

# 1 Project 7

In this project, you will implement the the clustering techniques that you've learned this week.

## 1.0.0.1 Step 1: Load the python libraries that you will need for this project

```
In [77]: import pandas as pd
import numpy as np
import scipy
from sklearn import preprocessing, decomposition

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

import psycopg2 as psy
import sqlalchemy

%matplotlib inline
```

Last executed 2016-11-06 22:14:13 in 783ms

```
In [2]: plt.style.use("fivethirtyeight")
```

Last executed 2016-11-06 21:44:57 in 5ms

```
In [3]: %load_ext sql
```

Last executed 2016-11-06 21:44:58 in 569ms

## 1.0.0.2 Step 2: Examine your data

```
In [4]: airports = pd.read_csv("../assets/airports.csv")
```

Last executed 2016-11-06 21:44:59 in 212ms

```
In [5]: airports.columns = [i.replace("_", " ").lower().replace(" ", "_") for i in
```

Last executed 2016-11-06 21:45:00 in 5ms

In [6]:	key	locid	ap_name	alias	facility_type	faa_region	county	city
Last executed 2016-11-06 21:45:01 in 171ms								
<class 'pandas.core.frame.DataFrame'>								
RangeIndex: 5167 entries, 0 to 5166								
Data columns (total 13 columns):								
key								
5164 non-null float64								
locid								
5152 non-null object								
ap_name								
5164 non-null object								
alias								
3498 non-null object								
facility_type								
5164 non-null object								
faa_region								
5164 non-null object								
county								
5164 non-null object								
city								
5164 non-null object								
state								
5164 non-null object								
ap_type								
5164 non-null object								
latitude								
5164 non-null float64								
longitude								
5164 non-null float64								
boundary_data_available								
5164 non-null object								
dtypes: float64(3), object(10)								
memory usage: 524.8+ KB								

In [7]:	airports.head()																																																														
Last executed 2016-11-06 21:45:01 in 115ms																																																															
Out[7]:																																																															
<table border="1"> <thead> <tr> <th></th><th>key</th><th>locid</th><th>ap_name</th><th>alias</th><th>facility_type</th><th>faa_region</th><th>county</th><th>city</th></tr> </thead> <tbody> <tr> <td>0</td><td>3443.0</td><td>STX</td><td>HENRY E ROHLSEN</td><td>Henry E Rohlsen Int'l Airport</td><td>Airport</td><td>ASO</td><td>-VIRGIN ISLANDS-</td><td>CHRIS</td></tr> <tr> <td>1</td><td>5088.0</td><td>X64</td><td>PATILLAS</td><td>Nan</td><td>Airport</td><td>ASO</td><td>#NAME?</td><td>PATILL</td></tr> <tr> <td>2</td><td>2886.0</td><td>PSE</td><td>MERCEDITA</td><td>Aeropuerto Mercedita</td><td>Airport</td><td>ASO</td><td>#NAME?</td><td>PONC</td></tr> <tr> <td>3</td><td>2879.0</td><td>VQS</td><td>ANTONIO RIVERA RODRIGUEZ</td><td>Aeropuerto Antonio Rivera Rodriguez</td><td>Airport</td><td>ASO</td><td>#NAME?</td><td>ISLA DE VIEQU</td></tr> <tr> <td>4</td><td>2883.0</td><td>X63</td><td>HUMACAO</td><td>Aeropuerto Regional De Humacao</td><td>Airport</td><td>ASO</td><td>#NAME?</td><td>HUMACAO</td></tr> </tbody> </table>											key	locid	ap_name	alias	facility_type	faa_region	county	city	0	3443.0	STX	HENRY E ROHLSEN	Henry E Rohlsen Int'l Airport	Airport	ASO	-VIRGIN ISLANDS-	CHRIS	1	5088.0	X64	PATILLAS	Nan	Airport	ASO	#NAME?	PATILL	2	2886.0	PSE	MERCEDITA	Aeropuerto Mercedita	Airport	ASO	#NAME?	PONC	3	2879.0	VQS	ANTONIO RIVERA RODRIGUEZ	Aeropuerto Antonio Rivera Rodriguez	Airport	ASO	#NAME?	ISLA DE VIEQU	4	2883.0	X63	HUMACAO	Aeropuerto Regional De Humacao	Airport	ASO	#NAME?	HUMACAO
	key	locid	ap_name	alias	facility_type	faa_region	county	city																																																							
0	3443.0	STX	HENRY E ROHLSEN	Henry E Rohlsen Int'l Airport	Airport	ASO	-VIRGIN ISLANDS-	CHRIS																																																							
1	5088.0	X64	PATILLAS	Nan	Airport	ASO	#NAME?	PATILL																																																							
2	2886.0	PSE	MERCEDITA	Aeropuerto Mercedita	Airport	ASO	#NAME?	PONC																																																							
3	2879.0	VQS	ANTONIO RIVERA RODRIGUEZ	Aeropuerto Antonio Rivera Rodriguez	Airport	ASO	#NAME?	ISLA DE VIEQU																																																							
4	2883.0	X63	HUMACAO	Aeropuerto Regional De Humacao	Airport	ASO	#NAME?	HUMACAO																																																							

In [8]:	cancellations = pd.read_csv("../assets/airport_cancellations.csv")
Last executed 2016-11-06 21:45:02 in 19ms	
In [9]: cancellations.columns = [i.replace("_", " ").lower().replace(" ", "_") for i in cancellations.columns]	
Last executed 2016-11-06 21:45:02 in 7ms	

```
In [10]: cancellations.info()
```

```
Last executed 2016-11-06 21:45:03 in 14ms
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 805 entries, 0 to 804
Data columns (total 6 columns):
 airport                  805 non-null object
 year                     805 non-null float64
 departure_cancellations 805 non-null float64
 arrival_cancellations   805 non-null float64
 departure_diversions    805 non-null float64
 arrival_diversions      805 non-null float64
 dtypes: float64(5), object(1)
memory usage: 37.8+ KB
```

```
In [11]: cancellations.head()
```

```
Last executed 2016-11-06 21:45:03 in 115ms
```

```
Out[11]:
```

	airport	year	departure_cancellations	arrival_cancellations	departure_diversions	arr...
0	ABQ	2004.0	242.0	235.0	71.0	46...
1	ABQ	2005.0	221.0	190.0	61.0	33...
2	ABQ	2006.0	392.0	329.0	71.0	12...
3	ABQ	2007.0	366.0	304.0	107.0	45...
4	ABQ	2008.0	333.0	300.0	79.0	42...

```
In [12]: operations = pd.read_csv("../assets/Airport_operations.csv")
```

```
Last executed 2016-11-06 21:45:04 in 21ms
```

```
In [13]: operations.columns = [i.replace("_", " ").replace("-", " ").lower().replac...
```

```
Last executed 2016-11-06 21:45:04 in 6ms
```

```
In [14]: operations.info()
```

Last executed 2016-11-06 21:45:05 in 16ms

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 841 entries, 0 to 840
Data columns (total 15 columns):
airport                      841 non-null object
year                          841 non-null int64
departures_for_metric_computation 841 non-null int64
arrivals_for_metric_computation   841 non-null int64
percent_on_time_gate_departures 841 non-null float64
percent_on_time_airport_departures 841 non-null float64
percent_on_time_gate_arrivals    841 non-null float64
average_gate_departure_delay    841 non-null float64
average_taxi_out_time          841 non-null float64
average_taxi_out_delay         841 non-null float64
average_airport_departure_delay 841 non-null float64
average_airborne_delay         841 non-null float64
average_taxi_in_delay          841 non-null float64
average_block_delay            841 non-null float64
average_gate_arrival_delay     841 non-null float64
dtypes: float64(11), int64(3), object(1)
memory usage: 98.6+ KB
```

```
In [15]: operations.head()
```

Last executed 2016-11-06 21:45:06 in 81ms

	airport	year	departures_for_metric_computation	arrivals_for_metric_computation	per
0	ABQ	2004	53971	53818	0.81
1	ABQ	2005	51829	51877	0.81
2	ABQ	2006	49682	51199	0.79
3	ABQ	2007	53255	53611	0.81
4	ABQ	2008	49589	49512	0.81

### 1.0.1 Intro: Write a problem statement / aim for this project

The FAA wants to cut down on delays nationwide, and the most important part of this task dataset of departure and operational delays.

Objective: Your task is to understand the distribution, characteristics, and components of

### 1.0.2 Part 1: Create a PostgreSQL database

#### 1.0.2.1 1. Let's create a database where we can house our airport data

```
In [ ]: # the speeddating database was created in terminal with "CREATE DATABASE
# use %sql magic to connect to the database
# %sql dialect+driver://username:password@host:port/database
%sql postgresql://joce@localhost:5432/airports
```

```
In [ ]: # check what tables are available in our database right now
Last executed 2016-11-06 21:45:08 in 2.36s
```

```
In [ ]: %%sql
SELECT table_name
FROM information_schema.tables
WHERE table_schema LIKE 'public';
```

```
Last executed 2016-11-06 21:45:09 in 1.36s
```

```
In [ ]: # to use the pandas.to_sql function, create a connection to the database
engine = sqlalchemy.create_engine("postgresql://joce@localhost:5432/airpc
Last executed 2016-11-06 21:45:09 in 1.03s
```

Load our csv files into tables

```
In [ ]: airports.to_sql("airports", con=engine, if_exists="replace")
Last executed 2016-11-06 21:45:10 in 618ms
```

```
In [ ]: %%sql
SELECT * FROM airports LIMIT 3;
Last executed 2016-11-06 21:45:10 in 402ms
```

```
In [ ]: cancellations.to_sql("cancellations", con=engine, if_exists="replace")
Last executed 2016-11-06 21:45:10 in 190ms
```

```
In [ ]: %%sql
SELECT * FROM cancellations LIMIT 3;
```

```
In [ ]: operations.to_sql("operations", con=engine, if_exists="replace")
```

```
In [ ]: %%sql
SELECT * FROM operations LIMIT 3;
```

Join airport\_cancellations.csv and airports.csv into one table

```
In [40]: a_and_c = cancellations.merge(airports, left_on="airport", right_on="loci
Last executed 2016-11-06 21:46:07 in 38ms
```

```
In [41]: a_and_c.head(3)
```

```
Last executed 2016-11-06 21:46:07 in 143ms
```

```
Out[41]:
```

	airport	year	departure_cancellations	arrival_cancellations	departure_diversions	arr
0	ABQ	2004.0	242.0	235.0	71.0	46.
1	ABQ	2005.0	221.0	190.0	61.0	33.
2	ABQ	2006.0	392.0	329.0	71.0	12.

```
In [42]: a_and_o = operations.merge(airports, left_on="airport", right_on="locid")
```

```
Last executed 2016-11-06 21:46:07 in 26ms
```

```
In [43]: a_and_o.head(3)
```

```
Last executed 2016-11-06 21:46:07 in 108ms
```

```
Out[43]:
```

	airport	year	departures_for_metric_computation	arrivals_for_metric_computation	per
0	ABQ	2004	53971	53818	0.81
1	ABQ	2005	51829	51877	0.8
2	ABQ	2006	49682	51199	0.79

3 rows × 28 columns

Query the database for our initial data

```
In [ ]: %%sql
```

```
SELECT COUNT(DISTINCT locid)
FROM airports;
```

```
In [ ]: %%sql
```

```
SELECT COUNT(DISTINCT airport)
FROM cancellations;
```

```
Last executed 2016-11-06 21:45:12 in 10ms
```

```
In [ ]: %%sql
```

```
SELECT COUNT(DISTINCT airport)
FROM operations;
```

```
In [44]: df = a_and_o.copy()
```

```
Last executed 2016-11-06 21:46:11 in 6ms
```

```
In [45]: df.info()
```

```
Last executed 2016-11-06 21:46:12 in 16ms
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 841 entries, 0 to 840
Data columns (total 28 columns):
airport                      841 non-null object
year                          841 non-null int64
departures_for_metric_computation 841 non-null int64
arrivals_for_metric_computation   841 non-null int64
percent_on_time_gate_departures 841 non-null float64
percent_on_time_airport_departures 841 non-null float64
percent_on_time_gate_arrivals    841 non-null float64
average_gate_departure_delay    841 non-null float64
average_taxi_out_time           841 non-null float64
average_taxi_out_delay          841 non-null float64
average_airport_departure_delay 841 non-null float64
average_airborne_delay          841 non-null float64
average_taxi_in_delay           841 non-null float64
average_block_delay              841 non-null float64
average_gate_arrival_delay      841 non-null float64
key                           841 non-null float64
locid                         841 non-null object
ap_name                        841 non-null object
alias                          841 non-null object
facility_type                  841 non-null object
faa_region                     841 non-null object
county                         841 non-null object
city                           841 non-null object
state                          841 non-null object
ap_type                        841 non-null object
latitude                       841 non-null float64
longitude                      841 non-null float64
boundary_data_available        841 non-null object
dtypes: float64(14), int64(3), object(11)
memory usage: 190.5+ KB
```

### 1.0.2.2 1.2 What are the risks and assumptions of our data?

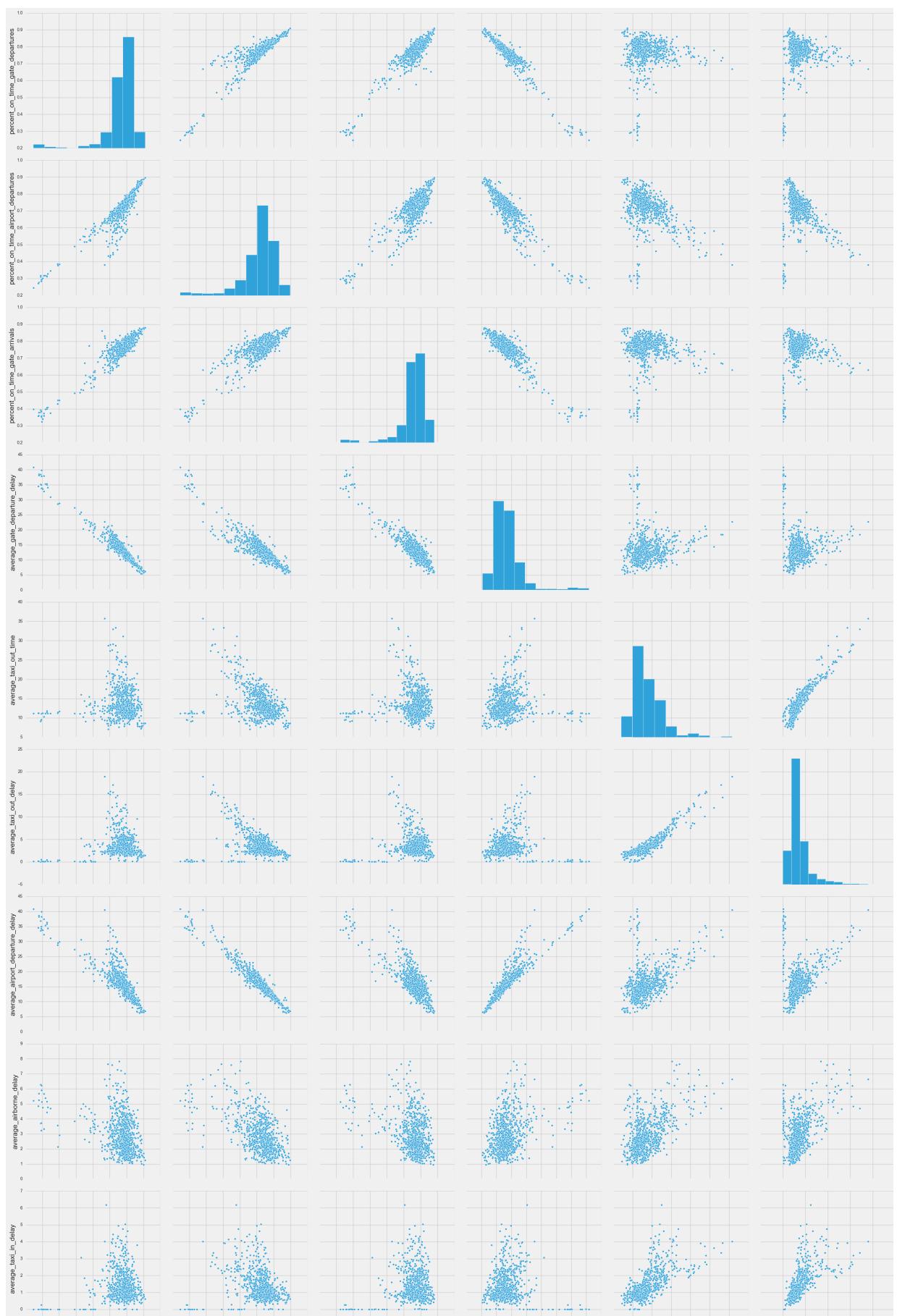
- Our data is only available by the year
  - Assumption: The time of the year/ seasons/ time of the day do not have any effect on delay
- Out of 5152 airports, we only have cancellation data for 74 airports and operations delay data
  - Assumption: The airports that we have data for is representative of all the airports
- We don't have airline data
  - Assumption: The airline does not have any effect on delays or cancellations

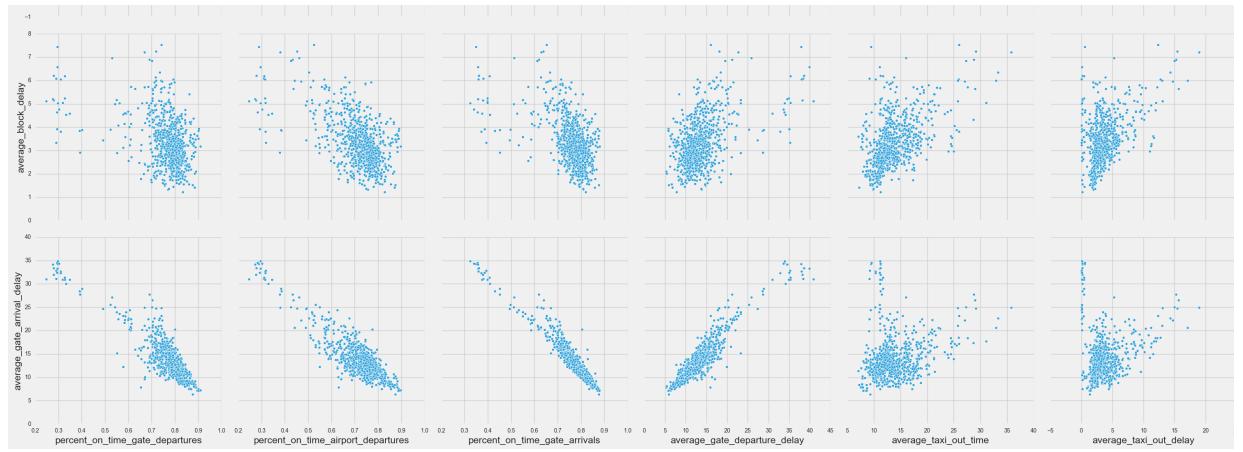
## 1.0.3 Part 2: Exploratory Data Analysis

### 1.0.3.1 2.1 Plot and Describe the Data

```
In [46]: sns.pairplot(df[[i for i in df.columns if "average" in i or "percent" in i]]);
```

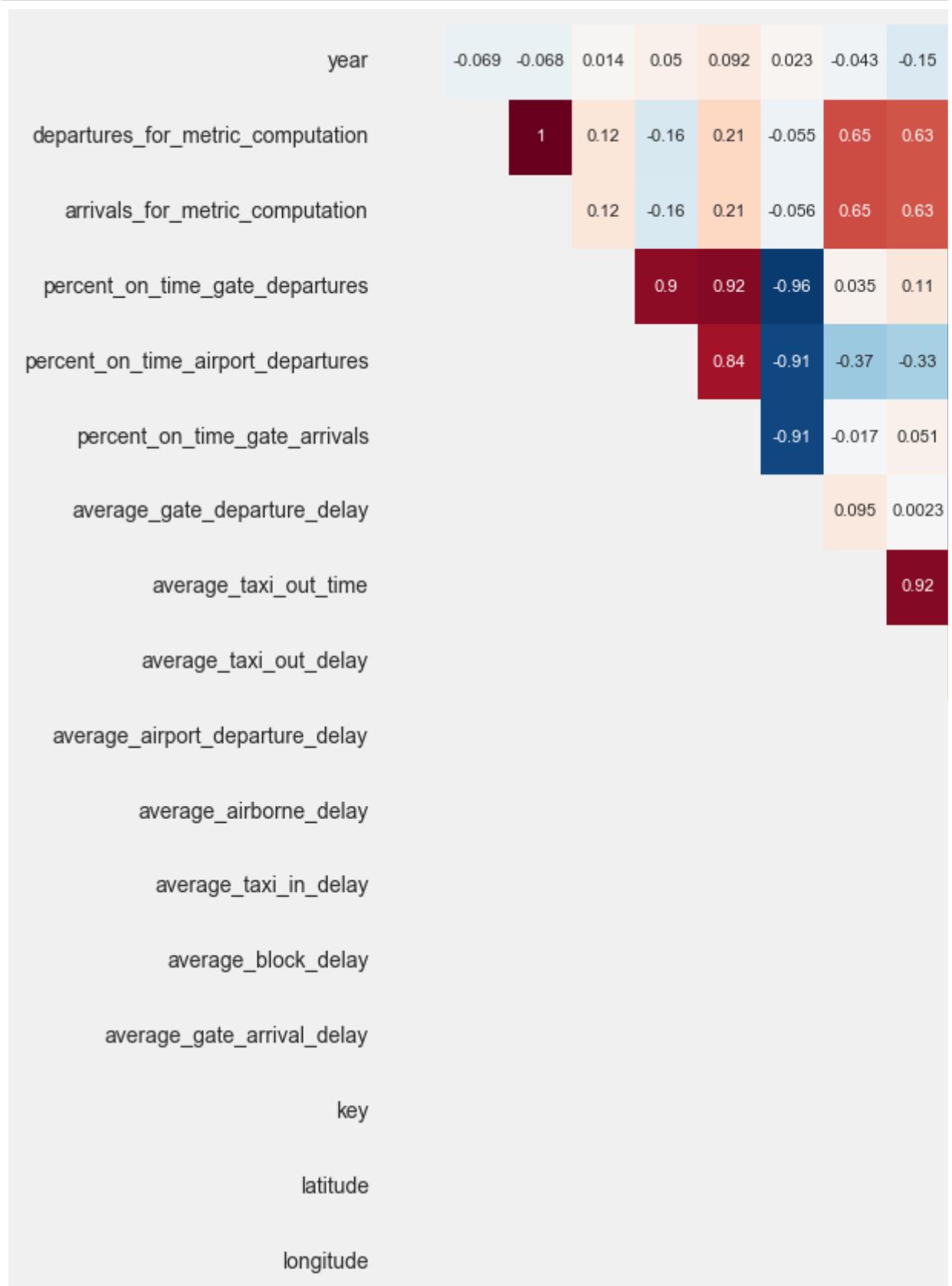
Last executed 2016-11-06 21:46:14 in 39.8s





```
In [47]: # create a mask so that only one half of our heatmap shows up
# this takes out duplicates in the grid and also the spots where a variak
mask = np.zeros_like(df.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
plt.subplots(figsize=(10,10));
sns.heatmap(df.corr(), mask=mask.T, annot=True, annot_kws={"fontsize": 8})
```

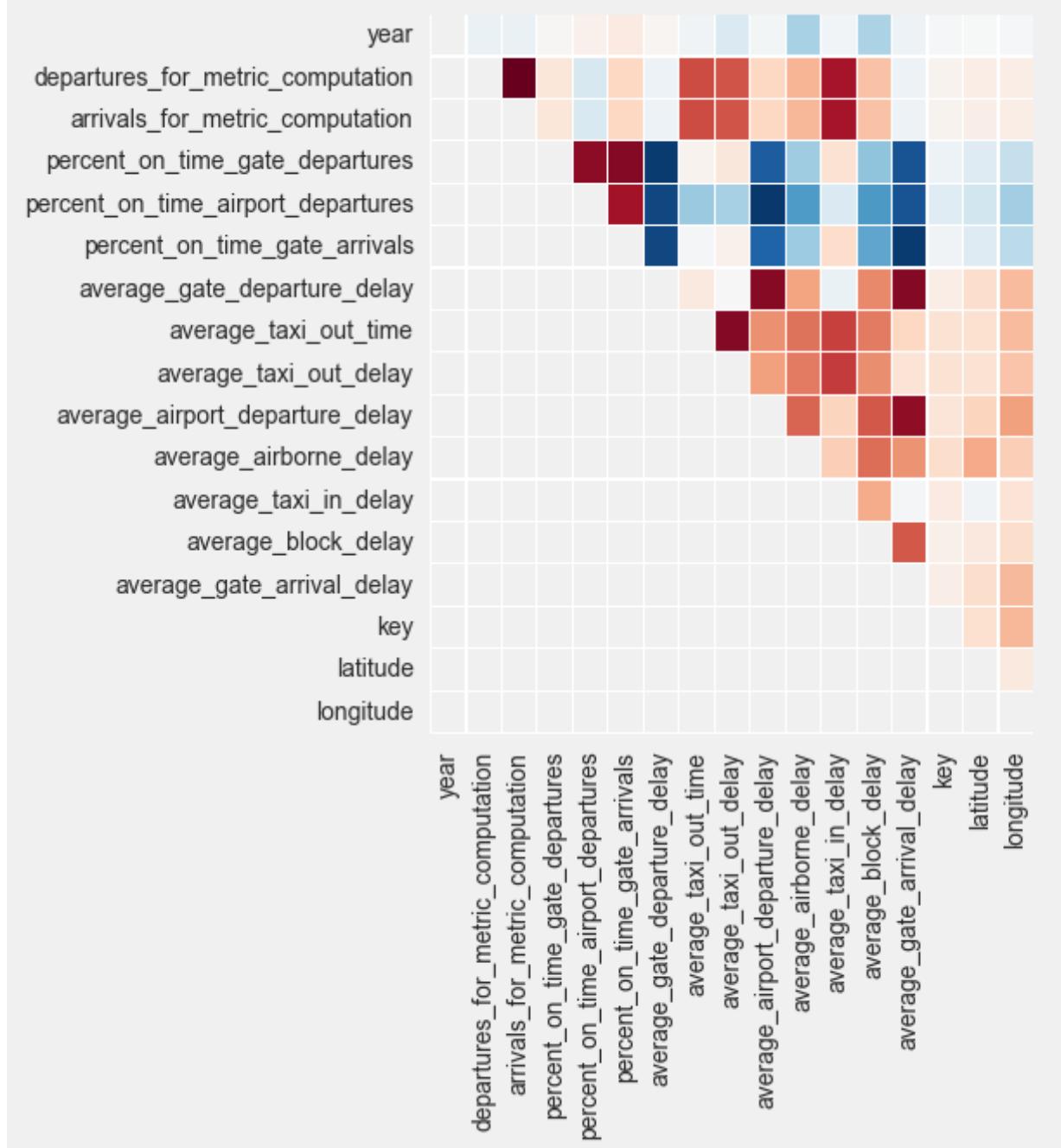
---



year  
departures\_for\_metric\_computation  
arrivals\_for\_metric\_computation  
percent\_on\_time\_gate\_departures  
percent\_on\_time\_airport\_departures  
percent\_on\_time\_gate\_arrivals  
average\_gate\_departure\_delay  
average\_taxi\_out\_time  
average\_taxi\_out\_delay

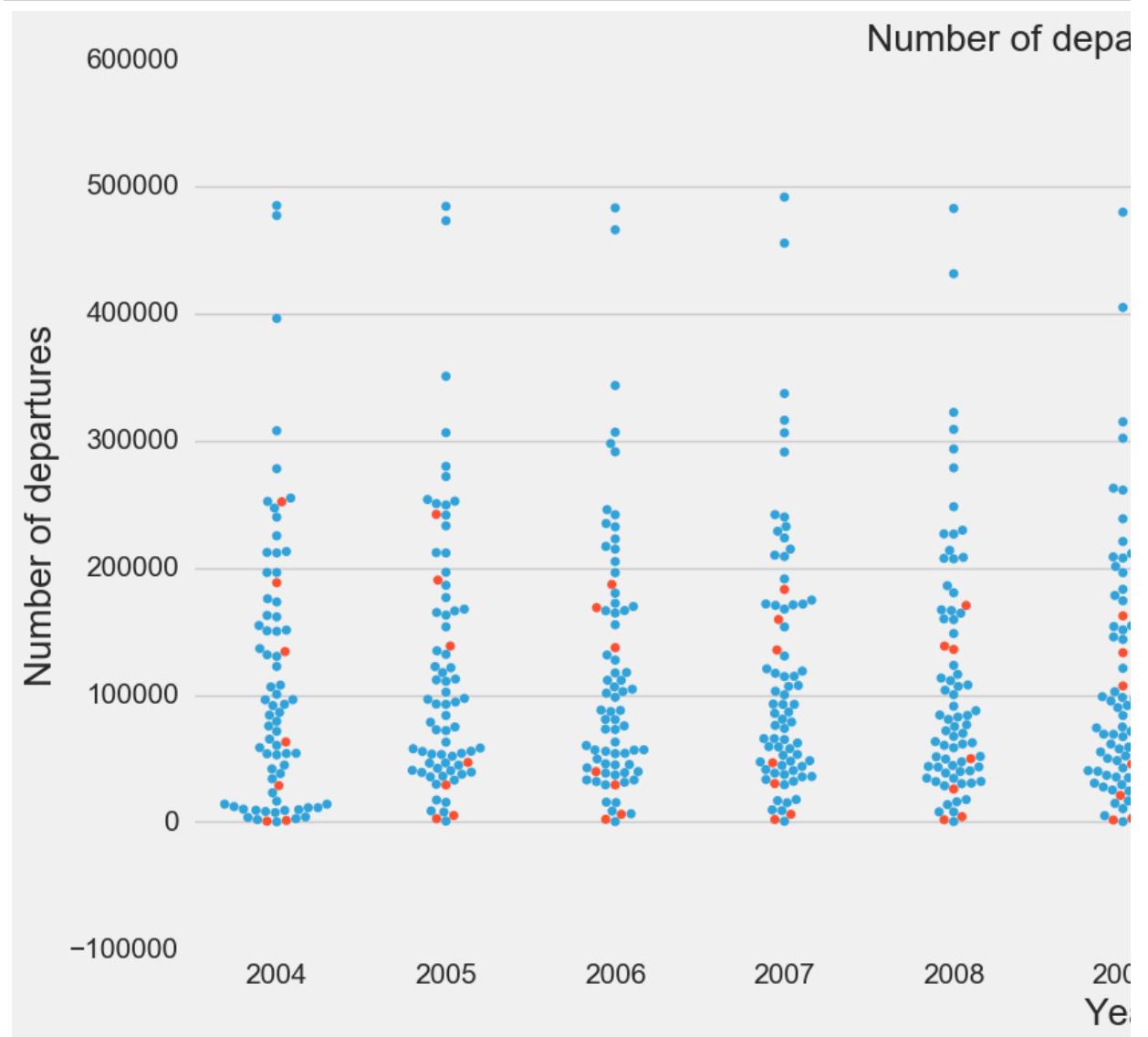
```
In [48]: # create a mask so that only one half of our heatmap shows up
# this takes out duplicates in the grid and also the spots where a variak
mask = np.zeros_like(df.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
plt.subplots(figsize=(5,5));
sns.heatmap(df.corr(), mask=mask.T, linewidths=0.1);
```

Last executed 2016-11-06 21:46:56 in 1.28s



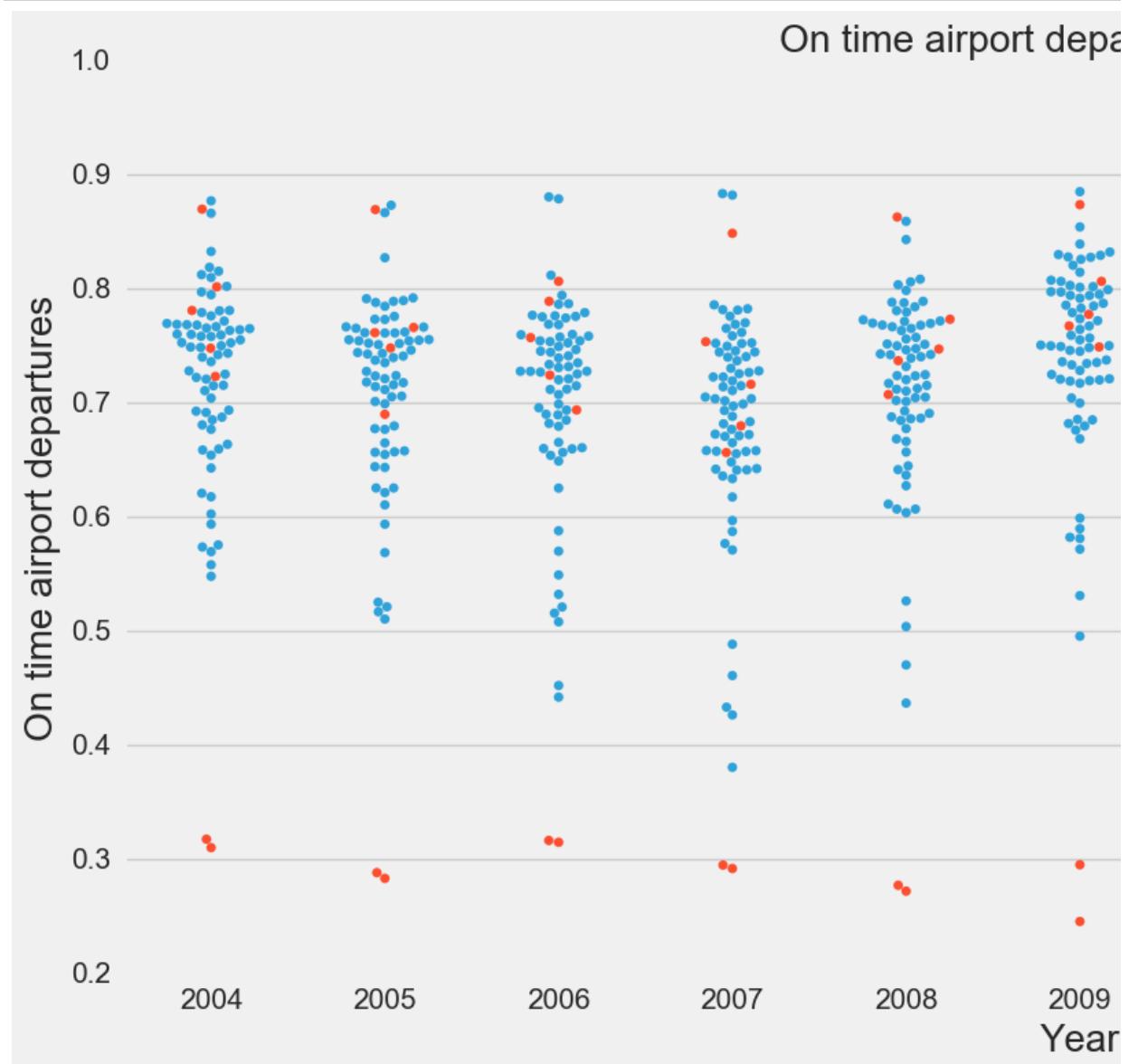
```
In [49]: # number of departures
plt.subplots(figsize=(16,8));
ax = sns.swarmplot(y="departures_for_metric_computation", x="year", hue="";
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("Number of departures", fontsize=20);
plt.xlabel("Year", fontsize=20);
plt.title("Number of departures per year");
```

Last executed 2016-11-06 21:46:57 in 2.10s

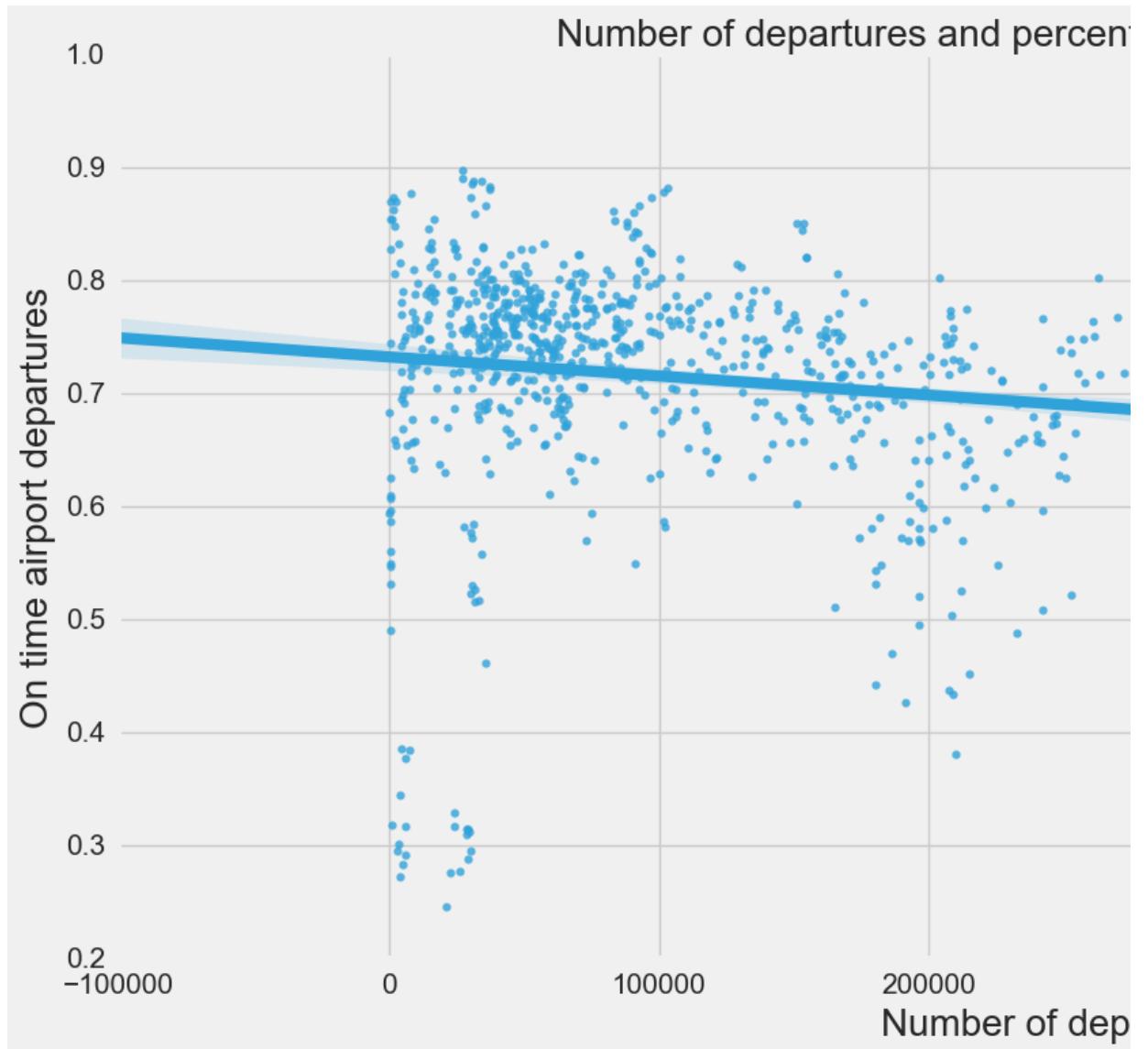


```
In [50]: # percentage of on time departures
plt.subplots(figsize=(16,8));
sns.swarmplot(y="percent_on_time_airport_departures", x="year", hue="ap_t
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("On time airport departures", fontsize=20);
plt.xlabel("Year", fontsize=20);
plt.title("On time airport departures per year");
```

Last executed 2016-11-06 21:46:59 in 1.99s

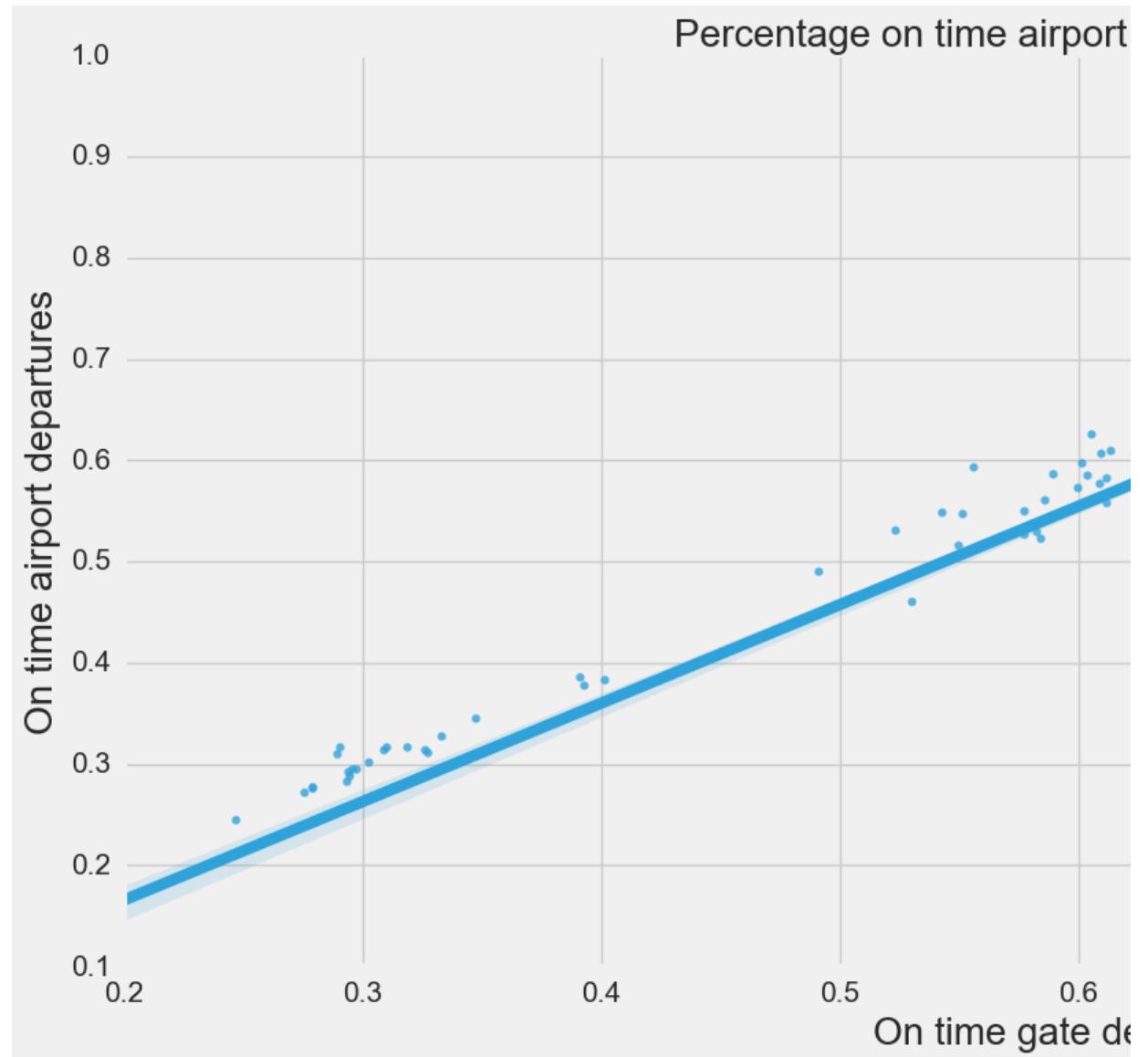


```
In [51]: # percentage of on time departures
plt.subplots(figsize=(16,8));
sns.regplot(y="percent_on_time_airport_departures", x="departures_for_met"
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("On time airport departures", fontsize=20);
plt.xlabel("Number of departures", fontsize=20);
plt.title("Number of departures and percentage of on time departures");
Last executed 2016-11-06 21:47:01 in 2.69s
```



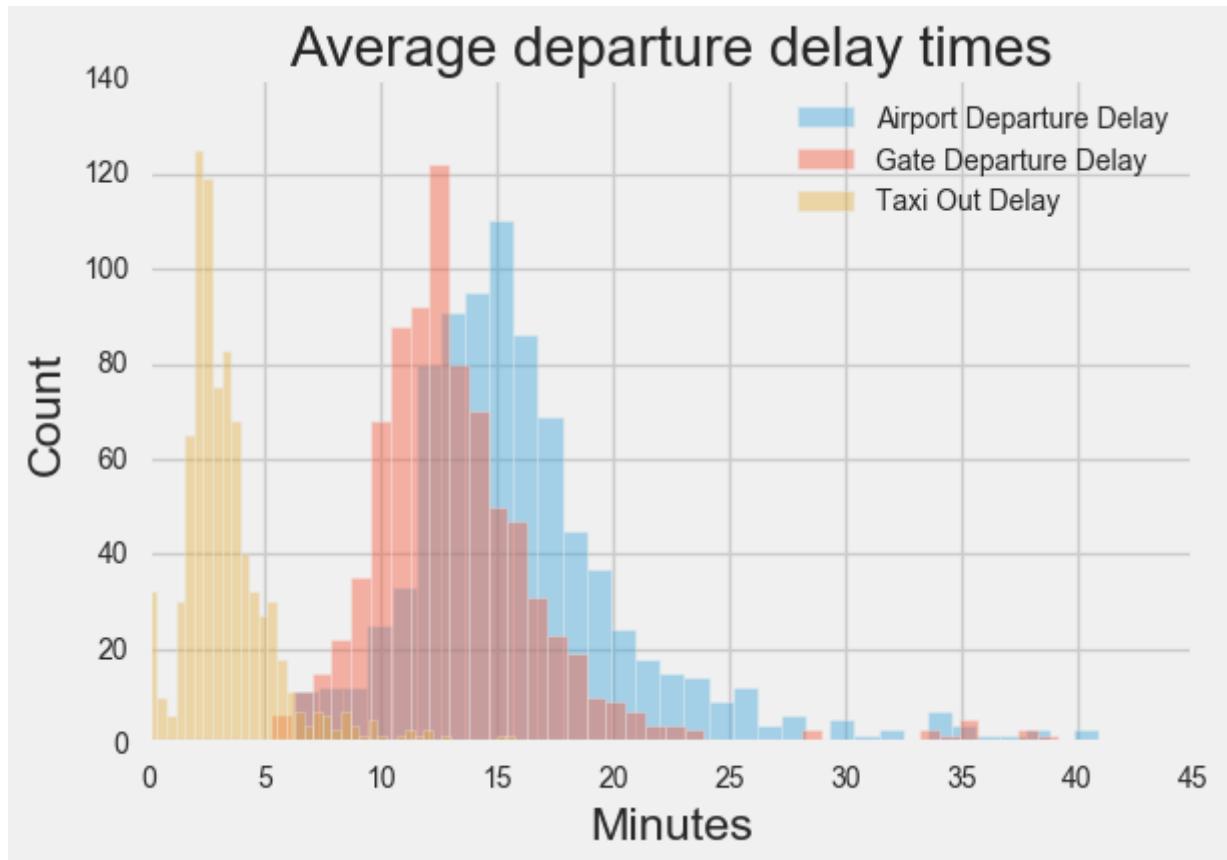
```
In [52]: # percentage of on time departures vs percentage of on time gate departure
plt.subplots(figsize=(16,8));
sns.regplot(y="percent_on_time_airport_departures", x="percent_on_time_gate_departures", fit_reg=True);
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("On time airport departures", fontsize=20);
plt.xlabel("On time gate departures", fontsize=20);
plt.title("Percentage on time airport and gate departures");
```

Last executed 2016-11-06 21:47:04 in 2.10s



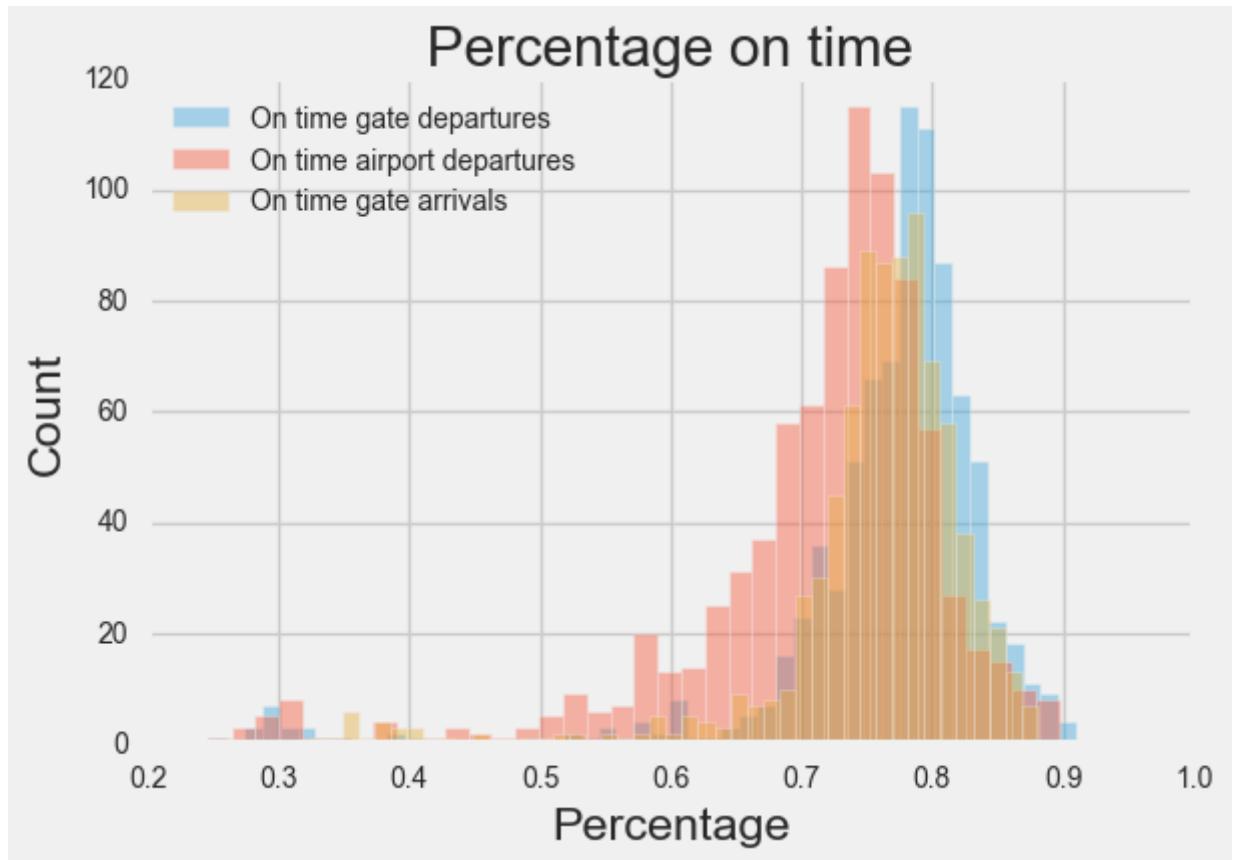
```
In [53]: sns.distplot(df["average_airport_departure_delay"], kde=False, label="Airport Departure Delay");
sns.distplot(df["average_gate_departure_delay"], kde=False, label="Gate Departure Delay");
sns.distplot(df["average_taxi_out_delay"], kde=False, label="Taxi Out Delay");
plt.legend(loc="best");
plt.title("Average departure delay times");
plt.ylabel("Count");
plt.xlabel("Minutes");
```

Last executed 2016-11-06 21:47:06 in 1.19s



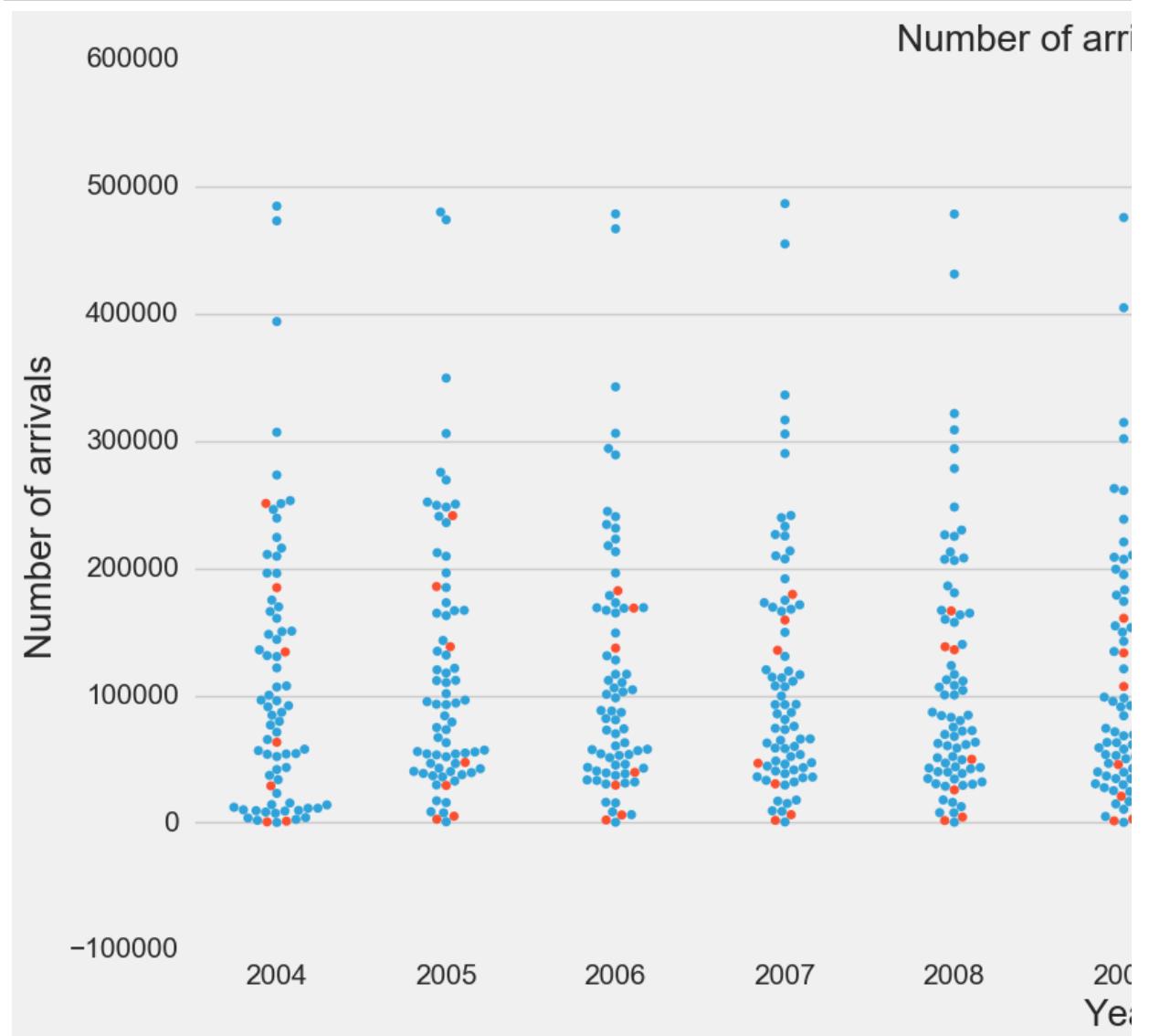
```
In [67]: sns.distplot(df["percent_on_time_gate_departures"], kde=False, label="On time gate departures");
sns.distplot(df["percent_on_time_airport_departures"], kde=False, label="On time airport departures");
sns.distplot(df["percent_on_time_gate_arrivals"], kde=False, label="On time gate arrivals");
plt.legend(loc="best");
plt.title("Percentage on time");
plt.ylabel("Count");
plt.xlabel("Percentage");
```

Last executed 2016-11-06 21:58:57 in 862ms



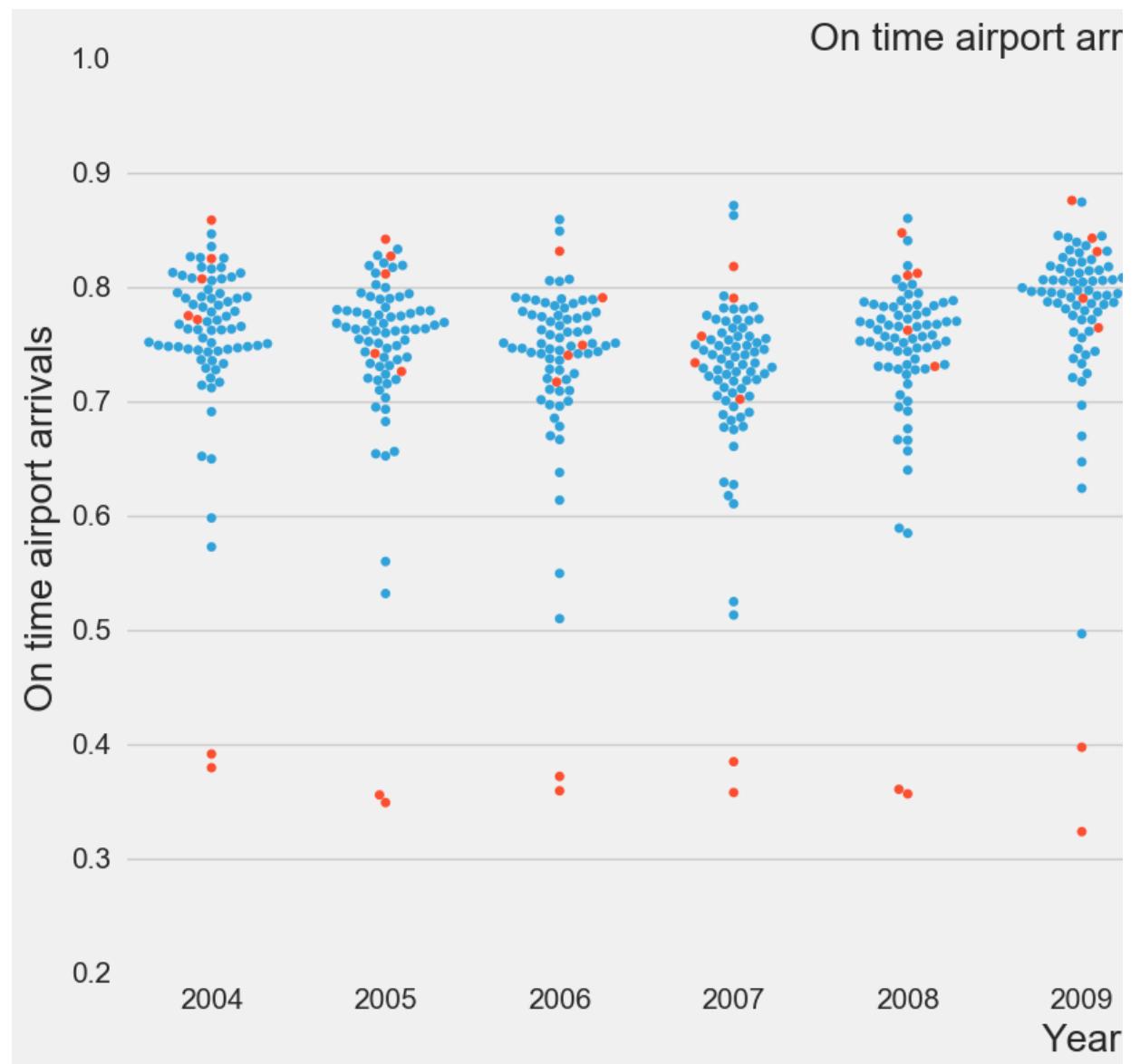
```
In [54]: # number of arrivals
plt.subplots(figsize=(16,8));
ax = sns.swarmplot(y="arrivals_for_metric_computation", x="year", hue="ap");
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("Number of arrivals", fontsize=20);
plt.xlabel("Year", fontsize=20);
plt.title("Number of arrivals per year");
```

Last executed 2016-11-06 21:47:07 in 1.99s



```
In [65]: # percentage of on time departures
plt.subplots(figsize=(16,8));
sns.swarmplot(y="percent_on_time_gate_arrivals", x="year", hue="ap_type",
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("On time airport arrivals", fontsize=20);
plt.xlabel("Year", fontsize=20);
plt.title("On time airport arrivals per year");
```

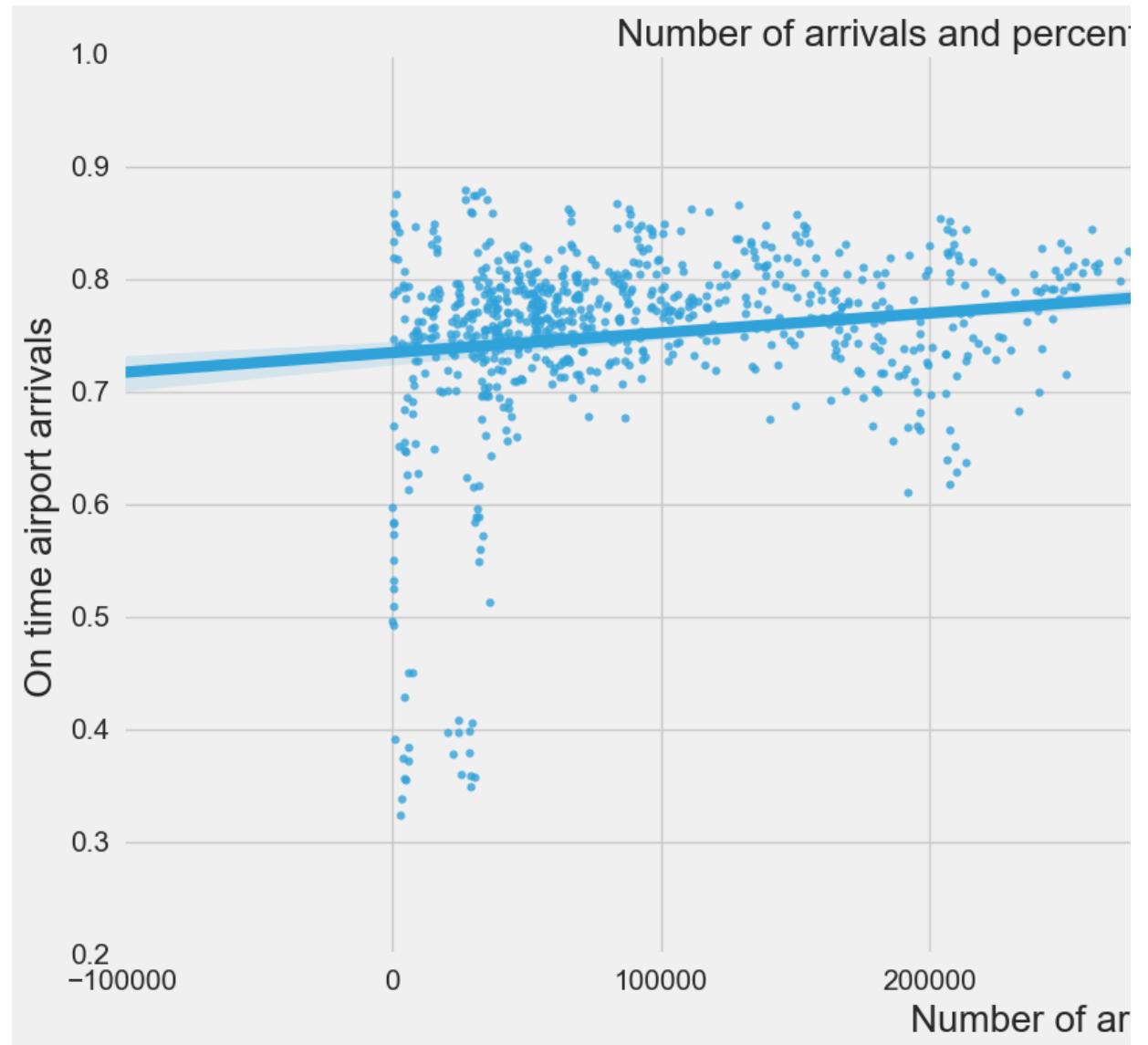
Last executed 2016-11-06 21:55:54 in 1.34s



```
In [55]: # percentage of on time arrivals
plt.subplots(figsize=(16,8));
sns.regplot(y="percent_on_time_gate_arrivals", x="arrivals_for_metric_com",
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("On time airport arrivals", fontsize=20);
plt.xlabel("Number of arrivals", fontsize=20);
plt.title("Number of arrivals and percentage of on time arrivals");

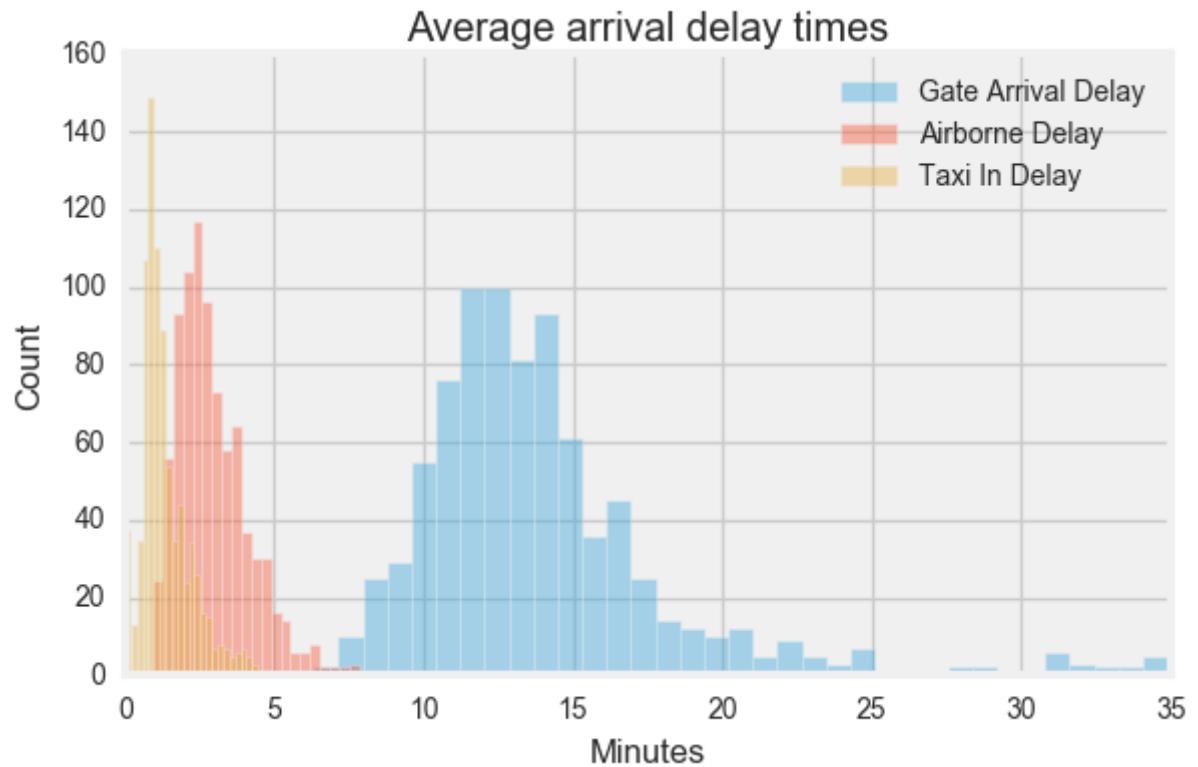
```

Last executed 2016-11-06 21:47:09 in 1.39s



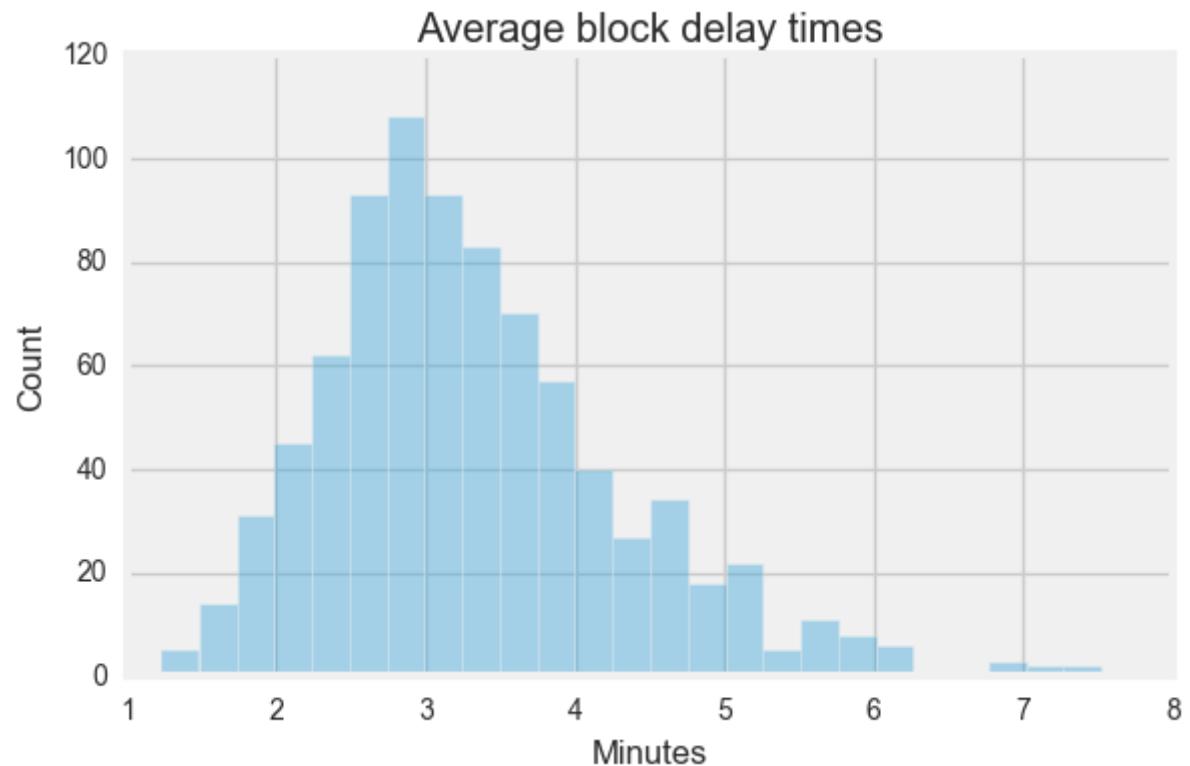
```
In [88]: sns.distplot(df["average_gate_arrival_delay"], kde=False, label="Gate Arrive  
sns.distplot(df["average_airborne_delay"], kde=False, label="Airborne Del  
sns.distplot(df["average_taxi_in_delay"], kde=False, label="Taxi In Delay  
plt.legend(loc="best");  
plt.title("Average arrival delay times");  
plt.ylabel("Count");  
plt.xlabel("Minutes");
```

Last executed 2016-11-06 22:23:15 in 985ms



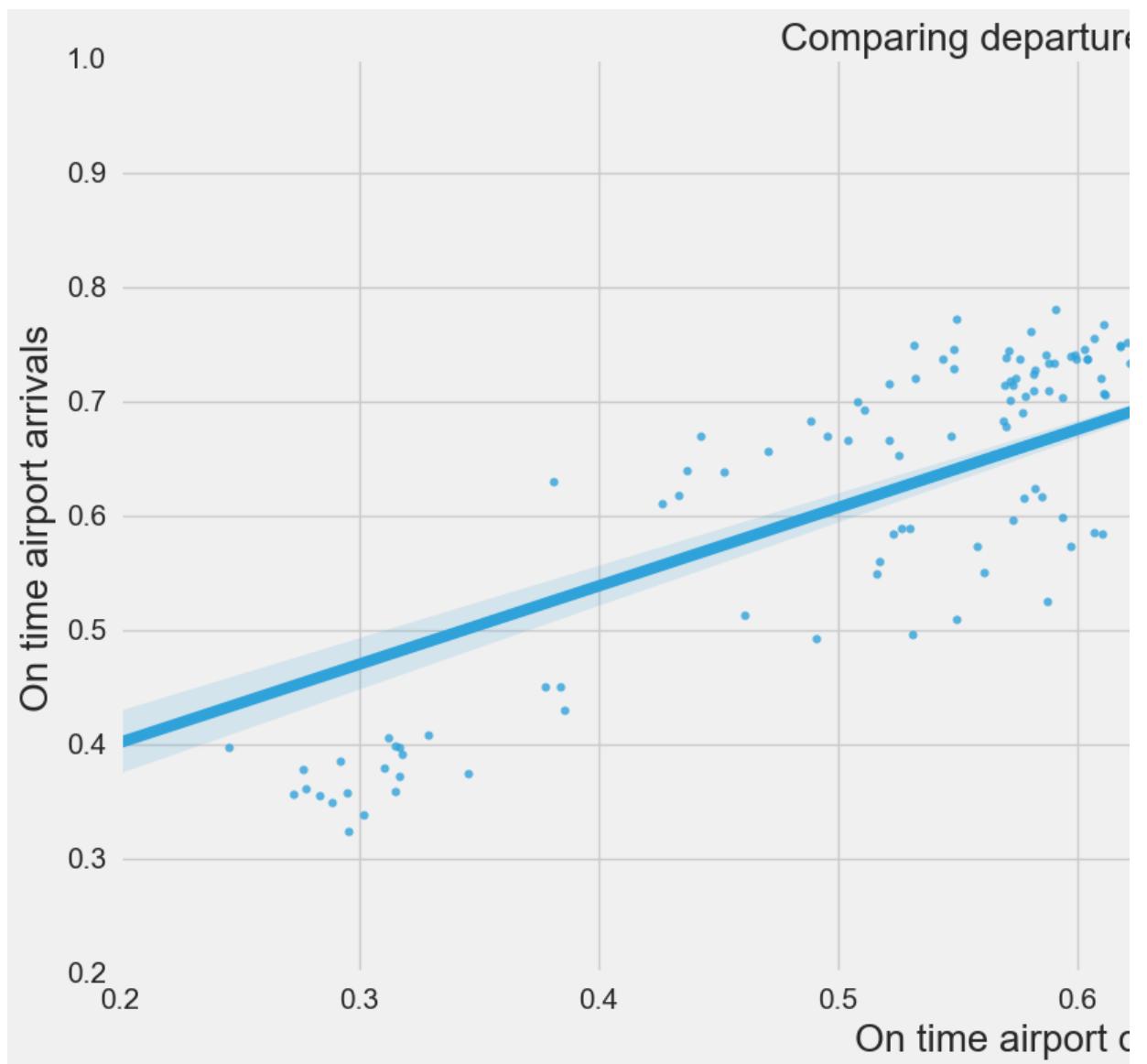
```
In [87]: sns.distplot(df["average_block_delay"], kde=False);
plt.title("Average block delay times");
plt.ylabel("Count");
plt.xlabel("Minutes");
```

Last executed 2016-11-06 22:22:50 in 698ms



```
In [64]: # percentage of on time arrivals
plt.subplots(figsize=(16,8));
sns.regplot(y="percent_on_time_gate_arrivals", x="percent_on_time_airport"
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("On time airport arrivals", fontsize=20);
plt.xlabel("On time airport departures", fontsize=20);
plt.title("Comparing departures and arrivals");
```

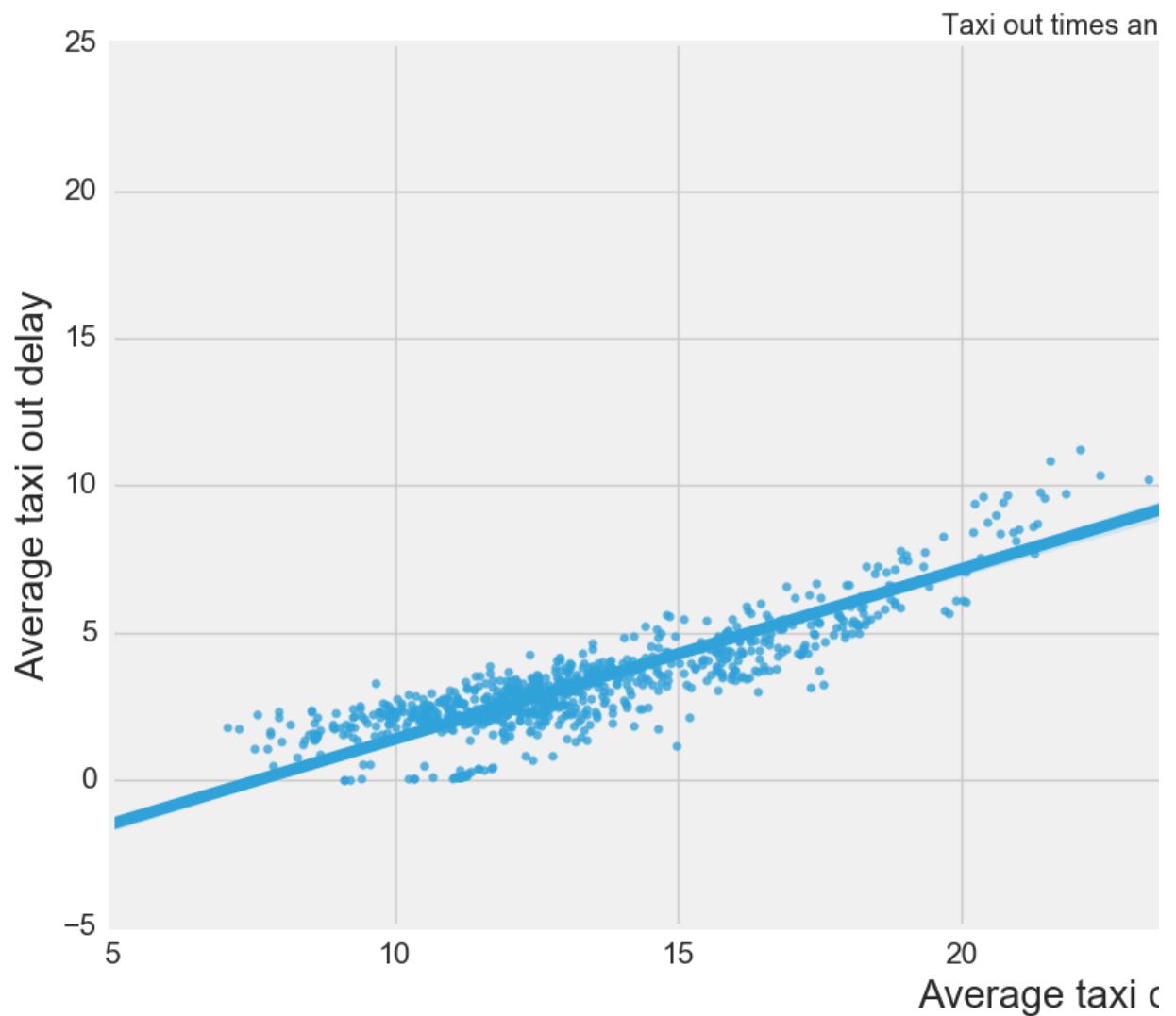
Last executed 2016-11-06 21:52:10 in 1.16s



```
# for year in df["year"].unique():
fig, ax = plt.subplots(figsize=(16,8));
sns.regplot(y="average_taxi_out_delay", x="average_taxi_out_time", data=df
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("Average taxi out delay", fontsize=20);
plt.xlabel("Average taxi out time", fontsize=20);
plt.title("Taxi out times and delays");
plt.show();
```

```
In [79]: plt.subplots(figsize=(16,8));
sns.regplot(y="average_taxi_out_delay", x="average_taxi_out_time", data=df
plt.xticks(fontsize=15);
plt.yticks(fontsize=15);
plt.ylabel("Average taxi out delay", fontsize=20);
plt.xlabel("Average taxi out time", fontsize=20);
plt.title("Taxi out times and delays");
```

Last executed 2016-11-06 22:15:14 in 1.46s



```
In [78]: le = preprocessing.LabelEncoder()
df["faa_num"] = le.fit_transform(df["faa_region"])
```

```
In [81]: df["faa_num"].unique()
```

Last executed 2016-11-06 22:16:29 in 8ms

```
Out[81]: array([7, 0, 6, 4, 2, 8, 3, 5, 1])
```

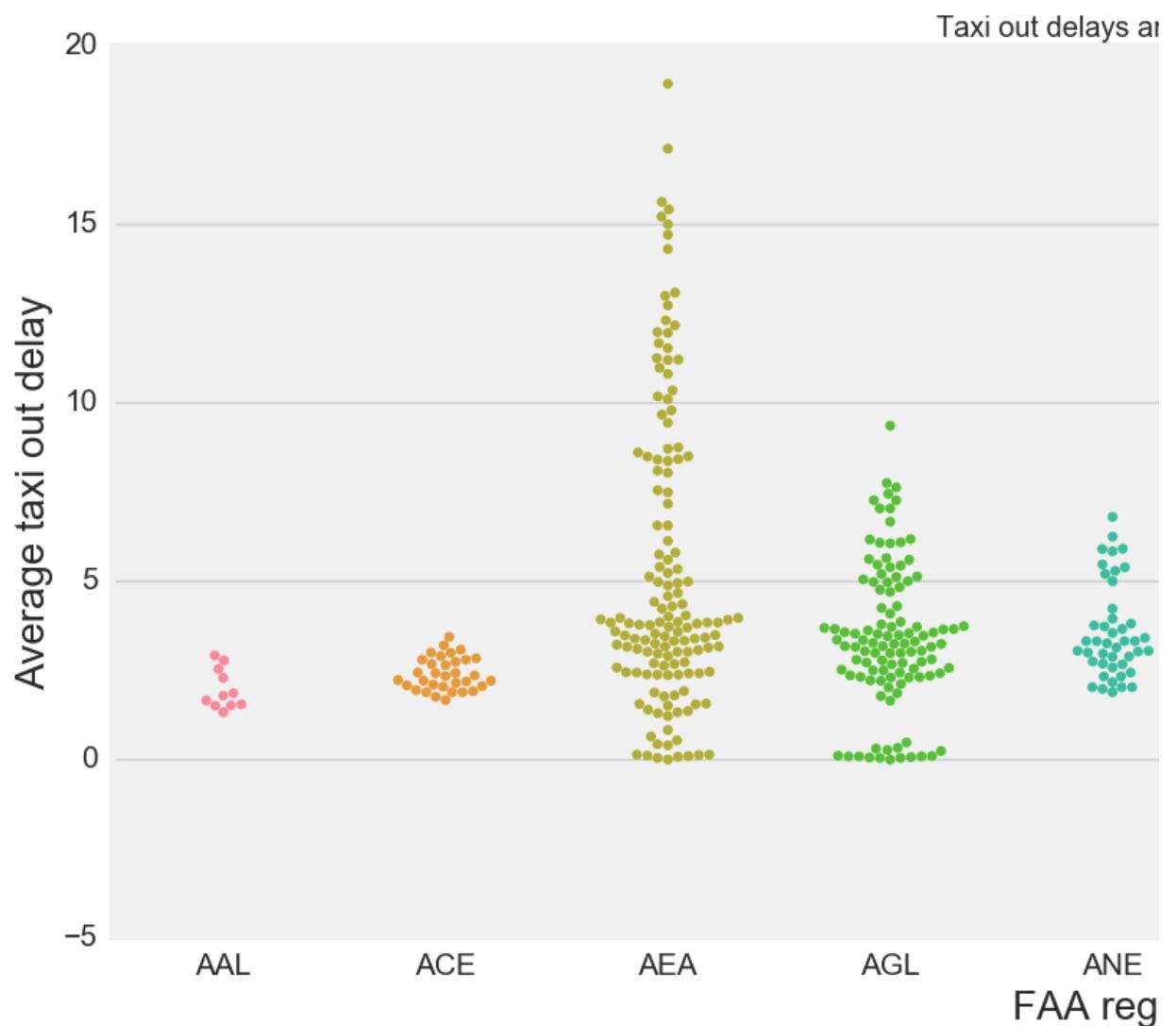
```
In [82]: le.inverse_transform(df["faa_num"].unique())
```

Last executed 2016-11-06 22:16:46 in 8ms

```
Out[82]: array(['ASW', 'AAL', 'ASO', 'ANE', 'AEA', 'AWP', 'AGL', 'ANM', 'ACE'], dt
```

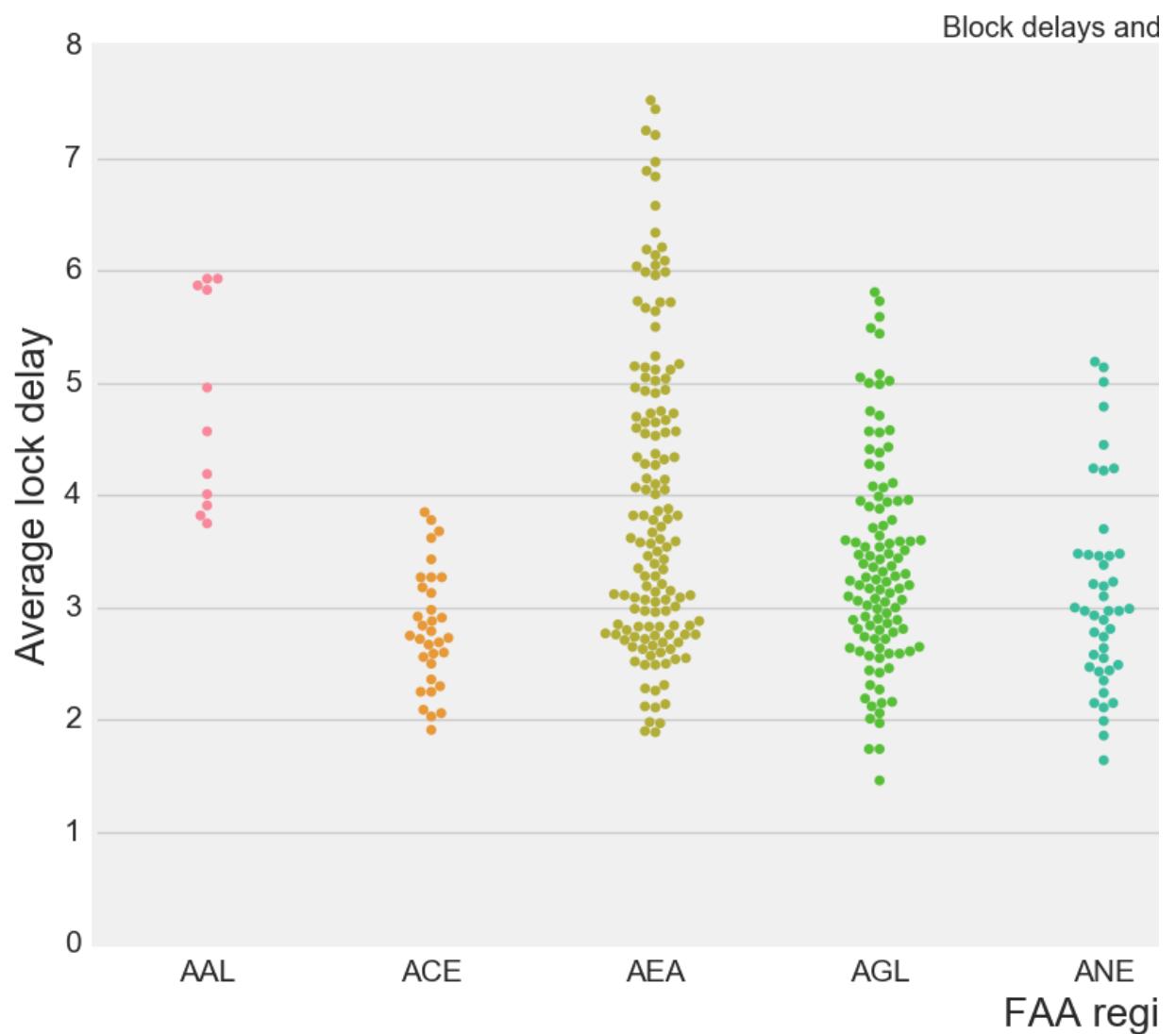
```
In [85]: plt.subplots(figsize=(16,8));
sns.swarmplot(y="average_taxi_out_delay", x="faa_num",data=df);
plt.xticks(df["faa_num"].unique(), le.inverse_transform(df["faa_num"].uni
plt.yticks(fontsize=15);
plt.ylabel("Average taxi out delay", fontsize=20);
plt.xlabel("FAA region", fontsize=20);
plt.title("Taxi out delays and regions");
```

Last executed 2016-11-06 22:18:57 in 2.12s



```
In [86]: plt.subplots(figsize=(16,8));
sns.swarmplot(y="average_block_delay", x="faa_num", data=df);
plt.xticks(df["faa_num"].unique(), le.inverse_transform(df["faa_num"].uni
plt.yticks(fontsize=15);
plt.ylabel("Average lock delay", fontsize=20);
plt.xlabel("FAA region", fontsize=20);
plt.title("Block delays and regions");
```

Last executed 2016-11-06 22:19:35 in 1.24s



#### 1.0.3.2 Are there any unique values?

In [102]:	df.describe()	locid	ap_name	alias	facility_type	faa_region	co
Last executed 2016-11-06 22:55:50 in 210ms							
Out[102]:	arrivals	average_gate_departure_delay	average_taxi_out_time	average_taxi_out_delay	ave		
	841.000000	841.000000	841.000000	841.000000	841.000000	841.000000	841.000000
	13.561403	13.703389	3.519750	16.200000	13.561403	16.200000	16.200000
	4.746563	3.792181	2.391278	5.300000	4.746563	5.300000	5.300000
	5.350000	7.040000	0.000000	6.200000	5.350000	6.200000	6.200000
	10.910000	11.260000	2.210000	12.000000	10.910000	12.000000	12.000000
	12.640000	12.810000	2.990000	15.000000	12.640000	15.000000	15.000000
	14.950000	15.650000	4.080000	17.000000	14.950000	17.000000	17.000000
	40.800000	35.760000	18.940000	40.000000	40.800000	40.000000	40.000000

In [106]: df.describe(include=["O"])

Last executed 2016-11-06 22:56:37 in 116ms

Out[106]:		airport	locid	ap_name	alias	facility_type	faa_region	co
	<b>count</b>	841	841	841	841	841	841	841
	<b>unique</b>	77	77	77	77	1	9	6!
	<b>top</b>	LAS	LAS	THEODORE FRANCIS GREEN STATE	Honolulu Int'l Airport	Airport	AWP	L
	<b>freq</b>	11	11	11	11	841	192	4

*facility\_type and boundary\_data\_available have only one value each, so we can drop them*

## 1.0.4 Part 3: Data Mining

### 1.0.4.1 3.1 Create Dummy Variables

In [121]: faa = ["alaskan", "central", "eastern", "great\_lakes", "new\_england", "nc", "southern", "southwest", "western\_pacific"]

Last executed 2016-11-06 23:34:29 in 5ms

In [122]: faa\_ab = ['AAL', 'ACE', 'AEA', 'AGL', 'ANE', 'ANM', 'ASO', 'ASW', 'AWP']

Last executed 2016-11-06 23:34:30 in 5ms

In [133]: faa\_d = {i:j for i, j in zip(faa\_ab, faa)}

Last executed 2016-11-06 23:36:23 in 6ms

```
In [134]: faa_d
```

```
Last executed 2016-11-06 23:36:23 in 7ms
```

```
Out[134]: {'AAL': 'alaskan',
'ACE': 'central',
'AEA': 'eastern',
'AGL': 'great_lakes',
'ANE': 'new_england',
'ANM': 'northwest_mountain',
'ASO': 'southern',
'ASW': 'southwest',
'AWP': 'western_pacific'}
```

```
In [140]: df2 = df.copy()
```

```
df2["faa_region"] = df2["faa_region"].map(faa_d)

df2 = pd.get_dummies(df2, columns=["faa_region", "state", "ap_type"])
del df2["faa_num"]
del df2["facility_type"]
del df2["boundary_data_available"]
del df2["city"]
del df2["county"]
del df2["alias"]
del df2["ap_name"]
del df2["ap_type_Public Use"]
del df2["faa_region_alaskan"]
```

```
Last executed 2016-11-06 23:37:10 in 42ms
```

```
In [141]: df2.columns
```

```
Last executed 2016-11-06 23:37:11 in 15ms
```

```
Out[141]: Index([u'airport', u'year', u'departures_for_metric_computation',
u'arrivals_for_metric_computation', u'percent_on_time_gate_departu
u'percent_on_time_airport_departures', u'percent_on_time_gate_arri
u'average_gate_departure_delay', u'average_taxi_out_time',
u'average_taxi_out_delay', u'average_airport_departure_delay',
u'average_airborne_delay', u'average_taxi_in_delay',
u'average_block_delay', u'average_gate_arrival_delay', u'key', u'l
u'latitude', u'longitude', u'faa_region_central', u'faa_region_eas
u'faa_region_great_lakes', u'faa_region_new_england',
u'faa_region_northwest_mountain', u'faa_region_southern',
u'faa_region_southwest', u'faa_region_western_pacific', u'state_AK
u'state_AL', u'state_AZ', u'state_CA', u'state_CO', u'state_CT',
u'state_DC', u'state_FL', u'state_GA', u'state_HI', u'state_IL',
u'state_IN', u'state_KY', u'state_LA', u'state_MA', u'state_MD',
u'state_MI', u'state_MN', u'state_MO', u'state_NC', u'state_NE',
u'state_NH', u'state_NJ', u'state_NM', u'state_NV', u'state_NY',
u'state_OH', u'state_OR', u'state_PA', u'state_PR', u'state_RI',
u'state_TN', u'state_TX', u'state_UT', u'state_WA', u'state_WI',
u'ap_type_Federalized/Commercial'],
dtype='object')
```

#### 1.0.4.2 3.2 Format and Clean the Data

In [2]: "s"

Last executed 2016-11-07 00:06:58 in 6ms

Out[2]: 's'

## 1.0.5 Part 4: Define the Data

### 1.0.5.1 4.1 Confirm that the dataset has a normal distribution. How can you tell?

In [ ]:

### 1.0.5.2 4.2 Find correlations in the data

In [ ]:

### 1.0.5.3 4.3 What is the value of understanding correlations before PCA?

Answer:

### 1.0.5.4 4.4 Validate your findings using statistical analysis

In [ ]:

### 1.0.5.5 4.5 How can you improve your overall analysis?

Answer:

## 1.0.6 Part 5: Perform a PCA

### 1.0.6.1 5.1 Conduct the PCA

In [ ]:

In [ ]:

## 1.0.7 Part 6: Additional Analysis

Include any other models you'd like to run here. These can include regressions, classifications, or

In [ ]:

In [ ]:

### 1.0.8 Part 7: Write an analysis plan of your findings

Create a writeup on the interpretation of findings including an executive summary with conclusions

Type *Markdown* and *LaTeX*:  $\alpha^2$

Which operational features are most correlated with delays?

What should the airport's next steps be?

### 1.0.9 Bonus: Copy your Database to AWS

Make sure to properly document all of the features of your dataset

In [ ]:

### 1.0.10 Bonus: Create a 3-Dimensional Plot of your new dataset with PC

In [ ]: