Introduction to R Programming

Slide Set 2: Exploratory Data Analysis with R

Maria Ptashkina

Barcelona GSE ITFD

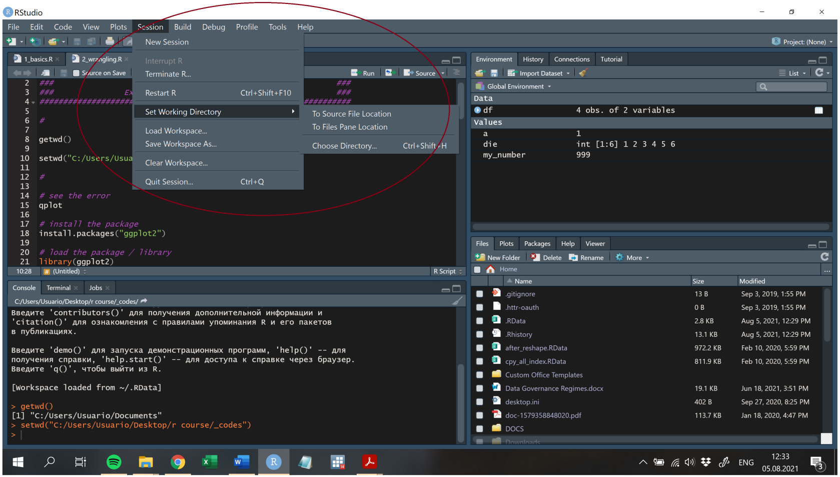
September 2021

Table of Contents

Workflow

- When you start working, you need to keep your files somewhere
- In R you need to declare a working directory
- A better way (which we will use for the applications next week) is using R Projects ▸ Projects
- If you decide to use R for your Master Projects, you'd need to collaborate on the code. For this purpose I recommend doing it with Dropbox and creating R projects ▸ Dropbox and R

Workflow



```
###                                    ###
###                  Ex                ###
####################          ##########

#

getwd()

setwd("C:/Users/Usua

#

# see the error
qplot

# install the package
install.packages("ggplot2")

# load the package / library
library(ggplot2)
```

```
Data
df                 4 obs. of 2 variables
Values
  a                1
  die              int [1:6] 1 2 3 4 5 6
  my_number        999
```

C:/Users/Usuario/Desktop/r course/_codes/

```
Введите 'contributors()' для получения дополнительной информации и
'citation()' для ознакомления с правилами упоминания R и его пакетов
в публикациях.

Введите 'demo()' для запуска демонстрационных программ, 'help()' -- для
получения справки, 'help.start()' -- для доступа к справке через браузер.
Введите 'q()', чтобы выйти из R.

[Workspace loaded from ~/.RData]

> getwd()
[1] "C:/Users/Usuario/Documents"
> setwd("C:/Users/Usuario/Desktop/r course/_codes")
> 
```

Name	Size	Modified
.gitignore	13 B	Sep 3, 2019, 1:55 PM
.httr-oauth	0 B	Sep 3, 2019, 1:55 PM
.RData	2.8 KB	Aug 5, 2021, 12:29 PM
.Rhistory	13.1 KB	Aug 5, 2021, 12:29 PM
after_reshape.RData	972.2 KB	Feb 10, 2020, 5:59 PM
cpy_all_index.RData	811.9 KB	Feb 10, 2020, 5:59 PM
Custom Office Templates		
Data Governance Regimes.docx	19.1 KB	Jun 18, 2021, 3:51 PM
desktop.ini	402 B	Sep 27, 2020, 8:25 PM
doc-1579358848020.pdf	113.7 KB	Jan 18, 2020, 4:47 PM
DOCS		
Downloads		

Packages

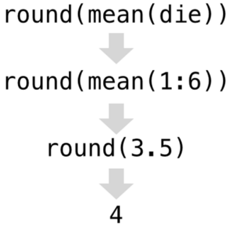
- So far we used ‘base packages’ which are part of `R` source code
- Most of the commands are part of the packages created by `R` users (there are 10,000+ user contributed packages and growing)
- Packages are collections of `R` functions, data, and compiled code in a well-defined format, created to add specific functionality
- You need to install a package only once using `install.packages("nameofpackage")` at the command line
- Then you need to call it in script using `library(nameofpackage)` every time you need to use it in a new `R` session
- If you'd like to use a specific command from a specific package use `package::command`

Getting Help

- Each `R` function comes with its own help page
- Type question mark and name of the function `?sqrt`
- If you forgot the function's name, you can search my keyword using two question marks `??log`
- This is all very formal and nice, but in reality you will literally only use Google and Stack Overflow

Functions

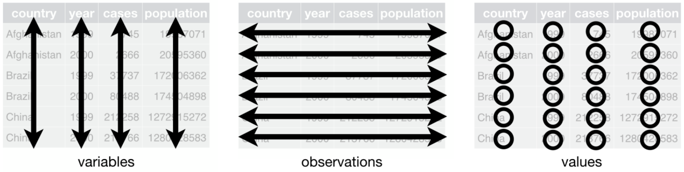
- Before we dive into data analysis, let's refresh a few key concepts
- Remember that `R` works with objects that you create
- The job is to apply functions to objects
- A standard way to work in `R` is to nest functions



```
round(mean(die))
round(mean(1:6))
round(3.5)
4
```

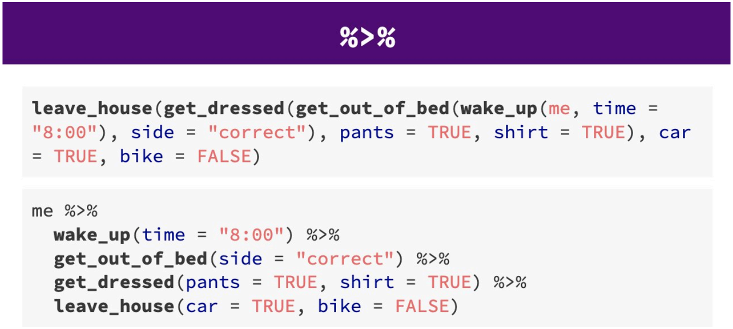
Tidyverse

- Tidyverse is a collection of integrated packages which was designed to make working with data more user friendly
- You can do all the same with nested functions and individual packages, if you prefer
- Tidyverse helps you to clean the data and put it into a format which you would need to build a model / run a regression



Pipes

- Stringing together commands can be difficult, especially if there are many nested functions
- The pipe allows the output of a previous command to be used as input to another command instead of using nested functions
- Shortcut: `Shift+Ctrl+M`



```
leave_house(get_dressed(get_out_of_bed(wake_up(me, time =
"8:00"), side = "correct"), pants = TRUE, shirt = TRUE), car
= TRUE, bike = FALSE)
```

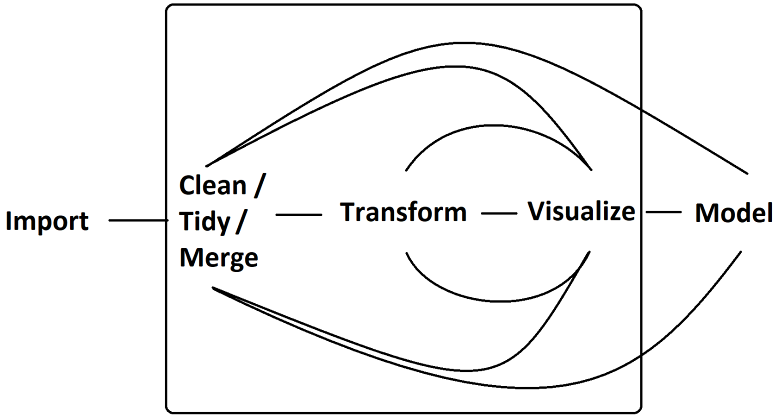
```
me %>%
  wake_up(time = "8:00") %>%
  get_out_of_bed(side = "correct") %>%
  get_dressed(pants = TRUE, shirt = TRUE) %>%
  leave_house(car = TRUE, bike = FALSE)
```

Tibbles

- A core component of the tidyverse is the tibble
- It's basically a modern (more flexible and efficient) version of a data frame
- Tibbles can be created directly using the `tibble()`
- Data frames can be converted into tibbles using `as_tibble(df)`

Table of Contents

Plan



Data types in Tibbles

- `int` stands for integers
- `dbl` stands for doubles, or real numbers
- `chr` stands for character vectors, or strings
- `dttm` stands for date-times (a date + a time)
- `lgl` stands for logical, vectors that contain only TRUE or FALSE
- `fctr` stands for factors, which R uses to represent categorical variables with fixed possible values
- `date` stands for date

dplyr Basics

- The vast majority of your data manipulation challenges are going to be handled using `dplyr` package (part of tidyverse):
  - Pick observations by their values `filter()`
  - Reorder the rows `arrange()`
  - Pick variables by their names `select()`
  - Create new variables with functions of existing variables `mutate()`
  - Collapse many values down to a single summary `summarise()`
- These can all be used in conjunction with `group_by()` which changes the scope of each function

Filter

- `filter`: subset observations based on their values
- Use logical operators we saw before
- Missing values: represented in `R` as `NA`
  - Missing values are "contagious": almost any operation involving an unknown value will also be unknown
  - Check for `NA` using `is.na()`
  - Note! `R` treats missing values differently from other statistical programs
    ▸ More Information
- As we discussed, the majority of tidyverse has equivalents in base `R`
  - `filter()` is equivalent to `subset()` ▸ Difference

Arrange

- `arrange()`: changes the order or rows
- Missing values are always sorted at the end
- As before you can use an equivalent base `R` command `order` (note how much less intuitive the syntax is) ▸ Order

Select

- `select()`: select the columns you need
- Helper functions
  - `starts_with("abc")`: matches names that begin with “abc”
  - `ends_with("xyz")`: matches names that end with “xyz”
  - `contains("ijk")`: matches names that contain “ijk”
  - `num_range("x", 1:3)`: matches x1, x2 and x3
- `rename()` is a variant of select()
- `select()` with `everything()` helper to move columns to the beginning
- Remember we were sub-setting vectors and matrices and data frames? You can do exactly the same as select with base `R` ▸ Sub-setting

Mutate

- `mutate()`: add new columns that are functions of existing columns
- If you only want to keep the new variables, use `transmute()`
- The function must be vectorised
- Frequently used functions
  - Arithmetic operators $+, -, *, /, \hat{}$
  - Logarithmic transformation
  - Offsets `lead()` and `lag()`
  - Cumulative and rolling aggregates
  - Logical comparisons, $<, <=, >, >=, != , ==$

Summarize

- `summarise()`: collapse a data frame to a single row generating a statistic
- Usually combined with `group_by()`
- Frequently used functions
  - Measures of location: mean, median, etc.
  - Measures of spread: standard deviation, median absolute deviation, etc.
  - Measures of rank: min, max, etc.
  - Counts: size of a particular group, number of unique values
  - Counts and proportions of logical values (when used with numeric functions, `TRUE` is converted to 1 and `FALSE` to 0)

Table of Contents

Importing Data

Data type	Extension	Function	Package
Coma separated values	csv	read.csv()	base
		read_csv()	readr (tidyverse)
Other delimited formats	txt	read.table()	base
		read_table()	readr (tidyverse)
		read_delim()	readr (tidyverse)
Excel	xlsx, xls	read_excel()	readxl (tidyverse)
Stata version 13 and above	dta	read_dta()	haven
Stata version 7-12	dta	read.dta()	foreign
SPSS	sav	read.spss()	foreign

It's advised to use `tidyverse` equivalents, because the functions

- Are x10 times faster
- Don't convert character vectors to factors
- Are more reproducible

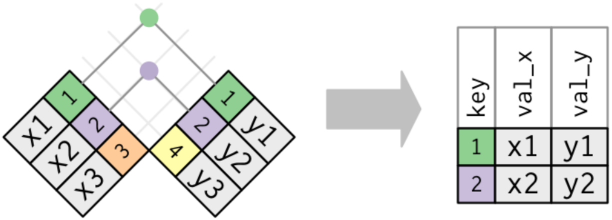
Exporting Data

- Similarly, you can export the data to other formats, using `write.table`, `write.csv` or `write_csv`, and `write.dta`
- You can also save data into `R` data format: `RDATA` and `RDS`
- You can read about other ways of exporting data ▸ Exporting

Merging / Joining / Relational Data

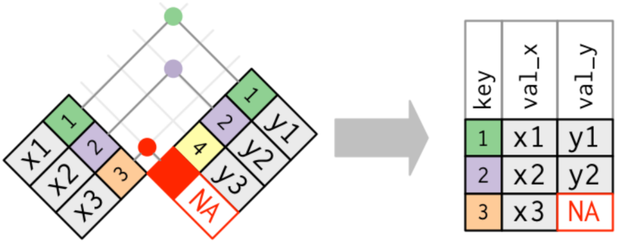
- Keys: the variables used to connect each pair of tables and *uniquely* identifies an observation
    - Primary key: uniquely identifies an observation in its own table
    - Foreign key uniquely identifies an observation in another table
- Relations are typically one-to-many or many-to-one, but sometimes are one-to-one
- Types of joins
    - Inner join
    - Outer joins: left, right, full

Inner Join



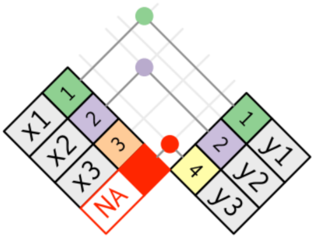
key	val_x	val_y
1	x1	y1
2	x2	y2

Left Join



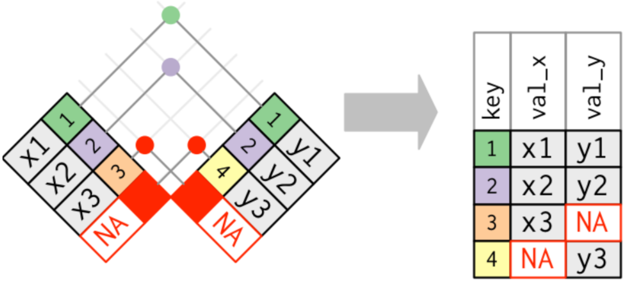
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Right Join



key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

Full Join



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3

Defining Key Columns

- If your data has many levels of variation, you might need to join by several variables
- By default, `by = NULL` uses all variables that appear in both tables
- You can define only some variables using a character vector
- You can also match variable *a* in table *x* to variable *b* in table *y*

Comparing with Base R

dplyr	base
inner_join(x, y)	merge(x, y)
left_join(x, y)	merge(x, y, all.x = TRUE)
right_join(x, y)	merge(x, y, all.y = TRUE)
full_join(x, y)	merge(x, y, all.x = TRUE, all.y = TRUE)

- In base R you can also merge by character vectors using `by.x=c()` and `by.y=()`

Filtering joins

- A handy tool for data inspection is to join by observations as opposed to variables
  - `semi_join(x, y)` keeps all observations in x that have a match in y
  - `anti_join(x, y)` drops all observations in x that have a match in y
- Anti-joins are useful for diagnosing join mismatches
- Be aware that in your research the data might be much nastier, so explore it carefully before!

References and Resources

- Introduction to R Flipped ▸ Tutorial
- Hands-On Programming with R ▸ Tutorial
- R for Data Science ▸ Tutorial
- Tidyverse in R – Complete Tutorial ▸ Tutorial
- Data Wrangling with Tidyverse ▸ Tutorial
- Pipes ▸ Pipes
- Parsing a File (chapter 11.4) ▸ Parsing
- Getting Started with the Tidyverse: Tutorial ▸ Parsing