DATA IN R

Dajana Adamietz
Natalie Mangels
Luise Großler

Ladislaus von Bortkiewicz Chair of Statistics
Humboldt–Universität zu Berlin
http://lvb.wiwi.hu-berlin.de

INTRODUCTION

- consider different options for creating and using data
- show the various possibilities and wide spread functionality of the programming language R

TABLE OF CONTENTS

MATHEMATICAL THEORY

We want to approximate $f'(1)$ of $f(x) = cos(x)$
and show the order of error.

- forward difference quotient: $D_h f(x) = \frac{f(x+h) - f(x)}{h}$
- order of absolute error: $\Delta_h = \left| \frac{f(x+h) - f(x)}{h} - f'(x) \right|$

```r
# define function for ...

# ... increment parameter
increment = function(x)
  return(10^(-x))

# ... absolute approximation
approximation = function(h)
  return(abs((cos(1 + h) - cos(1)) / h))

# ... absolute error
error = function(h)
  return(abs(((cos(1 + h) - cos(1)) / h) + sin(1)))
```

```
# create objects
a = 1:16
b = sapply(a, increment)
c = sapply(b, approximation)
d = sapply(b, error)

# generate table
M = cbind(b, c, d)
colnames(M) = c("increment", "approximation", "error")
rownames(M) = c(1:16)
print(M)

# create vectors
e = c(b[1:8])
f = c(d[1:8])

# generate graphic
plot(e, f, type = "b", xlab = "increment", ylab = "error",
     main = "order of error")
```

	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

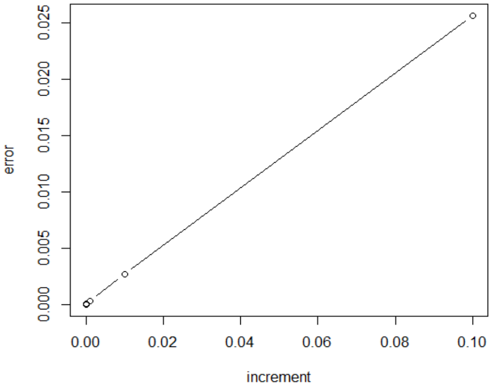

Figure 1: order of error SPL_fordiffquot

	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

MATHEMATICAL THEORY

We want to approximate $f'(1)$ of $f(x) = cos(x)$
and show the order of error.

- symmetric difference quotient: $D_h f(x) = \frac{f(x+h) - f(x-h)}{2 \cdot h}$
- order of absolute error: $\Delta_h = | f'(x) - \frac{f(x+h) - f(x-h)}{2 \cdot h} |$

```r
# define function for ...

# ... increment parameter
increment = function(x)
  return(10^(-x))

# ... absolute approximation
approximation = function(h)
  return(abs((cos(1 + h) - cos(1 - h)) / (2 * h)))

# ... absolute error
error = function(h)
  return(abs(-sin(1) - (cos(1 + h) - cos(1 - h)) / (2 * h)))
```

```
# create objects
a = 1:16
b = sapply(a, increment)
c = sapply(b, approximation)
d = sapply(b, error)

# generate table
M = cbind(b, c, d)
colnames(M) = c("increment", "approximation", "error")
rownames(M) = c(1:16)
print(M)

# create vectors
e = c(b[1:5])
f = c(d[1:5])

# generate graphics
plot(e, f, type = "b", xlab = "increment", ylab = "error",
     main = "order of error")
plot(e, f, type = "b", xlim = c(0, 0.015), ylim = c(0, 0.0001),
     xlab = "increment", ylab = "error", main = "order of error")
```

	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_symdiffquot

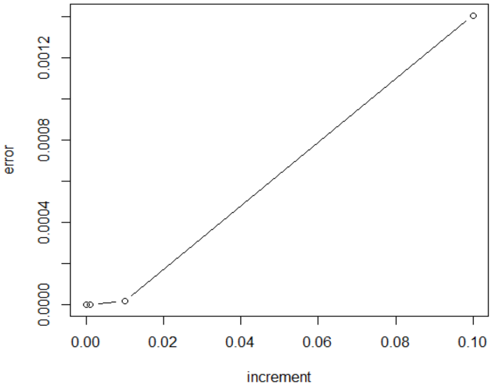

Figure 2: order of error SPL_symdiffquot

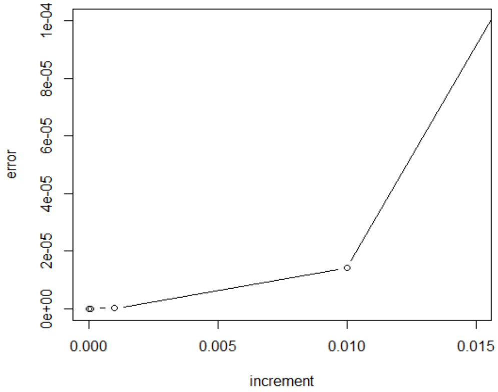

Figure 3: order of error SPL_symdiffquot

	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 SPL_symdiffquot

MATHEMATICAL THEORY I

We want to check whether the points $A(x_a/y_a)$ and $B(x_b/y_b)$ and $C(x_c/y_c)$ generate a triangle.

- $\overrightarrow{AB} = \begin{pmatrix} x_b - x_a \\ y_b - y_a \end{pmatrix}$ and $\overrightarrow{AC} = \begin{pmatrix} x_c - x_a \\ y_c - y_a \end{pmatrix}$

  define the matrix: $\begin{bmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{bmatrix}$

MATHEMATICAL THEORY II

- if the points arrange on a linear slope,
  the determinate of the matrix is equal to 0

$$det(M) = (x_b - x_a) \cdot (y_c - y_a) - (x_c - x_a) \cdot (y_b - y_a) = 0$$

- if not, the vectors are linear independent
  and create a triangle

$$det(M) = (x_b - x_a) \cdot (y_c - y_a) - (x_c - x_a) \cdot (y_b - y_a) \neq 0$$

```r
input=function(){

 # interactive part - user enter coordinates of points A, B and C
 pointAx = readline("Key in the first x coordinate and press enter :");
 pointAy = readline("Key in the first y coordinate and press enter :");
 pointA  = as.integer(c(pointAx, pointAy));
 pointBx = readline("Key in the second x coordinate and press enter:");
 pointBy = readline("Key in the second y coordinate and press enter:");
 pointB  = as.integer(c(pointBx, pointBy));
 pointCx = readline("Key in the third x coordinate and press enter :");
 pointCy = readline("Key in the third y coordinate and press enter :");
 pointC  = as.integer(c(pointCx, pointCy));

 # output - information about coordinates, which have been saved
 print("--------------------------------------------") # visible line
 print("The first point includes the coordinates : "); print(pointA)
 print("The second point includes the coordinates: "); print(pointB)
 print("The third point includes the coordinates : "); print(pointC)
```

```
  # define vectors (based on points A, B and C)
  vector1 = c(pointB - pointA)
  vector2 = c(pointC - pointA)

  # define matrix (based on vectors)
  matrix  = rbind(vector1, vector2)

  # check if points (do not) form a triangle
  if (det(matrix) == 0){
    print("These points do not generate a triangle")
  } else {
    matrix2 = rbind(pointA, pointB, pointC);
    plot(matrix2, type = "p", xlab = "x coordinate", ylab = "y coordinate");
    font = 3; cex = 1; lwd = 6
    }
 print("--------------------------------------------------") # visible line
 return("end of program" )
}
print(input())
```

```
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 4: output in case of linear dependence SPL_trian

```
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 5: output in case of linear independence SPL_trian

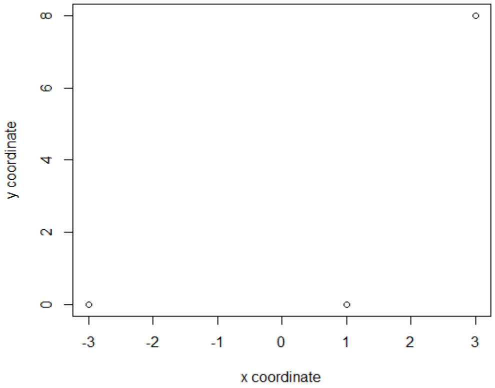

Figure 6: output in case of linear independence SPL_symdiffquot

IMPORT DATA

- data was loaded via *read.table*
- source of data was a survey,
  consists of 82 variables and 82 observations
- some variables were excluded by setting them to *NULL*
  $\rightarrow$ they are not in data frame

```
# path where file is saved
setwd("C:/Users/USER/Desktop/R/16.6")

# shows path which was set
getwd()

# name of file
data_file = "rdata_Pay_What_You_Want_Preismodell_2016-06-16_11-29.csv"
```

```
data = read.table(file = data_file, header = FALSE, sep = "\t",
                  quote = "\"", dec = ".", row.names = "CASE",
                  col.names = c("CASE", "NULL", ...
```

```
                                                ... , "NULL", "NULL"),
                  colClasses = c("integer", "NULL", ...
```

```
                                                ... , "NULL", "NULL"),
                  na.strings = "NA", skip = 1, check.names = TRUE,
                  fill = TRUE, comment.char = "")
```

IMPORT DATA

- all responses are just coded in numbers
- recoding possible answers which are not metric or binary to verbal expressions

```
# verbal expression of values of variables

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

```
data$J003    = factor(data$J003, levels = c("1", "2", "X-9"),
                      labels = c("Ja", "Nein", "nicht beantwortet"),
                      ordered = FALSE)
```

MATHEMATICAL THEORY

- linear model with price as variable we want to explain
- prediction: price paid would increase with income
  $\rightarrow$ independent variable
- to proof coded 4 dummy variables for income groups

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$$

```
# regression with income

# defining dummy variables for income

# vector for income group 500 - 1000 Euro
income.2 = ifelse(data$SD08 == "501 Euro - 1000 Euro", 1, 0)
income.2
plot(income.2, new.data, type = "p", main = "income.2",
     xlab = "income group: 501 Euro - 1000 Euro",
     ylab = "Price paid")
```

```
# vektor for income > 3000 Euro
income.5 = ifelse(data$SD08 == "ueber 3000 Euro", 1, 0)
income.5
plot(income.5, new.data, type = "p", main = "income.5",
     xlab = "income group: more than 3000 Euro",
     ylab = "Price paid")

# multiple regression
model = lm(new.data ~ income.2 + income.3 + income.4 + income.5)
model
summary(model)
```

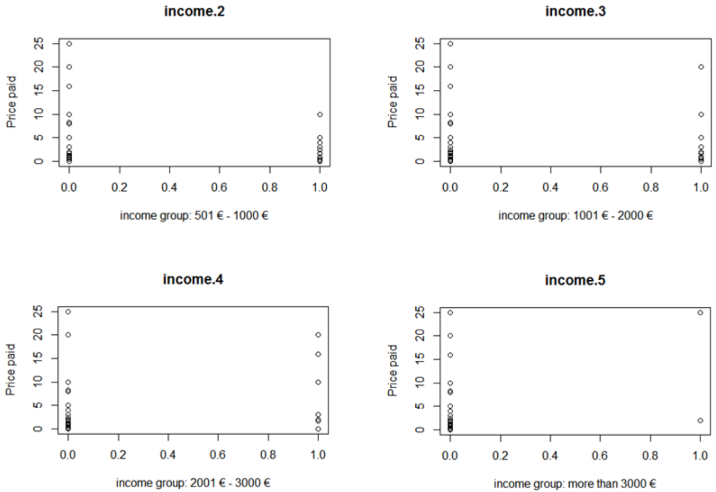

Figure 7: output dummy regression SPL_dumreg

OUTPUT

```
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
```

Figure 8: output dummy regression SPL_dumreg

OUTPUT

- lowest income group (0 - 500 Euro) works as intercept and gives average price paid by income group
- all coefficients of dummy variables show, how average prices of income groups are related to average price of lowest group
- no increase in price depending on income group

THANKS FOR YOUR ATTENTION

Dajana Adamietz
Natalie Mangels
Luise Großler

Ladislaus von Bortkiewicz Chair of Statistics
Humboldt–Universität zu Berlin
http://lvb.wiwi.hu-berlin.de