

# Introduction

## Data processing with the tidyverse



A. Ginolhac | rworkshop | 2021-09-08

# Good morning!

## What you can do now:

Check the material at the main site

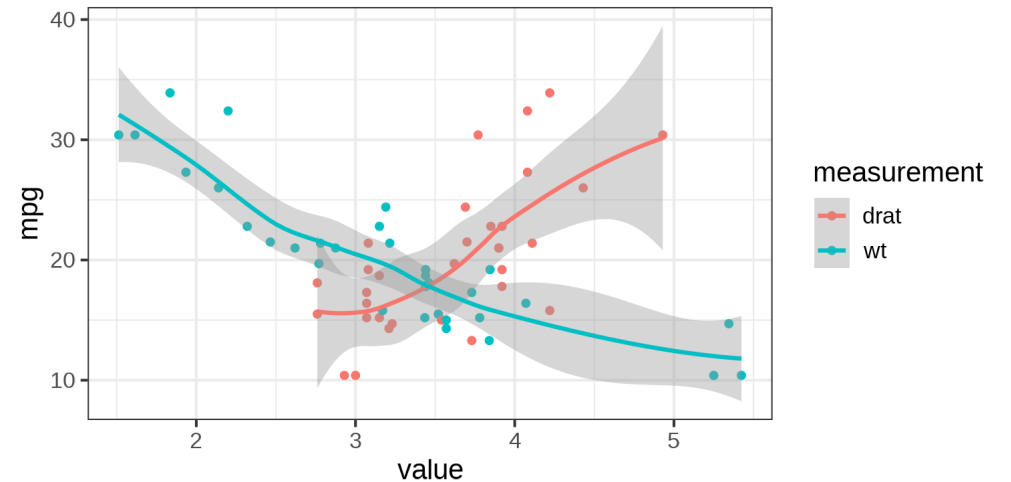
<https://rtidyverse2021.leibniz-psychology.org/>

Info and schedule

<https://leibniz-psychology.org/rtidyverse2021/>

Check your install

[https://rtidyverse2021.leibniz-psychology.org/install\\_tutorial.html](https://rtidyverse2021.leibniz-psychology.org/install_tutorial.html)





# Overview



This 3-day-course provides an introduction to the **tidyverse**, a  dialect.

- Focusing on loading and cleaning data for exploratory visualizations
- Speeding data manipulation is the mission of this course
- This workshop is composed of 30 hours:

## Lectures

- ~ 8 hours
- available online ( / )
- live **exercises** included

## Practicals


- ~ 15 hours
- Using your own laptop
- Supplementary exercises if needed

# Speakers



# Learning objectives

## You will learn


-  specificity
  - community
  - package ecosystem
  - vectorization
- opinionated tidyverse
- Rstudio layout
- basic data types (recap)
- basic data structures (recap)



# What is 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, NZ
- Focus on data analysis and plotting
-  is also shorthand for the ecosystem around this language
  - Book authors
  - Package developers
  - Ordinary useRs

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

# Why using R?



- It's *free!* and **open-source**
- easy to install / maintain
- multi-platform (Windows, macOS, GNU/Linux)
- can process big files and analyse huge amounts of data (db tools)
- integrated data visualization tools, even *dynamic* [shiny](#)
- fast, and even faster with C++ integration via [Rcpp](#) or [cpp11](#).
- easy to get help
  - [huge R community in the web](#)
  - [stackoverflow](#) with a lot of tags like **r**, **ggplot2** etc.
  - [rbloggers](#)

# About R is slow



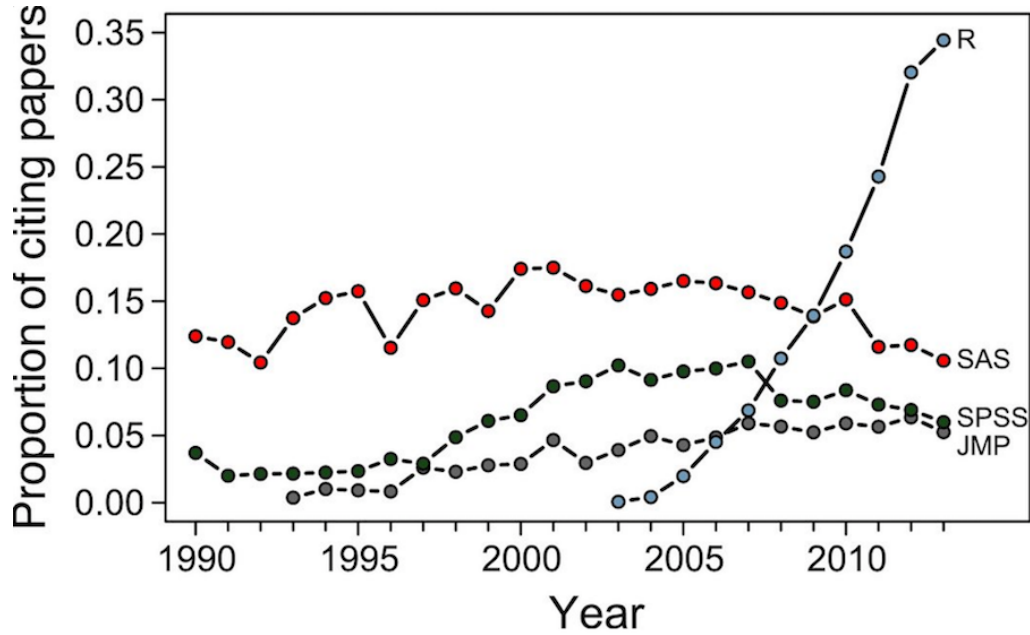
Source: [Jozef Hajnala](#)

# Twitter R community

[#rstats](#) on twitter



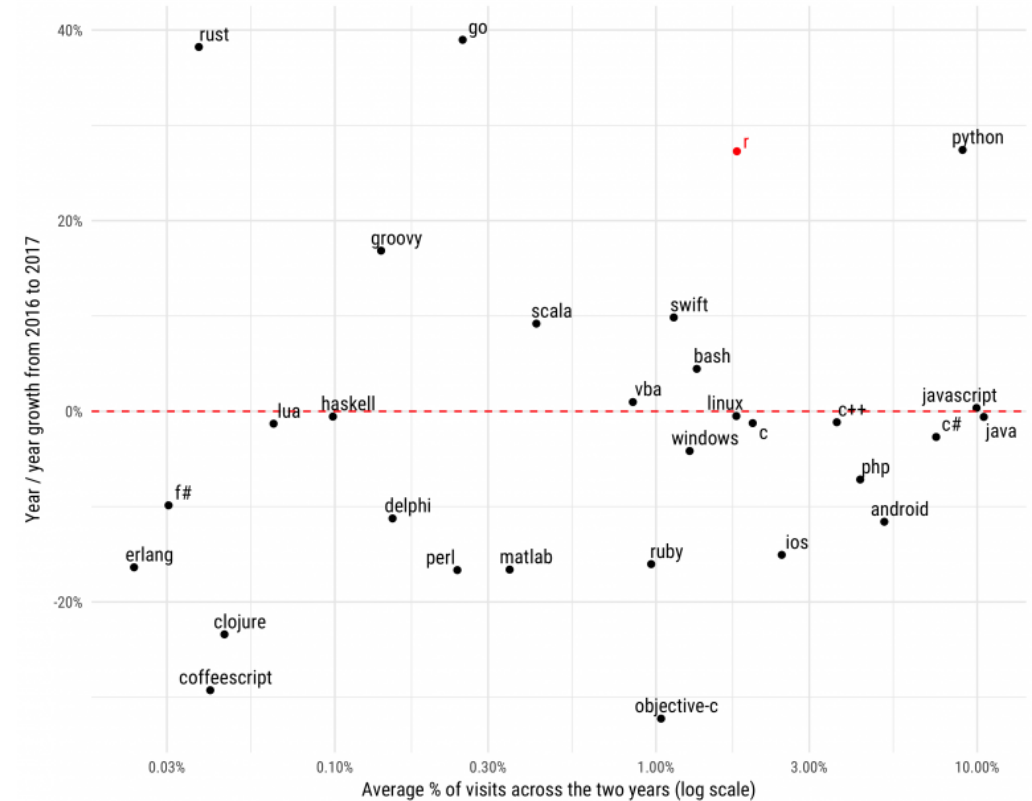
# Constant trend



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...](https://onlinelibrary.wiley.com/doi/10.1002/ec...)

## Year over year growth in traffic to programming languages/platforms

Comparing question views in January-September of 2016 and 2017, in World Bank high-income countries. TypeScript had a growth rate of 134% and an average size of .38%; and was omitted.



Source: [Touchon & McCoy. Ecosphere. 2016](#)

Source: [D. Robinson, StackOverflow blog](#)

# Packages

19791  
CRAN PACKAGES

2130  
BIOCONDUCTOR PACKAGES

2175  
R-FORGE PACKAGES

70463  
GITHUB PACKAGES

## CRAN

**reliable**: package is checked during submission process

MRAN for Windows users

## bioconductor

- dedicated to biology. {limma} example
- required dedicated package

typical install:

```
# install.packages("BiocManager")  
BiocManager::install("limma")
```

## GitHub



easy install thanks to remotes.

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

could be a security issue

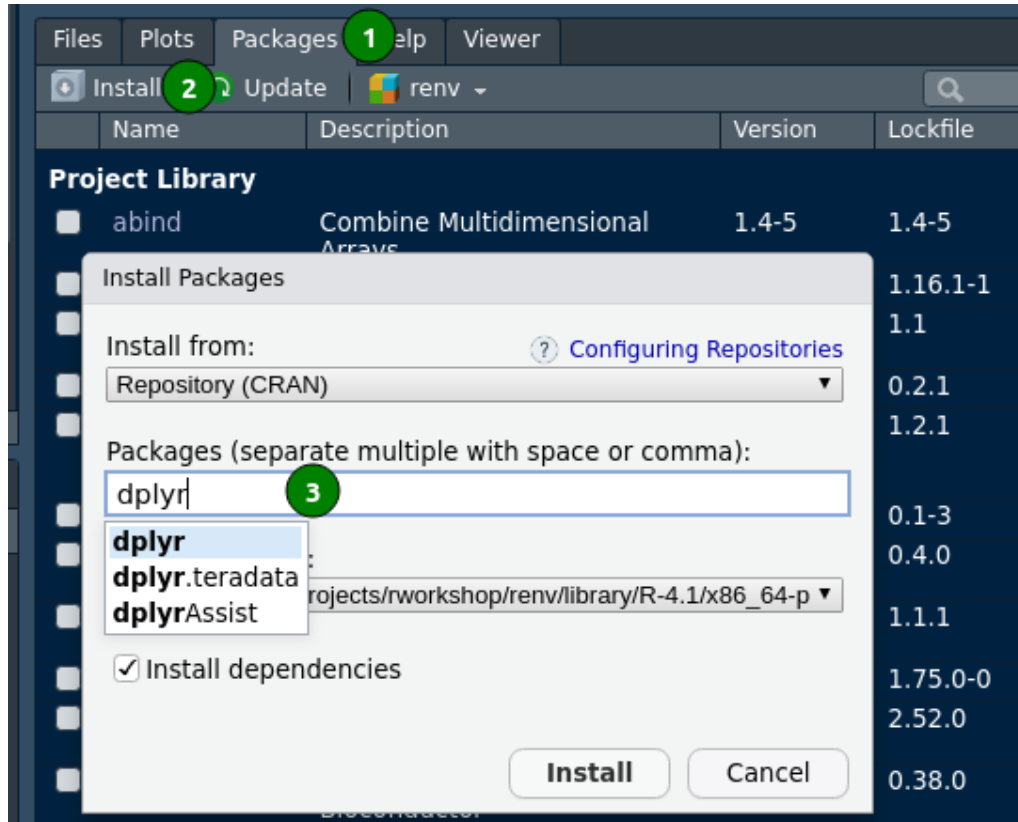
Source: rdr.io on 2021-05-04

# R-universe

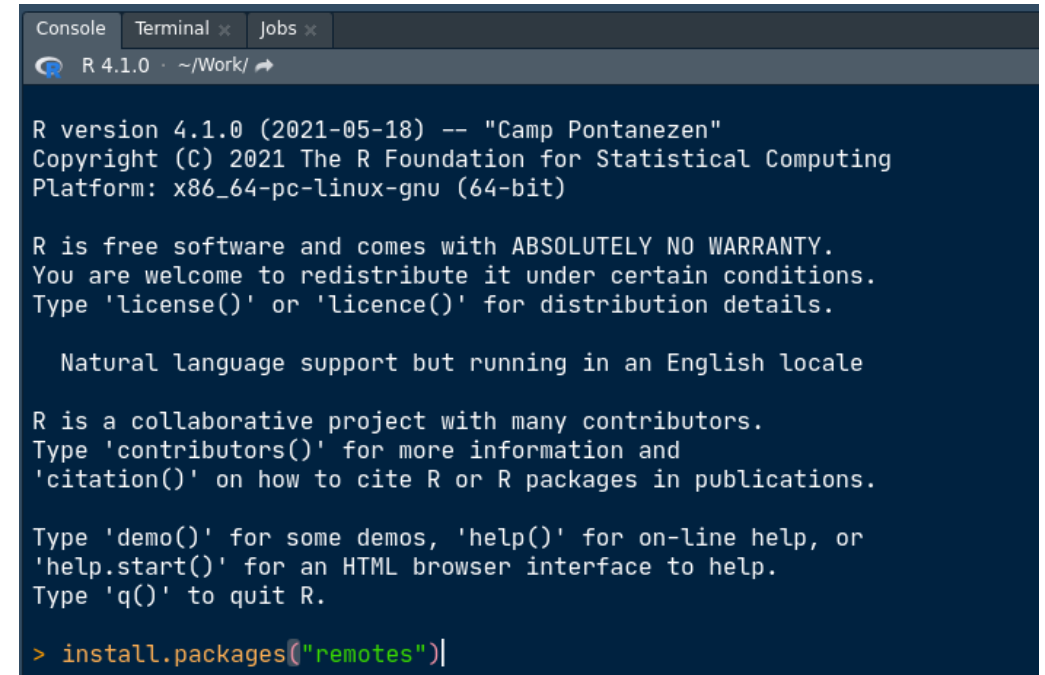
- [ropensci](#) initiative, by [Jeroen Ooms](#)
- [Video intro](#), everyone can have a domain for publishing their content
- Provides automated built vignettes and binary packages for  /  (also dev versions)
- No [remotes](#) / [devtools](#) / [rtools](#) required, design your rules outside [CRAN](#)

# Installation

CRAN install from Rstudio (autocompletion)



CRAN install from Rstudio' console



more in the article from [David Smith](#)

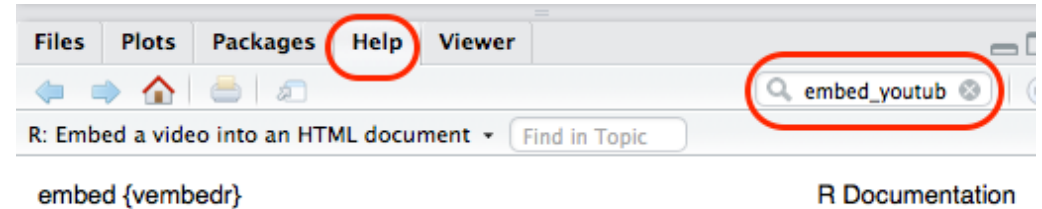
# Help pages

2 possibilities for manual pages.

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

Sadly, manpages are often unhelpful, [vignettes](#) or [articles](#) better described workflow.

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



## 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)
```

# Drawback: Steep learning curve

Period of much suckiness



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*.

— Hadley Wickham

source: [Hadley Wickham dplyr tutorial at useR 2014](#)

# R is hard to learn

**R base** is complex, has a long history and many contributors

## Why R is hard to learn

- Unhelpful help `?print`
- generic methods `print.data.frame`
- too many commands `colnames`, `names`
- inconsistent names `read.csv`, `load`, `readRDS`
- un-strict syntax, was designed for interactive usage
- too many ways to select variables `df$x`, `df$"x"`, `df[, "x"]`, `df[[1]]`
- [...] see [r4stats' post](#) for the full list
- the `tidyverse` curse

■ Navigating the balance between `base` R and the `tidyverse` is a challenge to learn

— Robert A. Muenchen

source: [Robert A. Muenchen' blog](#)



The ambiguity [of the S language] is real and goes to a key objective: we 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 **slide gradually into programming**, when the language and system aspects would become more important.

— John Chambers, "Stages in the Evolution of S"

source: [Teaching R to New Users: from tapply to Tydiverse 2018](#) by Roger D. Peng

# Tidyverse origin



H. Wickham

[Hadley](#) is Chief Scientist at **Rstudio**

- coined the *tidyverse* at [userR meeting in 2016](#)
- developed and maintains most of the core *tidyverse* packages



























































We think the tidyverse is better, especially for beginners. It is:

- recent (both an issue and an advantage)
- allows doing powerful things quickly
- unified
- consistent, one way to do things
- give strength to learn base R
- criticisms will come later

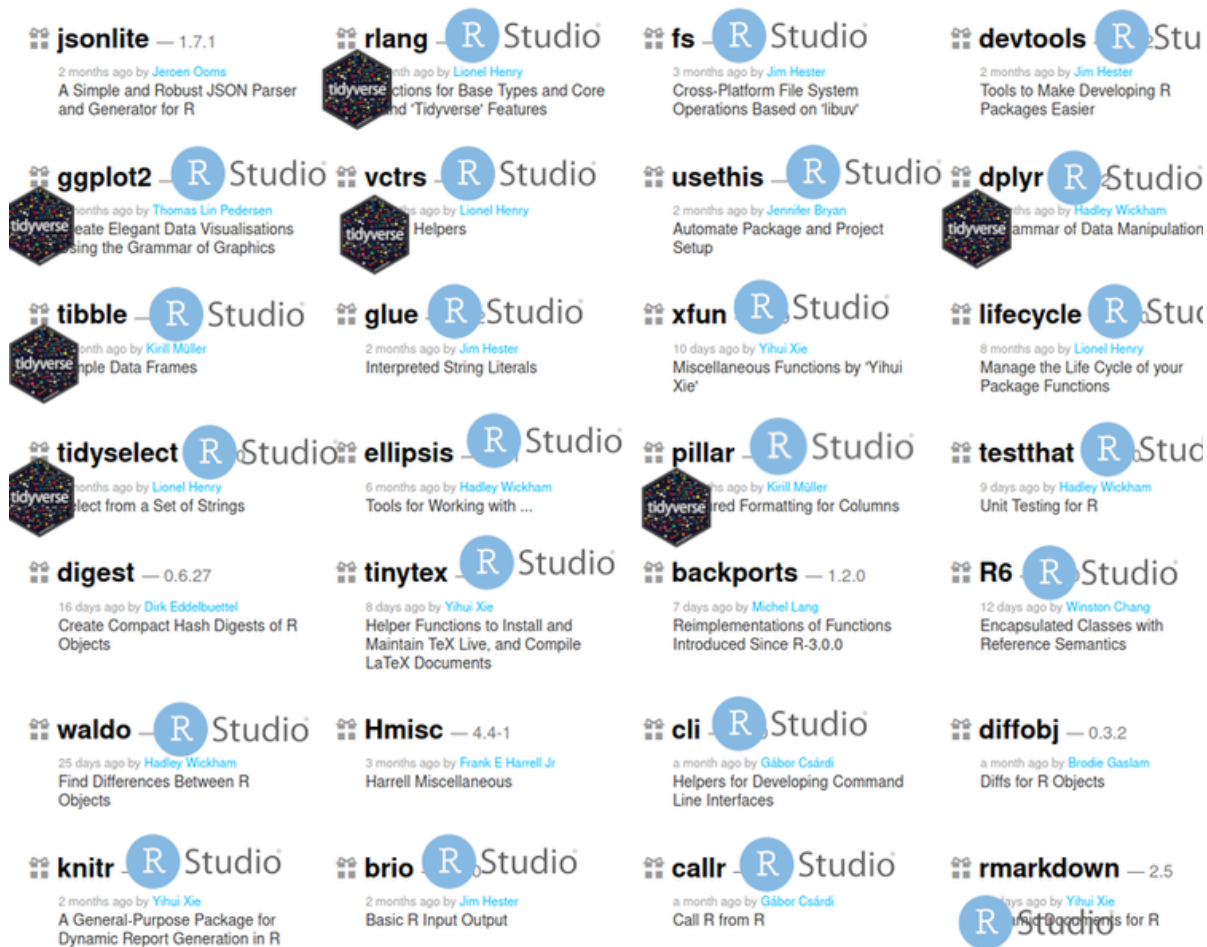
# Tidyverse, core packages

# Top download

  <b>jsonlite</b> — 1.7.1 2 months ago by <a href="#">Jeroen Ooms</a> A Simple and Robust JSON Parser and Generator for R	  <b>rlang</b> — 0.4.8 a month ago by <a href="#">Lionel Henry</a> Functions for Base Types and Core R and 'Tidyverse' Features	  <b>fs</b> — 1.5.0 3 months ago by <a href="#">Jim Hester</a> Cross-Platform File System Operations Based on 'libuv'	  <b>devtools</b> — 2.3.2 2 months ago by <a href="#">Jim Hester</a> Tools to Make Developing R Packages Easier
  <b>ggplot2</b> — 3.3.2 5 months ago by <a href="#">Thomas Lin Pedersen</a> Create Elegant Data Visualisations Using the Grammar of Graphics	  <b>vctrs</b> — 0.3.4 2 months ago by <a href="#">Lionel Henry</a> Vector Helpers	  <b>usethis</b> — 1.6.3 2 months ago by <a href="#">Jennifer Bryan</a> Automate Package and Project Setup	  <b>dplyr</b> — 1.0.2 3 months ago by <a href="#">Hadley Wickham</a> A Grammar of Data Manipulation
  <b>tibble</b> — 3.0.4 a month ago by <a href="#">Kirill Müller</a> Simple Data Frames	  <b>glue</b> — 1.4.2 2 months ago by <a href="#">Jim Hester</a> Interpreted String Literals	  <b>xfun</b> — 0.19 10 days ago by <a href="#">Yihui Xie</a> Miscellaneous Functions by 'Yihui Xie'	  <b>lifecycle</b> — 0.2.0 8 months ago by <a href="#">Lionel Henry</a> Manage the Life Cycle of your Package Functions
  <b>tidyselect</b> — 1.1.0 6 months ago by <a href="#">Lionel Henry</a> Select from a Set of Strings	  <b>ellipsis</b> — 0.3.1 6 months ago by <a href="#">Hadley Wickham</a> Tools for Working with ...	  <b>pillar</b> — 1.4.6 4 months ago by <a href="#">Kirill Müller</a> Coloured Formatting for Columns	  <b>testthat</b> — 3.0.0 9 days ago by <a href="#">Hadley Wickham</a> Unit Testing for R
  <b>digest</b> — 0.6.27 16 days ago by <a href="#">Dirk Eddelbuettel</a> Create Compact Hash Digests of R Objects	  <b>tinytex</b> — 0.27 8 days ago by <a href="#">Yihui Xie</a> Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents	  <b>backports</b> — 1.2.0 7 days ago by <a href="#">Michel Lang</a> Reimplementations of Functions Introduced Since R-3.0.0	  <b>R6</b> — 2.5.0 12 days ago by <a href="#">Winston Chang</a> Encapsulated Classes with Reference Semantics
  <b>waldo</b> — 0.2.2 25 days ago by <a href="#">Hadley Wickham</a> Find Differences Between R Objects	  <b>Hmisc</b> — 4.4-1 3 months ago by <a href="#">Frank E Harrell Jr</a> Harrell Miscellaneous	  <b>cli</b> — 2.1.0 a month ago by <a href="#">Gábor Csárdi</a> Helpers for Developing Command Line Interfaces	  <b>diffobj</b> — 0.3.2 a month ago by <a href="#">Brodie Gaslam</a> Diffs for R Objects
  <b>knitr</b> — 1.30 2 months ago by <a href="#">Yihui Xie</a> A General-Purpose Package for Dynamic Report Generation in R	  <b>brio</b> — 1.1.0 2 months ago by <a href="#">Jim Hester</a> Basic R Input Output	  <b>callr</b> — 3.5.1 a month ago by <a href="#">Gábor Csárdi</a> Call R from R	  <b>rmarkdown</b> — 2.5 19 days ago by <a href="#">Yihui Xie</a> Dynamic Documents for R

source: [METACRAN, top downloaded](#) (2020-11-09)


# Rstudio / tidyverse hard to ignore



source: [METACRAN, top downloaded](#) (2020-11-09)

# 18<sup>th</sup> of May, R 4.1

## Tidyverse

- pipe 
- lambda: `~ .x`
- `c(factor("a"), factor("b"))` is `[1] a b`



lambda functions are **anonymous**. Classic syntax is `(function(x) x + 2)(2)`  
**v4.0** brought `stringsAsFactors = FALSE` that `tibbles` have



- native pipe `|>`
- lambda: `\(x)`
- `combine factors` also works  
(before: `[1] 1 1`)



# What is it?

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

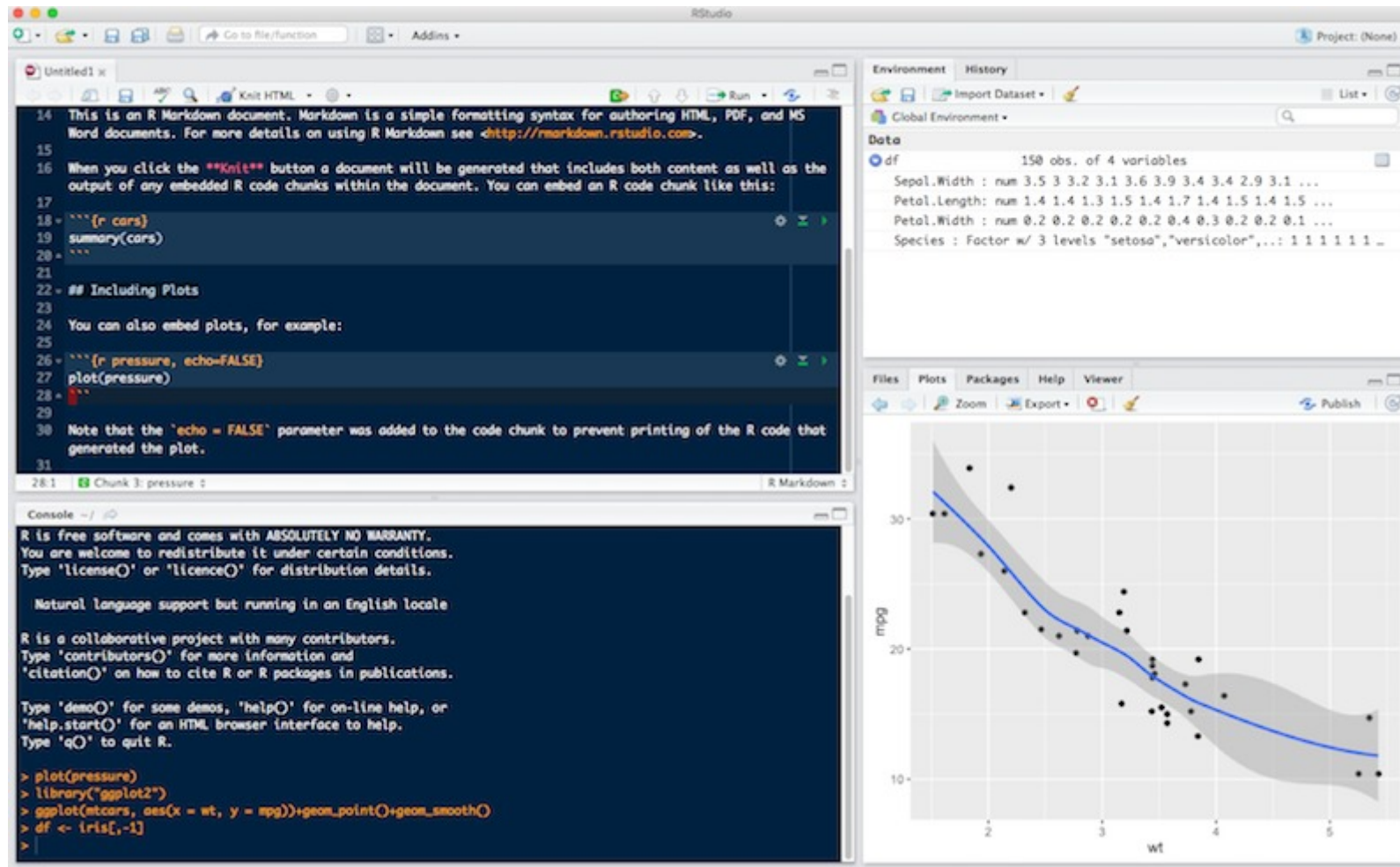
## Warning

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

## Features

- *Projects* to ease files organisation
- *Console* to run **R**, with syntax highlighter
- full support for *Rmarkdown* docs & chunks
- *Viewer* for data / plots / website
- *Package management* (including building, tests and development)
- *Autocompletion* using **TAB**
- [Cheatsheets](#) (**New 2021-08-23**)
- *Git* integration for versioning
- *Inline* outputs
- *Keyboard shortcuts*
- integrated *Terminal*
- *Jobs* for running long runs in a separated session

# The 4 panels layout



# Four panels

## scripting

- could be your main window
- should be a **Rmarkdown** doc
- tabs are great

## Console

- could be hidden with **inline** outputs
- **Rmarkdown** output logs
- optional, embed a nice **Terminal** tab
- optional, **Jobs** tab

## Environment

- **Environment**, display loaded objects and their `str()`
- History is useless IMO
- nice **git** integration
- **Build** for development
- database **connections** interface
- **Tutorial** integration of **learnr**

## Files / Plots / Help

- **Packages** management tab
- unnecessary **Plots** panel when using **inline** outputs
- **Help** tab
- Viewer for animations

# Rearrange panels

The screenshot displays the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a 'Go to file/function' search bar. The main source editor window shows an R Markdown document with a YAML header, R code chunks, and markdown text. A hand cursor is positioned over the text in the source editor. The bottom-left pane contains the Console, Terminal, and Jobs tabs, with the Console showing the R startup message. The right-hand side of the interface features several panels: 'Environment' (showing 'Global Environment' and 'Environment is empty'), 'History', 'Connections', 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. The 'Viewer' panel is currently active, displaying 'R Resources' and 'RStudio' links.

```
1 ---
2 title: "Untitled"
3 author: "sp"
4 date: "5/20/2019"
5 output: html_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for
15 authoring HTML, PDF, and MS Word documents. For more details on using R
16 Markdown see <http://rmarkdown.rstudio.com>.
17
18 When you click the Knit button a document will be generated that includes
19 both content as well as the output of any embedded R code chunks within the
```

3:12 Untitled R Markdown

You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.

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.

source:Shannon Dileggi  
workspace loaded from ~/.RData]

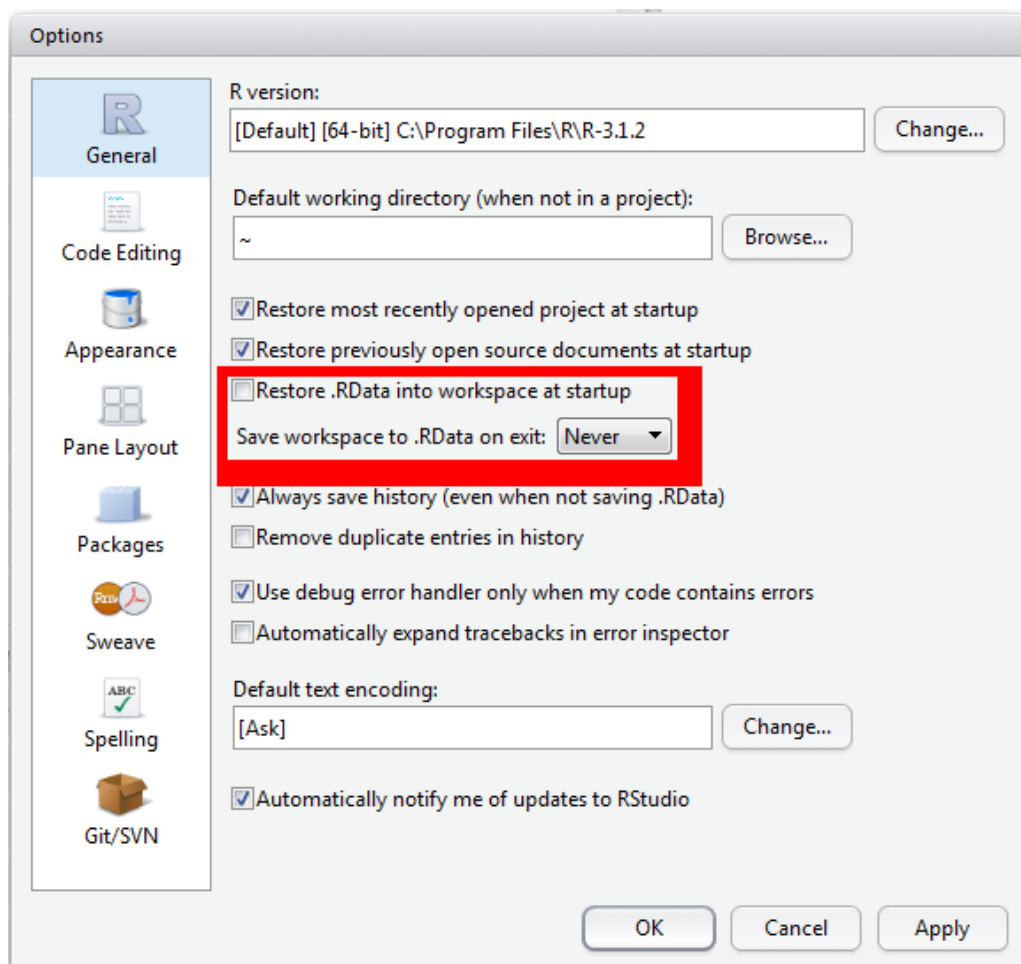
> |

Environment History Connections  
Global Environment  
Environment is empty

Files Plots Packages Help Viewer  
Home Find in Topic  
R Resources RStudio  
Learning R Online RStudio IDE Support  
CRAN Task Views RStudio Community Forum  
R on StackOverflow RStudio Cheat Sheets  
Getting Help with R RStudio Tip of the Day  
RStudio Packages  
RStudio Products

# For reproducibility

## Options to activate / deactivate



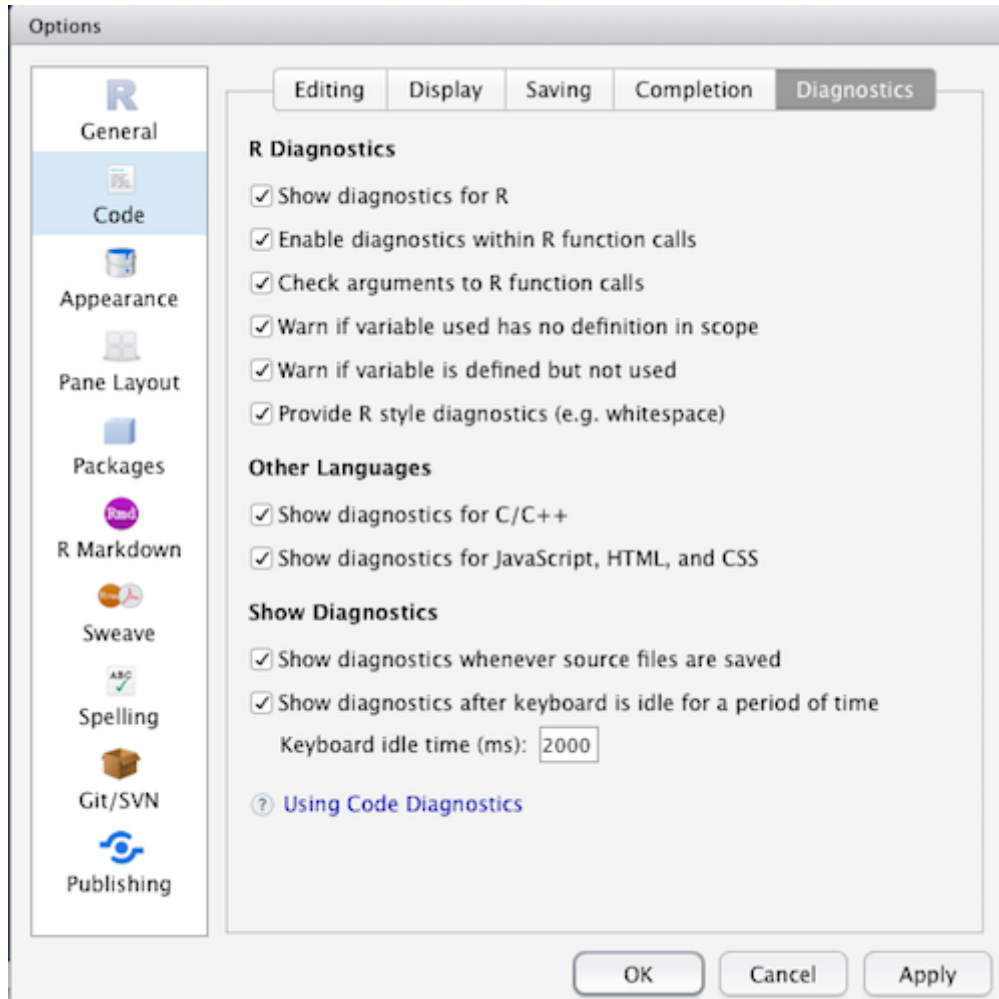
`rm(list = ls())` is not recommended

- does **not** make a fresh R session
  - `library()` calls remain
  - working directory not set!
  - modified functions, `evil == <- !=`
- knitting `Rmarkdown` files solves it

[source: Jenny Bryan article](#)

# Code diagnostics, highly recommended

using [Global Options](#) -> [Code](#) -> [Diagnostics](#) editing panel:



- check argument calls

```
1 add_numbers <- function(x, y) {  
2   x + y  
3 }  
4  
5 add_numbers(1)  
6  
argument 'y' is missing, with no default
```

- missing arguments

```
1 list(  
2   first = 1  
3   second = 2  
4 )  
5  
6  
expected ',' after expression
```

- variable definitions

```
1 hw <- HoltWinters(ldeaths)  
2 p <- predict(HW, n.ahead = 36, level = 0.95)  
3  
4  
no symbol named 'HW' in scope; did you mean 'hw'?
```

- unused variables & style recommendations

source: [Kevin Ushey's article](#)

# Symbol and their names for this course

## Common operators

= - equal  
.  
, - comma  
> - greater than  
< - less than  
~ - twiddle  
\* - star  
- - hyphen  
\_ - underscore

## Quotation and comments

" - double quotation marks  
' - single quotation marks  
` - backticks  
# - hash  
| - (vertical) bar  
/ - (forward) slash  
\ - backslash

## Enclosures

() - parentheses  
[] - (square) brackets  
{ } - (curly) braces  
<> - chevrons

## R-specific operators

<- - assignment (left)  
-> - right assignment  
%>% - (magrittr) pipe  
|> - (base) pipe

# Arithmetic operations

- `+`: addition
- `-`: subtraction
- `*`: multiplication
- `/`: division
- `^` or `**`: exponentiation
- `%%`: modulo (remainder after division)
- `%/%`: integer division

**Remember that R will:**

- first perform **exponentiation**
- then multiplications and/or divisions
- and finally additions and/or subtractions.

## prioritization

change the priority in evaluation:

- parentheses `(` and `)` to group calculations

# Using `library()`, ensure function' origin

with only `base` loaded

```
x <- 1:10  
filter(x, rep(1, 3))
```

```
Time Series:  
Start = 1  
End = 10  
Frequency = 1  
[1] NA 6 9 12 15 18 21 24 27 NA
```

**Conflict:** 2 packages export same function

with the same name, the latest loaded wins

```
library(dplyr)  
filter(x, rep(1, 3))
```

```
Error in UseMethod("filter"): no applicable method for 'filter'
```

**Solution:** use `::` to call functions from a specific package

```
stats::filter(x, rep(1, 3))
```

```
Time Series:  
Start = 1  
End = 10  
Frequency = 1  
[1] NA 6 9 12 15 18 21 24 27 NA
```

Or use the [conflicted](#) package

# **Data types and structures**

**R base**

# Getting started

Let's get ready to use **R** and **RStudio**

## do the following

- Open up RStudio
- Maximize the RStudio window
- Click the Console pane, at the prompt (**>**) type in **3 + 2** and hit enter

```
> 3 + 2
```

# 4 main types

Type	Example
numeric	integer (2), double (2.34)
character (strings)	'tidyverse!'
boolean	TRUE / FALSE
complex	2+0i

## Missing data and special cases

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

```
2L
```

```
[1] 2
```

```
typeof(2L)
```

```
[1] "integer"
```

```
2.34
```

```
[1] 2.34
```

```
typeof(2.34)
```

```
[1] "double"
```

```
"tidyverse!"
```

```
[1] "tidyverse!"
```

```
TRUE
```

```
[1] TRUE
```

# 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", "CC"))
```

```
[1] AA BB AA CC  
Levels: AA BB CC
```

## Lists

very important as it can contain anything

```
list(f = factor(c("AA", "AA")),  
     v = c(43, 5.6, 2.90),  
     s = 4L)
```

```
$f  
[1] AA AA  
Levels: AA  
  
$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

```
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

## Example, missing one element in v

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

```
Error in data.frame(f = factor(c("AA", "AA", "BB")), v = c(43,
```

# Concatenate atomic elements

## collection of simple *things*

- *things* are the smallest elements: **atomic**
- **must** be of same **mode**: **automatic** coercion
- indexed, from **1** to **length(vector)**
- created with the **c()** function

```
c(2, TRUE, "a string")
```

```
[1] "2"      "TRUE"    "a string"
```

## Manual coercion

```
as.character(c(2, TRUE, "a string"))
```

```
[1] "2"      "TRUE"    "a string"
```

```
as.double(c(2, TRUE, "a string"))
```

```
[1]  2 NA NA
```

```
as.double(c(2, 2.456, "a string"))
```

```
[1] 2.000 2.456  NA
```

# Assignment

operator is `<-`, associate a *name* to an object

```
my_vec <- c(3, 4, 1:3)
my_vec
```

```
[1] 3 4 1 2 3
```

## tip

Rstudio has the built-in shortcut **Alt+ -** for `<-`

## Rationale

if you don't assigned a name to a created object it is only temporary. Assigning allows to save and re-use the object for a downstream step.

# Binding names to values

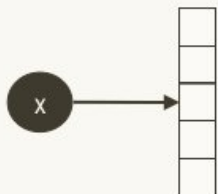
## Object has no name



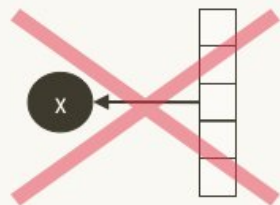
Hadley Wickham ✓  
@hadleywickham



In [#rstats](#), it's surprisingly important to realise that names have objects; objects don't have names



A name “has” an object



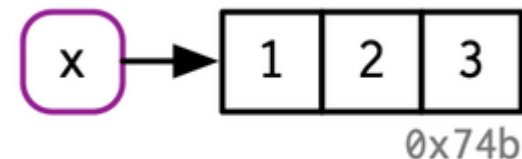
An object doesn't have a name

9:17 PM · May 16, 2016



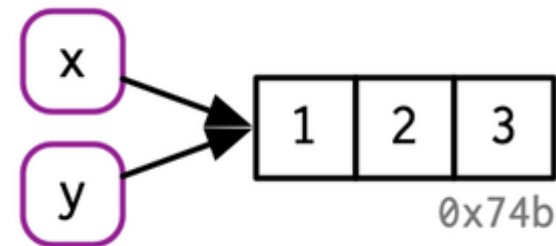
vector is bind to the name **x**

```
x <- 1:3
```



same vector is also bind to the name **y**

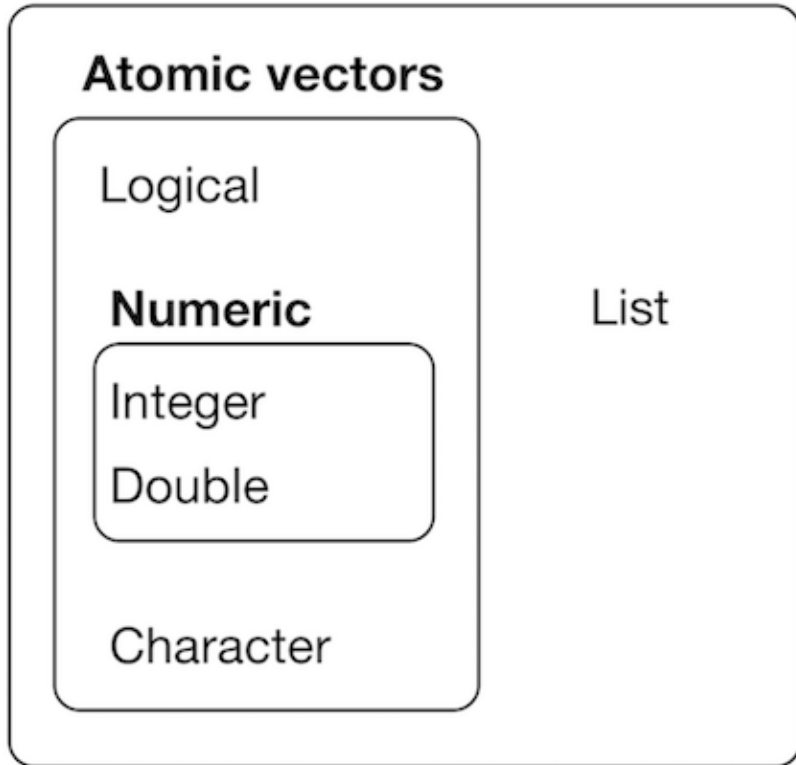
```
y <- x
```



source: [H. Wickham](#) - [Adv R](#), [licence CC](#)

# Hierarchy

## Vectors



NULL

```
is.vector(c("a", "c"))
```

```
[1] TRUE
```

```
is.vector(list(a = 1))
```

```
[1] TRUE
```

```
is.atomic(list(a = 1))
```

```
[1] FALSE
```

```
is.data.frame(list(a = 1))
```

```
[1] FALSE
```

source: [H. Wickham](#) - [R for data science](#), [licence CC](#)

# Subsetting vectors

## important ⚠

unlike **python** or **Perl**, vectors use **1-based** index!!

## : operator

generate integer sequence

```
3:10
```

```
[1] 3 4 5 6 7 8 9 10
```

## extract > 1 element

select elements from position **3** to **10**:

```
LETTERS[3:10]
```

```
[1] "C" "D" "E" "F" "G" "H" "I" "J"
```

## break in sequence, use `c()`

```
LETTERS[c(2:5, 7)]
```

```
[1] "B" "C" "D" "E" "G"
```

## negative selection

```
LETTERS[-(2:21)]
```

```
[1] "A" "V" "W" "X" "Y" "Z"
```

# Vectorized operation, one of the best R feature

```
my_vec <- 1:8  
my_vec
```

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

```
my_vec + 2
```

```
[1] 3 4 5 6 7 8 9 10
```

try this one

```
my_vec / 2
```

and this one, how does it work?

```
my_vec + 1:4
```

02:00

# Answers

```
my_vec / 2
```

```
[1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

```
my_vec + 1:4
```

```
[1] 2 4 6 8 6 8 10 12
```

**R** is **recycling** (*silently*) the shorter vector.

**But**, incompatible lengths create a *warning* !

```
my_vec + 1:3
```

```
Warning in my_vec + 1:3: longer object length is not a multiple of shorter object length
```

```
[1] 2 4 6 5 7 9 8 10
```

Avoid writing loops (and do **not** grow a vector)

```
res <- vector(mode = "numeric", length = length(my_vec))
for (i in seq_along(my_vec)) {
  res[i] <- my_vec[i] + 2
}
res
```

```
[1] 3 4 5 6 7 8 9 10
```

# Pipes



Output the result of one function as input for the next one

**R base**, classic parenthesis syntax

```
set.seed(12)
round(mean(rnorm(5)), 2)
```

```
[1] -0.76
```

And pipes are into **R base** since version **4.1** as `|>*`.

For **tidyverse** functions (**data** first argument), both pipes work similarly. Base being faster.

**magrittr** pipeline, (originally by Stefan M. Bache)

```
set.seed(12)
rnorm(5) %>%
  mean() %>%
  round(2)
```

```
[1] -0.76
```

Of note, **magrittr** needs to be loaded with either:

```
library(magrittr)
library(dplyr)
library(tidyverse)
```

\*: [Luke Tierney useR!2020](#)

# Before we stop

## You learned to:

- Introduction
  - R
  - Rstudio
  - tidyverse rationale
- data types
  - main categories
  - coerce
- data structures
  - main categories
  - sub-setting
  - vectorization



## Acknowledgments ☐ ☐

- Roland Krause
- Hadley Wickham
- Robert Muenchen
- Romain François
- Jenny Bryan

**Thank you for your attention!**