

---

**CURSO:** CC50 – ADMINISTRACION DE LA INFORMACION

**CLASE:** SESION #4 (PRACTICA)

**TEMA:** ADQUISICION, PREPARACION Y VISUALIZACION PRELIMINAR DE DATOS – WEB SCRAPING - CON R / RSTUDIO

**PROFESOR/A:** Ing. PATRICIA REYES SILVA

---

En esta clase, veremos en la práctica, como adquirir datos contenidos en paginas web.

Esta es una actividad que se realiza durante la primera fase del ciclo de vida de los datos: **Fase 1 - Creación de datos.**

Es necesario distinguir que las páginas web están diseñadas para ser leídas por humanos, pero también, por máquinas (programas de ordenador). Así, definiremos los siguientes tipos de acceso:

- **Acceso Humano:** podemos acceder a los datos utilizando técnicas de **rascado/arañado** de datos, lo que en ingles se denomina técnica de **web scraping**. Las páginas web se encuentran por lo general en formato HTML, por ello, accederemos a los datos contenidas en ellas utilizando técnicas de manejo de texto.
- **Acceso Máquina:** mediante el uso de servicios web que permiten el acceso a los datos a través de **APIs**. Son las máquinas las que se sirven de estas APIs y los formatos que comúnmente se utilizan son JSON y XML (los datos son organizados en forma de árbol).

## OBJETIVO PRINCIPAL

Adquirir datos de interés, localizados en distintas fuentes externas (páginas web), utilizando la técnica de web scraping (acceso humano).

## COMPETENCIAS

- Acceder a datos provenientes de páginas web
- Realizar las operaciones básicas de limpieza de los datos
- Archivar el conjunto de datos obtenido en un archivo de tipo .csv
- Visualizar información relevante a partir del conjunto de datos procesado

## ACTIVIDADES

1. Se instalarán los paquetes necesarios en R para realizar Web Scraping.
2. Se desarrollará la exploración y captura de datos de tres páginas web:

**Caso # 1:** Acceso a la página web de la UPC, específicamente a la de la carrera de Ciencias de la Computación.

**Caso # 2:** Acceso a la página web de Wikipedia para identificar datos contenidas en tablas.

**Caso # 3:** Acceso a la página web del IMDb, con el objetivo de capturar los 100 largometrajes más populares lanzados en el 2020, y hacer un pre-procesado y visualización grafica de los

datos obtenidos. Finalmente, el conjunto de datos resultante se almacenará en un archivo de tipo csv dentro del directorio de trabajo.

## INSTRUCCIONES EN R / R STUDIO – WEB SCRAPING

Se realizarán las siguientes tareas desde la consola en R:

En R, instalamos los siguientes paquetes: rvest, xml2 y XML

```
> install.packages("rvest")  
> install.packages("xml2")  
> install.packages("XML")
```

Accedemos a las librerías previamente descargadas:

```
> library(rvest)  
> library(xml2)  
> library(XML)
```

**Caso # 1:** Accederemos a la pagina web de la UPC, específicamente a la de la carrera de Ciencias de la Computación

URL: <https://pregrado.upc.edu.pe/facultad-de-ingenieria/ciencias-de-la-computacion/>



En la consola de R exploramos dicha pagina

a) Creamos la variable con el URL

```
> upc_url <- 'https://pregrado.upc.edu.pe/facultad-de-ingenieria/ciencias-de-la-computacion/'
```

b) Leemos cada línea de la pagina

```
> upc_read <- readLines(upc_url, encoding = "UTF-8", warn = FALSE)
```

c) Analizamos el contenido de la pagina

```
> parsed_upc <- htmlParse(upc_read, encoding = "UTF-8")
```

d) Identificamos los 'párrafos' de la pagina

```
> upc_enter_text <- parsed_upc["//p"]
```

e) Averiguamos cuantos párrafos existen en la pagina

```
> length(upc_enter_text)
[1] 42
```

f) Visualizamos el contenido de alguno de los párrafos

```
> upc_enter_text [[10]]
<p class="campoLaboral fuente-zizou-light mt-2" data-search-content="2">Te
podrás desempeñar en cargos como DIRECTOR, ANALISTA, LÍDER DE PROYECTOS, C
ONSULTOR en las siguientes especializaciones:
</p>
```

g) Averiguamos cuantos enlaces tiene la pagina

```
> length(getHTMLLinks(upc_read))
[1] 65
```

h) Averiguamos cuantas tablas tiene la pagina

```
> length((readHTMLTable(upc_read)))
[1] 0
```

**Caso # 2:** Como no hemos obtenido ninguna tabla en la anterior página (la de la UPC), cambiaremos de URL y accederemos a Wikipedia, a este enlace en particular:

<https://es.wikipedia.org/wiki/Ayuda:Tablas>

Y ejecutaremos las mismas instrucciones que para la URL de la UPC.

```
> wiki_url <- 'https://es.wikipedia.org/wiki/Ayuda:Tablas'
> wiki_read <- readLines(wiki_url, encoding = "UTF-8", warn = FALSE)
> parsed_wiki <- htmlParse(wiki_read, encoding = "UTF-8")
> wiki_intro_text <- parsed_wiki["//p"]
> length(wiki_intro_text)
[1] 150
> length(getHTMLLinks(wiki_read))
[1] 255
> length((readHTMLTable(wiki_read)))
[1] 104
```

En esta pagina observamos que si hay tablas.

```
> names(readHTMLTable(wiki_read))
[1] "NULL"
[3] "NULL"
[5] "NULL"
[7] "NULL"
[9] "NULL"
"NULL"
"NULL"
"TÍTULO: (wikitable)\n"
"NULL"
"NULL"
```

Observamos que existen nombres de tablas con valor “NULL”, así que visualizaremos el contenido de alguna cuyo nombre sea distinto a “NULL”:

$$\frac{1}{2}$$

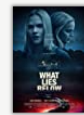


## Feature Film, Released between 2020-01-01 and 2020-12-31 (Sorted by Popularity Ascending)

1-100 of 9,889 titles. | [Next >](#)

View Mode: [Compact](#) | [Detailed](#)

Sort by: [Popularity](#) | [A-Z](#) | [User Rating](#) | [Number of Votes](#) | [US Box Office](#) | [Runtime](#) | [Year](#) | [Release Date](#) | [Date of Your Rating](#) | [Your Rating](#)



### 1. [What Lies Below](#) (2020)

TV-MA | 87 min | Horror, Mystery, Sci-Fi

★ 4.4

☆ [Rate this](#)

**35** Metascore

A 16 year-old girl returns home from camp and learns that her mother has a new boyfriend, one she intends to marry; a man whose charm, intelligence and beauty make him look like he's not human at all.

Director: [Braden R. Duemmler](#) | Stars: [Ema Horvath](#), [Troy Iwata](#), [Danny Corbo](#), [Mena Suvari](#)

Votes: 3,412



### 2. [The Father](#) (I) (2020)

PG-13 | 97 min | Drama

★ 8.3

☆ [Rate this](#)

**88** Metascore

A man refuses all assistance from his daughter as he ages. As he tries to make sense of his changing circumstances, he begins to doubt his loved ones, his own mind and even the fabric of his reality.

Director: [Florian Zeller](#) | Stars: [Anthony Hopkins](#), [Olivia Colman](#), [Mark Gatiss](#), [Olivia Williams](#)

Votes: 26,306



### 3. [Concrete Cowboy](#) (2020)

R | 111 min | Drama

★ 6.2

☆ [Rate this](#)

**67** Metascore

Sent to live with his estranged father for the summer, a rebellious teen finds kinship in a tight-knit Philadelphia community of Black cowboys.

Director: [Ricky Staub](#) | Stars: [Idris Elba](#), [Caleb McLaughlin](#), [Lorraine Toussaint](#), [Jharrel Jerome](#)

Votes: 3,600



### 4. [Run](#) (I) (2020)

PG-13 | 90 min | Mystery, Thriller

★ 6.7

☆ [Rate this](#)

**67** Metascore

A homeschooled teenager begins to suspect her mother is keeping a dark secret from her.

#### a) Cargamos los paquetes rvest y xml2

```
> library(xml2)
> library(rvest)
```

#### b) Especificamos la URL que deseamos analizar, utilizando la instrucción read\_html leemos el código HTML

```
> pelis <- read_html("https://www.imdb.com/search/title/?count=100&release_date=2020,2020&title_type=feature")
```

#### c) Obtenemos primero la clasificación (ranking de la película).

Ubicamos dentro de la página, el clasificador (tag CSS) que deseamos consultar. Posicionamos el puntero del mouse sobre el "1" de la primera película y con el botón derecho damos clic en 'inspect'. Al lado derecho de nuestro navegador se abrirá la ventana del desarrollador y el cursor se ubicará en el tag deseado.



```

<div class="sorting"></div>
<br class="clear">
<div class="listner list detail sub-list">
  <div class="listner-list">
    <div class="listner-item mode-advanced">
      <div class="listner-top-right"></div>
      <div class="listner-item-image float-left"></div>
      <div class="listner-item-content">
        <h3 class="listner-item-header">
          <span class="listner-item-index unbold text-primary">1.
            </span>
            <a href="/title/tt9264728/?ref=adv_li_tt">What Lies Below</a>
            <span class="listner-item-year text-muted unbold">(2020)
          </span>
        </h3>
        <p class="text-muted"></p>
        <div class="ratings-bar">
          <div class="inline-block ratings-imdb-rating" name="ir"
      
```

Observamos que el ranking de la película #1 se muestra en **“.text-primary”**

- Usamos el selector CSS para raspar la seccion de Ranking. A la instrucción `html_nodes` le especificamos que selector CSS queremos recuperar.

```
> rank_data_html <- html_nodes(pelis, '.text-primary')
```

- Convertimos a texto los datos de ranking recuperados y los visualizamos:

```

> rank_data <- html_text(rank_data_html)
> head(rank_data)
[1] "1." "2." "3." "4." "5." "6."

```

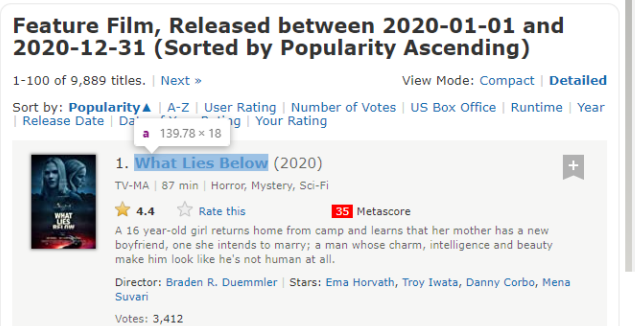
- Pre-procesamos estos datos convirtiendolos a numericos

```

> rank_data<-as.numeric(rank_data)
> head(rank_data)
[1] 1 2 3 4 5 6

```

#### d) Obtenemos ahora los títulos de la película.



```

<div class="nav"></div>
<br class="clear">
<div class="sorting"></div>
<br class="clear">
<div class="listner list detail sub-list">
  <div class="listner-list">
    <div class="listner-item mode-advanced">
      <div class="listner-top-right"></div>
      <div class="listner-item-image float-left"></div>
      <div class="listner-item-content">
        <h3 class="listner-item-header">
          <span class="listner-item-index unbold text-primary">1.
            </span>
            <a href="/title/tt9264728/?ref=adv_li_tt">What Lies Below</a>
            <span class="listner-item-year text-muted unbold">(2020)
          </span>
        </h3>
      
```

- Constatamos que el selector se llama **“.listner-item-header a”**, entonces, obtenemos los datos correspondientes a ese clasificador con la instrucción `html_nodes()` y vemos su contenido.

```

> tit_data_html <- html_nodes(pelis, '.listner-item-header a')
> tit_data <- html_text(tit_data_html)
> head(tit_data)
[1] "What Lies Below" "The Father" "Concrete Cowboy" "Run" "Promising Young Woman"
[6] "Wonder Woman 1984"

```

#### e) Obtenemos la descripción de la película

```
> desc_data_html <- html_nodes(pelis, '.ratings-bar+ .text-muted')
> desc_data <- html_text(desc_data_html)
> head(desc_data)
[1] "\n    A 16 year-old girl returns home from camp and learns that her mother has a new boyfriend, one she intends to marry; a man whose charm, intelligence and beauty make him look like he's not human at all."
[2] "\n    A man refuses all assistance from his daughter as he ages. As he tries to make sense of his changing circumstances, he begins to doubt his loved ones, his own mind and even the fabric of his reality."
[3] "\n    Sent to live with his estranged father for the summer, a rebellious teen finds kinship in a tight-knit Philadelphia community of Black cowboys."
[4] "\n    A homeschooled teenager begins to suspect her mother is keeping a dark secret from her."
[5] "\n    A young woman, traumatized by a tragic event in her past, seeks out vengeance against those who crossed her path."
[6] "\n    Diana must contend with a work colleague and businessman, whose desire for extreme wealth sends the world down a path of destruction, after an ancient artifact that grants wishes goes missing."
```

➤ **Pre-procesado:** Eliminamos "\n" en cada párrafo.

```
> desc_data <- gsub("\n", "", desc_data)
> head(desc_data)
```

```
[1] " A 16 year-old girl returns home from camp and learns that her mother has a new boyfriend, one she intends to marry; a man whose charm, intelligence and beauty make him look like he's not human at all."
[2] " A man refuses all assistance from his daughter as he ages. As he tries to make sense of his changing circumstances, he begins to doubt his loved ones, his own mind and even the fabric of his reality."
[3] " Sent to live with his estranged father for the summer, a rebellious teen finds kinship in a tight-knit Philadelphia community of Black cowboys."
[4] " A homeschooled teenager begins to suspect her mother is keeping a dark secret from her."
[5] " A young woman, traumatized by a tragic event in her past, seeks out vengeance against those who crossed her path."
[6] " Diana must contend with a work colleague and businessman, whose desire for extreme wealth sends the world down a path of destruction, after an ancient artifact that grants wishes goes missing."
```

#### f) Obtenemos la duración de la película

```
> runtime_data_html <- html_nodes(pelis, '.text-muted .runtime')
> runtime_data <- html_text(runtime_data_html)
> head(runtime_data)
[1] "87 min" "97 min" "111 min" "90 min" "113 min" "151 min"
```



- **Pre-procesado:** Eliminamos “min” en cada elemento de la lista y lo convertimos a numerico.

```
> runtime_data<-gsub(" min","",runtime_data)
> runtime_data<-as.numeric(runtime_data)
> head(runtime_data)
[1] 87 97 111 90 113 151
```

#### g) Obtenemos el género de la película

```
> genre_data_html <- html_nodes(pelis, '.genre')
> genre_data <- html_text(genre_data_html)
> head(genre_data)
[1] "\nHorror, Mystery, Sci-Fi          "      "\nDrama          "
"\nDrama          "
[4] "\nMystery, Thriller          "      "\nCrime, Drama, Thriller
"      "\nAction, Adventure, Fantasy
```

- **Pre-procesado:** Eliminamos “\n” en cada elemento de la lista y lo convertimos a numerico.

```
> genre_data<-gsub("\n","",genre_data)
> head(genre_data)
[1] "Horror, Mystery, Sci-Fi          "      "Drama          "
"Drama          "
[4] "Mystery, Thriller          "      "Crime, Drama, Thriller
"      "Action, Adventure, Fantasy
```

- **Pre-procesado:** Eliminamos los espacios en blanco en exceso.

```
> genre_data<-gsub(" ", "", genre_data)
> head(genre_data)
[1] "Horror,Mystery,Sci-Fi"      "Drama"      "Drama"
"Mystery,Thriller"      "Crime,Drama,Thriller"
[6] "Action,Adventure,Fantasy"
```

- **Pre-procesado:** Tomamos solo el primer genero de cada pelicula.

```
> genre_data<-gsub(",.*","",genre_data)
> head(genre_data)
[1] "Horror" "Drama" "Drama" "Mystery" "Crime" "Action"
```

- **Pre-procesado:** Convertimos cada texto de genero a factor.

```
> genre_data<-as.factor(genre_data)
> head(genre_data)
[1] Horror Drama Drama Mystery Crime Action
Levels: Action Adventure Animation Biography Comedy Crime Drama Horror Mys
tery Thriller
```

#### h) Obtenemos la calificación de la película

```
> rating_data_html <- html_nodes(pelis, '.ratings-imdb-rating strong')
```

```
> rating_data <- html_text(rating_data_html)
> head(rating_data)
[1] "4.4" "8.3" "6.2" "6.7" "7.5" "5.4"
```

➤ **Pre-procesado:** Convertimos cada texto a numero.

```
> rating_data<-as.numeric(rating_data)
> head(rating_data)
[1] 4.4 8.3 6.2 6.7 7.5 5.4
```

**i) Obtenemos el metascor de la película**

```
> metascor_data_html <- html_nodes(pelis, '.metascor')
> metascor_data <- html_text(metascor_data_html)
> head(metascor_data)
[1] "35" "88" "67" "67" "73" "60"
```

➤ **Pre-procesado:** Eliminamos los espacios en blanco en exceso y convertimos a numerico.

```
> metascor_data<-gsub(" ","",metascor_data)
> head(metascor_data)
[1] "35" "88" "67" "67" "73" "60"
> length(metascor_data)
[1] 89
> metascor_data<-as.numeric(metascor_data)
> head(metascor_data)
[1] 35 88 67 67 73 60
```

**j) Obtenemos los votos obtenidos por la película**

```
> votos_data_html <- html_nodes(pelis, '.sort-num_votes-visible span:nth-child(2)')
> votos_data <- html_text(votos_data_html)
> head(votos_data)
[1] "3,412" "26,306" "3,600" "39,210" "66,890" "189,922"
```

➤ **Pre-procesado:** Removemos las comas y convertimos a numerico

```
> votos_data<-gsub(",","",votos_data)
> votos_data<-gsub(",","",votos_data)
> head(votos_data)
[1] "3412" "26306" "3600" "39210" "66890" "189922"
```

**k) Obtenemos el Gross\_Earning\_in\_Mil de la película**

```
> gross_data_html <- html_nodes(pelis, '.ghost~ .text-muted+ span')
> gross_data <- html_text(gross_data_html)
> head(gross_data)
[1] "$53.80M" "$84.16M" "$64.91M" "$146.07M" "$61.56M" "$206.31M"
```

➤ **Pre-procesado:** Eliminamos los signos \$ y M y convertimos a numerico.

```
> gross_data<-gsub("M","",gross_data)
> gross_data<-substring(gross_data,2,6)
> head(gross_data)
[1] "53.80" "84.16" "64.91" "146.0" "61.56" "206.3"
```

```
> length(gross_data)
[1] 7
> gross_data<-as.numeric(gross_data)
> head(gross_data)
[1] 53.80 84.16 64.91 146.00 61.56 206.30
```

#### l) Obtenemos el director de la película

```
> director_data_html <- html_nodes(pelis, '.text-muted+ p a:nth-child(1)')
> director_data <- html_text(director_data_html)
> head(director_data)
[1] "Braden R. Duemmler" "Florian Zeller" "Ricky Staub" "Aneesh
Chaganty" "Emerald Fennell" "Patty Jenkins"
```

➤ **Pre-procesado:** Convertimos cada director a factor.

```
> director_data<-as.factor(director_data)
> head(director_data)
[1] Braden R. Duemmler Florian Zeller Ricky Staub Aneesh Chagan
ty Emerald Fennell Patty Jenkins
100 Levels: Aaron Schneider Aaron Sorkin Adil El Arbi Anders Thomas Jensen
Andy Goddard Aneesh Chaganty ... Zoe Lister-Jones
```

#### m) Obtenemos el actor de la película

```
> actor_data_html <- html_nodes(pelis, '.lister-item-content .ghost+ a')
> actor_data <- html_text(actor_data_html)
> head(actor_data)
[1] "Ema Horvath" "Anthony Hopkins" "Idris Elba" "Sarah Paulson"
"Carey Mulligan" "Gal Gadot"
```

➤ **Pre-procesado:** Convertimos cada actor a factor.

```
> actor_data<-as.factor(actor_data)
> head(actor_data)
[1] Ema Horvath Anthony Hopkins Idris Elba Sarah Paulson Carey
Mulligan Gal Gadot
97 Levels: Amy Adams Andrea Riseborough Andy Samberg Anna Kendrick Anna Ma
ria Sieklucka Anne Hathaway Anthony Hopkins ... Yolanda Kettle
```

Luego de concluir con el rescado o scraping de los 11 atributos de esta pagina web, debemos haber obtenido los siguientes objetos:

Environment

History

Connections

📁

📄

📊 Import Dataset

🔑

Global Environment

🔍

Data

▶ actor_data_html	List of 100	🔍
▶ desc_data_html	List of 99	🔍
▶ director_data_html	List of 100	🔍
▶ genre_data_html	List of 100	🔍
▶ gross_data_html	List of 7	🔍
▶ metascore_data_html	List of 89	🔍
▶ pelis	List of 2	🔍
▶ rank_data_html	List of 100	🔍
▶ rating_data_html	List of 99	🔍
▶ runtime_data_html	List of 100	🔍
▶ tit_data_html	List of 100	🔍
▶ votos_data_html	List of 99	🔍

Values

actor_data	Factor w/ 97 levels "Amy Adams","Andrea Riseborough",...: 26 ...
desc_data	chr [1:99] " A 16 year-old girl returns home from camp and l...
director_data	Factor w/ 100 levels "Aaron Schneider",...: 14 39 79 6 36 71 ...
genre_data	Factor w/ 10 levels "Action","Adventure",...: 8 7 7 9 6 1 7 1...
gross_data	num [1:7] 53.8 84.2 64.9 146 61.6 ...
metascore_data	num [1:89] 35 88 67 67 73 60 93 69 76 89 ...
rank_data	num [1:100] 1 2 3 4 5 6 7 8 9 10 ...
rating_data	num [1:99] 4.4 8.3 6.2 6.7 7.5 5.4 7.5 7.5 7.8 7.6 ...
runtime_data	num [1:100] 87 97 111 90 113 151 107 150 129 115 ...
tit_data	chr [1:100] "what Lies Below" "The Father" "Concrete Cowboy"...
votos_data	chr [1:99] "3412" "26306" "3600" "39210" "66890" "189922" "5...

Podemos observar que ciertos atributos no tienen valores y son aquellos cuya cantidad de observaciones es menor a 100 (`desc_data_html`, `gross_data_html`, `metascore_data_html`, `rating_data_html` y `votos_data_html`).

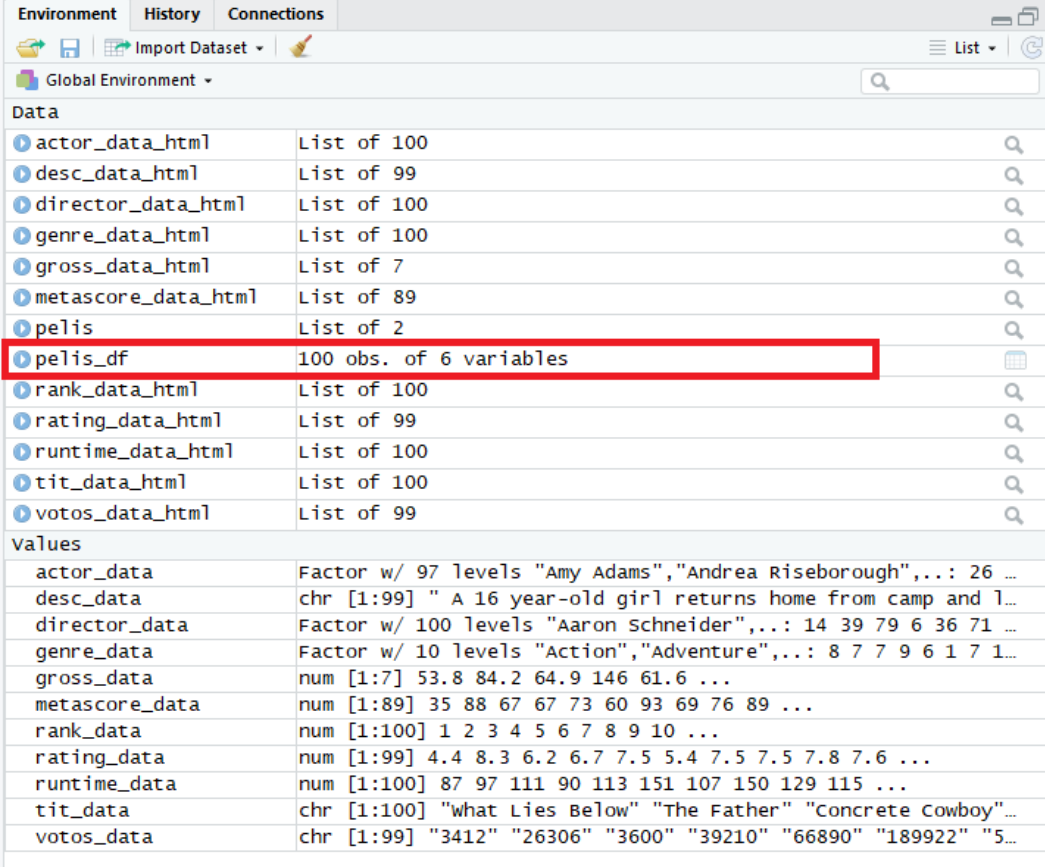
Los datos entonces no están completos para estos atributos. En un primer momento, no consideraremos dichos atributos para un futuro análisis mientras no se defina que valor asignarles.

## Almacenamiento del dataset obtenido

Este conjunto de datos lo podemos guardar en un **data.frame()** con la siguiente instrucción:

```
> pelis_df<-data.frame(Rank = rank_data, Titulo = tit_data, Runtime = runtime_data,
  Genero = genre_data, Director = director_data, Actor = actor_data)
```

Un **data.frame()** solo puede crearse con atributos que contengan la misma cantidad de observaciones, por ello, se ha considerado aquellos atributos que tienen completas las 100 observaciones (sólo 6 atributos).



Global Environment		
Data		
actor_data_html	List of 100	
desc_data_html	List of 99	
director_data_html	List of 100	
genre_data_html	List of 100	
gross_data_html	List of 7	
metascore_data_html	List of 89	
pelis	List of 2	
pelis_df	100 obs. of 6 variables	
rank_data_html	List of 100	
rating_data_html	List of 99	
runtime_data_html	List of 100	
tit_data_html	List of 100	
votos_data_html	List of 99	
Values		
actor_data	Factor w/ 97 levels "Amy Adams","Andrea Riseborough",...: 26 ...	
desc_data	chr [1:99] " A 16 year-old girl returns home from camp and l...	
director_data	Factor w/ 100 levels "Aaron Schneider",...: 14 39 79 6 36 71 ...	
genre_data	Factor w/ 10 levels "Action","Adventure",...: 8 7 7 9 6 1 7 1...	
gross_data	num [1:7] 53.8 84.2 64.9 146 61.6 ...	
metascore_data	num [1:89] 35 88 67 67 73 60 93 69 76 89 ...	
rank_data	num [1:100] 1 2 3 4 5 6 7 8 9 10 ...	
rating_data	num [1:99] 4.4 8.3 6.2 6.7 7.5 5.4 7.5 7.5 7.8 7.6 ...	
runtime_data	num [1:100] 87 97 111 90 113 151 107 150 129 115 ...	
tit_data	chr [1:100] "what Lies Below" "The Father" "Concrete Cowboy"...	
votos_data	chr [1:99] "3412" "26306" "3600" "39210" "66890" "189922" "5...	

La estructura del dataframe (que se convertirá en nuestro dataset) es el siguiente:

```
> str(pelis_df)
'data.frame': 100 obs. of 6 variables:
 $ Rank : num 1 2 3 4 5 6 7 8 9 10 ...
 $ Titulo : Factor w/ 100 levels "365 dni","A Quiet Place Part II",...: 98
80 14 64 60 100 54 74 89 48 ...
 $ Runtime : num 87 97 111 90 113 151 107 150 129 115 ...
 $ Genero : Factor w/ 10 levels "Action","Adventure",...: 8 7 7 9 6 1 7 1
7 7 ...
 $ Director: Factor w/ 100 levels "Aaron Schneider",...: 14 39 79 6 36 71 1
9 21 2 60 ...
 $ Actor : Factor w/ 97 levels "Amy Adams","Andrea Riseborough",...: 26 7
38 79 15 31 30 45 24 83 ...
```

Guardamos este conjunto de datos en un archivo .csv llamado **pelis\_df.csv**

```
> write.csv(pelis_df, 'pelis_df.csv', row.names = TRUE)
```

Observamos que el archivo se creo dentro de nuestro directorio de trabajo.



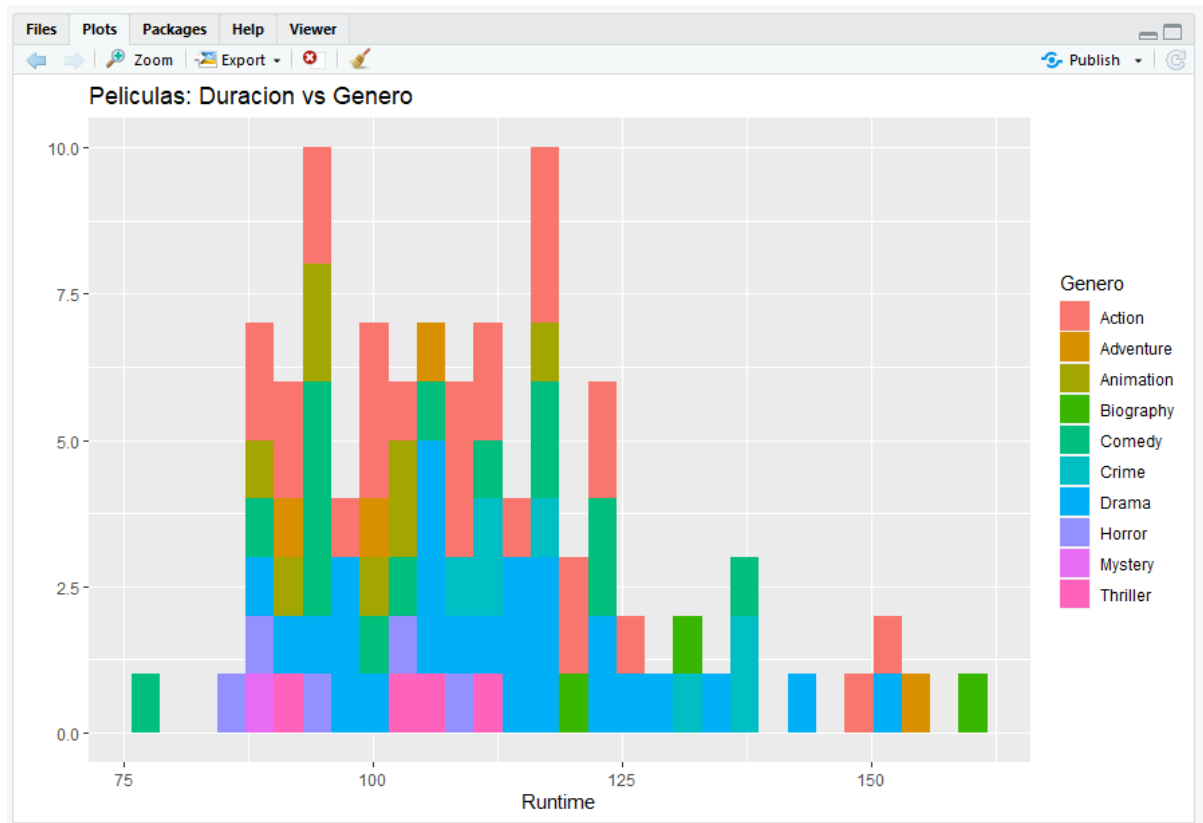
Y en Excel, abrimos el archivo:

	A	B	C	D	E	F	G	H	I
1		Rank	Titulo	Runtime	Genero	Director	Actor		
2	1	1	What Lies Be	87	Horror	Braden R. Du	Ema Horvath		
3	2	2	The Father	97	Drama	Florian Zelle	Anthony Hopkins		
4	3	3	Concrete Co	111	Drama	Ricky Staub	Idris Elba		
5	4	4	Run	90	Mystery	Aneesh Chag	Sarah Paulson		
6	5	5	Promising Yc	113	Crime	Emerald Fen	Carey Mulligan		
7	6	6	Wonder Wor	151	Action	Patty Jenkin	Gal Gadot		
8	7	7	Nomadland	107	Drama	Chloé Zhao	Frances McDormand		
9	8	8	Tenet	150	Action	Christopher	John David Washington		
10	9	9	The Trial of t	129	Drama	Aaron Sorkin	Eddie Redmayne		
11	10	10	Minari	115	Drama	Lee Isaac Chi	Steven Yeun		
12	11	11	I Care a Lot	118	Comedy	J Blakeson	Rosamund Pike		
13	12	12	365 dni	114	Drama	Barbara Bialc	Anna Maria Sieklucka		
14	13	13	Soul	100	Animation	Pete Docter	Jamie Foxx		
15	14	14	Hasta el cielo	121	Action	Daniel Calpa	Miguel Herrán		
16	15	15	Unhinged	90	Action	Derrick Borte	Russell Crowe		
17	16	16	The Dry	117	Crime	Robert Conn	Eric Bana		
18	17	17	Druk	117	Comedy	Thomas Vint	Mads Mikkelsen		
19	18	18	News of the	118	Action	Paul Greengl	Tom Hanks		
20	19	19	Palm Springs	90	Comedy	Max Barbakc	Andy Samberg		
21	20	20	Mank	131	Biography	David Finche	Gary Oldman		
22	21	21	The Courier	112	Thriller	Dominic Co	Benedict Cumberbatch		
23	22	22	Monster Hur	103	Action	Paul W.S. An	Milla Jovovich		
24	23	23	Shiva Baby	77	Comedy	Emma Selign	Rachel Sennott		
25	24	24	Antebellum	105	Drama	Gerard Bush	Janelle Monáe		
26	25	25	Pieces of a V	126	Drama	Kornél Munc	Vanessa Kirby		
27	26	26	Ma Rainey's	94	Drama	George C. W	Viola Davis		
28	27	27	Aves de pres	109	Action	Cathy Yan	Margot Robbie		
29	28	28	Honest Thief	99	Action	Mark William	Liam Neeson		
30	29	29	The Devil All	138	Crime	Antonio Can	Tom Holland		
31	30	30	Hillbilly Eleg	116	Drama	Ron Howard	Amy Adams		
32	31	31	Bill & Ted Fa	91	Adventure	Dean Parisot	Keanu Reeves		
33	32	32	Greenland	119	Action	Ric Roman W	Gerard Butler		
34	33	33	El hombre in	124	Drama	Leigh Whann	Elisabeth Moss		
35	34	34	One Night in	114	Drama	Regina King	Kingsley Ben-Adir		
36	35	35	Retfærdighe	116	Action	Anders Thon	Mads Mikkelsen		
37	36	36	Enola Holme	123	Action	Harry Bradbe	Millie Bobby Brown		

## Visualización de los datos

1) A continuación, creamos dos visualizaciones con los datos obtenidos y podremos inferir algunos resultados.

```
> library('ggplot2')
> qplot(data = pelis_df, Runtime, fill = Genero, bins = 30, main="Película: D
uración vs Género")
```



**Pregunta:** ¿Que podemos inferir a partir de esta visualización?

¿Qué géneros de películas tienen una duración menor a 100 minutos? ¿Cuáles más de 120 minutos?

2) Totalizamos las películas por Género y visualizamos el resultado en forma de tabla y en grafico de barras

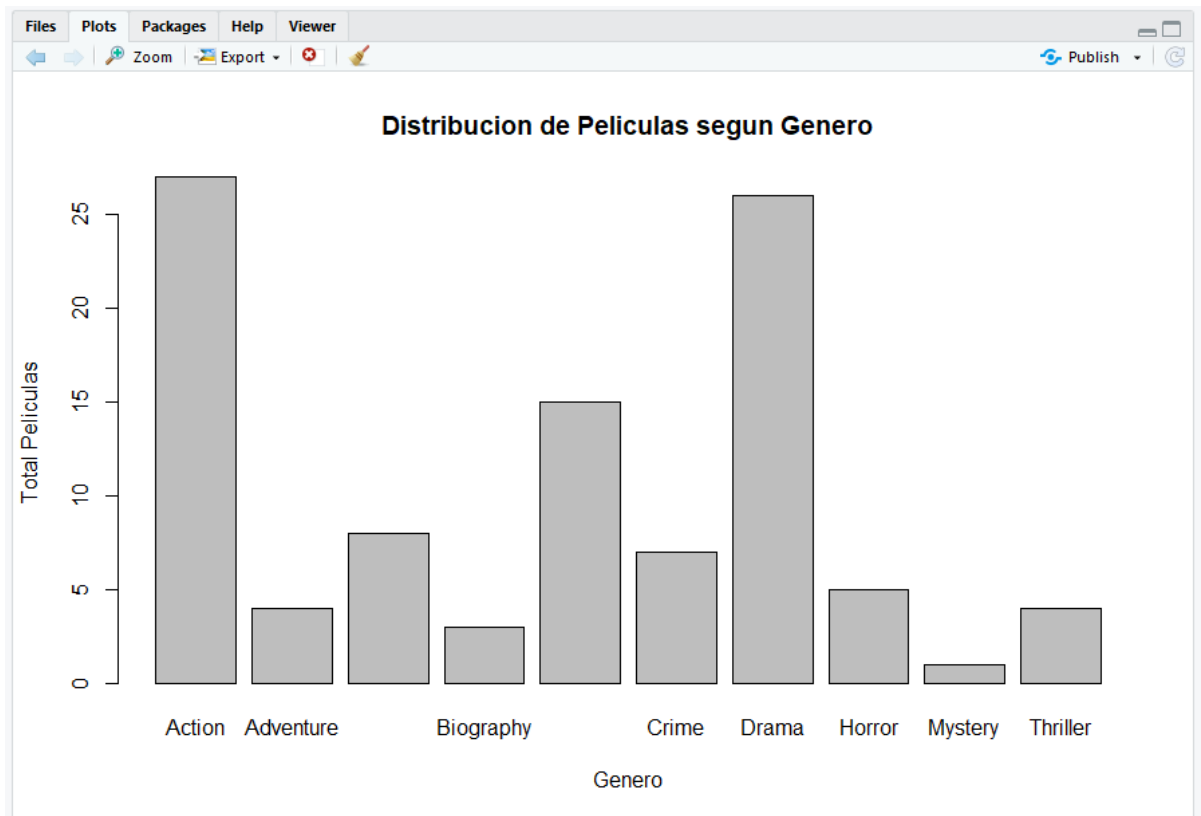
```
> tot_pelis <- table(pelis_df$Genero)
> tot_pelis
```

Genero	Count
Action	27
Adventure	1
Animation	4
Biography	8
Comedy	15
Crime	7
Drama	26
Horror	5
Mystery	4
Thriller	3

```
> head(tot_pelis)
```

Genero	Count
Action	27
Adventure	1
Animation	4
Biography	8
Comedy	15
Crime	7

```
> barplot(tot_pelis,main="Distribucion de Peliculas segun Genero",xlab="Ge  
nero",ylab="Total Peliculas")
```



**Pregunta:** ¿Qué genero de película fue la que tuvo mayores estrenos? ¿Cuál genero fue el menos producido?