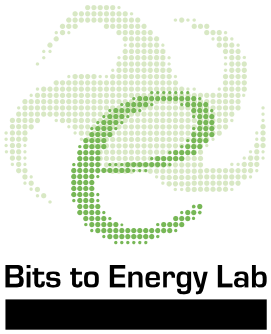

Applied Data Analytics and Machine Learning in R

Section 3, Introduction to R 2

Prof. Dr. Thorsten Staake, Carlo Stingl and Joanna Graichen

Version 1.2

Agenda

Working with R

- Load Data
- Data types
- Functions
- Packages
- Documentation
- Descriptive statistics

Advanced R for Data Analysis – dplyr and Piping

- Basic Data Operations
- Grouped Calculations
- Pipelining

Adapted version of Hadley Wickham's "Data Manipulation with Dplyr"

Load data

- Data can be loaded with
  - read.table()
  - read.csv()
  - read.csv2()

unspecific function with many parameters

comma (,) separated files

semicolon (;) separated files

- More information with ?read.table
- Tip: Read data always in UTF-8 format and convert other charsets to UTF-8 before loading

Load the data (Shower data)

- Download the first dataset from edX:
  Shower Dataset
- Load the dataset into your R session
- Inspect the dataset:
  - Use summary() and str() for a first overview
  - See the first and last n rows with head(,n) and tail(,n)
  - Count the number of rows and columns with nrow() and ncol()
  - Do you see columns that have a wrong data type?

Exercises: 26-28

Data types

- While importing data, most variables have the type “character”
- The most important types and related functions for type conversion:

	Boolean values	Integers and floating point numbers	Strings and letters	Categories
Create	logical()	numeric()	character()	factor()
Query	is.logical()	is.numeric()	is.character()	is.factor()
Conversion	as.logical()	as.numeric()	as.character()	as.factor()
Examples	TRUE; FALSE	5; 7.12; -3.5	“hello”; “world”; “2”	(A,B,B)

Functions

- The general command to create functions is:

```
<name> <- function(     <input1>,
                    <input2>=<default_value>){
  <instructions>
  return(<result>)
}
```

- E.g., pot <- function(x,y=2){return(x^y)}

Installation and use of additional packages

- install packages: `install.packages("<Name>")`
- use packages: `library(<Name>)` or `require(<Name>)`
- Some package recommendations:

Package name	Short description
VIM	visualizing analyzing missing values
class, caret, e1071	Functions for machine learning
sp, splines	Spatial analysis and representation in R
lattice, ggplot2	Data visualization
tm	Text mining package
dplyr, tidyr	Data manipulation

Inspect the "CRAN Task Views"!

http://cran.r-project.org/web/views/

Code documentation

- Always document your code!
  - What is this file / function / plot about?
  - Explain tricky aspects to others and yourself (in t > 100 days)
- Basic comments with #
- Enhanced **roxygen** comments with #'
  - Simple markup language
  - Syntax highlighting with RStudio
  - Direct export to .Rd files

```
#' Add together two numbers.
#'
#' @param x A number.
#' @param y A number.
#' @return The sum of \code{x} and \code{y}.
#' @examples
#' add(1, 1)
#' add(10, 1)
add <- function(x, y) {
  x + y
}
```

http://r-pkgs.had.co.nz/man.html

RMarkdown – Integrated data analytics and documentation

- Data analytics is not just scripting and programming!
  - During analysis you derive findings and conclusions
  - The “R script” is usually no program you re-run on new data, it is rather step-wise executed code that needs to be documented (for you & others)
- RMarkdown combines the code documentation hypertext language Markdown (*.md) with R code
- RMarkdown has a tight integration in RStudio and enables you to automatically generate reports

```
---
:itle: "My analysis"
uthor: "Konstantin Hopf"
late: "28 Oktober 2016"
utput: html_document
---

 Heading
# Subheading

lere you write your prosa text that explains your code.
larkdown is a simple formatting syntax for authoring HTML,
'DF, and MS Word documents. For more details on using R
larkdown see <http://rmarkdown.rstudio.com>.


``{r first plot}
lot(1:10, pch=1:10, col=1:10, cex=3, lwd=3)
``
```

http://rmarkdown.rstudio.com

Using R for calculating statistical measures

1. Compute the average Showertime using `mean()`.
2. Compute the Showertime variance with `var()`.
3. Compare mean Showertime to the result of `median()`.
4. Compute the standard deviation of Volume with `sd()`.
5. Compute Volume range `max()` and `min()`.
6. Apply `quantile()` to ShowerTime.

Dplyr – “a grammar of data manipulation“

- Dplyr allows us to analyze and evaluate data in a simple, syntactically consistent and very fast manner
- Basic data operations are provided by the following methods
  - Select
  - Filter
  - Summarize
  - Mutate
  - Arrange
- All of these methods take a data frame as a first argument, subsequent arguments specify what to do with the data frame

Dummy data set

- For demonstration purpose, we first specify an exemplary data frame containing five showers from two households

```
shower_data <- data.frame(Hh_ID = c(1,1,1,2,2), Volume  =
c(30, 33, 23, 40, 35))
```

Hh_ID	Volume
1	30
1	33
1	23
2	40
2	35

Filter

- filter allows to retrieve all rows fulfilling a given logical condition

```
filter(shower_data, Volume > 30)
```

Hh_ID	Volume
1	30
1	33
1	23
2	40
2	35

Hh_ID	Volume
1	33
2	40
2	35

- The non-dplyr equivalent would be

```
shower_data[shower_data$Volume > 30, ]
```

Filter (cont'd)

- filter allows to retrieve all rows fulfilling a given logical condition

```
filter(shower_data, Volume %in% c(33, 40))
```

Hh_ID	Volume
1	30
1	33
1	23
2	40
2	35

Hh_ID	Volume
1	33
2	40

Arrange

- arrange allows to order the rows in an ascending or descending order

```
arrange(shower_data, Volume)
```

Hh_ID	Volume
1	30
1	33
1	23
2	40
2	35

Hh_ID	Volume
1	23
1	30
1	33
2	35
2	40

```
Descending: arrange(shower_data, desc(Volume))
```

Exercise I (dplyr)

- Use the shower data set from the edX site. Install dplyr.
- Up to you:

1. Select all showers from Household “6395” and assign these to variable “a”.
2. Order the showers of household “6395” by the recorded Volume and assign these to variable “b”.
3. Select all showers with the exception of Household “6395” and “5307” and assign them to “c”.

Mutate

- mutate allows to add new columns based on other row-specific values

```
mutate(shower_data, squaredVolume = Volume^2)
```

Hh_ID	Volume
1	30
1	33
1	23
2	40
2	35

Hh_ID	Volume	squaredVolume
1	30	900
1	33	1089
1	23	529
2	40	1600
2	35	1225

Summarize

- Summarize transforms variables to numbers

```
summarize(shower_data, sumVolume = sum(Volume),
                       maxVolume = max(Volume))
```

Hh_ID	Volume
1	30
1	33
1	23
2	40
2	35

sumVolume	maxVolume
161	40

- Typically, you want to perform such operations on grouped data (i. e., on each household)

Grouping

- Grouping based on certain variables (here: Hh_ID)

```
grouped_showers <- group_by(data, Hh_ID)
summarize(grouped_showers, sumVolume = sum(Volume),
                           maxVolume = max(Volume))
```

Hh_ID	Volume
1	30
1	33
1	23
2	40
2	35

Hh_ID	sumVolume	maxVolume
1	86	33
2	75	40

Exercise II (dplyr)

- Up to you:

1. Determine the maximum and the minimum shower duration of the data set.
2. Introduce a new column “Avgtemperaturefahrenheit”. Therefore, convert the column “Avgtemperature” to the Fahrenheit unit using the package weathermetrics.
3. Calculate the average shower volume for each household , the average shower temperature and the average shower duration.

Data operations create a lot of intermediate variables

- In order to calculate specific measures, one often has to filter, group, summarize and arrange a data set.

- However, this procedure creates a lot of intermediate variables, which are not needed – resulting in very verbose code , such as:

```
filtered_data <- filter(exemplary_data, Hh_ID != 1)

grouped_showers <- group_by(filtered_data, Hh_ID)

summarized_showers <- summarize(grouped_showers, sumValue =
sum(Volume), maxValue = max(Volume))

ordered_showers <- arrange(summarized_showers, Hh_ID)
```

Data operations create a lot of intermediate variables (cont'd)

- The presented data operations always take a data frame as first argument and return a data frame
- dplyr provides a pipe operator, which can be used to pass the resulting data frame to the next operation

```
exemplary_data %>% filter(Hh_ID != 1) %>%
                   group_by(Hh_ID) %>%
                   summarise(sumValue = sum(Volume),
                             maxValue = max(Volume)) %>%
                   arrange(Hh_ID)
```

Exercise III (dplyr)

- Up to you! Use data pipes to

1. Calculate for each household the average shower volume, the average shower temperature and the average shower duration.
2. Determine the number of devices (=households), which have recorded more than 50 showers.
3. Calculate the average number of recorded showers per group.

End of this subsection.