University of Zurich UZH

MAKERERE UNIVERSITY

Data Analysis with R:
Lecture Slides (all)

Sonja Hartnack, Terence Odoch & Muriel Buri

July 2019

Goals of the course

To be able to...

- import data sets to R
- describe data with R
- apply basic statistical tests in R
- some ideas for more advanced statistical tools ...
- simulate a data set similar to own research

General remarks

Course schedule:

- Starting at 9:00am / 9:30am (?)
- Tea breaks in between
- Lunch break
- Teaching until 4.30pm ($\sim$ 5pm)

Obtaining a certificate is conditional on:

- active participation in class
- attending at least 75 % of the course (lecture & exercises)
- assignments during now and October
- short final exam in October (format to be defined)

Getting to know each other

- My name is ...
- I am doing a Master / a PhD in ...
- I hope to learn in this course how to ....
- My personal goal for this course is ...

How do we reach these goals

- hands on exercises with R:
    - `chickwts`
    - `ToothGrowth`
    - `bacteria`
    - `perulung`
    - ... and others.
- interactive discussions & student's present their own solutions
- ask us a lot of questions but also ask google for help!
- group work
- short motivational lectures

RStudio and R

Do you all have RStudio and R installed on your computers?

Get started with data set: chickwts

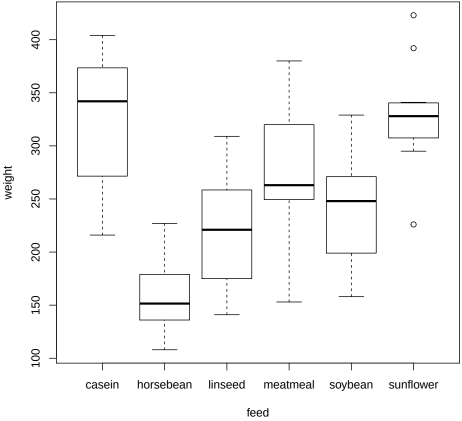
An experiment was conducted to measure and compare the effectiveness of various feed supplements on the growth rate of chickens.

```
# load data set "chickwts"
data("chickwts", package = "datasets")
# the head(...) function shows the first 6 observations
head(chickwts)

##   weight      feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
## 6    168 horsebean

# FUNCTION - open bracket - DATA SET / VARIABLE - close bracket
```

Ideas for plotting the data

Ideas for plotting the data

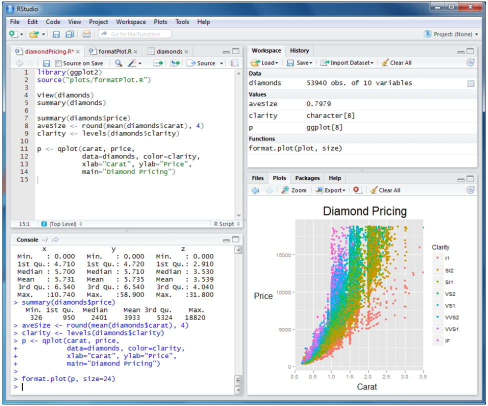
```r
# use x axis to show the categorical variable (feed),
# y axis to represent the continuous variable (weight)
# boxplot (y.cont.variable ~ x.cat.variable, data = dataset)
# ?boxplot
boxplot(weight ~ feed, data = chickwts)
```



Exercise: Statistical Terminologies

Functionalities in RStudio

- Source
- Console
- Environment, History, Files
- Files, Plots, Packages, Help



```
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")

```

```
       x                y                z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(p, size=24)
> 
```

Data	
diamonds	53940 obs. of 10 variables
Values	
aveSize	0.7979
clarity	character[8]
p	ggplot[8]
Functions	
format.plot(plot, size)	

Good housekeeping!

- Define manually a new folder called **rcourse** in your personal documents on your personal computer

- Know in which directory you are

```
getwd()

## [1] "/home/mburi/Documents/git_svn/DataAnalysisWithR/Lectures"
```

- Set directory path

```
# back- and forslash is dependent on the system
setwd("C:/Users/muriel/Documents/rcourse/")
setwd("C:\\Users\\muriel\\Documents\\rcourse\\")
```

- Always clean up before starting with new R-Script

```
rm(list=ls()) # empty workspace, delete previously saved variables
```

How to get help in R

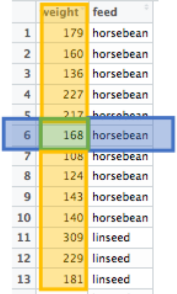
```
?chickwts
?boxplot
```

Also, have a look at the examples at the end of the help pages.

Exercise: Getting to know R and `chickwts`

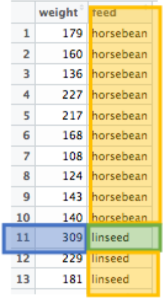
A data frame in R: `chickwts`

chickwts[ROWS , COLUMNS]



	weight	feed
1	179	horsebean
2	160	horsebean
3	136	horsebean
4	227	horsebean
5	217	horsebean
6	168	horsebean
7	108	horsebean
8	124	horsebean
9	143	horsebean
10	140	horsebean
11	309	linseed
12	229	linseed
13	181	linseed

chickwts[6, 1]



	weight	feed
1	179	horsebean
2	160	horsebean
3	136	horsebean
4	227	horsebean
5	217	horsebean
6	168	horsebean
7	108	horsebean
8	124	horsebean
9	143	horsebean
10	140	horsebean
11	309	linseed
12	229	linseed
13	181	linseed

chickwts[11, 2]

Rows and columns of a data frame: `chickwts`

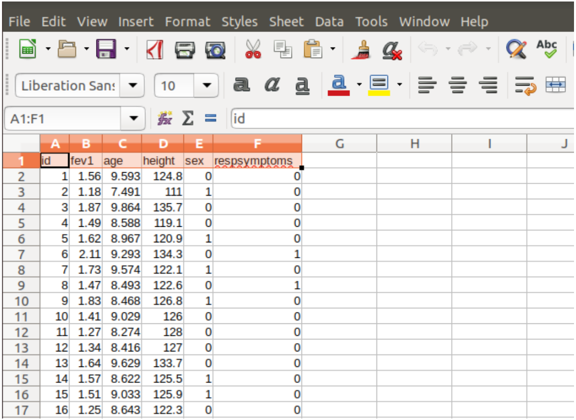
Values of ...

```r
# Load (internal) data set from R
data("chickwts")

# ... all columns of sixth observation:
chickwts[6, ]

# ... all columns of sixth to eleventh observation:
chickwts[c(6:11), ]

# ... all columns of sixth, eleventh and twentieth observation:
chickwts[c(6, 11, 20), ]

# ... all rows of first column (weight):
chickwts[ , 1]

# ... all rows of second column (feed):
chickwts[ , 2]

# or use the "$" sign as a reference to column "feed":
chickwts$feed
```

Exercise: Summary statistics for the `chickwts` data set

Rules for importing data into R

- First row of excel sheet contains **variable names**: `y, ap, hilo, week, ID, trt.`
- Columns of excel sheet represent **variables**.
- Rows of excel sheet represent **observations per individual** (except for the first row).



	A	B	C	D	E	F
1	id	fev1	age	height	sex	respsymptoms
2	1	1.56	9.593	124.8	0	0
3	2	1.18	7.491	111	1	0
4	3	1.87	9.864	135.7	0	0
5	4	1.49	8.588	119.1	0	0
6	5	1.62	8.967	120.9	1	0
7	6	2.11	9.293	134.3	0	1
8	7	1.73	9.574	122.1	1	0
9	8	1.47	8.493	122.6	0	1
10	9	1.83	8.468	126.8	1	0
11	10	1.41	9.029	126	0	0
12	11	1.27	8.274	128	0	0
13	12	1.34	8.416	127	0	0
14	13	1.64	9.629	133.7	0	0
15	14	1.57	8.622	125.5	1	0
16	15	1.51	9.033	125.9	1	0
17	16	1.25	8.643	122.3	0	0

Rules for naming variables

Variable names should ...

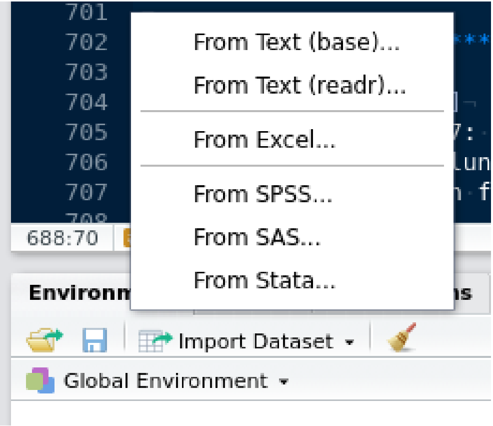
- start with a letter (not a number):
  `y, ap, hilo, week, ID, trt`

- longer variables names should be separated with dots:
  `time.in.weeks`

- do not use special characters, such as /, #, @, &, ⋆, ...

How to import external data files into R?

- > Import Dataset > **From Text (base)...** > CSV Files (.csv) or



How to import .txt and .csv files into R? (1/3)

- Environment (upper right corner)

- $>$ Import Dataset $>$ **From Text (base)...** $>$ CSV Files (.csv)

```r
perulung_ems <- read.csv("perulung_ems.csv", row.names = 1,
                         sep = ";")
lung <- data.frame(perulung_ems)
```

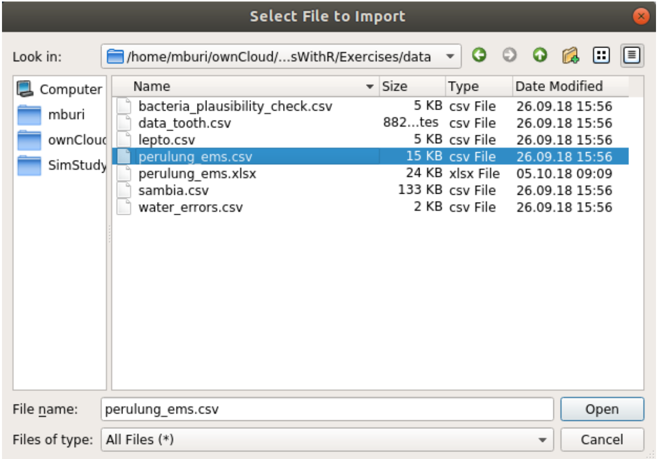
- $>$ Import Dataset $>$ **From Text (base)...** $>$ Text Files (.txt)

How to import .txt and .csv files into R? (2/3)

- Environment (upper right corner)
- > Import Dataset > From Text (base)... > CSV Files (.csv)

How to import .txt and .csv files into R? (2/3)

- Environment (upper right corner)
- > Import Dataset > From Text (base)... > CSV Files (.csv)



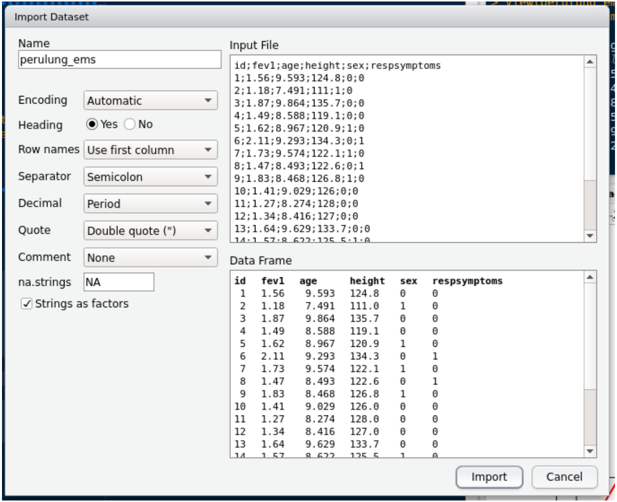
Look in: /home/mburi/ownCloud/...sWithR/Exercises/data

Name	Size	Type	Date Modified
bacteria_plausibility_check.csv	5 KB	csv File	26.09.18 15:56
data_tooth.csv	882...tes	csv File	26.09.18 15:56
lepto.csv	5 KB	csv File	26.09.18 15:56
perulung_ems.csv	15 KB	csv File	26.09.18 15:56
perulung_ems.xlsx	24 KB	xlsx File	05.10.18 09:09
sambia.csv	133 KB	csv File	26.09.18 15:56
water_errors.csv	2 KB	csv File	26.09.18 15:56

File name: perulung_ems.csv

Files of type: All Files (*)

How to import .txt and .csv files into R? (3/3)



Import Dataset

Name: perulung_ems

Encoding: Automatic

Heading: ● Yes ○ No

Row names: Use first column

Separator: Semicolon

Decimal: Period

Quote: Double quote (")

Comment: None

na.strings: NA

☑ Strings as factors

Input File

```
id;fev1;age;height;sex;respsymptoms
1;1.56;9.593;124.8;0;0
2;1.18;7.491;111;1;0
3;1.87;9.864;135.7;0;0
4;1.49;8.588;119.1;0;0
5;1.62;8.967;120.9;1;0
6;2.11;9.293;134.3;0;1
7;1.73;9.574;122.1;1;0
8;1.47;8.493;122.6;0;1
9;1.83;8.468;126.8;1;0
10;1.41;9.029;126;0;0
11;1.27;8.274;128;0;0
12;1.34;8.416;127;0;0
13;1.64;9.629;133.7;0;0
14;1.57;8.622;125.5;1;0
```

Data Frame

```
id  fev1   age    height  sex  respsymptoms
 1  1.56  9.593   124.8   0    0
 2  1.18  7.491   111.0   1    0
 3  1.87  9.864   135.7   0    0
 4  1.49  8.588   119.1   0    0
 5  1.62  8.967   120.9   1    0
 6  2.11  9.293   134.3   0    1
 7  1.73  9.574   122.1   1    0
 8  1.47  8.493   122.6   0    1
 9  1.83  8.468   126.8   1    0
10  1.41  9.029   126.0   0    0
11  1.27  8.274   128.0   0    0
12  1.34  8.416   127.0   0    0
13  1.64  9.629   133.7   0    0
14  1.57  8.622   125.5   1    0
```

```r
perulung_ems <- read.csv("perulung_ems.csv", row.names = 1,
                         sep = ";")
lung <- data.frame(perulung_ems)
```

Exercise: Data import to R and summary statistics `perulung_ems.csv`

Data from a study of lung function among children living in a deprived suburb of Lima, Peru. Data taken from Kirkwood and Sterne, 2nd edition.
Variables:

- `fev1`: in liter, ”forced expiratory volume in 1 second” measured by a spirometer. This is the maximum volume of air which the children could breath out in 1 second
- `age`: in years
- `height`: in cm
- `sex`: 0 = girl, 1 = boy
- `respsymp`: respiratory symptoms experienced by the child over the previous 12 months

What is a data frame in R?

A data frame is used for storing a list of vectors of equal length. For example, the following variable `df` is a data frame containing three vectors `n`, `s`, `b`.

```r
n <- c(2, 3, 5)
s <- c("aa", "bb", "cc")
b <- c(TRUE, FALSE, TRUE)
df <- data.frame(n, s, b) # df is a data frame
```

The characteristics of a data frame are:

- The column names should be non-empty.
- The row names should be unique.
- Each column should contain same number of data items.

Data frame in R

```
a <- c(1, 2, 3, 4)
a

## [1] 1 2 3 4

data.frame(a)

##   a
## 1 1
## 2 2
## 3 3
## 4 4

b <- c("d", "h", "h", "d")
mydat <- data.frame(a, b)
mydat

##   a b
## 1 1 d
## 2 2 h
## 3 3 h
## 4 4 d
```

Data frame in R: How to add a variable

```
vartoadd <- c(1.3, 1.5, 1.8, 2.4)
# use "$" to refer to the additional vector variable
mydat$myvar1 <- vartoadd
mydat$myvar2 <- vartoadd
mydat

##   a b myvar1 myvar2
## 1 1 d    1.3    1.3
## 2 2 h    1.5    1.5
## 3 3 h    1.8    1.8
## 4 4 d    2.4    2.4

# What is the dimension (number of rows and columns) of our data frame?
dim(mydat) # 4 rows and 4 columns

## [1] 4 4
```

Exercise: Defining a new data frame

Creating and assigning objects in R

Objects are assigned values using $< -$, an arrow formed out of $<$ and $-$. For example, the following command assigns the value 1 to the object a.

```r
a <- 1 # ALWAYS use "gets" assignment operator!
# a = 1 # DO NOT USE the equal sign as the assignment operator!
```

After this assignment, the object a contains the value 1. Another assignment to the same object will change its value.

```r
a <- 5
```

Examples of assigned objects: single number

```
a <- 1
b <- 2
c <- a + b # c = 3
c

## [1] 3
```

Examples of assigned objects: vector

```r
a <- c(1, 2, 3, 4, 5)
b <- 1
c <- a + b
c

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

Examples of assigned objects: model

```
anova_model <- aov(weight ~ feed, data = chickwts)
summary(anova_model)

##             Df Sum Sq Mean Sq F value   Pr(>F)
## feed         5 231129   46226   15.37 5.94e-10 ***
## Residuals   65 195556    3009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Examples of assigned objects: data frame

```
bac <- bacteria
str(bac) # $ week: int  0 2 4 11 0 2 6 11 0 2 ...

## 'data.frame': 220 obs. of  6 variables:
##  $ y   : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 1 2 2 2 ...
##  $ ap  : Factor w/ 2 levels "a","p": 2 2 2 2 1 1 1 1 1 1 ...
##  $ hilo: Factor w/ 2 levels "hi","lo": 1 1 1 1 1 1 1 1 2 2 ...
##  $ week: int  0 2 4 11 0 2 6 11 0 2 ...
##  $ ID  : Factor w/ 50 levels "X01","X02","X03",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ trt : Factor w/ 3 levels "placebo","drug",..: 1 1 1 1 3 3 3 3 2 2 ...

bac_sub <- subset(bac, week == 2)
str(bac_sub) # $ week: int  2 2 2 2 2 2 2 2 2 2 ...

## 'data.frame': 44 obs. of  6 variables:
##  $ y   : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 1 2 2 2 ...
##  $ ap  : Factor w/ 2 levels "a","p": 2 1 1 2 2 1 1 2 2 2 ...
##  $ hilo: Factor w/ 2 levels "hi","lo": 1 1 2 2 2 2 1 1 2 1 ...
##  $ week: int  2 2 2 2 2 2 2 2 2 2 ...
##  $ ID  : Factor w/ 50 levels "X01","X02","X03",..: 1 2 3 4 5 6 7 8 9 11 ...
##  $ trt : Factor w/ 3 levels "placebo","drug",..: 1 3 2 1 1 2 3 1 1 1 ...
```

Structure of a R objects

The `str` function displays the structure of an R object. One line for each ”basic” structure is displayed.

```
## 'data.frame': 44 obs. of  6 variables:
##  $ y   : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 1 2 2 2 ...
##  $ ap  : Factor w/ 2 levels "a","p": 2 1 1 2 2 1 1 2 2 2 ...
##  $ hilo: Factor w/ 2 levels "hi","lo": 1 1 2 2 2 2 1 1 2 1 ...
##  $ week: int  2 2 2 2 2 2 2 2 2 2 ...
##  $ ID  : Factor w/ 50 levels "X01","X02","X03",..: 1 2 3 4 5 6 7 8 9 11 ...
##  $ trt : Factor w/ 3 levels "placebo","drug",..: 1 3 2 1 1 2 3 1 1 1 ...
```

Exercise: Different bracket types within R

Data types in R

- numeric

```
data(ToothGrowth)
ToothGrowth$len[1:6]

## [1]  4.2 11.5  7.3  5.8  6.4 10.0

class(ToothGrowth$len[1:6])

## [1] "numeric"
```

- integers

```
bacteria$week[1:6]

## [1]  0  2  4 11  0  2

class(bacteria$week[1:6])

## [1] "integer"
```

- (un/ordered) factor

```
chickwts$feed[1:6]

## [1] horsebean horsebean horsebean horsebean horsebean horsebean
## Levels: casein horsebean linseed meatmeal soybean sunflower

levels(chickwts$feed)[1:3]

## [1] "casein"    "horsebean" "linseed"
```

Data types in R: Ordered Factors

Ordinal variables are represented as ordered factors:

```
bac_growth <- c("none", "+", "++", "+", "+++", "+", "none") # vector
bac_growth <- factor(bac_growth, levels = c("none", "+", "++", "+++"),
                     order = TRUE)
bac_growth

## [1] none +    ++   +    +++  +    none
## Levels: none < + < ++ < +++


#
mood <- c("OK", "Well", "Super", "Super", "Don't ask", "OK") # vector
mood <- factor(mood, levels = c("Don't ask", "Well", "OK", "Super"),
               order = TRUE)
mood

## [1] OK        Well      Super     Super     Don't ask OK
## Levels: Don't ask < Well < OK < Super
```

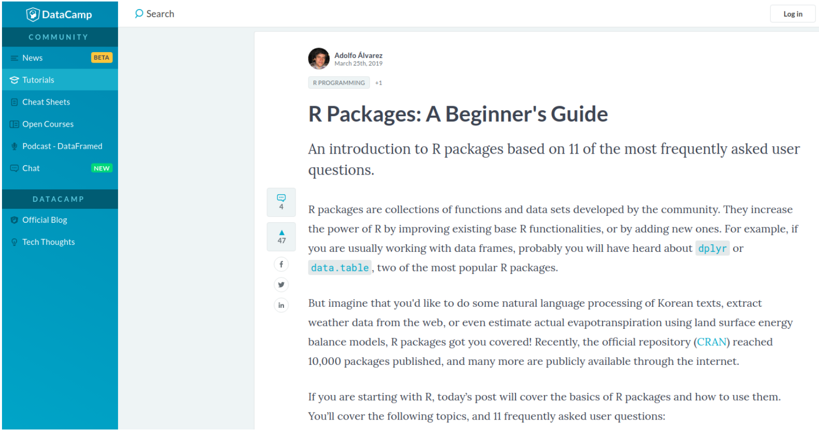
Examples of different data types

- numeric variable
- integer variable
- variable with two levels (binary factor)
- ordered variable with **more than** two levels (ordinal)
- unordered variable with **more than** two levels (nominal)

Exercise: Data type of `perulung_ems` data set

Introduction to R packages

https://www.datacamp.com/community/tutorials/r-packages-guide



Adolfo Álvarez
March 25th, 2019

R PROGRAMMING +1

R Packages: A Beginner's Guide

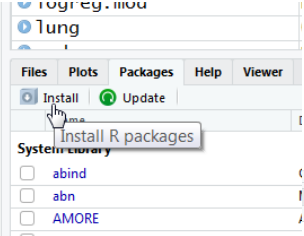
An introduction to R packages based on 11 of the most frequently asked user questions.

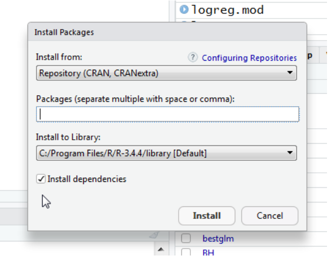
R packages are collections of functions and data sets developed by the community. They increase the power of R by improving existing base R functionalities, or by adding new ones. For example, if you are usually working with data frames, probably you will have heard about `dplyr` or `data.table`, two of the most popular R packages.

But imagine that you'd like to do some natural language processing of Korean texts, extract weather data from the web, or even estimate actual evapotranspiration using land surface energy balance models, R packages got you covered! Recently, the official repository (CRAN) reached 10,000 packages published, and many more are publicly available through the internet.

If you are starting with R, today's post will cover the basics of R packages and how to use them. You'll cover the following topics, and 11 frequently asked user questions:

How to install a package (manually) in R



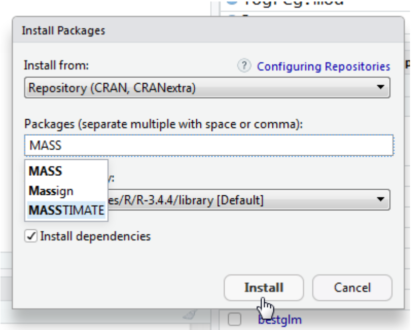

Install Packages

Install from: Repository (CRAN, CRANextra)

Packages (separate multiple with space or comma):

Install to Library: C:/Program Files/R/R-3.4.4/library [Default]

☑ Install dependencies


Install Packages

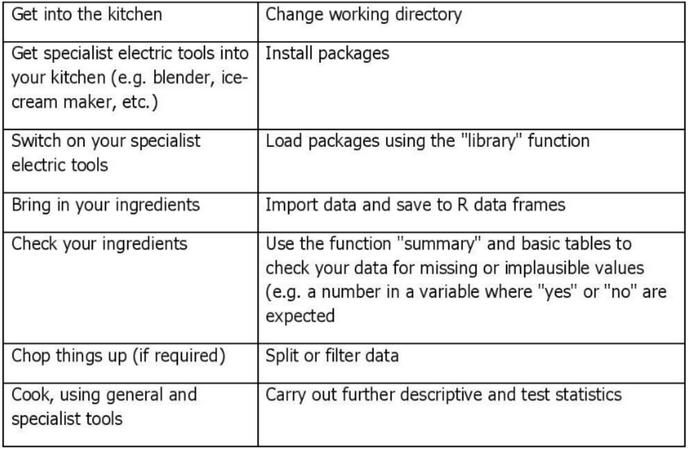
Install from: Repository (CRAN, CRANextra)

Packages (separate multiple with space or comma): MASS

es/R/R-3.4.4/library [Default]

☑ Install dependencies

Using R is like cooking ...



Get into the kitchen	Change working directory
Get specialist electric tools into your kitchen (e.g. blender, ice-cream maker, etc.)	Install packages
Switch on your specialist electric tools	Load packages using the "library" function
Bring in your ingredients	Import data and save to R data frames
Check your ingredients	Use the function "summary" and basic tables to check your data for missing or implausible values (e.g. a number in a variable where "yes" or "no" are expected
Chop things up (if required)	Split or filter data
Cook, using general and specialist tools	Carry out further descriptive and test statistics

How to install a package in R

```r
# INSTALL package (only done ONCE!)
install.packages("MASS")
# LOAD package (whenever you use something from it!)
library("MASS")
data(bacteria)
?bacteria
```

Exercise: Get to know `bacteria` data set

How to google for getting help in R

- Google for **select observations in R**.

Why do we need Statistics?

Repeatability of results:
Statistical science allows us to estimate what might happen if an experiment was repeated - but without having to actually repeat it!

Why do we need Statistics?

- Study results must be shown to be robust, i.e. real and not due to random chance
- Best way to demonstrate this is to repeat the same experiment/study many times each with **different** subjects (animals) drawn from the **same study population** and show that the result is truly repeatable
- It is generally totally impractical, in terms of both time and resources, to repeat an experiment many times!

Why do we need Statistics?

- Instead of repeating the experiment many times **probability theory i.e. statistics** is used to **estimate** what might have happened if the experiment had been repeated

- A mathematical model is used to fill this “data gap”

- Generally the most difficult task in statistics is to decide what “model” is most appropriate for a given experiment

What is Statistics? - A definition

A set of analytical tools designed to quantify uncertainty

- If an experiment or procedure is repeated, how likely is it that the new results will be similar to those already observed?

- What is the likely variation in results if the experiment was repeated?

What is Statistics? - A definition

The key scientific purpose of statistics

- to provide **evidence** of the existence of some “effect” of scientific interest
- i.e. evidence based medicine

As a reminder: The importance of study design

Even the most sophisticated statistical analyses cannot rescue a poorly designed study

$\rightarrow$ unreliable results

$\rightarrow$ inability to answer the main research question

Putting Statistics in Context

- Use common sense as a guide - be skeptical!
- Terminology can also differ greatly between textbooks...
- Wikipedia as good a resource

Exploratory Data Analysis

- get first impression and feeling of the data set
- detect outliers / mistake of data collection
- possibly recode variables

Summary Statistics
Continuous (Integers / Numeric)

- `Mean` - a measure of location. Always examine the average value of the response variable(s) for the different “treatment” effects in your data
- `Median` - a robust single value summary of a set of data (50% quantile point) - most useful in highly skewed data or data with outliers
- `Standard deviation (sd)` - a measure of spread, how variable the data are
- `Standard error of the mean (se)` - an estimate of how far the sample mean is likely to be from the population mean
- and others: `min, max, range, IQR, ...`

Continuous (Integers / Numeric) Summary Statistics

```
mean(x) # mean

median(x) # median

sd(x) # standard deviation

min(x) # minimum

max(x) # maximum

range(x) # range

IQR(x) # interquartile range
```

Continuous Data Summaries

standard deviation

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

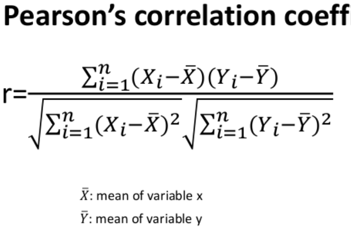
standard error

$$se = \frac{s}{\sqrt{n}}$$

Correlation coefficient

Combination of continuous and continuous

Correlation coefficient a measure association between two continuous variables (common but somewhat limited)



Pearson’s correlation coefficient r

$$r=\frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$$

$\bar{X}$: mean of variable x

$\bar{Y}$: mean of variable y

Correlation of continuous and ordinal variables

```r
# Test for Association/Correlation Between
# Paired Samples
cor.test(data$x, data$y, method = "pearson")
cor.test(data$x, data$y, method = "spearman")

# Scatterplot(s)
pairs(data$x ~ data$y)
pairs(data)
```

Summary Statistics

Continuous and ordinal variables

```r
tapply(data$x.cont, data$y.fac, mean)

tapply(data$x.cont, data$y.fac, median)

tapply(data$x.cont, data$y.fac, sd)
```

Summary Statistics
Ordinal

- `Median` - a robust single value summary of a set of data (50% quantile point) - most useful in highly skewed data or data with outliers

- `e.g.10th and 90th percentile` - a measure of spread, how variable the data are

```r
quantile(x, probs = c(0.1, 0.9))
```

- `proportions` - e.g. percentage per grade

```r
prop.table(table(data$x.fac))
prop.table(table(data$x.fac, data$y.fac))
```

Summary Statistics

Nominal

- `proportions` - percentage within the different categories
- `contingency tables` e. g. 2 x 2

```r
table(data$x.fac)
table(data$x.fac, data$y.fac)
prop.table(table(data$x.fac))
prop.table(table(data$x.fac, data$y.fac))
```

Exercise: Get to know `ToothGrowth` data set

How to deal with missing values in R? (1/3)

- In R, missing values are represented by the symbol **NA** (not available).
- Impossible values (e. g., dividing by zero) are represented by the symbol **NaN** (not a number).
- Ask yourself why a **NA** and / or **NaN** occurs!

How to deal with missing values in R? (2/3)

- Testing for Missing Values

```r
vec1 <- c(1, 2, 3, NA)
is.na(vec1) # returns a vector (FALSE, FALSE, FALSE, TRUE)
# The TRUE indicates the position of the NA in vec1.
```

- Recoding Values to Missing

```r
# recode specific values (e. g. 0.001) to missing for variable x
# select rows where x is 0.001 and recode value in column x with NA
tmp.row <- which(dat$x == 0.001)
dat$x[tmp.row] <- NA
```

How to deal with missing values in R? (3/3)

- Excluding Missing Values from specific function calls

```
a <- c(1, 2, NA, 3)
mean(a) # returns NA
mean(a, na.rm=TRUE) # returns 2
```

- Check for complete cases with function `complete.cases(...)`

```
# list rows of data that have missing values
dat[!complete.cases(dat),]
subdat <- dat[complete.cases(dat),]
```

- Create new dataset without missing data with function `na.omit(...)`

```
new.dat <- na.omit(dat)
```

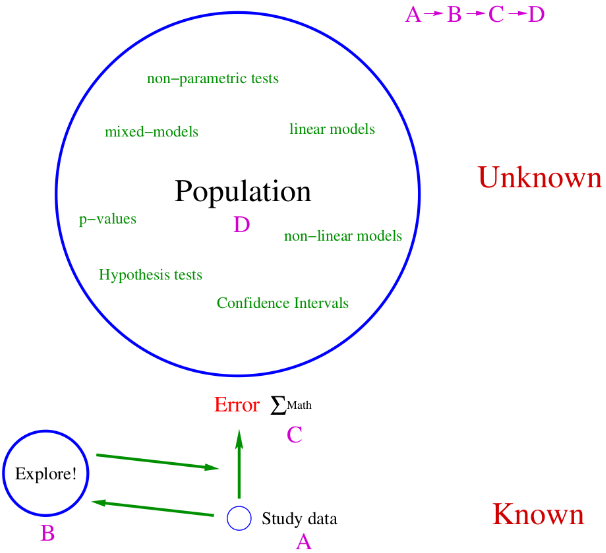
How to check your data for plausibility?

- Ask yourself what can go wrong?
- Implausible values?
- Impossible values?
- Logical errors?

Data plausibility checks & missing values

Exercise 10

Overview


A → B → C → D
non–parametric tests
mixed–models
linear models
Population
D
p–values
non–linear models
Hypothesis tests
Confidence Intervals
Unknown
Error
∑Math
C
Explore!
B
Study data
A
Known

Basic Statistical Tests

Study data is collected for a purpose - to answer one or more specific scientific questions. The classical way to perform a formal statistical analyses of these data is to formulate these research questions into statistical **hypothesis tests**.

In this section we will go through a simple example in detail to highlight some of the important concepts - the general approach for more complex analyses is exactly same. *Note: the precise technical details are much less important than the concepts!*

Simple Example - One Population

After six weeks will the mean weight of a chicken be more than 250 grams?

There are 71 observations in `chickwts` from which to answer this question. This can be formulated into a statistical hypothesis test. A hypothesis test has two parts, the null hypothesis and the alternative hypothesis. This is typically written as follows:

$$\begin{aligned} H_0: \quad \mu & \leq 250 \\ H_A: \quad \mu & > 250 \end{aligned}$$

where μ is the mean weight in the **population** of chickens from which the sample of 71 chickens was drawn. Remember - we know the mean weight in the sample of chickens is greater than 250 it is the **population** of chickens which we are interested in.

Simple Example - One Population

After six weeks will the mean weight of a chicken be at least 250 grams?

$$\begin{aligned} H_0 : \quad \mu &\leq 250 \\ H_A : \quad \mu &> 250 \end{aligned}$$

The null hypothesis (H_0) is the default situation, sometimes called the “state of nature”. In a treatment-control trial, H_0 is typically that the effect of the treatment is not different from the control. In this example our default position is that the mean weight of chickens is $\leq$ 250. This is called a single-sided hypothesis test.

Simple Example - One Population

We now analyse the 71 observations to see whether there is evidence to **REJECT** the null hypothesis H_0, and if the null hypothesis is rejected then we can conclude that the available evidence supports the alternative hypothesis.

```r
t.test(chickwts$weight, mu = 250, alternative = "greater")
t.test(chickwts$weight, mu = 250, alternative = "less")
```

Note that hypothesis testing is concerned with finding evidence in support of the null hypothesis H_0 - the default situation - rather than evidence in favour of the alternative hypothesis.

One Sample t-test

For the chicken weights data an appropriate formal analyses is to use a **one-sample t-test**, why this test is appropriate will be discussed later. This analysis involves calculating a simple summary statistic - called a *t*-statistic - which we do entirely from the observed data.

$$T_{obs} = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

where $\bar{x}$ is the sample mean, s the sample standard deviation and μ is the population mean in the null hypothesis which we wish to test for. We then look up the value of T_{obs} in a set of statistical tables/computer to see what the answer is to our research question.

Important concept - sampling

Why is $T_{obs} = \frac{\bar{x}-\mu}{s/\sqrt(n)}$ called a *t*-statistic?

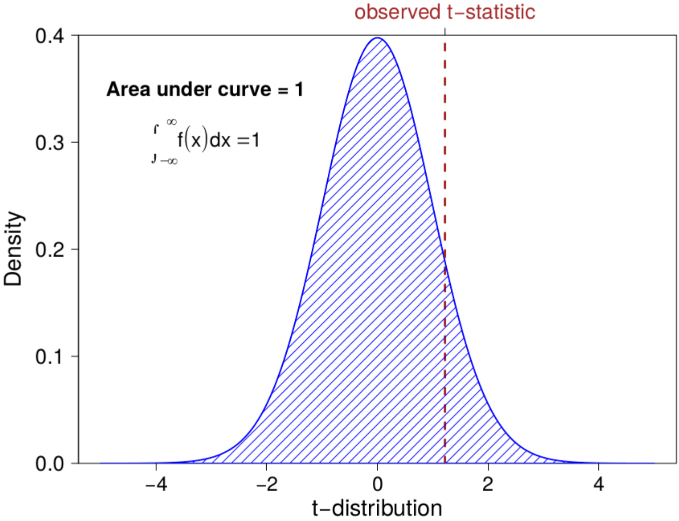
If another sample of 71 chickens from the same population were weighed then the values for $\bar{x}$ and s would be different, and hence the value for T_{obs}. If this was repeated many times and a histogram/Q-Q/P-P plot produced of the values for T_{obs} then this would follow the shape of a known distribution - **student-*t* probability distribution**. It is this piece of mathematics - knowing what the sampling distribution of T_{obs} is - which allows us to infer information about the population of chickens from which our original 71 chickens were sampled - without actually having to collect lots and lots of other samples of chickens! Mathematical theory is used to fill this data gap.

`chickwts`: t-test

$$T_{obs} = \frac{261.31 - 250}{78.07/\sqrt{71}} = 1.22$$

Put the values for the sample mean and standard deviation into the t-statistic formula along with the $\mu = 250$. We now look up the value of this in a t-distribution reference table. All this calculation will be done for you in R but it is important to understand the general process as this is the same for hypothesis testing in other more complex analyses.

One Sample, one-sided, t-test


observed t−statistic
Area under curve = 1
$\int_{-\infty}^{\infty} f(x)dx = 1$
Density
t−distribution