Universitat Autònoma de Barcelona

escola d'enginyeria

APPLICATION OF DATA SCIENCE TECHNIQUES TO THE FIELD OF AIR TRAFFIC CONTROL

Memòria del Treball Fi de Grau
Gestió Aeronàutica
realitzat per
Nur Ben-nasr Vilches
i dirigit per
Liana Napalkova
Sabadell, 10 de febrer de 2016

FULL DE RESUM – TREBALL FI DE GRAU DE L'ESCOLA D'ENGINYERIA

Títol del Treball Fi de Grau (obligatori en tres idiomes: Català, Castellà, Anglès)

Aplicación de la ciencia de dato al ámbito del control del tráfico aéreo

Aplicació de la ciència de dades en l'àmbit del control del tràfic aeri

Application of data science techniques to the field of air traffic control

Autor[a]: Nur Ben-nasr Vilches | **Data:** Febrer 2016

Tutor[a]/s[es]: Liana Napalkova

Titulació: Gestió Aeronàutica

Paraules clau

- **Català:** Rutes lliures en l'espai aeri, ciència de dades, gestió aeronàutica, Python.
- **Castellà:** Rutas libres en el espacio aéreo, ciencia de datos, gestión aeronáutica, Python.
- **Anglès:** Free routing airspace, data science, aeronautical management, Python.

Resum del Treball Fi de Grau

- **Català:**

El present treball de final de grau està dedicat al desenvolupament d'una interfície capaç de donar suport al controlador del tràfic aeri en els processos de decisió, tenint en compte la viabilitat de certs sectors del tràfic aeri per seguir rutes lliures, mitjançant l'anàlisis de dades. Permetent a un cert vol, la possibilitat de triar lliurement la ruta entre un determinat punt d'entrada i un determinat punt de sortida. Proporcionant a aquest vol, la possibilitat d'estalviar combustible, temps, i costos d'operació

- **Castellà:**

El presente trabajo de final de grado está dedicado al desarrollo de una interfaz capaz de dar apoyo al controlador del tráfico aéreo en los procesos de decisión, teniendo en cuenta la viabilidad de ciertos sectores del tráfico aéreo para seguir rutas libres, a través del análisis de datos. Permitiendo e un cierto vuelo, la posibilidad de elegir libremente la ruta entre un determinado punto de entrada y un determinado punto de salida. Proporcionando a este vuelo, la posibilidad de ahorrar combustible, tiempo, y costes de operación.

- **Anglès:**

The following final degree study is dedicated to the development of an interface able to support Air Traffic Controllers in the process of decision making regarding the availability of certain sectors for free routing, by using data analysis and visualization techniques. Allowing a certain flight, the liberty to free routing between a given entry point and a given ending point. Providing this plane the possibility to save fuel, time, and operations costs.

Index

List of figures

Figure 26: weather conditions.

Figure: 27: Polar projection indicating the wind direction.

List of tables

Introduction

State of the art

The following study is dedicated to the development of an interface using data analysis and visualization techniques that originate from the data science field (*a set of fundamental principles that support and guide the principled extraction of information and knowledge from data* ***[1]***), focused on the extraction of useless information from a given ATM data, and targeting SESAR (*Single European Sky ATM Research*) concepts.
This interface will give support to the air traffic controller in the process of decision making, specifically in the concept of free routing airspace (users preferred routing, whenever possible).

Nowadays the aviation sector generates a large amount of useful data from different sources and of a diverse nature: safety data and reports, flight plans, navigation data, forecast, airport data, radar tracks, etc. That is almost wasted. But, fortunately, the aviation sector is changing this concept. And in these days, many business functions have entered into a new era of decision making as a result of increasing abilities to store, access and analyze data. For example, many airlines are now focusing on managing information streaming from their different operations and developing new analysis techniques to get/extract information from them. And even an organization as EUROCONTROL () is changing its point of view of the data science concept, and it's sponsoring workshops **[2]** to help the developers in this area, knowing that this science can be really helpful for the future ATM, and to develop the SESAR concepts/technology.

So, the main idea of this study is to mix two concepts, such as data science, and free routing.
Free routing consists in allowing a user freely plan a route between a defined entry point and a defined exit point, but before this sector must have be "enabled" for free routing. According to many researchers the "keys" that ATM should take into account, to define a sector as able to free routing, are the air traffic flows, and the weather conditions (wind, wind disgust, turbulence, etc.). What this interface will do, Is to help the controller to define this sectors as valid or not, being connected to an ATM data provider, that will provide the air traffic flows by giving the exact position of the airplanes(latitude, longitude, altitude, plus time) , and the weather conditions(wind speed, turbulence, etc), in real time(tactical phase).

Objective and Tasks

The main objective of this study is to propose the methodology and technologies to support Air Traffic Controllers in the process of decision making regarding the availability of certain sectors for free routing, by using data science techniques. Allowing a certain flight, the liberty to free routing between a given entry point and a given ending point. Providing this plane the possibility to save fuel, time, and operations costs.

To achieve this objective, the following tasks have been formulated:

1. To analyze the current and future ATM situation, and how data science can be useful.
2. To get the necessary flight data to carry out this research.
3. To create the interface using Phyton.
4. To create a sector scenario(latitude, longitude), and carry on simulations , to see how the interface works.
5. To make conclusions regarding the results obtained.

Motivations

Firstly, one of the main motivations of this research is to contribute to SESAR project concepts, due to the fact that, according to forecasts, in Europe, the air traffic flow is going to increase by a factor of x2 or x3 by 2030, therefore I think that it is clearly necessary to find new ways of increasing air space capacity while keeping the safety and a sustainable environment.

Secondly when I first introduce to my coach, the idea of develop a research targeting SESAR concepts, she thought that it would be a good idea to carry out this research by using the data science. At this moment I started to read some articles about it, and I discovered that the aviation sector is increasingly using data science. The process where the data science is more useful is in the decision making. For this reason I chose the free routing concept to carry out this research, because it's a process where the decision making is essential to make the air traffic more efficient and safety.

Practical value

On the one hand, this study wants to contribute "prove" that the data science can contribute to develop new technologies, case studies, concepts, ect., that might be extremely useful, in the aviation sector and many other sectors as well.

On the other hand, the given work offers to the scientific ATM community, in order to facilitate the fast prototyping of new ATM concepts and the test of new Decision Support Tools that may contribute to a better understanding of the current and future ATM system.

It also wants to add value to the concepts that the SESAR program is introducing, as free-routing, 4D trajectories, etc., as well as to prove that the old fashion concepts that the current ATM system is using, are limiting the ATM capacity.

Methodology

The methodology consists in selecting appropriate ATM data collection, processing and visualization techniques, as well as a suitable development environment.

For the ATM data collection, FlightStats data provider will be used during this research in order to extract real-world information about air traffic, and weather conditions

For the purpose of data processing, statistical and analytical libraries from python, such as *scikit-learn*, *numpy* and *pandas* will be employed.

The work will use matplotlib and mpl_toolkits libraries in order to visualize output results of data queries.

Finally, JetBrains PyCharm (Python 2.7) has been selected as a development environment in order to support necessary functionality of the interface.

Thesis structure

1-SESAR, Free-routing, 4DT: During the first chapter the situation of the current and the future ATM will be exposed, explaining the SESAR project its concepts, its projects, also explaining the concept of Free-routing, and what data science can provide to the SESAR concepts.

2- Analysis of data science: During the second chapter the concept of data science will be explained, its history, what is it for, the techniques used, etc.

3-Case study-"developing the interface": During this chapter, first an overview of the study will be exposed, then all the steps before start programming: The scenario definition, the data provider, the development environment tool, the data collection. Finally the programming process of the interface will explained step by step.

4-Examples of use: The use of the interface will be explained, by showing some examples of its use.

Conclusions: Finally the conclusions will be exposed, in them will be discussed if this study has reached the initial objectives, it also will discuss if the interface works properly, its practical value, and the features to improve.

1.SESAR, Free-routing, 4DT.

1.1-Situation

Nowadays, and since 2005 the Air Traffic Management (ATM) model is evolving to a new way of control, known as the Single European Sky. The current Air Traffic Management (ATM) model is based on a largely fragmented and sectored management of the air traffic flows, and this model doesn't match the airspace requirements that come along with the new unifying process of the sky management(Airport capacity limit is reached, and the airspace is restricted by the controllers workload).

Because of that, the European Commission gave EUROCONTROL the task of designing the future European ATM, with the purpose of adapting it to the new needs and opportunities of the Single European Sky**.**

To achieve this purpose EUROCONTROL created the SESAR (Single European Sky ATM Research) program. The aim of this program is to define the technological and procedural requirements needed, in order to build the project of the Single European Sky, and also control and coordinate the subsequent development and deployment activities.

Implementing the SESAR program, EUROCONTROL expects to get the following benefits, beyond 2020:

- ✓ Capacity → increase the traffic flows by 3x.
- ✓ Safety → increase by a factor of 10.
- ✓ Costs → 50% reduction.
- ✓ Environment impact → 10% reduction.

And as a result of those targets, the benefits will be:

- ✓ Capacity → € 200-500 Million gains.
- ✓ Safety → Aviation even safer.
- ✓ Cost reduction 50% → € 3.4 Billion gains.
- ✓ Pollution -10% → 2 million tones of CO2 less.

Those are the SESAR targets/gains, but, the question is, how SESAR program will achieve them? Actually the SESAR program will introduce some new concepts to get those aims, such as: SWIM (System Wide Information Management), 4D Trajectory operations, free routing, and so on **[3]**.

1.2-Concepts

SWIM: A net-centric ATM intranet that will allow sharing information from and among all the stakeholders, like it enables a Collaborative Decision Making (CDM) philosophy.

It will cover all ATM information, including aeronautical, flight, aerodrome, meteorological, air traffic flow, and surveillance.

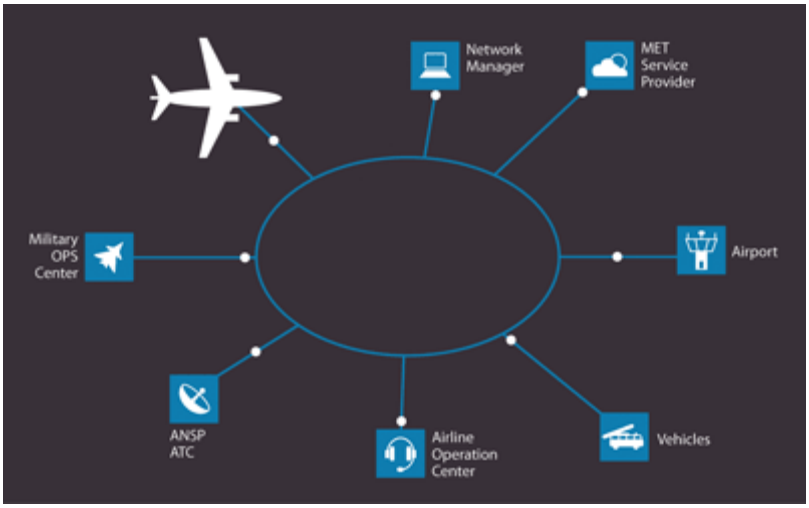


Figure 1: Swim Stakeholders.

Who share this information?

- **Pilots** – taking off, navigating and landing the aircraft
- **Airport Operations Centres** –managing departures, surface movements, gates and arrivals
- **Airline Operations Centres** – building schedules, planning flight routings and fuel uplift, ensuring passenger connections and minimizing the impact of delays
- **Air Navigation Service Providers (ANSPs)** –organizing and managing the airspace over a country and with Air Traffic Services – managing air traffic passing through their airspace
- **Meteorology Service Providers** – providing weather reports and forecasts
- **Military Operations Centers** – planning missions, blocking airspace to conduct training operations, fulfilling national security tasks

It will cover all ATM information, including:

- **Aeronautical -** Information resulting from the assembly, analysis and formatting of aeronautical data
- **Flight trajectory** – the detailed route of the aircraft defined in four dimensions (4D), so that the position of the aircraft is also defined with respect to the time component.
- **Aerodrome operations** – the status of different aspects of the airport, including approaches, runways, taxiways, gate and aircraft turn-around information.
- **Meteorological** – information on the past, current and future state of earth's atmosphere relevant for air traffic'.
- **Air traffic flow** – the network management information necessary to understand the overall air traffic and air traffic services situation.
- **Surveillance** – positioning information from radar, satellite navigation systems, aircraft datalinks, etc. **[4]**.

Figure 2: Swim definition**.**

4D TRAJECTORY: The purpose of the 4d trajectories is to control the air traffic flow under these four dimensions; latitude, longitude, altitude plus time, allowing more direct flights with benefits for passenger, airlines, and the environment.

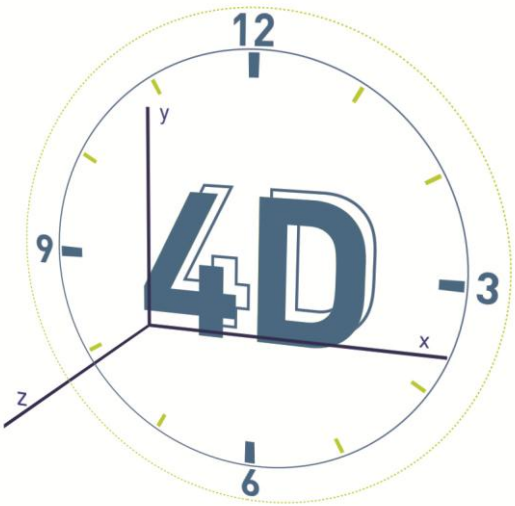


Figure 3: 4D trajectory.

FREE-ROUTING: Free route airspace is a specific airspace within which users shall freely plan their routes, the Airspace Users must be able to plan their trajectories freely between a defined entry point and a defined exit point of the free route airspace, with the possibility of deviating via intermediate navigation points without reference to the fixed route structure, which implies important fuel savings and pollution **[5]**.

Figure 4: Current ATM.

Figure 5: Future ATM.

Volume of air traffic is growing steadily every year, which implies the need of continuous improvement of European ATM. So it is necessary to increase capacity and efficiency of airspace, but with regard to environment and safety, which should be maintained or improved. The main way to reduce the environmental impact is to reduce aircraft fuel consumption. Achieving free routes, if the aircraft flew directly between two points, it would save an appreciable amount of miles and so tons of fuel, reducing CO2 emissions.

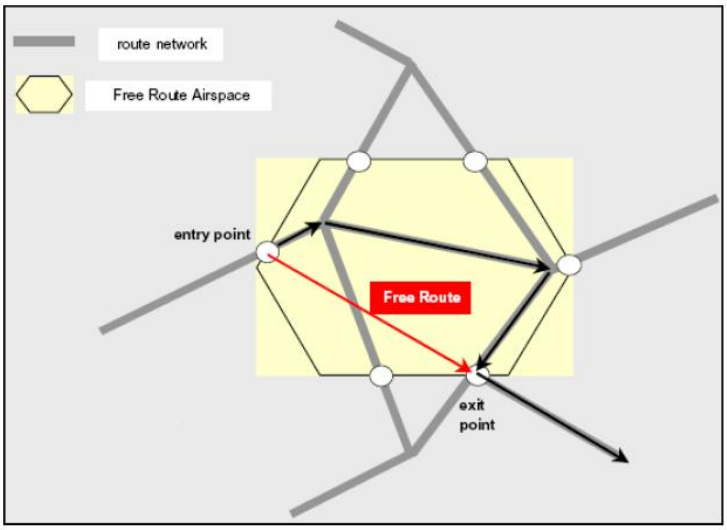


Figure 6: Free Route.

1.3-SESAR PROJECTS

Several projects received EU funds to contribute to the development of above-mentioned SESAR concepts. Among such projects, the following ones should be especially highlighted:

STREAM PROJECT: The main goal of this project has been investigating innovative computationally-efficient Conflict Detection and Resolution (CD&R) algorithms for strategic de-confliction of thousands of trajectories within few seconds or minutes taking into consideration the Aus (airspace users) preferences, and the network constraints**[6]**.

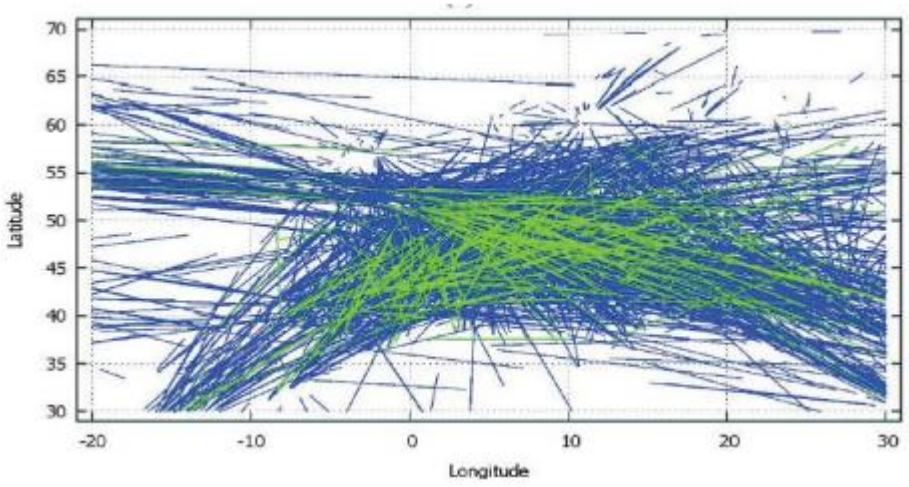


Figure 7: Conflict detection and resolution.

TESA: The aim of TESA (Trajectory prediction and conflict resolution for Enroute-to-en-route Seamless Air Traffic management) was to develop reliable Trajectory Prediction (TP) and Conflict Detection and Resolution (CDR) capabilities with the specific objectives to address the sources of error (uncertainty) in TP and to use the improved TP to optimize CDR (thereby enhancing safety)**[7]**.

1.4- Data science, how can it be applied to free routing?

The aviation sector gathers and stores a large amount of unstructured, heterogeneous data from different sources and of a diverse nature: safety data and reports, flight plans, navigation data, airport data, radar tracks, etc. From airlines to ANSPs or airports, the ability to collect information through different data sensors is growing exponentially. Nevertheless, how the different stakeholders take advantage of these data has not evolved so rapidly and there is still a large gap for improvement.

As an example, in the context of enhanced decision making and risk assessment, data science has great potential of applicability since quantitative risk-assessment techniques are not very common in aviation. When it does exist, there is usually not enough data to support it, or at least the optimal tools from data science are not being used. While a variety of data sources might be available for risk assessment in strategic (About one year before the flight takes place until one week before real time operations), pre-tactical(Six days before real time operations) and tactical(The day of operations) time scopes, the right principles need to be developed to ensure the information extraction is achievable within the given time frame. Additionally, a data-driven culture should formalize the decision-making process(free-routing), setting up standard procedures so decision makers can obtain and correctly use the most appropriate data for each situation. The process and human capabilities need to keep pace with the information-technology techniques.

The application of data science principles to the aviation sector can open the gate to significant improvements in many other key aspects of aviation such as safety enhancement, flight efficiency, environmental-impact mitigation or delay reduction **[8]**.

2-Analysis of data science.

2.1-What is data science?

Data science is the study of where information comes from, what it represents and how it can be turned into a valuable resource in the creation of technologies, business, and so on. Data science employs techniques and theories drawn from many fields within the broad areas of mathematics, statistics, chemometrics, information science, and computer science.

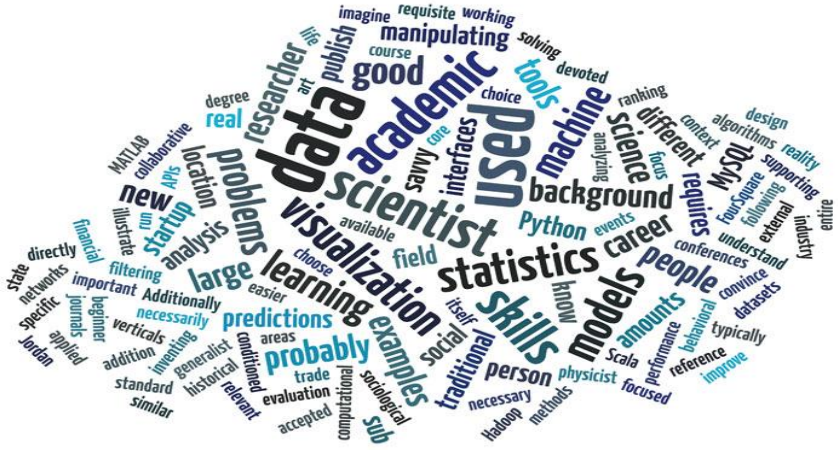


Figure 8: Data science concepts.

Including signal processing, probability model, machine learning, statistical learning, data mining, data base, data engineering, pattern recognition and learning, visualization, predictive analytics, uncertainty modeling, data warehousing, data compression, computer programming, artificial intelligence, and high performance computing **[9]**.

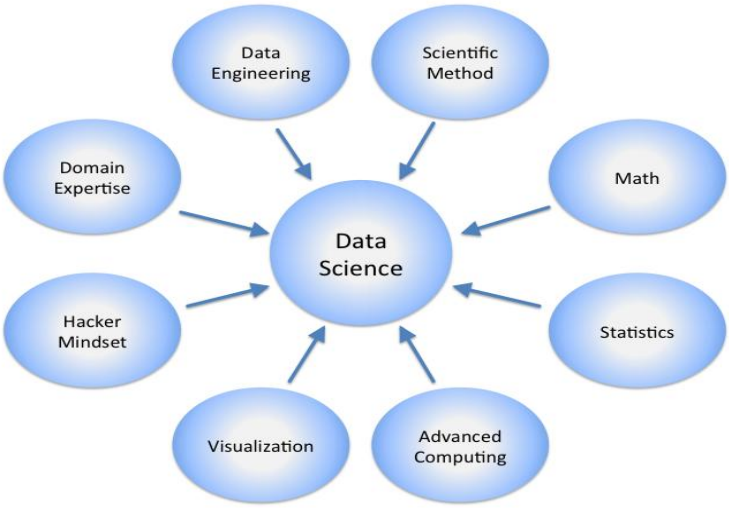


Figure 9: Data science techniques.

- **Domain expertise:** Is the knowledge of the subject, for example in this case the domain expertise would be the knowledge of the SESAR project, of the ATM concepts.

- **Data Engineering:** Is a multi-disciplinary practice of engineering computing systems, and computer software, used to extract information through the analysis of data. It is also a set of tasks able to planning, analyzing, designing, and implementing applications **[10]**.

- **Visualization:** The term visualization means, these tools that made the output data visible, visualization tools are apps, libraries able to show the data in diferents graphs, maps, polar projections, etc.

- **Statistics:** Is a mathematical study that collect, analyze, interprets, presents, and organize the data **[11]**.

- **Advanced Computing:** Is the process that allows the users to create an executable computer program from a given computing problem. Programming involves activities such as analysis, developing understanding, generating algorithms, verification of requirements of algorithms, including their correctness, and implementation of algorithms in a target programming language**[12]**.

- **Scientific Methods:** This involves a variety of advanced algorithms that are used for solving particular data science problem (i.e. prediction, classification, clustering, etc.). In particular, one can outline the group of machine learning techniques that are widely applied in data science projects. This is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. In other words, constist into create algorithms ables to generalize behaviors from a not structured data suministred as examples **[13]**.

- **Hacker Mindset :** Consist in different ways of extract data from internet, for example web scrapping is a technique used for the hackers to extract data from internet.

2.2-Data science history

Data science is for many of the data scientist, a newly established concept, but in fact the truth is that this concept has been used from almost fifty years, since the Danish scientist Peter Naur who firstly introduced it around the sixties. In 1974 Naur published *Concise Survey of Computer Methods* in which freely talked about the term data science and about the contemporary data processing methods that are used in a wide range of applications.

In 1996, members of the International Federation of Classification Societies (IFCS) met in Kobe for their biennial conference. Here, for the first time, the term data science is included in the title of the conference ("Data Science, classification, and related methods").

In November 1997, C.F. Jeff Wu gave the inaugural lecture entitled "Statistics = Data Science?" for his appointment to the H. C. Carver Professorship at the University of Michigan. In this lecture, he characterized statistical work as a trilogy of data collection, data modeling and analysis, and decision making. In his conclusion, he initiated the modern, non-computer science, usage of the term "data science" and advocated that statistics be renamed data science and statisticians data scientists

In 2001, William S. Cleveland introduced data science as an independent discipline, extending the field of statistics to incorporate "advances in computing with data" in his article "Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics,". In his report, Cleveland establishes six technical areas which he believed to encompass the field of data science: multidisciplinary investigations, models and methods for data, computing with data, pedagogy, tool evaluation, and theory **[14]**.

In April 2002, the International Council for Science: Committee on Data for Science and Technology (CODATA)[1] started the *Data Science Journal*, a publication focused on issues such as the description of data systems, their publication on the internet, applications and legal issues. Shortly thereafter, in January 2003, Columbia University began publishing *The Journal of Data Science*, which provided a platform for all data workers to present their views and exchange ideas. The journal was largely devoted to the application of statistical methods and quantitative research. In 2005, The National Science Board published "Long-lived Digital Data Collections: Enabling Research and Education in the 21st Century" defining data scientists as "the information and computer scientists, database and software and programmers, disciplinary experts, curators and expert annotators, librarians, archivists, and others, who are crucial to the successful management of a digital data collection" whose primary activity is to "conduct creative inquiry and analysis.

In 2008, DJ Patil and Jeff Hammerbacher used the term "data scientist" to define their jobs at LinkedIn and Facebook, respectively **[15]**.

The concept of data science had been used for almost fifty years, but the truth is that almost the ninety per cent of the data used, had been “collected” over the past 2 years, so it can be said that data science it’s at the same time a new and an old science.

2.3-Data science techniques used during this study

The term data science means a wide range of techniques used to turn the information into a valuable resource. During the following lines it will be explained which of this techniques will be used in this study and how.

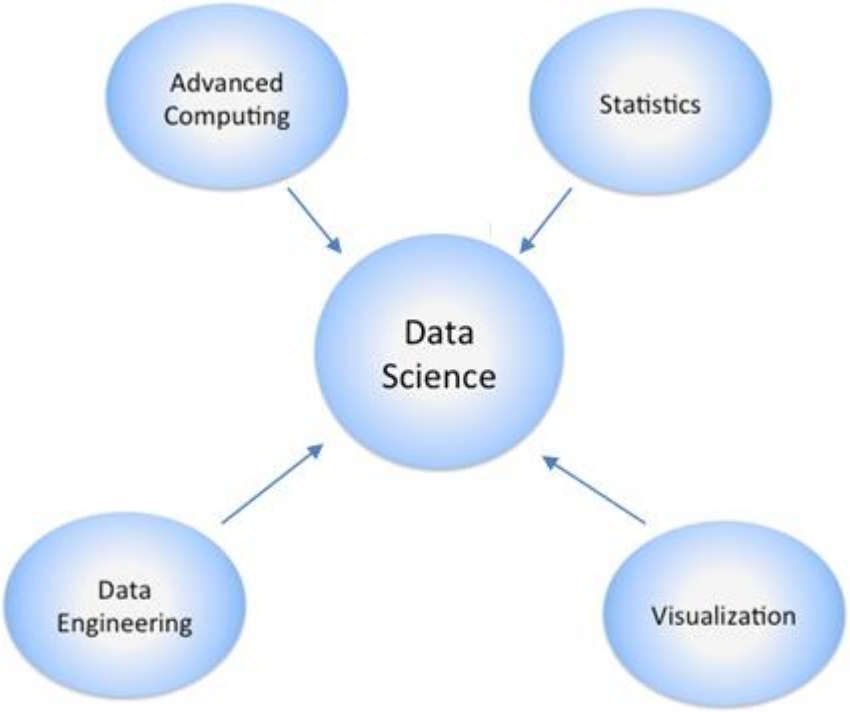


Figure 10: Data science techniques used

- **Data Engineering:** During the implementation of the interface the data engineering tasks of cleaning(take the relevant information) and merging(mix data from different origins) the data will be used.

- **Visualization:** As known the interface will be programmed using Python, due to this the visualization tools used will be libraries able to creat different graphs, tables maps, such as matplotlib(graphs), mtl_toolkits (geo-data), IPython (notebook).

- **Statistics:** Is a mathematical study that collect, analyze, interprets, presents, and organize the data. During this study it will be used to extract ratings and medians from some of the data outputs.

- **Advanced Computing:** Through this study the programming language used will be Python, using Pycharm, and Ipython as a development environment tools.

3. Case study-"developing the interface"

3.1- Overview of the study.

The study that will be carried out, consists in the creation of an interface able to help the air traffic controllers in the process of decision making, considering the availability of certain sectors for free routing, through the use of data science techniques. According to some studies supported by EUROCONTROL, to define a sector as able to free routing, the key that must be take it into account is the information, the information of the air traffic, and the information of the weather conditions(wind speed, wind direction,etc.), this is the reason why this research is part focused in the data science.

To develop this research/interface the following resource will be needed:

1. **Sector scenario:** To develop the interface it is needed the ATM data, and a programming tool, but first of all it has to be defined the location where the interface will take place, the sector where will be able to operate, in other words, the data must be located in order to make the interface work.

   By a sector scenario it is understood a follow number of coordinates(latitude, longitude), which define a sector.

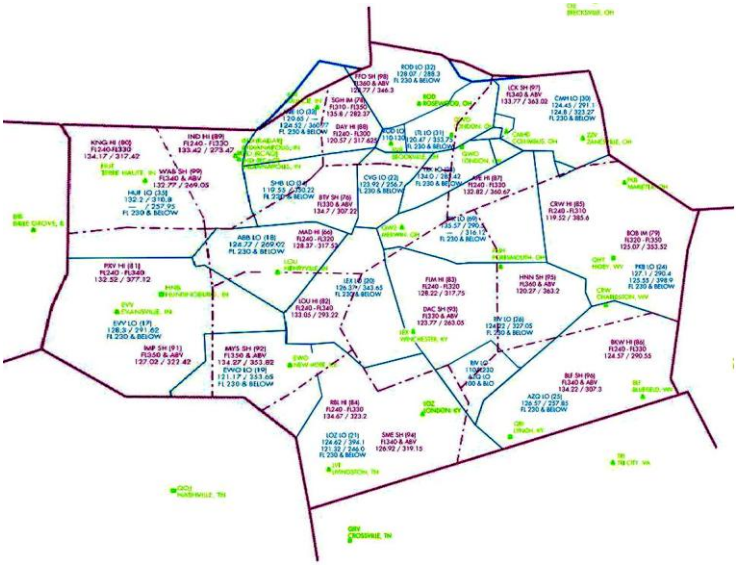


Figure 11: ATM sector

2. **Data provider:** Is who will provide the ATM data needed during the development of the interface(air traffic data, weather data).

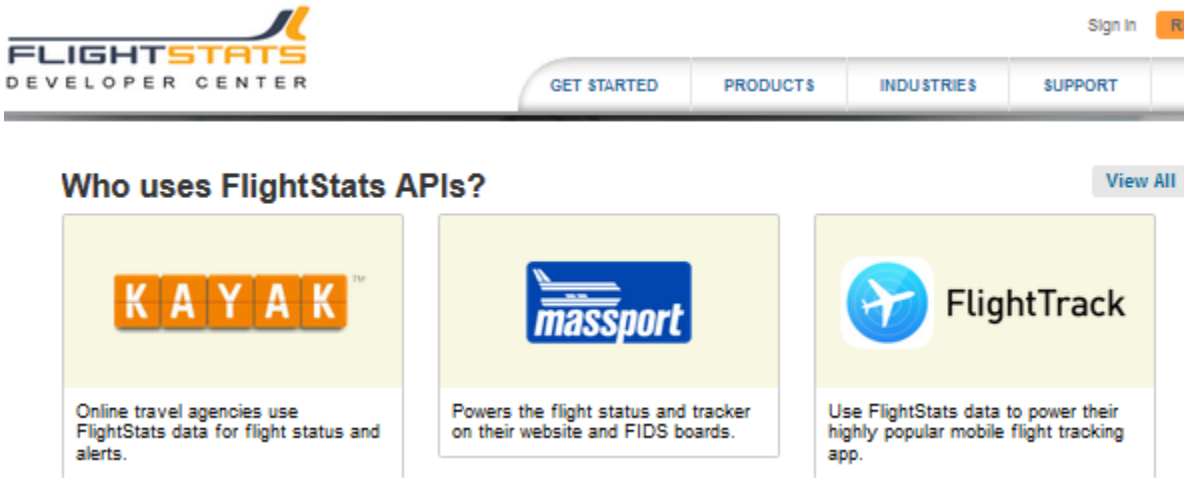


Figure 12: Flightstats Developer Center.

3. **Programing tool:** Is the development enviorment tool, in which the interface will be created.

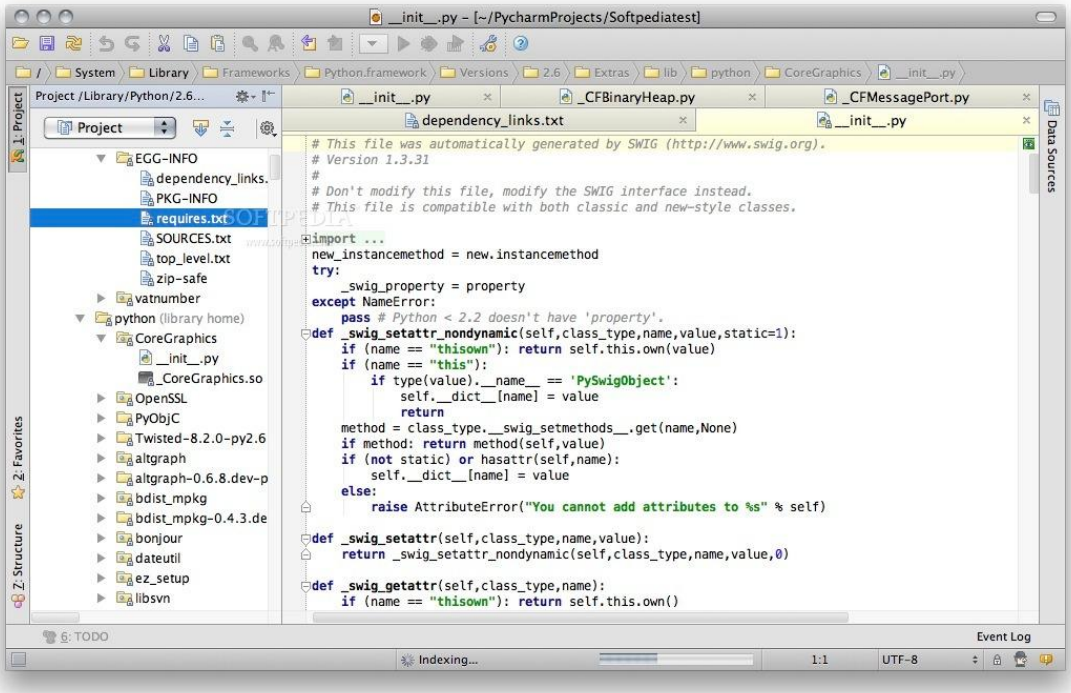


Figure 13: Pycharm environment

In the following subchapters it will be explained how this parameters have been gotten, and how have been used to develop the interface and carry out the study

3.2- Scenario Definition.

As it has been said, before start collecting the data, and programming the interface, the scenario must be defined. To define the scenario a set of coordinates somewhere in earth have been set:

- Top latitude: 46
- Left longitude: -125
- Bottom latitude: 34
- Right longitude: -118

Figure 14: Sector scenario

It has to be said that this coordinates have not been chosen for any special reason, the interface will be able to work with any coordinates.

3.3- Data Collection

3.3.1- Flightstats

Flighstats is a well known online flight data provider. Flightstats has established a leadership position as a provider of real-time global flight information, servicing airlines, airports, travel agencies, developers, consumers, and more.

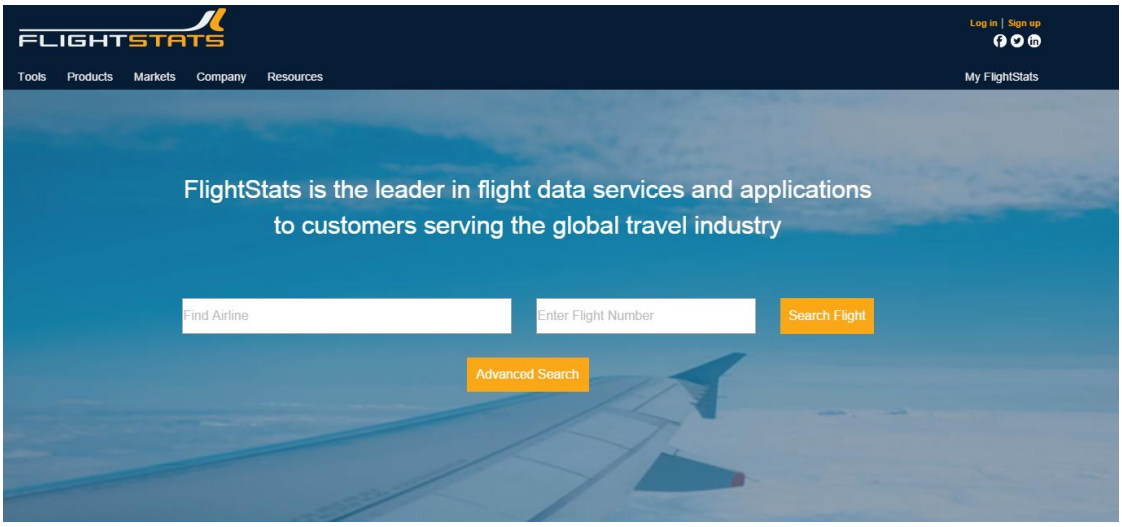


Figure 15: Flightstast web site.

Flightstats delivers global flight data and airport data that powers many of the world's most popular travel applications, it also provides web and mobile applications to the FlightStats community, helping travelers to better manage their travel day **[16]**.

Flightstats also has a developer online site, called "Flightstats Developer Center" in which the developers have access to global flights information. To access this information, the site has several API'S(Application Programming Interface) in order to provide different information, like weather information, air traffic information, delays, schedules, flight ratings, route ratings, and so on.

Figure 16: Flightstats API'S

3.3.2- Collecting the data

Once created the scenario and defined the data provider, the next step is collect the data. As it has been said, the data needed to develop this interface is:

- **Air traffic data:** The exact position of the flights that are passing around the sector scenario previously created, by giving the latitude, longitude, altitude, and time.

- **Weather data:** The weather forecast for the sector scenario created, getting, the METAR, wind speed, wind direction, wind gust, and so on.

To collect this data the online site flightstats developer center will be used. To access the data, first registering is required , and as a developer Flightstat gives monthly free subscription to download the data needed. Once gotten the access, the next step is using the API'S to collect the data required.

3.3.2.1-Collecting air traffic data

To collect the air traffic data the API ***Flight status and track*** will be used. Once inside the API, many different data options can be observed.

Flight Status & Track

Flight Status gives you access to current flight information, including scheduled, estimated and actual departure/arrival times, equipment type, delay calculations, terminal, gate and baggage carousel.
Flight Track gives you access to information on an active flight, including position (lat/long), previous positions, altitude, bearing, speed and route.
Flight Position provides recent positional information on flights in a defined area.

Entry Points	Name	Description	Response Elements
flight/status/	Flight Status by Flight	*Flight Status* for a specific flight identified by Carrier, Flight, and Date, or by unique FlightStats identifier.	request, appendix, error, flightStatus
flight/track/	Flight Track by Flight	*Flight Track* for a specific flight identified by Carrier, Flight, and Date, or by unique FlightStats identifier.	request, appendix, error, flightTrack
airport/status/	Flight Status by Airport	*Flight Status* for flights arriving at or departing from a specific airport.	request, appendix, error, flightStatus
airport/tracks/	Flight Track by Airport	*Flight Track* for flights arriving at or departing from a specific airport.	request, appendix, error, flightTrack
route/status/	Flight Status by Route	*Flight Status* for flights identified by departure airport, arrival airport, and date.	request, appendix, error, flightStatus
flightsNear/	Flights Near (an area)	*Positions* for flights currently within an area (identifed by point and radius, or by geographic boundaries).	request, appendix, error, flightPosition

Figure 17: Flight status & Track options

Flight status and track has also different API'S within, the API needed to get the flights position is ***FlightsNear.*** Once inside FlightsNear, it can be observed that has different information to provide.

Figure 18: FlightsNear options

The one to be used is ***Flights within bounding box,*** This API provide all flights with current positions inside a specified bounding box. Bounding box coordinates are specified in degrees of latitude/longitude.

PARAMETER	VALUE	DESCRIPTION
appId	(required)	Application ID
appKey	(required)	Application key
topLat	(required)	Latitude for top of bounding box
leftLon	(required)	Longitude for left side of bounding box
bottomLat	(required)	Latitude for bottom of bounding box
rightLon	(required)	Longitude for right side of bounding box

Figure 19: Parameters required by the *Flights within bounding box* API

To get the data from the API it is required to define first the access parameters that flightstats provide once registered, appId, and appKey, and then the coordinates of the bounding box. The coordinates used will be the same ones defined to create the sector scenario, allowing all the flights positions returned from this API, pass around the sector scenario.

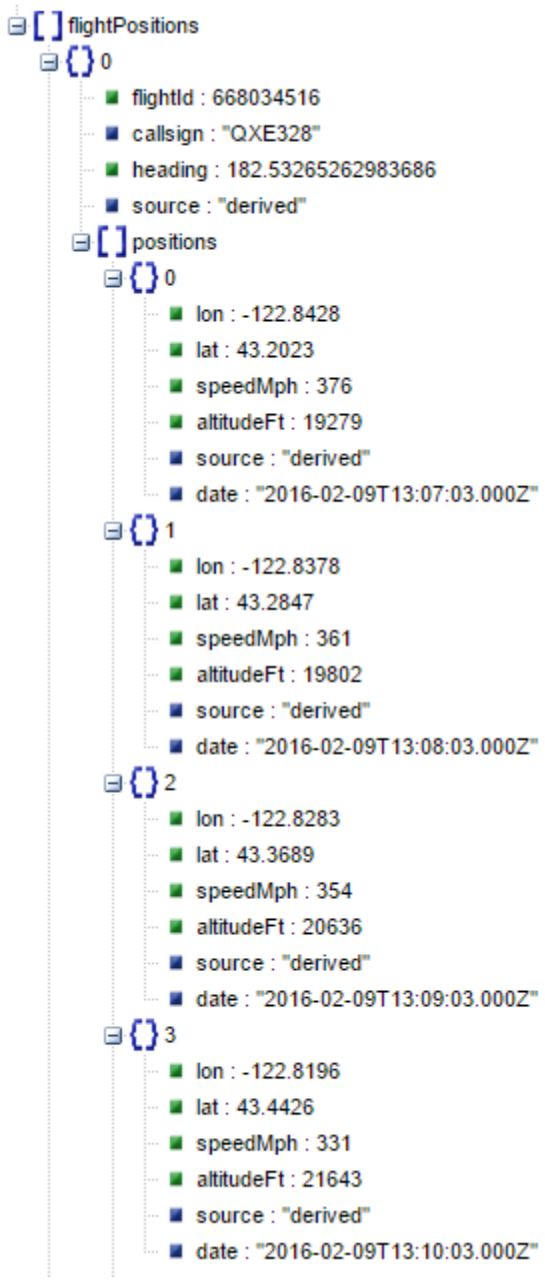


Figure 20: Air traffic data from Flightstats.

Here is an example of how the air traffic data from flightstats looks like (this data have been processed by an online data viewer to make it more visible).

3.3.2.1-Collecting weather data.

To collect the weather data, the same process as the air traffic data will be followed. To get the information, the **weather** API will be used. Once inside, it can be observed that it has different informations to provide.

Figure 21: weather API different options

The API used will be ***All weather products for airport,*** which retrieves all weather products (METAR, TAF, zone forecast) for the area within a 5 mile radius from the center of the airport.

PARAMETER	VALUE	DESCRIPTION
appId	(required)	Application ID
appKey	(required)	Application key
airport	ABQ	Airport code
codeType		Type of airport code: 'IATA', 'ICAO', or 'FS'. If not specified, all domains will be searched in the order stated.

Figure 22: Parameters required by the *All weather products for airport* API

To get the weather conditions, the access parameters will have to be defined first. Then the airport code. To cover all the surface of the sector scenario, the weather data from all the airports and aerodromes that are inside the sector scenario, must be collected in order to cover all the air space of the scenario.

Here is an example of how the weather data from flightstats looks like (this data have been processed by an online data viewer to make it more visible). Python also returns a URL in which the data requested, with the parameters requests, is saved. This makes easier the step of upload the data to python.

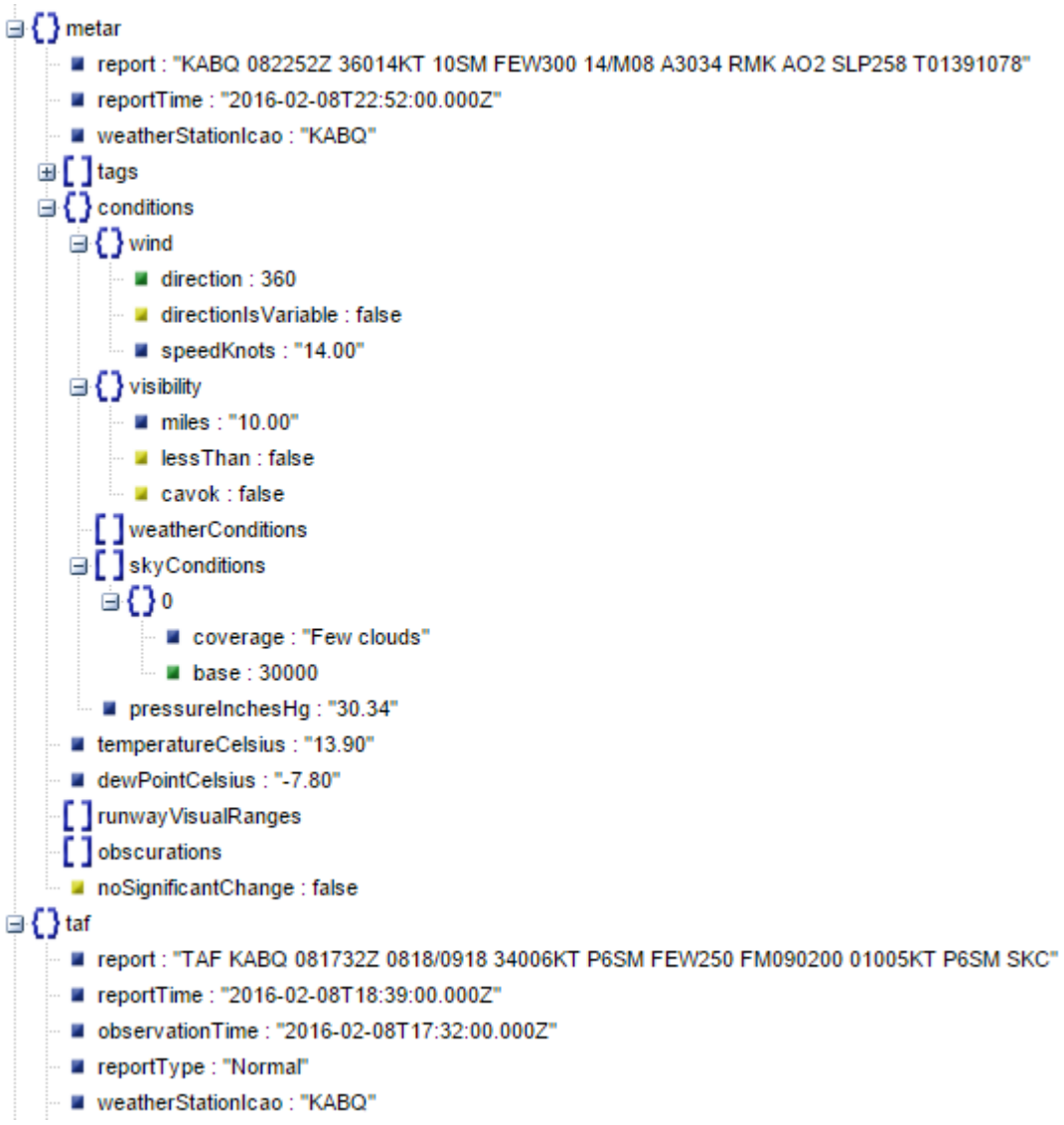


Figure 23: weather data from Flightstats.

3.4-Data processing and cleaning.

Once collected the data is time to start programming the interface, using the programming tool mentioned before, Pycharm(python 2.7). What it will be done during the programming phase is:

1. First get the data collected, upload it and make it processable to the programming tool(Pycharm).

2. Second, it will be merge it and clean it to extract the right information, to create an output data.

3. Third, some python libraries, and toolkits will be used in order to make the data output visible and easier to understand, creating maps, tables, graphs, etc.

3.4.1- Retrieving the data to Pycharm.

To retrieve the data form Flightstats, first of all it has to be indicated the access parameters appId, and appKey in our programming tool (Pycharm).

appID = '5e8642fb'
appKey = '8e598ccab0b9c7e82d523d3ec0fc3b3d'

Once indicated the appID and the appKey, the programming tool will be able to read any data from Flightstats.

3.4.2- Uploading, Merging, and cleaning the data.

Once Pycharm is ready to read the data, the next step will be to upload it, merge it, and clean it.
As it is known, sometimes the data has lots of irrelevant informations, for this reason it is necessary to clean this data, to separate the data necessary to make the interface work properly, from the irrelevant datas.

1- Air traffic data

To upload the air traffic data extracted earlier, flightstats gives an easy access, returning a URL where the requested data is located. So it just has to be indicated in our programming tool, and the data will be uploaded.
Once uploaded, the data has to be processable to the python libraries as well.

```
#---------- FLIGHTSNEAR API ----------
# OBTAINING FLIGHTS NEAR TO POINT A
request=Request('https://api.flightstats.com/flex/flightstatus/rest/v2/json/flightsNear/45/-125/40/-120?appId='+appID+'&appKey='+appKey+'&maxFlights=1000')
response_flights_near = urlopen(request)
fn = response_flights_near.read()
flights_near1 = json.loads(fn)
flights_near1 = pd.DataFrame(json_normalize(flights_near1['flightPositions']))

# OBTAINING FLIGHTS NEAR TO POINT B
request=Request('https://api.flightstats.com/flex/flightstatus/rest/v2/json/flightsNear/37/-125/32/-120?appId='+appID+'&appKey='+appKey+'&maxFlights=1000')
response_flights_near = urlopen(request)
fn = response_flights_near.read()
flights_near2 = json.loads(fn)
flights_near2 = pd.DataFrame(json_normalize(flights_near2['flightPositions']))
```

As shown the air traffic data it's extracted for two points, A, and B, this happens because the Flightstats API ***Flights within bounding box*** doesn't allow the sides of box(by a box it can be understood a sector) to exceed five degrees in longitude, and the defined sector in this study exceeded this rule. So, to solve this problem, it was divided the sector in two, and extracted to boxes, A and B, then merging the data of booth sectors, finally created the first sector defined.

```
# COMBINING FLIGHTS NEAR TO POINTS A AND B IN A SINGLE DATAFRAME
flights_near = flights_near1.append(flights_near2)
flights_near = flights_near.dropna(subset=['flightId'])
```

Once merged the air traffic data it's time to clean it and extract the important information, to create an output data.

```
# EXTRACTING ALTITUDE, LAT & LON FOR EACH FLIGHT
flightdata = []
for index, row in flights_near.iterrows():
    flight_id = row['flightId']
    for position in row['positions']:
        position['flightId'] = flight_id
        series = pd.Series(position)
        flightdata.append(series).
```

The relevant information inside the air traffic data downloaded is the trajectory. The parameters altitude, latitude and longitude can now be extracted.

2- Weather data

The weather data will follow the same process as the air traffic data, just with the difference that the weather data doesn't needs to be merged, due to the fact that all the "weather points" are inside the defined sector.

```
#---------- WEATHER API ----------
# EXTRACTING WEATHER DATA FOR POINTS A (AIRPORT 'PDX') AND B (AIRPORT 'LAX')
airport_from = 'PDX'
airport_to = 'LAX'
request =
Request('https://api.flightstats.com/flex/weather/rest/v1/json/metar/'+airport_from+'?appId='+appID+'&appKey
='+appKey)
response_weather = urlopen(request)
w = response_weather.read()
metar = json.loads(w)
wind1 = pd.DataFrame(json_normalize(metar['metar']))
wind_airport_from = pd.DataFrame(json_normalize(metar['metar']))

request =
Request('https://api.flightstats.com/flex/weather/rest/v1/json/metar/'+airport_to+'?appId='+appID+'&appKey='+
appKey)
response_weather = urlopen(request)
w = response_weather.read()
metar = json.loads(w)
wind_airport_to = pd.DataFrame(json_normalize(metar['metar']))
```

The weather information will not be cleaned, because all parameters within the weather data will be used, allowing the user accessing a large amount of weather information, that is clearly important.

3.4.3- Making the data output visible

At this point the programming phase is almost done. The next final step, and of this phase, consists into make the data output more visible to the controller, allowing the interface to create graphs, maps, tables, to make the understanding of the information easier.

To achieve this purpose, some python libraries such as mpl_toolkits, or matplotlib have been used.

```
#CREATING A MAP WITH  TRAJECTORIES

from mpl_toolkits.basemap import Basemap
import seaborn as sb
import matplotlib.pyplot as plt
import numpy as np

sb.set_style("whitegrid")


map2 = Basemap(projection='merc', lat_0 = 37, lon_0 = -125,
        resolution = 'h', area_thresh = 0.1,
        llcrnrlon=-125, llcrnrlat=37,
        urcrnrlon=-120, urcrnrlat=32)

# map.drawcoastlines()
# map.drawcountries()
# map.drawstates()
# map.fillcontinents(color = 'coral')
# map.drawmapboundary()
#
map2.drawcoastlines()
map2.drawcountries()
map2.drawstates()
map2.fillcontinents(color = 'coral')
map2.drawmapboundary()

lons = list(flights_positions['lon']) #[-135.3318, -134.8331, -134.6572]
lats = list(flights_positions['lat']) #[57.0799, 57.0894, 56.2399]
x,y = map2(lons, lats)
map2.plot(x, y, 'bo', markersize=18)

labels = list(flights_positions['flightId'])
x_offsets = [10000, -30000, -25000]
y_offsets = [5000, -25000, -25000]

for label, xpt, ypt, x_offset, y_offset in zip(labels, x, y, x_offsets, y_offsets):
    plt.text(xpt+x_offset, ypt+y_offset, label)

plt.show()
```

This map will show the extracted trajectories from the air traffic data, passing around the sector scenario previously defined.

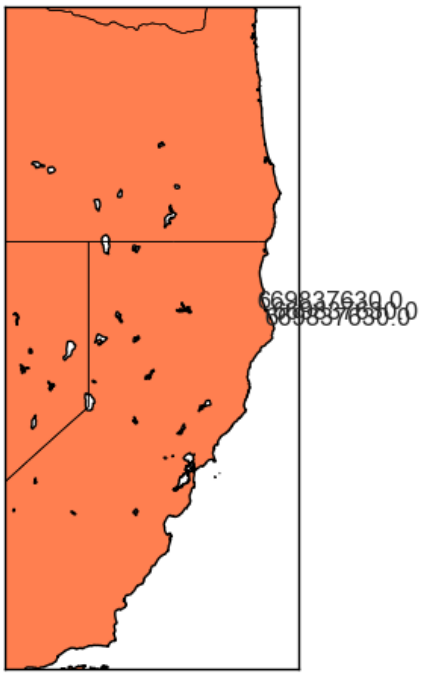


Figure 24: The sector scenario created by Python (without trajectories).

```
# POLAR PORJECTION INDICATING THE WIND DIRECTION

import matplotlib
import numpy as np
from matplotlib.pyplot import figure, show, rc, grid
import seaborn as sb

sb.set_style("whitegrid")

# radar green, solid grid lines
rc('grid', color='#316931', linewidth=1, linestyle='-')
rc('xtick', labelsize=15)
rc('ytick', labelsize=15)

# force square figure and square axes looks better for polar, IMO
width, height = matplotlib.rcParams['figure.figsize']
size = min(width, height)
# make a square figure
fig = figure(figsize=(size, size))

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8], polar=True, axisbg='#d5de9c')

ax.set_title("Your title", fontsize=20)
show()
```

This graph is a 360º polar projection that will show in which direction the wind comes from.

4.-Results

At this point, the interface is already finished, so it's time to run it and see how it works.

altitudeFt	date	flightId	lat	lon	source	speedMph
39000	2016-01-26T15:16:48.000Z	661604677	41.4200	-122.8542	derived	491
39000	2016-01-26T15:17:48.000Z	661604677	41.3017	-122.8503	derived	491
39000	2016-01-26T15:18:48.000Z	661604677	41.1794	-122.8421	derived	501
39000	2016-01-26T15:19:48.000Z	661604677	41.0689	-122.8356	derived	477
39000	2016-01-26T15:20:48.000Z	661604677	40.9496	-122.8283	derived	483
39000	2016-01-26T15:21:48.000Z	661604677	40.8274	-122.8211	derived	497
39000	2016-01-26T15:22:48.000Z	661604677	40.7045	-122.8137	derived	510
39000	2016-01-26T15:23:48.000Z	661604677	40.5850	-122.8069	derived	506
39000	2016-01-26T15:24:48.000Z	661604677	40.4624	-122.8001	derived	512
39000	2016-01-26T15:25:48.000Z	661604677	40.3439	-122.7930	derived	500
38000	2016-01-26T15:16:32.000Z	661605296	41.0085	-122.4785	derived	505

Table 1: Air traffic data output

As shown in the projection, the interface works properly, it returns the data cleaned and organized. This table shows the airplane trajectory by giving the altitude, latitude, longitude, plus the time, the date, and the speed.

It also returns the weather conditions, for example in the following projection, returns the wind direction, the wind speed, and the visibility.

	conditions.wind.direction	conditions.wind.speedKnots	conditions.visibility.miles
0	120	9.00	9.00

Table2: weather data output.

In the next subchapter it will be explained how it works and how to use it, showing examples of its use.

4.1- Examples of use

As shown before the interface works properly, returns the data cleaned and organized, but this data is still complicate to understand, it's not so visible, and being not so useful. During this subchapter it will be explained some examples of how to turn it into a data easy to understand, visible, and useful, by using different functions of the interface.

1- **Example 1:** How many flights are passing arround the defined sector, by an altitude of 32.000ft.

To know it, the controller will have to set the altitude on 32.000ft.

flights_positions =flights_positions [flights_positions ['altitudeFt']>32000]

Then excute the function ***#CREATING A MAP WITH TRAJECTORIES.***

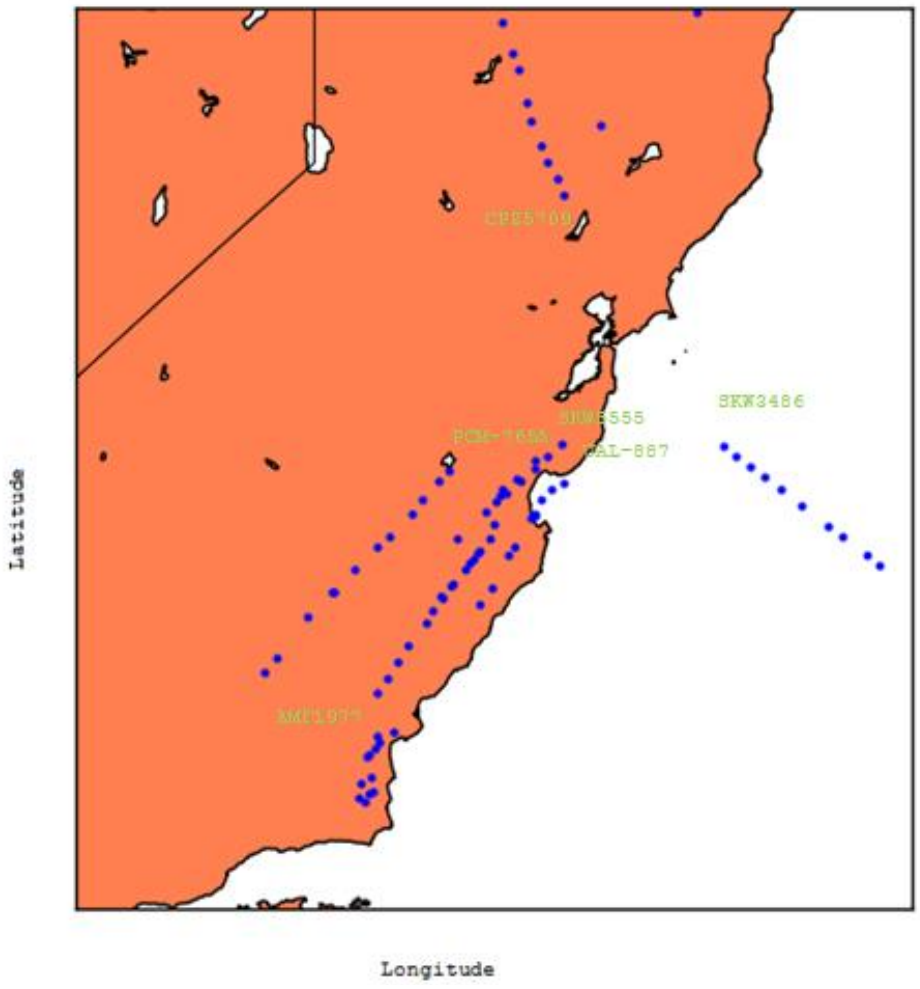


Figure 25: Flights passing around the sector scenario, by an altitude of 32000ft.

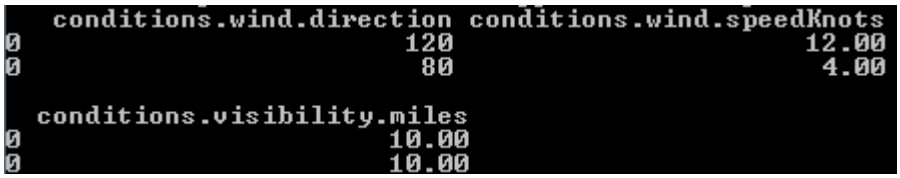
As it can be observed in the projection, the interface creates a map that simulates the sector scenario, showing the trajectories of the flights that are passing around with an altitude of 32000ft.

2- Example 2: The current weather conditions inside the sector scenario.

The controller will have to specify wich parameter wants to know.

```
columns = ['conditions.wind.direction',
           'conditions.wind.speedKnots',
           'conditions.visibility.miles']
display(wind_airport_from[columns].append(wind_airport_to[columns]))
```

In this case the parameters asked are wind speed, sky conditions, and the visibility.



	conditions.wind.direction	conditions.wind.speedKnots
0	120	12.00
0	80	4.00

	conditions.visibility.miles
0	10.00
0	10.00

Figure 26: weather conditions.

As observed, the table shows the parameter asked for the controller. The 0 observed at the left side of the image, means that this weather conditions are for the trajectory 0.

3- Example3: A graph showing the wind direction

To get this information, the controller will have to execute the function ***#POLAR PORJECTION INDICATING THE WIND DIRECTION.***

As shown in the projection, this graph shows a polar projection with a point indicating the direction in which the wind comes from.

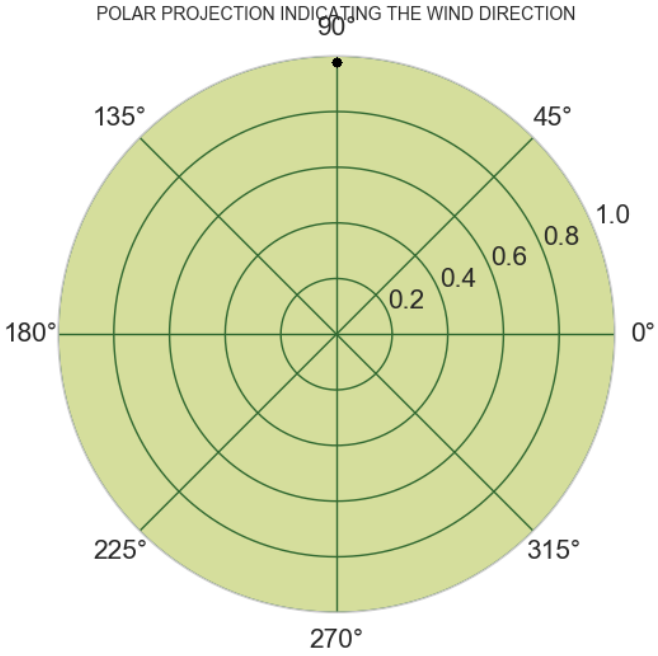


Figure 27: Polar projection indicating the wind direction.

So as seen the interface works properly, and it's able to provide valuable information to the controller, in real time. Helping it in the process of decision making regarding the availability of a certain sector, being defined as able for free-routing or not. As a result of receiving this information, now the controller has more facilities to take the right decision.

Conclusions

The main objective of this study was to create an interface able to help the controller in the process of decision making regarding the availability of certain sectors for free routing, by using data science techniques. It can be concluded that the satisfactory result has been achieved because the interface works properly (according to stated goal) taking into account the weather conditions and the air traffic flows.

The most innovative contribution of this study has been the application of data engineering, visualization, statistical analysis and software programming techniques (belonging to the data science field) in the aviation sector. It should be noticed, though the combined usage of these techniques can provide multiple benefits to the aviation sector and to the new SESAR concepts that are taking off nowadays, data science has still not been commonly used to retrieve knowledge from aeronautical data.

On the other hand it also has to be said that the final result of this study could improved. The first thing in which the interface could be improved is in the introduction of the weather conditions. As it has been explained during the study, to cover the entire sector scenario with the weather conditions, all the airports inside the sector scenario were used as a weather points, due to the fact that Flightstats just provides the weather conditions from the airports and an area of 5 miles from the center of it. This is no so helpful because a large amount of data must be collected, and handled. For this reason it would be a good idea to search another data provider able to provide just one weather data report for all the surface of the sector scenario.

Another aspect of the interface that might be improved is the addition of a conflict detection and resolution function, like the projects mentioned before, STREAM, and TESA. With a conflict detection and resolution function within, the interface would be able to detect if the trajectory of a flight that is free routing can enter in conflict with another plane trajectory free routing in the same sector.

As a final conclusion it has to be said that even thought the interface works properly and transforms the data received from FlightStats in a valuable data output, if the improvements mentioned before were applied (i.e. a better weather data provider, and a conflict detection and resolution function), the interface would significantly increase its applicability

References

[**1]-**What is data science? Last visit: 12/10/2015
http://www.complexworld.eu/datascienceworkshop/data-science/

[2]- 2nd workshop on data science in aviation. Last visit: 10/16/2015
http://www.complexworld.eu/2nd-workshop-on-data-science-in-aviation/

[3]-Ruiz, S. 2015. ATM actual y nuevo paradigma. Technical Innovation Cluster on Aeronautical Management, Universidad Autónoma de Barcelona, Sabadell(Barcelona), Spain.

[4]- System Wide Information Management (SWIM). Last visit: 10/11/2015
http://www.eurocontrol.int/swim

[5]- Kraus, J. 2011. Free Route Airspace FRA. Prague: Czech Technical University. Faculty of Transportation. Department of Air Transportation.

[6]-Ranieri, A., Martinez, R., Piera, M.A., Lopez, J., Vilaplana, M., 2011.
Strategic Trajectory De-confliction to Enable Seamless Aircraft Conflict

[7]-Schuster, W., Ochieng,W., Majumdar, A.,2011.
Trajectory prediction and conflict resolution for Enroute-to-enroute Seamless Air traffic management. Departament of Civil and Environmental Engineering, Imperial College London, London, United Kingdom.

[8]-What data science can do for aviation? Last visit: 12/20/2015
http://www.complexworld.eu/datascienceworkshop/data-science/

[9]-Data Science. Last visit: 01/10/2016
https://en.wikipedia.org/wiki/Data_science

[10]-Information Engineering. Last visit: 01/16/2016
https://en.wikipedia.org/wiki/Information_engineering

[11]-What Is Statistics? Last visit: 01/16/2016
http://www.amstat.org/careers/whatisstatistics.cfm

[12]-Advanced Computer Techniques. Last visit: 01/16/2016
https://en.wikipedia.org/wiki/Advanced_Computer_Techniques

[13]-Machine Learning What it is & why it matters. Last visit: 01/18/2016
http://www.sas.com/en_us/insights/analytics/machine-learning.html

[14]-A very short history of Data Science. Last visit: 01/15/2016
http://www.forbes.com/sites/gilpress/2013/05/28/a-very-short-history-of-data-science/#1ad6c39069fd

[15]-Data Science History. Last visit: 01/15/2016
https://en.wikipedia.org/wiki/Data_science

[16]- Who is Flightstats. Last visit: 01/25/2016
http://www.flightstats.com/company/corporate/about-us/

Annexes

Annex A

#Interface code

```
import pandas as pd
from urllib2 import Request, urlopen
import json
from pandas.io.json import json_normalize

appID = '5e8642fb'
appKey = '8e598ccab0b9c7e82d523d3ec0fc3b3d'


#---------- FLIGHTSNEAR API ----------
# OBTAINING FLIGHTS NEAR TO POINT A
request=Request('https://api.flightstats.com/flex/flightstatus/rest/v2/json
/flightsNear/45/-125/40/-
120?appId='+appID+'&appKey='+appKey+'&maxFlights=1000')
response_flights_near = urlopen(request)
fn = response_flights_near.read()
flights_near1 = json.loads(fn)
flights_near1 =
pd.DataFrame(json_normalize(flights_near1['flightPositions']))

# OBTAINING FLIGHTS NEAR TO POINT B
request=Request('https://api.flightstats.com/flex/flightstatus/rest/v2/json
/flightsNear/37/-125/32/-
120?appId='+appID+'&appKey='+appKey+'&maxFlights=1000')
response_flights_near = urlopen(request)
fn = response_flights_near.read()
flights_near2 = json.loads(fn)
flights_near2 =
pd.DataFrame(json_normalize(flights_near2['flightPositions']))

# COMBINING FLIGHTS NEAR TO POINTS A AND B IN A SINGLE DATAFRAME
flights_near = flights_near1.append(flights_near2)
flights_near = flights_near.dropna(subset=['flightId'])

# EXTRACTING ALTITUDE, LAT & LON FOR EACH FLIGHT
flightdata = []
for index, row in flights_near.iterrows():
    flight_id = row['flightId']
    for position in row['positions']:
        position['flightId'] = flight_id
        series = pd.Series(position)
        flightdata.append(series)

flights_positions = pd.DataFrame(flightdata)


#---------- WEATHER API ----------
# EXTRACTING WEATHER DATA FOR POINTS A (AIRPORT 'PDX') AND B (AIRPORT
'LAX')
airport_from = 'PDX'
airport_to = 'LAX'
```

```
request =
Request('https://api.flightstats.com/flex/weather/rest/v1/json/metar/'+airp
ort_from+'?appId='+appID+'&appKey='+appKey)
response_weather = urlopen(request)
w = response_weather.read()
metar = json.loads(w)
wind1 = pd.DataFrame(json_normalize(metar['metar']))
wind_airport_from = pd.DataFrame(json_normalize(metar['metar']))

request =
Request('https://api.flightstats.com/flex/weather/rest/v1/json/metar/'+airp
ort_to+'?appId='+appID+'&appKey='+appKey)
response_weather = urlopen(request)
w = response_weather.read()
metar = json.loads(w)
wind_airport_to = pd.DataFrame(json_normalize(metar['metar']))



#---------- RATINGS FOR ROUTE API ----------
# OBTAINING RATINGS FOR FLIGHTS FOR A GIVEN ROUTE
request=Request('https://api.flightstats.com/flex/ratings/rest/v1/json/rout
e/'+airport_from+'/'+airport_to+'?appId='+appID+'&appKey='+appKey)
response_ratings = urlopen(request)
fn = response_ratings.read()
ratings = json.loads(fn)
ratings = pd.DataFrame(json_normalize(ratings['ratings']))


#---------- PRINTING OUT THE RESULTS ----------
column_names_flights_near = ['callsign',
                             'flightId',
                             'tailNumber',
                             'heading']

column_names_positions = ['flightId','lat','lon','altitudeFt']

column_names_weather_wind = ['conditions.wind.direction',
                             'conditions.wind.speedKnots']

column_names_ratings = ['flightNumber',
                        'allDelayStars']

print flights_near[column_names_flights_near]
print flights_positions[column_names_positions]

flights_positions =
flights_positions[flights_positions['altitudeFt']>31000]

print wind_airport_from[column_names_weather_wind]
print wind_airport_to[column_names_weather_wind]
print ratings[column_names_ratings]

print "Average 'allDelayStars': {0}
".format(ratings['allDelayStars'].mean())



from mpl_toolkits.basemap import Basemap
import seaborn as sb
import matplotlib.pyplot as plt
```

```
import numpy as np

sb.set_style("whitegrid")

# map = Basemap(projection='merc', lat_0 = 40, lon_0 = -120,
#               resolution = 'h', area_thresh = 0.1,
#               llcrnrlon=-120, llcrnrlat=40.0,
#               urcrnrlon=-125, urcrnrlat=45.0)

map2 = Basemap(projection='merc', lat_0 = 37, lon_0 = -125,
              resolution = 'h', area_thresh = 0.1,
              llcrnrlon=-125, llcrnrlat=37,
              urcrnrlon=-120, urcrnrlat=32)

# map.drawcoastlines()
# map.drawcountries()
# map.drawstates()
# map.fillcontinents(color = 'coral')
# map.drawmapboundary()
#
# lons = list(flights positions['lon']) #[-135.3318, -134.8331, -134.6572]
# lats = list(flights_positions['lat']) #[57.0799, 57.0894, 56.2399]
# x,y = map(lons, lats)
# map.plot(x, y, 'bo', markersize=18)

map2.drawcoastlines()
map2.drawcountries()
map2.drawstates()
map2.fillcontinents(color = 'coral')
map2.drawmapboundary()

lons = list(flights_positions['lon']) #[-135.3318, -134.8331, -134.6572]
lats = list(flights_positions['lat']) #[57.0799, 57.0894, 56.2399]
x,y = map2(lons, lats)
map2.plot(x, y, 'bo', markersize=18)

labels = list(flights_positions['flightId']) #['Sitka', 'Baranof\n  Warm
Springs', 'Port Alexander']
x_offsets = [10000, -30000, -25000]
y_offsets = [5000, -25000, -25000]

for label, xpt, ypt, x_offset, y_offset in zip(labels, x, y, x_offsets,
y_offsets):
    plt.text(xpt+x_offset, ypt+y_offset, label)

plt.show()
```

Annex B

#Flightstats: Air traffic data


```
{
 "request": {
  "topLatitude": {
   "requested": "45.0",
   "interpreted": 45
  },
  "leftLongitude": {
   "requested": "-125.0",
   "interpreted": -125
  },
  "bottomLatitude": {
   "requested": "40.0",
   "interpreted": 40
  },
  "rightLongitude": {
   "requested": "-120.0",
   "interpreted": -120
  },
  "maxFlights": {
   "requested": "5",
   "interpreted": 5
  },
  "extendedOptions": {},
  "url": "https://api.flightstats.com/flex/flightstatus/rest/v2/json/flightsNear/45/-125/40/-120"
 },
 "appendix": {},
 "flightPositions": [
  {
   "flightId": 667994449,
   "callsign": "PEN160",
   "heading": 233.76043257393343,
   "source": "derived",
   "positions": [
    {
     "lon": -124.2342,
     "lat": 41.7731,
     "speedMph": 43,
     "source": "derived",
     "date": "2016-02-09T15:22:38.000Z"
    },
    {
     "lon": -124.0632,
     "lat": 41.8667,
     "speedMph": 178,
     "altitudeFt": 2037,
```

```
 "source": "derived",
 "date": "2016-02-09T15:22:53.000Z"
},
{
 "lon": -124.0366,
 "lat": 41.8667,
 "speedMph": 179,
 "altitudeFt": 2270,
 "source": "derived",
 "date": "2016-02-09T15:24:53.000Z"
},
{
 "lon": -124.3275,
 "lat": 41.8831,
 "speedMph": 332,
 "altitudeFt": 4811,
 "source": "derived",
 "date": "2016-02-09T15:25:53.000Z"
},
{
 "lon": -124.3048,
 "lat": 41.7477,
 "speedMph": 255,
 "altitudeFt": 6679,
 "source": "derived",
 "date": "2016-02-09T15:26:53.000Z"
},
{
 "lon": -124.1652,
 "lat": 41.7588,
 "speedMph": 228,
 "altitudeFt": 8549,
 "source": "derived",
 "date": "2016-02-09T15:27:53.000Z"
},
{
 "lon": -124.0931,
 "lat": 41.8344,
 "speedMph": 236,
 "altitudeFt": 10417,
 "source": "derived",
 "date": "2016-02-09T15:28:53.000Z"
}
]
},
{
"flightId": 668043543,
"callsign": "ASA833",
"heading": 28.561731649286138,
"source": "derived",
"positions": [
```

```
{
 "lon": -123.1134,
 "lat": 45.1901,
 "speedMph": 604,
 "altitudeFt": 19061,
 "source": "derived",
 "date": "2016-02-09T15:19:19.000Z"
},
{
 "lon": -123.1759,
 "lat": 45.1092,
 "speedMph": 521,
 "altitudeFt": 20618,
 "source": "derived",
 "date": "2016-02-09T15:20:19.000Z"
},
{
 "lon": -123.2813,
 "lat": 45.028,
 "speedMph": 512,
 "altitudeFt": 22600,
 "source": "derived",
 "date": "2016-02-09T15:21:19.000Z"
},
{
 "lon": -123.407,
 "lat": 44.9573,
 "speedMph": 503,
 "altitudeFt": 23944,
 "source": "derived",
 "date": "2016-02-09T15:22:19.000Z"
},
{
 "lon": -123.5445,
 "lat": 44.8837,
 "speedMph": 511,
 "altitudeFt": 25258,
 "source": "derived",
 "date": "2016-02-09T15:23:19.000Z"
},
{
 "lon": -123.692,
 "lat": 44.7971,
 "speedMph": 545,
 "altitudeFt": 26628,
 "source": "derived",
 "date": "2016-02-09T15:24:19.000Z"
},
{
 "lon": -123.8339,
 "lat": 44.7122,
```

```
  "speedMph": 562,
  "altitudeFt": 27549,
  "source": "derived",
  "date": "2016-02-09T15:25:19.000Z"
 },
 {
  "lon": -123.9714,
  "lat": 44.6302,
  "speedMph": 562,
  "altitudeFt": 28736,
  "source": "derived",
  "date": "2016-02-09T15:26:19.000Z"
 },
 {
  "lon": -124.1012,
  "lat": 44.5514,
  "speedMph": 543,
  "altitudeFt": 30256,
  "source": "derived",
  "date": "2016-02-09T15:27:19.000Z"
 },
 {
  "lon": -124.2374,
  "lat": 44.4827,
  "speedMph": 521,
  "altitudeFt": 31676,
  "source": "derived",
  "date": "2016-02-09T15:28:19.000Z"
 }
]
},
{
 "callsign": "BXR1966",
 "heading": 131.52422710474917,
 "source": "derived",
 "positions": [
  {
   "lon": -123.0705,
   "lat": 40.0358,
   "speedMph": 208,
   "altitudeFt": 8403,
   "source": "derived",
   "date": "2016-02-09T15:19:36.000Z"
  },
  {
   "lon": -123.1339,
   "lat": 40.0788,
   "speedMph": 238,
   "altitudeFt": 8449,
   "source": "derived",
   "date": "2016-02-09T15:20:36.000Z"
```

```
},
{
 "lon": -123.1865,
 "lat": 40.1121,
 "speedMph": 235,
 "altitudeFt": 8437,
 "source": "derived",
 "date": "2016-02-09T15:21:36.000Z"
},
{
 "lon": -123.2309,
 "lat": 40.1438,
 "speedMph": 219,
 "altitudeFt": 8366,
 "source": "derived",
 "date": "2016-02-09T15:22:36.000Z"
},
{
 "lon": -123.2712,
 "lat": 40.176,
 "speedMph": 199,
 "altitudeFt": 8293,
 "source": "derived",
 "date": "2016-02-09T15:23:36.000Z"
},
{
 "lon": -123.3119,
 "lat": 40.2089,
 "speedMph": 186,
 "altitudeFt": 8310,
 "source": "derived",
 "date": "2016-02-09T15:24:36.000Z"
},
{
 "lon": -123.3574,
 "lat": 40.2421,
 "speedMph": 187,
 "altitudeFt": 8285,
 "source": "derived",
 "date": "2016-02-09T15:25:36.000Z"
},
{
 "lon": -123.407,
 "lat": 40.2782,
 "speedMph": 201,
 "altitudeFt": 8275,
 "source": "derived",
 "date": "2016-02-09T15:26:36.000Z"
},
{
 "lon": -123.4524,
```

```
  "lat": 40.3106,
  "speedMph": 198,
  "altitudeFt": 8292,
  "source": "derived",
  "date": "2016-02-09T15:27:36.000Z"
 },
 {
  "lon": -123.4981,
  "lat": 40.3438,
  "speedMph": 198,
  "altitudeFt": 8354,
  "source": "derived",
  "date": "2016-02-09T15:28:36.000Z"
 }
]
},
{
 "flightId": 668051734,
 "callsign": "ASA342",
 "tailNumber": "N491AS",
 "heading": 355.4510772292107,
 "source": "derived",
 "positions": [
 {
  "lon": -122.7999,
  "lat": 41.2514,
  "speedMph": 513,
  "altitudeFt": 37000,
  "source": "derived",
  "date": "2016-02-09T15:20:11.000Z"
 },
 {
  "lon": -122.787,
  "lat": 41.1295,
  "speedMph": 505,
  "altitudeFt": 37000,
  "source": "derived",
  "date": "2016-02-09T15:21:11.000Z"
 },
 {
  "lon": -122.7742,
  "lat": 41.0077,
  "speedMph": 503,
  "altitudeFt": 37000,
  "source": "derived",
  "date": "2016-02-09T15:22:11.000Z"
 },
 {
  "lon": -122.7633,
  "lat": 40.9023,
  "speedMph": 458,
```

"altitudeFt": 37000,
"source": "derived",
"date": "2016-02-09T15:23:11.000Z"
},
{
"lon": -122.7499,
"lat": 40.7752,
"speedMph": 488,
"altitudeFt": 37000,
"source": "derived",
"date": "2016-02-09T15:24:11.000Z"
},
{
"lon": -122.737,
"lat": 40.6474,
"speedMph": 519,
"altitudeFt": 37000,
"source": "derived",
"date": "2016-02-09T15:25:12.000Z"
},
{
"lon": -122.7258,
"lat": 40.537,
"speedMph": 489,
"altitudeFt": 37000,
"source": "derived",
"date": "2016-02-09T15:26:12.000Z"
},
{
"lon": -122.713,
"lat": 40.4112,
"speedMph": 515,
"altitudeFt": 37000,
"source": "derived",
"date": "2016-02-09T15:27:12.000Z"
},
{
"lon": -122.6996,
"lat": 40.2817,
"speedMph": 540,
"altitudeFt": 37000,
"source": "derived",
"date": "2016-02-09T15:28:12.000Z"
},
{
"lon": -122.6869,
"lat": 40.1567,
"speedMph": 534,
"altitudeFt": 37000,
"source": "derived",
"date": "2016-02-09T15:29:12.000Z"

```
 }
]
},
{
 "flightId": 668043588,
 "callsign": "QXE571",
 "heading": 354.5804431860859,
 "source": "derived",
 "positions": [
 {
  "lon": -122.9065,
  "lat": 41.7263,
  "speedMph": 395,
  "altitudeFt": 25000,
  "source": "derived",
  "date": "2016-02-09T15:20:09.000Z"
 },
 {
  "lon": -122.8945,
  "lat": 41.6319,
  "speedMph": 396,
  "altitudeFt": 25000,
  "source": "derived",
  "date": "2016-02-09T15:21:09.000Z"
 },
 {
  "lon": -122.8824,
  "lat": 41.5503,
  "speedMph": 371,
  "altitudeFt": 25000,
  "source": "derived",
  "date": "2016-02-09T15:22:09.000Z"
 },
 {
  "lon": -122.8726,
  "lat": 41.4697,
  "speedMph": 349,
  "altitudeFt": 25000,
  "source": "derived",
  "date": "2016-02-09T15:23:09.000Z"
 },
 {
  "lon": -122.86,
  "lat": 41.3819,
  "speedMph": 373,
  "altitudeFt": 25000,
  "source": "derived",
  "date": "2016-02-09T15:24:03.000Z"
 },
 {
  "lon": -122.8617,
```

```
    "lat": 41.3869,
    "speedMph": 338,
    "altitudeFt": 25000,
    "source": "derived",
    "date": "2016-02-09T15:24:09.000Z"
   },
   {
    "lon": -122.8496,
    "lat": 41.2775,
    "speedMph": 366,
    "altitudeFt": 25000,
    "source": "derived",
    "date": "2016-02-09T15:25:18.000Z"
   },
   {
    "lon": -122.8396,
    "lat": 41.1884,
    "speedMph": 368,
    "altitudeFt": 25000,
    "source": "derived",
    "date": "2016-02-09T15:26:18.000Z"
   },
   {
    "lon": -122.824,
    "lat": 41.0837,
    "speedMph": 402,
    "altitudeFt": 25000,
    "source": "derived",
    "date": "2016-02-09T15:27:18.000Z"
   },
   {
    "lon": -122.8104,
    "lat": 40.9819,
    "speedMph": 422,
    "altitudeFt": 25000,
    "source": "derived",
    "date": "2016-02-09T15:28:18.000Z"
   }
  ]
 }
]
}
```

Annex C

#Flightstats: Weather data


```
{
 "request": {
  "url": "https://api.flightstats.com/flex/weather/rest/v1/json/all/LAX",
  "airport": {
   "requestedCode": "LAX",
   "fsCode": "LAX"
  },
  "codeType": {}
 },
 "metar": {
  "report": "KLAX 091453Z VRB04KT 10SM CLR 18/M03 A3014 RMK AO2 SLP204 T01781033 53012",
  "reportTime": "2016-02-09T14:53:00.000Z",
  "weatherStationIcao": "KLAX",
  "tags": [
   {
    "key": "Instrumentation",
    "value": "VFR"
   },
   {
    "key": "Prevailing Conditions",
    "value": "Clear"
   }
  ],
  "conditions": {
   "wind": {
    "directionIsVariable": true,
    "speedKnots": "4.00"
   },
   "visibility": {
    "miles": "10.00",
    "lessThan": false,
    "cavok": false
   },
   "weatherConditions": [],
   "skyConditions": [
    {
     "coverage": "Clear"
    }
   ],
   "pressureInchesHg": "30.14"
  },
  "temperatureCelsius": "17.80",
  "dewPointCelsius": "-3.30",
  "runwayVisualRanges": [],
  "obscurations": [],
  "noSignificantChange": false
 },
```

```
"taf": {
 "report": "TAF\n     AMD KLAX 091455Z 0915/1018 04005KT P6SM SKC \n     FM092100 27008KT
P6SM SKC \n     FM100300 VRB03KT P6SM SKC",
 "reportTime": "2016-02-09T15:57:00.000Z",
 "observationTime": "2016-02-09T14:55:00.000Z",
 "reportType": "Amendment",
 "weatherStationIcao": "KLAX",
 "forecasts": [
  {
   "type": "Base",
   "start": "2016-02-09T15:00:00.000Z",
   "end": "2016-02-10T18:00:00.000Z",
   "conditions": {
    "wind": {
     "direction": 40,
     "directionIsVariable": false,
     "speedKnots": "5.00"
    },
    "visibility": {
     "miles": "6.00",
     "lessThan": false,
     "cavok": false
    },
    "weatherConditions": [],
    "skyConditions": [
     {
      "coverage": "Sky clear"
     }
    ]
   }
  },
  {
   "type": "From",
   "start": "2016-02-09T21:00:00.000Z",
   "end": "2016-02-10T00:00:00.000Z",
   "conditions": {
    "wind": {
     "direction": 270,
     "directionIsVariable": false,
     "speedKnots": "8.00"
    },
    "visibility": {
     "miles": "6.00",
     "lessThan": false,
     "cavok": false
    },
    "weatherConditions": [],
    "skyConditions": [
     {
      "coverage": "Sky clear"
     }
```

```
    ]
   }
  },
  {
   "type": "From",
   "start": "2016-02-10T03:00:00.000Z",
   "end": "2016-02-11T00:00:00.000Z",
   "conditions": {
    "wind": {
     "directionIsVariable": true,
     "speedKnots": "3.00"
    },
    "visibility": {
     "miles": "6.00",
     "lessThan": false,
     "cavok": false
    },
    "weatherConditions": [],
    "skyConditions": [
     {
      "coverage": "Sky clear"
     }
    ]
   }
  }
 ]
},
 "zoneForecast": {
  "header": "LOS ANGELES COUNTY COAST INCLUDING DOWNTOWN LOS ANGELES-
\nINCLUDING...MALIBU...SANTA MONICA...BEVERLY HILLS...HOLLYWOOD...\nLONG BEACH\n330 AM
PST TUE FEB 9 2016\n",
  "general": "Expires:201602100015;;476950\nFPUS56 KLOX 091107\nZFPLOX\n\nZONE FORECASTS
FOR SOUTHWESTERN CALIFORNIA\nNATIONAL WEATHER SERVICE LOS ANGELES/OXNARD CA\n330
AM PST TUE FEB 9 2016\n\n",
  "zones": [
   "CAZ041"
  ],
  "reportTime": "2016-02-09T03:30:00.000-08:00",
  "cities": [
   "Malibu",
   "Santa Monica",
   "Beverly Hills",
   "Hollywood",
   "Long Beach"
  ],
  "zoneNames": [
   "Los Angeles County Coast Including Downtown Los Angeles"
  ],
  "dayForecasts": [
   {
    "day": "Today",
```

```
  "forecast": "Sunny. Highs in the 80s to around 90. From malibu through the hollywood hills...areas of north winds 15 to 25 mph.",
  "start": "2016-02-09T06:00:00.000",
  "end": "2016-02-09T18:00:00.000",
  "tags": [
   {
    "key": "Prevailing Conditions",
    "value": "Sunny"
   }
  ]
 },
 {
  "day": "Tonight",
  "forecast": "Clear. Lows in the mid 50s to around 60. ",
  "start": "2016-02-09T18:00:00.000",
  "end": "2016-02-10T06:00:00.000",
  "tags": [
   {
    "key": "Prevailing Conditions",
    "value": "Clear"
   }
  ]
 },
 {
  "day": "Wednesday",
  "forecast": "Sunny. Highs in the upper 70s and 80s. Local northeast winds around 15 mph in the morning.",
  "start": "2016-02-10T06:00:00.000",
  "end": "2016-02-10T18:00:00.000",
  "tags": [
   {
    "key": "Prevailing Conditions",
    "value": "Sunny"
   }
  ]
 },
 {
  "day": "Wednesday Night",
  "forecast": "Mostly clear. Lows in the lower to mid 50s. ",
  "start": "2016-02-10T18:00:00.000",
  "end": "2016-02-11T06:00:00.000",
  "tags": [
   {
    "key": "Prevailing Conditions",
    "value": "Mostly Clear"
   }
  ]
 },
 {
  "day": "Thursday",
```

```
   "forecast": "Sunny. Highs from the lower to mid 70s at the beaches to the lower to mid 80s
inland.",
   "start": "2016-02-11T06:00:00.000",
   "end": "2016-02-11T18:00:00.000",
   "tags": [
    {
     "key": "Prevailing Conditions",
     "value": "Sunny"
    }
   ]
  },
  {
   "day": "Thursday Night Through Washingtons Birthday",
   "forecast": "Mostly clear. Lows in the lower to mid 50s. Highs from the upper 60s to mid 70s at the
beaches to the lower 80s inland. ",
   "start": "2016-02-11T18:00:00.000",
   "end": "2016-02-12T18:00:00.000",
   "tags": [
    {
     "key": "Prevailing Conditions",
     "value": "Mostly Clear"
    }
   ]
  }
 ]
},
"appendix": {
 "airports": [
  {
   "fs": "LAX",
   "iata": "LAX",
   "icao": "KLAX",
   "faa": "LAX",
   "name": "Los Angeles International Airport",
   "street1": "One World Way",
   "street2": "",
   "city": "Los Angeles",
   "cityCode": "LAX",
   "stateCode": "CA",
   "postalCode": "90045-5803",
   "countryCode": "US",
   "countryName": "United States",
   "regionName": "North America",
   "timeZoneRegionName": "America/Los_Angeles",
   "weatherZone": "CAZ041",
   "localTime": "2016-02-09T07:32:56.971",
   "utcOffsetHours": -8,
   "latitude": 33.943399,
   "longitude": -118.408279,
   "elevationFeet": 126,
   "classification": 1,
```

```
    "active": true,
    "delayIndexUrl":
"https://api.flightstats.com/flex/delayindex/rest/v1/json/airports/LAX?codeType=fs"
   }
  ]
 }
}
```