

Intermediate Python & a Glimpse into AI applications

Class 3

Pepe Bonet Giner

12th January 2024

Index Class 3

Topic 1: Recap

Topic 2: Data Preparation - Data Exploration

Topic 3: Matplotlib & Seaborn

Topic 4: Data Exploration & Visualization

Topic 5: Feature Engineering

Topic 1: Recap

Recap Class 2: Pandas

Groupby

Apply

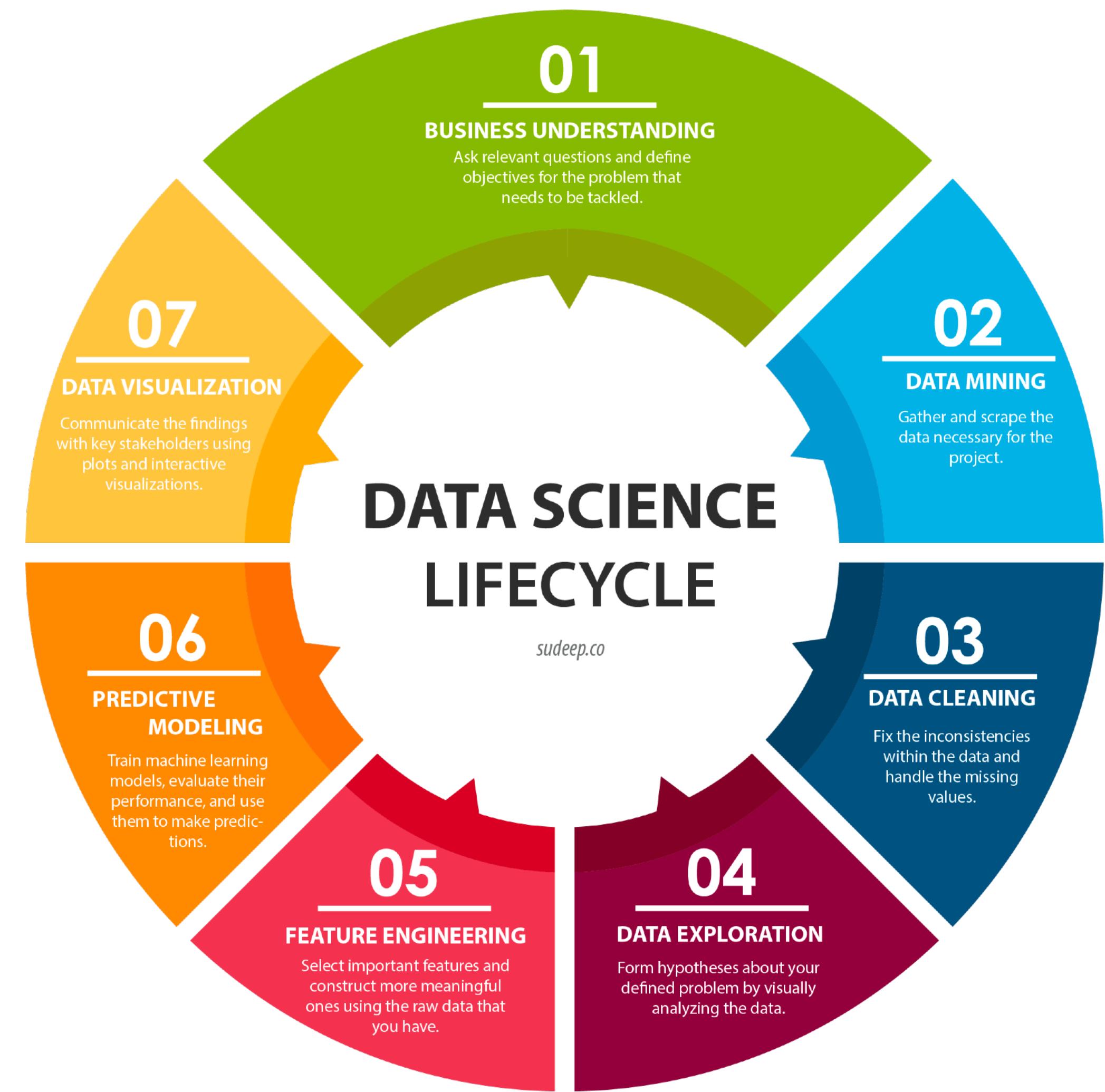
Lambda



Groupby + Apply + Lambda



Recap Class 2: The Data Science Lifecycle



Recap Class 2: Data Cleaning



- Step 1: Remove irrelevant data
- Step 2: Deduplicate your data
- Step 3: Fix structural errors
- Step 4: Deal with missing data
- Step 5: Filter out data outliers
- Step 6: Validate your data

Recap Class 2: Comments Exercises

- Be careful when you use drop(). If you run that cell twice it will give you an error.

```
/usr/local/lib/python3.8/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
  6015         if mask.any():
  6016             if errors != "ignore":
-> 6017                 raise KeyError(f"{labels[mask]} not found in axis")
  6018             indexer = indexer[~mask]
  6019         return self.delete(indexer)

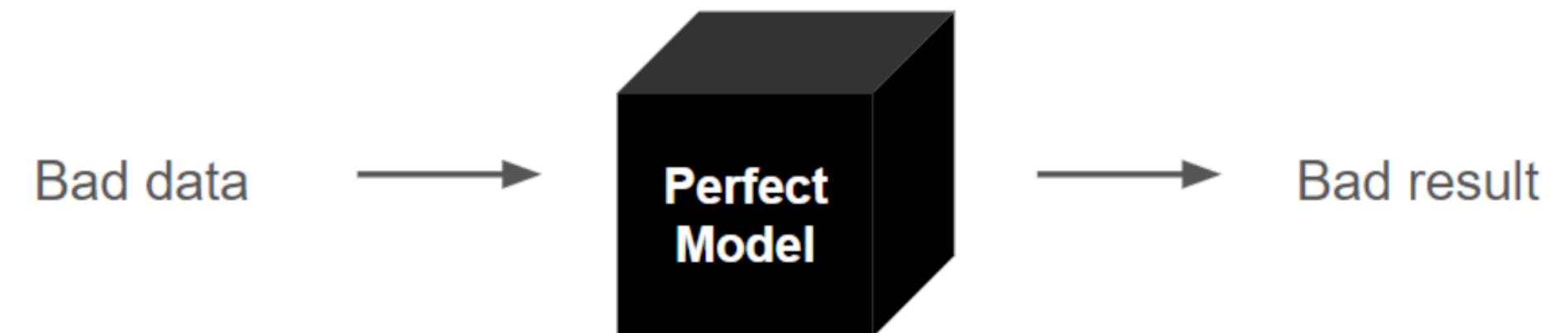
KeyError: "[ 'address' ] not found in axis"
```

Try to understand the errors that pop out

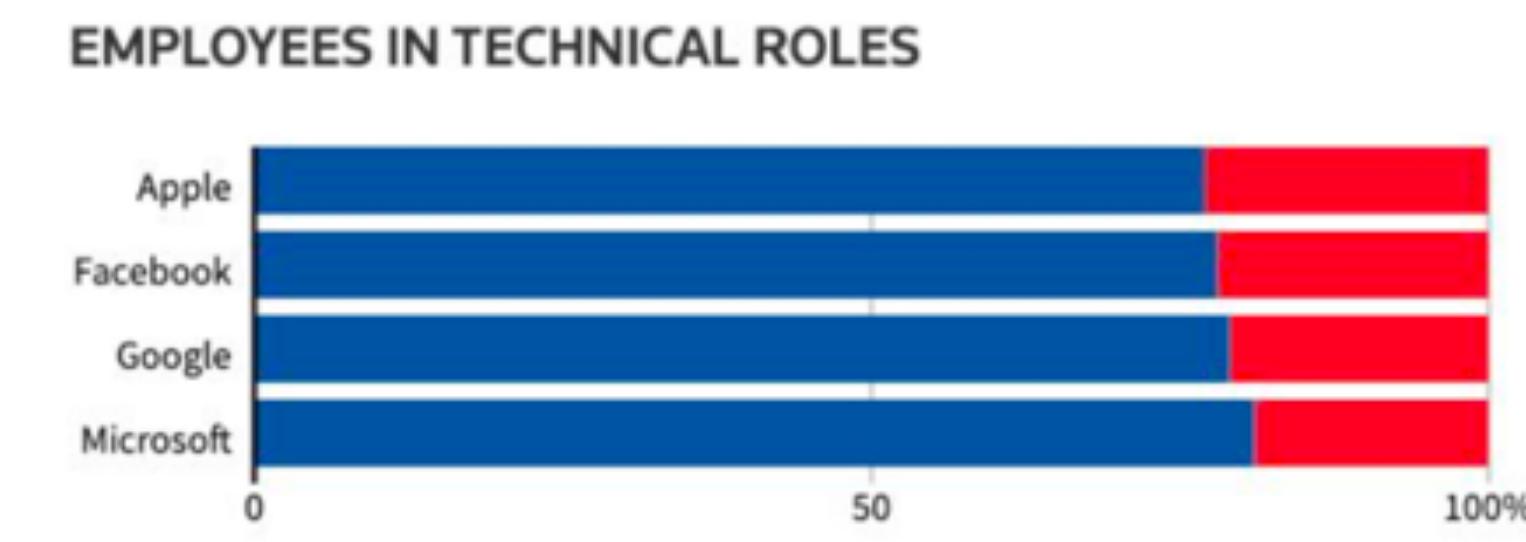
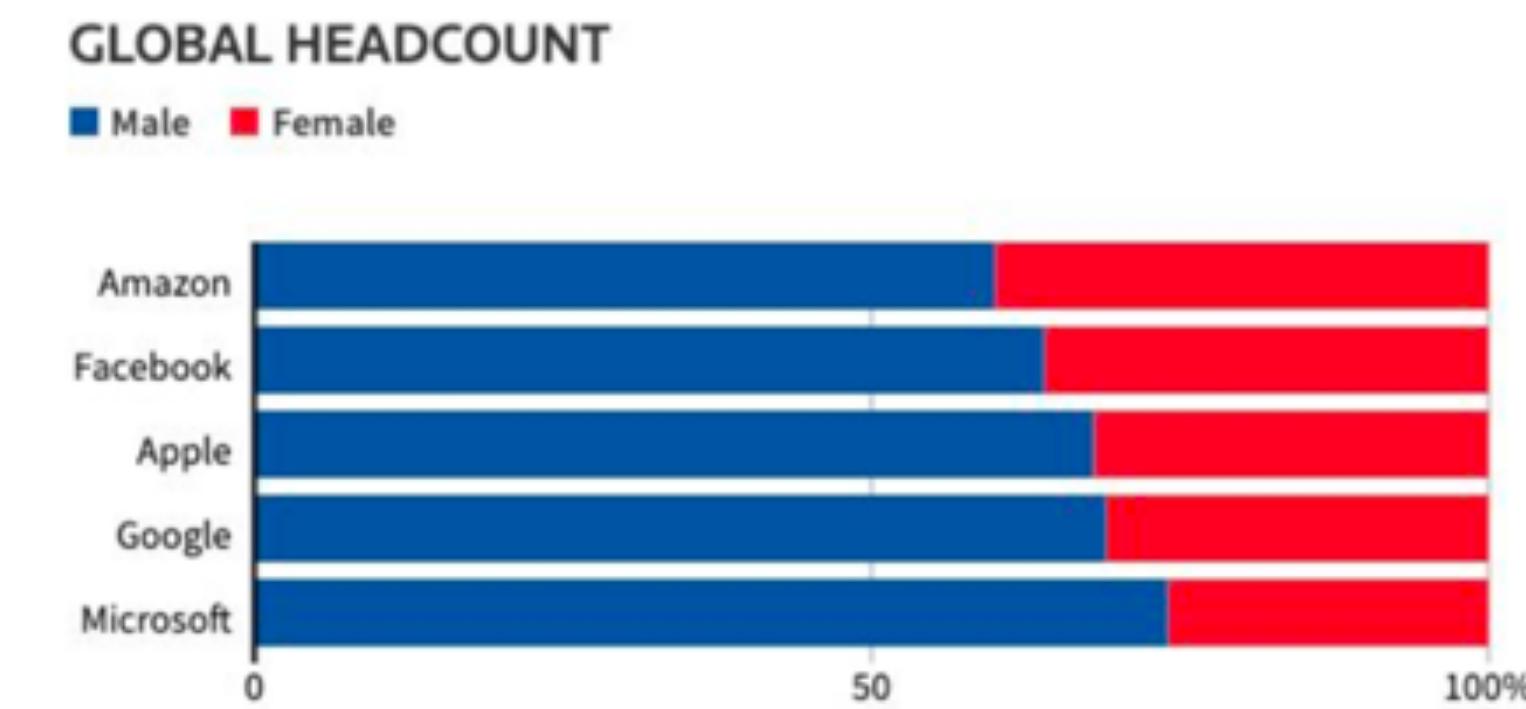
new_df =
Next line of code: df...

Topic 2: Data Preparation - Data Exploration

Data Exploration - Exploratory data analysis (EDA)



- A process where users look at and understand their data with statistical and visualization methods.
- Identify patterns and problems in the dataset
- Decide which model or algorithm to use in subsequent steps.



Preliminary data processing

Types of data

Numerical Data

Categorical Data

Year: 2022, 2021...
Grade: 90, 75, 45...
Age: 17, 25, 98...

Gender: Female, Male...
Race: A, B, C, D...
Parental Education:
HS, College, None...

Preliminary data processing

- Changes to the types of data of a column, i.e.:
 - Numerical to a string: int —> str
 - String to a date time: str —> datetime

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 962 entries, 0 to 1009
Data columns (total 11 columns):
 #   Column           Non-Null Count   Dtype  
--- 
 0   gender          962 non-null     object  
 1   race/ethnicity  962 non-null     object  
 2   parental level of education 962 non-null     object  
 3   lunch           962 non-null     object  
 4   test preparation course 962 non-null     object  
 5   math score      962 non-null     int64  
 6   reading score   962 non-null     int64  
 7   writing score   962 non-null     int64  
 8   id_student      962 non-null     int64  
 9   Year            962 non-null     int64  
 10  Age             962 non-null     float64
dtypes: float64(1), int64(5), object(5)
memory usage: 90.2+ KB
```

Exercise

- Change the data type of the id_student column from int to str

EDA steps

- Categorical EDA
- Numerical EDA
 - Univariate analysis
 - Bivariate analysis
(Categorical + Numerical)
 - Multivariate analysis

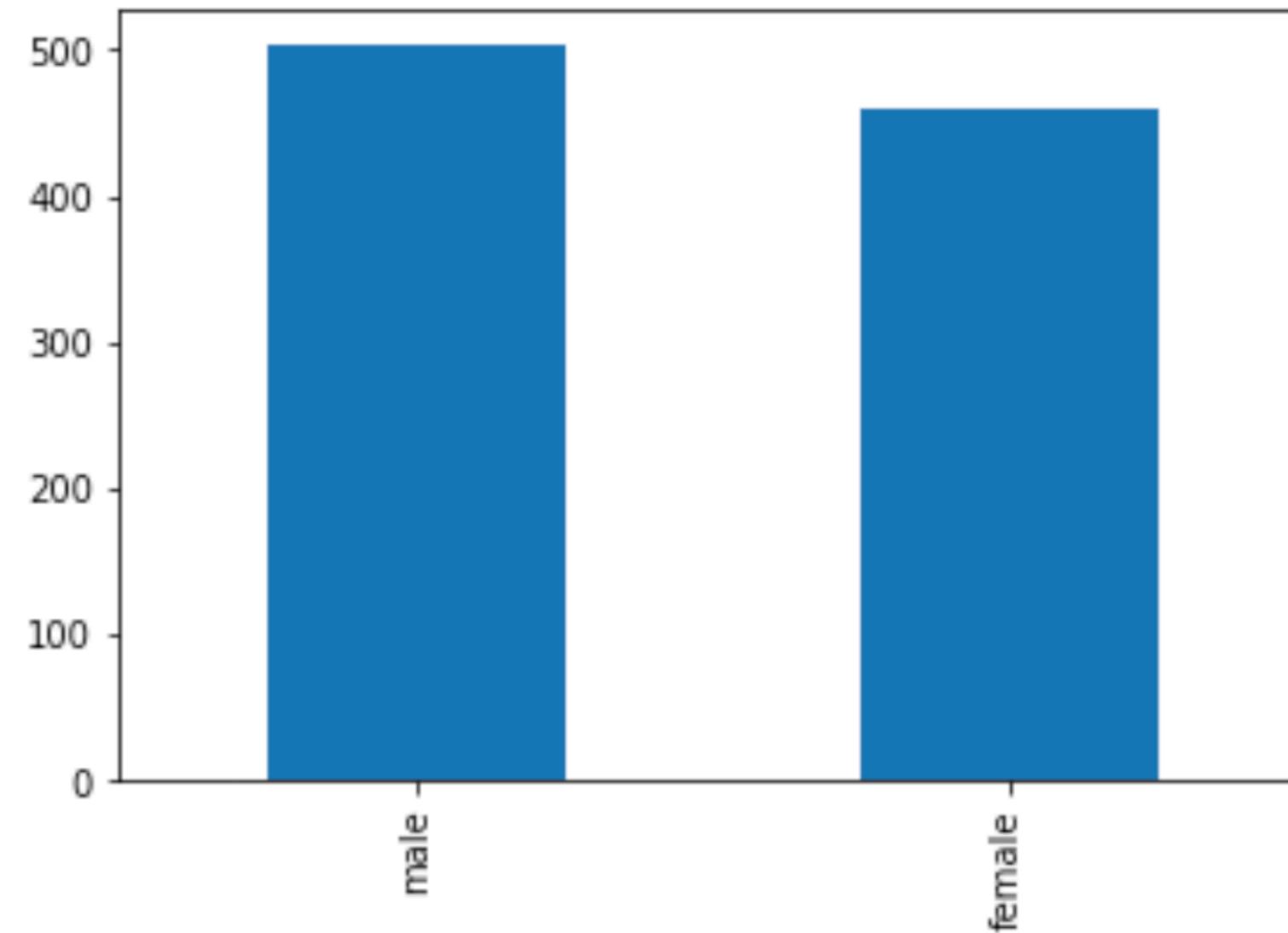


Categorical EDA

- Initial step: Explore group counts of categorical columns

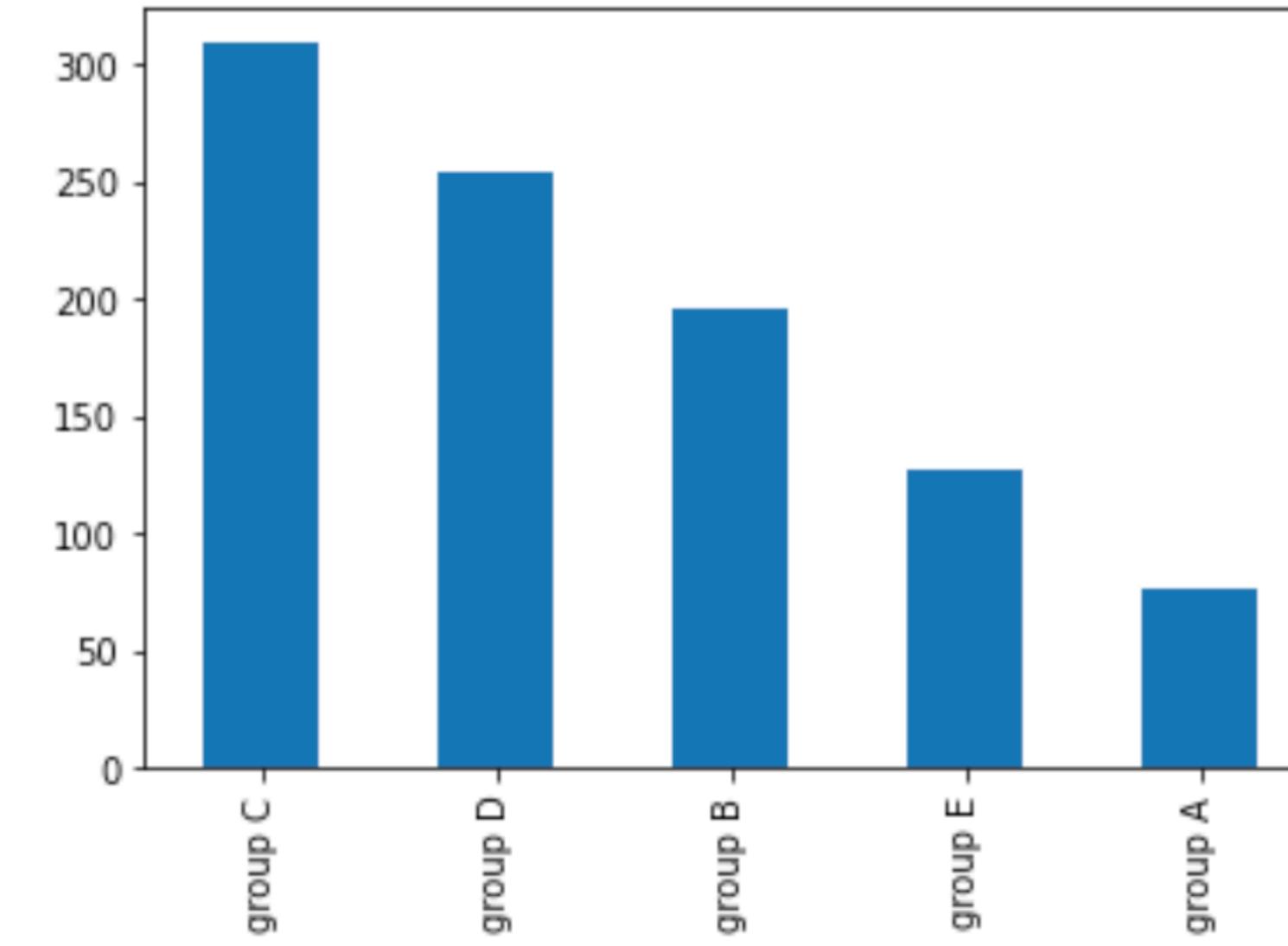
```
df['gender'].value_counts().plot(kind='bar')
```

<AxesSubplot:>



```
df['race/ethnicity'].value_counts().plot(kind='bar')
```

<AxesSubplot:>



Exercise

- Get a bar plot showing the number of students who completed the test preparation course and those who did not.

Numerical EDA

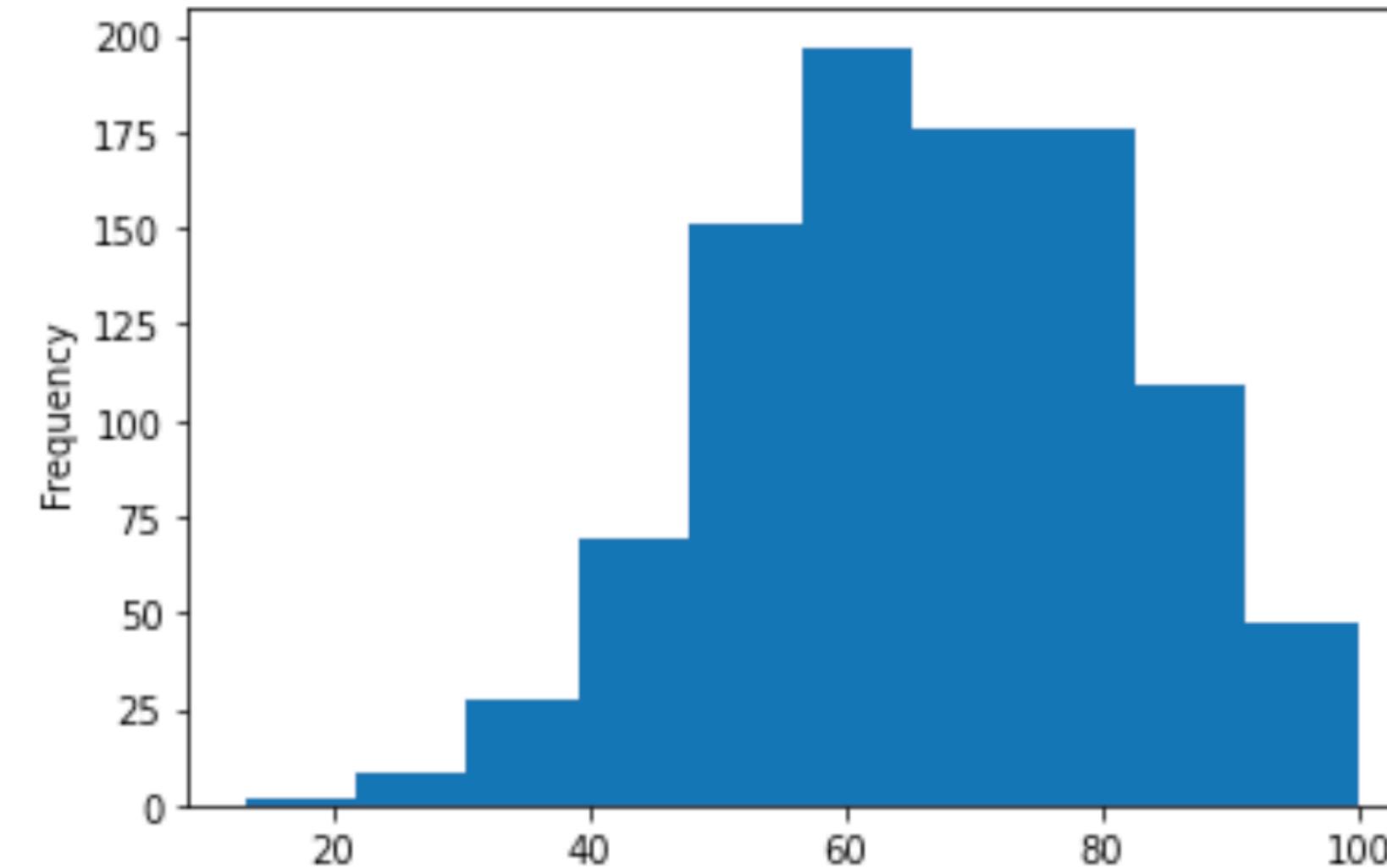
- How is the numerical data distributed?
- Can we find anything interesting?

```
df.describe()
```

| | math score | reading score |
|-------|------------|---------------|
| count | 962.000000 | 962.000000 |
| mean | 66.426195 | 68.943867 |
| std | 15.396706 | 14.768592 |
| min | 13.000000 | 27.000000 |
| 25% | 55.250000 | 59.000000 |
| 50% | 67.000000 | 70.000000 |
| 75% | 77.000000 | 79.000000 |
| max | 100.000000 | 100.000000 |

```
df['math score'].plot(kind='hist')
```

```
<AxesSubplot:ylabel='Frequency'>
```



Exercise

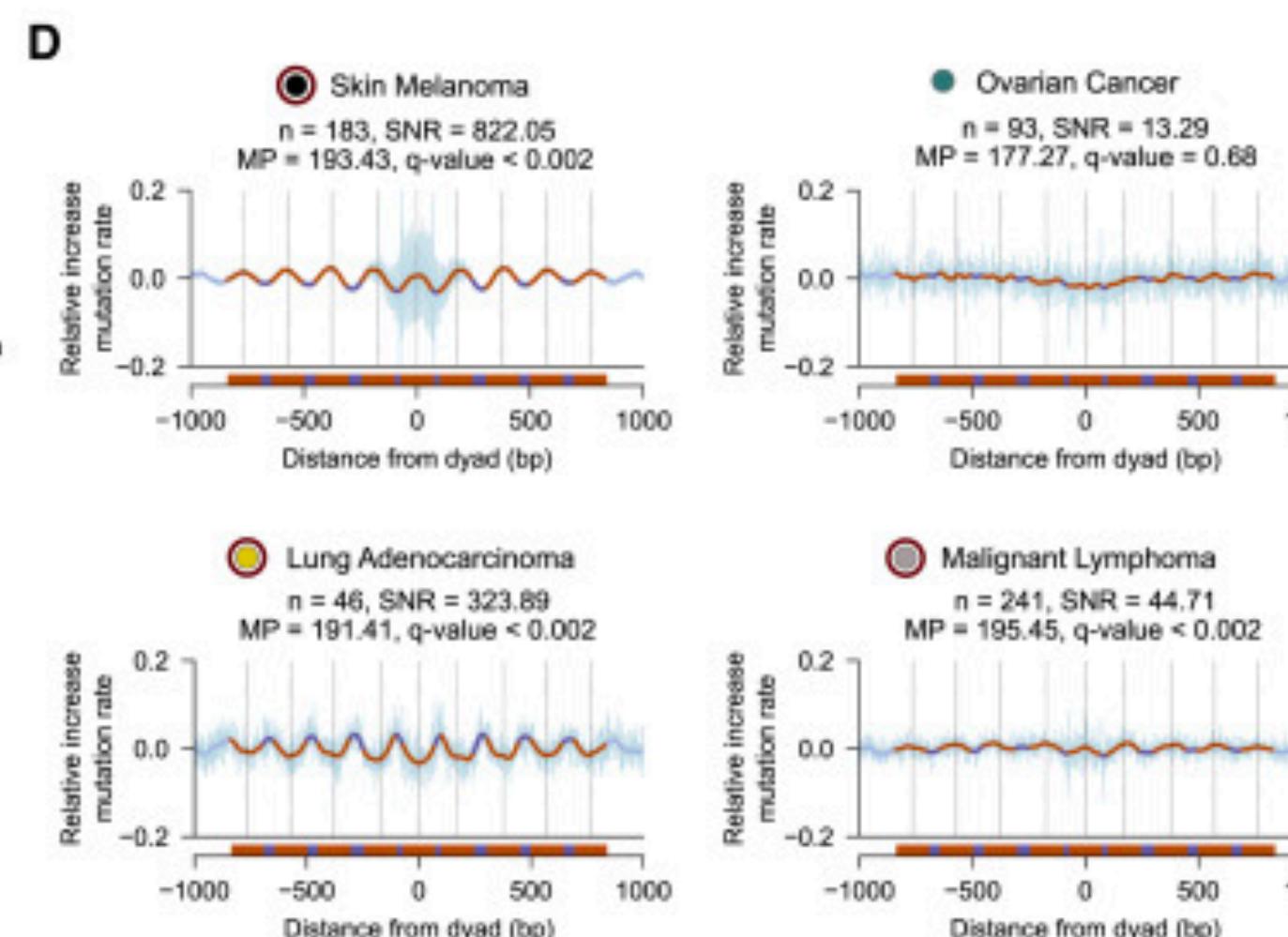
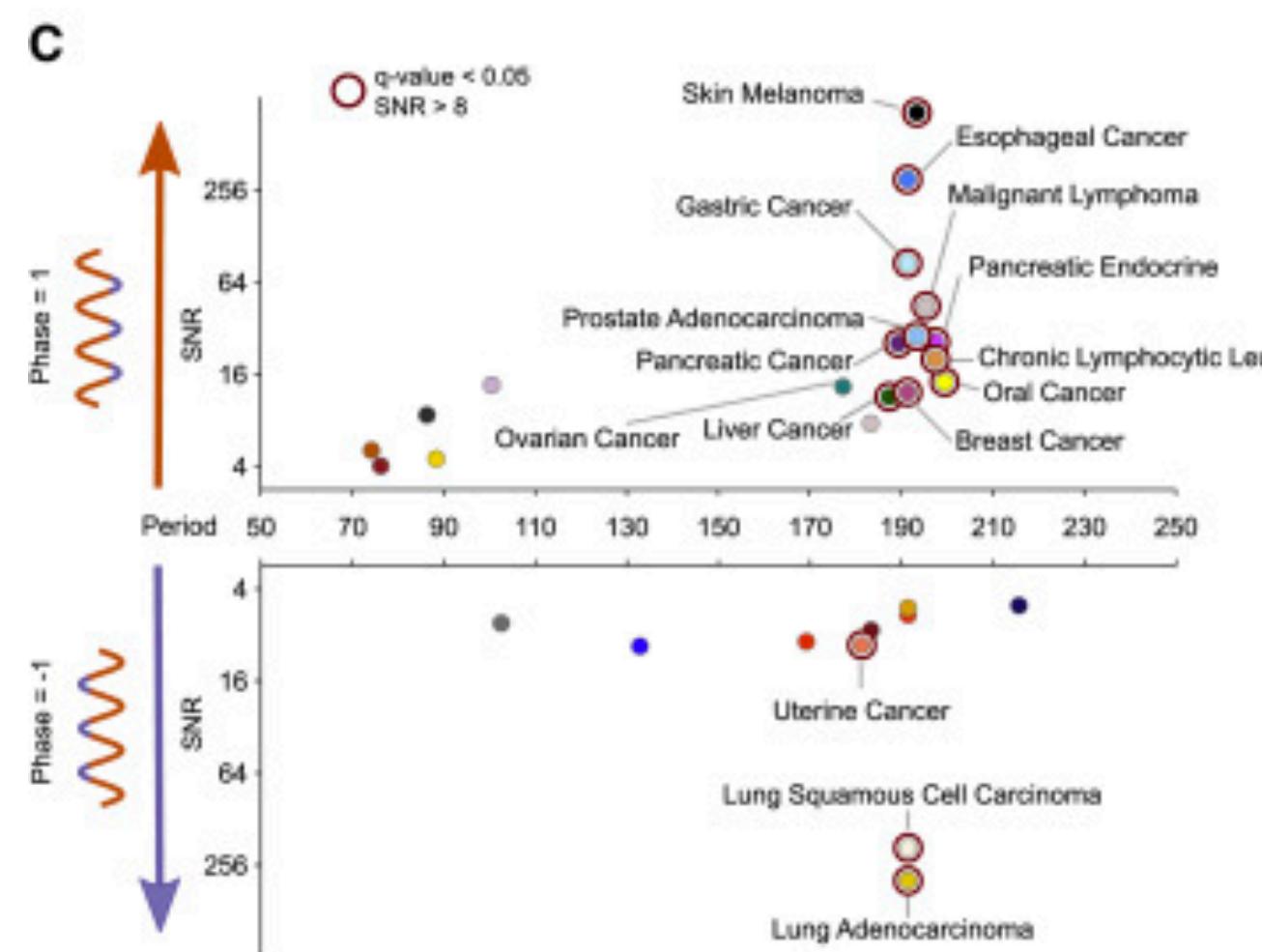
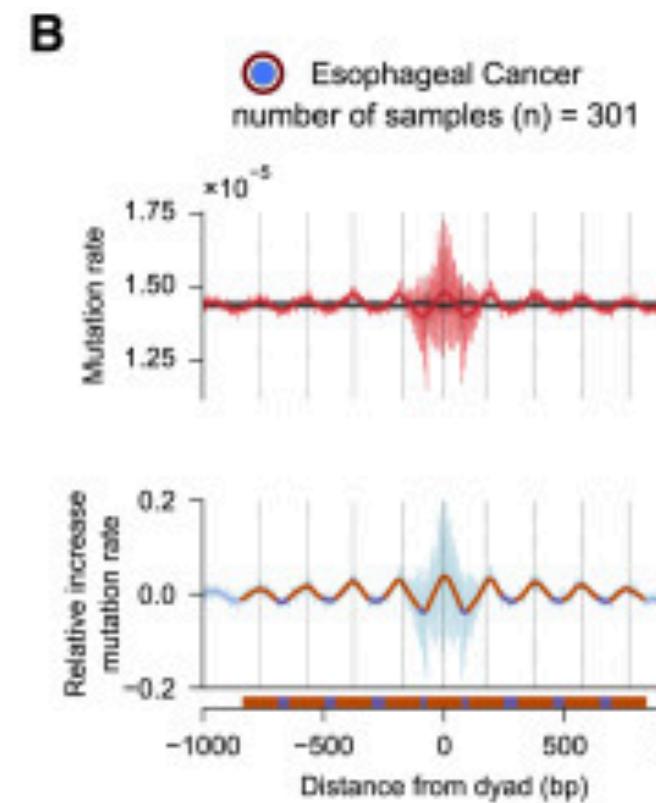
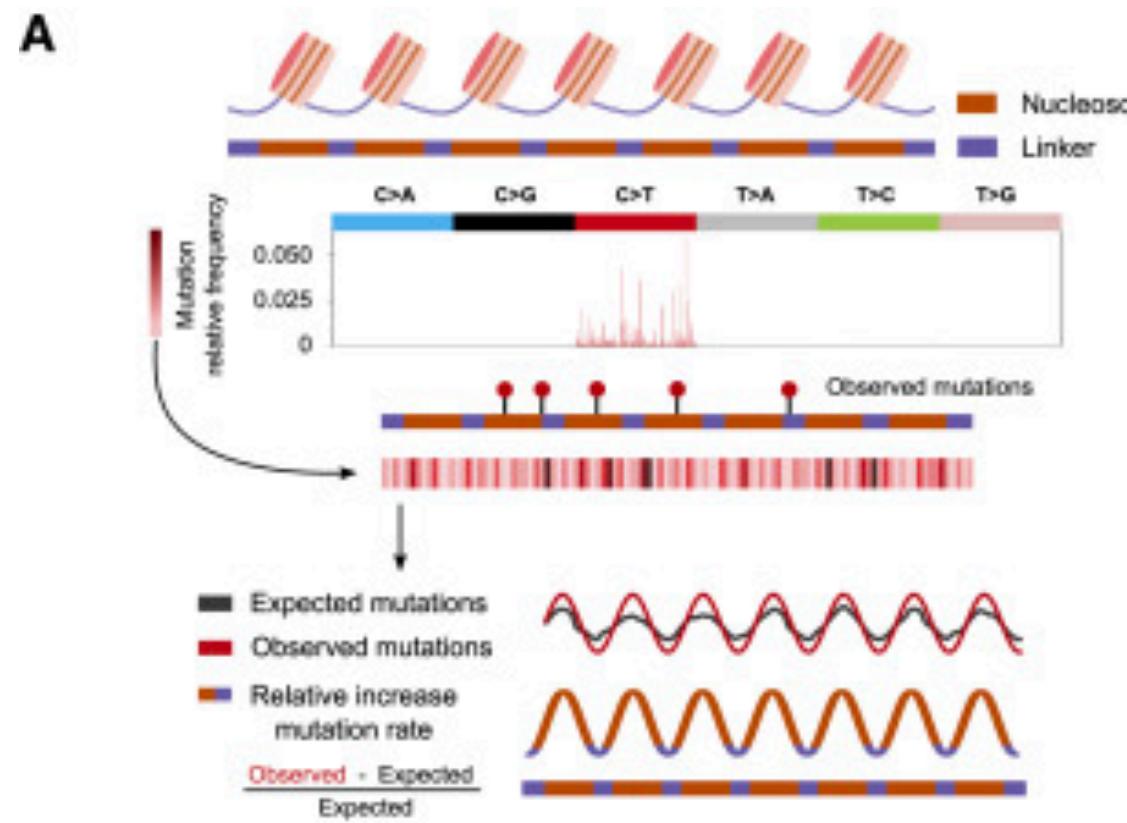
- Plot the histogram of the reading and writing scores. Do you see anything interesting?

Topic 3: Matplotlib & Seaborn

Matplotlib & Seaborn

matplotlib

seaborn



- Both are Python libraries to produce figures
- To explore the data —> visualizations of your results are needed
- Seaborn is built on top of matplotlib. It provides an easy-to-use interface for drawing attractive and informative graphics (Also from pandas dataframes)

Matplotlib & Seaborn

Mark Tenenholz 
@marktenenholz

Hello, I've been using Matplotlib for 7+ years and I still google how to do everything except plt.plot()

[Traducir Tweet](#)

2:00 p. m. · 10 ene. 2023 · 842,2 mil Reproducciones

569 Retweets 140 Tweets citados 8.607 Me gusta

Twittea tu respuesta [Responder](#)

Suhail  @Suhail · 10 ene.
En respuesta a [@marktenenholz](#)
I just copy and paste from stackoverflow. I have no shame. That library is whack.

por autor(a)

16,7 mil 8 5 221

Matplotlib building blocks

Load Libraries

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

Fully customizable

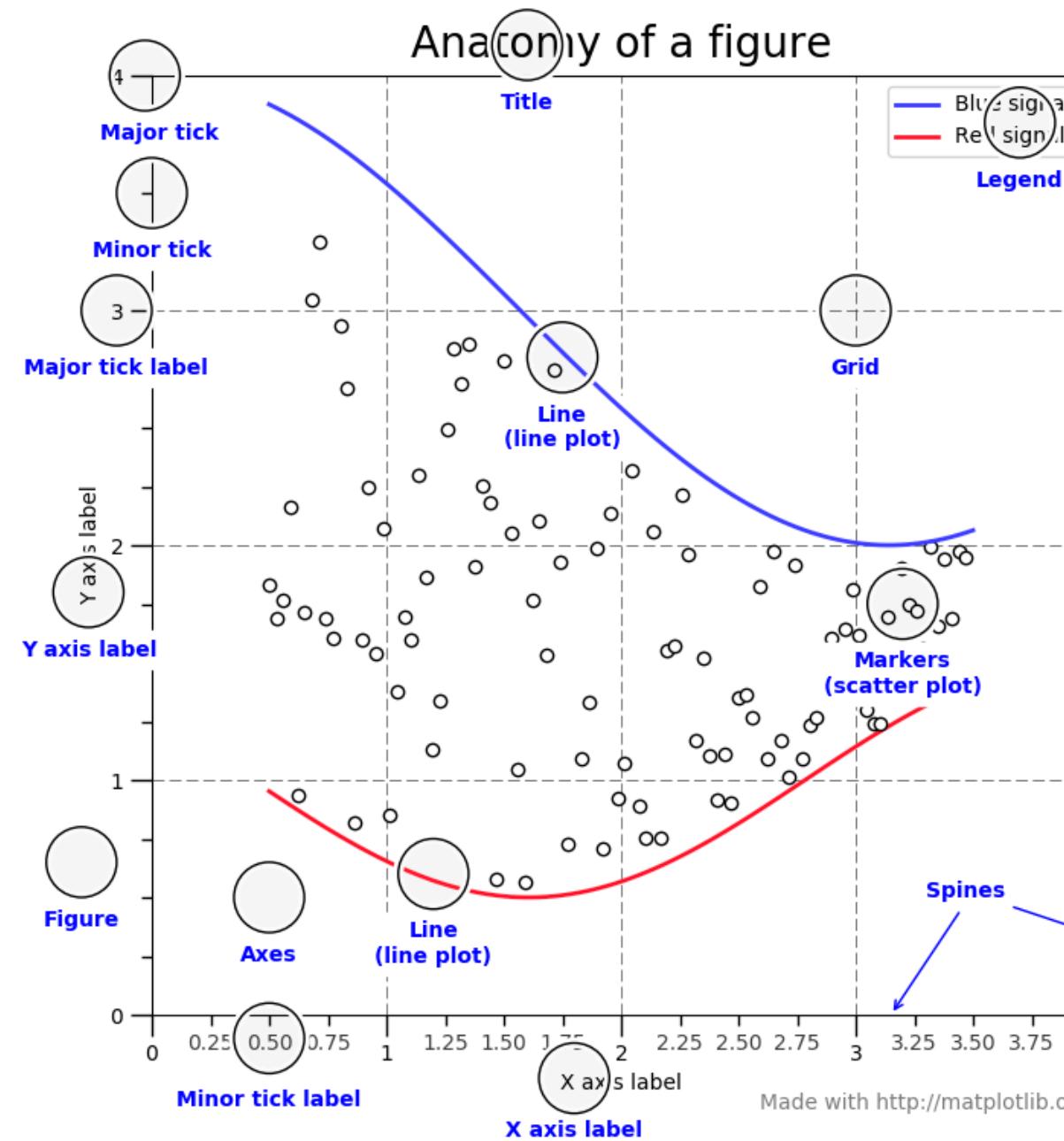


Figure structure

```
#Start Figure  
fig, ax = plt.subplots(figsize=(5, 5))  
  
#Body of the figure to build and the data to use  
sns.barplot(x=to_plot['gender'], y=to_plot['index'],  
             palette=['#08519c', '#f03b20'])  
  
#Change Axes  
ax.set_xlabel("Gender", fontsize=12)  
ax.set_ylabel("", fontsize=12)  
ax.set_yticklabels(['Male', 'Female'])  
ax.spines['top'].set_visible(False)  
ax.spines['right'].set_visible(False)  
  
# Add Numbers to plot  
for index, row in to_plot.iterrows():  
    ax.text(row.gender + 25, index, row.gender,  
            color='black', ha="center", fontsize=12)  
  
## Add legend  
custom_lines = []  
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:  
    custom_lines.append(  
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',  
                 mew=0, color=el[1], label=el[0])[0]  
    )  
ax.legend(  
    bbox_to_anchor=(0., 1.05, 1., .102),  
    handles=custom_lines, loc='upper center',  
    facecolor='white', ncol=1, fontsize=10, frameon=False  
)  
#Save or show  
plt.show()
```

Code for figure

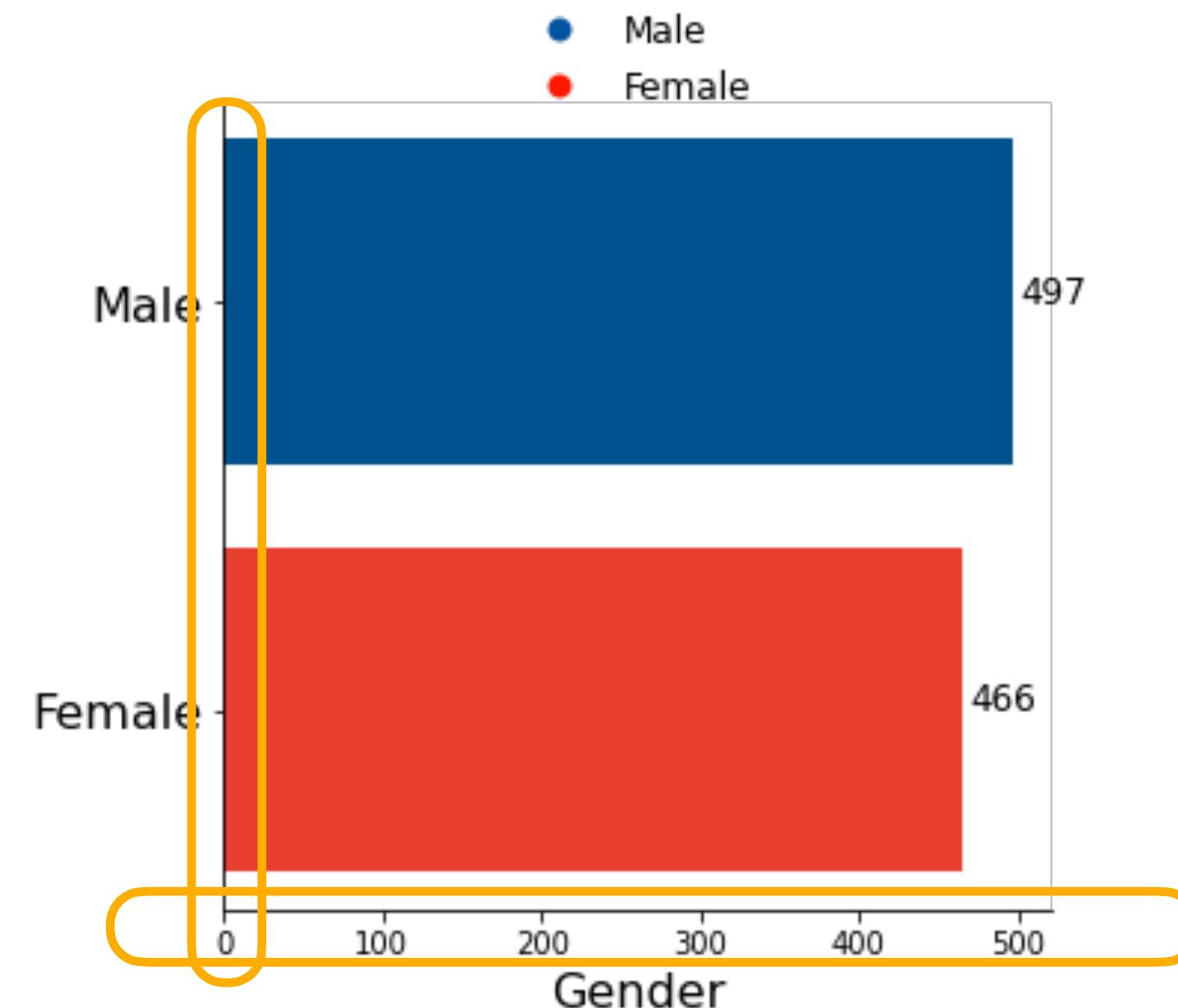
```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
             palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
            color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
                  mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
)
#Save or show
plt.show()
```



Code for figure

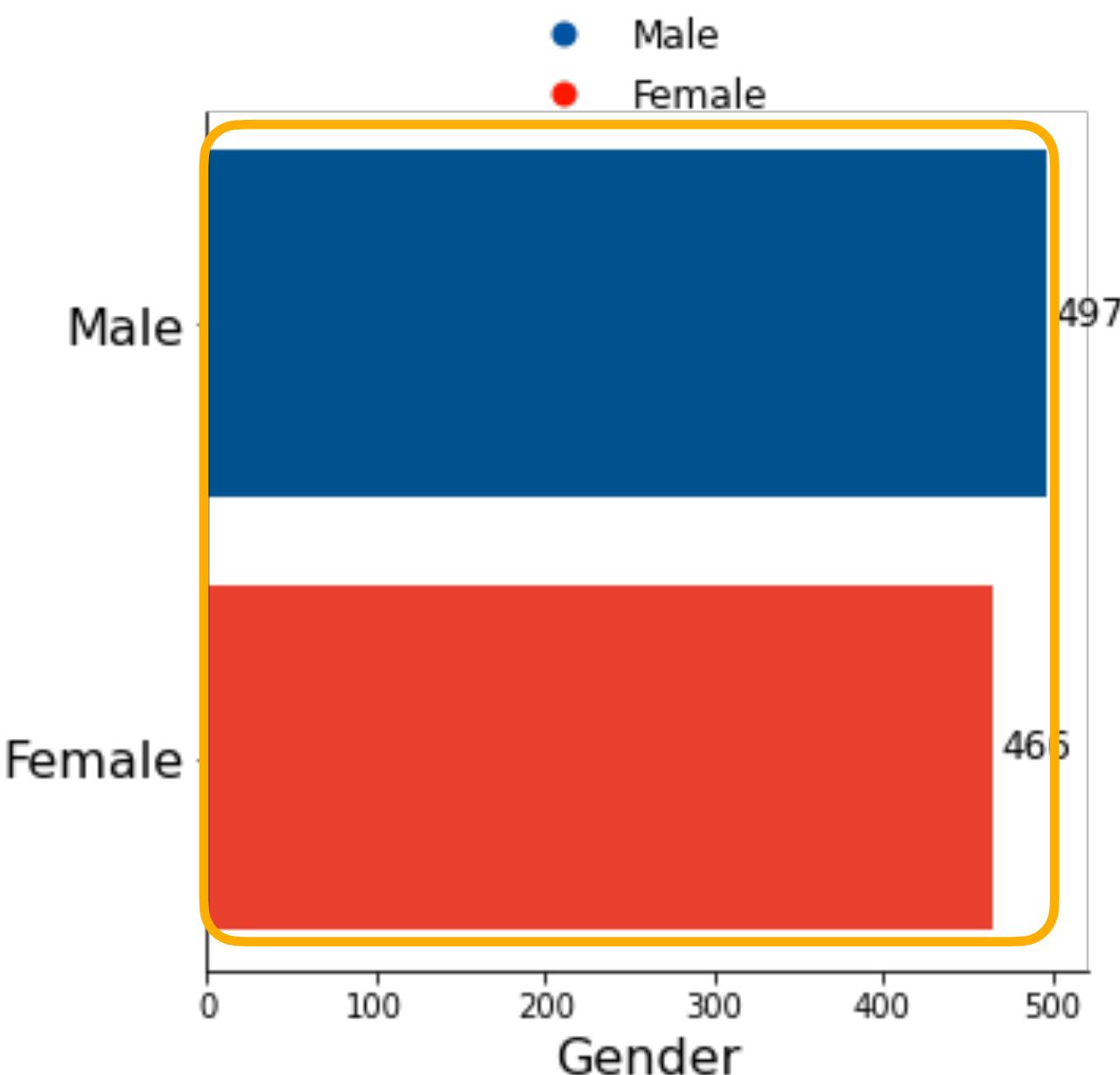
```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
             palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
            color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
                  mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
)
#Save or show
plt.show()
```



Code for figure

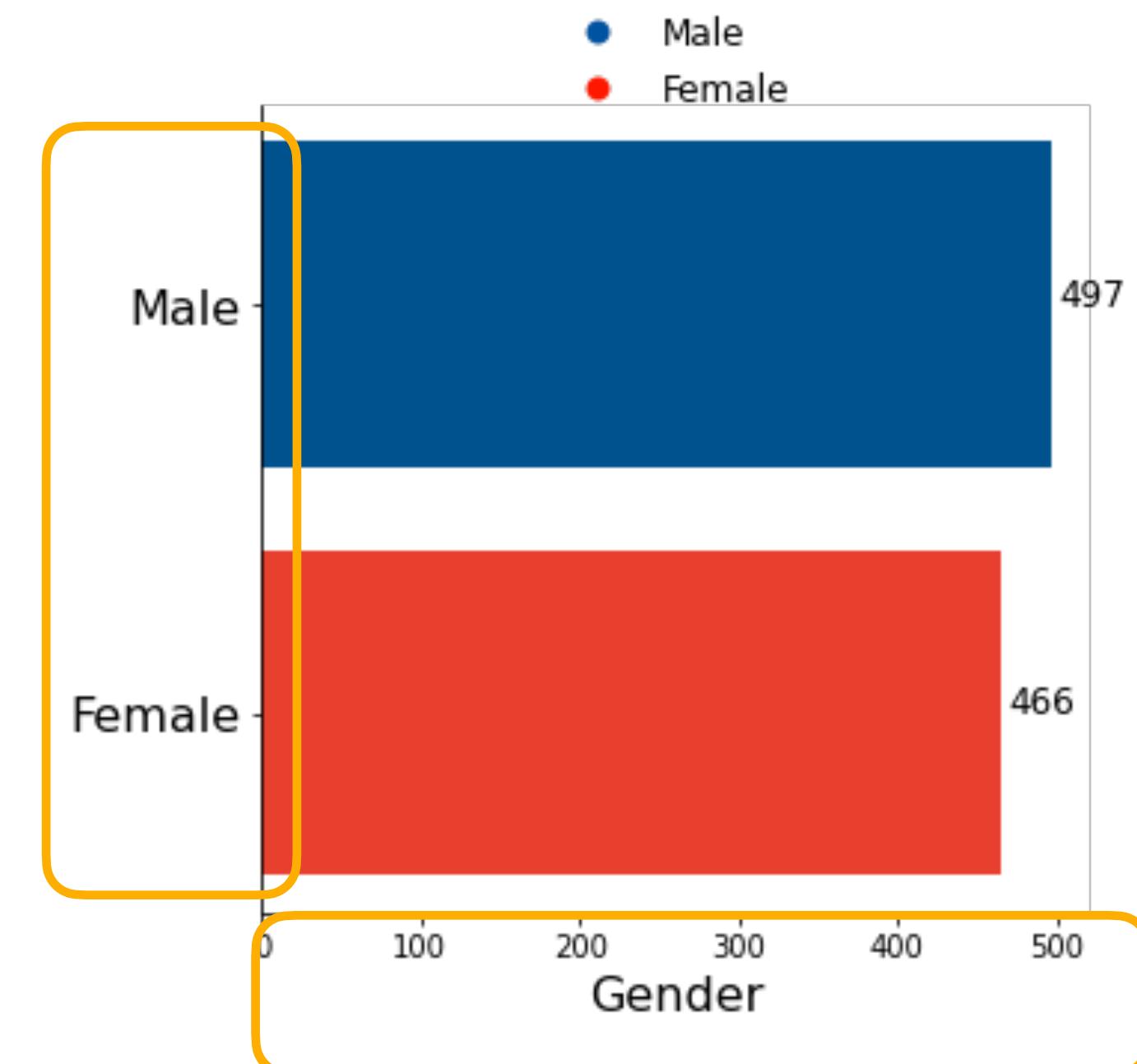
```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
             palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
            color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
                  mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
)
#Save or show
plt.show()
```



Code for figure

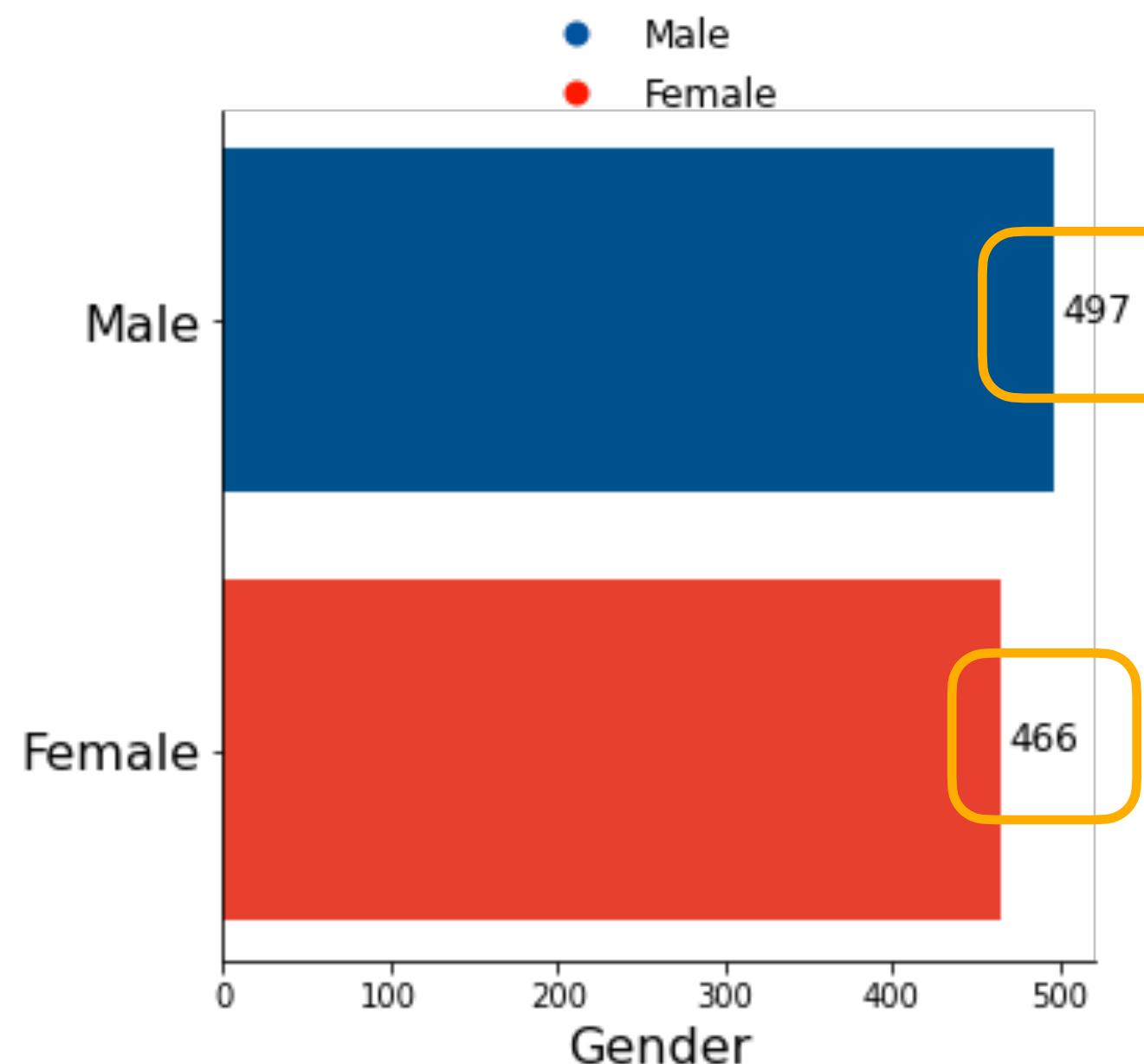
```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
             palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
            color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
                  mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
)
#Save or show
plt.show()
```



Code for figure

```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

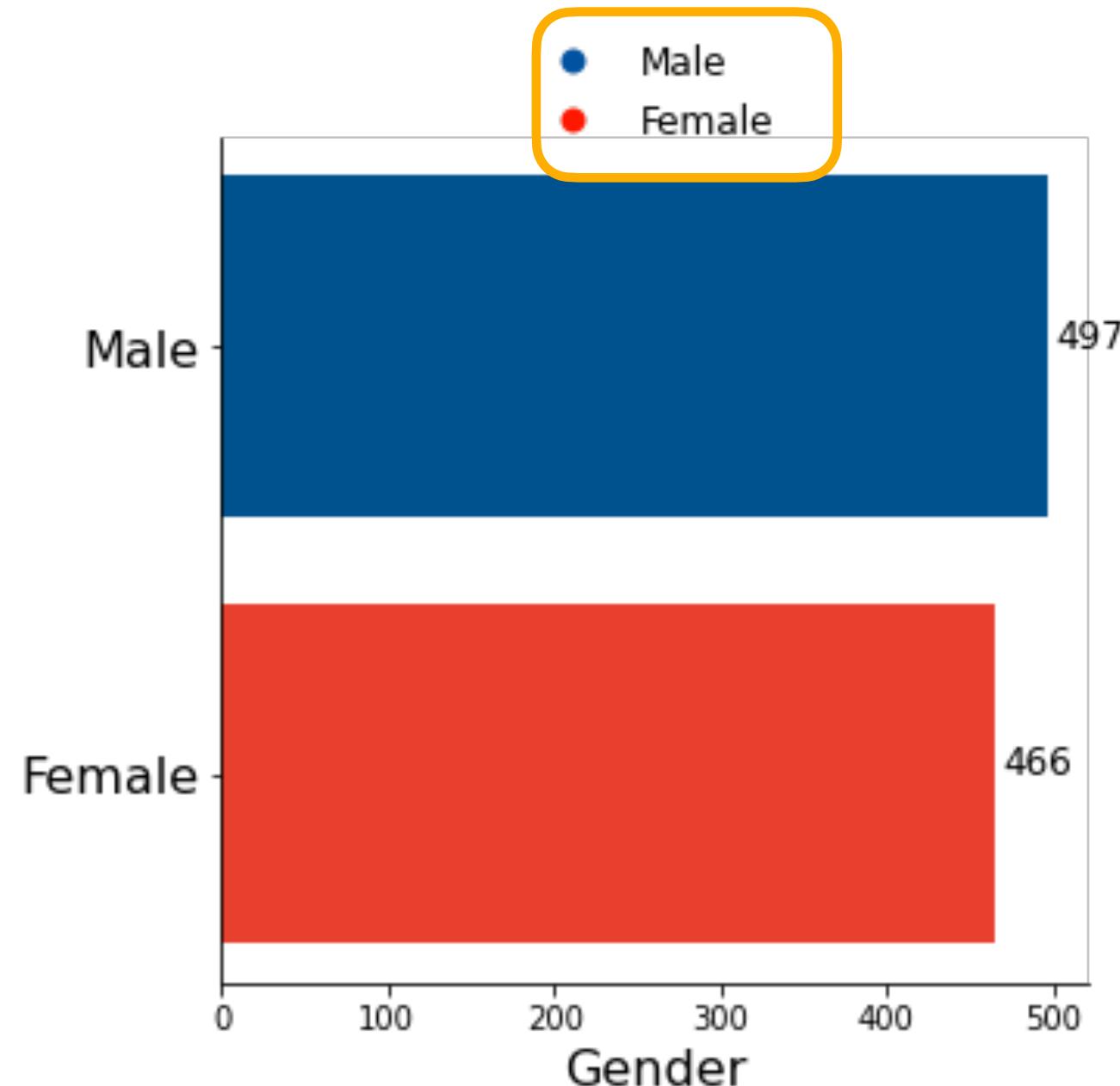
#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
             palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
            color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
                  mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
)

#Save or show
plt.show()
```



Code for figure

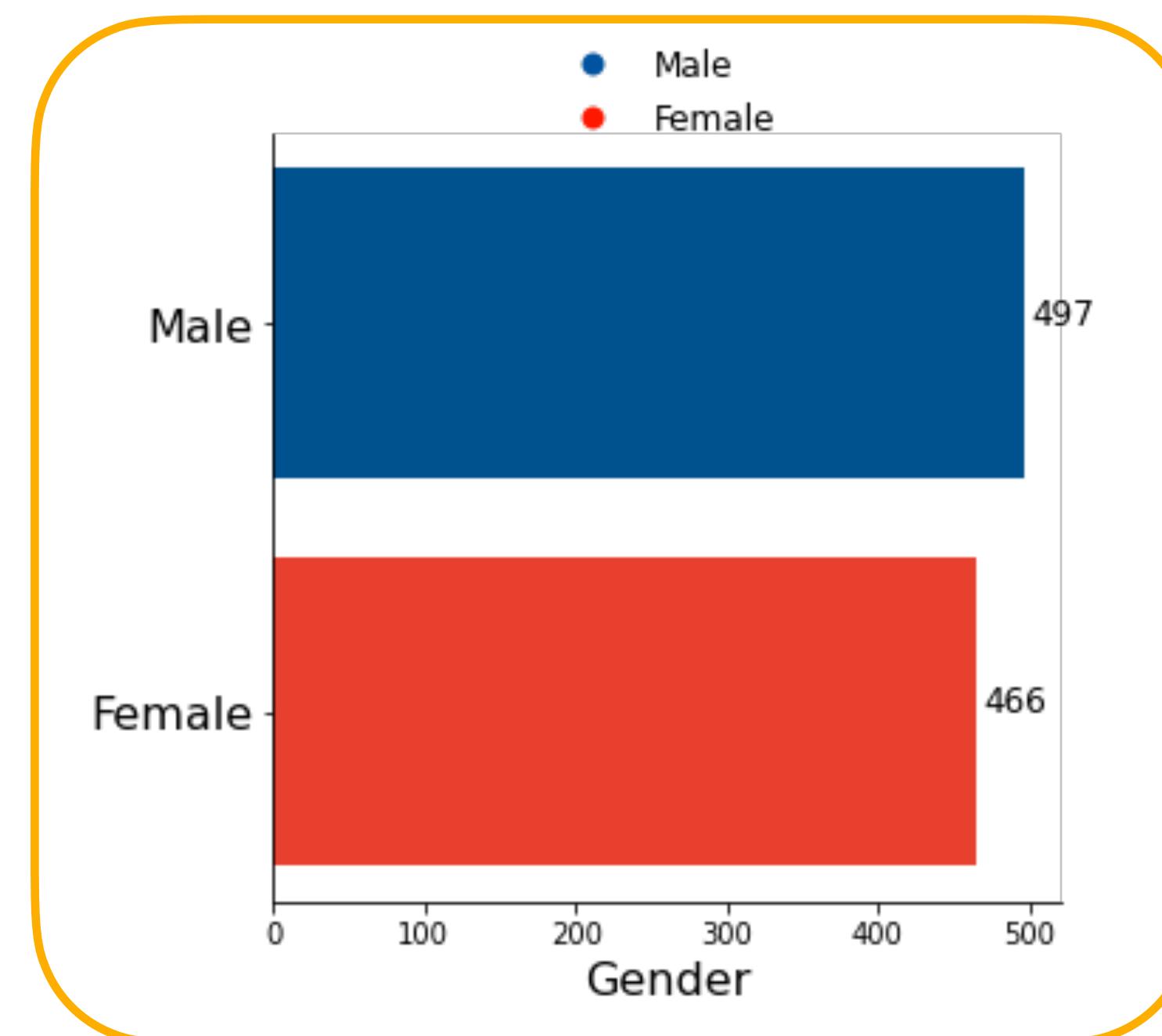
```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
             palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
            color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
                  mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
)
#Save or show
plt.show()
```

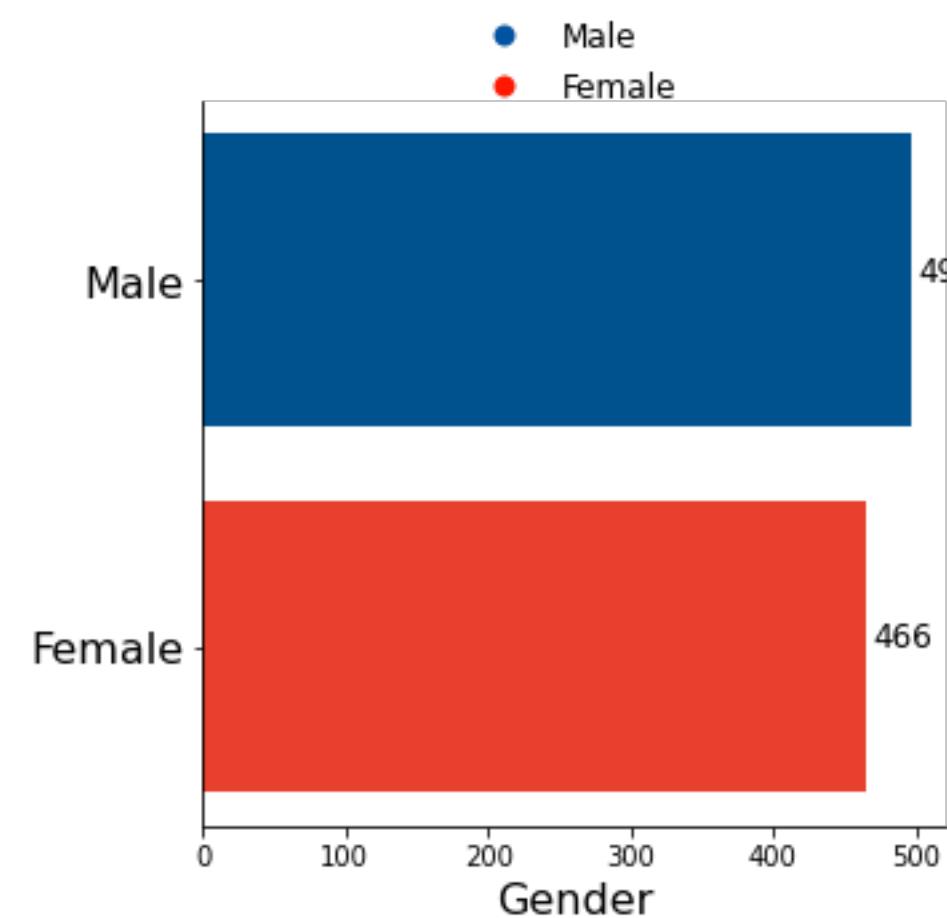


Barplot

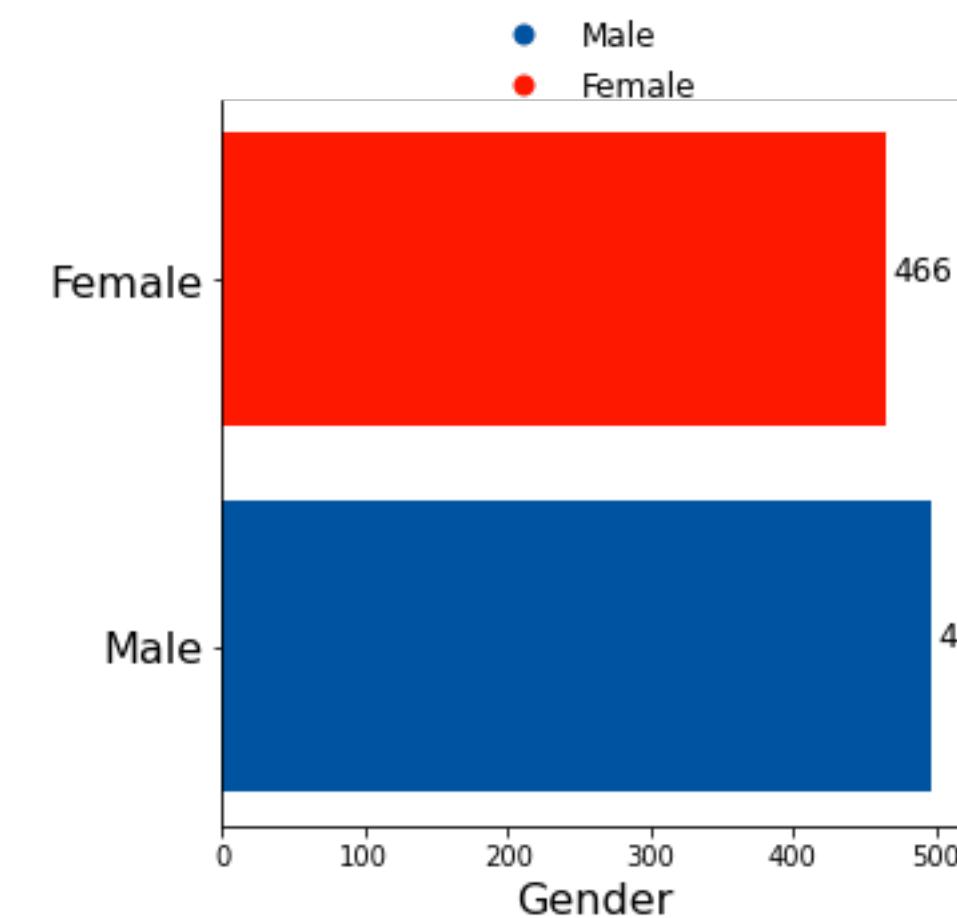
- Graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent



```
#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
             palette=['#08519c', '#f03b20'])
```



```
#Body of the figure to build and the data to use
plt.barh(width=to_plot['gender'], y=to_plot['index'],
          color=['#08519c', '#f03b20'])
```

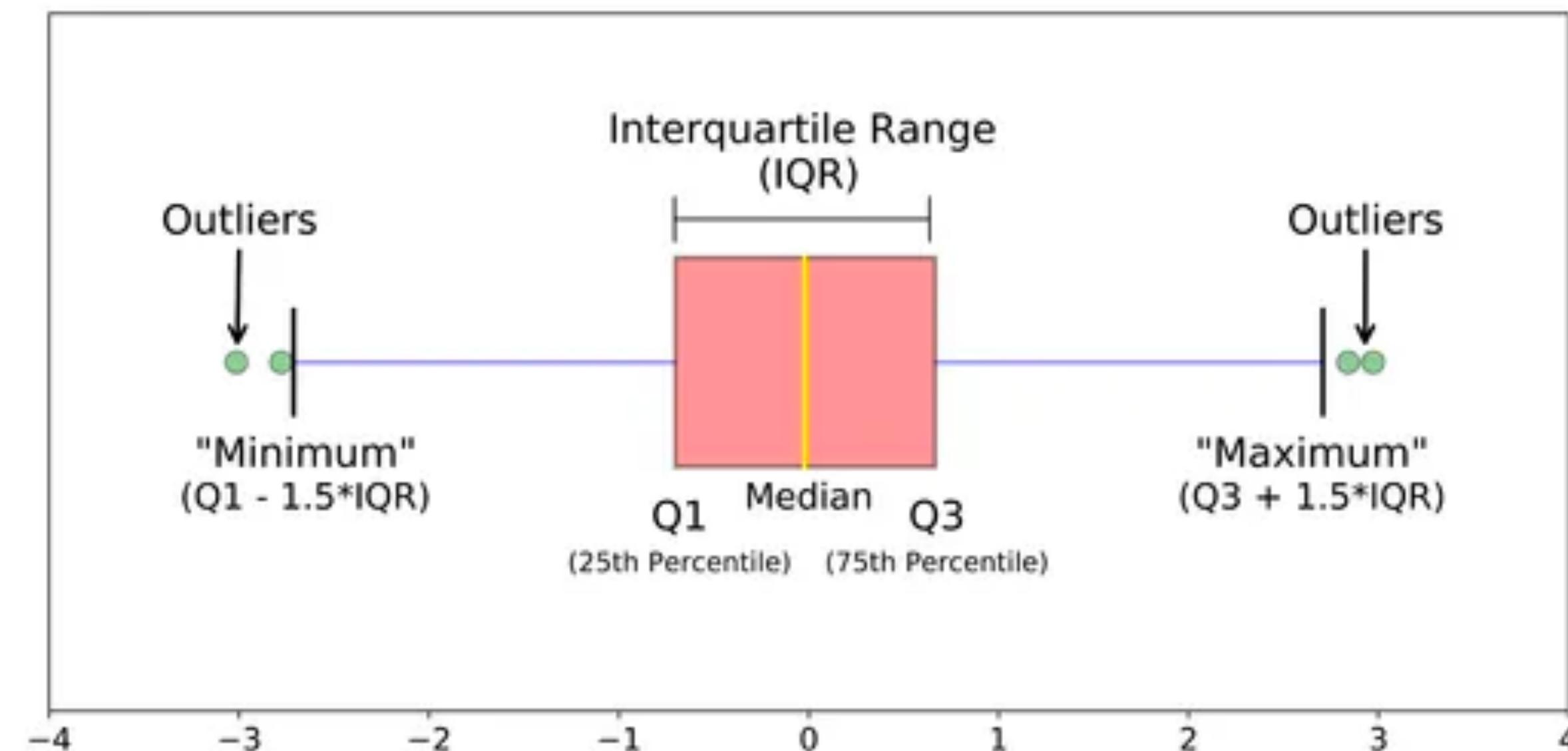


Exercise

- Write the barpot Python code for a categorical column that is not gender

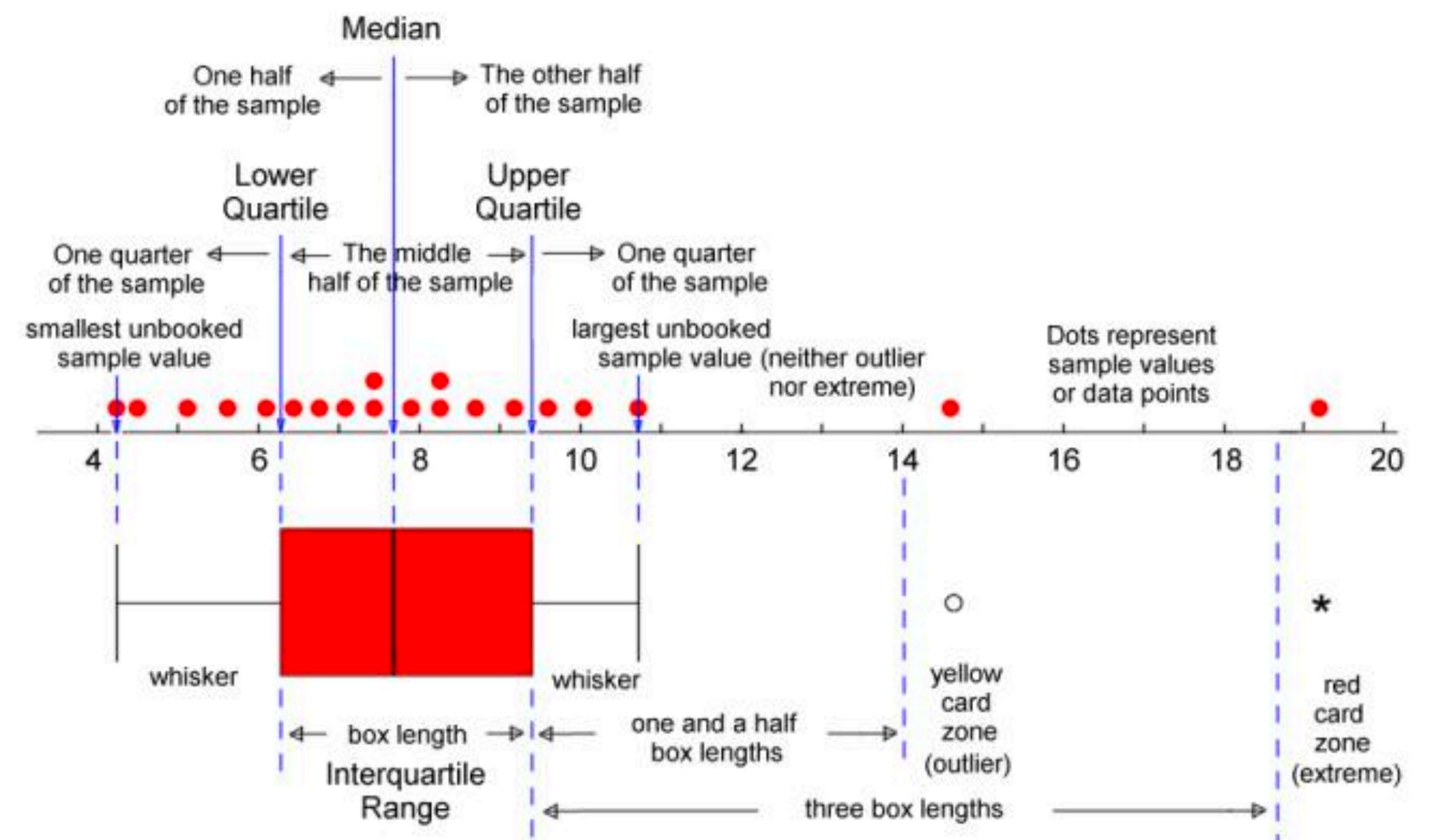
Boxplot

- A standardized way of displaying the distribution of data based on a five-number summary (“minimum”, first quartile [Q1], median, third quartile [Q3], and “maximum”)
- Graphically demonstrates the locality, spread, and skewness groups of numerical data through their quartiles



Boxplot Calculation

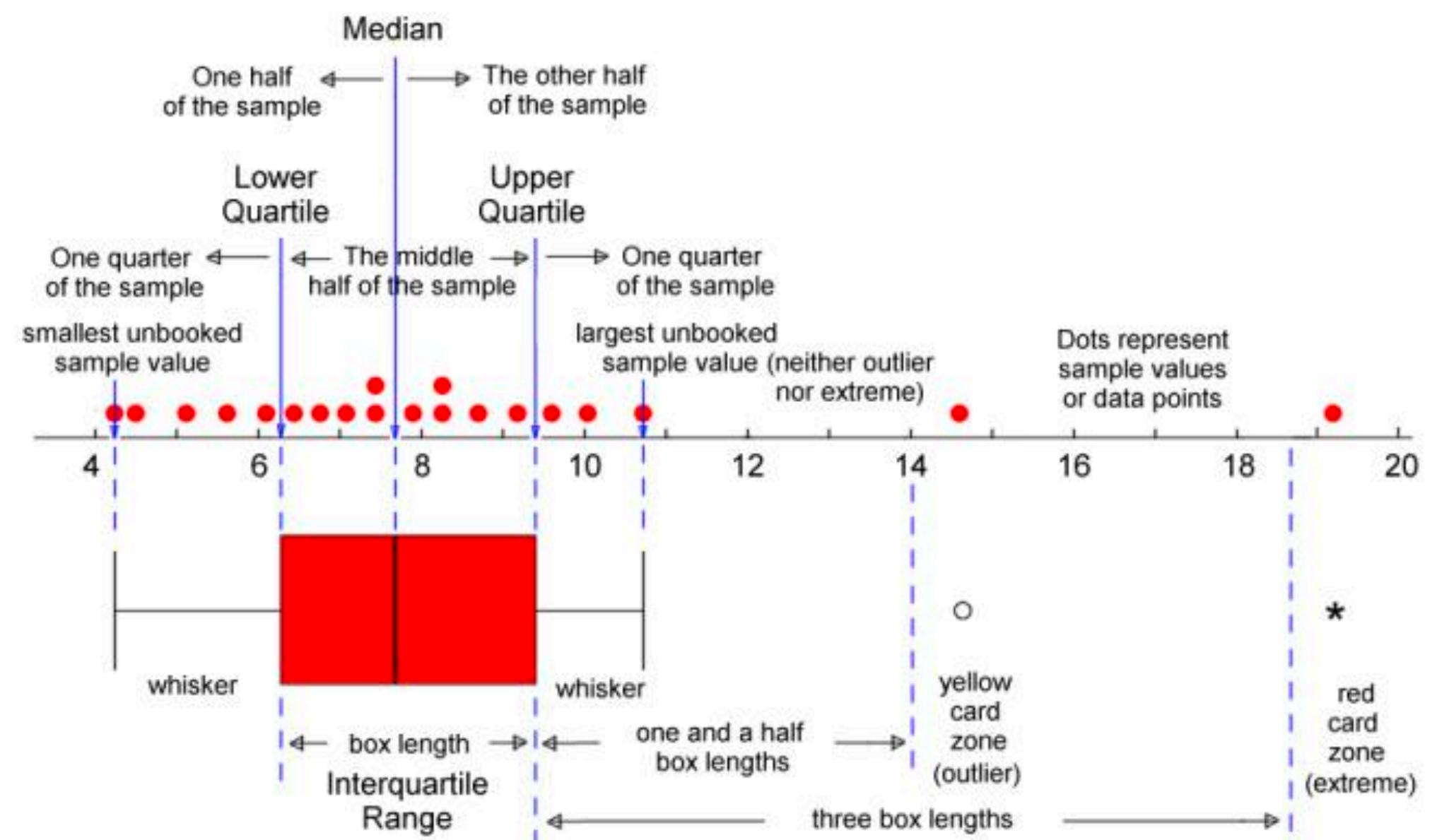
Interpretation



Boxplot Calculation

Data

Interpretation

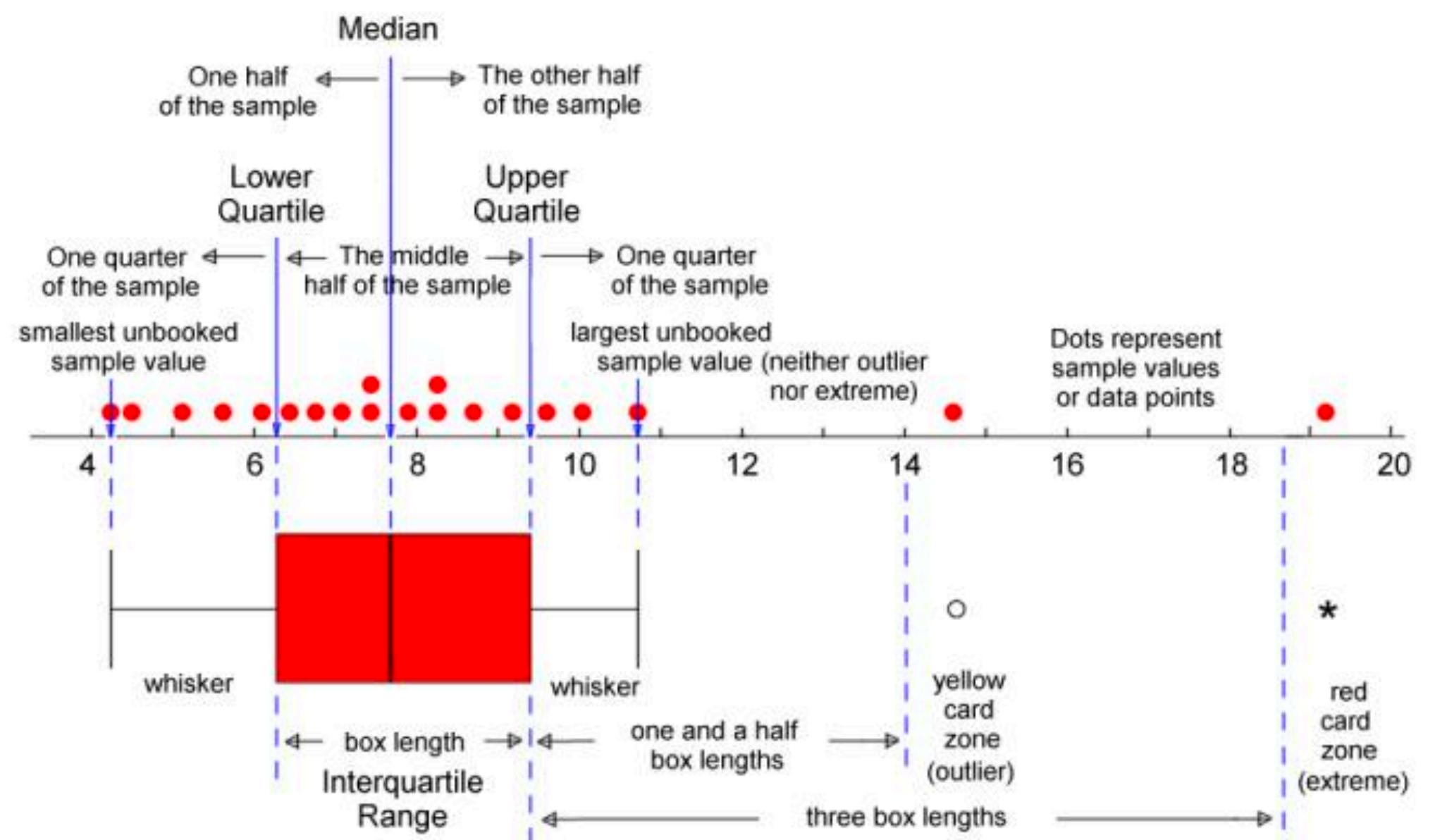


math score

| | |
|-------|------------|
| count | 963.000000 |
| mean | 66.234683 |
| std | 15.250540 |
| min | 13.000000 |
| 25% | 56.000000 |
| 50% | 66.000000 |
| 75% | 77.000000 |
| max | 100.000000 |

Boxplot Calculation

Interpretation

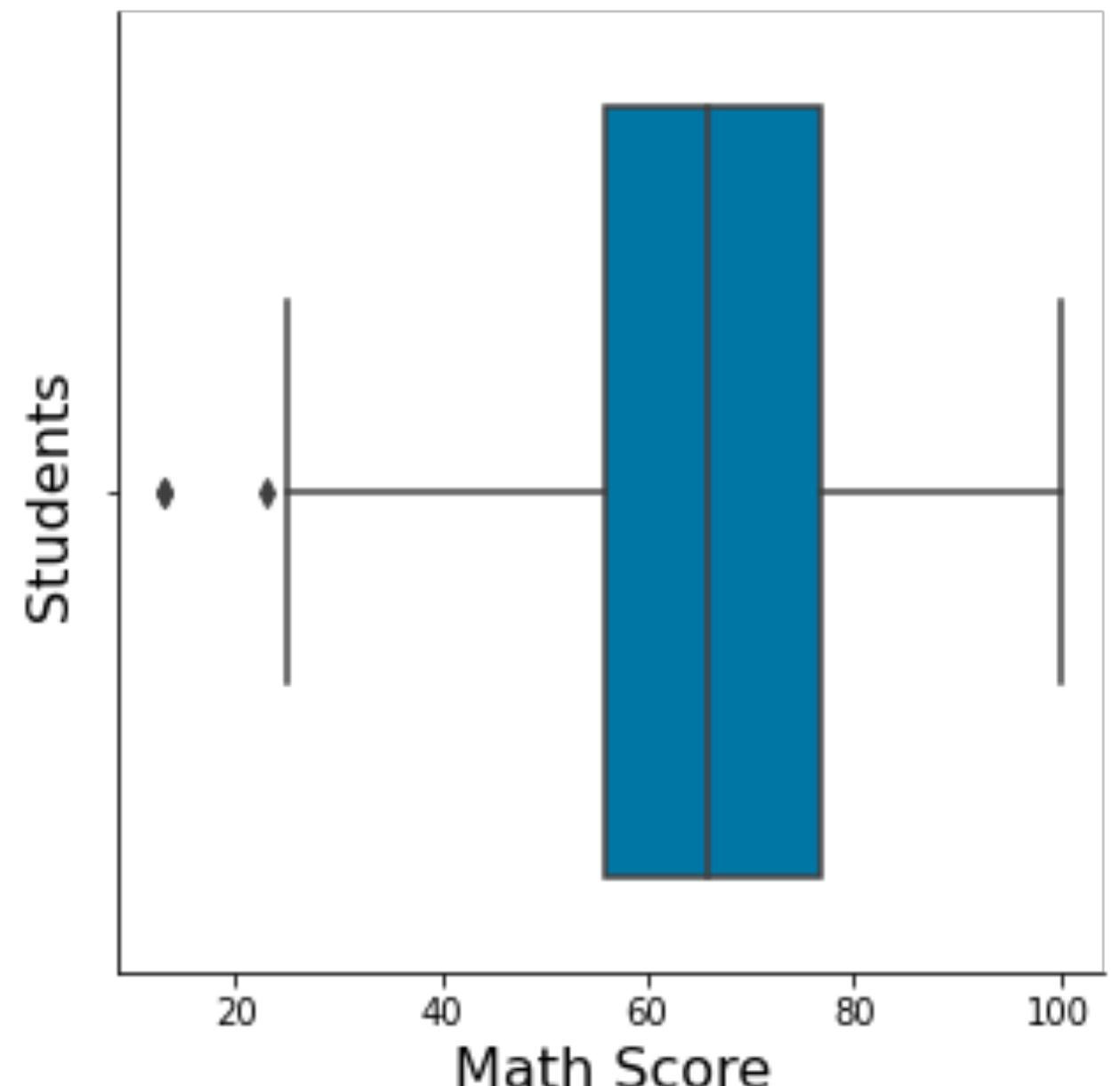


Data

math score

| | |
|-------|------------|
| count | 963.000000 |
| mean | 66.234683 |
| std | 15.250540 |
| min | 13.000000 |
| 25% | 56.000000 |
| 50% | 66.000000 |
| 75% | 77.000000 |
| max | 100.000000 |

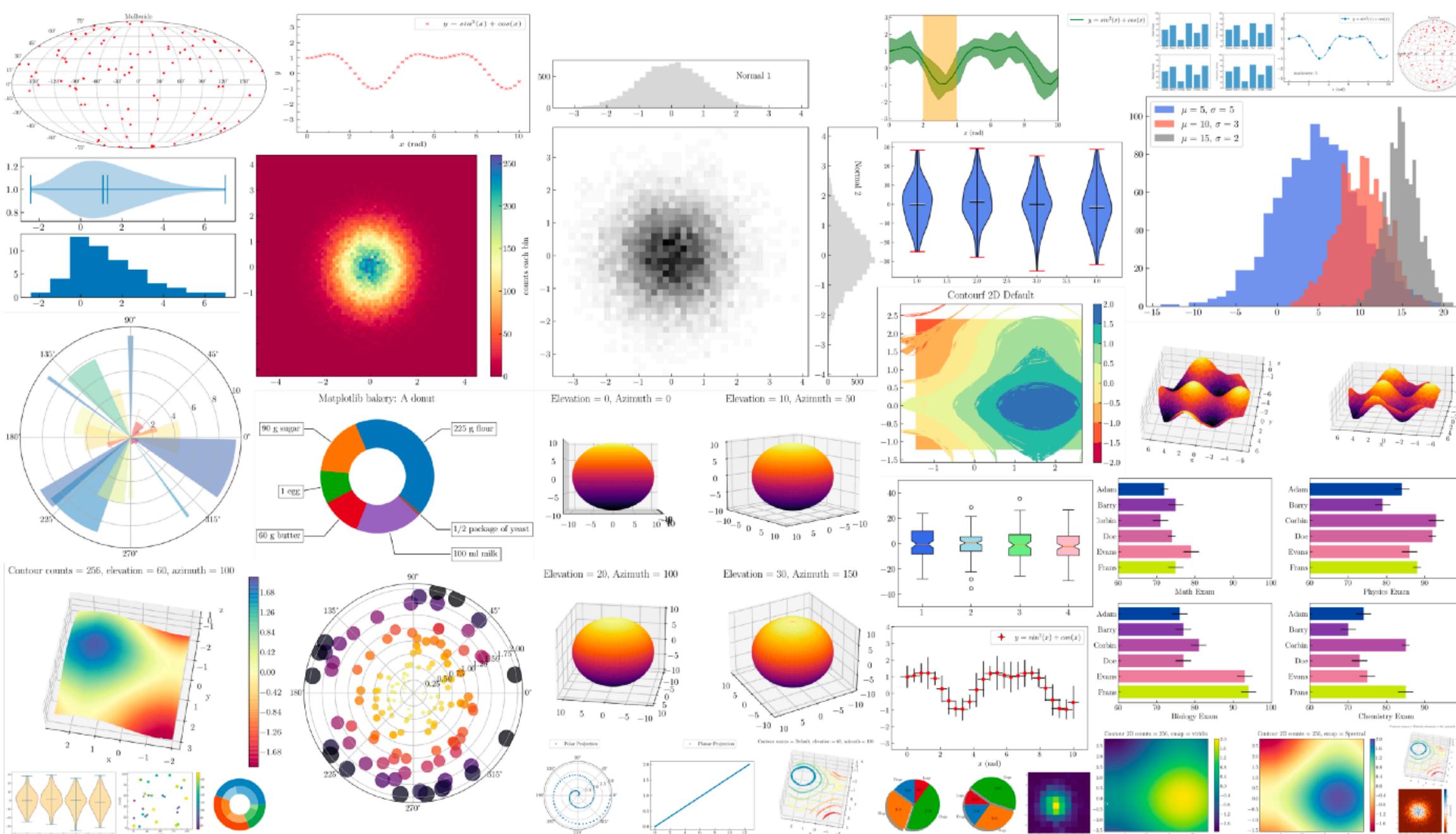
Figure Result



Exercise

- Write the boxplot Python code for the writing score column

There are infinite plot types that you can look into and use



<https://matplotlib.org/stable/gallery/index.html>

- Important to know how to use some of them quickly to explore your data:
 - Barplots and Boxplots mainly. Also, violin plots, scatter plots, etc...
- Plotting figures is an art and is crucial for communicating your results
- It is a time-consuming process. Go simple and improve when you want to show something

Topic 4: Data Exploration & Visualization

EDA steps

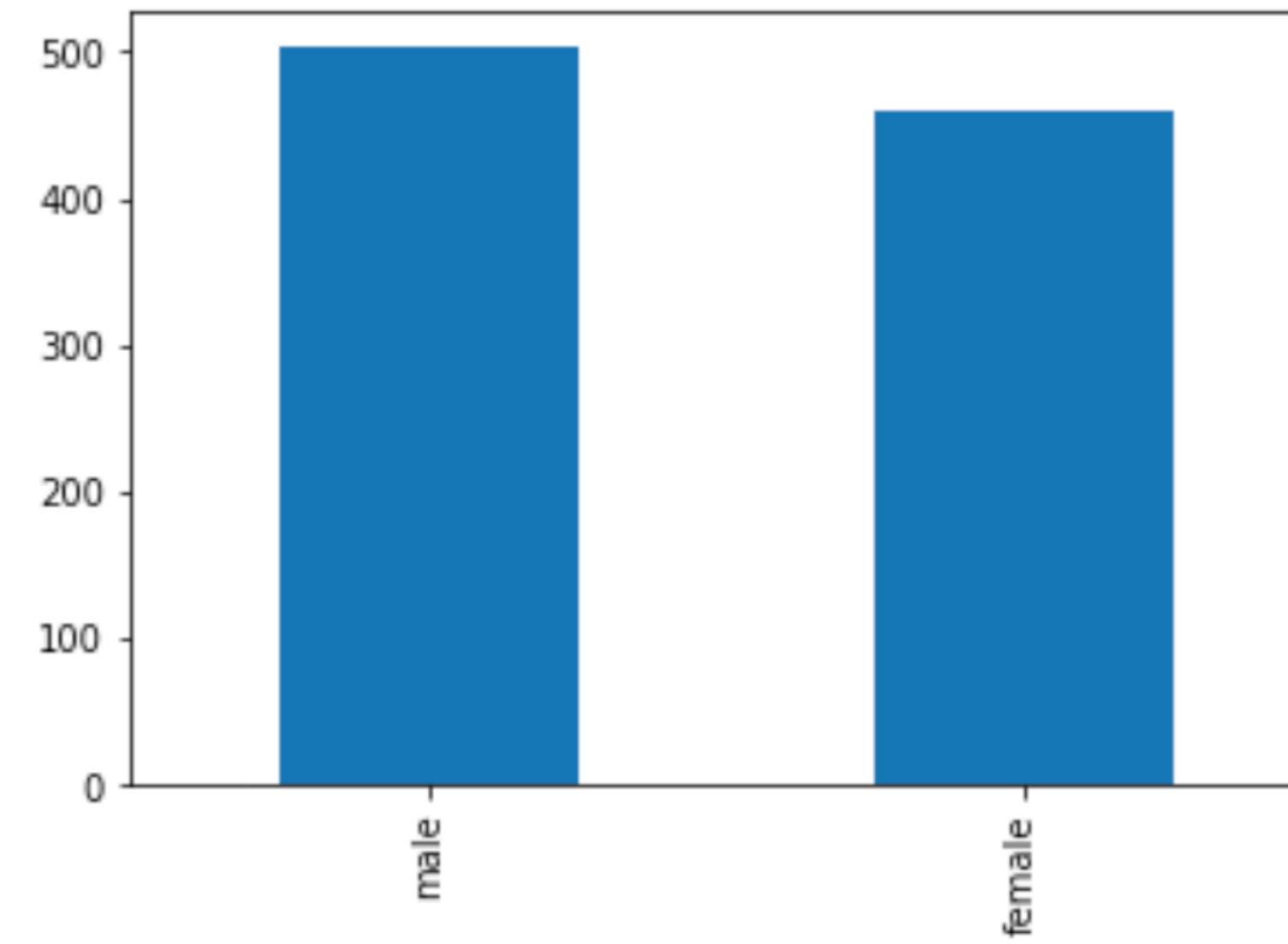
- Categorical EDA
- Numerical EDA
 - Univariate analysis
 - Bivariate analysis
(Categorical + Numerical)
 - Multivariate analysis



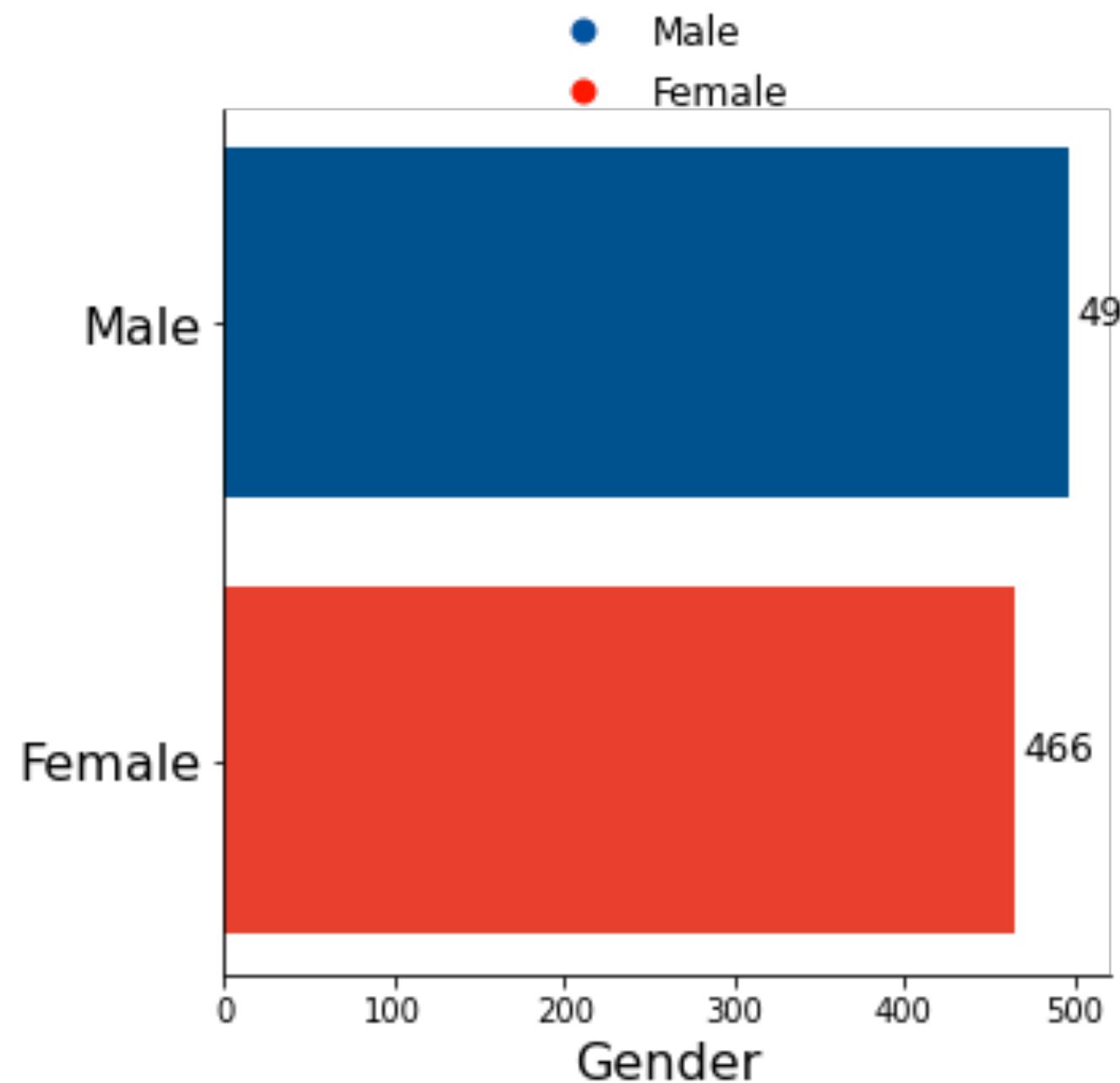
Categorical EDA

- Improve our plots: give them color, change axis names, etc...

Ugly but little code

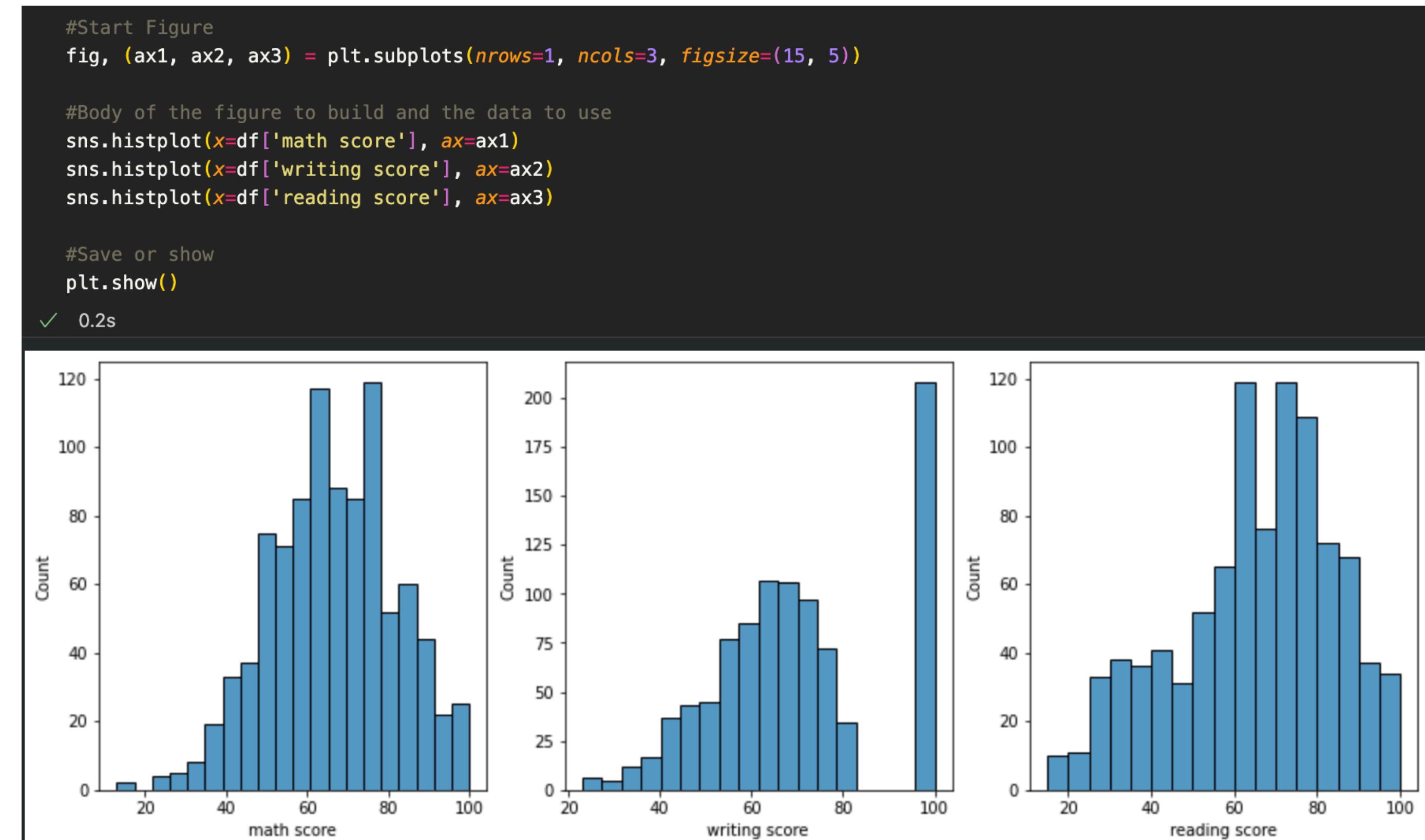


More informative but time-consuming



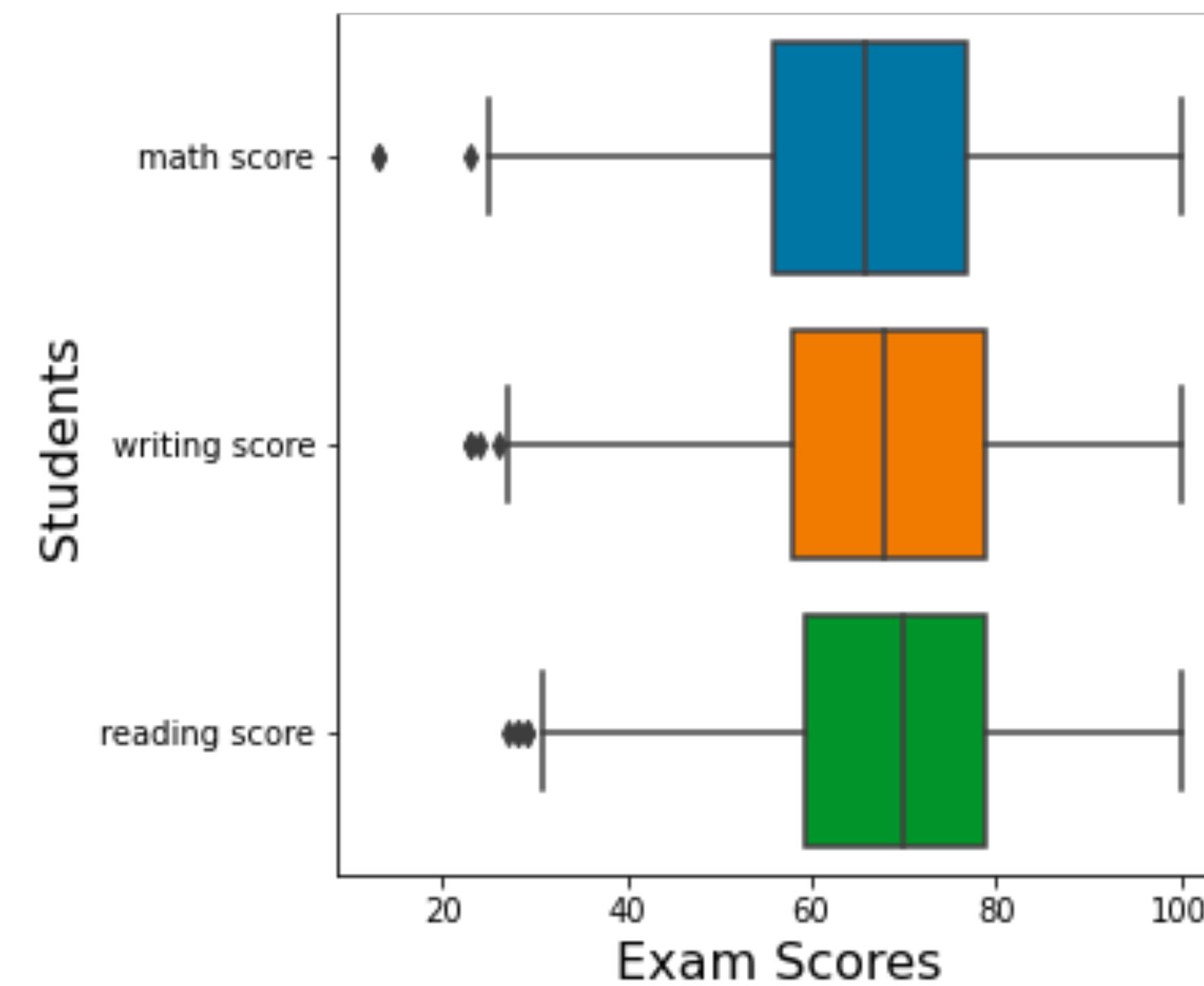
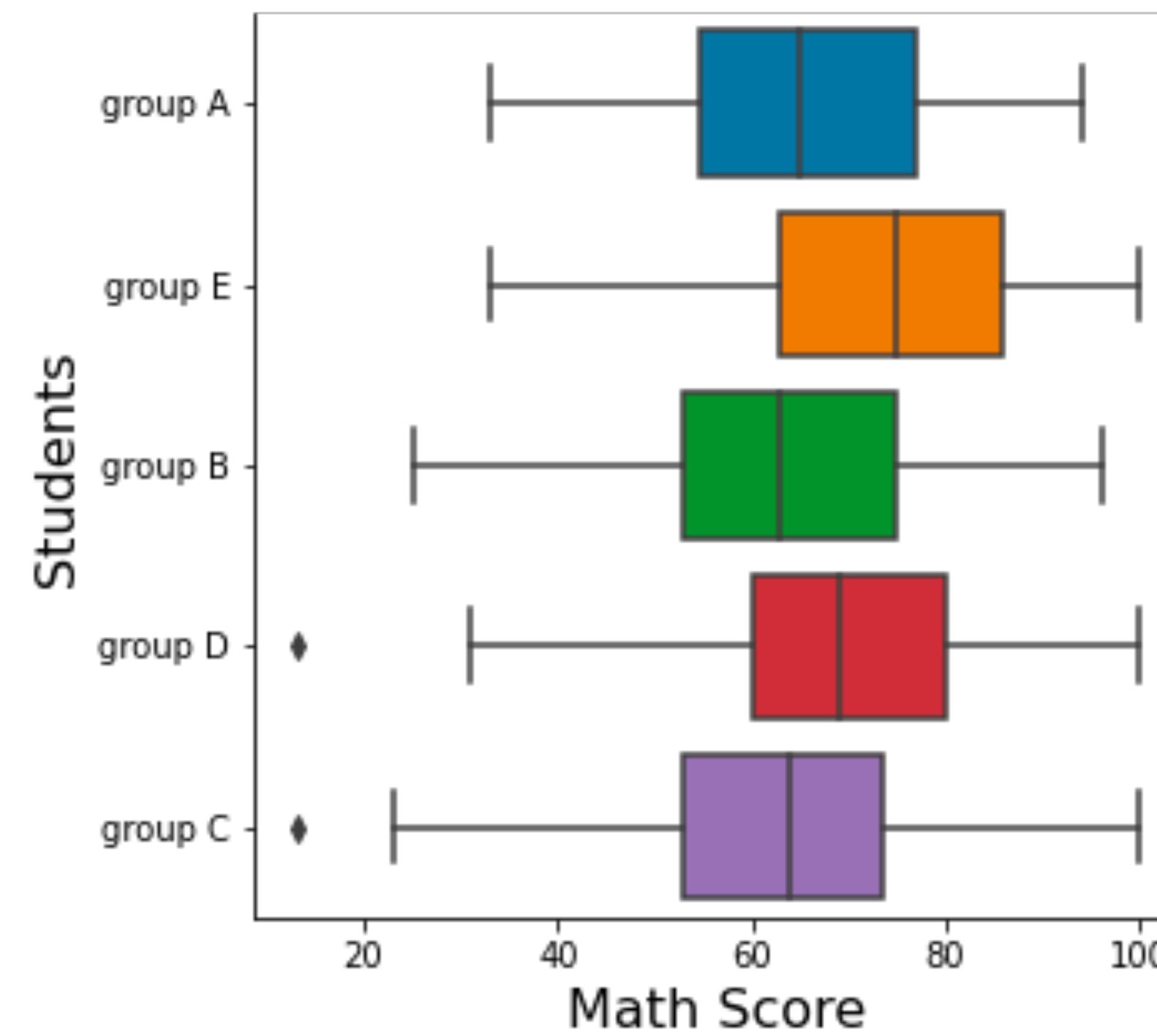
Numerical EDA - Univariate Analysis

- Study of single variables of the data by analysing the data distributions → Histograms & Boxplots
- Is my data normally distributed? Can I identify unseen outliers?



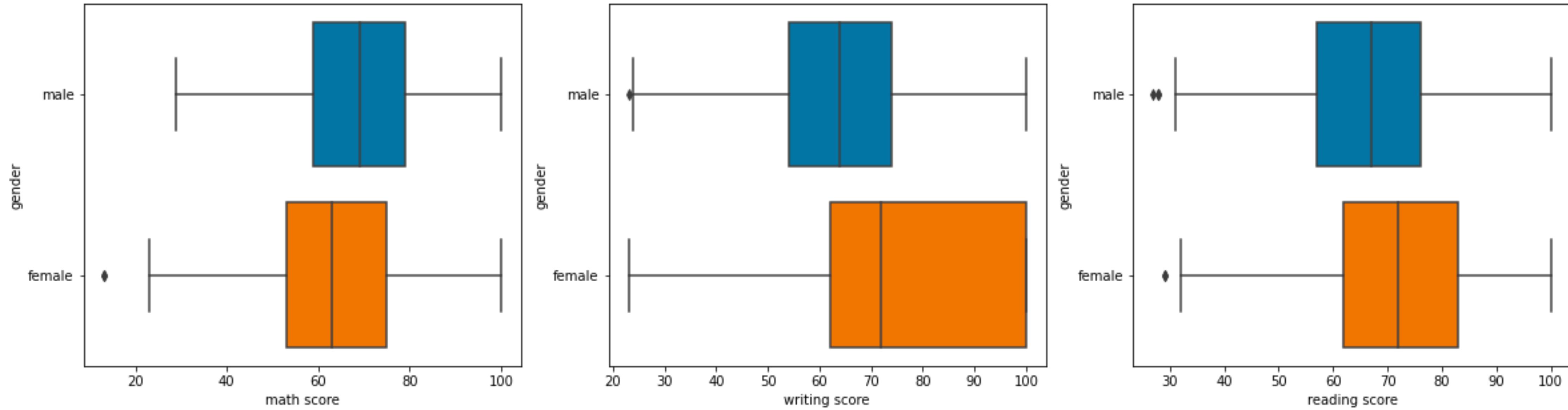
Numerical EDA - Bivariate Analysis (Categorical + Numerical)

- Combination of variables through boxplots



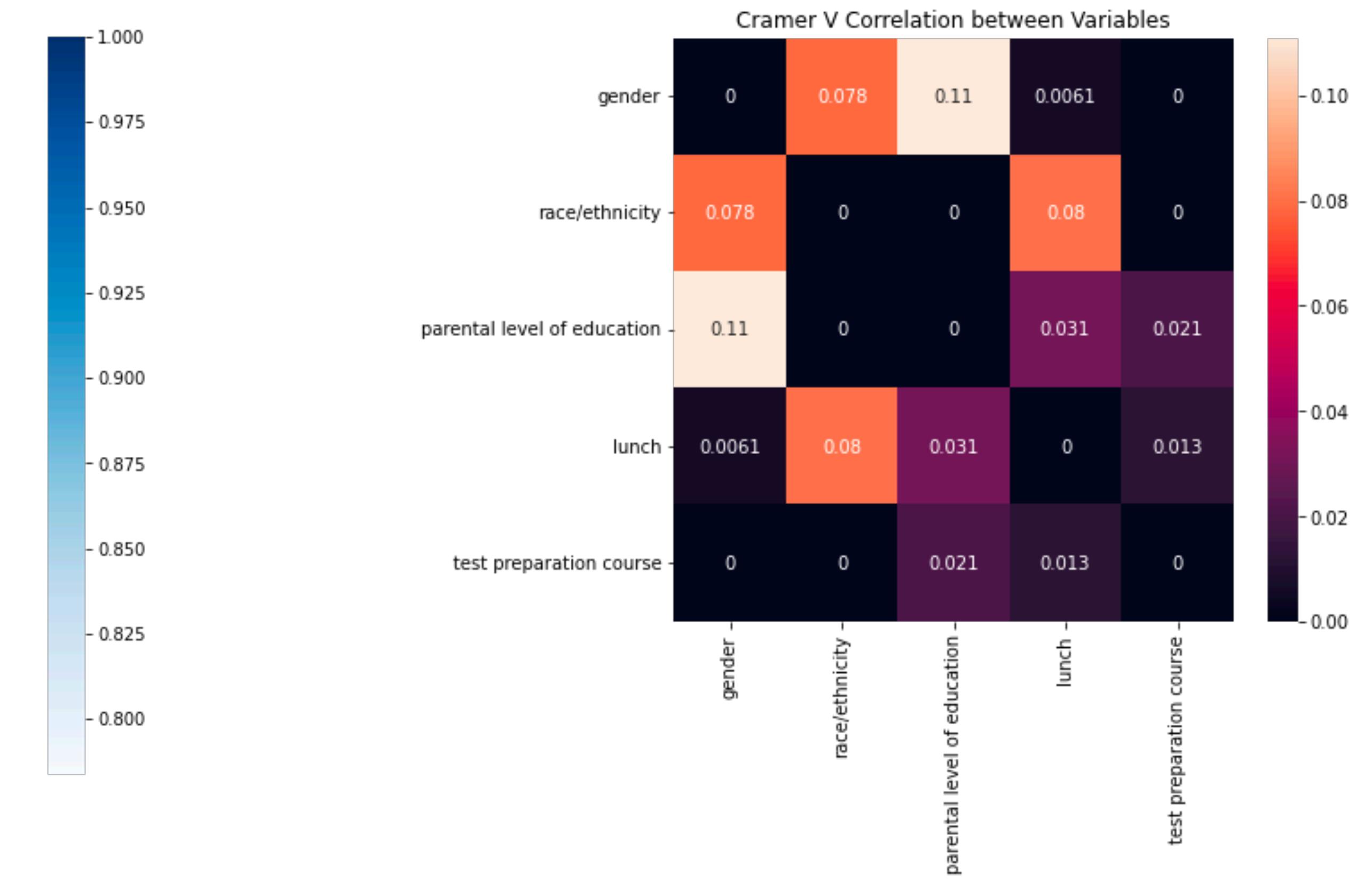
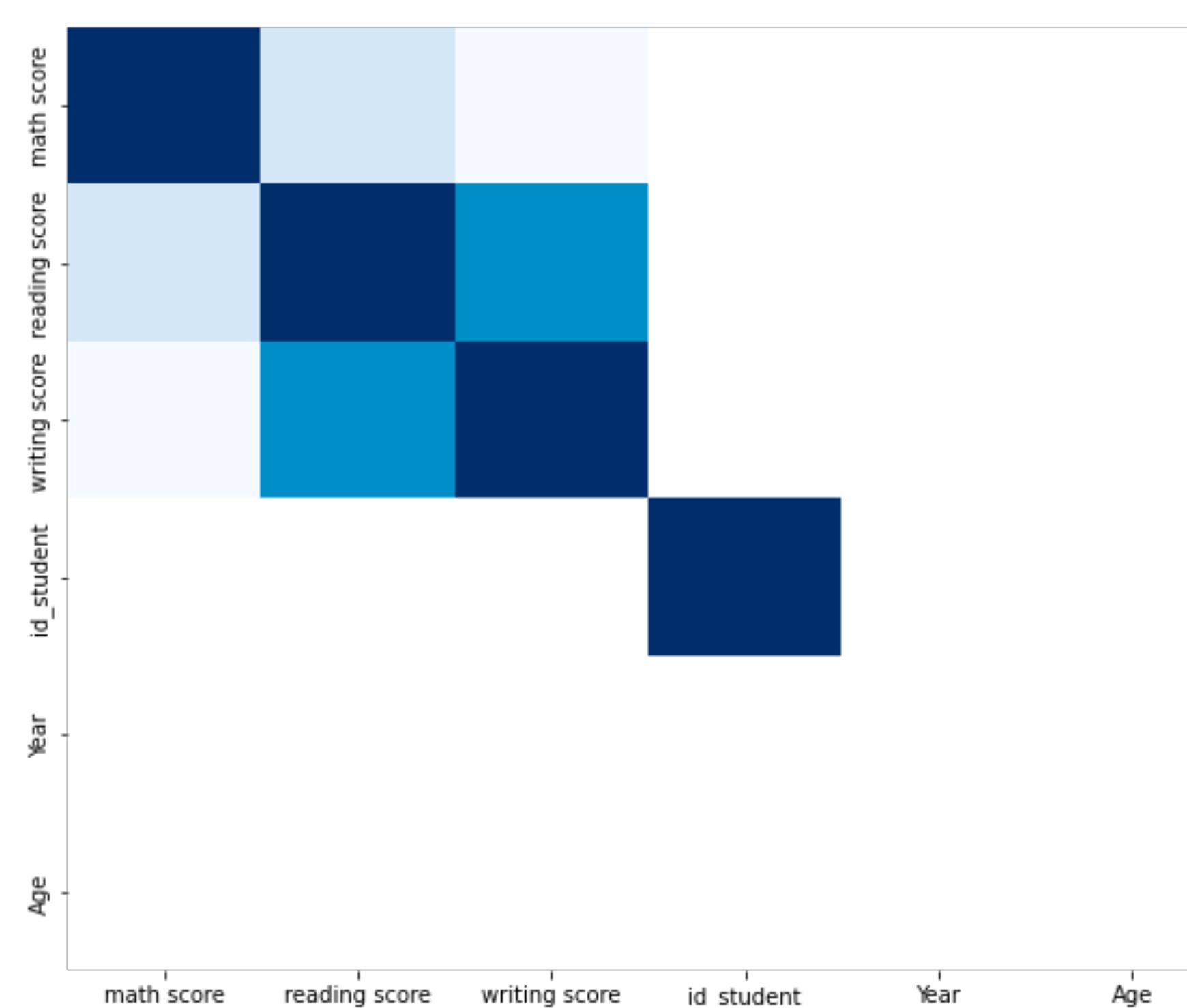
Exercise

- Can you get the plot below (They can be separated as well)?



Numerical EDA - Multivariate Analysis

- Correlation plots to see how all numerical or categorical variables are related
- This helps identify possible correlations in the data that made affect downstream models. We will come back to this



Additional Assignment: Use case with our data

- A school wants to hire us to see whether we are able to understand some problems they believe they are facing.
- Something is going wrong with the performance of the students but they are not able to figure it out and give reasons to it.
- Some parents are very disappointed with the grades of their kids and have complained continuously to the board of the school.
- Our company is low on budget and we accept the deal even if it is risky, stressful and the money won't cover much of our runaway either but we take what we get.

Can you spot which students are suffering from the problem, in which subject, and how it may be solved?

Documentation

Datacamp Courses

Python Fundamentals

Are you ready to gain the foundational skills you need to become a Python programmer? In this track, you'll learn the Python basics you need to start on your programming journey, including how to clean real-world data ready for analysis, use data visualization libraries, and even how to write your own Python functions.

Your instructor Hugo will introduce you to how companies worldwide use Python to gain a competitive edge. Through hands-on coding exercises you'll then learn how to store, manipulate, and explore data using NumPy. Then it's time to level-up as you learn how to visualize your data using Matplotlib, manipulate DataFrames and dictionaries using pandas, and write your own functions and list comprehension. Start this track to add these essential Python skills to your data science toolbox.

[Switch Track](#)

Python 15 hours 4 Courses 1 Skill Assessment

Data Scientist with Python

Gain the career-building Python skills you need to succeed as a data scientist. No prior coding experience required.

In this track, you'll learn how this versatile language allows you to import, clean, manipulate, and visualize data—all integral skills for any aspiring data professional or researcher. Through interactive exercises, you'll get hands-on with some of the most popular Python libraries, including pandas, NumPy, Matplotlib, and many more. You'll then work with real-world datasets to learn the statistical and machine learning techniques you need to train decision trees and use natural language processing (NLP). Start this track, grow your Python skills, and begin your journey to becoming a confident data scientist.

[Resume Track](#)

Python 88 hours 23 Courses 6 Projects 3 Skill Assessments

Free Books to learn Python & data analytics

