Funções Apply

Iven Valpassos

13/09/2021

Nós vimos na aula de estruturas de controle que existe uma forma mais eficiente de fazer iterações nos dados do que o “loop for”: nós utilizamos a função apply().

A aplicação destes métodos é tãoimportante, que vamos dedicar essa aula toda à família de funções apply().

As funções apply() fazem parte do pacote básico do R, as principais são:

1- apply();

2 - lapply();

3 - sapply();

4 - mapply().

Um bom ponto de partida para entender as funções apply é consultar a documentação do R. Cada função tem suas características, espera receber um tipo de dado ou uma estrutura de dados específica como entrada e retorna o resultado também em um formato específico.

help(apply)

## starting httpd help server ... done

Consultando a documentação vemos que a função apply espera receber uma matriz como dado de entrada e retorna o resultado como uma matriz também.

x<- matrix(rnorm(40),nrow = 5, ncol = 8)  
  
apply(x,2,mean)

## [1] 0.76108759 0.02268936 1.06322710 -0.52606052 0.35221757 -0.39047556  
## [7] 0.49674527 0.07664313

help(lapply)

A lapply() pode receber uma lista ou um dataframe como dado de entrada e retorna uma lista (daí o ‘l’ de lapply) com o mesmo comprimento que os dados de entrada.

lista <- list(x[,1],x[,2],x[,3])  
  
lapply(lista,sum)

## [[1]]  
## [1] 3.805438  
##   
## [[2]]  
## [1] 0.1134468  
##   
## [[3]]  
## [1] 5.316135

help(sapply)

A função sapply() simplifica o resultado da função lapply().

O ‘s’ é de simplificar. A função retorna o resultado em um formato de vetor oou de matriz.

lista <- list(x[,1],x[,2],x[,3])  
  
  
sapply(lista,sum)

## [1] 3.8054379 0.1134468 5.3161355

help(mapply)

mapply(rep,1:6,5)

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

O ‘m’ é de multivariada. A função mapply() permite a aplicação de uma função com múltiplos argumentos em vetores e listas.

Agora vamos aplicar um pouco do que vimos com um exemplo.

Uma das formas de simular a trajetória do preço de um ativo, a ação de uma empresa por exemplo, é considerar que o preço segue um movimento browniano. Não vamos entrar em detalhes sobre isso aqui, mas no nosso exemplo vamos ver como usar funções apply() ajudar a construir cenários e testar hipóteses de forma rápida e eficiente.

Primeiro, vamos criar uma função que gera as trajetórias dos preços.

BrownianoGeom<-function(start\_point,paths,count,mean,sigma)  
{  
   
 interval<-5/count  
   
 sample<-matrix(0,nrow=(count+1),ncol=paths)  
 for(i in 1:paths)  
 {  
 sample[1,i]<-start\_point  
 for(j in 2:(count+1))  
 {  
 sample[j,i]<-sample[j-1,i]\*exp(interval\*(mean-((sigma)^2)/2)+((interval)^.5)\*rnorm(1)\*sigma)  
 list\_sample<- lapply(seq\_len(ncol(sample)),function(i) sample[,i])  
 list\_sample<- lapply(list\_sample, as.data.frame)  
   
 }  
 }   
   
   
 return(list\_sample)  
}

library(zoo)

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

Agora vamos criar uma função que calcula medias móveis.

medias\_moveis<- function(x){  
   
 x['sma']<- rollapply(x[1],FUN = mean,width = 9,fill = NA,align = 'right')  
 x['lma']<- rollapply(x[1],FUN = mean,width = 18,fill = NA,align = 'right')  
 x<- as.data.frame(x)  
  
 return(x)  
}

Você deve lembrar que nós já criamos as medias móveis do preço da ação da Petrobras em uma aula anterior, certo?

Vamos criar médias em 10 trajetórias diferentes de preços, o que vamos fazer com isso vai ficar pro nosso estudo de caso, na próxima aula.

lista\_precos<- BrownianoGeom(start\_point = 20,paths = 10,count = 1000,mean=0.01,sigma = 0.35)  
  
lista\_medias<- lapply(lista\_precos, medias\_moveis)  
  
print(head(lista\_medias[1]))

## [[1]]  
## X[[i]] X[[i]] X[[i]]  
## 1 20.00000 NA NA  
## 2 20.35748 NA NA  
## 3 20.87977 NA NA  
## 4 21.30044 NA NA  
## 5 21.16880 NA NA  
## 6 21.55635 NA NA  
## 7 21.07266 NA NA  
## 8 21.65413 NA NA  
## 9 21.44334 21.04811 NA  
## 10 21.64732 21.23114 NA  
## 11 22.20996 21.43697 NA  
## 12 23.09765 21.68340 NA  
## 13 23.22059 21.89675 NA  
## 14 23.75091 22.18366 NA  
## 15 24.28625 22.48698 NA  
## 16 25.47235 22.97583 NA  
## 17 25.06356 23.35466 NA  
## 18 24.95572 23.74492 22.39651  
## 19 24.49098 24.06089 22.64601  
## 20 24.53819 24.31958 22.87828