Prof. Dr. Steffen Wagner
Angewandte Statistik
Fachbereich II
Berliner Hochschule für Technik

Exercise 04: Measures of Dispersion and Matrices in R

Statistical Computing – WiSe 2022/23

Preparations

a. Make sure you work inside your RStudio project. Details please see Exercise 01.

b. If not done until now, install the R package `carDate` by running the following code

```
install.packages("carData")
```

Another possibility to install packages is via the *Packages* tab in the lower right pane of RStudio.

c. Open a new R script file (either `Shift + Ctrl + N` or via the menu), type the following comment/code in the first line and save the file.

```
#============================================================================#
# Statistical Computing: exercise sheet 4                                    #
# Measures of Dispersion and Matrices in R                                   #
#============================================================================#
rm(list = ls(all.names = TRUE))
```

d. Explain what the command `rm(list = ls(all.names = TRUE))` does.

e. Work through the following examples as you do every week. In many of the code examples the output is not included, you should always read and understand the output. Please add comments (in your own words) to your code so that it will make sense to you when you return several weeks (or years) later.

f. When there is a question in the text e.g. *"What is the mean and standard deviation of the prestige Scores?"*, copy the question and the answer as a a comment to your R code. (Later on, we will use RMarkdown for this purpose.)

Standard deviation and variance in R

Last week you worked through an exercise using the Prestige data. You will be using the same data this week.

```
library(carData)
data("Prestige")
```

Warm up

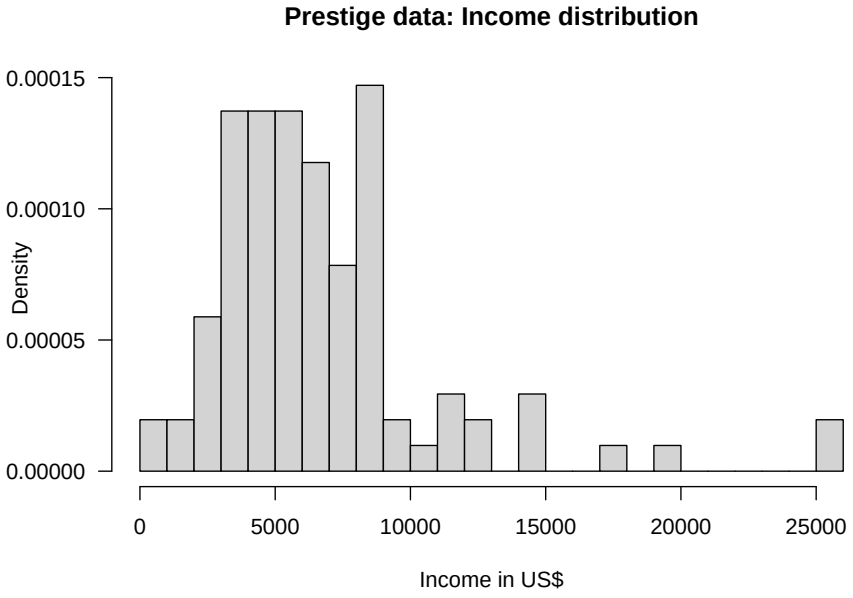
a. Plot the variable income in a histogram, to get a feel for the data.

```
hist(Prestige$income, breaks = 15)
```

and make follwing adjustements to the plot:

- Plot probability densities on the y-axis instead of frequencies.
- Adjust the breaks of the histogramm so that the class width is 1000 and the lowest class starts with a value of 0. The highest class should be $(25000, 26000]$
- Label the x-axis including correct units and add a title to the plot.
- Your histogram should now look like this:

```
hist(Prestige$income,
breaks = seq(0, 26000, by = 1000),
xlab = "Income in US$",
main = "Prestige data: Income distribution",
las = 1,
probability = TRUE)
```




```
# Calculation of the y-value for the mode:
# - as can be seen from the plot, the mode of the distribution is for
#   class (8000, 9000].
# - function `hist` uses left open intervals (as the help page for `hist`
```

```
#   tells you.)
# - the densitiy is given as d = (rel. Frequency)/(width of class)
#   This can be derived from the 2nd statement area = rel. Frequency and the
#   geometric consideration that area = width * height = width * density.
dMode <- mean(Prestige$income > 8000 & Prestige$income <= 9000)/1000
```

b. Interpret the values on the y-axis and calculate the value of 0.0001471 for the mode of the distribution by hand. Proceed as follows:

   - First derive the formula to calculate the density of the mode by considering the following general statements about a histogramm:
     - The total area of a histogram used for *probability density* is always normalized to 1.
     - The plotted area in a histogram for an individual class equals the relative frequency of observations in this class.
     - If the length of the intervals on the x-axis are all 1, then a histogram is identical to a relative frequency plot.
   - Then calculate the density for the mode.

c. In Worksheet 3 you came accross the R functions `var()` for the variance and `sd()` for the standard deviation. To understand these dispersion statistics better, compute these statistics from their definitions given in the lecture. Check your values using the standard R functions.

```
xbar <- mean(Prestige$income)
devs <- Prestige$income-xbar
sqdevs <- devs^2
ssqdevs <- sum(sqdevs)
varincome <- ssqdevs/(length(Prestige$income) - 1)
sdincome <- sqrt(varincome)

# check:
var(Prestige$income) == varincome
all.equal(var(Prestige$income), varincome)
all.equal(sd(Prestige$income), sdincome)
```

d. Obtain the range `range()` and interquartile range `IQR()` of the income data. Notice that the IQR is a roughly similar value to the standard deviation. This is often the case.

Linear transformations

You will transform the data so that income is measured on a scale from 0 (smallest income) to 1 the largest income. The formula to do this is

$$y_i = \frac{x_i}{\max x - \min x} - \frac{\min x}{\max x - \min x}.$$

This is a linear transformation of the form $y_i = ax_i + b$.

a. Use R to find the values of a and b in this case?

b. Calculate the mean and standard deviation of the transformed income using the formulae given in the lecture based on the result from part (2.c) above.

```r
a <- 1/(max(Prestige$income) - min(Prestige$income))
b <- -min(Prestige$income) * a
Prestige$income01 <- a * Prestige$income + b

# More elegeantly and covered in detail later in this course, is the usage of
# an individually created function:
linTrans01 <- function(x){
  a <- 1/(max(x) - min(x))
  b <- -min(x) * a
  a * x + b
}
# This function can now be applied directly to the original data
# - disadvantage: we don´t get the values of `a` and `b` because they only exist
#   in the environment (i.e. inside) the function
Prestige$income01 <- linTrans01(Prestige$income)

# Check of transformed values, e.g.
summary(Prestige$income01)
```

c. Store the transformed data as new variable `income01` in the `Prestige` data.frame.

d. Check that all the values of `Prestige$income01` lie between 0 and 1.

e. Calculate its mean and standard deviation and compare it to the expected values.

```r
# Check of mean
mean(Prestige$income01)
a * mean(Prestige$income) + b

# Check of standard deviation
sd(Prestige$income01)
sd(Prestige$income) * a
```

Coefficient of variation c_v

As standardized measure of dispersion we use

$$c_v = \frac{s_x}{\overline{x}}.$$

a. Calculate the CV for the variable `income` and `income01`

b. Discuss the results.

```r
# we create a function for more comfort:
cv <- function(x) sd(x)/mean(x)

# Since the applied transformtion is linear and not proportional we obtain
# different CV for the two variables
cv(Prestige$income)
cv(Prestige$income01)
```

Interpretation of the standard deviation

a. For the variable called `income` use R to calculate the interval: $[\overline{x} - 2s_x;\ \ \overline{x} + 2s_x]$.

b. How many data points actually lie in this interval?

c. What proportion of points lie in this interval?

d. Does this proportion fit with the rule of thumb?

e. Repeat part the above steps for the variable `education`.

```
# For income we obtain
lowBound <- mean(Prestige$income) - 2 * sd(Prestige$income)
uppBound <- mean(Prestige$income) + 2 * sd(Prestige$income)

# number of data points in this interval:
sum(lowBound <= Prestige$income & Prestige$income <= uppBound)
```

```
## [1] 98
```

```
# proportion of data points in this interval:
# from the rule of thumb we would expect 95%:
mean(lowBound <= Prestige$income & Prestige$income <= uppBound)
```

```
## [1] 0.9607843
```

Empirical cumulative distribution function

To get the R's default plot of the empirical cumulative distribution function (ecdf) use:

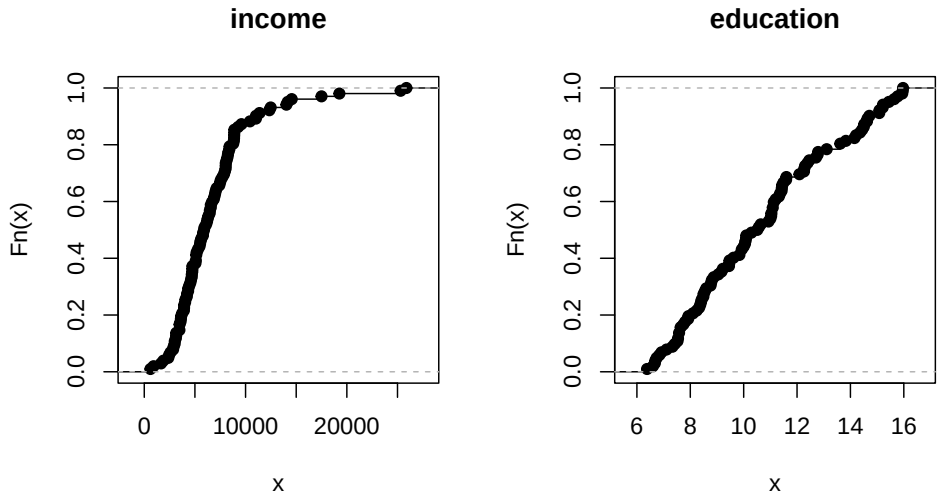
```
plot(ecdf(Prestige$income))
```

Note that the function `ecdf()` does the calculation and the `plot()` function displays it.

a. You can change the plot with the following arguments
   - To get rid of the big black dots use the argument `do.points = FALSE`.
   - To draw the jumps as steps add the argument `verticals = TRUE`.

b. Notice that the ECDF for `income` rises steeply between 2000 and 10000 dollars, and is then shallow for values above that. This tells us that the variable is skewed (not symmetric). Compare this with the shape of the ECDF for `education`, which is roughly symmetric.

```
parOld <- par(mfrow= c(1, 2))
plot(ecdf(Prestige$income), main = "income")
plot(ecdf(Prestige$education), main = "education")
par(parOld)
```


income
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
x
education
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
6
8
10
12
14
16
x

Matrices in R

Specifiying matrix objects

A vector is a one dimensional *collection* of atomic elements (eg numbers)

A matrix is a two dimensional *collection* of elements. An example of a matrix in mathematical notation is:

$$M = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

M is a matrix with 2 rows and 3 columns. $M_{1,3}$ is the element of M in the 1st row and 3rd column and equals 5.

To define a matrix in R use the `matrix()` function:

```
M <- matrix(1:6, nrow=2)
M
```

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

Note that the vector `1:6`$= (1, 2, 3, 4, 5, 6)$ is assigned to the matrix by column. `matrix()` takes an option called `byrow = TRUE` if you want to assign the values by row.

```
M <- matrix(1:6, nrow=2, byrow = TRUE)
M
```

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

You specify the number of rows and/or the number of columns using `nrow` or `ncol`, you probably want to supply at least one of them. `dim()` returns the number of rows and columns of a matrix:

```
M <- matrix(1:6,nrow=2)
M
dim(M)
M <- matrix(1:6,ncol=3)
M
dim(M)
M <- matrix(1:6)
M
dim(M)
```

A subset of elements in a vector can be accessed using square brackets. An example from Week 1 was:

```
fibonacci <- c(1, 1, 2, 3, 5, 8, 13, 21, 34, 55)
```

The elements of a matrix are identified by the row and the column, so the square bracket notation for a matrix requires two indices separated by a comma `[.,.]`. The row index always comes before the column index.

```
M <- matrix(1:6, nrow=2)
M[1, 3]
```

```
## [1] 5
```

You can specify multiple rows and/or columns:

```
M[1:2, 1]       # 1st and 2nd rows, 1st column
M[2, c(1 ,3)]    # 2nd row, 1st and 3rd columns
M[1:2, c(1, 3)] # 1st and 2nd rows, 1st and 3rd columns
```

The subsetting of vectors, matrices, arrays and data frames is a very useful tool and we will be using it a lot.

- To get one complete row of a matrix omit the term after the comma:

  ```
  M[2, ]
  ```

- To get one complete column of a matrix omit the term before the comma:

  ```
  M[ , 3]
  ```

- Note that in the last example R returns the column as a vector. This is usually what one wants so is the default behaviour. - To drop one row of a matrix use a minus sign in the row field:

  ```
  M[-2, ]  # all rows except the 2nd, all columns
  ```

- To drop one column of a matrix use a minus sign in the column field:

  ```
  M[, -1]  # all rows, all columns except the 1st
  ```

- You can even specify the rows or columns using an R-object

  ```
  x <- c(1, 3)
  M[, x]  #  all rows, 1st and 3rd columns.
  ```

Matrix arithmetic

It is likely that you have learnt matrices, matrix addition and matrix multiplication at some point in your undergraduate studies. You will certainly learn more about matrices in the *linear algebra* part of your *Mathematical Models* course. Very brief details about matrix arithmetic are given below in parallel with the R commands. This is intended as a short "recap". There is also a written exercise in Section 4. If this subject is new to you, you should consult a text book or do an online search into this subject and try more exercises.

Matrices have their special arithmetic rules. Addition of two matrices is as one would expect:

$$\text{If } A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 5 & 7 & 9 \\ 6 & 8 & 10 \end{bmatrix}, \quad \text{then} \quad A + B = \begin{bmatrix} 6 & 10 & 14 \\ 8 & 12 & 16 \end{bmatrix}.$$

Which is what + in R does:

```
A <- matrix(1:6, nrow = 2)
dim(A)
```

```
## [1] 2 3
```

```
B <- matrix(5:10, nrow = 2)
dim(B)
```

```
## [1] 2 3
```

```
A + B
```

```
##      [,1] [,2] [,3]
## [1,]    6   10   14
## [2,]    8   12   16
```

Multiplication of a scalar[1].} and a matrix is also straightforward:

$$2A = \begin{bmatrix} 2 & 6 & 10 \\ 4 & 8 & 12 \end{bmatrix}.$$

```
2 * A
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    4    8   12
```

However, **the multiplication of two matrices follows special rules**:

$$\text{If } A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \text{ and } C = \begin{bmatrix} 5 & 8 \\ 6 & 9 \\ 7 & 10 \end{bmatrix}, \text{ then } AC = \begin{bmatrix} 58 & 85 \\ 76 & 112 \end{bmatrix}.$$

In R `A * B` performs elementwise multiplication and `A %*% B` performs the matrix multiplication used in mathematics.

```
A <- matrix(1:6, nrow = 2)
B <- matrix(5:10, nrow = 2)
C <- matrix(5:10, nrow = 3)

A * B
```

```
##      [,1] [,2] [,3]
## [1,]    5   21   45
## [2,]   12   32   60
```

```
A %*% C
```

```
##      [,1] [,2]
## [1,]   58   85
## [2,]   76  112
```

`t()` transposes a matrix and `solve()` computes the inverse.

[1] Scalar is a mathematical term for a number with no dimension, the *every day numbers

```
t(A)
```

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

```
D <- matrix(1:4, nrow = 2)
solve(D)
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

```
solve(D) %*% D
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

There are many more functions which apply to matrices, you will learn them when they are needed.

Comparing data frames and matrices

Earlier in the course, the concept of a data frame was introduced. The `Prestige` data is a data frame, it contains data elements in the rows and variables in the columns. It is easy to think that a data frame is a type of matrix because it stores data in a rectangular format. There are similarities, but they are not the same:

- All entries in a *matrix* must be numeric, otherwise matrix addition and multiplication won't work.
- A *data frame* consists of a number of variables and each variable is represented as a column in the data frame. The length of each variable is equal.
- A *data frame* can have a mix of data types: numeric, factor, logical or character. All the elements of one variable must have the same *data type*.

The main similarity between a matrix and a data frame is that they are both represented as rows and columns and the `[., .]` notation can be used for both.

Example:

```
data(iris)
iris[10, ]            # returns the 10th row of the iris data frame
iris[1:10, 4]         # first 10 rows of the 4th variable in iris
iris[1:10, 'Species'] # first 10 rows of the variable 'Species'
```

The following three commands all give the same output

```
data(iris)
iris[1:10, 4]
iris[1:10, 'Species']
iris$Species[1:10]
```

The *environment* window

In the *Environment Window* top left you can see a list of all objects you have used today, one of which is called `iris`. On the right hand side of this row there is an icon symbolising a data matrix. Click on this icon. Notice that there are 4 continuous variables and one nominal variable. The R function `str()` outputs a summary of the object, which for a data frame includes the variable type.

```
str(iris)
```

Tidying up your R session

Tidy up your script file removing any mistyped commands that caused errors. Add comments to make the code readable and save your files.

Written Exercises

Matrix arithmetic

Here are three easy matrix exercises, to check you know how matrix arithmetic works. %If you are unsure about matrix addition or matrix multiplication then ask the lecturer.

$$\text{Let } A = \begin{bmatrix} 3 & 5 \\ 2 & 8 \end{bmatrix} \qquad B = \begin{bmatrix} -2 & 0 \\ 5 & -1 \end{bmatrix} \qquad \text{and} \qquad C = \begin{bmatrix} 5 & 0 \\ 4 & 5 \end{bmatrix}$$

Calculate $A + B$, $3B$ and AC.

Solution:

$$A + B = \begin{bmatrix} 1 & 5 \\ 7 & 7 \end{bmatrix} \qquad 3B = \begin{bmatrix} -6 & 0 \\ 15 & -3 \end{bmatrix} \qquad AC = \begin{bmatrix} 3 & 5 \\ 2 & 8 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 35 & 25 \\ 42 & 40 \end{bmatrix}.$$

ECDF

The following (artificial) data set has 12 values. Calculate the empirical cumulative distribution function $f_{12}(x)$ at each of the observed values and draw the function.

data:

$x_1 = 181$, $x_2 = 174$, $x_3 = 186$, $x_4 = 172$, $x_5 = 172$, $x_6 = 186$, $x_7 = 178$, $x_8 = 182$, $x_9 = 180$, $x_{10} = 196$, $x_{11} = 180$ and $x_{12} = 173$

i	x_i	n_i	h_i	H_i
1	172	2	0.167	0.167
2	173	1	0.083	0.250
3	174	1	0.083	0.333
4	178	1	0.083	0.417
5	180	2	0.167	0.583
6	181	1	0.083	0.667
7	182	1	0.083	0.750
8	186	2	0.167	0.917
9	196	1	0.083	1.000

```
plot(ecdf(x), main = "ECDF: x", col = "red")
```

Variance

Three numeric variables a, b and c are given in the following table.

Variable	Data
a	1, 1.1, 1.2, 1.2, 1.3, 1.4, 1.5
b	20, 22, 25, 28, 30, 35, 40
c	-50, 20, 134, 219, 298, 504, 780, 1293

The three variances for these variables in random order are: 201926.5, 0.03 and 50.62. Without using R match up each variable with its variance.

Solution:

$s_a^2 = 0.03$, $s_b^2 = 50.62$, $s_c^2 = 201926.5$

Calculating descriptive statistics

A small sample of lengths in millimetres is given as follows:

$x_1 = 185$, $x_2 = 148$, $x_3 = 172$, $x_4 = 126$, $x_5 = 197$, $x_6 = 198$, $x_7 = 121$, $x_8 = 158$, $x_9 = 153$ and $x_{10} = 142$

Calculate the following without using R. Include the units in your answer.

- the sample size,
- the mean,
- the variance,
- the standard deviation,
- the coefficient of variation,
- the median,
- the quartiles.

	result	unit
sample size n	10	*keine*
mean $\overline{x}$	160	*mm*
variance s_x^2	160	mm^2
sd s_x	27.447	*mm*
VC v_c	0.172	*keine*
median $x_{0.5}$	155.5	*mm*
quartiles $x_{0.5}$	143.5 und 181.75	*mm*

Linear transformation

a. Let $y_i = ax_i + b$. Use the definition of the mean $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ to prove that $\overline{y} = a\overline{x} + b$.

b. Use the definition of the variance to prove that $s_y^2 = a^2 s_x^2$.
Hint: start by considering $(y_i - \overline{y})^2$ for just one observation i.

Solution:

a.

$$\begin{aligned}
\overline{y} &= \frac{1}{n}\sum_{i=1}^{n} y_i \\
&= \frac{1}{n}\sum_{i=1}^{n}(ax_i + b) \\
&= \frac{1}{n}(a\sum_{i=1}^{n}(x_i) + nb) \\
&= a\frac{1}{n}\sum_{i=1}^{n}(x_i) + \frac{1}{n}nb \\
&= a\overline{x} + b
\end{aligned}$$

b.

$$
\begin{aligned}
(y_i - \bar{y})^2 &= (ax_i + b - (a\bar{x} + b))^2 \\
&= (ax_i - a\bar{x})^2 \\
&= a^2(x_i - \bar{x})^2
\end{aligned}
$$

$$
\begin{aligned}
s_y^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \\
&= \frac{1}{n-1}\sum_{i=1}^{n} a^2\,(x_i - \bar{x})^2 \\
&= a^2\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \\
&= a^2 s_x^2.
\end{aligned}
$$