YasiR: Yet another short introduction to R

Bernd Weiss
bernd.weiss@uni-koeln.de
Research Institute for Sociology
University of Cologne
Germany

April 24, 2012

Outline

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow

- Some basics
- LaTeXin 5 minutes

Useful books and websites

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics
- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow
- Some basics
- LaTeXin 5 minutes

Useful books and websites

Acknowledgment, license and downloads

- My presentation was created using Emacs' *org-mode* and *Babel: active code in Org-mode*. *Babel* is developed and maintained by Eric Schulte and Dan Davison who were extremely helpful in answering my questions or fixing bugs.
- Licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Germany license.
- Slides, dataset and R code can be downloaded from my github page: https://github.com/berndweiss/ps2012-intro_R (see "Downloads" button on the right-hand side).

Objectives

- Introduce *some* basic concepts of R
- Show some common steps in data preparation and data analysis (esp. meta-analysis)
- Introduce my (R) data analysis workflow philosophy
- …

What is R?

“R is a language and environment for statistical computing and graphics. [...] Many users think of R as a statistics system. We prefer to think of it of an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics” (http://www.r-project.org/about.html).

Why use R?

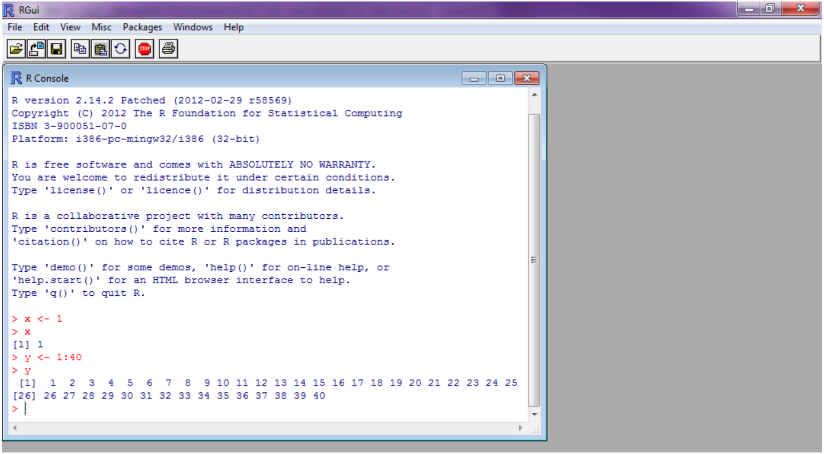
- It is free (GNU General Public Licence).
- Can be used on many platforms (MS Windows, Linux, Mac OS etc.).
- Includes cutting-edge statistical technologies and state-of-the-art graphics capabilities.
- Since R is a fully developed programming language, it is very (extremely) flexible.
- . . .

ℹ Inspired by Quick-R: Why Use R?

Why not use R?

- Steep learning curve.
- R is a programming language.
- Sometimes R lacks consistency (packages).
- ...

R under MS Windows



```
R version 2.14.2 Patched (2012-02-29 r58569)
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.

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.

> x <- 1
> x
[1] 1
> y <- 1:40
> y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
> 
```

Five basic R concepts you need to know

1. Objects
2. Packages
3. Grammar (Syntax) of R functions
4. Important data types/data structures
5. Missing values

It’s all about objects

- (Nearly) Everything in R is an object (some similarities to Stata’s container concept)
- What are objects? “The entities that R creates and manipulates are known as objects” (AItR: 5), e.g.:
  - Data sets
  - Variables
  - Results of any statistical calculation
  - . . .
- It is possible to access/manipulate pieces of more complex objects (e.g. datasets or regression results)

A first example of an R object

- “<-” means “assign”
- If you type in the object’s name, R prints out its value[1]
- “[...]” denotes R output (here, 1, only one element is shown)
- “#” is used for comments

```
x <- 1 ## assign value 1 to symbol/variable "x"
x      ## or: print(x)
```

```
[1] 1
```

```
x + x
```

```
[1] 2
```

```
x * 100
```

```
[1] 100
```

[1] Works in most but not all cases.

A second example of an R object

We also can create vectors (1×4) or matrices (2×3):

```r
x.vector <- c(1,2,3,4) ## c() means "concatenate"
x.vector
```

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

```r
x.matrix <- matrix(c(1,2,3,4,5,6), ncol = 3)
x.matrix
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

A third example of an R object

Conduct a t-test and save results in object `oTtest`.

```
oTtest <- t.test(rnorm(100) ~ sample(0:1, 100, replace = TRUE))
oTtest
```

```

	Welch Two Sample t-test

data:  rnorm(100) by sample(0:1, 100, replace = TRUE)
t = -0.3616, df = 94.733, p-value =
0.7184
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4470489  0.3092809
sample estimates:
mean in group 0 mean in group 1 
    -0.10122270     -0.03233873 
```

Get the test statistic...

```
oTtest$statistic
```

```
         t 
 -0.3616331 
```

...or the p-value.

```
oTtest$p.value
```

```
 [1] 0.7184316
```

Packages

- Most R functions are stored in packages.
- When you download R, it only comes with a very limited set of functions (e.g., it knows nothing about SPSS data files or meta-analysis).
- To load a particular package, use a command like `library(meta)`.
- However, before you can load a package, you have to install (download) it (only once). This can be done via `install.packages("meta")`.
- Whenever a new R session is started, the packages have to be loaded via `library(...)`.

ℹ AltR on packages
Quick-R on packages
UCLA's ATS on How can I manage R packages?

The basic syntax of an R function

- The general syntax is: `functionname(arglist)`
- `arglist`: A comma separated list of arguments which can be represented by `symbol = expression`
- Often, a symbol called `x` is used; `x` represents an R object
- Some simple examples:
  - Average of an object `a` (a vector): `mean(x = a)`
  - Standard deviation of `a`: `sd(x = a)`
  - Correlation between two vectors `a` and `b`: `cor(x = a, y = b)`
- Type `?functionname` and see “Usage” and “Arguments” for more information.

Important data types/data structures

- When you are used to SPSS or Stata, you never (rarely) had to deal with data types or structures.
- The next few slides introduce some important data types or data structures. For R novices in the social sciences, the most important data structure you encounter is called “data frame”. A data frame can be used to store a typical rectangular social sciences data set with varying data modes (numeric, character)
- Typically, a data set is provided as text, csv, SPSS, Stata, SAS etc. file. When this file is loaded into R, (in most cases) it is available as data frame.

Important data types/data structures (cont'd)

- Scalar

```
x.scalar <- 1
x.scalar
```

```
[1] 1
```

- Vector

```
x.vector <- c(1,2,3)
x.vector
```

```
[1] 1 2 3
```

- Factor (nominal scale; sth like `mean(x.factor)` does not work!)

```
x.factor <- factor(c(1,2,3), labels = c("low", "middle", "high"))
x.factor
```

```
[1] low    middle high
Levels: low middle high
```

Quick-R on Data Types

Important data types/data structures (cont'd)

- Data frame (each column can have a different data type)

```
x.df <- data.frame(ID = c(1,2,3), sex = factor(c("f", "f", "m")),
                   age = c(22, 45, 12))
x.df
```

```
   ID sex age
1  1   f  22
2  2   f  45
3  3   m  12
```

- List (most complex data structure)

```
x.list <- list(a = c(1,2,3), b = x.df)
x.list
```

```
$a
[1] 1 2 3

$b
  ID sex age
1  1   f  22
2  2   f  45
3  3   m  12
```

Missing data

- The symbol NA (Not Available) represents missing values.
- Unlike SPSS, most R functions do not use a listwise deletion strategy, e.g.:

```
x.na <- c(1,2,3, NA, 5)
mean(x.na)
```

```
[1] NA
```

However, if you specify `na.rm = TRUE` then `mean()` will calculate the mean:

```
mean(x.na, na.rm = TRUE)
```

```
[1] 2.75
```

Quick-R on Missing data

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics
- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow
- Some basics
- LaTeXin 5 minutes

Useful books and websites

Download and installation

- R can be downloaded from the Comprehensive R Archive Network (CRAN). The process is as follows:
  1. Go to http://www.r-project.org/, click on the link "CRAN", can be found in the left navigation bar
  2. Choose a CRAN mirror (e.g., Wirtschaftsuniversitaet Wien: http://cran.at.r-project.org/).
  3. Choose a precompiled binary distribution ("Download and Install R") (e.g., Windows).
  4. Choose binaries for base distribution and then "Download R 2.15.0 for Windows". (I mostly choose the "patched" version, see "Other builds")
- After downloading R-2.15.0-win.exe, execute the file and enjoy!

Getting help

- `help(functionname)` (or `?functionname`) opens the help pages (in rare cases you have to use quotation marks, e.g. `help("[")`).
- `help.search("keyword")` searches all installed packages for “keyword” (e.g., help.search(“meta-analysis”)).
- The package `sos` offers the function `findFn()` which is much more flexible than `help.search()`, (e.g., `findFn("meta-analysis")`).
- CRAN Task Views give an overview with respect to a certain topic (e.g.,

“CRAN Task View: Statistics for the Social Sciences” or “CRAN Task View: Psychometric Models and Methods”).

i Quick-R on Getting Help

Keeping R up-to-date

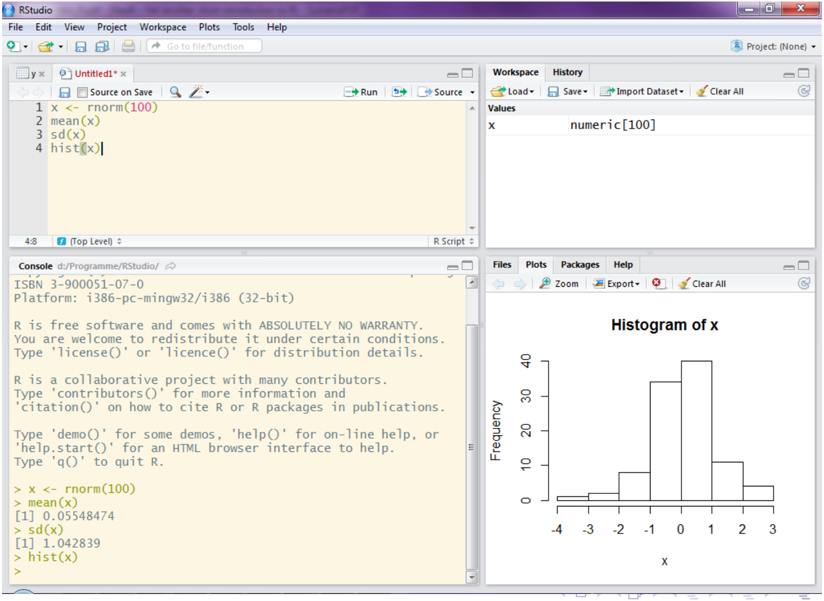
- Use the latest R version (updated twice a year).
- Updating packages is easy via `update.packages()`.

How to interact with R

(some statements refer to MS Windows only)

- Use the R console to type in R commands (REPL = read-eval-print loop).
- Use the built-in R script editor (see File - New script) to enter a (longer) sequence of R commands. Mark the lines which you want to run and press CTRL + r (STRG + r). This R script can be saved on you computer.
- An R script can also be "sourced", i.e. you can run the command `source("myRscript.R")` (in Stata: `use myStataFile.do`).
- Use a text editor which (at least) offers syntax highlighting.
  - A recommended solution is RStudio (see next slide; can be downloaded from http://rstudio.org/)
  - My preferred solution is Emacs + ESS.
  - See also The R GUI Projects website.

RStudio



```
x <- rnorm(100)
mean(x)
sd(x)
hist(x)
```

```
x                numeric[100]
```

```
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.

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.

> x <- rnorm(100)
> mean(x)
[1] 0.05548474
> sd(x)
[1] 1.042839
> hist(x)
>
```

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics
- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow
- Some basics
- LaTeXin 5 minutes

Useful books and websites

Overview

- R can handle many different data formats, e.g. SPSS, Stata, SAS, all sorts of text formats or DBMS.
- However, many data formats require you to load a certain package (e.g. `foreign`) which then provides a function to load the data.
- Whenever you load a specific dataset, you need to assign it to an object via `<-`.
- (Important!) Since R is supposed to work on different platforms, do not use the \-symbol (backslash) to specify a certain file within a certain folder. Instead, use / (slash) (or \\). This is okay: `c:/myfolder/script.R` and this is not going to work: `c:\myfolder\script.R`
- The functions `fix()` and `edit()` open a MS-Excel-like datasheet (under MS-Windows).

Loading a SPSS dataset

- `setwd()`: set working directory
- `library(foreign)`: Enables R to load SPSS datasets
- `read.spss()`: Read SPSS dataset (sometimes you receive a warning message “Warning message: In read.spss(… Unrecognized record type 7, subtype 18 encountered in system file”; this warning can be ignored.)
- `names()`: show column (“variable”) names of data object
- For a description see next slide

```
setwd("../data")
library(foreign)
dTeachExp <- read.spss(file = "dTeachExpRed.sav",
                       to.data.frame = TRUE,
                       use.value.labels = FALSE)
names(dTeachExp)
```

```
    [1] "ID"       "T"        "V"
[4] "weeks"   "weekcat"
```

The teacher-expectancy data set

XXX

Inspect your data I

```
head(dTeachExp) # prints first 6 cases
```

```
     ID     T         V weeks  weekcat
1  1  0.03 0.015625     2        2
2  2  0.12 0.021609    21        3
3  3 -0.14 0.027889    19        3
4  4  1.18 0.139129     0        0
5  5  0.26 0.136161     0        0
6  6 -0.06 0.010609     3        3
```

Another way to inspect your data is `edit()` or `fix()` (be careful not to modify your data unintentionally).

Inspect your data II

Here is a list of useful R functions to learn more about your data (object):

- `names()`: show column (“variable”) names of data object
- `dim()`: Retrieve (or set) the dimension of an R object, i.e. for an object of type `data.frame` it returns the number of rows and columns.
- `head()`: show first `n` cases (default is `n=6`)

Accessing elements of a data frame I

- Since R can handle many data objects, you first have to refer to a particular data object. Second, specify which element(s) you are interested in.
- There is a more general and a more specific method of accessing elements of a data frame: the `[`- and the $-operator.
- Using the $-operator, you only can access *one* element of the data frame. Using the `[`-operator, though, allows you to access more than one element.
- The use of `[`-operator depends on the number of dimensions of the R object. The different dimensions are separated by commas.

Accessing elements of a data frame II

```
dTeachExp[,"T"] # access variable T
```

```
 [1]  0.03  0.12 -0.14  1.18  0.26 -0.06
 [7] -0.02 -0.32  0.27  0.80  0.54  0.18
[13] -0.02  0.23 -0.18 -0.06  0.30  0.07
[19] -0.07
```

```
dTeachExp$T # access variable T, shortcut for dTeachExp[,"T"]
```

```
 [1]  0.03  0.12 -0.14  1.18  0.26 -0.06
 [7] -0.02 -0.32  0.27  0.80  0.54  0.18
[13] -0.02  0.23 -0.18 -0.06  0.30  0.07
[19] -0.07
```

```
dTeachExp[1:4, c("T", "weeks")] # access first 4 obs of T and weeks
dTeachExp2 <- dTeachExp[1:4, c("T", "weeks")] # new data frame-object
```

```
      T weeks
1  0.03     2
2  0.12    21
3 -0.14    19
4  1.18     0
```

Saving a dataset

- `save(object, file = "filename")` saves a particular data (or a list of objects) object to the specified file.
- `save.image(file = "filename")` saves the current workspace (i.e., all objects shown by `ls()` or `objects()`).
- `dump()` or `write.table()` saves data objects in plain text files.
- The `foreign` package has functions to save data objects as SPSS, Stata, SAS files.

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics
- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow
- Some basics
- LaTeXin 5 minutes

Useful books and websites

Overview

- Generate new variables
- Select cases (subsetting/indexing) and variables
- Missing values
- Recoding

Creating new variables

```r
dTeachExp$SE <- sqrt(dTeachExp$V) #or: dTeachExp[, "SE"]
head(round(dTeachExp, digits = 2))
```

```
  ID     T    V weeks weekcat   SE
1  1  0.03 0.02     2       2 0.12
2  2  0.12 0.02    21       3 0.15
3  3 -0.14 0.03    19       3 0.17
4  4  1.18 0.14     0       0 0.37
5  5  0.26 0.14     0       0 0.37
6  6 -0.06 0.01     3       3 0.10
```

Selecting/Removing cases I (subsetting/indexing)

Relational (<, >, <=, >=, ==, !=) and logical operators (&, |, !) can be used to select/remove certain cases.

```
subset(dTeachExp, weekcat == 0) #Keep weekcat == 0
```

```
   ID    T        V weeks weekcat    SE
4   4 1.18 0.139129     0       0 0.373
5   5 0.26 0.136161     0       0 0.369
9   9 0.27 0.026896     0       0 0.164
11 11 0.54 0.091204     0       0 0.302
12 12 0.18 0.049729     0       0 0.223
```

```
subset(dTeachExp, weekcat == 0 & T > 1)
```

```
  ID    T        V weeks weekcat    SE
4  4 1.18 0.139129     0       0 0.373
```

Selecting/Removing cases II

`subset()` is one way to create subsets. Another (and recommended) possibility is to use the `[`-operator.

```
dTeachExp[dTeachExp$weekcat == 0, ]
```

```
   ID    T        V weeks weekcat    SE
4   4 1.18 0.139129     0       0 0.373
5   5 0.26 0.136161     0       0 0.369
9   9 0.27 0.026896     0       0 0.164
11 11 0.54 0.091204     0       0 0.302
12 12 0.18 0.049729     0       0 0.223
```

```
dTeachExp[dTeachExp$weekcat == 0 & dTeachExp$T > 1, ]
```

```
  ID    T        V weeks weekcat    SE
4  4 1.18 0.139129     0       0 0.373
```

Selecting/Removing (or keeping) cases III

Say, you want to remove cases based on a list of person IDs. In that case, you can use the `%in%` function.

```
keep.ids <- c(1, 4, 6, 8)
dTeachExp.new <- dTeachExp[dTeachExp$ID %in% keep.ids, ]
dTeachExp.new
```

```
  ID     T        V weeks weekcat    SE
1  1  0.03 0.015625     2       2 0.125
4  4  1.18 0.139129     0       0 0.373
6  6 -0.06 0.010609     3       3 0.103
8  8 -0.32 0.048400    24       3 0.220
```

Removing missing values

```
dTeachExp.missing <- dTeachExp
dTeachExp.missing$T[c(1, 3, 6)] <- NA
dTeachExp.missing$weekcat[c(2, 3)] <- NA
head(dTeachExp.missing)
```

```
  ID    T        V weeks weekcat    SE
1  1   NA 0.015625     2       2 0.125
2  2 0.12 0.021609    21      NA 0.147
3  3   NA 0.027889    19      NA 0.167
4  4 1.18 0.139129     0       0 0.373
5  5 0.26 0.136161     0       0 0.369
6  6   NA 0.010609     3       3 0.103
```

Removing missing values (cont'd)

```
dTeachExp.missing[!is.na(dTeachExp.missing$T), ][1:6,]
```

```
   ID     T        V weeks weekcat    SE
 2  2  0.12 0.021609    21      NA 0.147
 4  4  1.18 0.139129     0       0 0.373
 5  5  0.26 0.136161     0       0 0.369
 7  7 -0.02 0.010609    17       3 0.103
 8  8 -0.32 0.048400    24       3 0.220
 9  9  0.27 0.026896     0       0 0.164
```

(For more information on using `is.na()` or similar functions, see slide 55.)

```
na.omit(dTeachExp.missing)[1:6,]
```

```
   ID     T        V weeks weekcat
4   4  1.18 0.139129     0       0
5   5  0.26 0.136161     0       0
7   7 -0.02 0.010609    17       3
8   8 -0.32 0.048400    24       3
9   9  0.27 0.026896     0       0
10 10  0.80 0.063001     1       1
      SE
4  0.373
5  0.369
7  0.103
8  0.220
9  0.164
10 0.251
```

Removing variables

```
(dTeachExp.names <- names(dTeachExp))
```

```
[1] "ID"      "T"       "V"
[4] "weeks"   "weekcat" "SE"
```

Remove the 1. and 3. variable

```
dTeachExp[1:2, c(dTeachExp.names)[-c(1,3)]]
```

```
     T weeks weekcat    SE
1 0.03     2       2 0.125
2 0.12    21       3 0.147
```

Remove `weeks` and `weekcat`.

```
!(dTeachExp.names %in% c("weeks", "weekcat"))
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE
```

```
dTeachExp[1:2,!(dTeachExp.names %in% c("weeks", "weekcat"))]
```

```
  ID    T        V    SE
1  1 0.03 0.015625 0.125
2  2 0.12 0.021609 0.147
```

Quick-R on Excluding (DROPPING) Variables

Recoding variables I

- There are several methods and functions available to recode variables.
- Using only R's base functions, recoding means to override values of a variable (or create a new variable) by new values based on a given condition.
- For example, replace all missing values (NA) in T with -99. Note that testing for equality does not work. Instead, use the `is.na()` function:

```
dTE.miss2 <- dTeachExp.missing # copy data object
dTE.miss2$T[is.na(dTE.miss2$T)] <- 99 # replace values
head(dTE.miss2, n = 3) # print out first 3 cases
```

```
  ID     T        V weeks weekcat    SE
1  1 99.00 0.015625     2       2 0.125
2  2  0.12 0.021609    21      NA 0.147
3  3 99.00 0.027889    19      NA 0.167
```

Quick-R on Recoding variables

Recoding variables II

- Another way to go is to use the function `ifelse`.
- The syntax is quite simple: `ifelse(test, yes, no)`
- `ifelse`-statements can be nested, i.e. `ifelse(test, yes, ifelse(test, yes, no))`

```
dTE.miss2$T <- ifelse(dTE.miss2$T == 99, NA,
                      dTE.miss2$T)
head(dTE.miss2, n = 4)
```

```
  ID    T        V weeks weekcat    SE
1  1   NA 0.015625     2       2 0.125
2  2 0.12 0.021609    21      NA 0.147
3  3   NA 0.027889    19      NA 0.167
4  4 1.18 0.139129     0       0 0.373
```

Recoding variables III

A third approach using `cut()` is handy when it comes to grouping a continuous variable, e.g. age.

```
dTE.miss2$weeks
```

```
 [1]  2 21 19  0  0  3 17 24  0  1  0  0
[13]  1  2 17  5  1  2  7
```

```
cut(dTE.miss2$weeks,
    breaks = c(0, 5, 10, 15, 20, 25),
    include.lowest = TRUE)
```

```
 [1] [0,5]   (20,25] (15,20] [0,5]  
 [5] [0,5]   [0,5]   (15,20] (20,25]
 [9] [0,5]   [0,5]   [0,5]   [0,5]  
[13] [0,5]   [0,5]   (15,20] [0,5]  
[17] [0,5]   [0,5]   (5,10] 
5 Levels: [0,5] (5,10] ... (20,25]
```

Recoding variables IV

Finally, there are several packages which offer “typical” recode- or replace-functions. Here, I will introduce the `recode()` function from the `car` package.

```
library(car)
dTE.miss2$weeks
```

```
 [1]  2 21 19  0  0  3 17 24  0  1  0  0
[13]  1  2 17  5  1  2  7
```

```
recode(dTE.miss2$weeks, "0:5 = ’niedrig’;
                         6:10 = ’bloed’;
                         11:15 = ’was?’;
                         else = ’rest’")
```

```
 [1] "niedrig" "rest"    "rest"
 [4] "niedrig" "niedrig" "niedrig"
 [7] "rest"    "rest"    "niedrig"
[10] "niedrig" "niedrig" "niedrig"
[13] "niedrig" "niedrig" "rest"
[16] "niedrig" "niedrig" "niedrig"
[19] "bloed"
```

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow

- Some basics
- LaTeXin 5 minutes

Useful books and websites

Make up some data

The next few slides will rely on some fake data.

```
df.fake <- data.frame(
             x = rnorm(10), # standard normal distr.
             y = rnorm(10, mean = 10, sd = 5),
             sex = factor(rep(c("f", "m"), 5))
             )
df.fake[1:4, ] # show rows 1 to 4
```

```
           x         y sex
1 -0.9527188 13.874167   f
2 -0.8485128  8.439426   m
3  0.2242950  4.076172   f
4  2.3423277  7.002842   m
```

Subsection overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow

Useful books and websites

The summary() function

```
summary(df.fake)
```

```
      x                   y
 Min.   :-2.03306   Min.   : 4.076
 1st Qu.:-0.92667   1st Qu.: 7.362
 Median : 0.19037   Median : 8.812
 Mean   :-0.01863   Mean   : 9.287
 3rd Qu.: 0.73509   3rd Qu.:12.131
 Max.   : 2.34233   Max.   :14.280
 sex
 f:5
 m:5
```

```
summary(df.fake$x)
```

```
     Min.  1st Qu.   Median     Mean
-2.03300 -0.92670  0.19040 -0.01863
 3rd Qu.     Max.
 0.73510  2.34200
```

The `mean()` and `median()` functions

```
mean(df.fake$x)
mean(df.fake$y)
```

```
      [1] -0.01863347
  [1] 9.286563
```

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow

- Some basics
- LaTeXin 5 minutes

Useful books and websites

The is.*() functions

- Sometimes, we want to check some properties of an R object, e.g. is a certain object of class “data frame” or does it contain missing values (NA=s). - R provides a number is =is.*()-functions which perform these tests and return a logical object (with values TRUE or FALSE).
- Some common examples:

```
x.df <- data.frame(x=1, y=2)
is.data.frame(x.df)
is.vector(x.df)
is.na(c(1, 2, 3, NA, NA))
```

```
[1] TRUE
[1] FALSE
[1] FALSE FALSE FALSE  TRUE  TRUE
```

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow

- Some basics
- LaTeXin 5 minutes

Useful books and websites

Subsection overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow

- Some basics
- LaTeXin 5 minutes

Useful books and websites

What is reproducible research?

“By reproducible research, we mean research papers with accompanying software tools that allow the reader to directly reproduce the results and employ the methods that are presented in the research paper” (Gentleman/Lang 2004: 1).

Requirements for the workflow: TREMMP

- Transparency (e.g., by using dynamic documents, “The source code is real”)
- Reproducibility (e.g., by using dynamic documents, “The source code is real”)
- Efficiency (a good workflow saves you time, by automating as much of the process as possible)
- Maintainability (standardized script names, good commenting practices, README files)
- Modularity (discrete tasks into separate components (e.g. scripts))
- Portability (e.g., by using relative (not absolute) pathnames)

(Source: David Smith on “A workflow for R”: http://blog.revolutionanalytics.com/2010/10/a-workflow-for-r.html)

The source code is real

“The source code is real. The objects are realizations of the source code. Source for EVERY user modified object is placed in a particular directory or directories, for later editing and retrieval” (Rossini et al. 2011:ESS - Emacs Speaks Statistics - Manual)

Use `source()` to read R code from a file

The R console can be used for short and temporary tests. In order to establish a TREMMP workflow, however, it is required to write R programs and to `source` them. So, use `source(file = "myfile.R")` to run an external R program. In SPSS, you would create an `.sps`-file, in Stata a `.do`-file.

More on reproducible research

- Kieran Healy: “Choosing Your Workflow Applications” http://www.kieranhealy.org/files/misc/workflow-apps.pdf
- … to be continued …

Subsection overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

(Some) Advanced functions of the R language

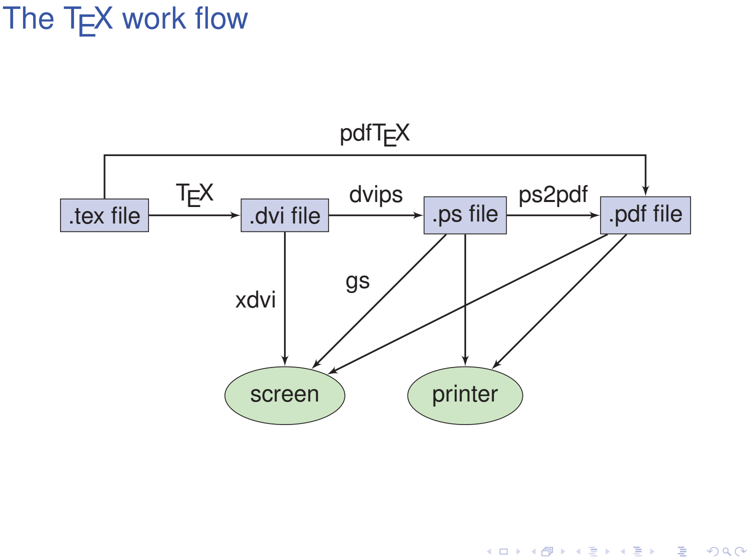
Reproducible research (RR) and workflow

- Some basics
- LaTeXin 5 minutes

Useful books and websites

What is LaTeX

- LaTeX is a markup language. Another markup language you might know is HTML.
- LaTeX provides high-quality typesetting features.
- The typical workflow is as follows:
  1. Create LaTeX source code file (`.tex`)
  2. Compile it via LaTeX or pdfLaTeX
  3. Use a viewer (PDF, DVI or via dvips PS) to view the compiled file
- In order to run LaTeX on your computer, you will need to install a LaTeX-distribution (e.g., MikTeX for MS-Windows).



The TEX work flow

pdfTEX

.tex file

TEX

.dvi file

dvips

.ps file

ps2pdf

.pdf file

xdvi

gs

screen

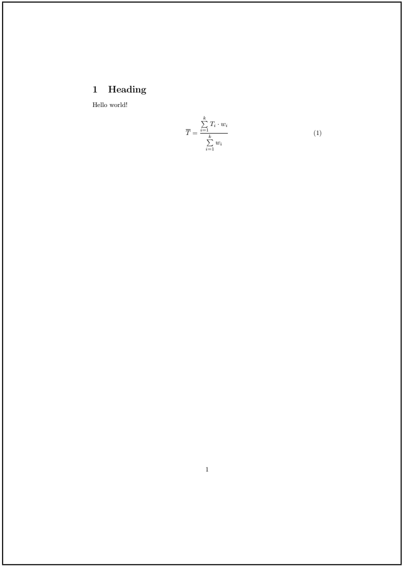
printer

Source:
http://media.texample.net/tikz/examples/PDF/tex-workflow.pdf

What a LaTeX file looks like

```latex
%% Part 1: Preamble
\documentclass{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[english]{babel}

%% Part 2: Body
\begin{document}

\section{Heading}

Hello world!

\begin{equation}
\overline{T} = \frac{\sum\limits^{k}_{i = 1} %
  T_{i}\cdot w_{i}}{\sum\limits^{k}_{i = 1}w_{i}}
\end{equation}

\end{document}
```

The compiled ‘Hello world’-example



1 Heading

Hello world!

$$\overline{T} = \frac{\sum\limits_{i=1}^{k} T_i \cdot w_i}{\sum\limits_{i=1}^{k} w_i} \tag{1}$$

1

Section overview

Introduction and 5 basic R concepts

Installation, help, maintenance and interacting with R

Loading a (SPSS) dataset

Data cleaning and data preparation

Descriptive statistics

- Mean, median & Co

(Some) Advanced functions of the R language

Reproducible research (RR) and workflow

- Some basics
- LaTeXin 5 minutes

Useful books and websites

Books and websites (in English)

- Websites
  - The R Manuals (esp. An Introduction to R)
  - Quick-R
  - Using R for psychological research: A simple guide to an elegant package
  - R for SAS and SPSS users (see “Free Version”)
  - See also the R Wiki
  - …
- Books
  - R for SAS and SPSS users by RA Muenchen
  - Introductory Statistics with R by P Dalgaard
  - See also Books related to R
  - …

Books and websites (in German)

- Books
  - Wikibooks GNU R: http://de.wikibooks.org/wiki/GNU_R
- Websites