Minitask1

Florencia Luque and Seyed Amirhossein Mosaddad

2024-10-10

Part 1 - Apply Family

The apply Family is a way to apply a function to different types of data like a row, vector, matrix, etc. This functions have been replace or you should replace it with others libraries that do the same with an optimal result. This function is an alternative to a loop when you know waht you need to do to the data. #explain why or something about the general ways of the apply family

The family is created by this functions; apply(), lapply() , sapply(), vapply(), mapply() and tapply() functions.

apply

This function can be use to an arrays in different dimensions like a matrix. You will need a X the array, Margin 1 o 2 if you want to apply the function by row (1) and over columns (2). You can also apply on both using Margin=c(1,2) and obviously the fun that will be the function that you would like to apply like the sum or the mean. Example:

```
mat = matrix(data = rnorm(20,20,4),nrow = 4,ncol = 5,byrow = TRUE)
print(mat)
```

```
##          [,1]     [,2]     [,3]      [,4]     [,5]
## [1,] 21.03825 22.73644 19.96812 21.144636 13.98009
## [2,] 21.59497 14.63263 23.90162  9.813275 14.77347
## [3,] 18.67105 22.54996 16.30895 28.584203 21.53923
## [4,] 17.62966 18.69997 18.25623 21.267816 15.68341
```

```
sum_by_column = apply(mat,MARGIN = 2,FUN = sum)

print(sum_by_column)
```

```
## [1] 78.93393 78.61899 78.43491 80.80993 65.97620
```

tapply

Is very similar to apply but you can apply the function by groups. You can use this function to a dataframe or a matrix and you will need the data, the index that will be the factor that you want to group by and the fun. You can have more arguments but is not essential to make it work. We can transform or data to aggregate a factor variable and create and example.

```r
data = data.frame(mat)
data$grupo = c("A","B","B","A")
colnames(data)[1] = "age"
colnames(data)[2] = "bmi"
data$grupo = as.factor(data$grupo)

tapply(data$age,data$grupo,mean)
```

```
##        A        B
## 19.33396 20.13301
```

lapply

This function is apply over a list or a vector and return a list of the same lenght of the original input. This is a way to apply a function to every element on a list. The function will need the list or vector and the function that you want to apply. This mean that if you give it a vector it will apply the function to each element and if you give it a list it will apply by elements of the list.

```r
lista = list(tr = c(1:6),mn = c(5:8))
lista
```

```
## $tr
## [1] 1 2 3 4 5 6
##
## $mn
## [1] 5 6 7 8
```

```r
lapply(lista, median)
```

```
## $tr
## [1] 3.5
##
## $mn
## [1] 6.5
```

```r
vec = sample(1:40,5)
vec
```

```
## [1]  8  1 30 40 11
```

```r
lapply(vec, FUN = function(x) 0.5*x)
```

```
## [[1]]
## [1] 4
##
## [[2]]
## [1] 0.5
##
## [[3]]
## [1] 15
```

```
##
## [[4]]
## [1] 20
##
## [[5]]
## [1] 5.5
```

sapply

This function can by apply by a vector or list and will return a vector or a list. This function is very similar to lapply but it try to simplify it. This function is a wrapper of the lapply it can return not only a list it can return a vector, matrix or an array instead of a list. Let's do the same exercise to see the difference between both functions.

```
lista = list(tr = c(1:6),mn = c(5:8))
lista
```

```
## $tr
## [1] 1 2 3 4 5 6
##
## $mn
## [1] 5 6 7 8
```

```
sapply(lista, median)
```

```
##  tr  mn
## 3.5 6.5
```

```
vec = sample(1:40,5)
vec
```

```
## [1] 33  6 38 14 30
```

```
sapply(vec, FUN = function(x) 0.5*x)
```

```
## [1] 16.5  3.0 19.0  7.0 15.0
```

This function give you the same type that you give it in the input.

mapply

This is the multivariate version of sapply. Applying a function to multiple arguments.

```
mapply(sum, 1:5, 6:10)
```

```
## [1]  7  9 11 13 15
```

vapply

This is a more strict version of sapply in where you need to specified the type of return that you would like.

```r
my_list <- list(a = 1:5, b = 6:10)
vapply(my_list, sum,numeric(1))
```

```
##  a  b
## 15 40
```

Part 2 - MLE

MLE for gamma distribution

```r
mle_gamma = function(n){


}
```