Control 2, respuestas

Web Scraping y acceso a datos desde la web

Cristián Ayala

Ponderación 20% de la nota final del curso

Formato Desarrollar esta tarea con Quarto o Rmarkdown generando un `.pdf`, agregando comentarios cuando sea necesario.

1 Objetivo:

Interesa indagar sobre el cine chileno. Queremos saber la evolución del número de películas chilenas estrenadas por año y su calificación según la nota dada por IMDb.

Para ello usaremos el sitio web IMDb para filtrar películas chilenas realizadas en Chile. En total son **465**[1] según se muestra en esta búsqueda:

https://www.imdb.com/search/title/?title_type=feature&countries=cl&locations=chile

2 Tareas:

2.1 Captura de datos 1 página

1) Desde esa página web se puede capturar las primeras 25 películas.

Los objetos están dentro de `<div>` con clase `.cVXqoq`.

- Index y título: `.ipc-title`
- Año de estreno: `.dli-title-metadata-item`
- Puntaje Metacritic: `.metacritic-score-box`
- Puntaje IMDb: `.ratingGroup--imdb-rating`

[1] Número de películas al momento de diseñar este control.

2.1.1 Lectura web

La página inicial carga 25 películas.

El género se puede encontrar en la página de cada película.

```
url_1 <- 'https://www.imdb.com/search/title/?title_type=feature&countries=cl&locations=chile'
url_parse_1 <- url_parse(url_1)

l_pelicula_1_html <- read_html(url_build(url_parse_1))
```

¿Cuántas son las películas totales que están presente en la búsqueda?

```
n_peliculas <- l_pelicula_1_html |>
  html_elements('.gJQFCa') |>
  html_text2()

n_peliculas <- n_peliculas |>
  str_extract(' of (\\d+)', group = 1) |>
  as.integer()

n_peliculas
```

```
[1] 465
```

```
l_pelicula_1_html |>
  html_elements('.cVXqoq .ipc-title') |>
  html_text2()
```

```
 [1] "1. La sociedad de la nieve"      "2. Knock Knock: Seducción Fatal"
 [3] "3. Caníbales"                    "4. Trauma"
 [5] "5. Los colonos"                  "6. Los 33"
 [7] "7. Diarios de motocicleta"       "8. El Conde"
 [9] "9. La contadora de películas"    "10. Dry Martina"
[11] "11. Una Mujer Fantástica"        "12. La danza de la realidad"
[13] "13. La Casa Lobo"                "14. La memoria infinita"
[15] "15. Cola de Mono"                "16. Aftershock"
[17] "17. El Príncipe"                 "18. Ema"
[19] "19. Joven y Alocada"             "20. Los Fuertes"
[21] "21. No"                          "22. Poesía Sin Fin"
[23] "23. El Club"                     "24. Pacto de Fuga"
[25] "25. Sin Filtro"
```

2.1.2 data.frame

Selección de datos de interés

```
# Obtener lista de nodo de películas
selectores <- c(index_titulo  = '.ipc-title',
                anio_duracion_clasificacion   = '.dli-title-metadata-item',
                rating_meta = '.metacritic-score-box',
                rating_imdb = '.ratingGroup--imdb-rating')

# Función para extraer la información de cada página capturada.
f_capturar_elementos <- function(.html,
                                 .selector,
                                 .names_sel){

  html <- .html |>
    html_elements('.cVXqoq') # caja de cada película.

  # Captura general del elemento de interés.
  data <- html |>
    html_element(.selector) |>
    html_text() |>
    str_squish()

  l_data <- setNames(list(data),
                     nm = .names_sel)

  # Captura de link a la película solo si estoy viendo elemento nominado título
  if (.names_sel == 'index_titulo'){
    links <- html |>
      html_element(paste0(.selector, ' a')) |>
      html_attr('href')

    l_data <- append(l_data,
                     list(link = links))
  }

  # Devuelvo los datos capturados: un vector con texto y links.
  l_data
}

# Itero todos los selectores en la única página que capturamos.
l_peliculas <- map(list(l_pelicula_1_html),
                   \(l_pel){
                     map2(selectores, names(selectores),
                          \(selector, names_sel){
                            f_capturar_elementos(l_pel, selector, names_sel)
                          }
                     )
```

```
                  }
)

# Función para construir una df a partir de lo capturado.
datos_a_df <- function(.datos){
  list_flatten(.datos) |> # quita un nivel
    discard(is.null) |>  # elimina variables vecias
    as_tibble() # transforma listas a df
}

# Lista de tibbles de cada página
l_peliculas_df <- l_peliculas |>
  map(datos_a_df)

# Creación de tibble única
df_peliculas <- l_peliculas_df |>
  list_rbind()

df_peliculas |> dim()
```

```
[1] 25  5
```

Mejora de nombres de las columnas de `df_peliculas`.

```
names(df_peliculas) <- str_replace(names(df_peliculas), "^(.*)_\\1$", "\\1")

df_peliculas |> names()
```

```
[1] "index_titulo"                "index_titulo_link"
[3] "anio_duracion_clasificacion" "rating_meta"
[5] "rating_imdb"
```

2) Guardar esa información en un data.frame

```
head(df_peliculas)
```

```
# A tibble: 6 x 5
  index_titulo  index_titulo_link anio_duracion_clasif~1 rating_meta rating_imdb
  <chr>         <chr>              <chr>                  <chr>       <chr>
1 1. La socied~ /title/tt1627724~ 2023                   72          7.8 (138K)
2 2. Knock Kno~ /title/tt3605418~ 2015                   53          4.9 (105K)
3 3. Caníbales  /title/tt2403021~ 2013                   38          5.4 (50K)
4 4. Trauma     /title/tt6705640~ 2017                   <NA>        4.8 (2.5K)
5 5. Los colon~ /title/tt1037081~ 2023                   80          7.0 (3.7K)
6 6. Los 33     /title/tt2006295~ 2015                   55          6.9 (40K)
# i abbreviated name: 1: anio_duracion_clasificacion
```

Lectura de cada página de películas en la lista

```
df_peliculas <- df_peliculas |>
  mutate(index_titulo_link = paste0('https://www.imdb.com', index_titulo_link))

df_peliculas <- df_peliculas |>
  rowwise() |>
  mutate(web_titulo = list(read_html(index_titulo_link)))
```

Extracción de género por película

```
f_genero <- function(.html){
  .html |>
    html_elements('.ipc-chip--on-baseAlt') |>
    html_text2()
}

df_peliculas <- df_peliculas |>
  mutate(genero = list(f_genero(web_titulo))) |>
  ungroup()
```

Corregiremos alguna de las variables extraídas para el análisis siguiente.

```
df_peliculas <- df_peliculas |>
  separate_wider_delim(index_titulo,
                       delim = '. ',
                       names = c('index', 'titulo')) |>
  separate_wider_regex(rating_imdb,
                       patterns = c(rating_imdb = '.*', ' \\(', votes_imdb = '.*', '\\)$'))

df_peliculas <- df_peliculas |>
  mutate(across(c(index, rating_meta), as.integer),
         across(c(rating_imdb), as.double),
         anio = as.Date(paste0(anio_duracion_clasificacion, '-01-01', '%Y-%M-$d')))

head(df_peliculas)
```

```
# A tibble: 6 x 10
  index titulo  index_titulo_link anio_duracion_clasif~1 rating_meta rating_imdb
  <int> <chr>   <chr>             <chr>                         <int>       <dbl>
1     1 La soc~ https://www.imdb~ 2023                             72         7.8
2     2 Knock ~ https://www.imdb~ 2015                             53         4.9
3     3 Caníba~ https://www.imdb~ 2013                             38         5.4
4     4 Trauma  https://www.imdb~ 2017                             NA         4.8
5     5 Los co~ https://www.imdb~ 2023                             80         7
6     6 Los 33  https://www.imdb~ 2015                             55         6.9
# i abbreviated name: 1: anio_duracion_clasificacion
# i 4 more variables: votes_imdb <chr>, web_titulo <list>, genero <list>,
#   anio <date>
```

2.1.3 Análisis

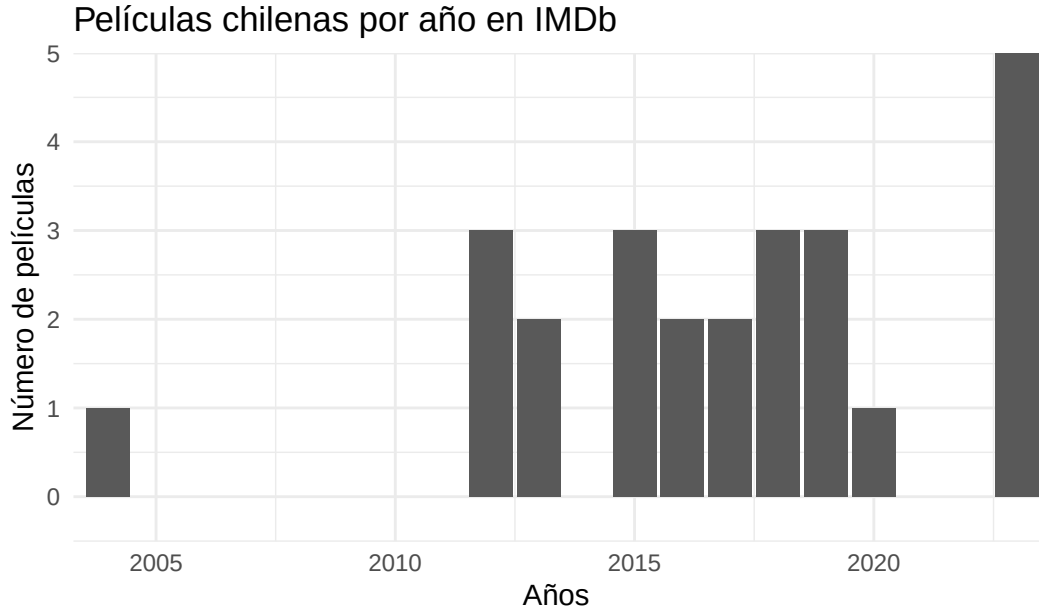
2.1.3.1 Número de películas

3) Graficar la evolución del el *número de películas* (eje y) estrenadas por *año* (eje x).

```
df_peliculas_anio <- df_peliculas |>
  count(anio, name = 'n_peliculas')

f_gg_peliculas_anio <- function(.df){
  .df |>
    ggplot(aes(x = anio, y = n_peliculas)) +
    geom_col() +
    scale_x_date('Años', expand = expansion(add = c(100, 0))) +
    scale_y_continuous(expand = expansion(add = c(0.5, 0))) +
    labs(title = 'Películas chilenas por año en IMDb',
         caption = 'Fuente: IMDb.com. Web Scraping y acceso a datos desde la web',
         y = 'Número de películas') +
    theme_minimal()
}

df_peliculas_anio |>
  f_gg_peliculas_anio()
```



2.1.3.2 Rating

5) Graficar la evolución del el *rating IMDb* promedio (eje y) *estrenadas desde 1990* a la fecha (eje x).

```
f_df_peliculas_rank <- function(.df,
                                .ranking){
  .df |>
    filter(anio >= as.Date("1990-01-01")) |>
    summarise(n_peliculas = n(),
              rating_imdb = mean({{ .ranking }}, na.rm = TRUE),
              .by = anio) |>
    arrange(anio)
}

df_peliculas_rank <- df_peliculas |>
  f_df_peliculas_rank(rating_imdb)

df_peliculas_rank |>
  head()
```

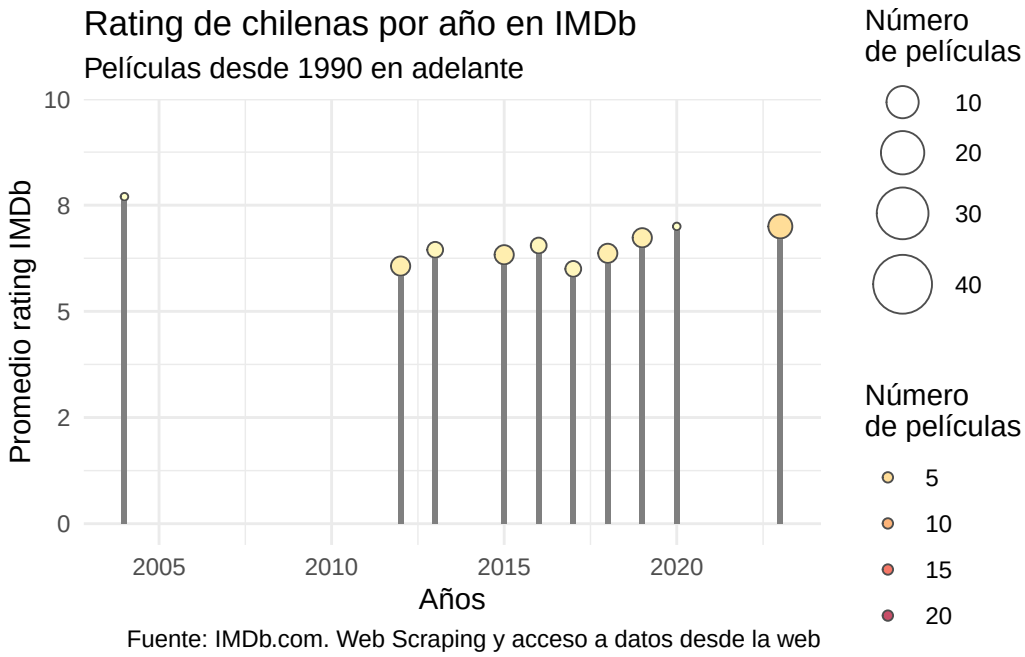
```
# A tibble: 6 x 3
  anio       n_peliculas rating_imdb
  <date>           <int>       <dbl>
1 2004-01-01           1        7.7
2 2012-01-01           3        6.07
3 2013-01-01           2        6.45
4 2015-01-01           3        6.33
5 2016-01-01           2        6.55
6 2017-01-01           2        6
```

```
f_gg_peliculas_rank <- function(.df){

  .df |>
    ggplot(aes(x = anio, y = rating_imdb,
               fill = n_peliculas,
               size = n_peliculas)) +
    geom_col(width = 60, fill = 'gray50',
             show.legend = F) +
    geom_point(colour = 'white') +
    geom_point(shape = 21,
               alpha = .7) +
    scale_x_date('Años',
                 expand = expansion(add = c(400, 400))) +
    scale_y_continuous(limits = c(0, 10),
                       expand = expansion(add = c(.5, 0)),
                       labels = round) +
    scale_fill_distiller('Número\nde películas',
```

```
                          palette = 'YlOrRd',
                          direction = 1,
                          limits = c(1, 20),
                          breaks = scales::pretty_breaks(4)) +
    scale_size_continuous('Número\nde películas',
                          range = c(1, 10),
                          limits = c(1, 40),
                          breaks = scales::pretty_breaks(5)) +
    guides(fill = guide_legend(),
           size = guide_legend()) +
    labs(title = 'Rating de chilenas por año en IMDb',
         subtitle = 'Películas desde 1990 en adelante',
         caption = 'Fuente: IMDb.com. Web Scraping y acceso a datos desde la web',
         y = 'Promedio rating IMDb') +
    theme_minimal()
}

df_peliculas_rank |>
  f_gg_peliculas_rank()
```



2.1.3.3 Género

6) ¿Cuál es el *género* que tienen el *mejor puntaje promedio* considerando películas estrenadas desde 1990 a la fecha?

Modificar base para que la unidad de análisis sea `genero`.

```
f_df_genero <- function(.df,
                        .rating){
  .df |>
    summarise(n_peliculas = n(),
              n_peliculas_con_rating = sum(!is.na({{ .rating }})),
              rating_imdb = mean({{ .rating }}, na.rm = TRUE),
              .by = genero) |>
    arrange(-rating_imdb)
}

df_genero <- df_peliculas |>
  filter(anio >= as.Date("1990-01-01")) |>
  select(index, rating_imdb, genero) |>
  unnest_longer(col = genero)

df_genero <- df_genero |>
  f_df_genero(rating_imdb)

head(df_genero)
```

```
# A tibble: 6 x 4
  genero      n_peliculas n_peliculas_con_rating rating_imdb
  <chr>             <int>                  <int>       <dbl>
1 Animation             1                      1        7.5
2 Biography             5                      5        7.48
3 Documentary           1                      1        7.4
4 Mystery               1                      1        7.2
5 Fantasy               3                      3        7.13
6 Romance               2                      2        6.95
```

El género con mejor puntaje promedio desde 1990 es **Biography**.

```
df_genero |>
  arrange(-n_peliculas) |>
  head()
```

```
# A tibble: 6 x 4
  genero    n_peliculas n_peliculas_con_rating rating_imdb
  <chr>           <int>                  <int>       <dbl>
1 Drama              19                     19        6.83
2 Biography           5                      5        7.48
3 History             4                      4        6.93
4 Adventure           4                      4        6.42
5 Thriller            4                      4        6.22
6 Comedy              4                      4        5.95
```

2.2 Captura de datos 465 películas

La información de películas adicionales se carga llamando a una API de imbd. Para ello, al momento de hacer una búsqueda, se solicita información de la siguiente url:

`https://caching.graphql.imdb.com/?operationName=AdvancedTitleSearch&variables=XXXXXX`

Luego de analizar la forma en que se piden los datos al servidor podemos construir el request correspondiente.

2.2.1 query

Lista en R con las variables que recibe el servidor. Esta lista contiene nuestros parámetros de interés.

```
l_query_variable <- list(
  "filmingLocationConstraint" = list(allLocations = list("chile")),
  "first" = 50, # ojo con esto. Podrá ser manipulado.
  "locale" = "en-US",
  "originCountryConstraint" = list(allCountries = list("CL")),
  "sortBy" = "POPULARITY",
  "sortOrder" = "ASC",
  "titleTypeConstraint" = list(
    anyTitleTypeIds = list("movie"),
    excludeTitleTypeIds = list("")
  )
)
```

Luego será traducida a un json al momento de generar el request.

2.2.2 request

A partir de la revisión del inspector, podemos construir el request necesario. La función tiene tres parámetros de utilidad:

- `.l_query_var`: list con parámetros para el query del request.
- `.first`: número de películas a buscar
- `.after`: parámetro para busqueda paginada. Señala hasta donde quedó la entrega de datos en una búsqueda anterior.

```
f_req_imbd <- function(.l_query_var,
                       .first = 50,
                       .after = NULL){

  .l_query_var$first <- .first
  .l_query_var$after <- .after

  json_query_variable <- .l_query_var |>
    jsonlite::toJSON(auto_unbox = TRUE)
```

```
  httr2::request(base_url = 'https://caching.graphql.imdb.com/') |>
    req_headers('Accept' = 'application/json',
                'Content-Type' = 'application/json') |>
    req_url_query(operationName = 'AdvancedTitleSearch',
                  variables = json_query_variable,
                  # Dato necesario
                  extensions = '{"persistedQuery":{"sha256Hash":"42714660b115c035a3c14572bfd2765c622e
    )
}
```

Al poder controlar el número de películas solicitadas, capturaré con una sola llamada las 465 películas de interés.

```
req_movies <- f_req_imbd(l_query_variable,
                         .first = 465)

# Formato de request
req_movies |>
  req_dry_run()
```

```
GET /?operationName=AdvancedTitleSearch&variables=%7B%22filmingLocationConstraint%22%3A%7B%22allLocat
Host: caching.graphql.imdb.com
User-Agent: httr2/1.0.1 r-curl/5.2.1 libcurl/8.6.0
Accept-Encoding: deflate, gzip
Accept: application/json
Content-Type: application/json
```

2.2.3 response

Lectura de datos

```
resp_movies <- req_perform(req_movies)

resp_movies
```

Recibimos un json como respuesta

```
j_movies <- resp_movies |>
  resp_body_json()

j_movies |> str(2)
```

```
List of 2
 $ data      :List of 1
  ..$ advancedTitleSearch:List of 6
```

```
 $ extensions:List of 2
  ..$ disclaimer        : chr "Public, commercial, and/or non-private use of the IMDb data provided b
  ..$ experimentalFields:List of 6
```

```
j_movies$data |> str(2)
```

```
List of 1
 $ advancedTitleSearch:List of 6
  ..$ total     : int 465
  ..$ pageInfo  :List of 4
  ..$ genres    :List of 22
  ..$ keywords  :List of 100
  ..$ titleTypes:List of 1
  ..$ edges     :List of 465
```

Datos de películas en `edges`.

```
j_movies$data$advancedTitleSearch$edges[[1]] |> str(3)
```

```
List of 1
 $ node:List of 1
  ..$ title:List of 16
  .. ..$ id               : chr "tt16277242"
  .. ..$ titleText        :List of 1
  .. ..$ titleType        :List of 4
  .. ..$ originalTitleText:List of 1
  .. ..$ primaryImage     :List of 5
  .. ..$ releaseYear      :List of 2
  .. ..$ ratingsSummary   :List of 2
  .. ..$ runtime          :List of 1
  .. ..$ certificate      :List of 1
  .. ..$ canRate          :List of 1
  .. ..$ titleGenres      :List of 1
  .. ..$ canHaveEpisodes  : logi FALSE
  .. ..$ plot             :List of 1
  .. ..$ latestTrailer    :List of 1
  .. ..$ series           : NULL
  .. ..$ metacritic       :List of 1
```

2.2.4 data.frame

Obtención de los datos de interés interés a partir del json capturado

```r
f_build_df_movie <- function(.l_json){

  # número de películas en la busqueda
  n_movie <- .l_json[['data']][['advancedTitleSearch']][['total']]

  # siguiente página de la búsqueda luego de la respuesta recibida
  next_page <- .l_json[['data']][['advancedTitleSearch']][['pageInfo']][['endCursor']]

  # datos de películas
  df_movie <- .l_json[['data']][['advancedTitleSearch']][['edges']] |>
    bind_rows() |>
    unnest_wider(node) |>
    unnest_wider(col = c(titleText,
                         originalTitleText,
                         ratingsSummary, releaseYear, metacritic),
                 names_sep = '_')

  list(n_movie = n_movie,
       next_page = next_page,
       df = df_movie)
}

l_movies <- f_build_df_movie(j_movies)

l_movies |> str(2)
```

```
List of 3
 $ n_movie  : int 465
 $ next_page: chr "eyJlc1Rva2VuIjpbIjI1NDA1MzAiLCIyNTQwNTMwIiwidHQxNjY4ODIyNCJdLCJmaWx0ZXIiOiJ7XCJjb2
 $ df       : tibble [465 x 18] (S3: tbl_df/tbl/data.frame)
```

data.frame de películas es el siguiente

```r
df_movies <- l_movies$df
```

Ajustes en variable de año.

```r
df_movies <- df_movies |>
  mutate(anio = as.Date(paste0(releaseYear_year, '-01-01', '%Y-%M-$d')))
```

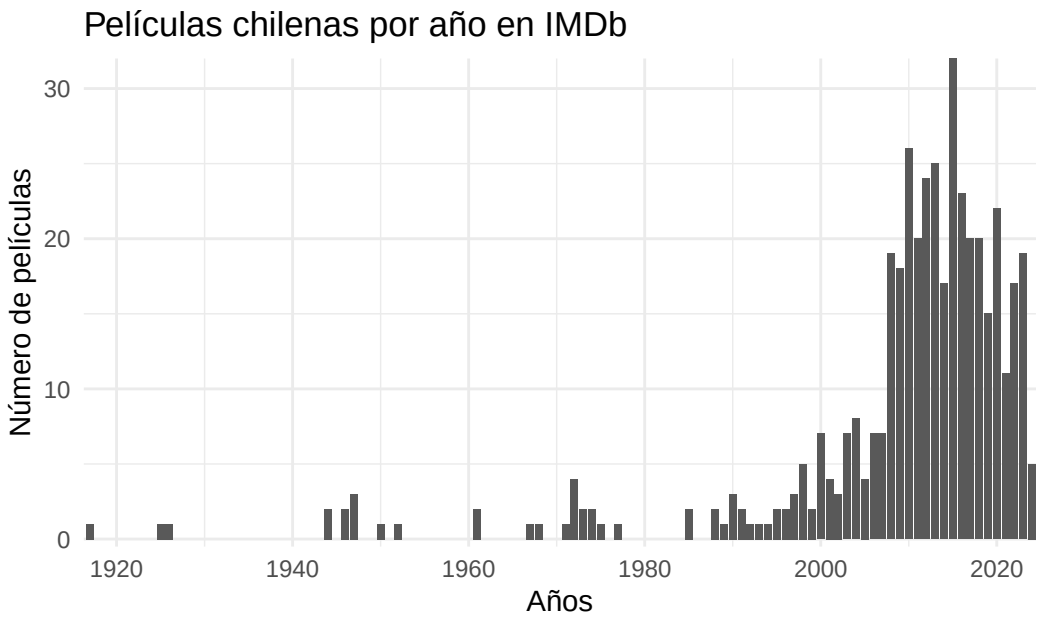
2.2.5 Análisis

2.2.5.1 Número de películas

3) Graficar la evolución del el *número de películas* (eje y) estrenadas por *año* (eje x).

```
df_movies_anio_all <- df_movies |>
  count(anio, name = 'n_peliculas')

df_movies_anio_all |>
  f_gg_peliculas_anio()
```



Fuente: IMDb.com. Web Scraping y acceso a datos desde la web

2.2.5.2 Rating

5) Graficar la evolución del el *rating IMDb* promedio (eje y) *estrenadas desde 1990* a la fecha (eje x).

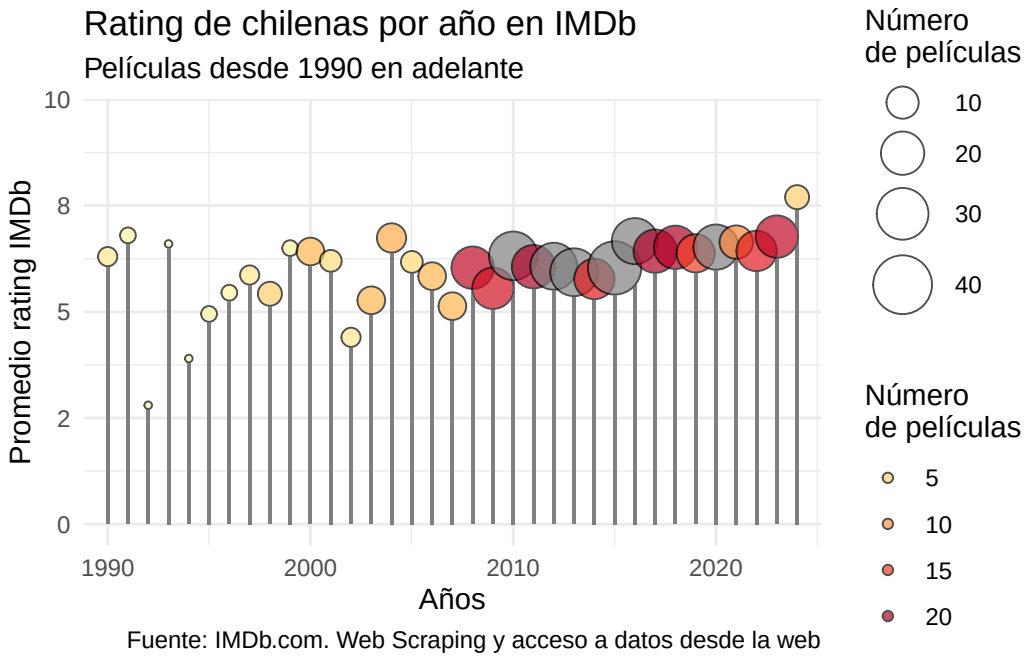
```
df_movies_rank <- df_movies |>
  f_df_peliculas_rank(ratingsSummary_aggregateRating)

df_movies_rank |>
  head()
```

```
# A tibble: 6 x 3
  anio       n_peliculas rating_imdb
  <date>           <int>       <dbl>
1 1990-01-01           3         6.3
2 1991-01-01           2         6.8
3 1992-01-01           1         2.8
4 1993-01-01           1         6.6
5 1994-01-01           1         3.9
```

```
6 1995-01-01           2        4.95
```

```
df_movies_rank |>
  f_gg_peliculas_rank()
```



Fuente: IMDb.com. Web Scraping y acceso a datos desde la web

2.2.5.3 Género

6) ¿Cuál es el *género* que tienen el *mejor puntaje promedio* considerando películas estrenadas desde 1990 a la fecha?

Modificar base para que la unidad de análisis sea `genero`.

```
f_df_genero <- function(.df,
                        .rating){
  .df |>
    summarise(n_peliculas = n(),
              n_peliculas_con_rating = sum(!is.na({{ .rating }})),
              rating_imdb = mean({{ .rating }}, na.rm = TRUE),
              .by = genero) |>
    arrange(-rating_imdb)
}

df_movies_genero <- df_movies |>
  filter(anio >= as.Date("1990-01-01"))

df_movies_genero <- df_movies_genero |>
```

```
  select(anio,
         titulo = originalTitleText_text,
         rating_imdb = ratingsSummary_aggregateRating,
         genero = titleGenres) |>
  mutate(genero = map(genero, \(x) pluck(x, 'genres', 1))) |>
  unnest_longer(genero)

df_movies_genero$genero <- df_movies_genero$genero |> unlist() |> as.character()

df_movies_genero_all <- df_movies_genero |>
  f_df_genero(rating_imdb)

head(df_movies_genero_all)
```

```
# A tibble: 6 x 4
  genero      n_peliculas n_peliculas_con_rating rating_imdb
  <chr>             <int>                  <int>       <dbl>
1 Music                 3                      1        7.9
2 Biography             9                      7        7.37
3 Documentary         114                     59        7.23
4 Animation             2                      2        7.15
5 Romance               2                      2        6.8
6 Drama               131                    113        6.16
```

El género con mejor puntaje promedio desde 1990 es **Music**.

```
df_movies_genero_all |>
  arrange(-n_peliculas) |>
  head()
```

```
# A tibble: 6 x 4
  genero      n_peliculas n_peliculas_con_rating rating_imdb
  <chr>             <int>                  <int>       <dbl>
1 Drama               131                    113        6.16
2 Documentary         114                     59        7.23
3 Comedy               62                     51        5.54
4 Action               26                     24        5.28
5 Crime                14                     13        5.93
6 Biography             9                      7        7.37
```