Unidad II (pre)

Unidad II: gráficos de bajo y alto nivel

Gráficos usando librerías bases de R

Los gráficos básicos en R permiten una visualización clara y efectiva de los datos, facilitando su exploración inicial. La función **`barplot()`** se utiliza para representar datos categóricos mediante barras, lo que permite comparar frecuencias o proporciones entre distintas categorías. Por su parte, **`hist()`** muestra la distribución de una variable numérica continua, agrupando los valores en intervalos y permitiendo identificar patrones como sesgos, simetrías o concentraciones. El **`boxplot()`**, o diagrama de caja, resume visualmente la distribución de una variable, destacando la mediana, los cuartiles y los valores atípicos, siendo útil para comparar varias distribuciones simultáneamente. Finalmente, **`plot()`** en su forma más básica genera gráficos de dispersión entre dos variables numéricas, revelando relaciones, tendencias o correlaciones. Estos gráficos constituyen herramientas fundamentales para el análisis exploratorio de datos.

```
library("knitr")
library(MPV)
```

```
Cargando paquete requerido: lattice
```

```
Cargando paquete requerido: KernSmooth
```

```
KernSmooth 2.23 loaded
Copyright M. P. Wand 1997-2009
```

```
data(WorldPhones)
str(WorldPhones)
```

```
 num [1:7, 1:7] 45939 60423 64721 68484 71799 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:7] "1951" "1956" "1957" "1958" ...
  ..$ : chr [1:7] "N.Amer" "Europe" "Asia" "S.Amer" ...
```

Gráfico de barras

```
WorldPhones51 <- WorldPhones[1, ]
WorldPhones51
```

```
  N.Amer   Europe     Asia   S.Amer  Oceania   Africa Mid.Amer
   45939    21574     2876     1815     1646       89      555
```

```
barplot(WorldPhones51, main = "Telephone Usage in 1951", cex.names = 0.75,
cex.axis = 0.75, ylab = "Telephones (in Thousands)", xlab="Region")
```

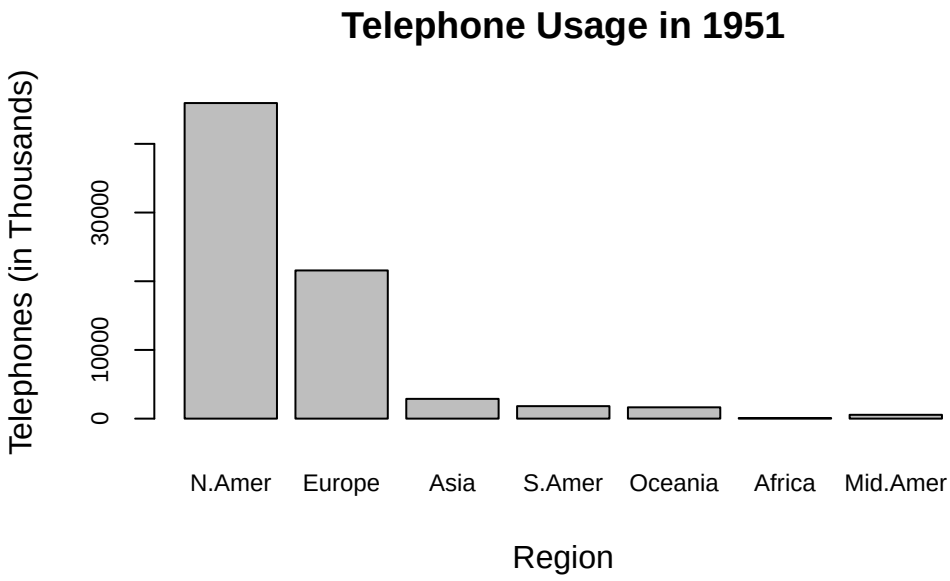


Gráfico de puntos

```
dotchart(WorldPhones51, xlab = "Numbers of Phones ('000s)")
```

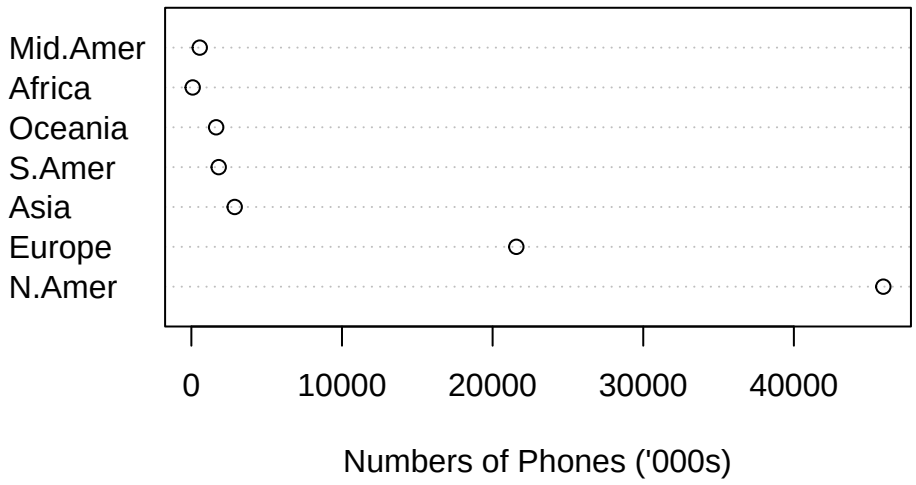


Gráfico de barras por grupos

```
barplot(VADeaths, beside = TRUE, legend = TRUE, ylim = c(0, 90),
ylab = "Deaths per 1000",
main = "Death rates in Virginia")
```

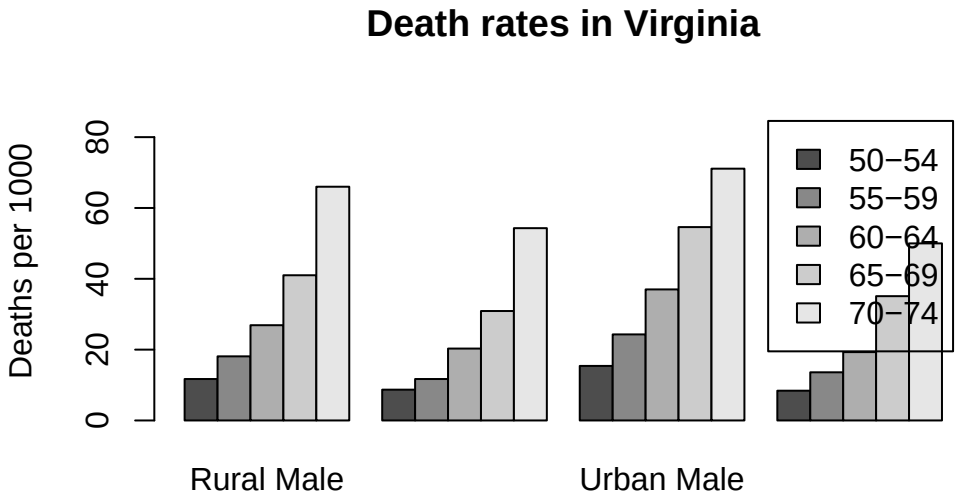


Gráfico de torta

```
groupsizes <- c(18, 30, 32, 10, 10)
labels <- c("A", "B", "C", "D", "F")
```

```
pie(groupsizes, labels,
col = c("grey40", "white", "grey", "black", "grey90"))
```

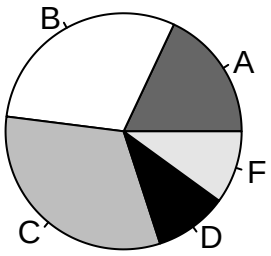


Gráfico de histograma

```
hist(log(1000*islands, 10), xlab = "Area (on base 10 log scale)",
main = "Areas of the World's Largest Landmasses")
```

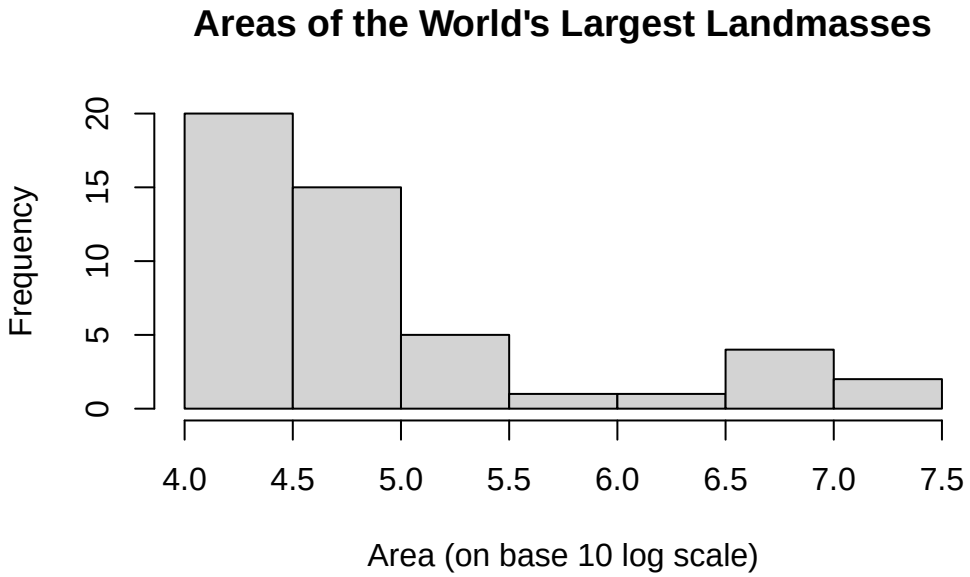
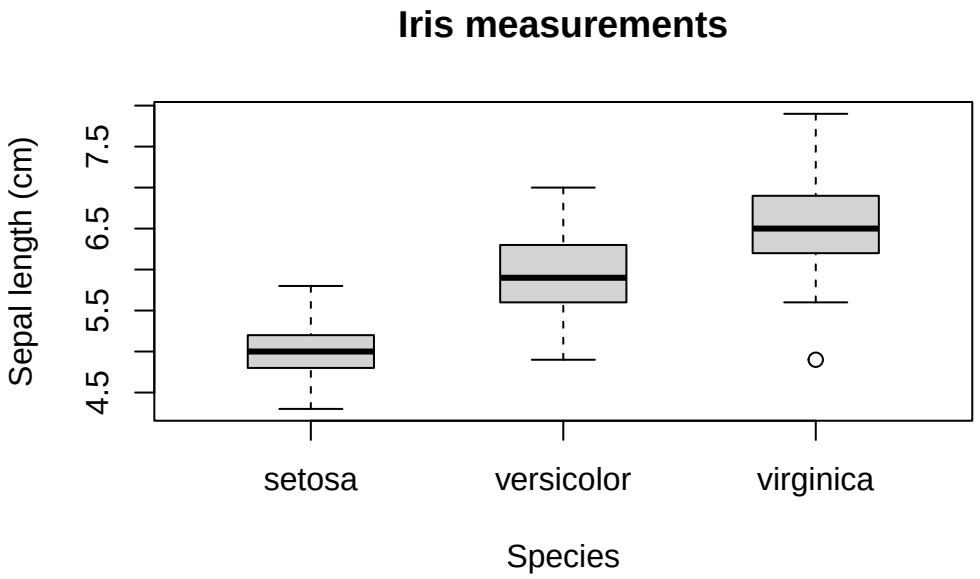


Gráfico de caja y bigotes

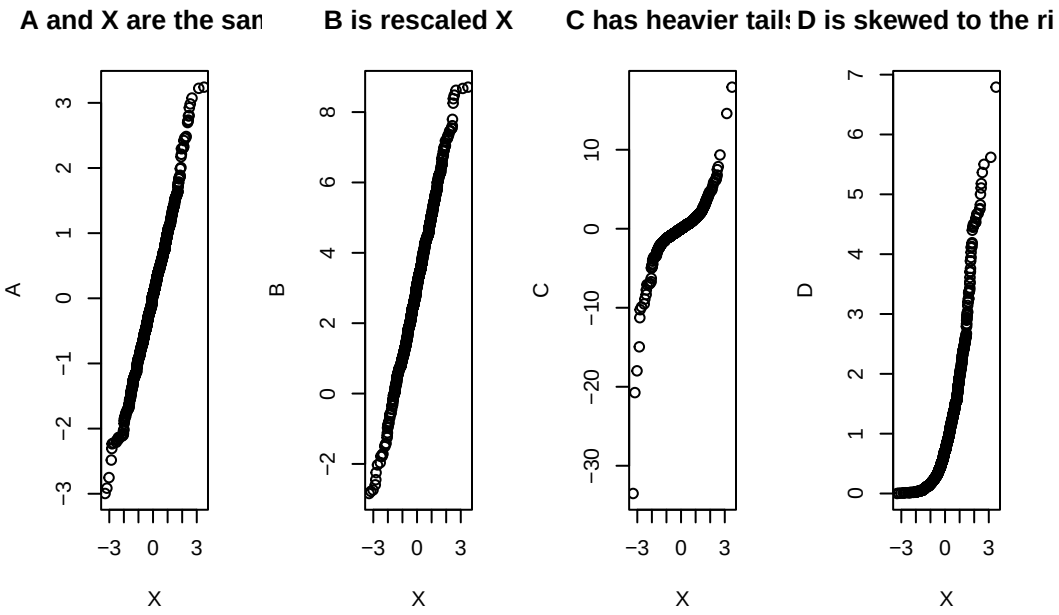
```
data(iris)

boxplot(Sepal.Length ~ Species, data = iris,
```

```
ylab = "Sepal length (cm)", main = "Iris measurements",
boxwex = 0.5)
```



Gráficos QQplots (quantile quantile)

```
par(mfrow = c(1,4))
X <- rnorm(1000)
A <- rnorm(1000)
qqplot(X, A, main = "A and X are the same")
B <- rnorm(1000, mean = 3, sd = 2)
qqplot(X, B, main = "B is rescaled X")
C <- rt(1000, df = 2)
qqplot(X, C, main = "C has heavier tails")
D <- rexp(1000)
qqplot(X, D, main = "D is skewed to the right")
```



La gramatica de los gráficos

El libro *The Grammar of Graphics* de Leland Wilkinson propone una visión estructurada y modular de la visualización de datos, en la que los gráficos no son simplemente imágenes estáticas, sino construcciones formales compuestas por elementos fundamentales. Esta obra establece una base teórica sólida para entender cómo se generan los gráficos, descomponiéndolos en componentes como datos, transformaciones estadísticas, geometrías, escalas, coordenadas y guías. Esta perspectiva ha influido profundamente en el desarrollo de herramientas modernas de visualización, como el paquete `ggplot2` en R, que implementa esta gramática de manera práctica y flexible, permitiendo a los usuarios construir gráficos complejos a partir de principios simples y combinables.

La idea de "gramática" en este contexto se refiere a un conjunto de reglas y estructuras que, al igual que en el lenguaje natural, permiten construir expresiones complejas a partir de unidades básicas. En el caso de los gráficos, estas unidades incluyen los datos (el contenido), las geometrías (cómo se representan visualmente), las escalas (cómo se mapean los valores), y las coordenadas (el sistema de referencia). Esta gramática permite que los gráficos sean generados de forma coherente, reproducible y extensible, lo que resulta especialmente útil en programación, donde la claridad y la modularidad son esenciales. Así, crear un gráfico en `ggplot2` no es simplemente dibujar, sino componer una estructura visual con significado, basada en reglas bien definidas.

`ggplot2` implementa una gramática de los gráficos inspirada en la obra de Leland Wilkinson, donde cada visualización se construye como una oración compuesta por elementos básicos que siguen reglas definidas. En esta gramática, los **datos** actúan como el sujeto, la **estética**

como la estructura gramátical que conecta variables con atributos visuales, y la **geometría** como el verbo que define la forma del gráfico (puntos, líneas, barras, etc.). A estos se suman componentes como transformaciones estadísticas, escalas, coordenadas, temas y etiquetas, que enriquecen y completan el significado visual. Esta estructura modular permite construir gráficos complejos de manera lógica, clara y reproducible, haciendo de `ggplot2` una herramienta poderosa y elegante para el análisis visual de datos. La fórmula que resume esta lógica es:

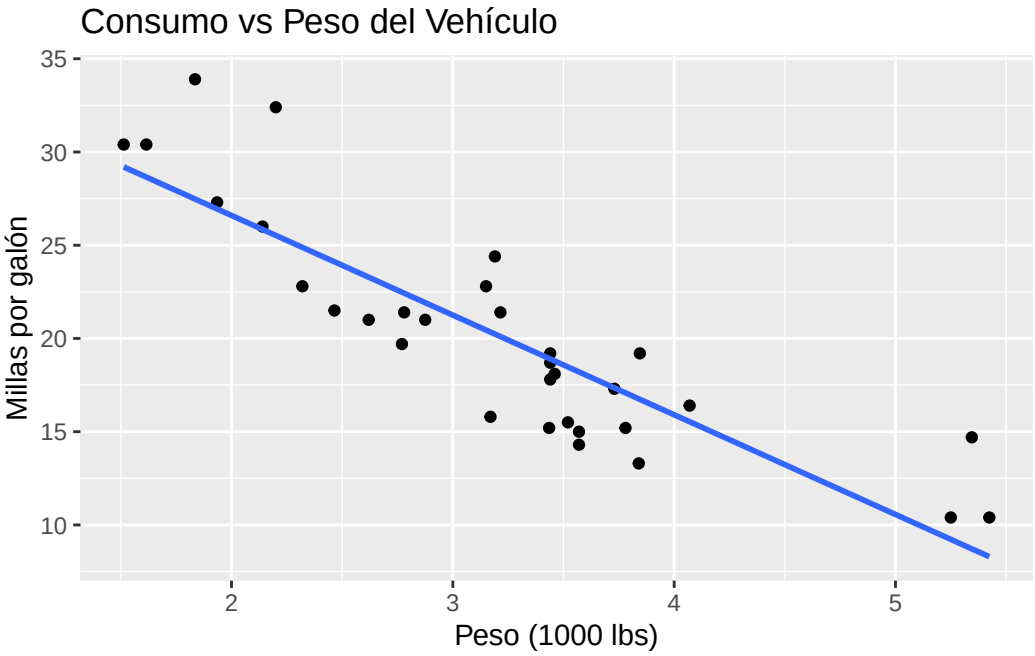
Gráfico = Datos + Estética + Geometría + (Transformaciones + Escalas + Coordenadas + Temas + Etiquetas),

lo que refleja cómo, al igual que en el lenguaje, se pueden formar expresiones ricas y precisas a partir de reglas simples y combinables.

Un ejemplo concreto de esta gramática en acción puede verse en el siguiente código en R, que utiliza `ggplot2` para explorar la relación entre el peso y el consumo de combustible de automóviles:

```
library(ggplot2)
data(mtcars)

ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Consumo vs Peso del Vehículo",
       x = "Peso (1000 lbs)",
       y = "Millas por galón")
```

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



En este gráfico, los datos (`mtcars`) se mapean a los ejes mediante `aes()`, se representan con puntos (`geom_point()`), se añade una capa de modelo lineal (`geom_smooth()`), y se etiquetan con `labs()`. Cada componente responde a una parte de la gramática, lo que permite construir visualizaciones claras, interpretables y adaptables a distintos contextos analíticos.

Gráficos personalizados con `ggplot2`

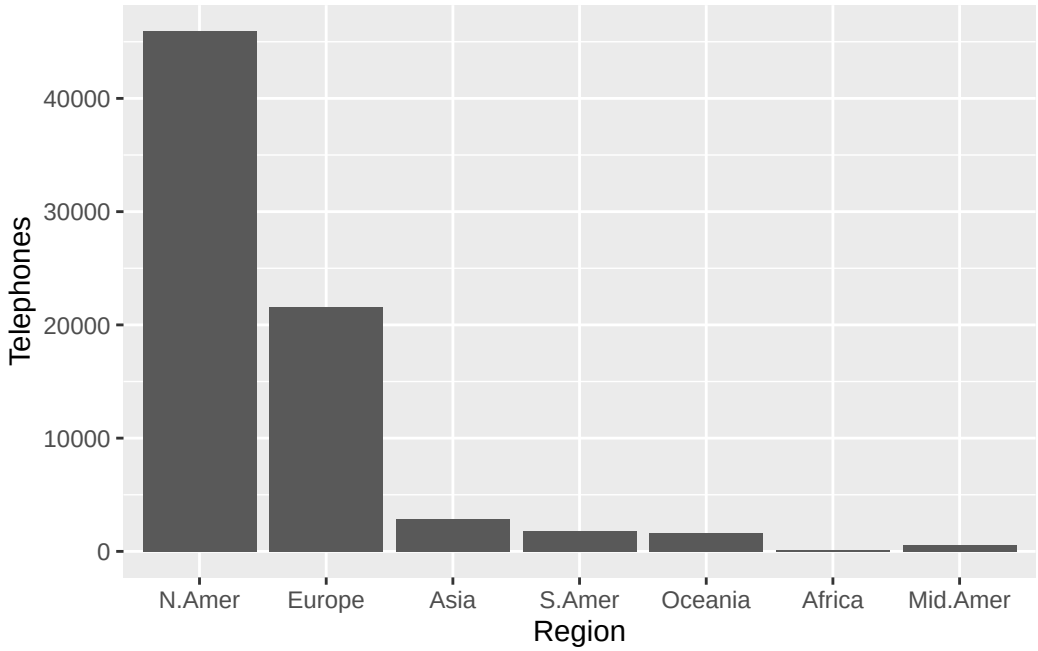
```
#library(ggplot2)
data("windWin80")
str(windWin80)
```

```
'data.frame':   366 obs. of  2 variables:
 $ h0 : int  4 13 0 20 15 20 37 19 22 17 ...
 $ h12: int  15 9 19 22 9 48 20 28 15 30 ...
```

Barplot con `ggplot2`

```
library(ggplot2)
region <- names(WorldPhones51)
phones51 <- data.frame(Region = factor(region, levels = region),
```

```
Telephones = WorldPhones51)
ggplot(data = phones51, aes(x = Region, y = Telephones)) + geom_col()
```



Agregando la grid

```
g1 <- ggplot(phones51, aes(Region, Telephones))
g1
```

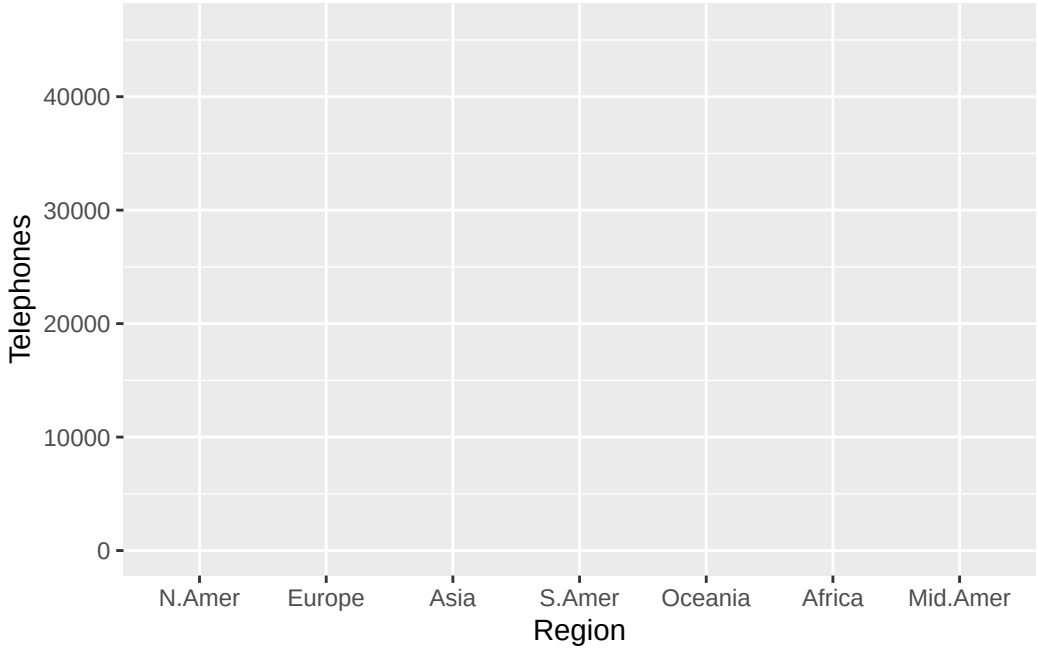


Gráfico circular en 'ggplot2'

```
ggplot(phones51, aes(x = "", y = Telephones, fill = Region)) +
coord_polar(theta = "y") +
geom_col()
```

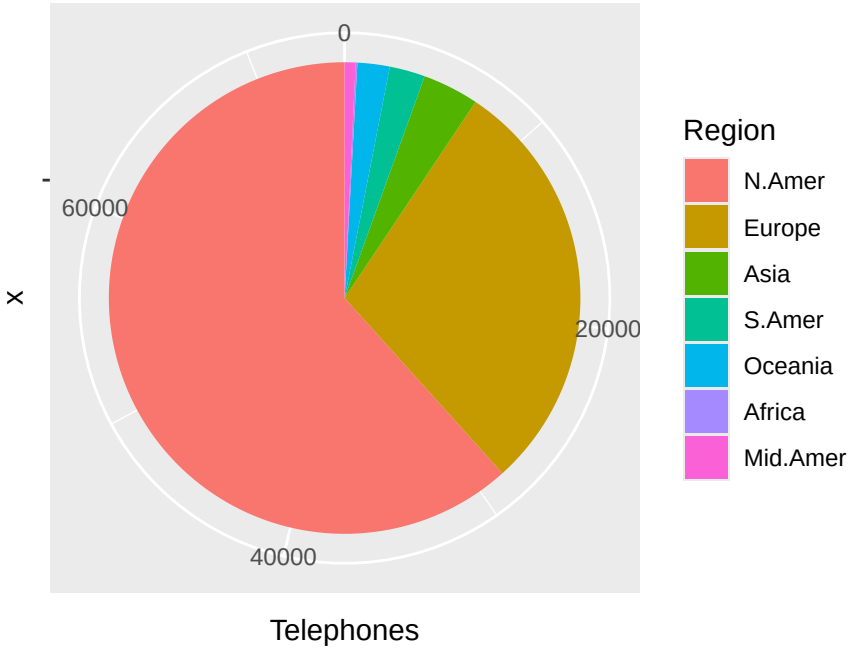
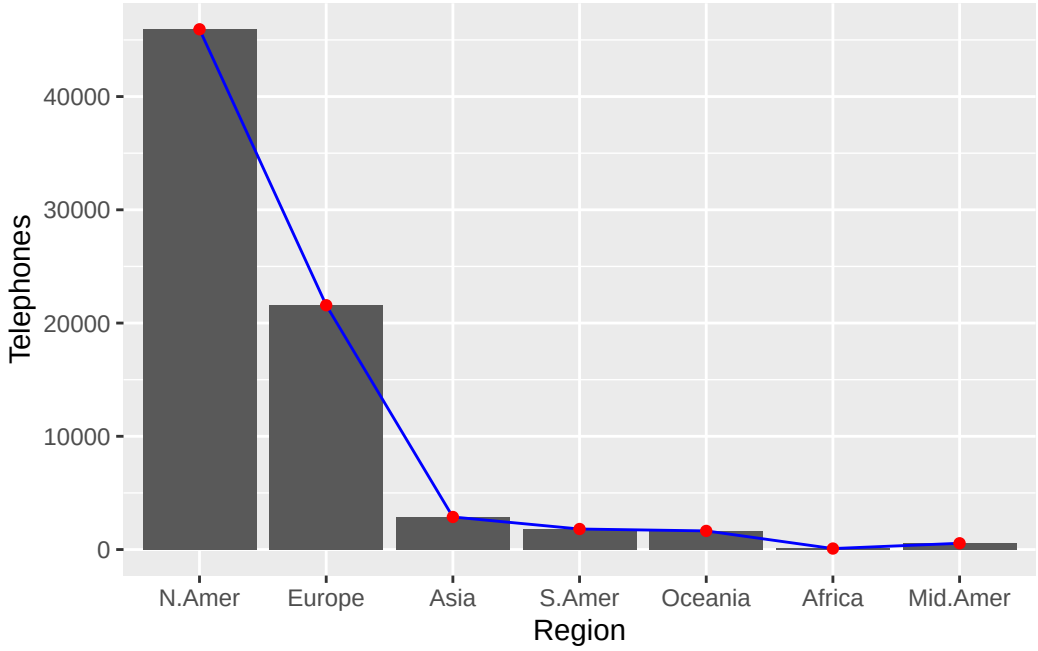


Gráfico de poligono

```
ggplot(phones51, aes(Region, Telephones)) +
geom_col() +
geom_line(col = "blue", aes(x = as.numeric(Region))) +
geom_point(col = "red")
```



Boxplot con 'ggplot2'

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) + geom_boxplot()
```

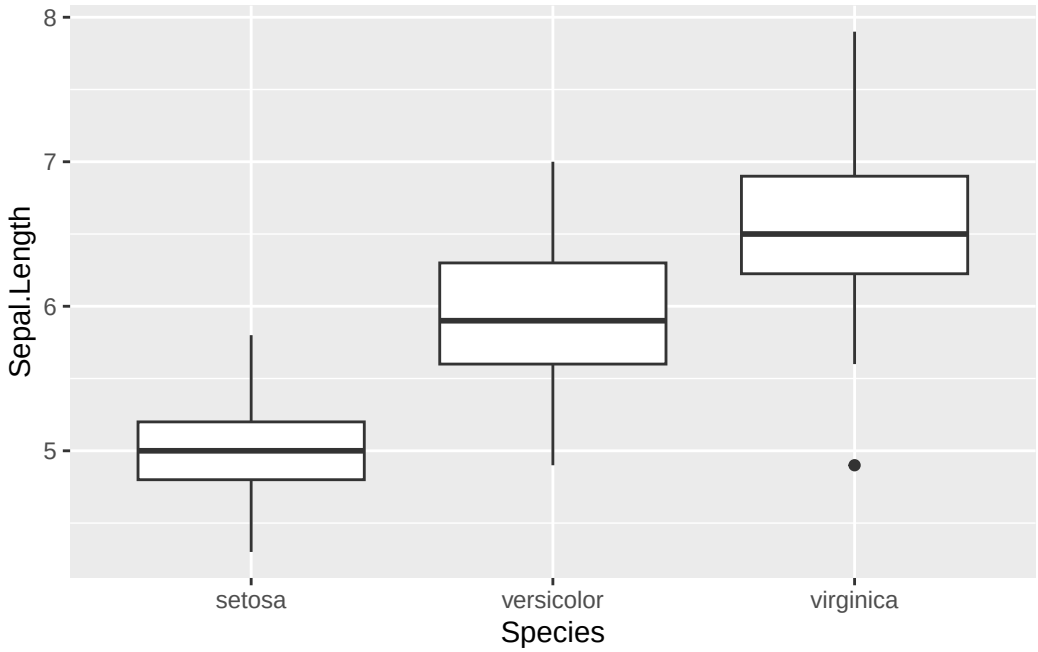
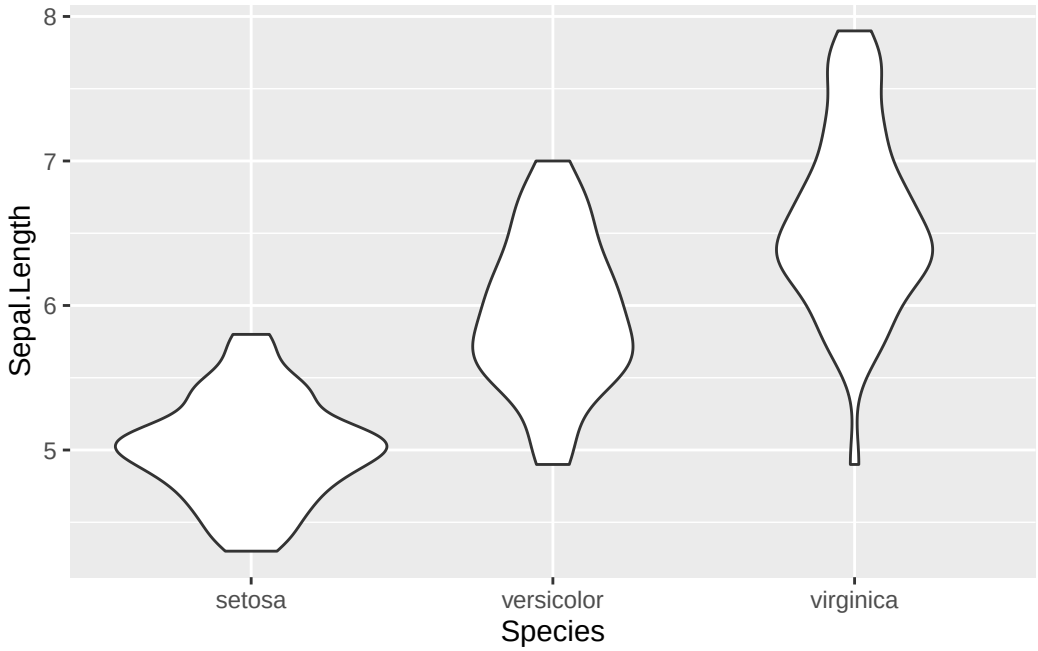


Gráfico de violin con 'ggplot2'

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) + geom_violin()
```



Paleta de colores

```
str(colors())
```

```
 chr [1:657] "white" "aliceblue" "antiquewhite" "antiquewhite1" ...
```

```
palette.pals()
```

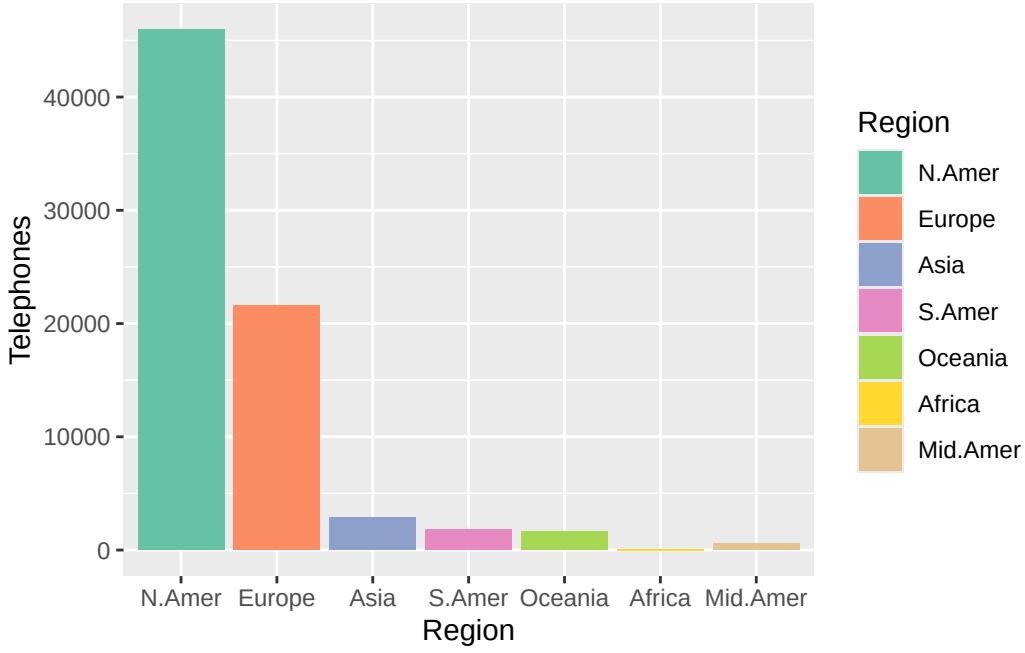
```
 [1] "R3"              "R4"              "ggplot2"         "Okabe-Ito"
 [5] "Accent"          "Dark 2"          "Paired"          "Pastel 1"
 [9] "Pastel 2"        "Set 1"           "Set 2"           "Set 3"
[13] "Tableau 10"      "Classic Tableau" "Polychrome 36"   "Alphabet"
```

```
palette()
```

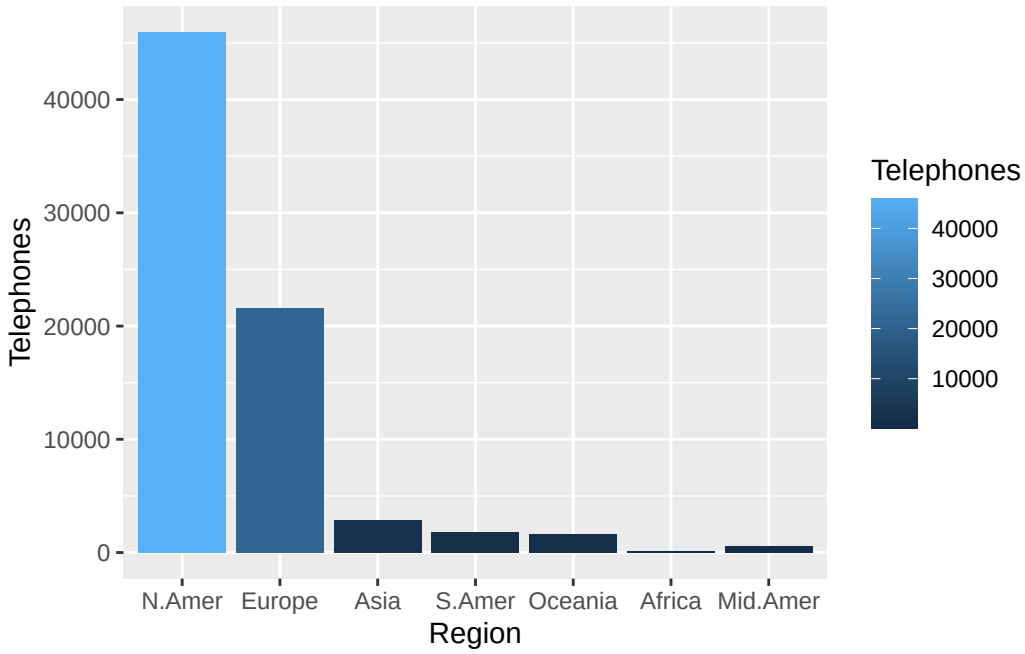
```
[1] "black"   "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC" "#F5C710"
[8] "gray62"
```

Personalizando gráficos con la paleta de colores

```
ggplot(phones51, aes(Region, Telephones, fill = Region)) +
geom_col() +
scale_fill_brewer(palette = "Set2")
```



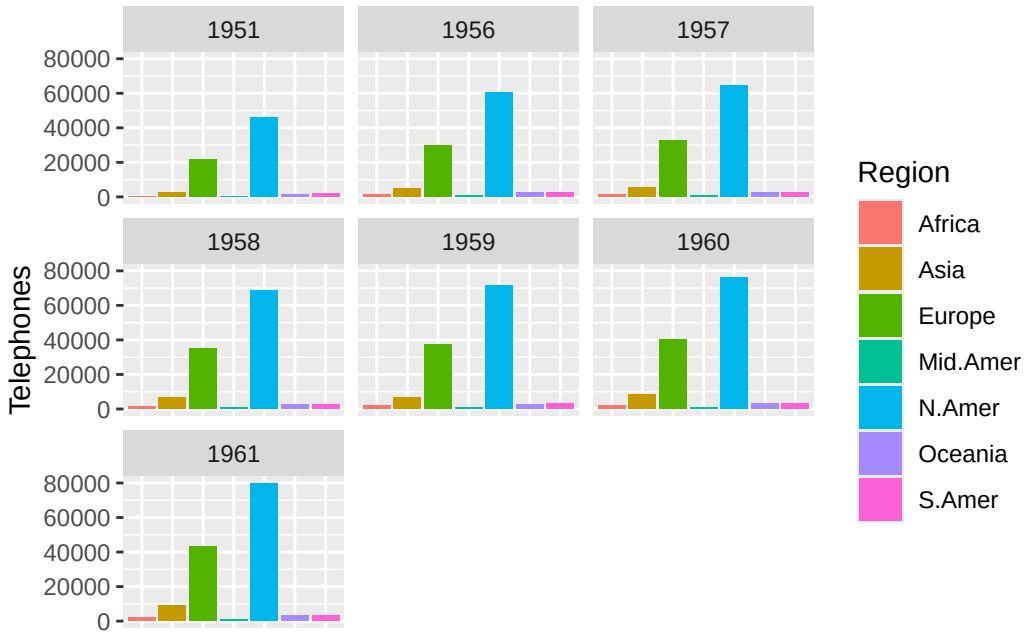

```
ggplot(phones51, aes(Region, Telephones, fill = Telephones)) +
geom_col()
```



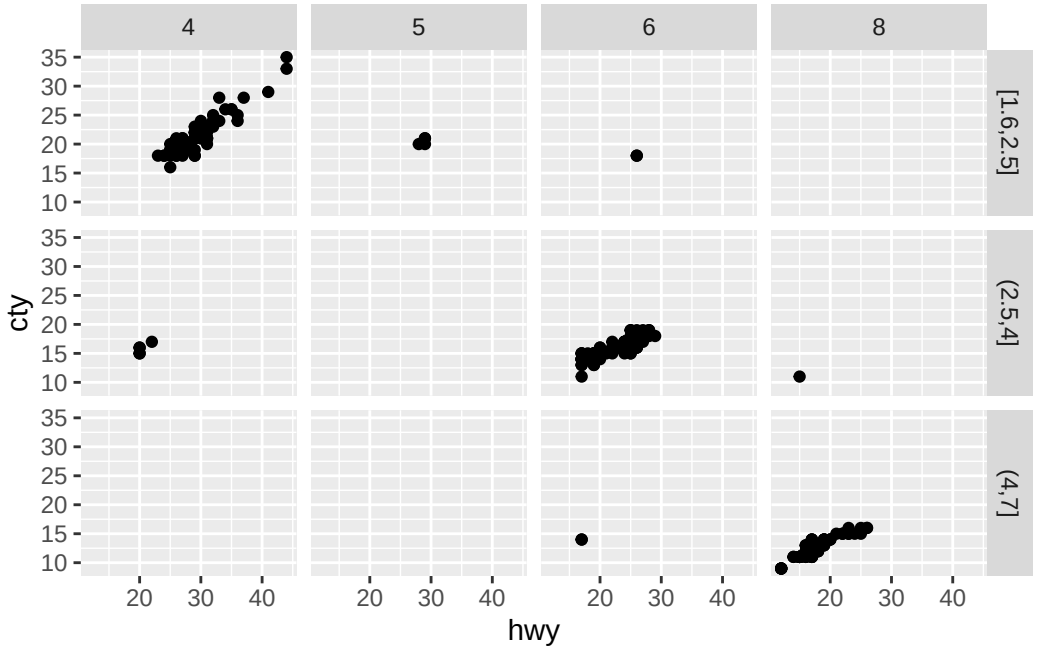
Subdivisión de datos y gráficos separados (Faceting)

Es una técnica utilizada en visualización de datos para analizar y mostrar relaciones complejas entre múltiples variables. La idea es dividir el conjunto de datos en grupos más pequeños (subconjuntos) basados en los valores de ciertas variables. Luego, se crean gráficos separados para cada subconjunto, mostrando cómo las otras variables se comportan dentro de esos grupos. Esto permite una comparación más clara y detallada de las relaciones entre las variables en diferentes contextos.

```
phones <- data.frame(Year = as.numeric(rep(rownames(WorldPhones), 7)),
Region = rep(colnames(WorldPhones), each = 7),
Telephones = as.numeric(WorldPhones))

ggplot(phones, aes(x = Region, y = Telephones, fill = Region)) +
geom_col() +
facet_wrap(vars(Year)) +
theme(axis.text.x = element_blank(), axis.ticks.x = element_blank()) +
xlab(element_blank())
```

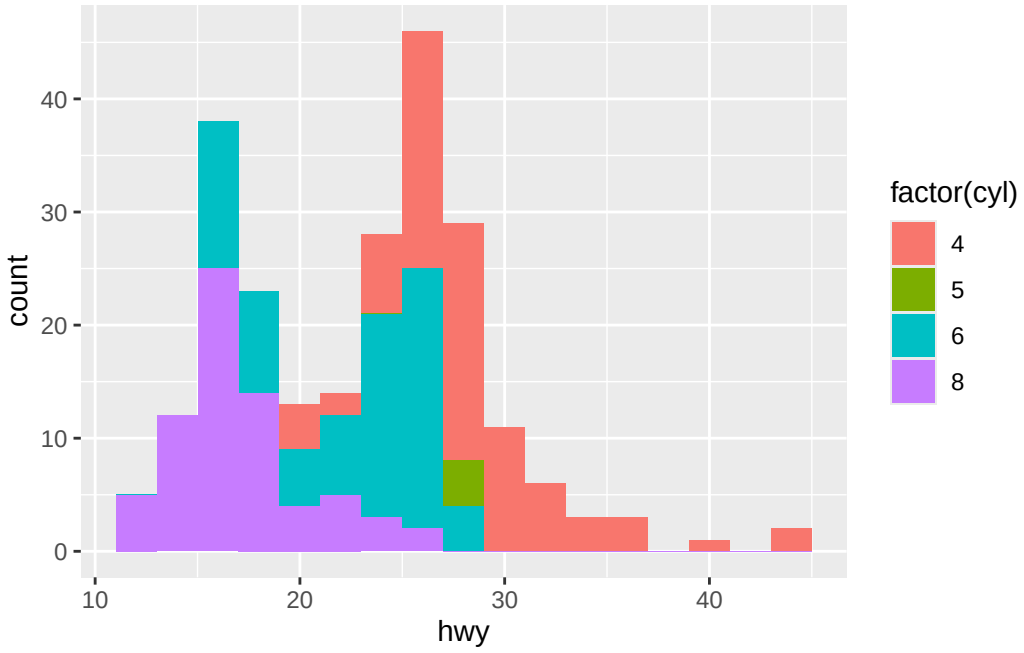



```
ggplot(mpg, aes(hwy, cty)) +
geom_point() +
facet_grid(cut_number(displ, 3) ~ cyl)
```



Gráficos separados en una misma grid

```
ggplot(mpg, aes(hwy, fill = factor(cyl))) +
geom_histogram(binwidth = 2)
```

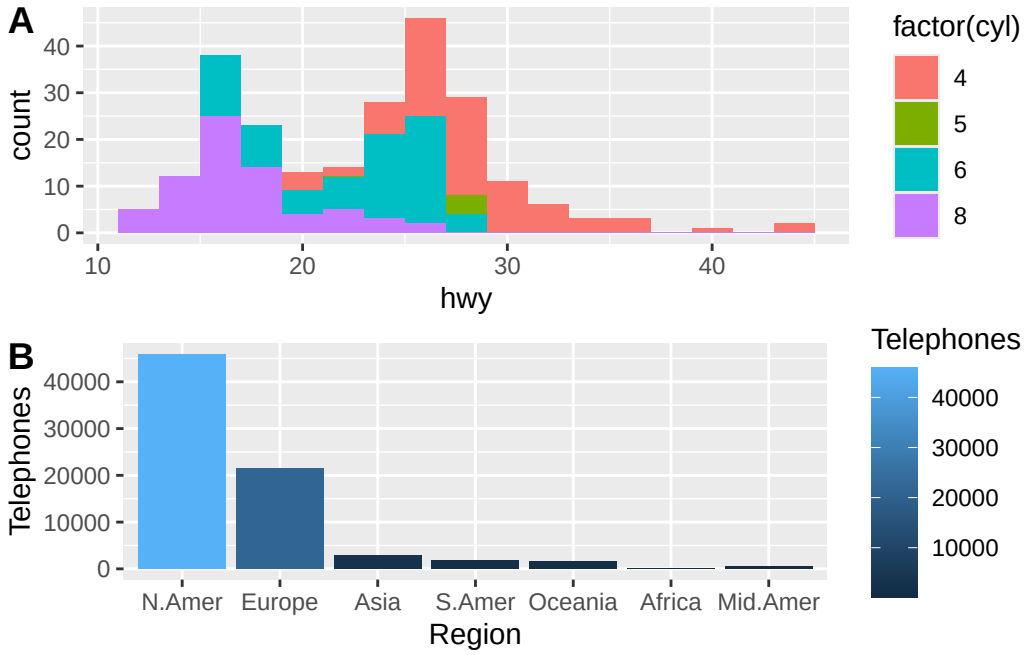


Matrices de gráficos

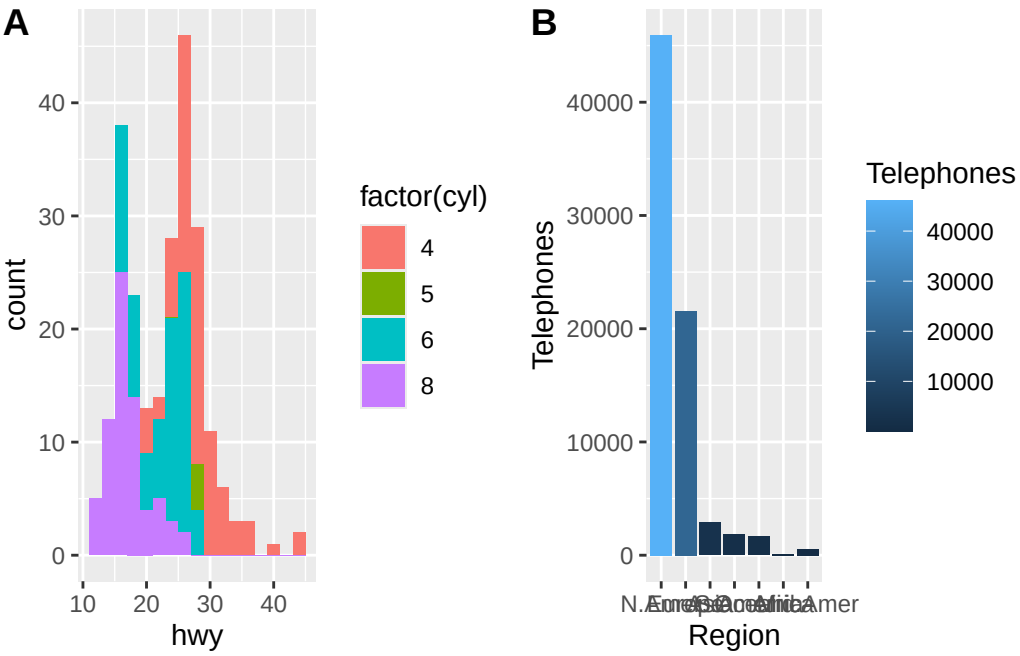
La función `ggarrange` del paquete `ggpubr` en R es una herramienta poderosa para organizar múltiples gráficos creados con `ggplot2` en una disposición de matriz. Esto es especialmente útil cuando se desea presentar varios gráficos de manera conjunta para facilitar la comparación visual y el análisis. Al utilizar `ggarrange`, puedes especificar el número de filas y columnas para la matriz de gráficos, así como ajustar el tamaño y la alineación de cada gráfico dentro del espacio común. Esta función simplifica la creación de paneles de gráficos complejos, permitiendo una presentación más clara y coherente de los datos.

```
library(ggpubr)

p1 <- ggplot(mpg, aes(hwy, fill = factor(cyl))) + geom_histogram(binwidth = 2)
p2 <- ggplot(phones51, aes(Region, Telephones, fill = Telephones)) +
geom_col()
```

```
ggarrange(p1, p2,
          labels = c("A", "B"),
          ncol = 1, nrow = 2)
```




```
ggarrange(p1, p2,
          labels = c("A", "B"),
          ncol = 2, nrow = 1)
```



SESIÓN 2: aplicación a un proyecto de Data Science

Dataset *Breast Cancer Diagnostic*

Este conjunto de datos proviene del estudio **Wisconsin Diagnostic Breast Cancer (WDBC)**, desarrollado para asistir en el diagnóstico médico de **tumores mamarios**. A través de imágenes digitalizadas obtenidas por **aspiración con aguja fina (FNA)** de masas mamarias, se segmentaron núcleos celulares y se calcularon automáticamente una serie de características geométricas y texturales. La información oficial esta presente en el siguiente Link 1 y en Link 2.

Objetivo del estudio

El objetivo es construir modelos predictivos que clasifiquen de forma automática los tumores como **malignos** o **benignos**, utilizando exclusivamente variables cuantitativas derivadas de imágenes médicas. Esto tiene aplicaciones clínicas relevantes, al permitir un diagnóstico temprano, no invasivo y respaldado por evidencia computacional.

Variables del dataset

Cada muestra corresponde a una imagen de tejido mamario. A partir de cada imagen se extrajeron **30 variables explicativas continuas**, agrupadas de la siguiente manera:

1. **Media de características** (`mean`)
2. **Error estándar de características** (`se`)
3. **Valor extremo o peor observación** (`worst`)

Las 10 características básicas medidas para cada grupo son:

- `radius`: distancia promedio del centro al borde del núcleo.
- `texture`: desviación estándar de los valores de intensidad.
- `perimeter`, `area`, `smoothness`, `compactness`, `concavity`, `concave points`, `symmetry`, `fractal dimension`.

Combinando 10 características $\times$ 3 estadísticas $\rightarrow$ se obtienen **30 variables numéricas**.

La variable objetivo (`target`) es binaria:

- `0` = **maligno**
- `1` = **benigno**

Enfoque de modelado

Se planea aplicar dos enfoques complementarios para modelar este conjunto de datos:

1. **Regresión logística con regularización**
    - Justificación: modelo interpretativo, robusto frente a multicolinealidad al usar **regularización L1 (Lasso)** o **L2 (Ridge)**.
    - Requiere: escalado de variables, análisis de correlación, posible selección de variables.
2. **Random Forest**
    - Justificación: modelo de árbol no paramétrico que **captura no linealidades** y **interacciones automáticas** entre variables.
    - Proporciona medidas de **importancia de variables** y suele requerir menos preprocesamiento.
    - Se utilizará como referencia de desempeño y para interpretación global del problema.

Análisis exploratorio sugerido

Antes de aplicar los modelos, es recomendable realizar un análisis exploratorio para:

- **Visualizar las distribuciones**: detectar variables sesgadas, multimodales o con valores atípicos.
- **Comparar clases**: usar `boxplots` o `violin plots` para ver cómo se distribuyen las variables según el tipo de tumor.
- **Detectar correlaciones fuertes** entre variables (útil para la regresión logística regularizada).
- **Reducir la dimensión** con PCA: puede ser útil para interpretación y validación visual de la separación de clases.
- **Evaluar la importancia preliminar de las variables** mediante análisis univariado.

Estudio exploraorio gráfico con `ggplot2`

Librerias y carga del dataset

```
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 lubridate 1.9.4     v tibble    3.2.1
v purrr     1.0.4     v tidyr     1.3.1
-- 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 beco
```

```
library(ggthemes)
library(ggrepel)
library(patchwork)
library(GGally)
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
library(FactoMineR)
library(factoextra)
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
# Carga de datos
# Carga de datos
df = read.csv("breast_cancer_data.csv")
head(df)
```

```
  mean.radius mean.texture mean.perimeter mean.area mean.smoothness
1       17.99        10.38         122.80    1001.0         0.11840
2       20.57        17.77         132.90    1326.0         0.08474
3       19.69        21.25         130.00    1203.0         0.10960
4       11.42        20.38          77.58     386.1         0.14250
5       20.29        14.34         135.10    1297.0         0.10030
6       12.45        15.70          82.57     477.1         0.12780
  mean.compactness mean.concavity mean.concave.points mean.symmetry
1          0.27760         0.3001              0.14710        0.2419
2          0.07864         0.0869              0.07017        0.1812
3          0.15990         0.1974              0.12790        0.2069
4          0.28390         0.2414              0.10520        0.2597
5          0.13280         0.1980              0.10430        0.1809
6          0.17000         0.1578              0.08089        0.2087
  mean.fractal.dimension radius.error texture.error perimeter.error area.error
1                0.07871       1.0950        0.9053           8.589     153.40
2                0.05667       0.5435        0.7339           3.398      74.08
3                0.05999       0.7456        0.7869           4.585      94.03
4                0.09744       0.4956        1.1560           3.445      27.23
5                0.05883       0.7572        0.7813           5.438      94.44
6                0.07613       0.3345        0.8902           2.217      27.19
  smoothness.error compactness.error concavity.error concave.points.error
1         0.006399           0.04904         0.05373              0.01587
2         0.005225           0.01308         0.01860              0.01340
3         0.006150           0.04006         0.03832              0.02058
4         0.009110           0.07458         0.05661              0.01867
5         0.011490           0.02461         0.05688              0.01885
6         0.007510           0.03345         0.03672              0.01137
  symmetry.error fractal.dimension.error worst.radius worst.texture
1        0.03003                0.006193        25.38         17.33
2        0.01389                0.003532        24.99         23.41
```

```
3         0.02250                0.004571         23.57          25.53
4         0.05963                0.009208         14.91          26.50
5         0.01756                0.005115         22.54          16.67
6         0.02165                0.005082         15.47          23.75
  worst.perimeter worst.area worst.smoothness worst.compactness worst.concavity
1          184.60     2019.0            0.1622             0.6656          0.7119
2          158.80     1956.0            0.1238             0.1866          0.2416
3          152.50     1709.0            0.1444             0.4245          0.4504
4           98.87      567.7            0.2098             0.8663          0.6869
5          152.20     1575.0            0.1374             0.2050          0.4000
6          103.40      741.6            0.1791             0.5249          0.5355
  worst.concave.points worst.symmetry worst.fractal.dimension target
1               0.2654          0.4601                 0.11890      0
2               0.1860          0.2750                 0.08902      0
3               0.2430          0.3613                 0.08758      0
4               0.2575          0.6638                 0.17300      0
5               0.1625          0.2364                 0.07678      0
6               0.1741          0.3985                 0.12440      0
```

Creación de las columnas de tipo de tumores según areas

```
summary(df$mean.area) ## milimetros
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  143.5   420.3   551.1   654.9   782.7  2501.0
```

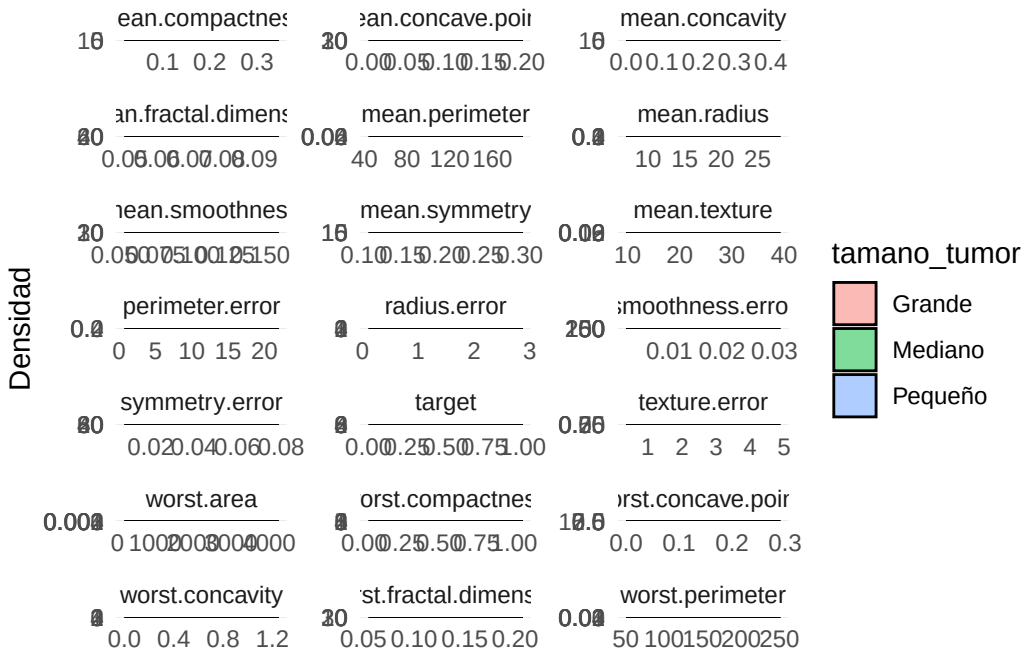
```
#Crear una nueva variable categórica basada en mean.area
df <- df %>%
  mutate(tamano_tumor = case_when(
    mean.area < 420.3 ~ "Pequeño",
    mean.area >= 420.3 & mean.area <= 782.7 ~ "Mediano",
    mean.area > 782.7 ~ "Grande"
  ))

# Verificar los primeros 6 registros
head(df %>% select(mean.area, tamano_tumor))
```

```
  mean.area tamano_tumor
1    1001.0       Grande
2    1326.0       Grande
```

```
3    1203.0       Grande
4     386.1      Pequeño
5    1297.0       Grande
6     477.1      Mediano
```

Densidades de las covariables por tamaño de tumor

```
# 1. Gráfico de distribuciones por tipo de tumor
p1 <-     df %>%
  pivot_longer(cols = -tamano_tumor) %>%
  ggplot(aes(x = value, fill = tamano_tumor)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ name, scales = "free", ncol = 3) +
  labs(title = "Distribuciones por variable y tipo de tumor",
       x = "Valor", y = "Densidad") +
  theme_minimal()

p1
```



Boxplot/violin para 'mean.compactness' por tamaño de tumor

```
# 2. Boxplots para comparar grupos en una variable
p2 <-   df %>%
  ggplot(aes(x = tamano_tumor, y = mean.compactness, fill = tamano_tumor)) +
  geom_violin(trim = FALSE, alpha = 0.5) +
  geom_boxplot(width = 0.1, outlier.shape = NA) +
  labs(title = "Distribución de 'mean.compactness' según tipo de tumor",
       y = "mean.compactness", x = "Tipo de tumor") +
  theme_classic()

p2
```

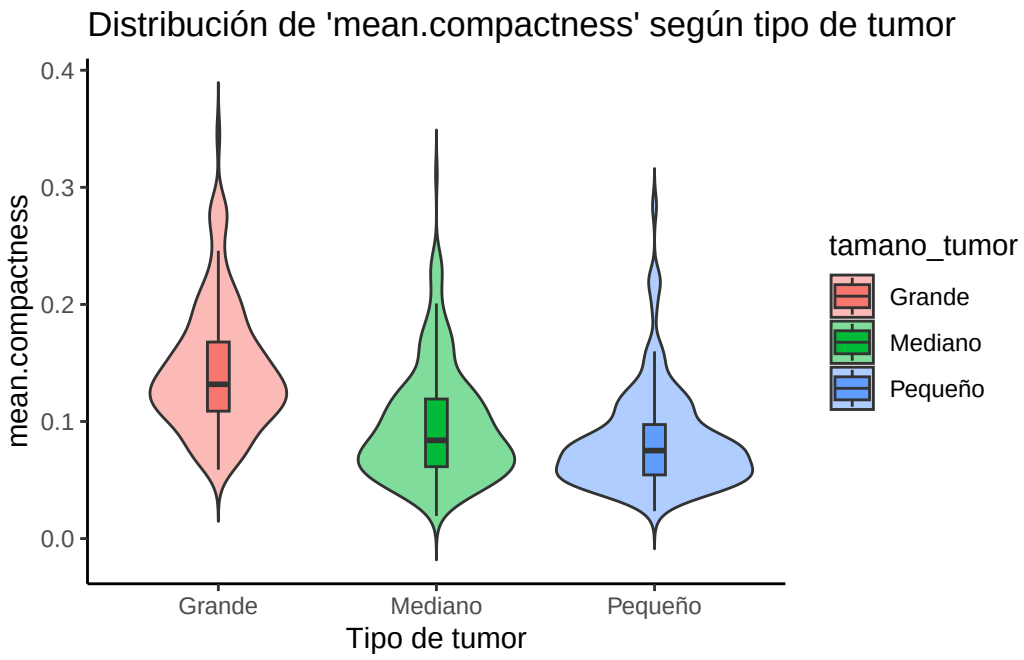


Diagrama de puntos de los Principal Components Analysis según tamaño de tumor

```
# 3. PCA con visualización de clases
# Asumiendo que solo columnas numéricas están en cols 2:11
df_pca <- df %>% select(-tamano_tumor)
pca_result <- PCA(df_pca, graph = FALSE)

# Agregamos clase a resultados para graficar
```

```r
pca_df <- data.frame(pca_result$ind$coord[, 1:2]) %>%
  mutate(diagnosis = df$tamano_tumor)

p3 <- ggplot(pca_df, aes(x = Dim.1, y = Dim.2, color = diagnosis)) +
  geom_point(alpha = 0.6) +
  labs(title = "PCA: Visualización de clases en 2D",
       x = "PC1", y = "PC2") +
  theme_minimal()

p3
```

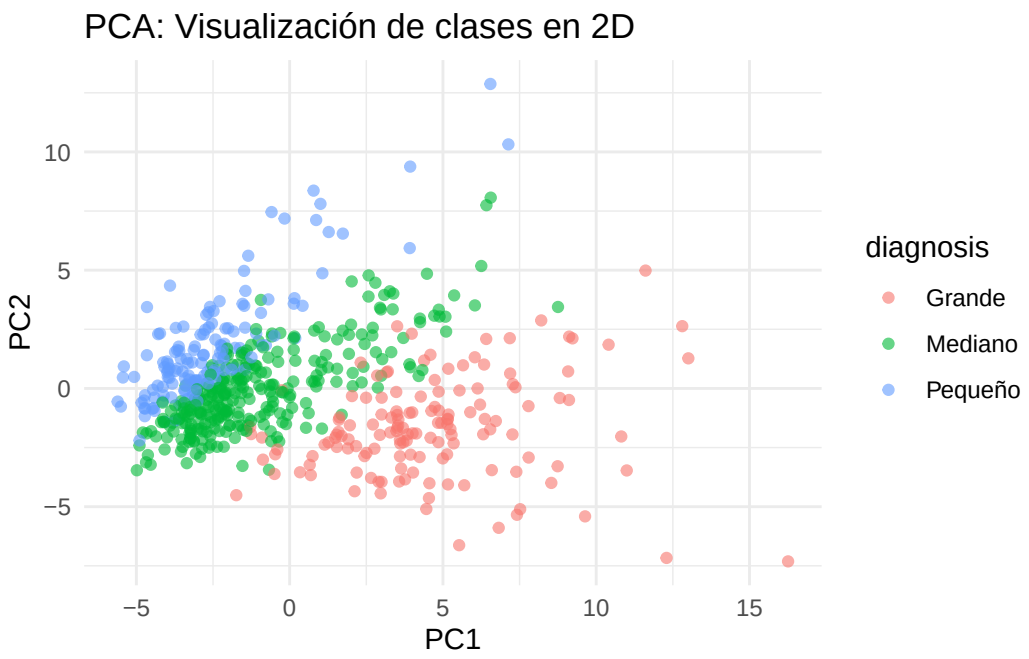


Gráfico de calor para la correlación de todas las variables númericas

```r
# Cargar paquetes necesarios
library(tidyverse)
library(reshape2)  # Para convertir la matriz de correlación en formato largo
```

```
Adjuntando el paquete: 'reshape2'
```

```
The following object is masked from 'package:tidyr':

    smiths
```

```r
# Seleccionar solo variables numéricas
df_numeric <- df %>% select(where(is.numeric))

# Calcular la matriz de correlación
cor_matrix <- cor(df_numeric, use = "complete.obs")

# Convertir la matriz en formato largo para ggplot2
cor_data <- melt(cor_matrix)

# Crear el gráfico de calor
p4 <- ggplot(cor_data, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlación") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  labs(title = "Mapa de calor de correlaciones",
       x = "", y = "")

p4
```