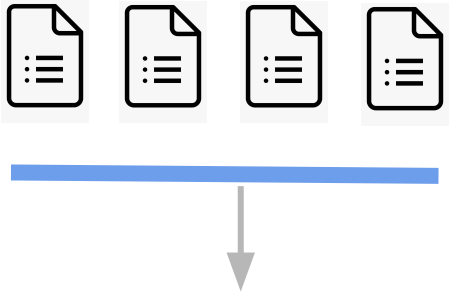
Iteration with purrr package

for automatized files management

1. Obtain a dataframe from multiple files
2. Automatize report generation

1. Obtain a dataframe from multiple files
2. Automatize report generation

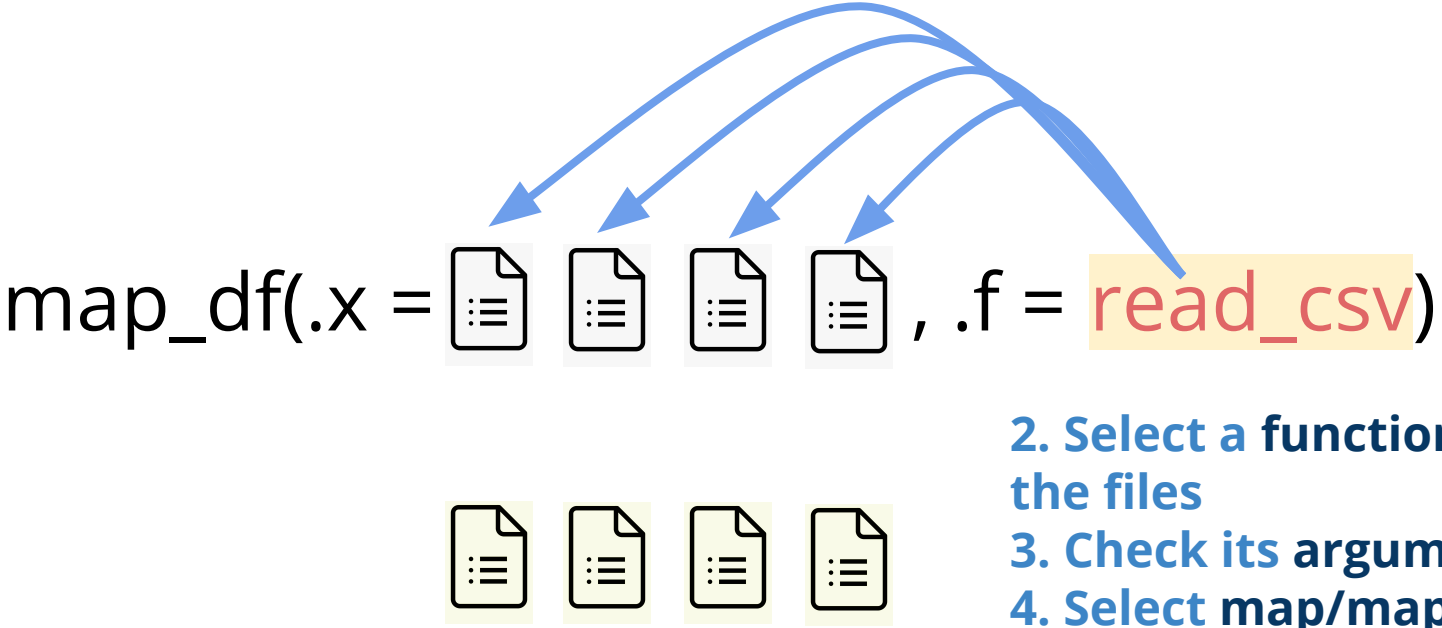
One dataframe from multiple files



1. Build a vector/list of filenames do you want to iterate over

```
map_df(.x =                    , .f = read_csv)
```

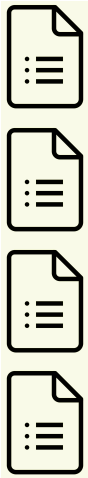
One dataframe from multiple files

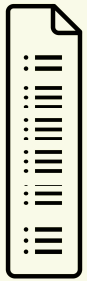


2. Select a **function** to read the files
3. Check its **arguments**
4. Select **map/map2/pmap**

Return value: a dataframe

map_df(.x =  , .f = read_csv)





5. Select map suffix in relation with the expected output

purrr::map( )

Let's practise

Which *map()* function should you use to obtain these return values?

_____(1:3, typeof)

[[1]]
[1] "integer"

[[2]]
[1] "integer"

[[3]]
[1] "integer"

_____(1:3, is.numeric)

[[1]]
[1] TRUE

[[2]]
[1] TRUE

[[3]]
[1] TRUE

_____(1:3, is.numeric)

[1] TRUE TRUE TRUE

_____(1:3, typeof)

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

_____(mtcars, typeof)

```
# A tibble: 1 x 11
  mpg    cyl    disp   hp     drat   wt     qsec   vs     am     gear   carb
  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>
1 double double double double double double double double double double double
```

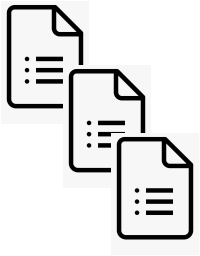
Reading multiple files

live coding #1

1. Build a vector of **filenames** do you want to iterate over
2. Select a **function** to read the files
3. Check the number of **arguments** the function need
4. Select among **map()/map2()/pmap()**
5. Select map_*() suffix in relation of the **desired output**

1. Obtain a dataframe from multiple files
2. Automatize report generation

Iterate using a function for its side effects

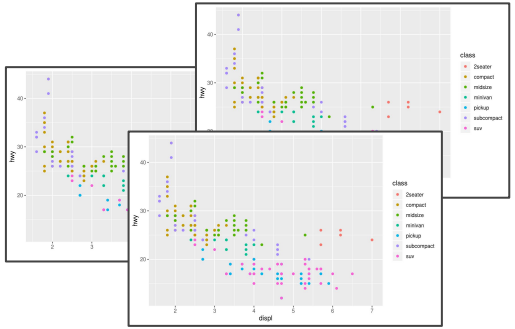


saving multiple files

```
$x
 [1]  4  4  7  7  8  9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14
[22] 14 14 15 15 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20
[43] 20 22 23 24 24 24 24 25

$y
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26
[17]  34  34  46  26  36  60  80  20  26  54  32  40  32  40  50  42
[33]  56  76  84  36  46  68  32  48  52  56  64  66  54  70  92  93
[49] 120  85
```

print on screen a big number of results



obtain several plots

Remember purrr::walk() family of functions

walk(.l, .f, ...)

A list of vectors
The length of .l determines the number of arguments that .f will be called with.

Function
Fórmula
Atomic vector.

Let’s practise

Which of these functions you would use with *walk()* family of functions?

- **purrr::safely()**
- **~ggplot(., aes(mpg, wt)) + geom_point())**
- **rmarkdown::render()**
- **~list(name = .)**

Can you explain your choice on using map() or walk() to iterate with these functions?

Generation of reports with RMarkdown

F:/certificacion - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

live_coding.R | reports | reports[[3]][[1]] | report_latam.Rmd* | file_genera

Knit Insert Run

```
---
title: "report_latam"
author: "Florencia"
date: "9/12/2019"
output: html_document
params:
  country: "Argentina"
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
```

Report
```

Automatization of report generation

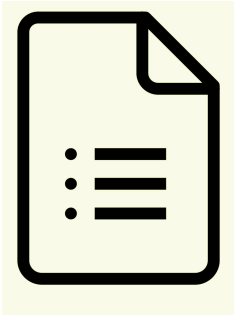
Template.Rmd

yaml

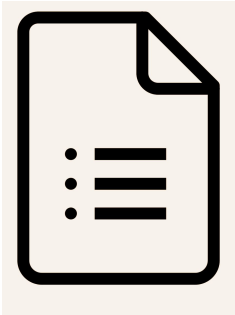
chunk

Parameter list

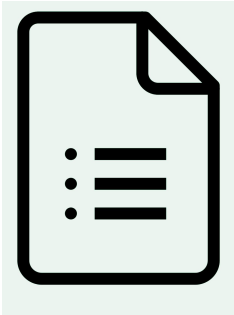
- red
- green
- blue
- yellow



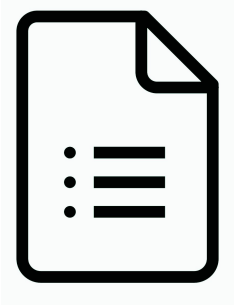
rep_yellow.html



rep_red.html



rep_blue.html



rep_green.html

Automatization of report generation live coding#2

Generate the different reports!

```
purrr::pwalk(rmarkdown::render,
             input = "Template.Rmd"
             output_file =  "rep_yellow.htm"  "rep_red.html"  "rep_blue.html" ...
             params =                          )
```

Parameter list

- red
- green
- blue
- yellow

Automatization of report generation

RMarkdown template file -> add parameters in YAML (.Rmd) and in the code

```
---
title: "nationa_report"
author: "Florencia"
date: "9/12/2019"
output: html_document
params:
  country: "Argentina"
---
```

YAML

```{r read, include=FALSE}

file <- read_csv(str_c(params$country, ".csv", sep = ""))

```

CHUNK

Let's practise

How would you modify this list for automatizing RMarkdown report generation?

Can you explain the changes?

1. Build a vector of **filenames** do you want to iterate over
2. Select a **function** to read the files
3. Check the number of **arguments** the function need
4. Select among **map()/map2()/pmap()**
5. Select map_*() suffix in relation of the **desired output**