**Sesión 11 Generation random numbers**

Contenido: random numberseed in R, Random sampling, Data visualization using R

R. **Actividad 11**

Aplicaciones en R utilizando los datos de esta sesión.

Datos aleatorios

generar 100 datos del 15 al 50

datos= sample(15:50,size=100, replace=TRUE)

datos

## [1] 34 21 18 37 35 49 23 22 28 25 44 40 21 25 19 30 44 50 48 23 44 32 16

## [24] 28 39 35 18 16 37 29 17 20 23 19 42 39 25 50 49 17 41 49 25 50 28 39

## [47] 44 43 40 32 23 23 29 36 33 25 34 20 27 42 43 25 30 24 36 50 22 28 22

## [70] 19 36 31 18 33 28 27 18 33 24 19 49 37 48 49 29 47 22 27 34 17 21 24

## [93] 30 46 50 15 41 27 21 29

media

media=mean(datos)

media

## [1] 31.43

mediana

mediana=median(datos)

mediana

## [1] 29

maximo

maximo=max(datos)

maximo

## [1] 50

minimo

minimo=min(datos)

minimo

## [1] 15

frecuencia

frecuencia=table(datos)

freq.relativa=(frecuencia)/margin.table(frecuencia)

frecuencia

## datos

## 15 16 17 18 19 20 21 22 23 24 25 27 28 29 30 31 32 33 34 35 36 37 39 40 41

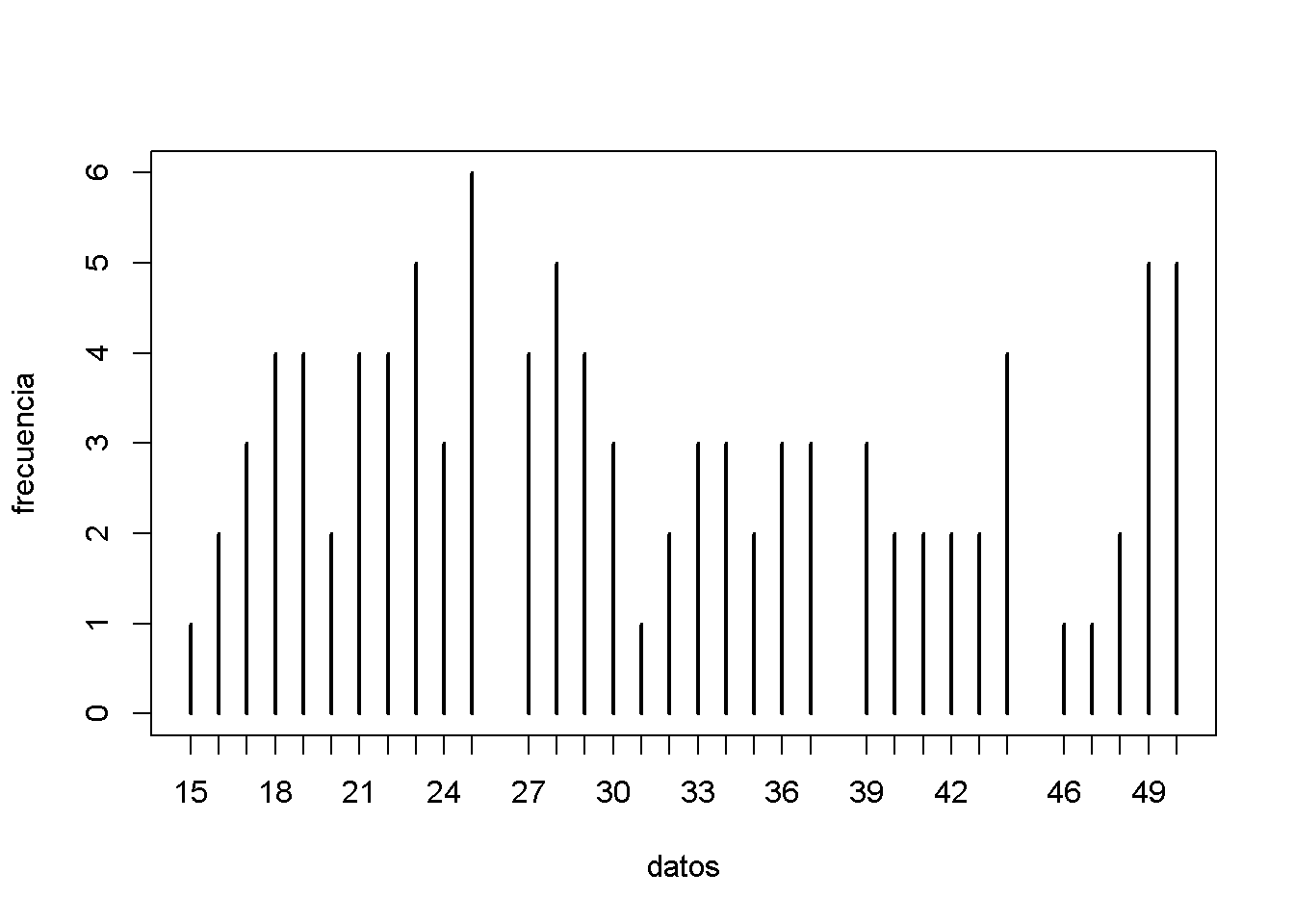
## 1 2 3 4 4 2 4 4 5 3 6 4 5 4 3 1 2 3 3 2 3 3 3 2 2

## 42 43 44 46 47 48 49 50

## 2 2 4 1 1 2 5 5

Plot

plot(frecuencia)

 knitr::opts\_chunk$set(echo = TRUE) ## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see [http://rmarkdown.rstudio.com](http://rmarkdown.rstudio.com/).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

summary(cars)

## speed dist

## Min. : 4.0 Min. : 2.00

## 1st Qu.:12.0 1st Qu.: 26.00

## Median :15.0 Median : 36.00

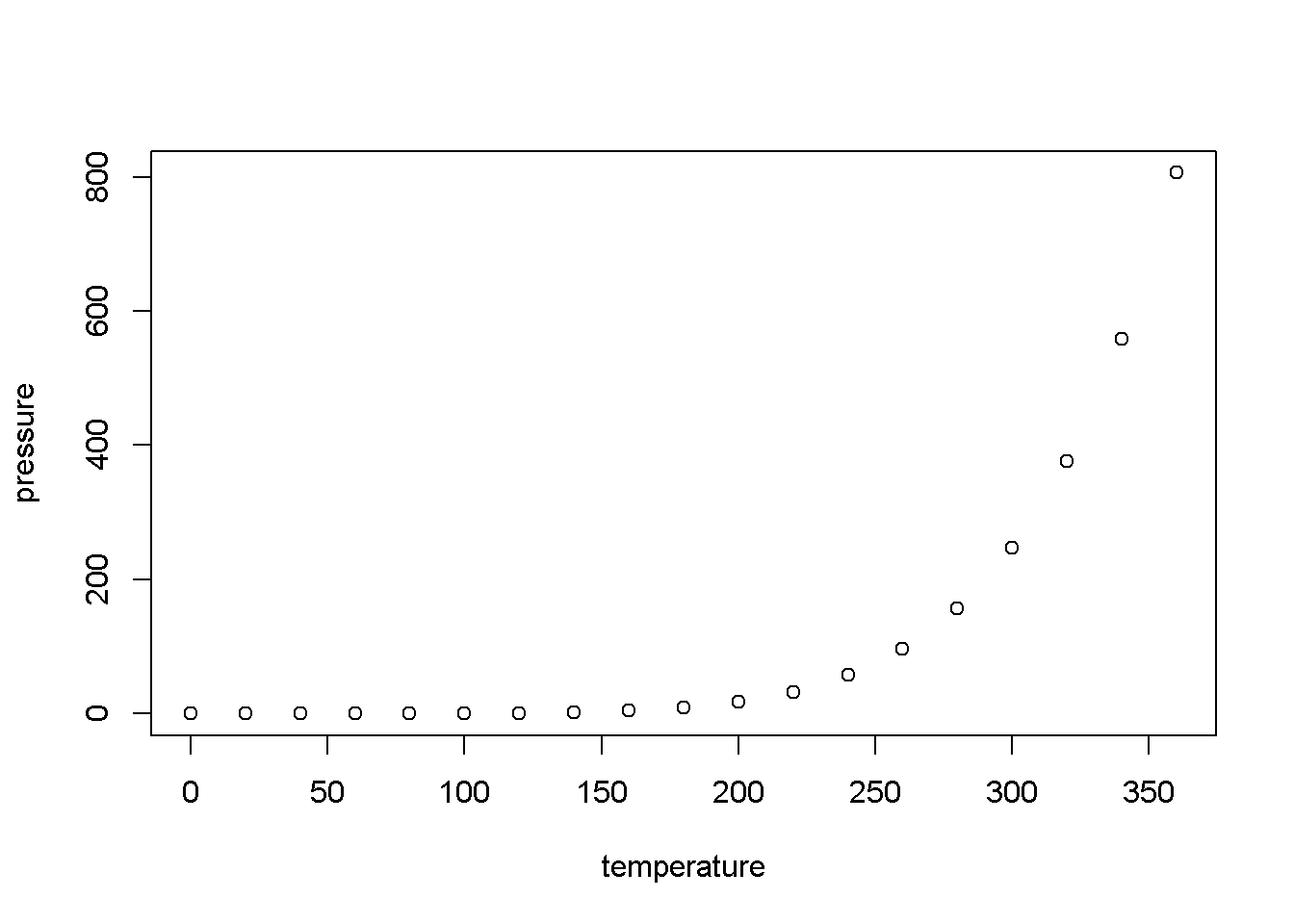
## Mean :15.4 Mean : 42.98

## 3rd Qu.:19.0 3rd Qu.: 56.00

## Max. :25.0 Max. :120.00

Including Plots

You can also embed plots, for example:



Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.

GENERAR UNA MUESTRA DE ELEMENTOS ALEATORIOS: Usando el comando sample()

En esta guía tratará de como crear una muestra aleatoria con el comando sample, esto es muy útil si se desea realizar rifas, sorteos, entre otros. Lo que realiza es de un vector universal escoger otro vector en donde los elementos han sido escogidos aleatoriamente.

CREAR ELEMENTOS DE MANERA ALEATORIA.

Para esto usaremos el comando sample que tiene la siguiente sintaxis.

sample("vector",size ="#", replace="T o F", prob="vector")

Donde sample es el comando para generar una muestra de manera aleatoria; vector es el vector en donde se encuentra el universo de elementos; en size irá el tamaño del vector, es decir, la cantidad de elementos deseados; replace es un argumento lógico en donde puede ir TRUE o FALSE, si es el primero le estaremos diciendo a **R** que para la selección de elementos se de con reemplazo caso contrario no se permitirá el reemplazo (el valor por defecto es FALSE); por útlimo, prob es el argumento que indica la probabilidad de ocurrencia de los elementos del vector universal.

Pero como se está escogiendo aleatoriamente cada vez que se corra el código se crearán vectores diferentes. Por tanto para que la aleatorización de elementos sea reproducible se usará el comando set.seed(), con este comando podremos reproducir los resultados obtenidos en cualquier momento.

A continuación unos ejemplos.

Si deseamos obtener de manera aleatoria un vector de elementos 1 y 2 y que tenga 20 elementos.

set.seed(1)

x<-sample(c(1,2), size = 20, replace = TRUE) *# Vector de 20 elementos*

x

## [1] 1 1 2 2 1 2 2 2 2 1 1 1 2 1 2 1 2 2 1 2

Como se puede observar primero definimos la semilla set.seed(1) esto para que al correr una y otra vez el mismo código nos resulte el mismo resultado. Posteriormente se ha generado el vector **x** de 1 y 2 como elementos, en donde se indica que el tamaño sea de 20 elementos (size=20), además se considera que se escoga con reemplazo. La cantidad de 1 y 2 resultante es aleatoria. Para ver la proporción usamos el comando fdt, pero primero se tendrá que instalar el paquete "fdth"

*# Para instalar el paquete fdth.*

install.packages("fdth")

Y cargamos el paquete:

**library**(fdth)

Ahora podremos usar el comando fdt para conocer las frecuencia de los elementos.

freq1<-fdt(x, k=2)

freq1

## Class limits f rf rf(%) cf cf(%)

## [0.99,1.5) 9 0.45 45 9 45

## [1.5,2) 11 0.55 55 20 100

De los 20 elementos se generaron 9 unos y 11 dos (esto se ve en la salida f) lo que da que la frecuencia relativa sea de 45% para el elemento 1 y 55% para el elemento 2 (esto se ve en la salida rf). Como se escoge de manera aleatoria esta proporción a la larga llevará a que la cantidad de 1 y 2 sea del 50% para cada elemento, esto es lo que indica la ley de grandes números. Veamos si se cumple tal afirmación.

set.seed(2)

x1<-sample(c(1,2), size = 100, replace = T) *# Vector de 100 elementos.*

x2<-sample(c(1,2), size = 1000, replace = T) *# Vector de 1000 elementos.*

x3<-sample(c(1,2), size = 100000, replace = T) *# Vector de 100000 elementos.*

x4<-sample(c(1,2), size = 1000000, replace = T) *# Vector de un millón de elementos.*

Mostraremos los resultados en tablas de frecuencias:

freqx1<-fdt(x1,k=2)

freqx2<-fdt(x2,k=2)

freqx3<-fdt(x3,k=2)

freqx4<-fdt(x4,k=2)

freqx1; freqx2; freqx3; freqx4

## Class limits f rf rf(%) cf cf(%)

## [0.99,1.5) 55 0.55 55 55 55

## [1.5,2) 45 0.45 45 100 100

## Class limits f rf rf(%) cf cf(%)

## [0.99,1.5) 497 0.5 49.7 497 49.7

## [1.5,2) 503 0.5 50.3 1000 100.0

## Class limits f rf rf(%) cf cf(%)

## [0.99,1.5) 49698 0.5 49.7 49698 49.7

## [1.5,2) 50302 0.5 50.3 100000 100.0

## Class limits f rf rf(%) cf cf(%)

## [0.99,1.5) 499555 0.5 49.96 499555 49.96

## [1.5,2) 500445 0.5 50.04 1000000 100.00

En efecto, como se puede observar, a mayor cantidad de elementos las distribuciones se acercan al 50%. Cuando se le indicó a **R** que realice una muestra de un millón de elementos las frecuencias relativas fueron de 49.96% para 1 y 50.04% para 2 (esto se ve en la salida rf(%)).

Pero nosotros podemos controlar las probabilidades de ocurrencia. Para esto, indicaremos la probabilidad con el argumento prob.

Si deseamos que la probabilidad de ocurrencia del elemento 1 sea del 20% y del elemento 2 sea del 80% (no necesariamente los porcentajes deben de sumar 100%).

*# 1 con 20% de probabilidad y 2 con 80% de probabilidad.*

set.seed(3)

x5<-sample(1:2, size = 20, replace = TRUE, prob = c(0.2,0.8))

x5

## [1] 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 1 2

En el argumento prob = c(0.2,0.8) es donde se indica que 1 tenga una probabilidad de ocurrencia de 20% y 2 de 80%.

Lo mostraremos en una tabla de frecuencias:

freqx5<-fdt(x5, k=2)

freqx5

## Class limits f rf rf(%) cf cf(%)

## [0.99,1.5) 4 0.2 20 4 20

## [1.5,2) 16 0.8 80 20 100

En efecto, se mantiene la probabilidad de ocurrencia señalada (Las frecuencias relativas indican 20% para 1 y 80% para 2). Es curioso que en este ejemplo nos haya resultado con 20% y 80% en las frecuencias relativas, esto no sucede comunmente, si usted desea vuelva a correr este código sin la semilla y verá que el resultado es distinto. Es decir, para este ejemplo se tuvo un chispazo de buena suerte.

A continuación se muestra un ejemplo para un vector caracter.

set.seed(4)

x6<-sample(c("Masculino","Femenino"), size = 30, replace = TRUE)

x6

## [1] "Femenino" "Masculino" "Masculino" "Masculino" "Femenino"

## [6] "Masculino" "Femenino" "Femenino" "Femenino" "Masculino"

## [11] "Femenino" "Masculino" "Masculino" "Femenino" "Masculino"

## [16] "Masculino" "Femenino" "Femenino" "Femenino" "Femenino"

## [21] "Femenino" "Femenino" "Femenino" "Masculino" "Femenino"

## [26] "Femenino" "Masculino" "Femenino" "Femenino" "Femenino"

Los códigos indicaron a **R** que seleccione una muestra de 30 elementos entre **Masculino** y **Femenino** con reemplazo.

Para verlo mejor lo presentamos en una tabla:

table(x6)

## x6

## Femenino Masculino

## 19 11

Se escogió una muestra con 19 Femeninos y 11 Masculinos.

Recordad:

* El comando set.sed() puede tomar como semilla cualquier número.
* Si usted usa una semilla determinada y realiza con esa semilla otra simulación, los resultados que se obtendrán serán iguales.

Para muestra el siguiente ejemplo:

*# Usamos la semilla 1.*

set.seed(1)

z<-sample(c("Lunes", "Martes"), size = 20, replace = T)

table(z)

## z

## Lunes Martes

## 9 11

como se observa Lunes se escogió 9 veces y Martes 11 veces (En el ejemplo de 1 y 2; con la semilla set.seed(1) los resultados fueron 9 veces 1 y 11 veces 2).

* Por tanto, es recomandable no usar semillas iguales.

En la siguiente guía se verá el tema de data frame.