Exercises chapter 16: Downloading data

Saul SL

June 2023

1 Exercise 16-1 Sitka Rainfall

Sitka is in a temperate rainforest, so it gets a fair amount of rainfall. In the data file `sitka_weather_2018_simple.csv` is a header called PRCP, which represents daily rainfall amounts. Make a visualization focusing on the data in this column. You can repeat the exercise for Death Valley if you're curious how little rainfall occurs in a desert.

```python
# Get indexes
import csv
from datetime import datetime
import matplotlib.pyplot as plt

filename = 'data/sitka_weather_2018_simple.csv'
with open(filename) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    for index, col_header in enumerate(header_row):
        print(index, col_header)

# Plot precipitation for sitka Alaska and Death valle, Tx
filename1 = 'data/sitka_weather_2018_simple.csv'
filename2 = 'data/death_valley_2018_simple.csv'
files = [filename1, filename2]
prpc_list = []
dates_list = []
for ifile in files:
    with open(ifile) as f:
        reader = csv.reader(f)
        header_row = next(reader)
        # Get precipitation data this file.
        dates, prcp = [], []
        for row in reader:
            current_date = datetime.strptime(row[2], '%Y-%m-%d')
            try:
                iprcp = float(row[3])
            except ValueError:
                pass
            else:
                dates.append(current_date)
                prcp.append(iprcp)
    prpc_list.append(prcp)
    dates_list.append(dates)

# Plot precipitation.
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(dates_list[0],
        prpc_list[0],
        c='blue', alpha=0.3)
ax.plot(dates_list[1],
        prpc_list[1],
        c='red', alpha=0.8)
# plt.fill_between(dates, highs, lows, facecolor='orange', alpha=0.1)

# Format plot.
plt.title("Precipitation in Sitka and Death valley - 2018", fontsize=20)
plt.xlabel('', fontsize=16)
fig.autofmt_xdate()
plt.ylabel("Precipitation (mm)", fontsize=16)
plt.tick_params(axis='both', which='major', labelsize=16)

plt.show()
```

2 Exercise 16-2 Sitka–Death Valley Comparison

The temperature scales on the Sitka and Death Valley graphs reflect the different data ranges. To accurately compare the temperature range in Sitka to that of Death Valley, you need identical scales on the y-axis. Change the settings for the y-axis on one or both of the charts in Figures 16-5 and 16-6. Then make a direct comparison between temperature ranges in Sitka and Death Valley (or any two places you want to compare).

```
filename1 = "data/sitka_weather_2018_full.csv"
filename2 = 'data/death_valley_2018_full.csv'

# See indices
with open(filename2) as f:
    reader = csv.reader(f)
    header_row = next(reader)
    first_row = next(reader)
    for index, col_header in enumerate(first_row):
        print(index, col_header)

# Get max temperature for sitka
with open(filename1) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    # Get temperature data this file.
    dates_sitka, tmax_sitka, tmin_sitka = [], [], []
    counter = 0
    for row in reader:
        try:
            counter += 1
            current_date = datetime.strptime(row[2], '%Y-%m-%d')
            imax = int(row[8])
            imin = int(row[9])
        except ValueError:
            print(f'Error in line: {counter}')
        else:
            dates_sitka.append(current_date)
            tmax_sitka.append(imax)
            tmin_sitka.append(imin)


# Get max temperature for death_valley
with open(filename2) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    # Get precipitation data this file.
    dates_dv, tmax_dv, tmin_dv = [], [], []
    counter = 0
    for row in reader:
        try:
            counter += 1
            current_date = datetime.strptime(row[2], '%Y-%m-%d')
            imax = int(row[6])
            imin = int(row[7])
        except ValueError:
            print(f'Error in line: {counter}')
        else:
            dates_dv.append(current_date)
            tmax_dv.append(imax)
            tmin_dv.append(imin)

# plots
plt.style.use('seaborn')

# Plot the high and low temperatures for Sitka.
fig, plt1 = plt.subplots()
plt1.plot(dates_sitka, tmax_sitka, c='red', alpha=0.5)
plt1.plot(dates_sitka, tmin_sitka, c='blue', alpha=0.5)
plt.fill_between(dates_sitka, tmax_sitka, tmin_sitka,
                 facecolor='orange', alpha=0.1)
plt.ylim(-20, 140)

plt.savefig('Temp_Sitka_2018v2.png')

# Plot the high and low temperatures for Death valley.
fig2, plt2 = plt.subplots()
plt2.plot(dates_dv, tmax_dv, c='red', alpha=0.5)
plt2.plot(dates_dv, tmin_dv, c='blue', alpha=0.5)
plt.fill_between(dates_dv, tmax_dv, tmin_dv,
                 facecolor='orange', alpha=0.1)
plt.ylim(-20, 140)
plt.savefig('Temp_DeathValley_2018v2.png')

# Plot the high and low temperatures for Sitka and DeathValley.
fig, ax = plt.subplots()
ax.plot(dates_sitka, tmax_sitka, c='#aa0000', alpha=0.5)
ax.plot(dates_sitka, tmin_sitka, c='#14289c', alpha=0.5)

ax.plot(dates_dv, tmax_dv, c='#ff2a00', alpha=0.5)
ax.plot(dates_dv, tmin_dv, c='#0055ff', alpha=0.5)

plt.fill_between(dates_sitka, tmax_sitka, tmin_sitka,
                 facecolor='orange', alpha=0.1)
plt.fill_between(dates_dv, tmax_dv, tmin_dv,
                 facecolor='orange', alpha=0.1)

plt.ylim(-20, 140)
plt.show()
```

3 Exercise 16-3 San Francisco

Are temperatures in San Francisco more like temperatures in Sitka or temperatures in Death Valley? Download some data for San Francisco, and generate a high-low temperature plot for San Francisco to make a comparison.

```
filename1 = "data/san_francisco_weather_2018.csv"
filename2 = 'data/death_valley_2018_full.csv'

# See indices
```

```python
with open(filename1) as f:
    reader = csv.reader(f)
    header_row = next(reader)
    first_row = next(reader)
    for index, col_header in enumerate(header_row):
        print(index, col_header)

# Get max temperature for san francisco
with open(filename1) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    # Get temperature data this file.
    dates_sf, tmax_sf, tmin_sf = [], [], []
    counter = 0
    for row in reader:
        try:
            counter += 1
            current_date = datetime.strptime(row[2], '%Y-%m-%d')
            imax = int(row[4])
            imin = int(row[5])
        except ValueError:
            print(f'Error in line: {counter}')
        else:
            dates_sf.append(current_date)
            tmax_sf.append(imax)
            tmin_sf.append(imin)


# Get max temperature for death_valley
with open(filename2) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    # Get precipitation data this file.
    dates_dv, tmax_dv, tmin_dv = [], [], []
    counter = 0
    for row in reader:
        try:
            counter += 1
            current_date = datetime.strptime(row[2], '%Y-%m-%d')
            imax = int(row[6])
            imin = int(row[7])
        except ValueError:
            print(f'Error in line: {counter}')
        else:
            dates_dv.append(current_date)
            tmax_dv.append(imax)
            tmin_dv.append(imin)

fig, ax = plt.subplots()
ax.plot(dates_sf, tmax_sf, c='#a40f14', alpha=0.5,
        label='San Francisco T. max')
ax.plot(dates_sf, tmin_sf, c='#fa6a49', alpha=0.5,
        label='San Francisco T. min')

ax.plot(dates_dv, tmax_dv, c='#243493', alpha=0.5,
        label='Death Valley T. max')
ax.plot(dates_dv, tmin_dv, c='#41b6c4', alpha=0.5,
        label='Death Valley T. min')

plt.fill_between(dates_sf, tmax_sf, tmin_sf,
                 facecolor='#fb9271', alpha=0.1)
```

```
plt.fill_between(dates_dv, tmax_dv, tmin_dv,
                 facecolor='#7ecdba', alpha=0.1)

plt.ylim(20, 140)
# Add a legend
plt.legend()
plt.show()
```

4 Exercise 16-4 Automatic Indexes

In this section, we hardcoded the indexes corresponding to the TMIN and TMAX columns. Use the header row to determine the indexes for these values, so your program can work for Sitka or Death Valley. Use the station name to automatically generate an appropriate title for your graph as well.

```
filename1 = "data/san_francisco_weather_2018.csv"
filename2 = 'data/death_valley_2018_full.csv'

# Get indices
with open(filename1) as f:
    reader = csv.reader(f)
    header_row = next(reader)
    T_max = None
    T_min = None
    for index, col_header in enumerate(header_row):
        if col_header == 'TMAX':
            T_max = index
        elif col_header == 'TMIN':
            T_min = index

with open(filename1) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    # Get temperature data this file.
    dates_sf, tmax_sf, tmin_sf = [], [], []
    counter = 0
    for row in reader:
        try:
            counter += 1
            current_date = datetime.strptime(row[2], '%Y-%m-%d')
            imax = int(row[T_max])
            imin = int(row[T_min])
        except ValueError:
            print(f'Error in line: {counter}')
        else:
            dates_sf.append(current_date)
            tmax_sf.append(imax)
            tmin_sf.append(imin)

# Death valley
with open(filename2) as f:
    reader = csv.reader(f)
    header_row = next(reader)
    T_max = None
    T_min = None
    for index, col_header in enumerate(header_row):
        if col_header == 'TMAX':
            T_max = index
        elif col_header == 'TMIN':
            T_min = index

```

```
with open(filename2) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    # Get precipitation data this file.
    dates_dv, tmax_dv, tmin_dv = [], [], []
    counter = 0
    for row in reader:
        try:
            counter += 1
            current_date = datetime.strptime(row[2], '%Y-%m-%d')
            imax = int(row[T_max])
            imin = int(row[T_min])
        except ValueError:
            print(f'Error in line: {counter}')
        else:
            dates_dv.append(current_date)
            tmax_dv.append(imax)
            tmin_dv.append(imin)

# Plot
fig, ax = plt.subplots()
ax.plot(dates_sf, tmax_sf, c='#a40f14', alpha=0.5,
        label='San Francisco T. max')
ax.plot(dates_sf, tmin_sf, c='#fa6a49', alpha=0.5,
        label='San Francisco T. min')

ax.plot(dates_dv, tmax_dv, c='#243493', alpha=0.5,
        label='Death Valley T. max')
ax.plot(dates_dv, tmin_dv, c='#41b6c4', alpha=0.5,
        label='Death Valley T. min')

plt.fill_between(dates_sf, tmax_sf, tmin_sf,
                 facecolor='#fb9271', alpha=0.1)
plt.fill_between(dates_dv, tmax_dv, tmin_dv,
                 facecolor='#7ecdba', alpha=0.1)

plt.ylim(20, 140)
# Add a legend
plt.legend()
plt.title('Test', fontsize=16)
plt.show()
```

5 Exercise 16-7 Automated Title

In this section, we specified the title manually when defining `my_layout`, which means we have to remember to update the title every time the source file changes. Instead, you can use the title for the data set in the metadata part of the JSON file. Pull this value, assign it to a variable, and use this for the title of the map when you're defining `my_layout`.

```
import json
from plotly.graph_objs import Scattergeo, Layout
from plotly import offline

filename = 'data/eq_data_30_day_m1.json'

# load data
with open(filename) as f:
    all_eq_data = json.load(f)

# make a list of all earthquakes
```

```
all_eq_dicts = all_eq_data['features']
ititle = all_eq_data['metadata']['title']

# Extract magnitudes and coordinates
mags, lons, lats, hover_texts = [], [], [], []
mags = [eq_dict["properties"]["mag"] for eq_dict in all_eq_dicts]
lons = [eq_dict["geometry"]["coordinates"][0] for eq_dict in all_eq_dicts]
lats = [eq_dict["geometry"]["coordinates"][1] for eq_dict in all_eq_dicts]
hover_texts = [eq_dict["properties"]["title"] for eq_dict in all_eq_dicts]

data = [{
    'type': 'scattergeo',
    'lon': lons,
    'lat': lats,
    'text': hover_texts,
    'marker': {
        'size': [3.5*mag for mag in mags],
        'color': mags,
        'colorscale': 'Electric',
        'reversescale': True,
        'colorbar': {'title': 'Magnitude'},
    },
}]
my_layout = Layout(title=ititle)

fig = {'data': data, 'layout': my_layout}
offline.plot(fig, filename='global_earthquakes_v4.html')
```

6 Exercise 16-8 Recent Earthquakes

You can find data files containing information about the most recent earthquakes over 1-hour, 1-day, 7-day, and 30-day periods online. Go to `https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php` and you'll see a list of links to data sets for various time periods, focusing on earthquakes of different magnitudes. Download one of these data sets, and create a visualization of the most recent earthquake activity.

```
filename = 'data/eq_data_4-5_30_day.json'

# load data
with open(filename) as f:
    all_eq_data = json.load(f)

# make a list of all earthquakes
all_eq_dicts = all_eq_data['features']
ititle = all_eq_data['metadata']['title']

# Extract magnitudes and coordinates
mags, lons, lats, hover_texts = [], [], [], []
mags = [eq_dict["properties"]["mag"] for eq_dict in all_eq_dicts]
lons = [eq_dict["geometry"]["coordinates"][0] for eq_dict in all_eq_dicts]
lats = [eq_dict["geometry"]["coordinates"][1] for eq_dict in all_eq_dicts]
hover_texts = [eq_dict["properties"]["title"] for eq_dict in all_eq_dicts]

data = [{
    'type': 'scattergeo',
    'lon': lons,
    'lat': lats,
    'text': hover_texts,
    'marker': {
        'size': [2.5*mag for mag in mags],
        'color': mags,
        'colorscale': 'Earth',
```

```python
        'reversescale': True,
        'colorbar': {'title': 'Magnitude'},
    },
}]
my_layout = Layout(title=ititle)

fig = {'data': data, 'layout': my_layout}
offline.plot(fig, filename='global_earthquakes_v5.html')
```

7 Exercise 16-9 World Fires

In the resources for this chapter, you'll find a file called `world_fires_1_day.csv`. This file contains information about fires burning in different locations around the globe, including the latitude and longitude, and the brightness of each fire. Using the data processing work from the first part of this chapter and the mapping work from this section, make a map that shows which parts of the world are affected by fires. You can download more recent versions of this data at `https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms/active-fire-data/`. You can find links to the data in CSV format in the TXT section.

```python
import numpy as np
import pandas as pd
import csv
filename = 'data/world_fires_1_day.csv'

# read data as csv
# data = np.loadtxt(filename, delimiter=',', skiprows=1)
data = pd.read_csv(filename, header=0)

# Get indices
with open(filename) as f:
    reader = csv.reader(f)
    header_row = next(reader)
    lat_indx = None
    lon_indx = None
    bright_indx = None
    for index, col_header in enumerate(header_row):
        if col_header == 'latitude':
            lat_indx = index
        elif col_header == 'longitude':
            lon_indx = index
        elif col_header == 'brightness':
            bright_indx = index

with open(filename) as f:
    reader = csv.reader(f)
    header_row = next(reader)

    # Get temperature data this file.
    brights, lons, lats = [], [], []
    counter = 0
    for row in reader:
        try:
            counter += 1
            lat = float(row[lat_indx])
            lon = float(row[lon_indx])
            bright = float(row[bright_indx])
        except ValueError:
            print(f'Error in line: {counter}')
        else:
            brights.append(bright)
            lons.append(lon)
            lats.append(lat)
```

```

data = [{
    'type': 'scattergeo',
    'lon': lons,
    'lat': lats,
    'text': hover_texts,
    'marker': {
        'size': [bgt/100 for bgt in brights],
        'color': brights,
        'colorscale': 'Earth',
        'reversescale': True,
        'colorbar': {'title': 'Brightness'},
    },
}]

my_layout = Layout(title='World Fires')

fig = {'data': data, 'layout': my_layout}
offline.plot(fig, filename='world_fires_v1.html')
```