Script3

Yosune Miquelajauregui

14/12/2017

Script de la tercera clase de estadística y modelación de sistemas socioecológicos en R.

Familia apply y graficación

Introducción al uso de funciones apply()

```
x <- matrix(rnorm(30,0.5,0.5), nrow=5, ncol=6)
head(x)
```

```
##            [,1]         [,2]      [,3]        [,4]        [,5]
## [1,]  1.1365905 -0.513696252 0.1922149  1.05134772  0.05365742
## [2,]  0.6545673  0.005077909 0.2320833 -0.06982773  0.38260089
## [3,]  0.6759349  0.557732760 0.4103448  0.26567783 -0.70557520
## [4,] -0.3078438  0.425511605 0.3272029  0.90548569  0.33163071
## [5,]  1.3168986  0.875075427 0.2238877  0.40774854  0.40875370
##              [,6]
## [1,]  0.697403966
## [2,] -0.661241239
## [3,]  0.007631297
## [4,]  0.387036513
## [5,]  0.486941524
```

Obtener la suma y el promedio para cada columna y renglón

```
apply(x,2,sum) ## suma de columnas
```

```
## [1] 3.4761476 1.3497014 1.3857337 2.5604320 0.4710675 0.9177721
```

```
apply(x,1,sum) ## suma de renglones
```

```
## [1] 2.6175183 0.5432604 1.2117464 2.0690236 3.7193055
```

```
apply(x,1,mean) ## promedio de renglones
```

```
## [1] 0.43625305 0.09054341 0.20195774 0.34483727 0.61988425
```

```
apply(x,2,mean) ## promedio de columnas
```

```
## [1] 0.6952295 0.2699403 0.2771467 0.5120864 0.0942135 0.1835544
```

Podemos crear una función y evaluarla dentro del apply()

```
apply (x,1,function(x) x/2)
```

```
##             [,1]         [,2]         [,3]       [,4]      [,5]
## [1,]  0.56829526  0.327283648  0.337967464 -0.1539219 0.6584493
## [2,] -0.25684813  0.002538954  0.278866380  0.2127558 0.4375377
## [3,]  0.09610747  0.116041652  0.205172415  0.1636015 0.1119438
## [4,]  0.52567386 -0.034913864  0.132838913  0.4527428 0.2038743
## [5,]  0.02682871  0.191300444 -0.352787600  0.1658154 0.2043768
## [6,]  0.34870198 -0.330620620  0.003815648  0.1935183 0.2434708
```

```
primerafuncion <- function (x){
     x <- x*5
     x
  }
apply (x,2,primerafuncion)
```

```
##           [,1]        [,2]      [,3]       [,4]       [,5]        [,6]
## [1,]  5.682953 -2.56848126 0.9610747  5.2567386  0.2682871  3.48701983
## [2,]  3.272836  0.02538954 1.1604165 -0.3491386  1.9130044 -3.30620620
## [3,]  3.379675  2.78866380 2.0517241  1.3283891 -3.5278760  0.03815648
## [4,] -1.539219  2.12755803 1.6360146  4.5274285  1.6581535  1.93518256
## [5,]  6.584493  4.37537714 1.1194384  2.0387427  2.0437685  2.43470762
```

Utilizar los datos implementados en R

```
attach(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Utilizar tapply() sobre una hoja de datos para una variable determinada dada por un factor. El resultado es un vector. Obtener la media del ancho de los pétalos por especie.

```
tapply(iris$Petal.Width,iris$Species, mean)
```

```
##     setosa versicolor  virginica
##      0.246      1.326      2.026
```

Obtener la suma del ancho de los pétalos por especie.

```
tapply(iris$Petal.Width,iris$Species, sum)
```

```
##     setosa versicolor  virginica
##       12.3       66.3      101.3
```

Se utiliza lapply() cuando se quiere evaluar una función en una lista. El resultado es también una lista.

```
c1 <- data.frame (1,1:10)
c2 <- data.frame (1, 20:30)
mi.lista <- list (c1, c2)
lapply (mi.lista, sum)

## [[1]]
## [1] 65
##
## [[2]]
## [1] 286
```

Se utiliza sapply() cuando se quiere evaluar una función en una lista. El resultado es un vector.

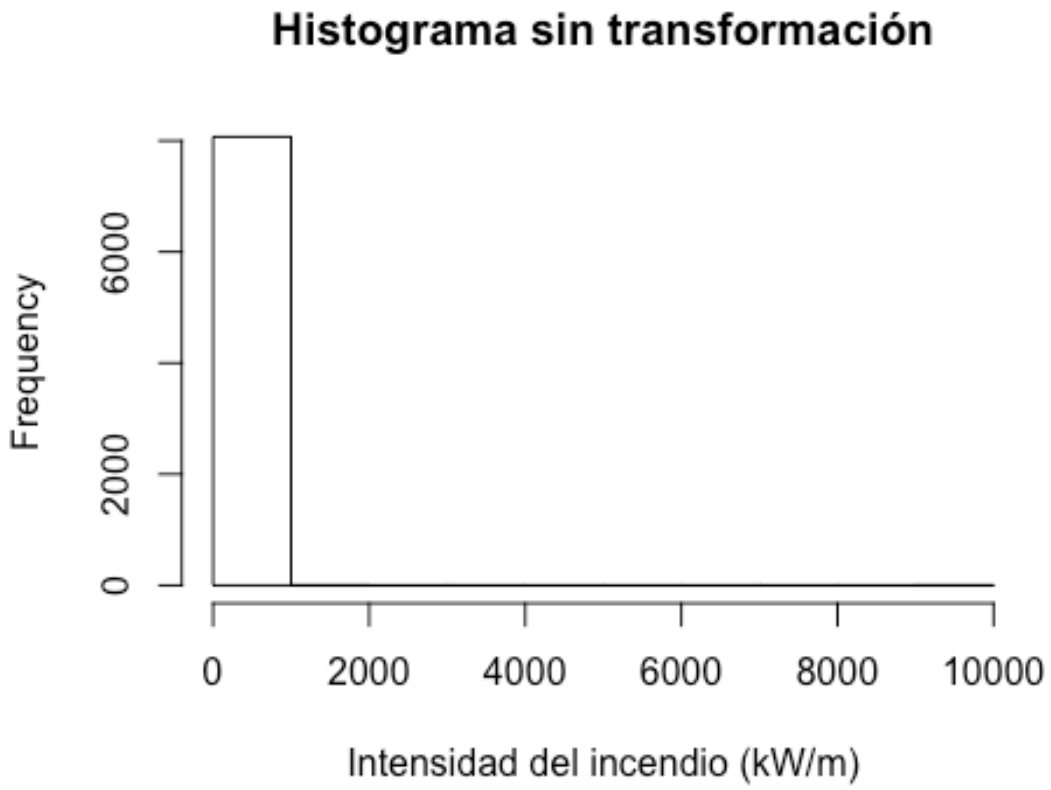
```
sapply (mi.lista, sum)

## [1]  65 286
```

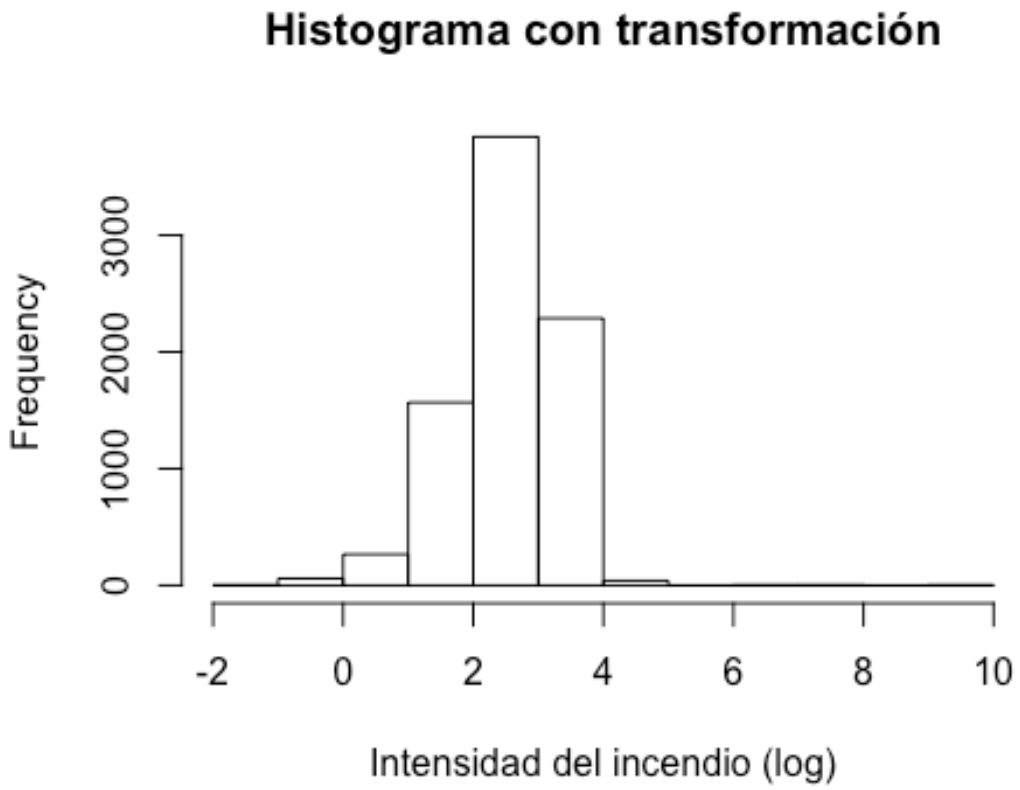
Bestiario de gráficas

1. Histograma: se utiliza para visualizar la distribución de una variable numérica

```
Incendio<-read.table("Fire_intensity.txt", header = T)
Incendio <- na.omit(Incendio)
hist(Incendio$Fire.intensity,main = "Histograma sin transformación",xlab
= "Intensidad del incendio (kW/m)")
```




```
Incendio$logFire.intensity <-
  log(Incendio$Fire.intensity)
hist(Incendio$logFire.intensity, xlab = "Intensidad del incendio (log)",
main = "Histograma con transformación")
```

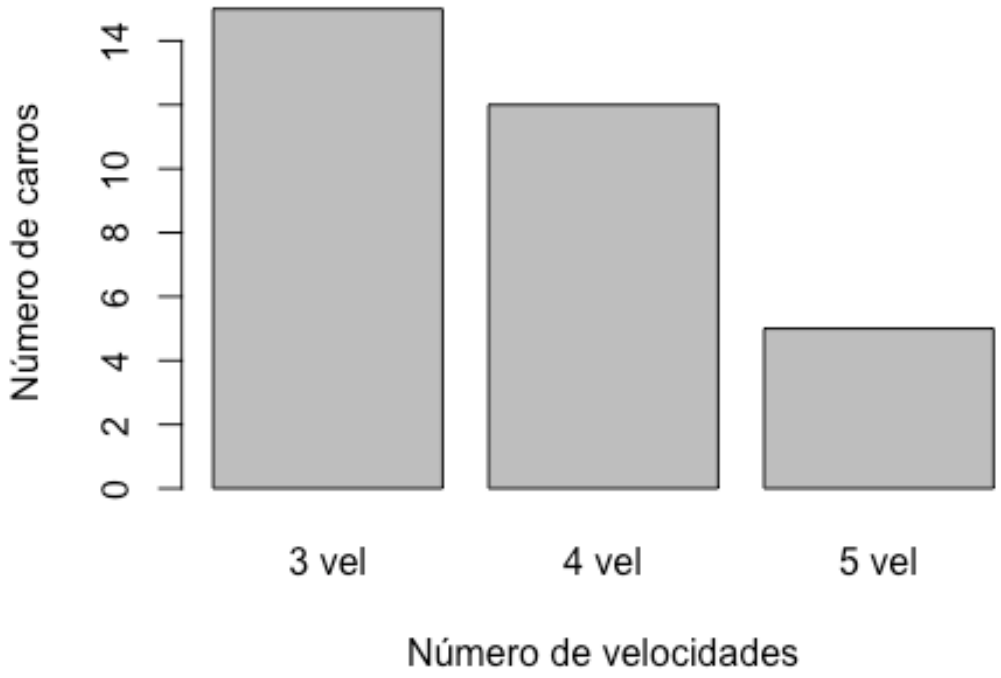


2. Barras: se utiliza para graficar frecuencias, medias,desviación estándar.

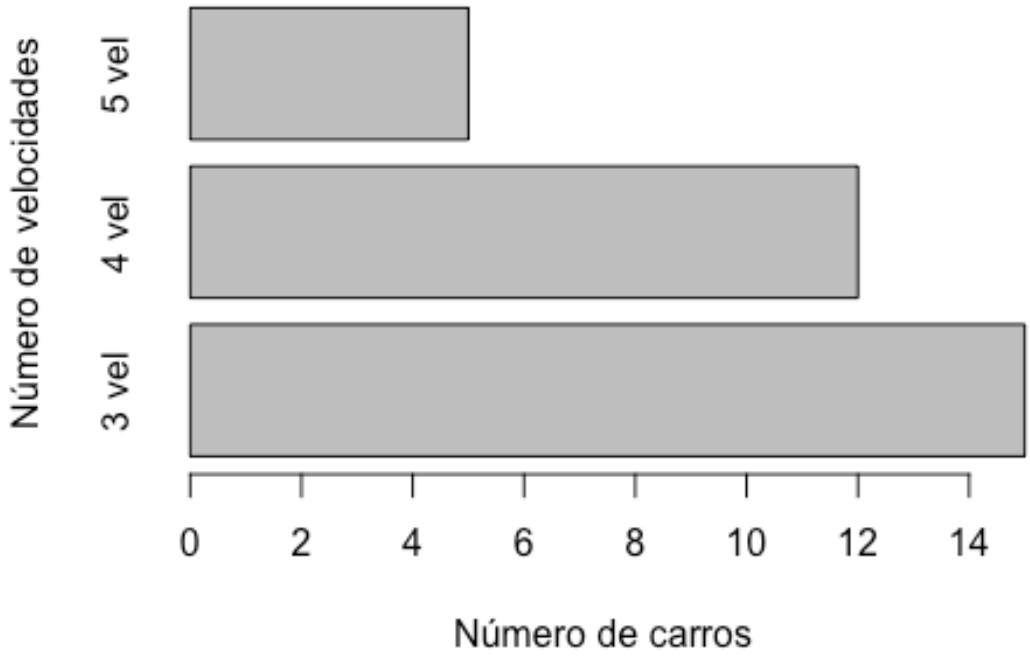
```
carros <- mtcars[c(1,2,6,10)]
names(carros)<- c("eficiencia","cilindros","peso","velocidades")
str(carros)

## 'data.frame':    32 obs. of  4 variables:
##  $ eficiencia : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cilindros  : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ peso       : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ velocidades: num  4 4 4 3 3 3 3 4 4 4 ...

conteo <- table(carros$velocidades)
barplot(conteo,names.arg=c("3 vel","4 vel","5 vel"),xlab="Número de veloc
idades",ylab="Número de carros") # barras verticales
```

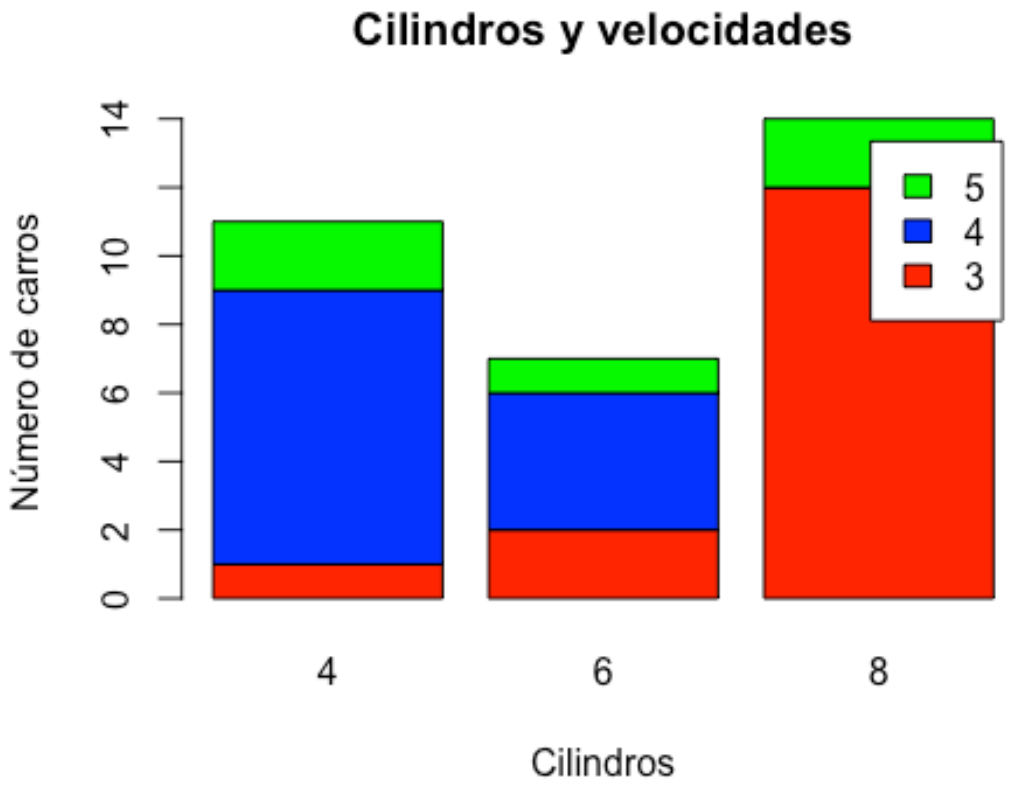



```
barplot(conteo, ylab="Número de velocidades",xlab="Número de carros", hor
iz=TRUE,names.arg=c("3 vel","4 vel","5 vel")) # barras horizontales
```

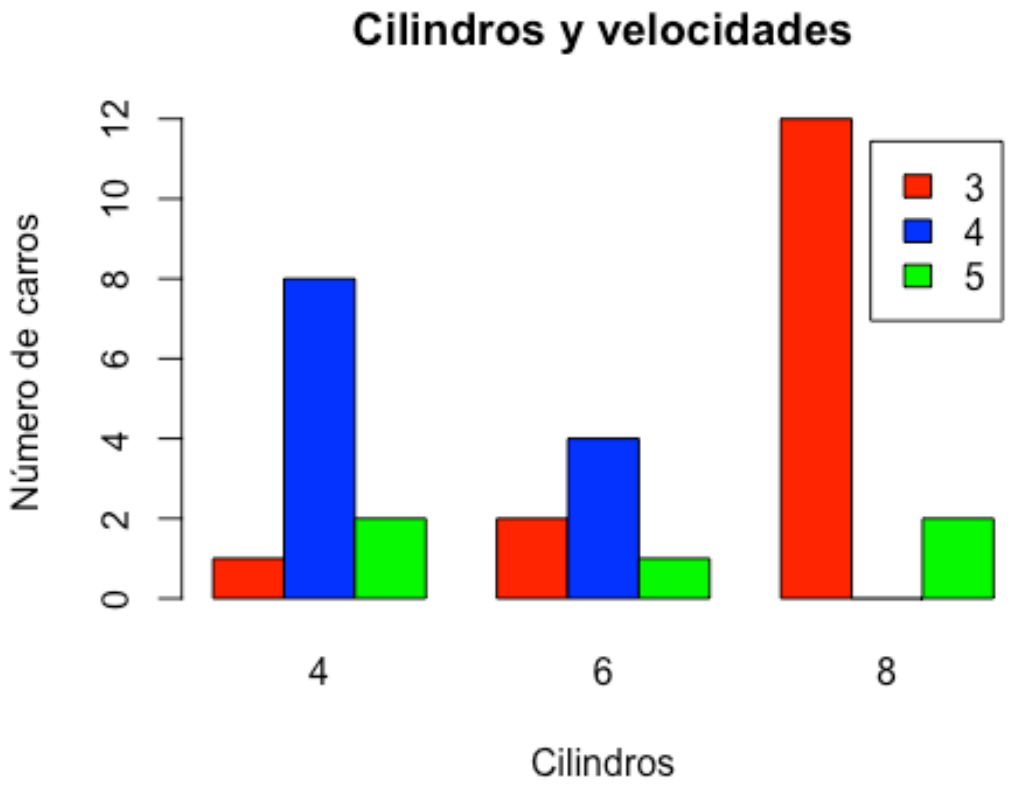


3. Barras apiladas

```
conteo2 <- table(carros$velocidades,carros$cilindros)
barplot(conteo2, main="Cilindros y velocidades"
, xlab="Cilindros",ylab= "Número de carros",col=c("red","blue","green"),
legend=rownames(conteo2))
```

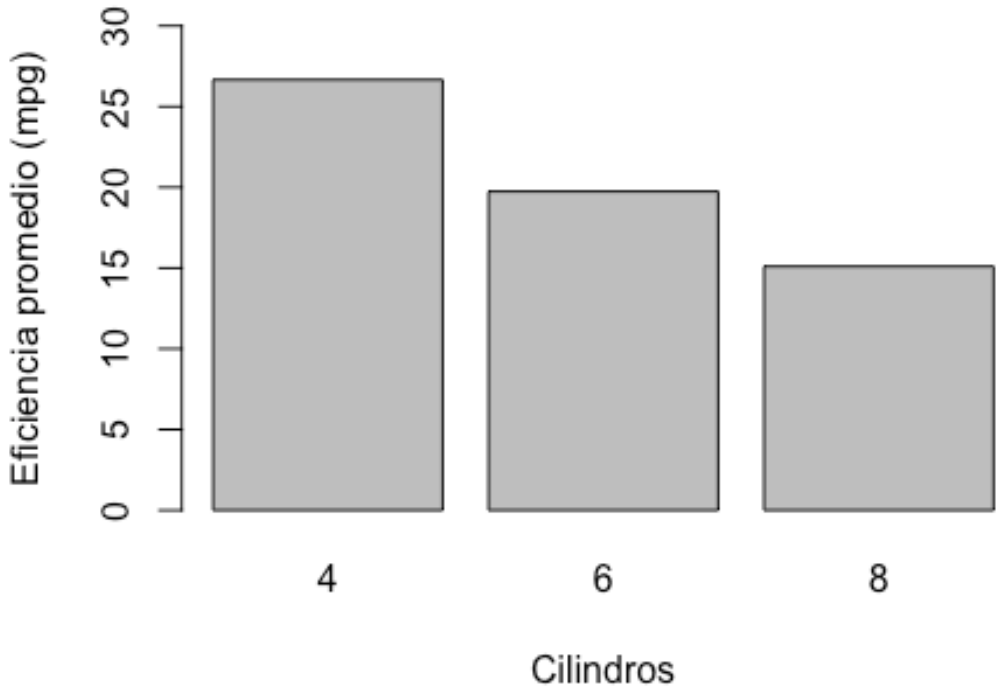



```
barplot(conteo2, main="Cilindros y velocidades"
, xlab="Cilindros",ylab="Número de carros",col=c("red","blue","green"),le
gend = rownames(conteo2),beside=TRUE)
```



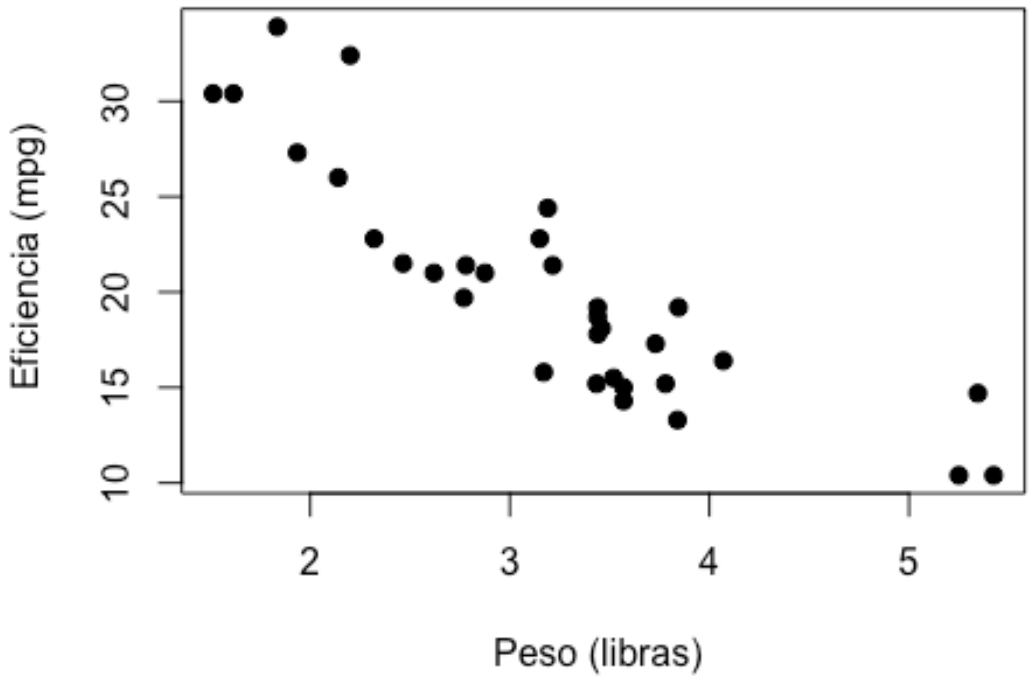
4. Usar la función aggregate() para obtener medias y otras métricas para una o más variables numéricas ygraficar con barplot()

```
aggdatos <-aggregate(carros$eficiencia,by=list(carros$cilindros),FUN=mean
, na.rm=TRUE)
names(aggdatos) <- c("Cilindros","Eficiencia")
barplot(aggdatos$Eficiencia, ylab= "Eficiencia promedio (mpg)",xlab="Cili
ndros", names.arg=aggdatos$Cilindros, ylim=c(0,30))
```



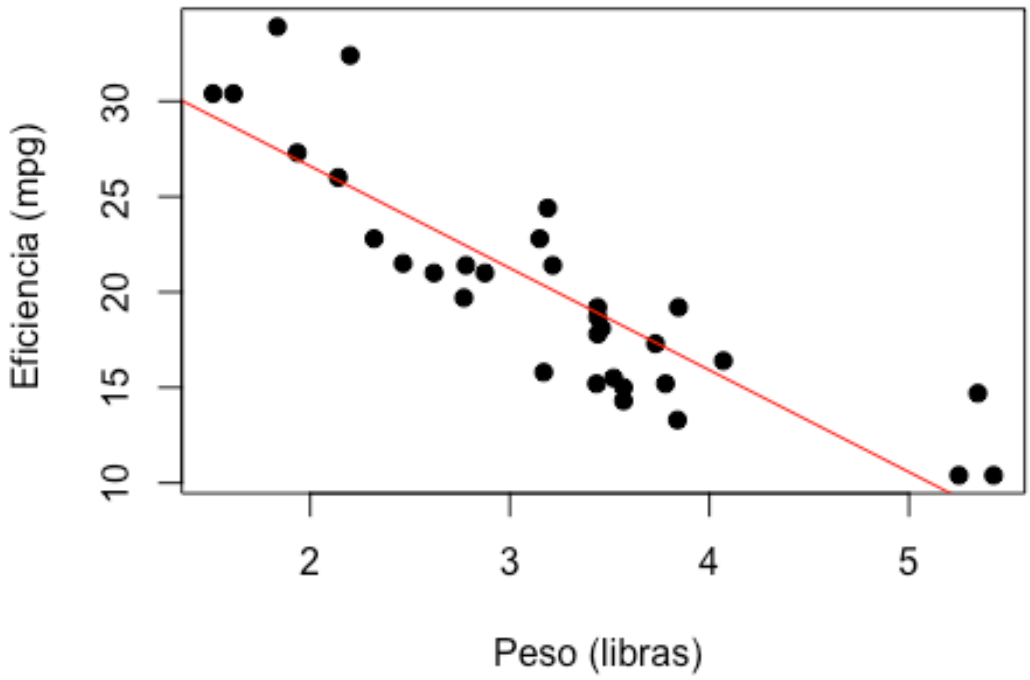
5. Gráfico de dispersión x-y plot()

```
plot(carros$peso, carros$eficiencia,
    xlab="Peso (libras) ", ylab="Eficiencia (mpg) ", pch=19)
```



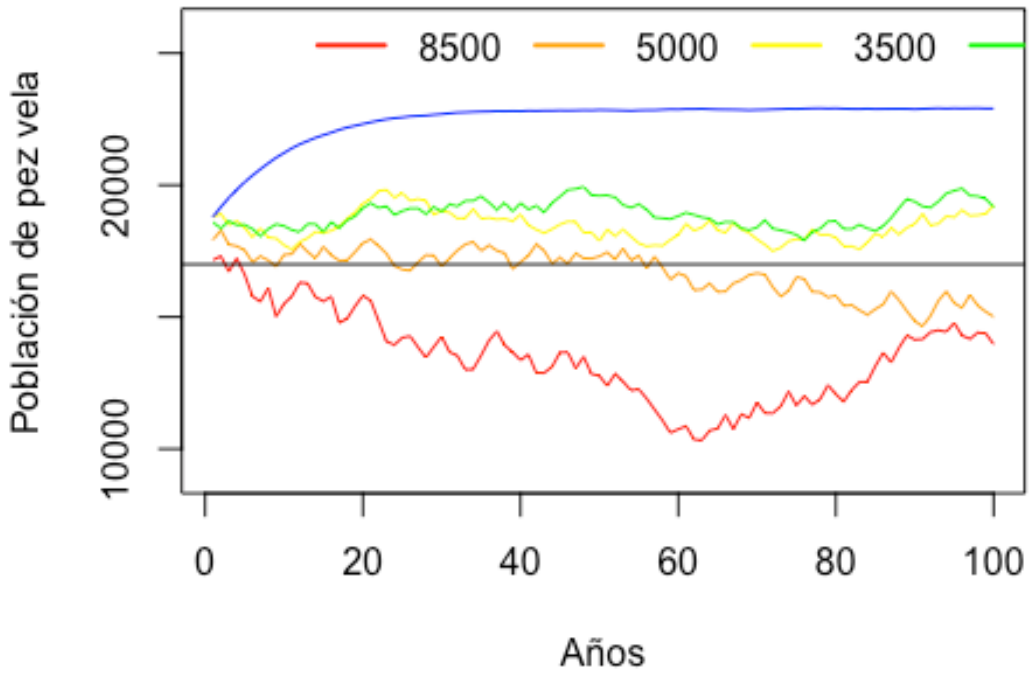
6. Añadir línea de ajuste y~x

```
plot(carros$peso, carros$eficiencia,
    xlab="Peso (libras) ", ylab="Eficiencia (mpg) ", pch=19)
abline(lm(carros$eficiencia~carros$peso), col="red")
```



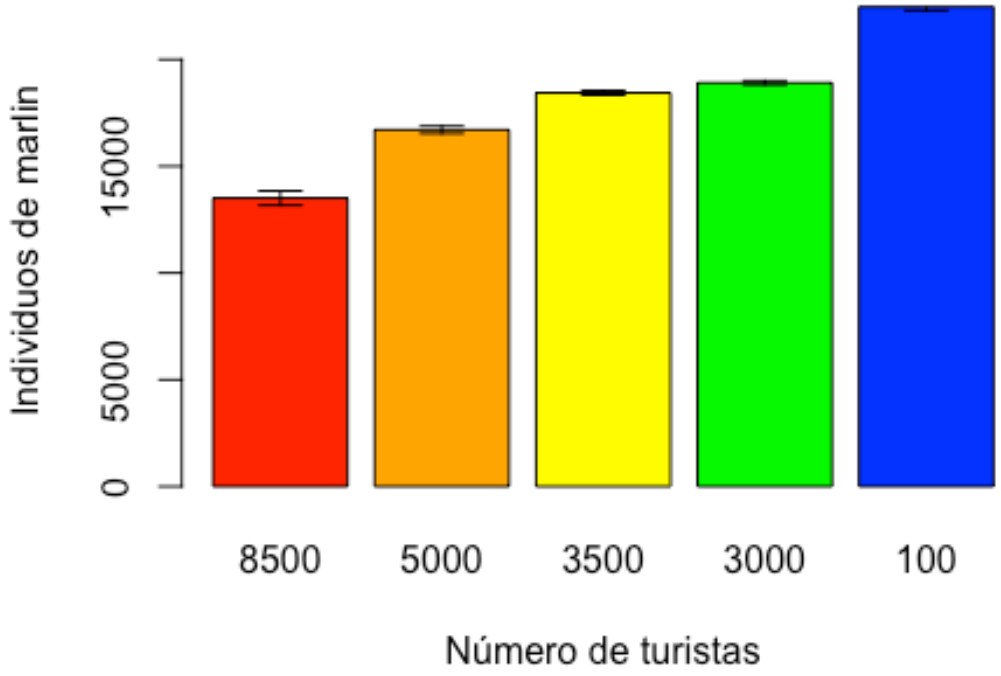
7. Grafica de lineas

```
ModeloResultados <- read.csv(file="ModeloPezVela.csv", header=T)
Años <- seq(1,100)# crear secuencia
plot(Años,ModeloResultados[,1],type="l", col="red", ylim = c(9000,26000),
ylab = "Población de pez vela")
lines(Años,ModeloResultados[,2],type="l", col="orange")
lines(Años,ModeloResultados[,3],type="l", col="yellow")
lines(Años,ModeloResultados[,4],type="l", col="green")
lines(Años,ModeloResultados[,5],type="l", col="blue")
abline( h = 17000)
legend(10,27000, c("8500","5000","3500","3000","100"), horiz=TRUE,bty="n"
,
lty=1,lwd=2,col=c("red","orange","yellow","green","blue"))
```



8. Obtener medias con colMeans() y/o apply(), calcular desviaciones estándar y graficar

```
names(ModeloResultados)<- c("8500","5000","3500","3000","100")
###función para graficar desviación estándar
error.bar <- function(x, y, upper, lower=upper, length=0.1,...){
if(length(x) != length(y) | length(y) !=length(lower) | length(lower) !=
length(upper))
stop("vectors must be same length")
arrows(x,y+upper, x, y-lower, angle=90, code=3, length=length, ...)
}
###Calcular medias
medias <- colMeans(ModeloResultados)
ymedia <- apply(ModeloResultados,2,mean)
ed.y <- apply(ModeloResultados,2,sd)
barx <-barplot(ymedia,col=c("red","orange","yellow","green","blue"), xlab
="Número de turistas", ylab="Individuos de marlin")
error.bar(barx,ymedia, 1.96*ed.y/10)
```



9.Para hacer gráficas en ggplot los datos deben estar estructurados como hoja de datos y en formato “largo” en vez de “ancho”.

```
#formato "largo"
#crear nueva variable "Escenario"
E <- c("8500","5000","3500","3000","100")
E2 <- rep(E, each=100)
E8500 <- ModeloResultados[,1]
E5000 <- ModeloResultados[,2]
E3500 <- ModeloResultados[,3]
E3000 <- ModeloResultados[,4]
E100  <- ModeloResultados[,5]
DataE <- c (E8500,E5000,E3500,E3000,E100)
Data <- as.data.frame(E2)
Data$Poblacion <- DataE
names(Data)<- c("Escenario","Poblacion")
Data$Escenario <- as.factor(Data$Escenario)
Data$Poblacion <- as.numeric(Data$Poblacion)
```

Función para calcular media, des. estandar, intervalos de confianza

```
summarySE <- function(data=NULL, measurevar, groupvars=NULL, na.rm=FALSE,
conf.interval=.95, .drop=TRUE) {
library(plyr)
```

```
  length2 <- function (x, na.rm=FALSE) {
  if (na.rm) sum(!is.na(x))
    else length(x)
    }

datac <- ddply(data, groupvars, .drop=.drop,
.fun = function(xx, col) {
c(N= length2(xx[[col]], na.rm=na.rm),
mean = mean (xx[[col]], na.rm=na.rm),
sd   = sd (xx[[col]], na.rm=na.rm))
  },measurevar)

datac <- rename(datac, c("mean" = measurevar))
datac$se <- datac$sd / sqrt(datac$N)
ciMult <- qt(conf.interval/2 + .5, datac$N-1)
datac$ci <- datac$se * ciMult
return(datac)
}
```

Gráfico con 95% intervalos de confianza

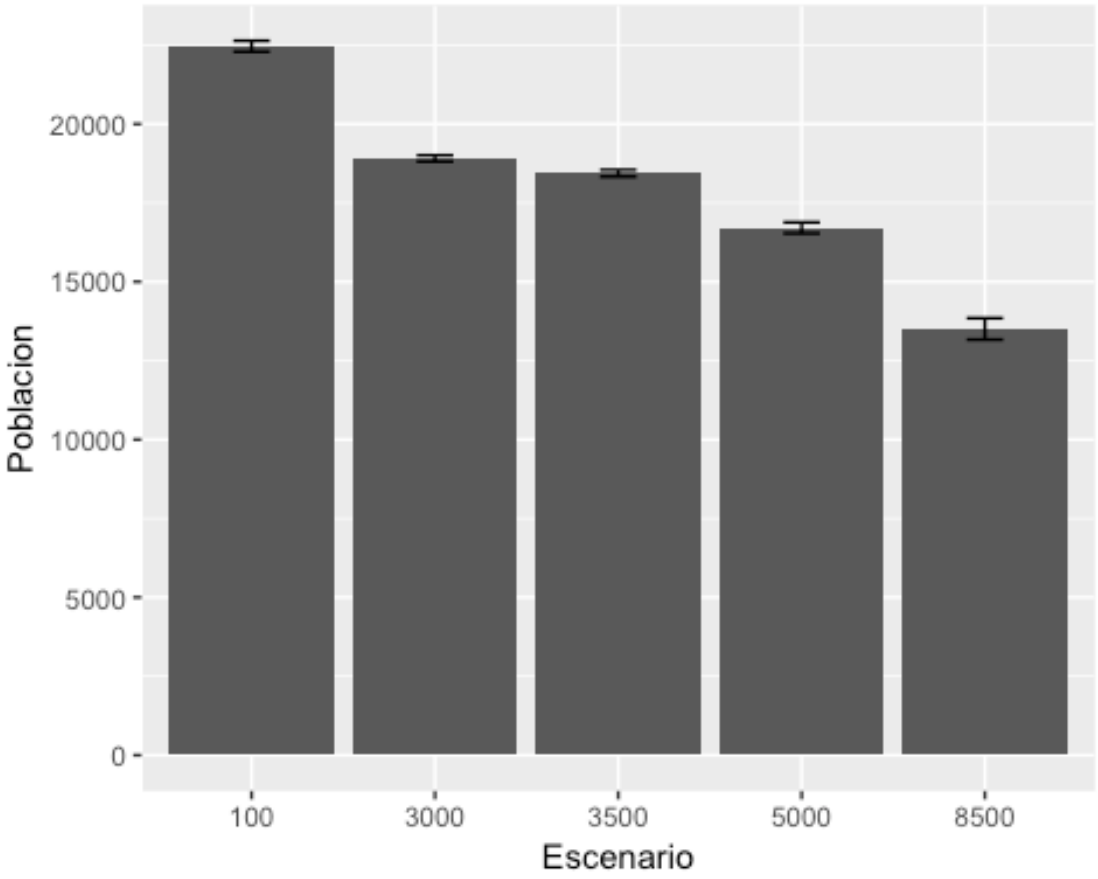
```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.3.2

resumen <- summarySE(Data, measurevar="Poblacion", groupvars=c("Escenario
"))
resumen

##   Escenario   N Poblacion        sd        se        ci
## 1       100 100  22463.37  868.2680  86.82680 172.28322
## 2      3000 100  18907.68  493.3825  49.33825  97.89778
## 3      3500 100  18437.15  534.2071  53.42071 105.99828
## 4      5000 100  16708.59  877.5461  87.75461 174.12419
## 5      8500 100  13507.28 1717.8586 171.78586 340.86042

grafica <- ggplot(resumen, aes(x=Escenario, y=Poblacion)) +
geom_bar(position=position_dodge(),stat="identity") + geom_errorbar(aes(y
min=Poblacion-ci, ymax=Poblacion+ci),width=.2,                  position
=position_dodge(.9))
grafica
```


Poblacion
20000
15000
10000
5000
0
100
3000
3500
5000
8500
Escenario