

SOCIAL DATA SCIENCE

INTRODUCTION TO R

Sebastian Barfort

August 07, 2016

University of Copenhagen
Department of Economics

Course Description

*The objective of this course is to learn how to **analyze**, **gather** and **work** with modern quantitative social science data.*

We will do this using a program called R.

Advantages

Free, open source statistical programming language

Offers a massive set of packages for statistical modelling, machine learning, visualisation, and importing and manipulating data

An enthusiastic community (Stackoverflow, R-help mailing list)

Used by New York Times, Facebook, Google, Twitter...

Disadvantages

R is not perfect

R is not always the best tool for everything

R works for small/medium sized data

Today

1. Arithmetic operations
2. Creating objects
3. Installing packages
4. Importing data
5. Functions
6. Getting help

Hadley Wickham

The bad news is that when ever you learn a new skill you're going to suck. It's going to be frustrating. The good news is that is typical and happens to everyone and it is only temporary. You can't go from knowing nothing to becoming an expert without going through a period of great frustration and great suckiness.

Kosuke Imai

One can learn data analysis only by doing, not by reading.

Do not use the console, write scripts instead

Be lazy (write functions)

Think before you code

Code is a medium of communication

1. Between you and the computer
2. Between you and other people (or future you)

DEAR FUTURE SELF,

YOU'RE LOOKING AT THIS FILE BECAUSE
THE PARSE FUNCTION FINALLY BROKE.

IT'S NOT FIXABLE. YOU HAVE TO REWRITE IT.
SINCERELY, PAST SELF

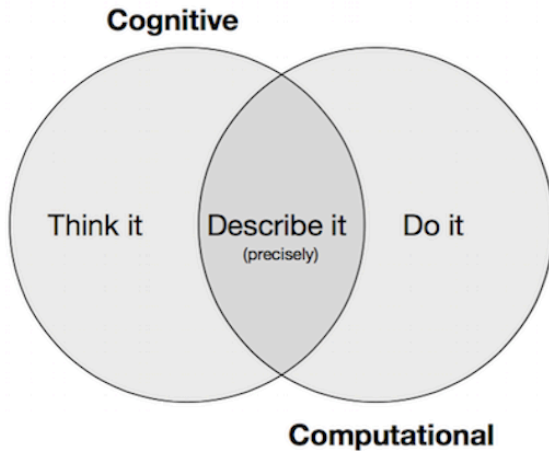
| DEAR PAST SELF, IT'S KINDA
| CREEPY HOW YOU DO THAT.

ALSO, IT'S PROBABLY AT LEAST |
2013. DID YOU EVER TAKE
THAT TRIP TO ICELAND? |

STOP JUDGING ME!



TWO TYPES OF CONSTRAINTS



Arithmetic in R

You can use R to do standard arithmetic operations

```
1 + 100
```

```
## [1] 101
```

```
7 / 2
```

```
## [1] 3.5
```

```
sqrt(3)
```

```
## [1] 1.732051
```

Objects

R Rules

Everything has a **name**

Everything is an **object**

Every object has a **class**

OBJECTS

R stores all information as an **object** with a **name** of our choice

Objects are created using an assignment operator (<- or =)

```
y = "welcome to social data science"  
y
```

```
## [1] "welcome to social data science"
```

```
class(y)
```

```
## [1] "character"
```

We create and manipulate objects by feeding them to **functions** and getting output back as a result.

```
x = c(1, 3, 100)
class(x)
```

```
## [1] "numeric"
```

```
x * 2
```

```
## [1] 2 6 200
```

```
y * 2
```

```
## Error in y * 2: non-numeric argument to binary operator
```

There is no agreement about how to name things, so you'll likely see a mixture of snake_case and CamelCase, based on the preferences of the person who originally wrote some function

Figure out where you are

```
getwd()
```

```
## [1] "/Users/sbarfort/git/sds_summer/_slides"
```

Like in **Unix**, in R you are always in a directory

Your actions are all relative to that directory

Examples of R objects

- character string (e.g. words)
- number
- vector
- matrix
- data frame
- list

We verify the class of an object using the `class` function

Question: What is the class of the objects given below?

```
z = "text"  
p = c(1, 3, 5)  
q = 2  
y = NA  
k = FALSE
```

- NA: not available, missing (`is.na`)
- NULL: undefined (`is.null`)
- TRUE: logical true (`isTRUE`)
- FALSE: logical false (`!isTRUE`)

Operator	Meaning
<	less than
>	greater than
==	equal to
<=	less than or equal to
>=	greater than or equal to
!=	not equal to
a b	a or b
a & b	a and b

The most basic type of R object is a vector.

There is really only one rule about vectors in R, which is that a vector can only contain objects of the same class.

We create vectors using the `c` (concatenate/combine) function

```
my_vector = c(1, 3, 5, 10)
another_vector = 1:100
a_third_vector = c("yes", "no", "hello")
my_logical_vector = c(TRUE, FALSE, FALSE, TRUE)
```

R stores spreadsheet like data in a **data frame**

These are really collections of vectors of the same length

Tip: Create data frames whenever you can

We select variables using \$, known as the component selector

You can also call variables/observations using indexing

We can select the first row and the first column

```
df[1, 1]
```

Select the entire first column

```
df[, 1]
```

Select the second row

```
df[2, ]
```

Some useful functions for working with data frames:

- **names**: returns the column names of the data frame
- **rownames**: returns the row names (if any) of the data frame
- **summary**: returns summary statistics
- **head**: returns the first 5 or 10 observations of the data frame

Installing Packages

On its own, R can't do all that much

To really make use of R's capabilities, we need packages

A package bundles together code, data, documentation, and tests

We install packages from two sources

- the Comprehensive R Archive Network (CRAN)
- github

We can install the `readr` package, for example, by running

```
install.packages("readr")
```

Afterwards, we can access all the functions available in the package by running

```
library("readr")
```

INSTALLING PACKAGES FROM GITHUB

It's slightly more difficult to install from github since we need to load a package from CRAN first: `devtools`

Installing from Github now looks like

```
library("devtools")  
install_github("hadley/purrr")
```

The `purrr` package can now be loaded using the `library` command

```
library("purrr")
```

Importing Data

There is a new package that reads almost all file formats: `rio`

```
install.packages("rio")
```

Only two functions: `import` and `export`

Base R includes functions for reading flat files: `read.csv`, `read.table`, etc.

But I suggest using them only if `rio` fails

They are slower and have bad defaults (`stringsAsFactors = TRUE`)

Functions

FUNCTIONS

Functions operate on objects

R has many built in functions such as `summary`, `mean`, `table`, etc

```
x = 1:10  
mean(x)
```

```
## [1] 5.5
```

```
sd(x)
```

```
## [1] 3.02765
```

```
median(x)
```

```
## [1] 5.5
```

```
my_function = function(input1, input2, ..., inputN)
{
  # define 'output' using input1,...,inputN
  return(output)
}
```

Getting Help

If you know the command, type `?` followed by the function in the console

```
?summary
```

Search your version of R using `??` followed by the function name

Use Google and Stackoverflow

Exercises

You will work in groups on Exercise 1.5.1 from Imai (2016).

The data is available [here](#)

You can read the data as follows

```
library("rio")  
filepath = "https://raw.githubusercontent.com/  
          kosukeimai/qss/master/INTRO/  
          turnout.csv"  
df = import(filepath)
```

QUESTIONS

1. Load the data into **R** and check the dimensions of the data. Also, obtain a summary of the data. How many observations are there? What is the range of years covered in this data set?
2. Calculate the turnout rate based on the voting age population or **VAP**. Note that for this data set, we must add the total number of eligible overseas voters since the **VAP** variable does not include these individuals in the count. Next, calculate the turnout rate using the voting eligible population or **VEP**. What difference do you observe?
3. Compute the difference between **VAP** and **ANES** estimates of turnout rate. How big is the difference on average? What is the range of the difference? Conduct the same comparison for the **VEP** and **ANES** estimates of voter turnout. Briefly comment on the results.