Probability and Statistics for Data Science

Data Science Course

Dr. Ariel Mantzura

2025-01-29

הטכניון
ביה"ס ללימודי המשך
של הטכניון ע"ש עזריאלי

Topics to be Covered in this Lecture

- Introduction
- Data generating process
- Probability vs Statistics
- Types of Variables
- Descriptive statistics
- Exercise
- Exploratory data distributions
- Outlier Detection Methods
- Exercise

Source: **Practical statistics for Data Scientist** Peter Bruce, Andrew Bruce & Peter Gedeck

Introduction

- Probability and statistics are important building blocks in statistical learning models.
- Generally speaking, the statistical learning methods that will be learned in this course are based on probability and statistics concepts.
- Moreover, performing data science tasks require a good foundation and understanding of these concepts.
- Probability and statistics are fields that deal with uncertainty.

Introduction

- In statistical learning uncertainty comes in many forms:
- For example, in prediction tasks we will ask what is the best prediction given some training data?
- Alternatively, we may ask what is the best model given some data?
- How do we estimate unknown parameters of our models?
- How much uncertainty are in our predictions?
- How do we evaluate uncertainty?
- Are there techniques that can mitigate or lower high uncertainty in our predictions?

Data generating process - Wikipedia definition

- A **Data Generating Process** is a process in the real world that generates the data one is interested in.
- Usually, analysts/scientists do not know the real data generating model.
- These models are often the distributions of the data in the population.
- Those distributions can often be presented via mathematical function.
- Some (simplified) examples of functions of data distributions are the Normal distribution, Bernoulli distribution and the Poisson distribution.

Data generating process - Simple example

- Consider a fair die (cube) with 6 numbers.
- In every throw each number has the same “chance” of falling.

- This defines the data generating process of independent throws of the cube.
- Alternatively,we can define the same data generation process as a uniform discrete distribution on the the values 1,2,3,4,5,6.
- Each value has a probability of $\frac{1}{6}$ in each independent throw.

Data generating process - Simple example

- In cases where we know the data generating process we can simulate data accordingly.
- For example, if we know that the data generating process is discrete uniform on the values 1,2,3,4,5,6 we can simulate a large number of throws from that process.

```
import random

sample = random.choices(range(1, 7), k=15)

print(sample)
```

```
## [2, 1, 2, 3, 1, 2, 4, 2, 6, 6, 5, 2, 4, 4, 6]
```

Data Generating Process - Examples

- If the data is generated from a continuous uniform distribution we can simulate independent observations using the following function.

```
import numpy as np
import pandas as pd
a = np.random.uniform(0, 1, 12)
b = a.reshape(-1, 3)
mean_a = np.mean(a)
print( b);print("Mean of a:", mean_a)
```

```
## [[0.07411078 0.77803964 0.32406805]
##  [0.27577341 0.53640225 0.74446561]
##  [0.50343335 0.69348872 0.46178284]
##  [0.72227268 0.50982532 0.49645035]]
## Mean of a: 0.5100094158016893
```

Data Generating Process - Examples

- If the data is generated from a standard Gaussian distribution we can simulate independent observations using the following function.

```
a = np.random.normal(0, 1, 9)
b = a.reshape(-1, 3)
mean_a = np.mean(a)
print(b);print("Mean of a:", mean_a)
```

```
## [[-1.34061025 -0.56858199  1.94094053]
##  [ 0.86670606 -0.48178864  0.48582513]
##  [-0.95655356  0.18017353  0.5936561 ]]
## Mean of a: 0.07997409955209672
```

- We will learn in more detail about different probability distributions or data generating processes later on.

Probability vs Statistics

- Generally, speaking, in probability, we start with a given data generating process or probability model.
- Given the model, we can compute probabilities of observing various outcomes.
- In statistics, we do the opposite: we start with observations and try to guess what model or data generating process produced them.
- Alternatively, in statistics after observing the data we ask questions regarding the data generating process.

Probability vs Statistics - Simple example

- **Probability**: We know that the die we are throwing is a fair one.
- We can ask the following probabilistic question: what is the probability that in 5 successive throws we will get the same number?

- **Statistics**: If we observe the following independent throws: 2 4 2 3 5 5 2 2 5 3 4 3 5 2.
- We may ask: Did these observations come from independent throws of a fair die.

Probability vs Statistics

- In data science we investigate data called the training set.
- In most cases, the data scientist does not fully know what is the data generating process.
- However, the data scientist may assume that the data was generated from some probability model where some information regarding the generating model is not known.
- For example: We may assume that the data came from a guassian distribution but the mean and variance are not known.
- Alternatively, the data scientist may make no assumptions regarding the "true" data generating process.
- In this case he will only drive conclusion directly from the data.

Types of variables

- There are two basic types of structured data: **numeric and categorical**.
- Numeric data comes in two forms: **continuous and discrete**.
- Categorical data takes only a fixed set of values.
- Binary data is an important special case of categorical data that takes on only one of two values, such as 0/1, yes/no, or true/false.
- Another useful type of categorical data is ordinal data in which the categories are ordered.
- An example of this is a numerical rating (1, 2, 3, 4, or 5).

Types of variables

- **Numeric** - Data that are expressed on a numeric scale.
  - **Continuous** - Data that can take on any value in an interval.
  - **Discrete** - Data that can take on only integer values, such as counts.
- **Categorical** - Data that can take on only a specific set of values representing a set of possible categories.
  - **Binary** - A special case of categorical data with just two categories of values, e.g., 0/1,
  - **Ordinal** - Categorical data that has an explicit ordering.

Types of variables

- **Rectangular Data** - The typical frame of reference for an analysis in data science is a rectangular data object, like a spreadsheet or database table.
- Rectangular data is the general term for a two-dimensional matrix with rows indicating records (cases) and columns indicating features (variables).
- **Data frame** is the specific format in Python.

Rectangular data

- **Data frame** - Rectangular data (like a spreadsheet) is the basic data structure for statistical and machine learning models.
- **Feature** - A column within a table is commonly referred to as a feature. Synonyms: attribute, input, predictor, variable
- **Outcome** - Many data science projects involve predicting an outcome. Synonyms: dependent variable, response, target, output
- **Records** - A row within a table is commonly referred to as a record. Synonyms: case, example, instance, observation, sample

Rectangular data

- n - number of observations
- p - number of variables
- x_{ij} - the value of the j^{th} variable for the i^{th} observation

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \dots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Descriptive statistics - Location

- **Average** - The sum of all values divided by the number of values.

$$\frac{\sum_{i=1}^{n} x_i}{n}$$

- **Weighted average** The sum of all values times a weight divided by the sum of the weights.

$$\frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} = \sum_{i=1}^{n} \alpha_i x_i$$

- where

$$\alpha_i = \frac{w_i}{\sum_{i=1}^{n} w_i}$$

Calculating weighted averages by matrix multiplication

- consider the following multiplication of matrix of weights W and vector X,

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \dots w_{1,p} \\ w_{2,1} & w_{2,2} & w_{2,3} \dots w_{2,p} \\ \vdots & & \\ w_{n,1} & w_{n,2} & w_{n,3} \dots w_{n,p} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{p} w_{1,i} x_i \\ \sum_{i=1}^{p} w_{2,i} x_i \\ \vdots \\ \sum_{i=1}^{p} w_{n,i} x_i \end{bmatrix}$$

Example

```
w = np.random.uniform(0, 10, 9).reshape(-1, 3)
x = np.random.normal(4, 3, 3).reshape(-1, 1)
colnames_w = ["w[,1]", "w[,2]", "w[,3]"]
colnames_x = ["X"]
wx = np.round(np.dot(w, x),1)
w_df = pd.DataFrame(w, columns=colnames_w)
x_df = pd.DataFrame(x, columns=colnames_x)
wx_df = pd.DataFrame(wx, columns=["wx"])
wx_combined = pd.concat([w_df, x_df,wx_df], axis=1)
print(wx_combined)
```

```
##       w[,1]     w[,2]     w[,3]         X    wx
## 0  0.939165  4.451687  3.392102  3.524327  23.3
## 1  6.677497  5.848997  5.138887  5.259835  49.1
## 2  7.611530  1.197005  3.551263 -1.019722  29.5
```

Descriptive statistics - Location

Median - The value such that one-half of the data lies above and below.

- The median is the middle number on a sorted list of the data.
- If there is an even number of data values, the middle value is one that is not actually in the data set, but rather the average of the two values that divide the sorted data into upper and lower halves.
- Compared to the mean, which uses all observations, the median depends only on the values in the center of the sorted data.

Descriptive statistics - Location

- **median for odd number of observations.**
- Let
$$X_1, X_2, \ldots, X_n$$
be n observations.
- Let
$$X_{(1)}, X_{(2)}, \ldots, X_{(n)}$$
be the ordered observations.
- The median is:
$$X_{\left(\frac{n+1}{2}\right)}$$
- **median for even number of observations.**
- The median is:
$$\frac{X_{\left(\frac{n}{2}\right)} + X_{\left(\frac{n}{2}+1\right)}}{2}$$

Descriptive statistics - Location

- **Percentile** - The value such that P percent of the data lies below.

.

- **Trimmed mean** - The average of all values after dropping a fixed number of extreme values.

$$\frac{\sum_{i=p+1}^{n-p} x_{(i)}}{n-2p}$$

- **Outlier** - A data value that is very different from most of the data.

Descriptive Statistics - weighted median

1. List the Data Points and Weights: You start with a set of data points, each associated with a weight.
2. Sort the Data Points: Arrange the data points in ascending order, keeping their weights associated.
3. Compute the cumulative sum of the weights.
4. Find the Median: The weighted median is the data point where the cumulative weight reaches at least half of the total weight.
5. If the total weight is W the weighted median is the smallest data point such that the cumulative weight is at least $\frac{W}{2}$.

Short Exercise

- Consider the following numbers:
  $x_1 = 45, x_2 = 62, x_3 = 23, x_4 = 47, x_5 = 56, x_6 = 73.$
- Calculate the average using function sum and arithmetic operators (+,-,*,/).
- $w_1 = 78, w_2 = 22, w_3 = 63, w_4 = 27, w_5 = 32, w_6 = 11.$
- Calculate the weighted average using function sum and arithmetic operators (+,-,*,/).
- Calculate the median using function sort.
- Calculate the weighted median using sort and np.cumsum functions.
- Write a function that calculates a trimmed mean.

Descriptive statistics - location

```
import seaborn as sns
import numpy as np
import pandas as pd
# Load the 'tips' dataset from seaborn
tips = sns.load_dataset('tips')
# Show the first 4 rows of the dataset
print(tips.head(4))
```

```
##    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
```

Descriptive statistics - location

```
tips['total_bill'].mean()
```

```
## 19.7859426229508
```

```
np.mean(tips['total_bill'].sort_values().iloc[
    int(0.1*len(tips)):int((1-0.1)*len(tips))])
```

```
## 18.663128205128206
```

```
tips['total_bill'].median()
```

```
## 17.795
```

```
np.average(tips['tip'], weights=tips['total_bill'])
```

```
## 3.417232138233593
```

Descriptive statistics - location

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

q25 = tips['total_bill'].quantile(0.25)
q50 = tips['total_bill'].quantile(0.50)
q75 = tips['total_bill'].quantile(0.75)
print(f"25th Percentile: {q25}")
```

```
## 25th Percentile: 13.3475
```

```
print(f"Median (50th Percentile): {q50}")
```

```
## Median (50th Percentile): 17.795
```

```
print(f"75th Percentile: {q75}")
```

```
## 75th Percentile: 24.127499999999998
```

Descriptive statistics - variability

- **Deviations** - The difference between the observed values and the estimate of location.

$$|x_1 - \bar{x}|, |x_2 - \bar{x}|, \ldots |x_n - \bar{x}|$$

$$|x_1 - median(x)|, |x_2 - median(x)|, \ldots, |x_n - median(x)|$$

- **Variance** - The sum of squared deviations from the mean divided by n – 1 where n is the number of data values.

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Descriptive statistics - variability

- **Standard deviation** - The square root of the variance.

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

- **Mean absolute deviation** - The mean of the absolute values of the deviations from the mean.

$$\frac{\sum_{i=1}^{n}|x_i - \bar{x}|}{n}$$

- **Median absolute deviation from the median(MAD)** - The median of the absolute values of the deviations from the median(m).

$$median(|x_1 - m|, |x_2 - m|, \ldots, |x_n - m|)$$

Descriptive statistics - variability

- **Range** - The difference between the largest and the smallest value in a data set.

$$max(x_1, x_2, \ldots, x_n) - min(x_1, x_2, \ldots, x_n)$$

- **Order statistics** - Metrics based on the data values sorted from smallest to biggest.
- **Interquartile range** - The difference between the 75th percentile and the 25th percentile.

$$percentile_{75} - percentile_{25}$$

Descriptive statistics - variability

```
variance = tips['total_bill'].var()
print(f"Variance: {variance}")
```

```
## Variance: 79.25293861397827
```

```
std_dev = tips['total_bill'].std()
print(f"Standard Deviation: {std_dev}")
```

```
## Standard Deviation: 8.902411954856856
```

```
# Interquartile range (IQR) of total_bill
iqr=(tips['total_bill'].quantile(0.75)
-tips['total_bill'].quantile(0.25))
print(f"Interquartile Range (IQR): {iqr}")
```

```
## Interquartile Range (IQR): 10.779999999999998
```

Descriptive statistics - variability

```
#tips['total_bill'].mad()  # mean absolute deviation
mean_value = tips['total_bill'].mean()
(tips['total_bill'] - mean_value).abs().mean()
```

```
## 6.8694400026874485
```

```
mad_median = np.median(np.abs(tips['total_bill']
-tips['total_bill'].median()))
print(f"Median Absolute Deviation (MAD): {mad_median}")
```

```
## Median Absolute Deviation (MAD): 5.03
```

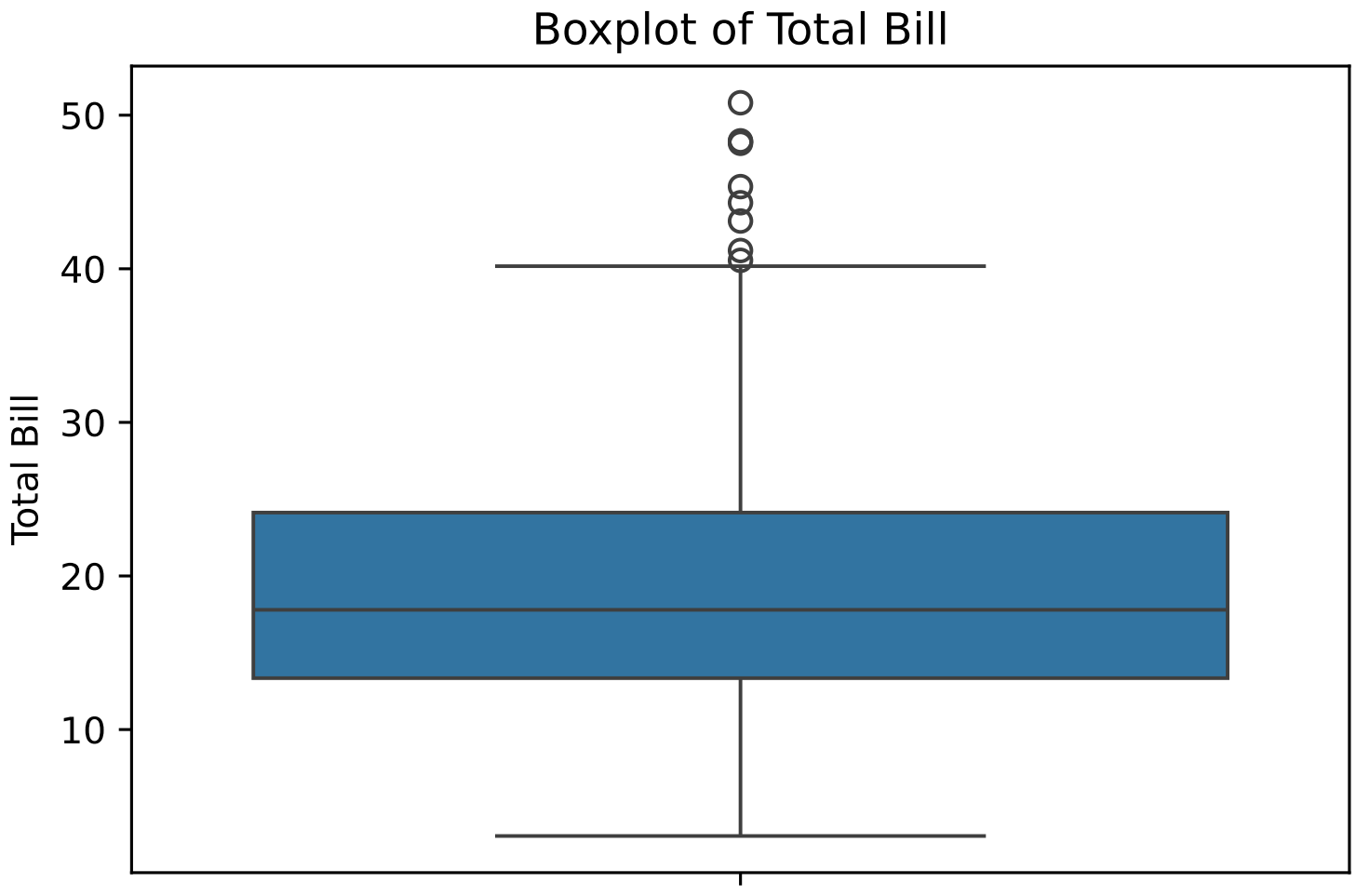
Exercise

- Create a Data frame with 100 observations from a Normal distribution with mean=3 and standard deviation=1 named normals.
- Add a new variable called lognorm that transforms the previous variable as follows: e^{normals}
- Plot the densities side by side (for lognormal use bandwidth=0.2).
- Calculate both variables: average, median, q_{25}, q_{75}, IQR, variance and Mad.
- Compare these values to the theoretical values. mean=$e^{(\mu+\frac{\sigma^2}{2})}$,median=$e^{\mu}$,variance=$(e^{\sigma^2}-1)(e^{2\mu+\sigma})$ where μ is the normals mean and σ^2 is the normals variance.
- Calculate a weighted mean of lognorm where the weights are $x_i - median(x)$ (of normals)

Descriptive statistics - exploring data distributions

- **Boxplot** - A plot that is used to visualize the distribution of data.
- **Frequency table** - A tally of the count of numeric data values that fall into a set of intervals (bins).
- **Histogram** - A plot of the frequency table with the bins on the x-axis and the count (or proportion) on the y-axis.
- **Density plot** - A smoothed version of the histogram, often based on a kernel density estimate.

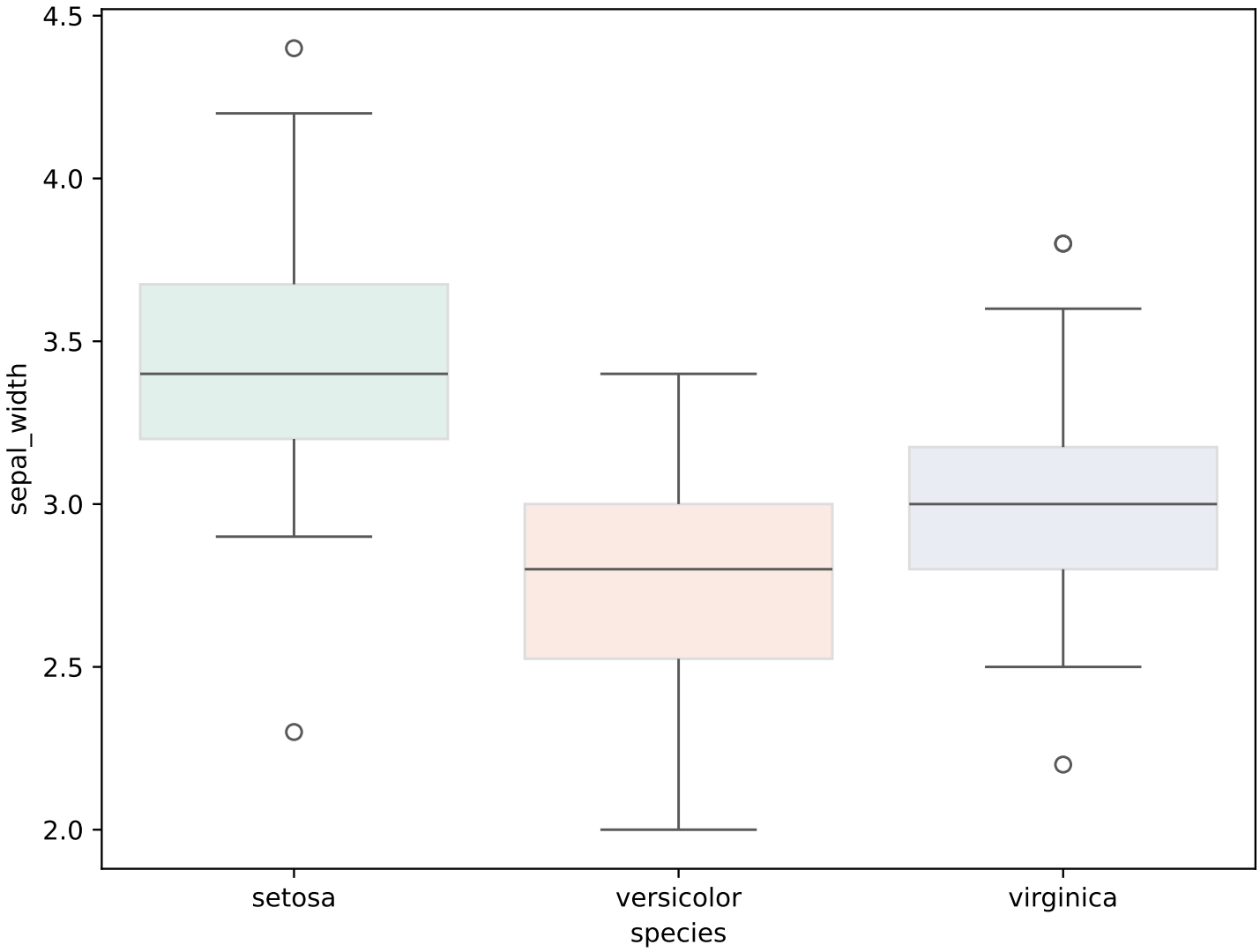
Descriptive statistics - boxplot



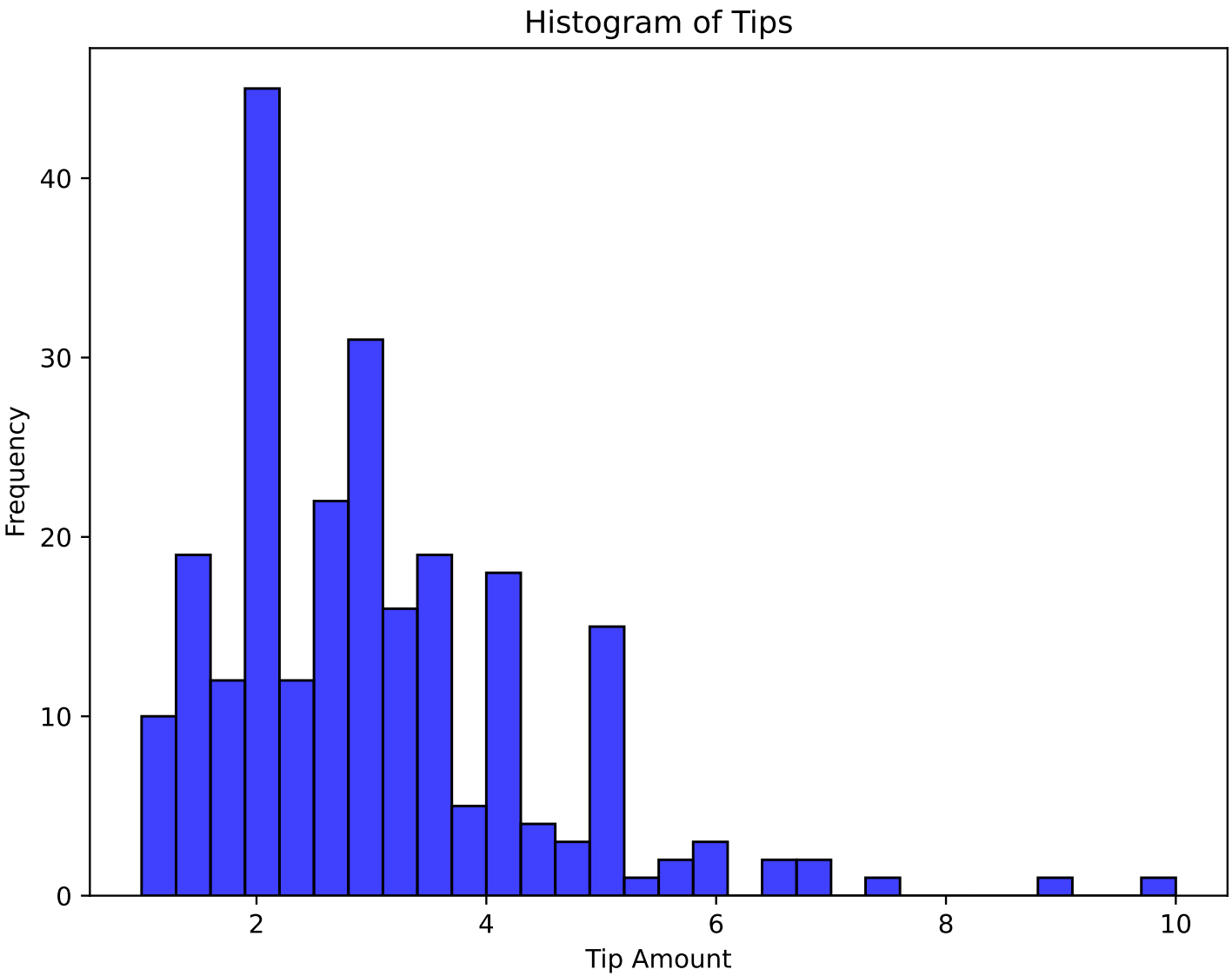
Descriptive statistics - boxplot explanation

- The line in the middle is the median.
- The upper limit of the box is the 75^{th} quantile.
- The lower limit of the box is the 25^{th} quantile.
- The upper whisker is the largest observation smaller than the q_3 plus 1.5 times the inter quantile range, i.e. $q_3 + 1.5 \times (q_3 - q_1)$.
- The lower whisker is the smalest observation larger than the q_1- 1.5 times the inter quantile range, i.e. $q_1 - 1.5 \times (q_3 - q_1)$.
- Observations beyond the whiskers are considered outliers.

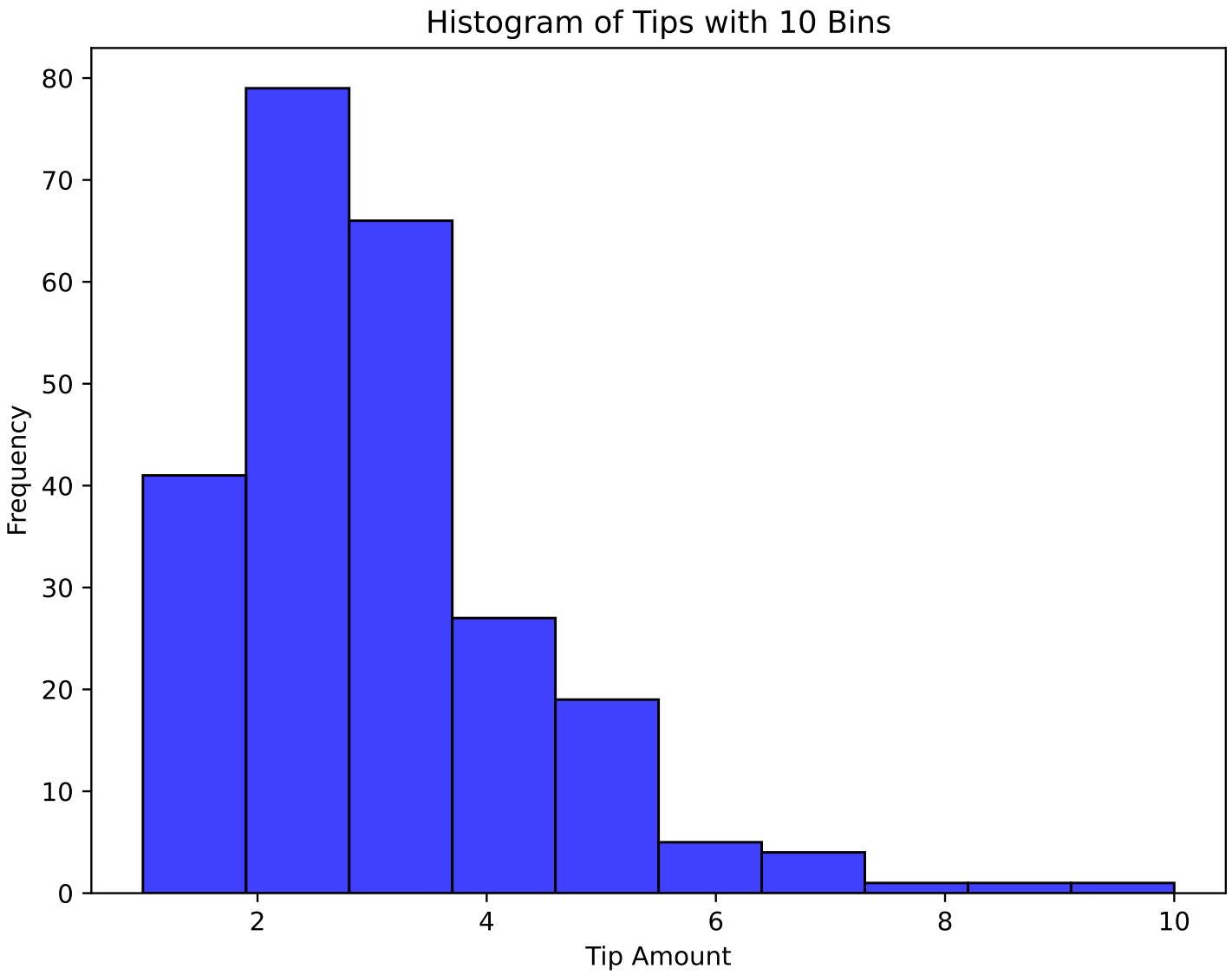
Descriptive statistics - multiple boxplots



Descriptive statistics - histogram

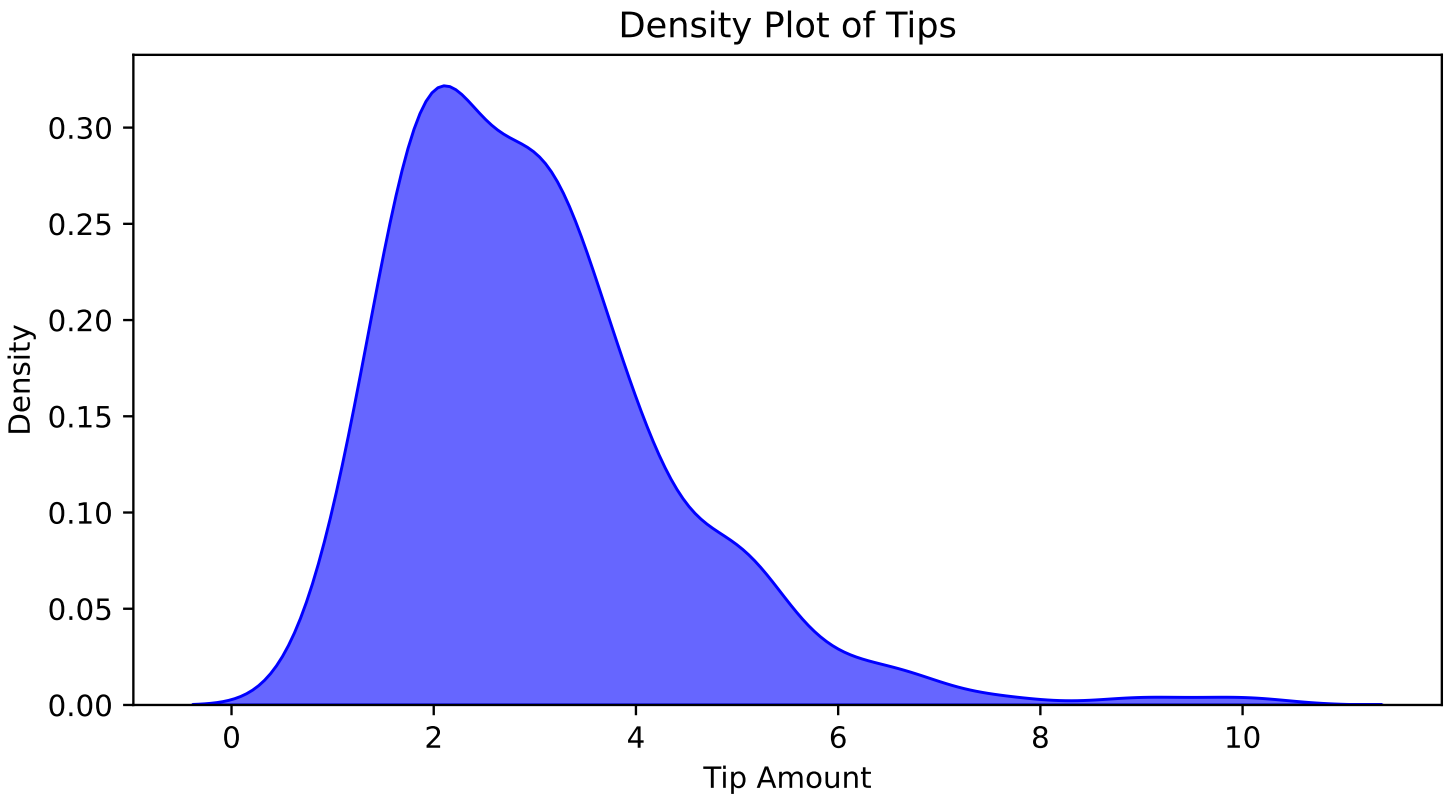


Descriptive statistics - histogram



Descriptive statistics - density

```
FALSE <module 'matplotlib.pyplot' from 'C:\\Users\\arima\\A
```



Descriptive statistics - (kernel) density function calculation

- Let $X_1, X_2, \ldots, X_n$ be n observations.
- The (kernel) density function $\hat{f}$ is calculated as follows:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} \mathcal{K}\left(\frac{|x_i - x|}{h} \right)$$

- Where $\mathcal{K}$ is called the kernel function.
- Typically, $\mathcal{K}$ is a weighting function which gives less weight to bigger values of $|x_i - x|$.
- As h the smoothing parameter is larger remote observation get larger weight in comparison to a smaller h.

Descriptive statistics - Normal (kernel) density function calculation

- One example for a kernel density function is the normal kernel:

$$\mathcal{K}(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$$

- So that

$$\frac{1}{nh} \sum_{i=1}^{n} \mathcal{K}\left(\frac{|x_i - x|}{h}\right) =$$

$$= \frac{1}{nh} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{\frac{-\left(\frac{|x_i - x|}{h}\right)^2}{2}}$$

Exercise - normal kernel function

- create a sequence of numbers seq1 starting from -5 to 5 with jumps of 0.02.
- define a function called norm_kernel as follows: def normal_kernel(x, v, h): return (1 / h) * np.mean(norm.pdf(np.abs((x - v) / h)))
- randomize 200 numbers from a standard normal distribution called std.
- Use the function to calculate the density with h=0.5,h=1,h=2,h=10.
- Plot the results.

Descriptive statistics - Skew

- Skewness is a measure of asymmetry that indicates whether the observations in a dataset are concentrated on one side.
- Right (positive) means that the outliers are to the right (long tail to the right).
- Left (negative) skewness means that the outliers are to the left.
- Formula to calculate skewness:

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Descriptive statistics - Skew

```
from scipy.stats import kurtosis, skew
# Calculate skewness of the 'tip' column
skewness_tips = skew(tips['tip'])
print(f"Skewness of 'tip' column: {skewness_tips}")
```

```
## Skewness of 'tip' column: 1.456426688422150 6
```

Descriptive statistics - Kurtosis

- Kurtosis is measure of whether or not a distribution is heavy tailed or light tailed relative to a Normal Distribution.
- Formula to calculate skewness:

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^2}$$

- The Kurtosis of a Normal distribution is 3 (in python 3 is suubtracted).

Descriptive statistics - Kurtosis

```
# Calculate kurtosis of the 'tip' column
kurtosis_tips = kurtosis(tips['tip'])
print(f"Kurtosis of 'tip' column: {kurtosis_tips}")
```

```
## Kurtosis of 'tip' column: 3.549551989345511 4
```

```
# Calculate kurtosis for a random normal distribution
random_normal = np.random.normal(0, 1, 10000)
kurtosis_random_normal = kurtosis(random_normal)
print(kurtosis_random_normal)
```

```
## -0.009727474042822148
```

Descriptive statistics - correlation

- **Correlation coefficient** - A metric that measures the extent to which numeric variables are associated with one another (ranges from –1 to +1).

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)\sigma_x\sigma_y}$$

- where

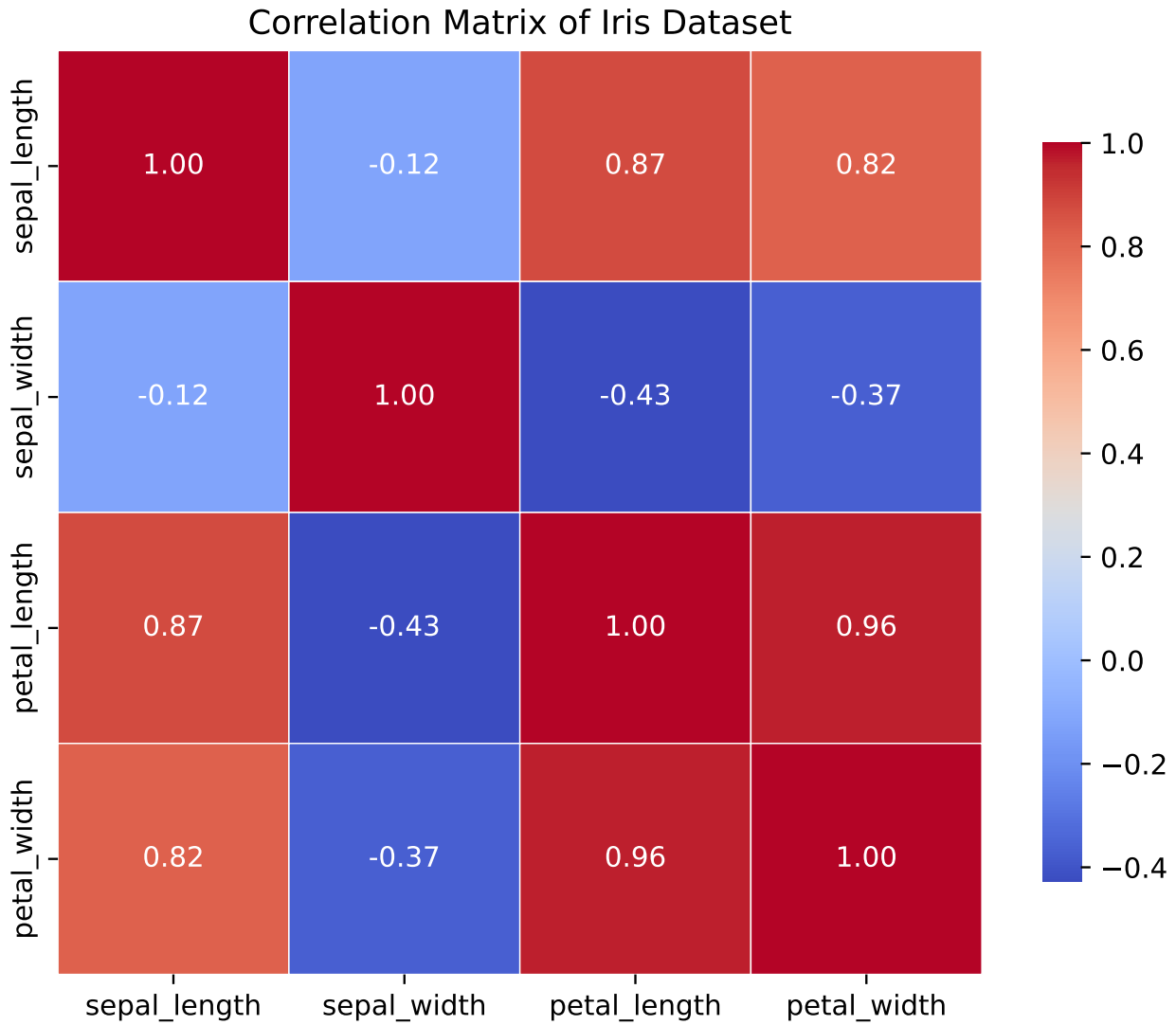
$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

  and σ_y is defined accordingly.

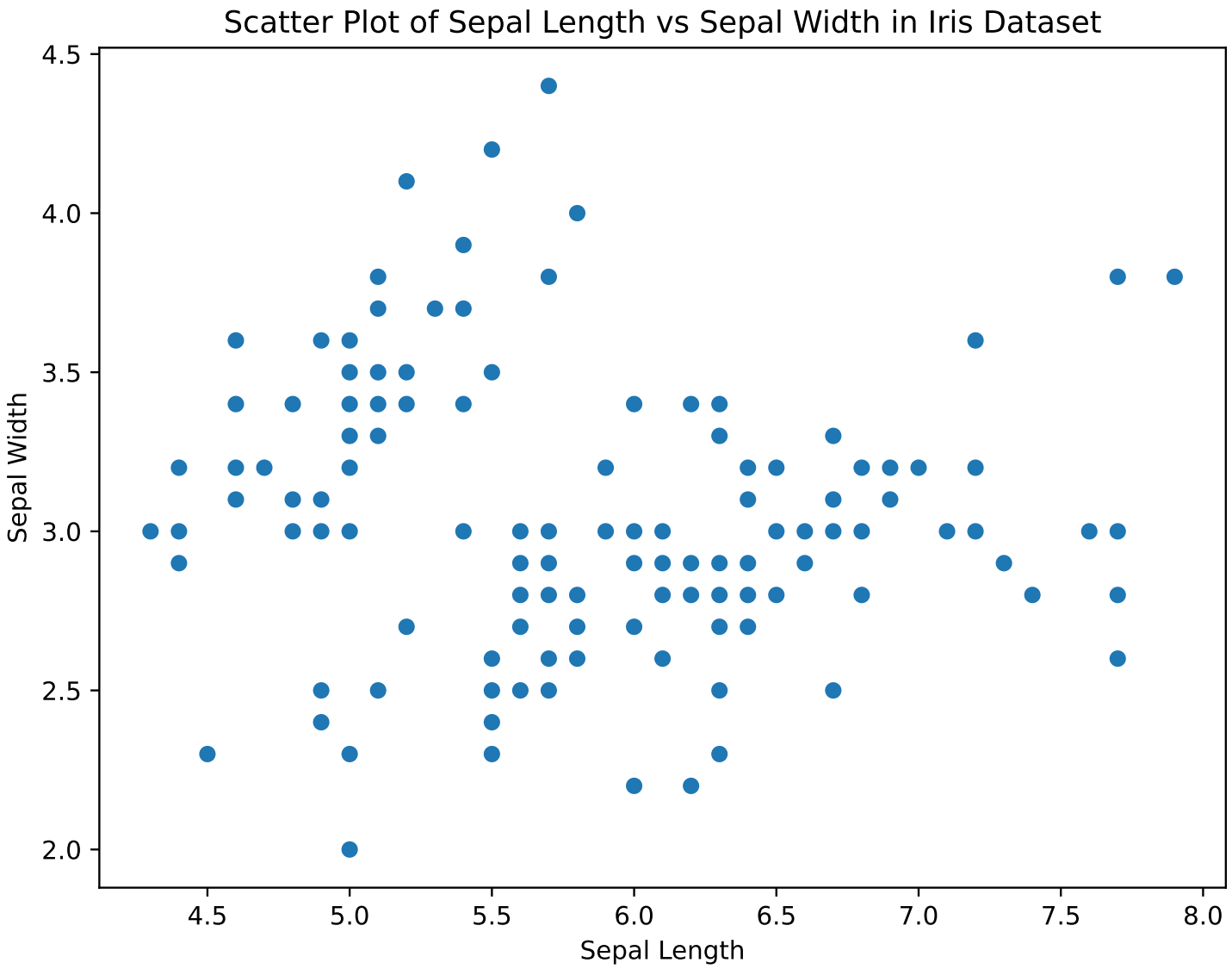
Descriptive statistics - correlation

- **Correlation matrix** - A table where the variables are shown on both rows and columns, and the cell values are the correlations between the variables.
- **Scatterplot** - A plot in which the x-axis is the value of one variable, and the y-axis the value of another

Descriptive statistics - correlation matrix



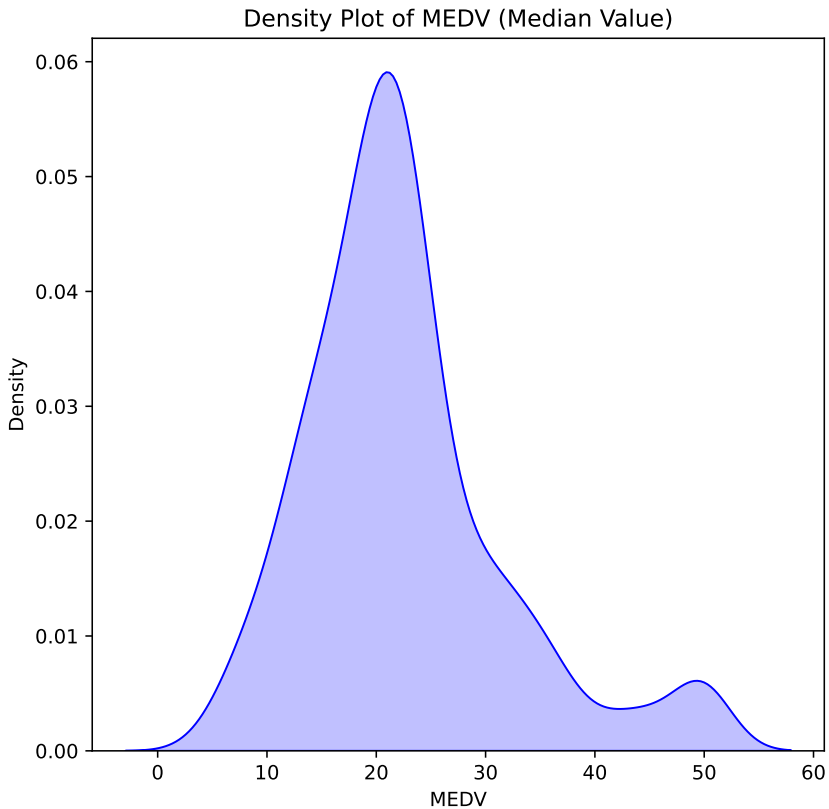
Descriptive statistics - scatterplot

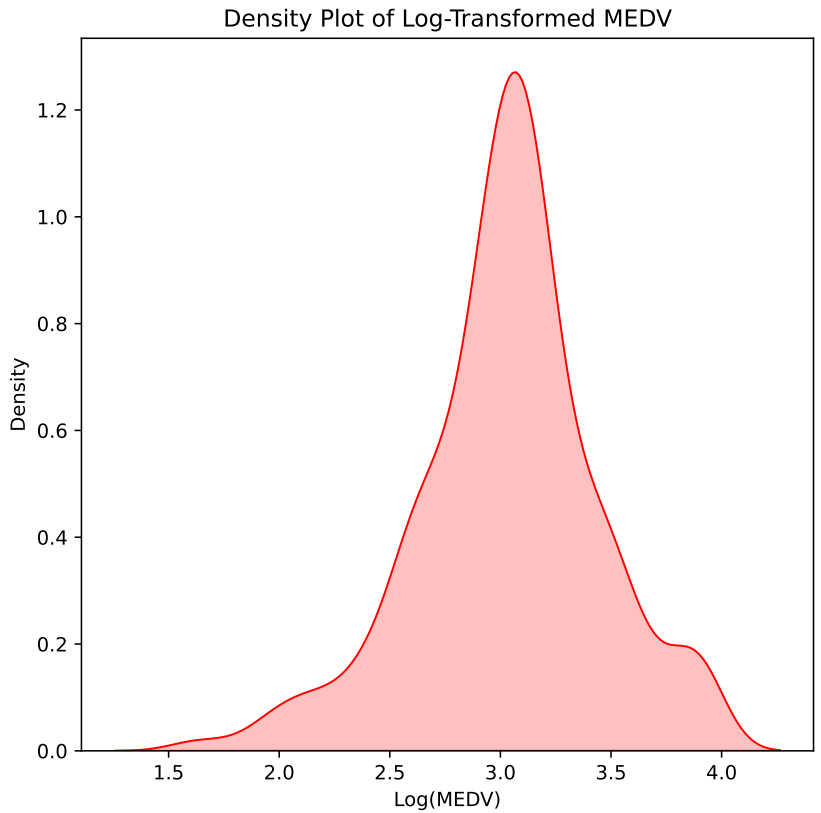


Transformations of the data the log transformation

- Many times we want our variable to be have a distribution that is more symmetric.
- For example, many methods for outlier detection are suited for relatively symmetric distributions.
- The log transformation tends to make non negative value distributions with a right skew more symmetric.
- We will exemplify this notion on the Population variable in the states data set.

Transformations of the data - the log transformation

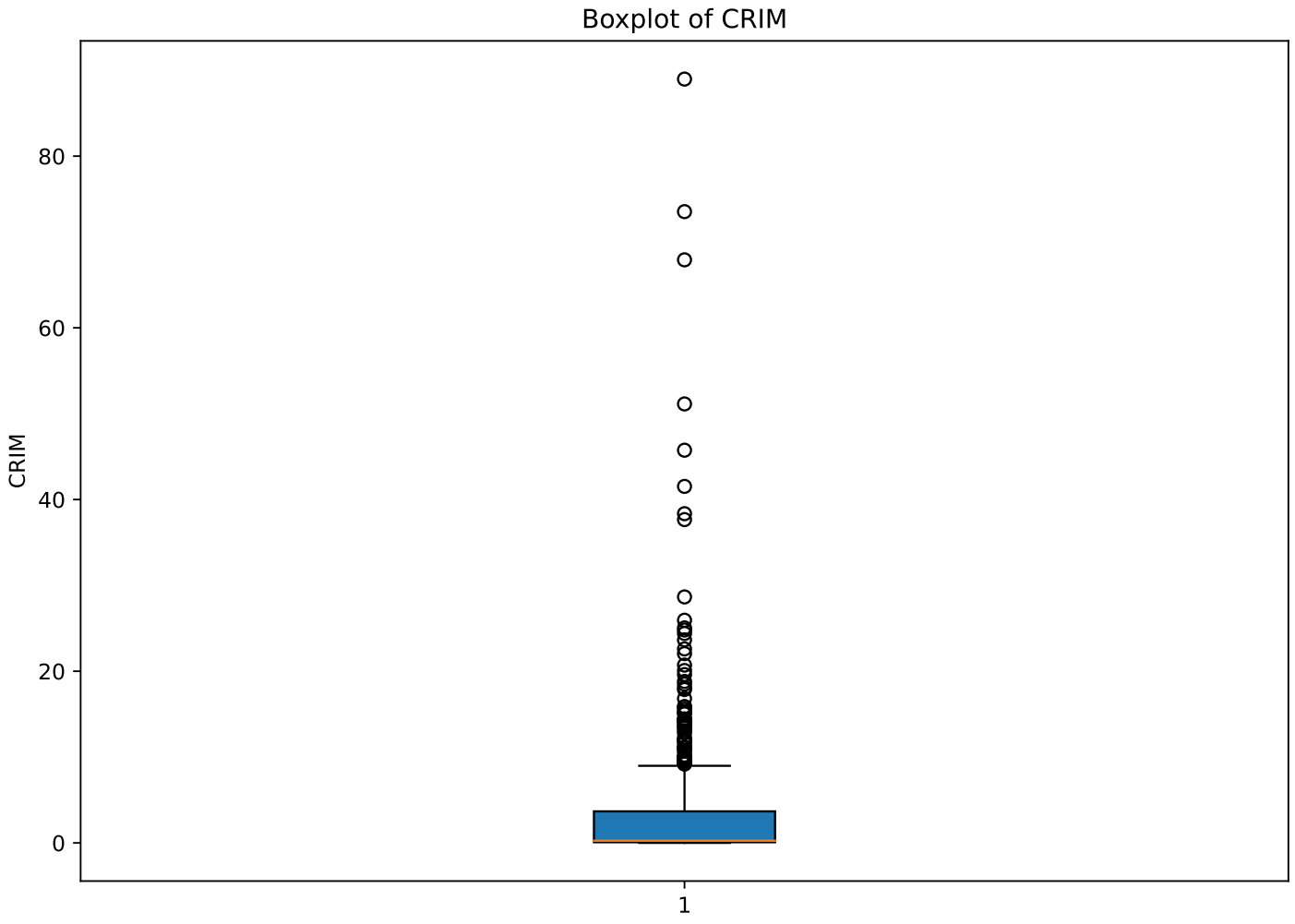




Outlier detection methods - extreme quantiles

- We can simply take the extreme quantiles, e.g. the 2^{nd} and 98^{th} quantile.
- Any observation smaller or larger will be defined as extreme or outliers.
- The problem with this method is that we will always have outliers and we know approximately the number of them.

Outlier detection methods - using boxplots



Outlier detection methods - average plus/minus 3 standard deviations

- A common method for outlier detection in symmetric distributions is

$$\bar{x} - 3 \times sd(x), \bar{x} + 3 \times sd(x)$$

- This method does not necessarily give outliers.

Outlier detection methods - Robust methods

- A robust method for outlier detection in symmetric distributions is

$$\text{median}(x) - 3 \times \text{mad}(x), \text{median}(x) + 3 \times \text{mad}(x)$$

- This method does not necessarily give outliers.
- Why is this method considered robust?

Exercise

- Calculate the skewness and kurtosis of the normals and log norm.
- Upload the data set Credit.
- Take a glimpse of the Credit data set.
- Calculate the skewness and Kurtosis of the Income variable and of log(Income).
- Detect outliers in the Income and log(Income) variable according to the learned methods by their suitability.
- Examine the boxplots of income by variable student.