Project Big Data, Business Intelligence and NoSQL   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Teamleden:

Thomas Billiet

Ishan Ameel

Sven Depickere

Jonas Anseel

Inhoud

[1. Preprocessing van de dataset 2](#_Toc513900177)

[1.1. Data laden 2](#_Toc513900178)

[1.2. Opkuis functies 3](#_Toc513900179)

[1.3 Converteer de militaire tijd 5](#_Toc513900180)

[1.4 Voeg tijdzones toe 6](#_Toc513900181)

[1.5 Zet alle tijden op UTC 7](#_Toc513900182)

[1.6 Bereken de vliegtijd 7](#_Toc513900183)

[1.7 Bereken de afstand 8](#_Toc513900184)

[1.8 Bereken de snelheid 8](#_Toc513900185)

[1.9 Verwijder te korte en te lange vluchten 8](#_Toc513900186)

[1.10 Uitvoeren van de opkuis-functies 9](#_Toc513900187)

[2. Tableau 10](#_Toc513900188)

[2.1 Algemene informatie over de vliegtuig maatschappijen 11](#_Toc513900189)

[3. Machine learning met Scikit learn 37](#_Toc513900190)

[4. NetworkX 38](#_Toc513900191)

[4.1 Laad de dataset 38](#_Toc513900192)

[4.2 Routes tussen luchthavens 38](#_Toc513900193)

[4.3 Grafiek van alle luchthavens en hun routes 39](#_Toc513900194)

[4.4 Vind de coördinaten van elke luchthaven 39](#_Toc513900195)

[4.5 Plaats de routes op kaart 40](#_Toc513900196)

# Preprocessing van de dataset

We hebben onze code in verschillende stappen opgesplitst namelijk:

* Data laden
* Opkuis functies
* Converteer de militaire tijd
* Voeg tijdzones toe
* Zet alle tijden op UTC
* Bereken de vliegtijd
* Bereken de afstand
* Bereken de snelheid
* Verwijder te korte en te lange vluchten
* Uitvoeren van de opkuis-functies

## Data laden

1. IMPORT\_PATH = '../Initial\_Data/Unzipped'
2. IMPORT\_ALL\_PATH = IMPORT\_PATH + '/\*'
4. import pandas as pd
5. import numpy as np
6. import glob as glob
8. def readAllFiles():
9. files = glob.glob(IMPORT\_ALL\_PATH)
10. frames = []
12. **for** file in files:
13. df = pd.read\_csv(file, index\_col = 0)
14. frames.append(df)
16. **return** pd.concat(frames)
18. def readOneFile(url):
19. **return** pd.read\_csv(url, index\_col = 0)
21. df = readAllFiles()

Hier lezen we de data van de originele csv files, die zich in de map */Initial\_Data/Unzipped* bevinden*,* in als 1 panda dataframe.

Om te beginnen hebben we **11.401.196** records.

## Opkuis functies

1. import pandas as pd
2. import numpy as np
3. import glob as glob
4. from math import \*
6. usaStates = [
7. "AL",
8. "AK",
9. "AZ",
10. "AR",
11. "CA",
12. "CO",
13. "CT",
14. "DE",
15. "FL",
16. "GA",
17. "HI",
18. "ID",
19. "IL",
20. "IN",
21. "IA",
22. "KS",
23. "KY",
24. "LA",
25. "ME",
26. "MD",
27. "MA",
28. "MI",
29. "MN",
30. "MS",
31. "MO",
32. "MT",
33. "NE",
34. "NV",
35. "NH",
36. "NJ",
37. "NM",
38. "NY",
39. "NC",
40. "ND",
41. "OH",
42. "OK",
43. "OR",
44. "PA",
45. "RI",
46. "SC",
47. "SD",
48. "TN",
49. "TX",
50. "UT",
51. "VT",
52. "VA",
53. "WA",
54. "WV",
55. "WI",
56. "WY",
57. ]
59. def deleteWrongStates(df):
60. print("Aantal records:", len(df))
62. **for** el in df.departure\_state.unique():
63. **if**(el not in usaStates):
64. df = df.drop(df[df['departure\_state'] == el].index)
65. print("Aantal records na verwijderen foute vertrek staat:", len(df))
67. **for** el in df.arrival\_state.unique():
68. **if**(el not in usaStates):
69. df = df.drop(df[df['arrival\_state'] == el].index)
70. print("Aantal records na verwijderen foute aankomst staat:", len(df))
72. **return** df
74. def convertColumnTypes(df):
75. df.departure\_schedule = df.departure\_schedule.astype(**int**)
76. df.departure\_delay = df.departure\_delay.astype(**float**)
77. df.arrival\_schedule = df.arrival\_schedule.astype(**int**)
78. df.arrival\_delay = df.arrival\_delay.astype(**float**)
79. df.arrival\_actual = df.arrival\_actual.astype(**int**)
80. df.departure\_actual = df.departure\_actual.astype(**int**)
81. **return** df
83. def dropMoreAdvancedDuplicates(df):
84. df = df.groupby(['date', 'airline', 'airline\_code', 'departure\_airport', 'departure\_state', 'departure\_lat', 'departure\_lon', 'departure\_schedule', 'arrival\_airport', 'arrival\_state', 'arrival\_lat', 'arrival\_lon', 'arrival\_schedule']).mean().reset\_index()
85. df = convertColumnTypes(df)
86. **if** 'index' in df.columns:
87. df = df.drop(['index'], axis=1) #remove old index
88. **return** df

De Verenigde Staten van Amerika telt 50 staten. In de data dat we gekregen hebben bevonden zich 3 foutieve, niet officiële staten. Deze hebben we verwijderd aangezien ze niet in deze dataset thuishoren.

De foutieve staten zijn:

* TT (Trust Territory of the Pacific Islands):
  + bestaat niet meer sinds 1986
  + bestaat nu uit:
    - Republic of the Marshall Islands
    - Federated States of Micronesia
    - Republic of Palau
    - Commenwealth of the Northern Mariana Islands
* PR (Puerto Rico):
  + is een territorium
* VI (U.S Virgin Islands) :
  + is een territorium

Er zijn **56 413** vluchten die vertrekken vanuit staten die geen officiële staten zijn.

Er zijn **53 184** vluchten die landen in staten die geen officiële staten zijn.

We veranderen ook de kolommen naar het gepaste type omdat de verdere verwerking makkelijker zou gaan.

Daarna controleren we op verborgen dubbels. Een verborgen dubbel komt voor indien date, airline, airline\_code, departure\_airport, departure\_state, departure\_lat, datarture\_lon, departure\_schedule, arrival\_airport, arrival\_state, arrival\_lat, arrival\_lon en arrival\_schedule hetzelfde zijn van meerdere records, maar bijvoorbeeld enkel de vertraging verschilt. In dat geval beschouwen we de records als dezelfde vlucht en nemen we het gemiddelde van de vertraging.

We controleren ook nog of er records zijn die een vertrektijd hebben die vroeger is dat de aankomsttijd. Omdat de aankomstdatum niet bijgehouden wordt, en we enkel de tijd ter beschikking hebben kan je niet zeker zijn of het om een foutief record gaat of dat de vlucht de volgende dag land.

Als extra controle kijken we of de geplande vliegtijd meer dan x aantal minuten te snel is. We hebben hier geen extreme waardes gevonden en daarom geen extra records verwijderd.

## Converteer de militaire tijd

1. import math
2. from datetime import datetime
4. def getTotalMinutes(time):
5. time\_str = str(time).rjust(4, "0")
6. hours = time\_str[:-2]
7. hours = **int**(hours)
8. **if** hours == 24:
9. hours = 0
10. minutes = **int**(time\_str[-2:])
11. **return** datetime(2010, 1, 1, hours, minutes, 0)
13. def convertMilitaryTime(df):
14. df["departure\_schedule"] = df["departure\_schedule"].map(getTotalMinutes)
15. df["arrival\_schedule"] = df["arrival\_schedule"].map(getTotalMinutes)
16. **return** df

De dataset bevat vertrek en aankomsttijden in militair formaat (bv. 615 = 06:15). Om met deze tijden te kunnen werken in Python moeten we ze omzetten naar een “datetime” object.

## Voeg tijdzones toe

1. # UTC offset by state
2. timezones = {
3. "AL" :-6,
4. "AK" :-9,
5. "AZ" :-7,
6. "AR" :-6,
7. "CA" :-8,
8. "CO" :-7,
9. "CT" :-5,
10. "DE" :-5,
11. "FL" :-5,
12. "GA" :-5,
13. "HI" :-10,
14. "ID" :-7,
15. "IL" :-6,
16. "IN" :-5,
17. "IA" : -6,
18. "KS" : -6,
19. "KY" : -6,
20. "LA" : -6,
21. "ME" : -5,
22. "MD" : -5,
23. "MA" : -5,
24. "MI" : -5,
25. "MN" : -6,
26. "MS" : -6,
27. "MO" : -6,
28. "MT" : -7,
29. "NE" : -6,
30. "NV" : -8,
31. "NH" : -5,
32. "NJ" : -5,
33. "NM" : -7,
34. "NY" : -5,
35. "NC" : -5,
36. "ND" : -6,
37. "OH" : -5,
38. "OK" : -6,
39. "OR" : -8,
40. "PA" : -5,
41. "RI" : -5,
42. "SC" : -5,
43. "SD" : -6,
44. "TN" : -6,
45. "UT": -7,
46. "TX": -6,
47. "VT": -5,
48. "VA": -5,
49. "WA": -8,
50. "DC": -5,
51. "WV": -5,
52. "WI": -6,
53. "WY": -5
54. }
56. def addTimezones(df):
57. df['arrival\_tz'] = df['arrival\_state'].map(lambda state: timezones[state])
58. df['departure\_tz'] = df['departure\_state'].map(lambda state: timezones[state])
59. **return** df

Alle tijden in de dataset zijn lokaal. Dus om ze bruikbaar te maken moeten we ze omzetten naar UTC-tijden. Hier voegen we de afwijking van UTC (in uren) toe aan elke vlucht.

## Zet alle tijden op UTC

1. from datetime import timedelta
3. def timeToUTC(schedule, tz):
4. tdelta = timedelta(hours=abs(tz))
5. **if** tz > 0:
6. schedule -= tdelta
7. **else**:
8. schedule += tdelta
9. **return** schedule
11. def convertLocalToUTC(df):
12. arrival\_schedules = []
13. departure\_schedules = []
14. **for** i, row in df.iterrows():
15. arrival\_schedule = row["arrival\_schedule"]
16. arrival\_tz = row["arrival\_tz"]
17. departure\_schedule = row["departure\_schedule"]
18. departure\_tz = row["departure\_tz"]
20. arrival\_schedules.append(timeToUTC(arrival\_schedule, arrival\_tz))
21. departure\_schedules.append(timeToUTC(departure\_schedule, departure\_tz))
23. df["arrival\_schedule"] = arrival\_schedules
24. df["departure\_schedule"] = departure\_schedules
25. **return** df

Nu we de tijdzones hebben kunnen we de tijden omzetten naar UTC.

## Bereken de vliegtijd

1. from datetime import timedelta
3. def timeToUTC(schedule, tz):
4. tdelta = timedelta(hours=abs(tz))
5. **if** tz > 0:
6. schedule -= tdelta
7. **else**:
8. schedule += tdelta
9. **return** schedule
11. def convertLocalToUTC(df):
12. arrival\_schedules = []
13. departure\_schedules = []
14. **for** i, row in df.iterrows():
15. arrival\_schedule = row["arrival\_schedule"]
16. arrival\_tz = row["arrival\_tz"]
17. departure\_schedule = row["departure\_schedule"]
18. departure\_tz = row["departure\_tz"]
20. arrival\_schedules.append(timeToUTC(arrival\_schedule, arrival\_tz))
21. departure\_schedules.append(timeToUTC(departure\_schedule, departure\_tz))
23. df["arrival\_schedule"] = arrival\_schedules
24. df["departure\_schedule"] = departure\_schedules
25. **return** df

Met de vertrek en aankomsttijden in dezelfde tijdzone kunnen we nu de duur afleiden van de vluchten.

## Bereken de afstand

1. def addDistances(df):
2. distances = []
3. **for** index, row in df.iterrows():
4. distances.append(calcTheDistance(row))
5. df["distance"] = distances
6. **return** df
8. def calcTheDistance(el):
9. slat = radians(**float**(el["arrival\_lat"]))
10. slon = radians(**float**(el["arrival\_lon"]))
11. elat = radians(**float**(el["departure\_lat"]))
12. elon = radians(**float**(el["departure\_lon"]))
13. **return** 6371.01 \* acos(sin(slat)\*sin(elat) + cos(slat)\*cos(elat)\*cos(slon - elon))

We kunnen de afstand berekenen aan de hand van coördinaten van de luchthavens.

We zijn er bewust van dat die afstand niet 100% perfect is vanwege de gebruikte methode de *Great-circle distance* bereking. Die is niet 100% perfect want de aarde geen perfecte bol. En vliegtuigen vliegen niet altijd dezelfde routes vanwege de straalstroom, weeromstandigheden… Maar het is een goede benadering.

## Bereken de snelheid

1. def addSpeed(df):
2. speeds = []
3. **for** i, row in df.iterrows():
4. duration = row["duration"]
5. speed = row["distance"] / ((duration.total\_seconds() / 60) / 60)
6. speeds.append(speed)
8. df["speed"] = speeds
9. **return** df

We berekenen de gemiddelde snelheid van het vliegtuig tijdens de geplande vlucht. Dit houd geen rekening met vertragingen.

## Verwijder te korte en te lange vluchten

1. def deleteImpossibleFlights(df):
2. shortest\_distance\_in\_us = 100 #KM
3. longest\_distance\_in\_us = 9000 #KM
4. shortest\_duration\_in\_us = timedelta(minutes=16)
5. longest\_duration\_in\_us = timedelta(hours=11, minutes=40)
7. df = df[df.distance.between(shortest\_distance\_in\_us, longest\_distance\_in\_us)]
8. df = df[df.duration.between(shortest\_duration\_in\_us,longest\_duration\_in\_us)]
9. **return** df

De kortste commerciële passagiersvlucht in de VS is 100km tussen San Francisco en Santa Rosa. De langste is 8019km tussen New York en Honolulu. Alle vluchten die hierbuiten vallen horen dus niet in onze dataset.

## Uitvoeren van de opkuis-functies

1. print("Aantal records om te beginnen:", len(df))
2. cleaning = df
4. cleaning = cleaning.drop\_duplicates()
5. print("Aantal records na verwijderen van dubbels:", len(cleaning))
7. cleaning = cleaning.reset\_index()
8. print("Aantal records na nieuwe index:", len(cleaning))
10. cleaning = cleaning.dropna()
11. print("Aantal records na verwijderen lege waarden:", len(cleaning))
13. cleaning = deleteWrongStates(cleaning)
14. cleaning = convertColumnTypes(cleaning)
16. cleaning = dropMoreAdvancedDuplicates(cleaning)
17. print("Aantal records na het verwijderen van de verborgen dubbels:", len(cleaning))
19. cleaning = convertMilitaryTime(cleaning)
20. cleaning = addTimezones(cleaning)
21. cleaning = convertLocalToUTC(cleaning)
22. cleaning = addDurations(cleaning)
23. cleaning = addDistances(cleaning)
24. cleaning = addSpeed(cleaning)
26. cleaning = deleteImpossibleFlights(cleaning)
27. print("Aantal records na het verwijderen van de onmogelijk lange of korte vluchten:", len(cleaning))
29. cleanData = cleaning

Als we de geschreven python functies voor het opkuisen van de data uitvoeren, print python de volgende uitvoer:

Aantal records om te beginnen: 11401196

Aantal records na verwijderen van dubbels: 10751921

Aantal records na nieuwe index: 10751921

Aantal records na verwijderen lege waarden: 10751919

Aantal records: 10751919

Aantal records na verwijderen foute vertrek staat: 10695506

Aantal records na verwijderen foute aankomst staat: 10642322

Aantal records na het verwijderen van de verborgen dubbels: 10642032

Aantal records na het verwijderen van de onmogelijk lange of korte vluchten: 10552133

We waren begonnen met **11.401.196** records, na het opkuisen van de data hebben we er nog **10.552.133** over. Dat wil zeggen dat er **849.063** verkeerde records waren, dat is ongeveer **7.447%** van de dataset.

# Tableau

We hebben met de data de volgende grafieken gemaakt:

* How many flights are being served by which carrier from 2011 to 2012
  + => p.
* How many flights did each carrier carried out from 2011 to 2012
  + => p.
* What are the hubs for each carrier?
  + => p.
* The total amount of scheduled flight time compared to the actual flight time per carrier from 2011 to 2012
  + => p.
* How many turns around the world could you travel from the total flow km per carrier from 2011 to 2012
  + => p.
* The flight distance compared to the scheduled flight time per carrier
  + => p.
* Per carrier what are the most popular departure states and arrival states
  + => p.
* What are the top 10 busiest airports and their average time delays
  + => p.
* Average scheduled flight time compared to the average delay per carrier from 2011 to 2012
  + => p.
* The average departure delay and the average arrival delay compared to the number of flights per month
  + => p.
* The average flight time and average delay per flight category compared with each carrier
  + => p.

## Algemene informatie over de vliegtuig maatschappijen

### American Airlines (AA)

American Airlines is opgericht in 1930.

Het hoofdkwartier is gelegen in Fort Worth in de staat Texas.

De main-hub is DFW en deze is ligt in de staat Texas.

### Alaska Airlines (AS)

Alaska Airlines is opgericht in 1932

Het hoofdkwartier is gelegen in Seattle in de staat Washington.

De main-hub is SEA en deze is ligt in de staat Washington.

### Jetblue Airways (B6)

Jetblue Airways is opgericht in 1998.

Het hoofdkwartier is gelegen in New York City in de staat New York.

De main-hub is JFK en deze is ligt in de staat New York.

### Continental Airlines (CO)

Continental Airlines is opgericht in 1934.

Het hoofdkwartier is gelegen in Houston in de staat Texas.

De main-hub is IAH en deze is ligt in de staat Texas.

Continental Airlines is echter in 2012 gestopt want het is overgenomen door United Airlines.

### Delta Air Lines (DL)

Delta Air Lines is opgericht in 1924.

Het hoofdkwartier is gelegen in Atlanta in de staat Georgia.

De main-hub is ATL en deze is ligt in de staat Georgia.

### ExpressJet Airlines (EV)

ExpressJet Airlines is opgericht in 1986

Het hoofdkwartier is gelegen in Atlanta in de staat Georgia.

De main-hub is IAH en deze is ligt in de staat Texas.

ExpressJet Airlines is echter een regionale maatschappij wilt dus zeggen dat ze in de naam van een grote speler vluchten uitvoeren.

### Frontier Airlines (F9)

Frontier Airlines is opgericht in 1994

Het hoofdkwartier is gelegen in Denver in de staat Colorado.

De main-hub is DEN en deze is ligt in de staat Colorado.

### AirTran Airways (FL)

AirTran Airways is opgericht in 1992.

Het hoofdkwartier is gelegen in Orlando in de staat Florida.

De main-hub is ATL en deze is ligt in de staat Florida.

AirTran Airways is echter in 2014 gestopt want het is overgenomen door Southwest Airlines.

### Hawaiian Airlines (HA)

Hawaiian Airlines is opgericht in 1929.

Het hoofdkwartier is gelegen in Honolulu in de staat Hawaii.

De main-hub is HNL en deze is ligt in de staat Hawaii.

### Envoy Air (MQ)

Envoy Air is opgericht in 1984.

Het hoofdkwartier is gelegen in Irving in de staat Texas.

De main-hub is DFW en deze is ligt in de staat Texas.

Envoy Air is echter een regionale maatschappij wilt dus zeggen dat ze in de naam van een grote speler vluchten uitvoeren.

### SkyWest Airlines (OO)

SkyWest Airlines is opgericht in 1972.

Het hoofdkwartier is gelegen in St. George in de staat Utah.

De main-hub is LA en deze is ligt in de staat California.

SkyWest Airlines is echter een regionale maatschappij wilt dus zeggen dat ze in de naam van een grote speler vluchten uitvoeren.

### United Airlines (UA)

United Airlines is opgericht in 1926.

Het hoofdkwartier is gelegen in Chicago in de staat Illinois.

De main-hub is ORD en deze is ligt in de staat Illinois.

### US Airways (US)

US Airways is opgericht in 1937.

Het hoofdkwartier is gelegen in Temp in de staat Arizona.

De main-hub is PHL en deze is ligt in de staat Pennsylvania.

US Airways is echter in 2013 gestopt want het bedrijf fuseerde met American Airlines.

### Virgin America (VX)

Virgin America is opgericht in 2004.

Het hoofdkwartier is gelegen in Burlingame in de staat California.

De main-hub is SFO en deze is ligt in de staat California.

Virgin America is echter in 2018 gestopt want het is overgenomen door Alaska Airlines.

### Southwest Airlines (WN)

Southwest Airlines is opgericht in 1967.

Het hoofdkwartier is gelegen in Dallas in de staat Texas.

De main-hub is MDW en deze is ligt in de staat Illinois.

### JetSuite (XE)

JetSuite is opgericht in 2006.

Het hoofdkwartier is gelegen in Irvine in de staat California.

De main-hub is IAH en deze is ligt in de staat Texas.

JetSuite is echter een private jet maatschappij.

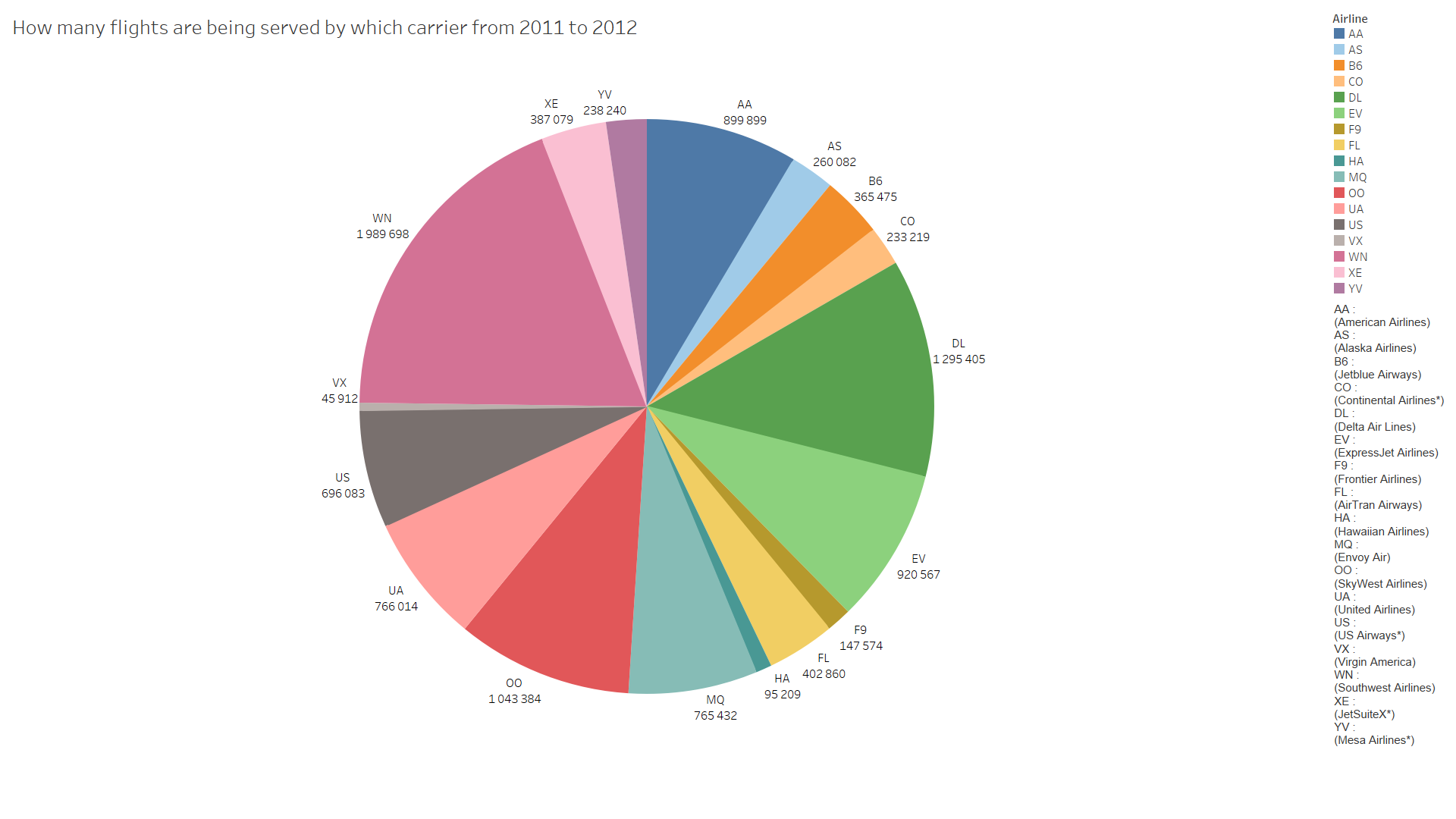
### Mesa Airlines (YV)

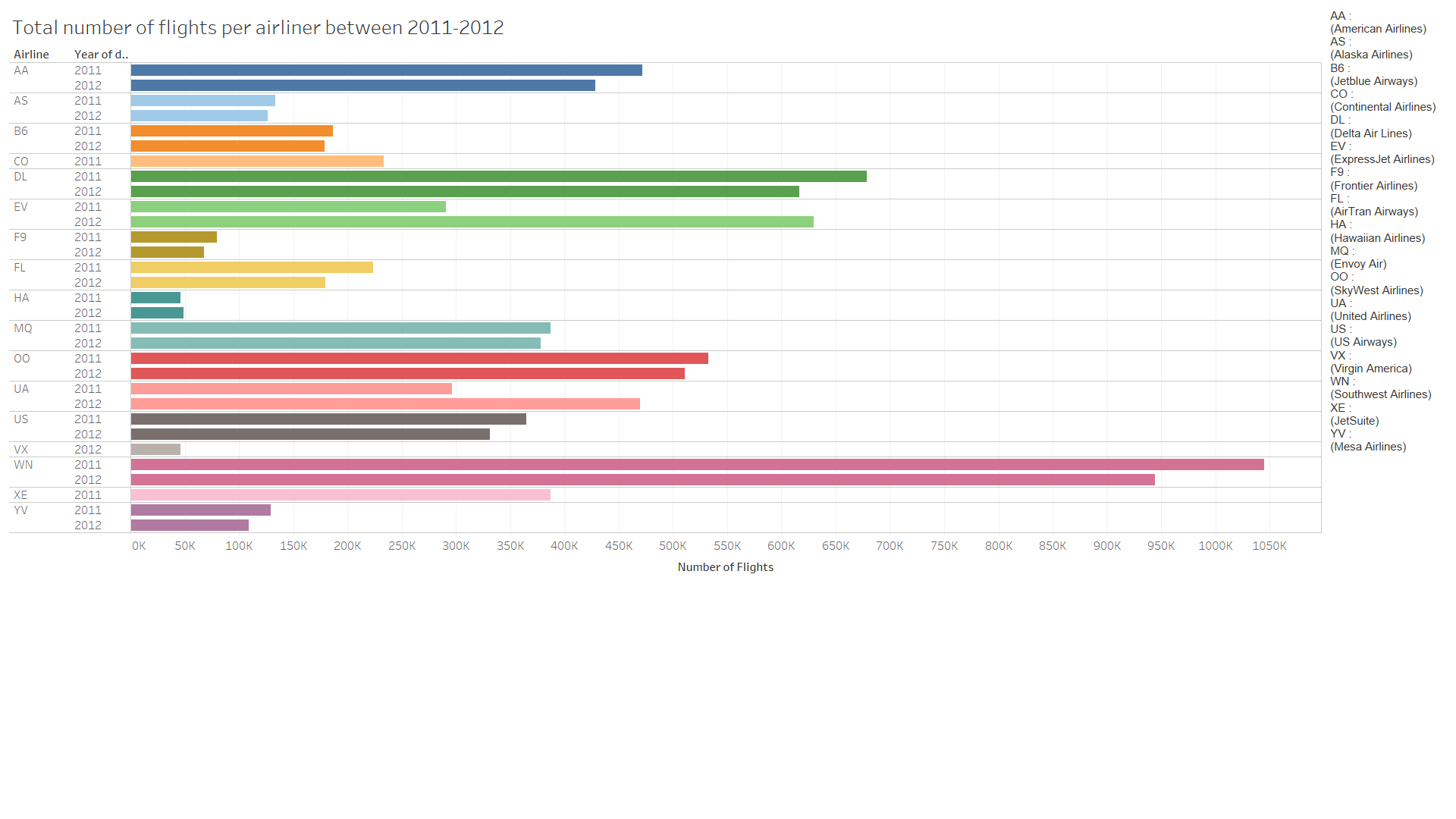
Mesa Airlines is opgericht in 1980.

Het hoofdkwartier is gelegen in Phoenix in de staat Arizona.

De main-hub is PHX en deze is ligt in de staat Arizona.

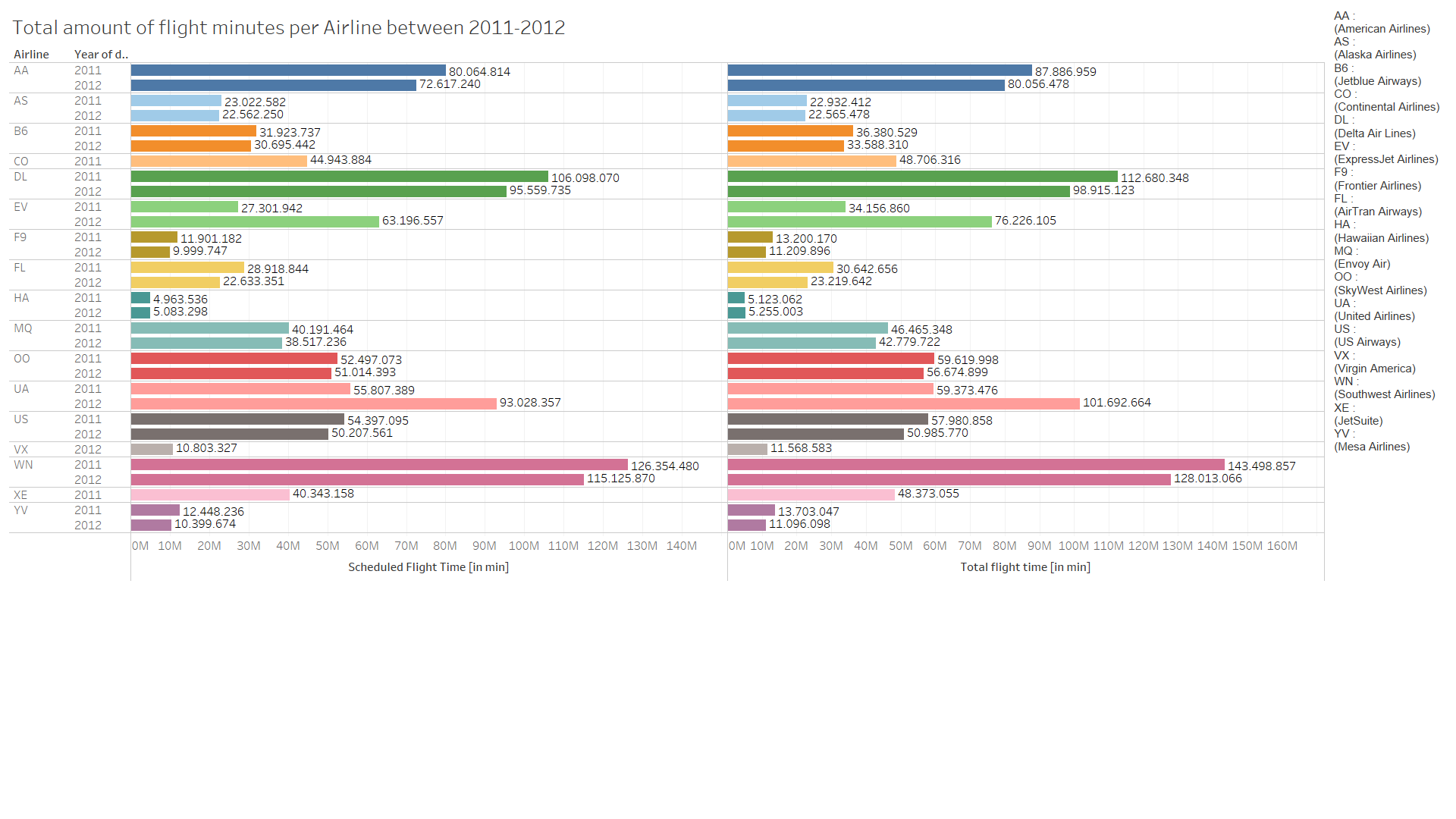
Mesa Airlines is echter een regionale maatschappij wilt dus zeggen dat ze in de naam van een grote speler vluchten uitvoeren.

Grafiek 1: How many flights are being serverd by which carrier from 2011 to 2012

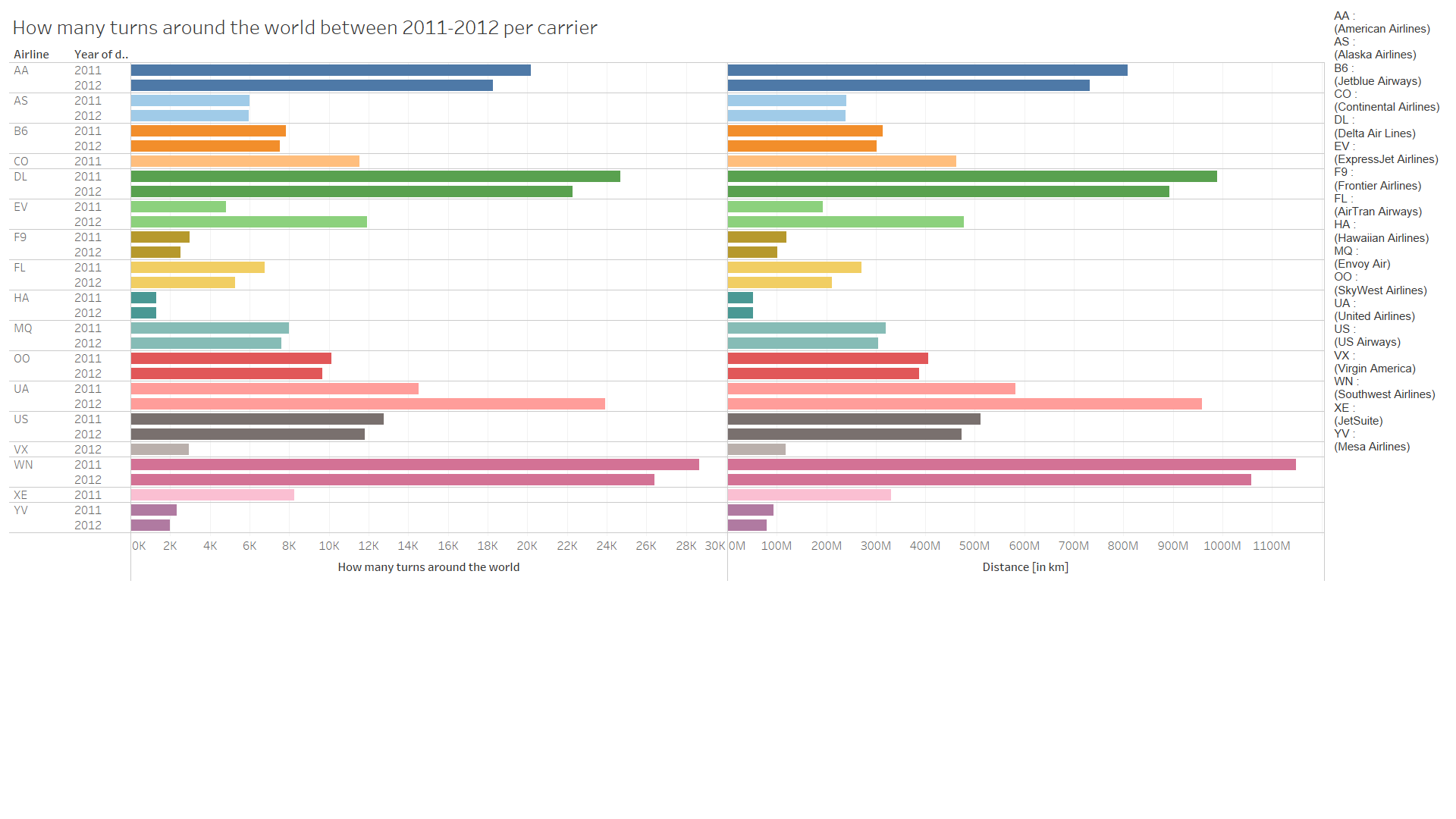
Grafiek 2: How many flights did each carrier carried out from 2011 to 2012

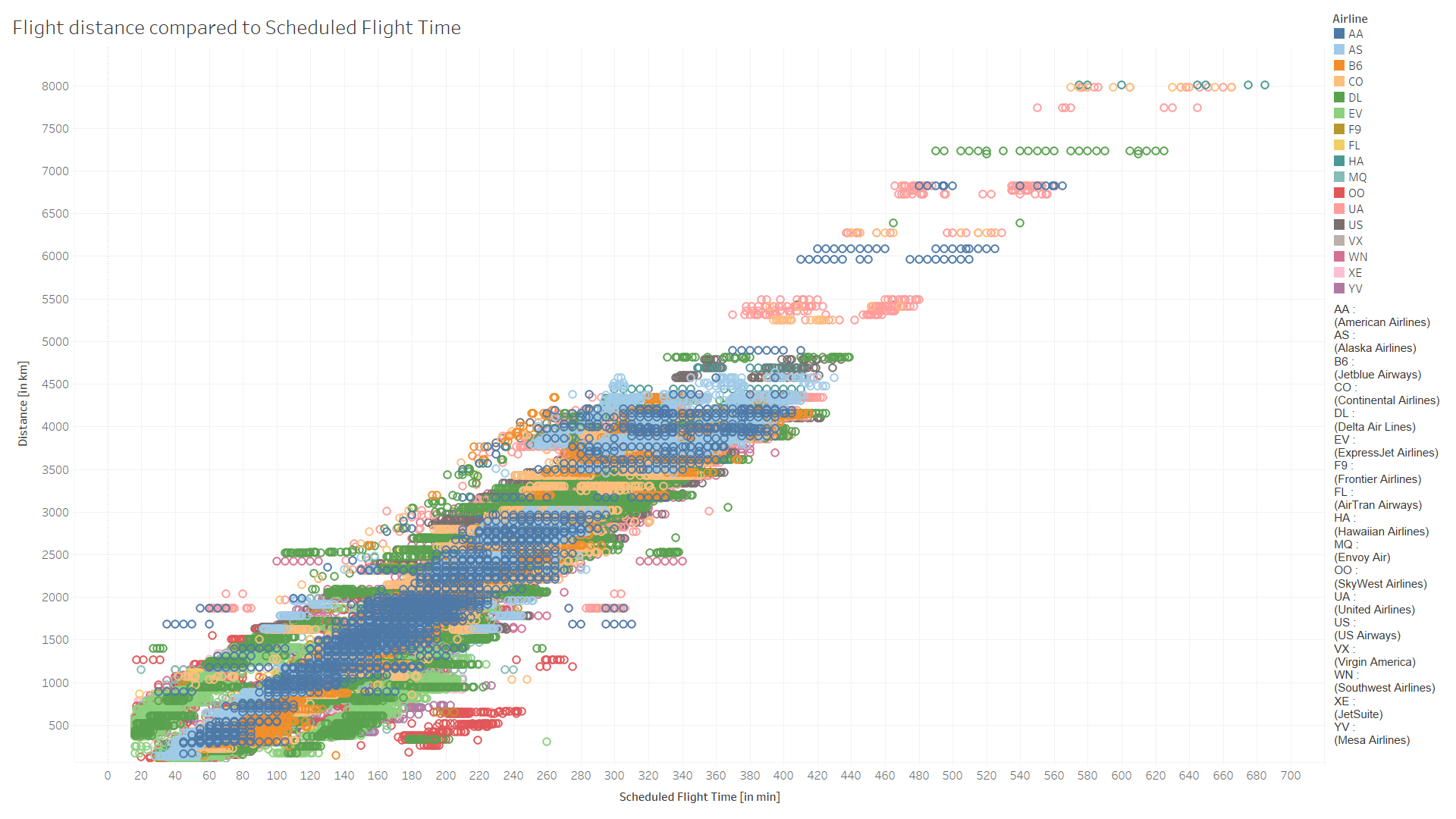
##### C:\Users\svend\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Hubs.pngGrafiek 3: What are the hubs for each carrier?

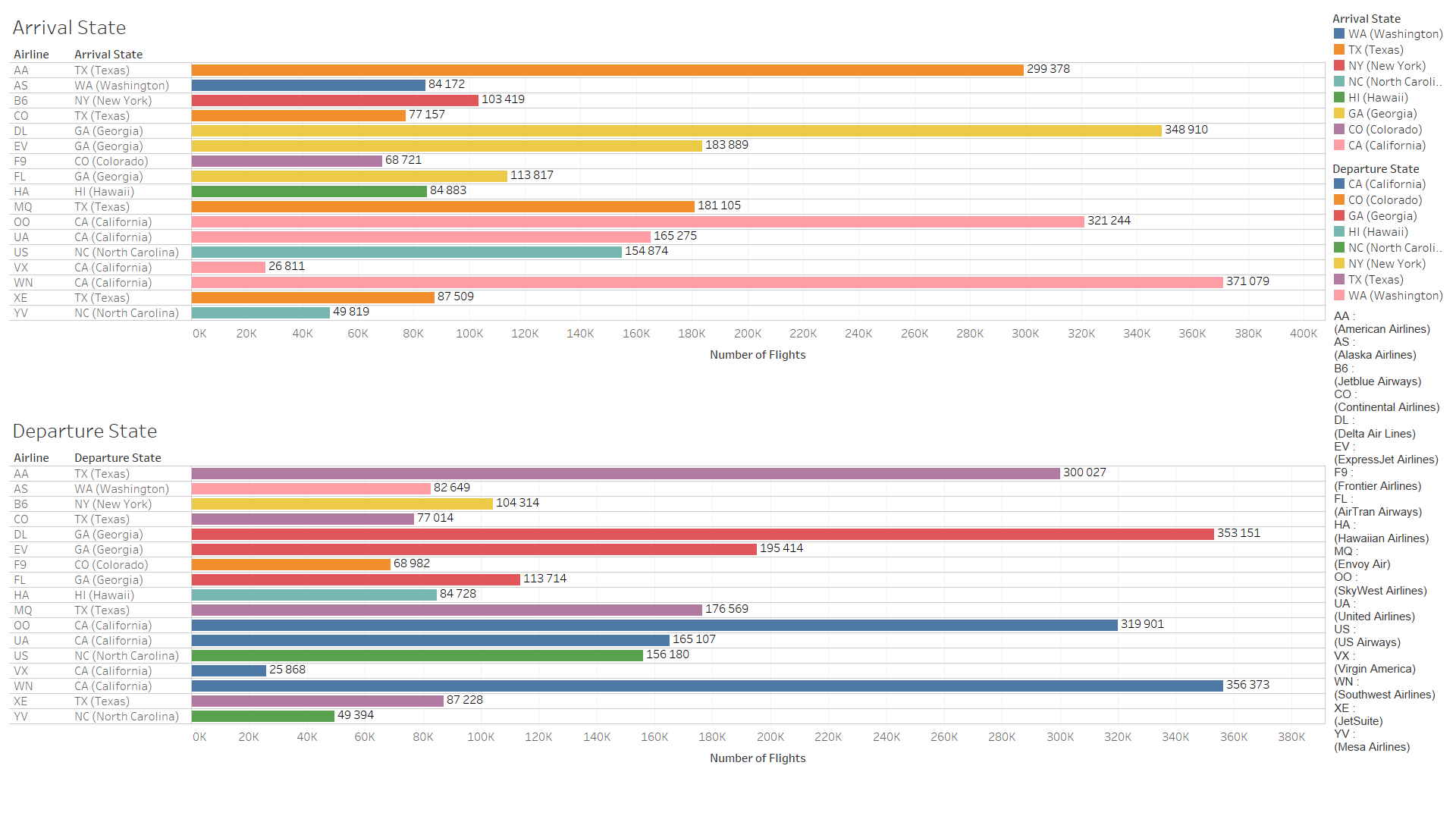
Grafiek 4: The total amount of scheduled flight time compared to the actual flight time per carrier from 2011 to 2012



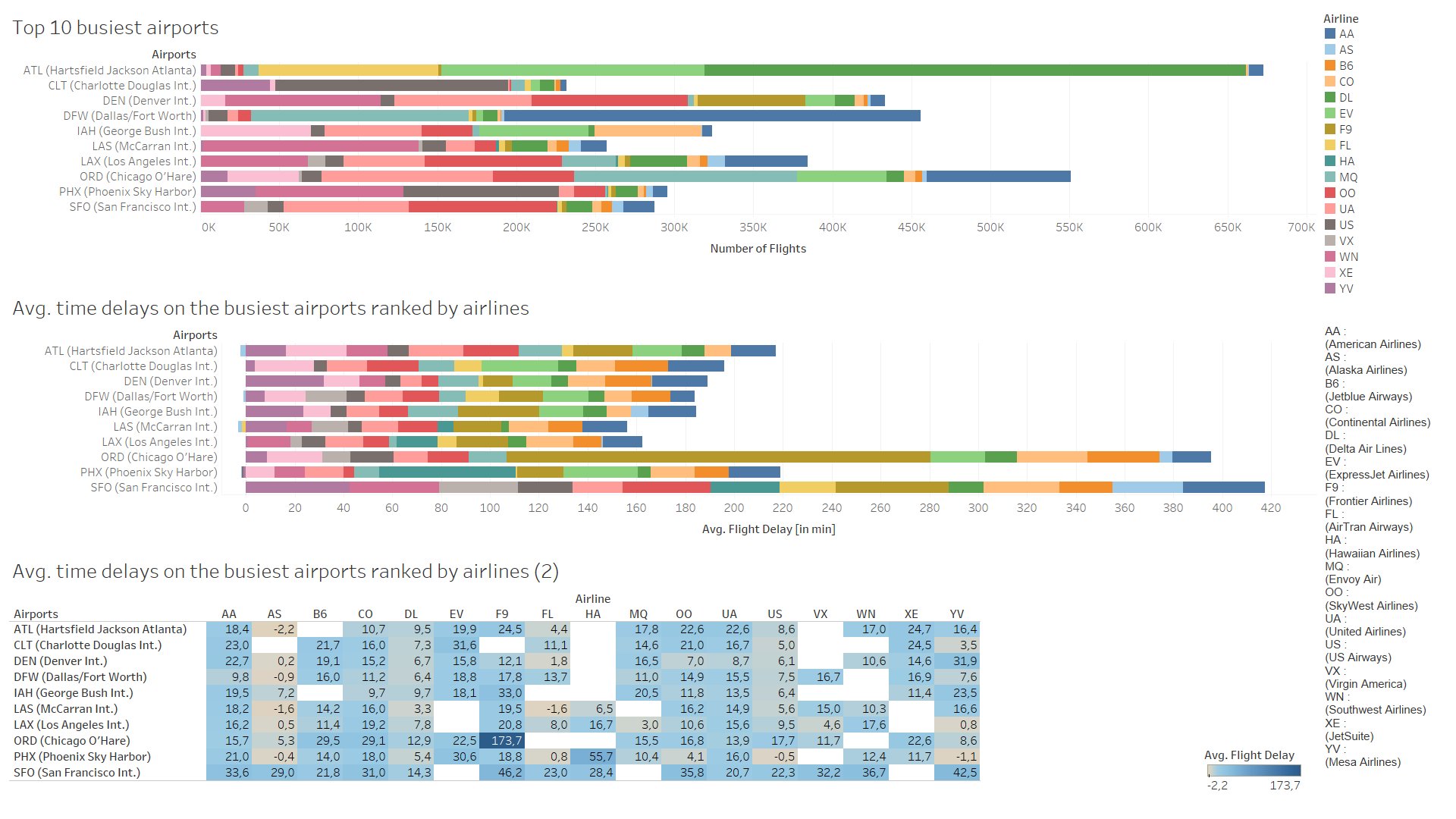
Grafiek 5: How many turns around the world could you travel from the total flown km per carrier from 2011 to 2012



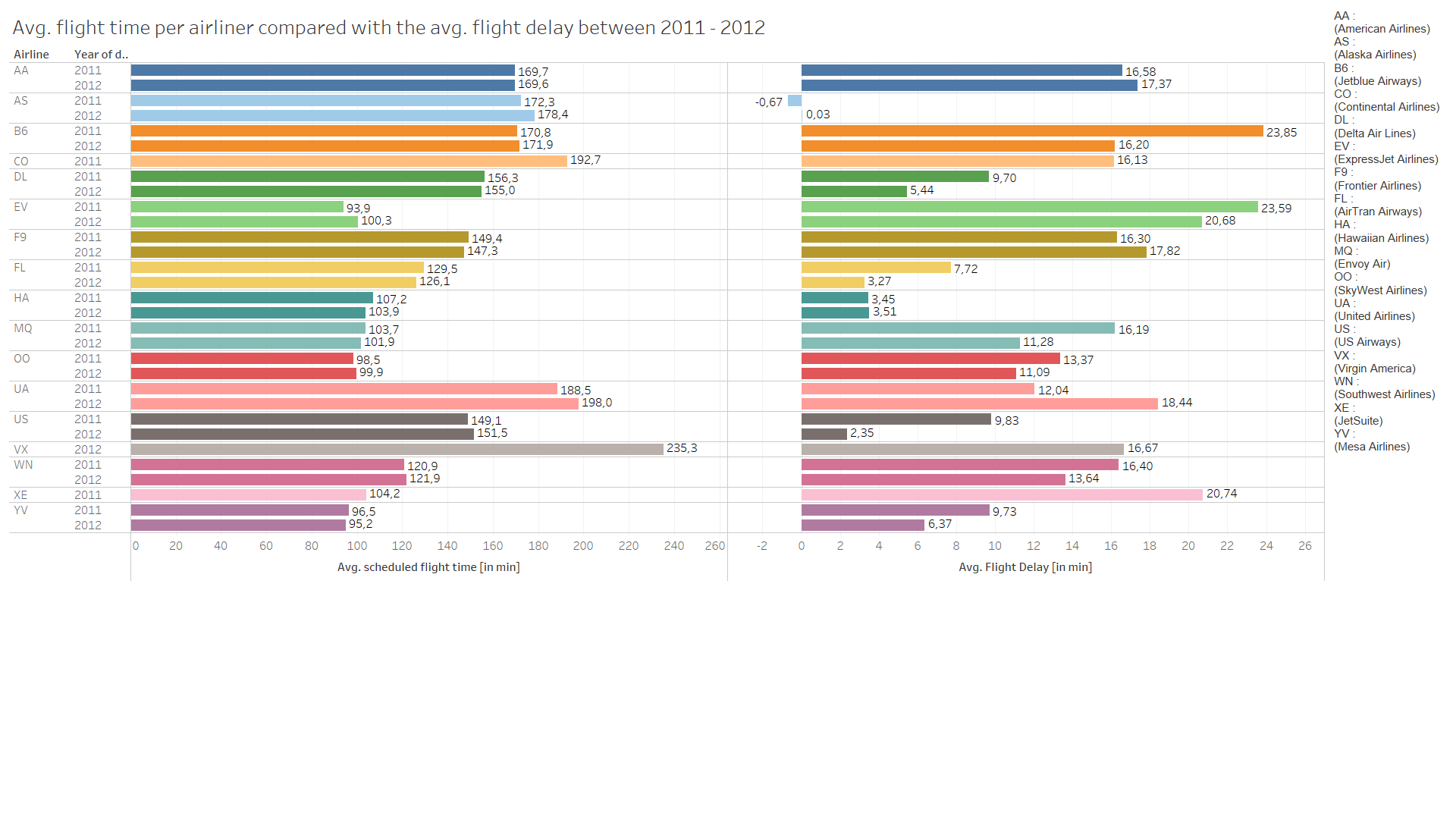
Grafiek 6: The fight distance compared to the scheduled flight time per carrier

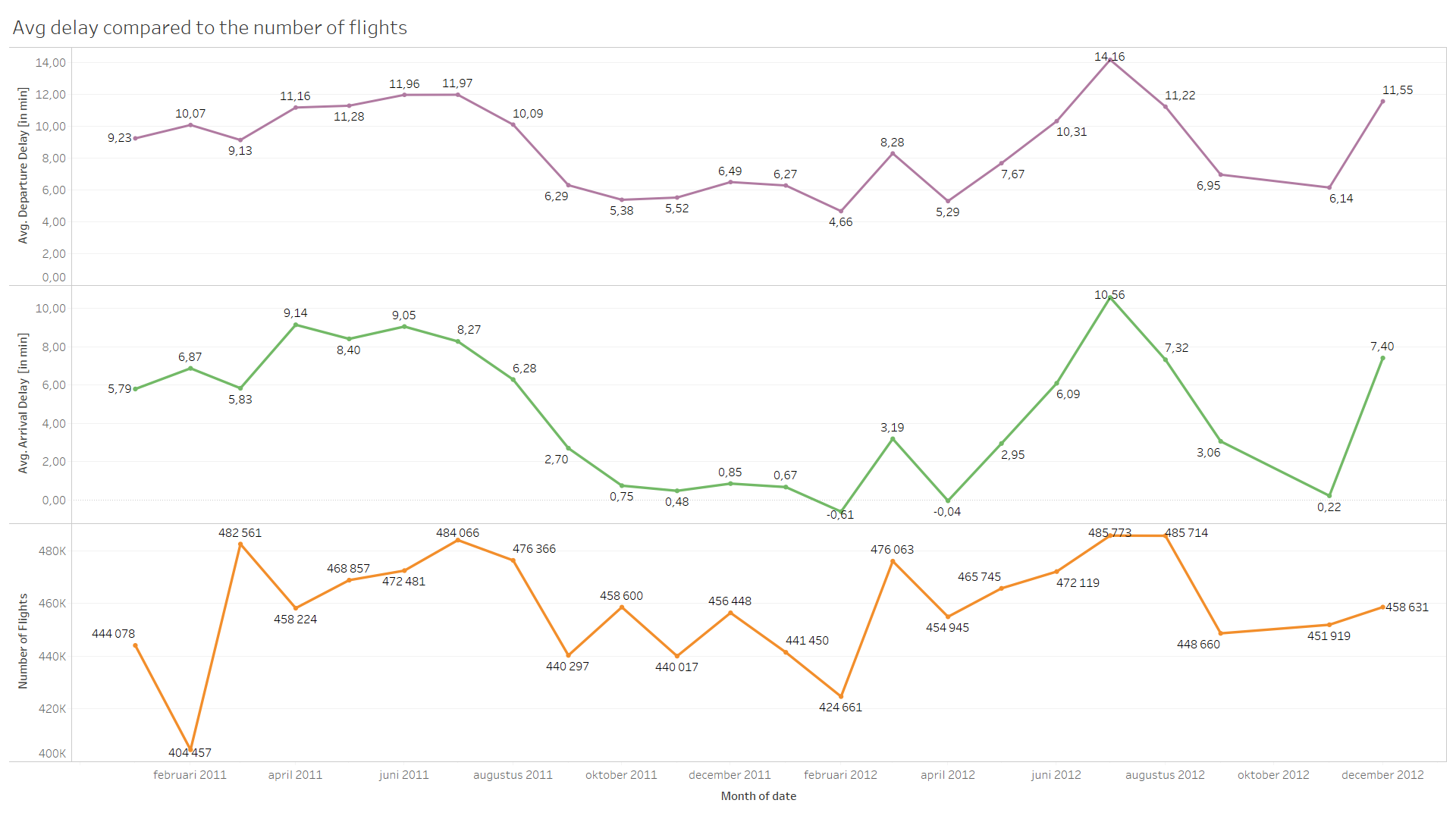
Grafiek 7: Per carrier what are the most popular departure states and arrival states

Grafiek 8: What are top 10 busiest airports and their average time delays

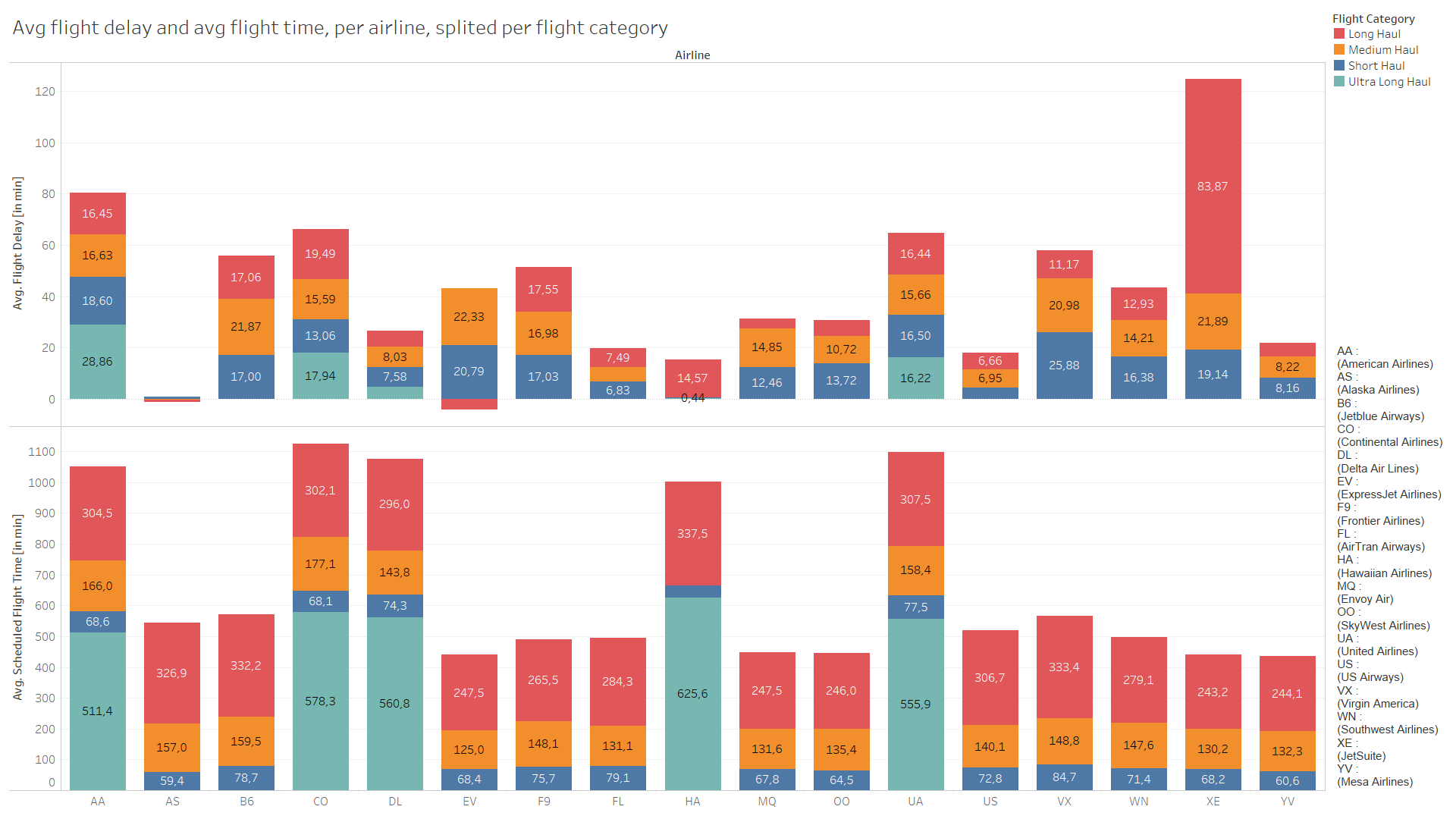


Grafiek 9: Average scheduled flight time compared the average delay per carrier from 2011 to 2012



Grafiek 10: The average depature delay and the average arrival delay compared to the number of flights per month

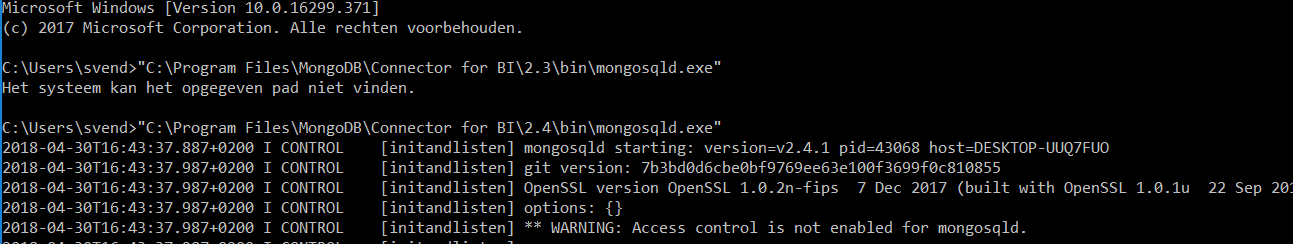
Grafiek 11: The average flight time and average delay per flight category compared with each carrier



"C:\Program Files\MongoDB\Connector for BI\2.4\bin\mongosqld.exe"

cd C:\Program Files\MongoDB\Server\3.6\bin

https://www.transtats.bts.gov/DL\_SelectFields.asp



# Machine learning met Scikit learn

# NetworkX

## Laad de dataset

1. **import** math
2. **import** json
3. **import** numpy as np
4. **import** pandas as pd
5. **import** networkx as nx
6. **import** cartopy.crs as ccrs
7. **import** matplotlib.pyplot as plt
8. **from** IPython.display **import** Image
9. %matplotlib inline
11. **import** glob
13. BASIC\_PATH = './Data/'
14. ALL\_FILES = BASIC\_PATH + '\*.csv'
16. **def** readOneFile(url):
17. **return** pd.read\_csv(url, index\_col = 0)
19. data = readOneFile(BASIC\_PATH + "flights\_2010\_9.csv")
20. flights = pd.DataFrame(data=data)
21. flights

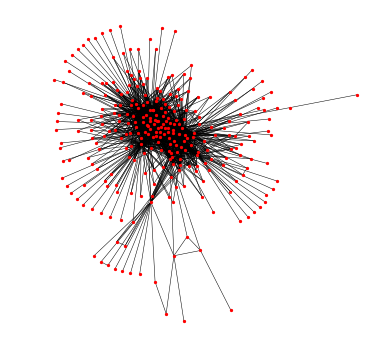
## Routes tussen luchthavens

1. edges = flights[['departure\_airport', 'arrival\_airport']].values
2. edges

Uit de vluchten leiden we de routes tussen luchthavens af.

## Grafiek van alle luchthavens en hun routes

1. g = nx.from\_edgelist(edges)
2. fig, ax = plt.subplots(1, 1, figsize=(6, 6))
3. nx.draw\_networkx(g, ax=ax, with\_labels=False,
4. node\_size=5, width=.5)
5. ax.set\_axis\_off()



## Vind de coördinaten van elke luchthaven

1. pos = {}
2. airports = flights[["departure\_airport", "departure\_lat", "departure\_lon"]].drop\_duplicates()
3. **for** i, row **in** airports.iterrows():
4. airport = row["departure\_airport"]
5. pos[airport] = (row["departure\_lon"], row["departure\_lat"])

Elke vlucht heeft coördinaten bij het vertrek punt en de aankomst. We kunnen dus de locatie van elke luchthaven hieruit afleiden.

## Plaats de routes op kaart

1. deg = nx.degree(g)
2. sizes = [5 \* deg[iata] **for** iata **in** g.nodes]
4. labels = {iata: iata **if** deg[iata] >= 60 **else** ''
5. **for** iata **in** g.nodes}
7. crs = ccrs.PlateCarree()
8. fig, ax = plt.subplots(
9. 1, 1, figsize=(12, 8),
10. subplot\_kw=dict(projection=crs))
12. ax.coastlines()
14. ax.set\_extent([-163, -67, 18, 64])
15. nx.draw\_networkx(g, ax=ax,
16. font\_size=14,
17. alpha=.5,
18. width=.07,
19. node\_size=sizes,
20. labels=labels,
21. pos=pos,
22. node\_color=sizes,
23. cmap=plt.cm.cool)

De volgende kaart bevat elke route en luchthaven in de dataset. De grote en kleur van een luchthaven geeft aan hoeveel connecties het bevat.

