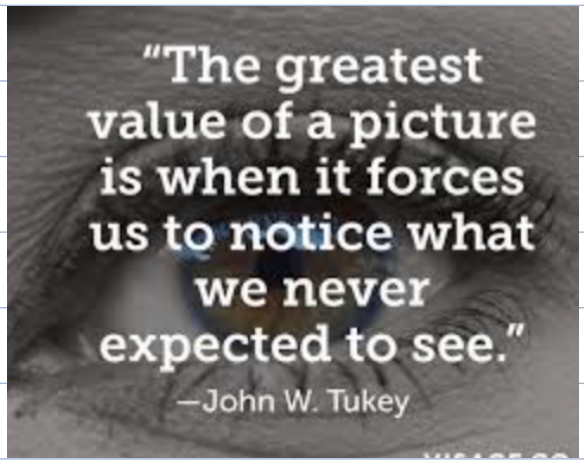
Content

- Intro to Matplotlib & Seaborn
- Tencent Use Case
- Anatomy of Matplotlib
- Components of a Matplotlib plot
- Univariate Data Visualization
    - Categorical
        - Bar chart
        - Countplot
        - Pie chart
    - Continous
        - Histogram
        - KDE plot
        - Box and Whiskers plot



Importing Matplotlib & Seaborn

In case of `matplotlib`,

- We don't need to import the entire library but just its sub-module `pyplot`.
- We'll use the alias name `plt`.

What is `pyplot`?

- `pyplot` is a **sub-module for visualization** in `matplotlib`.
- Think of it as a **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?

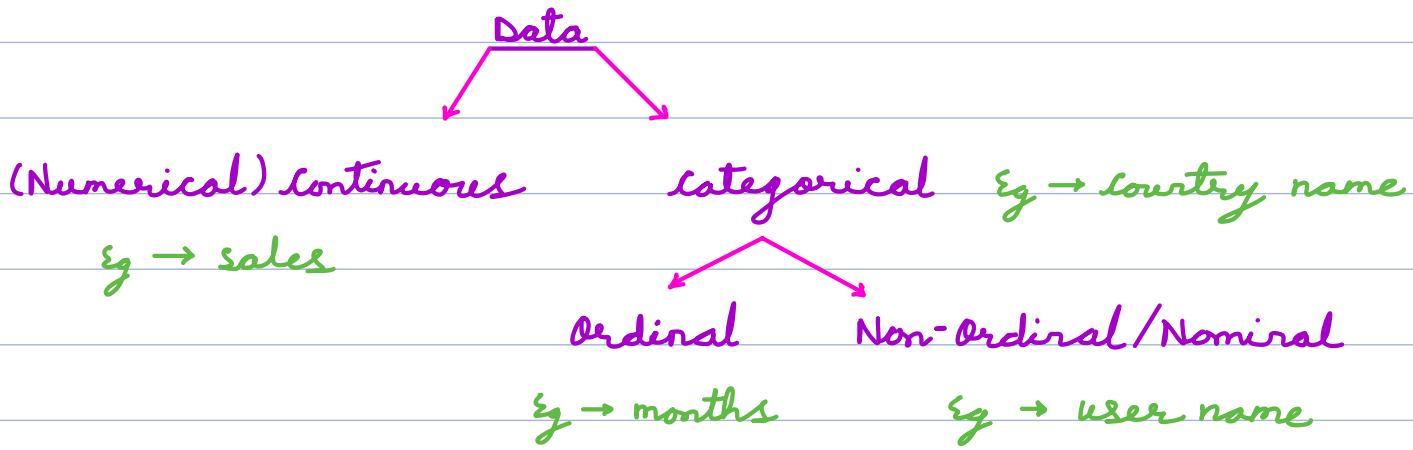
Seaborn is another visualization library which uses Matplotlib in the backend for plotting.

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

- `Seaborn` is built on the top of `Pandas` and `Matplotlib`.
- 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.

Why do even we need to visualize data?

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



Video Games Analysis

You are a Data Scientist at "**Tencent Games**".

You need to analyze what kind of games the company should create in order to perform better in the market.

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439352	1.991671	12.802935
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033887	0.394830	7.034163
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493442	0.408693	4.982552
3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	3.228043	0.394830	7.226880
4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	1.440483	1.493442	8.363113

Notice that,

- columns like `Platform`, `Genre` are Categorical
- columns like `NA_Sales`, `Global_Sales`, `Rank` are Continuous

Furthermore,

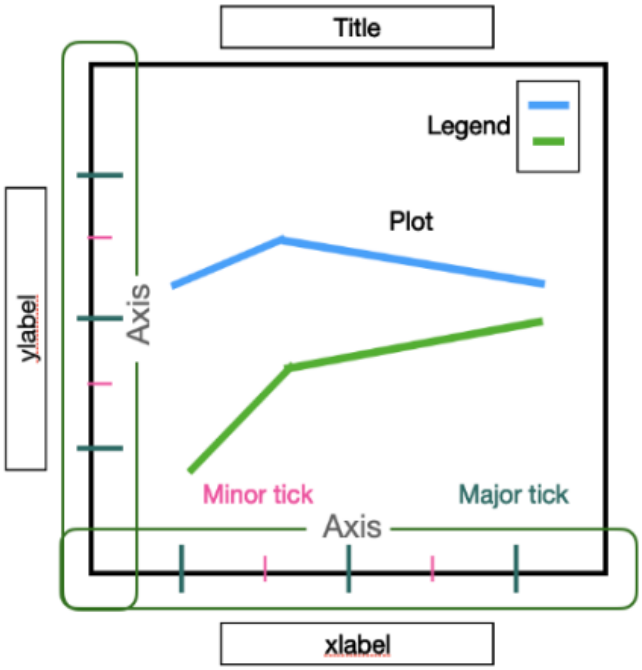
- `Platform` is of nominal type (no proper order between the categories)
- `Year` is of ordinal type (an order exists between the categories)

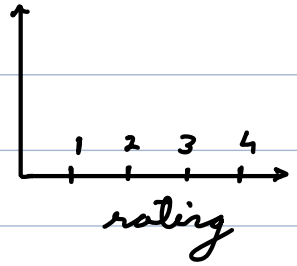
How can we draw a curve using `matplotlib`?

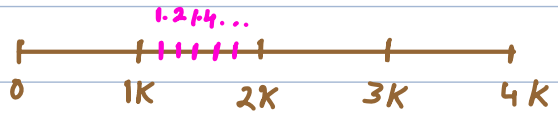
- by using the `plt.plot()` function

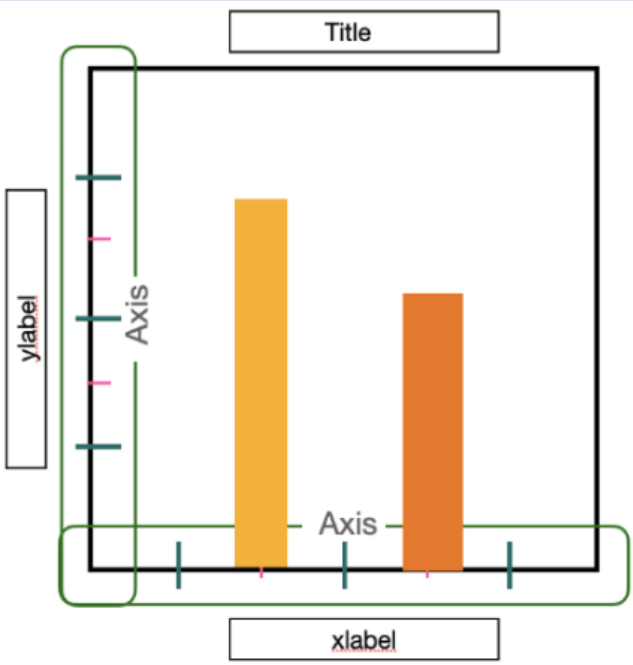
- `plt.plot()` automatically decided the scale of the plot.
- It also prints the **type of object** i.e. `matplotlib.lines.Line2D`

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

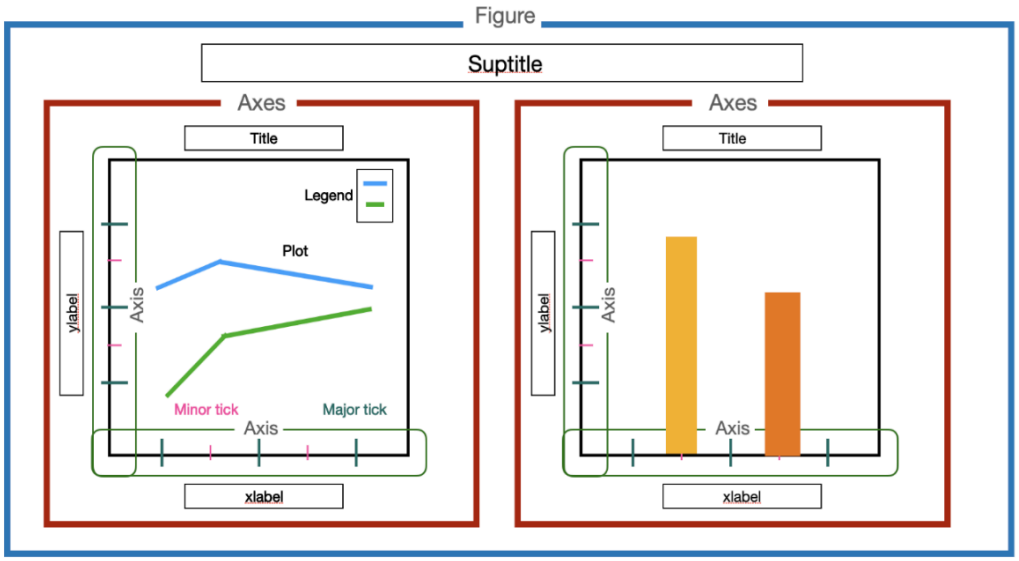

Title
Legend
Plot
ylabel
Axis
Minor tick
Major tick
Axis
xlabel


1 2 3 4
rating


1.2.4...
0
1K
2K
3K
4K


Title
ylabel
Axis
Axis
xlabel

Anatomy of Matplotlib



Components of a Matplotlib plot

- 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: You can add multiple **Axes** to the Figure, 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)**
- **Subtitle**: The common title of all the plots.
- **Legend**:
  - Describes the elements in the plot.
  - Blue and Green curves in this case.

These are the major components of a matplotlib plot.

How to choose the right plot?

Firstly, it 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
- 3+ Variables → Multivariate Analysis

What are the possible cases?

Univariate

- Numerical
- Categorical

Bivariate

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

Multivariate

Let's start with these and then we can generalize.

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

Univariate Data Visualization - Categorical Data

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

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

...and so on

How can we find the top-5 genres?

Code:

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

Output:

```
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
```

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.
- This is called a `Bar Chart` or a `Count Plot`.

Bar Chart

- We can draw a bar plot using `plt.bar()`.
- The data is binned here into categories.

Code:

```
x_bar=cat_counts.index
y_bar=cat_counts
plt.bar(x_bar, y_bar)
```

How can we handle overlapping labels?

1. Decrease the font size (not preferred)
2. Increase the figure size
3. Rotate the labels

How can we change the plot size?

Code:

```
plt.figure(figsize=(12,8))
plt.bar(x_bar, y_bar)
```

How can we rotate the tick labels, also increase the fontsize of the same?

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

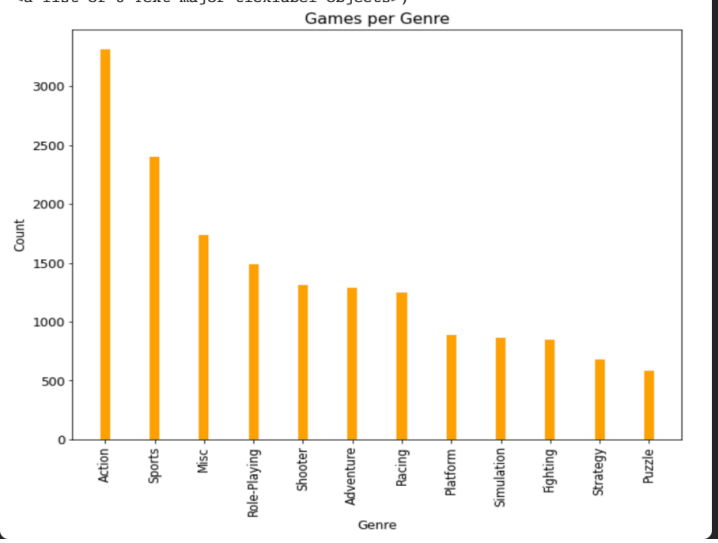
What about any adding some styling to the bars?

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

```
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)
```

Output:

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



How can we draw a bar chart in Seaborn?

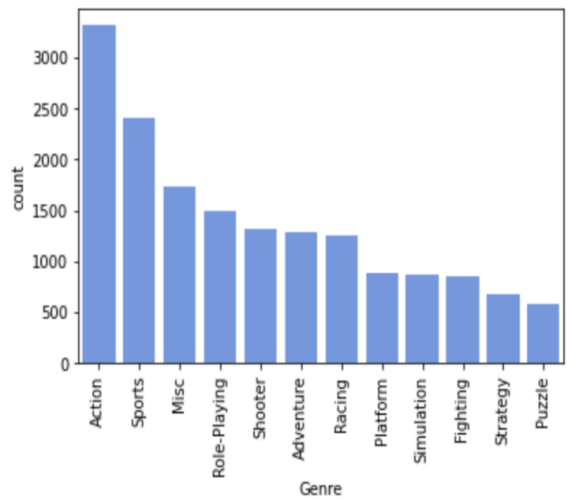
- In Seaborn, the same plot is called a **countplot**.
- It automatically does the counting of frequencies for you.

Code:

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

Output:

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



Pie Chart

What if instead of actual frequencues, we want see the proportion of the categories?

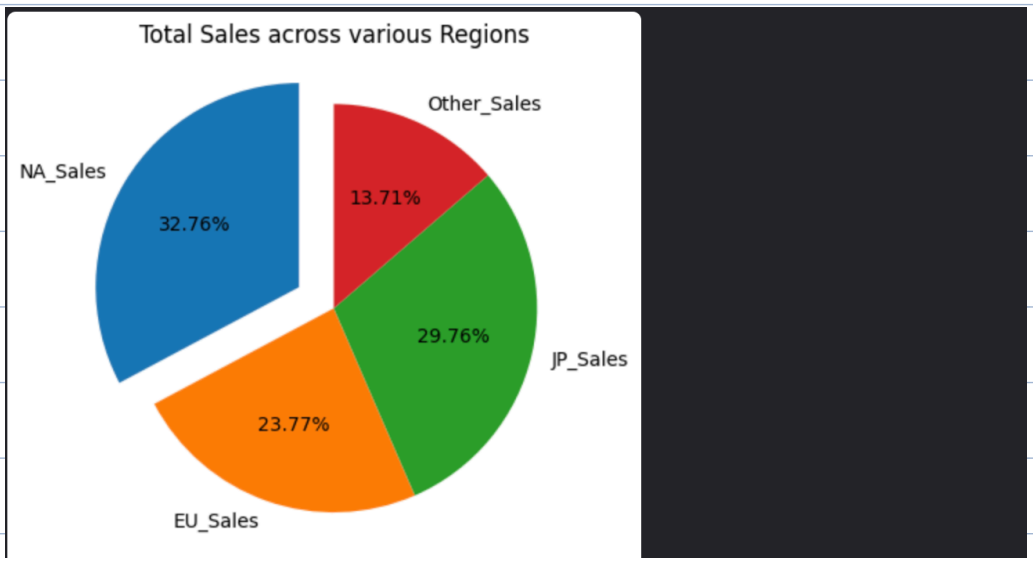
Say, we want to compare the distrubution/proportion of sales across different regions?

Which plot can we use for this? A pie-chart!

Code:

```
sales_data = data[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']]
region_sales = sales_data.T.sum(axis='columns')

plt.pie(region_sales,
        labels=region_sales.index,
        startangle=90,
        explode=(0.2,0,0,0),
        autopct = '%.2f%%') # label the wedges with their numeric value
plt.show()
```



Univariate Data Visualisation - Numerical Data

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

- How is the data distributed?
- Is the data skewed? Are there any outliers?
- How much percentage of data is below/above a certain number?
- Statistics like - Min, Max, Mean, Median, etc.

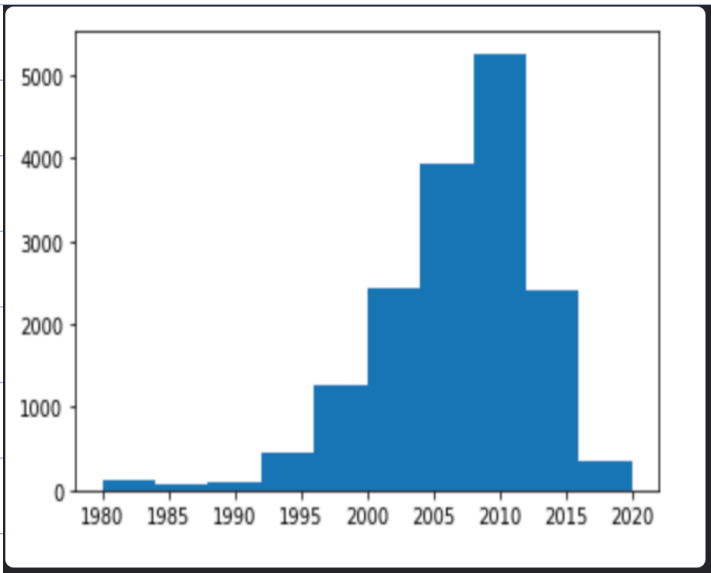
Now say you want to find the distribution of games released every year.

Unlike barplot, to see the distribution here we will have to `bin` the data.

Histogram

Code:

```
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 becoming highly popular in the last 1-2 decades.
- Maybe could point to increased usage of internet worldwide.

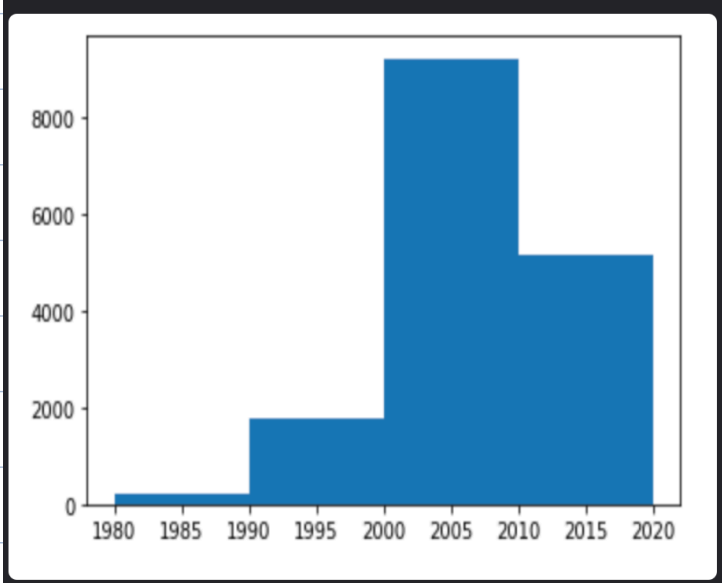
We can also vary the number of bins. **The default number of bins is 10**

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

Code:

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

> Output:

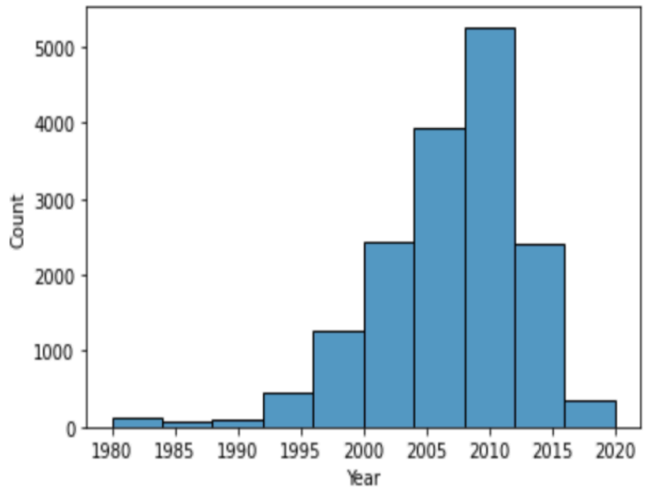


How can we plot a histogram in Seaborn?

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

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3cfa158cd0>
```



Notice that,

- 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**.

Why do we even need KDE plots?

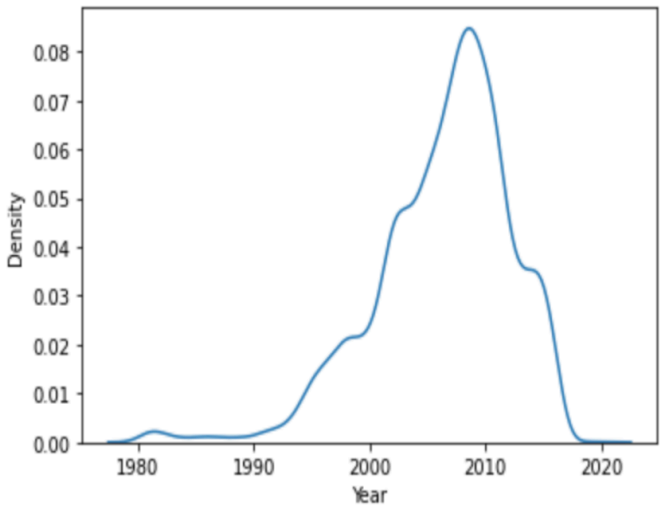
- Compared to histogram, KDE produces a plot which is **less cluttered** and **more interpretable**.
- Think of it as a **smoothened version** of a histogram.

Code:

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

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3cfa094e50>
```



Can you notice the difference between KDE plot and histogram?

The Y-axis has **probability density estimation** instead of count.

Boxplot

What if we want to find the aggregates like median, min, max and percentiles of the data.

Say I want the typical earnings of a game when it is published.

What kind of plot can we use here? Boxplot

What exactly is a Box plot?

- A box plot or **box and whiskers 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**.
- Except for points that are determined to be **"outliers"** using a method that is a function of the **inter-quartile range**.

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

1. Minimum Score

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

2. Lower Quartile

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

3. Median

- Median marks the **mid-point of the data**.
- It is shown by the **line that divides the box into two parts**.
- **Half the scores are greater than or equal to this value and half are less**.
- It is sometimes known as the **second quartile**.

4. Upper Quartile

- **75% of 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)

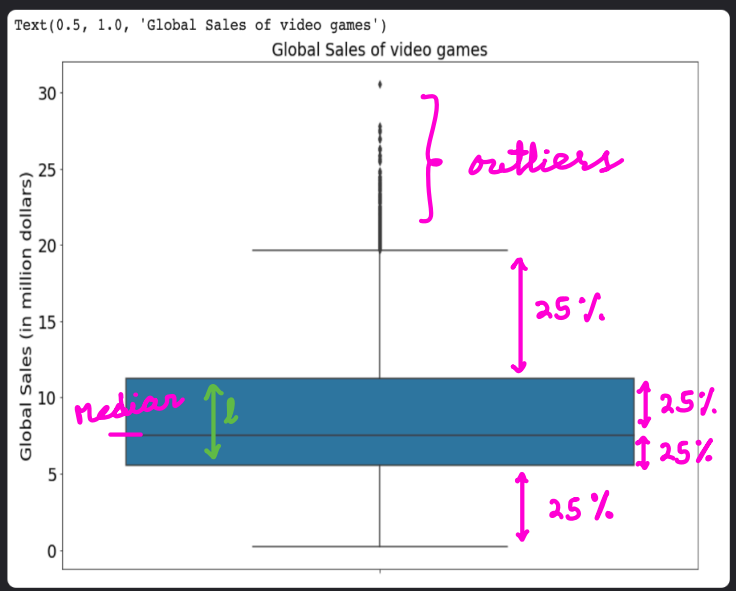
- This is the box plot showing the **middle 50% of scores**.
- It is the **range between the 25th and 75th percentile**.

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

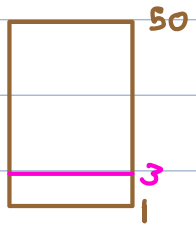
Code:

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

Output:



1 2 3 10 50



The 5 point estimates here are:

- Minimum, excluding outliers: 0
- Maximum, excluding outliers: 20 (in million dollars)
- 25th Quantile: 6 million
- Median: around 7 million
- 75th Quantile: 12 million

Note:

- The outliers always will appear either below the minimum or above the maximum.
- There are quite a few outliers above 20 million dollars, represented by black colored circles.