Outline

- **Uses/necessity of matplotlib**
- Tencent Use Case
- **Anatomy**
  - Figure
- Types of Data visualization
- Univariate Data Visualization
  - Categorical:
    - Bar chart
    - Countplot
    - Pie Chart
  - Continous
    - Histogram
    - KDE
    - Box and Whiskers Plot

Plots Presentation:

https://docs.google.com/presentation/d/1DkLTjTe6YmGbDHtr4v9Jso553DlCuP3cfSnwvUN1mgE
usp=sharing
(https://docs.google.com/presentation/d/1DkLTjTe6YmGbDHtr4v9Jso553DlCuP3cfSnwvUN1mg
usp=sharing)

Summary/Agenda

Where is all Data Visualization helpful? Why?

- Exploratory - EDA
- Explanatory - Storytelling

What is the Science in Data Visualization?

- Anatomy of a plot/chart
- How to use the right plot/chart for given data?

What is the Art in Data Visualization?

- Choose the right scale, labels, tick labels
- Identify and remove clutters in the plot
- Ways to highlight information in the plot

Importing Matplotlib and Seaborn

We don't need to import the entire library but just its submodule `pyplot`

We'll use the **alias name** `plt`

What is `pyplot` ?

- `pyplot` is a **sub-module for visualization** in `matplotlib`
- Think of it as **high-level API** which **makes plotting an easy task**
- Data Scientists **stick to using `pyplot` only unless** they want to create **something totally new**.

For seaborn, we will be importing the whole seaborn library as alias `sns`

What is seaborn?

Seaborn is another visualization library which uses matplotlib in the backend for plotting

What is the major difference then between both matplotlib and seaborn?

- Seaborn uses **fascinating themes** and **reduces number of code lines** by doing a lot of work in the backend
- While matplotlib is used to **plot basic plots and add more functionlaity** on top of that
- Seaborn is built on the top of Pandas and Matplotlib

As we proceed through the lecture, we will see the difference between both the libraries

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Before we dive into learning these libraries, lets answer some general questions

Why do even we need to visualize data? When do I even need to visualise?

Two reasons/scopes

- **Exploratory** - I can't see certain patterns just by crunching numbers (avg, rates, %ages)
- **Explanatory** - I can the numbers crunches and insights ready, but I'd like a visual art for storytelling

---

Lets talk about Science of Data Visualisation

Data

- Rows: Samples, Data-points, Records
- Columns: Features, Variables

How many kinds of data do we have?

At the fundamental level, it's just two types:

- Numerical/Continous
- Categorical

Categorical can be further divided into:

- **Ordinal:** Categorical Data with an order (E.g. low, medium, high)
- **Non-ordinal/nominal:** Categorical Data without any order (example gender as Male/Female)

Video Games Analysis

You are a data scientist at "Tencent Games". \

You need to analyze what kind of games they should start creating to get higher success in the market. \

In [ ]:
```
1 !wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/021/
```

```
--2022-12-12 10:16:41--  https://d2beiqkhq929f0.cloudfront.net/public_asset
s/assets/000/021/299/original/final_vg1_-_final_vg_%281%29.csv?1670840166 (h
ttps://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/021/299/origin
al/final_vg1_-_final_vg_%281%29.csv?1670840166)
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 1
3.226.251.24, 13.226.251.62, 13.226.251.107, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|
13.226.251.24|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2041483 (1.9M) [text/plain]
Saving to: ‘final_vg.csv’

final_vg.csv        100%[===================>]   1.95M  1.42MB/s    in 1.4s

2022-12-12 10:16:43 (1.42 MB/s) - ‘final_vg.csv’ saved [2041483/2041483]
```

In [ ]:
```
import pandas as pd
import numpy as np
data = pd.read_csv('final_vg.csv')
data.head()
```

Out[47]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439352
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033887
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493442
3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	3.228043
4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	1.440483

If you notice,

- Columns like `Platform`, `Genre` are categorical
- While columns like `NA_Sales`, `Global_Sales`, `Rank` are continuous

On noticing further,

- `Platform` is of nominal type, no proper order between the categories
- `Year` is of ordinal type, there's a order to the categories

Introduction to Matplotlib

Lets learn to create a basic plot using plt

Now say, we want to draw a curve passing through 3 points:

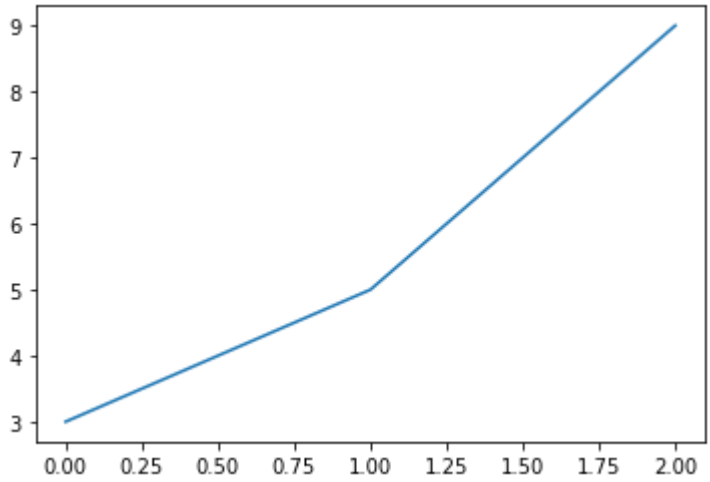
- (0, 3)
- (1, 5)
- (2, 9)

How can we draw a curve using matplotlib ?

By using `plt.plot() function`

In [ ]:
```
x_val = [0, 1, 2]
y_val = [3, 5, 9]
plt.plot(x_val, y_val)
```

Out[48]:
```
[<matplotlib.lines.Line2D at 0x7fcb8faf4790>]
```

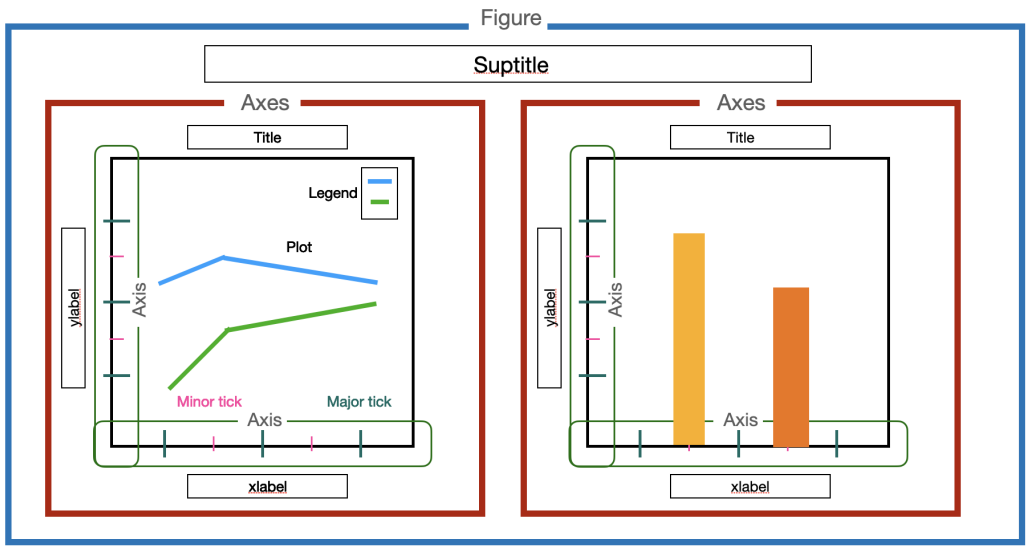


What can we observe from this plot ?

- `plt.plot()` automatically decided the scale of the plot
- It also prints the **type of object** `matplotlib.lines.Line2D`

While this command decided a lot of things for you, you can customise each of these by understanding **components of a matplotlib plot**

Anatomy of Matplotlib



Woah! There is a lot of information in this image. Let's understand them one at a time.

- Figure: The **overall window** or page that everything is drawn on.
  - You can create multiple independent Figures in Jupyter.
  - If you run the code in terminal, separate windows will pop-up
- Axes: To the figure you can add multiple **Axes** which represents a plot
- **Axis**: Simply the `x-axis` and `y-axis`
- **Axes**: - It is the **area** on which the **data is plotted** with functions such as `plot()`
  - **x-label**: Name of x-axis
  - **y-label**: Name of y-axis
  - **Major ticks**: subdivides the axis into major units. They appear by default during plotting
- **Minor ticks**: subdivides the major tick units. They are by default hidden and can be toggled on.
- **Title**: Title of each plot **(Axes)**, giving information about the same
- **Legend**: describes the elements in the plot, blue and green curves in this case
- **Suptitle**: The common title of all the plots

These are the major components of a matplotlib plot

Now, how to choose the right plot?

Firstly, depends on the what is your question of interest

When the question is clear:

- How many variables are involved?
- Whether the variable(s) are numerical or categorical?

How many variables are involved?

- 1 Variable - Univariate Analysis
- 2 Variables - Bivariate Analysis
- 2+ Variables - Multivariate Analysis

PS: Bivariate counts under multivariate, but let's keep it sep for ease of communication

What are the possible cases?

Univariate

- Numerical
- Categorical

Bivariate

- Numerical-Numerical
- Numerical-Categorical
- Categorical-Categorical

Multivariate

Let's start with 3 and then we can generalize

- Numerical-Numerical-Categorical
- Categorical-Categorical-Numerical
- Categorical-Categorical-Categorical
- Numerical-Numerical-Numerical

We will work on these one by one

Univariate Data Visualization - Categorical Data

What kind of questions we may want to ask for a categorical variable?

Questions like:

- What is the Distribution/Frequency of the data across different categories?
- What proportion does a particular category constitutes?

...and so on

Let's take the categorical column "Genre"

How can we find the top-5 genres?

Recall, how could we get this data using pandas?

In [ ]:
```
cat_counts = data['Genre'].value_counts()
cat_counts
```

Out[49]:
```
Action          3316
Sports          2400
Misc            1739
Role-Playing    1488
Shooter         1310
Adventure       1286
Racing          1249
Platform         886
Simulation       867
Fighting         848
Strategy         681
Puzzle           582
Name: Genre, dtype: int64
```

Now what kind of plot can we use to visualize this information?

- We can perhaps plot categories on X-axis and their corresponding frequencies on Y-axis
- Such chart is called a Bar Chart or a Count Plot
- Can also plot horizontally when the #categories are many

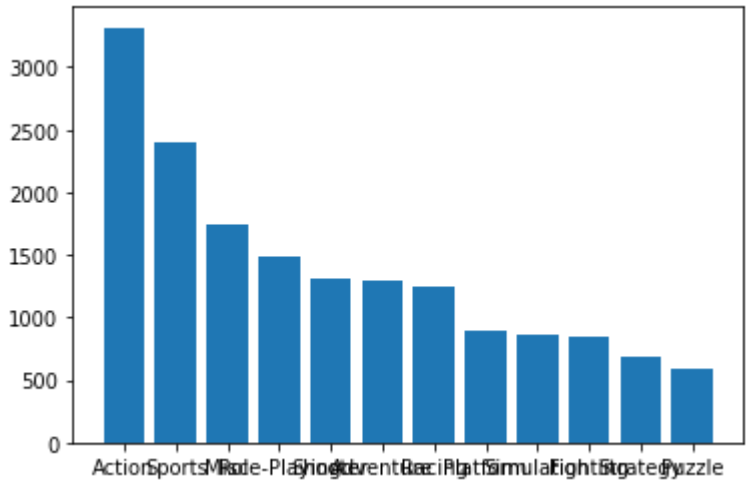
Bar Chart

The data is binned here into categories

How can we draw a Bar plot ?

In [ ]:
```
x_bar=cat_counts.index
y_bar=cat_counts
plt.bar(x_bar,y_bar)
```

Out[50]: <BarContainer object of 12 artists>



The names seem to be overlapping a lot

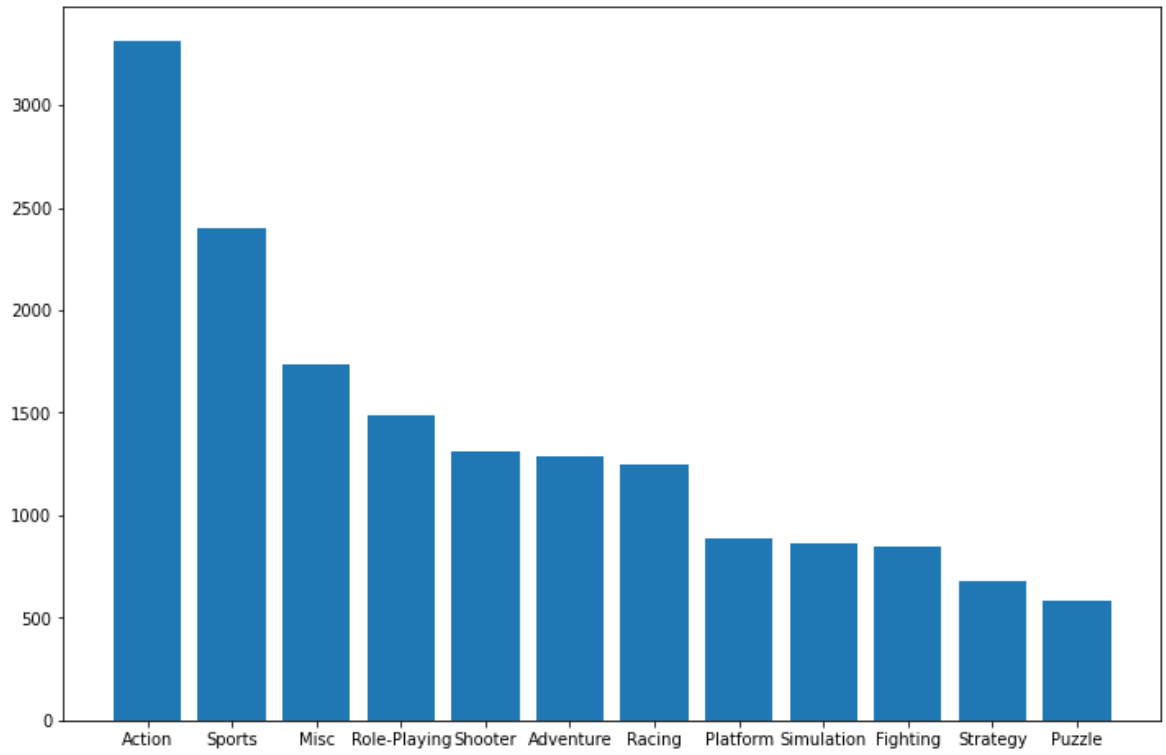
How can we handle overlapping labels?

- Maybe decrease the font size (not preferred though)
- Or maybe increase the figure size
- Or rotate the labels

How can we change the plot size?

In [ ]:
```
plt.figure(figsize=(12,8))
plt.bar(x_bar,y_bar)
```

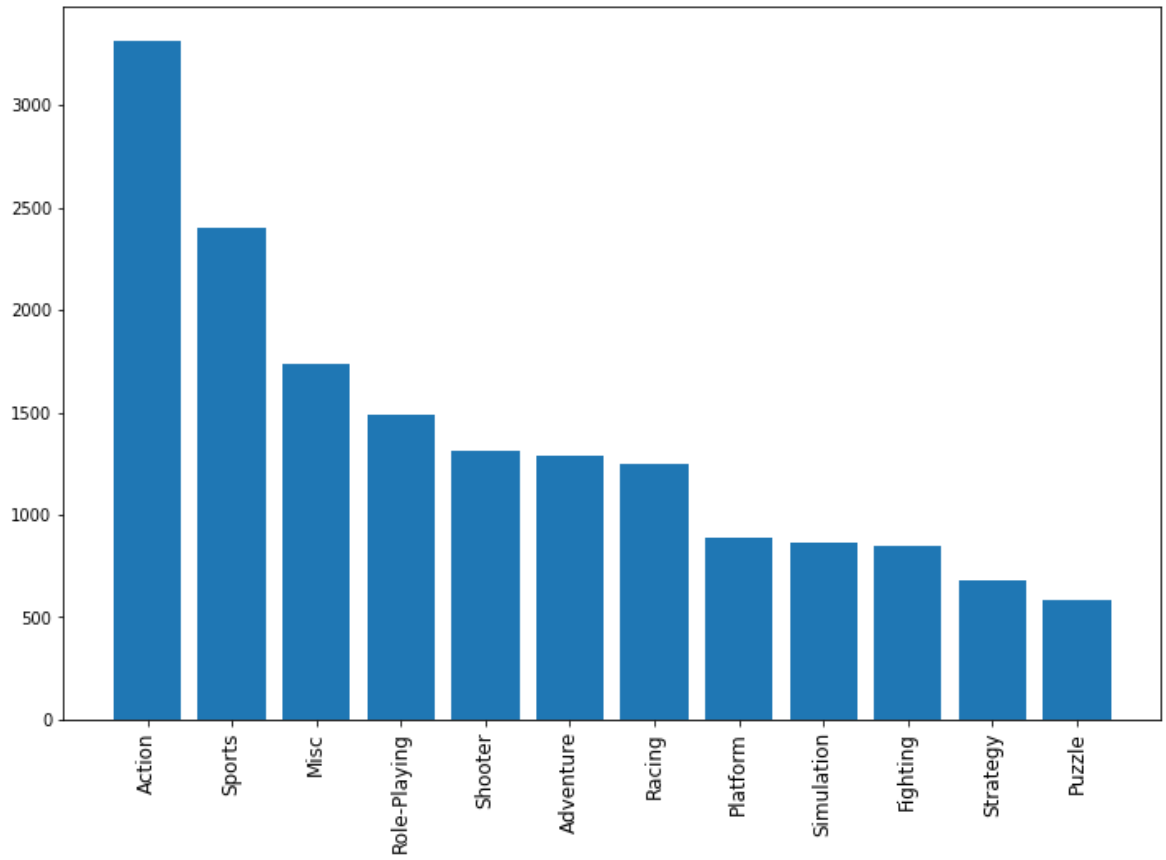
Out[51]: <BarContainer object of 12 artists>



And how can we rotate the tick labels, also maybe increase the fontsize of the same?

In [ ]:
```
plt.figure(figsize=(12,8))
plt.bar(x_bar,y_bar)
plt.xticks(rotation=90, fontsize=12)
```

Out[52]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 <a list of 12 Text major ticklabel objects>)
```

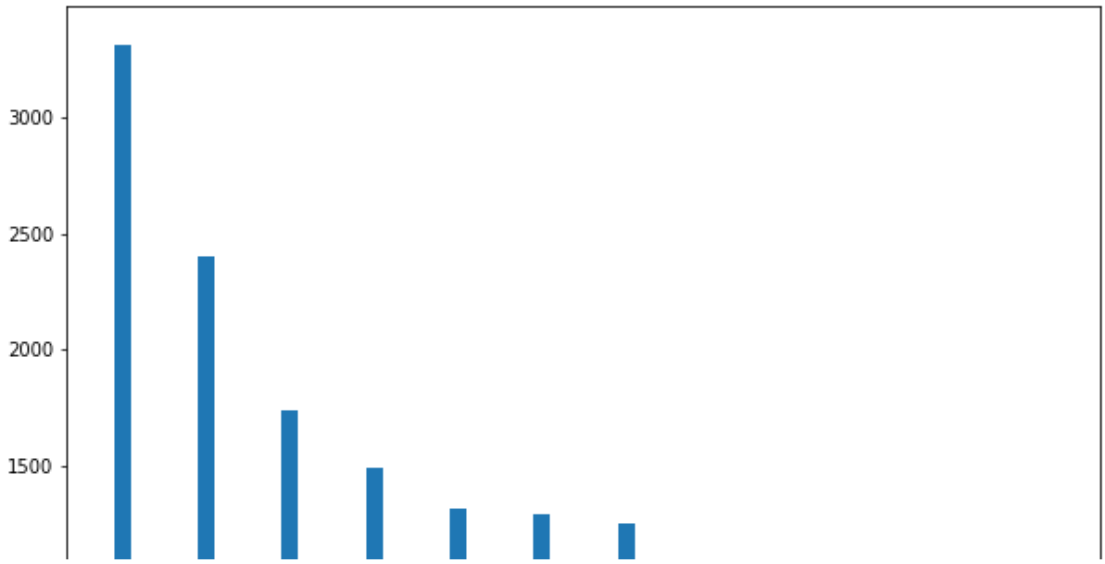


If you notice, the width of each bar is **1**

Can we change the width of these bars?

In [ ]:
```
# same code
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2)
plt.xticks(rotation = 90, fontsize=12)
```

Out[53]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 <a list of 12 Text major ticklabel objects>)
```

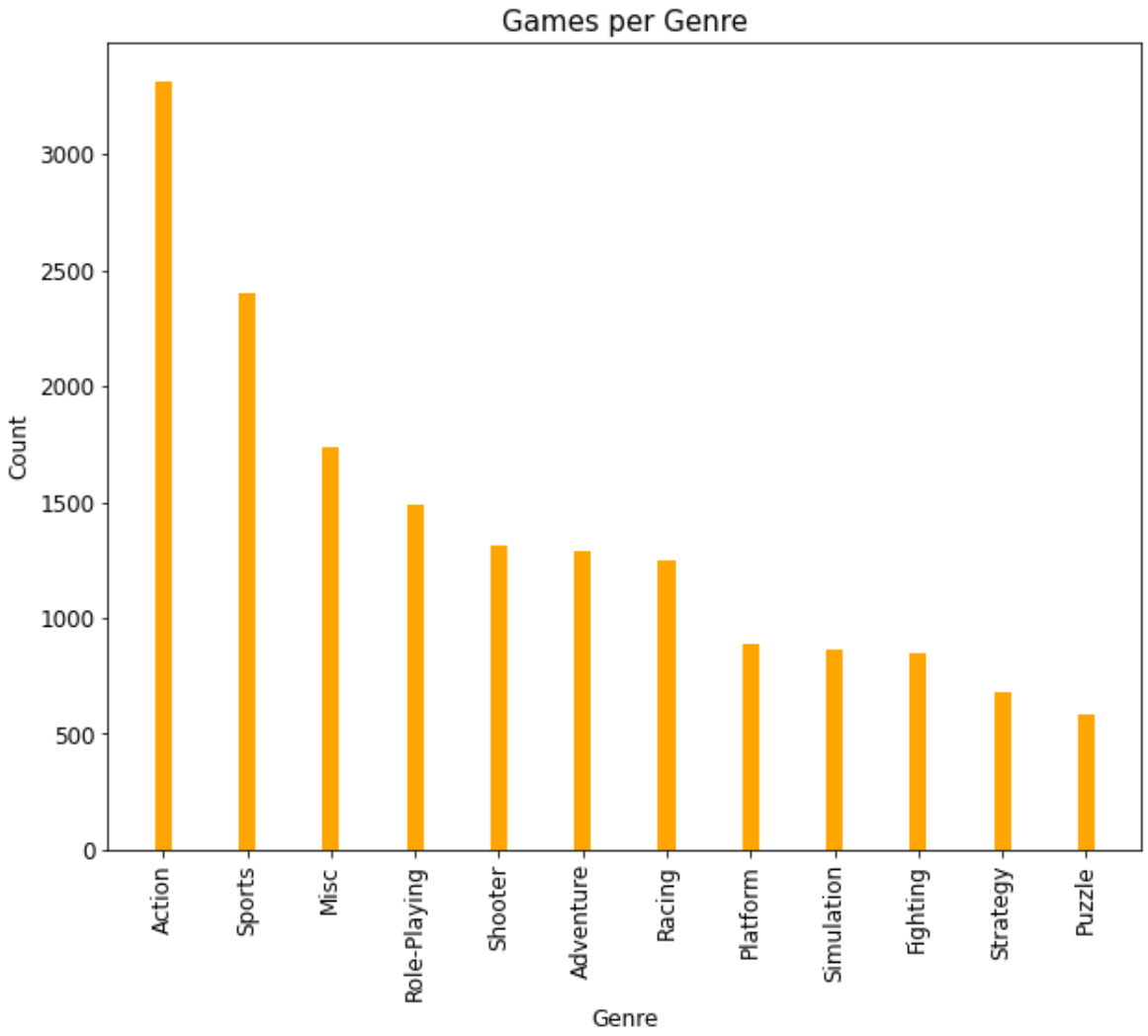


What about any additional styling to add to the bars ?

- We can **change colour of bars**
- We can add a **title to the axes**
- We can also add x and y labels

In [ ]:
```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2,color='orange')
plt.title('Games per Genre',fontsize=15)
plt.xlabel('Genre',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.xticks(rotation = 90, fontsize=12)
plt.yticks(fontsize=12)
```

Out[54]:
```
(array([   0.,  500., 1000., 1500., 2000., 2500., 3000., 3500.]),
 <a list of 8 Text major ticklabel objects>)
```



If you notice, there's some text printed always before the plots.

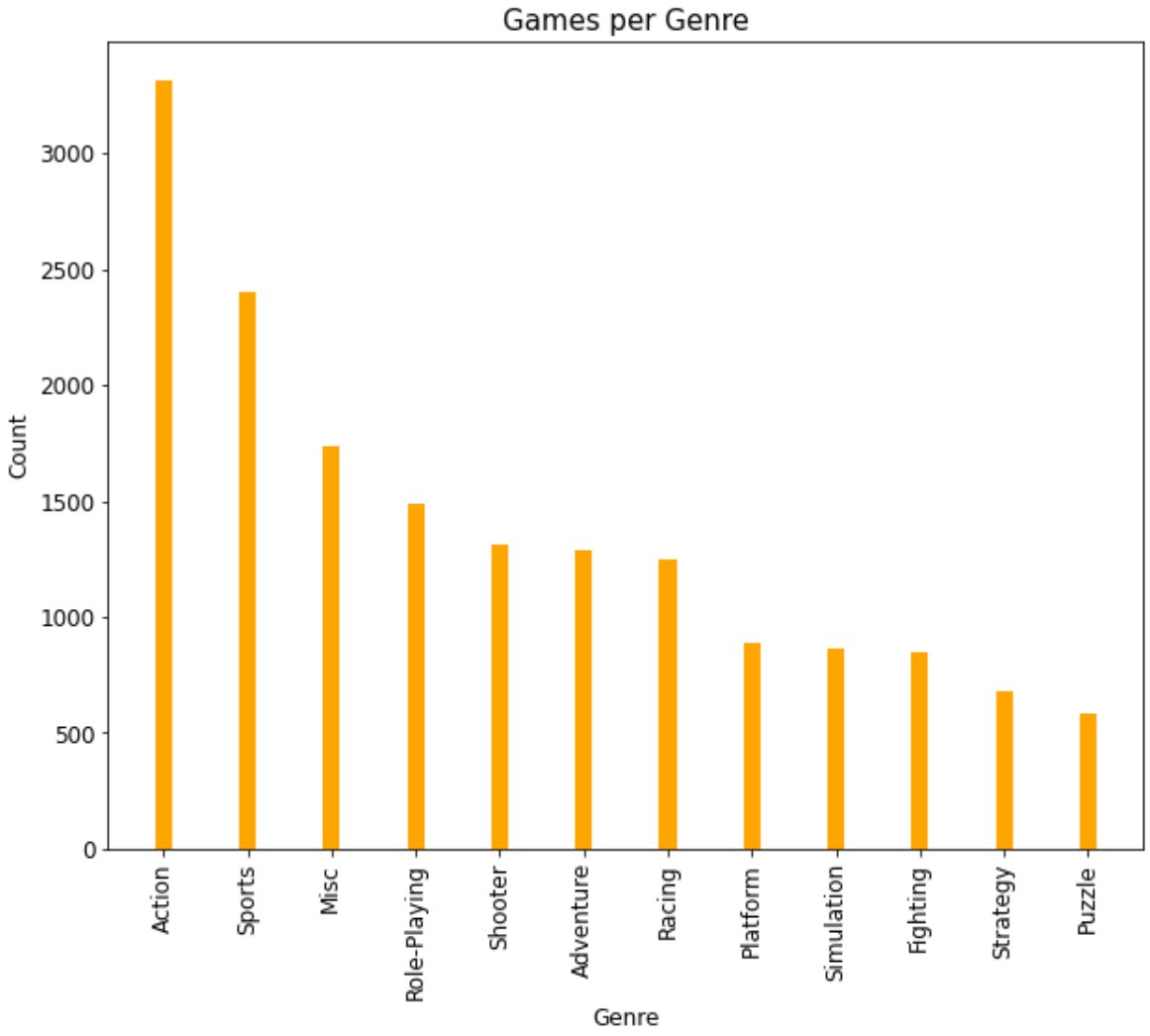
This contains the data information of the plot

How can we remove the text printed before the plot and just display the plot?

Using plt.show() at the end

In [ ]:

```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2,color='orange')
plt.title('Games per Genre',fontsize=15)
plt.xlabel('Genre',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.xticks(rotation = 90, fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



How can we draw a bar-chart in Seaborn?

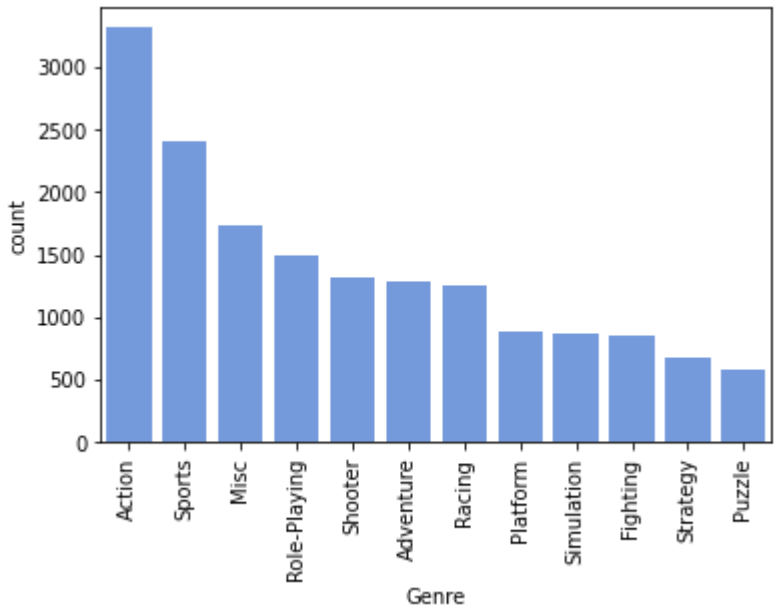
- In Seaborn, the same plot is called as **countplot**.
- Countplot automtically does even the counting of frequencies for you

Why not called a barplot?

There is **another function** in Seaborn called **barplot which has some other purpose** - discuss later

In [ ]:
```
sns.countplot(x = 'Genre', data = data, order=data['Genre'].value_counts(
plt.xticks(rotation=90)
```

Out[56]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 <a list of 12 Text major ticklabel objects>)
```



The top 5 genres are action, sports, misc, role player, and shooter

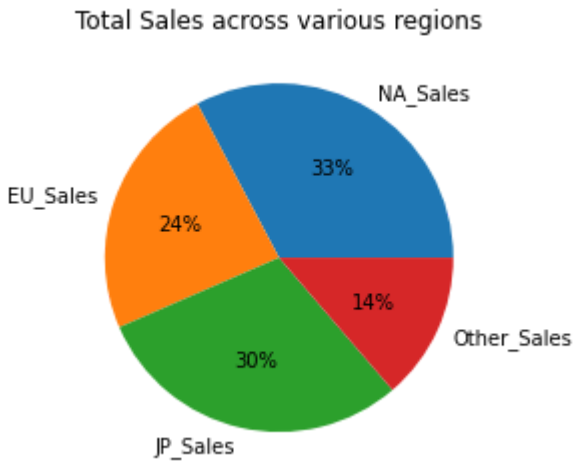
Pie charts

What if instead of actual frequencues, I want see the proportion of the categories with each other?

Say, we want to compare the distrubution/proportion of sales across the different regions?

Which plot can we use for this?

A pie-chart!



Do refer to the postread for the code of how to plot the pie-chart

Univariate Data Visualisation - Numerical Data

What kind of questions we may have regarding a numerical variable?

1. How is the data distributed? Say distribution of number of games published in a year.
2. Is the data skewed? Are there any outliers? - Extremely high selling games maybe?
3. How much percentage of data is below/above a certain number?
4. Some special numbers - Min, Max, Mean, Median, nth percentile?

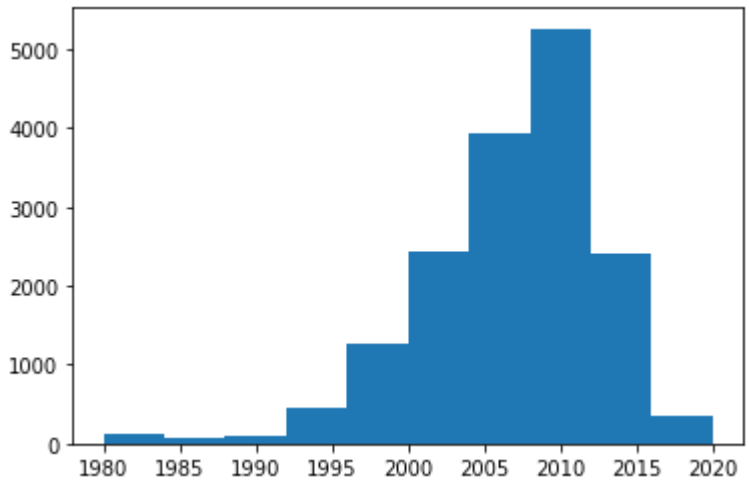
Now say, you want find the distribution of games released every year.

Unlike barplot, **to see the distribution we will need to `bin` the data**.

How can we understand popularity of video games year by year?

Histogram

In [ ]:
```
1 plt.hist(data['Year'])
2 plt.show()
```



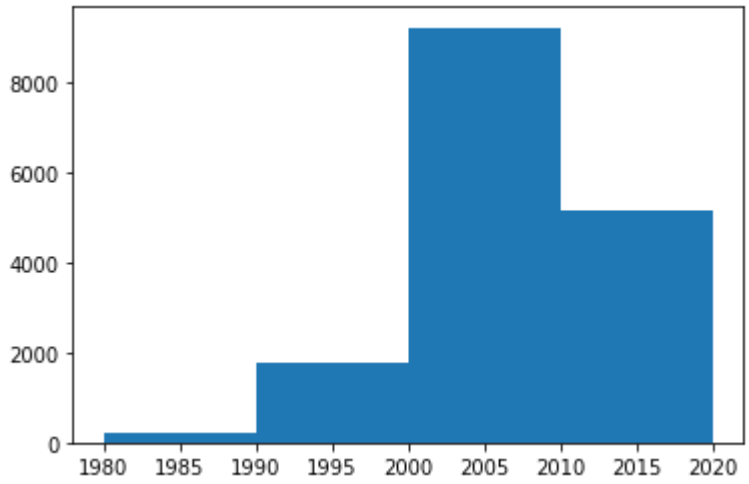
- The curve is left skewed, with a lot more games being published in 2005-2015
- This shows that games started being highly popular in the last 1-2 decades, maybe could point to increased usage of internet worldwide!

If you notice, histograms are basically frequency charts

We can also vary the number of bins, the **default number of bins is 10**

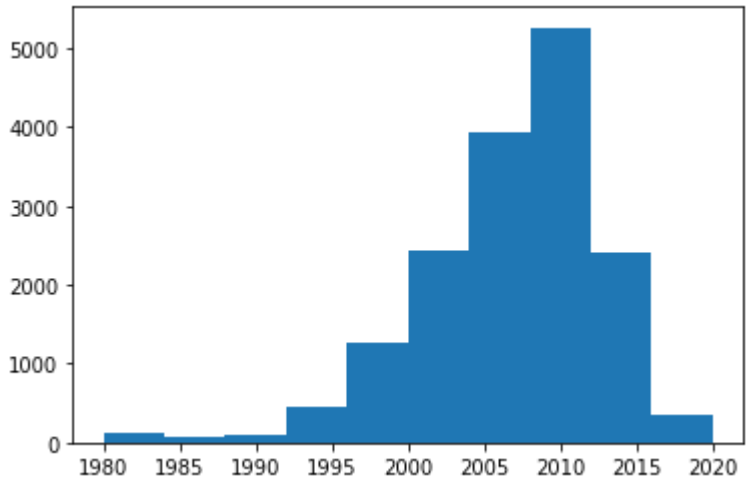
So if we would need to see this data per decade, we would need 40 years in 4 bins.

In [ ]:
```
plt.hist(data['Year'], bins=4)
plt.show()
```



We can also get the data of each bin, such as range of the boundaries, values, etc.

In [ ]:
```
count, bins, _ = plt.hist(data['Year'])
```



In [ ]:
```
count
```

Out[60]:
```
array([ 112.,   70.,   92.,  449., 1274., 2440., 3921., 5262., 2406.,
        355.])
```

In [ ]:
```
bins
```

Out[61]:
```
array([1980., 1984., 1988., 1992., 1996., 2000., 2004., 2008., 2012.,
       2016., 2020.])
```

Now what do these `count` and `bins` mean?

- **bins** provides bin edges

- **counts** provides it corresponding counts

What is the length of `count` ?

10

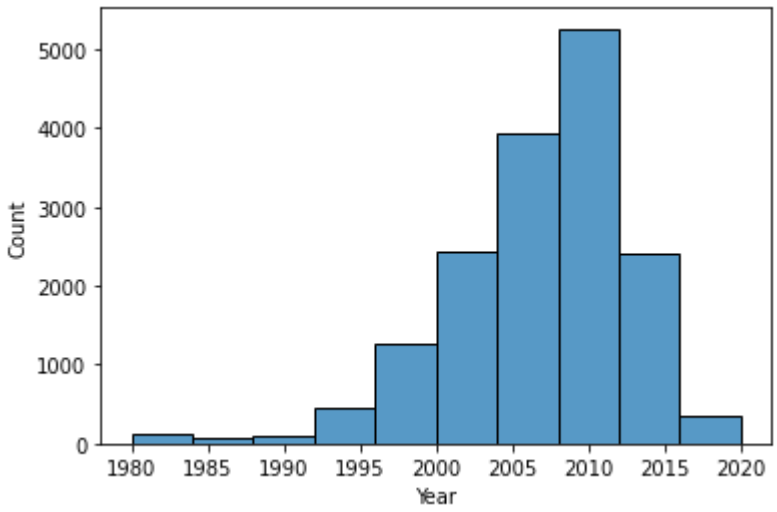
What shoud be the length of `bins` ?

10 + 1 = 11

How can we plot histogram in Seaborn?

In [ ]:
```
1 sns.histplot(data['Year'], bins=10)
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcb8f61deb0>
```



Notice,

- The boundaries are more defined than matplotlib's plotting
- The x and y axis are labelled automatically

Kernel Density Estimate (KDE) Plot

- A KDE plot, similar to histrogram, is a method for visualizing the distributions
- But instead of bars, KDE represents data using a **continuous probability density curve**

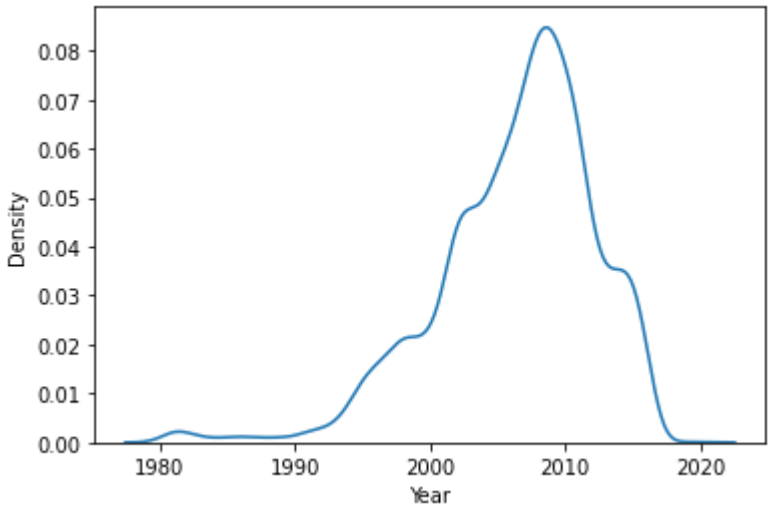
Now, Why do we even need KDE plots?

- Compared to histogram, KDE produces a plot which is **less cluttered** and **more interpretable**
- Think of it as a **smoothened version** of histogram

Let's plot KDE using `seaborn` 's `kdeplot`

In [ ]:
```
1 sns.kdeplot(data['Year'])
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcb8f64f220>



Can you notice the difference between KDE and histogram?

Y-Axis has **probability density estimation** instead of count

You can read more about this on:

https://en.wikipedia.org/wiki/Kernel_density_estimation (https://en.wikipedia.org/wiki/Kernel_density_estimation)
https://www.youtube.com/watch?v=DCgPRaIDYXA (https://www.youtube.com/watch?v=DCgPRaIDYXA)

Boxplot

Now say I want to find the typical earnings of a game when it is published.

Or maybe find the aggregates like median, min, max and percentiles of the data.

What kind of plot can we use to understand the typical earnings from a game?

Box Plot

What exactly is a Box Plot?

- A box plot or **box-and-whisker plot** shows the **distribution of quantitative data**
- It facilitates comparisons between
  - attributes
  - across levels of a categorical attribute.

The **box**: Shows the **quartiles** of the dataset

The **whiskers**: Show the **rest of the distribution**

Let's go through the terminology one-by-one

Box plots show the five-number summary of data:

1. Minimum score,
2. first (lower) quartile
3. Median
4. Third (upper) quartile
5. maximum score

Minimum Score

- It is the **lowest value**, excluding outliers
- It is shown at the **end of bottom whisker**

Lower Quartile

- **25% of values** fall below the lower quartile value
- It is also known as the **first quartile**.

Median

- Median marks the **mid-point of the data**
- **Half the scores are greater than or equal to this value and half are less**.
- It is sometimes known as the **second quartile**.

Upper Quartile

- **75% of the values fall below the upper quartile value**
- It is also known as the **third quartile**.

Maximum Score

- It is the **highest value**, excluding outliers
- It is shown at the **end of upper whisker**.

Whiskers

- The upper and lower whiskers represent **values outside the middle 50%**
- That is, the **lower 25% of values** and the **upper 25% of values**.

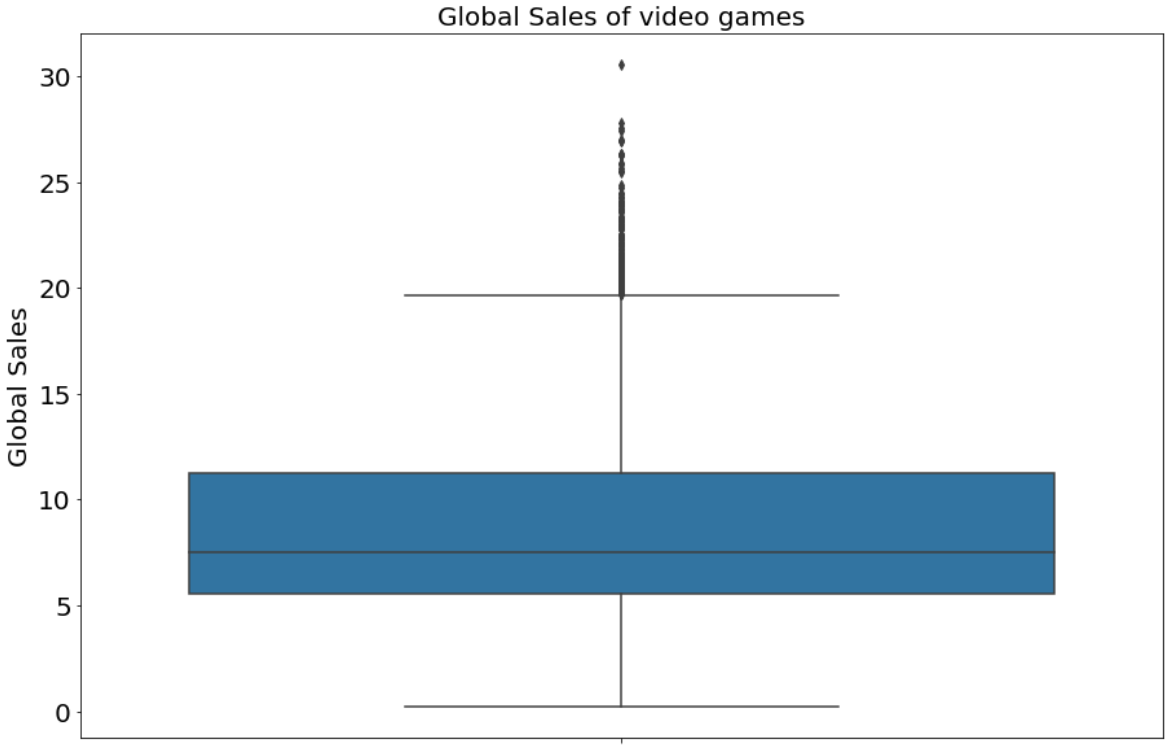
Interquartile Range (or IQR)

- This is the box plot showing the **middle 50% of scores**
- It is the **range between the 25th and 75th percentile**.

Now, Let's plot a box plot to find the average typical earnings for a game

In [ ]:
```
plt.figure(figsize=(15,10))
sns.boxplot(y = data["Global_Sales"])
plt.yticks(fontsize=20)
plt.ylabel('Global Sales', fontsize=20)
plt.title('Global Sales of video games', fontsize=20)
```

Out[64]: Text(0.5, 1.0, 'Global Sales of video games')



What can we infer from this?

The 5 point estimates (approx.) here are:

- Minimum, excluding outliers: 0
- Maximum, excluding outliers: 6.5 million dollars
- 25th Quantile: 2.5 million
- Median: around 3 million
- 75th Quantile: 4 million

There are few outliers towards 6-7 million dollars

Key Takeaways:

Categorical - Barplot, Pie Chart

Numerical - Histogram, KDE, Boxplot

Can explore more types: Violin plot, bee-swarm plot, etc.