Manejo de datos en R (II)

Introducción a la Línea de Comandos para Análisis Bioinformáticos

03 de Marzo, 2020

Dia martes

Slide con muchos tipos de graficos

Idea de lo que lleva hacer un grafico para hacerlo bien

Que hay detras de hacer estos graficos en R? Muchas librerias, de las cuales un monton son del universo tidyverse.

Los que no son de universo tidyverse igual tienen que ser retocadas generalmente, so... its good to learn tidyverse.

Cuales son las operaciones que generalmente se hacen con los datos?

- Cargar datos *crudos*/Guardar datos finales y tablas de interés.
- Filtrar datos (con criterio).
- Unir datos que vienen de diferentes fuentes, referentes a un mismo conjunto estudiado.
- Hacer modificaciones: crear *tags*, correcciones ortográficas, filas y columnas de tablas, etc. . .
- Generar nuevos datos: obtener promedios, medianas, aplicar funciones de librerías.
- Dejar anotado y reportar lo hecho.

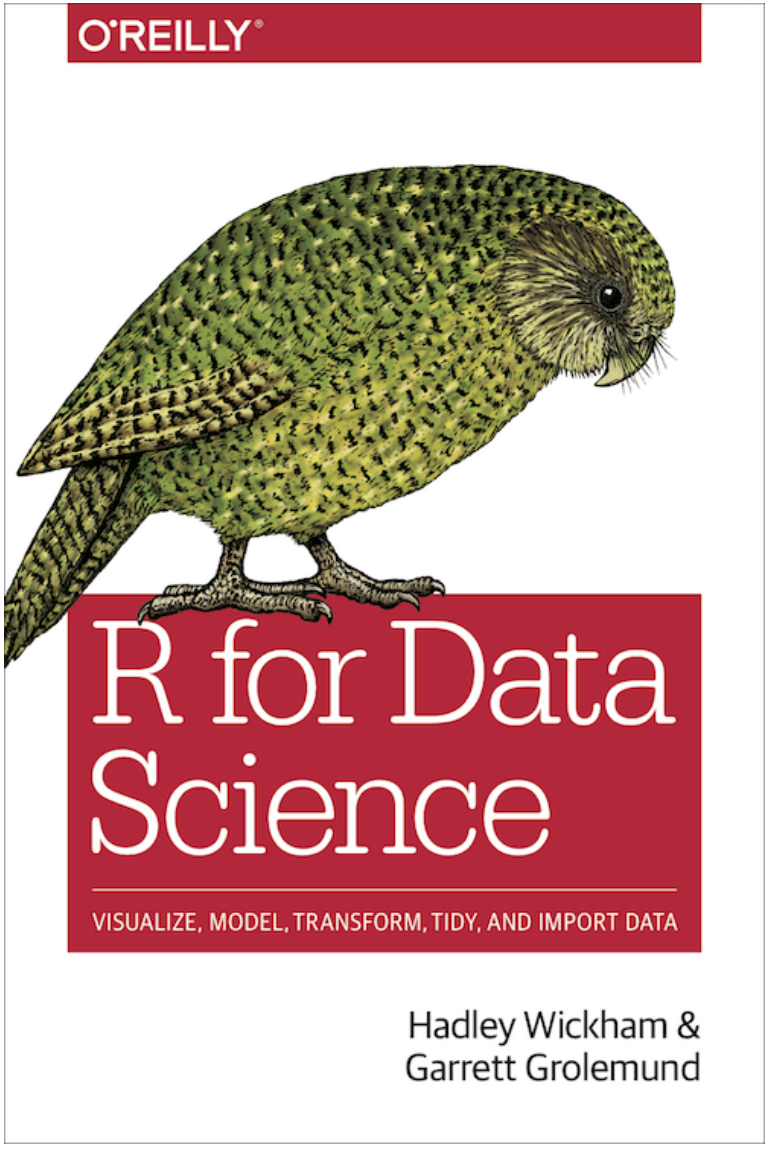
Una pequeña ilustración: es posible que querramos trabajar con un GFF

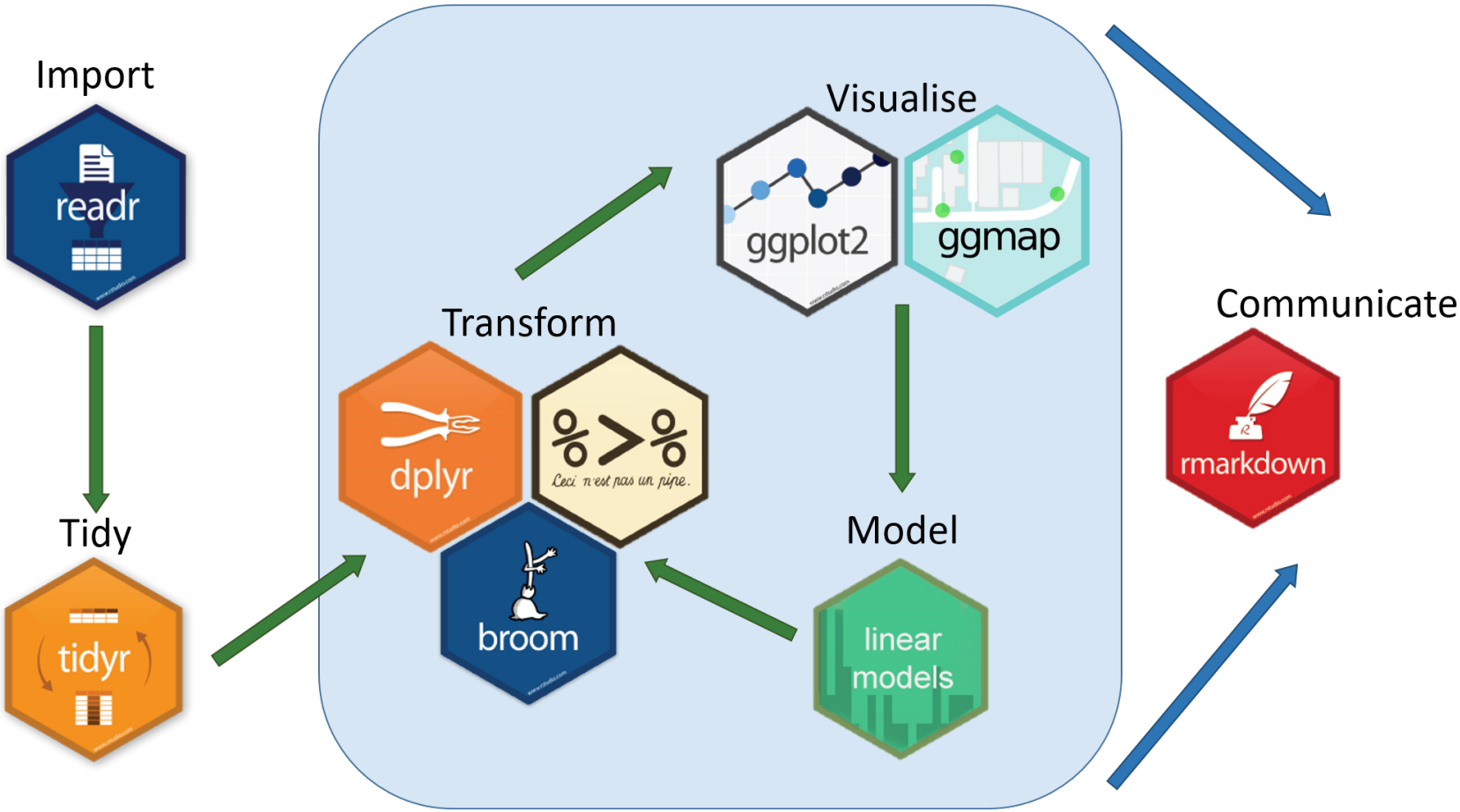
- Cargo un GFF
- Filtro para quedarme con transcriptos codificantes
- De ahí capaz quiero hacer una tabla con datos como largo de transcripto y contenido GC
- Entonces tengo que generarme una tabla que tenga las secuencias, y luego unir ambas
- Subsetear por transcripto , obtener el largo, y calcular GC
- Capaz me armo un tag que diga si son 'low GC', 'medium GC' o 'high GC'.
- Después puedo llegar a querer plotear.

Y así con todos los datos que nos vamos cruzando... nos vamos a ir encontrando cosas que vamos a tener que hacer.

Ahi recien paso a hablar del paquete

Bibliografía




tidyverse
Import
readr
Tidy
tidyr
Transform
dplyr
%>%
Leci n'est pas un pipe.
broom
Visualise
ggplot2
ggmap
Model
linear models
Communicate
rmarkdown

- Tibbles are data frames, but they tweak some older behaviours to make life a little easier.
- Almost all of the functions that you'll use in this book produce tibbles, as tibbles are one of the unifying features of the tidyverse
- It's possible for a tibble to have column names that are not valid R variable names: backtick!
- Another way to create a tibble is with tribble(), short for transposed tibble. tribble() is customised for data entry in code: column headings are defined by formulas (i.e. they start with ~), and entries are separated by commas.


```
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1            5.1         3.5          1.4         0.2
## 2            4.9         3.0          1.4         0.2
## 3            4.7         3.2          1.3         0.2
## 4            4.6         3.1          1.5         0.2
## 5            5.0         3.6          1.4         0.2
## 6            5.4         3.9          1.7         0.4
## 7            4.6         3.4          1.4         0.3
## 8            5.0         3.4          1.5         0.2
## 9            4.4         2.9          1.4         0.2
## 10           4.9         3.1          1.5         0.1
## 11           5.4         3.7          1.5         0.2
## 12           4.8         3.4          1.6         0.2
```

```
library(tibble)

as_tibble(iris)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Spe
##           <dbl>       <dbl>        <dbl>       <dbl> <fc
##  1          5.1         3.5          1.4         0.2 set
##  2          4.9         3            1.4         0.2 set
##  3          4.7         3.2          1.3         0.2 set
##  4          4.6         3.1          1.5         0.2 set
##  5          5           3.6          1.4         0.2 set
##  6          5.4         3.9          1.7         0.4 set
##  7          4.6         3.4          1.4         0.3 set
##  8          5           3.4          1.5         0.2 set
```

- read_csv() reads comma delimited files, read_tsv() reads tab delimited files, and read_delim() reads in files with any delimiter.
- read_fwf() reads fixed width files. You can specify fields either by their widths with fwf_widths() or their position with fwf_positions(). read_table() reads a common variation of fixed width files where columns are separated by white space.
- **These functions all have similar syntax: once you've mastered one, you can use the others with ease**
- Sometimes there are a few lines of metadata at the top of the file. You can use skip = n to skip the first n lines; or use comment = "#" to drop all lines that start with (e.g.) #.
- The data might not have column names. You can use

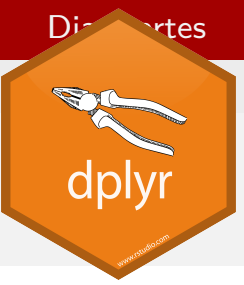
magrittr

magrittr

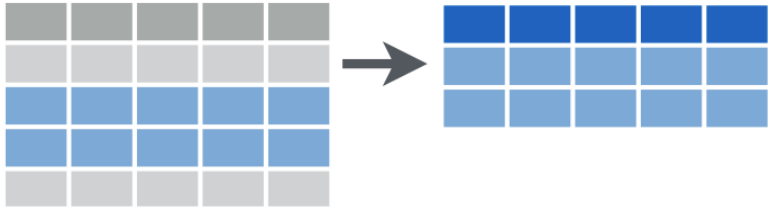
- Es el *pipe* de R.
- El uso es exactamente igual al '|' de Bash.
- Un único detalle: se utiliza `.` para hacer referencia a resultados intermedios en un pipe.

```
# hacer operaciones con una columna sin magrittr.

# Forma 1
Sepal.Width = iris$Sepal.Width
Sepal.Width.Median = median(Sepal.Width)

# Forma 2
Sepal.Width.Median =  median(iris$Sepal.Width)
```

```
# con magrittr
library(magrittr)
```



Subset Observations (Rows)



Subset Variables (Columns)

dplyr

Combine Data Sets

a

x1	x2
A	1
B	2
C	3

+

b

x1	x3
A	T
B	F
D	T

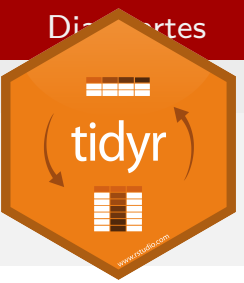
=

x1	x2	x3
A	1	T
B	2	F
C	3	NA

x1	x3	x2
A	T	1
B	F	2
D	T	NA

x1	x2	x3
A	1	T
B	2	F

x1	x2	x3
A	1	T
B	2	F



country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

tidyr

- You can represent the same underlying data in multiple ways.
  $\rightarrow$ ta bueno para mostrar diferentes tipos de datos **y decir por que esto es importante**.

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Spe
##           <dbl>       <dbl>        <dbl>       <dbl> <fc
##  1          5.1         3.5          1.4         0.2 set
##  2          4.9         3            1.4         0.2 set
##  3          4.7         3.2          1.3         0.2 set
##  4          4.6         3.1          1.5         0.2 set
##  5          5           3.6          1.4         0.2 set
##  6          5.4         3.9          1.7         0.4 set
##  7          4.6         3.4          1.4         0.3 set
##  8          5           3.4          1.5         0.2 set
##  9          4.4         2.9          1.4         0.2 set
## 10          4.9         3.1          1.5         0.1 set
```

tidyr

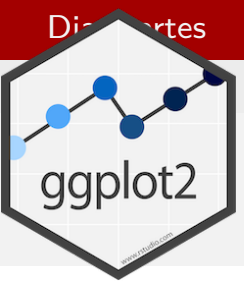
```
## # A tibble: 600 x 3
##    Species Variable     value
##    <fct>   <chr>        <dbl>
##  1 setosa  Sepal.Length   5.1
##  2 setosa  Sepal.Width    3.5
##  3 setosa  Petal.Length   1.4
##  4 setosa  Petal.Width    0.2
##  5 setosa  Sepal.Length   4.9
##  6 setosa  Sepal.Width    3
##  7 setosa  Petal.Length   1.4
##  8 setosa  Petal.Width    0.2
##  9 setosa  Sepal.Length   4.7
## 10 setosa  Sepal.Width    3.2
## # ... with 590 more rows
```

tidyr

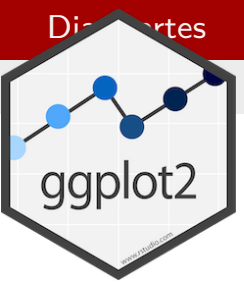
- There are three interrelated rules which make a dataset tidy:

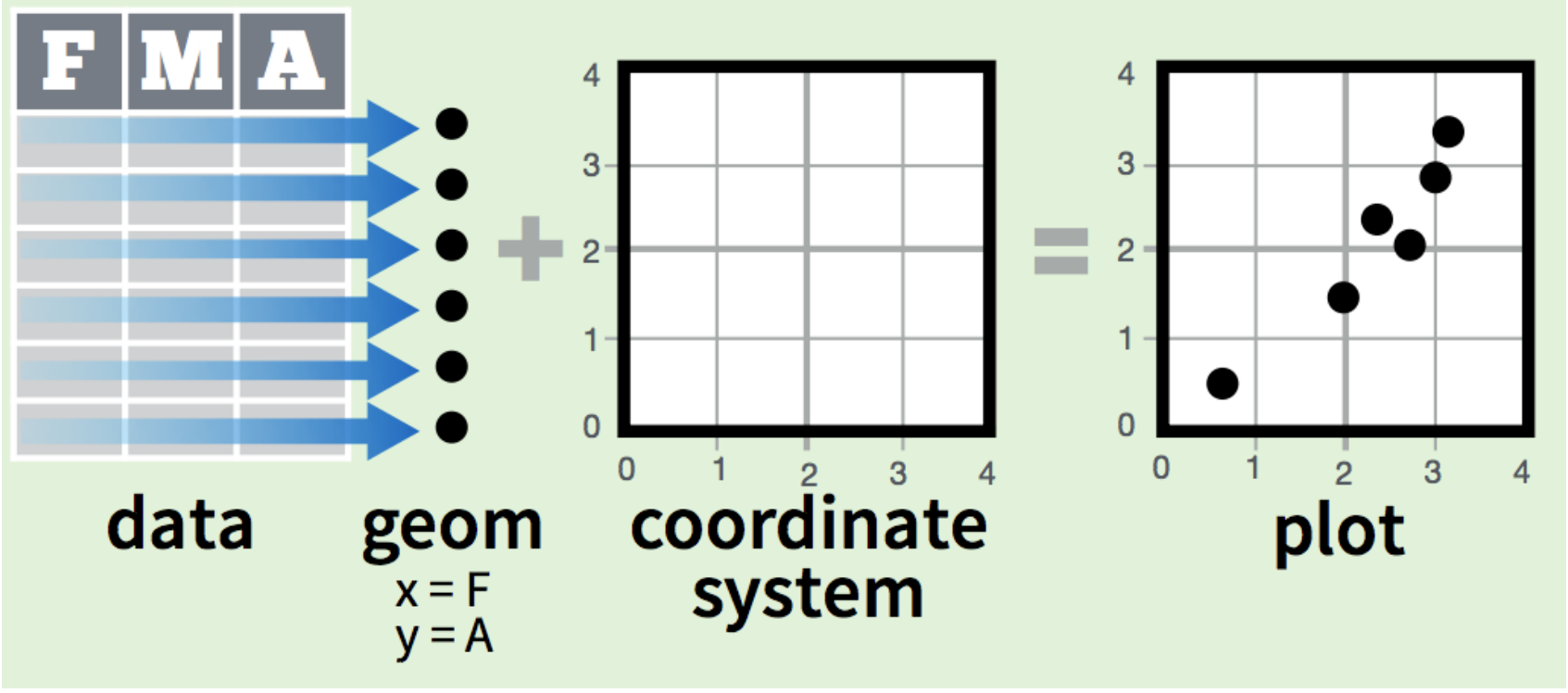
1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

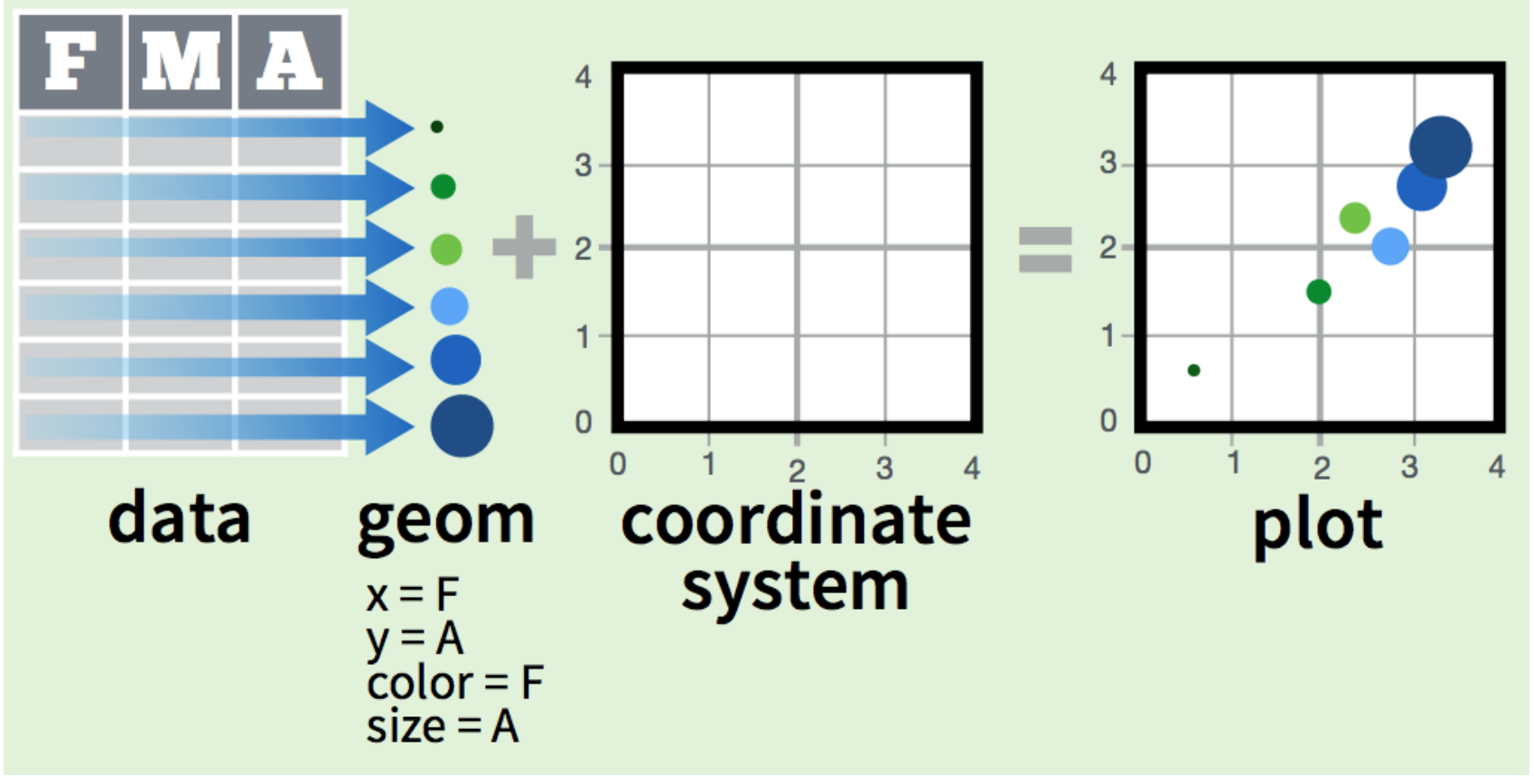
- [**most real world data is untidy**] This means for most real analyses, you'll need to do some tidying. The first step is always to figure out what the variables and observations are.
- The second step is to resolve one of two common problems:
    1. One variable might be spread across multiple columns.
    2. One observation might be scattered across multiple rows.
- Typically a dataset will only suffer from one of these problems; it'll only suffer from both if you're really unlucky! To fix these problems, **you'll need the two most important functions in tidyr**: `pivot_longer()` and `pivot_wider()`.

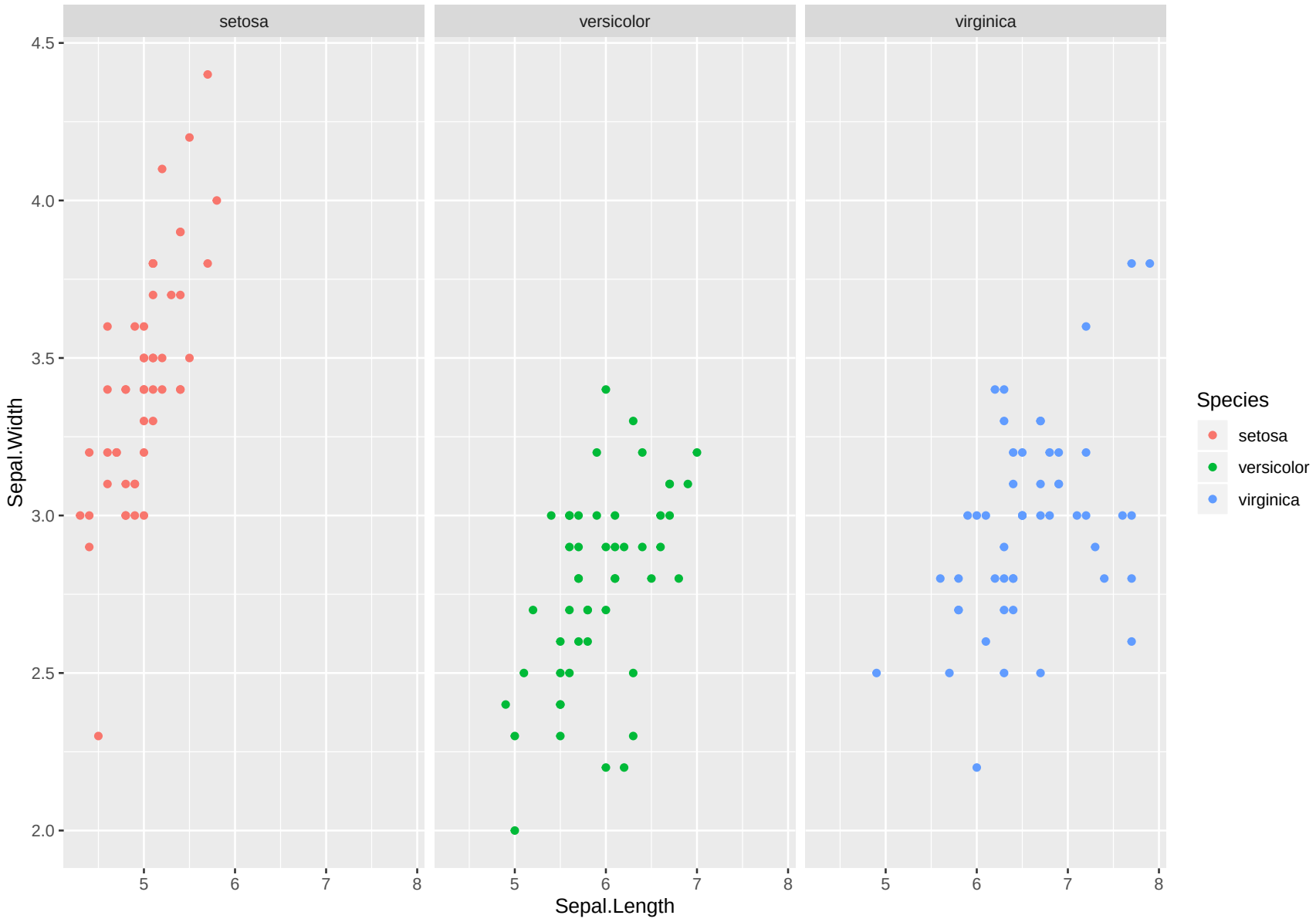


- Es importante aca explicar cosas como lo de **aes()** y cosas del estilo. Para eso hay que leer bien un articulo sobre ggplot2!


ggplot2


F
M
A
4
3
2
1
0
0
1
2
3
4
data
geom
x = F
y = A
coordinate
system
plot


F
M
A
data
geom
x = F
y = A
color = F
size = A
coordinate system
plot
0
1
2
3
4


setosa
versicolor
virginica
Sepal.Width
Sepal.Length
4.5
4.0
3.5
3.0
2.5
2.0
5
6
7
8
Species
setosa
versicolor
virginica

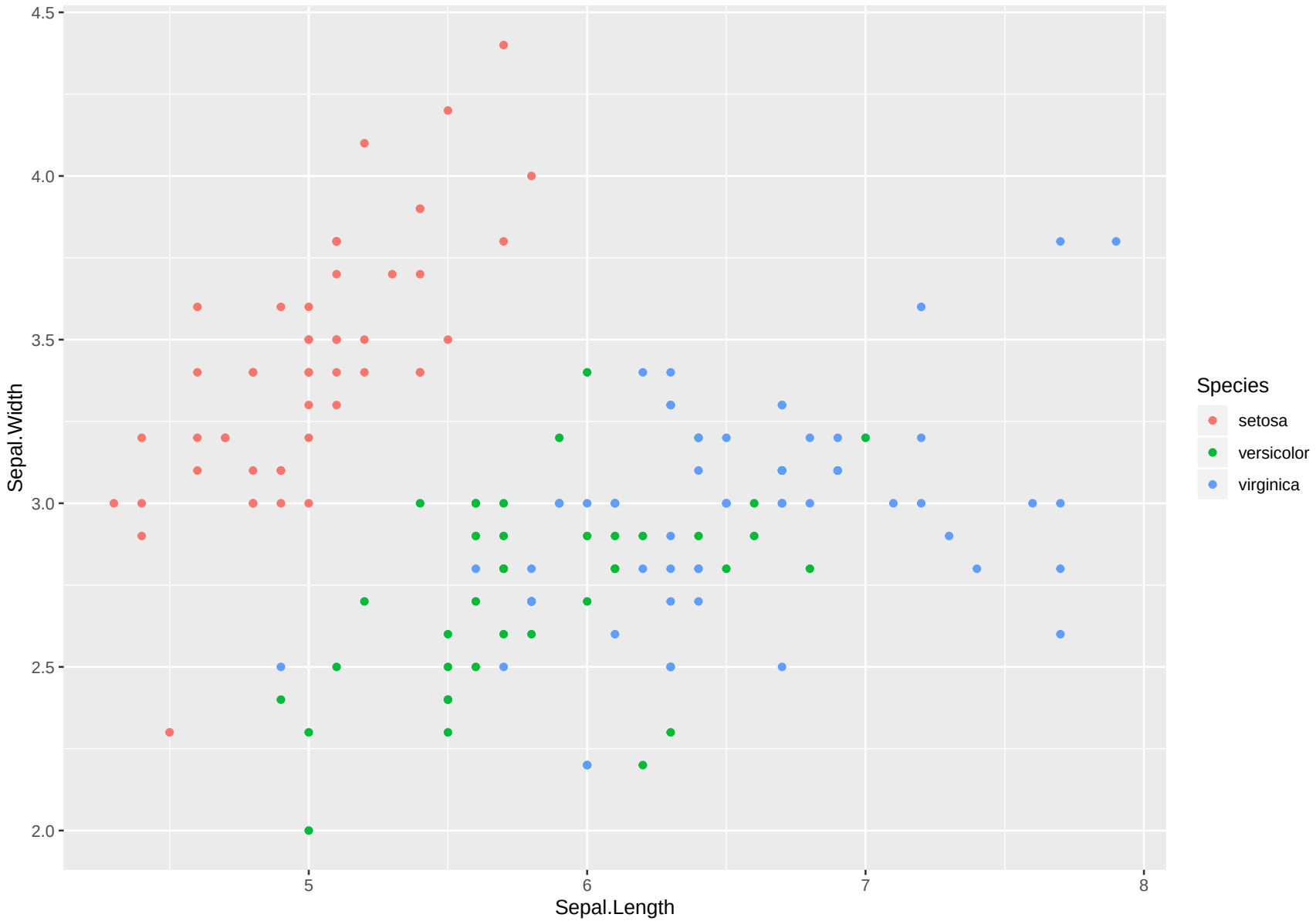
Una ventaja: reproduclbilidad

```
library(tidyverse)

# se utiliza el set de datos iris
iris %>%
  # se grafica Sepal.Length vs Sepal.Width,
  # coloreando segun Species
  ggplot(data = .,
         aes(x = Sepal.Length,
             y = Sepal.Width,
             color = Species,
             fill = Species)) +
  # se grafica utilizando puntos
  geom_point() +
  # se separa el set de datos segun Species
  facet_wrap(~Species)
```

Otra ventaja: fácil modificación

```
library(tidyverse)

# se utiliza el set de datos iris
iris %>%
  # se grafica Sepal.Length vs Sepal.Width,
  # coloreando segun Species
  ggplot(data = .,
         aes(x = Sepal.Length,
             y = Sepal.Width,
             color = Species,
             fill = Species)) +
  # se grafica utilizando puntos
  geom_point() +
#  # se separa el set de datos segun Species
#  facet_wrap(~Species)
```

Otra ventaja: fácil modificación



¿Donde encuentro info sobre estos paquetes?

- Cheatsheets
- Vignettes

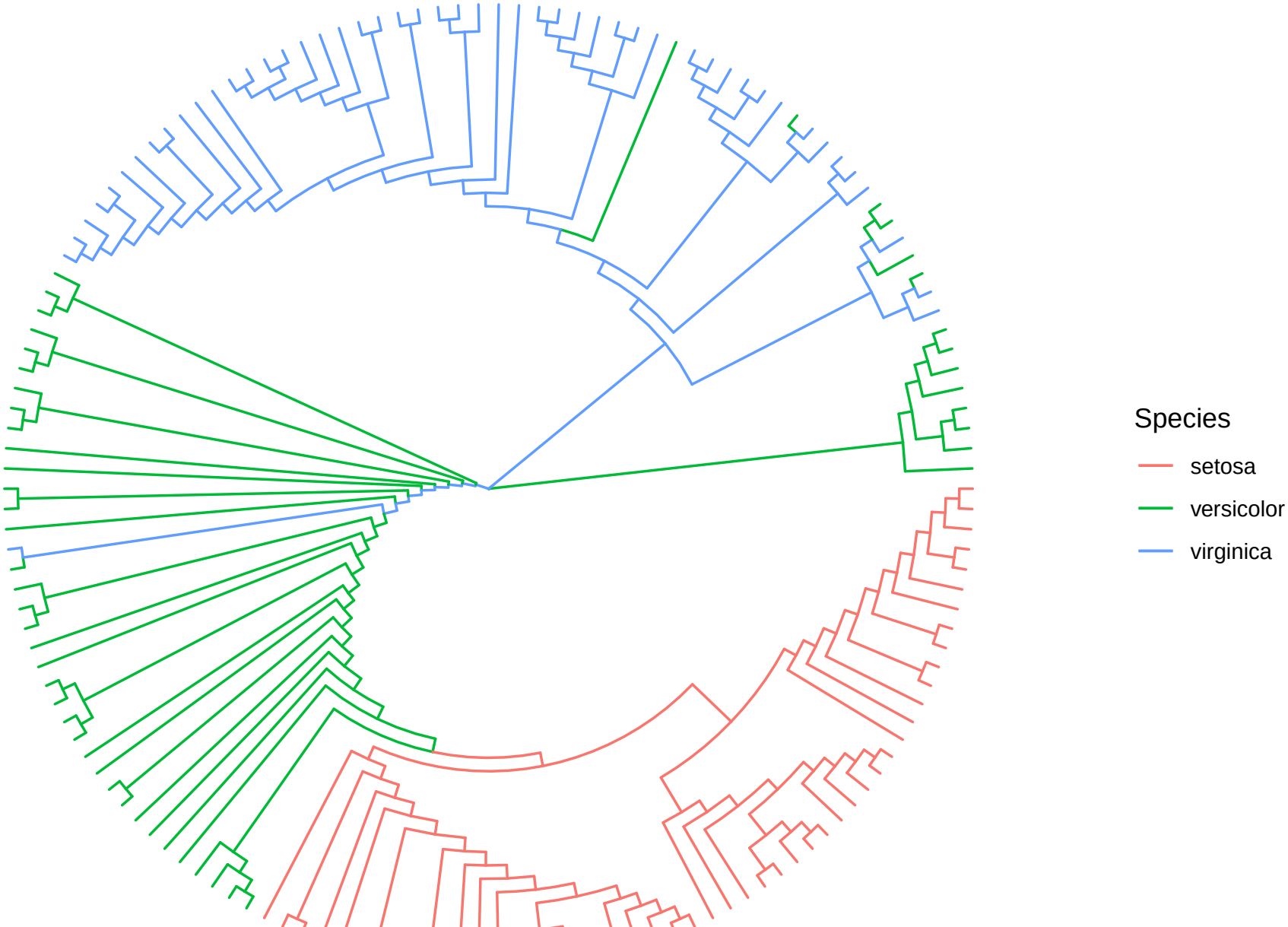
Otros ejemplos

Estadística: análisis multivariado

Clustering: librería pheatmap

GGally

Filogenética: librería ggtree



Genómica: BioCircos/rcirclize y gggnomics, ggbio

Biología estructural:

Espectrometria de masas

Ecologia