Data Visualization

- In the world of Analytics, the best way to get insights is by visualizing the data.
- We have already used Matplotlib, a 2D plotting library that allows us to plot different graphs and charts.
- Another complimentary package that is based on this data visualization library is Seaborn, which provides a high-level interface to draw statistical graphics.

Seaborn

CODING NINJAS

- Seaborn
  - Is a python data visualization library for statistical plotting
  - Is based on matplotlib (built on top of matplotlib)
  - Is designed to work with NumPy and pandas data structures
  - Provides a high-level interface for drawing attractive and informative statistical graphics.
  - Comes equipped with preset styles and color palettes so you can create complex, aesthetically pleasing charts with a few lines of code.

Seaborn

- Key Features
  - Seaborn is a statistical plotting library
  - It has beautiful default styles
  - It also is designed to work very well with Pandas dataframe objects.

Seaborn and Matplotlib

- Seaborn is built on top of Python's core visualization library matplotlib, but it's meant to serve as a complement, not a replacement.
- In most cases, we'll still use matplotlib for simple plotting
- On Seaborn's official website, they state:
  - "If matplotlib "tries to make easy things easy and hard things possible", seaborn tries to make a well-defined set of hard things easy too."

Seaborn and Matplotlib

- Seaborn helps resolve the two major problems faced by Matplotlib, the problems are −
  - Default Matplotlib parameters
  - Working with data frames

Seaborn Library

- import seaborn as sns
- sns.set()
- plt.show()

Datasets

- load_dataset()
- get_dataset_names()

Seaborn

- Data visualization using Seaborn
  - Visualizing statistical relationships
  - Plotting categorical data
  - Distribution of dataset
  - Visualizing Linear Relationships

Visualizing Statistical Relationships

- The process of understanding relationships between variables of a dataset and how these relationships, in turn, depend on other variables is known as statistical analysis

Visualizing Statistical Relationships

- relplot()
  - This is a figure-level-function that makes use of two other axes functions for Visualizing Statistical Relationships which are -
    - scatterplot()
    - lineplot()
  - By default it plots scatterplot()

Scatterplot

- Scatter plot -
  - A scatterplot is the most common example of visualizing relationships between two variables.
  - It depicts the joint distribution of two variables using a cloud of points, where each point represents an observation in the dataset.
  - This depiction allows the eye to infer a substantial amount of information about whether there is any meaningful relationship between them.

relplot() or scatterplot()

- Parameters -
  - x, y
  - data
  - hue
  - size
  - col
  - style

Lineplot

- Suitable when we want to understand change in one variable as a function of time or some other continuous variable.
- It depicts the relationship between continuous as well as categorical values in a continuous data point format.

Categorical Relationship

- Now we'll see the relationship between two variables of which one would be categorical.
- There are several ways to visualize a relationship involving categorical data in seaborn.
- catplot() - it gives a unified approach to plot different kind of categorical plots in seaborn

catplot() Function

- Categorical scatterplots:
  - stripplot() (with kind="strip"; the default)
  - swarmplot() (with kind="swarm")
- Categorical distribution plots:
  - boxplot() (with kind="box")
  - violinplot() (with kind="violin")
  - boxenplot() (with kind="boxen")
- Categorical estimate plots:
  - pointplot() (with kind="point")
  - barplot() (with kind="bar")
  - countplot() (with kind="count")

Categorical Relationship

- catplot() function is used to analyze the relationship between a numeric value and a categorical group of values together.
- Syntax:
  - seaborn.catplot(x=value, y=value, data=data)
- Categorical data is represented in x-axis and values correspond to them represented through y-axis (standard notation). We can reverse it.
- The default representation of the data in catplot() uses a scatterplot (stripplot).

catplot() Function

- Categorical scatterplots:
  - stripplot() (with kind="strip"; the default)
  - swarmplot() (with kind="swarm")

stripplot()

- A strip plot is a scatter plot where one of the variables is categorical.
- And for each category in the categorical variable, we see a scatter plot with respect to the numeric column.
- Parameters -
  - The first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset.
  - The jitter parameter controls the magnitude of jitter or disables it altogether. Jitter is the deviation from the true value.

swarmplot()

CODING NINJAS

- Similar to stripplot, only difference is that it does not allow overlapping of markers.
- It adjusts the points along the categorical axis using an algorithm that prevents them from overlapping.
- It can give a better representation of the distribution of observations.
- Although it only works well for relatively small datasets. Because sometimes it takes a lot of computation to arrange large data sets such that they are not overlapping.

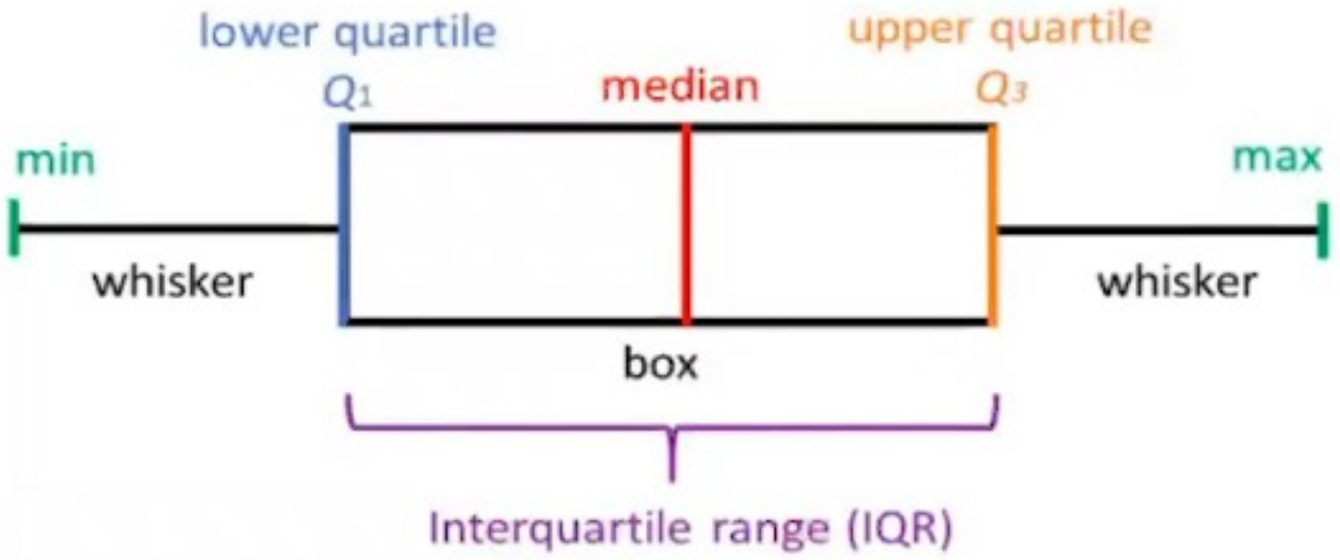
Distribution Plots

- As the size of the dataset grows, categorical scatter plots become limited in the information they can provide about the distribution of values within each category.
- In such cases we can use several approaches for summarizing the distributional information in ways that facilitate easy comparisons across the category levels.
- We use distribution plots to get a sense for how the variables are distributed.

catplot() Function

- Categorical distribution plots:
  - boxplot() (with kind="box")
  - boxenplot() (with kind="boxen")
  - violinplot() (with kind="violin")

Box Plot


lower quartile
Q_1
median
upper quartile
Q_3
min
max
whisker
whisker
box
Interquartile range (IQR)

Box Plot

- It is also named a box-and-whisker plot.
- Boxplots are used to visualize distributions. This is very useful when we want to compare data between two groups.
- Box Plot is the visual representation of the depicting groups of numerical data through their quartiles.
- Boxplot is also used for detect the outliers in data set.

Box Plot

- It captures the summary of the data efficiently with a simple box and whiskers and allows us to compare easily across groups.
- Boxplot summarizes a sample data using 25th, 50th and 75th percentiles. These percentiles are also known as the lower quartile, median and upper quartile.
- A box plot consist of 5 things.
  - Minimum
  - First Quartile or 25%
  - Median (Second Quartile) or 50%
  - Third Quartile or 75%
  - Maximum

Boxen Plot

- The Boxen plot is very similar to box plot, except for the fact that it plots different quartile values.
- By plotting different quartile values, we are able to understand the shape of the distribution particularly in the head end and tail end.

Violin Plot

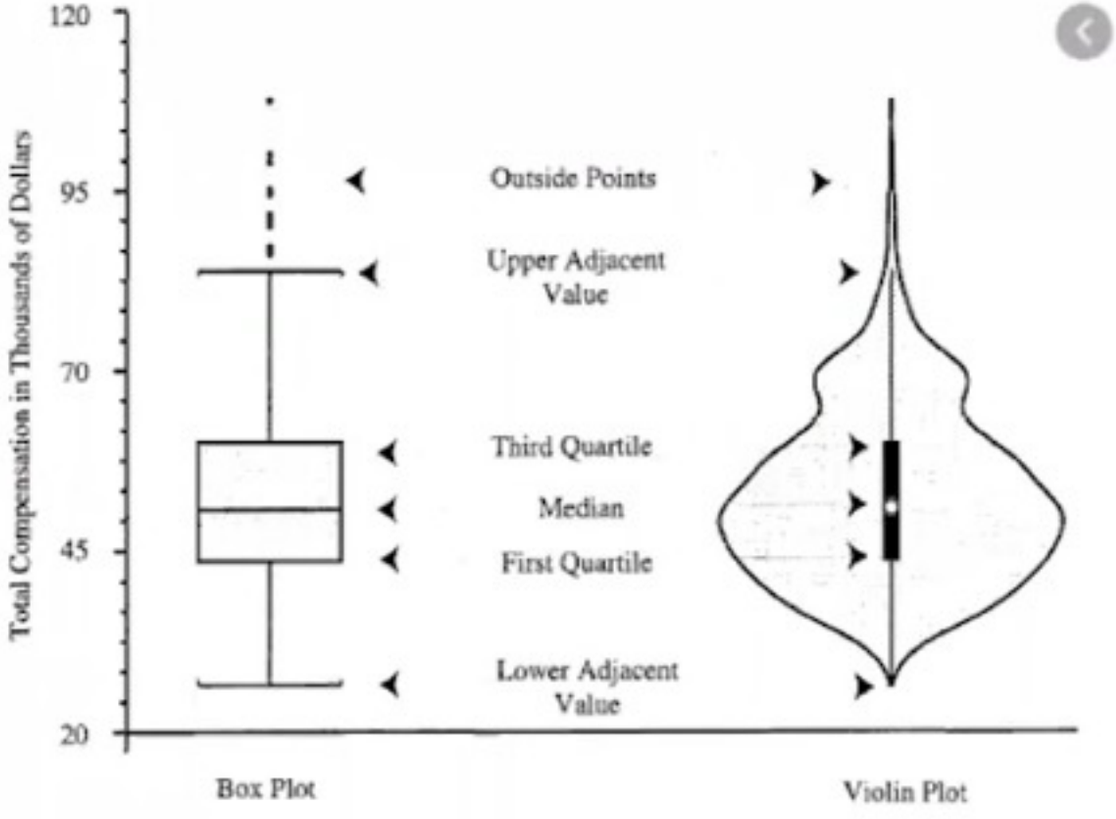


Image Source - https://miro.medium.com/max/1040/1*TTMQaNG1o4PgQd-e8LurMg.png

Violin Plot

- It shows the distribution of quantitative data across several levels of one (or more) categorical variables such that those distributions can be compared.
- It is really close from a boxplot, but allows a deeper understanding of the density.
- Unlike a box plot, in which all of the plot components correspond to actual data points, the violin plot features a kernel density estimation (KDE) of the underlying distribution.
- The Kernel Density Estimation is a mathematical process of finding an estimate probability density function of a random variable. The estimation attempts to infer characteristics of a population, based on a finite data set.

Violin Plot

- The density is mirrored and flipped over and the resulting shape is filled in, creating an image resembling a violin.
- The advantage of a violin plot is that it can show nuances in the distribution that aren't perceptible in a boxplot.
- On the other hand, the boxplot more clearly shows the outliers in the data.
- Violins are particularly adapted when the amount of data is huge and showing individual observations gets impossible.