Universidad Abierta de Cataluña
Tipología y ciclo de vida de los datos
8 de enero de 2018

Práctica 2. Limpieza de datos.

Alumnas:
Raquel Martín de Consuegra Domínguez
Marta García González

El objetivo de esta actividad será el tratamiento de un dataset, que puede ser el creado en la práctica 1 o cualquier dataset libre disponible en Kaggle (https://www.kaggle.com). Las diferentes tareas a realizar (y justificar) son las siguientes:

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

Para esta práctica hemos elegido un dataset de Kaggle (https://www.kaggle.com/tmdb/tmdb-movie-metadata/data) que contiene metadatos de 5.000 películas de la página web TMDb.

Para la industria cinematográfica, es muy importante vaticinar el éxito de una película antes de su lanzamiento. Los productores invierten ingentes cantidades económicas a la hora de lanzar una película y éstas, en ocasiones fracasan. Otras veces, se invierte menos, y las cintas resultan ser un éxito. La temática, el género, el reparto, el argumento, la productora… Muchos factores pueden influir en el éxito o el fracaso.

Los aficionados al cine tienen gustos muy diversos, se podría también predecir qué películas serán calificadas con puntuaciones altas independientemente de que hayan sido un éxito comercial o no.

Con este dataset podríamos lograr predecir este tipo de preguntas. Disponemos de información sobre recaudación, reparto, presupuesto, etc. de miles de películas.

Disponemos de dos datasets, uno sobre las películas y otro sobre los créditos de las mismas. A continuación vamos a describir la información que contiene cada uno de ellos.

Dataset tmdb_5000_movies.csv. Metadatos de películas. No contiene registros vacíos. Contiene 4803 filas y 20 columnas con la siguiente información.

- budget (numérico). Presupuesto.
- genres (string). JSON que contiene los distintos géneros a los que pertenece la película.
- homepage (string). Página web oficial de la película.
- id (numérico). Identificador de la película.
- keywords (string). Palabras clave de la película, tags. Es un JSON.
- original_languaje (string). Lenguaje original con el código del mismo. Ejemplo: en = inglés.
- original_title (string). Título original.
- overview (string). Sinopsis.
- popularity (numérico). Popularidad en cifras de la película.
- production_companies (string). JSON que contiene las productoras.
- production_countries (string). JSON que contiene las ciudades de producción.
- release_date (datetime). Fecha de estreno.
- revenue (numérico). Recaudación.
- runtime (numérico). Tiempo de duración de la película.
- spoken_languages (string). JSON que contiene los lenguajes en los que se ha rodado la película.

- status (string). Estado de la película. SI ha sido estrenada, si está en producción, etc.
- tagline (string). Lema de la película.
- title (string). Título.
- vote_average (numeric). Media de votos.
- vote_count (numeric). Conteo de votos.

Ejemplo:

budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	pro
237000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "name": "space war"}, {"id": 3388, "name": "space colony"}, {"id": 3679, "name": "society"}, {"id": 3801, "name": "space travel"}, {"id": 9685, "name": "futuristic"}, {"id": 9840, "name": "romance"}, {"id": 9882, "name": "space"}, {"id": 9951, "name": "alien"}, {"id": 10148, "name": "tribe"}, {"id": 10158, "name": "alien planet"}, {"id": 10987, "name": "cgi"}, {"id": 11399, "name": "marine"}, {"id": 13065, "name": "soldier"}, {"id": 14643, "name": "battle"}, {"id": 14720, "name": "love affair"}, {"id": 165431, "name": "anti war"}, {"id": 193554, "name": "power relations"}, {"id": 206690, "name": "mind	en	Avatar	In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization.	150.437577	[{"n Film 289 "Tw Fox "id" "Du "id" "Lig Ent 574

production_companies	production_countries	release_date	revenue	runtime	spoken_languages	status	tagline	title	vote_av
[{"name": "Ingenious Film Partners", "id": 289}, {"name": "Twentieth Century Fox Film Corporation", "id": 306}, {"name": "Dune Entertainment", "id": 444}, {"name": "Lightstorm Entertainment", "id": 574}]	[{"iso_3166_1": "US", "name": "United States of America"}, {"iso_3166_1": "GB", "name": "United Kingdom"}]	2009-12-10	2787965087	162	[{"iso_639_1": "en", "name": "English"}, {"iso_639_1": "es", "name": "Espa\u00f1ol"}]	Released	Enter the World of Pandora.	Avatar	

status	tagline	title	vote_average	vote_count
Released	Enter the World of Pandora.	Avatar	7.2	11800

Dataset tmdb_5000_credits.csv. Créditos de películas. No contiene registros vacíos. Contiene 4803 filas y 4 columnas con la siguiente información.

- movie_id (tipo numérico). Identificador numérico de la película.
- title (string). Título de la película.
- cast (string). JSON que contiene el reparto de la película.

- crew (string). JSON que contiene a los participantes de la película (técnicos, etc.)

Ejemplo:

movie_id	title	cast	crew
19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "credit_id": "5602a8a7c3a3685532001c9a", "gender": 2, "id": 65731, "name": "Sam Worthington", "order": 0}, {"cast_id": 3, "character": "Neytiri", "credit_id": "52fe48009251416c750ac9cb", "gender": 1, "id": 8691, "name": "Zoe Saldana", "order": 1}, {"cast_id": 25, "character": "Dr. Grace Augustine", "credit_id": "52fe48009251416c750aca39", "gender": 1, "id": 10205, "name": "Sigourney Weaver", "order": 2}, {"cast_id": 4, "character": "Col. Quaritch", "credit_id": "52fe48009251416c750ac9cf", "gender": 2, "id": 32747, "name": "Stephen Lang", "order": 3}, {"cast_id": 5, "character": "Trudy Chacon", "credit_id": "52fe48009251416c750ac9d3", "gender": 1, "id": 17647, "name":	[{"credit_id": "52fe48009251416c750aca23", "department": "Editing", "gender": 0, "id": 1721, "job": "Editor", "name": "Stephen E. Rivkin"}, {"credit_id": "539c47ecc3a36810e3001f87", "department": "Art", "gender": 2, "id": 496, "job": "Production Design", "name": "Rick Carter"}, {"credit_id": "54491c89c3a3680fb4001cf7", "department": "Sound", "gender": 0, "id": 900, "job": "Sound Designer", "name": "Christopher Boyes"}, {"credit_id": "54491cb70e0a267480001bd0", "department": "Sound", "gender": 0, "id": 900, "job": "Supervising Sound Editor", "name": "Christopher Boyes"}, {"credit_id": "539c4a4cc3a36810c9002101", "department": "Production", "gender": 1, "id": 1262, "job": "Casting", "name": "Mali Finn"}, {"credit_id": "5544ee3b925141499f0008fc", "department": "Sound", "gender": 2, "id": 1729, "job": "Original Music Composer", "name": "James Horner"}, {"credit_id": "52fe48009251416c750ac9c3", "department": "Directing", "gender": 2, "id": 2710, "job": "Director", "name": "James Cameron"}, {"credit_id": "52fe48009251416c750ac9d9", "department": "Writing", "gender": 2, "id": 2710, "job": "Writer", "name": "James Cameron"}, {"credit_id":

2. *Limpieza de los datos.*

2.1. *Selección de los datos de interés a analizar. ¿Cuáles son los campos más relevantes para responder al problema?*

Para el estudio que vamos a realizar, nos interesan los datos sobre los beneficios económicos de las películas, reparto, equipo, así como las puntuaciones de los usuarios y la popularidad. Con estos datos podremos realizar estimaciones sobre qué tipo de películas resultan más atrayentes para el espectador, o cuáles son mejor valoradas (independientemente de los beneficios que recauden).

De los datos que hemos descrito en el punto 1 de los dos datasets disponibles, nos quedaremos con los siguientes:

- id. Identificador numérico de la película.
- title. Título de la película.
- cast. JSON que parsearemos con R y del que obtendremos otro dataset parcial (denominado reparto) que asocie el id de la película con cada uno de los actores que intervienen en la misma.
- crew. JSON que parsearemos con R y del que obtendremos otro dataset parcial (denominado equipo) que asocie el id de la película con cada uno de los miembros del equipo que intervienen en la misma.
- keywords. JSON que parsearemos con R y del que obtendremos otro dataset parcial (denominado keywords) que asocie el id de la película con cada una de las palabras claves asociadas en la misma.

- genres. JSON que parsearemos con R y del que obtendremos otro dataset parcial (denominado generos) que asocie el id de la película con cada uno de los géneros asociados a la misma.
- production_companies. JSON que parsearemos con R y del que obtendremos otro dataset parcial (denominado productoras) que asocie el id de la película con cada una de las palabras productoras asociadas a la misma.
- spoken_languages. JSON que parsearemos con R y del que obtendremos otro dataset parcial (denominado lenguas) que asocie el id de la película con cada uno de los lenguajes que se hablan en la misma.
- production_countries. JSON que parsearemos con R y del que obtendremos otro dataset parcial (denominado países) que asocie el id de la película con cada uno de los países que producen.
- popularity. Popularidad en cifras.
- release_date. Fecha de estreno de la película.
- revenue. Recaudación.
- budget. Presupuesto.
- runtime. Duración.
- vote_average. Media de votos.
- vote_count. Número de votos.
- beneficios. Columna calculada restando a la recaudación el presupuesto.

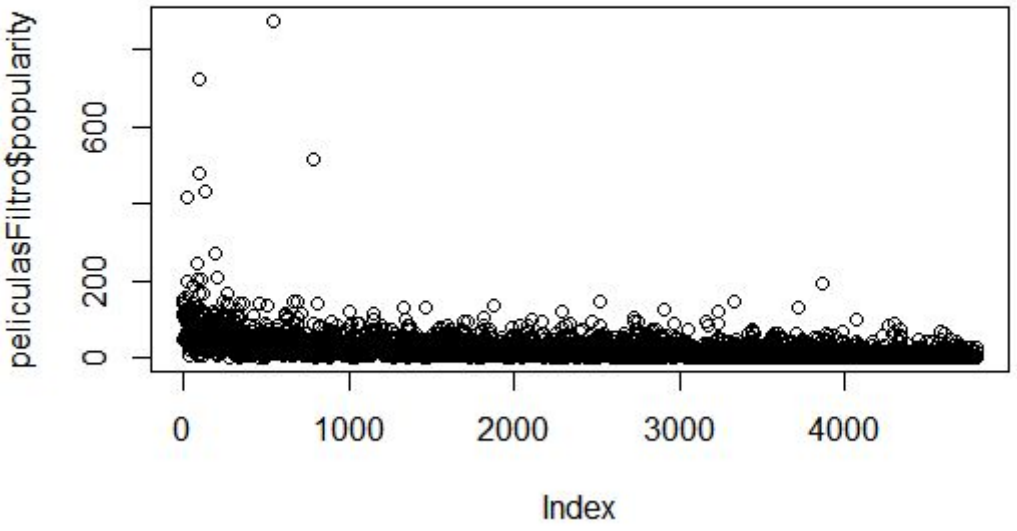
2.2. ¿Los datos contienen ceros o elementos vacíos? ¿Y valores extremos? ¿Cómo gestionarías cada uno de estos casos?

El dataset referente a las películas (tmdb_5000_movies.csv) tiene datos vacíos en algunos de los campos. Estos campos son: homepage, tagline, runtime, release_date y overview.

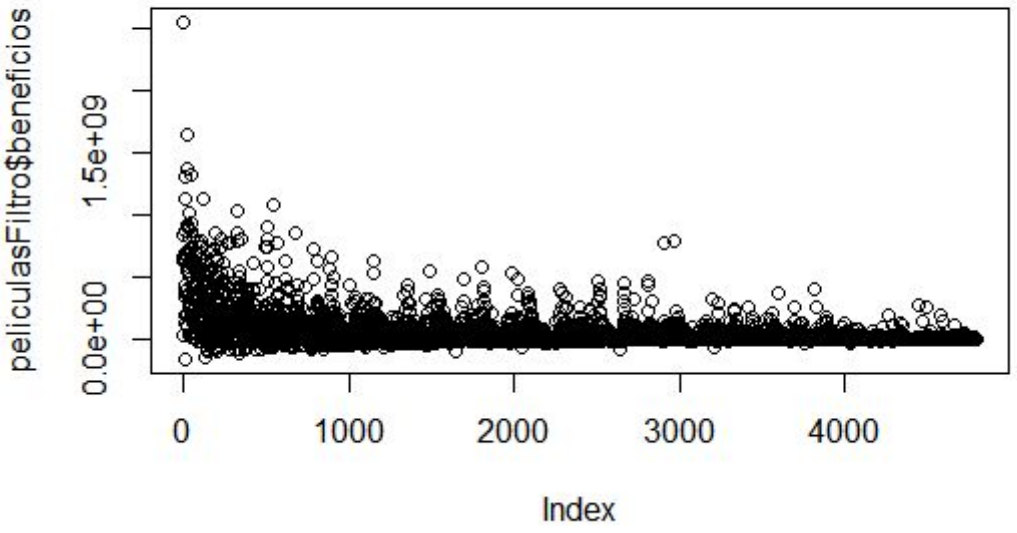
En nuestro caso no va a ser problemática la ausencia de valores en estos campos puesto que no son de importancia para el estudio que vamos a realizar, tal y como hemos indicado en la pregunta anterior. Únicamente nos puede resultar interesante el valor de runtime (duración de la película) y el del release_date (fecha de lanzamiento), pero como estos datos solo faltan en dos registros y en uno, respectivamente, vamos a rellenarlos a través de R.

La fila que no contiene fecha de estreno, la hemos eliminado. En cuanto a la duración de la película en los dos registros que no se indica este dato, hemos rellenado con la media aritmética de esta variable. Al ser pocos registros, este procedimiento es factible y no debería desvirtuar los datos.

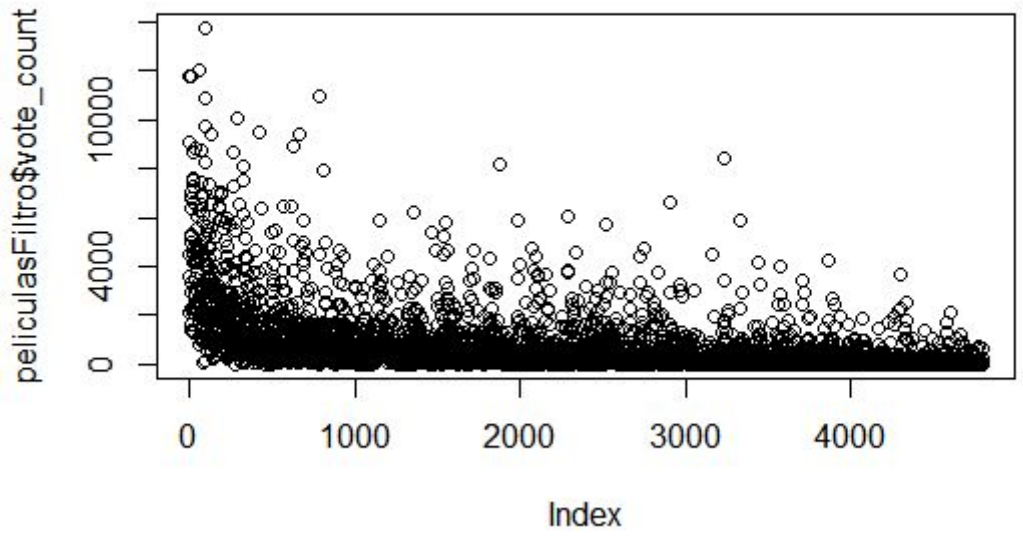
En cuanto a los valores extremos, hemos identificado algunos en el caso de la popularidad. Hay 6 películas que despuntan sobre el resto en cuanto a popularidad.



En cuanto a beneficios observamos un único valor que despunta:



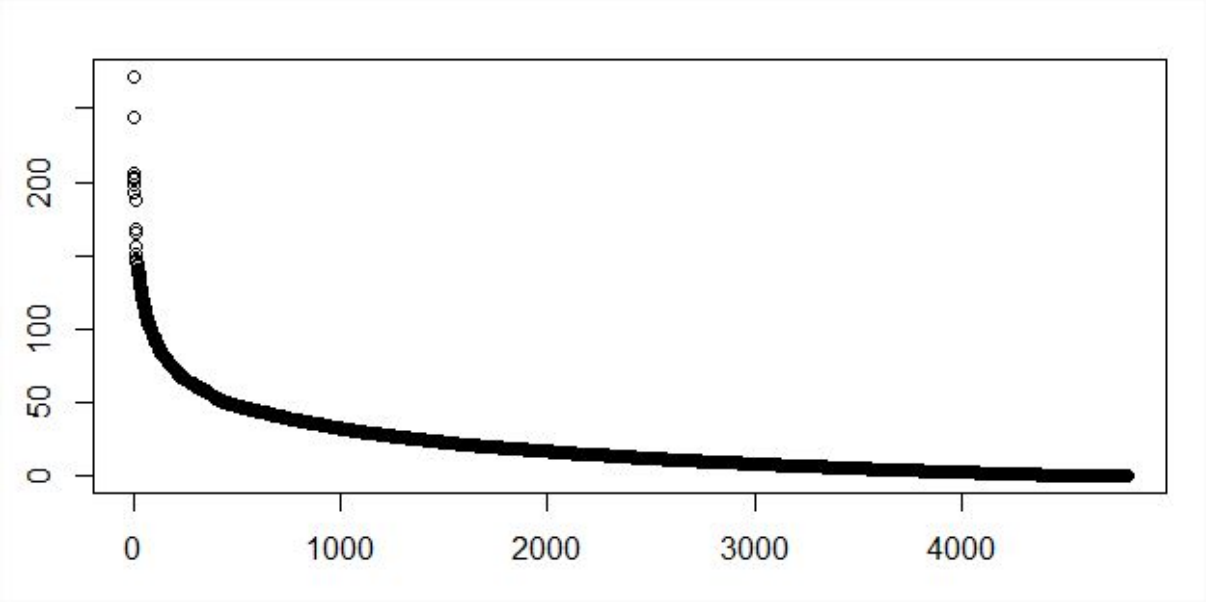
Y en cuanto a número de votos:



El hecho de que existan extreme score en los datos no significa que sean datos erróneos. En el caso de la popularidad por ejemplo nos encontramos con 6 extreme score estos son:

original_title
Minions
Interstellar
Deadpool
Guardians of the Galaxy
Mad Max: Fury Road
Jurassic World

Podemos ver que esos films han sido de gran popularidad y por lo tanto no son errores. Por otro lado si ignoramos estos datos podemos observar que la gráfica cambia a una distribución exponencial



Por este motivo es importante poder detectar los outliers que nos permitan poder hacer modelos más fiables.

El código R empleado para realizar este apartado ha sido el siguiente:

```
# importamos los datasets y contamos si hay nulos en alguna de las variables
peliculas <- read_csv(file="\\tmdb_5000_movies.csv",guess_max = 4803)
peliculasCredits <- read_csv(file="\\tmdb_5000_credits.csv",guess_max= 4803)
sapply(peliculas, function(x)(sum(is.na(x)))) # NA counts
sapply(peliculasCredits, function(x)(sum(is.na(x)))) # NA counts

#Eliminación del registro en el que falta el release_date
posicion <-which(is.na(peliculas$release_date))
registro <-peliculas[posicion,]
peliculasCreditsRegistro <- which(peliculasCredits$movie_id ==registro$id)
peliculasCredits <- peliculasCredits[-c(peliculasCreditsRegistro),]
peliculas <- peliculas[-c(posicion),]

# Media de los valores vacíos en runtime, y relleno de los mismos
summary(peliculas)
posicionRunTime <- which(is.na(peliculas$runtime))
peliculas[posicionRunTime[1],14] <- 106.9
peliculas[posicionRunTime[2],14] <- 106.9

#filtramos el dataset con las columnas de interés y añadimos la columna beneficios
peliculasFiltro <- peliculas[, c(1,4,9,12,13,14,19,20)]

peliculasFiltro <-mutate(peliculasFiltro, beneficios = peliculasFiltro$revenue -
peliculasFiltro$budget)

plot(peliculasFiltro$vote_count)
```

3. *Análisis de los datos.*

3.1. Selección de los grupos de datos que se quieren analizar/comparar.

Los grupos de datos que queremos analizar son los de las películas que han obtenido beneficios y no los han obtenido, para poder ponderar qué tipo de películas son las que más atraen al espectador en función de su reparto, sus productoras, su lenguaje, etc.

Lo que nos interesa analizar son los beneficios, los votos y la popularidad en función de los actores del reparto, el director(es) de la película, el género y la productora.

Para ello, hemos generado cuatro datasets distintos obtenidos a través de los dos datasets de origen.

1. actores: contiene los actores más populares, cuya variable vote_count es mayor de 10. Es decir, descarta los que tienen muy pocos votos. Este dataset contiene las siguientes variables.
    - id: identificador de la película.
    - n: número de películas en las que ha participado el actor.
    - weighted_mean_votos: media de votos ponderada en función al número de películas n y a las variables vote_count y vote_average.
    - weighted_mean_beneficios: media de beneficios ponderada en función a los beneficios y al número de películas n.
    - weighted_mean_popularidad: media de la popularidad ponderada en función a la popularity y al número de películas n.
2. directores: contiene los directores más populares, cuya variable vote_count es mayor de 10. Es decir, descarta los que tienen muy pocos votos. Este dataset contiene las siguientes variables.
    - id: identificador de la película.
    - n: número de películas en las que ha participado el director.
    - weighted_mean_votos: media de votos ponderada en función al número de películas n y a las variables vote_count y vote_average.
    - weighted_mean_beneficios: media de beneficios ponderada en función a los beneficios y al número de películas n.
    - weighted_mean_popularidad: media de la popularidad ponderada en función a la popularity y al número de películas n.
3. géneros: contiene los géneros más populares, cuya variable vote_count es mayor de 10. Es decir, descarta los que tienen muy pocos votos. Este dataset contiene las siguientes variables.
    - id: identificador de la película.
    - n: número de películas a las que pertenece el género indicado.
    - weighted_mean_votos: media de votos ponderada en función al número de películas n y a las variables vote_count y vote_average.
    - weighted_mean_beneficios: media de beneficios ponderada en función a los beneficios y al número de películas n.

- weighted_mean_popularidad: media de la popularidad ponderada en función a la popularity y al número de películas n.

4. productoras: contiene las productoras más populares, cuya variable vote_count es mayor de 10. Es decir, descarta los que tienen muy pocos votos. Este dataset contiene las siguientes variables.
   - id: identificador de la película.
   - n: número de películas producidas por la productora en cuestión.
   - weighted_mean_votos: media de votos ponderada en función al número de películas n y a las variables vote_count y vote_average.
   - weighted_mean_beneficios: media de beneficios ponderada en función a los beneficios y al número de películas n.
   - weighted_mean_popularidad: media de la popularidad ponderada en función a la popularity y al número de películas n.

A continuación, mostramos un ejemplo del dataset que nos queda para los directores:

Filter

	director	n	weighted_mean_votos	weighted_mean_beneficios	weighted_mean_popularidad
1	Tim Burton	14	8.2	105981275	127.525581
2	Stanley Kubrick	6	8.3	34542841	105.792936
3	Zhang Yimou	6	6.9	499390539	104.121555
4	Terry Gilliam	7	7.7	420000000	101.741550
5	Hayao Miyazaki	4	6.0	301818803	91.332849
6	Robert Zemeckis	13	7.7	150949000	79.754966
7	Chris Kentis	2	7.7	300114312	73.944049
8	Martin Brest	4	8.2	21836394	63.654244
9	Guy Ritchie	7	6.5	247850012	63.079003
10	Roger Spottiswoode	4	6.1	13712074	62.898336
11	Kevin Spacey	2	5.4	159047763	60.810723

```
#Parte 2. generación datasets para el estudio de los datos

keywords <- peliculas %>%
  filter(nchar(keywords)>2) %>%
  mutate(
    js = lapply(keywords, fromJSON)
  ) %>%
  unnest(js) %>%
  select(id, title, keyword=name)
slice(keywords, sample(1:4800, 5))

generos <- peliculas %>%
  filter(nchar(genres)>2) %>%
  mutate(
    js = lapply(genres, fromJSON)
  ) %>%
  unnest(js) %>%
  select(id, title, genre=name)
slice(generos, sample(1:4800, 5))
```

```
lenguas <- peliculas %>%
  filter(nchar(spoken_languages)>2) %>%
  mutate(
    js = lapply(spoken_languages, fromJSON)
  ) %>%
  unnest(js) %>%
  select(id, title, spoken_languages=name)


productoras <- peliculas %>%
  filter(nchar(production_companies)>2) %>%
  mutate(
    js = lapply(production_companies, fromJSON)
  ) %>%
  unnest(js) %>%
  select(id, title, production_companies=name)


paises <- peliculas %>%
  filter(nchar(production_countries)>2) %>%
  mutate(
    js = lapply(production_countries, fromJSON)
  ) %>%
  unnest(js) %>%
  select(id, title, production_countries=name)

reparto <- peliculasCredits %>%
  filter(nchar(cast)>2) %>%
  mutate(
    js = lapply(cast, fromJSON)
  ) %>%
  unnest(js) %>%
  select(id, title, cast=name)


equipo<- peliculasCredits %>%
  filter(nchar(crew)>2) %>%
  mutate(
    js = lapply(crew, fromJSON)
  ) %>%
  unnest(js) %>%
  select(id, title, job, crew=name)



directoresPelicula <- equipo %>%
  filter(job=="Director") %>%
  mutate(director=crew) %>%
  left_join(
    peliculasFiltro,
    by=c("id" = "id")
  )

directores <- directoresPelicula %>%
  group_by(director) %>%
  filter(vote_count>10) %>%
```

```
  summarise(
    n = n(),
    weighted_mean_votos =
      weighted.mean(vote_average, vote_count),
    weighted_mean_beneficios = weighted.mean(beneficios),
    weighted_mean_popularidad = weighted.mean(popularity)) %>%
  filter(n>1) %>%
  arrange(desc(weighted_mean_popularidad,weighted_mean_votos,
weighted_mean_beneficios))

# actores más populares
actoresPelicula <- reparto %>%
  mutate(actor=cast) %>%
  left_join(
    peliculasFiltro,
    by=c("id" = "id")
  )

actores <- actoresPelicula %>%
  group_by(actor) %>%
  filter(vote_count>10) %>%
  summarise(
    n = n(),
    weighted_mean_votos =
      weighted.mean(vote_average, vote_count),
    weighted_mean_beneficios = weighted.mean(beneficios),
    weighted_mean_popularidad = weighted.mean(popularity)) %>%  # and the weighted
average votes
  filter(n>1) %>%                          # and filter out directors of 1 title
  arrange(desc(weighted_mean_popularidad,weighted_mean_votos,
weighted_mean_beneficios))


# generos más populares
generosPelicula <- generos %>%
  mutate(genero=genre) %>%
  left_join(
    peliculasFiltro,
    by=c("id" = "id")
  )

generosPopulares <- generosPelicula %>%
  group_by(genero) %>%
  filter(vote_count>10) %>%
  summarise(
    n = n(),
    weighted_mean_votos =
      weighted.mean(vote_average, vote_count),
    weighted_mean_beneficios = weighted.mean(beneficios),
    weighted_mean_popularidad = weighted.mean(popularity)) %>%  # and the weighted
average votes
  filter(n>1) %>%                          # and filter out directors of 1 title
  arrange(desc(weighted_mean_popularidad,weighted_mean_votos,
weighted_mean_beneficios))

productorasPelicula <- productoras %>%
  mutate(productora=production_companies) %>%
```

```
  left_join(
    peliculasFiltro,
    by=c("id" = "id")
  )

productorasPopulares <- productorasPelicula %>%
  group_by(productora) %>%
  filter(vote_count>10) %>%
  summarise(
    n = n(),
    weighted_mean_votos =
      weighted.mean(vote_average, vote_count),
    weighted_mean_beneficios = weighted.mean(beneficios),
    weighted_mean_popularidad = weighted.mean(popularity)) %>%  # and the weighted
average votes
  filter(n>1) %>%                          # and filter out directors of 1 title
  arrange(desc(weighted_mean_popularidad,weighted_mean_votos,
weighted_mean_beneficios))

#guardamos los datos en un fichero de salida
write.table(directores, file = "\\directores.csv", append = FALSE, quote = TRUE, sep = ",",
            eol = "\n", na = "NA", dec = ".", row.names = FALSE,
            col.names = TRUE, qmethod = c("escape", "double"))

#guardamos los datos en un fichero de salida
write.table(actores, file = "\\actores.csv", append = FALSE, quote = TRUE, sep = ",",
            eol = "\n", na = "NA", dec = ".", row.names = FALSE,
            col.names = TRUE, qmethod = c("escape", "double"))

#guardamos los datos en un fichero de salida
write.table(generosPopulares, file = "\\generos.csv", append = FALSE, quote = TRUE, sep =
","
            eol = "\n", na = "NA", dec = ".", row.names = FALSE,
            col.names = TRUE, qmethod = c("escape", "double"))

#guardamos los datos en un fichero de salida
write.table(productorasPopulares, file = "\\productoras.csv", append = FALSE, quote =
TRUE, sep = ",",
            eol = "\n", na = "NA", dec = ".", row.names = FALSE,
            col.names = TRUE, qmethod = c("escape", "double"))
```

3.2. Comprobación de la normalidad y homogeneidad de la varianza. Si es necesario (y posible), aplicar transformaciones que normalicen los datos.

Para poder hacer las pruebas de normalidad hemos utilizado varios tipos de test que calculan el valor de p. En función del resultado de este valor, podremos rechazar o no la hipótesis nula (H0). En el caso de que p tenga valores muy pequeños, menores de 0.05, habrá que rechazar la hipótesis nula. Si p es mayor de 0.05, no podemos descartar la hipótesis, y podremos garantizar que nuestra muestra se aproxima a una distribución normal. Los test que hemos empleado son los siguientes:

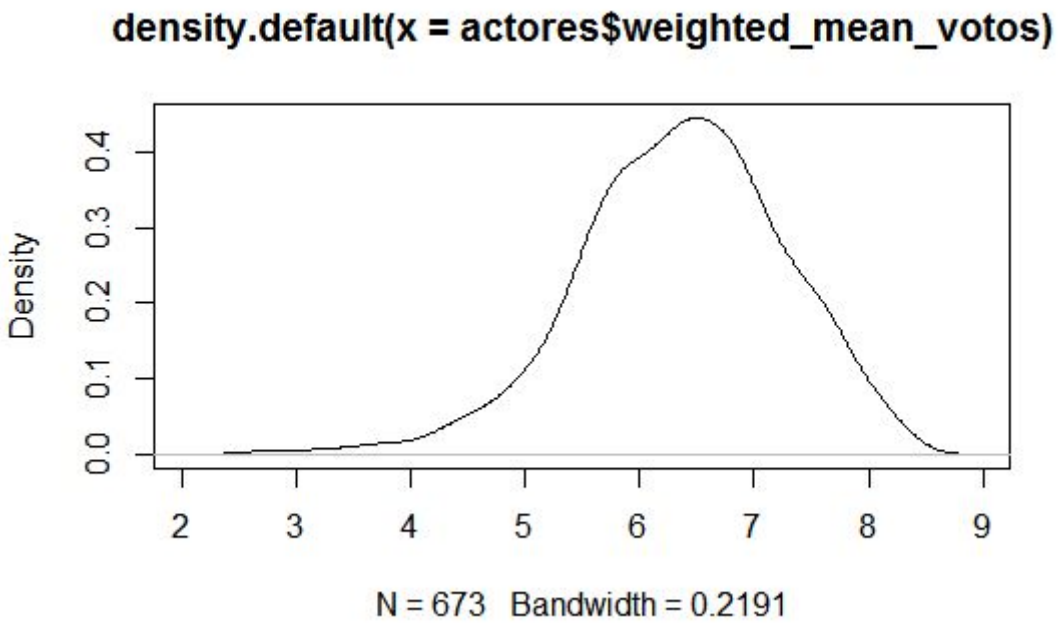
- ***Test Anderson-Darling***: este test se basa en usar las distancias al cuadrado ponderadas entre la línea de ajuste de la gráfica.
- ***Test Cramer-von Mises***: se emplea para juzgar la bondad de una función de distribución acumulada F* comparada con una función de distribución empírica Fn, o para comparar dos distribuciones empíricas. También se utiliza como parte de otros algoritmos, tal como la estimación de la distancia mínima.
- ***Test Shapiro-Wilk***: este test se usa para contrastar la normalidad de un conjunto de datos. Se plantea como hipótesis nula que una muestra x1, ..., xn proviene de una población normalmente distribuida. Es uno de los test más potentes para contrastar la normalidad con muestras pequeñas (menos de 50 muestras). En nuestro caso no es el más indicado ya que tenemos más de 50 muestras.
- ***Test de Lilliefors***: este test asume que la media y la varianza son desconocidas, y se utiliza específicamente para testear la normalidad
- ***Test de Pearson chi-square***: se considera una prueba no paramétrica que mide la discrepancia entre una distribución observada y otra teórica (bondad de ajuste), indicando en qué medida las diferencias existentes entre ambas, de haberlas, se deben al azar en el contraste de hipótesis.
- ***Test de Shapiro-Francia***: es una simplificación del test de Shapiro-Wilk que se utiliza también para contrastar la normalidad de un conjunto de datos. En la práctica las variantes de Shapiro-Wilk y Shapiro-Francia son igualmente buenas. De hecho, la variante Shapiro-Francia en realidad exhibe más poder para distinguir algunas hipótesis alternativas.

En este punto hemos realizado los test descritos en las distintas variables de los cuatro grupos utilizados, dándonos los siguientes resultados.

Media ponderada de los votos

Hemos representado las funciones de densidad y calculado la p por distintos métodos de test para ver si se aproxima o no a la distribución normal.

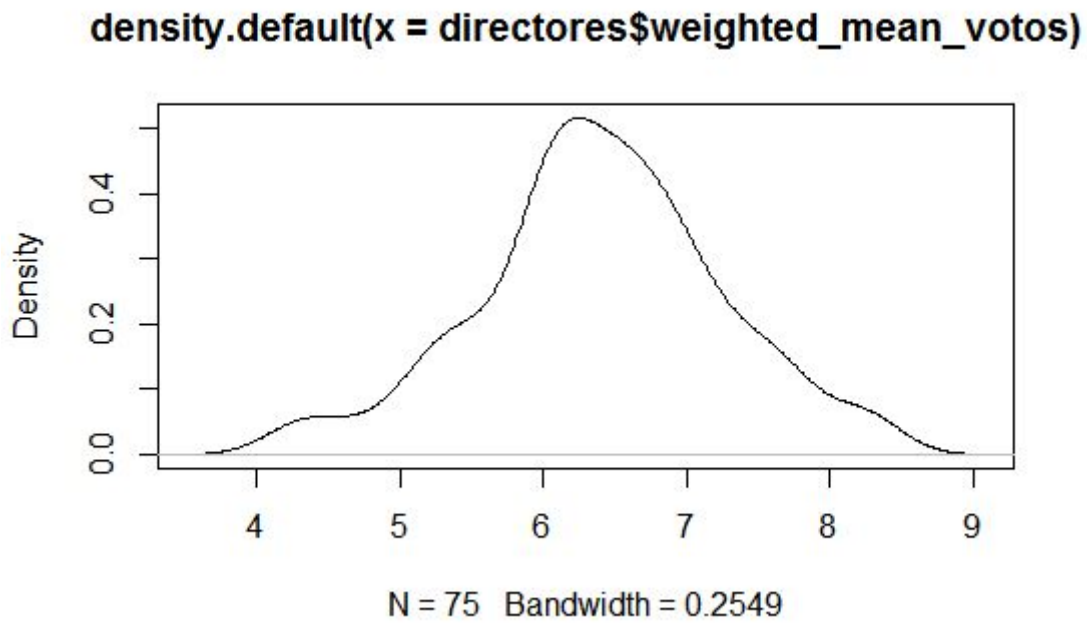
Grupo actores.




```
> ad.test(actores$weighted_mean_votos)$p.value
[1] 0.001251102
```

Observamos que para el test Anderson-Darling, el valor de p es menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_votos para los actores, podemos afirmar que no sigue una distribución normal.

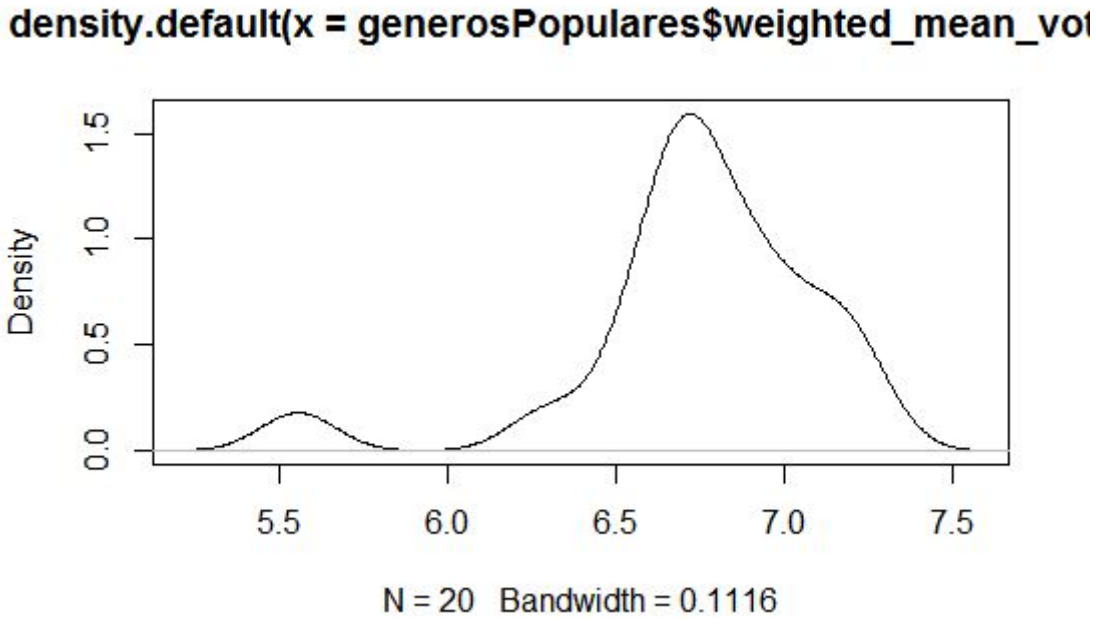
Grupo directores.



```
> # test Anderson-Darling
> ad.test(directores$weighted_mean_votos)$p.value
[1] 0.4772221
> # test Cramer-von Mises
> cvm.test(directores$weighted_mean_votos)$p.value
[1] 0.4050652
> # test Shapiro-Wilk
> shapiro.test(directores$weighted_mean_votos)$p.value
[1] 0.5510684
> #test de Lilliefors
> lillie.test(directores$weighted_mean_votos)$p.value
[1] 0.0355243
> # test de Pearson chi-square
> pearson.test(directores$weighted_mean_votos)$p.value
[1] 0.5301203
> # test de Shapiro-Francia
> sf.test(directores$weighted_mean_votos)$p.value
[1] 0.553442
```

Observamos que para todos los test excepto para el de Lilliefor, el valor de p es mayor de 0.05, por lo que no podemos rechazar la H0. Es decir, la variable weighted_mean_votos para los directores, podemos afirmar que sigue una distribución normal.

Grupo géneros.

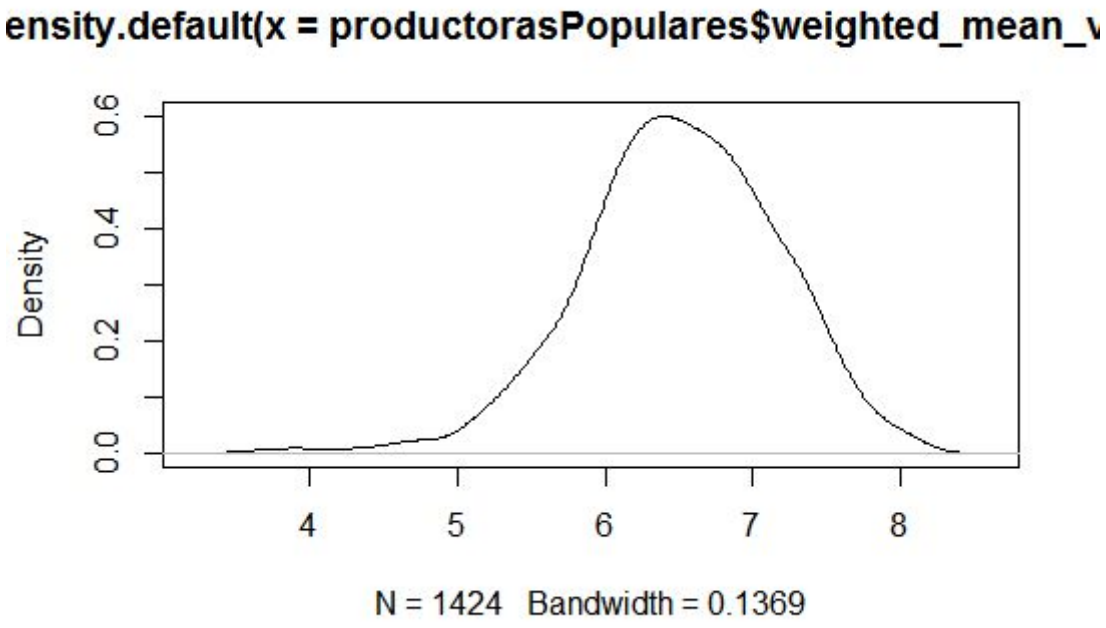



```
> pearson.test(generosPopulares$weighted_mean_votos)$p.value
```

```
[1] 0.2673849
> cvm.test(generosPopulares$weighted_mean_votos)$p.value
[1] 0.05489025
```

Observamos que para los test aplicados, el valor de p es mayor de 0.05, por lo que no podemos rechazar la H0. Es decir, la variable weighted_mean_votos para los géneros, podemos afirmar que sigue una distribución normal.

Grupo productoras.



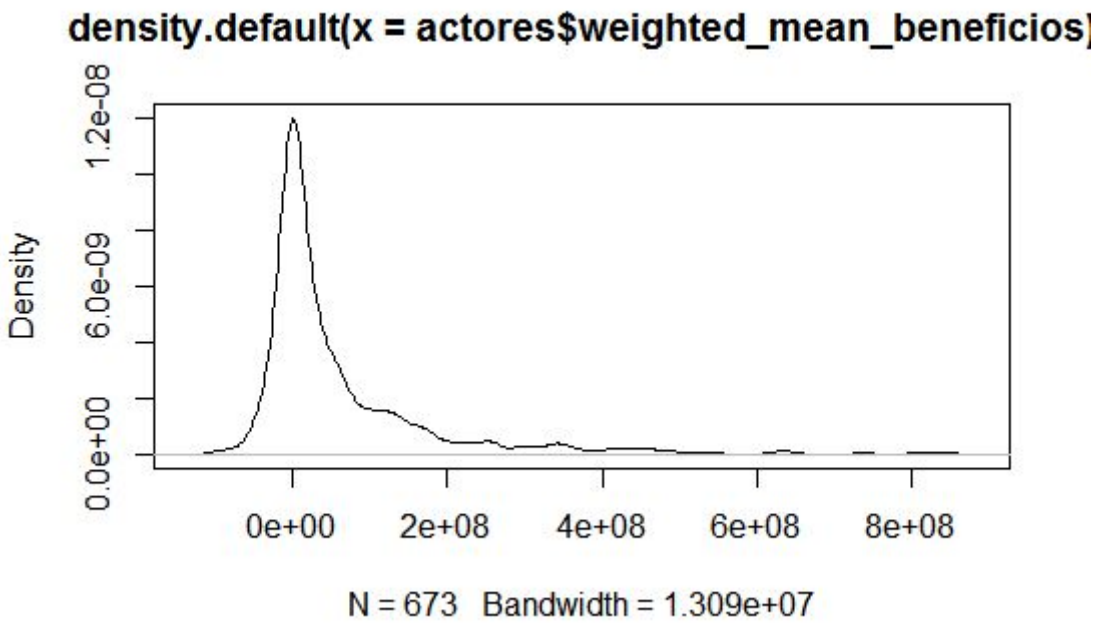

```
> ad.test(productorasPopulares$weighted_mean_votos)$p.value
[1] 0.0004961054
> # test Anderson-Darling
> cvm.test(productorasPopulares$weighted_mean_votos)$p.value
[1] 0.01185947
```

Observamos que para los test aplicados, el valor de p es menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_votos para las productoras, podemos afirmar que no sigue una distribución normal.

Media ponderada de los beneficios

Hemos representado las funciones de densidad y calculado la p por distintos métodos de test para ver si se aproxima o no a la distribución normal.

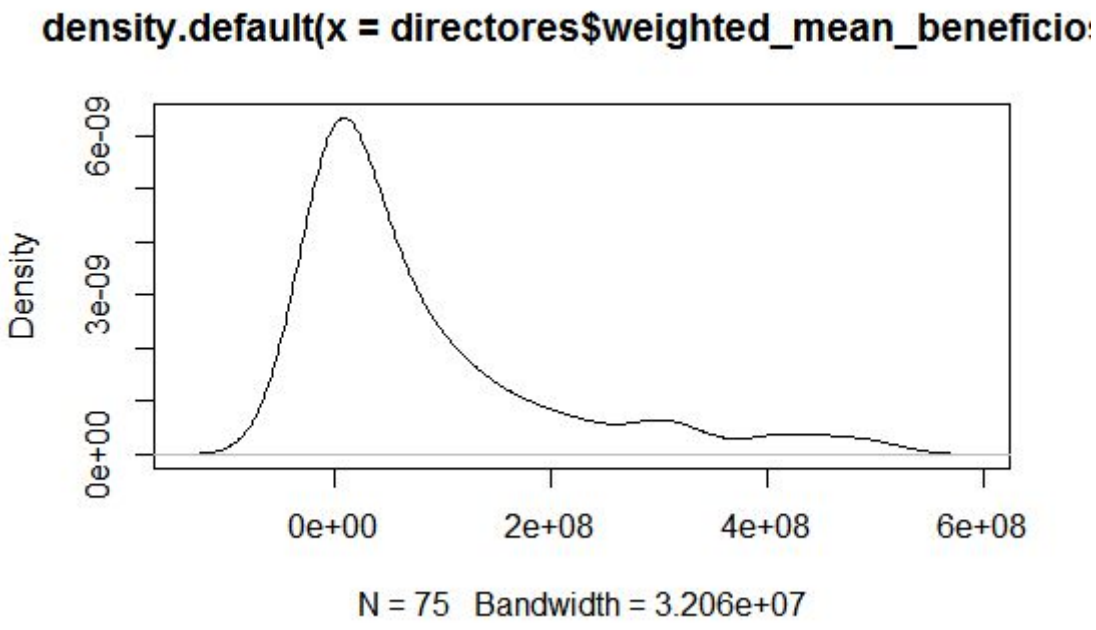
Grupo actores.




```
> ad.test(actores$weighted_mean_beneficios)$p.value
[1] 3.7e-24
```

Observamos que para los test aplicados, el valor de p es mucho menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_beneficios para los actores, podemos afirmar que no sigue una distribución normal.

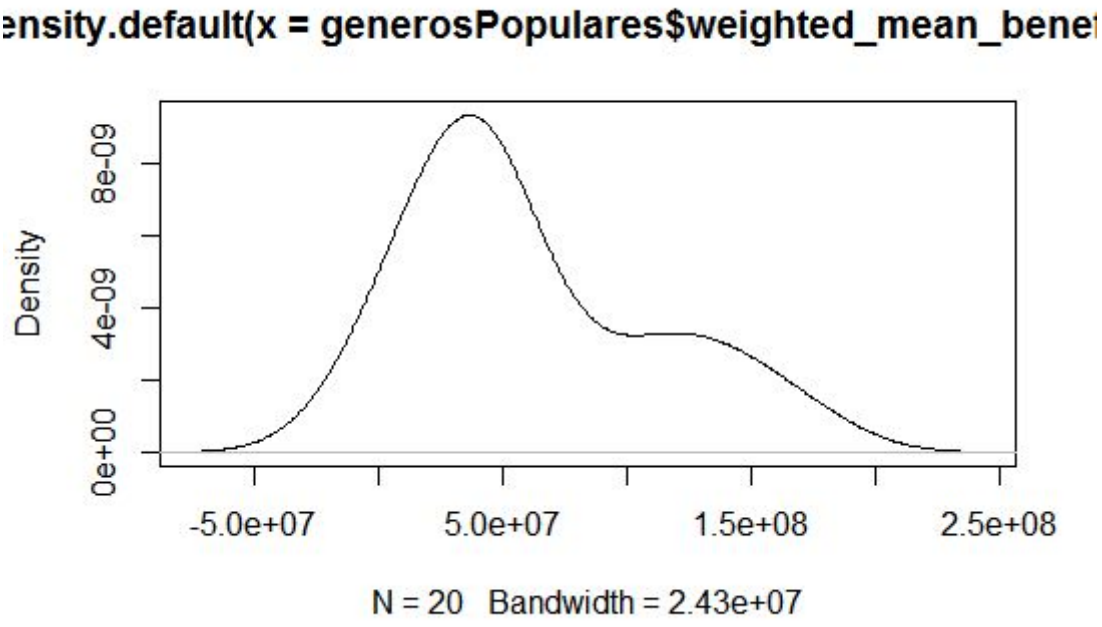
Grupo directores.



```
> ad.test(directores$weighted_mean_beneficios)$p.value
[1] 8.99459e-15
```

Observamos que para los test aplicados, el valor de p es mucho menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_beneficios para los directores, podemos afirmar que no sigue una distribución normal.

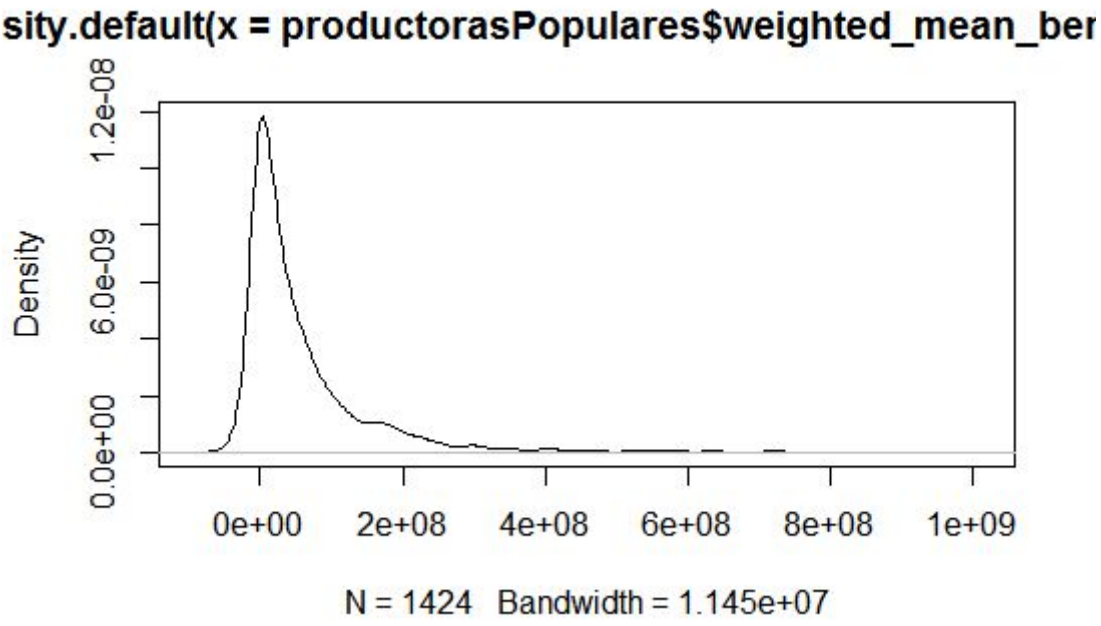
Grupo géneros.




```
> cvm.test(generosPopulares$weighted_mean_beneficios)$p.value
[1] 0.00649805
> ad.test(generosPopulares$weighted_mean_beneficios)$p.value
[1] 0.01309089
> pearson.test(generosPopulares$weighted_mean_beneficios)$p.value
[1] 0.002526813
```

Observamos que para los test aplicados, el valor de p es mucho menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_beneficios para los géneros, podemos afirmar que no sigue una distribución normal.

Grupo productoras.



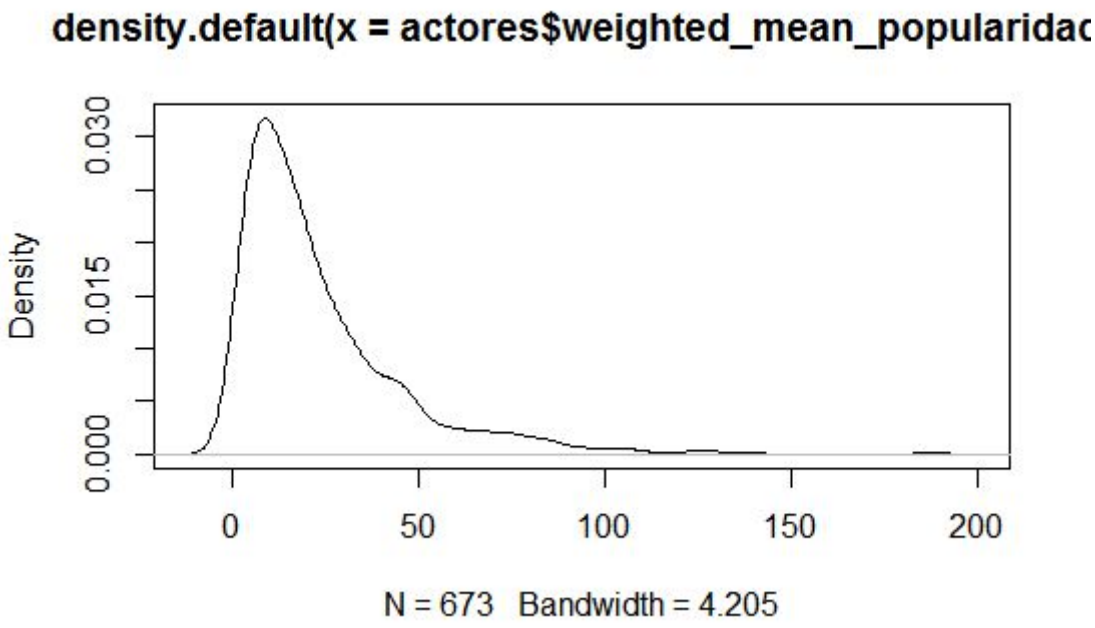

```
> ad.test(productorasPopulares$weighted_mean_beneficios)$p.value
[1] 3.7e-24
```

Observamos que para los test aplicados, el valor de p es mucho menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_beneficios para las productoras, podemos afirmar que no sigue una distribución normal.

Media ponderada de la popularidad

Hemos representado las funciones de densidad y calculado la p por distintos métodos de test para ver si se aproxima o no a la distribución normal.

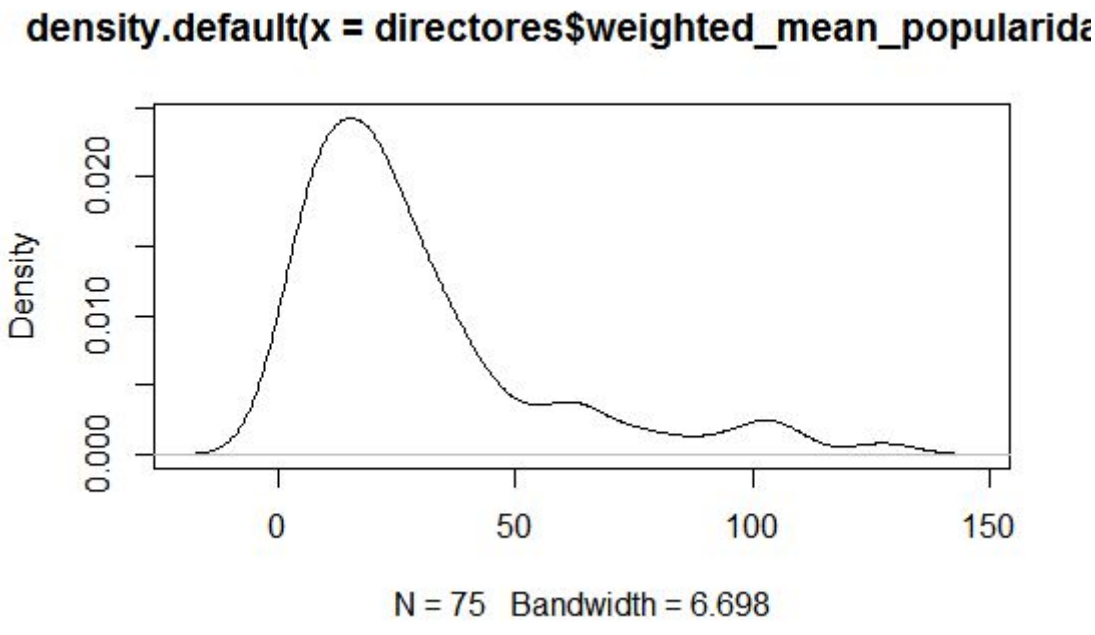
Grupo actores.




```
> ad.test(directores$weighted_mean_popularidad)$p.value
[1] 1.019893e-11
```

Observamos que para los test aplicados, el valor de p es mucho menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_popularidad para los actores, podemos afirmar que no sigue una distribución normal.

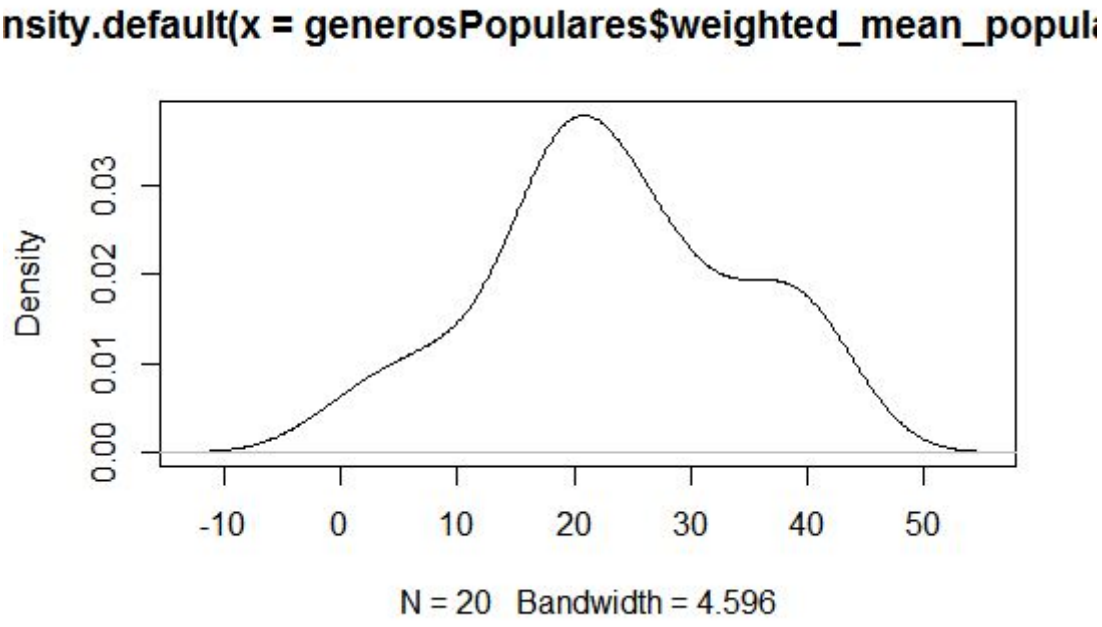
Grupo directores.



```
> ad.test(directores$weighted_mean_popularidad)$p.value
[1] 1.019893e-11
```

Observamos que para los test aplicados, el valor de p es mucho menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_popularidad para los directores, podemos afirmar que no sigue una distribución normal.

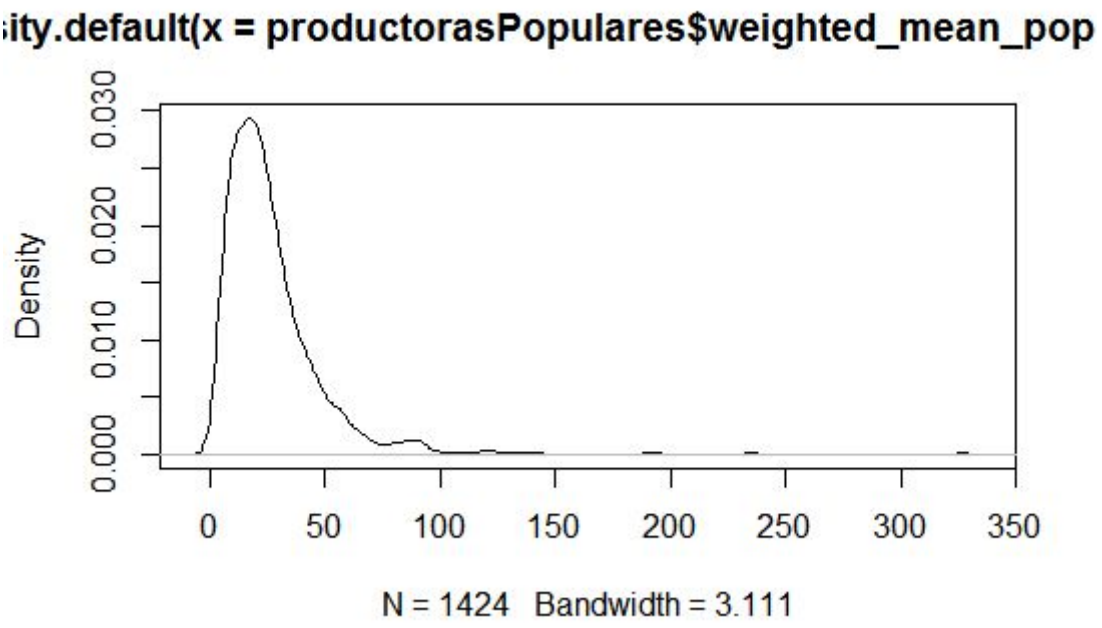
Grupo géneros.




```
> ad.test(generosPopulares$weighted_mean_popularidad)$p.value
[1] 0.6216736
> pearson.test(generosPopulares$weighted_mean_popularidad)$p.value
[1] 0.2067418
> cvm.test(generosPopulares$weighted_mean_popularidad)$p.value
[1] 0.575506
```

Observamos que para los test aplicados, el valor de p es mayor de 0.05, por lo que no podemos rechazar la H0. Es decir, la variable weighted_mean_popularidad para los géneros, podemos afirmar que sigue una distribución normal.

Grupo productoras.




```
> ad.test(productorasPopulares$weighted_mean_popularidad)$p.value
[1] 3.7e-24
```

Observamos que para los test aplicados, el valor de p es mucho menor de 0.05, por lo que podemos rechazar la H0. Es decir, la variable weighted_mean_popularidad para las productoras, podemos afirmar que no sigue una distribución normal.

El código R utilizado ha sido el siguiente:

```
# Comprobación normalidad
require(nortest)
install.packages("nortest")
library(nortest)

# Todas tienen valores muy pequeños de p, por lo que rechazan la H0. No siguen
distribución normal

install.packages("nortest")
require(nortest)

plot(density(directores$weighted_mean_votos))
# test Anderson-Darling
ad.test(directores$weighted_mean_votos)$p.value
# test Cramer-von Mises
cvm.test(directores$weighted_mean_votos)$p.value
# test Shapiro-Wilk
shapiro.test(directores$weighted_mean_votos)$p.value
#test de Lilliefors
lillie.test(directores$weighted_mean_votos)$p.value
# test de Pearson chi-square
```

```
pearson.test(directores$weighted_mean_votos)$p.value
# test de Shapiro-Francia
sf.test(directores$weighted_mean_votos)$p.value
# Todas tienen valores muy pequeños de p, por lo que rechazan la H0. No siguen
distribución normal
plot(density(directores$weighted_mean_beneficios))
# test Anderson-Darling
ad.test(directores$weighted_mean_beneficios)$p.value
# Todas tienen valores muy pequeños de p, por lo que rechazan la H0. No siguen
distribución normal
plot(density(directores$weighted_mean_popularidad))
# test Anderson-Darling
ad.test(directores$weighted_mean_popularidad)$p.value

plot(density(actores$weighted_mean_votos))
# test Anderson-Darling
pearson.test(actores$weighted_mean_votos)$p.value
plot(density(actores$weighted_mean_beneficios))
# test Anderson-Darling
ad.test(actores$weighted_mean_beneficios)$p.value
plot(density(actores$weighted_mean_popularidad))
# test Anderson-Darling
ad.test(directores$weighted_mean_popularidad)$p.value

plot(density(generosPopulares$weighted_mean_votos))
# test Anderson-Darling
cvm.test(generosPopulares$weighted_mean_votos)$p.value
plot(density(generosPopulares$weighted_mean_beneficios))
# test Anderson-Darling
cvm.test(generosPopulares$weighted_mean_beneficios)$p.value
ad.test(generosPopulares$weighted_mean_beneficios)$p.value
pearson.test(generosPopulares$weighted_mean_beneficios)$p.value

plot(density(generosPopulares$weighted_mean_popularidad))
# test Anderson-Darling
ad.test(generosPopulares$weighted_mean_popularidad)$p.value
pearson.test(generosPopulares$weighted_mean_popularidad)$p.value
cvm.test(generosPopulares$weighted_mean_popularidad)$p.value


plot(density(productorasPopulares$weighted_mean_votos))
# test Anderson-Darling
ad.test(productorasPopulares$weighted_mean_votos)$p.value
cvm.test(productorasPopulares$weighted_mean_votos)$p.value
plot(density(productorasPopulares$weighted_mean_beneficios))
# test Anderson-Darling
ad.test(productorasPopulares$weighted_mean_beneficios)$p.value
plot(density(productorasPopulares$weighted_mean_popularidad))
# test Anderson-Darling
ad.test(productorasPopulares$weighted_mean_popularidad)$p.value



# test Anderson-Darling
ad.test(peliculasFiltro$vote_count)$p.value
# test Cramer-von Mises
cvm.test(peliculasFiltro$vote_count)$p.value
```

```
# test Shapiro-Wilk
shapiro.test(peliculasFiltro$vote_count)$p.value
#test de Lilliefors
lillie.test(peliculasFiltro$vote_count)$p.value
# test de Pearson chi-square
pearson.test(peliculasFiltro$vote_count)$p.value
# test de Shapiro-Francia
sf.test(peliculasFiltro$vote_count)$p.value
# Todas tienen valores muy pequeños de p, por lo que rechazan la H0. No siguen
distribución normal

plot(density(peliculasFiltro$beneficios))
# test Anderson-Darling
ad.test(peliculasFiltro$beneficios)$p.value
# test Cramer-von Mises
cvm.test(peliculasFiltro$beneficios)$p.value
# test Shapiro-Wilk
shapiro.test(peliculasFiltro$beneficios)$p.value
#test de Lilliefors
lillie.test(peliculasFiltro$beneficios)$p.value
# test de Pearson chi-square
pearson.test(peliculasFiltro$beneficios)$p.value
# test de Shapiro-Francia
sf.test(peliculasFiltro$beneficios)$p.value


# test Anderson-Darling
ad.test(peliculasFiltro$popularity)$p.value
# test Cramer-von Mises
cvm.test(peliculasFiltro$popularity)$p.value
# test Shapiro-Wilk
shapiro.test(peliculasFiltro$popularity)$p.value
#test de Lilliefors
lillie.test(peliculasFiltro$popularity)$p.value
# test de Pearson chi-square
pearson.test(peliculasFiltro$popularity)$p.value
# test de Shapiro-Francia
sf.test(peliculasFiltro$popularity)$p.value



#buscamos extreme scores en cuanto a popularidad,por la gráfica vemos 6
plot(peliculasFiltro$popularity)
# creamos un data frame auxiliar sin extremes scores de la popularidad
peliculasFiltroOrder <- arrange(peliculasFiltro, desc(peliculasFiltro$popularity))
peliculasFiltroOrderDel <- peliculasFiltroOrder[-c(1, 2, 3, 4, 5, 6), ]
# comprobamos si quitando estos valores, sigue una normal y vemos que tampoco
ad.test(peliculasFiltroOrderDel$popularity)$p.value

plot(peliculasFiltroOrderDel$popularity)


# hacemos los test de homogeneidad de varianzas
# vamos a separar por grupos con o sin beneficios
# añadimos una nueva columna que nos indique si hay o no beneficios en la película
peliculasFiltro <- mutate(peliculasFiltro, swBeneficios = peliculasFiltro$beneficios > 0)
```

Por otro lado, para el estudio de la homogeneidad entre grupos, se pueden utilizar diversos métodos también (***Fligner-Killen***, ***Bartlett*** y ***Levene***). En nuestro caso, al tratarse algunas de distribuciones no normales, hemos tenido que decantarnos por el método de ***Fligner-Killen***, que es válido para todo tipo de distribuciones.

Este método permite comprobar la igualdad de varianzas de dos o más muestras que provienen de distribuciones que no son normales. Calcula la homogeneidad de las varianzas a través de la mediana, no con la media, por eso es adecuado en nuestro caso.

Para realizar este cálculo, hemos discretizado las columnas con respecto a las que queremos analizar los grupos para los que nos interesa hacer el cálculo de las varianzas.

SwBeneficios

Hemos añadido una columna a nuestros datos que contiene un true o un false en caso de que haya habido beneficios o no en una película. Así, hemos discretizado la columna beneficios. Si el valor es positivo, tenemos un true; en caso contrario, tenemos un false.

beneficios	swBeneficios
77551594	TRUE
0	FALSE
64596398	TRUE
103403799	TRUE
60293714	TRUE
11112916	TRUE
110268750	TRUE
69832389	TRUE
33318987	TRUE
-13297099	FALSE
47139399	TRUE

De este modo, a través de la función de R disponible para calcular la homogeneidad de las varianzas con el test de ***Fligner-Killen***, hemos podido pasar como variable discreta swBeneficios (para dividir los dos grupos existentes en cuanto a beneficios),y como variable continua la columna de vote_count. El resultado obtenido es el siguiente en este caso:

```
> fligner.test(peliculasFiltro$vote_count~peliculasFiltro$swBeneficios, peliculasFiltro)

        Fligner-Killeen test of homogeneity of variances

data:  peliculasFiltro$vote_count by peliculasFiltro$swBeneficios
Fligner-Killeen:med chi-squared = 1648.1, df = 1, p-value < 2.2e-16
```

Observamos que no hay homogeneidad en las varianzas para estos grupos (beneficios y no beneficios) con respecto a la variable vote_count.

Si hacemos el mismo cálculo para la variable popularity, observamos el mismo resultado. No hay homogeneidad.

```
> fligner.test(peliculasFiltro$popularity~peliculasFiltro$swBeneficios, peliculasFiltro)

        Fligner-Killeen test of homogeneity of variances

data:  peliculasFiltro$popularity by peliculasFiltro$swBeneficios
Fligner-Killeen:med chi-squared = 1081.1, df = 1, p-value < 2.2e-16
```

Podríamos hacer la misma prueba discretizando otra de las variables, por ejemplo, la variable vote_average. Podríamos establecer grupos dividiendo estas medias de votos por intervalos. Posteriormente, haríamos el mismo estudio con el test de Fligner (con respecto a las variables continuas de vote_count, beneficios y popularity).

```
#homogeneidad varianzas
install.packages("car")
require(car)
library("dplyr")
# hacemos los test de homogeneidad de varianzas
# vamos a separar por grupos con o sin beneficios
# añadimos una nueva columna que nos indique si hay o no beneficios en la película
peliculasFiltro <- mutate(peliculasFiltro, swBeneficios = peliculasFiltro$beneficios > 0)
# en este caso, el método que nos sirve es el fligner test, puesto que los datos no
# siguen una distribución normal
fligner.test(peliculasFiltro$vote_count~peliculasFiltro$swBeneficios, peliculasFiltro)
# p tiene un valor muy pequeño, así que tampoco hay homogeneidad en las varianzas
# probamos otros métodos y observamos que se ve el mismo resultado
bartlett.test(peliculasFiltro$vote_count, peliculasFiltro$swBeneficios, peliculasFiltro)
bartlett.test(peliculasFiltro$vote_count~peliculasFiltro$swBeneficios, peliculasFiltro)
leveneTest(peliculasFiltro$vote_count~peliculasFiltro$swBeneficios, peliculasFiltro)
```

3.3. Aplicación de pruebas estadísticas (tantas como sea posible) para comparar los grupos de datos.

A continuación vamos a representar las gráficas QQplot de los distintos grupos y sus correspondientes variables.

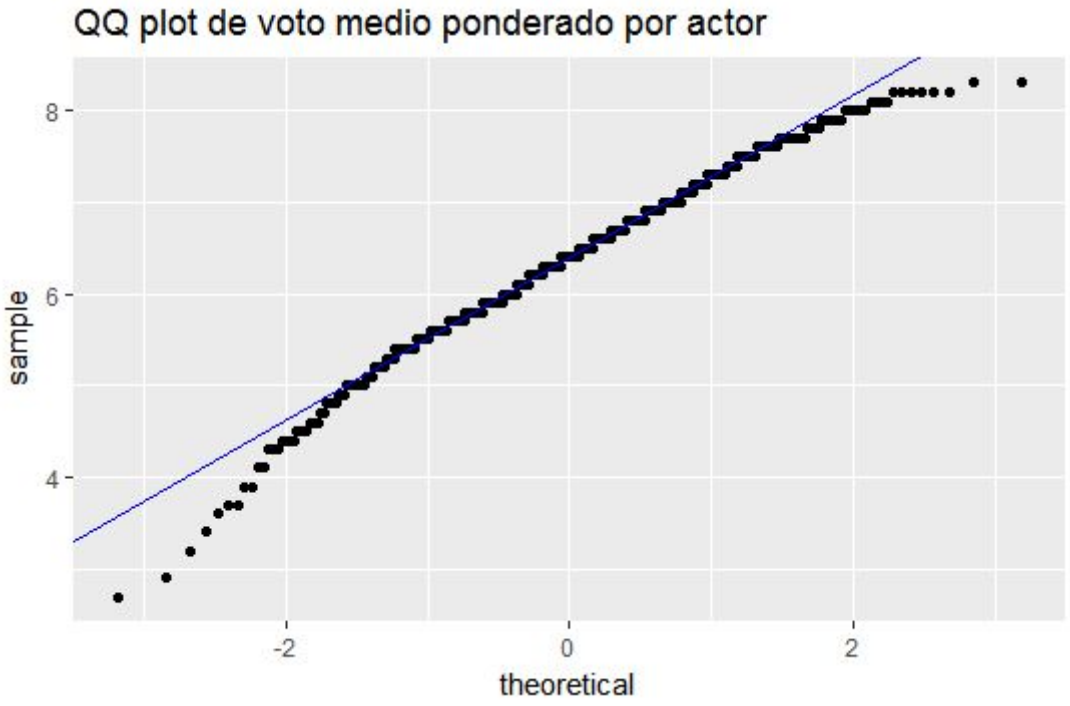
El QQ plot es un método gráfico para el diagnóstico de diferencias entre la distribución de probabilidad de una población de la que se ha extraído una muestra aleatoria y una distribución usada para la comparación.

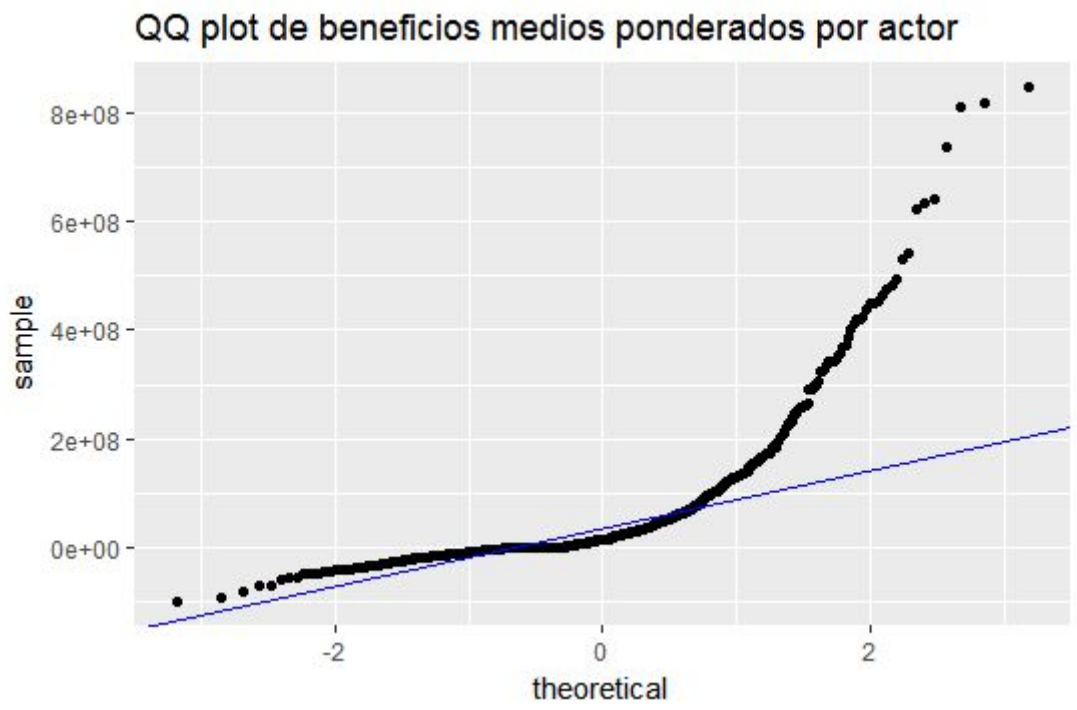
Para cada grupo, vamos a representar las tres variables ponderadas en base al grupo (actores, directores, géneros y productoras).

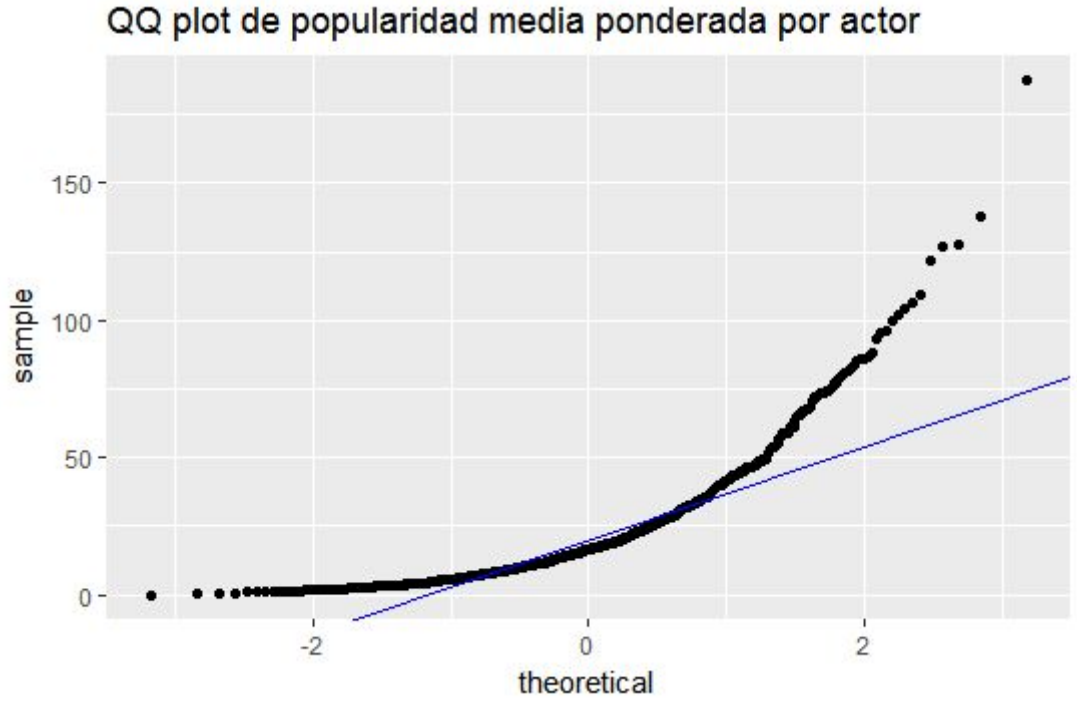
Si la distribución de la variable es la misma que la distribución de comparación se obtendrá una aproximación a una recta. En el caso de que se den desviaciones sustanciales de la linealidad, los estadísticos rechazan la hipótesis nula de similitud.

En las gráficas presentadas a continuación, se observa que las que se aproximan a la recta en su centro, son curiosamente, las variables referentes a los grupos de las que, en el apartado anterior, hemos obtenido que son distribuciones normales a través de los test.

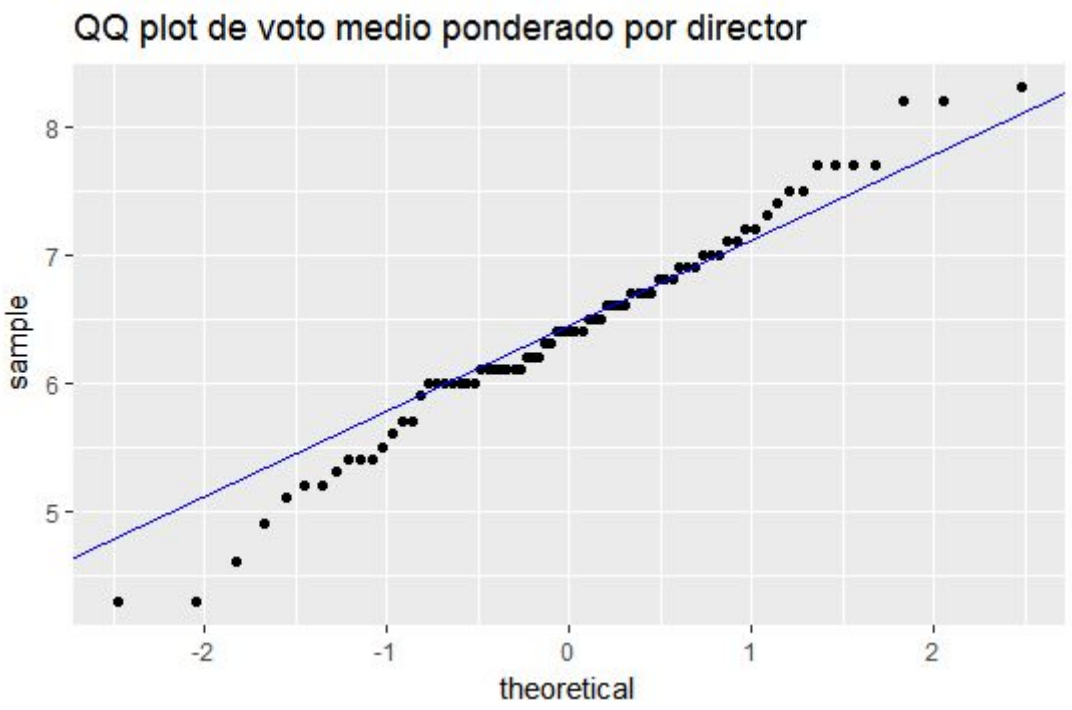
Grupo actores.

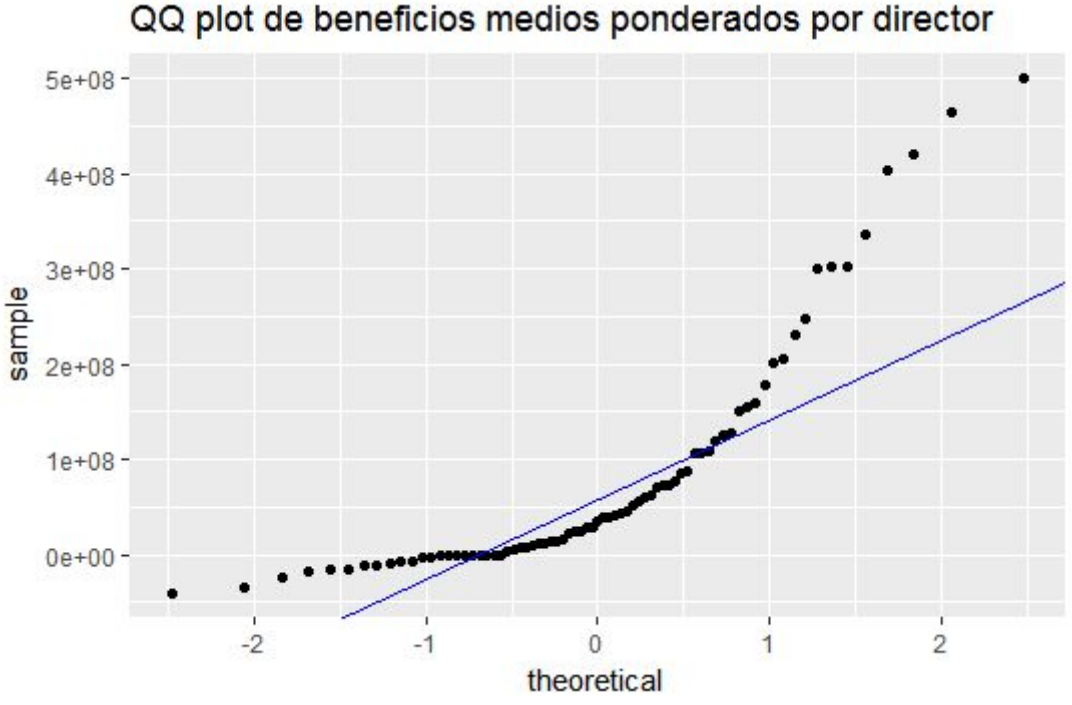


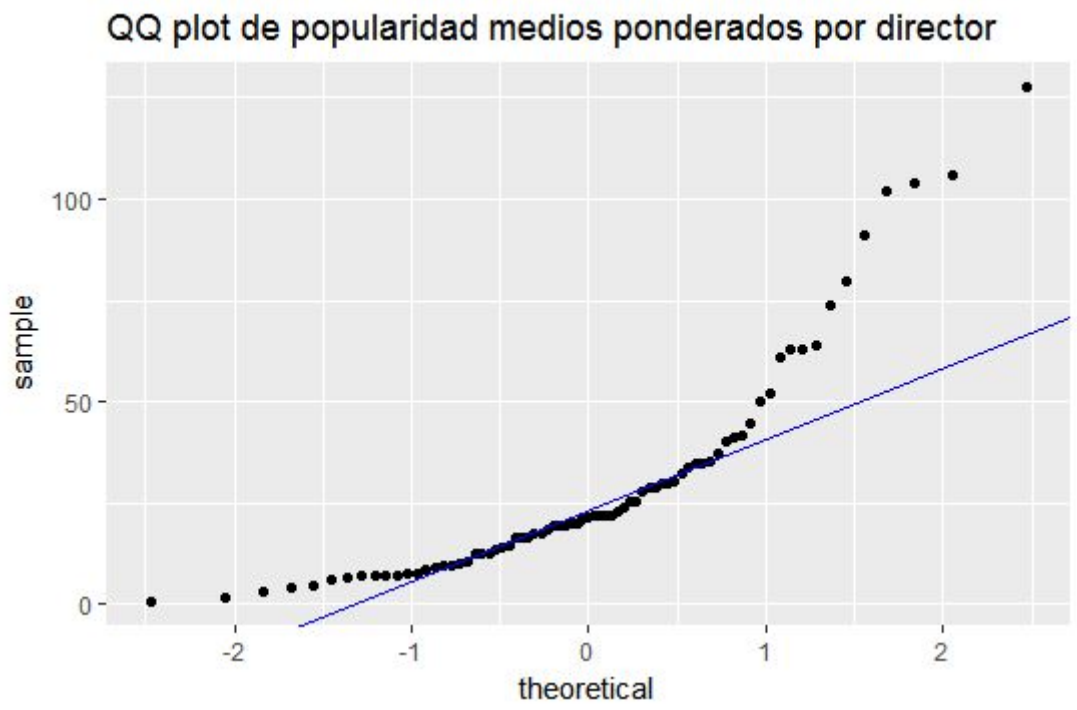




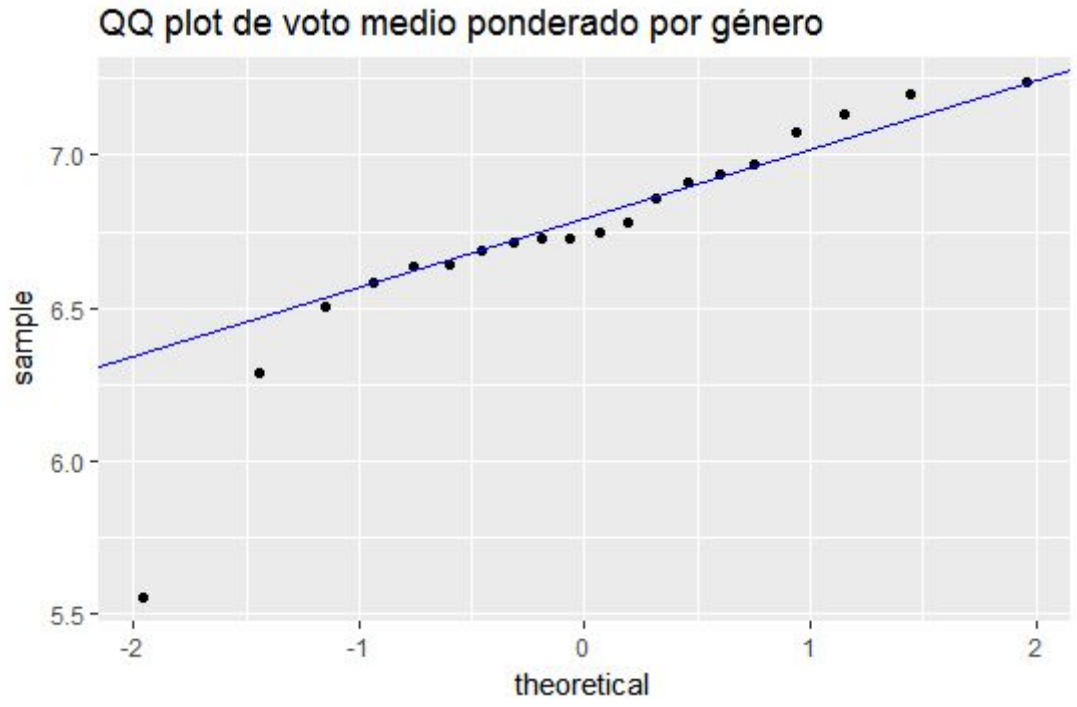
Grupo directores.

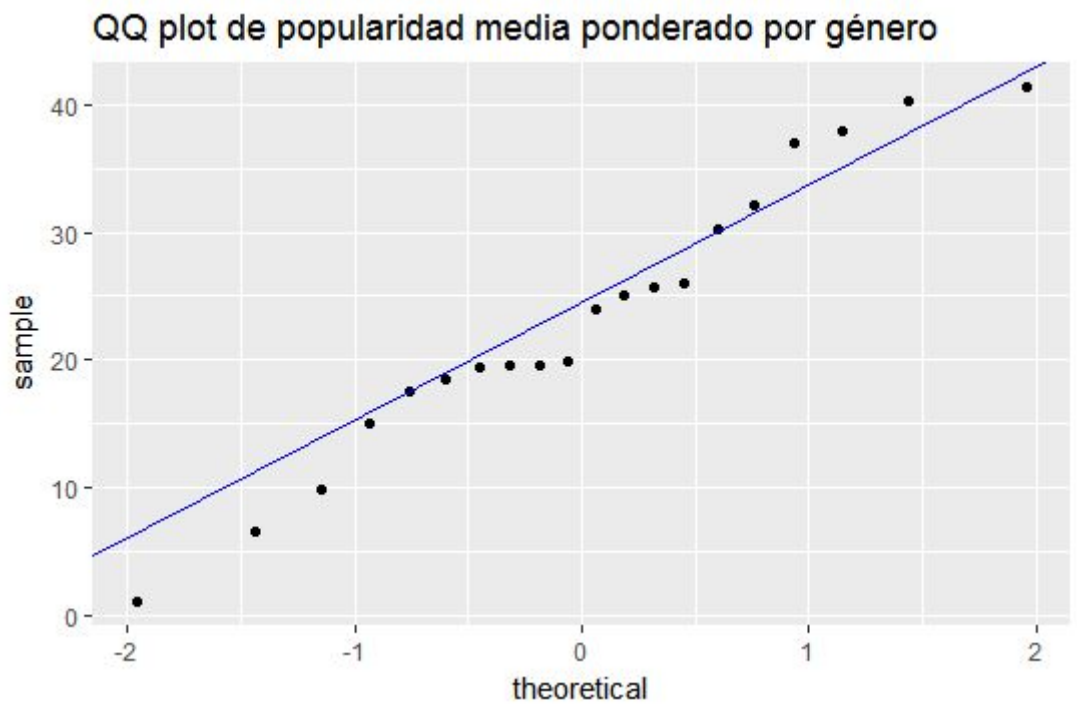

QQ plot de voto medio ponderado por director
sample
theoretical
8
7
6
5
-2
-1
0
1
2

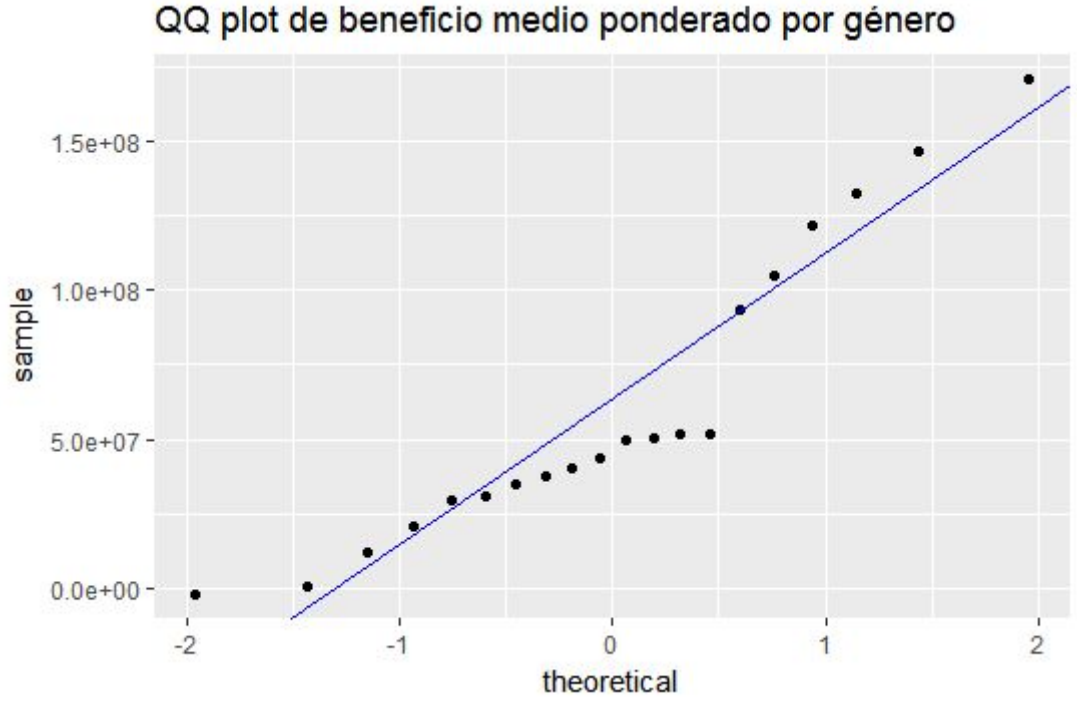

QQ plot de beneficios medios ponderados por director
sample
theoretical
5e+08
4e+08
3e+08
2e+08
1e+08
0e+00
-2
-1
0
1
2



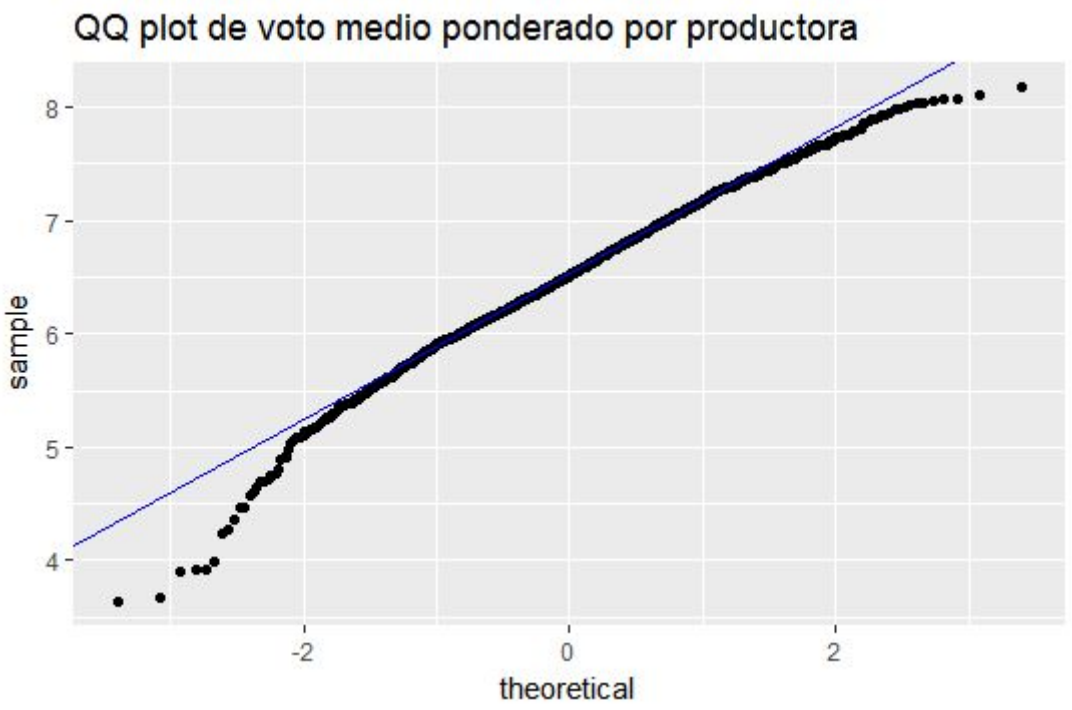
Grupo géneros.

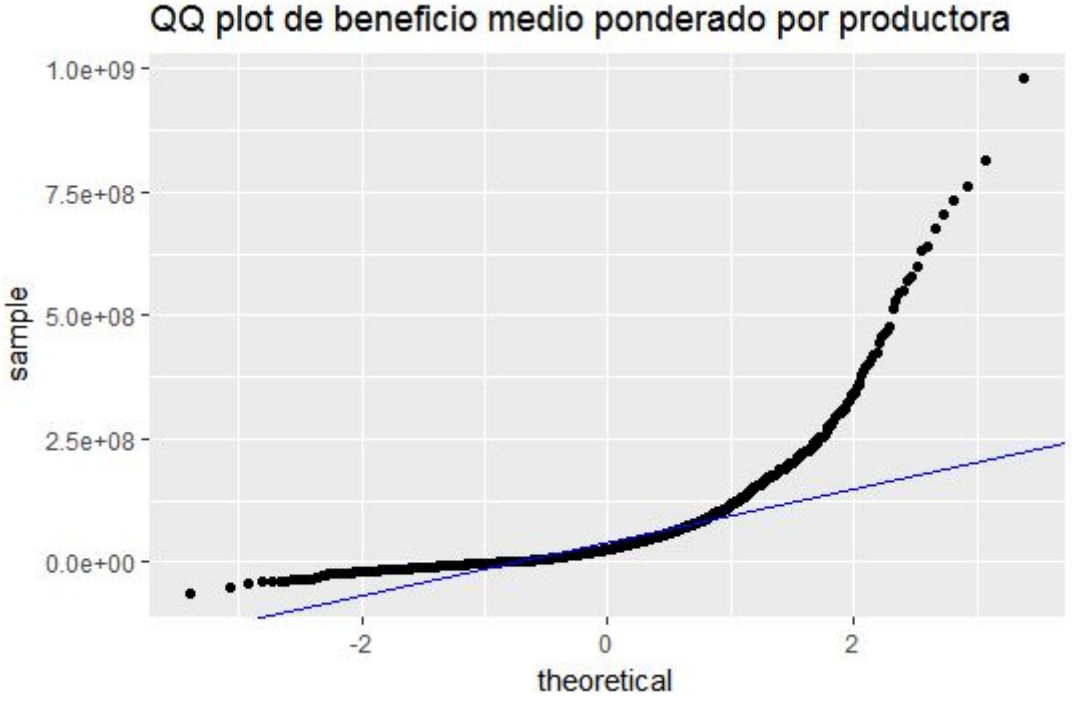


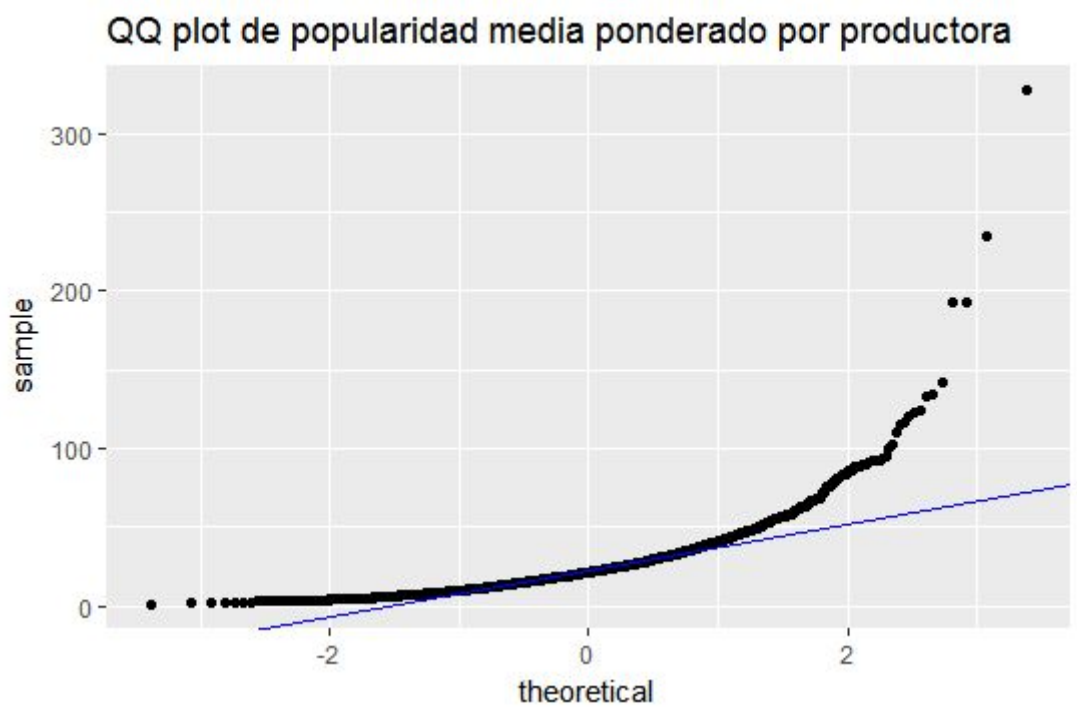




<u>Grupo productoras.</u>


QQ plot de voto medio ponderado por productora
sample
8
7
6
5
4
-2
0
2
theoretical


QQ plot de beneficio medio ponderado por productora
sample
1.0e+09
7.5e+08
5.0e+08
2.5e+08
0.0e+00
-2
0
2
theoretical




```
install.packages("ggplot2")
library(ggplot2)
#histogramas y QQ plot
hist(directores$weighted_mean_votos, xlab="Peso", ylab="Frecuencia", las=1, main="")
y <- quantile(directores$weighted_mean_votos, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(directores) + stat_qq(aes(sample=weighted_mean_votos))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de voto medio ponderado por director")

hist(directores$weighted_mean_beneficios, xlab="Peso", ylab="Frecuencia", las=1, main="")
y <- quantile(directores$weighted_mean_beneficios, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(directores) + stat_qq(aes(sample=weighted_mean_beneficios))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de beneficios medios ponderados por director")

hist(directores$weighted_mean_popularidad, xlab="Peso", ylab="Frecuencia", las=1,
main="")
y <- quantile(directores$weighted_mean_popularidad, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(directores) + stat_qq(aes(sample=weighted_mean_popularidad))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de popularidad medios ponderados por director")


hist(actores$weighted_mean_votos, xlab="Peso", ylab="Frecuencia", las=1, main="")
```

```
y <- quantile(actores$weighted_mean_votos, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(actores) + stat_qq(aes(sample=weighted_mean_votos))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de voto medio ponderado por actor")

hist(actores$weighted_mean_beneficios, xlab="Peso", ylab="Frecuencia", las=1, main="")
y <- quantile(actores$weighted_mean_beneficios, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(actores) + stat_qq(aes(sample=weighted_mean_beneficios))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de beneficios medios ponderados por actor")

hist(actores$weighted_mean_popularidad, xlab="Peso", ylab="Frecuencia", las=1, main="")
y <- quantile(actores$weighted_mean_popularidad, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(actores) + stat_qq(aes(sample=weighted_mean_popularidad))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de popularidad media ponderada por actor")


hist(generosPopulares$weighted_mean_votos, xlab="Peso", ylab="Frecuencia", las=1,
main="")
y <- quantile(generosPopulares$weighted_mean_votos, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(generosPopulares) + stat_qq(aes(sample=weighted_mean_votos))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de voto medio ponderado por género")

hist(generosPopulares$weighted_mean_beneficios, xlab="Peso", ylab="Frecuencia", las=1,
main="")
y <- quantile(generosPopulares$weighted_mean_beneficios, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(generosPopulares) + stat_qq(aes(sample=weighted_mean_beneficios))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de beneficio medio ponderado por género")

hist(generosPopulares$weighted_mean_popularidad, xlab="Peso", ylab="Frecuencia",
las=1, main="")
y <- quantile(generosPopulares$weighted_mean_popularidad, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(generosPopulares) + stat_qq(aes(sample=weighted_mean_popularidad))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de popularidad media ponderado por género")
```

```
hist(productorasPopulares$weighted_mean_votos, xlab="Peso", ylab="Frecuencia", las=1,
main="")
y <- quantile(productorasPopulares$weighted_mean_votos, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(productorasPopulares) + stat_qq(aes(sample=weighted_mean_votos))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de voto medio ponderado por productora")

hist(productorasPopulares$weighted_mean_beneficios, xlab="Peso", ylab="Frecuencia",
las=1, main="")
y <- quantile(productorasPopulares$weighted_mean_beneficios, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(productorasPopulares) + stat_qq(aes(sample=weighted_mean_beneficios))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de beneficio medio ponderado por productora")

hist(productorasPopulares$weighted_mean_popularidad, xlab="Peso", ylab="Frecuencia",
las=1, main="")
y <- quantile(productorasPopulares$weighted_mean_popularidad, c(0.25, 0.75))
x    <- qnorm( c(0.25, 0.75))
slope <- diff(y) / diff(x)
int   <- y[1] - slope * x[1]
ggplot(productorasPopulares) + stat_qq(aes(sample=weighted_mean_popularidad))  +
  geom_abline(intercept=int, slope=slope, color="blue") +
  labs(title="QQ plot de popularidad media ponderado por productora")
```

4. y 5. Representación de los resultados a partir de tablas y gráficas//Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Para estos dos apartados hemos relacionado las variables de las tablas resultantes creando asi 4 casos distintos

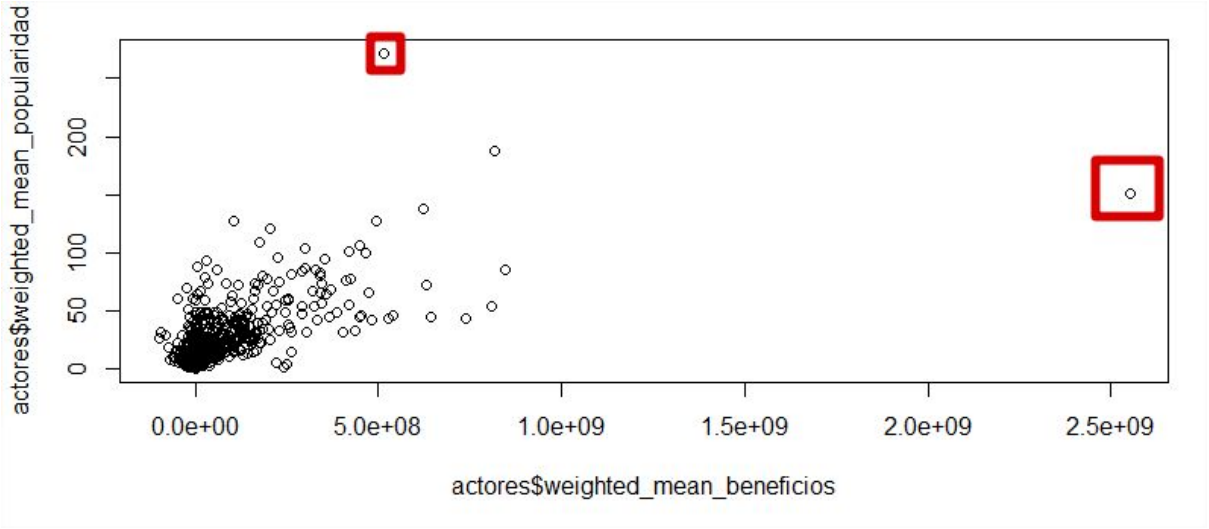
CASO 1: ***Actores***

Para este caso hemos investigado qué variables podrían tener relaciones entre ellas

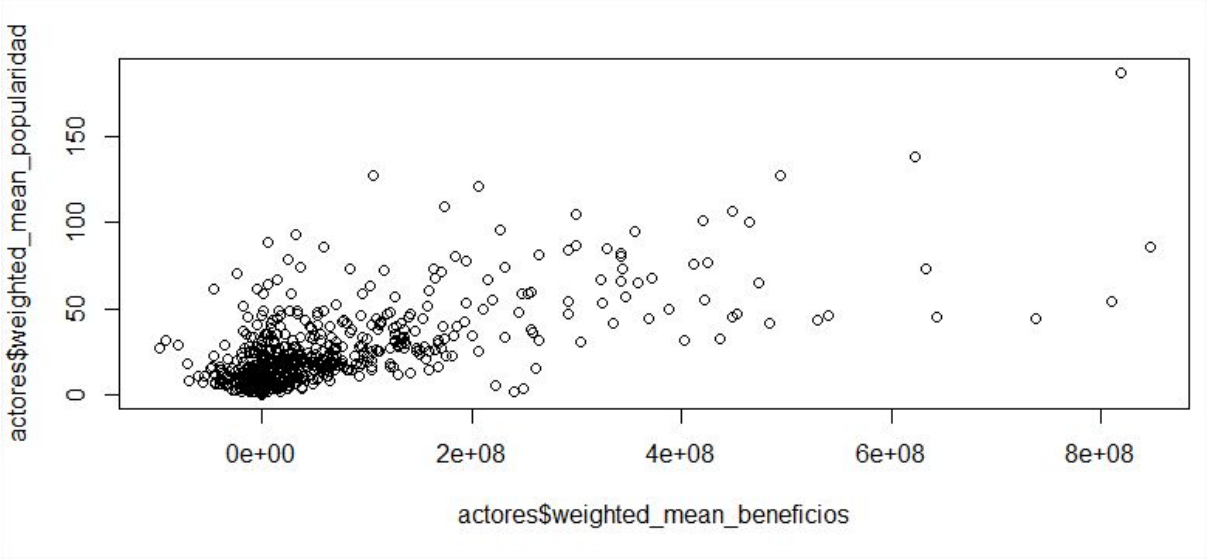
	n	weighted_mean_votos	weighted_mean_beneficios	weighted_mean_popularidad
n	1.00000000	0.1849760	0.09801526	0.1413921
weighted_mean_votos	0.18497604	1.0000000	0.20528835	0.4191839
weighted_mean_beneficios	0.09801526	0.2052884	[illegible]	0.6303387
weighted_mean_popularidad	0.14139208	0.4191839	0.63033870	1.0000000

Como podemos ver la relación entre la media de popularidad y la media de beneficios tiene la relación más alta de un 63%

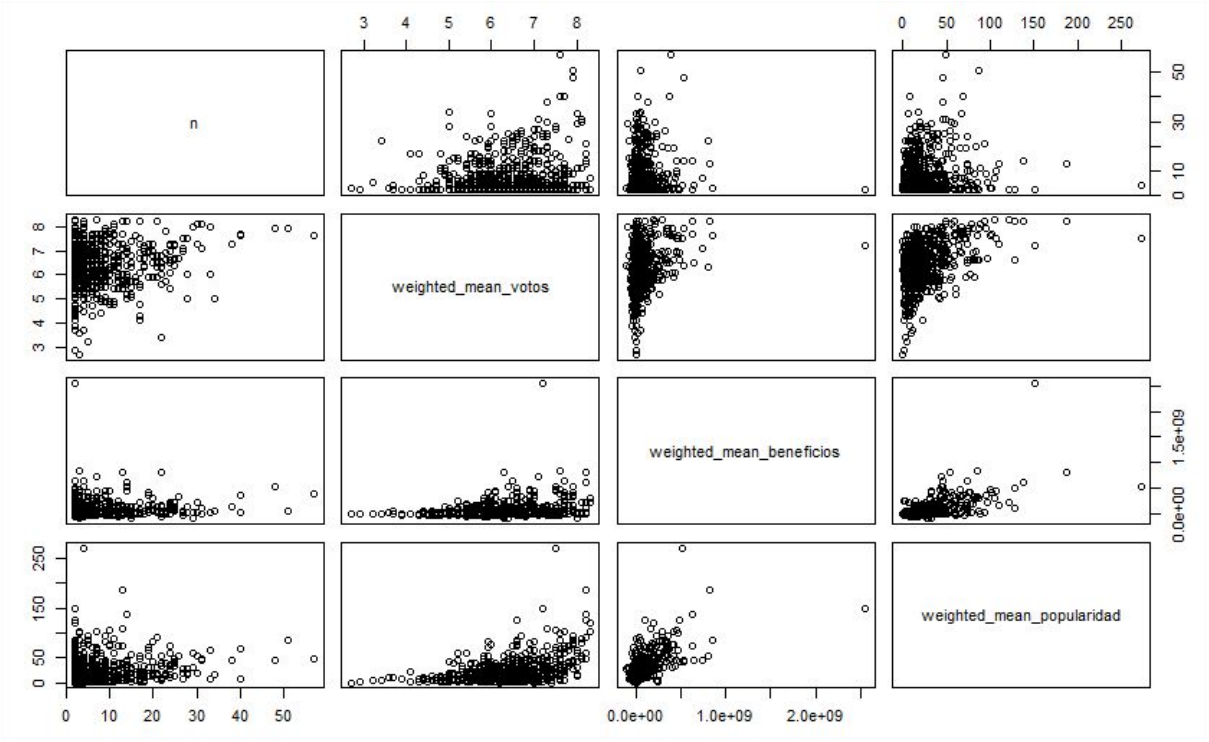
Si representamos los datos de esta relación podemos ver la siguiente gráfica



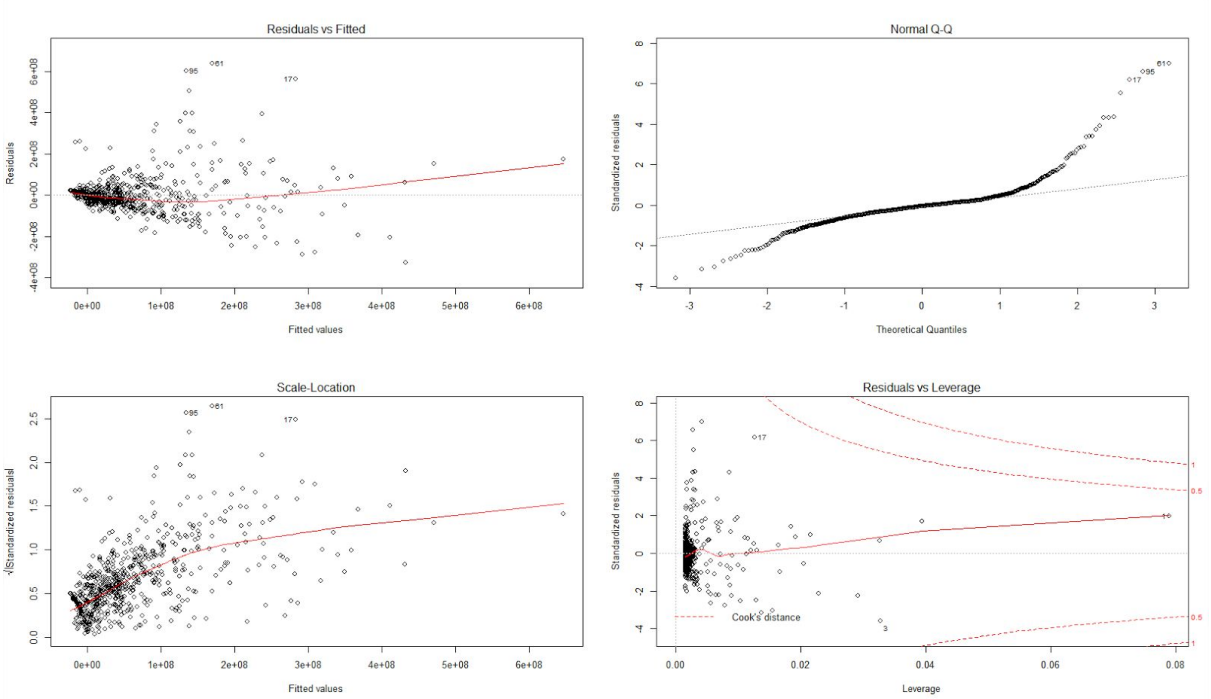
Como podemos observar existen extreme scores que no pueden desvirtuar los cálculos por este motivo hemos procedido a eliminar estos dos registros para poder tener una expedición es más factibles. Con la eliminación de estos registros la gráfica se quedaría de la siguiente forma



Y la representación de estas variables gráficamente se representa de la siguiente forma



Una vez ejecutado nuestro método de predicción obtenemos los siguientes resultados. Nos hallamos ante cuatro representaciones gráficas.
En la primera gráfica (residuals vs Fietted) muestra los datos aleatorios
Normal Q-Q otra más puntos sigan la línea que está trazada en esta gráfica indicará que es normal el conjunto datos
scale-Localtion: homosteosidad de los datos aquí podemos ver la tendencia de los datos
Residuals vs Leverage: Valores que puedan influenciar sobre la correlación, aquellos datos que sobrepasen las líneas rojas pueden variar el resultado



Si observamos la estadística de nuestro modelo podemos ver que tiene una fiabilidad de un 44% y que a pesar de esta probabilidad se indica que existe una relación entre ambos campos

```
Residuals:
       Min         1Q     Median         3Q        Max
-326003917  -35242776   -4187137   19794919  639910830

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              -23659484    5088211   -4.65    4e-06 ***
weighted_mean_popularidad  3572967     155755   22.94   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 91700000 on 671 degrees of freedom
Multiple R-squared:  0.4395,    Adjusted R-squared:  0.4387
F-statistic: 526.2 on 1 and 671 DF,  p-value: < 2.2e-16
```

Si acabas una predicción de los valores 80000 y 70000 podemos ver en la siguiente imagen los beneficios estimados dependiendo del actor hay que recordar que este modelo posee un ratio de un 44 % de aciertos por lo que no es muy fiable

```
           fit          lwr          upr
1 285813689995 261354839866 310272540123
2 250084021310 228683425457 271484617163
```

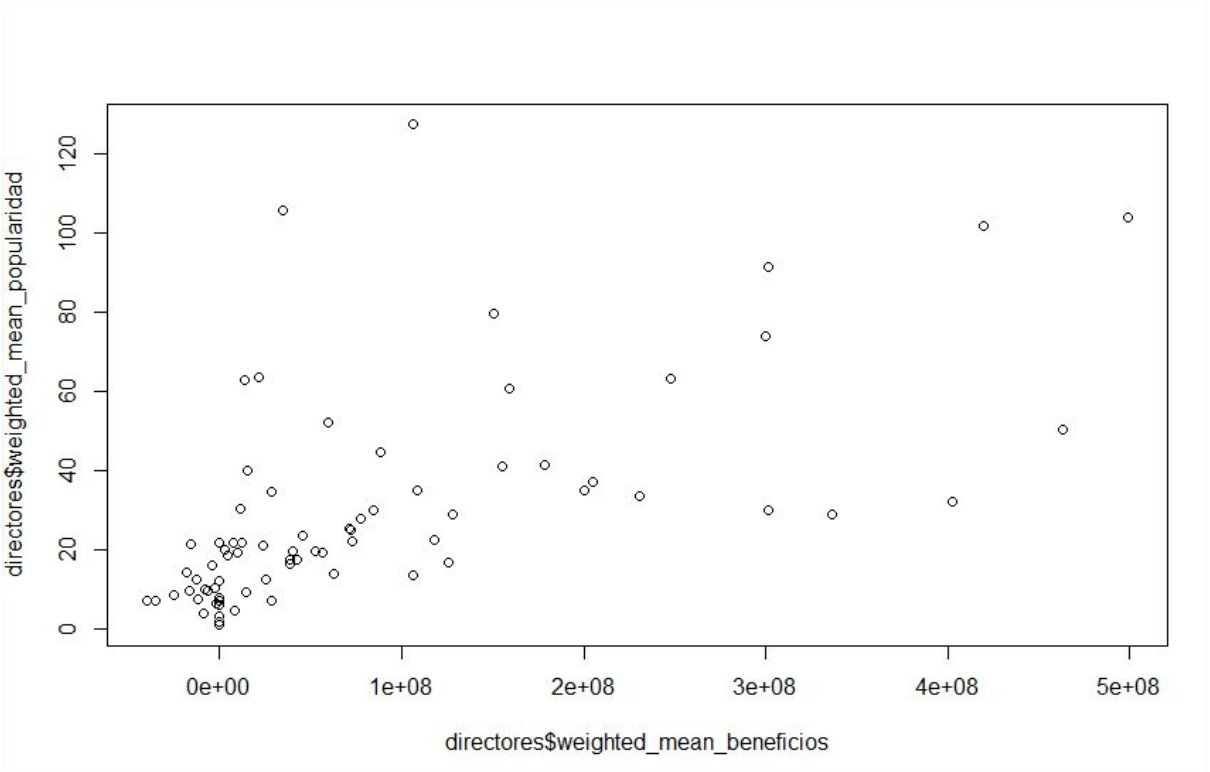
<u>CASO 2</u>*: *Directores

Para este caso hemos investigado qué variables podrían tener relaciones entre ellas

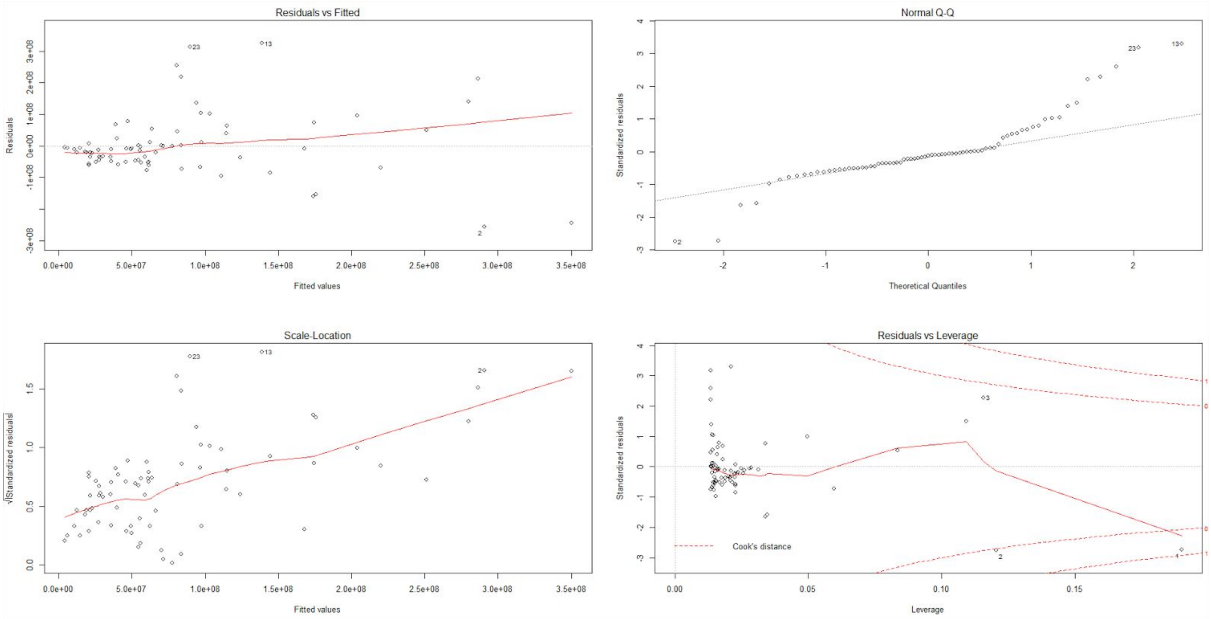
```
                          n weighted_mean_votos weighted_mean_beneficios weighted_mean_popularidad
n                 1.0000000           0.2858334               0.1785988               0.3258103
weighted_mean_votos       0.2858334           1.0000000               0.2263808               0.4933644
weighted_mean_beneficios  0.1785988           0.2263808               1.0000000               0.6000061
weighted_mean_popularidad 0.3258103           0.4933644               0.6000061               1.0000000
```

Como en el caso anterior vemos que la relación más alta es entre popularidad y beneficios pero en este caso la relación es aún menor por lo que nos va a indicar que el modelo resultante será poco fiable.

En la representación de los gráficos se podría identificar extreme score pero al ser un gran número no es eficiente eliminarlos del modelo ya que tampoco sería un modelo fiable ya que eliminamos muchos datos y no representaría adecuadamente el resultado aunque aumentara la confiabilidad del mismo



Como se puede ver en las graficas obtenidas del modelo en la gráfica residuals vs Fietted muestra los datos aleatorios y como estos se concentran en el principio de la gráfica
Normal Q-Q otra más puntos sigan la línea que está trazada en esta gráfica indicará que es normal el conjunto datos, es evidente por tanto que este conjunto de datos no sigue una distribución normal
scale-Localtion: homosteosidad de los datos aquí podemos ver la tendencia de los datos
Residuals vs Leverage: Valores que puedan influenciar sobre la correlación, aquellos datos que sobrepasen las líneas rojas pueden variar el resultado. Como se muestra en la gráfica existen puntos que sobrepasan las lineas rojas.



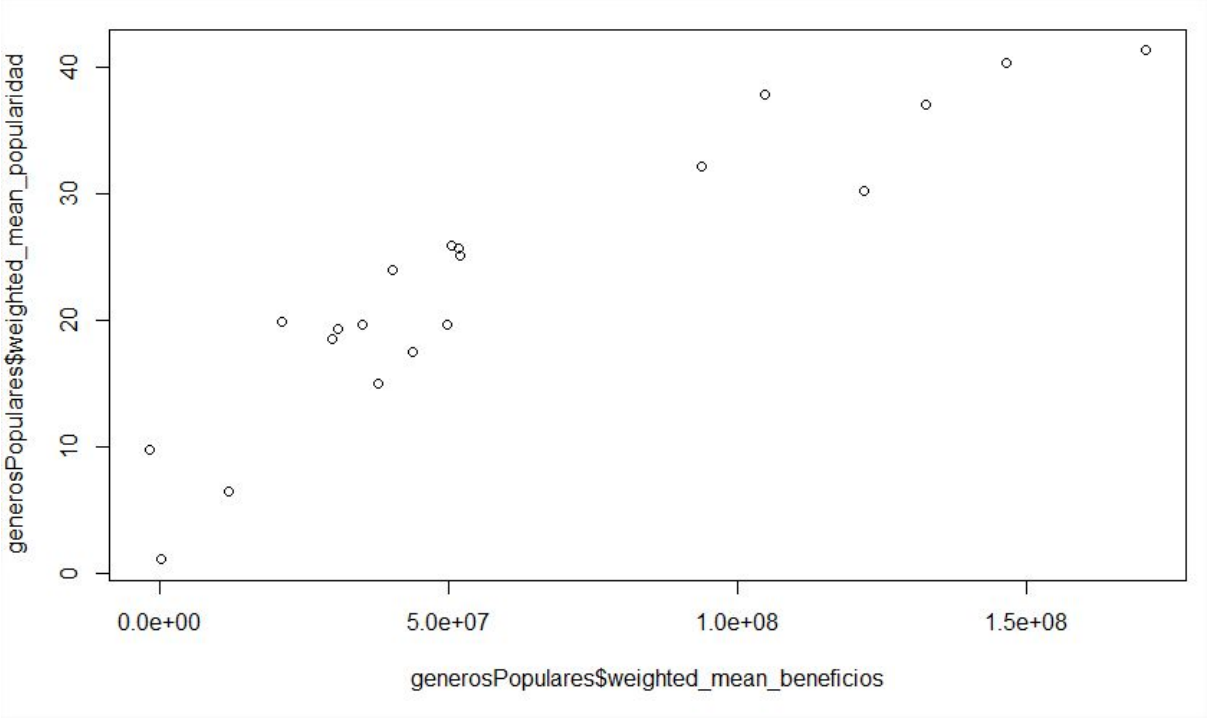
Estas gráficas nos indican que no es un buen modelo, si vemos el resumen del modelos

```
Residuals:
       Min         1Q     Median         3Q        Max
-256366802  -49200778  -12885132   17001057  324701993

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                1983178   17035024   0.116    0.908
weighted_mean_popularidad  2731056     426188   6.408 1.28e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 99310000 on 73 degrees of freedom
Multiple R-squared:   0.36,     Adjusted R-squared:  0.3512
F-statistic: 41.06 on 1 and 73 DF,  p-value: 1.277e-08
```

Podemos ver que su fiabilidad es de un 36% aun menor que el anterior por lo que sería un modelo desechable

<u>CASO 3</u>: ***Género***

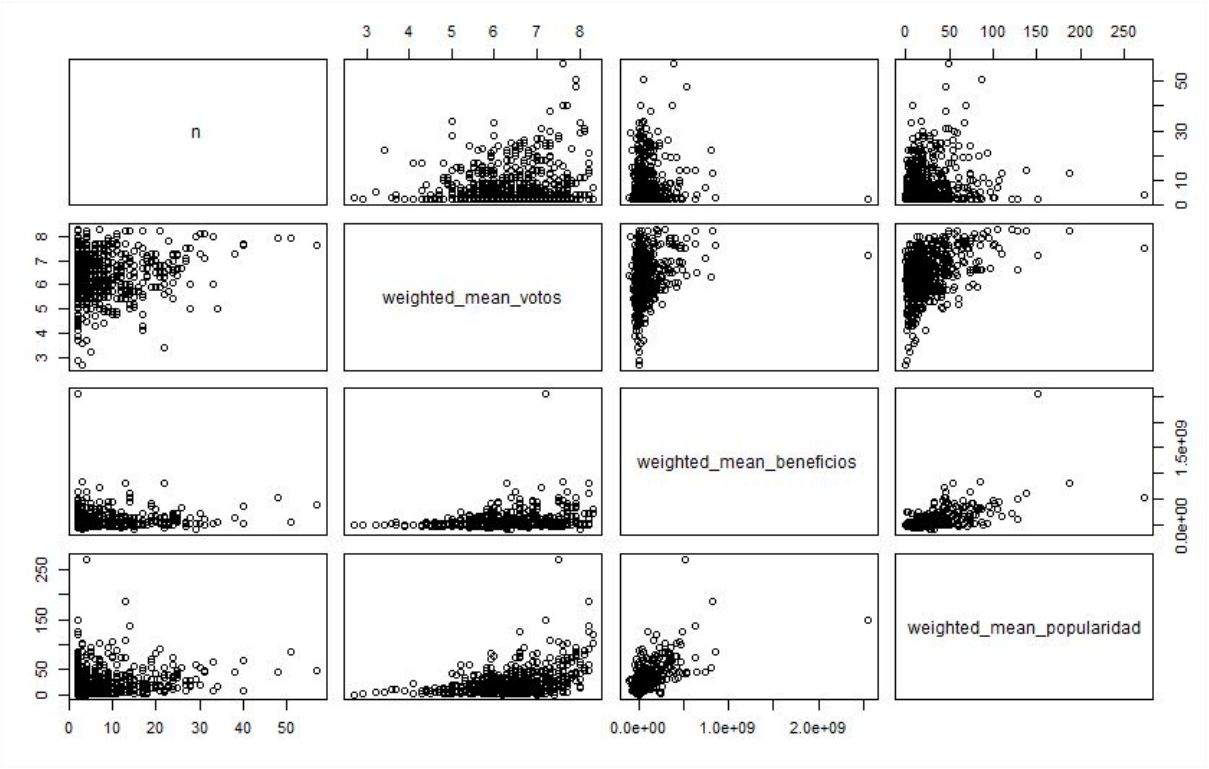
Aqui podemos ver la representación de los datos con los que vamos a crear nuestro modelo



Si observamos la relación que existe entre los datos podemos observar que entre popularidad y beneficio existe un 92% de relación, esto nos indica que podemos obtener un buen modelo

	n	weighted_mean_votos	weighted_mean_beneficios	weighted_mean_popularidad
n	1.00000000	0.04774165	0.1105882	0.2008131
weighted_mean_votos	0.04774165	1.00000000	0.1539515	0.2080590
weighted_mean_beneficios	0.11058816	0.15395149	1.0000000	0.9180353
weighted_mean_popularidad	0.20081313	0.20805898	0.9180353	[illegible]

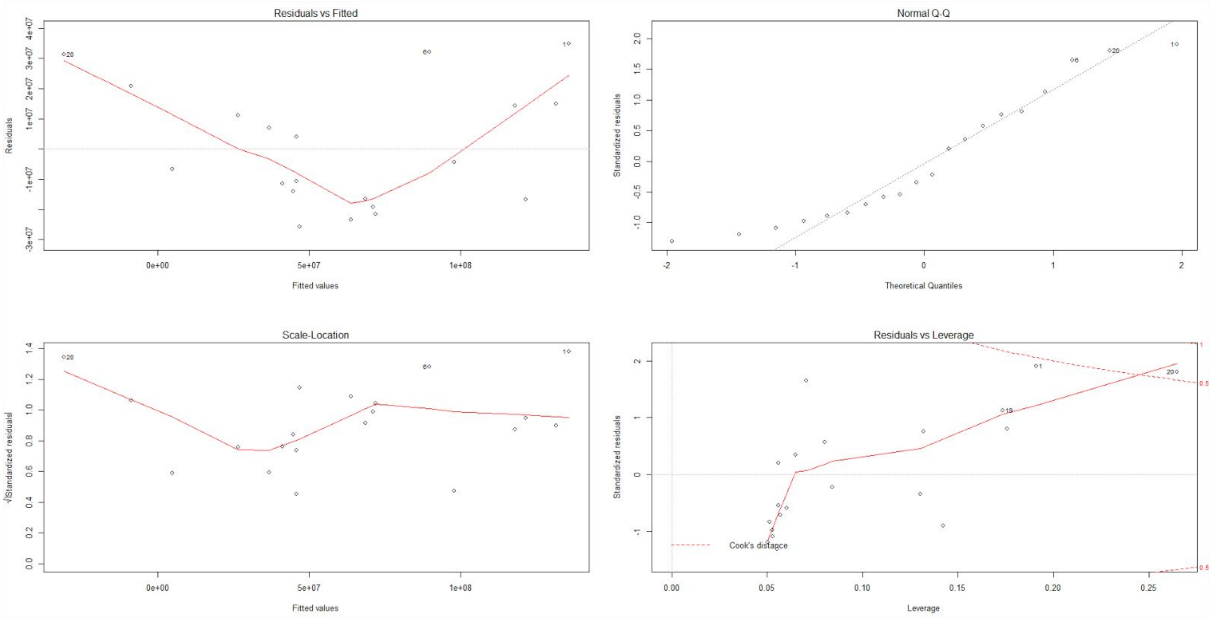
En la siguiente imagen podemos ver gráficamente esta relación, los gráficos de esta relación concentran los puntos más que el resto



Como se puede ver en las graficas obtenidas del modelo, en la gráfica residuals vs Fietted muestra los datos aleatorios y como estos se concentran se puede observar que los puntos se encuentran repartidos por toda la gráfica indicando alto nivel de aleatoriedad

Normal Q-Q En esta grafica se observa la normalidad de los datos. Como se ve en la gráfica aunque se acerca no es una distribución normal, esto es debido a que la distribución de popularidad si es normal pero la distribución de los beneficios no

Residuals vs Leverage: Valores que puedan influenciar sobre la correlación, aquellos datos que sobrepasen las líneas rojas pueden variar el resultado. Como se muestra en la gráfica existe un punto que puede variar nuestro resultado.



Si vemos el resumen estadístico de nuestro modelo podemos ver que tiene una confianza de un 84%

```
Residuals:
      Min        1Q    Median        3Q       Max
-25796317 -16612598  -5451789  14511131  34803279

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               -35705123   10845660  -3.292  0.00405 **
weighted_mean_popularidad   4146552     422117   9.823 1.17e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20260000 on 18 degrees of freedom
Multiple R-squared:  0.8428,    Adjusted R-squared:  0.8341
F-statistic:  96.5 on 1 and 18 DF,  p-value: 1.174e-08
```

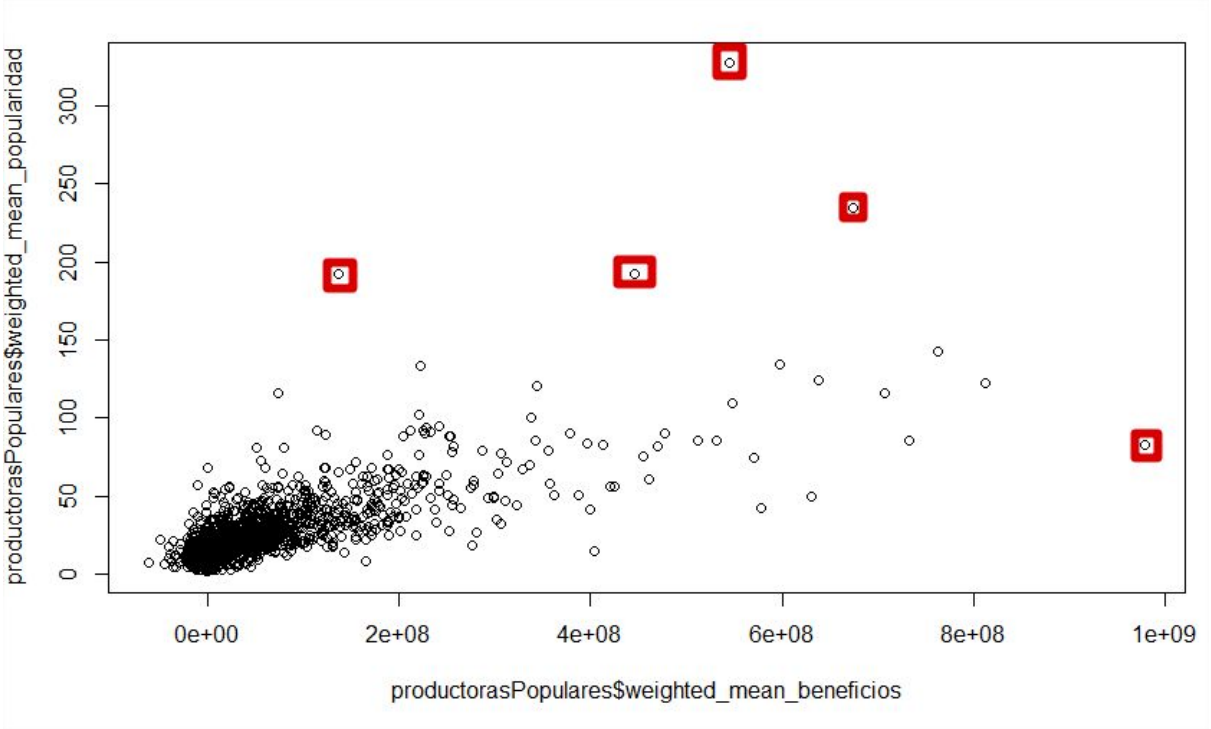
Si realizamos una predicción de los datos 70000 y 80000 obtenemos las siguientes resultados

```
        fit          lwr          upr
1 331688433972 260762286581 402614581363
2 290222916585 228165125316 352280707854
```

fit sera el valor predicho lwr el valor mínimo y upr el valor máximo

<u>CASO 3</u>*: *Productoras

Aqui podemos ver la representación de los datos con los que vamos a crear nuestro modelo

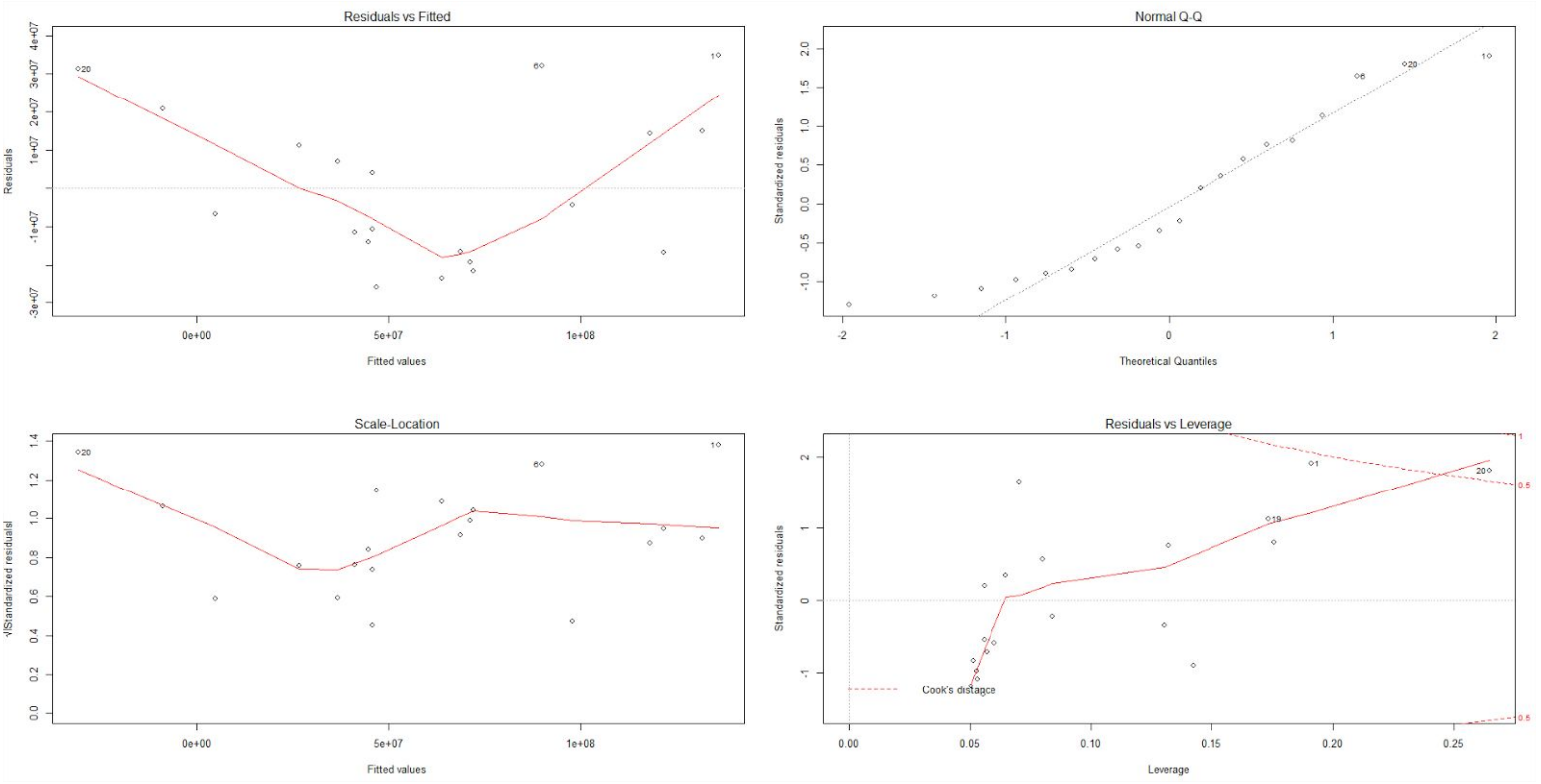


	n	weighted_mean_votos	weighted_mean_beneficios	weighted_mean_popularidad
n	1.00000000	0.06205902	0.06596785	0.05164444
weighted_mean_votos	0.06205902	1.00000000	0.15612805	0.24238322
weighted_mean_beneficios	0.06596785	0.15612805	1.00000000	0.73225603
weighted_mean_popularidad	0.05164444	0.24238322	0.73225603	1.00000000

si eliminamos los puntos señalados (Extremes scores) mejora un poco la relación entre las variables

	n	weighted_mean_votos	weighted_mean_beneficios	weighted_mean_popularidad
n	1.00000000	0.06273065	0.07220252	0.06526015
weighted_mean_votos	0.06273065	1.00000000	0.14653471	0.24619687
weighted_mean_beneficios	0.07220252	0.14653471	1.00000000	0.75339389
weighted_mean_popularidad	0.06526015	0.24619687	0.75339389	1.00000000

De esta forma nos quedamos con una relación de un 75 % esto nos indica que es probable que el modelo generado sea fiable



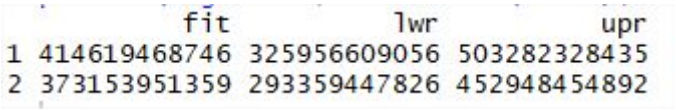
Una vez creado el modelo podemos ver las siguientes gráficas del modelo

En la gráfica residuals vs Fietted muestra los datos aleatorios y como estos se concentran se puede observar que los puntos se encuentran repartidos por toda la gráfica indicando alto nivel de aleatoriedad
Normal Q-Q En esta grafica se observa la normalidad de los datos. Como se ve en la gráfica aunque se acerca no es una distribución normal.
Residuals vs Leverage: Valores que puedan influenciar sobre la correlación, aquellos datos que sobrepasen las líneas rojas pueden variar el resultado. Como se muestra en la gráfica existe un punto (20) que puede variar nuestro resultado.

A continuación si vemos el resumen estadístico del modelo podemos decir:

```
Residuals:
      Min        1Q    Median        3Q       Max
-25796317 -16612598  -5451789  14511131  34803279

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               -35705123   10845660  -3.292  0.00405 **
weighted_mean_popularidad   4146552     422117   9.823 1.17e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20260000 on 18 degrees of freedom
Multiple R-squared:  0.8428,    Adjusted R-squared:  0.8341
F-statistic:  96.5 on 1 and 18 DF,  p-value: 1.174e-08
```

Que este modelo tiene una fiabilidad de un 84%, así podremos decir que es un modelo bueno para realizar predicciones. Si realizamos una predicción de los valores 100000 y 90000



	fit	lwr	upr
1	414619468746	325956609056	503282328435
2	373153951359	293359447826	452948454892

Hay que recordar que el modelo posee un 84% de acierto.

Conclusiones:

Podemos decir que los modelos generados sobre productoras y generos son las mas fiables y las que se propondrán para su uso. De esta forma el usuario tener una ayuda para decidir en el momento de ver una película o incluso a los trabajadores del sector a elegir en qué productora o genero quieren desarrollar sus tareas.
Por otro lado la práctica ha resultado costosa de desarrollar ya que no teníamos muy claro que es lo que se pedía exactamente, hemos echado en falta algún ejemplo resuelto para que nos sirviera de guia. Esperamos que hayamos acertado en el significado de las preguntas

El código R para esta parte ha sido el siguiente:

```
## Correlacion actores

#Para mejorar la confianza del algoritmo vamos a quitar algunos extreme score
plot(actores$weighted_mean_beneficios, actores$weighted_mean_popularidad)
posicionBene <- which(actores$weighted_mean_beneficios >900000000)
registroAct <-actores[posicionBene,]
actores <- actores[-c(posicionBene),]
posicionPopu <- which(actores$weighted_mean_popularidad >200)
actores <- actores[-c(posicionPopu),]
plot(actores$weighted_mean_beneficios, actores$weighted_mean_popularidad)
datosactores <- actores[,-c(1)]
#podemos ver que existe correlacion de la popularidad con los beneficios de un 63%
cor(datosactores)
#la representacion de las correlaciones
pairs(datosactores)
plot(actores$weighted_mean_beneficios, actores$weighted_mean_popularidad)
regresion <- lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data = actores)
# Errores errores estimados: 186419
par(mfrow=c(2,2))
plot(lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data =actores))
#residuals vs Fietted: muestra los datos aleatorios
#Normal Q-Q se si siguen la raya es que son normales
# scale-Localtion: Busca elasticidad muestra la tendencia
#Residuals vs Leverage realiza una estimacion todos los datos que no estan cerca de la linea no
estan dando valor

summary(regresion)
#Bondad de ajuste, esta es signidicativa H0 no hay correlacion, hay una correlacion diferende de 0
anova(regresion)
```

```
# si queremos la representaciÃ³n de los datos
plot(actores$weighted_mean_beneficios, actores$weighted_mean_popularidad)
abline(regresion)
datos <- data.frame(weighted_mean_popularidad =(c(80000,70000)))
predict(regresion,  datos)
predict(regresion, data.frame(datos), level = 0.95, interval = "confidence")



## Corelacion directores
#Para mejorar la confianza del algoritmo vamos a quitar algunos extreme score
plot(directores$weighted_mean_beneficios, directores$weighted_mean_popularidad)


datosdirectores <- directores[,-c(1)]
#podemos ver que existe correlacion de la popularidad con los beneficios de un 63%
cor(datosdirectores)
#la representacion de las correlaciones
pairs(datosactores)
plot(directores$weighted_mean_beneficios, directores$weighted_mean_popularidad)
regresion <- lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data = directores)
par(mfrow=c(2,2))
plot(lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data =directores))
#residuals vs Fietted: muestra los datos aleatorios
#Normal Q-Q se si siguen la raya es que son normales
# scale-Localtion: Busca elasticidad muestra la tendencia
#Residuals vs Leverage realiza una estimacion todos los datos que no estan cerca de la linea no
estan dando valor

summary(regresion)
#Bondad de ajuste, esta es signidicativa H0 no hay correlacion, hay una correlacion diferende de 0
anova(regresion)
# si queremos la representaciÃ³n de los datos
plot(actores$weighted_mean_beneficios, actores$weighted_mean_popularidad)
datos <- data.frame(weighted_mean_popularidad =(c(80000,70000)))
predict(regresion,  datos)
predict(regresion, data.frame(datos), level = 0.95, interval = "confidence")


## Corelacion genero
#Para mejorar la confianza del algoritmo vamos a quitar algunos extreme score
plot(generosPopulares$weighted_mean_beneficios,
generosPopulares$weighted_mean_popularidad)
datosgeneros <- generosPopulares[,-c(1)]
cor(datosgeneros)
#la representacion de las correlaciones
pairs(datosactores)
plot(generosPopulares$weighted_mean_beneficios,
generosPopulares$weighted_mean_popularidad)
regresion <- lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data =
generosPopulares)
par(mfrow=c(2,2))
plot(lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data =generosPopulares))
#residuals vs Fietted: muestra los datos aleatorios
#Normal Q-Q se si siguen la raya es que son normales
# scale-Localtion: Busca elasticidad muestra la tendencia
#Residuals vs Leverage realiza una estimacion todos los datos que no estan cerca de la linea no
```

```
estan dando valor

summary(regresion)
#Bondad de ajuste, esta es signidicativa H0 no hay correlacion, hay una correlacion diferende de 0
anova(regresion)
# si queremos la representaciÃ³n de los datos
plot(generosPopulares$weighted_mean_beneficios,
generosPopulares$weighted_mean_popularidad)
datos <- data.frame(weighted_mean_popularidad =(c(80000,70000)))
predict(regresion,  datos)
predict(regresion, data.frame(datos), level = 0.95, interval = "confidence")



## Corelacion Productora
#Para mejorar la confianza del algoritmo vamos a quitar algunos extreme score
plot(productorasPopulares$weighted_mean_beneficios,
productorasPopulares$weighted_mean_popularidad)

posicionBene <- which(productorasPopulares$weighted_mean_beneficios >900000000)
productorasPopulares <- productorasPopulares[-c(posicionBene),]
posicionPopu <- which(productorasPopulares$weighted_mean_popularidad >142.46500)
productorasPopulares <- productorasPopulares[-c(posicionPopu),]
datosproductora <- productorasPopulares[,-c(1)]
cor(datosproductora)
#la representacion de las correlaciones
pairs(datosactores)
plot(generosPopulares$weighted_mean_beneficios,
generosPopulares$weighted_mean_popularidad)
regresion <- lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data =
generosPopulares)
par(mfrow=c(2,2))
plot(lm(weighted_mean_beneficios ~ weighted_mean_popularidad, data =generosPopulares))
#residuals vs Fietted: muestra los datos aleatorios
#Normal Q-Q se si siguen la raya es que son normales
# scale-Localtion: Busca elasticidad muestra la tendencia
#Residuals vs Leverage realiza una estimacion todos los datos que no estan cerca de la linea no
estan dando valor

summary(regresion)
#Bondad de ajuste, esta es signidicativa H0 no hay correlacion, hay una correlacion diferende de 0
anova(regresion)
# si queremos la representación de los datos
plot(generosPopulares$weighted_mean_beneficios,
generosPopulares$weighted_mean_popularidad)
datos <- data.frame(weighted_mean_popularidad =(c(100000,90000)))
predict(regresion,  datos)
predict(regresion, data.frame(datos), level = 0.95, interval = "confidence")
```

4. *Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.*

Para realizar todo el análisis propuesto en los apartados anteriores, se ha utilizado código fuente en R.

El código fuente se encuentra en el repositorio git que indicamos a continuación, en un fichero con nombre :
https://github.com/Raquelmdcd/peliculas

Bibliografía.

[1] Wikipedia, Criterio de Cramér-von Mises
https://es.wikipedia.org/wiki/Criterio_de_Cram%C3%A9r-von_Mises
[2] Wikipedia, Test de Shapiro - Wilk
https://es.wikipedia.org/wiki/Test_de_Shapiro%E2%80%93Wilk
[3] Wikipedia, Prueba χ^2 de Pearson
https://es.wikipedia.org/wiki/Prueba_%CF%87%C2%B2_de_Pearson
[4] Viva el software libre
http://vivaelsoftwarelibre.com/test-de-kolmogorov-smirnov-en-r/
[6] Real Statistics Using Excel
http://www.real-statistics.com/tests-normality-and-symmetry/statistical-tests-normality-symmetry/lilliefors-test-normality/
[7] Wikipedia, Shapiro–Francia test
https://en.wikipedia.org/wiki/Shapiro%E2%80%93Francia_test
[8] Las pruebas paramétricas 1
https://www.uv.es/~friasnav/SupuestosParametrica.pdf
[9] Contrastes de normalidad
http://www.ub.edu/aplica_infor/spss/cap5-6.htm
[10] El estadístico de Anderson
https://support.minitab.com/es-mx/minitab/18/help-and-how-to/statistics/basic-statistics/supporting-topics/normality/the-anderson-darling-statistic/
[11] Contrastes de hipótesis de una población 1
https://www.uoc.edu/in3/emath/docs/CH_1Pob.pdf