Python programming and data analysis

Lecture 7

Seaborn for Statistical Data Visualisation + Intro to EDA

Robert Szmurło

e-mail: robert.szmurlo@ee.pw.edu.pl 2019L

Lecture outline

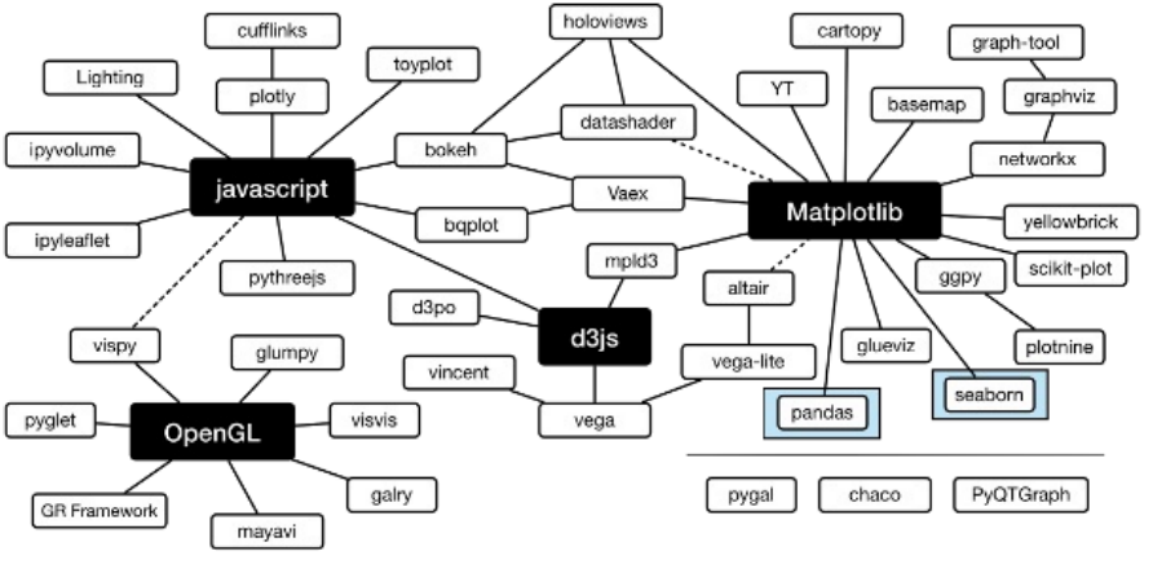
1. Introduction to Seaborn
2. Example visualisation and E-Commerce EDA

Next lecture

- Introduction to Machine Learning with Python

Python Visualization Landscape

- In Python we have plethora of options to data visualisation
- Easyly we can get lost...
- The visualization landscape is complex and overwhelming

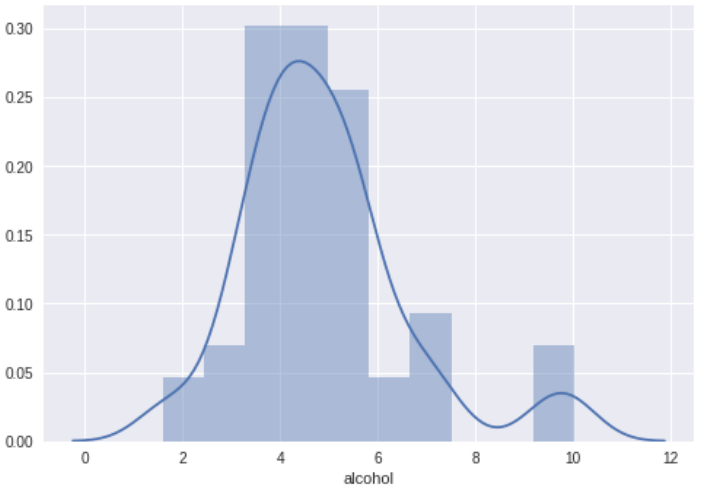


Seaborn

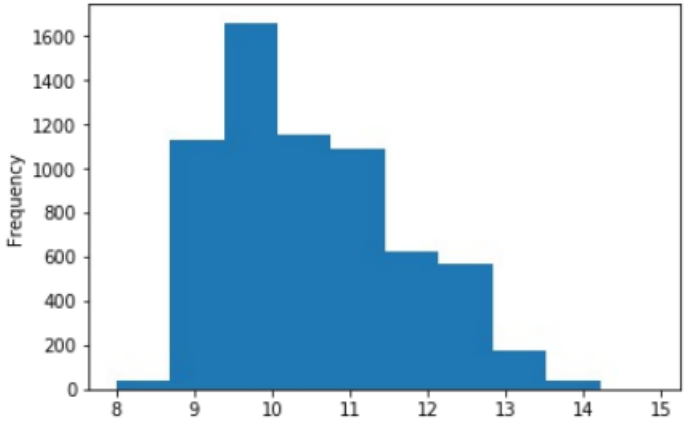
http://seaborn.pydata.org/tutorial.html

- Seaborn supports complex visualizations of data
- It is built on matplotlib and works best with pandas' dataframes
- The **distplot** is similar to the histogram from previous lectures
- By default, generates a Gaussian Kernel Density Estimate (KDE)

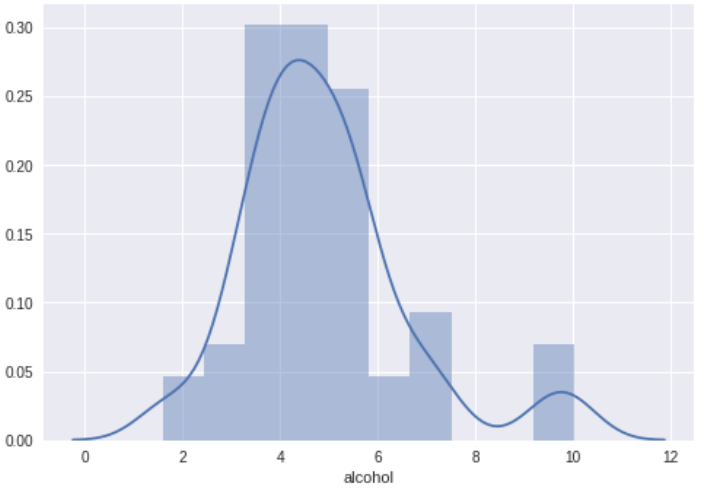
```
import pandas as pd
import numpy as np
import seaborn as sns
df = sns.load_dataset("car_crashes")
df.head()
sns.distplot(df['alcohol'])
```



Histogram vs. Distplot



- Actual frequency of
- observations
- No automatic labels
- Wide bins



- Automatic label on x axis
- Muted color palette
- KDE plot
- Narrow bins

Simple distplot with KDE

```
tips = sns.load_dataset('tips')
sns.get_dataset_names() # to show all available datasets
tips.head()
```

Even more datasets can be found on https://github.com/mwaskom/seaborn-data.

- Univariable data distribution - `sns.distplot()`

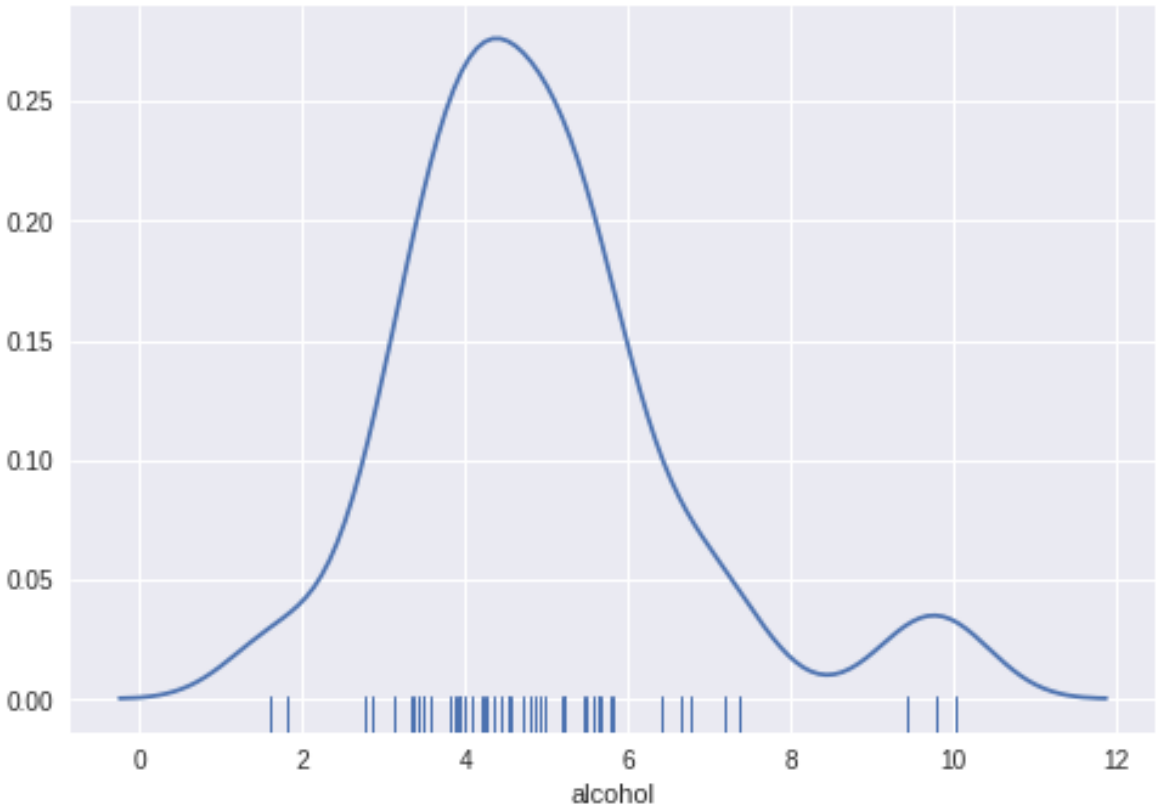
  ```
  sns.distplot(tips['total_bill'])
  sns.distplot(tips['total_bill'], bins=20)
  ```

  KDE - Kernel Density Estimation
- to much bins give weird plots, and we usually try to find a balance
- bins=30 gives a good idea of the information
- after the 20 the plots begins to fade away

Alternative data distributions

- A rug plot is an alternative way to view the distribution of data
- A KDE curve and rug plot can be combined
- We can turn off the histogram bars

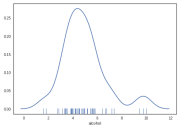
```
sns.distplot(df["alcohol"], rug=True, hist=False)
```

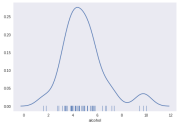


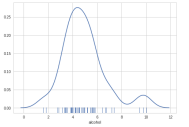
Theme examples with sns.set_style()

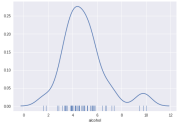
- Seaborn has default configurations that can be applied with `sns.set()`
- These styles can override matplotlib and pandas plots as well
- Reminder - Seaborn is based on Matplotlib (thats why we are using `plt.show()` to force plotting figure in Jupyter-Notebook)

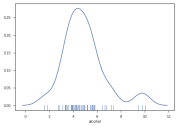
```
for style in ['white','dark','whitegrid',
              'darkgrid','ticks']:
  sns.set_style(style)
  sns.distplot(df["alcohol"], rug=True, hist=False)
  plt.show()
```











Joint plots

```
sns.jointplot( x='total_bill', y='tip', data=tips)
sns.jointplot( x='total_bill', y='tip', data=tips, size=10)
sns.jointplot( x='total_bill', y='tip', data=tips, size=10, kind='hex?reg')
```

https://seaborn.pydata.org/generated/seaborn.jointplot.html#seaborn.jointplot

- allows to join two plots (two distribution plots)
- bivariable data
- between we can see scatterplot
- as you go higher with total bill you will go higher in tip, tips are usually proportional to the bill
- kind=reg - regression line <- linear fit to the scatter data
- kde - density where these point match up the most use

Pair plots

```
sns.pairplot(tips)
sns.pairplot(tips, hue='smoker')
sns.pairplot(tips, hue='smoker', palette='muted')
```

https://seaborn.pydata.org/generated/seaborn.pairplot.html#seaborn.pairplot

- pairwise relationship in entire DataFrame (at least for the numerical columns)
- Simply: do the joint plot for avery single possible combination of the numerical columns
- do not plot large dataplots :-)
- histogram for diagonal
- hue='categorical column'
- palette - specific colors

Rug plots

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

- draws a dash mark for avery point for univariant distribution
- kind of histogram

Kernel Density Estimation

https://en.wikipedia.org/wiki/Kernel_density_estimation

Normal Gaussian Distibution for a rug plot dash mark

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

#Create dataset
dataset = np.random.randn(25)

# Create another rugplot
sns.rugplot(dataset);

# Set up the x-axis for the plot
x_min = dataset.min() - 2
x_max = dataset.max() + 2

# 100 equally spaced points from x_min to x_max
x_axis = np.linspace(x_min,x_max,100)

# Set up the bandwidth, for info on this:
url = 'http://en.wikipedia.org/wiki/Kernel_density_estimation#Practical_estimation_of_the_bandwidth'

bandwidth = ((4*dataset.std()**5)/(3*len(dataset)))**.2


# Create an empty kernel list
kernel_list = []

# Plot each basis function
for data_point in dataset:

    # Create a kernel for each point and append to list
    kernel = stats.norm(data_point,bandwidth).pdf(x_axis) # Probability Density Function
    kernel_list.append(kernel)

    #Scale for plotting
    kernel = kernel / kernel.max()
    kernel = kernel * .4
    plt.plot(x_axis,kernel,color = 'grey',alpha=0.5)

plt.ylim(0,1)
```

KDE Plot

```
# To get the kde plot we can sum these basis functions.

# Plot the sum of the basis function
sum_of_kde = np.sum(kernel_list,axis=0)

# Plot figure
fig = plt.plot(x_axis,sum_of_kde,color='indianred')

# Add the initial rugplot
sns.rugplot(dataset,c = 'indianred')

# Get rid of y-tick marks
plt.yticks([])

# Set title
plt.suptitle("Sum of the Basis Functions")
```

Categorical plots

- Seeing distributions of categorical columns and their reference to numerical columns
- Bar plots

Bar plot

https://seaborn.pydata.org/generated/seaborn.barplot.html#seaborn.barplot

```
sns.barplot(x='sex',y='total_bill',data=tips) # mean
sns.barplot(x='sex',y='total_bill',data=tips,estimator=np.std)
```

- visualsation of a group by action
- average value per category
- `estimator=np.std` this will tell you what the standard deviation of the total bill columns is per category (default is average mean)

countplot

https://seaborn.pydata.org/generated/seaborn.countplot.html#seaborn.countplot

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)
```

boxplot and violinplot

https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot

boxplots and violin plots are used to shown the distribution of categorical data. A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

```
sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow')
sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow', hue='smoker')

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

- distribution of the total bill per day
- the dots are outliers

Let we practice EDA with E-Commerce dataset

Thank you