The `apply()` family

Daniela Castro-Camilo

General use

- **apply(x, index, function):** Apply a function to the rows (index=1) or columns (index=2) of a matrix.

```r
set.seed(4032021)
A = matrix(sample(1:20, 15), 5, 3); A
```

```
##      [,1] [,2] [,3]
## [1,]   12    6   20
## [2,]   11    4   18
## [3,]    8   13    7
## [4,]    9    3   10
## [5,]    1    5   19
```

```r
f = function(x) (max(x) + min(x))/2
apply(A, 2, f)
```

```
## [1]  6.5  8.0 13.5
```

```r
apply(A, 1, f)
```

```
## [1] 13.0 11.0 10.0  6.5 10.0
```

- **lapply(x, function):** Apply a function to each element of the list `x`.

```r
x = list(c(1,2,3), c(4,5,6)); x
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 4 5 6
```

```r
lapply(x, f)
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 5
```

```r
x = list("x1" = A, "x2" = A+2); x
```

```
## $x1
##      [,1] [,2] [,3]
## [1,]   12    6   20
## [2,]   11    4   18
## [3,]    8   13    7
## [4,]    9    3   10
## [5,]    1    5   19
##
## $x2
##      [,1] [,2] [,3]
## [1,]   14    8   22
## [2,]   13    6   20
## [3,]   10   15    9
## [4,]   11    5   12
## [5,]    3    7   21
```

```r
g = function(y) apply(y, 2, f)
lapply(x, g)
```

```
## $x1
## [1]  6.5  8.0 13.5
##
## $x2
## [1]  8.5 10.0 15.5
```

```r
lapply(x, sum) # output is a list
```

```
## $x1
## [1] 146
##
## $x2
## [1] 176
```

- **sappply(x, function):** Same as `lapply`. The difference is that `sapply` will try to simplify as much as possible the output of `lapply`.

```r
sapply(x, sum) # output is a vector (simplified list)
```

```
##  x1  x2
## 146 176
```

```r
set.seed(4032021)
temp = round(runif(10, 0, 20),1)
h = function(x,y) {mean(x) > y}
sapply(temp, h, y = 15) # output is logical
```

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

- **vappply(x, function):** Same as `sapply`. The difference is that `vapply` allows you to specify the type of output.

```r
vapply(temp, h, numeric(1), y = 15) # output is numeric (1 = TRUE, 0 = FALSE)
```

```
##  [1] 0 0 1 0 1 0 0 0 0 0
```

- **tapply(x, y, function):** Apply a function to subsets of a vector `x`. The subsets are defined by the vector `y`.

```r
x = 1:10
y = rep(c(T,F), 5)
tapply(x, y, sum)
```

```
## FALSE  TRUE
##    30    25
```

- **mapply(function, x, y, ...):** Apply a function on multiple objects by elements (the `m` stands for `multivariate`).

```r
Q1 <- matrix(c(rep(1, 4), rep(2, 4), rep(3, 4), rep(4, 4)),4,4); Q1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    2    3    4
## [3,]    1    2    3    4
## [4,]    1    2    3    4
```

```r
# Faster using mapply()
Q2 <- mapply(rep,1:4,4); Q2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    2    3    4
## [3,]    1    2    3    4
## [4,]    1    2    3    4
```

- Parallelised versions of `lapply` are available in, e.g., the package `parallel`. These are useful when paralelising a code that needs to run on several cores. If interested, look for **mclapply, mcmapply**.

Data example

I illustrate the use of some of the functions in the `apply()` family using the dataset `airquality` available in R.

```r
data(airquality)
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 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
## 6    28      NA 14.9   66     5   6
```

```r
?airquality
# Data contains information on mean ozone, solar radiation, average wind speed
# and maximum daily temperature
```

1. [Apply] Find the days with the highest maximum temperature and mean ozone.

```r
# `id` contains the row indices of `irquality`
# with the highest ozone and temperate
id = apply(airquality[, c("Ozone","Temp")], 2, which.max)
airquality$Day[id[1]] # day with highest ozone
```

```
## [1] 25
```

```r
airquality$Day[id[2]] # day with highest temperate
```

```
## [1] 28
```

2. Find the number of days where solar radiation exceeded 180

```
which(airquality$Solar.R > 180)
```

```
##  [1]   1   4   7  10  12  13  14  16  17  19  22  26  29  30  31  32  33
## [18]  34  35  36  37  39  40  41  42  43  45  46  47  48  55  62  63  64
## [35]  67  68  69  70  73  75  77  78  79  80  81  83  84  85  86  89  90
## [52]  91  92  99 100 101 102 104 105 111 112 113 115 116 117 118 120 121
## [69] 122 123 125 126 127 130 131 132 133 134 135 136 139 140 142 143 144
## [86] 149 151 153
```

3. [lapply/sapply] Repeat 2 for the sequence of thresholds `thres = seq(180, 200)`.

```
thres = seq(180, 200)

# Using for()
out1 = list()
for(i in 1:length(thres)){
  out1[[i]] = which(airquality$Solar.R > thres[i])
}

# Using lapply
f = function(th,x) {which(x > th)}
out2 = lapply(thres, f, x = airquality$Solar.R)
```

`lapply` returns a list, which in the case is convenient since the length of the output changes with the threshold:

```
length(out2[[1]])
```

```
## [1] 88
```

```
length(out2[[20]])
```

```
## [1] 75
```

`sapply` is useful as well, but the output is exactly the same as with `lapply`. This is because `sapply` can't simplify a list whose elements have different lengths.

```
# Using sapply
out3 = sapply(thres, f, x = airquality$Solar.R)
```

3. [lapply] `lapply` can be very useful for data processing. Say we have 100 datasets measuring the same information contained in `airquality`. These 100 datasets are contained in a list called `my.airquality`. For the sake of this example, I will create `my.airquality` by randomly sampling from `airquality` 100 times:

```
# Creating `my.airquality`
my.airquality = list()
for(i in 1:100){
  id = sample(1:nrow(airquality), nrow(airquality), replace = T)
  my.airquality[[i]] = airquality[id, ]
  rownames(my.airquality[[i]]) = c() # set rownames to be 1,2,...,nrow(airquality)
}
# check the first entry
head(my.airquality[[1]])
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    45      212  9.7    79      8  24
## 2    13       27 10.3    76      9  18
## 3    48      260  6.9    81      7  16
## 4    40      314 10.9    83      7   6
## 5    30      193  6.9    70      9  26
## 6     7       49 10.3    69      9  24
```

Assume now that you want to create a new data frame that contains the solar radiation of all the 100 datasets as columns. Your new data frame, which we will call `all.solar`, should have `nrow(airquality)` rows and 100 columns. The following line of code selects the column `Solar.R` from the first element in `my.airquality`.

```
tmp = my.airquality[[1]][ "Solar.R"]
```

If we want to select the column `Solar.R` from all the element in `my.airquality`, we can use `lapply`:

```
all.solar = lapply(my.airquality, function(x) x[, "Solar.R"])
class(all.solar)
```

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

Note that `all.solar` is a list, not a data frame. To convert `all.solar` to a data frame, we can do:

```
all.solar = data.frame(matrix(unlist(all.solar), ncol = length(all.solar), byrow = FALSE))
names(all.solar) = paste0('dat', 1:100)
```