# The Tidyverse

## Become a R super user

Axel Thieffry, PhD

October 2021

# Agenda

- Base R *vs* The Tidyverse
- `magrittr` and the pipe(s)
- Tibbles and tidylog
- Diving into CAGE data

# Base R
# *vs*
# The Tidyverse

# The Tidyverse

## What is it?

- Collection of R libraries meant to:
  - facilitate and streamline data wrangling
  - ease programming
  - improve readability

### ggplot2

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Go to docs...

### dplyr

dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges. Go to docs...

### tidyr

tidyr provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable. Go to docs...

### readr

readr provides a fast and friendly way to read rectangular data (like csv, tsv, and fwf). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes. Go to docs...

### purrr

purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors. Once you master the basic concepts, purrr allows you to replace many for loops with code that is easier to write and more expressive. Go to docs...

### tibble

tibble is a modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not. Tibbles are data.frames that are lazy and surly: they do less and complain more forcing you to confront problems earlier, typically leading to cleaner, more expressive code. Go to docs...

### stringr

stringr provides a cohesive set of functions designed to make working with strings as easy as possible. It is built on top of stringi, which uses the ICU C library to provide fast, correct implementations of common string manipulations. Go to docs...

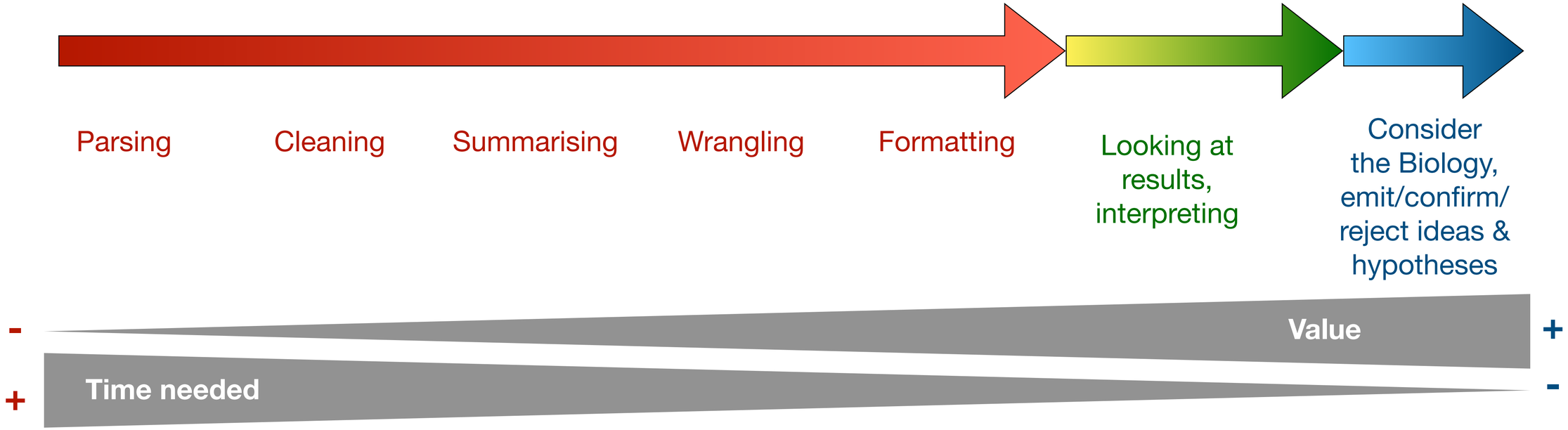
### forcats

forcats provides a suite of useful tools that solve common problems with factors. R uses factors to handle categorical variables, variables that have a fixed and known set of possible values. Go to docs...

# The Tidyverse

## Why?

- Typical timeline of a Data Science/Bioinformatics question:



# The Tidyverse

## Why?

```
Tidyverse
```

- Easier coding
- Faster coding
- Improved readability
- Somewhat standardised code

→

- Less time doing monkey-work
- More time spent on the high-value activities
- Make yourself **valuable**

# The Tidyverse

## Great website, plenty of tutorials and support online

https://www.tidyverse.org/

### Core tidyverse

The core tidyverse includes the packages that you're likely to use in everyday data analyses. As of tidyverse 1.3.0, the following packages are included in the core tidyverse:

**ggplot2**

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Go to docs...

**dplyr**

dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges. Go to docs...

**tidyr**

tidyr provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable. Go to docs...

**readr**

readr provides a fast and friendly way to read rectangular data (like csv, tsv, and fwf). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes. Go to docs...

**purrr**

purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors. Once you master the basic concepts, purrr allows you to replace many for loops with code that is easier to write and more expressive. Go to docs...

**tibble**

tibble is a modern re-imagining of the data frame, keeping what time has proven to be effective, and throwing out what it has not. Tibbles are data.frames that are lazy and surly: they do less and complain more forcing you to confront problems earlier, typically leading to cleaner, more expressive code. Go to docs...

👌✨

ggplot2 part of the tidyverse 3.3.5 Search... Reference N

### Overview

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

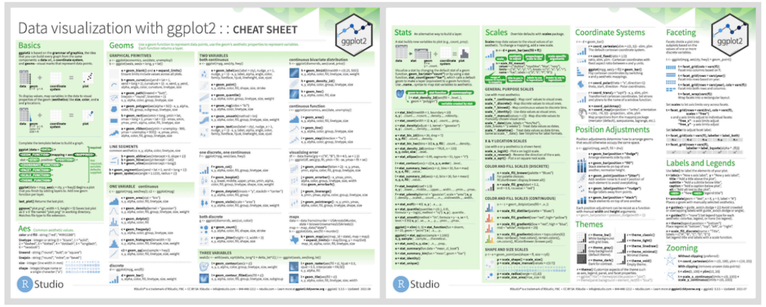
### Installation

```
# The easiest way to get ggplot2 is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just ggplot2:
install.packages("ggplot2")

# Or the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/ggplot2")
```

### Cheatsheet



# The Tidyverse

1) **Start RStudio**
2) **Load Tidyverse:** `library(tidyverse)`

# Base R *vs* Tidyverse

## What do these do?

```r
iris[order(iris$Sepal.Length), ]

dup <- duplicated(iris$Sepal.Length)
not_dup <- !duplicated(iris$Sepal.Length)
iris[not_dup, ]

iris$Sepal.Length==7.0
which(iris$Sepal.Length==7.0)
iris[which(iris$Sepal.Length==7.0), ]

iris$Sepal.Area <- iris$Sepal.Length * iris$Sepal.Width

iris['Sepal.Length']
iris[['Sepal.Length']]

names(iris)[names(iris) == 'Sepal.Area'] <- 'Sepal.cm2'

iris[, grepl("^Sep", names(iris))]
```

# Base R *vs* Tidyverse

## Compare:

**Base R**

```r
iris[order(iris$Sepal.Length), ]

dup <- duplicated(iris$Sepal.Length)
not_dup <- !duplicated(iris$Sepal.Length)
iris[not_dup, ]

iris$Sepal.Length==7.0
which(iris$Sepal.Length==7.0)
iris[which(iris$Sepal.Length==7.0), ]

iris$Sepal.Area <- iris$Sepal.Length * iris$Sepal.Width

iris['Sepal.Length']
iris[['Sepal.Length']]

names(iris)[names(iris) == 'Sepal.Area'] <- 'Sepal.cm2'

iris[, grepl("^Sep", names(iris))]
```

**Tidyverse (dplyr)**

```r
arrange(iris, Sepal.Length)

distinct(iris, Sepal.Length)



filter(iris, Sepal.Length==7)



mutate(iris, 'Sepal.Area'=Sepal.Length * Sepal.Width)

select(iris, Sepal.Length)
pull(iris, Sepal.Length)

rename(iris, 'Sepal.Area'='Sepal.cm2')

select(iris, matches('^Sep'))
```

# `magrittr` and the pipe(s)

# magrittr

## pipes in R

- <u>As you all know, pipe:</u>

  **output from program #1 becomes input for program #2**

- Streamline code
- Avoid unnecessary intermediate files/variables
- Save (disk/mental) space
- Improve code readability (avoid nested functions)
  follows a left-to-right flow instead of a eccentric flow

# magrittr

## pipes in R

- Eccentric code ex:

```
eat(wait(pan(mix(break(get_eggs(n=6), all=T), manual=T), heat=3), n=4, unit='min'))
```

- Piped code ex:

```
get_eggs(n=6) | break(all=T) | mix(manual=T) | pan(heat=3) | wait(n=4, unit='min') | eat()
```

# `magrittr`

## pipes in R

- `maggritr` pipe: `%>%`
- Using a pipe in R will make the next line indented, this is purely visual help
- Available with `library(tidyverse)` or `library(magrittr)`
- TLDR: `f(x)` can be rewritten into **`x %>% f()`**

# magrittr

## pipes in R

- Example

What does this code do?

**Base R**

```
sum(iris[iris$Sepal.Length > 7, ]$Species == 'virginica')
```

**Tidyverse**

```
iris %>%
    filter(Sepal.Length > 7) %>%
    count(Species)
```

# magrittr

## Other forms of pipe:

- %<>%
- input is passed to expression, output is reassigned to input variable

```
x ← f(x)
x %<>% f()
```

- Ex:

Apply the magrittr double pipe to the following example:

```
iris ← iris %>% filter(Sepal.Length > 7)

iris %<>% filter(Sepal.Length > 7)
```

# magrittr

## Other forms of pipe:

- %$%
- exposes the names of the left-hand side object to the right-hand side expression
- Ex:

```
iris %>%
  subset(Sepal.Length > 7) %>%
  pull(Petal.Length) %>%
  mean()

iris %>%
  subset(Sepal.Length > 7) %$%
  mean(Petal.Length)
```

# Note

## base-R pipe

- `magrittr` has become so famous and widely used that the R-core team has recently decided to make the pipe a feature in base-R (requires R 4.1+)
- base-R pipe has to be activated in the RStudio (***Preferences > Code > General***)
- base-R pipe is:
  Normal font: `|>`
  FiraCode: ▷

```
iris ▷ filter(Sepal.Length > 7)
```

- But there isn't any `%<>%` or `%$%` yet…

# Tibbles

# Tibbles

## A tidy table

- The Tidyverse has a library called `tibble`, who's main two functions are
  `tibble()`
  `as_tibble()`

- Those **create** or **coerce** a data frame into a tibble,
  which is essentially exactly the same **BUT**:

  - tibbles print way better than data frame

  - tibbles cannot have row names

# Tibbles

## Comparison

**data.frame**

```
Untitled1*
Source on Save
1  iris
2
2:1   (Top Level)
R 4.1.0 · ~/
112          6.4          2.7          5.3          1.9  virginica
113          6.8          3.0          5.5          2.1  virginica
114          5.7          2.5          5.0          2.0  virginica
115          5.8          2.8          5.1          2.4  virginica
116          6.4          3.2          5.3          2.3  virginica
117          6.5          3.0          5.5          1.8  virginica
118          7.7          3.8          6.7          2.2  virginica
119          7.7          2.6          6.9          2.3  virginica
120          6.0          2.2          5.0          1.5  virginica
121          6.9          3.2          5.7          2.3  virginica
122          5.6          2.8          4.9          2.0  virginica
123          7.7          2.8          6.7          2.0  virginica
124          6.3          2.7          4.9          1.8  virginica
125          6.7          3.3          5.7          2.1  virginica
126          7.2          3.2          6.0          1.8  vi
127          6.2          2.8          4.8          1.8  vi
128          6.1          3.0          4.9          1.8  vi
129          6.4          2.8          5.6          2.1  vi
130          7.2          3.0          5.8          1.6  vi
131          7.4          2.8          6.1          1.9  vi
132          7.9          3.8          6.4          2.0  vi
133          6.4          2.8          5.6          2.2  vi
134          6.3          2.8          5.1          1.5  vi
135          6.1          2.6          5.6          1.4  vi
136          7.7          3.0          6.1          2.3  vi
137          6.3          3.4          5.6          2.4  vi
138          6.4          3.1          5.5          1.8  vi
139          6.0          3.0          4.8          1.8  vi
140          6.9          3.1          5.4          2.1  vi
141          6.7          3.1          5.6          2.4  vi
142          6.9          3.1          5.1          2.3  vi
143          5.8          2.7          5.1          1.9  vi
144          6.8          3.2          5.9          2.3  vi
145          6.7          3.3          5.7          2.5  vi
146          6.7          3.0          5.2          2.3  vi
147          6.3          2.5          5.0          1.9  vi
148          6.5          3.0          5.2          2.0  vi
149          6.2          3.4          5.4          2.3  vi
150          5.9          3.0          5.1          1.8  vi
>
```

**tibble**

Automatically shows:
- Tibble dimensions
- Prints only the 10 first rows
- Data type of each column

```
> iris %>% as_tibble()
# A tibble: 150 × 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
 3          4.7         3.2          1.3         0.2 setosa
 4          4.6         3.1          1.5         0.2 setosa
 5          5           3.6          1.4         0.2 setosa
 6          5.4         3.9          1.7         0.4 setosa
 7          4.6         3.4          1.4         0.3 setosa
 8          5           3.4          1.5         0.2 setosa
 9          4.4         2.9          1.4         0.2 setosa
10          4.9         3.1          1.5         0.1 setosa
# … with 140 more rows
>
```

# Tibbles

## Comparison

- `mutate()`
  adds new variables (columns) to a data frame and keep existing ones

- `column_to_rownames()`
  “transfer” a data frame column to the data frame row names

```
> iris %>%
+     mutate('ID'=paste0('N_', 1:nrow(.))) %>%
+     column_to_rownames('ID') %>%
+     head()
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
N_1          5.1         3.5          1.4         0.2  setosa
N_2          4.9         3.0          1.4         0.2  setosa
N_3          4.7         3.2          1.3         0.2  setosa
N_4          4.6         3.1          1.5         0.2  setosa
N_5          5.0         3.6          1.4         0.2  setosa
N_6          5.4         3.9          1.7         0.4  setosa
> iris %>%
+     mutate('ID'=paste0('N_', 1:nrow(.))) %>%
+     column_to_rownames('ID') %>%
+     as_tibble()
# A tibble: 150 × 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
 3          4.7         3.2          1.3         0.2 setosa
 4          4.6         3.1          1.5         0.2 setosa
 5          5           3.6          1.4         0.2 setosa
 6          5.4         3.9          1.7         0.4 setosa
 7          4.6         3.4          1.4         0.3 setosa
 8          5           3.4          1.5         0.2 setosa
 9          4.4         2.9          1.4         0.2 setosa
10          4.9         3.1          1.5         0.1 setosa
# … with 140 more rows
>
```

# Tidylog

# Tidylog

## Tells you what happens

- `install.packages('devtools')`
  `devtools::install_github('elbersb/tidylog')`

- Collection of wrappers (mostly of `dplyr`) that output some stats on what you do

- Very useful to detect errors or incoherent behaviour (from you!) that would take a lot of time to figure out later on otherwise

```
> library(tidylog)

Attaching package: 'tidylog'

The following objects are masked from 'package:dplyr':

    add_count, add_tally, anti_join, count, distinct, distinct_all, distinct_at, distinct_if, filter, filter_all, filter_at,
    filter_if, full_join, group_by, group_by_all, group_by_at, group_by_if, inner_join, left_join, mutate, mutate_all,
    mutate_at, mutate_if, relocate, rename, rename_all, rename_at, rename_if, rename_with, right_join, sample_frac, sample_n,
    select, select_all, select_at, select_if, semi_join, slice, slice_head, slice_max, slice_min, slice_sample, slice_tail,
    summarise, summarise_all, summarise_at, summarise_if, summarize, summarize_all, summarize_at, summarize_if, tally,
    top_frac, top_n, transmute, transmute_all, transmute_at, transmute_if, ungroup

The following objects are masked from 'package:tidyr':

    drop_na, fill, gather, pivot_longer, pivot_wider, replace_na, spread, uncount

The following object is masked from 'package:stats':

    filter
```

# Tidylog

## Example:

```
> iris %>%
+     mutate('ID'=paste0('N_', 1:nrow(.))) %>%
+     filter(Sepal.Length > 7)
mutate: new variable 'ID' (character) with 150 unique values and 0% NA
filter: removed 138 rows (92%), 12 rows remaining
   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species    ID
1           7.1         3.0          5.9         2.1 virginica N_103
2           7.6         3.0          6.6         2.1 virginica N_106
3           7.3         2.9          6.3         1.8 virginica N_108
4           7.2         3.6          6.1         2.5 virginica N_110
5           7.7         3.8          6.7         2.2 virginica N_118
6           7.7         2.6          6.9         2.3 virginica N_119
7           7.7         2.8          6.7         2.0 virginica N_123
8           7.2         3.2          6.0         1.8 virginica N_126
9           7.2         3.0          5.8         1.6 virginica N_130
10          7.4         2.8          6.1         1.9 virginica N_131
11          7.9         3.8          6.4         2.0 virginica N_132
12          7.7         3.0          6.1         2.3 virginica N_136
>
```

tidylogs

# Tidylog

# Tidylog

## Tells you what happens

- `devtools::install_github('elbersb/tidylog')`
- Collection of wrappers (mostly of `dplyr`) that output some stats on what you do
- Very useful to detect errors or incoherent behaviour (from you!) that would take a lot of time to figure out later on otherwise

```
> library(tidylog)

Attaching package: 'tidylog'

The following objects are masked from 'package:dplyr':

    add_count, add_tally, anti_join, count, distinct, distinct_all, distinct_at, distinct_if, filter, filter_all, filter_at,
    filter_if, full_join, group_by, group_by_all, group_by_at, group_by_if, inner_join, left_join, mutate, mutate_all,
    mutate_at, mutate_if, relocate, rename, rename_all, rename_at, rename_if, rename_with, right_join, sample_frac, sample_n,
    select, select_all, select_at, select_if, semi_join, slice, slice_head, slice_max, slice_min, slice_sample, slice_tail,
    summarise, summarise_all, summarise_at, summarise_if, summarize, summarize_all, summarize_at, summarize_if, tally,
    top_frac, top_n, transmute, transmute_all, transmute_at, transmute_if, ungroup

The following objects are masked from 'package:tidyr':

    drop_na, fill, gather, pivot_longer, pivot_wider, replace_na, spread, uncount

The following object is masked from 'package:stats':

    filter
```

# Tidylog

## Example:

```
> iris %>%
+     mutate('ID'=paste0('N_', 1:nrow(.))) %>%
+     filter(Sepal.Length > 7)
mutate: new variable 'ID' (character) with 150 unique values and 0% NA
filter: removed 138 rows (92%), 12 rows remaining
   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species    ID
1           7.1         3.0          5.9         2.1 virginica N_103
2           7.6         3.0          6.6         2.1 virginica N_106
3           7.3         2.9          6.3         1.8 virginica N_108
4           7.2         3.6          6.1         2.5 virginica N_110
5           7.7         3.8          6.7         2.2 virginica N_118
6           7.7         2.6          6.9         2.3 virginica N_119
7           7.7         2.8          6.7         2.0 virginica N_123
8           7.2         3.2          6.0         1.8 virginica N_126
9           7.2         3.0          5.8         1.6 virginica N_130
10          7.4         2.8          6.1         1.9 virginica N_131
11          7.9         3.8          6.4         2.0 virginica N_132
12          7.7         3.0          6.1         2.3 virginica N_136
>
```

tidylogs

# More Tidyverse only makes sense with actual data: CAGE