Unidad: Introducción a R

Tidyverse: Parte 1

Nicolás Sidicaro
Marzo 2025

¿Qué es Tidyverse?

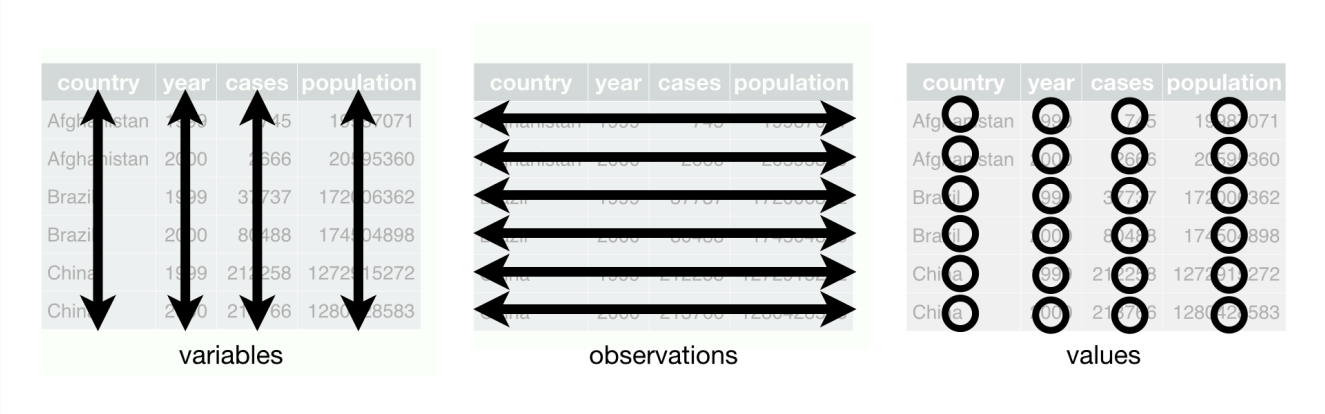
- **Tidyverse** es una colección de paquetes de R diseñados para ciencia de datos
- Desarrollado principalmente por **Hadley Wickham** y su equipo en RStudio (ahora Posit)
- Comparte una filosofía común, gramática y estructuras de datos
- Facilita enormemente el flujo de trabajo de análisis de datos

> "El Tidyverse es un conjunto de paquetes que trabajan en armonía porque comparten convenciones comunes de datos y API"

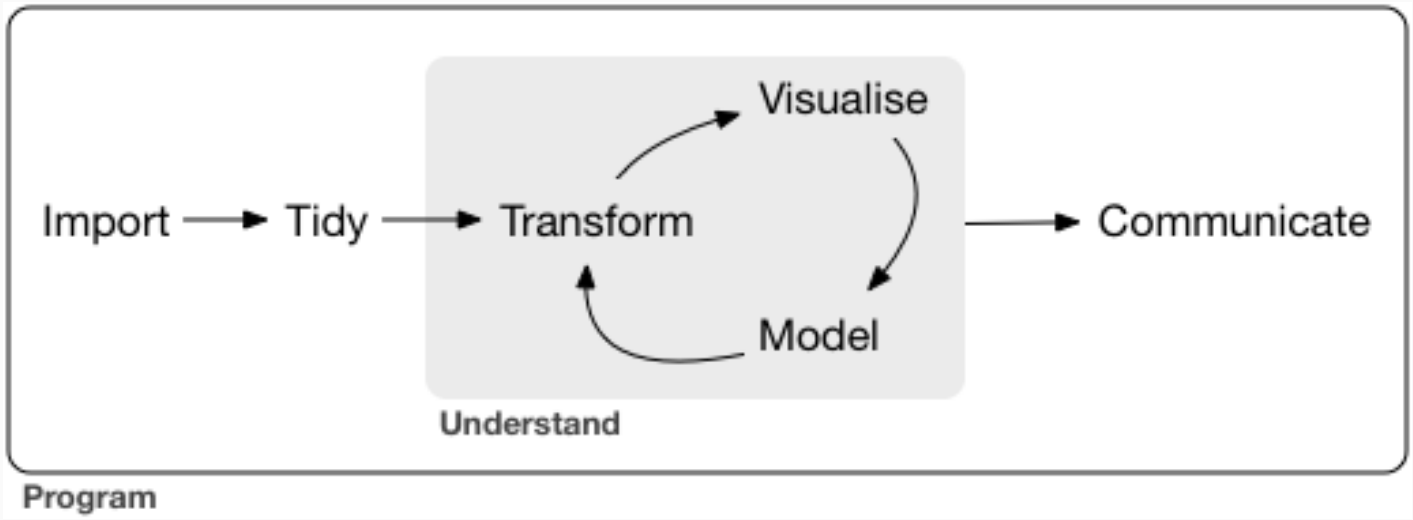
Filosofía del Tidyverse

La filosofía del Tidyverse se basa en el concepto de "tidy data":

- Cada variable forma una columna
- Cada observación forma una fila
- Cada tipo de unidad observacional forma una tabla



El Flujo de Trabajo en Ciencia de Datos



El **ciclo de la ciencia de datos** con Tidyverse:

1. **Importar** datos (readr, haven, jsonlite)
2. **Ordenar** datos (tidyr)
3. **Transformar** datos (dplyr)
4. **Visualizar** datos (ggplot2)
5. **Modelar** datos (modelr, caret)
6. **Comunicar** resultados (rmarkdown, shiny)

Principales Paquetes del Tidyverse

```
# Cargar todo el tidyverse con un solo comando
library(tidyverse)
```

Los principales paquetes incluyen:

- **ggplot2**: visualización de datos
- **dplyr**: manipulación de datos
- **tidyr**: ordenar datos
- **readr**: importación de datos
- **purrr**: programación funcional
- **tibble**: versión moderna de data frames
- **stringr**: manipulación de strings
- **forcats**: manejo de factores

Instalación del Tidyverse

Para instalar el Tidyverse completo:

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

O instalar paquetes individuales:

```
install.packages("dplyr")
install.packages("ggplot2")
```

Para cargar un paquete individual:

```
library(dplyr)
```

Chaining con el Operador Pipe %>%

El operador pipe (`%>%`) es fundamental en Tidyverse:

- Proviene del paquete **magrittr**
- Permite encadenar operaciones
- Hace que el código sea más legible y mantenible
- **Sintaxis**: `datos %>% función()`
- Equivale a: `función(datos)`

El valor de la izquierda se convierte en el primer argumento de la función de la derecha.

Ejemplos Básicos del Operador Pipe

Sin pipe:

```
resultado ← funcion3(funcion2(funcion1(datos, arg1), arg2), arg3)
```

Con pipe:

```
resultado ← datos %>%
  funcion1(arg1) %>%
  funcion2(arg2) %>%
  funcion3(arg3)
```

Más legible, más fácil de entender y modificar.

Tibbles: Data Frames Modernos

Los **tibbles** son una versión moderna de los data frames tradicionales:

- Más consistentes y estrictos que los data frames
- Imprimen mejor (muestran solo las primeras filas)
- No convierten automáticamente strings a factores
- No usan los nombres de filas como identificadores
- Muestran el tipo de cada columna

```
# Crear un tibble
tibble(
  x = 1:5,
  y = c("a", "b", "c", "d", "e"),
  z = x^2
)
```

```
## # A tibble: 5 × 3
##       x y         z
##   <int> <chr> <dbl>
## 1     1 a         1
## 2     2 b         4
## 3     3 c         9
## 4     4 d        16
## 5     5 e        25
```

Conversión entre Data Frames y Tibbles

Convertir un data frame en tibble:

```
# Data frame tradicional
df ← data.frame(x = 1:3, y = c("a", "b", "c"))

# Convertir a tibble
df_tbl ← as_tibble(df)
df_tbl
```

```
## # A tibble: 3 × 2
##       x y
##   <int> <chr>
## 1     1 a
## 2     2 b
## 3     3 c
```

Conversión entre Data Frames y Tibbles

Convertir un tibble en data frame:

```
# Convertir de vuelta a data frame
as.data.frame(df_tbl)
```

```
##   x y
## 1 1 a
## 2 2 b
## 3 3 c
```

Importación de Datos con readr

El paquete **readr** facilita la importación de datos:

```
# Importar CSV
datos_csv ← read_csv("datos.csv")

# Importar TSV (valores separados por tabulaciones)
datos_tsv ← read_tsv("datos.tsv")

# Importar archivos delimitados por otros caracteres
datos_delim ← read_delim("datos.txt", delim = "|")
```

Ventajas:

- Más rápido que las funciones base de R
- No convierte strings a factores
- Devuelve tibbles
- Infiere automáticamente el tipo de datos

Lectura de Otros Formatos

El ecosistema Tidyverse ofrece otros paquetes para importar datos:

```
# Excel (readxl)
library(readxl)
datos_excel ← read_excel("datos.xlsx", sheet = "Hoja1")

# SPSS, Stata, SAS (haven)
library(haven)
datos_spss ← read_spss("datos.sav")
datos_stata ← read_dta("datos.dta")
datos_sas ← read_sas("datos.sas7bdat")

# Datos web (rvest)
library(rvest)
pagina ← read_html("https://ejemplo.com")
```

Exportación de Datos

```r
# Guardar en CSV
write_csv(datos, "datos_procesados.csv")

# Guardar en TSV
write_tsv(datos, "datos_procesados.tsv")

# Guardar en RDS (formato nativo de R)
write_rds(datos, "datos_procesados.rds")

# Guardar en Excel (requiere writexl)
library(writexl)
write_xlsx(datos, "datos_procesados.xlsx")
```

Primeros Pasos con dplyr

dplyr es el paquete principal para manipulación de datos:

Las "cinco verbos" principales:

- `filter()`: filtrar filas según condiciones
- `select()`: seleccionar columnas
- `mutate()`: crear o transformar variables
- `arrange()`: ordenar filas
- `summarise()`: resumir datos

Primeros Pasos con dplyr

```r
# Dataset de ejemplo
mtcars_tbl ← as_tibble(mtcars, rownames = "car")
glimpse(mtcars_tbl)
```

```
## Rows: 32
## Columns: 12
## $ car  <chr> "Mazda RX4", "Mazda RX4 Wag", "Datsun 710", …
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24…
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8,…
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 36…
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 12…
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.…
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.…
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15…
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,…
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3,…
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4,…
```

filter(): Filtrar Filas

Utilizado para filtrar filas según condiciones lógicas:

```
# Filtrar coches con más de 6 cilindros y menos de 4000 lbs
mtcars_tbl %>%
  filter(cyl > 6, wt < 4)
```

```
## # A tibble: 10 × 12
##    car          mpg   cyl  disp    hp  drat    wt  qsec    vs
##    <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Hornet …  18.7      8  360    175  3.15  3.44  17.0     0
##  2 Duster …  14.3      8  360    245  3.21  3.57  15.8     0
##  3 Merc 45…  17.3      8  276.   180  3.07  3.73  17.6     0
##  4 Merc 45…  15.2      8  276.   180  3.07  3.78  18       0
##  5 Dodge C…  15.5      8  318    150  2.76  3.52  16.9     0
##  6 AMC Jav…  15.2      8  304    150  3.15  3.44  17.3     0
##  7 Camaro …  13.3      8  350    245  3.73  3.84  15.4     0
##  8 Pontiac…  19.2      8  400    175  3.08  3.84  17.0     0
##  9 Ford Pa…  15.8      8  351    264  4.22  3.17  14.5     0
## 10 Maserat…  15        8  301    335  3.54  3.57  14.6     0
## # ℹ 3 more variables: am <dbl>, gear <dbl>, carb <dbl>
```

filter(): Filtrar Filas

Operadores lógicos:

- `&` o `,` (equivalentes en filter): Y lógico
- `|` : O lógico
- `!` : Negación
- ==: Igualdad
- !=: Desigualdad

select(): Seleccionar Columnas

Para seleccionar subconjuntos de columnas:

```
# Seleccionar solo algunas variables
mtcars_tbl %>%
  select(car, mpg, hp, cyl)
```

```
## # A tibble: 32 × 4
##    car                 mpg    hp   cyl
##    <chr>             <dbl> <dbl> <dbl>
##  1 Mazda RX4          21     110     6
##  2 Mazda RX4 Wag      21     110     6
##  3 Datsun 710         22.8    93     4
##  4 Hornet 4 Drive     21.4   110     6
##  5 Hornet Sportabout  18.7   175     8
##  6 Valiant            18.1   105     6
##  7 Duster 360         14.3   245     8
##  8 Merc 240D          24.4    62     4
##  9 Merc 230           22.8    95     4
## 10 Merc 280           19.2   123     6
## # ℹ 22 more rows
```

select(): Seleccionar Columnas

Funciones auxiliares:

- `starts_with()`, `ends_with()`, `contains()`
- `everything()`: todas las demás columnas
- `-columna`: excluir columna

select(): Más Ejemplos

```
# Seleccionar columnas por patrón
mtcars_tbl %>%
  select(car, starts_with("c"), contains("p"))
```

```
## # A tibble: 32 × 6
##    car                 cyl  carb   mpg  disp    hp
##    <chr>             <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Mazda RX4             6     4  21    160    110
##  2 Mazda RX4 Wag         6     4  21    160    110
##  3 Datsun 710            4     1  22.8  108     93
##  4 Hornet 4 Drive        6     1  21.4  258    110
##  5 Hornet Sportabout     8     2  18.7  360    175
##  6 Valiant               6     1  18.1  225    105
##  7 Duster 360            8     4  14.3  360    245
##  8 Merc 240D             4     2  24.4  147.    62
##  9 Merc 230              4     2  22.8  141.    95
## 10 Merc 280              6     4  19.2  168.   123
## # ℹ 22 more rows
```

select(): Más Ejemplos

```
# Reordenar columnas
mtcars_tbl %>%
  select(car, mpg, everything())
```

```
## # A tibble: 32 × 12
##    car          mpg   cyl  disp    hp  drat    wt  qsec    vs
##    <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Mazda R…  21       6  160    110  3.9   2.62  16.5     0
##  2 Mazda R…  21       6  160    110  3.9   2.88  17.0     0
##  3 Datsun …  22.8     4  108     93  3.85  2.32  18.6     1
##  4 Hornet …  21.4     6  258    110  3.08  3.22  19.4     1
##  5 Hornet …  18.7     8  360    175  3.15  3.44  17.0     0
##  6 Valiant   18.1     6  225    105  2.76  3.46  20.2     1
##  7 Duster …  14.3     8  360    245  3.21  3.57  15.8     0
##  8 Merc 24…  24.4     4  147.    62  3.69  3.19  20       1
##  9 Merc 230  22.8     4  141.    95  3.92  3.15  22.9     1
## 10 Merc 280  19.2     6  168.   123  3.92  3.44  18.3     1
## # ℹ 22 more rows
## # ℹ 3 more variables: am <dbl>, gear <dbl>, carb <dbl>
```

mutate(): Crear o Transformar Variables

Para crear nuevas columnas o modificar existentes:

```
# Crear nuevas variables
mtcars_tbl %>%
  select(car, mpg, hp, wt) %>%
  mutate(
    kpl = mpg * 0.425,  # Convertir millas/galón a km/litro
    hp_por_peso = hp / wt  # Potencia por peso
  ) %>%
  head(5)
```

```
## # A tibble: 5 × 6
##   car                mpg    hp    wt   kpl hp_por_peso
##   <chr>            <dbl> <dbl> <dbl> <dbl>       <dbl>
## 1 Mazda RX4         21     110  2.62  8.92        42.0
## 2 Mazda RX4 Wag     21     110  2.88  8.92        38.3
## 3 Datsun 710        22.8    93  2.32  9.69        40.1
## 4 Hornet 4 Drive    21.4   110  3.22  9.09        34.2
## 5 Hornet Sportabout 18.7   175  3.44  7.95        50.9
```

arrange(): Ordenar Filas

```
# Ordenar por mpg (descendente) y hp (ascendente)
mtcars_tbl %>%
  select(car, mpg, hp) %>%
  arrange(desc(mpg), hp) %>%
  head(5)
```

```
## # A tibble: 5 × 3
##   car              mpg    hp
##   <chr>          <dbl> <dbl>
## 1 Toyota Corolla  33.9    65
## 2 Fiat 128        32.4    66
## 3 Honda Civic     30.4    52
## 4 Lotus Europa    30.4   113
## 5 Fiat X1-9       27.3    66
```

- Por defecto, ordena de forma ascendente
- `desc()` para orden descendente
- Se pueden combinar múltiples criterios

summarise(): Resumir Datos

Para calcular estadísticas resumidas:

```
# Calcular estadísticas resumidas
mtcars_tbl %>%
  summarise(
    mpg_promedio = mean(mpg),
    hp_maximo = max(hp),
    n_coches = n()
  )
```

```
## # A tibble: 1 × 3
##   mpg_promedio hp_maximo n_coches
##          <dbl>     <dbl>    <int>
## 1         20.1       335       32
```

- Reduce el dataframe a una sola fila
- Ideal para estadísticas descriptivas
- Se combina frecuentemente con `group_by()`

Operaciones por Grupo con group_by()

Permite realizar operaciones por grupos:

```
# Estadísticas por número de cilindros
mtcars_tbl %>%
  group_by(cyl) %>%
  summarise(
    n_coches = n(),
    mpg_promedio = mean(mpg),
    hp_promedio = mean(hp)
  )
```

```
## # A tibble: 3 × 4
##     cyl n_coches mpg_promedio hp_promedio
##   <dbl>    <int>        <dbl>       <dbl>
## 1     4       11         26.7        82.6
## 2     6        7         19.7       122.
## 3     8       14         15.1       209.
```

Combinación de Funciones: Ejemplo

```
agrup ← mtcars_tbl %>%
  # Filtrar coches automáticos
  filter(am == 0) %>%
  # Agrupar por cilindros y engranajes
  group_by(cyl, gear) %>%
  # Calcular estadísticas
  summarise(
    n_coches = n(),
    mpg_promedio = mean(mpg),
    hp_promedio = mean(hp),
    .groups = "drop"  # Eliminar agrupación
  ) %>%
  # Ordenar por número de cilindros y engranajes
  arrange(cyl, gear)
```

Combinación de Funciones: Ejemplo

```
print(agrup)
```

```
## # A tibble: 5 × 5
##     cyl  gear n_coches mpg_promedio hp_promedio
##   <dbl> <dbl>    <int>        <dbl>       <dbl>
## 1     4     3        1         21.5        97
## 2     4     4        2         23.6        78.5
## 3     6     3        2         19.8       108.
## 4     6     4        2         18.5       123
## 5     8     3       12         15.0       194.
```

Funciones Útiles en dplyr

- `distinct()`: eliminar filas duplicadas
- `count()`: contar observaciones por grupo
- `slice()`: seleccionar filas por posición
- `pull()`: extraer una columna como vector

```
# Contar coches por número de cilindros
mtcars_tbl %>%
  count(cyl, sort = TRUE)
```

```
## # A tibble: 3 × 2
##     cyl     n
##   <dbl> <int>
## 1     8    14
## 2     4    11
## 3     6     7
```

Funciones Útiles en dplyr

- `distinct()`: eliminar filas duplicadas
- `count()`: contar observaciones por grupo
- `slice()`: seleccionar filas por posición
- `pull()`: extraer una columna como vector

```
# Seleccionar las primeras 2 filas de cada grupo
mtcars_tbl %>%
  group_by(cyl) %>%
  slice_head(n = 2)
```

```
## # A tibble: 6 × 12
## # Groups:   cyl [3]
##   car          mpg   cyl  disp    hp  drat    wt  qsec    vs
##   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Datsun 7…  22.8     4  108      93  3.85  2.32  18.6     1
## 2 Merc 240D  24.4     4  147.     62  3.69  3.19  20       1
## 3 Mazda RX4  21       6  160     110  3.9   2.62  16.5     0
## 4 Mazda RX…  21       6  160     110  3.9   2.88  17.0     0
## 5 Hornet S…  18.7     8  360     175  3.15  3.44  17.0     0
## 6 Duster 3…  14.3     8  360     245  3.21  3.57  15.8     0
## # ℹ 3 more variables: am <dbl>, gear <dbl>, carb <dbl>
```

Manipulación de Strings con stringr

El paquete **stringr** facilita la manipulación de cadenas de texto:

```
# Vector de ejemplo
frutas ← c("manzana", "BANANA", "pera", "naranja")

# Detectar patrón
str_detect(frutas, "an")
```

```
## [1]  TRUE FALSE FALSE  TRUE
```

```
# Extraer parte de una cadena
str_extract(frutas, "an")
```

```
## [1] "an" NA   NA   "an"
```

```
# Reemplazar patrón
str_replace(frutas, "an", "AN")
```

```
## [1] "mANzana" "BANANA"  "pera"    "narANja"
```

Más Funciones de stringr

```
# Convertir a mayúsculas/minúsculas
str_to_upper(frutas)
```

```
## [1] "MANZANA" "BANANA"  "PERA"    "NARANJA"
```

```
str_to_lower(frutas)
```

```
## [1] "manzana" "banana"  "pera"    "naranja"
```

```
# Longitud de las cadenas
str_length(frutas)
```

```
## [1] 7 6 4 7
```

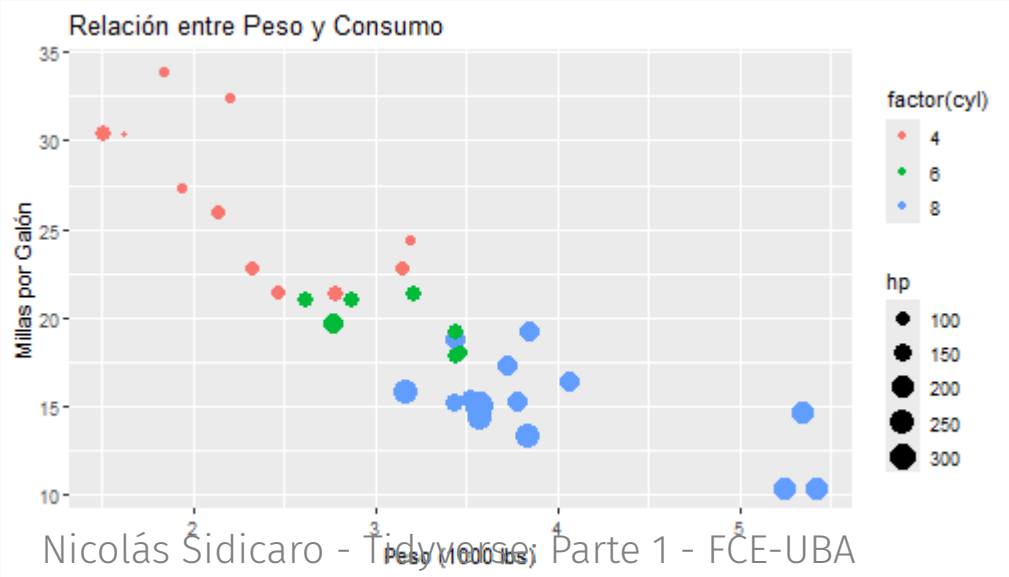
```
# Unir cadenas
str_c(frutas, " fresca", sep = "")
```

```
## [1] "manzana fresca" "BANANA fresca"  "pera fresca"
## [4] "naranja fresca"
```

Visualización con ggplot2

ggplot2 implementa la "Gramática de Gráficos":

```
# Gráfico básico de dispersión
ggplot(mtcars_tbl, aes(x = wt, y = mpg)) +
  geom_point(aes(color = factor(cyl), size = hp)) +
  labs(
    title = "Relación entre Peso y Consumo",
    x = "Peso (1000 lbs)",
    y = "Millas por Galón"
  )
```



Estructura de ggplot2

Un gráfico en ggplot2 tiene tres componentes principales:

1. **Datos**: el dataframe que contiene los datos
2. **Estética** (aes): mapeo de variables a propiedades visuales
3. **Geometrías** (geom): formas visuales que representan los datos

```
ggplot(datos, aes(x = var1, y = var2)) +
  geom_punto() +
  otras_capas()
```

Tipos de Gráficos Comunes en ggplot2

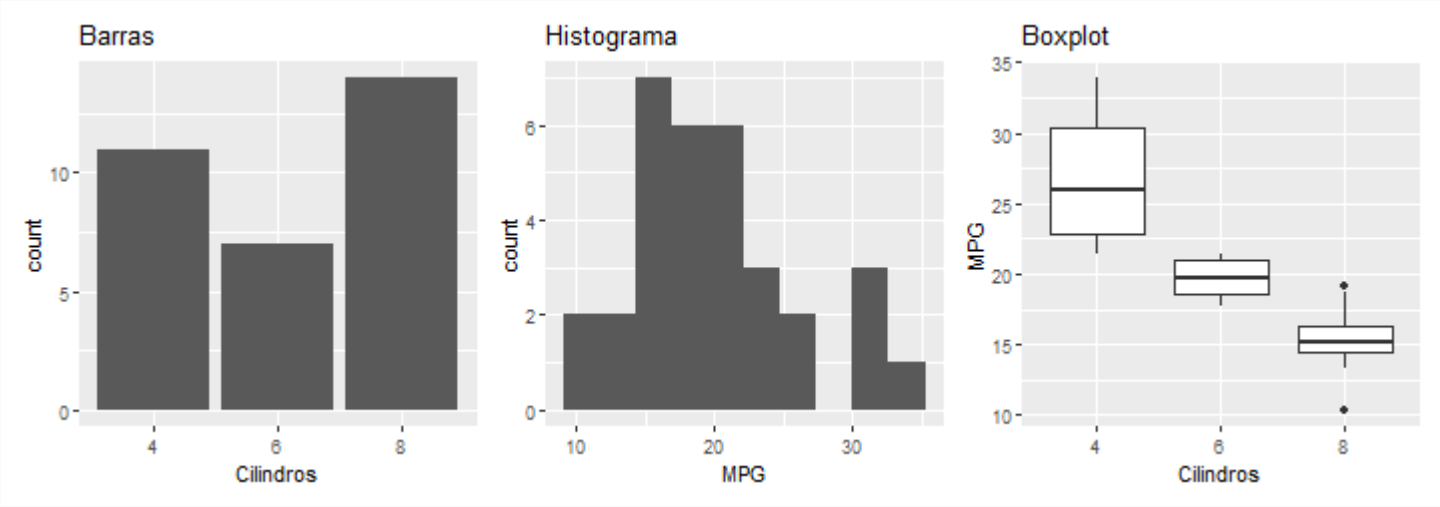
```
p1 ← ggplot(mtcars_tbl, aes(x = factor(cyl))) +
  geom_bar() +
  labs(title = "Barras", x = "Cilindros")

p2 ← ggplot(mtcars_tbl, aes(x = mpg)) +
  geom_histogram(bins = 10) +
  labs(title = "Histograma", x = "MPG")

p3 ← ggplot(mtcars_tbl, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot() +
  labs(title = "Boxplot", x = "Cilindros", y = "MPG")
```

Tipos de Gráficos Comunes en ggplot2

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version
## 4.4.1
```

```
p1 + p2 + p3
```



Recursos y Referencias

- **R for Data Science** (Hadley Wickham & Garrett Grolemund): https://r4ds.had.co.nz/
- **Tidyverse website**: https://www.tidyverse.org/
- **ggplot2: Elegant Graphics for Data Analysis**: https://ggplot2-book.org/
- **RStudio Cheatsheets**: https://www.rstudio.com/resources/cheatsheets/
- **Stack Overflow (tag R)**: https://stackoverflow.com/questions/tagged/r

Resumen

- **Tidyverse** es un conjunto coherente de paquetes para ciencia de datos
- El operador **pipe** (`%>%`) permite encadenar operaciones de forma legible
- **Tibbles** son data frames modernos con mejor comportamiento
- **readr** facilita la importación y exportación de datos
- **dplyr** proporciona verbos para manipulación de datos
- **ggplot2** implementa la gramática de gráficos para visualización

¡Gracias!

Próxima clase: Profundizaremos en operaciones más avanzadas del Tidyverse