

# RESEARCH METHODS FOR DATA SCIENCE

Assoc. Prof. Dr. Salha Alzahrani  
Department of Computer Science

# Step 7: Processing and displaying your data



## My Notes

~data exploration

~data  
preparation:  
cleaning /  
reduction

~visualization



# EXPLORE THE DATA

---



# Basic questions for data exploration

---

## Q #1 : Is the data organized or not?

- We are checking for whether or not the data is presented in a row/column structure.
- A general rule of thumb is that if we have unorganized data, we want to transform it into a row/column structure. For example, earlier in Ch2., we looked at ways to transform text into a row/column structure by counting the number of words/phrases.

# Basic questions for data exploration

---

## Q #2 : What does each row represent?

- Once we have an answer to how the data is organized and are now looking at a nice row/column based dataset, we should identify what each row actually represents.
- This step is usually very quick, and can help put things in perspective much more quickly.

# Basic questions for data exploration

---

## Q #3 : What does each column represent?

- We should identify each column by the level of data and whether or not it is quantitative/qualitative, and so on.
- This categorization might change as our analysis progresses, but it is important to begin this step as early as possible.

# Basic questions for data exploration

---

## Q #4 : Are there any missing data points?

- Data isn't perfect.
- Sometimes we might be missing data because of human or mechanical error.
- When this happens, we, as data scientists, must make decisions about how to deal with these discrepancies.

## Data Cleaning

---

- Even when the data is in the standard form it cannot be assumed that it is error free.
- **How errors occur?**

In real-world datasets, erroneous values can be recorded for a variety of reasons, including measurement errors, subjective judgements and malfunctioning/misuse of automatic recording equipment.



## What types of errors?

Erroneous values can be divided into those which are possible values of the attribute and those which are not.

- A **noisy value** is the value that is valid for the dataset, but is incorrectly recorded. For example, the number 69.72 may accidentally be entered as 6.972, or a categorical attribute value such as *brown* may accidentally be recorded as another of the possible values, such as *blue*.
- An **invalid value** is the value that is invalid for the dataset, such as 69.7X for 6.972 or *bbrown* for *brown*.

## How to clean data?

---

- Using software tools, especially to give an overall visual impression of the data, when some anomalous values or unexpected concentrations of values may stand out.
- Using some basic analysis of the values. For example, sorting the values into ascending order (which for fairly small datasets can be accomplished using just a standard spreadsheet) may reveal unexpected results.

## Missing Values

---

- In many real-world datasets, data values are not recorded for all attributes.
- **How missing data occur?**
  - This can happen simply because there are some attributes that are not applicable for some instances (e.g. certain medical data may only be meaningful for female patients).
  - It can also happen that there are attribute values that should be recorded that are missing. This can occur for several reasons, for example, a malfunction of the equipment used to record the data.

# How to deal with missing data?

## Strategy 1: Discard Instances

---

- This is the simplest strategy: delete all instances where there is at least one missing value and use the remainder.
- **Advantage:**
  - This strategy is a very **conservative** one, which has the advantage of avoiding introducing any data errors.
- **Disadvantage:**
  - Discarding instances may **damage the reliability** of the results derived from the data.
  - **Not usable** when all or a high proportion of all the instances have missing values.

# How to deal with missing data?

## Strategy 2: Replace by Most Frequent/Average Value

- Another approach is to estimate each of the missing values using the values that are present in the dataset.
  - For **categorical attribute**: use its most frequently occurring value.
  - For **continuous attribute**: use the average value.
- **Advantage:**
  - Replacing missing values by most frequent/average value may **preserve the reliability** of the results derived from the data.
- **Disadvantage:**
  - This strategy may **introduce noise** into the data if the proportion of missing values for a variable is big.



# Basic questions for data exploration

---

## Q #5 : Do we need to perform any transformations on the columns?

- Depending on what level/type of data each column is at, we might need to perform certain types of transformations.
- For example, for the sake of statistical modeling and machine learning, we would like each column to be numerical.
- Of course, we will use Python to make any and all transformations.

## How big number of attributes may occur?

- In some data mining applications, the availability of larger storage capacity has led to large numbers of attribute values being stored for every instance, e.g. information about all the purchases made by a supermarket customer for three months.
- For some datasets, there can be substantially more attributes than there are instances. Many of those attributes are **irrelevant** to the data mining application.

## Why we need to reduce the number of attributes?

---

- Irrelevant attributes will place an unnecessary computational overhead on any data mining algorithm.
- At worst, they may cause the algorithm to give poor results.

## How we can reduce the number of attributes?

- There are several methods in which the number of attributes (or 'features') can be reduced before a dataset is processed. These methods are called **feature reduction** or **dimension reduction** such as principle component analysis (PCA) and others.

# Preprocessing data

The `sklearn.preprocessing` package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.



# Preprocessing data

Standardization (or mean removal )

**Standardization** of datasets is a **common requirement for many machine learning estimators** implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with **zero mean and unit variance**.

In practice we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation.





# Preprocessing data

```
7
8 from sklearn import preprocessing
9 import numpy as np
10
11 X_train = np.array([[ 1., -1.,  2.],
12                    [ 2.,  0.,  0.],
13                    [ 0.,  1., -1.]])
14
15 X_scaled = preprocessing.scale(X_train)
```

X\_train - NumPy array

	0	1	2
0	1	-1	2
1	2	0	0
2	0	1	-1

Format Resize ☒ Background color

Save and Close Close

X\_scaled - NumPy array

	0	1	2
0	0	-1.22474	1.33631
1	1.22474	0	-0.267261
2	-1.22474	1.22474	-1.06904

Format Resize ☒ Background color

Save and Close Close



# Preprocessing data

Scaled data has zero mean and unit variance:

```
In [4]: X_scaled.mean(axis=0)
```

```
Out[4]: array([0., 0., 0.])
```

```
In [5]: X_scaled.std(axis=0)
```

```
Out[5]: array([1., 1., 1.])
```



# Preprocessing data

## Scaling to a range

An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value of each feature is scaled to unit size.

This can be achieved using [MinMaxScaler](#) or [MaxAbsScaler](#), respectively.



# Preprocessing data

```
7 from sklearn import preprocessing
8 import numpy as np
9
10 #scaling to a range [0,1]
11 X_train = np.array([[ 1., -1.,  2.],
12                    [ 2.,  0.,  0.],
13                    [ 0.,  1., -1.]])
14
15 min_max_scaler = preprocessing.MinMaxScaler()
16
17 X_train_minmax = min_max_scaler.fit_transform(X_train)
18
19
```

X\_train - NumPy array

	0	1	2
0	1	-1	2
1	2	0	0
2	0	1	-1

Format Resize ☒ Background color Save and Close Close

X\_train\_minmax - NumPy array

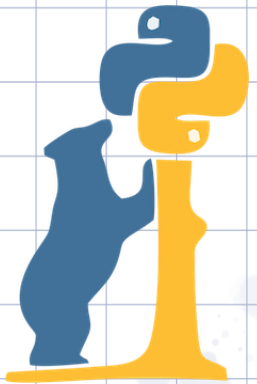
	0	1	2
0	0.5	0	1
1	1	0.5	0.333333
2	0	1	0

Format Resize ☒ Background color Save and Close Close

My Notes



Pandas



# Preprocessing data

## Normalization

**Normalization** is the process of **scaling individual samples to have unit norm**. This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

The function `normalize` provides a quick and easy way to perform this operation on a single array-like dataset, either using the  $l_1$  or  $l_2$  norms:





# Preprocessing data

```
8 from sklearn import preprocessing
9 import numpy as np
10
11 #normalization
12 X_train = np.array([[ 1., -1.,  2.],
13                    [ 2.,  0.,  0.],
14                    [ 0.,  1., -1.]])
15
16 X_normalized = preprocessing.normalize(X_train, norm='l2')
17
```

X\_train - NumPy array

	0	1	2
0	1	-1	2
1	2	0	0
2	0	1	-1

Format    Resize    ☒ Background color

Save and Close    Close

X\_normalized - NumPy array

	0	1	2
0	0.408248	-0.408248	0.816497
1	1	0	0
2	0	0.707107	-0.707107

Format    Resize    ☒ Background color

Save and Close    Close

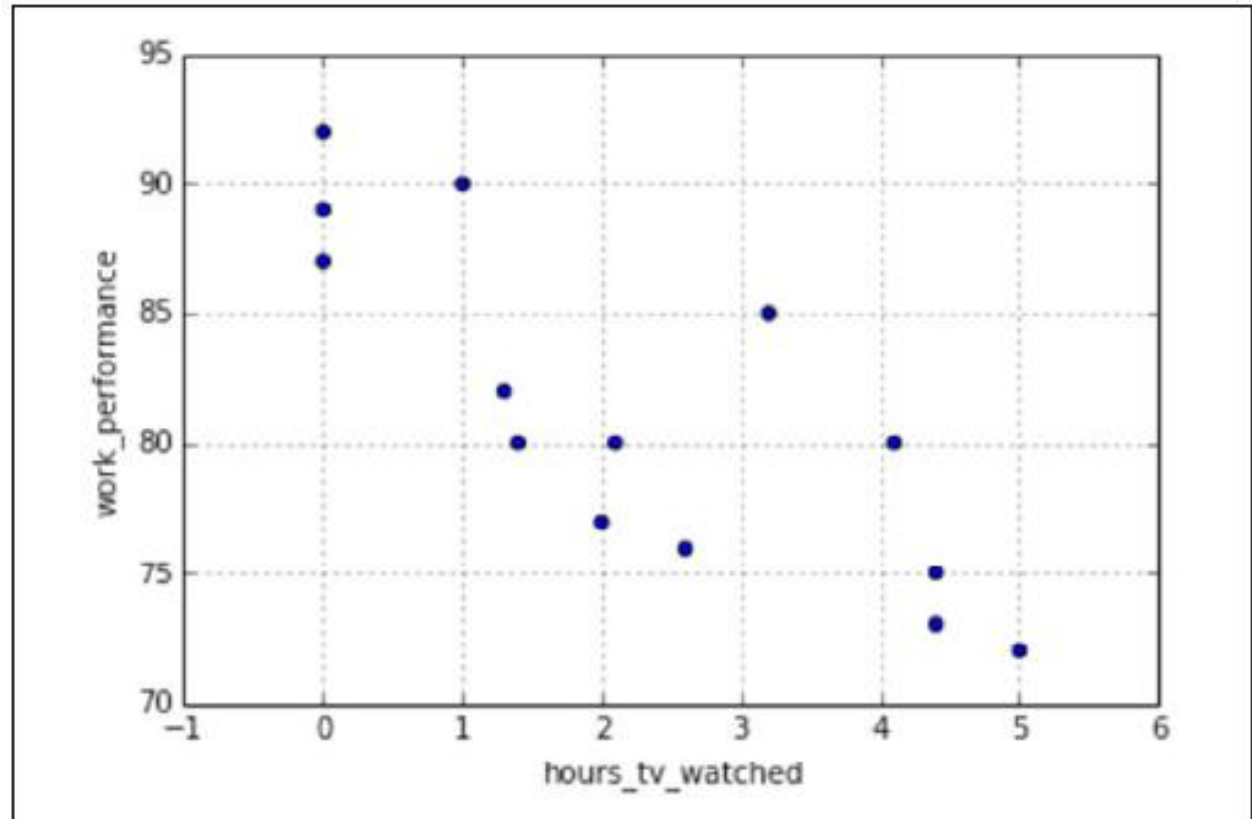


# VISUALIZE THE DATA



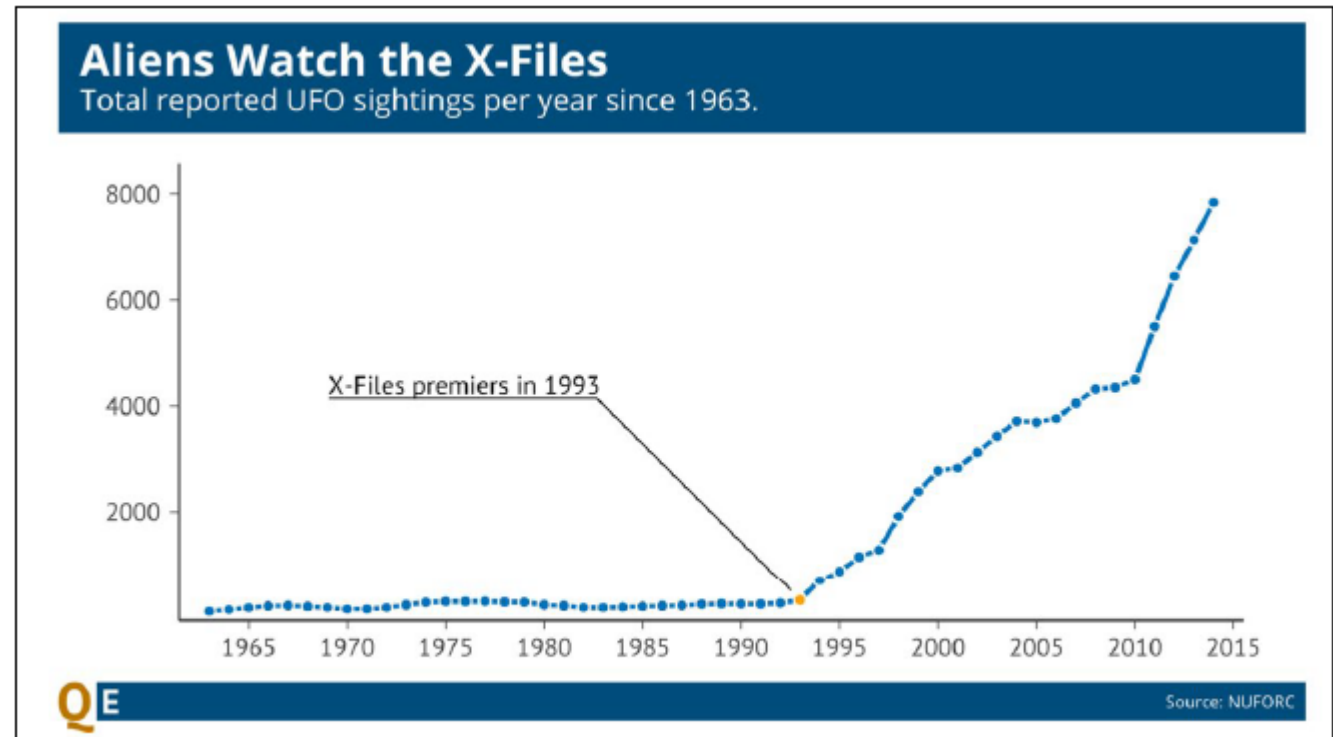
# Scatter plots

- A scatter plot is probably one of the simplest graphs to create.
- It is made by creating two **quantitative** axes and using data points to represent observations.
- The main goal of a scatter plot is to **highlight relationships between two variables** and, if possible, reveal a correlation.



# Line graphs

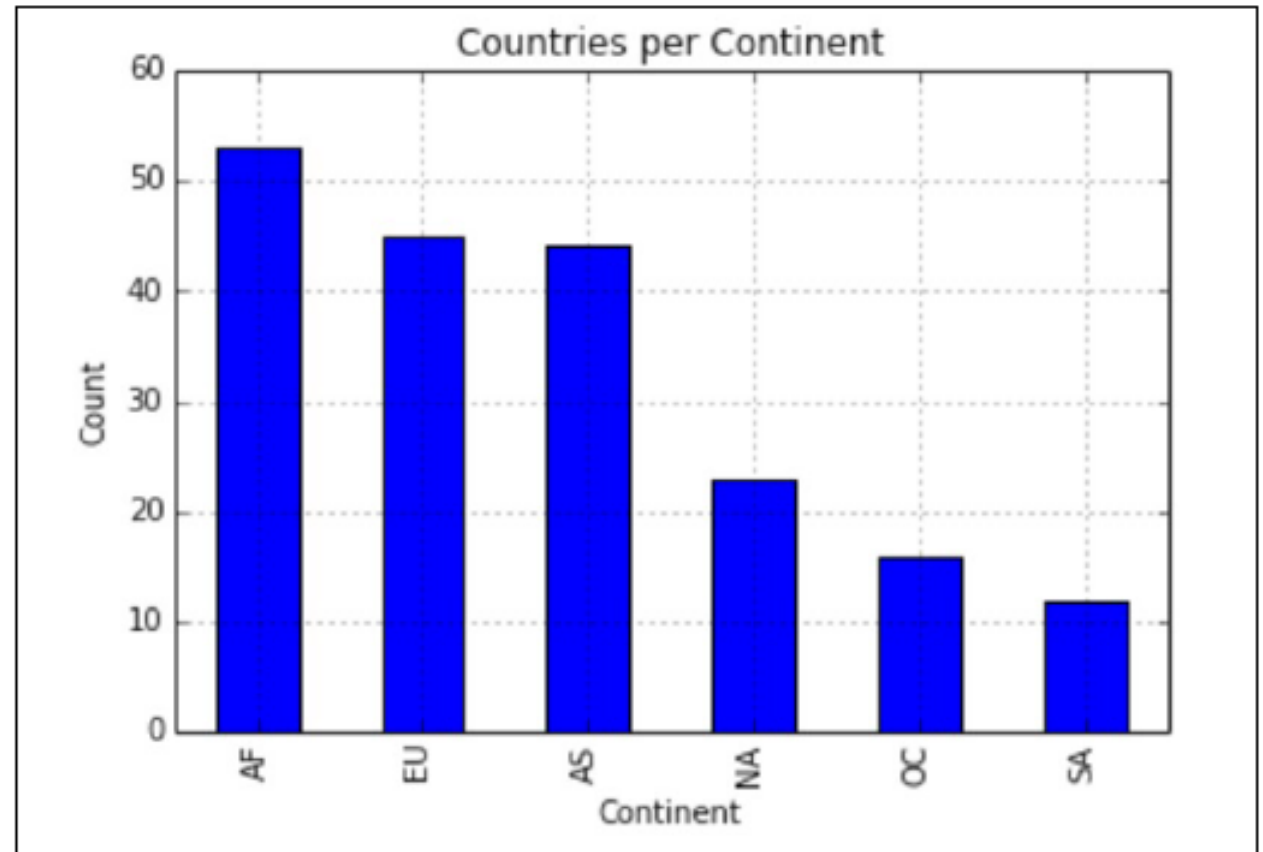
- Line graphs are, perhaps, one of the most widely used graphs in data communication.
- A line graph simply uses lines to **connect data points** and usually represents time on the x axis.
- Line graphs are a popular way to show **changes in variables over time**.
- The line graph, like the scatter plot, is used to plot **quantitative variables**.



Source: <http://www.questionable-economics.com/what-do-we-know-about-aliens/>

# Bar charts

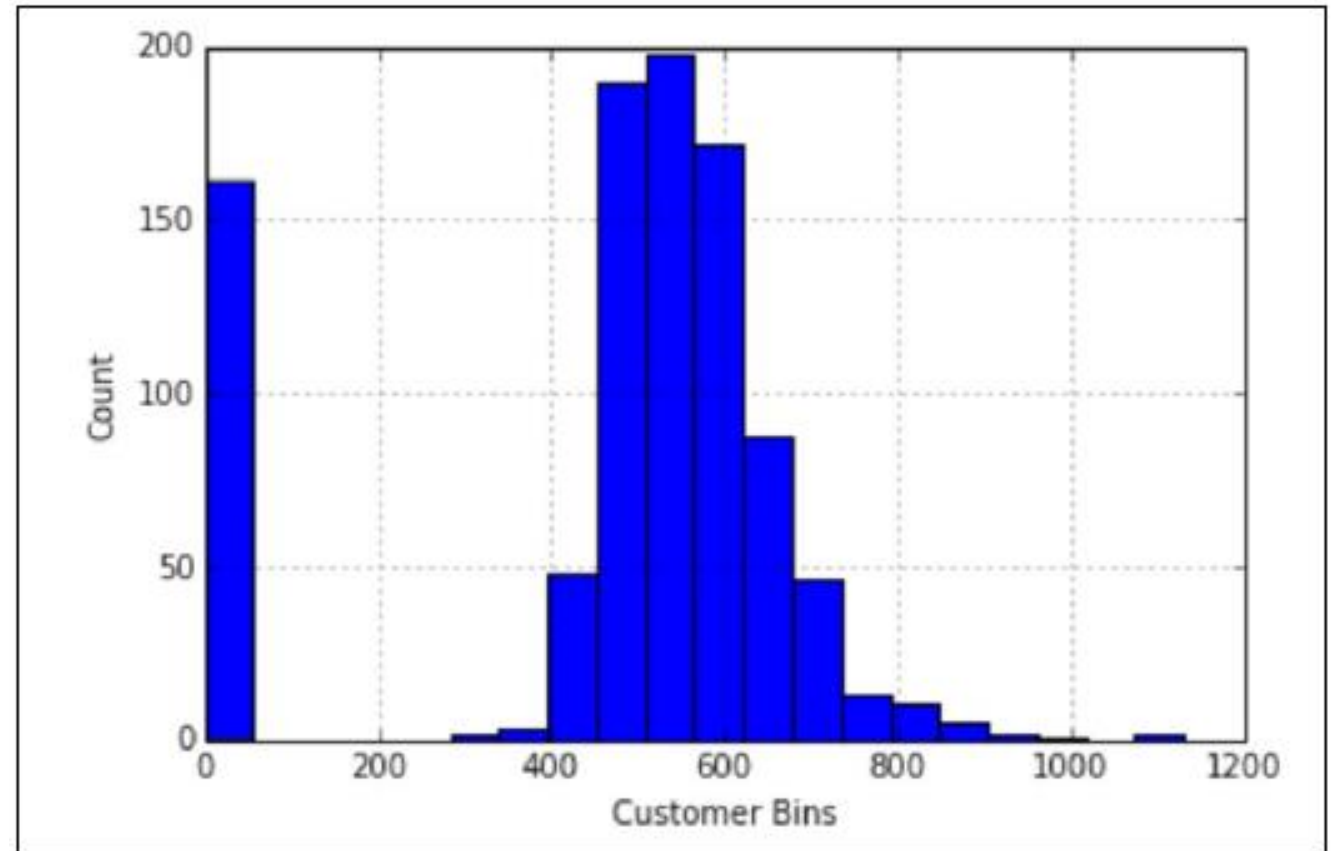
- We generally turn to bar charts when trying to **compare variables** across different groups.
- For example, we can plot the number of countries per continent using a bar chart.
- Note how the x axis does not represent a quantitative variable, in fact, when using a bar chart, the **x axis** is generally a **categorical variable**, while the **y axis** is **quantitative**.





# Histograms

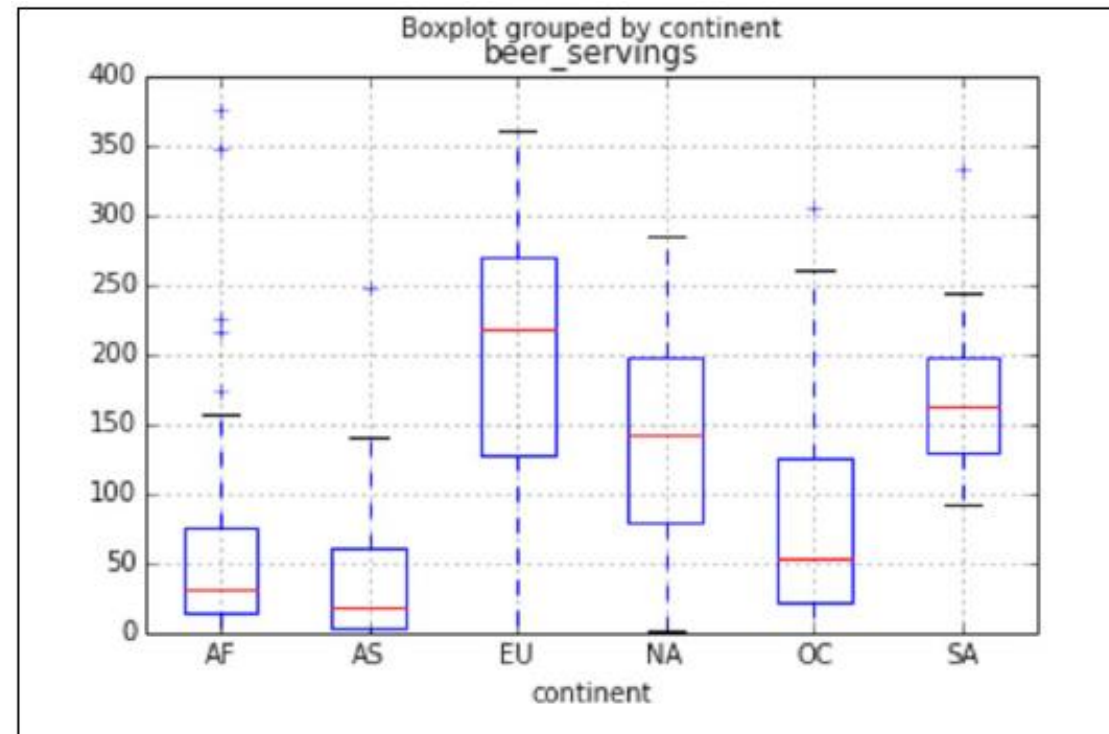
- Histograms show the **frequency distribution** of a single **quantitative** variable by splitting up the data, by range, into equidistant bins and plotting the raw count of observations in each bin.
- A histogram is effectively a bar chart where the **x axis** is a **bin (subrange) of values** and the **y axis** is a **count**.



# Box plots

Box plots are also used to show a **distribution of values**. They are created by plotting the five number summary, as follows:

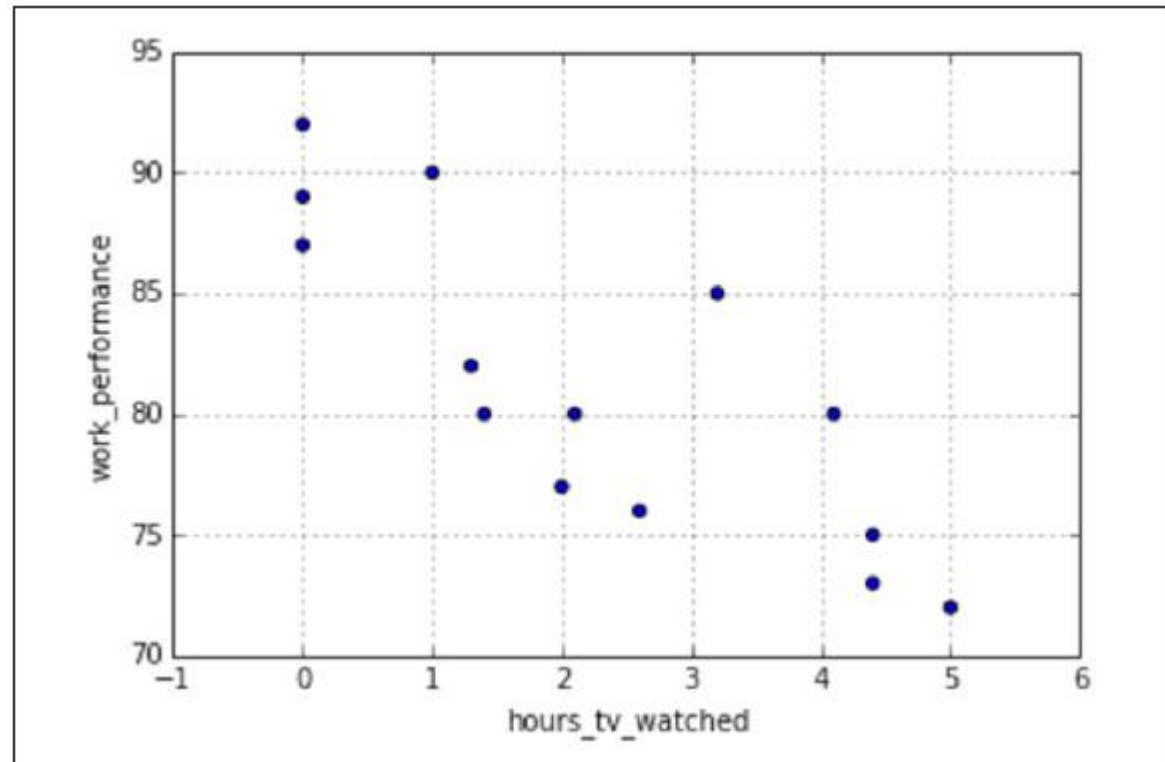
- The minimum value
- The first quartile (the number that separates the 25% lowest values from the rest)
- The median
- The third quartile (the number that separates the 25% highest values from the rest)
- The maximum value

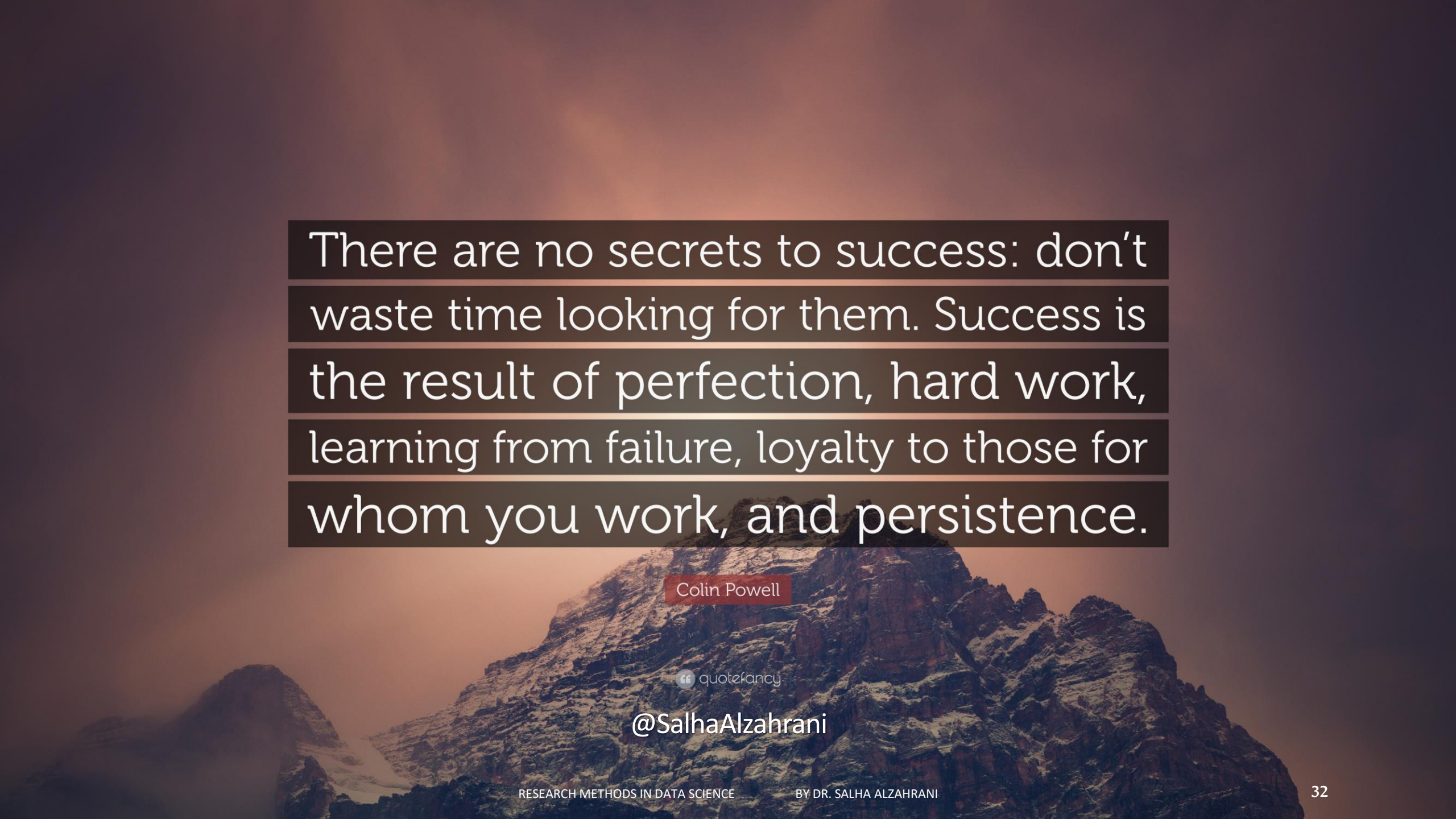


In Pandas, when we create box plots, the red line denotes the median, the top of the box (or the right if it is horizontal) is the third quartile, and the bottom (left) part of the box is the first quartile.

# Correlation versus causation

- **Correlation** is a quantitative metric between -1 and 1 that measures how two variables move with each other. If two variables have a correlation close to -1, it means that as one variable increases, the other decreases, and if two variables have a correlation close to +1, it means that those variables move together in the same direction—as one increases, so does the other, and vice versa.
- **Causation** is the idea that one variable affects another, or leads to another.





There are no secrets to success: don't waste time looking for them. Success is the result of perfection, hard work, learning from failure, loyalty to those for whom you work, and persistence.

Colin Powell

quote fancy

@SalhaAlzahrani