Introducción a la Ciencia de Datos

Guía de trabajos prácticos N°6

Limpieza y adecuación de datos

El objetivo de esta guía es aprender a utilizar herramientas para limpiar y adecuar nuestro *data set* para realizar un correcto análisis. Vamos a ver algunos ejemplos sobre datos que ya conocen pero este proceso deben realizarlo a conciencia sobre los datos que vayan a utilizar para su trabajo final.

Parte 1. Ver problemas de Importación

1. Como siempre, abran RStudio y comiencen un nuevo script (Ctrl+Shift+N o en el menú de arriba a la izquierda) sobre el cual van a trabajar.
2. Carguen tidyverse y el *data set* de arbolado de la comuna 14.

```
library(tidyverse)
df_arbol <- read_csv('arbolado_comuna14.csv')
```

3. Presten atención al mensaje de **warning** que aparece, ¿Pueden interpretar que está diciendo?.
4. Pueden utilizar `problems(df_arbol)` para obtener un detalle de los problemas de importación, ejecútenlo y traten de analizar el resultado.

Lo que reporta esta función son los problemas que tuvo al importar el *.csv*: la fila y la columna en la que encontró el problema, lo que esperaba encontrar y lo que encontró. En particular dice que en la columna 6 de las filas 19337 y 22346 esperaba un *double* y encontró otra cosa.

5. **Por defecto R no importó esos datos**, supongamos que en esas celdas el número está bien y la letra fue un error, editen el .csv para corregirlo y vuelvan a importarlo. A veces, dependiendo el programa que usen para esto, van a tener que probar diferentes opciones de exportación cuando vuelvan a grabar los datos, si usan un programa que edite texto plano no deberian tener problema, pero estén atentos cuando vuelvan a importar el archivo.
6. Usen `glimpse(df_arbol)` y verifiquen que los tipos de datos fueron correctos en la importación (o sea, que el tipo de dato de la columna se corresponda con los datos importados).
7. No parece haber problemas de tipos, si hubiese alguno podemos cambiar el tipo usando `as.integer(...), as.factor(...), as.numeric(...), as.character(...),`

`as.logical(...)`. Por ejemplo, la comuna está cargada como *double*, no está mal pero si quisiéramos transformarla en un entero podríamos hacer:

```
df_arbol <- df_arbol %>%
            mutate(comuna = as.integer(comuna))
```

Háganlo y verifiquen el resultado con glimpse.

NOTA: En este caso particular, donde son pocos los datos que presentan problemas durante la importación, se podría haber decidido directamente ignorar estos errores y continuar. La decisión dependerá del *data set* con el que estén trabajando, de la cantidad de datos de los que dispongan y del problema en particular que están buscando resolver.

Parte 2. Datos repetidos

Veamos ahora el tema de los datos repetidos. Para esto vamos a usar el *data set* de seguros, cárguenlo con:

```
df_seguros <- read_csv('insurance.csv')
```

1. Si queremos buscar las filas repetidas podemos hacer un summarise y usar `n_distinc(...)`. Esto simplemente nos dirá cuántas filas distintas hay.

Ejecuten:

```
df_seguros %>%
      summarise(distintos=n_distinct(df_seguros), total=n())
```

2. En este caso particular tenemos una fila duplicada, pero en realidad no sabemos con certeza si son datos repetidos o simplemente son dos personas que al tener las mismas características están pagando lo mismo. Definir si un dato está duplicado o no depende del *data set*, si cada fila tuviese alguna forma de identificar unívocamente a cada individuo (por ejemplo si tuviese algún DNI) estaríamos seguros. En este caso pueden elegir dejarlo o eliminarlo (siempre documentando la decisión). Si quieren eliminar los datos repetidos pueden directamente usar el comando `unique` en el *data frame*, esto se va a quedar solo con la primera aparición y va a descartar el resto.

```
df_seguros <- unique(df_seguros)
```

3. Un uso interesante del *unique* es para encontrar valores mal ingresados en atributos que toman pocos valores como por ejemplo, en el dataset de árboles, el estado de la plantera y el nivel de la plantera. Veamos todos los valores que pueden tomar:

```
unique(df_arbol$estado_plantera)
```

Hagan lo mismo sobre el atributo *nivel_plantera.* ¿Qué pasó?.

4. No solo apareció un **NA**, sino que hay capitalizaciones incorrectas. Si vamos a necesitar hacer un análisis donde tengamos en cuenta esta variable, deberíamos arreglarlo.

En este caso es muy simple, podríamos pasar todo a minúsculas con el comando `str_to_lower(...)`

```
df_arbol <- df_arbol %>%
   mutate(nivel_plantera = str_to_lower(nivel_plantera))
```

Otra solución un poco más sofisticada (que nos puede ser muy útil después, anoten) es reemplazar cada aparición de, "A Nivel" por "A nivel". Para eso podemos usar un *replace* dentro de un *mutate*.

```
df_arbol <- df_arbol %>%
   mutate(nivel_plantera =
      replace(nivel_plantera,
              nivel_plantera=="A Nivel",
             "A nivel"))
```

Utilicen alguna de las dos opciones para arreglar la variable *nivel_plantera* y repitan el mismo proceso sobre todos los atributos que tengan el mismo problema.

Parte 3. Irregularidades en los datos y datos faltantes

Algo que hacemos habitualmente es hacer gráficos de dispersión o histogramas sobre variables numéricas. También solemos ver ciertas métricas estadísticas e incluso utilizarlas para reemplazar datos faltantes. Analicemos un poco algunos atributos numéricos y veamos cuantos valores **NA** tenemos en la altura o el diámetro de los árboles de la comuna 14. Para esto pueden usar la función `is.na(...)`.

1. Encuentren cuántos árboles tienen valores faltantes en la altura o el diámetro, pueden usar:

```
df_arbol %>% summarise(cant_arboles = n(),
             na_altura = sum(is.na(altura_arbol)),
             na_diametro = sum(is.na(diametro_altura_pecho))
             )
```

2. Efectivamente hay valores faltantes, podemos dejarlos así (sabemos que cuando hacemos cuentas o estadísticas sobre estos datos se ignoran) o podemos reemplazarlos por algún valor, lo que suele ser útil cuando tenemos pocos datos y no queremos perder información. A modo de ejemplo reemplacemos los valores del diámetro por la media de todos los diámetros del *data set*. Vamos a crear otro data frame para no pisar el original:

```
df_arbol_2 <- df_arbol %>%
   mutate(diametro_altura_pecho
       = replace(diametro_altura_pecho,
         is.na(diametro_altura_pecho),
         median(diametro_altura_pecho, na.rm = TRUE)))
```

3. Esto parece tener sentido, pero ¿por qué mejor no analizamos los valores que toma el

diámetro y la altura? Hagan un histograma simple de cada uno de estos dos atributos utilizando el *data frame* original. ¿Notan algo raro? (TIP: ¿qué está pasando en el primer bin?)

4. El diámetro está en centímetros y la altura en metros, o sea que no tiene sentido que haya árboles de diámetro cero, son **valores erróneos**. Cuando se calculó la media sobre todos los árboles para reemplazar los valores **NA** se usaron todos estos ceros para la cuenta. Por lo que la media usada no representa la realidad de los árboles. **Primero deberíamos haber analizado las distribuciones de estos valores y DESPUÉS ver qué hacemos con los NA.**

Lo que podemos hacer es buscar los valores que sean cero y reemplazarlos por **NA,** así cuando calculemos la media ignora estos valores y es una media un poco más real. Para esto podemos usar el *replace*

```
df_arbol <- df_arbol %>%
   mutate(diametro_altura_pecho
        = replace(diametro_altura_pecho,
                  diametro_altura_pecho==0,
                  NA))
```

5. Verifiquen que efectivamente ahora esos valores fueron reemplazados por **NA.**
6. Ahora sí, reemplacen por el valor medio en el data frame original
7. Utilizando nuestro conocimiento del dominio del problema, podemos suponer que no todas las especies tienen la misma media de diámetro, si hacemos esto estamos asignando la media de TODOS los árboles de la comuna 14. Una mejor opción podría ser reemplazar los valores faltantes por la media de cada especie. Incluso podría hacerse por calle y/o por Comuna. Elijan alguna de las opciones y háganlo.
8. Realicen el mismo análisis sobre el atributo altura.
9. Cuando se sientan satisfechos con todos los cambios y piensen que tienen el dataset listo para trabajar, hagan una gráfica de latitud vs. longitud para los árboles. Piensen qué esperan obtener antes de hacerlo. Cuando vean el resultado, discutan si es lo que esperaban y si necesitarían hacer alguna limpieza más con el data set antes de arrancar. De ser así, háganlo.

   TIP profesional: Pueden usar el código en *mapa_caba.R* para darle una base al gráfico y, si quieren, repetir el procedimiento con el *data set* de arbolado público lineal completo, eligiendo y trabajando con otra comuna.

Parte 4. Documentación de cambios y decisiones

Durante el proceso de limpieza y adecuación de los datos es necesario informar las decisiones tomadas a la hora de trabajar con estos datos. No se espera algo muy formal, pero es muy importante que estén reportados los cambios que se hicieron ya que se vuelven hipótesis de nuestro análisis.

Por ejemplo, esto es lo que se hizo en el dataset de árboles:

* En la importación se ignoraron 733 filas con número de manzana.

* El atributo `nivel_plantera` tenía datos mal capitalizados, se corrigieron. No se hizo nada con datos NA.

* El atributo `diametro_altura_pecho` tenía valores cero, que son incorrectos en el contexto del problema. Se cambiaron a NA.

* Los valores NA de `diametro_altura_pecho` y `altura_arbol` se reemplazaron con los valores medios por especie.

A modo de práctica, revisen esta lista con los ejercicios realizados y completen cualquier otro cambio que hayan realizado sobre los datos.

Bibliografía obligatoria

Libro de Wickham

Importación de datos: https://es.r4ds.hadley.nz/09-wrangle.html

Datos ordenados: https://es.r4ds.hadley.nz/12-tidy.html