University of Zurich^UZH

MAKERERE UNIVERSITY

Data Analysis with R:

Lecture Slides: Day 2 - Tuesday

Sonja Hartnack, Terence Odoch & Muriel Buri

July 2019

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.

```
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.

```
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

```r
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
```

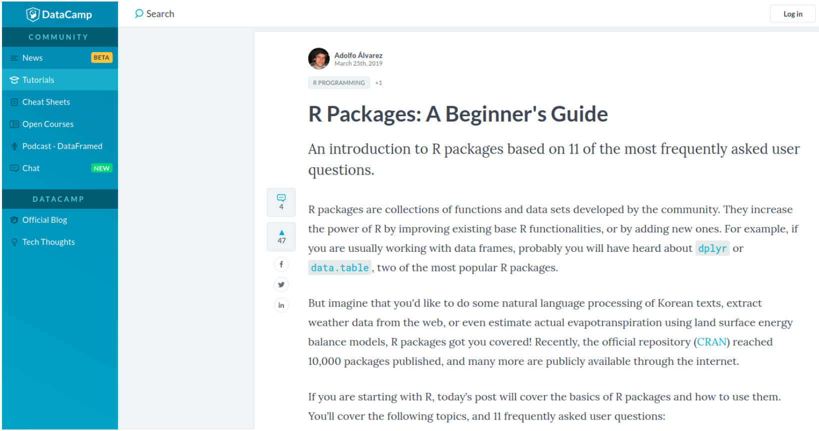
Examples of different data types

- 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 Packages: A Beginner's Guide

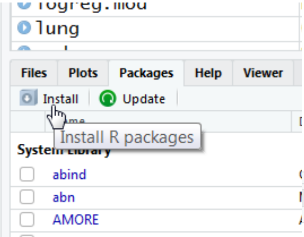
An introduction to R packages based on 11 of the most frequently asked user questions.

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.

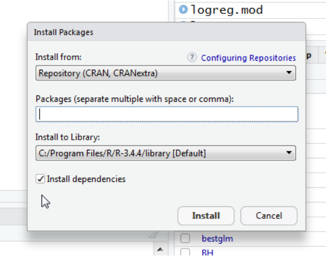
But imagine that you'd like to do some natural language processing of Korean texts, extract weather data from the web, or even estimate actual evapotranspiration using land surface energy balance models, R packages got you covered! Recently, the official repository (CRAN) reached 10,000 packages published,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



- □ abind
- □ abn
- □ AMORE



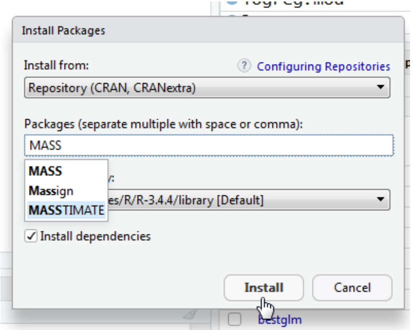
Install Packages

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

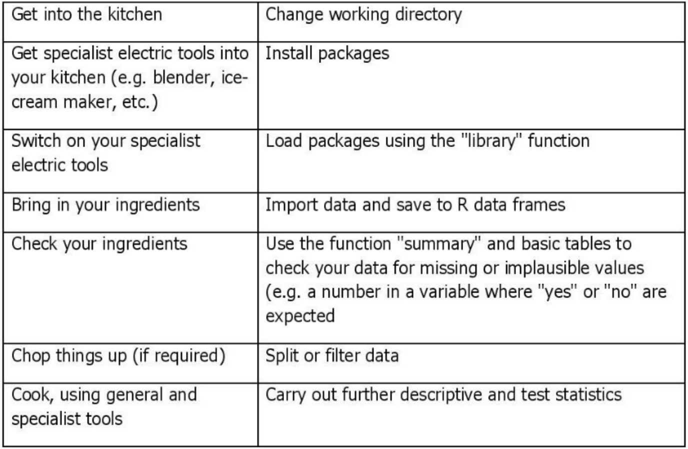
Install from: Repository (CRAN, CRANextra)

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

→ unreliable results

→ 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

```r
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 factor 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 factor 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

Factor (1/2)

- `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

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

Summary Statistics

Factor (2/2)

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

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

- `contingency tables` e. g. 2 x 2

```
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