Aprendizaje Extra

Tomás

20 de febrero de 2020

install.package(“dplyr”)

# Aprendizaje Extra

## Match() o %in%

match(data$vector, names(integer))

Funcion que me permite hacer un match entre los nombres de un int y sus valores, con las columnas de un dataframe. La funcion %in% realiza exactamente lo mismo que match (), solamente que me entrega TRUE or FALSE y no las posiciones.

data <- data.frame(Nombres = c('Tomas', 'Tomas', 'Tomas', 'Tomas', 'Martin', 'Martin', 'Martin', 'Martin'),  
 Dias = c("2020-02-10", "2020-02-10", "2020-02-11", "2020-02-11", "2020-02-10", '2020-02-10',  
 '2020-02-11', '2020-02-11'), Pasos = c(NA, 12, 13, 22, NA, 26, 42, 12))  
aux <- tapply(data$Pasos,data$Nombres, mean, na.rm = TRUE); aux ; cat("Los names son", names(aux))

## Martin Tomas   
## 26.66667 15.66667

## Los names son Martin Tomas

aux2 <- match(data$Nombres, names(aux)) ; cat("Las posiciones que hacen match son", aux2)

## Las posiciones que hacen match son 2 2 2 2 1 1 1 1

aux3 <- data$Nombres %in% names(aux)  
cat("Los valores para esas posiciones son", aux[aux2])

## Los valores para esas posiciones son 15.66667 15.66667 15.66667 15.66667 26.66667 26.66667 26.66667 26.66667

## Coalesce()

datavector1, data$vector2)

La funcion permite reemplazar los valores de un vector en otro, cuando encuentre en a valores NA.

library(dplyr)

data <- data.frame( a = c(1,NA,NA,5), b = c(1,3,6,4))  
aux <- coalesce(data$a,data$b); cat("El nuevo vector queda de la forma", aux)

## El nuevo vector queda de la forma 1 3 6 5

data$a <- aux ; head(data$a) #Sustituyo el vector

## [1] 1 3 6 5

## ymd()

La funcion permite facilmente pasar una fecha a formato date.

library(lubridate)

## Warning: package 'lubridate' was built under R version 3.6.2

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

date <- "2012-04-11"  
class(date)

## [1] "character"

date <- ymd(date)  
class(date)

## [1] "Date"

## table()

La funcion table permite contar cuantas veces se repite alguna observacion basado en una columna

ayuda <- data.frame("a" = c("Hola","Hola","Chao","Como"), 'b' = c(2,2,3,4))  
tabla <- table(ayuda$a); cat("Las palabras de la columna a se repiten:", names(tabla), tabla)

## Las palabras de la columna a se repiten: Chao Como Hola 1 1 2

## Grafico de barra con mas de una variable

data <- data.frame("Plataforma" = c("Switch", "PS4", "Xbox"), "Precio en USD" = c(200,100,50),  
 "Precio en CLP" = c(300,200,100), "Precio en ROCA" = c(20,10,5)); data

## Plataforma Precio.en.USD Precio.en.CLP Precio.en.ROCA  
## 1 Switch 200 300 20  
## 2 PS4 100 200 10  
## 3 Xbox 50 100 5

Para generar el grafico debo cambiar la forma en la que estan dispuesto los datos de manera vertical

library(tidyr)

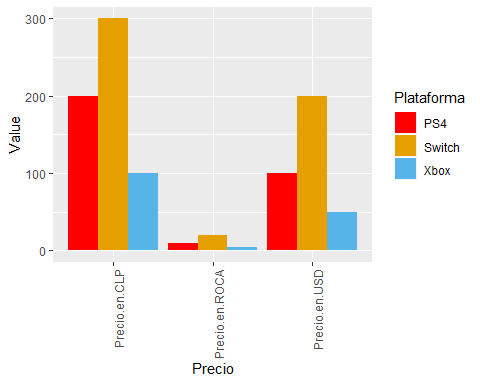
data2 <- data %>% gather(Precio, Value, -Plataforma); data2

## Plataforma Precio Value  
## 1 Switch Precio.en.USD 200  
## 2 PS4 Precio.en.USD 100  
## 3 Xbox Precio.en.USD 50  
## 4 Switch Precio.en.CLP 300  
## 5 PS4 Precio.en.CLP 200  
## 6 Xbox Precio.en.CLP 100  
## 7 Switch Precio.en.ROCA 20  
## 8 PS4 Precio.en.ROCA 10  
## 9 Xbox Precio.en.ROCA 5

Ahora puedo crear mi grafico de barras con mas variables

library(ggplot2)

ggplot(data2, aes(x = Precio, y = Value, fill = Plataforma)) + geom\_col(position = "dodge") +   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +  
 scale\_fill\_manual(values=c("red", "#E69F00", "#56B4E9"))



## dcast

Permite pasar de un formato de datos “largo”" a uno mas “ancho”. Necesita:

* Tabla
* Columnas que poner en fila (Separadas por “+”)
* Columnas disponer en columnas (Depsues de “~”)

library(reshape2)

data <- data.frame(Provincia = c("Huesca", "Teruel", "Zaragoza", "Huesca", "Teruel", "Zaragoza")  
 , Variables = c("Hombres","Hombres","Hombres","Mujeres","Mujeres","Mujeres")  
 , Valor = c(100,110,120,130,140,150))  
aux <- dcast(data, Provincia ~ Variables) # se puede usar value.var para hacer "override"

## Using Valor as value column: use value.var to override.

## Mutate

La funcion mutate me permite crear nuevas variables o editar las actuales en un dataframe

library(dplyr)

data <- data.frame(Value1 = c(1,2,3,4), Value2 = c(5,6,7,8)); data

## Value1 Value2  
## 1 1 5  
## 2 2 6  
## 3 3 7  
## 4 4 8

data <- mutate(data, add = Value1 + Value2); data

## Value1 Value2 add  
## 1 1 5 6  
## 2 2 6 8  
## 3 3 7 10  
## 4 4 8 12

data <- mutate(data, Value1 = Value2); data

## Value1 Value2 add  
## 1 5 5 6  
## 2 6 6 8  
## 3 7 7 10  
## 4 8 8 12

## Crear leyenda en un grafico

Importante crear el color y las lineas en un “aes” para despues llamar a la funcion y poder identificar las diferentes lineas.

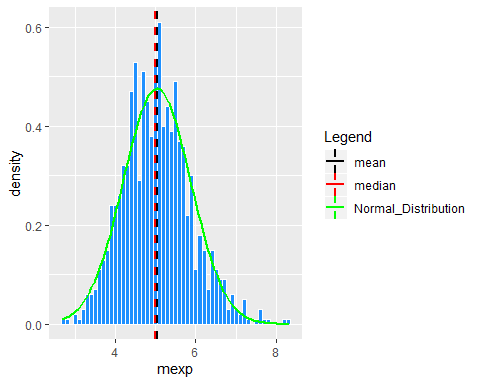
mexp = NULL  
lambda = 0.2  
for (i in 1 : 1000){  
 mexp = c(mexp, mean(rexp(40,lambda)))  
}  
sample\_mean <- mean(mexp) ; setNames(c(sample\_mean, 1/lambda),c("Sample Mean", "Theorical Mean"))

## Sample Mean Theorical Mean   
## 5.034487 5.000000

sample\_variance <- var(mexp); setNames(c(sample\_variance, (1/lambda)^2),c("Sample Variance", "Theorical Variance"))

## Sample Variance Theorical Variance   
## 0.7029568 25.0000000

data <- data.frame(name = c("Sample Mean", "Theorical Mean"), value = c(sample\_mean, 1/lambda))  
ggplot() + aes(mexp) + geom\_histogram(aes (y = ..density..),binwidth = 0.1, fill = "dodgerblue", color = "White") +   
 geom\_vline(data = data, mapping = aes(xintercept = value, color = c("mean", "median")), linetype = "dashed", size = 1) +  
 stat\_function(fun = dnorm, aes(color = "Normal\_Distribution"), args = list(mean = mean(mexp), sd = sd(mexp)), size = 1) +  
 scale\_color\_manual(name = "Legend", values = c(median = "red", mean = "black", Normal\_Distribution = "green"))



## Melt

Permite colapsar un dataframe

library(reshape2)

x = data.frame(  
 id = c(1, 1, 2, 2),  
 blue = c(1, 0, 1, 0),  
 red = c(0, 1, 0, 1)  
)  
  
# collapse the data frame  
melt(data = x, id.vars = "id", measure.vars = c("blue", "red"))

## id variable value  
## 1 1 blue 1  
## 2 1 blue 0  
## 3 2 blue 1  
## 4 2 blue 0  
## 5 1 red 0  
## 6 1 red 1  
## 7 2 red 0  
## 8 2 red 1

## Ifelse

Me permite crear un vector con una condicion

data <- data.frame(a = c(1,2,3,4,5,6,7), b = c(9,8,7,6,5,4,3))  
vector <- ifelse(data$a > 3, "Grande", "Chico"); vector

## [1] "Chico" "Chico" "Chico" "Grande" "Grande" "Grande" "Grande"