

Lecture 3: An overview of R: Part II

- Assess the values of an object
- Enter or import data into R
- Export data
- Save and load data
- View data

3.1 Assess the values of an object - the index system of R

Key Operators are "[" and "\$"

Recall object classes:

- Vector
- Matrix
- Array
 - Recall that these three are essentially the same thing.
- Data frame
- List
- (Factor)

3.1.1 Index a vector

In [1]:

```
vector <- 2:6  
vector
```

2 3 4 5 6

In [2]:

```
# Pick the 2nd  
vector[2]
```

3

In [3]:

```
# Pick 2nd - 4th  
vector[2:4]
```

3 4 5

In [4]:

```
# Pick no. 1, 3, 5  
vector[c(1, 3, 5)]
```

2 4 6

In [5]:

```
# Code like a pro  
# This good practice makes it clearer for revisits and/or edits  
# Reproducibility!  
  
# Pick no. 1, 3, 5  
index <- c(1, 3, 5)  
  
vector[index]
```

2 4 6

In [6]:

```
# Use the index system to re-order  
vector[c(5,4,3,2,1)]
```

6 5 4 3 2

Use the names

In [7]:

```
# Recall that we could give names to vector entries
names(vector) <- letters[2:6]; vector
```

```
b
2
c
3
d
4
e
5
f
6
```

In [8]:

```
vector["b"]
```

```
b: 2
```

Use "," to separate dimensions.

- 1st dimension: row
- 2nd dimension: column
- 3rd ...

3.1.2 Index a matrix

In [9]:

```
matrix <- matrix(c(3:14), nrow = 4, byrow = TRUE)
print(matrix)
# Note that the indices are given.
```

```
      [,1] [,2] [,3]
[1,]     3     4     5
[2,]     6     7     8
[3,]     9    10    11
[4,]    12    13    14
```

In [10]:

```
matrix[2, 3]
```

8

In [11]:

```
matrix[2, ]
```

6 7 8

In [12]:

```
matrix[ , c(1, 3)]
```

A matrix:

4 × 2 of

type int

3 5

6 8

9 11

12 14

In [13]:

```
# Change the order of columns.
```

```
matrix[ , c(3, 1)]
```

A matrix:

4 × 2 of

type int

5 3

8 6

11 9

14 12

Use the names

In [14]:

```
rownames(matrix)
```

NULL

In [15]:

```
# Recall that we could give names to columns and rows
```

```
row.names <- c("row1", "row2", "row3", "row4")
col.names <- c("col1", "col2", "col3")
rownames(matrix) <- row.names
colnames(matrix) <- col.names
print(matrix)
rownames(matrix)
```

	col1	col2	col3
row1	3	4	5
row2	6	7	8
row3	9	10	11
row4	12	13	14

'row1' 'row2' 'row3' 'row4'

In [16]:

```
matrix["row1", ]
# The output is a named vector as a result of dimension reduction
```

col1

3

col2

4

col3

5

In [17]:

```
matrix["row2", "col3"]
```

8

3.1.3 Index an array

In [18]:

```
array <- array(3:14, dim = c(2, 3, 2))  
print(array)
```

, , 1

	[,1]	[,2]	[,3]
[1,]	3	5	7
[2,]	4	6	8

, , 2

	[,1]	[,2]	[,3]
[1,]	9	11	13
[2,]	10	12	14

In [19]:

```
array[ , , 1]
```

A matrix:

2 × 3 of

type int

3 5 7

4 6 8

In [20]:

```
array[2, 3, 2]
```

14

In [21]:

```
array[1, , 2]
```

9 11 13

3.1.4 Index a data frame

In [22]:

```
df <- data.frame(names = c("Lucy", "John", "Mark", "Candy"),  
                  score = c(67, 56, 87, 91))
```

In [23]:

```
print(df)
```

```
  names score
1  Lucy   67
2  John   56
3  Mark   87
4  Candy   91
```

In [24]:

```
df[2, ]
```

A data.frame: 1 × 2

	names	score
	<fct>	<dbl>
2	John	56

In [25]:

```
df[ , 1]
```

Lucy John Mark Candy

► **Levels:**

Use the names

In [26]:

```
# There are (column) names that are ready to use in data frames.
names(df)
```

'names' 'score'

In [27]:

```
df$names
# data.frame$variable.name gives the variable.
```

Lucy John Mark Candy

► **Levels:**

Use conditions in the index

In [28]:

```
vector  
vector[vector>4]
```

b

2

c

3

d

4

e

5

f

6

e

5

f

6

In [29]:

```
vector>4
```

b

FALSE

c

FALSE

d

FALSE

e

TRUE

f

TRUE

In [30]:

```
# What is John's score?
df[df$names == "John",]
```

A data.frame: 1 × 2

	names	score
	<fct>	<dbl>
2	John	56

In [31]:

```
# How does this work?
df$names == "John"
```

FALSE TRUE FALSE FALSE

In [32]:

```
# Anyone scored 100?
print(df[df$score == 100,])
```

[1] names score
<0 rows> (or 0-length row.names)

In [33]:

```
# Highest score?
max(df$score) # max() for maximum
```

91

In [34]:

```
# Who had the highest score?
df[df$score == max(df$score), ]
```

A data.frame: 1 × 2

	names	score
	<fct>	<dbl>
4	Candy	91

In [35]:

```
# Note that this is still a data frame.
str(df[df$score == max(df$score), ])
```

```
'data.frame':  1 obs. of  2 variables:
 $ names: Factor w/ 4 levels "Candy","John",...: 1
 $ score: num 91
```

In [36]:

```
# I only need the name.
df[df$score == max(df$score), ]$names
```

Candy

► **Levels:**

In [37]:

```
# Change the order of columns
df[ , c("score", "names")]
# By now you should have realized that,
# we change the order of columns by picking the columns
# in the order that we want.
```

A data.frame: 4 ×

2

score	names
<dbl>	<fct>
67	Lucy
56	John
87	Mark
91	Candy

3.1.5 Index a list

In [38]:

```
list <- list("Red", factor(c("a", "b")), c(21, 32, 11), TRUE)
print(list)
```

```
[[1]]
[1] "Red"
```

```
[[2]]
[1] a b
Levels: a b
```

```
[[3]]
[1] 21 32 11
```

```
[[4]]
[1] TRUE
```

In [39]:

```
list[[1]]
```

```
'Red'
```

In [40]:

```
list[[3]][2]
```

```
32
```

3.2 Enter or import data into R

Here we talk about importing data frames.

3.2.1 Direct data entering

Recall the `data.frame()` function. See the first code chunk of this lecture.

3.2.2 Use datasets that come with R or R packages

Many R packages come with datasets that help explain how the packages and functions work, including those already installed when you download R and those already loaded everytime you open R.

In [41]:

```
head(mtcars) # You can use this dataset directly whenever you want.
```

A data.frame: 6 × 11

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

In [42]:

```
# data() # Shows all datasets in base R.
```

Some require loading the package, e.g. "survival" package has a demo data "cancer".

In [43]:

```
# head(cancer)
# Load 'cancer' data before loading the 'survival' package will result in error.
library(survival)
head(cancer)
```

A data.frame: 6 × 10

inst	time	status	age	sex	ph.ecog	ph.karno	pat.karno	meal.cal	wt.loss
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
3	306	2	74	1	1	90	100	1175	NA
3	455	2	68	1	0	90	90	1225	15
3	1010	1	56	1	0	90	90	NA	15
5	210	2	57	1	1	90	60	1150	11
1	883	2	60	1	0	100	90	NA	0
12	1022	1	74	1	1	50	80	513	0

3.2.3 Read data files

RStudio allows you to do everything in this section by clicking!!!

It is necessary to import data into R before we start working on our analysis. R offers a wide range of packages for importing data in any format.

1. For **.txt** and **.csv** files by default: `read.table()`, `read.csv()`, `read.csv2()`, `read.delim()` and `read.delim2()`.
2. Packages are needed to read files from **Excel**, **SPSS**, **SAS**, **Stata**, and various relational databases.

1. For .txt and .csv files

In [44]:

```
# ?read.table      # Uncomment to run the code
```

Example command

```
data <- read.table(file, header = TRUE, sep = "", quote = "\"", dec = ".", fill = TRUE, comment.char = "")
```

- file: A local **file** with complete path or a **URL**
- header: Whether use the first row as the names of the columns
- sep: What separates the entries, by default:
 - `read.table()`: white spaces, one or more
 - `read.csv()`: ,
 - `read.csv2()`: ;
 - ...
- ...

Data from Dr. Hanley's teaching website.

<http://www.medicine.mcgill.ca/epidemiology/hanley/bios602/MultilevelData/otitisDataTall.txt>
(<http://www.medicine.mcgill.ca/epidemiology/hanley/bios602/MultilevelData/otitisDataTall.txt>)

In [45]:

```
x <- read.csv(file = "http://www.medicine.mcgill.ca/epidemiology/hanley/bios602/
MultilevelData/otitisDataTall.txt",
              header = FALSE)
dim(x)
```

In [46]:

```
head(x) # head() Displays the first 6 (default) rows.
```

A data.frame: 6 × 3

V1	V2	V3
<fct>	<fct>	<fct>
family	proportion	zygosity
1	0	2
1	0.04646	2
2	0.05162	2
2	0	2
3	0	2

In [47]:

```
xx <- read.csv(file = "http://www.medicine.mcgill.ca/epidemiology/hanley/bios602/MultilevelData/otitisDataTall.txt",
               header = TRUE)
dim(xx)
```

257 3

In [48]:

```
head(xx) # Note that "header = TRUE" makes the first row column names.
```

A data.frame: 6 × 3

family	proportion	zygosity
<int>	<dbl>	<int>
1	0.00000	2
1	0.04646	2
2	0.05162	2
2	0.00000	2
3	0.00000	2
3	0.09130	2

For local files, we need to give the complete path to the file.

```
data <- read.csv(file = "~/Desktop/PhD3/Teaching/EPIB613/2018/classlist.csv", header = TRUE)
```

Or, set working directory to that folder

```
setwd("~/Desktop/PhD3/Teaching/EPIB613/2018")
```

```
data <- read.csv("classlist.csv", header = TRUE)
```

2. For Excel, SAS, SPSS, Stata, etc. files, Google!

- There are a lot of packages.
- Read the help files of the package/function you use.
- Check the data before moving on.

There are also a lot of tutorials online.

<https://www.datacamp.com/community/tutorials/r-data-import-tutorial>
(<https://www.datacamp.com/community/tutorials/r-data-import-tutorial>)

But you still need to google every time. Trust me.

(Optional) Something that I learned before I started using RStudio

In [49]:

```
# d <- read.csv(file.choose())
```

Bottom line - You can always click in RStudio, and if necessary, copy the code to your script for reproducibility.

3.3 Export data

Similar to reading data:

- For .txt and .csv files by default: write.table(), write.csv(), write.csv2().
- Packages are needed to write files to Excel, SPSS, SAS, Stata, and various relational databases.
 - The packages that read these file types usually also have functions that write to these file types.

In [50]:

```
df
write.csv(df, file = "~/Desktop/df.csv")
```

A data.frame: 4 ×

2

names	score
<fct>	<dbl>
Lucy	67
John	56
Mark	87
Candy	91

3.4 Save and load data in R

RStudio allows you to do everything in this section by clicking!!!

- Two functions: **save()** and **load()** allows saving and loading R workspace image.
 - Saving workspace image will create a .RData file in your working directory.
 - Your current work is saved.
- Yes I said do NOT save workspace images last class.
 - Unless you are working with a 5GB dataset that takes 30 minutes to load into R.

3.5 View data

It is very important to check the data immediately after we import it into R.

In [51]:

```
# Check the dimensions of the data frame.
dim(mtcars)
```

32 11

In [52]:

```
# Check the column names
names(mtcars)
```

'mpg' 'cyl' 'disp' 'hp' 'drat' 'wt' 'qsec' 'vs' 'am' 'gear' 'carb'

In [53]:

```
# Or if you remember the function str()
str(mtcars)
```

```
'data.frame':  32 obs. of  11 variables:
 $ mpg  : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl  : num   6  6  4  6  8  6  8  4  4  6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp   : num  110 110  93 110 175 105 245  62  95 123 ...
 $ drat: num   3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt   : num   2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs   : num   0  0  1  1  0  1  0  1  1  1 ...
 $ am   : num   1  1  1  0  0  0  0  0  0  0 ...
 $ gear: num   4  4  4  3  3  3  3  4  4  4 ...
 $ carb: num   4  4  1  1  2  1  4  2  2  4 ...
```

In [54]:

```
# Look at the first few rows, default is 6 rows
head(mtcars, n=10)
```

A data.frame: 10 × 11

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

In [55]:

```
# Check the last few rows, default is 6 rows
tail(mtcars, n = 3)
```

A data.frame: 3 × 11

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Ferrari Dino	19.7	6	145	175	3.62	2.77	15.5	0	1	5	6
Maserati Bora	15.0	8	301	335	3.54	3.57	14.6	0	1	5	8
Volvo 142E	21.4	4	121	109	4.11	2.78	18.6	1	1	4	2

In [56]:

```
# Quick summary of the data frame
summary(mtcars)
```

mpg	cyl	disp	hp
Min. :10.40	Min. :4.000	Min. : 71.1	Min. : 52.0
1st Qu.:15.43	1st Qu.:4.000	1st Qu.:120.8	1st Qu.: 96.5
Median :19.20	Median :6.000	Median :196.3	Median :123.0
Mean :20.09	Mean :6.188	Mean :230.7	Mean :146.7
3rd Qu.:22.80	3rd Qu.:8.000	3rd Qu.:326.0	3rd Qu.:180.0
Max. :33.90	Max. :8.000	Max. :472.0	Max. :335.0
drat	wt	qsec	vs
Min. :2.760	Min. :1.513	Min. :14.50	Min. :0.0000
1st Qu.:3.080	1st Qu.:2.581	1st Qu.:16.89	1st Qu.:0.0000
Median :3.695	Median :3.325	Median :17.71	Median :0.0000
Mean :3.597	Mean :3.217	Mean :17.85	Mean :0.4375
3rd Qu.:3.920	3rd Qu.:3.610	3rd Qu.:18.90	3rd Qu.:1.0000
Max. :4.930	Max. :5.424	Max. :22.90	Max. :1.0000
am	gear	carb	
Min. :0.0000	Min. :3.000	Min. :1.000	
1st Qu.:0.0000	1st Qu.:3.000	1st Qu.:2.000	
Median :0.0000	Median :4.000	Median :2.000	
Mean :0.4062	Mean :3.688	Mean :2.812	
3rd Qu.:1.0000	3rd Qu.:4.000	3rd Qu.:4.000	
Max. :1.0000	Max. :5.000	Max. :8.000	

In [57]:

```
# Check missing values
sum(is.na(mtcars))
# is.na() is true if a cell is "NA" - missing value
# sum() over all cells tells how many true's there are.
# Recall from Lecture 2.
```