PS10: introduction to R

UL HPC School 2017

Joseph Emeras, Aurélien Ginolhac

13/06/2017

What is R?

R is shorthand for "GNU R":

- An interactive programming language derived from **S** (**J. Chambers**, Bell Lab, 1976)
- Appeared in 1993, created by **R. Ihaka** and **R. Gentleman**, University of Auckland
- Focus on data analysis and plotting
- **R** is also shorthand for the ecosystem around this language
  - Book authors
  - Package developers
  - Ordinary useRs

Learning to use **R** will make you **more efficient** and **facilitate the use** of advanced data analysis tools

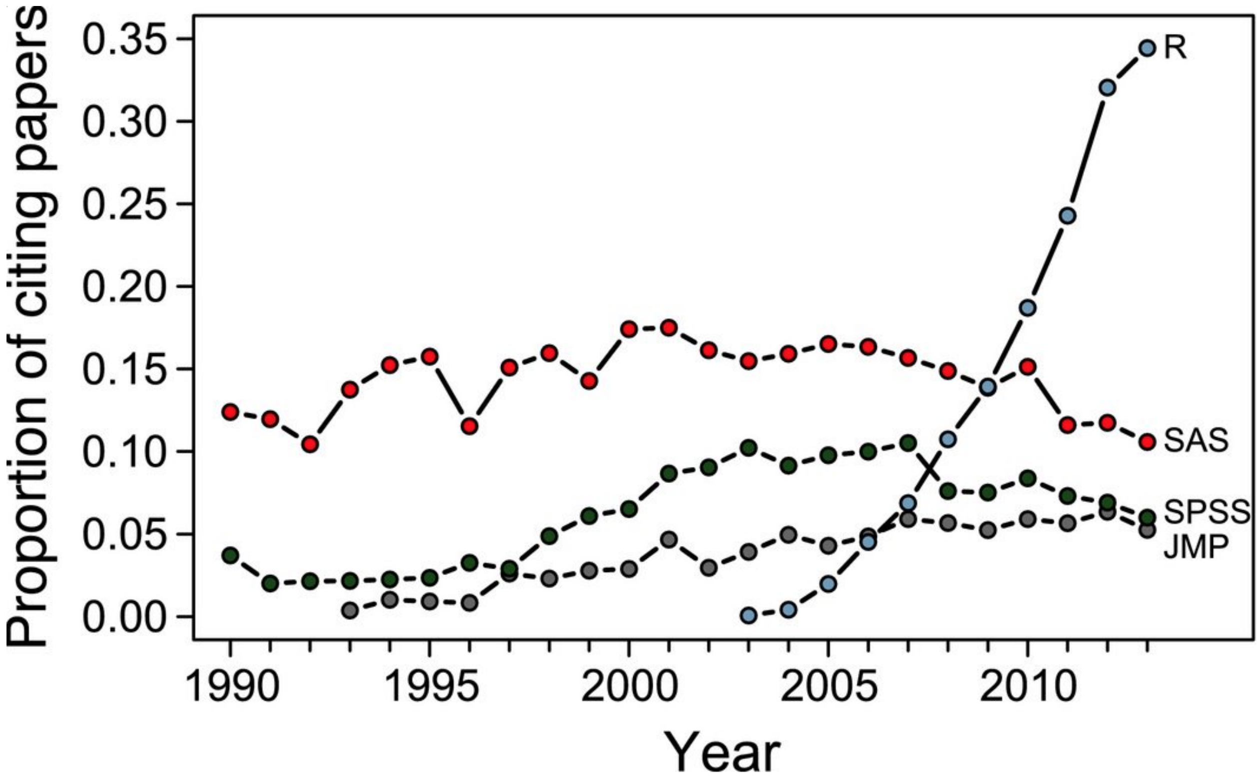
Why use R?

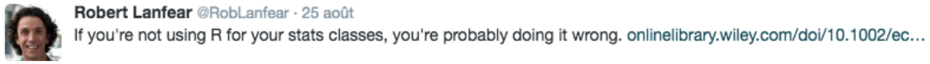
- It's *free!*
- easy to install / maintain
- easy to process big files and analyse huge amounts of data
- integrated data visualization tools, *even dynamic* via shiny
- fast, and even faster with *C++* integration via Rcpp.
- easy to get help
  - huge R community in the web
  - stackoverflow with a lot of tags like **r**, **tidyverse**, **dplyr**, **ggplot2** etc.
  - rbloggers

Twitter R community

`#rstats` on twitter

Constant trend





Robert Lanfear @RobLanfear · 25 août

If you're not using R for your stats classes, you're probably doing it wrong. onlinelibrary.wiley.com/doi/10.1002/ec…

From Touchon & McCoy. *Ecosphere*. 2016

Packages

+10,000 in Jan 2017

CRAN

reliable: package is checked during submission process

typical install:

```r
install.packages("ggplot2")
```

MRAN for Windows users

bioconductor

dedicated to biology. status

typical install:

```r
source("https://bioconductor.org/biocLite.R"
)
biocLite("limma")
```

GitHub

easy install thanks to devtools. status

```r
# install.packages("devtools")
devtools::install_github("tidyverse/readr")
```

loading packages

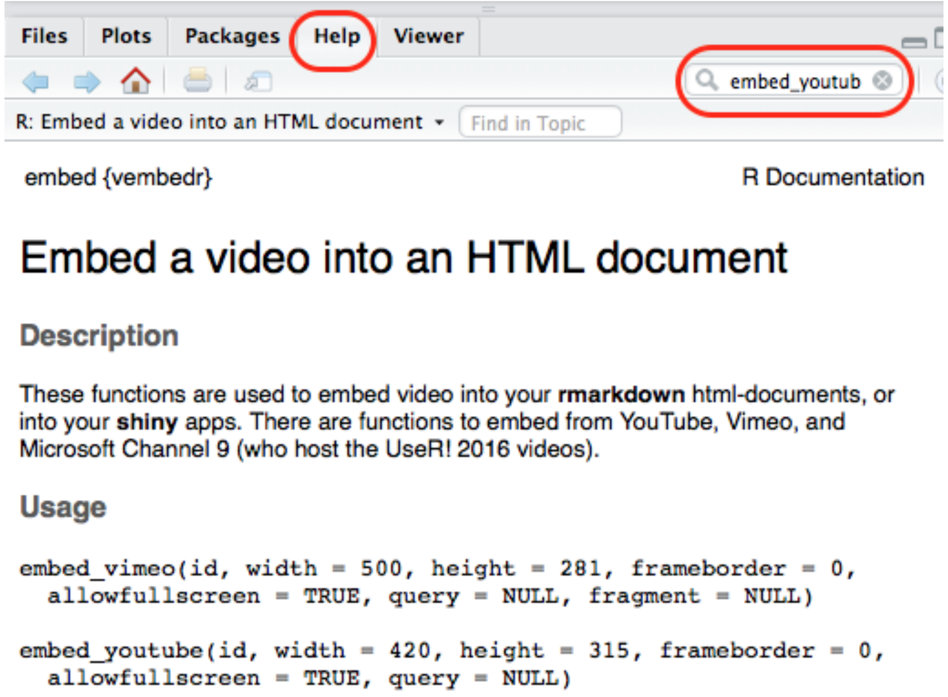
```r
library(ggplot2)
```

Getting help

2 possibilities for manual pages.

```
?log
help(log)
```

In **Rstudio**, the help page can be viewed in the bottom right panel



embed {vembedr} R Documentation

Embed a video into an HTML document

Description

These functions are used to embed video into your **rmarkdown** html-documents, or into your **shiny** apps. There are functions to embed from YouTube, Vimeo, and Microsoft Channel 9 (who host the UseR! 2016 videos).

Usage

```
embed_vimeo(id, width = 500, height = 281, frameborder = 0,
  allowfullscreen = TRUE, query = NULL, fragment = NULL)

embed_youtube(id, width = 420, height = 315, frameborder = 0,
  allowfullscreen = TRUE, query = NULL)
```

RStudio

Rstudio

What is it?

RStudio is an Integrated Development Environment.
It makes working with R much easier

Features

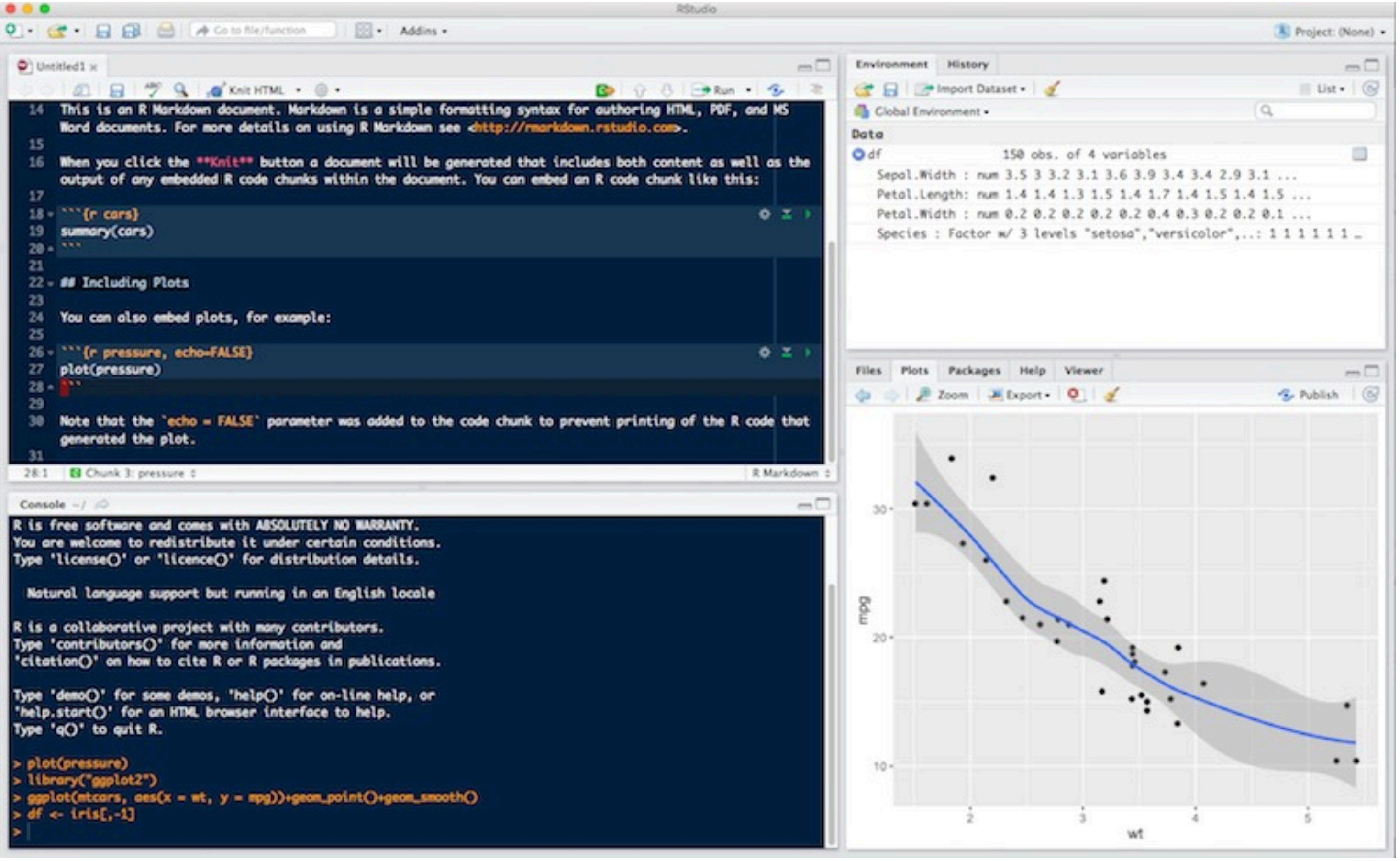
- *Console* to run **R**, with syntax highlighter
- *Editor* to work with scripts
- *Viewer* for data / plots / website
- *Package management* (including building)
- *Autocompletion* using TAB
- *Cheatsheets*
- *Git* integration for versioning
- *Inline* outputs (>= v1.03)
- *Keyboard shortcuts*
- *Notebooks*

Warning

Don't mix up **R** and **RStudio**.
R needs to be installed first.

Rstudio

The 4 panels layout



~~~
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS
Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the
output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
~~~

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that
generated the plot.
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> plot(pressure)
> library("ggplot2")
> ggplot(mtcars, aes(x = wt, y = mpg))+geom_point()+geom_smooth()
> df <- iris[,-1]
>
```

```
Data
df                  150 obs. of 4 variables
   Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
   Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
   Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
   Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 ...
```
```

## Useful Functions

- List all objects in memory: `ls()`
- Save an object: `save(obj, file)`
- Load an object: `load(file)`
- Set working directory: `setwd(dir)`

For the last one, using **Rstudio project** is recommended together with **Rmarkdown** files

## Data types

| Type | Example |
|---|---|
| numeric | integer (2), double (2.34) |
| string | "tidyverse !" |
| boolean | TRUE / FALSE |
| complex | 2+0i |

## Special cases

```
NA   # not available, missing data
NA_real_
NA_integer_
NA_character_
NA_complex_
NULL # empty
-Inf/Inf # infinite values
```

Data Structures

Vectors

`c()` is the function for **concatenate**

```
4                                  [1] 4
c(43, 5.6, 2.90)                   [1] 43.0  5.6  2.9
```

Factors

convert strings to factors, `levels` is the dictionary

```
factor(c("AA", "BB", "AA",         [1] AA BB AA CC
"CC"))                             Levels: AA BB CC
```

Matrix (2D), Arrays (≥ 3D)

won't dig into those

```
matrix(1:4, nrow = 2)                   [,1] [,2]
                                   [1,]    1    3
                                   [2,]    2    4
```

Lists

very important as can contain anything

```
list(f = c("rstudio", "rocks"),    $f
     v = c(43, 5.6, 2.90),         [1] "rstudio"  "rocks"
     s = 4)
                                   $v
                                   [1] 43.0  5.6  2.9

                                   $s
                                   [1] 4
```

Data frames are special lists

`data.frame`

same as list **but** where all objects *must* have the **same** length

Example

```
data.frame(f = factor(c("AA", "AA", "BB")),
           v = c(43, 5.6, 2.90),
           s = rep(4, 3))
```

```
   f    v s
1 AA 43.0 4
2 AA  5.6 4
3 BB  2.9 4
```

colum are atomic vectors

```
av <- c(2.5, 5.1)
av
[1] 2.5 5.1
c(av, "char")
[1] "2.5"  "5.1"  "char"
```

ok to mix in list-column, `tibbles` to avoid `I()`

```
tibble( f = c("AA", "AA", "BB"),
        v = c(43, 5.6, 2.90),
        l = list(av = c(2.5, 5.1),
                 lg = c(TRUE, FALSE),
                 st = "char"))
# A tibble: 3 x 3
      f     v         l
  <chr> <dbl>    <list>
1    AA  43.0 <dbl [2]>
2    AA   5.6 <lgl [2]>
3    BB   2.9 <chr [1]>
```

Data import

- Represents probably the first step of your work
- R can handle multiple data types
  - flat files (`.csv`, `.tsv`, ...)
  - excel files (`.xls`, `.xlsx`)
  - foreign statistical formats (`.sas` from SAS, `.sav` from SPSS, `.dta` from Stata)
  - databases (SQL, SQLite ...)

Tidyverse implementation

- R base already provides functions for text files (*i.e.* `read.csv()`)
- tidyverse redefines these functions:
  - **speed**
  - **characters are not coerced to factors by default**
  - generates tibbles

Tidyverse packages to import your data

- `read_csv()`: comma separated (CSV) files
- `read_tsv()`: tab separated files
- `read_delim()`: general delimited files
- `read_fwf()`: fixed width files
- `read_table()`: tabular files where colums are separated by white-space.
- `read_log()`: web log files

Data import *foreign softwares*

readxl

To import excel files (`.xls` and `.xlsx`):

- `read_excel()`
  - `read_xls()`
  - `read_xlsx()`

haven

- `read_sas()` for SAS
- `read_sav()` for SPSS
- `read_dta()` for Stata

Data Frames Most easy structure to use, have a matrix structure

> Tidy datasets are all alike; every messy dataset is messy in its own way
>
> — *Hadley Wickam*

Definitions

- **Variable**: A quantity, quality, or property that you can measure.
- **Observation**: A set of values that display the relationship between variables. To be an observation, values need to be measured under similar conditions, usually measured on the same observational unit at the same time.
- **Value**: The state of a variable that you observe when you measure it.

source: Garret Grolemund and `vignette("tidy-data")`

- Individual rows, columns, and cells in a data frame can be accessed through many methods of indexing.
- We most commonly use `object[row, column]` notation.

Accessing items in a `data.frame`

`mtcars` that can be used

```
head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
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
colnames(mtcars)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

single cell value

[**row** value, **column** value]

```
mtcars[2, 3]
[1] 160
```

one column

omitting row implies all

```
mtcars[2, ]
                   mpg cyl disp
hp drat    wt  qsec vs am
gear carb
Mazda RX4 Wag  21   6  160
110  3.9 2.875 17.02  0  1
4    4
```

one row

omitting column implies all

```
mtcars[, 3]
 [1] 160.0 160.0 108.0
258.0 360.0 225.0 360.0
146.7 140.8 167.6 167.6
[12] 275.8 275.8 275.8
472.0 460.0 440.0  78.7
75.7  71.1 120.1 318.0
[23] 304.0 350.0 400.0
79.0 120.3  95.1 351.0
145.0 301.0 121.0
```

Accessing items in a `data.frame`

named column

We can also access variables directly by using their **names** instead of indexes

Get **first 10 rows** of variable `mpg` using 3 notations:

first notation, `object[, "variable"]`

```
mtcars[1:10, "mpg"]
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2
```

second notation, `object$variable`

```
mtcars$mpg[1:10]
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2
```

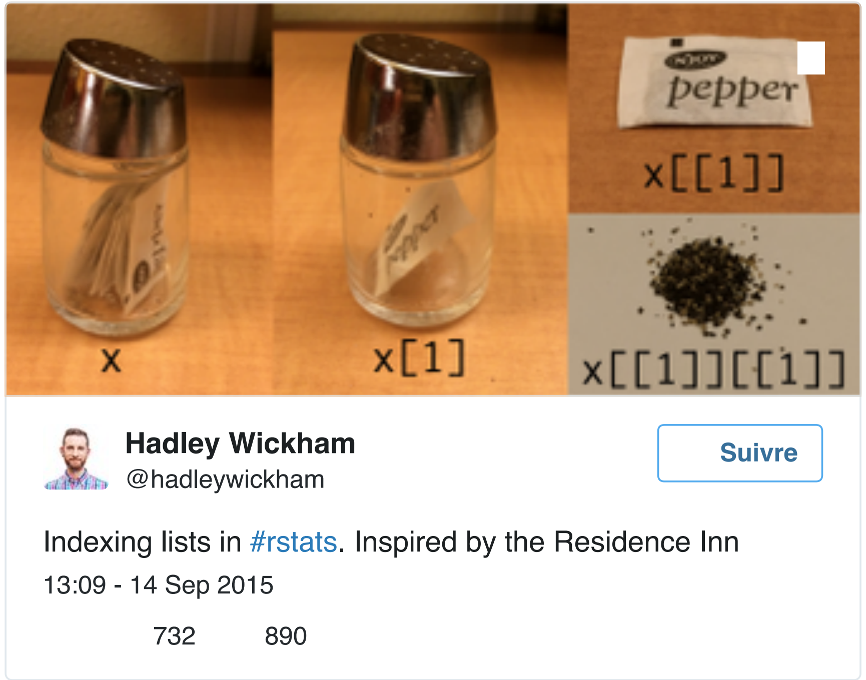
`$` is the shortcut for `[[`

third notation, `object[["variable"]]`

```
mtcars[["mpg"]][1:10]
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2
```

Lists

Pepper analogy



Hadley Wickham
@hadleywickham

Indexing lists in #rstats. Inspired by the Residence Inn

13:09 - 14 Sep 2015

732 890

Example

```
l <- list(name = "Farina",
          firstname = "Geoff",
          year = 1995)
```

```
l["firstname"]
$firstname
[1] "Geoff"
l[["firstname"]]
[1] "Geoff"
```

Question

How to subset a single pepper seed?

Description Of Dataset

Using `dim`

we get the number of observations(rows) and variables(columns) in the dataset.

```
dim(mtcars)
[1] 32 11
```

Using `str`

we get the structure of the dataset, including the `class(type)` of all variables.

```
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 ...
```

Exploring Data

Summary Stats Of Dataset

summary

when used on a dataset, returns distributional summaries of variables in the dataset.

```
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
```

quantile

function enables to get statistical metrics on the selected data

```
quantile(mtcars$mpg)
    0%    25%    50%    75%   100%
10.400 15.425 19.200 22.800 33.900
```

Exploring Data

Conditional Exploration

subset

enables to explore data conditionally

```
head(subset(mtcars, cyl <= 5), 10)
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
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
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

by

enables to call a particular function to sub-groups of data

```
by(mtcars, mtcars$cyl, summary)
mtcars$cyl: 4
      mpg             cyl         disp              hp        
 Min.   :21.40   Min.   :4   Min.   : 71.10   Min.   : 52.00  
 1st Qu.:22.80   1st Qu.:4   1st Qu.: 78.85   1st Qu.: 65.50  
 Median :26.00   Median :4   Median :108.00   Median : 91.00  
 Mean   :26.66   Mean   :4   Mean   :105.14   Mean   : 82.64  
 3rd Qu.:30.40   3rd Qu.:4   3rd Qu.:120.65   3rd Qu.: 96.00  
 Max.   :33.90   Max.   :4   Max.   :146.70   Max.   :113.00  
      drat             wt             qsec             vs        
 Min.   :3.690   Min.   :1.513   Min.   :16.70   Min.   :0.0000  
 1st Qu.:3.810   1st Qu.:1.885   1st Qu.:18.56   1st Qu.:1.0000  
 Median :4.080   Median :2.200   Median :18.90   Median :1.0000  
 Mean   :4.071   Mean   :2.286   Mean   :19.14   Mean   :0.9091  
 3rd Qu.:4.165   3rd Qu.:2.623   3rd Qu.:19.95   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :3.190   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.5000   1st Qu.:4.000   1st Qu.:1.000  
 Median :1.0000   Median :4.000   Median :2.000  
 Mean   :0.7273   Mean   :4.091   Mean   :1.545  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:2.000  
 Max.   :1.0000   Max.   :5.000   Max.   :2.000  
--------------------------------------------------------
mtcars$cyl: 6
      mpg             cyl         disp             hp       
 Min.   :17.80   Min.   :6   Min.   :145.0   Min.   :105.0  
 1st Qu.:18.65   1st Qu.:6   1st Qu.:160.0   1st Qu.:110.0  
 Median :19.70   Median :6   Median :167.6   Median :110.0  
 Mean   :19.74   Mean   :6   Mean   :183.3   Mean   :122.3  
 3rd Qu.:21.00   3rd Qu.:6   3rd Qu.:196.3   3rd Qu.:123.0  
 Max.   :21.40   Max.   :6   Max.   :258.0   Max.   :175.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :2.620   Min.   :15.50   Min.   :0.0000  
 1st Qu.:3.350   1st Qu.:2.822   1st Qu.:16.74   1st Qu.:0.0000  
 Median :3.900   Median :3.215   Median :18.30   Median :1.0000  
 Mean   :3.586   Mean   :3.117   Mean   :17.98   Mean   :0.5714  
 3rd Qu.:3.910   3rd Qu.:3.440   3rd Qu.:19.17   3rd Qu.:1.0000  
 Max.   :3.920   Max.   :3.460   Max.   :20.22   Max.   :1.0000  
```

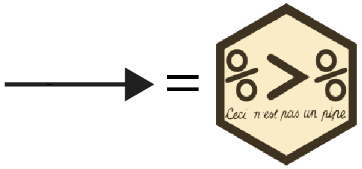
Tidyverse

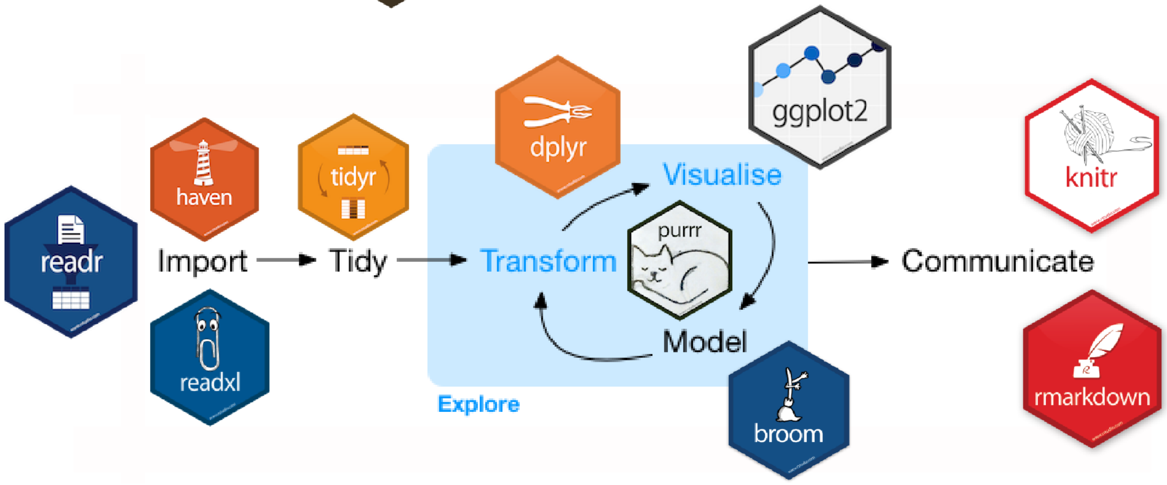
Components

dplyr
forcats
ggplot2
haven
purrr
readr
readxl
stringr
TIBBLE
tidyr
tidyverse

Tidyverse

packages in processes





data.frames

factors

strings

4 days workshop at the doctoral school@Uni last May 2017, probably again March 2018

Practical Session

Objectives

You will learn to:

- install and run R and Rstudio on your machine
- use R on the clusters
- download a file and process it
- create a simple *ggplot* remotely
- summarise a dataset using different packages and benchmark them
- demonstrate why packages are so much better than R base
- perform single machine parallelisation on `gaia`
- perform cluster parallelisation on `gaia`

Acknowledgements

- **Jospeh Emeras** who wrote most of this session
- Eric Koncina, slides prepared with his iosp R package
- Eric Koncina & Roland Krause for their content in the R workshop
- *HPC* team

- Practical here: https://github.com/ULHPC/tutorials/tree/devel/advanced/R
- Slides (html): https://cdn.rawgit.com/ULHPC/tutorials/devel/advanced/R/Intro_PS.html
- Slides (pdf): https://github.com/ULHPC/tutorials/raw/devel/advanced/R/Intro_PS.pdf