What R can do for you

Florian Privé

Grenoble RUG - September 13, 2018

Slides: `bit.ly/RUGgre11`

Contents

- Statistics & Data Science
- Visualization
- High Performance Computing
- Web
- Reporting
- RStudio IDE
- Community
- Learn R
- Program for this year

Statistics & Data Science

Statistics

> R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.
>
> -- https://www.r-project.org/about.html

Work with many kinds of data

- tabular tidy data (see this book)
- spatial (see this book and this blog)
- temporal (see this book)
- textual (see this book)
- networks (see this book)
- etc
- etc
- etc

CRAN task views

Browse https://cran.r-project.org/web/views/.

> CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic.

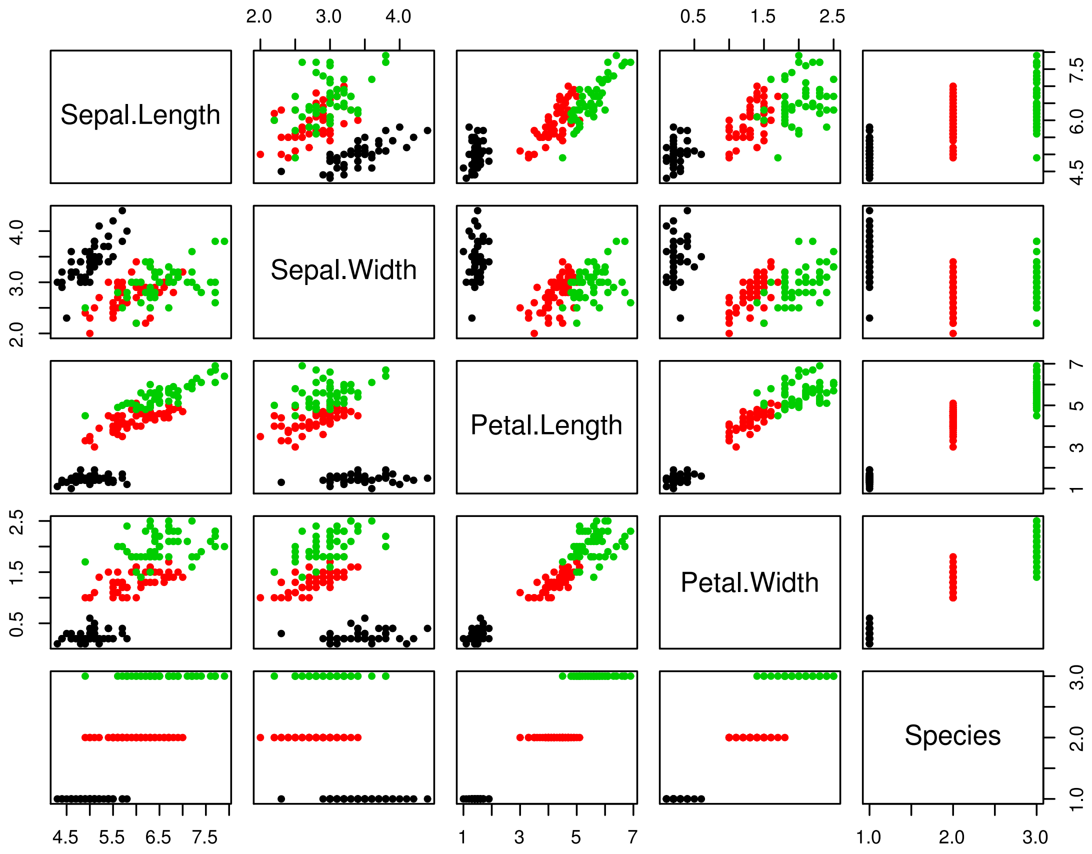
They are so useful to discover packages that are used in a field of research.

Bioconductor

Search engine: https://www.bioconductor.org/packages/devel/BiocViews.html

Simple example

```
plot(iris, pch = 20, col = iris$Species)
```



Simple example

```
pca <- prcomp(iris[, -5], center = TRUE, scale. = TRUE)
plot(pca$x, pch = 20, col = iris$Species)
```

Simple example (November session)

```
summary(fit <- lm(Petal.Length ~ ., data = iris))
```

```

Call:
lm(formula = Petal.Length ~ ., data = iris)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.78396 -0.15708  0.00193  0.14730  0.65418 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)       -1.11099    0.26987  -4.117 6.45e-05 ***
Sepal.Length       0.60801    0.05024  12.101  < 2e-16 ***
Sepal.Width       -0.18052    0.08036  -2.246   0.0262 *  
Petal.Width        0.60222    0.12144   4.959 1.97e-06 ***
Speciesversicolor  1.46337    0.17345   8.437 3.14e-14 ***
Speciesvirginica   1.97422    0.24480   8.065 2.60e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2627 on 144 degrees of freedom
Multiple R-squared:  0.9786,	Adjusted R-squared:  0.9778 
F-statistic:  1317 on 5 and 144 DF,  p-value: < 2.2e-16
```

Data manipulation with {dplyr} (May session)

```r
library(dplyr)
(flights <- nycflights13::flights)
```

```
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1  2013     1     1      517            515         2      830
 2  2013     1     1      533            529         4      850
 3  2013     1     1      542            540         2      923
 4  2013     1     1      544            545        -1     1004
 5  2013     1     1      554            600        -6      812
 6  2013     1     1      554            558        -4      740
 7  2013     1     1      555            600        -5      913
 8  2013     1     1      557            600        -3      709
 9  2013     1     1      557            600        -3      838
10  2013     1     1      558            600        -2      753
# ... with 336,766 more rows, and 12 more variables:
#   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>
```

Data manipulation with {dplyr}

R package {dplyr} aims to provide a function for each basic verb of data manipulation:

- `filter()`
- `arrange()`
- `select()`
- `mutate()`
- `group_by()`
- `summarise()`
- and many others..

Filtering observations

```r
filter(flights, month == 1, day == 1)
```

```
# A tibble: 842 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1  2013     1     1      517            515         2      830
 2  2013     1     1      533            529         4      850
 3  2013     1     1      542            540         2      923
 4  2013     1     1      544            545        -1     1004
 5  2013     1     1      554            600        -6      812
 6  2013     1     1      554            558        -4      740
 7  2013     1     1      555            600        -5      913
 8  2013     1     1      557            600        -3      709
 9  2013     1     1      557            600        -3      838
10  2013     1     1      558            600        -2      753
# ... with 832 more rows, and 12 more variables:
#   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>
```

Sorting

```
arrange(flights, desc(dep_delay))
```

```
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1  2013     1     9      641            900      1301     1242
 2  2013     6    15     1432           1935      1137     1607
 3  2013     1    10     1121           1635      1126     1239
 4  2013     9    20     1139           1845      1014     1457
 5  2013     7    22      845           1600      1005     1044
 6  2013     4    10     1100           1900       960     1342
 7  2013     3    17     2321            810       911      135
 8  2013     6    27      959           1900       899     1236
 9  2013     7    22     2257            759       898      121
10  2013    12     5      756           1700       896     1058
# ... with 336,766 more rows, and 12 more variables:
#   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>
```

Adding/replacing variables

```
mutate(flights, speed = distance / air_time * 60)
```

```
# A tibble: 336,776 x 20
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1  2013     1     1      517            515         2      830
 2  2013     1     1      533            529         4      850
 3  2013     1     1      542            540         2      923
 4  2013     1     1      544            545        -1     1004
 5  2013     1     1      554            600        -6      812
 6  2013     1     1      554            558        -4      740
 7  2013     1     1      555            600        -5      913
 8  2013     1     1      557            600        -3      709
 9  2013     1     1      557            600        -3      838
10  2013     1     1      558            600        -2      753
# ... with 336,766 more rows, and 13 more variables:
#   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, speed <dbl>
```

Piping operations

```
flights2 <- flights %>%
  filter(month == 1, day == 1) %>%
  arrange(desc(dep_delay)) %>%
  mutate(speed = distance / air_time * 60)
print(flights2, n = 6)
```

```
# A tibble: 842 x 20
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>
1  2013     1     1      848           1835       853     1001
2  2013     1     1     2343           1724       379      314
3  2013     1     1     1815           1325       290     2120
4  2013     1     1     2205           1720       285       46
5  2013     1     1     1842           1422       260     1958
6  2013     1     1     2115           1700       255     2330
# ... with 836 more rows, and 13 more variables:
#   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, speed <dbl>
```

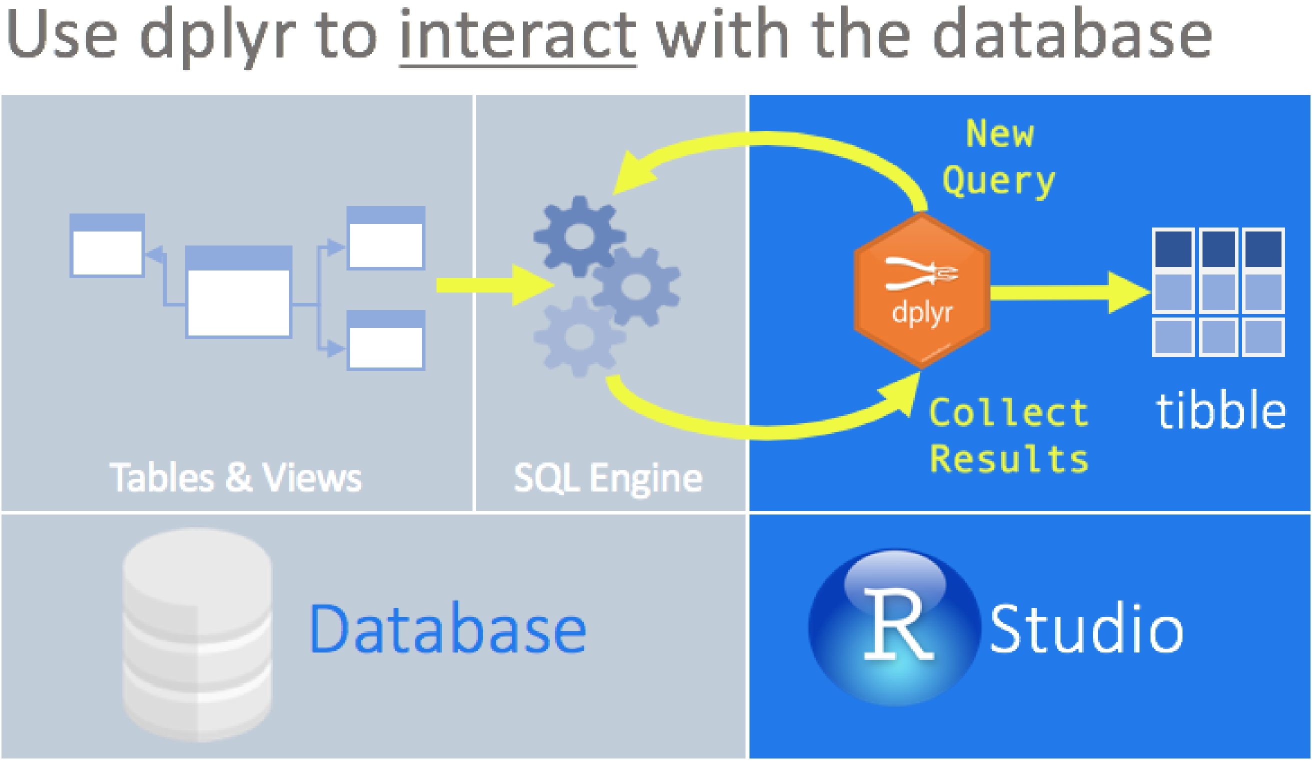
Summarizing by group

```
flights %>%
  group_by(carrier) %>%
  summarize(avg_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(avg_arr_delay)) %>%
  left_join(nycflights13::airlines)
```

```
Joining, by = "carrier"
```

```
# A tibble: 16 x 3
   carrier avg_arr_delay name
   <chr>           <dbl> <chr>
 1 F9              21.9  Frontier Airlines Inc.
 2 FL              20.1  AirTran Airways Corporation
 3 EV              15.8  ExpressJet Airlines Inc.
 4 YV              15.6  Mesa Airlines Inc.
 5 OO              11.9  SkyWest Airlines Inc.
 6 MQ              10.8  Envoy Air
 7 WN               9.65 Southwest Airlines Co.
 8 B6               9.46 JetBlue Airways
 9 9E               7.38 Endeavor Air Inc.
10 UA               3.56 United Air Lines Inc.
11 US               2.13 US Airways Inc.
12 VX               1.76 Virgin America
```

{dplyr} also works with databases



Learn more with this webinar.

Machine Learning & Deep Learning

Package {caret} (February session)

The caret package (short for **C**lassification **A**nd **RE**gression **T**raining) is a set of functions that attempt to streamline the process for creating predictive models (see the full documentation). The package contains tools for:

- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

Keras & TensorFlow in R (January session)

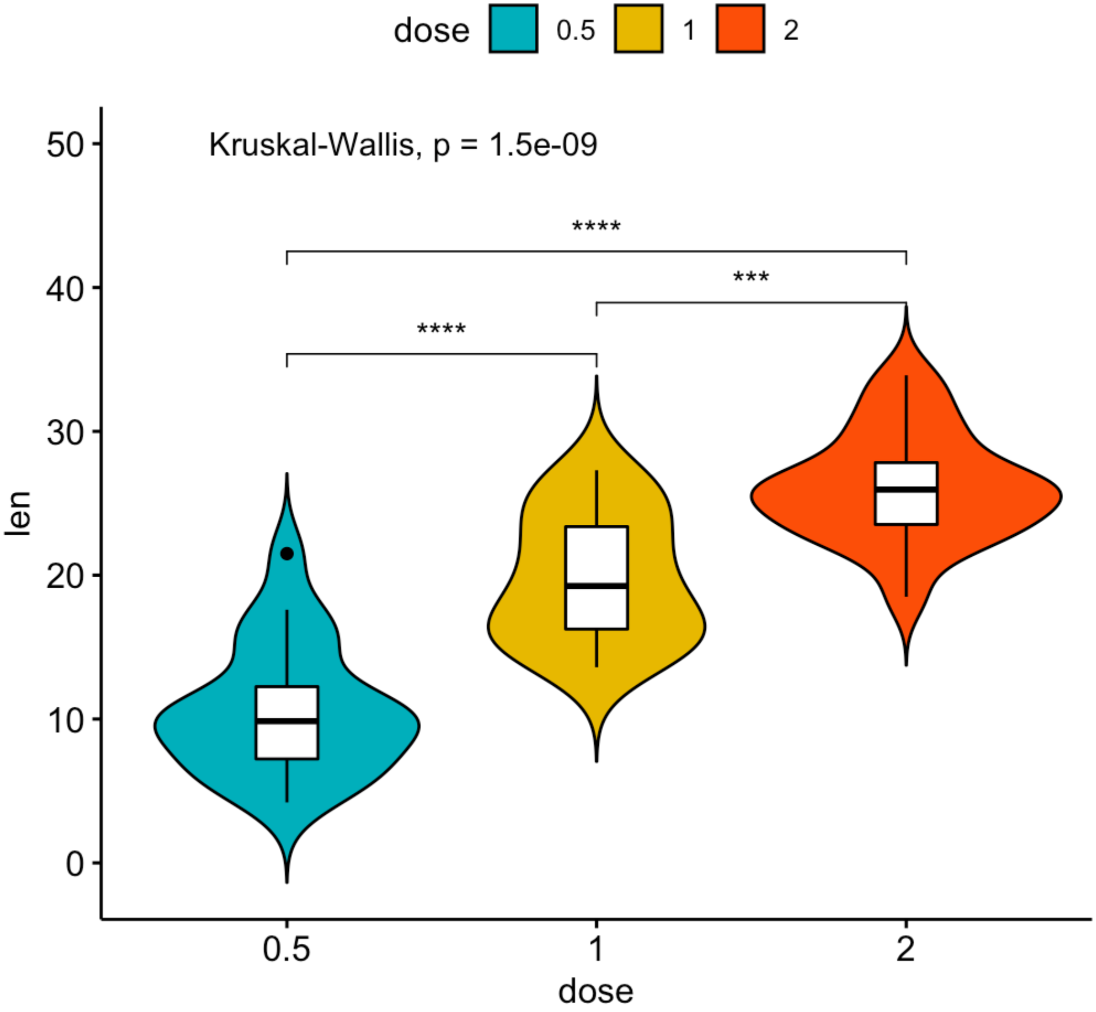
Keras & TensorFlow are integrated in R

- TensorFlow for R

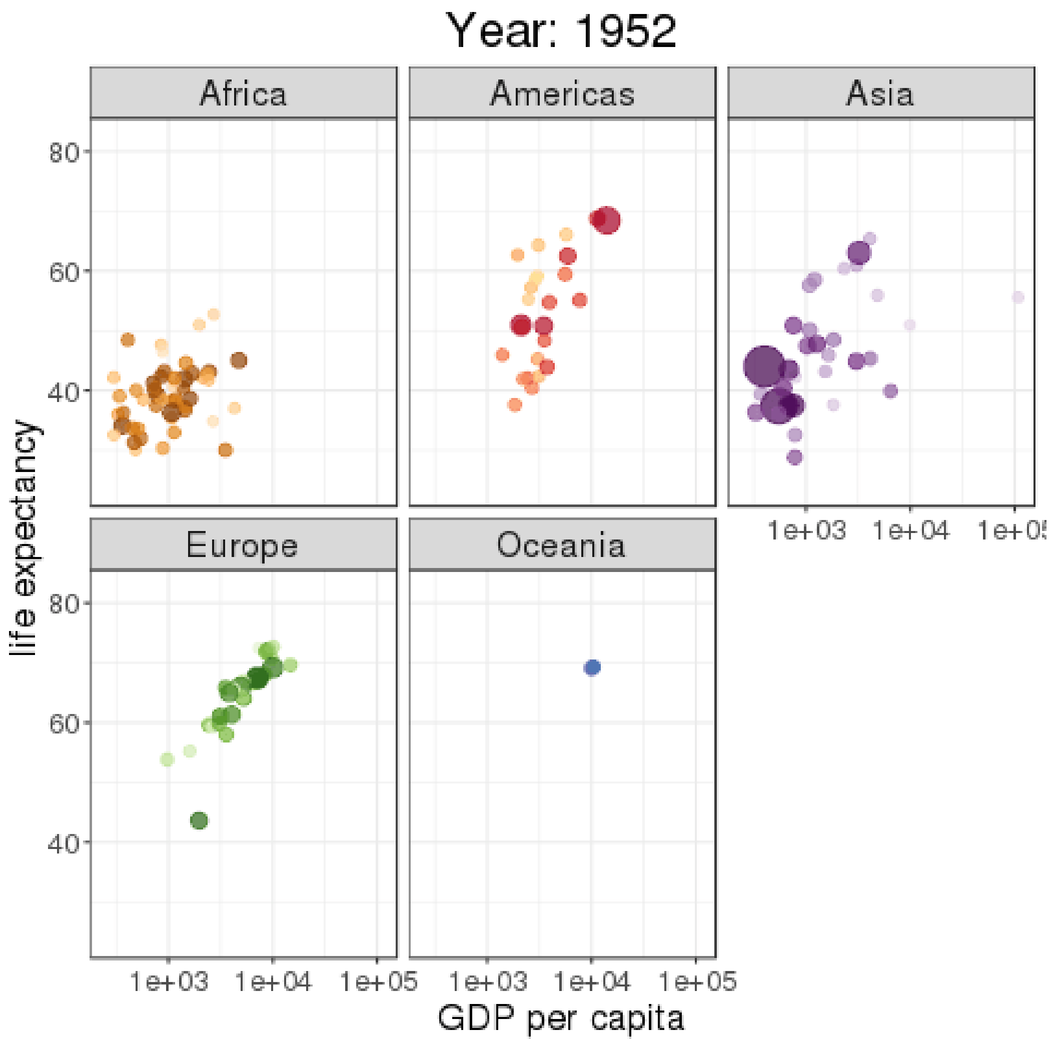
- TensorFlow for R blog

Visualization

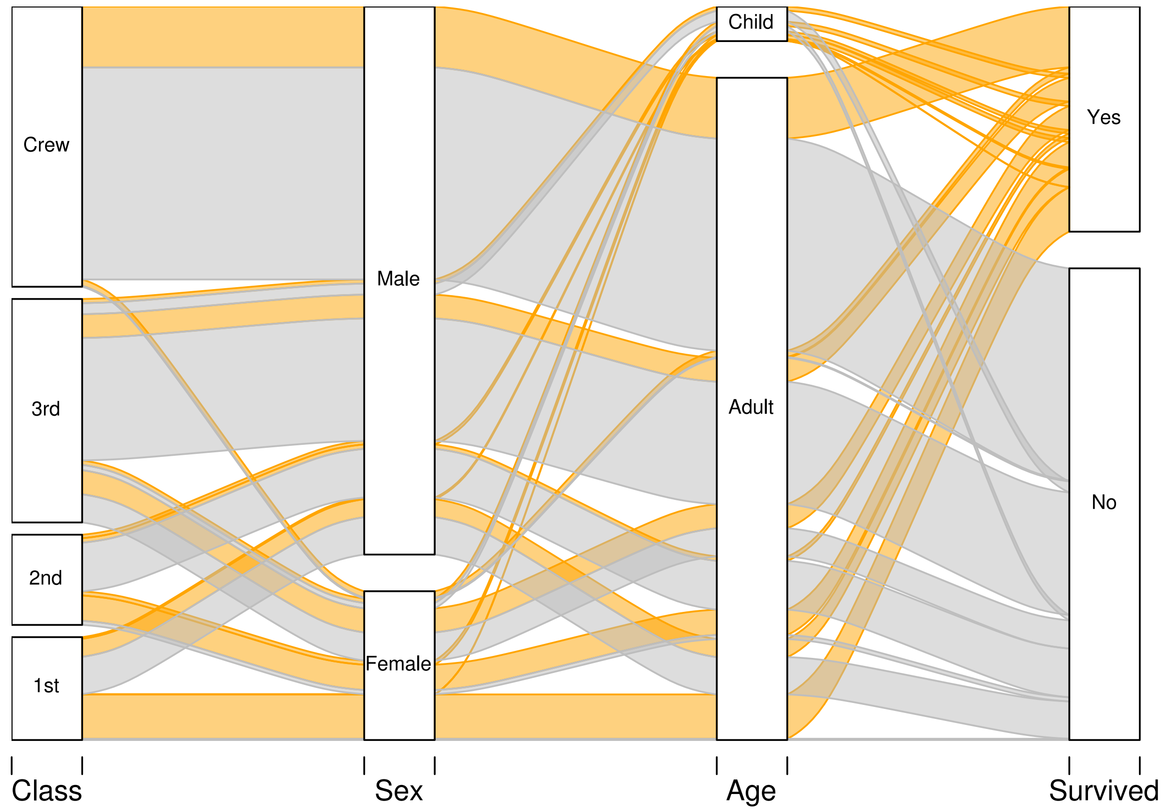
Package {ggplot2} and extensions (June session)



Animate graphics with {gganimate}



Fancy graphics: alluvial diagrams



More nice plots in the R Graph Gallery.

Image processing

- {magick}
- {imager} (October session)

Reporting

R Markdown (April session)

- Reports (analysis, etc) with text, code and results in the same place! With many possible output formats including HTML, PDF, MS Word, beamer, etc.
- HTML presentations (like this one! -- see source code)
- websites (such as the website of our R user group)
- books (or even a thesis)

Web

Web scrapping

```r
library(rvest)

read_html("https://r-in-grenoble.github.io/sessions.html") %>%
  html_nodes(".schedule") %>%
  html_nodes(".center-title") %>%
  html_text() %>%
  gsub("\n", "", .) %>%
  writeLines()
```

```
What R can do for you
Image processing with package {imager}
Linear models in R
Manage your workflow with package {drake}
Deep Learning with package {tensorflow}
Machine Learning with package {caret}
Best coding practices
R Markdown
Data manipulation with package {dplyr}
Data vizualisation with package {ggplot2}
```

Shiny apps: web apps in R

- Example 1: Airbnb visualization in New York
- Example 2: Make pixel art models

Learn more

High Performance Computing

Integrate C++ code with {Rcpp}

Rcpp lives between R and C++, so that you can get

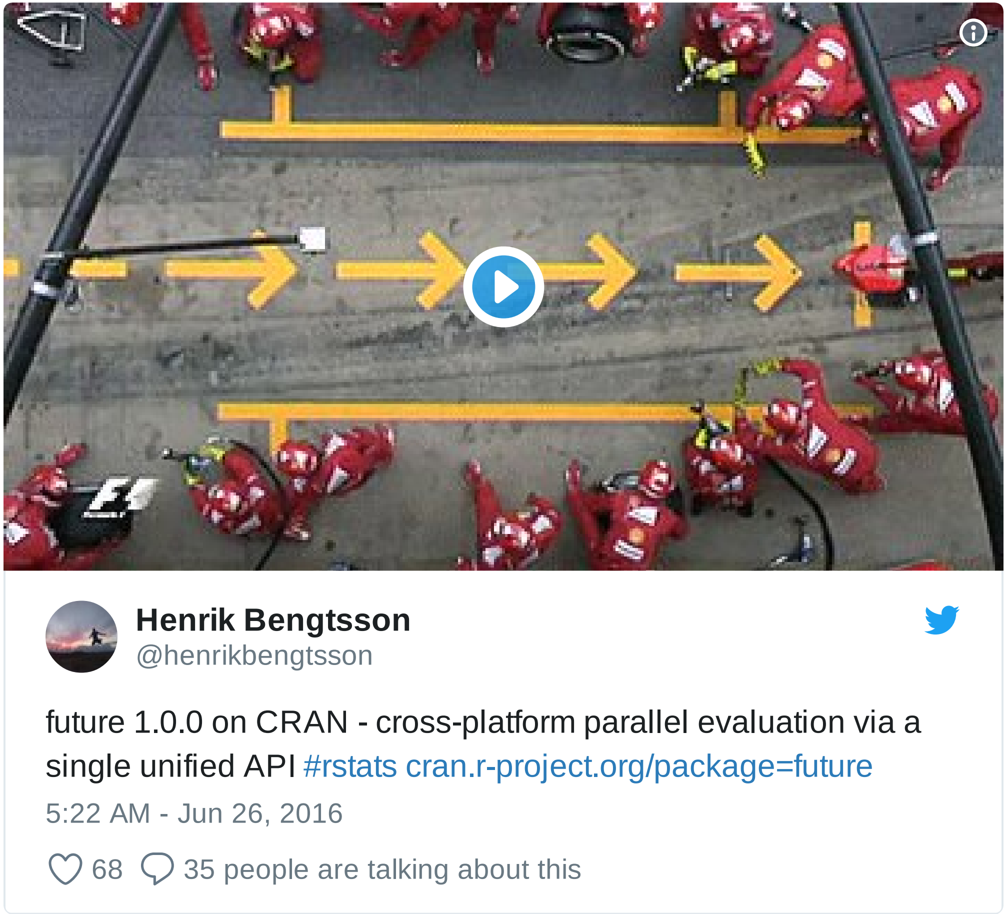
- the *performance of C*++ and
- the *convenience of R*.

As

- I love *performance* and
- I also enjoy *simplicity*,

Rcpp might be my favorite R package.

Easy parallelism with {future}



Henrik Bengtsson
@henrikbengtsson

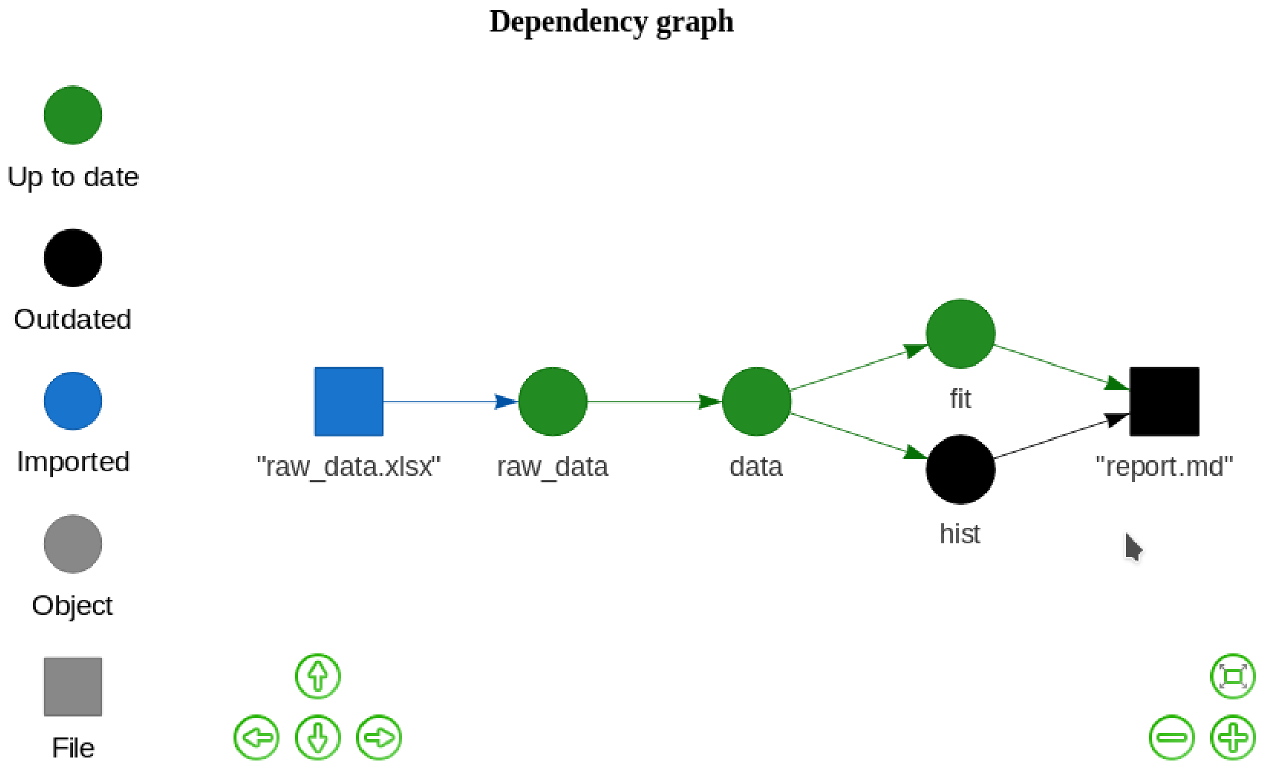
future 1.0.0 on CRAN - cross-platform parallel evaluation via a single unified API #rstats cran.r-project.org/package=future

5:22 AM - Jun 26, 2016

♡ 68 💬 35 people are talking about this

Also see my intro to parallelism with {foreach}.

Scalable reproducible workflow with {drake} (December session)



Large matrices with {bigstatsr}

Advantages of using FBM objects

- you can apply algorithms on **data larger than your RAM**,
- you can easily **parallelize** your algorithms because the data on disk is shared,
- you write **more efficient algorithms** (you do less copies and think more about what you're doing),
- you can use **different types of data**, for example, in my field, I’m storing my data with only 1 byte per element (rather than 8 bytes for a standard R matrix). See the documentation of the FBM class for details.

RStudio

RStudio IDE really helps

- console / scripts / environment / plots
- code diagnostics
- projects (+ git panel)
- viewer / debugger / profiler
- interactive import / connection
- integrated terminal / HTML viewer
- support many programming languages

Where to learn R?

Where to learn R?

- An Introduction to R by the R core team

- Introduction to R by DataCamp

- R for Data Science by Garrett Grolemund & Hadley Wickham, and some solutions

- Advanced R by Hadley Wickham, and some solutions

- Useful packages for Data Science

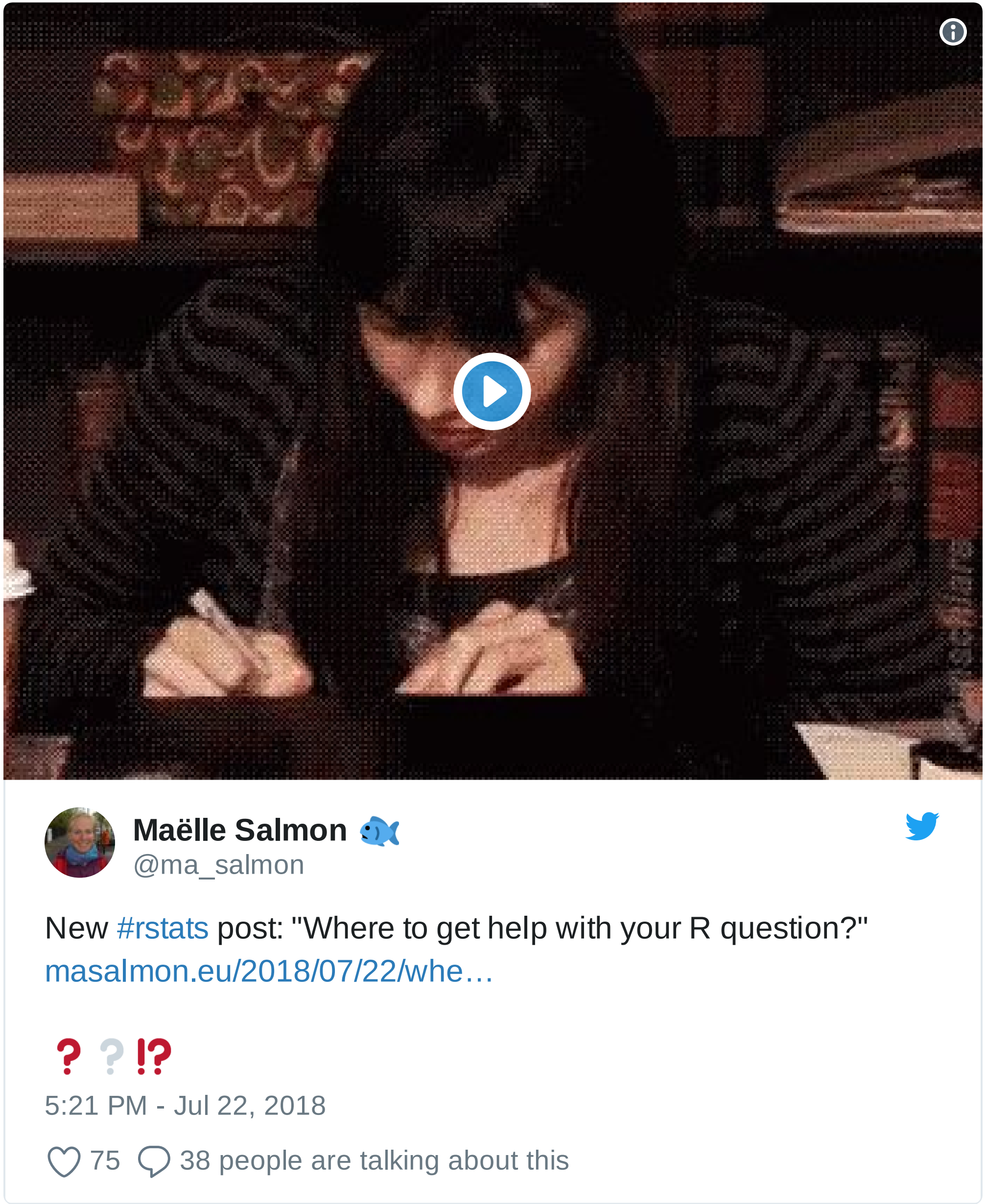
- CRAN Task Views

- Course: Advanced R course for PhD students in Grenoble (and 5 other open spots). **In French, but may be in English if enough demands.**

- Read code, documentation, blog posts, etc. And PRACTICE.

- Learn from others

    - join the French-speaking R community
    - join the R-Ladies community



Maëlle Salmon 🐟
@ma_salmon

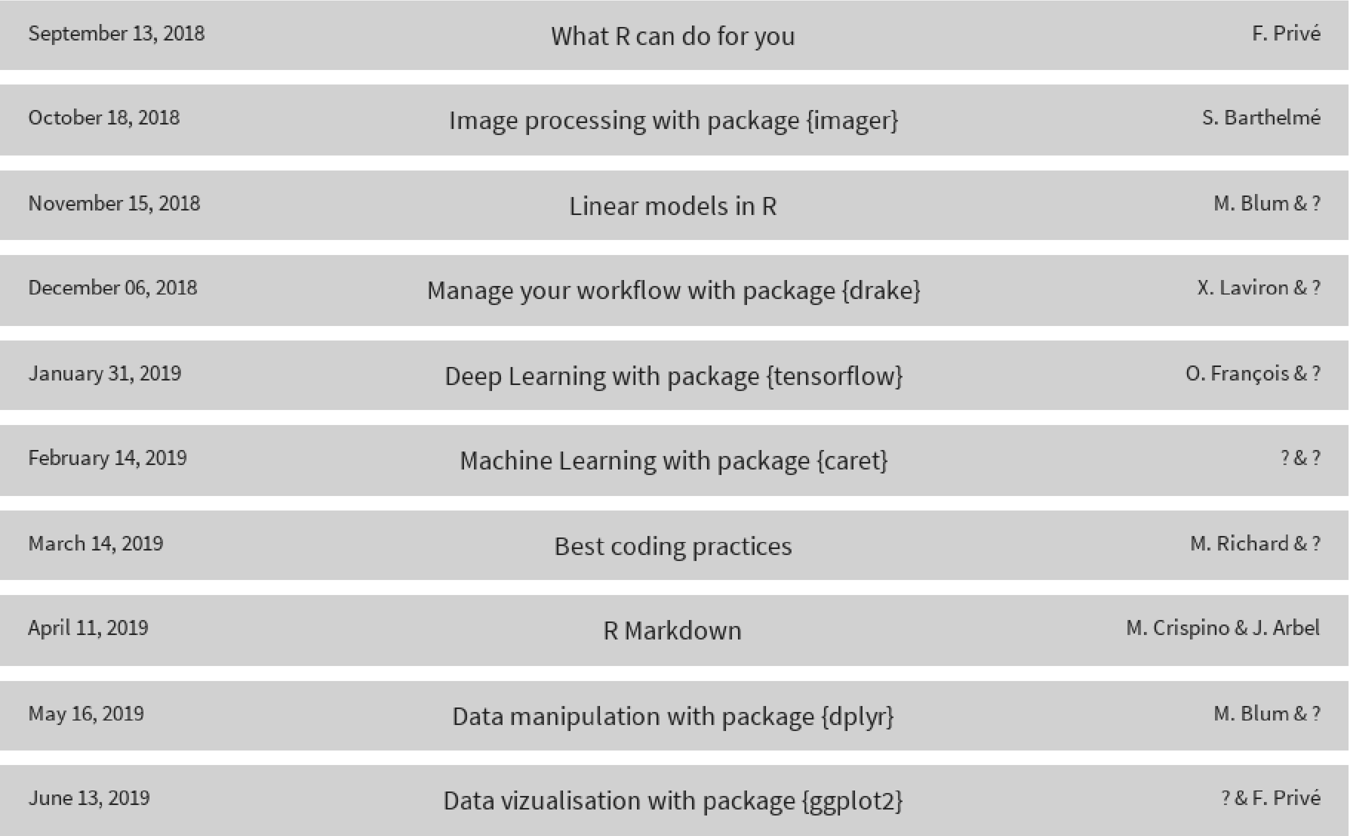
New #rstats post: "Where to get help with your R question?" masalmon.eu/2018/07/22/whe…

❓❔⁉️

5:21 PM - Jul 22, 2018

♡ 75 💬 38 people are talking about this

Schedule



September 13, 2018	What R can do for you	F. Privé
October 18, 2018	Image processing with package {imager}	S. Barthelmé
November 15, 2018	Linear models in R	M. Blum & ?
December 06, 2018	Manage your workflow with package {drake}	X. Laviron & ?
January 31, 2019	Deep Learning with package {tensorflow}	O. François & ?
February 14, 2019	Machine Learning with package {caret}	? & ?
March 14, 2019	Best coding practices	M. Richard & ?
April 11, 2019	R Markdown	M. Crispino & J. Arbel
May 16, 2019	Data manipulation with package {dplyr}	M. Blum & ?
June 13, 2019	Data vizualisation with package {ggplot2}	? & F. Privé

Thanks Grenoble Alpes Data Institute

Data Institute
Univ. Grenoble Alpes

for food, ecocups and stickers

Thanks!

Slides: `bit.ly/RUGgre11`

privefl privefl F. Privé

Slides created via the R package **xaringan**.