Clase 6 (Beta)

Alexis Zúñiga & Ernesto Barrios

Capítulo 16: Graficación Básica

La visualización de gráficas es una de las herramientas más poderosas y usadas de R. A continuación utilizaremos la función `plot()` que es una de las herramientas principales para hacerlo.

Preparar los datos

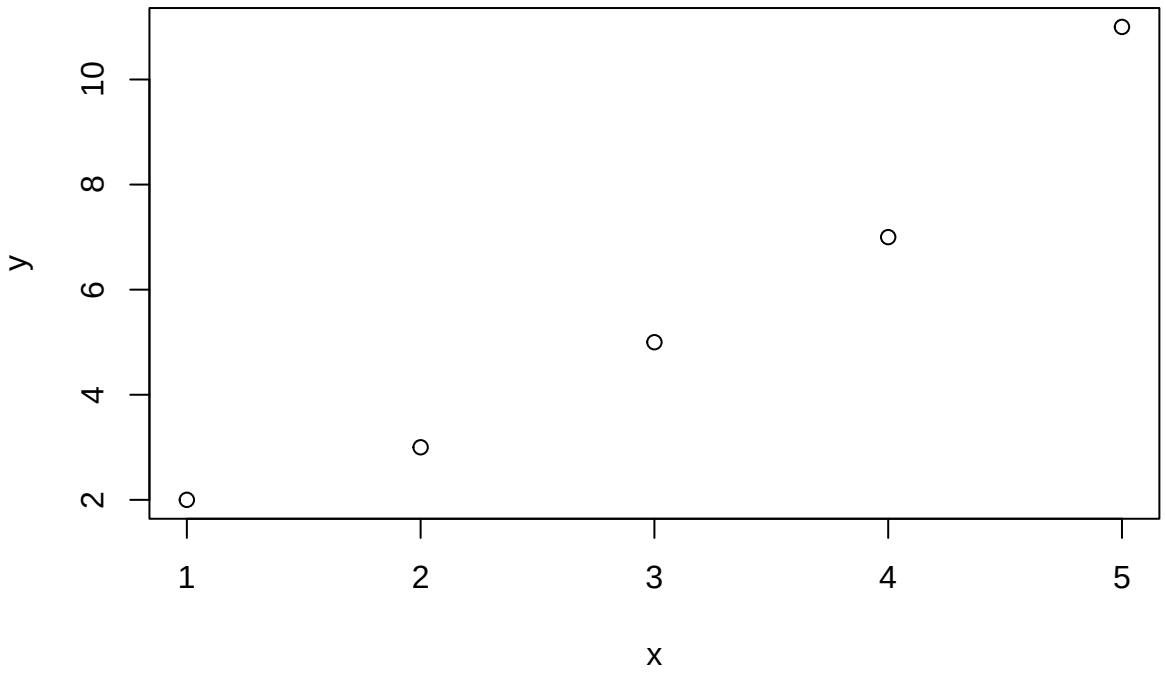
Uno de los primeros pasos para la graficación en R es tener qué vamos a graficar. Para ello debes preparar primero los datos, estos pueden ser vectores, arreglos, data frames, etc.

```
# Ejemplo de datos
x <- c(1, 2, 3, 4, 5)
y <- c(2, 3, 5, 7, 11)
```

Obtener el gráfico

La función `plot()` es la forma más sencilla de crear un gráfico en R . Aquí hay algunos ejemplos básicos:

```
#Grafico de dispersión
plot(x, y)
```

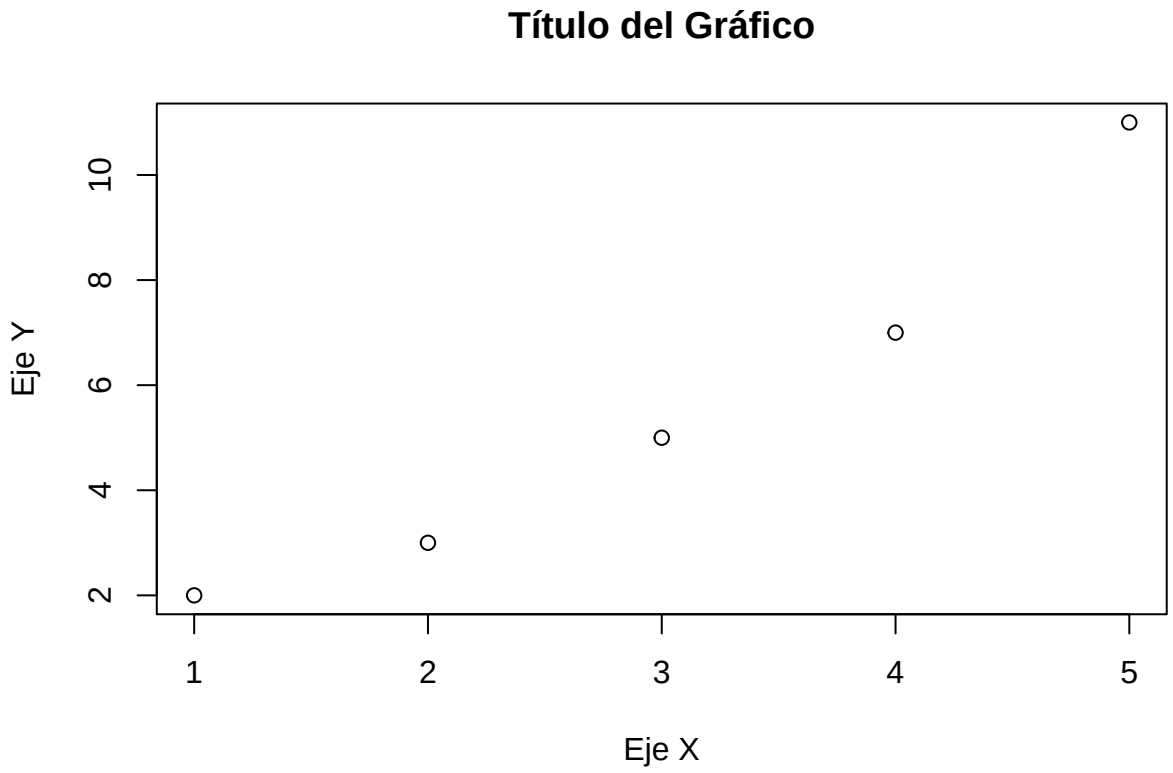


Este comando creará un gráfico de dispersión con x en el eje horizontal e y en el eje vertical.

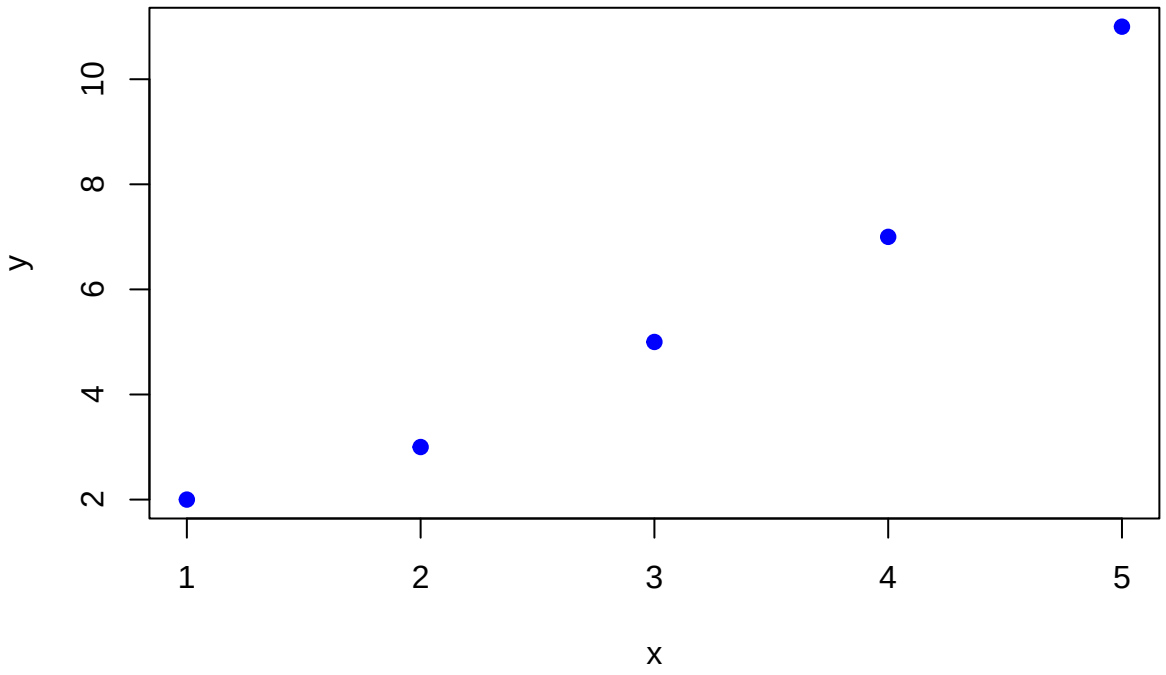
Personalizar el gráfico

Puedes personalizar el gráfico agregando títulos, etiquetas y cambiando el tipo de puntos o líneas.

```
#Titulos y etiquetas
plot(x, y,
     main = "Título del Gráfico",  # Título del gráfico
     xlab = "Eje X",               # Etiqueta del eje X
     ylab = "Eje Y")               # Etiqueta del eje Y
```



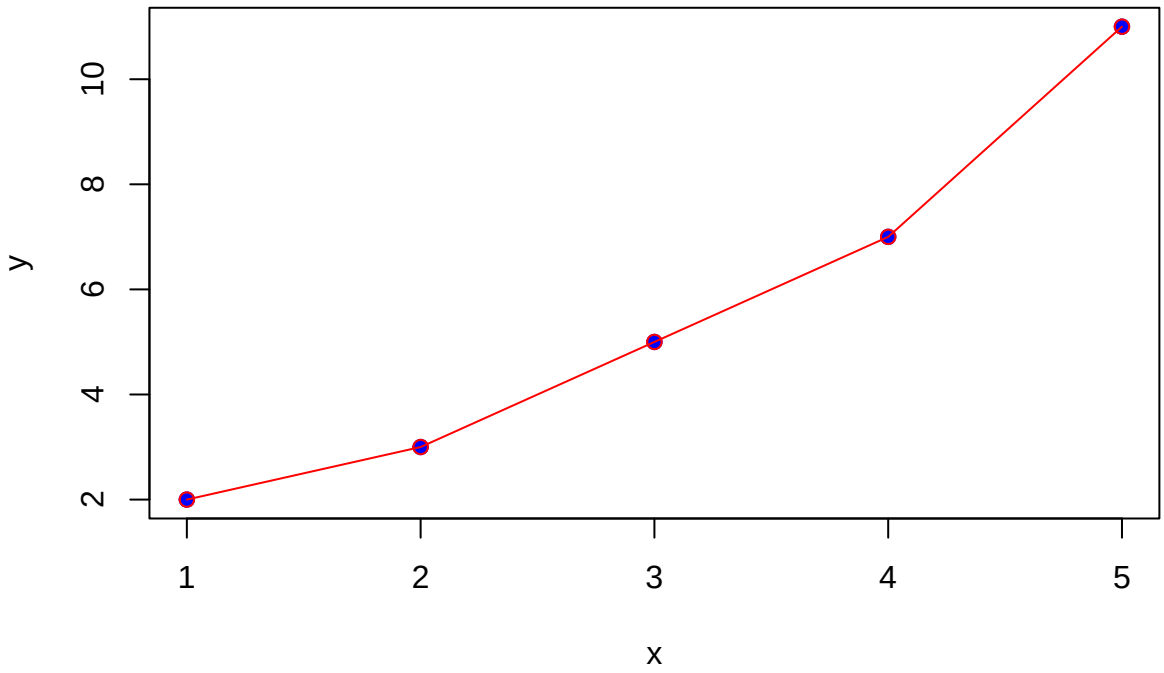

```
#Cambiar tipo de puntos y color
plot(x, y,
     pch = 19,                     # Tipo de punto (19 es un círculo sólido)
     col = "blue")                 # Color de los puntos
```



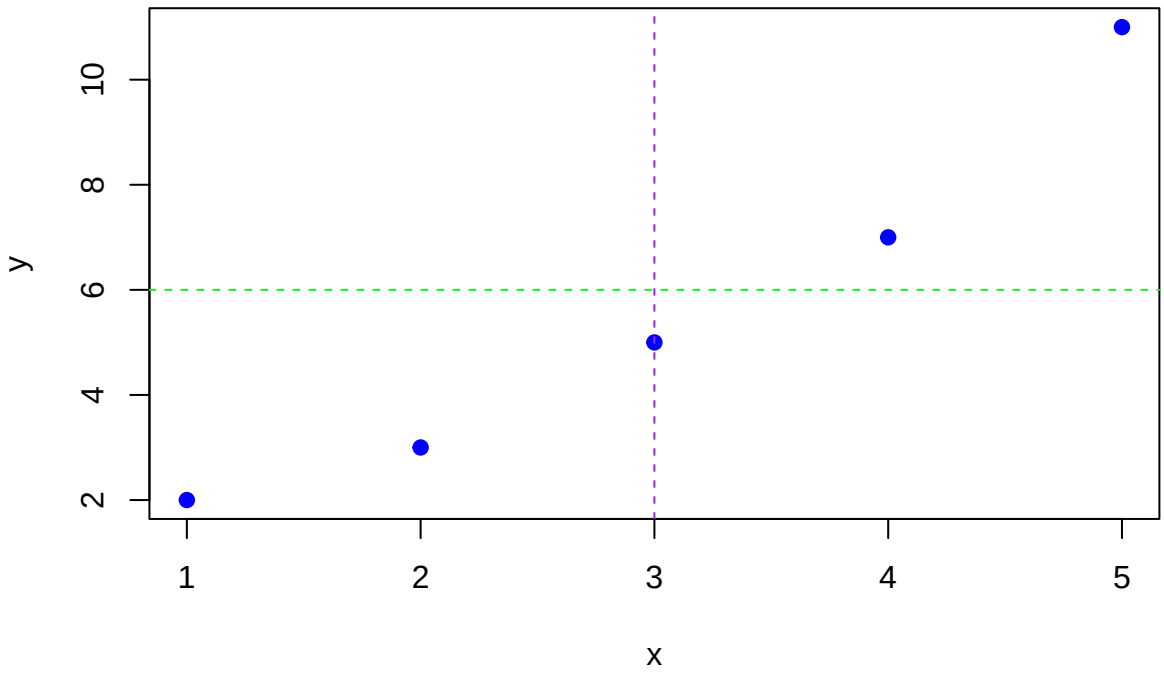
Añadir elemento al gráfico

Puedes agregar más elementos al gráfico existente usando funciones como `lines()`, `points()`, `abline()`, etc.

```
## Añadir una linea
# Crear el gráfico base
plot(x, y,
     pch = 19,
     col = "blue")

# Añadir una línea
lines(x, y, type = "o", col = "red")  # "o" indica que se dibujen líneas y puntos
```




```r
## Añadir una línea de referencia
# Crear el gráfico base
plot(x, y,
     pch = 19,
     col = "blue")

# Añadir una línea horizontal en y = 6
abline(h = 6, col = "green", lty = 2)  # lty = 2 es una línea discontinua

# Añadir una línea vertical en x = 3
abline(v = 3, col = "purple", lty = 2)
```



Guardar el gráfico

Puedes guardar el gráfico en diferentes formatos (PNG, PDF, etc.) usando funciones como `png()`, `pdf()`, `jpeg()`, etc.

```
# Abrir el dispositivo gráfico
png("mi_grafico.png")

# Crear el gráfico
plot(x, y,
     pch = 19,
     col = "blue",
     main = "Gráfico de Ejemplo",
     xlab = "Eje X",
     ylab = "Eje Y")

# Cerrar el dispositivo gráfico
dev.off()
```

```
## pdf
##   2
```

Ejemplo completo:

```
# Datos
x <- c(1, 2, 3, 4, 5)
y <- c(2, 3, 5, 7, 11)

# Abrir el dispositivo gráfico
png("mi_grafico_completo.png")

# Crear el gráfico
plot(x, y,
     main = "Gráfico Completo",
     xlab = "Eje X",
     ylab = "Eje Y",
     pch = 19,
     col = "blue")

# Añadir una línea de puntos y líneas
lines(x, y, type = "o", col = "red")

# Añadir líneas de referencia
abline(h = 6, col = "green", lty = 2)
abline(v = 3, col = "purple", lty = 2)

# Cerrar el dispositivo gráfico
dev.off()
```

```
## pdf
##   2
```

Gráficos especificos

Este tipo de gráfica que acabamo de ver no es la única forma, Rofrece gráficos más especificos dependiendo lo que se ocupe y los tipos de datos que se tengan.

Histogramas

Los histogramas se utilizan para visualizar la distribución de un conjunto de datos.

```r
# Datos
data <- rnorm(1000)  # Generar 1000 números aleatorios con distribución normal

# Crear histograma
hist(data,
     main = "Histograma de Distribución Normal",
     xlab = "Valor",
     col = "skyblue",
     border = "black")
```

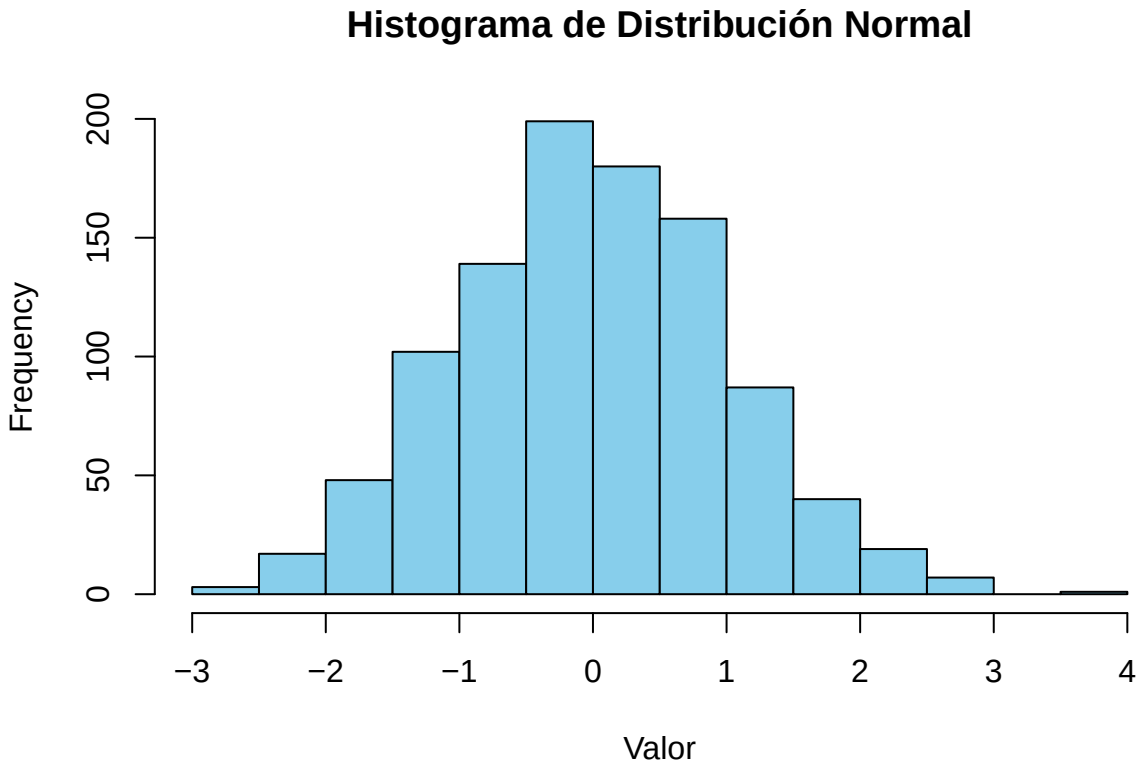
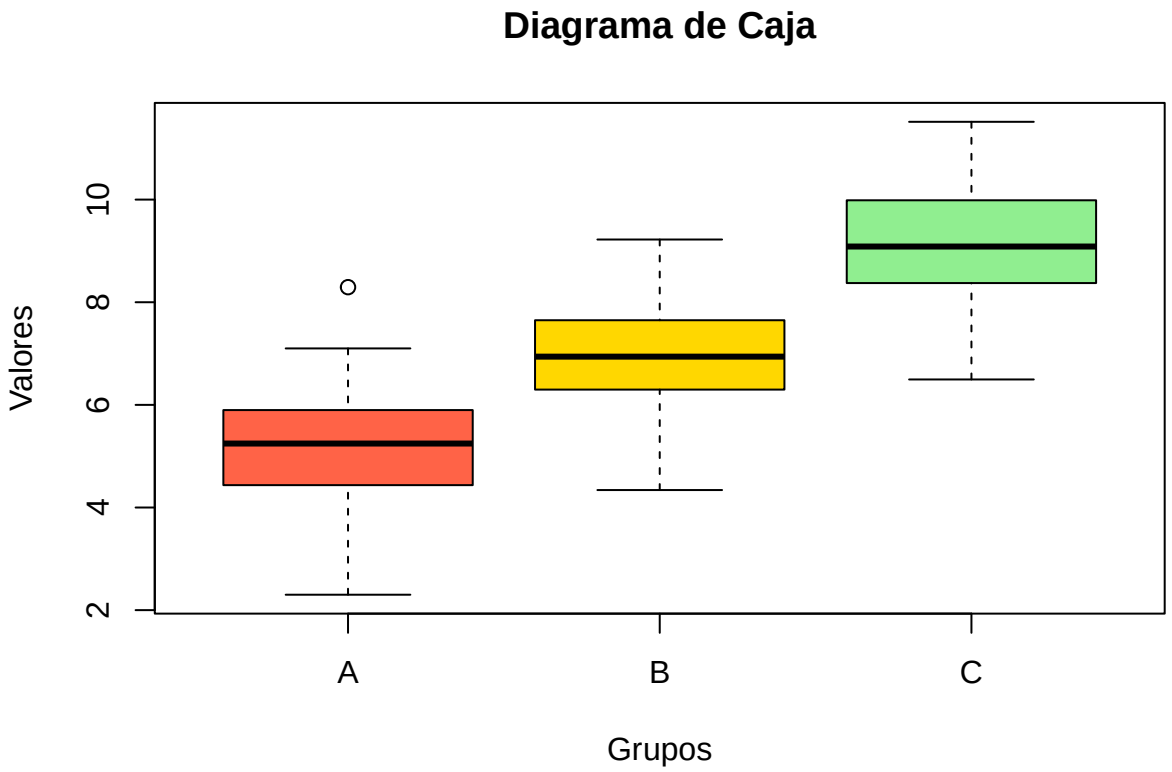


Diagrama de caja (boxplots)

Los diagramas de caja se utilizan para visualizar la distribución de datos y detectar valores atípicos.

```r
# Datos
data <- list(
  A = rnorm(100, mean = 5),
  B = rnorm(100, mean = 7),
  C = rnorm(100, mean = 9)
)

# Crear diagrama de caja
boxplot(data,
        main = "Diagrama de Caja",
        xlab = "Grupos",
        ylab = "Valores",
        col = c("tomato", "gold", "lightgreen"))
```



Gráficos de líneas

Los gráficos de líneas se utilizan para mostrar datos a lo largo de un intervalo de tiempo.

```r
# Datos
time <- 1:10
values <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)

# Crear gráfico de líneas
plot(time, values,
     type = "o",               # "o" para líneas y puntos
     main = "Gráfico de Líneas",
     xlab = "Tiempo",
     ylab = "Valores",
     col = "blue",
     pch = 16)                 # Tipo de punto
```

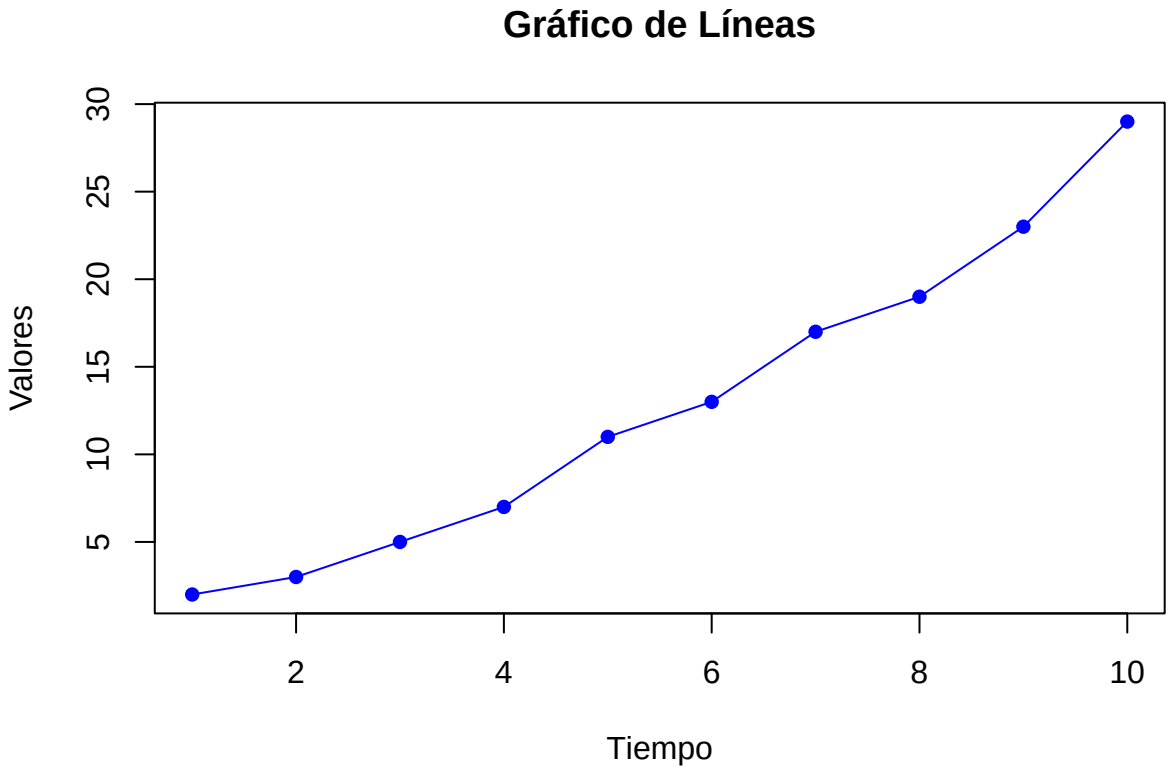


Gráfico de pie

Los gráficos de pie se utilizan para mostrar proporciones.

```r
# Datos
slices <- c(10, 20, 30, 40)
labels <- c("A", "B", "C", "D")

# Crear gráfico de sectores
pie(slices,
    labels = labels,
    main = "Gráfico de Sectores",
    col = rainbow(length(slices)))
```

Gráfico de Sectores

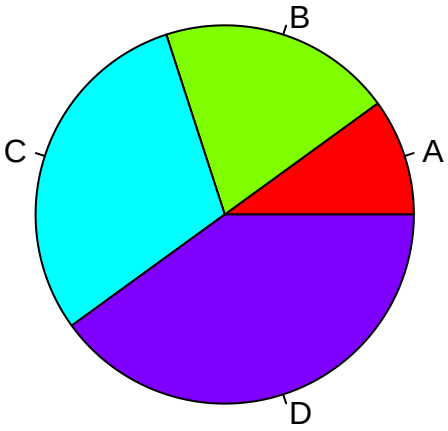
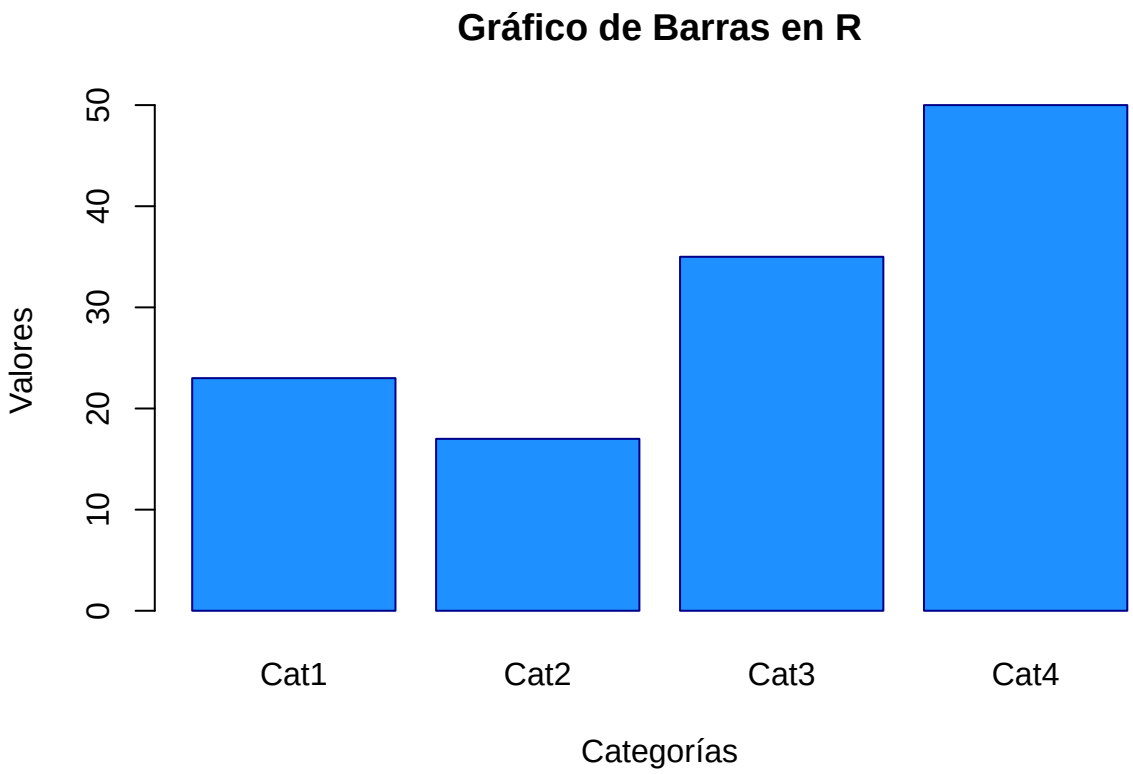


Gráfico de barras

Los gráficos de barras básicos se pueden crear usando la función `barplot()`.

```
# Datos
categories <- c("Cat1", "Cat2", "Cat3", "Cat4")
values <- c(23, 17, 35, 50)

# Crear gráfico de barras
barplot(
  height = values,
  names.arg = categories,
  col = "dodgerblue",
  border = "darkblue",
  main = "Gráfico de Barras en R",
  xlab = "Categorías",
  ylab = "Valores"
)
```



Gráficos de calor (heatmaps)

Los gráficos de calor se utilizan para mostrar la intensidad de valores en una matriz.

```
# Datos
matrix_data <- matrix(rnorm(100), nrow = 10)

# Crear gráfico de calor
heatmap(matrix_data,
        main = "Gráfico de Calor",
        col = heat.colors(256),
        scale = "column")
```

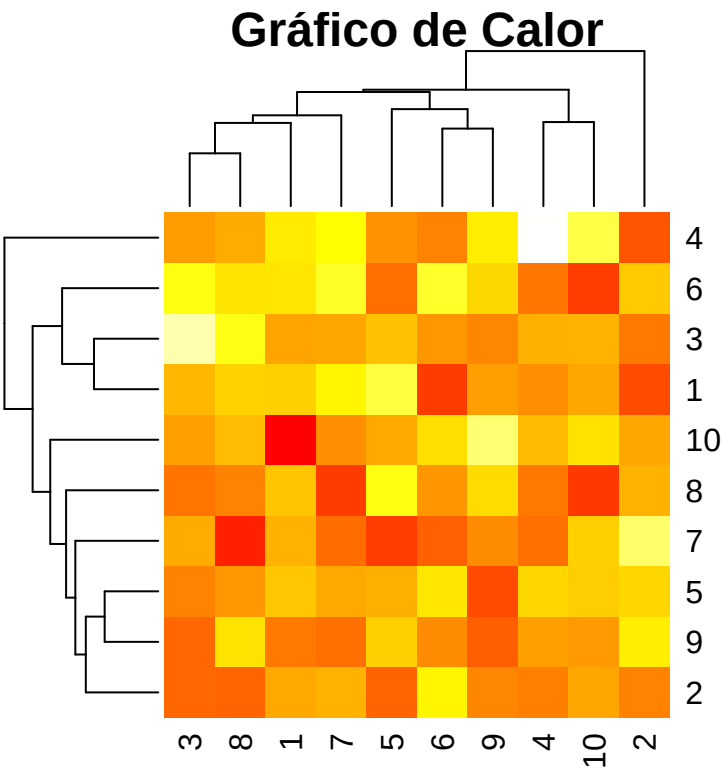


Gráfico de Calor
4
6
3
1
10
8
7
5
9
2
3
8
1
7
5
6
9
4
10
2

Tidyverse

La colección de paquetes tidyverse incluye ggplot2 junto con otras herramientas útiles para manipulación y análisis de datos como dplyr, tidyr, readr, entre otros.

```
install.packages("tidyverse")
```

```
## Installing package into 'C:/usr/R/contrib'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\usr\R\tmp\RtmpOsdTOw\downloaded_packages
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Gráfico de dispersión (scatter plot)

```
# Datos de ejemplo
df <- tibble(x = rnorm(100), y = rnorm(100))

# Crear gráfico de dispersión
ggplot(df, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Gráfico de Dispersión",
       x = "Eje X",
       y = "Eje Y")
```

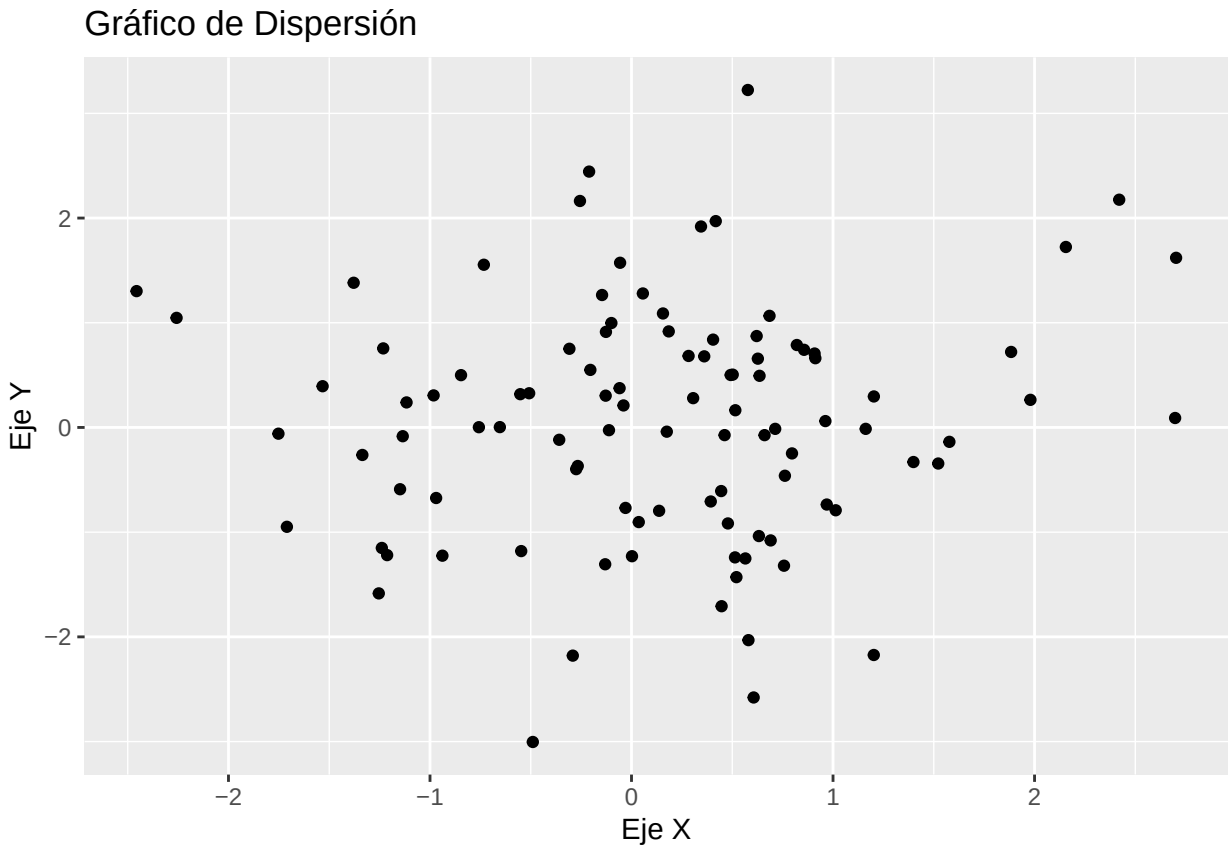


Gráfico de Barras

```
# Datos de ejemplo
categories <- c("Cat1", "Cat2", "Cat3", "Cat4")
values <- c(23, 17, 35, 50)
df <- tibble(category = categories, value = values)

# Crear gráfico de barras
ggplot(df, aes(x = category, y = value)) +
  geom_bar(stat = "identity", fill = "dodgerblue", color = "darkblue") +
  labs(title = "Gráfico de Barras",
       x = "Categorías",
       y = "Valores")
```

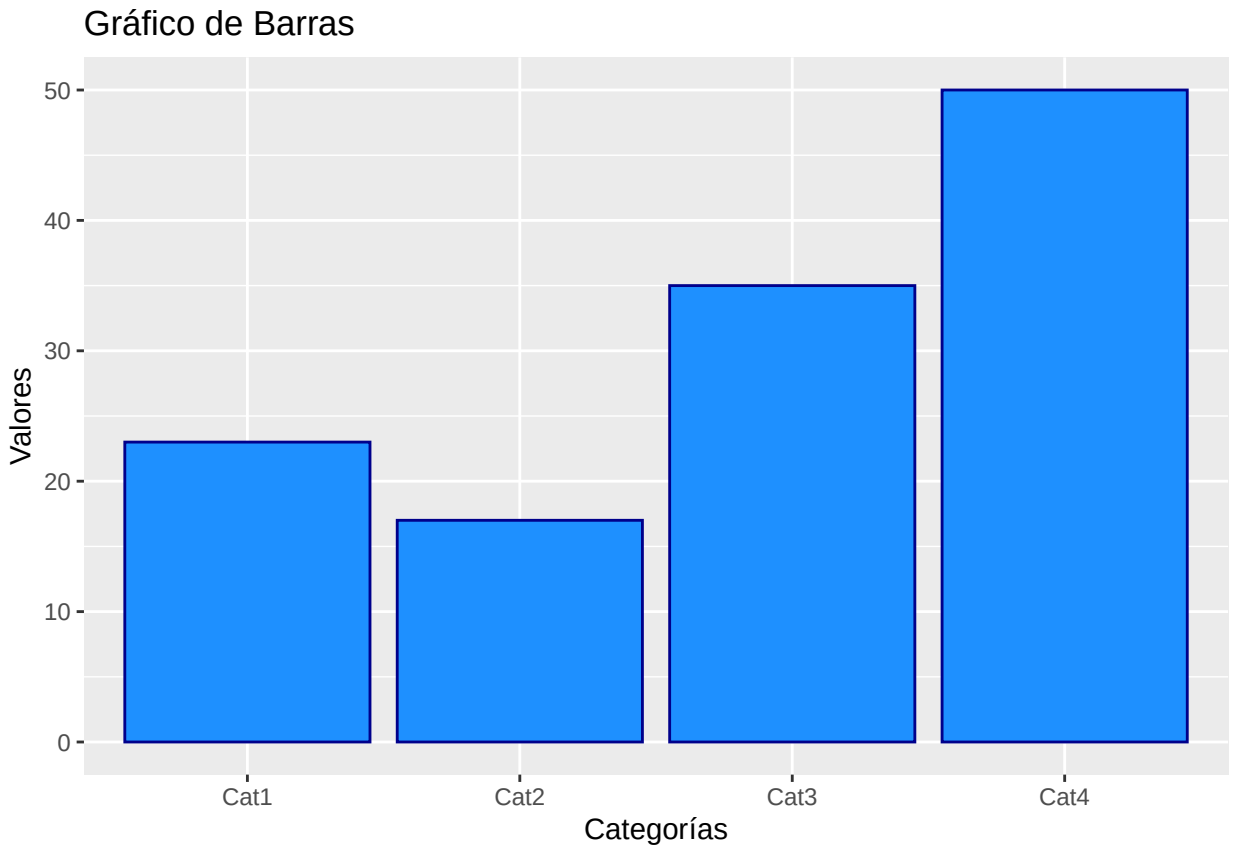
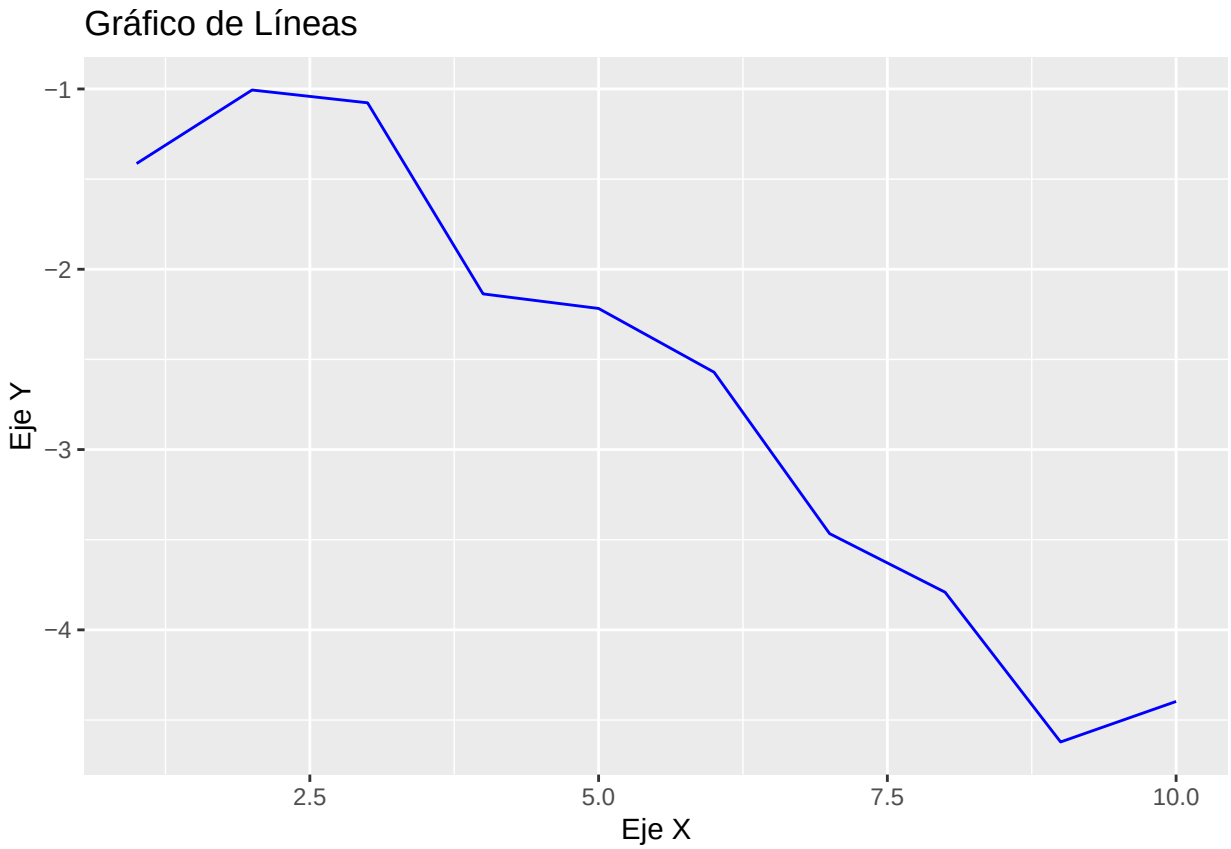


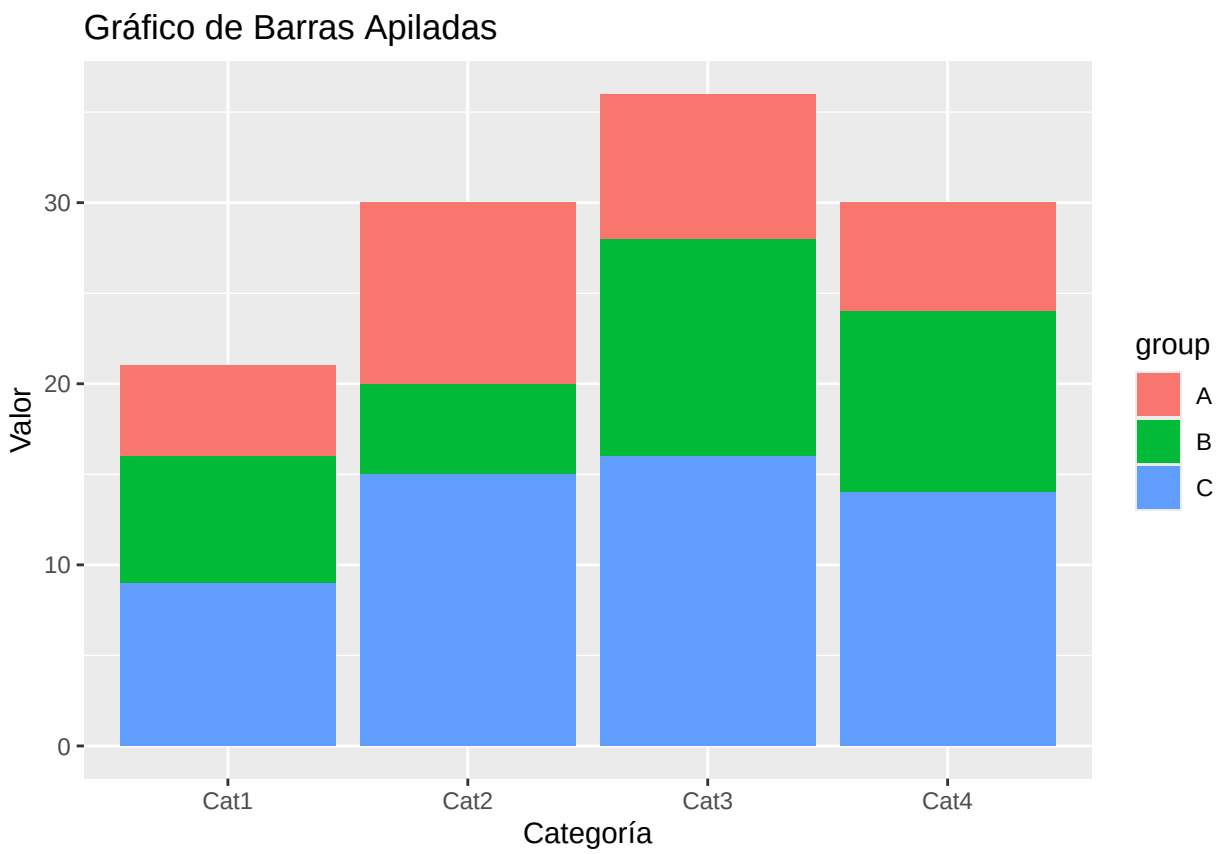
Gráfico de Linea

```r
# Datos de ejemplo
df <- tibble(x = 1:10, y = cumsum(rnorm(10)))

# Crear gráfico de líneas
ggplot(df, aes(x = x, y = y)) +
  geom_line(color = "blue") +
  labs(title = "Gráfico de Líneas",
       x = "Eje X",
       y = "Eje Y")
```



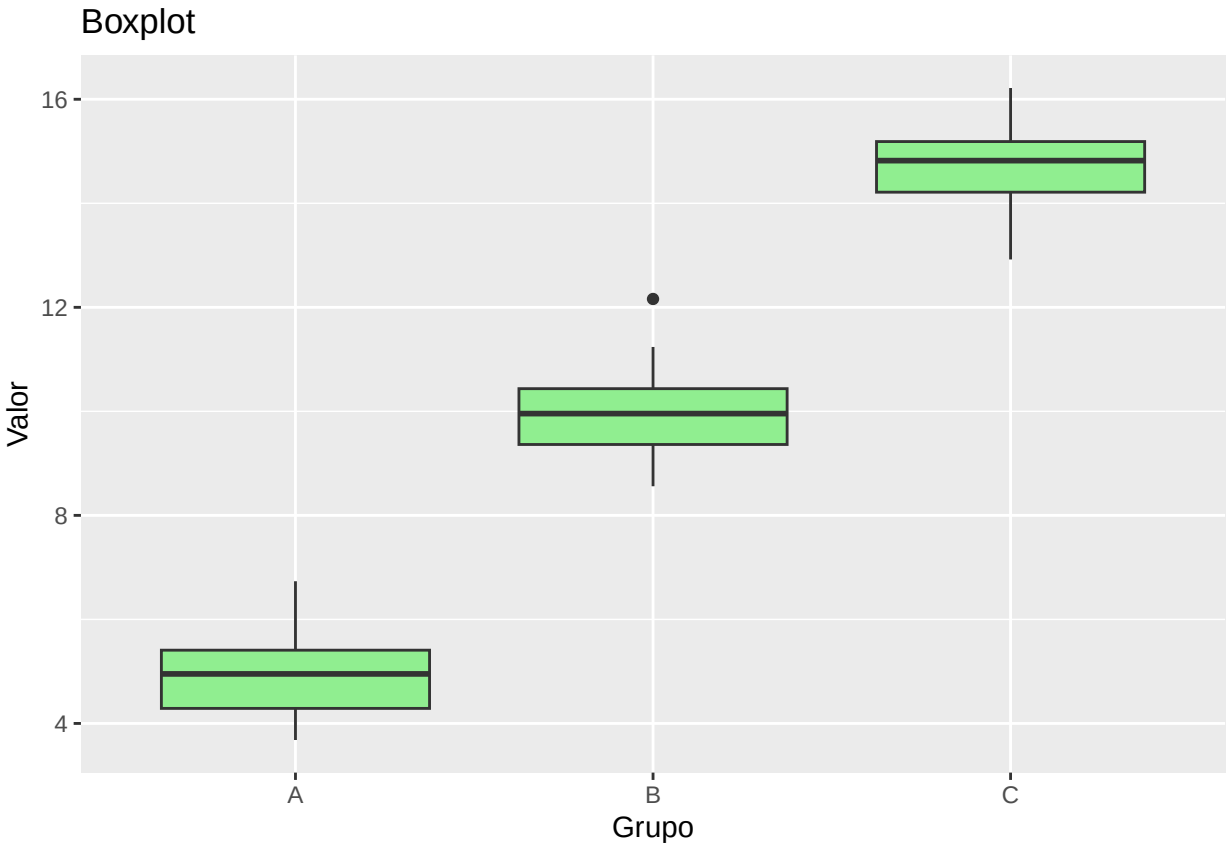
Barras ampliadas

```
# Datos de ejemplo
df <- tibble(group = rep(c("A", "B", "C"), times = 4),
             category = rep(c("Cat1", "Cat2", "Cat3", "Cat4"), each = 3),
             value = c(5, 7, 9, 10, 5, 15, 8, 12, 16, 6, 10, 14))

# Crear gráfico de barras apiladas
ggplot(df, aes(x = category, y = value, fill = group)) +
  geom_bar(stat = "identity") +
  labs(title = "Gráfico de Barras Apiladas",
       x = "Categoría",
       y = "Valor")
```



Boxplot

```r
# Datos de ejemplo
df <- tibble(group = rep(c("A", "B", "C"), each = 20),
             value = c(rnorm(20, mean = 5), rnorm(20, mean = 10), rnorm(20, mean = 15)))

# Crear boxplot
ggplot(df, aes(x = group, y = value)) +
  geom_boxplot(fill = "lightgreen") +
  labs(title = "Boxplot",
       x = "Grupo",
       y = "Valor")
```



- **Tibbles: Utilizamos tibble en lugar de data.frame porque es más eficiente y parte del tidyverse.**
- **Manipulación de Datos: Puedes usar dplyr y tidyr para manipular los datos antes de pasarlos a ggplot2. Esto puede ser útil para filtrado, agregación, transformación de datos, etc.**

Estos ejemplos demuestran cómo se pueden crear gráficos básicos utilizando ggplot2 dentro del tidyverse. Esta colección de paquetes facilita tanto la manipulación de datos como la visualización, proporcionando un flujo de trabajo cohesivo y eficiente para el análisis de datos en R.