Python for Exploratory Data Analysis (Workshop)

Exploratory Data Analysis (EDA) is about getting an overall understanding of data. EDA includes exploring data to find its main characteristics, identifying patterns and visualizations. EDA provides meaningful insights into data to be used in a variety of applications e.g,. machine learning. Python can be effectively used to do EDA as it has a rich set of easy-to-use libraries like Pandas, Seaborn, Numpy and Matplotlib. In this Workshop we will cover basics of EDA using a real world data set, including, but not limited to, Correlating, Converting, Completing, Correcting, Creating and Chartingthe data. In addition we will learn how to install and use Jupyter Notebooks (an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text).

Setting up Requirements:

First step is to understand and install all requirements. It also includes acquiring data (on which EDA is going to be done) from a given github link.
Following steps would be completed on all attendant's machines.

- Make sure python is installed and working
- A brief introduction on python virtual environment
    - Virtual environment is a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.
- Create a virtual environment
- A brief introduction on jupyter notebooks
    - https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html
- Install Jupyter notebook
    - http://jupyter.org/install.html
- Get data and requirement file from https://github.com/noraiz-anwar/exploratory-data-analysis
- Install all requirements using pip from given requirement file
- Check all requirements are satisfied

A brief introduction of installed libraries:

We will be using installed libraries to perform different operations on data. Let's explore these libraries a bit.

- Numpy
    - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
    - http://www.numpy.org/
- Pandas
    - pandas is a python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.
    - https://pandas.pydata.org/
- Seaborn
    - Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
    - https://seaborn.pydata.org/
- Matplotlib
    - Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
    - https://matplotlib.org/

Introduction of data:

We will be using data of olympic games here. This data holds 120 years of olympic history including bio of athletes and information about the game they participated in.

The file athlete_events.csv contains 271116 rows and 15 columns; Each row corresponds to an individual athlete competing in an individual Olympic event (athlete-events). Columns are the following:

1. ID - Unique number for each athlete;
2. Name - Athlete's name;
3. Sex - M or F;
4. Age - Integer;
5. Height - In centimeters;
6. Weight - In kilograms;
7. Team - Team name;
8. NOC - National Olympic Committee 3-letter code;
9. Games - Year and season;
10. Year - Integer;
11. Season - Summer or Winter;
12. City - Host city;
13. Sport - Sport;
14. Event - Event;
15. Medal - Gold, Silver, Bronze, or NA.

The file noc_regions.csv contains 230 rows and 3 columns. Each row contains a NOC and its related region and any notes. Columns are following:

1. NOC - National Olympic Committee 3-letter code;
2. Region - Name of country
3. Notes - String containing any useful information about region and NOC

Importing Data into Data Frames:

To start working on data first we need to import data from csv files to pandas DataFrame. This will be done using pandas' read_csv method. We will further learn how different delimiters are used by this function.

Collecting basic information about data:

We need to make sense of our data about how does it look like. We will explore some more pandas' function here like

- See data in tabular form using head.

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN

- Descriptive statistics using pandas' describe

	ID	Age	Height	Weight	Year
count	271116.000000	261642.000000	210945.000000	208241.000000	271116.000000
mean	68248.954396	25.556898	175.338970	70.702393	1978.378480
std	39022.286345	6.393561	10.518462	14.348020	29.877632
min	1.000000	10.000000	127.000000	25.000000	1896.000000
25%	34643.000000	21.000000	168.000000	60.000000	1960.000000
50%	68205.000000	24.000000	175.000000	70.000000	1988.000000
75%	102097.250000	28.000000	183.000000	79.000000	2002.000000
max	135571.000000	97.000000	226.000000	214.000000	2016.000000

- Overall summary of DataFrame

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
ID        271116 non-null int64
Name      271116 non-null object
Sex       271116 non-null object
Age       261642 non-null float64
Height    210945 non-null float64
Weight    208241 non-null float64
Team      271116 non-null object
NOC       271116 non-null object
Games     271116 non-null object
Year      271116 non-null int64
Season    271116 non-null object
City      271116 non-null object
Sport     271116 non-null object
Event     271116 non-null object
Medal     39783 non-null object
dtypes: float64(3), int64(2), object(10)
memory usage: 31.0+ MB
```

- we want to find out if there are any null values in columns. Check using pandas' isnull.

```
ID        False
Name      False
Sex       False
Age        True
Height     True
Weight     True
Team      False
NOC       False
Games     False
Year      False
Season    False
City      False
Sport     False
Event     False
Medal      True
dtype: bool
```

Querying Data:

Run different queries on data to extract further knowledge from data. We will discuss following important concepts and techniques..

Understanding Boolean Indexing:

Boolean indexing is used to perform general queries on a given pandas dataframe. This is an important concept to grasp. We will perform different operations on data to understand it e.g

- Count/Find how many records without any medal mentioned.
- Count/Find most young and most old people who got Gold medal
- Count/Find number of gold medals won by women of any specific country in a particular year

Explore some builtin functions:

We would explore some important panda library functions by using them e.g

- notnull
- loc
- Groupby
- Value_counts
- Pivot_table
- reindex

Cleaning and Completing Data:

At this point we are well aware of our data. We know that it has some missing values. We will perform different operations on it. E.g

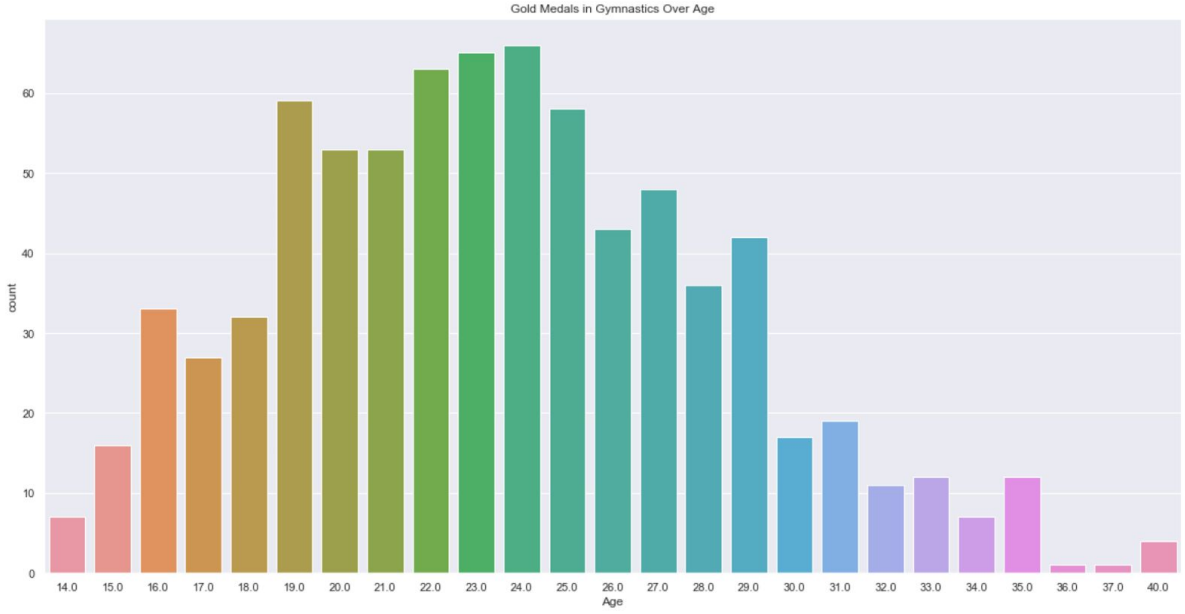
- Exclude all records from data where we don't have any information about medals.
- Fill missing age values with average age of other athletes.
- Fill missing height values for women and men with average height of women and men athletes respectively.
- Fill missing weight values for women and men with average weight of women and men athletes participating in same sports

Data Visualization:

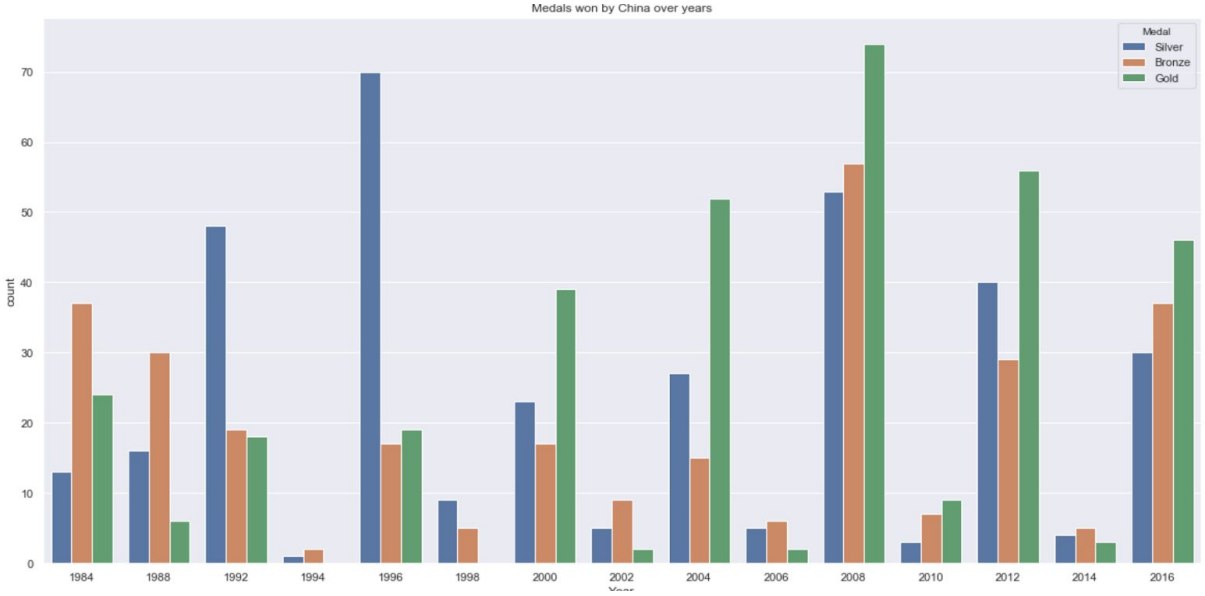
Visualizing data in different type of graphs will provide us with greater insights into our data. We will explore different options on visualizing our data and find out any patterns within it. From now on we will be using our previous knowledge of pandas library and try to grasp new concepts of seaborn and matplotlib.

Countplot examples:

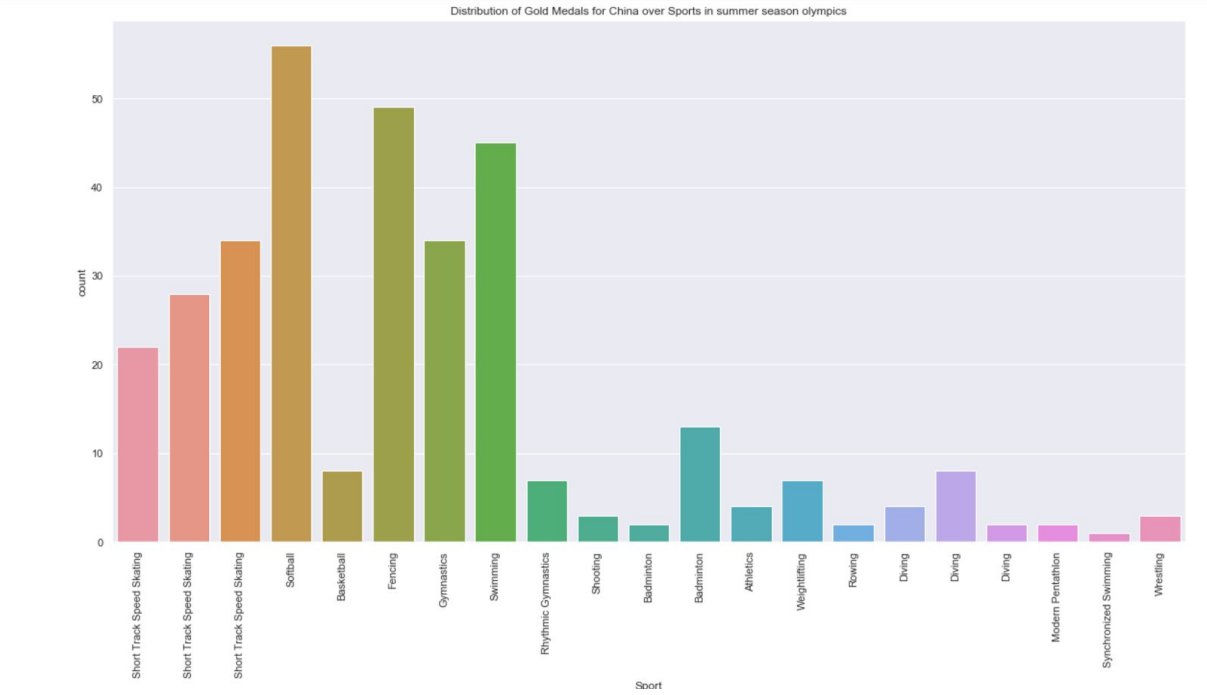
1. Gold medals in gymnastic over age



2. Medals won by China over years

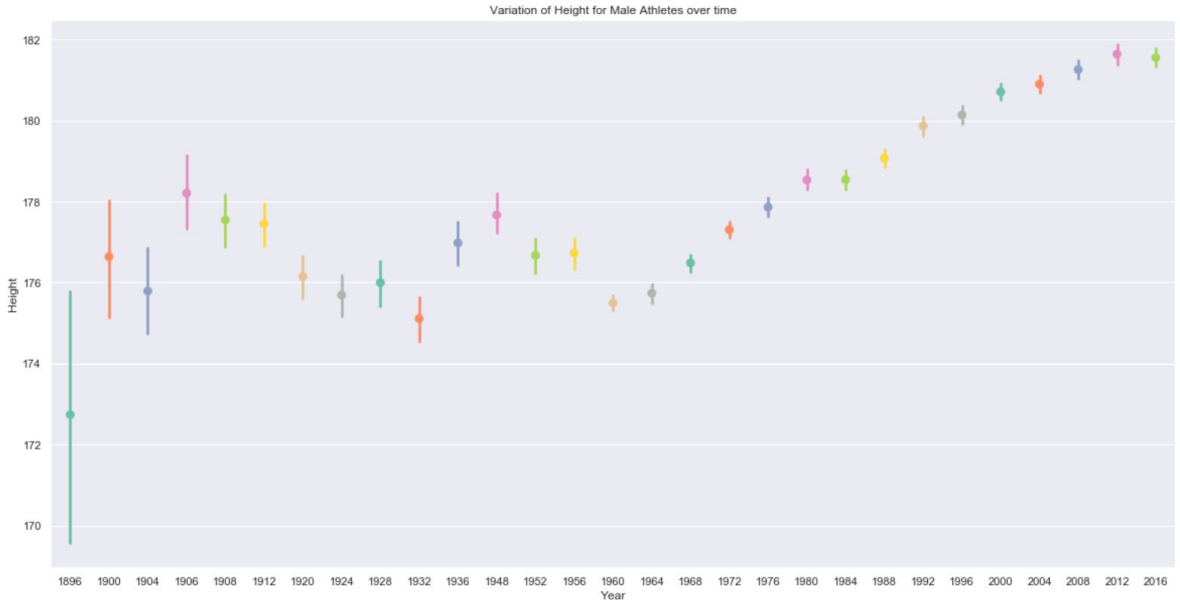


3. Gold medals won by china in summer olympics in sports

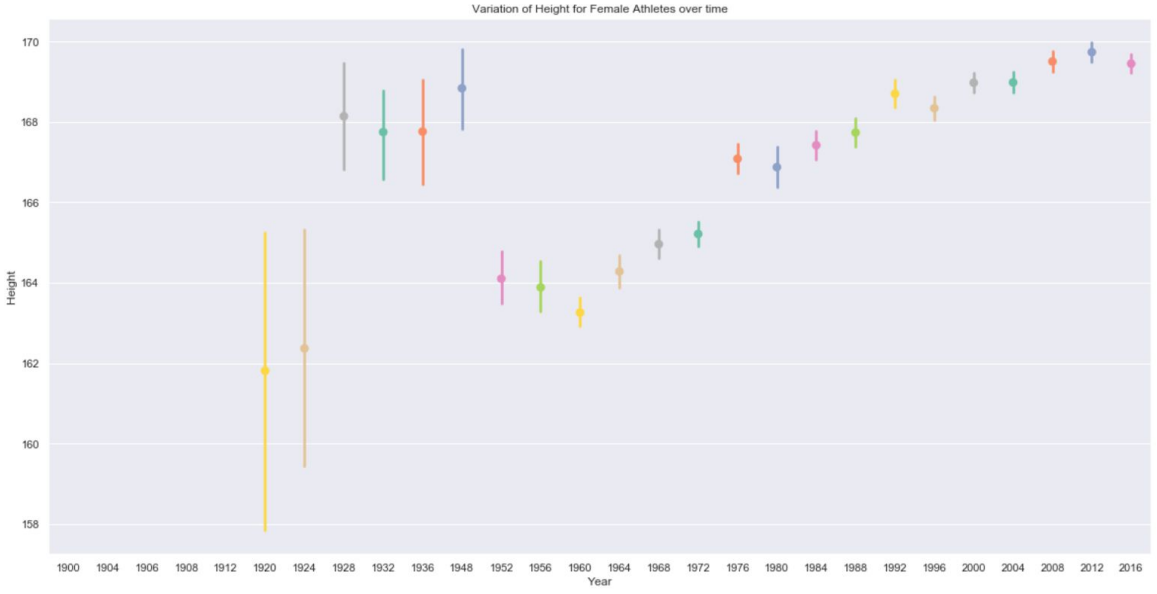


[Pointplot](#) examples:

1. Height of male athletes over years.

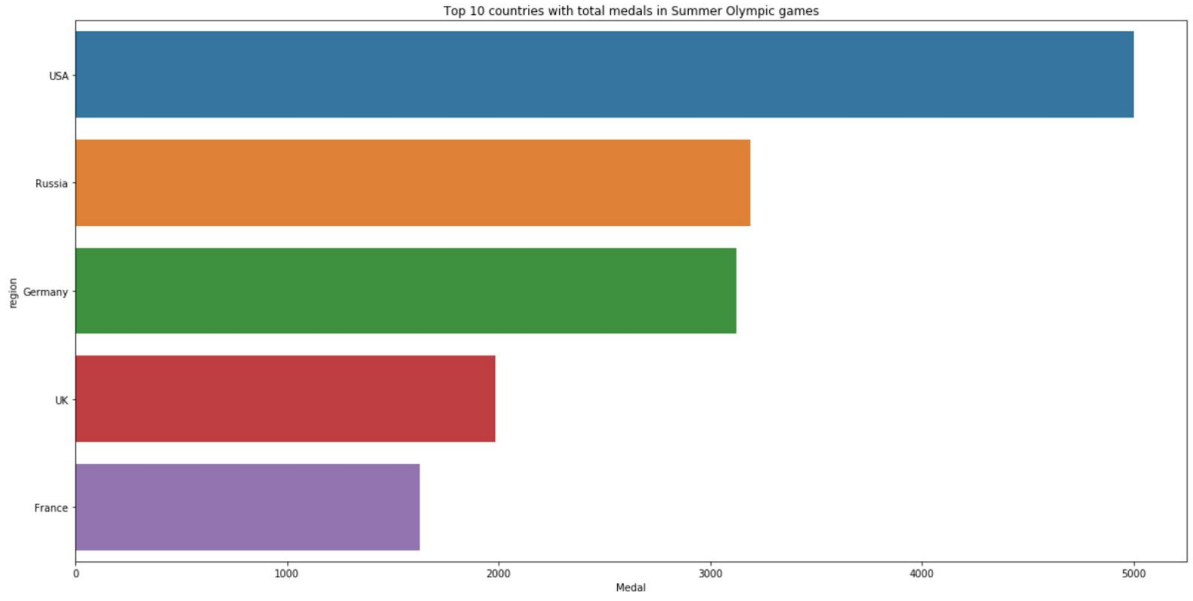


2. Height of female athletes over years.

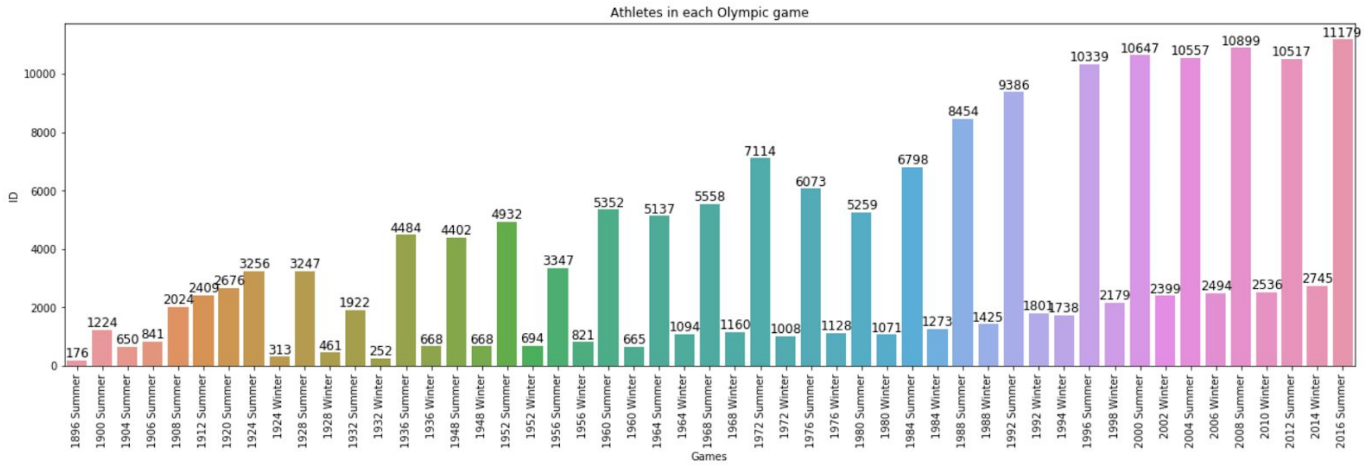


Barplot examples:

1. Top 5 countries with most medals

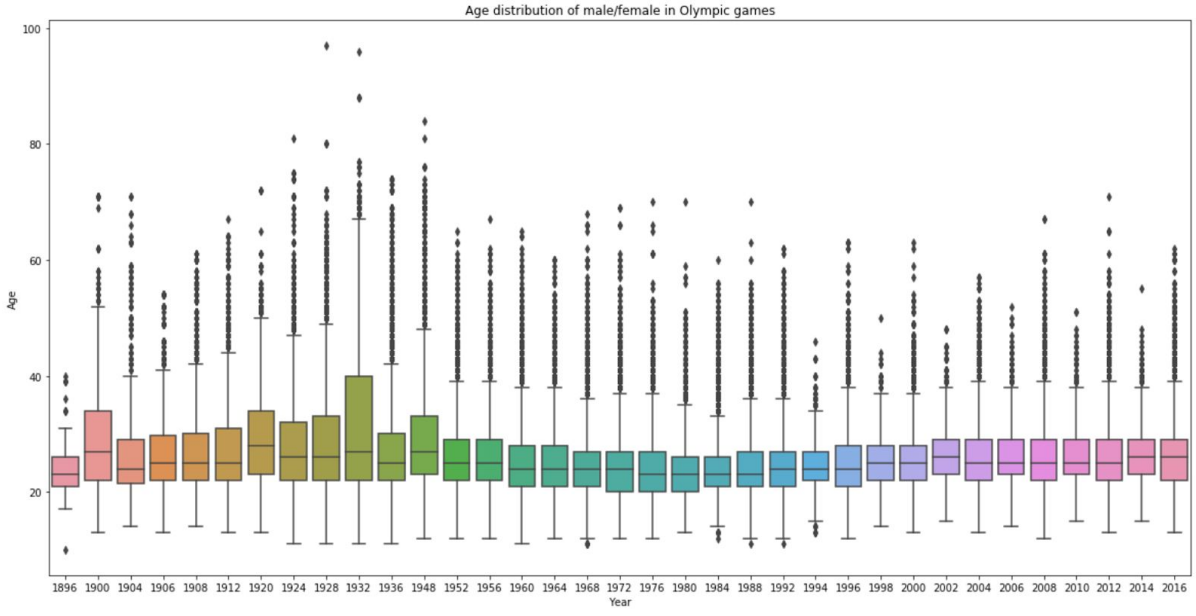


2. Number of athletes in each olympic game

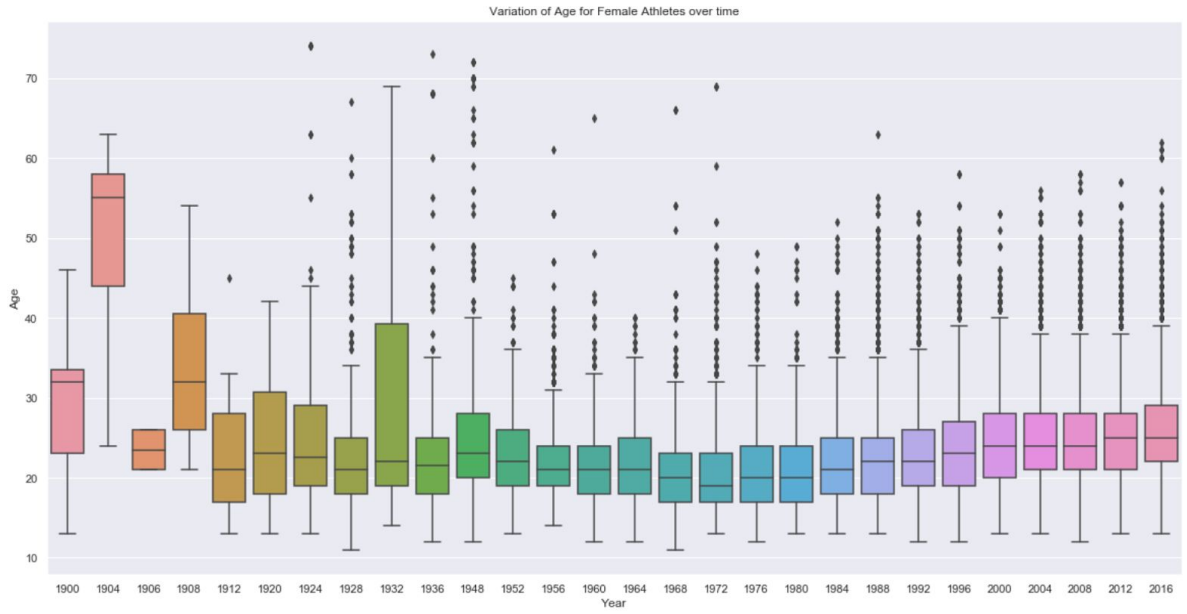


[Boxplot](#) Examples:

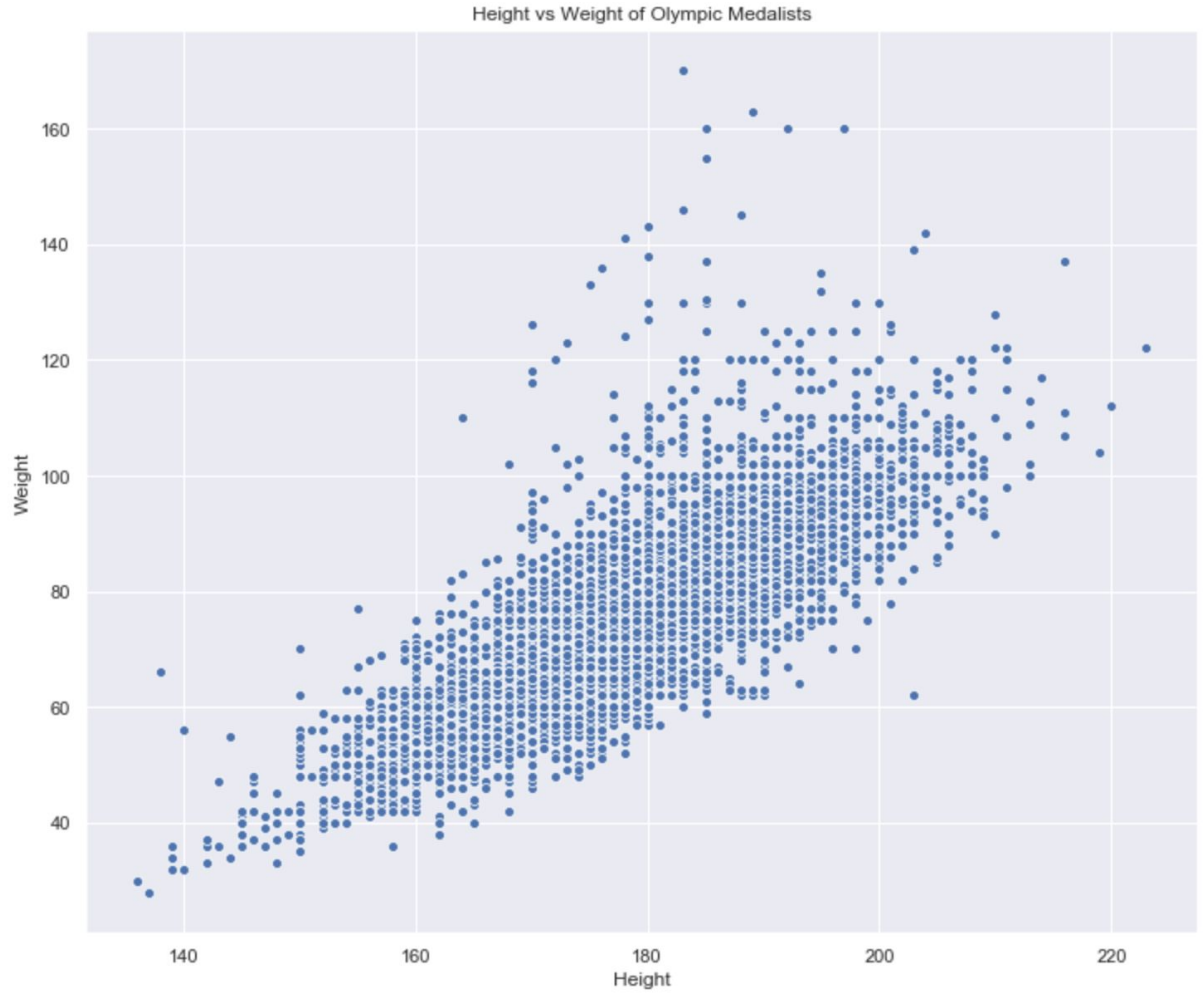
1. Age distribution of male/female in Olympic games



2. Variation of age for female over time

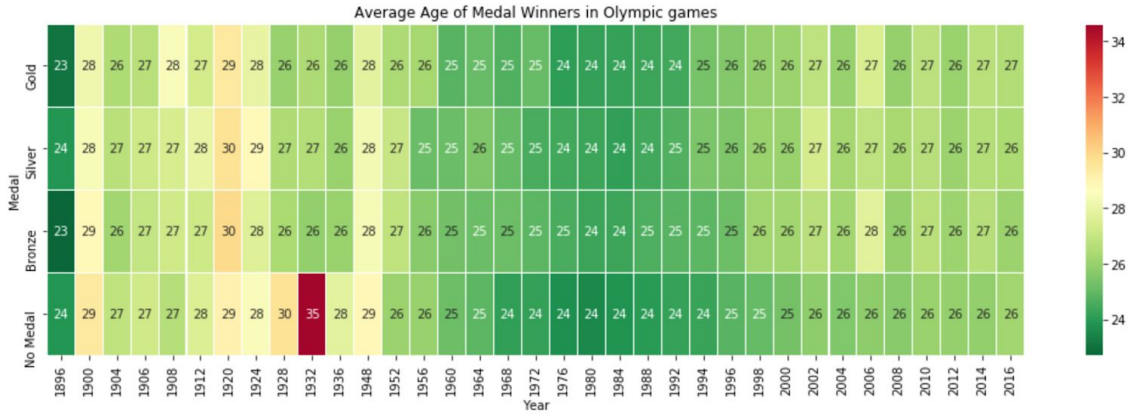


Scatterplot example:
Height and weight ratio of athletes



Heatmap example:

1. Average age of medal winners in olympic games.



In addition to this, please give some analysis trends and patterns while visualizing the data.

References:

- Data is taken from https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results
- This work is inspired by my fellow learners at kaggle:
    - https://www.kaggle.com/marcogdepinto/let-s-discover-more-about-the-olympic-games
    - https://www.kaggle.com/arunsankar/key-insights-from-olympic-history-data
    - And from other kaggle and great documentation of python libraries.