INTRODUCCIÓN

# 1. Visualización de datos

## 1.1 Introducción

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.0 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(palmerpenguins)  
library(ggthemes)

## 1.2 Primeros pasos

¿Los pinguinos con alas largas pesan mas o menos que los pinguienos con alas cortas? ¿Qué relación puede tener o cómo puede lucir? ¿Es positivo? ¿Negativo?¿Lineal? ¿No lineal? ¿Varia dependiendo de las especies? ¿Qué hay acerca de la isla en la que viven los pinguinos?

### 1.2.1 el penguins data frame

Es un data frame que se puede encontrar en palmerpenguins.

En un data frame:

* Las columnas son variables
* Las filas son observaciones

Un data frame es la colección de columnas y filas.

*Variable:* cantidad, cualidad o propiedad que puede medirse

*Valor:* esl el estado de una variable cuando se mide. El valor de una variable puede cambiar de medida en medida

*Observación:* es un grupo de medidas hechas bajo condiciones similares. Usualmente se realizan todas las medidas en una observación al mismo tiempo en un solo objeto. Una observación contendrá difrentes valores cada una asociada a una variable diferente. A las obserbaciones tambien se les llama data.

*Tabular data:* es un conjunto de valores cada uno asociado con una una variable y una observación. Se considera ordenada si cada valor está en su propia celda, cada variable está en su propia columna, y cada observación en su propia fila

penguins

## # A tibble: 344 × 8  
## species island bill\_length\_mm bill\_depth\_mm flipper\_length\_mm body\_mass\_g  
## <fct> <fct> <dbl> <dbl> <int> <int>  
## 1 Adelie Torgersen 39.1 18.7 181 3750  
## 2 Adelie Torgersen 39.5 17.4 186 3800  
## 3 Adelie Torgersen 40.3 18 195 3250  
## 4 Adelie Torgersen NA NA NA NA  
## 5 Adelie Torgersen 36.7 19.3 193 3450  
## 6 Adelie Torgersen 39.3 20.6 190 3650  
## 7 Adelie Torgersen 38.9 17.8 181 3625  
## 8 Adelie Torgersen 39.2 19.6 195 4675  
## 9 Adelie Torgersen 34.1 18.1 193 3475  
## 10 Adelie Torgersen 42 20.2 190 4250  
## # ℹ 334 more rows  
## # ℹ 2 more variables: sex <fct>, year <int>

glimpse(penguins)

## Rows: 344  
## Columns: 8  
## $ species <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel…  
## $ island <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse…  
## $ bill\_length\_mm <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, …  
## $ bill\_depth\_mm <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, …  
## $ flipper\_length\_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186…  
## $ body\_mass\_g <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, …  
## $ sex <fct> male, female, female, NA, female, male, female, male…  
## $ year <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007…

### 1.2.2 Meta final

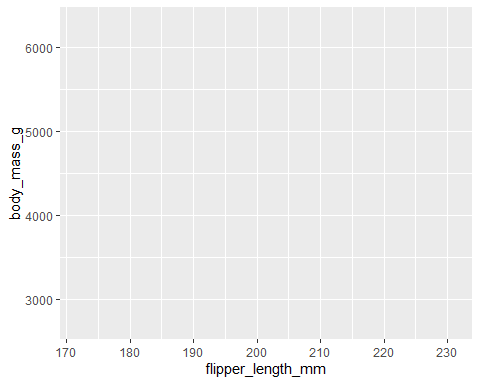
La meta es diseñar una visualización como la siguiente, mostrando la relación entre las alas y la masa corporal de los pinguinos, tomando en cuenta la especie.

### 1.2.3 Creating a ggplot

ggplot(data = penguins) # Grafica un liezno vacío. La idea es que sobre ese lienzo ir agregando o "sumando" capas una sobre la otra



ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g) # Ese argumento define como las variables del dataset se les asignan propiedades visuales.  
)

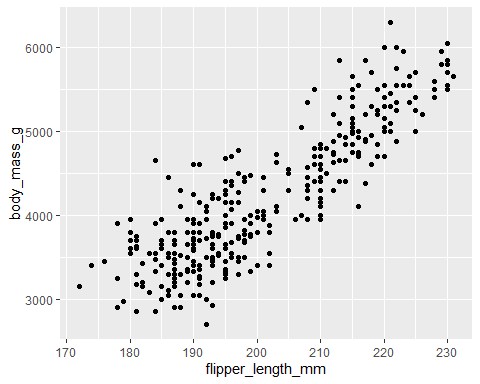


El argumento mapped siempre está definido en la función aes(). Pero los datos aún n se han graficado pues aún no se ha especificado en el codigo como representar las observaciones.

Falta definir el objeto geométrico para representar los datos. Este parámetro inicia siempre por geom\_

ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
) +  
 geom\_point() # En este caso se represantarann a través de un punto, siendo así un grpafico de dispersión

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



Con esto podemos ir tratando de sacar conclusiones: La relación aparenta ser positiva, bastante lineal, moderadamente fuere (no hay mucha dispersión al rededor de dicha línea). Pinguinos con alas largas son generalmente más pesados.

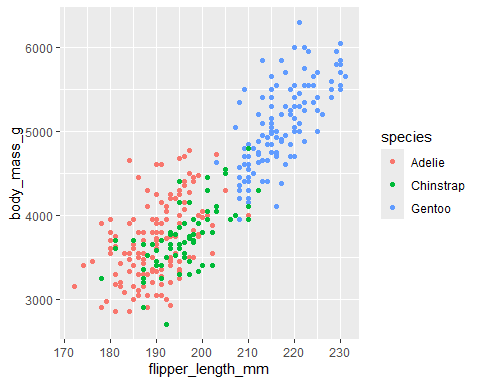
### 1.2.4 Adding aesthetics and layers

Las gráficas de disperisón son buenas mostrando relaciones entre dos variables, pero a veces puede haber otra variable que interfiera en el comportamiento. Por ejemplo, en este caso, las diferencias entre especies.

Para ralizarlo solo se agrega una nueva estética, en este caso puede ser de color

ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g, color = species)  
) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



Cuando una variable categóriga se le asigna una estética, ggplot2 automáticamente asignará un único valro estético a cada nivel de la variable. Esto es llamado *escalamiento*.

Podemos agregar una capa más, en este caso una curva suave que muestre la realción entre la masa corporal y la longitud de la aleta.

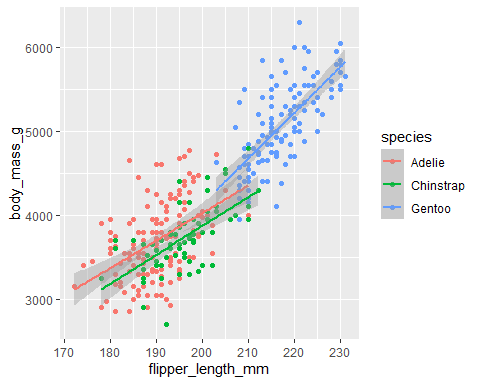
Este es un nuevo objeto geométrico por lo que se debe agregar una capa más.

ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g, color = species)  
) +  
 geom\_point() +  
 geom\_smooth(method = "lm") # El argumento "method = "lm"" se usa para espeficicar que dibuje la ínea basandose en un modelo linea

## `geom\_smooth()` using formula = 'y ~ x'

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



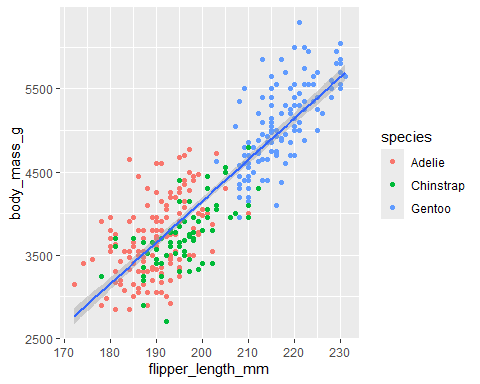
Pero se generaron varias líneas para cada especie, queremos que sea una sola para toda la gráfica. En este caso se definieron las asignaciones eséticas de forma global (en el ggplot) por lo que se aplican a las capas geométricas subsecentes de ahí hacia abajo. Como definimos los colores en el global, entonces eso dividió los datos en función de las especies, por lo que al palicar la aplicar la nueva gráfica se aplica esta división. Por s parte si definimos los colores dentro de la gráfica de dispersión, esta está dividida solo dentro de dicha grafica.

ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
) +  
 geom\_point(mapping = aes(color = species)) +  
 geom\_smooth(method = "lm")

## `geom\_smooth()` using formula = 'y ~ x'

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



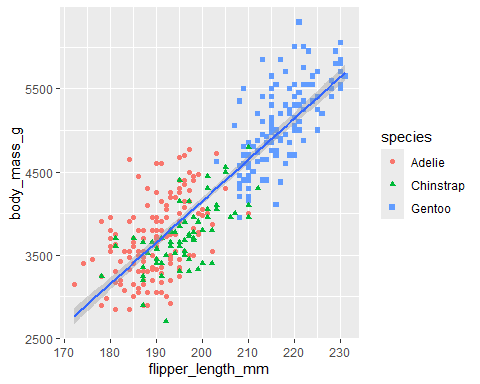
Podemos agregar adempas, formas para ayudar diferenciar los datos

ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
) +  
 geom\_point(mapping = aes(color = species, shape = species)) +  
 geom\_smooth(method = "lm")

## `geom\_smooth()` using formula = 'y ~ x'

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



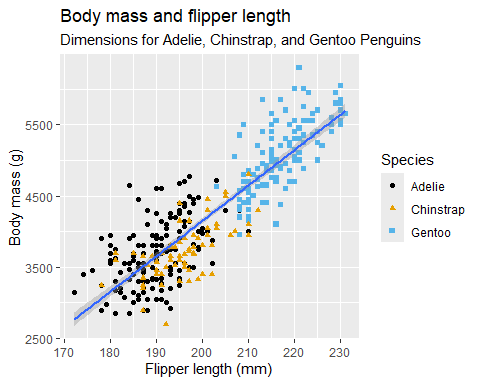
Finalmente podemos mejorar los ejes agregando otra capa. Algunes argumentos son explicatores, como el nombre de los ejes, pero además conserva lso argumentos estéticos anteriormente tratados.

ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
) +  
 geom\_point(aes(color = species, shape = species)) +  
 geom\_smooth(method = "lm") +  
 labs(  
 title = "Body mass and flipper length",  
 subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",  
 x = "Flipper length (mm)", y = "Body mass (g)",  
 color = "Species", shape = "Species"  
 ) +  
 scale\_color\_colorblind() # Esa nueva capa además me modifica los colores parwa que puedan ser vistos por daltónicos

## `geom\_smooth()` using formula = 'y ~ x'

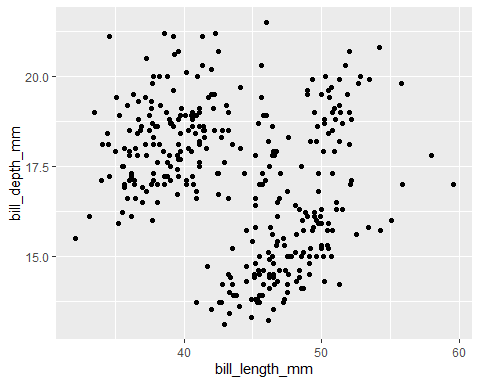
## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



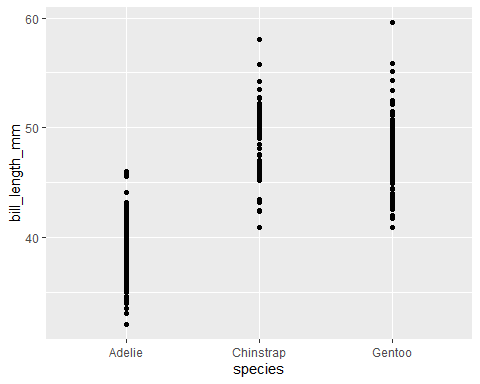
ggplot(  
 data = penguins,  
 mapping = aes(y = bill\_depth\_mm, x = bill\_length\_mm)  
) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



ggplot(  
 data = penguins,  
 mapping = aes(x = species, y = bill\_length\_mm)  
) +  
 geom\_point()

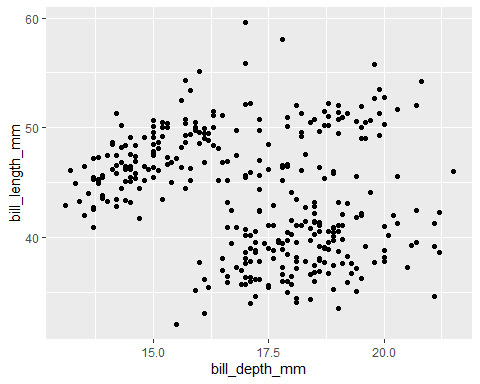
## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



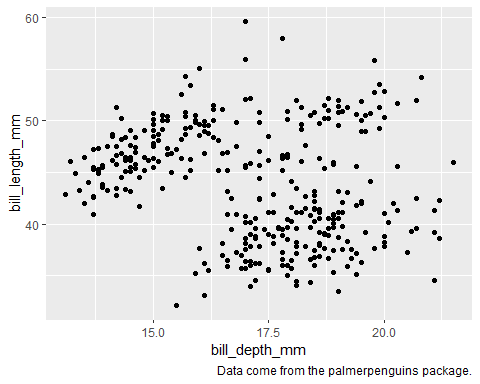
species es un dato categórico. Los gráficos de dispersión cuand ambos ejes son valores numéricos.

# ggplot(data = penguins) +  
# geom\_point()

ggplot(  
 data = penguins,  
 aes(x = bill\_depth\_mm, y = bill\_length\_mm)  
) +  
 geom\_point(na.rm = TRUE) # Remuve los valores desaparecidos sin un alerta

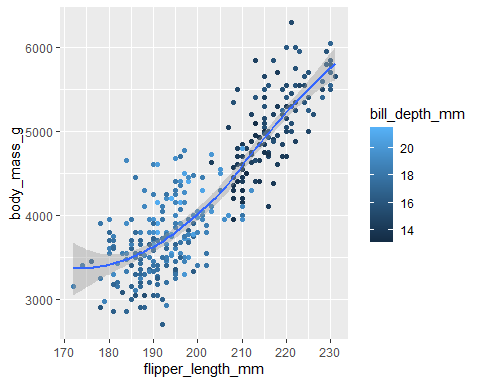


ggplot(  
 data = penguins,  
 aes(x = bill\_depth\_mm, y = bill\_length\_mm)  
) +  
 geom\_point(na.rm = TRUE) +  
 labs(caption = "Data come from the palmerpenguins package.") # Caption agrega un pie de página en la parte inferior derecha



ggplot(  
 data = penguins,  
 mapping = aes(y = body\_mass\_g, x = flipper\_length\_mm)  
) +  
 geom\_point(aes(color = bill\_depth\_mm), na.rm = TRUE) +  
 geom\_smooth(na.rm = TRUE)

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

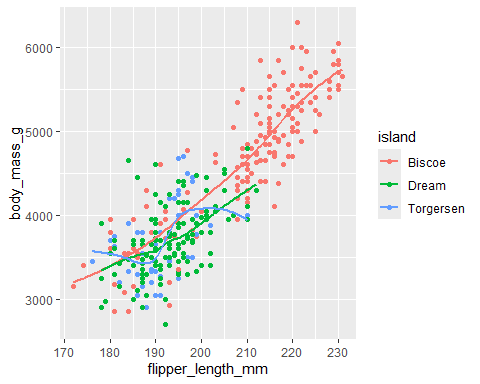


ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g, color = island)  
) +  
 geom\_point() +  
 geom\_smooth(se = FALSE) # Display confidence interval around smooth? (TRUE by default, see level to control.)

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

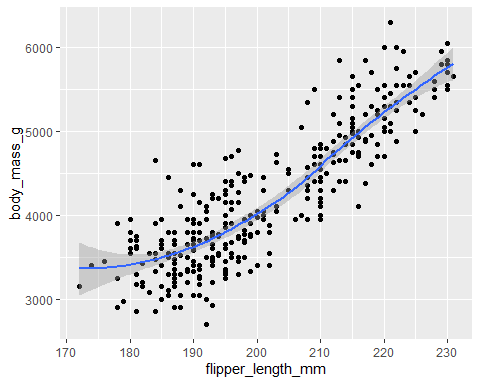


ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
) +  
 geom\_point() +  
 geom\_smooth()

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

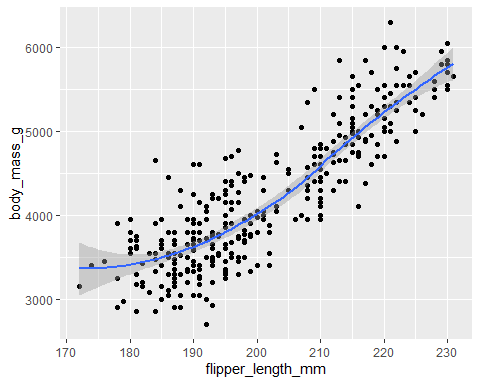
## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



ggplot() +  
 geom\_point(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
 ) +  
 geom\_smooth(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
 )

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

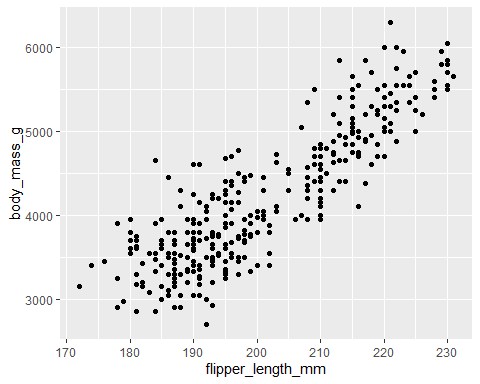
## Warning: Removed 2 rows containing non-finite outside the scale range (`stat\_smooth()`).  
## Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



## 1.3 ggplot2 calls

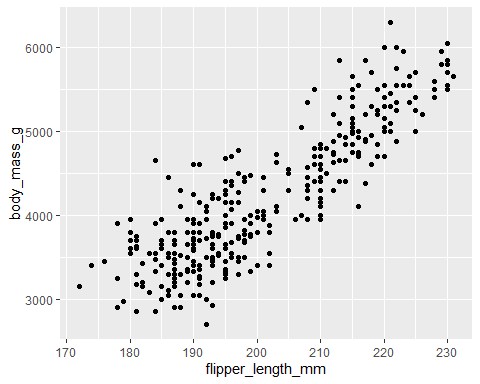
ggplot(  
 data = penguins,  
 mapping = aes(x = flipper\_length\_mm, y = body\_mass\_g)  
) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



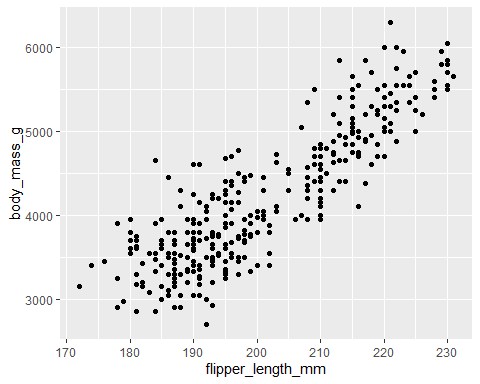
ggplot(penguins, aes(x = flipper\_length\_mm, y = body\_mass\_g)) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



penguins |>  
 ggplot(aes(x = flipper\_length\_mm, y = body\_mass\_g)) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



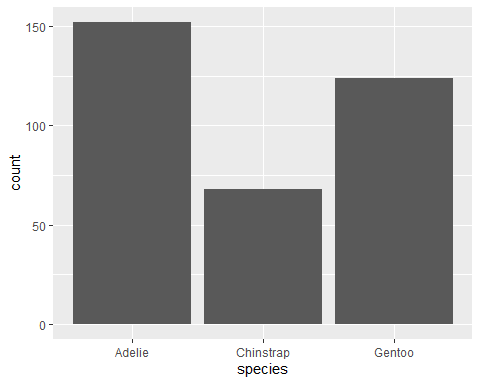
## 1.4 Visualizing distributions

El cómo se visualiza la distribución de una variable depende de el tipo de variable: Categórica o numérica

### 1.4.1 Variable categórica

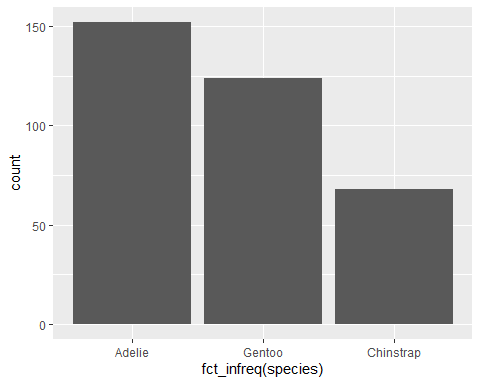
Lo es si toma un pequeño conjuunto de valores. Es mejor usar un diagrama de barras en este caso.

ggplot(penguins, aes(x = species)) +  
 geom\_bar()



Cuando no se tienen las categorías ordenes lo mejor es ordenarlas. Para eso se debe transformar la variable en un factor (Que es como R mejor maneja los datos) y reordenar los niveles de ese factor.

ggplot(penguins, aes(x = fct\_infreq(species))) + # Convierto la variable ne un factor  
 geom\_bar()

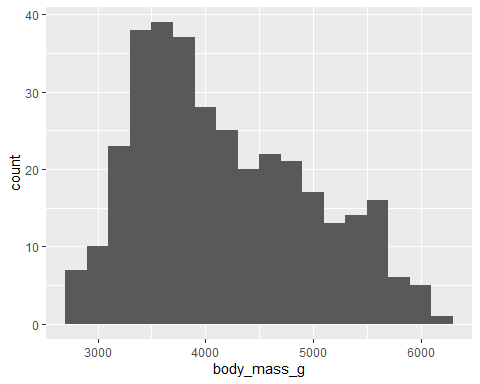


### 1.4.2 Variable numérica

Pueden tomar un rango de valores numéricos. Es suceptible de sumas, restas o promedios. Pueden ser continuas o discretas. Uno de los mejores gráficos para este tipo de daos es el histograma.

ggplot(penguins, aes(x = body\_mass\_g)) +  
 geom\_histogram(binwidth = 200)

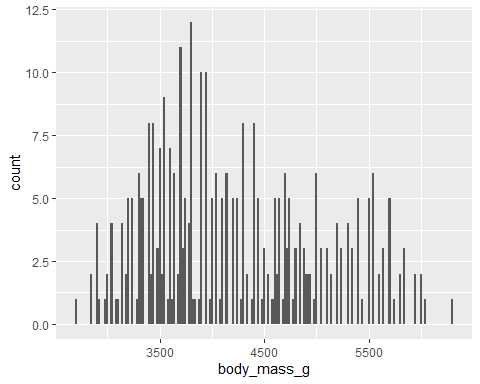
## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_bin()`).



Se puede modifcar el ancho del intervalo del histograma. Se hace con el argumento *bindwith*.

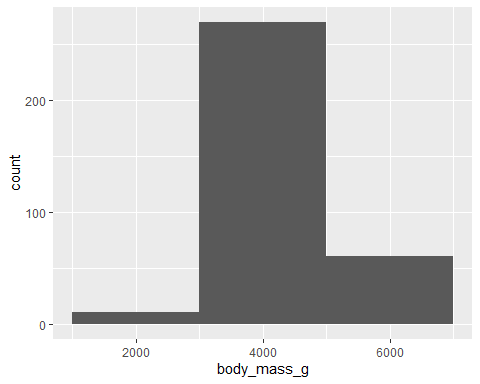
ggplot(penguins, aes(x = body\_mass\_g)) +  
 geom\_histogram(binwidth = 20)

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_bin()`).



ggplot(penguins, aes(x = body\_mass\_g)) +  
 geom\_histogram(binwidth = 2000)

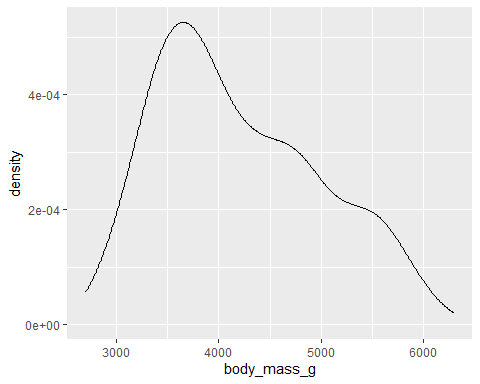
## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_bin()`).



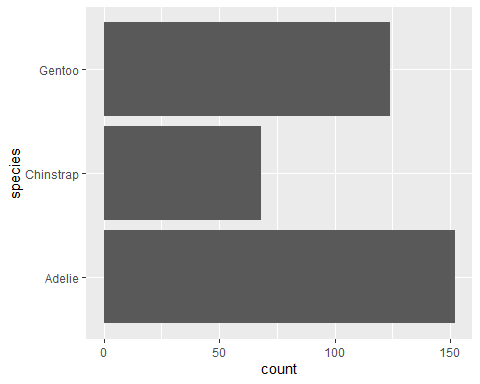
Otra alternativa para los datos numéricos es gráfico de densidad, el cual es una forma suavizada del histograma. Ideal para datos continuos.

ggplot(penguins, aes(x = body\_mass\_g)) +  
 geom\_density()

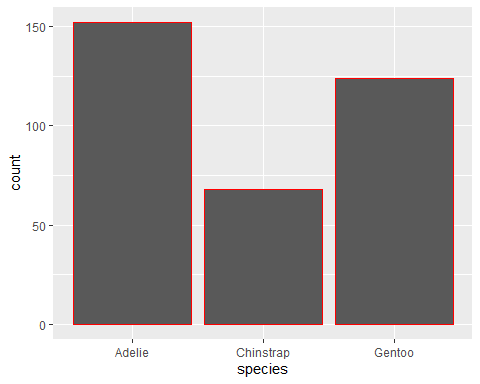
## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_density()`).



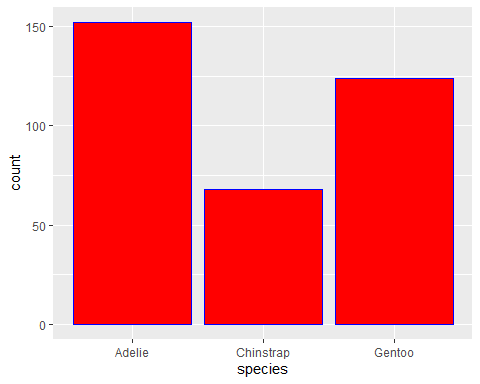
ggplot(  
 data = penguins,  
 mapping = aes(y = species)  
) +  
 geom\_bar()



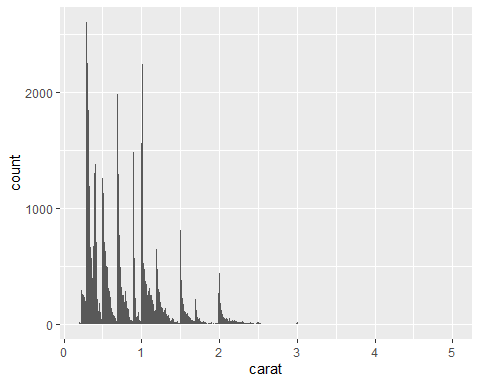
ggplot(penguins, aes(x = species)) +  
 geom\_bar(color = "red")



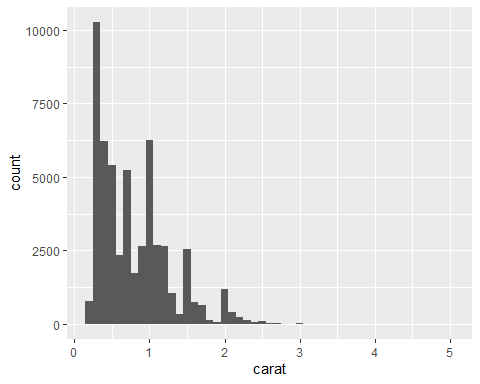
ggplot(penguins, aes(x = species)) +  
 geom\_bar(fill = "red", color = "blue")



ggplot(diamonds, aes(x = carat)) +  
 geom\_histogram(binwidth = 0.01)



ggplot(diamonds, aes(x = carat)) +  
 geom\_histogram(binwidth = 0.10)



ggplot(diamonds, aes(x = carat)) +  
 geom\_histogram(binwidth = 1)



## 1.5 Visualizing relationships

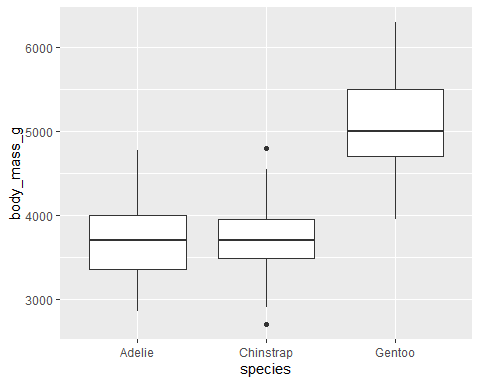
### 1.5.1 Variable numérica y categórica

Lo mejor es usar un box-plot. Es un tipo de gráfica para las medidas de posición (Percentiles) que describen una disribución. EL diagrama consta de:

* una caja que indica el rango intermedio de los datos. se extiende desde el percentil 25 hasta el 75 (llamado IQR). En el centro hay una linea que muestra la mediana, es decir, el perecentil 50. Esto da idea de la dispersión de la distribución.
* Puntos visuales que muestran oservaciones que caen más de 1,5 veces el IQR desde cualquier extremo de la caja. Son puntos inusuales, atípicos.
* Una línea (o extremo) que se extiende desde cada extremo de la caja y llega al punto no atípico más lejano en la distribución.

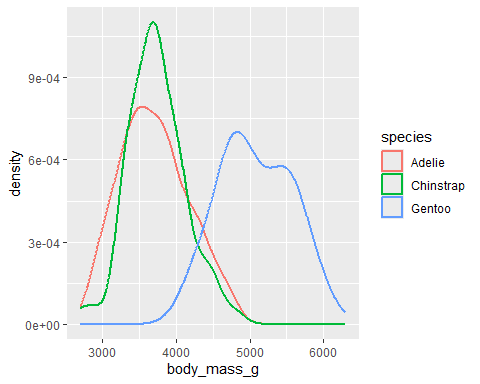
ggplot(penguins, aes(x = species, y = body\_mass\_g)) +  
 geom\_boxplot()

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_boxplot()`).



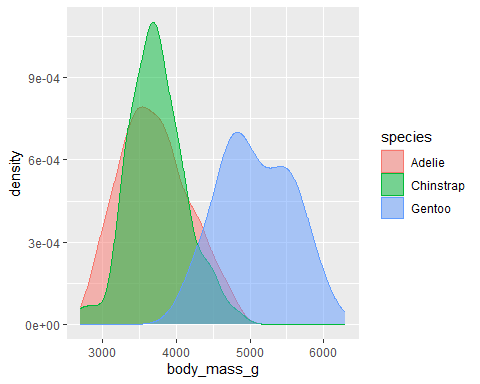
ggplot(penguins, aes(x = body\_mass\_g, color = species)) +  
 geom\_density(linewidth = 0.75) # Modifica el ancho de la línea

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_density()`).



ggplot(penguins, aes(x = body\_mass\_g, color = species, fill = species)) + # Relleno y linea basados en el mismo dato adaptando el mismo color  
 geom\_density(alpha = 0.5) # Establesco la opacidad del color establecido. Toma valores entre 0 y 1

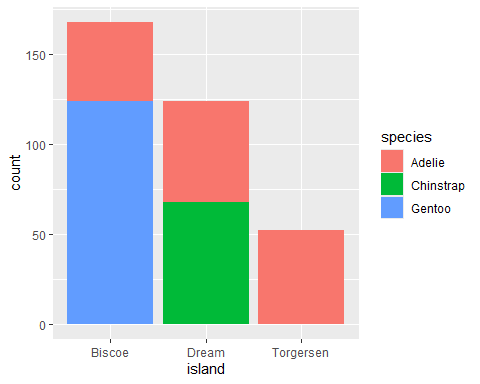
## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_density()`).



### 1.5.2 Dos variables categóricas

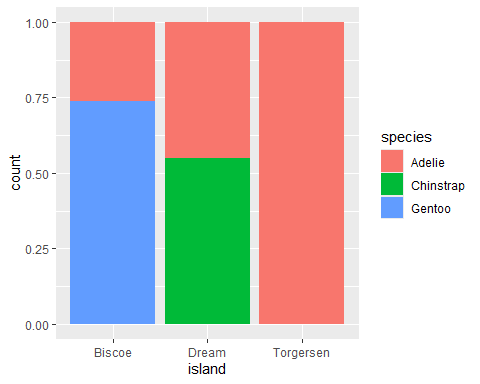
Podemos usar columnas apiladas

ggplot(penguins, aes(x = island, fill = species)) +  
 geom\_bar()



Con esto podemos ver la cantidad de pinguinos por islas y el tipo de pinguinos en cada isla, pero no podemos ver que porcentaje hay de cada uno así que podemos mostrarlo de la siguiente forma

ggplot(penguins, aes(x = island, fill = species)) +  
 geom\_bar(position = "fill") # Grafica con respecto al 100% mas no con la cantidad

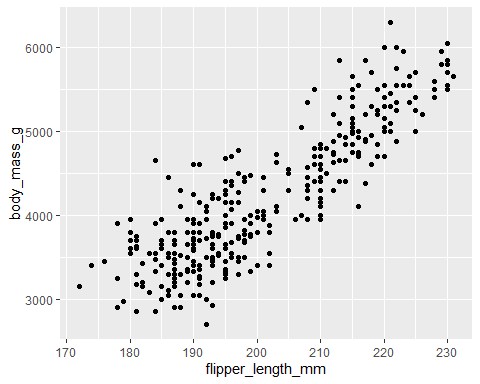


### 1.5.3 Dos variables numéricas

Como ya se mencionó anteriormente lo mejor en este caso es un gráfico de dispersión.

ggplot(penguins, aes(x = flipper\_length\_mm, y = body\_mass\_g)) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

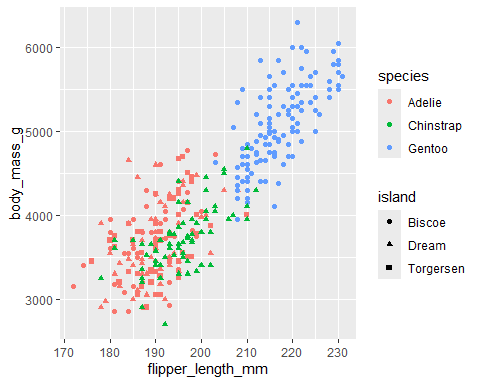


### 1.5.4 Tres o más variables

En la siguiente gráfica colores rpresentan especies y figuras rpresentan islas

ggplot(penguins, aes(x = flipper\_length\_mm, y = body\_mass\_g)) +  
 geom\_point(aes(color = species, shape = island))

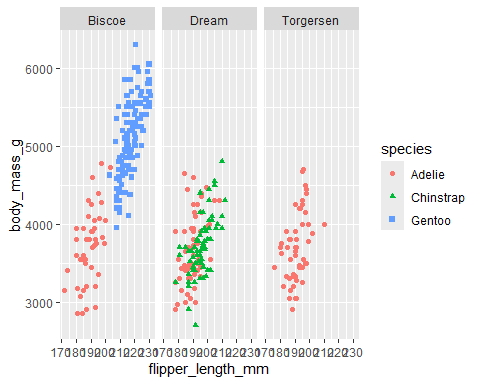
## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



Lo anterior hace ver desordenado y crea dificultad al entenderlo.. Para eso podemos dividir la gráfica en *facetas*. Son subgráficas que muestran un subconjunto cada una.

ggplot(penguins, aes(x = flipper\_length\_mm, y = body\_mass\_g)) +  
 geom\_point(aes(color = species, shape = species)) +  
 facet\_wrap(~island)

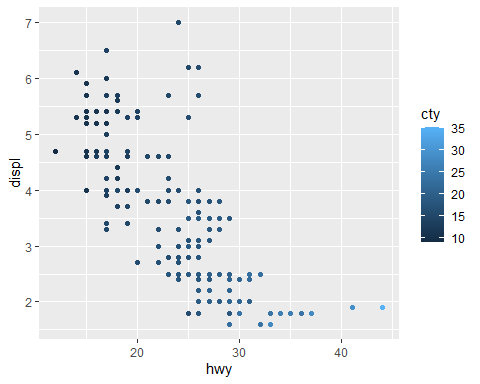
## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



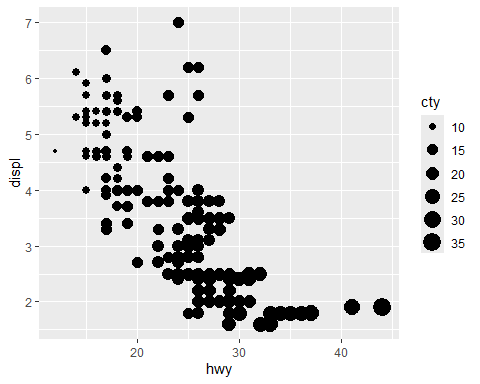
glimpse(mpg)

## Rows: 234  
## Columns: 11  
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "…  
## $ model <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "…  
## $ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.…  
## $ year <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200…  
## $ cyl <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, …  
## $ trans <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto…  
## $ drv <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4…  
## $ cty <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1…  
## $ hwy <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2…  
## $ fl <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p…  
## $ class <chr> "compact", "compact", "compact", "compact", "compact", "c…

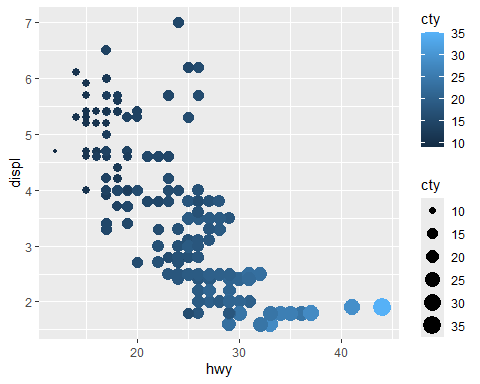
ggplot(  
 mpg,  
 aes(x = hwy, y = displ, color = cty)  
) +  
 geom\_point()



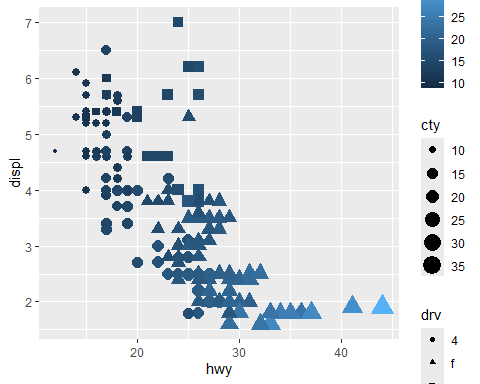
ggplot(  
 mpg,  
 aes(x = hwy, y = displ, size = cty)  
) +  
 geom\_point()



ggplot(  
 mpg,  
 aes(x = hwy, y = displ, size = cty, color = cty)  
) +  
 geom\_point()

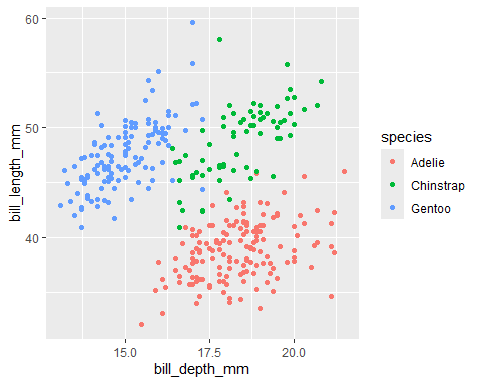


ggplot(  
 mpg,  
 aes(x = hwy, y = displ, size = cty, color = cty, shape = drv)  
) +  
 geom\_point()



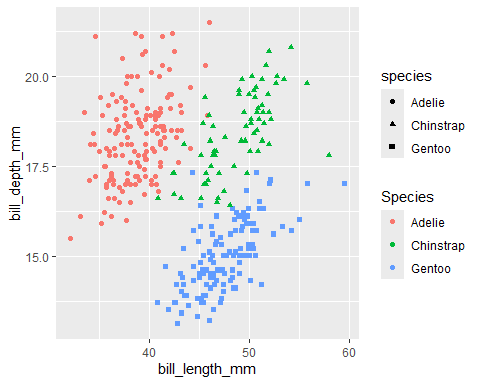
ggplot(  
 penguins,  
 aes(x = bill\_depth\_mm, y = bill\_length\_mm, color = species)  
) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



ggplot(  
 data = penguins,  
 mapping = aes(  
 x = bill\_length\_mm, y = bill\_depth\_mm,  
 color = species, shape = species  
 )  
) +  
 geom\_point() +  
 labs(color = "Species")

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

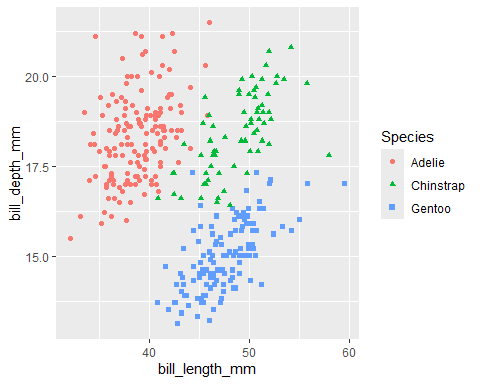


#>The code provided in the exercise yields two separate legends because the legend for color is renamed to "Species" but the legend for shape is not, and is named "species" by default instead. To fix it, we would need to explicitly rename the shape legend as well.

Solución al problema de la gráfica anterior:

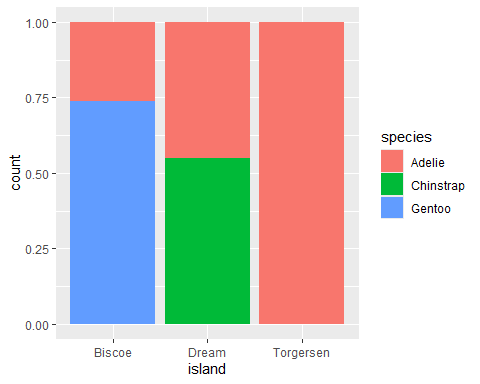
ggplot(  
 data = penguins,  
 mapping = aes(  
 x = bill\_length\_mm, y = bill\_depth\_mm,  
 color = species, shape = species  
 )  
) +  
 geom\_point() +  
 labs(  
 color = "Species",  
 shape = "Species"  
 )

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

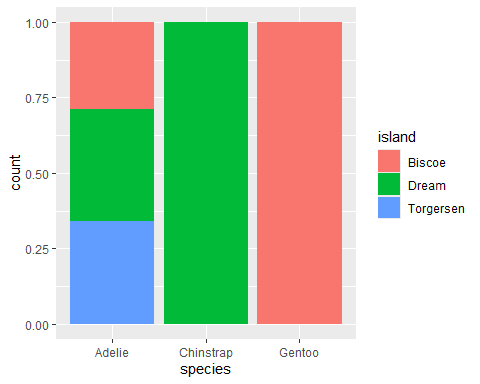


#> Warning: Removed 2 rows containing missing values (`geom\_point()`).

ggplot(penguins, aes(x = island, fill = species)) +  
 geom\_bar(position = "fill")



ggplot(penguins, aes(x = species, fill = island)) +  
 geom\_bar(position = "fill")

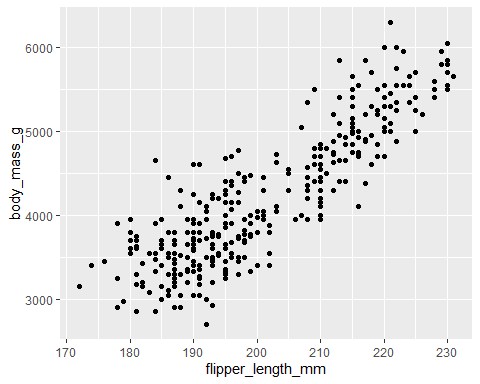


## 1.6 Saving your plots

La función que se encarga de guardar las gráficas como imágenes es ggsave(). Guarda la gráfica creada más recientemente el disco.

ggplot(penguins, aes(x = flipper\_length\_mm, y = body\_mass\_g)) +  
 geom\_point()

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

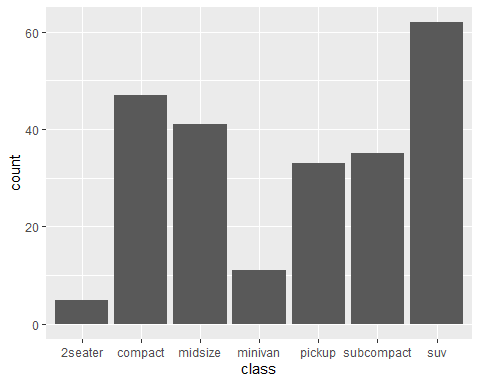


ggsave(filename = "penguin-plot.png")

## Saving 5 x 4 in image

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

ggplot(mpg, aes(x = class)) +  
 geom\_bar()



ggplot(mpg, aes(x = cty, y = hwy)) +  
 geom\_point()



ggsave("mpg-plot.pdf")

## Saving 5 x 4 in image

# 2. Flujo de trabajo: Basicos

1 / 200 \* 30

## [1] 0.15

#> [1] 0.15  
(59 + 73 + 2) / 3

## [1] 44.66667

#> [1] 44.66667  
sin(pi / 2)

## [1] 1

#> [1] 1

x <- 3 \* 4

primes <- c(2, 3, 5, 7, 11, 13)

primes \* 2

## [1] 4 6 10 14 22 26

#> [1] 4 6 10 14 22 26  
primes - 1

## [1] 1 2 4 6 10 12

#> [1] 1 2 4 6 10 12

# create vector of primes  
primes <- c(2, 3, 5, 7, 11, 13)  
  
# multiply primes by 2  
primes \* 2

## [1] 4 6 10 14 22 26

#> [1] 4 6 10 14 22 26

# function\_name(argument1 = value1, argument2 = value2, ...)

seq(from = 1, to = 10)

## [1] 1 2 3 4 5 6 7 8 9 10

# 3. Transformación de datos

library(nycflights13)  
flights

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

View(flights)

print(flights, width = Inf)

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## arr\_delay carrier flight tailnum origin dest air\_time distance hour minute  
## <dbl> <chr> <int> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 11 UA 1545 N14228 EWR IAH 227 1400 5 15  
## 2 20 UA 1714 N24211 LGA IAH 227 1416 5 29  
## 3 33 AA 1141 N619AA JFK MIA 160 1089 5 40  
## 4 -18 B6 725 N804JB JFK BQN 183 1576 5 45  
## 5 -25 DL 461 N668DN LGA ATL 116 762 6 0  
## 6 12 UA 1696 N39463 EWR ORD 150 719 5 58  
## 7 19 B6 507 N516JB EWR FLL 158 1065 6 0  
## 8 -14 EV 5708 N829AS LGA IAD 53 229 6 0  
## 9 -8 B6 79 N593JB JFK MCO 140 944 6 0  
## 10 8 AA 301 N3ALAA LGA ORD 138 733 6 0  
## time\_hour   
## <dttm>   
## 1 2013-01-01 05:00:00  
## 2 2013-01-01 05:00:00  
## 3 2013-01-01 05:00:00  
## 4 2013-01-01 05:00:00  
## 5 2013-01-01 06:00:00  
## 6 2013-01-01 05:00:00  
## 7 2013-01-01 06:00:00  
## 8 2013-01-01 06:00:00  
## 9 2013-01-01 06:00:00  
## 10 2013-01-01 06:00:00  
## # ℹ 336,766 more rows

### 3.1.3 dplyr basics

En las funciones:

* El primer argumento siempre es el dataframe.
* Los argumentos subsiguientes suelen describir sobre qué columnas operar, utilizando los nombres de las variables (sin comillas).
* La salida siempre es un nuevo dataframe

Debido a que cada verbo hace una cosa bien, resolver problemas complejos generalmente requerirá combinar múltiples verbos, y lo haremos con el operador de tubería, |>. x |> f(y) es equivalente a f(x, y). y x |> f(y) |> g(z) es equivalente a g(f(x, y), z).

flights |>  
 filter(dest == "IAH") |>  
 group\_by(year, month, day) |>  
 summarize(  
 arr\_delay = mean(arr\_delay, na.rm = TRUE)  
 )

## `summarise()` has grouped output by 'year', 'month'. You can override using the  
## `.groups` argument.

## # A tibble: 365 × 4  
## # Groups: year, month [12]  
## year month day arr\_delay  
## <int> <int> <int> <dbl>  
## 1 2013 1 1 17.8   
## 2 2013 1 2 7   
## 3 2013 1 3 18.3   
## 4 2013 1 4 -3.2   
## 5 2013 1 5 20.2   
## 6 2013 1 6 9.28  
## 7 2013 1 7 -7.74  
## 8 2013 1 8 7.79  
## 9 2013 1 9 18.1   
## 10 2013 1 10 6.68  
## # ℹ 355 more rows

## 3.2 Rows

Los verbos o funciones mas importantes son:

* filter(): cambia qué filas están presentes sin alterar su orden
* arrange(): cambia el orden de las filas sin alterar cuáles están presentes

### 3.2.2 filter()

flights |>  
 filter(dep\_delay > 120) # Muesra que filas tiene un retraso mayor a 120

## # A tibble: 9,723 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 848 1835 853 1001 1950  
## 2 2013 1 1 957 733 144 1056 853  
## 3 2013 1 1 1114 900 134 1447 1222  
## 4 2013 1 1 1540 1338 122 2020 1825  
## 5 2013 1 1 1815 1325 290 2120 1542  
## 6 2013 1 1 1842 1422 260 1958 1535  
## 7 2013 1 1 1856 1645 131 2212 2005  
## 8 2013 1 1 1934 1725 129 2126 1855  
## 9 2013 1 1 1938 1703 155 2109 1823  
## 10 2013 1 1 1942 1705 157 2124 1830  
## # ℹ 9,713 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 filter(month == 1 & day == 1) # Muestra los datos que sean del primero de enero

## # A tibble: 842 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 832 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights %>% # Es lo mismo que escribir |>  
 filter(month == 1 | month == 2)

## # A tibble: 51,955 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 51,945 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

%in%. Este mantiene las filas donde la variable es igual a uno de los valores a la derecha:

flights |>  
 filter(month %in% c(1, 2)) # meses correspondientes a enero y febrero

## # A tibble: 51,955 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 51,945 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

When you run [filter()](https://dplyr.tidyverse.org/reference/filter.html) dplyr executes the filtering operation, creating a new data frame, and then prints it. It doesn’t modify the existing flights dataset because dplyr functions never modify their inputs. To save the result, you need to use the assignment operator, <-

jan1 <- flights |>  
 filter(month == 1 & day == 2)

### 3.2.3 arrange()

Camhia el orden de las filas basado en el valor de las columnas. Si se usa más de una columna se usan para desempatar algunos valores

flights |>  
 arrange(year, month, day, dep\_time)

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Se puede usar des() dentro de arrange()

flights |>  
 arrange(desc(dep\_delay))

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 9 641 900 1301 1242 1530  
## 2 2013 6 15 1432 1935 1137 1607 2120  
## 3 2013 1 10 1121 1635 1126 1239 1810  
## 4 2013 9 20 1139 1845 1014 1457 2210  
## 5 2013 7 22 845 1600 1005 1044 1815  
## 6 2013 4 10 1100 1900 960 1342 2211  
## 7 2013 3 17 2321 810 911 135 1020  
## 8 2013 6 27 959 1900 899 1236 2226  
## 9 2013 7 22 2257 759 898 121 1026  
## 10 2013 12 5 756 1700 896 1058 2020  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

### 3.2.4 distinct()

Encuentra todos los valores únicos.

flights |>  
 distinct()

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |> # Find all unique origin and destination pairs  
 distinct(origin, dest)

## # A tibble: 224 × 2  
## origin dest   
## <chr> <chr>  
## 1 EWR IAH   
## 2 LGA IAH   
## 3 JFK MIA   
## 4 JFK BQN   
## 5 LGA ATL   
## 6 EWR ORD   
## 7 EWR FLL   
## 8 LGA IAD   
## 9 JFK MCO   
## 10 LGA ORD   
## # ℹ 214 more rows

Si quiero sacar los valores únicos de una columna pero quiero traerme otras columnas que se vean modificadas por esas solo tengo que hacer lo siguiente:

flights |>  
 distinct(origin, dest, .keep\_all = TRUE) # Traigo los valores únicos de origin, dest y mantengo las demás columnas

## # A tibble: 224 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 214 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Si quiero determinar el número de ocurrencias

flights |>  
 count(origin, dest, sort = TRUE) # Cuenta el número de veces que sucede algo

## # A tibble: 224 × 3  
## origin dest n  
## <chr> <chr> <int>  
## 1 JFK LAX 11262  
## 2 LGA ATL 10263  
## 3 LGA ORD 8857  
## 4 JFK SFO 8204  
## 5 LGA CLT 6168  
## 6 EWR ORD 6100  
## 7 JFK BOS 5898  
## 8 LGA MIA 5781  
## 9 JFK MCO 5464  
## 10 EWR BOS 5327  
## # ℹ 214 more rows

flights |>  
 filter(arr\_delay >= 120)

## # A tibble: 10,200 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 811 630 101 1047 830  
## 2 2013 1 1 848 1835 853 1001 1950  
## 3 2013 1 1 957 733 144 1056 853  
## 4 2013 1 1 1114 900 134 1447 1222  
## 5 2013 1 1 1505 1310 115 1638 1431  
## 6 2013 1 1 1525 1340 105 1831 1626  
## 7 2013 1 1 1549 1445 64 1912 1656  
## 8 2013 1 1 1558 1359 119 1718 1515  
## 9 2013 1 1 1732 1630 62 2028 1825  
## 10 2013 1 1 1803 1620 103 2008 1750  
## # ℹ 10,190 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 filter(dest %in% c("IAH", "HOU"))

## # A tibble: 9,313 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 623 627 -4 933 932  
## 4 2013 1 1 728 732 -4 1041 1038  
## 5 2013 1 1 739 739 0 1104 1038  
## 6 2013 1 1 908 908 0 1228 1219  
## 7 2013 1 1 1028 1026 2 1350 1339  
## 8 2013 1 1 1044 1045 -1 1352 1351  
## 9 2013 1 1 1114 900 134 1447 1222  
## 10 2013 1 1 1205 1200 5 1503 1505  
## # ℹ 9,303 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 filter(carrier %in% c("UA", "AA", "DL"))

## # A tibble: 139,504 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 554 600 -6 812 837  
## 5 2013 1 1 554 558 -4 740 728  
## 6 2013 1 1 558 600 -2 753 745  
## 7 2013 1 1 558 600 -2 924 917  
## 8 2013 1 1 558 600 -2 923 937  
## 9 2013 1 1 559 600 -1 941 910  
## 10 2013 1 1 559 600 -1 854 902  
## # ℹ 139,494 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 filter(month %in% c(6, 7, 8))

## # A tibble: 86,995 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 6 1 2 2359 3 341 350  
## 2 2013 6 1 451 500 -9 624 640  
## 3 2013 6 1 506 515 -9 715 800  
## 4 2013 6 1 534 545 -11 800 829  
## 5 2013 6 1 538 545 -7 925 922  
## 6 2013 6 1 539 540 -1 832 840  
## 7 2013 6 1 546 600 -14 850 910  
## 8 2013 6 1 551 600 -9 828 850  
## 9 2013 6 1 552 600 -8 647 655  
## 10 2013 6 1 553 600 -7 700 711  
## # ℹ 86,985 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 filter(arr\_delay >= 120 & dep\_delay <= 0) |>  
 view()  
flights |>  
 filter(dep\_delay >= 60 & dep\_delay - arr\_delay > 30)

## # A tibble: 1,844 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 2205 1720 285 46 2040  
## 2 2013 1 1 2326 2130 116 131 18  
## 3 2013 1 3 1503 1221 162 1803 1555  
## 4 2013 1 3 1839 1700 99 2056 1950  
## 5 2013 1 3 1850 1745 65 2148 2120  
## 6 2013 1 3 1941 1759 102 2246 2139  
## 7 2013 1 3 1950 1845 65 2228 2227  
## 8 2013 1 3 2015 1915 60 2135 2111  
## 9 2013 1 3 2257 2000 177 45 2224  
## 10 2013 1 4 1917 1700 137 2135 1950  
## # ℹ 1,834 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 arrange(desc(dep\_delay)) |>  
 arrange(sched\_dep\_time) |>  
 relocate(dep\_delay, sched\_dep\_time) # Cambia la posición de las columnas

## # A tibble: 336,776 × 19  
## dep\_delay sched\_dep\_time year month day dep\_time arr\_time sched\_arr\_time  
## <dbl> <int> <int> <int> <int> <int> <int> <int>  
## 1 NA 106 2013 7 27 NA NA 245  
## 2 188 500 2013 4 24 808 1008 640  
## 3 61 500 2013 9 13 601 732 648  
## 4 47 500 2013 3 9 547 733 648  
## 5 44 500 2013 6 8 544 727 640  
## 6 29 500 2013 4 17 529 704 640  
## 7 25 500 2013 6 30 525 703 640  
## 8 22 500 2013 11 28 522 658 651  
## 9 20 500 2013 4 21 520 705 640  
## 10 18 500 2013 6 3 518 654 640  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 mutate(speed = distance / (air\_time / 60)) |> # Crea nueva columas que en funcion de variables existentes. También puede modificar y eliminar columnas. En este caso crea la columna speed  
 arrange(desc(speed)) |>  
 relocate(speed)

## # A tibble: 336,776 × 20  
## speed year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <dbl> <int> <int> <int> <int> <int> <dbl> <int>  
## 1 703. 2013 5 25 1709 1700 9 1923  
## 2 650. 2013 7 2 1558 1513 45 1745  
## 3 648 2013 5 13 2040 2025 15 2225  
## 4 641. 2013 3 23 1914 1910 4 2045  
## 5 591. 2013 1 12 1559 1600 -1 1849  
## 6 564 2013 11 17 650 655 -5 1059  
## 7 557. 2013 2 21 2355 2358 -3 412  
## 8 556. 2013 11 17 759 800 -1 1212  
## 9 554. 2013 11 16 2003 1925 38 17  
## 10 554. 2013 11 16 2349 2359 -10 402  
## # ℹ 336,766 more rows  
## # ℹ 12 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 distinct(year, month, day) |>  
 nrow()

## [1] 365

flights |>  
 arrange(desc(distance)) |>  
 relocate(distance)

## # A tibble: 336,776 × 19  
## distance year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <dbl> <int> <int> <int> <int> <int> <dbl> <int>  
## 1 4983 2013 1 1 857 900 -3 1516  
## 2 4983 2013 1 2 909 900 9 1525  
## 3 4983 2013 1 3 914 900 14 1504  
## 4 4983 2013 1 4 900 900 0 1516  
## 5 4983 2013 1 5 858 900 -2 1519  
## 6 4983 2013 1 6 1019 900 79 1558  
## 7 4983 2013 1 7 1042 900 102 1620  
## 8 4983 2013 1 8 901 900 1 1504  
## 9 4983 2013 1 9 641 900 1301 1242  
## 10 4983 2013 1 10 859 900 -1 1449  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 arrange(distance) |>  
 relocate(distance)

## # A tibble: 336,776 × 19  
## distance year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <dbl> <int> <int> <int> <int> <int> <dbl> <int>  
## 1 17 2013 7 27 NA 106 NA NA  
## 2 80 2013 1 3 2127 2129 -2 2222  
## 3 80 2013 1 4 1240 1200 40 1333  
## 4 80 2013 1 4 1829 1615 134 1937  
## 5 80 2013 1 4 2128 2129 -1 2218  
## 6 80 2013 1 5 1155 1200 -5 1241  
## 7 80 2013 1 6 2125 2129 -4 2224  
## 8 80 2013 1 7 2124 2129 -5 2212  
## 9 80 2013 1 8 2127 2130 -3 2304  
## 10 80 2013 1 9 2126 2129 -3 2217  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

## 3.3 Columnas

Hay 4 verbos importantes que afectan las columnas sin cambiar las filas

### 3.3.1 mutate()

Agrega una columna calculada a partir de otras columnas

flights |>  
 mutate(  
 gain = dep\_delay - arr\_delay,  
 speed = distance / air\_time \* 60  
 )

## # A tibble: 336,776 × 21  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 13 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>, gain <dbl>, speed <dbl>

flights |>  
 mutate(  
 gain = dep\_delay - arr\_delay,  
 speed = distance / air\_time \* 60,  
 .before = 1 # Nos permite agregar las columnas a la izquierda en lugar de la derecha  
 )

## # A tibble: 336,776 × 21  
## gain speed year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <dbl> <dbl> <int> <int> <int> <int> <int> <dbl> <int>  
## 1 -9 370. 2013 1 1 517 515 2 830  
## 2 -16 374. 2013 1 1 533 529 4 850  
## 3 -31 408. 2013 1 1 542 540 2 923  
## 4 17 517. 2013 1 1 544 545 -1 1004  
## 5 19 394. 2013 1 1 554 600 -6 812  
## 6 -16 288. 2013 1 1 554 558 -4 740  
## 7 -24 404. 2013 1 1 555 600 -5 913  
## 8 11 259. 2013 1 1 557 600 -3 709  
## 9 5 405. 2013 1 1 557 600 -3 838  
## 10 -10 319. 2013 1 1 558 600 -2 753  
## # ℹ 336,766 more rows  
## # ℹ 12 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

El punto “.” es un indicio de que “.before” es un argumento para la función, no el nombre de una tercera variable nueva que estamos creando. También sse puede usar “.after” para agregar después de una variable, y tanto en “.before” como en “.after” puedes usar el nombre de la variable en lugar de una posición. Por ejemplo, se podría agregar las nuevas variables después de “day”:

flights |>  
 mutate(  
 gain = dep\_delay - arr\_delay,  
 speed = distance / air\_time \* 60,  
 .after = day  
 )

## # A tibble: 336,776 × 21  
## year month day gain speed dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <dbl> <dbl> <int> <int> <dbl> <int>  
## 1 2013 1 1 -9 370. 517 515 2 830  
## 2 2013 1 1 -16 374. 533 529 4 850  
## 3 2013 1 1 -31 408. 542 540 2 923  
## 4 2013 1 1 17 517. 544 545 -1 1004  
## 5 2013 1 1 19 394. 554 600 -6 812  
## 6 2013 1 1 -16 288. 554 558 -4 740  
## 7 2013 1 1 -24 404. 555 600 -5 913  
## 8 2013 1 1 11 259. 557 600 -3 709  
## 9 2013 1 1 5 405. 557 600 -3 838  
## 10 2013 1 1 -10 319. 558 600 -2 753  
## # ℹ 336,766 more rows  
## # ℹ 12 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

Alternativamente, puedes controlar qué variables se conservan con el argumento *.keep*. Si le ingresamos el argumento *used,* veremos solo las columnas que hayan estado involucradas en el cálculo de la nueva columna.

flights |>  
 mutate(  
 gain = dep\_delay - arr\_delay,  
 hours = air\_time / 60,  
 gain\_per\_hour = gain / hours,  
 .keep = "used"  
 )

## # A tibble: 336,776 × 6  
## dep\_delay arr\_delay air\_time gain hours gain\_per\_hour  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2 11 227 -9 3.78 -2.38  
## 2 4 20 227 -16 3.78 -4.23  
## 3 2 33 160 -31 2.67 -11.6   
## 4 -1 -18 183 17 3.05 5.57  
## 5 -6 -25 116 19 1.93 9.83  
## 6 -4 12 150 -16 2.5 -6.4   
## 7 -5 19 158 -24 2.63 -9.11  
## 8 -3 -14 53 11 0.883 12.5   
## 9 -3 -8 140 5 2.33 2.14  
## 10 -2 8 138 -10 2.3 -4.35  
## # ℹ 336,766 more rows

### 3.3.2 select()

Es común recibir datasets con miles de variables. Con esta función podemos elegir las variables que necesitamos.

* Sleccionar columnas por nombres
* flights |>  
   select(year, month, day)
* ## # A tibble: 336,776 × 3  
  ## year month day  
  ## <int> <int> <int>  
  ## 1 2013 1 1  
  ## 2 2013 1 1  
  ## 3 2013 1 1  
  ## 4 2013 1 1  
  ## 5 2013 1 1  
  ## 6 2013 1 1  
  ## 7 2013 1 1  
  ## 8 2013 1 1  
  ## 9 2013 1 1  
  ## 10 2013 1 1  
  ## # ℹ 336,766 more rows
* Seleccionar columnas entre columnas
* flights |>  
   select(year:day)
* ## # A tibble: 336,776 × 3  
  ## year month day  
  ## <int> <int> <int>  
  ## 1 2013 1 1  
  ## 2 2013 1 1  
  ## 3 2013 1 1  
  ## 4 2013 1 1  
  ## 5 2013 1 1  
  ## 6 2013 1 1  
  ## 7 2013 1 1  
  ## 8 2013 1 1  
  ## 9 2013 1 1  
  ## 10 2013 1 1  
  ## # ℹ 336,766 more rows
* Seleccionar todas las columnas excepto las que están entre año y día
* flights |>  
   select(!year:day) # En lugar de "!" también es usar ver códigos con "-"
* ## # A tibble: 336,776 × 16  
  ## dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time arr\_delay carrier  
  ## <int> <int> <dbl> <int> <int> <dbl> <chr>   
  ## 1 517 515 2 830 819 11 UA   
  ## 2 533 529 4 850 830 20 UA   
  ## 3 542 540 2 923 850 33 AA   
  ## 4 544 545 -1 1004 1022 -18 B6   
  ## 5 554 600 -6 812 837 -25 DL   
  ## 6 554 558 -4 740 728 12 UA   
  ## 7 555 600 -5 913 854 19 B6   
  ## 8 557 600 -3 709 723 -14 EV   
  ## 9 557 600 -3 838 846 -8 B6   
  ## 10 558 600 -2 753 745 8 AA   
  ## # ℹ 336,766 more rows  
  ## # ℹ 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
  ## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>
* Selecionar todas las comlunas que son caracteres
* flights |>  
   select(where(is.character))
* ## # A tibble: 336,776 × 4  
  ## carrier tailnum origin dest   
  ## <chr> <chr> <chr> <chr>  
  ## 1 UA N14228 EWR IAH   
  ## 2 UA N24211 LGA IAH   
  ## 3 AA N619AA JFK MIA   
  ## 4 B6 N804JB JFK BQN   
  ## 5 DL N668DN LGA ATL   
  ## 6 UA N39463 EWR ORD   
  ## 7 B6 N516JB EWR FLL   
  ## 8 EV N829AS LGA IAD   
  ## 9 B6 N593JB JFK MCO   
  ## 10 AA N3ALAA LGA ORD   
  ## # ℹ 336,766 more rows

Similar a is.character hay algunas funciones que nos ayudan a determinar los tipos de datos

* starts\_with("abc"): matches names that begin with “abc”.
* ends\_with("xyz"): matches names that end with “xyz”.
* contains("ijk"): matches names that contain “ijk”.
* num\_range("x", 1:3): matches x1, x2 and x3.

Se pueden renombrar variables usando select e =

flights |>  
 select(tail\_num = tailnum) # Nombre nuevo = Nombre anterior

## # A tibble: 336,776 × 1  
## tail\_num  
## <chr>   
## 1 N14228   
## 2 N24211   
## 3 N619AA   
## 4 N804JB   
## 5 N668DN   
## 6 N39463   
## 7 N516JB   
## 8 N829AS   
## 9 N593JB   
## 10 N3ALAA   
## # ℹ 336,766 more rows

### 3.3.3 rename()

Si a diferencia del anterior se requiere renombrar la columna sin eliminar las demás, se puede usar esta función

flights |>  
 rename(tail\_num = tailnum)

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tail\_num <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

# janitor::clean\_names() #Limpia lso nombres de las columnas

### 3.3.4 relocate()

Mueve variables. Por defecto las trae al frente

flights |>  
 relocate(time\_hour, air\_time)

## # A tibble: 336,776 × 19  
## time\_hour air\_time year month day dep\_time sched\_dep\_time  
## <dttm> <dbl> <int> <int> <int> <int> <int>  
## 1 2013-01-01 05:00:00 227 2013 1 1 517 515  
## 2 2013-01-01 05:00:00 227 2013 1 1 533 529  
## 3 2013-01-01 05:00:00 160 2013 1 1 542 540  
## 4 2013-01-01 05:00:00 183 2013 1 1 544 545  
## 5 2013-01-01 06:00:00 116 2013 1 1 554 600  
## 6 2013-01-01 05:00:00 150 2013 1 1 554 558  
## 7 2013-01-01 06:00:00 158 2013 1 1 555 600  
## 8 2013-01-01 06:00:00 53 2013 1 1 557 600  
## 9 2013-01-01 06:00:00 140 2013 1 1 557 600  
## 10 2013-01-01 06:00:00 138 2013 1 1 558 600  
## # ℹ 336,766 more rows  
## # ℹ 12 more variables: dep\_delay <dbl>, arr\_time <int>, sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,  
## # dest <chr>, distance <dbl>, hour <dbl>, minute <dbl>

Al igual que mutate() puedo usar .after y .before

flights |>  
 relocate(year:dep\_time, .after = time\_hour)

## # A tibble: 336,776 × 19  
## sched\_dep\_time dep\_delay arr\_time sched\_arr\_time arr\_delay carrier flight  
## <int> <dbl> <int> <int> <dbl> <chr> <int>  
## 1 515 2 830 819 11 UA 1545  
## 2 529 4 850 830 20 UA 1714  
## 3 540 2 923 850 33 AA 1141  
## 4 545 -1 1004 1022 -18 B6 725  
## 5 600 -6 812 837 -25 DL 461  
## 6 558 -4 740 728 12 UA 1696  
## 7 600 -5 913 854 19 B6 507  
## 8 600 -3 709 723 -14 EV 5708  
## 9 600 -3 838 846 -8 B6 79  
## 10 600 -2 753 745 8 AA 301  
## # ℹ 336,766 more rows  
## # ℹ 12 more variables: tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>, year <int>,  
## # month <int>, day <int>, dep\_time <int>

flights |>  
 relocate(starts\_with("arr"), .before = dep\_time)

## # A tibble: 336,776 × 19  
## year month day arr\_time arr\_delay dep\_time sched\_dep\_time dep\_delay  
## <int> <int> <int> <int> <dbl> <int> <int> <dbl>  
## 1 2013 1 1 830 11 517 515 2  
## 2 2013 1 1 850 20 533 529 4  
## 3 2013 1 1 923 33 542 540 2  
## 4 2013 1 1 1004 -18 544 545 -1  
## 5 2013 1 1 812 -25 554 600 -6  
## 6 2013 1 1 740 12 554 558 -4  
## 7 2013 1 1 913 19 555 600 -5  
## 8 2013 1 1 709 -14 557 600 -3  
## 9 2013 1 1 838 -8 557 600 -3  
## 10 2013 1 1 753 8 558 600 -2  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: sched\_arr\_time <int>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 relocate(dep\_time, sched\_dep\_time, dep\_delay)

## # A tibble: 336,776 × 19  
## dep\_time sched\_dep\_time dep\_delay year month day arr\_time sched\_arr\_time  
## <int> <int> <dbl> <int> <int> <int> <int> <int>  
## 1 517 515 2 2013 1 1 830 819  
## 2 533 529 4 2013 1 1 850 830  
## 3 542 540 2 2013 1 1 923 850  
## 4 544 545 -1 2013 1 1 1004 1022  
## 5 554 600 -6 2013 1 1 812 837  
## 6 554 558 -4 2013 1 1 740 728  
## 7 555 600 -5 2013 1 1 913 854  
## 8 557 600 -3 2013 1 1 709 723  
## 9 557 600 -3 2013 1 1 838 846  
## 10 558 600 -2 2013 1 1 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 select(dep\_time, dep\_delay, arr\_time)

## # A tibble: 336,776 × 3  
## dep\_time dep\_delay arr\_time  
## <int> <dbl> <int>  
## 1 517 2 830  
## 2 533 4 850  
## 3 542 2 923  
## 4 544 -1 1004  
## 5 554 -6 812  
## 6 554 -4 740  
## 7 555 -5 913  
## 8 557 -3 709  
## 9 557 -3 838  
## 10 558 -2 753  
## # ℹ 336,766 more rows

flights |>  
 select(starts\_with("dep"), starts\_with("arr"))

## # A tibble: 336,776 × 4  
## dep\_time dep\_delay arr\_time arr\_delay  
## <int> <dbl> <int> <dbl>  
## 1 517 2 830 11  
## 2 533 4 850 20  
## 3 542 2 923 33  
## 4 544 -1 1004 -18  
## 5 554 -6 812 -25  
## 6 554 -4 740 12  
## 7 555 -5 913 19  
## 8 557 -3 709 -14  
## 9 557 -3 838 -8  
## 10 558 -2 753 8  
## # ℹ 336,766 more rows

flights |>  
 select(dep\_time:arr\_time, -contains("sched"))

## # A tibble: 336,776 × 3  
## dep\_time dep\_delay arr\_time  
## <int> <dbl> <int>  
## 1 517 2 830  
## 2 533 4 850  
## 3 542 2 923  
## 4 544 -1 1004  
## 5 554 -6 812  
## 6 554 -4 740  
## 7 555 -5 913  
## 8 557 -3 709  
## 9 557 -3 838  
## 10 558 -2 753  
## # ℹ 336,766 more rows

flights |>  
 select(dep\_time, dep\_time, dep\_time, dep\_time)

## # A tibble: 336,776 × 1  
## dep\_time  
## <int>  
## 1 517  
## 2 533  
## 3 542  
## 4 544  
## 5 554  
## 6 554  
## 7 555  
## 8 557  
## 9 557  
## 10 558  
## # ℹ 336,766 more rows

flights |> select(contains("TIME")) # Contains ignora las mayúsculas

## # A tibble: 336,776 × 6  
## dep\_time sched\_dep\_time arr\_time sched\_arr\_time air\_time time\_hour   
## <int> <int> <int> <int> <dbl> <dttm>   
## 1 517 515 830 819 227 2013-01-01 05:00:00  
## 2 533 529 850 830 227 2013-01-01 05:00:00  
## 3 542 540 923 850 160 2013-01-01 05:00:00  
## 4 544 545 1004 1022 183 2013-01-01 05:00:00  
## 5 554 600 812 837 116 2013-01-01 06:00:00  
## 6 554 558 740 728 150 2013-01-01 05:00:00  
## 7 555 600 913 854 158 2013-01-01 06:00:00  
## 8 557 600 709 723 53 2013-01-01 06:00:00  
## 9 557 600 838 846 140 2013-01-01 06:00:00  
## 10 558 600 753 745 138 2013-01-01 06:00:00  
## # ℹ 336,766 more rows

flights |>  
 select(contains("TIME", ignore.case = FALSE)) # Para que tenga en cuenta las mayúsculas

## # A tibble: 336,776 × 0

flights |>  
 rename(air\_time\_min = air\_time) |>  
 relocate(air\_time\_min)

## # A tibble: 336,776 × 19  
## air\_time\_min year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <dbl> <int> <int> <int> <int> <int> <dbl> <int>  
## 1 227 2013 1 1 517 515 2 830  
## 2 227 2013 1 1 533 529 4 850  
## 3 160 2013 1 1 542 540 2 923  
## 4 183 2013 1 1 544 545 -1 1004  
## 5 116 2013 1 1 554 600 -6 812  
## 6 150 2013 1 1 554 558 -4 740  
## 7 158 2013 1 1 555 600 -5 913  
## 8 53 2013 1 1 557 600 -3 709  
## 9 140 2013 1 1 557 600 -3 838  
## 10 138 2013 1 1 558 600 -2 753  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

# flights |>  
# select(tailnum) |>  
# arrange(arr\_delay) #El error se da porque primero selecciono solo va columna tailnum, por lo que ya no hay más columnas para tener de referencia para ordenar las filas

## 3.4 The pipe

find the fastest flights to Houston’s IAH airport. Para eso se necesita usar filter(), mutate(), select(), and arrange()

flights |>  
 filter(dest == "IAH") |>  
 mutate(speed = distance / air\_time \* 60) |>  
 select(year:day, dep\_time, carrier, speed) |>  
 arrange(desc(speed))

## # A tibble: 7,198 × 6  
## year month day dep\_time carrier speed  
## <int> <int> <int> <int> <chr> <dbl>  
## 1 2013 7 9 707 UA 522.  
## 2 2013 8 27 1850 UA 521.  
## 3 2013 8 28 902 UA 519.  
## 4 2013 8 28 2122 UA 519.  
## 5 2013 6 11 1628 UA 515.  
## 6 2013 8 27 1017 UA 515.  
## 7 2013 8 27 1205 UA 515.  
## 8 2013 8 27 1758 UA 515.  
## 9 2013 9 27 521 UA 515.  
## 10 2013 8 28 625 UA 515.  
## # ℹ 7,188 more rows

La forma en que se vería el código anterior sin usar el pipe es la siguiente

arrange(  
 select(  
 mutate(  
 filter(  
 flights,  
 dest == "IAH"  
 ),  
 speed = distance / air\_time \* 60  
 ),  
 year:day, dep\_time, carrier, flight, speed  
 ),  
 desc(speed)  
)

## # A tibble: 7,198 × 7  
## year month day dep\_time carrier flight speed  
## <int> <int> <int> <int> <chr> <int> <dbl>  
## 1 2013 7 9 707 UA 226 522.  
## 2 2013 8 27 1850 UA 1128 521.  
## 3 2013 8 28 902 UA 1711 519.  
## 4 2013 8 28 2122 UA 1022 519.  
## 5 2013 6 11 1628 UA 1178 515.  
## 6 2013 8 27 1017 UA 333 515.  
## 7 2013 8 27 1205 UA 1421 515.  
## 8 2013 8 27 1758 UA 302 515.  
## 9 2013 9 27 521 UA 252 515.  
## 10 2013 8 28 625 UA 559 515.  
## # ℹ 7,188 more rows

O mediante objetos:

flights1 <- filter(flights, dest == "IAH")  
flights2 <- mutate(flights1, speed = distance / air\_time \* 60)  
flights3 <- select(flights2, year:day, dep\_time, carrier, flight, speed)  
arrange(flights3, desc(speed))

## # A tibble: 7,198 × 7  
## year month day dep\_time carrier flight speed  
## <int> <int> <int> <int> <chr> <int> <dbl>  
## 1 2013 7 9 707 UA 226 522.  
## 2 2013 8 27 1850 UA 1128 521.  
## 3 2013 8 28 902 UA 1711 519.  
## 4 2013 8 28 2122 UA 1022 519.  
## 5 2013 6 11 1628 UA 1178 515.  
## 6 2013 8 27 1017 UA 333 515.  
## 7 2013 8 27 1205 UA 1421 515.  
## 8 2013 8 27 1758 UA 302 515.  
## 9 2013 9 27 521 UA 252 515.  
## 10 2013 8 28 625 UA 559 515.  
## # ℹ 7,188 more rows

## 3.5 Groups

### 3.5.1 group\_by()

Divide el conjunto de datos en grupos significativos para el análisis:

flights |>  
 group\_by(month)

## # A tibble: 336,776 × 19  
## # Groups: month [12]  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Esto no cambia los datos pero se indica que están agrupados por mes(). Eso significa que las operaciones ahora funcionarán por mes.

### 3.5.2 summarize()

En conjunto con la función anterior podemos calcular el promedio con respecto a un tipo de dato agrupado. En este caso está grupado por meses.

flights |>  
 group\_by(month) |>  
 summarize(  
 avg\_delay = mean(dep\_delay)  
 )

## # A tibble: 12 × 2  
## month avg\_delay  
## <int> <dbl>  
## 1 1 NA  
## 2 2 NA  
## 3 3 NA  
## 4 4 NA  
## 5 5 NA  
## 6 6 NA  
## 7 7 NA  
## 8 8 NA  
## 9 9 NA  
## 10 10 NA  
## 11 11 NA  
## 12 12 NA

Nos devolvió resultados vacíos. Esto sucede porque hay algunas vacías y los valores van a tender a ese estado. Habría que ignorarlos de la siguiente forma:

flights |>  
 group\_by(month) |>  
 summarize(  
 avg\_delay = mean(dep\_delay, na.rm = TRUE)  
 )

## # A tibble: 12 × 2  
## month avg\_delay  
## <int> <dbl>  
## 1 1 10.0   
## 2 2 10.8   
## 3 3 13.2   
## 4 4 13.9   
## 5 5 13.0   
## 6 6 20.8   
## 7 7 21.7   
## 8 8 12.6   
## 9 9 6.72  
## 10 10 6.24  
## 11 11 5.44  
## 12 12 16.6

Se pueden crear varios resúmenes en una misma función sumarize(). Por ejemplo n() devuelve el número de filas en cada grupo:

flights |>  
 group\_by(month) |>  
 summarize(  
 avg\_delay = mean(dep\_delay, na.rm = TRUE),  
 n = n()  
 )

## # A tibble: 12 × 3  
## month avg\_delay n  
## <int> <dbl> <int>  
## 1 1 10.0 27004  
## 2 2 10.8 24951  
## 3 3 13.2 28834  
## 4 4 13.9 28330  
## 5 5 13.0 28796  
## 6 6 20.8 28243  
## 7 7 21.7 29425  
## 8 8 12.6 29327  
## 9 9 6.72 27574  
## 10 10 6.24 28889  
## 11 11 5.44 27268  
## 12 12 16.6 28135

### 3.5.3 The slice\_ functions

* df |> slice\_head(n = 1) takes the first row from each group.
* df |> slice\_tail(n = 1) takes the last row in each group.
* df |> slice\_min(x, n = 1) takes the row with the smallest value of column x.
* df |> slice\_max(x, n = 1) takes the row with the largest value of column x.
* df |> slice\_sample(n = 1) takes one random row.

Se puede seleccionar más de un valor, como por ejemplo seleccionar solo el 10%

flights |>  
 group\_by(dest) |>  
 slice\_max(arr\_delay, n = 1, with\_ties = FALSE) |>  
 relocate(dest)

## # A tibble: 105 × 19  
## # Groups: dest [105]  
## dest year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <chr> <int> <int> <int> <int> <int> <dbl> <int>  
## 1 ABQ 2013 7 22 2145 2007 98 132  
## 2 ACK 2013 7 23 1139 800 219 1250  
## 3 ALB 2013 1 25 123 2000 323 229  
## 4 ANC 2013 8 17 1740 1625 75 2042  
## 5 ATL 2013 7 22 2257 759 898 121  
## 6 AUS 2013 7 10 2056 1505 351 2347  
## 7 AVL 2013 8 13 1156 832 204 1417  
## 8 BDL 2013 2 21 1728 1316 252 1839  
## 9 BGR 2013 12 1 1504 1056 248 1628  
## 10 BHM 2013 4 10 25 1900 325 136  
## # ℹ 95 more rows  
## # ℹ 11 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Pero hay 108 filas, cuando en realidad solo ay 105 grupos. Lo que sucede es que se conservan los valores empatos por lo que hay que especificar que no queremos esos.

### 3.5.4 Agrupando en múltiples variables

You can create groups using more than one variable. For example, we could make a group for each date.

daily <- flights |>  
 group\_by(year, month, day)  
daily

## # A tibble: 336,776 × 19  
## # Groups: year, month, day [365]  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Cuando se resume un fragmento agrupado por más de una variable, cada resumen se desprende del último grupo. En retrospectiva, esta no fue una excelente manera de hacer que esta función funcione, pero es difícil de cambiar sin romper el código existente. Para que sea obvio lo que está sucediendo, dplyr muestra un mensaje que le indica cómo puede cambiar este comportamiento

# daily\_flights <- daily |>  
# summarize(n=n(  
# ))

daily\_flights <- daily |>  
 summarize(  
 n = n(),  
 .groups = "drop\_last"  
 )  
daily\_flights

## # A tibble: 365 × 4  
## # Groups: year, month [12]  
## year month day n  
## <int> <int> <int> <int>  
## 1 2013 1 1 842  
## 2 2013 1 2 943  
## 3 2013 1 3 914  
## 4 2013 1 4 915  
## 5 2013 1 5 720  
## 6 2013 1 6 832  
## 7 2013 1 7 933  
## 8 2013 1 8 899  
## 9 2013 1 9 902  
## 10 2013 1 10 932  
## # ℹ 355 more rows

### 3.5.5 Ungrouping

Remover grupos.

daily |>  
 ungroup()

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

daily |>  
 ungroup() |>  
 summarize(  
 avg\_delay = mean(dep\_delay, na.rm = TRUE),  
 flights = n()  
 )

## # A tibble: 1 × 2  
## avg\_delay flights  
## <dbl> <int>  
## 1 12.6 336776

### 3.5.6 .by

Es una alternativa a group\_by()

flights |>  
 summarize(  
 delay = mean(dep\_delay, na.rm = TRUE),  
 n = n(), # Numero de datos  
 .by = month  
 )

## # A tibble: 12 × 3  
## month delay n  
## <int> <dbl> <int>  
## 1 1 10.0 27004  
## 2 10 6.24 28889  
## 3 11 5.44 27268  
## 4 12 16.6 28135  
## 5 2 10.8 24951  
## 6 3 13.2 28834  
## 7 4 13.9 28330  
## 8 5 13.0 28796  
## 9 6 20.8 28243  
## 10 7 21.7 29425  
## 11 8 12.6 29327  
## 12 9 6.72 27574

flights |>  
 summarize(  
 delay = mean(dep\_delay, na.rm = TRUE),  
 n = n(),  
 .by = c(origin, dest)  
 )

## # A tibble: 224 × 4  
## origin dest delay n  
## <chr> <chr> <dbl> <int>  
## 1 EWR IAH 11.8 3973  
## 2 LGA IAH 9.06 2951  
## 3 JFK MIA 9.34 3314  
## 4 JFK BQN 6.67 599  
## 5 LGA ATL 11.4 10263  
## 6 EWR ORD 14.6 6100  
## 7 EWR FLL 13.5 3793  
## 8 LGA IAD 16.7 1803  
## 9 JFK MCO 10.6 5464  
## 10 LGA ORD 10.7 8857  
## # ℹ 214 more rows

flights |>  
 group\_by(carrier) |>  
 summarize(avg\_dep\_delay = mean(dep\_delay, na.rm = TRUE)) |>  
 arrange(desc(avg\_dep\_delay))

## # A tibble: 16 × 2  
## carrier avg\_dep\_delay  
## <chr> <dbl>  
## 1 F9 20.2   
## 2 EV 20.0   
## 3 YV 19.0   
## 4 FL 18.7   
## 5 WN 17.7   
## 6 9E 16.7   
## 7 B6 13.0   
## 8 VX 12.9   
## 9 OO 12.6   
## 10 UA 12.1   
## 11 MQ 10.6   
## 12 DL 9.26  
## 13 AA 8.59  
## 14 AS 5.80  
## 15 HA 4.90  
## 16 US 3.78

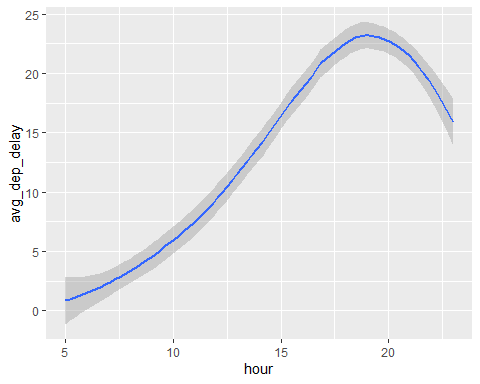
flights |>  
 group\_by(dest) |>  
 arrange(dest, desc(dep\_delay)) |>  
 slice\_head(n = 5) |>  
 relocate(dest, dep\_delay)

## # A tibble: 517 × 19  
## # Groups: dest [105]  
## dest dep\_delay year month day dep\_time sched\_dep\_time arr\_time  
## <chr> <dbl> <int> <int> <int> <int> <int> <int>  
## 1 ABQ 142 2013 12 14 2223 2001 133  
## 2 ABQ 139 2013 12 17 2220 2001 120  
## 3 ABQ 125 2013 7 30 2212 2007 57  
## 4 ABQ 125 2013 9 2 2212 2007 48  
## 5 ABQ 119 2013 7 23 2206 2007 116  
## 6 ACK 219 2013 7 23 1139 800 1250  
## 7 ACK 138 2013 7 2 1018 800 1119  
## 8 ACK 117 2013 7 4 957 800 1106  
## 9 ACK 101 2013 5 30 1321 1140 1419  
## 10 ACK 100 2013 6 24 940 800 1111  
## # ℹ 507 more rows  
## # ℹ 11 more variables: sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 group\_by(hour) |>  
 summarize(avg\_dep\_delay = mean(dep\_delay, na.rm = TRUE)) |>  
 ggplot(aes(x = hour, y = avg\_dep\_delay)) +  
 geom\_smooth()

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

## Warning: Removed 1 row containing non-finite outside the scale range  
## (`stat\_smooth()`).



flights |>  
 slice\_min(dep\_delay, n = -5) |> # Un valor negativo no provoca ningún cambio  
 relocate(dep\_delay)

## # A tibble: 336,776 × 19  
## dep\_delay year month day dep\_time sched\_dep\_time arr\_time sched\_arr\_time  
## <dbl> <int> <int> <int> <int> <int> <int> <int>  
## 1 -43 2013 12 7 2040 2123 40 2352  
## 2 -33 2013 2 3 2022 2055 2240 2338  
## 3 -32 2013 11 10 1408 1440 1549 1559  
## 4 -30 2013 1 11 1900 1930 2233 2243  
## 5 -27 2013 1 29 1703 1730 1947 1957  
## 6 -26 2013 8 9 729 755 1002 955  
## 7 -25 2013 10 23 1907 1932 2143 2143  
## 8 -25 2013 3 30 2030 2055 2213 2250  
## 9 -24 2013 3 2 1431 1455 1601 1631  
## 10 -24 2013 5 5 934 958 1225 1309  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 slice\_min(dep\_delay, n = 5) |>  
 relocate(dep\_delay)

## # A tibble: 5 × 19  
## dep\_delay year month day dep\_time sched\_dep\_time arr\_time sched\_arr\_time  
## <dbl> <int> <int> <int> <int> <int> <int> <int>  
## 1 -43 2013 12 7 2040 2123 40 2352  
## 2 -33 2013 2 3 2022 2055 2240 2338  
## 3 -32 2013 11 10 1408 1440 1549 1559  
## 4 -30 2013 1 11 1900 1930 2233 2243  
## 5 -27 2013 1 29 1703 1730 1947 1957  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 slice\_max(dep\_delay, n = -5) |>  
 relocate(dep\_delay)

## # A tibble: 336,776 × 19  
## dep\_delay year month day dep\_time sched\_dep\_time arr\_time sched\_arr\_time  
## <dbl> <int> <int> <int> <int> <int> <int> <int>  
## 1 1301 2013 1 9 641 900 1242 1530  
## 2 1137 2013 6 15 1432 1935 1607 2120  
## 3 1126 2013 1 10 1121 1635 1239 1810  
## 4 1014 2013 9 20 1139 1845 1457 2210  
## 5 1005 2013 7 22 845 1600 1044 1815  
## 6 960 2013 4 10 1100 1900 1342 2211  
## 7 911 2013 3 17 2321 810 135 1020  
## 8 899 2013 6 27 959 1900 1236 2226  
## 9 898 2013 7 22 2257 759 121 1026  
## 10 896 2013 12 5 756 1700 1058 2020  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

flights |>  
 slice\_max(dep\_delay, n = 5) |>  
 relocate(dep\_delay)

## # A tibble: 5 × 19  
## dep\_delay year month day dep\_time sched\_dep\_time arr\_time sched\_arr\_time  
## <dbl> <int> <int> <int> <int> <int> <int> <int>  
## 1 1301 2013 1 9 641 900 1242 1530  
## 2 1137 2013 6 15 1432 1935 1607 2120  
## 3 1126 2013 1 10 1121 1635 1239 1810  
## 4 1014 2013 9 20 1139 1845 1457 2210  
## 5 1005 2013 7 22 845 1600 1044 1815  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

df <- tibble(  
 x = 1:5,  
 y = c("a", "b", "a", "a", "b"),  
 z = c("K", "K", "L", "L", "K")  
)  
  
df |>  
 group\_by(y)

## # A tibble: 5 × 3  
## # Groups: y [2]  
## x y z   
## <int> <chr> <chr>  
## 1 1 a K   
## 2 2 b K   
## 3 3 a L   
## 4 4 a L   
## 5 5 b K

df |>  
 arrange(y)

## # A tibble: 5 × 3  
## x y z   
## <int> <chr> <chr>  
## 1 1 a K   
## 2 3 a L   
## 3 4 a L   
## 4 2 b K   
## 5 5 b K

df |>  
 group\_by(y) |>  
 summarize(mean\_x = mean(x))

## # A tibble: 2 × 2  
## y mean\_x  
## <chr> <dbl>  
## 1 a 2.67  
## 2 b 3.5

df |>  
 group\_by(y, z) |>  
 summarize(mean\_x = mean(x))

## `summarise()` has grouped output by 'y'. You can override using the `.groups`  
## argument.

## # A tibble: 3 × 3  
## # Groups: y [2]  
## y z mean\_x  
## <chr> <chr> <dbl>  
## 1 a K 1   
## 2 a L 3.5  
## 3 b K 3.5

df |>  
 group\_by(y, z) |>  
 summarize(mean\_x = mean(x), .groups = "drop")

## # A tibble: 3 × 3  
## y z mean\_x  
## <chr> <chr> <dbl>  
## 1 a K 1   
## 2 a L 3.5  
## 3 b K 3.5

df |>  
 group\_by(y, z) |>  
 summarize(mean\_x = mean(x))

## `summarise()` has grouped output by 'y'. You can override using the `.groups`  
## argument.

## # A tibble: 3 × 3  
## # Groups: y [2]  
## y z mean\_x  
## <chr> <chr> <dbl>  
## 1 a K 1   
## 2 a L 3.5  
## 3 b K 3.5

df |>  
 group\_by(y, z) |>  
 mutate(mean\_x = mean(x))

## # A tibble: 5 × 4  
## # Groups: y, z [3]  
## x y z mean\_x  
## <int> <chr> <chr> <dbl>  
## 1 1 a K 1   
## 2 2 b K 3.5  
## 3 3 a L 3.5  
## 4 4 a L 3.5  
## 5 5 b K 3.5

## **3.6 Case study: aggregates and sample size**

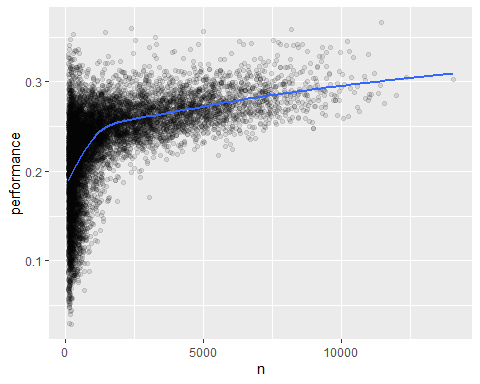
Siempre que realice una agregación, siempre es una buena idea incluir un recuento (). De esa manera, puede asegurarse de que no está sacando conclusiones basadas en cantidades muy pequeñas de datos.

batters <- Lahman::Batting |>  
 group\_by(playerID) |>  
 summarize(  
 performance = sum(H, na.rm = TRUE) / sum(AB, na.rm = TRUE),  
 n = sum(AB, na.rm = TRUE)  
 )  
batters

## # A tibble: 20,469 × 3  
## playerID performance n  
## <chr> <dbl> <int>  
## 1 aardsda01 0 4  
## 2 aaronha01 0.305 12364  
## 3 aaronto01 0.229 944  
## 4 aasedo01 0 5  
## 5 abadan01 0.0952 21  
## 6 abadfe01 0.111 9  
## 7 abadijo01 0.224 49  
## 8 abbated01 0.254 3044  
## 9 abbeybe01 0.169 225  
## 10 abbeych01 0.281 1756  
## # ℹ 20,459 more rows

batters |>  
 filter(n > 100) |>  
 ggplot(aes(x = n, y = performance)) +  
 geom\_point(alpha = 1 / 10) +  
 geom\_smooth(se = FALSE)

## `geom\_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'



batters |>  
 arrange(desc(performance))

## # A tibble: 20,469 × 3  
## playerID performance n  
## <chr> <dbl> <int>  
## 1 abramge01 1 1  
## 2 alberan01 1 1  
## 3 banisje01 1 1  
## 4 bartocl01 1 1  
## 5 bassdo01 1 1  
## 6 birasst01 1 2  
## 7 bruneju01 1 1  
## 8 burnscb01 1 1  
## 9 cammaer01 1 1  
## 10 campsh01 1 1  
## # ℹ 20,459 more rows

# 4. Flujo de trabajo: code style

install.packages("styler")

Ctrl + Shift + Py luego escribo “styler”

# 5. Data tidying

## 5.2 Tidy data

Se pueden representar los mismos datos de formas diferentes como se muestra a continuación

table1

## # A tibble: 6 × 4  
## country year cases population  
## <chr> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1999 745 19987071  
## 2 Afghanistan 2000 2666 20595360  
## 3 Brazil 1999 37737 172006362  
## 4 Brazil 2000 80488 174504898  
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

table2

## # A tibble: 12 × 4  
## country year type count  
## <chr> <dbl> <chr> <dbl>  
## 1 Afghanistan 1999 cases 745  
## 2 Afghanistan 1999 population 19987071  
## 3 Afghanistan 2000 cases 2666  
## 4 Afghanistan 2000 population 20595360  
## 5 Brazil 1999 cases 37737  
## 6 Brazil 1999 population 172006362  
## 7 Brazil 2000 cases 80488  
## 8 Brazil 2000 population 174504898  
## 9 China 1999 cases 212258  
## 10 China 1999 population 1272915272  
## 11 China 2000 cases 213766  
## 12 China 2000 population 1280428583

table3

## # A tibble: 6 × 3  
## country year rate   
## <chr> <dbl> <chr>   
## 1 Afghanistan 1999 745/19987071   
## 2 Afghanistan 2000 2666/20595360   
## 3 Brazil 1999 37737/172006362   
## 4 Brazil 2000 80488/174504898   
## 5 China 1999 212258/1272915272  
## 6 China 2000 213766/1280428583

These are all representations of the same underlying data, but they are not equally easy to use. One of them, table1, will be much easier to work with inside the tidyverse because it’s **tidy**.

Hay 3 reglas para hacer unos datos ordenados:

1. Each variable is a column; each column is a variable.
2. Each observation is a row; each row is an observation.
3. Each value is a cell; each cell is a single value.

# Compute rate per 10,000  
table1 |>  
 mutate(rate = cases / population \* 10000)

## # A tibble: 6 × 5  
## country year cases population rate  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1999 745 19987071 0.373  
## 2 Afghanistan 2000 2666 20595360 1.29   
## 3 Brazil 1999 37737 172006362 2.19   
## 4 Brazil 2000 80488 174504898 4.61   
## 5 China 1999 212258 1272915272 1.67   
## 6 China 2000 213766 1280428583 1.67

# Compute total cases per year  
table1 |>  
 group\_by(year) |>  
 summarize(total\_cases = sum(cases))

## # A tibble: 2 × 2  
## year total\_cases  
## <dbl> <dbl>  
## 1 1999 250740  
## 2 2000 296920

# Visualize changes over time  
ggplot(table1, aes(x = year, y = cases)) +  
 geom\_line(aes(group = country), color = "grey50") +  
 geom\_point(aes(color = country, shape = country)) +  
 scale\_x\_continuous(breaks = c(1999, 2000)) # x-axis breaks at 1999 and 2000

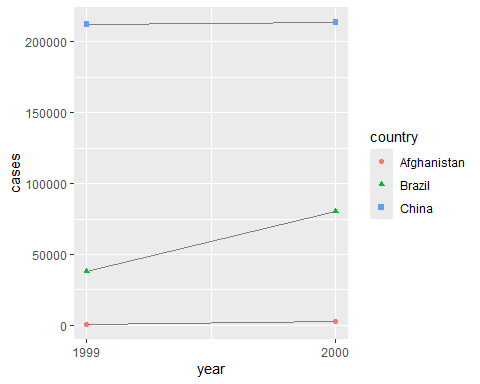


table2 |>  
 pivot\_wider(  
 names\_from = type,  
 values\_from = count  
 ) |>  
 mutate(rate = cases / population \* 10000)

## # A tibble: 6 × 5  
## country year cases population rate  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1999 745 19987071 0.373  
## 2 Afghanistan 2000 2666 20595360 1.29   
## 3 Brazil 1999 37737 172006362 2.19   
## 4 Brazil 2000 80488 174504898 4.61   
## 5 China 1999 212258 1272915272 1.67   
## 6 China 2000 213766 1280428583 1.67

#> # A tibble: 6 × 5  
#> country year cases population rate  
#> <chr> <dbl> <dbl> <dbl> <dbl>  
#> 1 Afghanistan 1999 745 19987071 0.373  
#> 2 Afghanistan 2000 2666 20595360 1.29  
#> 3 Brazil 1999 37737 172006362 2.19  
#> 4 Brazil 2000 80488 174504898 4.61  
#> 5 China 1999 212258 1272915272 1.67  
#> 6 China 2000 213766 1280428583 1.67

table3 |>  
 separate\_wider\_delim(  
 cols = rate,  
 delim = "/",  
 names = c("cases", "population"),  
 ) |>  
 mutate(  
 cases = as.numeric(cases),  
 population = as.numeric(population),  
 rate = cases / population \* 10000  
 )

## # A tibble: 6 × 5  
## country year cases population rate  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1999 745 19987071 0.373  
## 2 Afghanistan 2000 2666 20595360 1.29   
## 3 Brazil 1999 37737 172006362 2.19   
## 4 Brazil 2000 80488 174504898 4.61   
## 5 China 1999 212258 1272915272 1.67   
## 6 China 2000 213766 1280428583 1.67

#> # A tibble: 6 × 5  
#> country year cases population rate  
#> <chr> <dbl> <dbl> <dbl> <dbl>  
#> 1 Afghanistan 1999 745 19987071 0.373  
#> 2 Afghanistan 2000 2666 20595360 1.29  
#> 3 Brazil 1999 37737 172006362 2.19  
#> 4 Brazil 2000 80488 174504898 4.61  
#> 5 China 1999 212258 1272915272 1.67  
#> 6 China 2000 213766 1280428583 1.67

## **5.3 Lengthening data**

tidyr provides two functions for pivoting data: [pivot\_longer()](https://tidyr.tidyverse.org/reference/pivot_longer.html) and [pivot\_wider()](https://tidyr.tidyverse.org/reference/pivot_wider.html). We’ll first start with [pivot\_longer()](https://tidyr.tidyverse.org/reference/pivot_longer.html) because it’s the most common case. Let’s dive into some examples.

### **5.3.1 Data in column names**

billboard

## # A tibble: 317 × 79  
## artist track date.entered wk1 wk2 wk3 wk4 wk5 wk6 wk7 wk8  
## <chr> <chr> <date> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2 Pac Baby… 2000-02-26 87 82 72 77 87 94 99 NA  
## 2 2Ge+her The … 2000-09-02 91 87 92 NA NA NA NA NA  
## 3 3 Doors D… Kryp… 2000-04-08 81 70 68 67 66 57 54 53  
## 4 3 Doors D… Loser 2000-10-21 76 76 72 69 67 65 55 59  
## 5 504 Boyz Wobb… 2000-04-15 57 34 25 17 17 31 36 49  
## 6 98^0 Give… 2000-08-19 51 39 34 26 26 19 2 2  
## 7 A\*Teens Danc… 2000-07-08 97 97 96 95 100 NA NA NA  
## 8 Aaliyah I Do… 2000-01-29 84 62 51 41 38 35 35 38  
## 9 Aaliyah Try … 2000-03-18 59 53 38 28 21 18 16 14  
## 10 Adams, Yo… Open… 2000-08-26 76 76 74 69 68 67 61 58  
## # ℹ 307 more rows  
## # ℹ 68 more variables: wk9 <dbl>, wk10 <dbl>, wk11 <dbl>, wk12 <dbl>,  
## # wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>, wk18 <dbl>,  
## # wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>, wk24 <dbl>,  
## # wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>,  
## # wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>,  
## # wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, wk42 <dbl>, …

In this dataset, each observation is a song. The first three columns (artist, track and date.entered) are variables that describe the song. Then we have 76 columns (wk1-wk76) that describe the rank of the song in each week[1](https://r4ds.hadley.nz/data-tidy#fn1). Here, the column names are one variable (the week) and the cell values are another (the rank).

Estos datos se pueden organizar de la siguiente forma usando pivot\_longer(). De esta forma ya no tenemos 76 columnas de 76 semanas

billboard |>  
 pivot\_longer(  
 cols = starts\_with("wk"), # Selecciona las columnas  
 names\_to = "week", # Nombre de la nueva columna  
 values\_to = "rank", # envia los valores de las columnas a na nueva columna llamada rank  
 values\_drop\_na = TRUE # Suprimo los NA  
 )

## # A tibble: 5,307 × 5  
## artist track date.entered week rank  
## <chr> <chr> <date> <chr> <dbl>  
## 1 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk1 87  
## 2 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk2 82  
## 3 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk3 72  
## 4 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk4 77  
## 5 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk5 87  
## 6 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk6 94  
## 7 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk7 99  
## 8 2Ge+her The Hardest Part Of ... 2000-09-02 wk1 91  
## 9 2Ge+her The Hardest Part Of ... 2000-09-02 wk2 87  
## 10 2Ge+her The Hardest Part Of ... 2000-09-02 wk3 92  
## # ℹ 5,297 more rows

Podemos quitar el valor de de w en la columna y dejar solo el número de la semana

billboard\_longer <- billboard |>  
 pivot\_longer(  
 cols = starts\_with("wk"),  
 names\_to = "week",  
 values\_to = "rank",  
 values\_drop\_na = TRUE  
 ) |>  
 mutate(  
 week = parse\_number(week) # Extrae solo el número  
 )  
billboard\_longer

## # A tibble: 5,307 × 5  
## artist track date.entered week rank  
## <chr> <chr> <date> <dbl> <dbl>  
## 1 2 Pac Baby Don't Cry (Keep... 2000-02-26 1 87  
## 2 2 Pac Baby Don't Cry (Keep... 2000-02-26 2 82  
## 3 2 Pac Baby Don't Cry (Keep... 2000-02-26 3 72  
## 4 2 Pac Baby Don't Cry (Keep... 2000-02-26 4 77  
## 5 2 Pac Baby Don't Cry (Keep... 2000-02-26 5 87  
## 6 2 Pac Baby Don't Cry (Keep... 2000-02-26 6 94  
## 7 2 Pac Baby Don't Cry (Keep... 2000-02-26 7 99  
## 8 2Ge+her The Hardest Part Of ... 2000-09-02 1 91  
## 9 2Ge+her The Hardest Part Of ... 2000-09-02 2 87  
## 10 2Ge+her The Hardest Part Of ... 2000-09-02 3 92  
## # ℹ 5,297 more rows

Si graficamos:

billboard\_longer |>  
 ggplot(aes(x = week, y = rank, group = track)) +  
 geom\_line(alpha = 0.25) +  
 scale\_y\_reverse()



### **5.3.2 How does pivoting work?**

Suppose we have three patients with ids A, B, and C, and we take two blood pressure measurements on each patient. We’ll create the data with [tribble()](https://tibble.tidyverse.org/reference/tribble.html), a handy function for constructing small tibbles by hand:

df <- tribble(  
 ~id, ~bp1, ~bp2,  
 "A", 100, 120,  
 "B", 140, 115,  
 "C", 120, 125  
)

We want our new dataset to have three variables: id (already exists), measurement (the column names), and value (the cell values). To achieve this, we need to pivot df longer

df |>  
 pivot\_longer(  
 cols = bp1:bp2,  
 names\_to = "measurement",  
 values\_to = "value"  
 )

## # A tibble: 6 × 3  
## id measurement value  
## <chr> <chr> <dbl>  
## 1 A bp1 100  
## 2 A bp2 120  
## 3 B bp1 140  
## 4 B bp2 115  
## 5 C bp1 120  
## 6 C bp2 125

### **5.3.3 Many variables in column names**

A veces las columnas pueden tener varias piezas de información que puede ser util como el siguiente dataser

who2

## # A tibble: 7,240 × 58  
## country year sp\_m\_014 sp\_m\_1524 sp\_m\_2534 sp\_m\_3544 sp\_m\_4554 sp\_m\_5564  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1980 NA NA NA NA NA NA  
## 2 Afghanistan 1981 NA NA NA NA NA NA  
## 3 Afghanistan 1982 NA NA NA NA NA NA  
## 4 Afghanistan 1983 NA NA NA NA NA NA  
## 5 Afghanistan 1984 NA NA NA NA NA NA  
## 6 Afghanistan 1985 NA NA NA NA NA NA  
## 7 Afghanistan 1986 NA NA NA NA NA NA  
## 8 Afghanistan 1987 NA NA NA NA NA NA  
## 9 Afghanistan 1988 NA NA NA NA NA NA  
## 10 Afghanistan 1989 NA NA NA NA NA NA  
## # ℹ 7,230 more rows  
## # ℹ 50 more variables: sp\_m\_65 <dbl>, sp\_f\_014 <dbl>, sp\_f\_1524 <dbl>,  
## # sp\_f\_2534 <dbl>, sp\_f\_3544 <dbl>, sp\_f\_4554 <dbl>, sp\_f\_5564 <dbl>,  
## # sp\_f\_65 <dbl>, sn\_m\_014 <dbl>, sn\_m\_1524 <dbl>, sn\_m\_2534 <dbl>,  
## # sn\_m\_3544 <dbl>, sn\_m\_4554 <dbl>, sn\_m\_5564 <dbl>, sn\_m\_65 <dbl>,  
## # sn\_f\_014 <dbl>, sn\_f\_1524 <dbl>, sn\_f\_2534 <dbl>, sn\_f\_3544 <dbl>,  
## # sn\_f\_4554 <dbl>, sn\_f\_5564 <dbl>, sn\_f\_65 <dbl>, ep\_m\_014 <dbl>, …

La primera pieza, sp/rel/ep, describe el método utilizado para el diagnóstico, la segunda pieza, m/f, es el género (codificado como una variable binaria en este conjunto de datos), y la tercera pieza, 014/1524/2534/3544/4554/5564/65, es el rango de edad (014 representa 0-14, por ejemplo)

who2 |>  
 pivot\_longer(  
 cols = !(country:year),  
 names\_to = c("diagnosis", "gender", "age"), # Nuevas columnas  
 names\_sep = "\_", # La división de las nuevas columnas  
 values\_to = "count"  
 )

## # A tibble: 405,440 × 6  
## country year diagnosis gender age count  
## <chr> <dbl> <chr> <chr> <chr> <dbl>  
## 1 Afghanistan 1980 sp m 014 NA  
## 2 Afghanistan 1980 sp m 1524 NA  
## 3 Afghanistan 1980 sp m 2534 NA  
## 4 Afghanistan 1980 sp m 3544 NA  
## 5 Afghanistan 1980 sp m 4554 NA  
## 6 Afghanistan 1980 sp m 5564 NA  
## 7 Afghanistan 1980 sp m 65 NA  
## 8 Afghanistan 1980 sp f 014 NA  
## 9 Afghanistan 1980 sp f 1524 NA  
## 10 Afghanistan 1980 sp f 2534 NA  
## # ℹ 405,430 more rows

### **5.3.4 Data and variable names in the column headers**

household # Contiene valores de variables y nombres de variables

## # A tibble: 5 × 5  
## family dob\_child1 dob\_child2 name\_child1 name\_child2  
## <int> <date> <date> <chr> <chr>   
## 1 1 1998-11-26 2000-01-29 Susan Jose   
## 2 2 1996-06-22 NA Mark <NA>   
## 3 3 2002-07-11 2004-04-05 Sam Seth   
## 4 4 2004-10-10 2009-08-27 Craig Khai   
## 5 5 2000-12-05 2005-02-28 Parker Gracie

household |>  
 pivot\_longer(  
 cols = !family,  
 names\_to = c(".value", "child"),  
 names\_sep = "\_",  
 values\_drop\_na = TRUE  
 )

## # A tibble: 9 × 4  
## family child dob name   
## <int> <chr> <date> <chr>   
## 1 1 child1 1998-11-26 Susan   
## 2 1 child2 2000-01-29 Jose   
## 3 2 child1 1996-06-22 Mark   
## 4 3 child1 2002-07-11 Sam   
## 5 3 child2 2004-04-05 Seth   
## 6 4 child1 2004-10-10 Craig   
## 7 4 child2 2009-08-27 Khai   
## 8 5 child1 2000-12-05 Parker  
## 9 5 child2 2005-02-28 Gracie

## 5.4 Widening data

Es realizar todo lo contrario a lo anterior. Pasar datos de las filas a columnas

cms\_patient\_experience

## # A tibble: 500 × 5  
## org\_pac\_id org\_nm measure\_cd measure\_title prf\_rate  
## <chr> <chr> <chr> <chr> <dbl>  
## 1 0446157747 USC CARE MEDICAL GROUP INC CAHPS\_GRP… CAHPS for MI… 63  
## 2 0446157747 USC CARE MEDICAL GROUP INC CAHPS\_GRP… CAHPS for MI… 87  
## 3 0446157747 USC CARE MEDICAL GROUP INC CAHPS\_GRP… CAHPS for MI… 86  
## 4 0446157747 USC CARE MEDICAL GROUP INC CAHPS\_GRP… CAHPS for MI… 57  
## 5 0446157747 USC CARE MEDICAL GROUP INC CAHPS\_GRP… CAHPS for MI… 85  
## 6 0446157747 USC CARE MEDICAL GROUP INC CAHPS\_GRP… CAHPS for MI… 24  
## 7 0446162697 ASSOCIATION OF UNIVERSITY PHYSI… CAHPS\_GRP… CAHPS for MI… 59  
## 8 0446162697 ASSOCIATION OF UNIVERSITY PHYSI… CAHPS\_GRP… CAHPS for MI… 85  
## 9 0446162697 ASSOCIATION OF UNIVERSITY PHYSI… CAHPS\_GRP… CAHPS for MI… 83  
## 10 0446162697 ASSOCIATION OF UNIVERSITY PHYSI… CAHPS\_GRP… CAHPS for MI… 63  
## # ℹ 490 more rows

A continuación se puede observar que en realidad solo hay 6 datos distintos

cms\_patient\_experience |>   
 distinct(measure\_cd, measure\_title)

## # A tibble: 6 × 2  
## measure\_cd measure\_title   
## <chr> <chr>   
## 1 CAHPS\_GRP\_1 CAHPS for MIPS SSM: Getting Timely Care, Appointments, and Infor…  
## 2 CAHPS\_GRP\_2 CAHPS for MIPS SSM: How Well Providers Communicate   
## 3 CAHPS\_GRP\_3 CAHPS for MIPS SSM: Patient's Rating of Provider   
## 4 CAHPS\_GRP\_5 CAHPS for MIPS SSM: Health Promotion and Education   
## 5 CAHPS\_GRP\_8 CAHPS for MIPS SSM: Courteous and Helpful Office Staff   
## 6 CAHPS\_GRP\_12 CAHPS for MIPS SSM: Stewardship of Patient Resources

cms\_patient\_experience |>   
 pivot\_wider(  
 names\_from = measure\_cd,  
 values\_from = prf\_rate  
 )

## # A tibble: 500 × 9  
## org\_pac\_id org\_nm measure\_title CAHPS\_GRP\_1 CAHPS\_GRP\_2 CAHPS\_GRP\_3  
## <chr> <chr> <chr> <dbl> <dbl> <dbl>  
## 1 0446157747 USC CARE MEDICA… CAHPS for MI… 63 NA NA  
## 2 0446157747 USC CARE MEDICA… CAHPS for MI… NA 87 NA  
## 3 0446157747 USC CARE MEDICA… CAHPS for MI… NA NA 86  
## 4 0446157747 USC CARE MEDICA… CAHPS for MI… NA NA NA  
## 5 0446157747 USC CARE MEDICA… CAHPS for MI… NA NA NA  
## 6 0446157747 USC CARE MEDICA… CAHPS for MI… NA NA NA  
## 7 0446162697 ASSOCIATION OF … CAHPS for MI… 59 NA NA  
## 8 0446162697 ASSOCIATION OF … CAHPS for MI… NA 85 NA  
## 9 0446162697 ASSOCIATION OF … CAHPS for MI… NA NA 83  
## 10 0446162697 ASSOCIATION OF … CAHPS for MI… NA NA NA  
## # ℹ 490 more rows  
## # ℹ 3 more variables: CAHPS\_GRP\_5 <dbl>, CAHPS\_GRP\_8 <dbl>, CAHPS\_GRP\_12 <dbl>

cms\_patient\_experience |>   
 pivot\_wider(  
 id\_cols = starts\_with("org"),  
 names\_from = measure\_cd,  
 values\_from = prf\_rate  
 )

## # A tibble: 95 × 8  
## org\_pac\_id org\_nm CAHPS\_GRP\_1 CAHPS\_GRP\_2 CAHPS\_GRP\_3 CAHPS\_GRP\_5 CAHPS\_GRP\_8  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0446157747 USC C… 63 87 86 57 85  
## 2 0446162697 ASSOC… 59 85 83 63 88  
## 3 0547164295 BEAVE… 49 NA 75 44 73  
## 4 0749333730 CAPE … 67 84 85 65 82  
## 5 0840104360 ALLIA… 66 87 87 64 87  
## 6 0840109864 REX H… 73 87 84 67 91  
## 7 0840513552 SCL H… 58 83 76 58 78  
## 8 0941545784 GRITM… 46 86 81 54 NA  
## 9 1052612785 COMMU… 65 84 80 58 87  
## 10 1254237779 OUR L… 61 NA NA 65 NA  
## # ℹ 85 more rows  
## # ℹ 1 more variable: CAHPS\_GRP\_12 <dbl>

ejemplo de funcionaiento más sencillo sería:

df <- tribble(  
 ~id, ~measurement, ~value,  
 "A", "bp1", 100,  
 "B", "bp1", 140,  
 "B", "bp2", 115,   
 "A", "bp2", 120,  
 "A", "bp3", 105  
)

df |>   
 pivot\_wider(  
 names\_from = measurement,  
 values\_from = value  
 )

## # A tibble: 2 × 4  
## id bp1 bp2 bp3  
## <chr> <dbl> <dbl> <dbl>  
## 1 A 100 120 105  
## 2 B 140 115 NA

df |>   
 distinct(measurement) |>   
 pull()

## [1] "bp1" "bp2" "bp3"

df |>   
 select(-measurement, -value) |>   
 distinct()

## # A tibble: 2 × 1  
## id   
## <chr>  
## 1 A   
## 2 B

df |>   
 select(-measurement, -value) |>   
 distinct() |>   
 mutate(x = NA, y = NA, z = NA)

## # A tibble: 2 × 4  
## id x y z   
## <chr> <lgl> <lgl> <lgl>  
## 1 A NA NA NA   
## 2 B NA NA NA

usethis::use\_blank\_slate()

A partir de este punto se trabajará el código a través de un proyecto (Aunque probablemente tome apuntes en este lugar para llevar un buen control)