Seaborn

seaborn extends matplotlib visualization. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Keys Features:

Seaborn is a statistical plotting library

It has beautiful default styles

It also is designed to work very well with Pandas dataframe objects

Seaborn Vs Mpl

```
import numpy as np
import matplotlib as mpl
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Numpy's Plot:

Again, data flows on top of (dancing) underpants (and, whine, tickle)

```
x = np.linspace(0, 10, 500)
dashes = [10, 2, 10, 4]
plt.plot(x, np.sin(x), dashes=dashes)
plt.ylim(-1.5, 1.5);
```

```
[<matplotlib.lines.Line2D at 0x7f8a0a0a0a90>]
(-1.5, 1.5)
```

```
data.head()
```

	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

Numpy Plot:

Data flows on top of (dancing) underpants (and, whine, tickle)

```
sns.distplot(data['total_bill'])
```

```
/home/dell/anaconda3/envs/ai/lib/python3.7/site-packages/seaborn/distplot.py:183:FutureWarning:
```

Relational Plot:

Often that we want to view the joint in that we want to represent the data set in the relationship.

Here, plot by dataset scatter plot to have closer the relational plot between the total bill and tip of the data dataset.

```
sns.jointplot(x='total_bill', y='tip', data=data)
```

```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```

In relational plot with hue

We can also differentiate the plot points by other data that is numerical and binder data set hue can also be a with the help of hue

```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```

If we want the relationship with the smoker as well as time data with the x and y continuous data hue is best with any hue

Hues can use the palette to assign colours to the different labels and shapes to the form of the scatter. as time with the impact in the total bill and tip

Changing palettes

Relational plot with the hue and style.

```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```

Here is one hue plotted points for the smoker and style is kind of smoker of the data.

This is known as qualitative palettes.

```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```

If we think about the shape the size of the color instead of changing the color of the circle

hue changes colour, size changes in size

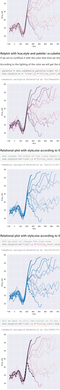
Relational plot with palette,size & palette

Palette can be used to change the colour of the dataset. Sometimes it is something that can be interpreted by. Pure colors palette. sns is following palettes that are known to explicitly exist.

```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```

If we want to increase the size of the the data in the graph but in the range.

```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```



New Datasets.

```
import numpy as np
import matplotlib as mpl
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
df = sns.load_dataset("iris")
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Relational plot

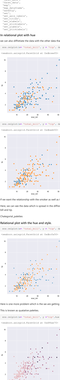
```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```


Size The plot Data.

```
sns.jointplot(x="sepal_length", y="sepal_width", data=df, size=5, kind="reg")
```

```
/home/dell/anaconda3/envs/ai/lib/python3.7/site-packages/seaborn/distrib.py:369:UserWarning:
```

```
<seaborn.axisgrid.JointGrid at 0x7f8a0a0a0a90>
```



New Datasets.

```
import numpy as np
import matplotlib as mpl
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
df = sns.load_dataset("iris")
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Relplot with hue

compared when with the help of the hue as well as we called the even it with the different colour

```
sns.relplot(x="sepal_length", y="sepal_width", data=df, hue="species", kind="scatter")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```

Relplot with hue,size

In that we had passed the data in the different format

```
sns.relplot(x="sepal_length", y="sepal_width", data=df, hue="species", size="petal_length", kind="scatter")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```


Relplot with hue,style,markers,dashes implementation.

Markers are used to change the shape of the plot

Dashes is used to map the dashes in lines to use such that a dashed line is removed or make it true

```
sns.relplot(x="sepal_length", y="sepal_width", data=df, hue="species", style="species", markers=True, dashes=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```

Relationship plot has,size and dash of the event in stem format

When you are working with seaborn however data that is you have with that seaborn analysis multiple times, you can also pick each

Relplot has the ability to use the same data set and plot it in different ways. Relplot is a figure level function

Pick keyword arguments for the plot using the kind keyword argument.

```
sns.relplot(x="sepal_length", y="sepal_width", hue="species", size="petal_length", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```



```
sns.relplot(x="sepal_length", y="sepal_width", hue="species", size="petal_length", data=df, kind="line")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```


New Datasets.

```
import numpy as np
import matplotlib as mpl
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
df = sns.load_dataset("tips")
df.head()
```

	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

Here we get the data set of the tips data set in the other format.

```
sns.relplot(x="day", y="total_bill", hue="smoker", data=df)
```

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

Relational plot with y,hue,kind

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```


Relational plot with hue,style

Hue is used to change the colour of the dataset in the different form.

Markers are used for the coherence as required and the case and hues is expressed with the two type differences.

```
sns.relplot(x="day", y="total_bill", hue="smoker", data=df, kind="line", style="smoker")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```


Relplot with hue,kind,palette,y,colorbrew,palette.

If we call to y,colour it plot the color that they are bounded by to use the palette.

According to the legend the colour, we can get the desired qualitative representation

```
sns.relplot(x="day", y="total_bill", hue="smoker", data=df, kind="line", palette="colorblind", palette="colorblind", palette="colorblind")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```


Relational plot with style,size according to the coherence.

Here changes the shape of the color based on the coherence

```
sns.relplot(x="day", y="total_bill", hue="smoker", data=df, kind="line", style="smoker", size="size")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```


Relational plot with hue,palette,kind according to the range

```
sns.relplot(x="day", y="total_bill", hue="smoker", data=df, kind="line", palette="colorblind", palette="colorblind", palette="colorblind")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```



```
sns.relplot(x="day", y="total_bill", hue="smoker", data=df, kind="line", palette="colorblind", palette="colorblind", palette="colorblind")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8a0a0a0a90>
```


Overall Presentation over the Seaborn Relational plot.

This will give you the big picture of the data trends.

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline
%load_ext rpy2.ipython
%R library(ggplot2); library(Cairo); library(grid); library("RColorBrewer")
```

```
rpy2.ipython extension already loaded. To reload it, use:
  %load_ext rpy2.ipython
```

```
%R -i df_long -w 1000 -h 400 -u px -o plot_height="300" -o plot_width="1000" -o plot_res="150" -o plot_units="in" -o plot_type="pdf" -o plot_name="Line_Chart" -o plot_ratio="2.5" -o plot_dpi="150" -o plot_bg="transparent" -o plot_family="Times" -o plot_pointsize="12" -o plot_antialias="TRUE" -o plot_quality="50" -o plot_compression="lzw" -o plot_colormodel="srgb" -o plot_scale="1" -o plot_encoding="utf8" -o plot_bookmark="FALSE" -o plot_title="" -o plot_xlab="" -o plot_ylab="" -o plot_xlim="" -o plot_ylim="" -o plot_asp="" -o plot_interpolate="FALSE" -o plot_lty="1" -o plot_lwd="1" -o plot_lwd.ticks="1" -o plot_lwd.axis="1" -o plot_lwd.grid="1" -o plot_lwd.lab="1" -o plot_lwd.main="1" -o plot_lwd.sub="1" -o plot_col="black" -o plot_col.ticks="black" -o plot_col.axis="black" -o plot_col.grid="black" -o plot_col.lab="black" -o plot_col.main="black" -o plot_col.sub="black" -o plot_fg="black" -o plot_bg="transparent" -o plot_cex="1" -o plot_cex.ticks="1" -o plot_cex.axis="1" -o plot_cex.grid="1" -o plot_cex.lab="1" -o plot_cex.main="1" -o plot_cex.sub="1" -o plot_pch="1" -o plot_pch.ticks="1" -o plot_pch.axis="1" -o plot_pch.grid="1" -o plot_pch.lab="1" -o plot_pch.main="1" -o plot_pch.sub="1" -o plot_mfrow="" -o plot_mfcol="" -o plot_mfrow.ticks="" -o plot_mfrow.axis="" -o plot_mfrow.grid="" -o plot_mfrow.lab="" -o plot_mfrow.main="" -o plot_mfrow.sub="" -o plot_mfcol.ticks="" -o plot_mfcol.axis="" -o plot_mfcol.grid="" -o plot_mfcol.lab="" -o plot_mfcol.main="" -o plot_mfcol.sub="" -o plot_mar="" -o plot_mar.ticks="" -o plot_mar.axis="" -o plot_mar.grid="" -o plot_mar.lab="" -o plot_mar.main="" -o plot_mar.sub="" -o plot_oma="" -o plot_oma.ticks="" -o plot_oma.axis="" -o plot_oma.grid="" -o plot_oma.lab="" -o plot_oma.main="" -o plot_oma.sub="" -o plot_xaxt="" -o plot_yaxt="" -o plot_xaxs="" -o plot_yaxs="" -o plot_xaxp="" -o plot_yaxp="" -o plot_xaxp.ticks="" -o plot_xaxp.axis="" -o plot_xaxp.grid="" -o plot_xaxp.lab="" -o plot_xaxp.main="" -o plot_xaxp.sub="" -o plot_yaxp.ticks="" -o plot_yaxp.axis="" -o plot_yaxp.grid="" -o plot_yaxp.lab="" -o plot_yaxp.main="" -o plot_yaxp.sub="" -o plot_xlog="" -o plot_ylog="" -o plot_xlog.ticks="" -o plot_xlog.axis="" -o plot_xlog.grid="" -o plot_xlog.lab="" -o plot_xlog.main="" -o plot_xlog.sub="" -o plot_ylog.ticks="" -o plot_ylog.axis="" -o plot_ylog.grid="" -o plot_ylog.lab="" -o plot_ylog.main="" -o plot_ylog.sub="" -o plot_xpd="" -o plot_xpd.ticks="" -o plot_xpd.axis="" -o plot_xpd.grid="" -o plot_xpd.lab="" -o plot_xpd.main="" -o plot_xpd.sub="" -o plot_bty="" -o plot_bty.ticks="" -o plot_bty.axis="" -o plot_bty.grid="" -o plot_bty.lab="" -o plot_bty.main="" -o plot_bty.sub="" -o plot_ann="" -o plot_ann.ticks="" -o plot_ann.axis="" -o plot_ann.grid="" -o plot_ann.lab="" -o plot_ann.main="" -o plot_ann.sub="" -o plot_las="" -o plot_las.ticks="" -o plot_las.axis="" -o plot_las.grid="" -o plot_las.lab="" -o plot_las.main="" -o plot_las.sub="" -o plot_tck="" -o plot_tck.ticks="" -o plot_tck.axis="" -o plot_tck.grid="" -o plot_tck.lab="" -o plot_tck.main="" -o plot_tck.sub="" -o plot_tcl="" -o plot_tcl.ticks="" -o plot_tcl.axis="" -o plot_tcl.grid="" -o plot_tcl.lab="" -o plot_tcl.main="" -o plot_tcl.sub="" -o plot_mgp="" -o plot_mgp.ticks="" -o plot_mgp.axis="" -o plot_mgp.grid="" -o plot_mgp.lab="" -o plot_mgp.main="" -o plot_mgp.sub
```

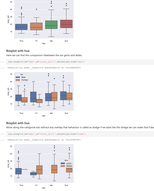
From the above categorical plot a numerical can be inferred that we can actually finding the statistical analysis over the data which is
Here we can find the mean, median, mode, standard deviation, variance, skewness, kurtosis.

In this we can find the plot having the outlier upper whisker that is outlying the Max and lower is minimum.
Upper quartile is the upper horizontal line that covering the graph from the upper quartile to the two data outlining.
Lower quartile is the horizontal line that covering the graph from the lower quartile to the two data outlining.
Middle portion line that containing standard type of structure in which data outlining is 50% that is median.
Data is increasing or decreasing from the lower quartile to upper quartile and generally according to the set var. measure from 1 to 5 by using the
Seaborn library.

Simple Boxplot

```
sns.boxplot(data=df,x='species',y='petal_length',palette='deep')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```



Boxplot with hue

Here we use the the response variable as the species and sepal.

```
sns.boxplot(data=df,x='species',y='petal_length',palette='deep',hue='sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

Boxplot with hue

Since doing the categorical we without any overlap that behaviour is called as dodge if we want the this dodge we can make that false

```
sns.boxplot(data=df,x='species',y='petal_length',palette='deep',hue='sex')
```

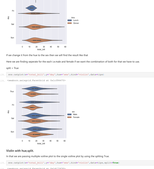
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

We have the categorical plotting in the form of the dodge behaviour as for make it false

Boxplot with hue,dodge=false to make the as the presentation of the graph

```
sns.boxplot(data=df,x='species',y='petal_length',palette='deep',hue='sex',dodge=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```



If we notice that the sepal.length is not giving the enough statistical info about the data by using the boxplot that show we are having
difficult to find the data.

Iris Dataset.

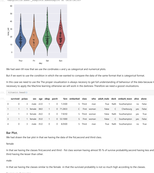
```
df=pd.read_csv('/content/iris.csv')
df.head(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

Catplot with hueue

```
sns.catplot(data=df,x='species',y='petal_length',palette='deep',hue='sepal_width',kind='swarm')
```

```
<matplotlib.axes._subplots.FacetGrid at 0x7fc4c0a8d0>
```



Boxenplot with dodge=false

```
sns.boxenplot(data=df,x='species',y='petal_length',palette='deep',hue='sepal_width',dodge=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

If we change it from the hue to the row then we will find the result like that

Here we are finding separate the row wise sepal.width and petal.length with the combination of both for that we have to use.

catplot.kind

Violin Plot

Violin plot with hue

In this we can observed that the petals has not done the other on the days that we and we can see and out of the other these were not in

```
sns.violinplot(data=df,x='species',y='petal_length',palette='deep',hue='sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

If we change it from the hue to the row then we will find the result like that

Here we are finding separate the row wise sepal.width and petal.length with the combination of both for that we have to use.

catplot.kind

Violin with hueplot

In that we are passing multiple values plot for the single outline plot by using the splitting True

```
sns.violinplot(data=df,x='species',y='petal_length',palette='deep',hue='sex',split=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

Violin with inner

We can modify stuff by using the some other arguments

In that we split open the pie plot inside the violin which is not split

```
sns.violinplot(data=df,x='species',y='petal_length',palette='deep',hue='sex',split=True,inner='stick')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

Violin plot inner hue and pie plot

In that we are finding the pie plot in that we having the hue and the inner stick.

```
df.head(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

Square plot and Violin plot negative

In that we put together the square plot and violin plot the pie plot data stored in the variable form and the violin is another plot and

a. ceaborn.kde. Returns the line object with the plot drawn onto it.

```
f,ax=plt.subplots(1,2,figure=None,dpi=None,facecolor=None,edgecolor=None,
                  subplot_kw=None,gridspec_kw=None,squeeze=True,clear=False,tight_layout=False)
sns.violinplot(data=df,x='species',y='petal_length',palette='deep',hue='sex',split=True,inner='stick',ax=ax[0])
sns.kde(data=df,x='species',y='petal_length',palette='deep',hue='sex',ax=ax[1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

We had seen 20 case that we are the confidence's and p. categorical and numerical plots.

In this case we have seen the violin plot in which the we seeked a compare the data of the species that a categorical format

In this case we seed in to see the proper visualization is always necessary to get full understanding of behaviour of the data because it
necessary to apply the machine learning algorithms and all and in the statistics. Therefore we apply a greater consistency.

Bar Plot

In that we having the Uber.csv file

```
df=pd.read_csv('/content/Uber.csv')
df.head(10)
```

	pickup_date	pickup_time	dropoff_date	dropoff_time	fare	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	key
0	2015-05-07	19:52:06	2015-05-07	19:54:00	7.5	-73.999817	40.738354	-73.999512	40.723217	1	2015-05-07 19:52:06.0000003
1	2009-07-17	20:04:56	2009-07-17	20:05:00	7.7	-73.994355	40.728225	-73.994710	40.750325	1	2009-07-17 20:04:56.0000002
2	2009-08-24	21:54:00	2009-08-24	22:01:00	12.9	-74.005043	40.740770	-73.962565	40.772647	1	2009-08-24 21:54:00.0000001
3	2009-08-26	08:08:26	2009-08-26	08:17:00	5.3	-73.976124	40.790844	-73.965316	40.799052	1	2009-08-26 08:08:26.0000002
4	2014-08-28	17:47:00	2014-08-28	17:55:00	16.0	-73.925023	40.744085	-73.973082	40.761247	2	2014-08-28 17:47:00.0000001
5	2012-10-28	10:26:00	2012-10-28	10:38:00	8.5	-73.955064	40.781779	-73.924466	40.746744	1	2012-10-28 10:26:00.0000001
6	2014-12-29	21:52:24	2014-12-29	22:09:33	16.9	-74.000964	40.731630	-73.972832	40.758092	2	2014-12-29 21:52:24.0000002
7	2015-05-18	08:18:00	2015-05-18	08:24:00	5.7	-73.951544	40.774138	-73.959875	40.770672	1	2015-05-18 08:18:00.0000006
8	2010-12-30	00:53:00	2010-12-30	01:00:00	7.7	-73.990091	40.751663	-73.979268	40.746916	1	2010-12-30 00:53:00.0000003
9	2011-11-29	09:12:00	2011-11-29	09:20:00	7.7	-73.990064	40.736462	-73.988161	40.748774	1	2011-11-29 09:12:00.0000005

Bar Plot

Here we have shown the bar plot in that we having the data of the fare,passenger and their class.

Bargraph

In that we having the Uber.csv file and and fare . Fare class summer having almost 55 % of survival probability second having low and
third having the lowest chance.

```
df=pd.read_csv('/content/Uber.csv')
```

In that we having the classes similar to the Titanic. In that the survival probability is not so much high according to the classes.

So you are discovering

Bar Plot with hue

```
sns.barplot(data=df,x='passenger_count',y='fare',palette='deep',hue='pickup_hour')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```


The hue is the variable in the dataset that we are using the data hue much is the proportionality that we the female having the lower proportionality.

Hue helps us to see the relationship between the two variables.

Hue high proportionality belonging to survived passes.

Count Plot

In that we having the sepal with the data column is the hue and the petal column that is changing according to the data.

Hue is most of the people through the hue is which

```
sns.countplot(data=df,x='passenger_count',palette='deep',hue='pickup_hour')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

Boxplot with Hue

We can use the Uber.csv file in the

```
df=pd.read_csv('/content/Uber.csv')
sns.boxplot(data=df,x='passenger_count',y='fare',palette='deep',hue='pickup_hour')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```

Pointplot

From the pointplot plot we easily understanding the if baseline is survival then the fare is higher chance to survive.

```
sns.pointplot(data=df,x='passenger_count',y='fare',palette='deep',hue='pickup_hour')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```


Visualizing Distribution Of The Data

Visualizing Distribution of the data

```
df=pd.read_csv('/content/Titanic.csv')
df.head(10)
```

Histogram

Here we understand and visualize the how the distribution of the data has been done.

Hue is a dimensional plot by another variable

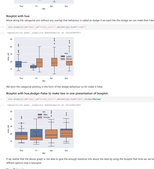
The plot is distributed in the universal distribution and the most convenient way to use the convenient way by using is to using
the seaborn.distplot() is mainly use.

Histplot

The distplot used the histogram,kernel density summarizer(kde)

```
sns.distplot(data=df,x='Age',bins=10)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```



This is plot is known as distribution plot or a plot is consisting in the histogram and KDE plot.

In the bag of the ceiling the observation get that and plot under the curve is always 1 that is known as normal distribution

If we want to remove the KDE plot that the bins is just plot in the bargraph that is 1.

In the bins having the width - 4 value only that we will get the bargraph that is and that is true

Kde Plot

If we want to not according to the axes we need to use the False and True is the parameters.

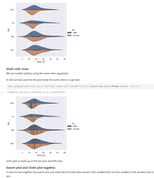
```
sns.distplot(data=df,x='Age',kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```


Rug plot

```
sns.distplot(data=df,x='Age',kde=False,rug=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```



If we want the rids set hue then

```
sns.distplot(data=df,x='Age',kde=False,rug=True,hue='Survived')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```


If we use the kde as true available in the graph it automatically adjusted that value according to personal purpose in PRINT case we can
see the bargraph that was 1.

```
sns.distplot(data=df,x='Age',kde=True,rug=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```


Count Plot

In that we have the survived column in the hue

```
sns.countplot(data=df,x='Survived',hue='Survived')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```


In that case we are taking the hue is the Survived

```
sns.countplot(data=df,x='Survived',hue='Survived',palette='deep')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4c0a8d0>
```


Jointplot

In that we having other set of the hue in the jointplot is a kind of the kde plot

```
sns.jointplot(data=df,x='Age',y='Survived',kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc4c0a8d0>
```

According to hueplot the kde set is not get distinguishing a smooth curve.

```
sns.jointplot(data=df,x='Age',y='Survived',kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc4c0a8d0>
```


Squid from that if we use hex as parameter that time

```
sns.jointplot(data=df,x='Age',y='Survived',kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc4c0a8d0>
```


New Dataset.

```
df=pd.read_csv('/content/Titanic.csv')
df.head(10)
```

	survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	Male	22.0	1	0	7.2500	S
1	1	1	Female	38.0	1	0	71.2833	C
2	1	3	Female	26.0	0	0	7.9250	S
3	1	1	Female	35.0	1	0	53.1000	S
4	0	3	Male	35.0	0	0	8.0500	S
5	0	3	Male	NaN	0	0	8.4583	Q
6	0	1	Male	54.0	0	0	51.8625	S
7	0	3	Male	2.0	3	1	21.0750	S
8	1	3	Female	27.0	0	2	11.1333	S
9	1	2	Female	14.0	1	0	30.0708	C

Joint Plot

In that we are having the jointplot is giving the a the stress with the with the joint plot.

```
df=pd.read_csv('/content/Titanic.csv')
df.head(10)
```

	survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	Male	22.0	1	0	7.2500	S
1	1	1	Female	38.0	1	0	71.2833	C
2	1	3	Female	26.0	0	0	7.9250	S
3	1	1	Female	35.0	1	0	53.1000	S
4	0	3	Male	35.0	0	0	8.0500	S
5	0	3	Male	NaN	0	0	8.4583	Q
6	0	1	Male	54.0	0	0	51.8625	S
7	0	3	Male	2.0	3	1	21.0750	S
8	1	3	Female	27.0	0	2	11.1333	S
9	1	2	Female	14.0	1	0	30.0708	C

Joint Plot with hue/ kind

```
sns.jointplot(data=df,x='Age',y='Survived')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc4c0a8d0>
```

```
[1] "Warning: Ignoring unknown parameters: binwidth, bins, pad"
```

Point Plot with hue = fare

```
sns.pairplot(df,hue='fare',palette='Paired')
plt.savefig('pairplot.jpeg')
```

```
[1] "Warning: Ignoring unknown parameters: palette"
```

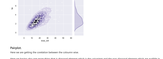
KDE plot with corner=True,diag_kind='kde',hue='fare'

```
sns.pairplot(df,corner=True,diag_kind='kde',hue='fare')
plt.savefig('pairplot.jpeg')
```

```
[1] "Warning: Ignoring unknown parameters: corner, diag_kind, palette"
```

```
[1] "Warning: Ignoring unknown parameters: palette"
```

```
[1] "Warning: Ignoring unknown parameters: palette"
```



Pairplot

Here we are getting the correlation between the columns area

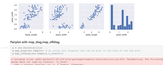
Here we having plot one more thing that is diagonal element which is the univariant and the rest diagonal element which are scatter in the bivariant relationship

```
sns.pairplot(df,diagonal_kind='hist')
```

Pairplot with map_diag=hist,y_vars

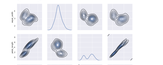
```
# y_vars=['Survived']
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'])
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired')
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'])
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2)
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5)
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg')
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10})
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5})
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5})
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True)
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True)
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'])
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'])
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True)
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True,hue_order=['Survived','fare'])
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True,hue_order=['Survived','fare'],diagonal_kind='kde')
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True,hue_order=['Survived','fare'],diagonal_kind='kde',diag_sharey=False)
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True,hue_order=['Survived','fare'],diagonal_kind='kde',diag_sharey=False,diagonal_kws={'bandwidth':0.5})
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True,hue_order=['Survived','fare'],diagonal_kind='kde',diag_sharey=False,diagonal_kws={'bandwidth':0.5},diag_kws={'bins':10})
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True,hue_order=['Survived','fare'],diagonal_kind='kde',diag_sharey=False,diagonal_kws={'bandwidth':0.5},diag_kws={'bins':10},diagonal_kws={'bins':10})
# sns.pairplot(df,hue='fare',diagonal_kind='hist',y_vars=['Survived'],palette='Paired',markers=['o','s','D'],height=2,aspect=1.5,kind='reg',diag_kws={'bins':10},plot_kws={'alpha':0.5},diagonal_kws={'alpha':0.5},corner=True,dropna=True,vars=['Survived','fare','Age'],x_vars=['Survived','fare'],diag_sharey=True,hue_order=['Survived','fare'],diagonal_kind='kde',diag_sharey=False,diagonal_kws={'bandwidth':0.5},diag_kws={'bins':10},diagonal_kws={'bins':10},diagonal_kws={'bins':10})
```

```
[1] "Warning: Ignoring unknown parameters: palette"
```



Linear Regression and Relationship:

```
sns.lmplot(x='Age',y='Fare',data=df)
```

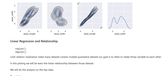


Lmplot:

We having another plot that is lmplot(linear model plot)

Linear plot is that we use to calculate the trend (AR is up)

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived')
```



Linear relationship between the age and fare

If we see that we are in increasing the fare is also increasing

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired')
```

Linear relationship between the age and fare

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'])
```

Lmplot with jitter

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5)
```

Lmplot with a scatterplot

We have plot the lmplot and scatterplot on the line which is represented as mean.

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True)
```

We will show the lmplot line with line which will represent very easily and it can also show us the intersection possible and by using that lmplot

Lmplot with a scatterplot

We have plot the lmplot and scatterplot on the line which is represented as mean.

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=False)
```

Certainly we can show the line which represents the mean value

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True)
```

Lets think that we having the data which is not fitted in the linear relationship so for that lets see.

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,order=2)
```

```
[1] "Warning: Ignoring unknown parameters: order"
```

Survived	fare	Age
0	7.2500	22.0
1	71.2833	38.0
2	7.9250	26.0
3	53.1000	35.0
4	8.0500	35.0

Here we can see that the each dataset containing the 11 values and not having the four types of dataset

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,order=4)
```

```
[1] "Warning: Ignoring unknown parameters: order"
```

Here we are going to plot the dataset which is 1 and 2 = Fare, with scatter 3rd.

Whenever we see the linear relationship between the variables (scatter,line,line) . Whenever Multivariant Regressional arguments to pass to sns.lmplot() and sns.jplot()

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},)
```

Similarly if we want to plot the data of the 0 dataset.

Lets see what happens here

We will observe that the data will go up the linear relationship with the a scatterplot.=>0.1

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},)
```

Lmplot with x=Pclass,y=Age,2

We need to describe the which kind of a personal relationship it is.

By passing the order we can see the the polynomial fitting the plot.

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},)
```

Lmplot with data only dataset is 10.

Here we see the scatter plot is not fitted the line that means there is high chance of the error

For that we need to remove the outlier by automatically or by manually

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},)
```

Lmplot with fit_reg=False

Here we have lmplot which plots scatter in the form of scale and fitted and it superposed with the help of the square line to show the linear relationship between two continuous and scale and scatter respectively.

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=False,scatter_kws={'s':50},)
```

To remove the outlier we need set the value = True

scatterplot(x = x_value, y = y_value,data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=False,scatter_kws={'s':50},)

Here we have set the hue = 'Survived' and the palette = 'Paired' and markers = ['o','D'] and x_jitter=0.5 and scatter=True

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=False,scatter_kws={'s':50},)
```

Lmplot with fit_reg=True and markers.

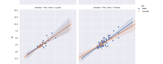
```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},)
```

We can observe in all the different subpoint

In this way we can get the linear subpoint with the help of the sub arguments.

Lmplot with hue=markers,col for time

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},col='Pclass')
```



In the plot we have plotted the lmplot with the help of the subplot respective of a column attribute we are passing here.

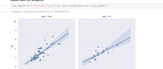
Lmplot with hue=markers,col and row.

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},col='Pclass',row='Sex')
```

In this case we have both feature row and col.

Lmplot with col_wrap=3

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},col='Pclass',col_wrap=3)
```

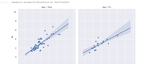


We can observe the length of the figure depending the subplots are length.

Lmplot with col=col_wrap,height to get good representation.

```
sns.lmplot(x='Fare',y='Age',data=df,hue='Survived',palette='Paired',markers=['o','D'],x_jitter=0.5,scatter=True,fit_reg=True,scatter_kws={'s':50},col='Pclass',col_wrap=3,height=3)
```

```
ggplot(titanic, aes(age, fill = Survived)) + geom_histogram()
```



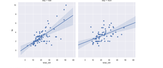
Controlling global aesthetics.

Controlling global aesthetics

```
ggplot(titanic)
  geom_point(
    aes(age, fare),
    color = "red",
    alpha = 0.5,
    size = 3)
```

By passing the aesthetics to the ggplot() function, we can change its shape and size.

```
ggplot(titanic)
  geom_point(aes(age, fare))
  geom_smooth(aes(age, fare))
  facet_wrap(~Survived)
```



We have by different groups of the ggplot() function and we will get to the names the whatever.

This is how we can plot the scatter in the ellipsoid style.

```
ggplot(titanic)
  geom_point(aes(age, fare))
  geom_smooth(aes(age, fare))
  facet_wrap(~Survived)
  coord_cartesian(
    xlim = c(0, 50),
    ylim = c(0, 100))
```

If we wanted that there is a long line or like a curve line we can call a line also.

```
ggplot(titanic)
  geom_point(aes(age, fare))
  geom_smooth(aes(age, fare))
  facet_wrap(~Survived)
  coord_cartesian(
    xlim = c(0, 50),
    ylim = c(0, 100))
  coord_flip()
```

If we want to remove the flat left hand shape as well as bottom as well.

```
ggplot(titanic)
  geom_point(aes(age, fare))
  geom_smooth(aes(age, fare))
  facet_wrap(~Survived)
  coord_cartesian(
    xlim = c(0, 50),
    ylim = c(0, 100))
  coord_flip()
  theme_bw()
```

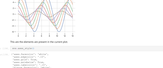
The data can be defined as a point in the current plot.

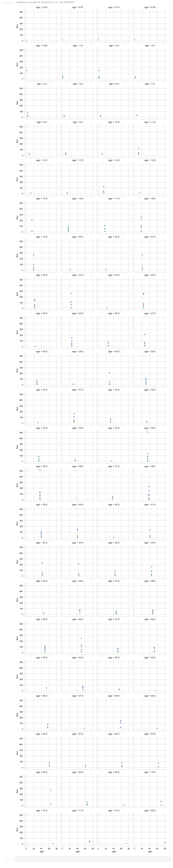
```
ggplot(titanic)
  geom_point(aes(age, fare))
  geom_smooth(aes(age, fare))
  facet_wrap(~Survived)
  coord_cartesian(
    xlim = c(0, 50),
    ylim = c(0, 100))
  coord_flip()
  theme_bw()
  theme(
    axis.text = element_blank()
    axis.ticks = element_blank()
    axis.titles = element_blank()
    panel.grid.major = element_blank()
    panel.grid.minor = element_blank()
    panel.background = element_blank()
    strip.background = element_blank()
    strip.text = element_blank()
    legend.position = "none")
  scale_color_manual(
    values = c("#ff0000", "#ff0000"))
  scale_fill_manual(
    values = c("#ff0000", "#ff0000"))
  scale_size_manual(
    values = c(3, 3))
  scale_shape_manual(
    values = c(19, 19))
  scale_linetype_manual(
    values = c("solid", "solid"))
  scale_alpha_manual(
    values = c(0.5, 0.5))
  scale_size_manual(
    values = c(3, 3))
  scale_shape_manual(
    values = c(19, 19))
  scale_linetype_manual(
    values = c("solid", "solid"))
  scale_alpha_manual(
    values = c(0.5, 0.5))
```

We can change the line color.

This is how we can change the line type using the line scale aesthetic.

```
ggplot(titanic)
  geom_point(aes(age, fare))
  geom_smooth(aes(age, fare))
  facet_wrap(~Survived)
  coord_cartesian(
    xlim = c(0, 50),
    ylim = c(0, 100))
  coord_flip()
  theme_bw()
  theme(
    axis.text = element_blank()
    axis.ticks = element_blank()
    axis.titles = element_blank()
    panel.grid.major = element_blank()
    panel.grid.minor = element_blank()
    panel.background = element_blank()
    strip.background = element_blank()
    strip.text = element_blank()
    legend.position = "none")
  scale_color_manual(
    values = c("#ff0000", "#ff0000"))
  scale_fill_manual(
    values = c("#ff0000", "#ff0000"))
  scale_size_manual(
    values = c(3, 3))
  scale_shape_manual(
    values = c(19, 19))
  scale_linetype_manual(
    values = c("solid", "solid"))
  scale_alpha_manual(
    values = c(0.5, 0.5))
  scale_size_manual(
    values = c(3, 3))
  scale_shape_manual(
    values = c(19, 19))
  scale_linetype_manual(
    values = c("solid", "solid"))
  scale_alpha_manual(
    values = c(0.5, 0.5))
  scale_color_manual(
    values = c("#ff0000", "#ff0000"))
```




age = 0.00
age = 0.08
age = 0.75
age = 0.83
age = 0.92
age = 1.00
age = 1.12
age = 1.33
age = 1.50
age = 1.58
age = 1.67
age = 1.75
age = 1.83
age = 1.92
age = 2.00
age = 2.12
age = 2.33
age = 2.50
age = 2.58
age = 2.67
age = 2.75
age = 2.83
age = 2.92
age = 3.00
age = 3.12
age = 3.33
age = 3.50
age = 3.58
age = 3.67
age =