



**Ladislaus von Bortkiewicz Chair of Statistics
Humboldt-Universität zu Berlin**

DATA IN R

STATISTICAL PROGRAMMING LANGUAGES

authors: Dajana Adamietz
Natalie Mangels
Luise Großler






correctors: Elisabeth Bommers
Christoph Schult
Franziska Schulz

closing date: 14.08.2016





TABLE OF CONTENTS

LIST OF FIGURES	III
LIST OF TABLES	IV
1 INTRODUCTION	1
1.1 SPL_fordiffquot / SPL_symdiffquot	1
1.2 SPL_trian	2
1.3 SPL_imp / SPL_dumreg	2
2 MATHEMATICAL THEORY	3
2.1 SPL_fordiffquot	3
2.2 SPL_symdiffquot	3
2.3 SPL_trian	3
2.4 SPL_imp / SPL_dumreg	5
3 IMPLEMENTATION	6
3.1 SPL_fordiffquot	6
3.2 SPL_symdiffquot	7
3.3 SPL_trian	9
3.4 SPL_imp	11
3.5 SPL_dumreg	12
4 TESTING	14
4.1 SPL_fordiffquot	14
4.2 SPL_symdiffquot	16
4.3 SPL_trian	19
4.4 SPL_imp	21
4.5 SPL_dumreg	22
5 CONCLUSION	24
5.1 SPL_fordiffquot / SPL_symdiffquot	24
5.2 SPL_trian	25
5.3 SPL_imp / SPL_dumreg	25
LIST OF BIBLIOGRAPHY	V
DECLARATION OF AUTHORSHIP	VI

LIST OF FIGURES

FIG. 1	options of position of three various points in \mathbb{R}^2	4
FIG. 2	 SPL_fordiffquot_graphic	15
FIG. 3	 SPL_syndiffquot_graphic1	17
FIG. 4	 SPL_syndiffquot_graphic2	18
FIG. 5	 SPL_trian_output1	19
FIG. 6	 SPL_trian_output2	20

LIST OF TABLES

TAB. 1	 SPL_fordiffquot_table1	14
TAB. 2	 SPL_fordiffquot_table2	15
TAB. 3	 SPL_symdiffquot_table1	16
TAB. 4	 SPL_symdiffquot_table2	18

1 INTRODUCTION

The main topic of this report is called DATA IN R. Our aim is to show the various possibilities of the programming language R. Therefore, we consider different options for creating and using data to emphasize the wide spread functionality of R. More exactly, we produce data by using the functional command, enter data by the input command and read external data. Furthermore, we use the generated data for different parts of mathematics. So we consider various issues of the numerical mathematics,¹ linear algebra and analytic geometry² and statistics,³ too.

1.1 SPL_fordiffquot / SPL_syndiffquot

The calculation of the first derivation f' of the function f is an important part of the numerical mathematics. To generate the approximation of the first derivation, we use the programming language R and another technical calculator. More exactly, we use the forward and symmetric difference quotient to calculate the first derivation $f'(x)$ of the cosine function $f(x) = \cos(x)$ at the position $x = 1$ and consider the order of the absolute error for several difference quotients. With the help of tabular overviews and graphical representations we compare the calculated data with the mathematical theory. Here we will see, that the corresponding theory and R not produce the same results. Our motivation to write about this topic is to demonstrate the limits of the programming language R, especially of calculations and graphical representations.⁴

¹ See quantlet SPL_fordiffquot and SPL_syndiffquot.

² See quantlet SPL_trian.

³ See quantlet SPL_imp and SPL_dumreg.

⁴ Cp. Rohwedder (2016): Angewandte Mathematik I - Programmieren mit Python.

1.2 SPL_trian

In this report we will show an example how linear algebra and analytic geometry can be handle with R. The programming language R is a multifunctional program, which can also manages mathematical contexts. Therefore, we develop a quantlet to determine, if three points form a triangle or not. Triangles are an important topic in mathematics, especially of analytic geometry and linear algebra, as you can dissect all geometric figures in triangles to analyse them. Therefore, the knowledge of handling triangles results in knowledge of all other figures. We want to develop a quantlet, which can be uses frequently for a quick request of various points. In detail, it is more comfortable to enter the x and y coordinates into the terminal instead of modifying several lines of the quantlet within the program R. That means, whenever you enter the six coordinates of three points, you will immediately get the desired information. If the inserted points form a triangle, they will represent in a graphic, to see how the triangle looks like.

1.3 SPL_imp / SPL_dumreg

As an analysis starts with having access to your data with which you want to work with in R, we give a closer look to this import to see what is necessary and possible. The programming language R can read different types of data, for example *txt* or *csv*. In this report we only focus on the preparation of *csv* files. For working and analyzing your own data, there have to be some commands to structure your data. More exactly, we describe how to get the data into R and how to prepare it so, that you can easily work with it in the program R. Our data set derives from a survey, where the motives of contributing to a public good were investigated. That data consists out of 82 complete observations and in total 83 variables. Therefor, it is necessary to put the answers into a verbal expression. In this report the statistical analysis does not stand in the foreground. Instead we want to show on an example of a linear regression how to prepare the variables, in that case the vectors, to analyze our data. That is very important, because in the programming language R you have to modify your data so, that you are able to use it in anyway.

2 MATHEMATICAL THEORY

2.1 SPL_fordiffquot

Let $f : (a, b) \rightarrow \mathbb{R}$ be a differentiable function, further be a and $b \in \mathbb{R}$, x and $x + h \in (a, b)$ and $h > 0$. The forward difference quotient of the function f at the position x is defined by $D_h f(x) = \frac{f(x+h)-f(x)}{h}$. The parameter h describes the increment. The absolute error of the forward difference quotient is defined by $\triangle_h = \left| \frac{f(x+h)-f(x)}{h} - f'(x) \right|$ and the associated order of absolute error is equal to one.⁵

2.2 SPL_symdiffquot

Let $f : (a, b) \rightarrow \mathbb{R}$ be a differentiable function, further be a and $b \in \mathbb{R}$, $x - h$ and $x + h \in (a, b)$ and $h > 0$. The symmetric difference quotient of the function f at the position x is defined by $D_h f(x) = \frac{f(x+h)-f(x-h)}{2 \cdot h}$. The parameter h describes the increment. The absolute error of the symmetric difference quotient is defined by $\Delta_h = \left| f'(x) - \frac{f(x+h)-f(x-h)}{2 \cdot h} \right|$ and the associated order of absolute error is equal to two.⁶

2.3 SPL_trian

If three disparate points are not arranged on a linear slope, they will define a triangle. These points are called basic points and look like $A(x_a/y_a)$, $B(x_b/y_b)$ and $C(x_c/y_c)$. The distance between two points is called flank. Points as well as flanks can be analytical described by a vector. Vectors have an x and y coordinate when we examine triangles in a two-dimensional space, more exactly in \mathbb{R}^2 . For example, a two-dimensional vector looks like $\begin{pmatrix} x \\ y \end{pmatrix}$.

⁵ Cp. Arens et al. (2012): Mathematik, p. 284ff.

⁶ Cp. Arens et al. (2012): Mathematik, p. 320ff, p. 980.

In that case, the vectors are called \vec{AB} and \vec{AC} for the distances between A and B further A and C , respectively. These vectors can be calculated in the following way $\vec{AB} = \begin{pmatrix} x_b - x_a \\ y_b - y_a \end{pmatrix}$ and $\vec{AC} = \begin{pmatrix} x_c - x_a \\ y_c - y_a \end{pmatrix}$. That is important for analyse the algebraic structure.

In a two-dimensional domain there are two different options of the position of three various points. In detail, the points form a triangle (A) or arrange on a linear slope (B). The following figure shows the different cases from the geometrical perspective.

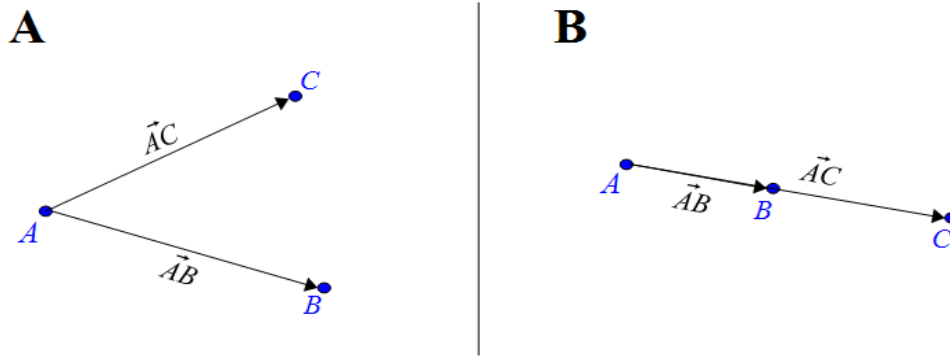


FIGURE 1: options of position of three various points in \mathbb{R}^2 ⁷

From the analytical perspective, the points A , B and C form a triangle when they are not arrange on a linear slope but span a plain. That means, the vectors \vec{AB} and \vec{AC} have to be linearly independent (in detail, a vector cannot be formed by transformation of another vector). The linear independence will show, if the determinant of the defined matrix M of the vectors \vec{AB} and \vec{AC} is not equal to zero.

$$\det(M) = \det \begin{bmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{bmatrix} = (x_b - x_a) \cdot (y_c - y_a) - (x_c - x_a) \cdot (y_b - y_a) \neq 0. \quad ^8$$

⁷ The figure 1 is created by Natalie Mangels.

⁸ Cp. Hainzl (1985): Mathematik für Naturwissenschaftler, p. 170-201.

2.4 SPL_imp / SPL_dumreg

The aim of the linear regression is to describe a variable with another. The model to show the relationship between the dependent variable y and the independent variables x_1, x_2, \dots, x_n is linear with these parameters and looks like $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + e$. Where y is the dependent variable we want to explain, x_1, x_2, \dots, x_n are the describing variables and e is the residual error, which is not observable. The variables $\beta_1, \beta_2, \dots, \beta_n$ are unknown parameters, which are estimated in this model by the observations of x_i for $i = 1, 2, \dots, n$ and $n \in \mathbb{N}$. The variable x is deterministic and not constant. The residual error e is a random variable and differs unsystematically. The interpretation of the result is how the value of the dependent variable changes when an independent variable is varied and all of the others are fixed.⁹

In this report we use the linear regression model to see whether there are dependences between the dependent variable *voluntarily paid price* and the independent variable *income*. To investigate that issue we prove the linear regression model with four dummy variables of the income groups as $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$ where β_0 is the intercept of the model and in our special case the average price of the income group 0 € - 500 € and of the incomes, where no information was given by the participants, and serves as a control variable. The coefficients $\beta_1, \beta_2, \beta_3$ and β_4 show how the average price of these income groups is related to the one of the control group. The variable y is as it was said the *voluntarily price paid* and the independent variables x_1, x_2, x_3 and x_4 are the dummy variables for the income groups. With that model we cannot get a regression line as a result but we would be able to see a linear relation between the price paid and income. Our prediction is, that the price rises with income. The coefficients in the equation are the change in the predicted value of the dependent variable y per unit of change in x_1 and so on. We want to find the best estimates for the coefficients by minimizing the residual error between the actual value of y and the predicted value of it. Therefor, the equation gets modified as $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \varepsilon$, where ε is the residual error.

⁹ Cp. Pruscha (2006): Statistisches Methodenbuch - Verfahren, Fallstudien, Programmcodes, p. 107f.

3 IMPLEMENTATION

3.1 SPL_fordiffquot

LINE 4-5 / 8-9 / 12-13

We use the command *function(parameter)* to define different functions. In detail, we define a function for the increment parameter h , another for the absolute approximation of the first derivation f' of the cosine function $f(x) = \cos(x)$ at the position $x = 1$, which based on the forward difference quotient, and another function for the absolute error of the associated absolute approximation. Therefor, we refer to the mathematical theory.

LINE 16-19

We use the command *supply(list, func)* to calculate data. Therefor, we define an object a , which includes a list of ascending numbers, in detail $a = 1, 2, \dots, 16$. Next we define an object b , which based on the list a and the function of the increment parameter h . The object b creates the following list of numbers $b = 10^{-1}, 10^{-2}, \dots, 10^{-16}$. Next we define an object c , which based on the list b and the function of the absolute approximation. The object c generates the following list of values $c = \left| \frac{\cos(1+10^{-1})-\cos(1)}{10^{-1}} \right|, \left| \frac{\cos(1+10^{-2})-\cos(1)}{10^{-2}} \right|, \dots, \left| \frac{\cos(1+10^{-16})-\cos(1)}{10^{-16}} \right|$. At the end we define an object d , which based on the list b and the function of the absolute error. The object d calculates the list $d = \left| \frac{\cos(1+10^{-1})-\cos(1)}{10^{-1}} + \sin(1) \right|, \left| \frac{\cos(1+10^{-2})-\cos(1)}{10^{-2}} + \sin(1) \right|, \dots, \left| \frac{\cos(1+10^{-16})-\cos(1)}{10^{-16}} + \sin(1) \right|$.

ADDITION

The command *supply(list, func)* enables the application of functions on every object of a list. That is why this method is a faster and more pleasant alternative instead of a for-loop. Further, the command *supply(list, func)* produces vectors. Because of this the generated objects b , c and d are vectors, too.

LINE 22-25

We use the command `cbind(vector1, vector2, vector3)` to create a tabular overview of the data. Therefor, we define a matrix M , which connects the vectors b , c and d in series. Next we refer to the columns of the matrix M the corresponding description and label the rows with ascending numbers. After that we print the matrix M .

LINE 28-29 / 32-33

We use the command `plot(vector1, vector2)` to generate a graphical representation of individual elements of the matrix M . Therefor, we define two vectors and call them e and f . The elements of the vector e are the first eight elements of the first column of the matrix M (rather of the vector b). The elements of the vector f are the first eight elements of the third column of the matrix M (rather of the vector d). At the end of this section, we plot these vectors without restrictions of axis scaling in a labeled graphic. Particularly, we choose a graphical representation, which includes dots and lines.

3.2 SPL_symdiffquot

LINE 4-5 / 8-9 / 12-13

We use the command `function(parameter)` to define the functions of the increment parameter h , the absolute approximation of the first derivation f' of the cosine function $f(x) = \cos(x)$ at the position $x = 1$, which based on the symmetric difference quotient, and the absolute error of the associated absolute approximation. Therefor, we refer to the mathematical theory, too.

LINE 16-19

We use the command *supply(list, func)* to calculate data. At first we define an object *a*, which includes a list of ascending numbers, in detail $a = 1, 2, \dots, 16$. Next we create an object *b*, which based on the list *a* and the function of the increment parameter *h*. More exactly, the object *b* generates the following list of numbers $b = 10^{-1}, 10^{-2}, \dots, 10^{-16}$. Next we define an object *c*, which based on the list *b* and the function of the absolute approximation. The object *c* generates the following list of numbers $c = \left| \frac{\cos(1+10^{-1}) - \cos(1-10^{-1})}{2 \cdot 10^{-1}} \right|, \left| \frac{\cos(1+10^{-2}) - \cos(1-10^{-2})}{2 \cdot 10^{-2}} \right|, \dots, \left| \frac{\cos(1+10^{-16}) - \cos(1-10^{-16})}{2 \cdot 10^{-16}} \right|$. At the end we generate an object *d*, which based on the list *b* and the function of the absolute error. More exactly, the object *d* calculates the following list of numbers $d = \left| -\sin(1) - \frac{\cos(1+10^{-1}) - \cos(1-10^{-1})}{2 \cdot 10^{-1}} \right|, \left| -\sin(1) - \frac{\cos(1+10^{-2}) - \cos(1-10^{-2})}{2 \cdot 10^{-2}} \right|, \dots, \left| -\sin(1) - \frac{\cos(1+10^{-16}) - \cos(1-10^{-16})}{10^{-16}} \right|$.

LINE 22-25

We also use the command *cbind(vector1, vector2, vector3)* to create a tabular overview of the calculated data. Therefor, we define a matrix *M*, which connects the vectors *b*, *c* and *d* in series. At the end we refer to the columns the corresponding descriptions, label the rows with ascending numbers and print the matrix *M*.

LINE 28-29 / 32-35

Finally we also use the command *plot(vector1, vector2)* to generate two graphical representations from individual elements of the matrix *M*. Therefor, we define two vectors and call them *e* and *f*, too. The elements of *e* are the first five elements of the first column of the matrix *M* (rather of the vector *b*). Further, the elements of *f* are the first five elements of the third column of the matrix *M* (rather of the vector *d*). At the end we plot the defined vectors without and with restrictions of axis scaling in a labeled graphic. Especially, we choose graphical representations, which includes dots and lines.

3.3 SPL_trian

LINE 1 / 36-38

As stated in chapter 3.1, there should be an interactive and a processing part in the quantlet. Therefore, a corresponding method/function has to be defined, to facilitate that the quantlet works with the inserted variables and generates the desired output. Our function should be called *input*. We do not need parameters inside the function, so the parentheses are empty. The functionality of the quantlet is described within the braces. A function needs a return value, which is normally the name of the function. We decided, that the return value should attest the *end of the program* in line 36. The command *print(input())* ensures a visible output. If the program shall continue consistently, then the code *return(input())* has to be used, followed by *print(input())* to get the corresponding output.

LINE 4-12

Here the quantlet starts with the interactive part. First the entered *x* coordinate in line 4 and the corresponding *y* coordinate in line 5 of point *A* have to be saved in a variable (cp. *pointAx* and *pointAy*). That is realized by the command *readline()*. The program should ask the user to enter the coordinates. For this reason we coded the command *Key in the first x coordinate and press enter*. Both values are transformed in line 6 to an integer vector. For this purpose we need the command *as.integer*. This data type provides good performance in the program R. The procedure continues until the coordinates of point *C*. As secondary effect the command *as.integer* ensures, that the inputs are numbers and not other variables (like the letter *a*). Decimal numbers are not of interest, but can also be coded in the programming language R.

LINE 15-18

The following commands are responsible for an understandable output. Line 15 should generate a visible line between the output lines in the terminal for optical reasons (see figure 5). The other lines shall give information about which coordinates have been saved by the program.

LINE 21-35

Now the analytical and calculative part starts, where the program R is processing the given input. The coordinates are bound in the $vector1 = \overrightarrow{AB} = \begin{pmatrix} x_b - x_a \\ y_b - y_a \end{pmatrix}$ and $vector2 = \overrightarrow{AC} = \begin{pmatrix} x_c - x_a \\ y_c - y_a \end{pmatrix}$ in line 21 and 22. By using the command $c()$, the program knows, that the object has to be a vector. Next we defined a matrix of these vectors in line 25 with the command $rbind()$, which is responsible for connecting vectors in series. To check, if the points create a triangle and if the determinant of the matrix is not equal to zero, we created a conditional execution for the two possible cases: (A) the coordinates form a triangle or (B) the coordinates do not form a triangle. Inside the parentheses of the command $if()$ the defined conditions of the variables are set. In our program, the defined condition is the determinant, which is equal to zero. If that is fulfilled, the code within the curly braces should be executed. Further, the text *The points do not generate a triangle* should be given as output. If the condition does not comply the command $else$ is executed and the points should be plotted. Therefore, we define a matrix of all points and the order $plot()$. Within the command $plot()$ the characteristics of graphic can be defined. First it has to be determined, what has to be plotted – here it is $matrix2$. Afterwards, the representation type should be defined – in this case points are plotted. The commands $xlab$ and $ylab$ define the description of the graphical axes. Line 33 formatting the graphic. Further, in line 35 we code one more visible line, which is an optical effect and does not essential to run the quantlet.

ADDITION

The linear dependence can also be determined by another method. As mentioned in chapter 2.3, two vectors in a two-dimensional space will be linear dependent, if they are multiples of each other. The alternative computational approach is $\overrightarrow{AB} = \lambda \cdot \overrightarrow{AC}$. If there exists $\lambda \neq 0$, the vectors will be linear dependent. However, the coding of that method is more inconvenient and needs more individual steps than our method. For the graphical representation the program R provides additional features, like colored arrangements and formatting of text modules.

3.4 SPL_imp

The quantlet loads the data into the program R via the commands *setwd()*, *getwd()* and further *read.table()*. With the command *setwd()* you give the program the path of the files so that R can get these data. Hereby, it is important to change the slashes when you copy the file, because the programming language R just can read forward slashes / or double backslashes \\ as path separators. It is also necessary to put the path into quotes. The command *getwd()* shows you the path for your control. Next step is to give the exact name of the file you want to import. The name of it has to be in quotes. We call that *data_file*. This name will be used for next commands to shorten the length. To import our data set, which is saved as csv file, we use the command *read.table()*. That command creates a data frame, where the rows of the table, which you load, are the cases and the columns stand for the variables. Within there are a lot of arguments to define. Here are the arguments we modify (arguments, which are not listed, are like the default).

- `file = data_file`
(the name we give as we reported above)
- `sep = " \t "`
(data in our csv file is separated by tabs)
- `quote = " \" "`
(means, the character strings are in double quotes)
- `rownames = " CASE "`
(the first column in the table which gives a number to each observation)
- `colnames = ...`
(listing every name of a column in this data set)
- `stringsAsFactors = FALSE`
(so the character vectors are not converted into factors)
- `skip = 1`
(because the first line consists of the names of columns)
- `fill = TRUE`
(because the rows have unequal length as there are different cases in the data set:
Reading news online? Yes/No)
- `comment.char = " "`
(to turn off the interpretation of comments)

Printing *data*, which is how we called the command *read.table()*, shows a huge matrix, where the responses of the survey are coded in numbers, which is difficult for interpretation. That is why we determine the value parameters by using the command *factor()*. The following example shows a nominal variable (**LINE 93-95**).

```
data$E001 = factor(data$E001, levels = c("1", "2", "x-9"),  
                  labels = c("Ja", "Nein", "nicht beantwortet"),  
                  ordered = FALSE)
```

The first argument is the vector of data, which we now prepare, *levels* is equal to the vector of values that for example *data\$E001* might have taken and *labels* is equal to a vector of the labels, which have to be in the same order as the equivalent values in *levels*. The argument *ordered* is equal to a logical flag, where *TRUE* means, that the class is integer, which means that the variable is ordinal and *FALSE* means, that there is no order in the values as it is with nominal variables. This command is not for dichotomous variables. For these variables we used the command *attr(x, which)* and give them the attributes *TRUE* = "*chosen*" or *FALSE* = "*not chosen*". It is also possible to specify these dichotomous variables with the command *comment()*. Though these comments are not visible in the data frame, you can look them up. Printing the data frame *data* after defining all value parameters shows a matrix, which is easier to read and interpret.

3.5 SPL_dumreg

In this quantlet we want to discover whether income has an impact on the variable price. As the variable price is numeric we do not determine it via the command *factor()* or in another way. So we define the 30th column as *price* and then it appear in the global environment. The price depends on another variable, the interval in which participants pay. There are three possible intervals in the survey, monthly or yearly or just once. To compare the prices in the different intervals we had to make them comparable. We decide to standardize them to a monthly interval.

But therefor, it was not possible to work with the prices, which are paid *just once*. That is why we left them out, also the not answered ones. In the following, we describe how we prepare the price vector. At first we create a new vector for the interval (**LINE 8-9**).

```
quest.Int = ifelse(data$P002 == "einmalig" | data$P002 == "nicht beantwortet",  
                  0, 1)
```

As you can see we get a binary vector by recoding *just once* and *not answered* into zero and *monthly* and *yearly* get one. When we print the command `table()` we get the counts of each combination of factor levels and we see, that there are just 71 observation left. Next we multiply our price vector with the created binary vector via the command `mapply()` (**LINE 16**).

```
new.price = mapply("*", price, quest.Int)
```

As a result we get a new price vector with the left prices. To make the price comparable we put them to a monthly interval (**LINE 20**).

```
new.data = ifelse(data$P002 == "jaehrlich", new.price/12, new.price)
```

Now we have a consistent price vector, which we can use for our analysis. For the linear regression we define four dummy variables for the income categories (**LINE 28 / 35 / 42 / 49**).

```
income.2 = ifelse(data$SD08 == "501 Euro - 1000 Euro", 1, 0)  
income.3 = ifelse(data$SD08 == "1001 Euro - 2000 Euro", 1, 0)  
income.4 = ifelse(data$SD08 == "2001 Euro - 3000 Euro", 1, 0)  
income.5 = ifelse(data$SD08 == "ueber 3000 Euro", 1, 0)
```

4 TESTING

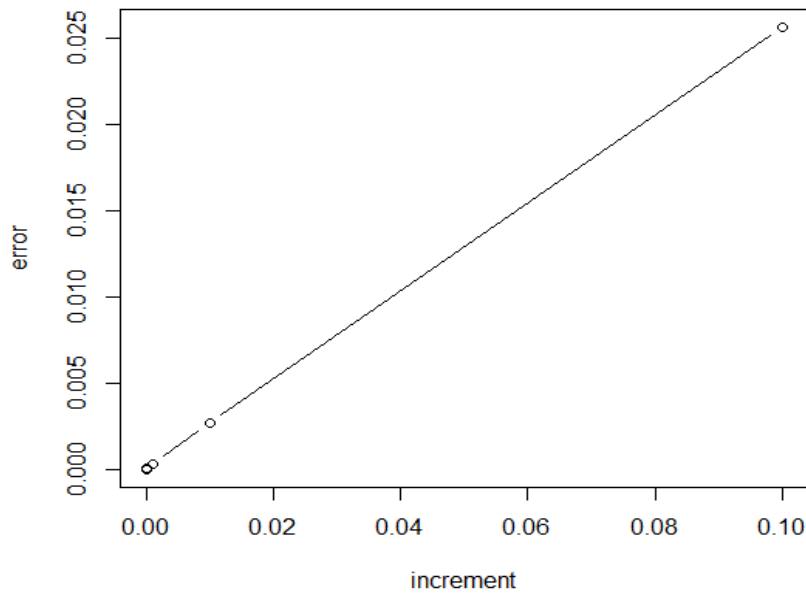

4.1 SPL_fordiffquot

This quantlet creates a table, which consists of four columns. The first column contains the line numbering. The second column shows the values of the increment. The third column includes the absolute values of approximation of the first derivation $f'(1)$ of the cosine function $f(x) = \cos(x)$ for associated increments, which calculated by the forward difference quotient. The fourth column consists of absolute values of the error for associated approximations. For a better overview, we split the table into two tables and explain them separated from each other.

	increment	approximation	error
1	10^{-1}	0.8670618	$2.559086 \cdot 10^{-2}$
2	10^{-2}	0.8441584	$2.687465 \cdot 10^{-3}$
3	10^{-3}	0.8417410	$2.700109 \cdot 10^{-4}$
4	10^{-4}	0.8414980	$2.701371 \cdot 10^{-5}$
5	10^{-5}	0.8414737	$2.701511 \cdot 10^{-6}$
6	10^{-6}	0.8414713	$2.700897 \cdot 10^{-7}$
7	10^{-7}	0.8414710	$2.806113 \cdot 10^{-8}$
8	10^{-8}	0.8414710	$3.025119 \cdot 10^{-9}$


TABLE 1: increment proportional to error  SPL_fordiffquot

The first table (table 1) shows a descending error which is proportional to the increment. In detail, for any further reduction of the increment the accuracy of approximation increases by one decimal place. In regard to the theory, the calculated values confirm the theoretical order of error of the forward difference quotient. Further, the table shows the smallest value of the error (line 8/table 1). This value represents the most accurate approximation, which can be calculated by R. But the most exactly absolute value of $|f'(1)|$ is equal to $|\sin(1)| = 0.841470985$. For that result we have used a calculator with 10 digits of precision, thus 9 digits display.

FIGURE 2: order of error  SPL_fordiffquot

The generated graphic (figure 2) includes the values of the increment and absolute error which are proportional to each other. In regard to the theory, the graphical representation of the first table (table 1) shows the linear relationship between the increment and associated error. That means, the graph confirms the theoretical order of error of the forward difference quotient.

	increment	approximation	error
9	10^{-9}	0.8414711	$1.302016 \cdot 10^{-7}$
10	10^{-10}	0.8414713	$3.522462 \cdot 10^{-7}$
11	10^{-11}	0.8414713	$3.522462 \cdot 10^{-7}$
12	10^{-12}	0.8415491	$7.806786 \cdot 10^{-5}$
13	10^{-13}	0.8415491	$7.806786 \cdot 10^{-5}$
14	10^{-14}	0.8437695	$2.298514 \cdot 10^{-3}$
15	10^{-15}	0.9992007	$1.577297 \cdot 10^{-1}$
16	10^{-16}	0.0000000	$8.414710 \cdot 10^{-1}$

TABLE 2: increment unproportional to error  SPL_fordiffquot

In contrast to the theory, the second table (table 2) shows an ascending absolute error, which is unproportional to the increment. In detail, for any further reduction of the increment the accuracy of the approximation decreases by an inconstant decimal place. But these values are calculated inaccurately by the programming language R. The reason for that is the cancellation effect, thus the subtraction of two almost equally large numbers. This effect can also leads to wrong results. According to the table, the approximation of the first derivation $f'(1)$ of the cosine function $f(x) = \cos(x)$ is equal to 0.0000000 for an increment of 10^{-16} (line 16/table 2). But the most correctly absolute value of $|f'(1)|$ is equal to $|\sin(1)| = 0.841470985$. Because of this the calculated value for the approximation is not only inaccurate but also wrong.

4.2 SPL_syndiffquot

This quantlet creates a tabular overview, which consists of four columns. The first column contains the line numbering of the table. The second column shows the values of the increment. The third column consists of the absolute values of the approximation of the first derivation $f'(1)$ of the cosine function $f(x) = \cos(x)$ for associated increments, which calculated by the symmetric difference quotient. The fourth column includes the absolute values of the error for associated approximations. For a better overview, we split the generated table into two different tables and explain them separated from each other.

	increment	approximation	error
1	10^{-1}	0.8400692	$1.401751 \cdot 10^{-3}$
2	10^{-2}	0.8414570	$1.402445 \cdot 10^{-5}$
3	10^{-3}	0.8414708	$1.402452 \cdot 10^{-7}$
4	10^{-4}	0.8414710	$1.402529 \cdot 10^{-9}$
5	10^{-5}	0.8414710	$1.086409 \cdot 10^{-11}$

TABLE 3: increment proportional to error  SPL_syndiffquot

The first table of this quantlet (table 3) shows a descending error, which is proportional to the increment, too. In contrast to the quantlet `SPL_fordiffquot` (table 1), these errors decrease faster. In detail, for any further reduction of the increment the accuracy of approximation increases by two decimal places. That is why the order of error of the symmetric difference quotient is equal to two. So the calculated values confirm the theoretical order of error. Further, the table shows the smallest value of the error (line 5/table 3). This value also represents the most accurate approximation of the first derivation $f'(1)$ of the cosine function $f(x) = \cos(x)$, which can be calculated by the programming language R. In compare to `SPL_fordiffquot` (line 8/table 1), this error is much smaller. Correct in regard to theory, the approximation of the first derivation f' of a function f , which calculated by the symmetric difference quotient, is more exactly.

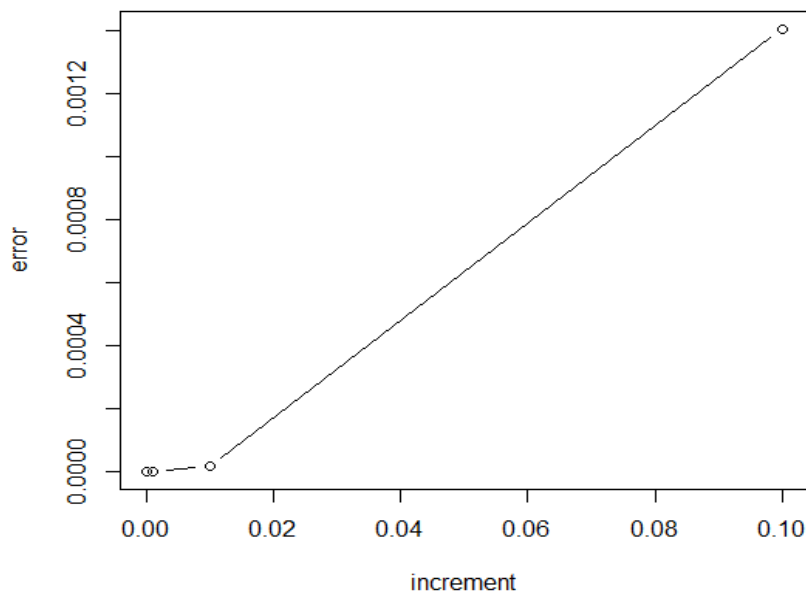


FIGURE 3: order of error  `SPL_syndiffquot`

The generated graphic (figure 3) includes the values of increment and absolute error, which are proportional to each other. In regard to theory, the graphical representation of the calculated values have to show a quadratic relationship between these parameters. But the programming language R connects points with straight instead of curved lines. That is why the graph does not exactly confirm the theoretical order of error of the symmetric difference quotient.

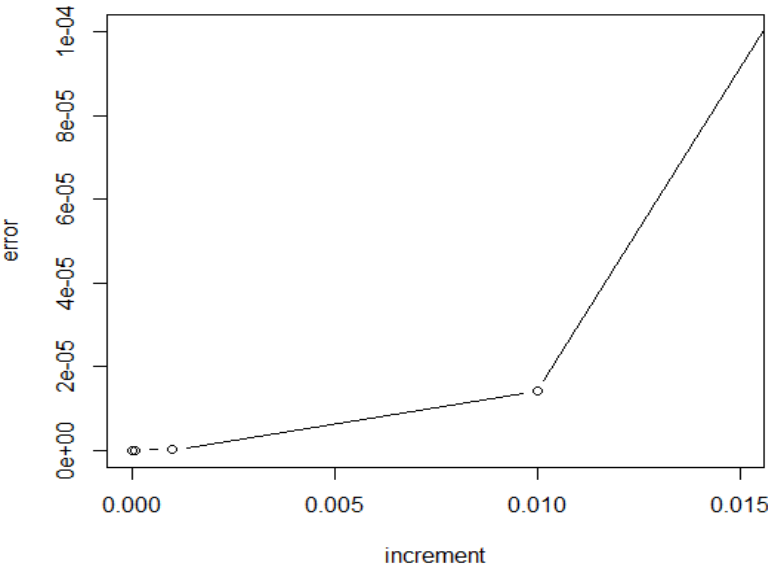


FIGURE 4: order of error 

An additional enlargement of the generated graphic (figure 4) shows the explained limit of graphical representations, which produced by the programming language R in more detail.

	increment	approximation	error
6	10^{-6}	0.8414710	$2.751732 \cdot 10^{-11}$
7	10^{-7}	0.8414710	$3.055496 \cdot 10^{-10}$
8	10^{-8}	0.8414710	$3.025119 \cdot 10^{-8}$
9	10^{-9}	0.8414710	$1.917934 \cdot 10^{-7}$
10	10^{-10}	0.8414708	$2.028653 \cdot 10^{-7}$
11	10^{-11}	0.8414713	$3.522462 \cdot 10^{-7}$
12	10^{-12}	0.8414935	$2.255671 \cdot 10^{-5}$
13	10^{-13}	0.8415491	$7.806786 \cdot 10^{-5}$
14	10^{-14}	0.8382184	$3.252601 \cdot 10^{-3}$
15	10^{-15}	0.8881784	$4.670743 \cdot 10^{-2}$
16	10^{-16}	0.0000000	$8.414710 \cdot 10^{-1}$

TABLE 4: increment unproportional to error 

In contrast to the theory, the second table of this quantlet (table 4) also shows an ascending error, which is unproportional to the increment. More exactly, for any further reduction of the increment the accuracy of the approximation decreases by an inconstant decimal place. But these values are also calculated inaccurately by the programming language R. The reason for that is the cancellation effect, too (cp. `SPL_fordiffquot` / table 2).


4.3 SPL_trian

By running the quantlet, the user is asked to enter several coordinates. After providing all requested data into the terminal, as expected, the first output of the quantlet is displayed. More exactly, the points and corresponding coordinates are described as *The first/second/third point includes the coordinates*. Between the input-lines and the desired output-lines a strip-line is shown for differentiation. In parallel the program R tests the points for linear independence. In case the determinant is equal to zero and the points are arranged in a linear slope, R displays *These points do not generate a triangle*. That is successfully tested for the following three points $A(1,2)$ and $B(3,4)$ and $C(5,6)$.

```

Key in the first x coordinate and press enter : 1
Key in the first y coordinate and press enter : 2
Key in the second x coordinate and press enter: 3
Key in the second y coordinate and press enter: 4
Key in the third x coordinate and press enter : 5
Key in the third y coordinate and press enter : 6
[1] "-----"
[1] "The first point includes the coordinates : "
[1] 1 2
[1] "The second point includes the coordinates: "
[1] 3 4
[1] "The third point includes the coordinates : "
[1] 5 6
[1] "These points do not generate a triangle"
[1] "-----"
[1] "end of program"

```

FIGURE 5: output in case of linear dependence  SPL_trian

In case that three points form a triangle, the calculated determinant is not equal to zero and the quantlet shows a graphical representation of the entered data points. That case is successfully tested with another three points $A(-3,0)$ and $B(1,0)$ and $C(3,8)$.

```

Key in the first x coordinate and press enter : -3
Key in the first y coordinate and press enter : 0
Key in the second x coordinate and press enter: 1
Key in the second y coordinate and press enter: 0
Key in the third x coordinate and press enter : 3
Key in the third y coordinate and press enter : 8
[1] "-----"
[1] "The first point includes the coordinates : "
[1] -3 0
[1] "The second point includes the coordinates: "
[1] 1 0
[1] "The third point includes the coordinates : "
[1] 3 8
[1] "-----"
[1] "end of program"

```

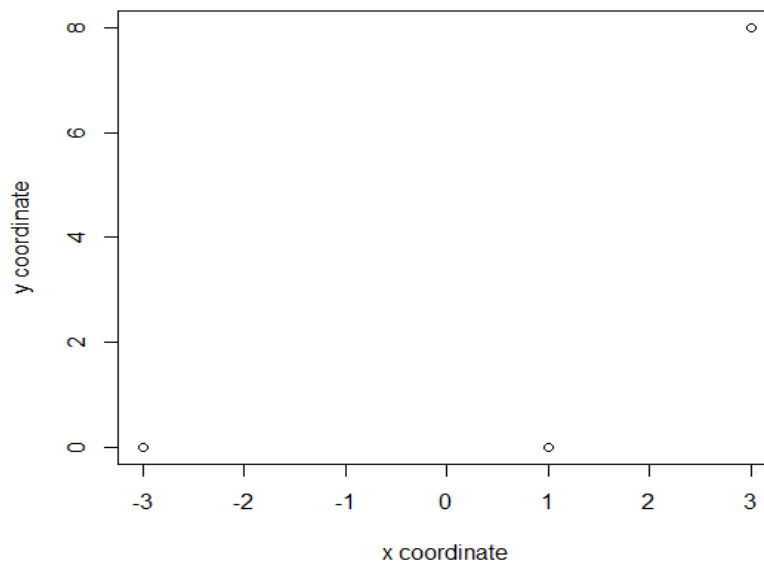


FIGURE 6: output in case of linear independence  SPL_trian

The points A and B and C are displayed in a kind of coordinate system, which links the analytical and the geometrical perspectives. In the case of linear independence, it is important to see, how the points are arranged in a two-dimensional system to form a triangle. If the points do not generate a triangle, it is not meaningful to know, where the points are located.

4.4 SPL_imp

INPUT

In our first try to import the data via the command `read.table()` we define the names of columns and the classes. When we have a closer look to the variables and how they are classified in the source, we decide to change the data types of three variables, because the classification in the source does not make sense to us.

EXPECTED OUTPUT

After importing the data via the command `read.table()` with defining the names of columns and their classes, we expect the *data* to appear in the *global environment*, so that we can print *data*.

ACTUAL OUTPUT

We get this error message *Error in scan (file, what, nmax, sep, dec, quote, skip, nlines, na.strings: scan() expected 'an integer', got 'dec="."', row.names="CASE", col.names = c("CASE", "SERIAL", "REF", "QUESTNNR", "MODE", "STARTED"))*. So the import does not work.

COMMENT

We think about the reason for this error and come to the conclusion, that the program R first needs the data as it is defined in the source from where we load. At first we import the data without specifying in detail, so that we can change the data type of some variables. After using the command `read.table()`, the data is uploaded in the program R. Now we change the data types (classes) of some variables. The variable *E004* gives the amount of choices to a question with multiple response possibilities and is defined in the data set as integer. But we change it into numeric. The variable *ZG01* counts the times the urn is filled again for a random sampling and is also defined as integer. We change it into numeric, too. The last variable *SD04*, again an integer variable, is same kind of question as with variable *E004*. We also change it into numeric.

Afterwards, we kind of reload the data set by using again the command *read.table()* and modifying the classes of columns. Here we classify the three variables as numeric, too. This reload is important because otherwise it is not possible to change the data types of the variables. When we read the table again into the programming language R, we define the same arguments as above with just adding *colclasses = c()* and giving each name of column the corresponding class. We also exclude some irrelevant variables, which are not needed for analysis by calling them *NULL*, further classifying them as *NULL*. After the first import there are 82 observations and 82 variables. By comparison, after the second import there are just 68 variables.

4.5 SPL_dumreg

INPUT I

As we want to do an analysis with the price and the income, we have to prepare the price vector. In the chapter IMPLEMENTATION, we already mention, that the price is depending on the interval in which the participant could pay. So we have to standardize the prices to monthly. The prices, which are paid just once are a problem. That is why we decide to exclude them. We try different ways to prepare to price vector depending on the results of the interval vector.

EXPECTED RESULT I

As a result we want to have a price vector with all the given prices from each participant to analyze them and to see, what might influence these given prices, e.g. the influence of income.

ACTUAL RESULT I

The price vector of our quantlet is still consisting of 82 values. Due to the command *mapply()* the prices, which are paid once and where the answer is missing, are equal to zero. Afterwards, the price vector is standardized to monthly payments for a better comparison.

COMMENT I

For an interpretation of this result it is plausible to say that a small amount of money, which is paid just once, is approximately equal to zero in the long time horizon. So it would have been also possible to divide every price, which is paid just once, with 120 months or even more and get prices nearly to zero afterwards.

INPUT / EXPECTED OUTPUT II

To analyze the relation between price and income, we operate a multiple regression with four dummy variables, which we have created. The model is as follows $new.data = \beta_0 + \beta_1 income.2 + \beta_2 income.3 + \beta_3 income.4 + \beta_4 income.5 + \varepsilon$. The intercept β_0 is consisting of the average of prices paid by the first income group and of the ones, who do not give an answer (missing value) or have chosen not to give this detail. Our prediction is that there is a linear relation and the price is rising with the income. This would be viable in positive and increasing coefficients.

ACTUAL OUTPUT II

As we can see, the average price paid by the lowest income group and where the detail about income is missing is 4,46 €. Surprisingly, the price paid by people with income 500 € - 2000 € is less. The very high deviation with income.5 is because there were just two observations and one of them is with a price about 25 € very high. So we cannot really prove our assumption. But it is important to say, that the intercept consists of five observations with no information about income and one of them is again with a price about 25 € very high.

Call:

```
lm(formula = new.data ~ income.2 + income.3 + income.4 + income.5)
```

Coefficients:

(Intercept)	income.2	income.3	income.4	income.5
4.46077	-1.25649	-0.07744	2.50219	9.03923

COMMENT II

Problem with this regression is, that income is a metric variable, but in this survey it is put into categories, so that it changes to an ordinal variable. Another point is, that the income groups have not the same size, which makes a statistical analysis difficult and might influence the result.

```
> table(data$SD08)
      0 € - 500 €      501 € - 1000 €
           8          23
    1001 € - 2000 €    2001 € - 3000 €
          35           9
    über 3000 € Ich möchte dazu keine Angabe machen.
           2           4
nicht beantwortet
           1
```

5 CONCLUSION**5.1 SPL_fordiffquot / SPL_syndiffquot**

In summary, the programming language R is a suitable program to consider issues of numerical mathematics. But you need to note the limits of correct calculations. The reason for that is the cancellation effect, thus the subtraction of two almost equally large numbers. We have shown that this effect leads to inaccurate and moreover to wrong results, although we correctly operate the mathematical theory. To minimize the cancellation effect you have to calculate an optimal increment $h_{optimal} \cong \sqrt{10^{-n} \cdot 2 \cdot \frac{|f(x)|}{|f''(x)|}}$. Furthermore, you need to note the limits of graphical representations. We have shown that R connects points with straight instead of curved lines. To generate an approximately curved graph you have to calculate more values for $h \geq h_{optimal}$.¹⁰

¹⁰ Rohwedder (2016): Angewandte Mathematik I - Programmieren mit Python.

5.2 SPL_trian

The programming language R is a multifunctional program, which can also handle much more mathematical topics than statistics. Our quantlet is a simple example for the realization of algebraic calculations. Complex mathematical structures and calculations can be executed with content packages, which can be loaded into the program R. In addition, we show that R can be used in interactive manner. That the user can input his relevant coordinates into the terminal, constitutes a huge easement. Instead of modifying the lines of quantlets themselves, which can be very challenging for inexperienced users, the data are requested by the quantlet and entered into the terminal interactively. After the testing of three points is finished, the quantlet can be easily restarted to test another set of data points.

5.3 SPL_imp / SPL_dumreg

After the preparation of the data we receive a data frame, which is better to read, so that the analysis and the interpretation is easier. In fact, there are several steps before you can really work with the data, but when this is done, you can save time, because you do not have to look up the verbal expressions of the values. Within our analysis we use the data from a real empirical study. In contrast to the data sets you can load in the program R the analysis and the interpretation is more difficult, because the results are not always consistent, but therefor very interesting.

LIST OF BIBLIOGRAPHY

Arens, T./Hettlich, F./Karfinger, Ch./Kockelkorn, U./Lichtenegger, K./Stachel, H.:

Mathematik, 2. Auflage, Spektrum Akademischer Verlag, Heidelberg, 2012.

Hainzl, Josef:

Mathematik für Naturwissenschaftler, 4. Auflage, Springer Fachmedien, Wiesbaden, 1985.

Ligges, Uwe:

Programmieren mit R - Statistik und ihre Anwendungen, 2. Auflage, Springer Verlag, Berlin, Heidelberg, 2006.

Pruscha, Helmut:

Statistisches Methodenbuch - Verfahren, Fallstudien, Programmcodes, Springer Verlag, Berlin, Heidelberg, New York, 2006.

Rohwedder, Thorsten:

Angewandte Mathematik I - Programmieren mit Python, Lehrstuhl für Mathematik, Humboldt-Universität zu Berlin, 2016.

DECLARATION OF AUTHORSHIP

We (Dajana Adamietz, Natalie Mangels, Luise Gossler) hereby declare that we have not previously submitted the present work for other examinations. We wrote this work independently. All sources, including sources from the internet, that we have reproduced in either an unaltered or modified form (particularly sources for texts, graphs, tables and images), have been acknowledged by us as such.



.....
(SIGNATURES)