BIO 4022. Manipulación de datos e investigación reproducible en R

Derek Corcoran

2018-08-04

Contents

Parte I

Chapter 1

Requerimientos

La última versión de RStudio y R (R Core Team, 2018), también se requiere de los paquetes *pacman*, *tidyverse* y *tinytex*.

Si no han usado R o RStudio, pueden ver un video de como instalar ambos programas, asi como los paquetes más necesarios para este curso, que son *pacman*, *rmarkdown*, *tidyverse* y *tinytex* en el siguiente link.

El código para la instalación de esos paquetes es el siguiente

```
install.packages("pacman", "rmarldown", "tidyverse", "tinytex")
```

Si necesitan ayuda para la instalación contactarse con elinstructor del curso.

1.1 Si nunca has usado R antes

Si nunca han usado `R` antes de este curso, porfavor installar el paquete Swirl (Kross et al., 2017) y realizar los primeros 7 modulos del programa *R Programming: The basics of programming in R* que incluye:

- Basic Building Blocks
- Workspace and Files
- Sequences of Numbers
- Vectors
- Missing Values
- Subsetting Vectors
- Matrices and Data Frames

Pueden ver un video explicativo de como usar swirl en el siguiente Video

1.2 Descripción

Este curso está enfocado en entregar los principios de investigación reproducible en R, con énfasis en la recopilación y/o lectura de datos de forma reproducible y automatizada. Para esto se trabajará con bases de datos complejos, las cuales deberán ser transformadas y organizadas para optimizar su análisis. Se generarán documentos reproducibles integrando en un documento código, bibliografía, exploración y análisis de datos. Se culminará el curso con la generación de un manuscrito, presentación y/o documento interactivo reproducible.

1.3 Objetivos

1. Conocer y entender el concepto de Investigación reproducible como un forma y filosofía de investigación que permite que las investigaciones sean más ordenadas y replicables, desde la toma de datos hasta la escritura de resultados.

2. Conocer y aplicar el concepto de pipeline, el cual permite generar una modularidad desde la toma de datos hasta la escritura de resultados, donde la corrección independiente de un paso tiene un efecto cascada sobre el resultado final.

3. Aprender buenas prácticas de recolección y estandarización de bases de datos, con la finalidad de optimizar el análisis de datos y la revisión de estas por pares.

4. Realizar análisis críticos de la naturaleza de los datos al realizar análisis exploratorios, que permitirán determinar la mejor forma de comprobar hipótesis asociadas a estas bases de datos.

1.4 Contenidos

- Capítulo 2 *Tidy Data*: En este capítulo, aprenderemos de como optimizar una de base de datos, limpieza y transformación de bases de datos, que es una base de datos *tidy*, y como manipular estas bases de datos con el paquete *dplyr* (**?**)

- Capitulo 3 *Investigación reproducible*: entenderemos el como generar un documento que combine códigos de `R` y texto para generar documentos reporducibles utilizando el paquete *rmarkdown* (Allaire et al., 2018), además veremos como al usar RStudio podemos guardar nuestro proyecto en un repositorio de github.

- Capitulo 4 *El tidyverso*, el concepto de pipeline. Limpieza de datos complejos

5. Visualización de datos, visualizar datos vs. visualizar modelos. Insertar gráficos con leyenda en un documento Rmd

6. Creación de funciones propias y loops. Generación de funciones propias en R y loops

7. Escritura de manuscritos en R, transformación de documentos Rmd en un manuscrito

8. Presentaciones en R y generar documentos interactivos. Transformación de datos en una presentación o en una Shiny app. Realizar una presentación o aplicación en R.

1.5 Metodología

Todas las clases serán prácticas y estarán divididas en dos partes: I. Clases de principios y herramientas, donde se presentarán los principios de investigación reproducible y tidy data, junto con las herramientas actuales más utilizadas, y II. Clases aplicadas donde se trabajará con datos propios para desarrollar un documento reproducible. A los estudiantes que no cuenten con datos propios, se les será proporcionado un set de datos o se simularán dependiendo del caso.

Además se deberán generar informes y presentaciones siguiendo los principios de investigación reproducible, usando datos propios o entregados. Se realizará un informe final, en el cual se espera un trabajo

1.6 Evaluación

- Evaluación 1: Informe exploratorio de base de datos 25%
- Evaluación 2: Presentación 25%
- Evaluación 3: Informe final 50%

1.7 Objetivo del curso

Aprender los principios de investigación reproducible y tidy data a través del aprendizaje de programación y uso de R. Los principios de este curso están explicados en los siguientes libros gratuitos.

- Gandrud, Christopher. Reproducible Research with R and R Studio. CRC Press, 2013. Available for free in the following link
- Stodden, Victoria, Friedrich Leisch, and Roger D. Peng, eds. Implementing reproducible research. CRC Press, 2014. Available for free in the following link

1.8 Bibliografía

Chapter 2

Tidy Data y manipulación de datos

En este capítulo explicaremos que es una base de datos *tidy* (Wickham et al., 2014) y aprenderemos a usar funciones del paquete *dplyr* (**?**) para manipular datos.

Recuerda que este libro es un apoyo para el curso BIO4022, puedes seguir la clase de este curso en este link, y en cuanto el video de la clase este disponible encontrarás un link aca.

2.1 Tidy data

Una base de datos tidy es una base de datos en la cuál (modificado de (Leek, 2015)):

- Cada varaible que midas debiera estar en una columna.
- Cada observación distinta de esa variable debiera estar en una fila diferente.

En general, la forma en que representarímos una base de datos *tidy* en `R` es usando un *data frame*.

2.2 dplyr

El paquete *dplyr* es definido por sus autores como una gramática para la manipulación de datos. De este modo sus funciones son conocidas como verbos. Un resumen útil de muchas de estas funciones se encuentra en este link.

Este paquete tiene un gran número de verbos y sería difícil ver todos en una clase, en este capítulo nos enfocaremos en sus funciones más utilizadas, las cuales son:

- *group_by* (agrupa datos)
- *summarize* (resume datos agrupados)
- *filter* (Encuentra filas con ciertas condiciones)
- *select* junto a *starts_with*, *ends_with* o *contains*
- *mutate* (Genera variables nuevas)
- *%>%* pipeline

Table 2.1: una tabla con 10 filas de la base de datos iris.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.8	4.0	1.2	0.2	setosa
4.7	3.2	1.6	0.2	setosa
5.1	3.8	1.9	0.4	setosa
5.2	2.7	3.9	1.4	versicolor
6.4	2.9	4.3	1.3	versicolor
5.5	2.5	4.0	1.3	versicolor
6.5	3.0	5.8	2.2	virginica
6.0	2.2	5.0	1.5	virginica
6.1	2.6	5.6	1.4	virginica
5.9	3.0	5.1	1.8	virginica

Table 2.2: Resumen del promedio y desviación estándar del largo de pétalo de las flores del generi Iris.

Mean.Petal.Length	SD.Petal.Length
3.758	1.765298

2.2.1 summarize

la función `summarize`, toma los datos de un data frame y las resume. Para usar esta función, el primer argumento que tomaríamos sería un data frame, seguido del nombre que queremos darle a una variable resumen, seguida del signo = y luego la formula a aplicar a una o mas columnas. Por ejemplo si tomaramos la base de datos `iris` (Anderson, 1935), que viene en `R`, y de las cual podemos ver parte de sus datos en la tabla 2.1

Si quisieramos resumir esa tabla y generar un par de variables que fueran la media y la deviación estándar de el largo del pétalo, lo haríamos con el siguiente código:

```
library(tidyverse)
Summary.Petal <- summarize(iris, Mean.Petal.Length = mean(Petal.Length), SD.Petal.Length = sd(Petal.Leng
```

El resultado lo vemos en la tabla 2.2, en el cuál obtenemos los promedios y desviaciones estándar de los largos de los pétalos. Es importante notar que al usar summarize, todas las otras variables desapareceran de la tabla.

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Table 2.3: Resumen del promedio y desviación estándar del largo de pétalo de las flores del generi Iris.

Species	Mean.Petal.Length	SD.Petal.Length
setosa	1.462	0.1736640
versicolor	4.260	0.4699110
virginica	5.552	0.5518947

2.2.2 group_by

La función `group_by` por si sola no genera cambios visibles en tu base de datos, sin embargo al ser utilizada en conjunto con `summarize` te permite resumir una variable agrupada (usualmente) basada en una o más variables categóricas.

Como ejemplo seguiremos con el caso de las plantas del género *Iris*, el resumen que teníamos en el caso de la tabla 2.2, no es tan útil considerando que tenemos tres especies presentes en la tabla de datos. Si queremos ver el promedio del largo del pétalo por especie, ocupamos la función `group_by` de la siguiente forma:

```
BySpecies <- group_by(iris, Species)
Summary.Byspecies <- summarize(BySpecies, Mean.Petal.Length = mean(Petal.Length), SD.Petal.Length = sd(
```

Esto nos dá como resultado la tabla 2.3, con la cuál ya podemos ver que *Iris setosa* es tiene pétalos mucho más cortos que las otras dos especies del mismo género.

2.2.2.1 group_by en mas de una variable

Podemos usar la función `group_by` en más de una variable, y esto generaría un resumen anidado. Como ejemplo usaremos la base de datos `mtcars` presente en R (Henderson and Velleman, 1981), en el cuál hay una variable *mpg* (Miles per gallon), una medida de eficiencia de combustible, y resumiremos esto en base a la variable *am*, (que se refiere al tipo de transmición, donde 0 es automático y 1 es manual) y al número de cilindros del motor, para eso usamos el siguiente código:

```
Grouped <- group_by(mtcars, cyl, am)
Eficiencia <- summarize(Grouped, Eficiencia = mean(mtcars))
```

```
Grouped <- group_by(mtcars, cyl, am)
Eficiencia <- summarize(Grouped, Eficiencia = mean(mtcars))
```

```
## Warning in mean.default(mtcars): argument is not numeric or logical:
## returning NA

## Warning in mean.default(mtcars): argument is not numeric or logical:
## returning NA

## Warning in mean.default(mtcars): argument is not numeric or logical:
## returning NA

## Warning in mean.default(mtcars): argument is not numeric or logical:
## returning NA

## Warning in mean.default(mtcars): argument is not numeric or logical:
```

Table 2.4: Millas por galón promedio en vehiculos automáticos (am = 0) y manuales (am = 1), con los distintos tipos de cilindros

cyl	am	Eficiencia
4	0	NA
4	1	NA
6	0	NA
6	1	NA
8	0	NA
8	1	NA

```
## returning NA

## Warning in mean.default(mtcars): argument is not numeric or logical:
## returning NA
```

Vemos en la tabla 2.4, que

Chapter 3

Investigación reproducible

Here is a review of existing methods.

Chapter 4

El Tidyverso

We describe our methods in this chapter.

Chapter 5

Visualización de datos

En este capítulo aprenderemos a usar el paquete *ggplot2* (Wickham, 2016), parte del paquete *tidyverse* (Wickham, 2017).

5.1 El esqueleto

El esqueleto de una visualización usando *ggplot2* es la siguiente

```
ggplot(data.frame, aes(nombres de columna)) + geom_algo(argumentos, aes(columnas)) + theme_algo()
```

Como ejemplo para discutir usaremos el siguiente código que genera la figura 5.1:

```
library(tidyverse)
data("diamonds")
ggplot(diamonds, aes(x = carat, y=price)) + geom_point(aes(color = cut)) + theme_classic()
```

En este caso general, lo primero que ponemos después de ggplot es el data.frame desde el cuál graficaremos algo, en el ejemplo de la figura 5.1 usamos la base de datos *diamonds* del paquete *ggplot2* (Wickham, 2016). Luego dentro de `aes` ponemos las columnas que graficaremos como x y/o y, en nuestro ejemplo dentro de aes ponemos como eje x los kilates de los diamantes (caret) y como y el precio de los mismos (price). La necesidad de poner `aes` en ggplot2 (algo que no había sido necesario cuando usamos *dplyr* o *tidyr*) es que ggplot2 es el paquete mas antiguo del *tidyverse*.

5.2 geom_algo

Luego de especificar una base de datos, esto viene seguido de un `geom_algo`, esto nos indicará que tipo de gráfico usaremos, estos pueden ser combinados como veremos en ejemplos futuros

5.2.1 Una variable categórica una continua

Primero veremos algunos de los *geom* que podemos utilizar con una variable categórica y una continua

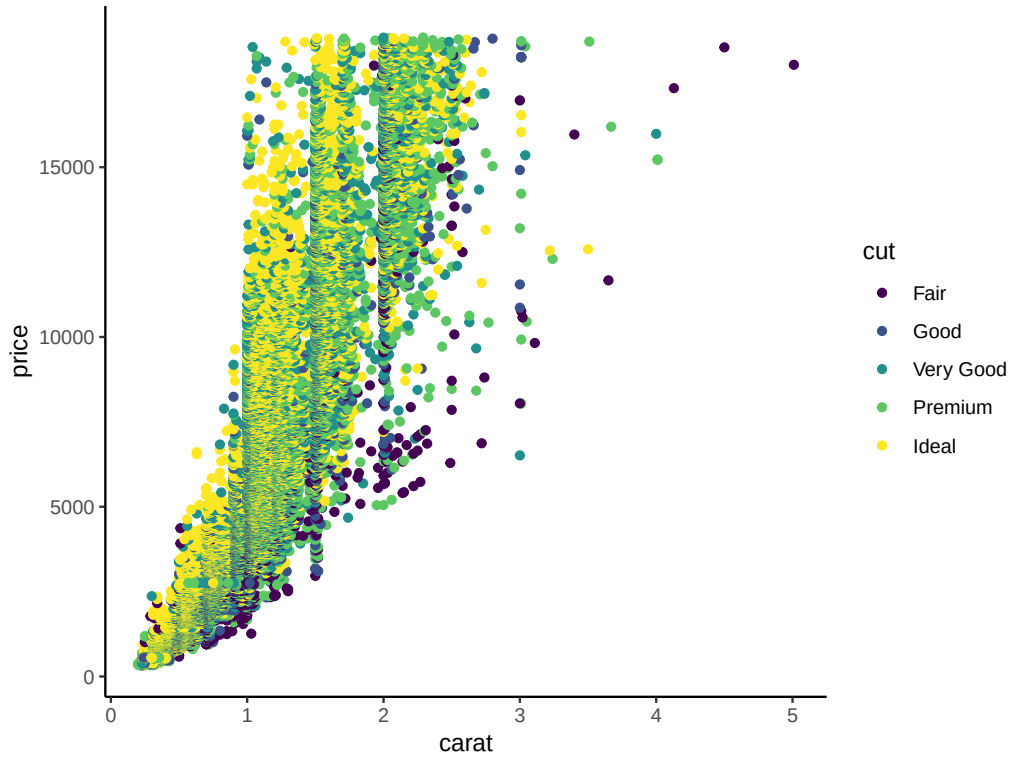

Figure 5.1: Gráfico en el cual gráficamos los quilates de diamantes versus su precio, con el corte del diamante representado por el color

5.2.1.1 geom_boxplot

En la figura 5.2, generado a partir del código a continuacón con la base de datos iris presente en `R` (Anderson, 1935).

```
data("iris")
ggplot(iris, aes(x = Species, y = Sepal.Length)) + geom_boxplot()
```

Los boxplots muestran una linea gruesa central (la mediana), una caja, que delimita el primer y tercer cuartil, y los bigotes, los cuales se extienden hasta los valores extremos. A menos que estos esten por sobre 1.5 veces la distance entre el primer y tercer cuartil, en cuyo caso se consideran outlyers, y estos son representados por puntos. En la figura 5.2, solo *Iris virginica* presenta un outlayer en cuanto a las medidas del largo del sepalo.

Los boxplots, como todos los gráficos pueden ser personalizados usando otros argumentos, los cuales son detallados en la sección 5.3, pero en los ejemplos que mostraremos en esta sección los iremos introduciendo de a poco. Si quisieramos por ejemplo que el color de las cajas del *boxplot* fuera deacuerdo a la especie, cambiamos el llenado (**fill**) de la caja, como vemos en el siguiente ejemplo y figura 5.3

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) + geom_boxplot(aes(fill = Species))
```

Dos cosas a notar en este ejemplo, por un lado la leyenda se genera de forma automática, y por otro lado, vemos que es necesario poner *Species* dentro de `aes`, esto es debido a que Species es una columna y como se explicó al principio de este capítulo, todas las columnas deben ser incuidas dentro de la función `aes` para poder ser referenciadas.

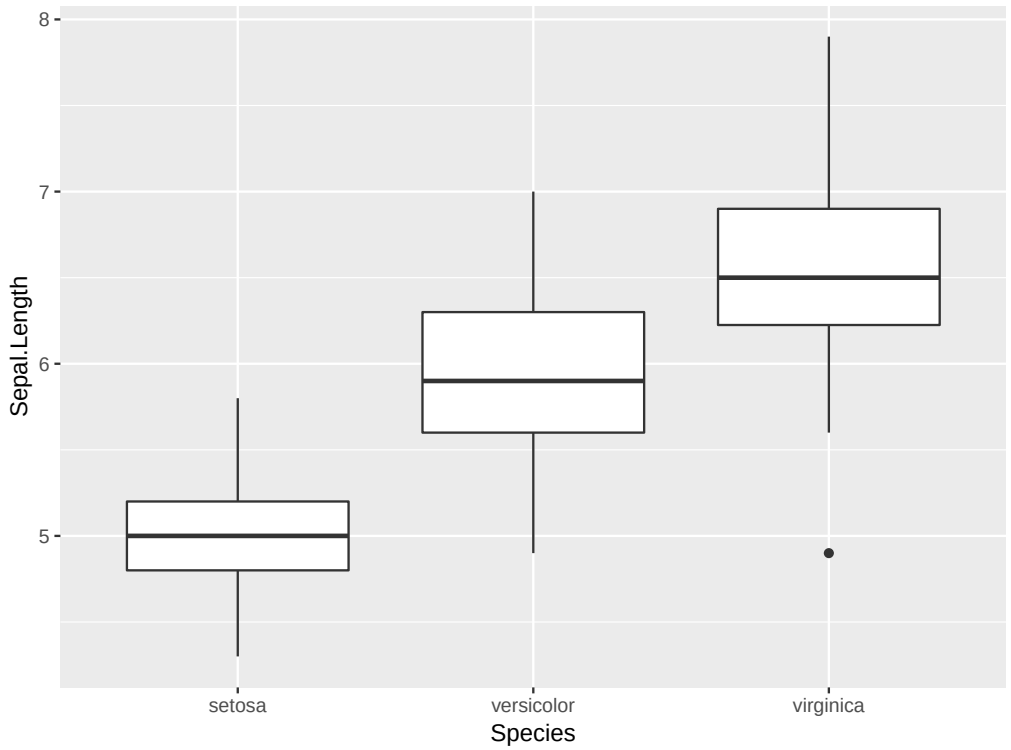

Figure 5.2: Boxplot que representa los largos del sépalo de tres especies del género Iris

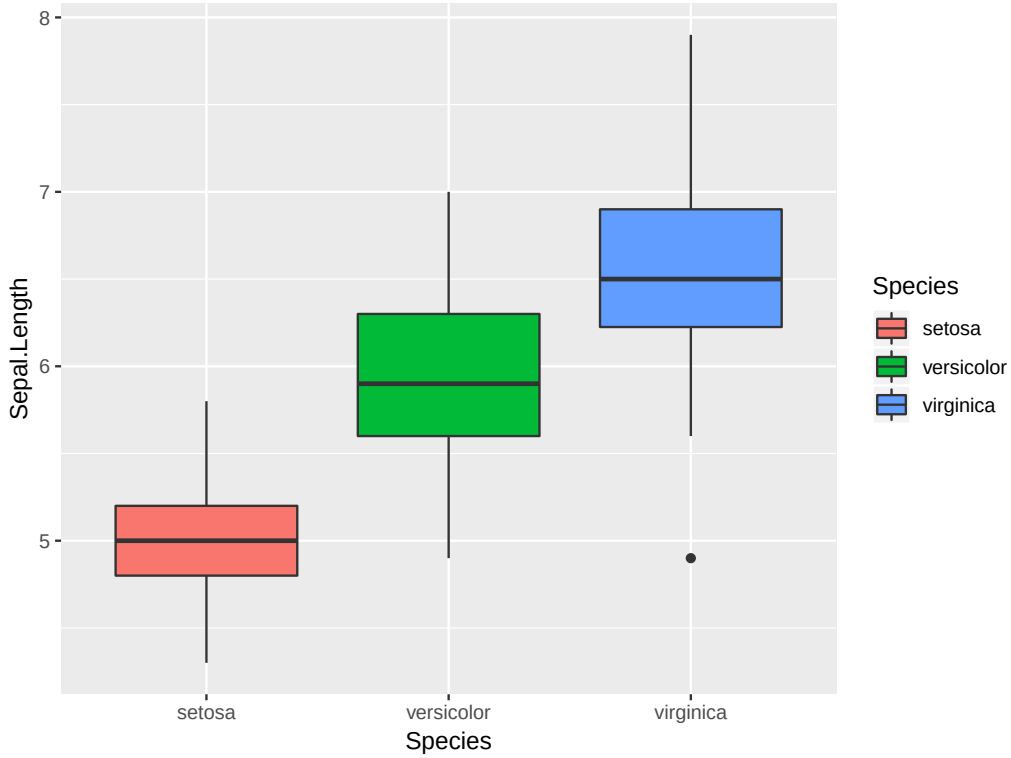

Figure 5.3: Boxplot que representa los largos del sépalo de tres especies del género Iris, en este caso el color de la caja representa la especie

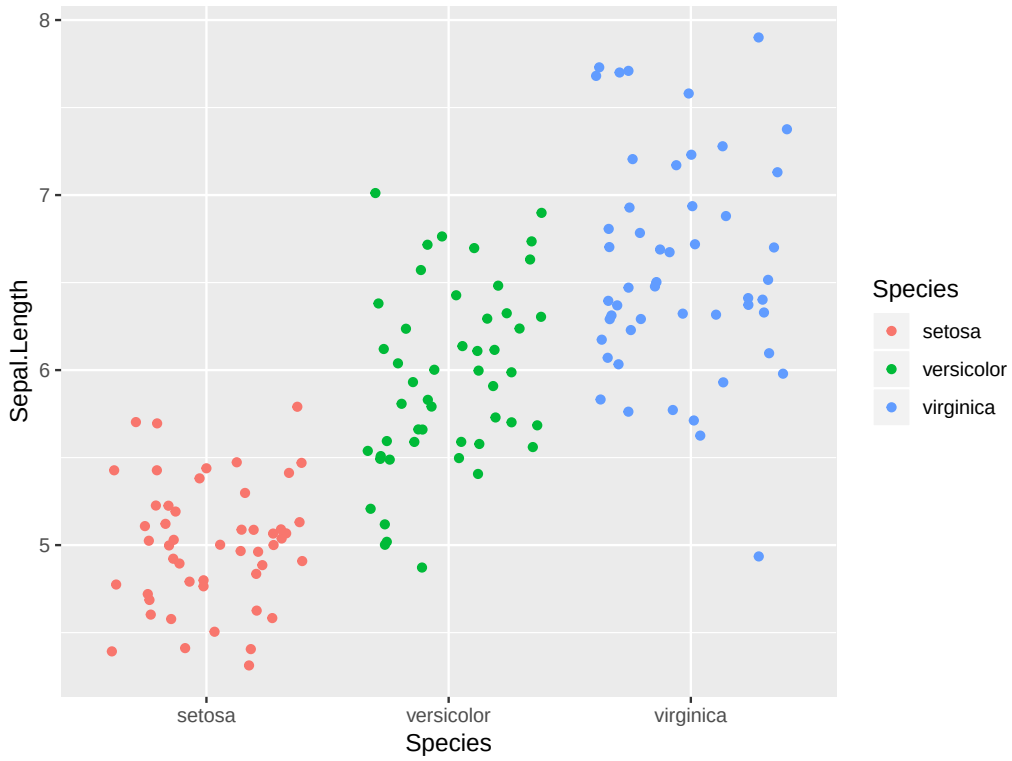

Figure 5.4: Boxplot que representa los largos del sépalo de tres especies del género Iris, en este caso el color de la caja representa la especie

5.2.1.2 geom_jitter

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) + geom_jitter(aes(color = Species))
```

5.3 Argumentos

Chapter 6

Modelos en R

We have finished a nice book.

Chapter 7

Loops (purrr) y bibliografía (rticles)

Chapter 8

Presentaciones en R

Bibliography

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., and Chang, W. (2018). *rmarkdown: Dynamic Documents for R*. R package version 1.10.

Anderson, E. (1935). The irises of the gaspe peninsula. *Bulletin of the American Iris society*, 59:2–5.

Henderson, H. V. and Velleman, P. F. (1981). Building multiple regression models interactively. *Biometrics*, pages 391–411.

Kross, S., Carchedi, N., Bauer, B., and Grdina, G. (2017). *swirl: Learn R, in R*. R package version 2.4.3.

Leek, J. (2015). The elements of data analytic style. *J. Leek.—Amazon Digital Services, Inc.*

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H. (2017). *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.2.1.

Wickham, H. et al. (2014). Tidy data. *Journal of Statistical Software*, 59(10):1–23.