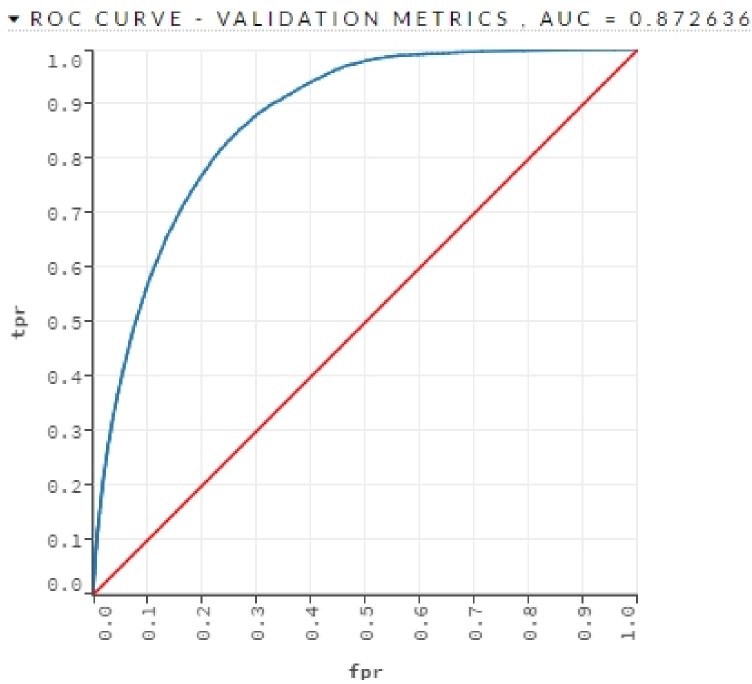
The model performance metrics are the following on the validation data set.

Metric name	Metric value
MSE	0.005418781
RMSE	0.07361237
LogLoss	0.02848955
Mean Per-Class Error	0.4132652
AUC	0.8726363
Gini	0.7452726

Confusion Matrix

	0	1	Error
0	3059226	48186	0.015507
1	14111	3288	0.811024
Totals	3073337	51474	0.019936

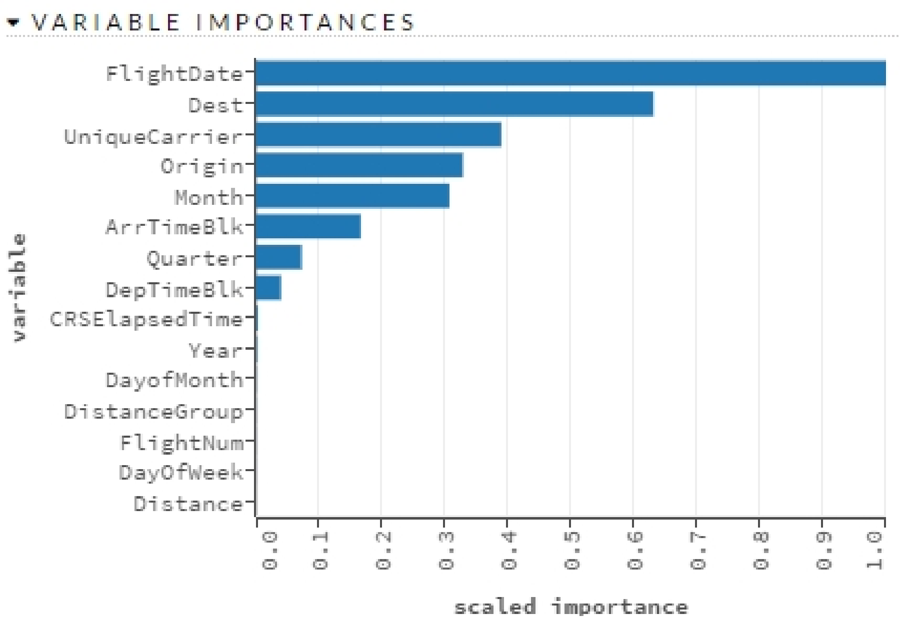
Receiver Operating Characteristic (ROC) curve



5.8.1.1.3 Variable Importances

The ten most importance predictor variables are the following:

	Variable	Relative	Scaled	Percentage
1	FlightDate	1700.398682	1.000000	0.340196
2	Dest	1072.529175	0.630752	0.214579
3	UniqueCarrier	662.508667	0.389620	0.132547
4	Origin	558.006104	0.328162	0.111639
5	Month	521.196716	0.306514	0.104275
6	ArrTimeBlk	282.533051	0.166157	0.056526
7	Quarter	122.561745	0.072078	0.024521
8	DepTimeBlk	66.628983	0.039184	0.013330
9	CRSElapsedTime	4.880882	0.002870	0.000977
10	Year	3.061660	0.001801	0.000613



5.8.1.1.4 Assessment

The performance of the model seemed to be excellent (AUCs of 0.8833 and 0.8726 with less than 2% incorrect hits) until I checked the variable importance and the false negative predictions.

The fact that the flight date got a very high relative importance and that the year predictor is in the top 10 as well, shows that for a future data set this model can't be used and the final model should not use these predictors.

The false negative hits in the model shows that even with the boosted (and reduced) data set the needle in the haystack effect is still present and in a real world scenario this model would not be very useful.

5.8.1.2 Final model

The final model using carefully selected model parameters provided the following performance metrics:

5.8.1.2.1 Reported on training data.

The model performance metrics are the following on the train data set.

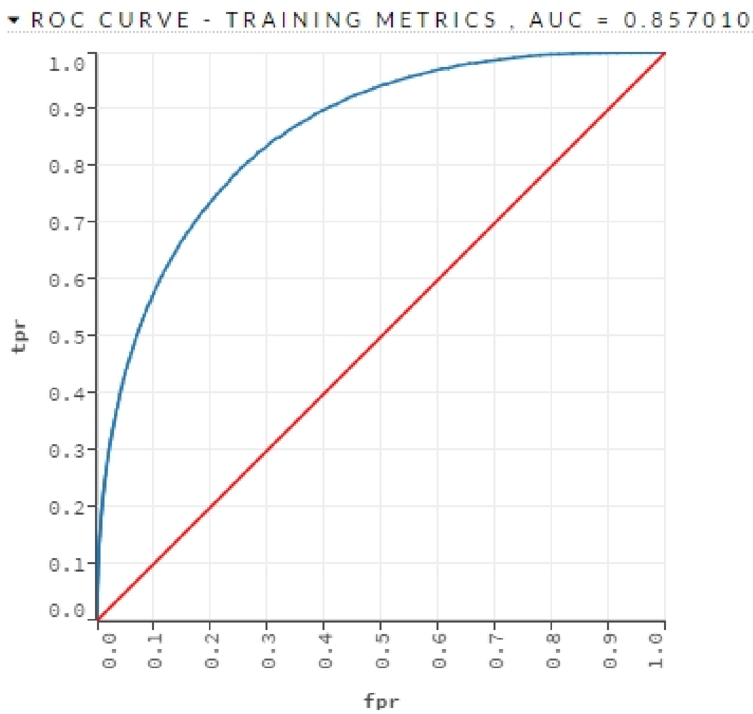
Metric name	Metric value
MSE	0.005349638
RMSE	0.07314122
LogLoss	0.02917852
Mean Per-Class Error	0.4221072
AUC	0.8570096
Gini	0.7140191

Confusion Matrix

	0	1	Error
0	9268544	56043	0.006010
1	43424	8382	0.838204

	0	1	Error
Totals	9311968	64425	0.010608

Receiver Operating Characteristic (ROC) curve



5.8.1.2.2 Reported on validation data.

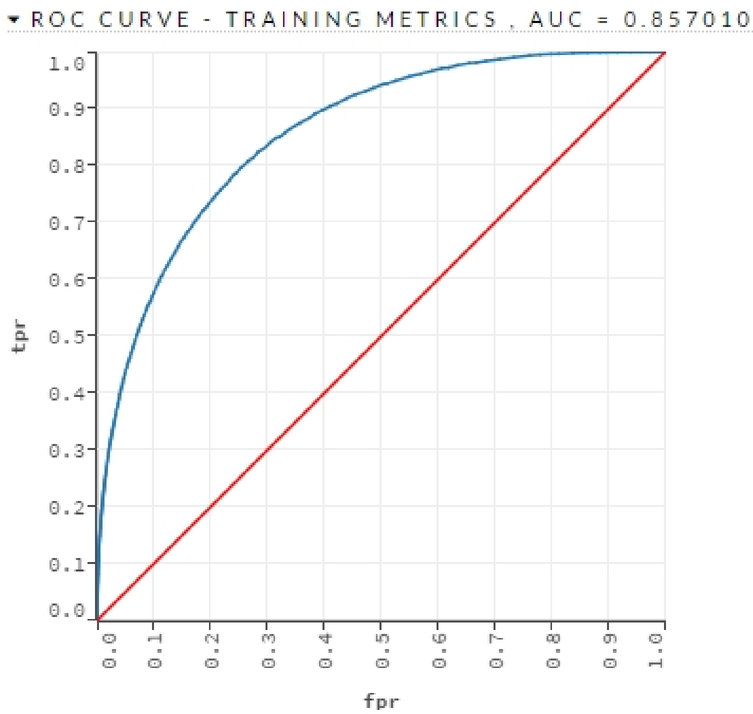
The model performance metrics are the following on the validation data set.

Metric name	Metric value
MSE	0.005611805
RMSE	0.07491198
LogLoss	0.03130175
Mean Per-Class Error	0.4426331
AUC	0.8215208
Gini	0.6430415

Confusion Matrix

	0	1	Error
0	3082632	24780	0.007974
1	15264	2135	0.877292
Totals	3097896	26915	0.012815

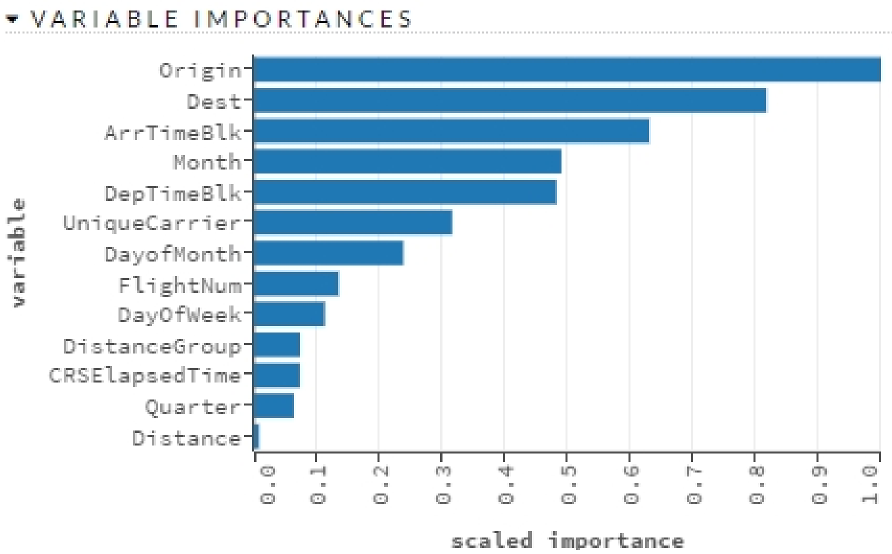
Receiver Operating Characteristic (ROC) curve



5.8.1.2.3 Variable Importances

The ten most importance predictor variables are the following:

	Variable	Relative	Scaled	Percentage
1	Origin	1068.777588	1.000000	0.225305
2	Dest	873.531250	0.817318	0.184146
3	ArrTimeBlk	673.985229	0.630613	0.142080
4	Month	523.576965	0.489884	0.110373
5	DepTimeBlk	515.629517	0.482448	0.108698
6	UniqueCarrier	337.862000	0.316120	0.071223
7	DayofMonth	254.428879	0.238056	0.053635
8	FlightNum	143.676224	0.134430	0.030288
9	DayOfWeek	120.171158	0.112438	0.025333
10	DistanceGroup	78.397850	0.073353	0.016527



5.8.1.2.4 Scoring

The scoring was done using the 20% of the records from data set separated at the beginning of the model building. The confusion matrix is the following:

	0	1	Error
0	3082474	25926	0.008341
1	14791	2224	0.869292
Totals	3097265	28150	0.013028

5.8.1.2.5 Assessment

While the changes in the model parameters and predictors made the model more useful from the business domain point of view, the performance of the model did not drop significantly (AUCs changed from 0.8833 to 0.8570 and from 0.8726 to 0.8215 with less than 1.3% incorrect hits). Unfortunately the same can be said about the false negative hits as well, which got even worst and still shows the needle in the haystack effect, making the model unrealistic for a real world scenario.

The scoring provided almost the same performance, what I got for the validation data assessment, confirming that the use of the model would not be beneficial in a real world scenario.

5.8.2 Revised Parameter Settings

I've did two main changes regarding the model parameters. The first major change was to remove those predictors, which were not useful for future runs, while the second major change was to increase the number of trees to be build from 50 to 200 while training the model and let the model building algorithm stop whenever more trees do not increase the model performance significantly.

6 Evaluation

6.1 Evaluate Results

6.1.1 Assessment of Data Mining Result with Business Success Criteria

The regressions and the models built and assessed shows that even with a quite limited and noisy data some level or certainty can be reached, but all these results needs to be used with caution.

Meaning that I was able to show that based on the data available some properties of airports could be identified having a significant effect on the number of animal strikes, and even for individual flights a prediction can be given for the strike, but there real world is much more complex, than the regressions and predictions build in this pet project.

6.1.2 Approved Models

As already mentioned in the initial resource plan, this final paper is a pet project, therefore the results of the project will never be put into a real production environment, meaning that no approval is required for any of the models created during the project.

6.2 Review Process

6.2.1 Review of Process

The data analysis and the model building exercises done during this project showed, that - even with a significant experience on these fields - the data can easily contradict any initial assumption made.

6.3 Determine Next Steps

6.3.1 List of Possible Actions

As already mentioned in the initial resource plan, this final paper is a pet project, therefore the results of the project will never be put into a real production environment, meaning that there will be no actions initiated by the results of the project.

6.3.2 Decision

As already mentioned in the initial resource plan, this final paper is a pet project, therefore the results of the project will never be put into a real production environment, meaning that no decisions are expected based on the results of the project.

7 Deployment

7.1 Plan Deployment

7.1.1 Deployment Plan

As already mentioned in the initial resource plan, this final paper is a pet project, therefore the results of the project will never be put into a real production environment, meaning that no deployment plan has to be created.

7.2 Plan Monitoring and Maintenance

7.2.1 Monitoring and Maintenance Plan

As already mentioned in the initial resource plan, this final paper is a pet project, therefore the results of the project will never be put into a real production environment, meaning that there will be entities to monitor and plan maintenance for.

7.3 Produce Final Report

7.3.1 Final Report

As already mentioned in the initial resource plan, this final paper is a pet project without real business stakeholders, therefore the results of the project will never be put into a real production environment, meaning that beyond the creation of this final paper and the expected presentation of the results of the project no additional reports are going to be created.

7.3.2 Final Presentation

The final presentation is the following:

Prediction of Animal Strike on US Commercial Flights

Gábor Horváth

Business objectives

- Create a statistical analysis to identify those reasons (based on the data available), which are determining the the risk of an animal strike for an airport.
- Create a prediction model, which can be used to predict the risk of an animal strike for a given flight.

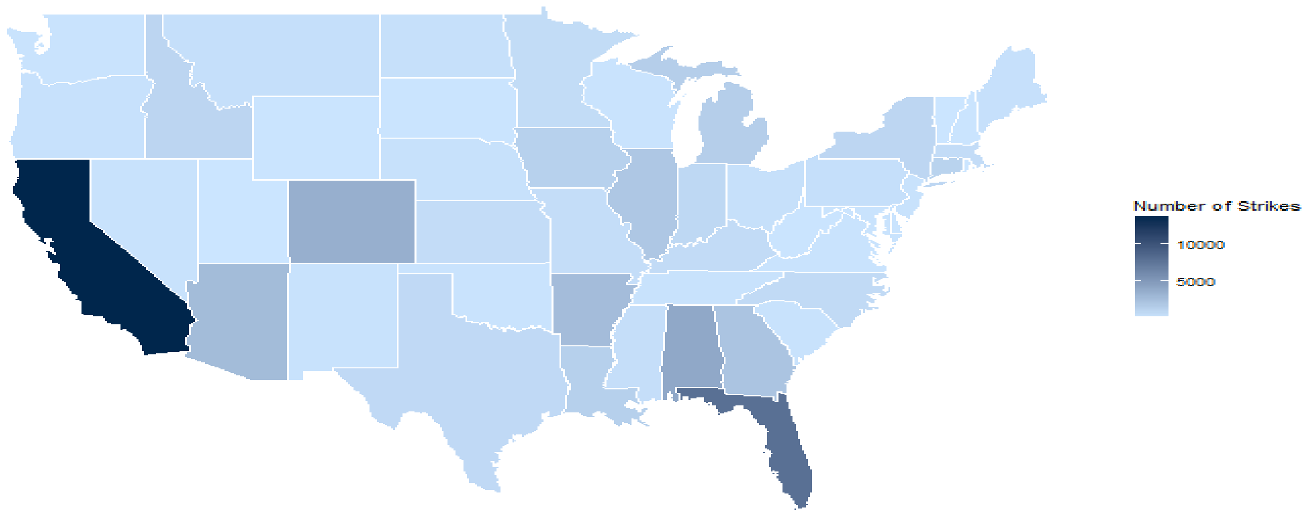
Data sources

- Federal Aviation Administration - Wildlife Strike Database
- United States Department of Transportation - Bureau of Transportation Statistics - Flight performance
- Federal Aviation Administration - Airport Data & Contact Information

Tools used

- R
- R Studio
- knitr, MiKTeX
- ggplot2
- H2O
- CRIPS-DM process model
- buckets, github

Strike distribution



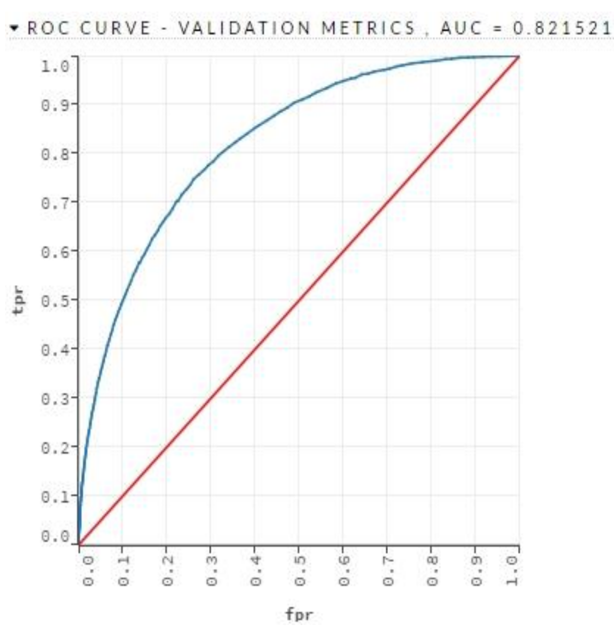
Model 01 - details

- log-log linear regression model
- categorical and continous predictor variables from airport point of view
- outcome variable: number of animal strike per airport
- location and traffic of the airport influence the outcome

Model 02 - details

- H2O platform - Java
- categorical and continous predictor variables from flight point of view
- outcome variable: flight got animal strike or not
- most influental predictors were the airports (origin, destination), the arrival and departure time blocks and the month

Receiver Operating Characteristic (ROC) curve



Confusion matrix

	0	1	Error
0	3082474	25926	0.008341
1	14791	2224	0.869292
Totals	3097265	28150	0.013028

Thank you

7.4 Review Project

7.4.1 Experience Documentation

The most remarkable remark is, that the use of the CRISP-DM process model helps covering all the aspects with ease of a data mining engagement. but of course there were several less significant observations as well, like:

- The quality of the date is rather bad in case it's based on voluntary manual reporting.
- The integration of different data sources might be more complex than the cleaning and exploring those data sets involved.
- Having data from goverment agencies (even if it's the same agency, but different office) can have notable quality differences.
- Using open source and freely available tools (including model building platforms as well) can give you the false sence of security when it comes to code quality (and hidden bugs).
- Never-ever underestimate the amunt of work required to clean, format and integrate the data.

8 Contributors

Student: Gábor Horváth
Mentor: Gergely Daróczi

9 Environment

The following language, tool and library versions have been used to create the project:

R Studio version 1.0.143

R version 3.4.0 (2017-04-21) 72570

Package versions:
- RODBC version 1.3.15
- knitr version 1.16
- data.table version 1.10.4
- dplyr version 0.7.0
- dtplyr version 0.0.2
- ReporteRs version 0.8.8
- ReporteRsjars version 0.0.2
- installr version 0.19.0
- stringr version 1.2.0
- ggplot2 version 2.2.1
- yaml version 2.1.14
- png version 0.1.7
- grid version 3.4.0
- maps version 3.2.0
- mapdata version 2.2.6
- sp version 1.2.4
- h2o version 3.10.5.2

Base package versions:
- stats version 3.4.0
- graphics version 3.4.0
- grDevices version 3.4.0
- utils version 3.4.0
- datasets version 3.4.0
- methods version 3.4.0
- base version 3.4.0

MiKTeX Package Manager 2.9.6200 (MiKTeX 2.9.6210 64-bit)
Copyright (C) 2005-2016 Christian Schenk
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Contents

10 Appendix 1 - Final Project Requirements

The following pages contain the Final project requirements received from the CEU Business School.

Final project

The goal of the final project is to expose the students in Business Analytics to a complete analytics workflow with a variety of tasks. They will use the full spectrum of skills acquired in the program, challenge themselves and learn something useful in the process and create value for the partner company. Throughout the project, students will interact with clients in the host company, analysts, IT engineers, and vendors of analytics solutions.

Examples

Insurance Ltd. sells many insurance products through a variety of channels. Customer data are stored in separate data silos for each market segment (e.g., life, home, car, travel), and there are often duplicates across sales channels (e.g., brokers do not check for existing customers but enter everyone as a new customer). In order to analyze customer behavior (e.g. churn) in all segments jointly, senior analysts need to merge all data by the same user. This requires entity resolution and unique user ID in all data silos. The student will study the various datasets, research entity resolution tools, conduct some tests with one or more prototype, and propose a solution to senior management.

Webstore Kft. is an online store of sporting goods. They want to evaluate the effectiveness of past social marketing campaigns. The management would like to know the average spending of new customers. Clickthrough rates are measured, but Webstore does not have information on conversion: if and what the newly acquired customers bought. Discussing with the person responsible for social campaigns, and the person running the website and maintaining the log, the student helps approximately identify new customers in the log and estimate their spending. She presents the results under alternative assumptions to the management. Together, they also propose a method for tracking conversion better.

Banking Ltd. is a financial company issuing credit cards. They have an existing model for predicting credit card non-payments which they want to improve. They have just launched a Hadoop project so they require a student with Hadoop expertise. Student meets with clients and analysts to understand current model and the need for improvement. Research current dataset and other data that can potentially be used to help predict default. Working as part of the analytics team, builds a prototype of a new machine learning model and tests its performance. Presents results to clients.

Resource needs

Each student has a **mentor** appointed by CEU and a **host** in their host company. The host company provides access to the necessary **space**, **people**, computer, software **tools** and **data**. The precise resource needs depend on the project and are negotiated in advance with the help of the mentor.

Benefits to host company

- Temporary staff with high technical skills and sensitive to the business environment; more dependable than entry-level interns.
- Consultations with CEU mentor.
- Access to latest technologies and trends.
- New perspective on a particular analytics problem or the analytics workflow.

Responsibilities

Student

- Select a host company and a project.
- Meet with mentor early on to discuss plans.
- Meet with mentor biweekly during the implementation of the project.
- Identify and understand business needs of host company clients.
- Select appropriate tools and provide best effort to address those needs.
- Complete deliverables by deadlines below.
- Maintain code of academic ethics, workplace rules of host company, and nondisclosure as agreed in project plan.
- Immediately raise concerns about project with mentor.

Mentor

- Help select a topic.
- Meet biweekly with the student to monitor progress and provide feedback.
- Verify project is feasible within the time frame.
- Discuss with host in case of concerns and problems.
- Verify successful project delivery at all stages.

Host

1. Propose analytics topics relevant to the host company.
2. Together with the mentor, identify the special needs in training, skills and tools.
3. Discuss with mentor and student the proposed project and agree on a plan.
4. Provide access for student to space, people, tools and data needed for successful completion of project.
5. Introduce student to other stakeholders at the company.

Deliverables

- Project plan. Describe the project and the resource needs in one page. Any special need in training, tools or any restrictions (e.g., non-disclosure agreement) should be specified here. Signed by student, host and mentor. Due April 3, 2017.
- Business needs. Student documents business needs as gathered from clients. User stories, scope of the project. Due April 30.
- Estimate of resource needs. Students estimates the resource needs of the project. Who needs to be involved? What time do they need to devote to the project? Any new software or data needs to purchased? Due April 30.
- Preliminary report. This contains the description of the business needs and the scope of the project, results of the analysis with exhibits, and recommendations for management. Due to host and mentor by June 30.

11 Appendix 2 - Project Plan

The following pages contain the Project Plan, which is the first deliverable described as the “Describe the project and the resource needs in one page. Any special need in training, tools or any restrictions (e.g., non-disclosure agreement) should be specified here.” in the final project requirements.

Project Plan of the Final Paper

for the CEU MSc in Business Analytics program

Gábor Horváth

2017

CEU | CENTRAL EUROPEAN UNIVERSITY

1 High level description

The goal of the project is to show - creating a risk evaluation of wildlife strikes of flights in the US - the techniques, methods, interpretations and understanding of the data analytic. The project is based on the Cross Industry Standard Process for Data Mining (CRISP-DM) process model, which is widely used worldwide for various scientific and business related data analytic projects. The use of the CRISP-DM process model will enable to for the project to cover all those areas (i.e. Business Understanding, Data understanding, Modelling, Evaluation, etc.), which are crucial of managing and delivering a successful data analytic project.

2 Resource needs

2.1 Training requirements

No additional organized / official training requirements are required above the trainings received during the courses in the program. There are tools and techniques used to fulfill the project which have not been described in the program at CEU, but there are several useful user manuals available on the webpages of the tool's creators, which would enable the use of these tools and resources for any student who have been part of the program.

2.2 Tools & resources used

Fulfilling the completion need for the project the following tools are planned to be used:

- Programming language:
  - R: https://www.r-project.org/
- IDE for the programming language:
  - RStudio: https://www.rstudio.com/
- Documentation is created using:
  - knitr: https://yihui.name/knitr/
  - MiKTeX: https://miktex.org/
  - ReporteRs: https://cran.r-project.org/web/packages/ReporteRs/index.html
- Data visualization:
  - ggplot2: http://ggplot2.org/
- Data manipulation:
  - access2csv: https://github.com/AccelerationNet/access2csv
  - dtplyr: https://cran.r-project.org/web/packages/dtplyr/index.html
- Project plan / task management:
  - Buckets: https://www.buckets.co/
- Source code repository:
  - GitHub: https://github.com/

Note: The list above do not contain the list of all the tools and packages used to create the project, but the full list will be provided in the source code.

2.3 Data sources

The project will use the following data provided by multiple US government agencies:

- Federal Aviation Administration: Wildlife Strike Database
- United States Department of Transportation: Bureau of Transportation Statistics

Note: In case data enrichment would be required for the successful risk modelling, additional data sources might be used as well. These possible additional data sources will be listed in the Final Paper.

2.4 Restrictions

Restrictions apply as per the restrictions set by the tools, data providers and owners of additional resources used. No additional restrictions have been identified and set regarding the use of the results of this project.

2.5 Contributors

Student: Gábor Horváth
Mentor: Gergely Daróczi

12 Appendix 3 - Business Needs

The following pages contain the Business Needs, which is the second deliverable described as the "Student documents business needs as gathered from clients. User stories, scope of the project." in the final project requirements.

Business needs of the Final Paper

for the CEU MSc in Business Analytics program

Gábor Horváth

2017

CEU CENTRAL EUROPEAN UNIVERSITY

1 Introduction

The structure of the document follows the Cross Industry Standard Process for Data Mining (CRISP-DM) process model, which is a non-proprietary, documented, and freely available data mining model (Shearer 2000). Whenever the model sections can be matched to (and can fulfill) the requirements stated by CEU for the Final Paper I'm using the appropriate section identified by the CRIPS-DM model. Please keep in mind that the model supports the full end-to-end process of a data mining project, but the project does not require the use of all the model elements.

2 Business understanding

2.1 Determine Business Objectives

2.1.1 Business Objectives

There are two main objectives what the project is aiming to complete.

1. Create a statistical analysis to identify those reasons (based on the data available), which are determining the the risk of an animal strike for an airport.
2. Create a prediction model, which can be used to predict the risk of an animal strike for a given flight.

The result of the statistical analysis could be used in the completion of the model building and evaluation the recommended order of the completion is the order of the objectives stated above.

2.1.2 Business Success Criteria

- Identification of features determining the risk potential of an airport
- Working model for animal strike prediction

2.2 Assess Situation

2.2.1 Inventory of resources

- Flight Data
- Animal Strike Data
- R
- Buckets

2.2.2 Requirements, Assumptions, and Constraints

- Additional requirements:
  - No additional requirements identified on top of the requirements already stated in this document.
- Assumptions
  - No initial assumptions made.
- Constraints
  - No initial hard constraints identified.

2.2.3 Risks and Contingencies

- Risks
  - No initial risks identified
- Contingencies
  - No initial contingencies identified

2.2.4 Terminology

The project is using different terminologies from the different domains. The terms/definitions used will not be marked or explained in details, if based on the context the reader can easily identify the domain of the particular term. In case there are uncertainties about a term (and it's not explained in the paper), the following sources can be used for the definitions:

- Aviation:
  - Aviation Terms / Directory: http://www.aviation-terms.com/index2.php
  - Aviation Glossary: http://www.aerofiles.com/glossary.html
  - Aviation Glossaries: https://www.flightsimaviation.com/_glossaries.html?s=aviation_terms
- Data Mining
  - Data Mining Glossary: http://www.thearling.com/glossary.htm
  - Data Mining - Terminologies: https://www.tutorialspoint.com/data_mining/dm_terminologies.htm
  - Data Mining and Predictive Analytics Glossary: http://www.kdnuggets.com/2015/06/data-mining-predictive-analytics-glossary.html
- Data Science / Big Data
  - Data Science Glossary: http://www.datascienceglossary.org/
  - Analytics and Big Data Glossary: http://data-informed.com/glossary-of-big-data-terms/
  - Data Science Glossary: http://www.kdnuggets.com/2015/09/data-science-glossary.html

2.2.5 Costs and Benefits

This is a one-man project, no significant cost is expected. Main benefit is to put to and almost end-to-end scenario the topics covered during the courses and discovering bits and bolts of the techniques for creating the project.

2.3 Determine Data Mining Goals

2.3.1 Data Mining Goals

- Understand, Analyse, Clean and Merge the source data correctly
- Create the required attributes
- Generate the required records (if applicable)

2.3.2 Data Mining Success Criteria

- Identification of featured determining the risk potential of an airport
- Working model for animal strike prediction

2.4 Produce Project Plan

2.4.1 Project Plan

The project is managed in an agile way, where all the tasks, requirements, issues, solutions, and ideas are kept in a project at buckets.

2.4.2 Initial Assessment of Tools and Techniques

- Programming language:
  - R: https://www.r-project.org/
- IDE for the programming language:
  - RStudio: https://www.rstudio.com/
- Documentation is created using:
  - knitr: https://yihui.name/knitr/
  - MiKTeX: https://miktex.org/
  - ReporteRs: https://cran.r-project.org/web/packages/ReporteRs/index.html
- Data visualization:

    - ggplot2: http://ggplot2.org/
- Data manipulation:
    - access2csv: https://github.com/AccelerationNet/access2csv
    - dtplyr: https://cran.r-project.org/web/packages/dtplyr/index.html
- Project plan / task management:
    - Buckets: https://www.buckets.co/
- Source code repository:
    - GitHub: https://github.com/

Note: The list above do not contain the list of all the tools and packages used to create the project, but the full list will be provided in the source code.

3 Data Understanding

3.1 Collect Initial Data

3.1.1 Initial Data Collection Report

This report will be part of the following documents:

- Preliminary Report
- Final Paper

3.2 Describe Data

3.2.1 Data Description Report

The two main data sources have the following column explanations, which is attached to the downloaded files as well, by the data provider agencies.

3.2.1.1 Animal strike data

Column name	Explanation of Column Name and Codes
INDEX NR	Individual record number
OPID	Airline operator code
OPERATOR	A three letter International Civil Aviation Organization code for aircraft operators. (BUS = business, PVT = private aircraft other than business, GOV = government aircraft, MIL - military aircraft.)
ATYPE	Aircraft
AMA	International Civil Aviation Organization code for Aircraft Make
AMO	International Civil Aviation Organization code for Aircraft Model
EMA	Engine Make Code (see Engine Codes tab below)
EMO	Engine Model Code (see Engine Codes tab below)
AC_CLASS	Type of aircraft (see Aircraft Type tab below)
AC_MASS	1 = 2,250 kg or less: 2 = ,2251-5700 kg: 3 = 5,701-27,000 kg: 4 = 27,001-272,000 kg: 5 = above 272,000 kg
NUM_ENGS	Number of engines
TYPE_ENG	Type of power A = reciprocating engine (piston): B = Turbojet: C = Turboprop: D = Turbofan: E = None (glider): F = Turboshaft (helicopter): Y = Other
ENG_1_POS	Where engine # 1 is mounted on aircraft (see Engine Position tab below)
ENG_2_POS	Where engine # 2 is mounted on aircraft (see Engine Position tab below)
ENG_3_POS	Where engine # 3 is mounted on aircraft (see Engine Position tab below)
ENG_4_POS	Where engine # 4 is mounted on aircraft (see Engine Position tab below)
REG	Aircraft registration
FLT	Flight number
REMAINS_COLLECTED	Indicates if bird or wildlife remains were found and collected
REMAINS_SENT	Indicates if remains were sent to the Smithsonian Institution for identification
INCIDENT_DATE	Date strike occurred
INCIDENT_MONTH	Month strike occurred
INCIDENT_YEAR	Year strike occurred
TIME_OF_DAY	Light conditions
TIME	Hour and minute in local time

Column name	Explanation of Column Name and Codes
AIRPORT_ID	International Civil Aviation Organization airport identifier for location of strike whether it was on or off airport
AIRPORT	Name of airport
STATE	State
FAAREGION	FAA Region where airport is located
ENROUTE	If strike did not occur on approach, climb, landing roll, taxi or take-off, aircraft was enroute. This shows location.
RUNWAY	Runway
LOCATION	Various information about aircraft location if enroute or airport where strike evidence was found. Some locations show the two airports for the flight departure and arrival if pilot was unaware of the strike.
HEIGHT	Feet Above Ground Level
SPEED	Knots (indicated air speed)
DISTANCE	Miles from airport
PHASE_OF_FLT	Phase of flight during which strike occurred
DAMAGE	
Blank	Unknown
M = minor	When the aircraft can be rendered airworthy by simple repairs or replacements and an extensive inspection is not necessary.
M? = uncertain level	The aircraft was damaged, but details as to the extent of the damage are lacking.
S = substantial	When the aircraft incurs damage or structural failure which adversely affects the structure strength, performance or flight characteristics of the aircraft and which would normally require major repair or replacement of the affected component.
D = Destroyed	When the damage sustained makes it inadvisable to restore the aircraft to an airworthy condition.
STR_RAD	Struck radome
DAM_RAD	Damaged radome
STR_WINDSHLD	Struck windshield
DAM_WINDSHLD	Damaged windshield
STR_NOSE	Struck nose
DAM_NOSE	Damaged nose
STR_ENG1	Struck Engine 1
DAM_ENG1	Damaged Engine 1
STR_ENG2	Struck Engine 2
DAM_ENG2	Damaged Engine 2
STR_ENG3	Struck Engine 3
DAM_ENG3	Damaged Engine 3
STR_ENG4	Struck Engine 4
DAM_ENG4	Damaged Engine 4
INGESTED	Engine ingested the bird/ animal
STR_PROP	Struck Propeller
DAM_PROP	Damaged Propeller
STR_WING_ROT	Struck Wing or Rotor
DAM_WING_ROT	Damaged Wing or Rotor
STR_FUSE	Struck Fuselage
DAM_FUSE	Damaged Fuselage
STR_LG	Struck Landing Gear
DAM_LG	Damaged Landing Gear
STR_TAIL	Struck Tail
DAM_TAIL	Damaged Tail

Column name	Explanation of Column Name and Codes
STR_LGHTS	Struck Lights
DAM_LGHTS	Damaged Lights
STR_OTHER	Struck Other than parts shown above
DAM_OTHER	Damaged Other than parts shown above
OTHER_SPECIFY	What part was struck other than those listed above
EFFECT	Effect on flight
EFFECT_OTHER	Effect on flight other than those listed on the form
SKY	Type of cloud cover, if any
PRECIP	Precipitation
SPECIES_ID	International Civil Aviation Organization code for type of bird or other wildlife
SPECIES	Common name for bird or other wildlife
BIRDS_SEEN	Number of birds/wildlife seen by pilot
BIRDS_STRUCK	Number of birds/wildlife struck
SIZE	Size of bird as reported by pilot is a relative scale. Entry should reflect the perceived size as opposed to a scientifically determined value. If more than one species was struck, larger bird is entered.
WARNED	Pilot warned of birds/wildlife
COMMENTS	As entered by database manager. Can include name of aircraft owner, types of reports received, updates, etc.
REMARKS	Most of remarks are from the form but some are data entry notes and are usually in parentheses.
AOS	Time aircraft was out of service in hours. If unknown, it is blank.
COST_REPAIRS	Estimated cost of repairs of replacement in dollars (USD)
COST_OTHER	Estimated other costs, other than those in previous field in dollars (USD). May include loss of revenue, hotel expenses due to flight cancellation, costs of fuel dumped, etc.
COST_REPAIRS_INFL_ADJ	Costs adjusted for inflation
COST_OTHER_INFL_ADJ	Other cost adjusted for inflation
REPORTED_NAME	Name(s) of person(s) filing report
REPORTED_TITLE	Title(s) of person(s) filing report
REPORTED_DATE	Date report was written
SOURCE	Type of report. Note: for multiple types of reports this will be indicated as Multiple. See "Comments" field for details
PERSON	Only one selection allowed. For multiple reports, see field "Reported Title"
NR_INJURIES	Number of people injured
NR_FATALITIES	Number of human fatalities
LUPDATE	Last time record was updated
TRANSFER	Unused field at this time
INDICATED_DAMAGE	Indicates whether or not aircraft was damaged

3.2.1.2 Flight data

Column name	Explanation of Column Name and Codes
Year	Year
Quarter	Quarter (1-4)
Month	Month
DayofMonth	Day of Month
DayOfWeek	Day of Week
FlightDate	Flight Date (yyyymmdd)

Column name	Explanation of Column Name and Codes
UniqueCarrier	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.
AirlineID	An identification number assigned by US DOT to identify a unique airline (carrier). A unique airline (carrier) is defined as one holding and reporting under the same DOT certificate regardless of its Code, Name, or holding company/corporation.
Carrier	Code assigned by IATA and commonly used to identify a carrier. As the same code may have been assigned to different carriers over time, the code is not always unique. For analysis, use the Unique Carrier Code.
TailNum	Tail Number
FlightNum	Flight Number
OriginAirportID	Origin Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
OriginAirportSeqID	Origin Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.
OriginCityMarketID	Origin Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
Origin	Origin Airport
OriginCityName	Origin Airport, City Name
OriginState	Origin Airport, State Code
OriginStateFips	Origin Airport, State Fips
OriginStateName	Origin Airport, State Name
OriginWac	Origin Airport, World Area Code
DestAirportID	Destination Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
DestAirportSeqID	Destination Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.
DestCityMarketID	Destination Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
Dest	Destination Airport
DestCityName	Destination Airport, City Name
DestState	Destination Airport, State Code
DestStateFips	Destination Airport, State Fips
DestStateName	Destination Airport, State Name
DestWac	Destination Airport, World Area Code
CRSDepTime	CRS Departure Time (local time: hhmm)
DepTime	Actual Departure Time (local time: hhmm)
DepDelay	Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
DepDelayMinutes	Difference in minutes between scheduled and actual departure time. Early departures set to 0.
DepDel15	Departure Delay Indicator, 15 Minutes or More (1=Yes)
DepartureDelayGroups	Departure Delay intervals, every (15 minutes from <-15 to >180)
DepTimeBlk	CRS Departure Time Block, Hourly Intervals
TaxiOut	Taxi Out Time, in Minutes
WheelsOff	Wheels Off Time (local time: hhmm)

Column name	Explanation of Column Name and Codes
WheelsOn	Wheels On Time (local time: hhmm)
TaxiIn	Taxi In Time, in Minutes
CRSArrTime	CRS Arrival Time (local time: hhmm)
ArrTime	Actual Arrival Time (local time: hhmm)
ArrDelay	Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
ArrDelayMinutes	Difference in minutes between scheduled and actual arrival time. Early arrivals set to 0.
ArrDel15	Arrival Delay Indicator, 15 Minutes or More (1=Yes)
ArrivalDelayGroups	Arrival Delay intervals, every (15-minutes from <-15 to >180)
ArrTimeBlk	CRS Arrival Time Block, Hourly Intervals
Cancelled	Cancelled Flight Indicator (1=Yes)
CancellationCode	Specifies The Reason For Cancellation
Diverted	Diverted Flight Indicator (1=Yes)
CRSElapsedTime	CRS Elapsed Time of Flight, in Minutes
ActualElapsedTime	Elapsed Time of Flight, in Minutes
AirTime	Flight Time, in Minutes
Flights	Number of Flights
Distance	Distance between airports (miles)
DistanceGroup	Distance Intervals, every 250 Miles, for Flight Segment
CarrierDelay	Carrier Delay, in Minutes
WeatherDelay	Weather Delay, in Minutes
NASDelay	National Air System Delay, in Minutes
SecurityDelay	Security Delay, in Minutes
LateAircraftDelay	Late Aircraft Delay, in Minutes
FirstDepTime	First Gate Departure Time at Origin Airport
TotalAddGTime	Total Ground Time Away from Gate for Gate Return or Cancelled Flight
LongestAddGTime	Longest Time Away from Gate for Gate Return or Cancelled Flight
DivAirportLandings	Number of Diverted Airport Landings
DivReachedDest	Diverted Flight Reaching Scheduled Destination Indicator (1=Yes)
DivActualElapsedTime	Elapsed Time of Diverted Flight Reaching Scheduled Destination, in Minutes. The ActualElapsedTime column remains NULL for all diverted flights.
DivArrDelay	Difference in minutes between scheduled and actual arrival time for a diverted flight reaching scheduled destination. The ArrDelay column remains NULL for all diverted flights.
DivDistance	Distance between scheduled destination and final diverted airport (miles). Value will be 0 for diverted flight reaching scheduled destination.
Div1Airport	Diverted Airport Code1
Div1AirportID	Airport ID of Diverted Airport 1. Airport ID is a Unique Key for an Airport
Div1AirportSeqID	Airport Sequence ID of Diverted Airport 1. Unique Key for Time Specific Information for an Airport
Div1WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code1
Div1TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code1
Div1LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code1
Div1WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code1
Div1TailNum	Aircraft Tail Number for Diverted Airport Code1
Div2Airport	Diverted Airport Code2
Div2AirportID	Airport ID of Diverted Airport 2. Airport ID is a Unique Key for an Airport
Div2AirportSeqID	Airport Sequence ID of Diverted Airport 2. Unique Key for Time Specific Information for an Airport
Div2WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code2
Div2TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code2

Column name	Explanation of Column Name and Codes
Div2LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code2
Div2WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code2
Div2TailNum	Aircraft Tail Number for Diverted Airport Code2
Div3Airport	Diverted Airport Code3
Div3AirportID	Airport ID of Diverted Airport 3. Airport ID is a Unique Key for an Airport
Div3AirportSeqID	Airport Sequence ID of Diverted Airport 3. Unique Key for Time Specific Information for an Airport
Div3WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code3
Div3TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code3
Div3LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code3
Div3WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code3
Div3TailNum	Aircraft Tail Number for Diverted Airport Code3
Div4Airport	Diverted Airport Code4
Div4AirportID	Airport ID of Diverted Airport 4. Airport ID is a Unique Key for an Airport
Div4AirportSeqID	Airport Sequence ID of Diverted Airport 4. Unique Key for Time Specific Information for an Airport
Div4WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code4
Div4TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code4
Div4LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code4
Div4WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code4
Div4TailNum	Aircraft Tail Number for Diverted Airport Code4
Div5Airport	Diverted Airport Code5
Div5AirportID	Airport ID of Diverted Airport 5. Airport ID is a Unique Key for an Airport
Div5AirportSeqID	Airport Sequence ID of Diverted Airport 5. Unique Key for Time Specific Information for an Airport
Div5WheelsOn	Wheels On Time (local time: hhmm) at Diverted Airport Code5
Div5TotalGTime	Total Ground Time Away from Gate at Diverted Airport Code5
Div5LongestGTime	Longest Ground Time Away from Gate at Diverted Airport Code5
Div5WheelsOff	Wheels Off Time (local time: hhmm) at Diverted Airport Code5
Div5TailNum	Aircraft Tail Number for Diverted Airport Code5

3.3 Explore Data

3.3.1 Data Exploration Report

This report will be part of the following documents:

- Preliminary Report
- Final Paper

3.4 Verify Data Quality

3.4.1 Data Quality Report

This report will be part of the following documents:

- Preliminary Report
- Final Paper

4 Contributors

Student: Gábor Horváth
Mentor: Gergely Daróczi

Contents

References

Shearer, Colin. 2000. "The Crisp-Dm Model - the New Blueprint for Data Mining." Journal of Data Warehousing 5 (4): 13–22.

13 Appendix 4 - Estimate of Resource Needs

The following pages contain the Estimate of Resource Needs, which is the third deliverable described as the “Students estimates the resource needs of the project. Who needs to be involved? What time do they need to devote to the project? Any new software or data needs to purchased?” in the final project requirements.

Estimate of resource needs of the Final Paper

for the CEU MSc in Business Analytics program

Gábor Horváth

2017

CENTRAL EUROPEAN UNIVERSITY

1 Human resource needs

1.1 Stakeholders & people to involve

As this final paper is a pet project, there is no actual business management behind the requirements, therefore no business stakeholders are identified and involved. The completion of the project requires feedback and guidance from the mentor (Gergely Daroczi), but no other person (or role) needs to be involved.

1.2 Dedication for the project

There are no additional time dedication requirements identified above the requirements stated by CEU in the Final Project description document.

1.3 Training requirements

As stated earlier no additional organized / official training requirements are required above the trainings received during the courses in the program. There are tools and techniques used to fulfill the project, which have not been described in the program at CEU. There are several useful user manuals available on the webpages of the creators of the tools, which would enable the use of these tools and resources for any student who have been part of the program.

2 Software and data resource needs

2.1 Tools & resources used

As stated earlier, fulfilling the completion need for the project the following tools are planned to be used:

- Programming language:
  - R: https://www.r-project.org/
- IDE for the programming language:
  - RStudio: https://www.rstudio.com/
- Documentation is created using:
  - knitr: https://yihui.name/knitr/
  - MiKTeX: https://miktex.org/
  - ReporteRs: https://cran.r-project.org/web/packages/ReporteRs/index.html
- Data visualization:
  - ggplot2: http://ggplot2.org/
- Data manipulation:
  - access2csv: https://github.com/AccelerationNet/access2csv
  - dtplyr: https://cran.r-project.org/web/packages/dtplyr/index.html
- Project plan / task management:
  - Buckets: https://www.buckets.co/
- Source code repository:
  - GitHub: https://github.com/

Note: The list above do not contain the list of all the tools and packages used to create the project, but the full list will be provided in the source code.

2.2 Data sources

As stated earlier, the project will use the following data provided by multiple US government agencies:

- Federal Aviation Administration: Wildlife Strike Database
- United States Department of Transportation: Bureau of Transportation Statistics

Note: In case data enrichment would be required for the successful risk modelling, additional data sources might be used as well. These possible additional data sources will be listed in the Final Paper.

3 Contributors

Student: Gábor Horváth
Mentor: Gergely Daróczi

14 Appendix 5 - Full Data Exploration Report (1990-2016)

14.1 Data Exploration Report (1990 - 2016)

14.1.1 Animal Strike Data (1990 - 2016)

The first summary table shows the number of distinct items for each year regarding the Airline operators, Aircraft, Aircraft types, Aircraft mass types, and Engine types, which have been reported as being affected in an animal strike. (Please note that the data for 2016 is available until 30-4-2016.)

Year	# of reports	Operators	Aircraft	Aircraft type	Aircraft mass type	Engine type
1990	1847	316	329	4	5	9
1991	2388	316	329	4	5	9
1992	2566	316	329	4	5	9
1993	2575	316	329	4	5	9
1994	2635	316	329	4	5	9
1995	2768	316	329	4	5	9
1996	2936	316	329	4	5	9
1997	3455	316	329	4	5	9
1998	3799	316	329	4	5	9
1999	5113	316	329	4	5	9
2000	6000	353	394	3	5	9
2001	5820	353	394	3	5	9
2002	6225	353	394	3	5	9
2003	6002	353	394	3	5	9
2004	6561	353	394	3	5	9
2005	7227	353	394	3	5	9
2006	7240	353	394	3	5	9
2007	7745	353	394	3	5	9
2008	7632	353	394	3	5	9
2009	9508	353	394	3	5	9
2010	9904	281	392	4	5	10
2011	10115	281	392	4	5	10
2012	10905	281	392	4	5	10
2013	11403	281	392	4	5	10
2014	13692	281	392	4	5	10
2015	13167	281	392	4	5	10
2016	1391	281	392	4	5	10

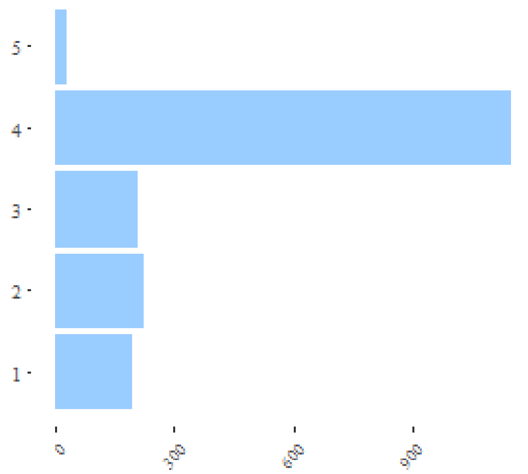
The second summary table shows the number of distinct items for each year regarding the Time of day, Airports, States, Phase of flight, weather conditions (Sky and Precipitation), and the flag for showing if the pilot has been warned or not about birds / wildlife in the reports. (Please note that the data for 2016 is available until 30-4-2016.)

Year	Time of day	Airports	States	Phase of flight	Sky	Precipitation	Warned
1990	5	1175	61	12	7	8	4
1991	5	1175	61	12	7	8	4
1992	5	1175	61	12	7	8	4
1993	5	1175	61	12	7	8	4
1994	5	1175	61	12	7	8	4
1995	5	1175	61	12	7	8	4
1996	5	1175	61	12	7	8	4
1997	5	1175	61	12	7	8	4

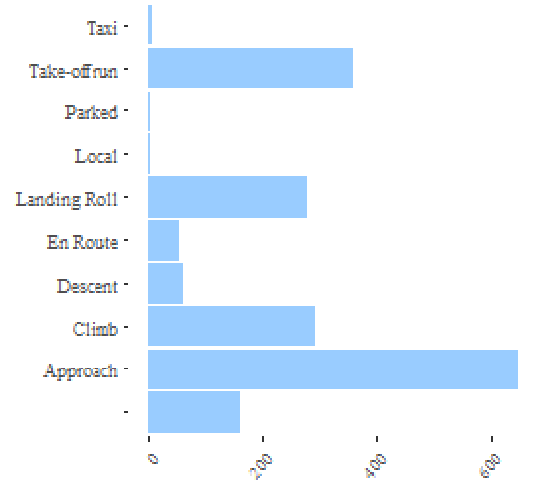
Year	Time of day	Airports	States	Phase of flight	Sky	Precipitation	Warned
1998	5	1175	61	12	7	8	4
1999	5	1175	61	12	7	8	4
2000	7	1499	63	12	5	9	5
2001	7	1499	63	12	5	9	5
2002	7	1499	63	12	5	9	5
2003	7	1499	63	12	5	9	5
2004	7	1499	63	12	5	9	5
2005	7	1499	63	12	5	9	5
2006	7	1499	63	12	5	9	5
2007	7	1499	63	12	5	9	5
2008	7	1499	63	12	5	9	5
2009	7	1499	63	12	5	9	5
2010	5	1497	63	12	4	9	5
2011	5	1497	63	12	4	9	5
2012	5	1497	63	12	4	9	5
2013	5	1497	63	12	4	9	5
2014	5	1497	63	12	4	9	5
2015	5	1497	63	12	4	9	5
2016	5	1497	63	12	4	9	5

The following graphs show the distributions of some of the selected distinct items summarized in the tables above.

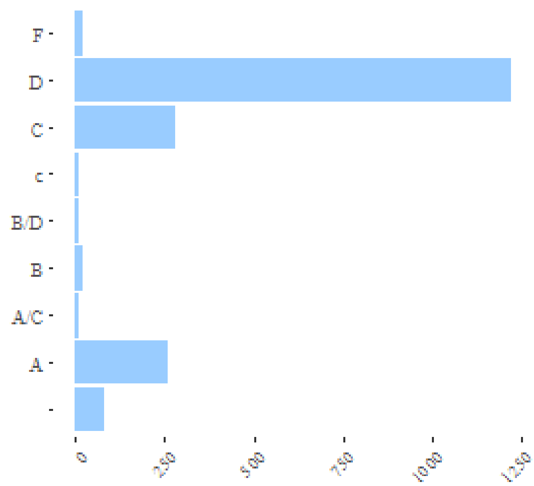
Data distribution of aircraft mass type in 1990



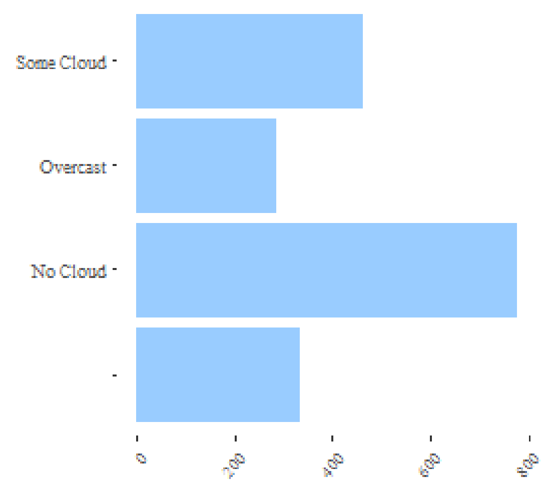
Data distribution of flight phase in 1990



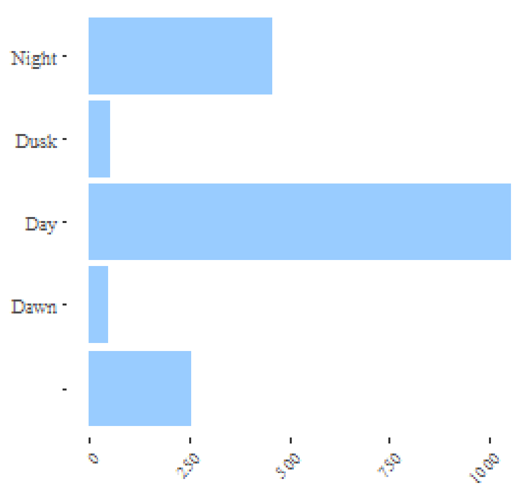
Data distribution of engine type in 1990



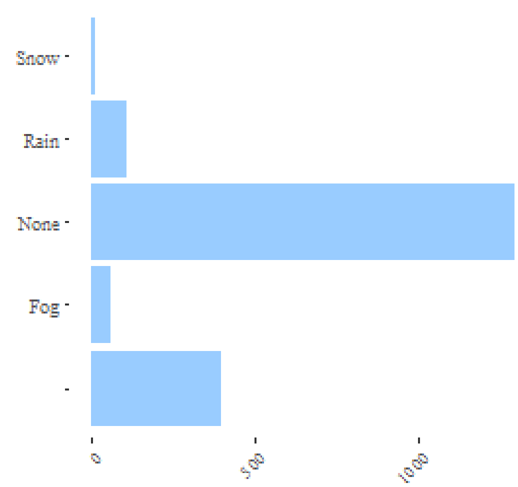
Data distribution of sky condition in 1990

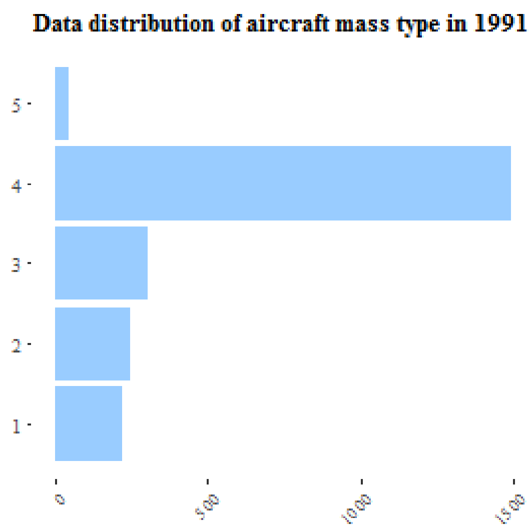


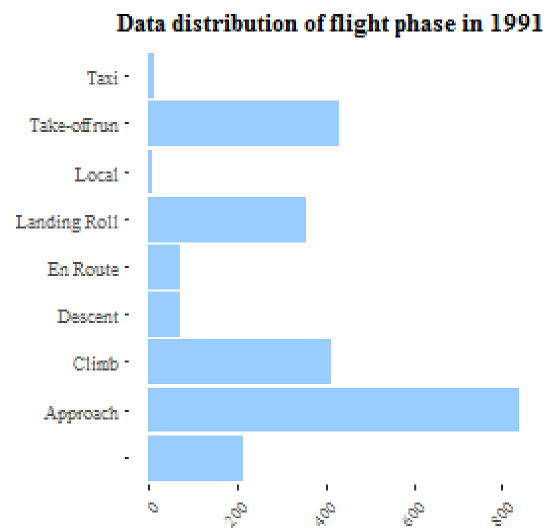
Data distribution of time of day in 1990

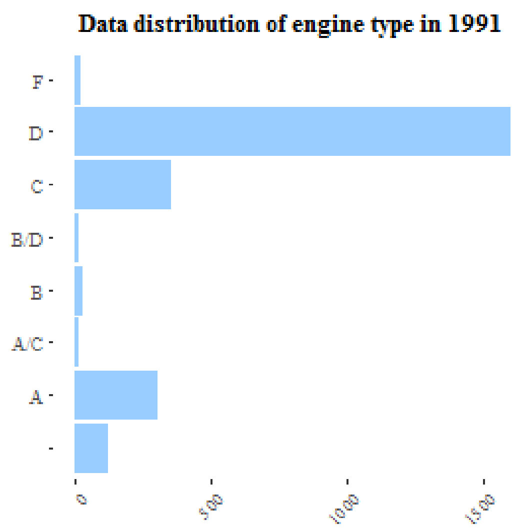


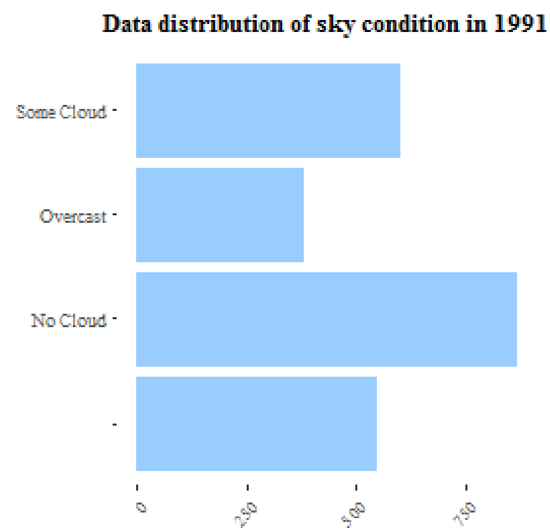
Data distribution of precipitation in 1990

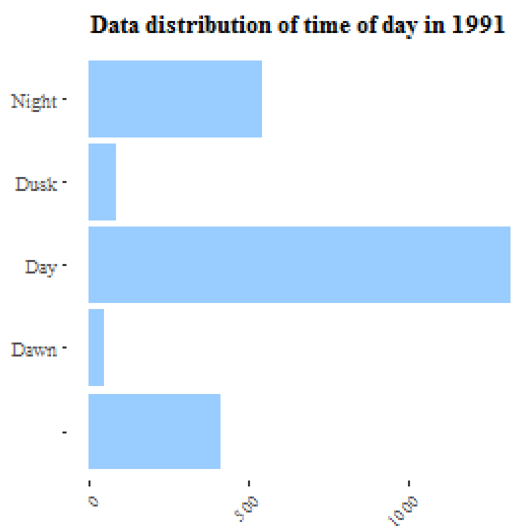


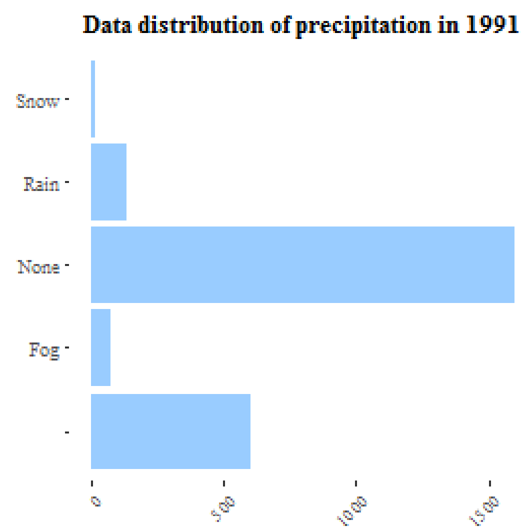

Data distribution of aircraft mass type in 1991
5
4
3
2
1
0
500
1000
1500

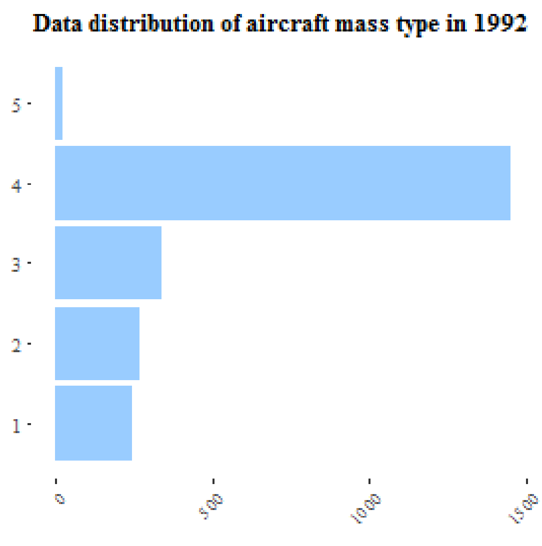

Data distribution of flight phase in 1991
Taxi
Take-off run
Local
Landing Roll
En Route
Descent
Climb
Approach
0
200
400
600
800

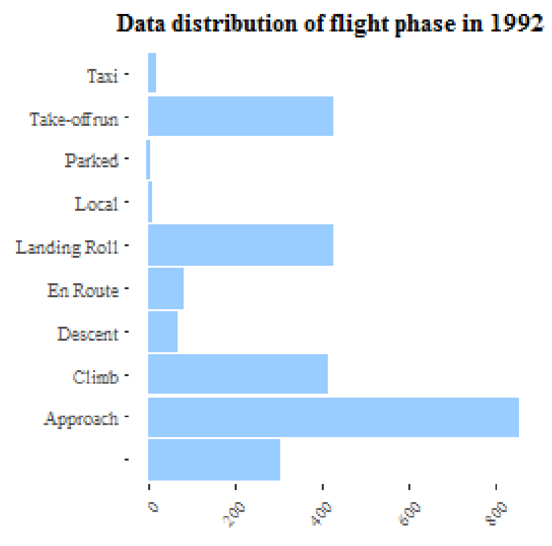

Data distribution of engine type in 1991
F
D
C
B/D
B
A/C
A
0
500
1000
1500

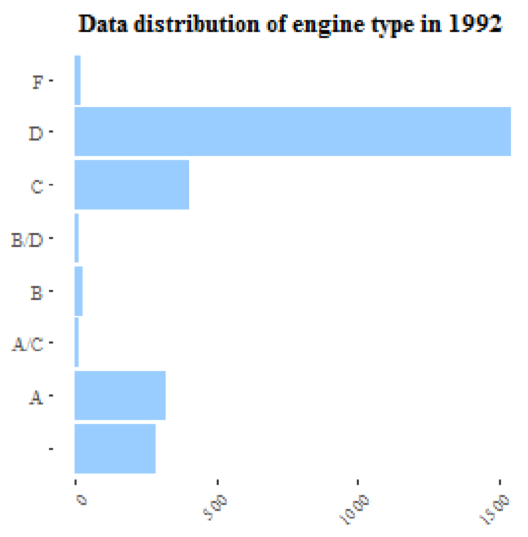

Data distribution of sky condition in 1991
Some Cloud
Overcast
No Cloud
0
250
500
750

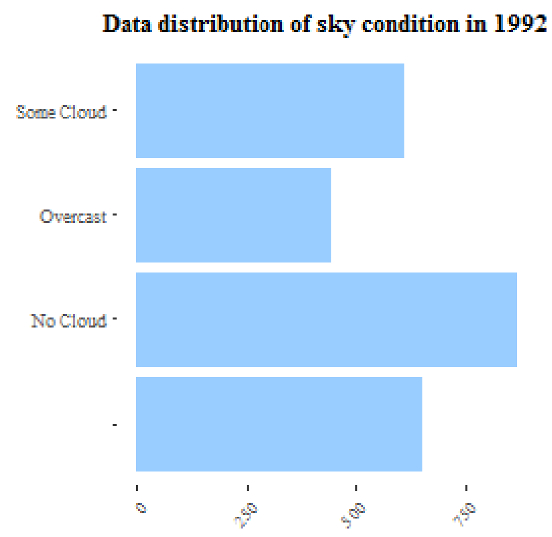

Data distribution of time of day in 1991
Night
Dusk
Day
Dawn
0
500
1000

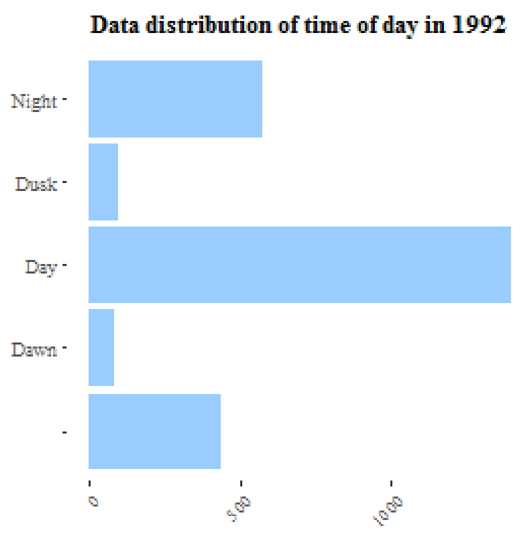

Data distribution of precipitation in 1991
Snow
Rain
None
Fog
0
500
1000
1500

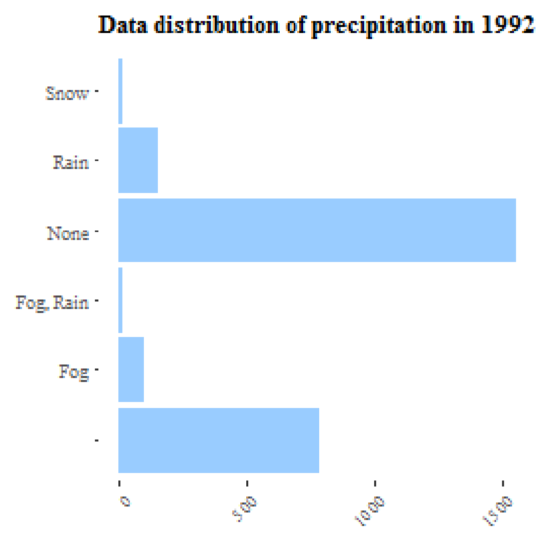

Data distribution of aircraft mass type in 1992
5
4
3
2
1
0
500
1000
1500


Data distribution of flight phase in 1992
Taxi
Take-off run
Parked
Local
Landing Roll
En Route
Descent
Climb
Approach
0
200
400
600
800

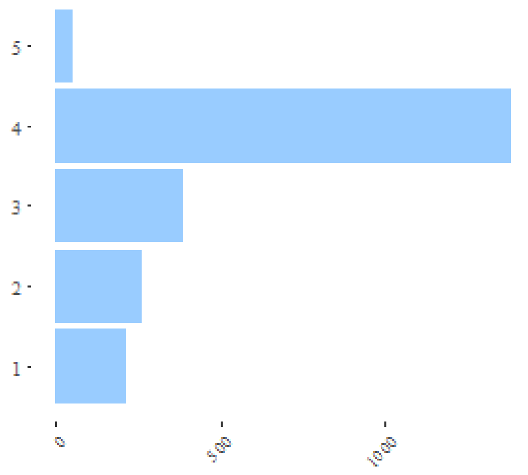

Data distribution of engine type in 1992
F
D
C
B/D
B
A/C
A
0
500
1000
1500


Data distribution of sky condition in 1992
Some Cloud
Overcast
No Cloud
0
250
500
750

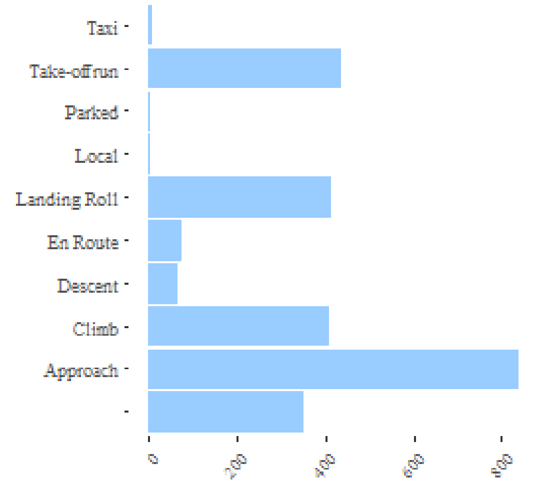

Data distribution of time of day in 1992
Night
Dusk
Day
Dawn
0
500
1000


Data distribution of precipitation in 1992
Snow
Rain
None
Fog, Rain
Fog
0
500
1000
1500

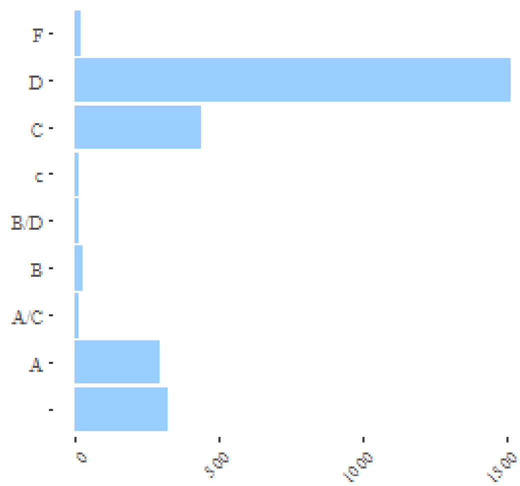
Data distribution of aircraft mass type in 1993



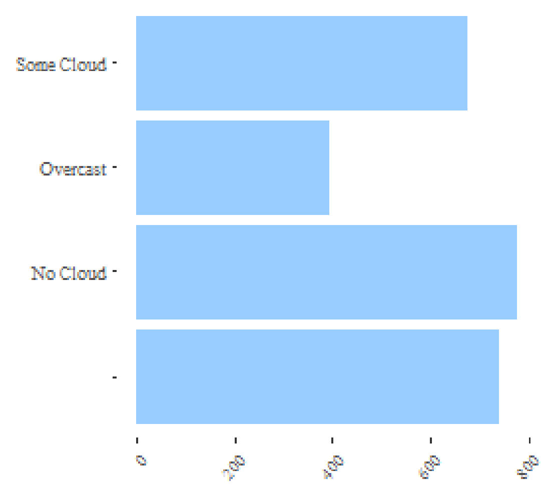
Data distribution of flight phase in 1993



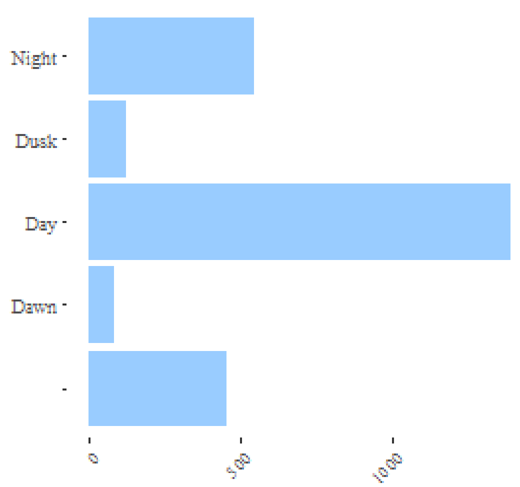
Data distribution of engine type in 1993



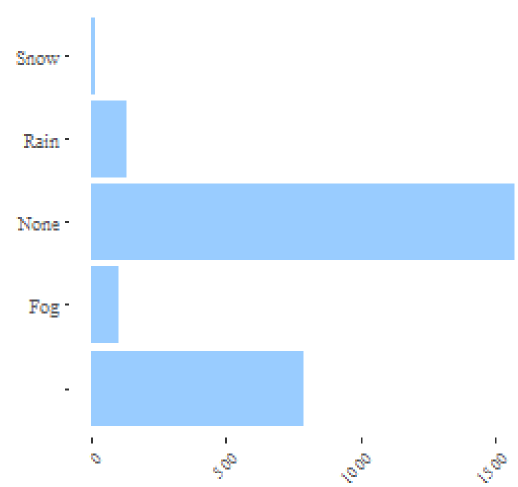
Data distribution of sky condition in 1993



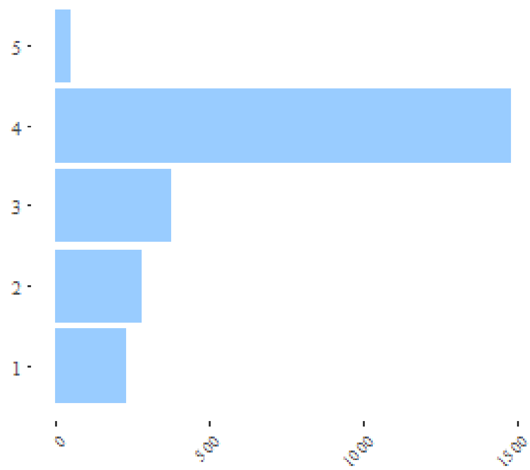
Data distribution of time of day in 1993



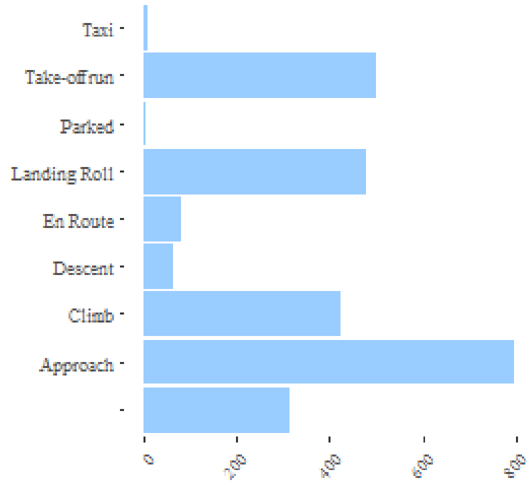
Data distribution of precipitation in 1993



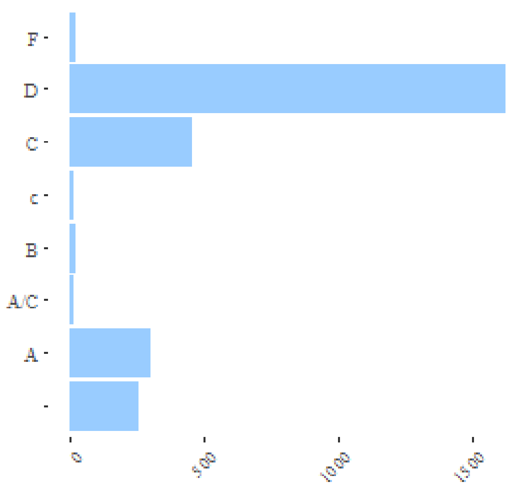
Data distribution of aircraft mass type in 1994



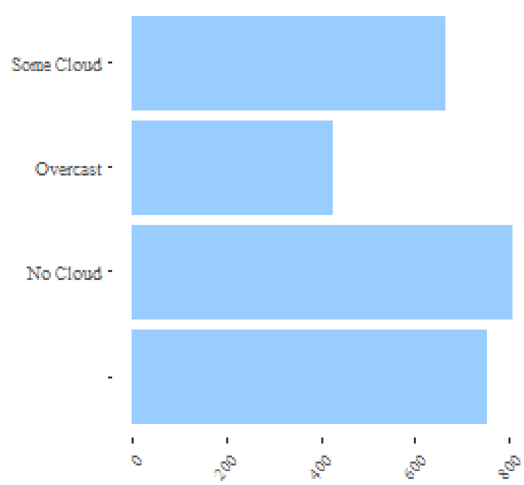
Data distribution of flight phase in 1994



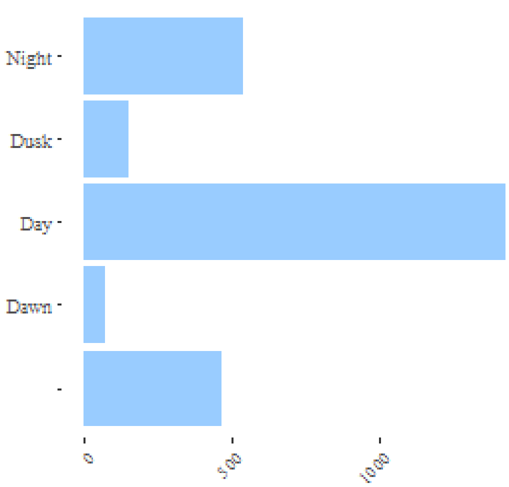
Data distribution of engine type in 1994



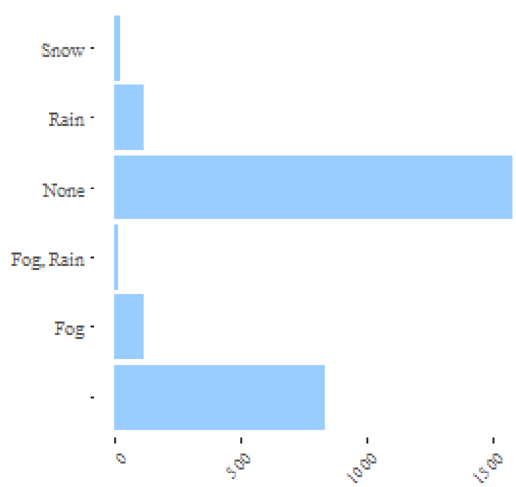
Data distribution of sky condition in 1994

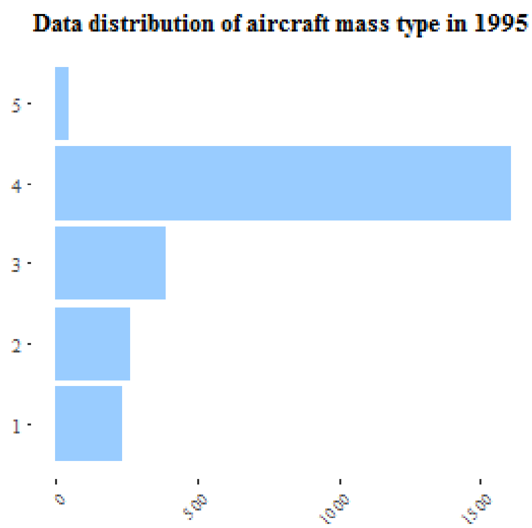


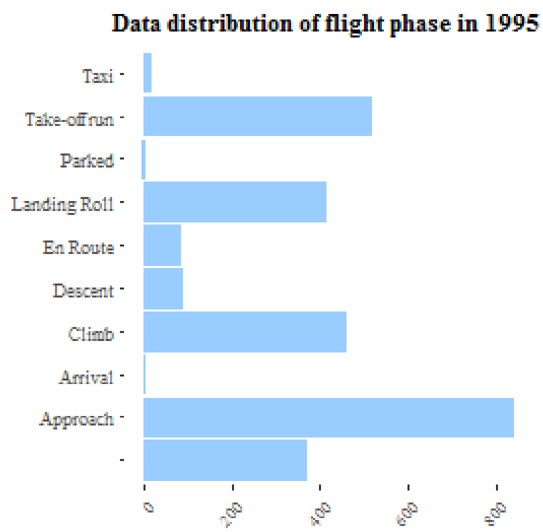
Data distribution of time of day in 1994

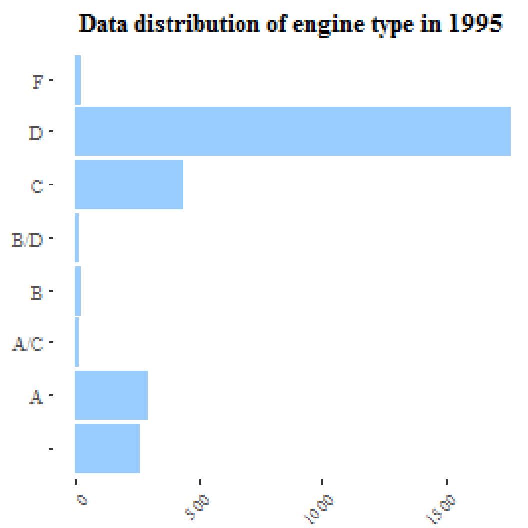


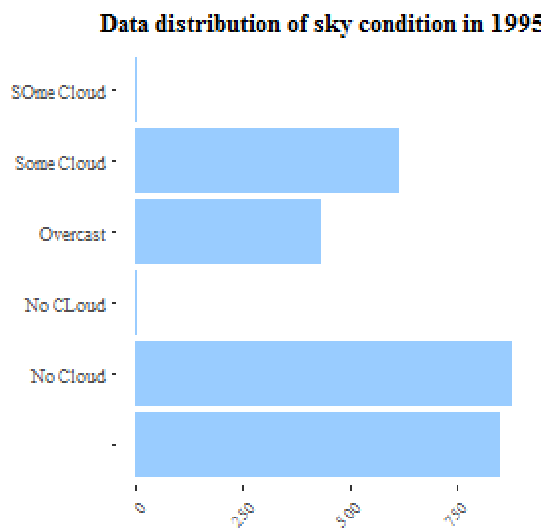
Data distribution of precipitation in 1994

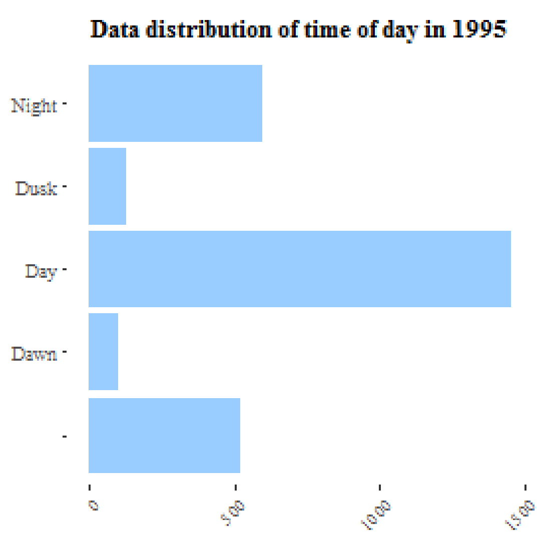


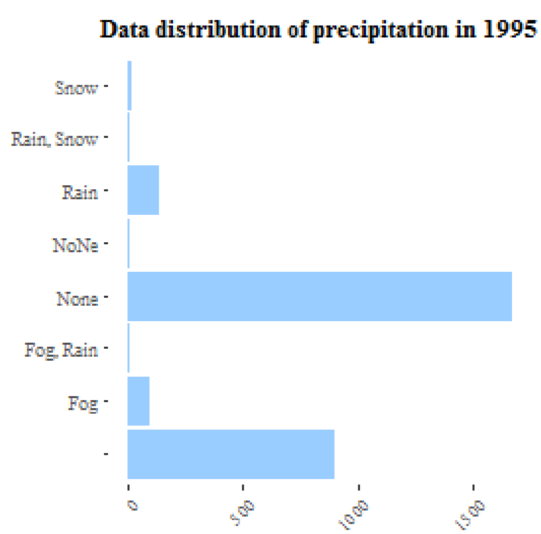

Data distribution of aircraft mass type in 1995
5
4
3
2
1
0
500
1000
1500

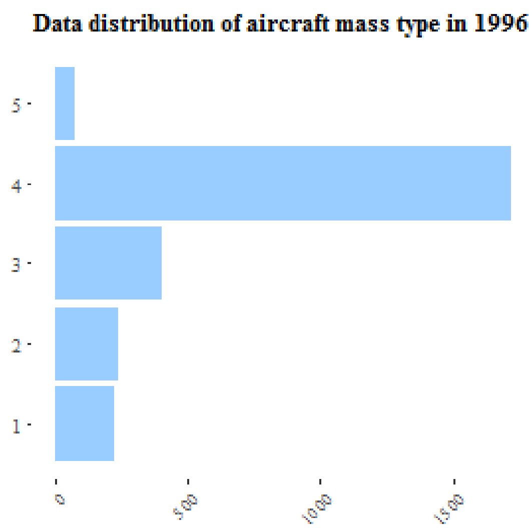

Data distribution of flight phase in 1995
Taxi
Take-off run
Parked
Landing Roll
En Route
Descent
Climb
Arrival
Approach
0
200
400
600
800

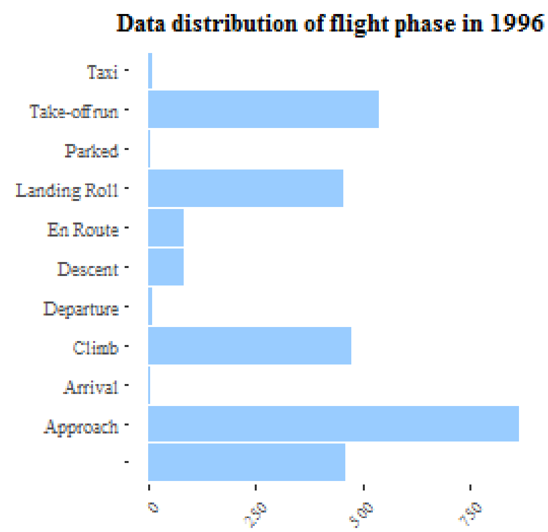

Data distribution of engine type in 1995
F
D
C
B/D
B
A/C
A
0
500
1000
1500

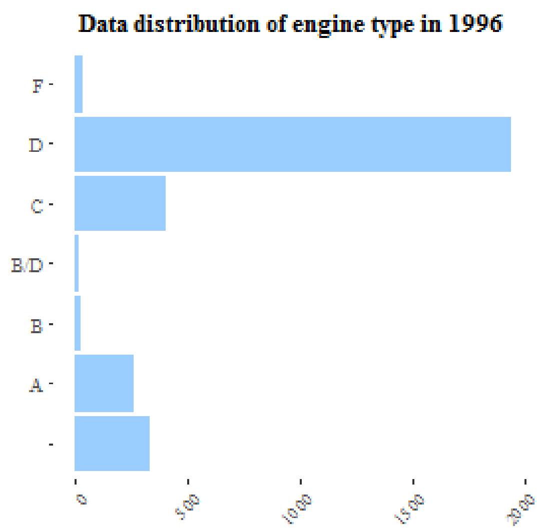

Data distribution of sky condition in 1995
SOme Cloud
Some Cloud
Overcast
No CLoud
No Cloud
0
250
500
750

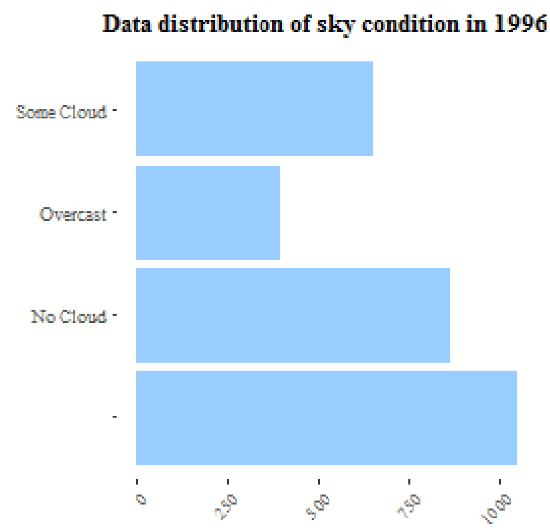

Data distribution of time of day in 1995
Night
Dusk
Day
Dawn
0
500
1000
1500

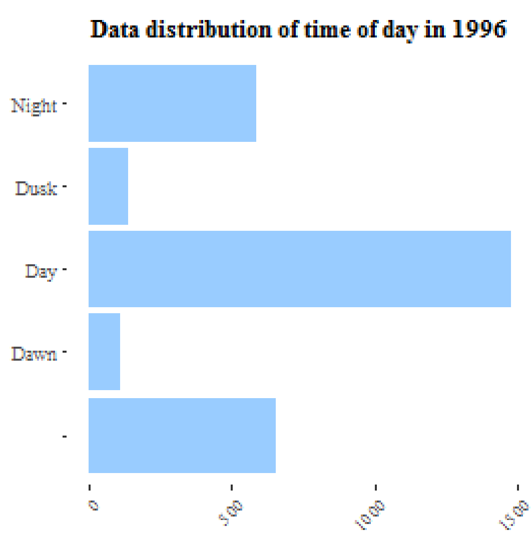

Data distribution of precipitation in 1995
Snow
Rain, Snow
Rain
NoNe
None
Fog, Rain
Fog
0
500
1000
1500

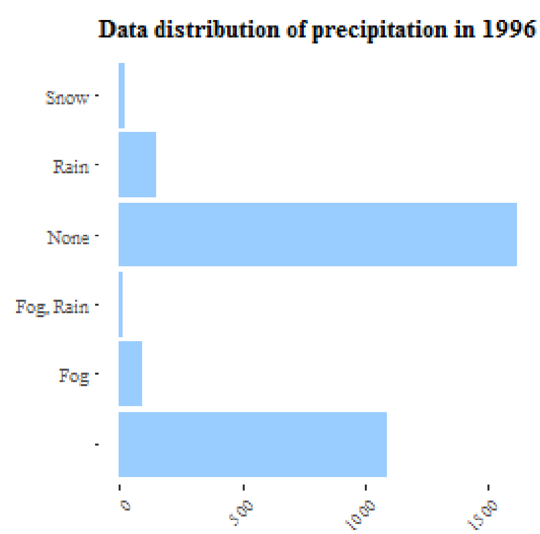

Data distribution of aircraft mass type in 1996
5
4
3
2
1
0
500
1000
1500


Data distribution of flight phase in 1996
Taxi
Take-off run
Parked
Landing Roll
En Route
Descent
Departure
Climb
Arrival
Approach
0
250
500
750

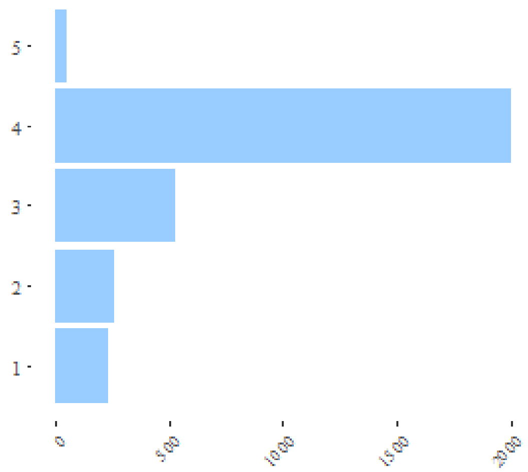

Data distribution of engine type in 1996
F
D
C
B/D
B
A
0
500
1000
1500
2000


Data distribution of sky condition in 1996
Some Cloud
Overcast
No Cloud
0
250
500
750
1000

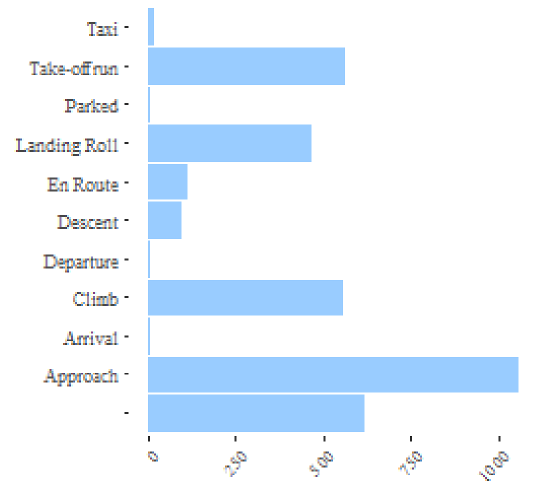

Data distribution of time of day in 1996
Night
Dusk
Day
Dawn
0
500
1000
1500


Data distribution of precipitation in 1996
Snow
Rain
None
Fog, Rain
Fog
0
500
1000
1500

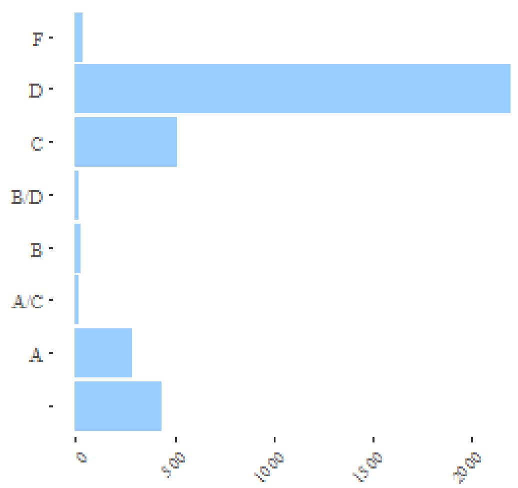
Data distribution of aircraft mass type in 1997



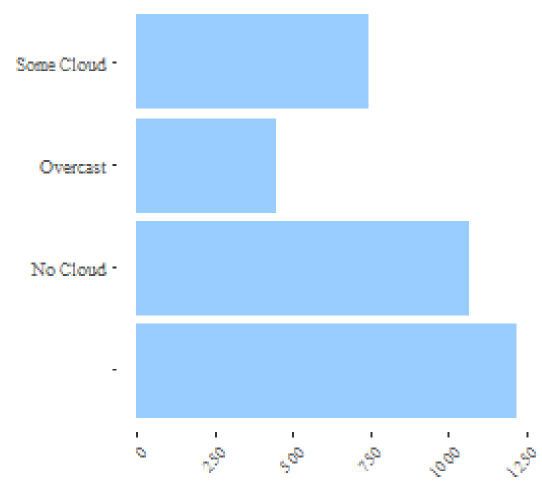
Data distribution of flight phase in 1997



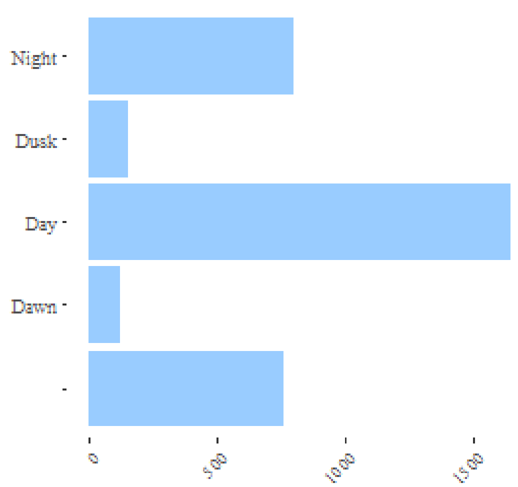
Data distribution of engine type in 1997



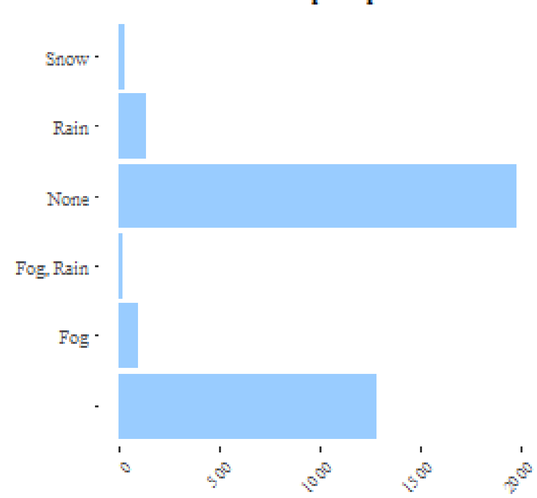
Data distribution of sky condition in 1997



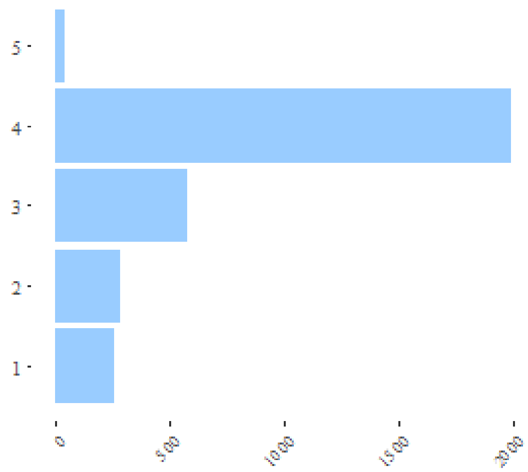
Data distribution of time of day in 1997



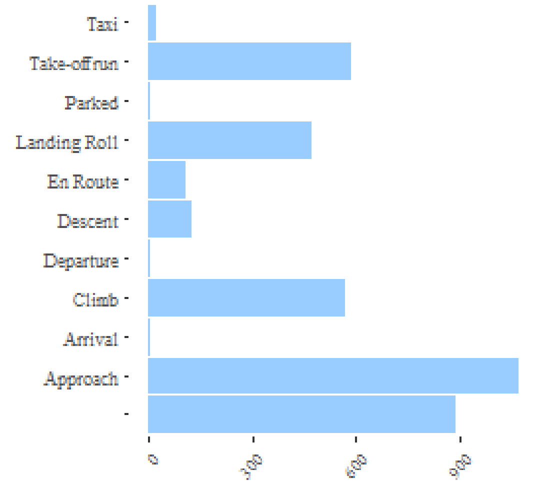
Data distribution of precipitation in 1997



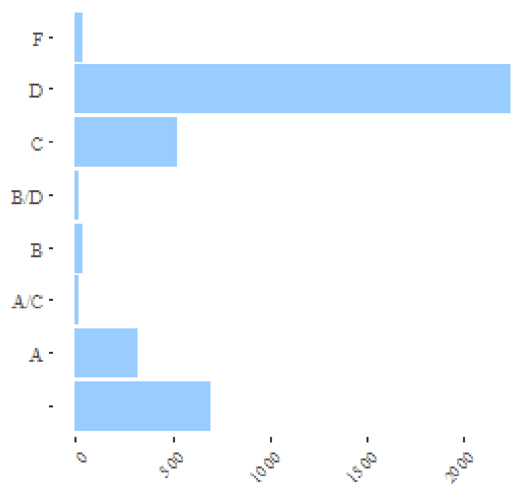
Data distribution of aircraft mass type in 1998



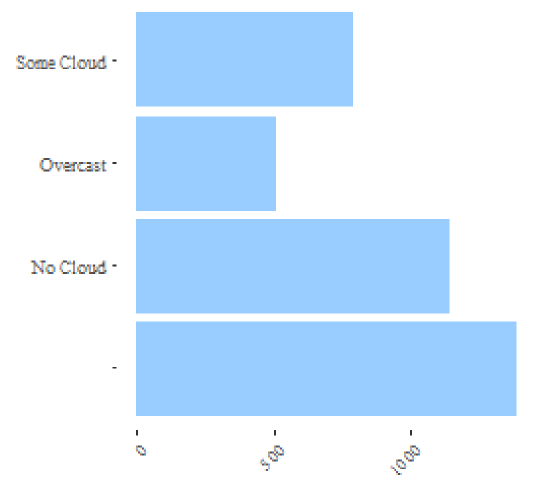
Data distribution of flight phase in 1998



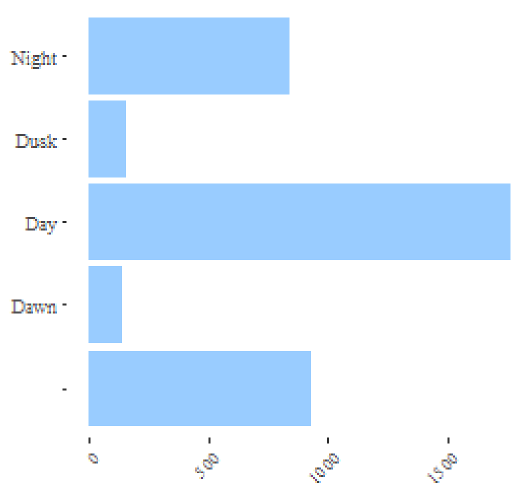
Data distribution of engine type in 1998



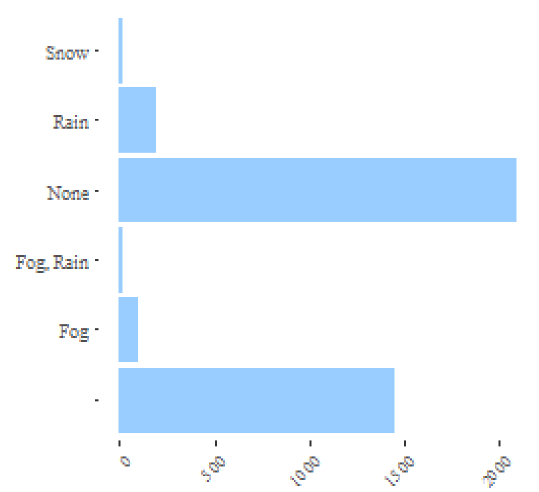
Data distribution of sky condition in 1998

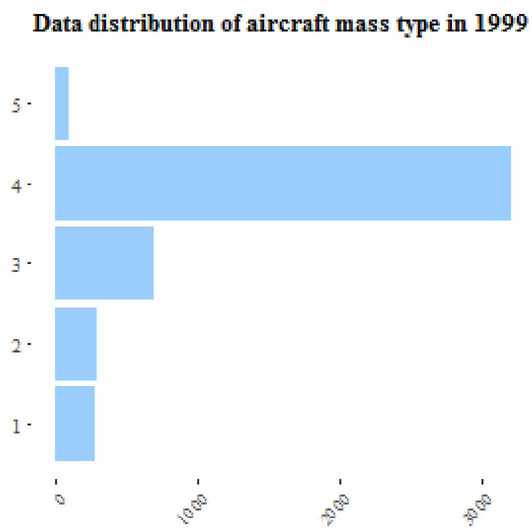


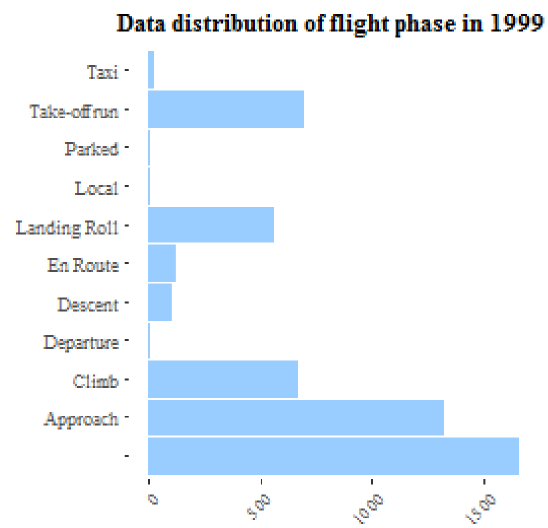
Data distribution of time of day in 1998

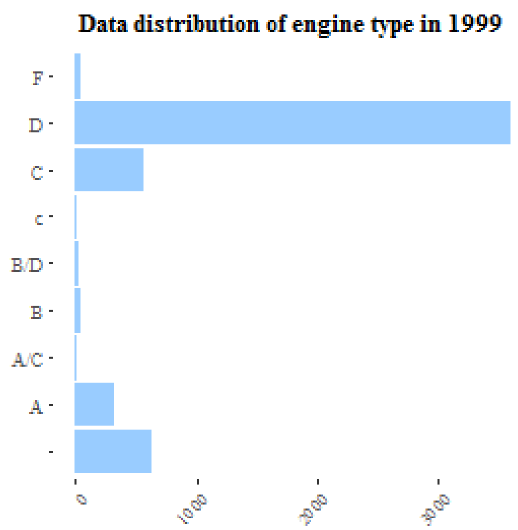


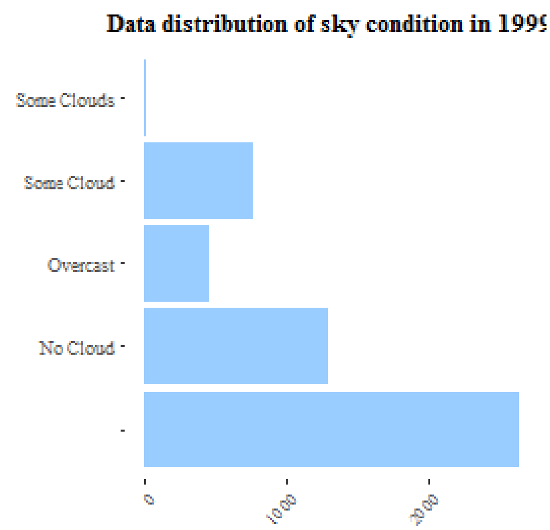
Data distribution of precipitation in 1998

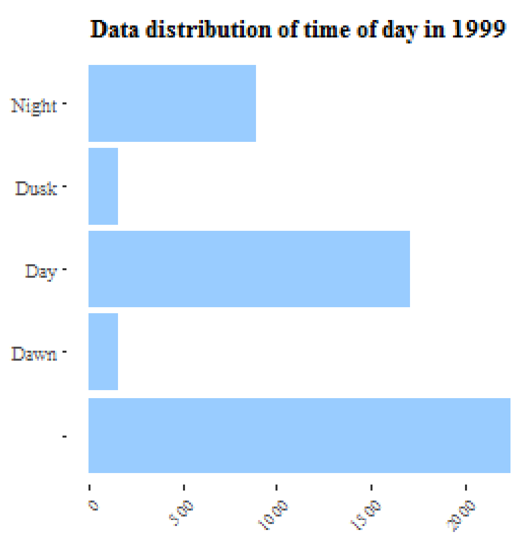


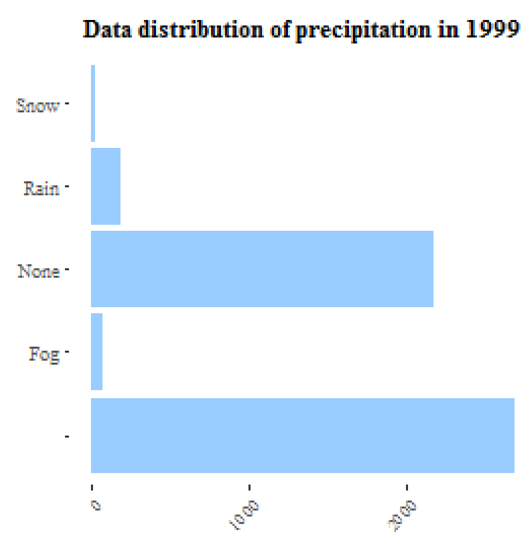

Data distribution of aircraft mass type in 1999
5
4
3
2
1
0
1000
2000
3000


Data distribution of flight phase in 1999
Taxi
Take-off run
Parked
Local
Landing Roll
En Route
Descent
Departure
Climb
Approach
0
500
1000
1500

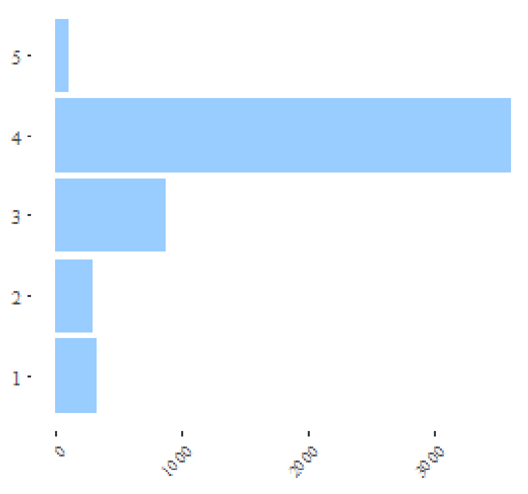

Data distribution of engine type in 1999
F
D
C
c
B/D
B
A/C
A
0
1000
2000
3000


Data distribution of sky condition in 199
Some Clouds
Some Cloud
Overcast
No Cloud
0
1000
2000

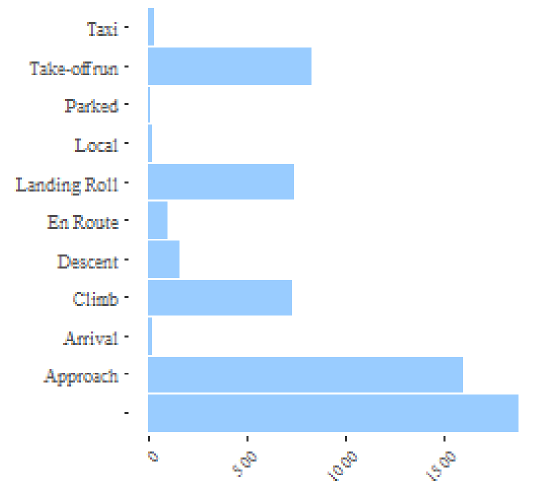

Data distribution of time of day in 1999
Night
Dusk
Day
Dawn
0
500
1000
1500
2000


Data distribution of precipitation in 1999
Snow
Rain
None
Fog
0
1000
2000

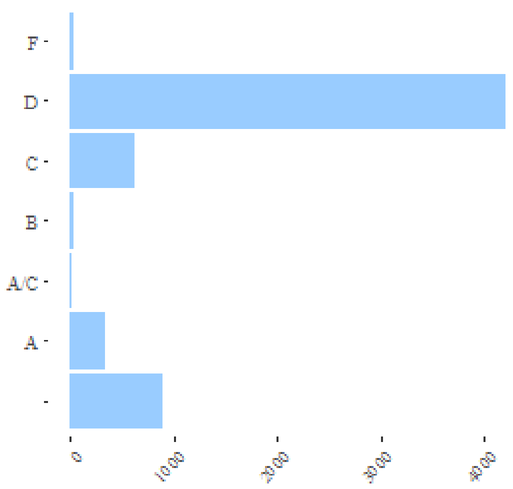
Data distribution of aircraft mass type in 2000



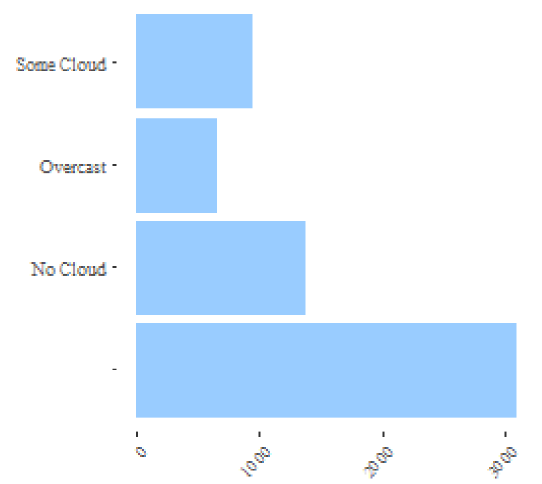
Data distribution of flight phase in 2000



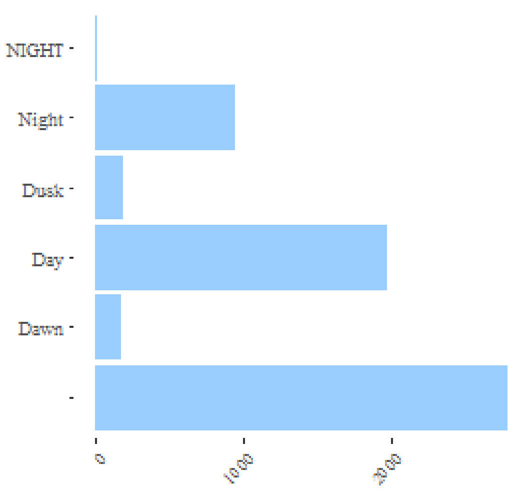
Data distribution of engine type in 2000



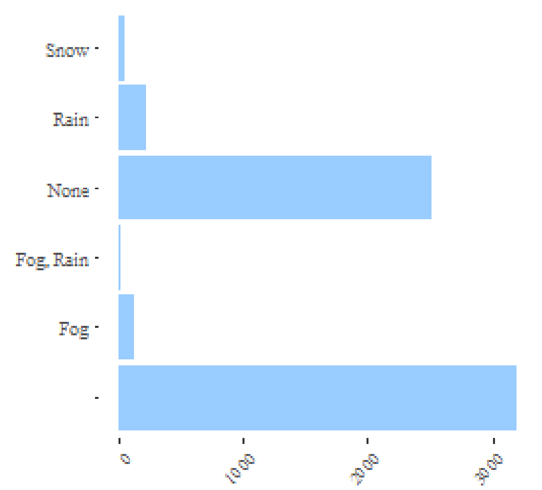
Data distribution of sky condition in 2000



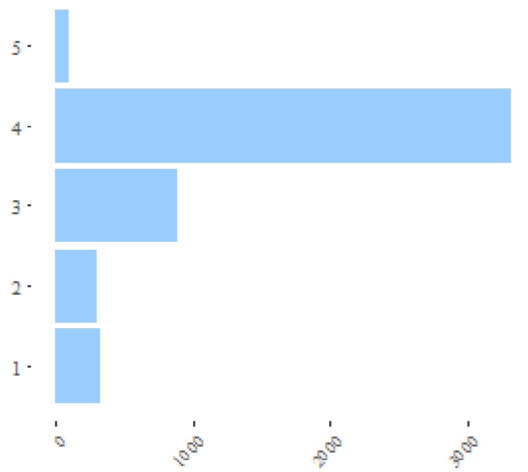
Data distribution of time of day in 2000



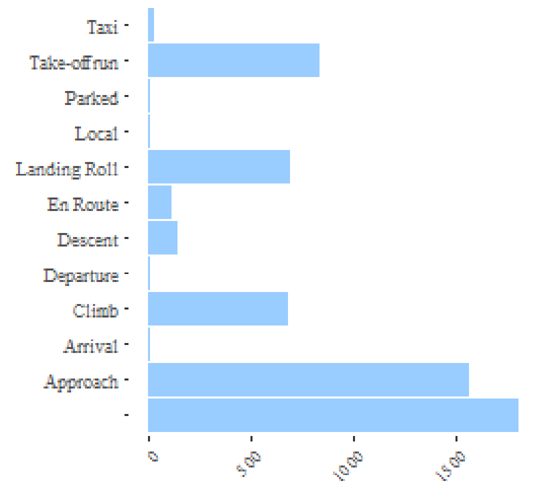
Data distribution of precipitation in 2000



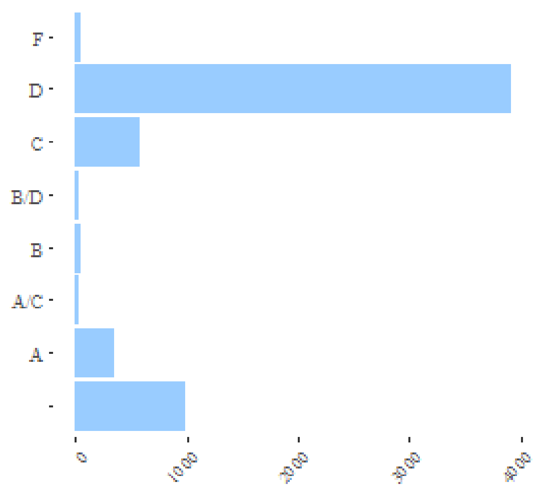
Data distribution of aircraft mass type in 2001



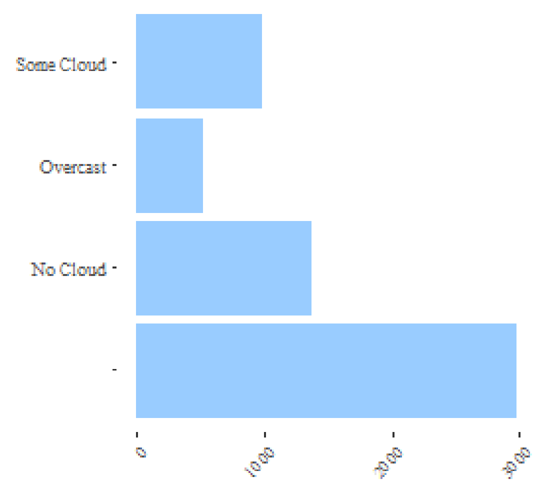
Data distribution of flight phase in 2001



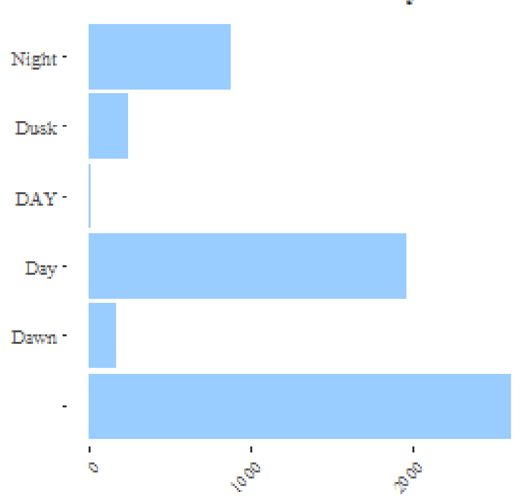
Data distribution of engine type in 2001



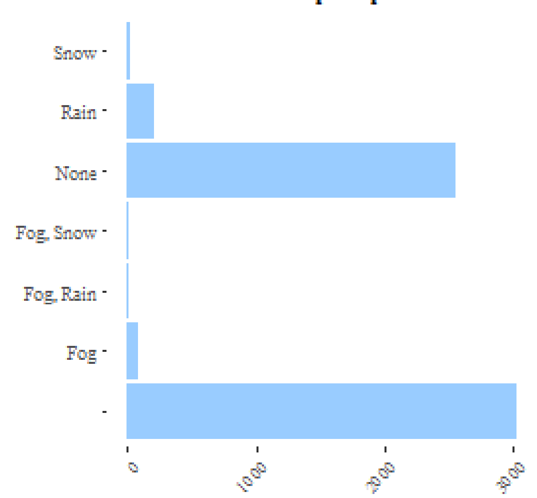
Data distribution of sky condition in 2001



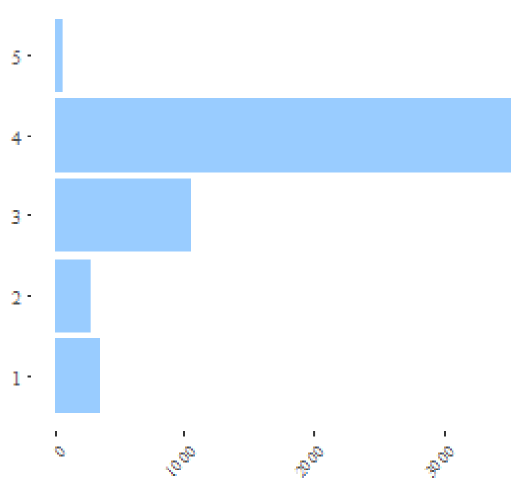
Data distribution of time of day in 2001



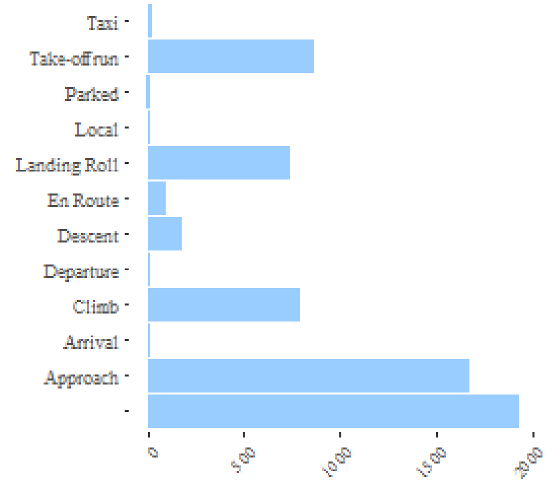
Data distribution of precipitation in 2001



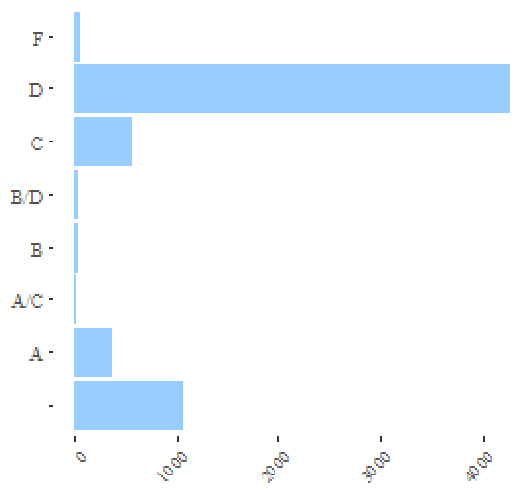
Data distribution of aircraft mass type in 2002



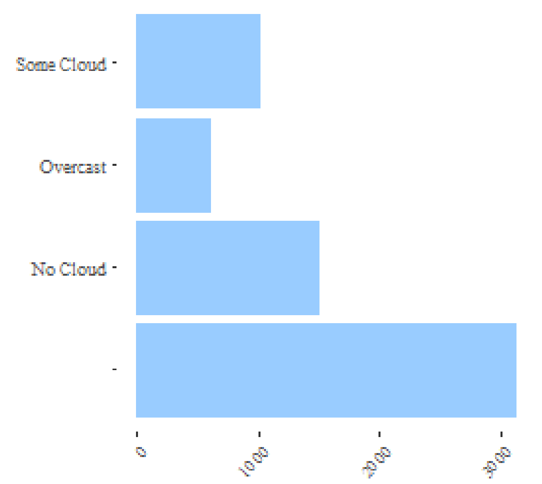
Data distribution of flight phase in 2002



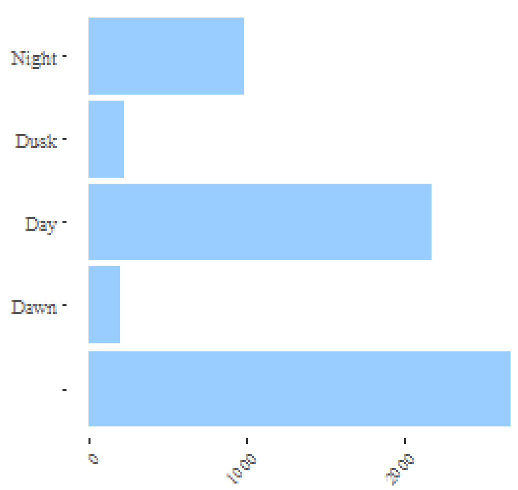
Data distribution of engine type in 2002



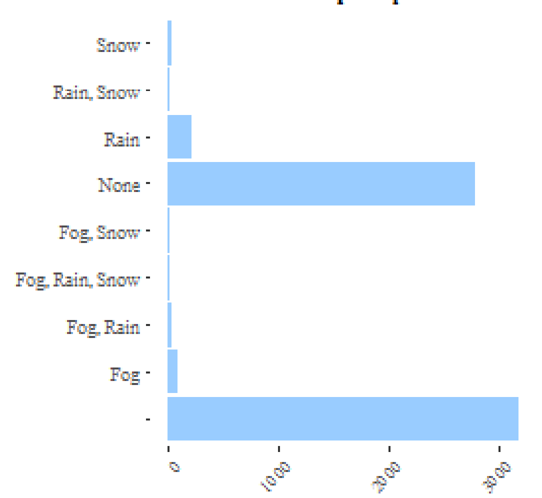
Data distribution of sky condition in 2002

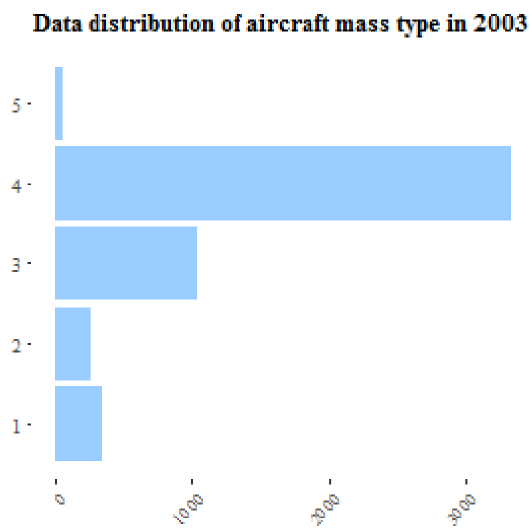


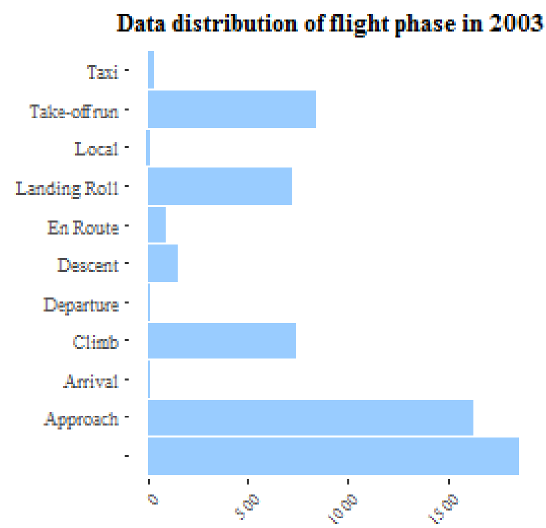
Data distribution of time of day in 2002

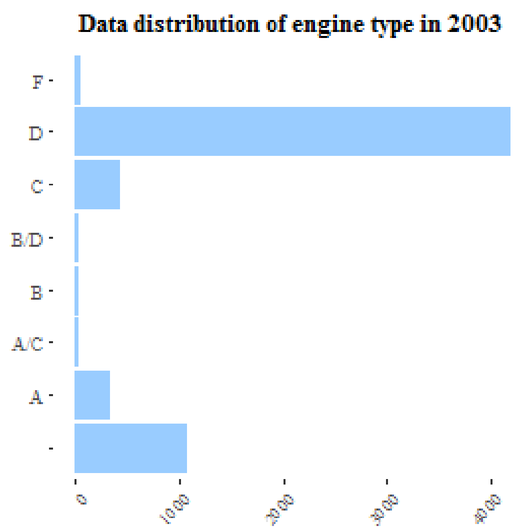


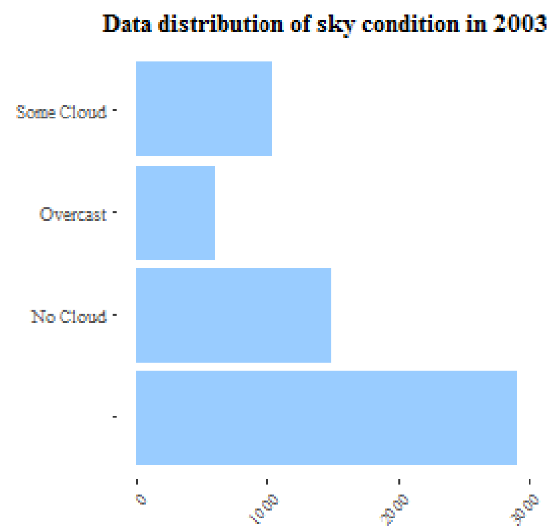
Data distribution of precipitation in 200

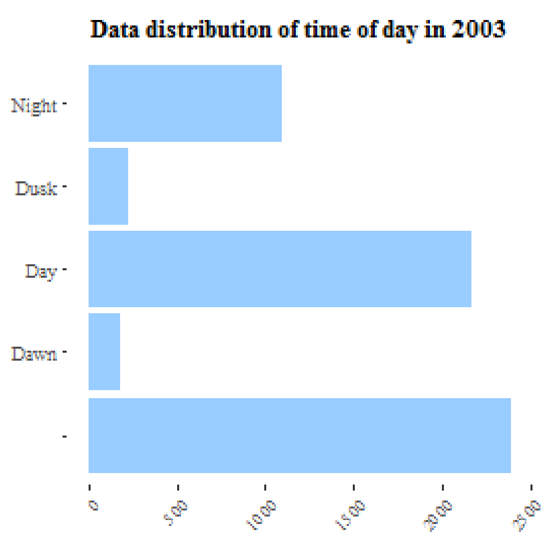


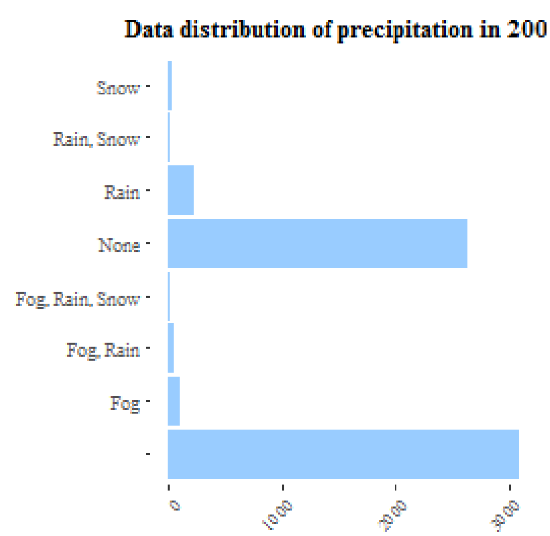

Data distribution of aircraft mass type in 2003
5
4
3
2
1
0
1000
2000
3000


Data distribution of flight phase in 2003
Taxi
Take-off run
Local
Landing Roll
En Route
Descent
Departure
Climb
Arrival
Approach
0
500
1000
1500

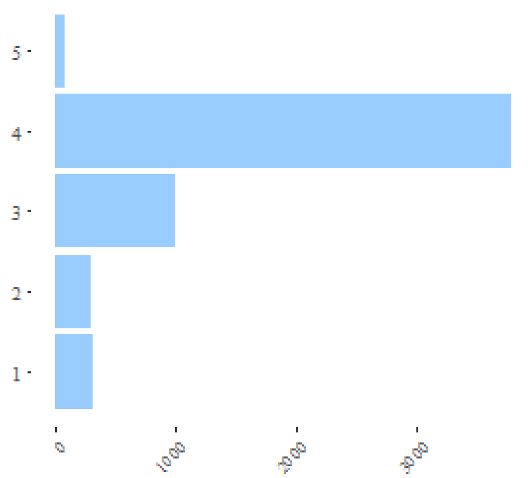

Data distribution of engine type in 2003
F
D
C
B/D
B
A/C
A
0
1000
2000
3000
4000


Data distribution of sky condition in 2003
Some Cloud
Overcast
No Cloud
0
1000
2000
3000

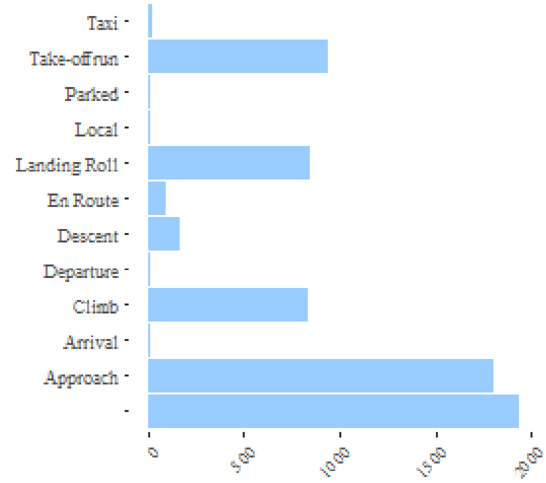

Data distribution of time of day in 2003
Night
Dusk
Day
Dawn
0
500
1000
1500
2000
2500


Data distribution of precipitation in 200
Snow
Rain, Snow
Rain
None
Fog, Rain, Snow
Fog, Rain
Fog
0
1000
2000
3000

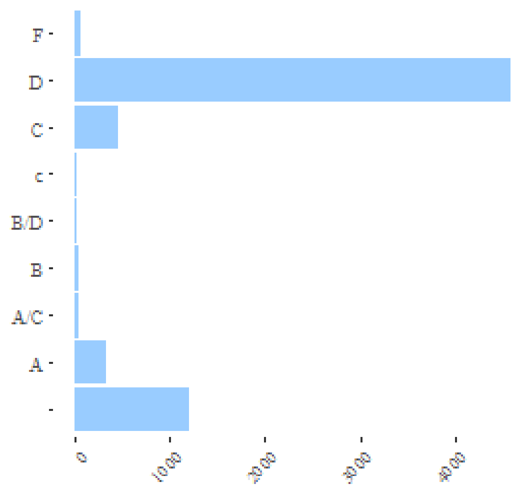
Data distribution of aircraft mass type in 2004



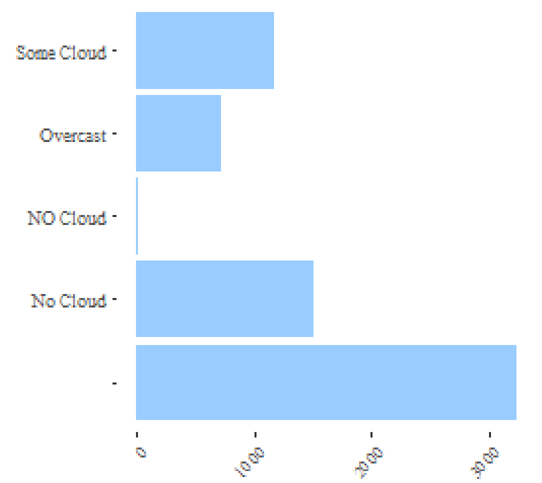
Data distribution of flight phase in 2004



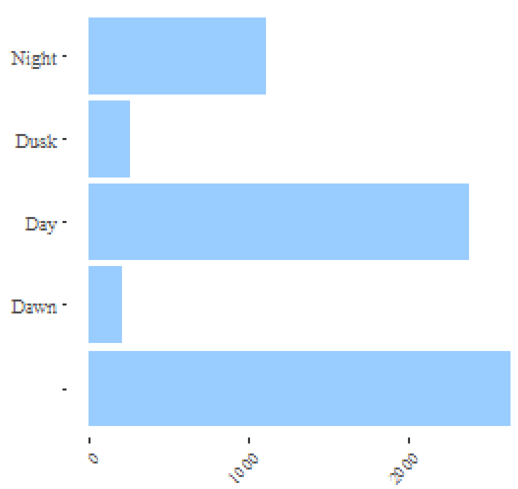
Data distribution of engine type in 2004



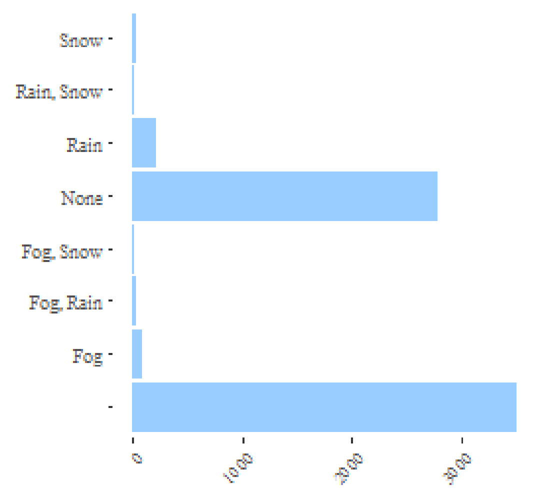
Data distribution of sky condition in 2004

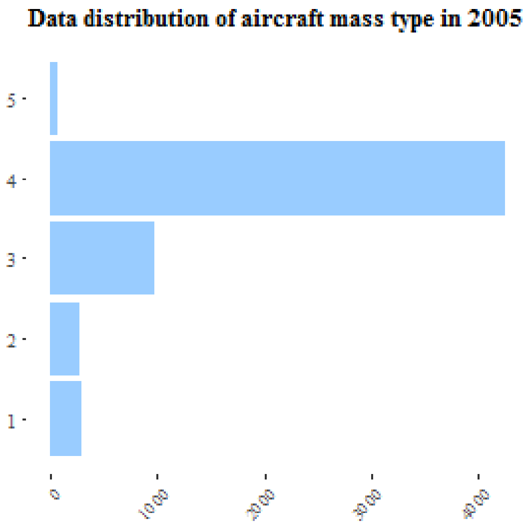


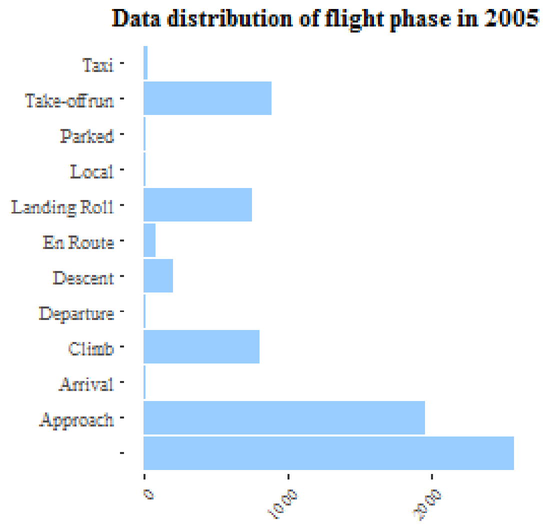
Data distribution of time of day in 2004

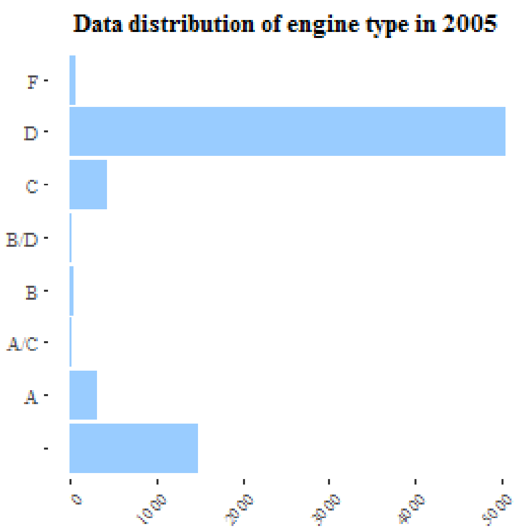


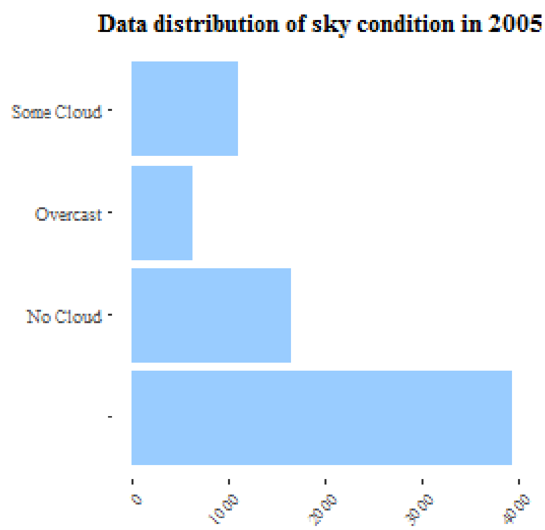
Data distribution of precipitation in 2004

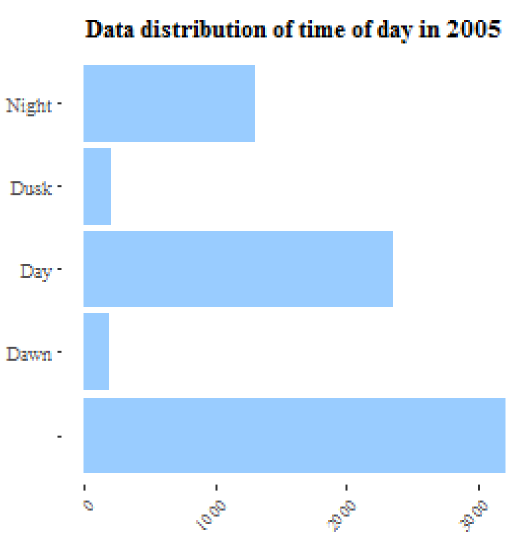


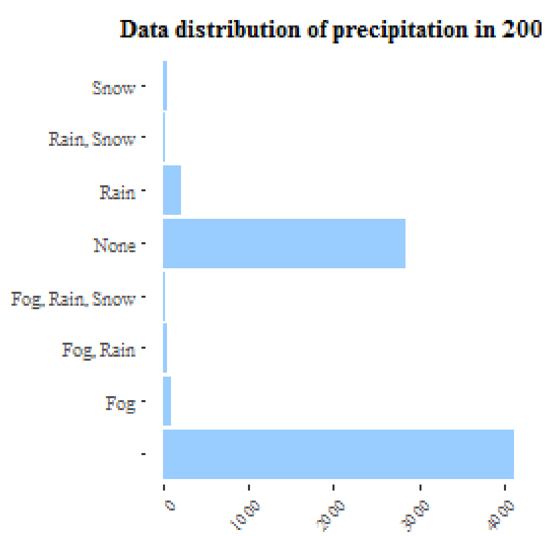

Data distribution of aircraft mass type in 2005
5
4
3
2
1
0
1000
2000
3000
4000


Data distribution of flight phase in 2005
Taxi
Take-off run
Parked
Local
Landing Roll
En Route
Descent
Departure
Climb
Arrival
Approach
0
1000
2000

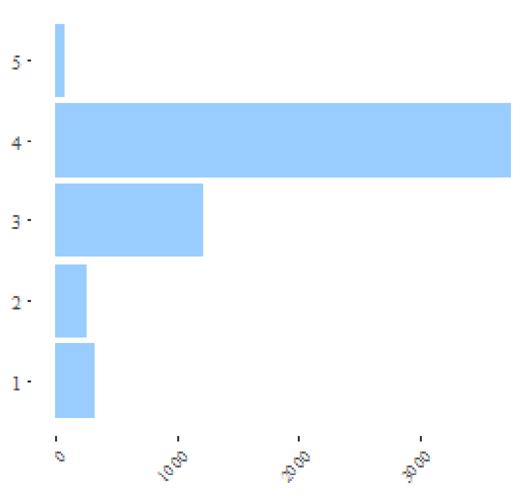

Data distribution of engine type in 2005
F
D
C
B/D
B
A/C
A
0
1000
2000
3000
4000
5000


Data distribution of sky condition in 2005
Some Cloud
Overcast
No Cloud
0
1000
2000
3000
4000

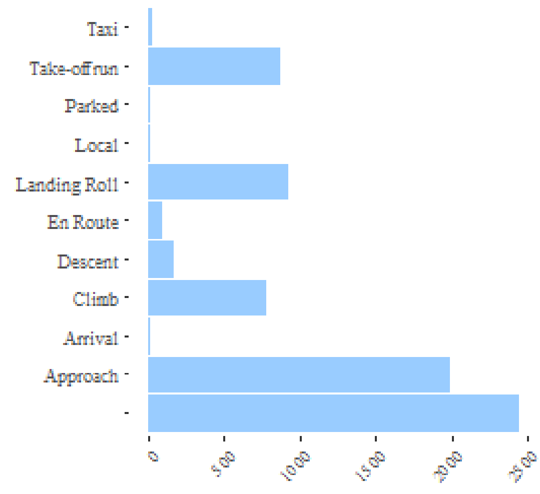

Data distribution of time of day in 2005
Night
Dusk
Day
Dawn
0
1000
2000
3000


Data distribution of precipitation in 200
Snow
Rain, Snow
Rain
None
Fog, Rain, Snow
Fog, Rain
Fog
0
1000
2000
3000
4000

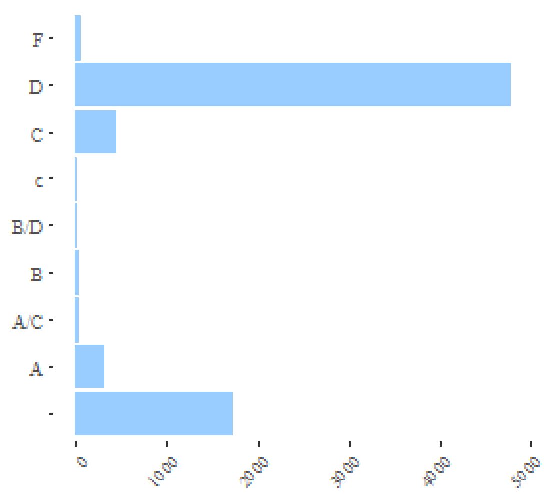
Data distribution of aircraft mass type in 2006



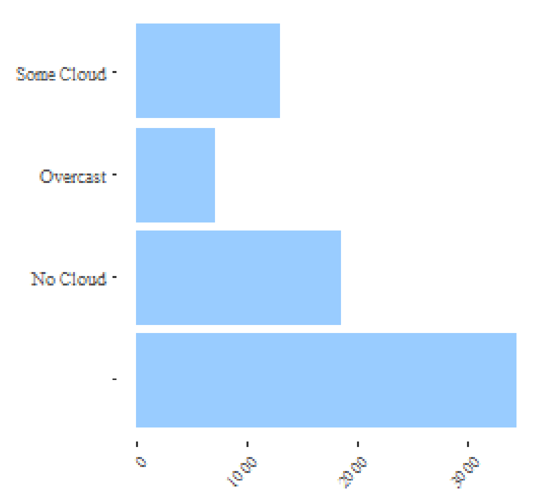
Data distribution of flight phase in 2006



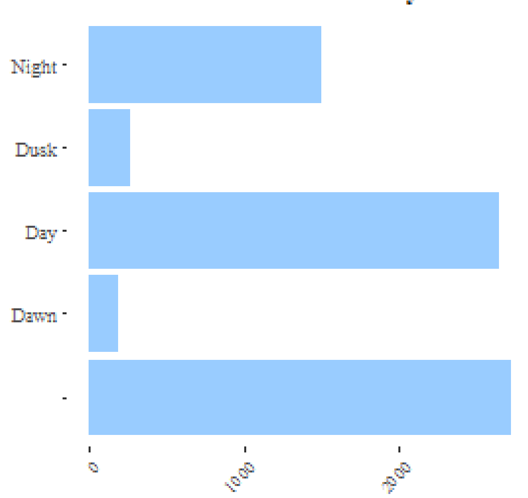
Data distribution of engine type in 2006



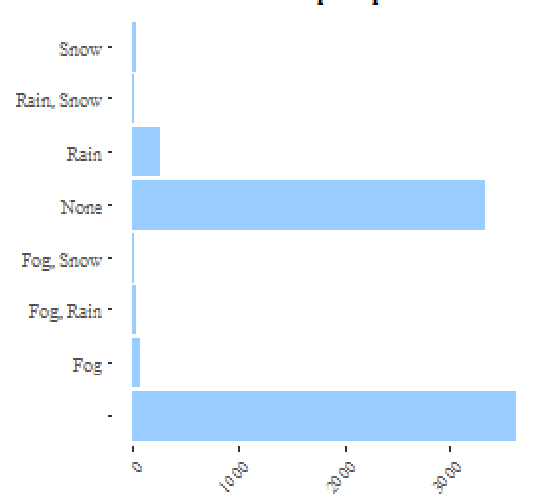
Data distribution of sky condition in 2006



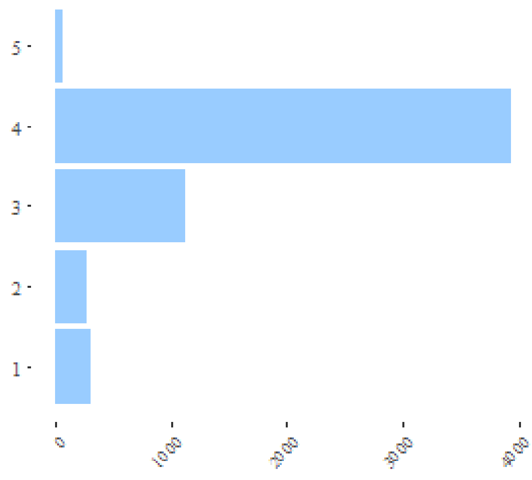
Data distribution of time of day in 2006



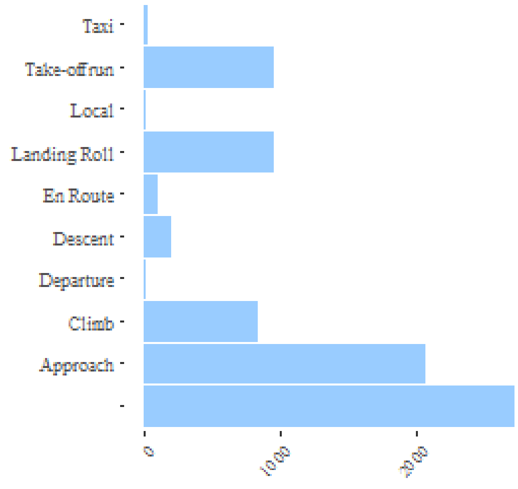
Data distribution of precipitation in 2006



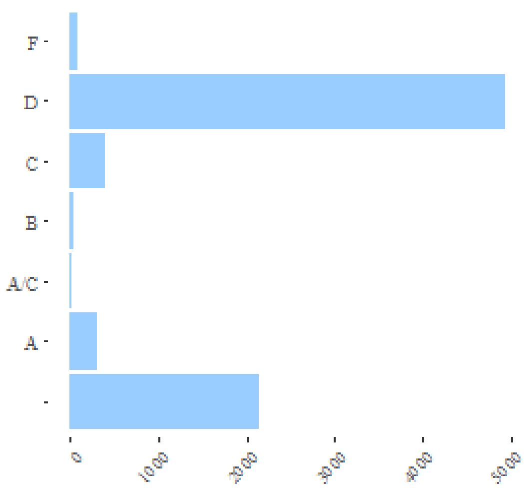
Data distribution of aircraft mass type in 2007



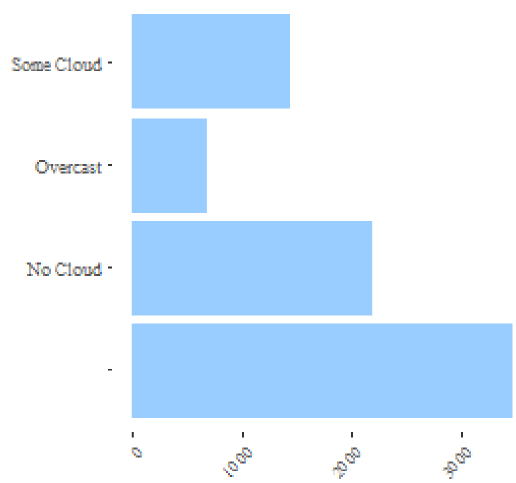
Data distribution of flight phase in 2007



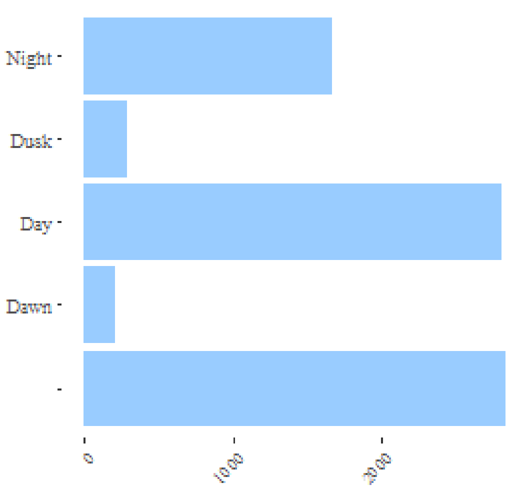
Data distribution of engine type in 2007



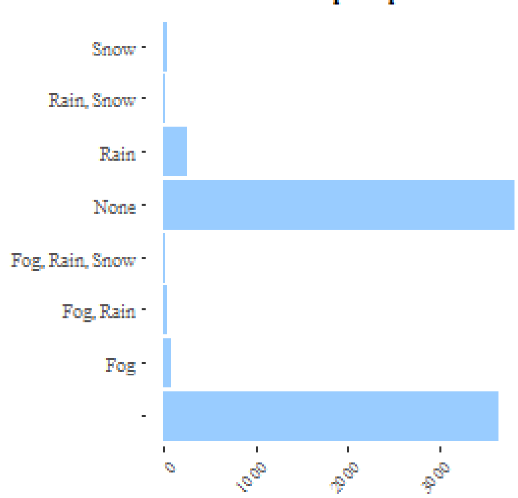
Data distribution of sky condition in 2007



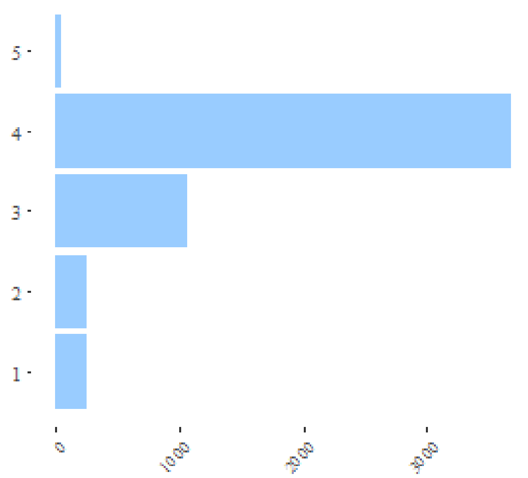
Data distribution of time of day in 2007



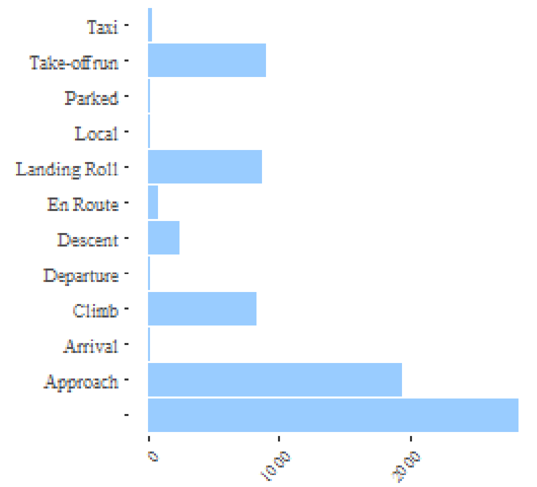
Data distribution of precipitation in 200



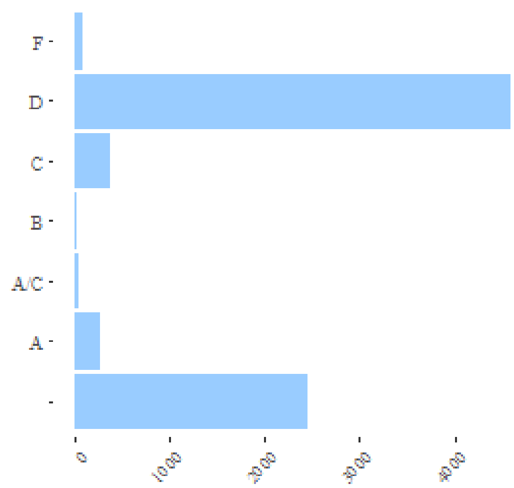
Data distribution of aircraft mass type in 2008



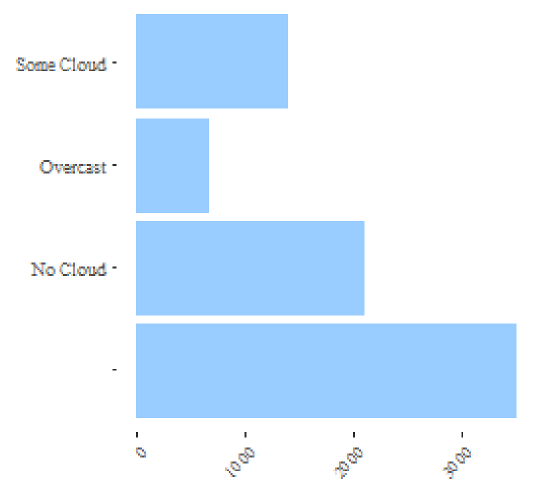
Data distribution of flight phase in 2008



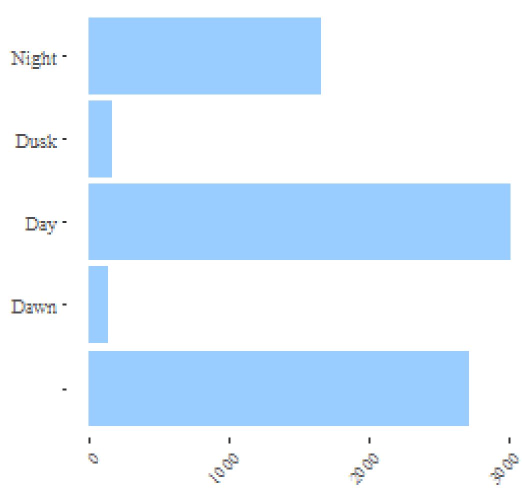
Data distribution of engine type in 2008



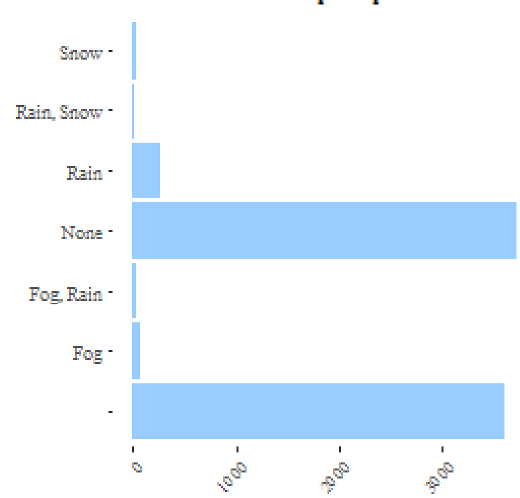
Data distribution of sky condition in 2008



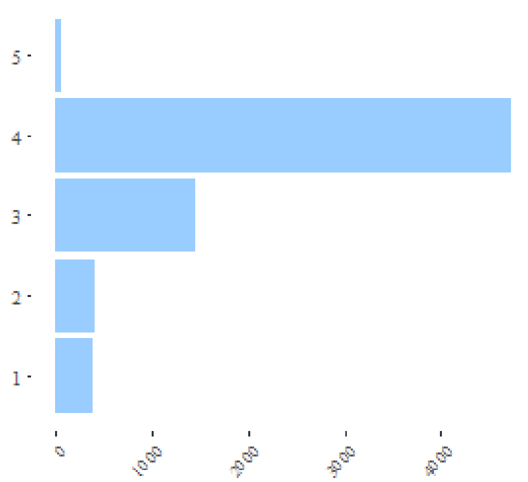
Data distribution of time of day in 2008



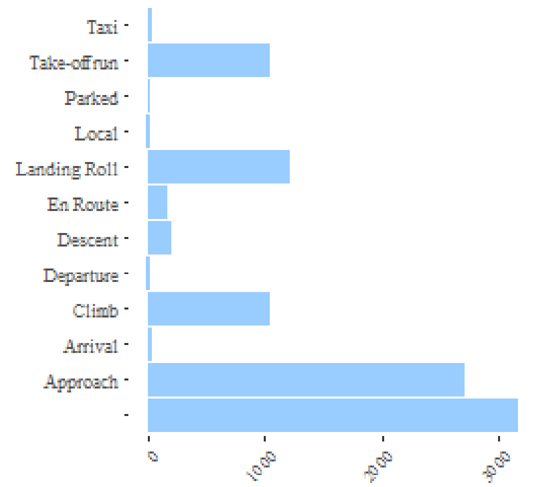
Data distribution of precipitation in 2008



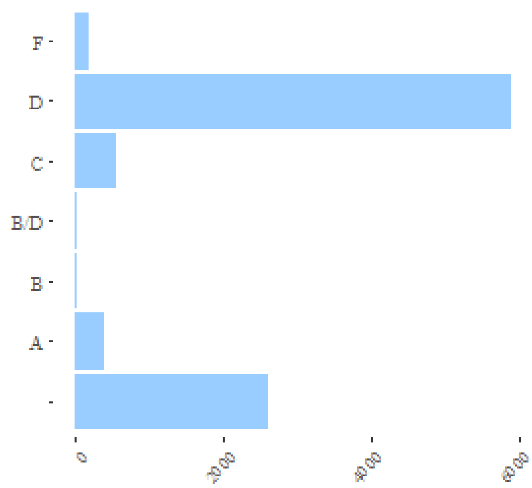
Data distribution of aircraft mass type in 2009



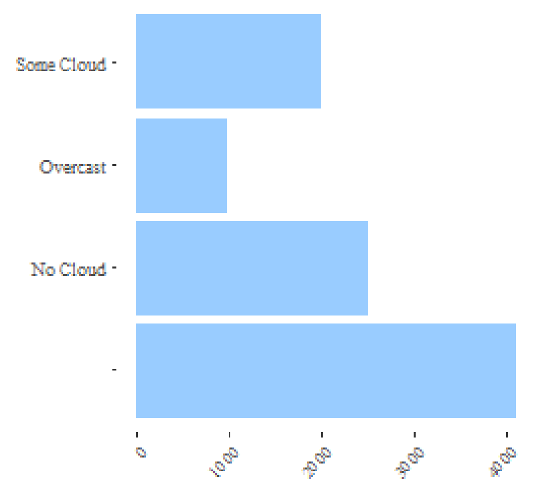
Data distribution of flight phase in 2009



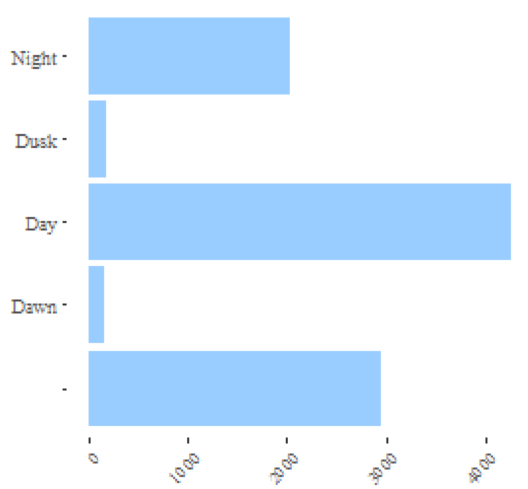
Data distribution of engine type in 2009



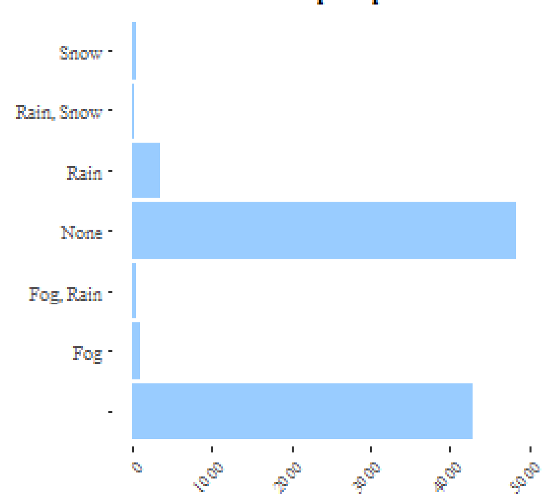
Data distribution of sky condition in 2009



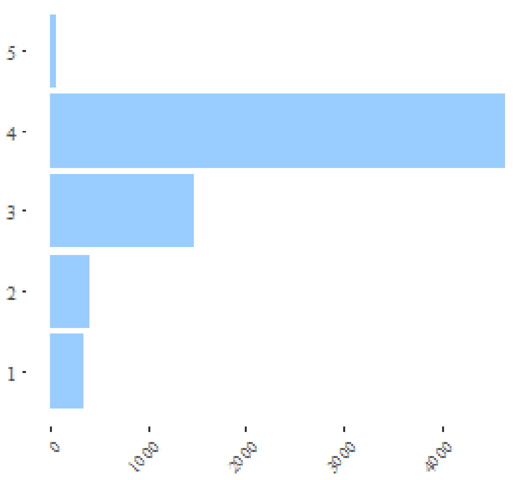
Data distribution of time of day in 2009



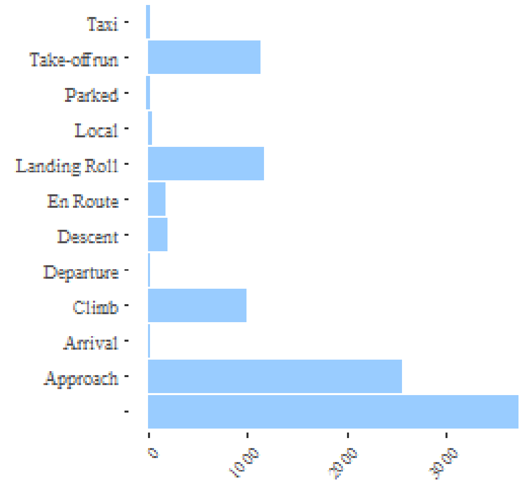
Data distribution of precipitation in 2009



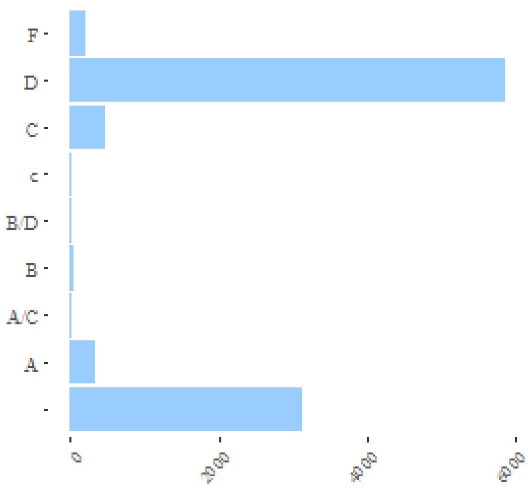
Data distribution of aircraft mass type in 2010



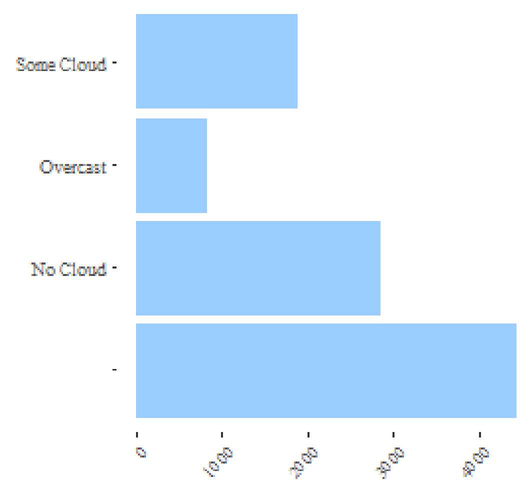
Data distribution of flight phase in 2010



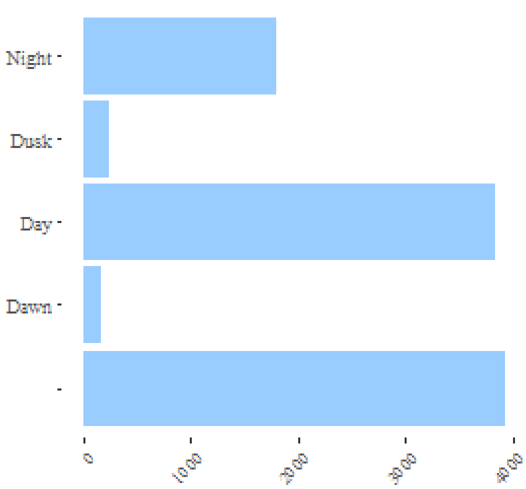
Data distribution of engine type in 2010



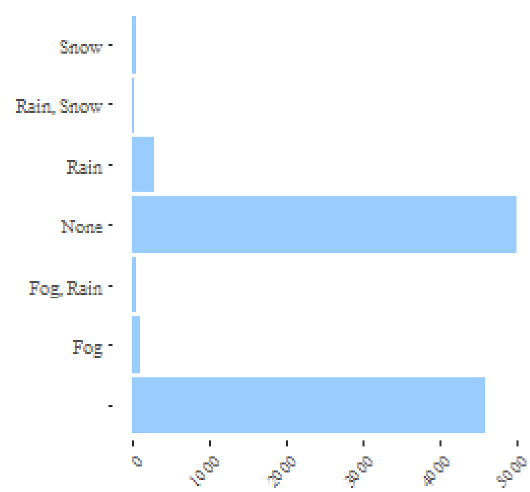
Data distribution of sky condition in 2010

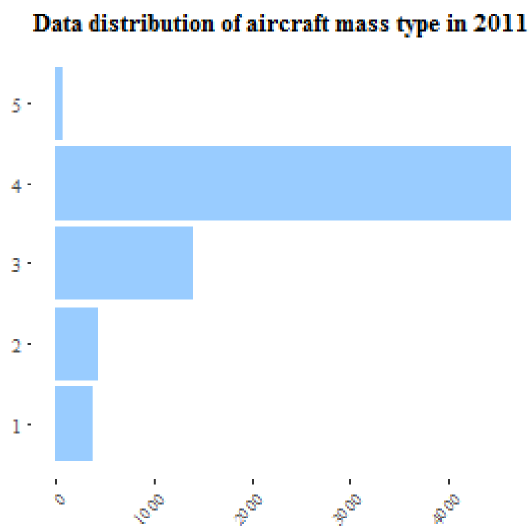


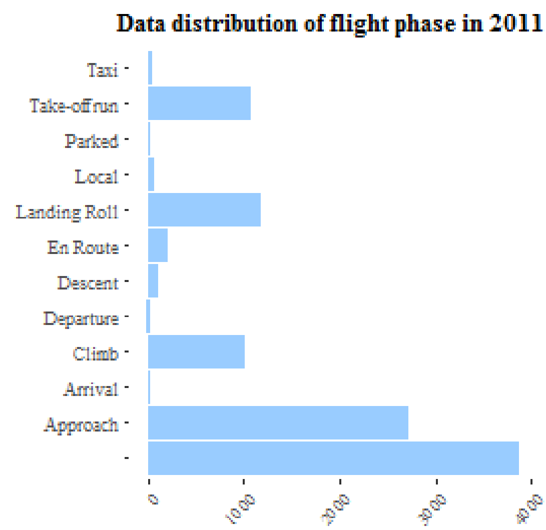
Data distribution of time of day in 2010

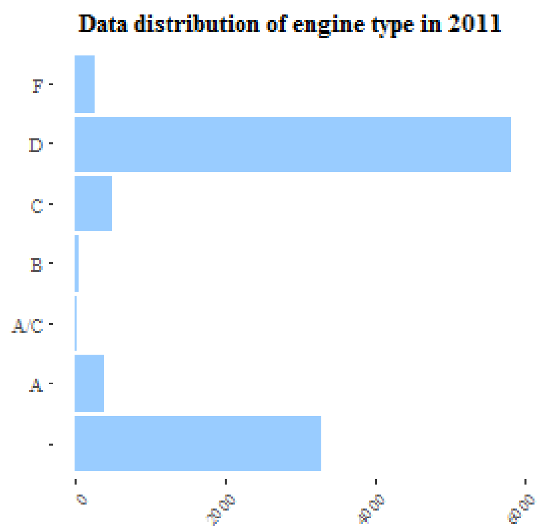


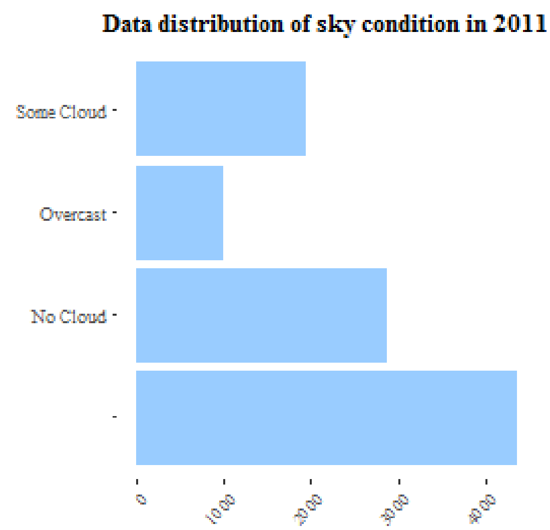
Data distribution of precipitation in 2010

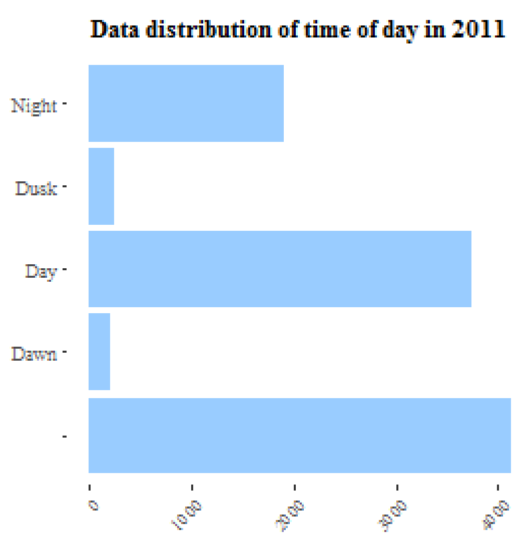


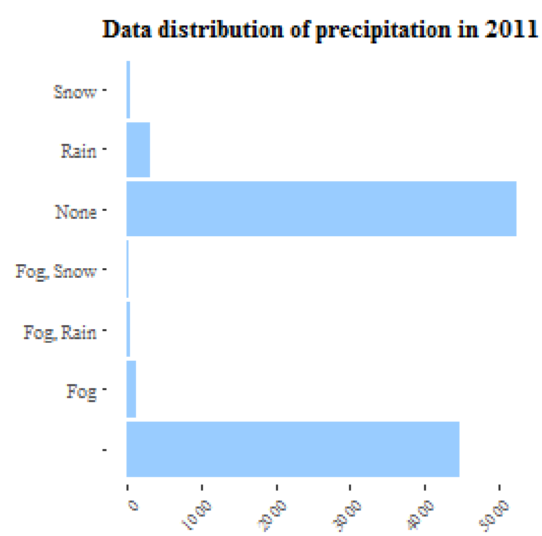

Data distribution of aircraft mass type in 2011
5
4
3
2
1
0
1000
2000
3000
4000


Data distribution of flight phase in 2011
Taxi
Take-off run
Parked
Local
Landing Roll
En Route
Descent
Departure
Climb
Arrival
Approach
0
1000
2000
3000
4000

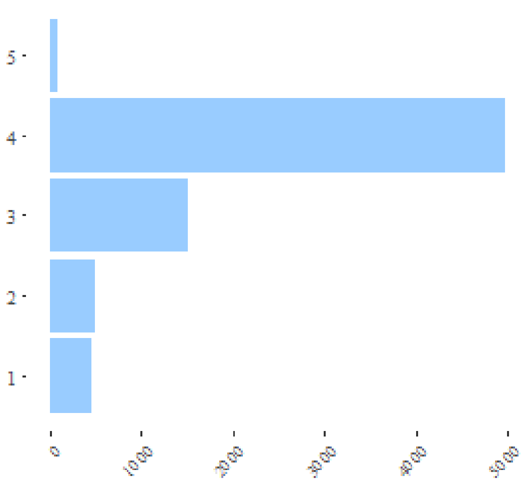

Data distribution of engine type in 2011
F
D
C
B
A/C
A
0
2000
4000
6000


Data distribution of sky condition in 2011
Some Cloud
Overcast
No Cloud
0
1000
2000
3000
4000

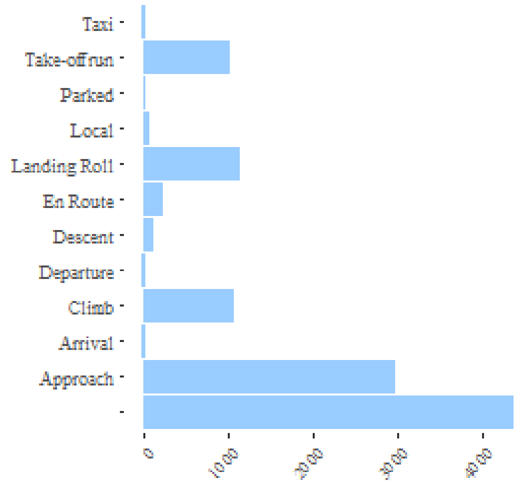

Data distribution of time of day in 2011
Night
Dusk
Day
Dawn
0
1000
2000
3000
4000


Data distribution of precipitation in 2011
Snow
Rain
None
Fog, Snow
Fog, Rain
Fog
0
1000
2000
3000
4000
5000

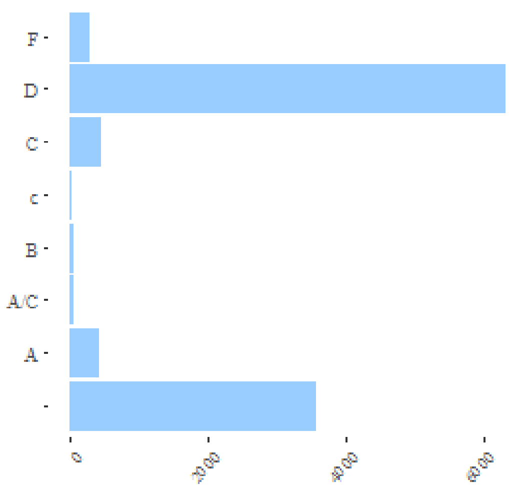
Data distribution of aircraft mass type in 2012



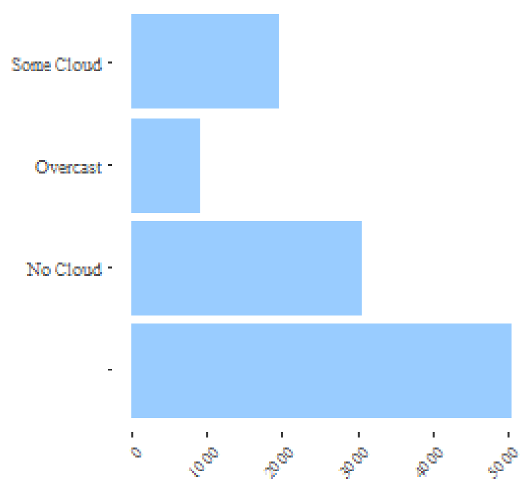
Data distribution of flight phase in 2012



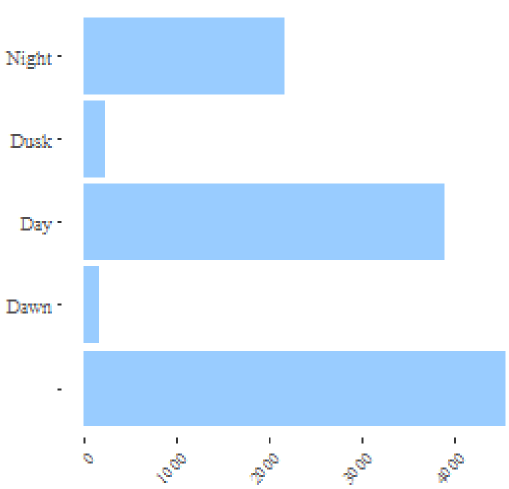
Data distribution of engine type in 2012



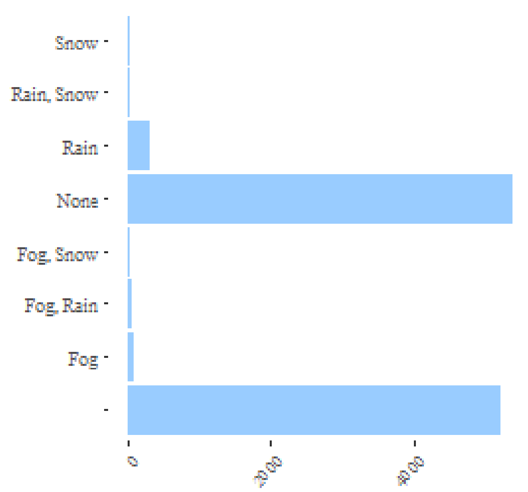
Data distribution of sky condition in 2012

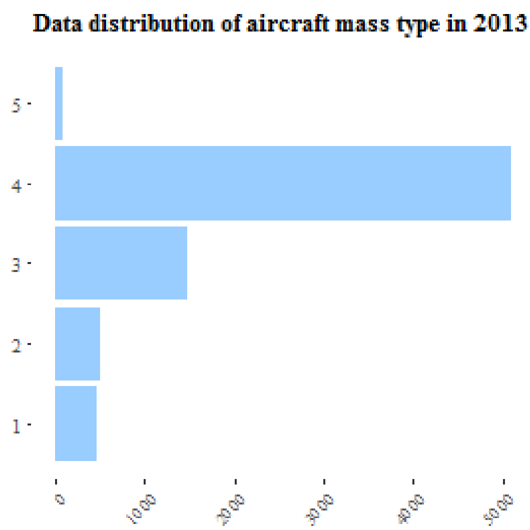


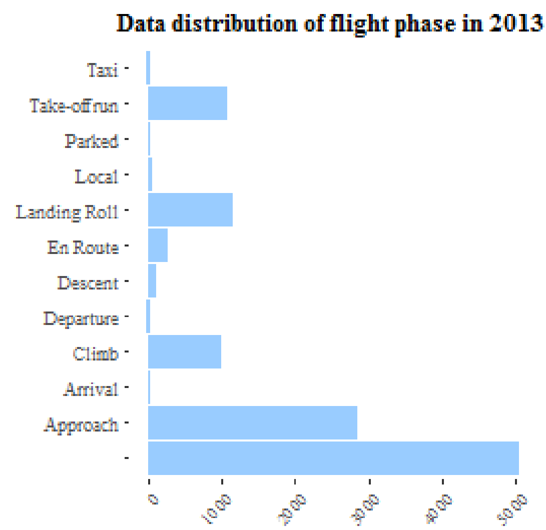
Data distribution of time of day in 2012

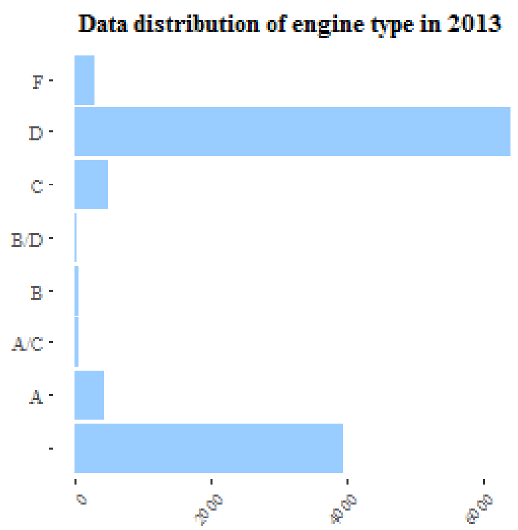


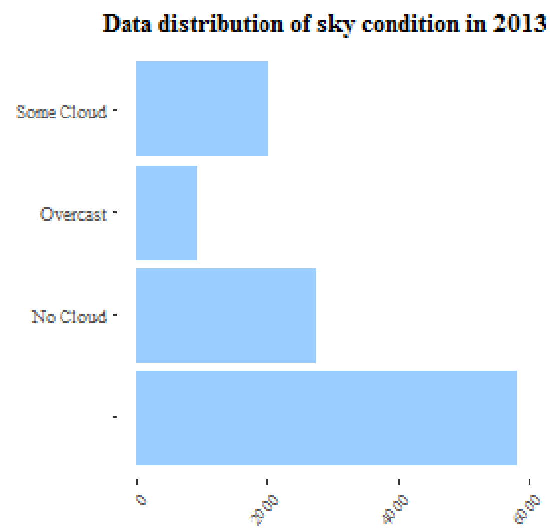
Data distribution of precipitation in 2012

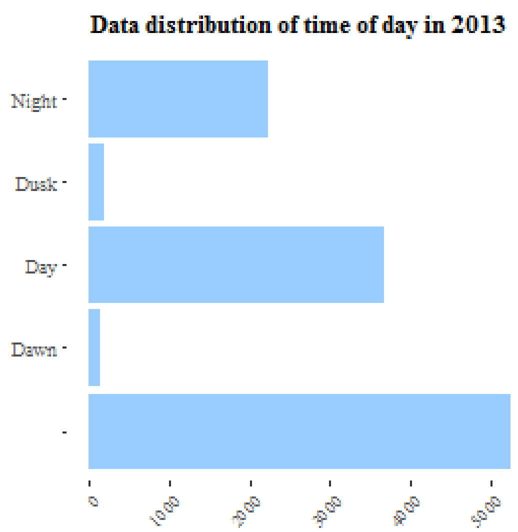


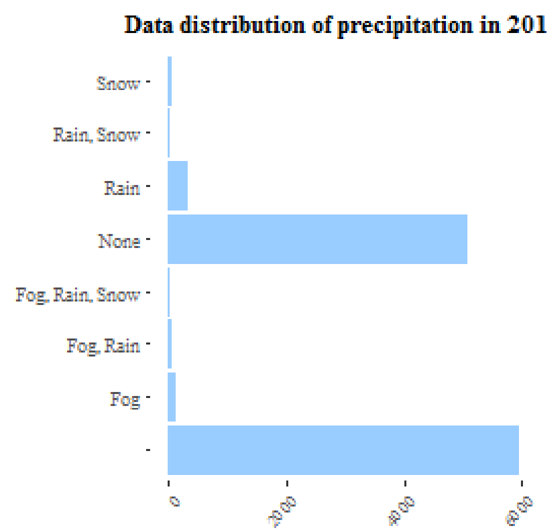

Data distribution of aircraft mass type in 2013
5
4
3
2
1
0
1000
2000
3000
4000
5000


Data distribution of flight phase in 2013
Taxi
Take-off run
Parked
Local
Landing Roll
En Route
Descent
Departure
Climb
Arrival
Approach
0
1000
2000
3000
4000
5000

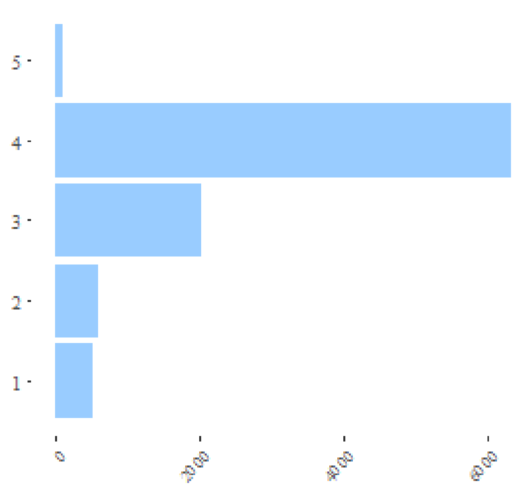

Data distribution of engine type in 2013
F
D
C
B/D
B
A/C
A
0
2000
4000
6000


Data distribution of sky condition in 2013
Some Cloud
Overcast
No Cloud
0
2000
4000
6000

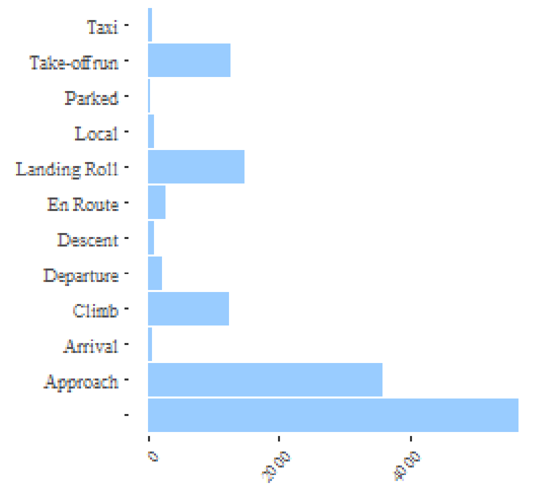

Data distribution of time of day in 2013
Night
Dusk
Day
Dawn
0
1000
2000
3000
4000
5000


Data distribution of precipitation in 201
Snow
Rain, Snow
Rain
None
Fog, Rain, Snow
Fog, Rain
Fog
0
2000
4000
6000

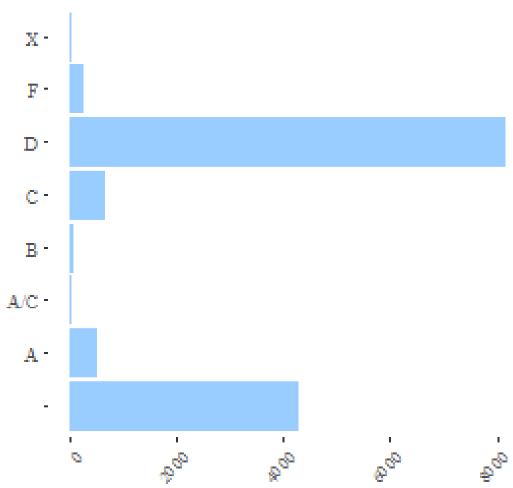
Data distribution of aircraft mass type in 2014



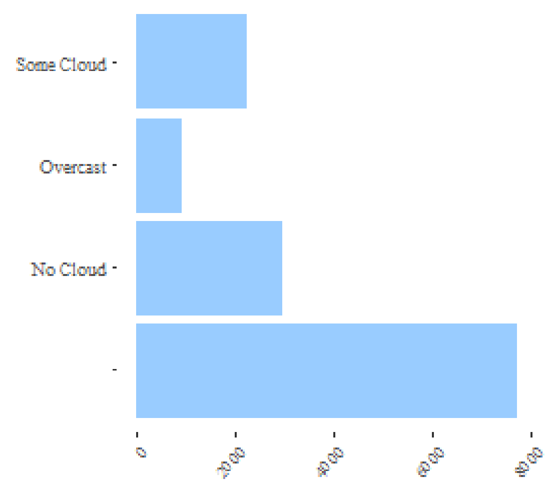
Data distribution of flight phase in 2014



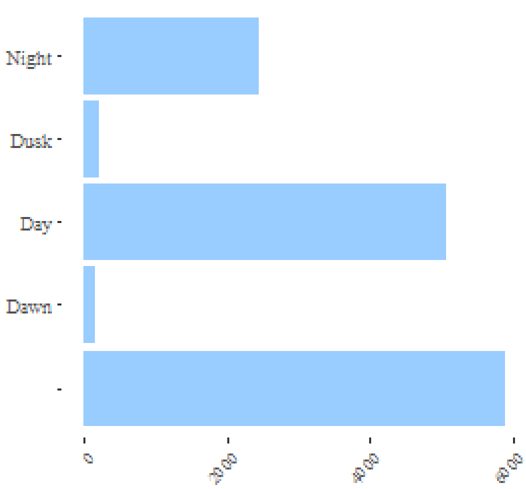
Data distribution of engine type in 2014



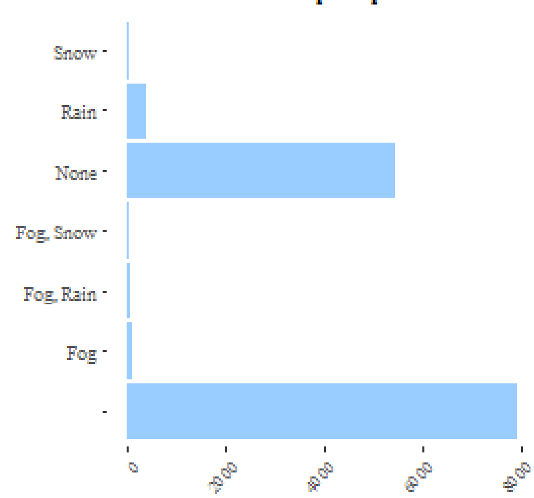
Data distribution of sky condition in 2014



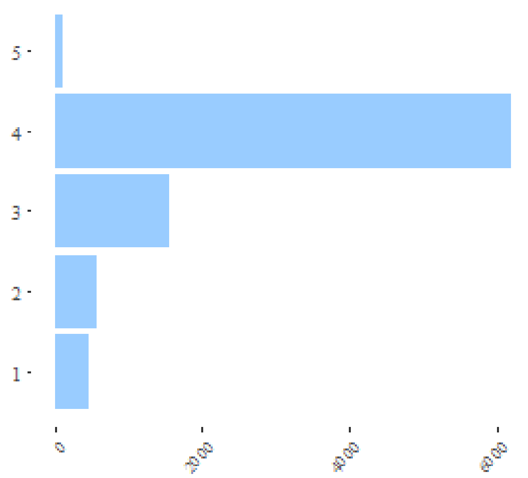
Data distribution of time of day in 2014



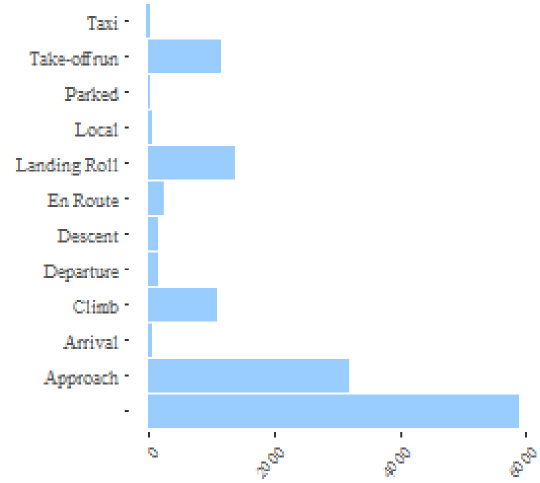
Data distribution of precipitation in 2014



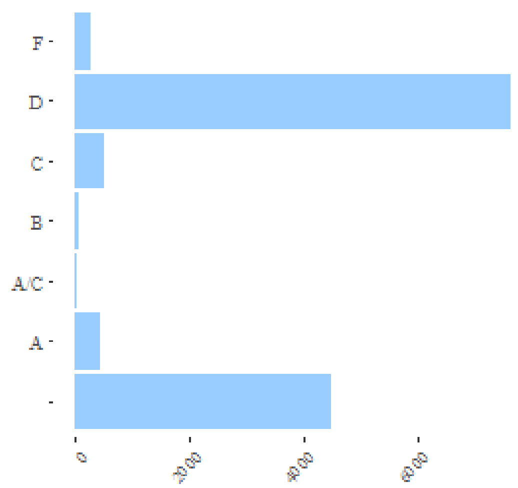
Data distribution of aircraft mass type in 2015



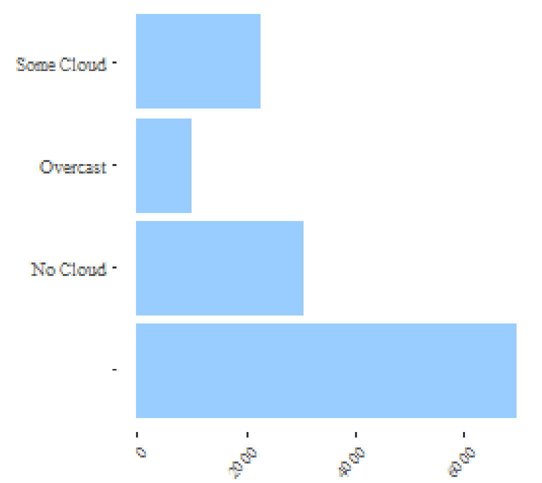
Data distribution of flight phase in 2015



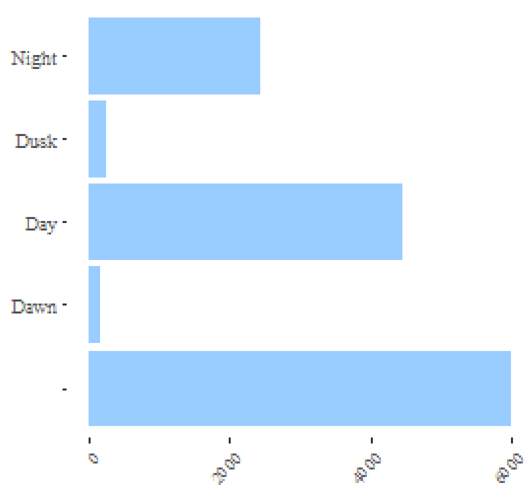
Data distribution of engine type in 2015



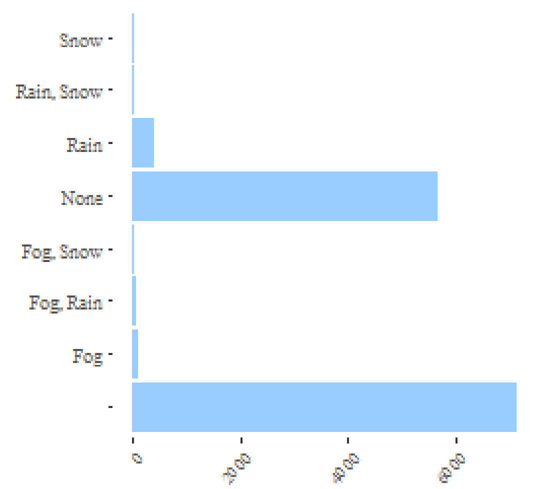
Data distribution of sky condition in 2015



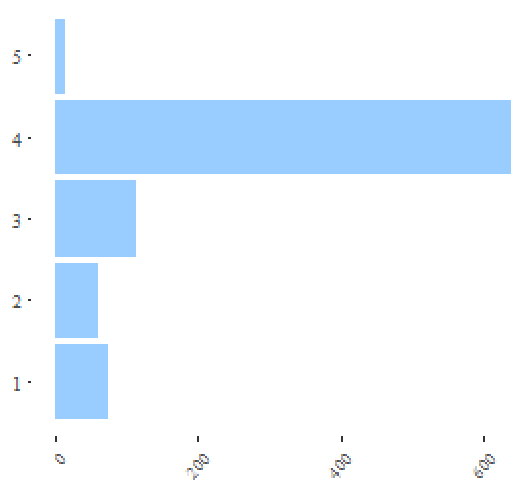
Data distribution of time of day in 2015



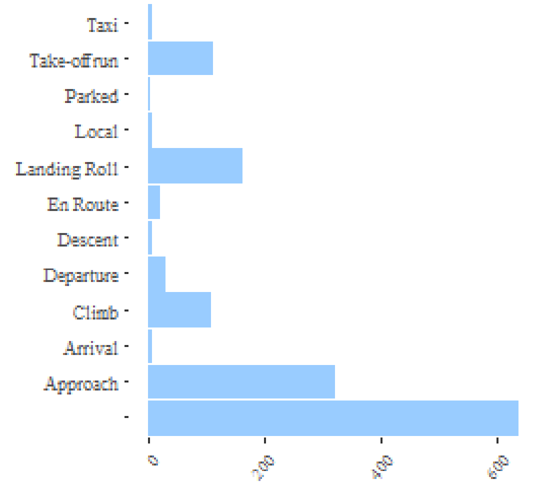
Data distribution of precipitation in 2015



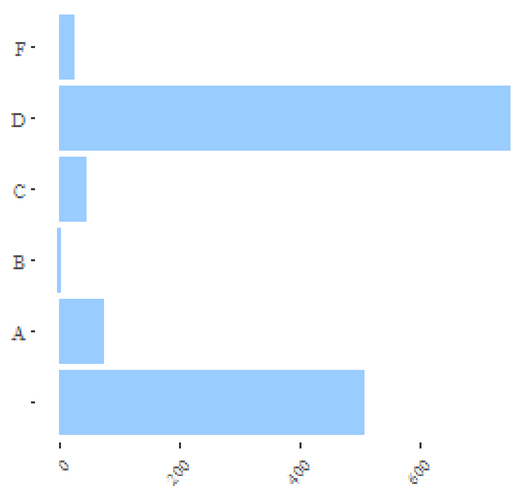
Data distribution of aircraft mass type in 2016



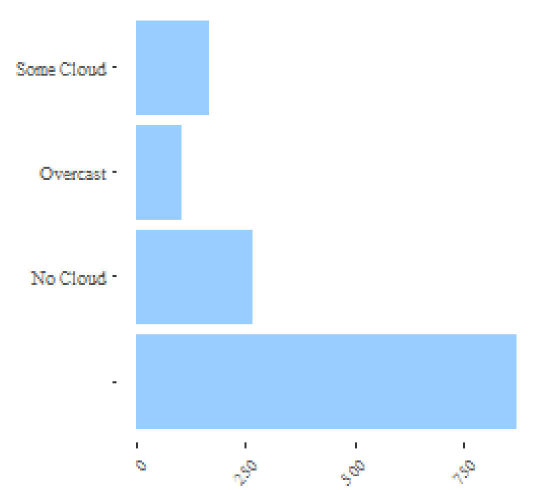
Data distribution of flight phase in 2016



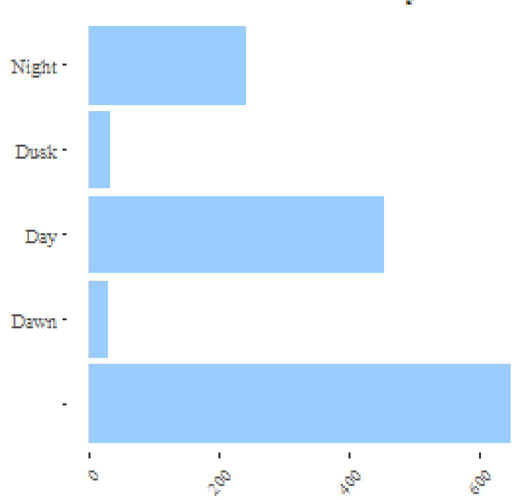
Data distribution of engine type in 2016



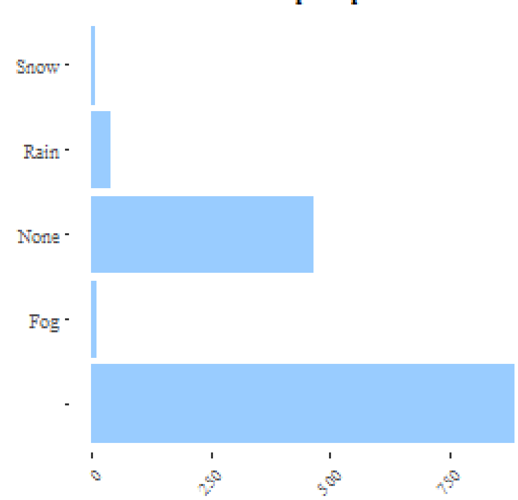
Data distribution of sky condition in 2016



Data distribution of time of day in 2016



Data distribution of precipitation in 2016



14.1.2 Flight Data (1990 - 2016)

The first summary table shows the number of distinct items for each year regarding the number of records, the carriers, and the origin and destination airports.

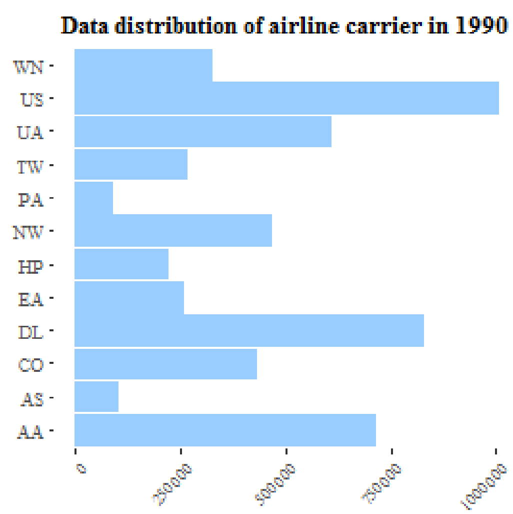
Year	# of flights	# of carriers	Origin airports	Origin states	Destination airports	Destination states
1990	5270893	12	235	53	236	53
1991	5076925	12	233	52	233	52
1992	5092157	10	233	52	233	52
1993	5070501	10	227	52	227	52
1994	5180048	10	224	52	225	52
1995	5327435	10	218	52	218	52
1996	5351983	10	211	52	212	52
1997	5411843	10	205	51	206	52
1998	5384721	10	207	51	208	51
1999	5527884	10	205	51	205	51
2000	5683047	11	206	51	206	51
2001	5967780	12	231	51	230	51
2002	5271359	10	218	50	219	50
2003	6488540	18	282	51	282	51
2004	7129270	19	285	51	288	51
2005	7140596	20	286	51	289	51
2006	7141922	21	289	52	296	52
2007	7455458	20	304	52	310	52
2008	7009726	20	303	51	304	51
2009	6450285	19	296	51	296	51
2010	6450117	18	305	52	305	52
2011	6085281	16	299	52	301	52
2012	6096762	15	312	52	313	52
2013	6369482	16	320	53	318	53
2014	5819811	14	325	53	324	53
2015	5819079	14	322	53	322	53
2016	5617658	12	313	52	310	52

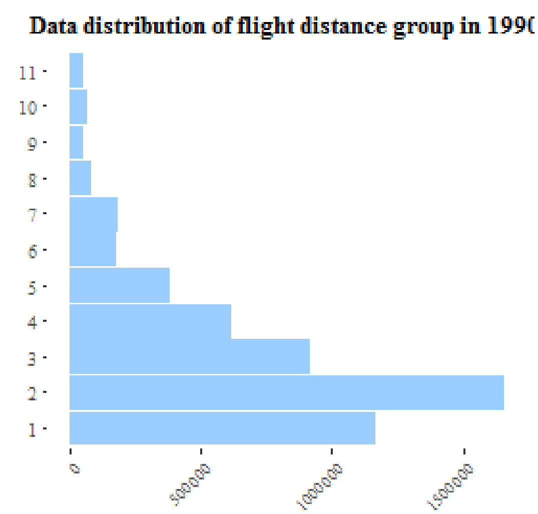
The second summary table shows the number of distinct items for each year the departure time group and distance between the airports.

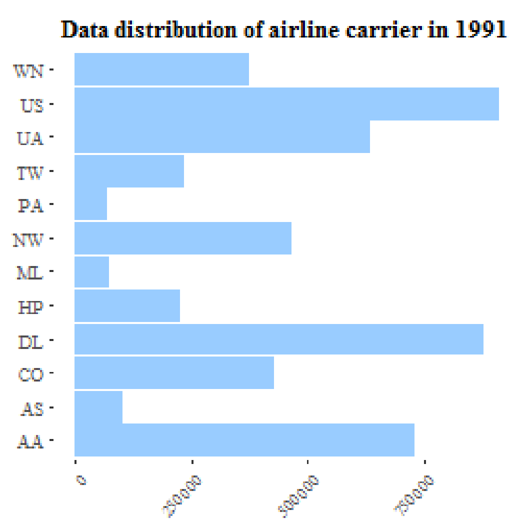
Year	Departure time block	Distance group
1990	19	11
1991	19	11
1992	19	11
1993	19	11
1994	19	11
1995	19	11
1996	19	11
1997	19	11
1998	19	11
1999	19	11
2000	19	11
2001	19	11
2002	19	11
2003	19	11

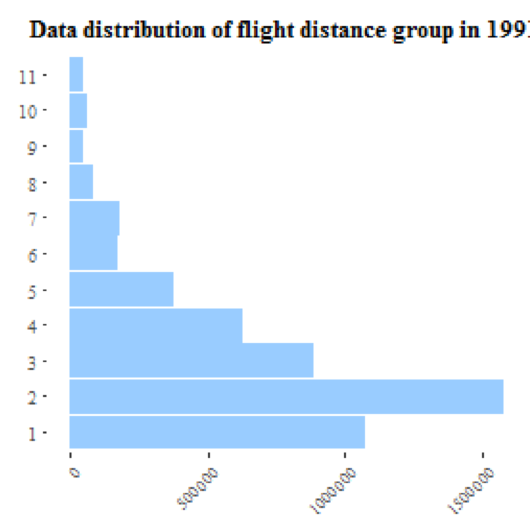
Year	Departure time block	Distance group
2004	19	11
2005	19	11
2006	19	11
2007	19	11
2008	19	11
2009	19	11
2010	19	11
2011	19	11
2012	20	11
2013	19	11
2014	19	11
2015	19	11
2016	19	11

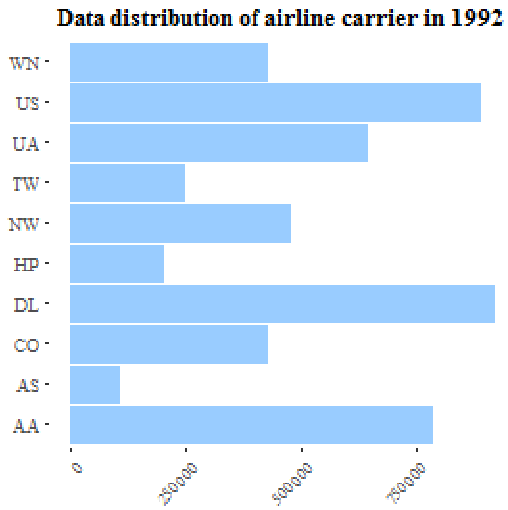
The following graphs show the distributions of some of the selected distinct items summarized in the tables above.

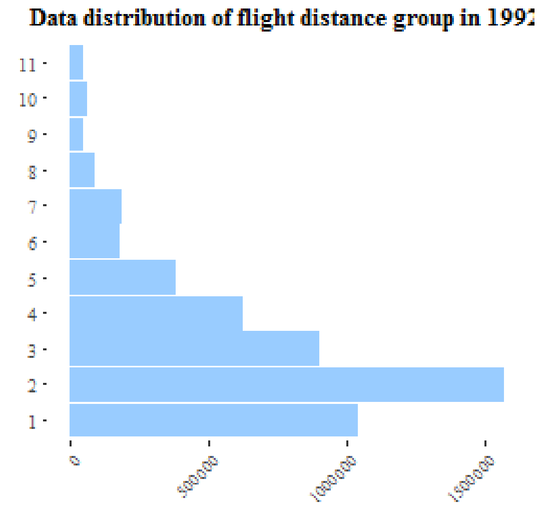


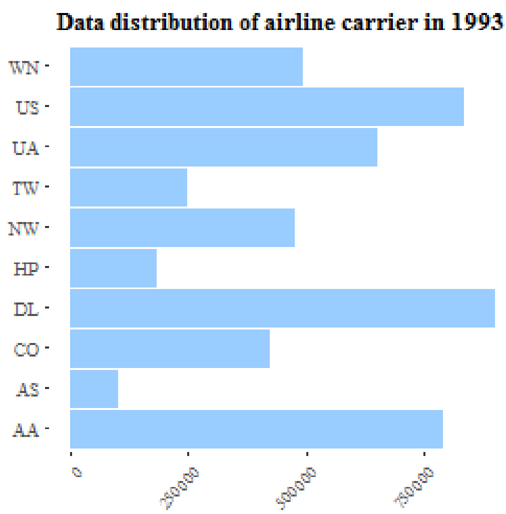


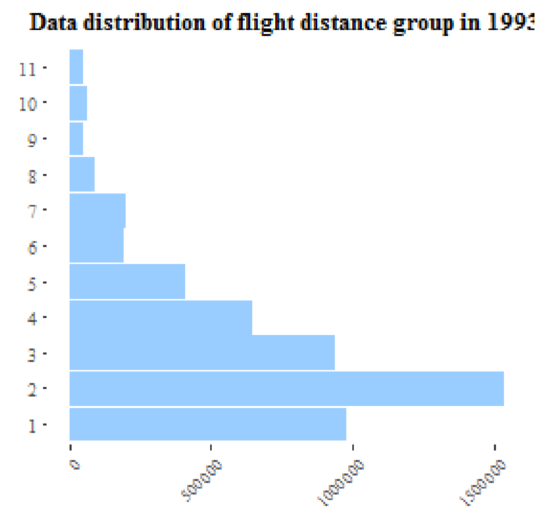


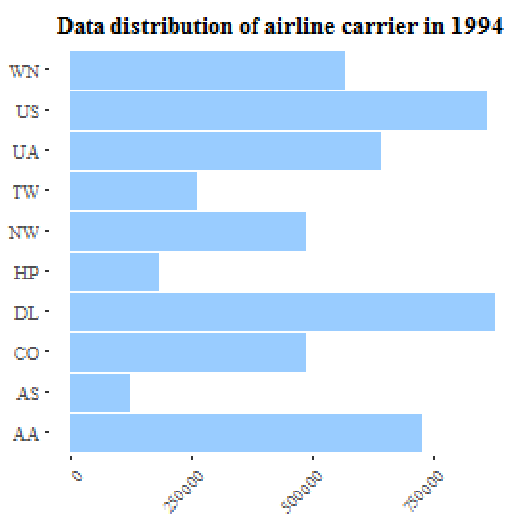


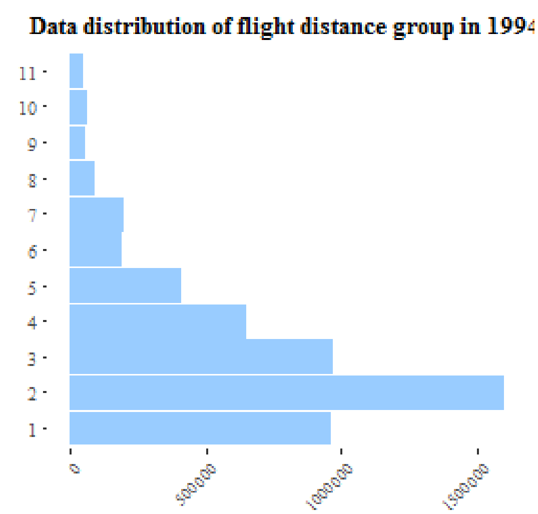

Data distribution of airline carrier in 1992
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

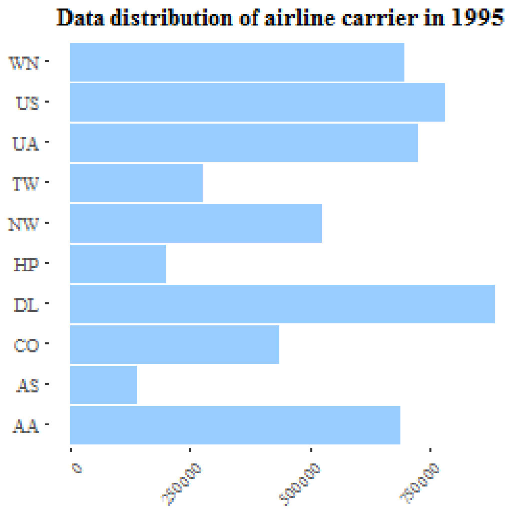

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

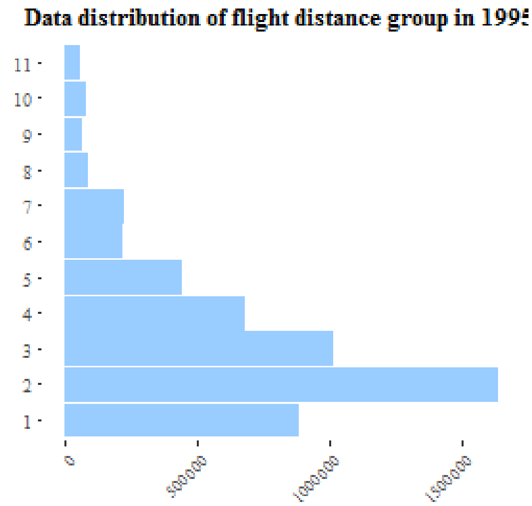

Data distribution of airline carrier in 1993
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

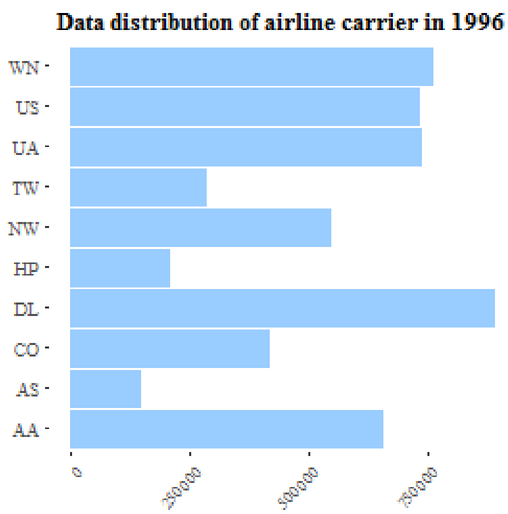

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

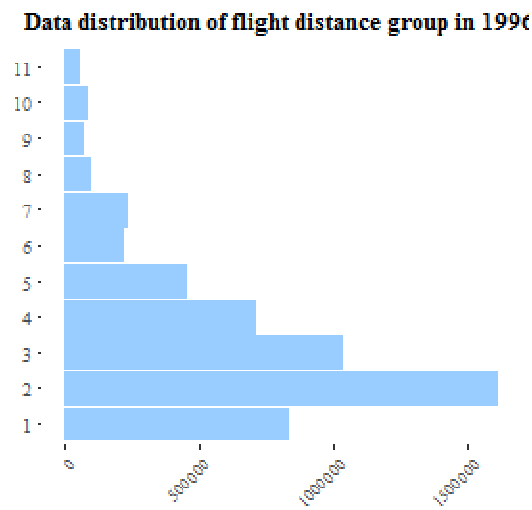

Data distribution of airline carrier in 1994
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

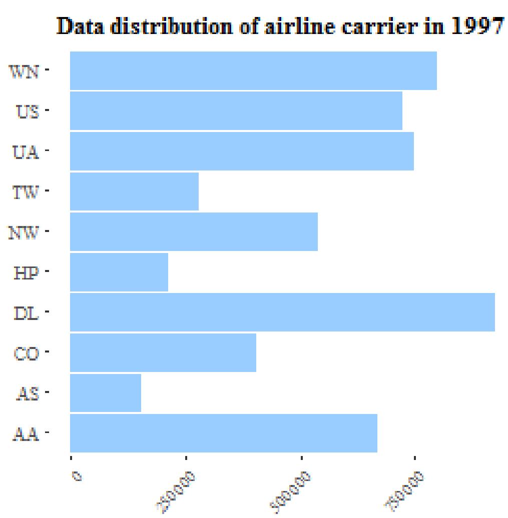

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

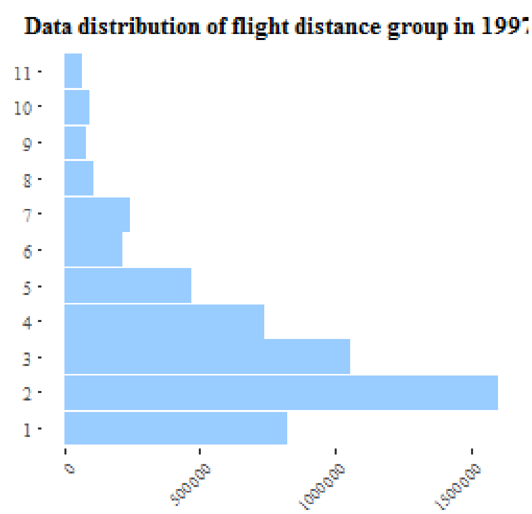

Data distribution of airline carrier in 1995
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

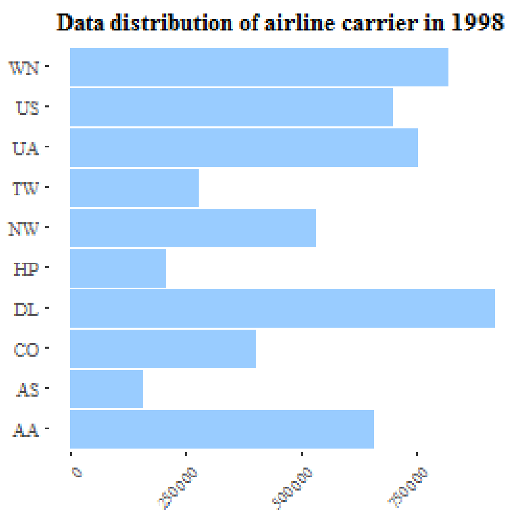

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

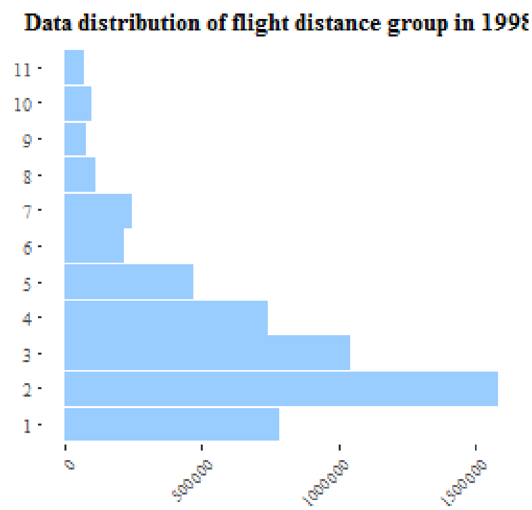

Data distribution of airline carrier in 1996
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

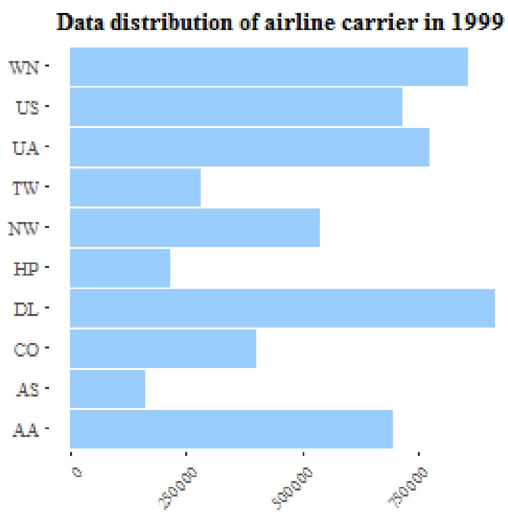

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

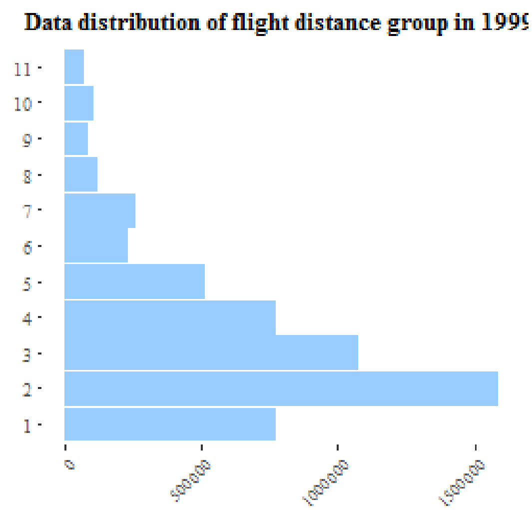

Data distribution of airline carrier in 1997
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

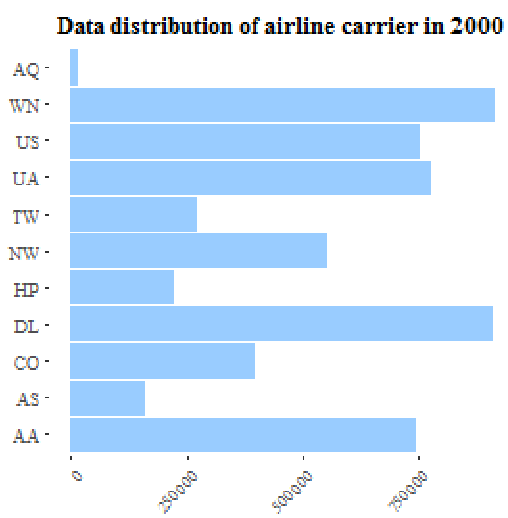

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

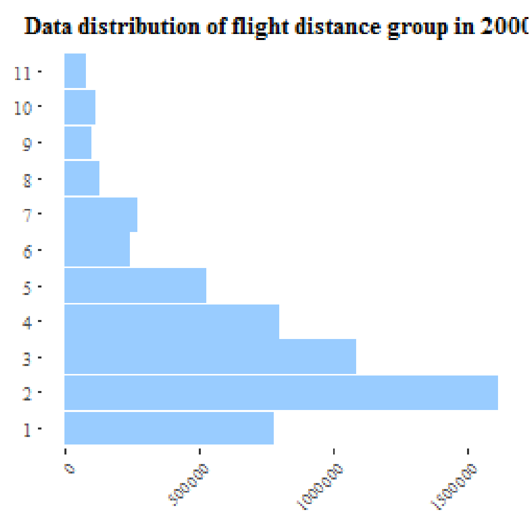

Data distribution of airline carrier in 1998
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

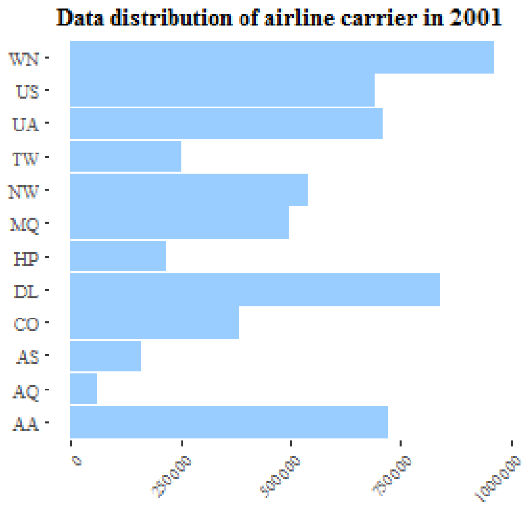

Data distribution of flight distance group in 1998
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

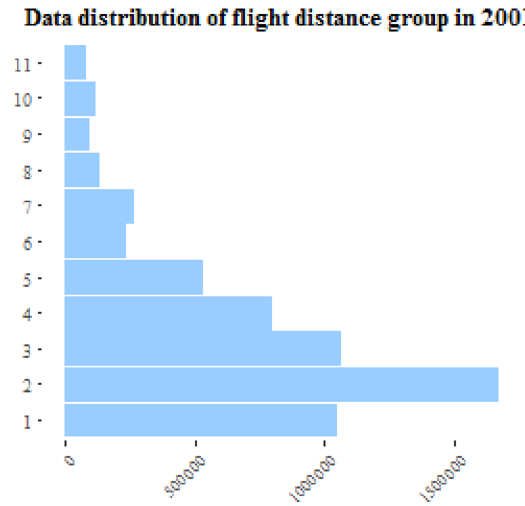

Data distribution of airline carrier in 1999
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

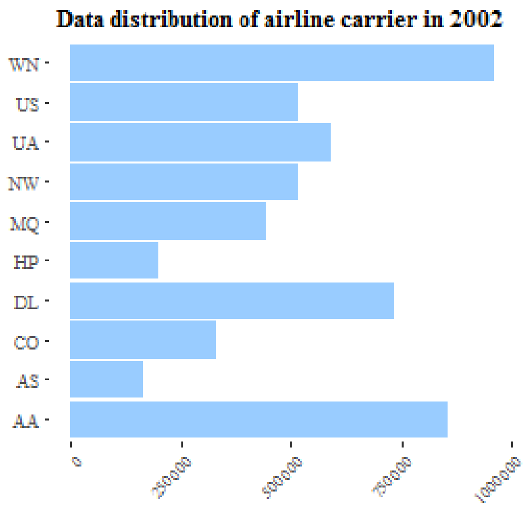

Data distribution of flight distance group in 1999
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

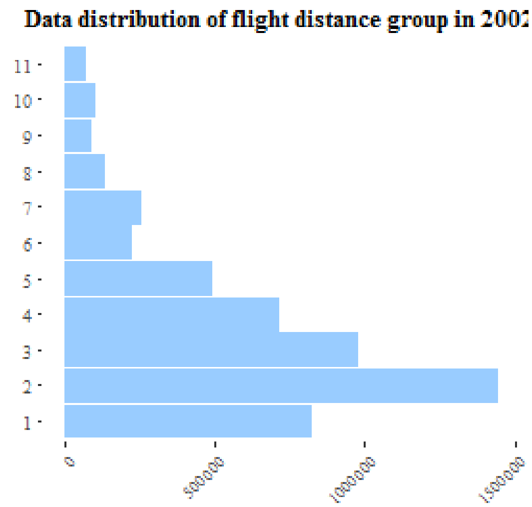

Data distribution of airline carrier in 2000
AQ
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

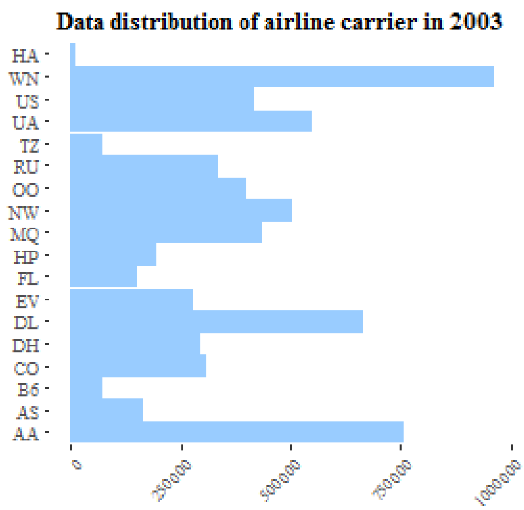

Data distribution of flight distance group in 2000
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

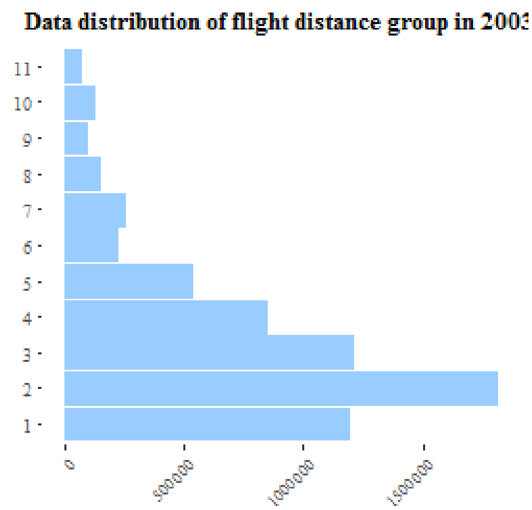

Data distribution of airline carrier in 2001
WN
US
UA
TW
NW
MQ
HP
DL
CO
AS
AQ
AA
0
250000
500000
750000
1000000

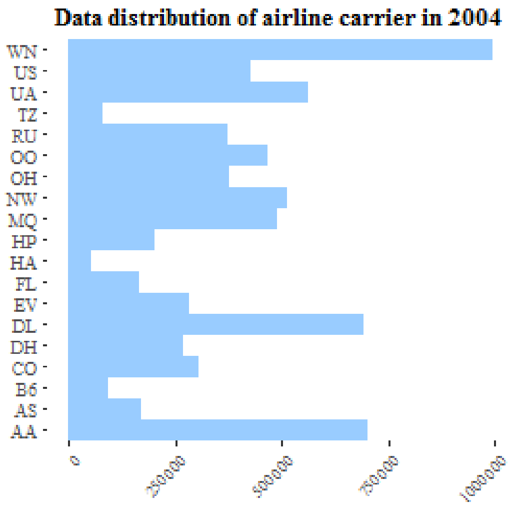

Data distribution of flight distance group in 2001
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

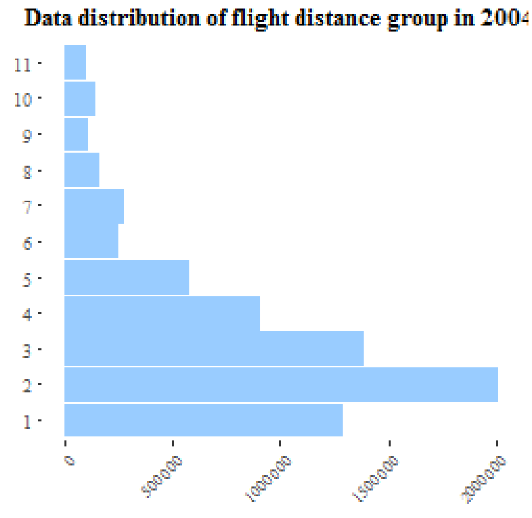

Data distribution of airline carrier in 2002
WN
US
UA
NW
MQ
HP
DL
CO
AS
AA
0
250000
500000
750000
1000000

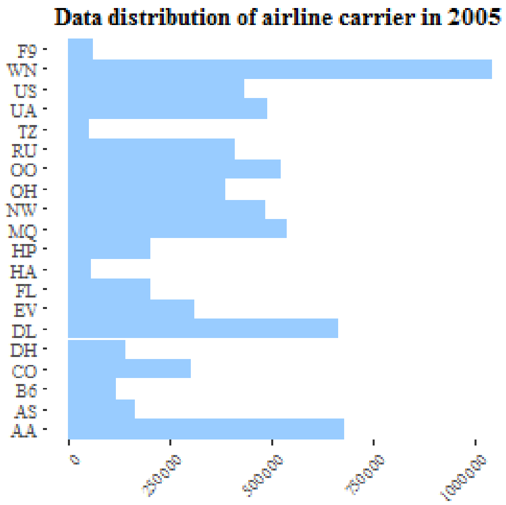

Data distribution of flight distance group in 2002
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

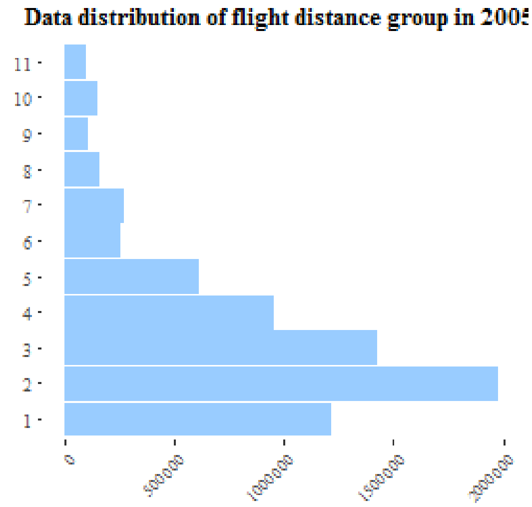

Data distribution of airline carrier in 2003
HA
WN
US
UA
TZ
RU
OO
NW
MQ
HP
FL
EV
DL
DH
CO
B6
AS
AA
0
250000
500000
750000
1000000

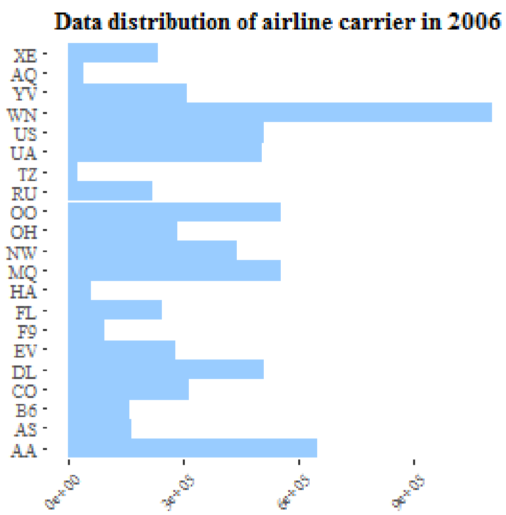

Data distribution of flight distance group in 2003
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

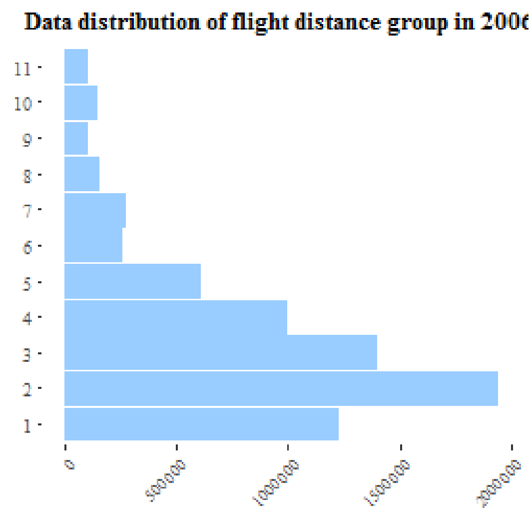

Data distribution of airline carrier in 2004
WN
US
UA
TZ
RU
OO
OH
NW
MQ
HP
HA
FL
EV
DL
DH
CO
B6
AS
AA
0
250000
500000
750000
1000000

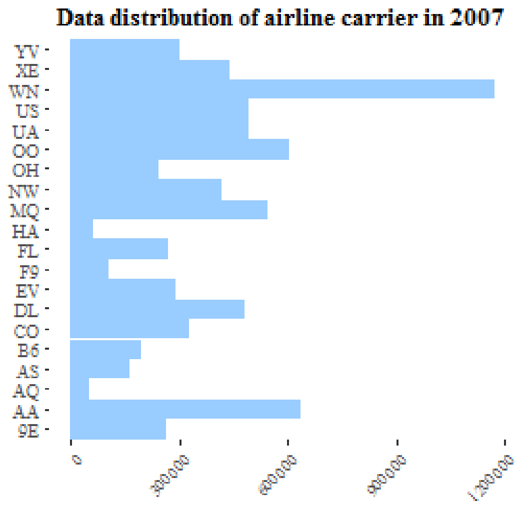

Data distribution of flight distance group in 2004
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

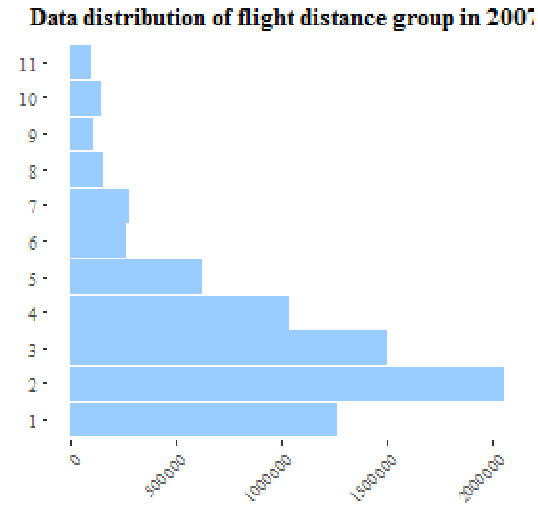

Data distribution of airline carrier in 2005
F9
WN
US
UA
TZ
RU
OO
OH
NW
MQ
HP
HA
FL
EV
DL
DH
CO
B6
AS
AA
0
250000
500000
750000
1000000

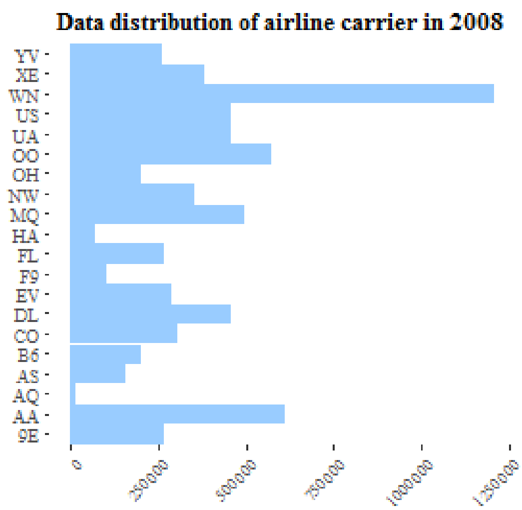

Data distribution of flight distance group in 2005
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

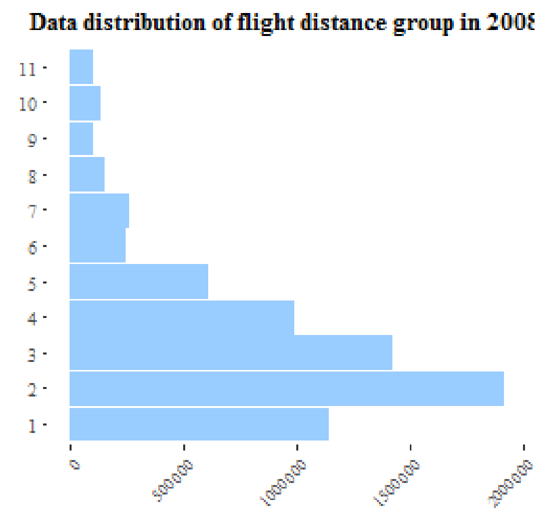

Data distribution of airline carrier in 2006
XE
AQ
YV
WN
US
UA
TZ
RU
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
0e+00
3e+05
6e+05
9e+05

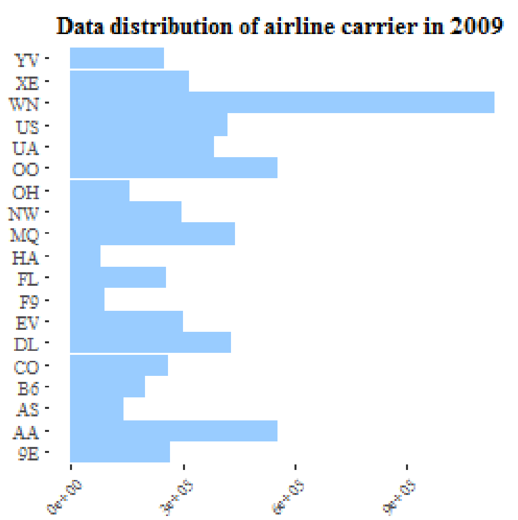

Data distribution of flight distance group in 2006
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

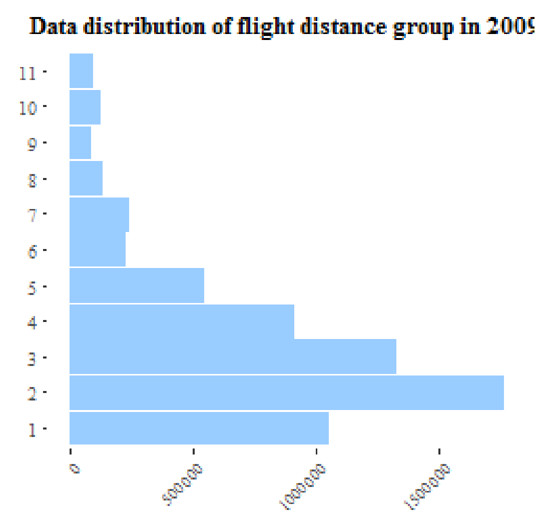

Data distribution of airline carrier in 2007
YV
XE
WN
US
UA
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AQ
AA
9E
0
300000
600000
900000
1200000

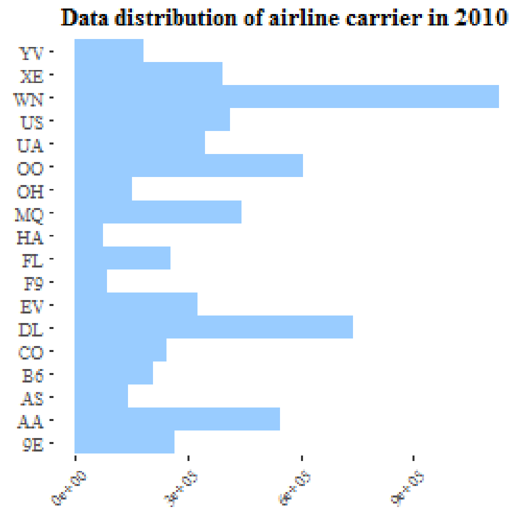

Data distribution of flight distance group in 2007
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

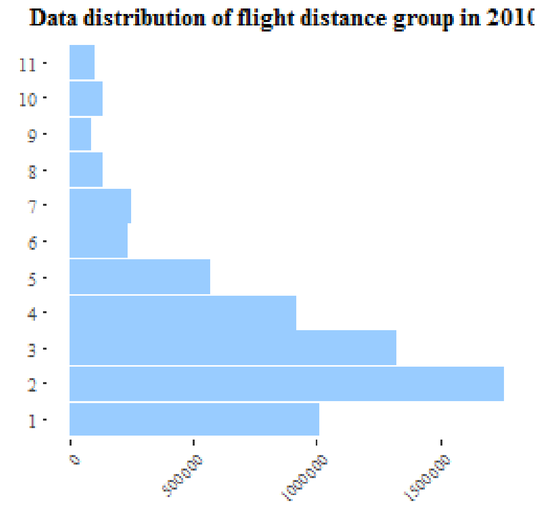

Data distribution of airline carrier in 2008
YV
XE
WN
US
UA
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AQ
AA
9E
0
250000
500000
750000
1000000
1250000

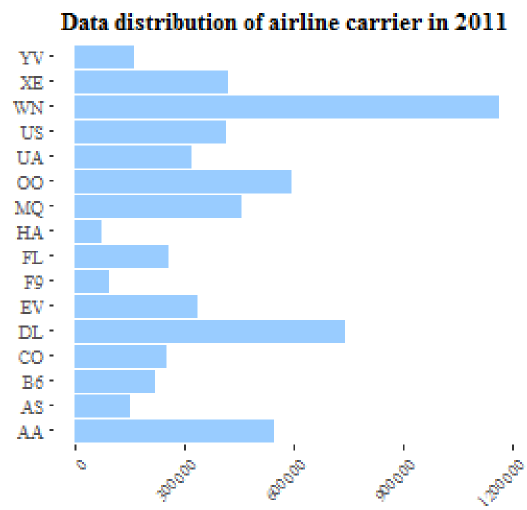

Data distribution of flight distance group in 2008
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

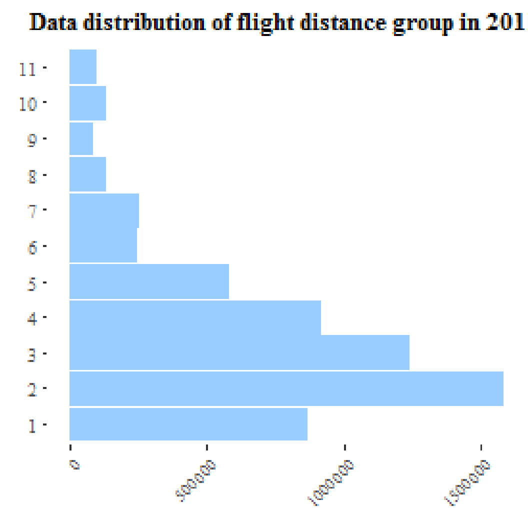

Data distribution of airline carrier in 2009
YV
XE
WN
US
UA
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
9E
0e+00
3e+05
6e+05
9e+05

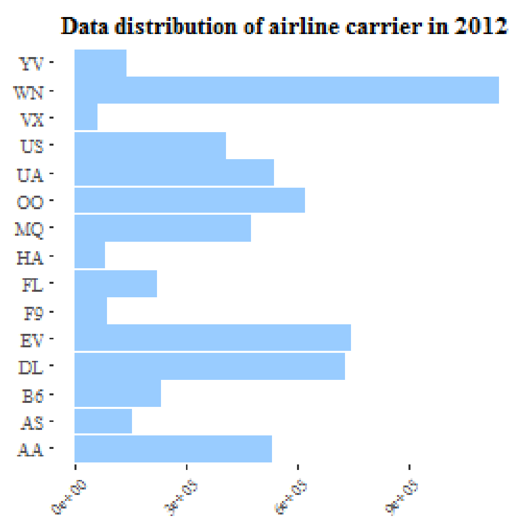

Data distribution of flight distance group in 2009
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

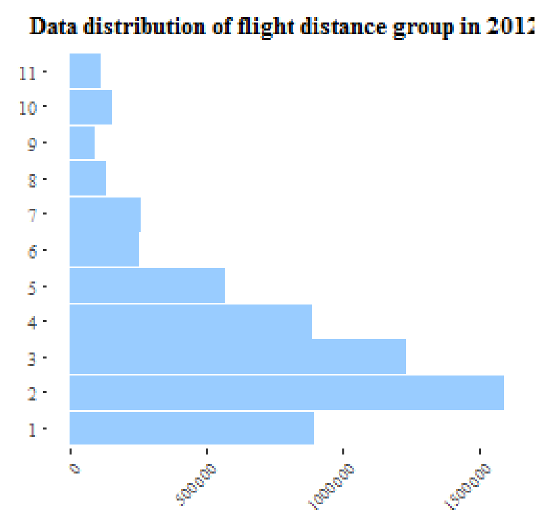

Data distribution of airline carrier in 2010
YV
XE
WN
US
UA
OO
OH
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
9E
0e+00
3e+05
6e+05
9e+05

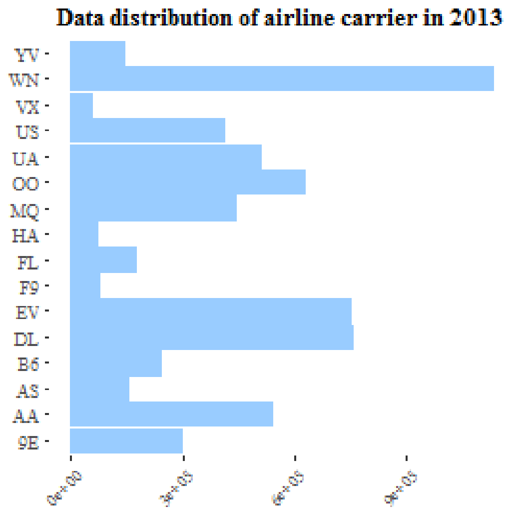

Data distribution of flight distance group in 201(
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

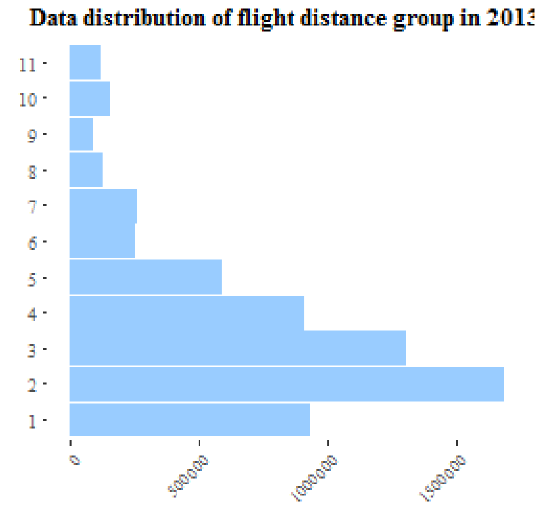

Data distribution of airline carrier in 2011
YV
XE
WN
US
UA
OO
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
0
300000
600000
900000
1200000

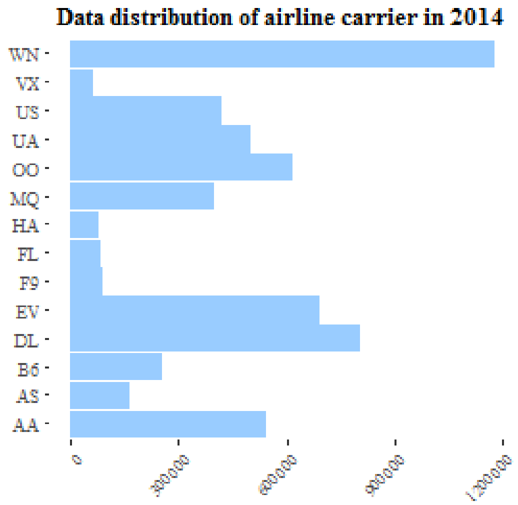

Data distribution of flight distance group in 2011
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

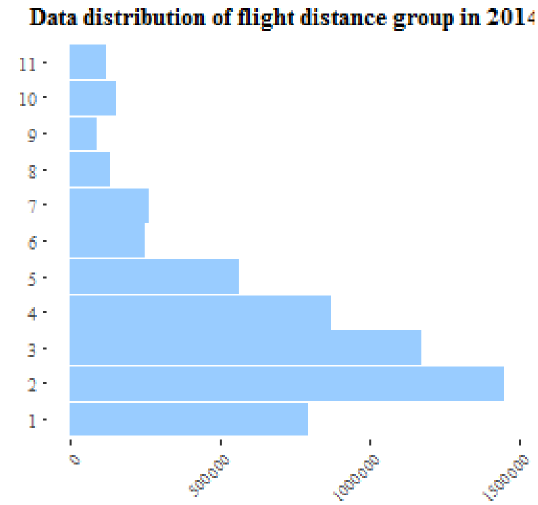

Data distribution of airline carrier in 2012
YV
WN
VX
US
UA
OO
MQ
HA
FL
F9
EV
DL
B6
AS
AA
0e+00
3e+05
6e+05
9e+05

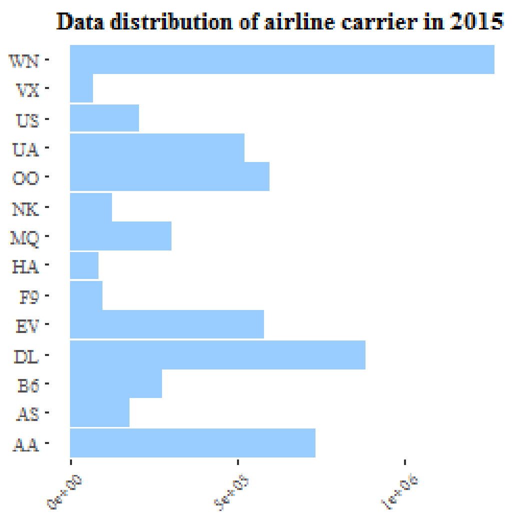

Data distribution of flight distance group in 2012
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

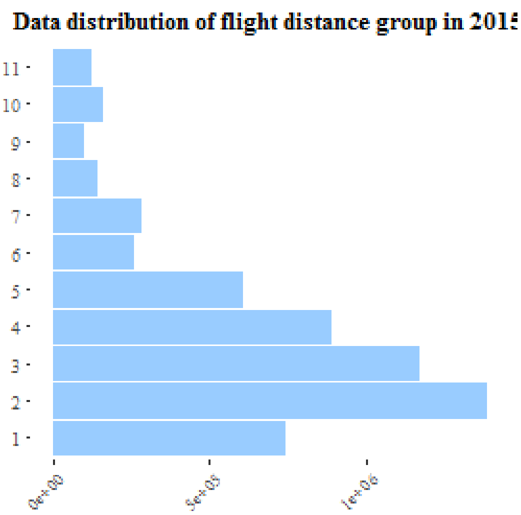

Data distribution of airline carrier in 2013
YV
WN
VX
US
UA
OO
MQ
HA
FL
F9
EV
DL
B6
AS
AA
9E
0e+00
3e+05
6e+05
9e+05

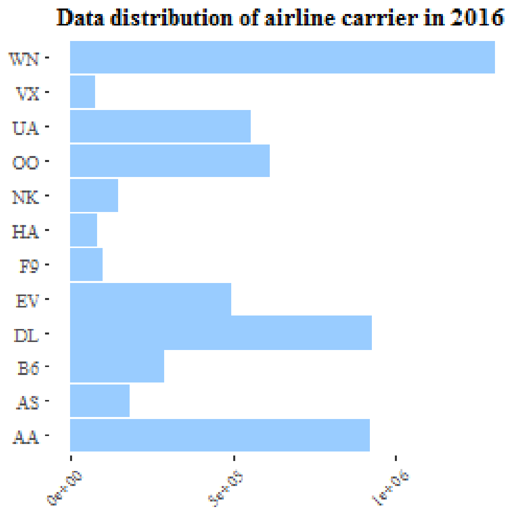

Data distribution of flight distance group in 201
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

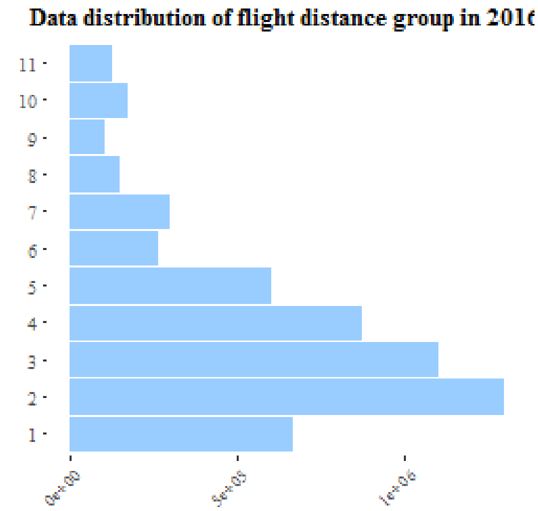

Data distribution of airline carrier in 2014
WN
VX
US
UA
OO
MQ
HA
FL
F9
EV
DL
B6
AS
AA
0
300000
600000
900000
1200000


Data distribution of flight distance group in 201
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000


Data distribution of airline carrier in 2015
WN
VX
US
UA
OO
NK
MQ
HA
F9
EV
DL
B6
AS
AA
0e+00
5e+05
1e+06


Data distribution of flight distance group in 201
11
10
9
8
7
6
5
4
3
2
1
0e+00
5e+05
1e+06


Data distribution of airline carrier in 2016
WN
VX
UA
OO
NK
HA
F9
EV
DL
B6
AS
AA
0e+00
5e+05
1e+06


Data distribution of flight distance group in 2016
11
10
9
8
7
6
5
4
3
2
1
0e+00
5e+05
1e+06

15 Appendix 6 - Full Data Exploration Report (1990-2016) after Cleanup

15.1 Data Exploration Report (1990 - 2016)

15.1.1 Animal Strike Data (1990 - 2016)

The first summary table shows the number of distinct items for each year regarding the Airline operators, Aircraft, Aircraft types, Aircraft mass types, and Engine types, which have been reported as being affected in an animal strike after the selection and cleanup tasks. (Please note that the data for 2016 is available until 30-4-2016.)

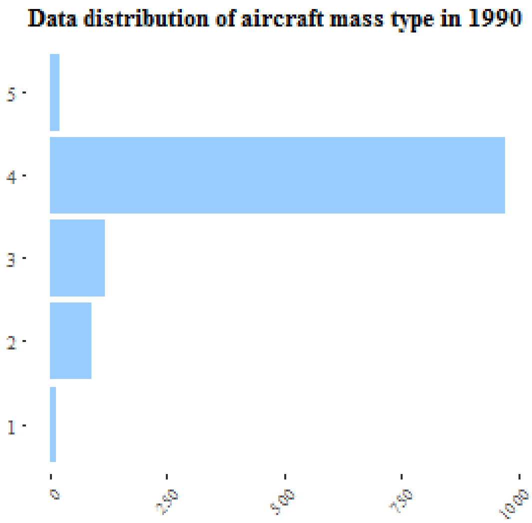
Year	# of reports	Operators	Aircraft	Aircraft type	Aircraft mass type	Engine type
1990	1190	94	79	1	5	4
1991	1576	112	94	1	5	4
1992	1648	112	90	1	5	4
1993	1640	114	89	1	5	4
1994	1730	124	92	1	5	4
1995	1760	134	98	1	5	4
1996	1798	107	82	1	5	3
1997	2083	113	82	1	5	4
1998	2160	120	100	1	5	4
1999	2730	116	98	1	5	4
2000	3144	133	109	1	5	4
2001	2998	122	114	1	5	4
2002	3228	114	107	1	5	4
2003	3179	129	118	1	5	4
2004	3562	131	117	1	5	4
2005	3659	137	109	1	5	3
2006	3787	130	107	1	5	4
2007	4104	128	104	1	5	3
2008	3891	128	97	1	5	4
2009	5072	138	92	1	5	4
2010	4903	139	103	1	5	4
2011	4854	136	107	1	5	3
2012	5049	149	103	1	5	4
2013	4916	134	99	1	5	4
2014	6234	148	98	1	5	4
2015	5637	133	96	1	5	3
2016	539	61	54	1	4	2

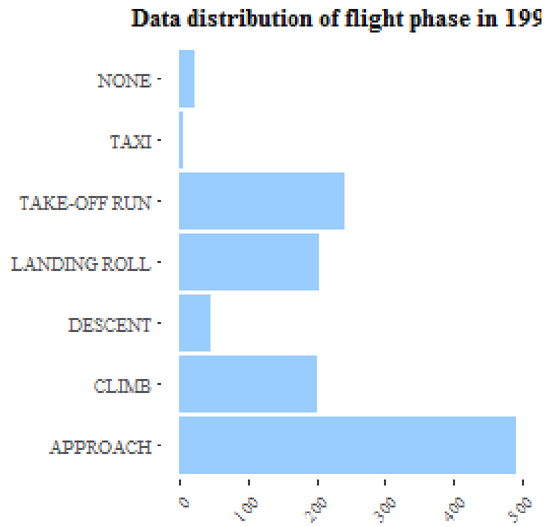
The second summary table shows the number of distinct items for each year regarding the Time of day, Airports, States, Phase of flight, weather conditions (Sky and Precipitation), and the flag for showing if the pilot has been warned or not about birds / wildlife in the reports after the selection and cleanup tasks. (Please note that the data for 2016 is available until 30-4-2016.)

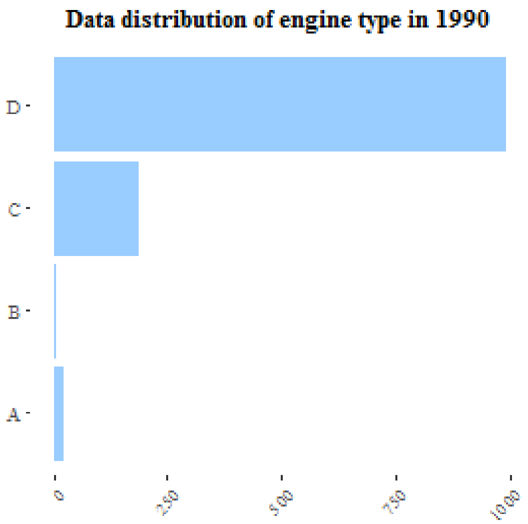
Year	Time of day	Airports	States	Phase of flight	Sky	Precipitation	Warned
1990	5	208	49	7	4	4	3
1991	5	223	47	8	4	4	3
1992	5	242	48	9	4	5	3
1993	5	229	48	9	4	4	3
1994	5	241	49	7	4	5	3
1995	5	235	48	9	4	5	3
1996	5	216	48	9	4	4	3

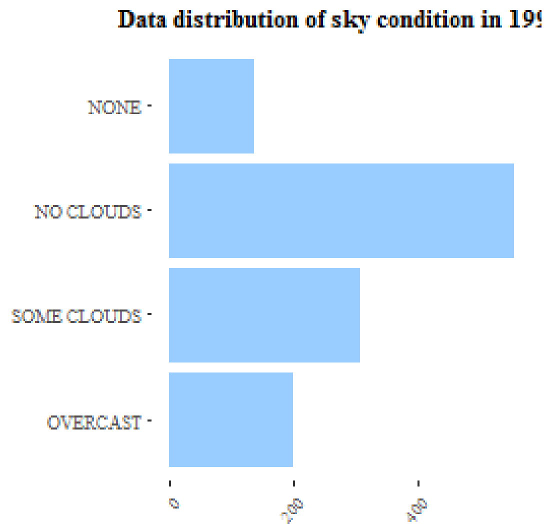
Year	Time of day	Airports	States	Phase of flight	Sky	Precipitation	Warned
1997	5	255	49	9	4	5	3
1998	5	262	49	8	4	4	3
1999	5	278	49	9	4	4	3
2000	5	289	48	10	4	5	3
2001	5	293	50	10	4	5	3
2002	5	281	50	10	4	8	3
2003	5	289	49	9	4	7	3
2004	5	277	49	10	4	7	3
2005	5	296	50	11	4	7	3
2006	5	288	50	9	4	6	3
2007	5	293	49	9	4	5	3
2008	5	292	49	10	4	5	3
2009	5	336	49	11	4	6	3
2010	5	328	49	10	4	6	3
2011	5	314	49	10	4	6	3
2012	5	349	50	10	4	7	3
2013	5	321	50	10	4	6	3
2014	5	354	50	10	4	6	3
2015	5	329	50	11	4	7	3
2016	5	143	45	9	4	4	3

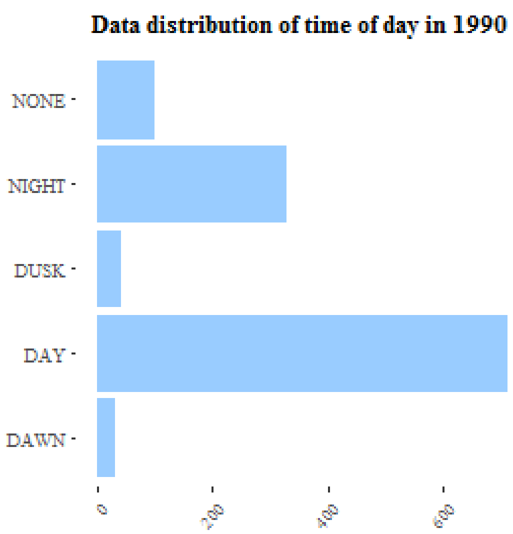
The following graphs show the distributions of some of the selected distinct items summarized in the tables above.

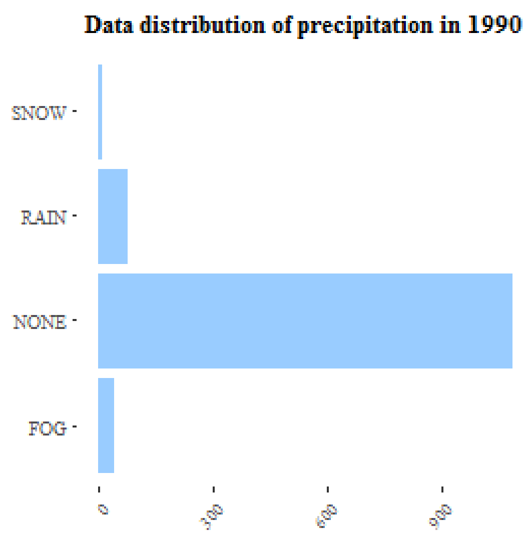

Data distribution of aircraft mass type in 1990
5
4
3
2
1
0
250
500
750
1000

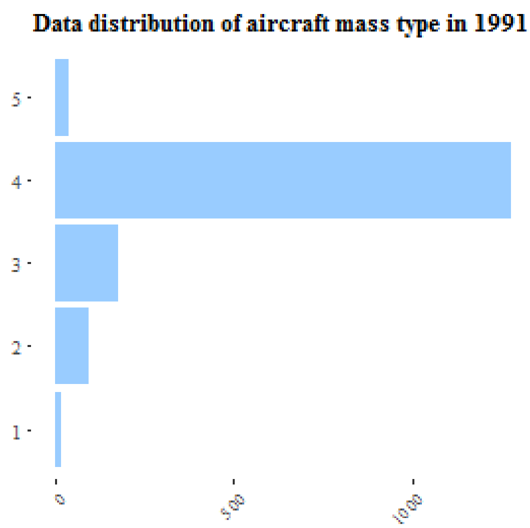

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
100
200
300
400
500

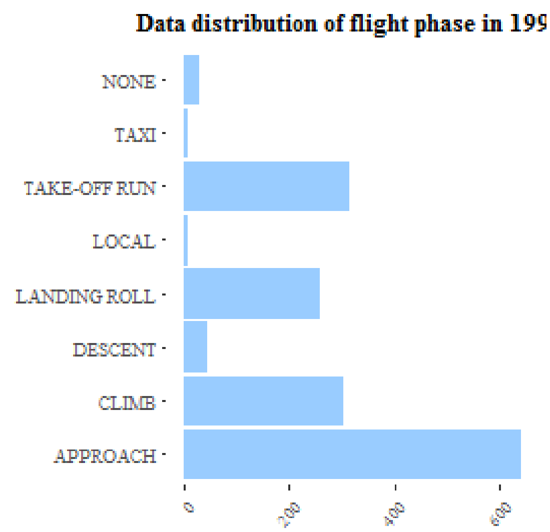

Data distribution of engine type in 1990
D
C
B
A
0
250
500
750
1000

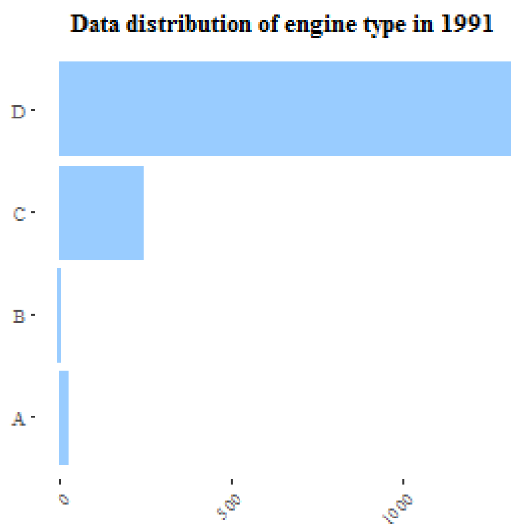

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400

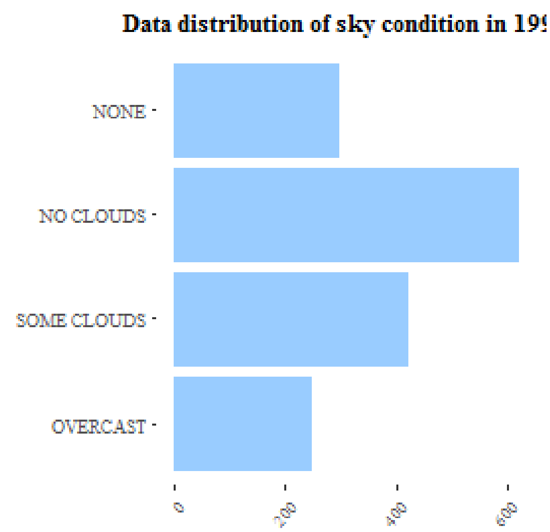

Data distribution of time of day in 1990
NONE
NIGHT
DUSK
DAY
DAWN
0
200
400
600

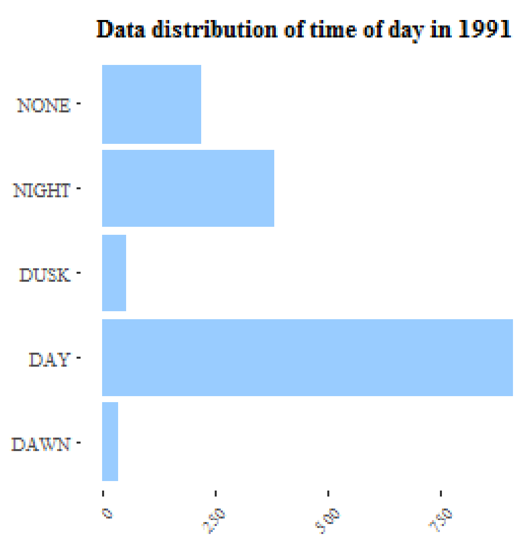

Data distribution of precipitation in 1990
SNOW
RAIN
NONE
FOG
0
300
600
900

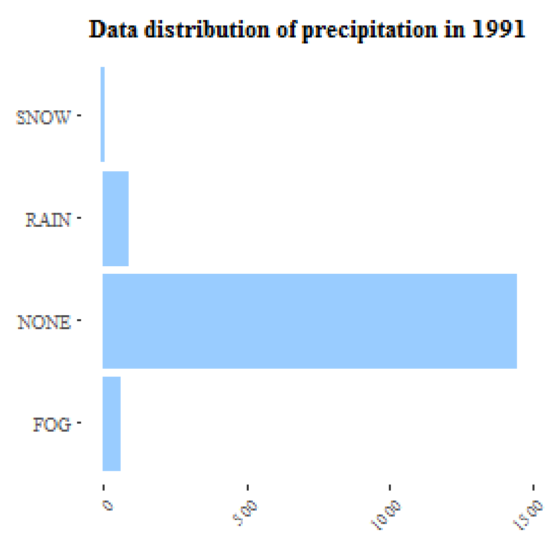

Data distribution of aircraft mass type in 1991
5
4
3
2
1
0
500
1000

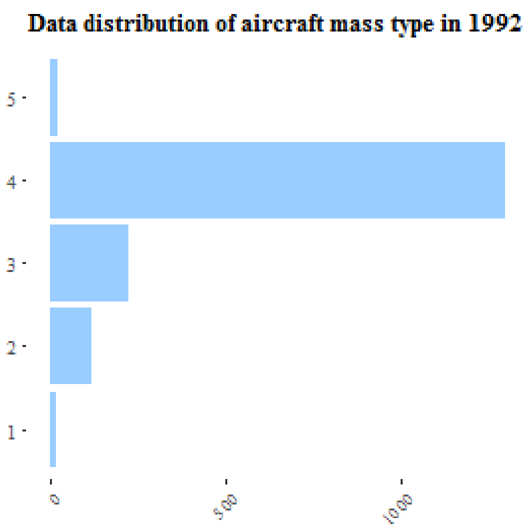

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
LOCAL
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
200
400
600

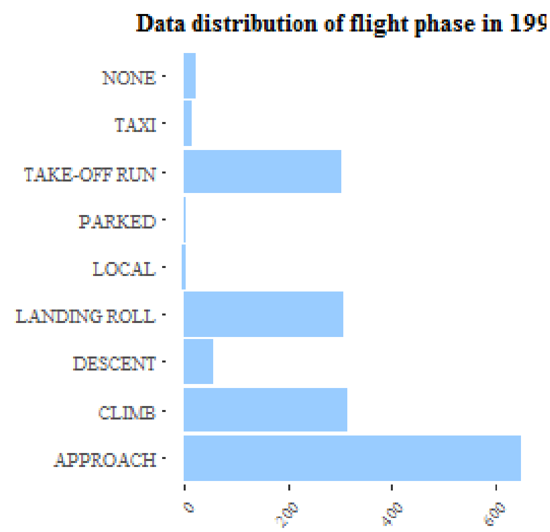

Data distribution of engine type in 1991
D
C
B
A
0
500
1000

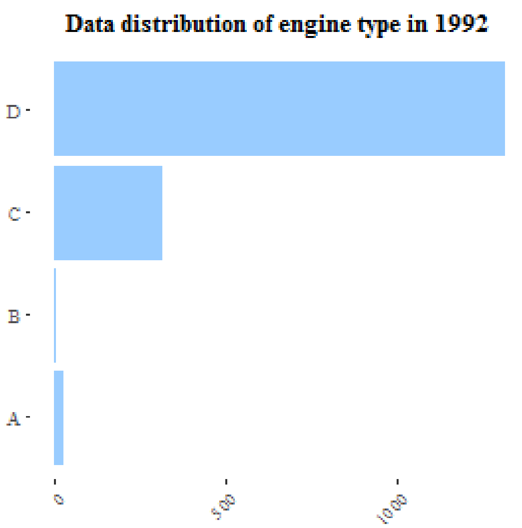

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600

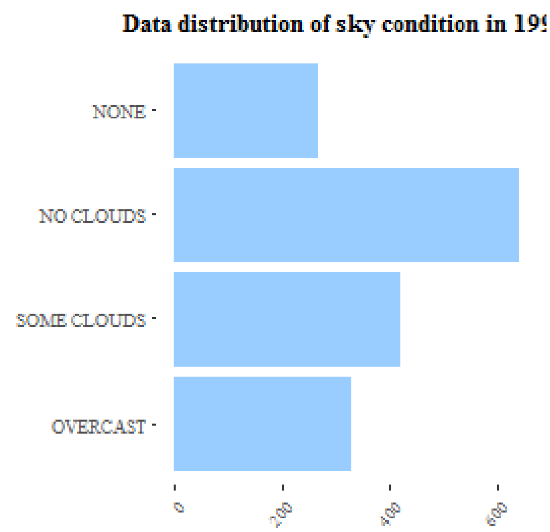

Data distribution of time of day in 1991
NONE
NIGHT
DUSK
DAY
DAWN
0
250
500
750

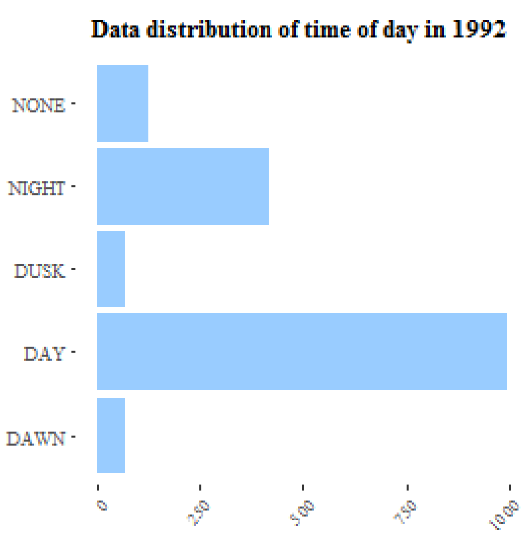

Data distribution of precipitation in 1991
SNOW
RAIN
NONE
FOG
0
500
1000
1500

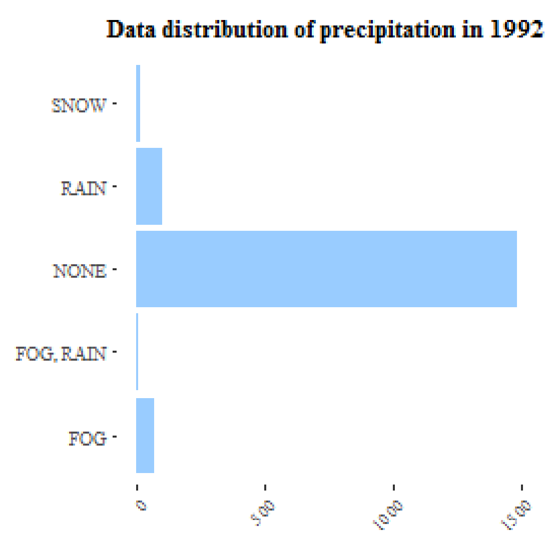

Data distribution of aircraft mass type in 1992
5
4
3
2
1
0
500
1000

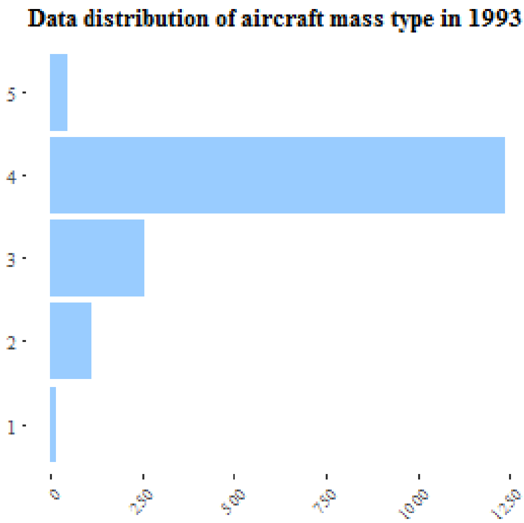

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
200
400
600

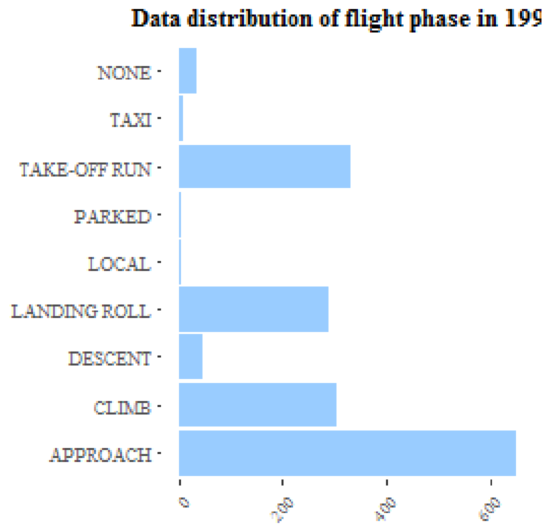

Data distribution of engine type in 1992
D
C
B
A
0
500
1000

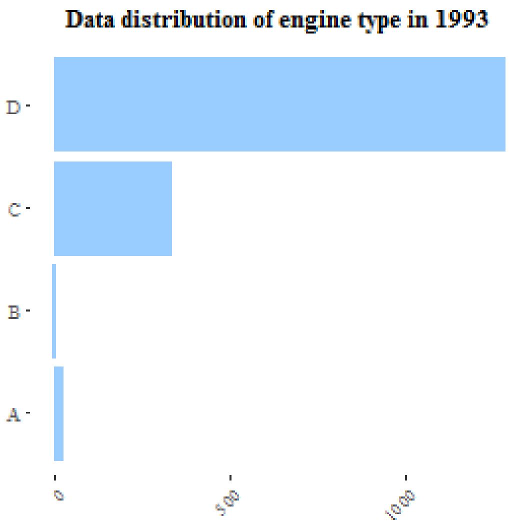

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600

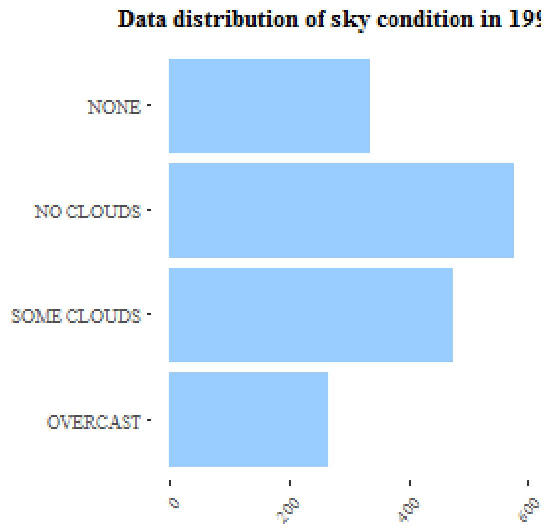

Data distribution of time of day in 1992
NONE
NIGHT
DUSK
DAY
DAWN
0
250
500
750
1000

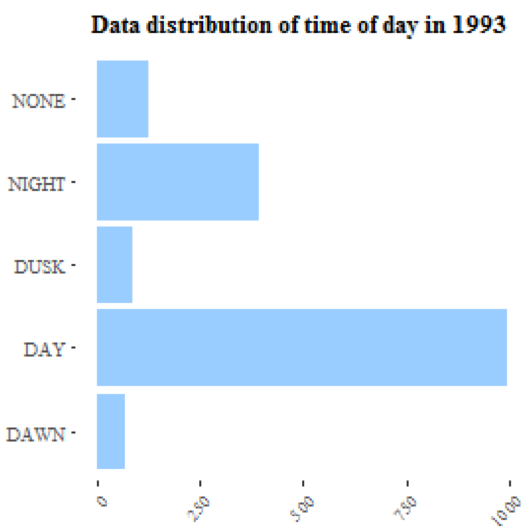

Data distribution of precipitation in 1992
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
500
1000
1500

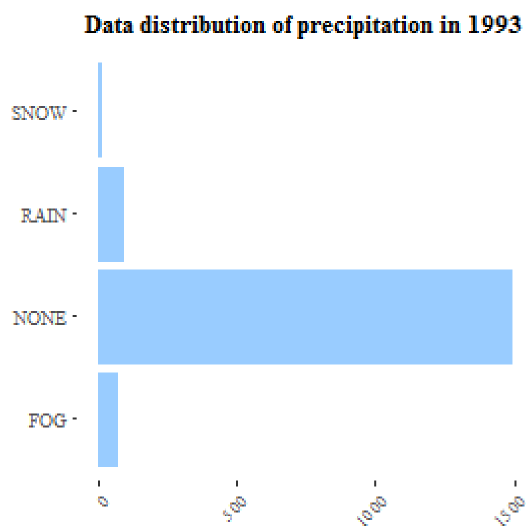

Data distribution of aircraft mass type in 1993
5
4
3
2
1
0
250
500
750
1000
1250

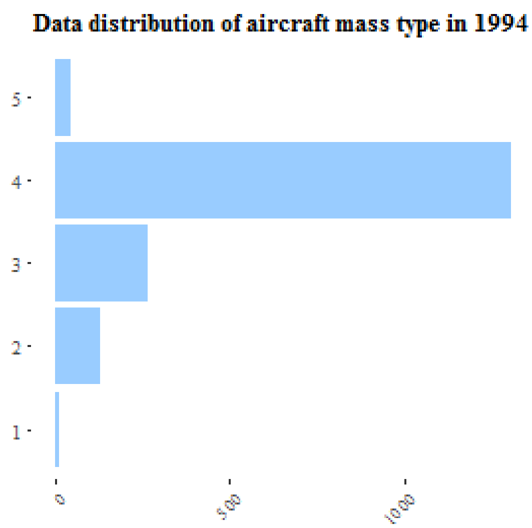

NONE
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
200
400
600

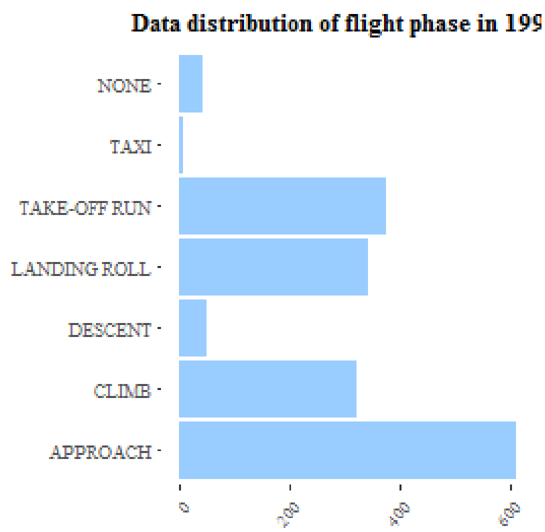

Data distribution of engine type in 1993
D
C
B
A
0
500
1000

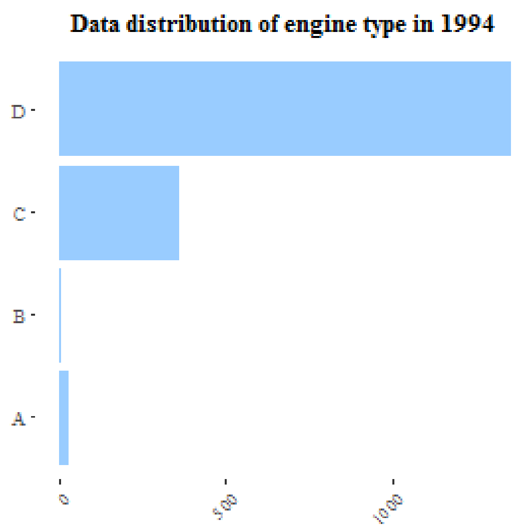

NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600

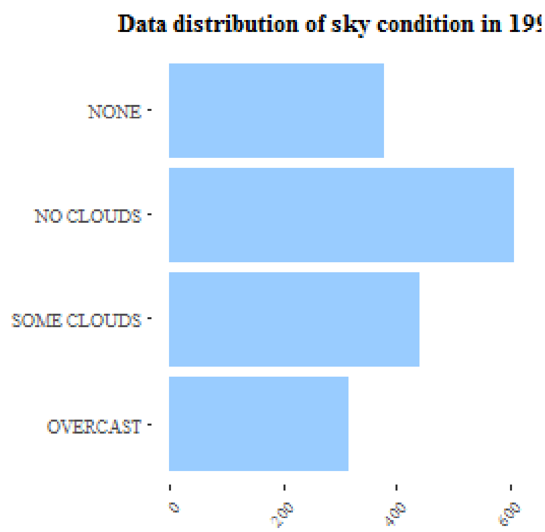

Data distribution of time of day in 1993
NONE
NIGHT
DUSK
DAY
DAWN
0
250
500
750
1000

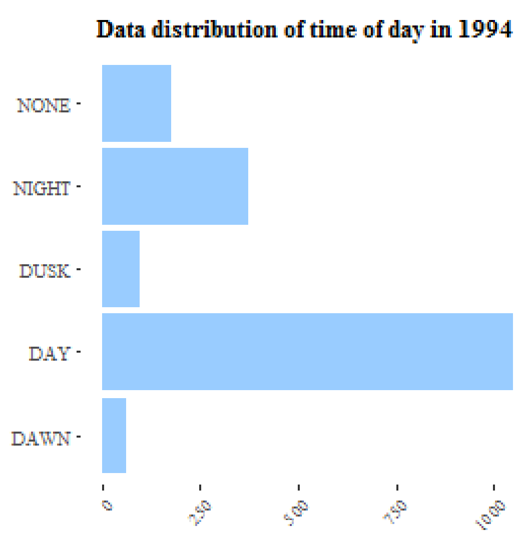

Data distribution of precipitation in 1993
SNOW
RAIN
NONE
FOG
0
500
1000
1500

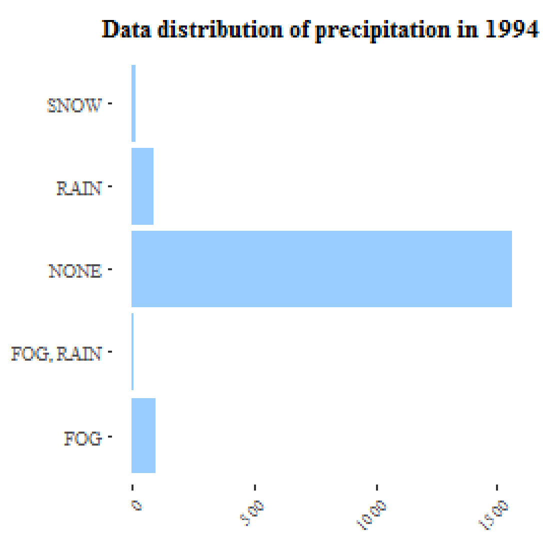

Data distribution of aircraft mass type in 1994
5
4
3
2
1
0
500
1000

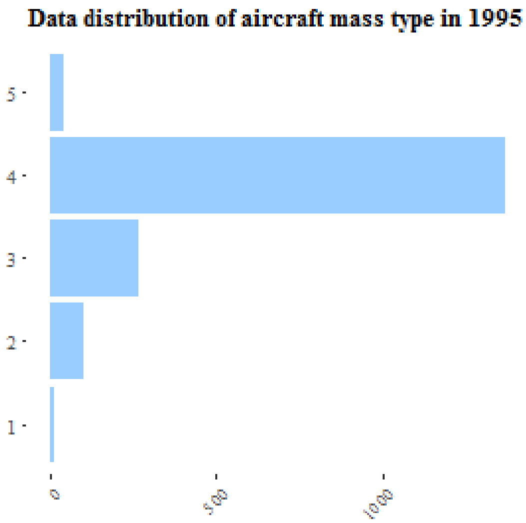

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
200
400
600

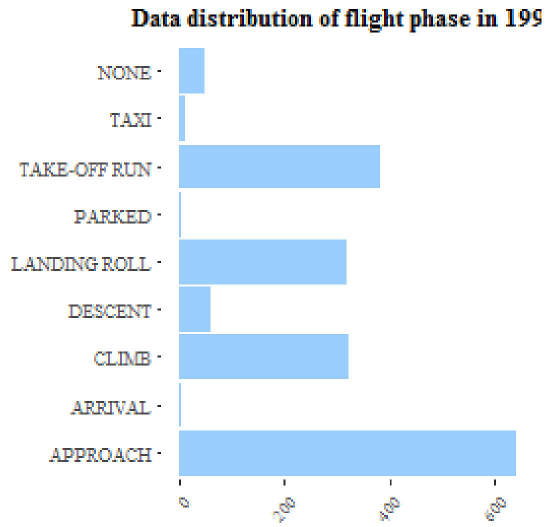

Data distribution of engine type in 1994
D
C
B
A
0
500
1000

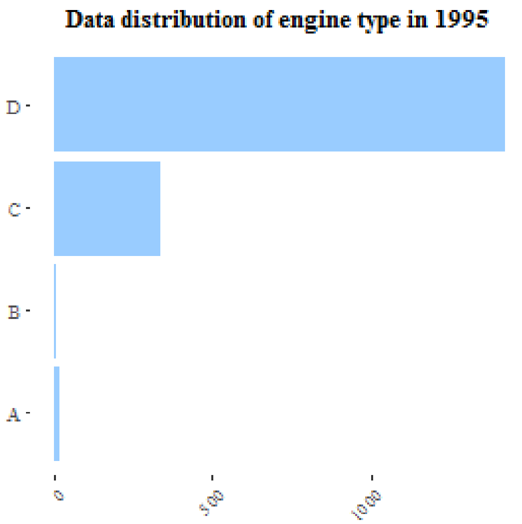

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600

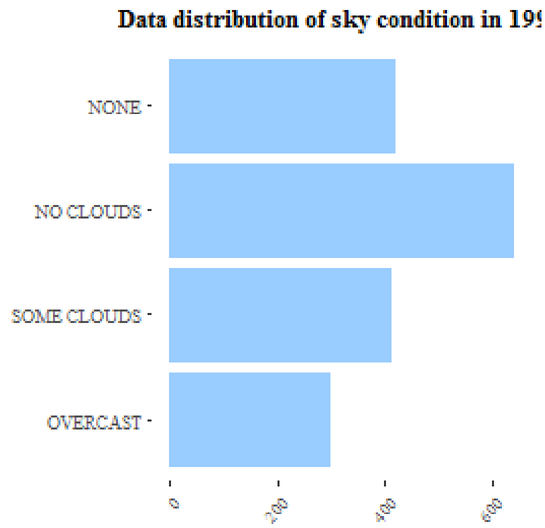

Data distribution of time of day in 1994
NONE
NIGHT
DUSK
DAY
DAWN
0
250
500
750
1000

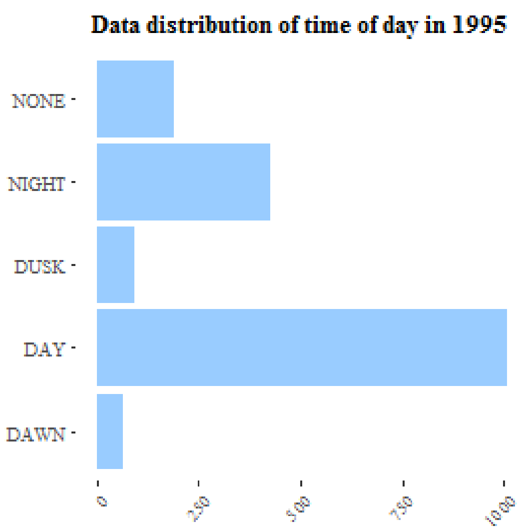

Data distribution of precipitation in 1994
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
500
1000
1500

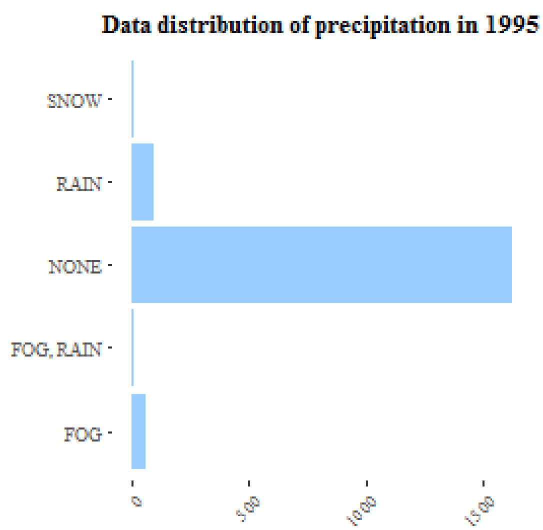

Data distribution of aircraft mass type in 1995
5
4
3
2
1
0
500
1000

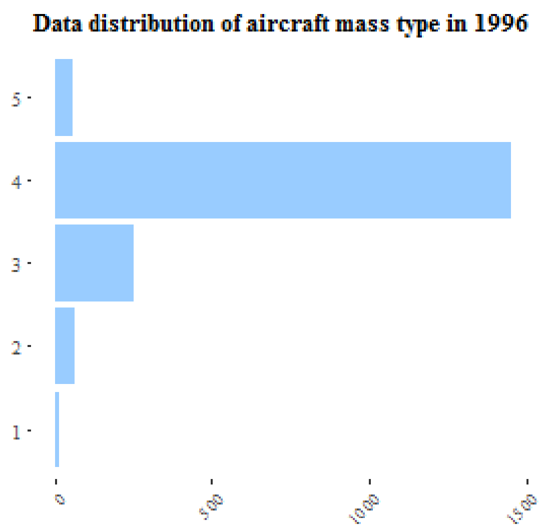

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
PARKED
LANDING ROLL
DESCENT
CLIMB
ARRIVAL
APPROACH
0
200
400
600


Data distribution of engine type in 1995
D
C
B
A
0
500
1000

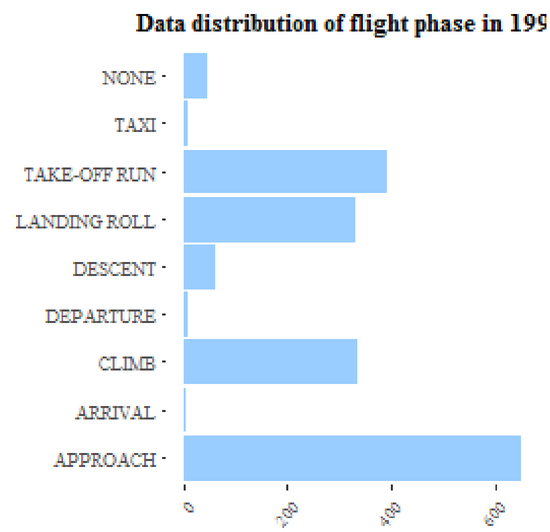

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600


Data distribution of time of day in 1995
NONE
NIGHT
DUSK
DAY
DAWN
0
250
500
750
1000

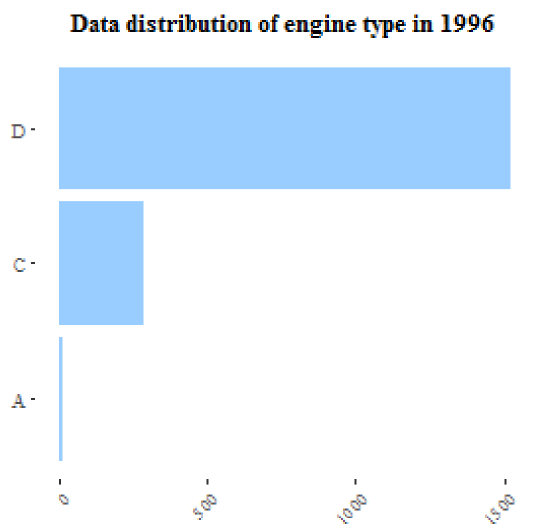

Data distribution of precipitation in 1995
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
500
1000
1500



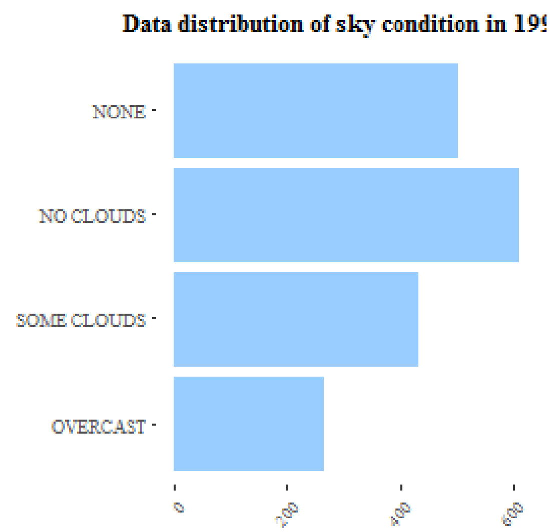
Data distribution of aircraft mass type in 1996
5
4
3
2
1
0
500
1000
1500



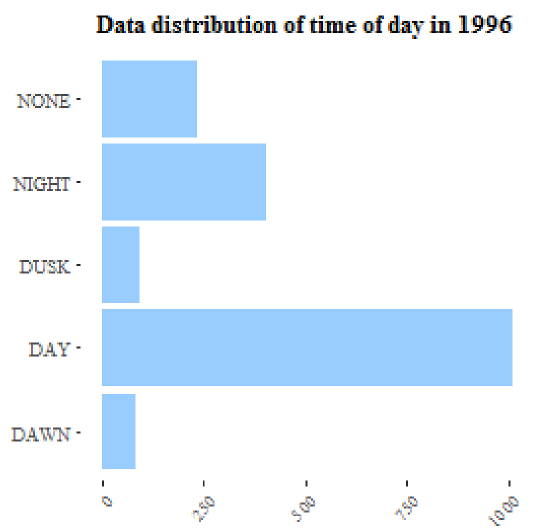
Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
200
400
600



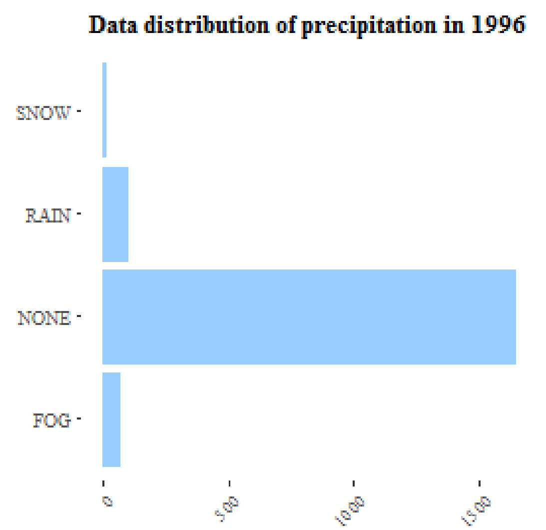
Data distribution of engine type in 1996
D
C
A
0
500
1000
1500



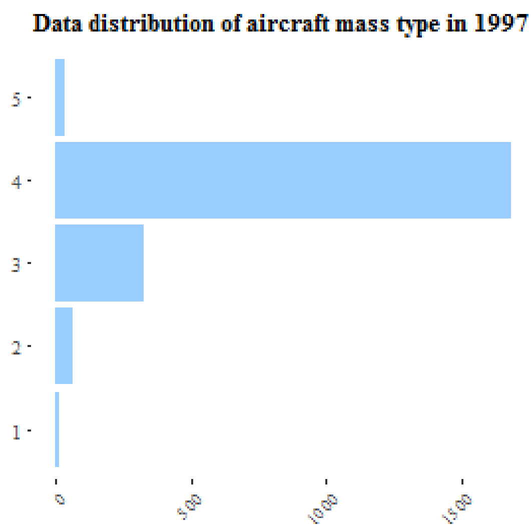
Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600

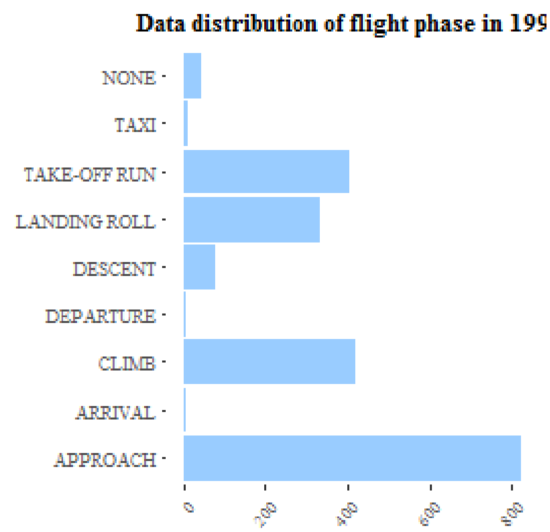


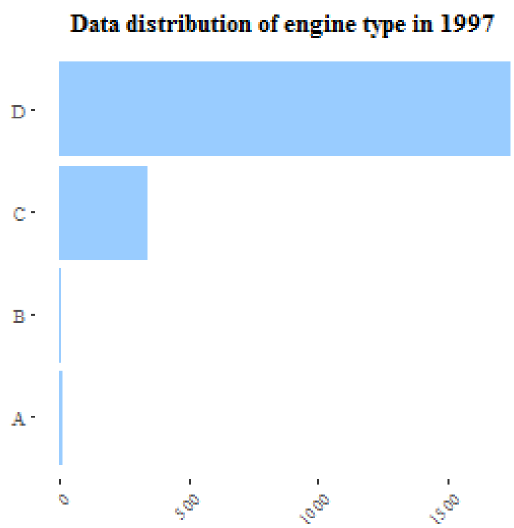
Data distribution of time of day in 1996
NONE
NIGHT
DUSK
DAY
DAWN
0
250
500
750
1000

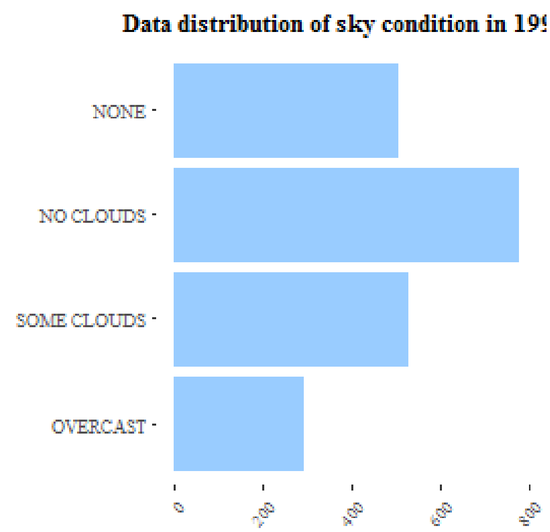


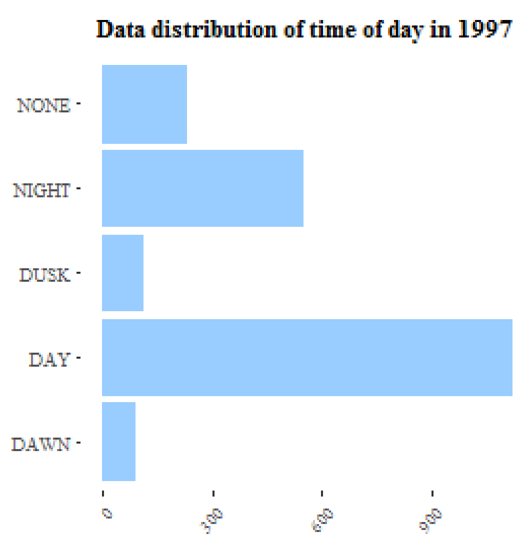
Data distribution of precipitation in 1996
SNOW
RAIN
NONE
FOG
0
500
1000
1500

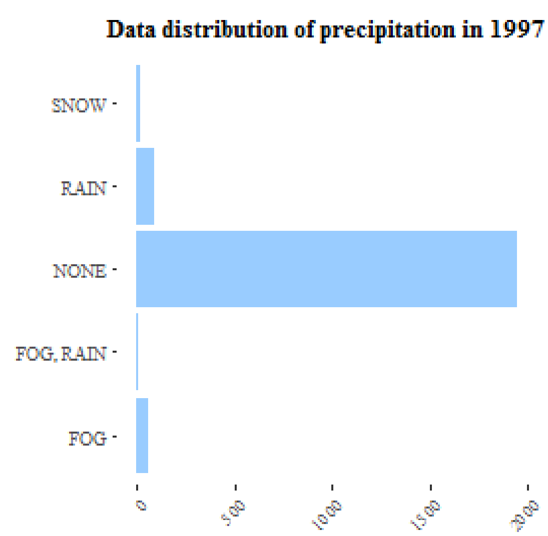

Data distribution of aircraft mass type in 1997
5
4
3
2
1
0
500
1000
1500

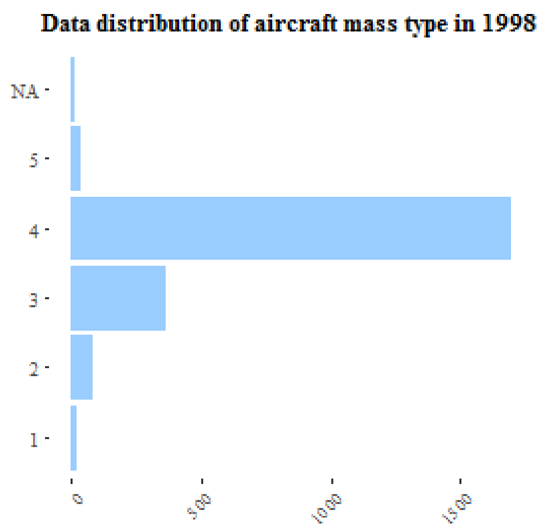

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
200
400
600
800

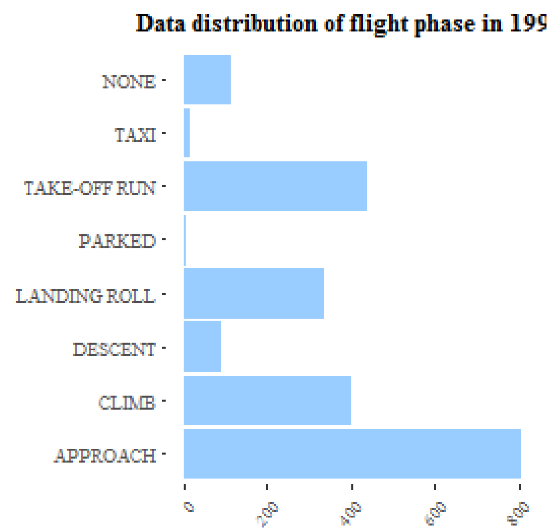

Data distribution of engine type in 1997
D
C
B
A
0
500
1000
1500

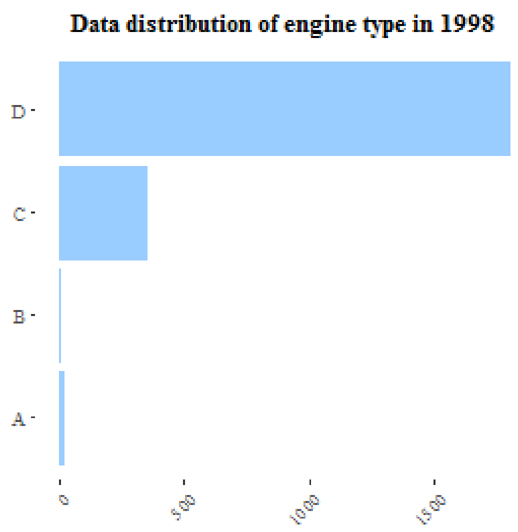

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600
800

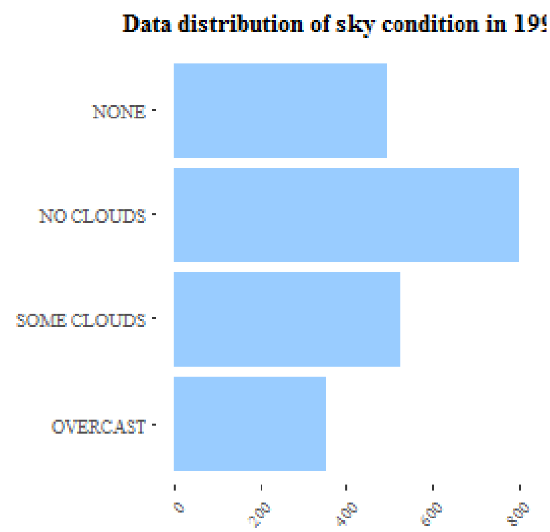

Data distribution of time of day in 1997
NONE
NIGHT
DUSK
DAY
DAWN
0
300
600
900

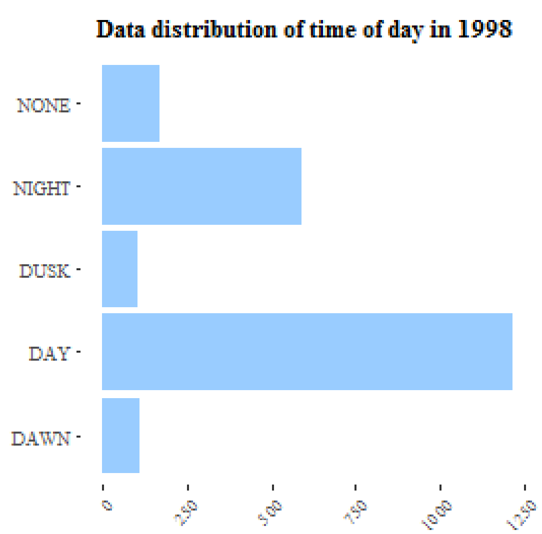

Data distribution of precipitation in 1997
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
500
1000
1500
2000

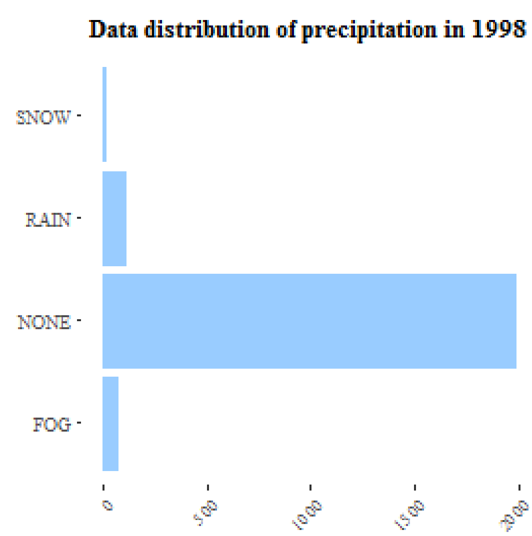

Data distribution of aircraft mass type in 1998
NA
5
4
3
2
1
0
500
1000
1500

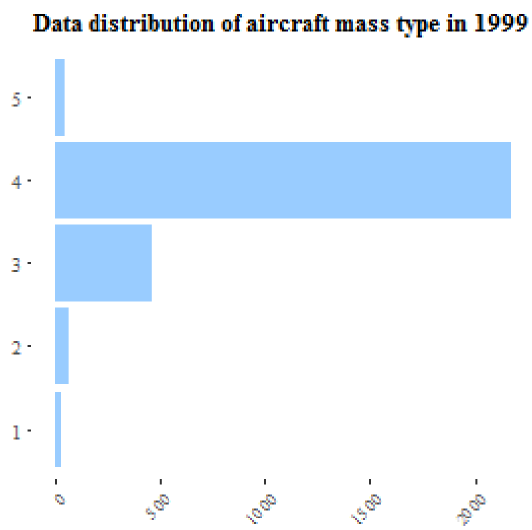

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
PARKED
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
200
400
600
800

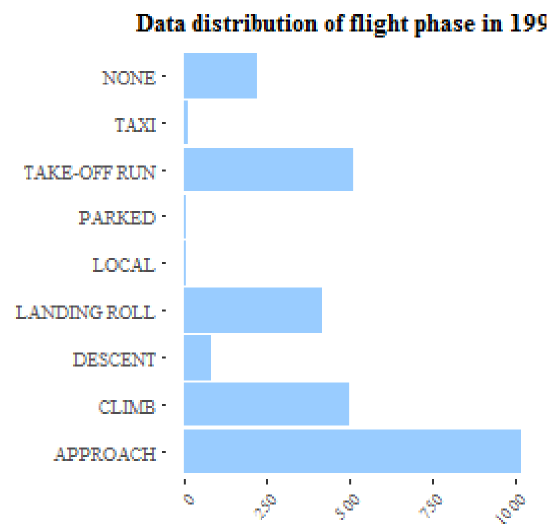

Data distribution of engine type in 1998
D
C
B
A
0
500
1000
1500

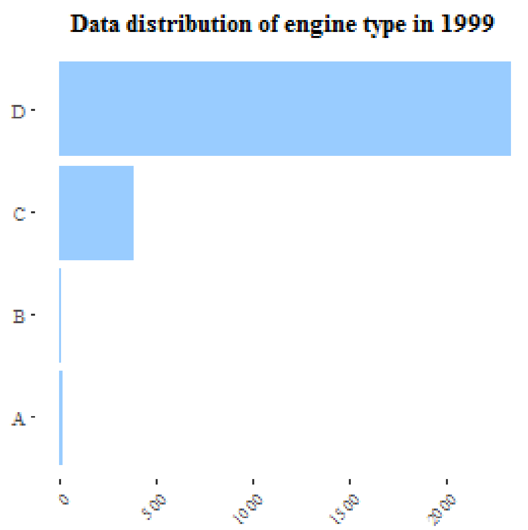

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
200
400
600
800

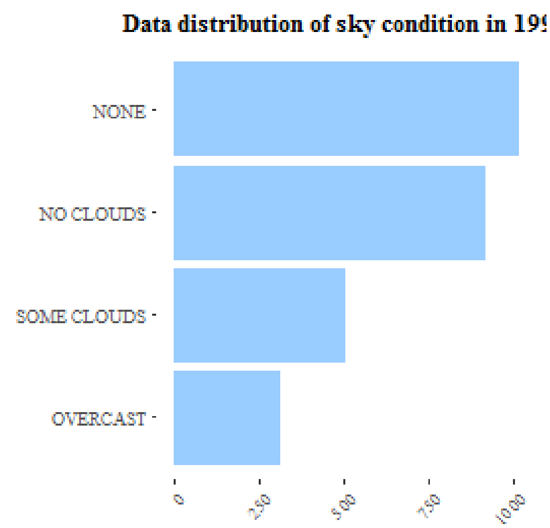

Data distribution of time of day in 1998
NONE
NIGHT
DUSK
DAY
DAWN
0
250
500
750
1000
1250

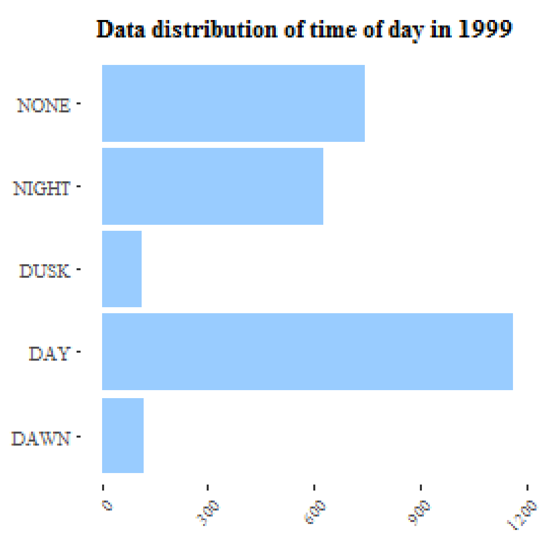

Data distribution of precipitation in 1998
SNOW
RAIN
NONE
FOG
0
500
1000
1500
2000

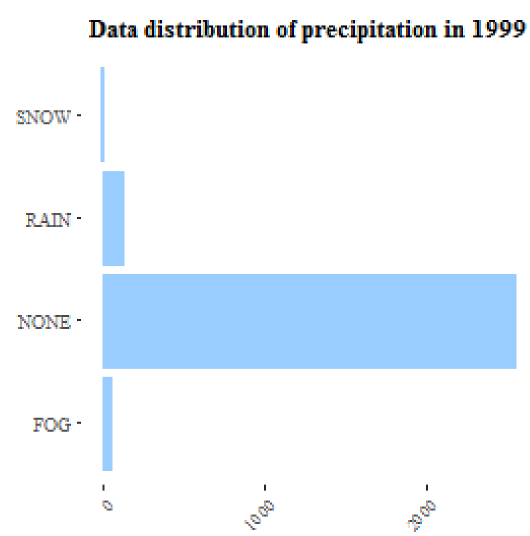

Data distribution of aircraft mass type in 1999
5
4
3
2
1
0
500
1000
1500
2000

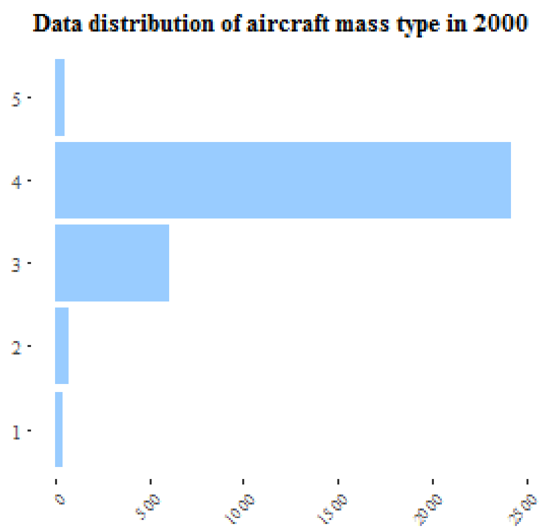

Data distribution of flight phase in 199
NONE
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
CLIMB
APPROACH
0
250
500
750
1000

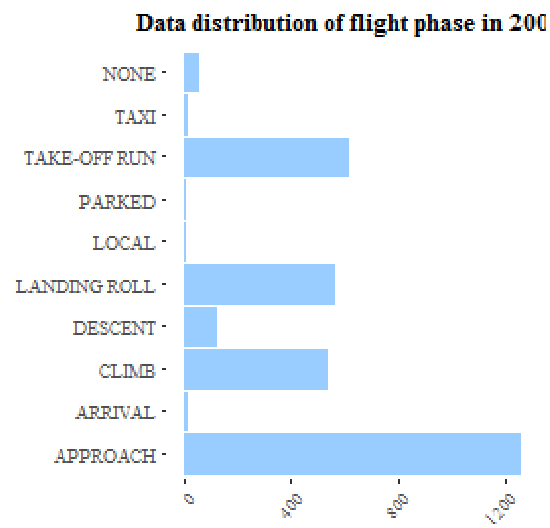

Data distribution of engine type in 1999
D
C
B
A
0
500
1000
1500
2000

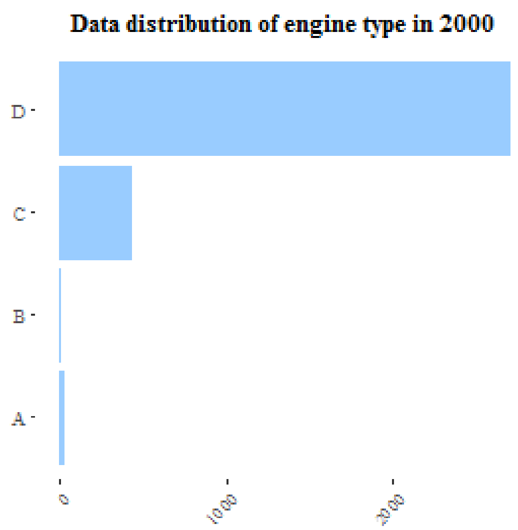

Data distribution of sky condition in 19
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
250
500
750
1000

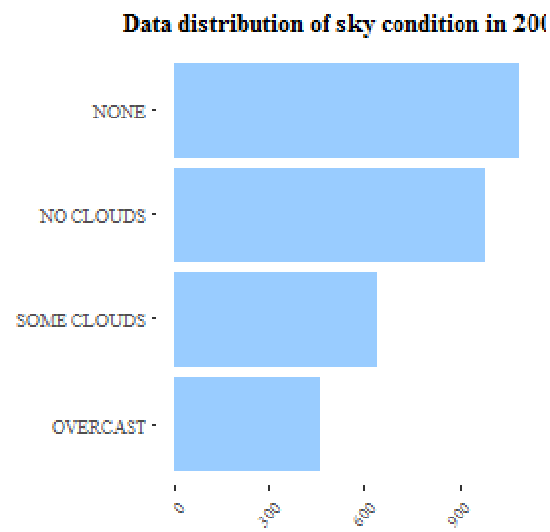

Data distribution of time of day in 1999
NONE
NIGHT
DUSK
DAY
DAWN
0
300
600
900
1200

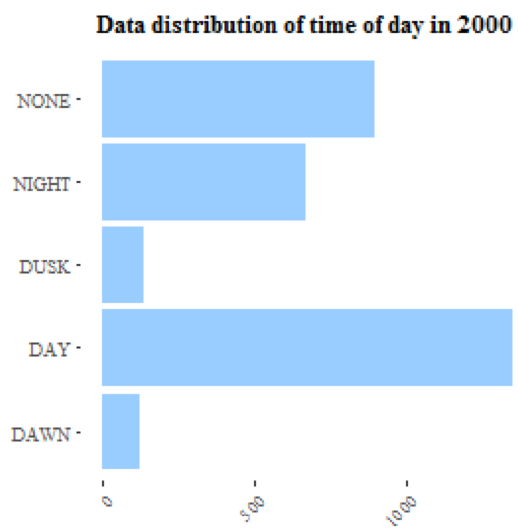

Data distribution of precipitation in 1999
SNOW
RAIN
NONE
FOG
0
1000
2000

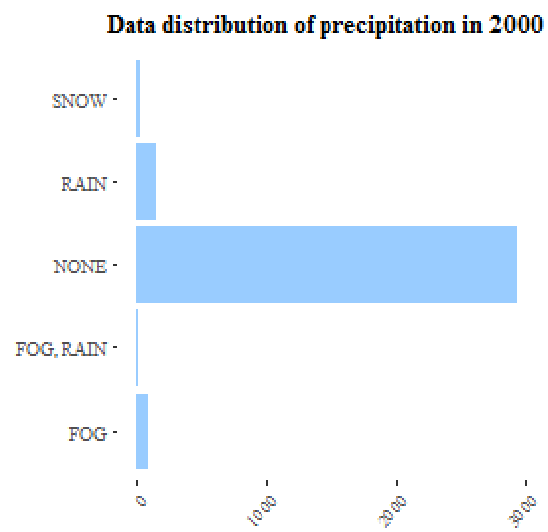

Data distribution of aircraft mass type in 2000
5
4
3
2
1
0
500
1000
1500
2000
2500

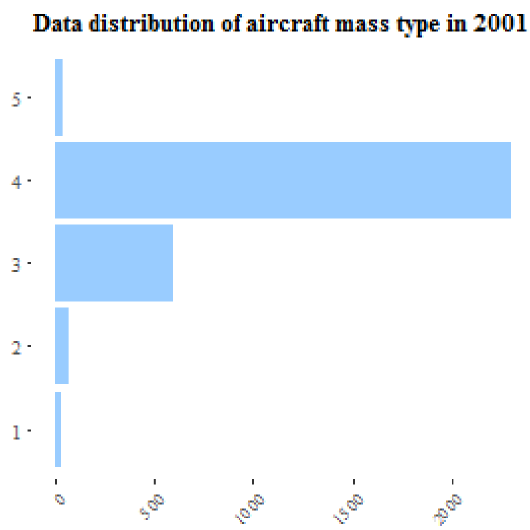

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
CLIMB
ARRIVAL
APPROACH
0
400
800
1200

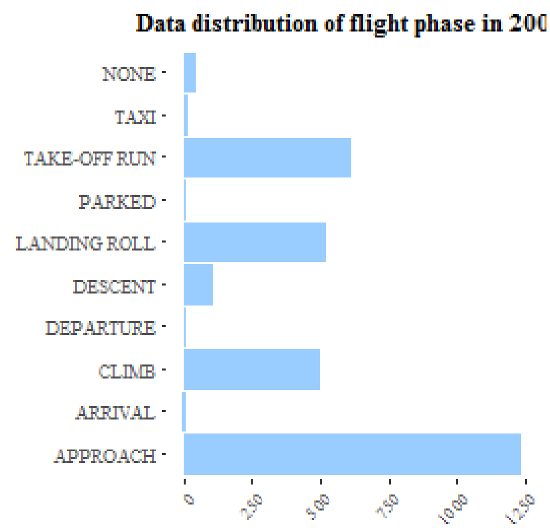

Data distribution of engine type in 2000
D
C
B
A
0
1000
2000

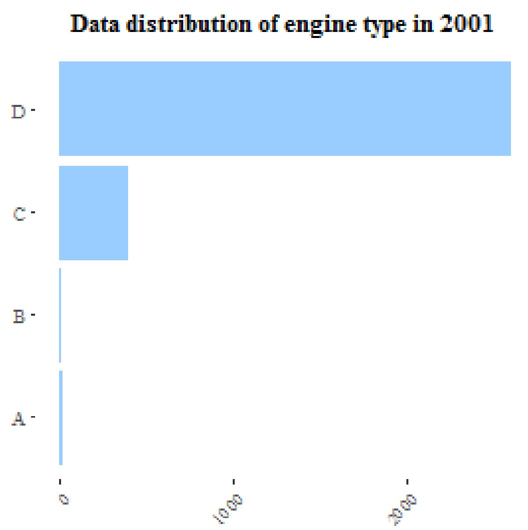

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
300
600
900

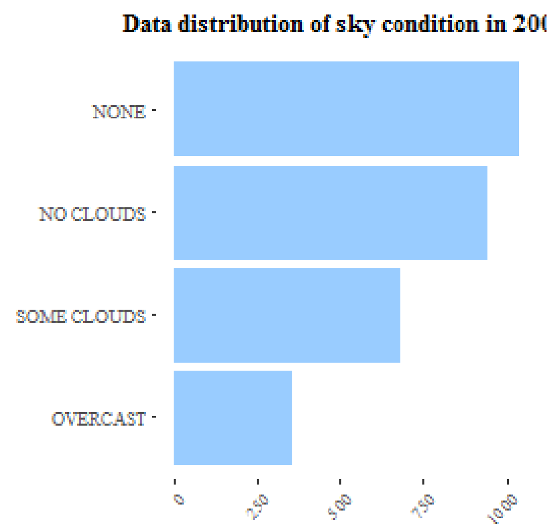

Data distribution of time of day in 2000
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000

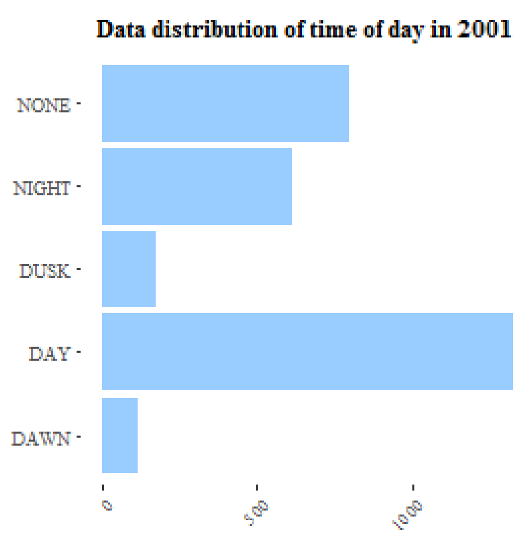

Data distribution of precipitation in 2000
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
1000
2000
3000

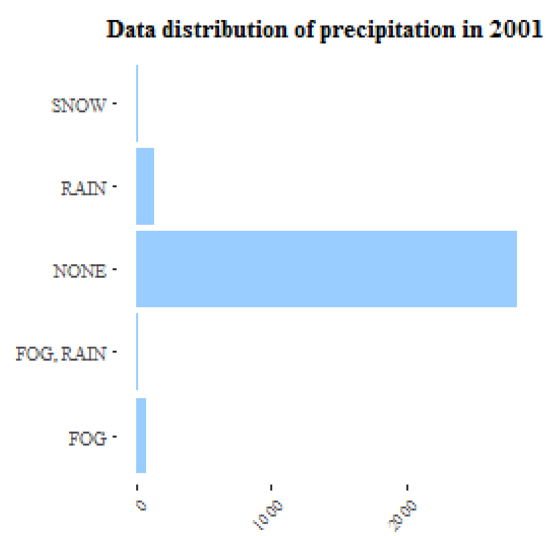

Data distribution of aircraft mass type in 2001
5
4
3
2
1
0
500
1000
1500
2000

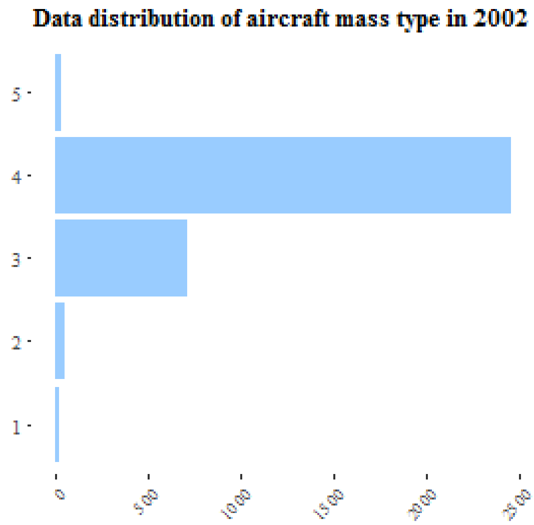

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
PARKED
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
250
500
750
1000
1250

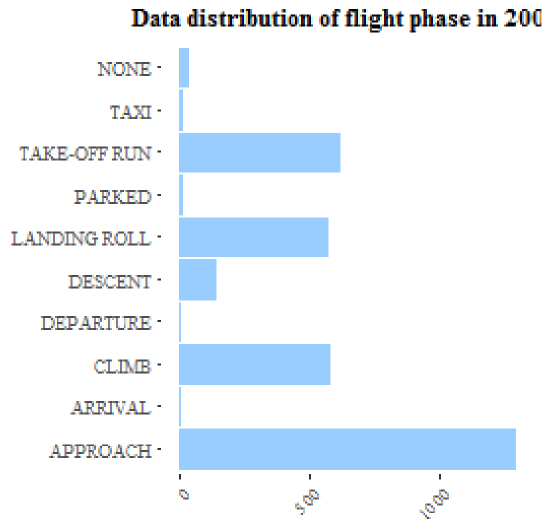

Data distribution of engine type in 2001
D
C
B
A
0
1000
2000

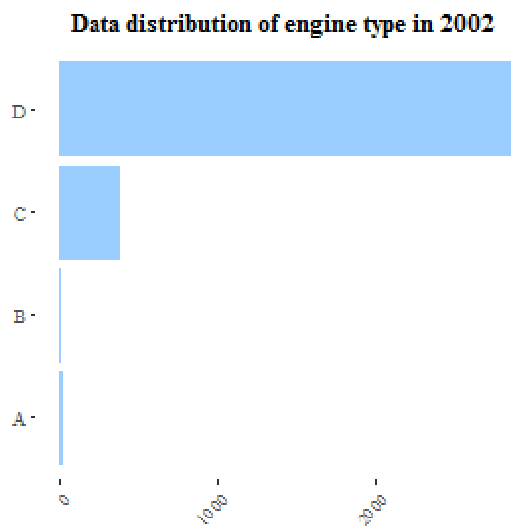

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
250
500
750
1000

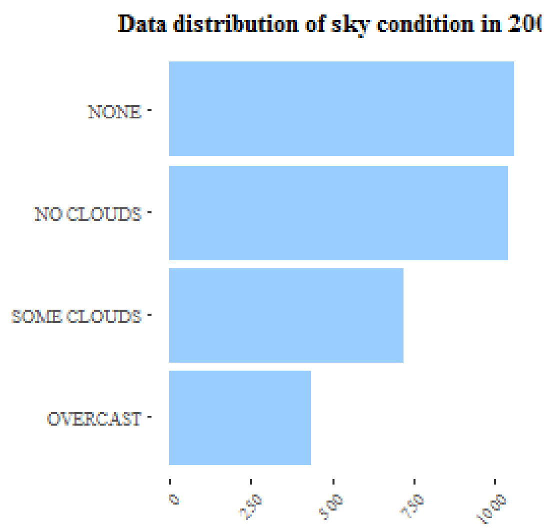

Data distribution of time of day in 2001
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000

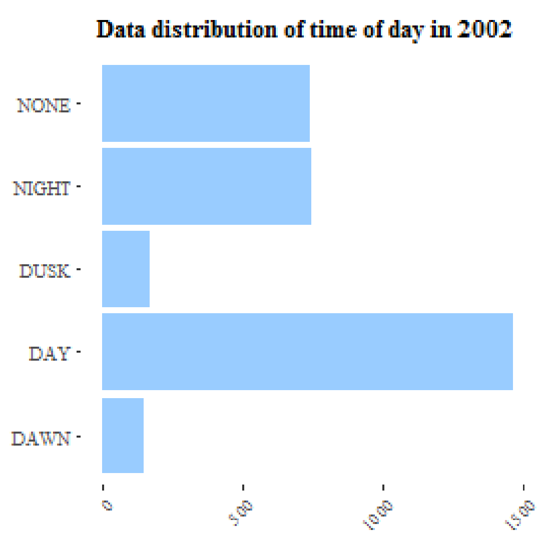

Data distribution of precipitation in 2001
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
1000
2000

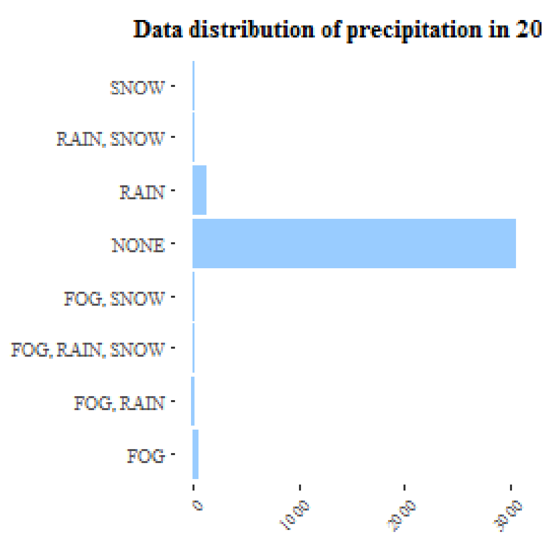

Data distribution of aircraft mass type in 2002
5
4
3
2
1
0
500
1000
1500
2000
2500

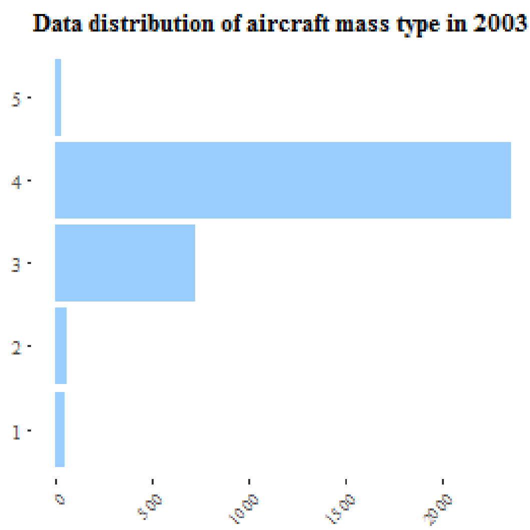

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
PARKED
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000

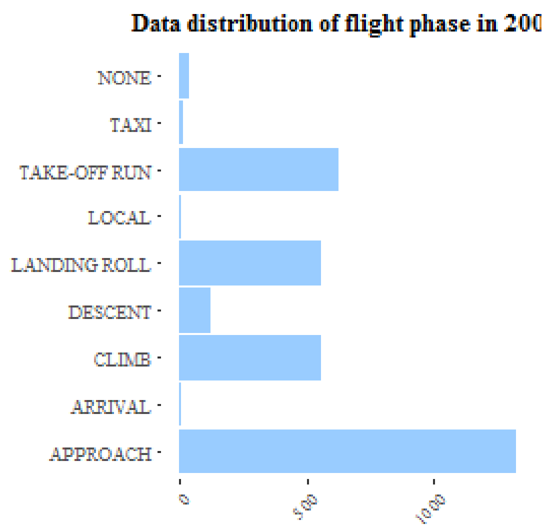

Data distribution of engine type in 2002
D
C
B
A
0
1000
2000

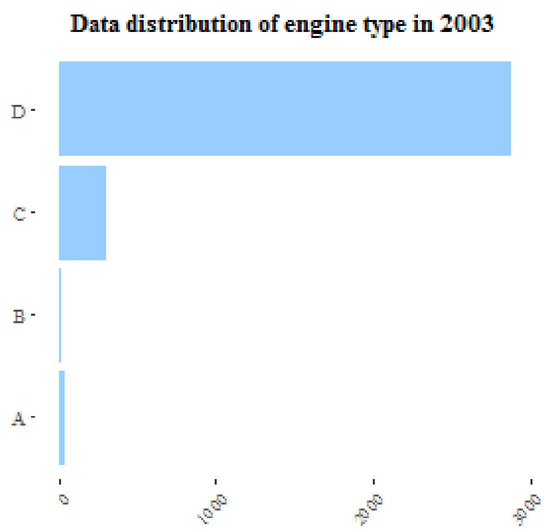

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
250
500
750
1000

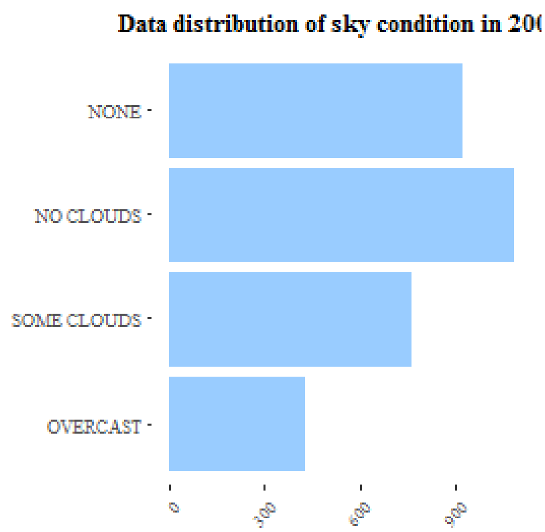

Data distribution of time of day in 2002
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000
1500

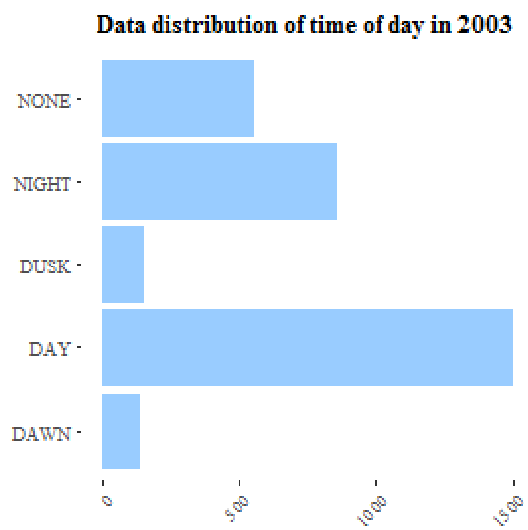

Data distribution of precipitation in 20
SNOW
RAIN, SNOW
RAIN
NONE
FOG, SNOW
FOG, RAIN, SNOW
FOG, RAIN
FOG
0
1000
2000
3000

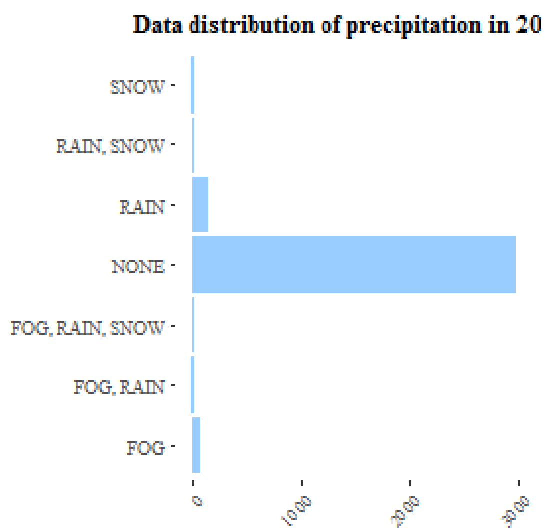

Data distribution of aircraft mass type in 2003
5
4
3
2
1
0
500
1000
1500
2000

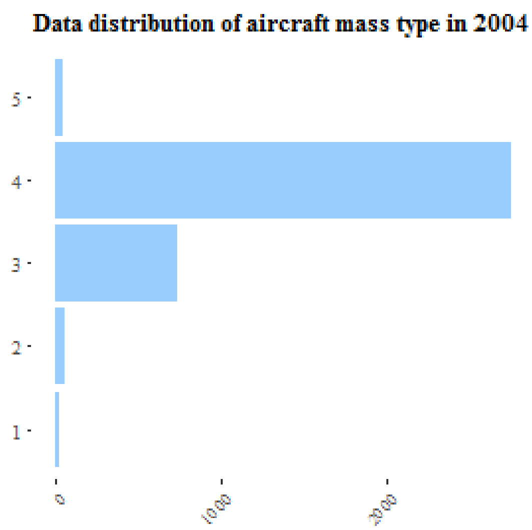

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
LOCAL
LANDING ROLL
DESCENT
CLIMB
ARRIVAL
APPROACH
0
500
1000

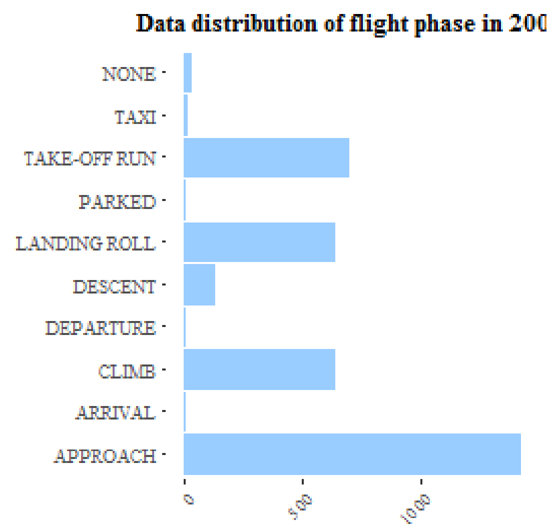

Data distribution of engine type in 2003
D
C
B
A
0
1000
2000
3000

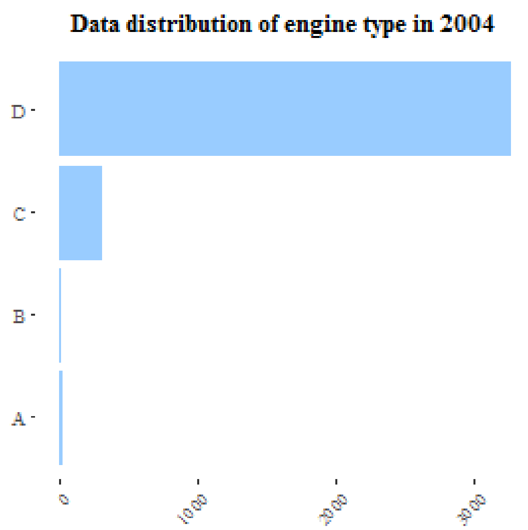

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
300
600
900

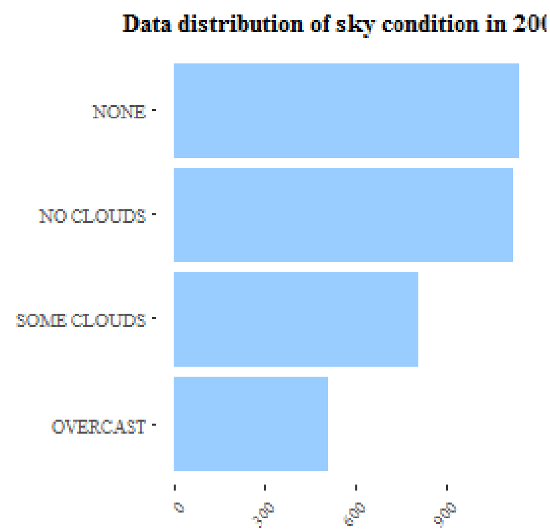

Data distribution of time of day in 2003
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000
1500

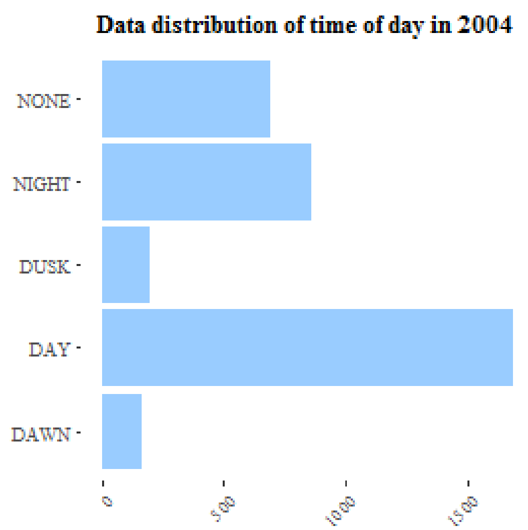

Data distribution of precipitation in 20
SNOW
RAIN, SNOW
RAIN
NONE
FOG, RAIN, SNOW
FOG, RAIN
FOG
0
1000
2000
3000

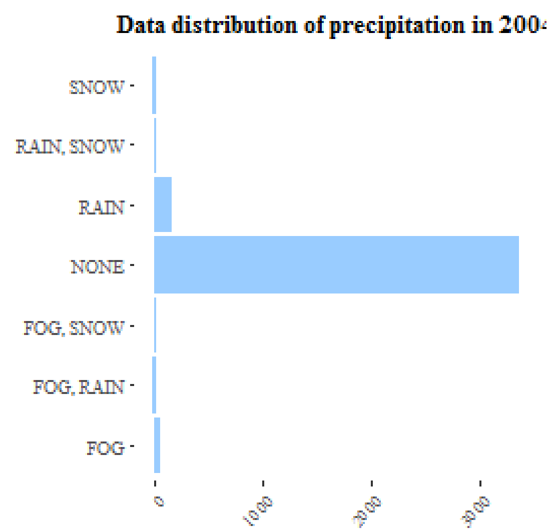

Data distribution of aircraft mass type in 2004
5
4
3
2
1
0
1000
2000

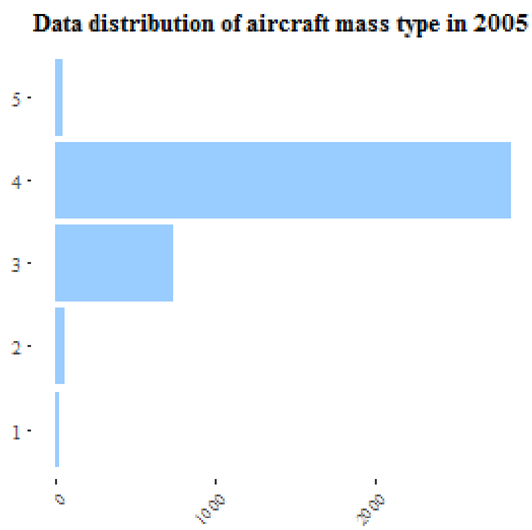

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
PARKED
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000

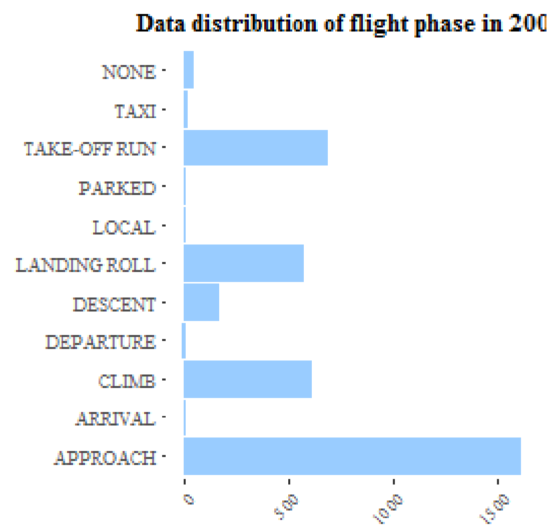

Data distribution of engine type in 2004
D
C
B
A
0
1000
2000
3000

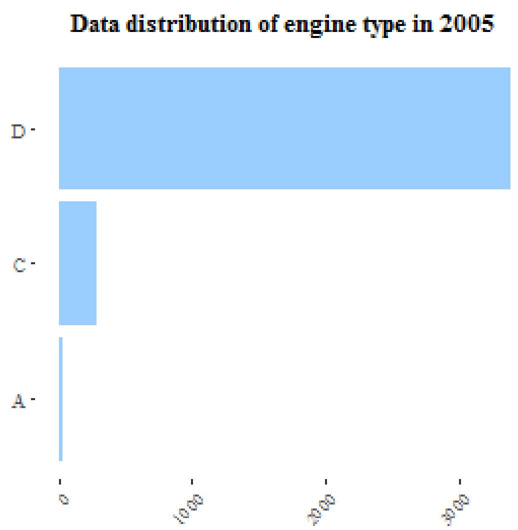

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
300
600
900

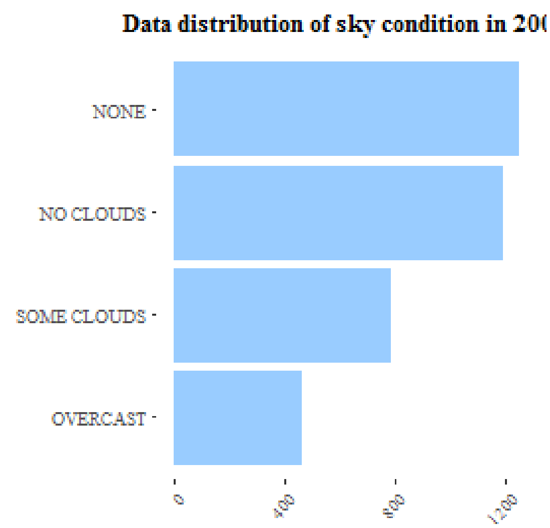

Data distribution of time of day in 2004
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000
1500

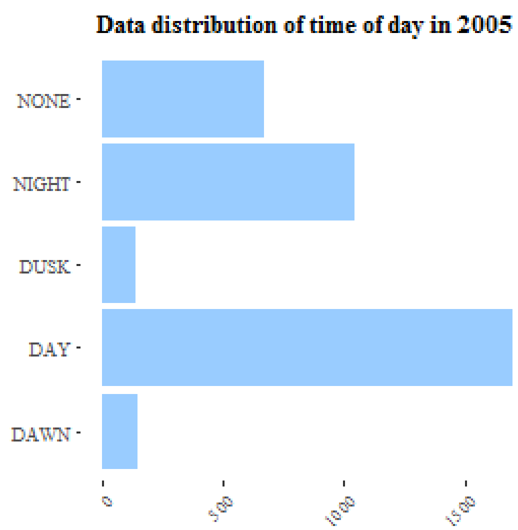

Data distribution of precipitation in 200
SNOW
RAIN, SNOW
RAIN
NONE
FOG, SNOW
FOG, RAIN
FOG
0
1000
2000
3000

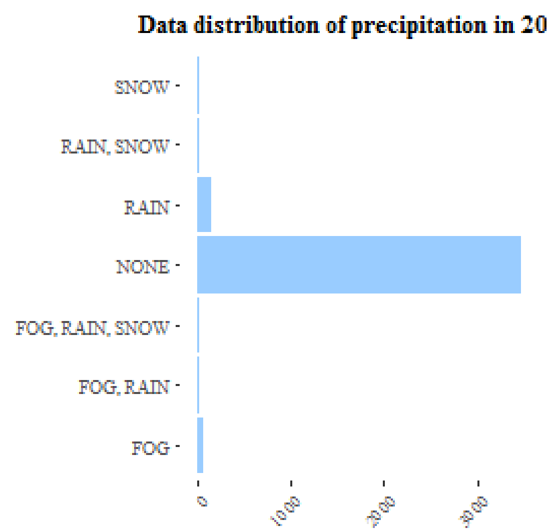

Data distribution of aircraft mass type in 2005
5
4
3
2
1
0
1000
2000

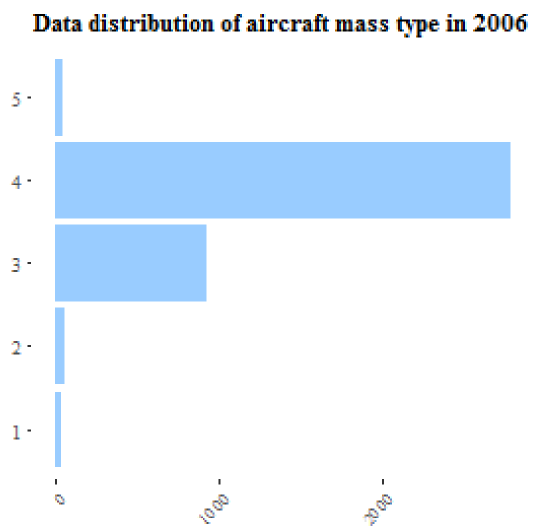

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000
1500

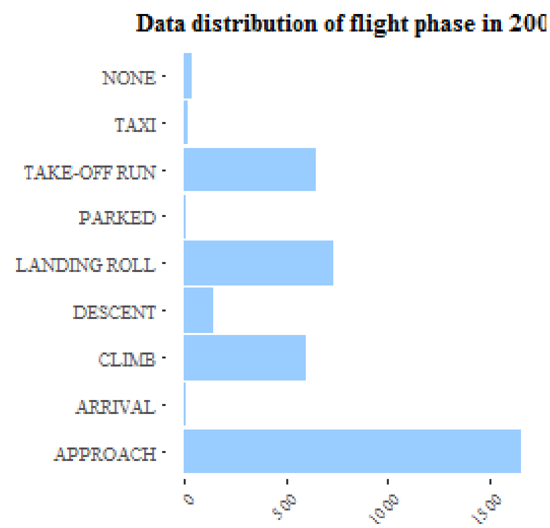

Data distribution of engine type in 2005
D
C
A
0
1000
2000
3000

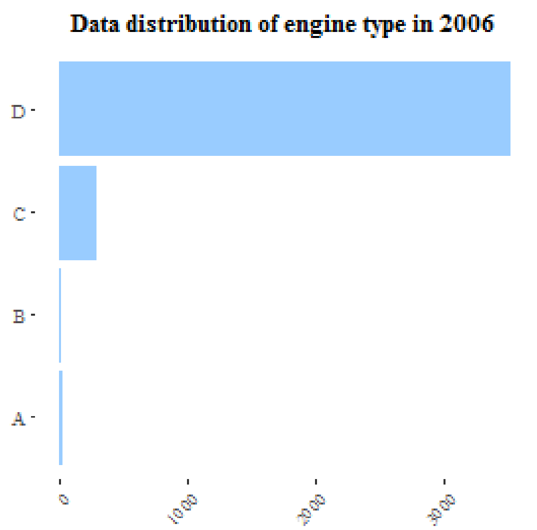

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
400
800
1200

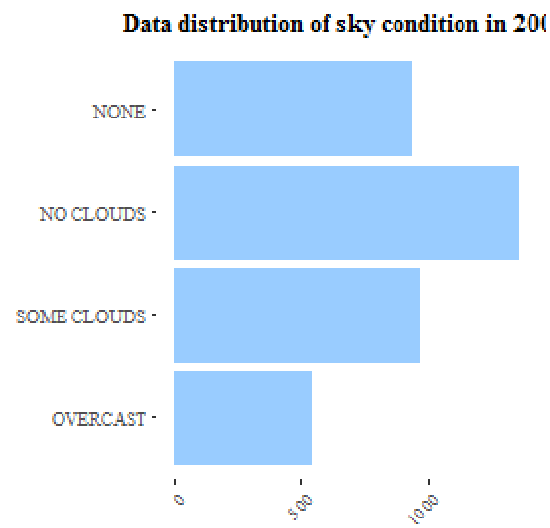

Data distribution of time of day in 2005
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000
1500

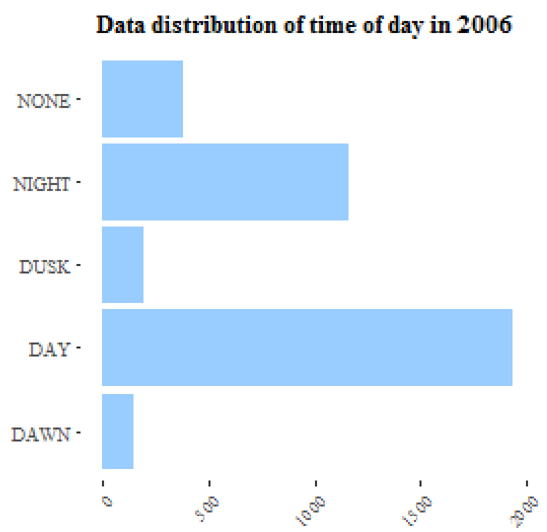

Data distribution of precipitation in 20
SNOW
RAIN, SNOW
RAIN
NONE
FOG, RAIN, SNOW
FOG, RAIN
FOG
0
1000
2000
3000

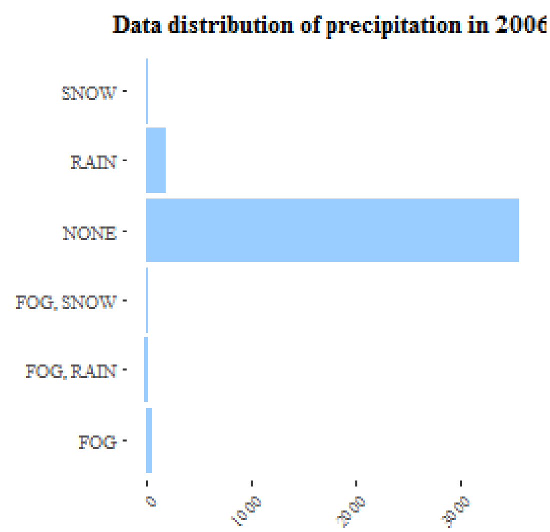

Data distribution of aircraft mass type in 2006
5
4
3
2
1
0
1000
2000

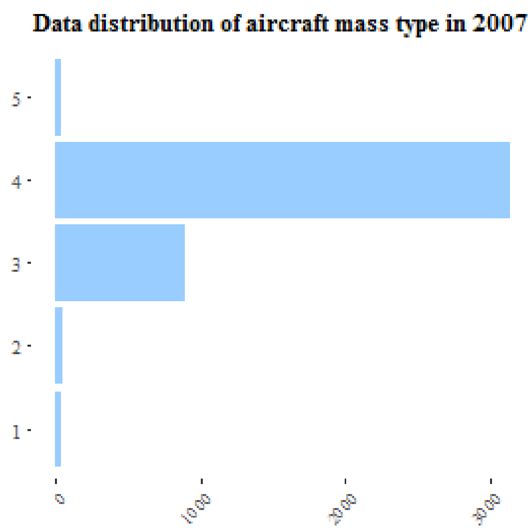

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
PARKED
LANDING ROLL
DESCENT
CLIMB
ARRIVAL
APPROACH
0
500
1000
1500

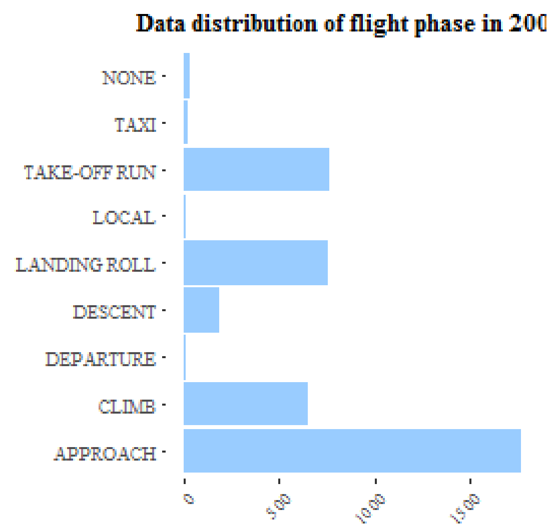

Data distribution of engine type in 2006
D
C
B
A
0
1000
2000
3000

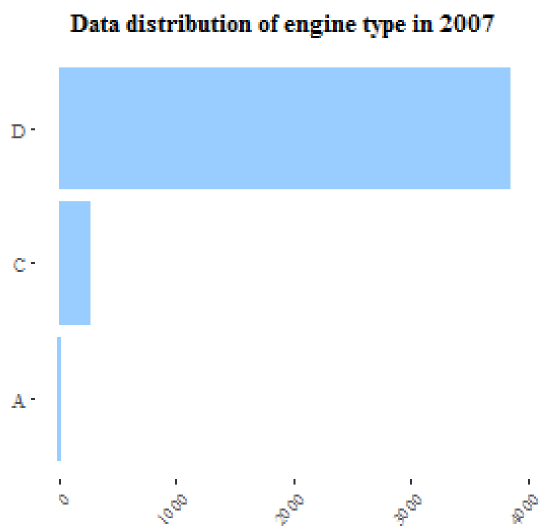

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000

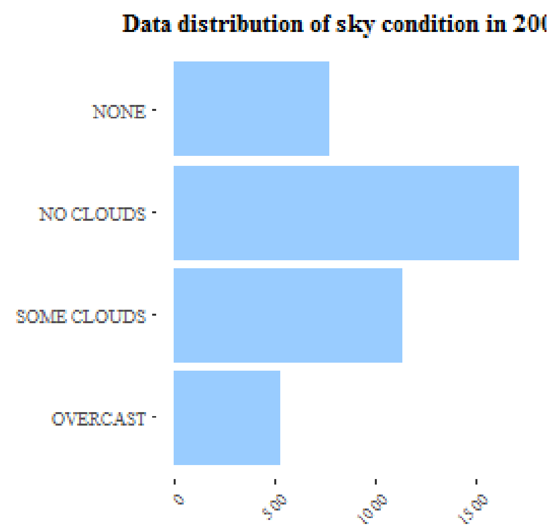

Data distribution of time of day in 2006
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000
1500
2000

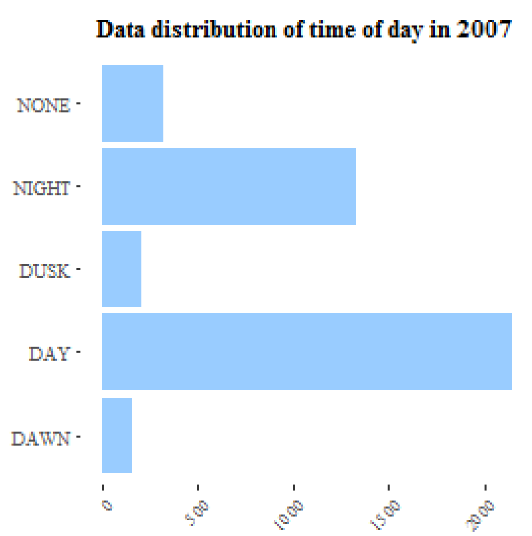

Data distribution of precipitation in 2006
SNOW
RAIN
NONE
FOG, SNOW
FOG, RAIN
FOG
0
1000
2000
3000

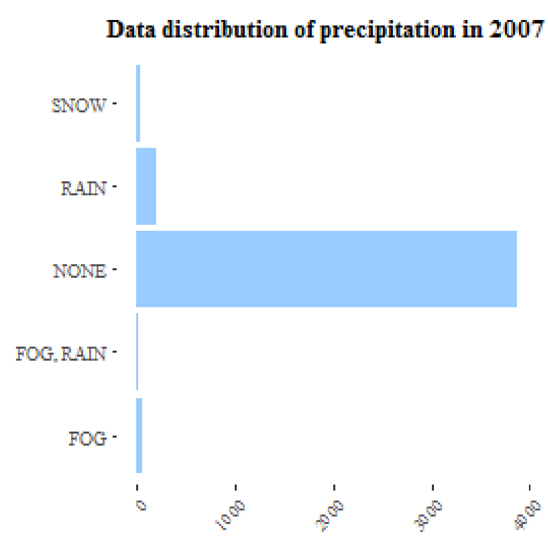

Data distribution of aircraft mass type in 2007
5
4
3
2
1
0
1000
2000
3000

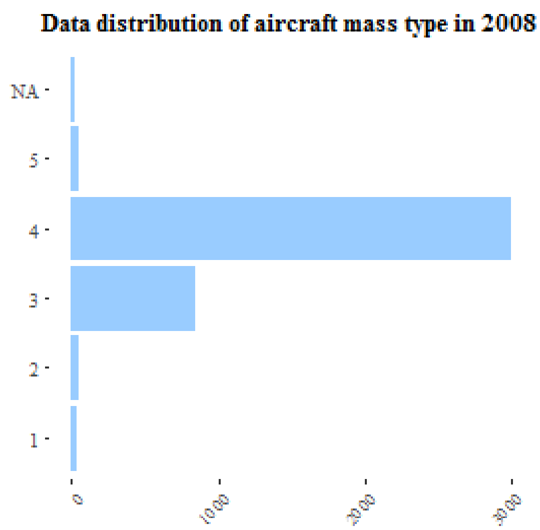

Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
APPROACH
0
500
1000
1500

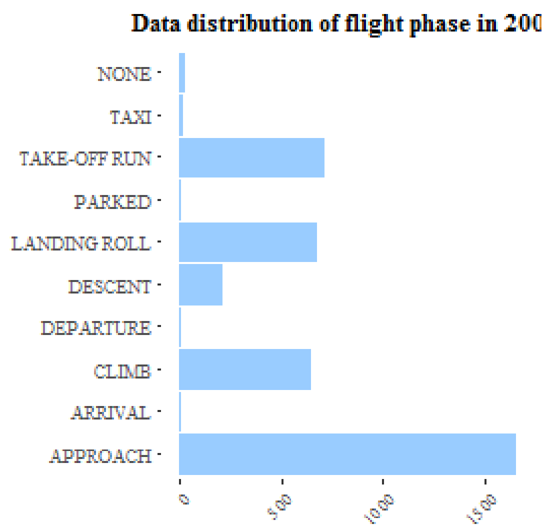

Data distribution of engine type in 2007
D
C
A
0
1000
2000
3000
4000

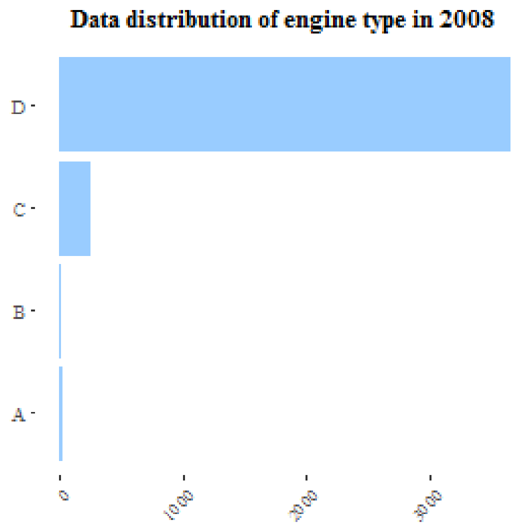

Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500

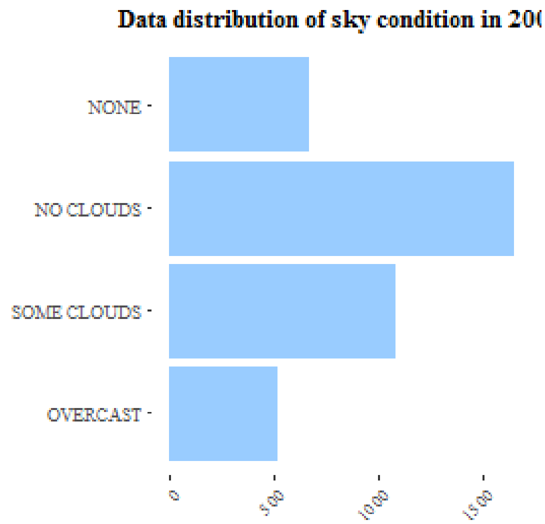

Data distribution of time of day in 2007
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000
1500
2000

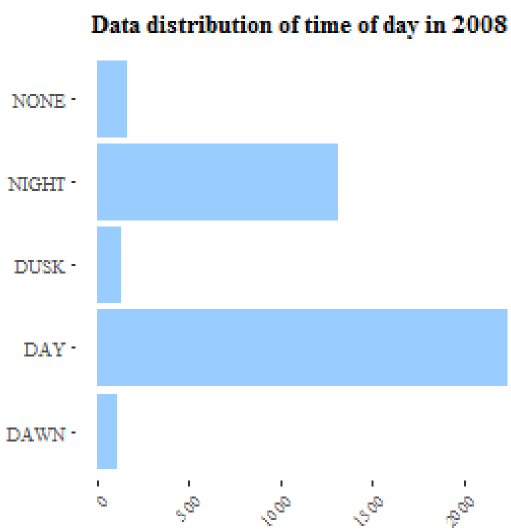

Data distribution of precipitation in 2007
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
1000
2000
3000
4000

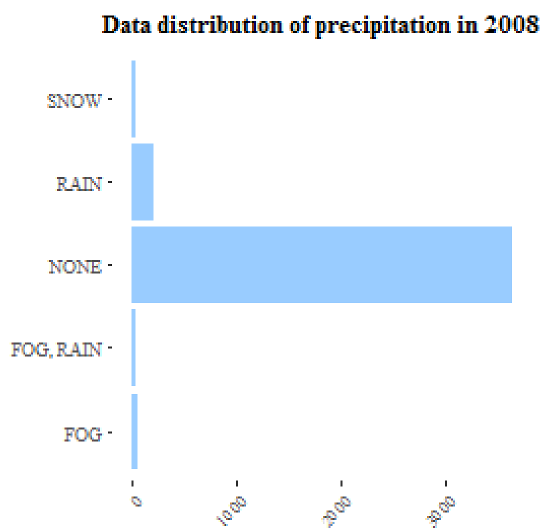

Data distribution of aircraft mass type in 2008
NA
5
4
3
2
1
0
1000
2000
3000


Data distribution of flight phase in 200
NONE
TAXI
TAKE-OFF RUN
PARKED
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000
1500

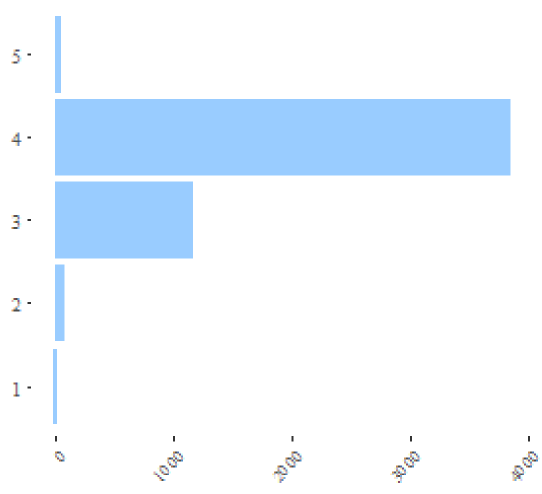

Data distribution of engine type in 2008
D
C
B
A
0
1000
2000
3000


Data distribution of sky condition in 20
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500

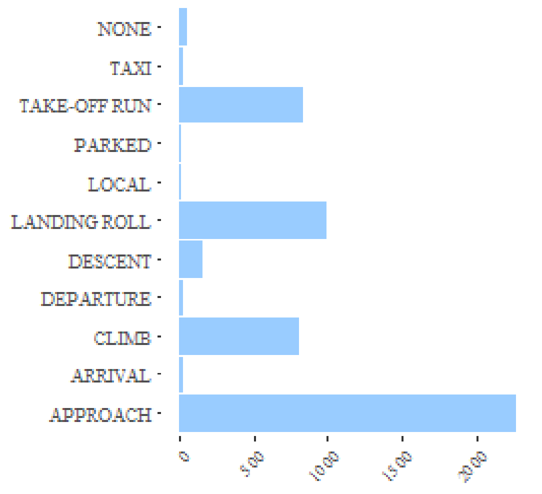

Data distribution of time of day in 2008
NONE
NIGHT
DUSK
DAY
DAWN
0
500
1000
1500
2000


Data distribution of precipitation in 2008
SNOW
RAIN
NONE
FOG, RAIN
FOG
0
1000
2000
3000

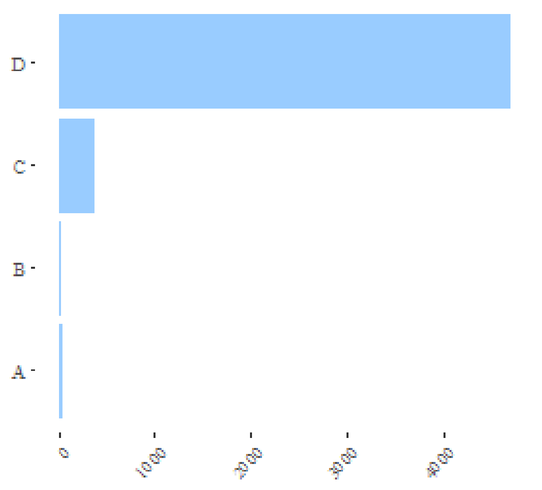
Data distribution of aircraft mass type in 2009



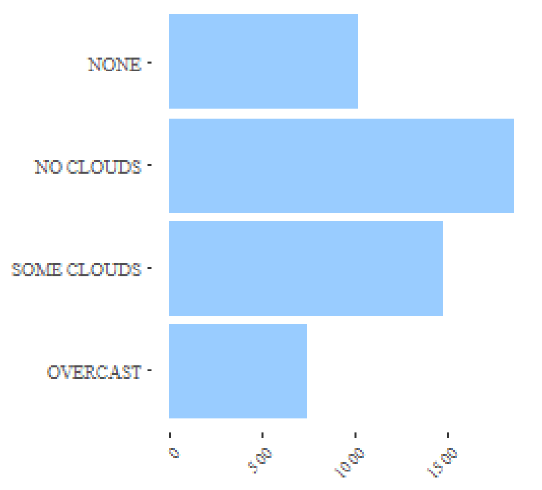
Data distribution of flight phase in 200



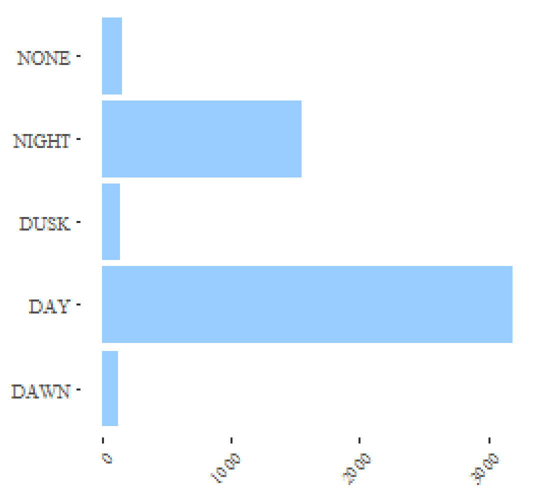
Data distribution of engine type in 2009



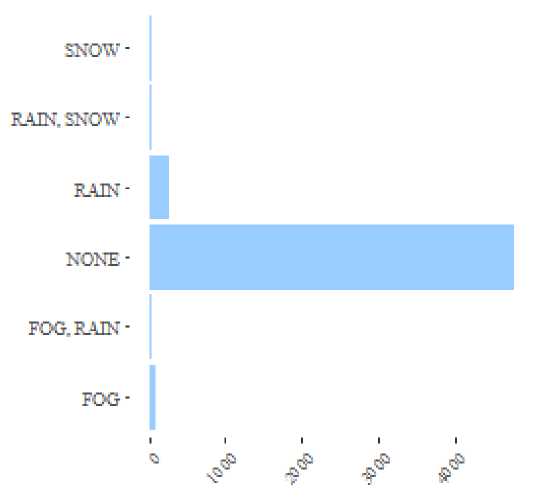
Data distribution of sky condition in 20

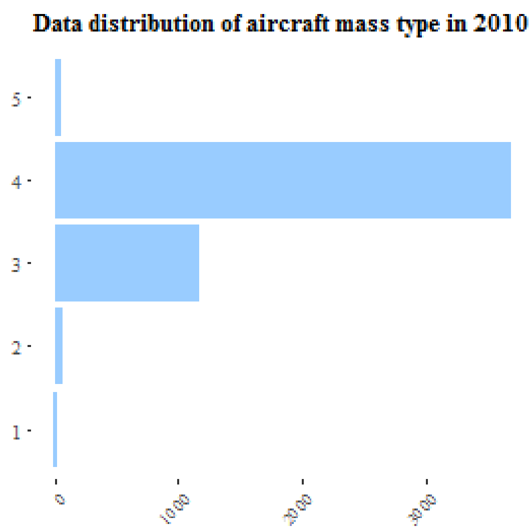


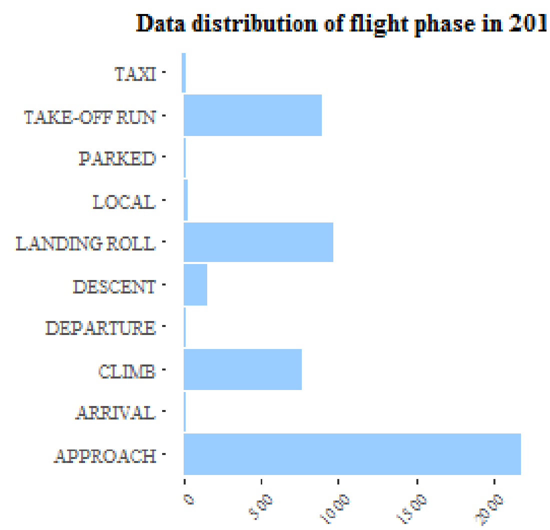
Data distribution of time of day in 2009

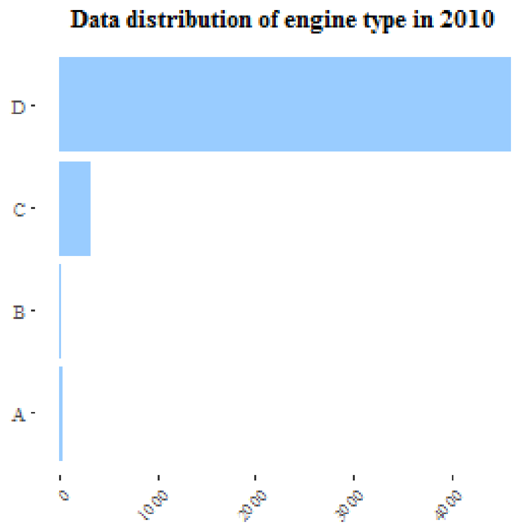


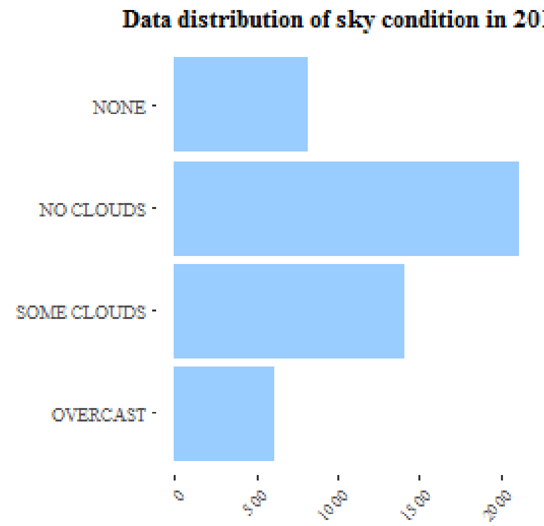
Data distribution of precipitation in 200

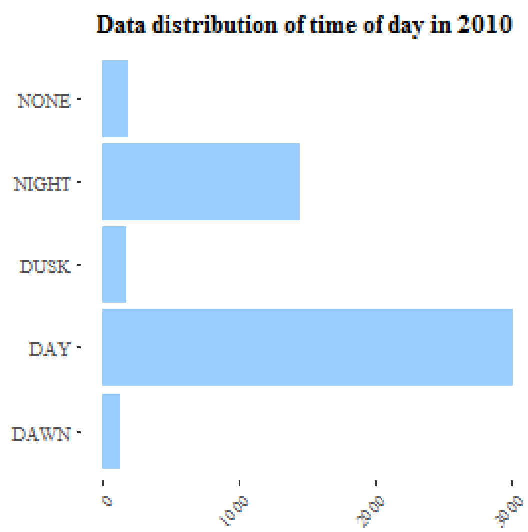


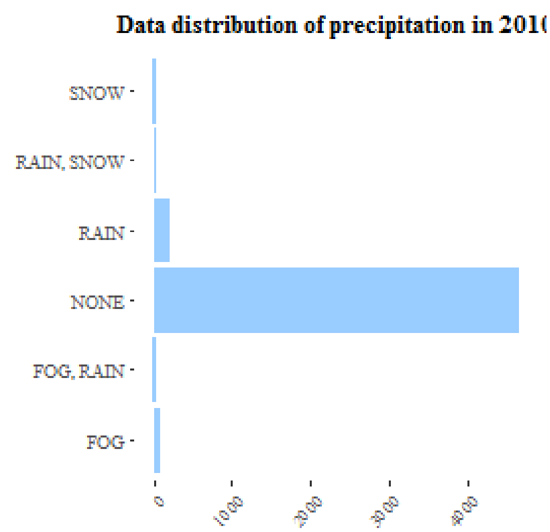

Data distribution of aircraft mass type in 2010
5
4
3
2
1
0
1000
2000
3000

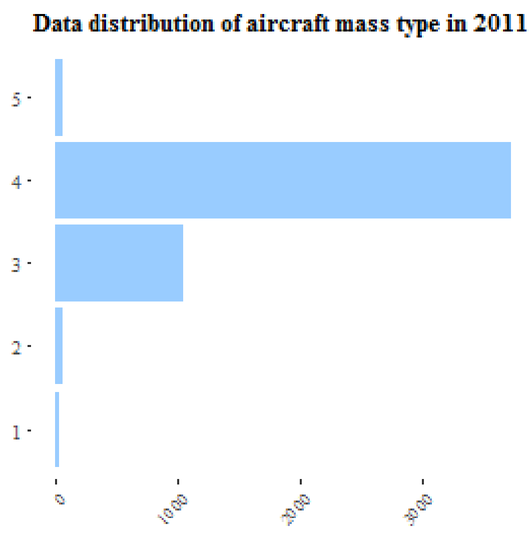

Data distribution of flight phase in 201
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000
1500
2000

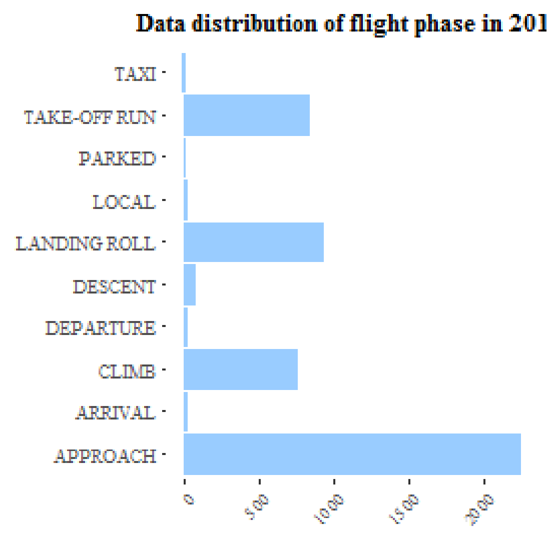

Data distribution of engine type in 2010
D
C
B
A
0
1000
2000
3000
4000

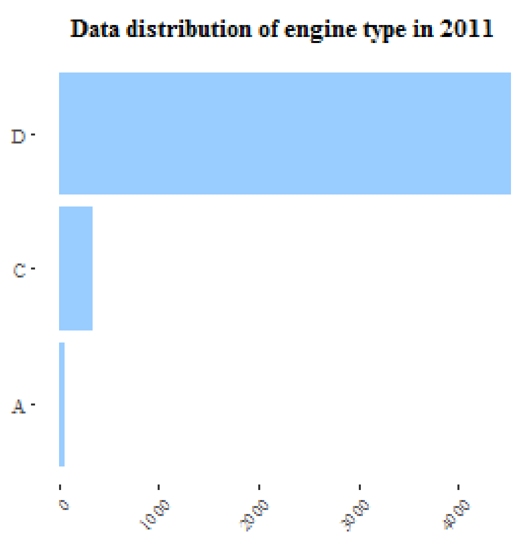

Data distribution of sky condition in 201
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500
2000

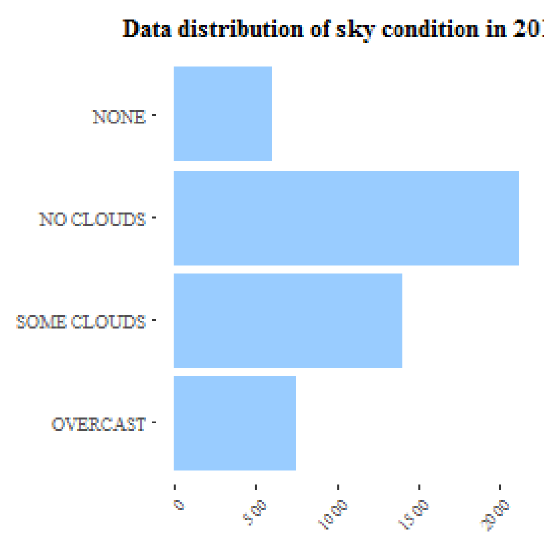

Data distribution of time of day in 2010
NONE
NIGHT
DUSK
DAY
DAWN
0
1000
2000
3000

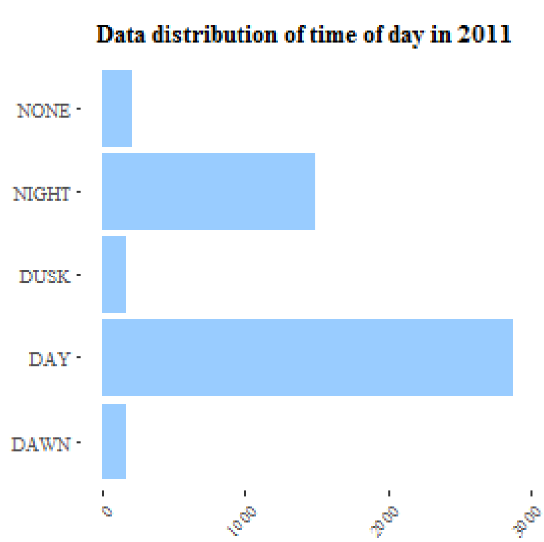

Data distribution of precipitation in 201
SNOW
RAIN, SNOW
RAIN
NONE
FOG, RAIN
FOG
0
1000
2000
3000
4000

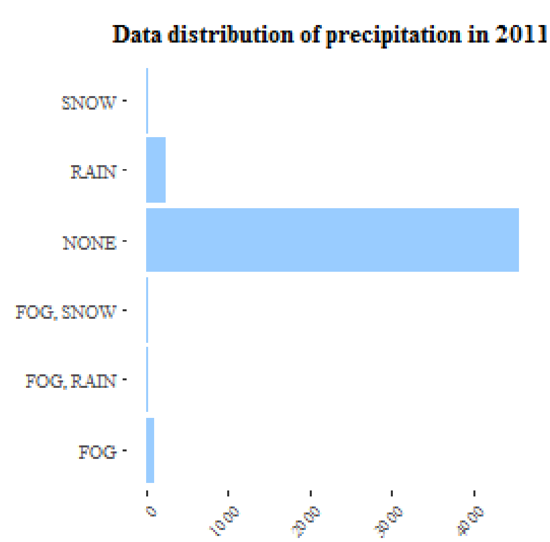

Data distribution of aircraft mass type in 2011
5
4
3
2
1
0
1000
2000
3000

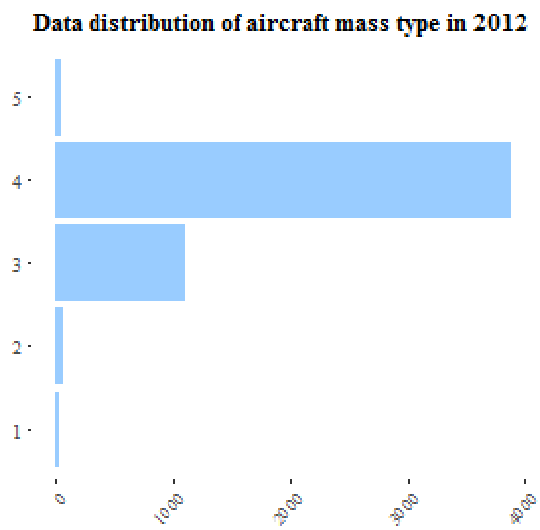

Data distribution of flight phase in 201
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000
1500
2000

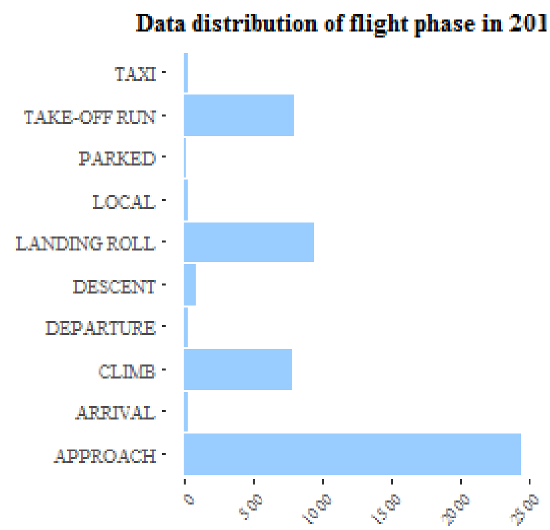

Data distribution of engine type in 2011
D
C
A
0
1000
2000
3000
4000

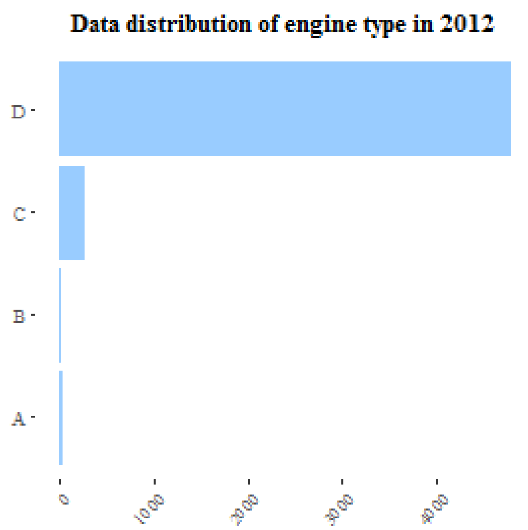

Data distribution of sky condition in 201
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500
2000

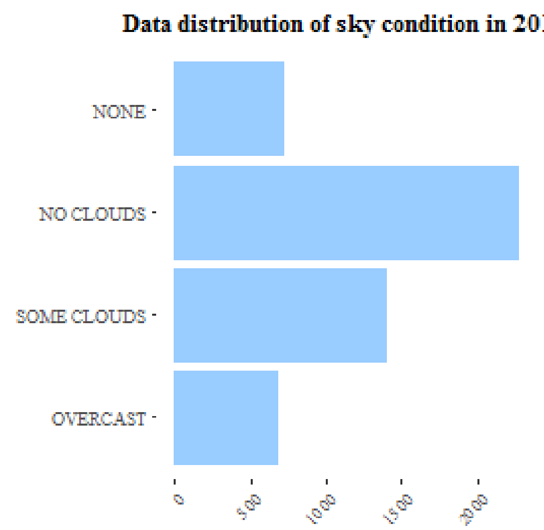

Data distribution of time of day in 2011
NONE
NIGHT
DUSK
DAY
DAWN
0
1000
2000
3000

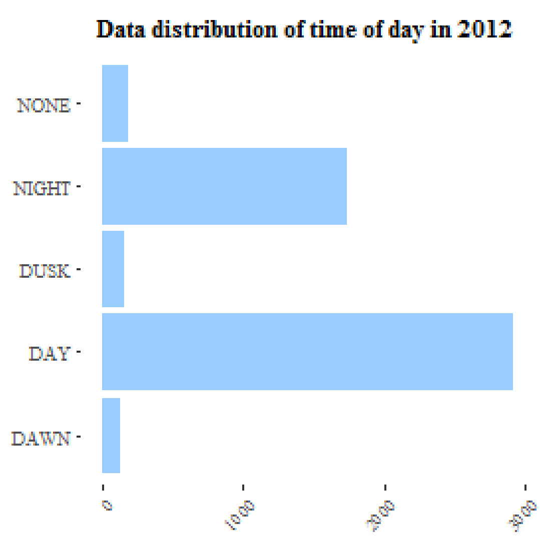

Data distribution of precipitation in 2011
SNOW
RAIN
NONE
FOG, SNOW
FOG, RAIN
FOG
0
1000
2000
3000
4000

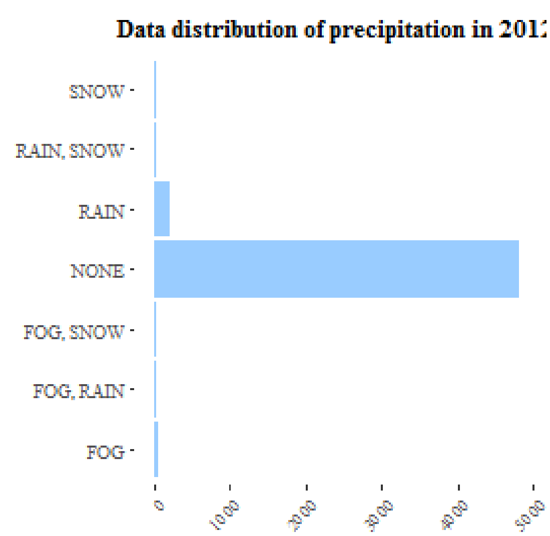

Data distribution of aircraft mass type in 2012
5
4
3
2
1
0
1000
2000
3000
4000

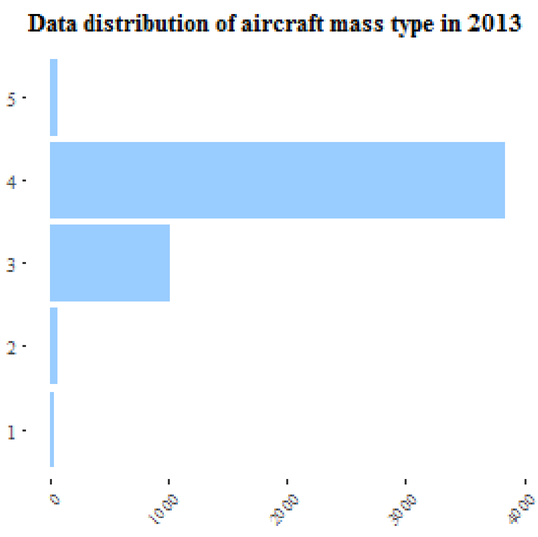

Data distribution of flight phase in 201
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000
1500
2000
2500

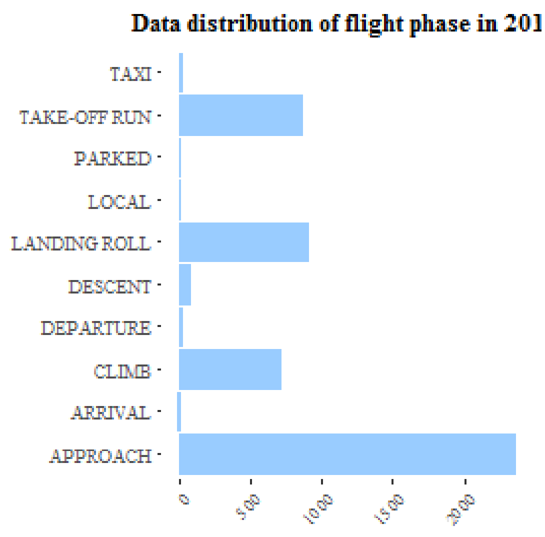

Data distribution of engine type in 2012
D
C
B
A
0
1000
2000
3000
4000

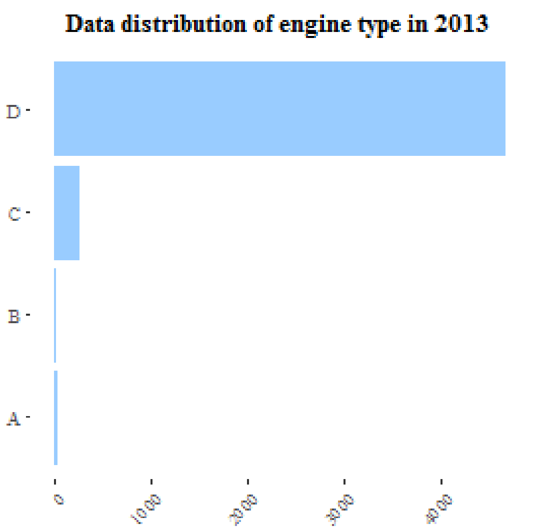

Data distribution of sky condition in 201
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500
2000

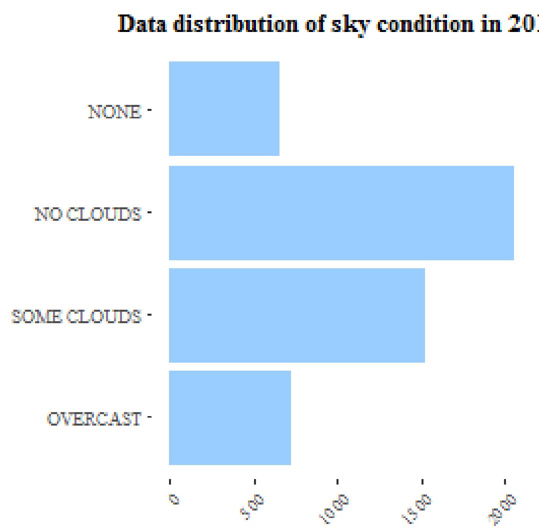

Data distribution of time of day in 2012
NONE
NIGHT
DUSK
DAY
DAWN
0
1000
2000
3000

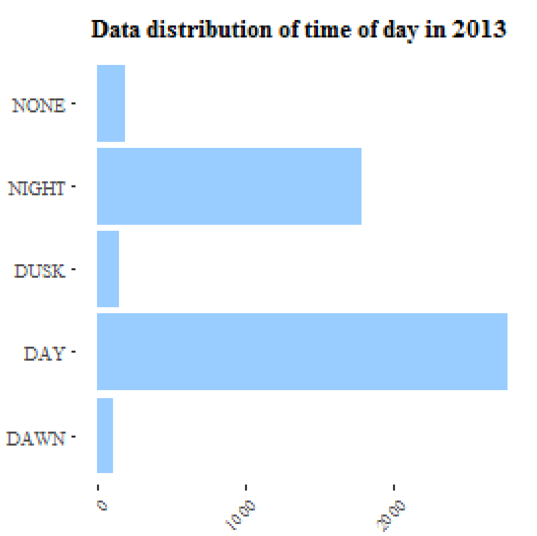

Data distribution of precipitation in 201
SNOW
RAIN, SNOW
RAIN
NONE
FOG, SNOW
FOG, RAIN
FOG
0
1000
2000
3000
4000
5000

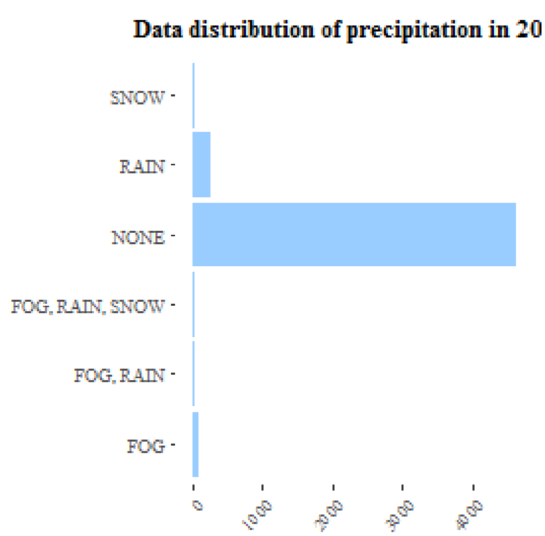

Data distribution of aircraft mass type in 2013
5
4
3
2
1
0
1000
2000
3000
4000

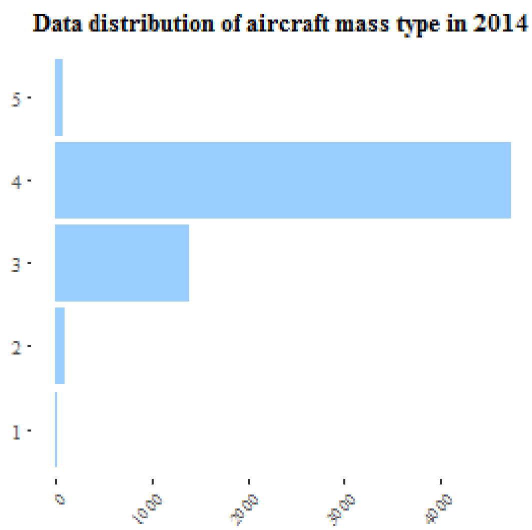

Data distribution of flight phase in 201
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
500
1000
1500
2000

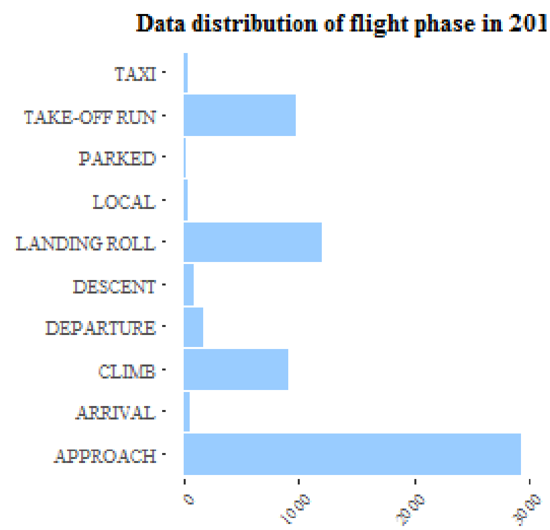

Data distribution of engine type in 2013
D
C
B
A
0
1000
2000
3000
4000

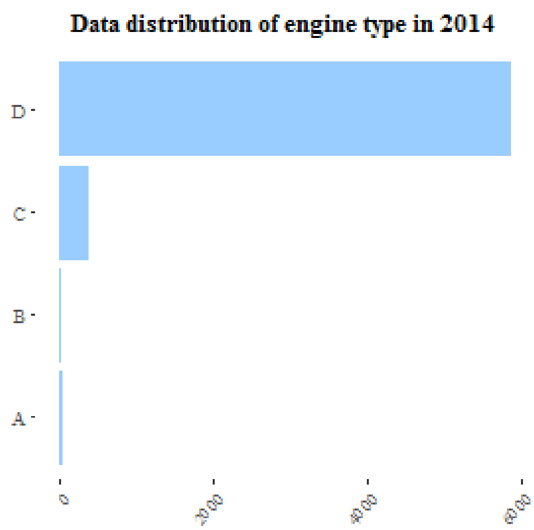

Data distribution of sky condition in 201
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500
2000

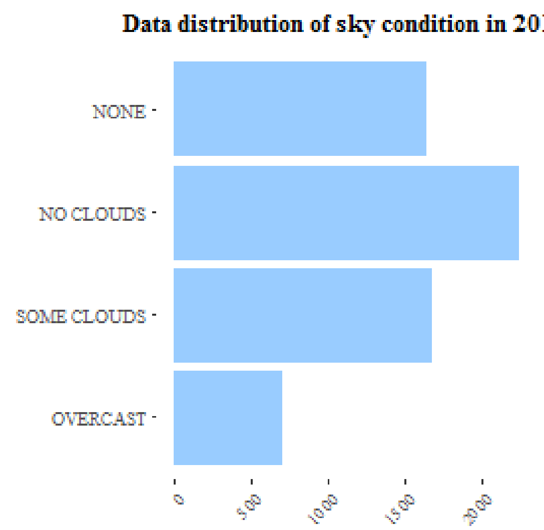

Data distribution of time of day in 2013
NONE
NIGHT
DUSK
DAY
DAWN
0
1000
2000

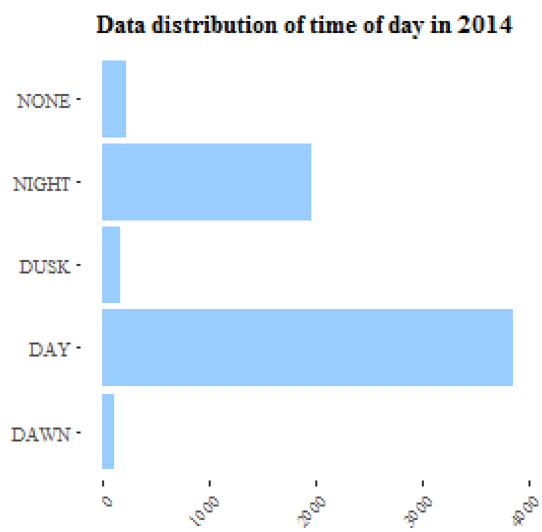

Data distribution of precipitation in 20
SNOW
RAIN
NONE
FOG, RAIN, SNOW
FOG, RAIN
FOG
0
1000
2000
3000
4000

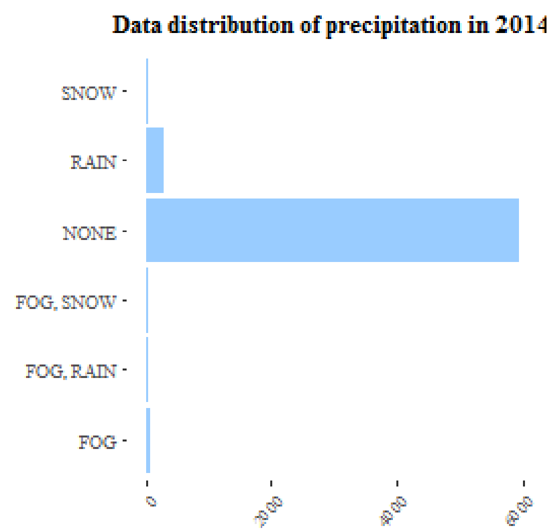

Data distribution of aircraft mass type in 2014
5
4
3
2
1
0
1000
2000
3000
4000

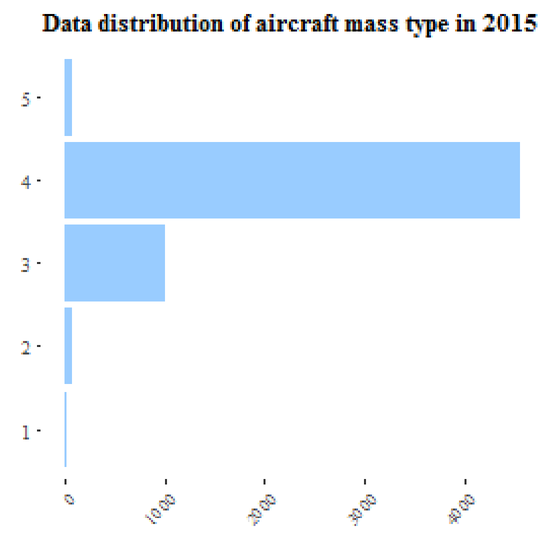

Data distribution of flight phase in 201
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
1000
2000
3000

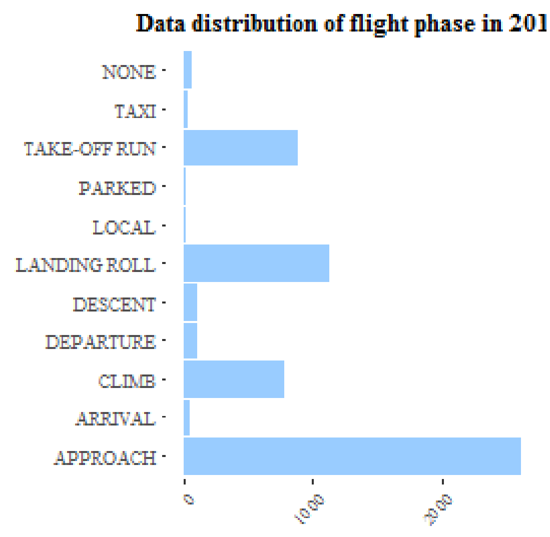

Data distribution of engine type in 2014
D
C
B
A
0
2000
4000
6000

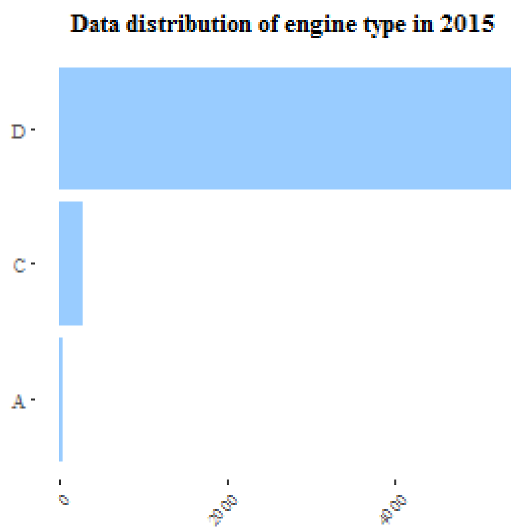

Data distribution of sky condition in 201
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500
2000

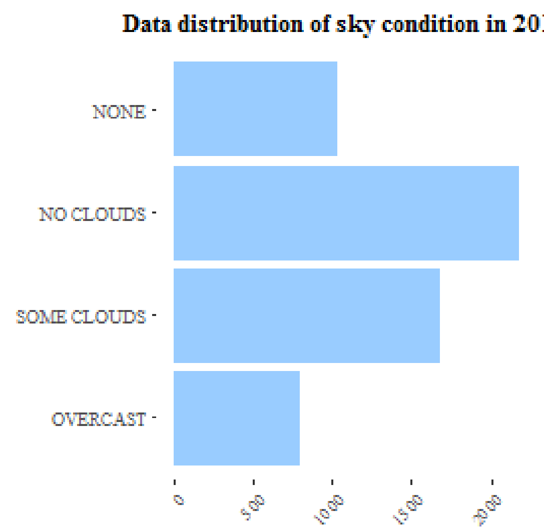

Data distribution of time of day in 2014
NONE
NIGHT
DUSK
DAY
DAWN
0
1000
2000
3000
4000

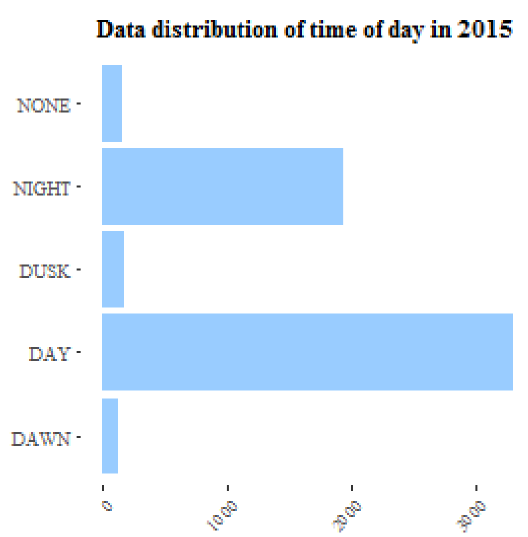

Data distribution of precipitation in 2014
SNOW
RAIN
NONE
FOG, SNOW
FOG, RAIN
FOG
0
2000
4000
6000

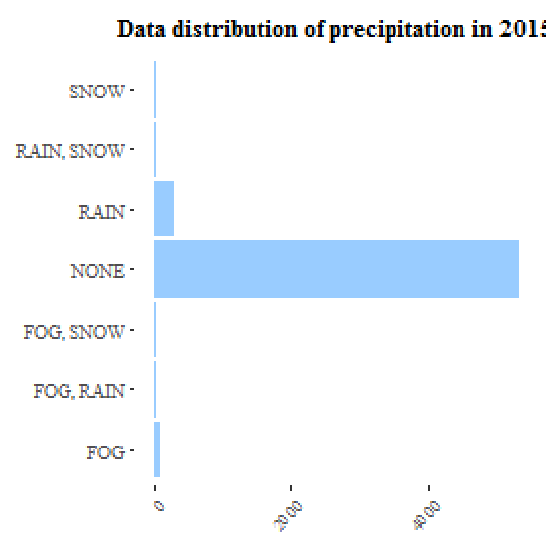

Data distribution of aircraft mass type in 2015
5
4
3
2
1
0
1000
2000
3000
4000

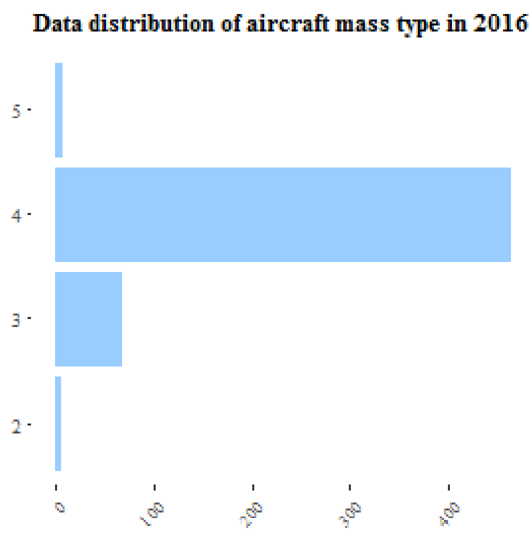

Data distribution of flight phase in 201
NONE
TAXI
TAKE-OFF RUN
PARKED
LOCAL
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
1000
2000

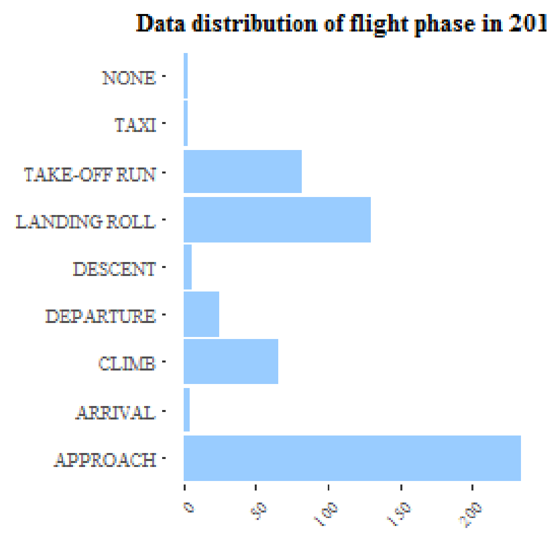

Data distribution of engine type in 2015
D
C
A
0
2000
4000

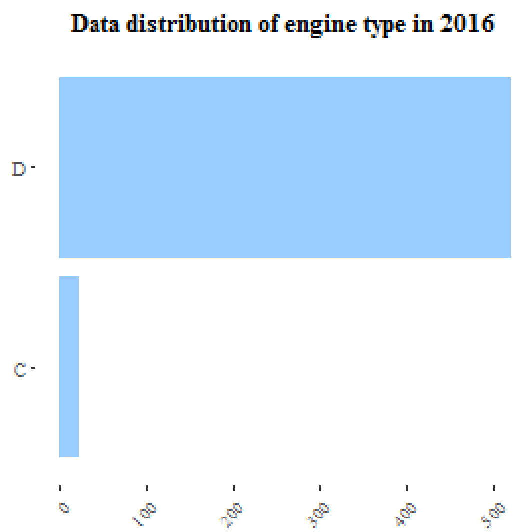

Data distribution of sky condition in 201
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
500
1000
1500
2000

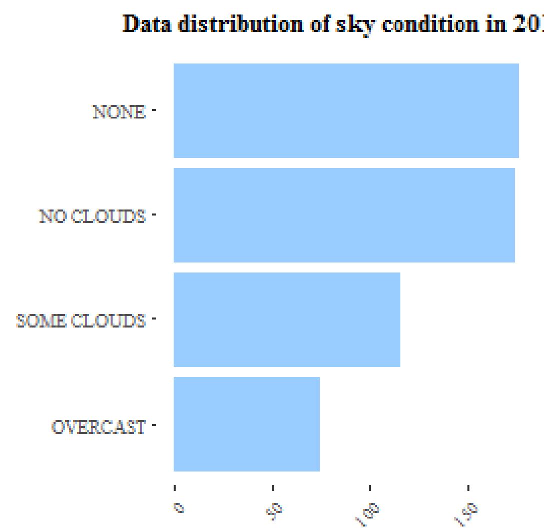

Data distribution of time of day in 2015
NONE
NIGHT
DUSK
DAY
DAWN
0
1000
2000
3000

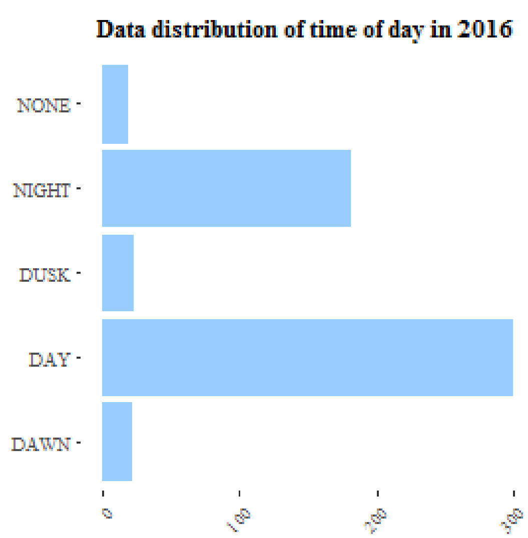

Data distribution of precipitation in 201
SNOW
RAIN, SNOW
RAIN
NONE
FOG, SNOW
FOG, RAIN
FOG
0
2000
4000

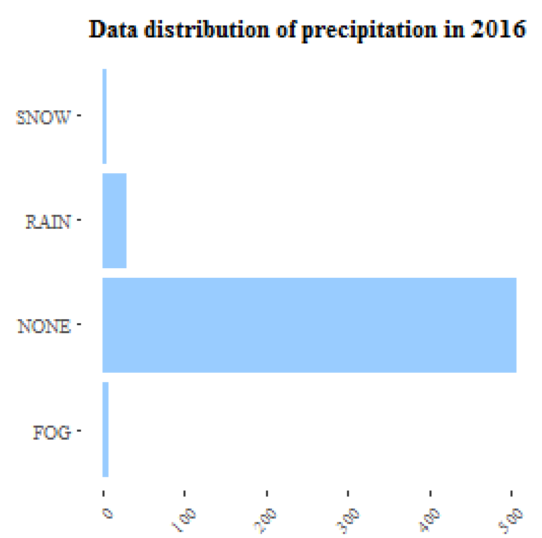

Data distribution of aircraft mass type in 2016
5
4
3
2
0
100
200
300
400


Data distribution of flight phase in 201
NONE
TAXI
TAKE-OFF RUN
LANDING ROLL
DESCENT
DEPARTURE
CLIMB
ARRIVAL
APPROACH
0
50
100
150
200


Data distribution of engine type in 2016
D
C
0
100
200
300
400
500


Data distribution of sky condition in 201
NONE
NO CLOUDS
SOME CLOUDS
OVERCAST
0
50
100
150


Data distribution of time of day in 2016
NONE
NIGHT
DUSK
DAY
DAWN
0
100
200
300


Data distribution of precipitation in 2016
SNOW
RAIN
NONE
FOG
0
100
200
300
400
500

15.1.2 Flight Data (1990 - 2016)

The first summary table shows the number of distinct items for each year regarding the number of records, the carriers, and the origin and the destination airports after the selection and cleanup tasks.

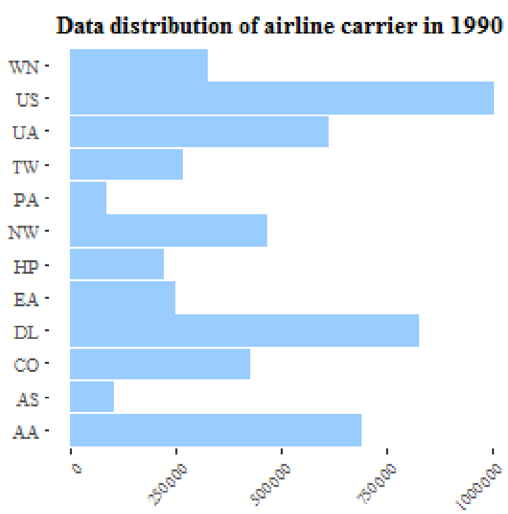
Year	# of flights	# of carriers	Origin airports	Origin states	Destination airports	Destination states
1990	5220743	12	226	49	227	49
1991	5025091	12	225	49	225	49
1992	5040279	10	226	49	226	49
1993	5019147	10	219	49	219	49
1994	5133635	10	218	49	219	49
1995	5277791	10	213	49	213	49
1996	5308054	10	206	49	207	49
1997	5367484	10	202	49	202	49
1998	5339590	10	204	49	205	49
1999	5479428	10	202	49	202	49
2000	5626936	11	202	49	202	49
2001	5908140	12	226	49	225	49
2002	5217254	10	213	48	214	48
2003	6433097	18	278	49	278	49
2004	7068362	19	281	49	284	49
2005	7080554	20	281	49	284	49
2006	7081884	21	284	50	291	50
2007	7397375	20	299	50	305	50
2008	6955720	20	298	49	299	49
2009	6396564	19	291	49	291	49
2010	6394653	18	299	49	299	49
2011	6028609	16	292	49	294	49
2012	6034248	15	306	49	306	49
2013	6305662	16	312	50	310	50
2014	5754680	14	317	50	316	50
2015	5751630	14	315	50	315	50
2016	1786119	12	290	49	290	49

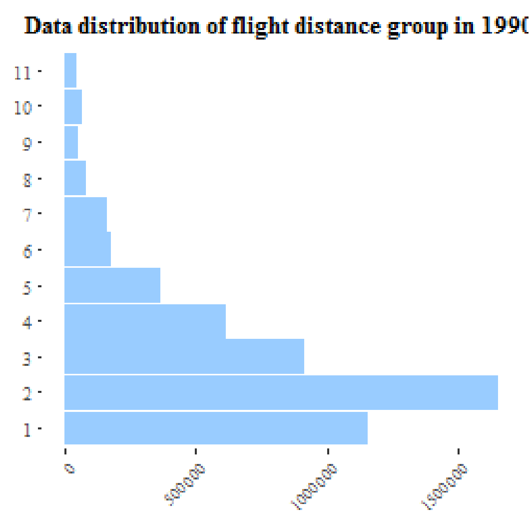
The second summary table shows the number of distinct items for each year the departure time group and distance between the airports after the selection and cleanup tasks.

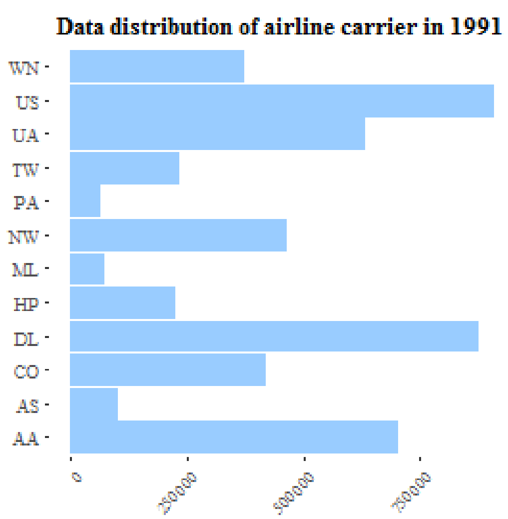
Year	Departure time block	Distance group
1990	19	11
1991	19	11
1992	19	11
1993	19	11
1994	19	11
1995	19	11
1996	19	11
1997	19	11
1998	19	11
1999	19	11
2000	19	11
2001	19	11
2002	19	11
2003	19	11

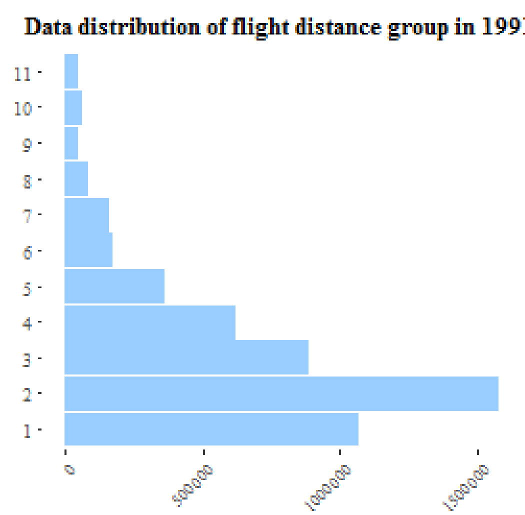
Year	Departure time block	Distance group
2004	19	11
2005	19	11
2006	19	11
2007	19	11
2008	19	11
2009	19	11
2010	19	11
2011	19	11
2012	20	11
2013	19	11
2014	19	11
2015	19	11
2016	19	11

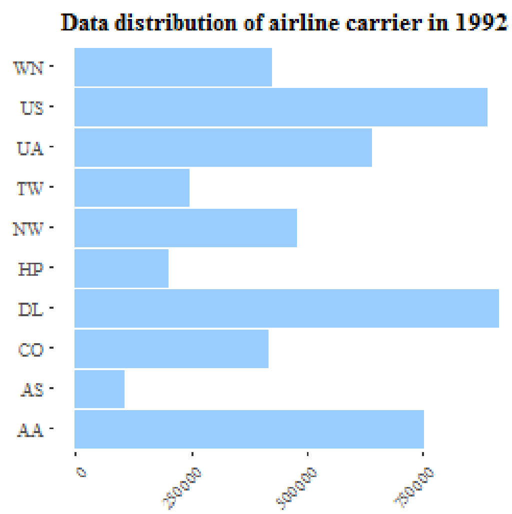
The following graphs show the distributions of some of the selected distinct items summarized in the tables above.

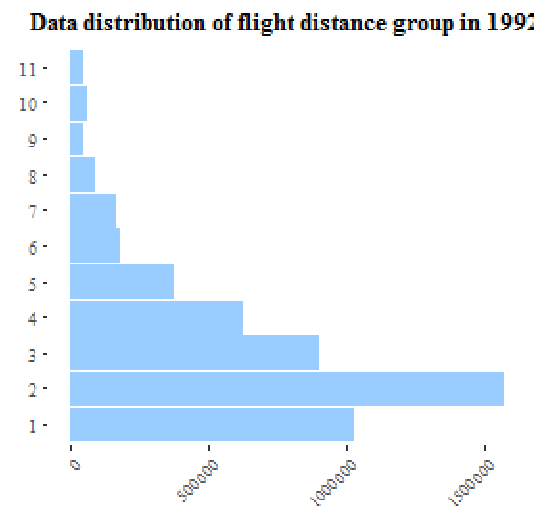


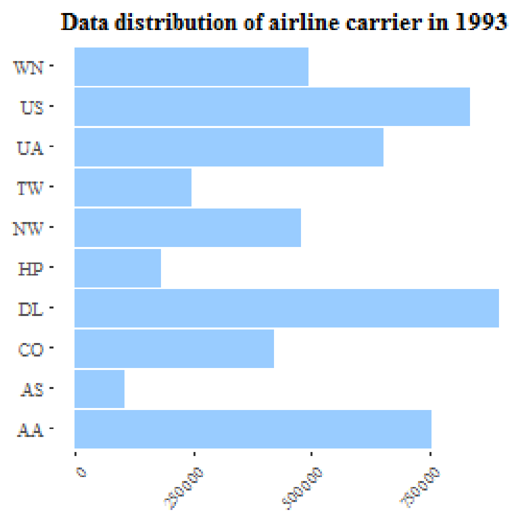


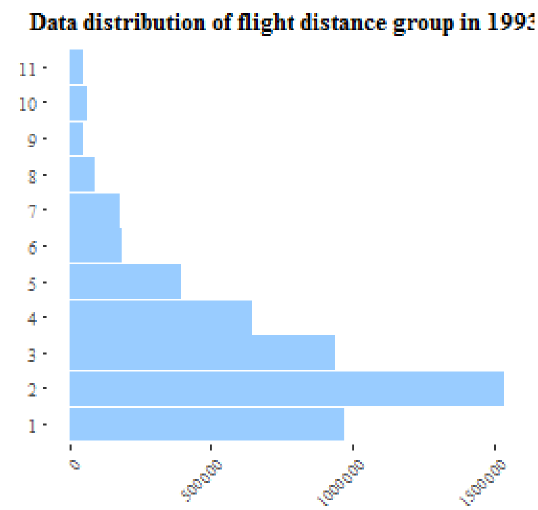


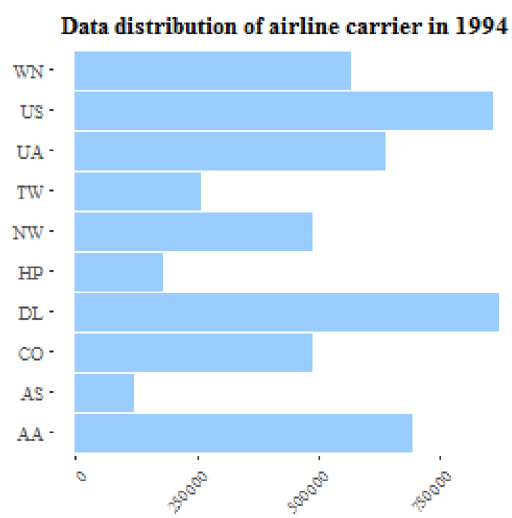


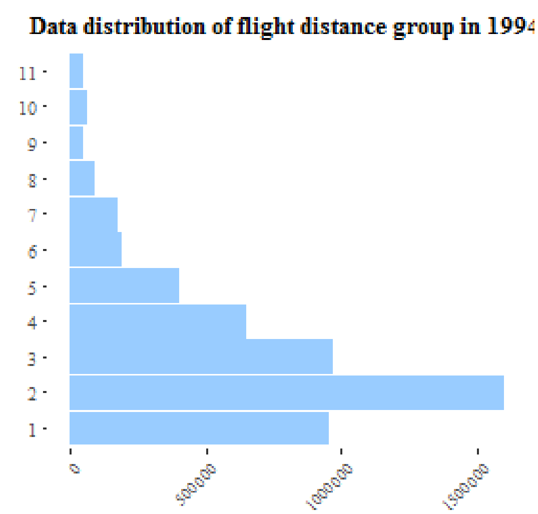

Data distribution of airline carrier in 1992
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

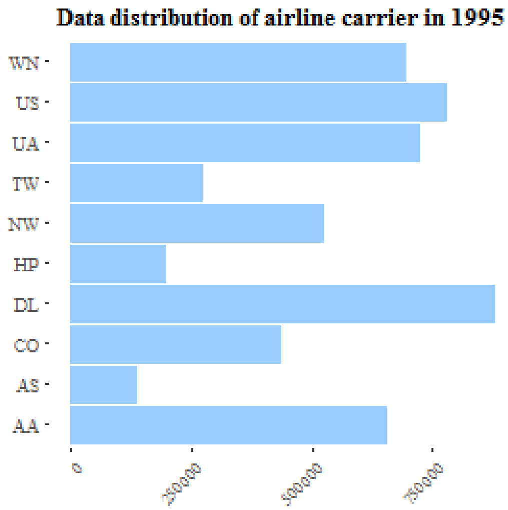

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

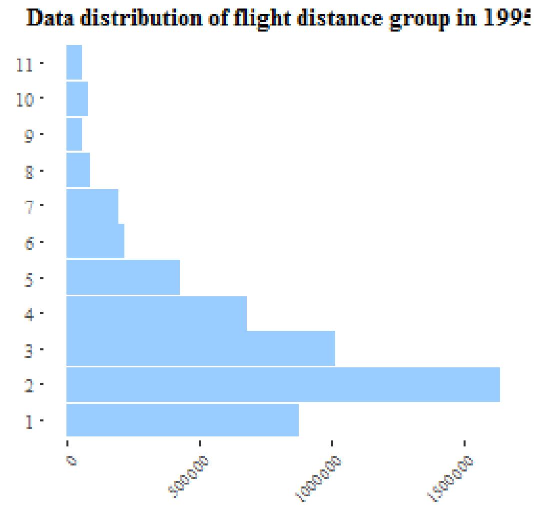

Data distribution of airline carrier in 1993
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

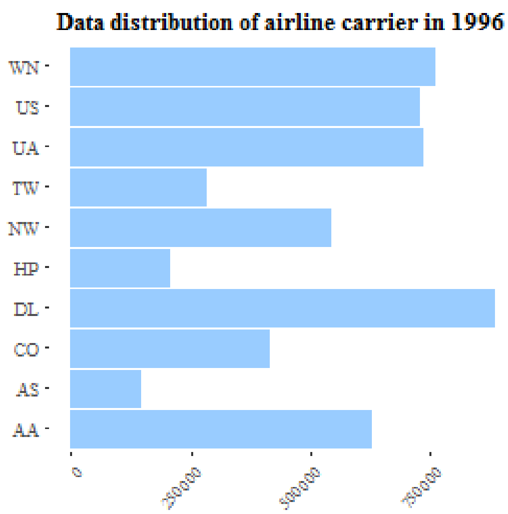

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

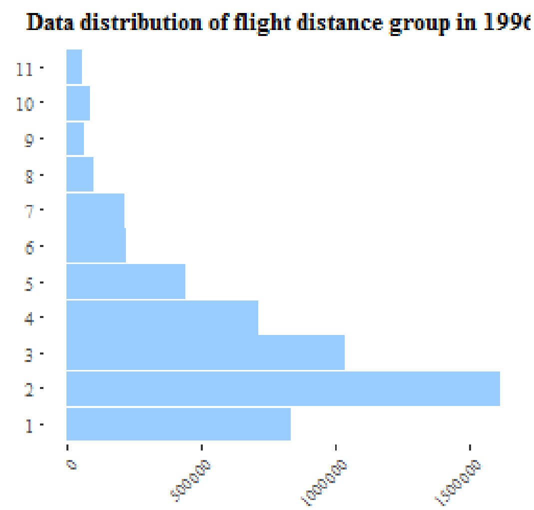

Data distribution of airline carrier in 1994
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

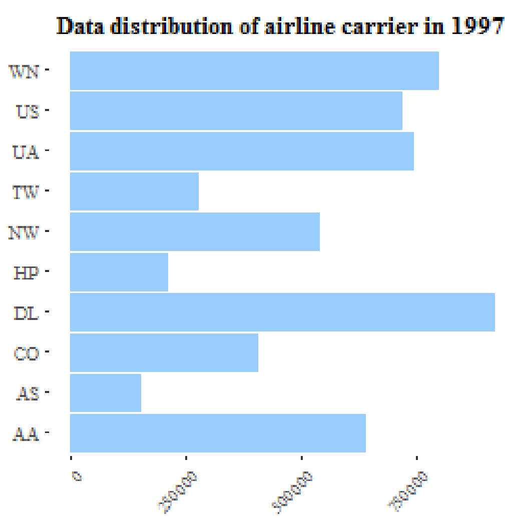

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

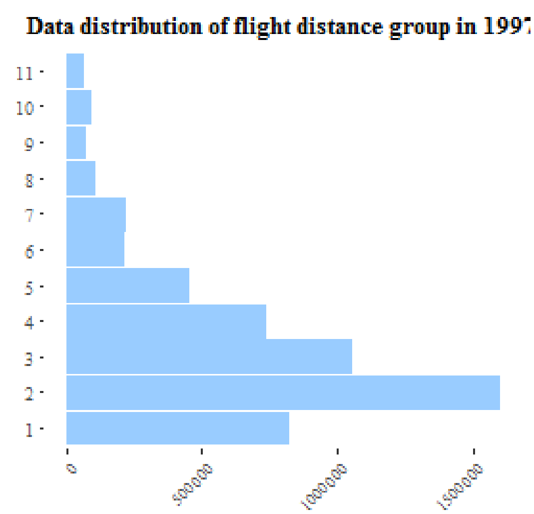

Data distribution of airline carrier in 1995
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

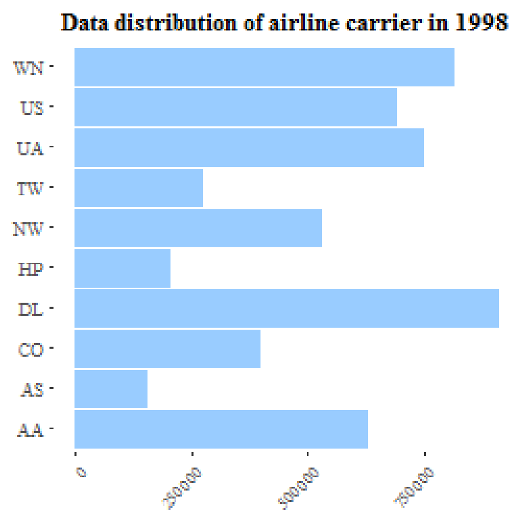

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

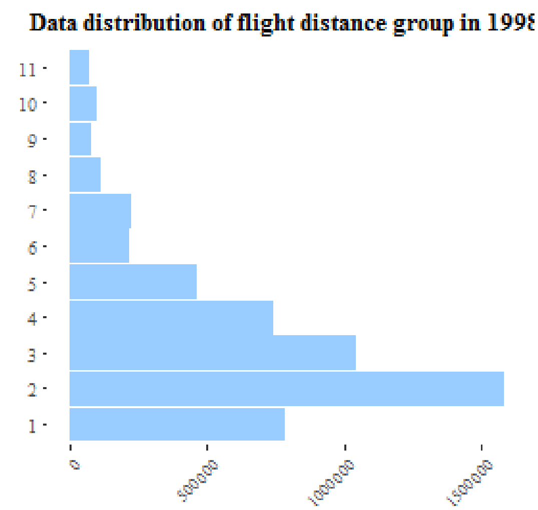

Data distribution of airline carrier in 1996
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

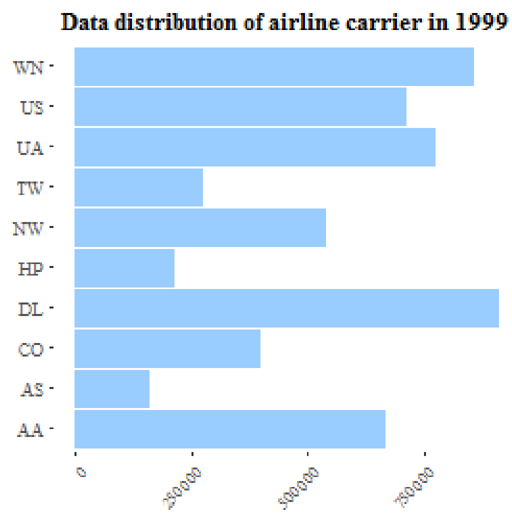

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

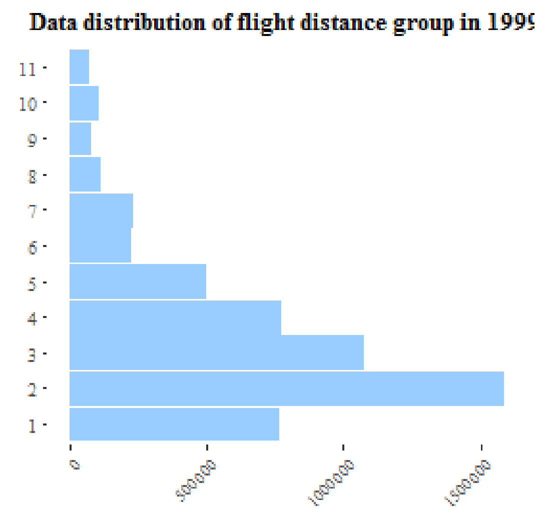

Data distribution of airline carrier in 1997
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

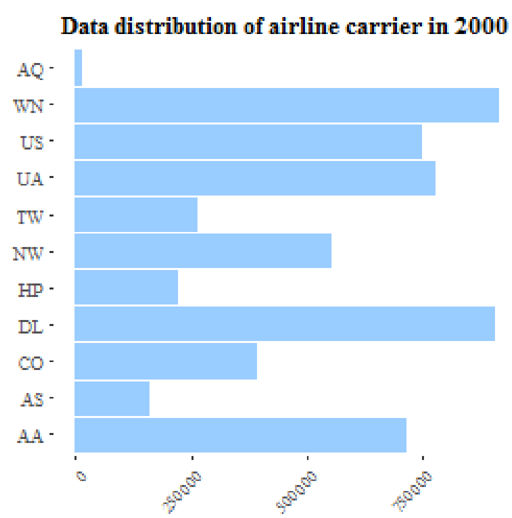

Data distribution of flight distance group in 199
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

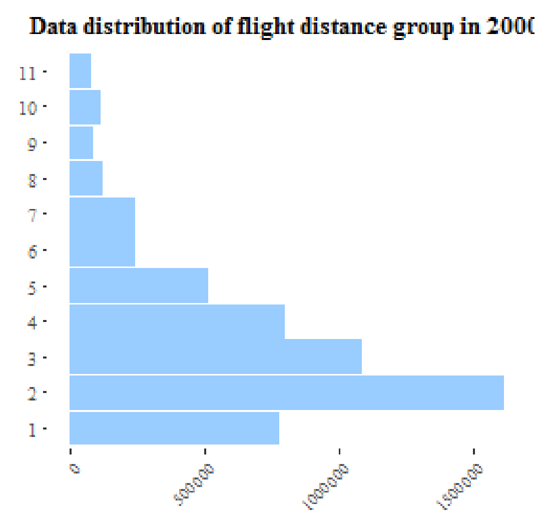

Data distribution of airline carrier in 1998
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

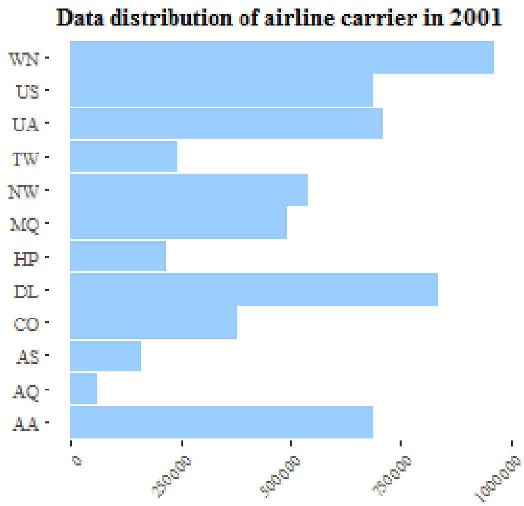

Data distribution of flight distance group in 1998
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

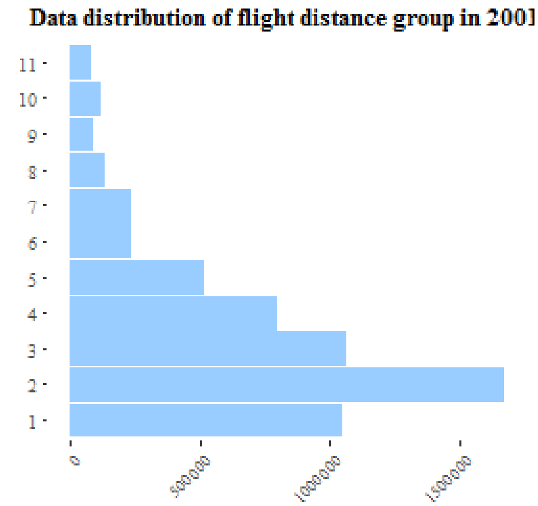

Data distribution of airline carrier in 1999
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

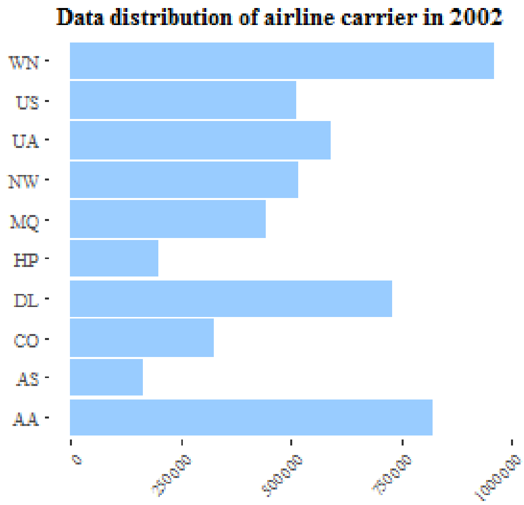

Data distribution of flight distance group in 1999
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

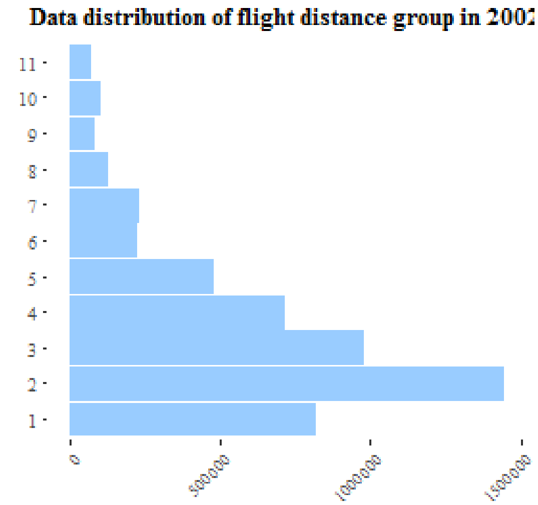

Data distribution of airline carrier in 2000
AQ
WN
US
UA
TW
NW
HP
DL
CO
AS
AA
0
250000
500000
750000

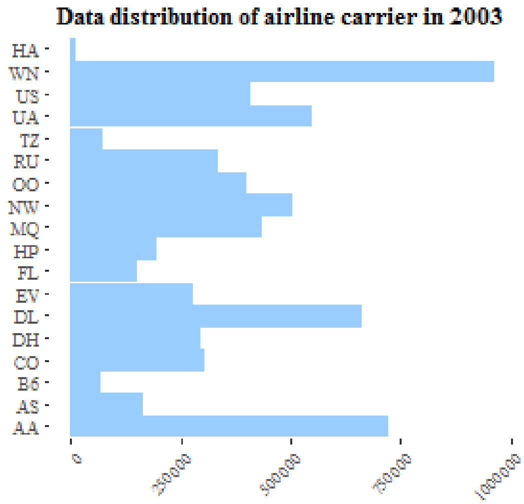

Data distribution of flight distance group in 2000
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

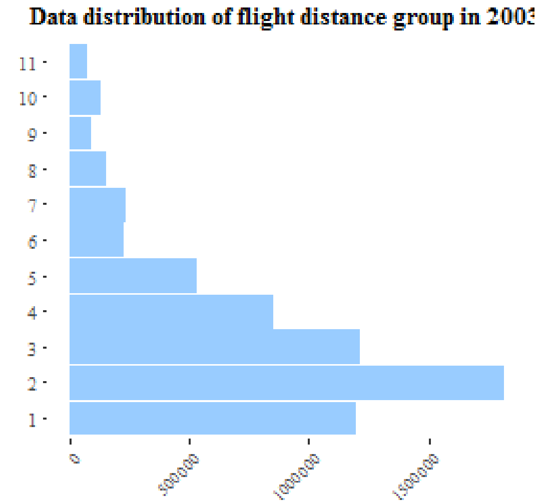

Data distribution of airline carrier in 2001
WN
US
UA
TW
NW
MQ
HP
DL
CO
AS
AQ
AA
0
250000
500000
750000
1000000

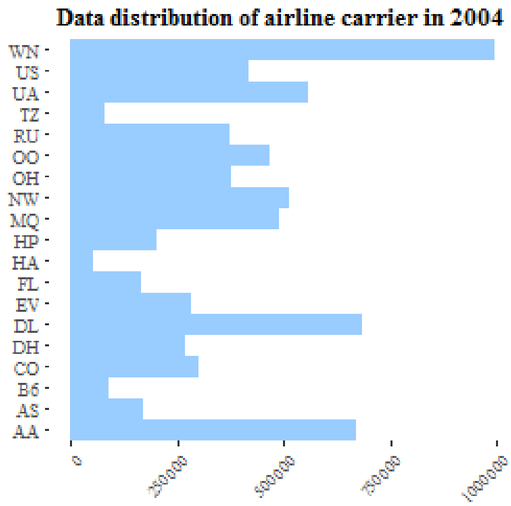

Data distribution of flight distance group in 2001
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

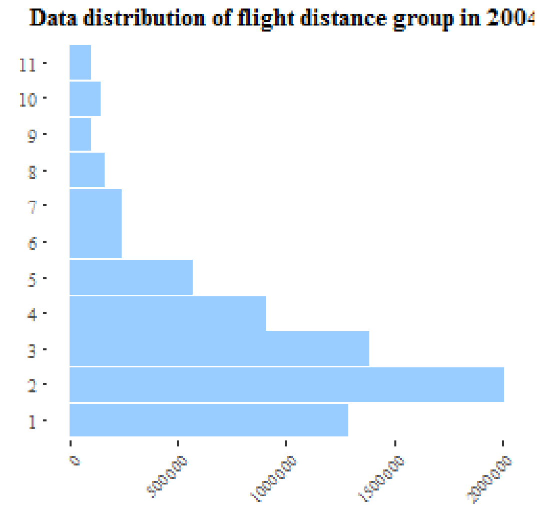

Data distribution of airline carrier in 2002
WN
US
UA
NW
MQ
HP
DL
CO
AS
AA
0
250000
500000
750000
1000000

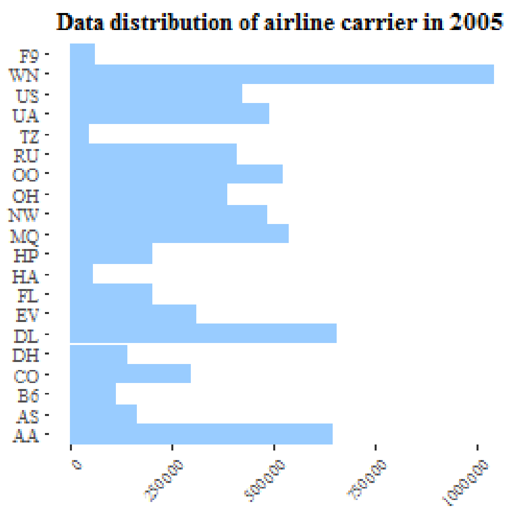

Data distribution of flight distance group in 2002
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

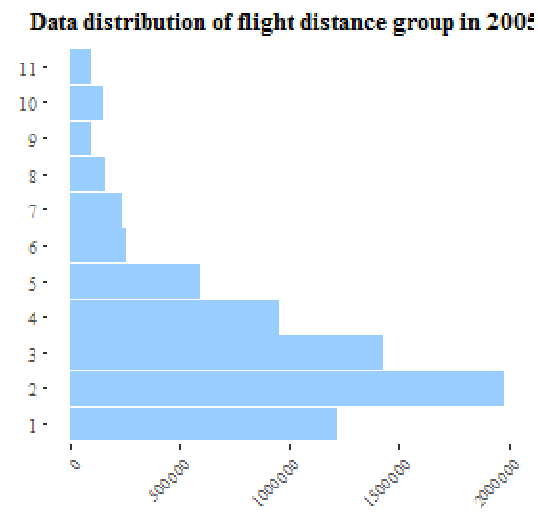

Data distribution of airline carrier in 2003
HA
WN
US
UA
TZ
RU
OO
NW
MQ
HP
FL
EV
DL
DH
CO
B6
AS
AA
0
250000
500000
750000
1000000

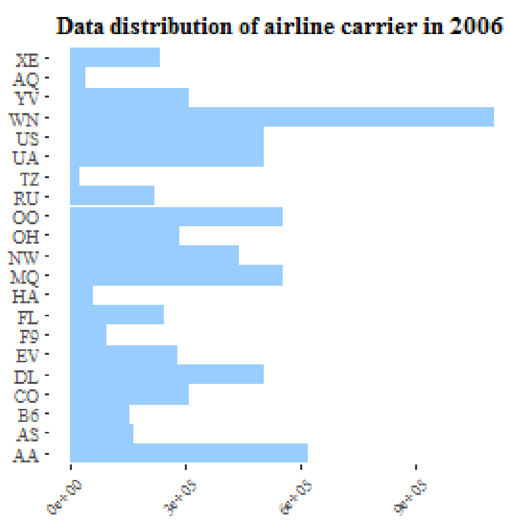

Data distribution of flight distance group in 2003
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

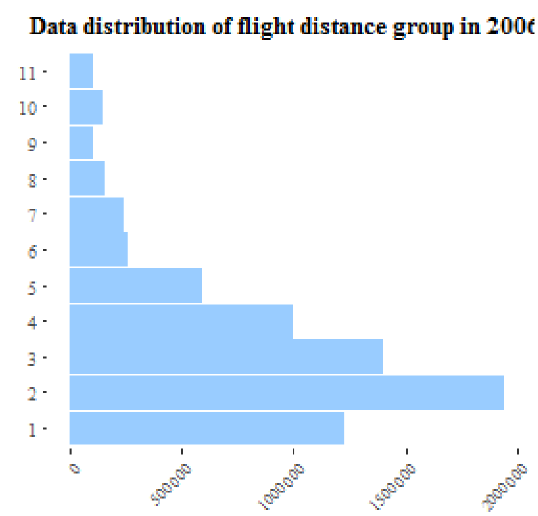

Data distribution of airline carrier in 2004
WN
US
UA
TZ
RU
OO
OH
NW
MQ
HP
HA
FL
EV
DL
DH
CO
B6
AS
AA
0
250000
500000
750000
1000000

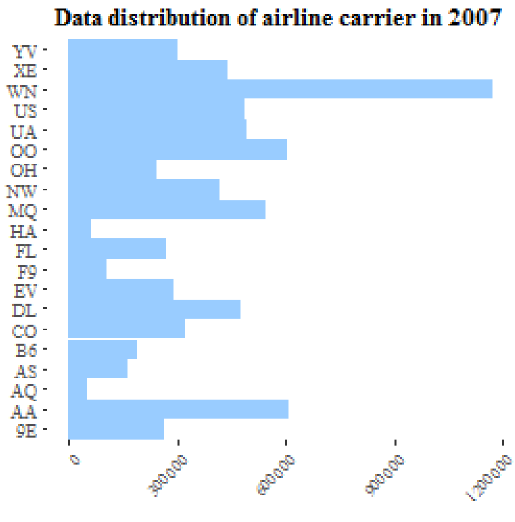

Data distribution of flight distance group in 2004
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

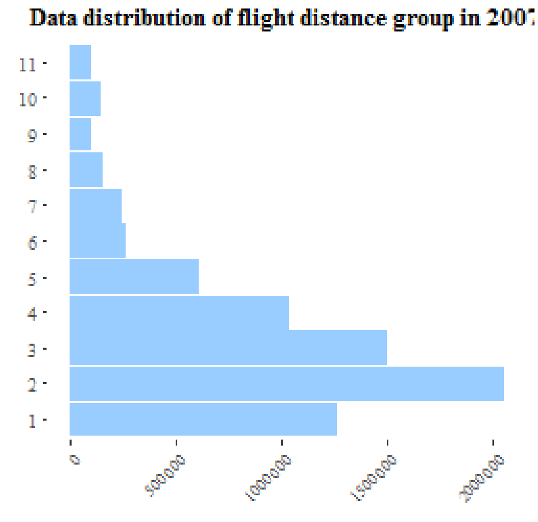

Data distribution of airline carrier in 2005
F9
WN
US
UA
TZ
RU
OO
OH
NW
MQ
HP
HA
FL
EV
DL
DH
CO
B6
AS
AA
0
250000
500000
750000
1000000

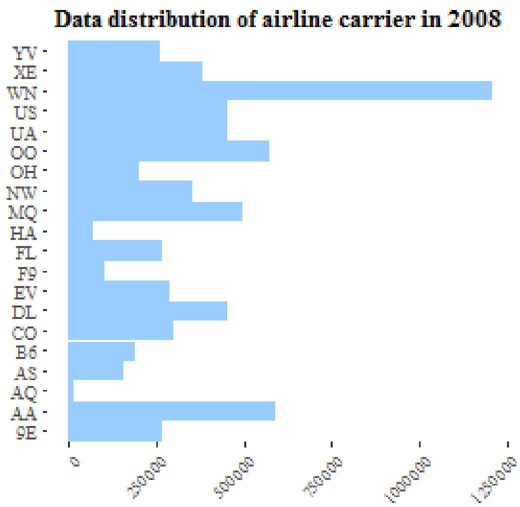

Data distribution of flight distance group in 2005
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

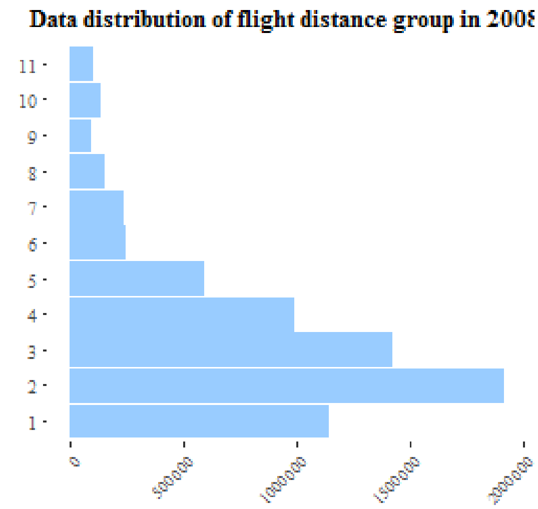

Data distribution of airline carrier in 2006
XE
AQ
YV
WN
US
UA
TZ
RU
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
0e+00
3e+05
6e+05
9e+05

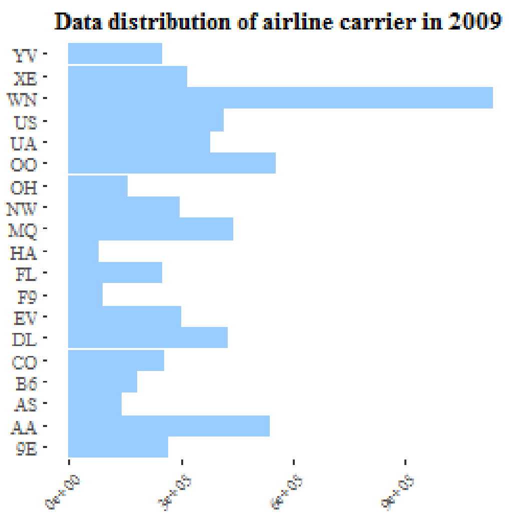

Data distribution of flight distance group in 2006
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

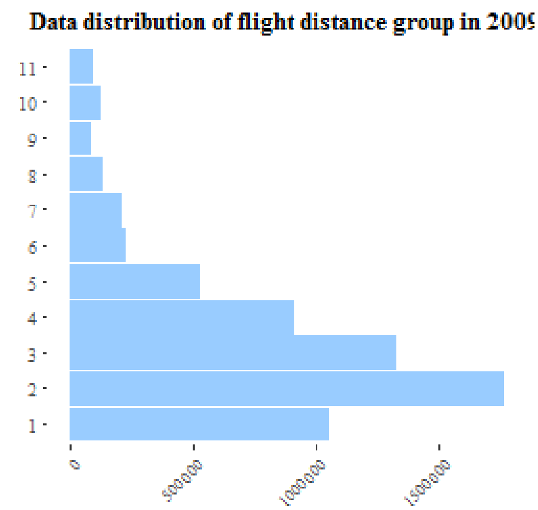

Data distribution of airline carrier in 2007
YV
XE
WN
US
UA
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AQ
AA
9E
0
300000
600000
900000
1200000

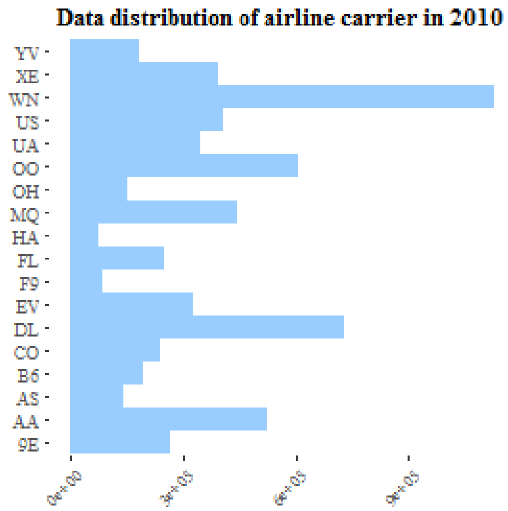

Data distribution of flight distance group in 2007
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

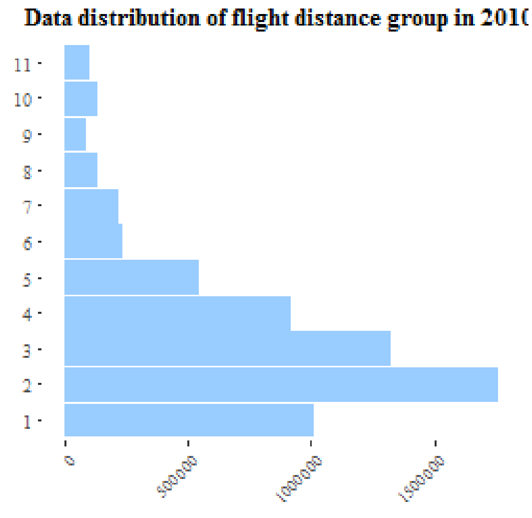

Data distribution of airline carrier in 2008
YV
XE
WN
US
UA
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AQ
AA
9E
0
250000
500000
750000
1000000
1250000

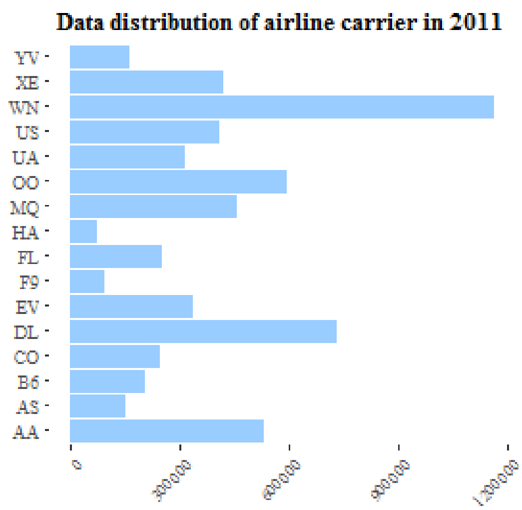

Data distribution of flight distance group in 2008
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000
2000000

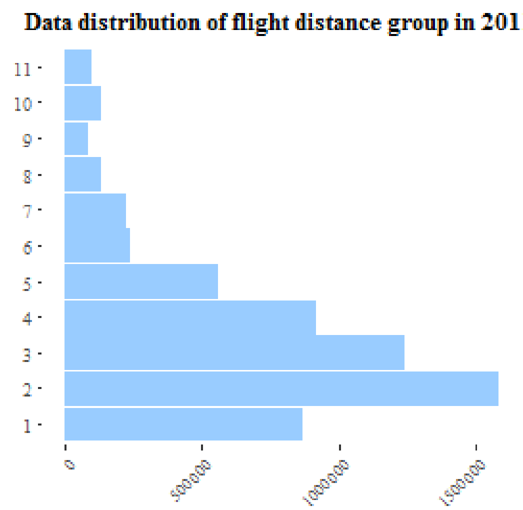

Data distribution of airline carrier in 2009
YV
XE
WN
US
UA
OO
OH
NW
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
9E
0e+00
3e+05
6e+05
9e+05

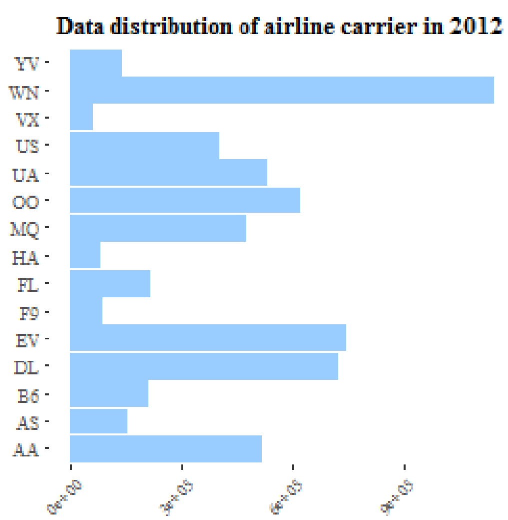

Data distribution of flight distance group in 2009
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

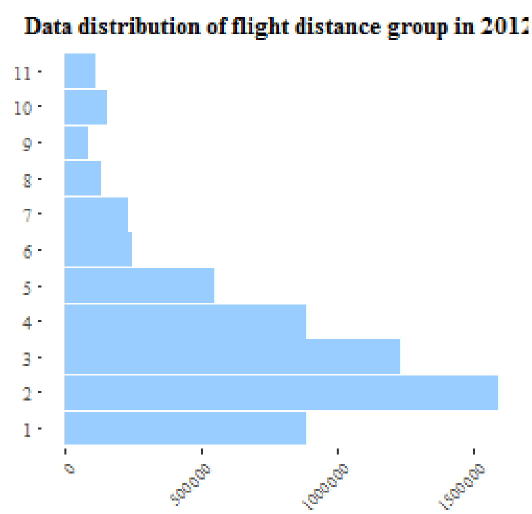

Data distribution of airline carrier in 2010
YV
XE
WN
US
UA
OO
OH
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
9E
0e+00
3e+05
6e+05
9e+05

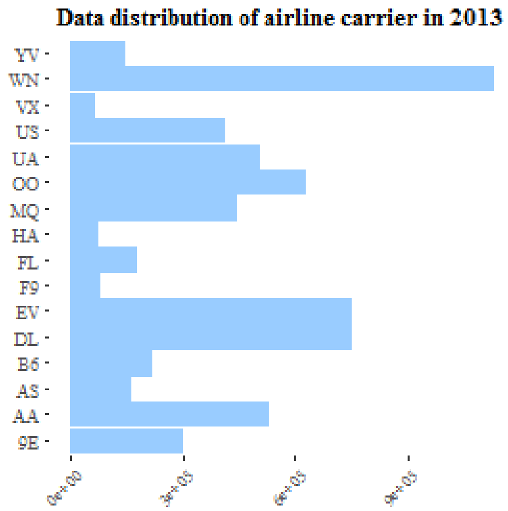

Data distribution of flight distance group in 2010
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

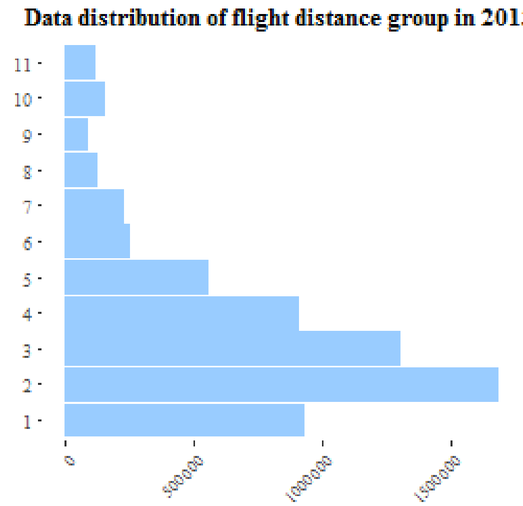

Data distribution of airline carrier in 2011
YV
XE
WN
US
UA
OO
MQ
HA
FL
F9
EV
DL
CO
B6
AS
AA
0
300000
600000
900000
1200000

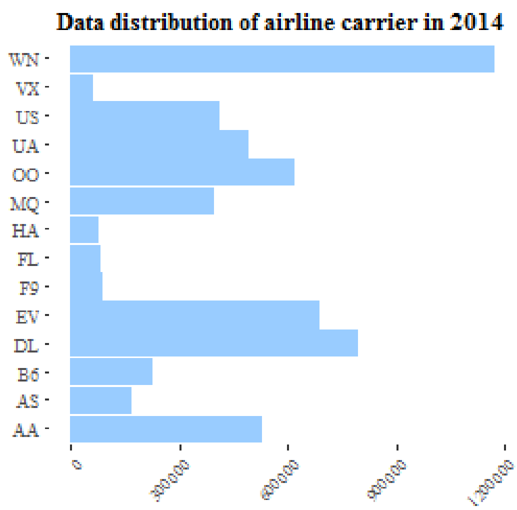

Data distribution of flight distance group in 2011
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

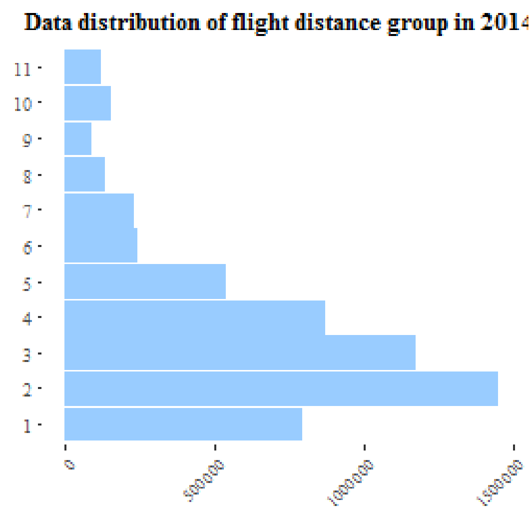

Data distribution of airline carrier in 2012
YV
WN
VX
US
UA
OO
MQ
HA
FL
F9
EV
DL
B6
AS
AA
0e+00
3e+05
6e+05
9e+05

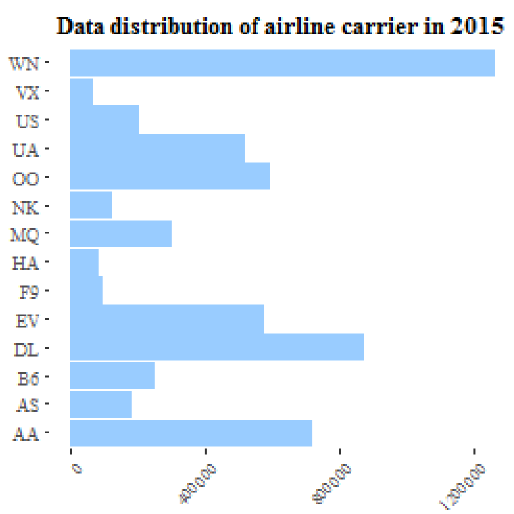

Data distribution of flight distance group in 2012
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

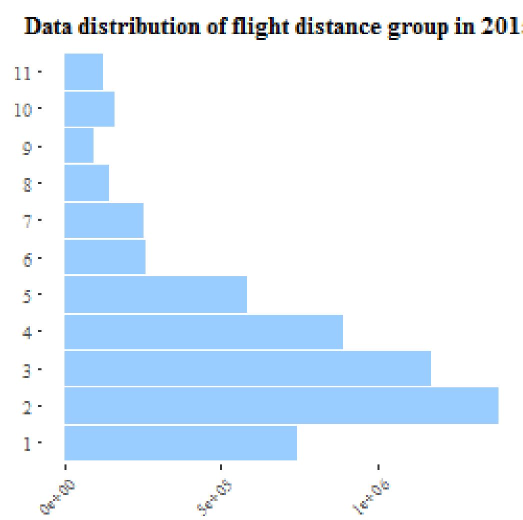

Data distribution of airline carrier in 2013
YV
WN
VX
US
UA
OO
MQ
HA
FL
F9
EV
DL
B6
AS
AA
9E
0e+00
3e+05
6e+05
9e+05

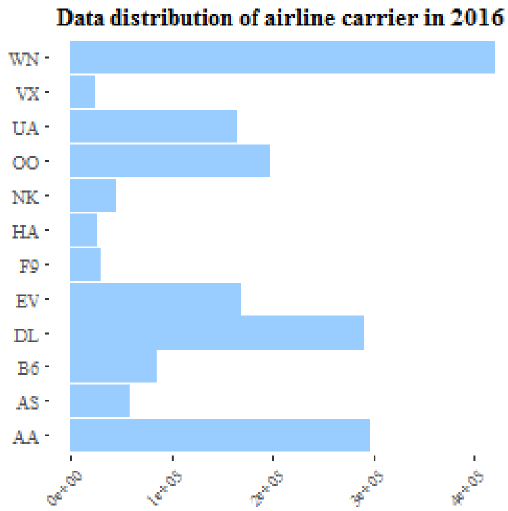

Data distribution of flight distance group in 201
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000

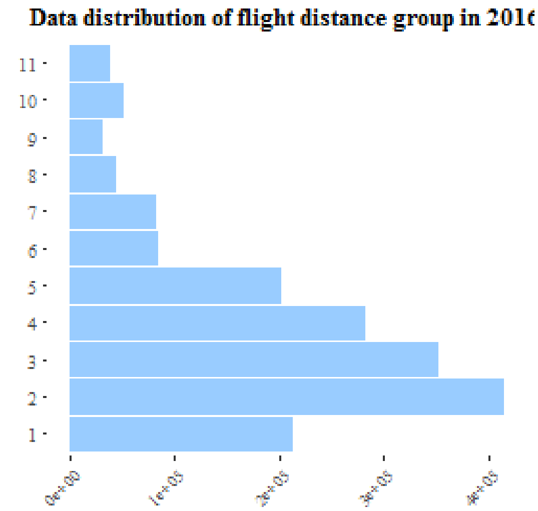

Data distribution of airline carrier in 2014
WN
VX
US
UA
OO
MQ
HA
FL
F9
EV
DL
B6
AS
AA
0
300000
600000
900000
1200000


Data distribution of flight distance group in 201
11
10
9
8
7
6
5
4
3
2
1
0
500000
1000000
1500000


Data distribution of airline carrier in 2015
WN
VX
US
UA
OO
NK
MQ
HA
F9
EV
DL
B6
AS
AA
0
400000
800000
1200000


Data distribution of flight distance group in 201
11
10
9
8
7
6
5
4
3
2
1
0e+00
5e+05
1e+06


Data distribution of airline carrier in 2016
WN
VX
UA
OO
NK
HA
F9
EV
DL
B6
AS
AA
0e+00
1e+05
2e+05
3e+05
4e+05


Data distribution of flight distance group in 2016
11
10
9
8
7
6
5
4
3
2
1
0e+00
1e+05
2e+05
3e+05
4e+05

16 Appendix 7 - Source Code

The following pages contain the source code of the project.

```
# #'
# #' \code{wildLifeStrikeDataSet} based on the configuration
# #' items checks if the wildlife strike data set file has been:
# #'  - downloaded
# #'  - uncompressed
# #'  - included tables extracted
# #' if not, then execute these tasks.
# #'
# #' @examples
# #' wildLifeStrikeDataSet()
# #'
# wildLifeStrikeDataSet <- function() {
#   #setting the download parameters
#   URL <- getWData()
#   destfile <- paste(getDataDir(), "wildlife.zip", sep = "/")
#
#   method="auto"
#
#   #if the file exists then do not download again
#   if (file.exists(destfile) != TRUE)
#   {
#     download.file(URL, destfile, method)
#   } else
#   {
#     message("File exists no download required.")
#   }
#
#   destdir <- getDataDir()
#
#   #unzip the file
#   unzip(destfile, exdir = destdir)
#
#   csvfile <- paste(destdir,
#                    "/STRIKE_REPORTS (1990-1999).csv",
#                    sep="")
#
#   if (file.exists(csvfile) != TRUE)
#   {
#     setwd(getDataDir())
#     system(paste("java -jar ",
#                  getDataDir(),
#                  "/access2csv.jar ",
#                  getDataDir(),
#                  "/wildlife.accdb",
#                  sep = ""))
#     setwd(getMainDir())
#   } else
#   {
#     message("File exists no extract required.")
#   }
```

```
#
# }
# #'
# #' \code{onTimeFlightPerformanceDataSet} based on the configuration
# #' items checkes if the commercial flight data set files have been:
# #'  - downloaded
# #'  - uncompressed
# #' if not, then execute these tasks.
# #'
# #' @examples
# #' onTimeFlightPerformanceDataSet()
# #'
# onTimeFlightPerformanceDataSet <- function() {
#
#   method="auto"
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#   startMonth <- getStartMonth()
#   endMonth <- getEndMonth()
#
#   for (i in startYear:endYear){
#     for (j in startMonth:endMonth){
#
#       variableName <- paste("On_Time_On_Time_Performance_",
#                             i,
#                             "_",
#                             j,
#                             sep = "")
#
#       sourceFile <- paste(variableName, ".zip", sep = "")
#       URL <- paste(getFData(), sourceFile, sep = "")
#       destinationFile <- paste(dataDir, "/", sourceFile, sep = "")
#
#       #if the file exists then do not download again
#       if (file.exists(destinationFile) != TRUE)
#       {
#         message("Downloading ", sourceFile)
#         download.file(URL, destinationFile, method)
#         Sys.sleep(0.1)
#       } else
#       {
#         message(sourceFile,
#                 " file exists, no download is required.")
#       }
#
#       zippedFileName <- sourceFile
#       zippedFile <- destinationFile
#       unzippedFileName <- paste(variableName,
#                                 ".csv",
#                                 sep = "")
#       unzippedFile <- paste(dataDir, "/", unzippedFileName, sep = "")
#
```

```
#       #if the file exists then do not unzip it again
#       if (file.exists(unzippedFile) != TRUE)
#       {
#         message("Unzipping ", zippedFileName)
#         unzip(zippedFile,
#               overwrite = FALSE,
#               exdir = dataDir) #No overwrite
#         #Clear warnings
#         assign("last.warning", NULL, envir = baseenv())
#       } else
#       {
#         message(unzippedFileName,
#                 " file exists, no unzip is required.")
#       }
#
#     } #end of "for (j in startMonth:endMonth)"
#   } #end of "for (i in startYear:endYear)"
#
# }
```

```
# #'
# #' \code{wildLifeStrikeDataSetSplitByYear} splits the strike data
# #' into RDS files by year, so that the data files would be aligned
# #' across the different data sets
# #'
# #' @examples
# #' wildLifeStrikeDataSetSplitByYear()
# #'
# wildLifeStrikeDataSetSplitByYear <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   for (i in startYear:endYear){
#     RDSFileName <- paste(i,
#                          "_Animal_Strikes_01_Orig.rds",
#                          sep = "")
#
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#
#     if (file.exists(RDSFile) != TRUE){
#
#       if (exists("sr_1990_1999") != TRUE){
#
#         message("Reading sr_1990_1999")
#
#         variableName <- "sr_1990_1999"
#
#         assign(variableName,
#                data.table(
```

```
#                  read.csv(
#                    paste(
#                      dataDir,
#                      "/STRIKE_REPORTS (1990-1999).csv",
#                      sep=""),
#                    header = FALSE)),
#                envir = .GlobalEnv)
#
#         names(sr_1990_1999) <- c("INDEX_NR",
#                                  "OPID",
#                                  "OPERATOR",
#                                  "ATYPE",
#                                  "AMA",
#                                  "AMO",
#                                  "EMA",
#                                  "EMO",
#                                  "AC_CLASS",
#                                  "AC_MASS",
#                                  "NUM_ENGS",
#                                  "TYPE_ENG",
#                                  "ENG_1_POS",
#                                  "ENG_2_POS",
#                                  "ENG_3_POS",
#                                  "ENG_4_POS",
#                                  "REG",
#                                  "FLT",
#                                  "REMAINS_COLLECTED",
#                                  "REMAINS_SENT",
#                                  "INCIDENT_DATE",
#                                  "INCIDENT_MONTH",
#                                  "INCIDENT_YEAR",
#                                  "TIME_OF_DAY",
#                                  "TIME",
#                                  "AIRPORT_ID",
#                                  "AIRPORT",
#                                  "STATE",
#                                  "FAAREGION",
#                                  "ENROUTE",
#                                  "RUNWAY",
#                                  "LOCATION",
#                                  "HEIGHT",
#                                  "SPEED",
#                                  "DISTANCE",
#                                  "PHASE_OF_FLT",
#                                  "DAMAGE",
#                                  "STR_RAD",
#                                  "DAM_RAD",
#                                  "STR_WINDSHLD",
#                                  "DAM_WINDSHLD",
#                                  "STR_NOSE",
#                                  "DAM_NOSE",
#                                  "STR_ENG1",
#                                  "DAM_ENG1",
#                                  "STR_ENG2",
```

```
#                                    ”DAM_ENG2”,
#                                    ”STR_ENG3”,
#                                    ”DAM_ENG3”,
#                                    ”STR_ENG4”,
#                                    ”DAM_ENG4”,
#                                    ”INGESTED”,
#                                    ”STR_PROP”,
#                                    ”DAM_PROP”,
#                                    ”STR_WING_ROT”,
#                                    ”DAM_WING_ROT”,
#                                    ”STR_FUSE”,
#                                    ”DAM_FUSE”,
#                                    ”STR_LG”,
#                                    ”DAM_LG”,
#                                    ”STR_TAIL”,
#                                    ”DAM_TAIL”,
#                                    ”STR_LGHTS”,
#                                    ”DAM_LGHTS”,
#                                    ”STR_OTHER”,
#                                    ”DAM_OTHER”,
#                                    ”OTHER_SPECIFY”,
#                                    ”EFFECT”,
#                                    ”EFFECT_OTHER”,
#                                    ”SKY”,
#                                    ”PRECIP”,
#                                    ”SPECIES_ID”,
#                                    ”SPECIES”,
#                                    ”BIRDS_SEEN”,
#                                    ”BIRDS_STRUCK”,
#                                    ”SIZE”,
#                                    ”WARNED”,
#                                    ”COMMENTS”,
#                                    ”REMARKS”,
#                                    ”AOS”,
#                                    ”COST_REPAIRS”,
#                                    ”COST_OTHER”,
#                                    ”COST_REPAIRS_INFL_ADJ”,
#                                    ”COST_OTHER_INFL_ADJ”,
#                                    ”REPORTED_NAME”,
#                                    ”REPORTED_TITLE”,
#                                    ”REPORTED_DATE”,
#                                    ”SOURCE”,
#                                    ”PERSON”,
#                                    ”NR_INJURIES”,
#                                    ”NR_FATALITIES”,
#                                    ”LUPDATE”,
#                                    ”TRANSFER”,
#                                    ”INDICATED_DAMAGE”)
#        }
#
#        if (exists(”sr_2000_2009”) != TRUE){
#
#          message(”Reading sr_2000_2009”)
#
```

```
#         variableName <- "sr_2000_2009"
#
#         assign(variableName,
#                data.table(
#                  read.csv(
#                    paste(
#                      dataDir,
#                      "/STRIKE_REPORTS (2000-2009).csv",
#                      sep=""),
#                    header = FALSE)),
#                envir = .GlobalEnv)
#
#         names(sr_2000_2009) <- c("INDEX_NR",
#                                  "OPID",
#                                  "OPERATOR",
#                                  "ATYPE",
#                                  "AMA",
#                                  "AMO",
#                                  "EMA",
#                                  "EMO",
#                                  "AC_CLASS",
#                                  "AC_MASS",
#                                  "NUM_ENGS",
#                                  "TYPE_ENG",
#                                  "ENG_1_POS",
#                                  "ENG_2_POS",
#                                  "ENG_3_POS",
#                                  "ENG_4_POS",
#                                  "REG",
#                                  "FLT",
#                                  "REMAINS_COLLECTED",
#                                  "REMAINS_SENT",
#                                  "INCIDENT_DATE",
#                                  "INCIDENT_MONTH",
#                                  "INCIDENT_YEAR",
#                                  "TIME_OF_DAY",
#                                  "TIME",
#                                  "AIRPORT_ID",
#                                  "AIRPORT",
#                                  "STATE",
#                                  "FAAREGION",
#                                  "ENROUTE",
#                                  "RUNWAY",
#                                  "LOCATION",
#                                  "HEIGHT",
#                                  "SPEED",
#                                  "DISTANCE",
#                                  "PHASE_OF_FLT",
#                                  "DAMAGE",
#                                  "STR_RAD",
#                                  "DAM_RAD",
#                                  "STR_WINDSHLD",
#                                  "DAM_WINDSHLD",
#                                  "STR_NOSE",
```

```
#                                        ”DAM_NOSE”,
#                                        ”STR_ENG1”,
#                                        ”DAM_ENG1”,
#                                        ”STR_ENG2”,
#                                        ”DAM_ENG2”,
#                                        ”STR_ENG3”,
#                                        ”DAM_ENG3”,
#                                        ”STR_ENG4”,
#                                        ”DAM_ENG4”,
#                                        ”INGESTED”,
#                                        ”STR_PROP”,
#                                        ”DAM_PROP”,
#                                        ”STR_WING_ROT”,
#                                        ”DAM_WING_ROT”,
#                                        ”STR_FUSE”,
#                                        ”DAM_FUSE”,
#                                        ”STR_LG”,
#                                        ”DAM_LG”,
#                                        ”STR_TAIL”,
#                                        ”DAM_TAIL”,
#                                        ”STR_LGHTS”,
#                                        ”DAM_LGHTS”,
#                                        ”STR_OTHER”,
#                                        ”DAM_OTHER”,
#                                        ”OTHER_SPECIFY”,
#                                        ”EFFECT”,
#                                        ”EFFECT_OTHER”,
#                                        ”SKY”,
#                                        ”PRECIP”,
#                                        ”SPECIES_ID”,
#                                        ”SPECIES”,
#                                        ”BIRDS_SEEN”,
#                                        ”BIRDS_STRUCK”,
#                                        ”SIZE”,
#                                        ”WARNED”,
#                                        ”COMMENTS”,
#                                        ”REMARKS”,
#                                        ”AOS”,
#                                        ”COST_REPAIRS”,
#                                        ”COST_OTHER”,
#                                        ”COST_REPAIRS_INFL_ADJ”,
#                                        ”COST_OTHER_INFL_ADJ”,
#                                        ”REPORTED_NAME”,
#                                        ”REPORTED_TITLE”,
#                                        ”REPORTED_DATE”,
#                                        ”SOURCE”,
#                                        ”PERSON”,
#                                        ”NR_INJURIES”,
#                                        ”NR_FATALITIES”,
#                                        ”LUPDATE”,
#                                        ”TRANSFER”,
#                                        ”INDICATED_DAMAGE”)
#         }
#
```

```
#       if (exists(”sr_2010_Current”) != TRUE){
#
#         message(”Reading sr_2010_Current”)
#
#         variableName <- ”sr_2010_Current”
#
#         assign(variableName,
#                data.table(
#                  read.csv(
#                    paste(
#                      dataDir,
#                      ”/STRIKE_REPORTS (2010-Current).csv”,
#                      sep=””),
#                    header = FALSE)),
#                envir = .GlobalEnv)
#
#         names(sr_2010_Current) <- c(”INDEX_NR”,
#                                     ”OPID”,
#                                     ”OPERATOR”,
#                                     ”ATYPE”,
#                                     ”AMA”,
#                                     ”AMO”,
#                                     ”EMA”,
#                                     ”EMO”,
#                                     ”AC_CLASS”,
#                                     ”AC_MASS”,
#                                     ”NUM_ENGS”,
#                                     ”TYPE_ENG”,
#                                     ”ENG_1_POS”,
#                                     ”ENG_2_POS”,
#                                     ”ENG_3_POS”,
#                                     ”ENG_4_POS”,
#                                     ”REG”,
#                                     ”FLT”,
#                                     ”REMAINS_COLLECTED”,
#                                     ”REMAINS_SENT”,
#                                     ”INCIDENT_DATE”,
#                                     ”INCIDENT_MONTH”,
#                                     ”INCIDENT_YEAR”,
#                                     ”TIME_OF_DAY”,
#                                     ”TIME”,
#                                     ”AIRPORT_ID”,
#                                     ”AIRPORT”,
#                                     ”STATE”,
#                                     ”FAAREGION”,
#                                     ”ENROUTE”,
#                                     ”RUNWAY”,
#                                     ”LOCATION”,
#                                     ”HEIGHT”,
#                                     ”SPEED”,
#                                     ”DISTANCE”,
#                                     ”PHASE_OF_FLT”,
#                                     ”DAMAGE”,
#                                     ”STR_RAD”,
```

```
#                                              ”DAM_RAD”,
#                                              ”STR_WINDSHLD”,
#                                              ”DAM_WINDSHLD”,
#                                              ”STR_NOSE”,
#                                              ”DAM_NOSE”,
#                                              ”STR_ENG1”,
#                                              ”DAM_ENG1”,
#                                              ”STR_ENG2”,
#                                              ”DAM_ENG2”,
#                                              ”STR_ENG3”,
#                                              ”DAM_ENG3”,
#                                              ”STR_ENG4”,
#                                              ”DAM_ENG4”,
#                                              ”INGESTED”,
#                                              ”STR_PROP”,
#                                              ”DAM_PROP”,
#                                              ”STR_WING_ROT”,
#                                              ”DAM_WING_ROT”,
#                                              ”STR_FUSE”,
#                                              ”DAM_FUSE”,
#                                              ”STR_LG”,
#                                              ”DAM_LG”,
#                                              ”STR_TAIL”,
#                                              ”DAM_TAIL”,
#                                              ”STR_LGHTS”,
#                                              ”DAM_LGHTS”,
#                                              ”STR_OTHER”,
#                                              ”DAM_OTHER”,
#                                              ”OTHER_SPECIFY”,
#                                              ”EFFECT”,
#                                              ”EFFECT_OTHER”,
#                                              ”SKY”,
#                                              ”PRECIP”,
#                                              ”SPECIES_ID”,
#                                              ”SPECIES”,
#                                              ”BIRDS_SEEN”,
#                                              ”BIRDS_STRUCK”,
#                                              ”SIZE”,
#                                              ”WARNED”,
#                                              ”COMMENTS”,
#                                              ”REMARKS”,
#                                              ”AOS”,
#                                              ”COST_REPAIRS”,
#                                              ”COST_OTHER”,
#                                              ”COST_REPAIRS_INFL_ADJ”,
#                                              ”COST_OTHER_INFL_ADJ”,
#                                              ”REPORTED_NAME”,
#                                              ”REPORTED_TITLE”,
#                                              ”REPORTED_DATE”,
#                                              ”SOURCE”,
#                                              ”PERSON”,
#                                              ”NR_INJURIES”,
#                                              ”NR_FATALITIES”,
#                                              ”LUPDATE”,
```

```
#                                   ”TRANSFER”,
#                                   ”INDICATED_DAMAGE”)
#       }
#
#       #STRIKE_REPORTS_BASH --> contains only military data, not required
#
#
#       if (i >= 1990 && i <= 1999) {
#         dataOfWholeYear <- sr_1990_1999[INCIDENT_YEAR == i]
#       }
#       else if (i >= 2000 && i <= 2009) {
#         dataOfWholeYear <- sr_2000_2009[INCIDENT_YEAR == i]
#       }
#       else if (i >= 2010 && i <= 2019) {
#         dataOfWholeYear <- sr_2010_Current[INCIDENT_YEAR == i]
#       }
#
#       saveRDS(dataOfWholeYear, file = RDSFile)
#       message(RDSFileName,” created.”)
#
#       #free up memory
#       rm(dataOfWholeYear)
#       rm(list = ls(pattern = ”sr_*”))
#       gc()
#
#     }
#     else {
#       message(RDSFileName,
#               ” exists, no further action is required.”)
#     }
#
#
#   } #end of ”for (i in startYear:endYear)”
#
# }
# #'
# #' \code{onTimeFlightPerformanceDataSetMergeByYear} merges the flight data
# #' into RDS files by year, so that working with the data would not consume
# #' all the memory of the running environment
# #'
# #' @examples
# #' onTimeFlightPerformanceDataSetMergeByYear()
# #'
# onTimeFlightPerformanceDataSetMergeByYear <- function() {
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#   startMonth <- getStartMonth()
#   endMonth <- getEndMonth()
#
#   for (i in startYear:endYear){
#
#     RDSFileName <- paste(i,
```

```
#                          ”_On_Time_On_Time_Performance_01_Orig.rds”,
#                          sep = ””)
#
#      RDSFile <- paste(dataDir,
#                      ”/”,
#                      RDSFileName,
#                      sep = ””)
#
#      #Create the RDS files only if they do not exist yet
#      if (file.exists(RDSFile) != TRUE){
#
#        for (j in startMonth:endMonth){
#
#          variableName <- paste(”On_Time_On_Time_Performance_”,
#                               i,
#                               ”_”,
#                               j,
#                               sep = ””)
#
#          unzippedFileName <- paste(variableName,
#                                   ”.csv”,
#                                   sep = ””)
#
#          unzippedFile <- paste(dataDir,
#                               ”/”,
#                               unzippedFileName,
#                               sep = ””)
#
#          assign(variableName,
#                 data.table(read.csv(unzippedFile,
#                                     header = TRUE)))
#
#          if (j == startMonth){
#            dataOfWholeYear <- get(variableName)
#            rm(list = ls(pattern = ”On_Time_On_Time_Performance*”))
#            gc()
#          }
#          else {
#            dataOfWholeYear <- rbindlist(list(dataOfWholeYear,
#                                              get(variableName)))
#            rm(list = ls(pattern = ”On_Time_On_Time_Performance*”))
#            gc()
#          }
#
#        } #end of ”for (j in startMonth:endMonth)”
#
#     dataOfWholeYear$DistanceGroup <- as.factor(dataOfWholeYear$DistanceGroup)
#
#        saveRDS(dataOfWholeYear, file = RDSFile)
#        message(RDSFileName,” created.”)
#
#        #free up memory
#        rm(dataOfWholeYear)
#        gc()
```

```
#
#     }
#     else {
#       message(RDSFileName,
#               " exists, no further action is required.")
#     }
#
#   } #end of "for (i in startYear:endYear)"
#
# }
# #'
# #' \code{ExploreWildLifeStrikeDataSet} creates the inputs
# #' for the Data Exploration Report based on the WildLife
# #' strike data set
# #'
# #' @param createPNG boolean
# #' Flag to decide to create the PNG images or not
# #'
# #' @examples
# #' ExploreWildLifeStrikeDataSet(FALSE)
# #'
# ExploreWildLifeStrikeDataSet <- function(createPNG) {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   dataSummary <- data.table(
#     dataYear = character(),
#     numberOfRecords = integer(),
#     factorOPID = integer(),
#     factorATYPE = integer(),
#     factorAC_CLASS = integer(),
#     factorAC_MASS = integer(),
#     factorTYPE_ENG = integer(),
#     factorTIME_OF_DAY = integer(),
#     factorAIRPORT_ID = integer(),
#     factorSTATE = integer(),
#     factorPHASE_OF_FLT = integer(),
#     factorSKY = integer(),
#     factorPRECIP = integer(),
#     factorWARNED = integer()
#   )
#
#   dataSummaryState <- data.table(
#     state = character()
#   )
#
#   for (i in startYear:endYear){
#     RDSFileName <- paste(i,
#                          "_Animal_Strikes_01_Orig.rds",
#                          sep = "")
#
```

```
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#
#     if (file.exists(RDSFile) != TRUE){
#       message(RDSFileName,
#               "is not available, ",
#               "please re-run the preparation scripts!")
#     } else {
#       #Read the data file into a variable
#       variableName <- paste("AS_", i, sep="")
#       assign(variableName, readRDS(file = RDSFile))
#
#       dataSummary <- rbindlist(
#         list(
#           dataSummary,
#           list(
#             as.character(i),
#             nrow(get(variableName)),
#             length(levels(get(variableName)$OPID)),
#             length(levels(get(variableName)$ATYPE)),
#             length(levels(get(variableName)$AC_CLASS)),
#             length(levels(as.factor(get(variableName)$AC_MASS))),
#             length(levels(get(variableName)$TYPE_ENG)),
#             length(levels(get(variableName)$TIME_OF_DAY)),
#             length(levels(get(variableName)$AIRPORT_ID)),
#             length(levels(get(variableName)$STATE)),
#             length(levels(get(variableName)$PHASE_OF_FLT)),
#             length(levels(get(variableName)$SKY)),
#             length(levels(get(variableName)$PRECIP)),
#             length(levels(get(variableName)$WARNED))
#             )
#           )
#         )
#
#       dataSummaryState <- rbindlist(
#         list(
#           dataSummaryState,
#           unique(get(variableName)[,"STATE"], by = c("STATE"))
#         )
#       )
#
#       if (createPNG == TRUE) {
#         #Save the plots as PNG files
#         saveBarPlotPNG(DataYear = i,
#                        DataSet = "AnimalStrike",
#                        DataField = "AC_CLASS",
#                        DataStage = "01_Orig",
#                        DataObject = get(variableName))
#         saveBarPlotPNG(DataYear = i,
#                        DataSet = "AnimalStrike",
#                        DataField = "AC_MASS",
#                        DataStage = "01_Orig",
```

```
#                         DataObject = get(variableName))
#         saveBarPlotPNG(DataYear = i,
#                         DataSet = ”AnimalStrike”,
#                         DataField = ”TYPE_ENG”,
#                         DataStage = ”01_Orig”,
#                         DataObject = get(variableName))
#         saveBarPlotPNG(DataYear = i,
#                         DataSet = ”AnimalStrike”,
#                         DataField = ”TIME_OF_DAY”,
#                         DataStage = ”01_Orig”,
#                         DataObject = get(variableName))
#         saveBarPlotPNG(DataYear = i,
#                         DataSet = ”AnimalStrike”,
#                         DataField = ”PHASE_OF_FLT”,
#                         DataStage = ”01_Orig”,
#                         DataObject = get(variableName))
#         saveBarPlotPNG(DataYear = i,
#                         DataSet = ”AnimalStrike”,
#                         DataField = ”SKY”,
#                         DataStage = ”01_Orig”,
#                         DataObject = get(variableName))
#         saveBarPlotPNG(DataYear = i,
#                         DataSet = ”AnimalStrike”,
#                         DataField = ”PRECIP”,
#                         DataStage = ”01_Orig”,
#                         DataObject = get(variableName))
#       }
#
#       #Free up the memory
#       rm(list = variableName)
#       rm(variableName)
#       gc()
#
#     } #end of ”if (file.exists(RDSFile) != TRUE)”
#
#   } #end of ”for (i in startYear:endYear)”
#
#
#   RDSExpFileName <- ”01_EXP_Animal_Strikes.rds”
#
#   RDSExpFile <- paste(dataDir,
#                        ”/”,
#                        RDSExpFileName,
#                        sep = ””)
#
#   if (file.exists(RDSExpFile) != TRUE) {
#     saveRDS(dataSummary, file = RDSExpFile)
#   } else {
#     file.remove(RDSExpFile)
#     saveRDS(dataSummary, file = RDSExpFile)
#   }
#
#   RDSExpStateFileName <- ”01_EXP_Animal_Strikes_States.rds”
#
```

```
#   RDSExpStateFile <- paste(dataDir,
#                        "/",
#                        RDSExpStateFileName,
#                        sep = "")
#
#   dataSummaryStateFinal <-
#     unique(dataSummaryState[,"state"], by = c("state"))
#
#   #dataSummaryStateFinal <- dataSummaryStateFinal[order(state)]
#
#   if (file.exists(RDSExpStateFile) != TRUE) {
#     saveRDS(dataSummaryStateFinal, file = RDSExpStateFile)
#   } else {
#     file.remove(RDSExpStateFile)
#     saveRDS(dataSummaryStateFinal, file = RDSExpStateFile)
#   }
#
#
# }
# #'
# #' \code{ExploreOnTimeFlightPerformanceDataSet} creates
# #' the inputs for the Data Exploration Report based on
# #' the Flight data set
# #'
# #' @param createPNG boolean
# #' Flag to decide to create the PNG images or not
# #'
# #' @examples
# #' ExploreOnTimeFlightPerformanceDataSet(FALSE)
# #'
# ExploreOnTimeFlightPerformanceDataSet <- function(createPNG) {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   dataSummary <- data.table(
#     dataYear = character(),
#     numberOfRecords = integer(),
#     factorCarrier = integer(),
#     factorOrigin = integer(),
#     factorOriginState = integer(),
#     factorDest = integer(),
#     factorDestState = integer(),
#     factorDepTimeBlk = integer(),
#     factorDistanceGroup = integer()
#   )
#
#   dataSummaryOriginState <- data.table(
#     originState = character(),
#     originStateName = character()
#   )
#
```

```
#   dataSummaryDestState <- data.table(
#     destState = character(),
#     destStateName = character()
#   )
#
#   for (i in startYear:endYear){
#     RDSFileName <- paste(i,
#                          "_On_Time_On_Time_Performance_01_Orig.rds",
#                          sep = "")
#
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#
#     if (file.exists(RDSFile) != TRUE){
#       message(RDSFileName,
#               "is not available, ",
#               "please re-run the preparation scripts!")
#     } else {
#       #Read the data file into a variable
#       variableName <- paste("FP_", i, sep="")
#       assign(variableName, readRDS(file = RDSFile))
#
#       dataSummary <- rbindlist(
#         list(
#           dataSummary,
#           list(as.character(i),
#                nrow(get(variableName)),
#                length(levels(get(variableName)$Carrier)),
#                length(levels(get(variableName)$Origin)),
#                length(levels(get(variableName)$OriginState)),
#                length(levels(get(variableName)$Dest)),
#                length(levels(get(variableName)$DestState)),
#                length(levels(get(variableName)$DepTimeBlk)),
#                length(levels(as.factor(get(variableName)$DistanceGroup)))
#                )
#           )
#         )
#
#       dataSummaryOriginState <- rbindlist(
#         list(
#           dataSummaryOriginState,
#           unique(get(variableName)[,c("OriginState",
#                                       "OriginStateName")],
#                  by = c("OriginState",
#                         "OriginStateName"))
#           )
#         )
#
#       dataSummaryDestState <- rbindlist(
#         list(
#           dataSummaryDestState,
#           unique(get(variableName)[,c("DestState",
```

```
#                                       ”DestStateName”)],
#                   by = c(”DestState”,
#                         ”DestStateName”))
#           )
#         )
#
#       if (createPNG == TRUE) {
#         #Save the plots as PNG files
#         saveBarPlotPNG(DataYear = i,
#                        DataSet = ”FlightData”,
#                        DataField = ”Carrier”,
#                        DataStage = ”01_Orig”,
#                        DataObject = get(variableName))
#
#         saveBarPlotPNG(DataYear = i,
#                        DataSet = ”FlightData”,
#                        DataField = ”DistanceGroup”,
#                        DataStage = ”01_Orig”,
#                        DataObject = get(variableName))
#       }
#
#       #Free up the memory
#       rm(list = variableName)
#       rm(variableName)
#       gc()
#
#     } #end of ”if (file.exists(RDSFile) != TRUE)”
#
#   } #end of ”for (i in startYear:endYear)”
#
#   RDSExpFileName <- ”02_EXP_Flight_Data.rds”
#
#   RDSExpFile <- paste(dataDir,
#                       ”/”,
#                       RDSExpFileName,
#                       sep = ””)
#
#   if (file.exists(RDSExpFile) != TRUE) {
#     saveRDS(dataSummary, file = RDSExpFile)
#   } else {
#     file.remove(RDSExpFile)
#     saveRDS(dataSummary, file = RDSExpFile)
#   }
#
#   RDSExpStateFileName <- ”02_EXP_Flight_Data_O_States.rds”
#
#   RDSExpStateFile <- paste(dataDir,
#                            ”/”,
#                            RDSExpStateFileName,
#                            sep = ””)
#
#   dataSummaryOriginState <-
#     unique(dataSummaryOriginState[,c(”originState”,
#                                      ”originStateName”)],
```

```
#             by = c("originState",
#                    "originStateName"))
#
#   dataSummaryOriginState <- dataSummaryOriginState[order(originState)]
#
#   if (file.exists(RDSExpStateFile) != TRUE) {
#     saveRDS(dataSummaryOriginState,
#             file = RDSExpStateFile)
#   } else {
#     file.remove(RDSExpStateFile)
#     saveRDS(dataSummaryOriginState,
#             file = RDSExpStateFile)
#   }
#
#
#   RDSExpStateFileName <- "02_EXP_Flight_Data_D_States.rds"
#
#   RDSExpStateFile <- paste(dataDir,
#                            "/",
#                            RDSExpStateFileName,
#                            sep = "")
#
#   dataSummaryDestState <-
#     unique(dataSummaryDestState[,c("destState",
#                                    "destStateName")],
#             by = c("destState",
#                    "destStateName"))
#
#   dataSummaryDestState <- dataSummaryDestState[order(destState)]
#
#   if (file.exists(RDSExpStateFile) != TRUE) {
#     saveRDS(dataSummaryDestState,
#             file = RDSExpStateFile)
#   } else {
#     file.remove(RDSExpStateFile)
#     saveRDS(dataSummaryDestState,
#             file = RDSExpStateFile)
#   }
#
# }

# #'
# #' \code{DescribeWildLifeStrikeDataSet} re-creates the
# #' inputs based on the column selection of the data
# #' verification report
# #'
# #' @examples
# #' DescribeWildLifeStrikeDataSet()
# #'
# DescribeWildLifeStrikeDataSet <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
```

```
#
#    for (i in startYear:endYear){
#      RDSFileName <- paste(i,
#                           ”_Animal_Strikes_01_Orig.rds”,
#                           sep = ””)
#
#      RDSFile <- paste(dataDir,
#                       ”/”,
#                       RDSFileName,
#                       sep = ””)
#
#      RDSFileNameDescibed <- paste(i,
#                                   ”_Animal_Strikes_02_Desc.rds”,
#                                   sep = ””)
#
#      RDSFileDescibed <- paste(dataDir,
#                               ”/”,
#                               RDSFileNameDescibed,
#                               sep = ””)
#
#      if (file.exists(RDSFile) != TRUE){
#        message(RDSFileName,
#                ”is not available, ”,
#                ”please re-run the preparation scripts!”)
#      } else {
#
#        if (file.exists(RDSFileDescibed) == TRUE){
#          message(RDSFileNameDescibed,
#                  ” exists, no further action is required.”)
#        } else {
#
#          #Read the data file into a variable
#          variableName <- paste(”AS_”, i, sep=””)
#          assign(variableName, readRDS(file = RDSFile))
#
#          #set the required column names
#          ColumnNames <- c(”INDEX_NR”,
#                           ”OPID”,
#                           ”OPERATOR”,
#                           ”ATYPE”,
#                           ”AC_CLASS”,
#                           ”AC_MASS”,
#                           ”TYPE_ENG”,
#                           ”REG”,
#                           ”FLT”,
#                           ”INCIDENT_DATE”,
#                           ”INCIDENT_MONTH”,
#                           ”INCIDENT_YEAR”,
#                           ”TIME_OF_DAY”,
#                           ”TIME”,
#                           ”AIRPORT_ID”,
#                           ”AIRPORT”,
#                           ”STATE”,
#                           ”FAAREGION”,
```

```
#                           ”ENROUTE”,
#                           ”RUNWAY”,
#                           ”HEIGHT”,
#                           ”SPEED”,
#                           ”DISTANCE”,
#                           ”PHASE_OF_FLT”,
#                           ”SKY”,
#                           ”PRECIP”,
#                           ”WARNED”)
#
#         #Move reduces data into a new data set
#         describedDataSet <- get(variableName)[, ..ColumnNames]
#
#         saveRDS(describedDataSet, file = RDSFileDescibed)
#
#         #Free up the memory
#         rm(list = variableName)
#         rm(variableName)
#         rm(describedDataSet)
#         gc()
#
#       } #end of ”if (file.exists(RDSFileDescibed) == TRUE)”
#
#     } #end of ”if (file.exists(RDSFile) != TRUE)”
#
#   } #end of ”for (i in startYear:endYear)”
#
# }
# #'
# #' \code{DescribeOnTimeFlightPerformanceDataSet} re-creates the
# #' inputs based on the column selection of the data
# #' verification report
# #'
# #' @examples
# #' DescribeOnTimeFlightPerformanceDataSet()
# #'
# DescribeOnTimeFlightPerformanceDataSet <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   for (i in startYear:endYear){
#     RDSFileName <- paste(i,
#                          ”_On_Time_On_Time_Performance_01_Orig.rds”,
#                          sep = ””)
#
#     RDSFile <- paste(dataDir,
#                      ”/”,
#                      RDSFileName,
#                      sep = ””)
#
#     RDSFileNameDescibed <- paste(i,
```

```
#                                ”_On_Time_On_Time_Performance_02_Desc.rds”,
#                                 sep = ””)
#
#     RDSFileDescibed <- paste(dataDir,
#                             ”/”,
#                             RDSFileNameDescibed,
#                             sep = ””)
#
#     if (file.exists(RDSFile) != TRUE){
#       message(RDSFileName,
#               ” is not available, ”,
#               ”please re-run the preparation scripts!”)
#     } else {
#
#       if (file.exists(RDSFileDescibed) == TRUE){
#         message(RDSFileNameDescibed,
#                 ” exists, no further action is required.”)
#       } else {
#
#         #Read the data file into a variable
#         variableName <- paste(”FP_”, i, sep=””)
#         assign(variableName, readRDS(file = RDSFile))
#
#         #set the required column names
#         ColumnNames <- c(”Year”,
#                          ”Quarter”,
#                          ”Month”,
#                          ”DayofMonth”,
#                          ”DayOfWeek”,
#                          ”FlightDate”,
#                          ”Carrier”,
#                          ”UniqueCarrier”,
#                          ”FlightNum”,
#                          ”Origin”,
#                          ”OriginCityName”,
#                          ”OriginState”,
#                          ”OriginStateName”,
#                          ”Dest”,
#                          ”DestCityName”,
#                          ”DestState”,
#                          ”DestStateName”,
#                          ”CRSDepTime”,
#                          ”DepTimeBlk”,
#                          ”CRSArrTime”,
#                          ”ArrTimeBlk”,
#                          ”CRSElapsedTime”,
#                          ”Distance”,
#                          ”DistanceGroup”)
#
#         #Move reduces data into a new data set
#         describedDataSet <- get(variableName)[, ..ColumnNames]
#
#
#         saveRDS(describedDataSet, file = RDSFileDescibed)
```

```
#
#         #Free up the memory
#         rm(list = variableName)
#         rm(variableName)
#         rm(describedDataSet)
#         gc()
#
#       } #end of "if (file.exists(RDSFileDescibed) == TRUE)"
#
#     } #end of "if (file.exists(RDSFile) != TRUE)"
#
#   } #end of "for (i in startYear:endYear)"
#
# }
# #'
# #' \code{SelectWildLifeStrikeDataSet} executes the identified
# #' exclusions and inclusions in the data set validation report
# #'
# #' @examples
# #' SelectWildLifeStrikeDataSet()
# #'
# SelectWildLifeStrikeDataSet <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   for (i in startYear:endYear){
#     RDSFileName <- paste(i,
#                          "_Animal_Strikes_02_Desc.rds",
#                          sep = "")
#
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#
#     RDSFileNameSelected <- paste(i,
#                                  "_Animal_Strikes_03_Sel.rds",
#                                  sep = "")
#
#     RDSFileSelected <- paste(dataDir,
#                              "/",
#                              RDSFileNameSelected,
#                              sep = "")
#
#     if (file.exists(RDSFile) != TRUE){
#       message(RDSFileName,
#               "is not available, ",
#               "please re-run the preparation scripts!")
#     } else {
#
#
```

```
#       if (file.exists(RDSFileSelected) == TRUE){
#         message(RDSFileNameSelected,
#                 " exists, no further action is required.")
#       } else {
#
#         #Read the data file into a variable
#         variableName <- paste("AS_", i, sep="")
#         assign(variableName, readRDS(file = RDSFile))
#
#         #OPID column selection
#         selectedDataSet <- get(variableName)[!OPID %in% c("PVT",
#                                                            "BUS",
#                                                            "GOV",
#                                                            "MIL",
#                                                            "UNKC",
#                                                            "UNK"),]
#
#         #AC_CLASS selection
#         selectedDataSet <- selectedDataSet[!AC_CLASS %in% c("B",
#                                                              "C",
#                                                              "D",
#                                                              "F",
#                                                              "I",
#                                                              "J",
#                                                              "Y",
#                                                              "Z",
#                                                              ""),]
#         #TYPE_ENG selection
#         selectedDataSet <- selectedDataSet[!TYPE_ENG %in% c("E",
#                                                              "F",
#                                                              ""),]
#
#         #STATE selection
#         selectedDataSet <- selectedDataSet[STATE %in% getStates(),]
#
#         #AC_MASS resetting
#         selectedDataSet$AC_MASS <- as.factor(selectedDataSet$AC_MASS)
#
#         #Resetting the factors of the data table
#         selectedDataSet[] <-
#           lapply(selectedDataSet,
#                  function(x) if(is.factor(x)) factor(x) else x)
#
#         saveRDS(selectedDataSet, file = RDSFileSelected)
#
#         #Free up the memory
#         rm(list = variableName)
#         rm(variableName)
#         gc()
#
#       } #end of "if (file.exists(RDSFileSelected) == TRUE)"
#
#     } #end of "if (file.exists(RDSFile) != TRUE)"
#
```

```
#   } #end of "for (i in startYear:endYear)"
#
# }

# #'
# #' \code{SelectOnTimeFlightPerformanceDataSet} executes the identified
# #' exclusions and inclusions in the data set validation report
# #'
# #' @examples
# #' SelectOnTimeFlightPerformanceDataSet()
# #'
# SelectOnTimeFlightPerformanceDataSet <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   for (i in startYear:endYear){
#
#     RDSFileName <- paste(i,
#                          "_On_Time_On_Time_Performance_02_Desc.rds",
#                          sep = "")
#
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#
#     RDSFileNameSelected <- paste(i,
#                                  "_On_Time_On_Time_Performance_03_Sel.rds",
#                                  sep = "")
#
#     RDSFileSelected <- paste(dataDir,
#                              "/",
#                              RDSFileNameSelected,
#                              sep = "")
#
#
#     if (file.exists(RDSFile) != TRUE){
#       message(RDSFileName,
#               "is not available, ",
#               "please re-run the preparation scripts!")
#     } else {
#
#       if (file.exists(RDSFileSelected) == TRUE){
#         message(RDSFileNameSelected,
#                 " exists, no further action is required.")
#       } else {
#
#         #Read the data file into a variable
#         variableName <- paste("FP_", i, sep="")
#         assign(variableName, readRDS(file = RDSFile))
#
#         if (i == 2016) {
```

```
#           selectedDataSet <- get(variableName)[Month < 5,]
#         } else {
#           selectedDataSet <- get(variableName)
#         }
#
#         #OriginState selection
#         selectedDataSet <- selectedDataSet[OriginState %in% getStates(),]
#
#         #DestState selection
#         selectedDataSet <- selectedDataSet[DestState %in% getStates(),]
#
#         #DistanceGroup resetting
#      selectedDataSet$DistanceGroup <- as.factor(selectedDataSet$DistanceGroup)
#
#         #Resetting the factors of the data table
#         selectedDataSet[] <-
#           lapply(selectedDataSet,
#                  function(x) if(is.factor(x)) factor(x) else x)
#
#         saveRDS(selectedDataSet, file = RDSFileSelected)
#
#         #Free up the memory
#         rm(list = variableName)
#         rm(variableName)
#         gc()
#
#       } #end of "if (file.exists(RDSFileSelected) == TRUE)"
#
#     } #end of "if (file.exists(RDSFile) != TRUE)"
#
#   } #end of "for (i in startYear:endYear)"
#
# }
# #'
# #' \code{CleanupWildLifeStrikeDataSet} cleans up the data
# #' set quality isses based on the data quality report
# #' findings and creates secondary exploration report for
# #' comparism purposes
# #'
# #' @param createPNG boolean
# #' Flag to decide to create the PNG images or not
# #'
# #' @examples
# #' CleanupWildLifeStrikeDataSet()
# #'
# CleanupWildLifeStrikeDataSet <- function(createPNG) {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   dataSummary <- data.table(
#     dataYear = character(),
```

```
#     numberOfRecords = integer(),
#     factorOPID = integer(),
#     factorATYPE = integer(),
#     factorAC_CLASS = integer(),
#     factorAC_MASS = integer(),
#     factorTYPE_ENG = integer(),
#     factorTIME_OF_DAY = integer(),
#     factorAIRPORT_ID = integer(),
#     factorSTATE = integer(),
#     factorPHASE_OF_FLT = integer(),
#     factorSKY = integer(),
#     factorPRECIP = integer(),
#     factorWARNED = integer()
#   )
#
#   dataSummaryAirport <- data.table(
#     airport = character()
#   )
#
#   for (i in startYear:endYear){
#     RDSFileName <- paste(i,
#                          "_Animal_Strikes_03_Sel.rds",
#                          sep = "")
#
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#
#     RDSFileNameCleaned <- paste(i,
#                                 "_Animal_Strikes_04_Cle.rds",
#                                 sep = "")
#
#     RDSFileCleaned <- paste(dataDir,
#                             "/",
#                             RDSFileNameCleaned,
#                             sep = "")
#
#     if (file.exists(RDSFile) != TRUE){
#       message(RDSFileName,
#               "is not available, ",
#               "please re-run the preparation scripts!")
#     } else {
#
#       if (file.exists(RDSFileCleaned) == TRUE){
#         message(RDSFileNameCleaned,
#                 " exists, no further action is required.")
#       } else {
#
#         #Read the data file into a variable
#         variableName <- paste("AS_", i, sep="")
#         assign(variableName, readRDS(file = RDSFile))
#
#         cleanedDataSet <- get(variableName)
```

```
#
#         #Convert the factor characters to uppercase
#         cleanedDataSet[] <-
#           lapply(cleanedDataSet,
#                  function(x) if(is.factor(x))
#                    as.factor(toupper(as.character(x))) else x)
#
#         #Change values to plural
#         cleanedDataSet[SKY == ”SOME CLOUD”, SKY:= ”SOME CLOUDS”]
#         cleanedDataSet[SKY == ”NO CLOUD”, SKY:= ”NO CLOUDS”]
#
#         #populate empty factors to none for selected columns
#         cleanedDataSet[TIME_OF_DAY == ””, TIME_OF_DAY:= ”NONE”]
#         cleanedDataSet[PHASE_OF_FLT == ””, PHASE_OF_FLT:= ”NONE”]
#         cleanedDataSet[SKY == ””, SKY:= ”NONE”]
#         cleanedDataSet[PRECIP == ””, PRECIP:= ”NONE”]
#         cleanedDataSet[WARNED == ””, WARNED:= ”NONE”]
#
#         #change engine type
#         cleanedDataSet[TYPE_ENG == ”A/C”, TYPE_ENG:= ”A”]
#         cleanedDataSet[TYPE_ENG == ”B/D”, TYPE_ENG:= ”B”]
#
#         #change the date to the incident day
#         cleanedDataSet[,INCIDENT_DATE := substr(INCIDENT_DATE, 9, 10)]
#
#         #Resetting the factors of the data table
#         cleanedDataSet[] <-
#           lapply(cleanedDataSet,
#                  function(x) if(is.factor(x)) factor(x) else x)
#
#         dataSummary <- rbindlist(
#           list(
#             dataSummary,
#             list(as.character(i),
#                  nrow(cleanedDataSet),
#                  length(levels(cleanedDataSet$OPID)),
#                  length(levels(cleanedDataSet$ATYPE)),
#                  length(levels(cleanedDataSet$AC_CLASS)),
#                  length(levels(cleanedDataSet$AC_MASS)),
#                  length(levels(cleanedDataSet$TYPE_ENG)),
#                  length(levels(cleanedDataSet$TIME_OF_DAY)),
#                  length(levels(cleanedDataSet$AIRPORT_ID)),
#                  length(levels(cleanedDataSet$STATE)),
#                  length(levels(cleanedDataSet$PHASE_OF_FLT)),
#                  length(levels(cleanedDataSet$SKY)),
#                  length(levels(cleanedDataSet$PRECIP)),
#                  length(levels(cleanedDataSet$WARNED))
#                  )
#             )
#         )
#
#         dataSummaryAirport <- rbindlist(
#           list(
#             dataSummaryAirport,
```

```
#             unique(cleanedDataSet[,c("AIRPORT_ID")],
#                    by = c("AIRPORT_ID"))
#           )
#         )
#
#
#         if (createPNG == TRUE) {
#
#           #Save the plots as PNG files
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = "AnimalStrike",
#                          DataField = "AC_CLASS",
#                          DataStage = "04_Cleaned",
#                          DataObject = cleanedDataSet)
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = "AnimalStrike",
#                          DataField = "AC_MASS",
#                          DataStage = "04_Cleaned",
#                          DataObject = cleanedDataSet)
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = "AnimalStrike",
#                          DataField = "TYPE_ENG",
#                          DataStage = "04_Cleaned",
#                          DataObject = cleanedDataSet)
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = "AnimalStrike",
#                          DataField = "TIME_OF_DAY",
#                          DataStage = "04_Cleaned",
#                          DataObject = cleanedDataSet)
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = "AnimalStrike",
#                          DataField = "PHASE_OF_FLT",
#                          DataStage = "04_Cleaned",
#                          DataObject = cleanedDataSet)
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = "AnimalStrike",
#                          DataField = "SKY",
#                          DataStage = "04_Cleaned",
#                          DataObject = cleanedDataSet)
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = "AnimalStrike",
#                          DataField = "PRECIP",
#                          DataStage = "04_Cleaned",
#                          DataObject = cleanedDataSet)
#         }
#
#         saveRDS(cleanedDataSet, file = RDSFileCleaned)
#
#         #Free up the memory
#         rm(list = variableName)
#         rm(variableName)
#         gc()
#
#       } #end of "if (file.exists(RDSFileCleaned) == TRUE)"
```

```
#
#     } #end of ”if (file.exists(RDSFile) != TRUE)”
#
#   } #end of ”for (i in startYear:endYear)”
#
#
#   RDSExpFileName <- ”03_CLEANED_Animal_Strikes.rds”
#
#   RDSExpFile <- paste(dataDir,
#                       ”/”,
#                       RDSExpFileName,
#                       sep = ””)
#
#   if (file.exists(RDSExpFile) != TRUE) {
#     saveRDS(dataSummary, file = RDSExpFile)
#   } else {
#     file.remove(RDSExpFile)
#     saveRDS(dataSummary, file = RDSExpFile)
#   }
#
#   RDSExpFileName <- ”03_CLEANED_Animal_Strikes_Airports.rds”
#
#   RDSExpFile <- paste(dataDir,
#                            ”/”,
#                       RDSExpFileName,
#                            sep = ””)
#
#   dataSummaryAirport <-
#     unique(dataSummaryAirport[,c(”airport”)],
#            by = c(”airport”))
#
#
#   if (file.exists(RDSExpFile) != TRUE) {
#     saveRDS(dataSummaryAirport,
#             file = RDSExpFile)
#   } else {
#     file.remove(RDSExpFile)
#     saveRDS(dataSummaryAirport,
#             file = RDSExpFile)
#   }
#
#
# }
# #'
# #' \code{CleanupOnTimeFlightPerformanceDataSet} cleans up
# #' the data set quality isses based on the data quality
# #' report findings and creates secondary exploration
# #' report for comparism purposes
# #'
# #' @param createPNG boolean
# #' Flag to decide to create the PNG images or not
# #'
# #' @examples
```

```
# #' CleanupOnTimeFlightPerformanceDataSet()
# #'
# CleanupOnTimeFlightPerformanceDataSet <- function(createPNG) {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   dataSummary <- data.table(
#     dataYear = character(),
#     numberOfRecords = integer(),
#     factorCarrier = integer(),
#     factorOrigin = integer(),
#     factorOriginState = integer(),
#     factorDest = integer(),
#     factorDestState = integer(),
#     factorDepTimeBlk = integer(),
#     factorDistanceGroup = integer()
#   )
#
#   dataSummaryOriginAirport <- data.table(
#     originAirport = character()
#   )
#
#   dataSummaryDestinationAirport <- data.table(
#     destinationAirport = character()
#   )
#
#   for (i in startYear:endYear){
#     RDSFileName <- paste(i,
#                          ”_On_Time_On_Time_Performance_03_Sel.rds”,
#                          sep = ””)
#
#     RDSFile <- paste(dataDir,
#                      ”/”,
#                      RDSFileName,
#                      sep = ””)
#
#     RDSFileNameCleaned <- paste(i,
#                                 ”_On_Time_On_Time_Performance_04_Cle.rds”,
#                                 sep = ””)
#
#     RDSFileCleaned <- paste(dataDir,
#                             ”/”,
#                             RDSFileNameCleaned,
#                             sep = ””)
#
#     if (file.exists(RDSFile) != TRUE){
#       message(RDSFileName,
#               ”is not available, ”,
#               ”please re-run the preparation scripts!”)
#     } else {
#
#       if (file.exists(RDSFileCleaned) == TRUE){
```

```
#         message(RDSFileNameCleaned,
#                 ” exists, no further action is required.”)
#       } else {
#
#         #Read the data file into a variable
#         variableName <- paste(”FP_”, i, sep=””)
#         assign(variableName, readRDS(file = RDSFile))
#
#
#         cleanedDataSet <- get(variableName)
#
#
#         #Resetting the factors of the data table
#         cleanedDataSet[] <-
#           lapply(cleanedDataSet,
#                  function(x) if(is.factor(x)) factor(x) else x)
#
#         dataSummary <- rbindlist(
#           list(
#             dataSummary,
#             list(as.character(i),
#                  nrow(cleanedDataSet),
#                  length(levels(cleanedDataSet$Carrier)),
#                  length(levels(cleanedDataSet$Origin)),
#                  length(levels(cleanedDataSet$OriginState)),
#                  length(levels(cleanedDataSet$Dest)),
#                  length(levels(cleanedDataSet$DestState)),
#                  length(levels(cleanedDataSet$DepTimeBlk)),
#                  length(levels(cleanedDataSet$DistanceGroup))
#                  )
#             )
#           )
#
#         dataSummaryOriginAirport <- rbindlist(
#           list(
#             dataSummaryOriginAirport,
#             unique(cleanedDataSet[,c(”Origin”)],
#                    by = c(”Origin”))
#           )
#         )
#
#         dataSummaryDestinationAirport <- rbindlist(
#           list(
#             dataSummaryDestinationAirport,
#             unique(cleanedDataSet[,c(”Dest”)],
#                    by = c(”Dest”))
#           )
#         )
#
#         if (createPNG == TRUE) {
#
#           #Save the plots as PNG files
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = ”FlightData”,
```

```
#                          DataField = ”Carrier”,
#                          DataStage = ”04_Cleaned”,
#                          DataObject = cleanedDataSet)
#
#           saveBarPlotPNG(DataYear = i,
#                          DataSet = ”FlightData”,
#                          DataField = ”DistanceGroup”,
#                          DataStage = ”04_Cleaned”,
#                          DataObject = cleanedDataSet)
#         }
#
#         saveRDS(cleanedDataSet, file = RDSFileCleaned)
#
#         #Free up the memory
#         rm(list = variableName)
#         rm(variableName)
#         gc()
#
#       } #end of ”if (file.exists(RDSFileCleaned) == TRUE)”
#
#     } #end of ”if (file.exists(RDSFile) != TRUE)”
#
#   } #end of ”for (i in startYear:endYear)”
#
#   RDSExpFileName <- ”03_CLEANED_Flight_Data.rds”
#
#   RDSExpFile <- paste(dataDir,
#                       ”/”,
#                       RDSExpFileName,
#                       sep = ””)
#
#   if (file.exists(RDSExpFile) != TRUE) {
#     saveRDS(dataSummary, file = RDSExpFile)
#   } else {
#     file.remove(RDSExpFile)
#     saveRDS(dataSummary, file = RDSExpFile)
#   }
#
#   RDSExpStateFileName <- ”03_CLEANED_Flight_Data_O_Airports.rds”
#
#   RDSExpStateFile <- paste(dataDir,
#                            ”/”,
#                            RDSExpStateFileName,
#                            sep = ””)
#
#   dataSummaryOriginAirport <-
#     unique(dataSummaryOriginAirport[,c(”originAirport”)],
#            by = c(”originAirport”))
#
#
#   if (file.exists(RDSExpStateFile) != TRUE) {
#     saveRDS(dataSummaryOriginAirport,
#             file = RDSExpStateFile)
#   } else {
```

```
#     file.remove(RDSExpStateFile)
#     saveRDS(dataSummaryOriginAirport,
#             file = RDSExpStateFile)
#   }
#
#
#   RDSExpStateFileName <- ”03_CLEANED_Flight_Data_D_Airports.rds”
#
#   RDSExpStateFile <- paste(dataDir,
#                            ”/”,
#                            RDSExpStateFileName,
#                            sep = ””)
#
#   dataSummaryDestinationAirport <-
#     unique(dataSummaryDestinationAirport[,c(”destinationAirport”)],
#            by = c(”destinationAirport”))
#
#   if (file.exists(RDSExpStateFile) != TRUE) {
#     saveRDS(dataSummaryDestinationAirport,
#             file = RDSExpStateFile)
#   } else {
#     file.remove(RDSExpStateFile)
#     saveRDS(dataSummaryDestinationAirport,
#             file = RDSExpStateFile)
#   }
#
# }
# #'
# #' \code{AirportDataSetDataPreparation} based on the configuration
# #' items checks if the airport data set file has been:
# #'  - downloaded
# #'  - saved as an R object
# #'
# #' @examples
# #' AirportDataSetDataPreparation()
# #'
# AirportDataSetDataPreparation <- function() {
#
#   dataDir <- getDataDir()
#   sourceFile <- paste(dataDir, ”NfdcFacilities.xls”, sep = ”/”)
#   destFile <- paste(dataDir, ”NfdcFacilities.csv”, sep = ”/”)
#
#   method=”auto”
#
#   #if the file exists then do not download again
#   if (file.exists(sourceFile) != TRUE) {
#     message(”Please download the aiport data set file.”)
#   } else
#   {
#     RDSFileName <- ”04_ORIG_Airport.rds”
#
#     RDSFile <- paste(dataDir,
#                      ”/”,
```

```
#                       RDSFileName,
#                       sep = "")
#
#     #copy and rename the file
#     file.copy(sourceFile, destFile, overwrite = TRUE)
#
#     DT <- data.table(read.delim(destFile))
#
#     saveRDS(DT, file = RDSFile)
#     message(RDSFileName," created.")
#
#   } #end of "if (file.exists(sourceFile) != TRUE)"
#
# }
# #'
# #' \code{DescribeAirportDataSet} re-creates the
# #' inputs based on the column selection of the data
# #' verification report
# #'
# #' @examples
# #' DescribeAirportDataSet()
# #'
# DescribeAirportDataSet <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   RDSFileName <- "04_ORIG_Airport.rds"
#
#   RDSFile <- paste(dataDir,
#                    "/",
#                    RDSFileName,
#                    sep = "")
#
#   RDSFileNameDescibed <- "05_DESC_Airport.rds"
#
#   RDSFileDescibed <- paste(dataDir,
#                            "/",
#                            RDSFileNameDescibed,
#                            sep = "")
#
#   if (file.exists(RDSFile) != TRUE){
#     message(RDSFileName,
#             "is not available, ",
#             "please re-run the preparation scripts!")
#   } else {
#
#     if (file.exists(RDSFileDescibed) == TRUE){
#       message(RDSFileNameDescibed,
#               " exists, no further action is required.")
#     } else {
#
```

```
#       #Read the data file into a variable
#       originalDataSet <- readRDS(file = RDSFile)
#
#       #set the required column names
#       ColumnNames <- c("Type",
#                        "LocationID",
#                        "Region",
#                        "State",
#                        "StateName",
#                        "City",
#                        "FacilityName",
#                        "ARPLatitude",
#                        "ARPLatitudeS",
#                        "ARPLongitude",
#                        "ARPLongitudeS",
#                        "ARPElevation",
#                        "LandAreaCoveredByAirport",
#                        "AirportStatusCode",
#                        "IcaoIdentifier"
#                        )
#
#       #Move reduces data into a new data set
#       describedDataSet <- originalDataSet[, ..ColumnNames]
#
#       saveRDS(describedDataSet, file = RDSFileDescibed)
#       message(RDSFileNameDescibed,
#               " created.")
#
#     } #end of "if (file.exists(RDSFileDescibed) == TRUE)"
#
#   } #end of "if (file.exists(RDSFile) != TRUE)"
#
# }
# #'
# #' \code{SelectAirportDataSet} executes the identified
# #' exclusions and inclusions in the data set validation report
# #'
# #' @examples
# #' SelectAirportDataSet()
# #'
# SelectAirportDataSet <- function() {
#
#   dataDir <- getDataDir()
#
#   RDSFileName <- "05_DESC_Airport.rds"
#
#   RDSFile <- paste(dataDir,
#                    "/",
#                    RDSFileName,
#                    sep = "")
#
#   RDSFileNameSelected <- "06_SEL_Airport.rds"
#
```

```
#   RDSFileSelected <- paste(dataDir,
#                            "/",
#                            RDSFileNameSelected,
#                            sep = "")
#
#   if (file.exists(RDSFile) != TRUE){
#     message(RDSFileName,
#             "is not available, ",
#             "please re-run the preparation scripts!")
#   } else {
#
#
#     if (file.exists(RDSFileSelected) == TRUE){
#       message(RDSFileNameSelected,
#               " exists, no further action is required.")
#     } else {
#
#       #Read the data file into a variable
#       originalDataSet <- readRDS(file = RDSFile)
#
#       #TYPE column selection
#       selectedDataSet <- originalDataSet[Type == "AIRPORT",]
#
#       #STATE selection
#       selectedDataSet <- selectedDataSet[State %in% getStates(),]
#
#       #Status selection
#       selectedDataSet <- selectedDataSet[AirportStatusCode == "O",]
#
#       #Resetting the factors of the data table
#       selectedDataSet[] <-
#         lapply(selectedDataSet,
#                function(x) if(is.factor(x)) factor(x) else x)
#
#       saveRDS(selectedDataSet, file = RDSFileSelected)
#       message(RDSFileNameSelected,
#               " created.")
#
#
#     } #end of "if (file.exists(RDSFileSelected) == TRUE)"
#
#   } #end of "if (file.exists(RDSFile) != TRUE)"
#
# }

# #'
# #' \code{CleanupAirportDataSet} cleans up the data
# #' set quality isses based on the data quality report
# #' findings
# #'
# #' @examples
# #' CleanupAirportDataSet()
# #'
# CleanupAirportDataSet <- function() {
```

```
#
#   dataDir <- getDataDir()
#
#   RDSFileName <- ”06_SEL_Airport.rds”
#
#   RDSFile <- paste(dataDir,
#                    ”/”,
#                    RDSFileName,
#                    sep = ””)
#
#   RDSFileNameCleaned <- ”07_CLE_Airport.rds”
#
#   RDSFileCleaned <- paste(dataDir,
#                           ”/”,
#                           RDSFileNameCleaned,
#                           sep = ””)
#
#   if (file.exists(RDSFile) != TRUE){
#     message(RDSFileName,
#             ”is not available, ”,
#             ”please re-run the preparation scripts!”)
#   } else {
#
#     if (file.exists(RDSFileCleaned) == TRUE){
#       message(RDSFileNameCleaned,
#               ” exists, no further action is required.”)
#     } else {
#
#       #Read the data file into a variable
#       cleanedDataSet <- readRDS(file = RDSFile)
#
#       #remove the ' character from the location id
#       cleanedDataSet$LocationID <- cleanedDataSet[,sub('.','',LocationID)]
#
#       #make the location id a factor
#       cleanedDataSet$LocationID <- as.factor(cleanedDataSet$LocationID)
#
#       #create the lat and long decimal values
#       cleanedDataSet <-
#         cleanedDataSet[,
#                        lat := convertToDMSNumber(as.character(ARPLatitude))]
#       cleanedDataSet <-
#         cleanedDataSet[,
#                      long := convertToDMSNumber(as.character(ARPLongitude))]
#
#       #Resetting the factors of the data table
#       cleanedDataSet[] <-
#         lapply(cleanedDataSet,
#                function(x) if(is.factor(x)) factor(x) else x)
#
#       saveRDS(cleanedDataSet, file = RDSFileCleaned)
#       message(RDSFileNameCleaned,
#               ” created.”)
#
```

```
#
#     } #end of "if (file.exists(RDSFileCleaned) == TRUE)"
#
#   } #end of "if (file.exists(RDSFile) != TRUE)"
#
# }

# #'
# #' \code{DeriveAirportAttributes} created airport specific attribures
# #' based on the flight performance data set and animal strike data set
# #'
# #' @examples
# #' DeriveAirportAttributes()
# #'
# DeriveAirportAttributes <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   RDSFileName <- "07_CLE_Airport.rds"
#
#   RDSFile <- paste(dataDir,
#                     "/",
#                   RDSFileName,
#                     sep = "")
#
#   airportsData <- readRDS(file = RDSFile)
#
#   for (i in startYear:endYear){
#
#     RDSFileNameFP <- paste(i,
#                         "_On_Time_On_Time_Performance_04_Cle.rds",
#                         sep = "")
#
#     RDSFileFP <- paste(dataDir,
#                     "/",
#                     RDSFileNameFP,
#                     sep = "")
#
#     RDSFileNameAS <- paste(i,
#                           "_Animal_Strikes_04_Cle.rds",
#                           sep = "")
#
#     RDSFileAS <- paste(dataDir,
#                       "/",
#                       RDSFileNameAS,
#                       sep = "")
#
#     if ((file.exists(RDSFileFP) != TRUE) ||
#         (file.exists(RDSFileAS) != TRUE)){
#       message(RDSFileNameFP,
#               " or ",
#               RDSFileNameAS,
```

```
#               " is not available, ",
#               "please re-run the preparation scripts!")
#     } else {
#
#       #Read the data file into a variable
#       variableName <- paste("FP_", i, sep="")
#       assign(variableName, readRDS(file = RDSFileFP))
#
#       dataSet <- get(variableName)
#
#       originData <- dataSet[,
#                             .(.N,
#                               max(Distance),
#                               min(Distance),
#                               sum(Distance)),
#                             by = c("Origin",
#                                    "Year")
#                             ]
#
#       names(originData) <- c("Airport",
#                              "Year",
#                              "OriginCount",
#                              "OriginMaxDistance",
#                              "OriginMinDistance",
#                              "OriginSumDistance")
#
#       setkey(originData, Airport, Year)
#
#       destData <- dataSet[,
#                           .(.N,
#                             max(Distance),
#                             min(Distance),
#                             sum(Distance)),
#                           by = c("Dest",
#                                  "Year")
#                           ]
#
#       names(destData) <- c("Airport",
#                            "Year",
#                            "DestinationCount",
#                            "DestinationMaxDistance",
#                            "DestinationMinDistance",
#                            "DestinationSumDistance")
#
#       setkey(destData, Airport, Year)
#
#       #Free up the memory
#       rm(list = variableName)
#       rm(variableName)
#       rm(dataSet)
#       gc()
#
#       #Read the data file into a variable
#       variableName <- paste("AS_", i, sep="")
```

```
#       assign(variableName, readRDS(file = RDSFileAS))
#
#       dataSet <- get(variableName)
#
#       strikeData <- dataSet[,
#                             .(.N),
#                             by = c("AIRPORT_ID",
#                                    "INCIDENT_YEAR")
#                             ]
#
#       names(strikeData) <- c("Airport",
#                              "Year",
#                              "StrikeNo")
#
#       strikeData$Airport <-
#         airportsData[IcaoIdentifier %in% strikeData$Airport,
#                      LocationID]
#
#       setkey(strikeData, Airport, Year)
#
#       #Free up the memory
#       rm(list = variableName)
#       rm(variableName)
#       rm(dataSet)
#       gc()
#
#
#       if (i == startYear){
#
#         originDataFull <- originData
#         destDataFull <- destData
#         strikeDataFull <- strikeData
#
#       } else {
#
#         originDataFull <- rbindlist(
#           list(
#             originDataFull,
#             originData),
#           fill = TRUE
#         )
#
#         destDataFull <- rbindlist(
#           list(
#             destDataFull,
#             destData),
#           fill = TRUE
#         )
#
#         strikeDataFull <- rbindlist(
#           list(
#             strikeDataFull,
#             strikeData),
#           fill = TRUE
```

```
#         )
#
#       }
#
#     } #end of ”if (file.exists(RDSFile) != TRUE)”
#
#   } #end of ”for (i in startYear:endYear)”
#
#   summedData <-
#     originDataFull[,.(
#       sum(OriginCount),
#       max(OriginMaxDistance),
#       min(OriginMinDistance),
#       sum(OriginSumDistance)/sum(OriginCount)
#     ),
#     by = ”Airport”]
#
#   names(summedData) <- c(”Airport”,
#                          ”OriginCount”,
#                          ”OriginMaxDistance”,
#                          ”OriginMinDistance”,
#                          ”OriginAvgDistance”)
#
#   #Resetting the factors of the data table
#   summedData[] <-
#     lapply(summedData,
#            function(x) if(is.factor(x)) factor(x) else x)
#
#   RDSFileName <- paste(”08_DER_Airport_Origin.rds”,sep = ””)
#
#   RDSFile <- paste(dataDir,”/”,RDSFileName,sep = ””)
#
#   if (file.exists(RDSFile) != TRUE) {
#     saveRDS(summedData, file = RDSFile)
#   } else {
#     file.remove(RDSFile)
#     saveRDS(summedData, file = RDSFile)
#   }
#
#
#   summedData <-
#     destDataFull[,.(
#       sum(DestinationCount),
#       max(DestinationMaxDistance),
#       min(DestinationMinDistance),
#       sum(DestinationSumDistance)/sum(DestinationCount)
#     ),
#     by = ”Airport”]
#
#   names(summedData) <- c(”Airport”,
#                        ”DestinationCount”,
#                        ”DestinationMaxDistance”,
#                        ”DestinationMinDistance”,
#                        ”DestinationAvgDistance”)
```

```
#
#   #Resetting the factors of the data table
#   summedData[] <-
#     lapply(summedData,
#            function(x) if(is.factor(x)) factor(x) else x)
#
#   RDSFileName <- paste("08_DER_Airport_Destination.rds",sep = "")
#
#   RDSFile <- paste(dataDir,"/",RDSFileName,sep = "")
#
#   if (file.exists(RDSFile) != TRUE) {
#     saveRDS(summedData, file = RDSFile)
#   } else {
#     file.remove(RDSFile)
#     saveRDS(summedData, file = RDSFile)
#   }
#
#
#   summedData <-
#     strikeDataFull[,.(
#       sum(StrikeNo)
#     ),
#     by = "Airport"]
#
#   names(summedData) <- c("Airport",
#                          "StrikeNo")
#
#   #Resetting the factors of the data table
#   summedData[] <-
#     lapply(summedData,
#            function(x) if(is.factor(x)) factor(x) else x)
#
#   RDSFileName <- paste("08_DER_Airport_Strike.rds",sep = "")
#
#   RDSFile <- paste(dataDir,"/",RDSFileName,sep = "")
#
#   if (file.exists(RDSFile) != TRUE) {
#     saveRDS(summedData, file = RDSFile)
#   } else {
#     file.remove(RDSFile)
#     saveRDS(summedData, file = RDSFile)
#   }
#
# }
# #'
# #' \code{IntegrateAttributesM1} integrates the attributes to
# #' be used by the first model
# #'
# #' @examples
# #' IntegrateAttributesM1()
# #'
# IntegrateAttributesM1 <- function() {
#
```

```
#   dataDir <- getDataDir()
#
#   RDSFileName <- ”07_CLE_Airport.rds”
#
#   RDSFile <- paste(dataDir,
#                    ”/”,
#                    RDSFileName,
#                    sep = ””)
#
#   RDSFileNameDest <- ”08_DER_Airport_Destination.rds”
#
#   RDSFileDest <- paste(dataDir,
#                    ”/”,
#                    RDSFileNameDest,
#                    sep = ””)
#
#   RDSFileNameOrig <- ”08_DER_Airport_Origin.rds”
#
#   RDSFileOrig <- paste(dataDir,
#                    ”/”,
#                    RDSFileNameOrig,
#                    sep = ””)
#
#   RDSFileNameStr <- ”08_DER_Airport_Strike.rds”
#
#   RDSFileStr <- paste(dataDir,
#                    ”/”,
#                    RDSFileNameStr,
#                    sep = ””)
#
#
#   if ((file.exists(RDSFile) != TRUE) ||
#       (file.exists(RDSFileDest) != TRUE) ||
#       (file.exists(RDSFileOrig) != TRUE) ||
#       (file.exists(RDSFileStr) != TRUE)
#       ){
#     message(RDSFileName,
#             ” or ”,
#             RDSFileNameDest,
#             ” or ”,
#             RDSFileNameOrig,
#             ” or ”,
#             RDSFileNameStr,
#             ” is not available, ”,
#             ”please re-run the preparation scripts!”)
#   } else {
#
#     airportData <- readRDS(file = RDSFile)
#     setkey(airportData, LocationID)
#     airportDestination <- readRDS(file = RDSFileDest)
#     setkey(airportDestination, Airport)
#     airportOrigin <- readRDS(file = RDSFileOrig)
#     setkey(airportOrigin, Airport)
#     airportStrike <- readRDS(file = RDSFileStr)
```

```
#     setkey(airportStrike, Airport)
#
#     mergedAirportFlightData <- merge(airportOrigin,
#                                      airportDestination,
#                                      all = FALSE)
#
#     #Resetting the NA values to zero
#     mergedAirportFlightData[is.na(mergedAirportFlightData)] <- 0
#
#     airportDatawFD <-
#       airportData[LocationID %in% mergedAirportFlightData$Airport]
#
#     #enrich data with the flight data
#     airportDatawFD <- merge(airportDatawFD,
#                             mergedAirportFlightData,
#                             by.x = ”LocationID”,
#                             by.y = ”Airport”,
#                             all.x = TRUE)
#
#     #enrich the data, with the strike data
#     airportDatawST <- merge(airportDatawFD,
#                             airportStrike,
#                             by.x = ”LocationID”,
#                             by.y = ”Airport”,
#                             all.x = TRUE)
#
#     airportDatawST[is.na(airportDatawST)] <- 0
#
#     #Resetting the factors of the data table
#     airportDatawST[] <-
#       lapply(airportDatawST,
#              function(x) if(is.factor(x)) factor(x) else x)
#
#
#     airportDataForModel01 <-
#       airportDatawST[,c(”LocationID”,
#                         ”Region”,
#                         ”State”,
#                         ”City”,
#                         ”FacilityName”,
#                         ”ARPElevation”,
#                         ”LandAreaCoveredByAirport”,
#                         ”OriginCount”,
#                         ”OriginMaxDistance”,
#                         ”OriginMinDistance”,
#                         ”OriginAvgDistance”,
#                         ”DestinationCount”,
#                         ”DestinationMaxDistance”,
#                         ”DestinationMinDistance”,
#                         ”DestinationAvgDistance”,
#                         ”StrikeNo”
#                         )
#                     ]
#
```

```
#     #Resetting the factors of the data table
#     airportDataForModel01[] <-
#       lapply(airportDataForModel01,
#              function(x) if(is.factor(x)) factor(x) else x)
#
#
#     RDSFileNameModel01 <- ”09_Model_01_Data.rds”
#
#     RDSFileModel01 <- paste(dataDir,
#                             ”/”,
#                             RDSFileNameModel01,
#                             sep = ””)
#
#     if (file.exists(RDSFileModel01) != TRUE) {
#       saveRDS(airportDataForModel01,
#               file = RDSFileModel01)
#     } else {
#       file.remove(RDSFileModel01)
#       saveRDS(airportDataForModel01,
#               file = RDSFileModel01)
#     }
#
#     saveMapPNG(airportDatawST$State, airportDatawST)
#
#     currentWorkingDir <- getwd()
#     setwd(getDocInputDir())
#
#     base_map <- map_data(”state”)
#
#     plotData <-
#       airportDatawST[!State %in% c(”AK”,
#                                    ”HI”),
#                      sum(StrikeNo),
#                      by = ”StateName”]
#     plotData$StateName <- tolower(plotData$StateName)
#     names(plotData) <- c(”region”,
#                          ”NumberOfStrikes”)
#
#     plotDataMapUSA <-
#       merge(statesDT,
#             plotData,
#             by=”region”,
#             all = TRUE)
#
#     plotDataMapUSA <-
#       plotDataMapUSA[plotDataMapUSA$region!=”district of columbia”,]
#
#     targetFileName <- ”USA_Airports.png”
#
#     ggplot() +
#       geom_polygon(data=plotDataMapUSA,
#                    aes(x=long,
#                        y=lat,
#                        group = group,
```

```
#                         color = "white",
#                         fill=plotDataMapUSA$NumberOfStrikes),
#                     colour="white") +
#        scale_fill_continuous(low = "#CBE5FF",
#                              high = "#00264C",
#                              guide="colorbar") +
#        theme_bw() +
#        labs(fill = "Number of Strikes",
#             x="",
#             y="") +
#        scale_y_continuous(breaks=c()) +
#        scale_x_continuous(breaks=c()) +
#        theme(panel.border =  element_blank())
#
#      ggsave(
#        targetFileName,
#        units = "in", #units are in pixels
#       width = 10, #width of the plot in in (should be the same as the height)
#       height = 7, #height of the plot in in (should be the same as the width)
#        dpi = 72 #nominal resolution in ppi (pixels per inch)
#      )
#
#      setwd(currentWorkingDir)
#
#    } #end of "if (file.exists(RDSFile) != TRUE)"
# }
#
#
#
# #'
# #' \code{IntegrateAttributesM2} integrates the attributes to
# #' be used by the second model
# #'
# #' @examples
# #' IntegrateAttributesM2()
# #'
# IntegrateAttributesM2 <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#
#   RDSFileName <- "07_CLE_Airport.rds"
#   RDSFile <- paste(dataDir,
#                    "/",
#                    RDSFileName,
#                    sep = "")
#
#   dataSummary <- data.table(
#     numberOfFlightRecords = integer(),
#     numberOfStrikesLinked = integer()
#   )
#
```

```
#
#   if (file.exists(RDSFile) != TRUE){
#     message(RDSFileName,
#             " is not available,",
#             " please re-run the preparation scripts!")
#   } else {
#     airportData <- readRDS(file = RDSFile)
#   }
#
#   for (i in startYear:endYear){
#     RDSFlightFileName <- paste(i,
#                                "_On_Time_On_Time_Performance_04_Cle.rds",
#                                sep = "")
#
#     RDSFlightFile <- paste(dataDir,
#                            "/",
#                            RDSFlightFileName,
#                            sep = "")
#
#     RDSStrikeFileName <- paste(i,
#                                "_Animal_Strikes_04_Cle.rds",
#                                sep = "")
#
#     RDSStrikeFile <- paste(dataDir,
#                            "/",
#                            RDSStrikeFileName,
#                            sep = "")
#
#     if ((file.exists(RDSFlightFile) != TRUE) ||
#         (file.exists(RDSStrikeFile) != TRUE)){
#       message(RDSFlightFileName,
#               " or ",
#               RDSStrikeFileName,
#               " is not available,",
#               " please re-run the preparation scripts!")
#     } else {
#
#       #Flight data
#       variableName <- paste("FP_", i, sep="")
#       assign(variableName, readRDS(file = RDSFlightFile))
#       flightDataSet <- get(variableName)
#       rm(list = variableName)
#       rm(variableName)
#
#       #handling carrier changes in the data
#       flightDataSet[,UniqueCarrier := substr(UniqueCarrier, 1, 2)]
#
#       #adding the name of the carrier to the data
#       flightDataSet[,CarrierName := getAirlineName(UniqueCarrier),
#                     by = UniqueCarrier]
#
#       #set the airline list
#       airlineList <- unique(flightDataSet$CarrierName)
#
```

```
#       #set the merge key
#       flightDataSet[, mergeKeyD := paste(FlightDate,
#                                          Dest,
#                                          DestState,
#                                          CarrierName,
#                                          FlightNum,
#                                          sep = "_")]
#
#       #set the merge key
#       flightDataSet[, mergeKeyO := paste(FlightDate,
#                                           Origin,
#                                           OriginState,
#                                           CarrierName,
#                                           FlightNum,
#                                           sep = "_")]
#       #reset the factors
#       flightDataSet[] <-
#         lapply(flightDataSet,
#                function(x) if(is.factor(x)) factor(x) else x)
#
#       #Strike data
#       variableName <- paste("AS_", i, sep="")
#       assign(variableName, readRDS(file = RDSStrikeFile))
#       strikeDataSet <- get(variableName)
#       rm(list = variableName)
#       rm(variableName)
#       gc()
#
#       #date setting
#     strikeDataSet[,FlightDate := as.Date(format(as.Date(paste(INCIDENT_YEAR,
#                                                           INCIDENT_MONTH,
#                                                            INCIDENT_DATE,
#                                                              sep = "-")),
#                                                 format = "%Y-%m-%d"))]
#       #set the required column names
#       ColumnNames <- c("OPERATOR",
#                        "FLT",
#                        "TIME",
#                        "AIRPORT_ID",
#                        "STATE",
#                        "PHASE_OF_FLT",
#                        "FlightDate")
#
#       #Move reduces data into a new data set
#       strikeDataSet <- strikeDataSet[, ..ColumnNames]
#
#       #remove those records where the flight number is empty
#       strikeDataSet <- strikeDataSet[!FLT=="",]
#
#       #typo handling
#       strikeDataSet[OPERATOR == "1US AIRWAYS",
#                     OPERATOR := "US AIRWAYS"]
#
#     #select only those strike reports which have the airline from the flight data
```

```
#       strikeDataSet <- strikeDataSet[OPERATOR %in% airlineList,]
#
#       #remove those records where the flight phase does not indicate if
#       #the strike has been on the origin or destination airport
#       strikeDataSet <- strikeDataSet[!PHASE_OF_FLT %in% c("TAXI",
#                                                            "NONE",
#                                                            "LOCAL",
#                                                            "PARKED"),]
#
#       #set airports
#       #Origin
#       strikeDataSet[PHASE_OF_FLT %in% c("CLIMB",
#                                          "TAKE-OFF RUN",
#                                          "DEPARTURE"),
#                     Origin := airportData[IcaoIdentifier == AIRPORT_ID,
#                                            LocationID],
#                     by = AIRPORT_ID]
#       #Destination
#       strikeDataSet[PHASE_OF_FLT %in% c("APPROACH",
#                                          "DESCENT",
#                                          "LANDING ROLL",
#                                          "ARRIVAL"),
#                     Dest := airportData[IcaoIdentifier == AIRPORT_ID,
#                                          LocationID],
#                     by = AIRPORT_ID]
#       #setting factors
#       strikeDataSet$Origin <- as.factor(strikeDataSet$Origin)
#       strikeDataSet$Dest <- as.factor(strikeDataSet$Dest)
#
#       #set the merge key
#       strikeDataSet[, mergeKeyD := paste(FlightDate,
#                                           Dest,
#                                           STATE,
#                                           OPERATOR,
#                                           FLT,
#                                           sep = "_")]
#
#       #set the merge key
#       strikeDataSet[, mergeKeyO := paste(FlightDate,
#                                           Origin,
#                                           STATE,
#                                           OPERATOR,
#                                           FLT,
#                                           sep = "_")]
#
#
#       #Setting the flag for the modelling
#       flightDataSet[,strikeFlag := 0]
#       flightDataSet[mergeKeyD %in% strikeDataSet$mergeKeyD,
#                     strikeFlag := 1]
#       flightDataSet[mergeKeyO %in% strikeDataSet$mergeKeyO,
#                     strikeFlag := 1]
#       flightDataSet$strikeFlag <- as.factor(flightDataSet$strikeFlag)
#
```

```
#       #Resetting the factors of the data tables
#       strikeDataSet[] <-
#         lapply(strikeDataSet,
#                function(x) if(is.factor(x)) factor(x) else x)
#
#
#       #set the required column names
#       ColumnNames <- c("Year",
#                        "Quarter",
#                        "Month",
#                        "DayofMonth",
#                        "DayOfWeek",
#                        "FlightDate",
#                        "UniqueCarrier",
#                        "FlightNum",
#                        "Origin",
#                        "Dest",
#                        "DepTimeBlk",
#                        "ArrTimeBlk",
#                        "CRSElapsedTime",
#                        "Distance",
#                        "DistanceGroup",
#                        "strikeFlag")
#
#       #Move reduces data into a new data set
#       flightDataSet <- flightDataSet[, ..ColumnNames]
#
#       dataSummary <- rbindlist(
#         list(
#           dataSummary,
#           data.table(
#             flightDataSet[,.N],
#             flightDataSet[strikeFlag == 1, .N]
#           )
#         )
#       )
#
#       #save the data for modeling
#       RDSModel02FileName <- paste(i,
#                             "_On_Time_On_Time_Performance_05_Mod.rds",
#                             sep = "")
#       RDSModel02File <- paste(dataDir,
#                         "/",
#                         RDSModel02FileName,
#                         sep = "")
#
#       saveRDS(flightDataSet, file = RDSModel02File)
#
#       #memory cleanup
#       rm(strikeDataSet)
#       rm(flightDataSet)
#       gc()
#
#
```

```
#     } # end of ”if ((file.exists(RDSFlightFile) != TRUE)
#       #          || (file.exists(RDSStrikeFile) != TRUE))”
#
#
#   } #end of ”for (i in startYear:endYear)”
#
#  dataSummaryFinal <- dataSummary[, .(sum(numberOfFlightRecords), sum(numberOfStrikesLink
#  names(dataSummaryFinal) <- c(”numberOfFlightRecords”,”numberOfStrikesLinked”)
#
#   RDSFileName <- ”11_Model_02_SummaryData.rds”
#   RDSFile <- paste(dataDir,
#                    ”/”,
#                    RDSFileName,
#                    sep = ””)
#
#   if (file.exists(RDSFile) != TRUE) {
#     saveRDS(dataSummaryFinal, file = RDSFile)
#   } else {
#     file.remove(RDSFile)
#     saveRDS(dataSummaryFinal, file = RDSFile)
#   }
#
# }
# #'
# #' \code{BuildModel01} builds the first model
# #'
# #' @examples
# #' BuildModel01()
# #'
# BuildModel01 <- function() {
#
#   dataDir <- getDataDir()
#
#   RDSFileName <- ”09_Model_01_Data.rds”
#
#   RDSFile <- paste(dataDir,
#                    ”/”,
#                    RDSFileName,
#                    sep = ””)
#
#   if (file.exists(RDSFile) != TRUE) {
#     message(RDSFileName,
#             ” is not available, ”,
#             ”please re-run the preparation scripts!”)
#   } else {
#
#     modelData <- readRDS(file = RDSFile)
#
#     modelData <- modelData[StrikeNo > 0]
#     modelData <- modelData[LandAreaCoveredByAirport > 0]
#     modelData <- modelData[ARPElevation > 0]
#
#     #Create the histograms
```

```
#     binSize <- (max(modelData$StrikeNo) -
#                 min(modelData$StrikeNo))/10
#     saveModelingHistogramPNG(FieldName =  ”StrikeNo”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$OriginCount) -
#                 min(modelData$OriginCount))/10
#     saveModelingHistogramPNG(FieldName =  ”OriginCount”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$OriginMaxDistance) -
#                 min(modelData$OriginMaxDistance))/10
#     saveModelingHistogramPNG(FieldName =  ”OriginMaxDistance”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$OriginMinDistance) -
#                 min(modelData$OriginMinDistance))/10
#     saveModelingHistogramPNG(FieldName =  ”OriginMinDistance”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$OriginAvgDistance) -
#                 min(modelData$OriginAvgDistance))/10
#     saveModelingHistogramPNG(FieldName =  ”OriginAvgDistance”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$DestinationCount) -
#                 min(modelData$DestinationCount))/10
#     saveModelingHistogramPNG(FieldName =  ”DestinationCount”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$DestinationMaxDistance) -
#                 min(modelData$DestinationMaxDistance))/10
#     saveModelingHistogramPNG(FieldName =  ”DestinationMaxDistance”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$DestinationMinDistance) -
#                 min(modelData$DestinationMinDistance))/10
#     saveModelingHistogramPNG(FieldName =  ”DestinationMinDistance”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$DestinationAvgDistance) -
#                 min(modelData$DestinationAvgDistance))/10
#     saveModelingHistogramPNG(FieldName =  ”DestinationAvgDistance”,
#                              DataObject = modelData,
#                              BinSize = binSize)
#
```

```
#     binSize <- (max(modelData$ARPElevation) -
#                 min(modelData$ARPElevation))/10
#     saveModelingHistogramPNG(FieldName =  "ARPElevation",
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     binSize <- (max(modelData$LandAreaCoveredByAirport) -
#                 min(modelData$LandAreaCoveredByAirport))/10
#     saveModelingHistogramPNG(FieldName =  "LandAreaCoveredByAirport",
#                              DataObject = modelData,
#                              BinSize = binSize)
#
#     modelDataLog <- modelData[]
#
#     modelDataLog[] <-
#       lapply(modelDataLog,
#              function(x) if(is.numeric(x)) log(x) else x)
#
#     modelDataLog[] <-
#       lapply(modelDataLog,
#              function(x) if(is.integer(x)) log(x) else x)
#
#     #Create the histograms of the log values
#     binSize <- (max(modelDataLog$StrikeNo) -
#                 min(modelDataLog$StrikeNo))/10
#     saveModelingHistogramLogPNG(FieldName =  "StrikeNo",
#                                 DataObject = modelDataLog,
#                                 BinSize = binSize)
#
#     binSize <- (max(modelDataLog$OriginCount) -
#                 min(modelDataLog$OriginCount))/10
#     saveModelingHistogramLogPNG(FieldName =  "OriginCount",
#                                 DataObject = modelDataLog,
#                                 BinSize = binSize)
#
#     binSize <- (max(modelDataLog$OriginMaxDistance) -
#                 min(modelDataLog$OriginMaxDistance))/10
#     saveModelingHistogramLogPNG(FieldName =  "OriginMaxDistance",
#                                 DataObject = modelDataLog,
#                                 BinSize = binSize)
#
#     binSize <- (max(modelDataLog$OriginMinDistance) -
#                 min(modelDataLog$OriginMinDistance))/10
#     saveModelingHistogramLogPNG(FieldName =  "OriginMinDistance",
#                                 DataObject = modelDataLog,
#                                 BinSize = binSize)
#
#     binSize <- (max(modelDataLog$OriginAvgDistance) -
#                 min(modelDataLog$OriginAvgDistance))/10
#     saveModelingHistogramLogPNG(FieldName =  "OriginAvgDistance",
#                                 DataObject = modelDataLog,
#                                 BinSize = binSize)
#
#     binSize <- (max(modelDataLog$DestinationCount) -
```

```
#                  min(modelDataLog$DestinationCount))/10
#      saveModelingHistogramLogPNG(FieldName =  "DestinationCount",
#                                  DataObject = modelDataLog,
#                                  BinSize = binSize)
#
#      binSize <- (max(modelDataLog$DestinationMaxDistance) -
#                  min(modelDataLog$DestinationMaxDistance))/10
#      saveModelingHistogramLogPNG(FieldName =  "DestinationMaxDistance",
#                                  DataObject = modelDataLog,
#                                  BinSize = binSize)
#
#      binSize <- (max(modelDataLog$DestinationMinDistance) -
#                  min(modelDataLog$DestinationMinDistance))/10
#      saveModelingHistogramLogPNG(FieldName =  "DestinationMinDistance",
#                                  DataObject = modelDataLog,
#                                  BinSize = binSize)
#
#      binSize <- (max(modelDataLog$DestinationAvgDistance) -
#                  min(modelDataLog$DestinationAvgDistance))/10
#      saveModelingHistogramLogPNG(FieldName =  "DestinationAvgDistance",
#                                  DataObject = modelDataLog,
#                                  BinSize = binSize)
#
#      binSize <- (max(modelDataLog$ARPElevation) -
#                  min(modelDataLog$ARPElevation))/10
#      saveModelingHistogramLogPNG(FieldName =  "ARPElevation",
#                                  DataObject = modelDataLog,
#                                  BinSize = binSize)
#
#      binSize <- (max(modelDataLog$LandAreaCoveredByAirport) -
#                  min(modelDataLog$LandAreaCoveredByAirport))/10
#      saveModelingHistogramLogPNG(FieldName =  "LandAreaCoveredByAirport",
#                                  DataObject = modelDataLog,
#                                  BinSize = binSize)
#
#      #Creating the models
#
#      RDSFileNameM1 <- "10_Model_01_01.rds"
#      RDSFileM1 <- paste(dataDir,
#                     "/",
#                     RDSFileNameM1,
#                     sep = "")
#
#      model01 <- lm(StrikeNo ~
#                    OriginCount +
#                    DestinationCount
#                  , data = modelDataLog)
#
#      saveRDS(model01, file = RDSFileM1)
#
#
#      RDSFileNameM2 <- "10_Model_01_02.rds"
#      RDSFileM2 <- paste(dataDir,
#                       "/",
```

```
#                         RDSFileNameM2,
#                         sep = "")
#
#      model02 <- lm(StrikeNo ~
#                      OriginCount +
#                      OriginMaxDistance +
#                      OriginMinDistance +
#                      OriginAvgDistance +
#                      DestinationCount +
#                      DestinationMaxDistance +
#                      DestinationMinDistance +
#                      DestinationAvgDistance
#                    , data = modelData)
#
#      saveRDS(model02, file = RDSFileM2)
#
#
#      RDSFileNameM3 <- "10_Model_01_03.rds"
#      RDSFileM3 <- paste(dataDir,
#                         "/",
#                         RDSFileNameM3,
#                         sep = "")
#
#      model03 <- lm(StrikeNo ~
#                      ARPElevation +
#                      LandAreaCoveredByAirport
#                    , data = modelData)
#
#      saveRDS(model03, file = RDSFileM3)
#
#
#      RDSFileNameM4 <- "10_Model_01_04.rds"
#      RDSFileM4 <- paste(dataDir,
#                         "/",
#                         RDSFileNameM4,
#                         sep = "")
#
#      model04 <- lm(StrikeNo ~
#                      Region
#                    , data = modelData)
#
#      saveRDS(model04, file = RDSFileM4)
#
#      RDSFileNameM5 <- "10_Model_01_05.rds"
#      RDSFileM5 <- paste(dataDir,
#                         "/",
#                         RDSFileNameM5,
#                         sep = "")
#
#      model05 <- lm(StrikeNo ~
#                      Region +
#                      State
#                    , data = modelData)
#
```

```
#     saveRDS(model05, file = RDSFileM5)
#
#
#     RDSFileNameM6 <- "10_Model_01_06.rds"
#     RDSFileM6 <- paste(dataDir,
#                        "/",
#                        RDSFileNameM6,
#                        sep = "")
#
#     model06 <- lm(StrikeNo ~
#                     State +
#                     OriginCount +
#                     DestinationCount +
#                     ARPElevation
#                   , data = modelData)
#
#     saveRDS(model06, file = RDSFileM6)
#
#   } #end of "if (file.exists(RDSFile) != TRUE)"
#
# }
```

```
# #'
# #' \code{BuildModel02Data} builds data the second model
# #'
# #' @examples
# #' BuildModel02Data()
# #'
# BuildModel02Data <- function() {
#
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#   set.seed(42)
#
#   trainDataFull <- data.table(
#     Year = integer(),
#     Quarter = integer(),
#     Month = integer(),
#     DayofMonth = integer(),
#     DayOfWeek = integer(),
#     FlightDate = factor(),
#     UniqueCarrier = factor(),
#     FlightNum = integer(),
#     Origin = factor(),
#     Dest = factor(),
#     DepTimeBlk = factor(),
#     ArrTimeBlk = factor(),
#     CRSElapsedTime = numeric(),
#     Distance = numeric(),
#     DistanceGroup = factor(),
#     strikeFlag = factor()
#   )
#
```

```
#    for (i in startYear:endYear){
#      RDSModelFileName <- paste(i,
#                                "_On_Time_On_Time_Performance_05_Mod.rds",
#                                sep = "")
#
#      RDSModelFile <- paste(dataDir,
#                            "/",
#                            RDSModelFileName,
#                            sep = "")
#
#      if (file.exists(RDSModelFile) != TRUE) {
#        message(RDSModelFileName,
#                " is not available, ",
#                "please re-run the preparation scripts!")
#      } else {
#
#        trainData <- readRDS(file = RDSModelFile)
#
#        #take only the striked data
#        trainDataStriked <-
#          trainData[strikeFlag == "1",]
#        #boost the number of striked data
#        trainDataStriked <-
#          trainDataStriked[rep(1:.N, each = 3)]
#
#        #take only the unstriked data
#        trainDataNonStriked <-
#          trainData[strikeFlag == "0",]
#        #take only 1% of the original data amount
#        trainDataNonStriked <-
#          trainDataNonStriked[sample(.N,round(.N*0.1))]
#
#        #merge the reduced and boosted data tables
#        trainData <- rbindlist(
#          list(
#            trainDataStriked,
#            trainDataNonStriked
#          )
#        )
#
#        #set the required column names
#        ColumnNames <- c("Year",
#                         "Quarter",
#                         "Month",
#                         "DayofMonth",
#                         "DayOfWeek",
#                         "FlightDate",
#                         "UniqueCarrier",
#                         "FlightNum",
#                         "Origin",
#                         "Dest",
#                         "DepTimeBlk",
#                         "ArrTimeBlk",
#                         "CRSElapsedTime",
```

```
#                        ”Distance”,
#                        ”DistanceGroup”,
#                        ”strikeFlag”)
#
#       #Move reduces data into a new data set
#       trainData <- trainData[, ..ColumnNames]
#
#       #merge striked and unstriked data
#       trainDataFull <- rbindlist(
#           list(
#             trainDataFull,
#             trainData
#           )
#         )
#
#
#       #free memory
#       rm(trainData)
#       rm(trainDataStriked)
#       rm(trainDataNonStriked)
#       gc()
#
#     } # end of ”if (file.exists(RDSModelFile) != TRUE)”
#   } #end of ”for (i in startYear:endYear)”
#
#   #Resetting the factors of the data tables
#   trainDataFull[] <-
#     lapply(trainDataFull,
#            function(x) if(is.factor(x)) factor(x) else x)
#
#
#   #save the data for modeling
#   RDSFileName <- ”12_Model_02_Data.rds”
#   RDSFile <- paste(dataDir,
#                           ”/”,
#                           RDSFileName,
#                           sep = ””)
#
#   saveRDS(trainDataFull, file = RDSFile)
#
# }

# #'
# #' \code{BuildModel02} builds the second model
# #'
# #' @examples
# #' BuildModel02()
# #'
# BuildModel02 <- function() {
#
#   dataDir <- getDataDir()
#   set.seed(42)
#
#   RDSFileName <- ”12_Model_02_Data.rds”
```

```
#   RDSFile <- paste(dataDir,
#                    "/",
#                    RDSFileName,
#                    sep = "")
#
#   if (file.exists(RDSFile) != TRUE) {
#     message(RDSFileName,
#             " is not available, ",
#             "please re-run the preparation scripts!")
#   } else {
#
#     #read the data for model building
#     trainData <- readRDS(file = RDSFile)
#
#     #start H2O
#     h2o.init(
#       nthreads=-1,
#       max_mem_size = "5G"
#     )
#
#     #clenaup the cluster
#     h2o.removeAll()
#
#     #upload the data to H2O
#     trainDataH2O <- as.h2o(trainData)
#
#     #remove the data object
#     rm(trainData)
#     gc()
#
#     #create test and train sets
#     splits <- h2o.splitFrame(
#       trainDataH2O,
#       c(0.6,0.2),
#       seed=42)
#
#     #data sets
#     trainH2O <- h2o.assign(splits[[1]], "train.hex")
#     validH2O <- h2o.assign(splits[[2]], "valid.hex")
#     testH2O <- h2o.assign(splits[[3]], "test.hex")
#
#     #train the initial model
#     trainedModelInit <- h2o.gbm(
#       training_frame = trainH2O,
#       validation_frame = validH2O,
#       x=1:15,
#       y=16,
#       seed = 42)
#
#     #summary(trainedModelInit)
#
#     ColumnNames <- c("Quarter",
#                      "Month",
#                      "DayofMonth",
```

```
#                      "DayOfWeek",
#                      "UniqueCarrier",
#                      "FlightNum",
#                      "Origin",
#                      "Dest",
#                      "DepTimeBlk",
#                      "ArrTimeBlk",
#                      "CRSElapsedTime",
#                      "Distance",
#                      "DistanceGroup")
#
#     #train the model
#     trainedModel <- h2o.gbm(
#       training_frame = trainH2O,
#       validation_frame = validH2O,
#       x = ColumnNames,
#       y = 16,
#       ntrees = 200,
#       learn_rate = 0.2,
#       max_depth = 10,
#       stopping_rounds = 2,
#       stopping_metric = "logloss",
#       stopping_tolerance = 0.01,
#       score_each_iteration = T,
#       seed = 42)
#
#     #summary(trainedModel)
#
#     #save the model
#     model_path_init <-
#       h2o.saveModel(object=trainedModelInit,
#                     path=getDataDir(),
#                     force=TRUE)
#     RDSFileName <- "13_Model_02_Path_Init.rds"
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#     saveRDS(model_path_init, file = RDSFile)
#
#
#     model_path <-
#       h2o.saveModel(object=trainedModel,
#                     path=getDataDir(),
#                     force=TRUE)
#     RDSFileName <- "14_Model_02_Path.rds"
#     RDSFile <- paste(dataDir,
#                      "/",
#                      RDSFileName,
#                      sep = "")
#     saveRDS(model_path, file = RDSFile)
#
#
#     #load the model
```

```
#     # #RDSFileName <- "13_Model_02_Path_Init.rds"
#     # #RDSFileName <- "14_Model_02_Path.rds"
#     # RDSFile <- paste(dataDir,
#     #                  "/",
#     #                  RDSFileName,
#     #                  sep = "")
#     # model_path <- readRDS(file = RDSFile)
#     # saved_model <- h2o.loadModel(model_path)
#
#
#     #final scoring
#     # perf <-
#     #   h2o.performance(model = saved_model,
#     #                   newdata = testH2O)
#
#     #create the confusion matrix for the scored data
#     # h2o.confusionMatrix(perf)
#
#     h2o.shutdown(prompt = FALSE)
#
#   } # end of "if (file.exists(RDSModelFile) != TRUE)"
#
# }
# #'
# #' \code{BuildPresentation} creates the final presentation
# #'
# #' @examples
# #' BuildPresentation()
# #'
# #BuildPresentation <- function() {
#
#   docDir <- getDocDir()
#   docInputDir <- getDocInputDir()
#
#
#   pptx.file <-
#     paste(docDir,
#           "FinalPresentation.pptx",
#           sep="/")
#
#   doc <-
#     pptx(title="Prediction of Animal Strike on US Commercial Flights")
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title Slide")
#   doc <-
#     addTitle(doc,
#              "Prediction of Animal Strike on US Commercial Flights")
#   doc <-
#     addSubtitle(doc,
#                 "GÃ¡bor HorvÃ¡th")
#
```

```
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
#     addTitle(doc,"Business objectives" )
#  text = c("Create a statistical analysis to identify those reasons (based on the data availa
#        "Create a prediction model, which can be used to predict the risk of an animal strike
#   doc <-
#     addParagraph(doc,
#                  value = text )
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
#     addTitle(doc,"Data sources" )
#   text = c("Federal Aviation Administration - Wildlife Strike Database",
#        "United States Department of Transportation - Bureau of Transportation Statistics -
#        "Federal Aviation Administration - Airport Data & Contact Information")
#   doc <-
#     addParagraph(doc,
#                  value = text )
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
#     addTitle(doc,"Tools used" )
#   text = c("R",
#            "R Studio",
#            "knitr, MiKTeX",
#            "ggplot2",
#            "H2O",
#            "CRIPS-DM process model",
#            "buckets, github")
#   doc <-
#     addParagraph(doc,
#                  value = text )
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
#     addTitle(doc,"Strike distribution" )
#   image <- paste(docInputDir,
#                  "USA_Airports.png",
#                  sep="/")
#   doc <- addImage(doc, image)
#
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
```

```
#     addTitle(doc,"Model 01 - details" )
#   text = c("log-log linear regression model",
#        "categorical and continous predictor variables from airport point of view",
#            "outcome variable: number of animal strike per airport",
#            "location and traffic of the airport influence the outcome")
#   doc <-
#     addParagraph(doc,
#                  value = text )
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
#     addTitle(doc,"Model 02 - details" )
#   text = c("H2O platform - Java",
#        "categorical and continous predictor variables from flight point of view",
#            "outcome variable: flight got animal strike or not",
#        "most influental predictors were the airports (origin, destination), the arrival an
#   doc <-
#     addParagraph(doc,
#                  value = text )
#
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
#     addTitle(doc,"Receiver Operating Characteristic (ROC) curve" )
#   image <- paste(docInputDir,
#                  "GBM_Model_ROC_valid.jpg",
#                  sep="/")
#   doc <- addImage(doc,
#                   image,
#                   width = 5,
#                   height = 5,
#                   par.properties = parProperties(align = "center"))
#
#
#   doc <-
#     addSlide(doc,
#              slide.layout="Title and Content")
#   doc <-
#     addTitle(doc,"Confusion matrix" )
#   image <- paste(docInputDir,
#                  "GBM_Model_scoring.jpg",
#                  sep="/")
#   doc <- addImage(doc,
#                   image,
#                   width = 7,
#                   height = 5,
#                   par.properties = parProperties(text.align = "center"))
#
#   doc <-
#     addSlide(doc,
```

```
#               slide.layout="Title Slide")
#   doc <-
#     addTitle(doc,
#              "Thank you")
#
#   writeDoc(doc,
#            file = pptx.file)
#
# #}
```

```
# Warning in readLines("../90-UserDefinedFunctions.R"): incomplete final line
# found on '../90-UserDefinedFunctions.R'
```

```
# #'
# #' \code{loadLibraries} checkes if the required libraries are
# #'  - installed and
# #'  - loaded
# #' if not, the it installs (if required) and loads them.
# #'
# #' @examples
# #' loadLibraries()
# #'
# loadLibraries <- function() {
#   if (!require(installr)) {install.packages("installr"); require(installr)}
#   if (!require(RODBC)) {install.packages("RODBC"); require(RODBC)}
#   if (!require(knitr)) {install.packages("knitr"); require(knitr)}
#  if (!require(data.table)) {install.packages("data.table"); require(data.table)}
#   if (!require(dplyr)) {install.packages("dplyr"); require(dplyr)}
#   if (!require(dtplyr)) {install.packages("dtplyr"); require(dtplyr)}
#   if (!require(ggplot2)) {install.packages("ggplot2"); require(ggplot2)}
#  if (!require(ReporteRs)) {install.packages("ReporteRs"); require(ReporteRs)}
#   if (!require(yaml)) {install.packages("yaml"); require(yaml)}
#   if (!require(png)) {install.packages("png"); require(png)}
#   if (!require(grid)) {install.packages("grid"); require(grid)}
#   if (!require(maps)) {install.packages("maps"); require(maps)}
#   if (!require(mapdata)) {install.packages("mapdata"); require(mapdata)}
#   if (!require(sp)) {install.packages("sp"); require(sp)}
#   if (!require(h2o)) {install.packages("h2o"); require(h2o)}
#
#   #update R
#   updateR(TRUE)
# }
#
#
# #'
# #' \code{versionDetails} provides details about the running environment
# #'
# #' @return text with the versions of R, RStudio, and used packages
# #'
# #' @examples
# #' versionDetails()
# #'
# versionDetails <- function() {
#
#   cat(paste(
```

```
#      ”R Studio version 1.0.143\n\n”,
#      version$version.string, ” ”, version$`svn rev`,”\n\n”,
#      ”Package versions:\n”,
#      ”- RODBC version ”, packageVersion(”RODBC”),”\n”,
#      ”- knitr version ”, packageVersion(”knitr”),”\n”,
#      ”- data.table version ”, packageVersion(”data.table”),”\n”,
#      ”- dplyr version ”, packageVersion(”dplyr”),”\n”,
#      ”- dtplyr version ”, packageVersion(”dtplyr”),”\n”,
#      ”- ReporteRs version ”, packageVersion(”ReporteRs”),”\n”,
#      ”- ReporteRsjars version ”, packageVersion(”ReporteRsjars”),”\n”,
#      ”- installr version ”, packageVersion(”installr”),”\n”,
#      ”- stringr version ”, packageVersion(”stringr”),”\n”,
#      ”- ggplot2 version ”, packageVersion(”ggplot2”),”\n”,
#      ”- yaml version ”, packageVersion(”yaml”),”\n”,
#      ”- png version ”, packageVersion(”png”),”\n”,
#      ”- grid version ”, packageVersion(”grid”),”\n”,
#      ”- maps version ”, packageVersion(”maps”),”\n”,
#      ”- mapdata version ”, packageVersion(”mapdata”),”\n”,
#      ”- sp version ”, packageVersion(”sp”),”\n”,
#      ”- h2o version ”, packageVersion(”h2o”),”\n\n”,
#      ”Base package versions:\n”,
#      ”- stats version ”, packageVersion(”stats”),”\n”,
#      ”- graphics version ”, packageVersion(”graphics”),”\n”,
#      ”- grDevices version ”, packageVersion(”grDevices”),”\n”,
#      ”- utils version ”, packageVersion(”utils”),”\n”,
#      ”- datasets version ”, packageVersion(”datasets”),”\n”,
#      ”- methods version ”, packageVersion(”methods”),”\n”,
#      ”- base version ”, packageVersion(”base”),sep=””))
# }
#
#
# #'
# #' \code{versionDetailsMiKTeX} provides details about the running
# #' environment
# #'
# #' @return text with the versions of MiKTeX
# #'
# #' @examples
# #' versionDetailsMiKTeX()
# #'
# versionDetailsMiKTeX <- function() {
#   cat(system(”mpm --version”, intern = TRUE), sep = '\n')
# }
#
#
# #'
# #' \code{versionDetailsMiKTeXPackages} provides details about the
# #' running environment
# #'
# #' @return text with the versions of the installed MiKTeX packages
# #'
# #' @examples
# #' versionDetailsMiKTeXPackages()
# #'
```

```
# versionDetailsMiKTeXPackages <- function() {
#   cat(system("mpm --list", intern = TRUE), sep = '\n')
# }
#
#
# #'
# #' \code{readConfigFile} reads the YAML config file into a global
# #' environment variable
# #'
# #' @param a boolean
# #' The YAML config file is in the working directory or in the parent
# #' directory (i.e. one directory above)
# #'
# #' @examples
# #' readConfigFile(TRUE)
# #'
# readConfigFile <- function(a) {
#   vName <- "config"
#   if (a == TRUE){
#     assign(vName, yaml.load_file("91-Config.yaml"), envir = .GlobalEnv)
#   }
#   else {
#     assign(vName, yaml.load_file("../91-Config.yaml"), envir = .GlobalEnv)
#   }
# }
#
#
# #'
# #' \code{getMainDir} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the maindir configuration item
# #'
# #' @examples
# #' getMainDir()
# #'
# getMainDir <- function() {
#   return(config$directories$maindir)
# }
#
#
# #'
# #' \code{getBackupDir} provides the value of the specific
# #' configuration item and creates the directory if it does
# #' not exist
# #'
# #' @return the value of the backupdir configuration item
# #'
# #' @examples
# #' getBackupDir()
# #'
# getBackupDir <- function() {
#   backupdir <- config$directories$backupdir
#   subdir <- Sys.Date()
```

```
#   returnvalue <- file.path(backupdir, subdir)
#
#   if (!file.exists(returnvalue)){
#     dir.create(returnvalue)
#     dir.create(file.path(returnvalue, "Documents"))
#   }
#   return(returnvalue)
# }
#
#
# #'
# #' \code{getDocDir} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the documents configuration item
# #'
# #' @examples
# #' getDocDir()
# #'
# getDocDir <- function() {
#   return(config$directories$documents)
# }
#
#
# #'
# #' \code{getDocInputDir} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the documentinput configuration item
# #'
# #' @examples
# #' getDocInputDir()
# #'
# getDocInputDir <- function() {
#   return(config$directories$documentinput)
# }
#
#
# #'
# #' \code{getDocOutputDir} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the documentoutput configuration item
# #'
# #' @examples
# #' getDocOutputDir()
# #'
# getDocOutputDir <- function() {
#   return(config$directories$documentoutput)
# }
#
#
# #'
# #' \code{getDataDir} provides the value of the specific
```

```
# #' configuration item
# #'
# #' @return the value of the datasets configuration item
# #'
# #' @examples
# #' getDataDir()
# #'
# getDataDir <- function() {
#   return(config$directories$datasets)
# }
#
#
# #'
# #' \code{getStartYear} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the startyear configuration item
# #'
# #' @examples
# #' getStartYear()
# #'
# getStartYear <- function() {
#   return(config$years$startyear)
# }
#
#
# #'
# #' \code{getEndYear} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the endyear configuration item
# #'
# #' @examples
# #' getEndYear()
# #'
# getEndYear <- function() {
#   return(config$years$endyear)
# }
#
#
# #'
# #' \code{getStartMonth} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the startmonth configuration item
# #'
# #' @examples
# #' getStartMonth()
# #'
# getStartMonth <- function() {
#   return(config$months$startmonth)
# }
#
#
```

```
# #'
# #' \code{getEndMonth} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the endmonth configuration item
# #'
# #' @examples
# #' getEndMonth()
# #'
# getEndMonth <- function() {
#   return(config$months$endmonth)
# }
#
#
# #'
# #' \code{backupFiles} makes a copy of the most important
# #' files to a safe location set by the YAML configuration
# #' file
# #'
# #' @examples
# #' backupFiles()
# #'
# backupFiles <- function() {
#   #Main directory files
#   filesMain <- list.files(getMainDir(), full.names = TRUE)
#   file.copy(filesMain, getBackupDir(), overwrite = TRUE)
#   #Documents folder
#   filesDocuments <- list.files(getDocDir(), full.names = TRUE)
#   file.copy(filesDocuments,
#             file.path(getBackupDir(),
#                       ”Documents”),
#             overwrite = TRUE)
# }
#
#
# #'
# #' \code{getWData} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the wildlife configuration item
# #'
# #' @examples
# #' getWData()
# #'
# getWData <- function() {
#   return(config$sources$wildlife)
# }
#
#
# #'
# #' \code{getFData} provides the value of the specific
# #' configuration item
# #'
# #' @return the value of the flightdata configuration item
```

```
# #'
# #' @examples
# #' getFData()
# #'
# getFData <- function() {
#   return(config$sources$flightdata)
# }
#
#
# #'
# #' \code{removeDataSetVariables} removes the data set variables
# #' from the memory and calls the garbage collection to free up
# #' memory - currently disabled
# #'
# #' @examples
# #' removeDataSetVariables()
# #'
# removeDataSetVariables <- function() {
#   # rm(list = ls(pattern = ”On_Time_On_Time_Performance*”,
#   #               envir = .GlobalEnv),
#   #    envir = .GlobalEnv)
#   # rm(list = ls(pattern = ”sr_*”,
#   #               envir = .GlobalEnv),
#   #    envir = .GlobalEnv)
#   # gc()
# }
#
#
# #'
# #' \code{loadSourceCodeFunctions} makes the functions created
# #' in different R files available for further use and process
# #' management
# #'
# #' @examples
# #' loadSourceCodeFunctions()
# #'
# loadSourceCodeFunctions <- function() {
#   source(”01-WildLiveStrikeDataSetDataPreparation.R”)
#   source(”02-OnTimeFlightPerformanceDataSetDataPreparation.R”)
#   source(”03-WildLifeStrikeDataSetSplitByYear.R”)
#   source(”04-OnTimeFlightPerformanceDataSetMergeByYear.R”)
#   source(”05-ExploreWildLifeStrikeDataSet.R”)
#   source(”06-ExploreOnTimeFlightPerformanceDataSet.R”)
#   source(”07-DescribeWildLifeStrikeDataSet.R”)
#   source(”08-DescribeOnTimeFlightPerformanceDataSet.R”)
#   source(”09-SelectWildLifeStrikeDataSet.R”)
#   source(”10-SelectOnTimeFlightPerformanceDataSet.R”)
#   source(”11-CleanupWildLifeStrikeDataSet.R”)
#   source(”12-CleanupOnTimeFlightPerformanceDataSet.R”)
#   source(”13-AirportDataSetDataPreparation.R”)
#   source(”14-DescribeAirportDataSet.R”)
#   source(”15-SelectAirportDataSet.R”)
#   source(”16-CleanupAirportDataSet.R”)
#   source(”17-DeriveAirportAttributes.R”)
```

```
#   source("18-IntegrateAttributes.R")
#   source("19-Model01.R")
#   source("20-Model02.R")
#   source("21-BuildModel02.R")
#   source("22-BuildPresentation.R")
# }
#
#
# #'
# #' \code{saveBarPlotPNG} saves the required bar plot based on
# #' the details in the YAML config file
# #'
# #' @param DataYear integer
# #' The year of the data set being used for the plot
# #'
# #' @param DataSet string
# #' The name of the data set the plot is being created from
# #'
# #' @param DataField string
# #' The name of the data field the plot is being created from
# #'
# #' @param DataStage string
# #' The name of the stage of the data
# #'
# #' @param DataObject object
# #' The data object to create the plot
# #'
# #' @examples
# #' saveBarPlotPNG(1990, "Animal Strike", "State", "01_Origin", DT)
# #'
# saveBarPlotPNG <- function(DataYear, DataSet, DataField, DataStage, DataObject) {
#   currentWorkingDir <- getwd()
#   setwd(getDocInputDir())
#   targetFileName <- paste(DataYear,
#                           "_",
#                           DataSet,
#                           "_",
#                           DataField,
#                           "_",
#                           DataStage,
#                           ".png",
#                           sep="")
#
#   plotText <- data.table(
#     keys = c(
#       "AC_CLASS",
#       "AC_MASS",
#       "TYPE_ENG",
#       "TIME_OF_DAY",
#       "PHASE_OF_FLT",
#       "SKY",
#       "PRECIP",
#       "Carrier",
#       "DistanceGroup"
```

```
#       ),
#     texts = c(
#       ”Aircraft class”,
#       ”Aircraft mass type”,
#       ”Engine type”,
#       ”Time of day”,
#       ”Flight phase”,
#       ”Sky condition”,
#       ”Precipitation”,
#       ”Airline carrier”,
#       ”Flight distance group”
#       )
#   )
#
#   if (!is.empty(tolower(plotText[keys==DataField,texts]))) {
#     lowerPlotText <- tolower(plotText[keys==DataField,texts])
#     labelAxisX <- plotText[keys==DataField,texts]
#   } else {
#     message(”Key not found”)
#     return()
#   }
#
#   plotTitle <- paste(”Data distribution of ”
#                    ,lowerPlotText,
#                    ” in ”,
#                    DataYear,
#                    sep=””)
#
#
#   test <- DataObject
#
#   ggplot(data = DataObject, aes(get(DataField))) +
#     ggtitle(plotTitle) + #plot title
#     geom_bar(fill = ”#99ccff”, color = ”#99ccff”) + #plotting a bar chart
#    coord_flip() + #flip the drawing of the axises --> Y will be the horizontal
#     #xlab(labelAxisX) + #set the vertical (coordflip!) axis text
#     xlab(””) + #set the vertical (coordflip!) axis text
#     ylab(””) + #set the horizontal (coordflip!) axis text
#     theme(
#       #align title to the center
#       plot.title = element_text(hjust = 0.5, face=”bold”),
#       #set plot background colors
#       plot.background = element_rect(fill = ”white”, colour = ”white”),
#       #set panel background colors
#       panel.background = element_rect(fill = ”white”, colour = ”white”),
#       #set the fonts to serif, which is set to Times New Roman
#       text = element_text(family = ”serif”),
#       #change the angle of the axis text
#       axis.text.x = element_text(angle=45, hjust=1, vjust=1)
#
#     )
#
#   ggsave(
#     targetFileName,
```

```
#     units = ”in”, #units are in pixels
#     width = 5, #width of the plot in in (should be the same as the height)
#     height = 5, #height of the plot in in (should be the same as the width)
#     dpi = 72 #nominal resolution in ppi (pixels per inch)
#   )
#
#   setwd(currentWorkingDir)
# }
#
#
# #'
# #' \code{saveMapPNG} saves the required state map
# #' the details in the YAML config file
# #'
# #' @param DataState string list
# #' The name of the state
# #'
# #' @param DataObject object
# #' The data object to create the map
# #'
# #' @examples
# #' saveMapPNG(”TX”, DT)
# #'
# saveMapPNG <- function(DataState, DataObject) {
#   currentWorkingDir <- getwd()
#   setwd(getDocInputDir())
#
#   stateNames <- data.table(
#     state = c(
#       ”AL”,
#       ”AK”,
#       ”AZ”,
#       ”AR”,
#       ”CA”,
#       ”CO”,
#       ”CT”,
#       ”DE”,
#       ”FL”,
#       ”GA”,
#       ”HI”,
#       ”ID”,
#       ”IL”,
#       ”IN”,
#       ”IA”,
#       ”KS”,
#       ”KY”,
#       ”LA”,
#       ”ME”,
#       ”MD”,
#       ”MA”,
#       ”MI”,
#       ”MN”,
#       ”MS”,
#       ”MO”,
```

```
#       ”MT”,
#       ”NE”,
#       ”NV”,
#       ”NH”,
#       ”NJ”,
#       ”NM”,
#       ”NY”,
#       ”NC”,
#       ”ND”,
#       ”OH”,
#       ”OK”,
#       ”OR”,
#       ”PA”,
#       ”RI”,
#       ”SC”,
#       ”SD”,
#       ”TN”,
#       ”TX”,
#       ”UT”,
#       ”VT”,
#       ”VA”,
#       ”WA”,
#       ”WV”,
#       ”WI”,
#       ”WY”
#     ),
#     stateName = c(
#       ”Alabama”,
#       ”Alaska”,
#       ”Arizona”,
#       ”Arkansas”,
#       ”California”,
#       ”Colorado”,
#       ”Connecticut”,
#       ”Delaware”,
#       ”Florida”,
#       ”Georgia”,
#       ”Hawaii”,
#       ”Idaho”,
#       ”Illinois”,
#       ”Indiana”,
#       ”Iowa”,
#       ”Kansas”,
#       ”Kentucky”,
#       ”Louisiana”,
#       ”Maine”,
#       ”Maryland”,
#       ”Massachusetts”,
#       ”Michigan”,
#       ”Minnesota”,
#       ”Mississippi”,
#       ”Missouri”,
#       ”Montana”,
#       ”Nebraska”,
```

```
#       ”Nevada”,
#       ”New Hampshire”,
#       ”New Jersey”,
#       ”New Mexico”,
#       ”New York”,
#       ”North Carolina”,
#       ”North Dakota”,
#       ”Ohio”,
#       ”Oklahoma”,
#       ”Oregon”,
#       ”Pennsylvania”,
#       ”Rhode Island”,
#       ”South Carolina”,
#       ”South Dakota”,
#       ”Tennessee”,
#       ”Texas”,
#       ”Utah”,
#       ”Vermont”,
#       ”Virginia”,
#       ”Washington”,
#       ”West Virginia”,
#       ”Wisconsin”,
#       ”Wyoming”
#     )
#   )
#
#   for (actualState in DataState){
#     targetFileName <- paste(”State_”,
#                             actualState,
#                             ”_Airports.png”,
#                             sep=””)
#
#
#     plotTitle <- paste(”Modeled airports in ”,
#                        stateNames[state==actualState,
#                                   ”stateName”],
#                        sep=””)
#
#     if (actualState == ”AK”){
#       base_map <- map_data(”world”, ”USA:alaska”)
#     } else if (actualState == ”HI”) {
#       base_map <- map_data(”world”, ”USA:hawaii”)
#     } else {
#       base_map <- map_data(”state”,
#                            tolower(stateNames[state==actualState,
#                                               ”stateName”])
#       )
#     }
#
#     temp <- DataObject
#     rm(temp)
#
#     ggplot()  +
#       geom_polygon(data=base_map,
```

```
#                     aes(x=long,
#                         y=lat,
#                         group=group),
#                     color="darkblue",
#                     fill = "#99CCFF") +
#        geom_point(aes(x = DataObject[State==actualState,
#                                      long],
#                       y = DataObject[State==actualState,
#                                      lat],
#                       size = DataObject[State==actualState,
#                                         StrikeNo]),
#                   color = "#FFCC99") +
#        theme_classic() +
#        ggtitle(plotTitle) +
#        xlab("") +
#        ylab("") +
#        scale_size_continuous(range = c(1, 6)) +
#        theme(
#          plot.title = element_text(hjust = 0.5,
#                                    face="bold"),
#          legend.position = "none",
#          axis.line = element_blank(),
#          axis.text = element_blank(),
#          axis.ticks = element_blank(),
#          text = element_text(family = "serif")
#        )
#
#      ggsave(
#        targetFileName,
#        units = "in", #units are in pixels
#       width = 5, #width of the plot in in (should be the same as the height)
#      height = 5, #height of the plot in in (should be the same as the width)
#        dpi = 72 #nominal resolution in ppi (pixels per inch)
#      )
#
#    }
#
#    setwd(currentWorkingDir)
#  }
#
#
#
#
# #'
# #' \code{printTable} prints an rmarkdown table based on the
# #' input variables
# #'
# #' @param Full boolean
# #' Flag to have the full data set or just the first year to print
# #'
# #' @param DataFile string
# #' The name of the RDS data file to load the data from
# #'
# #' @param ColumnNames string list
```

```
# #' The name of columns to be extracted from the data table
# #'
# #' @param ColumnTitles string list
# #' The titles the columns should have in the table
# #'
# #' @examples
# #' printTable(
# #'   TRUE,
# #'   ”example.rds”,
# #'   c(”V1”, ”V2”),
# #'   c(”Column1”, ”Column2”)
# #' )
# #'
# printTable <- function(Full, DataFile, ColumnNames, ColumnTitles) {
#
#   dataDir <- getDataDir()
#
#   RDSExpFile <- paste(dataDir,
#                       ”/”,
#                       DataFile,
#                       sep = ””)
#
#   dataTable <- readRDS(file = RDSExpFile)
#
#   if (Full == TRUE) {
#     kable(dataTable[, ..ColumnNames],
#           col.names = ColumnTitles,
#           align = ”c”)
#   } else {
#     kable(dataTable[1, ..ColumnNames],
#           col.names = ColumnTitles,
#           align = ”c”)
#   }
#
# }
#
#
# #'
# #' \code{getStates} returns the U.S. state abbreviations
# #'
# #' @examples
# #' getStates()
# #'
# getStates <- function() {
#   return(
#     c(
#       ”AL”,
#       ”AK”,
#       ”AZ”,
#       ”AR”,
#       ”CA”,
#       ”CO”,
#       ”CT”,
#       ”DE”,
```

```
#       ”FL”,
#       ”GA”,
#       ”HI”,
#       ”ID”,
#       ”IL”,
#       ”IN”,
#       ”IA”,
#       ”KS”,
#       ”KY”,
#       ”LA”,
#       ”ME”,
#       ”MD”,
#       ”MA”,
#       ”MI”,
#       ”MN”,
#       ”MS”,
#       ”MO”,
#       ”MT”,
#       ”NE”,
#       ”NV”,
#       ”NH”,
#       ”NJ”,
#       ”NM”,
#       ”NY”,
#       ”NC”,
#       ”ND”,
#       ”OH”,
#       ”OK”,
#       ”OR”,
#       ”PA”,
#       ”RI”,
#       ”SC”,
#       ”SD”,
#       ”TN”,
#       ”TX”,
#       ”UT”,
#       ”VT”,
#       ”VA”,
#       ”WA”,
#       ”WV”,
#       ”WI”,
#       ”WY”
#       )
#     )
# }
#
#
# #'
# #' \code{printStates} prints the U.S. state abbreviations and names
# #' in an rmarkdown table
# #'
# #' @examples
# #' printStates()
# #'
```

```
# printStates <- function() {
#
#   dataState <- data.table(
#     state = c(
#       "AL",
#       "AK",
#       "AZ",
#       "AR",
#       "CA",
#       "CO",
#       "CT",
#       "DE",
#       "FL",
#       "GA",
#       "HI",
#       "ID",
#       "IL",
#       "IN",
#       "IA",
#       "KS",
#       "KY",
#       "LA",
#       "ME",
#       "MD",
#       "MA",
#       "MI",
#       "MN",
#       "MS",
#       "MO"
#     ),
#     stateName = c(
#       "Alabama",
#       "Alaska",
#       "Arizona",
#       "Arkansas",
#       "California",
#       "Colorado",
#       "Connecticut",
#       "Delaware",
#       "Florida",
#       "Georgia",
#       "Hawaii",
#       "Idaho",
#       "Illinois",
#       "Indiana",
#       "Iowa",
#       "Kansas",
#       "Kentucky",
#       "Louisiana",
#       "Maine",
#       "Maryland",
#       "Massachusetts",
#       "Michigan",
#       "Minnesota",
```

```
#       ”Mississippi”,
#       ”Missouri”
#       ),
#     e1 = c(
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”
#     ),
#     e2 = c(
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
#       ” ”,
```

```
#       " ",
#       " "
#     ),
#     state2 = c(
#       "MT",
#       "NE",
#       "NV",
#       "NH",
#       "NJ",
#       "NM",
#       "NY",
#       "NC",
#       "ND",
#       "OH",
#       "OK",
#       "OR",
#       "PA",
#       "RI",
#       "SC",
#       "SD",
#       "TN",
#       "TX",
#       "UT",
#       "VT",
#       "VA",
#       "WA",
#       "WV",
#       "WI",
#       "WY"
#     ),
#     stateName2 = c(
#       "Montana",
#       "Nebraska",
#       "Nevada",
#       "New Hampshire",
#       "New Jersey",
#       "New Mexico",
#       "New York",
#       "North Carolina",
#       "North Dakota",
#       "Ohio",
#       "Oklahoma",
#       "Oregon",
#       "Pennsylvania",
#       "Rhode Island",
#       "South Carolina",
#       "South Dakota",
#       "Tennessee",
#       "Texas",
#       "Utah",
#       "Vermont",
#       "Virginia",
#       "Washington",
#       "West Virginia",
```

```
#       "Wisconsin",
#       "Wyoming"
#     )
#   )
#
#   kable(dataState,
#           col.names = c("Abbreviation","Name","","","Abbreviation","Name"),
#           align = "c")
#
# }
#
#
# #'
# #' \code{regeneratePlots} regenerates the plots based on the data sets
# #'
# #' @examples
# #' regeneratePlots()
# #'
# regeneratePlots <- function(){
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#
#   for (i in startYear:endYear){
#     RDSFileName_01 <- paste(i,
#                          "_Animal_Strikes_01_Orig.rds",
#                          sep = "")
#
#     RDSFile_01 <- paste(dataDir,
#                       "/",
#                       RDSFileName_01,
#                       sep = "")
#
#     #Read the data file into a variable
#     variableName_01 <- paste("AS_", i, sep="")
#     assign(variableName_01, readRDS(file = RDSFile_01))
#
#     #Save the plots as PNG files
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "AnimalStrike",
#                    DataField = "AC_CLASS",
#                    DataStage = "01_Orig",
#                    DataObject = get(variableName_01))
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "AnimalStrike",
#                    DataField = "AC_MASS",
#                    DataStage = "01_Orig",
#                    DataObject = get(variableName_01))
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "AnimalStrike",
#                    DataField = "TYPE_ENG",
#                    DataStage = "01_Orig",
#                    DataObject = get(variableName_01))
#     saveBarPlotPNG(DataYear = i,
```

```
#                     DataSet = "AnimalStrike",
#                     DataField = "TIME_OF_DAY",
#                     DataStage = "01_Orig",
#                     DataObject = get(variableName_01))
#       saveBarPlotPNG(DataYear = i,
#                     DataSet = "AnimalStrike",
#                     DataField = "PHASE_OF_FLT",
#                     DataStage = "01_Orig",
#                     DataObject = get(variableName_01))
#       saveBarPlotPNG(DataYear = i,
#                     DataSet = "AnimalStrike",
#                     DataField = "SKY",
#                     DataStage = "01_Orig",
#                     DataObject = get(variableName_01))
#       saveBarPlotPNG(DataYear = i,
#                     DataSet = "AnimalStrike",
#                     DataField = "PRECIP",
#                     DataStage = "01_Orig",
#                     DataObject = get(variableName_01))
#
#       #Free up the memory
#       rm(list = variableName_01)
#       rm(variableName_01)
#       gc()
#
#       RDSFileName_02 <- paste(i,
#                              "_Animal_Strikes_04_Cle.rds",
#                              sep = "")
#
#       RDSFile_02 <- paste(dataDir,
#                          "/",
#                          RDSFileName_02,
#                          sep = "")
#
#       #Read the data file into a variable
#       variableName_02 <- paste("AS_", i, sep="")
#       assign(variableName_02, readRDS(file = RDSFile_02))
#
#       #Save the plots as PNG files
#       saveBarPlotPNG(DataYear = i,
#                     DataSet = "AnimalStrike",
#                     DataField = "AC_CLASS",
#                     DataStage = "04_Cleaned",
#                     DataObject = get(variableName_02))
#       saveBarPlotPNG(DataYear = i,
#                     DataSet = "AnimalStrike",
#                     DataField = "AC_MASS",
#                     DataStage = "04_Cleaned",
#                     DataObject = get(variableName_02))
#       saveBarPlotPNG(DataYear = i,
#                     DataSet = "AnimalStrike",
#                     DataField = "TYPE_ENG",
#                     DataStage = "04_Cleaned",
#                     DataObject = get(variableName_02))
```

```
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "AnimalStrike",
#                    DataField = "TIME_OF_DAY",
#                    DataStage = "04_Cleaned",
#                    DataObject = get(variableName_02))
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "AnimalStrike",
#                    DataField = "PHASE_OF_FLT",
#                    DataStage = "04_Cleaned",
#                    DataObject = get(variableName_02))
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "AnimalStrike",
#                    DataField = "SKY",
#                    DataStage = "04_Cleaned",
#                    DataObject = get(variableName_02))
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "AnimalStrike",
#                    DataField = "PRECIP",
#                    DataStage = "04_Cleaned",
#                    DataObject = get(variableName_02))
#
#     #Free up the memory
#     rm(list = variableName_02)
#     rm(variableName_02)
#     gc()
#
#
#     RDSFileName_03 <- paste(i,
#                          "_On_Time_On_Time_Performance_01_Orig.rds",
#                          sep = "")
#
#     RDSFile_03 <- paste(dataDir,
#                      "/",
#                      RDSFileName_03,
#                      sep = "")
#
#     #Read the data file into a variable
#     variableName_03 <- paste("FP_", i, sep="")
#     assign(variableName_03, readRDS(file = RDSFile_03))
#
#     #Save the plots as PNG files
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "FlightData",
#                    DataField = "Carrier",
#                    DataStage = "01_Orig",
#                    DataObject = get(variableName_03))
#
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "FlightData",
#                    DataField = "DistanceGroup",
#                    DataStage = "01_Orig",
#                    DataObject = get(variableName_03))
#
#     #Free up the memory
```

```
#     rm(list = variableName_03)
#     rm(variableName_03)
#     gc()
#
#
#     RDSFileName_04 <- paste(i,
#                             "_On_Time_On_Time_Performance_04_Cle.rds",
#                             sep = "")
#
#     RDSFile_04 <- paste(dataDir,
#                         "/",
#                         RDSFileName_04,
#                         sep = "")
#
#     #Read the data file into a variable
#     variableName_04 <- paste("FP_", i, sep="")
#     assign(variableName_04, readRDS(file = RDSFile_04))
#
#     #Save the plots as PNG files
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "FlightData",
#                    DataField = "Carrier",
#                    DataStage = "04_Cleaned",
#                    DataObject = get(variableName_04))
#
#     saveBarPlotPNG(DataYear = i,
#                    DataSet = "FlightData",
#                    DataField = "DistanceGroup",
#                    DataStage = "04_Cleaned",
#                    DataObject = get(variableName_04))
#
#     #Free up the memory
#     rm(list = variableName_04)
#     rm(variableName_04)
#     gc()
#
#   } #end of "for (i in startYear:endYear)"
#
# }
#
#
# #'
# #' \code{convertToDMSNumber}
# #'
# #' @param inputString string
# #' The DMS (Degrees Minutes Seconds) input string in the following format:
# #' DD-MM-SS.####c
# #'
# #' @return the numeric value of the latitude / longitude received
# #' in a character string
# #'
# #' @examples
# #' convertToDMSNumber("")
# #'
```

```
#
# convertToDMSNumber <- function(inputString){
#
#   getDot <- regexpr(pattern ='\\.',inputString)
#   getLastChar <- substring(inputString, nchar(as.character(inputString)))
#
#   return(
#     inputString %>%
#       sub('-', 'd', .) %>%
#       sub('-', '\'', .) %>%
#       substr(. , 1, getDot-1) %>%
#       paste(.,'”', getLastChar, sep = ””) %>%
#       char2dms %>%
#       as.numeric
#     )
# }
#
# #'
# #' \code{saveModelingHistogramPNG} saves the required histogram
# #'
# #' @param FieldName string
# #' The field name
# #'
# #' @param DataObject object
# #' The data to create the plot
# #'
# #' @param BinSize number
# #' The bin size to be used in the plot
# #'
# #' @examples
# #' saveModelingHistogramPNG(”Field”, DT, 20)
# #'
# saveModelingHistogramPNG <- function(FieldName, DataObject, BinSize) {
#   currentWorkingDir <- getwd()
#   setwd(getDocInputDir())
#   targetFileName <- paste(”Histogram_of_”,
#                           FieldName,
#                           ”.png”,
#                           sep=””)
#
#   plotText <- data.table(
#     keys = c(
#       ”StrikeNo”,
#       ”OriginCount”,
#       ”OriginMaxDistance”,
#       ”OriginMinDistance”,
#       ”OriginAvgDistance”,
#       ”DestinationCount”,
#       ”DestinationMaxDistance”,
#       ”DestinationMinDistance”,
#       ”DestinationAvgDistance”,
#       ”ARPElevation”,
#       ”LandAreaCoveredByAirport”
#     ),
```

```
#     texts = c(
#       ”number of animal strikes”,
#       ”number of flights originated”,
#       ”maximum flight distance originated”,
#       ”minimum flight distance originated”,
#       ”average flight distance originated”,
#       ”number of flights departed”,
#       ”maximum flight distance departed”,
#       ”minimum flight distance departed”,
#       ”average flight distance departed”,
#       ”airport elevation”,
#       ”land covered by the airport”
#     )
#   )
#
#   if (!is.empty(tolower(plotText[keys==FieldName,texts]))) {
#     lowerPlotText <- tolower(plotText[keys==FieldName,texts])
#     labelAxisX <- plotText[keys==FieldName,texts]
#   } else {
#     message(”Key not found”)
#     return()
#   }
#
#   plotTitle <- paste(”Histogram of ”
#                    ,lowerPlotText,
#                    sep=””)
#
#   test <- DataObject
#
#   ggplot(data=modelData, aes(get(FieldName))) +
#     geom_histogram(binwidth = BinSize, fill = ”#99ccff”, color = ”white”) +
#     ggtitle(plotTitle) + #plot title
#     xlab(””) + #set the vertical (coordflip!) axis text
#     ylab(””) + #set the horizontal (coordflip!) axis text
#     theme(
#       #align title to the center
#       plot.title = element_text(hjust = 0.5, face=”bold”),
#       #set plot background colors
#       plot.background = element_rect(fill = ”white”, colour = ”white”),
#       #set panel background colors
#       panel.background = element_rect(fill = ”white”, colour = ”white”),
#       #set the fonts to serif, which is set to Times New Roman
#       text = element_text(family = ”serif”),
#       #change the angle of the axis text
#       axis.text.x = element_text(angle=45, hjust=1, vjust=1)
#     )
#
#   ggsave(
#     targetFileName,
#     units = ”in”, #units are in pixels
#     width = 5, #width of the plot in in (should be the same as the height)
#     height = 5, #height of the plot in in (should be the same as the width)
#     dpi = 72 #nominal resolution in ppi (pixels per inch)
#   )
```

```
#
#   setwd(currentWorkingDir)
# }
#
# #'
# #' \code{saveModelingHistogramLogPNG} saves the required histogram
# #'
# #' @param FieldName string
# #' The field name
# #'
# #' @param DataObject object
# #' The data to create the plot
# #'
# #' @param BinSize number
# #' The bin size to be used in the plot
# #'
# #' @examples
# #' saveModelingHistogramLogPNG(”Field”, DT, 20)
# #'
# saveModelingHistogramLogPNG <- function(FieldName, DataObject, BinSize) {
#   currentWorkingDir <- getwd()
#   setwd(getDocInputDir())
#   targetFileName <- paste(”Histogram_of_log_”,
#                           FieldName,
#                           ”.png”,
#                           sep=””)
#
#   plotText <- data.table(
#     keys = c(
#       ”StrikeNo”,
#       ”OriginCount”,
#       ”OriginMaxDistance”,
#       ”OriginMinDistance”,
#       ”OriginAvgDistance”,
#       ”DestinationCount”,
#       ”DestinationMaxDistance”,
#       ”DestinationMinDistance”,
#       ”DestinationAvgDistance”,
#       ”ARPElevation”,
#       ”LandAreaCoveredByAirport”
#     ),
#     texts = c(
#       ”number of animal strikes”,
#       ”number of flights originated”,
#       ”maximum flight distance originated”,
#       ”minimum flight distance originated”,
#       ”average flight distance originated”,
#       ”number of flights departed”,
#       ”maximum flight distance departed”,
#       ”minimum flight distance departed”,
#       ”average flight distance departed”,
#       ”airport elevation”,
#       ”land covered by the airport”
#     )
```

```
#   )
#
#   if (!is.empty(tolower(plotText[keys==FieldName,texts]))) {
#     lowerPlotText <- tolower(plotText[keys==FieldName,texts])
#     labelAxisX <- plotText[keys==FieldName,texts]
#   } else {
#     message("Key not found")
#     return()
#   }
#
#   plotTitle <- paste("Histogram of "
#                      ,lowerPlotText,
#                      " (log)",
#                      sep="")
#
#   test <- DataObject
#
#   ggplot(data=modelData, aes(log(get(FieldName)))) +
#     geom_histogram(binwidth = BinSize, fill = "#99ccff", color = "white") +
#     ggtitle(plotTitle) + #plot title
#     xlab("") + #set the vertical (coordflip!) axis text
#     ylab("") + #set the horizontal (coordflip!) axis text
#     theme(
#       #align title to the center
#       plot.title = element_text(hjust = 0.5, face="bold"),
#       #set plot background colors
#       plot.background = element_rect(fill = "white", colour = "white"),
#       #set panel background colors
#       panel.background = element_rect(fill = "white", colour = "white"),
#       #set the fonts to serif, which is set to Times New Roman
#       text = element_text(family = "serif"),
#       #change the angle of the axis text
#       axis.text.x = element_text(angle=45, hjust=1, vjust=1)
#     )
#
#   ggsave(
#     targetFileName,
#     units = "in", #units are in pixels
#     width = 5, #width of the plot in in (should be the same as the height)
#     height = 5, #height of the plot in in (should be the same as the width)
#     dpi = 72 #nominal resolution in ppi (pixels per inch)
#   )
#
#   setwd(currentWorkingDir)
# }
#
# #'
# #' \code{saveModelingHistogramLogPNG} saves the required histogram
# #'
# #' @param AirlineCode string
# #' The code of the airline
# #'
# #' @return the airline name in uppercase
# #'
```

```
# #' @examples
# #' saveModelingHistogramLogPNG("Field", DT, 20)
# #'
# getAirlineName <- function(AirlineCode) {
#
#   airlines <- data.table(
#     CarrierCode = c(
#       "9E",
#       "AA",
#       "AQ",
#       "AS",
#       "B6",
#       "CO",
#       "DH",
#       "DL",
#       "EV",
#       "F9",
#       "FL",
#       "HA",
#       "HP",
#       "MQ",
#       "NK",
#       "NW",
#       "OH",
#       "OO",
#       "PA",
#       "TW",
#       "TZ",
#       "UA",
#       "US",
#       "VX",
#       "WN",
#       "XE",
#       "YV",
#       "EA",
#       "ML",
#       "KH"
#     ),
#     CarrierName = c(
#       "Pinnacle",
#       "American Airlines",
#       "Aloha Airlines",
#       "Alaska Airlines",
#       "JetBlue Airways",
#       "Continental Airlines",
#       "Atlantic Coast Airlines",
#       "Delta Air Lines",
#       "Atlantic Southeast",
#       "Frontier Airlines",
#       "AirTran Airways",
#       "Hawaiian Air",
#       "America West Airlines",
#       "American Eagle Airlines",
#       "Spirit Airlines",
```

```
#       ”Northwest Airlines”,
#       ”Comair Airlines”,
#       ”SkyWest Airlines”,
#       ”Pan Am”,
#       ”Trans World Airlines”,
#       ”ATA Airlines”,
#       ”United Airlines”,
#       ”US Airways”,
#       ”Virgin America”,
#       ”Southwest Airlines”,
#       ”ExpressJet Airlines”,
#       ”Mesa Airlines”,
#       ”Eastern Airline”,
#       ”Midway Airlines”,
#       ”Aloha Air Cargo”
#     )
#   )
#
#   return(toupper(airlines[CarrierCode == AirlineCode,CarrierName]))
#
# }
```

References

Gergely Daróczi, Renáta Németh, and Gergely Tóth. 2015. Mastering Data Analysis with R. First edition. Birmingham, UK: Packt Publishing.

Shearer, Colin. 2000. "The Crisp-Dm Model - the New Blueprint for Data Mining." Journal of Data Warehousing 5 (4): 13–22.