# temporal_data

March 29, 2023

[ ]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
# read data file
df = pd.read_csv ('AirPassengers.csv')
```

[ ]:
```
# basic statistics of the data frame
df.shape
```

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

[ ]:
```
df.head()
```

[ ]:
```
df.tail()
```

[ ]:
```
# convert column Date from object to datetime
df['Date'] = pd.to_datetime (df['Date'])
```

[ ]:
```
# check the conversion
df.info()
```

[ ]:
```
# check
df.head()
```

[ ]:
```
# timedelta is used for calculating an absolute time duration
week_delta = pd.to_timedelta(np.arange(5), unit = 'w')
```

[ ]:
```
print (week_delta)
```

[ ]:
```
# add a time delta to a date
dates = pd.to_datetime(['03/27/2023', '03/27/2023', '03/27/2023',
                        '03/27/2023', '03/27/2023'])
print (dates + week_delta)
```

[ ]:
```
# time spans
pd.Period ('2023')
```

[ ]:
```
pd.Period ('2023-03')
```

[ ]:
```
pd.Period ('2023-03-27')
```

[ ]:
```
# date offsets - daylight savings time in 2023
timestamp = pd.Timestamp ('2023-03-12 00:00:00', tz = 'US/Central')
print (timestamp + pd.Timedelta(days=1))
```

[ ]:
```
# create month, day, and day_name columns
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day
df['day_name'] = df['Date'].dt.day_name()
```

[ ]:
```
# check that columns were added
df.head()
```

[ ]:
```
# aggregate passengers by month
passengers_per_month = df.groupby(['month'])[('#Passengers')].agg('sum')
passengers_per_month = passengers_per_month.reset_index()
```

[ ]:
```
print (passengers_per_month)
```

[ ]:
```
# create bar plot
ax = sns.barplot (x = 'month', y = '#Passengers', data = passengers_per_month)
ax.set_title ('Plot of Passengers per Month')
for p, v in zip (ax.patches, passengers_per_month['#Passengers']):
    height = p.get_height()
    ax.text (p.get_x() + p.get_width() / 2, height + 5, v,
             ha = 'center', va = 'bottom')
plt.show()
```

[ ]:
```
# calculate mean passengers per month
mean_passengers_per_month = df.groupby(['month'])[['#Passengers']].agg('mean').
 ↪reset_index()
print (mean_passengers_per_month)
```

[ ]:
```
# calculate the median passengers per month
median_passengers_per_month = df.groupby(['month'])[['#Passengers']].
 ↪agg('median').reset_index()
print (median_passengers_per_month)
```

[ ]:
```
# use lineplot to display
ax = sns.lineplot (x = 'month', y = '#Passengers', data = df, errorbar = ('ci',␣
 ↪80))
```

```
ax.set_title ('Line Plot Mean and Standard Deviation per Month')
plt.show()
```

[ ]:
```
# compute zscore
df['mean'] = df['#Passengers'].mean()
df['std'] = df['#Passengers'].std()
df['zscore'] = (df['#Passengers'] - df['mean']) / (df['std'])
df['zscore_abs'] = abs (df['zscore'])
df.sort_values (by= 'zscore_abs', ascending = False).head(100)
```

[ ]:
```
# obtain the outliers
df_high = df.sort_values (by = 'zscore', ascending = False).head(10)
df_low = df.sort_values (by = 'zscore', ascending = True).head(10)
```

[ ]:
```
# plot the outliers
plt.figure (figsize = (15,8))
plt.grid = True
plt.title ('Top 10 High Traffic Passenger Count')
ax = sns.lineplot (x = 'Date', y = '#Passengers', data = df)
ax = sns.scatterplot (x = 'Date', y = '#Passengers',
                      data = df_high, size = '#Passengers')
ax = sns.scatterplot (x = 'Date', y = '#Passengers',
                      data = df_low, size = '#Passengers')
ax = sns.lineplot (x = 'Date', y = 'mean', data = df)
ax.grid()
```

Resampling Temporal Data

**Upsampling:** change time from higher granularity to lower granularity, for example, from minutes to seconds. Allows us to visualize data in moredetail.

**Downsampling:** change time from lower granularity to higher granularity, for example from months to years. Allows us to get a general sense of the trends in the data.

Parking Birmingham Data Set

**SystemCodeNumber:** Car Park ID

**Capacity:** Car Park capacity

**Occupancy:** Car Park Occupancy rate

**LastUpdated:** Date and Time of the measure

[ ]:
```
# read data file
df1 = pd.read_csv ('carpark.csv', parse_dates=['LastUpdated'])
```

[ ]:
```
# sanity checks
df1.shape
```

[ ]:
```
df1.info()
```

[ ]:
```
df1.head()
```

[ ]:
```
df1.tail()
```

[ ]:
```
df1.describe()
```

[ ]:
```
# set DatetimeIndex
df2 = df1.set_index (pd.DatetimeIndex(df1['LastUpdated'])).drop('LastUpdated',␣
 ↪axis = 1)
```

[ ]:
```
df2.head()
```

[ ]:
```
# resample on a daily basis
df3 = pd.DataFrame()
df3['Occupancy'] = df2['Occupancy'].resample('D').mean()
```

[ ]:
```
print (df3)
```

[ ]:
```
# make a plot with seaborn
ax = sns.lineplot (x = 'LastUpdated', y = 'Occupancy', data = df3)
ax.set_title ('Plot of Daily Sampling')
ax.set_xlabel ('Last Updated')
ax.set_ylabel ('Occupancy')
plt.show()
```

[ ]:
```
# resample on a weekly basis
df4 = pd.DataFrame()
df4['Occupancy'] = df2['Occupancy'].resample('W').mean()
```

[ ]:
```
# make a plot with seaborn
ax = sns.lineplot (x = 'LastUpdated', y = 'Occupancy', data = df4)
ax.set_title ('Plot of Weekly Sampling')
ax.set_xlabel ('Last Updated')
ax.set_ylabel ('Occupancy')
plt.show()
```

[ ]:
```
# resample on a monthly basis
df5 = pd.DataFrame()
df5['Occupancy'] = df2['Occupancy'].resample('M').mean()
```

[ ]:
```
# make a plot with seaborn
ax = sns.lineplot (x = 'LastUpdated', y = 'Occupancy', data = df5)
ax.set_title ('Plot of Monthly Sampling')
ax.set_xlabel ('Last Updated')
ax.set_ylabel ('Occupancy')
plt.show()
```