NOVA IMS
Information Management School

3

DATA UNDERSTANDING

Machine Learning for Marketing

Acreditações e Certificações

Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa

NOVA IMS Information Management School

Summary

1. Exploratory data analysis (EDA)
2. Univariate EDA
3. Bivariate EDA
4. Multivariate EDA
5. Data quality
6. Application exercise

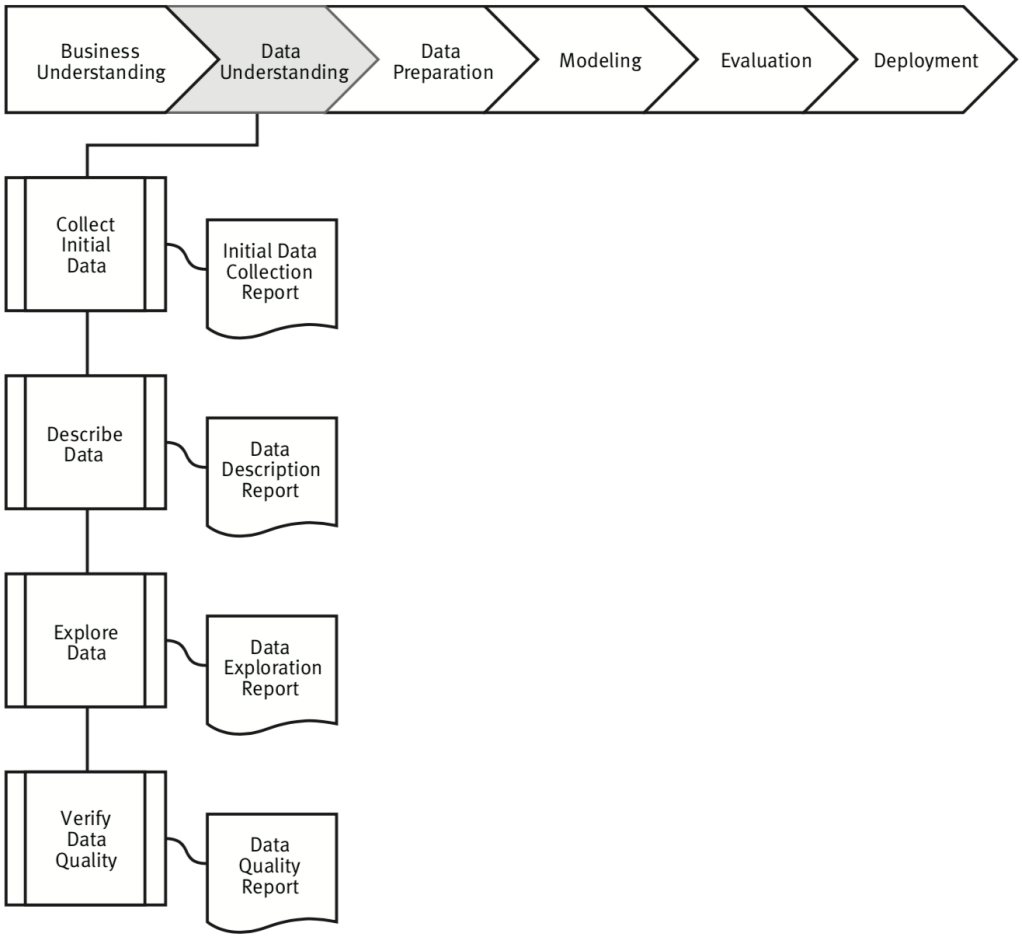
NOVA IMS Information Management School

3.1

Exploratory Data Analysis (EDA)

Data understanding

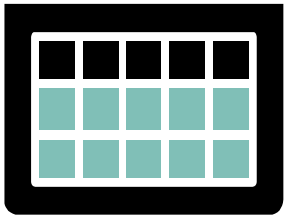
Data understanding



Data

Two-dimensional dataset:

- **Lines**: unit of analysis. For example: for customer analytics (e.g., churn analysis or segmentation) this is typically a customer. For credit fraud this maybe a transaction
- **Columns**: are the measures of the unit of analysis. For example, for customer analytics, the last date the customer renew an insurance policy, the number of days since policy subscription, etc. Columns are also called attributes, descriptors, variables, fields, or features



Exploratory Data Analysis (1/2)

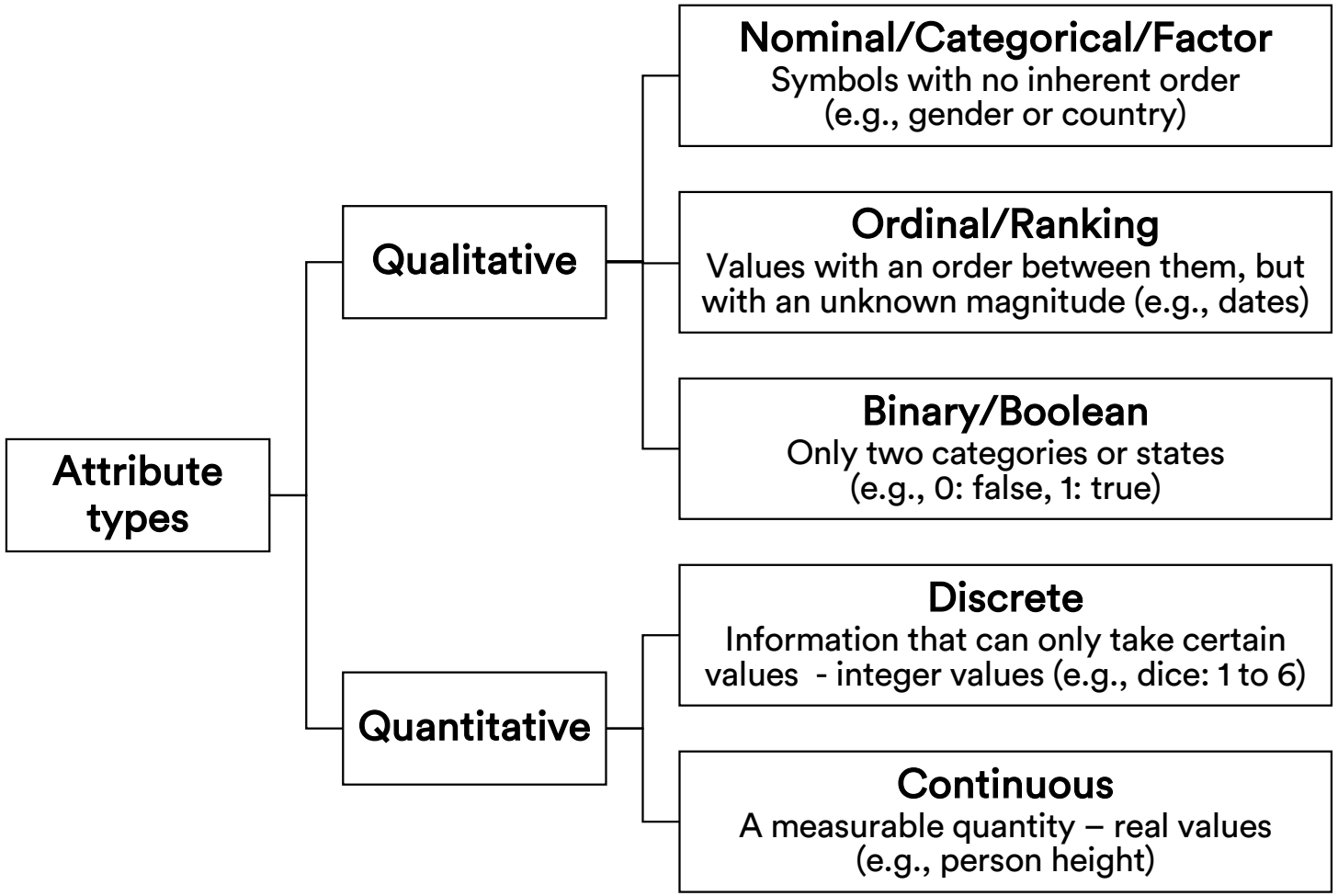
Combination of **numerical** and **visualization** techniques to **summarize data** and **provide understanding** of the dataset. EDA should identify:

- What are type of attributes or variables that make up the data
- Missing values. If so, are there patterns per variables?
- Strange minimum or maximum values
- Strange mean values or a big difference between the mean and median
- Variables with large skew or excess kurtosis (particularly if modeling algorithms assume a normal distribution)

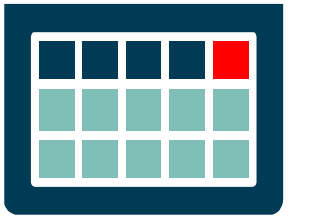
Exploratory Data Analysis (2/2)

- Gaps in distributions (bi-modal or multi-modal)
- Categorical variables with invalid or unexpected values
- Categorical variables with high cardinality
- Categorical variables where one level represents most of the observations
- Strong correlations with the target variable
- Strong correlations between the different variables

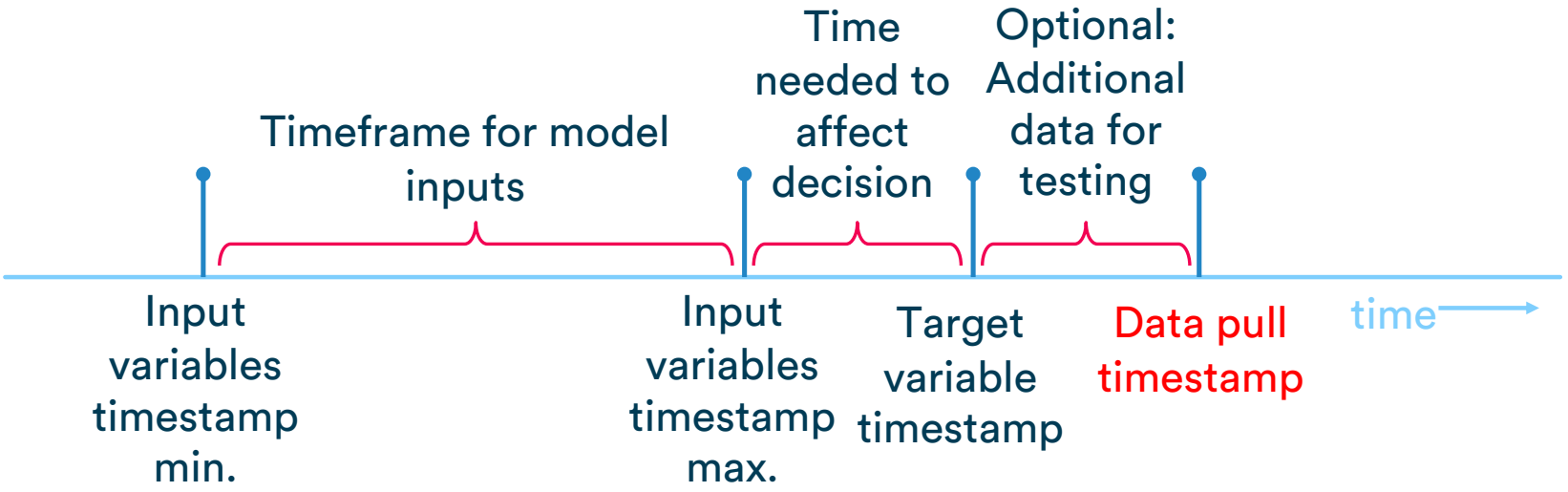
Attributes types



Target or label



Column(s) in the dataset that contains the values to be estimated or predicted as the business objectives. It can be a numeric or categorical value.



Churn prediction: Data collection example

An insurance company wants to predict at the last day of each month (t), which customers are likely to cancel their car insurance policy during the following two months (t+2). During the two months the company plan to target the customers with campaigns to decrease the cancellation likelihood.

To train the model the company will extract a dataset from its database on the 31st of March 2021.

Churn prediction: Data collection example

Approach A - When historic/snapshot data is available

Extract data from all customers in the database

label

inputs (values at… depend on the label)

Churn	Months As Customer	Claims	Subscribed policy	Age	Premium
0	23	0	A	37	€ 100
1	12	3	B	29	€ 110
1	66	0	A	34	€ 105
0	35	0	D	65	€ 140
…	…	…	…	…	…

Churn date	Inputs' values at	Label at
-	2021-01-31	2021-03-31
2020-07-07	2020-05-31	2021-03-31
2019-06-20	2019-04-30	2021-03-31
-	2021-01-31	2021-03-31
…	…	…

Churn prediction: Data collection example

Approach B - When historic/snapshot data is NOT available

Extract data in two steps (will result in having less observations)

at 2021-03-31 collect current inputs values for all non-churned (active) customers (dataset A)

dataset B

Months As Customer	Claims	Subscribed policy	Age	Premium	Churn
23	0	A	37	€ 100	0
35	0	D	65	€ 140	1
32	0	B	55	€ 120	0
67	1	C	32	€ 100	0
...	...	...	...	...	...

at 2021-05-30 collect the label value for all the customers collected in the dataset A and merge the label in the dataset

NOVA IMS Information Management School

3.2

Univariate EDA

Data understanding

NOVA IMS Information Management School

Numeric attributes

Univariate EDA

Numeric attributes

Understand if data has many missing values, outliers, and distribution patterns, using:

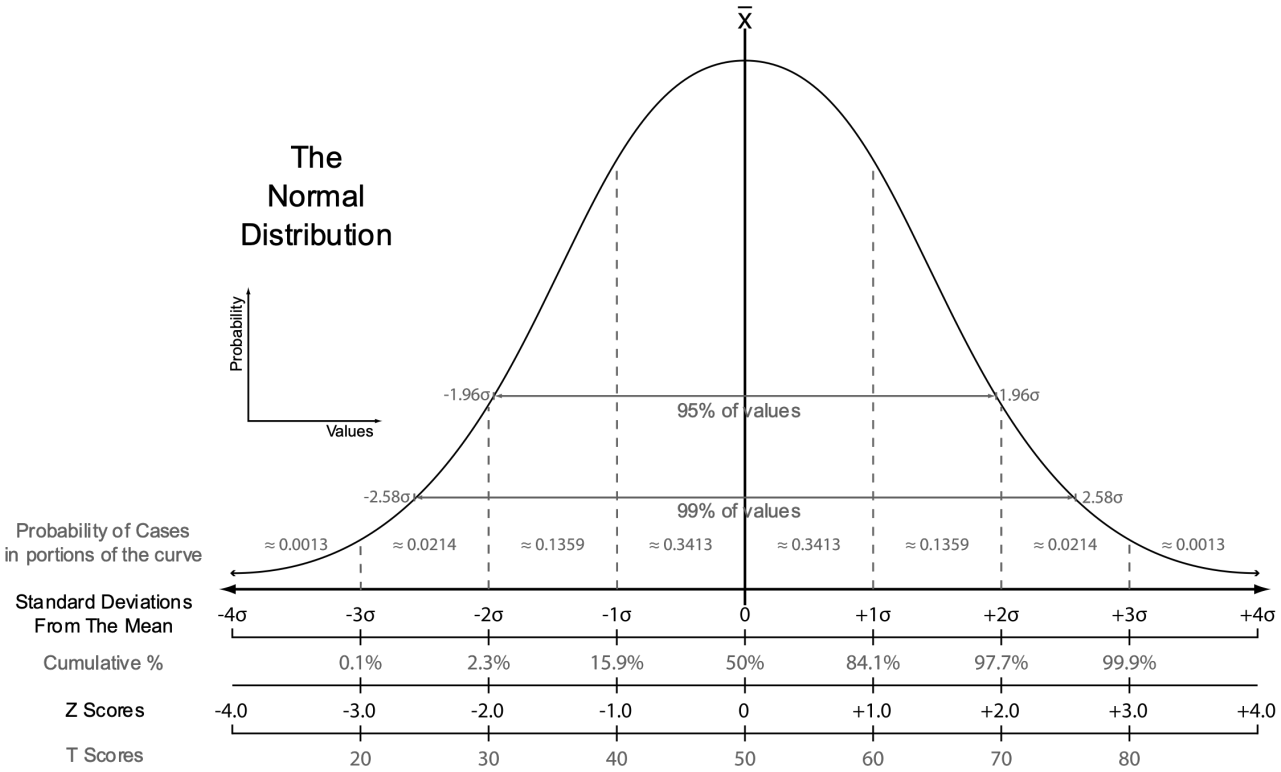
Measures of central tendency: mean, median, and mode

Measures of dispersion: range, quartiles, variance, standard deviation and IQ

Distribution measures: skewness, kurtosis

Normal attributions

Many algorithms assume normal distributions



source: https://en.wikipedia.org

Measures of central tendency

Mean: arithmetic average ($\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$)

The mean is sensitive to extremes values (outliers)

Example: [2, 3, 4, 7, 8, 8, 9]=41/7=5.86

Median: the value that separates the higher half from the lower half of data

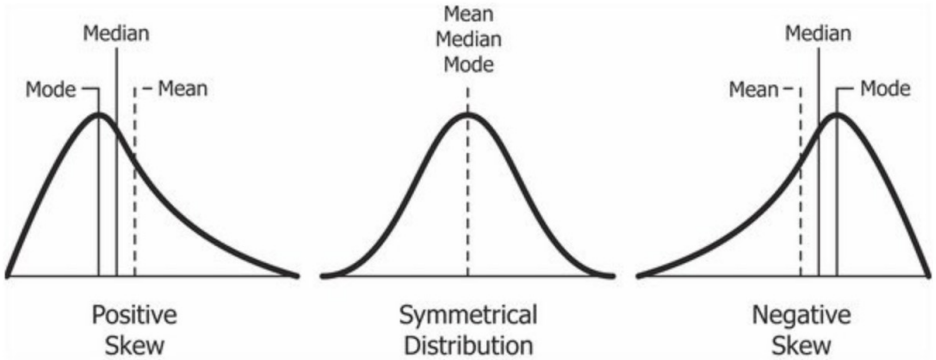
Examples: [2, 3, 4, **7**, 8, 8, 9]=7 or [1, 2, 3, **4**, **7**, 8, 8, 9]=(4+7)/2=5.5

More appropriate to measure the center in skewed data

Mode: the most frequent value

Example: [2, 3, 4, 7, **8**, **8**, 9]=8

Data with one, two, or mode modes are called unimodal, bimodal, or trimodal (with two or more are usually called multimodal)

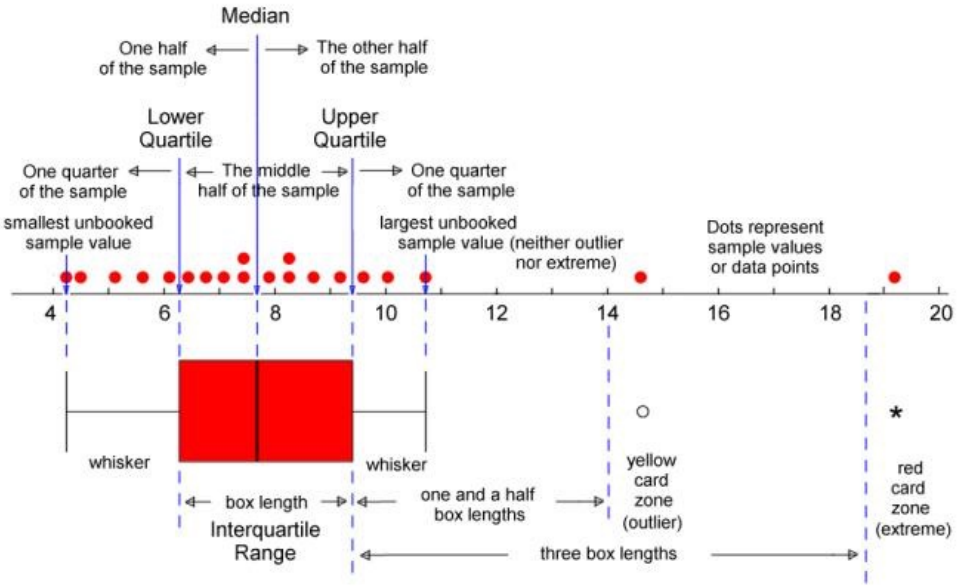


source: https://en.wikipedia.org

Outliers

Observations that lie outside the overall pattern of a distribution

⚠ Not all outliers are extreme values (e.g., near zero corporate yearly earnings)



source: http://web.pdx.edu

Measures of dispersion
Standard deviation and Variance

Standard deviation and **Variance** are measures of data dispersion

Variance: $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$

Standard deviation: $\sigma = \sqrt{\sigma^2}$

A low standard deviation means that the data observations tend to be very close to the mean

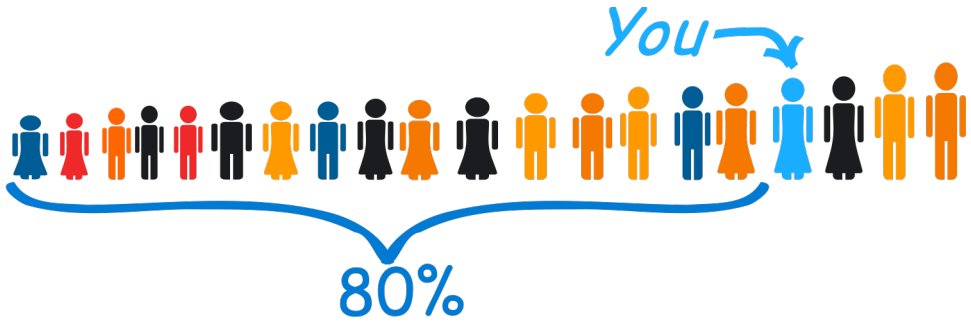
250 marathoners

250 airplane passengers

Mean weight in both datasets: 70 Kg.
In which dataset should the standard deviation be smaller (closer to the mean)?

Measures of dispersion
Percentiles

- Percentile is the value at which a percentage of data falls



source: https://mathisfun.prg

- Deciles are similar to percentiles, but they split data into 10% groups
- Quartiles split the data into quarters

NOVA IMS Information Management School

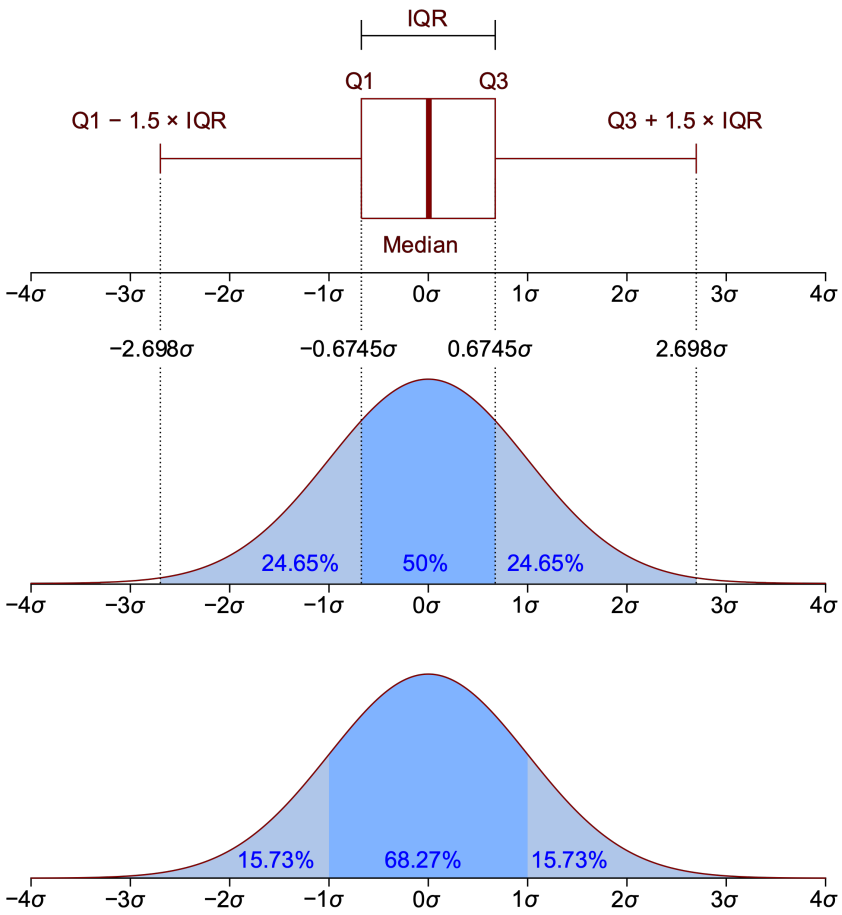
Measures of dispersion

Range, Quartiles, and IQR

Range: max. value – min. value

Quartiles: points taken at regular intervals of a data distribution, dividing it four equal-size consecutive sets (a percentile divide the data into 100 equal-sized consecutive sets)

IQR (InterQuartile Range): measure of spread that is calculated with the formula Q3-Q1



source: https://en.wikipedia.org

Distribution measures
Measures of shape

Kurtosis: measure of the "tail" of the distribution. Given by the formula $\frac{\frac{\sum_{i=1}^{N}(x_i-\bar{x})^4}{N}}{\sigma^4}$

A normal distribution has a kurtosis of 3

A distribution with a kurtosis <3 is platykurtic (compared to a normal distribution its tails are shorter and thinner)

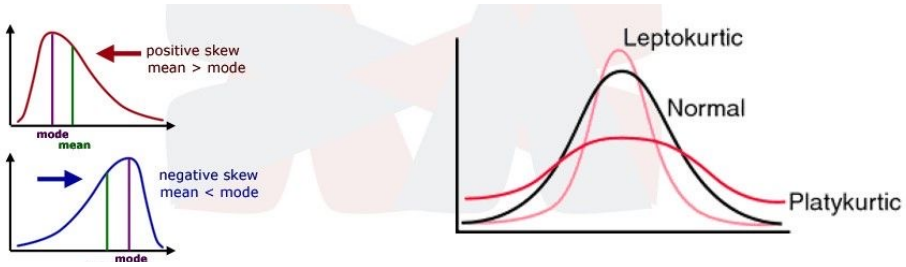
A distribution with a kurtosis >3 is leptokurtic (compared to a normal distribution its tails are longer and fatter)

Skewness: measure of the asymmetry of the distribution. Given by the formula $\sqrt{N}\frac{\sum_{i=1}^{N}(x_i-\bar{x})^3}{\left(\sum_{i=1}^{N}(x_i-\bar{x})^2\right)^{\frac{3}{2}}}$

If less than -1 or greater than +1, the distribution is highly skewed

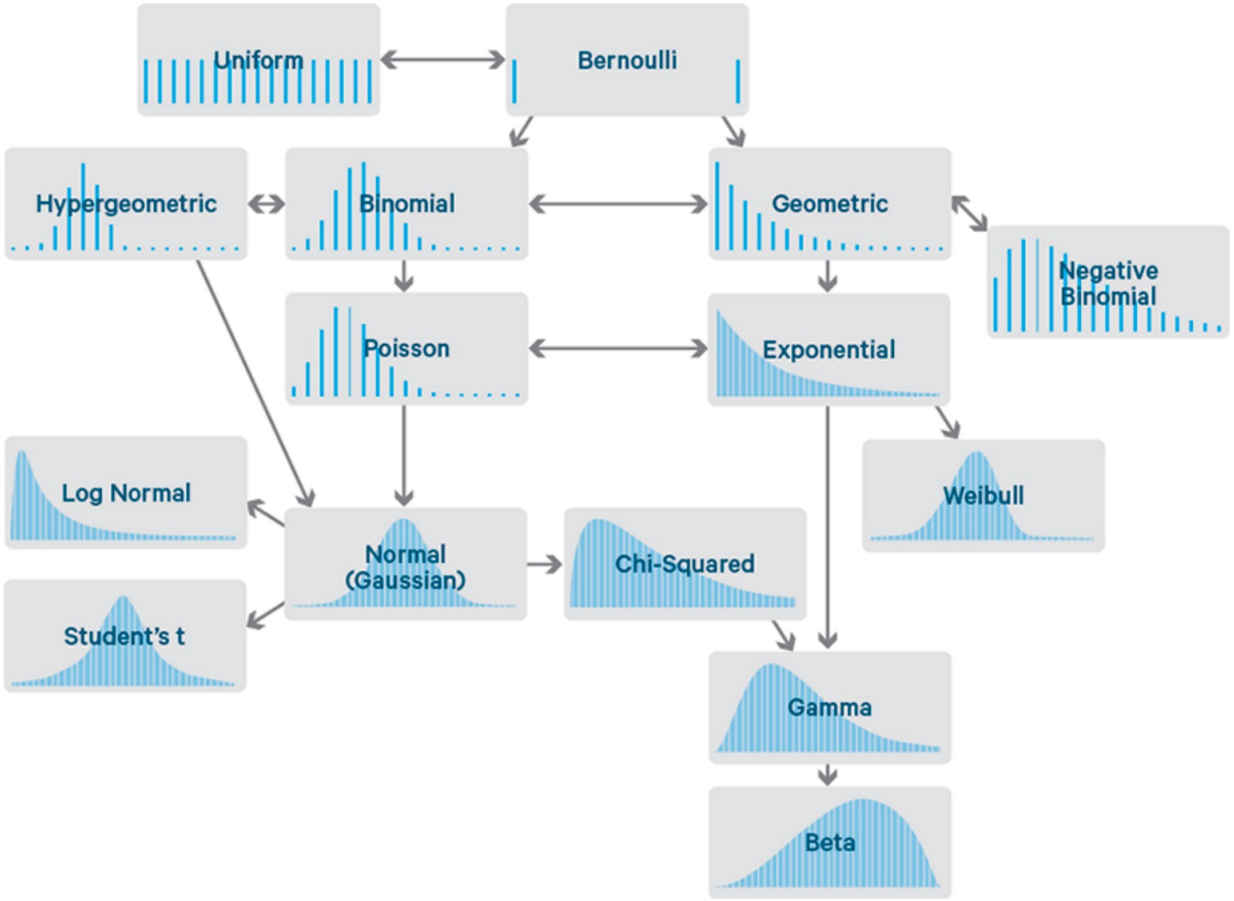
If is between -1 and -0.5 or between +0.5 and +1, the distribution is moderately skewed

If is between -0.5 and +0.5, the distribution is approximately symmetric



source: www.vasantkothari.com

DISTRIBUTIONS



source: https://rstudio-pubs-static.s3.amazonaws.com/296091_5a01e2f5ae474b98a67d6526a50d3a3d.html

Missing values

Missing values are probably one of the most challenging data problems. First, because ”missing values” can be represented in a myriad of ways

Possible representation	Description
Null, or empty string (“”)	For numerical or categorial variables
NaN	For numerical variables
0	For numerical variables that are never equal to zero
-1	For numerical variables that are never negative
99, 999, -99, 999, …	For numeric variables that have values less or more than x
0000-000, 999999999, 11/11/11	For zip codes, telephone number, dates or other specific size variables

NOVA IMS Information Management School

Categorical attributes

Univariate EDA

Categorical attributes

Verify the frequency of each level. High cardinality (too many different values/levels) usually means that some levels have very few observations, which:

- Makes inferences made by the algorithms about the relationship between the target and the level to be noisy (at least)
- May lead models to overfit
- May difficult and slowdown model training

NOVA IMS Information Management School

Data visualization

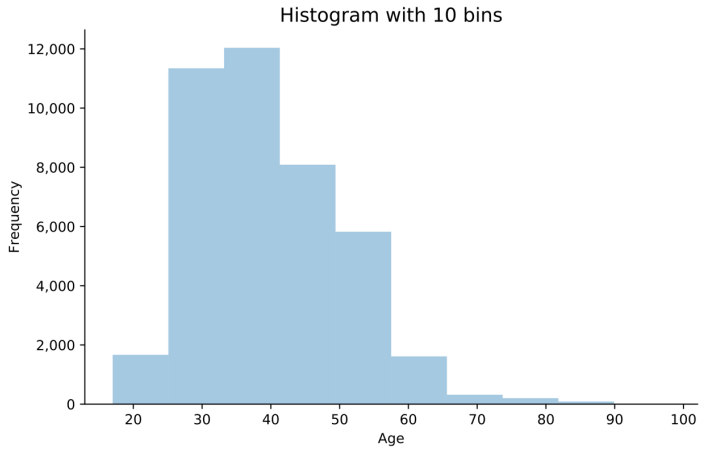
Univariate EDA

Data visualization

Take advantage of histograms, density plots, and other data visualizations to better the data

To optimize histograms:

- Change the number of bins
- Change the y-axis scale



NOVA IMS Information Management School

3.3

Bivariate EDA

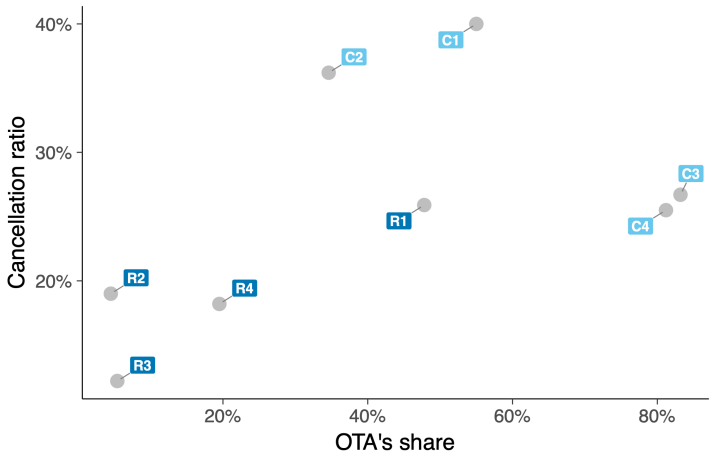
Data understanding

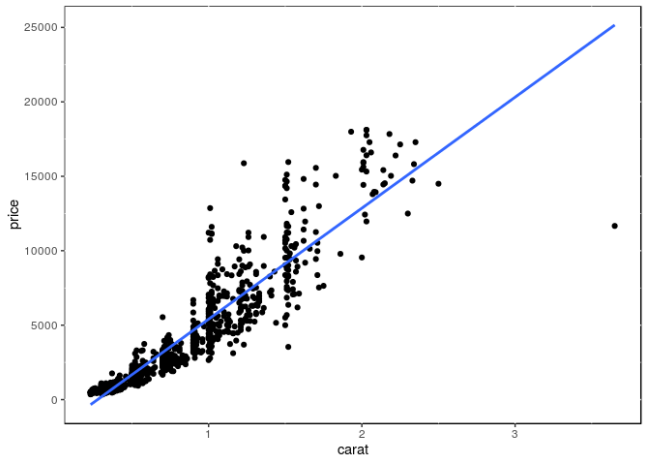
Scatter plots

Good for showing the relationship between numerical variables.

Overall pattern: can display linear, curvy, or exponential patterns

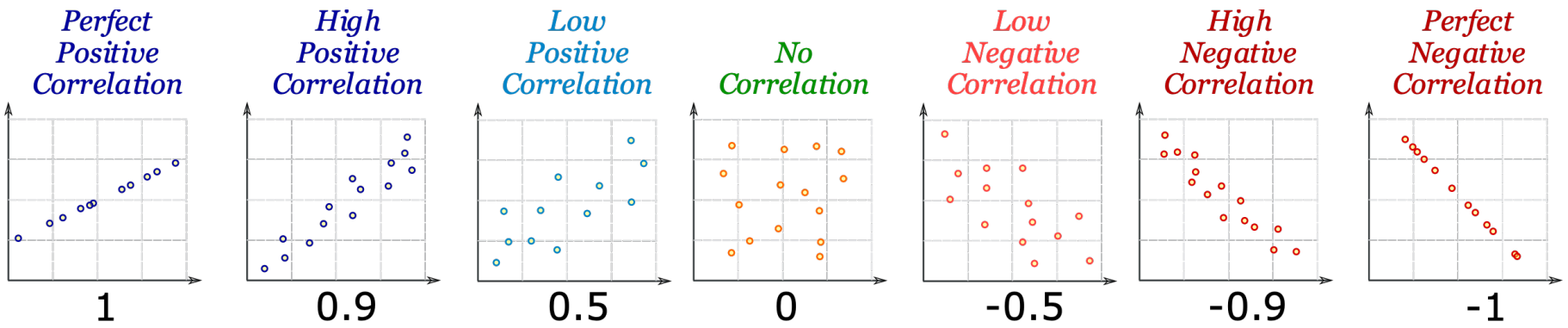
Strength/noise: how closely the points follow the pattern. Noise will be displayed in the form of the amount of deviation or dispersion of the points in relation to the overall pattern





Correlations (1/4)

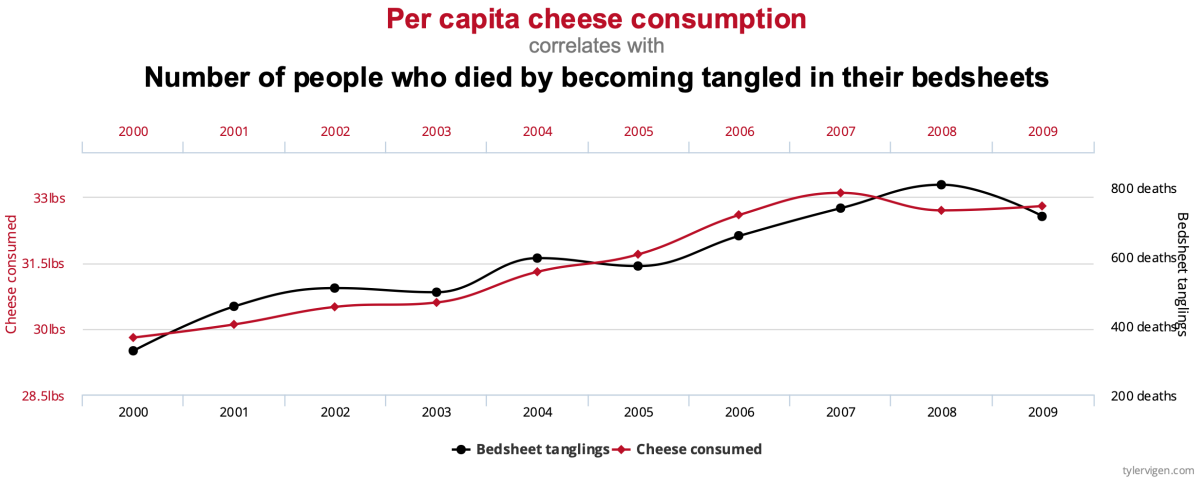
The Pearson correlation coefficient is the most popular tool for comparing two numerical features. It is a numerical indication of the strength of linear association between two numerical features

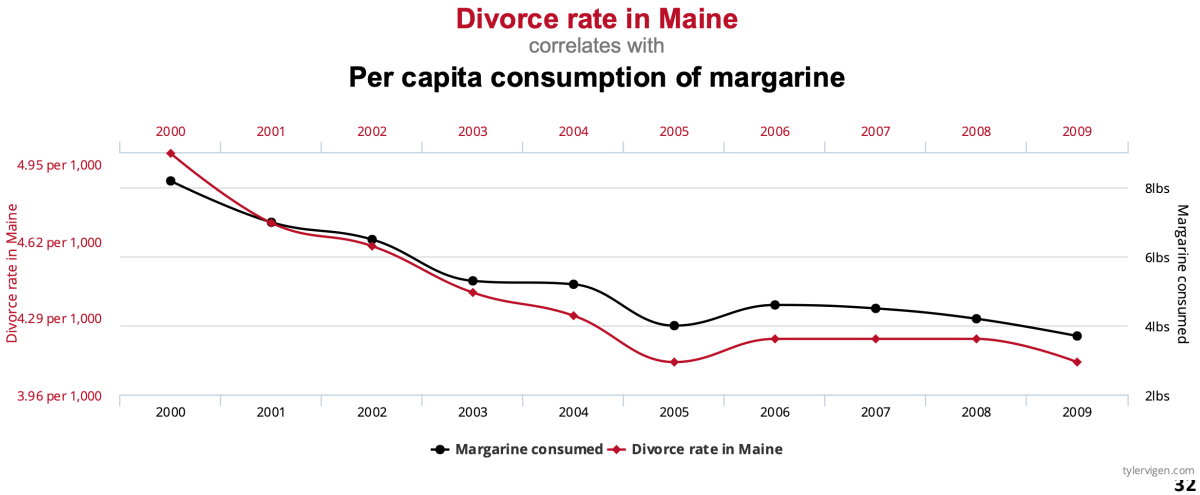


source: https://www.mathsisfun.com

Correlations (2/4)

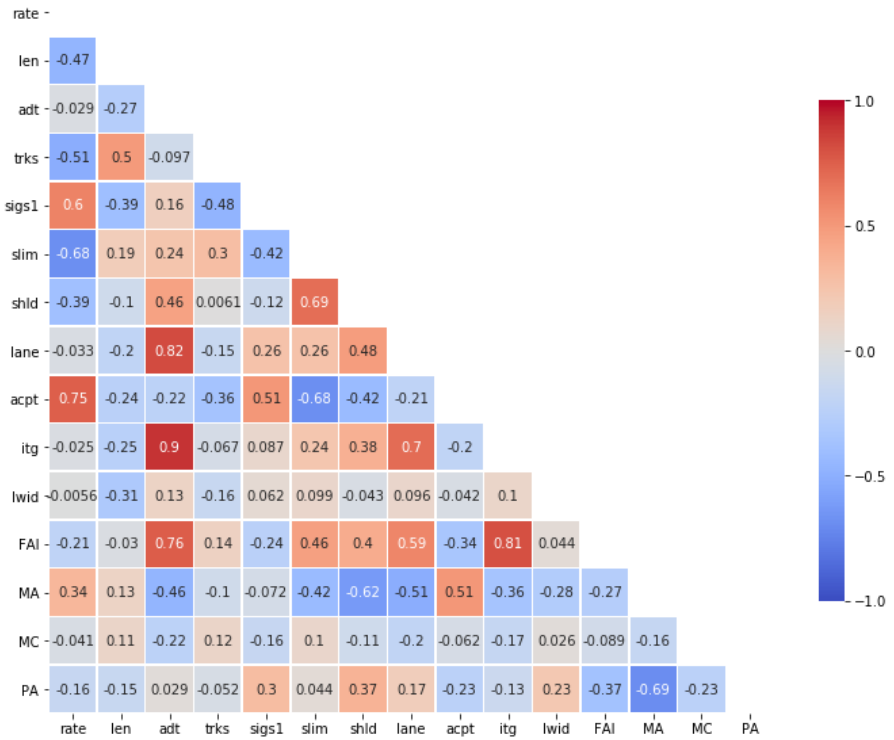
⚠ The existence of linear correlation between two variables does not mean one variable's meaning is related to another's





Correlations (3/4)

⚠ If the relationship between the variables is nonlinear, then the coefficient could be misleading



Correlations (4/4)

Correlations' coefficients are not only for numeric variables

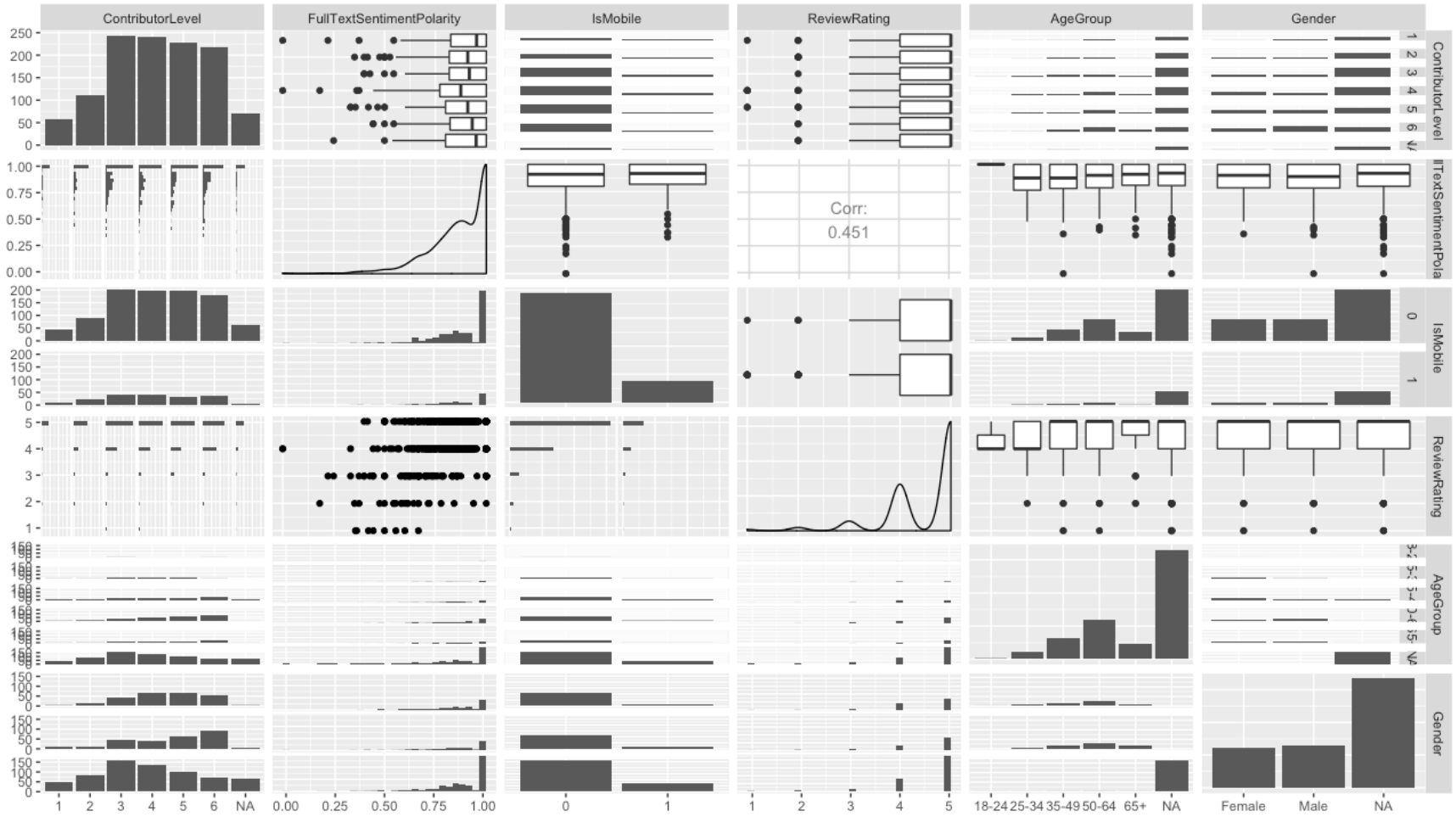
	Numeric X	Ranking X	Categorical X
Numeric Y	Pearson	Biserial	Point Biserial
Ranking Y	Biserial	Spearman Rho	Rank Biserial
Categorical Y	Point Biserial	Rank Biserial	Phi, L, C, Lambda

Cross-tabulations (Cross tabs)

Counts of the intersection between two variables (typically categorical variables)

DistributionChannel MarketSegment	Corporate	Direct	GDS	TA/TO
Aviation	254	0	0	4
Complementary	9	332	0	63
Corporate	1687	39	0	471
Direct	4	8369	4	143
Groups	330	107	1	6583
Offline TA/TO	96	21	36	10608
Online TA	28	71	461	37208

Pair plots


ContributorLevel
FullTextSentimentPolarity
IsMobile
ReviewRating
AgeGroup
Gender
Corr:
0.451
18-24
25-34
35-49
50-64
65+
Female
Male

NOVA IMS Information Management School

3.4

Multivariate EDA

Data understanding

Duplicates

It is common for real-world datasets to have duplicate instances, even when they should not exist (e.g., having two instances of the same customer profile)

- Check if there are exact-match duplicates, with all columns having the same values
- Check if there are rows with the same IDs (e.g., having two customer profiles with the same email address)

Redundancy

Redundancy is an important issue in data quality. An attribute is redundant when it can be derived from another attribute or set of attributes. For example, BMI (Body Mass Index) is usually highly correlated to a person's weight

Statistic tests to verify redundancy:

- Numerical data:
  - Pearson correlation coefficient (linear relationship)
  - Covariance
- Numerical or ordinal data:
  - Spearman correlation coefficient (monotonic relationship)
- Categorical data:
  - Chi-square (x^2)

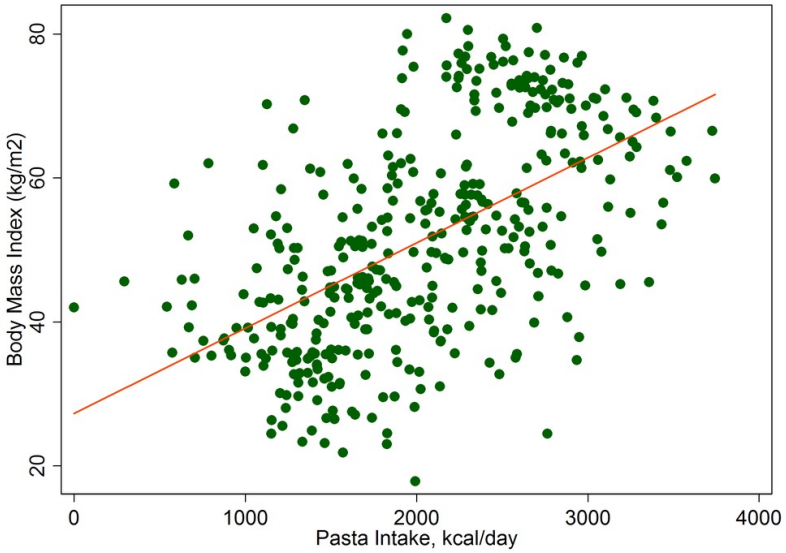
Data visualization

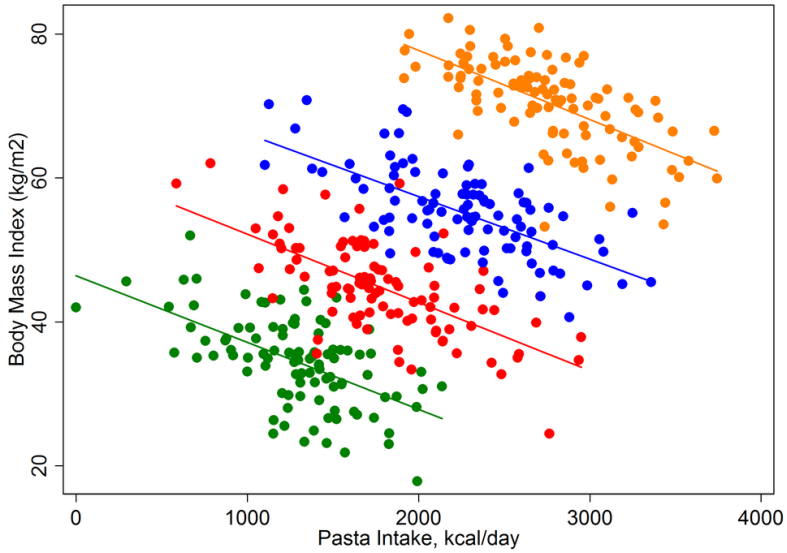
Visualizing data in two or more dimensions provides additional insights into data

⚠Due to the number of possible combinations, the number of plots can become unmanageable. **Domain knowledge is required to select which relationships to explore**

Simpson's paradox

A trend seen in individual variables disappears or is reversed when variables are combined





NOVA IMS Information Management School

3.5

Data quality

Data understanding

Data quality

Data must have **quality** to satisfy the requirements of use.

Factors comprising data quality are…

Data quality

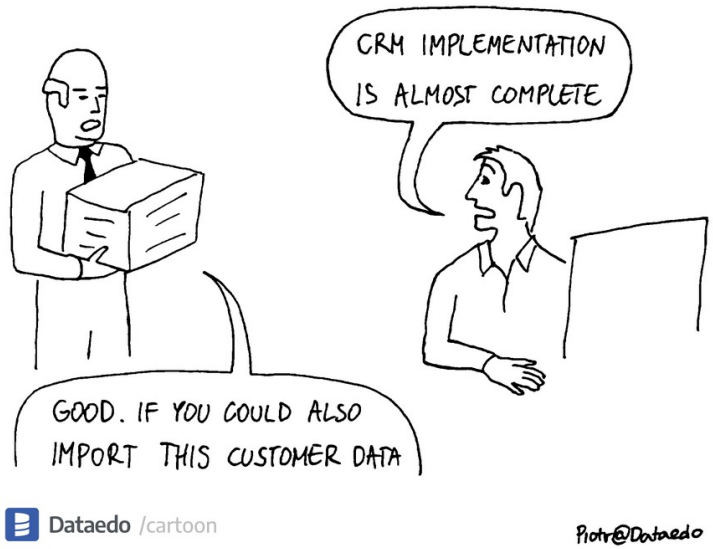
Accuracy

Data must be accurate. For example, the customer birthdate recorded as 1917-01-01, instead of 1971-01-01

Data quality

Completeness

Data should be complete. For example, legacy systems not considering important fields as mandatory



Data quality

Consistency

Data should be consistent. For example, a “gender” field should not accept values “M”, “m”, “Male” or “male” for “male”

Data quality

Timeliness

Data should be available when necessary for mining. For example, if a campaign is to be created based on customers purchases, the most recent purchases should be used, not purchases from a year ago

Data quality

Believability

Users should believe in data. For example, if users know that an analysis is made based on improperly created data, they will not believe in the analysis itself

Data quality

Interpretability

Data should be easy to interpret. For example, if there is a field “Sales”, the documentation should state if the value includes taxes or not

NOVA IMS Information Management School

3.6

Application exercise

Data understanding

Business problem

For an insurance company to make money, it needs to collect more in yearly premiums than it spends on medical care to its beneficiaries. As a result, insurers invest a great deal of time and money to develop models that accurately forecast medical expenses.

Medical expenses are difficult to estimate because the costliest conditions are rare and seemingly random. Still, some conditions are more prevalent for certain segments of the population. For instance, lung cancer is more likely among smokers than non-smokers, and heart disease may be more likely among the obese.

[use case from Lantz, B (2013)]

Business objective

Estimate the medical care expenses per individual. These estimates could be used to create actuarial tables which set the price of yearly premiums higher or lower depending on the expected treatment costs

Understanding key drivers of the estimates

NOVA IMS Information Management School

Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The **Pandas** library is built on NumPy and provides easy-to-use **data structures** and **data analysis** tools for the Python programming language.

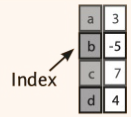
Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

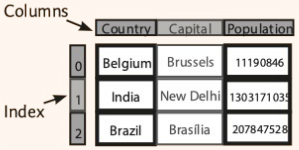
Series

A **one-dimensional** labeled array capable of holding any data type




```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame



A **two-dimensional** labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
-5
```
Get one element

```
>>> df[1:]
   Country    Capital  Population
1   India  New Delhi  1303171035
2  Brazil   Brasília   207847528
```
Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
```
Select single value by row & column

```
>>> df.iat([0],[0])
'Belgium'
```

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
```
Select single value by row & column labels

```
>>> df.at([0], ['Country'])
'Belgium'
```

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital    Brasilia
Population  207847528
```
Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasília
```
Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```
Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
```
Series `s` where value is not >1

```
>>> s[(s < -1)  |  (s > 2)]
```
`s` where value is <-1 or >2

```
>>> df[df['Population']>1200000000]
```
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```
Set index `a` of Series `s` to 6

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

```
>>> df.to_sql('myDf', engine)
```

Dropping

```
>>> s.drop(['a', 'c'])
```
Drop values from rows (axis=0)

```
>>> df.drop('Country', axis=1)
```
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index()
```
Sort by labels along an axis

```
>>> df.sort_values(by='Country')
```
Sort by the values along an axis

```
>>> df.rank()
```
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
```
(rows,columns)

```
>>> df.index
```
Describe index

```
>>> df.columns
```
Describe DataFrame columns

```
>>> df.info()
```
Info on DataFrame

```
>>> df.count()
```
Number of non-NA values

Summary

```
>>> df.sum()
```
Sum of values

```
>>> df.cumsum()
```
Cummulative sum of values

```
>>> df.min()/df.max()
```
Minimum/maximum values

```
>>> df.idxmin()/df.idxmax()
```
Minimum/Maximum index value

```
>>> df.describe()
```
Summary statistics

```
>>> df.mean()
```
Mean of values

```
>>> df.median()
```
Median of values

Applying Functions

```
>>> f = lambda x: x*2
```

```
>>> df.apply(f)
```
Apply function

```
>>> df.applymap(f)
```
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b     NaN
c     5.0
d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

NOVA IMS Information Management School

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))
```

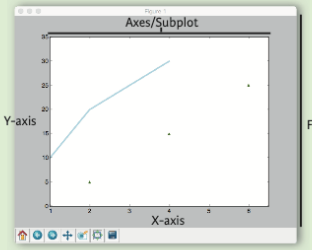
Axes

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

```
>>> 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 0
```

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 2D vector fields
```

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
```

DataCamp
Learn Python for Data Science Interactively

NOVA IMS Information Management School

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")
>>> sns.factorplot(x="pclass",
                   y="survived",
                   hue="sex",
                   data=titanic)
>>> sns.lmplot(x="sepal_width",
               y="sepal_length",
               hue="species",
               data=iris)
```
Subplot grid for plotting conditional relationships

Draw a categorical plot onto a Facetgrid

Plot data and regression model fits across a FacetGrid

```
>>> h = sns.PairGrid(iris)
>>> h = h.map(plt.scatter)
>>> sns.pairplot(iris)
>>> i = sns.JointGrid(x="x",
                      y="y",
                      data=data)
>>> i = i.plot(sns.regplot,
               sns.distplot)
>>> sns.jointplot("sepal_length",
                  "sepal_width",
                  data=iris,
                  kind='kde')
```
Subplot grid for plotting pairwise relationships
Plot pairwise bivariate distributions
Grid for bivariate plot with marginal univariate plots

Plot bivariate distribution

Categorical Plots

Scatterplot

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

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)
>>> sns.boxplot(data=iris,orient="h")
```
Boxplot

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")
>>> h.set(xlim=(0,5),
          ylim=(0,5),
          xticks=[0,2.5,5],
          yticks=[0,2.5,5])
```
Remove left spine
Set the labels of the y-axis
Set the tick labels for x
Set the axis labels

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

DataCamp
Learn Python for Data Science Interactively

NOVA IMS Information Management School

Predict medical expenses

1. Copy from the datasets folder copy the dataset “medical_expenses.csv”
2. Copy and open the Jupyter notebook “PredictMedicalExpenses_Init.ipynb”
3. Follow the presentation of the notebook, answer questions, and explore the challenges

Questions?

Machine Learning for Marketing

Acreditações e Certificações

Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa