Manejo de datos en R (II)

Introducción a la Línea de Comandos para Análisis Bioinformáticos

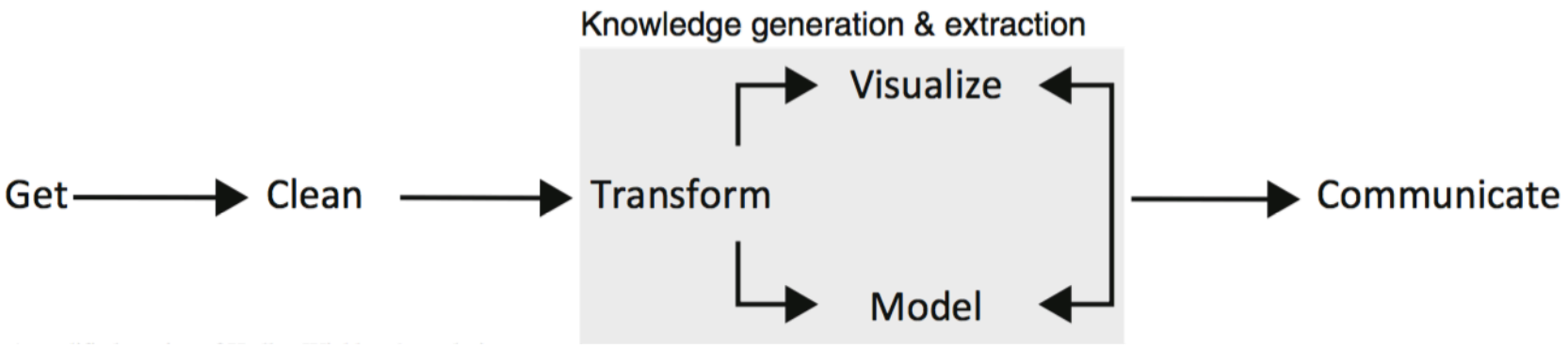
11 de Agosto, 2021

Breve repaso

Algunas ventajas de usar *R*

- Lenguaje potente pero de relativa facil entradac
- Buena integracion de:
  - importe de datos de diferente formato
  - edicion y filtrado de datos
  - visualizacion/modelado de datos (analisis exploratorio)
  - comunicacion de resultados (RMarkdown)
- Es particularmente popular en biologia computacional (especialmente eco-evo y genomicas)
- Queremos hacer graficos de alta calidad!

Manejo de datos y análisis reproducible

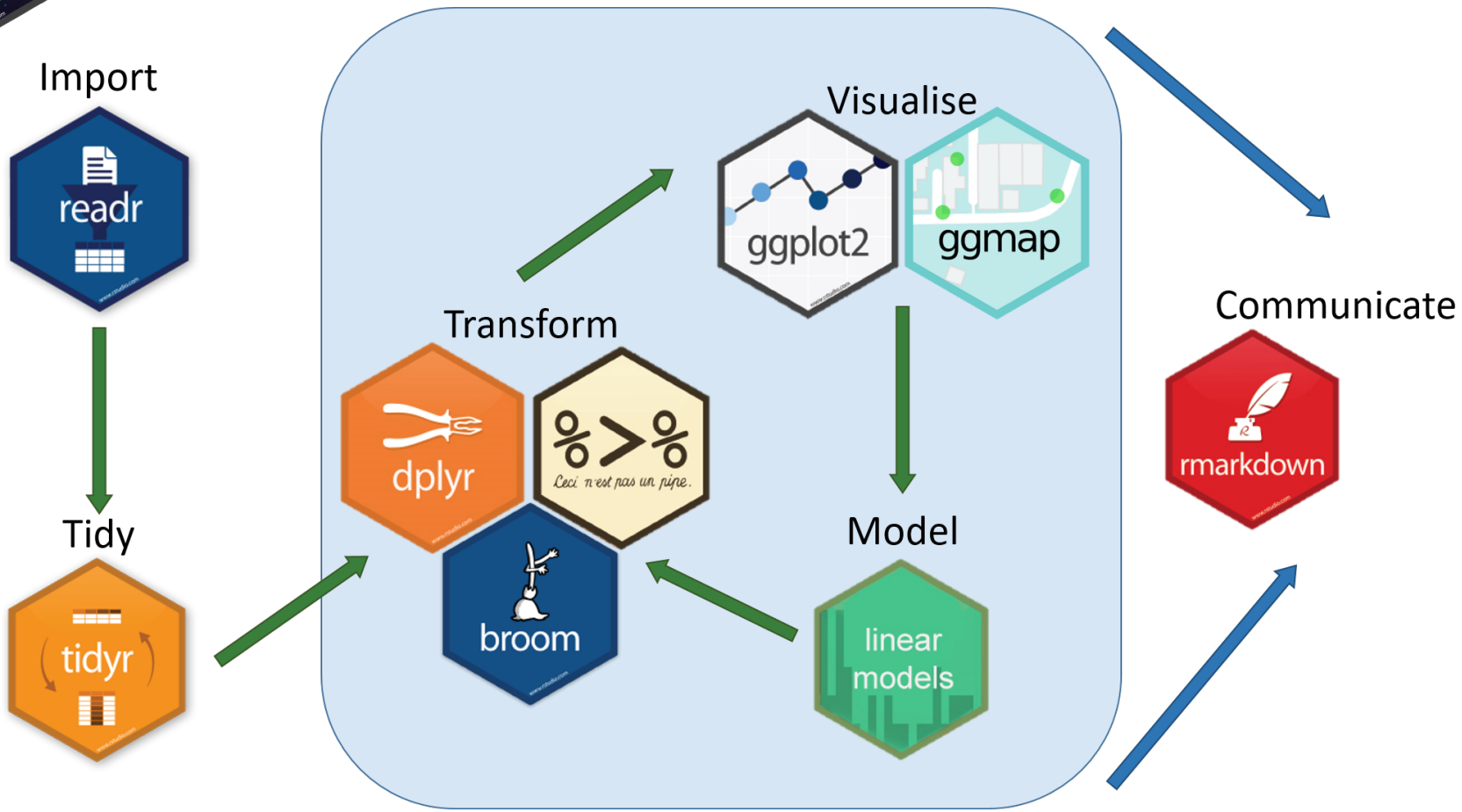


Data Wrangling with R (Boehmke, 2016)

Operaciones sobre 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.
- Visualiar/modelar datos
- Dejar anotado y reportar lo hecho.

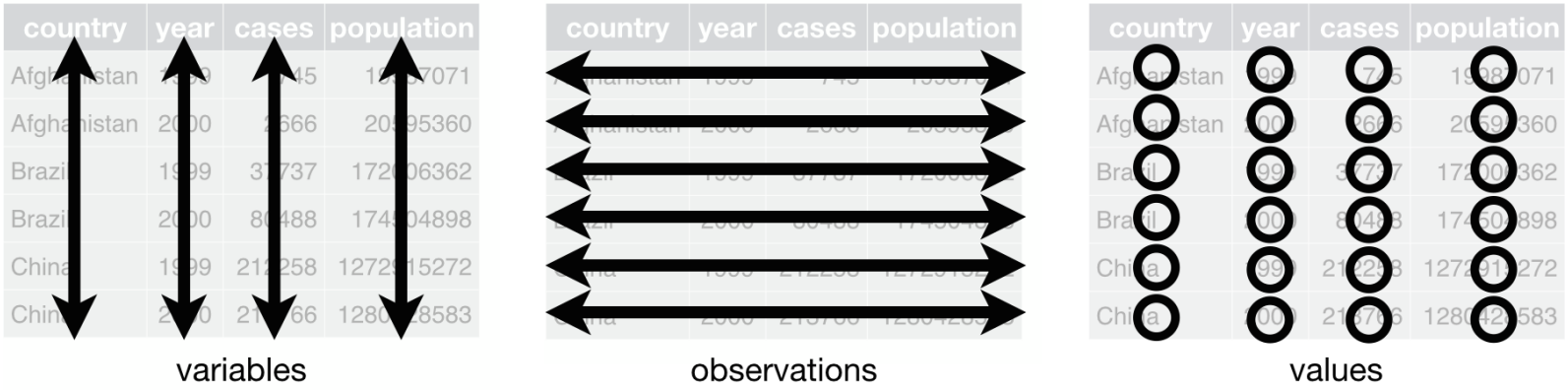
tidyverse: una forma de programar en *R*



- Hay librerias del universo *tidyverse* para cada paso clave de analisis de datos.
- Hay, ademas, librerias con su logica para analisis especificos: estadistica, modelado de datos, filogenetica, genomica, etc...

tidy data

"*Tidy*" en ingles significa "ordenado", de ahi el nombre de la libreria.



- Caracteristicas de *tidy* data:
  - Cada fila corresponde a una observacion
  - Cada columna corresponde a una variable
- **Ventajas**:
  - Estandarizacion
  - Se basa, y le saca jugo, a la logica vectoria de *R*
  - Ya tenemos todos los paquetes de *tidyverse* para trabajar arriba de la *tidy* data! (y generarla)

Operaciones sobre datos (con *tidyverse*)

- Cargar datos *crudos*/Guardar datos finales y tablas de interés. (**readr**)
- Filtrar datos (con criterio). (**dplyr**)
- dplyr.svg Unir datos que vienen de diferentes fuentes, referentes a un mismo conjunto estudiado. (**dplyr**)
- Hacer modificaciones: crear *tags*, correcciones ortográficas, filas y columnas de tablas, etc... (**tidyr**)
- magrittr_logo.png dplyr.png Generar nuevos datos: obtener promedios, medianas, aplicar funciones de librerías. (**magrittr** + **dplyr**)
- Visualiar/modelar datos (**ggplot2** + **tidymodels**)
- Dejar anotado y reportar lo hecho. (**rmarkdown**, **blogdown**)


```
library(tibble)

as_tibble(iris)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

- En *tidyverse* se suele trabajar sobre tablas en formato *tidy*, alojadas en objetos de clase **tibble**
- A las mismas se le aplican varios procesos

- 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.

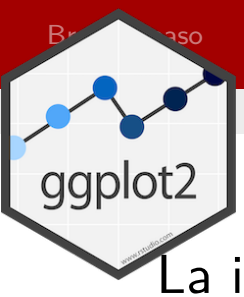
Operando secuencialmente sobre un dataset

```
# cargo las librerias que utilizamos en el practico
library(magrittr)
library(tidyverse)

# cargo tabla CSV, separo datos de una columna y filtro datos de una columna
aeropuertos_tibble = readr::read_csv('airport-codes.csv') %>%
  tidyr::separate(data = .,
                  col = 'coordinates',
                  into = c('lon', 'lat'),
                  sep = ', ') %>%
  dplyr::filter(is.na(iso_country))
```

Esta logica secuencial es sumamente practica. Pueden verla en accion si siguen este link a clase de practico, junto a otras utilidades de *tidyverse*.

Gramatica de graficos en capas



La idea central es que un grafico puede ser construido a traves de capas de elementos.

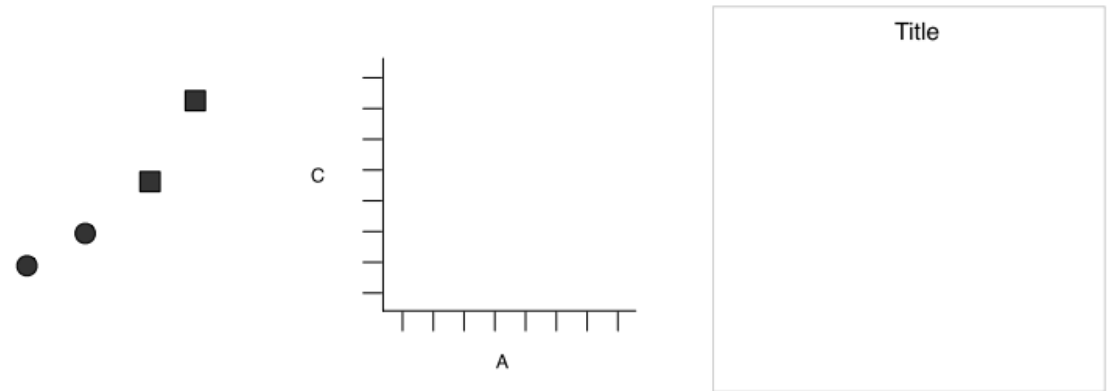


Figure 1. Graphics objects produced by (from left to right): geometric objects, scales and coordinate system, plot annotations.

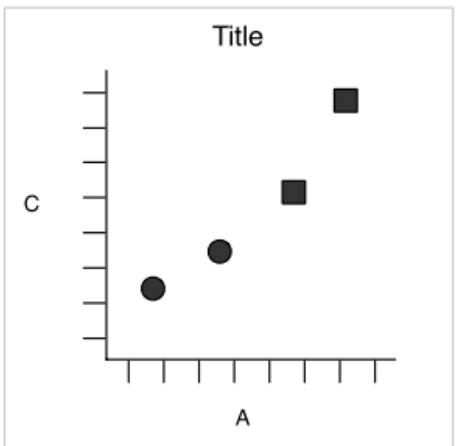
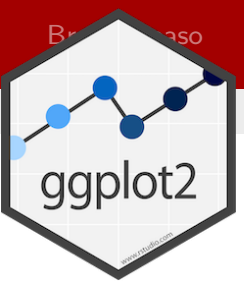
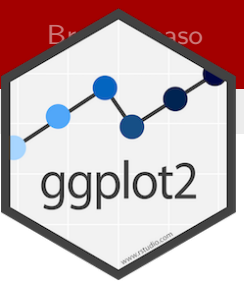


Figure 2. The final graphic, produced by combining the pieces in Figure 1.

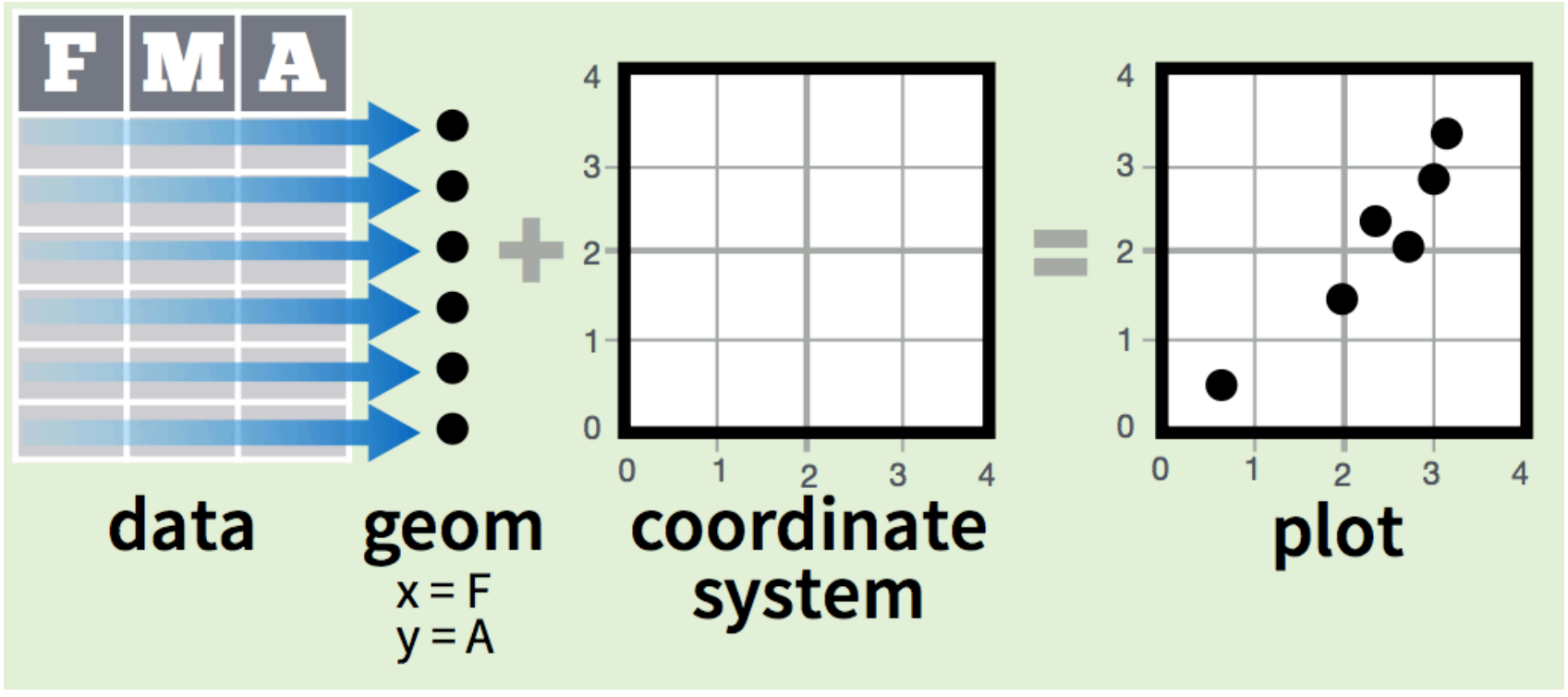


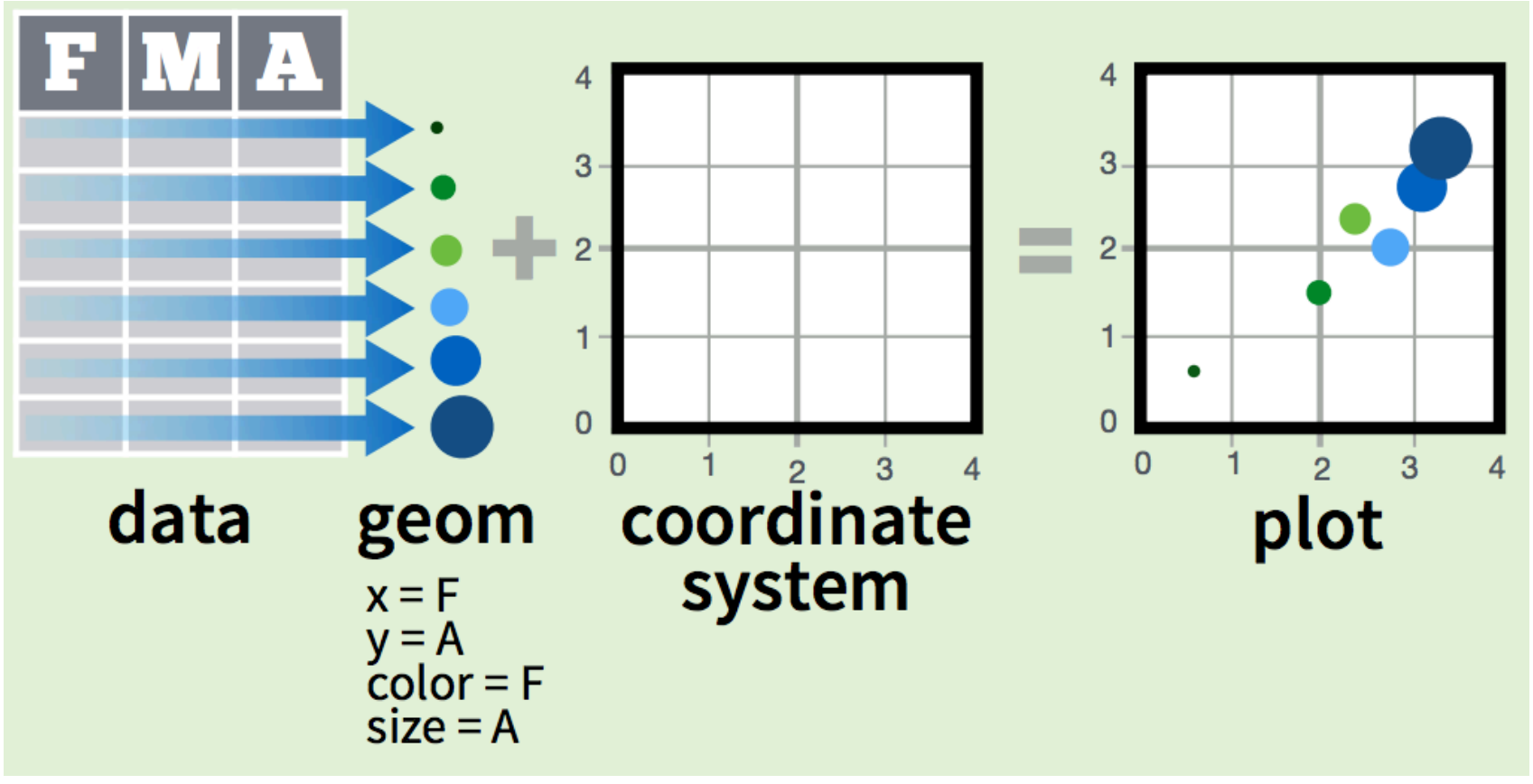
La idea central es que un grafico puede ser construido a traves de capas de elementos.

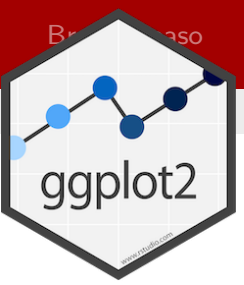
- Para realizar un gráfico preciso especificar:
  - Los **datos** sobre los que trabajo
  - Un sistema de coordenadas
  - Una especificación de qué representa cada dato a nivel **estético**
  - Una **forma geométrica** para representar estos datos
- Además, podríamos considerar
  - Especificar **funciones** que operen sobre nuestros datos, agrupándolos o transformándolos (pasa en histogramas, por ejemplo)
  - **Subdivisiones** de nuestros datos en base a factores.



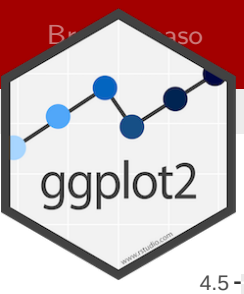
- Para realizar un gráfico preciso especificar:
  - Los **datos** sobre los que trabajo → funcion **ggplot()**
  - Un sistema de coordenadas → funcion **ggplot()**
  - Una especificación de qué representa cada dato a nivel **estético** → funcion **aes()**
  - Una **forma geométrica** para representar estos datos funciones → funciones **geom()_***
- Además, podríamos considerar
  - Especificar **funciones** que operen sobre nuestros datos, agrupándolos o transformándolos (pasa en histogramas, por ejemplo) → funcion **stat()**
  - **Subdivisiones** de nuestros datos en base a factores. → funcion **facet_wrap**

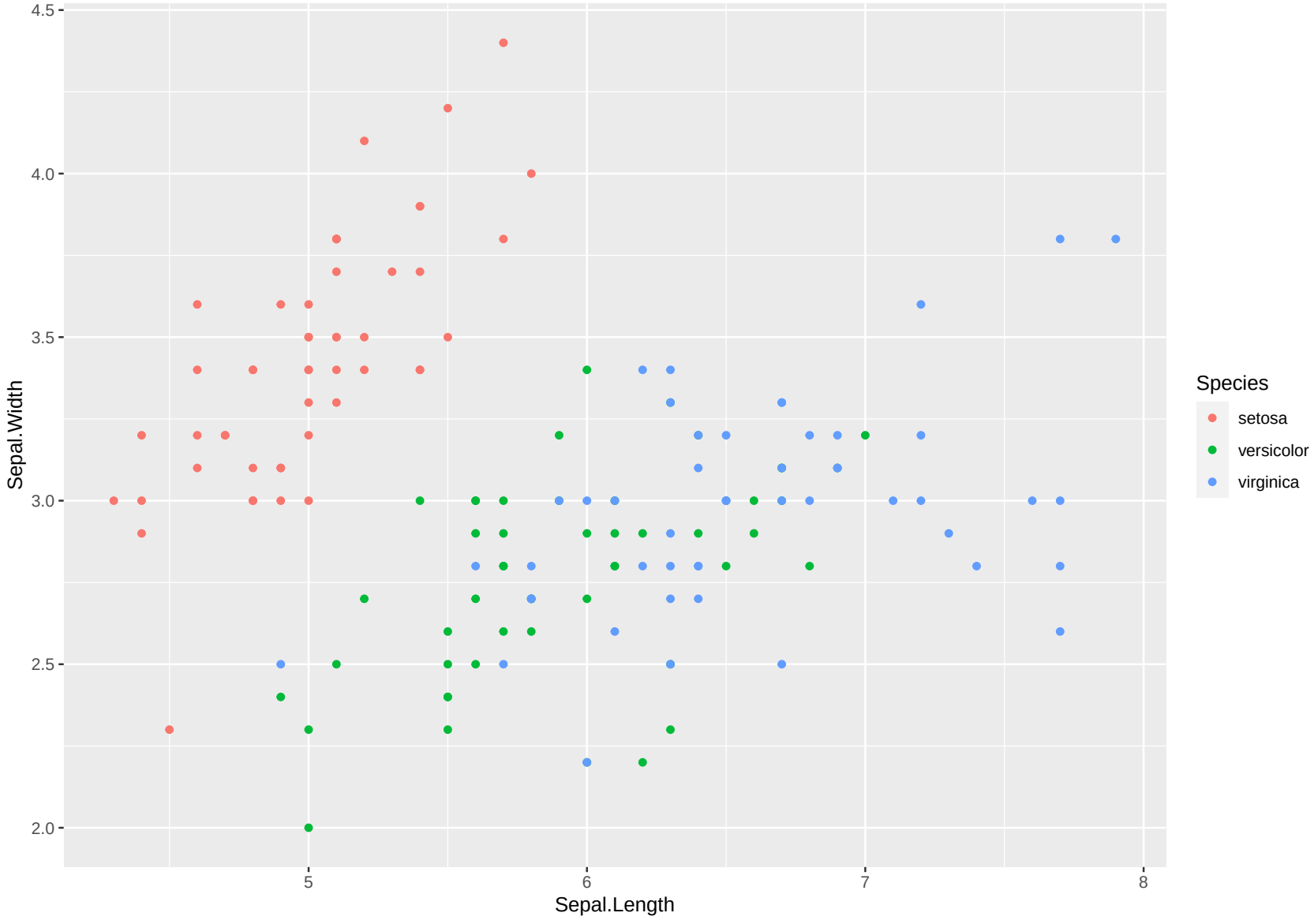

F
M
A
data
geom
x = F
y = A
coordinate
system
plot
0
1
2
3
4


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

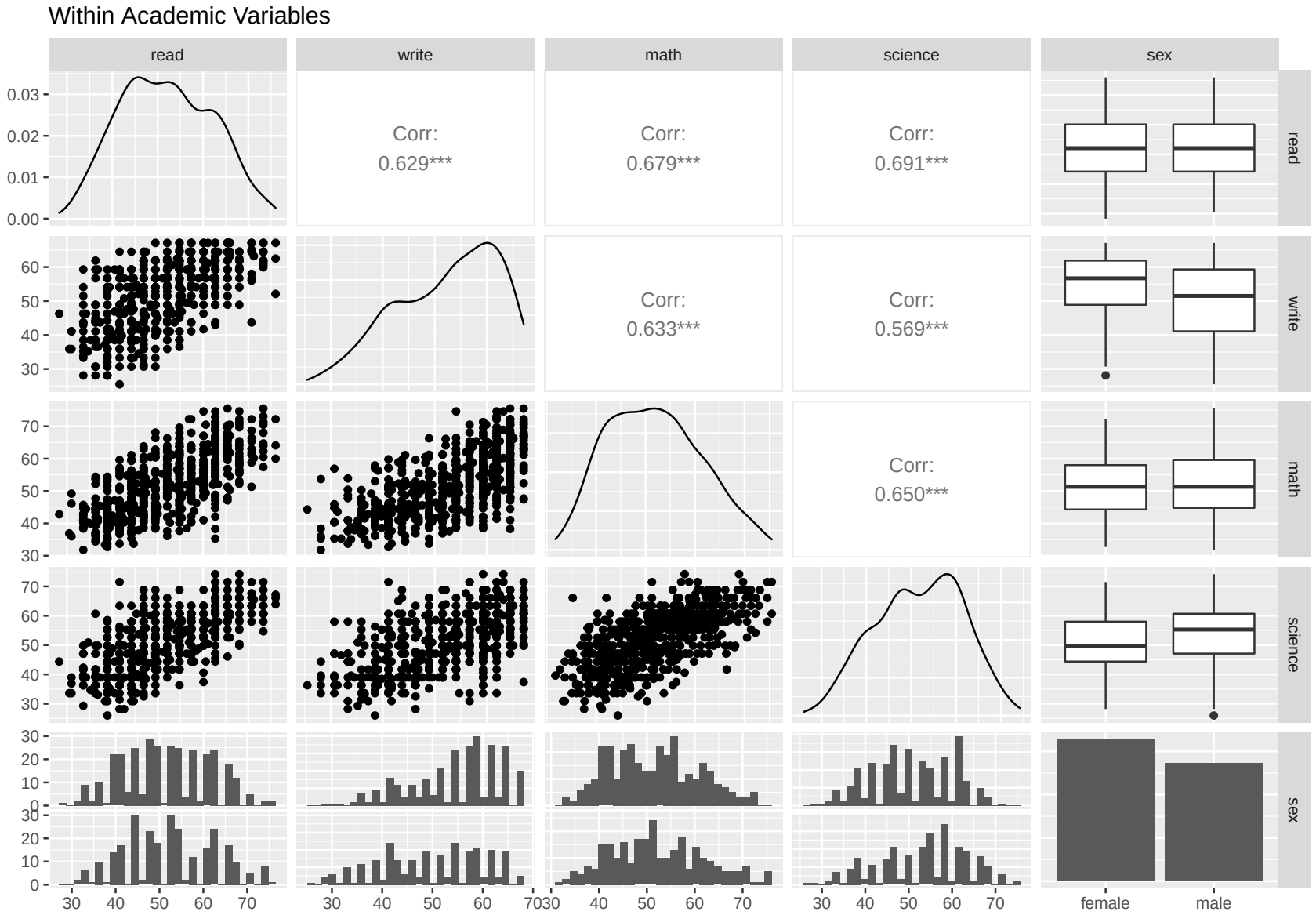



```
library(tidyverse)
  # se grafica Sepal.Length vs Sepal.Width,
  # coloreando segun Species
ggplot(data = iris,
       mapping = aes(x = Sepal.Length,
                     y = Sepal.Width,
                     color = Species,
                     fill = Species)) +
  # se grafica utilizando puntos
  geom_point()
```

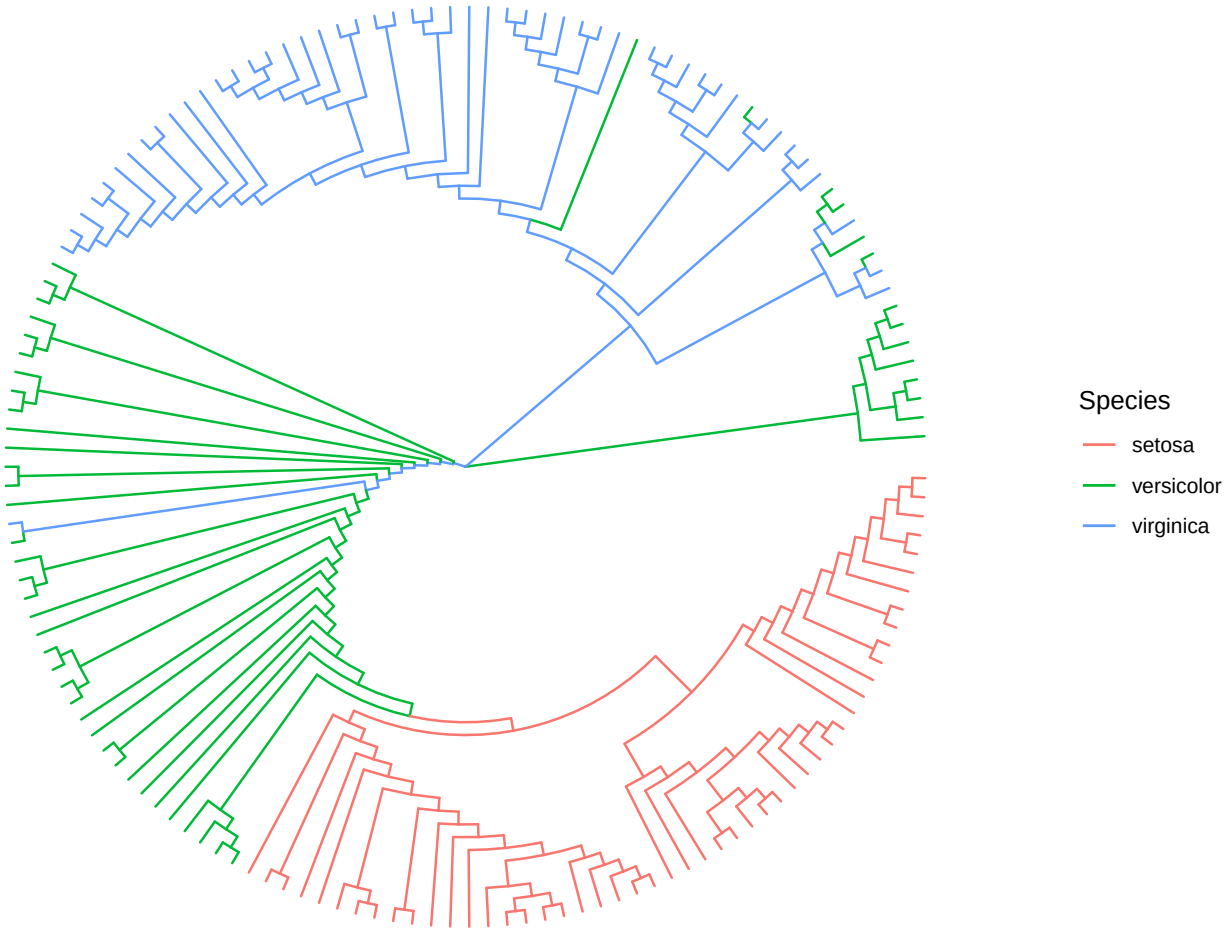

ggplot2


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

GGally: análisis exploratorios y otros

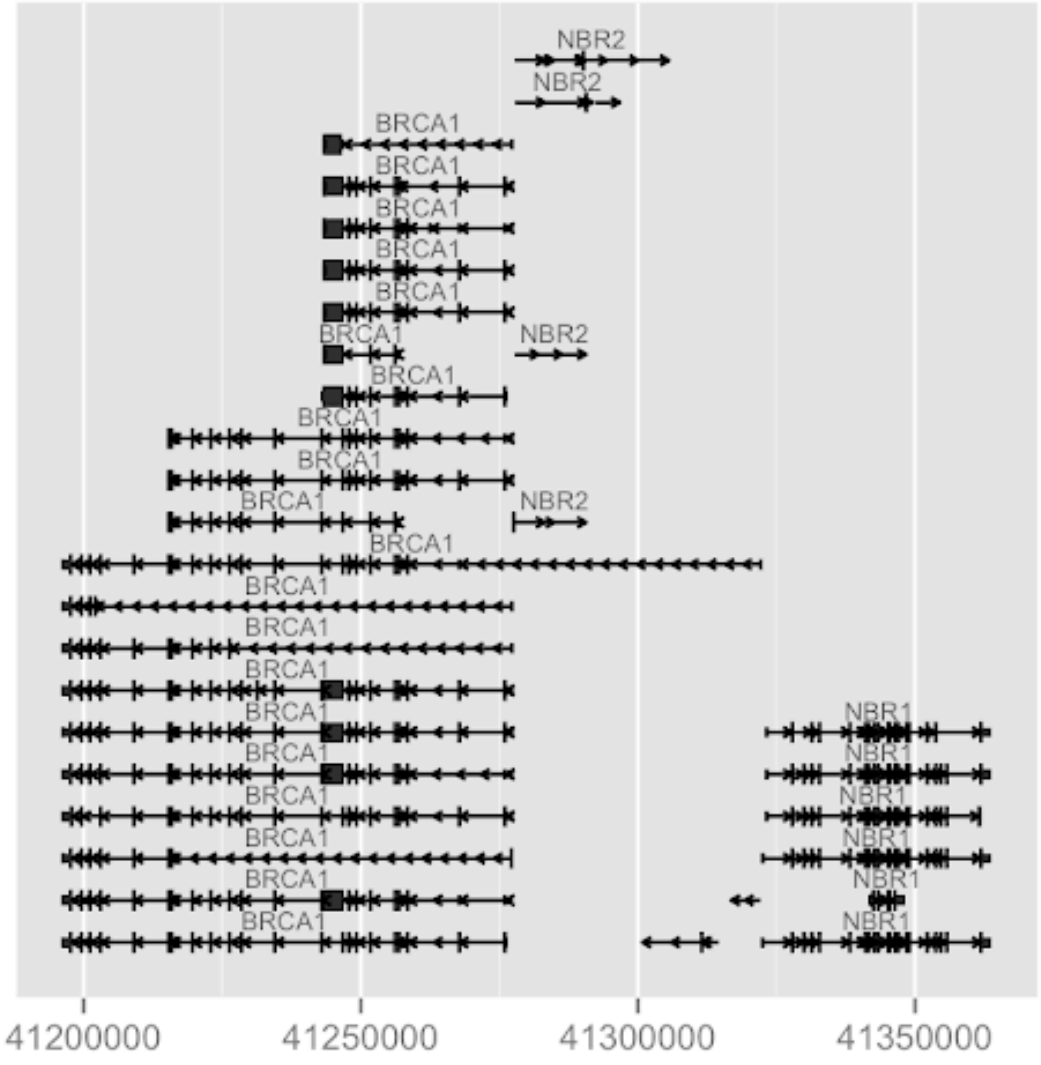


Filogenética: librería ggtree

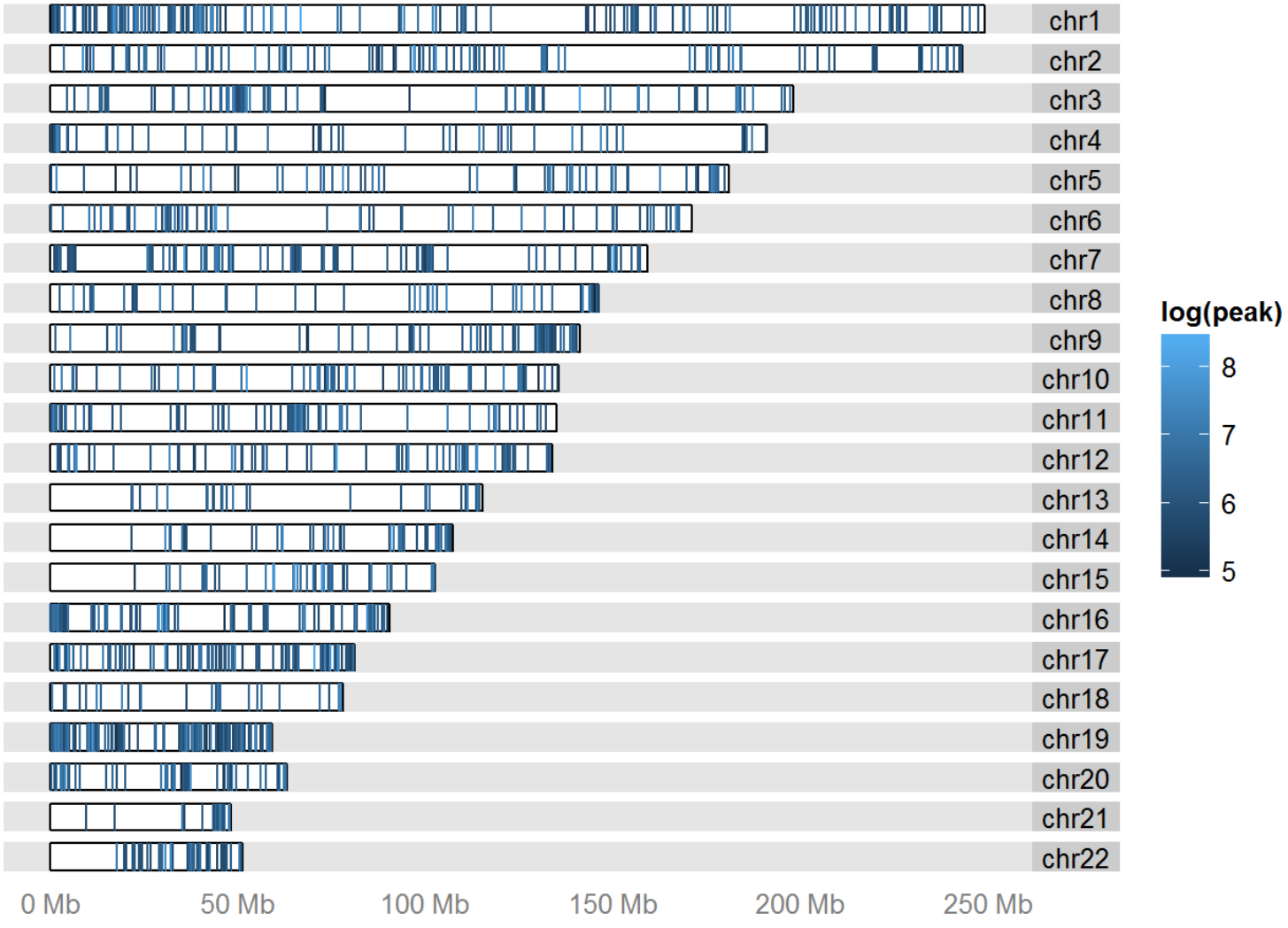


Genómica: BioCircos/**rcirclize** y gggnomics, ggbio

Genómica: BioCircos/rcirclize y gggnomics, **ggbio**

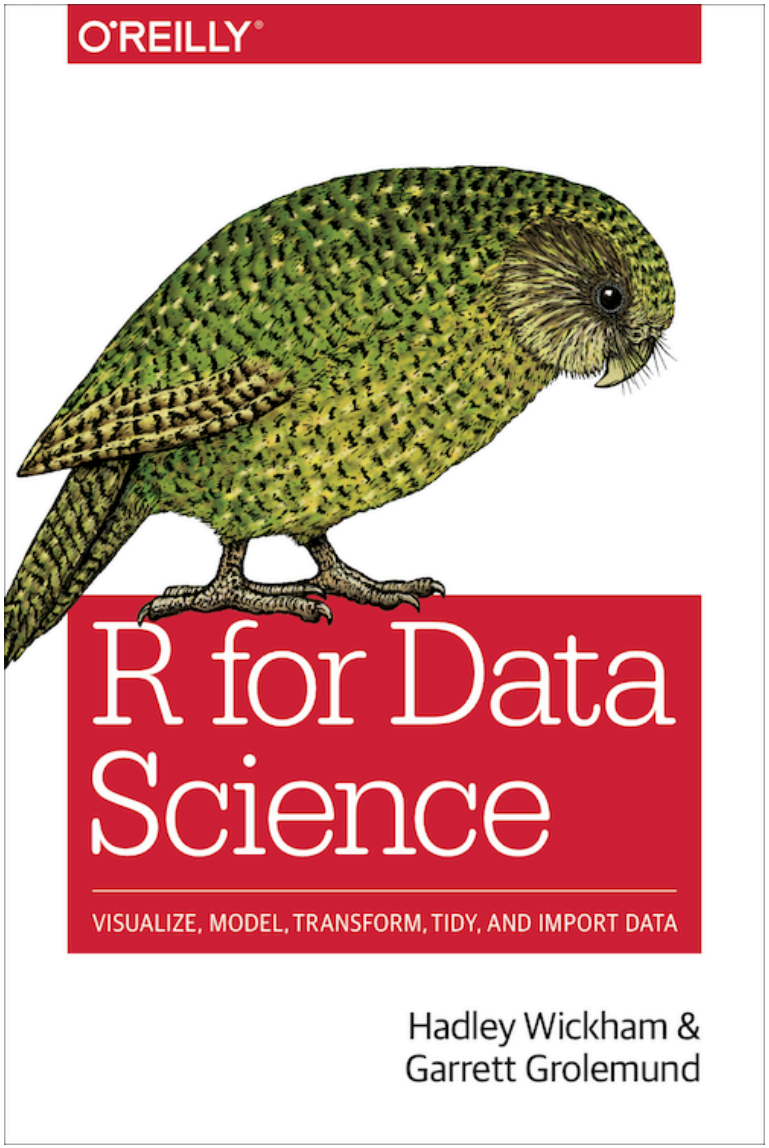


Genómica: BioCircos/rcirclize y gggnomics, **ggbio**



Lugares interesantes para leer

¿Donde encuentro info sobre estos paquetes?



¿Donde encuentro info sobre estos paquetes?

- *Cheatsheets* y *Vignettes* de las librerias
- *From Data to Viz* (link): una buena pagina para ver como se hacen algunos graficos especificos
- Paginas con cursos cortos, en especial:
  - *Software carpentry* (link), con cursos en ingles y espanhol
  - *Our coding club* (link), de un grupo de ecologia de Escocia
- Papers del grupo de Hadley Wickham (creador de tidyverse).
  - Estan bastante a la mano (no son de programacion pura y dura), y ayudan a entender la logica de fondo
  - Paper sobre *Tidy data* (link)
  - Paper sobre *Layered grammar of graphics* (link)