Seaborn L-4

Distribution plot -

(1) Distplot
(2) Joint Plot
(3) Pair plot
(4) Rug plot
(5) KDE Plot

Seaborn

```
Import seaborn as sns
%. matplotlib in line
```

Seaborn comes with built-in data sets
tips = sns.load_dataset('tips')
tips.head()

Distplot

→ The distplot shows the distribution of univariate set of observation.
sns.distplot(tips['total_bill']).

→ To remove the kde layer and just have the histogram use:
sns.distplot(tips['total_bill'], kde=False, bins=30)

Joint plot

→ Jointplot() allows you to basically match up two distplots for bivariate data. with your choice of what kind of parameter to compare with.

(1) scatter
(2) Reg
(3) resid
(4) kde
(5) hex

sns.jaintplot(x = 'total_bill', y = 'tip', data = tips, kind = 'scatter').

sns.jaintplot(x = 'total_bill', y = 'tip', data = tips, kind = 'hex')

sns.jaintplot(x = 'total_bill', y = 'tip', data = tips, kind = 'reg')

Page No.
Date :

Pairplot :- Pair plot will plot pairwise relationship across an entire dataframe and support a color hue argument

```
sns.pairplot(tips)
sns.pairplot(tips, hue='sex', palette='coolwarm')
```

Rugplots :- Rug plot are actually a very simple concept they just draw a dash mark for every point on a univariate distribution. They are building block of a KDE plot

```
sns.Rugplot(tips['total_bill'])
```

Categorical Plots

1. Factorplot
2. Box plot
3. Violinplot
4. Stripplot
5. Swarmplot
6. barplot
7. countplot

Pairplot :- Pair plot will plot pairwise relationship across an entire dataframe and support a color hue argument

sns.pairplot(tips)

sns.pairplot(tips, hue='sex', palette='coolwarm')

Rugplots :- Rug plot are actually a very simple concept they just draw a dash mark for every point on a univariate distribution. They are building block of a KDE plot

sns.Rugplot(tips['total_bill'])

Categorical Plots

1. Factorplot
2. Box plot
3. Violinplot
4. Stripplot
5. Swarmplot
6. boxplot
7. countplot

Barplot and countplot

→ Allows you to get Aggregate data of a categorical feature in your data. Barplot is a generalplot that allows you to aggregate the categorical data based on some function

sns.barplot(x='sex', y='total_bill', data=tips)

→ Import numpy as np

You can change the Estimator object to your own function, that converts a vector to a scalar.

sns.barplot(x='sex', y='total_bill', data=tips, Estimator=np.std);

Count Plot

This is essentially the same as barplot except the Estimator is explicitly counting the number of occurrences. which is why we only pass the x value.

sns.countplot(x='sex', data=tips).

box plot and Violinplot

Boxplots and Violinplots are used to show the distribution of categorical data. A box plot shows the distribution of quantitative data in a way that facilitates comparison between variables or across level of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of distribution except for points that are determined to be outliers using a method that is a function of the interquartile range.

-> sns.boxplot(x = "day", y = "total_bill", data = tips, palette = "rainbow").

# we can do entire data frame with orient = 'h'
sns.boxplot(data = tips, palette = 'rainbow', orient = 'h')

-> sns.boxplot(x = "day", y = "total_bill", hue = 'smoker', data = tips, palette = "coolwarm").

Violin Plot

A Violin plot plays a similar role as a box and whisker plot. It shows the distribution of quantitative data across several levels of one categorical variables such that those distributions can be compared. Unlike a box plot, in which all the plot components correspond to actual data points, the violinplot features a KDE of the underlying distribution

-> sns.violinplot(x = "day", y = "total_bill", data = tips, palette = 'rainbow')

-> sns.violinplot(x = "day", y = "total_bill", data = tips, hue = 'sex', palette = 'set1')

-> sns.violinplot(x = "day", y = "total_bill", data = tips, hue = 'sex', split = true, palette = 'set1').

Page No.
Date :

Stripplot and Swarmplot

→ The Stripplot will draw a scatterplot where one variable is categorical. A stripplot can be drawn on its own, but it is also a good complement to a box or violin plot.

→ The swarmplot is similar to 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 observation.

→ sns.stripplot(x = "day", y = "total_bill", data = tips)
sns.stripplot(x = "day", y = "total_bill", data = tips, jitter = True)
sns.stripplot(x = "day", y = "total_bill", data = tips, jitter = True, hue = 'sex', palette = 'Set2')
sns.stripplot(x = "day", y = "total_bill", data = tips, jitter = True, hue = 'sex', palette = 'Set1', split = True)

↓

You can use dodge as a split has been replaced.

→ sns.swarmplot(x = "day", y = "total_bill", data = tips)
sns.swarmplot(x = "day", y = "total_bill", hue = 'sex', data = tips, palette = "Set2", split = True)

Combining Categorical plots

sns.violinplot(x="tip", y="day", data=tips, palette='rainbow')
sns.swarmplot(x="tip", y="day", data=tips, colors='black', size=3)

factorplot

sns.factorplot(x='sex', y='total_bill', data=tips, kind='bar')

Matrix plot

Matrix plot allows you to plot data as color encoded matrices and can also be used to indicate clusters within the data

```
Hights = sns.load_dataset('Hights')
tips = sns.load_dataset('tips')
tips.head()
Hights.head()
```

Heat maps

→ For working of heatmaps you need to have your data in a correlation matrix form. basically just colors it in for you.

```
tips.head()
#matrix form for correlation data
tips.corr()
sns.heatmap(tips.corr())
sns.heatmap(tips.corr(), cmap='coolwarm', annot=True)
```

Page No.
Date :

```
Hights pivat_table ( values = 'passengers', index = 'month'
                     (columns = 'year')

PVHights = Hights . pivat_table (values = 'passengers',
                                 index = 'month',
                                 columns = 'year')

sns.heatmap (PVHights).

sns. heatmap (PVHights, cmap = 'magma', linecolor = 'white',
              linewidths = 1)
```

→ Clustermap

The clustermap uses hierarchical clustering to produce a clustered version of the heatmap.

```
sns. Clustermap (PVHights)
sns. Clustermap (PVHights, cmap = 'coolwarm', standard_scale = 1)
```

Regression plots

Implot allows you to display linear models, but it also conviniently allows you to split up these plots based off features, as well as coloring the hue based off features.

```
Import seaborn as sns
% matplotlib inline
tips = sns. load_dataset ('tips')
tips. head ()
```

lmplot()

sns.lmplot(x='total_bill', y='tip', data=tips)
sns.lmplot(x='total_bill', y='tip', data=tips, hue='sex')
sns.lmplot(x='total_bill', y='tip', data=tips, hue='sex',
palette='coolwarm')

Working with markers

lmplot kwargs get passed through to regplot which is a more general form of lmplot(). regplot has a scatter_kws parameter that gets passed to plt.scatter. So you want to set the s parameter in the dictionary.

sns.lmplot(x='total_bill', y='tip', data=tips, hue='sex',
palette='coolwarm', markers=['o','v'],
scatter_kws={'s':100})

Using a grid

We can add more variable separation through columns and rows with the use of a grid.

sns.lmplot(x='total_bill', y='tip', data=tips, col='sex')
sns.lmplot(x='total_bill', y='tip', row='sex', col='time',
data=tips)
sns.lmplot(x='total_bill', y='tip', data=tips, col='day',
hue='sex', palette='coolwarm')

Aspect and size

sns.lmplot(aspect=0.6, size=8)