DATA SCIENCE IN GCP

CS6030 – SUMMER 2 2018

TEAM MEMBERS:

- ANDRE SIHOMBING

OUTLINE

- Data Science Definition
- Data Science on GCP
- Qwiklabs

DATA SCIENCE DEFINITION

- **Data Science** is the extraction of actionable knowledge directly from data through a process of **discovery, hypothesis, and analytical hypothesis analysis.**
- **A Data scientist** is a practitioner who has sufficient knowledge of the overlapping regimes of expertise in business needs, domain knowledge, **analytical skills** and programming expertise to manage the end-to-end scientific method process through each stage in the big data lifecycle.
- **Big Data** refers to digital data volume, velocity and/or variety whose management requires scalability across coupled horizontal resources.

K. Hwang and M. Chen, *Big-Data Analytics for Cloud, IoT and Cognitive Computing*, First. John Wiley & Sons Ltd, 2017

?

=

- Data science is applying the scientific method to your data. This doesn't assume that your data has to be big.
- In fact there's a great book called Data Smart, which introduces data science statistics using only spreadsheets.
- Nevertheless, one of the most active areas in data science is around big data.

Image source: http://crossing-technologies.com/big-data-analytics/ & https://towardsdatascience.com/the-best-data-science-learning-resources-out-there-and-my-journey-into-data-science-81c5a6ec67bc

COMMONLY TOOLS IN DATA SCIENCE

- **Storing : Hadoop, Cassandra, and POST REST SQL**
- **Scrubbing: scripting tools and programming languages like Python and SCALLOP**
- **Analyzing: R, SBSS, and Python's data libraries**

Note:

- We need tools to make discoveries but learning all this tools not make you a data scientist.
- It's the scientific method, and not the tools, that make someone a data scientist.
- Data scientists usually spend most their time scrubbing the data. Some of them say that they spend up to 90% of their time scrubbing their data to make it more usable

DIFFERENT FORMS OF DATA

- **Structured:** has a set order and a consistent format, **inflexible and required a lot of upfront design**
  - **Example: XLS spread sheet, Databases, CSV File**

- **Semistructured:** Does not have a proper format asociated
  - **Example: Email, Log File, Doc File**

- **Unstructured:** Does not have any format associated
  - **Example: Image files, Audio files, video files**

Note:

- Most of data in the world does not follow a specific format and structure. Data like videos, pictures, and audio have no defined structure.
- The best way to think about data science is to focus on the science, not the data
- It is not enough to simply collect the data or use it as is. We must understand the data

DATA SCIENCE ON GCP

Because of the ongoing movement to the cloud, data engineers can do the job that used to be done by four people with four different sets of skills. With the advent of autoscaling, serverless, managed infrastructure that is easy to program, there are more and more people who can build scalable systems.

With the introduction of tools like BigQuery, in which tables are denormalized and the administration overhead is minimal, the role of a database administrator is considerably diminished.

Roles in Data science

1. Data analysts
2. Database administrators
3. Data scientists
4. Systems programmers

Complexity reduce by

autoscaled, fully managed services, and simpler and simpler data science packages

Engineer write a data science workload, submit it to the cloud, and have that workload be executed automatically in an autoscaled manner

Google Cloud Platform Provide autoscaling, fully managed, & serverless services.

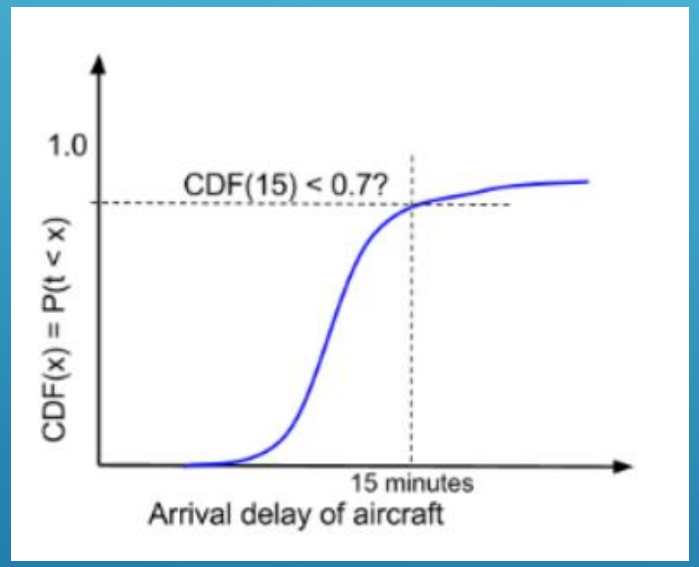
- BigQuery (for SQL analytics)
- Cloud Dataflow (for data processing pipelines)
- Google Cloud Pub/Sub (for message-driven systems)
- Google Cloud Bigtable (for high-throughput ingest)
- Google App Engine (for web applications)
- Cloud Machine Learning Engine (for machine learning).

Study Case:

Imagine that you are about to take a flight and just before the flight takes off from the runway (and you are asked to switch off your phone), you have the opportunity to send one last text message. It is past the published departure time and you are a bit anxious.

The reason for your anxiety is that you have scheduled an important meeting with a client at its offices.
You have taken the airline at its word with respect to when the flight would arrive, accounted for the time to hail a taxi, and used an online mapping tool to estimate the time to the client office.
Then, you added some leeway (say 30 minutes) and told the client what time you meet her.
And now, it turns out that the flight is departing late. So, should you send a text informing your client that you will not be able to make the meeting because your flight will be late or should you not?

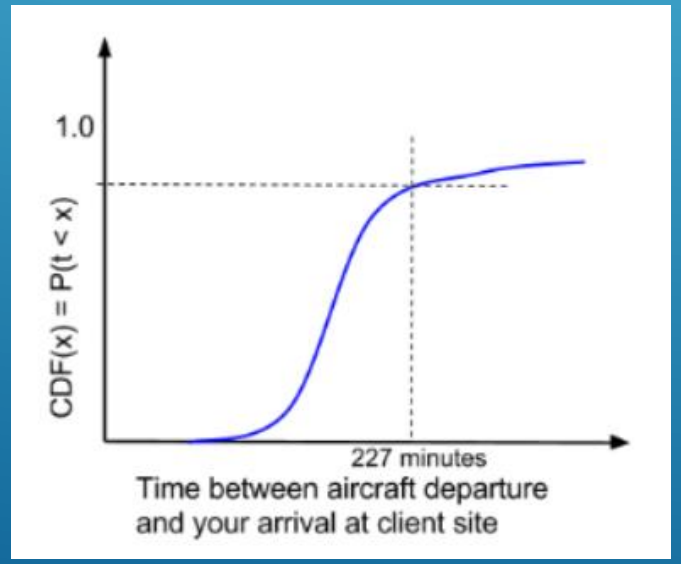
- This decision could be made in many ways, including by gut instinct and using heuristics.
- Being very rational people, we (you and I) will make this decision informed by data.
- Let's build a data framework to solve this problem.



Our decision criterion is to cancel the meeting if the Cumulative Distribution Function (CDF) of an arrival delay of 15 minutes is less than 70%.
Loosely speaking, we want to be 70% sure of the aircraft arriving no more than 15 minutes late.

Some other factor to consider:

- Although the airline company says that the flight is 127 minutes long and publishes an arrival time, not all flights are exactly 127 minutes long. If the plane happens to take off with the wind, catch a tail wind, and land against the wind, the flight might take only 90 minutes. Flights for which the winds are all precisely wrong might take 127 minutes (i.e., the airline might be publishing worst-case scenarios for the route).
- Google Maps is publishing predicted journey times based on historical data, and the actual journeys by taxi might be centered around those times.
- Your estimate of how long it takes to walk from the airport gate to the taxi stand might be predicated on landing at a specific gate, and actual times may vary.

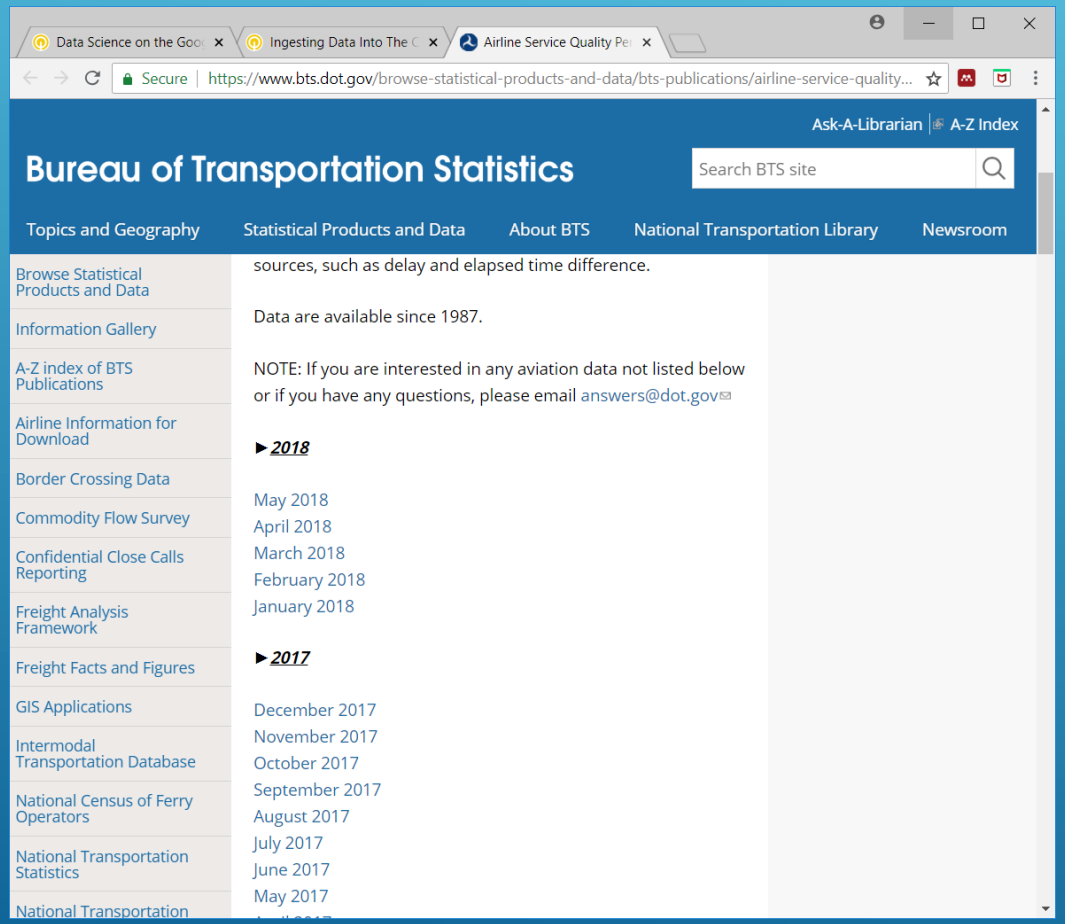


Because the time to get from the arrival airport to the client's office is unaffected by the flight departure delay, we can ignore it in our modeling.

We will use historical flight arrival data published by the **US Bureau of Transportation Statistics**, analyze it, and use it to inform our decision.

Qwiklab: Ingesting Data Into The Cloud

Download the on-time performance data from the BTS website as comma-separated value (CSV) files



Download Manually:
This is not the most helpful way to provide data for download.

- Data can be downloaded only one month at a time.
- We must select the fields that we want.
  - Imagine that you want to download all of the data for 2015. In that scenario, we select the fields we want for January 2015, submit the form, and then have to repeat the process for February 2015. If we forgot to select a field in February, that field would be missing, and we wouldn't know until we began analyzing the data

Solution: script the download to make life more easy and ensure consistensy

Fetch a sample data file using curl

For example, use the following `curl` command to fetch the data from January 2015:

```
curl
https://www.bts.dot.gov/sites/bts.dot.gov/files/docs/legacy/addi
attachment-files/ONTIME.TD.201501.REL02.04APR2015.zip --output
data.zip
```

Explore the downloaded data file to see what it looks like:

```
unzip data.zip
head ontime.td.201501.asc
```

You'll see something similar to the following:

```
AA|1|JFK|LAX|20150101|4|900|900|855|1230|1230|1237|0|0|...
AA|1|JFK|LAX|20150102|5|900|900|850|1230|1230|1211|0|0|...
```

This file doesn't include header information so it is not clear what each field is for, and it appears to contain a lot of data that isn't really needed, but it demonstrates one way to acquire a starting data set.

Download.sh file

```
YEAR=2015
for MONTH=`seq -w 1 12`; do
PARAMS="UserTableName...
RESPONSE=$(curl -X POST --data "$PARAMS"
http://www.transtats.bts.gov/DownLoad_Table.asp?Table_ID=236&Has_Group=3&Is_
Zipped=0)
echo "Received $RESPONSE"
ZIPFILE=$(echo $RESPONSE | tr '\"' '\n' | grep zip)
echo $ZIPFILE
curl -o $YEAR$MONTH.zip $ZIPFILE
done
```

Fetch using script file

```
bash ../02_ingest/download.sh
```

It will take about five minutes to fetch all twelve source data files.

Confirm that you have all of the files by entering the following command:

```
ls -ltr
```

Now rename the source files to make the names more human-frien

```
for month in `seq -w 1 12`; do
   unzip 2015$month.zip
   mv *ONTIME.csv 2015$month.csv
   rm 2015$month.zip
done
```

Check that the data looks correct:

```
head 201503.csv
```

The output should looks similar to the following:

```
"FL_DATE","UNIQUE_CARRIER","AIRLINE_ID","CARRIER","FL_NUM","ORI

TAXI_OUT","WHEELS_OFF","WHEELS_ON","TAXI_IN","CRS_ARR_TIME","AR

2015-03-
01,"B6",20409,"B6","1170",13204,1320402,31454,"MCO",14524,14524

2015-03-
01,"B6",20409,"B6","1171",12953,1295302,31703,"LGA",11697,11697
```

command for upload the files to Cloud Storage

Make a storage bucket using the `project-id` as a suffix and copy all of the data files to the storage bucket:

```
export PROJECT_ID=$(gcloud info --
format='value(config.project)')
gsutil mb gs://${PROJECT_ID}-ml
gsutil -m cp *.csv gs://${PROJECT_ID}-ml/flights/raw/
```

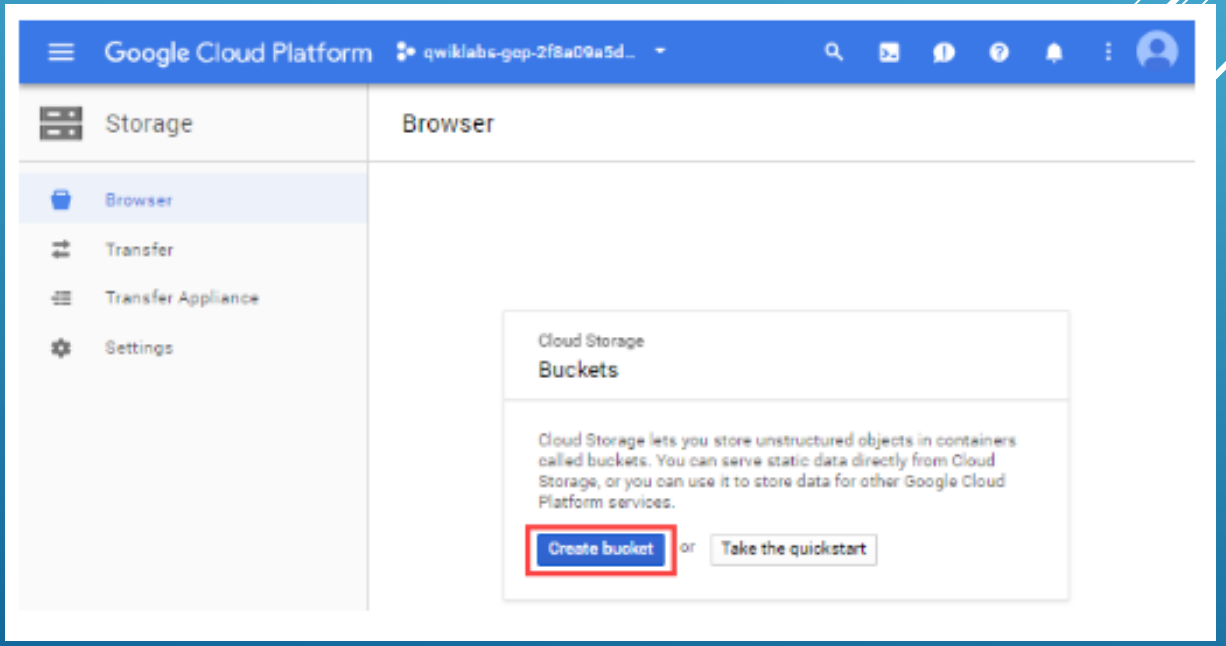
Clean up the data by removing the trailing comma and quotation marks using `sed`:

```
for month in `seq -w 1 12`; do
    sed 's/,$//g' 2015$month.csv | sed 's/"//g' > tmp
    mv tmp 2015$month.csv
done
```

Result :

```
2015-03-01,UA,19977,UA,1025,14771,1477101,32457,SFO,11618,1161802,31703,EWR,
0637,0644,7.00,15.00,0659,1428,12.00,1450,1440,-10.00,0.00,,0.00,2565.00
```

Another option to make storage bucket : In the Console, go to **Products & services** > **Storage**, then click **Create Bucket**.



Qwiklab: Ingesting Data Into The Cloud Using Google Cloud App Engine

Now that we have some historical flight data in our Cloud Storage bucket.
The question now how to keep the bucket current ? After all, airlines didn't stop flying in 2015, and the BTS continues to refresh its website on a monthly basis.

Solution:
Scheduling Monthly Downloads
It keep ourselves synchronized with the BTS.

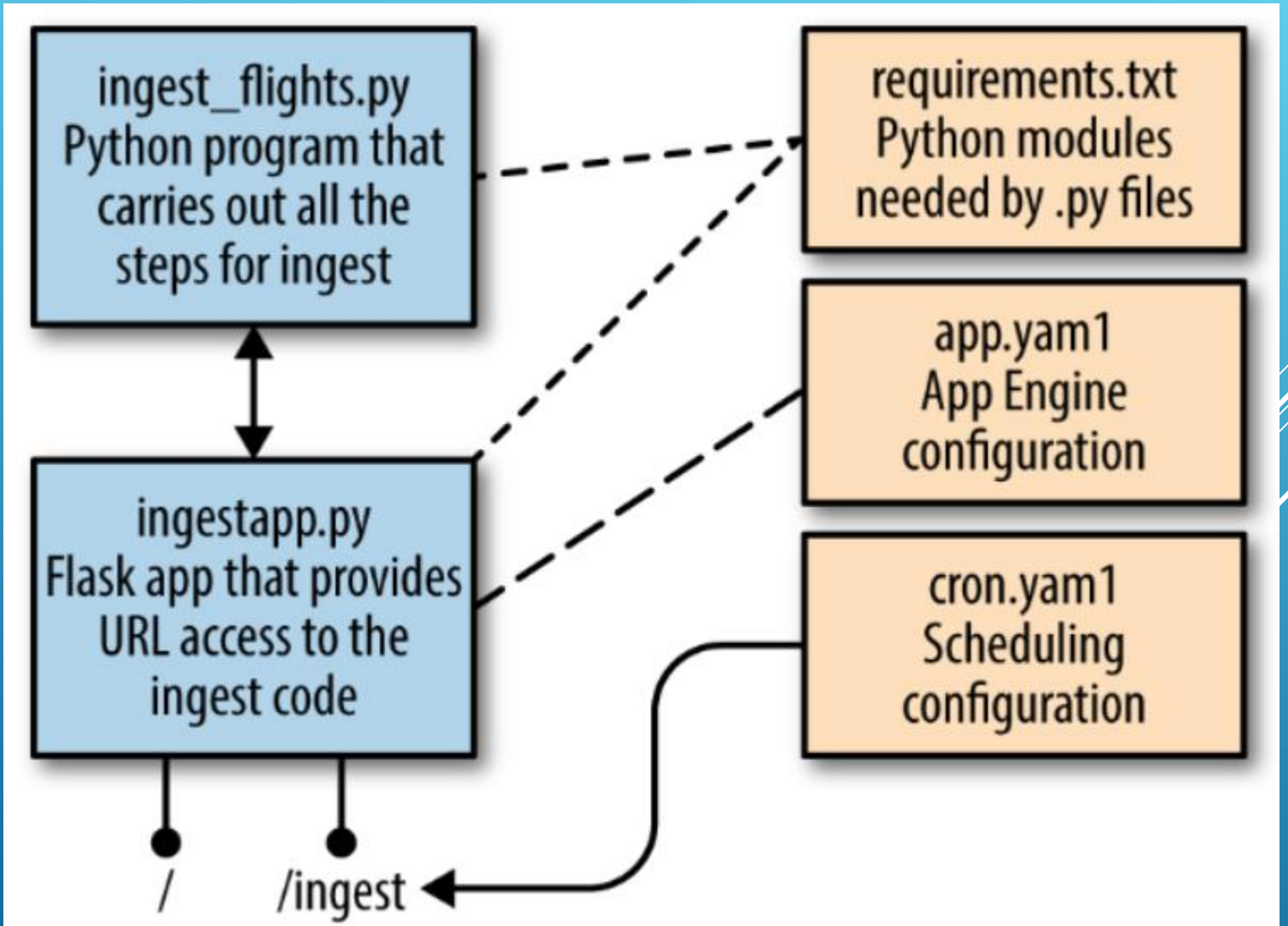


Figure 2-14. The architecture of the monthly ingest job in App Engine

- a standalone ingest_flights.py application that is capable of downloading the data for a specific year/month and uploading the data to Cloud Storage.

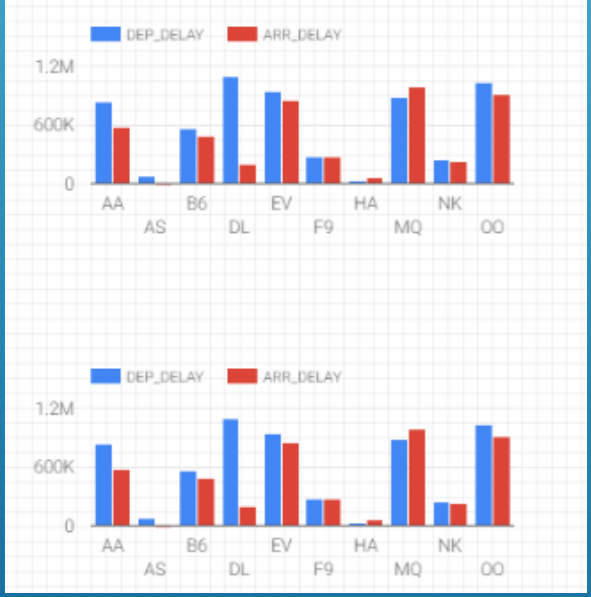
- The way scheduling works in App Engine is that we must specify a URL that will be invoked. App Engine will visit the URL according to the schedule that we specify. Because our ingest code is a standalone Python program, we will wrap that ingest code into a Flask web application (ingestapp.py) so that we can invoke it by using a URL.

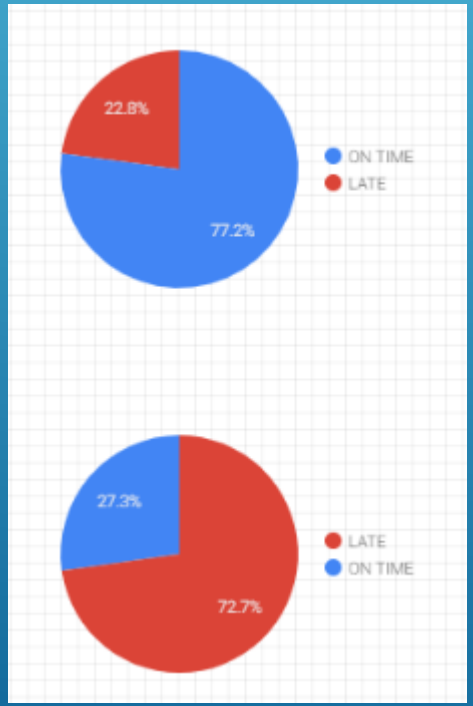
- The Flex version of App Engine requires us to specify our dependencies so that it can create an appropriate Dockerfile; we do this in requirements.txt.
- The App Engine configuration (URL mapping, security, etc.) goes in app.yaml.
- The schedule itself goes in cron.yaml.
- The entire package is deployed into App Engine, which publishes the application online and then manages everything.

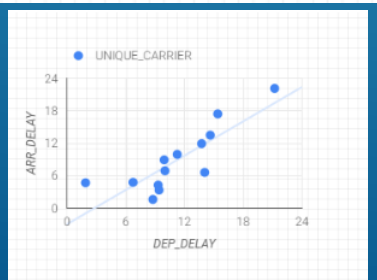
Qwiklab: Loading Data into Google Cloud SQL & Visualizing Data with Google Data Studio

Objective in this quest:

- Build an initial **data model** using queries
- Create Cloud SQL database views
- Create a Cloud SQL Datasource in Google Data Studio
- Create a Data Studio report with a date range control
- Create multiple charts using Cloud SQL database views








```
select count(dest) from flights where arr_delay < 15 and
dep_delay < 15;
select count(dest) from flights where arr_delay >= 15 and
dep_delay < 15;
select count(dest) from flights where arr_delay < 15 and
dep_delay >= 15;
select count(dest) from flights where arr_delay >= 15 and
dep_delay >= 15;
```

This will provide the following totals:

- True Negative : 672038
- False Negative: 44855
- False Positive: 35991
- True Positive: 146275

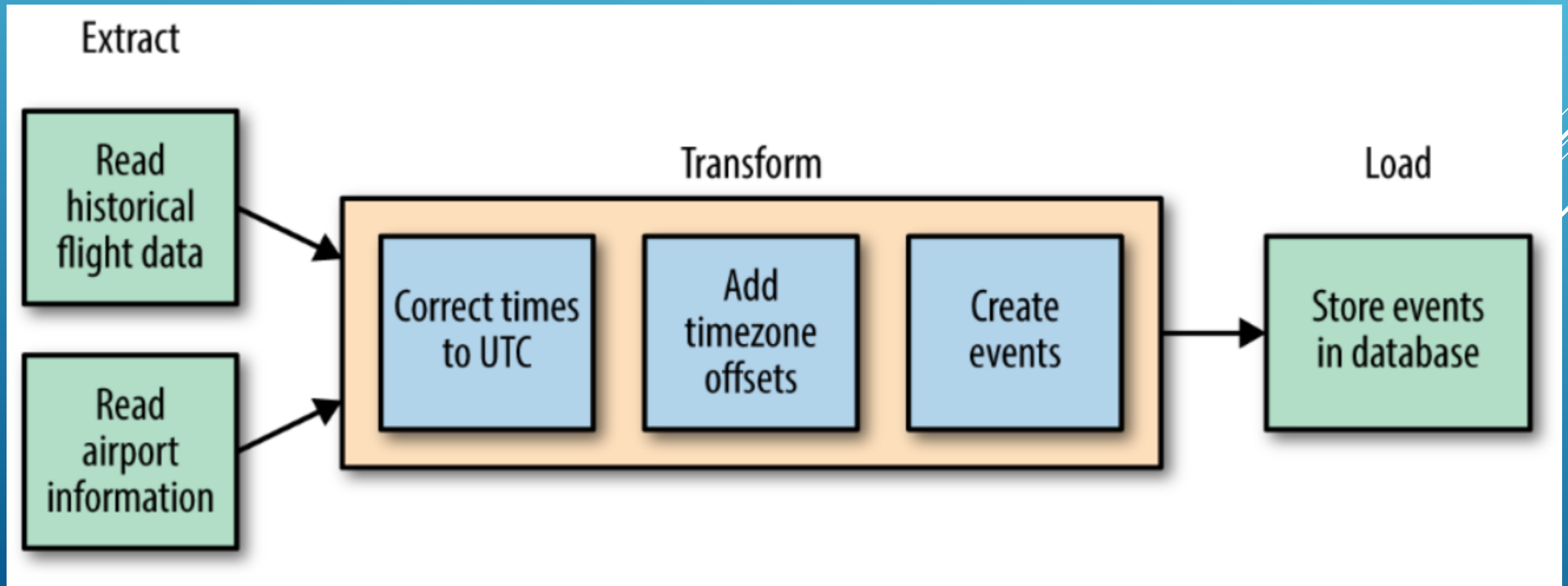
Where the arrival delay is less than 15 minutes and departure delay was also less than 15 minutes. This is a true negative.
Where the arrival delay is greater than 15 minutes and departure delay was less than 15 minutes. This is a false negative.
Where the arrival delay is less than 15 minutes and departure delay was greater than 15 minutes. This is a false positive.
Where the arrival delay is greater than 15 minutes and the departure delay is also greater than 15 minutes. This is a true positive.

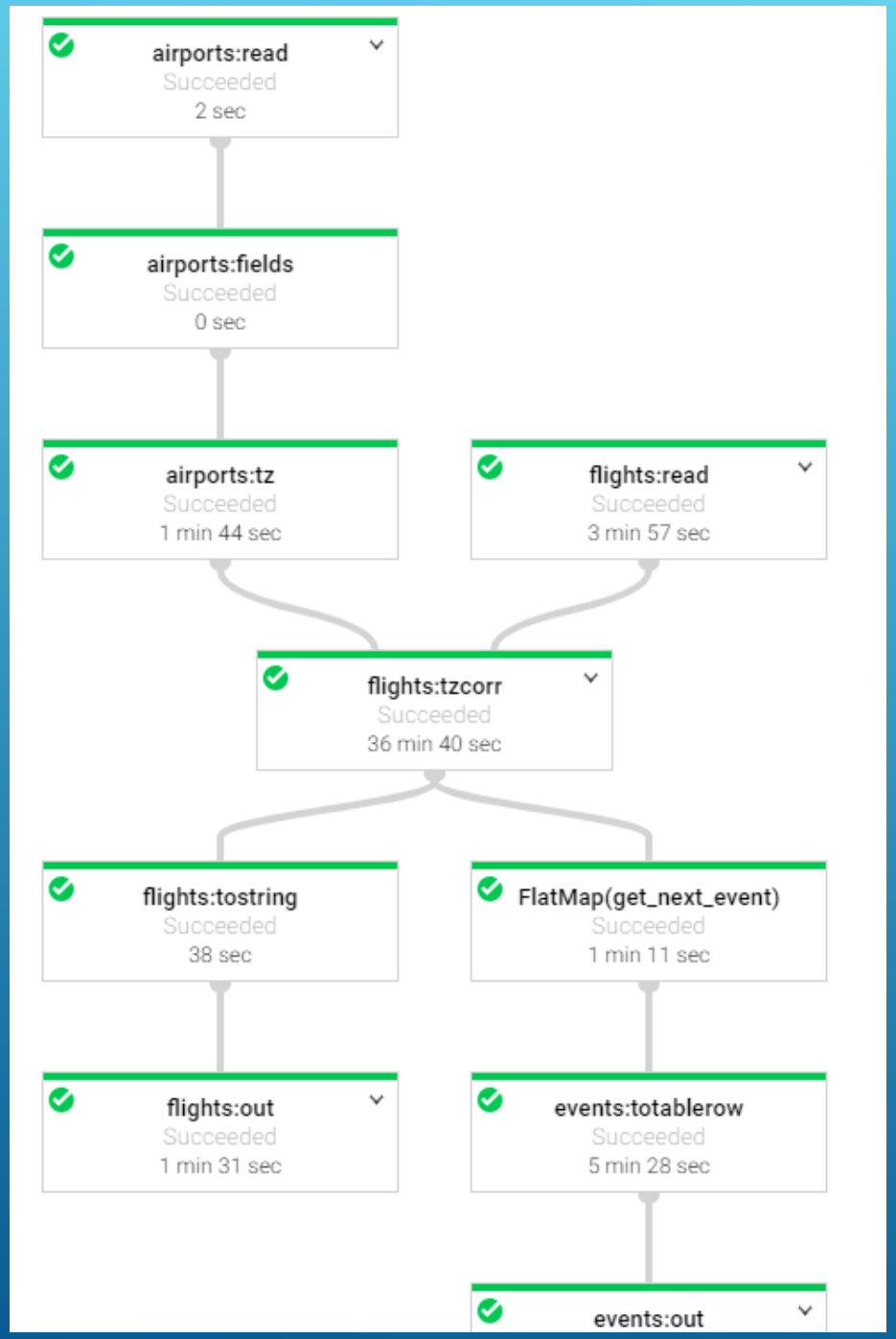
- In this part, writer discussed the importance of bringing the insights of our end users into our data modeling.
- We populated our dashboard from Cloud SQL, a transactional, relational database whose management is simplified by virtue of it running on the cloud and being managed by Google Cloud Platform.
- A dashboard is an end-user report that is interactive, tailored to end users, and continually refreshed with new data.
- The data model that we built was to suggest that our road warriors cancel their immediately scheduled meeting if the departure delay of the flight was more than 10 minutes. This would enable them to make 70% of their meetings with 15 minutes to spare.
- Built a dashboard in Data Studio to explain the contingency table model.
- The purpose of this step in the modeling process is not simply to depict the data, but to improve user's understanding of why the model is the way it is.
- Whenever we are designing the display of a dataset, evaluate the design in terms of three aspects:
    - Does it accurately and honestly depict the data?
    - How well does it help envision not just the raw data, but the information content embodied in the data?
    - Is it constructed in such a way that it explains the model being used to provide recommendations?

Qwiklab: Processing Data with Google Cloud Dataflow

What will we do in this qwiklab ?

- We simulate process on how to build a real-time analysis pipeline to carry out streaming analytics and populate real-time dashboards.
- Remember in this case the dataset we are using is not available in real time.
- In the process of building out the simulation, we realized that time handling in the original dataset was problematic. Therefore, we improved the handling of time in the original data and created a new dataset with UTC timestamps and local offsets.





1. Parsing Airports Data to get time zone for each airport
2. Takes the fields as input and outputs a tuple of the (1000401, (68.08333333,-163.16666667)).
   The first number is the unique airport code and the next two numbers are the latitude/longitude pair for the airport's location
3. join the flights data with the airports data to find the time zone corresponding to each flight.
4. After we have our time-corrected data, we can move on to creating events.
5. As a final touch, we store the time-corrected flight data as CSV files in Cloud Storage but store the events in BigQuery.
   BigQuery is Google Cloud Platform's data warehouse that supports SQL queries and makes it easier if you want to pull out a subset of events to simulate.

Qwiklab: Visualize Real Time Geospatial Data with Google Data Studio

What wil we do in this qwiklab ?

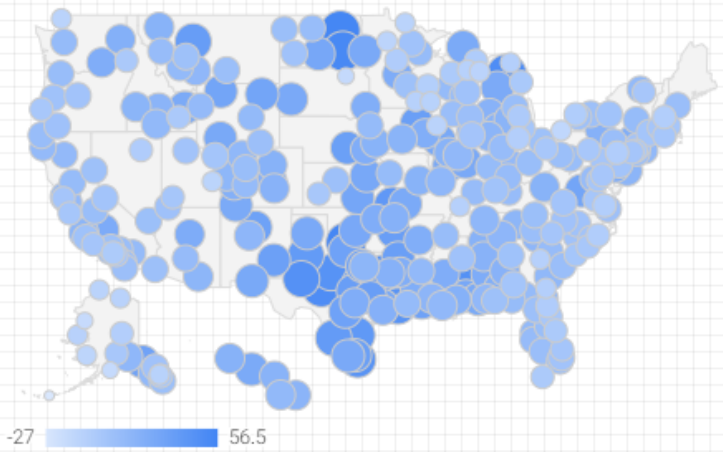
- Now that we have the source events from the raw flight data so we will Create a Google Dataflow processing job for streaming data

- Generate real-time streaming data using Python.
  - Streaming data in Google Cloud Platform is typically published to Cloud Pub/Sub, a serverless real-time messaging service. Cloud Pub/Sub provides reliable delivery and can scale to more than a million messages per second

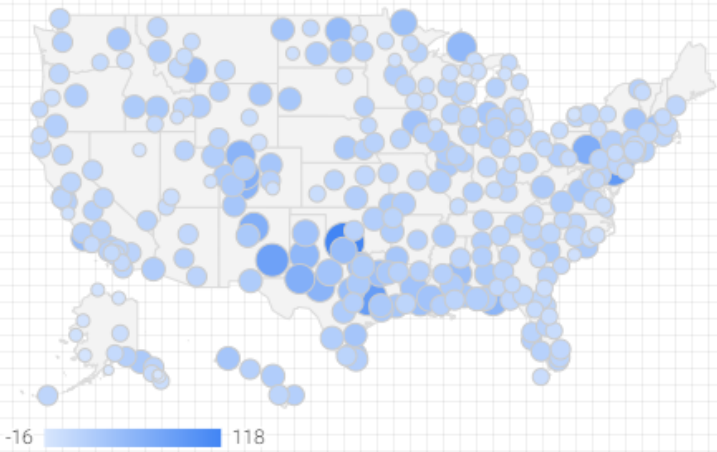
- Analyze streaming data in Google BigQuery
  - Queries on streaming data will be useful when we begin to build our dashboard. For example, query will allow us to build a map of average delays across the country.

- Create a real-time geospatial dashboard for streaming data
  - Now that we have streaming data in BigQuery and a way to analyze it as it is streaming in, we can create a dashboard that shows departure and arrival delays in context.
  - To pull the data to populate these charts, we need to add a BigQuery data source in Data Studio.

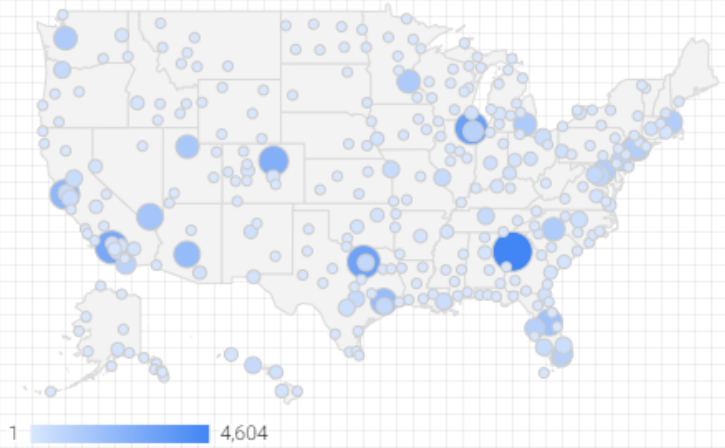
Geospatial dashboard



Arrival Delay



Departure Delay



Total Number of Flights

Qwiklab: Visualize Real Time Geospatial Data with Google Data Studio

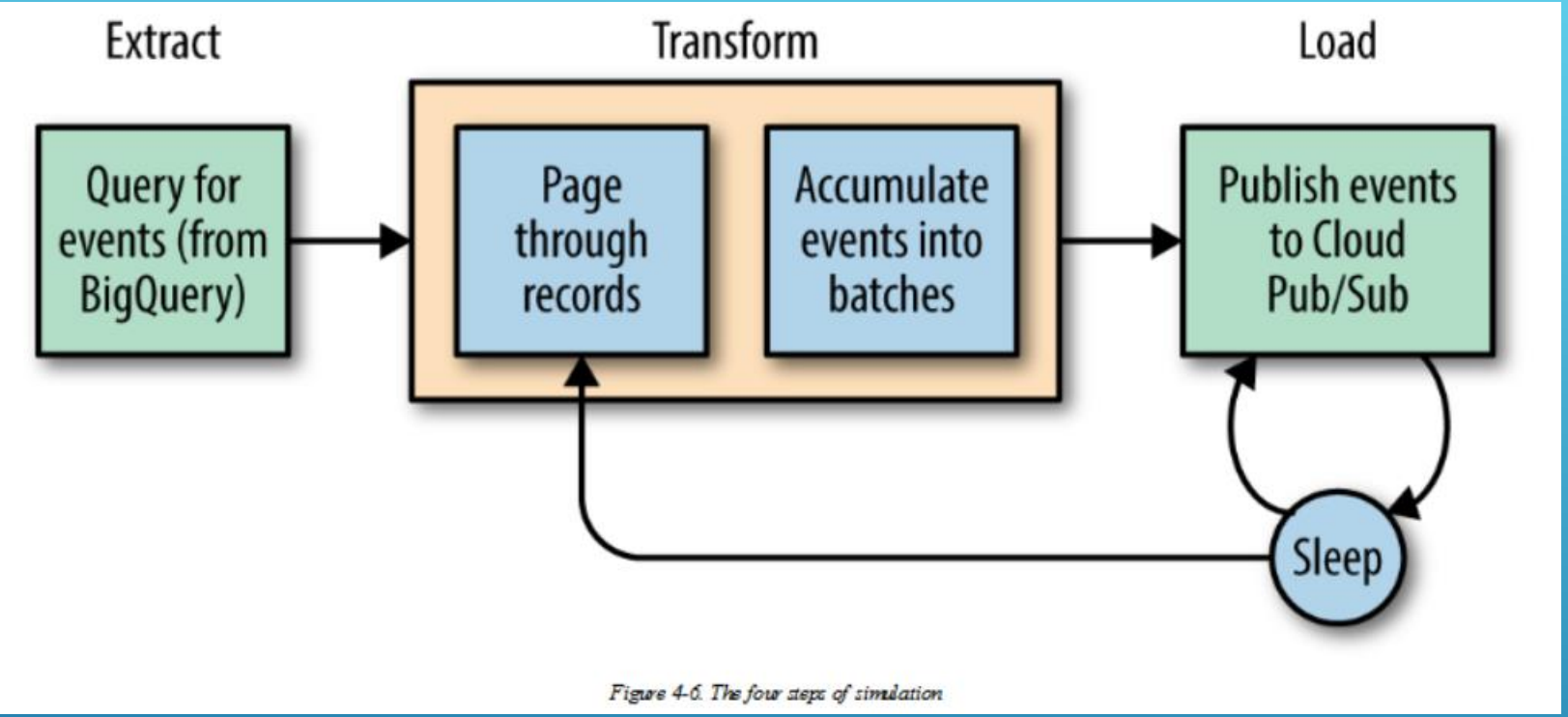
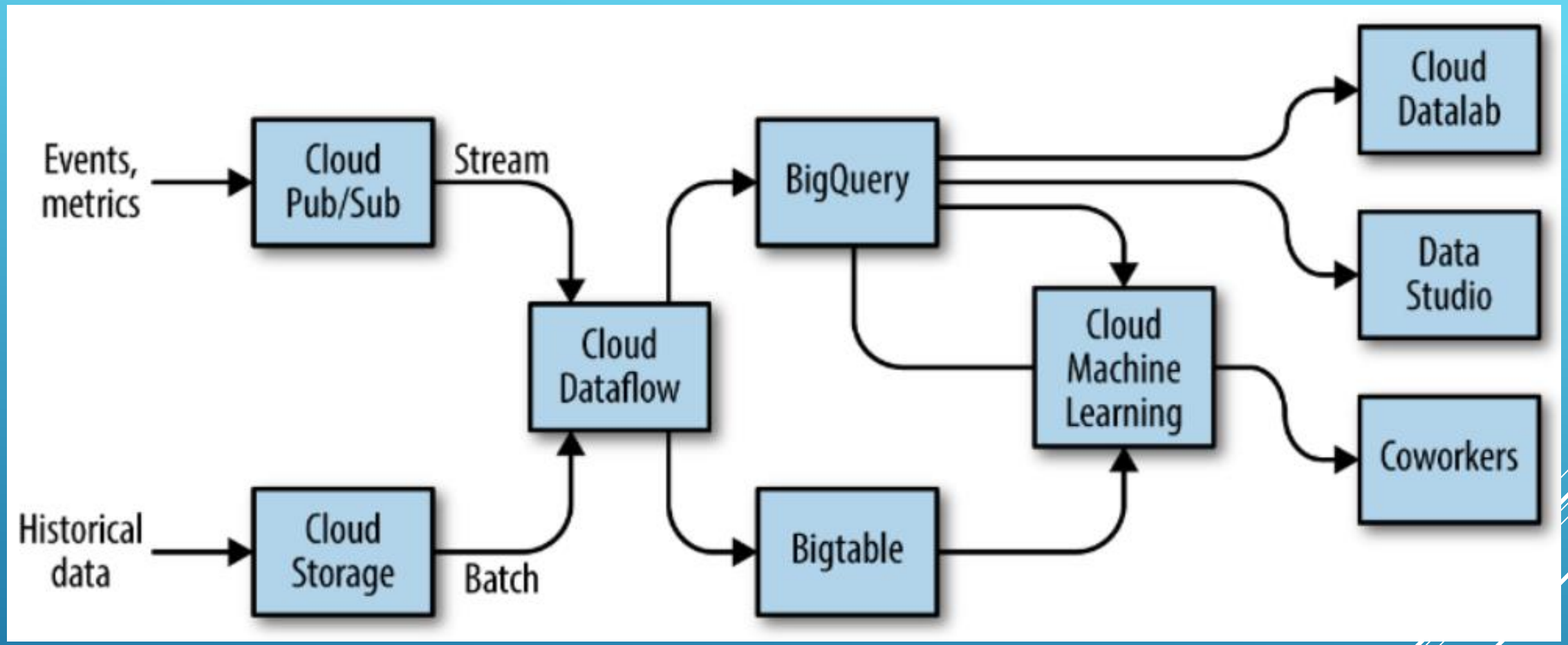


Figure 4-6. The four steps of simulation

1. Run the query to get the set of flight event records to publish.
2. Page through the query.
3. Accumulate events to publish as a batch.
4. Publish accumulated events and sleep as necessary.

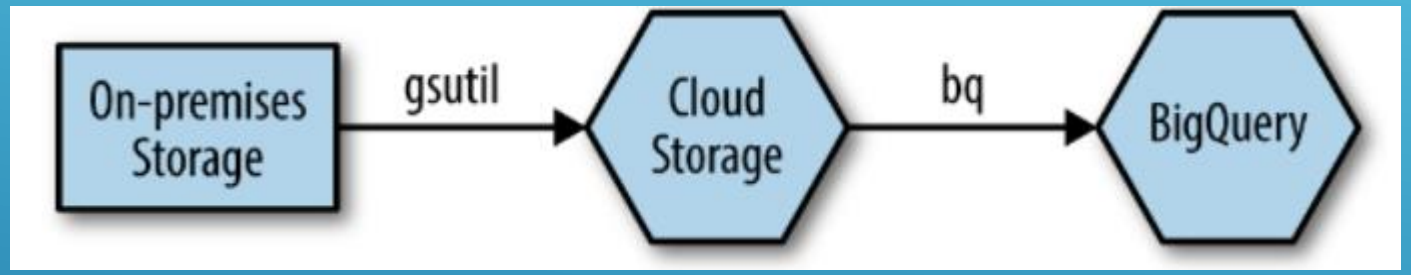
- Our simulator will read from the events table in BigQuery (populated in the previous section) and publish messages to Cloud Pub/Sub based on a mapping between the event notification time (arrival or departure time based on event)
- Paging Through Records
  - As we page through the query results, we need to publish to Cloud Pub/Sub. There is a separate topic per event type (i.e., an arrived topic and a departed topic)



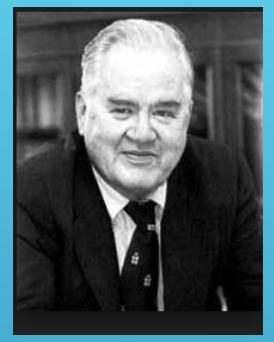
Reference architecture for data processing on Google Cloud Platform

Qwiklab: Loading Data into Google BigQuery for Exploratory Data Analysis

To ingest data (especially larger files) from outside Google Cloud Platform to BigQuery, it is preferable to first load it into Cloud Storage and use Cloud Storage as the staging ground for BigQuery.



* gsutil takes advantage of multithreaded, resumable uploads and is better suited to the public internet.



Developed by John W. Tukey

1. Identify important variables
2. Discover underlying structure
3. Develop parsimonious models, and use that model to identify unusual patterns and values.

The aim of dashboard creation I n previous Qwiklabs is to crowd-source insight into the working of models from end users and is, therefore, primarily about presenting an explanation of the models to **end users.**

The aim of EDA is for the **data engineer**, to develop insights about the data before we delve into developing sophisticated models. The audience for EDA is typically other members of your team, not end users.

Qwiklab: Loading Data into Google BigQuery for Exploratory Data Analysis

Create a new Google Cloud Datalab instance

Cloud Datalab provides a hosted version of Jupyter on Google Cloud Platform; it knows how to authenticate against Google Cloud Platform so as to provide easy access to Cloud Storage, BigQuery, Cloud Dataflow, Cloud ML Engine, and so on

Jupyter Notebooks
Jupyter is open source software that provides an interactive scientific computing experience in a variety of languages including Python, R, and Scala. The key unit of work is a Jupyter Notebook, which is a web application that serves documents with code, visualizations, and explanatory text. Jupyter also supports the creation of interactive widgets for manipulating and visualizing data.

!pwd
you will get the folder in which that notebook appears

!pip freeze
This lists the Python packages installed.

!pip install google-cloud
Installed package in Cloud Datalab

```
import matplotlib.pyplot as plt
import seaborn as sb
import pandas as pd
import numpy as np
```

- Numpy is the canonical numerical Python library that provides efficient ways of working with arrays of numbers.
- Pandas is an extremely popular data analysis library that provides a way to do operations such as group by and filter on in-memory dataframes.
- Matplotlib is a Matlab-inspired module to create graphs in Python.
- Seaborn provides extra graphics capabilities built on top of the basic functionality provided by matplotlib. All these are open source packages that are installed in Cloud Datalab by default.

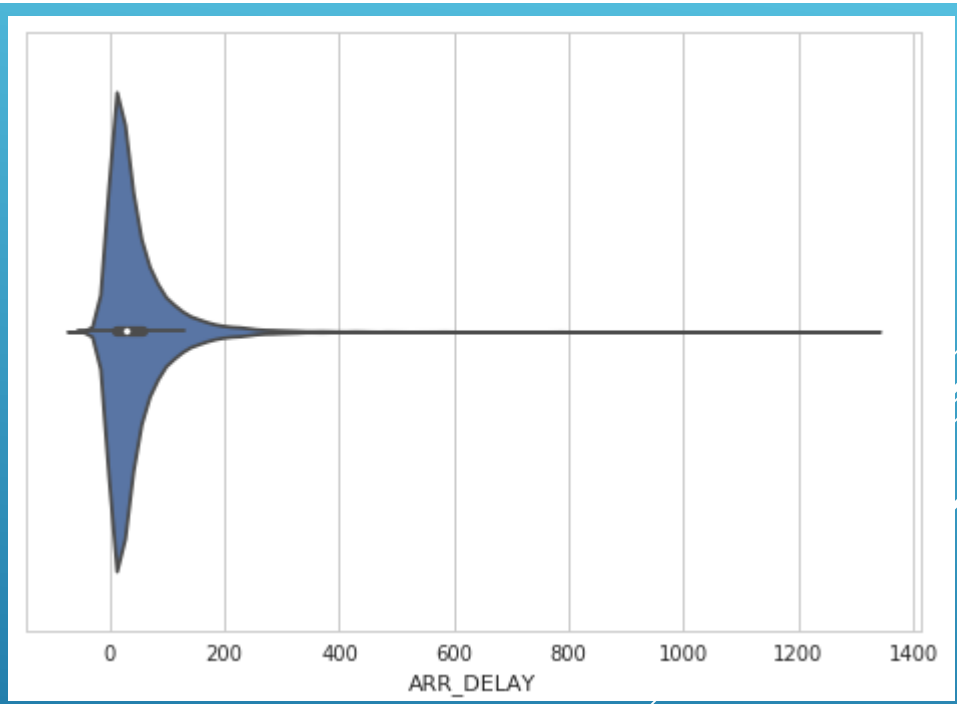
Qwiklabs: Exploratory Data Analysis Using Google Cloud Datalab

```
sql = """
SELECT ARR_DELAY, DEP_DELAY
FROM `flights.tzcorr`
WHERE DEP_DELAY >= 10 AND RAND() < 0.01
"""
df = client.query(sql).to_dataframe()
df.describe()
```

select a random sample of flights that departed 10 or more minutes late.

	ARR_DELAY	DEP_DELAY
count	12898.000000	12973.000000
mean	45.907815	50.620134
std	62.639830	60.068364
min	-60.000000	10.000000
25%	11.000000	17.000000
50%	27.000000	30.000000
75%	58.000000	60.000000
max	903.000000	892.000000

```
sns.set_style("whitegrid")
sns.set(font="DejaVu Sans")
ax = sns.violinplot(data=df, x='ARR_DELAY', inner='box',
orient='h')
```



Graphs the distribution of arrival delay values

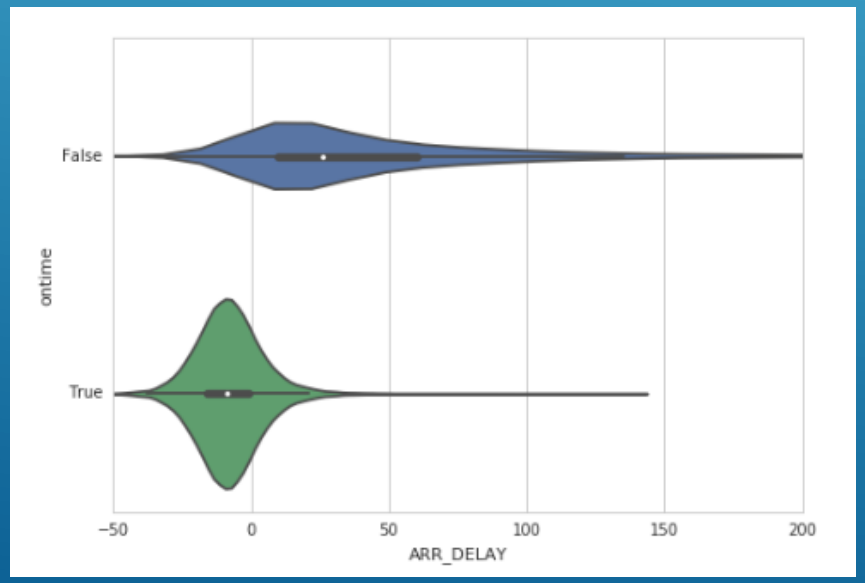
- The peak in the number of flights that arrive late happens around the 10 minute mark.
- flights arriving ahead of time are relatively rare.

Qwiklabs: Exploratory Data Analysis Using Google Cloud Datalab

Difference between flights that depart more than 10 minutes late and those that depart less than 10 minutes late

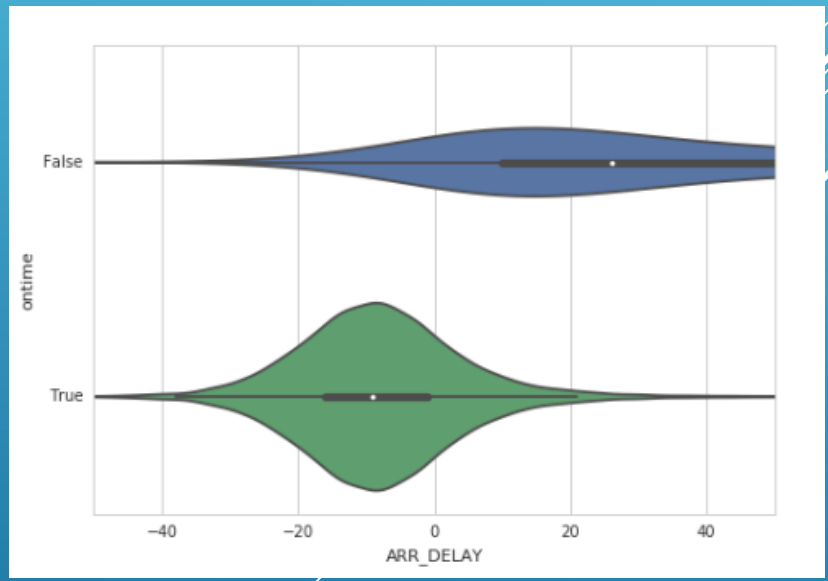
```
sql = """
SELECT ARR_DELAY, DEP_DELAY
FROM `flights.tzcorr`
WHERE RAND() < 0.001
"""
df = client.query(sql).to_dataframe()
df['ontime'] = df['DEP_DELAY'] < 10
ax = sns.violinplot(data=df, x='ARR_DELAY', y='ontime',
                    inner='box', orient='h')
ax.set_xlim(-50, 200);
```

The top plot is clipped because Seaborn's default smoothing parameter is too coarse. In order to fix this we narrow down the x axis limits.




```
sql = """
SELECT ARR_DELAY, DEP_DELAY
FROM `flights.tzcorr`
WHERE RAND() < 0.001
"""
df = client.query(sql).to_dataframe()
df['ontime'] = df['DEP_DELAY'] < 10
ax = sns.violinplot(data=df, x='ARR_DELAY', y='ontime',
                    inner='box', orient='h', gridsize=1000)
ax.set_xlim(-50, 50);
```

Flights that depart 10 or more minutes late are clearly asymmetrically distributed towards longer delay times. Flights that depart less than 10 minutes late are centered around the peak arrival time



Qwiklabs: Exploratory Data Analysis Using Google Cloud Datalab

```
#standardsql
SELECT
  DEP_DELAY,
  AVG(ARR_DELAY) AS arrival_delay,
  COUNT(ARR_DELAY) AS numflights
FROM
  `flights.tzcorr`
GROUP BY
  DEP_DELAY
ORDER BY
  DEP_DELAY
```

Results | Details

Row	DEP_DELAY	arrival_delay	numflights
1	null	null	0
2	-82.0	-80.0	1
3	-68.0	-87.0	1
4	-61.0	-77.0	1
5	-56.0	-26.0	1
6	-55.0	-60.0	1
7	-52.0	-61.0	1
8	-48.0	-34.5	2
9	-47.0	-51.0	1
10	-46.0	-7.0	1
11	-45.0	-34.0	4
12	-44.0	-33.0	1
13	-43.0	-39.0	2
14	-42.0	-34.8	5

Results | Details

Row	DEP_DELAY	arrival_delay	numflights
52	-4.0	-9.250346462024526	443050
53	-3.0	-8.511708563393041	454411
54	-2.0	-7.6017360583323335	434202
55	-1.0	-6.499271693319495	386513
56	0.0	-5.100334305600405	328442
57	1.0	-4.1882858556938665	159619
58	2.0	-3.2246696399074937	121080
59	3.0	-2.195782178407907	104177
60	4.0	-1.21018607307166	92813
61	5.0	-0.16180371352785197	84448
62	6.0	0.789009220541097	75809

The first few rows, which show the extreme values for negative departure delays, i.e. flights leaving well ahead of schedule, have very few flights each.

It's pretty clear that the extreme values represent such a small proportion of the data that they can be ignored. So we will remove this small portion of data

Qwiklabs: Evaluating Data Model

Query string that captures the contingency table that allows to score the effectiveness of the model:

```
evalquery = """
SELECT
  SUM(IF(DEP_DELAY < 16
      AND arr_delay < 15, 1, 0)) AS correct_nocancel,
  SUM(IF(DEP_DELAY < 16
      AND arr_delay >= 15, 1, 0)) AS wrong_nocancel,
  SUM(IF(DEP_DELAY >= 16
      AND arr_delay < 15, 1, 0)) AS wrong_cancel,
  SUM(IF(DEP_DELAY >= 16
      AND arr_delay >= 15, 1, 0)) AS correct_cancel
FROM (
  SELECT
    DEP_DELAY,
    ARR_DELAY
  FROM
    `flights.tzcorr` f
  JOIN
    `flights.trainday` t
  ON
    f.FL_DATE = t.FL_DATE
  WHERE
    t.is_train_day = 'False' )
"""
eval = client.query(evalquery).to_dataframe()
eval
```

	correct_nocancel	wrong_nocancel	wrong_cancel	correct_cancel
0	1259740	66081	52827	217669

Python code to display the ratio of correct to incorrect calls for the full evaluation data set.

```
print eval['correct_nocancel'] / (eval['correct_nocancel'] +
\
eval['wrong_nocancel'])
print eval['correct_cancel'] / (eval['correct_cancel'] + \
eval['wrong_cancel'])
```

```
0    0.950158
dtype: float64
0    0.804703
dtype: float64
```

- 95% of the recommendations to not cancel meetings are correct and 80% of the recommendations to cancel the meetings are correct.
- This is higher than the model which is intended to cancel flights where the probability of an arrival delay is greater than 30%.
- The difference arises because the evaluation data set includes the full distribution across all possible delay values, including the outliers and flights with non-outlier data, such as flights leaving 20 minutes late, where the probability of the arrival delay is much greater than 30%.

Qwiklabs: Evaluating Data Model

Query for flights where DEP_DELAY is 15 minutes, and flights are not cancelled or 16 minutes and flights are cancelled:

```
evalquery2="""
SELECT
  SUM(IF(DEP_DELAY = 15
      AND arr_delay < 15, 1, 0)) AS correct_nocancel,
  SUM(IF(DEP_DELAY = 15
      AND arr_delay >= 15, 1, 0)) AS wrong_nocancel,
  SUM(IF(DEP_DELAY = 16
      AND arr_delay < 15, 1, 0)) AS wrong_cancel,
  SUM(IF(DEP_DELAY = 16
      AND arr_delay >= 15, 1, 0)) AS correct_cancel
FROM (
  SELECT
    DEP_DELAY,
    ARR_DELAY
  FROM
    `flights.tzcorr` f
  JOIN
    `flights.trainday` t
  ON
    f.FL_DATE = t.FL_DATE
  WHERE
    t.is_train_day = 'False' )
"""
eval = client.query(evalquery2).to_dataframe()
eval
```

	correct_nocancel	wrong_nocancel	wrong_cancel	correct_cancel
0	7684	2935	6787	2942

Python code to display the ratio of correct to incorrect calls.

```
print eval['correct_nocancel'] / (eval['correct_nocancel'] +
\
eval['wrong_nocancel'])
print eval['correct_cancel'] / (eval['correct_cancel'] + \
eval['wrong_cancel'])
```

```
0    0.723609
dtype: float64
0    0.302395
dtype: float64
```

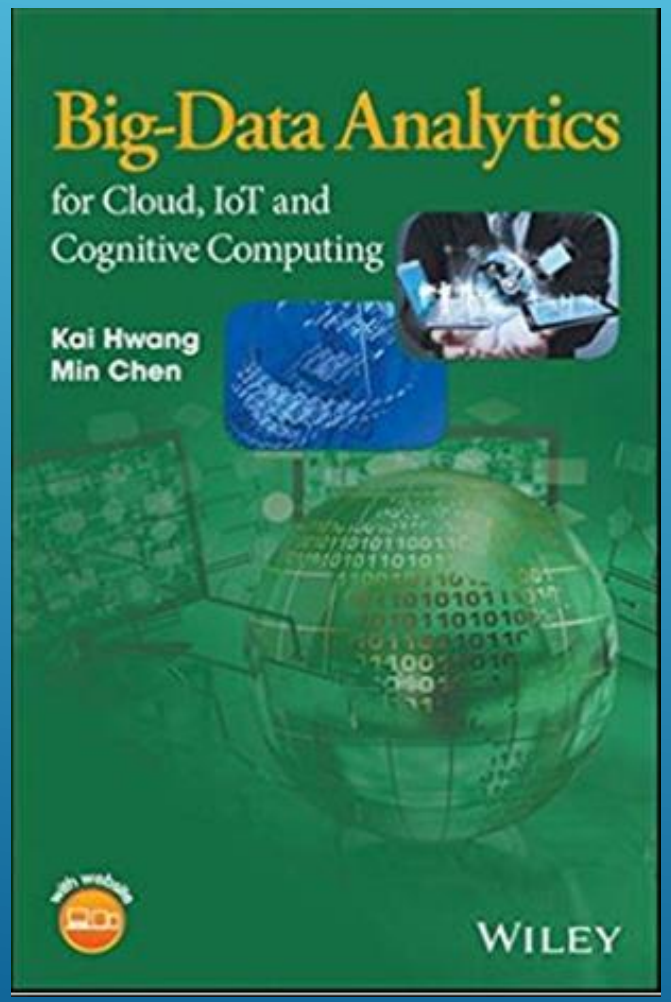
- 72% of the recommendations to not cancel meetings, the first number, are correct and 30% of the recommendations to cancel the meetings are correct.
- The correct cancellation call is made 30.2% of the time which is very close to the target of 30%

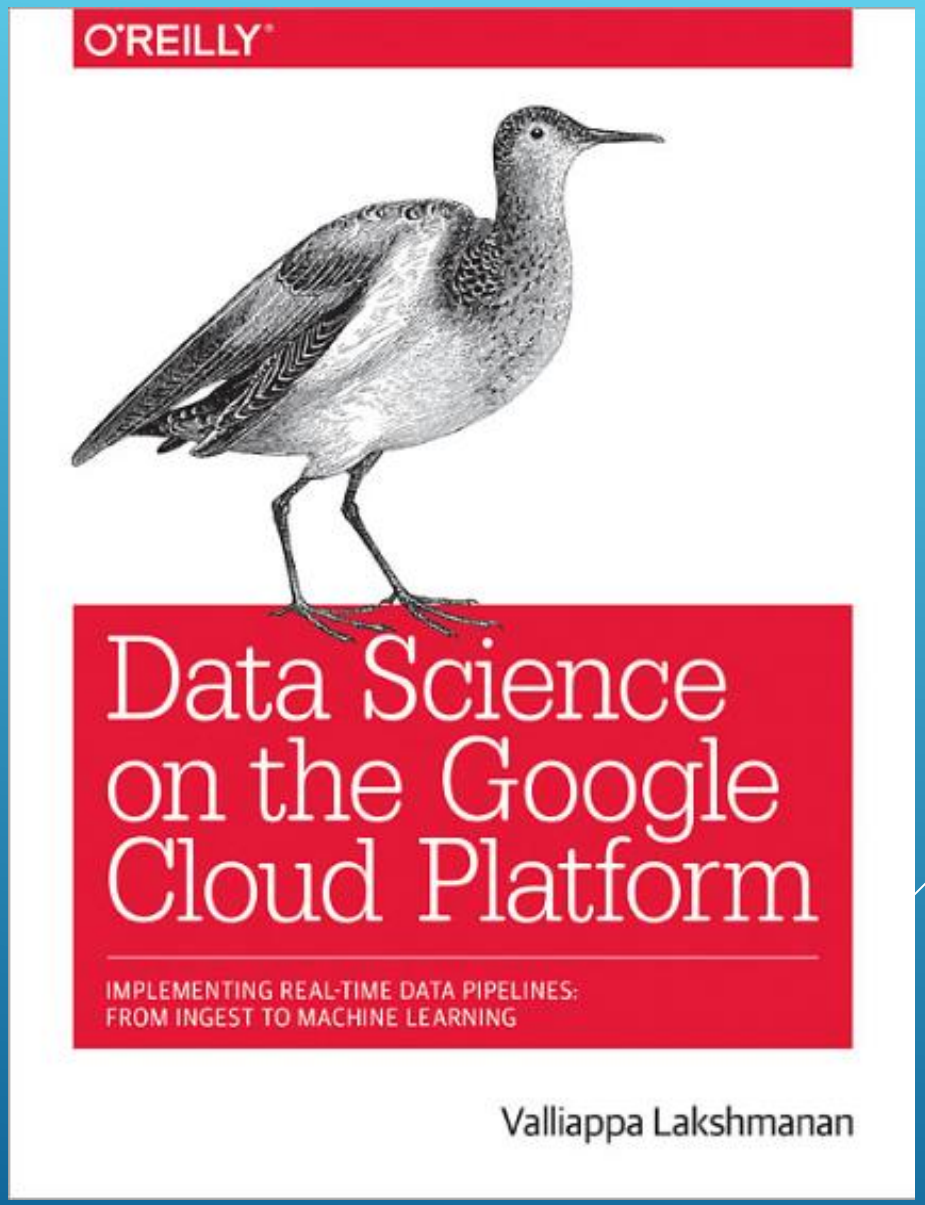
What we have learn from this quest:

1. How to Ingest data into Cloud
2. How to Ingest data into Cloud using App Engine
3. Build Dashboard or Data Visualization
4. Process Streaming Data with Dataflow
5. Google BigQuery
6. Datalab with Jupyter notebook

Conclusion :
Google Cloud Platform, though, is designed to allow you to forget about infrastructure, the practice of data science has become easier thanks to the advent of serverless data processing and machine learning systems that are integrated into powerful statistical and visualization software tools. (Valliappa Lakshmanan 2017)

REFERENCE


Big-Data Analytics
for Cloud, IoT and
Cognitive Computing
Kai Hwang
Min Chen
WILEY


O'REILLY
Data Science
on the Google
Cloud Platform
IMPLEMENTING REAL-TIME DATA PIPELINES:
FROM INGEST TO MACHINE LEARNING
Valliappa Lakshmanan

Picture Source : https://sd.keepcalm-o-matic.co.uk/i/thank-you-for-your-attention-any-questions-78.png