Unidad 3: Graficos

Olga Andrea Hernández Miranda

2024-06-14

Para graficar en R es necesario utilizar comandos para decirle a la maquina que hacer en vez de utilizar clicks como en excel. Comando = hacer un click en elgún menu. Ventajas: -Obtener plantillas para reproducr un mismo grafico. -Multiples opciones gráficas. Desventaja: -Curva de aprendizaje mayor.

Para representar graficamente los datos en R, existen dos tipos de funciones. Alto nivel: Generan graficos completos. plot() (gráficos de nubes de puntos, entre otros). hist() (histogramas). barplot() (diagramas de barras). boxplot() (diagramas de caja y bigote). pie() (diagrama de sectores). pers() (superficies en 3D). xlim, ylim (controlan, respectivamente la extensión de los ejes X e Y). xlim=c(0,10) (indica que el eje X se extiende de 0 a 10). ylim=c(-5,5) (indica que el eje Y va de -5 a 5). xlab e ylab (especifican las etiquetas para los ejes X e Y). main (indica el título del gráfico). Bajo nivel: Añaden elementos a un grafico existente. lines() (Permite añadir lineas (uniendo puntos concretos) a una gráfica ya existente). abline() (Añade lineas horizontales, verticales u oblicuas, indicando pendiente y ordenada). points() (Permite añadir puntos). legend() (Permite añadir una leyenda). text() (Añade texto en las posiciones que se indiquen). grid() (Añade una malla de fondo). title() (permite añadir un título o subtítulo). Para iniciar vamos a crear un data frame manualmente de la expresión genetica de cuatro genes en la etapa del desarrollo uno.

```
DatosE <- data.frame(
  gen = c("ARF", "TIR", "CO", "GA"),
  ed = c(2,10,10,15)
)
```

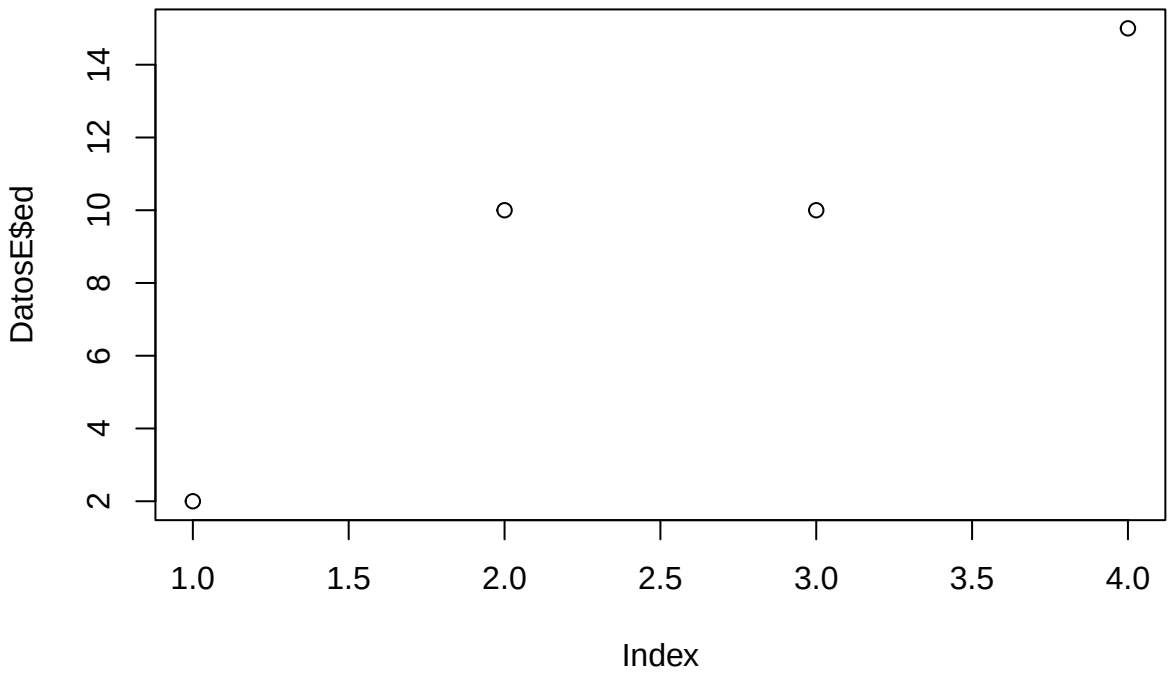
#Vemos los datos

```
DatosE
```

```
##   gen ed
## 1 ARF  2
## 2 TIR 10
## 3  CO 10
## 4  GA 15
```

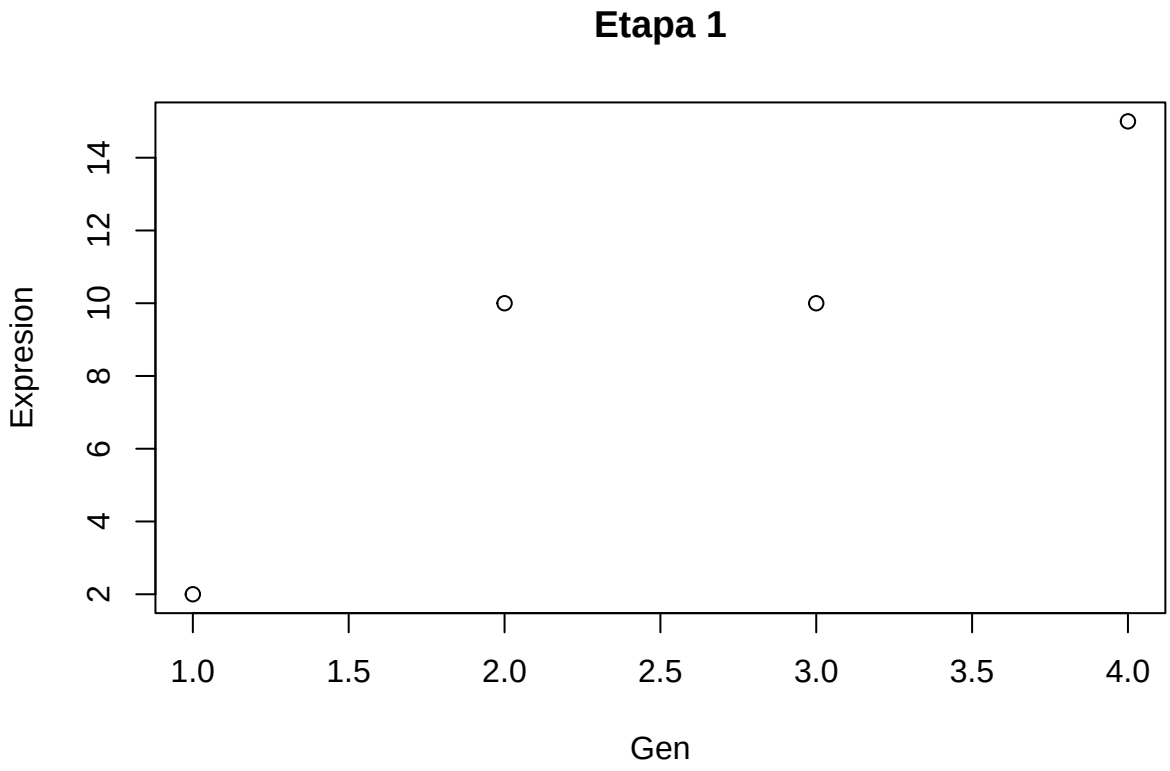
Graficamos los datos con la función “plot()”

```
plot(DatosE$ed)
```



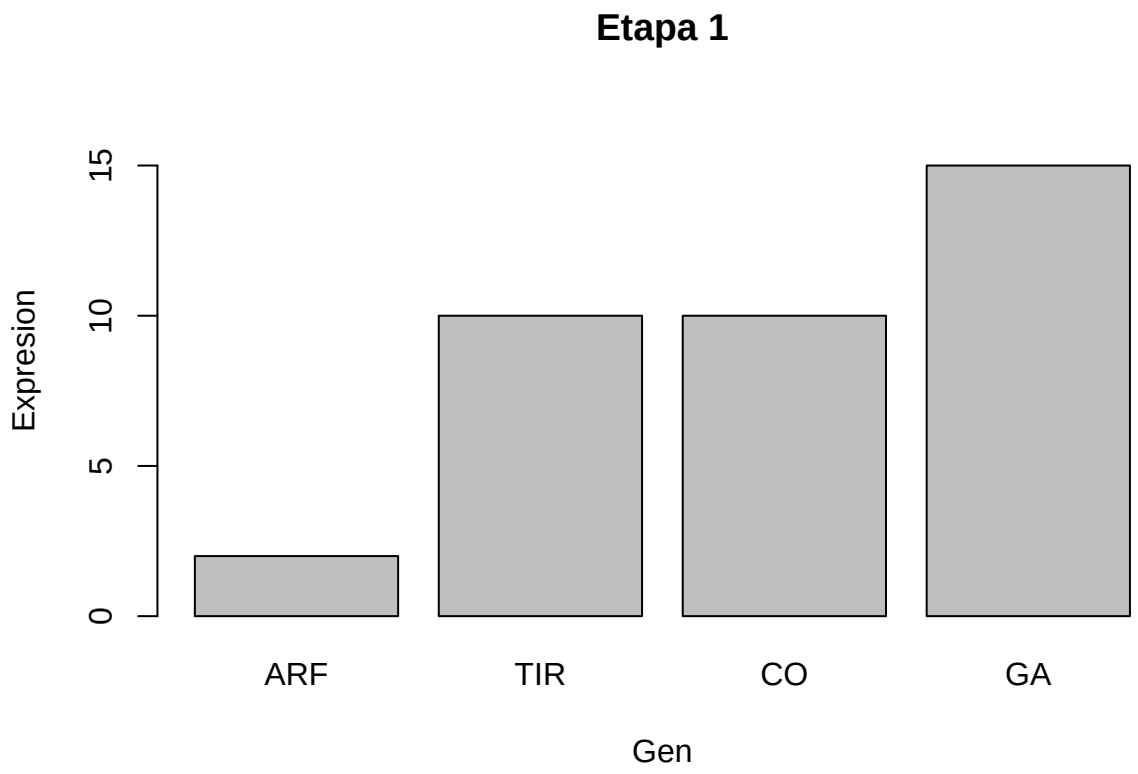
Vamos a agregar los titulos de los ejes

```
plot(DatosE$ed,
     main = "Etapa 1", #titulo
     xlab = "Gen", #titulo eje x
     ylab = "Expresion") #titulo eje y
```



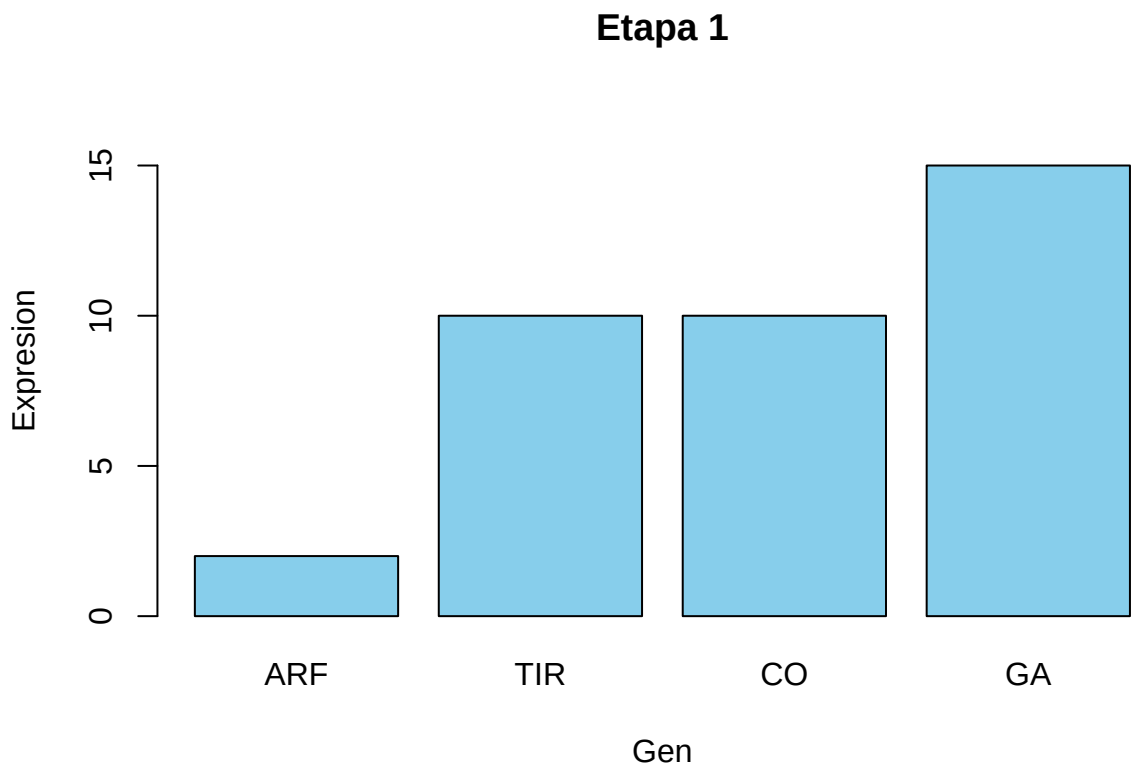
Generamos graficas de barras con la función “barplot()”.

```
barplot(DatosE$ed,
        names.arg = DatosE$gen, #agregamos el nombre de los genes
        main = "Etapa 1",
        xlab = "Gen",
        ylab = "Expresion",
        ylim = c(0, max(DatosE$ed) + 2) #el eje y va de 0 al maximo de la expresion
)
```



Agregamos color

```
barplot(DatosE$ed,
        names.arg = DatosE$gen,
        main = "Etapa 1",
        xlab = "Gen",
        ylab = "Expresion",
        col = "skyblue",#color
        border = "black", #color del borde
        ylim = c(0, max(DatosE$ed) + 2) #Limite del eje y
)
```



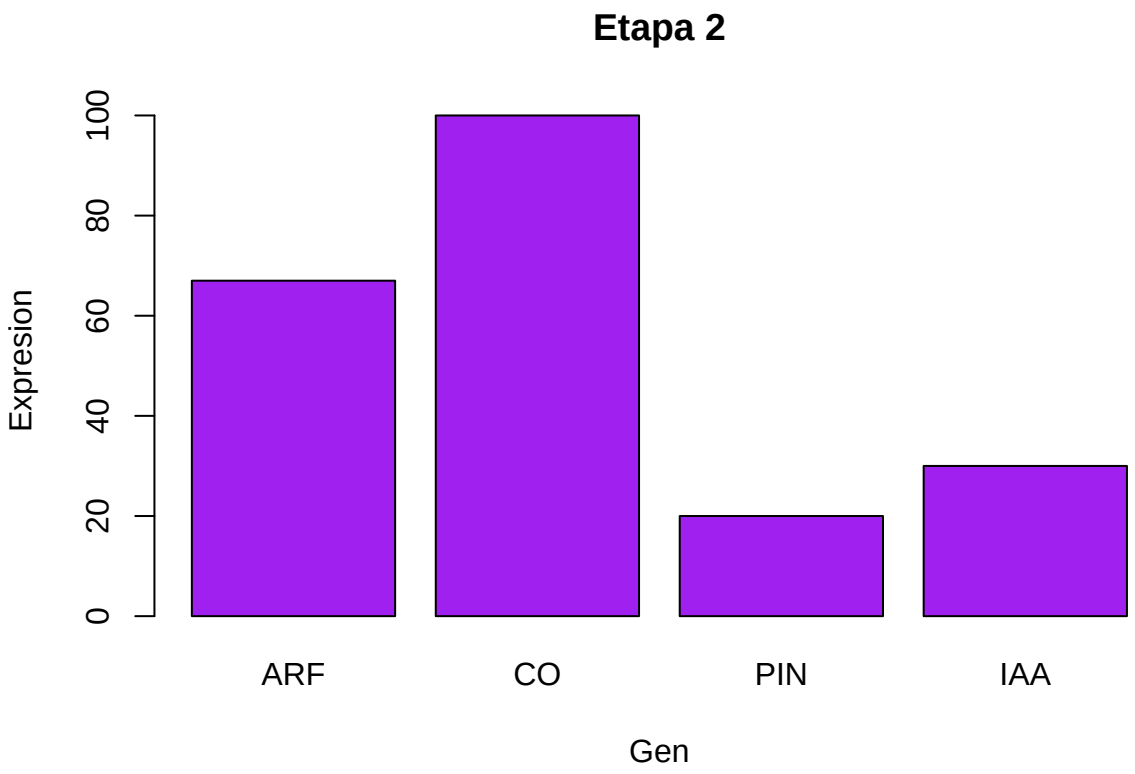
Cargamos los datos de la etapa dos

```
directorio <- "C:/Users/andii/OneDrive/Documents/02Fun-R-transcript/data" #indicamos el directorio de t
setwd(directorio)

DatosE2 <- read.table("U3_1.csv", sep = ",", header = T) #Se lee el arcivo
DatosE2
```

```
##   Gen  ED
## 1 ARF  67
## 2  CO 100
## 3 PIN  20
## 4 IAA  30
```

Graficamos

```
barplot(DatosE2$ED,
        names.arg = DatosE2$Gen,
        main = "Etapa 2",
        xlab = "Gen",
        ylab = "Expresion",
        col = "purple",
        border = "black",
        ylim = c(0, max(DatosE2$ED) + 2)
)
```



Vamos abrir una base de datos de expresión genética para generar un histograma. Guardamos los datos en el objeto E

```
E <- read.table("U3_2.csv", sep = ",", header = T)
head(E)
```

```
##   TAR ARF  CO  GA Etapas
## 1 5.1 3.5 1.4 0.2 Etapa1
## 2 4.9 3.0 1.4 0.2 Etapa1
## 3 4.7 3.2 1.3 0.2 Etapa1
## 4 4.6 3.1 1.5 0.2 Etapa1
## 5 5.0 3.6 1.4 0.2 Etapa1
## 6 5.4 3.9 1.7 0.4 Etapa1
```

Generamos un histograma con la función “hist()”

```
hist(E$ARF,
     main="Histograma de datos de expresion de ARF",
     xlab="Expresion", ylab="Frecuencia",
     col="yellow"
     )
```

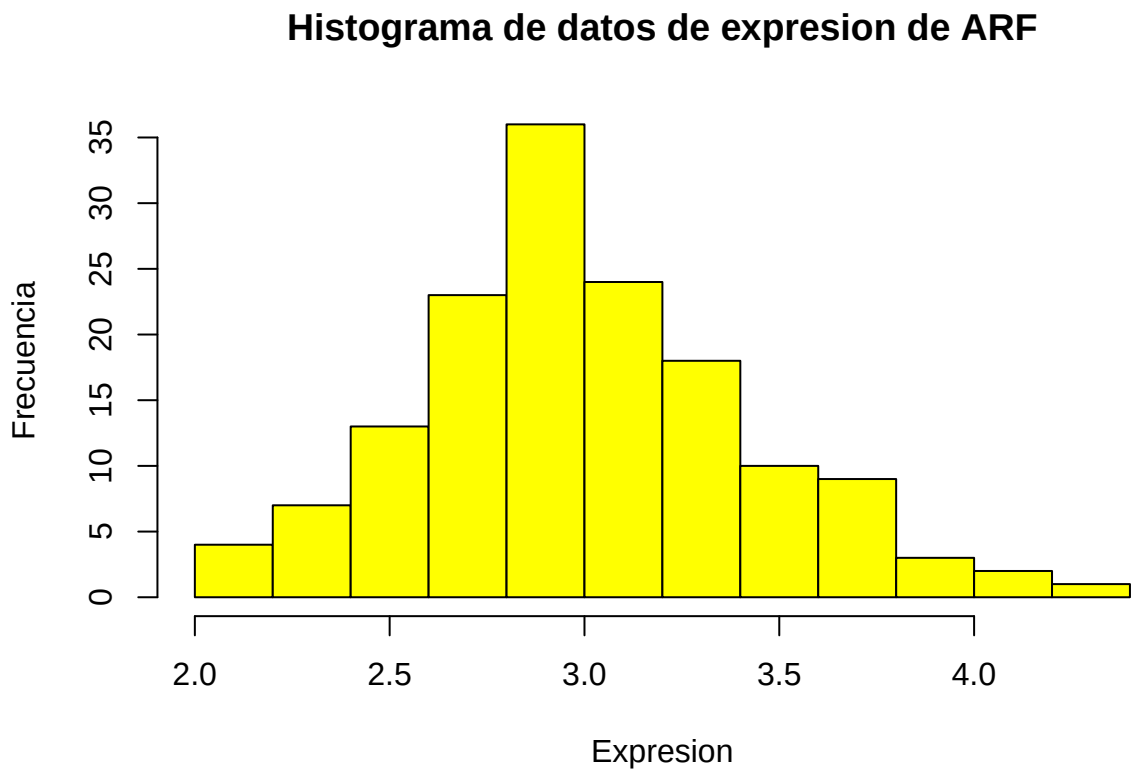


Diagrama de caja y bigote Q1, el Primer Cuartil es el valor mayor al 25% de los valores de la distribución.

Q2, el Segundo Cuartil es la mediana de la distribución, es el valor de la variable que ocupa el lugar central en un conjunto de datos ordenados.

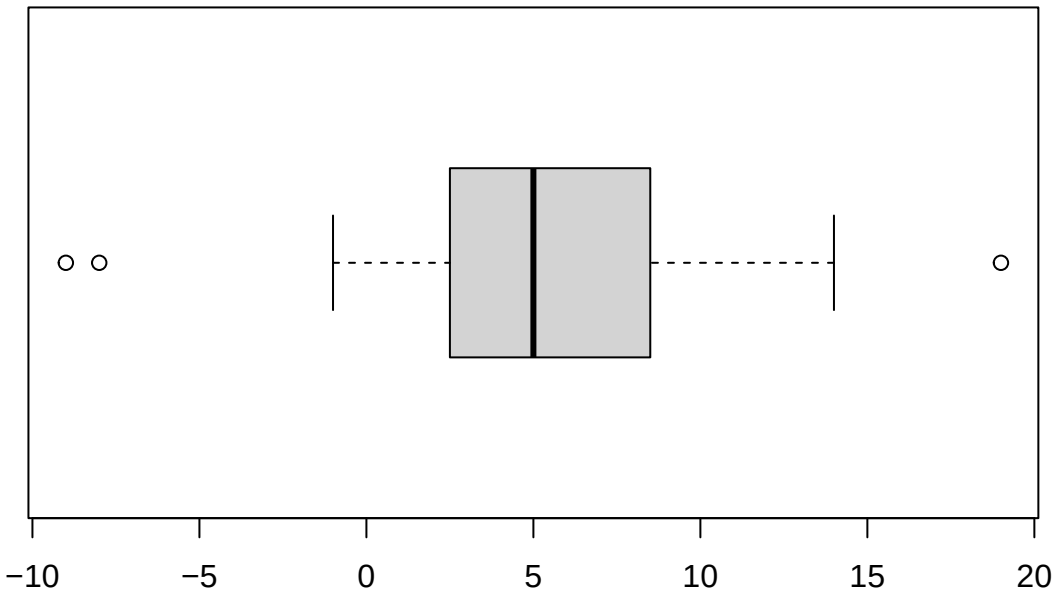
Q3, el Tercer Cuartil, el 25% de los datos sobrepasan este cuartil

Generamos un vector para grafcarlo con la función "boxplot()".

```
TAR <- c(8, 5, 14, -9, 19, 12, 3, 9, 7, 4,
       4, 6, 8, 12, -8, 2, 0, -1, 5, 3)
```

Graficamos el vector

```
boxplot(TAR,
        horizontal = T)
```



Utilizamos los datos del archivo U3_2.csv para crear diagramas de cajas y bigotes.

Si el conjunto de datos tiene una variable categórica que contiene grupos, puedes crear un diagrama de caja especificando la fórmula (variable_continua ~ variable_categorica).

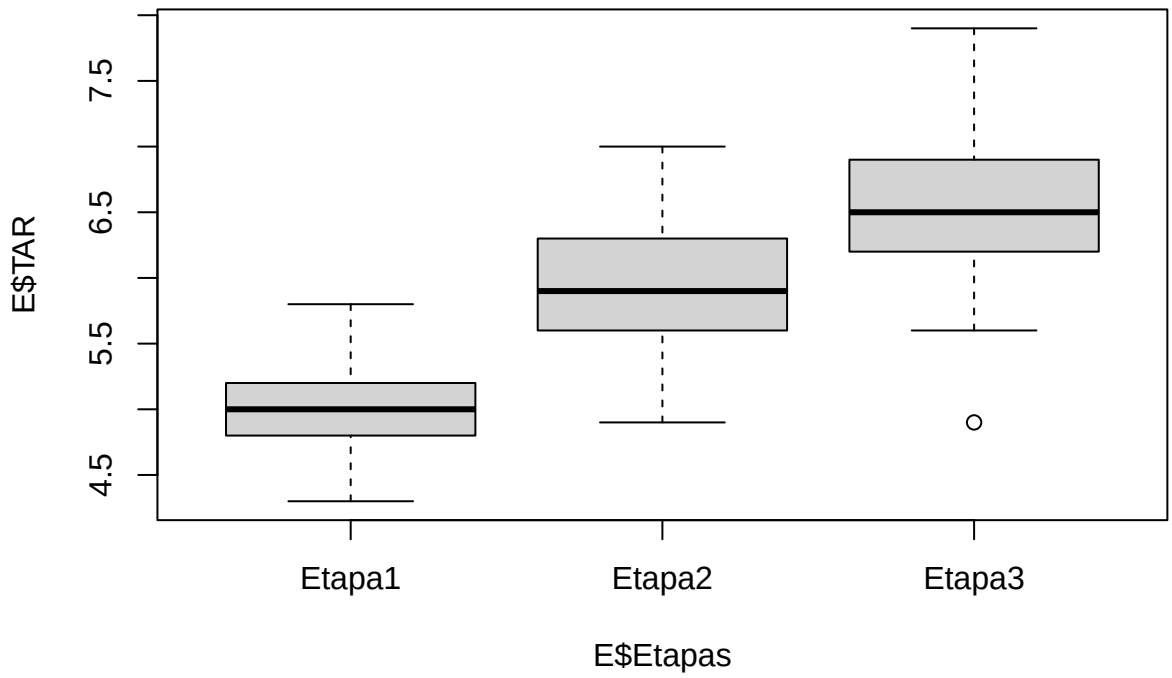
Vemos los datos

```
head(E)
```

```
##   TAR ARF  CO  GA Etapas
## 1 5.1 3.5 1.4 0.2 Etapa1
## 2 4.9 3.0 1.4 0.2 Etapa1
## 3 4.7 3.2 1.3 0.2 Etapa1
## 4 4.6 3.1 1.5 0.2 Etapa1
## 5 5.0 3.6 1.4 0.2 Etapa1
## 6 5.4 3.9 1.7 0.4 Etapa1
```

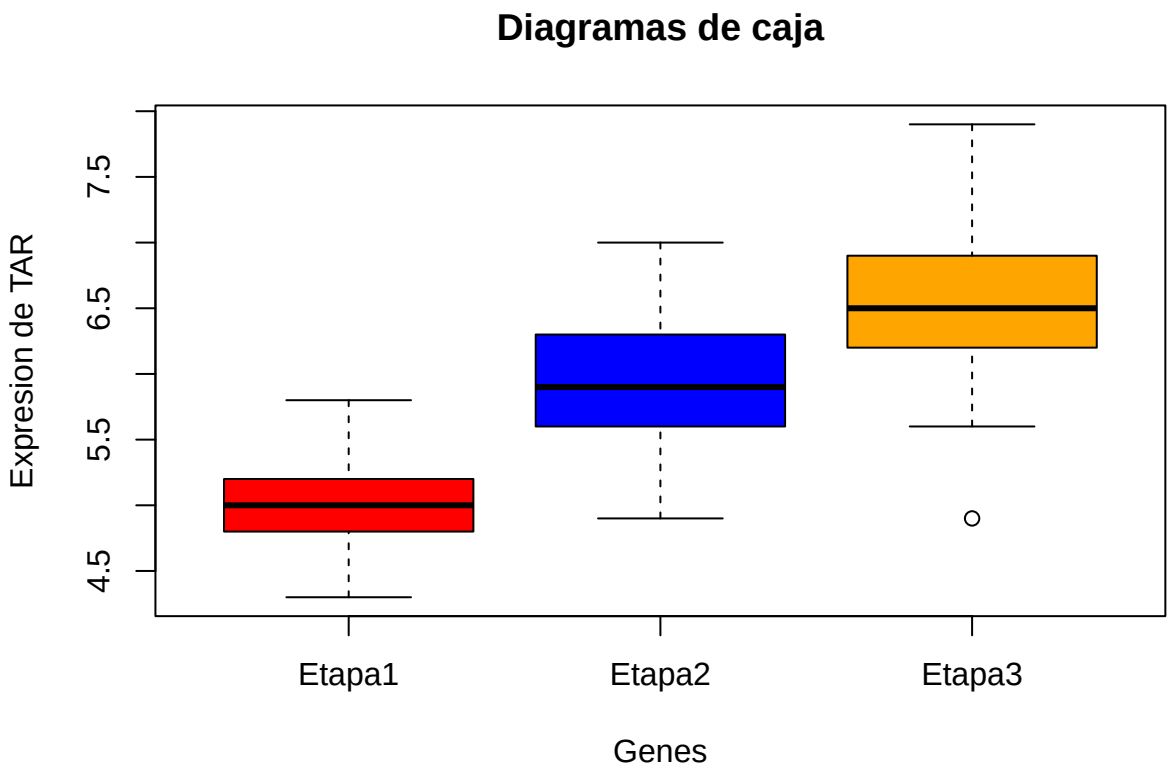
Grficamos los datos con la funcion “boxplot()”.

```
boxplot(E$TAR~ E$Etapas) #columnas
```



Agregamos color y titulos

```
boxplot(E$TAR~ E$Etapas,#agregamos todas las columnas que vamos a graficar
        main="Diagramas de caja",#titulo
        xlab="Genes",#titulo x
        ylab="Expresion de TAR",
        col=c("red", "blue", "orange")#colores
        )
```



Graficos en ggplot2

Vamos a utilizar la librería ggplot2 para graficar los datos de una manera más estetica; sin embargo, tiene una sintaxis especifica dividida en tres elementos:

La información (data) a utilizar. La estética (aesthetics) o la definición de los ejes donde se posicionarán los datos a visualizar. La geometría (geometry) o los elementos visuales que se posicionarán en la gráfica.

Además, los gráficos se construyen en base a una serie de capas de información que superpuestas, configuran el resultado final.

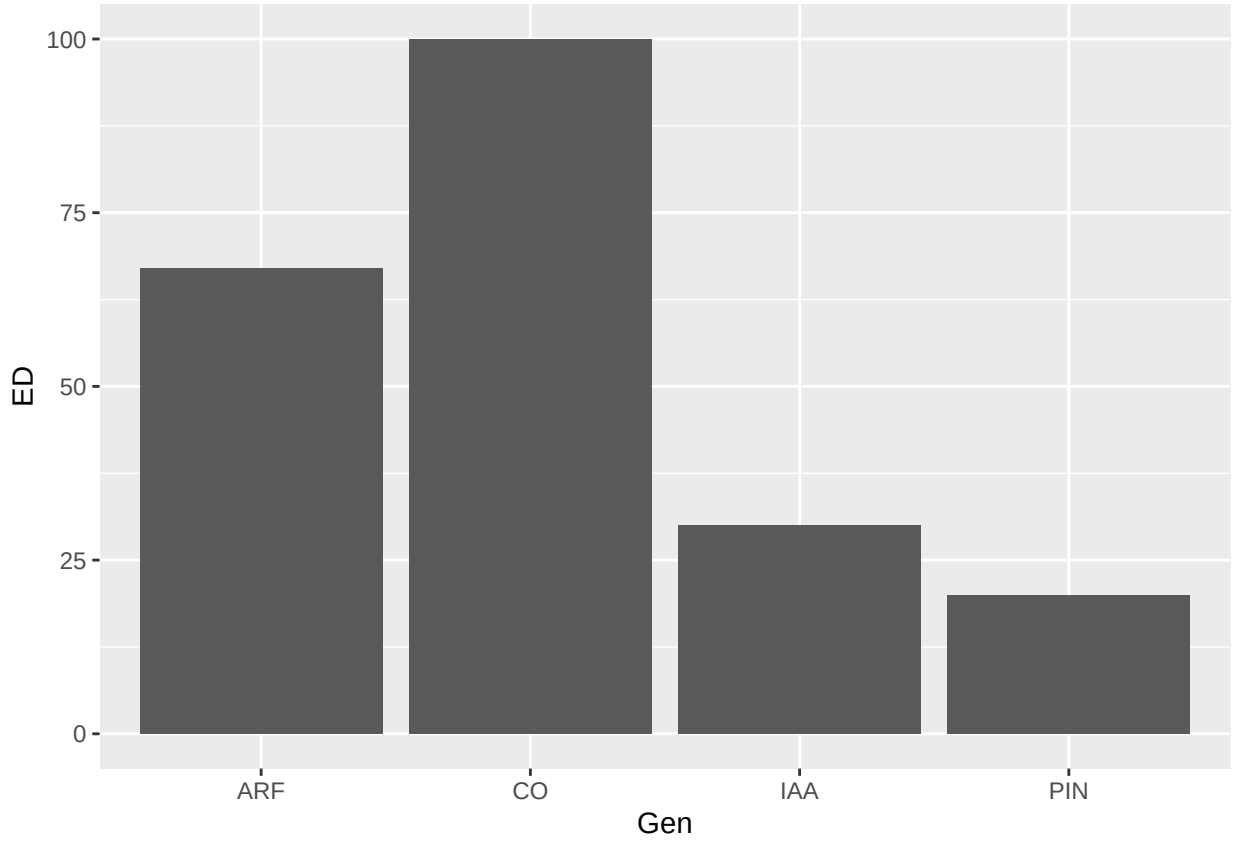
Vamos a iniciar instalando la librería "ggplot2" y cargandola para después visualizar los datos del archivo Dia3_1.

```
# install.packages("ggplot2")
library(ggplot2)
head(DatosE2)
```

```
##   Gen  ED
## 1 ARF  67
## 2  CO 100
## 3 PIN  20
## 4 IAA  30
```

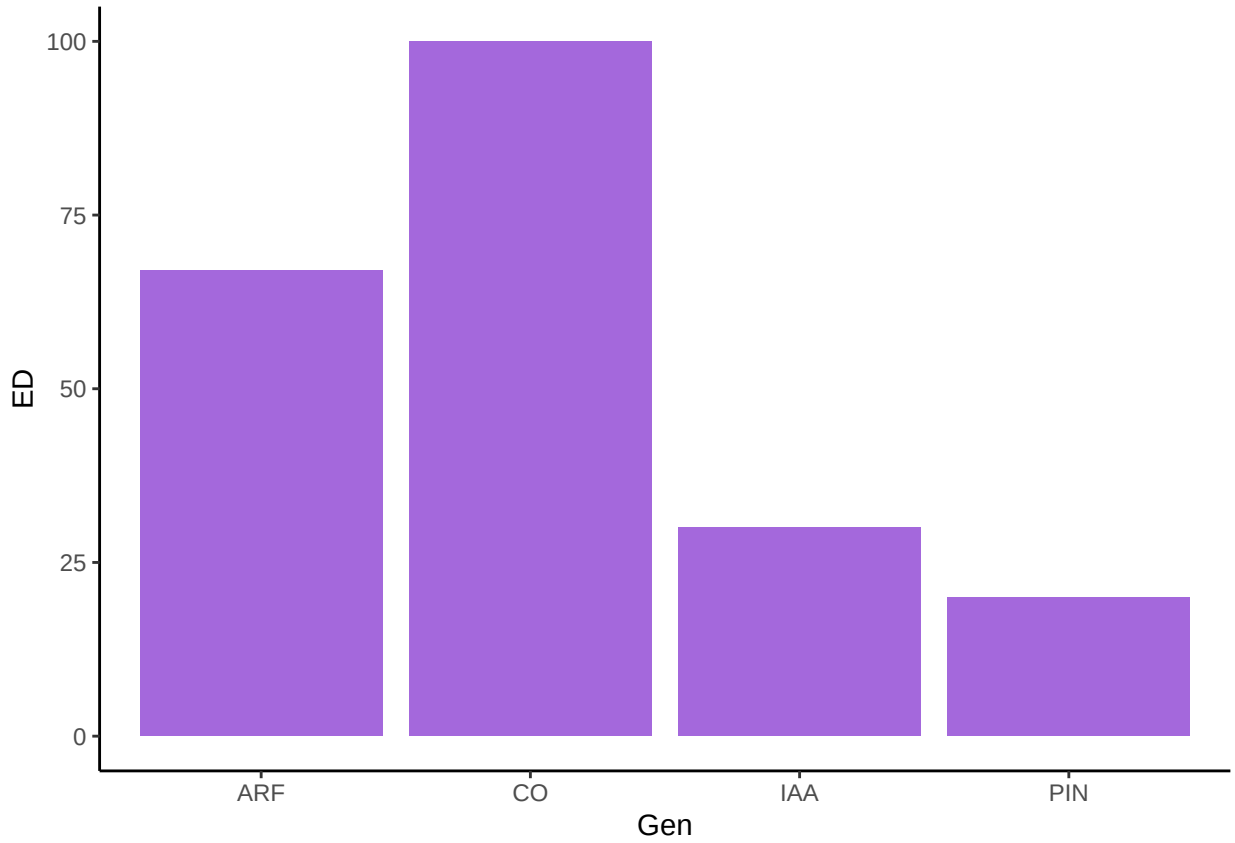
Gráfico de barras

```r
ggplot(DatosE2,aes(x = Gen, y = ED)) + #Indicamos los datos y ejes
  geom_bar(stat = 'identity') #Indicamos la geometria (como se posicionará)
```



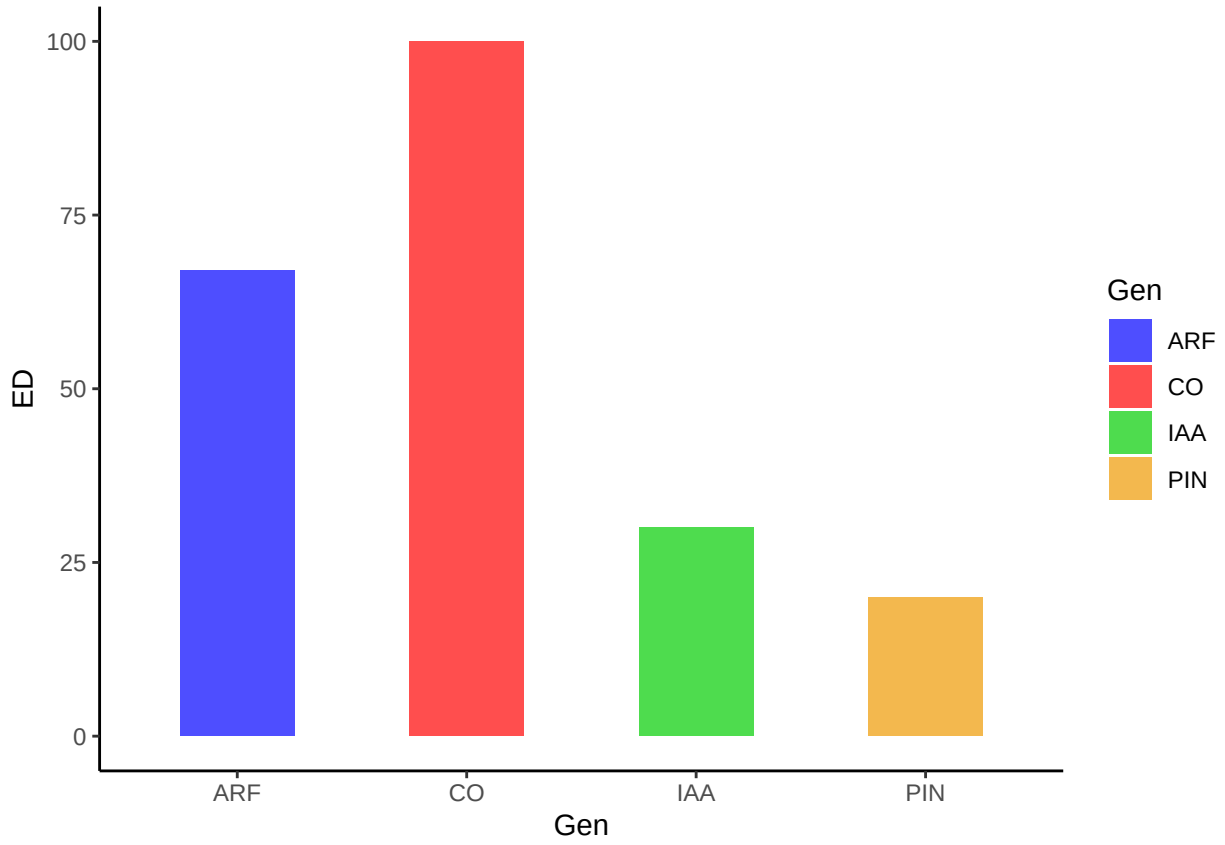
Cambiamos las barras a color morado transparente y tema clasico

```r
ggplot(DatosE2, aes(x = Gen, y = ED)) + # Indicamos los datos y ejes
  geom_bar(stat = 'identity', fill = "purple3", alpha = 0.7) + # Indicamos la geometría y el color de l
  theme_classic() # Aplicamos el tema clásico
```



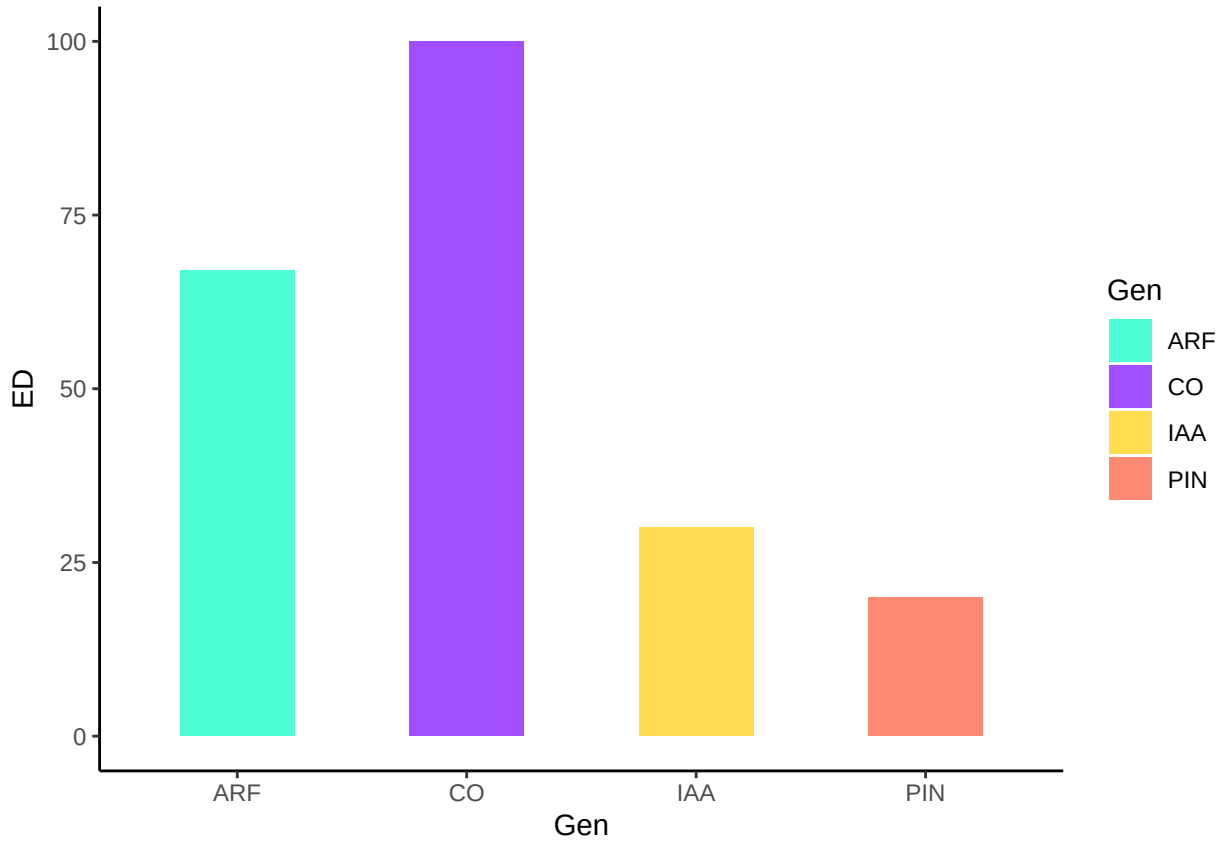
Para personalizar paletas de colores

```
colores <- c("blue", "red", "green3", "orange2")

ggplot(DatosE2, aes(x = Gen, y = ED, fill = Gen)) +
  geom_bar(stat = 'identity', alpha = 0.7, width = 0.5) + #geometría, transparencia y ancho de las barr
  theme_classic() + # Aplicamos el tema clásico
  scale_fill_manual(values = colores)  # Asignamos los colores manualmente
```



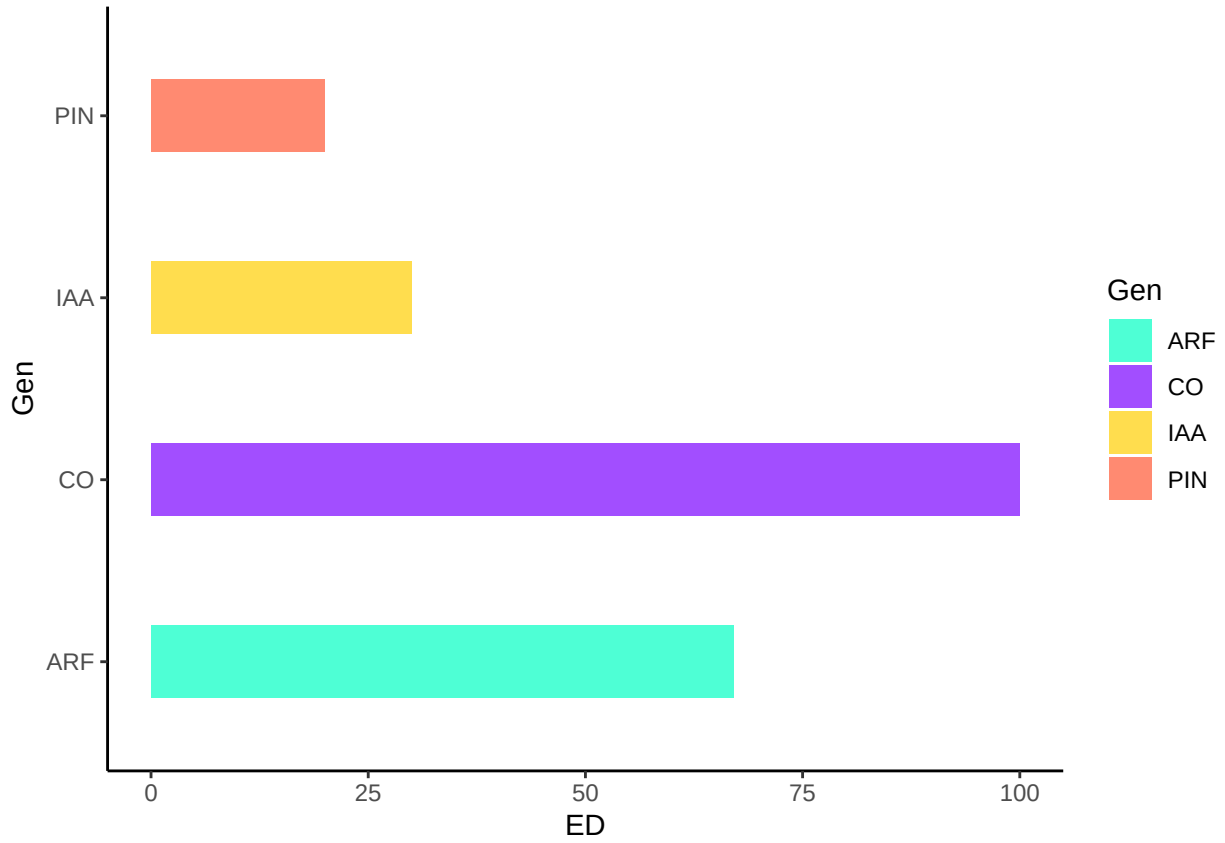
Paleta de colores con código

```
colores2 <- c("#00FFC3", "#7A00FF", "#FFCF00", "#FF5733")

ggplot(DatosE2, aes(x = Gen, y = ED, fill = Gen)) +
  geom_bar(stat = 'identity', alpha = 0.7, width = 0.5) +
  theme_classic() +
  scale_fill_manual(values = colores2)
```



Para colocar las barras horizontalmente sólo movemos el orden de los ejes x y y

```
ggplot(DatosE2, aes(x = ED, y = Gen, fill = Gen)) +
  geom_bar(stat = 'identity', alpha = 0.7, width = 0.4) +
  theme_classic() +
  scale_fill_manual(values = colores2)
```

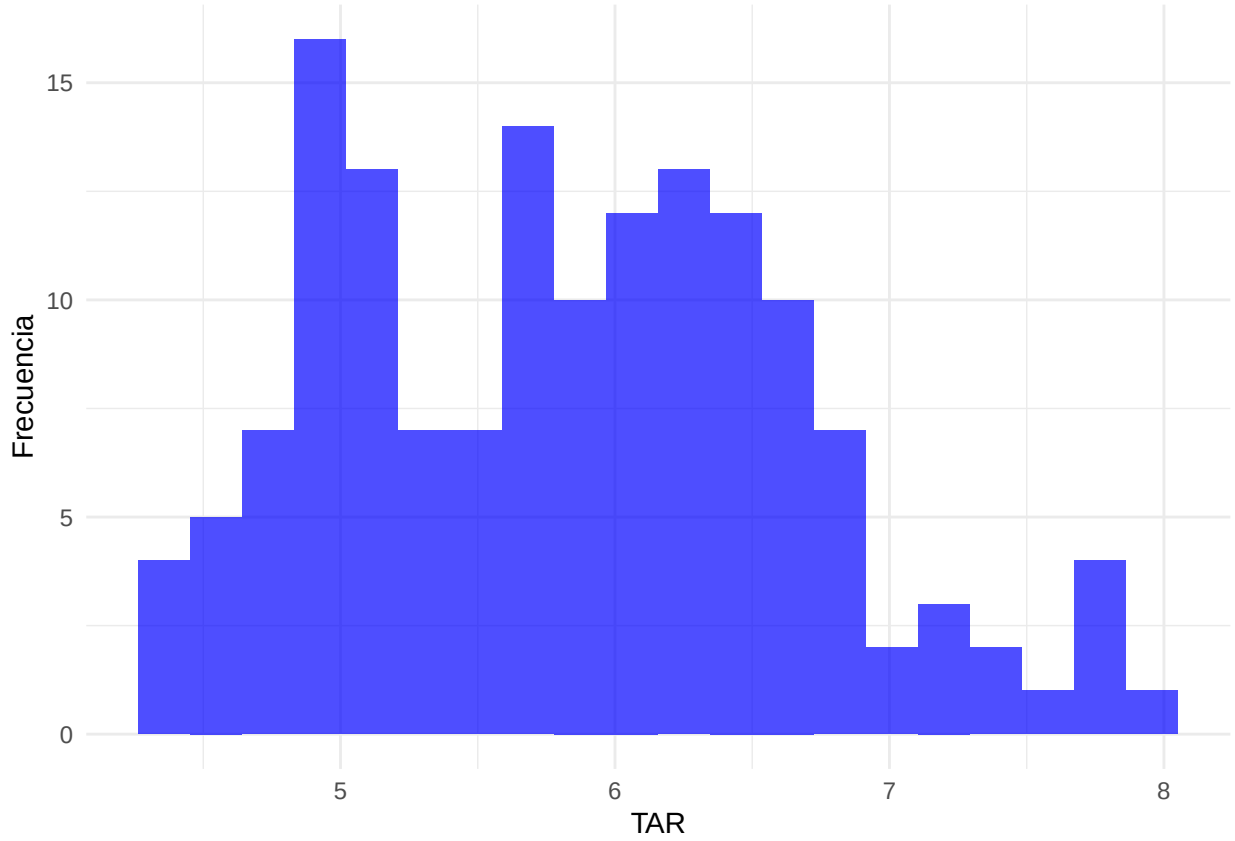


Para crear un histograma de la frecuencia de espresión que tiene el gen TAR. Primero vemos los datos guardados en la variable (objeto E) y graficamos la columna TAR.

```
head(E)
```

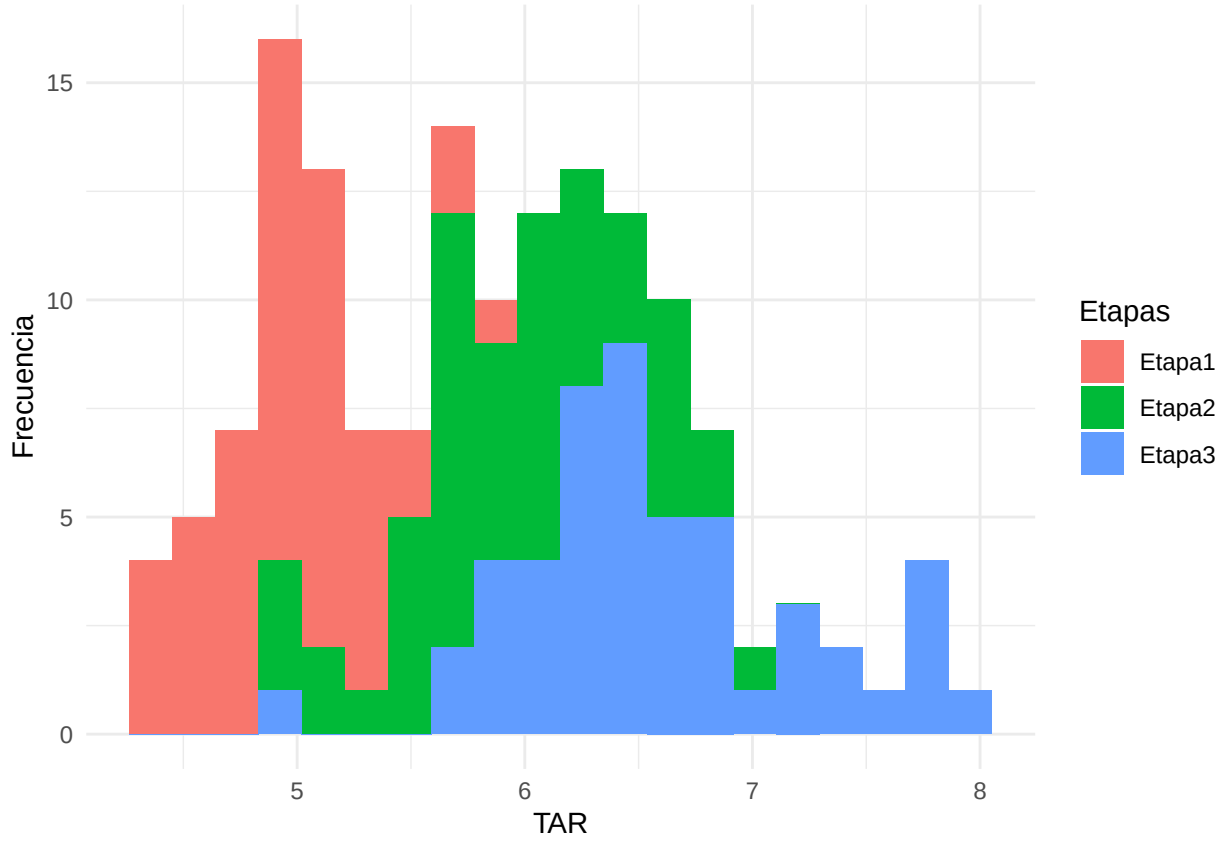
```
##   TAR ARF  CO  GA Etapas
## 1 5.1 3.5 1.4 0.2 Etapa1
## 2 4.9 3.0 1.4 0.2 Etapa1
## 3 4.7 3.2 1.3 0.2 Etapa1
## 4 4.6 3.1 1.5 0.2 Etapa1
## 5 5.0 3.6 1.4 0.2 Etapa1
## 6 5.4 3.9 1.7 0.4 Etapa1
```

```
ggplot(E, aes(x = TAR)) + #variable (objeto E), columna TAR
  geom_histogram(fill = "blue", bins = 20, alpha = 0.7) +  #geometria histograma, tamaño barra
  labs(x = "TAR", y = "Frecuencia") +
  theme_minimal()
```



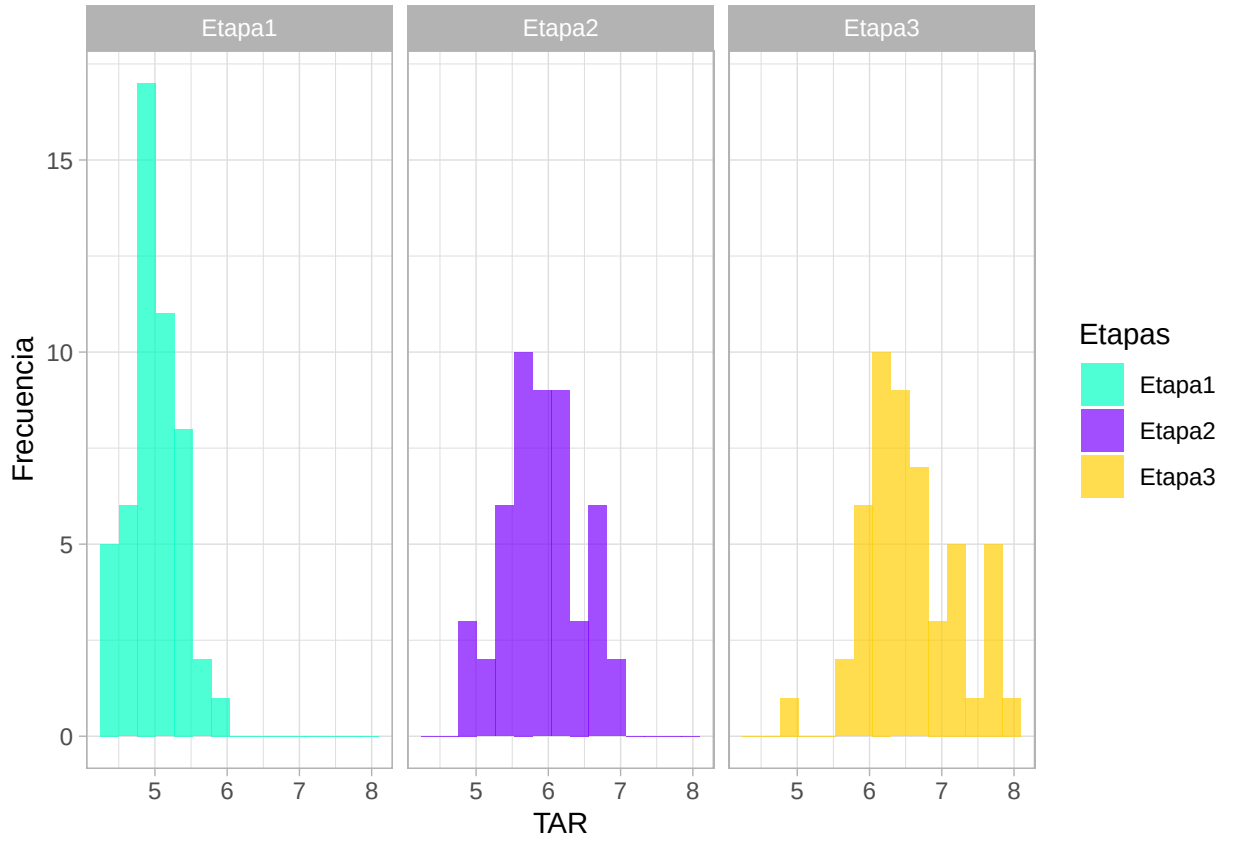
Difereciar datos que pertenecen a cada etapa del desarrollo

```
#Difereciar datos que pertenecen a cada etapa del desarrollo
ggplot(E, aes(x = TAR, fill = Etapas)) + #Separar datos por etapa
  geom_histogram(bins = 20) +
  labs(x = "TAR", y = "Frecuencia") +
  theme_minimal ()
```



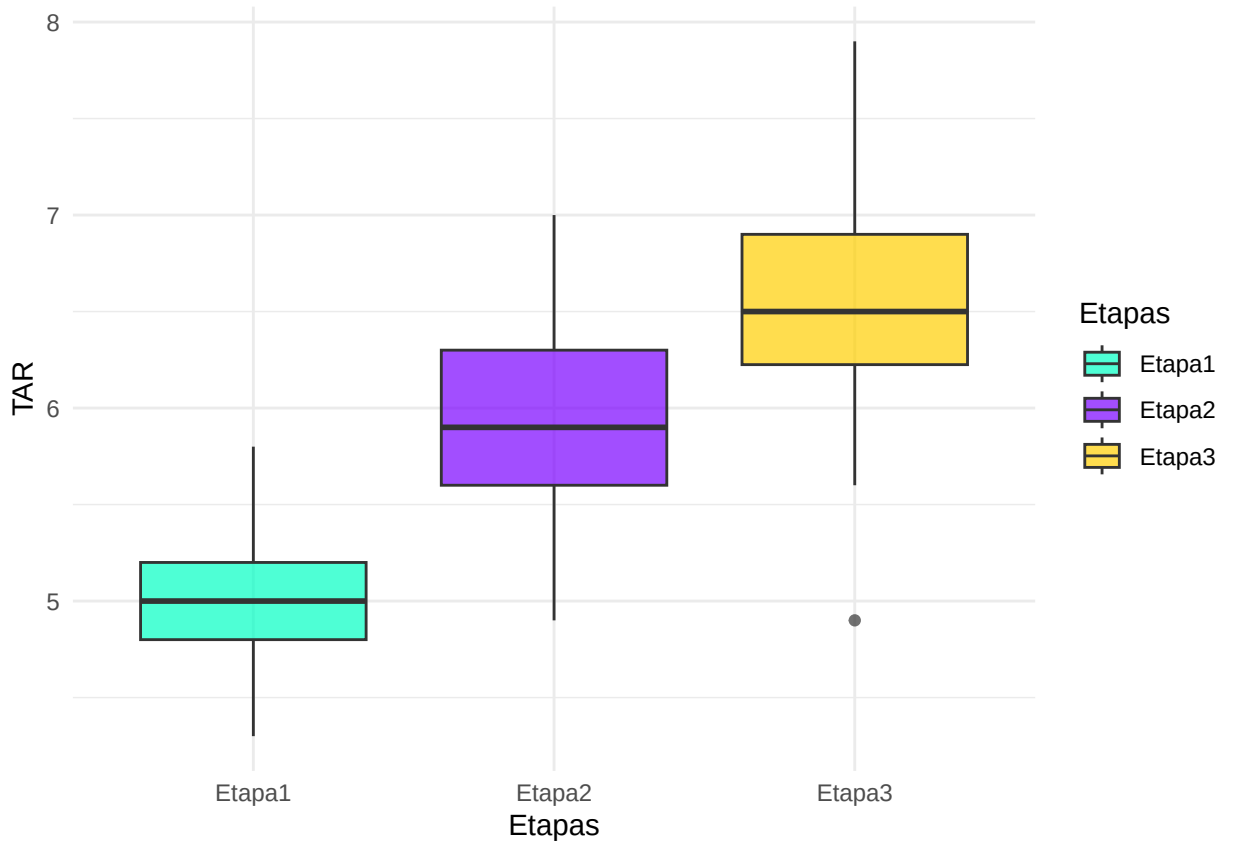
Graficar por separado los datos de cada etapa

```
ggplot(E, aes(x = TAR, fill = Etapas)) +
  geom_histogram(bins = 15, alpha = 0.7) +
  labs(x = "TAR", y = "Frecuencia") +
  facet_wrap(~ Etapas) + #Graficar Etapas por separado
  scale_fill_manual(values = c("#00FFC3", "#7A00FF", "#FFCF00")) +
  theme_light()
```



Boxplot

```
ggplot(data = E, aes(x = Etapas, y = TAR, fill = Etapas)) +
  geom_boxplot(alpha = 0.7) +  # Añadir cajas con transparencia
  scale_fill_manual(values = c("#00FFC3", "#7A00FF", "#FFCF00")) +
  theme_minimal()
```



Aplicación:

Vamos a utilizar los datos de Expresion del gen TAR y optener el promedio y desviación estandar de la expresión genética por etapa

Instalamos la librería “dplyer”si se requiere y la cargamos

```
#install.packages("dplyr")
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
```

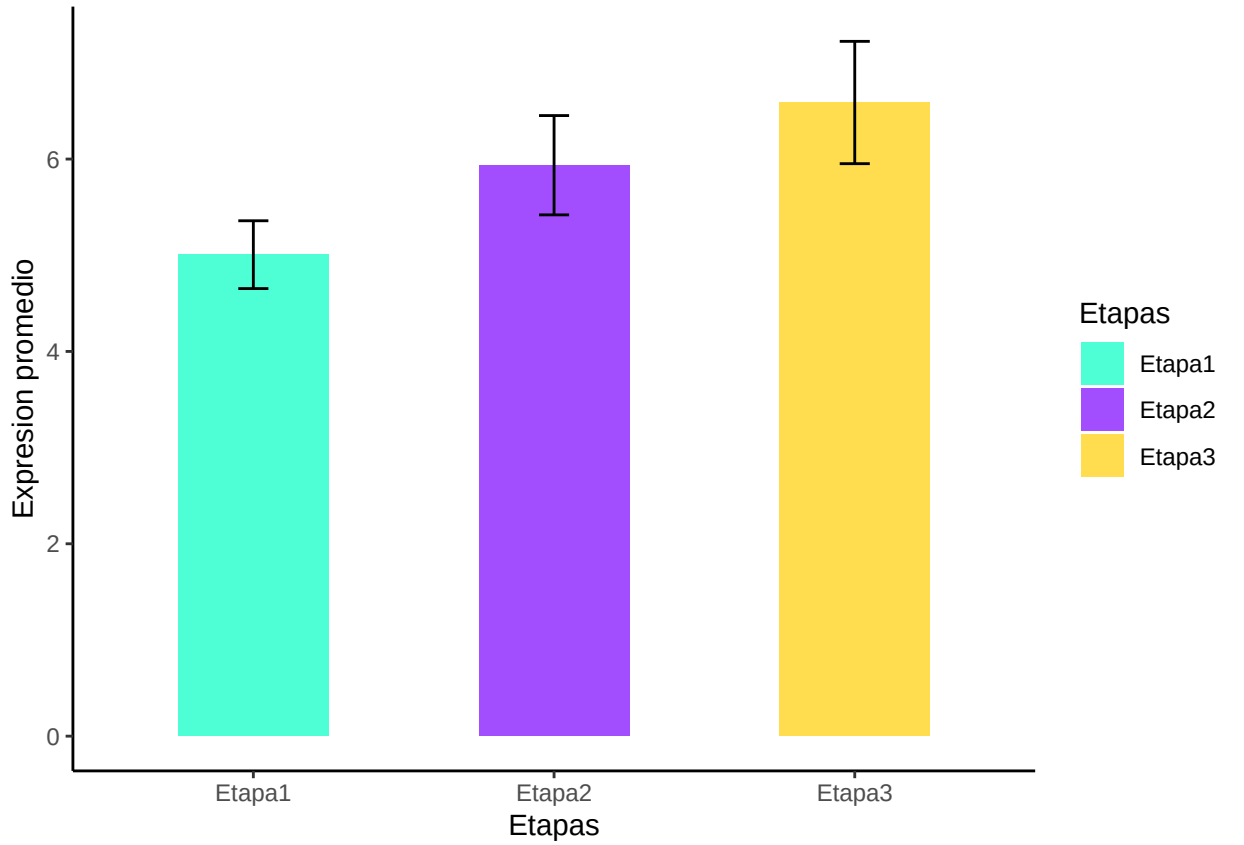
```
# Calcular el promedio y la desviación estándar con dplyr
resumen_E <- E %>% #Especificamos donde se va a guardar
  group_by(Etapas) %>% #Agrupamos por etapas
  summarise(Expresion_promedio = mean(TAR), DE = sd(TAR)) #calculamos promedio y desviación

resumen_E #vemos la variable
```

```
## # A tibble: 3 x 3
##   Etapas Expresion_promedio    DE
##   <chr>               <dbl> <dbl>
## 1 Etapa1               5.01 0.352
## 2 Etapa2               5.94 0.516
## 3 Etapa3               6.59 0.636
```

Grafica de barras con barras de error

```
#Definimos los colores para cada especie
colores <- c("#00FFC3", "#7A00FF", "#FFCF00")

ggplot(resumen_E, aes(x = Etapas, y = Expresion_promedio, fill = Etapas)) +
  geom_bar(stat = "identity", alpha = 0.7, width = 0.5) +
  geom_errorbar(aes(ymin = Expresion_promedio - DE,
                    ymax = Expresion_promedio + DE),
                width = 0.1, color = "black",
                position = position_dodge(0.9), size = 0.5) + #datos de la barra de error
  scale_fill_manual(values = colores) +  # colores
  labs(x = "Etapas", y = "Expresion promedio") +
  theme_classic()
```

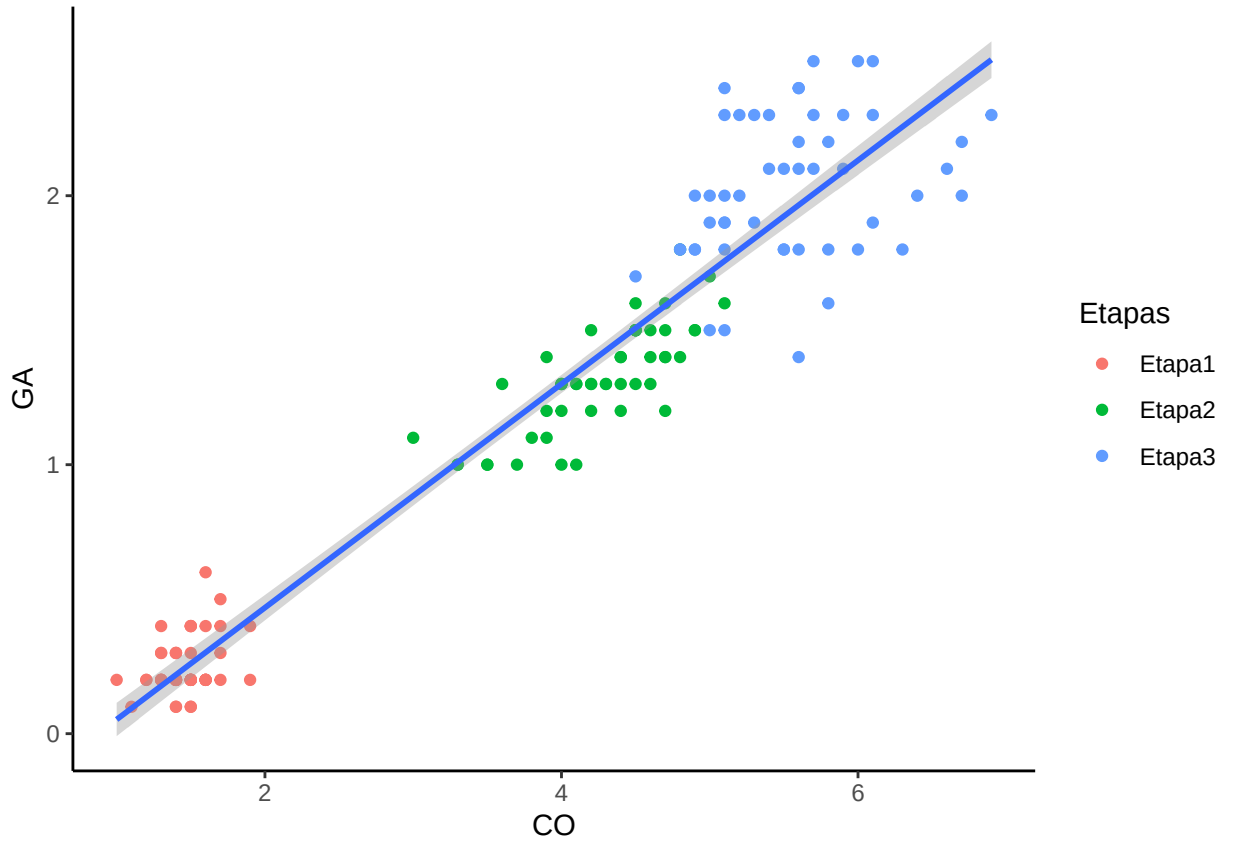
```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



Agregar recta de regresión lineal

```
ggplot(data = E, mapping = aes(x = CO, y = GA)) + #mapeamos los datos
  geom_point(aes(color = Etapas)) + #Diferenciamos con puntos entre etapas
  geom_smooth(method = "lm") + #Utilizamos el método de regresión llineal
  theme_classic()
```

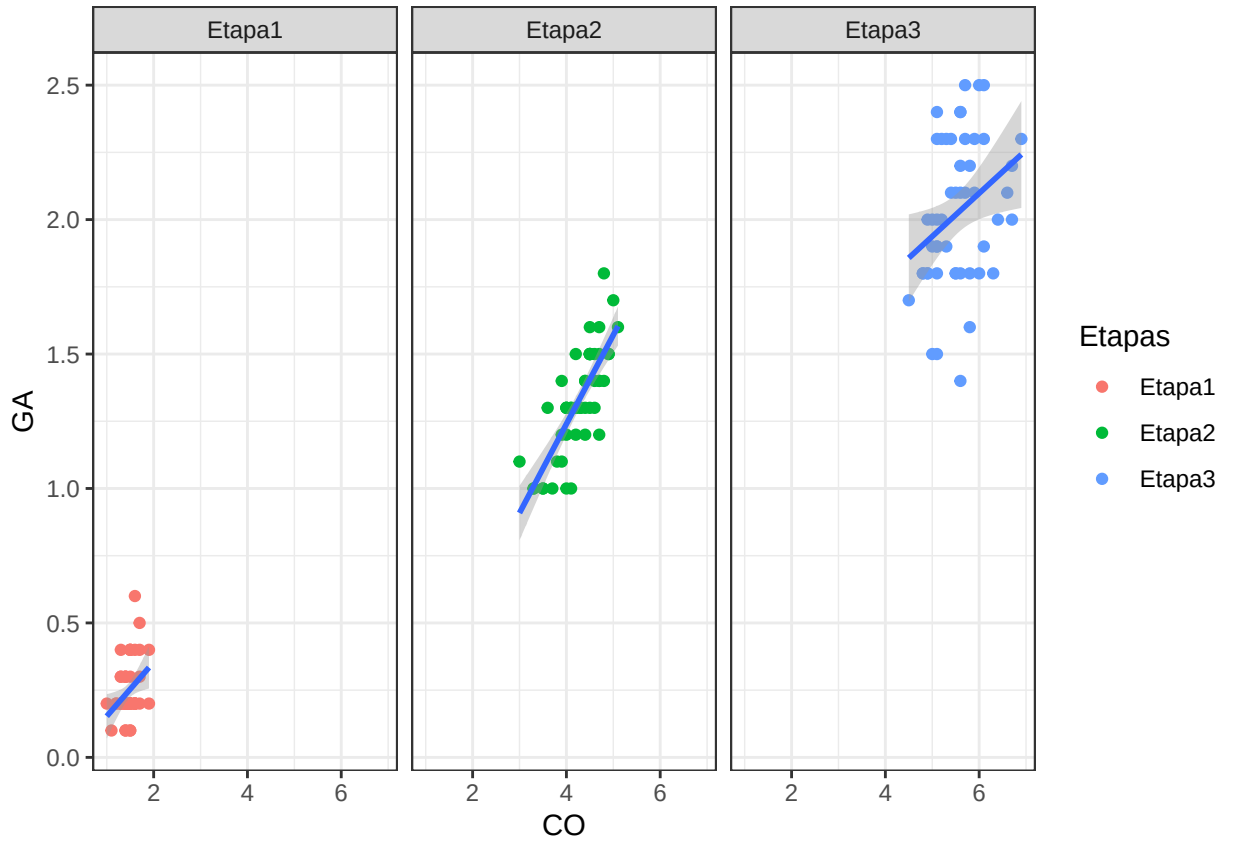
```
## `geom_smooth()` using formula = 'y ~ x'
```



Tambien podemos separar los datos de cada etapa

```
ggplot(data = E, mapping = aes(x = CO, y = GA)) + #mapeamos los datos
  geom_point(aes(color = Etapas)) + #Diferenciamos con puntos entre etapas
  geom_smooth(method = "lm") +
  facet_wrap(~Etapas) + #separamos datos por etapa
  theme_bw()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Exportar el gráfico en jpg y con 300 dpi
ggsave("U3_plot21v2.jpg", width = 25, height = 14, units = c("cm"), dpi = 300)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
# Exportar en otro formato, tiff, png, pdf solo hay que cambiarlo en el nombre
ggsave("U3_plot21v3.tiff", width = 25, height = 14, units = c("cm"), dpi = 300)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```