R basics

Oleksii Yehorchenkov, Ph.D.

Part I. R Basics

What is R?

R is a dialect of the S language.

R Philosophy

In “Stages in the Evolution of S”, John Chambers writes:

“[W]e wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important.”

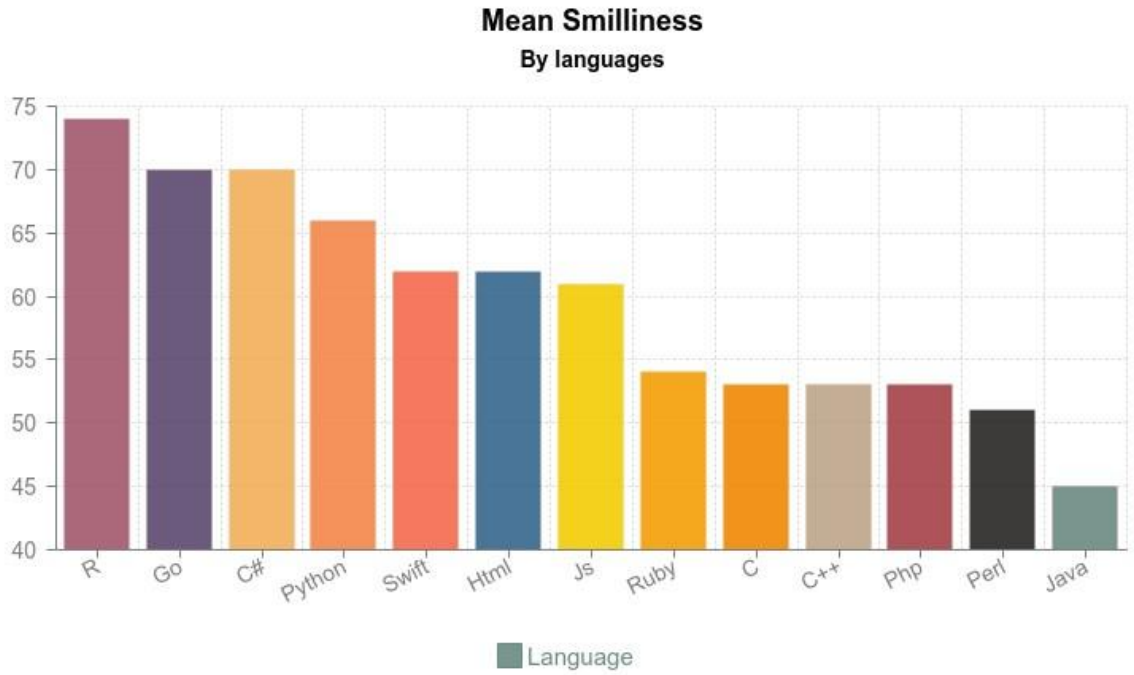
What is R?

- R is a free language and environment for statistical computing and graphics
- R is an interpreted language, not a compiled one, meaning that all commands typed on the keyboard are directly executed without requiring to build a complete program (this is like Python and unlike C, Fortran, Pascal, etc.)
- R has existed for over 25 years
- R is modular — most functionality is from add-on packages. So the language can be thought of as a *platform* for creating and running a large number of useful packages.

Why use R?

- The most popular software for data analysis
- Extremely flexible: can be used to manipulate, analyze, and visualize any kind of data
- Cutting edge statistical tools
- Publication quality graphics
- 15,000+ add on packages covering all aspects of statistics and machine learning
- Active community of users

Every programmer has a language he doesn’t like, so much so he can’t even smile for his profile photo. Face API measures the amount that you are smiling, using a value between 0 and 1



https://medium.com/swlh/what-programming-language-has-the-happiest-developers-f0636b08e898

Syntax rules

- R is case sensitive
- R ignores white space
- Variable names should start with a letter (A-Z and a-z) and can include letters, digits (0-9), dots (.), and underscores (_)
- Comments can be inserted using a hash # symbol
- Functions must be written with parentheses, even if there is nothing within them; for example: ls()

Objects

R has five basic or “atomic” classes of objects:

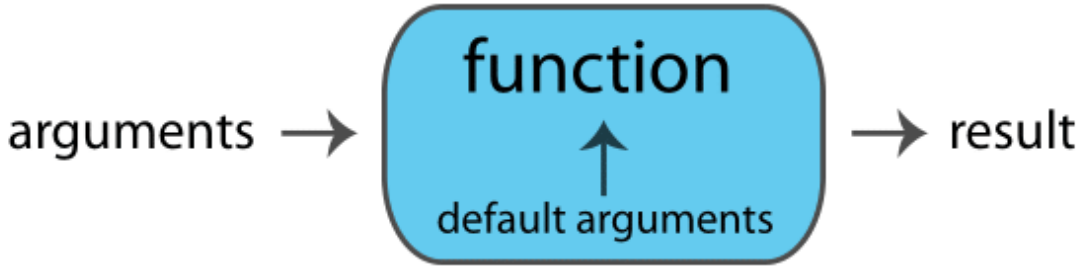
- character
- numeric (real numbers)
- integer
- complex
- logical (True/False)

The most basic object is a vector

- A **vector** can only contain objects of the same class
- BUT: The one exception is a **list**, which is represented as a vector but can contain objects of different classes (indeed, that’s usually why we use them)

Function calls

Functions perform actions — they take some input, called arguments and return some output (i.e., a result). Here's a schematic of how a function works




```
round(x = 2.34, digits = 1) # match by name
## [1] 2.3
```

```
round(2.34, 1) # match by position
## [1] 2.3
```

Assignment

Objects (data structures) can be assigned names and used in subsequent operations:

- The **gets** <- operator (less than followed by a dash) is used to save objects
- The name on the left **gets** the object on the right

```
sqrt(10) # calculate square root of 10; result is not stored anywhere

## [1] 3.162278

x <- sqrt(10) # assign result to a variable named x
```

Asking for help

1. You can ask R for help using the **help** function, or the **?** Shortcut

```
help(help)

?help

?sqrt
```

The help function can be used to look up the documentation for a function, or to look up the documentation to a package. We can learn how to use the stats package by reading its documentation like this:

```
help(package = "stats")
```

2. If you know the name of the package you want to use, then Googling “R package-name” will often get you to the documentation.

Installing & using packages

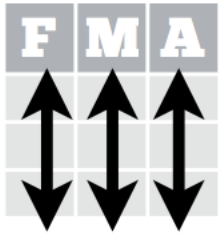
R is a modular environment that is extended by the use of packages. Packages are collections of functions or commands that are designed to perform specific tasks (e.g., fit a type of regression model). A large number of contributed packages are available (> 15,000).

Using an R package is a two step process:

- Install the package onto your computer using the **install.packages()** function. This only needs to be done the first time you use the package.
- Load the package into your R session's search path using the **library()** function. This needs to be done each time you use the package.

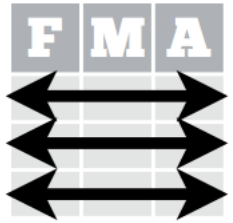
The tidyverse

In a tidy data set:



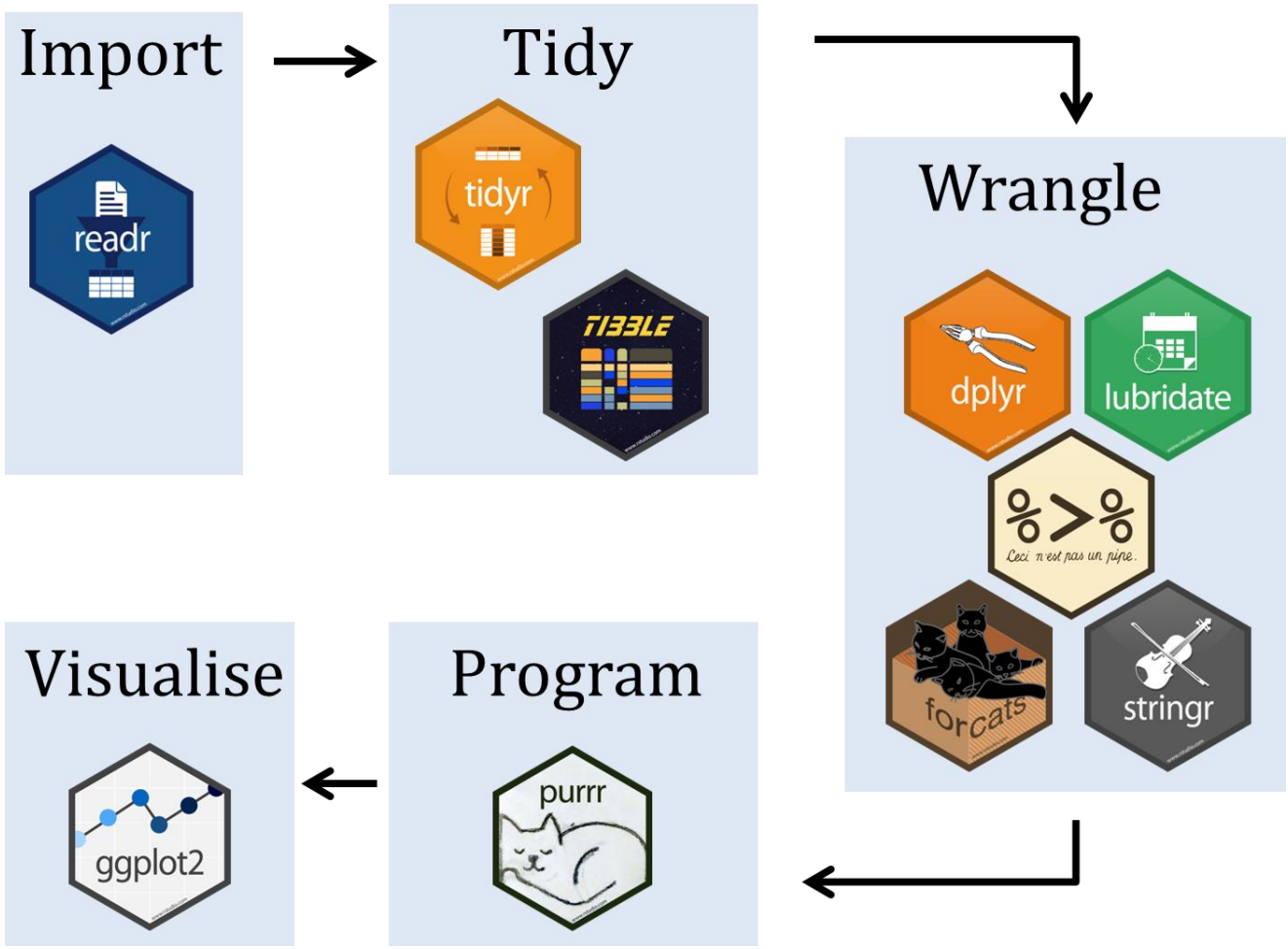
Each **variable** is saved in its own **column**

&



Each **observation** is saved in its own **row**

The tidyverse



Data Frames

- Data frames are used to store tabular data
- They are represented as a special type of list where every element of the list has to have the same length
- Each element of the list can be thought of as a column and the length of each element of the list is the number of rows
- Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element be the same class
- Data frames also have a special attribute called row.names
- Data frames are usually created by calling read.table () or read.csv()
- Can be converted to a matrix by calling data.matrix

How to read data from a file

To read data from a file, you have to know what kind of file it is. The table below lists functions from the readr package, which is part of tidyverse, that can import data from common plain-text formats.

- **read_csv()** reads comma delimited files, **read_csv2()** reads semicolon separated files (common in countries where , is used as the decimal place), **read_tsv()** reads tab delimited files, and **read_delim()** reads in files with any delimiter.
- **read_fwf()** reads fixed width files. You can specify fields either by their widths with **fwf_widths()** or their position with **fwf_positions()**. **read_table()** reads a common variation of fixed width files where columns are separated by white space.

How to read data from a file

```
# read data to dataframe “df”

df <- read_csv("dataSets/states.csv")

# show the six first rows
head(df)
# A tibble: 6 x 21
  state region    pop   area density metro waste energy miles toxic
  <chr> <chr>   <dbl>  <dbl>   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
1 Alab~ South  4.04e6  52423  77.1    67.4 1.11     393 10500 27.9
2 Alas~ West   5.50e5 570374   0.960  41.1 0.910    991  7200 37.4
3 Ariz~ West   3.66e6 113642  32.2    79   0.790    258  9700 19.6
4 Arka~ South  2.35e6  52075  45.2    40.1 0.850    330  8900 24.6
5 Cali~ West   2.98e7 155973 191.     95.7 1.51     246  8700  3.26
6 Colo~ West   3.29e6 103730  31.8    81.5 0.730    273  8300  2.25
# ... with 11 more variables: green <dbl>, house <dbl>, senate <dbl>,
#   csat <dbl>, vsat <dbl>, msat <dbl>, percent <dbl>, expense <dbl>,
#   income <dbl>, high <dbl>, college <dbl>
```