

Project: Taxi Data - Solution

Module 2 | Chapter 7 | Notebook 2

In this lesson you will be working with a relatively large data set representing taxi journeys in New York.

This data set contains 300000 journeys (rows). Data sets like this really reveal Python's strengths. For example, Excel can't work with that many rows, and would also take up a lot of memory to store less data, resulting in long wait times.

In this exercise you will:

- Prepare a data set independently
- Answer specific questions and create visualizations by yourself

We recommend that you do this project in this notebook, with relatively little instruction. If you get stuck, or if you need more help, you can always switch to [Project Assistance: Taxi Data](#). That notebook contains the same project, but with more instruction.

Scenario: A company based in New York wants to optimize the number of its taxis waiting at JFK international airport. The airport is located quite a distance from the main business area Manhattan. This means that it takes a long time for taxis waiting there to be available in other areas. On the other hand, customers arriving at the airport are take much longer journeys, and are more lucrative. How many taxis should the company keep waiting at JFK airport?

To help you answer the question, the company has given you all their data from 2016 in the file *2016_Yellow_Taxi_prepared.csv*.

You will need to complete the following steps:

1) Import, check and clean the data

2) Filter the data

3) What proportion of taxi journeys start at the airport in total? You can use the following formula to work this out:

$$\text{Proportion}_{\text{JFK}} = \frac{\text{journeys}_{\text{JFK}}}{\text{journeys}_{\text{all \ locations}}}$$

4) Where do people get taxis from in New York? Create a visualization showing the taxi journey starting points.

5) What proportion of taxi journeys start at the airport on each day of the week? Which day of the week shows the highest proportion, and which day shows the lowest? You can use the following formula to work this out:

$$\text{Proportion}_{\text{day of the week, JFK}} = \frac{\text{journeys}_{\text{day \ of \ the \ week, JFK}}}{\text{journeys}_{\text{all \ locations}}}$$

6) Create a visualization to show what proportion of journeys take place on each **day of the week**. Do this for journeys from the airport as well as for all locations to see if you can observe any differences. The following formulas represent these two visualizations:

```
\begin{equation*} \text{Proportion}_{\text{day of the week, all locations}} = \frac{\text{journeys}_{\text{day of the week, all locations}}}{\text{journeys}_{\text{all days, all locations}}} \end{equation*} \begin{equation*} \text{Proportion}_{\text{day of the week, JFK}} = \frac{\text{journeys}_{\text{day of the week, JFK}}}{\text{journeys}_{\text{all days, JFK}}} \end{equation*}
```

7) Create a visualization to show what proportion of total journeys are taken each **hour**. Do this for journeys from the airport as well as for all locations to see if you can observe any differences. The following formulas represent these two visualizations:

```
\begin{equation*} \text{Proportion}_{\text{hour, all locations}} = \frac{\text{journeys}_{\text{hour, all locations}}}{\text{journeys}_{\text{all hours, all locations}}} \end{equation*} \begin{equation*} \text{Proportion}_{\text{day of the week, JFK}} = \frac{\text{journeys}_{\text{hour, JFK}}}{\text{journeys}_{\text{all hours, JFK}}} \end{equation*}
```

8) Customize the visualizations

9) Make a recommendation

Good luck with the exercise and have fun!

We recommend that you follow the order that the tasks are listed in above.

You have a lot of space to experiment in this notebook. The steps above are repeated below as subheadings, and further down you will find example visualizations for the visual tasks. If you need additional code cells, you can add new ones by clicking on the + button in the top bar, next to the save button (see [A First Look at Python](#)).

 New_Cell

Try things out in this exercise. If you are having problems, have a look at the lesson entitled [Project Assistance: Taxi Data](#).

That notebook contains more suggestions and tips on how to proceed. It's up to you which notebook you use to complete this project. Remember, if you get stuck, search engines are your best friend! If that still doesn't help, you can also get in touch with StackFuel Support.

The data is structured as follows:

Column number	Column name	Type	Description
0	'pickup_weekday'	categorical (ordinal)	Day of the week when the journey started (Monday = 0, Sunday = 6).
1	'pickup_hour'	categorical (ordinal)	Hour when the journey started.
2	'pickup_longitude'	numerical	Longitude where the journey started.
3	'pickup_latitude'	numerical	Latitude where the journey started.
4	'dropoff_longitude'	numerical	Longitude where the journey ended.

Column number	Column name	Type	Description
5	'pickup_latitude'	numerical	Latitude where the journey ended.
6	'passenger_count'	categorical (ordinal)	Number of passengers in the car. This is manually recorded.
7	'trip_distance'	numerical	Journey distance in miles.
8	'fare_amount'	numerical	Amount on the meter based on duration and distance.
9	'tip_amount'	numerical	Tip given on card payments (0.00 if payment made in cash).
10	'tolls_amount'	numerical	Tolls incurred.
11	'payment_type'	categorical (nominal)	Payment type (1 = credit card, 2 = cash, 3 = no fee, 4 = dispute).

You are also given the following coordinates to determine which journeys start JFK airport:

Variable	Value	Description
jfk_max_lat	40.66018	Maximum pickup latitude for airport journeys
jfk_min_lat	40.62666	Minimum pickup latitude for airport journeys
jfk_max_long	-73.76599	Maximum pickup longitude for airport journeys
jfk_min_long	-73.80822	Minimum pickup longitude for airport journeys

New York City is determined approximately by the following coordinates:

Variable	Value	Description
nyc_max_lat	40.9176	Maximum latitude for New York City
nyc_min_lat	40.5774	Minimum latitude for New York City
nyc_max_long	-73.7004	Maximum longitude for New York City
nyc_min_long	-74.15	Minimum longitude for New York City

1) Importing and cleaning the data

For this step you will need to use skills you learned in the following lessons:

- [Preparing Data with pandas](#) (Chapter 1)
- [Exploring Data](#) (Chapter 2)
- [Project: Beer Ratings \(Data Cleaning\)](#) (Chapter 3))
- [Boolean Masking](#) (Chapter 4)
- [Project: Share Prices \(Preparing Data\)](#) (Chapter 6)

Important: The data set to be imported is relatively large with 300000 taxi journeys and therefore takes up quite a lot of resources. Since running code in the Data Lab runs via a server, it may take a little longer until the all the data is imported, depending on the current server load. Due to the file's size, please do not open this file in the Data Lab's editor! However, there is a simple way to significantly reduce the RAM requirement itself. The `pd.read_csv()` function offers the parameter `dtype`. This allows you to assign a data type manually to each column. By

default, integers are read in as `int64`, for example. This means that each number requires 64 bits in memory. The larger this value, the larger the integers that can be stored. For floating point numbers, the precision with which they are stored increases. However, our data is in a range that does not require such a large memory size. For example, if you import the numbers as `int32`, the `DataFrame` only needs half the amount of the RAM. You can use the following *dictionary* import the data set with less memory:

```
col_dtypes = {'pickup_weekday': 'int16',
              'pickup_hour': 'int16',
              'pickup_longitude': 'float32',
              'pickup_latitude': 'float32',
              'dropoff_longitude': 'float32',
              'dropoff_latitude': 'float32',
              'passenger_count': 'int16',
              'trip_distance': 'float32',
              'fare_amount': 'float32',
              'tip_amount': 'float32',
              'tolls_amount': 'float32',
              'payment_type': 'int16'}

df = pd.read_csv('2016_Yellow_Taxi_prepared.csv', dtype=col_dtypes)
```

In [1]:

```
# Solution:

# import pandas and numpy
import numpy as np
import pandas as pd

# read and check the data
col_dtypes = {'pickup_weekday': 'int16',
              'pickup_hour': 'int16',
              'pickup_longitude': 'float32',
              'pickup_latitude': 'float32',
              'dropoff_longitude': 'float32',
              'dropoff_latitude': 'float32',
              'passenger_count': 'int16',
              'trip_distance': 'float32',
              'fare_amount': 'float32',
              'tip_amount': 'float32',
              'tolls_amount': 'float32',
              'payment_type': 'int16'}

df = pd.read_csv('2016_Yellow_Taxi_prepared.csv', dtype=col_dtypes)
display(df.head(), '\n')

# check the dtypes
print(df.dtypes, '\n')

# turn categorical columns into categorical dtype (optional)
df.loc[:, 'pickup_weekday'] = df.loc[:, 'pickup_weekday'].astype('category')
df.loc[:, 'pickup_hour'] = df.loc[:, 'pickup_hour'].astype('category')
df.loc[:, 'passenger_count'] = df.loc[:, 'passenger_count'].astype('category')
df.loc[:, 'payment_type'] = df.loc[:, 'payment_type'].astype('category')

# check the dtypes
print(df.dtypes, '\n')

# check for missing data
print(df.isna().sum(), '\n')
```

```
# check for obviously wrong values
print('Check summary of longitude:')
print(df.loc[:, 'pickup_longitude'].describe(), '\n') # check summary of Longitudes
print('Check summary of latitude:')
print(df.loc[:, 'pickup_latitude'].describe(), '\n') # check summary of Latitudes
print('Check summary of trip distance:')
print(df.loc[:, 'trip_distance'].describe(), '\n') # check summary of trip_distance
```

	pickup_weekday	pickup_hour	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
0	3	19	-73.789970	40.646660	-74.005051	40.7480
1	5	3	-73.986237	40.746513	-73.996796	40.7425
2	4	20	-73.874634	40.773960	-73.959923	40.7628
3	5	2	-73.952477	40.772064	-73.949371	40.6751
4	4	21	-73.988281	40.764488	-73.996513	40.7532



'\n'

```
pickup_weekday      int16
pickup_hour          int16
pickup_longitude     float32
pickup_latitude      float32
dropoff_longitude    float32
dropoff_latitude     float32
passenger_count      int16
trip_distance        float32
fare_amount          float32
tip_amount           float32
tolls_amount         float32
payment_type         int16
dtype: object
```

```
pickup_weekday      category
pickup_hour          category
pickup_longitude     float32
pickup_latitude      float32
dropoff_longitude    float32
dropoff_latitude     float32
passenger_count      category
trip_distance        float32
fare_amount          float32
tip_amount           float32
tolls_amount         float32
payment_type         category
dtype: object
```

```
pickup_weekday      0
pickup_hour          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
trip_distance        0
fare_amount          0
tip_amount           0
tolls_amount         0
payment_type         0
dtype: int64
```

```

Check summary of longitude:
count    300000.000000
mean      -72.790047
std        9.285509
min      -130.829086
25%       -73.991676
50%       -73.981552
75%       -73.966751
max         0.000000
Name: pickup_longitude, dtype: float64

```

```

Check summary of latitude:
count    300000.000000
mean       40.098637
std        5.115375
min         0.000000
25%        40.736401
50%        40.753294
75%        40.767811
max        48.857597
Name: pickup_latitude, dtype: float64

```

```

Check summary of trip distance:
count    300000.000000
mean       2.847915
std        3.532295
min         0.000000
25%         1.000000
50%         1.660000
75%         3.040000
max       179.300003
Name: trip_distance, dtype: float64

```

In [2]:

```

# Solution:

# Look at potentially wrong values
mask1 = (df.loc[:, 'pickup_longitude'] == 0) # select Longitudes with value 0
print('Number of wrong values:', mask1.sum()) # how many are there?
## -> will be removed with jfk mask

mask2 = (df.loc[:, 'pickup_longitude'] < -100) # select Longitudes with extreme values
print('Number of wrong values:', mask2.sum()) # how many are there?
## -> will be removed with jfk mask

mask3 = (df.loc[:, 'trip_distance'] <= 0) # select trip_distances with zero distance
print('Number of wrong values:', mask3.sum()) # how many are there?
display(df.loc[mask3, :]) # take a look into them
## -> relevant data points have no common structure, seem to be realistic
## maybe distances between 0 and 0.01 miles? -> keeping in dataset as other information

print(df.shape) # check DataFrame size (same?)

```

```

Number of wrong values: 4803
Number of wrong values: 1
Number of wrong values: 1792

```

	pickup_weekday	pickup_hour	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
162	2	11	-73.993385	40.742016	-73.993393	40.742016
372	4	16	-73.983620	40.756027	0.000000	40.756027
446	3	22	-74.037781	40.721294	-74.037788	40.721294

	pickup_weekday	pickup_hour	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff
509	0	12	-73.939072	40.805225	-73.940498	4
544	0	5	-73.945328	40.751686	-73.945328	4
...
298387	4	19	-73.983170	40.730667	-73.983170	4
298643	2	14	-73.863503	40.769787	0.000000	
299549	3	17	-73.952812	40.748260	-73.952812	4
299621	4	13	0.000000	0.000000	-73.976379	4
299774	0	0	-73.984192	40.764008	-73.984802	4

1792 rows × 12 columns

(300000, 12)

Congratulations: Now the data should be relatively clean and you can start working with it.

2) Selecting data

In this step you will use techniques you learned in the following lesson:

- [Boolean Masking](#) (Chapter 4)

In [3]:

```
# Solution:

#define variables for coordinates
jfk_max_lat = 40.66018 # define maximal Latitude
jfk_min_lat = 40.62666 # define minimal Latitude
jfk_max_long = -73.76599 # define maximal Longitude
jfk_min_long = -73.80822 # define minimal Longitude

#create masks and select the data
mask_lat_jfk = (df.loc[:, 'pickup_latitude'] >= jfk_min_lat) & (df.loc[:, 'pickup_la
mask_long_jfk = (df.loc[:, 'pickup_longitude'] >= jfk_min_long) & (df.loc[:, 'pickup
mask_jfk_pickup = (mask_lat_jfk & mask_long_jfk) # combine both masks
print(mask_jfk_pickup.sum()) # how many values are in the airport data?

df_jfk = df.loc[mask_jfk_pickup, :] # define new DataFrame for airport data
```

5790

Congratulations: Now you have extracted the airport data and can start working on the task itself.

3) Proportion of taxis from the airport

For this step you should answer the following question:

What proportion of all taxi journeys start at the airport?

$$\text{Proportion}_{\{\text{JFK}\}} = \frac{\text{journeys}_{\{\text{JFK}\}}}{\text{journeys}_{\{\text{all locations}\}}}$$

In [4]:

```
# Solution:

# First Question: What is the proportion of trips starting from JFK?
print(len(df_jfk) / len(df)) # use the ratio of their lengths either with len
#print(df_jfk.shape[0] / df.shape[0]) # or with shape
```

0.0193

Congratulations: Now the taxi company knows what proportion of journeys start at the airport.


4) Visualizing the starting points

Create a visualization showing the New York taxi journey starting points.

In this step you will use techniques you learned in the following lesson:

- [Visualizing Data with pandas](#) (Chapter 1)
- [Visualizing Data Distributions with Histograms](#) (Chapter 2)
- [Visualizing Correlations with Scatterplots](#) (Chapter 2)
- [Visualizing Categories](#) (Chapter 2)
- [Combining Visualizations with matplotlib](#) (Chapter 2)
- [Customizing Visualizations with matplotlib](#) (Chapter 2)
- [Project: Share Prices Customizing the Visualization](#) (Chapter 6)

Here is an example of how your visualization would look like:

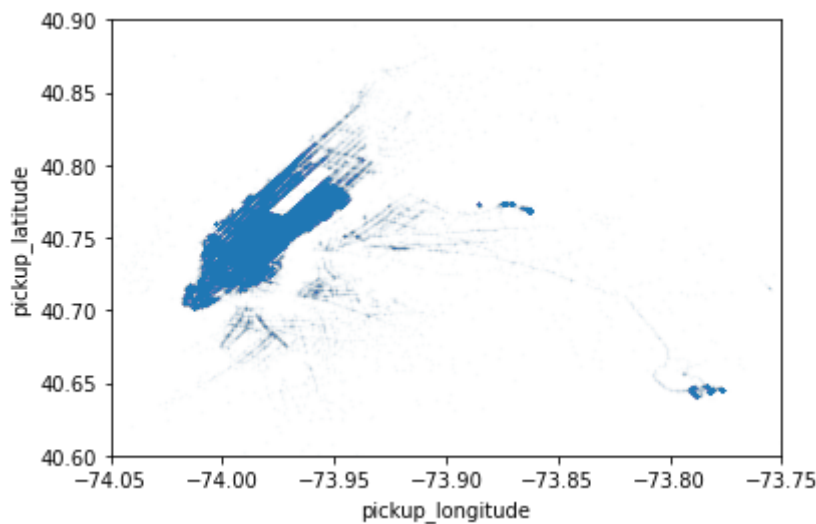
 ratio_weekdays

In [5]:

```
# Solution:

# Pickup visualisation
%matplotlib inline
import matplotlib.pyplot as plt

fig, ax= plt.subplots()
df.plot(kind='scatter',
        x='pickup_longitude',
        y='pickup_latitude',
        alpha=0.05,
        s=0.05,
        ax=ax); # draw scatter plot of coordinates with small dots (s) which are ne
ax.set(xlim=[-74.05, -73.75],
       ylim=[40.60, 40.90]); # adjust axis limits
```

Congratulations: Now the taxi company has an overview of where its taxis begin their journeys.

5) Proportion of airport taxis on each day

For this step you should answer the following question:

What proportion of taxi journeys start at the airport on each day of the week?

$$\text{Proportion}_{\text{day of the week, JFK}} = \frac{\text{journeys}_{\text{day of the week, JFK}}}{\text{journeys}_{\text{day of the week, all locations}}}$$

You will use skills you learned in the following lessons:

- [Importing Data with pandas](#) (Chapter 1)
- [Exploring Categories](#) (Chapter 2)

In [6]:

```
# Solution:

# What is the ratio of trips starting from JFK on each weekday?
counts_days_jfk = pd.crosstab(index=df_jfk.loc[:, 'pickup_weekday'],
                              columns='count') # select the number of trips per wee
counts_days_all = pd.crosstab(index=df.loc[:, 'pickup_weekday'],
                              columns='count') # select the number of trips per wee
print(counts_days_jfk / counts_days_all) # print their ratio

print((counts_days_jfk / counts_days_all).mean()) # check whether the mean proporti
```

```
col_0      count
pickup_weekday
0          0.026142
1          0.020047
2          0.017100
3          0.018116
4          0.018723
5          0.014440
6          0.020485
col_0
count      0.019293
dtype: float64
```

Congratulations: Now the taxi company knows what proportion of their fleet starts at the airport on each day of the week.

6) Proportion of journeys on each day of the week from all locations and those starting from the airport

For this step you should do the following:

6) Create a visualization to show what proportion of journeys are taken on each day of the week. Do this for journeys from the airport as well as for all journeys to see if you can observe any differences.

$$\begin{aligned} \text{Proportion}_{\text{day of the week, all locations}} &= \frac{\text{journeys}_{\text{day of the week, all locations}}}{\text{journeys}_{\text{all days, all locations}}} \\ \text{Proportion}_{\text{day of the week, JFK}} &= \frac{\text{journeys}_{\text{day of the week, JFK}}}{\text{journeys}_{\text{all days, JFK}}} \end{aligned}$$

You can use skills you learned in the following lessons:

- [Exploring Categories \(Chapter 2\)](#)
- [Visualizing Categories \(Chapter 2\)](#)

Here is an example of how your visualization could like:

 ratio_weekdays

In [7]:

```
# Solution:

# Which weekdays have the most trips?
counts_days_all = pd.crosstab(index=df.loc[:, 'pickup_weekday'],
                              columns='count') # count the number of trips on every
counts_days_jfk = pd.crosstab(index=df_jfk.loc[:, 'pickup_weekday'],
                              columns='count') # count the number of trips at every

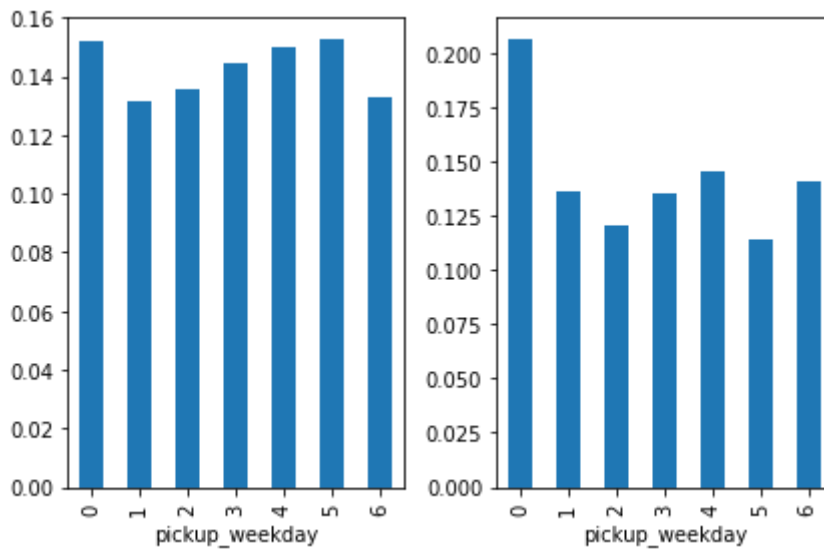
fig, axs = plt.subplots(ncols=2) # make a plot with 2 subplots and share the y axis

counts_days_ratio_all = counts_days_all.loc[:, 'count'] / len(df) # calculate the r
counts_days_ratio_all = counts_days_ratio_all.sort_index()

counts_days_ratio_all.plot(kind='bar',
                           ax=axs[0]); # make a bar chart
counts_days_ratio_jfk = counts_days_jfk.loc[:, 'count'] / len(df_jfk) # calculate t
counts_days_ratio_jfk = counts_days_ratio_jfk.sort_index()

counts_days_ratio_jfk.plot(kind='bar',
                           ax=axs[1]); # make a bar chart

fig.tight_layout()
```



Congratulations: Now the taxi company knows what proportion of taxi journeys start at the airport on each day of the week.

7) Proportion each hour for all journeys and journeys from the airport

In this step you should do the following:

Create a visualization to show what proportion of journeys are taken at different times of day.

Do this for journeys from the airport as well as for all journeys to see if you can observe any differences.

$$\text{Proportion}_{\text{hour, all locations}} = \frac{\text{journeys}_{\text{hour, all locations}}}{\text{journeys}_{\text{all hours, all locations}}}$$

$$\text{Proportion}_{\text{day of the week, JFK}} = \frac{\text{journeys}_{\text{hour, JFK}}}{\text{journeys}_{\text{all hours, JFK}}}$$

You can use skills you learned in the following lessons:

- [Visualizing Data with pandas \(Chapter 1\)](#)
- [Exploring Categories \(Chapter 2\)](#)
- [Visualizing Categories \(Chapter 2\)](#)

Here is an example of what your visualization could look like:

 ratio_hours

In [8]:

```
# Solution:

# Which times have the most trips?
counts_hours_all = pd.crosstab(index=df.loc[:, 'pickup_hour'],
                               columns='count') # count the number of trips at every hour
counts_hours_jfk = pd.crosstab(index=df_jfk.loc[:, 'pickup_hour'],
                               columns='count') # count the number of trips at every hour

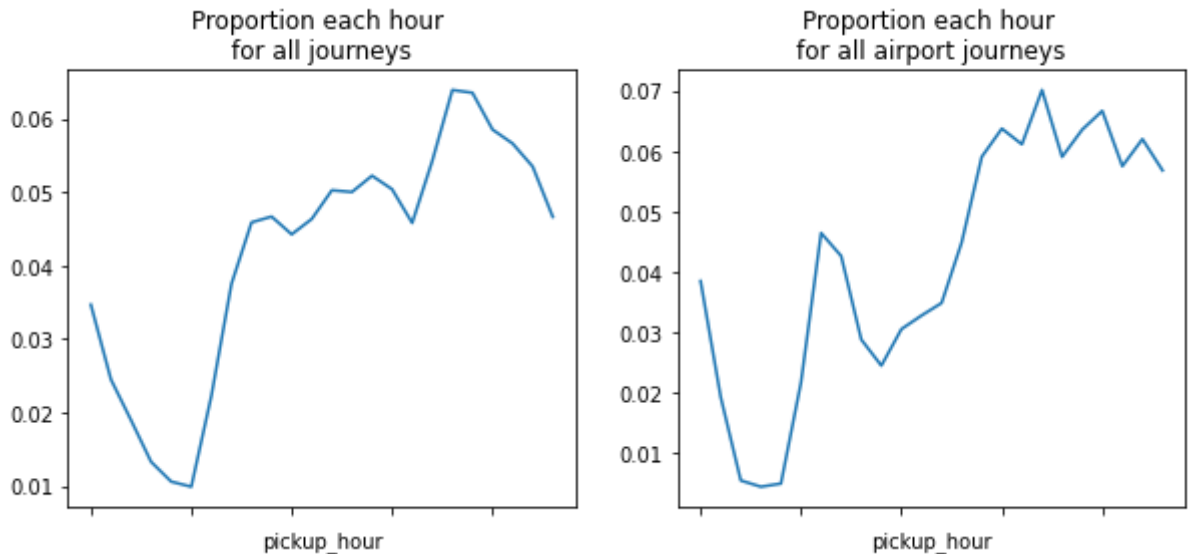
fig, axs = plt.subplots(ncols=2, figsize=[10, 4]) # make a plot with 2 subplots and 2 axes

counts_hours_ratio_all = counts_hours_all.loc[:, 'count'] / len(df) # calculate the proportion of trips at every hour
counts_hours_ratio_all = counts_hours_ratio_all.sort_index()
counts_hours_ratio_all.plot(ax=axs[0],
```

```

title='Proportion each hour \nfor all journeys'); # make
counts_hours_ratio_jfk = counts_hours_jfk.loc[:, 'count'] / len(df_jfk) # calculate
counts_hours_ratio_jfk = counts_hours_ratio_jfk.sort_index()
counts_hours_ratio_jfk.plot(ax=axis[1],
                             title='Proportion each hour \nfor all airport journeys')

```



Congratulations: Now you have a good overview of how taxi journeys from the airport compare with journeys from all locations at different hours of the day.

8) Customizing the visualizations

Now you can improve the appearance of the visualizations.

You can look back at the following lessons for inspiration:

- [Combining Visualizations with matplotlib](#) (Chapter 2)
- [Customizing Visualizations with matplotlib](#) (Chapter 2)
- [Project: Share Prices \(Customizing the Visualization\)](#) (Chapter 6)

Play around with different options until you are satisfied with the result.

In [10]:

```

# Solution:

#####
# pickup locations
plt.style.use('seaborn') # use your favorite style
fig, ax = plt.subplots(figsize=[6, 6]) # initialise Figure and Axes
df.plot(kind='scatter',
         x='pickup_longitude',
         y='pickup_latitude',
         alpha=0.05,
         s=0.05,
         ax=ax) # draw pickup-coordinates with scatterplot, dots are small (s) and q
ax.set(xlim=[-74.05, -73.75],
      ylim=[40.60, 40.90],
      xlabel=' ',
      ylabel=' ',
      title='Taxi pick-ups in New York') # adjust Axes
ax.annotate(s='JFK airport',
           xy=[-73.79, 40.642],
           xytext=[-73.88, 40.6],

```

```

        arrowprops=dict(facecolor='black')) # point out JFK
ax.grid(False, axis='x') # remove horizontal grid lines
ax.grid(False, axis='y') # remove vertical grid lines
ax.set_xticks([]) # remove x-tick-labels
ax.set_yticks([]) # remove y-tick-labels
ax.set_facecolor('whitesmoke') # set Axes background colour

#####
#weekdays
plt.style.use('fivethirtyeight') # use your favorite style
#counts_days_all = pd.crosstab(index=df.loc[:, 'pickup_weekday'], columns='count')
#counts_days_all = counts_days_all.sort_index()
#counts_days_jfk = pd.crosstab(index=df_jfk.loc[:, 'pickup_weekday'], columns='count')
#counts_days_jfk = counts_days_jfk.sort_index()

fig, axs = plt.subplots(ncols=2, figsize=[16, 5], sharey=True, sharex=True) # make

my_colors = ['blue', 'red', 'orange', 'green', 'grey', 'purple', 'darkblue'] # creat

#counts_days_ratio_all = counts_all.loc[:, 'count'] / len(df) # calculate the ratio
#counts_days_ratio_all = counts_days_ratio_all.sort_index()
counts_days_ratio_all.plot(kind='bar',
                           ax=axs[0],
                           color=my_colors,
                           title='Proportion of journeys per week day') # make a ba

#counts_days_ratio_jfk = counts_jfk.loc[:, 'count'] / len(df_jfk) # calculate the r
#counts_hours_ratio_jfk = counts_hours_ratio_jfk.sort_index()
counts_days_ratio_jfk.plot(kind='bar',
                           ax=axs[1],
                           color=my_colors,
                           title='Proportion of airport journeys per week day') # m

axs[0].set(xlabel='Day of the week',
           ylabel='Proportion of journeys',
           ylim=[0.0, 0.22]) # set labels and y-limits for readability
axs[1].set(xlabel='Day of the week',
           ylabel='Proportion of journeys',
           ylim=[0.0, 0.22]) # set labels and y-limits for readability

axs[0].set_xticklabels(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'], rotation=0)
axs[1].set_xticklabels(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'], rotation=0)

plt.gca().set_yticklabels(['{:.0f}%'.format(x*100) for x in plt.gca().get_yticks()])

fig.tight_layout()

#####
# Hours

plt.style.use('fivethirtyeight') # use your favorite style
#counts_hours_all = pd.crosstab(index=df.loc[:, 'pickup_hour'],
#                               columns='count') # count the number of trips at every hou
#
#counts_hours_jfk = pd.crosstab(index=df_jfk.loc[:, 'pickup_hour'],
#                               columns='count') # count the number of trips at every hou

fig, axs= plt.subplots(ncols=2, figsize=[16,6], sharey=True, sharex=True) # make a

#counts_hours_ratio_all = counts_hours_all.loc[:, 'count'] / len(df) # calculate th
#counts_hours_ratio_all = counts_hours_ratio_all.sort_index()
counts_hours_ratio_all.plot(ax=axs[0], title='Proportion each hour for all journeys'

#counts_hours_ratio_jfk = counts_hours_jfk.loc[:, 'count'] / len(df_jfk) # calculat
#counts_hours_ratio_jfk = counts_hours_ratio_jfk.sort_index()

```

```

counts_hours_ratio_jfk.plot(ax=axis[1], title='Proportion each hour for all airport j
axis[0].set(xlabel='hour',
            ylabel='Proportion of journeys',
            ylim=[0.0, 0.08]); # set labels and limits for readability
axis[1].set(xlabel='Hour',
            ylabel='Proportion of journeys',
            ylim=[0.0, 0.08]); # set labels and limits for readability

axis[0].set_xticklabels(['', '0', '5', '10', '15', '20']);

plt.gca().set_yticklabels(['{:0.0f}%'.format(x*100) for x in plt.gca().get_yticks()])

fig.tight_layout()

```



Congratulations: Now you have presented the data in an eye-catching way. This really supports your recommendation for the taxi company.

9) Making a recommendation

The main question for this project was:

- How many taxis should the Taxi company keep waiting at JFK airport for each hour and day of the week?

How would you answer this question now? When should the taxi company station more taxis at the airport, and when should they keep fewer taxis there?

Congratulations: You have completed this project! You used a lot of skills you learned in previous lessons. Now you are ready for the final project!

Remember:

- The best way to solve problems is to split them up into smaller steps
- If something doesn't work, searching for it on the internet can work wonders

Do you have any questions about this exercise? Look in the [forum](#) to see if they have already been discussed.

Found a mistake? Contact Support at support@stackfuel.com.
