Introducción a la Ciencia de Datos

Guía de trabajos prácticos N°4

Análisis dataset árboles (comuna 14)

En esta guía vamos a trabajar con el dataset de arbolado lineal de la Ciudad de Buenos Aires, filtrado para conservar sólo los árboles de la comuna 14.

Vamos a dar los primeros pasos en el terreno de la **estadística descriptiva**. Para eso, vamos a introducir el concepto de **métrica de resumen** (más formalmente un **estadístico de resumen** / *summary statistics*); vamos a ver la utilidad de calcular estas métricas para sacar información sobre los datos que de otra forma sería difícil de obtener. Y, finalmente, presentaremos un nuevo tipo de gráfico extremadamente útil para resumir la información: el diagrama de caja (*box plot*)

Parte 1. Lectura de los datos

1. Abran RStudio desde Ubuntu. Pueden hacerlo desde el menú de aplicaciones (abajo a la derecha) o tocando la tecla "Meta" (Windows) y tipeando "Rstudio". Antes de que terminen va a aparecer el ícono de la aplicación.
2. Comiencen un nuevo script (**Ctrl+Shift+N**) o en el menú de arriba a la izquierda) sobre el cual van a trabajar. Recuerden que, para ejecutar líneas individuales, hay que seleccionarlas o pararse sobre ellas y usar **Ctrl+Enter**.
3. Carguen la biblioteca 'tidyverse' con el comando

   ```
   library(tidyverse)
   ```

   Si siguieron las guías anteriores todo esto deberia haber sido bastante automático.

4. Carguen los datos que están en el archivo `arbolado_comuna14.csv`. Para eso, pueden revisar la Guía 1 para recordar el uso de la función `read_csv`. ¡Ojo, no confundir con el comando `read.csv`! y recuerden que es más facil para trabajar con archivos guardar el script en el mismo directorio e indicarle a RStudio el directorio de trabajo (`Session -> Set Working Directory -> To Source File Location`).

```
df <- read_csv("ruta_al_archivo")
```

5. Respondan a las siguientes preguntas

   a. ¿Cuántas filas tiene el dataset?

   b. ¿Cuántas columnas?

   c. ¿Qué hace el comando `length(df)`? ¿Y `nrow(df)`? Ambos son de R base, pero se llevan bien con los formatos de tablas del paquete `tibble`.

Parte 2. Filtrado

6. Usen el comando `filter` para crear una tabla que contenga solo los árboles de estas tres especies:

   - 'Tilia x moltkei',
   - 'Fraxinus excelsior',
   - 'Melia azedarach'

   Para lograr esto, pueden usar la magia del operador `%in%` dentro de `filter` (vean cápsula correspondiente):

```
df %>% filter(nombre_cientifico %in%
c('Tilia x moltkei', 'Fraxinus excelsior', 'Melia azedarach'))
```

   Recuerden asignar el resultado a una nueva variable, porque vamos a usarlo.

   Para entender qué hace este operador, prueben qué hace el comando.

```
df$nombre_cientifico %in%
c('Tilia x moltkei','Fraxinus excelsior', 'Melia azedarach')
```

7. **Respondan** ¿Cuántos árboles de cada especie hay? Vimos cómo hacer esto en la Guía 2.

Parte 3. Visualización

Vamos a querer responder la pregunta "¿Qué especie de árboles (entre las tres elegidas) es la más alta?".

8. Como siempre, empezamos con una visualización, para entender mejor nuestro dataset. Hagan un gráfico de dispersión (*scatter plot*; recuerden el uso de `geom_point`, de la Guía 1).

   > **Nota**
   >
   > Hasta ahora, en las guías solo vimos gráficos de scatter usando dos variables continuas. Esto está bien; el gráfico de dispersión se usa *principalmente* para ver la relación entre dos variables continuas. Pero a veces uno puede permitirse otros usos, como el que hacemos acá.

   Vamos a usar la variable `altura_arbol` en el eje `y`, la variable `nombre_científico` en el eje `x` (¡usada aquí como categórica!). Antes de hacer el gráfico, **piensen en grupo y boceteen en un papel el tipo de gráfico que esperan obtener**.

9. Ahora sí, hagan el gráfico y comparen con sus expectativas. Deténganse a mirar en detalle el gráfico. ¿Cuántos puntos hay? ¿Cómo se compara esto con la respuesta en el punto 7 de arriba? ¿Qué piensan que puede estar pasando?

   > **Pista**
   >
   > Si no se les ocurre nada, usen el argumento `alpha` de `geom_point`. Por ejemplo, usen `alpha=0.2`.

10. Cuando los puntos de un conjunto se acumulan en los mismos lugares, a veces es útil usar la función `geom_jitter()`, que es igual a `geom_point`, pero que desplaza a los puntos individuales tanto vertical como horizontalmente, agregando un pequeño número al azar (ruido) a los valores de cada variable.

    > **Nota**
    >
    > Esto permite muchas veces encontrar patrones que de otra manera se perderían. Parece paradójico que agregando ruido uno pueda encontrar más cosas, ¿no?

    Prueben repetir el gráfico de arriba reemplazando `geom_point` por `geom_jitter`. Prueben los argumentos `width=0` y después `height=0` dentro de `geom_jitter()`.

¿Cuál de las dos opciones es más apropiada en este caso? ¿Por qué? Comparen con el gráfico anterior.

11. ¿Pueden a partir de este gráfico ordenar las especies por altura? ¿Cuáles serían sus argumentos para cada orden? Piensen en grupo y discutan sus ideas con otro grupo.

Parte 4. Resúmenes

Tal vez ya esté claro que, para ordenar las especies por altura, necesitamos reducir / resumir la información de la altura de los árboles de cada especie a un número. Este número servirá para realizar una comparación entre las especies. Por supuesto, el tipo de resumen que elijamos tiene que ser el mismo para las tres especies.

12. Para esta actividad, es natural elegir como métrica de resumen el **promedio** de las alturas de todos los árboles, que se obtiene sumando todos los valores, $\left(\sum_i altura_i\right)$ y dividiendo por la cantidad total de valores (N):

$$<Altura> = \frac{\sum_i altura_i}{N} \quad .$$

> **Nota**
>
> En estadística, este cálculo produce un estadístico que se llama la **media aritmética**.

Calculen este valor para las tres especies.

> **Tip**
>
> Pueden usar la función `group_by`, seguida del comando `summarise` (vean la guía 2 y la cápsula correspondiente si no se acuerdan cómo funciona). Pueden usar la función `sum` o directamente hacer uso de la función `mean`.

> **Advertencia**
>
> ¡Ojo! ¿Qué pasa si no prestamos atención a los valores faltantes? Tanto la función `sum` como `mean` tienen el argumento `na.rm` ("quitar NA") que permite deshacerse de estos valores (para el cálculo) de manera automática (lo vimos en la guía 2).

Salven el resultado del cálculo en un nuevo *data frame* (llamado *df_resumen*).

13. A partir de los valores de altura promedio. ¿Cómo podrían ordenarse por alturas las especies?

Parte 5. Cuantiles

Ahora imaginemos que queremos saber hasta qué altura puede crecer cada especie. Una situación que podría requerir esto es si plantamos un árbol y no queremos que, al crecer, perturbe el tendido eléctrico o tape la vista desde una ventana.

14. Piensen qué métrica podríamos usar para definir hasta qué altura puede llegar cada especie. Una idea intuitiva podría ser tomar la altura del árbol más grande para cada especie. ¿Qué problemas o limitaciones les parece que puede tener esta métrica?

> **Tip**
>
> Usen nuevamente `group_by` seguido de `mutate` para calcular el máximo de cada especie y agregarlo al *data frame* `df` como una variable de nombre `altura_max`.
>
> ```
> df <- df %>%
>   group_by(nombre_cientifico) %>%
>   mutate(altura_max = max(altura_arbol, na.rm=T))
> ```

15. ¿Qué pasaría si alguien viniera y les comentara que el valor extremo de las alturas de *Fraxinus excelsior* es un error en la entrada de datos (alguien le puso un cero de más, por ejemplo)? ¿Cómo cambiaría el resultado del ejercicio anterior?

> **Extra**
>
> Pueden probarlo, usando este código para reemplazar el valor máximo por el valor "corregido" , y repitiendo los pasos anteriores en esta nueva tabla dff obtenida a partir del `group_by` y `summarise` de arriba..

```
# Obtenemos el valor de altura máxima de Fraxinus

max_fraxinus <- df[df$nombre_cientifico=='Fraxinus excelsior',]$altura_max

# Creamos una nuevo data frame donde le sacamos un cero a ese valor)
```

```
nuevo_df <- mutate(df,altura_arbol = if_else(altura_arbol == max_fraxinus &
                                             nombre_cientifico == 'Fraxinus excelsior',
                                             max_fraxinus/10,
                                             altura_arbol) )
```

Usen también `group_by` con `summarise` para generar un nuevo dataframe (¿cuántas filas tendrá?) que permita ver fácilmente la altura máxima para cada especie. Comparen los valores obtenidos con las alturas promedias calculadas arriba.

16. Al quitar un solo punto, el resultado del análisis puede cambiar drásticamente si usamos el máximo para resumir la información. Una forma más robusta (es decir, que depende menos de puntos individuales) es utilizar los cuantiles.

    > **!** Cuantiles
    >
    > El cuantil q de una serie de valores (en este caso, de alturas) es el valor tal que el q * 100 % de los valores son menores que ese valor. Por ejemplo, el cuantil 0,95 es el valor de altura que deja solo el 5% de los árboles por arriba. También se llama a este valor el percentil q*100 (es decir, en este caso, el percentil 95).

    Existe una función en R que calcula esto: se llama `quantile`, del paquete `stats`. Prueben calcular el cuantil 0,95 de todos los árboles de la comuna 14:

    ```
    quantile(df$altura_arbol, probs = 0.95, na.rm=TRUE)
    ```

    (noten de nuevo la aparición de `na.rm=TRUE`).

    Una cosa práctica de esta función es que uno le puede pasar una lista de valores y calcular todos los cuantiles correspondientes con el mismo comando. Por ejemplo, si quisiéramos los percentiles 5 y 95.

    ```
    quantile(df$altura_arbol, probs = c(0.05, 0.95), na.rm=TRUE)
    ```

    El percentil 50 (cuantil 0,5), el valor que divide a la lista en dos mitades, se llama **mediana** y es un sustituto robusto al promedio, que es mucho menos sensible a valores extremos o erróneos.

    ```
    quantile(df$altura_arbol, probs = c(0.05, 0.5, 0.95), na.rm=TRUE)
    ```

17. Usen nuevamente `group_by` y `summarise` para calcular el máximo de cada especie y los cuantiles 0.05 y 0.95 y agregar todo en un dataframe. Prueben usar directamente quantile en un `summarise` ¿Qué obtienen como resultado en cada caso?

Miren el formato de la tabla. ¿Se cumple la condición de que cada línea corresponde a una unidad? Este formato no es "tidy", pero esto vamos a aprender a arreglarlo más adelante.

18. Otros percentiles con nombres propios son el 25 y el 75, que se llaman **primer cuartil** y **tercer cuartil**, respectivamente.

    La distancia entre ellos se conoce como la **distancia intercuartil (IQR**, por las siglas en inglés), y es una manera robusta de informar la dispersión de las mediciones. Más de esto en la próxima guía. Por lo pronto, pueden calcular estos percentiles como lo venimos haciendo, y la IQR con la función `stats::IQR`.

    Calculen estos percentiles para cada especie.

19. La combinación de la **mediana** o el promedio (la **media**) y el **primer y tercer cuartil** se usan muchísimo para describir de manera sucinta una serie de datos. Son tan comunes que R tienen varias funciones para calcular esto. Prueben `summary()` o `fivenum()` en alguna columna del dataset: `summary(df$altura_arbol)`, por ejemplo. Por supuesto, también se puede usar `summarise` en combinación con `fivenum`. Probar el código siguiente. Analizar el resultado.

```
df %>% group_by(nombre_cientifico) %>%
  summarise(name = c('min', '1st', 'median', '3rd', 'max'), value =
                                                    fivenum(altura_arbol))
```

    Si bien la información que queremos está ahí, el formato no es *tidy*. En este caso, podemos acomodarlo fácilmente, usando el paquete tidyr (incluido en tidyverse). Agregar al código de arriba el siguiente paso:

```
pivot_wider(names_from=name, values_from=value).
```

    Por supuesto, también pueden agregar cada percentil individualmente con `summarise`:

```
df %>% group_by(nombre_cientifico) %>%
  summarise(min=min(altura_arbol, na.rm=T),
            primer=quantile(altura_arbol, 0.25, na.rm=T),
            median=median(altura_arbol, na.rm=T),
            tercer=quantile(altura_arbol, 0.75, na.rm=T),
            max=max(altura_arbol, na.rm=T))
```

Parte 6. Diagrama de caja (*boxplots*), su nuevo mejor amigo.

El diagrama de cajas permite visualizar muchas de estas métricas de resumen de una manera conveniente y muy frecuente.

20. Prueben agregar un `geom_boxplot` al gráfico de *jitter* de arriba. Usen los mismos parámetros para el `aes` (de hecho, pueden pasarlo al argumento de `ggplot`). Agreguen transparecnia a las cajas con el argumento `alpha` (p.ej. `alpha=0.4`).

    Discutan el resultado en comparación con el gráfico anterior. ¿Entienden qué representa cada línea de la caja?

    > Tip
    >
    > Para verificar las intuiciones que puedan tener, usen `geom_hline` para agregar líneas horizontales en el gráfico. La altura a la que se ubica la recta horizontal está definida por el parámetro `yintercept`.

La otra parte del diagrama de cajas son los bigotes (*whiskers*). El bigote superior se extiende desde el tercer cuartil hasta el último punto, pero no más allá de 1,5 veces el IQR. Algo equivalente pasa para el bigote inferior. Va desde el primer cuartil hasta el punto más bajo, siempre y cuando no se extienda más allá de 1,5 IQR desde la caja. Los puntos que quedan por fuera de los bigotes se denominan "outliers" y se grafican individualmente.

21. ¿Cuántos outliers hay para cada especie? ¿Se animan a escribir un código de R que cuente los outliers de cada especie?

Parte 9. *Violinplots*, el primo de su nuevo mejor amigo.

Una alternativa interesante para comparar distribuciones de varios grupos, es el primo del gráfico de cajas: el gráfico de violines (`geom_violin`). La diferencia esencial con el gráfico de caja es que el de violines realiza estimaciones de densidad para cada grupo, como si fuera un geom_density para cada grupo. El gráfico de densidad aparece duplicado, de forma que crea algo parecido a un violín.

22. Prueben crear un gráfico de violines para comparar las alturas de las diferentes especies.

23. Una cosa que unx extraña inmediatamente, es la presencia de marcas que señalen los cuantiles más importantes. Pero `geom_violin` tiene un argumento `draw_quantiles`, que puede usarse para esto. Prueben con `draw_quantiles = c(0.25, 0.5, 0.75)`. Por supuesto, también pueden mapear otra variable a los colores de las líneas o de relleno de los violines. Pruébenlo.

Parte 8. Poniendo todo en práctica.

Escriban un reporte de menos de una carilla que describa de la manera más sucinta a los árboles de la comuna 14. Pueden incluir gráficos como con los que hemos trabajado en la Clase 1, 2, 3 y 4. Luego de cada uno de ellos, incluyan una breve descripción de la información que pueden extraer del mismo.

Resumen de lo visto en esta guia

- Uso del pipe `%>%`.
- Describir una variable numérica con estadística, valores medios, quartiles, dispersión. Mostrar todo esto usando diagramas de cajas.
- Funciones:`filter`,`geom_jitter`,`summarise`, `group_by`,`mutate`, `quantile`, `fivenum`, `geom_boxplot` y `geom_violin`.

Bibliografía obligatoria

Relación entre los `geom` y los `stat`

Wickham:

- 1ra ed. Sección 3.7
- 2da ed. Sección 9.5

Gráficos de cajas

Wickham:

- 1ra ed. Sección 7.5
- 2da ed. Sección 10.5

Discusiones sobre las limitaciones del boxplot: https://www.data-to-viz.com/caveat/boxplot.html

Bibliografía para profundizar

Más detalles técnicos y muchos tips: https://r-graph-gallery.com/boxplot.html

Uso de `boxplot` con R base: https://bookdown.org/jboscomendoza/r-principiantes4/diagramas-de-caja.html