(A bit of) Advanced R

Towards better programming with base R

Julien Chiquet

Université Paris Dauphine

Juin 2018

http://github/jchiquet/CourseAdvancedR

Resources

- Wickham (2014): Advanced R, retrieved from http://adv-r.had.co.nz/

Part 0: Prerequesties

- Struture (atomic vector, List, Data Frame)

Outline

The [a-z]*pply family I

Some texte here... (check slides from L3)

Example with factors (tapply)

```
data <- rnorm(100)
sexe <- factor(sample(c("H","F"),100,rep=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)
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 I

> *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 disposable, 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]]
```

The do.call function II

```
##   method       mse      timing
## 1  lasso 0.9078272   0.5146003
## 2  ridge 0.7162294   0.3136347
## 3  bayes 0.9604167 116.9485047
```

```
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*

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

Say more... (map, Reduce)

A Reduce example: "jacknifing" 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)
```

twidthtwidth.

A Reduce example: "jacknifing" a lasso solution path II I

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

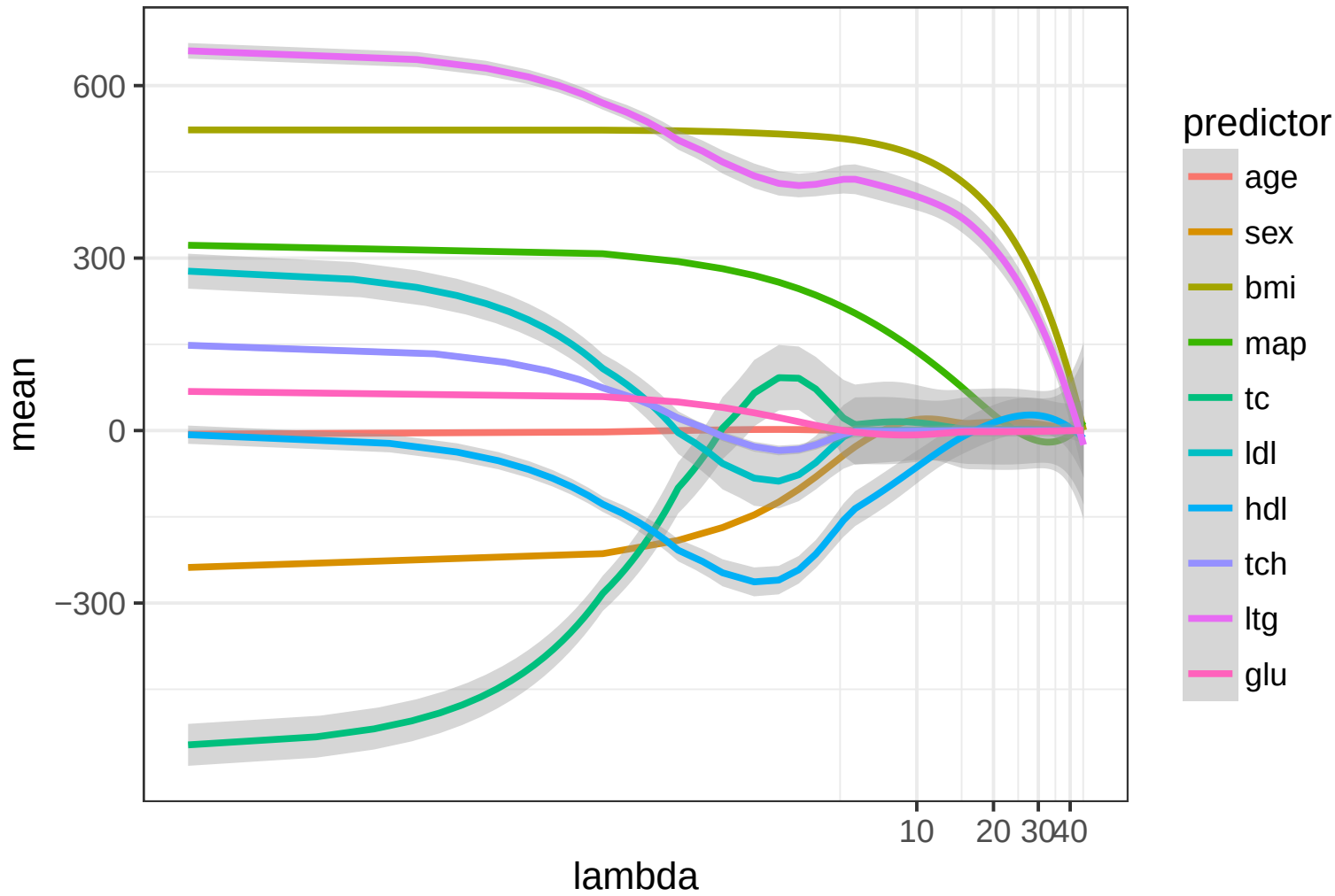
```
paths <- parallel::mclapply(1:n, function(i) {
    glmnet(x[-i, ], y[-i], lambda = lasso$lambda)$beta
}, mc.cores = 4)
```

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: "jacknifing" a lasso solution path II II



References I

Wickham, H. (2014). *Advanced r*. CRC Press. Retrieved from http://adv-r.had.co.nz/