

(A bit of) Advanced R

Part 1 - towards better R-base programming

Julien Chiquet

<http://github/jchiquet/CourseAdvancedR>

Université Paris Dauphine, Juin 2018



Outline

- 1 Function, Functionals

References

- R Core Team (2017): A Language and Environment for Statistical Computing
<https://www.R-project.org/>
- Wickham (2014): Advanced R, retrieved from <http://adv-r.had.co.nz/>
- Gillespie & Lovelace (2016): efficient R programming
<https://bookdown.org/csgillespie/efficientR/>

Prerequisites

Data Structure in base R

- ① Atomic vector (integer, double, logical, character)
- ② Recursive vector (list)
- ③ Factors
- ④ Matrices and array
- ⑤ Data Frame

→ Creation, Basic Operation, Manipulation, Representation

Resources

- Advanced R, chapters I.2, I.3 (Wickham, 2014, <http://adv-r.had.co.nz/>)
- An introduction to R programming
http://julien.cremeriefamily.info/teachings_L3BI_ISV51.html

Outline

① Function, Functionals

The [a-z]*ply family I

Example with factors (tapply)

```
data <- rnorm(100)
sexe <- factor( sample(c("H","F"), 100, replace = TRUE))
mean.1 <- tapply(data, sexe, mean) ## good
mean.2 <- c() ## complicated
for (l in levels(sexe))
  mean.2 <- c(mean.2, mean(data[sexe == l]))
```

Example with list or data.frame (sapply/lapply)

The [a-z]*pply family II

```
data(oats, package = "MASS")
oats[1:2, ]
```

```
##      B      V      N      Y
## 1 I Victory 0.0cwt 111
## 2 I Victory 0.2cwt 130
```

```
sapply(oats, is.factor) ## readable
```

```
##      B      V      N      Y
## TRUE TRUE TRUE FALSE
```

```
for (c in 1:ncol(oats)) ## less readable (I think)
  print(is.factor(oats[,c]))
```

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

The do.call function |

constructs and executes a function call from a name or a function and a list of arguments to be passed to it

Suppose you have the outputs of 100 simulations at your disposal, stored in a list like that

```
res <- replicate(100, rbind(data.frame(method="lasso", mse=runif(1,.75,1) , timing=runif(1)),
                             data.frame(method="ridge", mse=runif(1,.5,.75), timing=runif(1)),
                             data.frame(method="bayes", mse=runif(1,.85,1) , timing=runif(1,100),
                             ), simplify=FALSE)
```

```
class(res)
```

```
## [1] "list"
```

```
res[[1]]
```

```
##      method      mse      timing
## 1  lasso 0.9468897  0.463805
## 2  ridge 0.5309500  0.309606
## 3  bayes 0.9477400 141.573739
```


The `do.call` function II

```
length(res)
```

```
## [1] 100
```

How would you store them in a single data frame?

```
all.res <- do.call(rbind, res)  
dim(all.res)
```

```
## [1] 300 3
```

The Reduce function

'Reduce' uses a binary function to successively combine the elements of a given vector

↪ can be use to post-process your list of simulations obtained via `mclapply` just like `do.call`

Say more... (map, Reduce)

A Reduce example: “jackknifing” a lasso solution path

A single Lasso fit of the diabete data set

```
library(glmnet)
library(lars) # the diabetes data set (part of the lars package)
data(diabetes)
y <- diabetes$y
x <- diabetes$x
n <- length(y)
lasso <- glmnet(x,y)
plot(lasso)
```

twidhtwidth.

A Reduce example: “jackknifing” a lasso solution path II I

Compute the regularization paths for all subsets, removing one individual at once

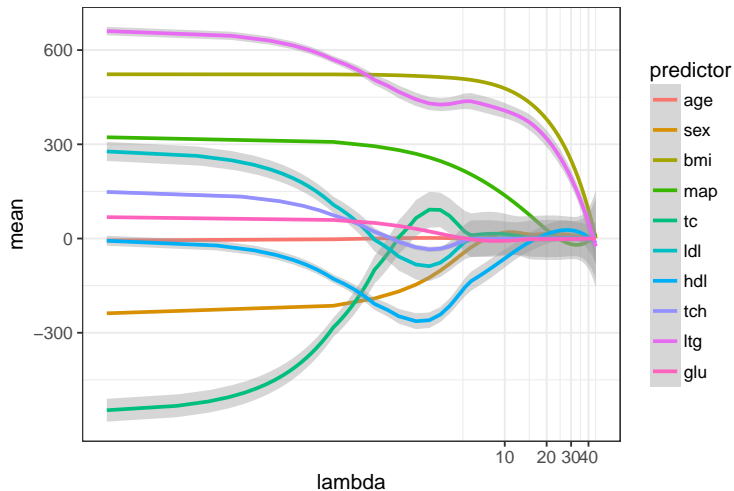
```
paths <- parallel::mclapply(1:n, function(i) {  
  glmnet(x[-i, ], y[-i], lambda = lasso$lambda)$beta  
}, mc.cores = 4)
```

Computing the envelop around the average regularization path with Reduce

```
mean.path <- Reduce("+", paths)/n  
sdev.path <- sqrt(Reduce("+", lapply(paths, function(path) path**2))/n -  
  mean.path**2)
```

```
library(ggplot2)  
library(reshape2)  
mean.path <- t(as.matrix(mean.path)); rownames(mean.path) <- lasso$lambda  
sdev.path <- t(as.matrix(sdev.path)); rownames(sdev.path) <- lasso$lambda  
dplot <- cbind(reshape2::melt(mean.path), reshape2::melt(sdev.path)[, 3])  
colnames(dplot) <- c("lambda", "predictor", "mean", "sdev")  
ggobj <- ggplot(dplot, aes(x=lambda, y=mean, group=predictor, color=predictor)) + geom_smooth(a  
print(ggobj + coord_trans(x="log10"))
```

A Reduce example: “jackknifing” a lasso solution path II II



References

- Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., ... Chang, W. (2018). *Rmarkdown: Dynamic documents for r*. Retrieved from <https://CRAN.R-project.org/package=rmarkdown>
- Burns, P. (2012). *The r inferno*. Lulu. com. Retrieved from <http://www.burns-stat.com/documents/books/the-r-inferno/>
- Chang, W. (2012). *R graphics cookbook: Practical recipes for visualizing data*. "O'Reilly Media, Inc." Retrieved from <http://www.cookbook-r.com/Graphs/>
- Eddelbuettel, D. (2013). *Seamless r and c++ integration with rcpp*. Springer. Retrieved from <http://dirk.eddelbuettel.com>
- Gandrud, C. (2016). *Reproducible research with r and r studio*. Chapman; Hall/CRC. Retrieved from <https://github.com/christophergandrud/Rep-Res-Book>
- Gillespie, C., & Lovelace, R. (2016). *Efficient r programming*. "O'Reilly Media, Inc." Retrieved from <https://bookdown.org/csgillespie/efficientR/>
- R Core Team. (2017). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from <https://www.R-project.org/>
- Wickham, H. (2014). *Advanced r*. CRC Press. Retrieved from