

Análisis de datos de Twitch

Víctor López Martínez

2024-03-13

Contents

1	Introducción	1
2	Análisis previo	1
3	Análisis de Componentes Principales	5
4	Análisis de cluster	7
4.1	K-means	8
4.2	Análisis cluster jerarquizado	12
5	Análisis Discriminante	15
5.1	Análisis Discriminante Cuadrático (QDA)	17
6	Regresión Lineal Múltiple	17
6.1	Transformaciones de los datos	21
6.2	Estimación de los parámetros del modelo RLM y métodos de selección de regresores	23
6.3	Validación del modelo	29
6.4	Inferencias en el modelo RLM	36
7	Conclusiones	38

1 Introducción

Twitch es una de las plataformas de retransmisión en directo más importantes de Internet. Entre 2019 y 2020 la plataforma sufrió un importante crecimiento debido al decrecimiento de plataformas competidoras como Mixer y a otros acontecimientos como la pandemia de Covid-19 y el consiguiente confinamiento en gran parte del mundo. Esto último provocaría que el principal medio de entretenimiento se encontrara en Internet y especialmente en aquellos sitios que emitían contenido de su interés prácticamente las 24 horas del día.

El fichero `twitchdata-update.csv` contiene un set de datos con 11 columnas en las que se registraron datos de los 1000 canales más exitosos de Twitch entre 2019 y 2020. A partir de él vamos a tratar de averiguar cuales son las características o variables más preponderantes a la hora de determinar cuales son los canales más exitosos, cuales son los que destacan en distintos aspectos (picos de audiencia, aumento de seguidores, etc). Para ello, se aplicarán distintas técnicas estadística multivariante que nos van a permitir reducir la dimensionalidad de las variables, hacer agrupaciones y averiguar las tendencias que siguen los datos.

2 Análisis previo

En primer lugar haremos un breve repaso de lo que representa cada variable:

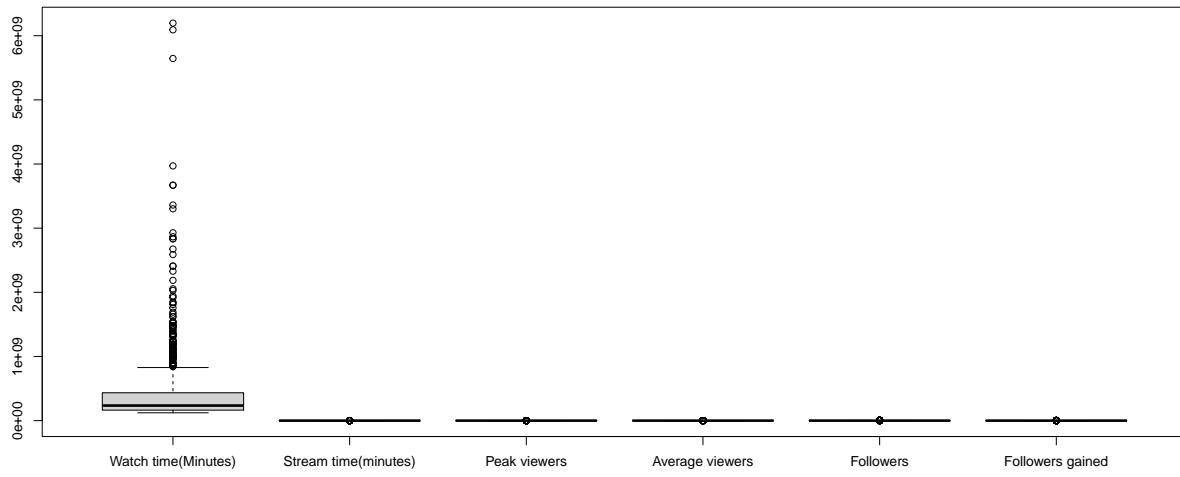
- **Channel:** Nombre del canal de Twitch (chr)

- **Watch time (minutes)**: Suma de los minutos de visualización del canal entre todos los usuarios durante un año (num).
- **Stream time (minutes)**: Registro del tiempo (en minutos) que el canal ha estado emitiendo contenido durante ese año (num)
- **Peak viewers**: Máxima cantidad de espectadores en directo (num)
- **Average viewers**: Media de espectadores en directo del canal (num)
- **Followers**: Número de seguidores del canal al final del año (num)
- **Followers gained**: Seguidores ganados durante ese año (num)
- **Views gained**: Número de visitas durante ese año (num)
- **Partnered**: Si el canal es socio de Twitch, es decir, si cumple unos requisitos mínimos de horas en directo, mínimo de días diferentes que ha estado retransmitiendo o si mantiene una media mínima de espectadores (bool)
- **Mature**: Si el contenido del canal es solo apto para adultos (bool)
- **Language**: Idioma del canal (bool)

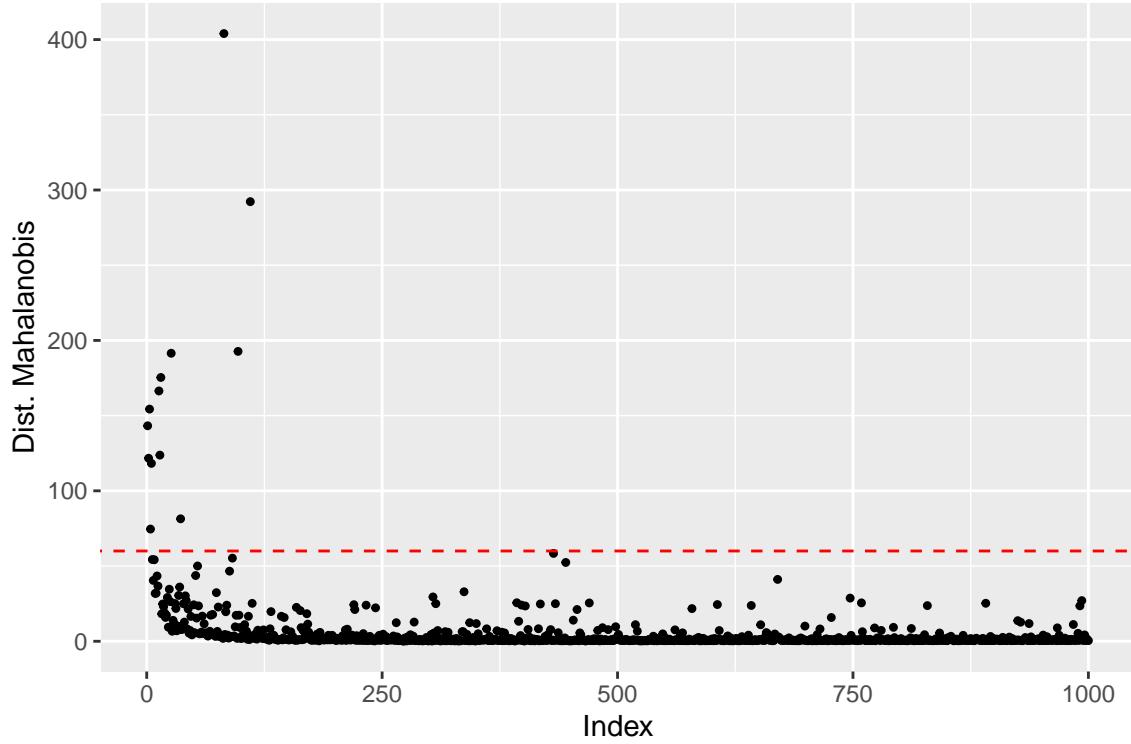
```

##      Channel      Watch time(Minutes) Stream time(minutes) Peak viewers
## Length:1000      Min.    :1.222e+08   Min.    : 3465      Min.    : 496
## Class :character 1st Qu.:1.632e+08   1st Qu.: 73759     1st Qu.: 9114
## Mode  :character Median :2.350e+08   Median :108240     Median : 16676
##                  Mean   :4.184e+08   Mean   :120515     Mean   : 37065
##                  3rd Qu.:4.337e+08   3rd Qu.:141844     3rd Qu.: 37570
##                  Max.   :6.196e+09   Max.   :521445     Max.   :639375
## Average viewers   Followers   Followers gained   Views gained
## Min.    : 235    Min.    : 3660    Min.    :-15772     Min.    : 175788
## 1st Qu.: 1458   1st Qu.: 170546   1st Qu.: 43758     1st Qu.: 3880602
## Median  : 2425   Median : 318063   Median : 98352     Median : 6456324
## Mean    : 4781   Mean   : 570054   Mean   : 205519     Mean   : 11668166
## 3rd Qu.: 4786   3rd Qu.: 624332   3rd Qu.: 236131     3rd Qu.: 12196762
## Max.   :147643   Max.   :8938903   Max.   :3966525     Max.   :670137548
## Partnered        Mature       Language
## Mode  :logical   Mode  :logical  Length:1000
## FALSE:22         FALSE:770     Class :character
## TRUE  :978        TRUE :230      Mode  :character
##
```

Como punto de partida, realizaremos un estudio inicial para comprender cuál es el comportamiento de los datos del dataset, el tipo de datos que almacenan y su rango de valores.



Observamos como las varianzas son distintas en cada variable numérica y con escalas muy diferentes. Teniendo rangos de valores tan dispares en cada columna, podría ser interesante escalarlos a la hora de aplicar ciertas técnicas de clasificación.

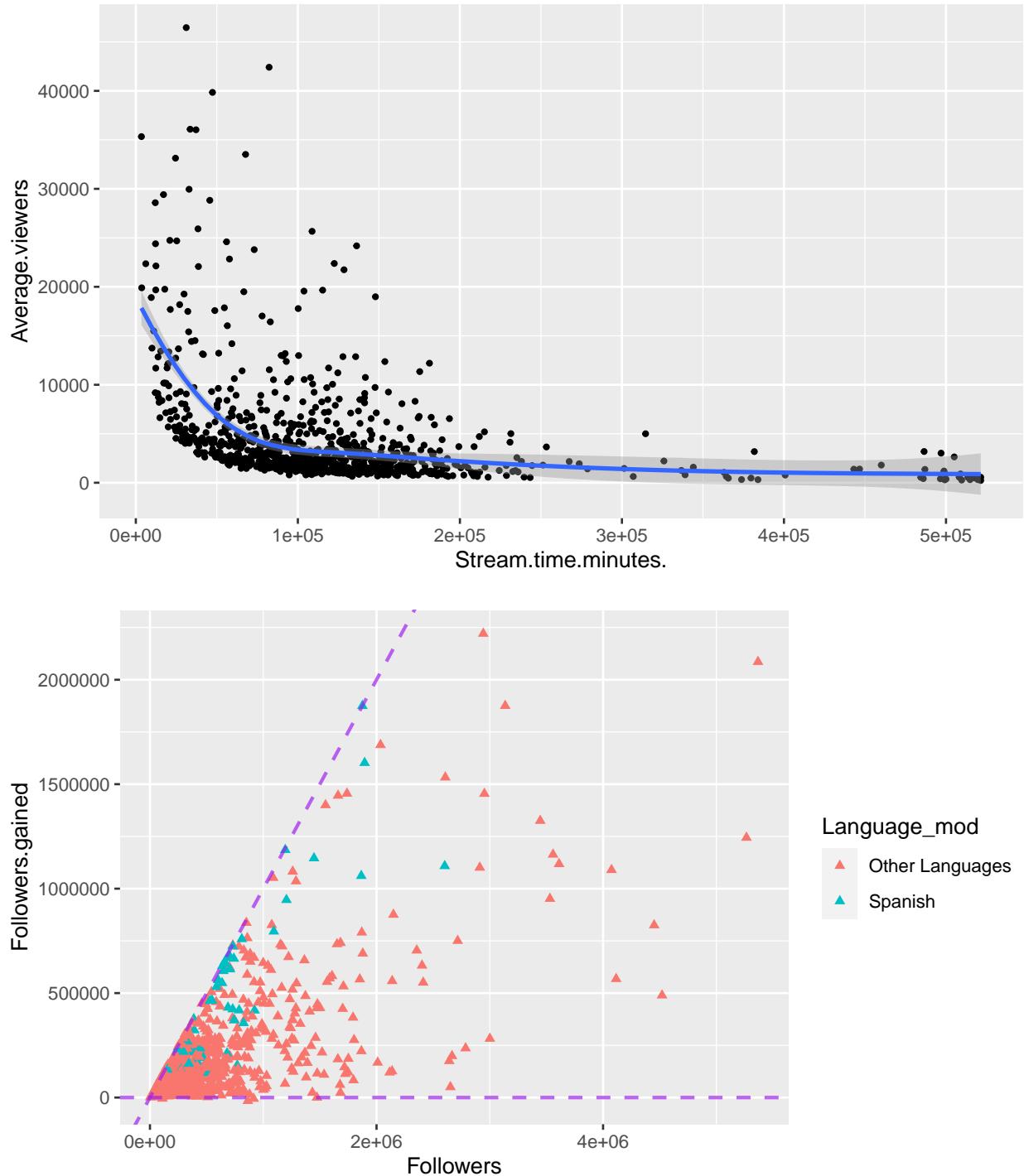


Usando las columnas numéricas podemos calcular las distancias de Mahalanobis y ver cual es el canal de Twitch con las estadísticas más raras. En este caso `dota2ti` tiene los datos más alejados de la media. Se trata de un canal que retransmite directos esporádicos de torneos del videojuego Dota 2 que tienen mucho éxito a pesar de la poca actividad en la plataforma (lo que explica su rareza).

```
## [1] "dota2ti"
```

Con el gráfico de distancias de Mahalanobis podríamos identificar los datos atípicos para poder ser eliminados

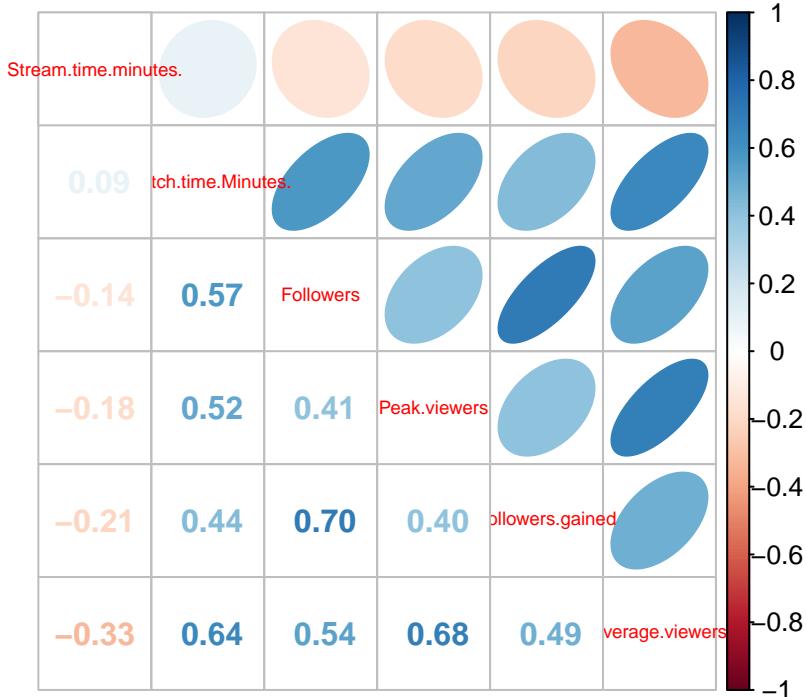
de modo que no obstaculicen el análisis del resto. Por ejemplo, podríamos proponer eliminar todas aquellas observaciones con una distancia mayor a 60. Con este filtro se eliminarían 13 observaciones (teniendo en cuenta que partimos de 1000 observaciones).



Las nubes de puntos nos pueden ayudar a apreciar como se comportan las variables al enfrentarlas con el resto. De aquí se sacan conclusiones como que existe una tendencia de los datos a decrecer la media de espectadores del canal según va aumentando las horas de retransmisión que acumula o la relación positiva entre los seguidores del canal a final del año y el aumento de esta cifra durante este mismo periodo.

Nótese de este último gráfico la forma de punta de flecha que tiene la nube de puntos. La envolvente inferior que acota los puntos sería la recta horizontal que pasa por el origen. Los puntos cercanos a esta recta serán aquellos canales que obtienen peor proporción de seguidores ganados respectos sus seguidores finales. En cambio, los puntos cercanos a la envolvente superior serán aquellos que mejor rendimiento han conseguido (la mayoría de canales hispanohablantes se encuentran cerca de esta recta). Queda decir que la ecuación de esta envolvente sería $y = x$ (bisectriz del primer cuadrante), ya que no es posible que un canal obtenga una cifra mayor de seguidores ganados durante el año que la de seguidores con la que finaliza. Por ende, los canales que se encuentren sobre la recta serán los que iniciaron este año con 0 seguidores.

OBSERVACIÓN: Las ganancias de seguidores también pueden ser negativas porque hayan perdido más seguidores.



Ahora usando la matriz de correlaciones podemos llegar a la conclusión de que existe una relación inversamente proporcional entre el tiempo de retransmisión de un canal y el resto de las variables (salvo para el tiempo de visualización de su canal, con el que tiene una correlación prácticamente nula (serían variables independientes).

3 Análisis de Componentes Principales

Nuestra primera decisión para calcular las componentes principales será si usar la matriz de covarianzas o la matriz de correlaciones. Como hemos apreciado en el análisis previo, parece ser que las variables tienen rangos muy diferentes, por lo que optaremos por la segunda opción.

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6
Stan..dev	1.7937403	1.0409409	0.8940181	0.6463896	0.5181791	0.4618879
Var	3.2175041	1.0835580	0.7992683	0.4178195	0.2685096	0.2133404
Prop..Var.	0.5362507	0.1805930	0.1332114	0.0696366	0.0447516	0.0355567
Cum..Prop	0.5362507	0.7168437	0.8500551	0.9196917	0.9644433	1.0000000

Para decidir el número de componentes vamos a emplear la **regla de Rao**, de modo que solo nos quedaremos

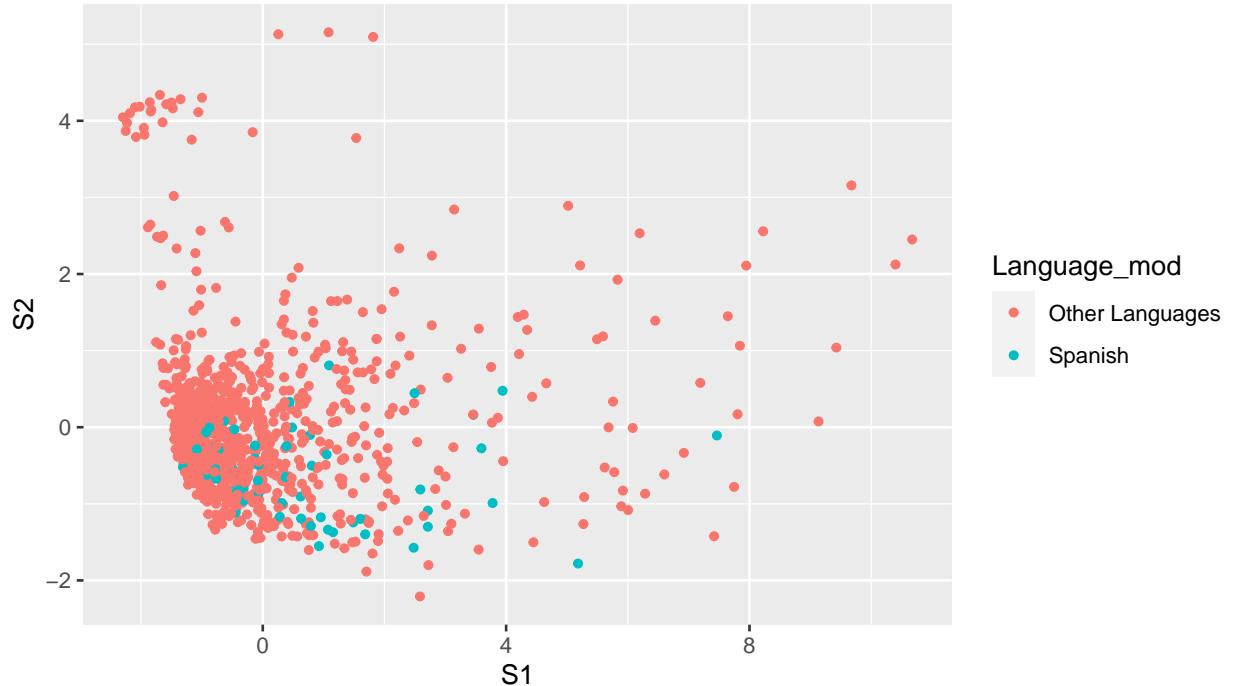
con aquellas componentes que tengan una varianza o valor propio mayor que la variabilidad mínima de las variables originales. Puesto que estamos usando variables estandarizadas, nos quedaremos con aquellas componentes que tengan un valor propio (o varianza) $\lambda \geq 1$. Siguiendo este criterio nos quedamos con las 2 primeras componentes principales.

Table 2: Cargas en cada componente

	Comp.1	Comp.2
Watch.time.Minutes.	0.4302860	0.4174647
Stream.time.minutes.	-0.1582771	0.8946743
Peak.viewers	0.4213678	-0.0239920
Average.viewers	0.4801438	-0.1114862
Followers	0.4489693	0.1007145
Followers.gained	0.4244290	-0.0461836

Analizando el valor de las cargas para la primera componente, las variables que más influyen son **Average.viewers**, **Followers**, **Watch.time.Minutes**, **Followers.Gained** y **Peak.Viewers** (influyen positivamente todas de una forma positiva). Por lo tanto, Y_1 nos indicará lo exitoso que es el canal.

Para la segunda componente, las cargas nos indican que **Stream.time.minutes** y **Watch.time.Minutes** son las variables que más influyen. Por ello, esta componente podría indicar el compromiso que hay entre el propio streamer y su audiencia (si el canal acumula mucho tiempo en directo y sus espectadores acumulan mucho tiempo viéndolo, entonces hay compromiso). Los canales que tengan valores grandes en Y_2 serán por lo general canales que llevan tiempo en la plataforma y que han consolidado una comunidad de Twitch comprometida con el canal.



Si representamos en un diagrama de dispersión las puntuaciones de las dos primeras componentes, podemos ver una gran nube de puntos aglutinada entorno al origen (0,0). En la parte superior izquierda del gráfico podríamos encontrar un grupo relativamente denso y lejano del resto de puntos. Por su ubicación en el gráfico diremos que se trata de un grupo de canales no muy exitosos que destaca por el número de minutos retransmitidos en la plataforma y los minutos que acumulan sus espectadores (canales con una comunidad

consolidada). A la derecha del gráfico los puntos se encuentran mucho más dispersos y corresponderán a los canales que tienen mayor éxito en Twitch.

Usando la matriz de saturaciones podemos calcular como de bien están representadas las variables originales en cada componente:

Table 3: Matriz de saturaciones

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6
Watch.time.Minutes.	0.7718214	0.4345561	0.1609589	0.3429415	0.1044599	0.2470309
Stream.time.minutes.	-0.2839080	0.9313031	-0.0014940	-0.1736977	0.0190310	-0.1467498
Peak.viewers	0.7558243	-0.0249743	0.4709892	-0.4210325	-0.1285013	0.1117766
Average.viewers	0.8612533	-0.1160505	0.3069575	0.1545024	0.1106162	-0.3382975
Followers	0.8053343	0.1048378	-0.4381337	0.0771845	-0.3733157	-0.0562357
Followers.gained	0.7613154	-0.0480744	-0.5151149	-0.2508843	0.2985339	0.0260571

Vemos como la variable que mejor queda representada con la primera componente es `Average.Viewers`, mientras que `Stream.time.minutes` es la que peor representa quedando (se obtiene muy poca información de ella con Y_1). Con la segunda componente conseguimos representar mucho mejor `Average.Viewers` que con la primera componente.

Sin embargo, lo que nos interesa saber es cuanta información de cada variable se conserva con la representación de las dos primeras componentes principales. Tendremos que calcular las **comunalidades**:

```
## Watch.time.Minutes. Stream.time.minutes.      Peak.viewers
##          0.7845473           0.9479292          0.5718941
## Average.viewers          Followers   Followers.gained
##          0.7552249           0.6595544          0.5819122
```

Table 4: Comunalidades

	Comp.1	Comp.2	Comunalidad
Watch.time.Minutes.	0.5957082	0.1888390	0.7845473
Stream.time.minutes.	0.0806037	0.8673255	0.9479292
Peak.viewers	0.5712704	0.0006237	0.5718941
Average.viewers	0.7417572	0.0134677	0.7552249
Followers	0.6485634	0.0109910	0.6595544
Followers.gained	0.5796011	0.0023111	0.5819122

Por un lado tenemos que la variable de la que menos información se pierde es `Average.Viewers`, que conserva un 94.8 % de la información. No obstante, en otras variables como `Peak.viewers` y `Followers.gained` permanece menos de un 60 % de la información.

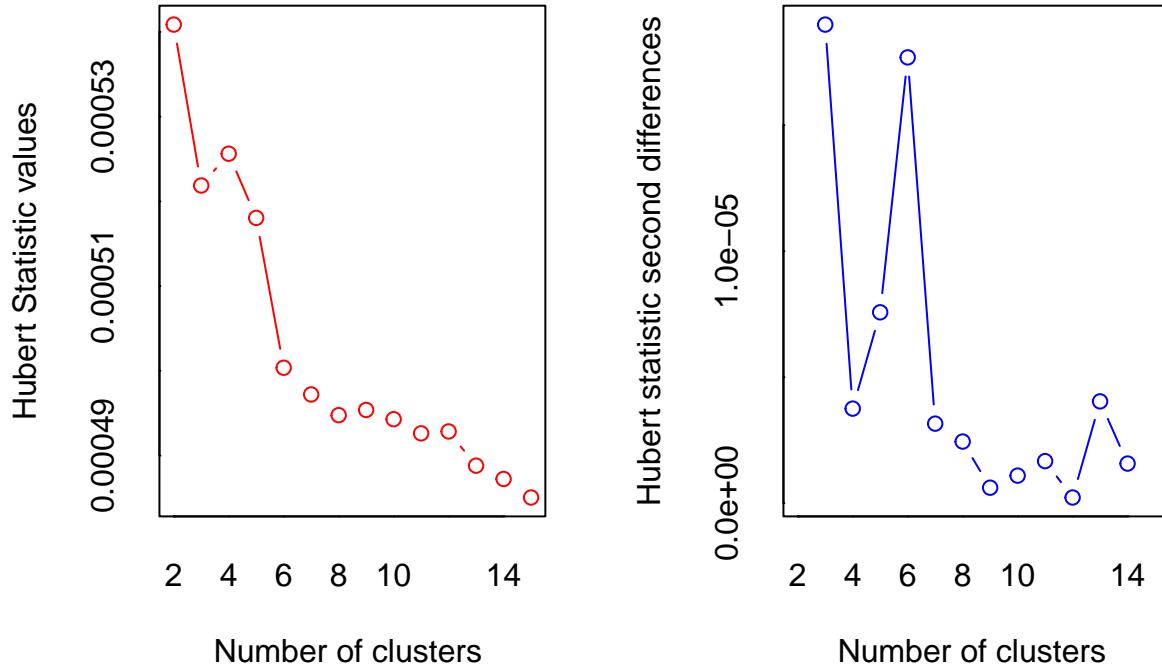
4 Análisis de cluster

Anteriormente, haciendo PCA hemos podido apreciar visualmente en el gráfico de las dos primeras componentes la presencia de un grupo denso y aislado del resto. No obstante, podríamos encontrar más grupos con una serie de características muy concretas, por lo que vamos a optar por hacer un análisis de cluster. Para ello vamos a estandarizar las variables.

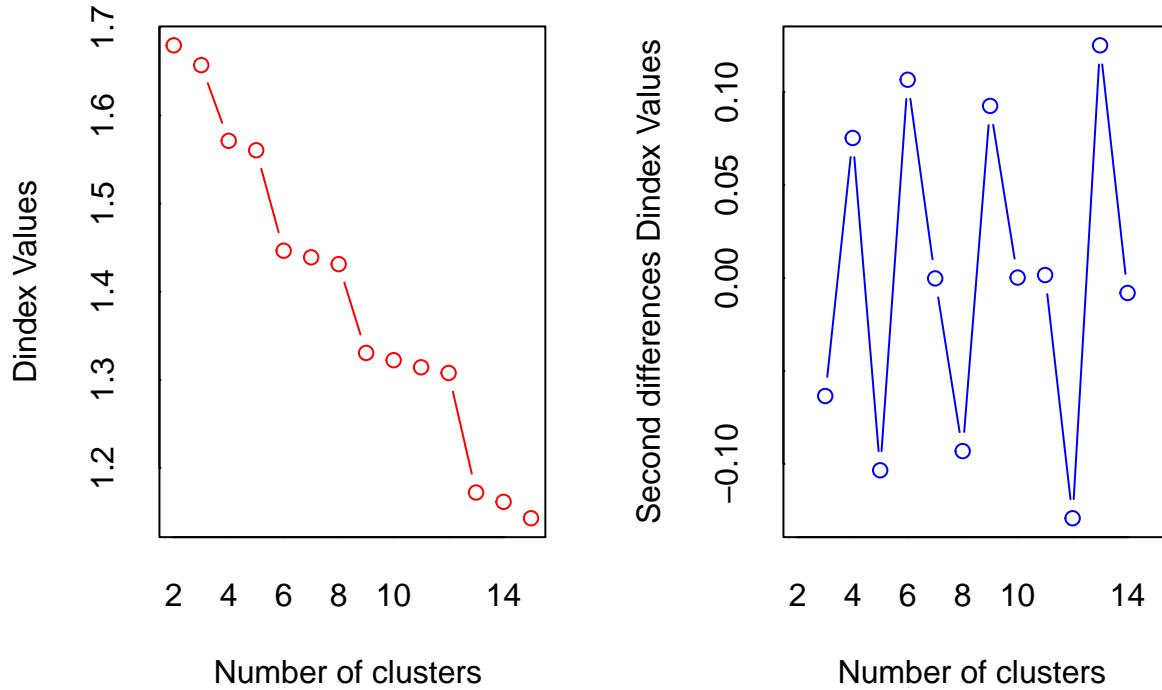
```
set.seed(4444)
ds <- as.data.frame(scale(d[,2:7]))
```

4.1 K-means

Primero lo haremos con K-means, por lo que tendremos que decidir el número de centroides. Consideraremos múltiples criterios para ello y el k más propuesto será el que elegiremos:



```
## *** : The Hubert index is a graphical method of determining the number of clusters.  
## In the plot of Hubert index, we seek a significant knee that corresponds to a  
## significant increase of the value of the measure i.e the significant peak in Hubert  
## index second differences plot.  
##
```



```

## *** : The D index is a graphical method of determining the number of clusters.
## In the plot of D index, we seek a significant knee (the significant peak in Dindex
## second differences plot) that corresponds to a significant increase of the value of
## the measure.
##
## *****
## * Among all indices:
## * 8 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 4 proposed 6 as the best number of clusters
## * 3 proposed 12 as the best number of clusters
## * 2 proposed 13 as the best number of clusters
## * 2 proposed 15 as the best number of clusters
##
## **** Conclusion ****
##
## * According to the majority rule, the best number of clusters is 2
##
## *****
##          KL      CH Hartigan     CCC      Scott      Marriot    TrCovW
## Number_clusters 13.000   2.000  12.0000  15.00   6.000 6.000000e+00   6.0
## Value_Index     40.659 297.362 194.2671 -12.34 1224.531 1.380328e+17 189198.5
## TraceW Friedman Rubin Cindex      DB Silhouette  Duda

```

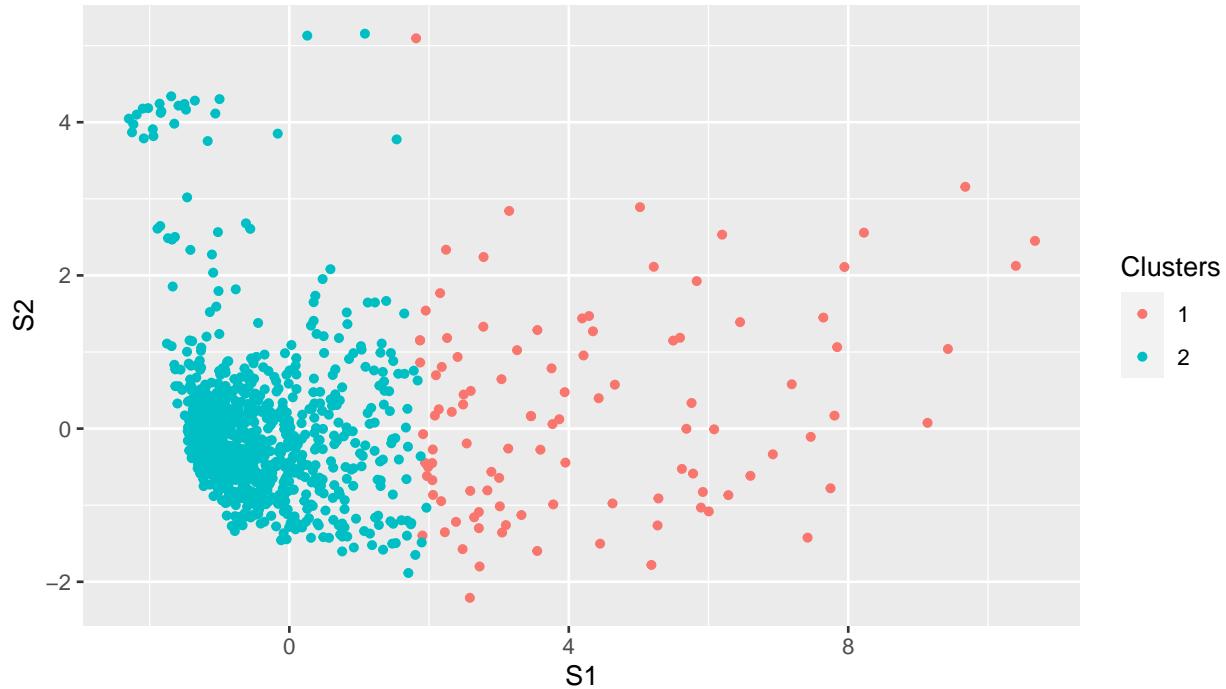
```

## Number_clusters 4.0000 6.0000 13.000 2.0000 2.0000      2.000 2.000
## Value_Index     596.0926 3.7255 -0.448 0.1879 0.8566      0.705 0.633
## PseudoT2        Beale Ratkowsky      Ball PtBiserial   Frey McClain
## Number_clusters 2.0000 3.000      2.0000 3.0000 12.0000 5.0000 2.0000
## Value_Index      13.9137 0.754      0.3065 814.7136 0.7602 9.0823 0.0148
## Dunn            Hubert SDindex Dindex      SDbw
## Number_clusters 5.0000      0 12.0000      0 15.0000
## Value_Index      0.2061      0 1.5291      0 0.7614

```

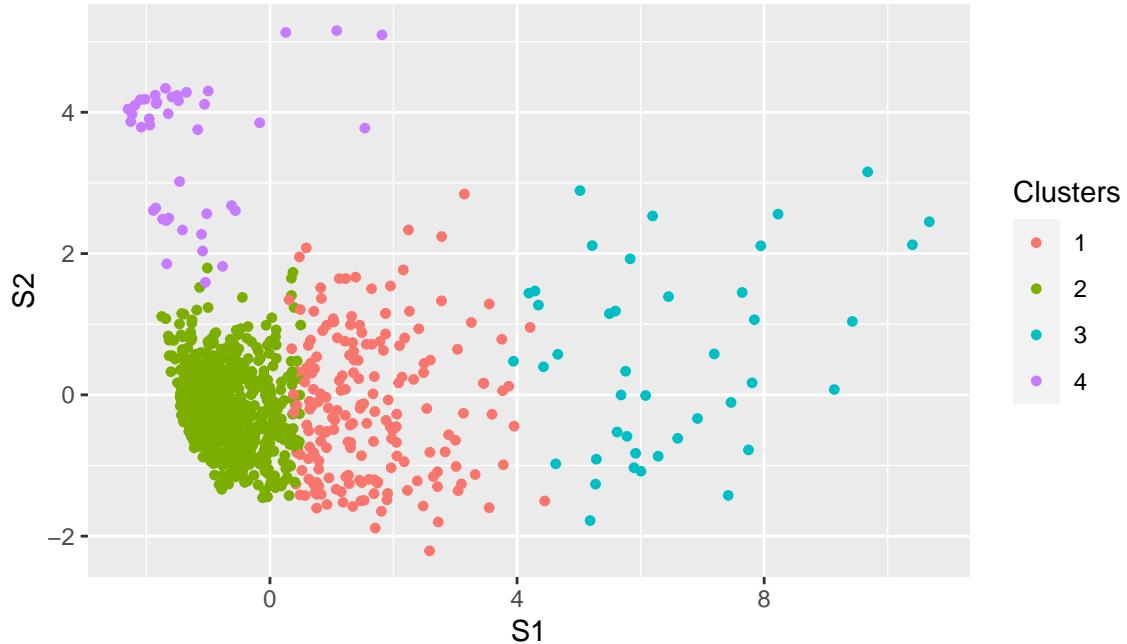
Según esta regla elegiríamos $k = 2$ aunque hay cuatro métodos que recomiendan $k = 6$. También podemos hacer esta elección por el método gráfico. Si analizamos las gráficas de índices Hubert, encontraríamos el codo en $k = 3$ por lo que podríamos elegir $k = 2$.

Probaremos primero K-means con $k = 2$

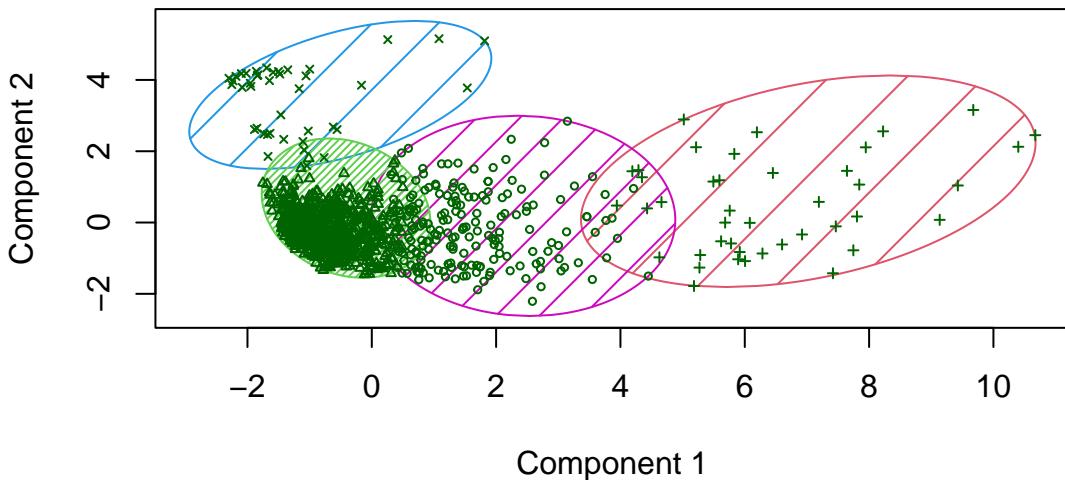


Aquí obtenemos un grupo a la izquierda y otro a la derecha del gráfico de las dos primeras componentes principales. Parece que los puntos se distinguen en cada cluster por su puntuación en la primera componente. El grupo de la izquierda son los canales con menor puntuación en la primera componente, es decir, los canales menos exitosos. Al contrario, los puntos del cluster de la derecha son los que tienen mejor puntuación en esta componente, es decir, son los canales más exitosos. En resumen, esta agrupación en dos cluster distinguiría los canales entre los exitosos y los que no.

Como parece que todavía hay mucha varianza entre los grupos, vamos a probar a aumentar el valor de k a 4.



CLUSPLOT(ds)



These two components explain 71.68 % of the point variability.

Esta distribución de los cuatro grupos parece más interesante de analizar. Para empezar, cabe señalar que el **Cluster 4** se correspondería con el grupo que habíamos identificado previamente en el PCA, por lo que los canales perteneciente a este cluster serán aquellos que no son exitosos pero que acumulan una buena cantidad de minutos en la plataforma y tienen seguidores que consumen bastante su contenido (comunidades grandes).

Por otro lado, debajo de este cluster, encontraríamos en el **Cluster 2** aquellos canales que ni son existentes ni tienen una gran trayectoria en la plataforma. Podríamos atrevernos a decir que estos son los canales más novatos en la plataforma.

En el **Cluster 1**, a la derecha del cluster mencionado anteriormente, estarían los canales que tendrían un éxito notable (todos toman valores positivos en las puntuaciones de la primera componente).

Aún así, si queremos hablar de aquellos canales que son sin lugar a dudas los más importantes y exitosos en la plataforma, tenemos que referirnos a aquellos que pertenecen al **Cluster 3**, porque son los que obtienen una puntuación significativamente alta con la primera componente.

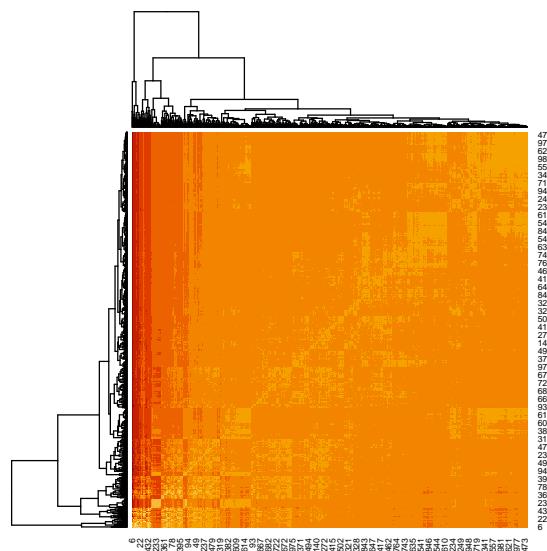
Respecto a lo anterior nos preguntamos si hay algún streamer de habla hispana en el cluster de canales exitosos.

LISTADO DE STREAMERS HISPANOHABLANTES MÁS EXITOSOS

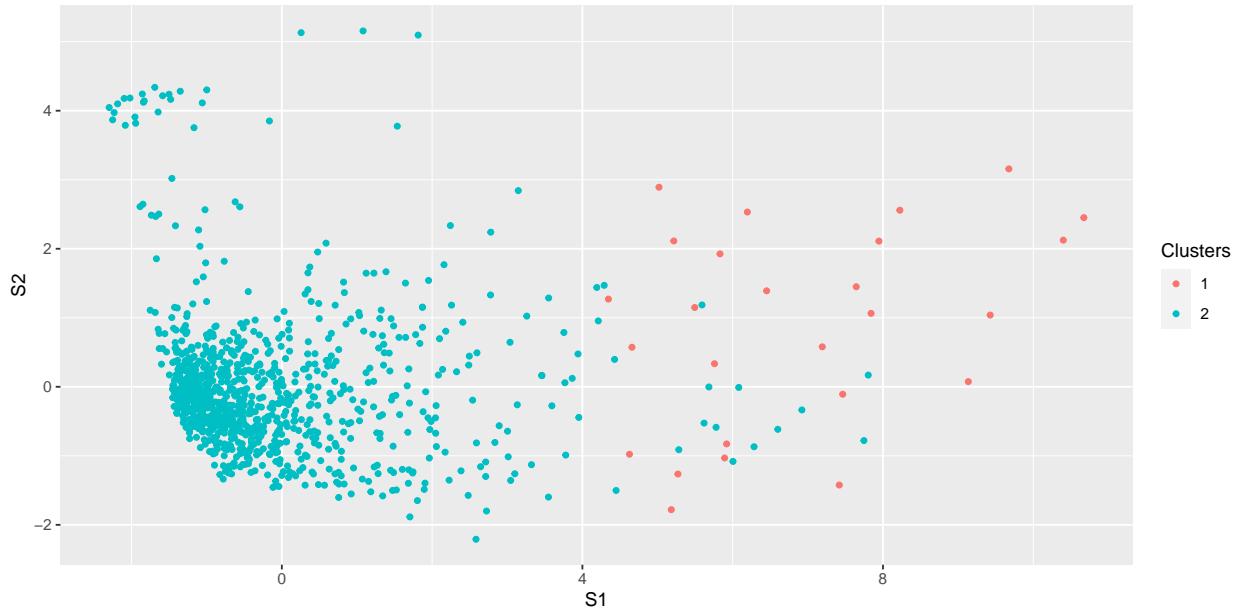
```
## [1] "ibai"      "elded"     "SLAKUN10"
```

4.2 Análisis cluster jerarquizado

En esta ocasión nos basaremos en un dendograma para analizar los grupos que se van formando localmente (por su proximidad a sus vecinos). Para ello, primero calcularemos la matriz de distancias (usaremos la distancia euclídea).

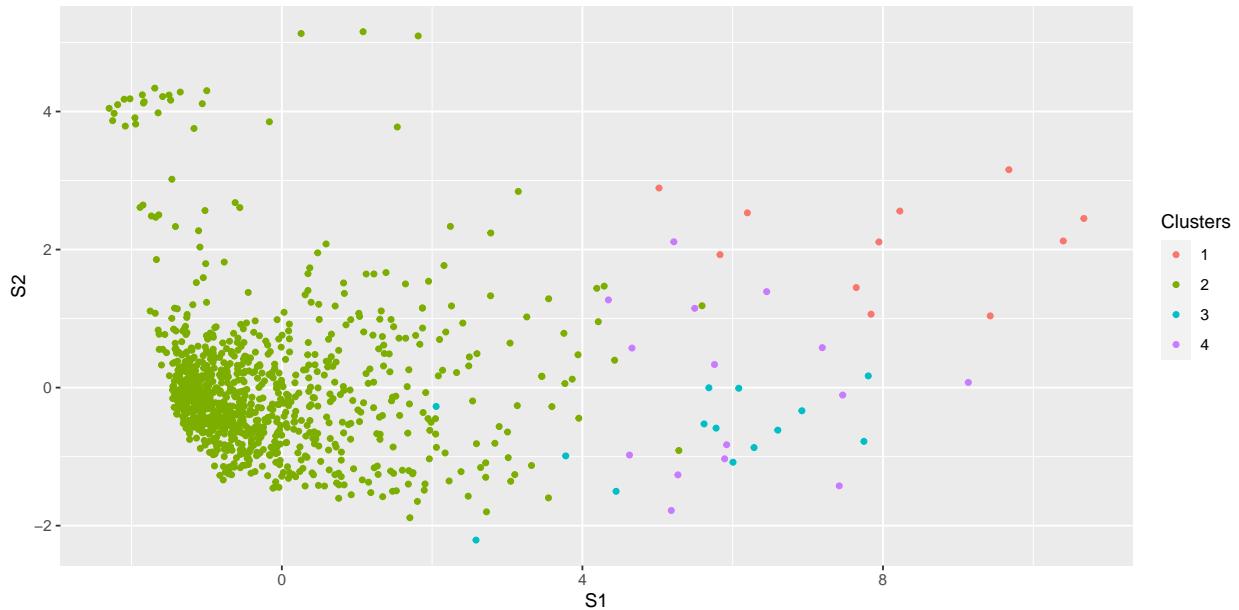


Empezaremos haciendo CA jerarquizado formando dos grupos para averiguar si se parecen a los formados por el algoritmo de K-means:



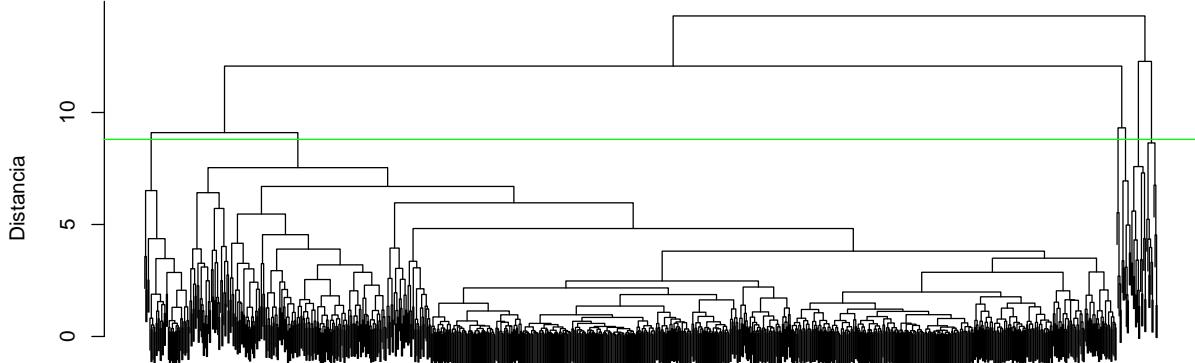
Como resultado obtenemos que 909 canales clasifican igual con los dos algoritmos. La principal diferencia de agrupación es que bastantes canales que eran clasificados en el cluster 1 en K-means (como exitosos) ahora son clasificados en el cluster 2 (como no exitosos). Sin embargo, esta distribución de los grupos no aporta mucha separabilidad en un gráfico donde se representan las dos primeras componentes, por lo que no sería muy riguroso etiquetar los clusters como exitosos y no exitosos.

A continuación, podaremos el dendrograma en $k=4$ (grupos que se forman a esa distancia euclídea) y comprobaremos la forma que tienen los grupos:



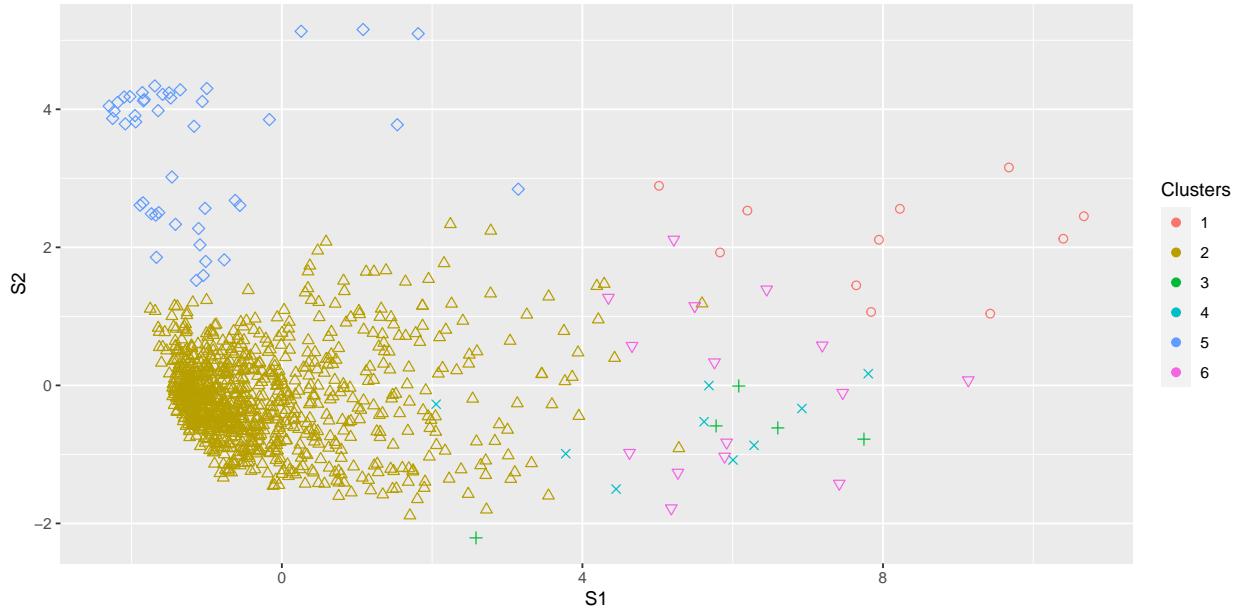
El resultado no es satisfactorio, ya que los puntos escalados que están cerca del origen están mucho más aglomerados que los de la derecha, por lo que los puntos de la izquierda formarán un gran grupo y el resto a la derecha formarán los tres restantes. Será buena idea representar el dendrograma y deducir cuál sería el mejor k para formar grupos.

Dendograma



Observaciones

He decidido, arbitrariamente, que la mejor poda del endograma sería para $k = 6$.



Como podemos observar, el **Cluster 2** forma una gran nube de puntos entorno al origen donde se encontrarán los canales de Twitch que no destacan por su éxito ni por su compromiso con su comunidad. El **Cluster 5** se correspondería con el **Cluster 4** formado con K-means (canáles no exitosos con gran compromiso con su comunidad). Los **clusters 3 y 4** son grupos muy pequeños que comparten un éxito notable respecto a la mayoría de canales, pero que no tienen un gran compromiso con su comunidad de Twitch (se encuentran abajo a la derecha del gráfico). En el caso del **Cluster 6**, sus rasgos son parecidos a los de 3 y 4, pero en este caso también constituyen el cluster canales con gran compromiso con su comunidad. Por último, el **Cluster 1** lo forman los canales más exitosos del dataset que a su vez tienen un gran compromiso con su comunidad.

LISTADO DE STREAMERS DEL CLUSTER 1

```
## [1] "Asmongold"      "NICKMERCS"      "Fextralife"      "loltyler1"
## [5] "Anomaly"        "TimTheTatman"    "LIRIK"           "MontanaBlack88"
```

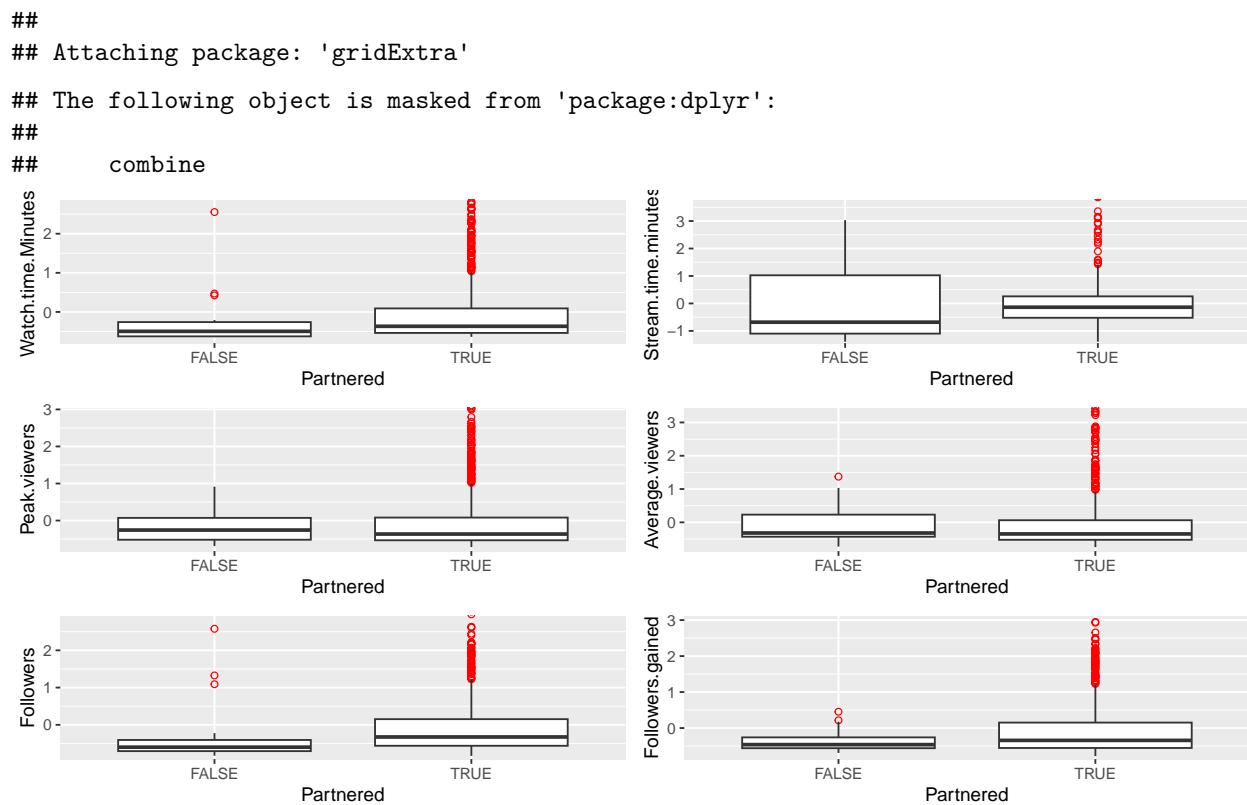
```
## [9] "sodapoppin"      "alanzoka"        "DrDisrespect"
```

5 Análisis Discriminante

Hasta ahora hemos tratado de agrupar los canales por el valor de métricas como número de seguidores, visitas ganadas este último año, número de minutos en directo, etc. En este capítulo trataremos con aquellas variables categóricas (`Partnered`, `Mature`, `Language`) que nos pueden servir a la hora de establecer una separación de los canales en grupos.

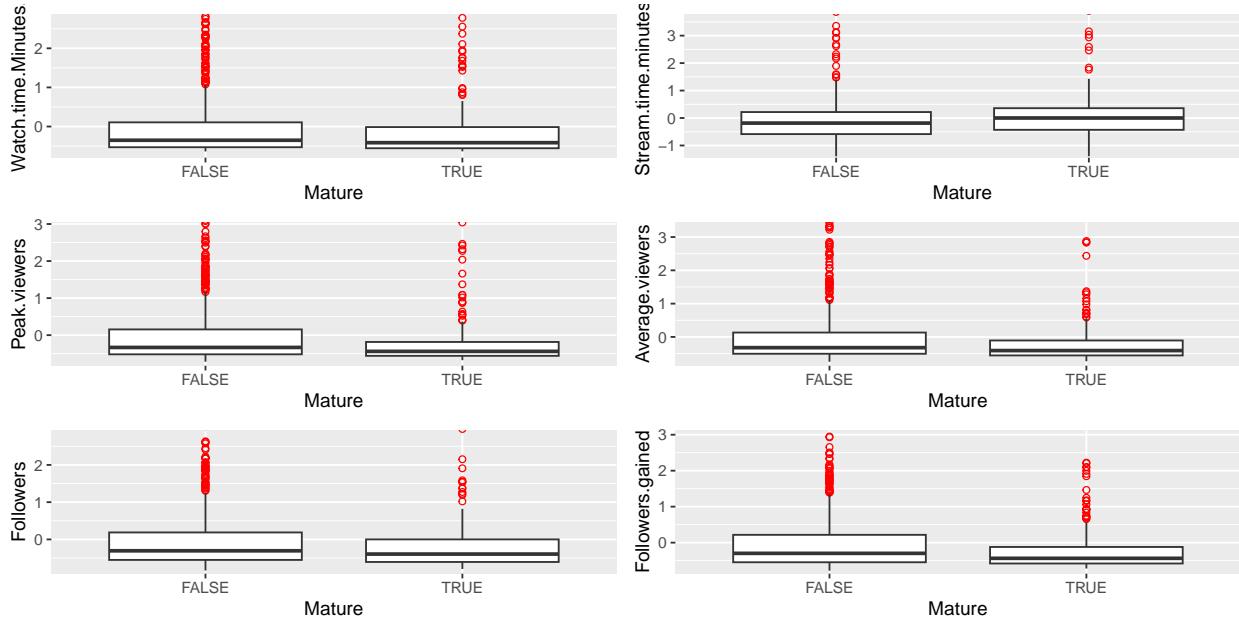
Como un rápido análisis previo vamos a usar gráficos boxplot de cada variable numérica para cada valor de la correspondiente variable categórica. Para distinguir bien el rango de valores, vamos a quitar del gráfico los valores atípicos que están muy alejados de la mediana.

Primero, vamos a buscar variables que discriminen grupos en función de la variable booleana `Partnered` (para mayor comodidad usaremos las variables escaladas).



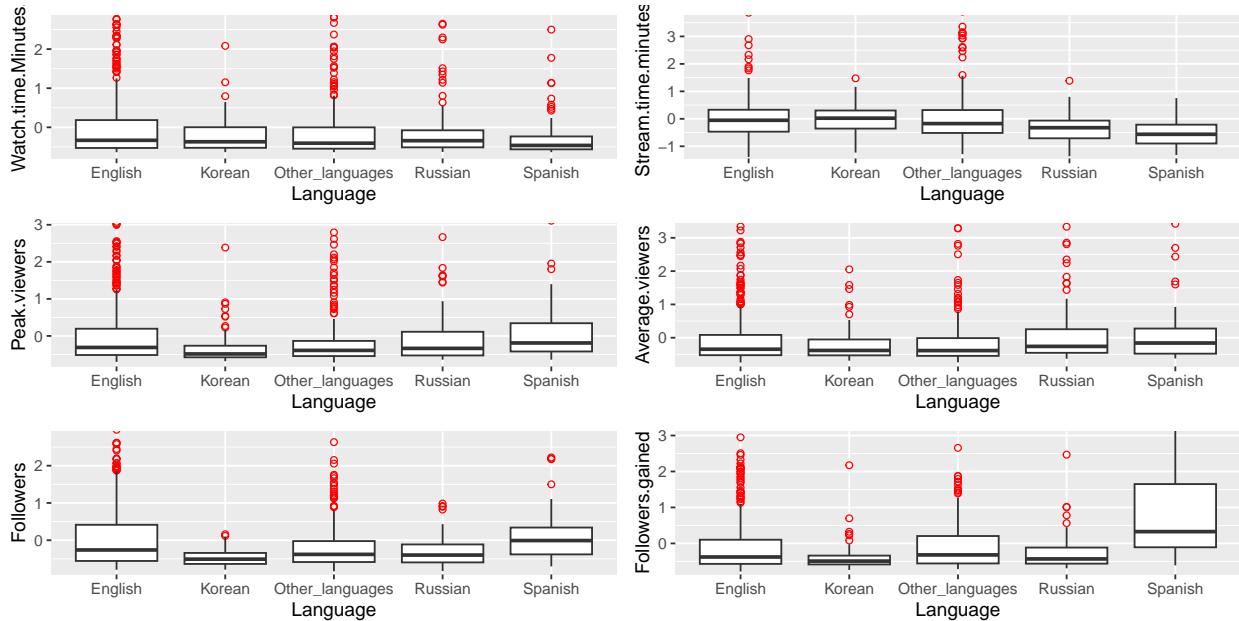
Parece que la que mejor podría discriminar es `Followers`.

Si discriminamos por la variable `Mature`.



Parece que no hay ninguna variable numérica que pueda discriminar bien en función de **Mature**, por lo no la vamos a utilizar esta etiqueta.

Como última comprobación antes de hacer análisis discriminante, vamos a intentar agrupar por el idioma del canal. Como existen 21 valores distintos que toma la variable **Language**, vamos a quedarnos con los principales idiomas y el resto lo vamos a etiquetar como **Other_languages**.



Vemos como usando estos grupos la separabilidad es mucho mayor. Podemos sacar conclusiones como que los canales coreanos son los que tienen **Peak.viewers** más bajo o que los canales cuyo idioma es el español son los que más seguidores ganaron (**Followers.gained**).

5.1 Análisis Discriminante Cuadrático (QDA)

En el análisis previo dijimos que las variables numéricas poseían valores en rangos muy distintos, por lo que llegamos a suponer que las varianzas teóricas serían distintas. Por este motivo no vamos a emplear el LDA (no vamos a obtener resultados muy satisfactorios).

En su lugar, vamos a hacer análisis discriminante cuadrático (QDA) bajo la suposición de normalidad de las variables. Lo haremos mediante validación cruzada.

Table 5: Matriz de confusión

	English	Korean	Russian	Spanish	Sum
English	193	192	47	45	477
Korean	9	58	9	1	77
Russian	13	42	10	8	73
Spanish	11	17	5	32	65
Sum	226	309	71	86	692

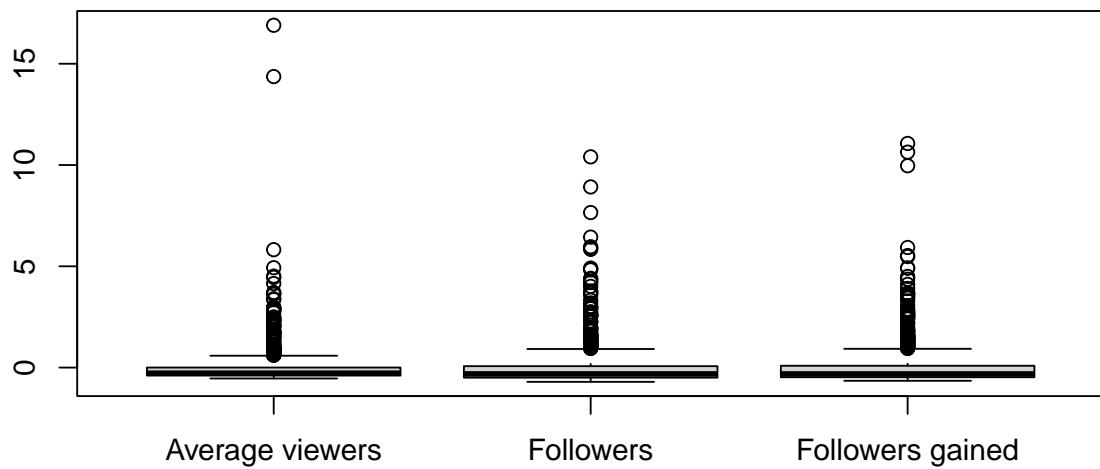
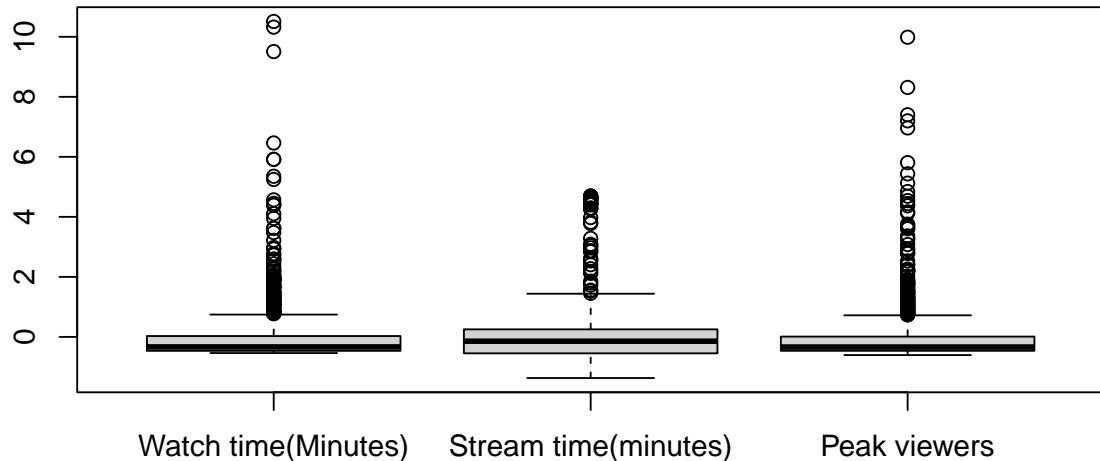
Con probabilidad de acierto igual a 0.3558526 concluimos que hemos obtenido un mal modelo. Es posible que nuestra suposición de normalidad de las variables no sea la más pertinente en este caso. Por ello, haremos un test de normalidad multivariante para averiguar si esta suposición es acertada o no.

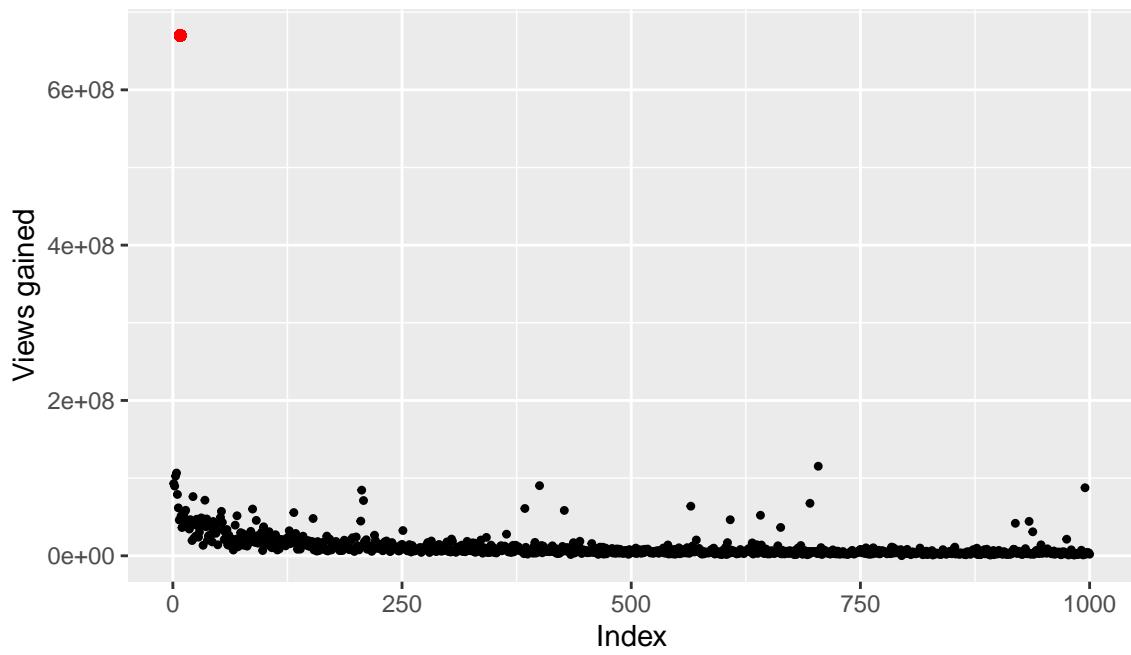
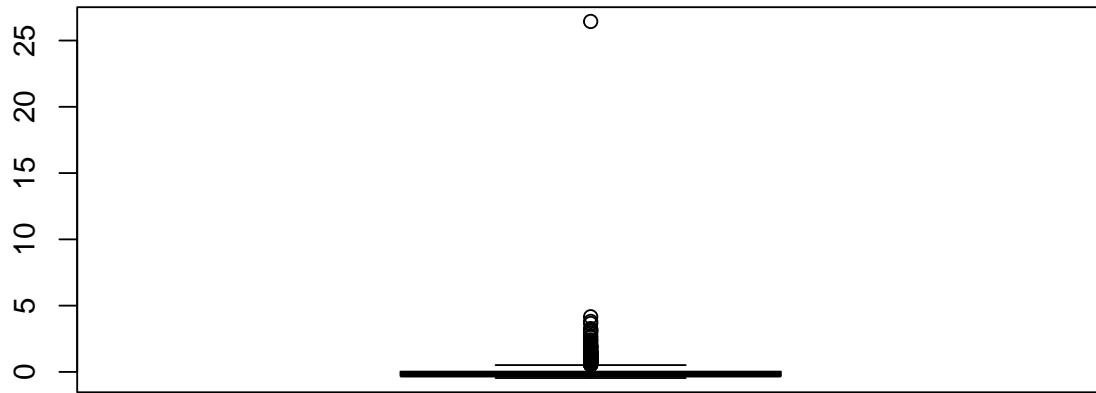
Con un p-valor igual a $1.9162009 \times 10^{-41}$ (muy bajo) y a un nivel de significación $\alpha = 0.01$ descartamos la suposición de la normalidad multivariante de las variables. En este sentido, parece que las condiciones en las que se encuentran los datos no parecen las óptimas para aplicar técnicas de análisis discriminante.

6 Regresión Lineal Múltiple

Hasta el momento, la única variable que no hemos empleado para ningún estudio estadístico ha sido `Views.gained` (visitas ganadas en 2020). El objetivo en este apartado será averiguar cuáles son las variables que influyen principalmente en el número de visitas ganadas ese año y proveer una ecuación que modelice la relación en aras de realizar predicciones a partir de nuevos datos de las variables predictoras. Por ende, `Views.gained` será el objetivo de la regresión y el resto de variables numéricas (las tratadas en las técnicas de clasificación) serán los predictores del modelo.

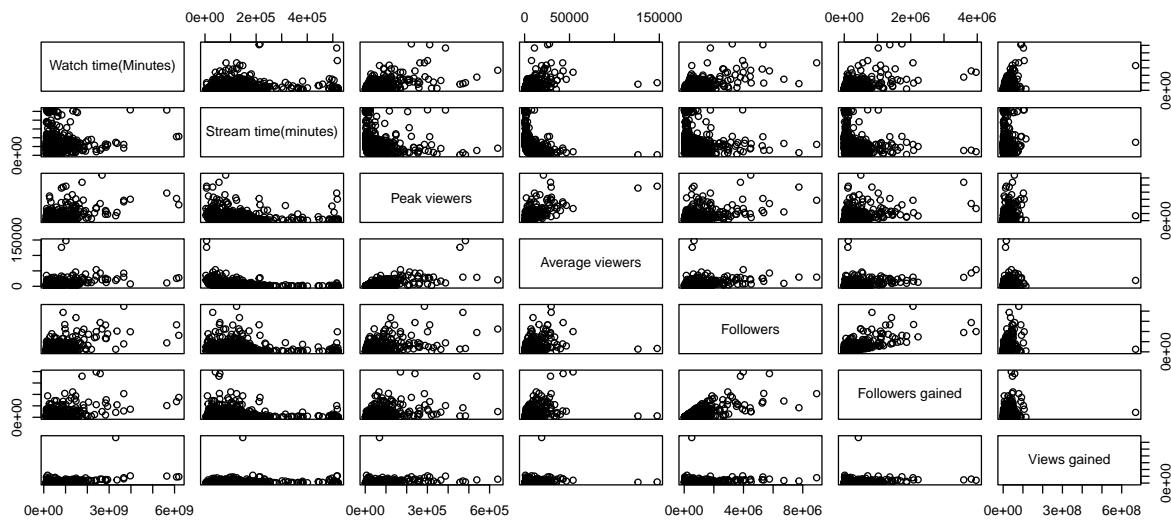
A continuación, mostramos gráficos de caja bigotes con las variables escaladas.





Como podemos observar en el anterior gráfico, hay un valor atípico, el cual se aleja significativamente de los demás, que puede causar problemas al aplicar técnicas de regresión. Se trata del canal n° 8 (Fextralife).

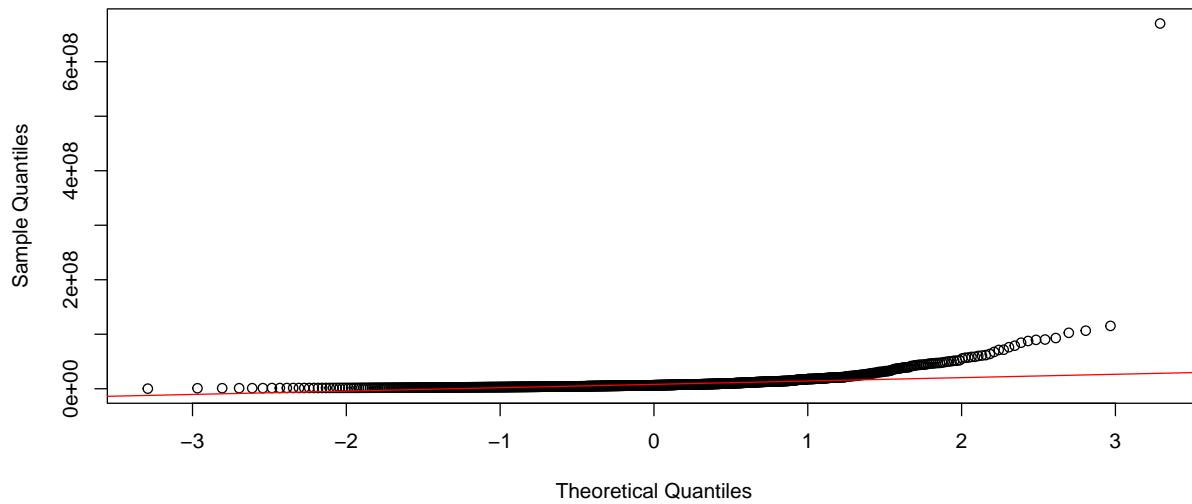
Con atípicos



Ya vimos en el análisis previo que no existe aparentemente gran correlación entre ningún par de variables. En los gráficos con atípicos se puede apreciar claramente la observación atípica de la fila 8.

```
##  
## Shapiro-Wilk normality test  
##  
## data: df$`Views gained`  
## W = 0.26274, p-value < 2.2e-16
```

Normal Q-Q Plot



Vemos que la hipótesis de normalidad queda totalmente rechazada para estos datos, pues obtenemos un p-valor significativamente bajo y el gráfico muestra como algunos cuantiles muestrales se alejan completamente.

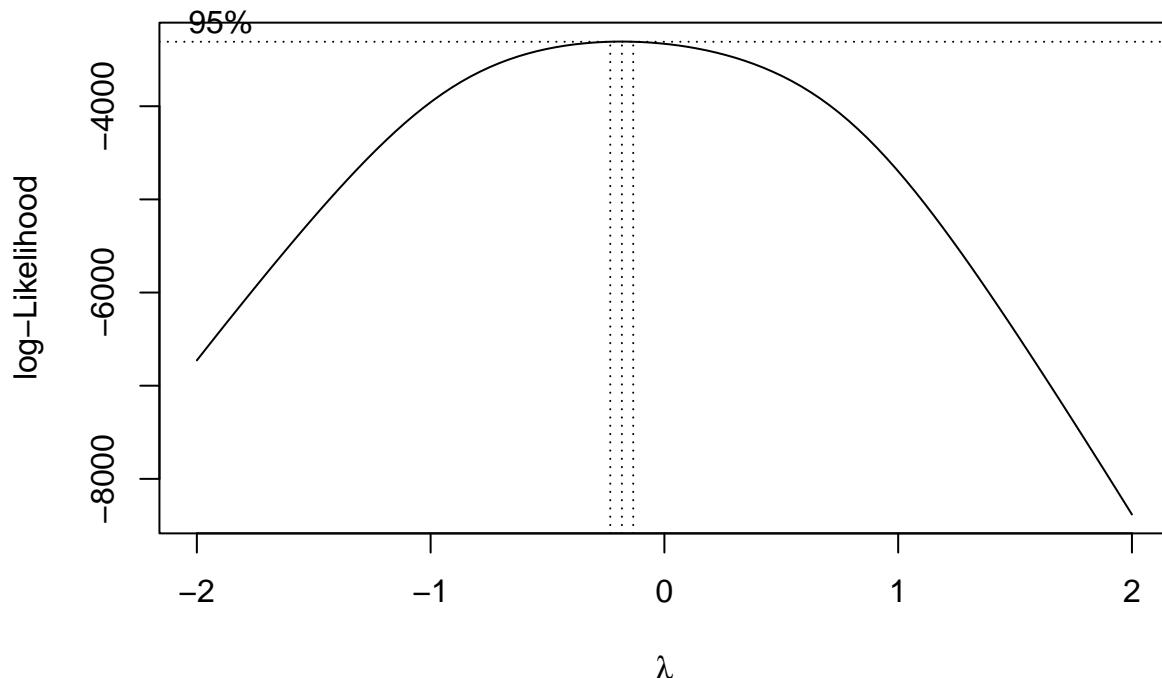
Es posible (dada la exponencialidad de los datos) que la variable siga una distribución log-normal, por lo que tendremos que tomar el logaritmo de los datos.

6.1 Transformaciones de los datos

Puesto que no hay indicios de normalidad en los datos vamos a probar a transformarlos. Para ello, vamos a comprobar si esa mejor transformación podría ser tomar logaritmos.

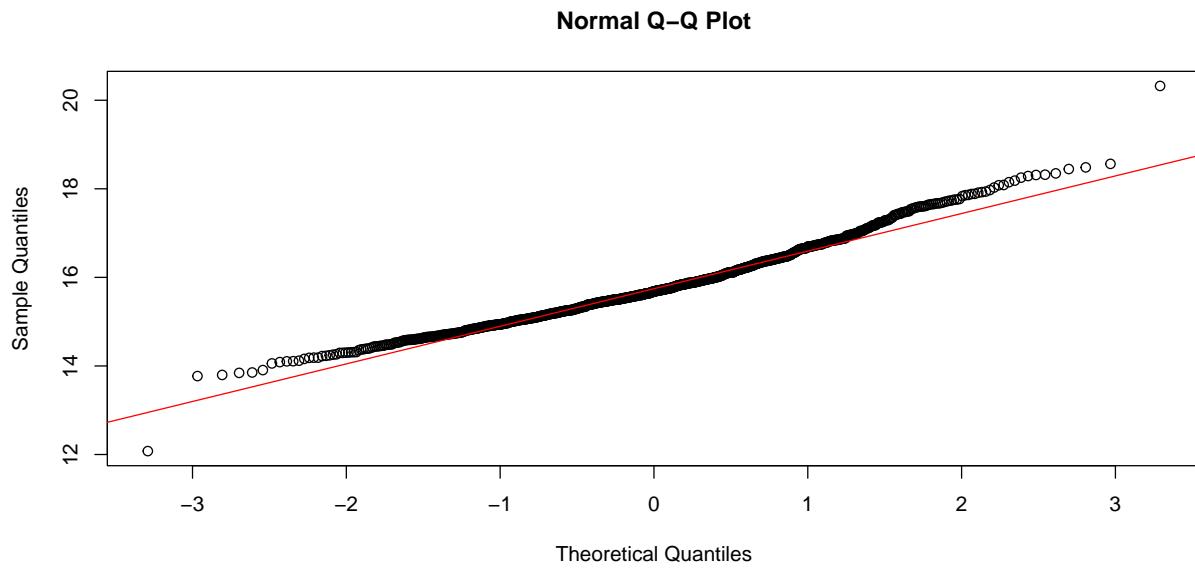
```
library("MASS")
datos <- data.frame(df[,c(2:8)])

boxcox(lm(Views.gained ~ 1, data = datos), lambda = seq(-2, 2, 1/10))
```



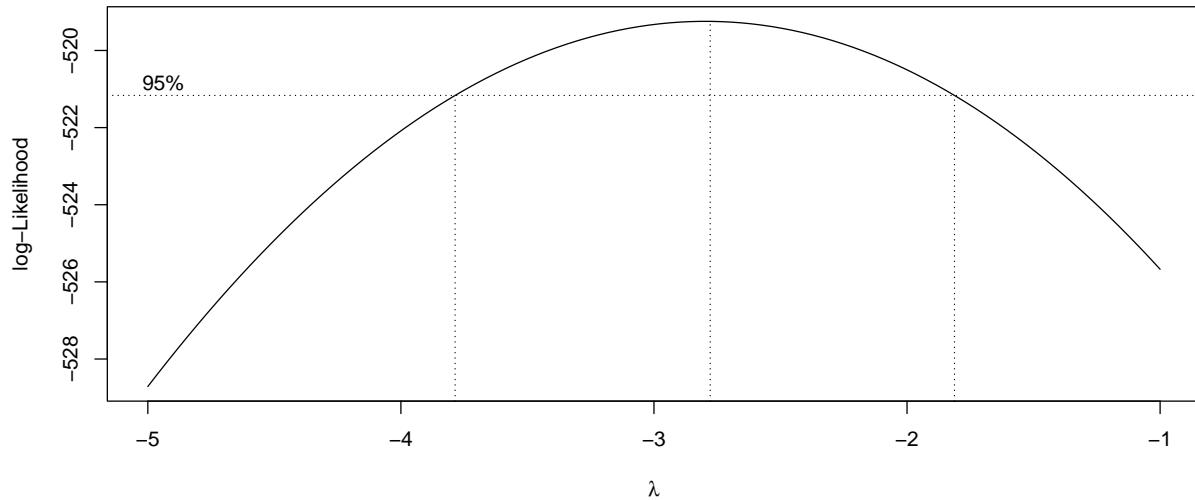
La máxima verosimilitud se alcanza cerca de $\lambda = 0$, tomaremos logaritmos.

```
## 
## Shapiro-Wilk normality test
##
## data: log(datos$Views.gained)
## W = 0.97604, p-value = 8.822e-12
```

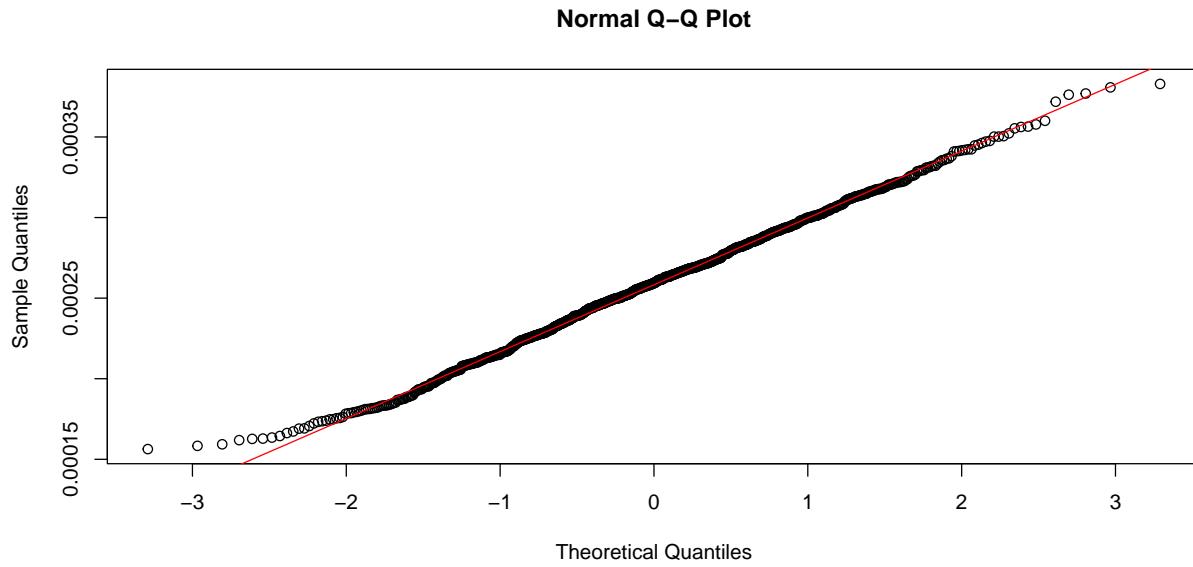


Vemos que hay dos valores de los cuantiles muestrales (uno en cada extremo) que se alejan demasiado. Decidimos eliminarlos y buscar el mejor λ para transformar los datos.

```
datos<-datos[-which.min(datos$Views.gained),]
datos<-datos[-which.max(datos$Views.gained),]
```



Como podemos observar, la máxima verosimilitud se alcanza entorno a $\lambda = -3$. Elevamos a ese valor y comprobamos si podemos aceptar la hipótesis de log-normalidad.



Tras realizar el test de Shapiro-Wilk el p-valor da 0.0844781. A un nivel de significación $\alpha = 0.05$ aceptamos la hipótesis de log-normalidad de la variable `Views.gained`

6.2 Estimación de los parámetros del modelo RLM y métodos de selección de regresores

Empezamos realizando una estimación de los parámetros del **modelo RLM usando todos los predictores**.

```
##
## Call:
## lm(formula = Y_new ~ . - Views.gained, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.157e-04 -2.230e-05 -3.251e-06  2.001e-05  1.398e-04
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)            2.778e-04  2.143e-06 129.600 < 2e-16 ***
## Watch.time.Minutes. -3.360e-14  2.883e-15 -11.655 < 2e-16 ***
## Stream.time.minutes. 8.696e-12  1.359e-11   0.640  0.52234  
## Peak.viewers         -5.607e-11  2.629e-11  -2.133  0.03318 *  
## Average.viewers      -2.462e-10  1.767e-10  -1.393  0.16389  
## Followers           -5.655e-12  2.077e-12  -2.723  0.00658 ** 
## Followers.gained     1.404e-12  4.495e-12   0.312  0.75482  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.276e-05 on 991 degrees of freedom
## Multiple R-squared:  0.36, Adjusted R-squared:  0.3562 
## F-statistic: 92.92 on 6 and 991 DF,  p-value: < 2.2e-16
```

El R-cuadrado ajustado tan bajo nos indica que no se ha conseguido un buen ajuste. En el contraste de si los coeficientes de los regresores son significativos encontramos un p-valor alto, lo que indica que el modelo es reducible. Para reducir el modelo aplicaremos el **método de selección de regresores backward**.

```

## Start: AIC=-20604.41
## Y_new ~ (Watch.time.Minutes. + Stream.time.minutes. + Peak.viewers +
##           Average.viewers + Followers + Followers.gained + Views.gained) -
##           Views.gained
##
##                                     Df  Sum of Sq      RSS      AIC
## - Followers.gained          1 1.0500e-10 1.0636e-06 -20606
## - Stream.time.minutes.    1 4.4000e-10 1.0639e-06 -20606
## - Average.viewers          1 2.0830e-09 1.0655e-06 -20605
## <none>                      1 1.0635e-06 -20604
## - Peak.viewers             1 4.8820e-09 1.0683e-06 -20602
## - Followers                1 7.9580e-09 1.0714e-06 -20599
## - Watch.time.Minutes.     1 1.4578e-07 1.2092e-06 -20478
##
## Step: AIC=-20606.31
## Y_new ~ Watch.time.Minutes. + Stream.time.minutes. + Peak.viewers +
##           Average.viewers + Followers
##
##                                     Df  Sum of Sq      RSS      AIC
## - Stream.time.minutes.    1 3.9100e-10 1.0640e-06 -20608
## - Average.viewers          1 2.0300e-09 1.0656e-06 -20606
## <none>                      1 1.0636e-06 -20606
## - Peak.viewers             1 4.8410e-09 1.0684e-06 -20604
## - Followers                1 9.8880e-09 1.0735e-06 -20599
## - Watch.time.Minutes.     1 1.4667e-07 1.2102e-06 -20479
##
## Step: AIC=-20607.95
## Y_new ~ Watch.time.Minutes. + Peak.viewers + Average.viewers +
##           Followers
##
##                                     Df  Sum of Sq      RSS      AIC
## <none>                      1 1.0640e-06 -20608
## - Average.viewers          1 2.7540e-09 1.0667e-06 -20607
## - Peak.viewers              1 4.9370e-09 1.0689e-06 -20605
## - Followers                 1 1.0962e-08 1.0749e-06 -20600
## - Watch.time.Minutes.      1 1.6212e-07 1.2261e-06 -20468

```

Observamos que primero se elimina la variable `Followers.gained` (la que mayor p-valor tenía) y después quita `Stream.time.minutes.`, quedándose con 4 variables (`Average.viewers`, `Peak.viewers`, `Followers` y `Watch.time.Minutes.`). Los coeficientes serían:

```

modelo_backward$coefficients

##            (Intercept) Watch.time.Minutes.      Peak.viewers      Average.viewers
## 2.788700e-04       -3.288572e-14      -5.632781e-11      -2.717092e-10
##           Followers
## -5.494044e-12

```

Una vez probado el modelo backward, probaremos con el modelo forward partiendo solo con la constante para ir añadiendo nuevas variables.

```

modelo_cte <- lm(Y_new ~1, data = datos)

modelo_forward <- step(modelo_cte, direction = "forward",
                       scope = formula(modelo_completo))

```

```

## Start: AIC=-20170.97
## Y_new ~ 1
##
## Df Sum of Sq RSS AIC
## + Watch.time.Minutes. 1 5.6059e-07 1.1011e-06 -20580
## + Followers 1 3.3918e-07 1.3226e-06 -20397
## + Peak.viewers 1 3.1646e-07 1.3453e-06 -20380
## + Average.viewers 1 2.2369e-07 1.4380e-06 -20313
## + Followers.gained 1 2.1880e-07 1.4429e-06 -20310
## <none> 1.6617e-06 -20171
## + Stream.time.minutes. 1 5.5000e-10 1.6612e-06 -20169
##
## Step: AIC=-20579.66
## Y_new ~ Watch.time.Minutes.
##
## Df Sum of Sq RSS AIC
## + Peak.viewers 1 2.2815e-08 1.0783e-06 -20599
## + Followers 1 2.0486e-08 1.0807e-06 -20596
## + Average.viewers 1 1.7880e-08 1.0833e-06 -20594
## + Followers.gained 1 8.6597e-09 1.0925e-06 -20586
## + Stream.time.minutes. 1 8.2104e-09 1.0929e-06 -20585
## <none> 1.1011e-06 -20580
##
## Step: AIC=-20598.56
## Y_new ~ Watch.time.Minutes. + Peak.viewers
##
## Df Sum of Sq RSS AIC
## + Followers 1 1.1613e-08 1.0667e-06 -20607
## + Followers.gained 1 3.4430e-09 1.0749e-06 -20600
## + Average.viewers 1 3.4058e-09 1.0749e-06 -20600
## + Stream.time.minutes. 1 2.8134e-09 1.0755e-06 -20599
## <none> 1.0783e-06 -20599
##
## Step: AIC=-20607.37
## Y_new ~ Watch.time.Minutes. + Peak.viewers + Followers
##
## Df Sum of Sq RSS AIC
## + Average.viewers 1 2.7542e-09 1.0640e-06 -20608
## <none> 1.0667e-06 -20607
## + Stream.time.minutes. 1 1.1156e-09 1.0656e-06 -20606
## + Followers.gained 1 3.5600e-12 1.0667e-06 -20605
##
## Step: AIC=-20607.95
## Y_new ~ Watch.time.Minutes. + Peak.viewers + Followers + Average.viewers
##
## Df Sum of Sq RSS AIC
## <none> 1.0640e-06 -20608
## + Stream.time.minutes. 1 3.9123e-10 1.0636e-06 -20606
## + Followers.gained 1 5.6430e-11 1.0639e-06 -20606
modelo_forward$coefficients

## (Intercept) Watch.time.Minutes. Peak.viewers Followers
## 2.788700e-04 -3.288572e-14 -5.632781e-11 -5.494044e-12
## Average.viewers
```

```

## -2.717092e-10

Observamos que se obtiene el mismo modelo con cuatro variables que se obtuvo con el método backward.
Por último, probaremos con el método stepwise, que va añadiendo y quitando variables al modelo.

modelo_stepwise <- step(modelo_cte, direction = "both",
                           scope = formula(modelo_completo))

## Start: AIC=-20170.97
## Y_new ~ 1
##
##          Df  Sum of Sq      RSS      AIC
## + Watch.time.Minutes. 1 5.6059e-07 1.1011e-06 -20580
## + Followers            1 3.3918e-07 1.3226e-06 -20397
## + Peak.viewers         1 3.1646e-07 1.3453e-06 -20380
## + Average.viewers     1 2.2369e-07 1.4380e-06 -20313
## + Followers.gained    1 2.1880e-07 1.4429e-06 -20310
## <none>                1 6.617e-06 -20171
## + Stream.time.minutes. 1 5.5000e-10 1.6612e-06 -20169
##
## Step: AIC=-20579.66
## Y_new ~ Watch.time.Minutes.
##
##          Df  Sum of Sq      RSS      AIC
## + Peak.viewers         1 2.2810e-08 1.0783e-06 -20599
## + Followers            1 2.0490e-08 1.0807e-06 -20596
## + Average.viewers     1 1.7880e-08 1.0833e-06 -20594
## + Followers.gained    1 8.6600e-09 1.0925e-06 -20586
## + Stream.time.minutes. 1 8.2100e-09 1.0929e-06 -20585
## <none>                1 1.1011e-06 -20580
## - Watch.time.Minutes. 1 5.6059e-07 1.6617e-06 -20171
##
## Step: AIC=-20598.56
## Y_new ~ Watch.time.Minutes. + Peak.viewers
##
##          Df  Sum of Sq      RSS      AIC
## + Followers            1 1.1613e-08 1.0667e-06 -20607
## + Followers.gained    1 3.4430e-09 1.0749e-06 -20600
## + Average.viewers     1 3.4060e-09 1.0749e-06 -20600
## + Stream.time.minutes. 1 2.8130e-09 1.0755e-06 -20599
## <none>                1 1.0783e-06 -20599
## - Peak.viewers         1 2.2815e-08 1.1011e-06 -20580
## - Watch.time.Minutes. 1 2.6695e-07 1.3453e-06 -20380
##
## Step: AIC=-20607.37
## Y_new ~ Watch.time.Minutes. + Peak.viewers + Followers
##
##          Df  Sum of Sq      RSS      AIC
## + Average.viewers     1 2.7540e-09 1.0640e-06 -20608
## <none>                1 1.0667e-06 -20607
## + Stream.time.minutes. 1 1.1160e-09 1.0656e-06 -20606
## + Followers.gained    1 4.0000e-12 1.0667e-06 -20605
## - Followers            1 1.1613e-08 1.0783e-06 -20599
## - Peak.viewers         1 1.3942e-08 1.0807e-06 -20596
## - Watch.time.Minutes. 1 1.6691e-07 1.2336e-06 -20464

```

```

## 
## Step: AIC=-20607.95
## Y_new ~ Watch.time.Minutes. + Peak.viewers + Followers + Average.viewers
##
##                               Df  Sum of Sq      RSS      AIC
## <none>                         1.0640e-06 -20608
## - Average.viewers             1 2.7540e-09 1.0667e-06 -20607
## + Stream.time.minutes.       1 3.9100e-10 1.0636e-06 -20606
## + Followers.gained           1 5.6000e-11 1.0639e-06 -20606
## - Peak.viewers               1 4.9370e-09 1.0689e-06 -20605
## - Followers                  1 1.0962e-08 1.0749e-06 -20600
## - Watch.time.Minutes.        1 1.6212e-07 1.2261e-06 -20468
modelo_stepwise$coefficients

```

```

##          (Intercept) Watch.time.Minutes.      Peak.viewers      Followers
## 2.788700e-04     -3.288572e-14    -5.632781e-11   -5.494044e-12
## Average.viewers
## -2.717092e-10

```

En esta ocasión, a través de los tres métodos se ha llegado al mismo modelo.

```

modelo_final <- modelo_backward
summary(modelo_final)

```

```

## 
## Call:
## lm(formula = Y_new ~ Watch.time.Minutes. + Peak.viewers + Average.viewers +
##     Followers, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.153e-04 -2.218e-05 -3.035e-06  1.986e-05  1.380e-04
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2.789e-04  1.353e-06 206.072 < 2e-16 ***
## Watch.time.Minutes. -3.289e-14  2.673e-15 -12.301 < 2e-16 ***
## Peak.viewers   -5.633e-11  2.624e-11 -2.147  0.03206 *  
## Average.viewers -2.717e-10  1.695e-10 -1.603  0.10919    
## Followers     -5.494e-12  1.718e-12 -3.199  0.00143 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.273e-05 on 993 degrees of freedom
## Multiple R-squared:  0.3597, Adjusted R-squared:  0.3571 
## F-statistic: 139.5 on 4 and 993 DF,  p-value: < 2.2e-16

```

Observamos que el R-cuadrado sigue siendo muy bajo (0.3571), es decir solo explica el 35.7 % de la información. Al igual que pasaba con `Views.gained`, es posible que sea necesario tomar los logaritmos de los regresores dada la exponencialidad de los datos.

```

datos[,c(1:6)] <- log(datos[,c(1:6)])

```

```

## Warning in FUN(X[[i]], ...): Se han producido NaNs

```

```

modelo_log_completo <- lm(Y_new ~ . - Views.gained ,
                           data = datos)

summary(modelo_log_completo)

##
## Call:
## lm(formula = Y_new ~ . - Views.gained, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.280e-04 -1.596e-05  6.500e-08  1.623e-05  8.980e-05 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 9.966e-04  2.534e-05 39.321 < 2e-16 ***
## Watch.time.Minutes. -2.447e-05  8.196e-06 -2.985  0.0029 ** 
## Stream.time.minutes. -9.231e-06  7.940e-06 -1.162  0.2453  
## Peak.viewers      -5.923e-06  1.381e-06 -4.289 1.97e-05 *** 
## Average.viewers    -1.100e-05  7.831e-06 -1.405  0.1604  
## Followers        -1.298e-06  1.335e-06 -0.972  0.3314  
## Followers.gained   6.372e-07  1.016e-06  0.627  0.5307  
## ---                
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 2.704e-05 on 988 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.5647, Adjusted R-squared:  0.5621 
## F-statistic: 213.7 on 6 and 988 DF,  p-value: < 2.2e-16

```

Parece que esta transformación de los datos ha mejorado significativamente el modelo.

```
modelo_log_final <- step(modelo_completo, direction = "backward")
```

```

## Start:  AIC=-20604.41
## Y_new ~ (Watch.time.Minutes. + Stream.time.minutes. + Peak.viewers +
##           Average.viewers + Followers + Followers.gained + Views.gained) -
##           Views.gained
##
##              Df  Sum of Sq      RSS      AIC
## - Followers.gained     1 1.0500e-10 1.0636e-06 -20606
## - Stream.time.minutes. 1 4.4000e-10 1.0639e-06 -20606
## - Average.viewers      1 2.0830e-09 1.0655e-06 -20605
## <none>                      1.0635e-06 -20604
## - Peak.viewers          1 4.8820e-09 1.0683e-06 -20602
## - Followers             1 7.9580e-09 1.0714e-06 -20599
## - Watch.time.Minutes.   1 1.4578e-07 1.2092e-06 -20478
##
## Step:  AIC=-20990.68
## Y_new ~ Watch.time.Minutes. + Stream.time.minutes. + Peak.viewers +
##           Average.viewers + Followers
##
##              Df  Sum of Sq      RSS      AIC
## - Followers            1 4.8030e-10 7.2408e-07 -20992
## - Stream.time.minutes. 1 1.0085e-09 7.2461e-07 -20991

```

```

## - Average.viewers      1 1.4286e-09 7.2503e-07 -20991
## <none>                  7.2360e-07 -20991
## - Watch.time.Minutes.  1 6.5131e-09 7.3012e-07 -20984
## - Peak.viewers         1 1.3086e-08 7.3669e-07 -20975
##
## Step: AIC=-20992.02
## Y_new ~ Watch.time.Minutes. + Stream.time.minutes. + Peak.viewers +
##       Average.viewers
##
##                               Df  Sum of Sq      RSS      AIC
## - Stream.time.minutes.  1 1.0439e-09 7.2513e-07 -20993
## <none>                      7.2408e-07 -20992
## - Average.viewers        1 1.5671e-09 7.2565e-07 -20992
## - Watch.time.Minutes.   1 6.5892e-09 7.3067e-07 -20985
## - Peak.viewers          1 1.4407e-08 7.3849e-07 -20974
##
## Step: AIC=-20992.58
## Y_new ~ Watch.time.Minutes. + Peak.viewers + Average.viewers
##
##                               Df  Sum of Sq      RSS      AIC
## <none>                      7.2513e-07 -20993
## - Average.viewers         1 1.5400e-09 7.2666e-07 -20993
## - Peak.viewers           1 1.3410e-08 7.3854e-07 -20976
## - Watch.time.Minutes.    1 3.2598e-07 1.0511e-06 -20624

```

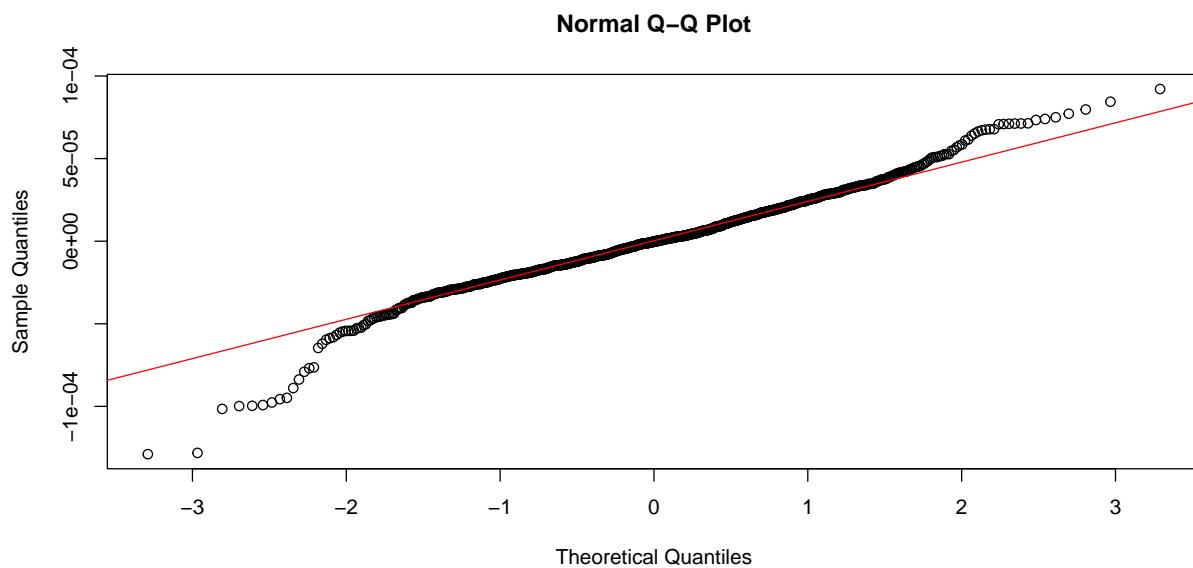
6.3 Validación del modelo

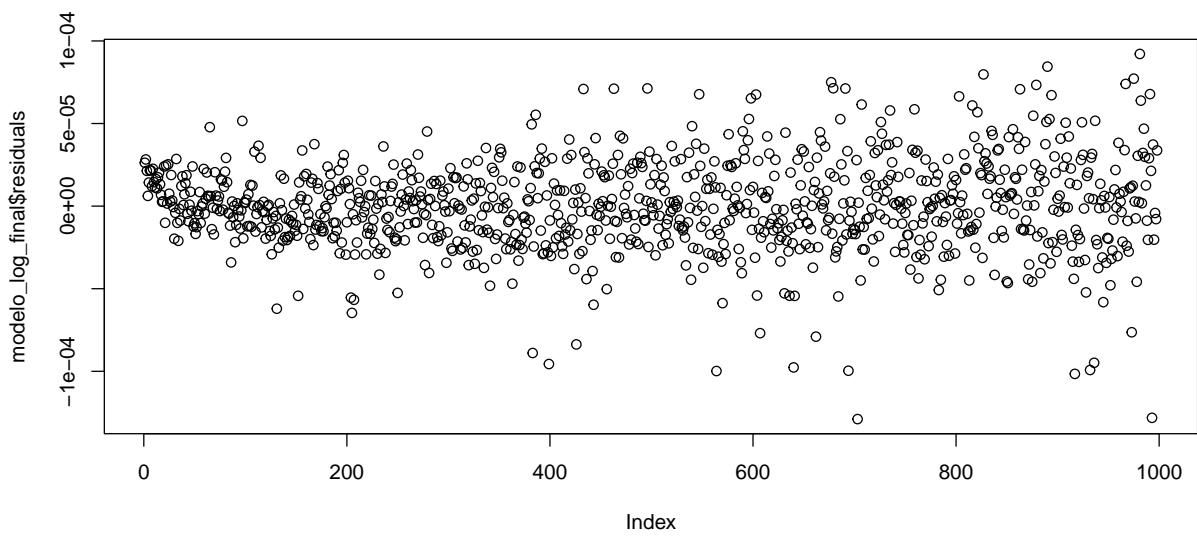
Para poder hacer inferencia y predicciones sobre los datos tendremos que validar nuestro modelo.

```

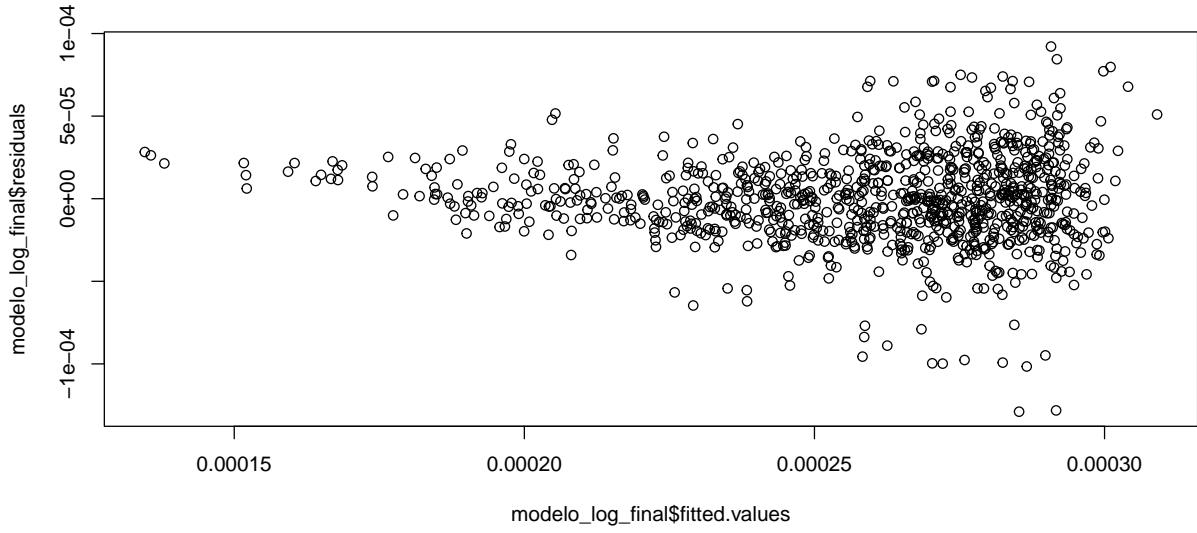
##
## Shapiro-Wilk normality test
##
## data: residuos
## W = 0.97317, p-value = 1.281e-12

```

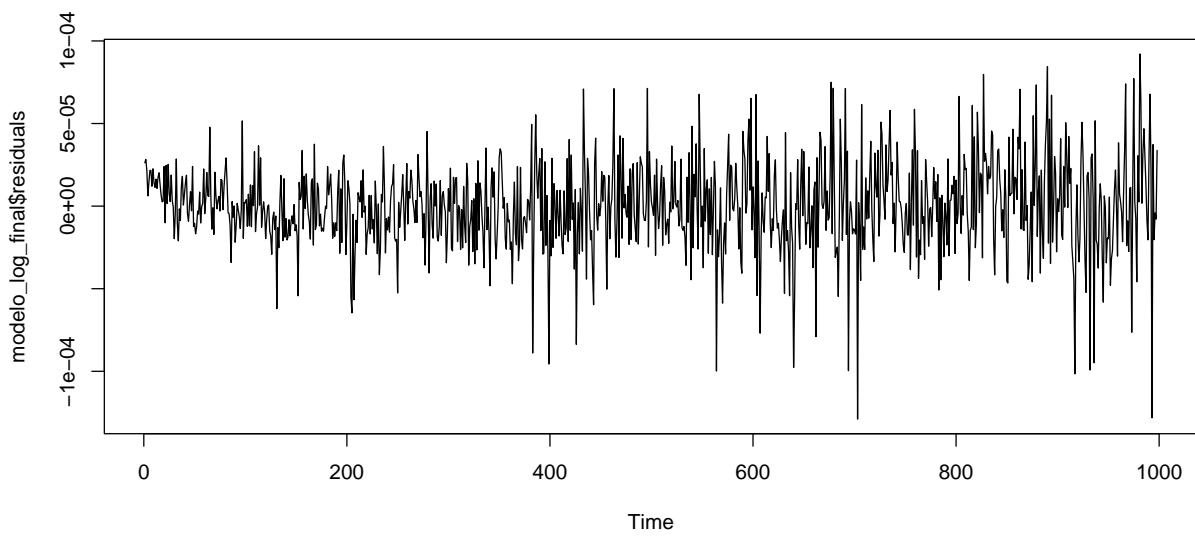




No se cumple la normalidad de los residuos.



La varianza de los residuos aumenta según aumenta el valor que predice el modelo. No se cumple la propiedad de homocedasticidad.



```

## Warning: package 'lmtest' was built under R version 4.3.3
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##       as.Date, as.Date.numeric
##
## Durbin-Watson test
##
## data: modelo_log_final
## DW = 1.9101, p-value = 0.1459
## alternative hypothesis: true autocorrelation is not 0

```

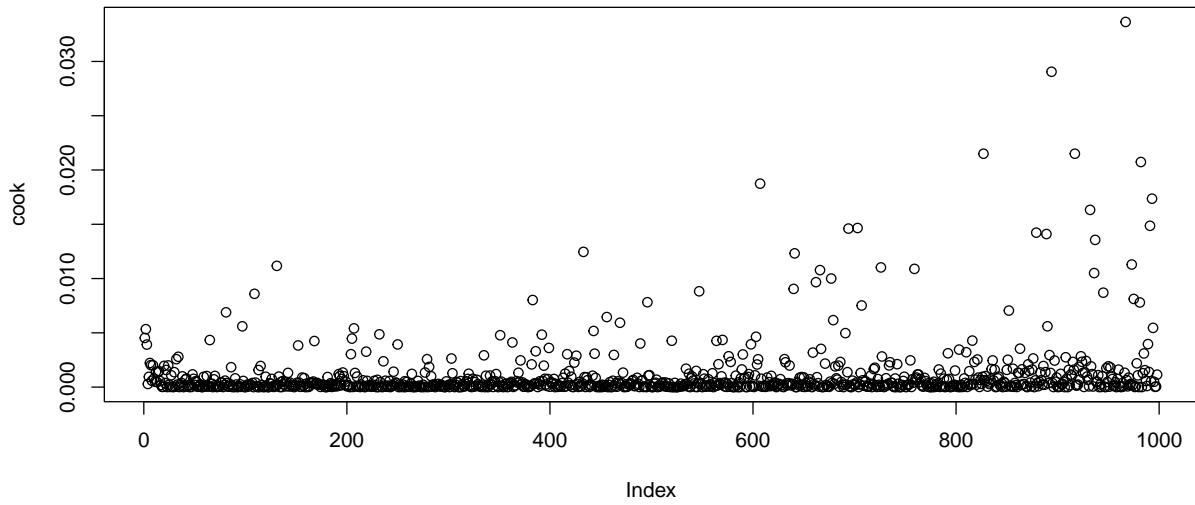
Como el p-valor del contraste Durbin-Watson es significativamente alto (superior a 0.10), podemos suponer independencia.

Seguidamente, comprobaremos que no hay multicolinealidad.

```
vif(modelo_log_final)

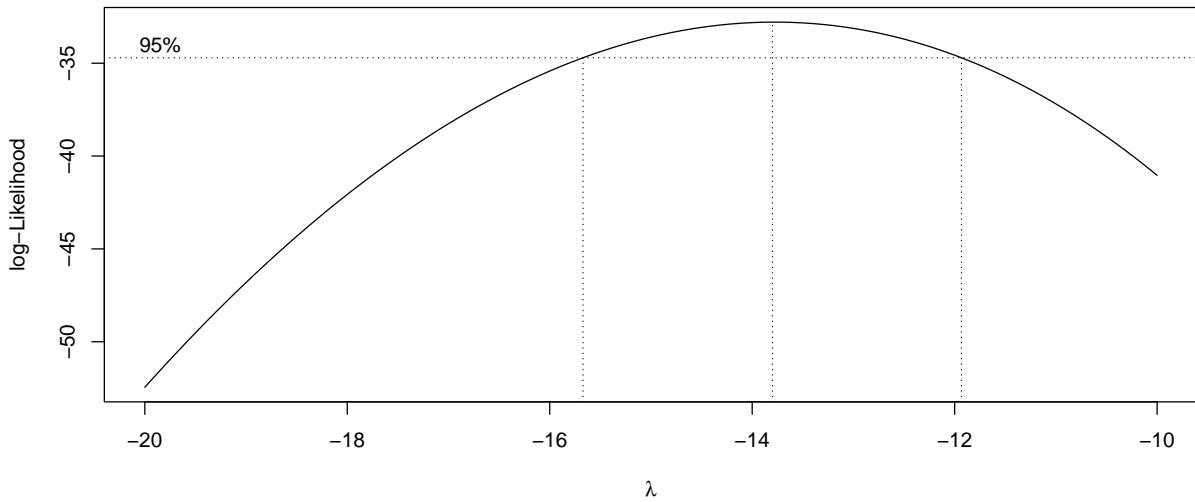
## Watch.time.Minutes.      Peak.viewers      Average.viewers
##           1.947333          2.424966          2.888362
```

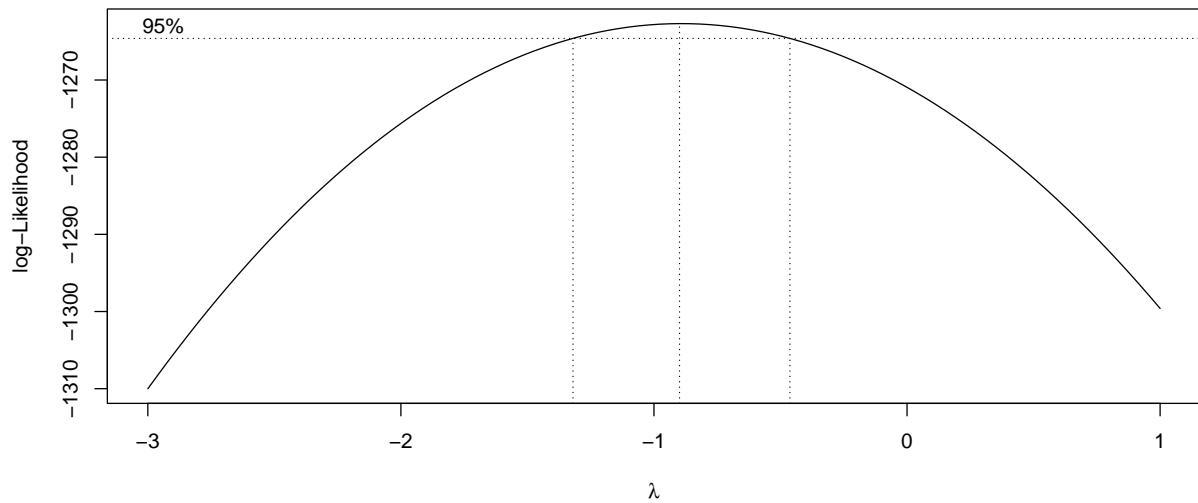
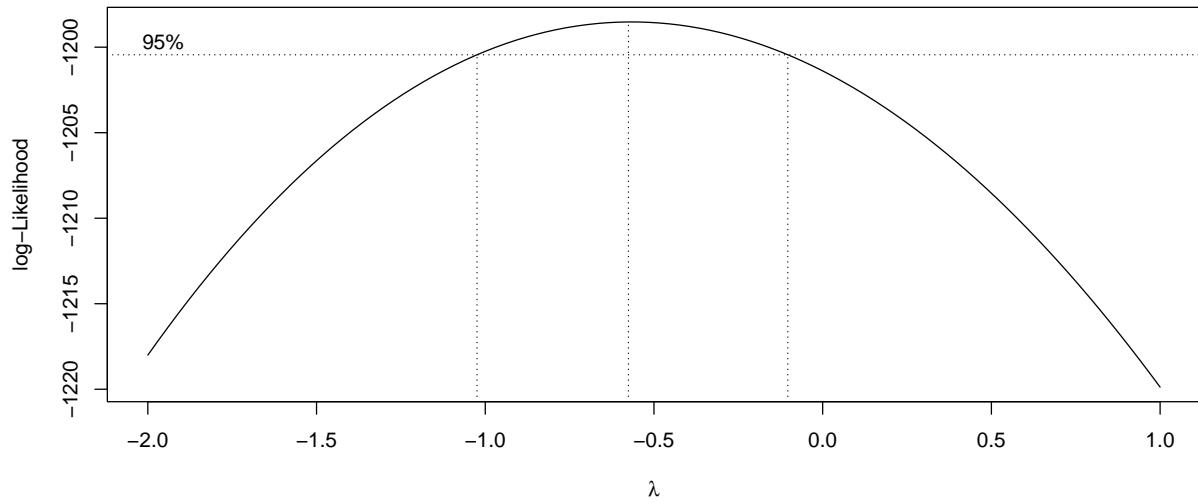
Por último, calcularemos las distancias de Cook para asegurar que no hay una observación influyente.



Hemos visto que nuestro modelo no pasa el test de normalidad y tampoco el de homocedasticidad. Para solucionarlo, vamos a hacer transformaciones en las variables predictoras de nuestro modelo y vamos a eliminar los datos con residuos más significativos.

Para encontrar la transformación más pertinente de los predictores, usamos la función `boxcox()`.





Observamos que las mejores transformaciones que se obtienen son:

- Para `Watch.time.Minutes.` : $\lambda = -14$ (aproximadamente)
- Para `Peak.viewers` : $\lambda = -1/2$ (aproximadamente)
- Para `Average.viewers` : $\lambda = -1$ (aproximadamente)

```
modelo_log_final <- lm(Y_new ~ I(Watch.time.Minutes. ^ (-14)) +
                           I(Peak.viewers ^ (-0.5)) +
                           I(Average.viewers ^ (-1)),
                           data = datos)

summary(modelo_log_final)

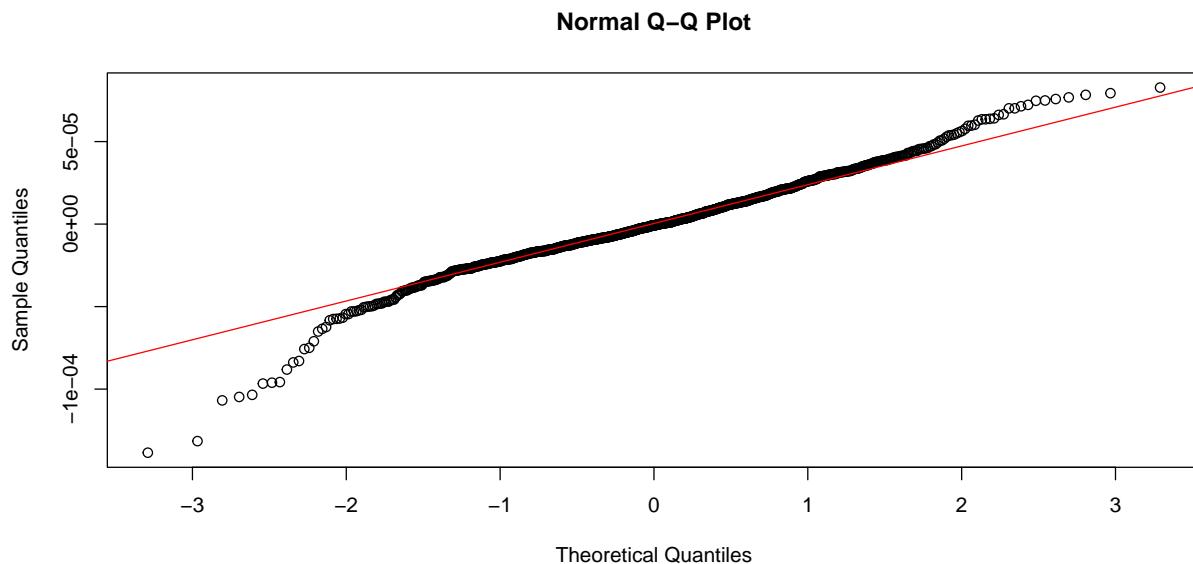
##
```

```

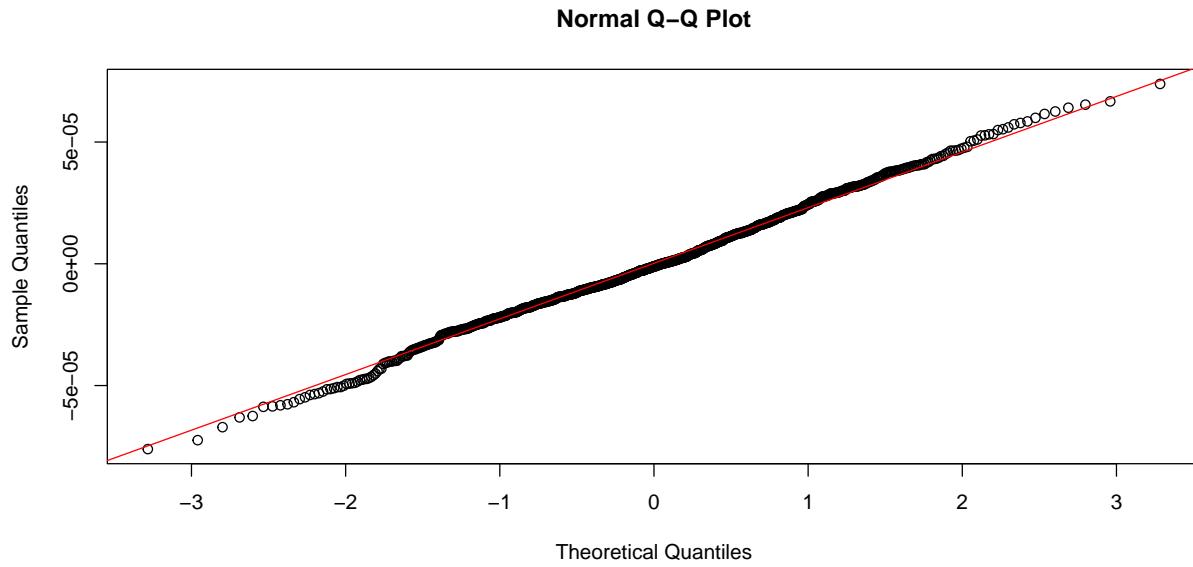
## Call:
## lm(formula = Y_new ~ I(Watch.time.Minutes.^(-14)) + I(Peak.viewers^(-0.5)) +
##      I(Average.viewers^(-1)), data = datos)
##
## Residuals:
##      Min        1Q    Median        3Q       Max 
## -1.385e-04 -1.547e-05 -8.170e-07  1.622e-05  8.275e-05 
## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)            4.316e-05  1.871e-05   2.307   0.0213 *  
## I(Watch.time.Minutes.^(-14)) 5.829e+13  2.785e+12  20.932  < 2e-16 *** 
## I(Peak.viewers^(-0.5))      3.942e-04  7.705e-05   5.115 3.76e-07 *** 
## I(Average.viewers^(-1))     2.485e-04  9.839e-05   2.526   0.0117 *  
## ---                        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 2.708e-05 on 994 degrees of freedom
## Multiple R-squared:  0.5614, Adjusted R-squared:  0.56 
## F-statistic:  424 on 3 and 994 DF,  p-value: < 2.2e-16

```

Volvemos a representar el gráfico de cuantiles, para identificar visualmente donde se encuentran los mayores residuos.



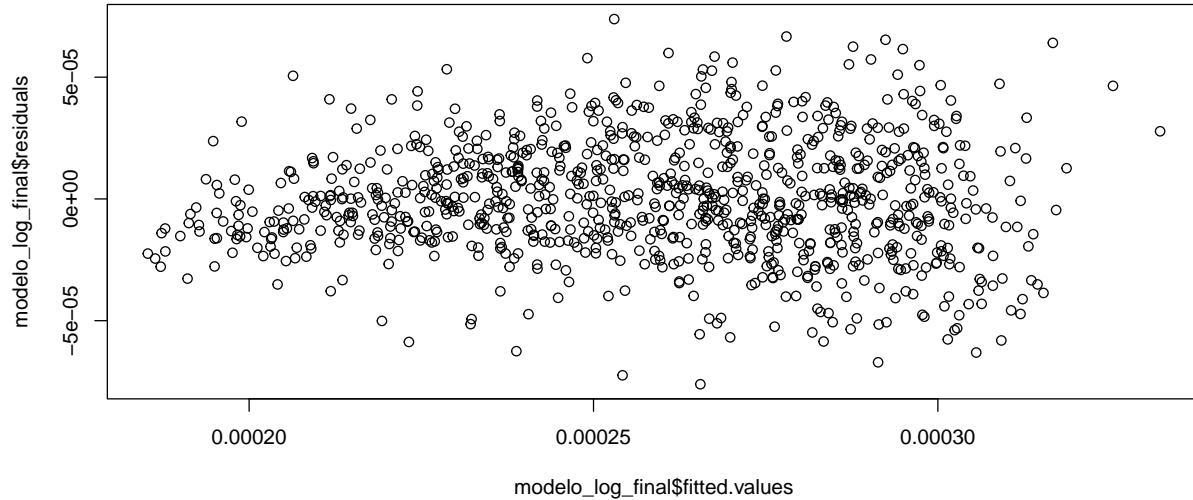
El gráfico de cuantiles tras eliminar estas observaciones (que generan estos residuos tan grandes) queda así.



```
residuos <- modelo_log_final$residuals
p.valor <- shapiro.test(residuos)$p.value
```

Obteniendo un p-valor de 0.0578402 y a un nivel de significación $\alpha = 0.05$ aceptamos la hipótesis de normalidad de los residuos.

Respecto a la suposición homocedasticidad, tenemos el siguiente resultado tras los pasos realizados:



La dispersión de los residuos respecto a 0 es más constante que antes, por lo que podríamos decir que ahora también se cumple la hipótesis de homocedasticidad. En definitiva, nuestro modelo ya estaría validado y listo para hacer inferencias con él.

6.4 Inferencias en el modelo RLM

Las inferencias que vamos a realizar consisten en la obtención de intervalos de confianza para los parámetros de regresión, intervalo de confianza para la media de la variable respuesta `Y_new` (la transformación de `Views.gained`) y los intervalos de predicción para la respuesta.

Table 6: Intervalos de confianza predictores

	2.5 %	97.5 %
(Intercept)	2.050000e-05	8.500000e-05
I(Watch.time.Minutes.^(-14))	5.488968e+13	6.461559e+13
I(Peak.viewers^(-0.5))	2.209000e-04	4.867000e-04
I(Average.viewers^(-1))	9.470000e-05	4.372000e-04

Por tanto, cualquier modelo teórico con coeficientes dentro de los intervalos dados serían válidos teniendo en cuenta los datos muestrales.

Posteriormente, haremos el intervalo de confianza para la media de la variable respuesta y el intervalo de predicción para la respuesta (al 95%). Por ejemplo, empleamos los valores $\log(\text{Watch.time.Minutes.}) = 19.5624$, $\log(\text{Peak.viewers}) = 9.3792$ y $\log(\text{Average.viewers}) = 7.6161$.

Table 7: Intervalo de confianza y (95%)

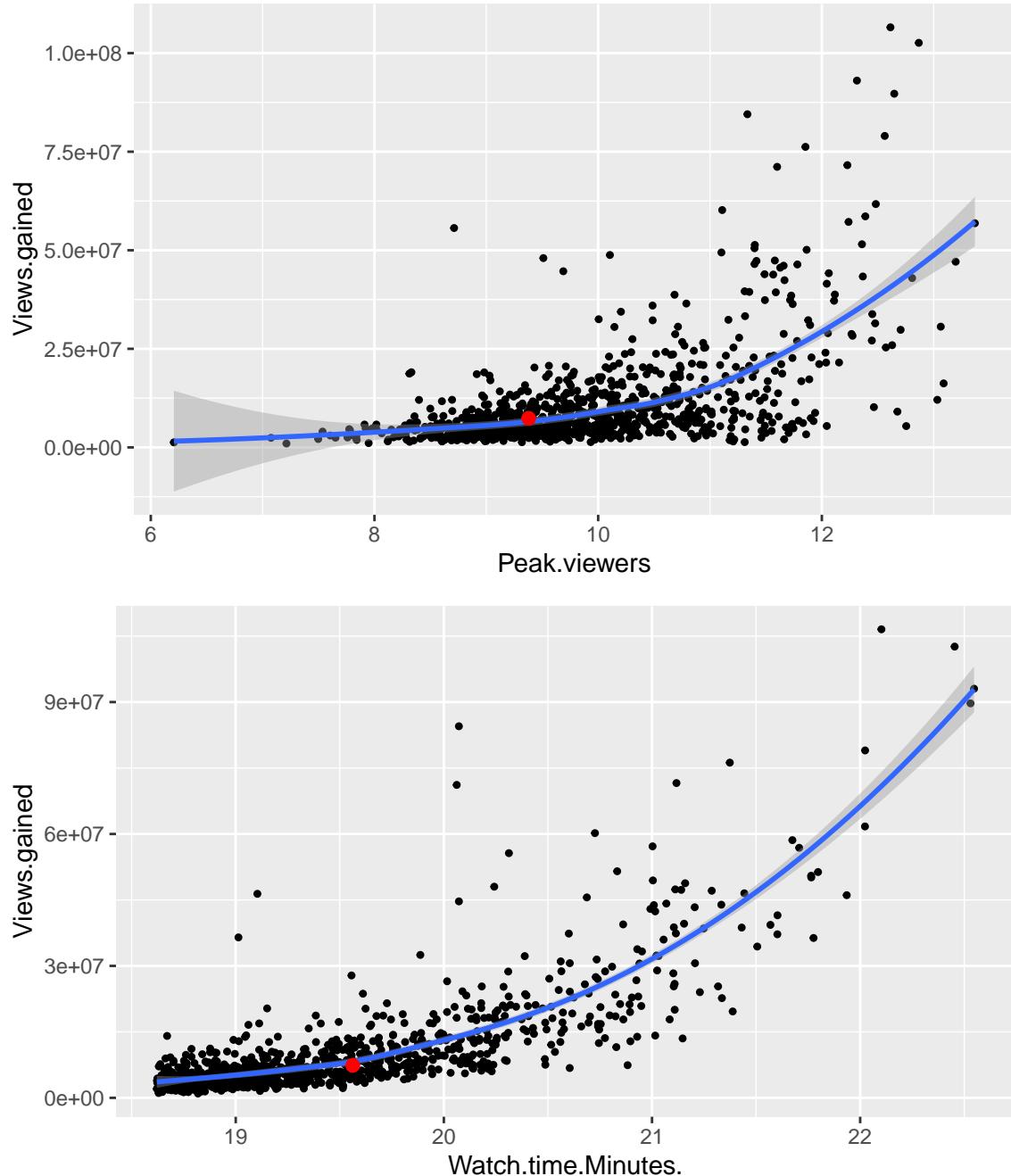
fit	lwr	upr
0.0002529	0.0002509	0.0002549

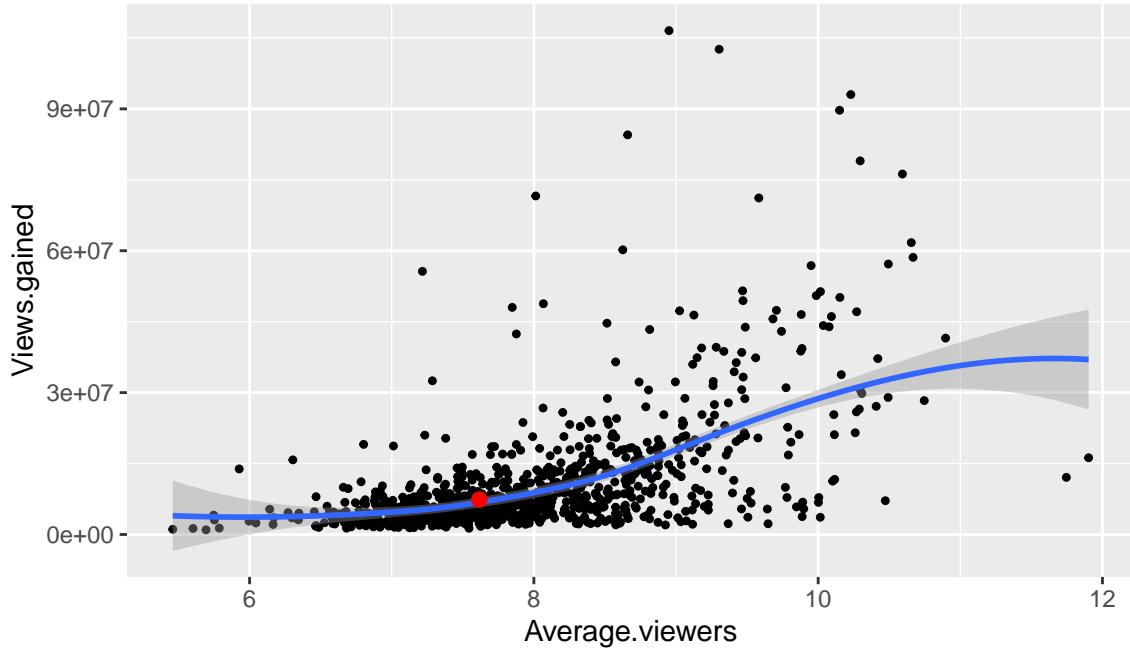
Table 8: Intervalo de predicción y (95%)

fit	lwr	upr
0.0002529	0.0002066	0.0002992

Consecuentemente, podemos decir que el valor esperado para `Y_new` (cuando el canal tiene esos valores de los logaritmos de `Watch.time.Minutes.`, `Peak.viewers` y `Average.viewers`) estaría entre 2.5088751×10^{-4} y 2.5487399×10^{-4} , y su predicción entre 2.065759×10^{-4} y 2.9918561×10^{-4} , con una confianza del 95%. El valor que devolvería el modelo ante dicha entrada sería de 2.5288075×10^{-4} para `Y_new`. Sin embargo, nos interesaría más saber el valor de la variable original `Views.gained`, por lo que sería tan simple como hacer la transformación inversa de la variable aplicando $e^{Y^{-1/3}}$, por lo que las visitas ganadas por ese canal durante el año 2020 serían 7.3742343×10^6 . A continuación, las gráficas donde aparece la predicción de nuestro modelo ante esta observación.

NOTA: Las curvas trazadas en las gráficas no pertenecen al modelo, sino a la trazada con el comando `geom_smooth()` con el método LOESS





7 Conclusiones

Para finalizar, recopilaremos las conclusiones a las que hemos llegado a lo largo de este informe.

- En primer lugar, los streamers hispanos fueron los que relativamente más crecieron en número de seguidores a lo largo de este periodo pandémico.
- En segundo lugar, los canales que llevan mucho tiempo acumulado retransmitiendo son más propensos a obtener peores valores en características como la media de espectadores en un directo o los seguidores ganados.
- Además, hemos podido establecer agrupaciones mediante técnicas de aprendizaje no supervisado (como K-means) en las que con el uso de PCA hemos podido catalogar distintos canales por su éxito en la plataforma y su compromiso con su respectiva comunidad.
- Por otra parte, a pesar de contar con distintas variables categóricas, no se ha podido aplicar técnicas de aprendizaje supervisado como análisis discriminante o el uso de la regresión logística para clasificar, debido a que la cantidad de observaciones de cada clase estaba bastante descompensada en algunos casos.
- Con la regresión multinomial, hemos podido trazar una curva que explica la tendencia de los datos de la variable respuesta (`Views.gained`) ante la variación de los valores de sus predictores. Para ello, se ha hecho una selección de características empleando distintos métodos y se ha concluido que los factores que más afectan a las visitas ganadas son el pico de viewers de los canales (lógico que cuanto más alto sea el pico más visitas nuevas se produzcan), las horas que lleva retransmitiendo el canal (porque cuanto más tiempo más probable es recibir nuevas visitas) y la media de espectadores en un directo. La eliminación de datos residuales junto a la transformación de las variables han permitido hallar una tendencia de los datos mucho más significativa respecto al estado inicial en el que se encontraban los datos.

En definitiva, se ha conseguido información sumamente concluyente de como fue este año para los canales de Twitch aplicando técnicas de análisis multivariante.