Introduction to R

Exercise: Business Intelligence (Part 2)
Summer Term 2014
Stefan Feuerriegel

Today's Lecture

Objectives

1. Being able to perform simple calculations in R
2. Understanding the concepts of variables
3. Handling vectors and matrices

Outline

1. General Information
2. Operations, Functions, Variables
3. Vectors
4. Matrices
5. Extensibility
6. Wrap-Up

Outline

1 General Information

2 Operations, Functions, Variables

3 Vectors

4 Matrices

5 Extensibility

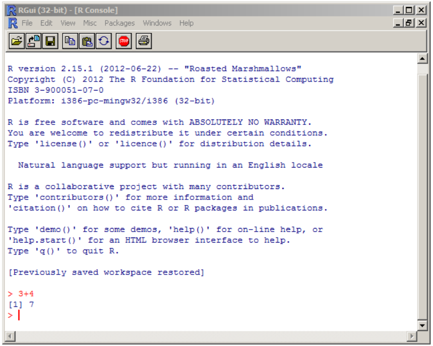
6 Wrap-Up

Examples of Statistical Software

Excel	Limited capabilities for statistics; good for data preprocessing
SPSS	Easy/good for standard procedures
SAS	Good for large data sets and complicated analysis
STATA	Common in research; various estimators and statistical tests
EViews	Strong focus on time series analysis
Matlab	Mathematical programming, but statistical methods limited

What is R?

- Free software environment aimed at statistical computing
- Supports many operating systems (Linux, Mac OS X, Windows)
- Very frequently used in psychology, bioinformatics, statistics, econometrics, and machine learning



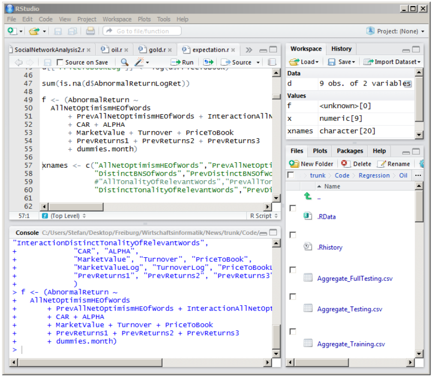
```
R version 2.15.1 (2012-06-22) -- "Roasted Marshmallows"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> 3+4
[1] 7
> 
```

Retrieving R

Download at `http://www.r-project.org`

R Studio as Editor

- Instead of typing commands into the R Console, you can generate commands by an editor and then send them to the R window
- . . . and later modify (correct) them and send again



```
sum(is.na(d$AbnormalReturnLogRet))

f <- (AbnormalReturn ~
  AllNetOptimismHEOfWords
      + PrevAllNetOptimismHEOfWords + InteractionAllN
      + CAR + ALPHA
      + MarketValue + Turnover + PriceToBook
      + PrevReturns1 + PrevReturns2 + PrevReturns3
      + dummies.month)

xnames <- c("AllNetOptimismHEOfWords","PrevAllNetOpti
            "DistinctBNSOfWords","PrevDistinctBNSOfWc
            #"AllTonalityOfRelevantWords","PrevAllTor
            "DistinctTonalityOfRelevantWords","PrevDi
```

Console C:/Users/Stefan/Desktop/Freiburg/Wirtschaftsinformatik/News/trunk/Code/

```
"InteractionDistinctTonalityOfRelevantWords",
+               "CAR", "ALPHA",
+               "MarketValue", "Turnover", "PriceToBook",
+               "MarketValueLog", "TurnoverLog", "PriceToBookL
+               "PrevReturns1", "PrevReturns2", "PrevReturns3'
+               )
> f <- (AbnormalReturn ~
+   AllNetOptimismHEOfWords
+       + PrevAllNetOptimismHEOfWords + InteractionAllNetOpt
+       + CAR + ALPHA
+       + MarketValue + Turnover + PriceToBook
+       + PrevReturns1 + PrevReturns2 + PrevReturns3
+       + dummies.month)
> 
```

Data	
d	9 obs. of 2 variables
Values	
f	<unknown>[0]
x	numeric[9]
xnames	character[20]

Retrieving R Studio (recommended)

Download at `http://www.rstudio.com/`

Outline

1 General Information

2 Operations, Functions, Variables

3 Vectors

4 Matrices

5 Extensibility

6 Wrap-Up

First Example

→ Live Demonstration

```
3 * (4 + 2)

## [1] 18
```

Arithmetic Operations

```
1 + 2 * 3

## [1] 7

3/4 + 2

## [1] 2.75

2 * pi - pi

## [1] 3.142

0/0

## [1] NaN
```

Operation	**Description**	**Example**	**Result**
+	Plus	`3+4`	7
−	Minus	`3-4`	-1
*	Times	`3*4`	12
/	Divide	`3/4`	0.75
^	Exponentiation	`3^4`	$3^4 = 81$

Logic Operators

Comparison Operators

Operators `<`, `<=`, `==`, `!=`, `>=`, `>` return boolean values `TRUE` or `FALSE`

```
3 < 4

## [1] TRUE

3 > 4

## [1] FALSE

3 <= 4

## [1] TRUE
```

```
4 == 4

## [1] TRUE

3 != 4

## [1] TRUE
```

Brackets, Comments and Decimal Points

- Brackets can be used to prioritize evaluations

```
3 * (4 + 2)

## [1] 18
```

- Important to use a point instead of a comma!

```
3.141

## [1] 3.141
```

- Comments via #

```
3 + 4  # will be ignored

## [1] 7
```

Mathematical Functions

- Square root

```r
sqrt(1 + 1)

## [1] 1.414
```

- Logarithm to the base 10

```r
log10(10 * 10 * 10)

## [1] 3
```

- Sinus function and rounding

```r
sin(pi)   # rarely exact: R uses limited number of digits

## [1] 1.225e-16

round(sin(pi))

## [1] 0
```

Mathematical Functions

Function	Description	Example	Result
`abs()`	Absolute Value	`3-4`	$+1$
`round()`	Rounding	`round(3.14)`	≈ 3
`sqrt()`	Square Root	`sqrt(81)`	$\sqrt{81} = 9$
`sin()`	Sine	`sin(0)`	$\sin 0 = 0$
`cos()`	Cosine	`cos(0)`	$\cos 0 = 1$
`tan()`	Tangent	`tan(0)`	$\tan 0 = 0$
`log()`	Natural Logarithm	`log(e)`	$\ln e = 1$
`log10()`	Common Logarithm	`log10(100)`	$\log_{10} 100 = 2$

Exercise: Mathematical Functions

Question

- What is the value of `abs(3-4*5)`?
- Visit `http://pingo.upb.de` with code 1523

Variables

```
x <- 2
x

## [1] 2

x + 3

## [1] 5

x

## [1] 2

x <- x + 4
x

## [1] 6
```

- Variables store values during a session
- Value on right is assigned to variable preceding "<−"
- No default output after assignment
- Recommended names consist of letters A–Z plus "_" and "."
- Must not contain minus!
  - Should be different from function names, e. g. `sin`
  - Good: `x`, `fit`, `ratio`, etc.
- Warning: naming is case-sensitive
  - i. e. `x` and `X` are different

Exercise: Variables

Question

- What is the value of `z`?
- Visit `http://pingo.upb.de` with code 1523

```r
x <- 2
x <- x + 1
y <- 4
z <- x + y
x <- x + 1
z <- z + x
```

Strings

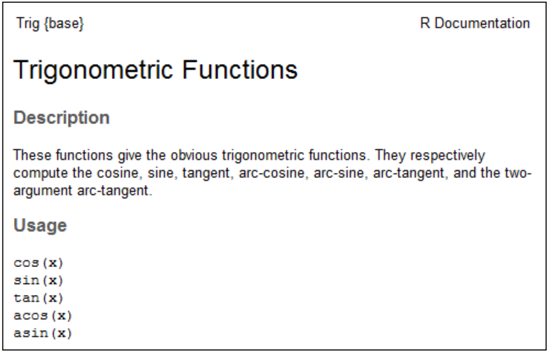
- Sequence of characters are named strings
- Surrounded by double quotes (")
- Necessary for e. g. naming column names

```
"Text"

## [1] "Text"

"3.14"

## [1] "3.14"

"3.14" + 1  # mixing strings and numbers does not work

## Error:  non-numeric argument to binary operator
```

Help Pages

Accessing help pages for each function via `help(func)`

```
help(sin)
```



Trig {base}　　R Documentation

Trigonometric Functions

Description

These functions give the obvious trigonometric functions. They respectively compute the cosine, sine, tangent, arc-cosine, arc-sine, arc-tangent, and the two-argument arc-tangent.

Usage

```
cos(x)
sin(x)
tan(x)
acos(x)
asin(x)
```

Outline

1 General Information

2 Operations, Functions, Variables

3 Vectors

4 Matrices

5 Extensibility

6 Wrap-Up

Creating and Accessing Vectors

- Create vector filled with zeros via `numeric(n)`

```
numeric(4)

## [1] 0 0 0 0
```

- Vector elements are concatenated via `c(...)`

```
x <- c(4, 0, 6)
x

## [1] 4 0 6
```

- Accessing individual elements via squared brackets `[]`

```
x[1]  # first component

## [1] 4
```

- Selecting a range of elements

```
x[c(2, 3)]

## [1] 0 6
```

- Selecting everything but a subset of elements

```
x[-1]

## [1] 0 6

x[-c(2, 3)]

## [1] 4
```

- Dimension via `length()`

```
length(x)

## [1] 3
```

Updating Vectors

```
x <- c(4, 0, 6)
```

- Replacing values

```
x[1] <- 1  # replace first component
x

## [1] 1 0 6
```

- Appending elements

```
y <- c(x, 8)  # append an element
y

## [1] 1 0 6 8
```

Vectors: Concatenation

```
x <- c(4, 0, 6)
y <- c(8, 9)
```

- Combining several vectors is named concatenation

```
z <- c(x, y)  # concatenating two vectors
z

## [1] 4 0 6 8 9
```

- Replicating elements by `rep(val, count)` to form vectors

```
rep(1, 5)  # 5-fold replication of the value 1

## [1] 1 1 1 1 1

rep(c(1, 2), 3)  # repeat vector 3 times

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

Vector Functions

```
x <- c(1, 2, 3, 0, 10)
```

- Average value

```
mean(x)

## [1] 3.2
```

- Variance

```
var(x)

## [1] 15.7
```

- Sum of all elements

```
sum(x)

## [1] 16
```

Exercise: Vectors

Question

- How to compute a standard deviation of $x = \begin{bmatrix} 1 \\ 4 \\ 9 \end{bmatrix}$?
- Visit `http://pingo.upb.de` with code 1523

Vector Operations

```r
x <- c(1, 2)
y <- c(5, 6)
```

- Scaling

```r
10 * x

## [1] 10 20
```

- Addition

```r
x + y

## [1] 6 8

10 + x

## [1] 11 12
```

- Be careful with functions such as `sin()` on vectors!

Generating Sequences

- Integer sequences

```
1:4
```

```
## [1] 1 2 3 4
```

```
4:1
```

```
## [1] 4 3 2 1
```

- Arbitrary sequences

```
(1:10)/10
```

```
##  [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
seq(4, 5, 0.1)  # notation: start, end, step size
```

```
##  [1] 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0
```

Exercise: Vectors

Question

- How to compute $\sum_{i=1}^{100} i$?
- Visit `http://pingo.upb.de` with code 1523

Outline

1. General Information
2. Operations, Functions, Variables
3. Vectors
4. **Matrices**
5. Extensibility
6. Wrap-Up

Matrices from Combining Vectors

- Generating matrices by combining vectors with `cbind(...)`

```
height <- c(163, 186, 172)
shoe_size <- c(39, 44, 41)
m <- as.data.frame(cbind(height, shoe_size))
```

  ... but exhausting!

- `as.data.frame(...)` necessary to avoid so-called `factor` objects

Files formatted as Comma Separated Values

- Support of naive Excel format is unsatisfactory
- Recommended: Export as Comma Separated Values (CSV)
- In Excel via Save As → file type is CSV (Comma separated)
- Then: right mouse click → Open with → Text Editor → Check if there are commas

Example File: persons.csv

```
name,height,shoesize,age
Julia,163,39,24
Robin,186,44,26
Kevin,172,41,21
Max,184,43,22
Jerry,193,45,31
```

Matrices from Text Files

`read.csv(filename, ...)` imports data frame from text file

- `header=TRUE` specifies whether columns have names
- `sep=","` specifies column delimiter
- `as.data.frame(...)` guarantees output as data frame

```
d <- as.data.frame(read.csv("persons.csv",
     header=TRUE, sep=","))
d

##    name height shoesize age
## 1 Julia    163       39  24
## 2 Robin    186       44  26
## 3 Kevin    172       41  21
## 4   Max    184       43  22
## 5 Jerry    193       45  31
```

- Alternatively, choose path to file via `file.choose()` manually

```
d <- as.data.frame(read.csv(file.choose(),
     header=TRUE, sep=","))
```

Output: Matrices

- Show first 6 rows only (useful for large files)

```
head(d)

##    name height shoesize age
## 1 Julia    163       39  24
## 2 Robin    186       44  26
## 3 Kevin    172       41  21
## 4   Max    184       43  22
## 5 Jerry    193       45  31
```

- Show column names

```
str(d)

## 'data.frame':  5 obs. of  4 variables:
##  $ name    : Factor w/ 5 levels "Jerry","Julia",..: 2 5 3 4 1
##  $ height  : int  163 186 172 184 193
##  $ shoesize: int  39 44 41 43 45
##  $ age     : int  24 26 21 22 31
```

Accessing Matrices

- Dimension (#rows, #columns) or number of rows/columns

```
dim(d)

## [1] 5 4
```

```
nrow(d)

## [1] 5

ncol(d)

## [1] 4
```

- Access columns by name

```
d$height

## [1] 163 186 172 184 193

d[["height"]]

## [1] 163 186 172 184 193
```

- Accessing an individual element (notation: #row, #column)

```
d[1, 2]

## [1] 163
```

Selecting Elements

- Using single condition to select a subset of rows

```
d[d$age > 25, ]
```

```
##    name height shoesize age
## 2 Robin    186       44  26
## 5 Jerry    193       45  31
```

```
d[d$age == 32, ]
```

```
## [1] name     height   shoesize age
## <0 rows> (or 0-length row.names)
```

- Connecting several conditions (`&` is and, `|` is or)

```
d[d$age < 25 & d$height <= 163, ]
```

```
##    name height shoesize age
## 1 Julia    163       39  24
```

Adding Columns and Column Names

- Adding column

```
d[["heightInInch"]] <- d$height/2.51
d$heightInInch

## [1] 64.94 74.10 68.53 73.31 76.89
```

- Getting column names via `colnames()`

```
colnames(d)

## [1] "name"         "height"       "shoesize"     "age"
## [5] "heightInInch"
```

- Updating column names

```
colnames(d) <- c("name", "waist", "weight", "shoes",
                 "books")
colnames(d)

## [1] "name"   "waist"  "weight" "shoes"  "books"
```

Outline

1 General Information

2 Operations, Functions, Variables

3 Vectors

4 Matrices

5 **Extensibility**

6 Wrap-Up

Extending R: Packages

- Most routines (from e. g. time series, statistical tests, plotting) are in so-called packages
- Packages must be downloaded & installed before usage
- When accessing routines, must be loaded via `library(package)`
- Installing packages by clicking:

In R Console

- Menu Packages
- Install package(s) . . .
- Choose arbitrary server
- Choose package

In R Studio

- Menu Tools
- Install packages
- Enter package name in middle input box
- Press Install

Outline

1 General Information

2 Operations, Functions, Variables

3 Vectors

4 Matrices

5 Extensibility

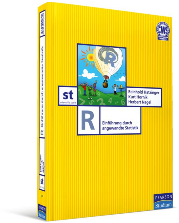
6 Wrap-Up

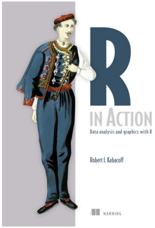
Tutorials on Using R

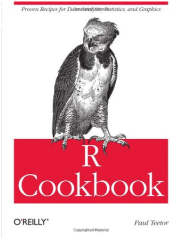
- Search Internet → many tutorials available online
- R Manual is the official introductory document
  → `http://cran.r-project.org/doc/manuals/R-intro.pdf`
- Helpful examples and demonstrations
  → `http://www.statmethods.net`
- Help pages in R describe parameters in detail, contain examples, but aim at advanced audience

Recommended Books

- German books
  - R-Einführung: Einführung durch angewandte Statistik (Pearson, 2011, by Hatzinger, Hornik & Nagel) `http://lib.myilibrary.com/Open.aspx?id=404906`
- English books (highly recommended)
  - R in Action: Data Analysis and Graphics with R (Manning, 2011, by Kabacoff, same as `statmethods.net`)
  - R Cookbook (O’Reilly, 2011, by Teetor)







Summary: Commands

+, −, etc.	Algebraic operators
&, |, <, <=, etc.	Logic operators
`help(func)`	Help pages
`mean(),var()`	Functions on vectors
`sd()`	Standard deviation
`seq()`	Generate sequences
`d$column`	Accessing columns of a matrix
`read.csv()`	Reading text files

Outlook

Additional Material

- Short summary of today’s lecture → Seminar Paper
- Further exercises as homework

Future Exercises

R will be used to solve sample problems from Business Intelligence