University of Zurich[UZH]

MAKERERE UNIVERSITY

Data Analysis with R:
Day 2 - Lecture Slides

Sonja Hartnack, Terence Odoch & Muriel Buri

October 2018

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`.

```
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")
dat <- data.frame(a, b)
dat
```

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

Data frame in R: How to add a variable (var)

```
my.var <- c(1.3, 1.5, 1.8, 2.4)
# use "$" to refer to the additional vector variable
dat$my.var1 <- my.var
dat$my.var2 <- my.var
dat
```

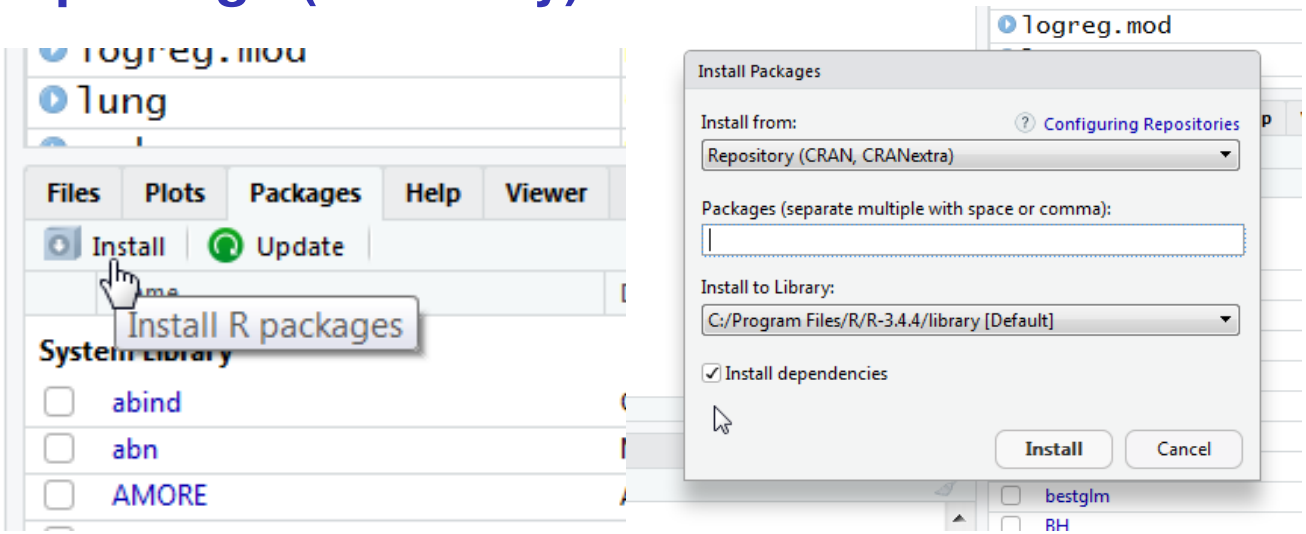
```
##   a b my.var1 my.var2
## 1 1 d     1.3     1.3
## 2 2 h     1.5     1.5
## 3 3 h     1.8     1.8
## 4 4 d     2.4     2.4
```

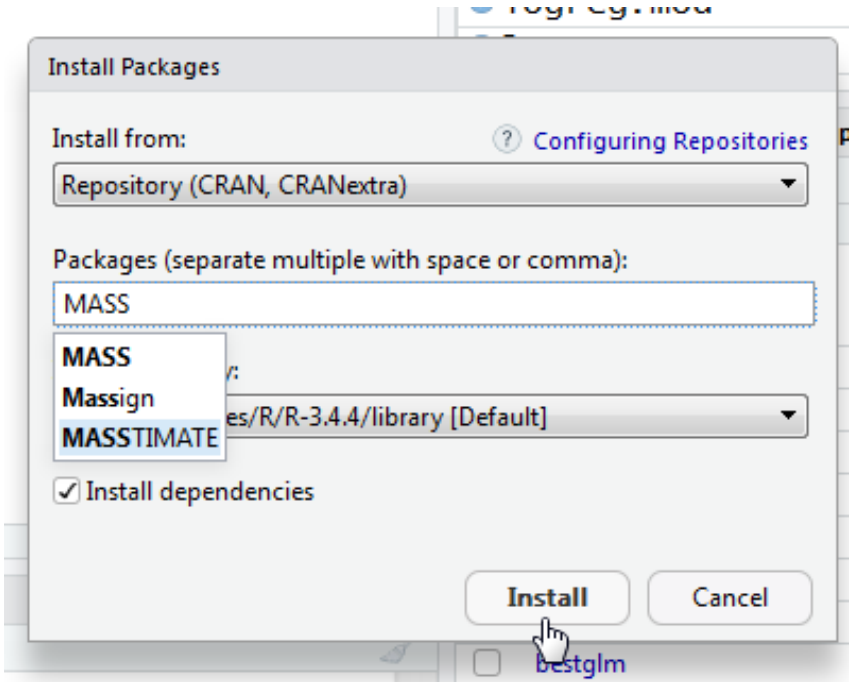
```
# What is the dimension (number of rows and columns) of our data frame?
dim(dat) # 4 rows and 4 columns
```

```
## [1] 4 4
```

Exercise 2

How to install a package (manually) in R





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

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

Exercise 3

How to google for getting help in R

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

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

```
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 4

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
```

Exercise 5

Exercise 6