Basic statistic techniques for (archaeological) data analysis in R

02_introduction_in_r

First steps, data entry, data access, read and write

Start R

Start of the system:
After R is started, you end on the prompt. Poss. a workspace saved before will be loaded.

```
>
```

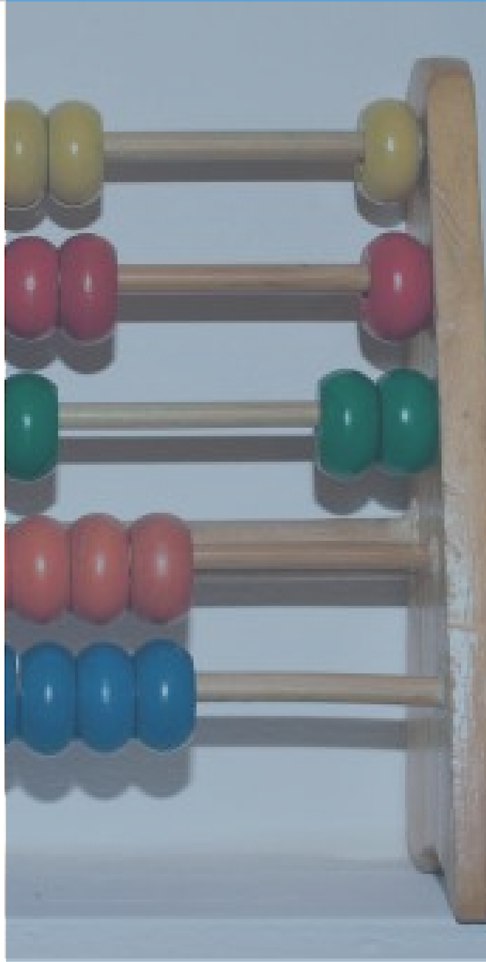
Change the working directory:

```
> getwd()
[1] "/home/martin" # oder etwas anderes...
> setwd(“U:\R“)
```

Change the path according to your needs

Graphical User Interface:
R-Commander

```
> library(Rcmdr)
```



R as calculator

Simplest way of use:

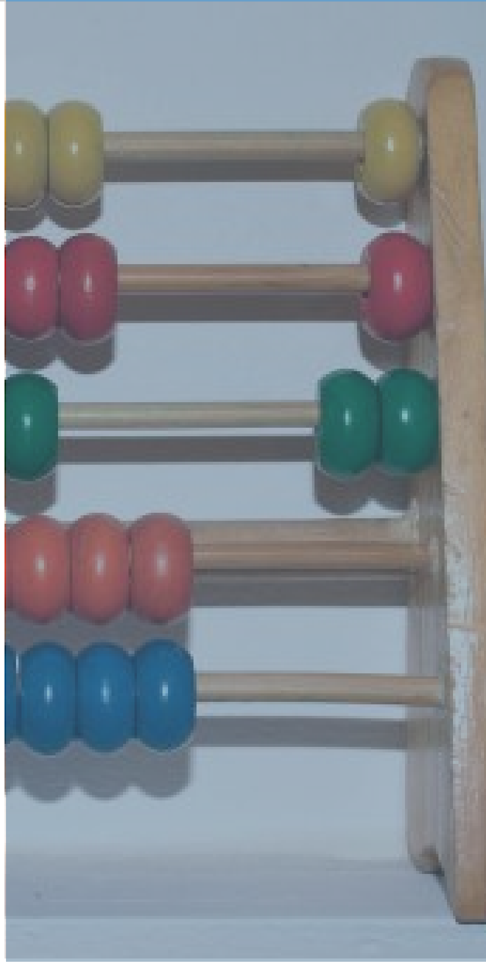
```
> 2+2
[1] 4
> 2^2
[1] 4
```

Multiple commands are separated by ;

```
> (1-2)*3; 1-2*3
[1] -3
[1] -5
```

Using functions:

```
> sqrt(2)         #square root
[1] 1.414214
> log(10)         #logarith base e
[1] 2.303
> log(10, 10)     #logarith base 10, like log(10,
base=10)
[1] 1
```



Getting help

Call of the help function:

```
> help(sqrt)
```

…

Quit with `q`

Even simpler?

```
> ? sqrt
```

Searching the help:

```
> help.search("logarithm")
```

…

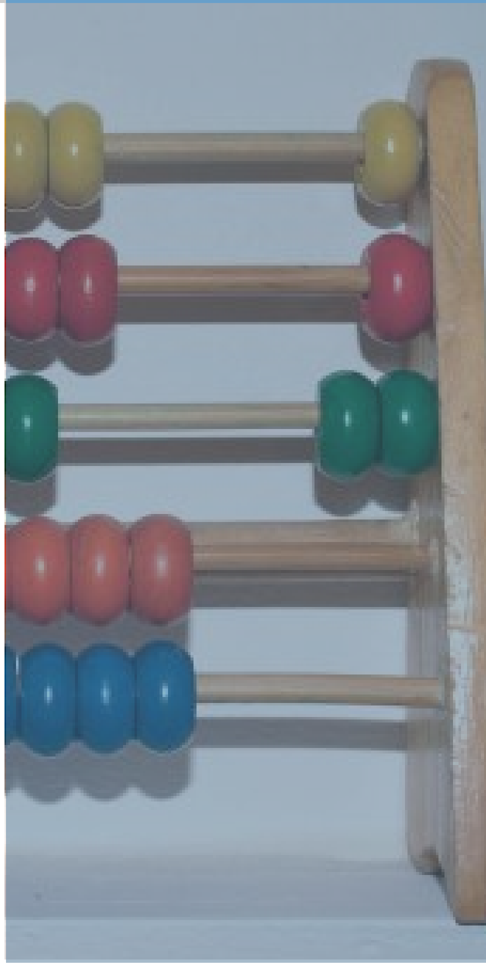
Call the help pages as HTML (Internet pages)

```
> help.start()
```

…

Back to normal help:

```
> options(htmlhelp = FALSE)
```



Christian-Albrechts-Universität zu Kiel

Assignment of data to variables

Naming variables for Values (Assignment):

```
> x<-2       #no message will be given back
> x
[1] 2
> pi         #buildin variable
[1] 3.141593
```

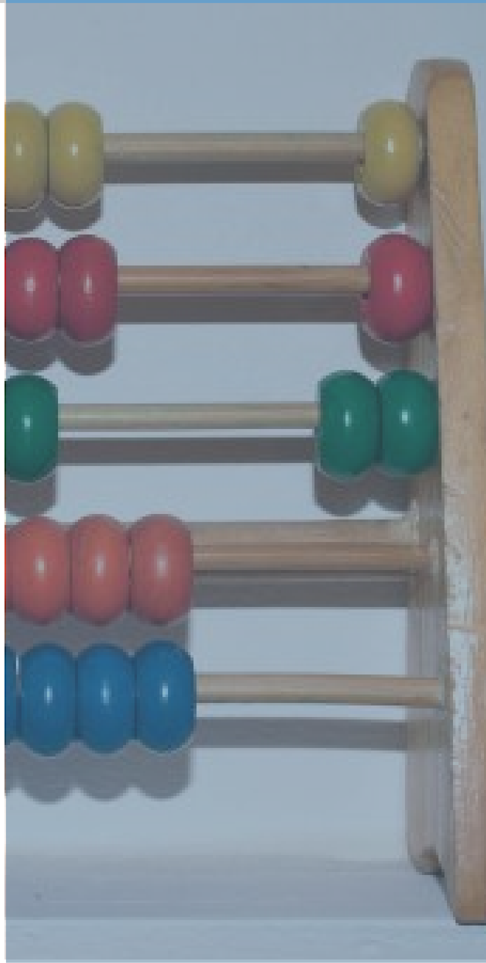
Arrow or equal sign?

Classic assignment in R is the arrow. Also possible:

```
> x=2        #no message will be given back
> x
[1] 2
```

Both is in (newer) versions possible. Matter of tast.

<- is clearer, will be used by me



Work with variables

Display of already uses variables:

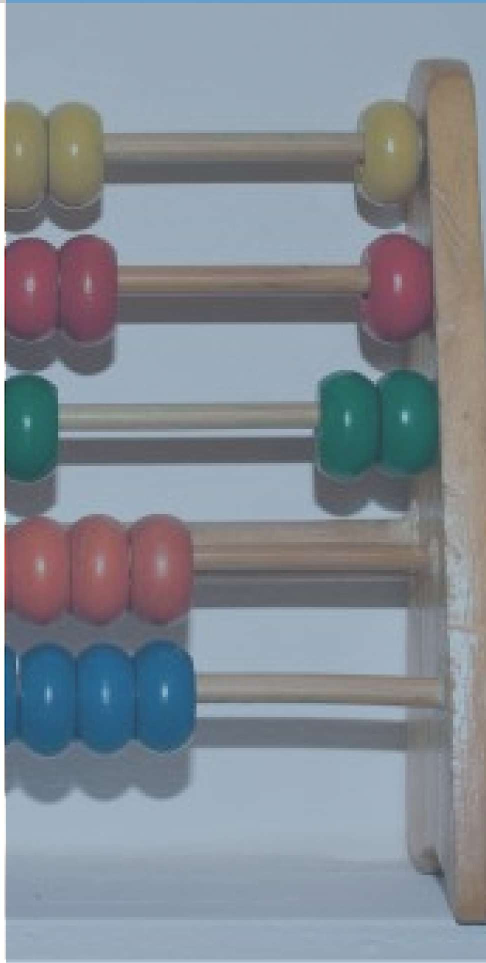
```
> ls()
[1] "x"
```

Delete a variable:

```
> rm(x)      #no message will be given back
> ls()
[1] character(0)
```

Calculations with variables:

```
> x<-2          #no message will be given back
> y<-2*x        #no message will be given back
> z<-sqrt(x)    #no message will be given back
> ls()
[1] "x" "y" "z"
> y
[1] 4
> z
[1] 1.414214
```

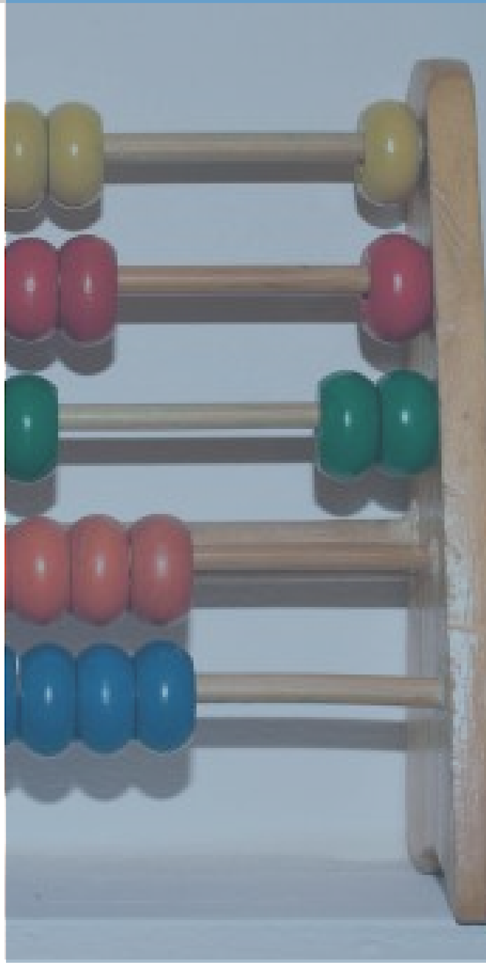


Christian-Albrechts-Universität zu Kiel

Exercise variables

Calculation of a circle:
Given is a circle with the radius *r*=5. Calculate the diameter *d*, the circumference *u* (2πr) and the area *a* (πr²).

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



Basic statistic techniques for (archaeological) data analysis in R

Christian-Albrechts-Universität zu Kiel

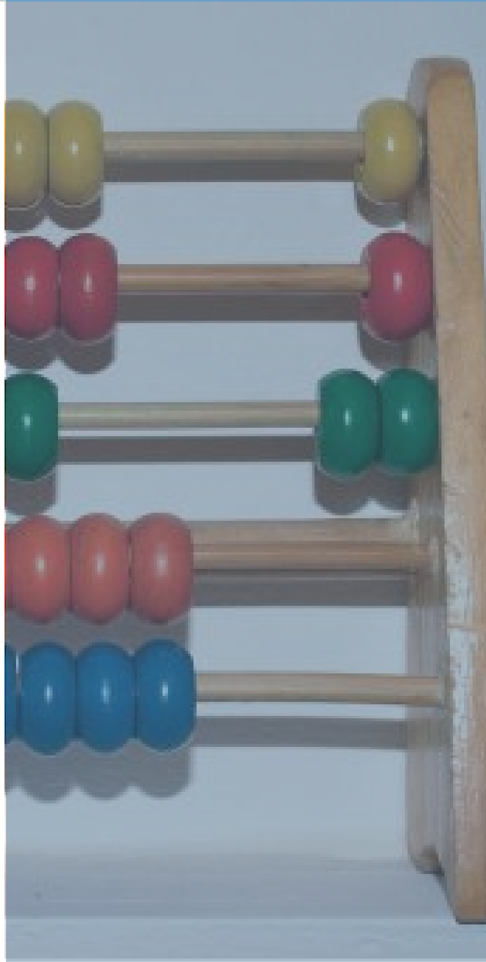
Exercise variables

Calculation of a circle:
Given is a circle with the radius *r*=5. Calculate the diameter *d*, the circumference *u* (2πr) and the area *a* (πr²).

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

Result:

```
> ls()
[1] "d" "r" "v" "x" "y" "z"
> v
[1] 109.9557
>
```



Basic statistic techniques for (archaeological) data analysis in R

Christian-Albrechts-Universität zu Kiel

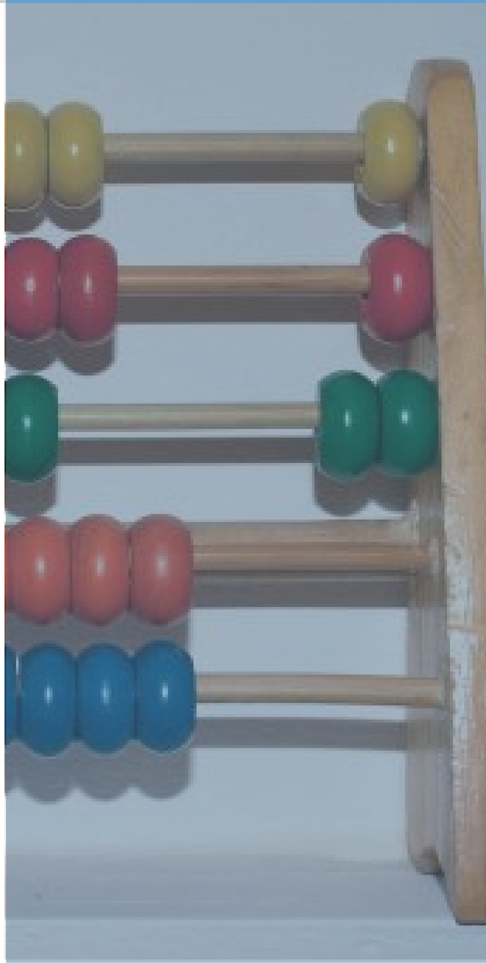
Scalars, vectors, matrices, data frames

Scalar:
A single number or date
```
> pi
[1] 3.141593
```

Vector:
A row of numbers or data
```
> ls()
[1] "d" "r" "v" "x" "y" "z"
```

Matrix:
A table of data of the same kind
```
> euro.cross
…
```

Data frame:
A table of data of different kind
```
> mtcars
...
```



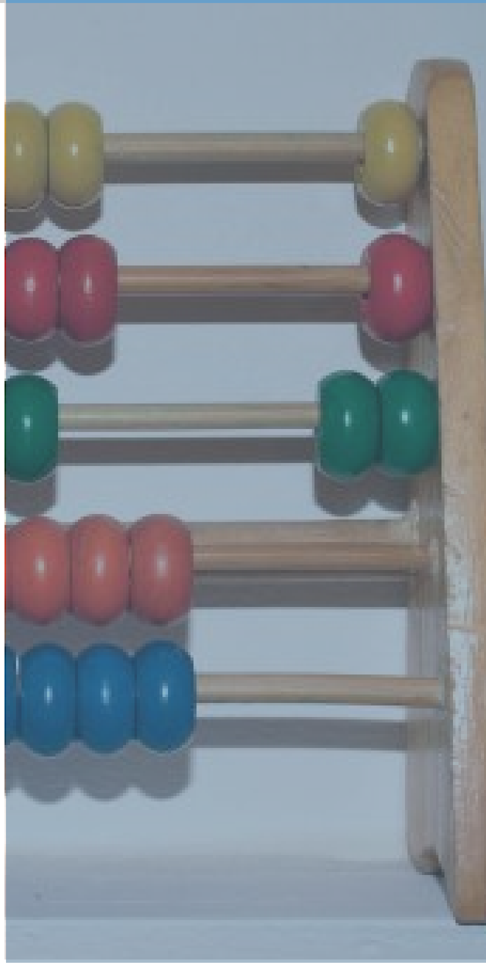
Using c() for data entry

Assignment of values to a vector:

```
> places <- c("Leubingen", "Melz", "Bruszczewo")
> places
[1] "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:

```
participants<-c("Ria", "Anja", "Hannes", "Moritz",
"Basti", "Kay", "Björn", "Cristin", "Martin")
height<-c(174, 163, 182, 175, 173, 198, 179, 163, 181)
names(height)<-participants
```

Sum:

```
> sum(height)
[1] 1588
```

Count:

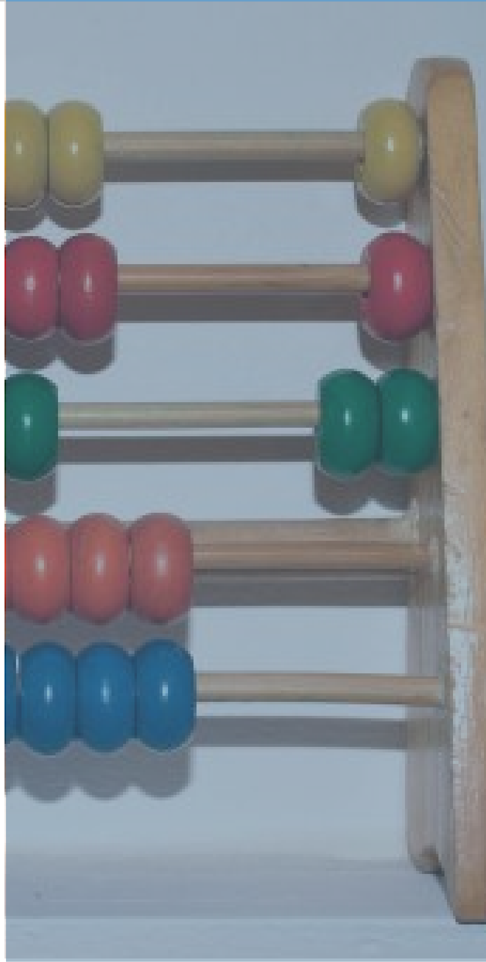
```
> length(height)
[1] 9
```

mean:

```
> sum(height)/length(height)
[1] 176.4444
```

Or more convenient:

```
> mean(height)
[1] 176.4444
```



Basic statistic techniques for (archaeological) data analysis in R

Christian-Albrechts-Universität zu Kiel

Functions on vectors [2]

sort:

```
> sort(height)
   Anja Cristin   Basti     Ria  Moritz   Björn  Martin  Hannes     Kay
    163     163     173     174     175     179     181     182     198
```

minimum:

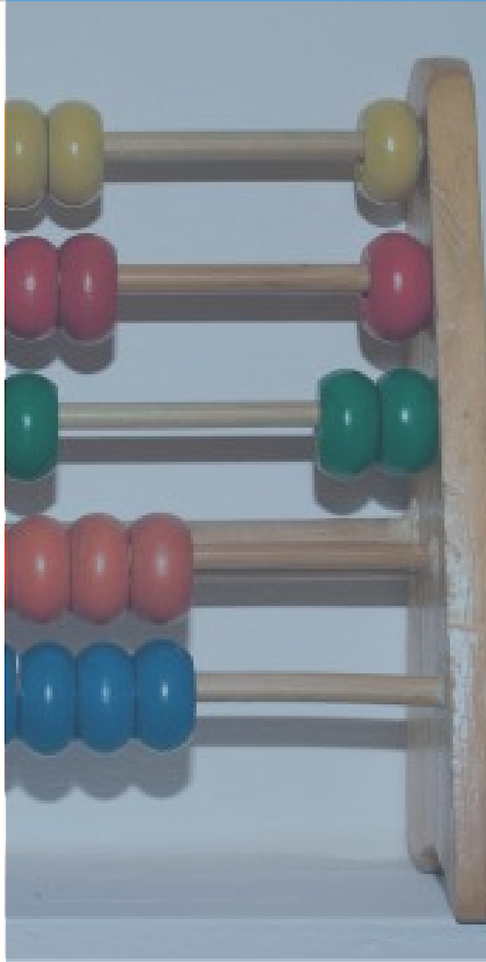
```
> min(height)
[1] 163
```

maximum:

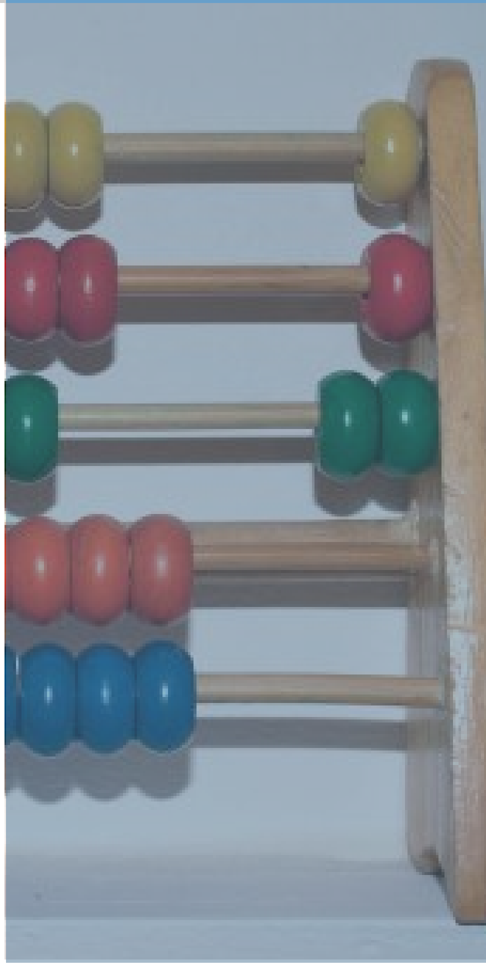
```
> max(height)
[1] 198
```

Or more convenient:

```
> range(height)
[1] 163 198
```



Functions on vectors [3]



Change of the values through calculation:

```
> height.in.m <- height/100
> height.in.m
    Ria    Anja  Hannes  Moritz   Basti     Kay   Björn Cristin  Martin
   1.74    1.63    1.82    1.75    1.73    1.98    1.79    1.63    1.81
```

but:

```
> test<-c(1,2,3,4,5,6,7,8,9)
> height.in.m + test
    Ria    Anja  Hannes  Moritz   Basti     Kay   Björn Cristin  Martin
   2.74    3.63    4.82    5.75    6.73    7.98    8.79    9.63   10.81
```

Basic statistic techniques for (archaeological) data analysis in R

Christian-Albrechts-Universität zu Kiel

Exercise vectors

Data collection ceramics:
An excavation produced the following numbers of flint artefacts:

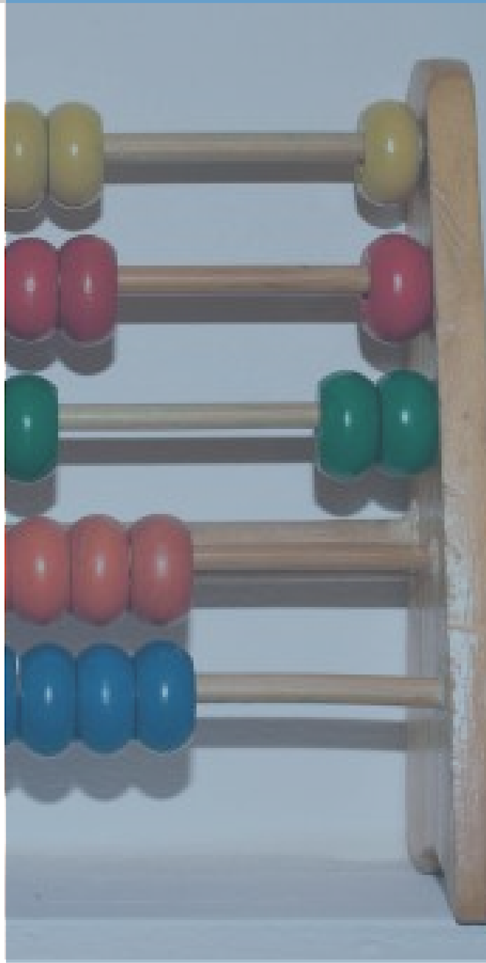
flakes	blades	cores	debris
506	104	30	267

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

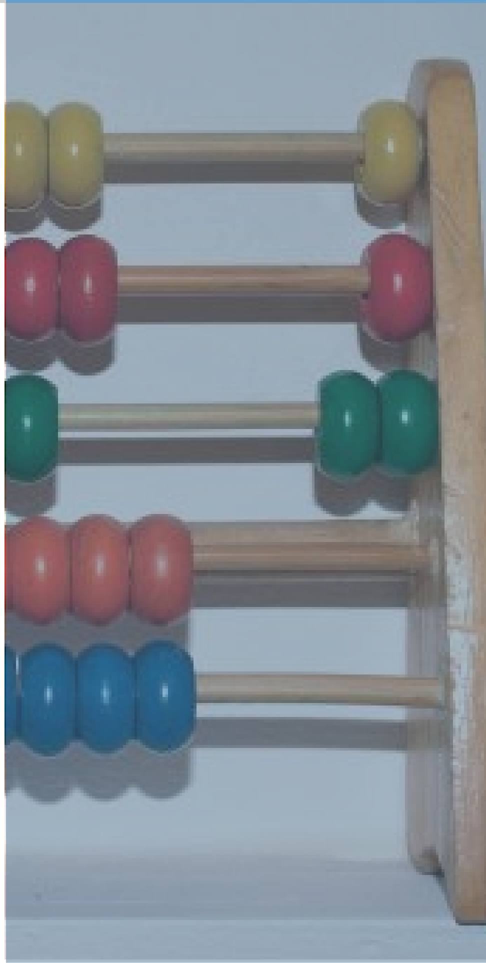
Moreover were 10 items each artefact type missing. Make a vector for the box, add it and the 10 missing to the original data and repeat the calculations.



Exercise vectors

Data collection ceramics:

```
> ww1<-c(506,104,30,267) #enter the values
> names(ww1)<-c("flakes","blades","cores","debris")
#Namen
> ww1.percent<-ww1/sum(ww1) #calculate the proportions
> sort(ww1.percent) #display sorted
     cores     blades     debris     flakes
0.03307607 0.11466373 0.29437707 0.55788313
> ww2<-c(52,24,15,83) #missing box
> ww3<-ww1+ww2 #add the missing box
> ww3<-ww3+10 #add 10 to all values
> ww3.percent<-ww3/sum(ww3) #calculate the proportions
> sort(ww3.percent) #display sorted
     cores     blades     debris     flakes
0.04906334 0.12310437 0.32114184 0.50669045
```



Sequences and repeated data

Simple sequence:

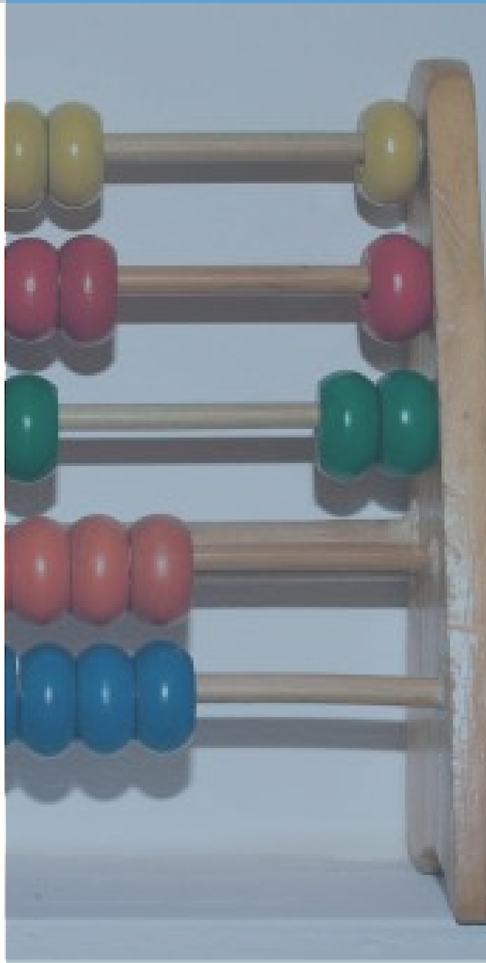
```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

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
> rep(1:3,3)
[1] 1 2 3 1 2 3 1 2 3
> rep(c("Anton","Berta","Claudius"),3)
 [1] "Anton"    "Berta"    "Claudius" "Anton"    "Berta"    "Claudius"
 [7] "Anton"    "Berta"    "Claudius"
```



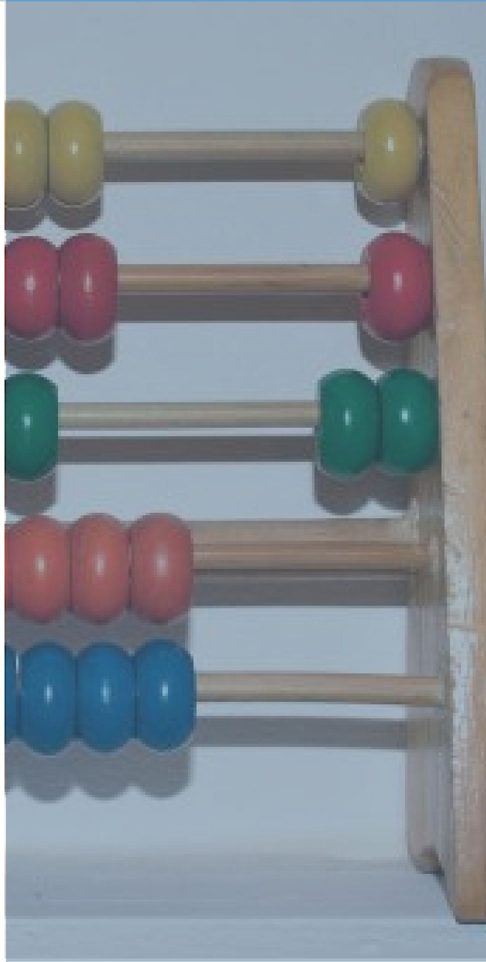
Data access by index

Access by position:

```
> height[1]
Ria
174
> height[5]
Basti
  173
> height[1:3]
   Ria    Anja Hannes
   174     163    182
> height[-(1:3)]
 Moritz    Basti      Kay    Björn Cristin   Martin
    175      173      198      179      163      181
```

Access by name:

```
> height["Kay"]
Kay
198
```



Data entry into vectors

Entry by position:

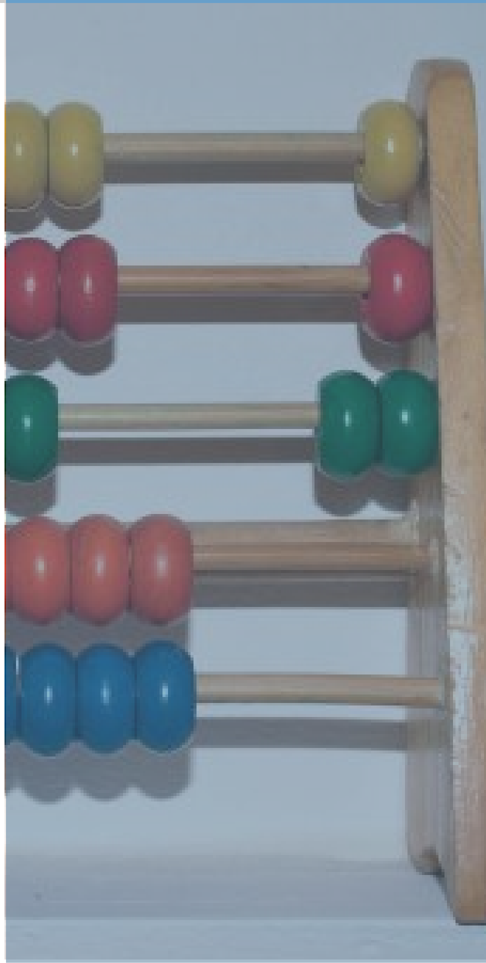
```
> ww1
    flakes      blades        cores      debris
       506         104           30         267
> ww1[1]<-483
> ww1[1]
    flakes
       483
```

Entry by name:

```
> ww1["cores"]<-26
> ww1
    flakes      blades        cores      debris
       483         104           26         267
```

Recycling:

```
> ww1[1:length(ww1)]<-c(30,50)
> ww1
    flakes      blades        cores      debris
        30          50           30          50
```



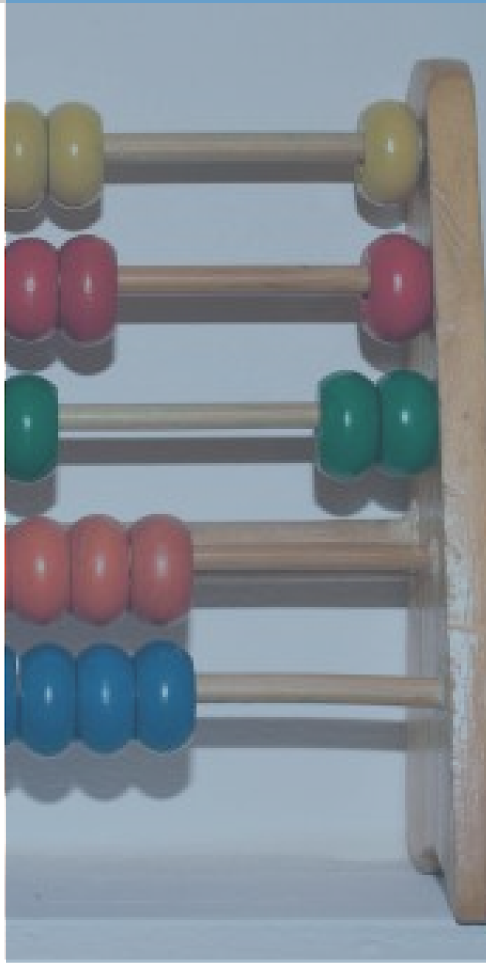
Logical values

true/false-values:

```
> pi>4
[1] FALSE
> height > 175
    Ria    Anja  Hannes  Moritz   Basti     Kay   Björn Cristin  Martin
  FALSE   FALSE    TRUE   FALSE   FALSE    TRUE    TRUE   FALSE    TRUE
```

Could be used for selection of values:

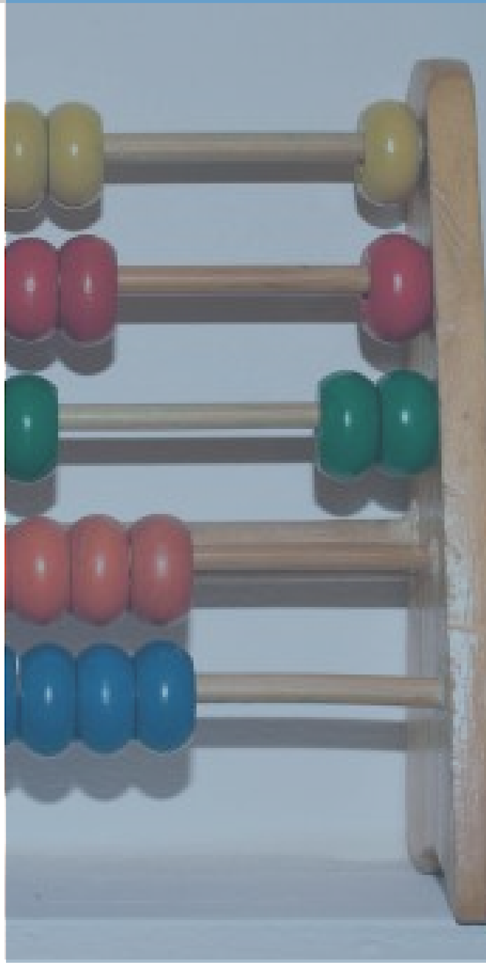
```
> height[height>175]
Hannes    Kay  Björn Martin
   182    198    179    181
> which(height>175)
Hannes    Kay  Björn Martin
     3      6      7      9
> sum(height>175)/length(height)
[1] 0.4444444
```



factors

For encoding nominal values:

```
> sex <- factor(c("f", "f", "m", "m", "m", "m", "m",
"f", "m"))
> sex
[1] f f m m m m m f m
Levels: m f
```



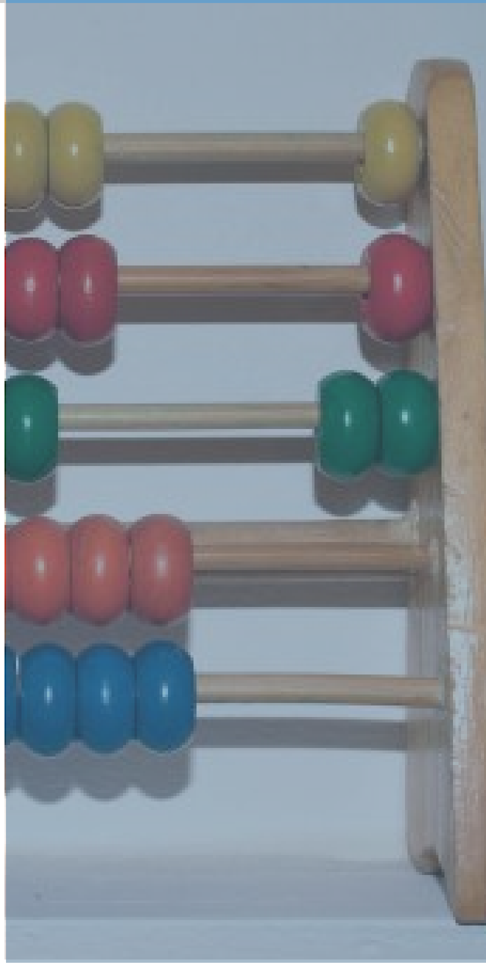
missing (NA) values

Problem: values are missing

```
> age<-c(26,24,25,23,23,24,30,20,0)
> names(age)<-participants
> age
    Ria     Anja  Hannes  Moritz    Basti      Kay   Björn Cristin  Martin
     26       24      25      23       23       24      30      20       0
> mean(age)
[1] 21.66667
> sum(age)/8
[1] 24.375
```

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

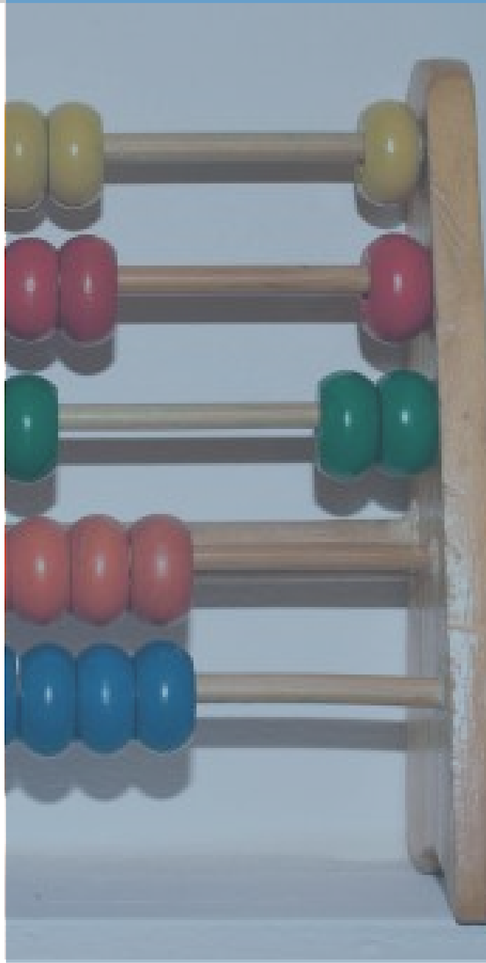
```
> age<-c(26,24,25,23,23,24,30,20,NA)
> names(age)<-participants
> age
> age
[1] 26 24 25 23 23 24 30 20 NA
> mean(age)
[1] NA
> mean(age,na.rm=T)
[1] 24.375
>
```



matrices [1]

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

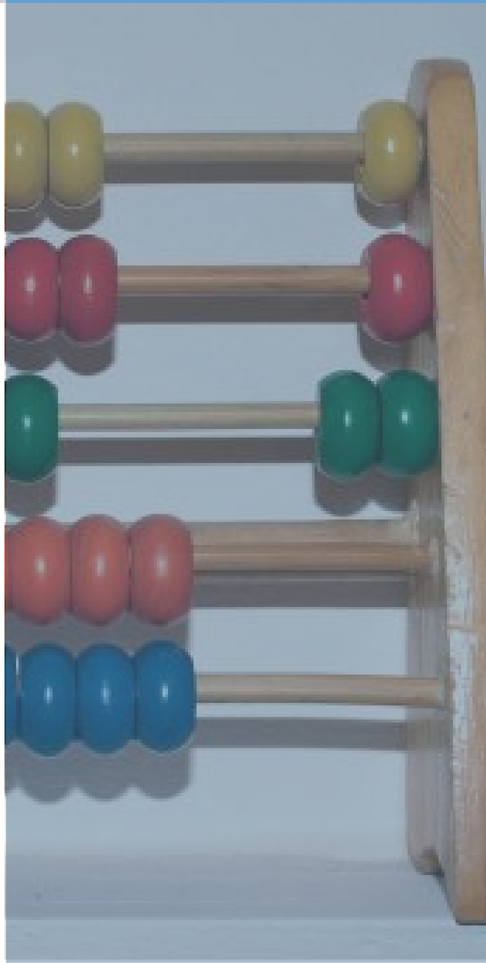
```
> kursmatrix<-matrix(c(height,age),9,2)
> kursmatrix
      [,1] [,2]
 [1,]  174   26
 [2,]  163   24
 [3,]  182   25
 [4,]  175   23
 [5,]  173   23
 [6,]  198   24
 [7,]  179   30
 [8,]  163   20
 [9,]  181   NA
> rownames(kursmatrix)<-participants
> colnames(kursmatrix)<-c("height","age")
> kursmatrix
                height   age
Ria                174    26
Anja               163    24
Hannes             182    25
Moritz             175    23
Basti              173    23
Kay                198    24
Björn              179    30
Cristin            163    20
Martin             181    NA
```



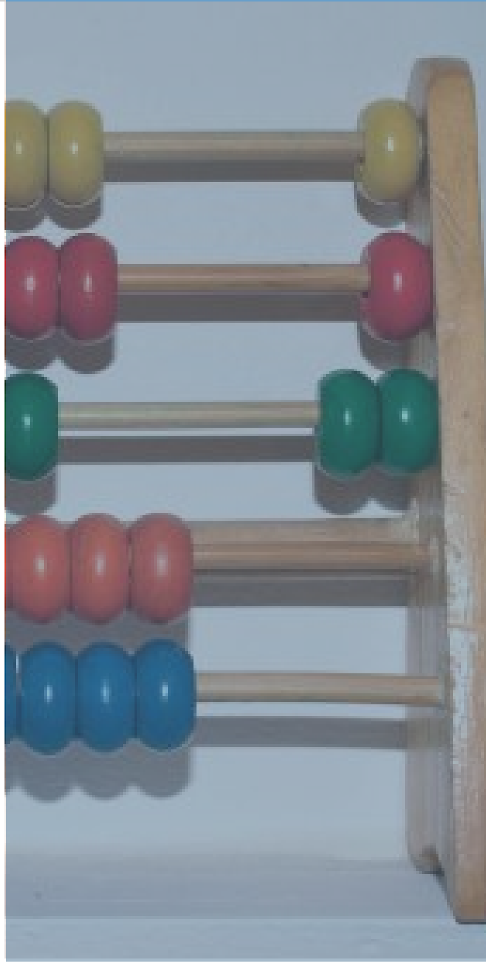
matrices [2]

Functions on matrices

```
> dim(kursmatrix)
[1]  9 2
> length(kursmatrix)
[1]  18
> kursmatrix[3,]
        height            age
           182             25
> kursmatrix[,1]
    Ria     Anja  Hannes  Moritz    Basti      Kay    Björn Cristin  Martin
    174      163     182     175      173      198      179     163     181
> t(kursmatrix)
               Ria  Anja Hannes Moritz Basti Kay Björn Cristin Martin
height         174  163     182    175   173 198   179     163    181
age             26   24      25     23    23  24    30      20     NA
```



matrices [2]



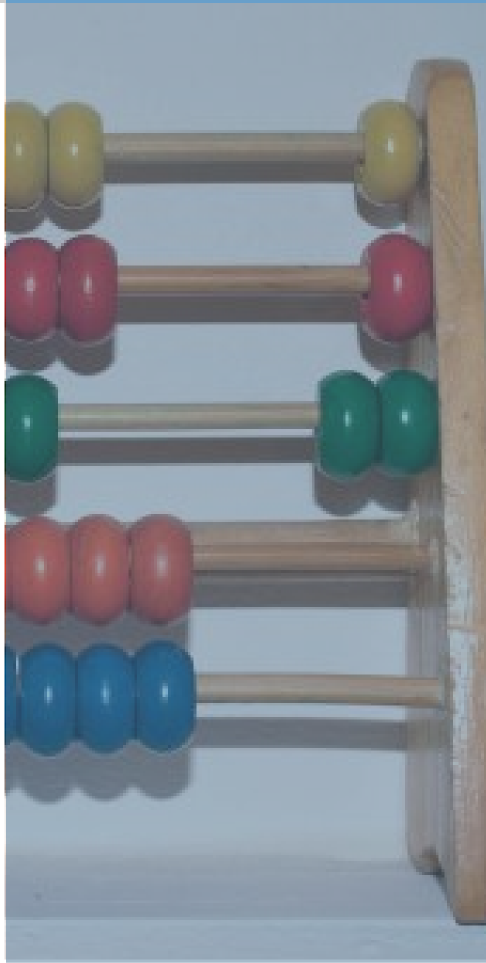
Functions on matrices

```
> kursmatrix/100
                height   age
Ria               1.74  0.26
Anja              1.63  0.24
Hannes            1.82  0.25
Moritz            1.75  0.23
Basti             1.73  0.23
Kay               1.98  0.24
Björn             1.79  0.30
Cristin           1.63  0.20
Martin            1.81    NA
> kursmatrix[,1]/100
    Ria    Anja  Hannes  Moritz   Basti     Kay   Björn Cristin  Martin
   1.74    1.63    1.82    1.75    1.73    1.98    1.79    1.63    1.81
> kursmatrix / c(100,200,300,400,500,1,1,1,1,1,1,1,1,1,1,1,1,1)
                height   age
Ria          1.7400000    26
Anja         0.8150000    24
Hannes       0.6066667    25
Moritz       0.4375000    23
Basti        0.3460000    23
Kay        198.0000000    24
Björn      179.0000000    30
Cristin    163.0000000    20
Martin     181.0000000    NA
```

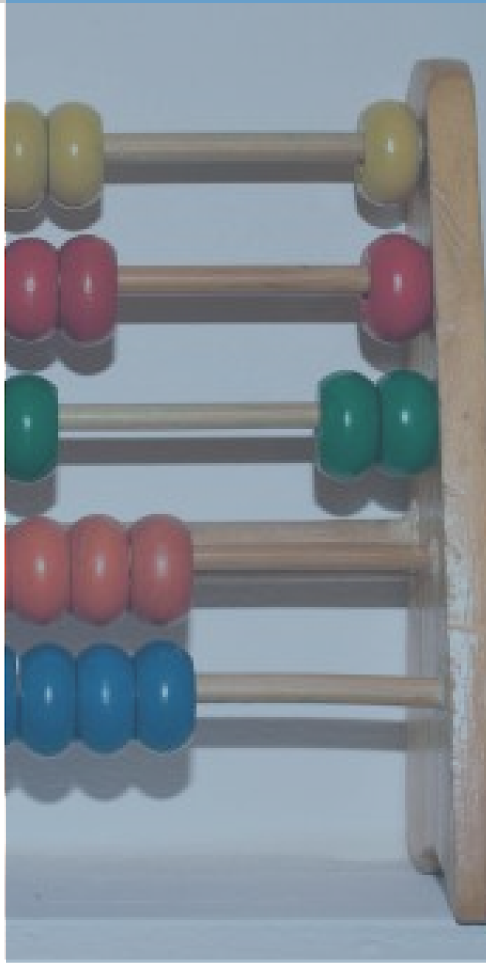
Data frames [1]

Data of different kind (numbers and factors and...):

```
> kursdata<-data.frame(age,height,sex)
> kursdata
          Age          height        sex
Ria        26             174          f
Anja       24             163          f
Hannes     25             182          m
Moritz     23             175          m
Basti      23             173          m
Kay        24             198          m
Björn      30             179          m
Cristin    20             163          f
Martin     NA             181          m

> kursdata[,"age"]
[1] 26 24 25 23 23 24 30 20 NA
> kursdata$age
[1] 26 24 25 23 23 24 30 20 NA
```



Data frames [2]



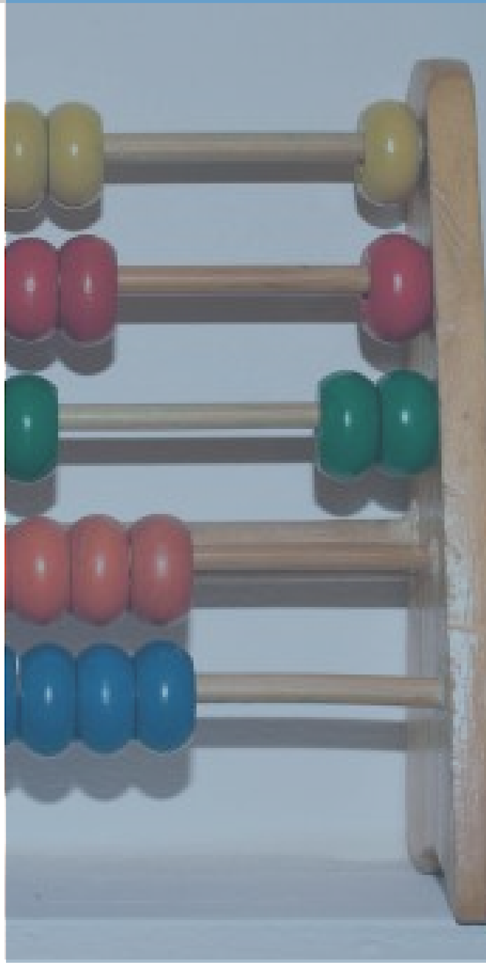
Functions on data frames

```
> kursdaten$height/100
[1] 1.74 1.63 1.82 1.75 1.73 1.98 1.79 1.63 1.81

> summary(kursdaten)
          age             height      sex  
 Min.    :20.00   Min.    :163.0   m:6  
 1st Qu.:23.00   1st Qu.:173.0   f:3  
 Median :24.00   Median :175.0        
 Mean    :24.38   Mean    :176.4        
 3rd Qu.:25.25   3rd Qu.:181.0        
 Max.    :30.00   Max.    :198.0        
 NA's    : 1.00                         

> tapply(kursdaten$age, kursdaten$sex, mean, na.rm="T")
       m        f 
25.00000 23.33333 
```

Buildin datasets

Test data for playing around with:

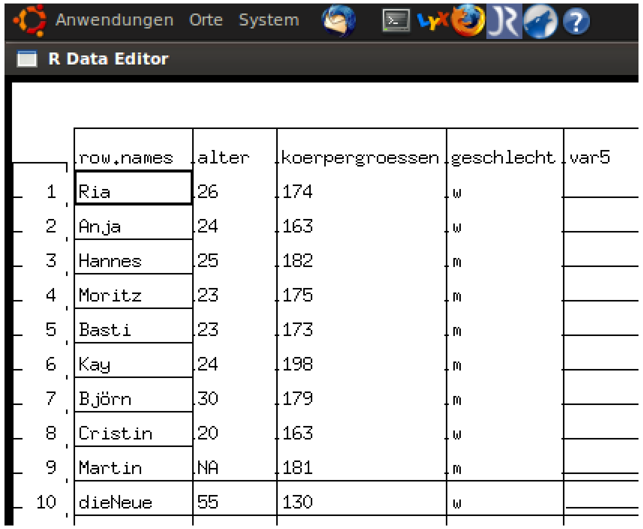
```
> data()
Data sets in package ‘datasets’:

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 Indomethicin
...
```



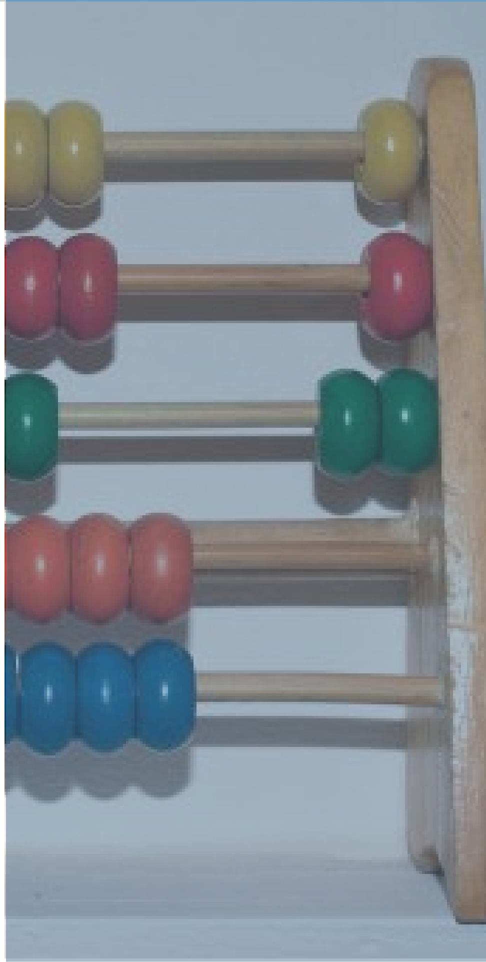
Changing data, the convenient way...

To change data frames:

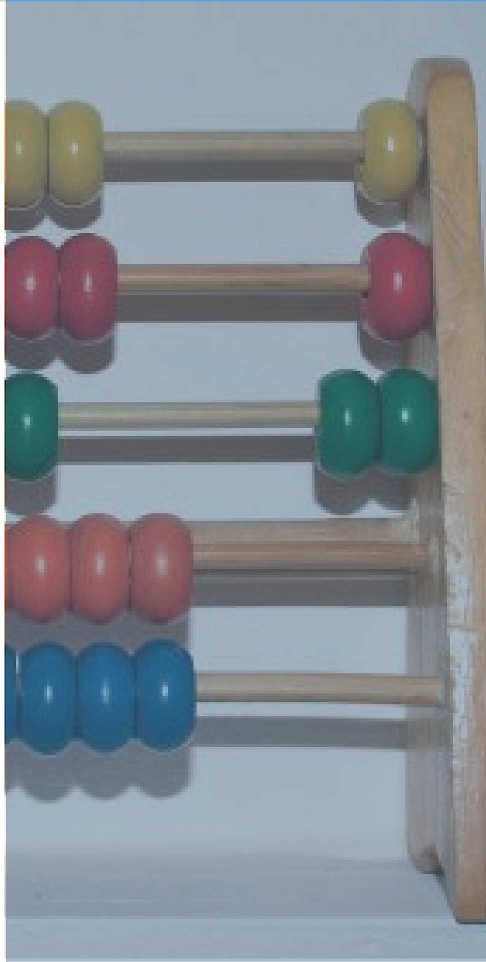
```
>kursdaten.neu<-edit(kursdaten)
> kursdaten.neu
           age          height         sex
Ria         26             174           f
Anja        24             163           f
Hannes      25             182           m
Moritz      23             175           m
Basti       23             173           m
Kay         24             198           m
Björn       30             179           m
Cristin     20             163           f
Martin      NA             181           m
dieNeue     55             130           f
```



```
    row.names  alter  koerpergroessen  geschlecht  var5
1   Ria        26     174              w
2   Anja       24     163              w
3   Hannes     25     182              m
4   Moritz     23     175              m
5   Basti      23     173              m
6   Kay        24     198              m
7   Björn      30     179              m
8   Cristin    20     163              w
9   Martin     NA     181              m
10  dieNeue    55     130              w
```



Data export through save



Simple text file:

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

Data frame as simple text file:

```
> write.table(kursdaten,"kursdaten.txt")
```

Data frame as csv file:

```
> write.csv2(kursdaten,"kursdaten.csv")
```

Attention: decimal separator is . not ,

```
> kursdaten$height<-kursdaten$height/100
> write.csv2(kursdaten,"kursdaten.csv")
```

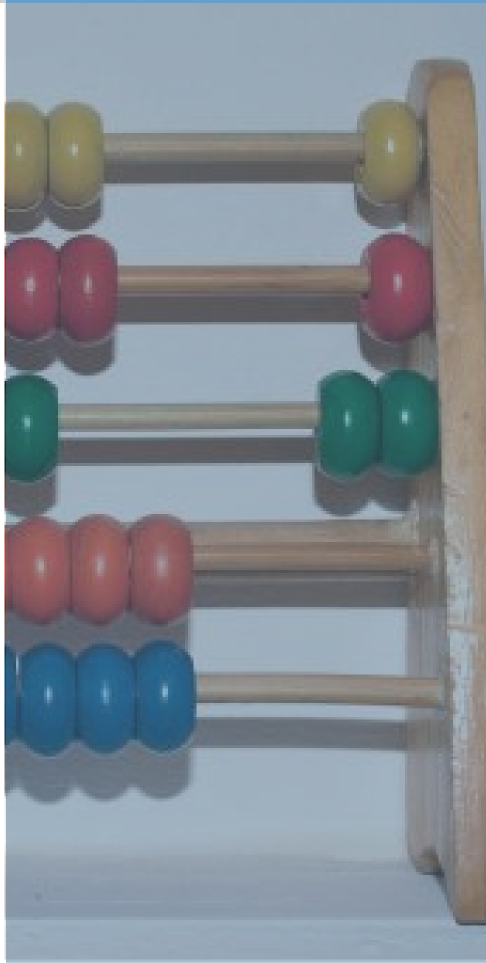
-problems with importing such csv into e.g. Excel-

therefore:

```
> write.csv2(kursdaten,"kursdaten.csv",dec=",")
Warning message:
In write.csv2(kursdaten, "kursdaten.csv", dec = ",") :
  Versuch 'dec' auf ignoriert zu setzen
```

Christian-Albrechts-Universität zu Kiel

Data import through reading of files



remember:

```
> getwd()
[1] "/home/martin" # oder etwas anderes...
> setwd("U:\R") # oder etwas anderes...
```

Simple text file:

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

Data frame as simple text file:

```
> kursdaten.gelesen<-read.table("kursdaten.txt")
```

Data frame as csv file:

```
> kursdaten.gelesen<-read.csv2("kursdaten.csv")
```

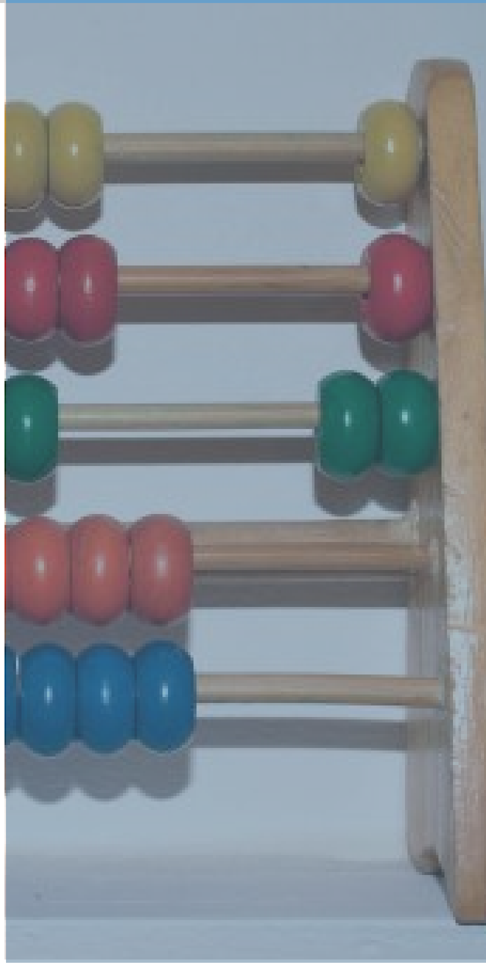
Read with rownames

```
> kursdaten.gelesen<-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



Christian-Albrechts-Universität zu Kiel

Quit R

```
q()
```

Save workspace image? [y/n/c]:

Saves the actual working environment with all variables for the next session, if wanted

y	save
n	not save
c	not quit