Homework 2 - Statistical and Numerical Visualization

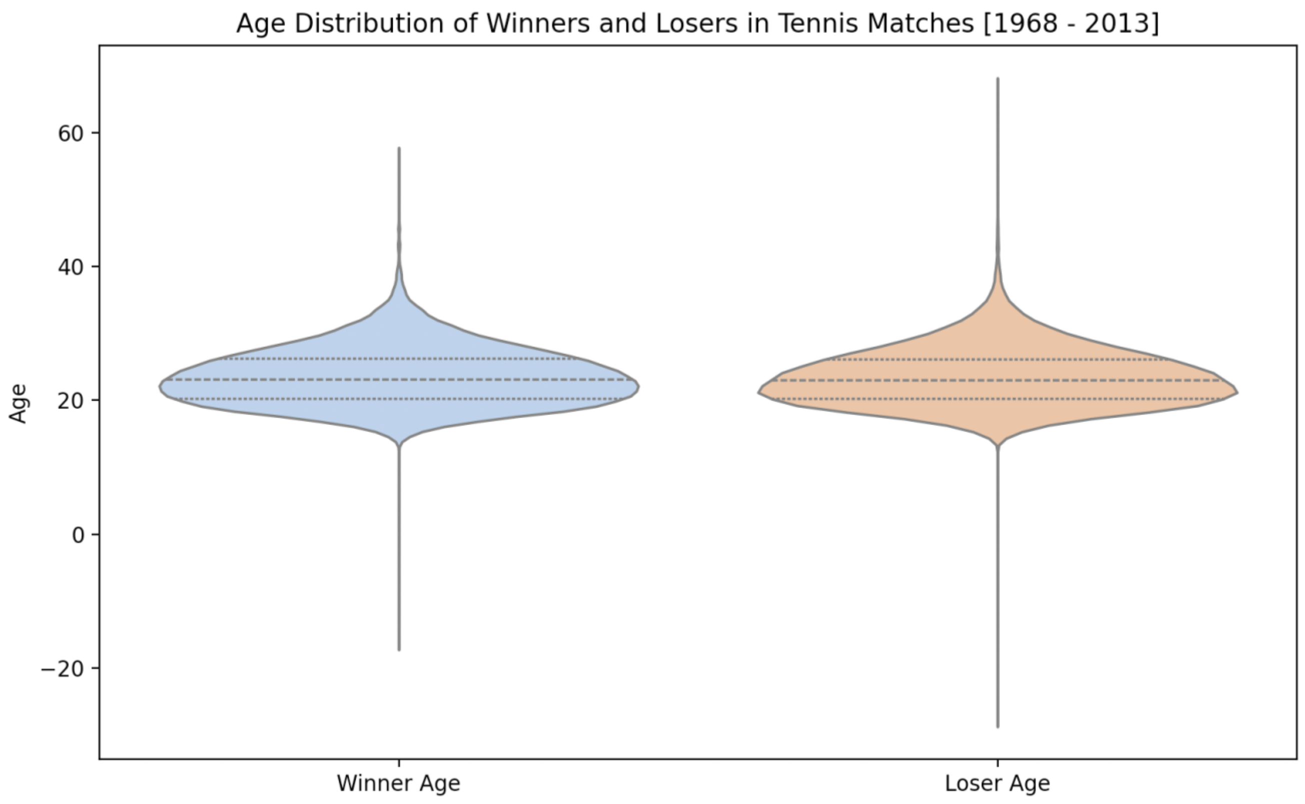
Name: Ashutosh Chauhan
UIN: 232009024

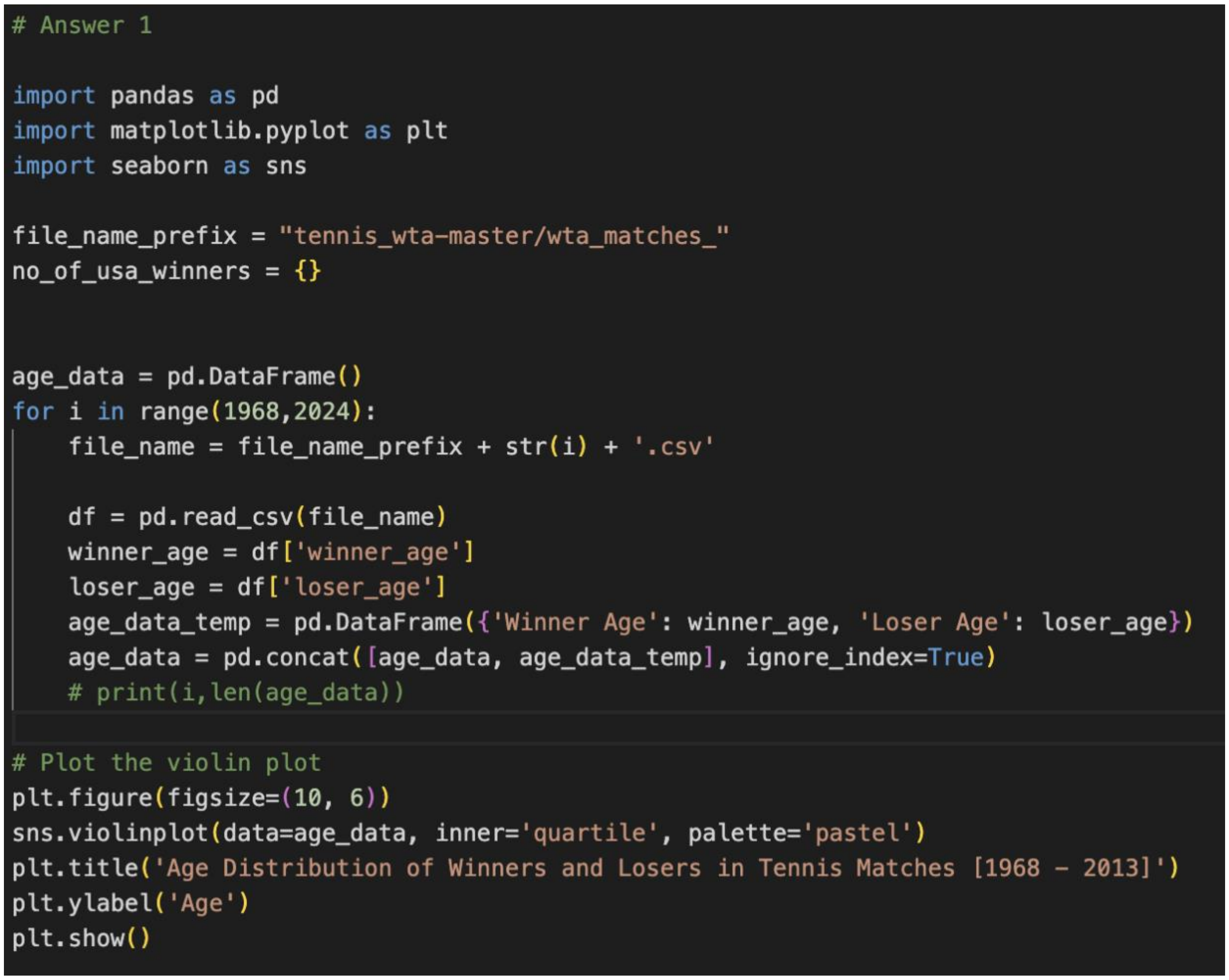
1. At least one showing distributions of a variable

   Type of Visualization: Violin Plot

   Dataset Description:
   In this plot, I am using tennis matches data from year 1968 to 2023 stored in files 'wta_matches_1968' – 'wta_matches_2023'. I am extracting 'winner_age' and 'loser_age' columns from each file and appending them to a single data frame. This dataframe I am using to plot age distribution of winners and losers in tennis matches in form of violin plot.

   Visualization Description:
   For the above dataset we have 1 dependent variable (age). We want to know what is the mean/mode of players who win/lose tennis matches. We used violin plot here because using it we can easily know the mean and mode of age parameter. Also, Violin plots allow for easy comparison of distributions across different categories or groups. We can observe from the below plot that both winners and losers have mode age around 23 and winning does not actually depend on age. It can also easily handle skewed data and outliers.





```python
# Answer 1

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

file_name_prefix = "tennis_wta-master/wta_matches_"
no_of_usa_winners = {}


age_data = pd.DataFrame()
for i in range(1968,2024):
    file_name = file_name_prefix + str(i) + '.csv'

    df = pd.read_csv(file_name)
    winner_age = df['winner_age']
    loser_age = df['loser_age']
    age_data_temp = pd.DataFrame({'Winner Age': winner_age, 'Loser Age': loser_age})
    age_data = pd.concat([age_data, age_data_temp], ignore_index=True)
    # print(i,len(age_data))


# Plot the violin plot
plt.figure(figsize=(10, 6))
sns.violinplot(data=age_data, inner='quartile', palette='pastel')
plt.title('Age Distribution of Winners and Losers in Tennis Matches [1968 - 2013]')
plt.ylabel('Age')
plt.show()
```

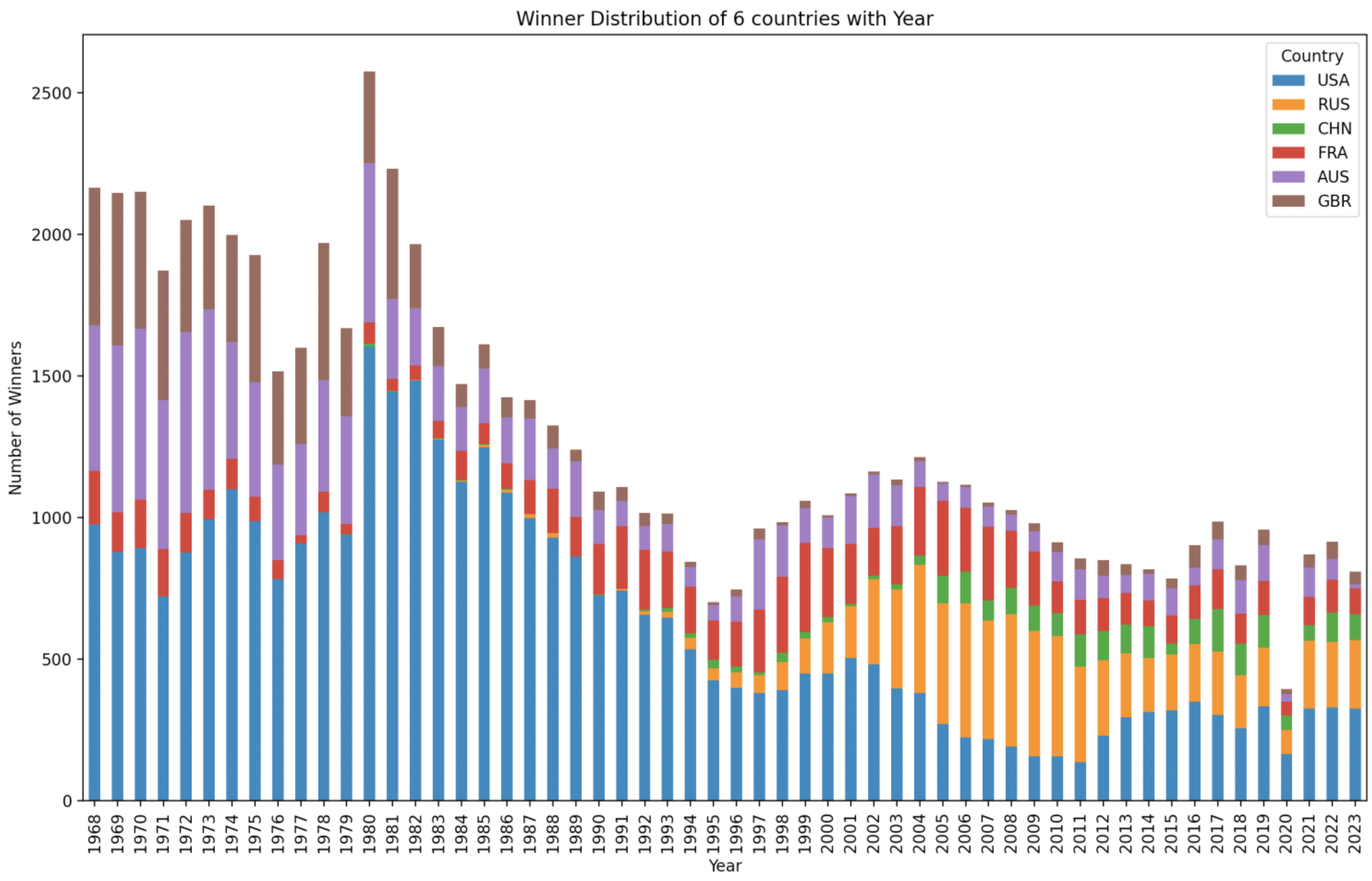
2. At least three plots from the limited number of variables section. At least one of these should be from outside the pie/bar/line group (i.e. one should be from those listed as Population Pyramid and below in the list).

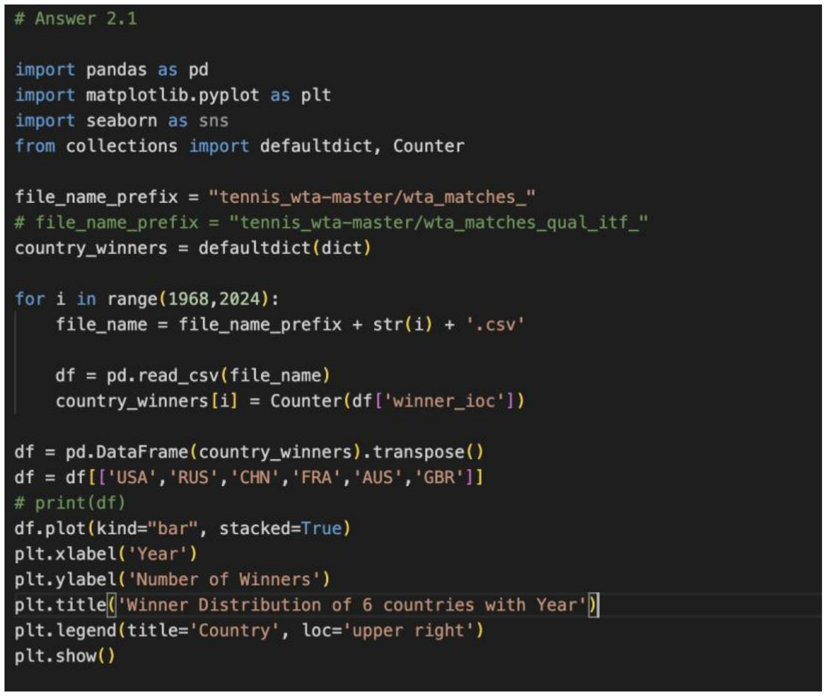
<u>2.1.</u> **<u>Type of Visualization:</u>** Stacked Bar Chart

<u>Dataset Description:</u>
In this plot, I am using tennis matches data from year 1968 to 2023 stored in files ‘wta_matches_1968’ – ‘wta_matches_2023’. I am extracting ‘winner_ioc’ i.e winner’s country columns from each file. I am storing the number of times a country wins match in a ‘country_winners’ dictionary for every year. I am then extracting 6 countries columns ('USA','RUS','CHN','FRA','AUS','GBR') from the dataframe and using it to plot the stacked bar plot.

<u>Visualization Description:</u>
For the above dataset we have 2 dependent variable (Country, Number of winners) and 1 independent variable (year). We want to know how the composition of 6 countries changes with the year therefore we used the stacked bar chart as it helps to illustrate the composition of a whole by showing the relative proportions of different categories that make up the total.

This makes it easy to compare the contribution of each category to the overall total. We can very easily see the variation in the number of winners for any country just by seeing plot. For example, we can see that the number of winners from the USA has reduced in 2023 wrt 90's.



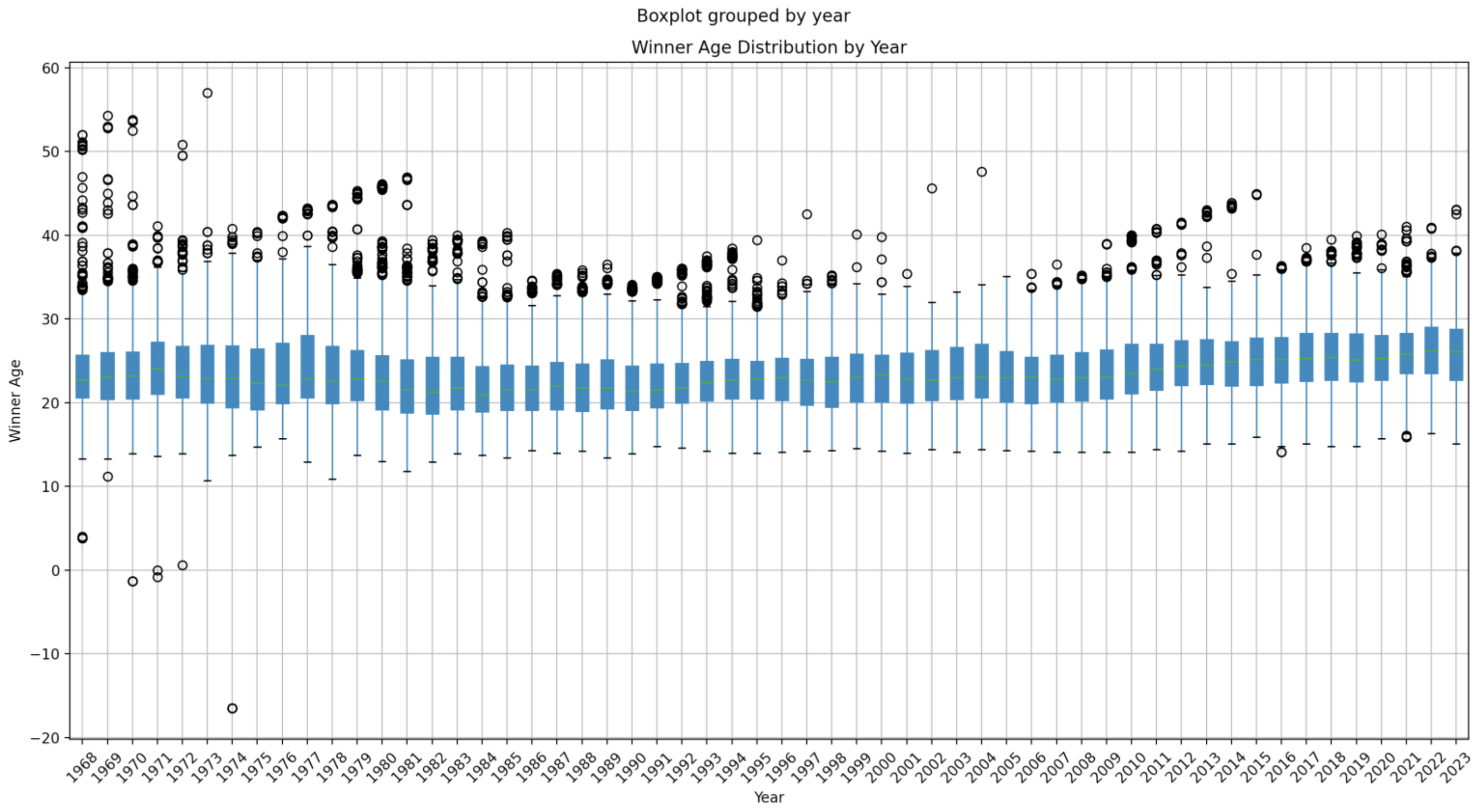


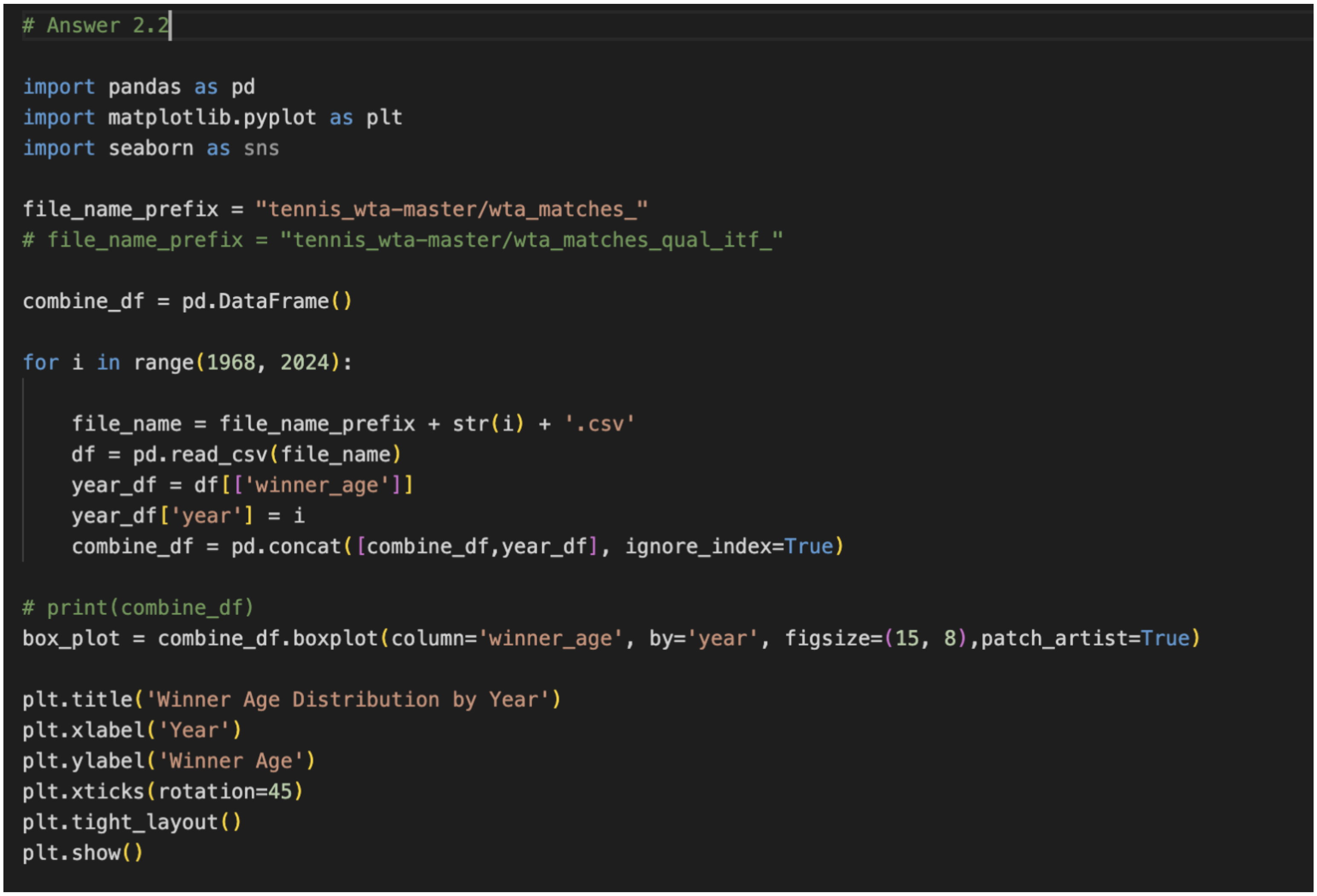
```python
# Answer 2.1

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import defaultdict, Counter

file_name_prefix = "tennis_wta-master/wta_matches_"
# file_name_prefix = "tennis_wta-master/wta_matches_qual_itf_"
country_winners = defaultdict(dict)

for i in range(1968,2024):
    file_name = file_name_prefix + str(i) + '.csv'

    df = pd.read_csv(file_name)
    country_winners[i] = Counter(df['winner_ioc'])

df = pd.DataFrame(country_winners).transpose()
df = df[['USA','RUS','CHN','FRA','AUS','GBR']]
# print(df)
df.plot(kind="bar", stacked=True)
plt.xlabel('Year')
plt.ylabel('Number of Winners')
plt.title('Winner Distribution of 6 countries with Year')
plt.legend(title='Country', loc='upper right')
plt.show()
```

<u>2.2 Type of Visualization</u>: Multiple Box plot

<u>Dataset Description:</u>
In this plot, I am using tennis matches data from year 1968 to 2023 stored in files 'wta_matches_1968' – 'wta_matches_2023'. I am extracting 'winner_age' columns from each file and appending them to a single data frame along with the 'year'. This dataframe I am using to plot multiple box plot showing age distribution of winners with year.

<u>Visualization Description:</u>
For the above dataset we have 1 dependent variable (age) and 1 independent variable (year). We are interested to know the trend in age distribution with year. Using multiple box plot helped us to understand how the distribution of age differs or remains consistent with year. Each box plot presents summary statistics such as median, mode and potential outliers for each group. This makes it easy to compare these statistics across year. By placing box plots side by side, the variability in the data with year becomes more apparent.





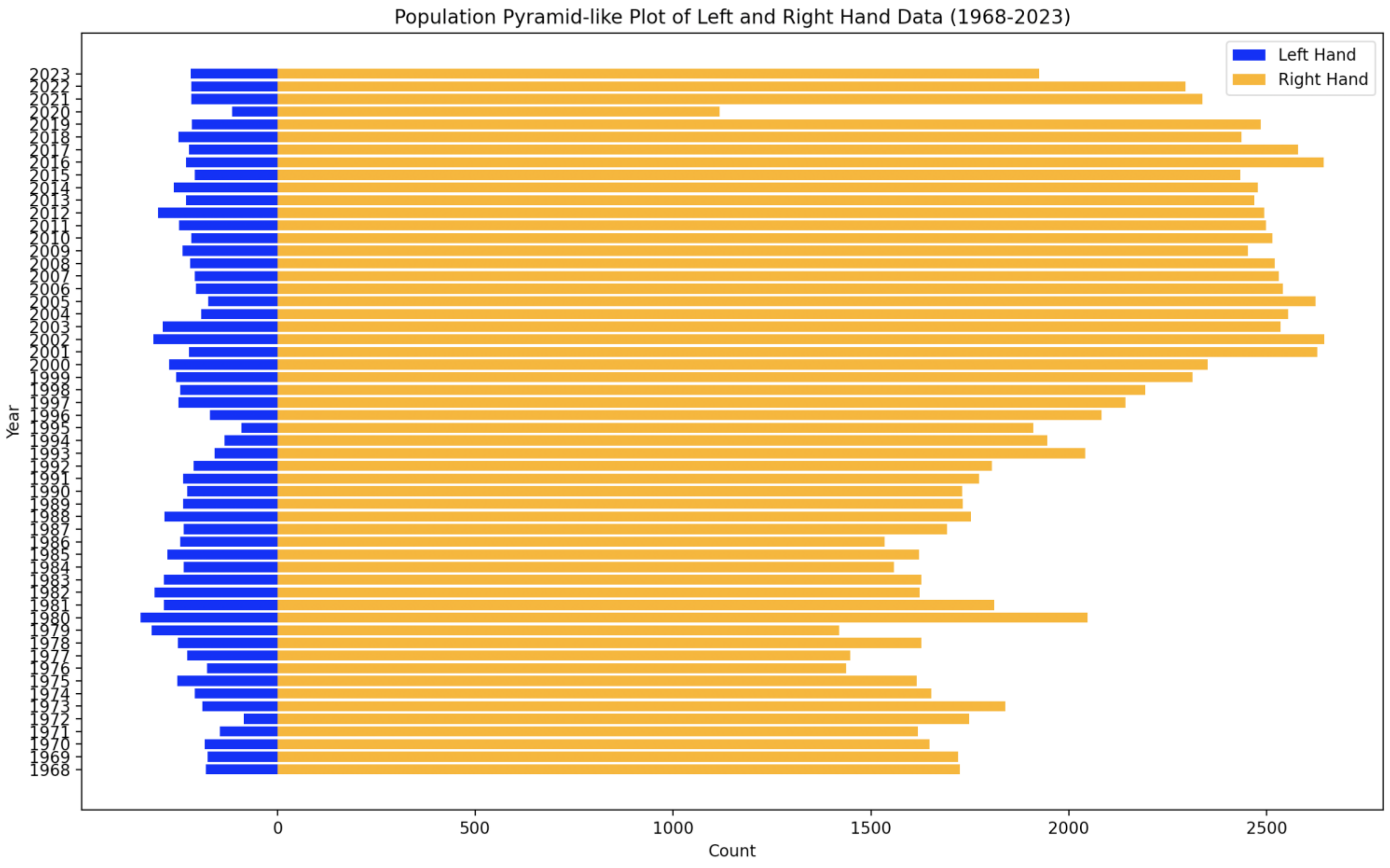
```
# Answer 2.2

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

file_name_prefix = "tennis_wta-master/wta_matches_"
# file_name_prefix = "tennis_wta-master/wta_matches_qual_itf_"

combine_df = pd.DataFrame()

for i in range(1968, 2024):

    file_name = file_name_prefix + str(i) + '.csv'
    df = pd.read_csv(file_name)
    year_df = df[['winner_age']]
    year_df['year'] = i
    combine_df = pd.concat([combine_df,year_df], ignore_index=True)

# print(combine_df)
box_plot = combine_df.boxplot(column='winner_age', by='year', figsize=(15, 8),patch_artist=True)

plt.title('Winner Age Distribution by Year')
plt.xlabel('Year')
plt.ylabel('Winner Age')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

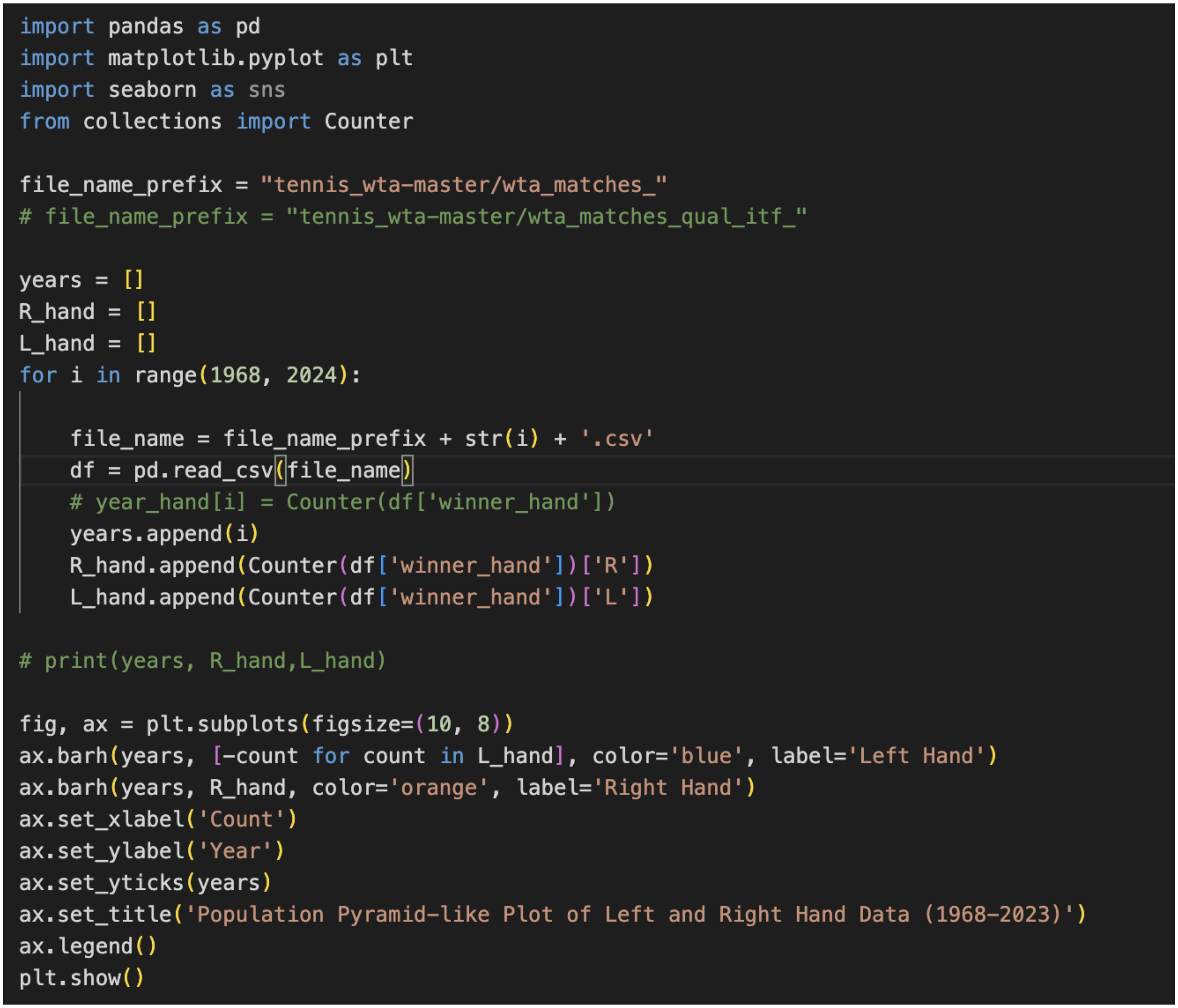
<u>2.3 Type of Visualization</u>: Population Pyramid

<u>Dataset Description:</u>
In this plot, I am using tennis matches data from year 1968 to 2023 stored in files 'wta_matches_1968' – 'wta_matches_2023'. I am appending number of winners who played with left hand in L_hand list and who played with right hand in 'R_hand' list for every year from 1968 to 2023. I am then using L_hand list to plot the left side of population pyramid by multiplying each element by -1. I am using the R_hand list to plot the right side of population pyramid.

Visualization Description:

For the above dataset we have 2 dependent variable (hand, count) and 1 independent variable (year). We usually use population pyramid to study the demographic data. Here we are interested to know the trend of the hand that winners used while playing. By presenting count and hand distributions side by side, a population pyramid allows for direct comparisons and insights into the relative sizes of left hand and right hand winners with year. We can even check the change in the number of winners of each category with year easily using below plot.





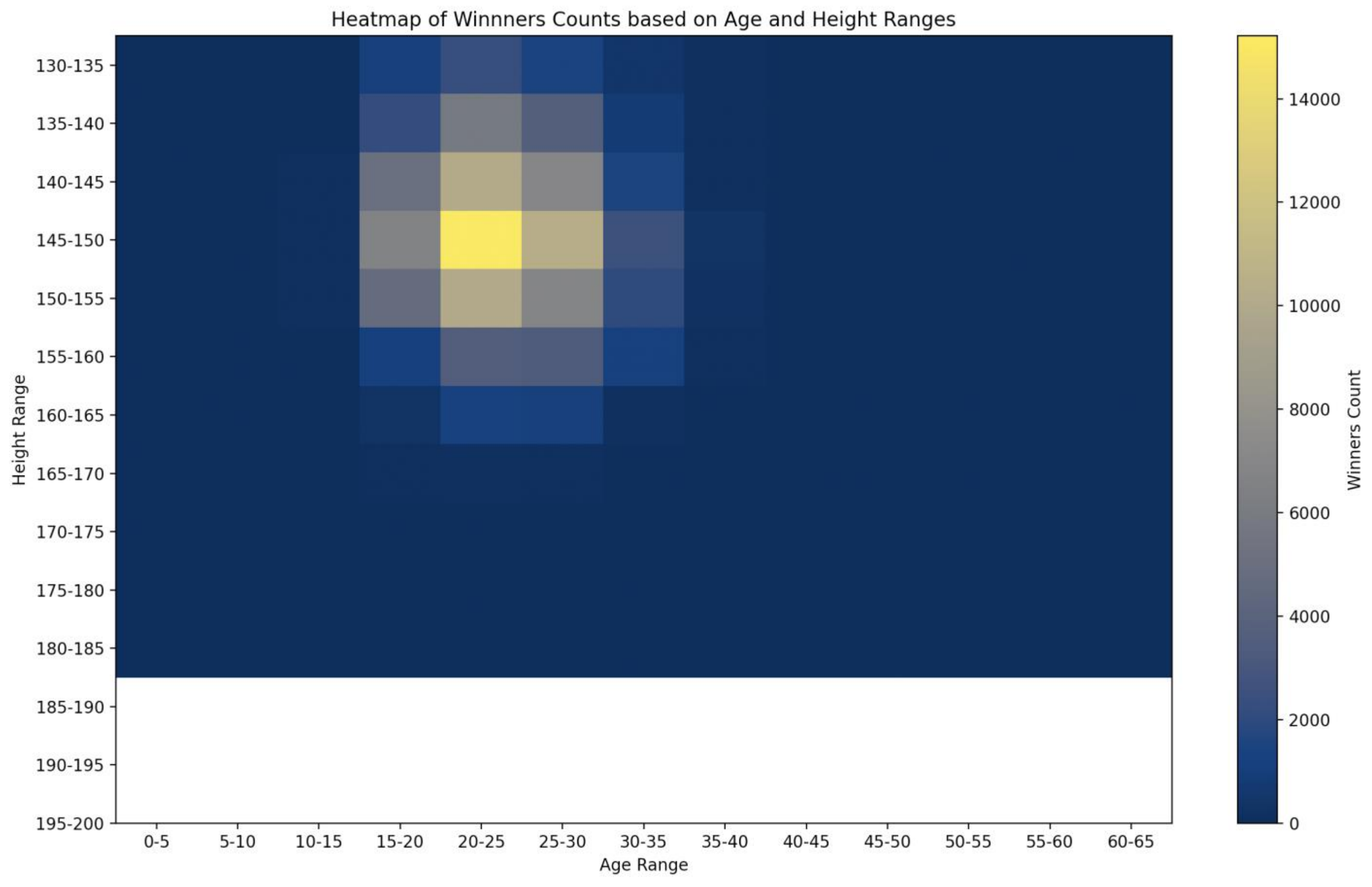
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

file_name_prefix = "tennis_wta-master/wta_matches_"
# file_name_prefix = "tennis_wta-master/wta_matches_qual_itf_"

years = []
R_hand = []
L_hand = []
for i in range(1968, 2024):

    file_name = file_name_prefix + str(i) + '.csv'
    df = pd.read_csv(file_name)
    # year_hand[i] = Counter(df['winner_hand'])
    years.append(i)
    R_hand.append(Counter(df['winner_hand'])['R'])
    L_hand.append(Counter(df['winner_hand'])['L'])

# print(years, R_hand,L_hand)

fig, ax = plt.subplots(figsize=(10, 8))
ax.barh(years, [-count for count in L_hand], color='blue', label='Left Hand')
ax.barh(years, R_hand, color='orange', label='Right Hand')
ax.set_xlabel('Count')
ax.set_ylabel('Year')
ax.set_yticks(years)
ax.set_title('Population Pyramid-like Plot of Left and Right Hand Data (1968-2023)')
ax.legend()
plt.show()
```

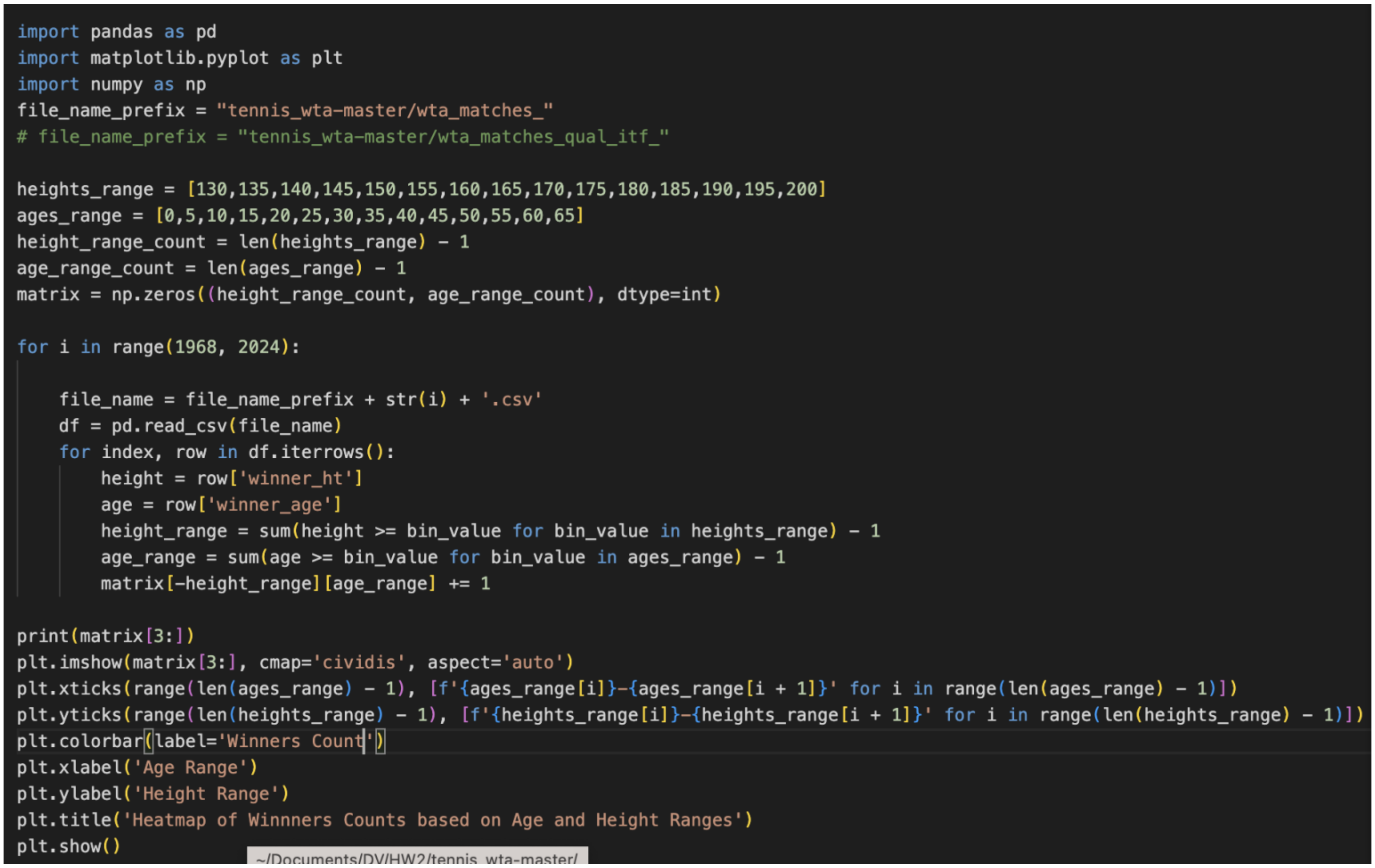
3. At least one from the plots of more variables

   Type of Visualization: Heat Map

   Dataset Description:
   In this plot, I am using tennis matches data from year 1968 to 2023 stored in files 'wta_matches_1968' – 'wta_matches_2023'. At first, I am creating a matrix to store the number of winners of every height-age pair. The rows of this matrix represent various height bins and the columns represent various age bins. I am extracting 'winner_ht' and 'winner_age' data for each winner and increasing its corresponding count in the matrix. I am then using this matrix to plot the head map.

   Visualization Description:
   For the above dataset, we have 3 dependent variables (height, age, count). We want to know if there is any correlation between height and age of winners. Heat maps help in recognizing patterns and trends in large datasets, which might be difficult to identify in raw numerical data or traditional tables. Using below heat map we can directly extract the insight that a vast majority of winners have ages in range 20-25 and heights in range 145-150. Heat maps are particularly effective for large datasets as they condense large data into a visually manageable representation. In the below case as well, we were dealing with thousands of data samples and a heat map helped to directly extract information from it.





```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
file_name_prefix = "tennis_wta-master/wta_matches_"
# file_name_prefix = "tennis_wta-master/wta_matches_qual_itf_"

heights_range = [130,135,140,145,150,155,160,165,170,175,180,185,190,195,200]
ages_range = [0,5,10,15,20,25,30,35,40,45,50,55,60,65]
height_range_count = len(heights_range) - 1
age_range_count = len(ages_range) - 1
matrix = np.zeros((height_range_count, age_range_count), dtype=int)

for i in range(1968, 2024):

    file_name = file_name_prefix + str(i) + '.csv'
    df = pd.read_csv(file_name)
    for index, row in df.iterrows():
        height = row['winner_ht']
        age = row['winner_age']
        height_range = sum(height >= bin_value for bin_value in heights_range) - 1
        age_range = sum(age >= bin_value for bin_value in ages_range) - 1
        matrix[-height_range][age_range] += 1

print(matrix[3:])
plt.imshow(matrix[3:], cmap='cividis', aspect='auto')
plt.xticks(range(len(ages_range) - 1), [f'{ages_range[i]}-{ages_range[i + 1]}' for i in range(len(ages_range) - 1)])
plt.yticks(range(len(heights_range) - 1), [f'{heights_range[i]}-{heights_range[i + 1]}' for i in range(len(heights_range) - 1)])
plt.colorbar(label='Winners Count')
plt.xlabel('Age Range')
plt.ylabel('Height Range')
plt.title('Heatmap of Winnners Counts based on Age and Height Ranges')
plt.show()
```

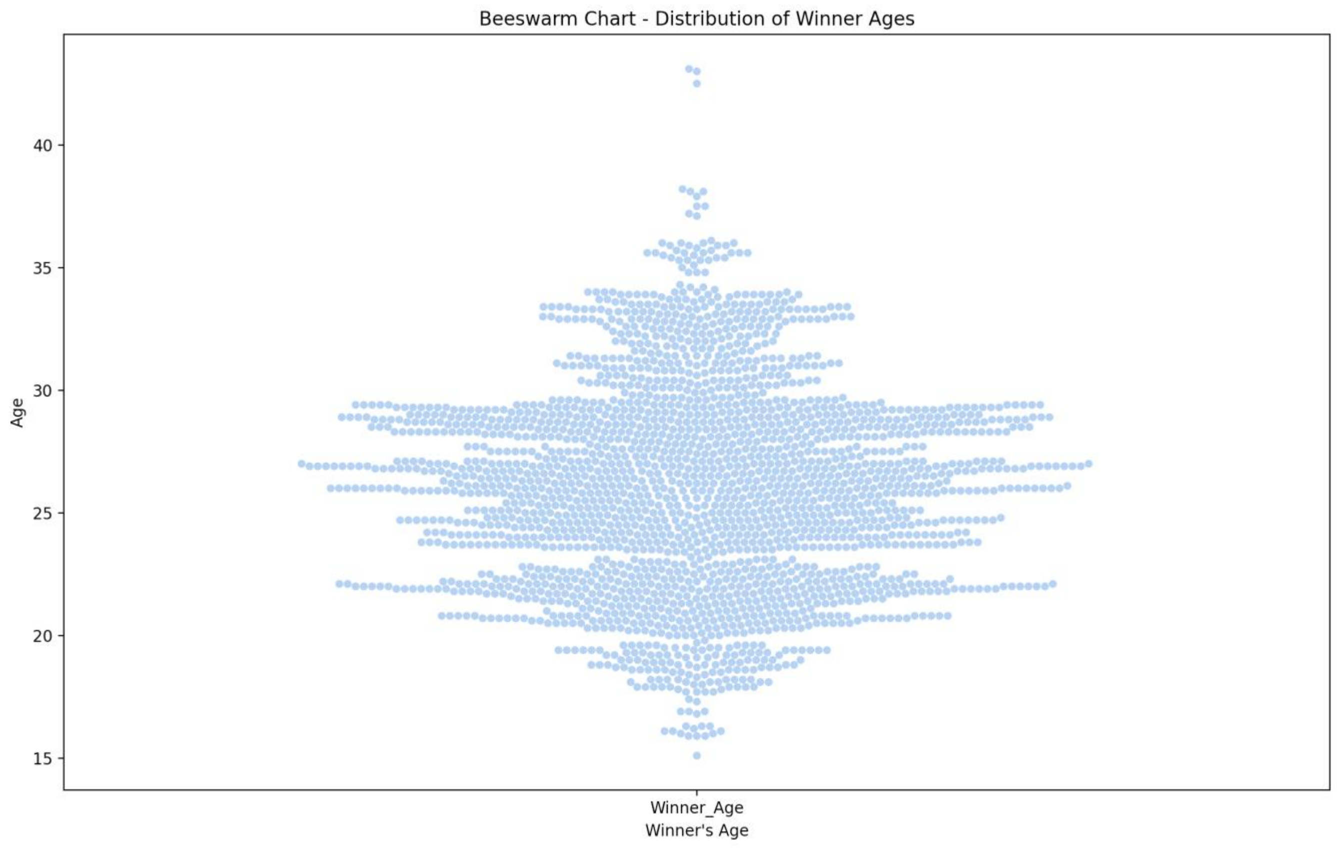
~/Documents/DV/HW2/tennis_wta-master/

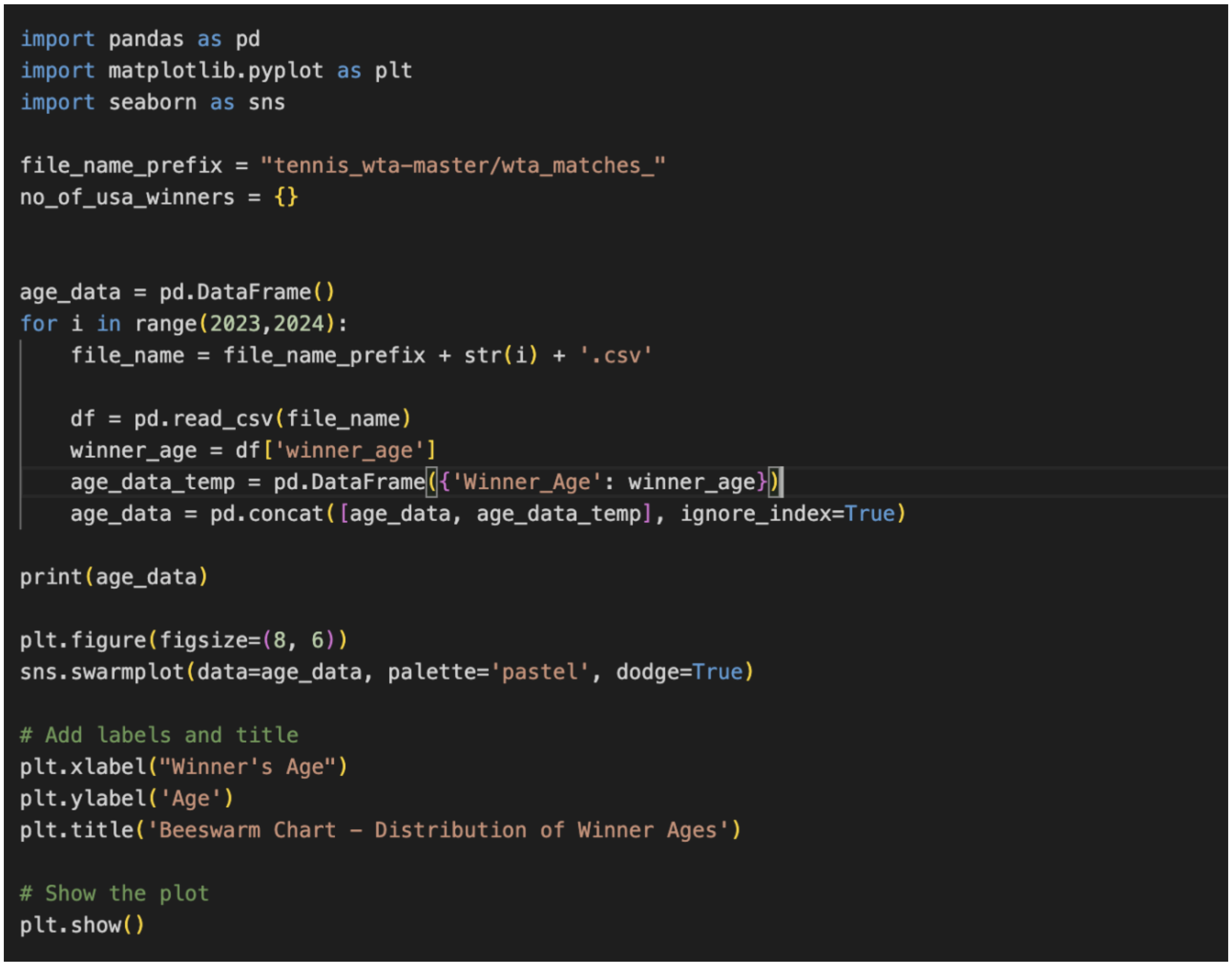
4. Plots of distributions of a variable: 2 points

   <u>Type of Visualization</u>: Beeswarm Chart

   <u>Dataset Description:</u>
   In this plot, I am using tennis matches data for year 2023 stored in file 'wta_matches_2023'. I am extracting 'winner_age' columns and using it to plot age distribution of winners in tennis matches in form of beeswarm plot.

   <u>Visualization Description:</u>
   For the above dataset we have 1 dependent variable (age). We want to know the distribution of age of tennis match winners. We used Beeswarm plots because it display individual data points, providing a detailed view of the data distribution and allowing for the identification of outliers or specific data patterns. Also, it arranges data points in a way that minimizes overlap and maintains visibility.





```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

file_name_prefix = "tennis_wta-master/wta_matches_"
no_of_usa_winners = {}


age_data = pd.DataFrame()
for i in range(2023,2024):
    file_name = file_name_prefix + str(i) + '.csv'

    df = pd.read_csv(file_name)
    winner_age = df['winner_age']
    age_data_temp = pd.DataFrame({'Winner_Age': winner_age})
    age_data = pd.concat([age_data, age_data_temp], ignore_index=True)

print(age_data)

plt.figure(figsize=(8, 6))
sns.swarmplot(data=age_data, palette='pastel', dodge=True)

# Add labels and title
plt.xlabel("Winner's Age")
plt.ylabel('Age')
plt.title('Beeswarm Chart - Distribution of Winner Ages')

# Show the plot
plt.show()
```

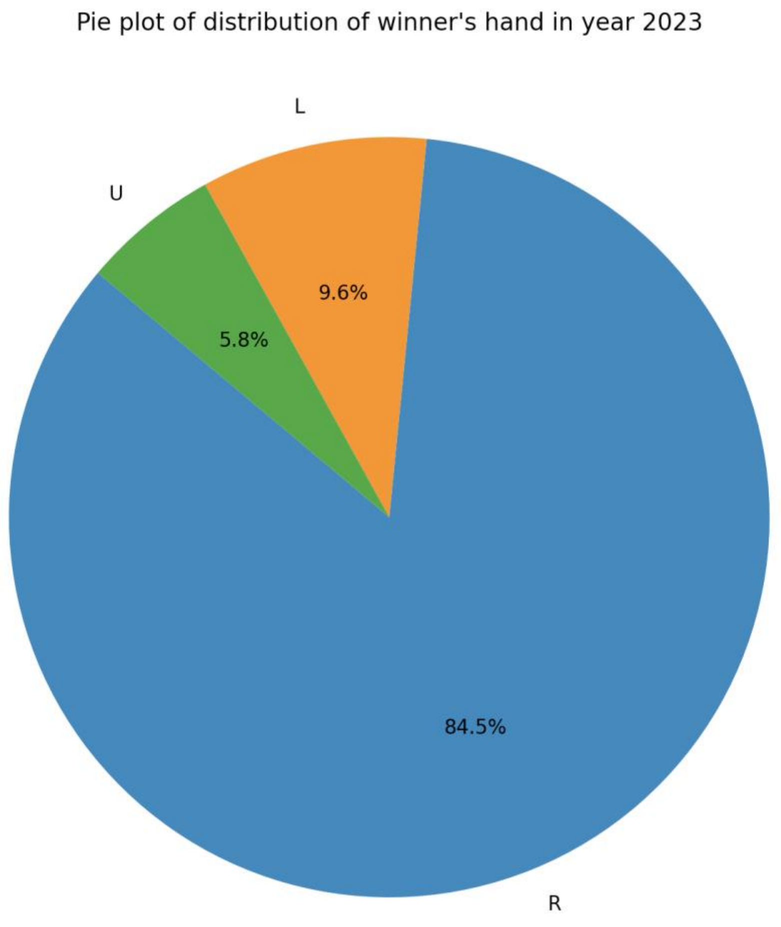
5. Plots of limited number of variables for those in the pie/bar/line category: 2 points

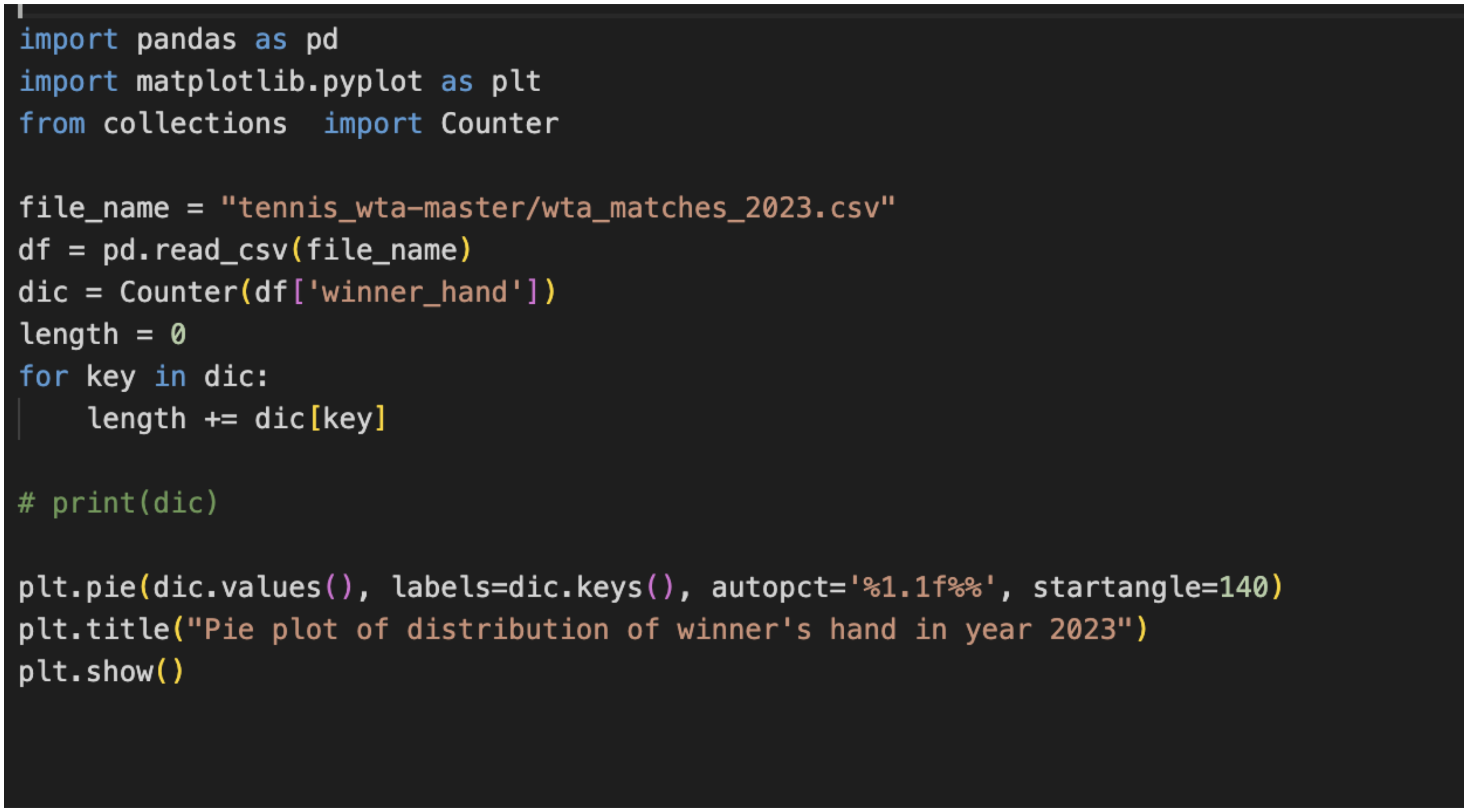
   Type of Visualization: Pie Chart

   Dataset Description:
   In this plot, I am using tennis matches data for year 2023 stored in file 'wta_matches_2023'. I am extracting 'winner_hand' columns and using it to plot hand distribution of winners in tennis matches in form of pie chart.

   Visualization Description:

   For the above dataset we have 2 dependent variable (hand and count). Here we are interested to know the composition of the hand that winners used while playing. Pie charts effectively display the contribution or proportion of each component to the total. This will help us to grasp the relative size of each part in relation to the whole.




```
import pandas as pd
import matplotlib.pyplot as plt
from collections  import Counter

file_name = "tennis_wta-master/wta_matches_2023.csv"
df = pd.read_csv(file_name)
dic = Counter(df['winner_hand'])
length = 0
for key in dic:
    length += dic[key]

# print(dic)

plt.pie(dic.values(), labels=dic.keys(), autopct='%1.1f%%', startangle=140)
plt.title("Pie plot of distribution of winner's hand in year 2023")
plt.show()
```

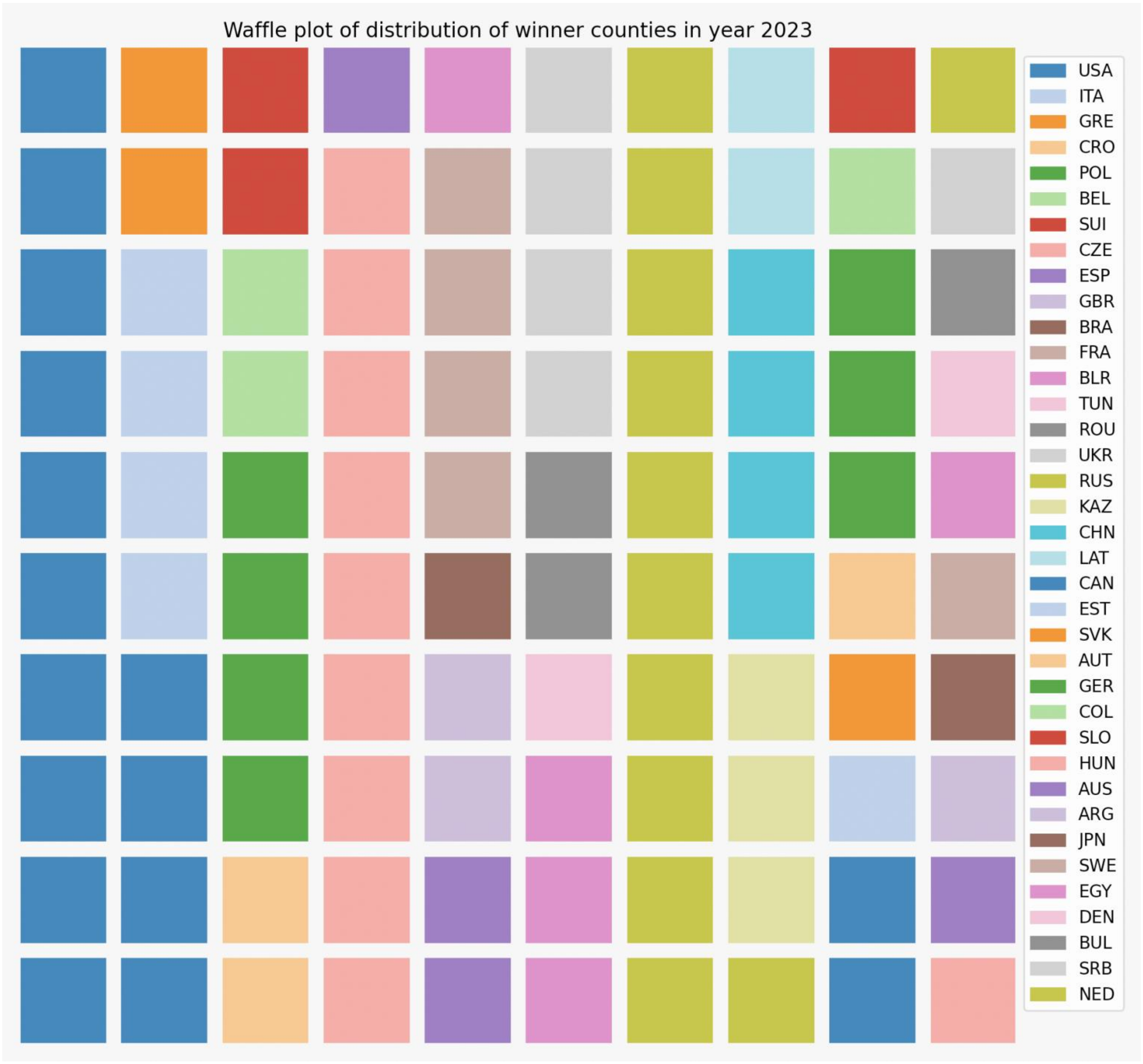
6. Plots of limited number of variables for those after the pie/bar/line category: 3 points

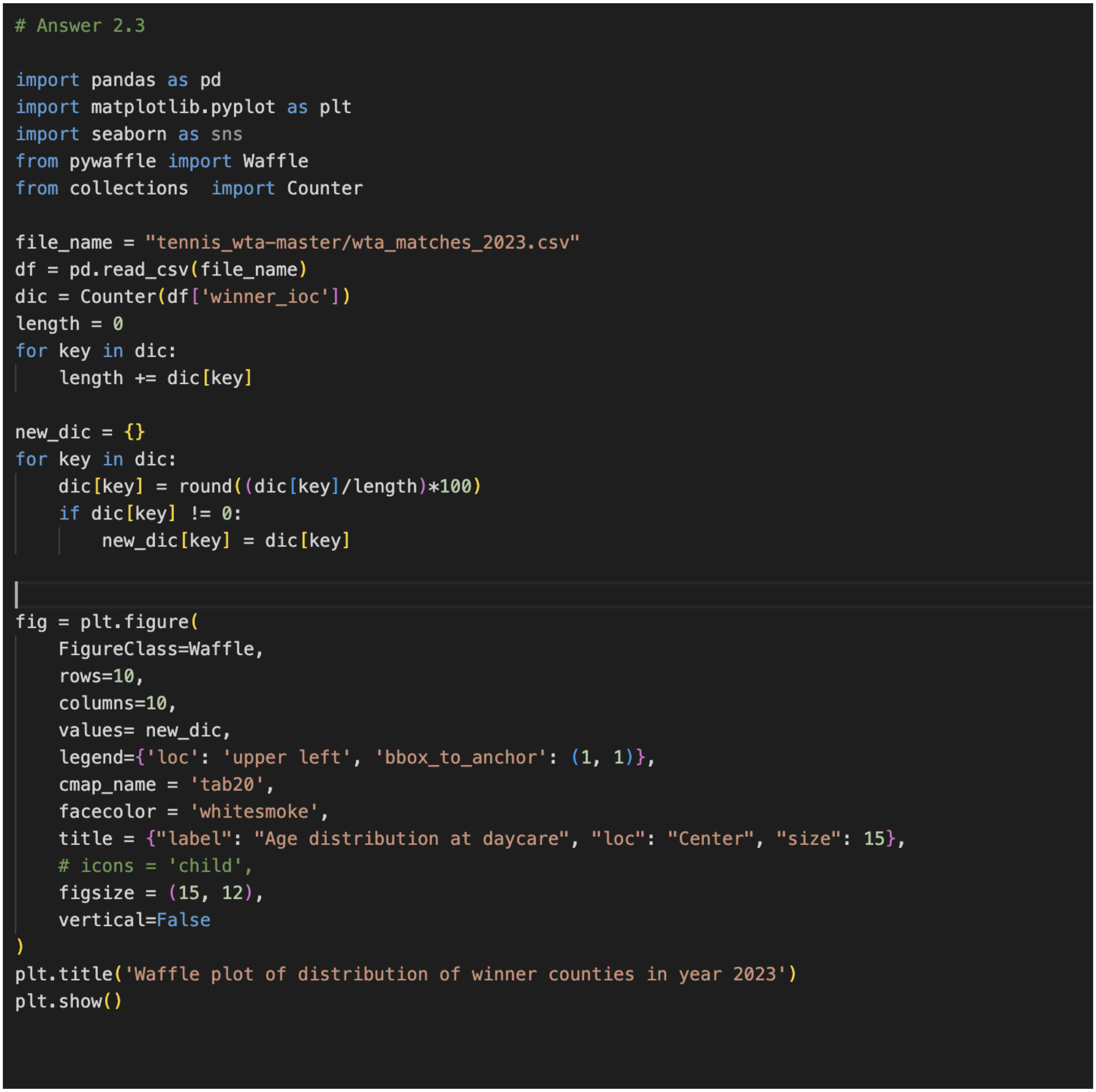
   Type of Visualization: Waffle Chart

   Dataset Description:
   In this plot, I am using tennis matches data for year 2023 stored in file 'wta_matches_2023'. I am extracting 'winner_ioc' columns and using it to plot winner countries distribution in tennis matches in form of Waffle chart.

   Visualization Description:

   For the above dataset we have 2 dependent variables (Country, Number of winners). We want to know the composition of winner countries in the year 2023. We used waffle chart as it enable straightforward comparison of proportions or percentages between different categories. This allows us to easily visualize sizes and contributions.


Waffle plot of distribution of winner counties in year 2023
USA
ITA
GRE
CRO
POL
BEL
SUI
CZE
ESP
GBR
BRA
FRA
BLR
TUN
ROU
UKR
RUS
KAZ
CHN
LAT
CAN
EST
SVK
AUT
GER
COL
SLO
HUN
AUS
ARG
JPN
SWE
EGY
DEN
BUL
SRB
NED



```python
# Answer 2.3

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pywaffle import Waffle
from collections  import Counter

file_name = "tennis_wta-master/wta_matches_2023.csv"
df = pd.read_csv(file_name)
dic = Counter(df['winner_ioc'])
length = 0
for key in dic:
    length += dic[key]

new_dic = {}
for key in dic:
    dic[key] = round((dic[key]/length)*100)
    if dic[key] != 0:
        new_dic[key] = dic[key]


fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=10,
    values= new_dic,
    legend={'loc': 'upper left', 'bbox_to_anchor': (1, 1)},
    cmap_name = 'tab20',
    facecolor = 'whitesmoke',
    title = {"label": "Age distribution at daycare", "loc": "Center", "size": 15},
    # icons = 'child',
    figsize = (15, 12),
    vertical=False
)
plt.title('Waffle plot of distribution of winner counties in year 2023')
plt.show()
```

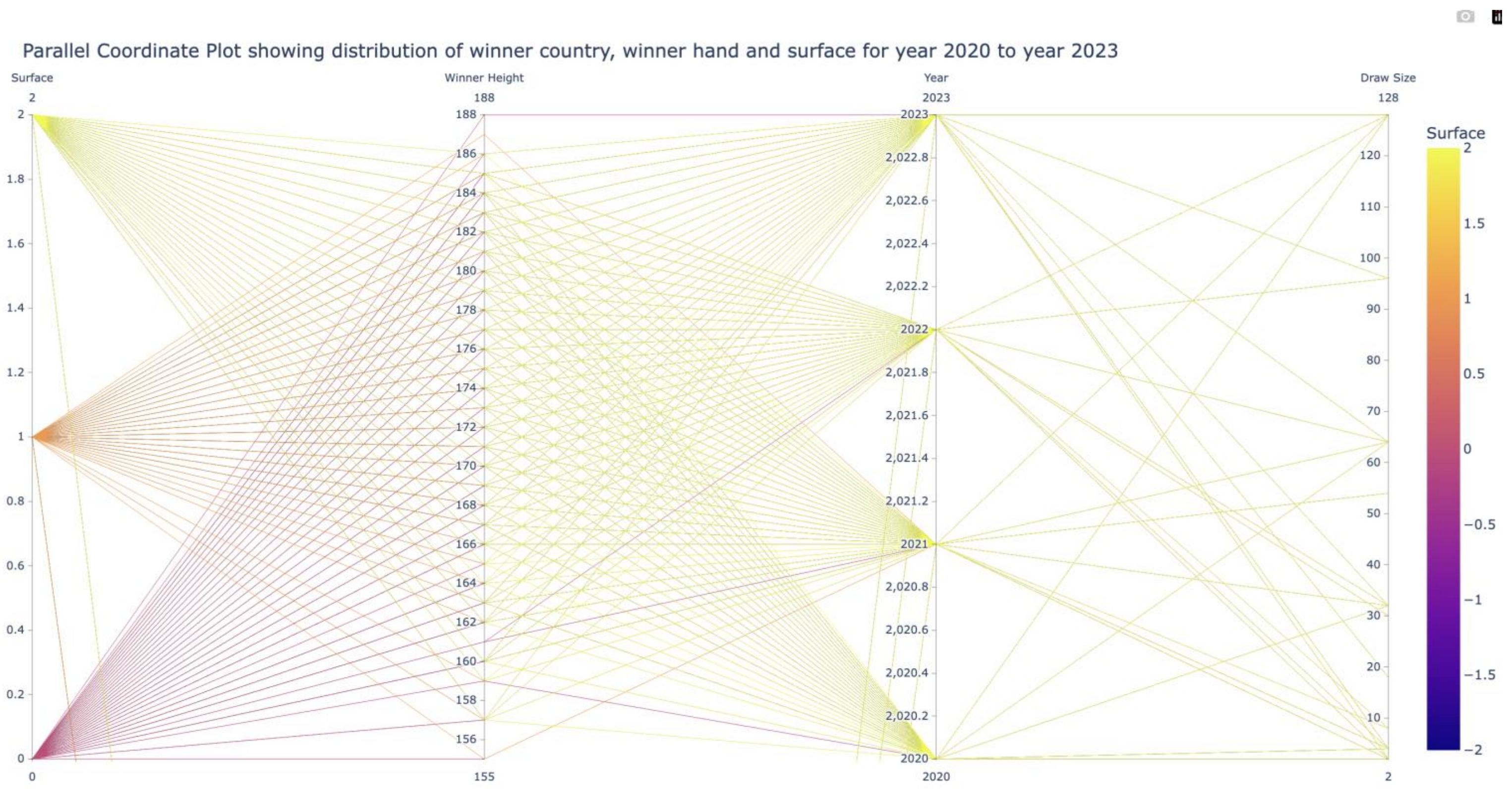
7. Plots of more variables: 4 points

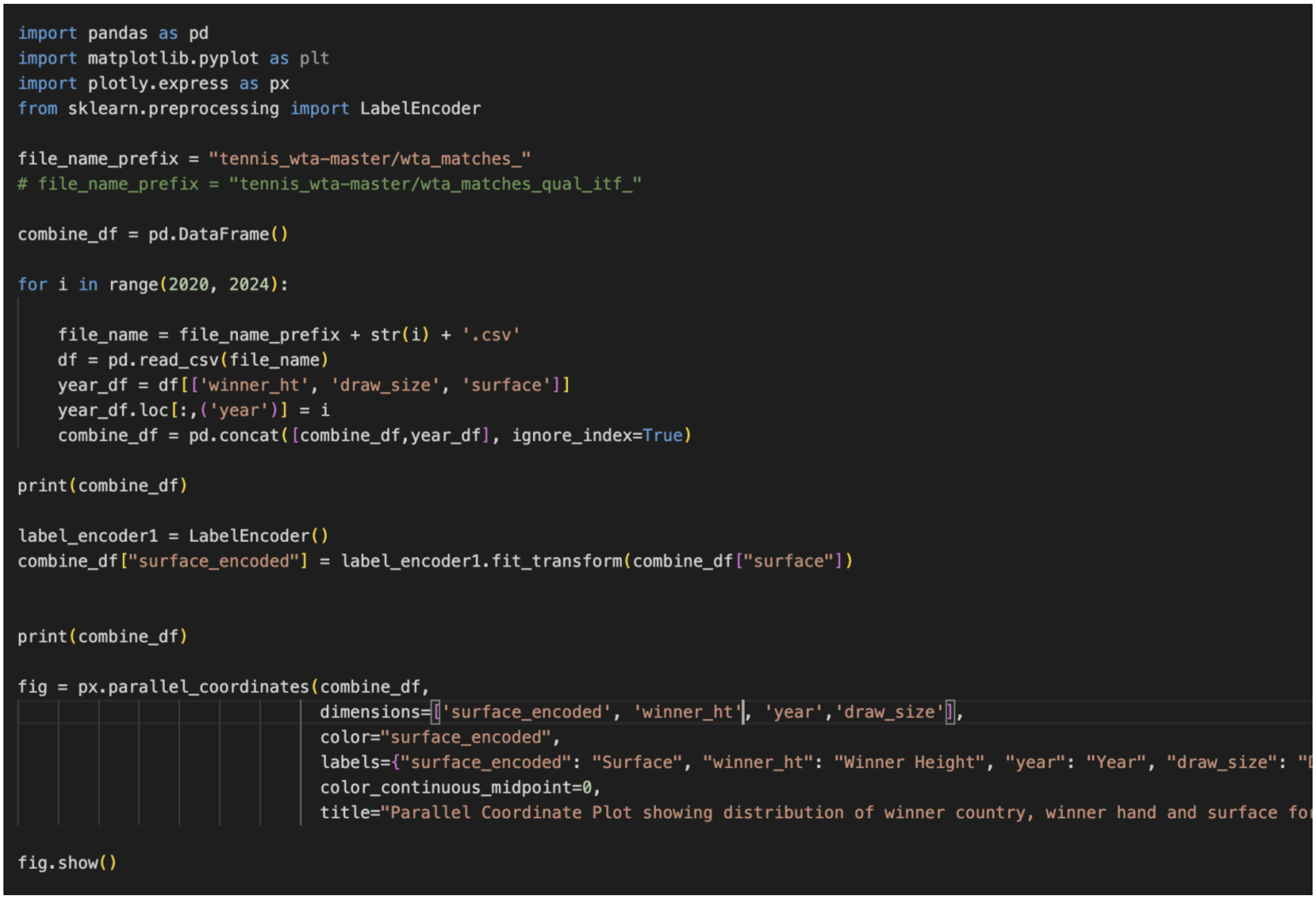
Type of Visualization: Parallel Coordinates

Dataset Description:
In this plot, I am using tennis matches data from year 2020 to 2023 stored in files 'wta_matches_2020' – 'wta_matches_2023'. I am extracting 'surface', 'winner_ht' and 'draw size' columns from each file and appending them to a single data frame along with the 'year'. This dataframe I am using to plot parallel coordinate plot.

Visualization Description:

For the above dataset, we have 4 dependent variables (surface, winner height, year, draw size). We want to know the flow and relation between all 4 variables therefore we used parallel coordinate plot since it can handle multivariate data effectively. They allow visualization of patterns and relationships between multiple variables simultaneously. By observing the patterns of the lines it becomes easy to identify patterns.





```python
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import LabelEncoder

file_name_prefix = "tennis_wta-master/wta_matches_"
# file_name_prefix = "tennis_wta-master/wta_matches_qual_itf_"

combine_df = pd.DataFrame()

for i in range(2020, 2024):

    file_name = file_name_prefix + str(i) + '.csv'
    df = pd.read_csv(file_name)
    year_df = df[['winner_ht', 'draw_size', 'surface']]
    year_df.loc[:,('year')] = i
    combine_df = pd.concat([combine_df,year_df], ignore_index=True)

print(combine_df)

label_encoder1 = LabelEncoder()
combine_df["surface_encoded"] = label_encoder1.fit_transform(combine_df["surface"])


print(combine_df)

fig = px.parallel_coordinates(combine_df,
                              dimensions=['surface_encoded', 'winner_ht', 'year','draw_size'],
                              color="surface_encoded",
                              labels={"surface_encoded": "Surface", "winner_ht": "Winner Height", "year": "Year", "draw_size": "D
                              color_continuous_midpoint=0,
                              title="Parallel Coordinate Plot showing distribution of winner country, winner hand and surface fo

fig.show()
```