Clase 4

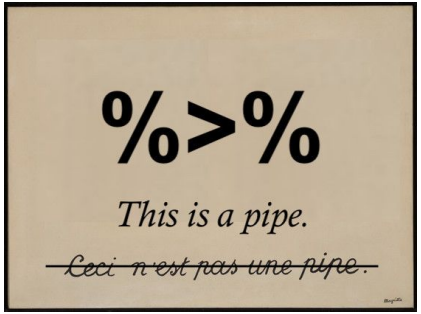
unab

VISUALIZACIÓN DE LA INFORMACIÓN 2024

Trabajar con datos - dplyr

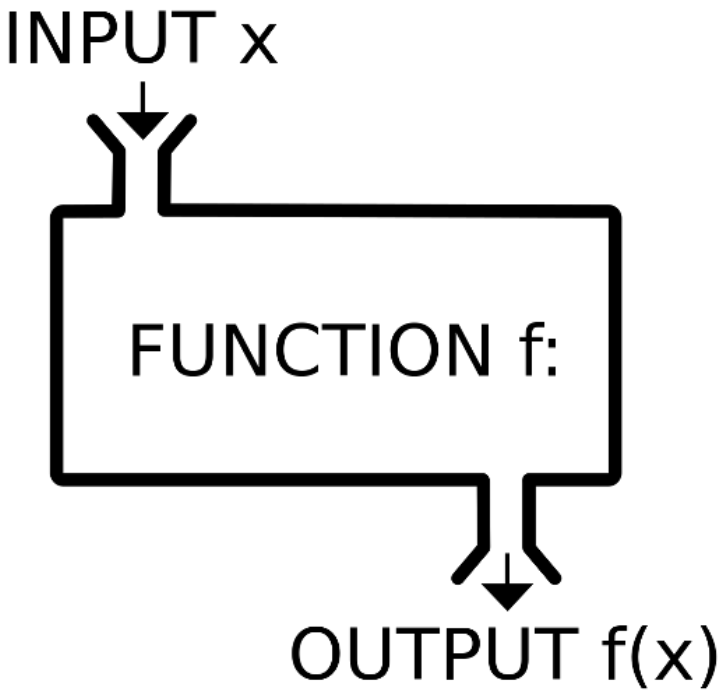
dplyr (data player) es una librería de R que nos permite seleccionar partes de ellos, filtrarlos, agruparlos y modificarlos.
Parte de un paradigma de operacion de datos que se denomina **gramática de datos**.
Para los programadores resultara como una forma procedural de hacer consultas **SQL** (lenguaje descriptivo)
Consta de **verbos** que realizan operaciones con los datos.

E incorpora otra librería que nos permite la concatenación de funciones, convirtiendo a R en un **lenguaje funcional**.



Lenguaje funcional

- Es un paradigma de programación.
- TODO es una función.
- Con una entrada y una salida.
- R es multiparadigma : (procedural, objetos, funcional, etc)
- El operador que concatena es %>% (Hay otros, ojo)



Verbos de Dplyr

- **Por Filas**:
  - `filter()` selecciona filas segun la condicion que cumplan los valores en algunas columnas.
  - `slice()` selecciona un numero de filas.
  - `arrange()` cambia el orden de las filas.
- **Por Columnas:**
  - `select()` decide que columnas son elegidas.
  - `rename()` cambia el nombre de las columnas.
  - `mutate()` cambia el valor de las columnas o crea nuevas.
  - `relocate()` reordena las columnas.
- **Agrupaciones:**
  - `summarise()` opera sobre los grupos aplicandoles funciones.
  - `group_by()` marca filas según alguna condición formando un grupo

Por columnas - select

Es lo más sencillo que podemos hacer.
Creamos un nuevo dataframe usando solo algunas columnas de otro.
Se pueden seleccionar por nombre o por posición.

```
nuevo <- original %>%
select(segunda, tercera, quinta)

nuevo <- original %>%
select(c(2:3,5))
```

Hay *helpers* que nos ayudan a simplificar la búsqueda de columnas: `starts_with()`, `ends_with()`, `matches()` o `contains()`

Con ?select tenemos la ayuda.

Por columnas - rename

Cambia el nombre de una columna

```
nuevo <- original %>% rename(second = segunda)
```

Por columnas - relocate

Cambia la posición de una columna

```
nuevo <- original %>% relocate(segunda) # segunda queda primera
nuevo <- original %>% relocate(segunda, after=primera) # segunda
queda segunda de vuelta
nuevo <- original %>% relocate(cuarta, before=tercera) # cuarta
queda tercera
```

Por columnas - mutate

Agrega una nueva columna a partir de datos de otra u otras columnas o de un valor calculado u obtenido por código.

```
nuevo <- original %>% mutate(total =
precio * unidades)
```

```
nuevo <- original %>%
mutate(peso_kilos = peso_gramos /
1000))
```

Por filas - filter

Genera un nuevo dataframe con info de filas seleccionadas por alguna condición.

```
nuevo <- original %>% filter(edad >= 25)
nuevo <- original %>% filter(edad >= 25, sexo="Masculino")
nuevo <- original %>% filter(altura = min(altura))
nuevo <- original %>% filter(n() == 1)
```

Por filas - arrange

Genera un nuevo dataframe con las filas ordenadas según algún criterio.

```
nuevo <- original %>% arrange(desc(edad))
nuevo <- original %>% arrange(desc(edad), asc(altura))
```

Agrupamientos - group_by

"Marca" con un metadato todas las filas que cumplan con algún criterio o que tengan un valor específico.

```
nuevo <- original %>% group_by(estudios)
nuevo <- original %>% group_by(estudios, puestos)
nuevo %>% tally()   # muestra los grupos marcados y su cantidad
```

col 1	col 2	col 3...l
group_a	group_a.1	other data
		other data
		other data
	group_a.2	other data
		other data
		other data
	group_a.3	other data
		other data
		other data
group_b	group_b.1	other data
		other data
		other data
	group_b.2	other data
		other data
		other data
	group_b.3	other data
		other data
		other data

Agrupamientos - summarise

Cuando ya tenemos las filas *marcadas* por `group_by()` podemos operar con esos grupos

```
nuevo <- original %>% group_by(estudios)
reporte <- nuevo %>% summarise(cantidad = n(),
                               sueldo_medio = mean(sueldo))
```

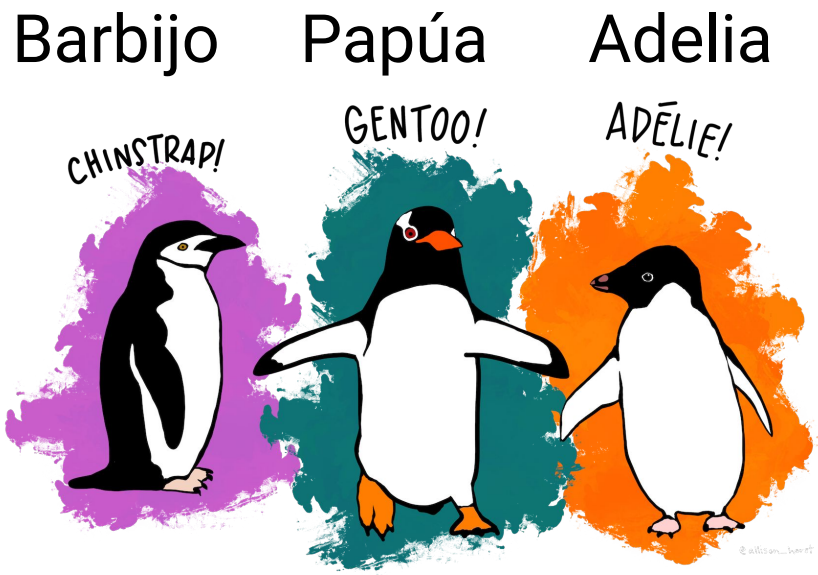
estudios	**cantidad**	**sueldo_medio**
primarios	30	350000
secundarios	22	870000
universitarios	7	1407000

¡Descanso de 10 minutos!!

No se desconecten pero retirense de las pantallas

Visualizar cantidades

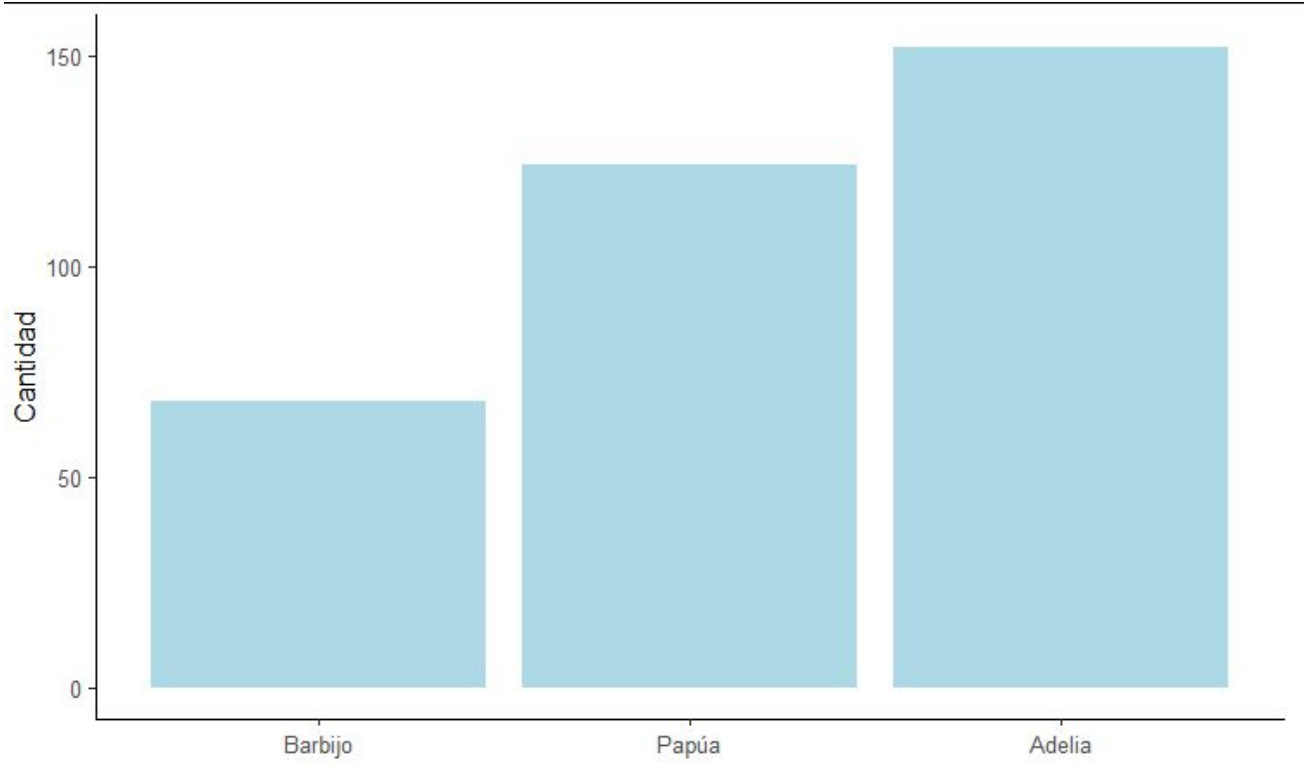
```
install::packages("datos")
pinguinos <- datos::pinguinos
```



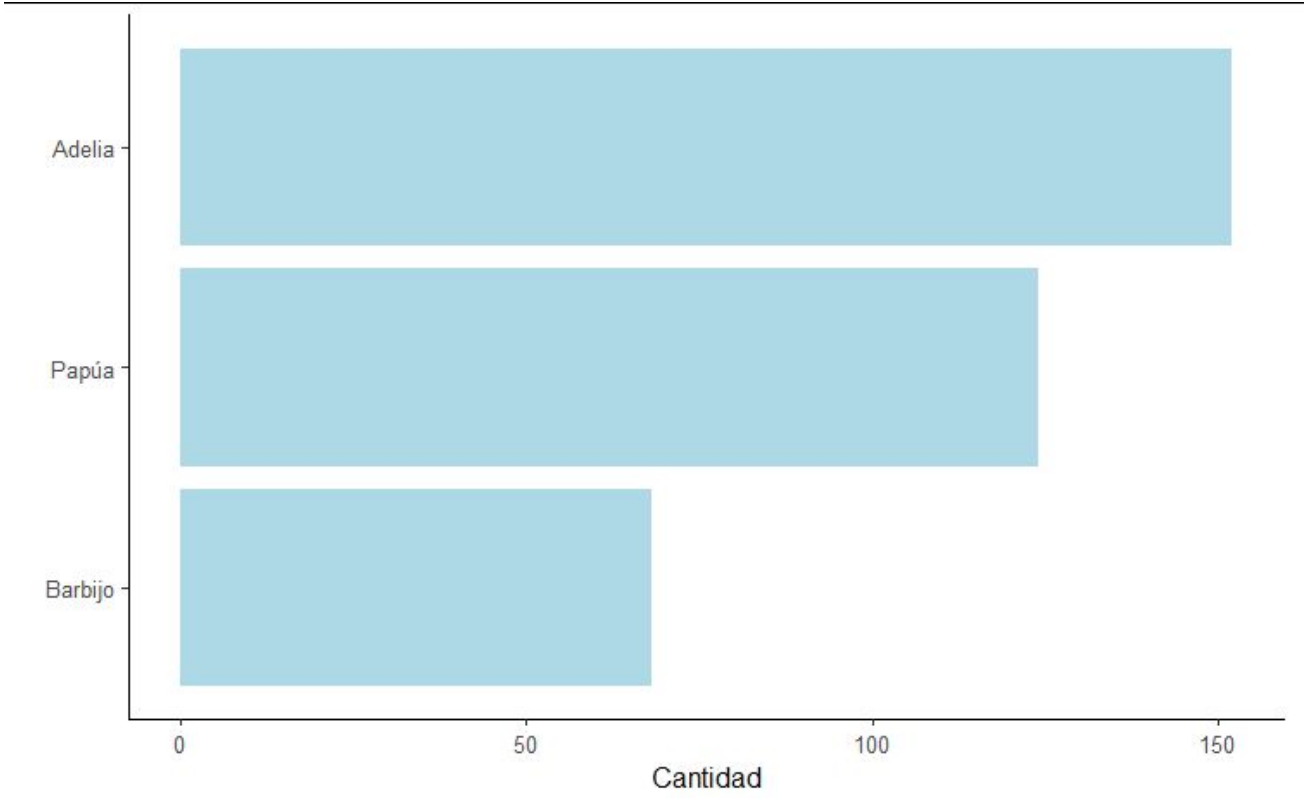
Conjuntos de datos con cantidades calculadas

```
# A tibble: 3 x 2
  especie     n
  <fct>   <int>
1 Adelia      152
2 Barbijo  68
3 Papúa       124
```

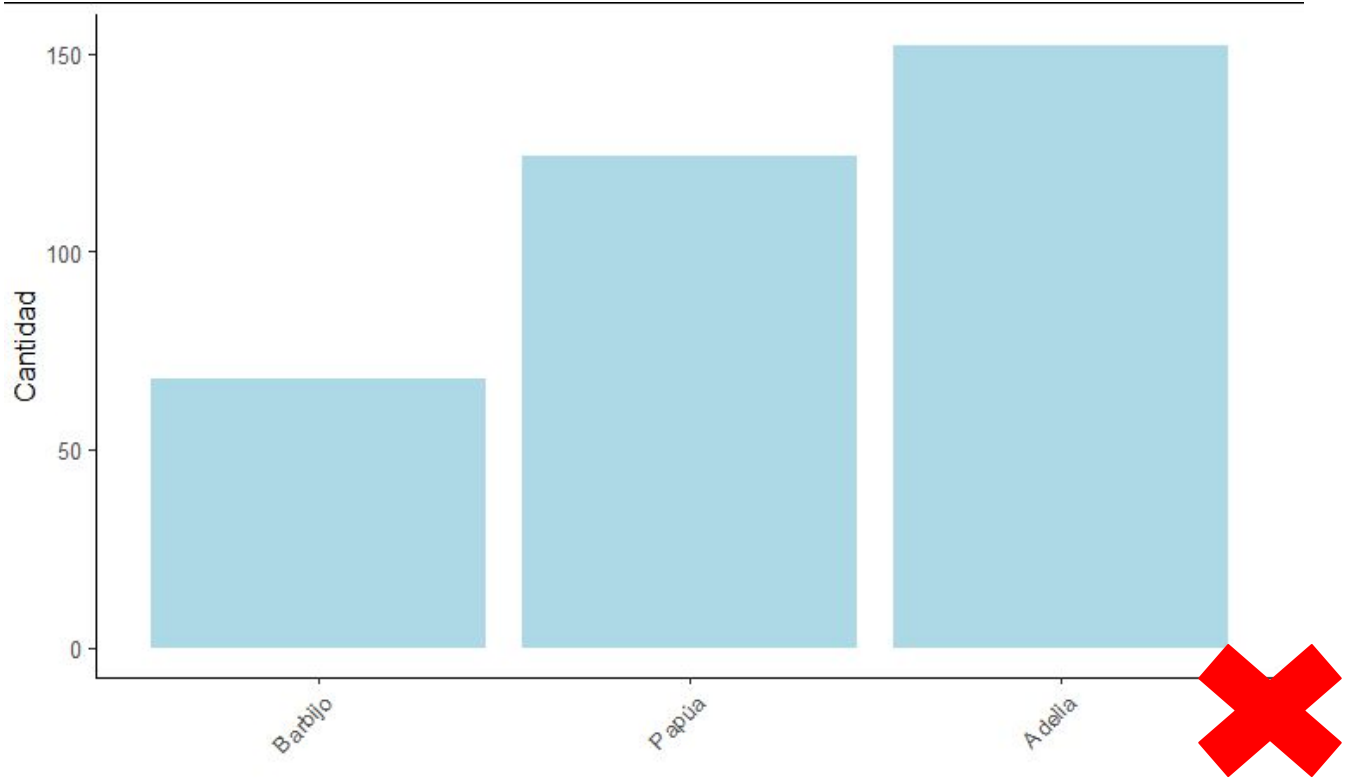
Podemos visualizar cantidades con barras



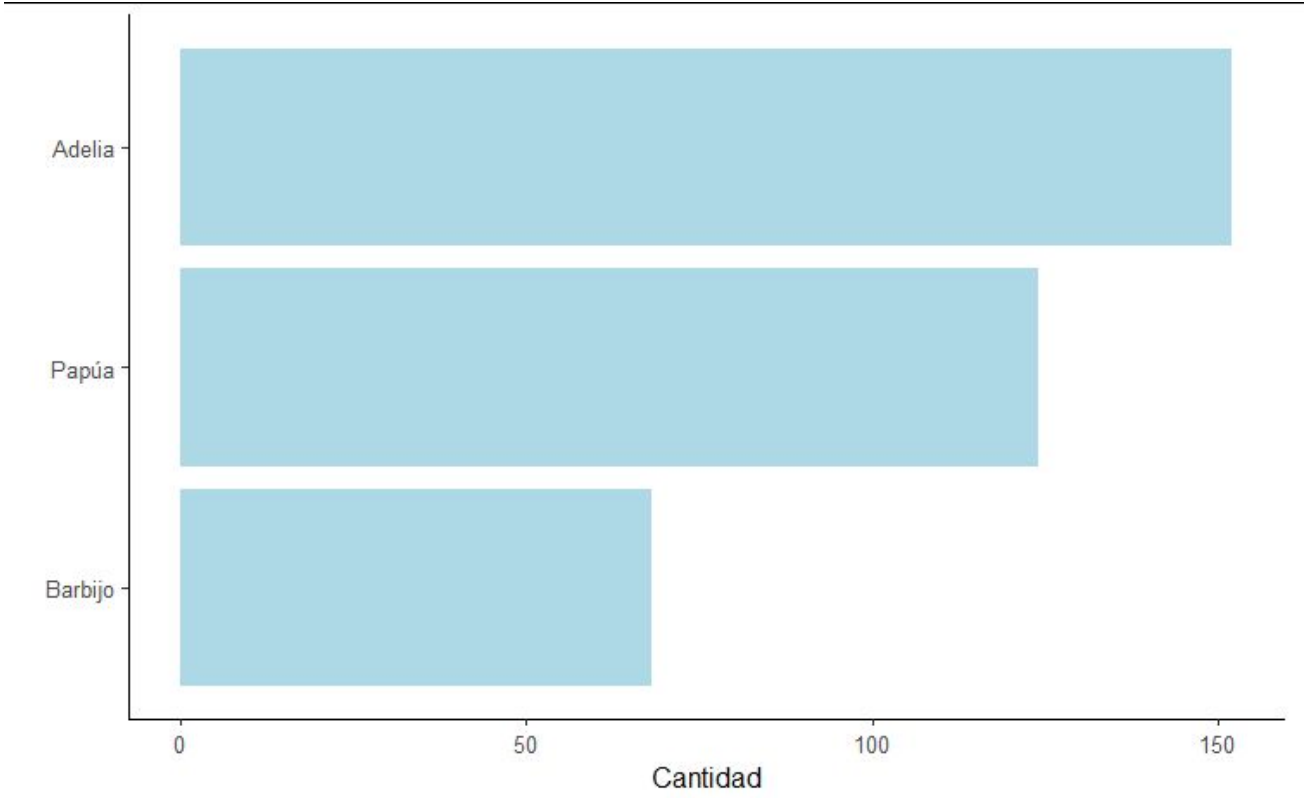
Las barras pueden ser horizontales



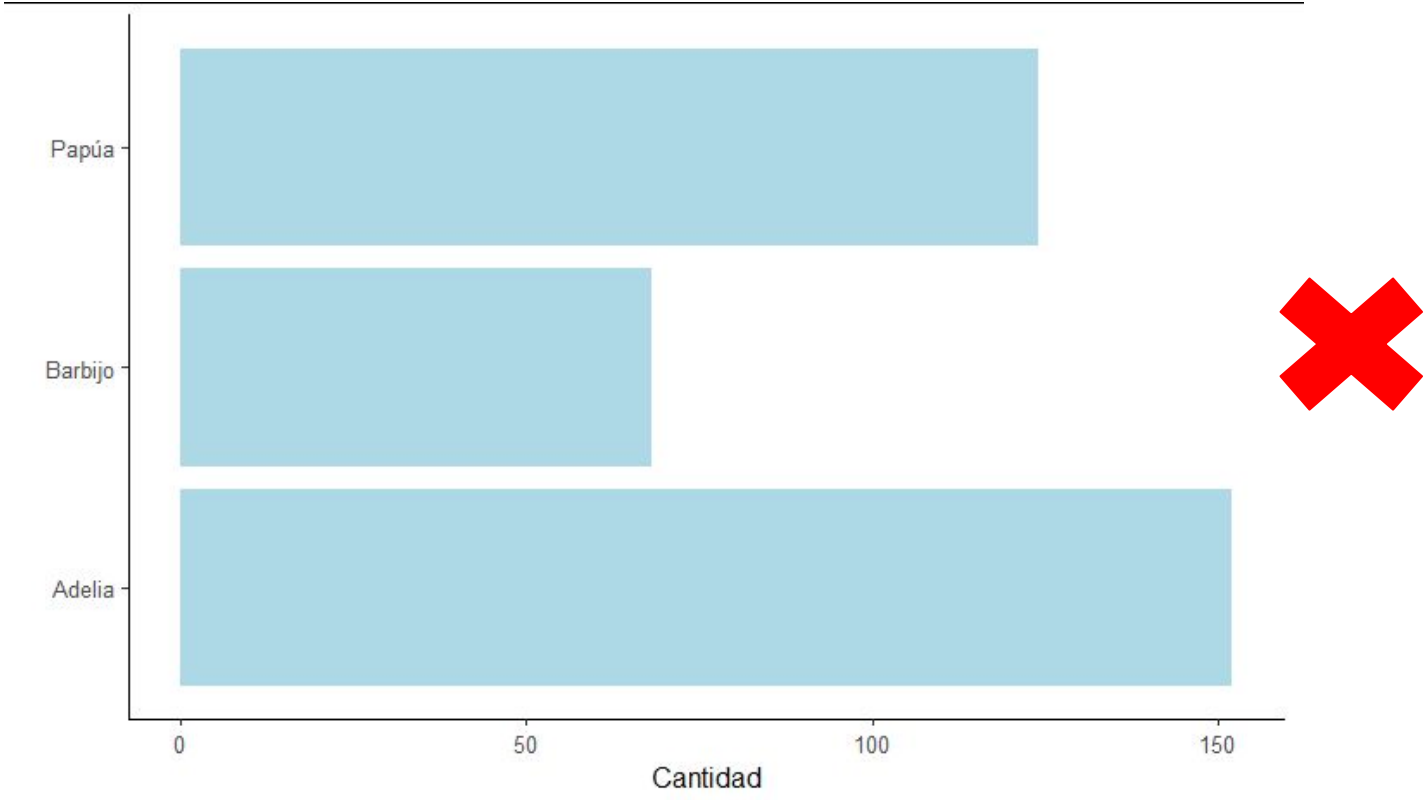
Evitar rotar las etiquetas de los ejes



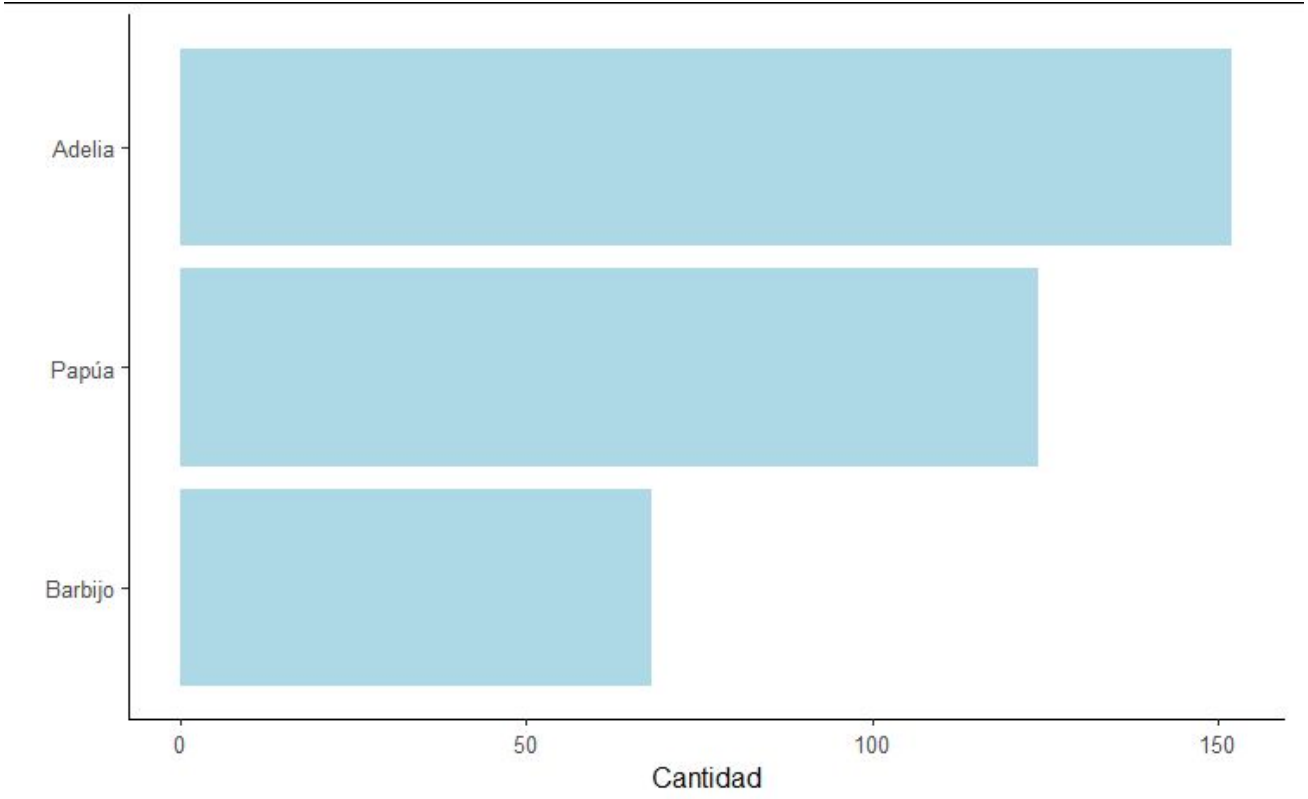
Para eso es buena idea usar barras horizontales



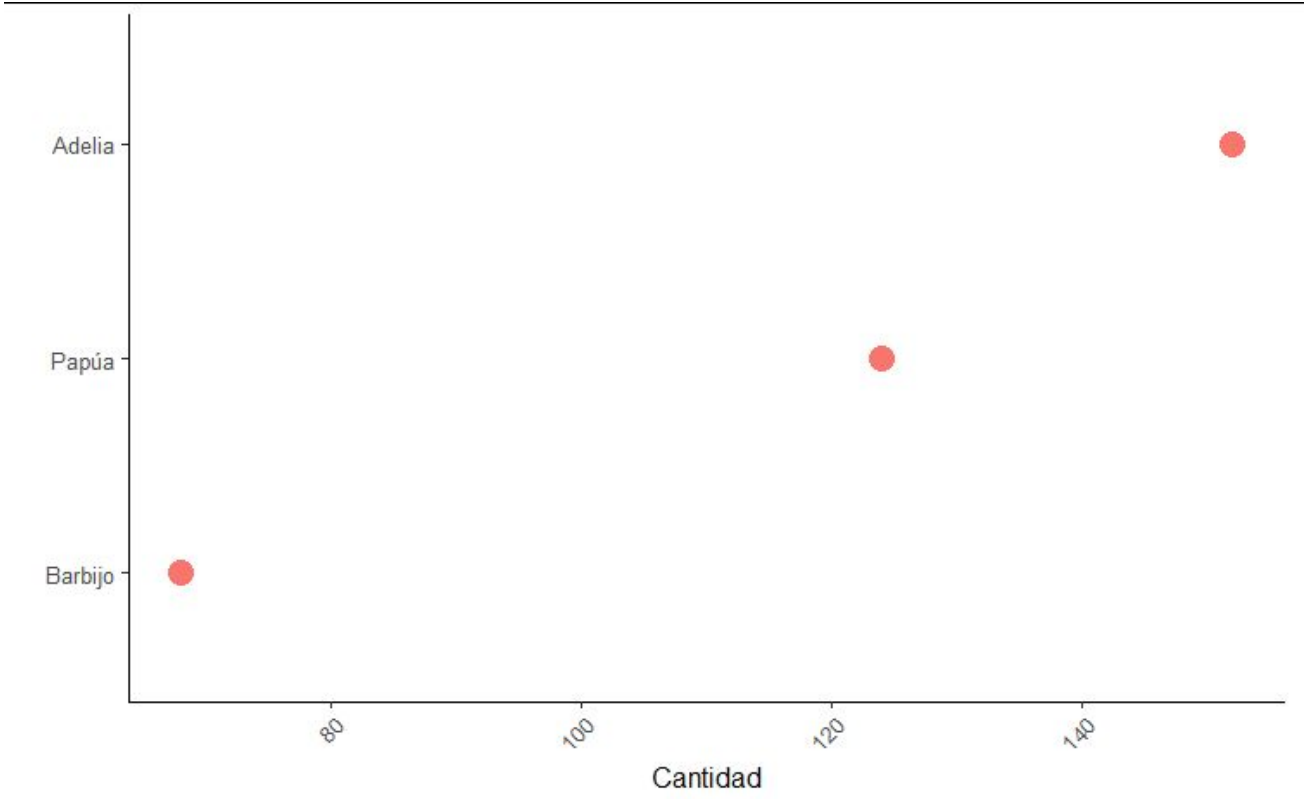
Prestar atención al orden de las barras



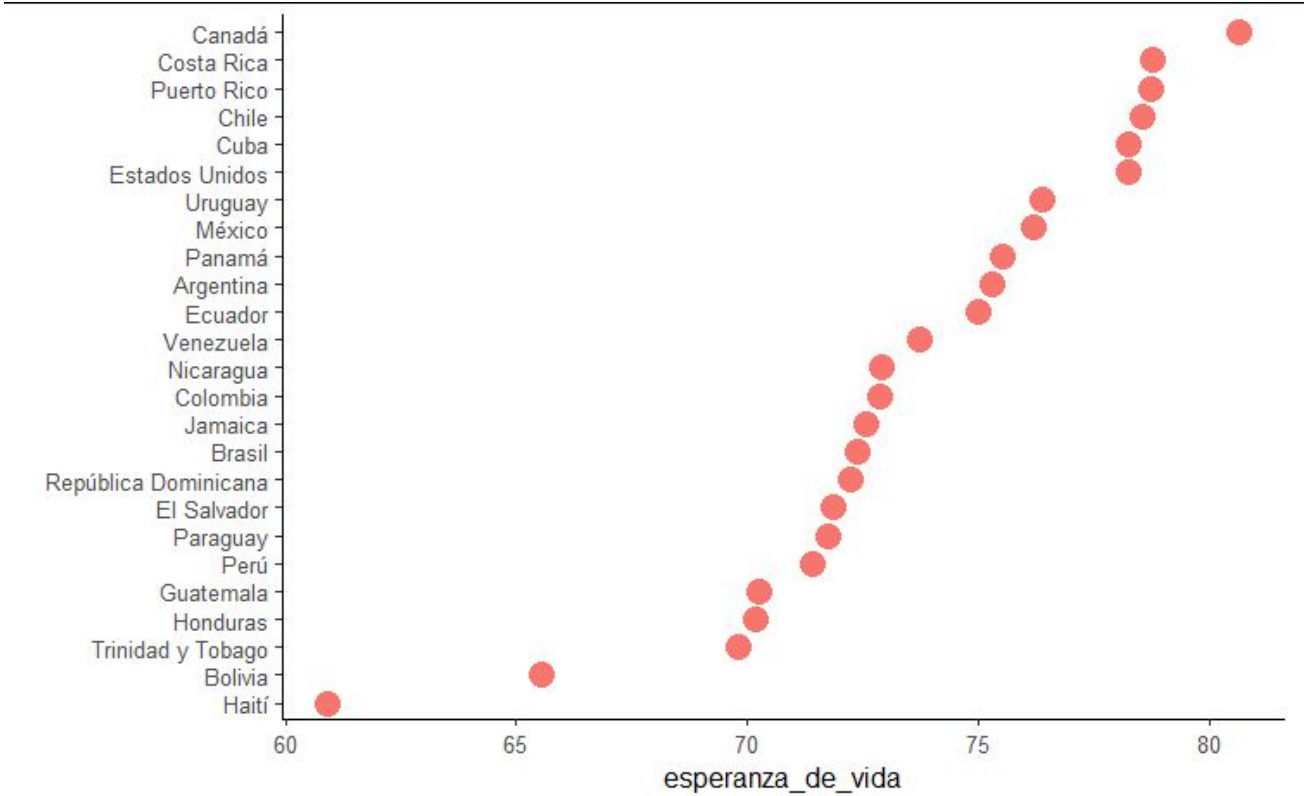
Prestar atención al orden de las barras



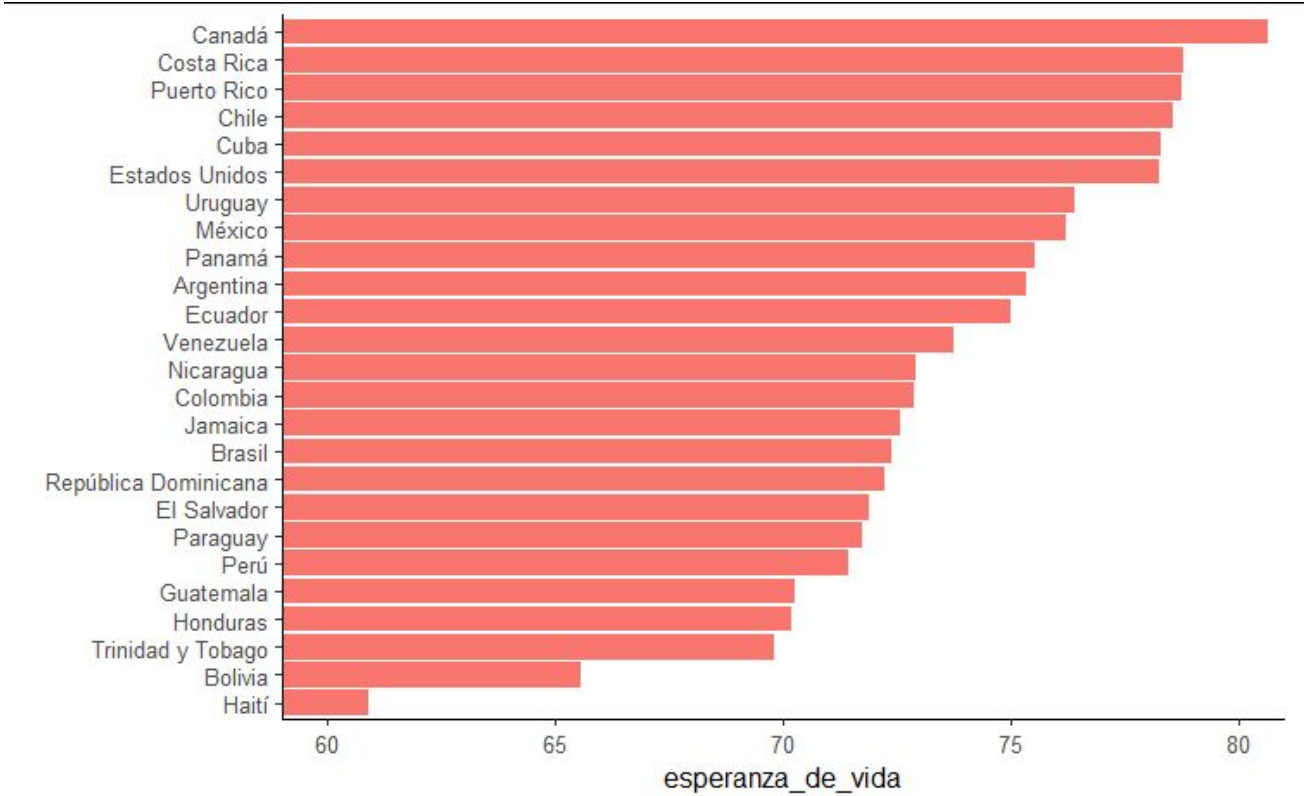
Podemos usar puntos



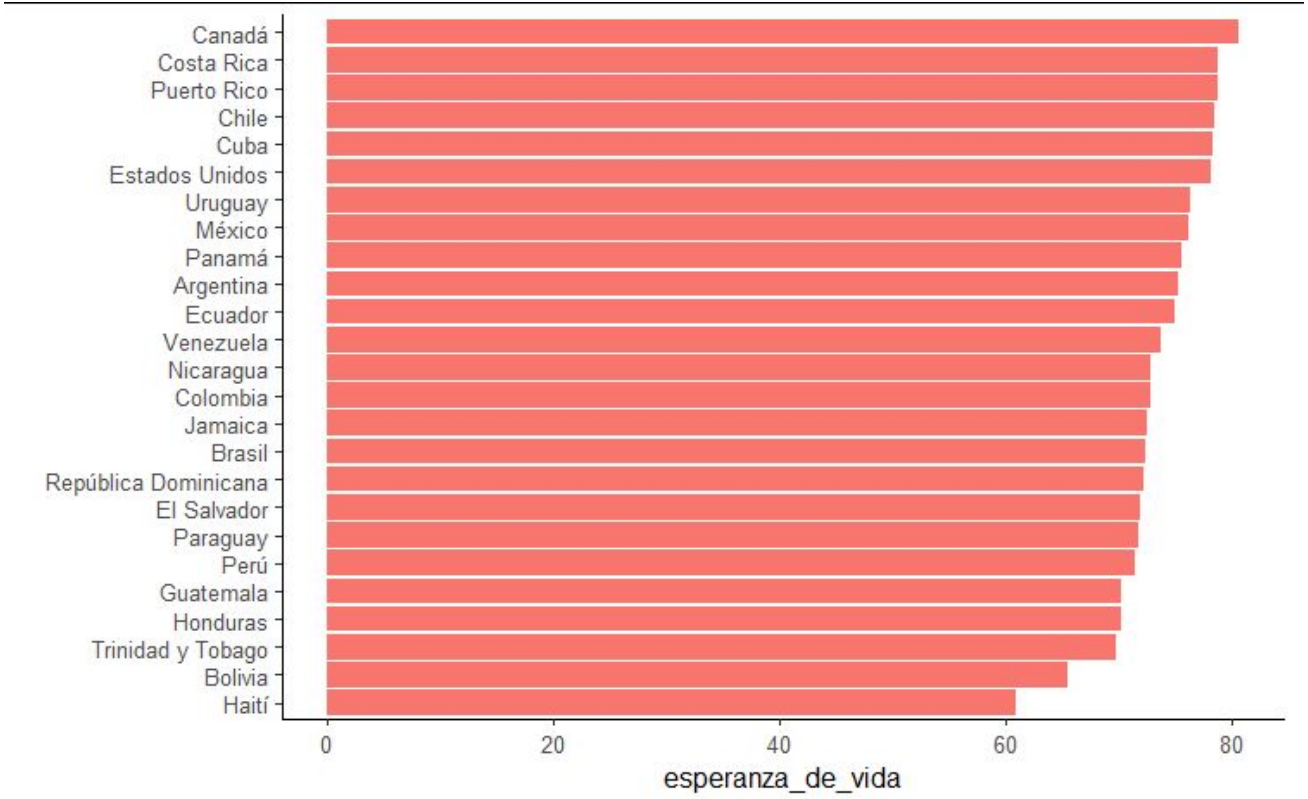
Los puntos son preferibles si queremos truncar los ejes



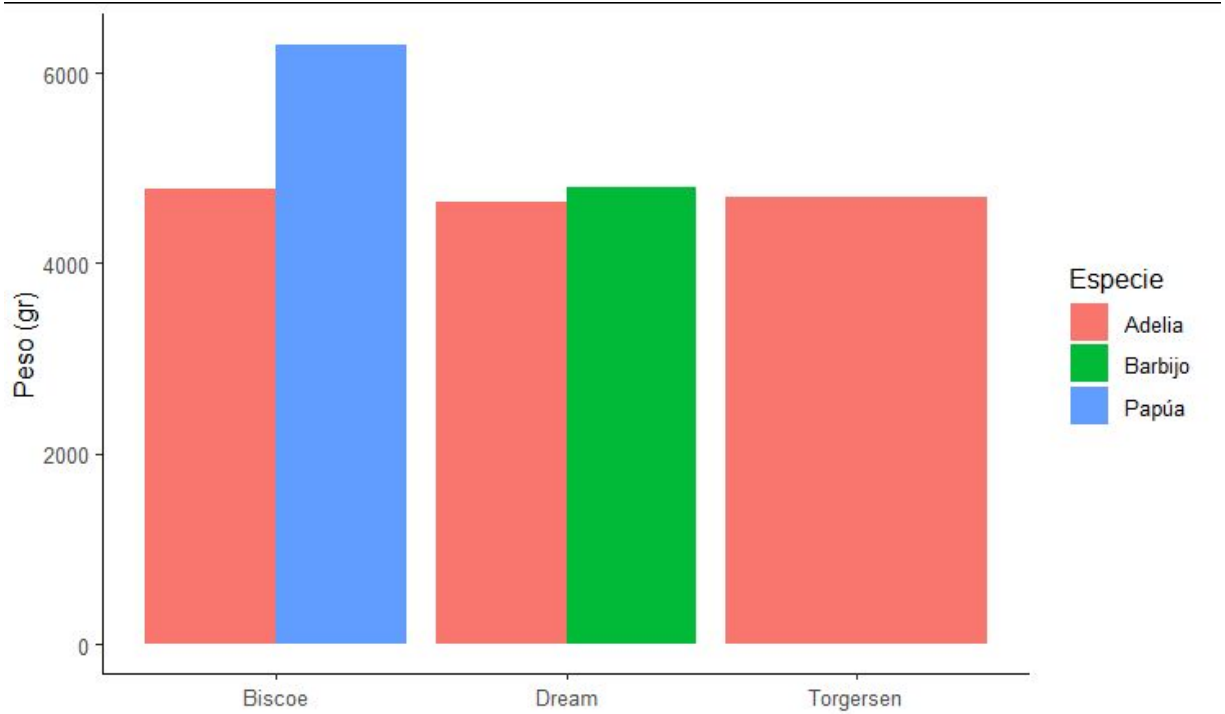
Los puntos son preferibles si queremos truncar los ejes



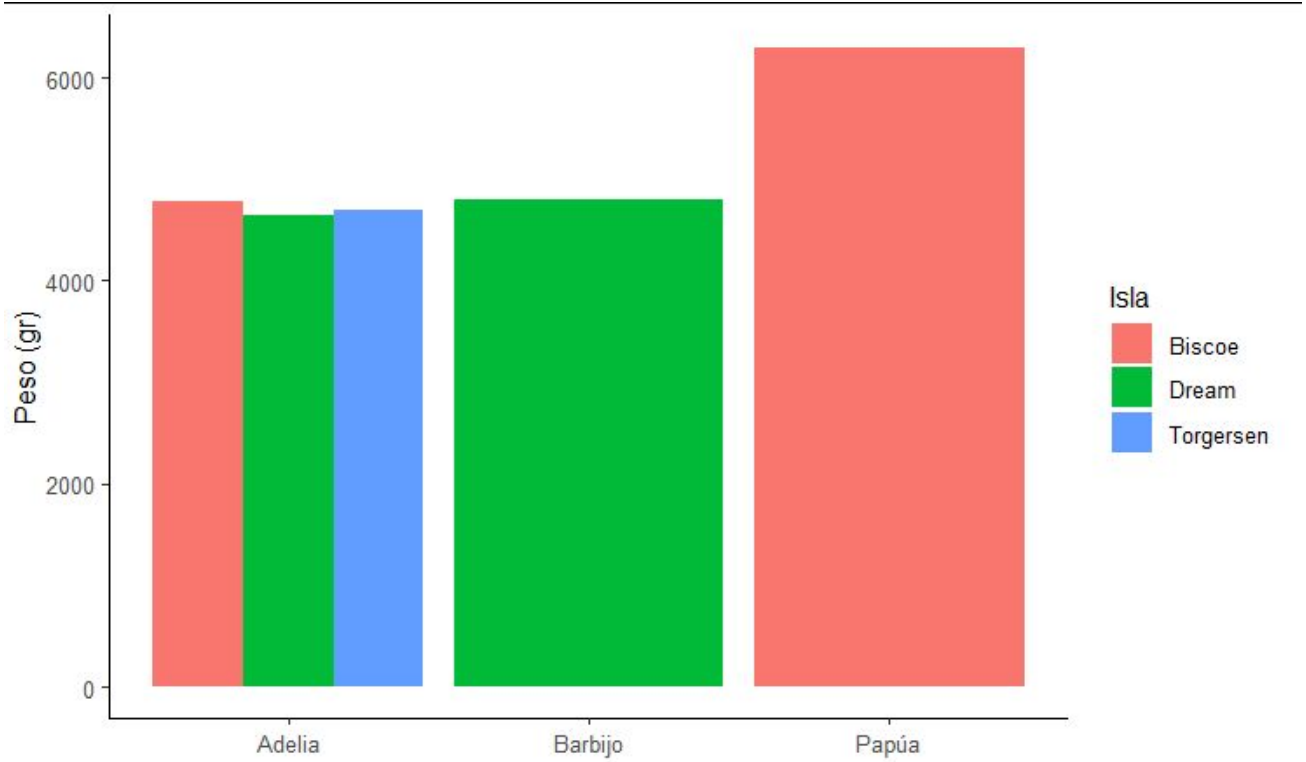
Los puntos son preferibles si queremos truncar los ejes



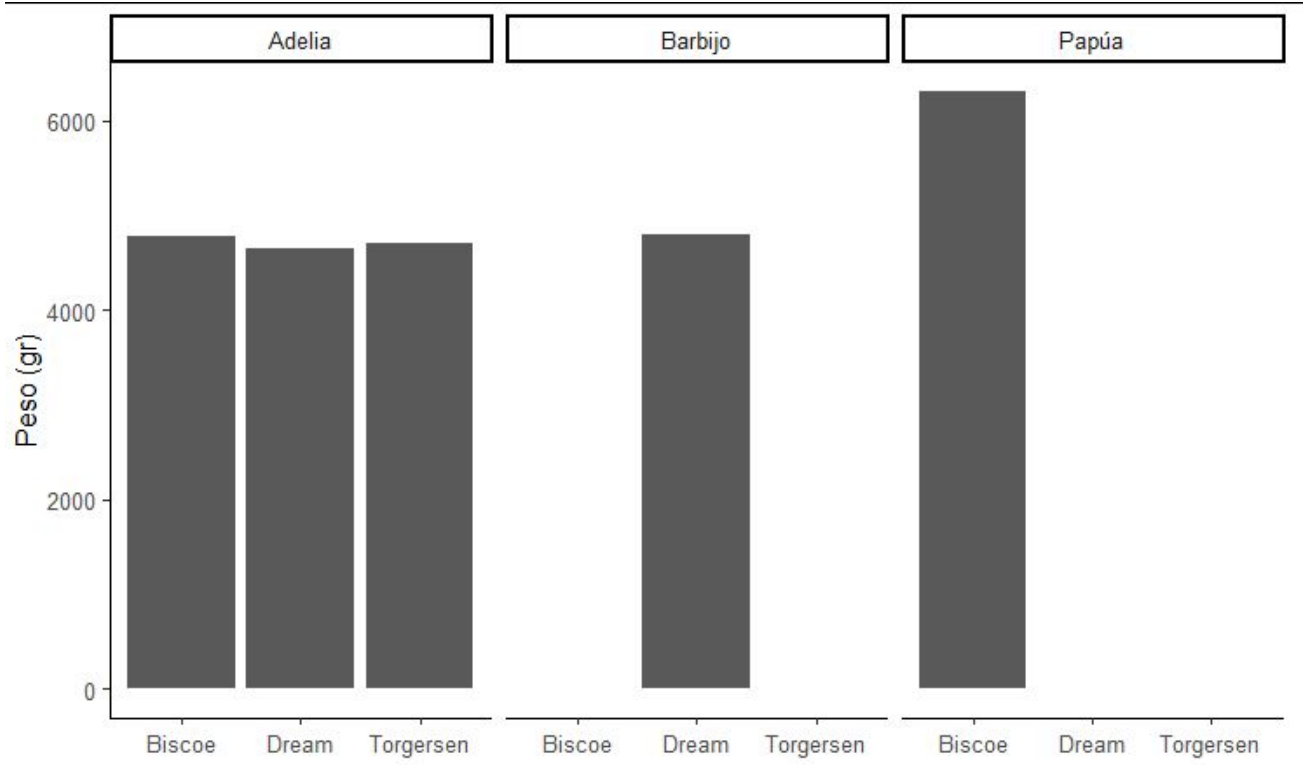
Utilizamos barras agrupadas para datos con varias dimensiones



Somos libres de elegir por qué variable agrupar



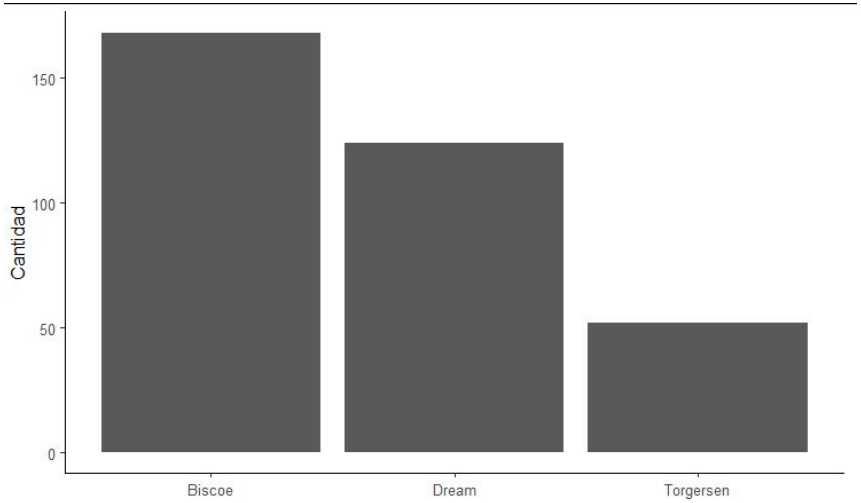
También podemos usar multiple paneles (facetas)

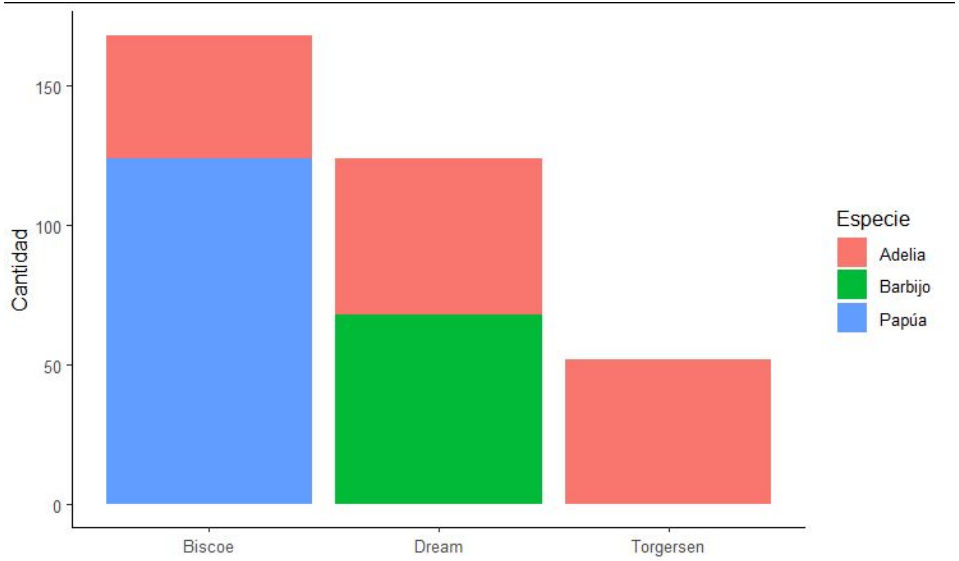


Conjuntos de datos SIN cantidades calculadas

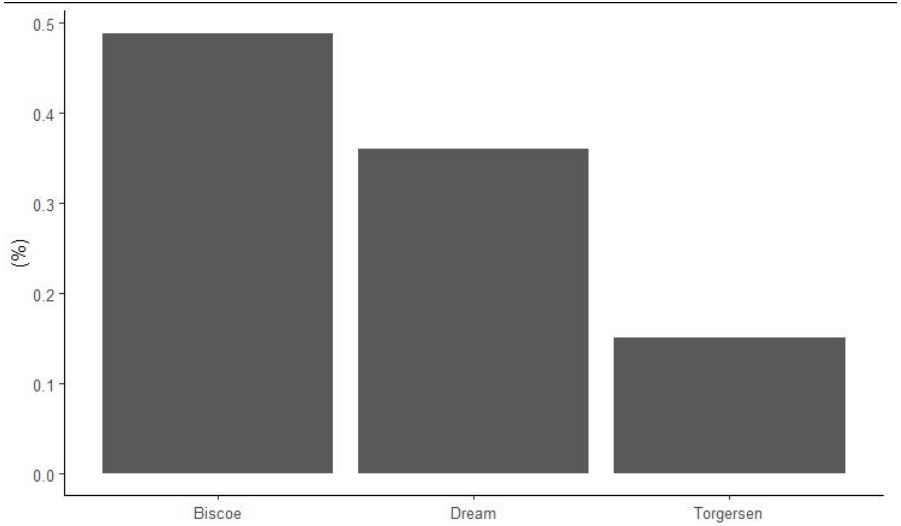
```
# A tibble: 344 x 8
   especie isla     largo_pico_mm alto_pico_mm largo_aleta_mm masa_corporal_g sexo   anio
   <fct>   <fct>            <dbl>        <dbl>          <int>          <int> <fct> <int>
 1 Adelia  Torgers~          39.1         18.7            181           3750 macho  2007
 2 Adelia  Torgers~          39.5         17.4            186           3800 hemb~  2007
 3 Adelia  Torgers~          40.3         18              195           3250 hemb~  2007
 4 Adelia  Torgers~          NA           NA              NA              NA NA       2007
 5 Adelia  Torgers~          36.7         19.3            193           3450 hemb~  2007
 6 Adelia  Torgers~          39.3         20.6            190           3650 macho  2007
 7 Adelia  Torgers~          38.9         17.8            181           3625 hemb~  2007
 8 Adelia  Torgers~          39.2         19.6            195           4675 macho  2007
 9 Adelia  Torgers~          34.1         18.1            193           3475 NA     2007
10 Adelia  Torgers~          42           20.2            190           4250 NA     2007
# ... with 334 more rows
```

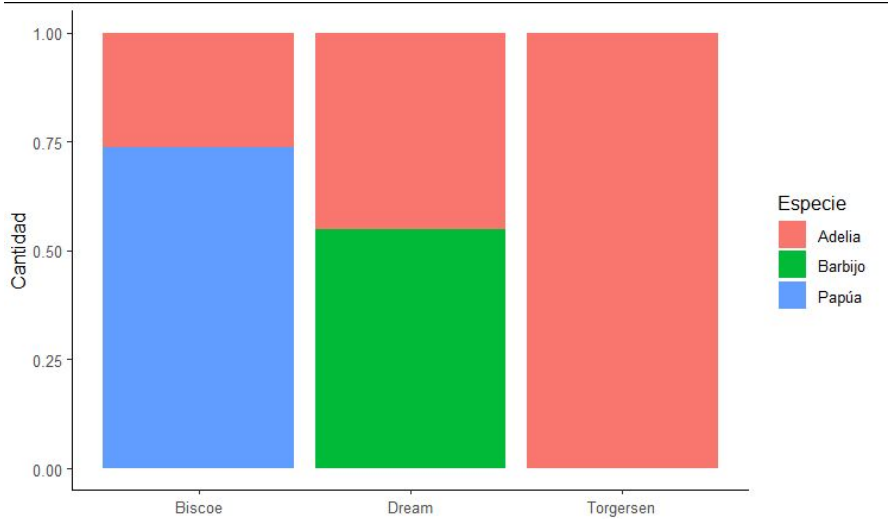
Podemos graficar cantidades…





…y también proporciones





¡Descanso de 10 minutos!!

No se desconecten pero retirense de las pantallas

Trabajando con categorías

Variables categóricas

```
  [1] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [11] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [21] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [31] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [41] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [51] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [61] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [71] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [81] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
 [91] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
[101] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
[111] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
[121] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
[131] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
[141] Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia  Adelia
[151] Adelia  Adelia  Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[161] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[171] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[181] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[191] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[201] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[211] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[221] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[231] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[241] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[251] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[261] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Papúa
[271] Papúa   Papúa   Papúa   Papúa   Papúa   Papúa   Barbijo Barbijo Barbijo Barbijo
```

→ Columna (variable) especie del dataset pingüinos

~ vector de caracteres

… pero guarda información extra

- son categorías
- ¿tienen un orden intrínseco?

Factores

```
> str(pinguinos)
tibble [344 × 8] (S3: tbl_df/tbl/data.frame)
 $ especie        : Factor w/ 3 levels "Adelia","Barbijo",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ isla           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ largo_pico_mm  : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
 $ alto_pico_mm   : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
 $ largo_aleta_mm : int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
 $ masa_corporal_g: int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
 $ sexo           : Factor w/ 2 levels "hembra","macho": 2 1 1 NA 1 2 1 2 NA NA ...
 $ anio           : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

→ especie contiene un tipo de dato “factor”

Factores

Los factores tienen “niveles” que definen el orden de las categorías*

```
> levels(pinguinos$especie)
[1] "Adelia"  "Barbijo" "Papúa"
```

*en este caso coincide con el orden alfabético, pero podría querer un orden distinto.

Factores creacion

Creo un array de strings

```
mes <- c("Dic", "Abr", "Ene", "Mar")
```

Agrego un mes mal escrito

```
mes[5] <- "Jus"
```

Creo un array de levels

```
meses_del_anio <- c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep",
"Oct","Nov","Dic")
```

Ahora creo un array de factores con los leves permitidos

```
mes_factor <- factor(c("Dic", "Abr", "Ene", "Mar"), levels = meses_del_anio)
```

Si intento agregar un valor que no esta en los levels da un error

```
mes_factor[5] <- "Eme"
```

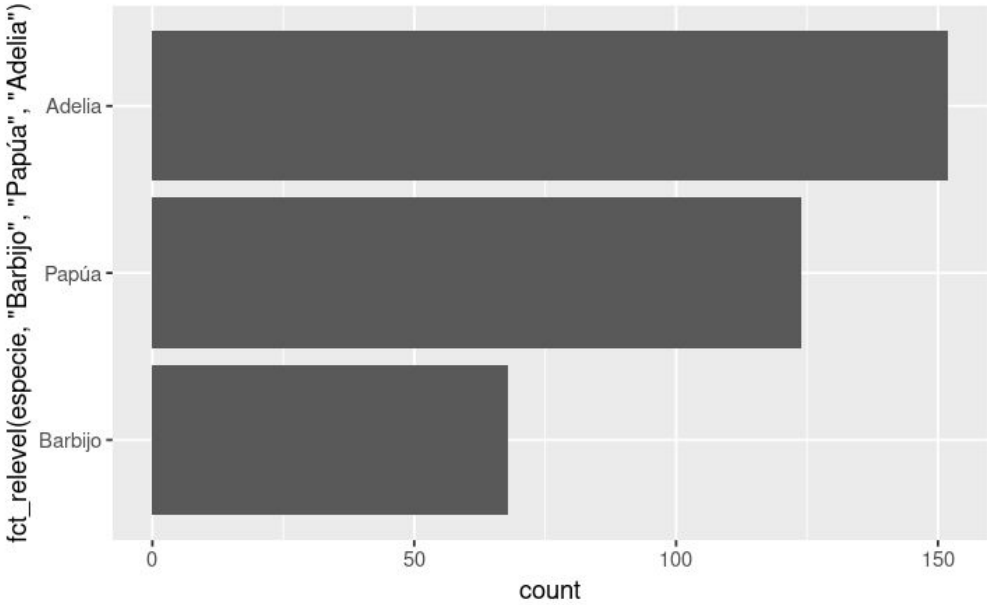
```
Warning in `[<-.factor`(`*tmp*`, 5, value = "Eme") :
  invalid factor level, NA generated
[1] Dic  Abr  Ene  Mar  <NA>
Levels: Ene Feb Mar Abr May Jun Jul Ago Sep Oct Nov Dic
```

forcats

fct_relevel()

Permite ordenar manualmente las categorías

```
ggplot(pinguinos, aes(y = fct_relevel(especie, "Barbijo", "Papúa", "Adelia"))) +
  geom_bar()
```

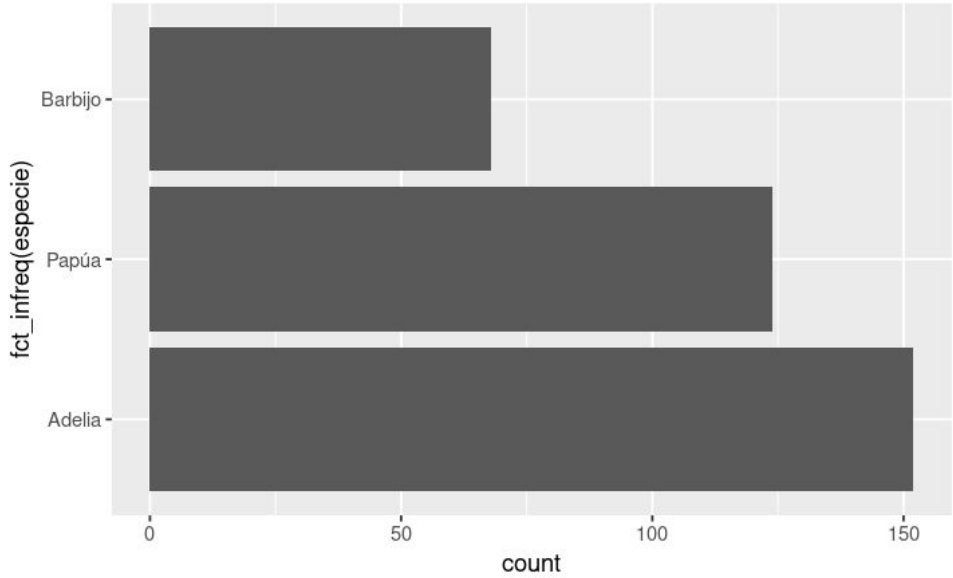


*este paso puede hacerse antes de generar el gráfico, manipulando los datos con dplyr

fct_infreq()

Permite ordenar las categorías de acuerdo a la frecuencia (que calcula internamente)

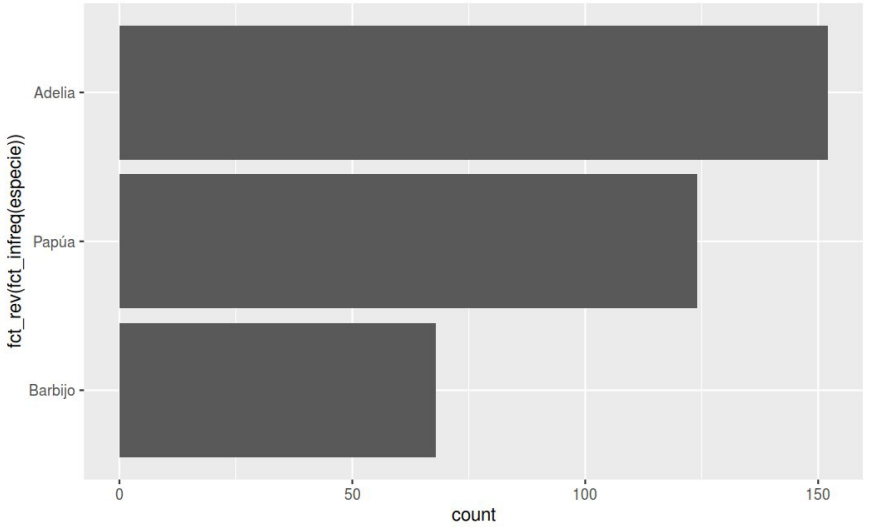
```
ggplot(pinguinos, aes(y = fct_infreq(especie))) +
  geom_bar()
```



fct_infreq() y fct_rev()

Combinando estas funciones, invertimos el orden

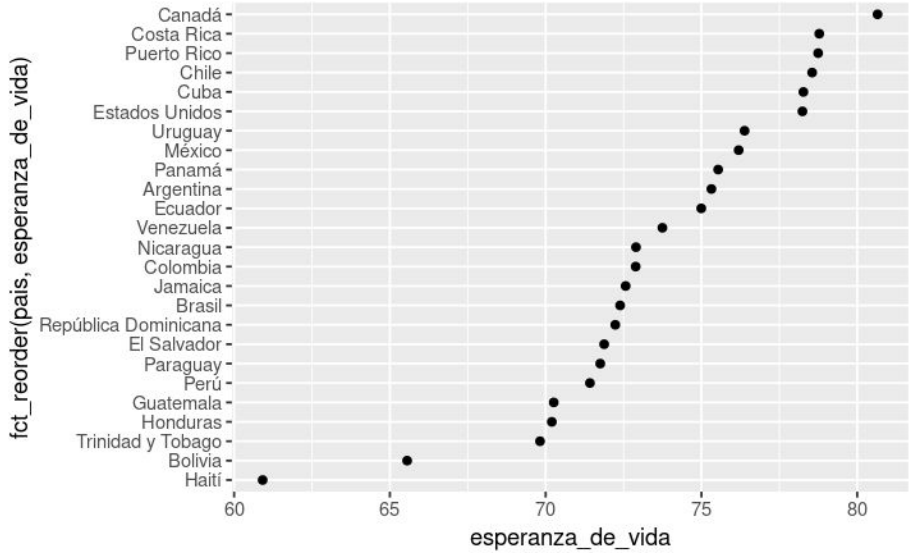
```
ggplot(pinguinos, aes(y= fct_rev(fct_infreq(especie)))) +
  geom_bar()
```



fct_reorder()

Permite ordenar las categorías de acuerdo a otra variable del dataset

```
paises %>%
  filter(anio == 2007, continente == "Américas") %>%
  ggplot(aes(esperanza_de_vida, fct_reorder(pais, esperanza_de_vida))) +
  geom_point()
```

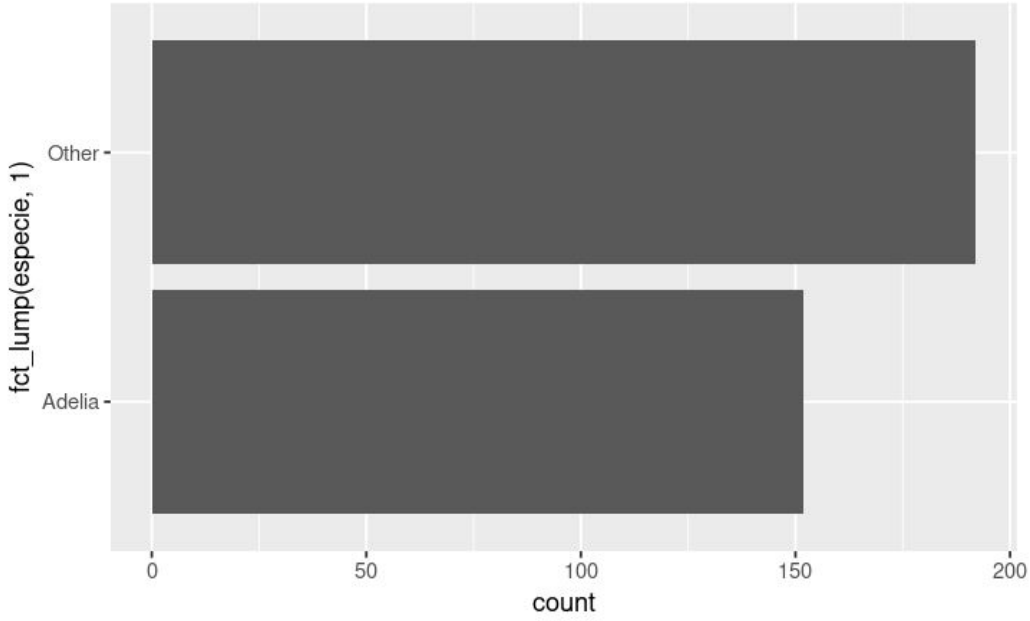


*permite definir que función se aplica para ordenar y el orden.
*para ordenar por 2 variables: `fct_reorder2()`

fct_lump()

Agrupa las categorías menos frecuentes en “Otros”

```
ggplot(pinguinos, aes(y = fct_lump(especie, 1))) +
  geom_bar()
```



*hay distintos criterios para definir “las categorías menos frecuentes”