UNIVERSITÄT BERN

Statistical methods for archaeological data analysis I: Basic methods

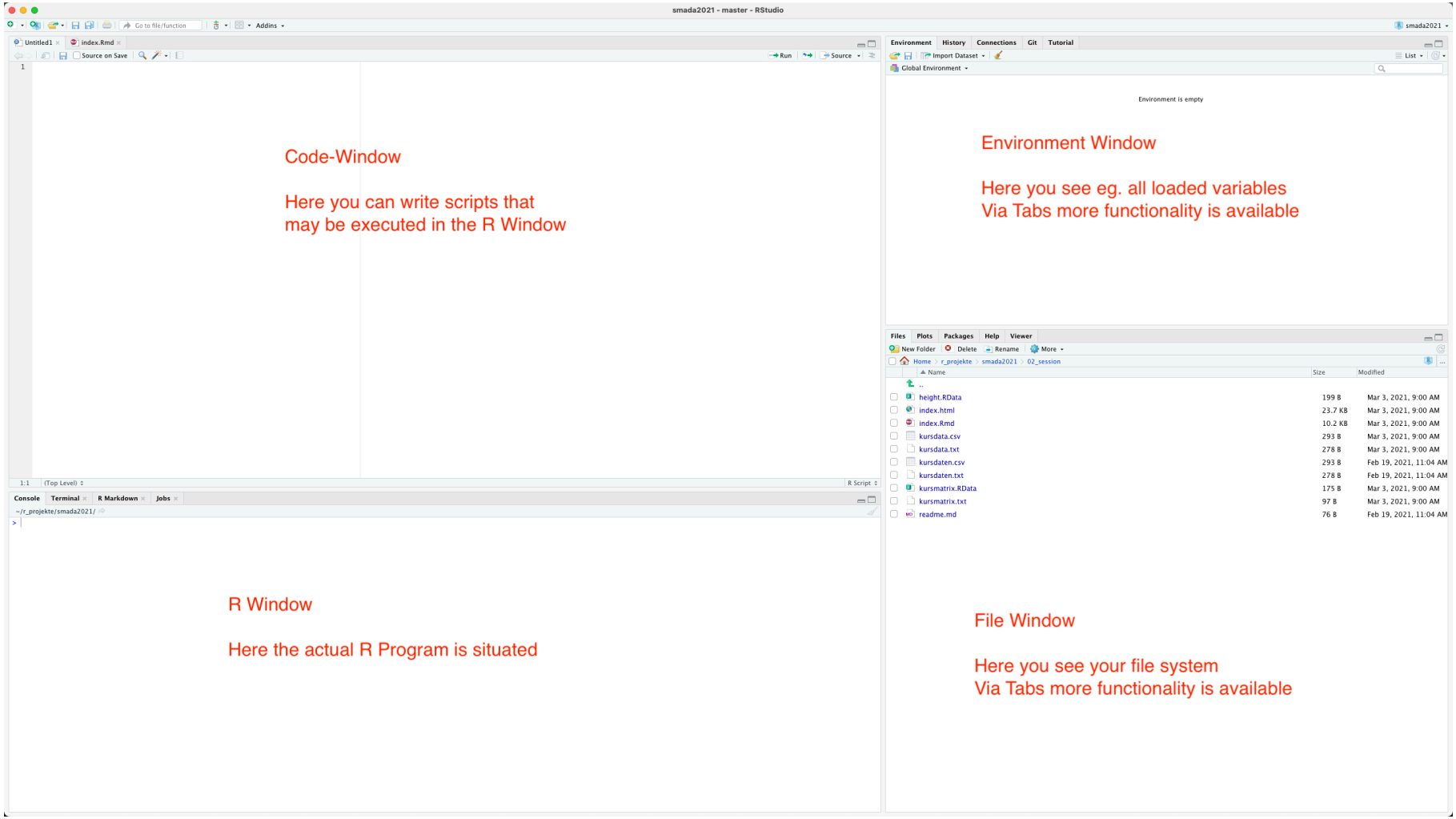
02 - Introduction into R

Martin Hinz

Institut für Archäologische Wissenschaften, Universität Bern

03.03.2021

Start R-Studio



Using R

Start of the system:

After R is started, you end on the prompt.

>

Change the working directory:

```
getwd() # or something else

setwd("U:\R") # or something else
```

Change the path according to your needs

R as calculator

Simplest way of use:

```
2+2
```

```
## [1] 4
```

```
2^2
```

```
## [1] 4
```

Multiple commands are separated by ;

```
(1 - 2) * 3; 1 - 2 * 3
```

```
## [1] -3
```

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

R as calculator

Using functions:

```
sqrt(2) #square root
```

```
## [1] 1.414214
```

```
log(10) #logarith base e
```

```
## [1] 2.302585
```

```
log(10, 10) #logarith base 10, like log(10, base=10)
```

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

Getting help

Call of the help function:

```
help(sqrt)
```

Even simpler?

```
? sqrt
```

Searching the help:

```
help.search('logarithm')
```

Assignment of data to variables

Naming variables for Values (Assignment):

```
x <- 2 # no message will be given back

x
```

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

```
pi # build in variable
```

```
## [1] 3.141593
```

Arrow or equal sign?

Classic assignment symbol in R is the arrow. Also possible:

```
x=2
```

Both are possible. Matter of tast. <- is clearer, I am using it that way

Working with variables

Display of already uses variables:

```
ls()
```

```
## [1] "x"
```

Delete a variable:

```
rm(x) # no message will be given back
ls()
```

```
## character(0)
```

Using variables

Calculations with variables:

```
x <- 2
y <- 2 * x
z <- sqrt(x) # no message will be given back
```

```
ls()
```

```
## [1] "x" "y" "z"
```

```
y
```

```
## [1] 4
```

```
z
```

```
## [1] 1.414214
```

Exercise variables

Calculation of a circle:

Given is a circle with the radius r=5. Calculate the diameter d (2 * r), the circumference u (2 * π * r) and the area a (π * r^2).

Add area a and circumference u, assign the result to the variable v and delete u and a.

Scalars, vectors, matrices, data frames

Data types in R

Scalar

A single number or date

```
pi
```

```
## [1] 3.141593
```

Vector

A row of numbers or data

```
ls()
```

```
## [1] "x" "y" "z"
```

Scalars, vectors, matrices, data frames

Data types in R

Matrix:

A table of data of the same kind

```
euro.cross
```

```
##              ATS         BEF         DEM         ESP         FIM         FRF
## ATS  1.000000000  2.93161486 0.142135709  12.0917422 0.432093050 0.476702543
## BEF  0.341108927  1.00000000 0.048483759   4.1246012 0.147390797 0.162607493
## DEM  7.035529673 20.62546336 1.000000000  85.0718109 3.040003477 3.353854885
## ESP  0.082701069  0.24244768 0.011754775   1.0000000 0.035734557 0.039423810
## FIM  2.314316324  6.78468413 0.328946992  27.9841163 1.000000000 1.103240477
## FRF  2.097744212  6.14977811 0.298164361  25.3653822 0.906420695 1.000000000
## IEP 17.471976881 51.22110711 2.483391826 211.2666399 7.549519785 8.328935807
## ITL  0.007106602  0.02083382 0.001010102   0.0859312 0.003070713 0.003387735
## LUF  0.341108927  1.00000000 0.048483759   4.1246012 0.147390797 0.162607493
## NLG  6.244151907 18.30544854 0.887516960  75.5026750 2.698054644 2.976603092
## PTE  0.068636087  0.20121457 0.009755639   0.8299299 0.029657176 0.032718997
##              IEP         ITL         LUF         NLG         PTE
## ATS 0.0572345080  140.714229  2.93161486 0.160149851  14.5695951
```

Scalars, vectors, matrices, data frames

Data types in R

Data frame:

A table of data of different kind

```
mtcars
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C         17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE        16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL        17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
```

Using c() for data entry

Assignment of values to a vector:

```
places <- c("Leubingen", "Melz", "Bruszczewo")
```

```
categories <- c("Grab", "Hort", "Siedlung")
categories
```

```
## [1] "Grab"     "Hort"     "Siedlung"
```

```
c(places, categories)
```

```
## [1] "Leubingen"  "Melz"       "Bruszczewo" "Grab"       "Hort"
## [6] "Siedlung"
```

Naming the positions in a vector

```
names(places)<-categories
places
```

```
##         Grab         Hort     Siedlung
## "Leubingen"       "Melz" "Bruszczewo"
```

Functions on vectors [1]

Data:

```
load("height.RData")
height
```

```
## Hannah   Leon  Lukas Leonie
##    154    167    187    165
##   Luka    Lea   Lena    Mia
##    190    176    167    156
##    Tim   Fynn   Anna  Emily
##    154    165    167    171
##  Felix
##    154
```

```
# Sum:
sum(height)
```

```
## [1] 2173
```

```
# Count:
length(height)
```

```
## [1] 13
```

```
# Mean:
sum(height)/length(height)
```

```
## [1] 167.1538
```

```
# Or more convenient:
mean(height)
```

```
## [1] 167.1538
```

Functions on vectors [2]

```
# sort:
sort(height)
```

```
## Hannah    Tim  Felix    Mia Leonie   Fynn   Leon   Lena   Anna  Emily    Lea
##    154    154    154    156    165    165    167    167    167    171    176
##  Lukas   Luka
##    187    190
```

```
# minimum:
min(height)
```

```
## [1] 154
```

```
# maximum:
max(height)
```

```
## [1] 190
```

```
# Or more convenient:
range(height)
```

```
## [1] 154 190
```

Functions on vectors [3]

Change of the values through calculation:

```
height.in.m <- height/100
height.in.m
```

```
## Hannah   Leon  Lukas Leonie   Luka    Lea   Lena    Mia    Tim   Fynn   Anna
##   1.54   1.67   1.87   1.65   1.90   1.76   1.67   1.56   1.54   1.65   1.67
##  Emily  Felix
##   1.71   1.54
```

but:

```
test<-c(1,2,3,4,5,6,7,8,9,11,12,13,14)
height.in.m + test
```

```
## Hannah   Leon  Lukas Leonie   Luka    Lea   Lena    Mia    Tim   Fynn   Anna
##   2.54   3.67   4.87   5.65   6.90   7.76   8.67   9.56  10.54  12.65  13.67
##  Emily  Felix
##  14.71  15.54
```

Exercise vectors

Data collection ceramics:

An excavation produced the following numbers of flint artefacts:

flakes	blades	cores	debris
506	104	30	267

Assign the values to a named vector, calculate the proportion of the artefacts and sort the vector according to their percentage

During the data collection on box with artefacts was missing, the following numbers has to be added to the vector:

flakes	blades	cores	debris
52	24	15	83

Moreover were 10 items each artefact type missing. Make a vector for the box, add it and the 10 missing to the original data and repeat the calculations.

Sequences and repeated data

Simple sequence:

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Sequence with start value, end value and step size:

```
seq(1,10,by=2)
```

```
## [1] 1 3 5 7 9
```

```
seq(1,20,length=5)
```

```
## [1]  1.00  5.75 10.50 15.25 20.00
```

Repeated data:

```
rep(1,10)
```

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

Data access by index

Access by position:

```
height[1]
```

```
## Hannah
##    154
```

```
height[5]
```

```
## Luka
##  190
```

```
height[1:3]
```

```
## Hannah   Leon  Lukas
##    154    167    187
```

```
height[-(1:3)]
```

```
## Leonie   Luka    Lea   Lena    Mia    Tim   Fynn   Anna  Emily  Felix
##    165    190    176    167    156    154    165    167    171    154
```

Access by name:

```
height["Hannah"]
```

```
## Hannah
##    154
```

Data entry into vectors

Entry by position:

```
height
```

```
## Hannah   Leon  Lukas Leonie   Luka    Lea   Lena    Mia    Tim   Fynn   Anna
##    154    167    187    165    190    176    167    156    154    165    167
##  Emily  Felix
##    171    154
```

```
height[1] <- 168
height
```

```
## Hannah   Leon  Lukas Leonie   Luka    Lea   Lena    Mia    Tim   Fynn   Anna
##    168    167    187    165    190    176    167    156    154    165    167
##  Emily  Felix
##    171    154
```

Entry by name:

```
height["Tim"] <- 181
height
```

```
## Hannah   Leon  Lukas Leonie   Luka    Lea   Lena    Mia    Tim   Fynn   Anna
##    168    167    187    165    190    176    167    156    181    165    167
##  Emily  Felix
##    171    154
```

Logical values

true/false-values:

```
pi>4
```

```
## [1] FALSE
```

```
height > 175
```

```
##  Hannah    Leon   Lukas  Leonie    Luka     Lea    Lena     Mia     Tim    Fynn 
##   FALSE   FALSE    TRUE   FALSE    TRUE    TRUE   FALSE   FALSE    TRUE   FALSE 
##    Anna   Emily   Felix 
##   FALSE   FALSE   FALSE
```

Logical values

Can be used for selection of values:

```
height[height>175]
```

```
## Lukas  Luka   Lea   Tim
##   187   190   176   181
```

```
which(height>175)
```

```
## Lukas  Luka   Lea   Tim
##     3     5     6     9
```

```
sum(height>175)/length(height)
```

```
## [1] 0.3076923
```

Factors

For encoding nominal values:

```
sex <- factor(c("f", "m", "m", "f", "m", "f", "f", "f", "m", "m", "f", "f", "m
```

```
sex
```

```
##  [1] f m m f m f f f m m f f m m
## Levels: f m
```

missing (NA) values

Problem: values are missing

```
height["Martin"] <- 0

mean(height)
```

```
## [1] 158.1429
```

```
sum(height)/13
```

```
## [1] 170.3077
```

therefore: code as N(ot)A(vailable)

```
height["Martin"] <- NA

mean(height)
```

```
## [1] NA
```

```
mean(height, na.rm=T)
```

```
## [1] 170.3077
```

matrices [1]

Data of the same kind (numbers, factors...)

```
load("kursmatrix.RData")
kursmatrix
```

```
##       [,1] [,2]
##  [1,]  168   24
##  [2,]  167   18
##  [3,]  187   20
##  [4,]  165   24
##  [5,]  190   23
##  [6,]  176   24
##  [7,]  167   25
##  [8,]  156   25
##  [9,]  181   21
## [10,]  165   23
## [11,]  167   22
## [12,]  171   22
## [13,]  154   19
## [14,]   NA   NA
```

```
rownames(kursmatrix) <- names(height
colnames(kursmatrix)<-c("height","ag
kursmatrix
```

```
##         height age
## Hannah     168  24
## Leon       167  18
## Lukas      187  20
## Leonie     165  24
## Luka       190  23
## Lea        176  24
## Lena       167  25
## Mia        156  25
## Tim        181  21
## Fynn       165  23
## Anna       167  22
## Emily      171  22
## Felix      154  19
## Martin      NA  NA
```

matrices [2]

Operations on matrices

```
kursmatrix / 100
```

```
##         height  age
## Hannah    1.68 0.24
## Leon      1.67 0.18
## Lukas     1.87 0.20
## Leonie    1.65 0.24
## Luka      1.90 0.23
## Lea       1.76 0.24
## Lena      1.67 0.25
## Mia       1.56 0.25
## Tim       1.81 0.21
## Fynn      1.65 0.23
## Anna      1.67 0.22
## Emily     1.71 0.22
## Felix     1.54 0.19
## Martin      NA   NA
```

```
kursmatrix[, 1] / 100
```

```
## Hannah   Leon  Lukas Leonie   Luka    Lea
##   1.68   1.67   1.87   1.65   1.90   1.76
##  Emily  Felix Martin
##   1.71   1.54     NA
```

```
kursmatrix / c(1:14, rep(2, 14))
```

```
##           height  age
## Hannah 168.00000 12.0
## Leon    83.50000  9.0
## Lukas   62.33333 10.0
## Leonie  41.25000 12.0
## Luka    38.00000 11.5
## Lea     29.33333 12.0
## Lena    23.85714 12.5
## Mia     19.50000 12.5
## Tim     20.11111 10.5
```

Data frames [1]

```
kursdata <-
  data.frame(age = kursmatrix[,2],
             height = kursmatrix[,1]
             sex=sex)
kursdata
```

```
##         age height sex
## Hannah   24    168   f
## Leon     18    167   m
## Lukas    20    187   m
## Leonie   24    165   f
## Luka     23    190   m
## Lea      24    176   f
## Lena     25    167   f
## Mia      25    156   f
## Tim      21    181   m
## Fynn     23    165   m
## Anna     22    167   f
## Emily    22    171   f
## Felix    19    154   m
## Martin   NA     NA   m
```

```
kursdata[,"age"]
```

```
##  [1] 24 18 20 24 23 24 25 25 21 23 22 22
```

```
kursdata$age
```

```
##  [1] 24 18 20 24 23 24 25 25 21 23 22 22
```

Data frames [2]

Operation on data frames

```
kursdata$height / 100
```

```
##  [1] 1.68 1.67 1.87 1.65 1.90 1.76 1.67 1.56 1.81 1.65 1.67 1.71 1.54   NA
```

```
summary(kursdata)
```

```
##       age             height      sex
##  Min.   :18.00   Min.   :154.0   f:7
##  1st Qu.:21.00   1st Qu.:165.0   m:7
##  Median :23.00   Median :167.0
##  Mean   :22.31   Mean   :170.3
##  3rd Qu.:24.00   3rd Qu.:176.0
##  Max.   :25.00   Max.   :190.0
##  NA's   :1       NA's   :1
```

```
tapply(kursdata$height, kursdata$sex, mean, na.rm=T)
```

```
##        f        m
## 167.1429 174.0000
```

Build in datasets

```
data()
```

```
Data sets in package 'datasets':

AirPassengers            Monthly Airline Passenger Numbers 1949-1960
BJsales                  Sales Data with Leading Indicator
BJsales.lead (BJsales)
                         Sales Data with Leading Indicator
BOD                      Biochemical Oxygen Demand
CO2                      Carbon Dioxide Uptake in Grass Plants
ChickWeight              Weight versus age of chicks on different diets
DNase                    Elisa assay of DNase
EuStockMarkets           Daily Closing Prices of Major European Stock
                         Indices, 1991-1998
Formaldehyde             Determination of Formaldehyde
HairEyeColor             Hair and Eye Color of Statistics Students
Harman23.cor             Harman Example 2.3
Harman74.cor             Harman Example 7.4
Indometh                 Pharmacokinetics of Indomethacin
InsectSprays             Effectiveness of Insect Sprays
JohnsonJohnson           Quarterly Earnings per Johnson & Johnson Share
LakeHuron                Level of Lake Huron 1875-1972
```

Data export through save

Simple text file:

```
write(kursmatrix,"kursmatrix.txt")
```

Data frame as simple text file:

```
write.table(kursdata,"kursdata.txt")
```

Data frame as csv file:

```
write.csv2(kursdata,"kursdata.csv")
```

Attention: decimal separator is . not ,

```
kursdata$height <- kursdata$height/100

write.csv(kursdata,"kursdata.csv")
```

problems with importing such csv into e.g. Excel therefore:

```
write.csv2(kursdata,"kursdata.csv")
```

Data import through reading of files

remember:

```
getwd()
setwd("my/location/of/my/working/directory")
```

Simple text file:

```
kursmatrix.loaded <- matrix(scan("kursmatrix.txt"),ncol=2)
```

Data frame as simple text file:

```
kursdata.loaded <- read.table("kursdata.txt")
```

Data frame as csv file:

```
kursdata.loaded <- read.csv2("kursdata.csv")
```

Read with rownames

```
kursdaten.loaded <- read.csv2("kursdaten.csv",row.names = 1)
```

R <-> Excel

Always save as csv

There are packages for R to read and write Excel files but for them additional software (Perl, Python e.a.) is neccessary