

The Tidyverse

Become a R super user

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



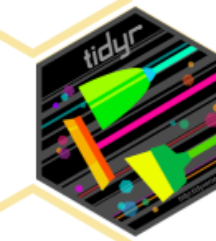
ggplot2

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...](#)



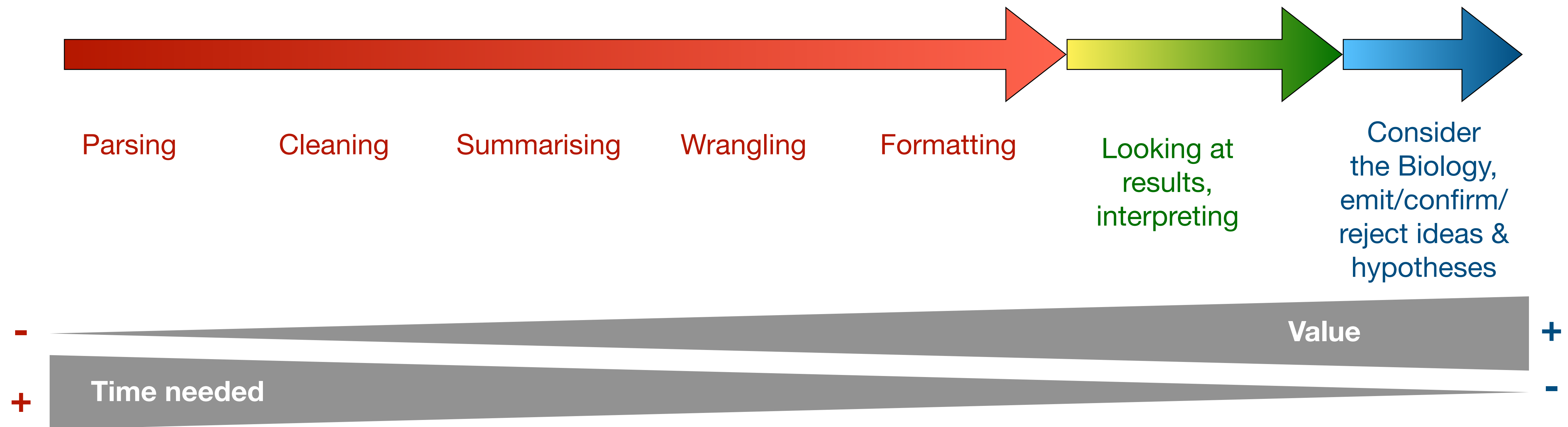
forcats

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:



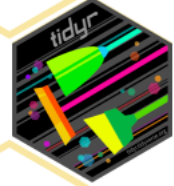
ggplot2

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...](#)



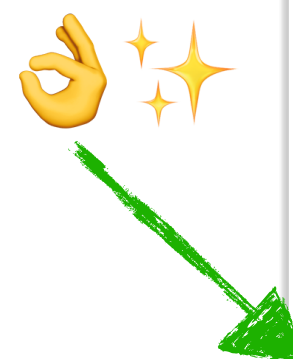
purrr

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...](#)





part of the tidyverse

3.3.5

Search...

Reference

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. Install Tidyverse (only once):

```
install.packages('tidyverse')
```

3. Load Tidyverse (quotes not necessary):

```
library(tidyverse)
```


Base R vs Tidyverse

What do these do?

NB: each bloc of code has a little goal in itself.

```
1  iris[order(iris$Sepal.Length), ]
2
3  dup ← duplicated(iris$Sepal.Length)
4  not_dup ← !duplicated(iris$Sepal.Length)
5  iris[not_dup, ]
6
7  iris$Sepal.Length=7.0
8  which(iris$Sepal.Length=7.0)
9  iris[which(iris$Sepal.Length=7.0), ]
10
11 iris$Sepal.Area ← iris$Sepal.Length * iris$Sepal.Width
12
13 iris['Sepal.Length']
14 iris[['Sepal.Length']]
15
16 names(iris)[names(iris) = 'Sepal.Area'] ← 'Sepal.cm2'
17
18 iris[, grepl("^Sep", names(iris))]
```

Base R vs Tidyverse

Compare:

Base R

```
1 iris[order(iris$Sepal.Length), ]
2
3 dup ← duplicated(iris$Sepal.Length)
4 not_dup ← !duplicated(iris$Sepal.Length)
5 iris[not_dup, ]
6
7 iris$Sepal.Length=7.0
8 which(iris$Sepal.Length=7.0)
9 iris[which(iris$Sepal.Length=7.0), ]
10
11 iris$Sepal.Area ← iris$Sepal.Length * iris$Sepal.Width
12
13 iris['Sepal.Length']
14 iris[['Sepal.Length']]
15
16 names(iris)[names(iris) = 'Sepal.Area'] ← 'Sepal.cm2'
17
18 iris[, grepl("^Sep", names(iris))]
```

Tidyverse (dplyr)

```
1 arrange(iris, Sepal.Length)
2
3 distinct(iris, Sepal.Length)
4
5
6
7 filter(iris, Sepal.Length=7)
8
9
10
11 mutate(iris, 'Sepal.Area'=Sepal.Length * Sepal.Width)
12
13 select(iris, Sepal.Length)
14 pull(iris, Sepal.Length)
15
16 rename(iris, 'Sepal.Area'='Sepal.cm2')
17
18 select(iris, matches('^Sep'))
```

Dataframe slicing

Select, filter, pull

`filter(iris,
 Sepal.Length=4.7)
(or subset())`

⇒ dataframe

`select(iris, Sepal.Length, Sepal.Width)`

⇒ dataframe

`pull(iris, Petal.Width)`

⇒ vector

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	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	5.0	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	5.0	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

magrittr and the pipe(s)

magrittr

pipes in R



- As you all know, pipe:
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 the natural left-to-right information flow instead of a eccentric flow

magrittr

pipes in R



- Example of eccentric code, e.g. base R:

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

- Same example, but with magrittr pipes:

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


magrittr

pipes in R



- **Simple** 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:



- **Double** pipe: `%<%`
- input is passed to expression...
and final output is reassigned to the initial input variable

```
x ← f(x)  
x %<% f()
```

- Ex: Apply the `magrittr` double pipe to the following example:

```
1 iris ← iris %>% filter(Sepal.Length > 7)  
2  
3 iris %<% filter(Sepal.Length > 7)
```

magrittr

Other forms of pipe:




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

```
1  iris %>%
2    subset(Sepal.Length > 7) %>%
3    pull(Petal.Length) %>%
4    mean()
5
6  iris %>%
7    subset(Sepal.Length > 7) %$%
8    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 a pipe feature in base-R (requires R 4.1+)
- base-R pipe has to be activated in RStudio (***Preferences > Code > General***)
- base-R pipe is:
Normal font: `|>`
FiraCode: `▷`
A screenshot of a code editor with a dark blue background. It shows the R code `iris ▷ filter(Sepal.Length > 7)` where the pipe operator `▷` is a light blue right-pointing triangle, `filter` is orange, `Sepal.Length` is white, `>` is light blue, and `7` is red.
- But there isn't any `%◇%` or `%%$%`, yet...

Tibbles

Tibbles

A tidy table

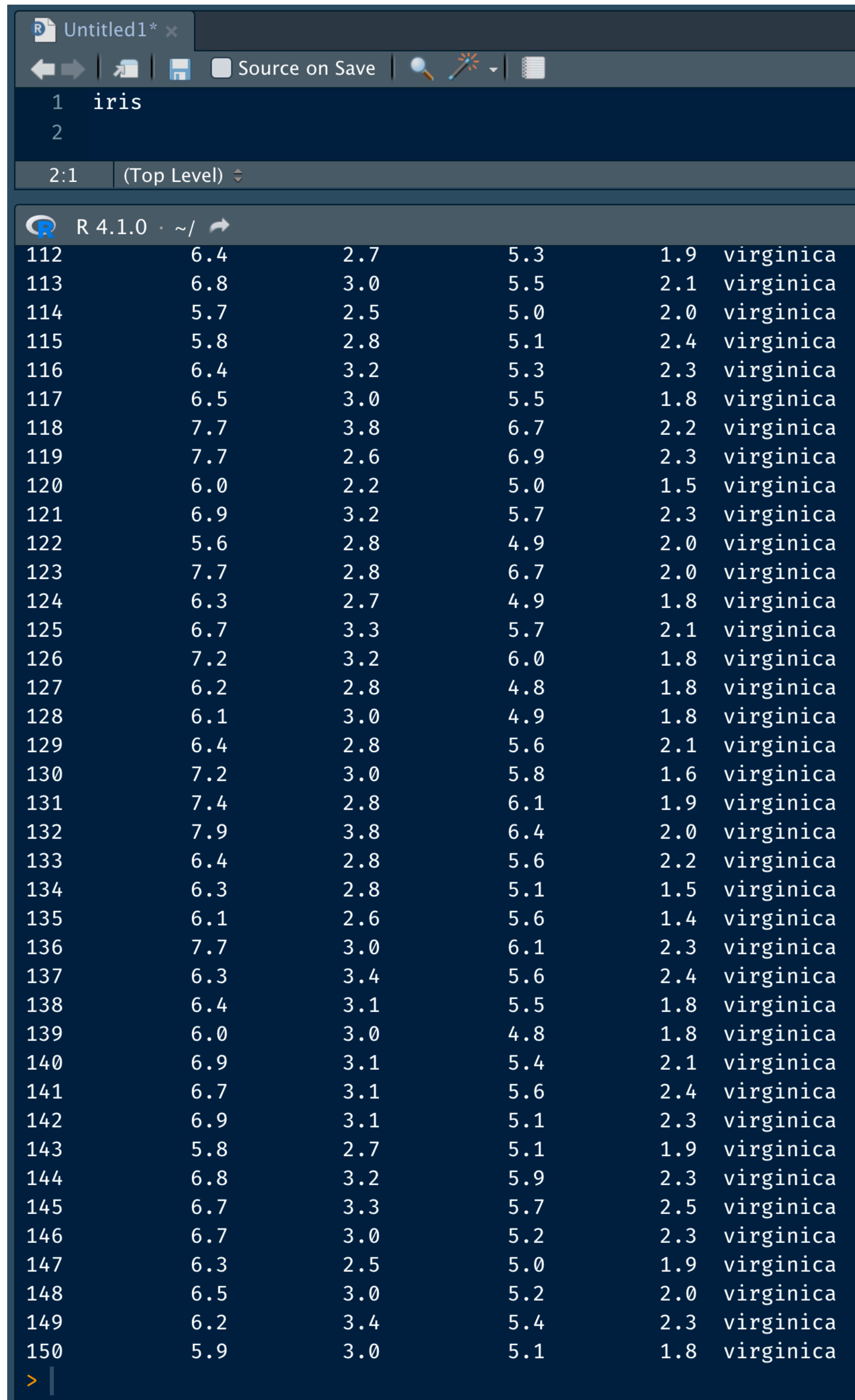


- The Tidyverse has a library called `tibble`, who's main functions are:
 - `tibble()`
to create a tibble
 - `as_tibble()`
to coerce a dataframe into a tibble
- Tibbles vs data frames: essentially the same, except...
 - tibbles printing format is way better than data frame
 - tibbles cannot have rownames, row names are apparently considered a bad practice...

Tibbles

Comparison

data.frame



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	virginica
127	6.2	2.8	4.8	1.8	virginica
128	6.1	3.0	4.9	1.8	virginica
129	6.4	2.8	5.6	2.1	virginica
130	7.2	3.0	5.8	1.6	virginica
131	7.4	2.8	6.1	1.9	virginica
132	7.9	3.8	6.4	2.0	virginica
133	6.4	2.8	5.6	2.2	virginica
134	6.3	2.8	5.1	1.5	virginica
135	6.1	2.6	5.6	1.4	virginica
136	7.7	3.0	6.1	2.3	virginica
137	6.3	3.4	5.6	2.4	virginica
138	6.4	3.1	5.5	1.8	virginica
139	6.0	3.0	4.8	1.8	virginica
140	6.9	3.1	5.4	2.1	virginica
141	6.7	3.1	5.6	2.4	virginica
142	6.9	3.1	5.1	2.3	virginica
143	5.8	2.7	5.1	1.9	virginica
144	6.8	3.2	5.9	2.3	virginica
145	6.7	3.3	5.7	2.5	virginica
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica



tibble

Automatically shows:

- dimensions
- only the 10 first rows
- class of each column
- NA's in red
- simplify big numbers if not enough printing space
- clips column names but tell you about it

```
> iris %>% as_tibble()
# A tibble: 150 × 5
  Sepal.Length Sepal.Width Petal...¹ Petal...² Species
      <dbl>         <dbl>   <dbl>    <dbl> <fct>
1         5.1      3.5e12     1.4      0.2 setosa
2        NA        3 e12      1.4      0.2 setosa
3        NA      3.2e12     1.3      0.2 setosa
4        NA      3.1e12     1.5      0.2 setosa
5         5        3.6e12     1.4      0.2 setosa
6         5.4      3.9e12     1.7      0.4 setosa
7        NA      3.4e12     1.4      0.3 setosa
8         5        3.4e12     1.5      0.2 setosa
9        NA      2.9e12     1.4      0.2 setosa
10       NA      3.1e12     1.5      0.1 setosa
# ... with 140 more rows, and abbreviated variable
#   names ¹Petal.Length, ²Petal.Width
# i Use `print(n = ...)` to see more rows
>
```

Tibbles

Comparison

- Explains the difference between the two blocs of code. What happened?
- `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

Simply tells you what happened

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

```
> 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 output