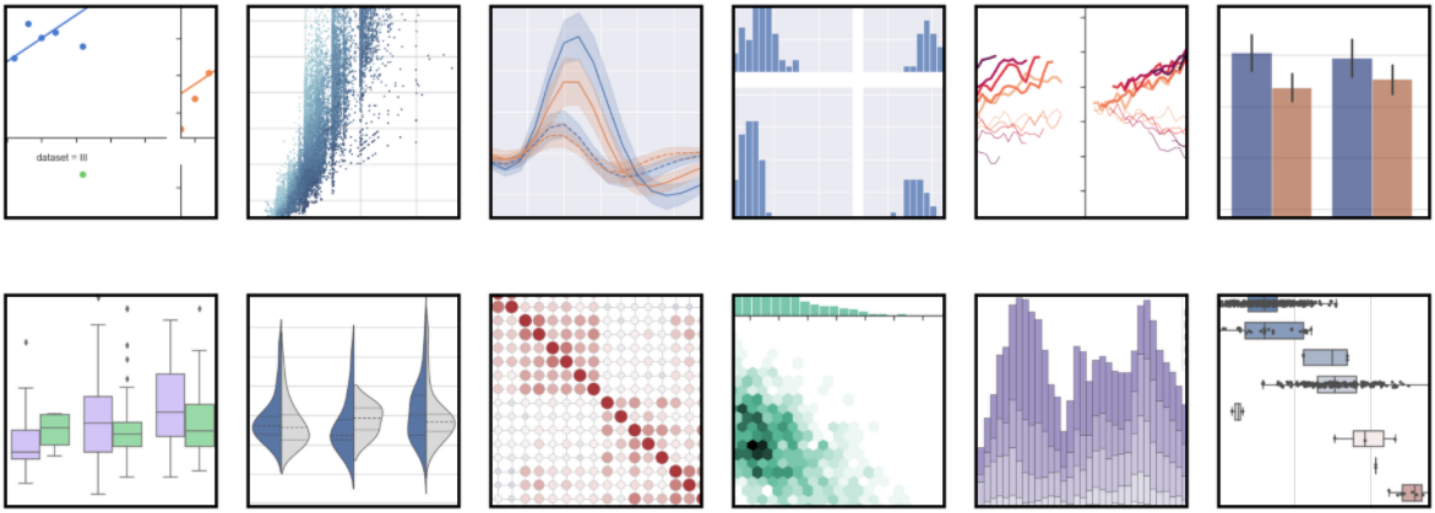
data science

A simple cheat sheet for Seaborn Data Visualization

Martin Høst Normark

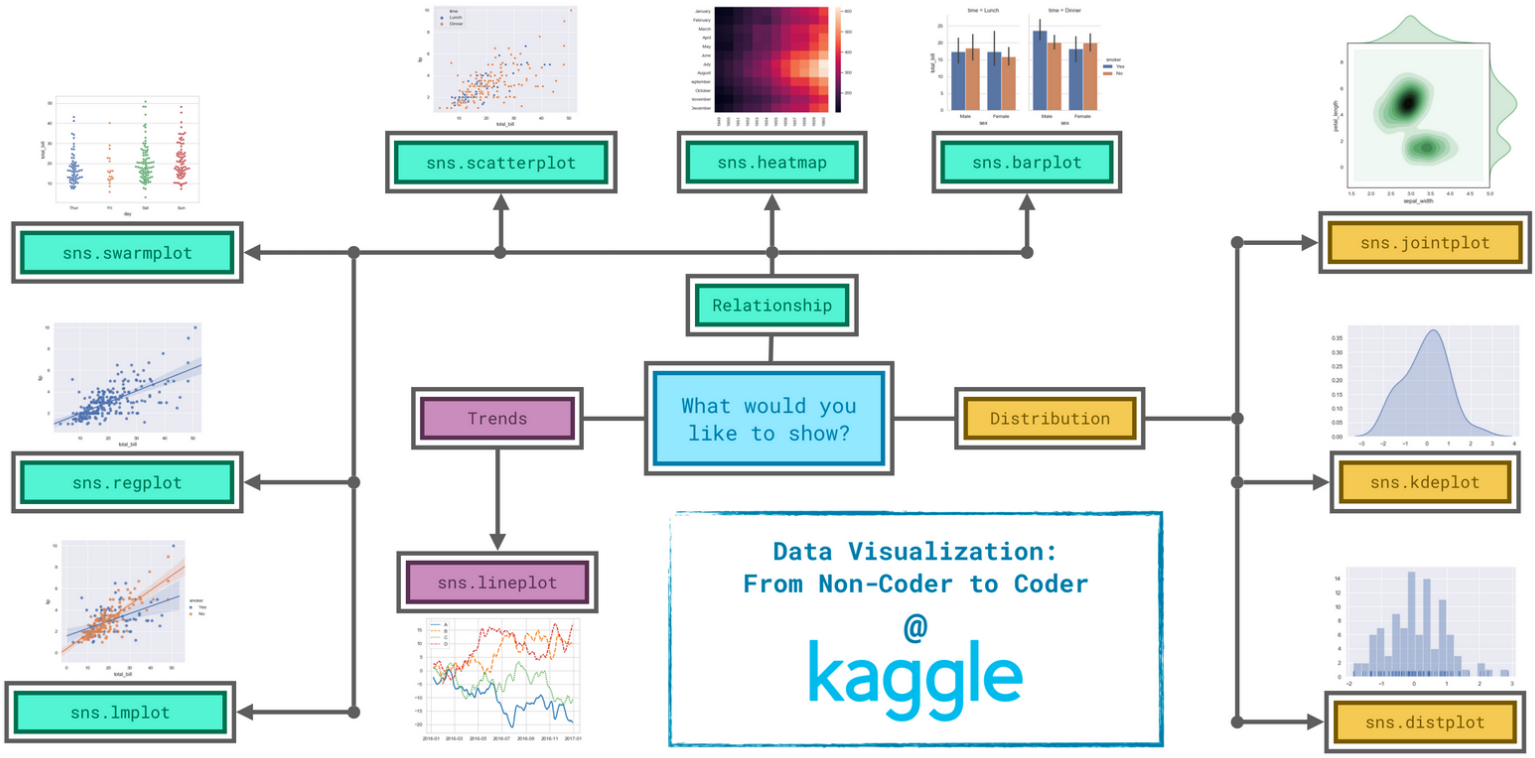
Oct 14, 2020 • 1 min read



Just gonna put this out here, courtesy of Kaggle's Data Visualization course.

It is a super simple description of the different plots you can do with Seaborn, simply divided into the type of story you're trying to tell.

There's even an awesome photo!



Trends - A trend is defined as a pattern of change.

- `sns.lineplot` - **Line charts** are best to show trends over a period of time, and multiple lines can be used to show trends in more than one group.

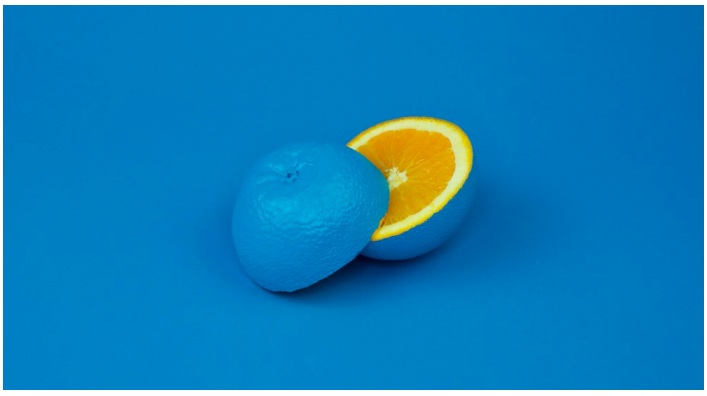
Relationship - There are many different chart types that you can use to understand relationships between variables in your data.

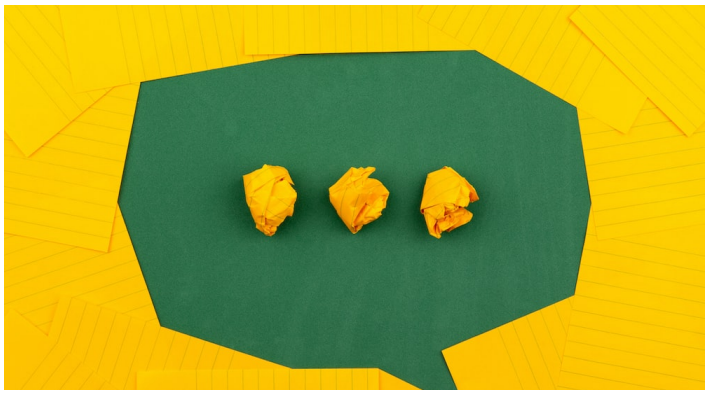
- `sns.barplot` - **Bar charts** are useful for comparing quantities corresponding to different groups.
- `sns.heatmap` - **Heatmaps** can be used to find color-coded patterns in tables of numbers.
- `sns.scatterplot` - **Scatter plots** show the relationship between two continuous variables; if color-coded, we can also show the relationship with a third categorical variable.
- `sns.regplot` - Including a **regression line** in the scatter plot makes it easier to see any linear relationship between two variables.

- `sns.lmplot` - This command is useful for drawing multiple regression lines, if the scatter plot contains multiple, color-coded groups.
- `sns.swarmplot` - **Categorical scatter plots** show the relationship between a continuous variable and a categorical variable.

Distribution - We visualize distributions to show the possible values that we can expect to see in a variable, along with how likely they are.

- `sns.distplot` - **Histograms** show the distribution of a single numerical variable.
- `sns.kdeplot` - **KDE plots** (or **2D KDE plots**) show an estimated, smooth distribution of a single numerical variable (or two numerical variables).
- `sns.jointplot` - This command is useful for simultaneously displaying a 2D KDE plot with the corresponding KDE plots for each individual variable.





Cheatography

Python - Seaborn Cheat Sheet

by DarioPittera (aggialavura) via cheatography.com/83764/cs/19851/

TO START

```
import seaborn as sns
# If working on a notebook
%matplotlib inline
```

DISTRIBUTION PLOTS

sns.distplot(df['col'])	distribution plot
- bin = x	number of bins
- kde = False	remove the line
sns.jointplot(x,y,df)	plot 2 variables
- kind = ''	kind of plot*
sns.pairplot(df)	plot all vars combin
- hue='categ var'	distinguish per var
- palette=''	set a color palette
sns.rugplot(df['col'])	idea of distribution
sns.kdeplot(df['col'])	kde plot

"kind=" E.g.: hex, reg, kde.

CATEGORICAL PLOTS

sns.barplot(x,y,df)	bar plot
- estimator="*	bar values
sns.countplot(x,df)	bars = count
sns.boxplot(x,y,,df)	box plot
- hue='categ var'	divide per var
- palette=''	set palette
- orient='h'	horiz. plot
sns.violinplot(x,y,df)	violin plot*
- hue='categ var'	divide per var
- palette=''	set palette
sns.stripplot(x,y,df)	bars = scatter
- jitter = True	add noise
- hue='categ var'	divide per var
- palette=''	set palette
- split = True	split by hue

CATEGORICAL PLOTS (cont)

sns.swarmplot(x,y,df)	swarm plot
- hue='categ var'	divide per var
- palette=''	set palette
- split = True	split by hue

** You can alo combine more plots by calling them one after each other.

sns.factorplot(x,y,df,kind)*
general categorical form of graph

TIP: when you call a plot function, press **"shift + tab"** to show the parameters needed.
estimator= can be, mean, std, or whatever function. It will display the bars or whatever you choose.
General form, kind=: e.g., point, bar, violin, etc.

ON CATEGORICAL PLOTS...

What is a violin plot?

> It has a similar role of a box and whisker plots. It shows the distribution of quantitative data across several levels of one (or more) categorical variables. The violin plot features a kernel density estimation of the underlying distribution.

What is a strip plot?

> It will draw a scatterplot where one variable is categorical. It is also a good complement to a box or violin plot in cases where you want to show all observations along with some representation of the underlying distribution.

What is a swarm plot

> It is similar to a stripplot(), but the points are adjusted (only along the categorical axis) so that they don't overlap. This gives a better representation of the distribution of values, although it does not scale as well to large numbers of observations.

MATRIX PLOTS

sns.heatmap(df.corr())*	heat map plot
- annot = True	add actual values
- cmap=''	set a color palette
- linecolor=''	set borders
- linewidths=x	set border width
sns.clustermap(matrix)	hierarc. clustering
- cmap=''	set a color palette
- standard_scale = 1	normalise data

Heat map plot needs a correlation matrix, or more generally, a matrix. You can use the **pivot_table(index,columns,values)** function to convert a dataframe.

GRIDS

sns.pairplot(df)	plot all vars combination
- hue='categ var'	divide per var
- palette=''	set palette
g = **sns.PairGrid(df)**	set (empty) axis of pairplot
-g.map(plt.scatter)	populate axis with some plot
-g.map_diag(plt.hist)	set diag plots
-g.map_upper(plt.scatter)	set upper plots
-g.map_lower(sns.kdeplot)	set lower plots
g = sns.FacetGrid(df,c,r)	empty axis
-g = g.map(**plt**.hist, "c")	populate axis histograms
-..g.map(sns.distplot, "c")	populate axis with distplots

now some more complex stuff

By **DarioPittera** (aggialavura)

cheatography.com/aggialavura/
www.dariopittera.com

Not published yet.
Last updated 15th June, 2019.

Cheatography

Python - Seaborn Cheat Sheet

by DarioPittera (aggialavura) via cheatography.com/83764/cs/19851/

GRIDS (cont)

- hue='categ var'	divide per var

-g = g.map(plt.scatter, "c", "c").add_legend()

g = sns.JointGrid(x,y,df)
general form of jointplot()

g = g.plot(sns.regplot, sns.distplot)
join two plots

REGRESSION PLOTS

sns.lmplot(x,y,df)	creat reg plot
- hue='categ var'	divide per var
- palette=''	set palette
- markers='' *	set mark shape
- scatter_kws='dict' *	set marker size
sns.lmplot(x,y,df,**col**)	create a grid plot
sns.lmplot(x,y,df,**row,col**)	X*X grid
sns.lmplot(x,y,df,**row,col,hue**)	X*X*X grid
- aspect = x	choose ratio
- size = x	set size

markers='': e.g., o,v,etc.
scatter_kws='' e.g.: {'s':100}, it is a call to matplotlib. It will be hard to remember how to use these special cases, so no worries, you will have a look online.

STYLE and COLOR

sns.set_style('dar-kgrid')	apply darkgrid style
sns.set_style('ticks')	apply ticks style
sns.despine()	remove borders
sns.despine(lef-t=True)	remove left border
plt.figure(figsize=(x,x))	choose fig size
sns.set_context('talk') *	set context
sns.set_context (font_scale)	set font size

.set_context(''): e.g.: paper, poster, talk, notebook, etc.

By **DarioPittera** (aggialavura)

cheatography.com/aggialavura/
www.dariopittera.com

Not published yet.
Last updated 15th June, 2019.

An Ultimate Cheat Sheet for Data Visualization Techniques in Seaborn

Vildansarikaya · Follow

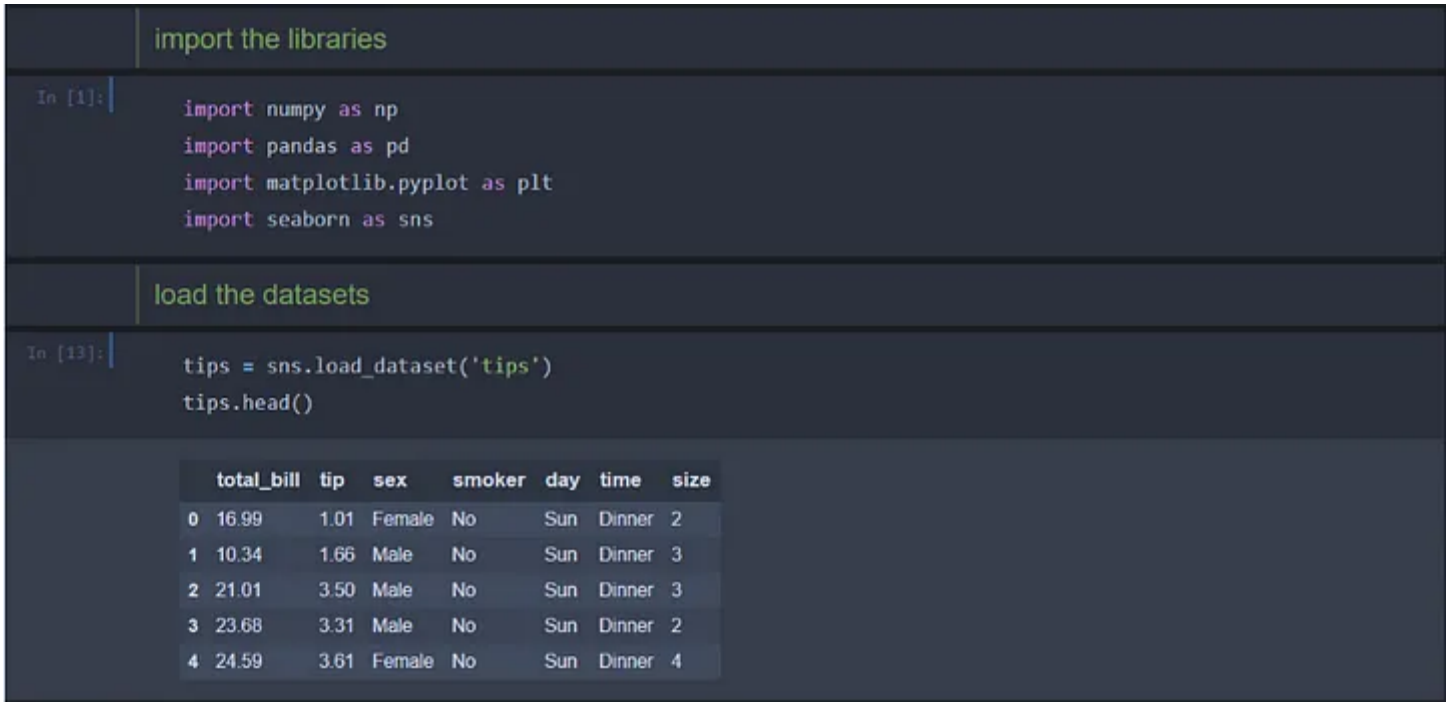
Published in Clarusway · 5 min read · Dec 26, 2020

Data visualization signs to the techniques used to information by encoding it as visual objects such as bars, points and lines in graphics. The goal is to communicate data and information to users in a clear and efficient manner. Data visualization is helpful to understand the structure of the data, determine the outlier, see the trends or patterns of the data, evaluate the results, and communicate these findings effectively to audience. This is one of the steps in data analysis, and this step is called with Exploratory Data Analysis (EDA).

Python provides many visualization packages for EDA. One of them is Seaborn library. Seaborn library makes statistical graphics in python, and it helps users explore and understand the data with graphics.

Here will be explained Seaborn plot types, how to plot with real data, when to use them, and finally provide a brand-new, broadly explained cheat sheet.

Let's start with importing required libraries and loading the dataset which contains some built-in datasets in seaborn. The "tips" dataset, one of built-in datasets, will be used.

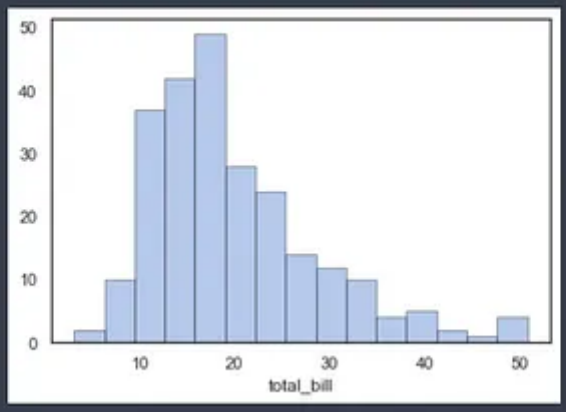


	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

1.DISTRIBUTION PLOT

A)Distplot: A distplot is used a univariate distribution of observation.

```
In [22]: sns.distplot(tips['total_bill'], hist_kws=dict(edgecolor="k", linewidth=1), bins=15, kde=False);
```



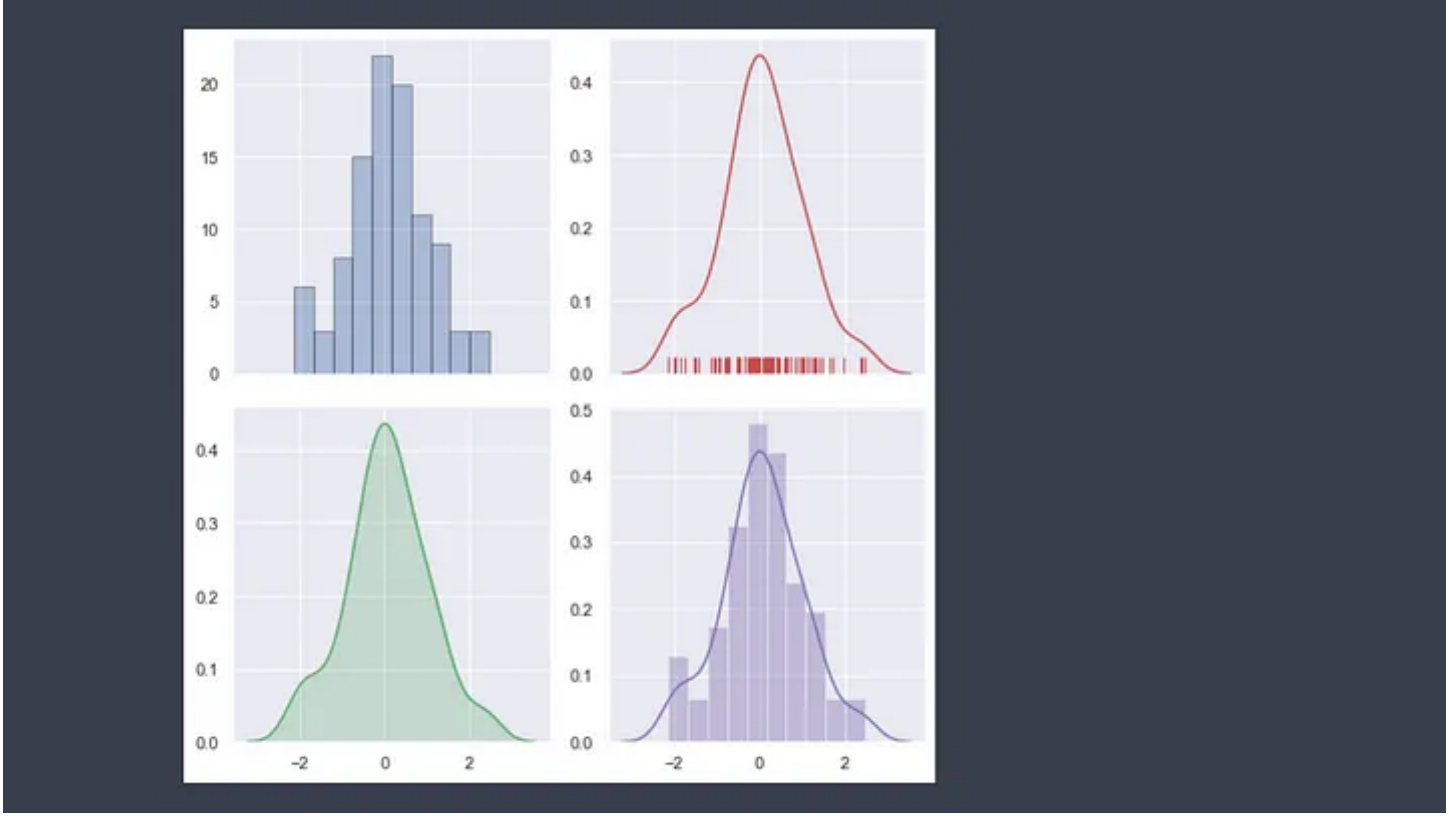
tips['total_bill']: It is a numeric column in tips dataset.

hist_kws: The keyword argument to change histogram format.

bins: It is used to set the number of bins, and it depends on your datasets.

kde: Kernel Density Estimate.

```
In [42]: sns.set(style="darkgrid")

         # Set up the matplotlib figure
         f, axes = plt.subplots(2, 2, figsize=(7, 7), sharex=True)
         sns.despine()

         # Generate a random univariate dataset
         rs = np.random.RandomState(10)
         d = rs.normal(size=100)

         # Plot a simple histogram with binsize determined automatically
         sns.distplot(d, kde=False, hist_kws=dict(edgecolor="k", linewidth=1), color="b", ax=axes[0, 0])

         # Plot a kernel density estimate and rug plot
         sns.distplot(d, hist=False, rug=True, color="r", ax=axes[0, 1])

         # Plot a filled kernel density estimate
         sns.distplot(d, hist=False, color="g", kde_kws={"shade": True}, ax=axes[1, 0])

         # Plot a histogram and kernel density estimate
         sns.distplot(d, color="m", ax=axes[1, 1])

         plt.tight_layout()
```



2.CATEGORICAL PLOT

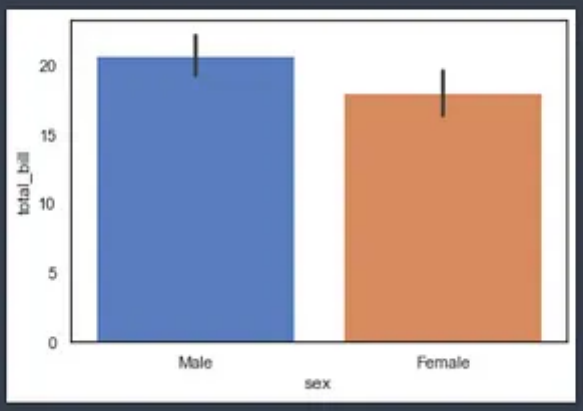
A categorical variable has two or more categories, nonnumeric values. For instance, “sex” column is a categorical variable column, having two categories (male and female) at “tips” dataset. “day” column is a categorical variable column, having a number of categories (thursday, friday, saturday, sunday, etc.) at “tips” dataset. Categorical plot can be used to visualize categorical variables.

A)Bar Plot: Bar plot is useful for displaying relationships between categorical variable and numerical variable. Bar plot has two methods. Dataset’s columns can be used or dataset’s columns assigned to x and y parameters are used.

Method 1

method 1

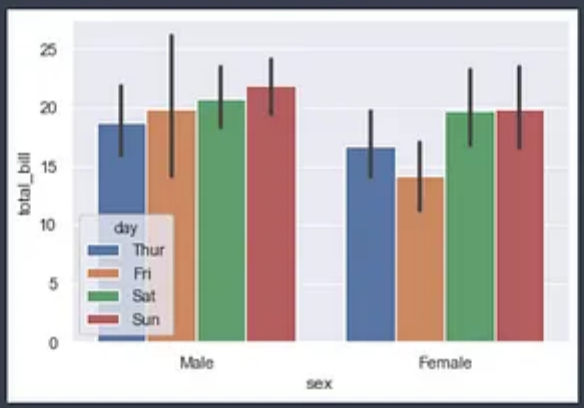
```
In [15]: sns.barplot(tips["sex"], tips["total_bill"]);
```



Method 2

method 2

```
In [6]: sns.barplot(x="sex", y="total_bill", hue="day", data=tips);
```



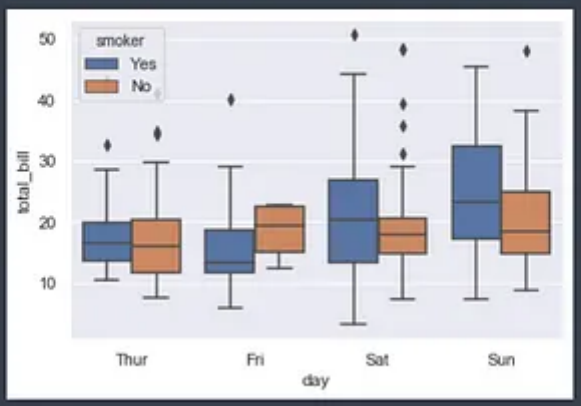
hue: determines which column in the data frame should be used for colour encoding.

B)Count Plot: Count plot shows the counts of observations in each categorical variable.

```
In [27]: sns.countplot(x='day', order=tips.day.value_counts().index,  data=tips);
```

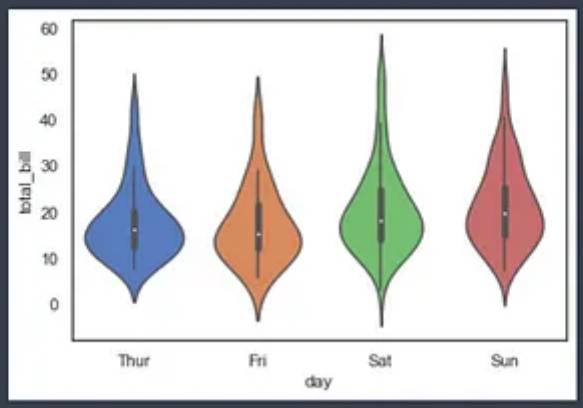
C)Box Plot: Box plot visualizes summarizing numeric data over a set of categorical variables, and it provides some information about data. This information consists of minimum score, maximum score, first quartile (25% of data), second quartile-median (50% of data), and third quartile (75% of data).

```
In [8]: sns.boxplot(x="day", y="total_bill", hue="smoker", data=tips);
```



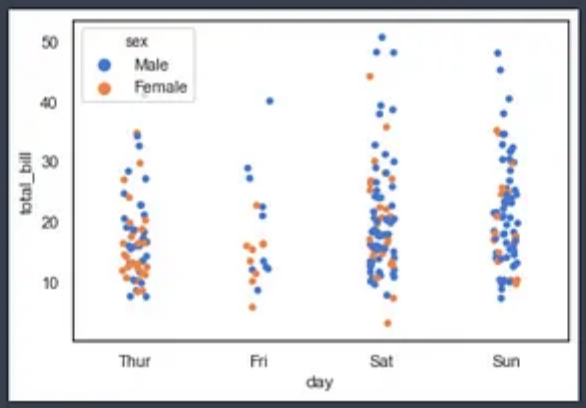
D)Violin Plot: Violin plot is a similar role as a box plot.

```
In [17]: sns.violinplot(x="day", y="total_bill", data=tips);
```



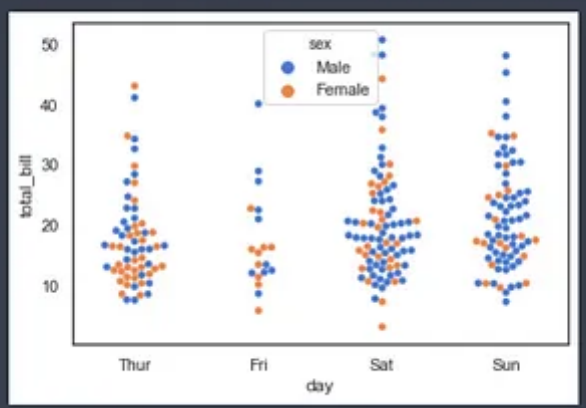
E)Strip Plot: Strip plot is a scatter plot. It shows the relationship between two variables.

```
In [18]: sns.stripplot(x="day", y="total_bill", hue="sex", data=tips);
```



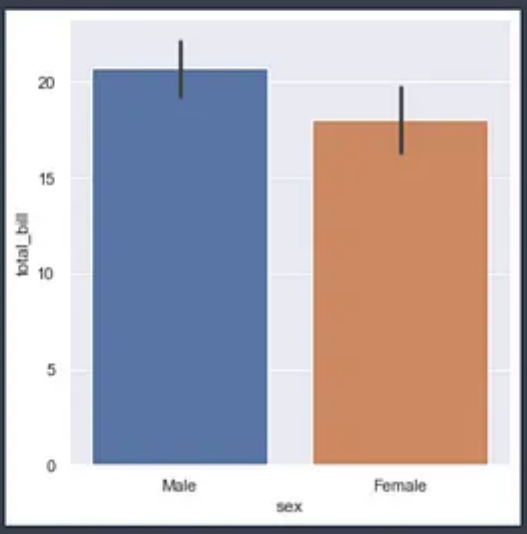
F)Swarm Plot(Violin Plot + Strip Plot): As the name suggests, it consists of the combination of a violin plot and a strip plot.

```
In [19]: sns.swarmplot(x="day", y="total_bill", hue="sex", data=tips);
```



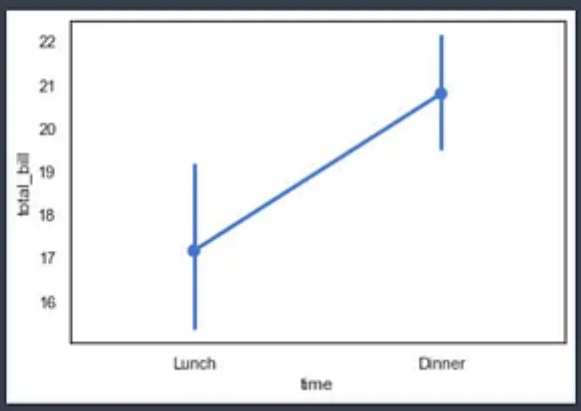
G)Cat Plot(Former Name: Factor Plot): This plot provides the relationship between a numerical and one or more categorical variables with several visual options.

```
In [46]: sns.catplot(x='sex',y='total_bill', data=tips, kind='bar')
         # kind options: bar, swarm, strip(default), box, violin, point and count.

<seaborn.axisgrid.FacetGrid at 0x17c982e7cc8>
```



H)Point Plot: Point plot shows an estimate of central tendency(center of the data distribution) for a numerical variable.

```
In [21]: sns.pointplot(x="time", y="total_bill", data=tips)

<matplotlib.axes._subplots.AxesSubplot at 0x17c9ca62648>
```



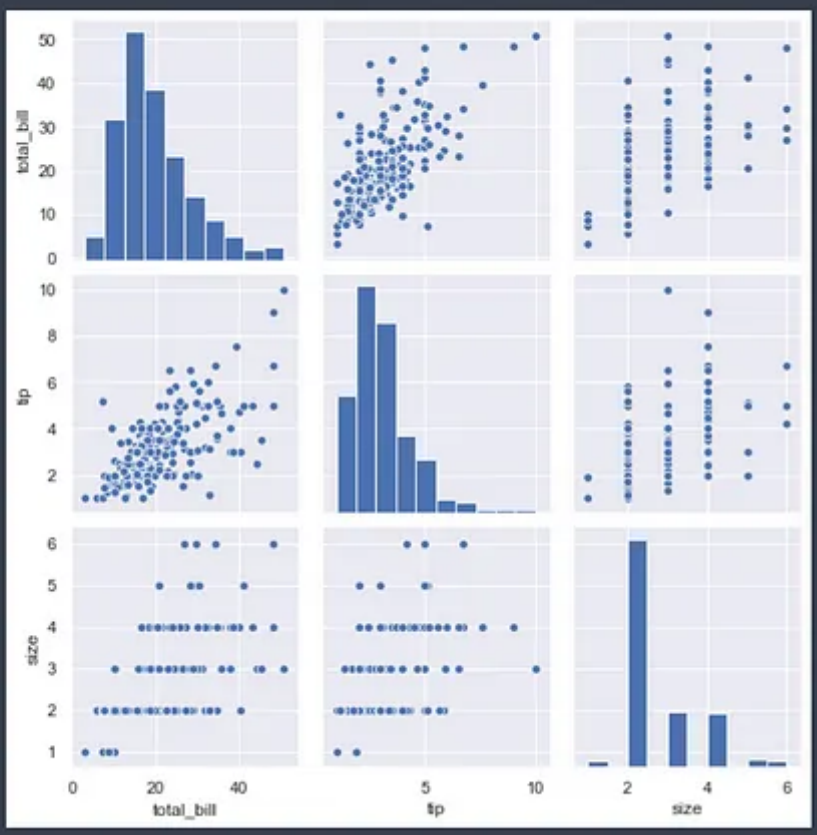
3.MATRIX AND GRID PLOT

A)Heat Map: The heat map shows which variables are correlated to each other. It helps to understand correlation, and it provides a colorful visualization about correlation.

```
In [28]: sns.heatmap(tips.corr(), cmap="coolwarm", annot=True);
```

B)Pair Plot: The pair plot shows pairwise relationships in a dataset.

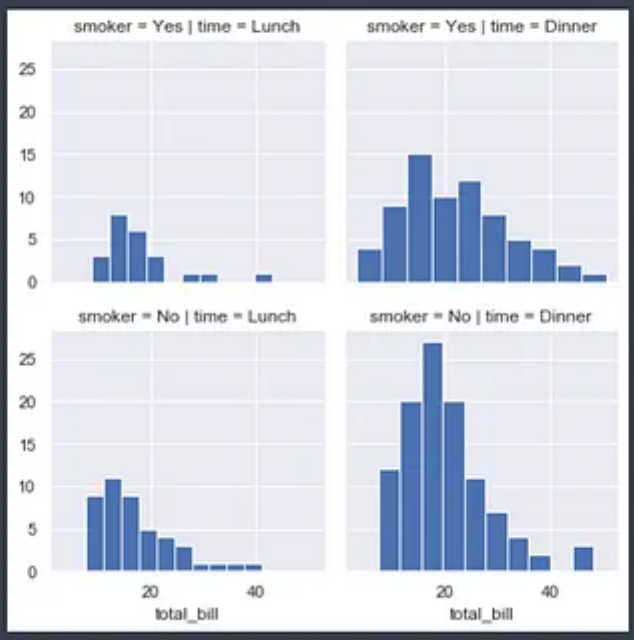
```
In [45]: sns.pairplot(tips);
```



C)FacetGrid: FacetGrid helps to understand distribution of one variable as well as the relationship between multiple variables with using separate subsets.

```
In [17]: g = sns.FacetGrid(tips, col="time", row="smoker")
         g.map(plt.hist, "total_bill")
```

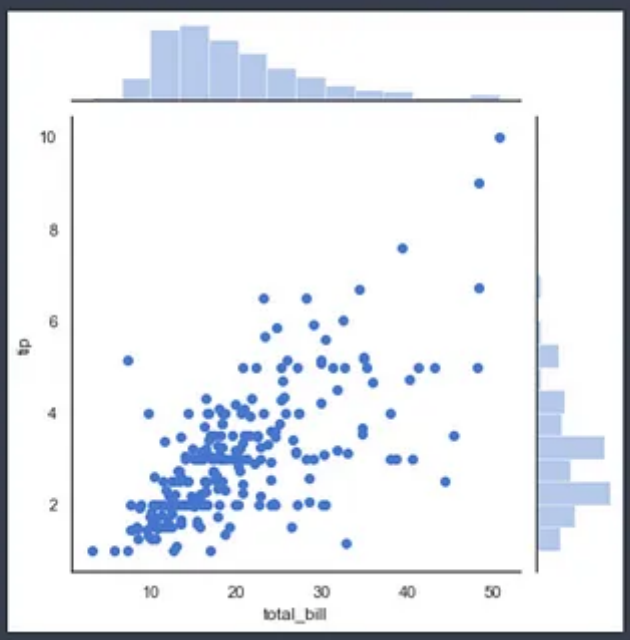
```
<seaborn.axisgrid.FacetGrid at 0x259d7b55f08>
```



4)JOINTPLOT

Jointplot visualizes two variables with bivariate and univariate graphs.

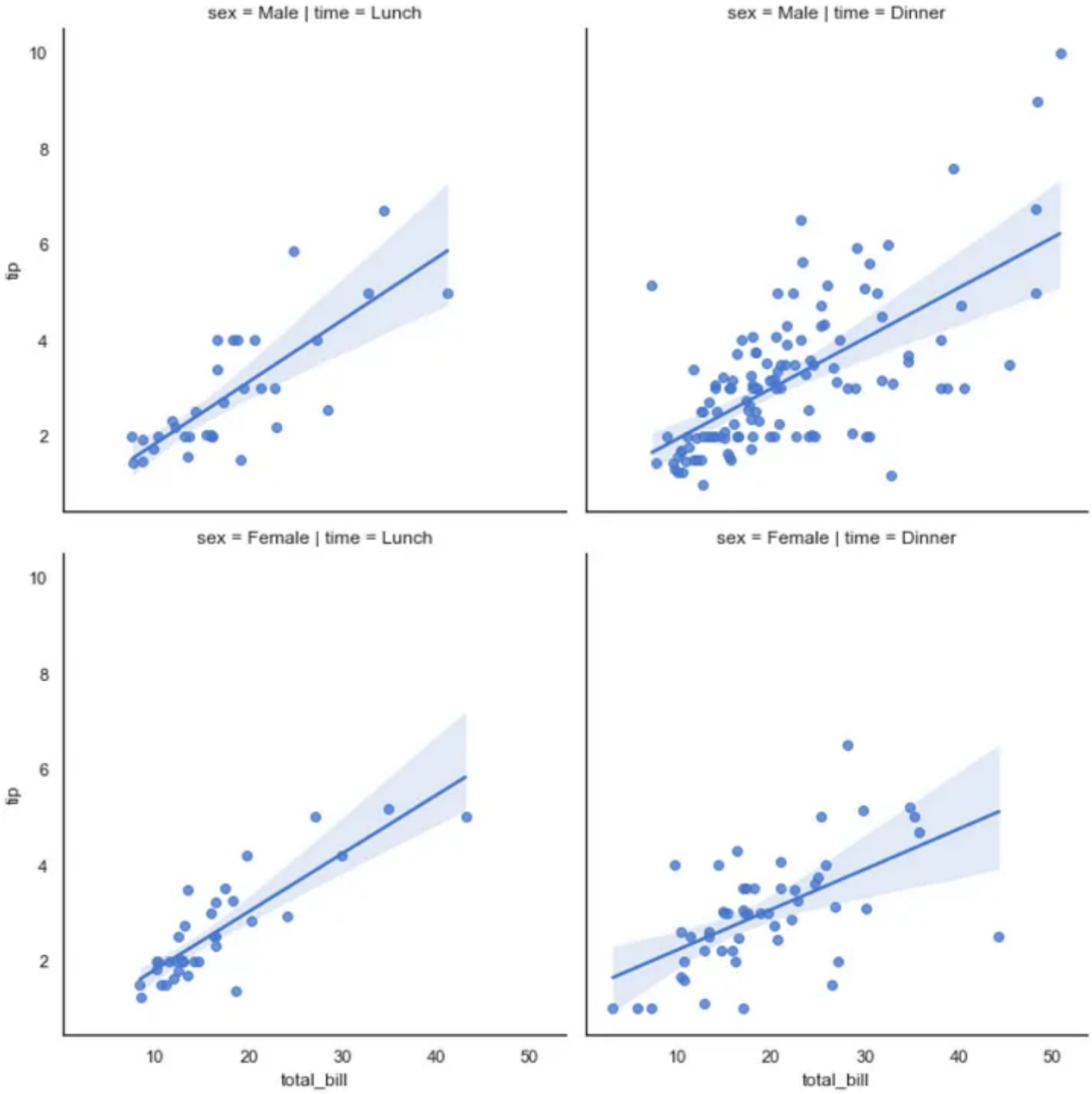
```
In [37]: sns.jointplot(x="total_bill", y="tip", kind="scatter", data=tips);
         # kind options =scatter (default), reg, resid, kde, hex.
```



5)LMPLOT

Lmplot can be used for drawing a scatter plot onto a FacetGrid.

```
In [19]: sns.lmplot(x="total_bill", ci=95, y="tip", data=tips, row="sex", col="time", aspect=1, height=5)
```



To sum up:

- Seaborn is a Python data visualization library built on Matplotlib.
- It provides a high-level interface for drawing attractive and informative statistical graphics.
- Seaborn is helpful to understand structure of the data, determine the outliers, and understand many things through visualization.

- Seaborn has powerful data visualization capabilities for EDA. All charts mentioned in the article can be used for EDA.
- This cheat sheet is a useful reference where you can find all the examples in the article and easily access them whenever you want.

You can download the **cheat sheet** containing all the codes from **here.**

Cheatsheet | Data Visualization | Seaborn | Data Science | Data Analysis

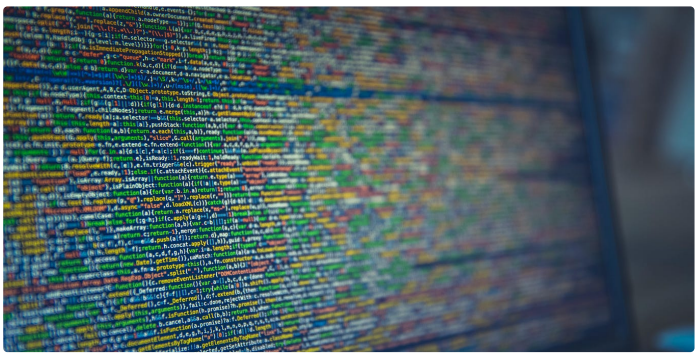
Written by Vildansarikaya

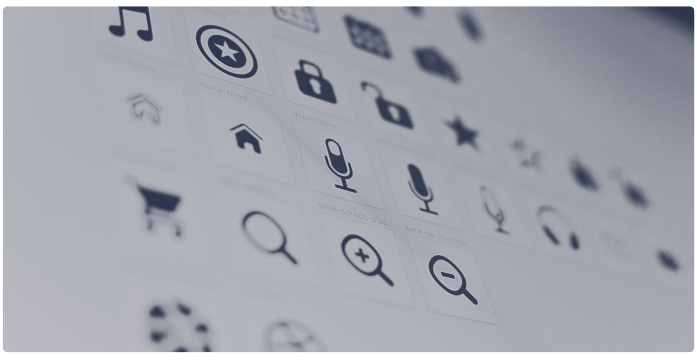
66 Followers · Writer for Clarusway

Data Scientist, Industrial Engineer

Follow

More from Vildansarikaya and Clarusway





Vildansarikaya in Clarusway

TYPES OF RELATIONSHIPS IN DATA MODELLING

Start by learning the meaning of the relational database. A relational database is a type of...

4 min read · Nov 24, 2020

545 8

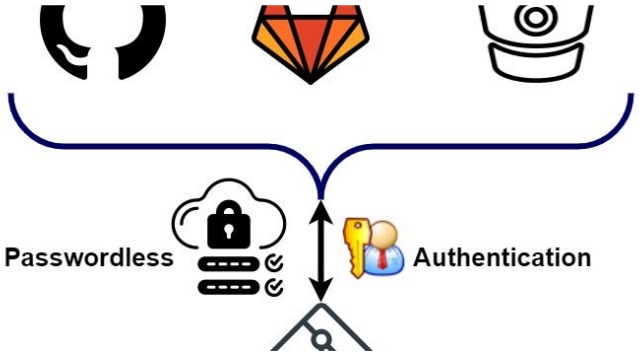
Semih Durmus in Clarusway

Setting up React Native Vector Icons for IOS

Icons tell more than a plenty of words do. This is one of the reasons that icons are widely...

4 min read · Dec 20, 2020

1.3K 9



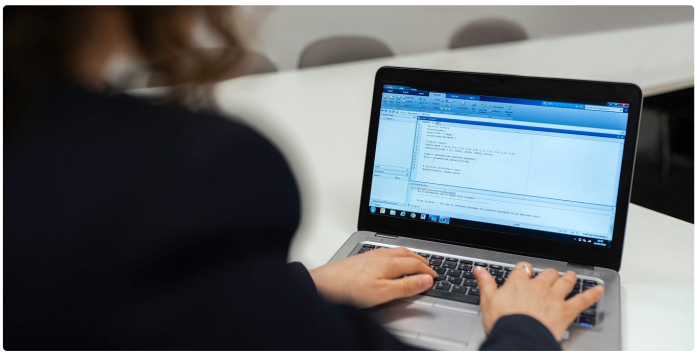
Joe Blue in Clarusway

How to Use Git/GitHub without asking for authentication always:...

The authentication to access the source code repositories like GitHub, GitLab, BitBucket,...

7 min read · Dec 30, 2020

944 9



Alex Duncan in Clarusway

Installing and Upgrading Ansible with *pip

You can install Ansible on many systems with pip. Pip is the main Python package manager.

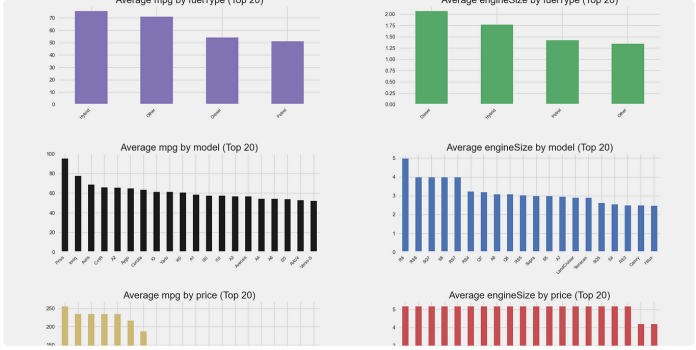
3 min read · Jul 4, 2021

435

See all from Vildansarikaya

See all from Clarusway

Recommended from Medium



Mutiara Tambunan

Automated EDA and Creating a Simple Interactive Dashboard in...

In today's data-driven world, the ability to analyze and visualize data is fundamental for...

6 min read · Sep 10

263

Robert Daly in Python in Plain English

Preprocessing Data for Linear Regwith Scikit-Learn

This week I learned how to create a Linear Regression model. I think it's so cool that...

5 min read · Aug 15

20

Lists

Predictive Modeling w/ Python

20 stories · 659 saves

New_Reading_List

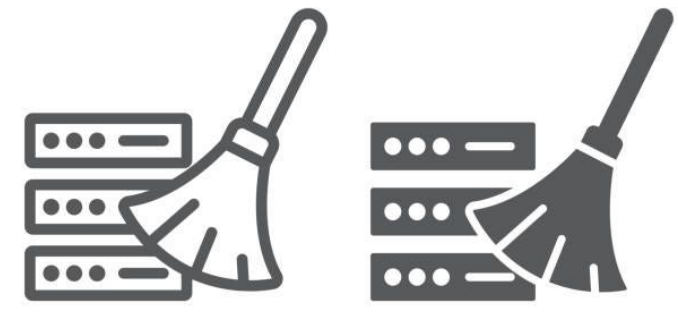
174 stories · 209 saves

Practical Guides to Machine Learning

10 stories · 739 saves

ChatGPT prompts

30 stories · 747 saves



Nikhil Malkari

Manoj Das

Mastering Data Cleaning Techniques: Handling Missing...

INTRODUCTION

9 min read · Jun 21

2 1

Discover DataPrep: Make Exploratory Data Analysis Easier i...

What is DataPrep package in Python? How to do EDA using DataPrep? Example of EDA wit...

7 min read · Jul 7

3 1

Mochamad Kautzar Ichramsyah in CodeX

Automate the exploratory data analysis (EDA) to understand the...

What is EDA?

11 min read · Jul 12

1.97K 23

Bradley Stephen Shaw in Towards Data Science

Make Your Tables Look Glorious

Simple formatting tricks to make your pandas DataFrames presentation-ready

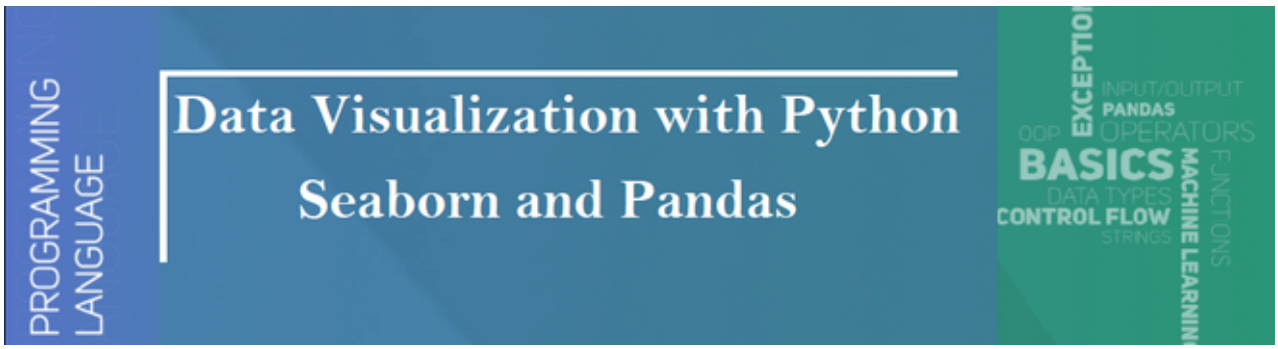
13 min read · Jan 11

440 8

See more recommendations

Data Visualization with Python Seaborn

Data Visualization is the presentation of data in pictorial format. It is extremely important for Data Analysis, primarily because of the fantastic ecosystem of data-centric Python packages. And it helps to understand the data, however, complex it is, the significance of data by summarizing and presenting a huge amount of data in a simple and easy-to-understand format and helps communicate information clearly and effectively.



Pandas and Seaborn is one of those packages and makes importing and analyzing data much easier. In this article, we will use Pandas and Seaborn to analyze data.

Pandas

Pandas offer tools for cleaning and process your data. It is the most popular Python library that is used for data analysis. In pandas, a data table is called a dataframe.

So, let's start with creating Pandas data frame:

Example 1:

Python3

```
# Python code demonstrate creating

import pandas as pd

# initialise data of lists.
data = {'Name':[ 'Mohe' , 'Karnal' , 'Yrik' , 'jack' ],
        'Age':[ 30 , 21 , 29 , 28 ]}

# Create DataFrame
df = pd.DataFrame( data )

# Print the output.
df
```

Output:

	Name	Age
0	Mohe	30
1	Karnal	21
2	Yrik	29
3	jack	28

Example 2: load the CSV data from the system and display it through pandas.

Python3

```
# import module
import pandas

# load the csv
data = pandas.read_csv("nba.csv")

# show first 5 column
data.head()
```

Output:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

Seaborn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It is built on the top of matplotlib library and also closely integrated into the data structures from pandas.

Installation

For python environment :

```
pip install seaborn
```

For conda environment :

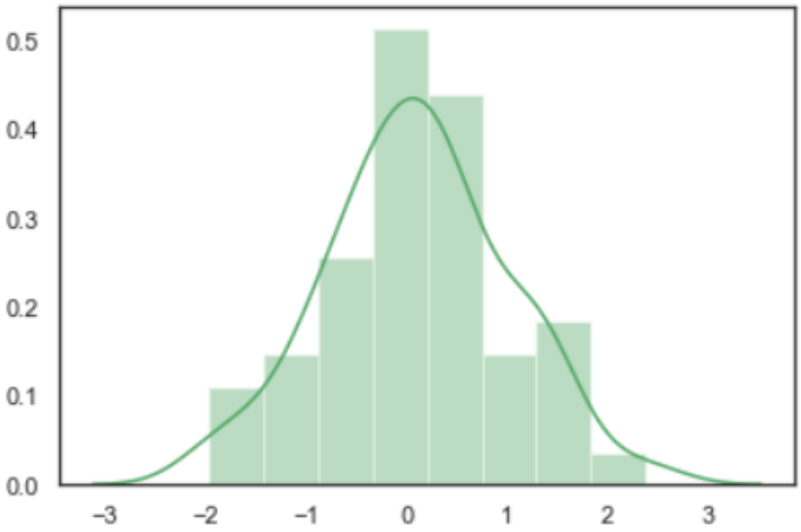
```
conda install seaborn
```

Let's create Some basic plots using seaborn:

Python3

```
# Importing libraries
import numpy as np
import seaborn as sns


# Selecting style as white,
# dark, whitegrid, darkgrid
# or ticks
sns.set( style = "white" )

# Generate a random univariate
# dataset
rs = np.random.RandomState( 10 )
d = rs.normal( size = 50 )

# Plot a simple histogram and kde
# with binsize determined automatically
sns.distplot(d, kde = True, color = "g")
```

Output:



Seaborn: statistical data visualization

Seaborn helps to visualize the statistical relationships, To understand how variables in a dataset are related to one another and how that relationship is dependent on other variables, we perform statistical analysis. This Statistical analysis helps to visualize the trends and identify various patterns in the dataset.

These are the plot will help to visualize:

- Line Plot
- Scatter Plot
- Box plot
- Point plot
- Count plot
- Violin plot

- Swarm plot
- Bar plot
- KDE Plot

Line plot:

Lineplot Is the most popular plot to draw a relationship between x and y with the possibility of several semantic groupings.

> ***Syntax :*** *sns.lineplot(x=None, y=None)*
>
> ***Parameters:***
>
> ***x, y:*** *Input data variables; must be numeric. Can pass data directly or reference columns in data.*

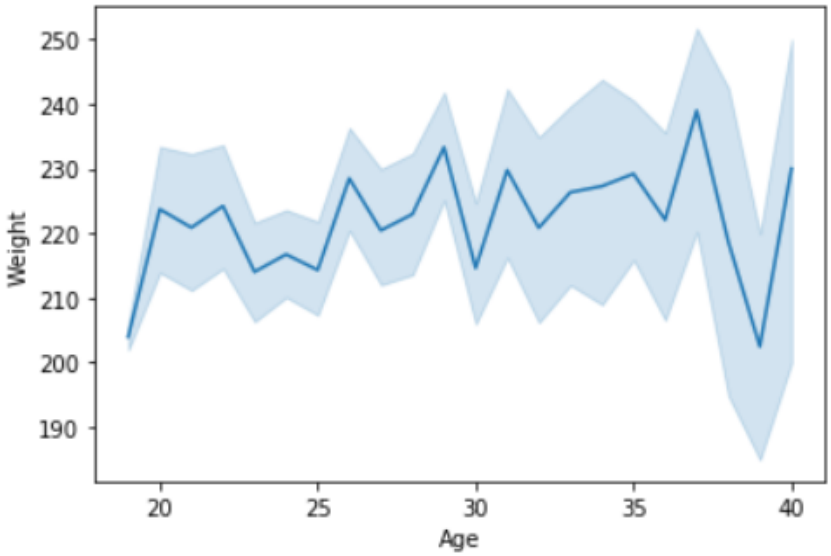
Let's visualize the data with a line plot and pandas:

Example 1:

Python3

```
# import module
import seaborn as sns
import pandas

# loading csv
data = pandas.read_csv("nba.csv")

# plotting lineplot
sns.lineplot( data['Age'], data['Weight'])
```

Output:

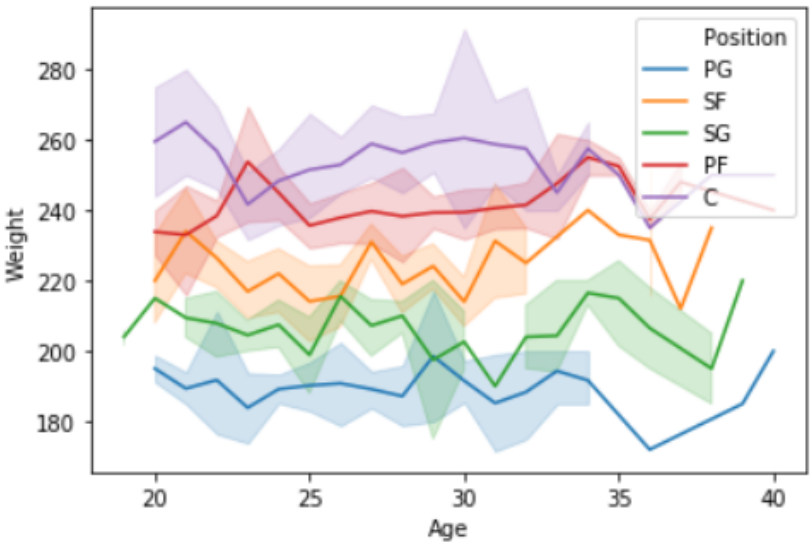


Example 2: Use the hue parameter for plotting the graph.

Python3

```
# import module
import seaborn as sns
import pandas

# read the csv data
data = pandas.read_csv("nba.csv")

# plot
sns.lineplot(data['Age'],data['Weight'], hue =data["Position"])
```

Output:



Scatter Plot:

Scatterplot Can be used with several semantic groupings which can help to understand well in a graph against continuous/categorical data. It can draw a two-dimensional graph.

Syntax: *seaborn.scatterplot(x=None, y=None)*

Parameters:

x, y: *Input data variables that should be numeric.*

Returns: *This method returns the Axes object with the plot drawn onto it.*

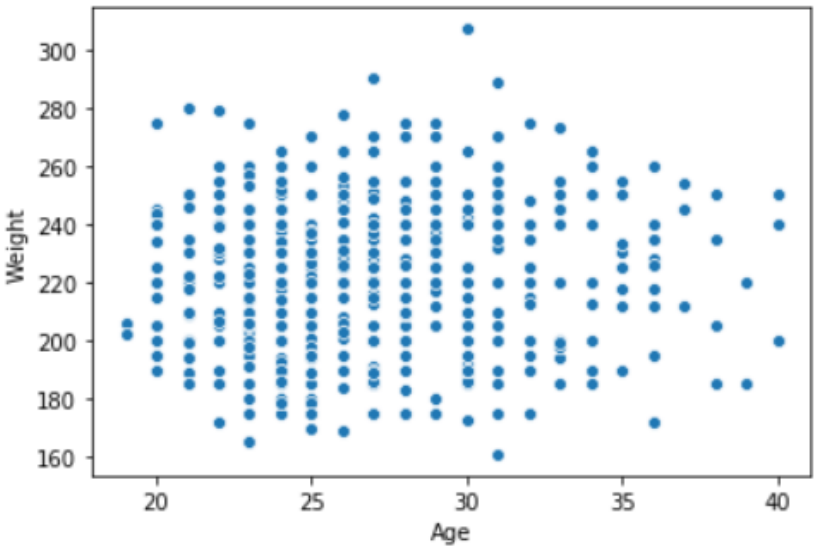
Let's visualize the data with a scatter plot and pandas:

Example 1:

Python3

```
# import module
import seaborn
import pandas

# load csv
data = pandas.read_csv("nba.csv")

# plotting
seaborn.scatterplot(data['Age'],data['Weight'])
```

Output:

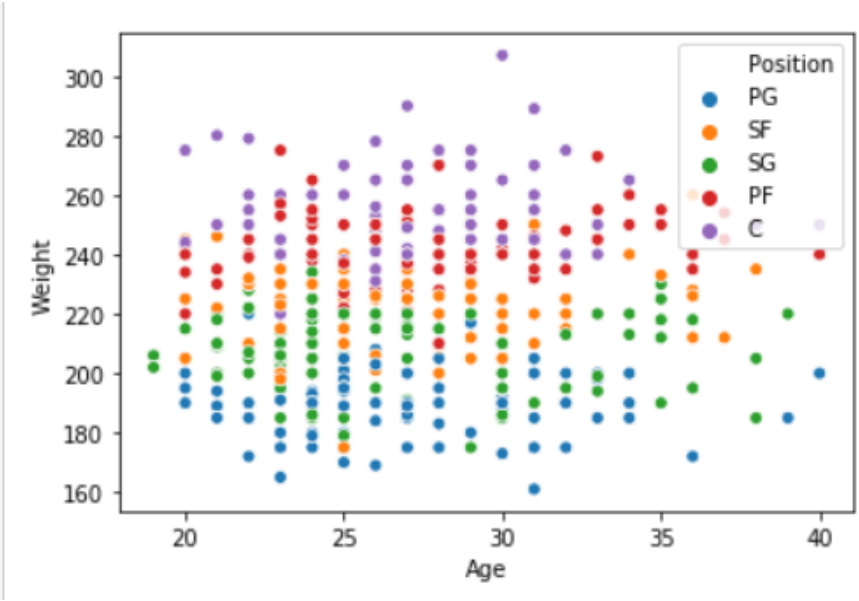


Example 2: Use the hue parameter for plotting the graph.

Python3

```
import seaborn
import pandas
data = pandas.read_csv("nba.csv")

seaborn.scatterplot( data['Age'], data['Weight'], hue =data["Posit
```

Output:



Box plot:

A box plot (or box-and-whisker plot) s is the visual representation of the depicting groups of numerical data through their quartiles against continuous/categorical data.

A box plot consists of 5 things.

- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%
- Third Quartile or 75%
- Maximum

Syntax:

seaborn.boxplot(x=None, y=None, hue=None, data=None)

Parameters:

- ***x, y, hue:*** *Inputs for plotting long-form data.*
- ***data:*** *Dataset for plotting. If x and y are absent, this is interpreted as wide-form.*

Returns: *It returns the Axes object with the plot drawn onto it.*

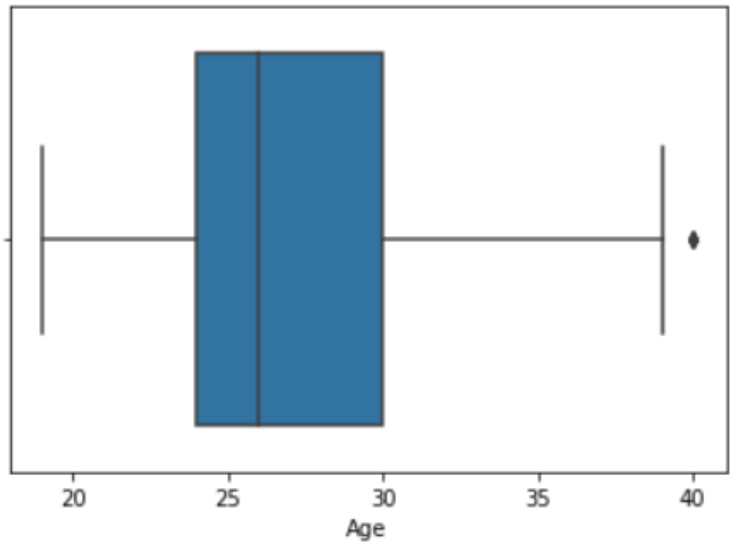
Draw the box plot with Pandas:

Example 1:

Python3

```
# import module
import seaborn as sns
import pandas

# read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'] )
```

Output:

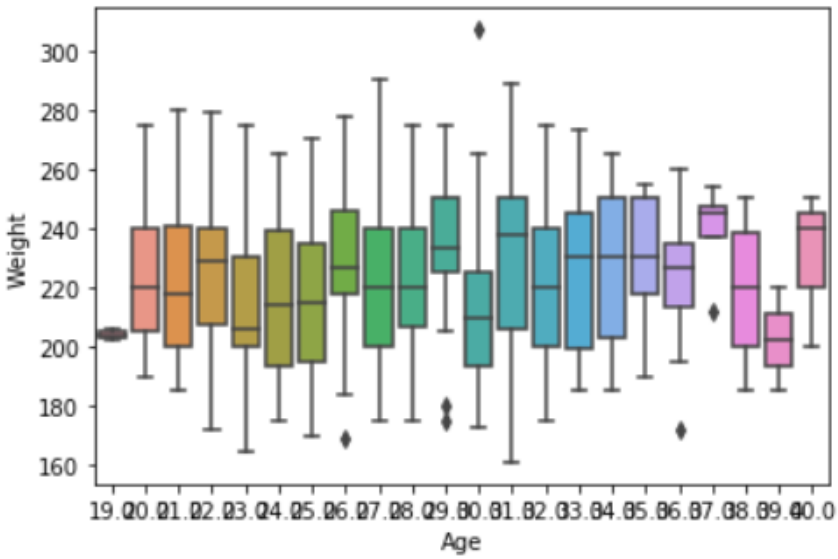


Example 2:

Python3

```
# import module
import seaborn as sns
import pandas

# read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'], data['Weight'])
```

Output:



Violin Plot:

A violin plot is similar to a boxplot. It shows several quantitative data across one or more categorical variables such that those distributions can be compared.

> ***Syntax:*** *seaborn.violinplot(x=None, y=None, hue=None, data=None)*
>
> ***Parameters:***
>
> - ***x, y, hue:*** *Inputs for plotting long-form data.*
> - ***data:*** *Dataset for plotting.*

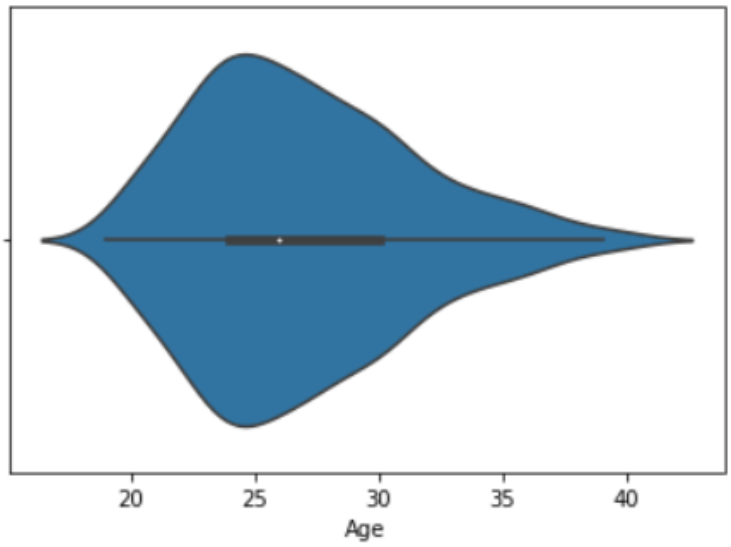
Draw the violin plot with Pandas:

Example 1:

Python3

```
# import module
import seaborn as sns
import pandas

# read csv and plot
data = pandas.read_csv("nba.csv")
sns.violinplot(data['Age'])
```

Output:

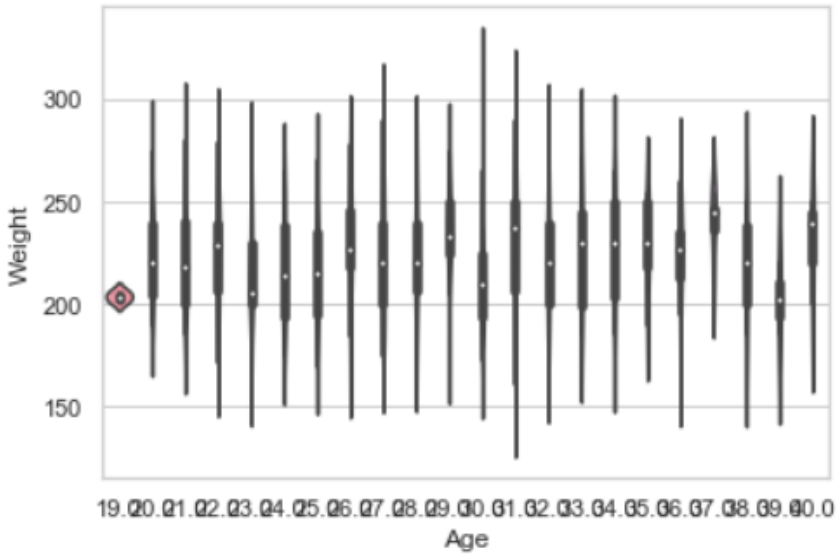


Example 2:

Python3

```
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.violinplot(x ="Age", y ="Weight",data = data)
```

Output:



Swarm plot:

A swarm plot is similar to a strip plot, We can draw a swarm plot with non-overlapping points against categorical data.

Syntax: *seaborn.swarmplot(x=None, y=None, hue=None, data=None)*

Parameters:

- ***x, y, hue:*** *Inputs for plotting long-form data.*
- ***data:*** *Dataset for plotting.*

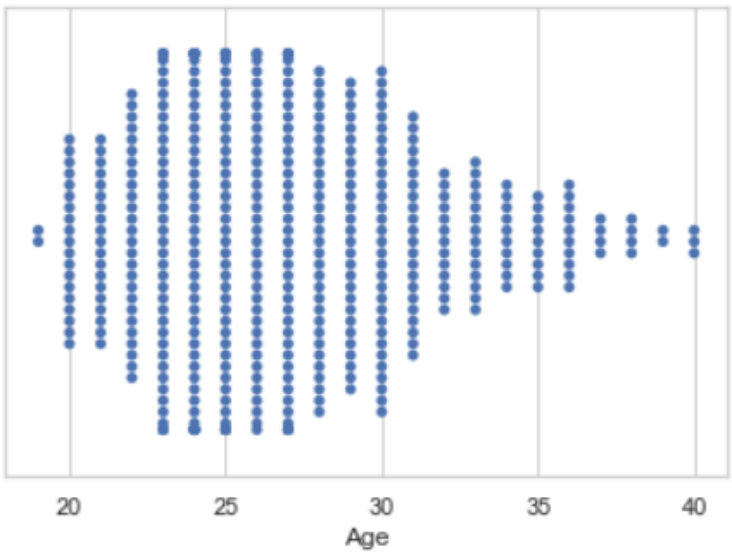
Draw the swarm plot with Pandas:

Example 1:

Python3

```
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv( "nba.csv" )
seaborn.swarmplot(x = data["Age"])
```

Output:



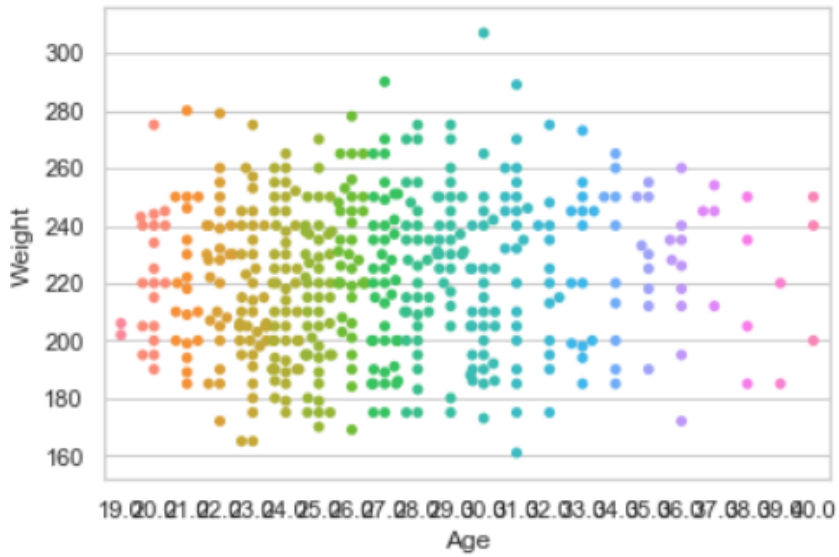
Example 2:

Python3

```
# import module
import seaborn
```

```
seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.swarmplot(x ="Age", y ="Weight",data = data)
```

Output:



Bar plot:

Barplot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars.

Syntax : *seaborn.barplot(x=None, y=None, hue=None, data=None)*

Parameters :

- ***x, y :*** *This parameter take names of variables in data or vector data, Inputs for plotting long-form data.*
- ***hue :*** *(optional) This parameter take column name for colour encoding.*
- ***data :*** *(optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.*

Returns : *Returns the Axes object with the plot drawn onto it.*

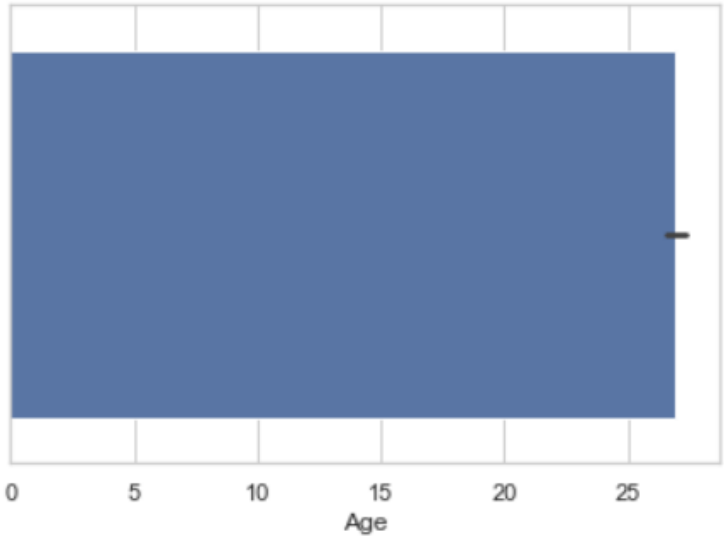
Draw the bar plot with Pandas:

Example 1:

Python3

```
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.barplot(x =data["Age"])
```

Output:

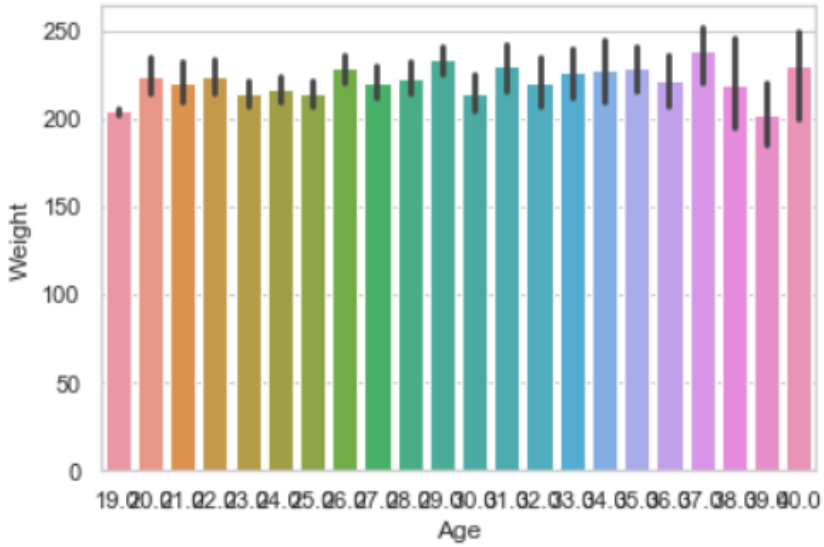


Example 2:

Python3

```
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.barplot(x ="Age", y ="Weight", data = data)
```

Output:



Point plot:

Point plot used to show point estimates and confidence intervals using scatter plot glyphs. A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.

> ***Syntax:*** *seaborn.pointplot(x=None, y=None, hue=None, data=None)*
>
> ***Parameters:***
>
> - ***x, y:*** *Inputs for plotting long-form data.*
> - ***hue:*** *(optional) column name for color encoding.*
> - ***data:*** *dataframe as a Dataset for plotting.*
>
> ***Return:*** *The Axes object with the plot drawn onto it.*

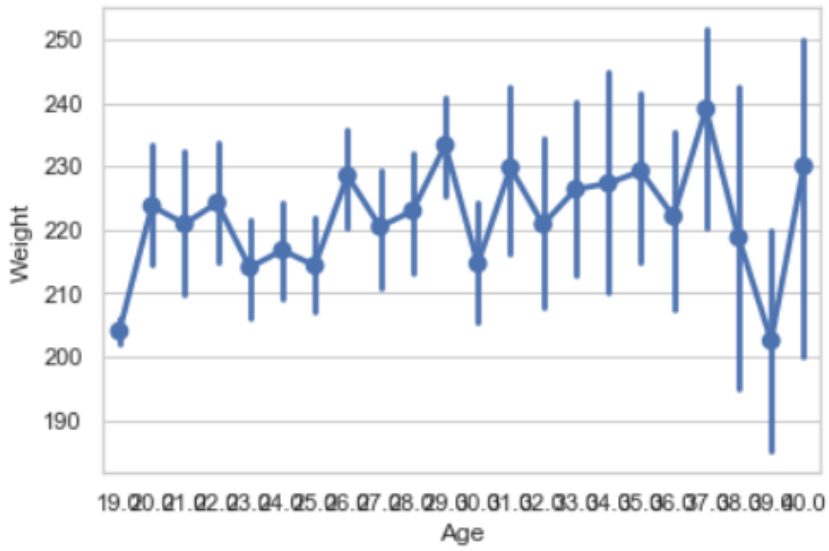
Draw the point plot with Pandas:

Example:

Python3

```
# import module
import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.pointplot(x = "Age", y = "Weight", data = data)
```

Output:



Count plot:

Count plot used to Show the counts of observations in each categorical bin using bars.

Syntax : *seaborn.countplot(x=None, y=None, hue=None, data=None)*

Parameters :

- ***x, y:*** *This parameter take names of variables in data or vector data, optional, Inputs for plotting long-form data.*
- ***hue :*** *(optional) This parameter take column name for color encoding.*
- ***data :*** *(optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise, it is expected to be long-form.*

Returns: *Returns the Axes object with the plot drawn onto it.*

Draw the count plot with Pandas:

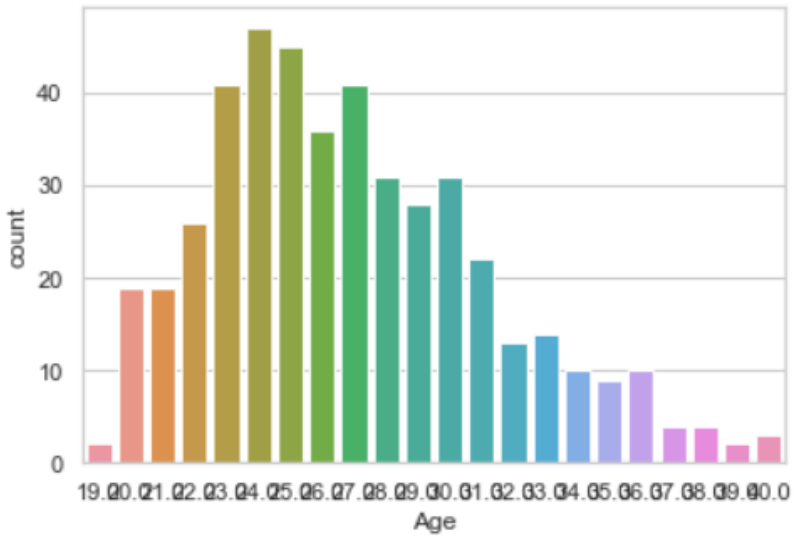
Example:

Python3

```
# import module
import seaborn

seaborn.set(style = 'whitegrid')
```

```
# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.countplot(data["Age"])
```

Output:



KDE Plot:

KDE Plot described as **Kernel Density Estimate** is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.

> ***Syntax:*** *seaborn.kdeplot(x=None, *, y=None, vertical=False, palette=None, **kwargs)*
>
> ***Parameters:***
>
> ***x, y :*** *vectors or keys in data*
>
> ***vertical :*** *boolean (True or False)*
>
> ***data :*** *pandas.DataFrame, numpy.ndarray, mapping, or sequence*

Draw the KDE plot with Pandas:

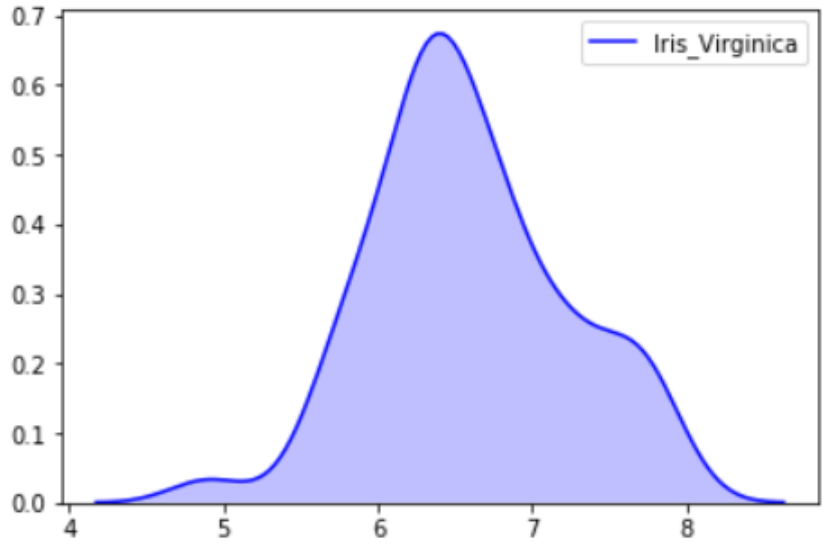
Example 1:

Python3

```
# importing the required libraries
from sklearn import datasets
import pandas as pd
```

```
import seaborn as sns

# Setting up the Data Frame
iris = datasets.load_iris()

iris_df = pd.DataFrame(iris.data, columns=['Sepal_Length',
                       'Sepal_Width', 'Patal_Length', 'Petal_Width'

iris_df['Target'] = iris.target

iris_df['Target'].replace([0], 'Iris_Setosa', inplace=True)
iris_df['Target'].replace([1], 'Iris_Vercicolor', inplace=True)
iris_df['Target'].replace([2], 'Iris_Virginica', inplace=True)

# Plotting the KDE Plot
sns.kdeplot(iris_df.loc[(iris_df['Target'] =='Iris_Virginica'),
            'Sepal_Length'], color = 'b', shade = True, Label ='Ir
```

Output:

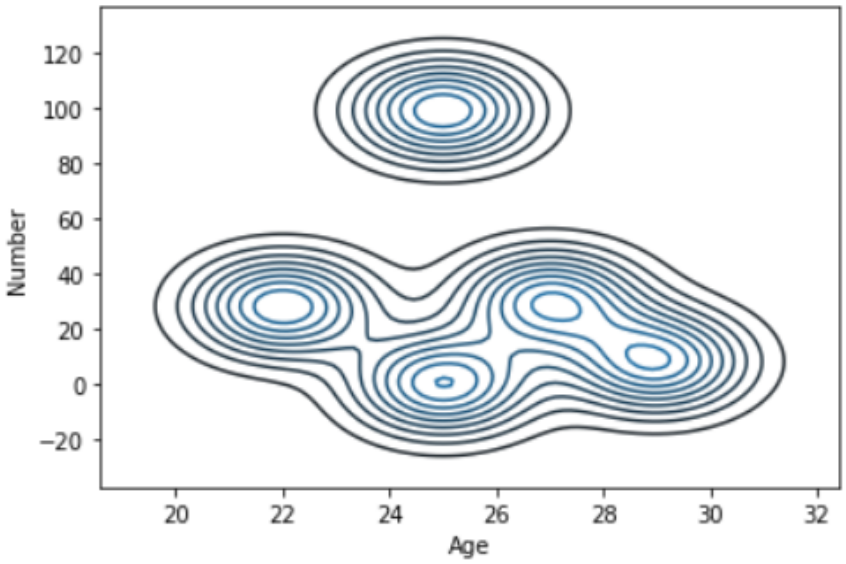


Example 2:

Python3

```
# import module
import seaborn as sns
import pandas

# read top 5 column
data = pandas.read_csv("nba.csv").head()

sns.kdeplot( data['Age'], data['Number'])
```

Output:



Bivariate and Univariate data using seaborn and pandas:

Before starting let's have a small intro of bivariate and univariate data:

Bivariate data: This type of data involves **two different variables**. The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship between the two variables.

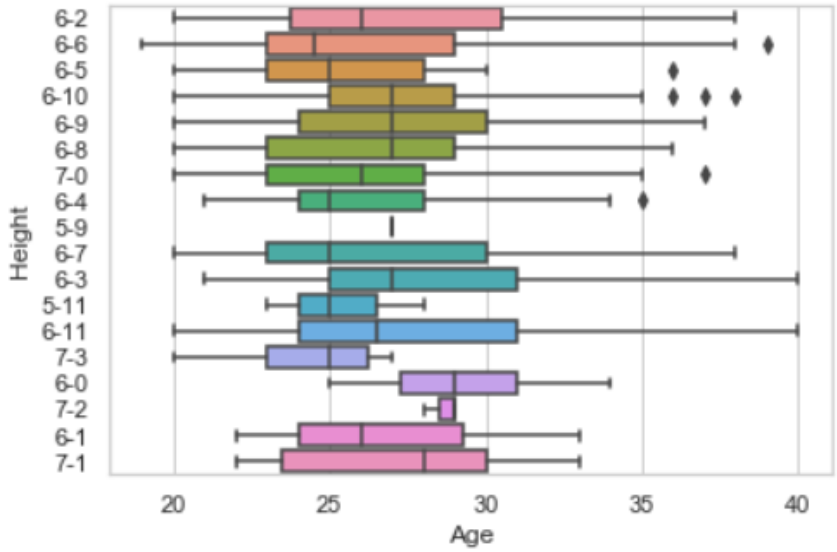
Univariate data: This type of data consists of **only one variable**. The analysis of univariate data is thus the simplest form of analysis since the information deals with only one quantity that changes. It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it.

Let's see an example of Bivariate data :
Example 1: Using the box plot.

Python3

```
# import module
import seaborn as sns
import pandas

# read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'], data['Height'])
```

Output:

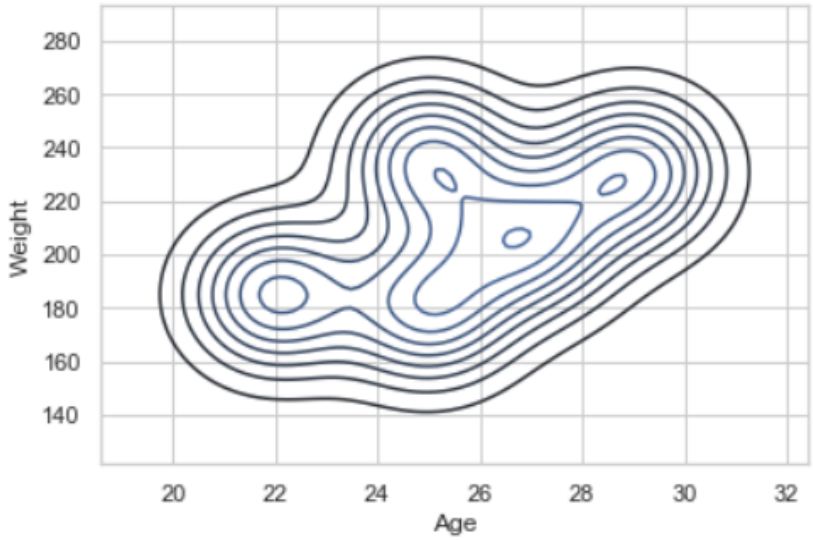


Example 2: using KDE plot.

Python3

```
# import module
import seaborn as sns
import pandas

# read top 5 column
data = pandas.read_csv("nba.csv").head()

sns.kdeplot( data['Age'], data['Weight'])
```

Output:



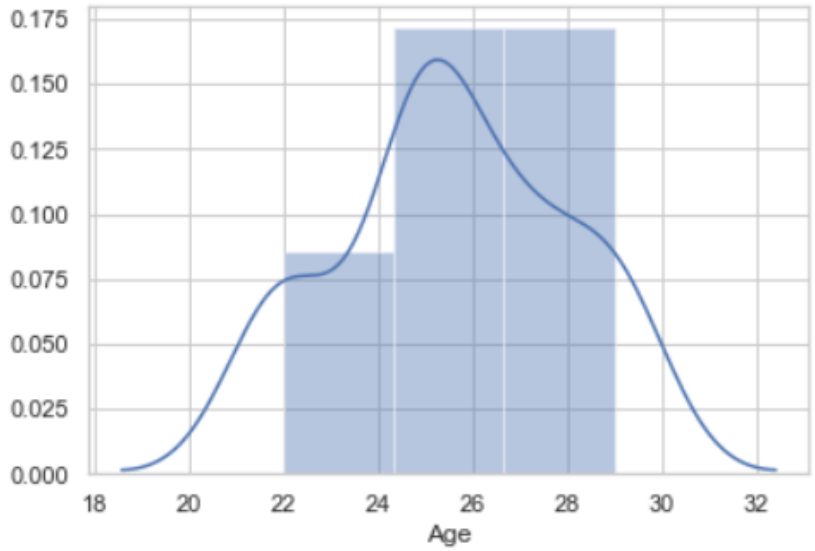
Let's see an example of univariate data distribution:

Example: Using the dist plot

Python3

```
# import module
import seaborn as sns
```

```
import pandas

# read top 5 column
```

Last Updated : 15 Jan, 2022

14

Previous

Python - Data visualization tutorial

Next

Python - Data visualization tutorial

Similar Reads

Data Visualization with Seaborn Line Plot

Data visualization with Pairplot Seaborn and Pandas

Box plot visualization with Pandas and Seaborn

KDE Plot Visualization with Pandas and Seaborn

Difference Between Data Science and Data Visualization

Why is Data Visualization so Important in Data Science?

Conditional Data Visualization Using Google Data Studio

Python | Titanic Data EDA using Seaborn

Data Visualisation in Python using Matplotlib and Seaborn

How to Make Boxplots with Data Points using Seaborn in Python?

Complete Tutorials

OpenAI Python API - Complete Guide

Pandas AI: The Generative AI Python Library

Python for Kids - Fun Tutorial to Learn Python Programming

Data Analysis Tutorial

Flask Tutorial

Article Contributed By :

kumar_satyam

kumar_satyam

Vote for difficulty

Current difficulty : Hard

Easy | Normal | Medium | Hard | Expert

Improved By : adnanirshad158, sooda367, ruhelaa48

Article Tags : Data Visualization , Python-pandas , Python-Seaborn , Technical Scripter 2020 , Data Science , Python , Technical Scripter

Practice Tags : python

Improve Article | Report Issue

GeeksforGeeks
Sanchhaya Education Private Limited
A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

Apply for Mentor

GeeksforGeeks Videos

All Cheat Sheets

NumPy Tutorial

Deep Learning Tutorial

Computer Science

GATE CS Notes

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Python

Python Programming Examples

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

Python Interview Question

DevOps

Git

AWS

Docker

Kubernetes

Azure

GCP

DevOps Roadmap

Competitive Programming

Top DS or Algo for CP

Top 50 Tree

Top 50 Graph

Top 50 Array

Top 50 String

Top 50 DP

Top 15 Websites for CP

System Design

What is System Design

Monolithic and Distributed SD

High Level Design or HLD

Low Level Design or LLD

Crack System Design Round

System Design Interview Questions

Grokking Modern System Design

JavaScript

TypeScript

ReactJS

NextJS

AngularJS

NodeJS

Express.js

Lodash

Web Browser

NCERT Solutions

NCERT Solutions for Class 12

NCERT Solution for Class 11

NCERT Solutions for Class 10

NCERT Solutions for Class 9

NCERT Solutions for Class 8

Complete Study Material

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

Accountancy

Business Studies

Indian Economics

Macroeconomics

Microeconimics

Statistics for Economics

Management & Finance

Management

HR Managament

Income Tax

Finance

Economics

UPSC

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economics Notes

Important Topics in Ethics

UPSC Previous Year Papers

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

SSC CGL Practice Papers

Colleges

Companies

IT Companies

Preparation Corner

Exams

JEE Mains

More Tutorials

Write & Earn

Write an Article

Python For Data Science *Cheat Sheet*

Matplotlib

Learn Python Interactively at www.DataCamp.com

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

matplotlib

1 Prepare The Data

Also see Lists & NumPy

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 + Y
>>> V = 1 + X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

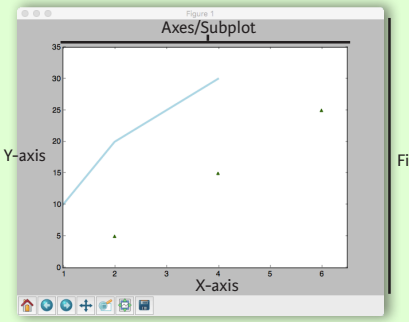
Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2,ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

Plot Anatomy & Workflow

Plot Anatomy



Workflow

The basic steps to creating plots with matplotlib are:

1 Prepare data 2 Create plot 3 Plot 4 Customize plot 5 Save plot 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4]                          Step 1
>>> y = [10,20,25,30]
>>> fig = plt.figure()                     Step 2
>>> ax = fig.add_subplot(111)              Step 3
>>> ax.plot(x, y, color='lightblue', linewidth=3)   Step 3, 4
>>> ax.scatter([2,4,6],
               [5,15,25],
               color='darkgreen',
               marker='^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png')
>>> plt.show()                             Step 6
```

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img,
                   cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker=".")
>>> ax.plot(x,y,marker="o")
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-.')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,
            -2.1,
            'Example Graph',
            style='italic')
>>> ax.annotate("Sine",
                xy=(8, 0),
                xycoords='data',
                xytext=(10.5, 0),
                textcoords='data',
                arrowprops=dict(arrowstyle="->",
                                connectionstyle="arc3"),)
```

Mathtext

```
>>> plt.title(r'$sigma_i=15$', fontsize=20)
```

Limits, Legends & Layouts

Limits & Autoscaling

```
>>> ax.margins(x=0.0,y=0.1)                  Add padding to a plot
>>> ax.axis('equal')                         Set the aspect ratio of the plot to 1
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])    Set limits for x-and y-axis
>>> ax.set_xlim(0,10.5)                      Set limits for x-axis
```

Legends

```
>>> ax.set(title='An Example Axes',          Set a title and x-and y-axis labels
           ylabel='Y-Axis',
           xlabel='X-Axis')
>>> ax.legend(loc='best')                    No overlapping plot elements
```

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),           Manually set x-ticks
                 ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y',                 Make y-ticks longer and go in and out
                   direction='inout',
                   length=10)
```

Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,         Adjust the spacing between subplots
                         hspace=0.3,
                         left=0.125,
                         right=0.9,
                         top=0.9,
                         bottom=0.1)
>>> fig.tight_layout()                       Fit subplot(s) in to the figure area
```

Axis Spines

```
>>> ax1.spines['top'].set_visible(False)     Make the top axis line for a plot invisible
>>> ax1.spines['bottom'].set_position(('outward',10))   Move the bottom axis line outward
```

3 Plotting Routines

1D Data

```
>>> fig, ax = plt.subplots()
>>> lines = ax.plot(x,y)                     Draw points with lines or markers connecting them
>>> ax.scatter(x,y)                          Draw unconnected points, scaled or colored
>>> axes[0,0].bar([1,2,3],[3,4,5])           Plot vertical rectangles (constant width)
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])      Plot horiontal rectangles (constant height)
>>> axes[1,1].axhline(0.45)                  Draw a horizontal line across axes
>>> axes[0,1].axvline(0.65)                  Draw a vertical line across axes
>>> ax.fill(x,y,color='blue')                Draw filled polygons
>>> ax.fill_between(x,y,color='yellow')      Fill between y-values and o
```

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)             Add an arrow to the axes
>>> axes[1,1].quiver(y,z)                    Plot a 2D field of arrows
>>> axes[0,1].streamplot(X,Y,U,V)            Plot a 2D field of arrows
```

Data Distributions

```
>>> ax1.hist(y)                              Plot a histogram
>>> ax3.boxplot(y)                           Make a box and whisker plot
>>> ax3.violinplot(z)                        Make a violin plot
```

2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,                      Colormapped or RGB arrays
                   cmap='gist_earth',
                   interpolation='nearest',
                   vmin=-2,
                   vmax=2)
```

```
>>> axes2[0].pcolor(data2)                   Pseudocolor plot of 2D array
>>> axes2[0].pcolormesh(data)                Pseudocolor plot of 2D array
>>> CS = plt.contour(Y,X,U)                  Plot contours
>>> axes2[2].contourf(data1)                 Plot filled contours
>>> axes2[2]= ax.clabel(CS)                  Label a contour plot
```

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
```

Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

```
>>> plt.show()
```

Close & Clear

```
>>> plt.cla()                                Clear an axis
>>> plt.clf()                                Clear the entire figure
>>> plt.close()                              Close a window
```

python-cheat-sheet / matplotlib / visualization / visualization.ipynb

xkhanhnguyen Visualization 9e67651 · 3 years ago History

3.38 MB

In [136…

```
import pandas as pd
import numpy as np
import plotly.offline as py
import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, i
init_notebook_mode(connected=True)

import seaborn as sns
plt.style.use('seaborn-whitegrid') # Set the aesthetic style of the plots
sns.set(style="white", color_codes=True)

# import matplotlib and set inline for jupyter notebook
import matplotlib.pyplot as plt
```

Relationship

1. Scatter plot using Matplotlib
2. Marginal Histogram
3. Scatter plot using Seaborn
4. Pair Plot in Seaborn
5. Heat Map

In [129…

```
# Source: https://www.kaggle.com/benhamner/python-data-visualizations

# load the iris data
iris = datasets.load_iris()
type(iris.data)

# Build a DataFrame for iris dataset
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['target']=iris.target
iris_df['species'] = iris_df['target'].map({0:iris.target_names[0],
                                            1:iris.target_names[1],
                                            2:iris.target_names[2]})
iris_df.head(10)
```

Out[129…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	species
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa
5	5.4	3.9	1.7	0.4	0	setosa
6	4.6	3.4	1.4	0.3	0	setosa
7	5.0	3.4	1.5	0.2	0	setosa
8	4.4	2.9	1.4	0.2	0	setosa

9	4.9	3.1	1.5	0.1	0	setosa

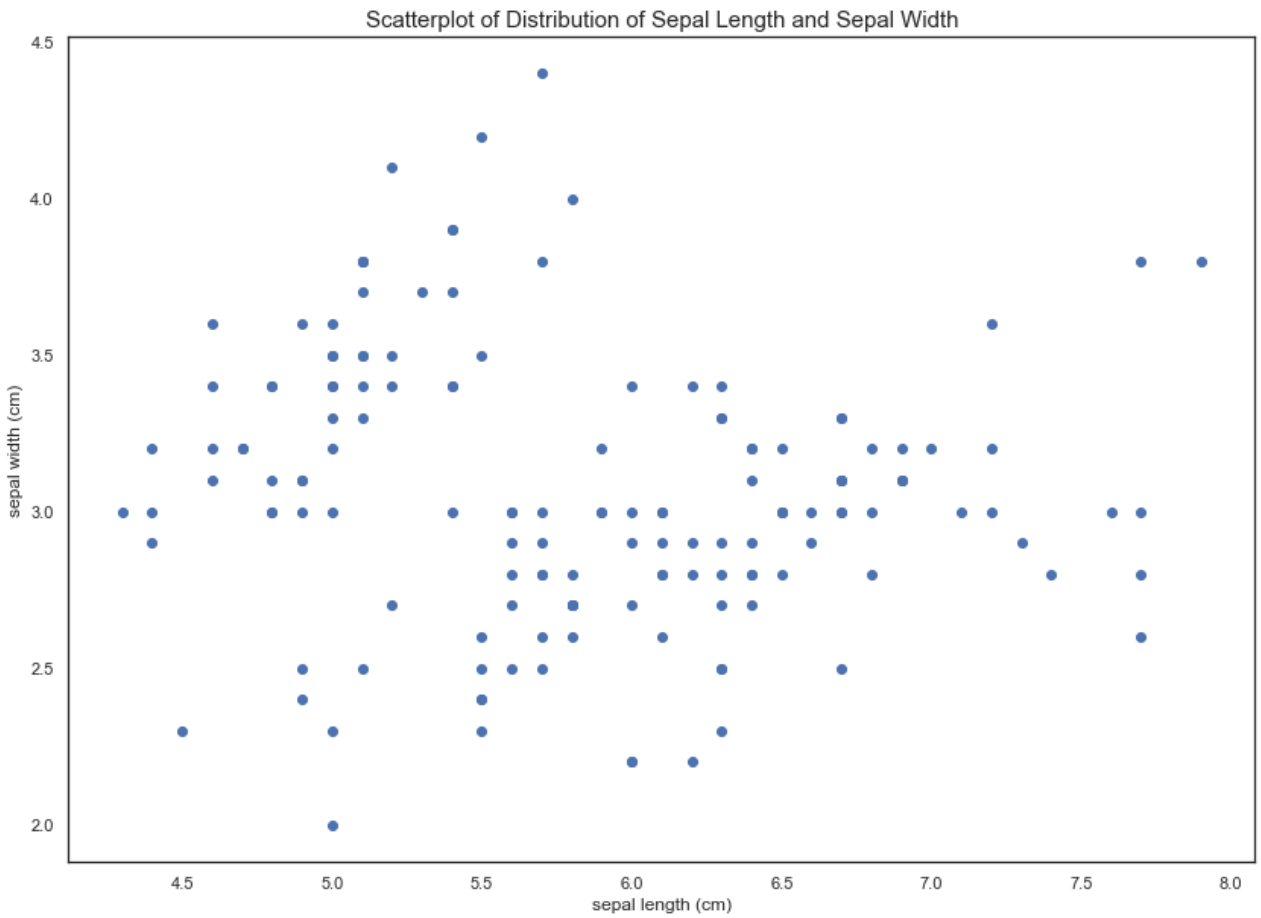
In [129…
```
# Let's see how many examples we have of each species
iris_df["species"].value_counts()
```

Out[129…
```
setosa        50
virginica     50
versicolor    50
Name: species, dtype: int64
```

1. Scatter Plot

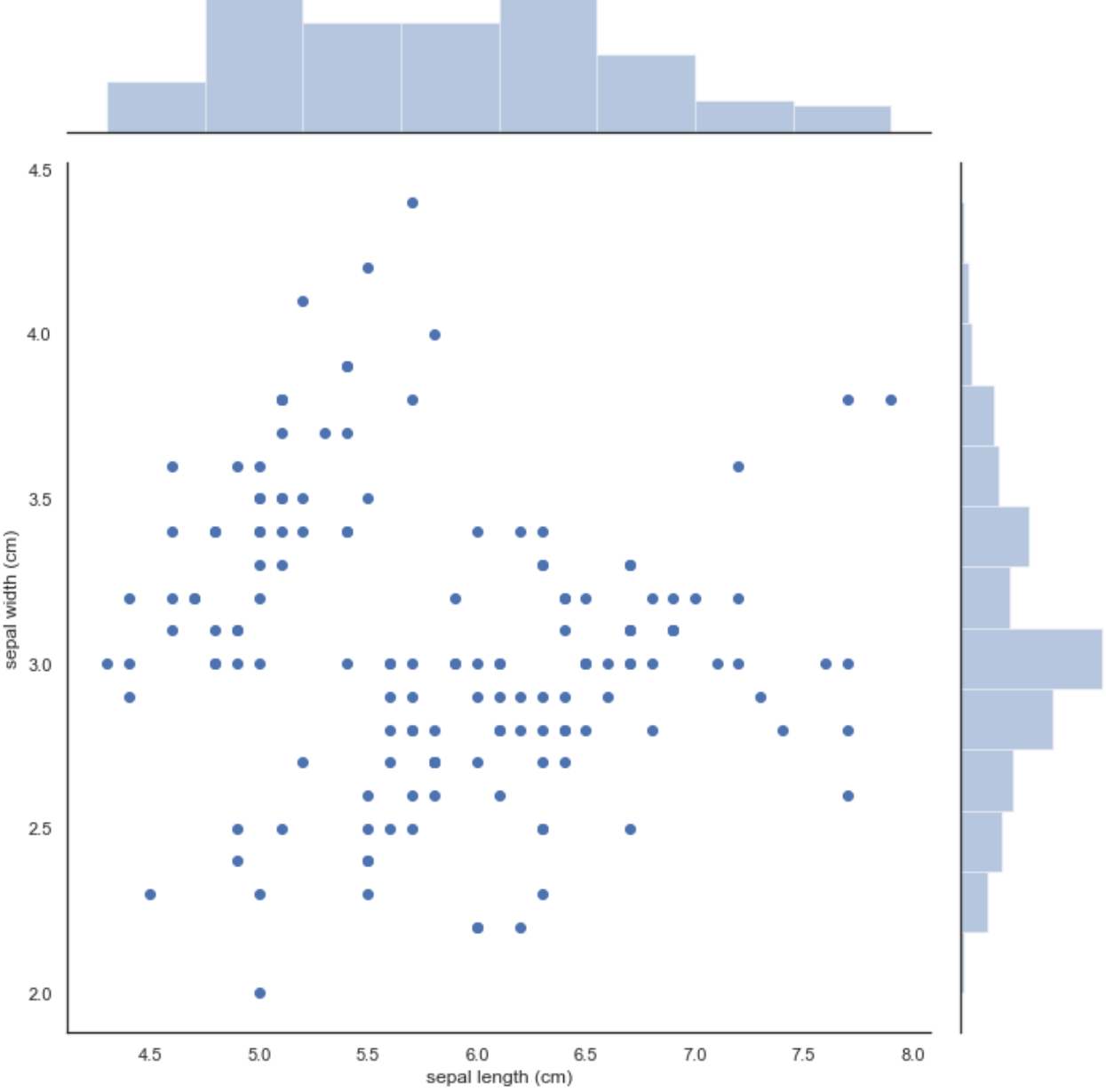
In [129…
```
#scatter plot of distribution of sepal length and sepal width

fig = plt.figure(figsize=(14,10))
plt.scatter(iris_df['sepal length (cm)'], iris_df['sepal width (cm)'])
plt.title('Scatterplot of Distribution of Sepal Length and Sepal Width',
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
```

Out[129…
```
Text(0, 0.5, 'sepal width (cm)')
```



2. Marginal Histogram

In [129…
```
# We can also use the seaborn library to make a similar plot
# A seaborn jointplot shows bivariate scatterplots and univariate histogr
p = sns.jointplot(iris_df['sepal length (cm)'], iris_df['sepal width (cm)
```



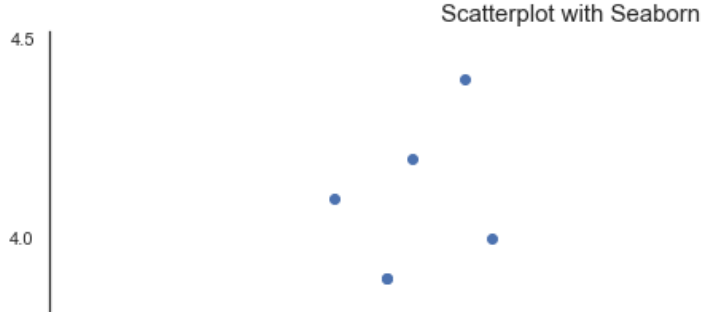
3. Scatter plot using Seaborn

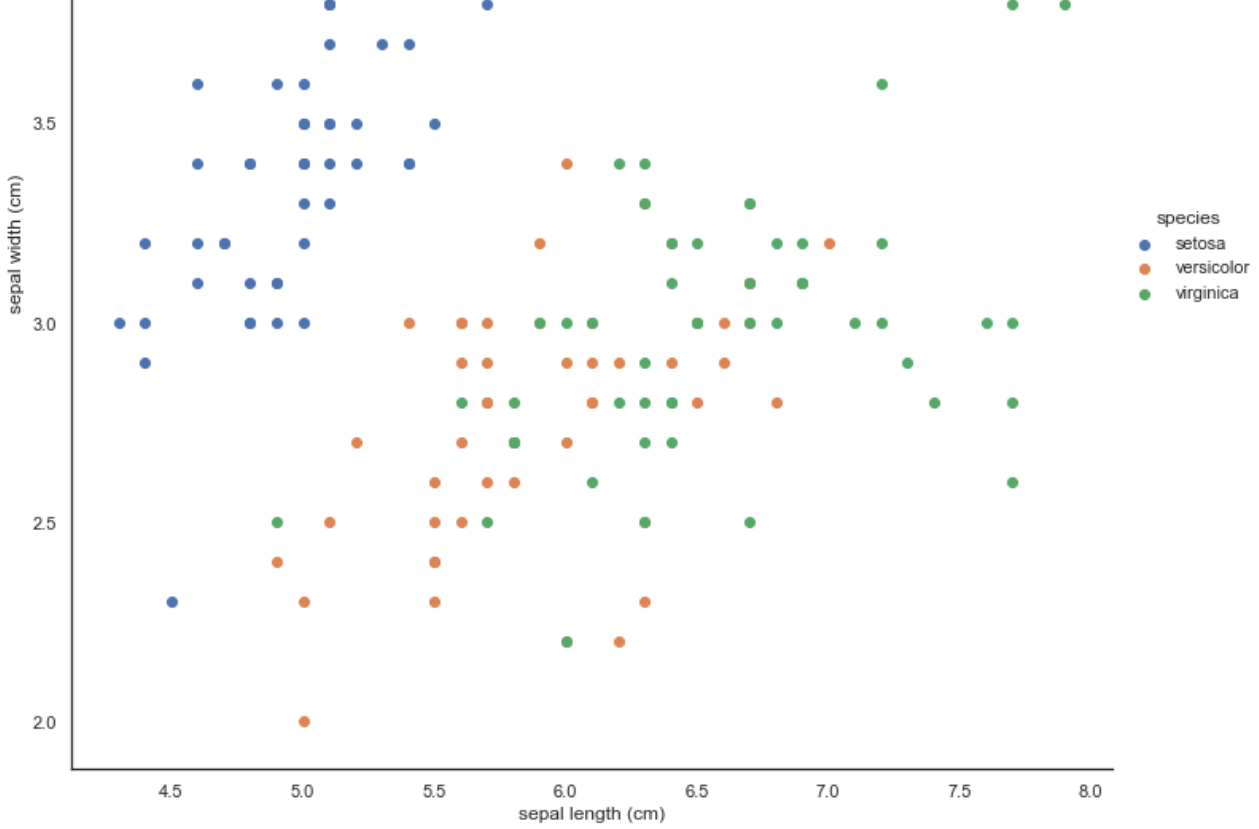
In [129…

```
# One piece of information missing in the plots above is what species eac
# We'll use seaborn's FacetGrid to color the scatterplot by species
sns.FacetGrid(iris_df, hue='species', size=10) \
   .map(plt.scatter, 'sepal length (cm)', 'sepal width (cm)') \
   .add_legend()
plt.title('Scatterplot with Seaborn', fontsize=15)
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packag
es/seaborn/axisgrid.py:243: UserWarning:

The `size` parameter has been renamed to `height`; please update your code.
```

Out[129… `Text(0.5, 1.0, 'Scatterplot with Seaborn')`



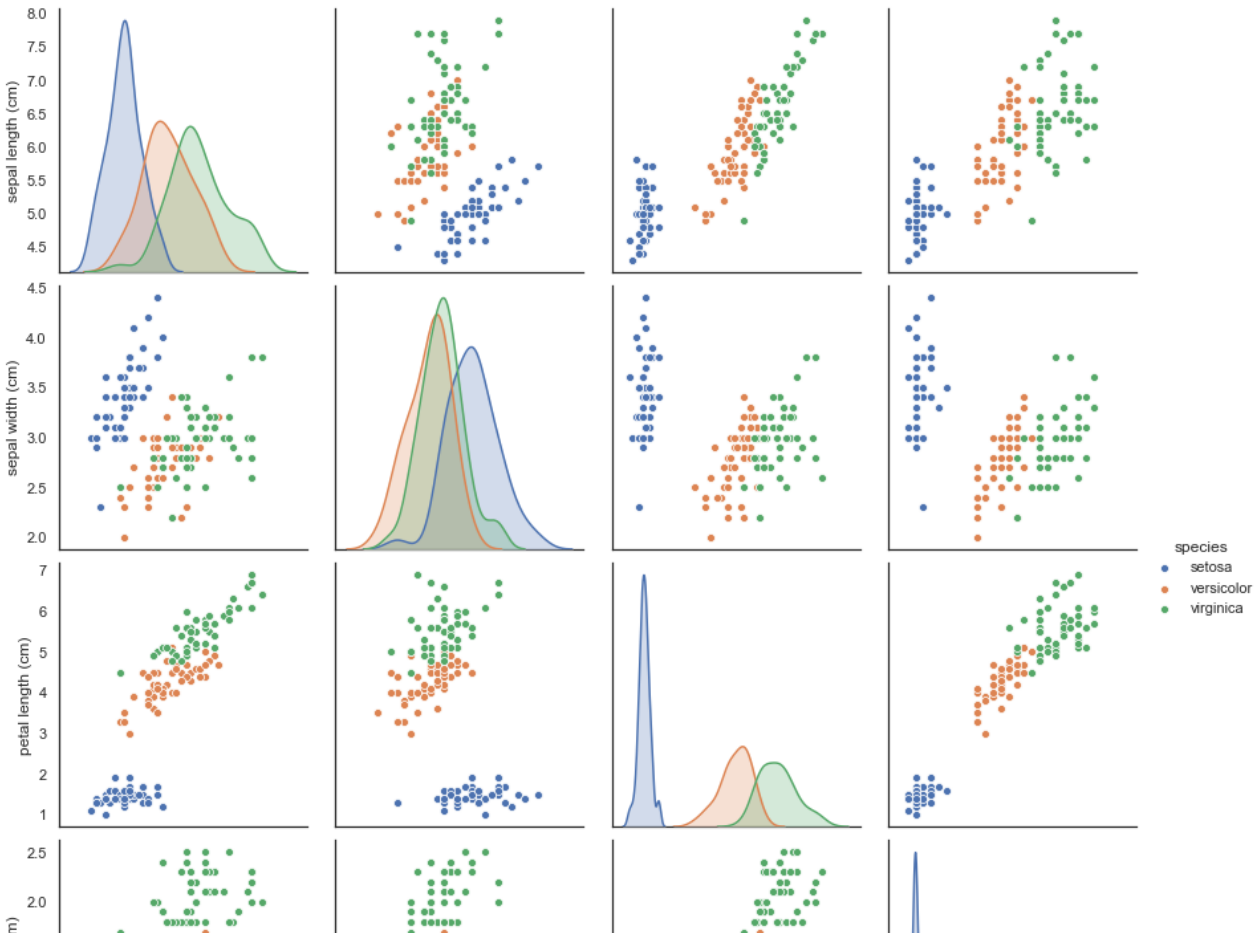


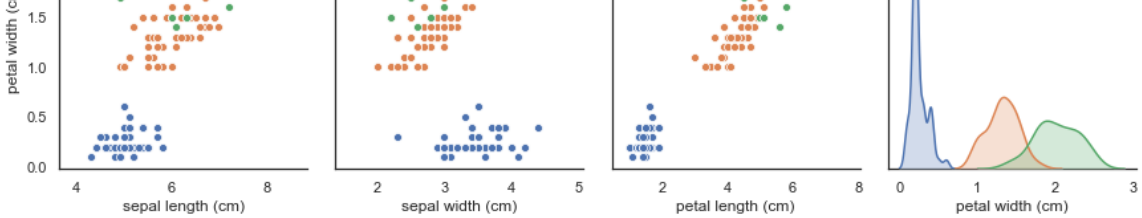
4. Pair plot

In [129…

```
# Another useful seaborn plot is the pair plot,
# which shows the bivariate relationship between each pair of features.
# From the pair plot, we'll see that the Iris-setosa species is separated
# from the other two across all feature combinations
sns.pairplot(iris_df.drop("target", axis=1), hue="species", height=3)
```

Out[129… <seaborn.axisgrid.PairGrid at 0x7f9bbcd5d8b0>



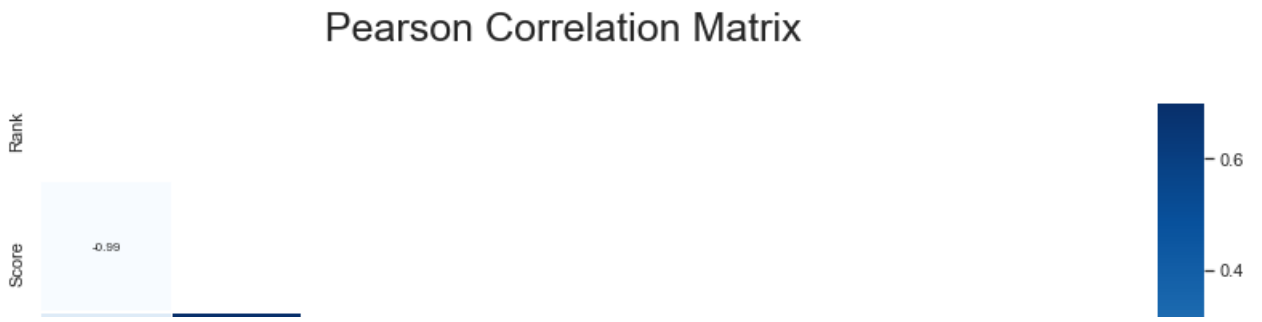


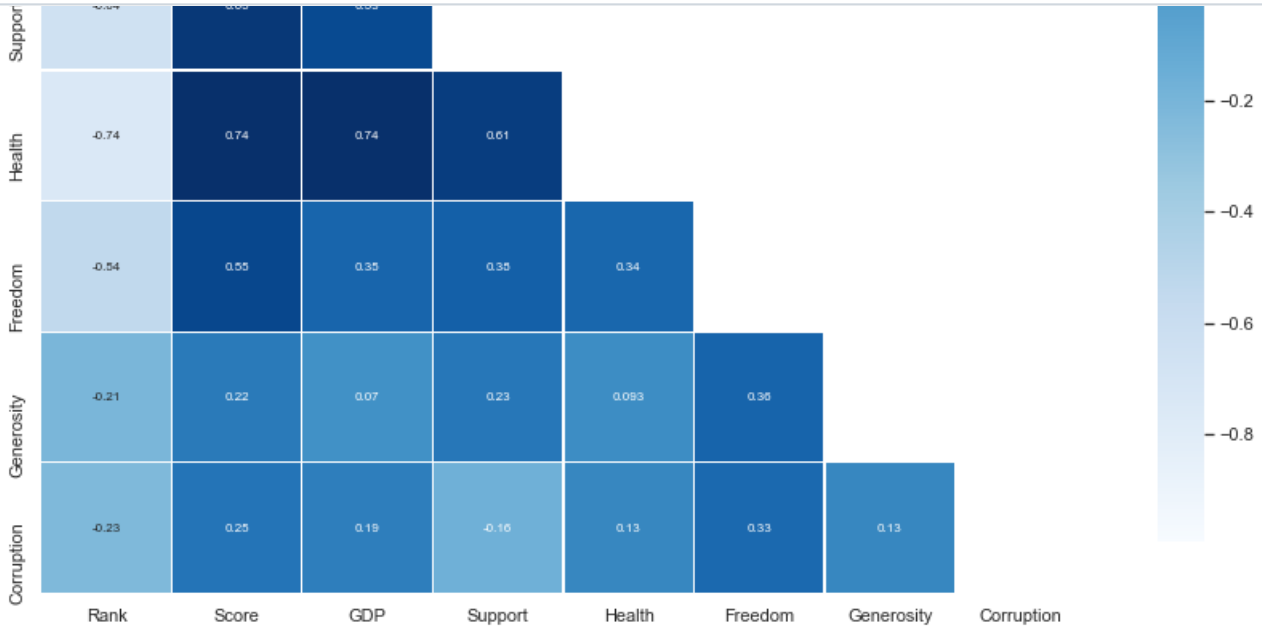
5. Heat map

In [129…

```
happy = pd.read_csv('happiness_rank.csv')
# Correlation Among variables
usecols = ['Rank','Country','Score','GDP','Support',
                'Health','Freedom','Generosity','Corruption']
mask = np.zeros_like(happy[usecols].corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(16, 12))
plt.title('Pearson Correlation Matrix',fontsize=25)

sns.heatmap(happy[usecols].corr(),
                        linewidths=0.25,vmax=0.7,square=True,cmap="Blues"
            linecolor='w',annot=True,annot_kws={"size":8},mask=mask,cbar_
```

Out[129… <AxesSubplot:title={'center':'Pearson Correlation Matrix'}>





Deviation plots

6. Line Chart

7. Area Chart

8. Stack Area Chart

9. Area Chart Unstacked

In [132…

```
# Build data frame from inputted data
data = pd.DataFrame(data = {'Month': ['Jan', 'Feb', 'March', 'April','May
                                      'July', 'Aug', 'Sept', 'Oct','Nov',
 'All Views': [728820, 862775, 1001285,817075,973996,877365,
               539134, 978768, 1792495, 569553, 989850, 1012303],
 'Internal Views': [374872, 466159, 578368,530037,608158,543964,
                    345293, 485921, 843923, 284741, 494991, 472819],
 'External Views': [260911, 301914,310423,197089,237425,226267,
                    193841, 492847, 948572, 284712, 494859, 539484]})
data.to_csv('Views.csv')
data
```

Out[132…

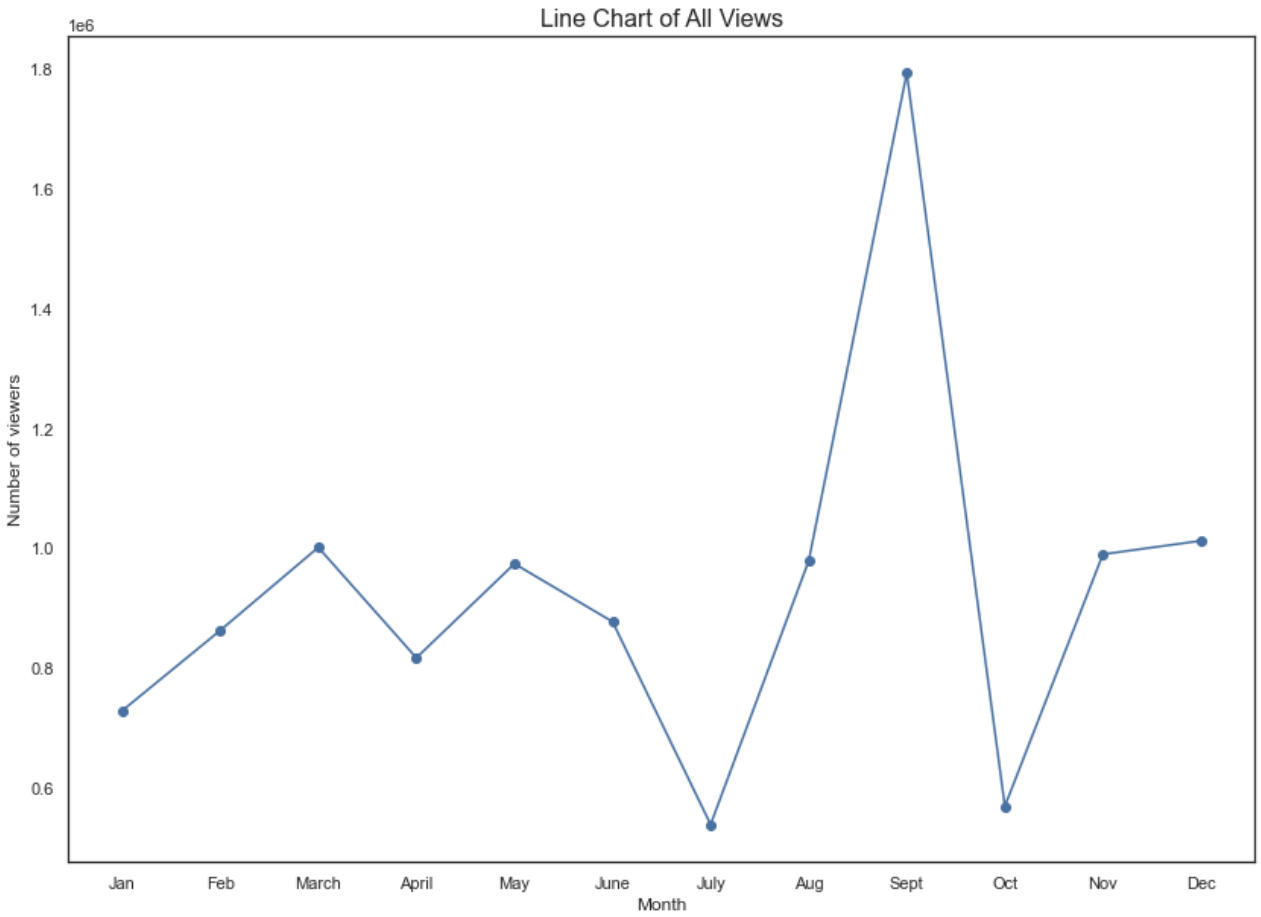
	Month	All Views	Internal Views	External Views
0	Jan	728820	374872	260911
1	Feb	862775	466159	301914
2	March	1001285	578368	310423
3	April	817075	530037	197089
4	May	973996	608158	237425
5	June	877365	543964	226267
6	July	539134	345293	193841
7	Aug	978768	485921	492847
8	Sept	1792495	843923	948572
9	Oct	569553	284741	284712
10	Nov	989850	494991	494859
11	Dec	1012303	472819	539484

6. Line Chart

In [133…

```
# instanciate the figure
fig = plt.figure(figsize = (14, 10))


# plot the data
plt.plot(data['Month'], data['All Views'], color='#4870a0', marker='o')

# set label and title for the plot
plt.xlabel('Month')
plt.ylabel('Number of viewers')
plt.title('Line Chart of All Views', fontsize = 16)
```

Out[133… Text(0.5, 1.0, 'Line Chart of All Views')

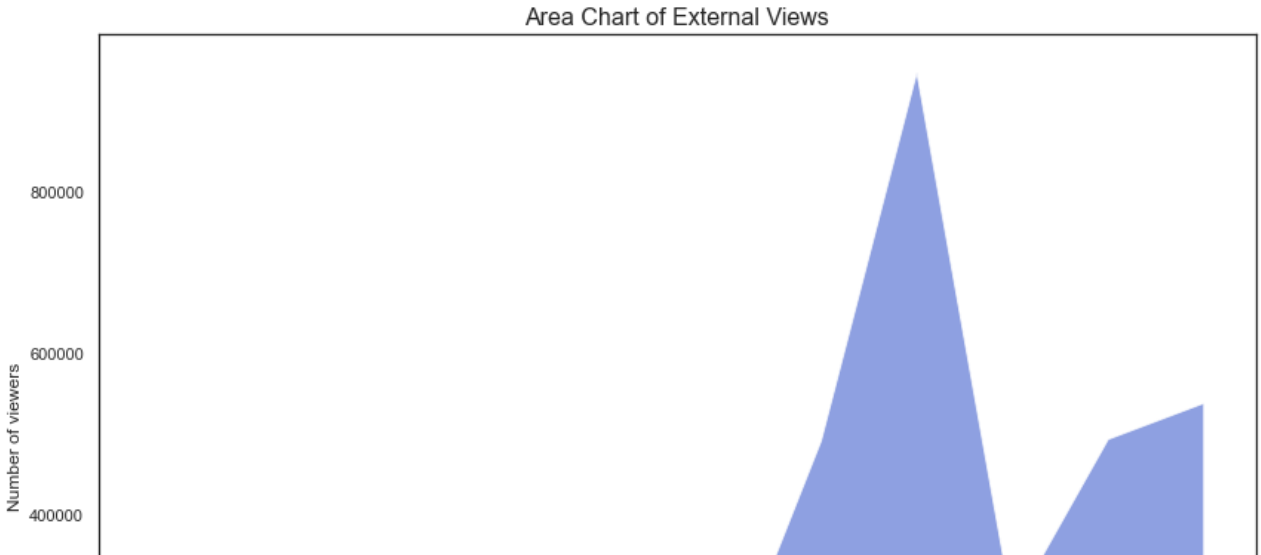


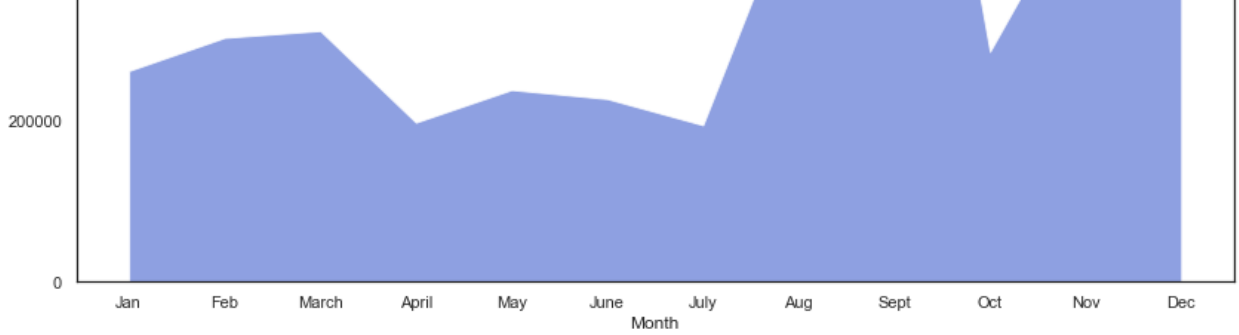
7. Area Chart

In [133…

```
# instanciate the figure
fig = plt.figure(figsize = (14, 10))


# plot the data
plt.stackplot(data['Month'], data['External Views'], colors='#7289da', al

# set label and title for the plot
plt.xlabel('Month')
plt.ylabel('Number of viewers')
plt.title('Area Chart of External Views', fontsize = 16)
```

Out[133… Text(0.5, 1.0, 'Area Chart of External Views')



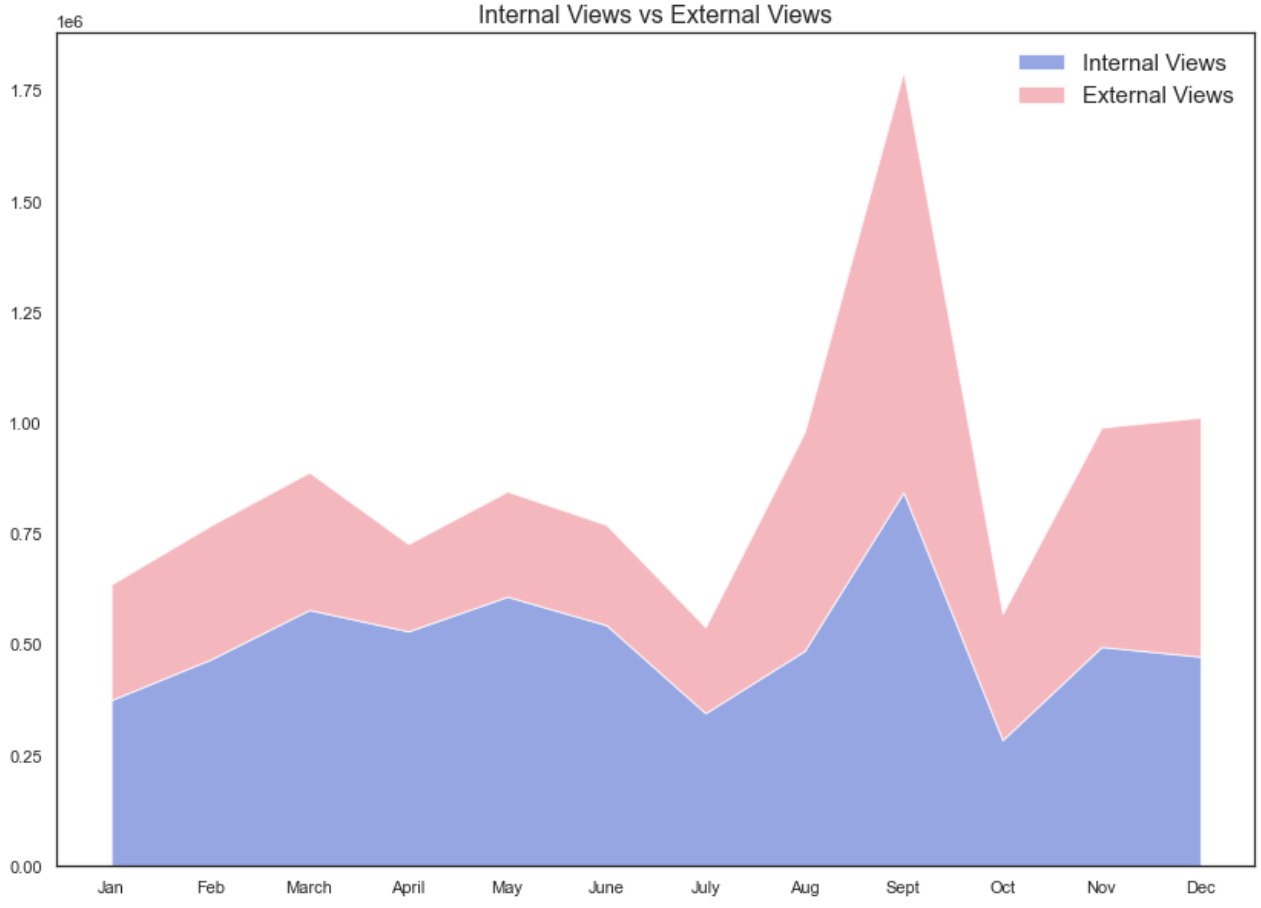


8. Stack Area Chart

In [134…

```
# instanciate the figure
fig = plt.figure(figsize = (14, 10))


# plot the data
plt.stackplot(data['Month'], data['Internal Views'], data['External Views
              alpha=0.75,
              colors=['#7289da','#f29fa9'],
             labels=['Internal Views', 'External Views'])


# set a title and a legend
plt.title('Internal Views vs External Views', fontsize=16)
plt.legend(fontsize=15)
```

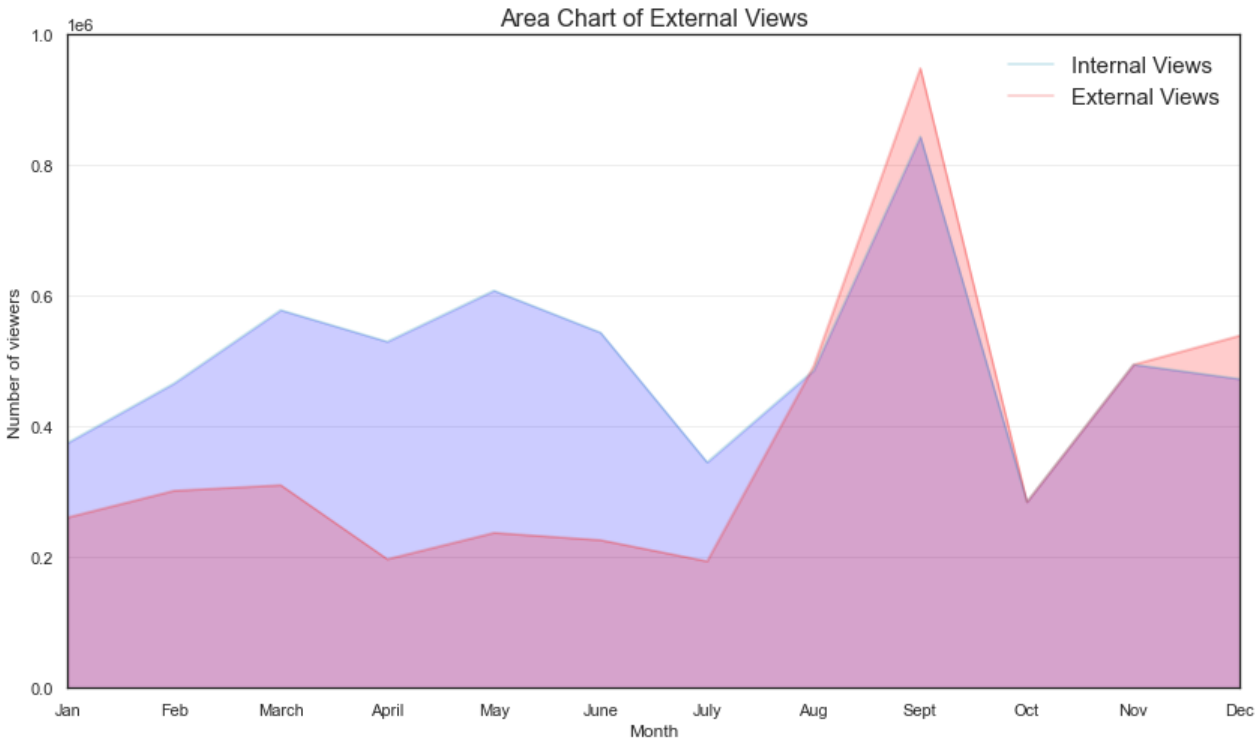
Out[134… <matplotlib.legend.Legend at 0x7f9ba5892490>



9. Area Chart Unstacked

In [136…

```
# instanciate the figure
fig = plt.figure(figsize=(14, 8))
ax = fig.add_subplot()

# plot the data
ax.plot(x, color='#49a7c3', alpha=0.3, label='Internal Views')
ax.plot(y, color='#f04747', alpha=0.3, label='External Views')

# fill the areas between the plots and the x axis
# this can create overlapping areas between lines
ax.fill_between(x.index, 0, x, color='blue', alpha=0.2)
ax.fill_between(x.index, 0, y, color='red', alpha=0.2)

# set ticks, label and title for the plot
plt.xticks(np.arange(12), data['Month'])
plt.xlabel('Month')
plt.ylabel('Number of viewers')
plt.title('Area Chart of External Views', fontsize = 16)


# change the x-ylim
ax.set_xlim(0, data.index[-1])
ax.set_ylim(0, 1e6)


# set a legend and the y grid for the plot
ax.legend(fontsize=15)
ax.grid(axis='y', alpha=0.3)
```



Ranking Plot

11. Vertical Bar Chart
12. Horizontal Bar Chart
13. Multi-set Bar Chart
14. Stack Bar Chart
15. Lollipop Chart

11. Vertical Bar Chart

In [123…

```
car = pd.read_csv('mpg_ggplot2.csv')
car
```

Out[123…

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	clas
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compac
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compac
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compac
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compac
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compac
...	...	...	...	...	...	...	...	...	...	...	.
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsiz
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsiz
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsiz
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsiz
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsiz

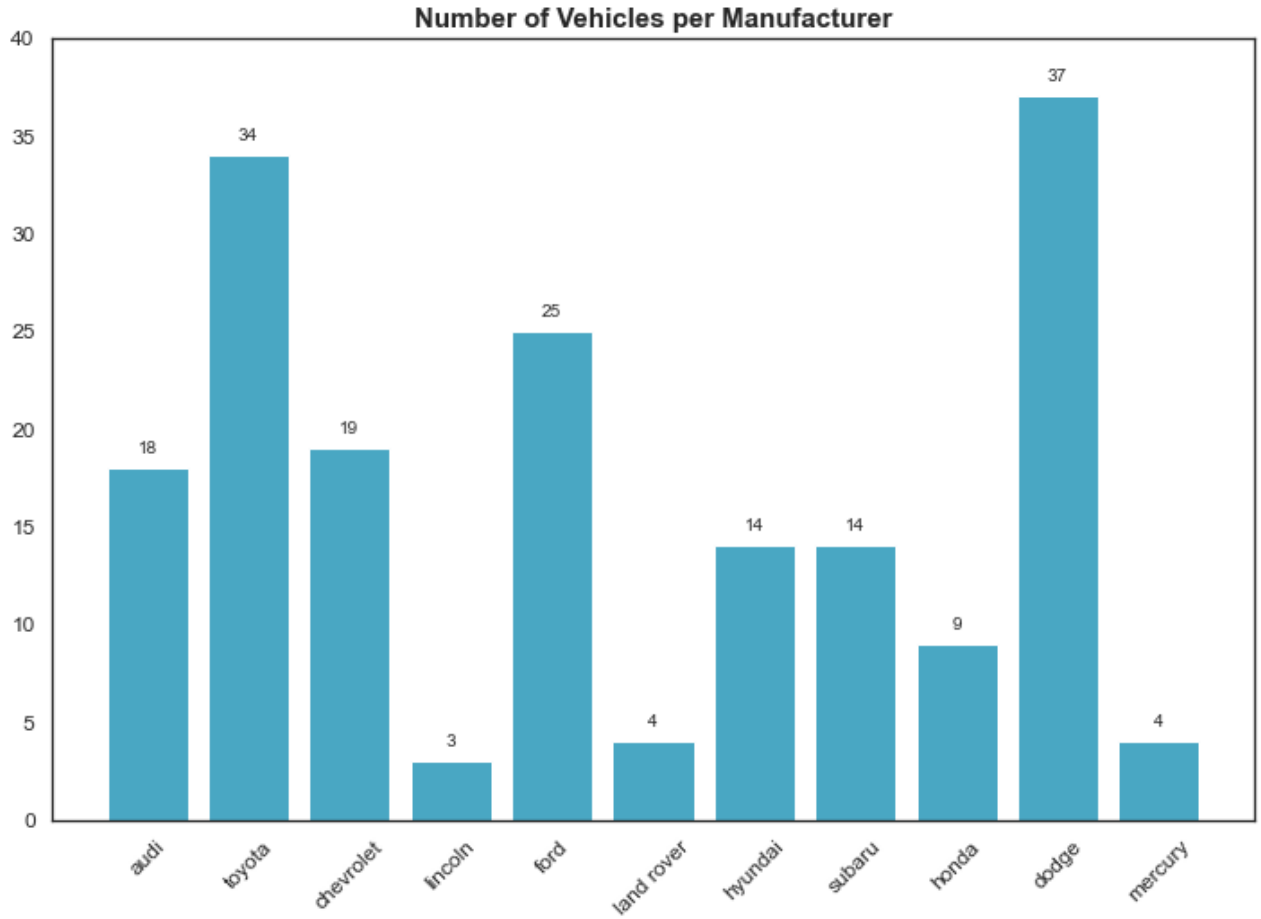
234 rows × 11 columns

In [127…

```
# Vertical bar chart
# prepare data
value_count =  car["manufacturer"].value_counts(sort=False)
value_count = value_count[:11,]


# instanciate the figure
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot()

# plot the data using matplotlib
ax.bar(value_count.index, value_count.values, color='#49a7c3') # color=co

# iterate over every x and y and annotate the value on the top of the bar
for i, (k, v) in enumerate(value_count.items()):
    ax.text(k, # where to put the text on the x coordinates
            v + 1, # where to put the text on the y coordinates
            v, # value to text
            #color=colors[i], # color corresponding to the bar
            fontsize=10, # fontsize
            horizontalalignment='center', # center the text to be more pl
            verticalalignment='center'
           )


# size of the x and y ticks
ax.tick_params(axis='x', labelrotation=45, labelsize=12)
ax.tick_params(axis='y', labelsize=12)

# change the ylim
ax.set_ylim(0, 40)
```

```
# set a title for the plot
ax.set_title('Number of Vehicles per Manufacturer', fontsize=15, fontweigl
```

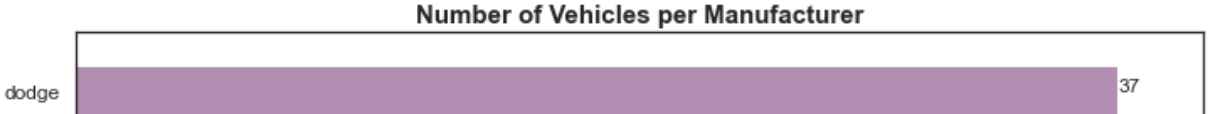
Out[127… `Text(0.5, 1.0, 'Number of Vehicles per Manufacturer')`

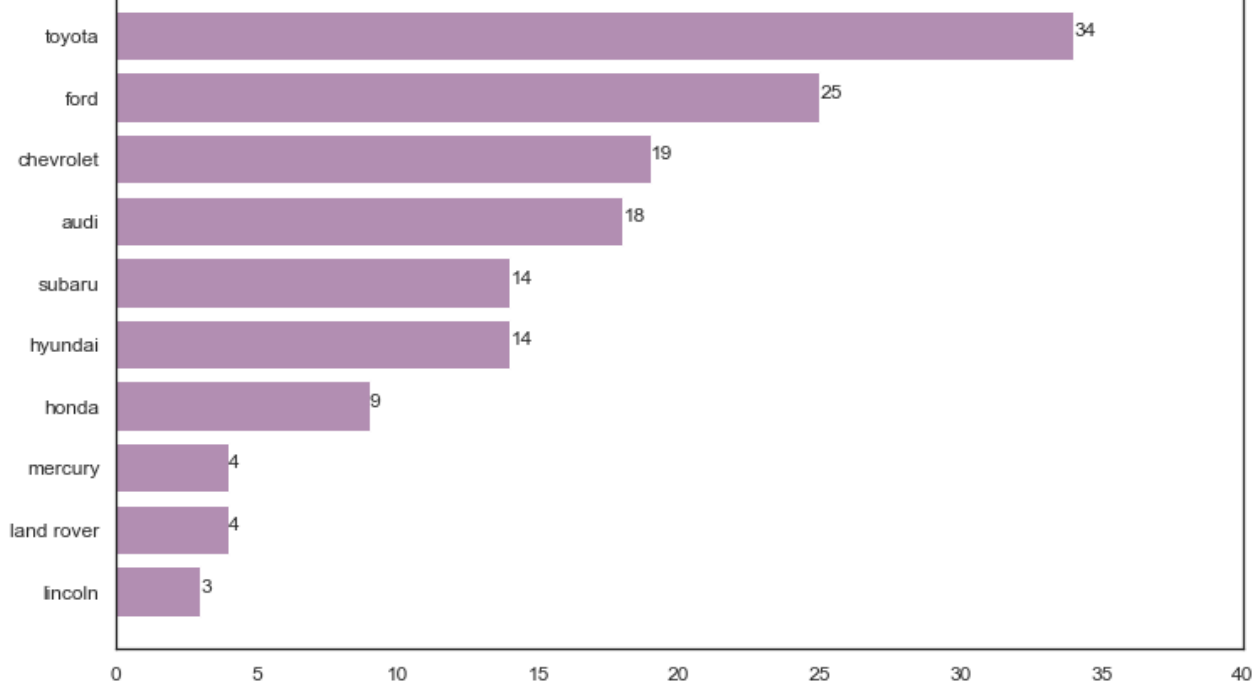


12. Horizontal bar chart

In [123…

```
value_count = value_count.sort_values()
# Vertical bar chart
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot()


# plot the data using matplotlib
plt.barh(value_count.index, value_count.values, color='#b28eb2')

for index, value in enumerate(value_count.values):
    plt.text(value, index, str(value))


# size of the x and y ticks
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)

# change the ylim
ax.set_xlim(0, 40)

# set a title for the plot
ax.set_title('Number of Vehicles per Manufacturer', fontsize=15, fontweigl
```

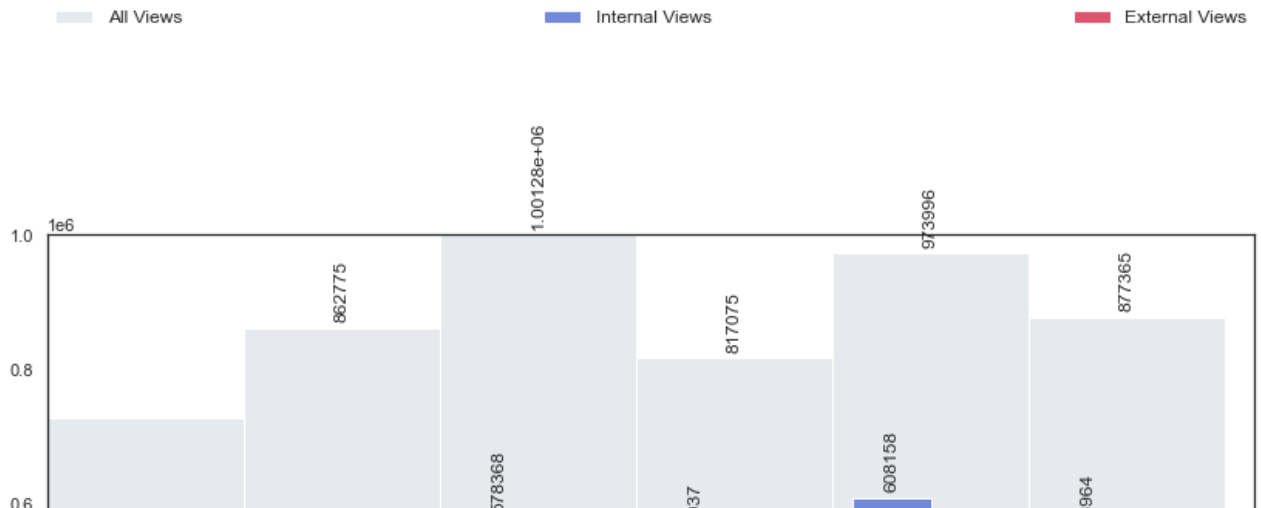


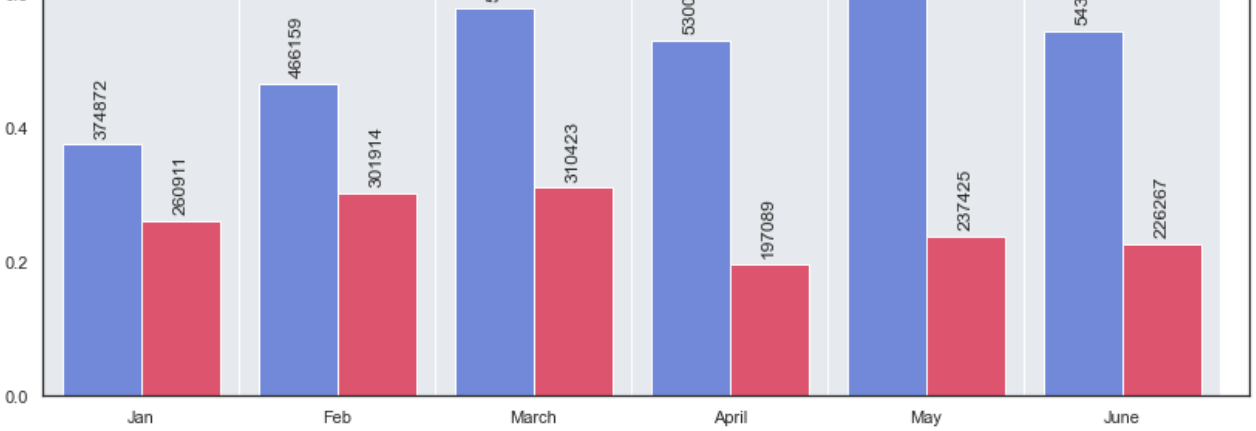


13. Multi-set Bar Chart

In [127…

```
views = pd.DataFrame(data, columns=['Month', 'All Views'])
df = pd.DataFrame(data,
                  columns=['Month', 'Internal Views', 'External Views'])

# Plotting the bars
ax = views.plot.bar(rot=0,color='#E6E9ED',width=1, figsize=(14,8))
ax = df.plot.bar(rot=0, ax=ax, color=['#7289da', '#dd546e', '#99aab5', '#
                 width=0.8, figsize=(14,8))


for p in ax.patches[1:]:
    h = p.get_height()
    x = p.get_x()+p.get_width()/2.
    if h != 0:
        ax.annotate("%g" % p.get_height(), xy=(x,h), xytext=(0,4), rotati
                    textcoords="offset points", ha="center", va="bottom")

ax.set_xlim(-0.5, None)
ax.margins(y=0)
ax.legend(ncol=len(df.columns), loc="lower left", bbox_to_anchor=(0,1.3,1
          borderaxespad=0, mode="expand", fontsize=12)
ax.set_xticklabels(df["Month"])
plt.show()
```

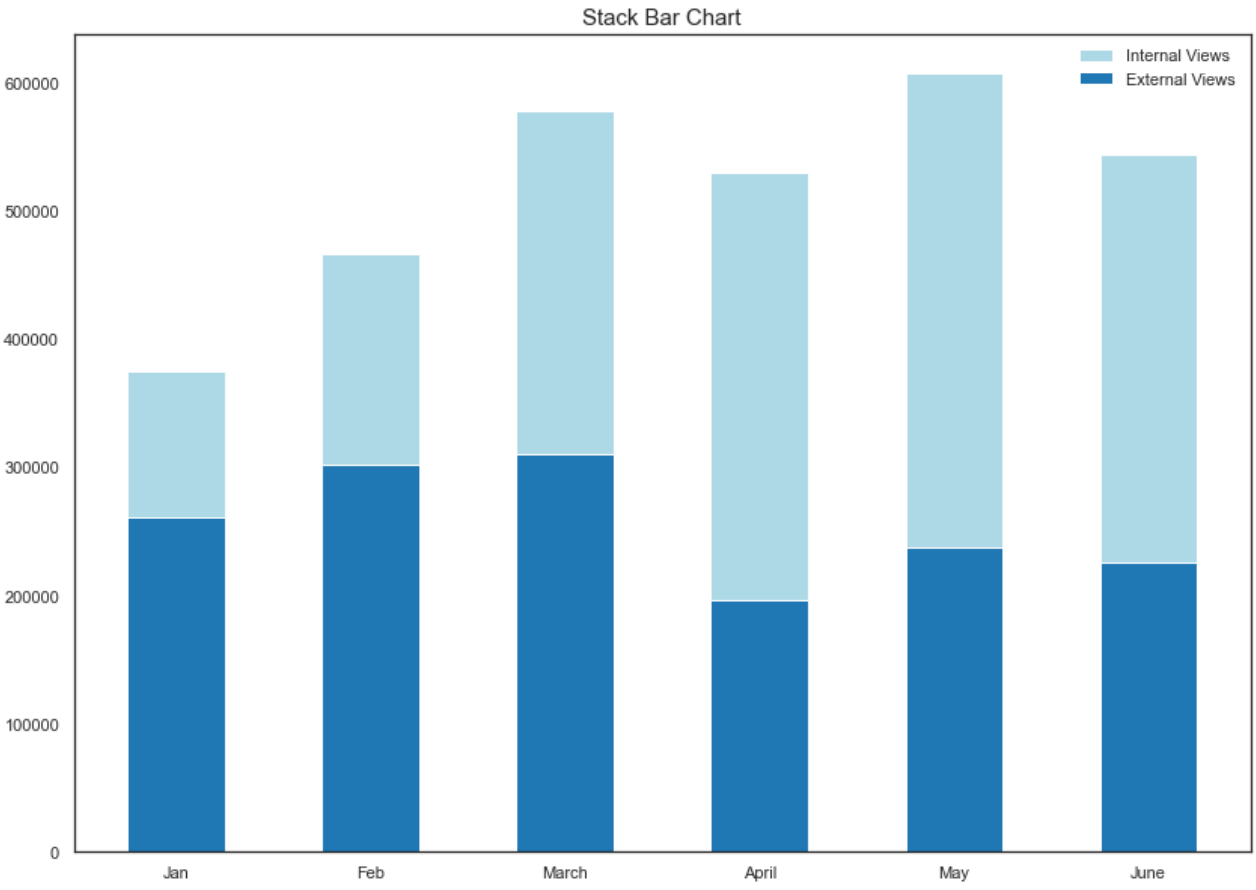




14. Stacked bar chart

In [123…

```
fig = plt.figure(figsize=(14,10))
rect1 = plt.bar(data['Month'] ,data['Internal Views'],
                width=0.5, color='lightblue')
rect2 = plt.bar(data['Month'], data['External Views'],
                width=0.5, color='#1f77b4')

plt.legend((rect1[0], rect2[0]), ('Internal Views', 'External Views'))
plt.title('Stack Bar Chart', fontsize=15)
plt.show()
```



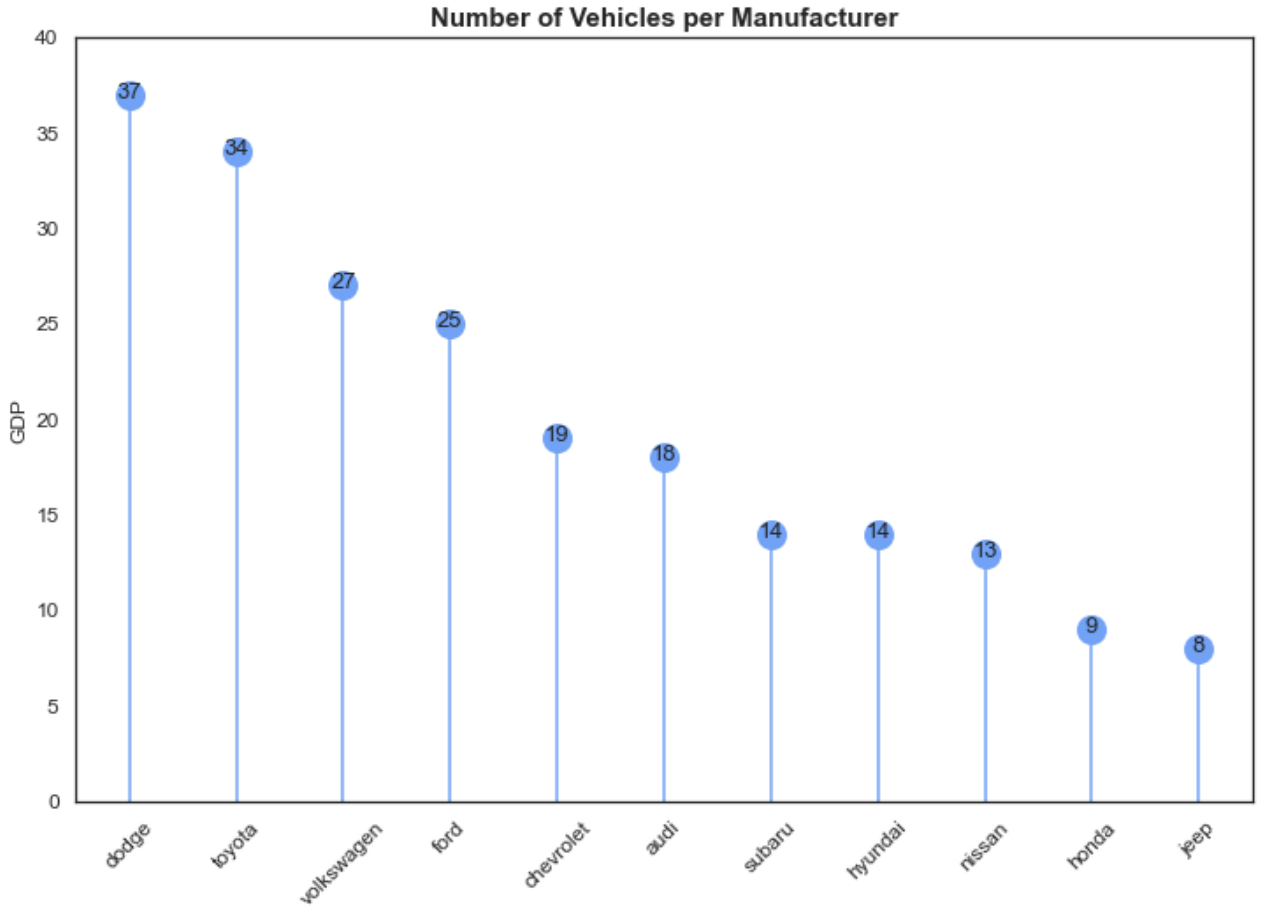
15. Lollipop Chart

In [123…

```
# Horizontal bar chart
# prepare data
value_count =  car["manufacturer"].value_counts(sort=True)
value_count = value_count[:11,]
# create n colors based on the number of labels we have
```

```
# colors = [plt.cm.Spectral(i/float(len(d.keys()))) for i in range(len(d.

# instanciate the figure
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot()

(markerline, stemlines, baseline) = plt.stem(value_count.index, value_cou

# iterate over every x and y and annotate the value on the top of the bar
for i, (k, v) in enumerate(value_count.items()):
    ax.text(k, # where to put the text on the x coordinates
            v + 0.1, # where to put the text on the y coordinates
            v, # value to text
            #color=colors[i], # color corresponding to the bar
            fontsize=13, # fontsize
            horizontalalignment='center', # center the text to be more pl
            verticalalignment='center'
           )

plt.setp(markerline, marker='o', markersize=15,
         markeredgewidth=2, color='#71a2f6')
plt.setp(stemlines, color='#71a2f6')
plt.setp(baseline, visible=False)
plt.tick_params(labelsize=12)
plt.ylabel('GDP', size=12)
plt.ylim(0, 40)
# size of the x and y ticks
plt.tick_params(axis='x', labelrotation=45, labelsize=12)
plt.tick_params(axis='y', labelsize=12)

# set a title for the plot
plt.title('Number of Vehicles per Manufacturer', fontsize=15, fontweight=
```

Out[123… Text(0.5, 1.0, 'Number of Vehicles per Manufacturer')



Distribution Plot

Distribution Plot

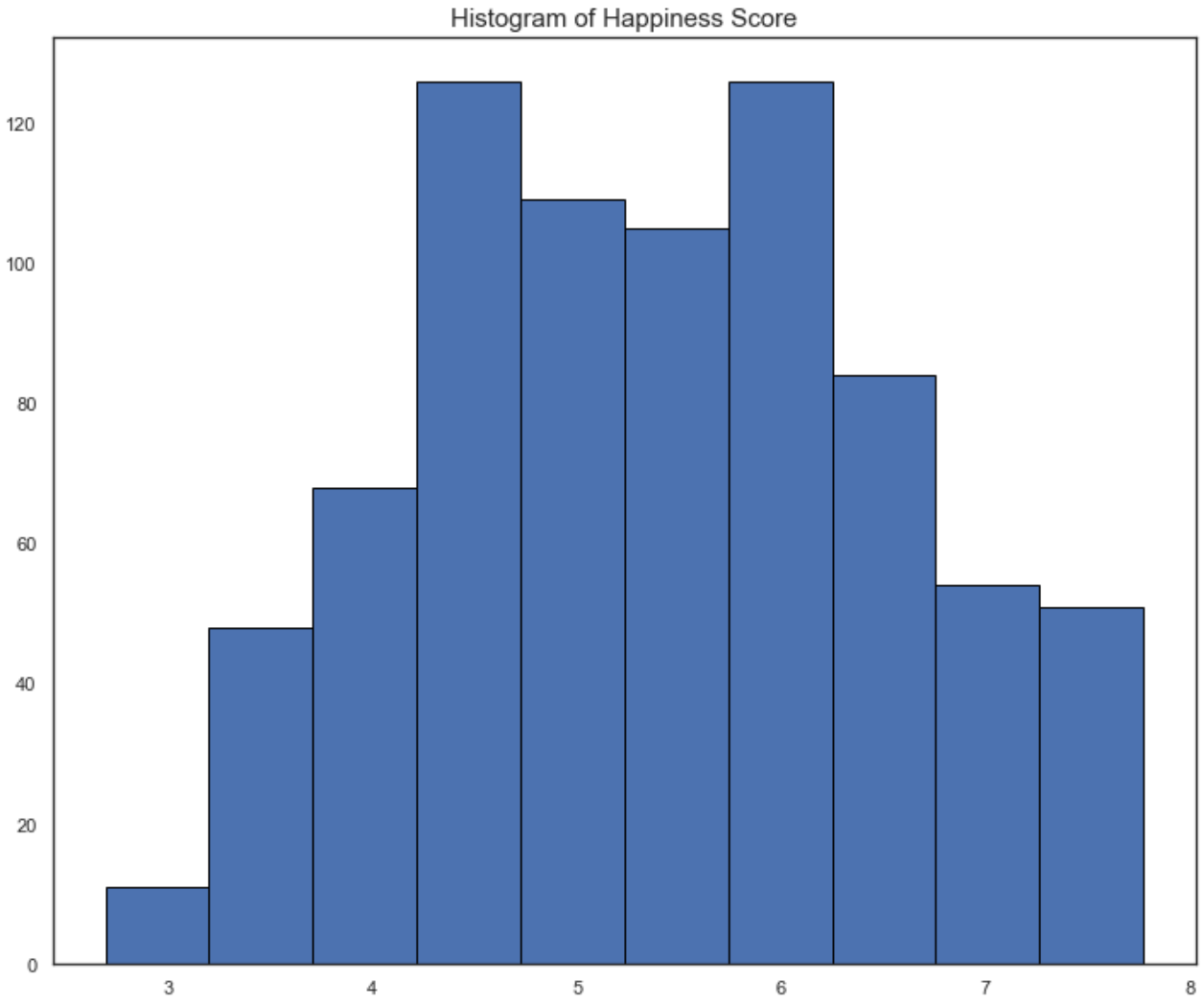
15. Histogram
16. Density Curve with Histogram
17. Density Plot
18. Box Plot
19. Strip Plot
20. Violin Plot
21. Population Pyramid

15. Histogram

In [128…

```
fig = plt.figure(figsize=(12,10))
plt.hist(happy['Score'], edgecolor = 'black')
plt.title('Histogram of Happiness Score', fontsize=15)
```

Out[128… `Text(0.5, 1.0, 'Histogram of Happiness Score')`

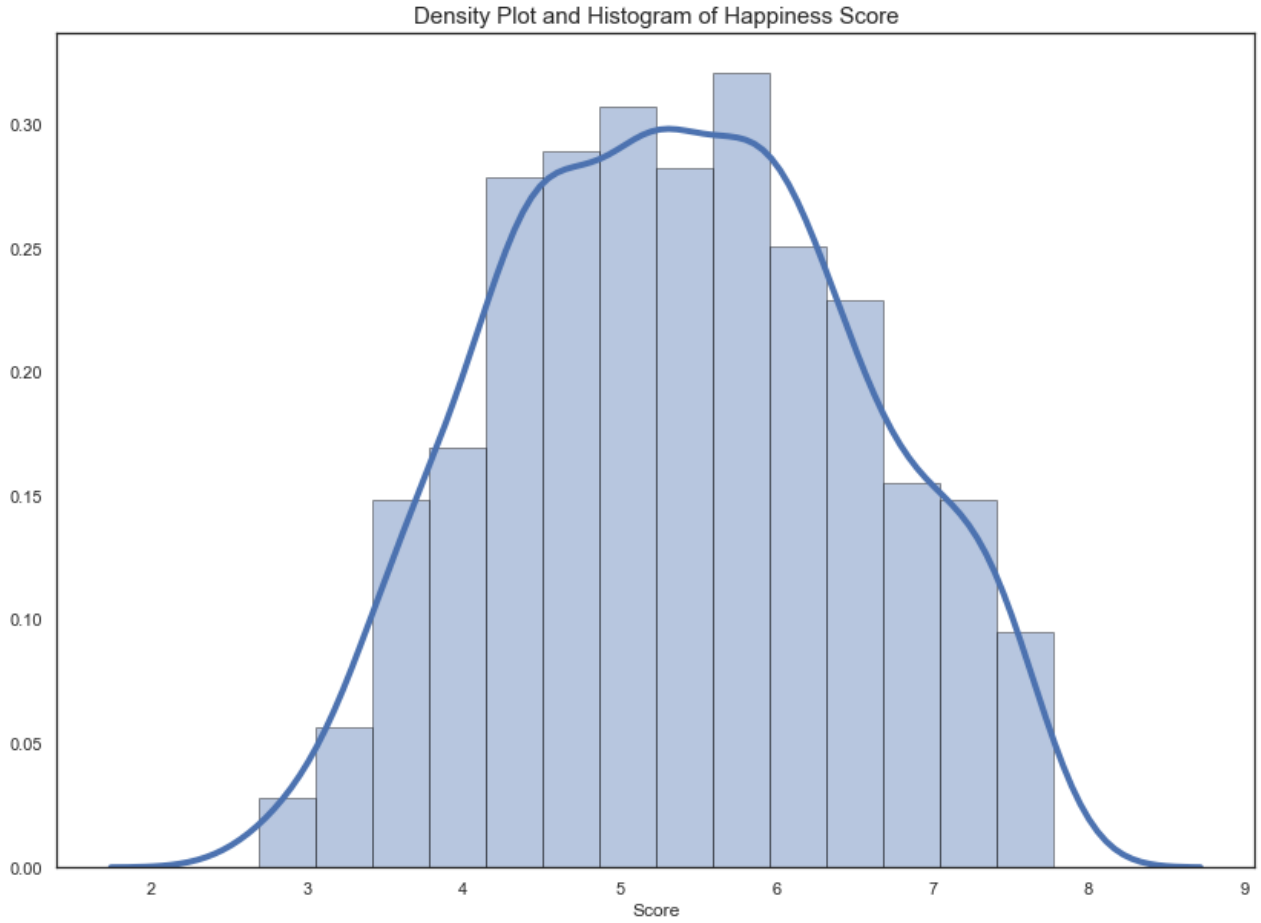


16. Density Curve with Histogram

In [123…

```
fig = plt.figure(figsize=(14,10))
sns.distplot(happy['Score'], hist=True, kde=True,
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
plt.title('Density Plot and Histogram of Happiness Score', fontsize=15)
```

Out[123… Text(0.5, 1.0, 'Density Plot and Histogram of Happiness Score')



17. Density Plot

In [128…

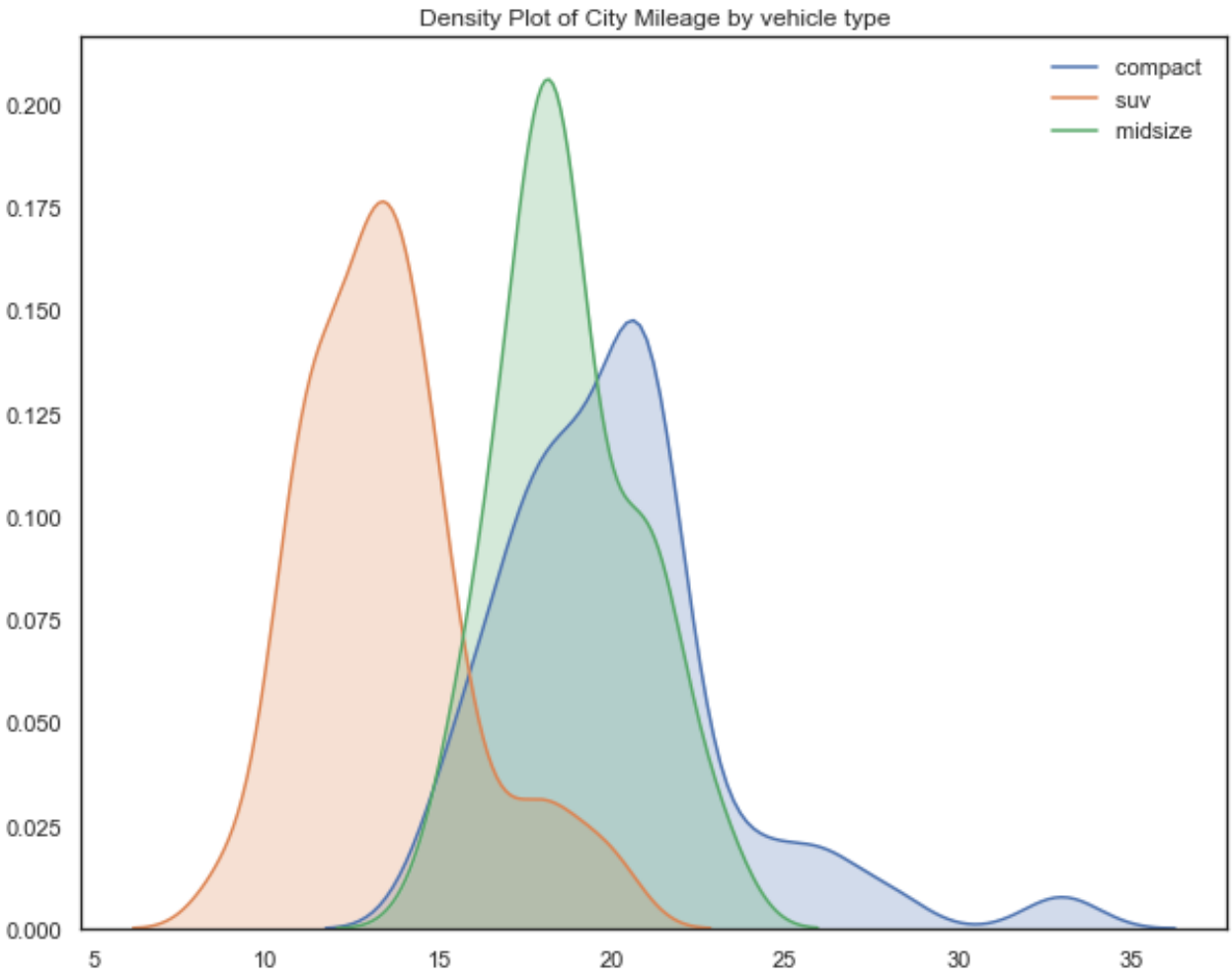
```
# simple density plot
# instanciate the figure
fig = plt.figure(figsize = (10, 8))
sns.kdeplot(car['displ'], shade=True)
```

Out[128… <AxesSubplot:>

In [124…

```
# instanciate the figure
fig = plt.figure(figsize = (10, 8))

# plot the data
# the idea is to iterate over each class
# extract their data ad plot a sepate density plot
for class_ in ['compact', 'suv', 'midsize']:
    # extract the data
    x = car[car['class'] == class_]['cty']
    # plot the data using seaborn
    sns.kdeplot(x, shade=True, label = '{} '.format(class_))

# set the title of the plot
plt.title('Density Plot of City Mileage by vehicle type')
```

Out[124… `Text(0.5, 1.0, 'Density Plot of City Mileage by vehicle type')`



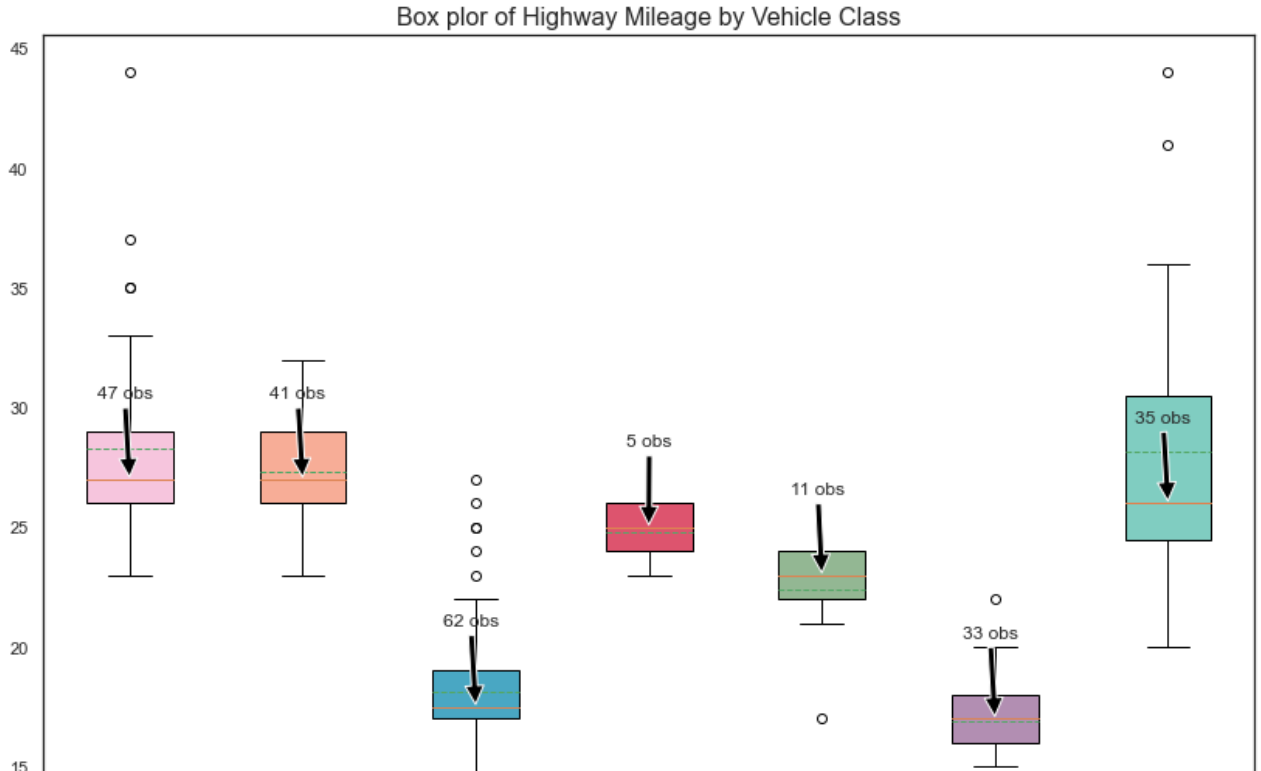
18. Box Plot

In [124…

```
# vectors to plot
vects = [car[car["class"]==cars]["hwy"] for cars in car["class"].unique()]

# labels for the x axis
labels = [class_ for class_ in car["class"].unique()]
```

```
# handpicked colors
colors = ['#f6c5dd', '#f7ad97', '#49a7c3', '#dd546e', '#93b793', '#b28eb2



# instanciate the figure
fig = plt.figure(figsize=(14, 10))
ax = fig.add_subplot()

# plot the data using matplotlib
plot1 = ax.boxplot(vects,
    notch=False, vert=True,
    meanline=True,showmeans=True,
    patch_artist=True
)

# iterate over every box and add some annotations
for box, color, vect, label, tick in zip(plot1['boxes'], # this line to i
                                         colors, vects, labels, ax.get_xt
    # change the color of the box
    box.set(facecolor=color)
    # add text
    ax.annotate("{} obs".format(len(vect)),
                xy=(tick, np.median(vect)),
               xytext=(15, 60), textcoords='offset points',
                 arrowprops=dict(facecolor='black', shrink=0.03),
            horizontalalignment='right', verticalalignment='top',
            )


# prettify the plot
# change the x labels
ax.set_xticklabels(labels=labels)

# change the rotation and the size of the x ticks (numbers of x axis)
ax.tick_params(axis='x',labelsize=12)

# set the title for the plot
ax.set_title('Box plor of Highway Mileage by Vehicle Class', fontsize=16)
```

Out[124… `Text(0.5, 1.0, 'Box plor of Highway Mileage by Vehicle Class')`

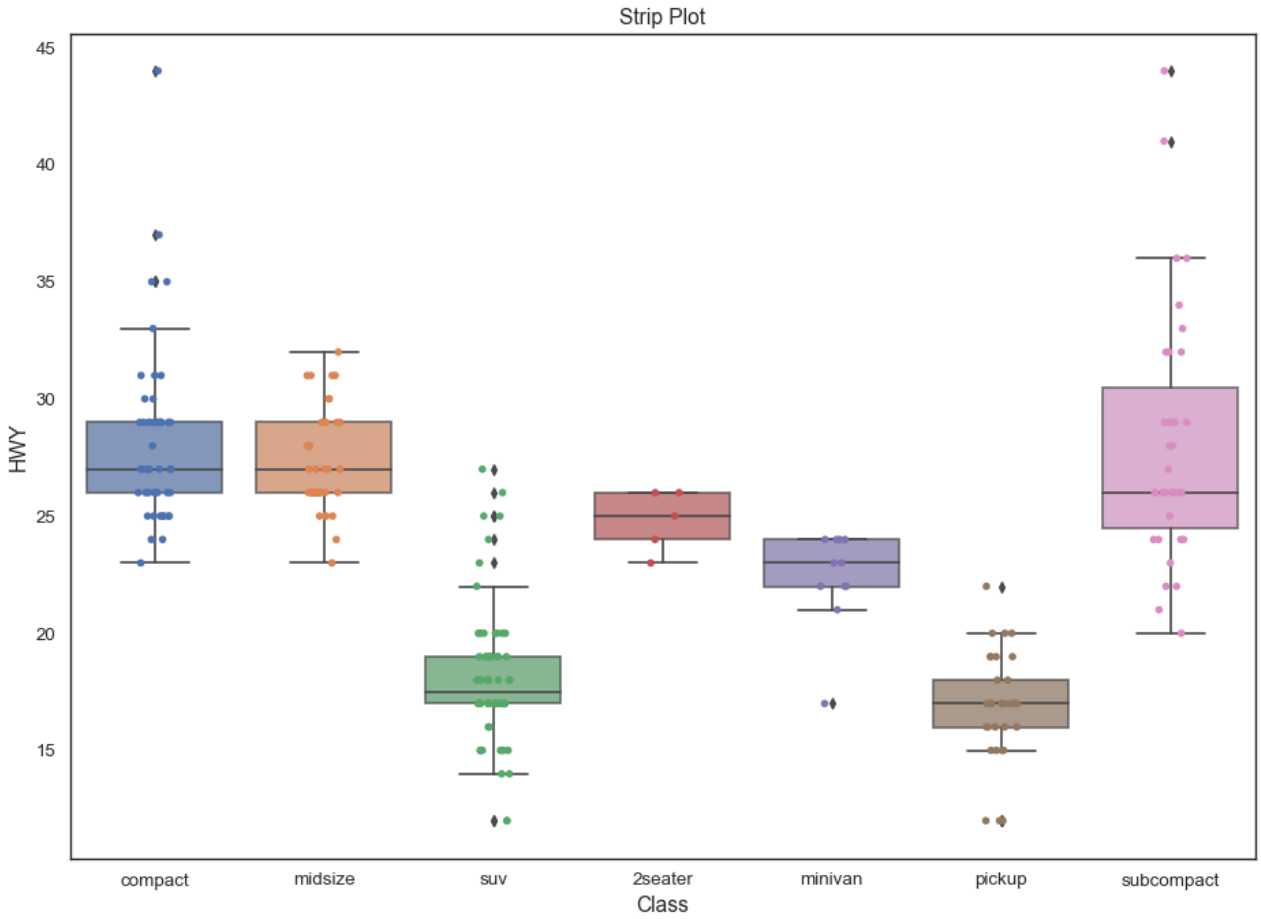


19. Strip Plot

In [124…

```
plt.figure(figsize=(14, 10), dpi=80)

ax = sns.boxplot(car['class'], car['hwy'], boxprops=dict(alpha=0.75))
ax = sns.stripplot(car['class'], car['hwy'], jitter=True, edgecolor="gray"
# change the font of the x and y ticks (numbers on the axis)
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)

# set and x and y label
ax.set_xlabel("Class", fontsize=14)
ax.set_ylabel("HWY", fontsize=14)

# set a title
ax.set_title("Strip Plot", fontsize=14)
```

Out[124… `Text(0.5, 1.0, 'Strip Plot')`



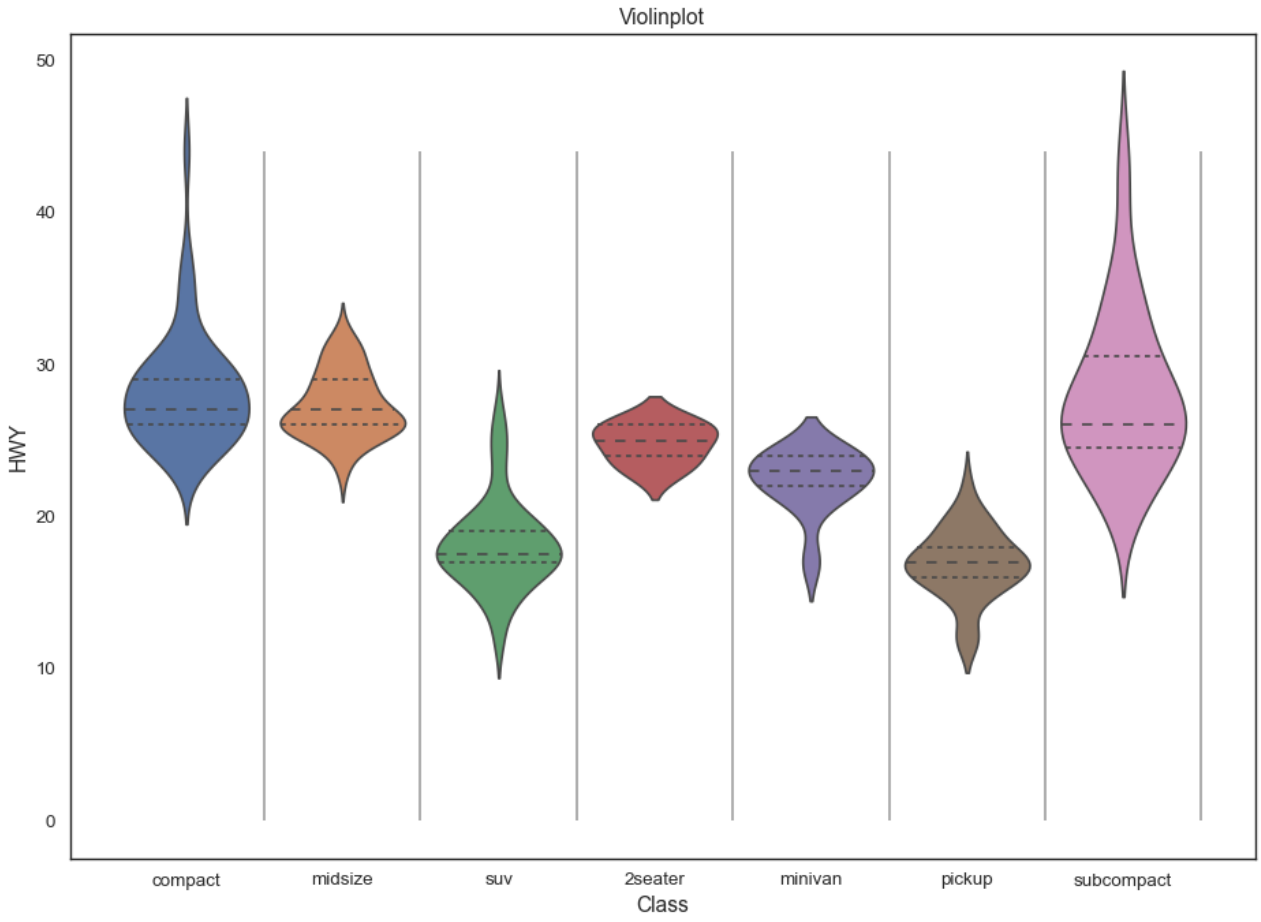
20. Violin Plot

In [124…

```
plt.figure(figsize=(14, 10), dpi=80)
sns.violinplot(car['class'], car['hwy'],
               scale='width', inner='quartile')

# prettify the plot
```

```
# get the current figure
ax = plt.gca()
# get the xticks to iterate over
xticks = ax.get_xticks()

# iterate over every xtick and add a vertical line
# to separate different classes
for tick in xticks:
    ax.vlines(tick+0.5, 0, np.max(car["hwy"]), color="grey", alpha=0.75)

# rotate the x and y ticks
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)

# add x and y label
ax.set_xlabel("Class", fontsize=14)
ax.set_ylabel("HWY", fontsize=14)

# set title
ax.set_title("Violinplot", fontsize=14)
```

Out[124… `Text(0.5, 1.0, 'Violinplot')`

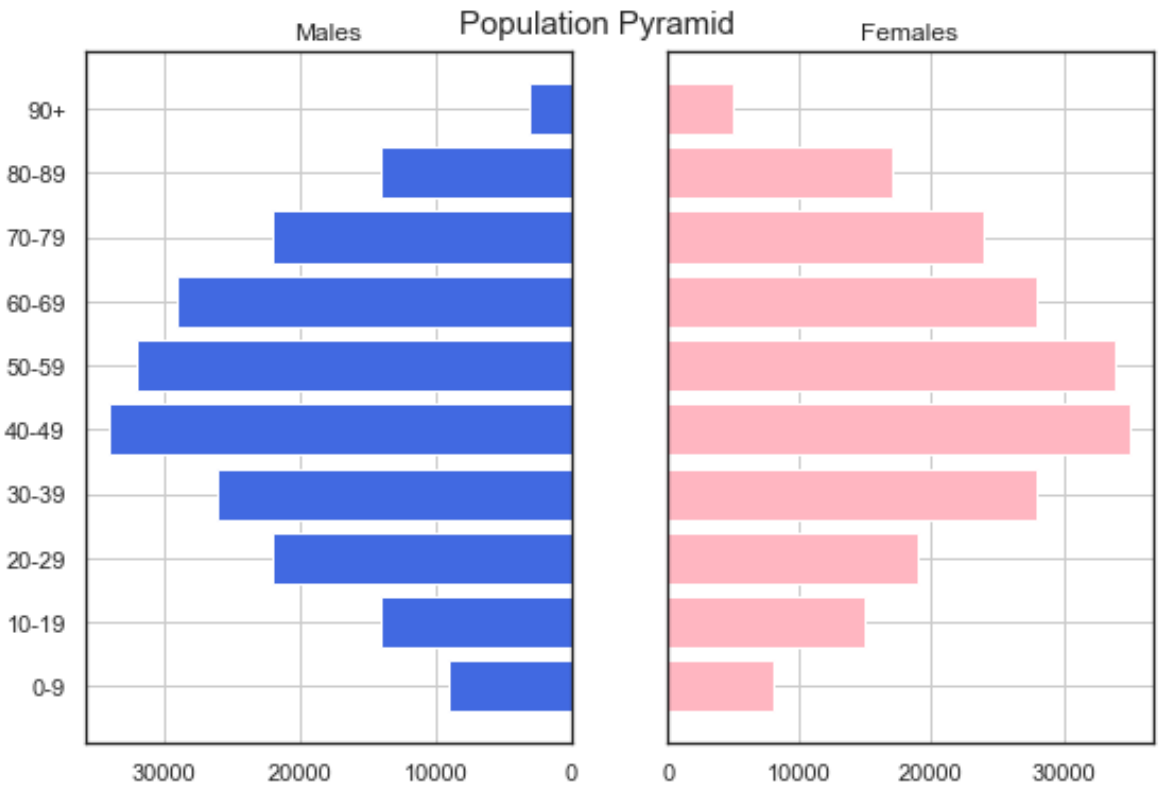


21. Population Pyramid

In [124…

```
#create dataframe
df = pd.DataFrame({'Age': ['0-9','10-19','20-29','30-39','40-49','50-59',
                   'Male': [9000, 14000, 22000, 26000, 34000, 32000, 290
                   'Female': [8000, 15000, 19000, 28000, 35000, 34000, 2


#define x and y limits
y = range(0, len(df))
x_male = df['Male']
```

```
x_female = df['Female']

#define plot parameters
fig, axes = plt.subplots(ncols=2, sharey=True, figsize=(9, 6))

#specify background color and plot title
fig.patch.set_facecolor('white')
plt.figtext(.5,.9,"Population Pyramid ", fontsize=15, ha='center')

#define male and female bars
axes[0].barh(y, x_male, align='center', color='royalblue')
axes[0].set(title='Males')
axes[1].barh(y, x_female, align='center', color='lightpink')
axes[1].set(title='Females')

#adjust grid parameters and specify labels for y-axis
axes[1].grid()
axes[0].set(yticks=y, yticklabels=df['Age'])
axes[0].invert_xaxis()
axes[0].grid()

#display plot
plt.show()
```



Comparision Plot

22. Bubble Chart
23. Bullet Chart
24. Pie Chart
25. Net Pie Chart
26. Donut Chart
27. TreeMap
28. Diverging Bar
29. Choropleth Map
30. Bubble Map

22. Bubble Chart

In [128…

```
happy = pd.read_csv('happiness_rank.csv')
'''Happiness score vs gdp per capital'''
px.scatter(happy, x="GDP", y="Score", animation_frame="Year",
           animation_group="Country",
           size="Rank", color="Country", hover_name="Country",
           trendline= "ols")
```

23. Bullet Chart

In [125…

```
# Bullet Chart
import plotly.figure_factory as ff

data = (
  {"label": "Happiness", "sublabel":"score",
   "range": [5, 6, 8], "performance": [5.5, 6.5], "point": [7]},
  {"label": "Economy", "sublabel": "score", "range": [0, 1, 2],
   "performance": [1, 1.5], "sublabel":"score","point": [1.5]},
  {"label": "Family","sublabel":"score", "range": [0, 1, 2],
   "performance": [1, 1.5],"sublabel":"score", "point": [1.3]},
  {"label": "Freedom","sublabel":"score", "range": [0, 0.3, 0.6],
   "performance": [0.3, 0.4],"sublabel":"score", "point": [0.5]},
  {"label": "Trust", "sublabel":"score","range": [0, 0.2, 0.5],
   "performance": [0.3, 0.4], "point": [0.4]}
)


fig = ff.create_bullet(
    data, titles='label', subtitles='sublabel', markers='point',
    measures='performance', ranges='range', orientation='h',
    measure_colors=['#1e747c', '#7ac7bf'],
    range_colors=['#F5E1DA', '#F1F1F1']
   )
py.iplot(fig, filename='bullet chart from dict')
```

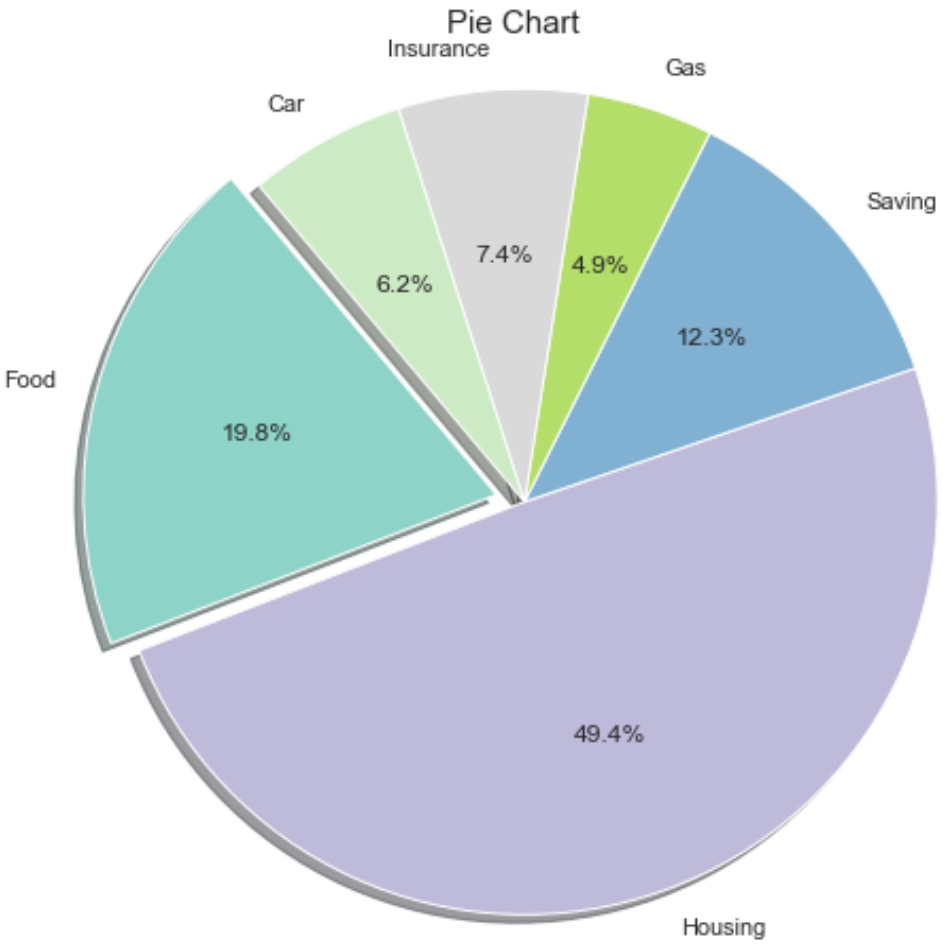
24. Pie Chart

In [125…

```
fig = plt.figure(figsize=(8,8))
labels = 'Food', 'Housing', 'Saving', 'Gas', 'Insurance', 'Car'
spend = [800, 2000, 500, 200, 300, 250]

# create n colors based on the number of labels we have
colors = [plt.cm.Set3(i/float(len(labels))) for i in range(len(labels))]

p = plt.pie(spend, # Value
            labels=labels, # Labels for each sections
            explode=(0.07, 0, 0, 0, 0, 0), # To slice the perticuler sect
            colors=colors, # Color of each section
            autopct='%1.1f%%', # Show data in persentage for with 1 decim
            startangle=130, # Start angle of first section
```

```
            shadow=True # Showing shadow of pie chart
           )


plt.axis('equal')

plt.title('Pie Chart', fontsize=15)
```

Out[125… Text(0.5, 1.0, 'Pie Chart')



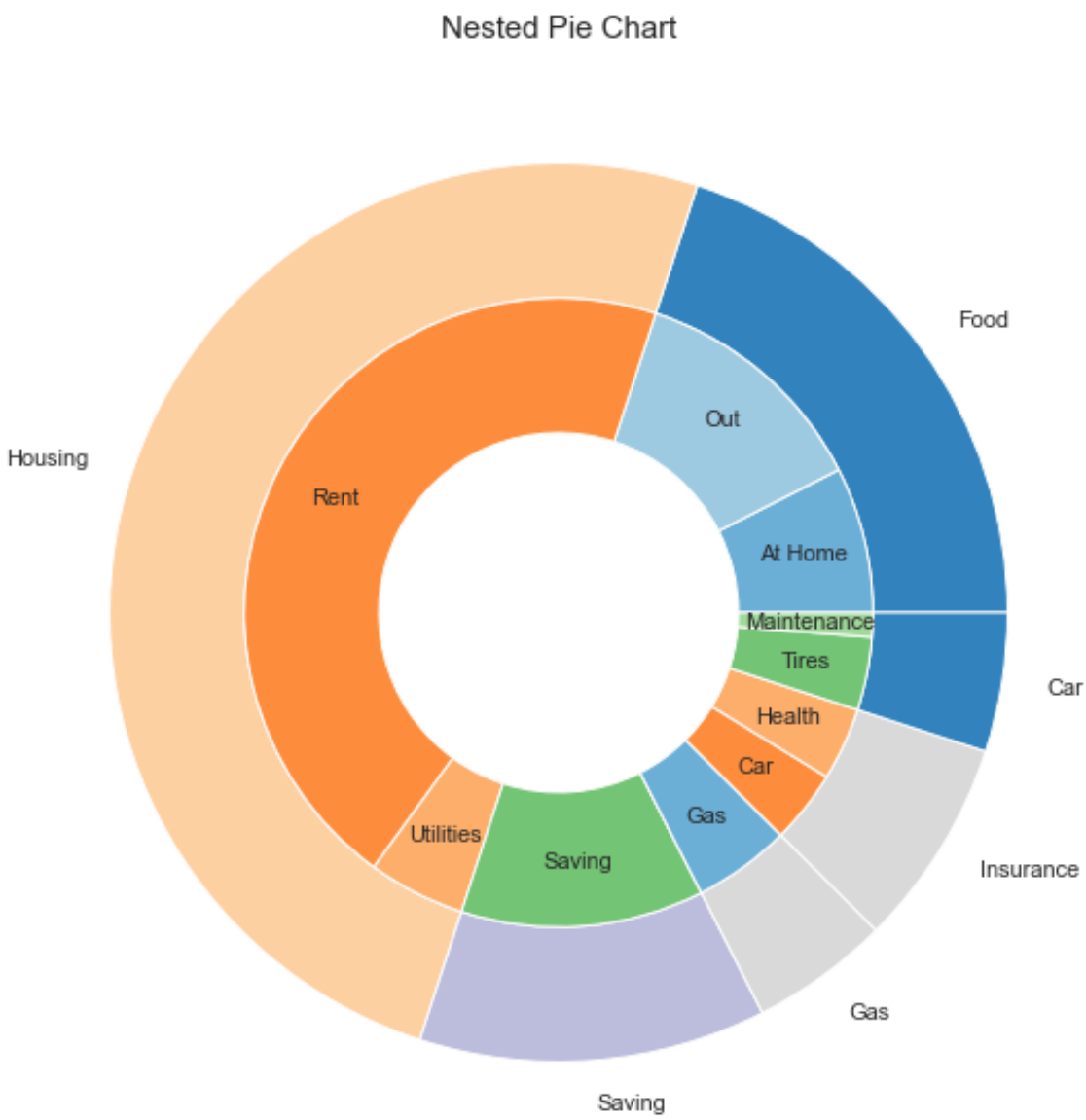
25. Net Pie Chart

In [125…

```
# get the data
size = 0.3
labels = 'Food', 'Housing', 'Saving', 'Gas', 'Insurance', 'Car'
spend = [800, 2000, 500, 200, 300, 250]
vals = np.array([[300., 500.], [1800., 200.], [500., 0.],[200., 0.], [150
in_labels = 'At Home','Out', 'Rent','Utilities','Saving','', 'Gas','','Ca
# create the outer and inner colors
cmap = plt.get_cmap("tab20c")
outer_colors = cmap(np.arange(5)*7)
inner_colors = cmap(np.array([1, 2, 5, 6, 9, 10]))

# instanciate the figure
fig = plt.figure(figsize=(14, 10))
ax = fig.add_subplot()

# plot the data
# outer level
ax.pie(vals.sum(axis=1), # plot the total [60., 32.] = 92
       radius=1, # Radius to increase or decrease the size of pie chart
       labels=labels, # Labels for each sections
       colors=outer_colors, # Color of each section
```

```
        wedgeprops=dict(width=size, edgecolor='w') # Add edges to each por
       )

# inner level
patches, texts = ax.pie(vals.flatten(), # using flatten we plot 60, 32 se
       radius=1-size,
       labels=in_labels,
       labeldistance=0.8,
       colors=inner_colors,
       wedgeprops=dict(width=size, edgecolor='w'))

for t in texts:
    t.set_horizontalalignment('center')


# set the title for the plot
plt.title('Nested Pie Chart', fontsize=15)
```

Out[125… Text(0.5, 1.0, 'Nested Pie Chart')



26. Donut Chart

In [125…

```
df15 = pd.read_csv('2015.csv')
d2015 = df15['Region'].value_counts()

label_d2015 = d2015.index
```

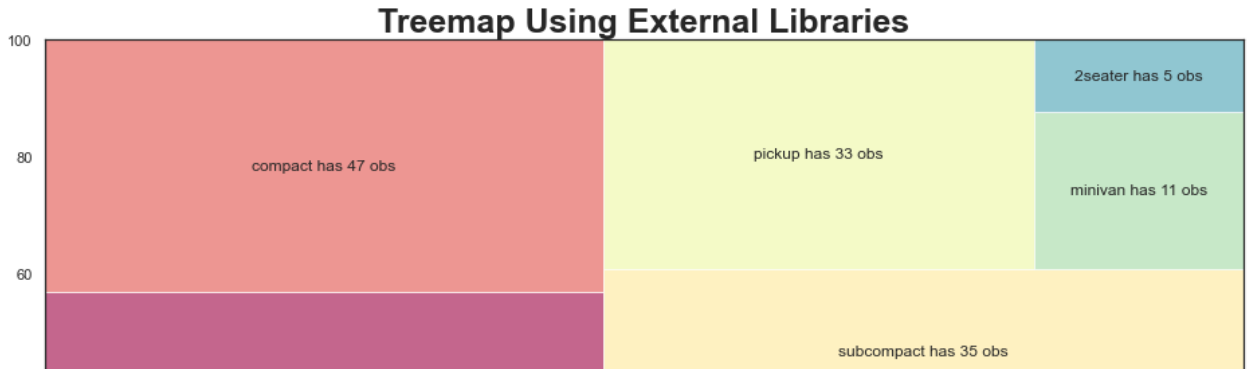
```
size_d2015 = d2015.values


colors = ['#4870a0', '#eb96aa', '#7ac7bf', '#b28eb2', '#d9a3d8',
          '#f3c366','#a1cdec', '#38b6ff','#92406e','#f5907b']

trace = go.Pie( labels=label_d2015,
                values=size_d2015,
                marker=dict(colors = colors),
                name='2015',
                hole=0.3)

data = [trace]

layout1 = go.Layout(
           title='Regions')

fig = go.Figure(data=data, layout=layout1)

py.iplot(fig)
```

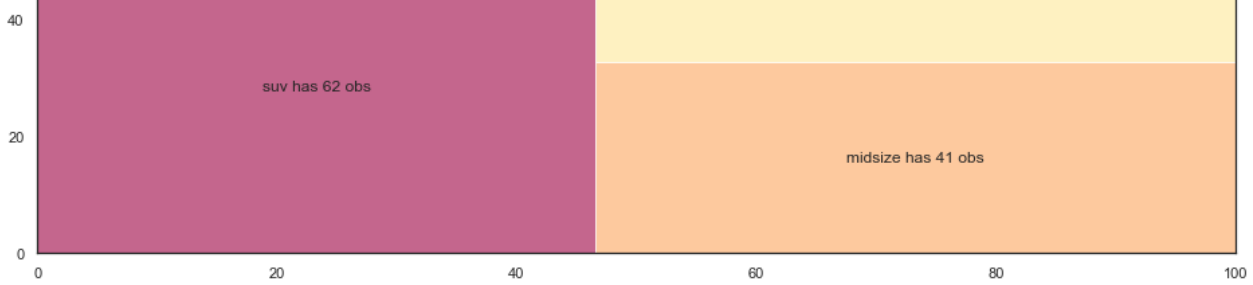
27. Tree Map

In [120…

```
import squarify

# prepare the data for plotting
# get the values
label_value = car['class'].value_counts().to_dict()

# create the labels using a list comprehesion
labels = ['{} has {} obs'.format(class_, obs) for class_, obs in label_va

# create n colors based on the number of labels we have
colors = [plt.cm.Spectral(i/float(len(labels))) for i in range(len(labels


# instanciate the figure
fig = plt.gcf()
fig.set_size_inches(16,8)

# plot the data using squarify
squarify.plot(sizes=label_value.values(), label=labels, color=colors, alp

# prettify the plot
# add a title to the plot
plt.title('Treemap Using External Libraries', fontsize=25, fontweight='bo
```

Out[120… Text(0.5, 1.0, 'Treemap Using External Libraries')





28. Diverging Bar Chart

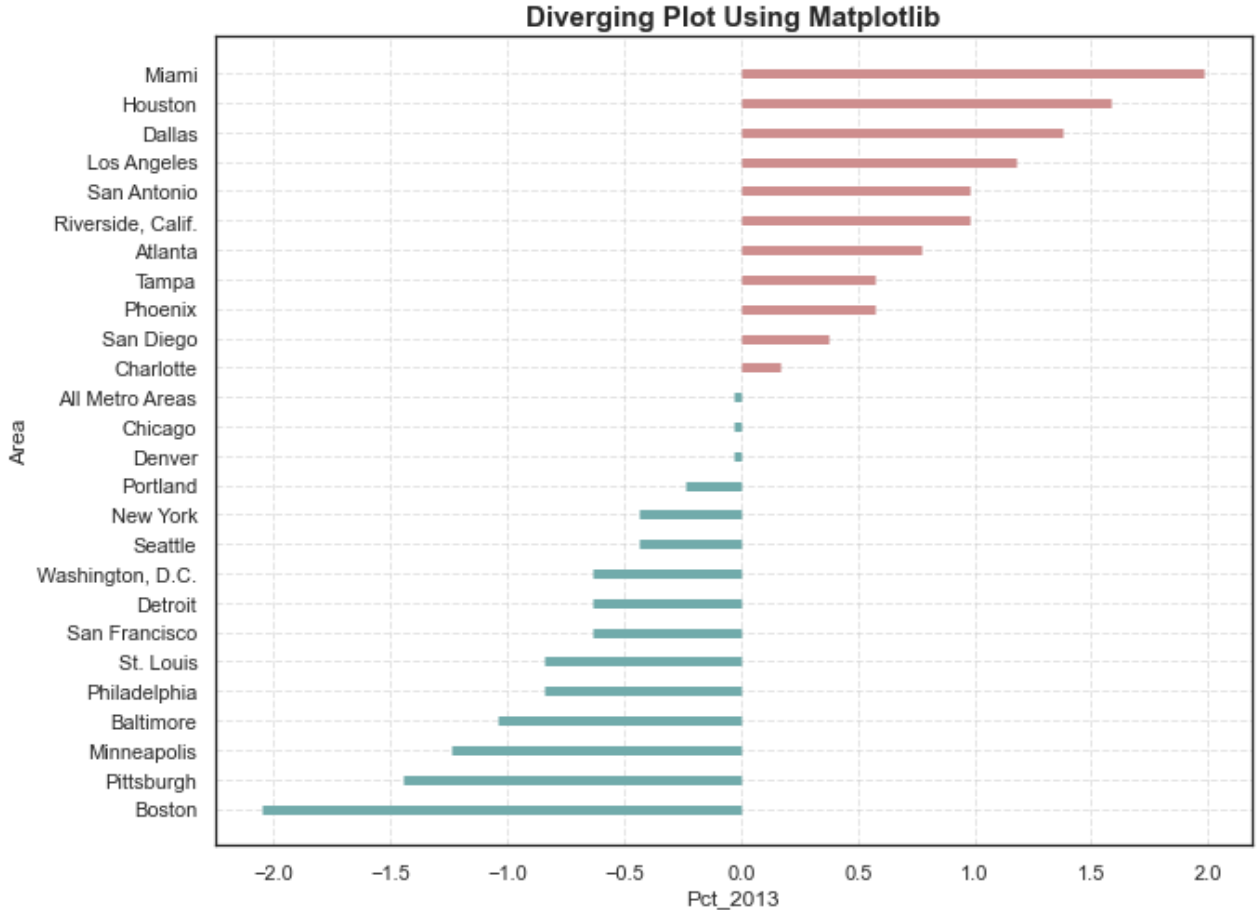
In [121…

```
health = pd.read_csv('health.csv')
# prepare the data for plotting
# here we standarize the data
# More info:
# https://statisticsbyjim.com/glossary/standardization/
health['x_plot'] = (health['pct_2013'] - health['pct_2013'].mean())/healt

# sort value and reset the index
health.sort_values('x_plot', inplace = True)
health.reset_index(inplace = True)

# create a color list, where if value is above > 0 it's green otherwise r
colors = ['#71abab' if x < 0 else '#ce8e8e' for x in health['x_plot']]

# instanciate the figure
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot()
# plot using horizontal lines and make it look like a column by changing
ax.hlines(y=health.index, xmin=0 , xmax=health['x_plot'],  color=colors,

# prettify the plot
# set x and y axis
ax.set_xlabel('Pct_2013')
ax.set_ylabel('Area')

# set a title
ax.set_title('Diverging Plot Using Matplotlib', fontsize=15, fontweight='

# make a grid to help separate the lines
ax.grid(linestyle='--', alpha=0.5)

# change the y ticks
# first you set the yticks
# then you change them using the area names

plt.yticks(health.index, health.Area)
```

Out[121…

```
([<matplotlib.axis.YTick at 0x7f9bb0aee730>,
  <matplotlib.axis.YTick at 0x7f9bb0b108e0>,
  <matplotlib.axis.YTick at 0x7f9bb0b10970>,
  <matplotlib.axis.YTick at 0x7f9bad384a90>,
  <matplotlib.axis.YTick at 0x7f9bad384fa0>,
  <matplotlib.axis.YTick at 0x7f9bad3904f0>,
  <matplotlib.axis.YTick at 0x7f9bad390a00>,
  <matplotlib.axis.YTick at 0x7f9bad390f10>,
  <matplotlib.axis.YTick at 0x7f9bad395460>,
  <matplotlib.axis.YTick at 0x7f9bad395970>,
  <matplotlib.axis.YTick at 0x7f9bad390610>,
  <matplotlib.axis.YTick at 0x7f9bad3846a0>,
  <matplotlib.axis.YTick at 0x7f9bad395190>,
```

```
 <matplotlib.axis.YTick at 0x7f9bad39c220>,
 <matplotlib.axis.YTick at 0x7f9bad39c6d0>,
 <matplotlib.axis.YTick at 0x7f9bad39cbe0>,
 <matplotlib.axis.YTick at 0x7f9bb0a51130>,
 <matplotlib.axis.YTick at 0x7f9bb0a51640>,
 <matplotlib.axis.YTick at 0x7f9bb0a51b50>,
 <matplotlib.axis.YTick at 0x7f9bb0a51880>,
 <matplotlib.axis.YTick at 0x7f9bad39c910>,
 <matplotlib.axis.YTick at 0x7f9bad395c40>,
 <matplotlib.axis.YTick at 0x7f9bb0a5a250>,
 <matplotlib.axis.YTick at 0x7f9bb0a5a760>,
 <matplotlib.axis.YTick at 0x7f9bb0a5ac70>,
 <matplotlib.axis.YTick at 0x7f9bb0a5f1c0>],
[Text(0, 0, 'Boston'),
 Text(0, 1, 'Pittsburgh'),
 Text(0, 2, 'Minneapolis'),
 Text(0, 3, 'Baltimore'),
 Text(0, 4, 'Philadelphia'),
 Text(0, 5, 'St. Louis'),
 Text(0, 6, 'San Francisco'),
 Text(0, 7, 'Detroit'),
 Text(0, 8, 'Washington, D.C.'),
 Text(0, 9, 'Seattle'),
 Text(0, 10, 'New York'),
 Text(0, 11, 'Portland'),
 Text(0, 12, 'Denver'),
 Text(0, 13, 'Chicago'),
 Text(0, 14, 'All Metro Areas'),
 Text(0, 15, 'Charlotte'),
 Text(0, 16, 'San Diego'),
 Text(0, 17, 'Phoenix'),
 Text(0, 18, 'Tampa'),
 Text(0, 19, 'Atlanta'),
 Text(0, 20, 'Riverside, Calif.'),
 Text(0, 21, 'San Antonio'),
 Text(0, 22, 'Los Angeles'),
 Text(0, 23, 'Dallas'),
 Text(0, 24, 'Houston'),
 Text(0, 25, 'Miami')])
```



29. World Map

In [125…

```
'''World Map
Happiness Rank Accross the World'''

happiness_rank = dict(type='choropleth',
           locations=happy['Country'],
           locationmode='country names',
           z=happy['Rank'],
           text=happy['Country'],
           colorscale='bluyl',
           autocolorscale=False,
           reversescale=True,
           marker_line_color='darkgray',
           marker_line_width=0.5)
layout = dict(title='Happiness Rank Across the World',
             geo=dict(showframe=False,
                      projection={'type': 'equirectangular'}))
world_map_1 = go.Figure(data=[happiness_rank], layout=layout)
iplot(world_map_1)
```

30. Bubble Map

In [128…

```
# to use unverified ssl
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
covid = pd.read_csv('https://opendata.ecdc.europa.eu/covid19/casedistribu
covid.head(10)
```

Out[128…

	dateRep	day	month	year	cases	deaths	countriesAndTerritories	geoId	c
0	03/09/2020	3	9	2020	38	3	Afghanistan	AF	
1	02/09/2020	2	9	2020	9	0	Afghanistan	AF	
2	01/09/2020	1	9	2020	34	4	Afghanistan	AF	
3	31/08/2020	31	8	2020	19	0	Afghanistan	AF	
4	30/08/2020	30	8	2020	3	0	Afghanistan	AF	
5	29/08/2020	29	8	2020	11	1	Afghanistan	AF	
6	28/08/2020	28	8	2020	3	0	Afghanistan	AF	
7	27/08/2020	27	8	2020	55	4	Afghanistan	AF	
8	26/08/2020	26	8	2020	1	0	Afghanistan	AF	
9	25/08/2020	25	8	2020	71	10	Afghanistan	AF	

In [128…

```
# Remove unuseful columns
covid = covid[['dateRep', 'cases', 'deaths', 'countriesAndTerritories', '
# Rename columns
```

```
covid = covid.rename(columns={
    'dateRep': 'date',
    'countriesAndTerritories': 'country',
    'countryterritoryCode': 'countryCode',
    'continentExp': 'continent'
})
# Convert string to datetime
covid['date'] = pd.to_datetime(covid['date'], format='%d/%m/%Y')
# Preview the data frame
covid.sample(10)
```

Out[128…

	date	cases	deaths	country	countryCode	continent
26975	2020-06-25	0	0	Nicaragua	NIC	America
34080	2020-08-07	20	1	South_Korea	KOR	Asia
37725	2020-04-18	0	0	Uganda	UGA	Africa
317	2020-06-16	69	0	Albania	ALB	Europe
16775	2020-06-06	0	0	Holy_See	VAT	Europe
11121	2020-04-29	1018	208	Ecuador	ECU	America
13375	2020-07-16	927	91	France	FRA	Europe
4614	2020-07-27	1148	48	Bolivia	BOL	America
15013	2020-06-04	0	0	Greece	GRC	Europe
35626	2020-05-28	0	0	Switzerland	CHE	Europe

In [129…

```
from datetime import datetime
# Get today as string
today = datetime.now().strftime('%Y-%m-%d')
# Get a data frame only for today
df_today = covid[covid.date == today]
# Preview the data frame
df_today.head()
```

Out[129…

	date	cases	deaths	country	countryCode	continent
0	2020-09-03	38	3	Afghanistan	AFG	Asia
238	2020-09-03	122	6	Albania	ALB	Europe
417	2020-09-03	325	7	Algeria	DZA	Africa
660	2020-09-03	15	0	Andorra	AND	Europe
834	2020-09-03	75	1	Angola	AGO	Africa

In [129…

```
import plotly.express as px
fig = px.scatter_geo(
    df_today, # provide the Pandas data frame
    locations='countryCode', # indicate locations
    color='continent',
    hover_name='country', # what to display when the mouse hovering on th
    size='cases', # how large the bubble is
    projection='equirectangular',
    title=f'World COVID-19 Cases for {today}'
)
fig.show()
```

datacamp

Python For Data Science
Seaborn Cheat Sheet

Learn Seaborn online at www.DataCamp.com

Statistical Data Visualization With Seaborn

The Python visualization library **Seaborn** is based on matplotlib and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

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

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot
5. Show your plot

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips") #Step 1
>>> sns.set_style("whitegrid") #Step 2
>>> g = sns.lmplot(x="tip", #Step 3
                   y="total_bill",
                   data=tips,
                   aspect=2)
>>> g = (g.set_axis_labels("Tip","Total bill(USD)").
set(xlim=(0,10),ylim=(0,100)))
>>> plt.title("title") #Step 4
>>> plt.show(g) #Step 5
```

1 Data

Also see Lists, NumPy & Pandas

```
>>> import pandas as pd
>>> import numpy as np
>>> uniform_data = np.random.rand(10, 12)
>>> data = pd.DataFrame({'x':np.arange(1,101),
                         'y':np.random.normal(0,4,100)})
```

Seaborn also offers built-in data sets:

```
>>> titanic = sns.load_dataset("titanic")
>>> iris = sns.load_dataset("iris")
```

2 Figure Aesthetics

Also see Matplotlib

```
>>> f, ax = plt.subplots(figsize=(5,6)) #Create a figure and one subplot
```

Seaborn styles

```
>>> sns.set() #(Re)set the seaborn default
>>> sns.set_style("whitegrid") #Set the matplotlib parameters
>>> sns.set_style("ticks", #Set the matplotlib parameters
                  {"xtick.major.size":8,
                   "ytick.major.size":8})
#Return a dict of params or use with with to temporarily set the style
>>> sns.axes_style("whitegrid")
```

Context Functions

```
>>> sns.set_context("talk") #Set context to "talk"
>>> sns.set_context("notebook", #Set context to "notebook",
                    font_scale=1.5, #Scale font elements and
                    rc={"lines.linewidth":2.5}) #override param mapping
```

Color Palette

```
>>> sns.set_palette("husl",3) #Define the color palette
>>> sns.color_palette("husl") #Use with with to temporarily set palette
>>> flatui = ["#9b59b6","#3498db","#95a5a6","#e74c3c","#34495e","#2ecc71"]
>>> sns.set_palette(flatui) #Set your own color palette
```

3 Plotting With Seaborn

Axis Grids

```
>>> g = sns.FacetGrid(titanic, #Subplot grid for plotting conditional relationships
                       col="survived",
                       row="sex")
>>> g = g.map(plt.hist,"age")
>>> sns.factorplot(x="pclass", #Draw a categorical plot onto a Facetgrid
                   y="survived",
                   hue="sex",
                   data=titanic)
>>> sns.lmplot(x="sepal_width", #Plot data and regression model fits across a FacetGrid
               y="sepal_length",
               hue="species",
               data=iris)
>>> h = sns.PairGrid(iris) #Subplot grid for plotting pairwise relationships
>>> h = h.map(plt.scatter)
>>> sns.pairplot(iris) #Plot pairwise bivariate distributions
>>> i = sns.JointGrid(x="x", #Grid for bivariate plot with marginal univariate plots
                      y="y",
                      data=data)
>>> i = i.plot(sns.regplot,
               sns.distplot)
>>> sns.jointplot("sepal_length", #Plot bivariate distribution
                  "sepal_width",
                  data=iris,
                  kind='kde')
```

4 Further Customizations

Also see Matplotlib

Axisgrid Objects

```
>>> g.despine(left=True) #Remove left spine
>>> g.set_ylabels("Survived") #Set the labels of the y-axis
>>> g.set_xticklabels(rotation=45) #Set the tick labels for x
>>> g.set_axis_labels("Survived", #Set the axis labels
                      "Sex")
>>> h.set(xlim=(0,5), #Set the limit and ticks of the x-and y-axis
          ylim=(0,5),
          xticks=[0,2.5,5],
          yticks=[0,2.5,5])
```

Plot

```
>>> plt.title("A Title") #Add plot title
>>> plt.ylabel("Survived") #Adjust the label of the y-axis
>>> plt.xlabel("Sex") #Adjust the label of the x-axis
>>> plt.ylim(0,100) #Adjust the limits of the y-axis
>>> plt.xlim(0,10) #Adjust the limits of the x-axis
>>> plt.setp(ax,yticks=[0,5]) #Adjust a plot property
>>> plt.tight_layout() #Adjust subplot params
```

Regression Plots

```
>>> sns.regplot(x="sepal_width", #Plot data and a linear regression model fit
                y="sepal_length",
                data=iris,
                ax=ax)
```

Distribution Plots

```
>>> plot = sns.distplot(data.y, #Plot univariate distribution
                        kde=False,
                        color="b")
```

Matrix Plots

```
>>> sns.heatmap(uniform_data,vmin=0,vmax=1) #Heatmap
```

Categorical Plots

Scatterplot

```
>>> sns.stripplot(x="species", #Scatterplot with one categorical variable
                  y="petal_length",
                  data=iris)
>>> sns.swarmplot(x="species", #Categorical scatterplot with non-overlapping points
                  y="petal_length",
                  data=iris)
```

Bar Chart

```
>>> sns.barplot(x="sex", #Show point estimates & confidence intervals with scatterplot glyphs
                y="survived",
                hue="class",
                data=titanic)
```

Count Plot

```
>>> sns.countplot(x="deck", #Show count of observations
                  data=titanic,
                  palette="Greens_d")
```

Point Plot

```
>>> sns.pointplot(x="class", #Show point estimates & confidence intervals as rectangular bars
                  y="survived",
                  hue="sex",
                  data=titanic,
                  palette={"male":"g",
                  "female":"m"},
                  markers=["^","o"],
                  linestyles=["-","--"])
```

Boxplot

```
>>> sns.boxplot(x="alive", #Boxplot
                y="age",
                hue="adult_male",
                data=titanic)
>>> sns.boxplot(data=iris,orient="h") #Boxplot with wide-form data
```

Violinplot

```
>>> sns.violinplot(x="age", #Violin plot
                   y="sex",
                   hue="survived",
                   data=titanic)
```

5 Show or Save Plot

Also see Matplotlib

```
>>> plt.show() #Show the plot
>>> plt.savefig("foo.png") #Save the plot as a figure
>>> plt.savefig("foo.png", #Save transparent figure
                transparent=True)
```

> Close & Clear

Also see Matplotlib

```
>>> plt.cla() #Clear an axis
>>> plt.clf() #Clear an entire figure
>>> plt.close() #Close a window
```

Python For Data Science *Cheat Sheet*

Seaborn

Learn Data Science Interactively at www.DataCamp.com

Statistical Data Visualization With Seaborn

The Python visualization library **Seaborn** is based on matplotlib and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

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

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips")                    Step 1
>>> sns.set_style("whitegrid")                         Step 2
>>> g = sns.lmplot(x="tip",                            Step 3
                   y="total_bill",
                   data=tips,
                   aspect=2)
>>> g = (g.set_axis_labels("Tip","Total bill(USD)").
set(xlim=(0,10),ylim=(0,100)))                         Step 4
>>> plt.title("title")
>>> plt.show(g)                                        Step 5
```

1 Data

Also see Lists, NumPy & Pandas

```
>>> import pandas as pd
>>> import numpy as np
>>> uniform_data = np.random.rand(10, 12)
>>> data = pd.DataFrame({'x':np.arange(1,101),
                         'y':np.random.normal(0,4,100)})
```

Seaborn also offers built-in data sets:

```
>>> titanic = sns.load_dataset("titanic")
>>> iris = sns.load_dataset("iris")
```

2 Figure Aesthetics

Also see Matplotlib

```
>>> f, ax = plt.subplots(figsize=(5,6))
```
Create a figure and one subplot

Seaborn styles

```
>>> sns.set()
>>> sns.set_style("whitegrid")
>>> sns.set_style("ticks",
                  {"xtick.major.size":8,
                   "ytick.major.size":8})
>>> sns.axes_style("whitegrid")
```
(Re)set the seaborn default
Set the matplotlib parameters
Set the matplotlib parameters

Return a dict of params or use with `with` to temporarily set the style

Context Functions

```
>>> sns.set_context("talk")
>>> sns.set_context("notebook",
                    font_scale=1.5,
                    rc={"lines.linewidth":2.5})
```
Set context to `"talk"`
Set context to `"notebook"`, scale font elements and override param mapping

Color Palette

```
>>> sns.set_palette("husl",3)
>>> sns.color_palette("husl")
>>> flatui = ["#9b59b6","#3498db","#95a5a6","#e74c3c","#34495e","#2ecc71"]
>>> sns.set_palette(flatui)
```
Define the color palette
Use with `with` to temporarily set palette
Set your own color palette

3 Plotting With Seaborn

Axis Grids

```
>>> g = sns.FacetGrid(titanic,
                      col="survived",
                      row="sex")
>>> g = g.map(plt.hist,"age")
```
Subplot grid for plotting conditional relationships

```
>>> sns.factorplot(x="pclass",
                   y="survived",
                   hue="sex",
                   data=titanic)
```
Draw a categorical plot onto a Facetgrid

```
>>> sns.lmplot(x="sepal_width",
               y="sepal_length",
               hue="species",
               data=iris)
```
Plot data and regression model fits across a FacetGrid

```
>>> h = sns.PairGrid(iris)
>>> h = h.map(plt.scatter)
>>> sns.pairplot(iris)
```
Subplot grid for plotting pairwise relationships
Plot pairwise bivariate distributions

```
>>> i = sns.JointGrid(x="x",
                      y="y",
                      data=data)
>>> i = i.plot(sns.regplot,
               sns.distplot)
```
Grid for bivariate plot with marginal univariate plots

```
>>> sns.jointplot("sepal_length",
                  "sepal_width",
                  data=iris,
                  kind='kde')
```
Plot bivariate distribution

Categorical Plots

Scatterplot

```
>>> sns.stripplot(x="species",
                  y="petal_length",
                  data=iris)
```
Scatterplot with one categorical variable

```
>>> sns.swarmplot(x="species",
                  y="petal_length",
                  data=iris)
```
Categorical scatterplot with non-overlapping points

Bar Chart

```
>>> sns.barplot(x="sex",
                y="survived",
                hue="class",
                data=titanic)
```
Show point estimates and confidence intervals with scatterplot glyphs

Count Plot

```
>>> sns.countplot(x="deck",
                  data=titanic,
                  palette="Greens_d")
```
Show count of observations

Point Plot

```
>>> sns.pointplot(x="class",
                  y="survived",
                  hue="sex",
                  data=titanic,
                  palette={"male":"g",
                           "female":"m"},
                  markers=["^","o"],
                  linestyles=["-","--"])
```
Show point estimates and confidence intervals as rectangular bars

Boxplot

```
>>> sns.boxplot(x="alive",
                y="age",
                hue="adult_male",
                data=titanic)
```
Boxplot

```
>>> sns.boxplot(data=iris,orient="h")
```
Boxplot with wide-form data

Violinplot

```
>>> sns.violinplot(x="age",
                   y="sex",
                   hue="survived",
                   data=titanic)
```
Violin plot

Regression Plots

```
>>> sns.regplot(x="sepal_width",
                y="sepal_length",
                data=iris,
                ax=ax)
```
Plot data and a linear regression model fit

Distribution Plots

```
>>> plot = sns.distplot(data.y,
                        kde=False,
                        color="b")
```
Plot univariate distribution

Matrix Plots

```
>>> sns.heatmap(uniform_data,vmin=0,vmax=1)
```
Heatmap

4 Further Customizations

Also see Matplotlib

Axisgrid Objects

```
>>> g.despine(left=True)
>>> g.set_ylabels("Survived")
>>> g.set_xticklabels(rotation=45)
>>> g.set_axis_labels("Survived",
                      "Sex")
```
Remove left spine
Set the labels of the y-axis
Set the tick labels for x
Set the axis labels

```
>>> h.set(xlim=(0,5),
          ylim=(0,5),
          xticks=[0,2.5,5],
          yticks=[0,2.5,5])
```
Set the limit and ticks of the x-and y-axis

Plot

```
>>> plt.title("A Title")
>>> plt.ylabel("Survived")
>>> plt.xlabel("Sex")
>>> plt.ylim(0,100)
>>> plt.xlim(0,10)
>>> plt.setp(ax,yticks=[0,5])
>>> plt.tight_layout()
```
Add plot title
Adjust the label of the y-axis
Adjust the label of the x-axis
Adjust the limits of the y-axis
Adjust the limits of the x-axis
Adjust a plot property
Adjust subplot params

5 Show or Save Plot

Also see Matplotlib

```
>>> plt.show()
>>> plt.savefig("foo.png")
>>> plt.savefig("foo.png",
                transparent=True)
```
Show the plot
Save the plot as a figure
Save transparent figure

Close & Clear

Also see Matplotlib

```
>>> plt.cla()
>>> plt.clf()
>>> plt.close()
```
Clear an axis
Clear an entire figure
Close a window