OPERABILITY DEPARTMENT (EIO43)

INTERNSHIP FINAL REPORT

LIMA User Guide

Author:
Federico Semeraro

Manager:
Andy Williams

June 21, 2016

Abstract

This report has the objective to explain how to use LiMA (Lightweight Maturity and Availability) and to give a quick introduction to the main R-Shiny App syntax as well as its main functions. It is meant to be for either new users or future developers.

Contents

1. Introduction

As for the majority of industrial giants, Airbus is striving to introduce digitalization as a core component of the full life-cycle of its products, from design to in-service utilization. During my year placement, I have been asked to explore different sides of Big Data analytic: I have witnessed and contributed to the early stage of it, starting from data acquisition from online databases, such as FlightAware and FlightRadar24 (FR24), and processing it using Excel VBA in order to compare their capabilities, up to experimenting with two days of raw data from FR24 using R. It was only in early March when, during the exploration process of R's capabilities, I came across a very powerful library called Shiny, which allows the creation of web applications. This completely revolutionized my work, allowing me to create a stand-alone application.

The information that FR24 provides comes from the collection of several data sources including ADS-B, MLAT and radar data. The first one is the primary technology it relies on and it works in the following way: the aircraft receives its GPS location from a satellite which is then retransmitted via the ADS-B transponder on-board together with several other information (e.g. Registration number, Flight Number etc.); this signal is then picked up by a receiver connected to FR24 (this service is strongly dependent on private ADS-B receivers) and the database is assembled.

The two years of raw data that Airbus has purchased contains mistakes, inconsistencies and gaps, therefore requiring processing. Here is where my role lies: creating smart algorithms to process data can enable effective visualizations and statistics useful to a very wide range of applications. The airlines' usage of Airbus products can be studied and compared to the performance of other manufacturers, which can then influence engineering and marketing decisions. In this report, the main features of the application will be discussed, going in depth on how they were programmed, and some ideas for future features will be proposed.

2. Installation and Requirements

In order to make LiMA a standalone app, we need R-Portable and Google Chrome Portable, linked to each other through a VBScript. However, the app was developed using RStudio, the most common R editor, which has an intuitive "Run-App" button via which you can run the app.R file; this launches the app on your default web-browser (see Appendix to launch the app in another browser).

In order to run LiMA in RStudio for the first time there are a few steps that have to be followed and some requirements:

STEP 1: Software/ Driver Installation:
- Install RStudio from https:/ / www.rstudio.com/ products/ rstudio/ download/
- Install Oracle Client 10g 32bit Win64 available in PC Services

STEP 2: Connection Setup:
- FR24 database connection: add a new connection to ODBC Data Source in the User DSN tab; choose SQL server, then set Name and Description to "FR24", Server to "fr0-csdb-p02,10001" (the server or password might be subjected to changes over time, this is referred to June 2015)
- OAG schedules database connection (see screenshots in Appendix): since this database requires 32bit drivers, open the following C:\ Windows\ SysWOW64\ odbcad32.exe and add a new User DSN; choose Oracle in OraClient 10g server (or Oracle in OraClient 11g_home1); set as follows: Data Source Name to "OAG", TNS Service Name to "DBUPMF30" and User ID to "AIRBUS". By clicking Test Connection you should be able to connect by setting the password to "OAG_MF30" (last step not required but good for checking the connection).

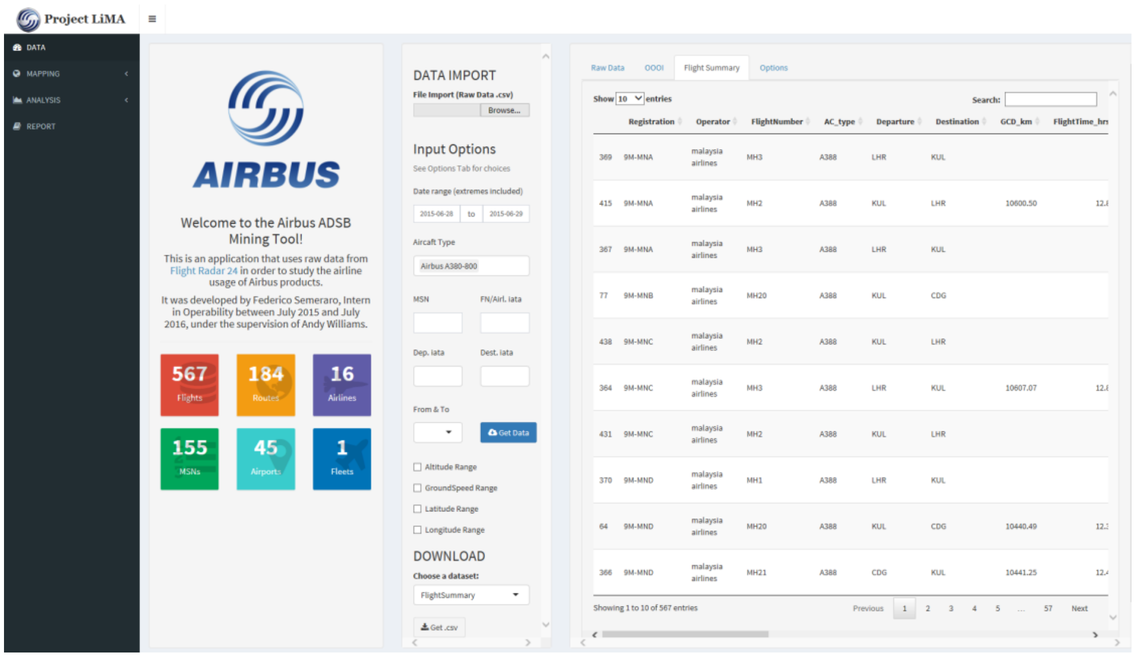
STEP 3: First time RStudio is opened:
- Go to Tools\ Global Options and then change the R version in the General tab to [Default] [32-bit] C:\ Program Files\ R\ R-3.2.2 if not already selected
- Run the app via the Run App button and wait while the packages are installed (you can find the list of packages required in "setup.R" script, which also imports all the functions required for the app). In case any package fails to download or install see Appendix for how to use the function "install_local".

In order to install and launch the app as a standalone, i.e. without RStudio, you just need to open the LiMA installation file and follow the installation steps. Note that this procedure is still highly experimental and should be developed further (especially for the fact that whenever Google Chrome browser is closed, R process has to be terminated through the Tast Manager). Moreover, note that step 2, i.e. the connection setup, still has to be completed (in further developments this should be integrated and automated within LiMA installation).

3. Main Features

The application is in a folder called ./ LiMA/ FR24_AnalysisTool and it comprises of the app.R file, the main source code, and four subfolders: "Data", which contains the airport features (see Section 2.2.1), weatherdata.rds and info.RData that contains the lists of MSNs, routes etc.; "Functions & Scripts", containing the different functions called in the app.R file and later discussed; "Particular Packages", which collects the packages not available in CRAN (the main R-packages repository) and that have to be installed locally (see Appendix A); "www" contains the logos and images used in the app. A shiny app (i.e. app.R code) is divided into two different pieces of code that communicate to each other: the user interface (UI) and the server. These two codes can be either split in two files (namely ui.R and server.R) or linked together in an app.R file. The user interface deals with inputs and outputs to and from the server, whereas the server either calculates or calls the functions to calculate the data outputted in the ui.

3.1 Data Import and Processing



Welcome to the Airbus ADSB Mining Tool!

This is an application that uses raw data from Flight Radar 24 in order to study the airline usage of Airbus products.

It was developed by Federico Semeraro, Intern in Operability between July 2015 and July 2016, under the supervision of Andy Williams.

567 Flights	184 Routes	16 Airlines
155 MSNs	45 Airports	1 Fleets

DATA IMPORT

File Import (Raw Data .csv) Browse...

Input Options

See Options Tab for choices

Date range (extremes included): 2015-06-28 to 2015-06-29

Aircaft Type: Airbus A380-800

MSN

FN/Airl. iata

Dep. iata

Dest. iata

From & To

Get Data

☐ Altitude Range

☐ GroundSpeed Range

☐ Latitude Range

☐ Longitude Range

DOWNLOAD

Choose a dataset: FlightSummary

Get .csv

Raw Data | OOOI | Flight Summary | Options

Show 10 entries Search:

	Registration	Operator	FlightNumber	AC_type	Departure	Destination	GCD_km	FlightTime_hrs
369	9M-MNA	malaysia airlines	MH3	A388	LHR	KUL		
415	9M-MNA	malaysia airlines	MH2	A388	KUL	LHR	10600.50	12.
367	9M-MNA	malaysia airlines	MH3	A388	LHR	KUL		
77	9M-MNB	malaysia airlines	MH20	A388	KUL	CDG		
438	9M-MNC	malaysia airlines	MH2	A388	KUL	LHR		
364	9M-MNC	malaysia airlines	MH3	A388	LHR	KUL	10607.07	12.
431	9M-MNC	malaysia airlines	MH2	A388	KUL	LHR		
370	9M-MND	malaysia airlines	MH1	A388	LHR	KUL		
64	9M-MND	malaysia airlines	MH20	A388	KUL	CDG	10440.49	12.
366	9M-MND	malaysia airlines	MH21	A388	CDG	KUL	10441.25	12.

Showing 1 to 10 of 567 entries

Previous 1 2 3 4 5 … 57 Next

Figure 1. LiMA front page

As you launch the app, the front page appears prompting the user to choose some data import options (Date range, Aircraft Type, MSN etc.); in order to know what the database contains, the "Options" tab on the right shows all the lists of data available that can be chosen in the "Input Options". Once the options have been selected, the "Get

Data" button sends the call through SQL to retrieve the raw data and then calls the processing functions directly. Once these steps are completed, the different tabs of data showing the data processing steps appear on the right and some more generic information about the data downloaded appear under the App description on the left. An important aspect to note is that .csv files already downloaded can be imported in the App directly via the "Browse" button at the top of the Data import panel. Moreover, datasets that are downloaded via SQL can be saved through the "Get .csv" button at the bottom of the same panel.

3.1.1 Programming Explanation

dataimport.R

The function takes eleven inputs (altitude, groundspeed, latitude, longitude, fleet, msn, flight number, departure and destination airports or a single airport and the date range). After this, everything the function does is assembling the SQL query by first selecting the raw data columns by default and then adding the input information by pasting the filtering options in the required style either via loops or not depending on the kind of information (e.g. altitude directly, fleets with a loop). It is important to note that the inputs are NULL if the input options haven't been chosen, so each for-loop checks if the input "!is.null". Finally, the SQL query is used to retrieve data via "odbcConnect" and "sqlQuery" functions. The raw data is then turned into a dataframe and calls it "fr24data".

DP1.R

This function processes fr24data and it is the first step of data processing. First, it orders it by timestamp and then by registration number, so that now we have the flights registration by registration in chronological order. Then, the row is captured and the time difference between each signal and the previous one is calculated (usually less than 10 seconds). After this, the first very important step is the On-ground estimation: this is estimated via the Altitude column, as expected, but also via the two neighboring signals (i.e. if the altitude suddenly drops for only one signal, this is filtered out – this has often been observed). The second fundamental step is the OFF and ON estimation: this is again fairly straightforward since it depends on the change of the new On-ground parameter from 1 to 0, then checking that the registration number of the aircraft and other parameters are constant. It is important to note that, in order to flag the identified OFF and ON signals, their timestamps are passed to the already initialized "OFFtimestamp" and "ONtimestamp" columns. The last step in this function is the identification of OUT and IN signal "candidates": this is made by checking the "TimeDiff" parameter and making a reasonable stop assumption between two subsequent flights of at least 15 minutes and then checking only if they have the same registration number in this case (this is because once aircraft are on the ground, they often stop transmitting the flight number and several other parameters).

DP2.R

This function is the second step of data processing and it accepts the already processed fr24data. The function creates a new dataframe, called "oooi" by subsetting fr24data, keeping the OFF and ON signals and the OUT and IN candidates (using the "subset" function, one of the most useful and effective in R). A fundamental step now is passing the OFF timestamp to the ON row (by checking again msn, flight number etc.). This is important because it is the method used to identify a flight cycle (see Figure2): the ON time is "referenced" using the OFF time; similarly the OUT and IN times are referenced using the OFF and ON time respectively, becoming a sort of chain of events. From the OUT and IN candidates, the closest to the OFF and ON times are chosen and then others filtered out.

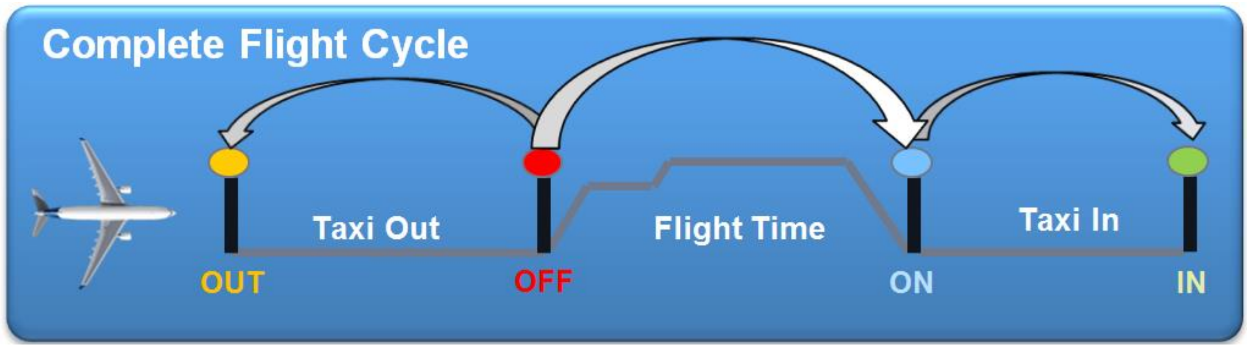


Figure 2. Complete Flight Cycle with event color identifiers used in the app

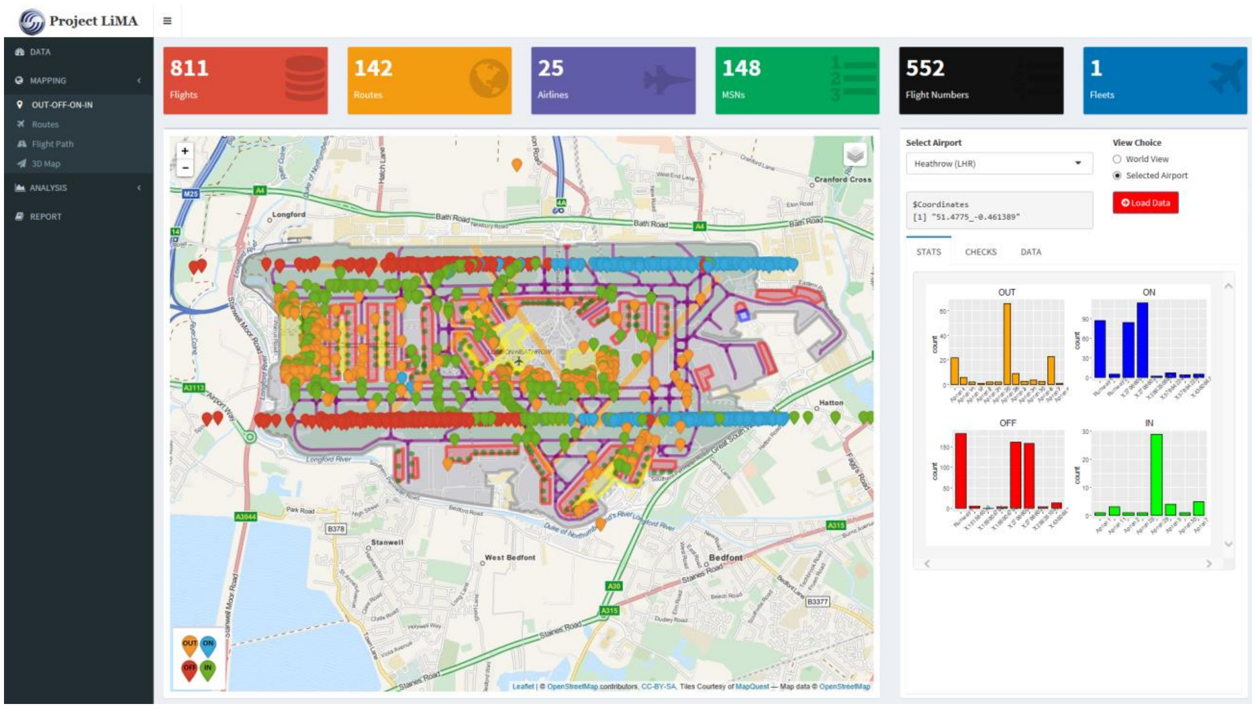
DP3.R

The last function accepts oooi and it has the objective to create the "FlightSummary" dataframe where each row is now a flight. First, some temporary dataframes are created, one for each event identified (e.g. OUTdata); these are then merged to each other (using the "merge" function, again another of the most powerful ones). From this point on, the newly created FlightSummary goes through post-processing: timestamps are transformed from seconds from the 1st January 1970 to a readable format using POSIXct and format functions and then the dataframe is cleaned deleting the possible errors. After this, FlightSummary gets merged both by registration numbers and by IATA (first two letters in FlightNumber column) with the MSNs dataframe contained in info.RData which includes Operators. Then, the flight time, taxi times, turnaround time and great-circle distance are calculated and post-processed. Finally, the dataframe is merged with METAR weather data coming from WUnderground by rounding the OFF and ON times to the closest half hour, since the weather data comes with this frequency. The columns of FlightSummary are then reordered and renamed appropriately.

3.2 Mapping

Statistics and tables of data are useful and often effective, but it is when data is linked and plotted onto maps that it comes alive and shows all the potential and capabilities that it carries. It is therefore clear why the mapping features are the ones that took a significant part of my time to develop.

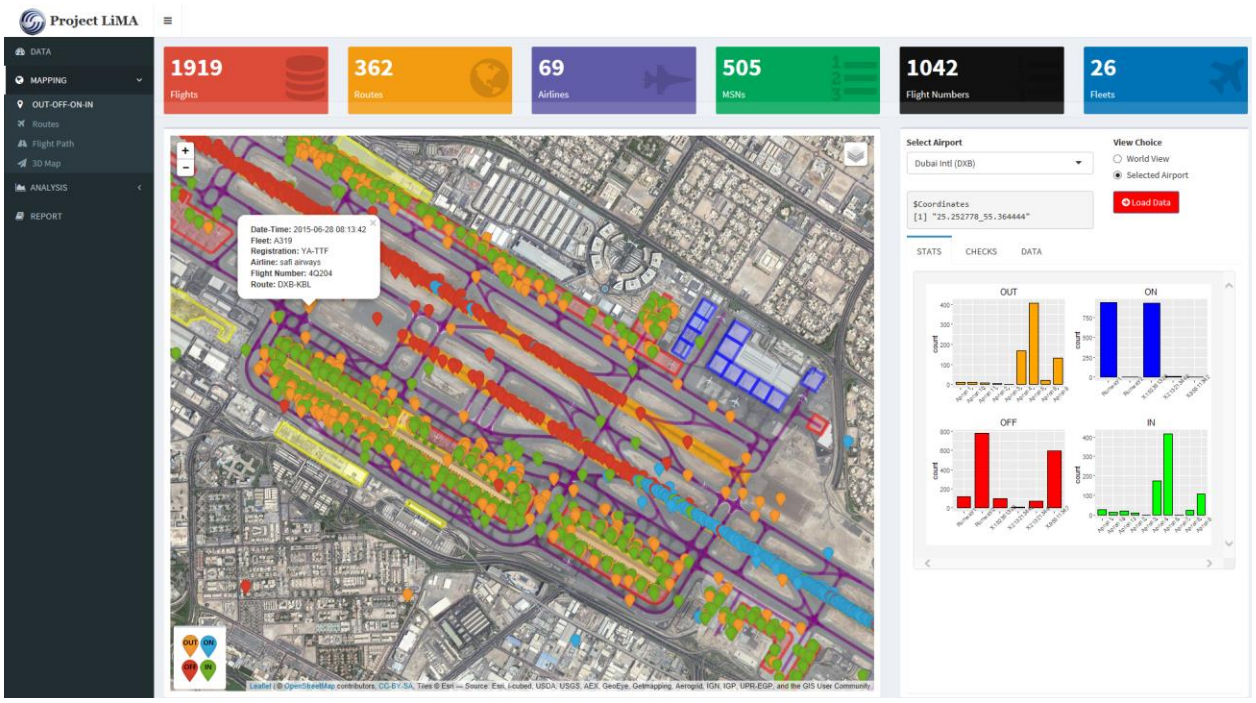
3.2.1 OUT-OFF-ON-IN Map



```
$Coordinates
[1] "51.4775_-0.461389"
```

Figure 3a. OUT-OFF-ON-IN Map for A320 data at LHR on 28-29 June 2015

As the first one of the mapping tabs is opened, a map of the world from Open Street Map appears. Onto this map the places of the events discussed in the DP3.R in the previous section can be plotted. The airport selection is in the top right corner, where the user can choose the airport that needs to be analyzed and zoom directly onto it. By clicking the “Load Data” button, the OOOI locations appear together with the airport features of the airport selected. These locations are checked as they are imported looking for overlaps and nearness: if these are found, they are reported in the histograms and “Checks” table below the airport selection. Another aspect to note are the info boxes at the top of the page, which update reactively as the user zooms in or out a specific location (the data visualized can also be found in the “Data” table). Finally, the background map can be changed to satellite or a few other views via the layer control in the top right corner of the map (See Figure 3a and 3b).



Date-Time: 2015-06-28 08:13:42
Fleet: A319
Registration: YA-TTF
Airline: safi airways
Flight Number: 4Q204
Route: DXB-KBL

```
$Coordinates
[1] "25.252778_55.364444"
```

Figure 3b. Another view for all flights to and from DXB on 28-29 June 2015

3.2.1.1 Programming Explanation

To develop this and the next two features, a library called “Leaflet” was used extensively. This links R programming to maps from Open Street Map (OSM) and it allows plotting layers onto them. For this particular application, features from OSM were downloaded using another library called “Osmar” which lets you store the airport features as “sp” files, i.e. either as SpatialPolygons, lines or points (for a detailed procedure see Appendix). These are then imported into the app as event reactive objects that change according to the airport chosen and imported using “readRDS”. After this, the icons are created, which correspond to the legend in the bottom left corner of the map. Finally, map bounds are used to filter FlightSummary data in order to output the data displayed info in the boxes at the top and the “Data” table on the right. Once these objects are plotted together with the OOOI points, the “airportchecks.R” function is run on the points and the airport features. This complex function checks if the OOOI data is contained or is near to any airport feature. It uses “point.in.polygon” and “gDistance” functions for aprons (SpatialPolygons) and runways (SpatialLines) to check for overlap or check the distance respectively. In the second case, if then the distance is less than 10^{-4} (which is what it was observed to be a reasonable assumption for runways to say that an OFF or ON point is on them). After that, histograms and “Checks” table are created and outputted to the UI.

3.2.2 Routes Map

In order to run the second tab of mapping, the button “Plot Data” has to be pressed. This processes the fr24data, synchronizing the points by interpolating them to the same time reference. The user can scroll the time variable, following the path of the flights throughout the day. It is interesting to note the flight patterns that arise as compared to the flight traffic density. To track the latter parameter, an info box is placed at the top of the map, showing the amount of flights currently displayed. The playback capability has been introduced to this feature, even though it still results in a motion whose fluidity should be improved.

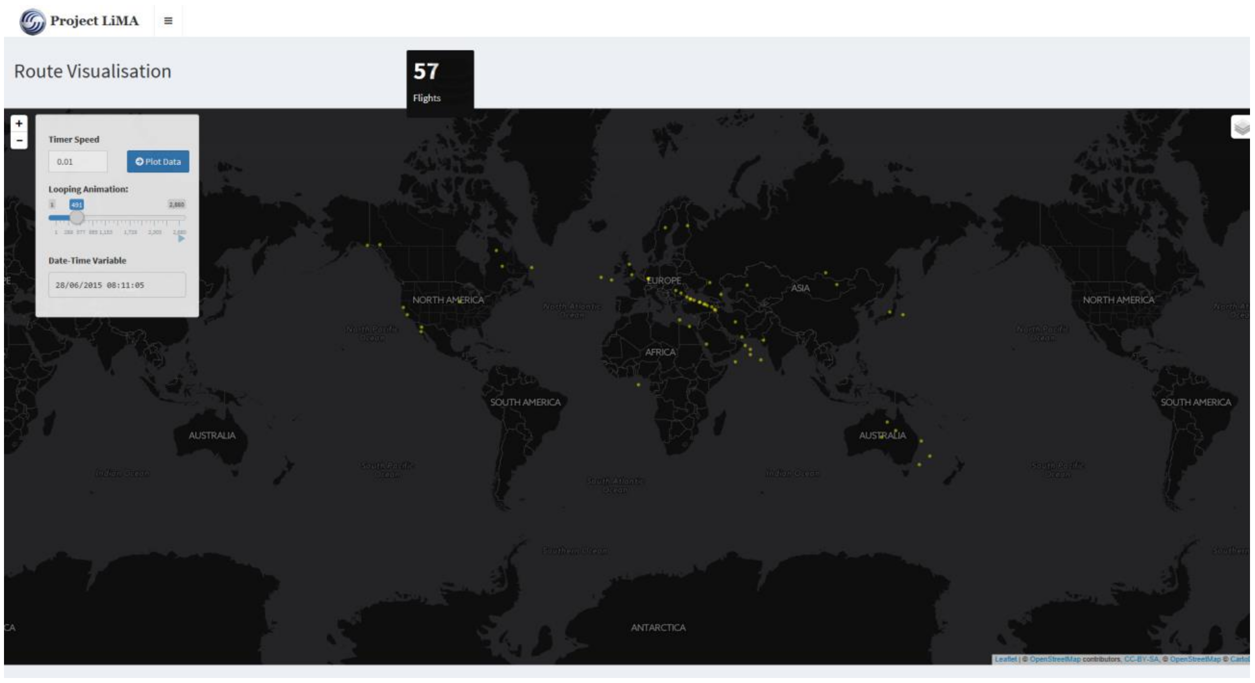
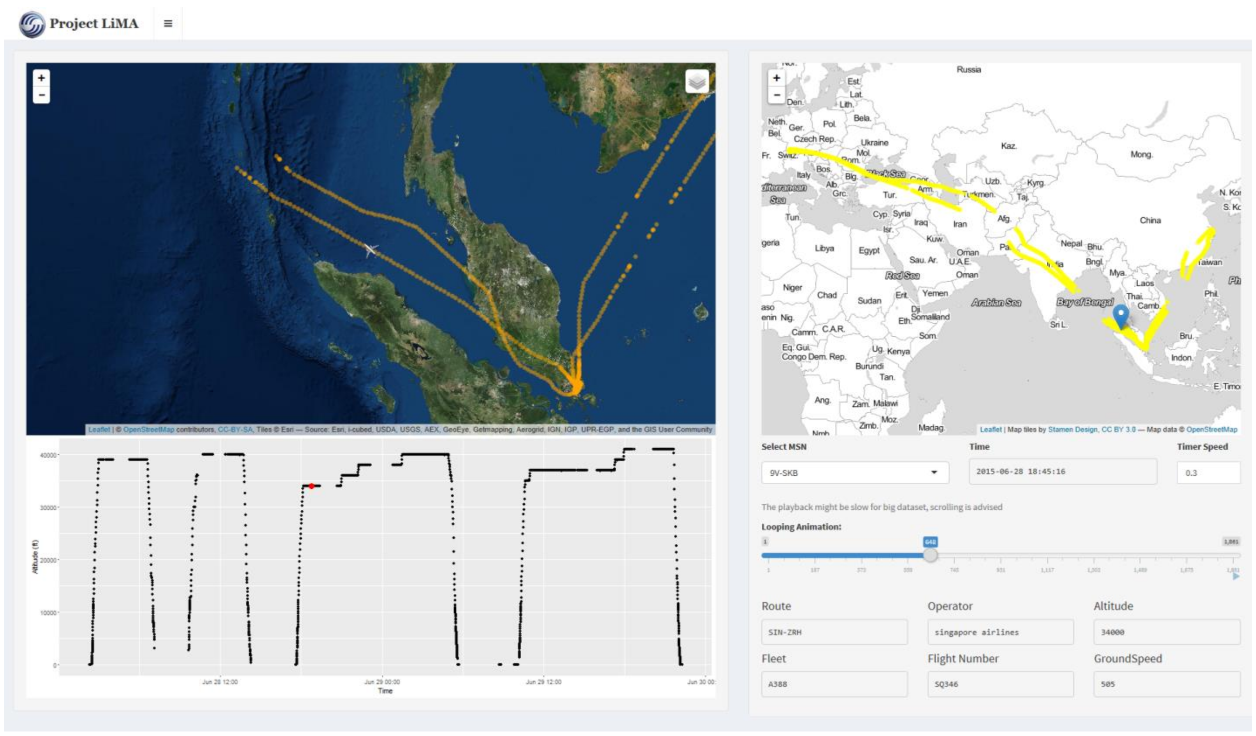


Figure 4. Routes Map for A380 data on 28-29th June 2015

3.2.2.1 Programming Explanation

The main challenge for this feature was to synchronize fr24data to the same time reference. This task is done in the “fr24datainterp.R” function, which is called when action button “Plot Data” is pressed. The function has runs with a for-loop, which makes it relatively slow to run on a big dataset. Since it is said that an expert R-programmer should *never* use loops, but only exploit R powerful functions, the task to come up with a different solution is left as a future task to be accomplished. However, as far as the current function is concerned, it takes the minimum and maximum OUT and IN timestamps from fr24data and it divides their difference by 60, which therefore gives the amount of minutes in the timeframe. After this, a new matrix called "timeframe" is created, which has the time reference on the first column with a 10 minutes step and then, via the loop, the linear interpolation of each flight cycle's latitude and longitude is added next to each other. The data for the flight cycle is selected from fr24data by using the OUT and IN row numbers from the FlightSummary dataframe.

3.2.3 Flight Path Map



Select MSN: 9V-SKB
Time: 2015-06-28 18:45:16
Timer Speed: 0.3
Route: SIN-ZRH
Operator: singapore airlines
Altitude: 34000
Fleet: A388
Flight Number: SQ346
GroundSpeed: 505

Figure 5. Flight Path Map for A380 on SIN-ZHR route on the 28th June 2015

The third mapping feature (the last one having been developed) has the objective of digging deeper in the raw data. It has the appearance of a control platform, with two maps at the top checking the flight path with different resolutions, the left one with a satellite zoomed view and the right one showing the whole route. Beneath the satellite map, the altitude is plotted against the timestamp, so that it allows another view on the flight pattern. Information about the flight status and details is shown on the bottom right corner. Finally, the user can scroll the time variable and follow the flight path reactively and change the registration number selected. Further capabilities about statistics on the altitude are currently being considered.

3.2.3.1 Programming Explanation

This feature is relatively straightforward but it has been very beneficial to track the performance of the core algorithms on fr24data since the oooi events can be plotted on the altitude vs time plot, therefore identifying the flights that haven't been identified. It uses again leaflet for the background maps and ggplot2 for the altitude tracking, both made reactive through Shiny.

3.2.4 3D Globe

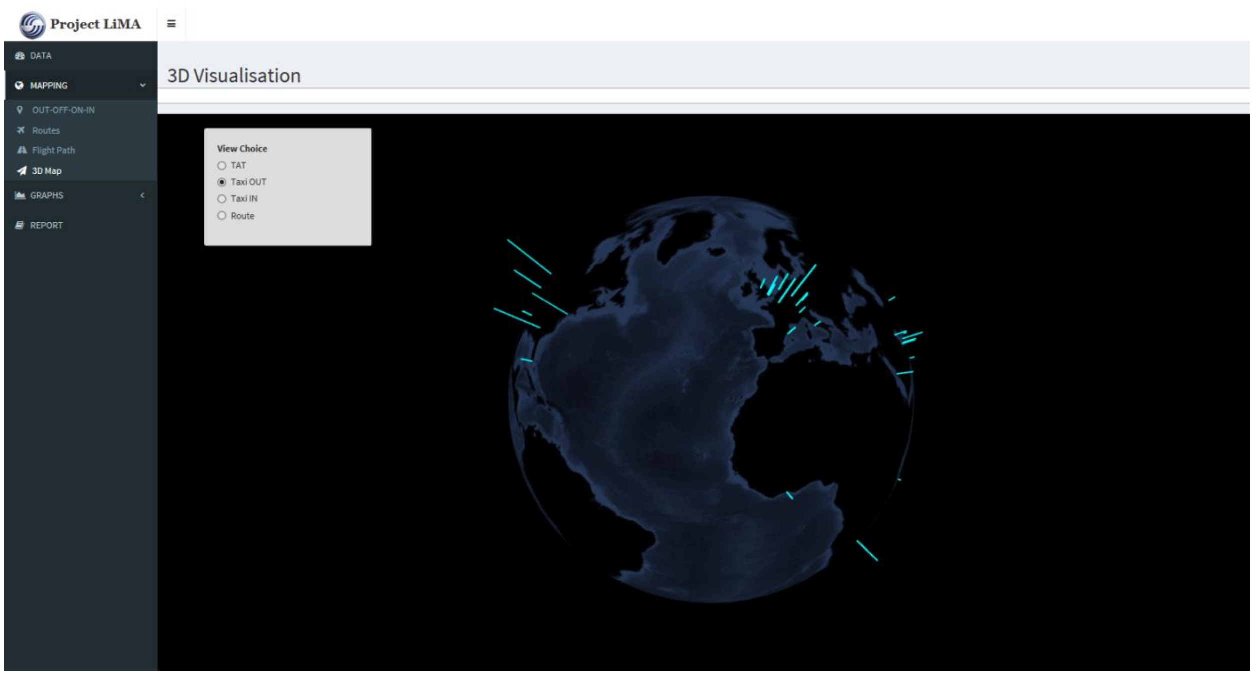


Figure 6. 3D Globe for A380 Taxi OUT for the 28-29th June 2015

The fourth and last mapping feature is a 3D globe onto which layers of data in the form of parallelepiped. The user can choose either to plot TAT, taxi times or the flight paths of a specific registration number using the panel on the top left.

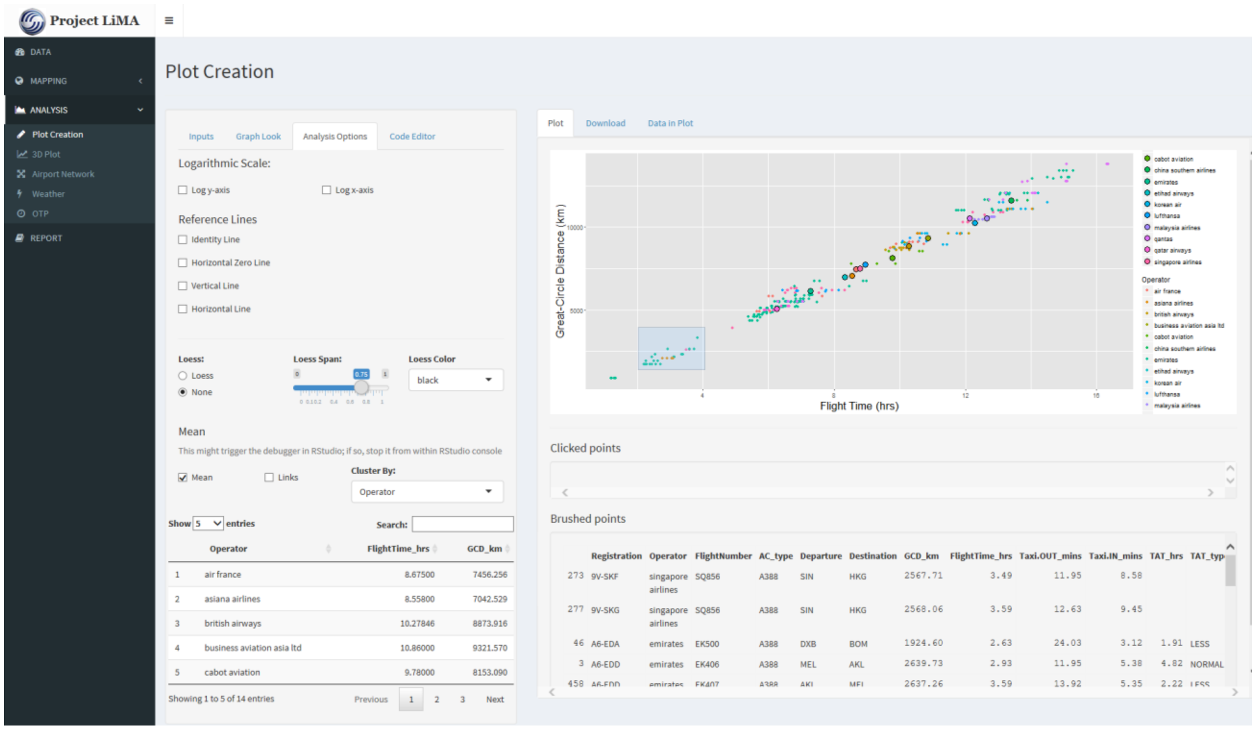
3.2.4.1 Programming Explanation

For the development of this feature, a package unavailable in CRAN called ShinyGlobe was used. This is still under development, but the fundamental idea is similar to Leaflet, i.e. having a “background” 3D globe, whose views might be changeable in the future, onto which layers of data can be plotted.

3.3 Analysis

In this section of the app more in detail statistics can be found or created by the user. This is where Shiny shows its real power, by letting the user create ad-hoc graphical analysis thanks to its reactivity capabilities.

3.3.1 Plot Creation



Plot Creation

Logarithmic Scale:

☐ Log y-axis ☐ Log x-axis

Reference Lines

☐ Identity Line
☐ Horizontal Zero Line
☐ Vertical Line
☐ Horizontal Line

Loess: ○ Loess [✓ ● None]

Loess Span: 0.75

Loess Color: black

Mean

This might trigger the debugger in RStudio; if so, stop it from within RStudio console

☑ Mean ☐ Links

Cluster By: Operator

Show 5 entries Search:

	Operator	FlightTime_hrs	GCD_km
1	air france	8.67500	7456.256
2	asiana airlines	8.55800	7042.529
3	british airways	10.27846	8873.916
4	business aviation asia ltd	10.86000	9321.570
5	cabot aviation	9.78000	8153.090

Showing 1 to 5 of 14 entries Previous 1 2 3 Next

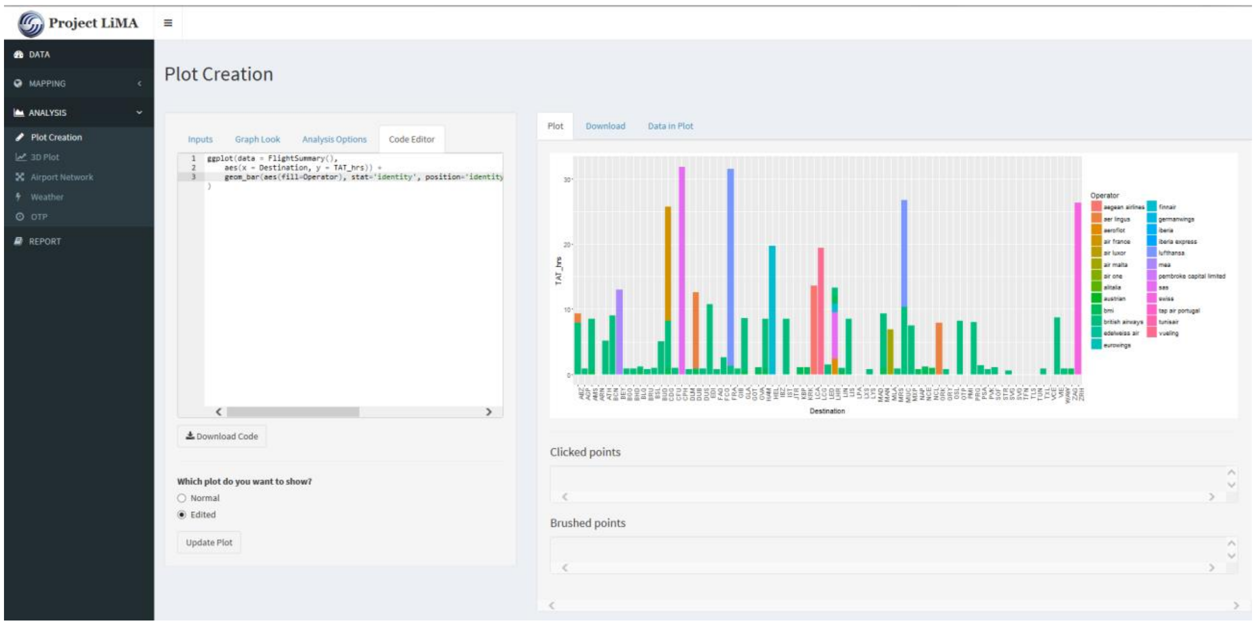
Clicked points

Brushed points

	Registration	Operator	FlightNumber	AC_type	Departure	Destination	GCD_km	FlightTime_hrs	Taxi.OUT_mins	Taxi.IN_mins	TAT_hrs	TAT_typ
273	9V-SKF	singapore airlines	SQ856	A388	SIN	HKG	2567.71	3.49	11.95	8.58		
277	9V-SKG	singapore airlines	SQ856	A388	SIN	HKG	2568.06	3.59	12.63	9.45		
46	A6-EDA	emirates	EK500	A388	DXB	BOM	1924.60	2.63	24.03	3.12	1.91	LESS
3	A6-EDD	emirates	EK406	A388	MEL	AKL	2639.73	2.93	11.95	5.38	4.82	NORMAL
458	A6-EDD	emirates	EK407	A388	AKL	MEL	2637.26	3.59	13.92	5.35	2.22	LESS

Figure 7a. Plot Creation with Flight Time vs Great-Circle Distance for A380 data on the 28-29th June 2015 with averages calculated (also note brushed points)

The first graphical feature gives the user complete freedom in the axis choice, grouping and filtering of the data. The data is also reactive and when clicking or brushing points, the data selected is shown. Moreover, different datasets can be chosen (i.e. FlightSummary or fr24data) or even imported as .csv files. Furthermore, the graph look can also be customized to create professional looking graphs for reports. In addition, analysis options can be chosen, such as regression analysis, logarithmic scales, reference lines or averages (the latter also shown in tabular form). Finally, a newly introduced feature is the code editor tab, which allows R-coding experts to type their own ggplot, plotting it on the right and downloading it.



```
ggplot(data = FlightSummary(),
    aes(x = Destination, y = TAT_hrs)) +
    geom_bar(aes(fill=Operator), stat='identity', position='identity
)
```

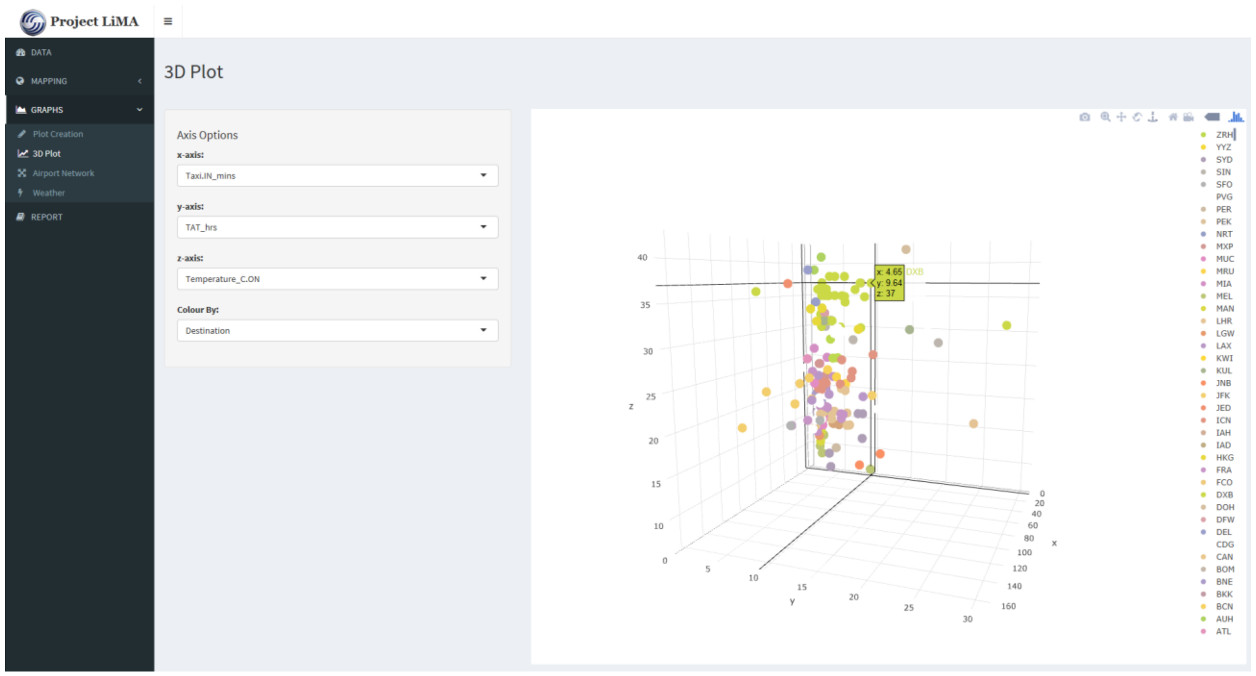
Which plot do you want to show?
Normal [✓ Edited]

Figure 7b. The code editor for A320 data at LHR, showing a custom ggplot code about TAT at the Destination airports, colored by Operator.

3.3.1.1 Programming Explanation

This feature makes one of the most powerful R libraries, "ggplot2", reactive through Shiny. Ggplot has a very free and intuitive creation process, namely the fact that options can be added to a plot, stored in a variable, just by summing them. This is the key to making it reactive through the input options that a user can choose from the UI. As follows, the codes to create a ggplot are very much alike the ones to create the already discussed SQL query, where bit by bit the plot is shaped. Moreover, Shiny capabilities make the plot download very easy. It has to be noted that the whole Plot Creation code is within the app.R file, without any calls to external functions.

3.3.2 3D Plot



Axis Options

x-axis: Taxi.IN_mins

y-axis: TAT_hrs

z-axis: Temperature_C.ON

Colour By: Destination

Figure 8. 3D Plot with Taxi IN-OUT on the x-y axis and Temperature on the z-axis colored by Destination Airport for A380 data on the 28-29th June 2015

The second graphical visualization is a three dimensional plot, for which the user can define the x-y-z axis and color the points by an inputted parameter. This is linked to the FlightSummary dataframe and it has been recently integrated to also display weather data. The plot is reactive and as the user hover on a point, the data related to it is displayed. This can be particularly useful when big datasets are to be displayed.

3.3.2.1 Programming Explanation

It is probably the most straight forward feature coding-wise since it simply links a normal 3D scatter plot created through a library called Plotly to Shiny reactivity. Not all the columns of FlightSummary are passed, but only a selection of them.

3.3.3 Airport Network

This feature creates a network of airport departure and destination according to the filtering options selected (if nothing is selected, the whole FlightSummary database is displayed). Each bubble is sized by the amount of flights it is referenced by and the arrows link the different routes and show the city-pairs. This visualization is particularly valuable to identify the hubs of particular fleets or operators as well as highlighting the strategy, i.e. Hub-and-Spoke vs Point-to-Point models.

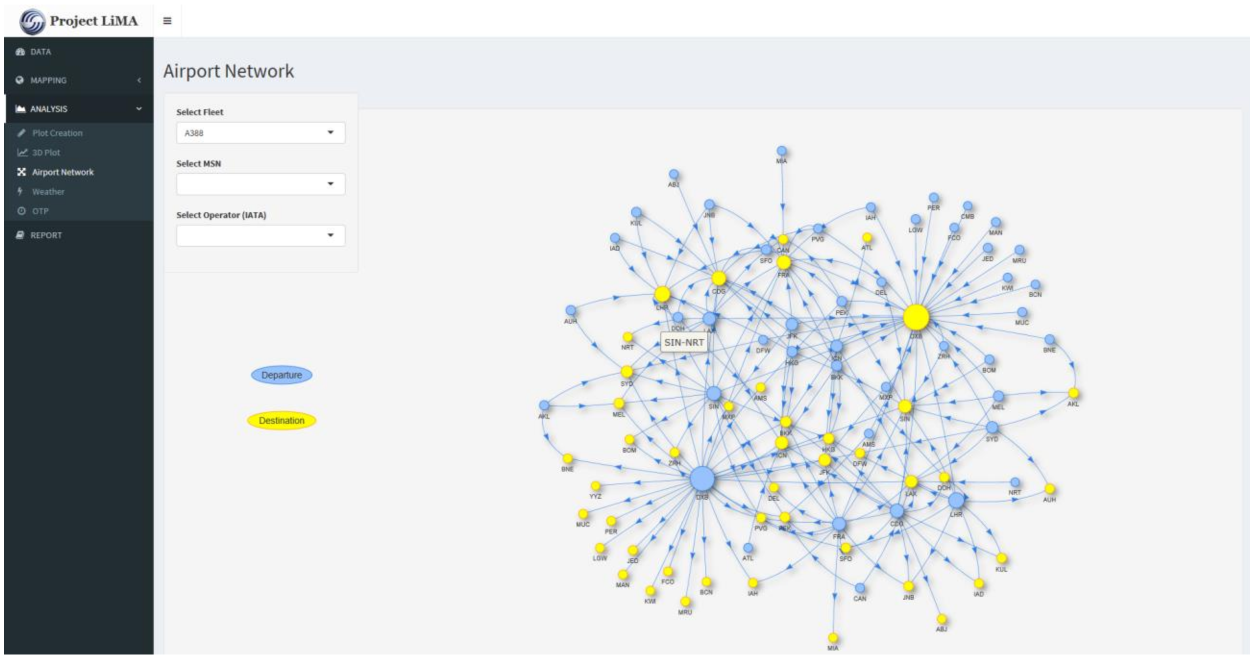


Figure 9a. Airport Networks for all A380s on the 28-29th June 2015

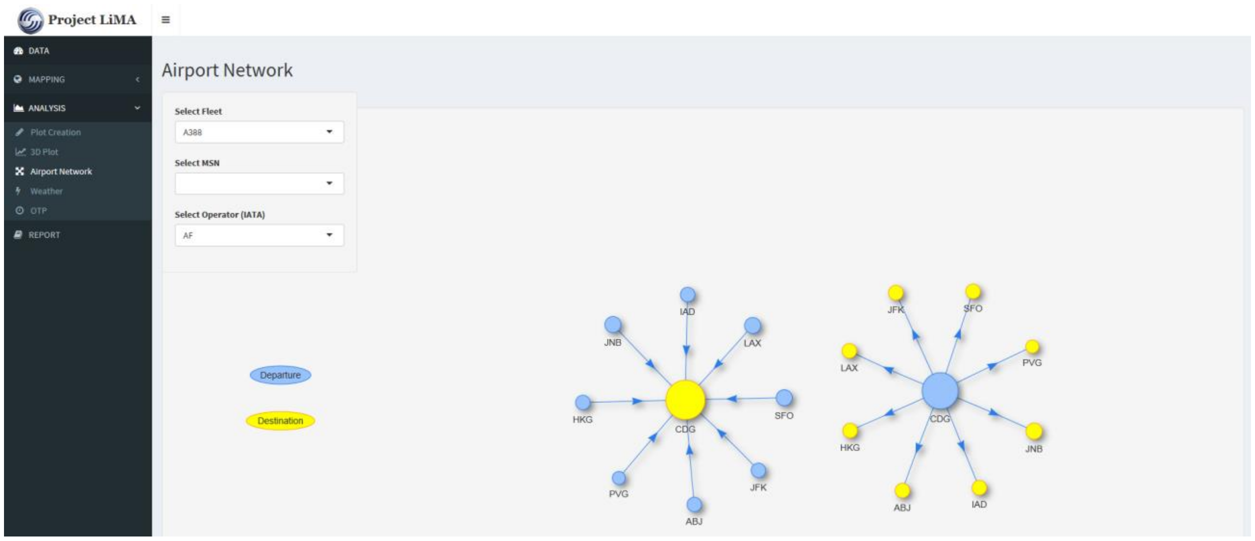
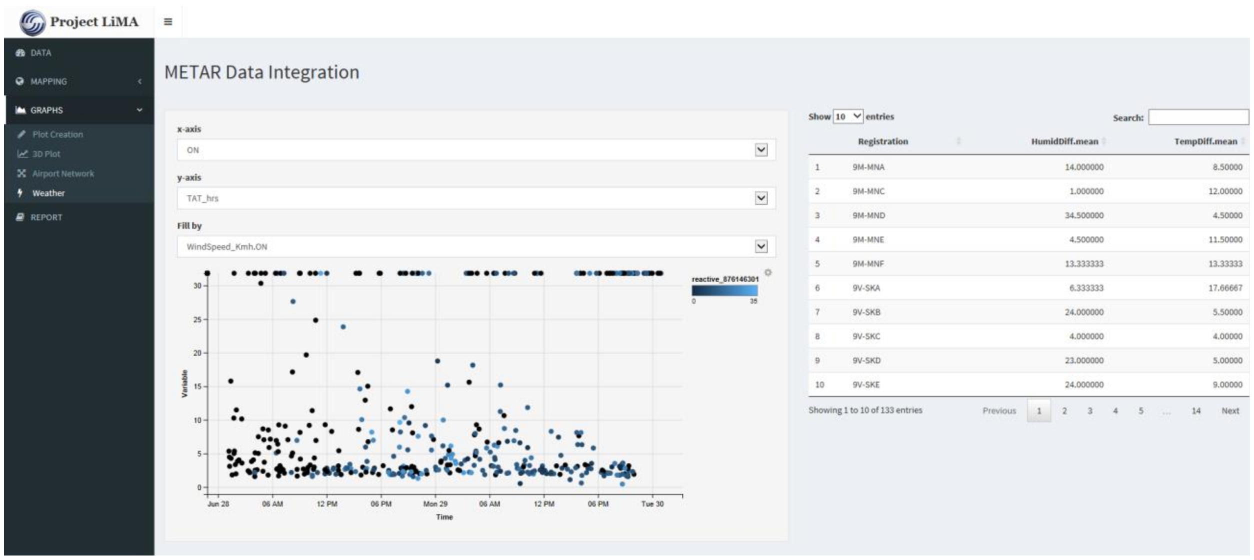


Figure 9b. Another view for Air France A380 data for the same dates

3.3.3.1 Programming Explanation

The code for this feature is again contained in the app.R without any calls to external functions, even though future developments might include the creation of a separate file. The main library used is visNetwork which translates R codes into Java Scripts, the original language for this kind of visualization. The building blocks are two: the nodes sized by a certain parameter (in our case the flight count) and the edges that link the nodes couples by unique nodes ids combinations. In our case these couples are Departures and Destinations nodes, which have to both be created separately beforehand. The data provided comes from FlightSummary dataframe, upon which filters are applied according to the user input choices.

3.3.4 Weather Integration



x-axis
ON

y-axis
TAT_hrs

Fill by
WindSpeed_Kmh.ON

Show 10 entries Search:

	Registration	HumidDiff.mean	TempDiff.mean
1	9M-MNA	14.000000	8.50000
2	9M-MNC	1.000000	12.00000
3	9M-MND	34.500000	4.50000
4	9M-MNE	4.500000	11.50000
5	9M-MNF	13.333333	13.33333
6	9V-SKA	6.333333	17.66667
7	9V-SKB	24.000000	5.50000
8	9V-SKC	4.000000	4.00000
9	9V-SKD	23.000000	5.00000
10	9V-SKE	24.000000	9.00000

Showing 1 to 10 of 133 entries Previous 1 2 3 4 5 … 14 Next

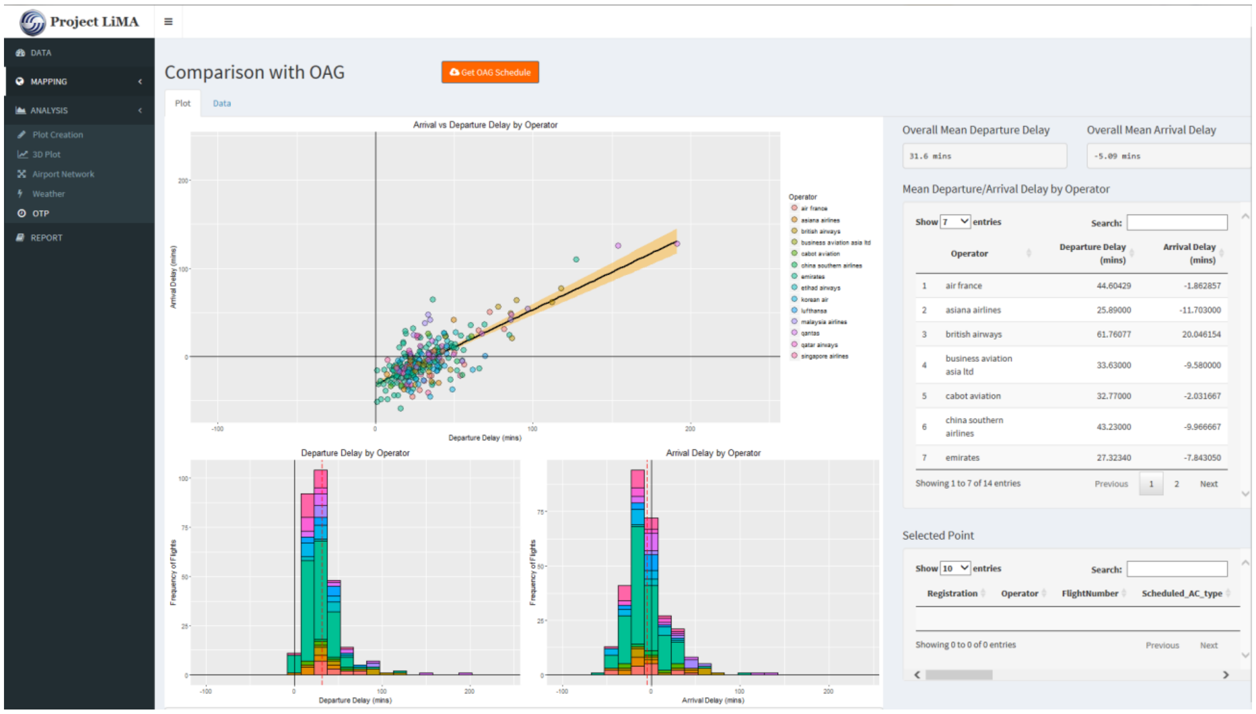
Figure 10. Weather Integration for TAT vs ON time, filled by WindSpeed at ON time for A380 data on the 28-29th June 2015

The weather dashboard displays a customizable plot on the left and a stats table on the right. The user can choose the axis and coloring options, with x always being a time variable (either OFF or ON). This technique is used to effectively analyze the influence of weather (namely windspeed, visibility, temperature, humidity etc.) onto different parameters (mainly TAT and taxi times). Moreover, the table is meant to be a monitoring tool to track the differences in what aircraft experience.

3.3.4.1 Programming Explanation

This feature exploits the integration of METAR weather data from WUnderground with FR24 data which happens in “DP3.R” function. The main library used is called “ggvis” which was developed by Google and it is meant to make ggplot more interactive. The library is still vastly under development, but it has already some useful functions (e.g. it allows the axis choice in much more straightforward way).

3.3.5 On-Time Performance



Comparison with OAG

Overall Mean Departure Delay: 31.6 mins

Overall Mean Arrival Delay: -5.09 mins

Mean Departure/Arrival Delay by Operator

Show 7 entries Search:

	Operator	Departure Delay (mins)	Arrival Delay (mins)
1	air france	44.60429	-1.862857
2	asiana airlines	25.89000	-11.703000
3	british airways	61.76077	20.046154
4	business aviation asia ltd	33.63000	-9.580000
5	cabot aviation	32.77000	-2.031667
6	china southern airlines	43.23000	-9.966667
7	emirates	27.32340	-7.843050

Showing 1 to 7 of 14 entries Previous 1 2 Next

Selected Point

Show 10 entries Search:

Registration	Operator	FlightNumber	Scheduled_AC_type

Showing 0 to 0 of 0 entries Previous Next

Figure 11. On-Time Performance for A380 data on the 28-29th June 2015

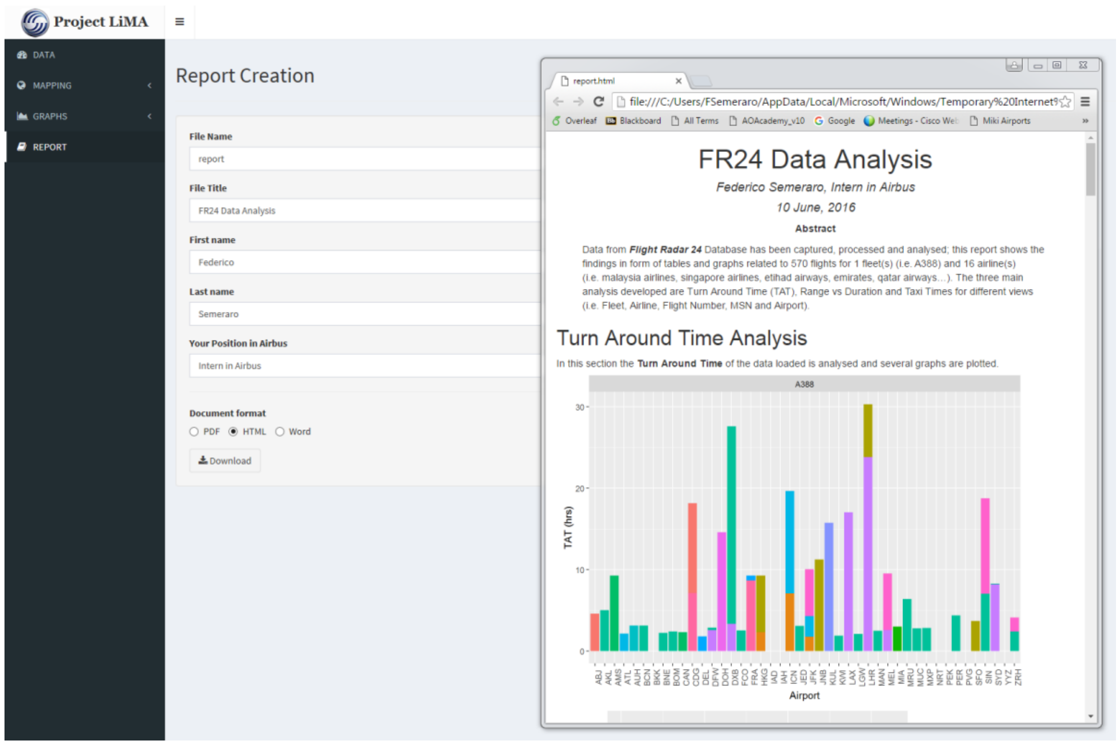
The fifth and last analysis tab is the one that links FlightSummary to OAG schedules for the On-Time Performance (OTP). By clicking on the “Get OAG Schedules” button, a function is called that imports the data through SQL and processes it by merging it with FlightSummary. The outputs are three plots, one scatter plot and two histograms, which give different views on the delays. Moreover, on the right there is a table with these statistics and below it the points hovered over in the scatter plot. Finally, the data behind these statistics and visualizations can be found and download in “Data” tab in the top left corner of the page.

3.3.5.1 Programming Explanation

This feature requires the ODBC connection as explained in Section 2. The main function behind this feature is called “oagimport.R”, which creates the SQL query in a very similar manner as for FR24 raw data import. The only parameter used to link schedules data is the flight number, since this is the only reference the schedules data comes with. The OAG summarizes the schedules per months and it puts a check on the “FREQ_DAY” column of the month that it is supposed to fly. Therefore, during the download, some extra data is imported and the “FREQ_DAY” columns are how the schedules is filtered, i.e. keeping only the days of interest. The FlightSummary dataframe is then merged with OAG schedules by flight number, departure and destination. Finally, other data-processing is developed, including the delays calculations (N.B. OAG gives the OUT and the ON, the second one is calculated by adding the OUT to the flight time – called the “LEG_FLYING_TIME”).

3.4 Automated Reports

The automation of reports is still a work-in-progress feature, more a proof of concept than a real useful application. It uses a report template containing some graphs that show the link to the data imported. The user can choose the format of the report, between PDF, HTML and Word.



Report Creation

File Name: report

File Title: FR24 Data Analysis

First name: Federico

Last name: Semeraro

Your Position in Airbus: Intern in Airbus

Document format: PDF [✓ HTML] Word

file:///C:/Users/FSemeraro/AppData/Local/Microsoft/Windows/Temporary%20Internet

FR24 Data Analysis

Federico Semeraro, Intern in Airbus

10 June, 2016

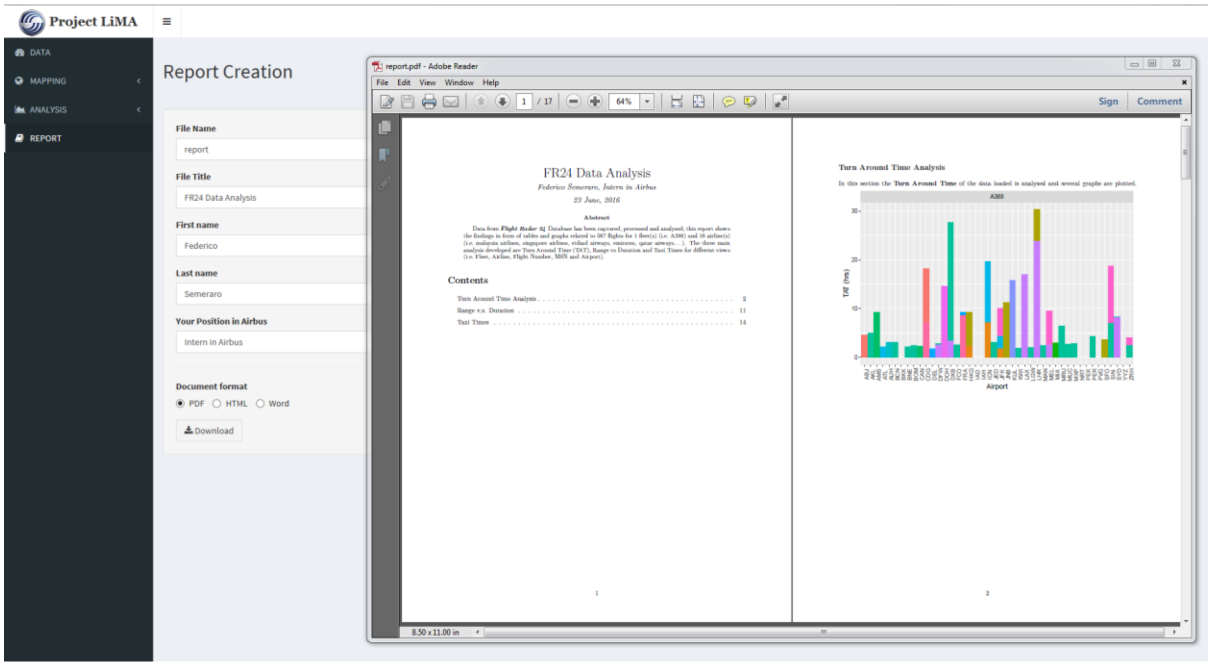
Abstract

Data from ***Flight Radar 24*** Database has been captured, processed and analysed; this report shows the findings in form of tables and graphs related to 570 flights for 1 fleet(s) (i.e. A388) and 16 airline(s) (i.e. malaysia airlines, singapore airlines, etihad airways, emirates, qatar airways…). The three main analysis developed are Turn Around Time (TAT), Range vs Duration and Taxi Times for different views (i.e. Fleet, Airline, Flight Number, MSN and Airport).

Turn Around Time Analysis

In this section the **Turn Around Time** of the data loaded is analysed and several graphs are plotted.

Figure 12b. Automated Reports in HTML



Report Creation

File Name: report

File Title: FR24 Data Analysis

First name: Federico

Last name: Semeraro

Your Position in Airbus: Intern in Airbus

Document format: [✓ PDF] HTML Word

Download

FR24 Data Analysis

Federico Semeraro, Intern in Airbus

23 June, 2016

Abstract

Data from ***Flight Radar 24*** Database has been captured, processed and analysed; this report shows the findings in form of tables and graphs related to 567 flights for 1 fleet(s) (i.e. A388) and 16 airline(s) (i.e. malaysia airlines, singapore airlines, etihad airways, emirates, qatar airways…). The three main analysis developed are Turn Around Time (TAT), Range vs Duration and Taxi Times for different views (i.e. Fleet, Airline, Flight Number, MSN and Airport).

Contents

Turn Around Time Analysis 2
Range v.s. Duration 11
Taxi Times 14

1

Turn Around Time Analysis

In this section the **Turn Around Time** of the data loaded is analysed and several graphs are plotted.

2

Figure 12b. Automated Reports PDF

3.4.1 Programming Explanation

The report automation is developed through the use of a library called “knitr”, which links Latex programming to R Markdown. The app takes the data imported and processed and calls a report template called “report_template.Rmd”. This has several graphs and visualizations that are linked to the data. The report can be assembled and downloaded in different formats and these options can be chosen through a Shiny “radioButton”.

4. Future Developments

There are several areas in which there is still a lot of room for improvement and development and in which some new functionalities and features could be added. The following is a non-comprehensive list of what I would personally develop next:

Improvements

Data Processing:
Through the usage of Flight Path dashboard (look at the list of registration numbers appearing below the Altitude vs Time plot), it was observed that the algorithm works for the majority of flights, but quite often it misses a flight because the ADS-B transponder is turned just *after* take-off (therefore OFF estimation fails) and just *before* landing (thefore ON estimation fails). This is particularly observed for Singapore Airlines's A380s. What is therefore required is a post-processing algorithm that looks for the missing city-pair in the raw data, also exploiting the fact that we roughly know the rows number where it should be. However, in order for this to be quick, loops should be avoided. Moreover, something that would be useful to several people is the identification of other events during the flight cycle, such as start of cruise, rate of descent etc. These are left as the main challenge to the next developer.

Routes Dashboard:
The "fr24datainterp.R" function relies on a for-loop for the creation of the timeframe dataframe. This significantly impacts the speed of the data processing, requiring an improvement in which loops should be avoided.

Weather Data Access:
The weather feature relies on weather file downloaded via an Excel Macro. However, this has only been done for the 28-29 June 2015 and not for all the airports. An improvement would look at automating this process within the app through libraries such as "weatherData" which link WUnderground to R.

Automated Reports:
The idea behind it would be to easily transfer the graphs created in the Analysis dashboards to a report within the app or have an easy link with Word.

New Features

Statistical Tables:
It would be very convenient creating a way to filter, subset or cluster data in a tabular fashion, which could quickly answer requests without requiring ad-hoc algorithms.

Moreover, thanks to the introduction of smartphones, I would start looking at how to make LiMA run on an Android system.

5. Conclusion

R-programming turned out to be the right choice to deal with FR24 raw data and a Shiny app was exactly what was needed to make the visualization accessible to a wider range of people. Its agility and simple implementation are the key ingredients for such a result. If Airbus was willing to create an internal community similar to Github (as there is discussion about), I would strongly recommend R-Shiny as the core language. In that case, I would envision a community that easily shares pieces of code, improving each other's work continuously and more openly. This would deeply revolutionize the definition itself of working in this industry. LiMA would become only the first of a list of useful app that would spring from this process.

Industrial giants are today in front of an unstoppable digitalization requirement and embracing this movement becomes a must to remain competitive. There are clear indications that Airbus is moving in the right direction: the adaption to smartphones as business phones, the interest in Big Data activities linked to an aircraft telemetry that is arising etc. I personally think that a real strong structure behind all this is still to be developed and a good start would be allowing people creating their own app that can be installed from and internal App Store. This would increase the awareness in people as well as triggering interest and creativity.

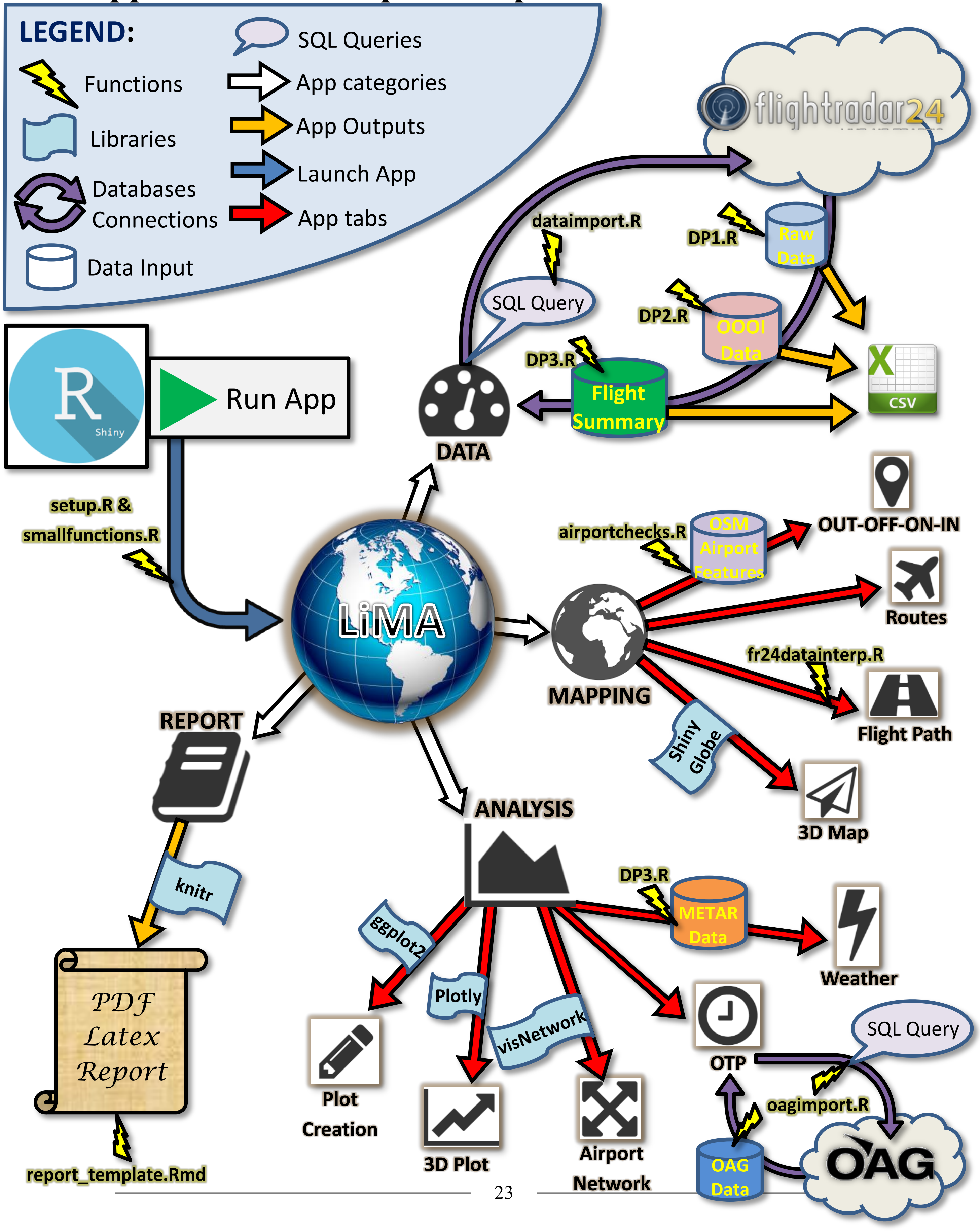
As far as LiMA is concerned, a first step would be uploading the app to an online server to which people could easily access. The app discussed in this document has been tested with only two years of raw data and, when run for big fleets, the data processing times become significant. With the two years of data that will soon be available, more computational power is required together with an optimization of the app's functions. LiMA could potentially run on a cluster of computers, making the data processing considerably faster.

Note

The work discussed in this document is the result of extensive self-research and ideas inspired by either online applications or feedback from others. I had no previous experience in data science, R or web development.

I would like to thank my manager and Airbus for the opportunity given to me.

Appendix A: Conceptual Map



Appendix B: Lessons Learnt and useful R codes

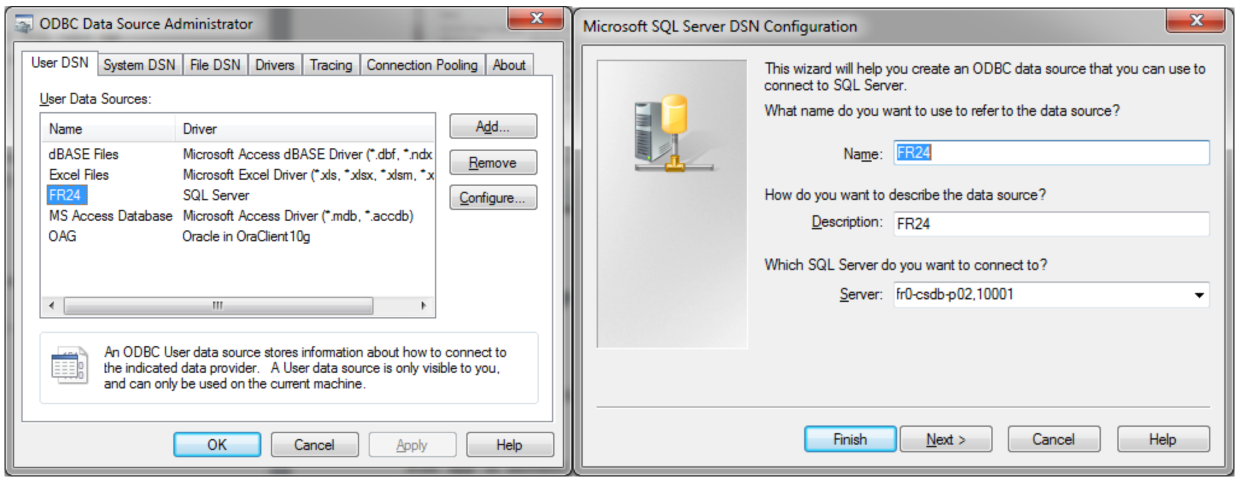
There are several recommendations that I would like to give to whoever will take up my role. These come from a year of coding experience and might seem common sense, but turned out to be very useful:

- Always strive to keep the big picture in mind
- Identify and listen to the ultimate customer
- Google has an answer for (almost) anything in programming… use it!
- Spending more time on automating a process spares you a lot of time in the future (lesson particularly learnt from Macros in Excel VBA)
- Always make a copy of your work once you reach a milestone (just look at LiMA development at C:\ Users\ FSemeraro\ Desktop\ R - Functions copies)

Useful Links where to get inspired or get help

- http:/ / www.showmeshiny.com/ which often has links to code on Github (on which you can download the whole zipped folder of an app)
- http:/ / shiny.rstudio.com/ which has both a gallery of apps and widgets or a very easy but detailed tutorial guide
- https:/ / plot.ly/
- http:/ / stackoverflow.com/

Screenshots Connection procedures (Step 2 of Section 2.)



User Data Sources:

Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.ndx
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.x
FR24	SQL Server
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)
OAG	Oracle in OraClient10g

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: FR24

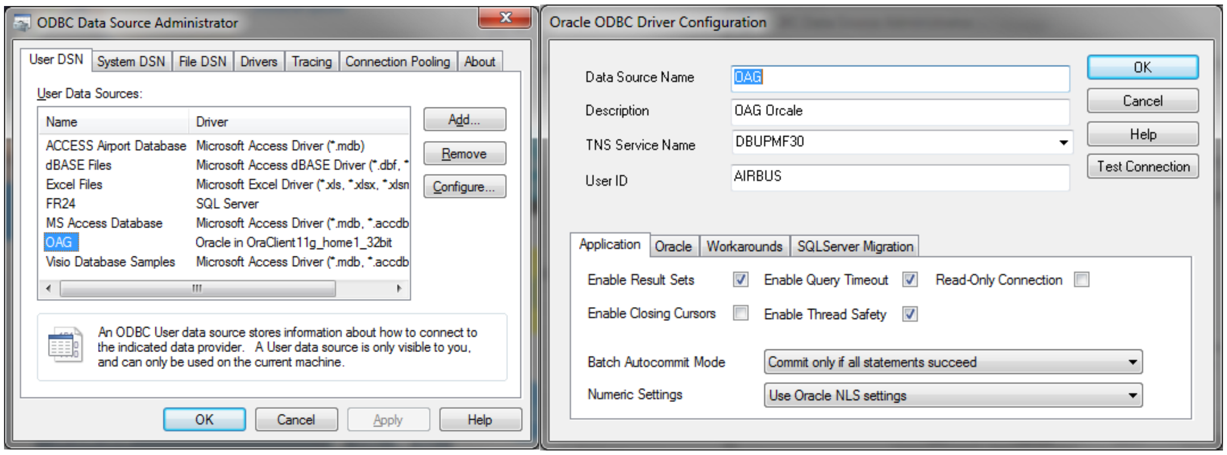
How do you want to describe the data source?

Description: FR24

Which SQL Server do you want to connect to?

Server: fr0-csdb-p02,10001

Figure 13. FR24 Connection setup



User Data Sources:

Name	Driver
ACCESS Airport Database	Microsoft Access Driver (*.mdb)
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsn
FR24	SQL Server
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb
OAG	Oracle in OraClient11g_home1_32bit
Visio Database Samples	Microsoft Access Driver (*.mdb, *.accdb

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Data Source Name	OAG
Description	OAG Orcale
TNS Service Name	DBUPMF30
User ID	AIRBUS

Enable Result Sets ✓ Enable Query Timeout ✓ Read-Only Connection ☐
Enable Closing Cursors ☐ Enable Thread Safety ✓

Batch Autocommit Mode: Commit only if all statements succeed
Numeric Settings: Use Oracle NLS settings

Figure 14. OAG Connection setup

Standalone App

In order to create a standalone app the following tutorial was followed:
http://blog.skyscorer.com/2014/03/17/packaging-your-shiny-app-as-windows/

Codes:

Clear working Environment and Console:

```
closeAllConnections()
rm(list=ls())
cat("\014")
```

Run app in another browser:

```
browser.path =
file.path(getwd(),"/GoogleChromePortable/GoogleChromePortable.exe")
options(browser = browser.path)
shiny::runApp("./FR24_AnalysisTool/app.R"),launch.browser=TRUE)
```

Install packages not available in CRAN:
Download the .tar.gz file and then run the following

```
Install.packages(path_to_file, repos=NULL, type(‘source’))
```

Import Excel (irrelevant example):

```
install.packages("readxl")
library(readxl)
setwd("O:/ENGINEERING/EIO/UK_E86_R_and_T/DEGs_and_Interns/INT_FSemeraro/Big
Data Analysis Project/Availability/Analysis")
file_path <-
'O:/ENGINEERING/EIO/UK_E86_R_and_T/DEGs_and_Interns/INT_FSemeraro/ShareRNet/F
iles'
path <- "O:/ENGINEERING/EIO/UK_E86_R_and_T/DEGs_and_Interns/INT_FSemeraro/Big
Data Analysis Project/Availability/Analysis/Overall.xlsm"
data <- read_excel(path, sheet = "OF_Airport_Info")
data <- data.frame(data)
```

App template:

```
ui <- fluidPage(
)
server <- function(input, output, session) {
}
shinyApp(ui = ui, server = server)
```

Cluster Data (example):

```
aggregate(cbind(x = oagdata()$Dep_Delay_mins, y = oagdata()$Arr_Delay_mins),
FUN=mean, by=list(oagdata()$Operator))
```

Process info:
The material to assemble the info.RData is in the folder
O:\ ENGINEERING\ EIO\ UK_E86_R_and_T\ DEGs_and_Interns\ INT_FSemeraro\ ShareRNet\ FR24 - R Analysis Tool\ Data for App. The script used to process those files is the following

```
load("C:/Users/FSemeraro/Desktop/LiMA/FR24_AnalysisTool/Data/Info.RData")
## Aircraft & Helicopters
iatacodes <- read.csv("C:/Users/FSemeraro/Downloads/ACiatacodes.csv")
test <- merge(AC_FR24, iatacodes[,c("ICAO","Description")], by.x=1,
by.y="ICAO", all.x=T)
test <- merge(test, iatacodes[,c("IATA","Description")], by.x=1, by.y="IATA",
all.x=T)
names(test) <- c("AC_FR24", "FleetName", "FleetName2")
test$FleetName <- ifelse(!is.na(test$FleetName2),
as.character(test$FleetName2), as.character(test$FleetName))
test <- test[,c("AC_FR24", "FleetName")]
test$FleetName <- ifelse(is.na(test$FleetName), as.character(test$AC_FR24),
as.character(test$FleetName))
AC_FR24 <- test[!duplicated(test$AC_FR24), ]
Utilisation_FR24 <- merge(Utilisation_FR24, subset(AC_FR24,
!is.na(FleetName)), by.x="Equipement", by.y="AC_FR24", all.x=T)
rm(test, iatacodes)
```

Airport features download:
You can find the scripts used to download and process and info from OSM at
O:\ ENGINEERING\ EIO\ UK_E86_R_and_T\ DEGs_and_Interns\ INT_FSemeraro\ ShareRNet\ FR24 - R Analysis Tool\ OSM Data Download.
Note that Airbus does not allow the connection to OSM database, therefore requiring to access it externally.

Process weather:
The functions that were used to process weather data with the excel files is at:
O:\ ENGINEERING\ EIO\ UK_E86_R_and_T\ DEGs_and_Interns\ INT_FSemeraro\ ShareRNet\ FR24 - R Analysis Tool\ Weather