assignment4

May 5, 2021

1 Assignment 4

1.1 Description

In this assignment you must read in a file of metropolitan regions and associated sports teams from `assets/wikipedia_data.html` and answer some questions about each metropolitan region. Each of these regions may have one or more teams from the "Big 4": NFL (football, in `assets/nfl.csv`), MLB (baseball, in `assets/mlb.csv`), NBA (basketball, in `assets/nba.csv` or NHL (hockey, in `assets/nhl.csv`). Please keep in mind that all questions are from the perspective of the metropolitan region, and that this file is the "source of authority" for the location of a given sports team. Thus teams which are commonly known by a different area (e.g. "Oakland Raiders") need to be mapped into the metropolitan region given (e.g. San Francisco Bay Area). This will require some human data understanding outside of the data you've been given (e.g. you will have to hand-code some names, and might need to google to find out where teams are)!

For each sport I would like you to answer the question: **what is the win/loss ratio's correlation with the population of the city it is in?** Win/Loss ratio refers to the number of wins over the number of wins plus the number of losses. Remember that to calculate the correlation with `pearsonr`, so you are going to send in two ordered lists of values, the populations from the wikipedia_data.html file and the win/loss ratio for a given sport in the same order. Average the win/loss ratios for those cities which have multiple teams of a single sport. Each sport is worth an equal amount in this assignment (20%*4=80%) of the grade for this assignment. You should only use data **from year 2018** for your analysis – this is important!

1.2 Notes

1. Do not include data about the MLS or CFL in any of the work you are doing, we're only interested in the Big 4 in this assignment.
2. I highly suggest that you first tackle the four correlation questions in order, as they are all similar and worth the majority of grades for this assignment. This is by design!
3. It's fair game to talk with peers about high level strategy as well as the relationship between metropolitan areas and sports teams. However, do not post code solving aspects of the assignment (including such as dictionaries mapping areas to teams, or regexes which will clean up names).
4. There may be more teams than the assert statements test, remember to collapse multiple teams in one city into a single value!

1.3 Question 1

For this question, calculate the win/loss ratio's correlation with the population of the city it is in for the **NHL** using **2018** data.

[11]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import re

nhl_df=pd.read_csv("assets/nhl.csv")
cities=pd.read_html("assets/wikipedia_data.html")[1]
cities=cities.iloc[:-1,[0,3,5,6,7,8]]

def nhl_correlation():
                    # All the modifications regarding `nhl_df` data:

    # 1. get only 2018 records.
    nhl_18_df = nhl_df[nhl_df['year'] == 2018]
    # remove `*` that written besides teams' names.
    nhl_18_df= nhl_18_df.replace('\*','',regex=True)
    # specify our columns of interest which are `team` ,`W` &  `L
    nhl_18_df = nhl_18_df.iloc[:,[0,2,3]]

    # get rid of the unwanted rows that holds similar value in all columns:
    # create a list to store the index of the unwanted rows.
    unwanted_rows_indexes = []
    # iterate over all the indexes in the dataframe.
    for i in nhl_18_df.index :
        # use the indexes to hold every row.
        row = nhl_18_df.iloc[i]
        # set the condition that has to be met,then append the index in
        # the list.
        if row['team'] == row['W'] == row['L']:
            unwanted_rows_indexes.append(i)

    # drop the unwanted rows in action.
    nhl_18_df.drop(unwanted_rows_indexes, inplace = True)

    # remove the entire team's name except the last name for merging purposes.
    nhl_18_df['team'] = nhl_18_df['team'].str.replace('[\w.]* ','')

    # set the dtype of each column, then calculate `the win_to_lose_ratio`.
    nhl_18_df = nhl_18_df.astype({'team': str, 'W': int, 'L': int})
    nhl_18_df['W_L_Ratio'] = nhl_18_df['W']/(nhl_18_df['W']+nhl_18_df['L'])
```

```
                        # All the modifications regarding `cities` dataFrame.

   # read our html file from wikipedia.
   cities=pd.read_html("assets/wikipedia_data.html")[1]
   # select our columns of interest. keep in mind to exclude the last row that had the total.
   cities=cities.iloc[:-1,[0,3,5,6,7,8]]

   # remove `[note ]` in the names of the teams.
   cities= cities.replace('\[.*\]','',regex=True)
   # rename the pop column to be more direct.
   cities.rename(columns = {'Population (2016 est.)[8]':'Population'}, inplace= True)

   # so important `regex. with str.extract()`
   # the purpose of this regex is to extract the teams' names that are
   # concatenated with each other like: RangersIslandersDevils and make
   # each name in seperate column.
   team = cities['NHL'].str.extract('([A-Z]{0,2}[a-z0-9]*[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
   # since team dataframe was part of cities dataframe so they have the same
   # num of rows therefore i can make 'Metropolitan area' col accordingly.
   team['Metropolitan area'] = cities['Metropolitan area']

   # here we want to get the teams' names that each one of them was in separate
   # column all in one column, using melt() this brings 'variable' col (holds the column index or name) &
   # `value` col (holds teams' names)
   team = pd.melt(team, id_vars=['Metropolitan area'])

   team = team.replace('',np.NaN)                          # assign NaN values for the '-' & ''.
   team = team.replace('',np.NaN)
   team = team.dropna()                    # keep in mind that dropna() don't rearange the indexes so you need to reset it.
   team = team.drop('variable',axis = 1)                   # drop the 'variable' col cuz it is no longer needed.
   team = team.reset_index()                               # reset the index because dropna() just droped them only.
   team.rename(columns = {'value':'team'},inplace = True)  # rename the 'value' we got from melt() to be `team`
```

```
    # remove the entire team's name except the last name for merging purposes.
    team['team']= team['team'].str.replace('[\w.]*\ ','')



                                            # creating `team_df`

    # now, the time of merging back between `cities` & `team` but make sure to␣
↪make the join
    # with respect to `team` to get only the 31 teams we got as result  of the␣
↪cleaning steps.
    team_df = pd.merge(team,cities,how = 'left' , on= 'Metropolitan area')
    team_df = team_df.iloc[:,1:4]                                              ␣
↪         # select our 3 columns of interest.
    team_df = team_df.astype({'Metropolitan area': str, 'team': str,␣
↪'Population': int}) # assign the types of our 3 cols.

                                           # creating `final_df`
    # the purpose of this merging is getting `W` (num of wins) `L` (num loses)␣
↪columns.
    # and `w_l_ratio`. each df has 31 team.
    final_df =pd.merge(team_df,nhl_18_df,how='inner',on='team')
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W_L_Ratio':np.
↪nanmean,
                                             'Population':np.nanmean})

    # raise NotImplementedError()
    # pass in metropolitan area population from cities
    population_by_region =aggregated_df['Population']
    # pass in win/loss ratio from nhl_df in the same order as
    # cities["Metropolitan area"]
    win_loss_by_region =aggregated_df['W_L_Ratio']

    assert len(population_by_region) == len(win_loss_by_region), "Q1: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 28, "Q1: There should be 28 teams being␣
↪analysed for NHL"

    return stats.pearsonr(population_by_region, win_loss_by_region)[0]
```

[ ]:

1.4 Question 2

For this question, calculate the win/loss ratio's correlation with the population of the city it is in for the **NBA** using **2018** data.

[31]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import re

nba_df=pd.read_csv("assets/nba.csv")
cities=pd.read_html("assets/wikipedia_data.html")[1]
cities=cities.iloc[:-1,[0,3,5,6,7,8]]

def nba_correlation():
    # All the modifications regarding `nhl_df` data:

    # 1. get only 2018 records.
    nba_18_df = nba_df[nba_df['year'] == 2018]
    # remove `*` that written besides teams' names.
    nba_18_df['team'] = nba_18_df['team'].str.replace('[\*]',"")
    nba_18_df['team'] = nba_18_df['team'].str.replace('\(\d*\)',"")
    nba_18_df['team'] = nba_18_df['team'].str.replace('[\xa0]',"")
    # specify our columns of interest which are `team` ,`W` &  `L
    nba_18_df = nba_18_df.iloc[:,[0,3]]
    # remove the entire team's name except the last name for merging purposes.
    nba_18_df['team'] = nba_18_df['team'].str.replace('[\w.]* ','')
    # set the dtype of each column, then calculate `the win_to_lose_ratio`.
    nba_18_df = nba_18_df.astype({'team': str, 'W/L%': float})


                            # All the modifications regarding `cities`␣
↪dataFrame.

    # read our html file from wikipedia.
    cities=pd.read_html("assets/wikipedia_data.html")[1]
    # select our columns of interest. keep in mind to exclude the last row that␣
↪had the total.
    cities=cities.iloc[:-1,[0,3,5,6,7,8]]
     # remove `[note ]` in the names of the teams.
    cities= cities.replace('\[.*\]','',regex=True)
    # rename the pop column to be more direct.
    cities.rename(columns = {'Population (2016 est.)[8]':'Population'},␣
↪inplace= True)

    # so important `regex. with str.extract()`
    # the purpose of this regex is to extract the teams' names that are
    # concatenated with each other like: RangersIslandersDevils and make
    # each name in seperate column.
```

```
   team = cities['NBA'].str.extract('([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
   # since team dataframe was part of cities dataframe so they have the same
   # num of rows therefore i can make 'Metropolitan area' col accordingly.
   team['Metropolitan area'] = cities['Metropolitan area']

   # here we want to get the teams' names that each one of them was in␣
↪separate
   # column all in one column, using melt() this brings 'variable' col (holds␣
↪the column index or name) &
   # `value` col (holds teams' names)
   team = pd.melt(team, id_vars=['Metropolitan area'])

   team = team.replace('',np.NaN)                          # assign NaN values␣
↪for the '-' & ''.
   team = team.replace('',np.NaN)
   team = team.dropna()                  # keep in mind that dropna() don't␣
↪rearange the indexes so you need to reset it.
   team = team.drop('variable',axis = 1)                    # drop the␣
↪'variable' col cuz it is no longer needed.
   team = team.reset_index()                                # reset the index␣
↪because dropna() just droped them only.
   team.rename(columns = {'value':'team'},inplace = True)  # rename the␣
↪'value' we got from melt() to be `team`

   # remove the entire team's name except the last name for merging purposes.
   team['team']= team['team'].str.replace('[\w.]*\ ','')

   # now, the time of merging back between `cities` & `team` but make sure to␣
↪make the join
   # with respect to `team` to get only the 31 teams we got as result  of the␣
↪cleaning steps.
   team_df = pd.merge(team,cities,how = 'left' , on= 'Metropolitan area')
   team_df = team_df.iloc[:,1:4]                                              ␣
↪         # select our 3 columns of interest.
   team_df = team_df.astype({'Metropolitan area': str, 'team': str,␣
↪'Population': int}) # assign the types of our 3 cols.



                                        # creating `final_df`
   # the purpose of this merging is getting `W` (num of wins) `L` (num loses)␣
↪columns.
   # and `w_l_ratio`. each df has 31 team.
```

```
    final_df =pd.merge(team_df,nba_18_df,how='outer',on='team')
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W/L%':np.
↪nanmean,
                                                'Population':np.nanmean})

    # pass in metropolitan area population from cities
    population_by_region =aggregated_df['Population']
    # pass in win/loss ratio from nhl_df in the same order as
    # cities["Metropolitan area"]
    win_loss_by_region =aggregated_df['W/L%']

    # YOUR CODE HERE
    # raise NotImplementedError()

    population_by_region = aggregated_df['Population'] # pass in metropolitan␣
↪area population from cities
    win_loss_by_region = aggregated_df['W/L%'] # pass in win/loss ratio from␣
↪nba_df in the same order as cities["Metropolitan area"]

    assert len(population_by_region) == len(win_loss_by_region), "Q2: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 28, "Q2: There should be 28 teams being␣
↪analysed for NBA"

    return stats.pearsonr(population_by_region, win_loss_by_region)[0]
```

[ ]:

1.5 Question 3

For this question, calculate the win/loss ratio's correlation with the population of the city it is in for the **MLB** using **2018** data.

[5]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import re

mlb_df=pd.read_csv("assets/mlb.csv")
cities=pd.read_html("assets/wikipedia_data.html")[1]
cities=cities.iloc[:-1,[0,3,5,6,7,8]]

def mlb_correlation():
    cities=pd.read_html("assets/wikipedia_data.html")[1]
    cities=cities.iloc[:-1,[0,3,5,6,7,8]]
    cities.rename(columns={"Population (2016 est.)[8]": "Population"},
              inplace=True)
```

```
   # so important `regex. with str.extract()`
   # the purpose of this regex is to extract the teams' names that are
   # concatenated with each other like: RangersIslandersDevils and make
   # each name in seperate column.
   team = cities['MLB'].str.extract('([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
   # since team dataframe was part of cities dataframe so they have the same
   # num of rows therefore i can make 'Metropolitan area' col accordingly.
   team['Metropolitan area'] = cities['Metropolitan area']

   # here we want to get the teams' names that each one of them was in␣
↪separate
   # column all in one column, using melt() this brings 'variable' col (holds␣
↪the column index or name) &
   # `value` col (holds teams' names)
   team = pd.melt(team, id_vars=['Metropolitan area'])

   team = team.replace('',np.NaN)                          # assign NaN values␣
↪for the '-' & ''.
   team = team.replace('',np.NaN)
   team = team.dropna()                  # keep in mind that dropna() don't␣
↪rearange the indexes so you need to reset it.
   team = team.drop('variable',axis = 1)                    # drop the␣
↪'variable' col cuz it is no longer needed.
   team = team.reset_index()                                # reset the index␣
↪because dropna() just droped them only.
   team.rename(columns = {'value':'team'},inplace = True)  # rename the␣
↪'value' we got from melt() to be `team`

   # remove the entire team's name except the last name for merging purposes.
   team['team']=team['team'].str.replace('\ Sox','Sox')
   team['team']= team['team'].str.replace('[\w.]*\ ','')


   team=pd.merge(team,cities,how='left',on = 'Metropolitan area').iloc[:,1:4]
   team = team.astype({'Metropolitan area': str, 'team': str, 'Population':␣
↪int})



   mlb_df=pd.read_csv("assets/mlb.csv")
```

```
    # All the modifications regarding `nhl_df` data:

    # 1. get only 2018 records.
    mlb_18_df = mlb_df[mlb_df['year'] == 2018]
    # remove `*` that written besides teams' names.
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('[\*]',"")
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('\(\d*\)',"")
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('[\xa0]',"")
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('\ Sox','Sox')

    # specify our columns of interest which are `team` ,`W` &  `L
    mlb_18_df = mlb_18_df.iloc[:,[0,3]]
    mlb_18_df['team']= mlb_18_df['team'].str.replace('[\w.]*\ ','')
    mlb_18_df = mlb_18_df.astype({'team': str,'W-L%': float})



    final_df =pd.merge(team,mlb_18_df , how='outer',on='team')
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W-L%':np.
↪nanmean,
                                                'Population':np.nanmean})


    # raise NotImplementedError()
    population_by_region = aggregated_df['Population'] # pass in metropolitan␣
↪area population from cities
    win_loss_by_region = aggregated_df['W-L%'] # pass in win/loss ratio from␣
↪mlb_df in the same order as cities["Metropolitan area"]

    assert len(population_by_region) == len(win_loss_by_region), "Q3: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 26, "Q3: There should be 26 teams being␣
↪analysed for MLB"

    return stats.pearsonr(population_by_region, win_loss_by_region)[0]
```

[ ]:

1.6 Question 4

For this question, calculate the win/loss ratio's correlation with the population of the city it is in for the **NFL** using **2018** data.

[11]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import re
```

```
nfl_df=pd.read_csv("assets/nfl.csv")
cities=pd.read_html("assets/wikipedia_data.html")[1]
cities=cities.iloc[:-1,[0,3,5,6,7,8]]

def nfl_correlation():
    nfl_df=pd.read_csv("assets/nfl.csv")
    cities=pd.read_html("assets/wikipedia_data.html")[1]
    cities=cities.iloc[:-1,[0,3,5,6,7,8]]
    cities.rename(columns={"Population (2016 est.)[8]": "Population"},
              inplace=True)


    nfl_df = nfl_df[nfl_df['year'] == 2018]
    nfl_df = nfl_df[['team','W-L%']]
    unwanted_rows = []
    for i in nfl_df.index:
        row = nfl_df.iloc[i]
        if row['team'] == row['W-L%']:
            unwanted_rows.append(i)

    nfl_df = nfl_df.drop(unwanted_rows)

    nfl_df['team'] = nfl_df['team'].str.replace('[\*]',"")
    nfl_df['team'] = nfl_df['team'].str.replace('\(\d*\)',"")
    nfl_df['team'] = nfl_df['team'].str.replace('[\xa0]',"")
    nfl_df['team'] = nfl_df['team'].str.replace('\+',"")
    nfl_df['team'] = nfl_df['team'].str.replace('[\w.]* ',"")

    nfl_df = nfl_df.astype({'team': str , 'W-L%':float})


    cities = cities[['Metropolitan area','Population' , 'NFL']]
    team = cities['NFL']
    team= team.str.replace('\[\w.*\]',"")
    team = team.str.extract('([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
    team['Metropolitan area'] = cities['Metropolitan area']

    team = pd.melt(team,id_vars= 'Metropolitan area')

    team = team.replace('',np.NaN)
    team = team.replace('',np.NaN)
    team = team.dropna()
    team = team.drop('variable',axis = 1)
```

```
    team = team.reset_index()
    team.rename(columns = {'value':'team'},inplace = True)
    team =team[['Metropolitan area', 'team']]

    # doing the mege to get population col
    team = pd.merge(team,cities,how='left' , on='Metropolitan area')
    team = team.astype({'Metropolitan area':str , 'team' : str
                        ,'Population':int})

    final_df = pd.merge(team,nfl_df,how='outer',on= 'team')
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W-L%':np.
↪nanmean,'Population':np.nanmean})

    # raise NotImplementedError()
    population_by_region = aggregated_df['Population']  # pass in metropolitan␣
↪area population from cities
    win_loss_by_region = aggregated_df['W-L%']  # pass in win/loss ratio from␣
↪nfl_df in the same order as cities["Metropolitan area"]

    assert len(population_by_region) == len(win_loss_by_region), "Q4: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 29, "Q4: There should be 29 teams being␣
↪analysed for NFL"

    return stats.pearsonr(population_by_region, win_loss_by_region)[0]
```

[ ]:

1.7 Question 5

In this question I would like you to explore the hypothesis that **given that an area has two sports teams in different sports, those teams will perform the same within their respective sports**. How I would like to see this explored is with a series of paired t-tests (so use `ttest_rel`) between all pairs of sports. Are there any sports where we can reject the null hypothesis? Again, average values where a sport has multiple teams in one region. Remember, you will only be including, for each sport, cities which have teams engaged in that sport, drop others as appropriate. This question is worth 20% of the grade for this assignment.

[1]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import re

mlb_df=pd.read_csv("assets/mlb.csv")
nhl_df=pd.read_csv("assets/nhl.csv")
nba_df=pd.read_csv("assets/nba.csv")
nfl_df=pd.read_csv("assets/nfl.csv")
```

```
cities=pd.read_html("assets/wikipedia_data.html")[1]
cities=cities.iloc[:-1,[0,3,5,6,7,8]]

# first:
import pandas as pd
import numpy as np
import scipy.stats as stats
import re
def nhl_correlation():
    nhl_df=pd.read_csv("assets/nhl.csv")
    nhl_18_df = nhl_df[nhl_df['year'] == 2018]
    nhl_18_df= nhl_18_df.replace('\*','',regex=True)
    nhl_18_df = nhl_18_df.iloc[:,[0,2,3]]

    unwanted_rows_indexes = []
    for i in nhl_18_df.index :
        row = nhl_18_df.iloc[i]
        if row['team'] == row['W'] == row['L']:
            unwanted_rows_indexes.append(i)

    nhl_18_df.drop(unwanted_rows_indexes, inplace = True)
    nhl_18_df['team'] = nhl_18_df['team'].str.replace('[\w.]* ','')
    nhl_18_df = nhl_18_df.astype({'team': str, 'W': int, 'L': int})
    nhl_18_df['W/L%'] = nhl_18_df['W']/(nhl_18_df['W']+nhl_18_df['L'])



    cities=pd.read_html("assets/wikipedia_data.html")[1]
    cities=cities.iloc[:-1,[0,3,5,6,7,8]]
    cities= cities.replace('\[.*\]','',regex=True)
    cities.rename(columns = {'Population (2016 est.)[8]':'Population'},␣
 ↪inplace= True)
    team = cities['NHL'].str.extract('([A-Z]{0,2}[a-z0-9]*\␣
 ↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
 ↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
 ↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
    team['Metropolitan area'] = cities['Metropolitan area']
    team = pd.melt(team, id_vars=['Metropolitan area'])
    team = team.replace('',np.NaN)
    team = team.replace('',np.NaN)
    team = team.dropna()
    team = team.drop('variable',axis = 1)
    team = team.reset_index()
    team.rename(columns = {'value':'team'},inplace = True)
    team['team']= team['team'].str.replace('[\w.]*\ ','')
    team_df = pd.merge(team,cities,how = 'left' , on= 'Metropolitan area')
    team_df = team_df.iloc[:,1:4]
```

```
    team_df = team_df.astype({'Metropolitan area': str, 'team': str,␣
↪'Population': int})



    final_df =pd.merge(team_df,nhl_18_df,how='inner',on='team')
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W/L%':np.
↪nanmean,
                                               'Population':np.nanmean})




    population_by_region = aggregated_df['Population']  # pass in metropolitan␣
↪area population from cities
    win_loss_by_region = aggregated_df['W/L%'] # pass in win/loss ratio from␣
↪nhl_df in the same order as cities["Metropolitan area"]

    assert len(population_by_region) == len(win_loss_by_region), "Q1: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 28, "Q1: There should be 28 teams being␣
↪analysed for NHL"

    return aggregated_df[['W/L%']]


# second:
import pandas as pd
import numpy as np
import scipy.stats as stats
import re
nba_df=pd.read_csv("assets/nba.csv")
cities=pd.read_html("assets/wikipedia_data.html")[1]
cities=cities.iloc[:-1,[0,3,5,6,7,8]]

def nba_correlation():
    import pandas as pd
    import numpy as np
    import scipy.stats as stats
    import re

    nba_df=pd.read_csv("assets/nba.csv")
    nba_18_df = nba_df[nba_df['year'] == 2018]
    nba_18_df['team'] = nba_18_df['team'].str.replace('[\*]','')
    nba_18_df['team'] = nba_18_df['team'].str.replace('\(\d*\)','')
    nba_18_df['team'] = nba_18_df['team'].str.replace('[\xa0]','')
```

```
    nba_18_df = nba_18_df.iloc[:,[0,3]]
    nba_18_df['team'] = nba_18_df['team'].str.replace('[\w.]* ','')
    nba_18_df = nba_18_df.astype({'team': str, 'W/L%': float})



    cities=pd.read_html("assets/wikipedia_data.html")[1]
    cities=cities.iloc[:-1,[0,3,5,6,7,8]]
    cities= cities.replace('\[.*\]','',regex=True)
    cities.rename(columns = {'Population (2016 est.)[8]':'Population'},␣
↪inplace= True)
    team = cities['NBA'].str.extract('([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
    team['Metropolitan area'] = cities['Metropolitan area']
    team = pd.melt(team, id_vars=['Metropolitan area'])
    team = team.replace('',np.NaN)
    team = team.replace('',np.NaN)
    team = team.dropna()
    team = team.drop('variable',axis = 1)
    team = team.reset_index()
    team.rename(columns = {'value':'team'},inplace = True)
    team['team']= team['team'].str.replace('[\w.]*\ ','')
    team_df = pd.merge(team,cities,how = 'left' , on= 'Metropolitan area')
    team_df = team_df.iloc[:,1:4]
    team_df = team_df.astype({'Metropolitan area': str, 'team': str,␣
↪'Population': int})



    final_df =pd.merge(team_df,nba_18_df, how='left',on='team')
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W/L%':np.
↪nanmean,
                                                'Population':np.nanmean})

    population_by_region = aggregated_df['Population'] # pass in metropolitan␣
↪area population from cities
    win_loss_by_region = aggregated_df['W/L%'] # pass in win/loss ratio from␣
↪nba_df in the same order as cities["Metropolitan area"]

    assert len(population_by_region) == len(win_loss_by_region), "Q2: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 28, "Q2: There should be 28 teams being␣
↪analysed for NBA"
```

```
    return aggregated_df[['W/L%']]


# third :

import pandas as pd
import numpy as np
import scipy.stats as stats
import re
def mlb_correlation():
    cities=pd.read_html("assets/wikipedia_data.html")[1]
    cities=cities.iloc[:-1,[0,3,5,6,7,8]]
    cities.rename(columns={"Population (2016 est.)[8]": "Population"},
              inplace=True)
    team = cities['MLB'].str.extract('([A-Z]{0,2}[a-z0-9]*\␣
 ↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
 ↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
 ↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
    team['Metropolitan area'] = cities['Metropolitan area']
    team = pd.melt(team, id_vars=['Metropolitan area'])
    team = team.replace('',np.NaN)
    team = team.replace('',np.NaN)
    team = team.dropna()
    team = team.drop('variable',axis = 1)
    team = team.reset_index()
    team.rename(columns = {'value':'team'},inplace = True)
    team['team']=team['team'].str.replace('\ Sox','Sox')
    team['team']= team['team'].str.replace('[\w.]*\ ','')
    team=pd.merge(team,cities,how='left',on = 'Metropolitan area').iloc[:,1:4]
    team = team.astype({'Metropolitan area': str, 'team': str, 'Population':␣
 ↪int})


    mlb_df=pd.read_csv("assets/mlb.csv")
    mlb_18_df = mlb_df[nba_df['year'] == 2018]
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('[\*]',"")
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('\(\d*\)',"")
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('[\xa0]',"")
    mlb_18_df['team'] = mlb_18_df['team'].str.replace('\ Sox','Sox')
    mlb_18_df = mlb_18_df.iloc[:,[0,3]]
    mlb_18_df['team']= mlb_18_df['team'].str.replace('[\w.]*\ ','')
    mlb_18_df.rename(columns={"W-L%": "W/L%"},inplace=True)
    mlb_18_df = mlb_18_df.astype({'team': str,'W/L%': float})



    final_df =pd.merge(team,mlb_18_df , how='outer',on='team')
```

```
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W/L%':np.
↪nanmean,
                                                'Population':np.nanmean})


    population_by_region = aggregated_df['Population'] # pass in metropolitan␣
↪area population from cities
    win_loss_by_region = aggregated_df['W/L%'] # pass in win/loss ratio from␣
↪mlb_df in the same order as cities["Metropolitan area"]

    assert len(population_by_region) == len(win_loss_by_region), "Q3: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 26, "Q3: There should be 26 teams being␣
↪analysed for MLB"

    return aggregated_df[['W/L%']]


# forth:
import pandas as pd
import numpy as np
import scipy.stats as stats
import re



def nfl_correlation():
    nfl_df=pd.read_csv("assets/nfl.csv")
    cities=pd.read_html("assets/wikipedia_data.html")[1]
    cities=cities.iloc[:-1,[0,3,5,6,7,8]]
    cities.rename(columns={"Population (2016 est.)[8]": "Population"},
              inplace=True)


    nfl_df = nfl_df[nfl_df['year'] == 2018]
    nfl_df = nfl_df[['team','W-L%']]
    unwanted_rows = []
    for i in nfl_df.index:
        row = nfl_df.iloc[i]
        if row['team'] == row['W-L%']:
            unwanted_rows.append(i)
    nfl_df = nfl_df.drop(unwanted_rows)

    nfl_df['team'] = nfl_df['team'].str.replace('[\*]',"")
    nfl_df['team'] = nfl_df['team'].str.replace('\(\d*\)',"")
    nfl_df['team'] = nfl_df['team'].str.replace('[\xa0]',"")
    nfl_df['team'] = nfl_df['team'].str.replace('\+',"")
```

```
    nfl_df['team'] = nfl_df['team'].str.replace('[\w.]* ','')
    nfl_df.rename(columns={"W-L%": "W/L%"},inplace=True)
    nfl_df = nfl_df.astype({'team': str , 'W/L%':float})


    cities = cities[['Metropolitan area','Population' , 'NFL']]
    team = cities['NFL']
    team= team.str.replace('\[\w.*\]','')
    team = team.str.extract('([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)([A-Z]{0,2}[a-z0-9]*\␣
↪[A-Z]{0,2}[a-z0-9]*|[A-Z]{0,2}[a-z0-9]*)')
    team['Metropolitan area'] = cities['Metropolitan area']
    team = pd.melt(team,id_vars= 'Metropolitan area')
    team = team.replace('',np.NaN)
    team = team.replace('',np.NaN)
    team = team.dropna()
    team = team.drop('variable',axis = 1)
    team = team.reset_index()
    team.rename(columns = {'value':'team'},inplace = True)
    team =team[['Metropolitan area', 'team']]
    team = pd.merge(team,cities,how='left' , on='Metropolitan area')
    team = team.astype({'Metropolitan area':str , 'team' : str
                                              ,'Population':int})

    final_df = pd.merge(team,nfl_df,how='outer',on= 'team')
    aggregated_df = final_df.groupby('Metropolitan area').agg({'W/L%':np.
↪nanmean,'Population':np.nanmean})

    population_by_region = aggregated_df['Population']  # pass in metropolitan␣
↪area population from cities
    win_loss_by_region = aggregated_df['W/L%']  # pass in win/loss ratio from␣
↪nfl_df in the same order as cities["Metropolitan area"]

    assert len(population_by_region) == len(win_loss_by_region), "Q4: Your␣
↪lists must be the same length"
    assert len(population_by_region) == 29, "Q4: There should be 29 teams being␣
↪analysed for NFL"

    return aggregated_df[['W/L%']]

def generate_dataframe(league_name):
    if league_name == 'NHL':
        return nhl_correlation()
    if league_name == 'NBA':
        return nba_correlation()
```

```
    if league_name == 'MLB':
        return mlb_correlation()
    if league_name == 'NFL':
        return nfl_correlation()
    else:
        return "inappropiate input"




def sports_team_performance():
    # YOUR CODE HERE
    # raise NotImplementedError()

    # Note: p_values is a full dataframe, so df.loc["NFL","NBA"] should be the␣
↪same as df.loc["NBA","NFL"] and
    # df.loc["NFL","NFL"] should return np.nan
    sports = ['NFL', 'NBA', 'NHL', 'MLB']
    p_values = pd.DataFrame({k:np.nan for k in sports}, index=sports)
    for league1 in sports:
        for league2 in sports:
            if league1 != league2:
                master_df = pd.
↪merge(generate_dataframe(league1),generate_dataframe(league2),how='inner' ,␣
↪on = 'Metropolitan area')
                p_values.loc[league1 , league2] = stats.ttest_rel(master_df['W/
↪L%_x'],master_df['W/L%_y'])[1]

    assert abs(p_values.loc["NBA", "NHL"] - 0.02) <= 1e-2, "The NBA-NFL p-value␣
↪should be around 0.02"
    assert abs(p_values.loc["MLB", "NFL"] - 0.80) <= 1e-2, "The MLB-NFL p-value␣
↪should be around 0.80"
    return p_values
```

[ ]: