Introduction to the Tidyverse

How to be a tidy data scientist

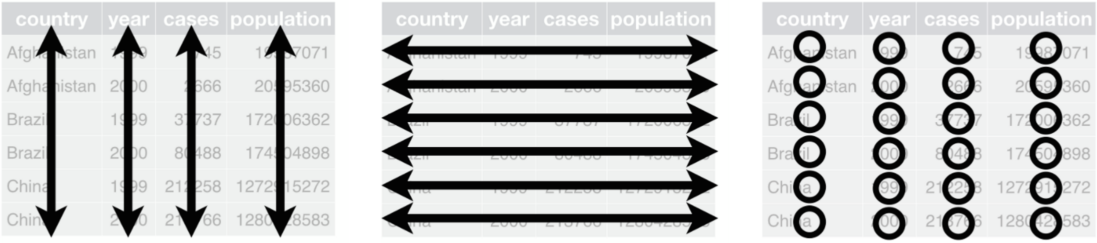
Olivier Gimenez
2019-01-14 (updated: 2019-01-15)

Tidyverse

- *Tidy* pour "bien rangé" et *verse* pour "univers"
- A collection of R 📦 developed by H. Wickham and others at Rstudio

Tidyverse

- "Un modèle d’organisation des données qui vise à faciliter le travail souvent long et fastidieux de nettoyage et de préparation préalable à la mise en oeuvre de méthodes d’analyse" (Julien Barnier).
- Les principes d’un tidy data sont :
  - chaque variable est une colonne
  - chaque observation est une ligne
  - chaque type d’observation est dans une table différente



variables observations values

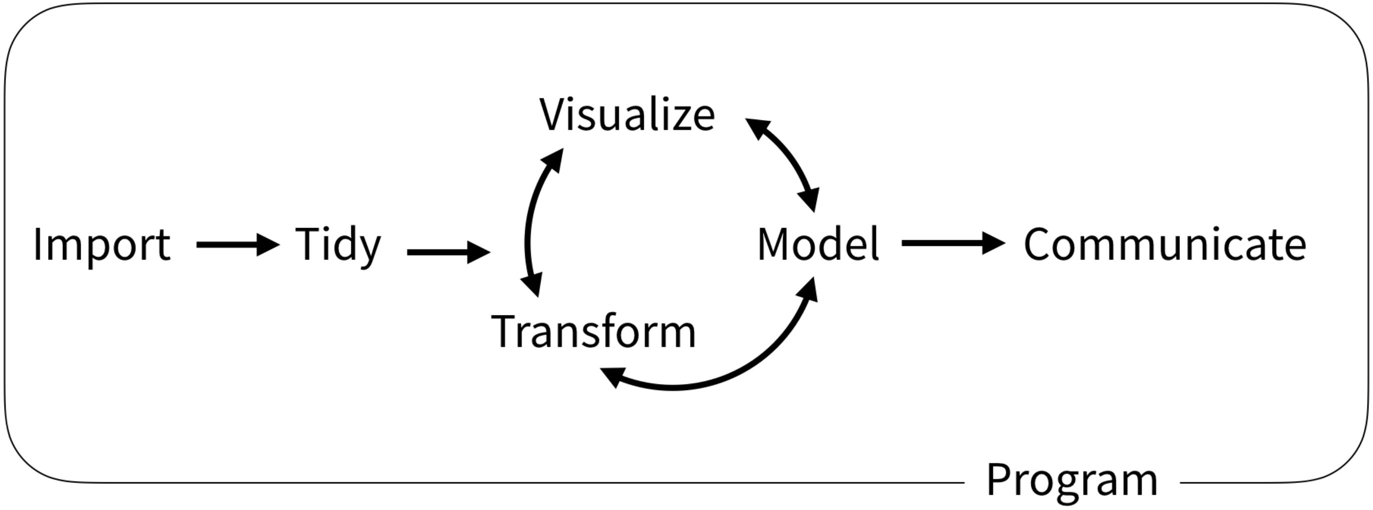
Tidyverse is a collection of R 📦

- `ggplot2` - visualisation
- `dplyr` - manipulation et synthèse des données
- `tidyr` - manipulation des données
- `purrr` - programmation avancée
- `readr` - importation de données
- `tibble` - tableaux de données data.frame améliorés
- `forcats` - variables qualitatives
- `stringr` - chaînes de caractères

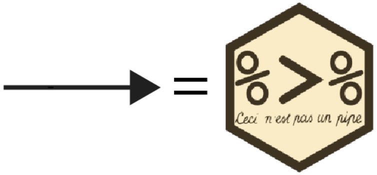
Tidyverse is a collection of R 📦

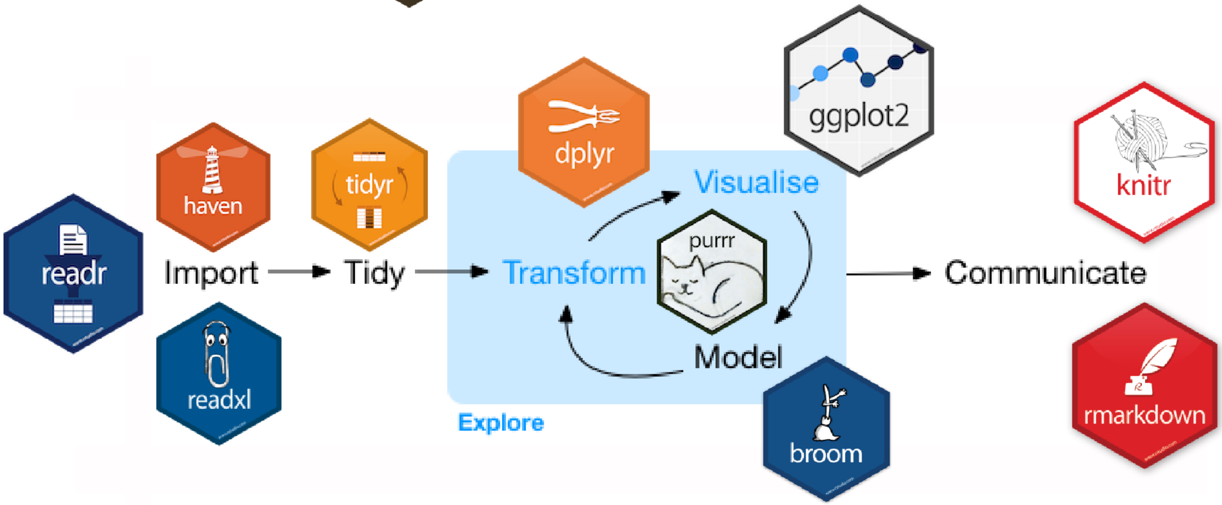
- `ggplot2` - visualisation
- `dplyr` - manipulation et synthèse des données
- `tidyr` - manipulation des données
- `purrr` - programmation avancée
- `readr` - importation de données
- `tibble` - tableaux de données data.frame améliorés
- `forcats` - variables qualitatives
- `stringr` - chaînes de caractères

Workflow in data science



Workflow in data science, with **Tidyverse**





Import → Tidy → Transform

Visualise

Model

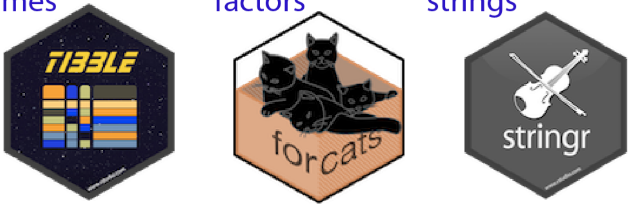
Explore

→ Communicate

data.frames

factors

strings



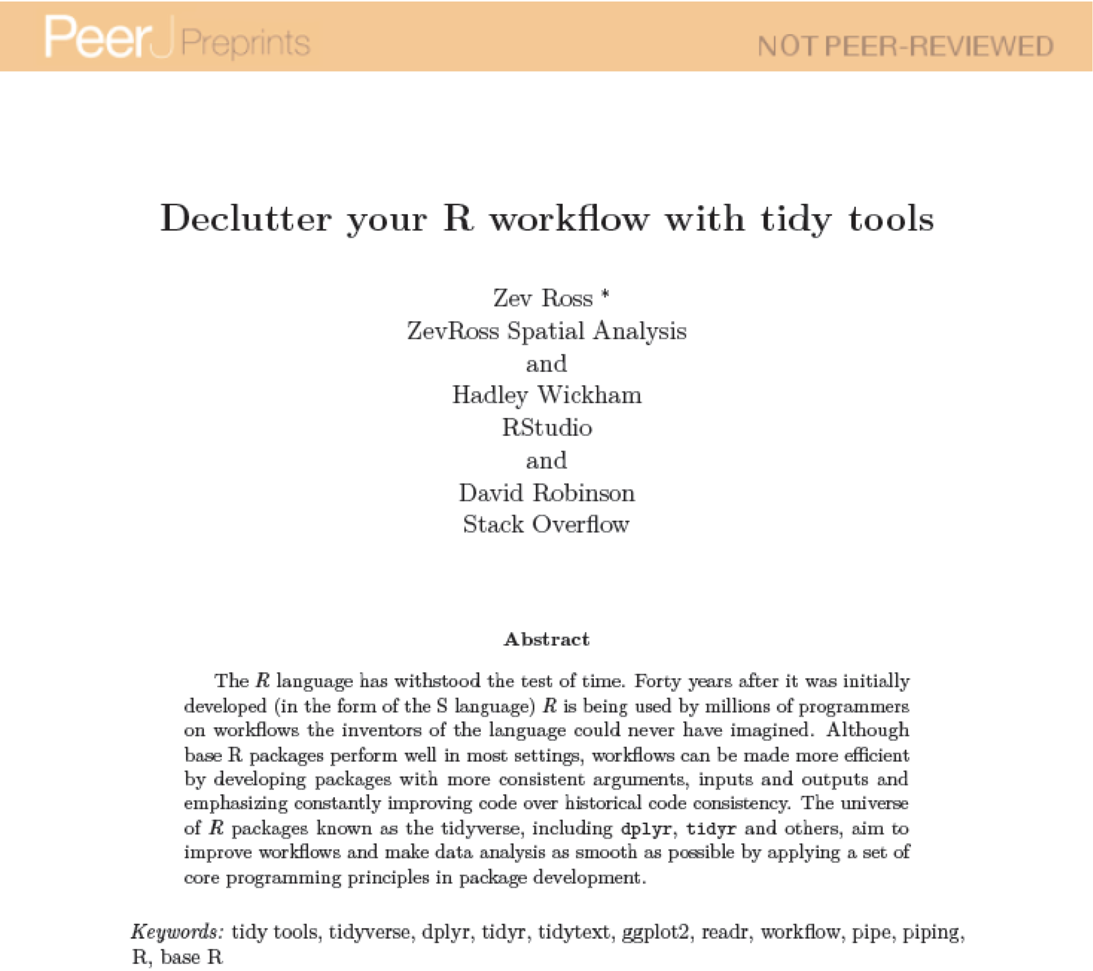
Load tidyverse 📦

```r
#install.packages("tidyverse")
library(tidyverse)
```

```
## ── Attaching packages ──────────────────────────────────

## ✔ ggplot2 3.1.0.9000     ✔ purrr   0.2.5
## ✔ tibble  2.0.1          ✔ dplyr   0.7.8
## ✔ tidyr   0.8.2          ✔ stringr 1.3.1
## ✔ readr   1.3.1          ✔ forcats 0.3.0

## Warning: package 'tibble' was built under R version 3.5.2

## ── Conflicts ───────────────────────────────────────────
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

Import, Tidy, Transform

Case study with Shakespeare's word usage



PeerJ Preprints NOT PEER-REVIEWED

Declutter your R workflow with tidy tools

Zev Ross *
ZevRoss Spatial Analysis
and
Hadley Wickham
RStudio
and
David Robinson
Stack Overflow

Abstract

The *R* language has withstood the test of time. Forty years after it was initially developed (in the form of the S language) *R* is being used by millions of programmers on workflows the inventors of the language could never have imagined. Although base R packages perform well in most settings, workflows can be made more efficient by developing packages with more consistent arguments, inputs and outputs and emphasizing constantly improving code over historical code consistency. The universe of *R* packages known as the tidyverse, including `dplyr`, `tidyr` and others, aim to improve workflows and make data analysis as smooth as possible by applying a set of core programming principles in package development.

Keywords: tidy tools, tidyverse, dplyr, tidyr, tidytext, ggplot2, readr, workflow, pipe, piping, R, base R

Import data

readr::read_csv function:

- keeps input types as is (no conversion to factor)
- creates `tibbles` instead of `data.frame`
    - pas de noms de lignes (rownames)
    - autorisent noms de colonnes avec caractères spéciaux ou nombres
    - s'affichent plus intelligemment que les data.frames
    - pas de partial matching sur noms de colonnes
    - avertissement si on essaie d’accéder à une colonne inexistante
- is damn fast 🏎

Import data

readr::read_csv function:

```
shakespeare <- read_csv('https://gist.githubusercontent.com/zross/ab72ba3250a3ee
shakespeare
```

```
## # A tibble: 164,656 x 5
##       X1 word        word_count corpus          corpus_date
##    <dbl> <chr>            <dbl> <chr>                 <dbl>
##  1     1 hive                 1 loverscomplaint        1609
##  2     2 plaintful            1 loverscomplaint        1609
##  3     3 Are                  1 loverscomplaint        1609
##  4     4 Than                 1 loverscomplaint        1609
##  5     5 attended             1 loverscomplaint        1609
##  6     6 That                 7 loverscomplaint        1609
##  7     7 moisture             1 loverscomplaint        1609
##  8     8 praised              1 loverscomplaint        1609
##  9     9 particular           1 loverscomplaint        1609
## 10    10 tend                 1 loverscomplaint        1609
## # … with 164,646 more rows
```

Group by variable to perform operation

dplyr::group_by function:

```
shakespeare_corpus <- group_by(shakespeare, corpus)
shakespeare_corpus
```

```
## # A tibble: 164,656 x 5
## # Groups:   corpus [42]
##       X1 word       word_count corpus          corpus_date
##    <dbl> <chr>           <dbl> <chr>                 <dbl>
##  1     1 hive                1 loverscomplaint        1609
##  2     2 plaintful           1 loverscomplaint        1609
##  3     3 Are                 1 loverscomplaint        1609
##  4     4 Than                1 loverscomplaint        1609
##  5     5 attended            1 loverscomplaint        1609
##  6     6 That                7 loverscomplaint        1609
##  7     7 moisture            1 loverscomplaint        1609
##  8     8 praised             1 loverscomplaint        1609
##  9     9 particular          1 loverscomplaint        1609
## 10    10 tend                1 loverscomplaint        1609
## # … with 164,646 more rows
```

Summarise data by group

dplyr::summarise function:

```
mean_shakespeare_corpus <- summarise(shakespeare_corpus,avg=mean(word_count))
mean_shakespeare_corpus
```

```
## # A tibble: 42 x 2
##    corpus                 avg
##    <chr>                <dbl>
##  1 1kinghenryiv          6.12
##  2 1kinghenryvi          5.24
##  3 2kinghenryiv          6.13
##  4 2kinghenryvi          5.80
##  5 3kinghenryvi          6.45
##  6 allswellthatendswell  6.23
##  7 antonyandcleopatra    5.96
##  8 asyoulikeit           6.28
##  9 comedyoferrors        5.70
## 10 coriolanus            6.35
## # … with 32 more rows
```

Cleaner code with "pipe" operator %>%

```
shakespeare %>%
  group_by(corpus) %>% # group by corpus
  summarise(avg = mean(word_count)) # compute mean
```

```
## # A tibble: 42 x 2
##    corpus                 avg
##    <chr>                <dbl>
##  1 1kinghenryiv          6.12
##  2 1kinghenryvi          5.24
##  3 2kinghenryiv          6.13
##  4 2kinghenryvi          5.80
##  5 3kinghenryvi          6.45
##  6 allswellthatendswell  6.23
##  7 antonyandcleopatra    5.96
##  8 asyoulikeit           6.28
##  9 comedyoferrors        5.70
## 10 coriolanus            6.35
## # … with 32 more rows
```

Compare to base R

```
with(shakespeare, tapply(word_count, corpus, mean))
```

```
##          1kinghenryiv          1kinghenryvi          2kinghenryiv
##              6.115590              5.240261              6.132682
##          2kinghenryvi          3kinghenryvi  allswellthatendswell
##              5.800769              6.449460              6.234996
##    antonyandcleopatra           asyoulikeit        comedyoferrors
##              5.960279              6.280468              5.700697
##            coriolanus             cymbeline                hamlet
##              6.347518              5.996103              6.101166
##          juliuscaesar            kinghenryv         kinghenryviii
##              6.263612              5.465125              6.200425
##              kingjohn              kinglear         kingrichardii
##              5.462972              5.853052              5.750000
##        kingrichardiii       loverscomplaint      loveslabourslost
##              6.761723              2.164017              5.474268
##               macbeth     measureformeasure      merchantofvenice
##              4.810592              6.155045              6.104977
##   merrywivesofwindsor midsummersnightsdream   muchadoaboutnothing
##              6.578976              5.093365              6.755714
##               othello  periclesprinceoftyre         rapeoflucrece
##              6.553688              5.286628              3.843687
##        romeoandjuliet               sonnets      tamingoftheshrew
##              6.191962              4.842263              6.044336
##               tempest         timonofathens       titusandronicus
##              4.838559              5.228195              5.663220
##    troilusandcressida          twelfthnight  twogentlemenofverona
##              5.805422              6.121392              5.923447
```

Deselect columns

dplyr::select function:

```
shakespeare %>%
  select(-X1, -corpus_date) # deselect columns
```

```
## # A tibble: 164,656 x 3
##    word        word_count corpus
##    <chr>            <dbl> <chr>
##  1 hive                 1 loverscomplaint
##  2 plaintful            1 loverscomplaint
##  3 Are                  1 loverscomplaint
##  4 Than                 1 loverscomplaint
##  5 attended             1 loverscomplaint
##  6 That                 7 loverscomplaint
##  7 moisture             1 loverscomplaint
##  8 praised              1 loverscomplaint
##  9 particular           1 loverscomplaint
## 10 tend                 1 loverscomplaint
## # … with 164,646 more rows
```

Select columns

dplyr::select function:

```
shakespeare %>%
  select(word, word_count, corpus) # select columns
```

```
## # A tibble: 164,656 x 3
##    word        word_count corpus
##    <chr>            <dbl> <chr>
##  1 hive                 1 loverscomplaint
##  2 plaintful            1 loverscomplaint
##  3 Are                  1 loverscomplaint
##  4 Than                 1 loverscomplaint
##  5 attended             1 loverscomplaint
##  6 That                 7 loverscomplaint
##  7 moisture             1 loverscomplaint
##  8 praised              1 loverscomplaint
##  9 particular           1 loverscomplaint
## 10 tend                 1 loverscomplaint
## # … with 164,646 more rows
```

Syntax with pipe

- Verb(Subject,Complement) replaced by Subject %>% Verb(Complement)
- No need to name unimportant intermediate variables
- Clear syntax (readability)

Create new column

dplyr::mutate function:

```
shakespeare %>%
  mutate(word = str_to_lower(word)) # convert words to lowercase
```

```
## # A tibble: 164,656 x 5
##       X1 word       word_count corpus          corpus_date
##    <dbl> <chr>           <dbl> <chr>                 <dbl>
##  1     1 hive                1 loverscomplaint        1609
##  2     2 plaintful           1 loverscomplaint        1609
##  3     3 are                 1 loverscomplaint        1609
##  4     4 than                1 loverscomplaint        1609
##  5     5 attended            1 loverscomplaint        1609
##  6     6 that                7 loverscomplaint        1609
##  7     7 moisture            1 loverscomplaint        1609
##  8     8 praised             1 loverscomplaint        1609
##  9     9 particular          1 loverscomplaint        1609
## 10    10 tend                1 loverscomplaint        1609
## # … with 164,646 more rows
```

Count number of words in each corpus

```
shakespeare %>%
  mutate(word = str_to_lower(word)) %>% # convert words to lowercase
  group_by(word, corpus) %>% # group by word within corpus
  summarize(n = sum(word_count)) # count
```

```
## # A tibble: 147,219 x 3
## # Groups:   word [?]
##    word  corpus                   n
##    <chr> <chr>                <dbl>
##  1 '     1kinghenryiv            33
##  2 '     1kinghenryvi            14
##  3 '     2kinghenryiv            38
##  4 '     2kinghenryvi            22
##  5 '     3kinghenryvi            26
##  6 '     allswellthatendswell    23
##  7 '     antonyandcleopatra      23
##  8 '     asyoulikeit             33
##  9 '     comedyoferrors          22
## 10 '     coriolanus              50
## # … with 147,209 more rows
```

Order stuff

dplyr::arrange function:

```
shakespeare %>%
  mutate(word = str_to_lower(word)) %>% # convert words to lowercase
  group_by(word, corpus) %>% # group by word within corpus
  summarize(n = sum(word_count)) %>% # count
  arrange(desc(n)) # decreasing order (wo desc for increasing)
```

```
## # A tibble: 147,219 x 3
## # Groups:   word [26,928]
##    word  corpus              n
##    <chr> <chr>           <dbl>
##  1 the   hamlet           1143
##  2 the   coriolanus       1127
##  3 the   kinghenryv       1085
##  4 and   kinghenryv       1004
##  5 the   2kinghenryiv      997
##  6 the   kingrichardiii    991
##  7 the   cymbeline         971
##  8 and   hamlet            969
##  9 and   2kinghenryvi      949
## 10 the   2kinghenryvi      949
## # … with 147,209 more rows
```

Nb of times a word occurs across all corpus

```
shakespeare %>%
  mutate(word = str_to_lower(word)) %>% # convert words to lowercase
  group_by(word) %>% # group by word
  summarize(n = sum(word_count),
            corpus = n_distinct(corpus)) %>% # count
  arrange(desc(n)) # decreasing order
```

```
## # A tibble: 26,928 x 3
##    word       n corpus
##    <chr> <dbl>  <int>
##  1 the   29801     42
##  2 and   27529     42
##  3 i     21029     42
##  4 to    20957     42
##  5 of    18514     42
##  6 a     15370     42
##  7 you   14010     42
##  8 my    12936     42
##  9 in    11722     42
## 10 that  11519     42
## # … with 26,918 more rows
```

Name this processed dataset

```r
shakespeare_processed <- shakespeare %>%
  mutate(word = str_to_lower(word)) %>% # convert words to lowercase
  group_by(word) %>% # group by word
  summarize(n = sum(word_count),
            corpus = n_distinct(corpus)) %>% # count
  arrange(desc(n)) # decreasing order
shakespeare_processed
```

```
## # A tibble: 26,928 x 3
##    word       n corpus
##    <chr> <dbl>  <int>
##  1 the   29801     42
##  2 and   27529     42
##  3 i     21029     42
##  4 to    20957     42
##  5 of    18514     42
##  6 a     15370     42
##  7 you   14010     42
##  8 my    12936     42
##  9 in    11722     42
## 10 that  11519     42
## # … with 26,918 more rows
```

Get rid of common words

dplyr::anti_join function:

```
words <- shakespeare_processed %>%
  group_by(word) %>% # group by word within corpus
  anti_join(tidytext::stop_words) %>% # returns all rows from
                                      # shakespeare that do not
                                      # match the stop words
  arrange(desc(n))
words
```

```
## # A tibble: 26,322 x 3
## # Groups:   word [26,322]
##    word      n corpus
##    <chr> <dbl>  <int>
##  1 thou   5771     42
##  2 thy    4269     42
##  3 thee   3383     42
##  4 lord   3357     40
##  5 king   3322     40
##  6 sir    3025     37
##  7 enter  2406     39
##  8 love   2249     42
##  9 hath   2032     42
## 10 'tis   1435     42
## # … with 26,312 more rows
```

Find words > 4 char not in all corpus

dplyr::filter function to select rows:

```
words <- words %>%
  filter(corpus < 42, nchar(word) > 4) %>%
  arrange(desc(n))
words
```

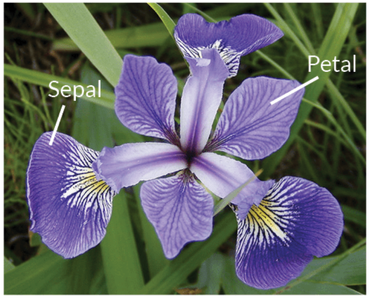
```
## # A tibble: 23,890 x 3
## # Groups:   word [23,890]
##    word        n corpus
##    <chr>   <dbl>  <int>
##  1 enter    2406     39
##  2 henry    1311     13
##  3 speak    1194     40
##  4 exeunt   1061     37
##  5 queen    1005     35
##  6 death     933     41
##  7 night     933     41
##  8 father    868     40
##  9 scene     825     38
## 10 master    803     39
## # … with 23,880 more rows
```

Check out other functions

- **spread()** and **gather()** from package `tidyr` to reshape tibbles
- **inner_join()**, **left_join()**, **right_join()**, **full_join()**, **semi_join()** and **anti_join()** from package `dplyr` to join two tibbles together
- **year()**, **month()**, etc… from package `lubridate` to manipulate dates
- Feel free to explore other Tidyverse packages, in particular `forcats` and `stringr`

Visualize

Case study on Fisher's iris dataset



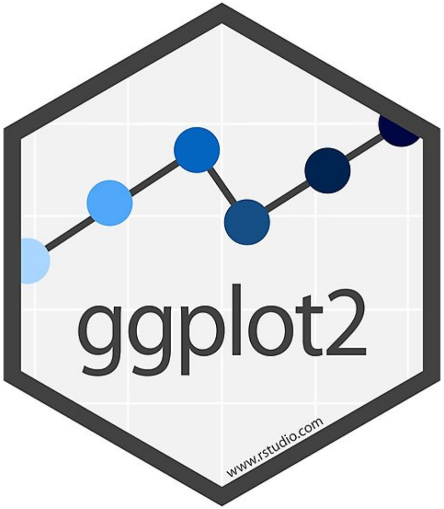
Iris Versicolor

Iris Setosa

Iris Virginica

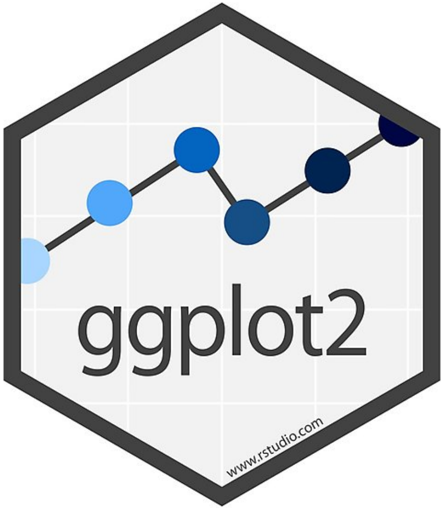
Visualization with ggplot2

- The package ggplot2 implements a **g**rammar of **g**raphics
- Operates on data.frames or tibbles, not vectors like base R
- Explicitly differentiates between the data and its representation



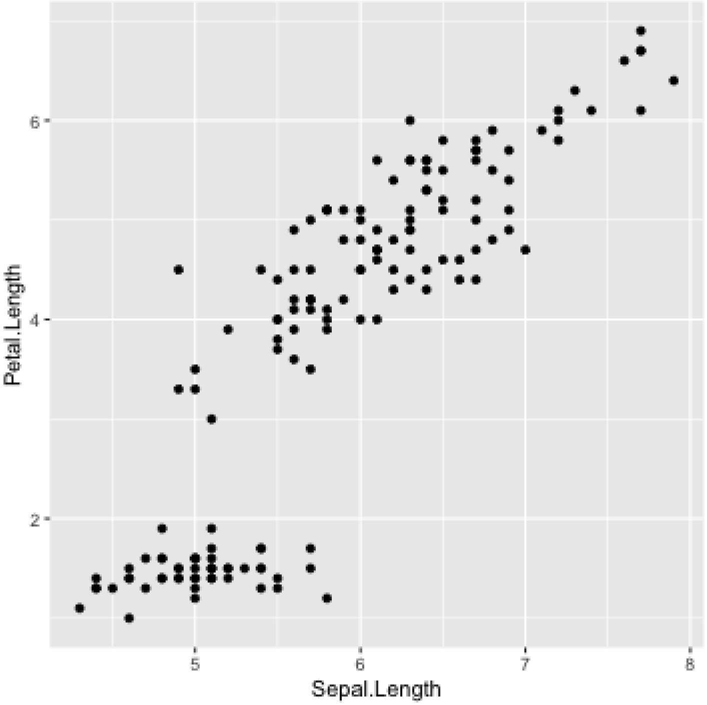
The ggplot2 grammar

Grammar element	What it is
Data	The data frame being plotted
Geometrics	The geometric shape that will represent the data (e.g., point, boxplot, histogram)
Aesthetics	The aesthetics of the geometric object (e.g., color, size, shape)



Scatterplots

```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length))
```



Scatterplots

```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length))
```

- Pass in the data frame as your first argument

Scatterplots

```r
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length))
```

- Pass in the data frame as your first argument
- Aesthetics maps the data onto plot characteristics, here x and y axes

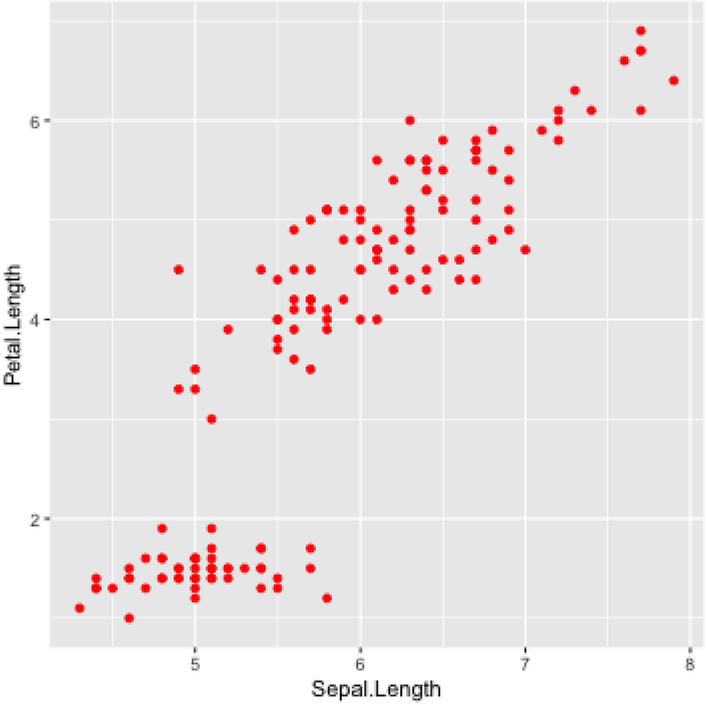
Scatterplots

```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length))
```

- Pass in the data frame as your first argument
- Aesthetics maps the data onto plot characteristics, here x and y axes
- Display the data geometrically as points

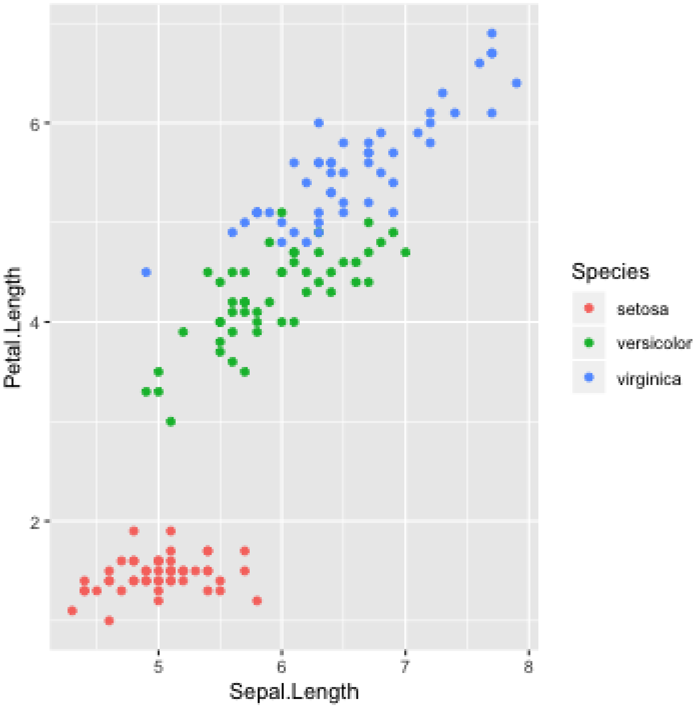
Scatterplots, with colors

```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length), color = "red")
```



Scatterplots, with species-specific colors

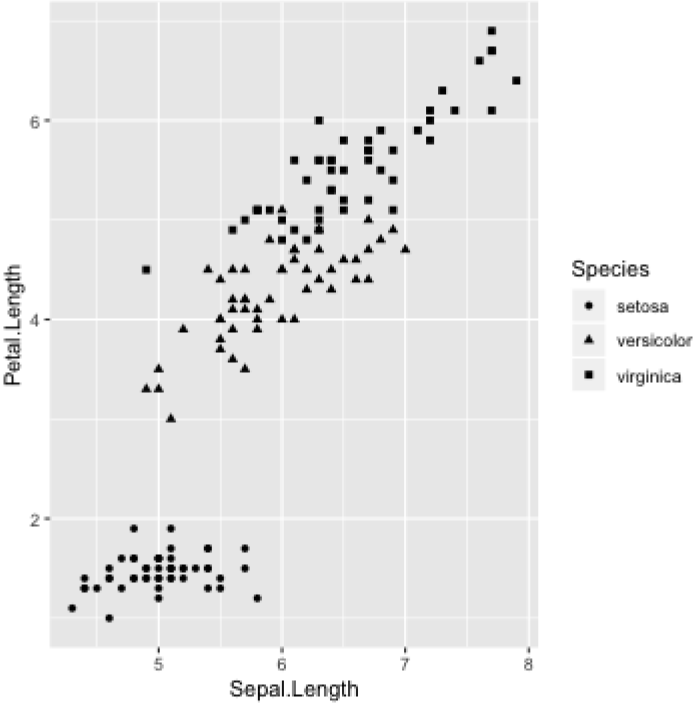
```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length, color = Species))
```



- Placing color inside aesthetic maps it to the data

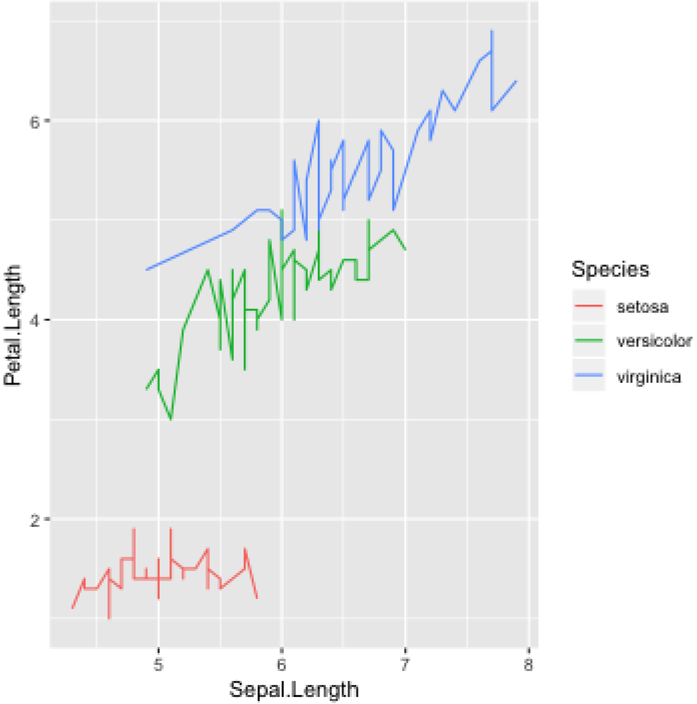
Scatterplots, with species-specific shapes

```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length, shape = Species))
```



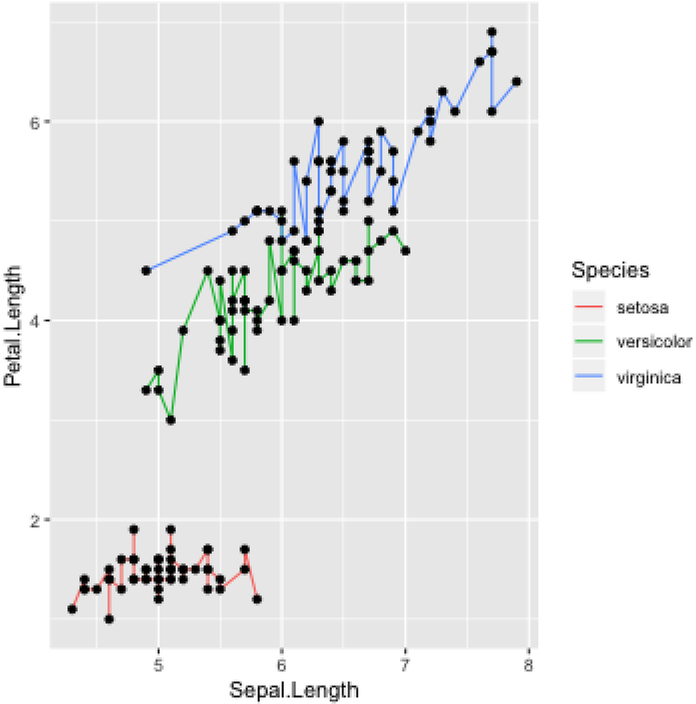
Scatterplots, lines instead of points

```
iris %>%
  ggplot() +
  geom_line(aes(x = Sepal.Length, y = Petal.Length, color = Species))
```



Scatterplots, add points

```
iris %>%
  ggplot() +
  geom_line(aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(aes(x = Sepal.Length, y = Petal.Length))
```

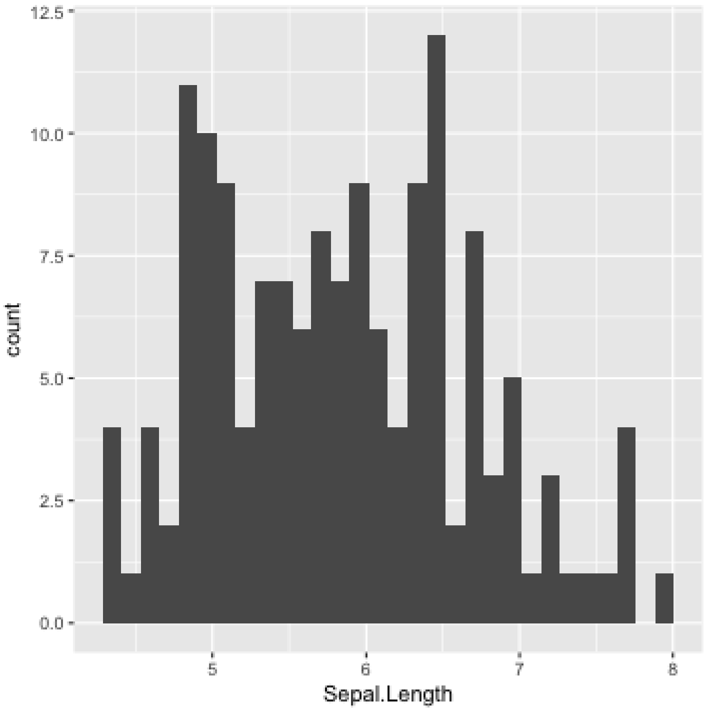


aes ou pas aes?

- Si on établit un lien entre les valeurs d'une variable et un attribut graphique, on définit un mappage, et on le déclare dans aes().
- Sinon, on modifie l'attribut de la même manière pour tous les points, et on le définit en dehors de la fonction aes().

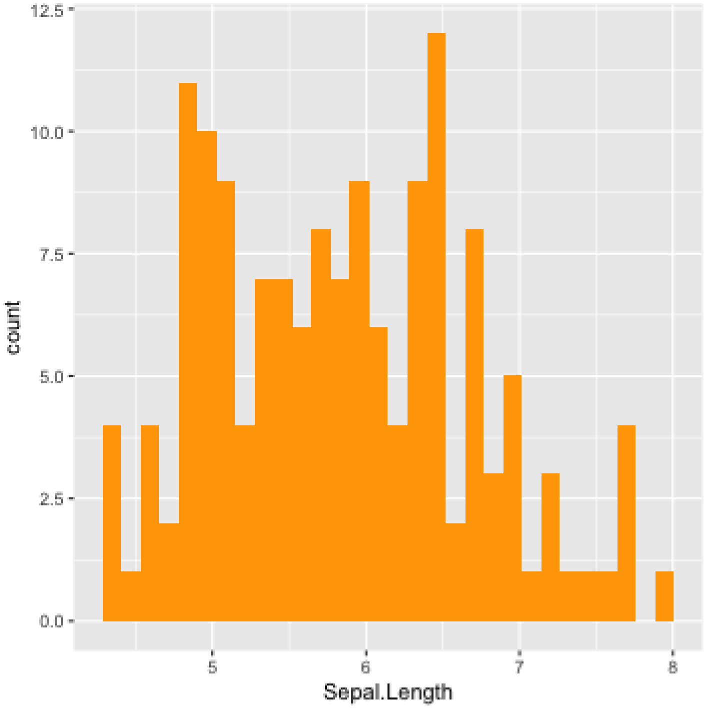
Histograms

```
iris %>%
  ggplot() +
  geom_histogram(aes(x = Sepal.Length))
```



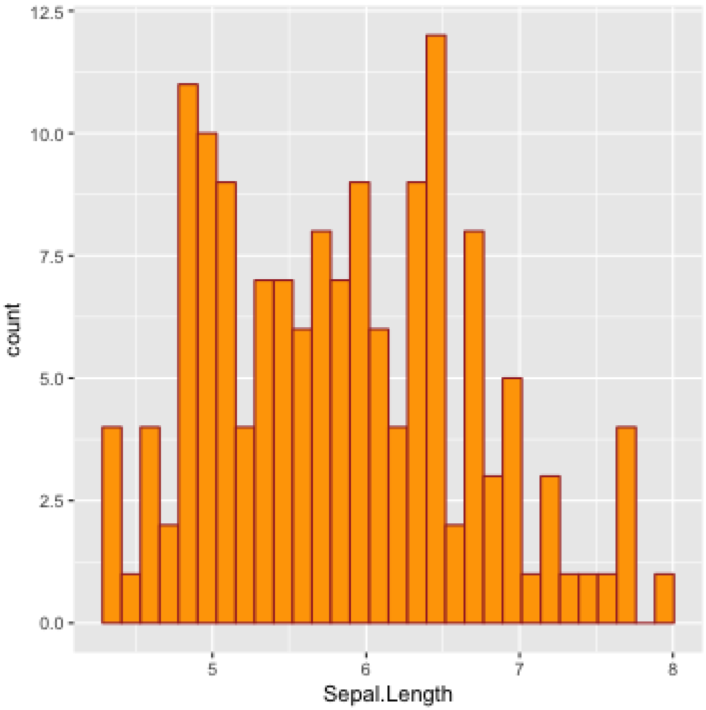
Histograms, with colors

```
iris %>%
  ggplot() +
  geom_histogram(aes(x = Sepal.Length), fill = "orange")
```



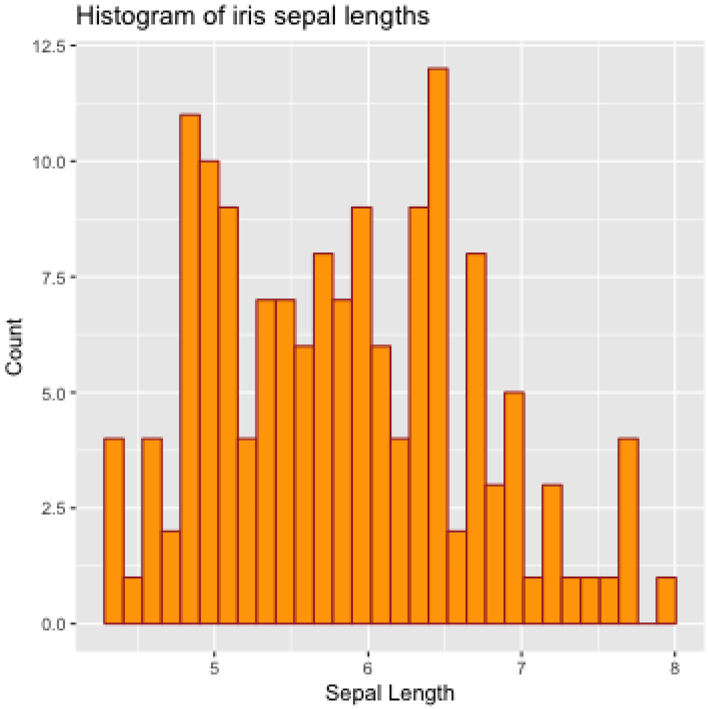
Histograms, with colors

```
iris %>%
  ggplot() +
  geom_histogram(aes(x = Sepal.Length), fill = "orange", color = "brown")
```



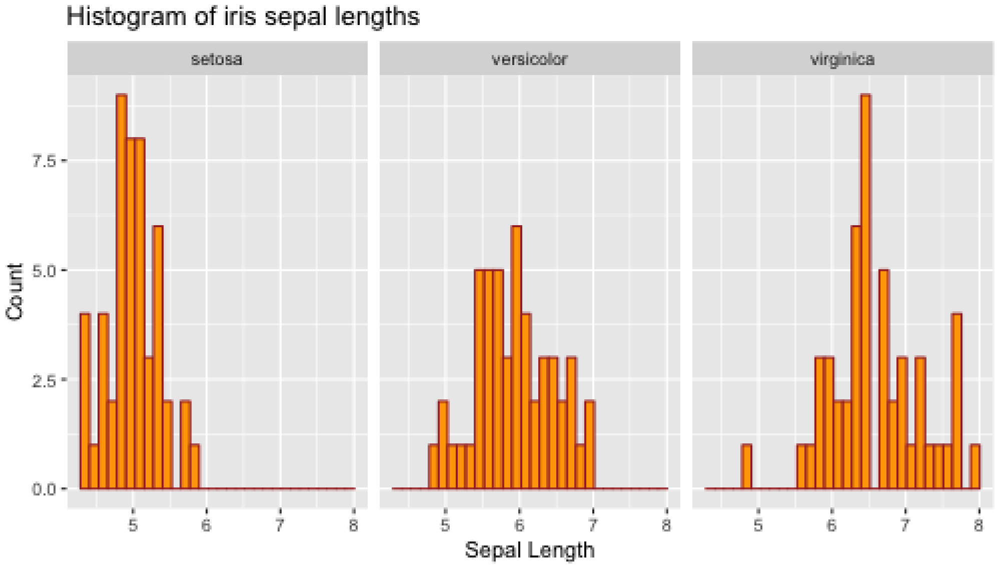
Histograms, with labels and title

```
iris %>%
  ggplot() +
  geom_histogram(aes(x = Sepal.Length), fill = "orange", color = "brown") +
  xlab("Sepal Length") +
  ylab("Count") +
  ggtitle("Histogram of iris sepal lengths")
```



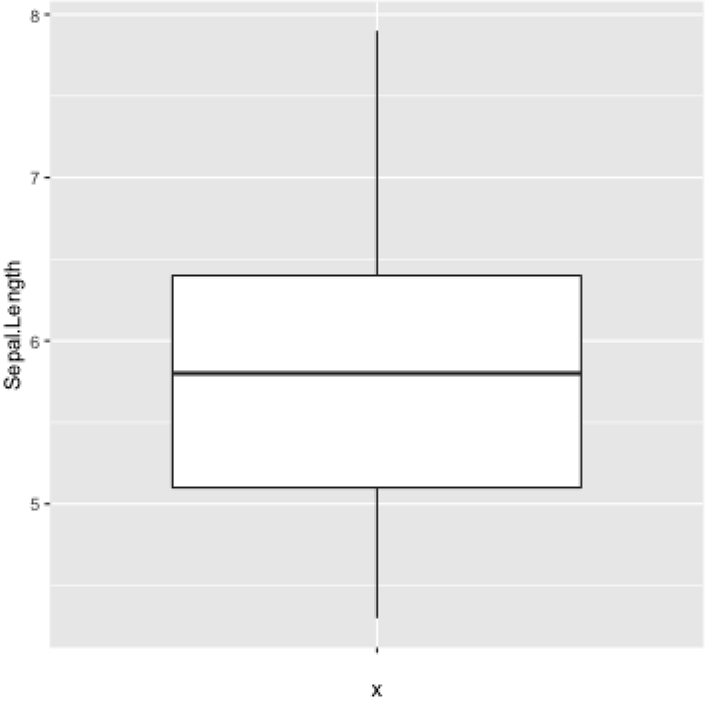
Histograms, by species

```
iris %>%
  ggplot() +
  geom_histogram(aes(x = Sepal.Length), fill = "orange", color = "brown") +
  xlab("Sepal Length") +
  ylab("Count") +
  ggtitle("Histogram of iris sepal lengths") +
  facet_wrap(aes(Species))
```



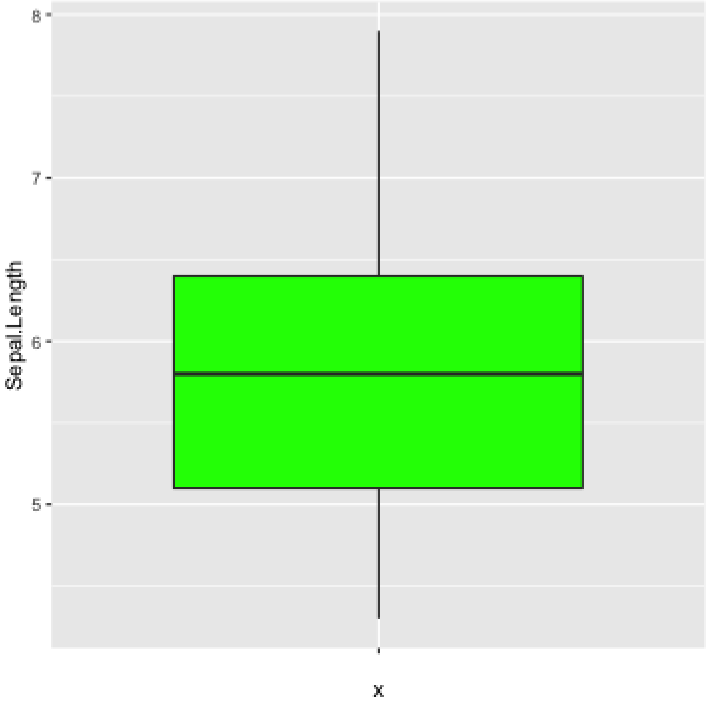
Boxplots

```
iris %>%
  ggplot() +
  geom_boxplot(aes(x = "", y = Sepal.Length))
```



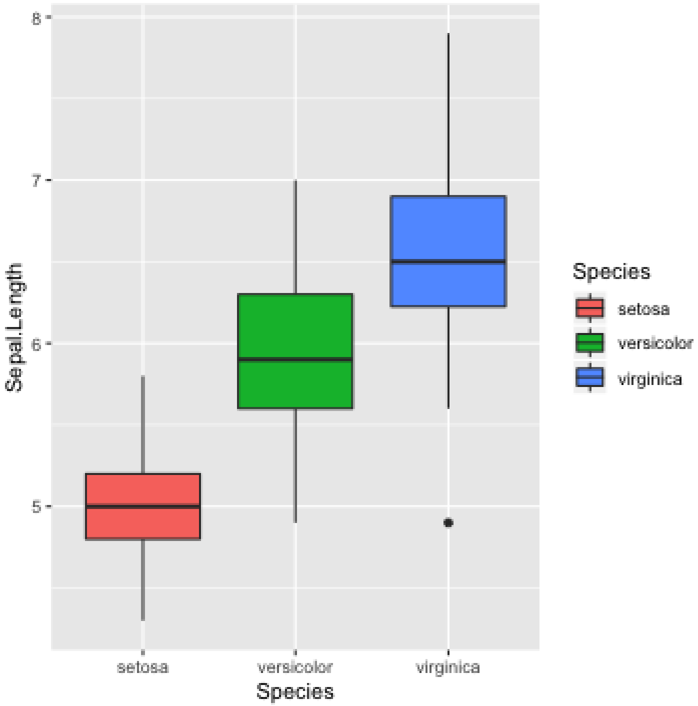
Boxplots with colors

```
iris %>%
  ggplot() +
  geom_boxplot(aes(x = "", y = Sepal.Length), fill = "green")
```



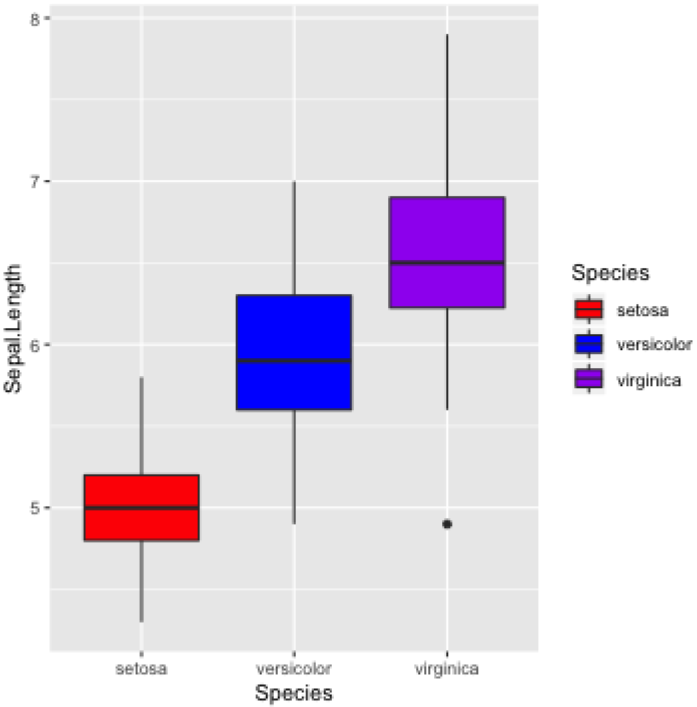
Boxplots with colors by species

```r
iris %>%
  ggplot() +
  geom_boxplot(aes(x = Species, y = Sepal.Length, fill = Species))
```



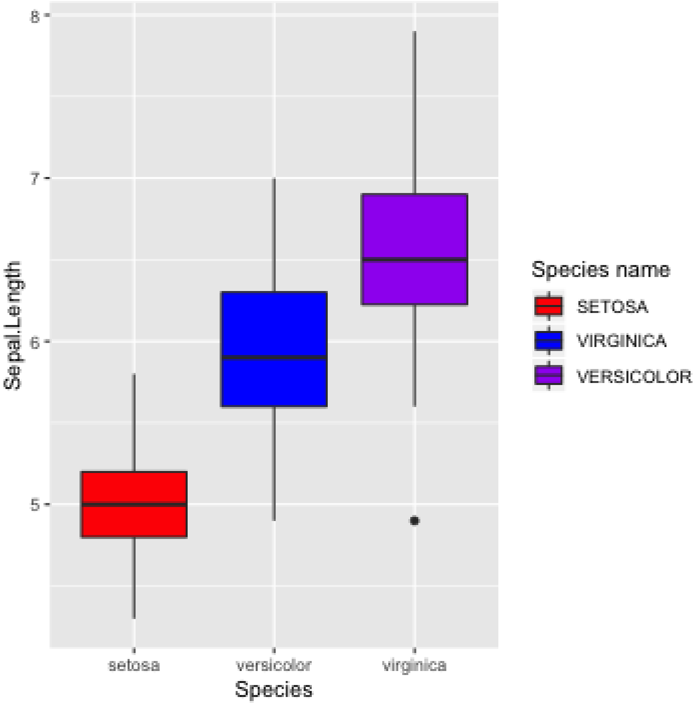
Boxplots, user-specified colors by species

```
iris %>%
  ggplot() +
  geom_boxplot(aes(x = Species, y = Sepal.Length, fill = Species)) +
  scale_fill_manual(values=c("red", "blue", "purple"))
```



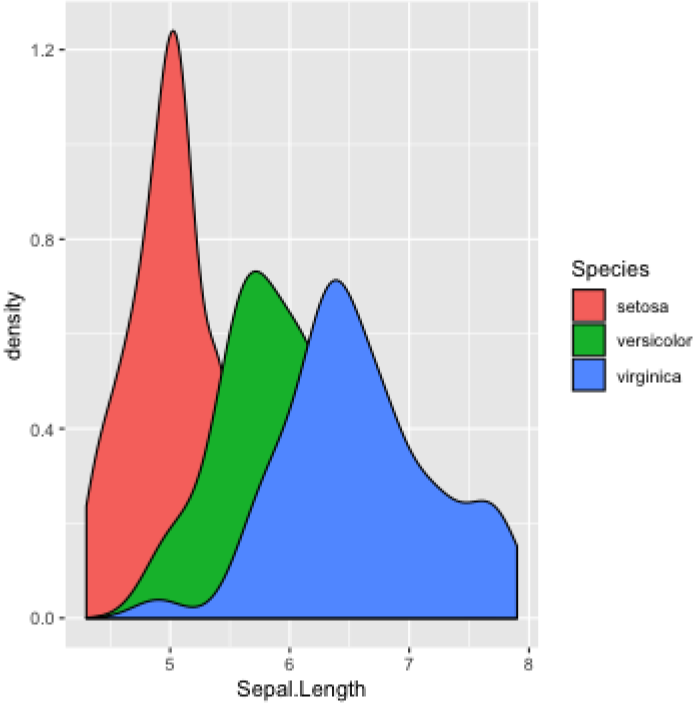
Boxplots, change legend settings

```
iris %>%
  ggplot() +
  geom_boxplot(aes(x = Species, y = Sepal.Length, fill = Species)) +
  scale_fill_manual(
    values=c("red", "blue", "purple"),
    name = "Species name",
    labels=c("SETOSA", "VIRGINICA", "VERSICOLOR"))
```



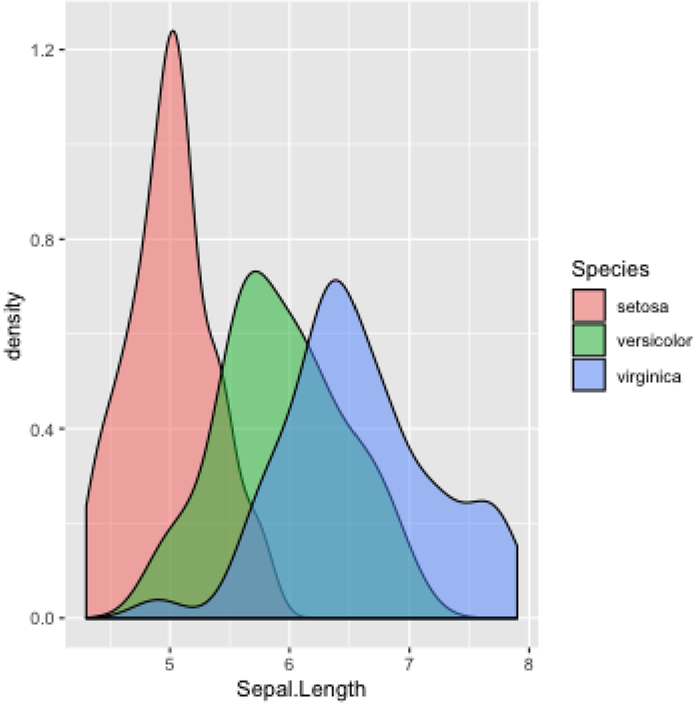
Density plots

```
iris %>%
  ggplot() +
  geom_density(aes(x = Sepal.Length, fill = Species))
```



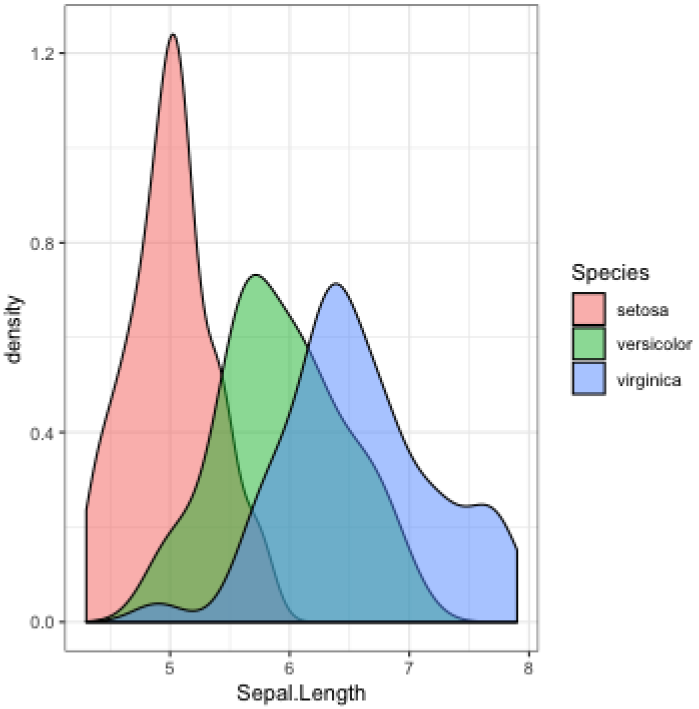
Density plots, control transparency

```
iris %>%
  ggplot() +
  geom_density(aes(x = Sepal.Length, fill = Species), alpha = 0.5)
```



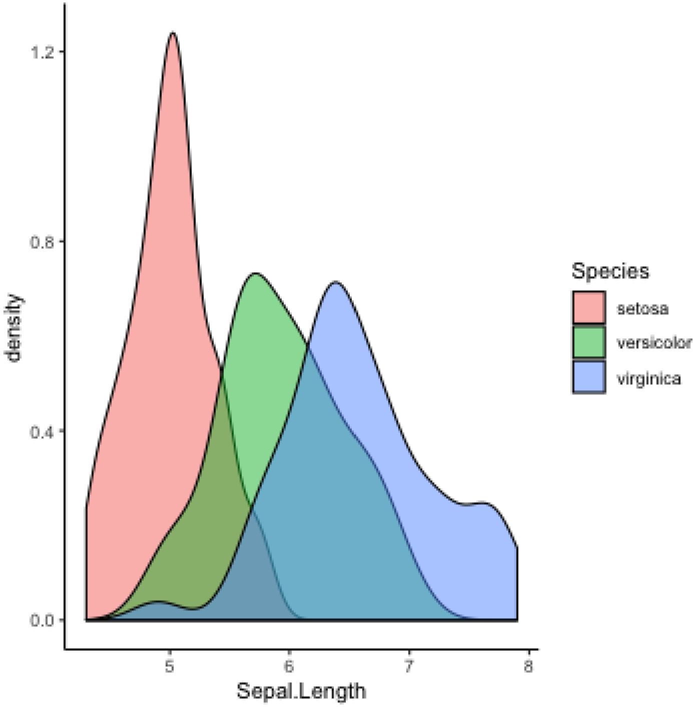
Change default background theme 1/3

```
iris %>%
  ggplot() +
  geom_density(aes(x = Sepal.Length, fill = Species), alpha = 0.5) +
  theme_bw()
```



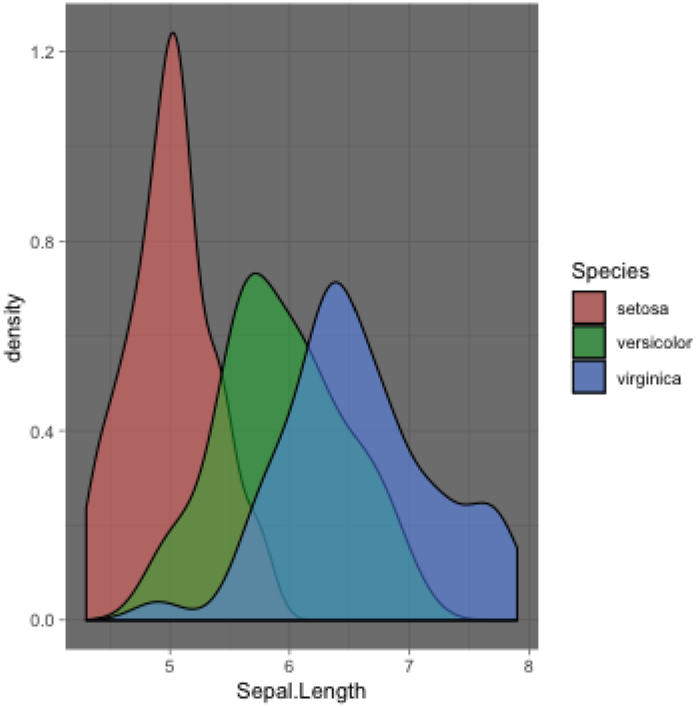
Change default background theme 2/3

```r
iris %>%
  ggplot() +
  geom_density(aes(x = Sepal.Length, fill = Species), alpha = 0.5) +
  theme_classic()
```



Change default background theme 3/3

```
iris %>%
  ggplot() +
  geom_density(aes(x = Sepal.Length, fill = Species), alpha = 0.5) +
  theme_dark()
```



tidyverse
www.rstudio.com

To dive even deeper in the tidyverse

- Learn the tidyverse: books, workshops and online courses

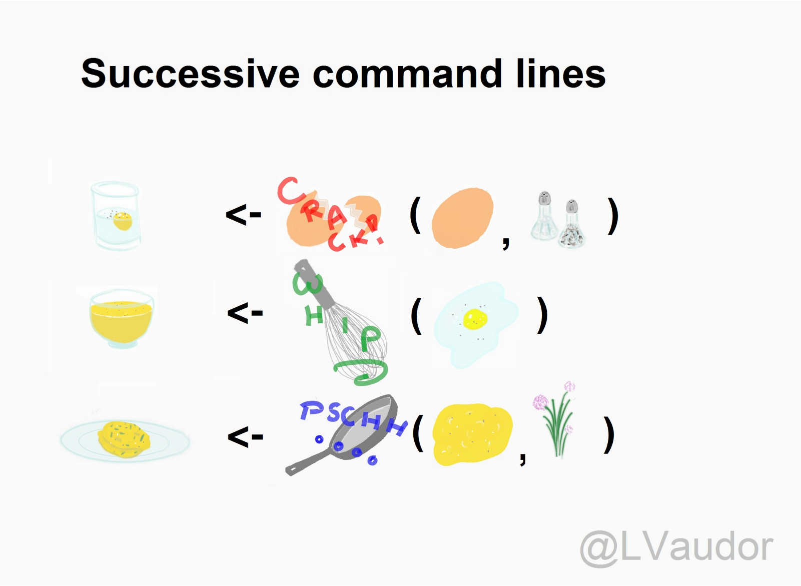
- My selection of books:
  - R for Data Science et Advanced R
  - Introduction à R et au tidyverse
  - Fundamentals of Data visualization
  - Data Visualization: A practical introduction

- Tidy Tuesdays videos by D. Robinson chief data scientist at DataCamp

Blog Lise Vaudor

Base R

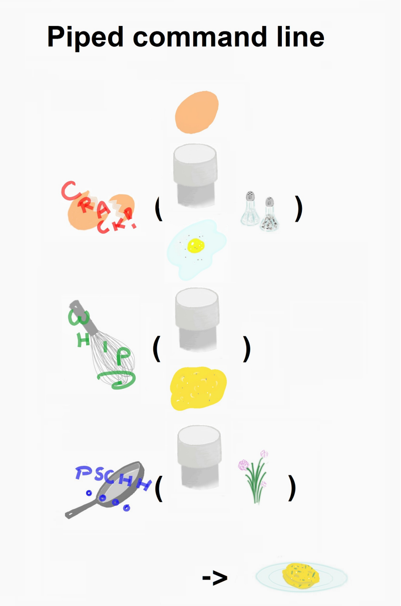
```
white_and_yolk <- crack(egg, add_seasoning)
omelette_batter <- beat(white_and_yolk)
omelette_with_chives <- cook(omelette_batter,add_chives)
```



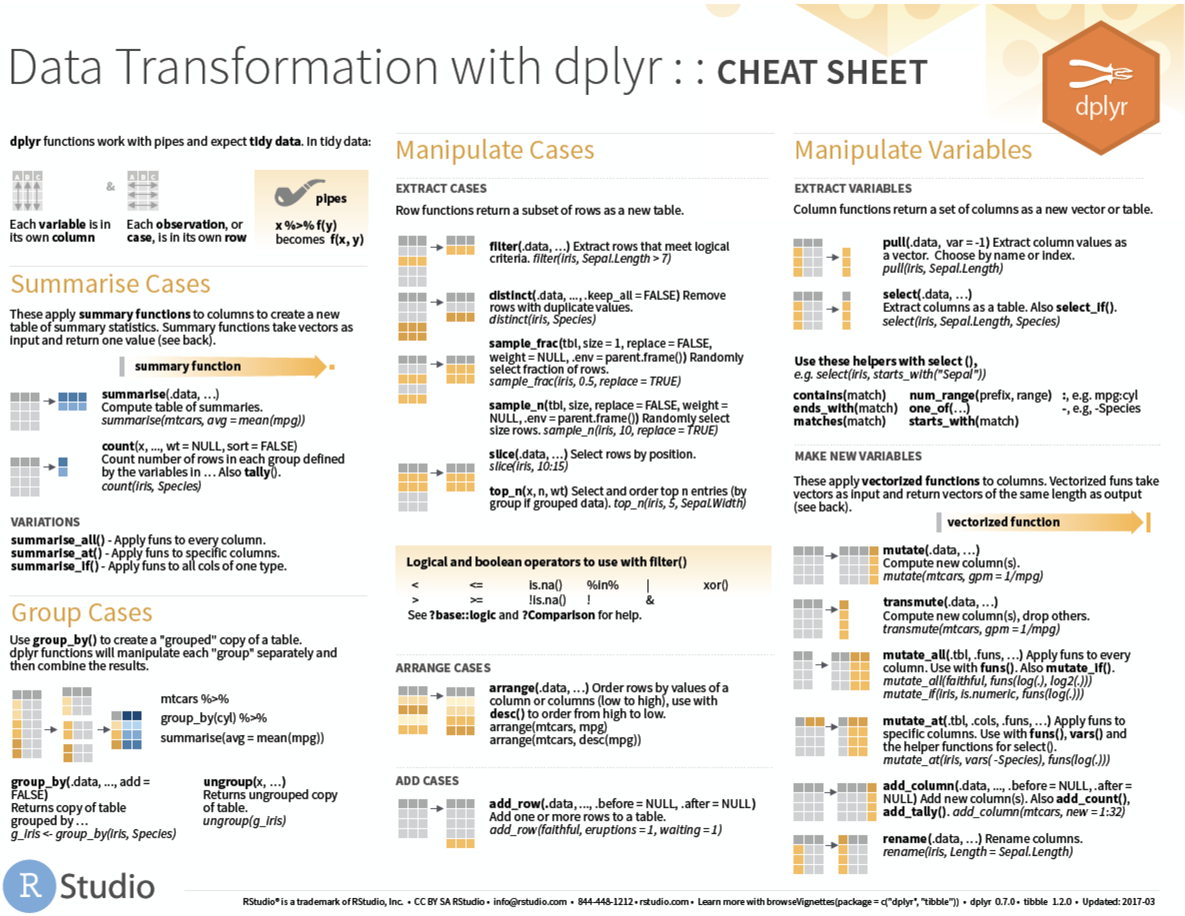
Blog Lise Vaudor

Piping

```
egg %>%
  crack(add_seasoning) %>%
  beat() %>%
  cook(add_chives) -> omelette_with_chives
```



The RStudio Cheat Sheets



Data Transformation with dplyr : : CHEAT SHEET

dplyr

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes

x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

group_by(.data, ..., add = FALSE)
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …)
Returns ungrouped copy of table.
ungroup(g_iris)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL)
Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match)	**num_range**(prefix, range)	:, e.g. mpg:cyl
ends_with(match)	**one_of**(…)	-, e.g, -Species
matches(match)	**starts_with**(match)	

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, …)
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …)
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with **funs()**. Also **mutate_if()**.
mutate_all(faithful, funs(log(.), log2(.)))
mutate_if(iris, is.numeric, funs(log(.)))

mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().
mutate_at(iris, vars(-Species), funs(log(.)))

add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also **add_count()**, **add_tally()**. *add_column(mtcars, new = 1:32)*

rename(.data, …) Rename columns.
rename(iris, Length = Sepal.Length)

RStudio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browseVignettes(package = c("dplyr", "tibble")) • dplyr 0.7.0 • tibble 1.2.0 • Updated: 2017-03

Thanks!

I created these slides with xaringan and RMarkdown using the rutgers css that I slightly modified.

Credit: I used material from Cécile Sauder, Stephanie J. Spielman and Julien Barnier.

olivier.gimenez@cefe.cnrs.fr
https://oliviergimenez.github.io/
@oaggimenez
@oliviergimenez