


# Estadística y modelación de sistemas socioecológicos en R



Laboratorio  
Nacional  
de Ciencias  
de la Sostenibilidad

**Dra. Yosune Miquelajauregui Graf**

# Plan del día

1. Familia de funciones “apply ()”
  2. Graficación básica
  3. Ejercicios
- 


# Familia de funciones “apply ()”

- Las funciones `apply ()` están diseñadas para utilizarse como alternativas de los bucles.
- La familia `apply ()` contiene funciones para manipular secciones de datos de matrices, listas y hojas de datos de manera repetitiva. Es decir, permite llevar a cabo operaciones con pocas líneas de código.
- Ejemplos son : `apply()`, `tapply()`, `lapply()`, `sapply()`, `vapply()`, `mapply()`, y `rapply()`. El uso de cada una de ellas depende de la estructura del objeto y del tipo de resultado que se desea obtener.

# Familia de funciones “apply ()”

apply () :

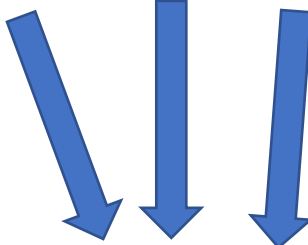
```
x <- matrix(rnorm(30,0.5,0.5), nrow=5, ncol=6)
```

x		<b>Dimensión 2</b> 					
<b>Dimensión 1</b>		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
	[1,]	-0.10508735	0.1851367	1.05801854	0.38356653	0.79066192	0.78990003
	[2,]	0.93453881	0.5663632	0.53933448	1.07830640	1.11610311	1.26004642
	[3,]	0.03519395	0.9910418	0.83950909	1.22801833	0.41278923	0.21095532
	[4,]	0.09143190	-0.1411810	0.06006402	0.64694183	0.56570592	-0.45893986
	[5,]	0.94650379	0.5647327	0.64440925	0.04881696	-0.06497283	0.02686407

# Familia de funciones “apply ()”

apply () :

objeto   columna   función

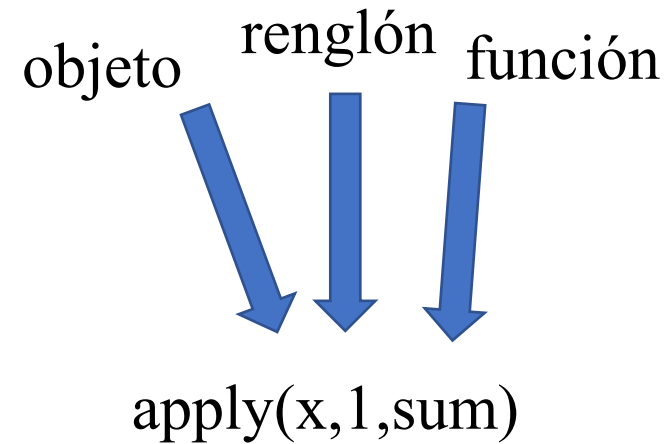


apply(x,2,sum)

1.902581 2.166093 3.141335 3.385650 2.820287 1.828826

# Familia de funciones “apply ()”

apply () :

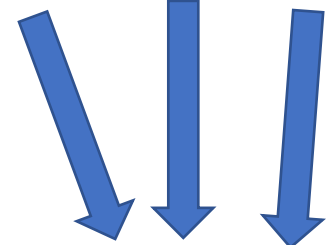


3.1021964 5.4946924 3.7175078 0.7640228 2.1663539

# Familia de funciones “apply ()”

apply () :

objeto    renglón    función



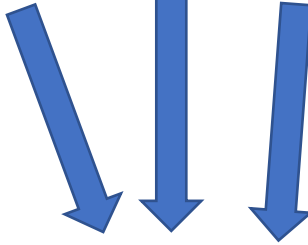
apply(x,1,mean)

0.5170327 0.9157821 0.6195846 0.1273371 0.3610590

# Familia de funciones “apply ()”

apply () :

objeto   columna   función



apply(x,2,mean)

0.3805162 0.4332187 0.6282671 0.6771300[5] 0.5640575 0.3657652



# Familia de funciones “apply ()”

apply () :

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

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	-0.05254367	0.4672694	0.01759697	0.04571595	0.47325190
[2,]	0.09256837	0.2831816	0.49552092	-0.07059052	0.28236635
[3,]	0.52900927	0.2696672	0.41975454	0.03003201	0.32220462
[4,]	0.19178326	0.5391532	0.61400916	0.32347091	0.02440848
[5,]	0.39533096	0.5580516	0.20639461	0.28285296	-0.03248641
[6,]	0.39495002	0.6300232	0.10547766	-0.22946993	0.01343204

# Familia de funciones “apply ()”

apply () :

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

```
apply (x,2,primerafuncion)
```

# Familia de funciones “apply ()”

tapply () :

attach(iris)

```
str(iris)'data.frame':
```

```
150 obs. of 5 variables:
```

```
$ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
```

```
$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
```

```
$ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
```

```
$ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
```

```
$ Species: Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 1
```

```
...
```

# Familia de funciones “apply ()”

`tapply ()` :Utilizarla sobre una hoja de datos para una variable determinada dada por un factor. El resultado es un vector.

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

setosa	versicolor	virginica
0.246	1.326	2.026

# Familia de funciones “apply ()”

tapply () :

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

setosa	versicolor	virginica
12.3	66.3	101.3

# Familia de funciones “apply ()”

`lapply ()`: Se utiliza una función dada a una lista y se obtiene una lista como resultado.


```
lapply (mi.lista, sum)
```

```
[[1]]
```

```
[1] 65
```

```
[[2]]
```

```
[1] 286
```



# Familia de funciones “apply ()”

sapply (): Se utiliza una función dada a una lista y se obtiene un vector como resultado.

```
sapply(mi.lista, sum)  
[1] 65 286
```

# Graficación: Histogramas

```
Incendio<-read.table("Fire_intensity.txt", header = T)
```

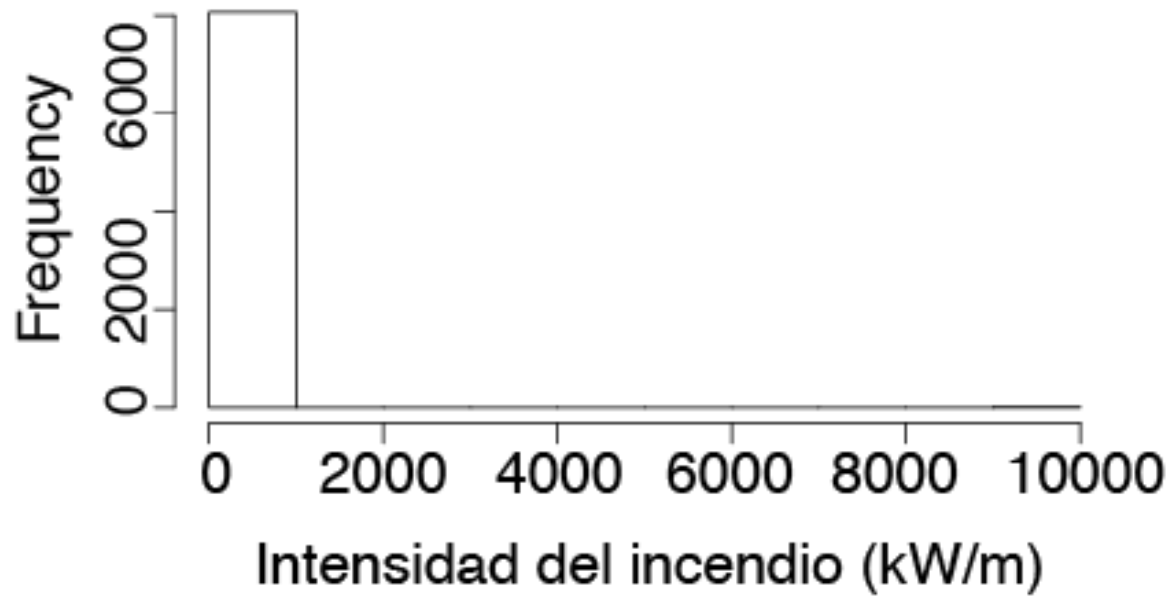
```
str(Incendio)'data.frame':      8077 obs. of  7 variables:  
 $ Probability.mortality: num  0.0401 0.0237 0.0296 0.0664  
 $ Stand.age : Factor w/ 3 levels "Mature","Old", "Young"  
 $ Fire.intensity : num  13.97 8.12 10.19 23.75 25.3 ...  
 $ Region: Factor w/ 3 levels "A2","C3","D4"  
 $ Stand.id : int  1 1 1 1 1 1 1 1 2 2  
 $ Tree.id : int  1 2 3 4 5 6 7 10 11 12  
 $ logFire.intensity : num  2.64 2.09 2.32 3.17 3.23
```



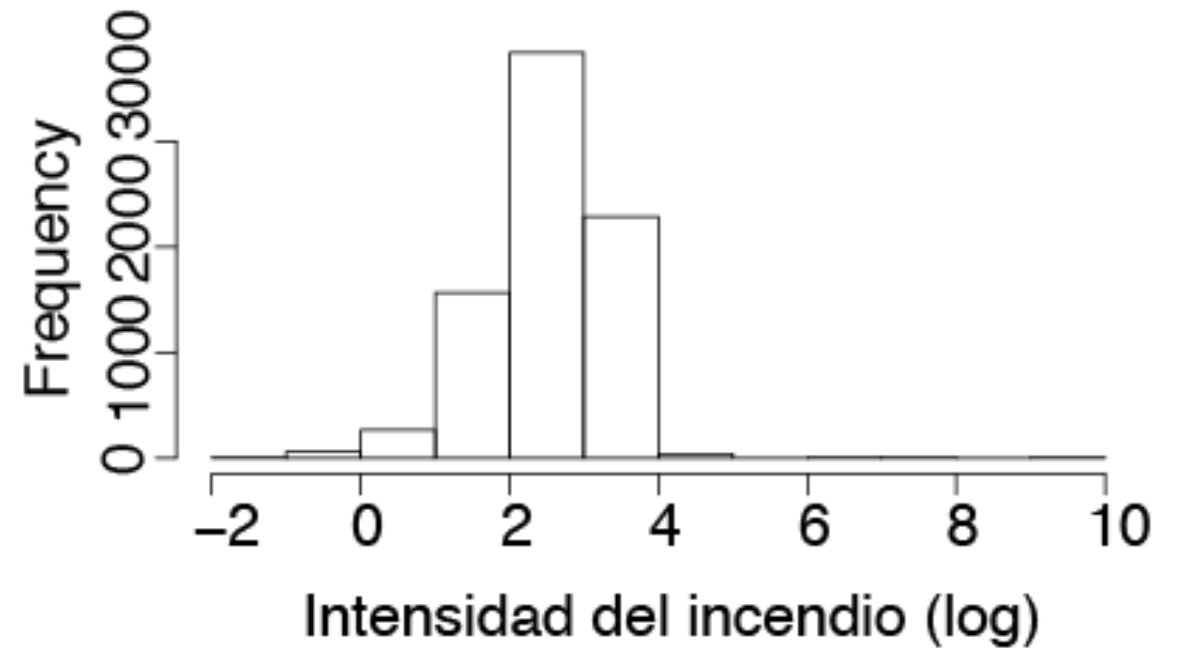
# Graficación: Histogramas

**Función R : hist ()**

**Histograma sin transformación**



**Histograma con transformación**

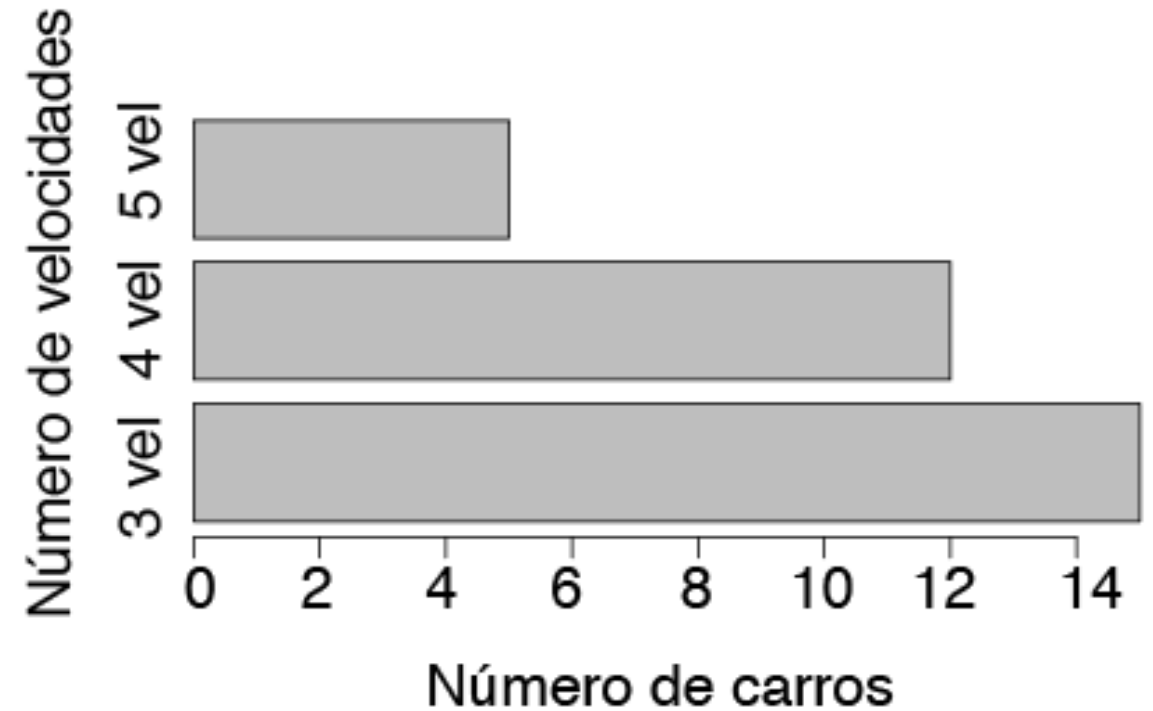
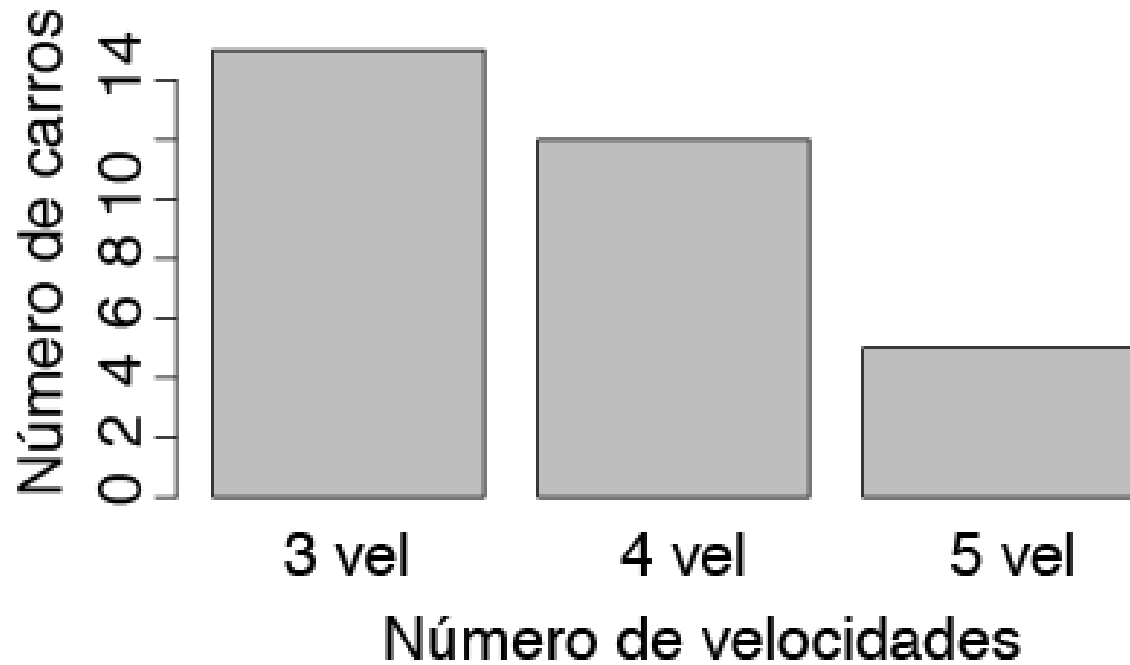


# Graficación: Barras

```
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  
$ 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
```

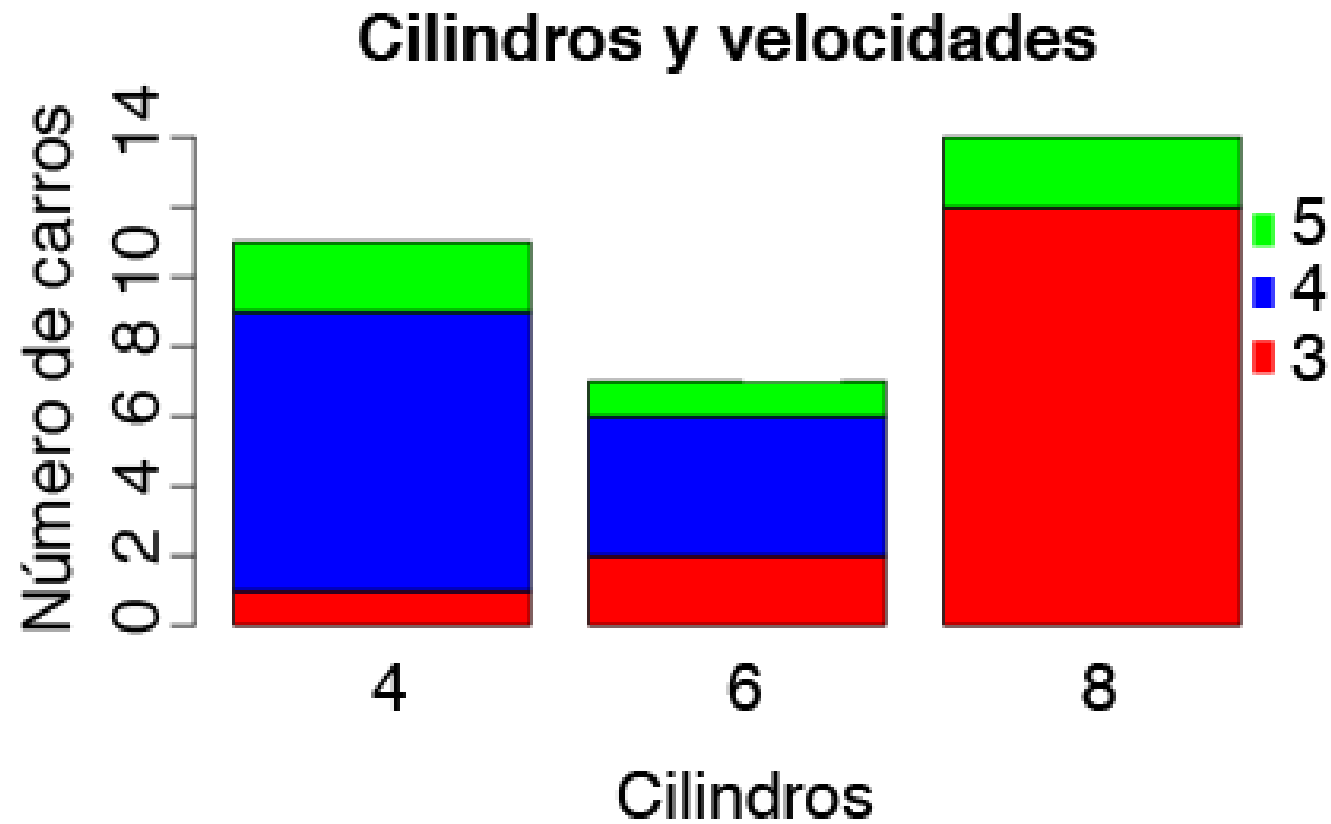
# Graficación: Barras

**Función R : barplot ()**



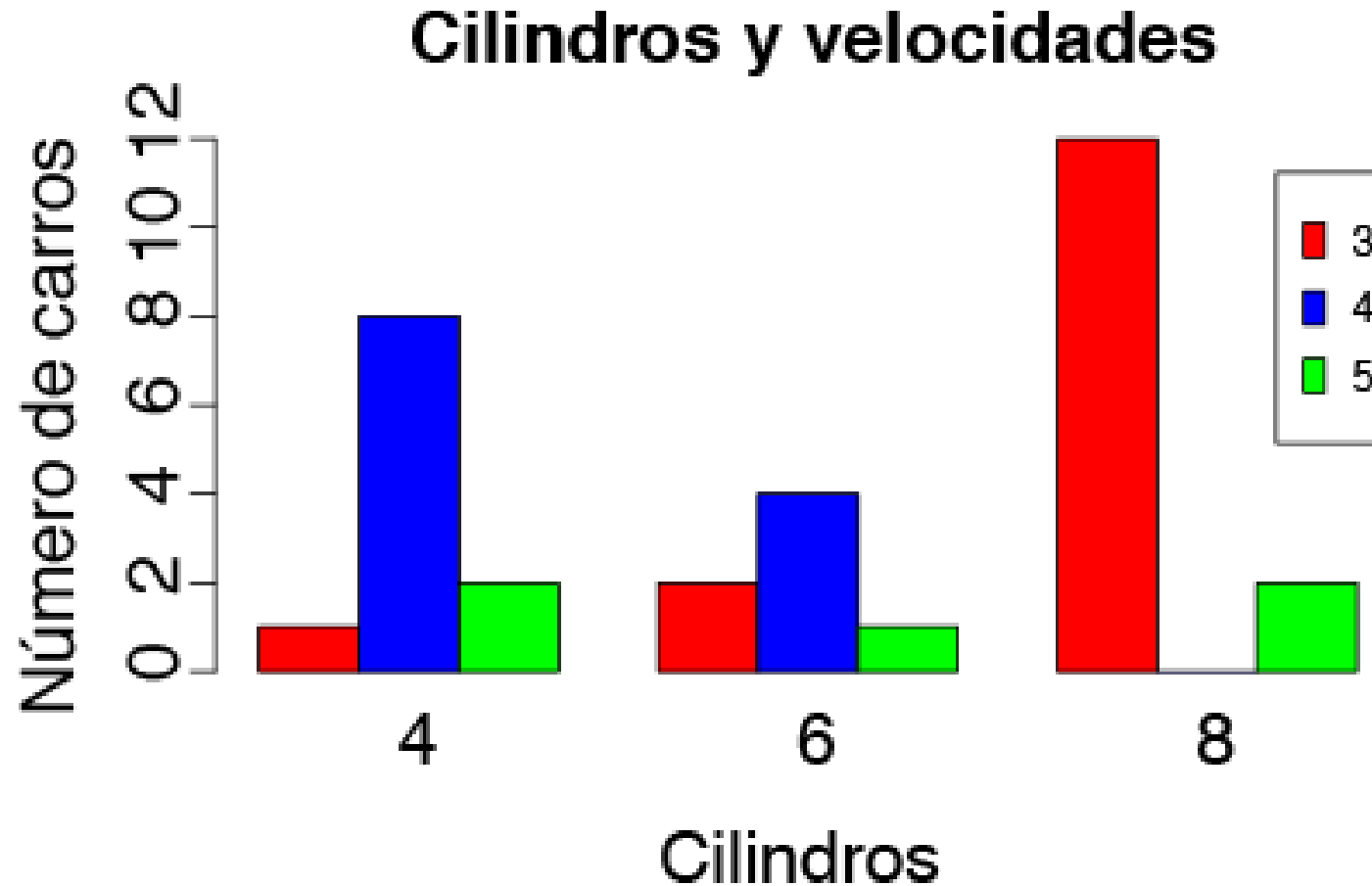
# Graficación: Barras apiladas

Función R : `barplot ()`



# Graficación: Barras laterales

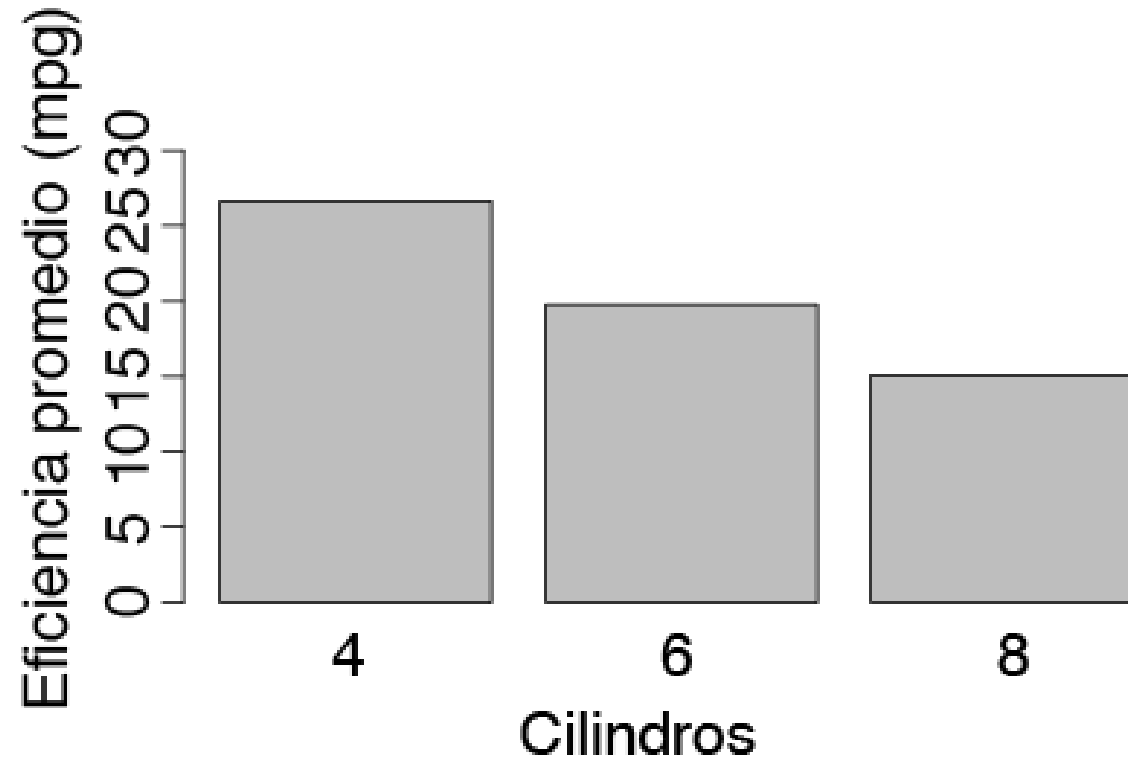
Función R : `barplot ()`



# Graficación: Agregar datos y obtener medias

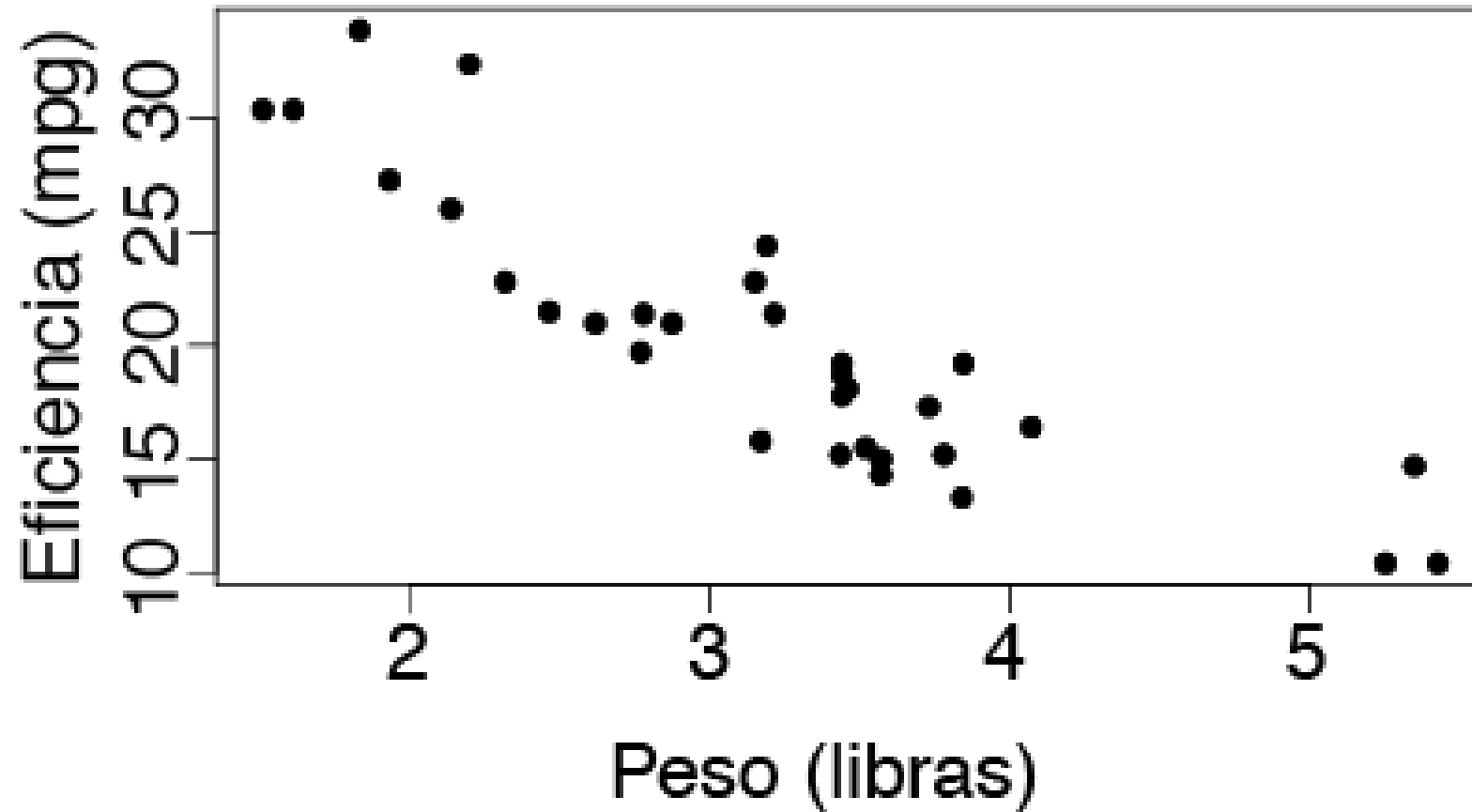
**Función R : barplot ()**

```
aggdatos <- aggregate(carros$eficiencia, by=list(carros$cilindros), FUN=mean,  
na.rm=TRUE)
```



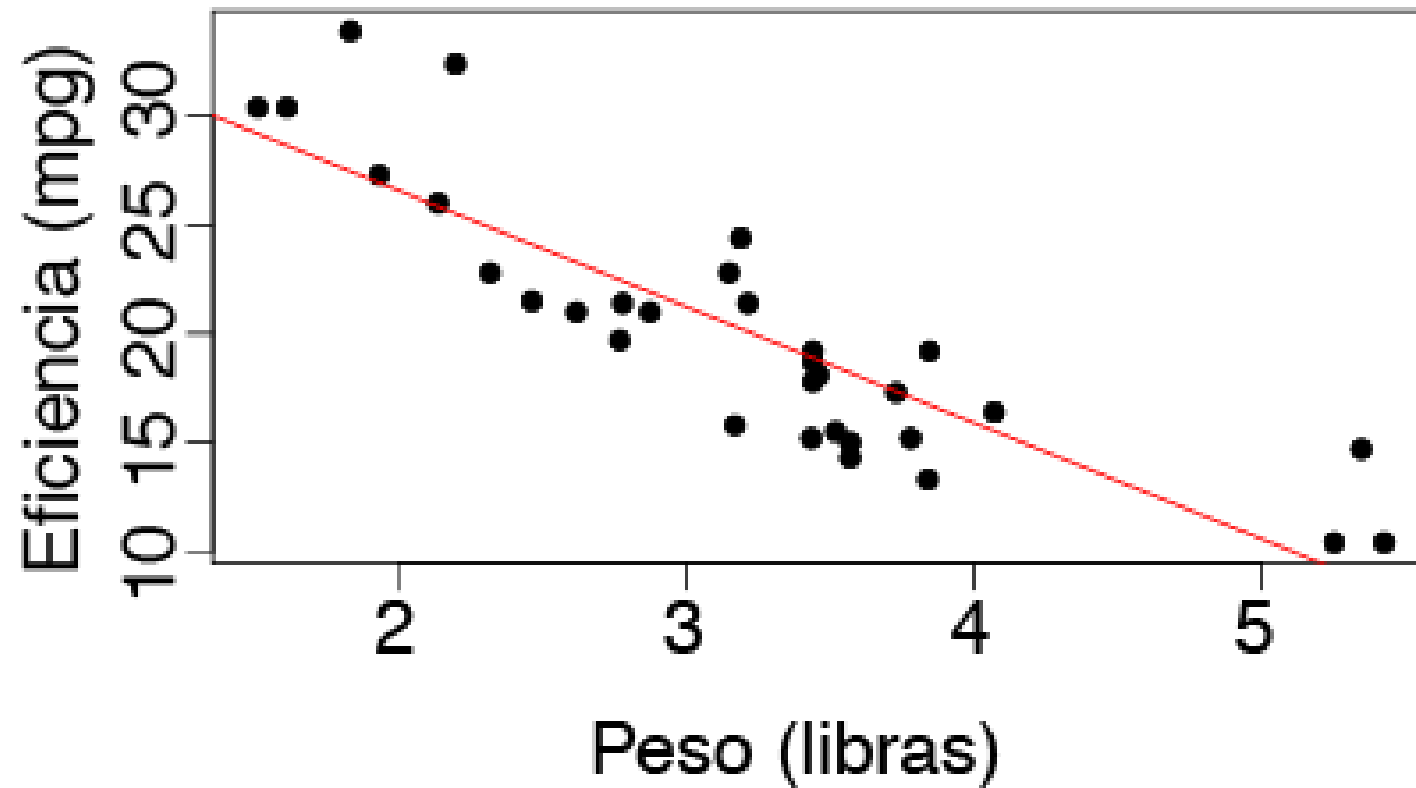
# Graficación: Dispersión

Función R : `plot ()`



# Graficación: Dispersión - añadir línea de ajuste

Función R : `plot ()`





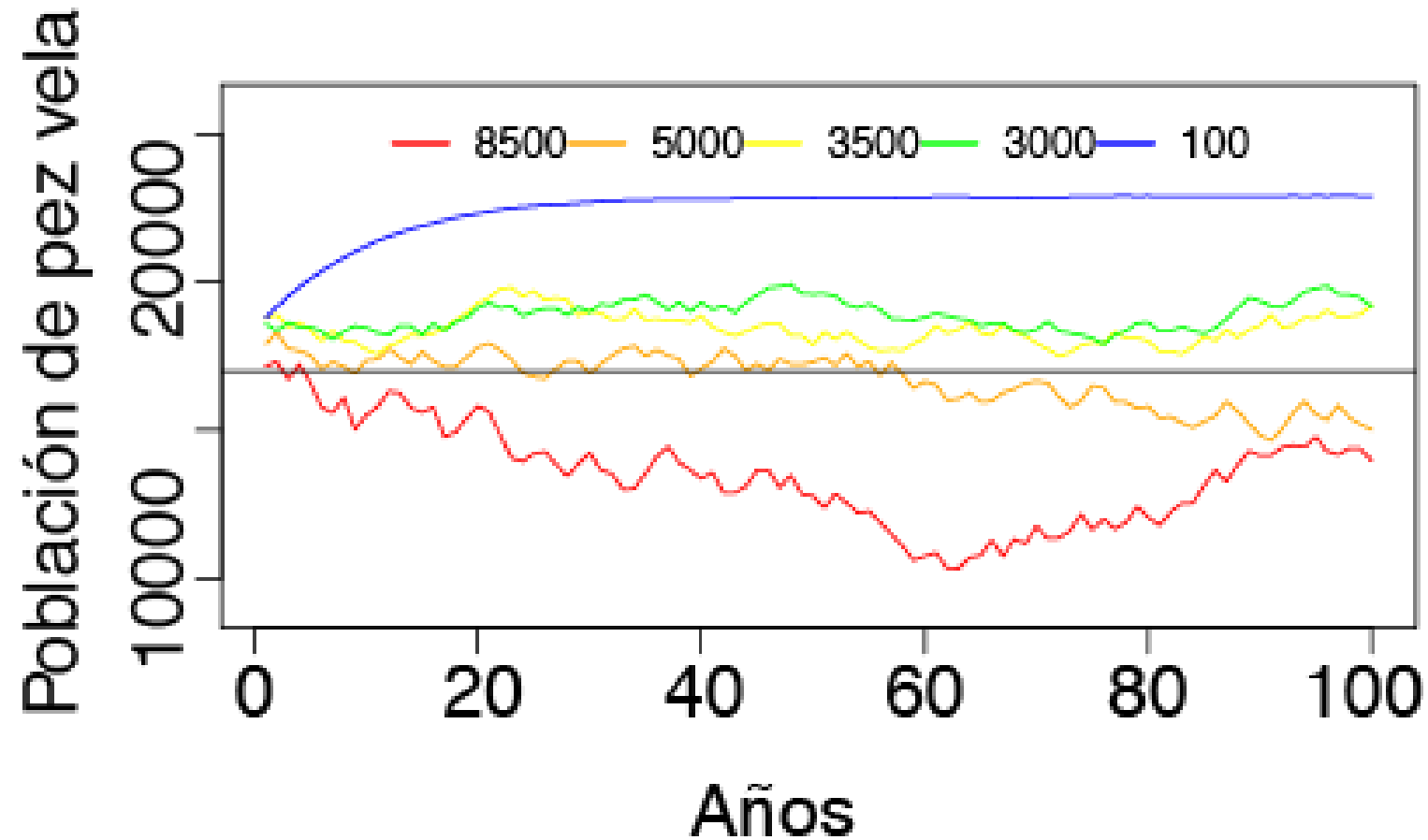
# Graficación: Líneas

**Función R : plot () y lines ()**

```
str(ModeloResultados)'data.frame':  100 obs. of  5 variables:  
 $ PoblacionMarlin8500: num  17165 17305 16722 17222 16613  
 $ PoblacionMarlin5000: num  17924 18290 17751 17687 17533  
 $ PoblacionMarlin3500: num  18795 18912 18490 18632 18393  
 $ PoblacionMarlin3000: num  18583 18366 18636 18489 18492  
 $ PoblacionMarlin100 : num  18797 19173 19515 19826 20113
```

# Graficación: Líneas

Función R : `plot ()` y `lines ()`

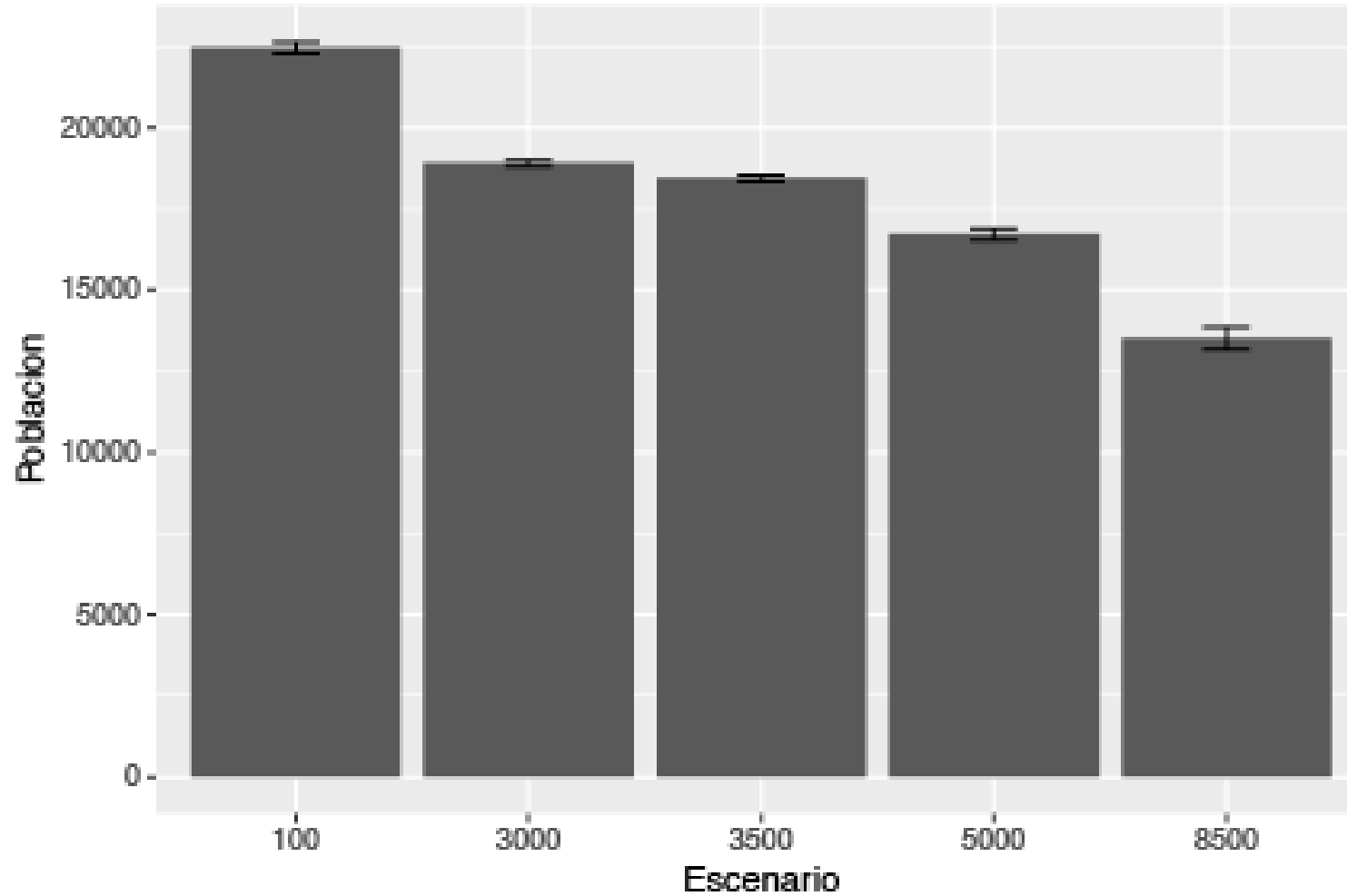


# Graficación: ggplot

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

```
str(Data)'data.frame':500 obs. of 2 variables:  
 $ Escenario: Factor w/ 5 levels  
 "100","3000","3500","5000","8500"  
 $ Poblacion: num 17165 17305 16722 17222 16613 ..
```

# Graficación: ggplot



# Ejercicio 5

1. Importar la base de datos Urea.csv
2. Hacer una grafica de dispersión con línea de ajuste de Progesterona ~ Urea, sin tomar en cuenta el nivel de dosis.
3. Obtener la media de la variable Urea para cada nivel de dosis y hacer una grafica de barras.
4. Hacer una gráfica de barras en ggplot de la media de progesterona (media  $\pm 95\%$  intervalo de confianza) para cada nivel de dosis.
5. Hacer una grafica de dispersión Progesterona ~ Urea, tomando en cuenta el nivel de dosis- investigar en línea.