Introduction to R

Martin Weber
Awdesch Melzer

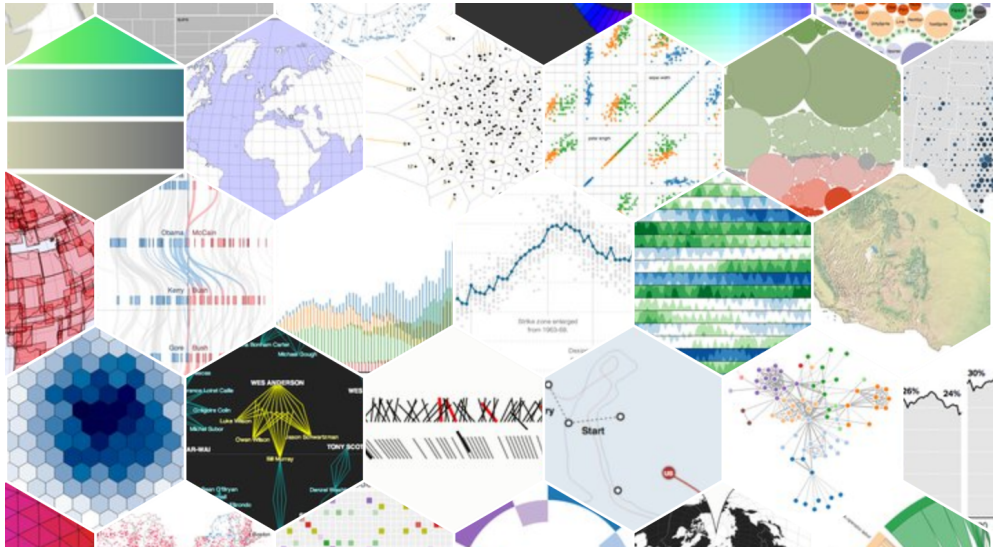
Institute of Finance
Humboldt-Universität zu Berlin
http://wiwi.hu-berlin.de/finanz

What is R?

- S language, developed by R. Becker and J. Chambers in 1984
- GNU implementation of S 1992 by R. Ihaka and R. Gentleman in NZ
- great variety of packages covering all fields of statistics
- non-commercial use mostly for free

What can R do?

- Empirical modelling
- Visualisation
- Simulations
- Interactive analysis
- ...



Arithmetic

```
2 + 3                  # add
[1] 5
4 * 5 / 6              # multiply and divide
[1] 3.333333
7^8                    # 7 to the 8th power
[1] 5764801
sqrt(2)                # square root
[1] 1.414214
exp(1)                 # Euler's constant
[1] 2.718282
pi
[1] 3.141593
5 %/% 2                # 2; integer division
5 %% 2                 # 1; modulo division
```


Assignments

```
x <- 7*41/pi      # R2L assignment
x                 # show entries of x
[1] 91.35494

x = 7*41/pi       # equality assignment (R2L)
x
[1] 91.35494

7*41/pi -> x      # L2R
x
[1] 91.35494
```


Object types

```
1  a <- 3; a                  # integer
2  [1] 3
3
4  b <- pi; b                 # double
5  [1] 3.141593
6
7  c <- "character"; c # alternative with '
8  [1] "character"
9
10 d <- (a < 10); d           # logical
11 [1] TRUE
```


Logical functions

```
1  <     # smaller
2  <=    # smaller or equal
3  >     # bigger
4  >=    # bigger or equal
5  !=    # unequal
6  ==    # logical equal
7  !     # logical NOT (unary)
8  &     # logical AND (vector)
9  |     # logical OR (vector)
10 &&    # logical AND (no vector)
11 ||    # logical  OR (no vector)
```


Vectors

```
1 a = 1:3
2 b = 2:4
3 c(a,b)          # [1] 1 2 3 2 3 4
4 c(1,1:3)        # [1] 1 1 2 3
5 array(1,4)      # [1] 1 1 1 1; array(input, dim)
```


More vectors

```
1 seq(1,3)                          # [1] 1 2 3
2 seq(3)                            # [1] 1 2 3
3 seq(1,2,by=0.1)                   # [1] 1.1 1.2 1.3 1.4 1.5
4 seq(1,3,0.5)                      # [1] 1.0 1.5 2.0 2.5 3
5 seq(1,3,length.out = 4)           # [1] 1.00 1.67 2.33 3.00
6 rep(1:4,2)                        # [1] 1 2 3 4 1 2 3 4
7 rep(1:4,each = 2)                 # [1] 1 1 2 2 3 3 4 4
8 rep(c(7,9,3), 1:3)                # [1] 7 9 9 3 3 3
```


More vectors

```
1  a <- c(2,3,1,4)  # double vector [1] 2 3 1 4
2  length(a)        # [1] 4
3  rev(a)           # [1] 4 1
4  a[<i>]           # returns
5  a[1:2]           # [1] 2 3
6  a[-1]            # [1] 3 1 4
7  a[-c(1,2)]       # [1]
8  a[a < 3]         # [1] 1 2
9  which(a == 3)    # [1] 2
10 a > 1            # [1] TRUE TRUE FALSE TRUE
```


More vectors

```
a <- letters [1:3]; a
[1] "a" "b" "c"

b <- LETTERS [1:3]; b
[1] "A" "B" "C"

c <- month.abb[1:6]; c
[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun"

d <- month.name[1:12]; d
[1] "January"   "February"  "March"    ...
```


More vectors

```
1  a <- c(1,2,3,4) # double vector [1] 1 2 3 4
2  t(a) # returns d as row vector (transposes d), but
     is already a matrix
3        [,1] [,2] [,3] [,4]
4  [1,]    1    2    3    4
5  t(t(a)) # column vector, is also matrix
6             [ ,1]
7       [1,]    1
8       [2,]    2
9       [3,]    3
10      [4,]    4
```


Matrices

```
1  matrix(1:12, nrow=3)
2       [,1] [,2] [,3] [,4]
3  [1,]    1    4    7   10
4  [2,]    2    5    8   11
5  [3,]    3    6    9   12
6  matrix(1:12, nrow=3, ncol=4, byrow = T)
7       [,1] [,2] [,3] [,4]
8  [1,]    1    2    3    4
9  [2,]    5    6    7    8
10 [3,]    9   10   11   12
11 diag(1, nrow=2, ncol=2) # diagonal matrix
12      [,1] [,2]
13 [1,]    1    0
14 [2,]    0    1
```


Merging vectors to matrices

```
x = 1:3
y = 4:6
rbind(x,y)
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6
cbind(x,y)
     x y
[1,] 1 4
[2,] 2 5
[3,] 3 6
```


Matrices: Size

```
x <- matrix(1:10, 2, 5)
dim(x)              # size of matrix x
col(x)              # column indices of ALL elements
row(x)              # row indices of ALL elements
x[<i>,<j>]          # extract i-th row and j-th column
x[row(x) == col(x)] # extract the diagonal
```


Sums and products

```
x = matrix (1:20 ,4 ,5)
sum(x)
[1] 210
prod (x)
[1] 2.432902e+18
colSums(x)
[1] 10 26 42 58 74
rowSums(x)
[1] 45 50 55 60
rowMeans(x)
[1] 9 10 11 12
colMeans(x)
[1] 2.5 6.5 10.5 14.5 18.5
```


Loops and conditions: FOR

```
1 for (i in 1:4){ print(i) }
2
3 for (i in letters[1:4]){ print(i) }
4
5 a <- numeric(400) # generate empty a of length 400
6 for (i in 1:400){ a[i]=i } # fill a with 1:400
7 # takes much longer than a <- 1:400
```


Loops and conditions: WHILE

```
1 i <- 0
2 while(i<4){
3     i <- i+1
4     print(i)
5 }
```


Loops and conditions: REPEAT

```
1 i <- 0;
2 repeat{
3     i <- i+1;
4     print(i);
5     if (i==4) break
6 }
```

If no break is given, loops runs forever!

Loops and conditions: IFELSE

`ifelse(boolean check, if-case, else-case)`

```
1 x <- c(6:-4)
2 sqrt(x)                          # gives warning
3 sqrt(ifelse(x >= 0, x, NA))      # no warning
```


Functions

```
1 col.means <- function(input){
2     n     = nrow(input)
3     ones  =
4     return((rep(1,n) %*% input)/n)
5 }
6
7 colMeans(matrix(1:12,3,4))
8 col.means(matrix(1:12,3,4))
```


Task

Write a function that calculates the **column means** of a dataset using the `for` **loop**.

Load datasets

```
1 setwd("C:/...")
2 setwd("/Users/...")
3 data <- read.csv("DAX30.csv", sep=",") # data set is
    a CSV-file
```