Introducción a la Ciencia de Datos - 2C 2022

Guía de Trabajos Prácticos Nº 3

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 métricas de resumen para obtener información sobre los datos que de otra forma sería difícil de obtener. Finalmente, presentaremos un nuevo tipo de gráfico, extremadamente útil para resumir la información y muy utilizado: 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 (a menos que sea una sola) y usar **Ctrl+Enter**.

3. Carguen la biblioteca 'tidyverse' con el comando

```
library(tidyverse)
```

Hasta acá, debería ser casi automático, si siguieron las guías anteriores.

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"!

```
df <- read_csv(camino_al_archivo)
```

5. Respondan a las siguientes preguntas (en el formulario)

   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 el operador, prueben qué hace el comando.

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

7. **Respondan** (en el formulario) ¿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). Hasta ahora, solo vimos gráficos de scatter usando dos variables continuas. Pero ahora 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 boceten 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 ruido a cada variable. Esto permite muchas veces encontrar patrones que de otra manera se perderían (Nota: 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()`. 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 en 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 (<altura>), que se obtiene sumando todos los valores ($\Sigma_i\ altura_i$) y dividiendo por la cantidad total de valores (N).

$$< Altura >= \frac{\Sigma_i\ altura_i}{N}$$

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

Calculen este valor para las tres especies. 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`. ¡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). Usen `na.rm=TRUE`. Salven el resultado del cálculo en un nuevo *data frame* (llamado df_resumen).

13. Tracen tres líneas horizontales en el gráfico de puntos de arriba, con los valores medios de las alturas para cada especie. Usen la función `geom_hline` para hacer las líneas. El argumento clave es `yintercept`, pero también van a querer que el color o el tipo de línea (`linetype`) esté mapeado a la especie, para poder distinguirlos en el gráfico:

```
… + geom_hline(data=df_resumen, aes(yintercept=media,
color=nombre_cientifico)
```

14. A partir de los valores de altura promedio. ¿Cómo podrían ordenarse por alturas las especies? Respondan en el formulario.

Parte 5. Cuantiles.

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

15. Piensen en 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? Usen nuevamente `group_by` y `summarise` para calcular el máximo de cada especie y agregarlo a un *data frame* con el nombre `altura_max`. Comparen los resultados obtenidos con los valores de las respectivas medias.

16. ¿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?

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 <- dff[dff$nombre_cientifico=='Fraxinus
excelsior',]$altura_max
```

```
# Creamos una nueva tabla donde le sacamos un cero a ese valor)
> nueva_tabla <- mutate(df,
                      altura_arbol = if_else(altura_arbol == max_fraxinus &
                                  nombre_cientifico == 'Fraxinus excelsior',
                                  max_fraxinus/10, altura_arbol)
                             )
```

17. Como vemos, entonces, 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. 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 probabilidades 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.

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

Contesten en el formulario qué información dan estos números.

18. 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.

19. 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 y completen el formulario.

20. 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:

```
dff %>% 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))
```

Usando este resultado (¡guárdenlo en una variable!) agreguen líneas horizontales al gráfico del punto 14 en las ubicaciones del primer y tercer cuartil. Usen líneas rojas para la mediana, y negras para los cuartiles. Usen el atributo `linetype` para mapear la especie. Partan del código del punto 13.

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

21. El gráfico obtenido en el punto 20 tiene mucha información pero es complicado de leer. Una versión mucho más ordenada es lo que lleva a los diagramas de caja o *boxplots*. Prueben reemplazar los tres `geom_line` del punto anterior simplemente por `geom_boxplot()`. Discutan el resultado en comparación con el gráfico anterior. ¿Entienden qué representa cada línea de la caja?

22. 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. ¿Cuántos outliers hay para cada especie? **Extra**: ¿se animan a escribir un código de R que cuente los outliers de cada especie?

Parte 7. 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 y 3. Luego de cada uno de ellos, incluyan una breve descripción de la información que pueden extraer del mismo.

Para trabajar en casa

Si aún no lo hiciste, contestá el Cuestionario de la Clase 3 antes de la clase virtual de esta semana.

Análisis dataset vuelos (flights)

Vamos a usar la tabla `vuelos` sobre la cual charlamos en la exposición de la primera clase. Les recordamos que este dataset contiene los vuelos que salieron de la ciudad de Nueva York en el año 2013, información procedente de la Oficina de Estadísticas de Transporte de Estados Unidos. Vamos a estudiar los retrasos de tres aerolíneas y usar las métricas de resumen que discutimos esta semana para compararlas.

Parte 1. Check-in

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. Por las dudas de que no esté instalado el paquete que tiene el set de vuelos, en la consola ingresen

   ```
   > install.packages('nycflights13')
   ```

   Si el paquete ya está instalado, esto no va a hacer nada.

3. Abran 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 (a menos que sea una sola) y usar **Ctrl+Enter**.

4. En las primeras líneas, escriban los comandos que cargan los paquetes que vamos a usar y ejecuten:

   ```
   library(tidyverse)
   library(nycflights13)
   ```

   Ahora deberían tener definida la variable flights, que contiene la tabla de todos los vuelos. Prueben meter en la consola

   `> flights` o bien `> view(flights)`

   ¿Cuántas filas tiene el dataset? ¿Cuántas columnas?
   ¿Qué hace el comando `length(flights)`? ¿Y `nrow(flights)`?

5. Identifiquen el atributo que indica la aerolínea del vuelo. (Tip: con el comando `colnames()` se imprimen los *headers* del *dataframe*) ¿Entienden qué es cada columna? Si no, pueden usar `?flights` en la consola para leer documentación.

Parte 2. Abordando

6. Averigüen cuáles son las aerolíneas que operaron vuelos desde Nueva York en 2013. Para eso, pueden usar el comando `unique()` aplicado a una sola columna de la tabla.

   Para elegir una columna, hay varias opciones:
   a. `flights['nombre_de_la_columna']`
   b. `select(flights, nombre_de_la_columna)`
   c. `flights$nombre_de_la_columna`

   Aplicar el comando `unique()` a estas opciones da resultados diferentes. ¿Cuál de estas opciones les devuelve una tabla? ¿Cuántas aerolíneas diferentes hay en el dataset?

7. Agrupen `flights` por aerolínea y cuenten cuántos vuelos realizó cada una en el año. Filtren la tabla original para quedarse solo con las aerolíneas que operaron más de 1000 vuelos en el año. Para esto, tienen que hacer una combinación de `group_by`, `filter` y `ungroup`.

   ```
   > flights %>% group_by(carrier) %>% filter(n() >= 1000) %>% ungroup()
   ```

   ¿Cuántas aerolíneas quedan seleccionadas? ¿Cuáles son?

8. Filtren la tabla resultante para quedarse solo con los vuelos que iban a Los Ángeles (aeropuerto de destino, LAX). ¿Cuántas aerolíneas quedan? ¿Cuántos vuelos realizó cada una?

Parte 3. Despegue

9. Grafiquen los minutos de retraso en las salidas (`dep_delay`) para estas aerolíneas, usando `geom_jitter` y superpongan un boxplot. ¿Pueden usar este gráfico para ordenar a las aerolíneas? Comparen estos gráficos con los que vimos en clase para las alturas de los árboles. ¿Qué diferencias ven? ¿Cómo se comparan los números de puntos fuera de las cajas?

10. Usen **el gráfico** para identificar a la aerolínea que tuvo el máximo retraso en vuelos de Nueva York a Los Ángeles en 2013. Es decir, no queremos que calculen el máximo, sino que identifiquen a la aerolínea a partir del gráfico.

11. Calculen métricas de resumen como hicimos para los árboles. Si tuvieran que usar estos datos para recomendar una aerolínea sobre otra, ¿qué métrica usarían? ¿Qué aerolínea recomendarían para este trayecto en base a este análisis?

Entrega 3

Armen un .pdf que contenga lo siguiente:

1. *scatter plot de los retrasos* + boxplot (similar a lo que se hizo en el laboratorio), con los nombres de los ejes y un título descriptivo.
2. Una breve descripción de no más de 100 palabras de lo que puede verse en el gráfico.
3. Una tabla reportando las métricas de resumen que hayan elegido para comparar las aerolíneas.
4. Un texto de no más de 250 palabras recomendando una aerolínea y justificando la elección en base al gráfico y/o a la tabla.

Suban al campus en PDF en la sección de “Entrega N° 3”.

Nota: En la solapa donde aparecen los gráficos en RStudio van a encontrar una opción que dice “Export” que va a permitirles copiar el gráfico para agregarlo al documento.