Practica de Análisis de datos con R

Abiel Guillermo Flores Buezo

3/22/2016

Práctica de analisis de datos con R

```
## R version 3.2.3 (2015-12-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux stretch/sid
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.12.3  ggvis_0.4.2   reshape_0.8.5
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.4     digest_0.6.9    dplyr_0.4.3     assertthat_0.1
##  [5] mime_0.4        R6_2.1.2        plyr_1.8.3      xtable_1.8-2
##  [9] DBI_0.3.1       formatR_1.3     magrittr_1.5    evaluate_0.8.3
## [13] stringi_1.0-1   rmarkdown_0.9.5 tools_3.2.3     stringr_1.0.0
## [17] shiny_0.13.2    httpuv_1.3.3    parallel_3.2.3  yaml_2.1.13
## [21] htmltools_0.3.5
```

Introducción

Esta es una práctica de análisis y visualización de datos utilizando **RStudio**, y el modulo **knitr** del mismo. Para llevarla a cabo, haré uso del siguiente dataset Student Performance Data Set cuya información puede ser encontrada aquí.

Lo primero que debemos hacer es situarnos en eldirectorio de trabajo que vamos a utilizar. Para establecer nuestro directorio de trabajo utilizamos el comando `setwd("/home/rstudio/dataAnalysisPractice")`. Si queremos saber cual es el directorio de trabajo en el cual nos encontramos ejecutamos el comando `getwd()`.

Los archivos de datos son los siguientes:

```
## [1] "student-mat.csv"
## [1] "student-merge.R"
## [1] "student-por.csv"
## [1] "student.txt"
```

`student-mat.csv` y `student-por.csv` son los archivos que nos interesan, ya que contienen los datos. `student-merge.R` es un Rscript para juntar los datos de los anteriores archivos, pero no lo haremos de esa manera. Finalmente `student.txt` contiene la descripción de los datos en los primeros dos archivos.

Lectura de datos

Ya que los archivos a los que vamos a acceder tienen formato *CSV*, sabemos que los valores están "ordenados" y separados por algún tipo de separador. Si echamos un vistazo en los archivos, veremos que el separador utilizado es el punto y coma (;). En este punto podríamos utilizar diversos métodos, pero R cuenta con el comando `read.csv`, el cual leerá estos datos y creará un dataframe a partir de ellos:

```
dt.students.mat <- read.csv("./data/student-mat.csv", header=TRUE, sep = ";", quote="\"")
dt.students.por <- read.csv("./data/student-por.csv", header=TRUE, sep = ";", quote="\"")
```

Con los parámetros que pasamos, le indicamos varias cosas. En concreto, que la primera linea de el archivo contiene los nombres de los campos en esas posiciones, y que deseamos que las reconozca así, que el separador es el *punto y coma*, y que algunos campos (sobre todo los campos de tipo cadena de caracteres) están entre *comillas dobles*.

Ahora disponemos de dos dataframes, uno con los alumnos en la clase de matematicas, y otro con los alumnos en la clase de portugués.

Cada dataframe creado consta de 33 columnas, o *variables*. Esta es una muestra de los datos a los que nos enfrentamos en la clase de matemáticas, con solo 10 *variables*:

```
head(dt.students.mat[1:10])
```

```
##   school sex age address famsize Pstatus Medu Fedu     Mjob     Fjob
## 1     GP   F  18       U     GT3       A    4    4  at_home  teacher
## 2     GP   F  17       U     GT3       T    1    1  at_home    other
## 3     GP   F  15       U     LE3       T    1    1  at_home    other
## 4     GP   F  15       U     GT3       T    4    2   health services
## 5     GP   F  16       U     GT3       T    3    3    other    other
## 6     GP   M  16       U     LE3       T    4    3 services    other
```

Preparación y transformación de datos

Los datos no siempre están ordenados o son útiles cuando nos los entregan. Estos datos en su estado bruto pueden contar de varios inconvenientes: falta de valores, tipos de datos incorrectos, unidades de medida incompatibles, codificado de variables incorrecto, etc... En estas situaciónes es necesario que preparemos y transformemos los datos para que puedan ser utilizados en el análisis de datos que estamos por ejecutar. Este proceso tiene varios nombres. Los más sonados *Data Munging, Data Cleaning, Data Cleansing, Data Wrangling*... Y consta de tareas como las siguientes:

- Renombrar variables
- Convertir tipos de dato
- Codificar valores
- Juntar datasets
- Convertir unidades
- Hacer algo con los valores que faltan
- Hacer algo con los valores anomalos o sin sentido
- etc...

Afortunadamente los datasets de los que constamos están más bien limpios, y no necesitamos realizar muchos cambios. Y como nos interesa hacer un pequeño análisis de las clases por separado, no juntaremos los datasets hasta que lo necesitemos.

Una de las cosas que podemos hacer es formatear los nombres de las columnas, para ayudar a trabajar más flexiblemente con ellas.

```
names(dt.students.mat) <- tolower(names(dt.students.mat))
names(dt.students.por) <- tolower(names(dt.students.por))
names(dt.students.mat)
```

```
##  [1] "school"     "sex"        "age"        "address"    "famsize"
##  [6] "pstatus"    "medu"       "fedu"       "mjob"       "fjob"
## [11] "reason"     "guardian"   "traveltime" "studytime"  "failures"
## [16] "schoolsup"  "famsup"     "paid"       "activities" "nursery"
## [21] "higher"     "internet"   "romantic"   "famrel"     "freetime"
## [26] "goout"      "dalc"       "walc"       "health"     "absences"
## [31] "g1"         "g2"         "g3"
```

Los valores de la columna `famsize` son o bien `LE3` o `GT3`, lo cual significa que o bien la familia consta de 3 o menos integrantes, o bien consta de más de 3 integrantes. Si agregaramos valores, nosotros mismos, podríamos quitarle validez a los datos, así que es mejor que permanezcan así.

Por comprobar un poco si tenemos algún valor *NA* podemos utilizar estos comandos para saber el número de valores *NA* que hay en cada columna:

```
# sapply(dt.students.mat, function(x) sum(is.na(x)))
# apply(is.na(dt.students.mat),2,sum)
# colSums(is.na(dt.students.mat))
sum(is.na(dt.students.mat))
```

```
## [1] 0
```

Afortunadamente no tenemos campos vacios, y los datos están ya en un formato que se puede utilizar.

Análisis exploratorio

Cada una de las entradas en estos datos corresponde a los datos de un alumno. Si echamos un vistazo o si relacionamos las variables entre ellas podemos obtener información interesante.

Por ejemplo. El número total de alumnos de cada escuela:

```
## [1] "Numero de alumnos para cada escuela, en la clase de matemáticas:"
```

```
##
##  GP  MS
## 349  46
```

```
## [1] "Numero de alumnos para cada escuela, en la clase de portugués:"
```

```
##
##  GP  MS
## 423 226
```

En general hay menos alumnos en la clase de Matemáticas.

```
print("Clase de matemáticas. Alumnos con dirección Rural o Urbana por escuelas:")
```

```
## [1] "Clase de matemáticas. Alumnos con dirección Rural o Urbana por escuelas:"
```

```
cast(dt.students.mat, school~address, length, value=c("g3"))
```

```
##   school  R   U
## 1     GP 63 286
## 2     MS 25  21
```

```
print("Clase de Portugués Alumnos con dirección Rural o Urbana por escuelas:")
```

```
## [1] "Clase de Portugués Alumnos con dirección Rural o Urbana por escuelas:"
```

```
cast(dt.students.por, school~address, length, value=c("g3"))
```

```
##   school   R   U
## 1     GP  78 345
## 2     MS 119 107
```

Hemos descubierto que la escuela "Mousinho da Silveira" tiene más alumnos que proceden de algún pueblo, probablemente rodeando la ciudad, que la escuela "Gabriel Pereira" que seguramente esté ubicada más centricamente dentro de la ciudad, o tenga mejores accesos a los transportes de la misma.

Otro sitio donde podemos fijar nuestra mirada es en las variables correspondientes al tiempo que le toma al alumno llegar a la escuela, y el tiempo de estudio de el que dispone.

La lógica nos dice que si un alumno gasta tiempo "viajando", reduce el tiempo que tiene disponible para estudiar. Veamos si esto concuerda con los datos de los que disponemos, comprobando la covarianza con `cov()`:

- Covarianza matemáticas: -0.0590696
- Covarianza portugués: -0.0392199

La covarianza que obtenemos es negativa, lo cual significa que si el tiempo de viaje incrementa, el tiempo de estudio decrementa. Aun así necesitamos saber qué tan crítica es la relación entre estas dos variables, para lo que utilizaremos `cor()` para saber la correlación:

- Matemáticas: -0.1009091
- Portugués: -0.0631539

La correlación al ser tan baja nos indica que aunque estas variables pueden estar relacionadas, la influencia que tienen la una en la otra, no es significativa.

Vamos a continuar uniendo los dos dataframes para trabajar con uno solo. Ya que la descripción nos indica que hay unos 382 alumnos que están repetidos entre los 2 dataframes, vamos a unirlos de manera que trabajemos con esos, ya que asisten a ambas clases.

Para unir los dataframes hacemos de la siguiente manera:

```
dt.students.mp = merge(dt.students.mat, dt.students.por, by = c("school", "sex",
  "age", "address", "famsize", "pstatus", "medu", "fedu", "mjob", "fjob", "reason",
  "nursery", "internet"), all=FALSE, suffixes = c('mat', 'por'))
```

Visualización de datos

El conjunto de datos de que disponemos no es el mejor para realizar modelos, y/o predicciones. Así que el informe no irá más alla de la visualización de los datos.

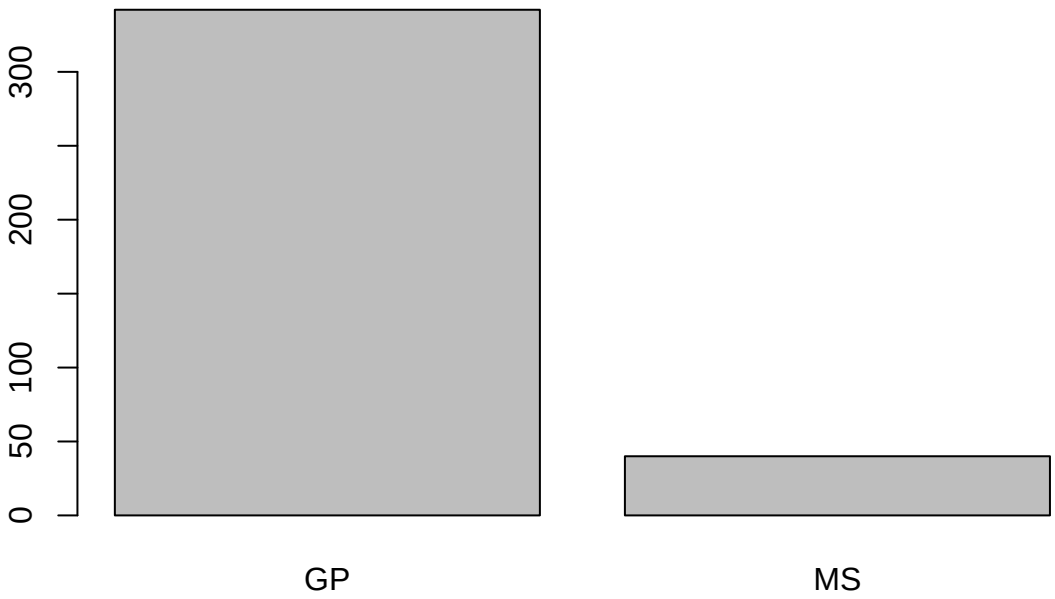
El objetivo de visualizar los datos es el poder hacer más rápida la comprensión de los datos que estamos exponiendo. De manera que no sea necesario entender de "" y/o "programación". Por ejemplo, antes hemos comparado los valores de los alumnos que asisten a cada escuela. Los alumnos están en las dos clases se distribuyen en las escuelas de la siguiente manera:

```
table(dt.students.mat$school)
```

```
##
##  GP  MS
## 349  46
```

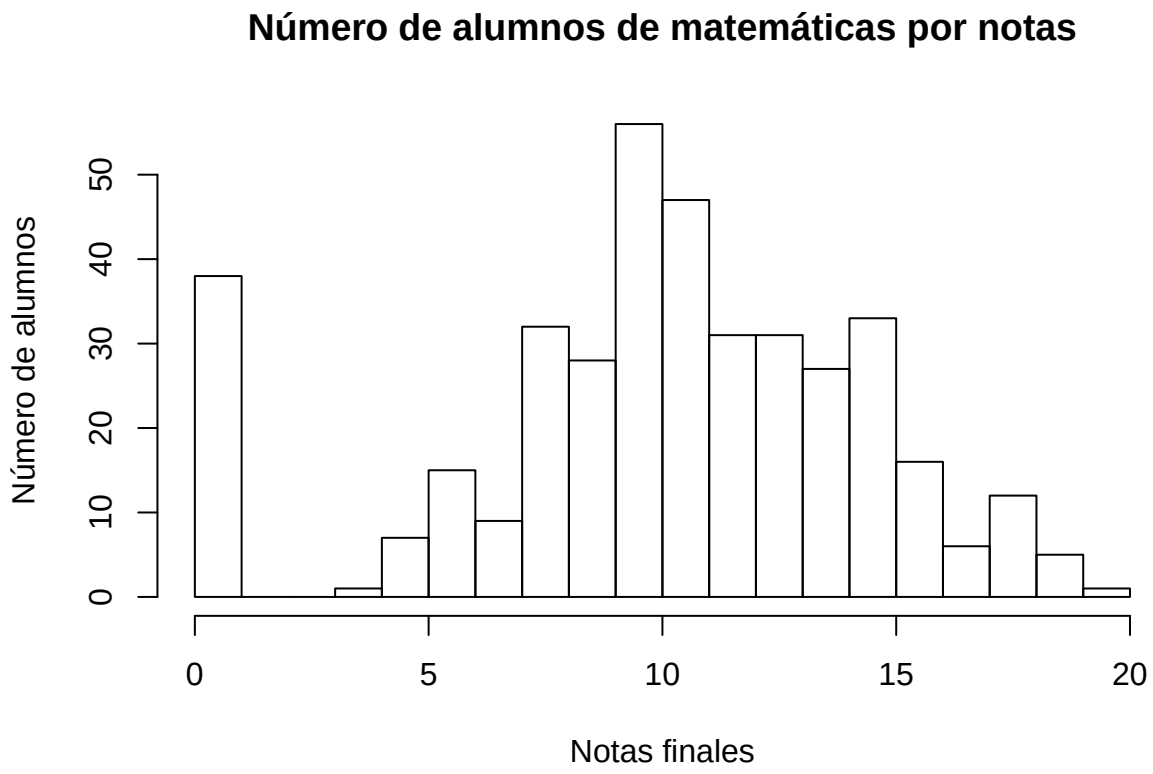
Pero si lo vemos representado entendemos la idea inmediatatamente:

```
plot(dt.students.mp$school)
```

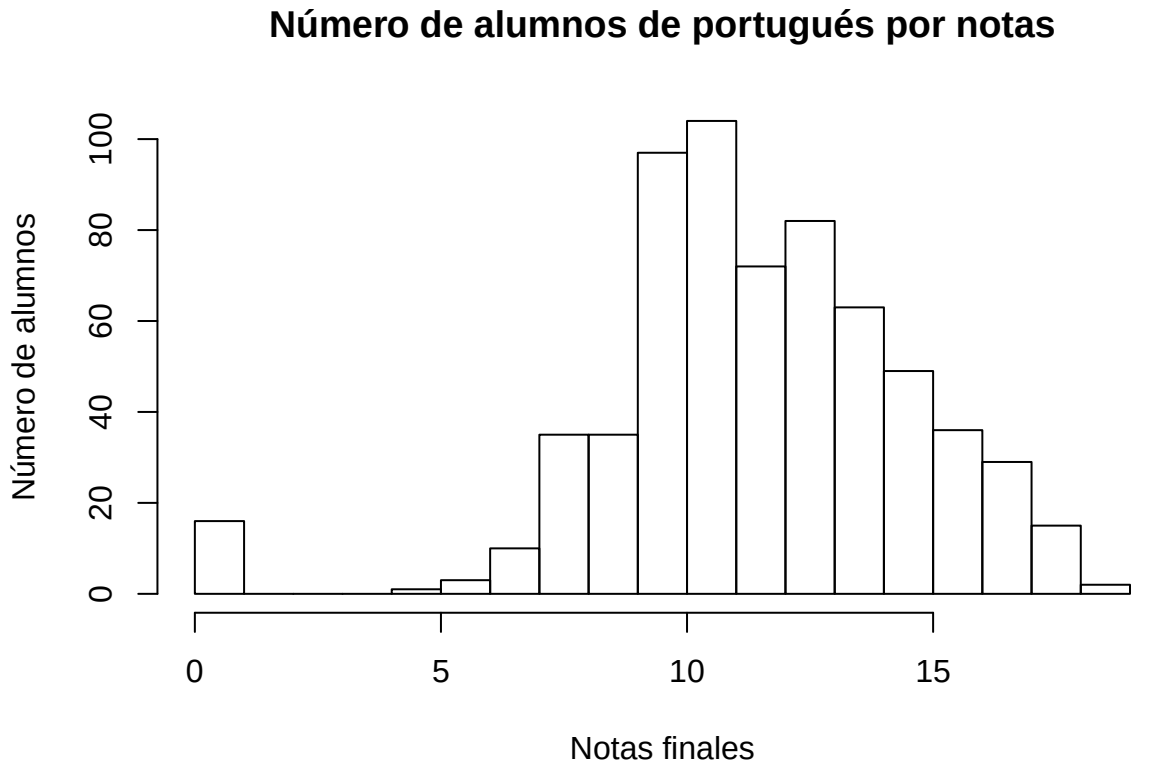


Esta es la manera más básica de representar los datos. Pero hay otras formas más atractivas de hacerlo.

```
hist(x = dt.students.mat$g3, breaks = 20, xlab = "Notas finales", ylab = "Número de alumnos",
     main = "Número de alumnos de matemáticas por notas")
```



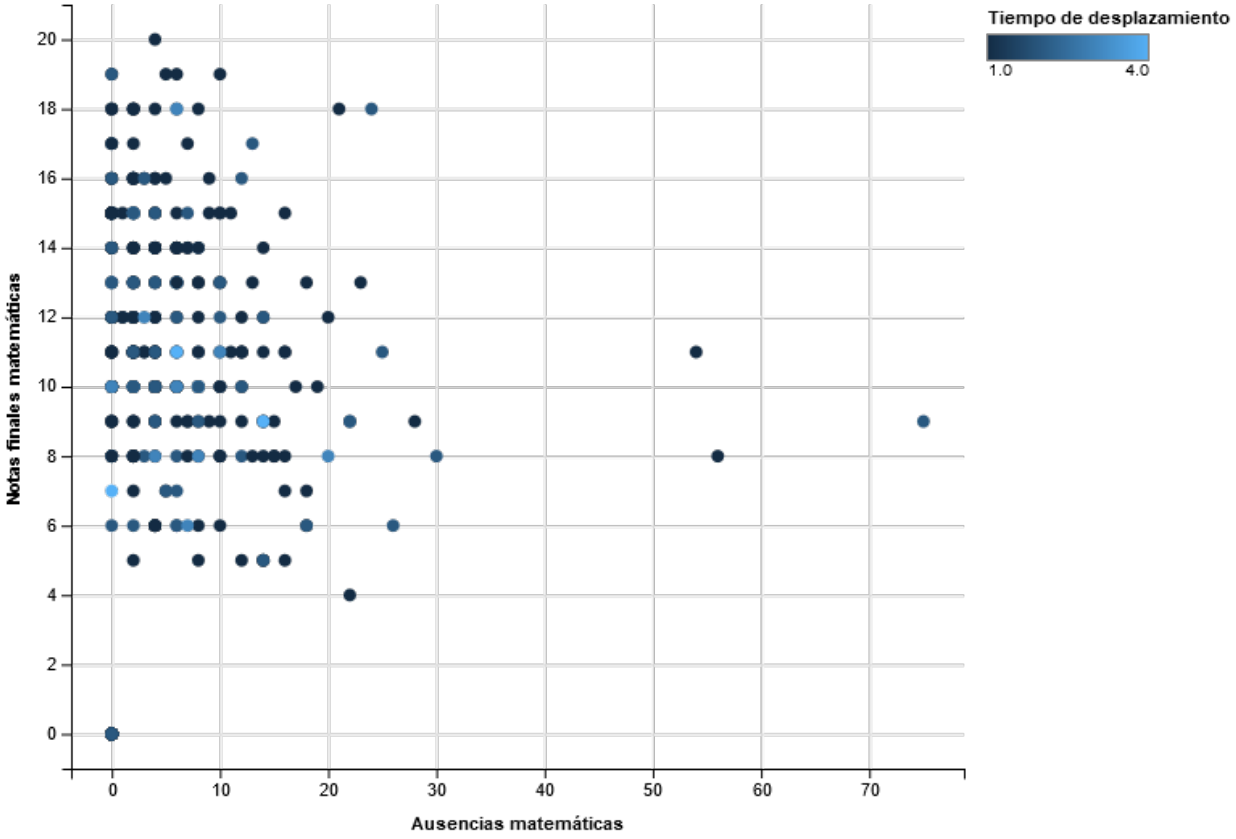

```
hist(x = dt.students.por$g3, breaks = 20, xlab = "Notas finales", ylab = "Número de alumnos",
     main = "Número de alumnos de portugués por notas")
```



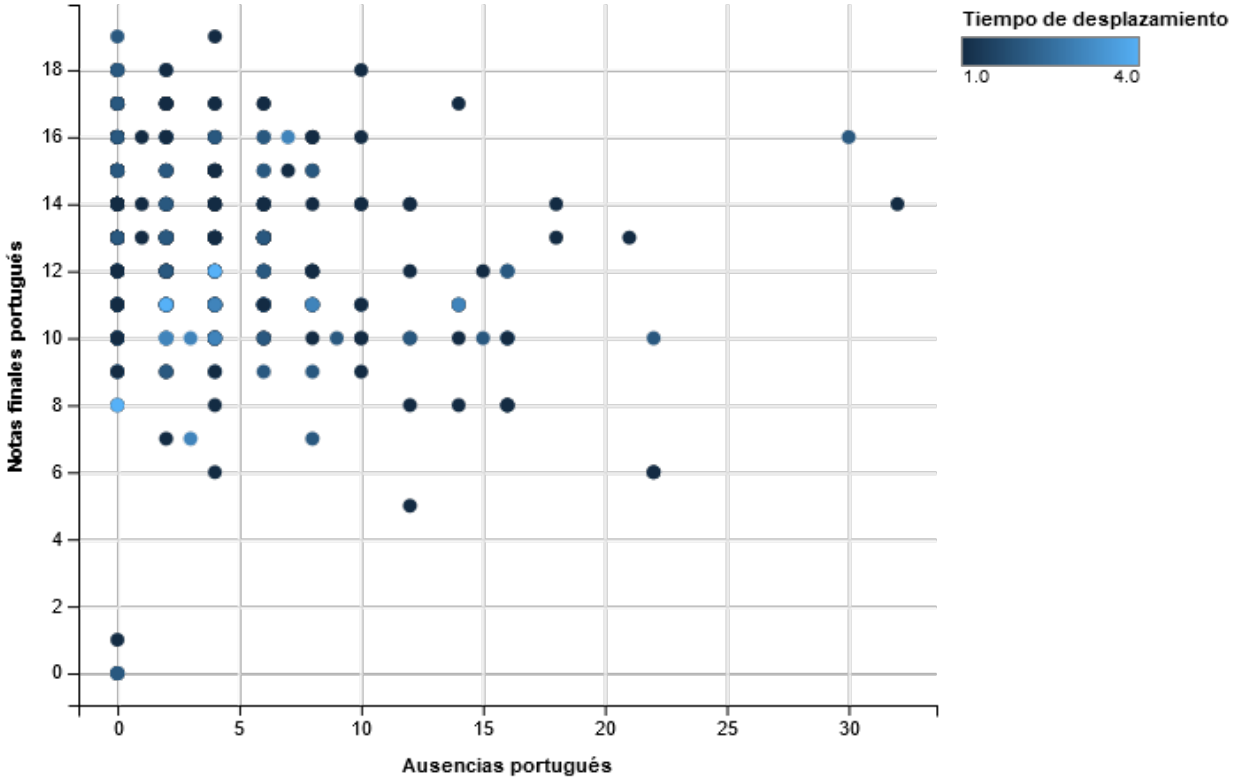
Hay muchas maneras de representar los datos en R. Incluso puedes crear fácilmente una forma de hacerlo tu

mismo. Personalmente, la que más me llama la atención es la librería `ggvis` que esta, en parte, basada en `ggplot2`. Veamos algunos ejemplos:

```
dt.students.mp %>%
  ggvis(x = ~absencesmat, y = ~g3mat, fill = ~traveltimemat) %>%
  layer_points() %>%
  add_axis("x", title = "Ausencias matemáticas") %>%
  add_axis("y", title = "Notas finales matemáticas") %>%
  add_legend("fill", title = "Tiempo de desplazamiento")
```



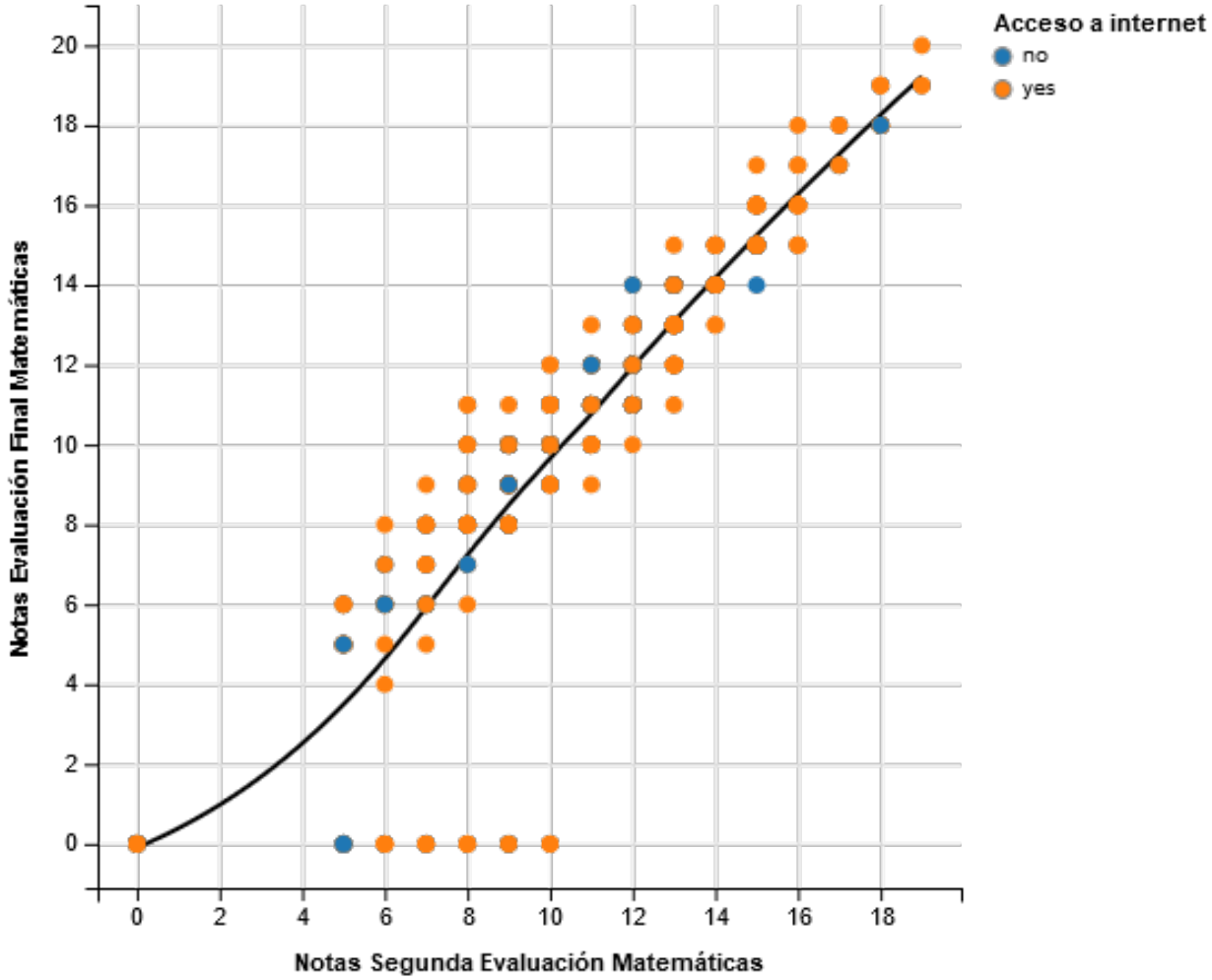

```
dt.students.mp %>%
  ggvis(x = ~absencespor, y = ~g3por, fill = ~traveltimepor) %>%
  layer_points() %>%
  add_axis("x", title = "Ausencias portugués") %>%
  add_axis("y", title = "Notas finales portugués") %>%
  add_legend("fill", title = "Tiempo de desplazamiento")
```



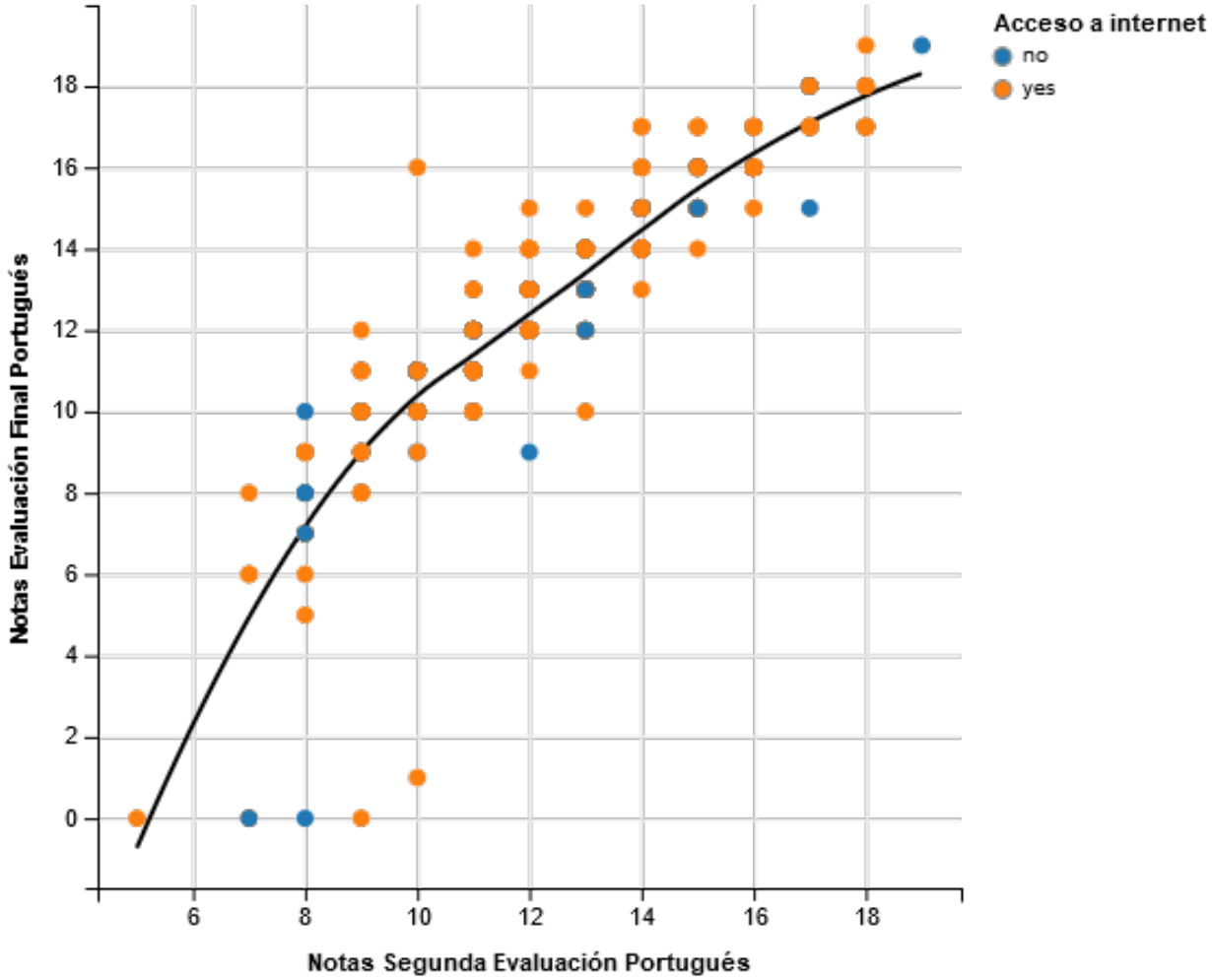
En los graficos anteriores observamos que las ausencias de un alumno en una clase, y sus notas finales, no se relacionan. Además de ello, estaba representado el grado del tiempo de desplazamiento hasta el centro docente, desde casa. Al igual que las otras variables, estaba distribuido sin seguir ningún patrón reconocible, así que tampoco se reaciona mucho.

Finalmente, varias capas:

```
dt.students.mp %>%
  ggvis(x = ~g2mat, y = ~g3mat, fill = ~internet) %>%
  layer_smooths() %>%
  layer_points() %>%
  add_axis("x", title = "Notas Segunda Evaluación Matemáticas" ) %>%
  add_axis("y", title = "Notas Evaluación Final Matemáticas") %>%
  add_legend("fill", title = "Acceso a internet")
```




```
dt.students.mp %>%
  ggvis(x = ~g2por, y = ~g3por, fill = ~internet) %>%
  layer_smooths() %>%
  layer_points() %>%
  add_axis("x", title = "Notas Segunda Evaluación Portugués" ) %>%
  add_axis("y", title = "Notas Evaluación Final Portugués") %>%
  add_legend("fill", title = "Acceso a internet")
```



Esta librería ofrece varios recursos interactivos, de manera que la información es más atractiva. Además de esta, hay bastantes mas librerías, cada una con características únicas que pueden servir para representar debidamento la información que manejas. Además puedes utilizar el paquete `shiny` de RStudio, el cual tiene una gran cantidad de buenas herramientas para crear aplicaciones web interactivas. Pero todo eso (*quizá*) venga en una futura actualización de este documento.