Limpieza y Validacion de Datos - Práctica Nº2

Darwin Padilla

08 de junio 2021

Contents

Objetivo

Tratamiento de un dataset, justificar las tareas del dataset propio o extradio desitioscomo www.kaggle.com

1. Descripción del dataset: ¿Por qué es importante y qué pregunta/problema pretende responder?

Para este ejercicio se ha decidido utilizar el dataset City Lines, el cual esta disponible en Kaggle, contiene información de las líneas del metro, (suburbanos o metropolitanos), además de informaciòn como ciudades y lìneas de cobertura, existe también información acerca de nùmero de estaciones y ubicaciòn, fechas de apertura y cierre, entre otras.

2. Integración y selección de los datos de interés a analizar

En total posee 7 tablas, las cuales se detallan a continuación:

- 1. Ciudades
- 2. Estaciones
- 3. Líneas
- 4. Líneas y sus estaciones
- 5. Líneas y sus tramos
- 6. Sistemas
- 7. Tramos

Para el siguiente paso se realiza la lectura de los ficheros, así

2.1. Carga de los set de datos correspondientes

La data presentada en los set de datos pueden ser de utilidad para responder la siguiente pregunta, ¿Es necesario expandir la cobertura que el metro tiene actualmente?, cabe indicar que en muchos casos estas decisiones obedecen a temas políticos, antes que a temas técnicos, sin embargo para ejemplarizar esta situación se utilizará el set de datos presentado para tratar de contestar esta inquietud. Se considerará la ciudad de Madrid, vamos a ello.

3. Limpieza de datos

Para conocer de forma específica cada uno de los campos que cada tabla posee, se puede utilizar los siguientes comandos con la finalidad de identificar campos en común, mismos que puedan relacionarse y de ser el caso tratarlos para limpiarlos y que la data sea mucho más legible.

```
nms_cities <- tibble(names(cities))
nms_stations <- tibble(names(stations))
nms_lines <- tibble(names(lines))
nms_lines_st <- tibble(names(lines_st))
nms_lines_tr <- tibble(names(lines_tr))
nms_syst <- tibble(names(syst))
nms_tracks <- tibble(names(tracks))
```

```
nms_cities
```

```
## # A tibble: 7 x 1
##   `names(cities)`
##   <chr>
## 1 id
## 2 name
## 3 coords
## 4 start_year
## 5 url_name
## 6 country
## 7 country_state
```

```
nms_stations
```

```
## # A tibble: 7 x 1
##   `names(stations)`
##   <chr>
## 1 id
## 2 name
## 3 geometry
## 4 buildstart
## 5 opening
## 6 closure
## 7 city_id
```

```
nms_lines
```

```
## # A tibble: 7 x 1
##   `names(lines)`
##   <chr>
## 1 id
## 2 city_id
## 3 name
## 4 url_name
## 5 color
## 6 system_id
## 7 transport_mode_id
```

```
nms_lines_st
```

```
## # A tibble: 6 x 1
##   `names(lines_st)`
##   <chr>
## 1 id
## 2 station_id
## 3 line_id
## 4 city_id
## 5 created_at
## 6 updated_at
```

```
nms_lines_tr
```

```
## # A tibble: 6 x 1
##   `names(lines_tr)`
##   <chr>
## 1 id
## 2 section_id
## 3 line_id
## 4 created_at
## 5 updated_at
## 6 city_id
```

```
nms_syst
```

```
## # A tibble: 3 x 1
##   `names(syst)`
##   <chr>
## 1 id
## 2 city_id
## 3 name
```

```
nms_tracks
```

```
## # A tibble: 7 x 1
##   `names(tracks)`
##   <chr>
## 1 id
## 2 geometry
## 3 buildstart
## 4 opening
## 5 closure
## 6 length
## 7 city_id
```

Como se puede apreciar el campo city_id es compartido por los set de datos.

Además el campo stations_id es compartido con el set stations y el set lines_st.

El campo line_id es compartido por lines y lines_tr.

A partir de la descripción anterior se puede identificar que la data necesaria para responder la pregunta inicial se puede realizar con los siguientes campos:

- a. Del set cities:
  - id
  - name
  - start_year
  - country
- b. Del set stations:
  - id
  - name
  - city_id
  - opening
- b. Del set lines:
  - id
  - name
  - city_id
- b. Del set tracks:
  - id
  - city_id
  - opening
  - length

Por esta razón se realiza cambio de nombres de variables para poder identificarlos de mejor manera acorde a su fuente y se seleccionan los campos necesarios para el análisis.

```
cities_rdx <- cities %>% rename(city_id = id, city_name = name) %>% select(city_id, city_name, country)
stations_rdx <- stations %>%
  rename(station_id = id, station_name = name, opening_stat = opening) %>%
  select(station_id, station_name, opening_stat, city_id)
lines_rdx <- lines %>% mutate(line_id = id, line_name = name) %>% select(line_id, line_name, city_id)
tracks_rdx <- tracks %>% rename(track_id = id, opening_trck = opening) %>% select(track_id, city_id, op
```

Las transformaciones anteriores son necesarias para poder contestar las siguientes preguntas.

3.1. y 3.2 ¿Los datos contienen ceros o elementos vacíos? ¿Y valores extremos? ¿Cómo gestionarías cada uno de estos casos?. Identificación y tratamiento de valores extremos

Obtenemos el resumen de los datos:

```
summary(cities_rdx)
```

```
##     city_id        city_name           country
##  Min.   :  1.00   Length:334         Length:334
##  1st Qu.: 86.25   Class :character   Class :character
##  Median :169.50   Mode  :character   Mode  :character
##  Mean   :169.51
##  3rd Qu.:252.75
##  Max.   :338.00
```

```
summary(stations_rdx)
```

```
##    station_id    station_name        opening_stat       city_id
##  Min.   :    1   Length:15794       Min.   :     0   Min.   :  1.0
##  1st Qu.: 4203   Class :character   1st Qu.:  1903   1st Qu.: 78.0
##  Median : 8330   Mode  :character   Median :  1937   Median :107.0
##  Mean   : 8316                      Mean   :  3720   Mean   :123.3
##  3rd Qu.:12500                      3rd Qu.:  2000   3rd Qu.:139.0
##  Max.   :16558                      Max.   :999999   Max.   :331.0
##                                     NA's   :73
```

Una de las variables que presenta observaciones es la fecha de apertura (opening_stat) ya que contiene contiene valores extremos además de valores faltantes (73 fechas marcadas no disponibles) y otras marcadas como 0.

```
summary(lines_rdx)
```

```
##     line_id        line_name           city_id
##  Min.   :   5.0   Length:1343        Min.   :  1
##  1st Qu.: 389.5   Class :character   1st Qu.: 77
##  Median : 760.0   Mode  :character   Median :106
##  Mean   : 772.1                      Mean   :118
##  3rd Qu.:1136.5                      3rd Qu.:124
##  Max.   :1614.0                      Max.   :331
```

```
summary(tracks_rdx)
```

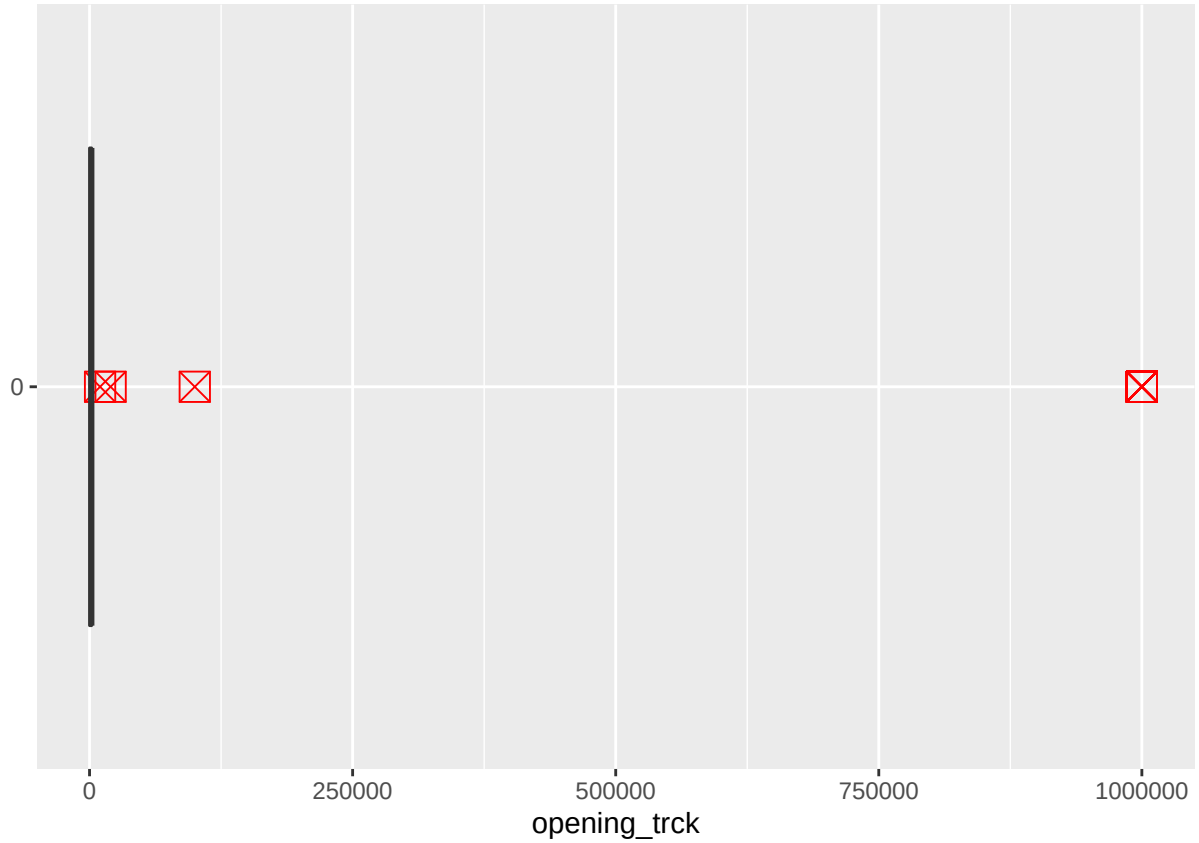
```
##     track_id        city_id       opening_trck        length
##  Min.   :    1   Min.   :  1.0   Min.   :     0   Min.   :     0
##  1st Qu.: 2696   1st Qu.: 74.0   1st Qu.:     0   1st Qu.:   118
##  Median : 5206   Median :114.0   Median :  1930   Median :   578
##  Mean   : 5294   Mean   :151.6   Mean   :  2766   Mean   :  2666
##  3rd Qu.: 7982   3rd Qu.:265.0   3rd Qu.:  2002   3rd Qu.:  2182
##  Max.   :10534   Max.   :331.0   Max.   :999999   Max.   :148768
##                                  NA's   :21
```

Para los siguientes set de datos se puede apreciar algo similar con la fecha de apertura de la vía (opening_trck), datos con valores ceros, vacíos y no disponibles y valores extremos, probablemente provienen de errores de entrada en los datos.

Por tal razón es necesarios utilizar las variables opening_trck, y opening_stat, se iniciará con un gráfico box-plot.

```
tracks_rdx %>%
  ggplot(aes(x = factor(0), y = opening_trck)) +
  geom_boxplot(outlier.color = "red",
               outlier.size =  5,
               outlier.shape = 7) +
  xlab(" ") +
  coord_flip()
```

```
## Warning: Removed 21 rows containing non-finite values (stat_boxplot).
```



Se añade la variable ciudad para conocer cual de ellas presenta mayor inconveniente.

```
options(tibble.width = Inf)
tracks_rdx %>%
  inner_join(cities_rdx) %>%
  filter(opening_trck > 9999) %>%
  select(city_name, opening_trck, length)
```

```
## Joining, by = "city_id"
```

```
## # A tibble: 15 x 3
##    city_name    opening_trck length
##    <chr>               <dbl>  <dbl>
##  1 Nantes             999999   2517
##  2 Nantes             999999   2302
##  3 São Paulo          999999  14613
##  4 Lima                99999   4154
##  5 Paris               20008   1082
##  6 São Paulo          999999   6936
##  7 São Paulo          999999  14946
##  8 Sydney             999999   5562
##  9 Sydney             999999   2983
## 10 Buenos Aires       999999   8705
## 11 São Paulo          999999   1101
## 12 Buenos Aires       999999   4944
## 13 Buenos Aires       999999   2032
```

```
## 14 Buenos Aires       999999   3738
## 15 Buenos Aires       999999   4294
```

Tambìen es interesante conocer aquellas ciudades que presentan valores 0:

```
options(tibble.width = Inf)
tracks_rdx %>%
  inner_join(cities_rdx) %>%
  filter(opening_trck == 0) %>%
  select(city_name, opening_trck, length) %>%
  count(city_name)
```

```
## Joining, by = "city_id"
```

```
## # A tibble: 47 x 2
##    city_name            n
##    <chr>            <int>
##  1 Barcelona            7
##  2 Brest                1
##  3 Budapest            38
##  4 Caracas              2
##  5 Chicago             35
##  6 Clermont-Ferrand     2
##  7 Edinburgh           22
##  8 Glasgow            894
##  9 Grenoble             2
## 10 Guadalajara          4
## # ... with 37 more rows
```

En cuanto a los valores NA, se tiene:

```
options(tibble.width = Inf)
tracks_rdx %>%
  inner_join(cities_rdx) %>%
  filter(is.na(opening_trck)) %>%
  select(city_name, opening_trck, length) %>%
  count(city_name)
```

```
## Joining, by = "city_id"
```

```
## # A tibble: 6 x 2
##   city_name          n
##   <chr>          <int>
## 1 Buenos Aires       6
## 2 Milan              3
## 3 Rio de Janeiro     2
## 4 Rome               1
## 5 São Paulo          8
## 6 Sydney             1
```

Las fechas de apertura de las estaciones se tiene:

```r
options(tibble.width = Inf)
stations_rdx %>%
  inner_join(cities_rdx) %>%
  filter(opening_stat > 9999) %>%
  select(city_name, opening_stat)
```

```
## Joining, by = "city_id"
```

```
## # A tibble: 31 x 2
##    city_name    opening_stat
##    <chr>               <dbl>
##  1 São Paulo          999999
##  2 São Paulo          999999
##  3 São Paulo          999999
##  4 São Paulo          999999
##  5 Buenos Aires       999999
##  6 Lima               999999
##  7 Lima               999999
##  8 Lima               999999
##  9 Lima               999999
## 10 Lima               999999
## # ... with 21 more rows
```

Con valores 0

```r
options(tibble.width = Inf)
stations_rdx %>%
  inner_join(cities_rdx) %>%
  filter(opening_stat == 0) %>%
  select(city_name, opening_stat) %>%
  count(city_name)
```

```
## Joining, by = "city_id"
```

```
## # A tibble: 51 x 2
##    city_name     n
##    <chr>     <int>
##  1 Amsterdam     1
##  2 Angers        1
##  3 Barcelona    11
##  4 Beijing       3
##  5 Berlin        3
##  6 Boston        1
##  7 Brussels      4
##  8 Budapest     62
##  9 Caracas       9
## 10 Chicago      73
## # ... with 41 more rows
```

y con valores NA

```
options(tibble.width = Inf)
stations_rdx %>%
  inner_join(cities_rdx) %>%
  filter(is.na(opening_stat)) %>%
  select(city_name, opening_stat) %>%
  count(city_name)
```

```
## Joining, by = "city_id"
```

```
## # A tibble: 8 x 2
##   city_name          n
##   <chr>          <int>
## 1 Buenos Aires      10
## 2 Grenoble           1
## 3 Milan             23
## 4 Paris              5
## 5 Rio de Janeiro     1
## 6 Rome               2
## 7 São Paulo         27
## 8 Sydney             4
```

Para poder dar la respuesta a la pregunta inicial se procede a segmentar la informaciòn con respecto a la ciudad de Madrid.

Por esta razón se creará un nuevo set de datos que incluya la información sobre tramos y estacione.

```
tracks_mb <- tracks_rdx %>%
              inner_join(cities_rdx) %>%
              filter(city_name == "Madrid")
```

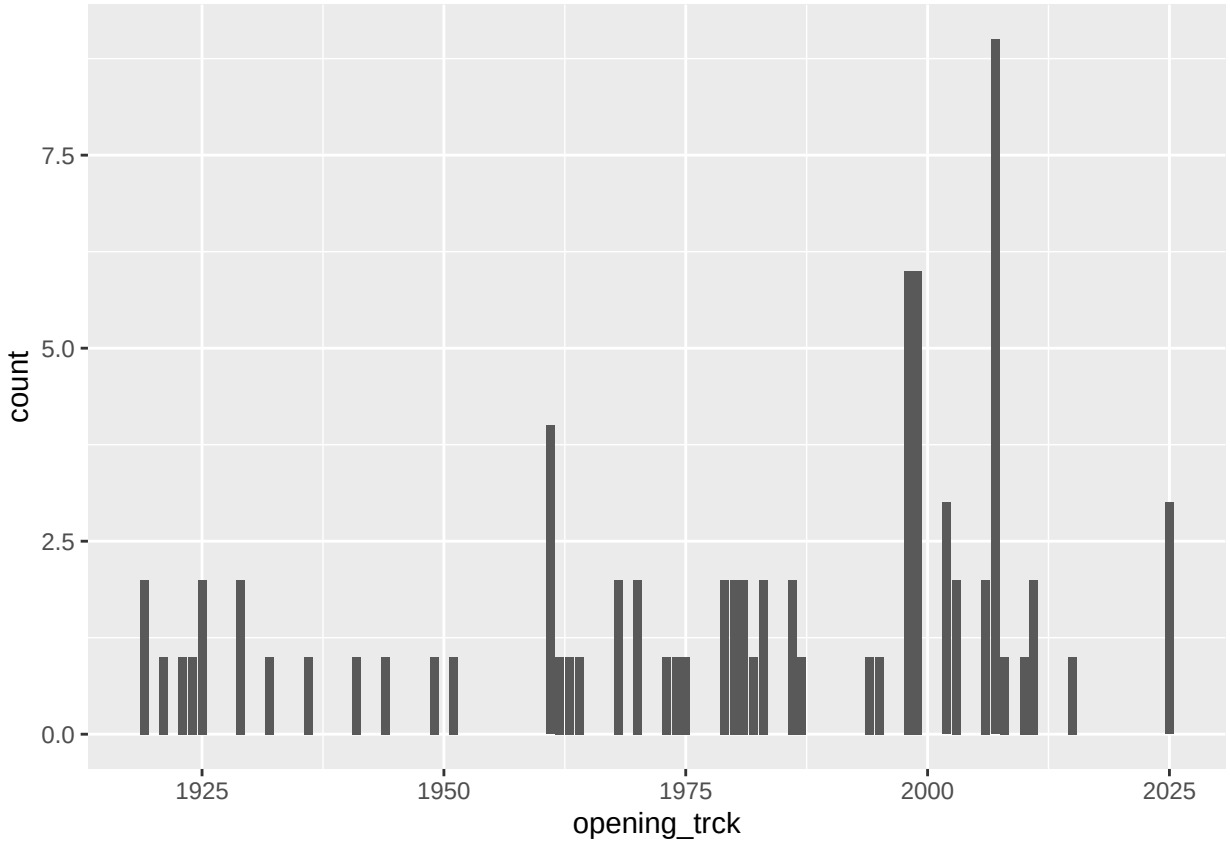
```
## Joining, by = "city_id"
```

```
tracks_mb %>% head()
```

```
## # A tibble: 6 x 6
##   track_id city_id opening_trck length city_name country
##      <dbl>   <dbl>        <dbl>  <dbl> <chr>     <chr>
## 1      236      71         2002   1332 Madrid    Spain
## 2      192      71         1949   1304 Madrid    Spain
## 3      195      71         1963    574 Madrid    Spain
## 4      219      71         1982   5774 Madrid    Spain
## 5      269      71         2007   2704 Madrid    Spain
## 6      267      71         2010   2380 Madrid    Spain
```

Por tal razòn se realizará una revisión a los años de apertura.

```
tracks_mb %>%
  ggplot(aes(opening_trck)) +
  geom_bar(stat = "count")
```



Claramente se puede apreciar que existe problemas en la informaciòn pues existen datos que indican que se han habilitado tramos en el año 2025, lo cual se puede deber a errores de digitalizaciòn.

Por esta razón se realizará un ejercicio similar con el set de estaciones.

```
stations_mb <- stations_rdx %>%
               inner_join(cities_rdx) %>%
               filter(city_name == "Madrid") %>%
               mutate(station_name = str_replace(station_name, "\n\n", ""))
```

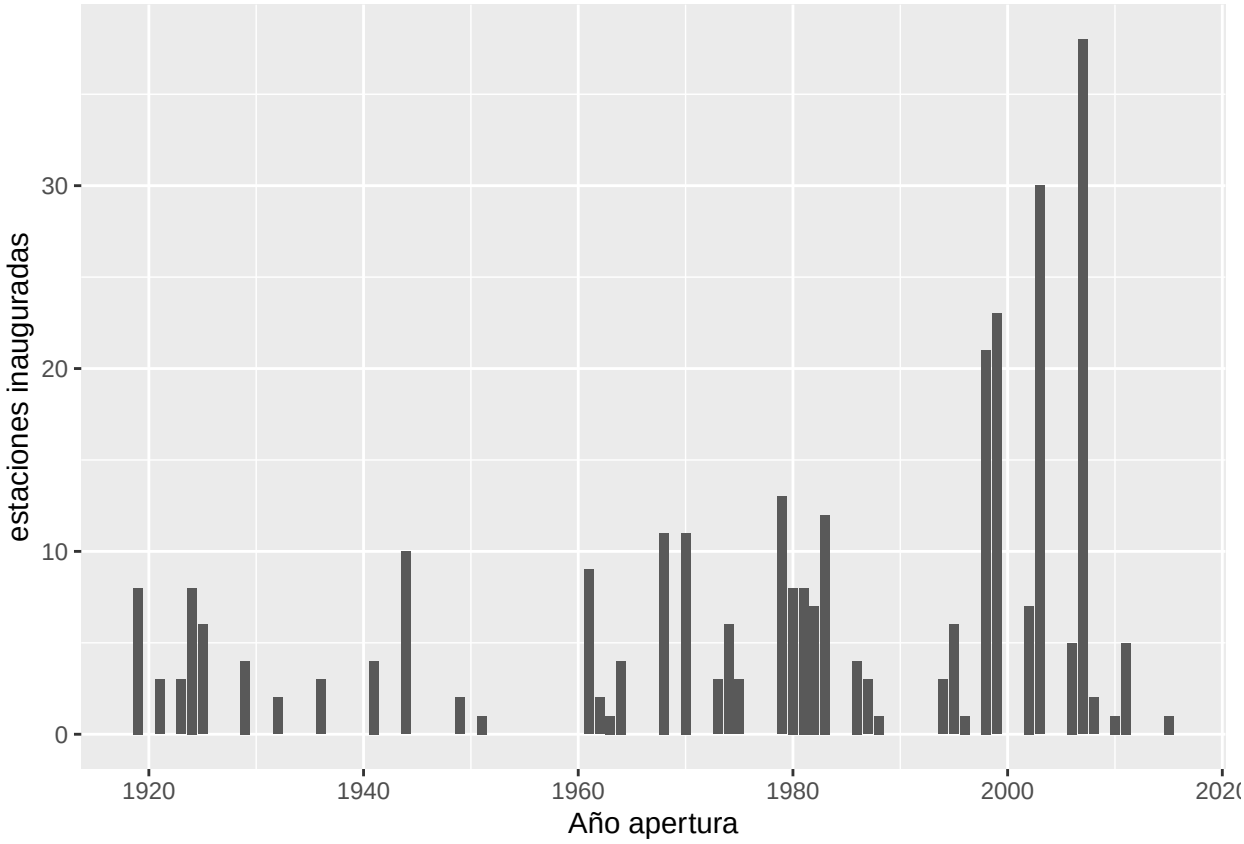
```
## Joining, by = "city_id"
```

```
stations_mb %>% head()
```

```
## # A tibble: 6 x 6
##   station_id station_name     opening_stat city_id city_name country
##        <dbl> <chr>                   <dbl>   <dbl> <chr>     <chr>
## 1        521 Miguel Hernández         1994      71 Madrid    Spain
## 2        545 <NA>                     2003      71 Madrid    Spain
## 3        561 Arganda del Rey          1999      71 Madrid    Spain
## 4        309 Sol                      1919      71 Madrid    Spain
## 5        324 Sevilla                  1924      71 Madrid    Spain
## 6        311 Tribunal                 1919      71 Madrid    Spain
```

En cuanto a su gráfica tenemos:

```
stations_mb %>%
  ggplot(aes(opening_stat)) +
  geom_bar(stat = "count") +
  ylab("estaciones inauguradas") +
  xlab("Año apertura")
```



Esta gráfica un comportamiento interesante, ya que se pueden observar picos interesantes. Podemos ver que en 2015, 2011, 2007, 2003, 1999 se inauguraron varias estaciones de metro. Estos años coinciden con elecciones municipales en la Comunidad de Madrid.

Se realiza un cruce con la tabla de las estaciones, con la finalidad de poder profundizar en la data

```
stations_mb <- stations_mb %>%
               inner_join(lines_st)
```

```
## Joining, by = c("station_id", "city_id")
```

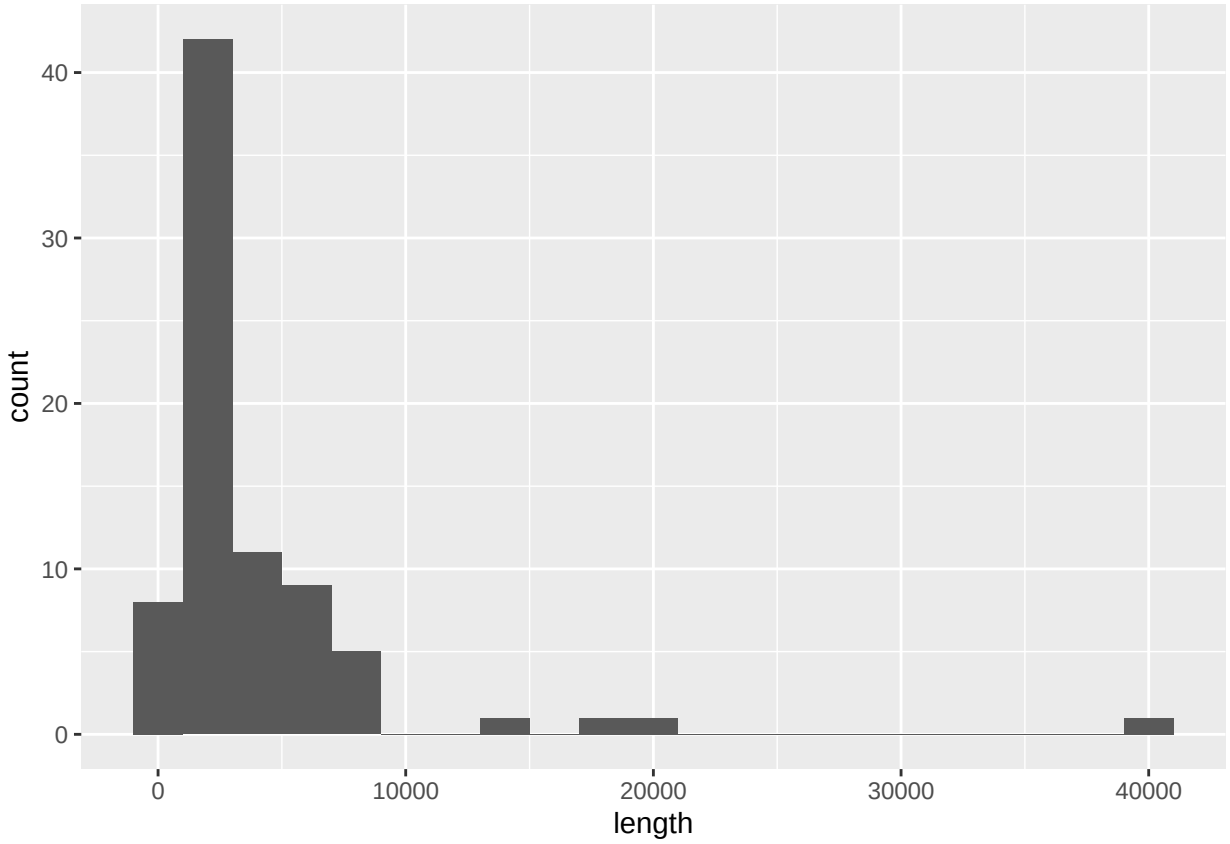
```
stations_mb <- stations_mb %>% select(station_name, line_id, opening_stat)
```

4. Análisis de datos

Todo lo realizado anteriormente se utilizará para un análisis y comparaciones de variables.

Tomando como premisa la longitud en kilómetros de tramos de metro en la cuidad de Madrid, se realiza un histograma de frecuencias para identificar si la distribución en kilómetros de los tramos del metro de Madrid presenta una distribución normal.

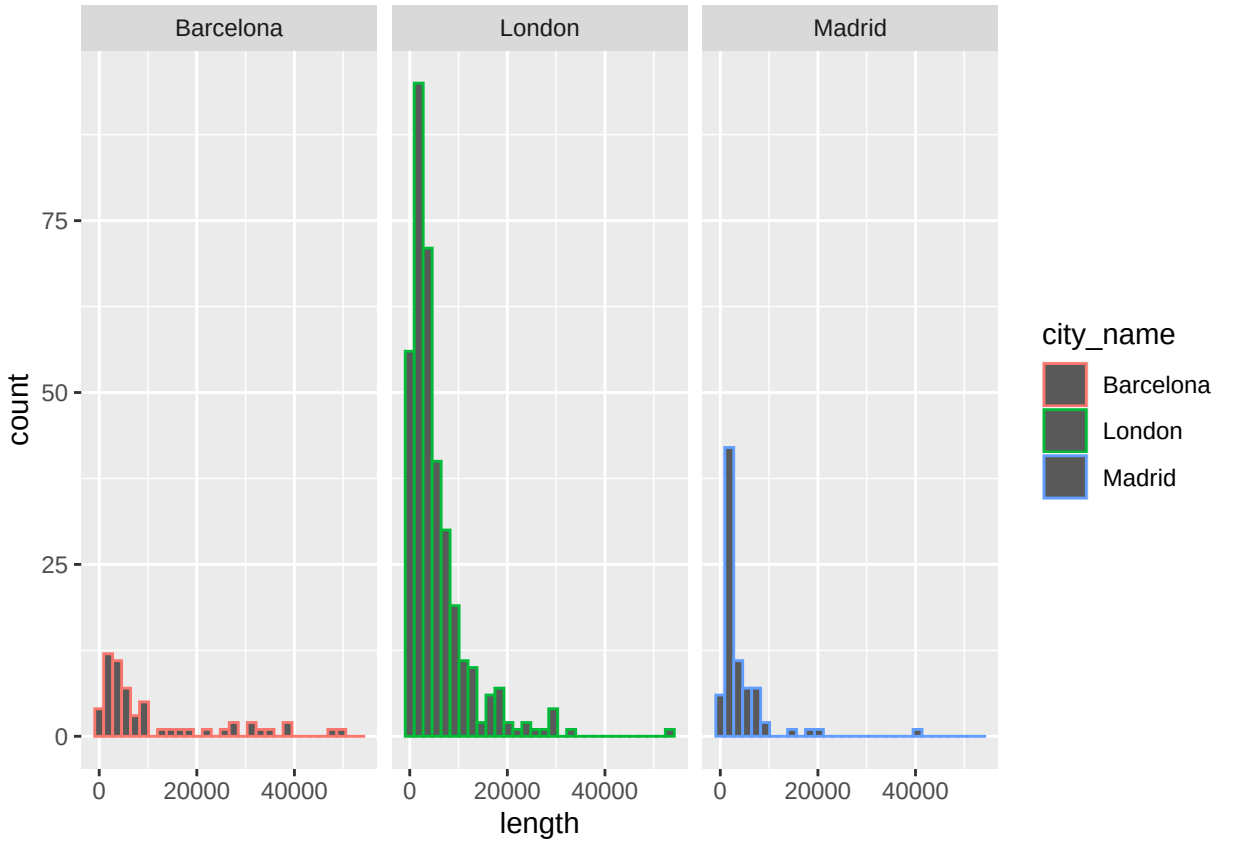
```
tracks_mb %>%
  ggplot(aes(length)) +
  geom_histogram(binwidth = 2000)
```




```
tracks_rdx %>%
        inner_join(cities_rdx) %>%
        filter(city_name %in% c("Madrid", "Barcelona", "London")) %>%
        ggplot(aes(length, color = city_name)) +
        geom_histogram() +
        facet_wrap( ~ city_name)
```

```
## Joining, by = "city_id"
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Todas las ilustraciones presentadas son ilustrativas, pues las extensiones de las vías dependerán en gran medida a la zona geográfica de la ciudad, antiguedad de las vías. La idea principal de la comparación de la extensión, por ejemplo, Madrid muestran en el gráfico un grupo mayor de estaciones entre 3 y 5 kilómetros.

Es necesario tomar en cuenta que la tabla tracks_rdx no está normalizada como la tabla tracks_mb para Madrid. Sin embargo es importante preguntarse cuáles son las causas que hacen que se aumente la red del suburbano en tiempos en los que hay más incidencia del automóvil y el despliegue de otras formas de movilidad como el uso de bicicletas.

Todo esto debido a que en Madrid, existen comentarios sobre el sobre-dimensionamiento de la red, lo cual puede obedecer decisiones que no estén responden a necesidades sociales, por lo menos no directamente, y quizá sea una respuesta a una estrategia política.

5. Representación a partir de tablas y gráficos

Las tablas y gráficos se presentan a lo largo del desarrollo de los puntos anteriores.

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones, ¿Los resultados permiten responder al problema?

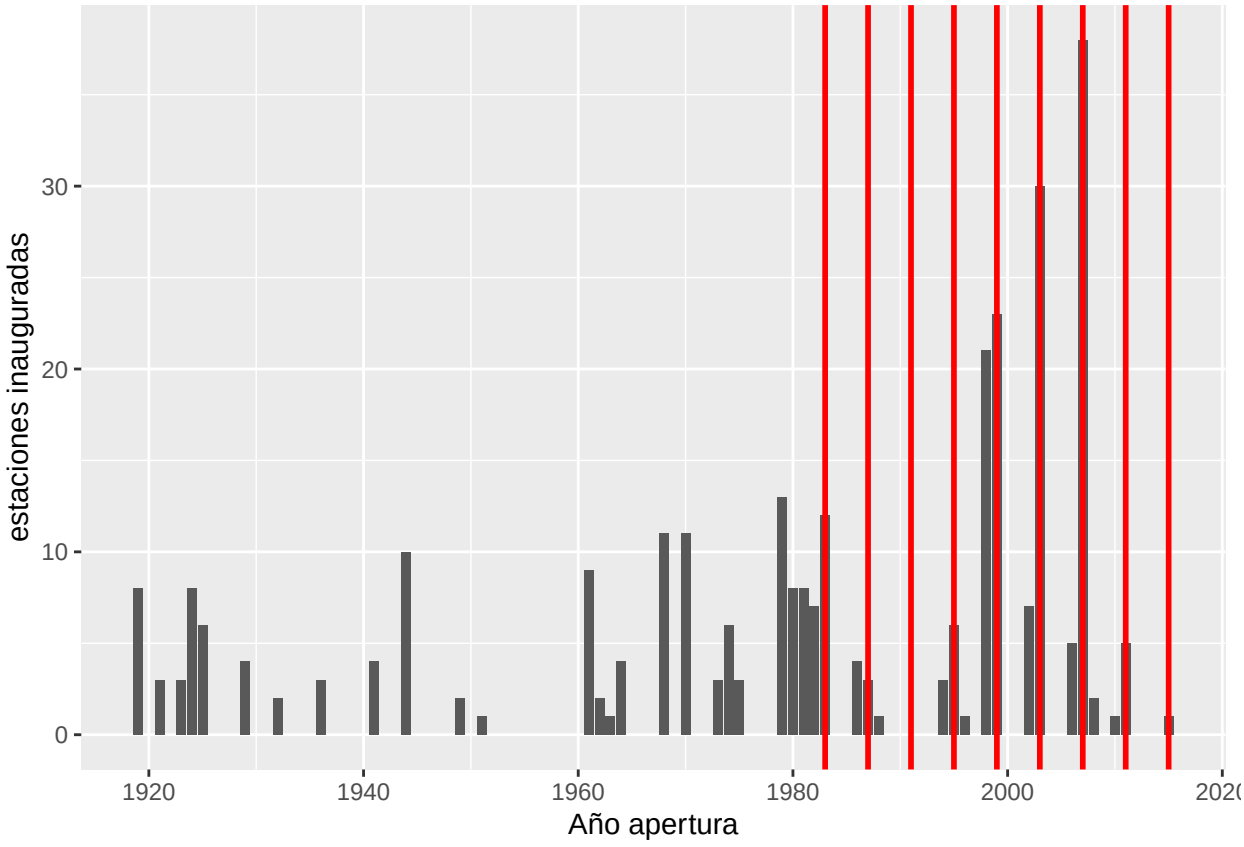
La amplicaciòn de la rede del metro, en tramos como en estaciones de la Ciudad de Madrid, acorde a lo fundamentado por la informaciòn vista, obedece más a estratégias polìticas de cara a elecciones, antes que a cubrir necesidades de movilidad de la poblaciòn.

Las investigaciones sobre estaciones infrautilizadas e inversiones no corresponden con la realidad que los usuarios se han ido encontrado retrasos y poca rentabilidad por encima del servicio.

Para conocimiento general se tiene que las elecciones para la Asamblea de Madrid, se realizan cada 4 años, desde el año 1983 en el inicio de la democracia. Las últimas se realizaron en 2019. Las próximas serán en 2019.

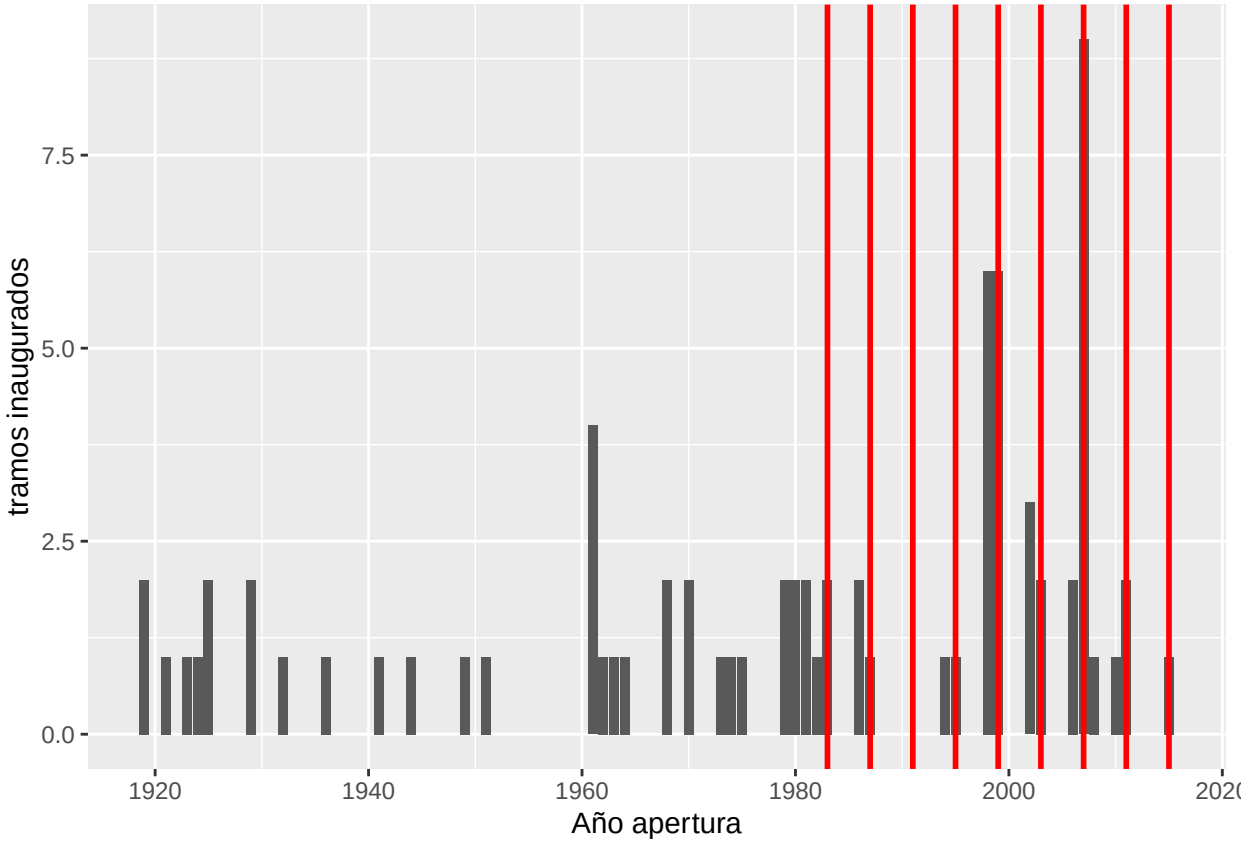
Para mostrar lo argumentado anteriormente se puede apreciar en la siguiente gráfica, donde se muestra que previo los años de elecciones para cargos políticos se incrementa la apertura de tramos y estaciones.

```
stations_mb %>%
  ggplot(aes(opening_stat)) +
  geom_bar(stat = "count") +
  ylab("estaciones inauguradas") +
  xlab("Año apertura") +
  geom_vline(xintercept = seq(1983, 2015, by = 4), color = "red", size = 1.0)
```



Considerando la apertura de los tramos se tiene:

```
tracks_mb %>%
  filter(opening_trck != 2025) %>%
  ggplot(aes(opening_trck)) +
  geom_bar(stat = "count") +
  ylab("tramos inaugurados") +
  xlab("Año apertura") +
  geom_vline(xintercept = seq(1983, 2015, by = 4), color = "red", size = 1.0)
```



Un uso adecuado de los recursos destinados a mejora las infraestructuras de tramos y estaciones, actualmente es primordial y ya se han hecho acciones encaminadas a ello y que no coinciden con calendarios electorales, sin embargo muchos de las mejoras realizadas se utilizan como una plataforma polìtica, seguramente para buscar reelecciones o continuidad en el partido político de turno.

Finalmente, se añade que se tiene previsto para 2019 y 2023, según el calendario municipal, ampliaciones en la red y aperturas de nuevas estaciones que beneficiarán la periferia de Madrid.

Firma de responsabilidad

- Contribuciones Firma

Investigación previa: Darwin Padilla

Redacción de las respuestas: Darwin Padilla

Desarrollo código: Darwin Padilla

```
#Guardar archivos analizados CSV
write.csv(cities_rdx,file = "datanew/cities_rdx.csv")
write.csv(lines_rdx,file = "datanew/lines_rdx.csv")
write.csv(stations_mb,file = "datanew/stations_mb.csv")
write.csv(stations_rdx,file = "datanew/stations_rdx.csv")
write.csv(tracks_mb,file = "datanew/tracks_mb.csv")
write.csv(tracks_rdx,file = "datanew/tracks_rdx.csv")
```