# MiniCourse R packages Day2

Charles Plessy

30/09/2021

## Day 2 contents

- Bioconductor
  - Core packages
  - Core classes
  - Release cycle
  - Submission process
- GitHub
  - Package hosting
  - Continuous integration
  - GitHub actions
- Practical: package website with pkgdown

## Bioconductor

Bioconductor (or just “Bioc“) is *“open source software for bioinformatics”*, based on more than 2000 R packages. It is old enough to have its own Wikipedia page!

It can be installed. . . from CRAN.

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.13")
```

Hold on if you are not bioinformatician, we will use it to learn more about *S4* classes.

## Bioconductor core packages

Canonical URLs: `https://bioconductor.org/packages/nameOfThePackage`

The core Bioconductor packages provide core classes that you will find being used in almost every other packages.

```
# Attaches a lot of stuff !!!
BiocManager::install("GenomicRanges")
library("GenomicRanges")
```

```r
# Use this in your R Markdown files...
suppressPackageStartupMessages(library("GenomicRanges"))

# Or since R4.1
library("GenomicRanges") |> suppressPackageStartupMessages()
```

## Output of library("GenomicRanges")

```
# Note the difference between "loading" and "attaching"
> library("GenomicRanges")

Restarting R session...

> library("GenomicRanges")
Loading required package: stats4
Loading required package: BiocGenerics
Loading required package: parallel

Attaching package: ‘BiocGenerics’

The following objects are masked from ‘package:parallel’:

    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport, clu
    parApply, parCapply, parLapply, parLapplyLB, parRapply, parSapply, parSappl

The following objects are masked from ‘package:stats’:

    IQR, mad, sd, var, xtabs

The following objects are masked from ‘package:base’:

    anyDuplicated, append, as.data.frame, basename, cbind, colnames, dirname, d
    duplicated, eval, evalq, Filter, Find, get, grep, grepl, intersect, is.unso
    lapply, Map, mapply, match, mget, order, paste, pmax, pmax.int, pmin, pmin.
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort, table, tapp
```

## Bioconductor core classes

- Use the S4 class system.
- Often names start with capital letters.
- Some of the core classes are in the core packages that have cryptic names.
- Today I will not cover annotation databases or genome objects.

## DataFrame

```r
DataFrame()
```

```
## DataFrame with 0 rows and 0 columns
```

```r
as(airquality, "DataFrame")
```

```
## DataFrame with 153 rows and 6 columns
##         Ozone   Solar.R      Wind      Temp     Month       Day
##     <integer> <integer> <numeric> <integer> <integer> <integer>
## 1          41       190       7.4        67         5         1
## 2          36       118       8.0        72         5         2
## 3          12       149      12.6        74         5         3
## 4          18       313      11.5        62         5         4
## 5          NA        NA      14.3        56         5         5
## ...       ...       ...       ...       ...       ...       ...
## 149        30       193       6.9        70         9        26
## 150        NA       145      13.2        77         9        27
## 151        14       191      14.3        75         9        28
## 152        18       131       8.0        76         9        29
## 153        20       223      11.5        68         9        30
```

```r
# or just `DataFrame(airquality)` ...
```

## Rle

Run-length encoding (`Rle`) is a simple and efficient way to compress data.

```r
Rle(rpois(1e7, 1e-3))
```

```
## integer-Rle of length 10000000 with 20021 runs
##   Lengths: 2986    1  242    1  386    1  900 ...    1  858    1  868    1
##   Values :    0    1    0    1    0    1    0 ...    1    0    1    0    1
```

A data structure that I like is a `DataFrame` of `Rle` values.

```r
airquality |> lapply(Rle) |> DataFrame() |> head(3)
```

```
## DataFrame with 3 rows and 6 columns
##   Ozone Solar.R  Wind  Temp Month   Day
##   <Rle>   <Rle> <Rle> <Rle> <Rle> <Rle>
## 1    41     190   7.4    67     5     1
## 2    36     118     8    72     5     2
## 3    12     149  12.6    74     5     3
```

## Other important classes

- `GRanges`
- `SummarizedExperiment`
- `MultiAssayExperiment`

## A few words on the S4 class system

- polymorphic and functional: core function names dispatch on different methods according to the class of their arguments.
- Objects have “slots” accessed with the `@` sign.
- copy-on-write / copy-on-modify semantics.
- Useful as data structure but also for type safety

Learn more with https://adv-r.hadley.nz/s4.html

## Let’s extend a S4 class

It is strongly recommended to re-use the core Bioc classes in packages designed for Bioconductor.

```
library(methods)
setClass("BetterList", contains = "SimpleList")
setMethod("show", "BetterList", function(object) {
  callNextMethod()
  cat("This superior version of the SimpleList class is brought to you by the O
})
SimpleList(a=1, b=2) |> as("BetterList")
```

```
## BetterList of length 2
## names(2): a b
## This superior version of the SimpleList class is brought to you by the OIST
```

## It also works on S3 classes

It is a matter of taste whether to do so or not.

```r
setClass("BetterList2", contains = "list")
setMethod("show", "BetterList2", function(object) {
  callNextMethod()
  cat("This superior version of the list class is also brought to you by the OI
})
list(c=3) |> as("BetterList2")
```

```
## An object of class "BetterList2"
## [[1]]
## [1] 3
##
## This superior version of the list class is also brought to you by the OIST m
```

## Use for type safety

```r
setClass("ListOfChars", contains = "SimpleList", validity = function (object){
  # Actually not so safe, what is the list contains sublists?
  all(sapply(object, is.character))
})
setMethod("show", "ListOfChars", function(object) {
  callNextMethod()
  cat("This safer version of the SimpleList class is surely brought to you by t
})
SimpleList("haha", "hoho") |> as("ListOfChars") |> validObject()
```

```
## [1] TRUE
```

```r
# Try this!
# SimpleList("haha", 1) |> as("ListOfChars") |> validObject()
```

## When to Depend or Import Bioc packages

- If you use their functions internally, Import the packages in the DESCRIPTION file and import their functions in NAMESPACE. Typical examples: `IRanges`, `S4Vectors`.
- If you want their functions to be easily available to their users, just Depend on the packages in the DESCRIPTION file. Typycal examples: `GenomicRanges`, `SummarizedExperiment`, `ggplot2`.

## Tip for easier debugging of your package.

Do the ground work in a S3 function, and wrap it in the S4 system.

```
setGeneric("countRows", function(x) standardGeneric("countRows") )
```

```
## [1] "countRows"
```

```
.countRows <- function(x) {
  if (nrow(x) > 0) cat("Owow, there are rows \n")
  cat("I found ", nrow(x), "rows.\n")
  cat("I think I finished counting\n")
}
setMethod("countRows", "DataFrame", .countRows)
airquality |> DataFrame() |> countRows()
```

```
## Owow, there are rows
## I found  153 rows.
## I think I finished counting
```

And now you can set the debugger to jump straight in your code instead of navigating through multiple layers of S4 dispatch.

```
debugonce(rowCounteR:::.countRows)
airquality |> DataFrame() |> countRows()
```

## More on Bioconductor

- Releases twice a year, shortly after R releases.
- Maintains a `release` and a `devel` branch in parallel. Version numbers are even and odd respectively.
  http://www.bioconductor.org/developers/how-to/version-numbering/
- Support site: https://support.bioconductor.org/
- Browse all soure code at https://code.bioconductor.org/
- Submission process takes place on GitHub by oppening an issue on `Bioconductor/Contributions`

## GitHub

- Not the only hosting platform, there is also GitLab, etc.
- Useful to manage source code of course, but also
- There are functions to install a package directly from a GitHub repository.

```
remotes::install_github("user/repo"), remotes::install_bioc()
```

- Reminder: never use passowrdless SSH keys, use `ssh-add` instead (you might need to run `eval $(ssh-agent)` first). Maye your keys transiently available on Deigo with `ssh -A`.

## Get a Zenodo DOI for packages via Git repositories

CRAN or Bioconductor, and journals provide DOIs. How about non-peer-reviewed packages ?

You can get DOIs from Zenodo, or alternatively Dryad, figshare, etc.

Zenodo has a good integration with GitHub. Random example. Documentation

## GitHub actions

GitHub actions for regression testing: example with the `charles-plessy/CAGEr` package.

```
usethis::use_github_action()
biocthis::use_bioc_github_action()
```

## Pkgdown

GitHub actions are also used for building GitHub pages.

The website of `pkgdown` is of course built with `pkgdown`, so let's have a look.

https://pkgdown.r-lib.org/

https://pkgdown.r-lib.org/articles/linking.html

Example with the oist/GenomicBreaks package

Now, let's try it together !