Sistemas de recomendación con R

Joaquín Amat Rodrigo j.amatrodrigo@gmail.com

Marzo, 2018

Tabla de contenidos

Versión PDF: Github

Introducción

Los sistemas de recomendación pueden definirse como herramientas diseñadas para interactuar con conjuntos de información grandes y complejos con la finalidad de proporcionar al usuario información o ítems que sean de su interés, todo ello de forma automatizada. Su funcionamiento se basa en el empleo de métodos matemáticos y estadísticos capaces de explotar la información previamente almacenada y crear recomendaciones adaptadas a cada usuario. En la actualidad, los sistemas de recomendación son una tecnología implementada en la mayoría de plataformas online como *Amazon*, *Neflix*, *eBay*... ya que han dado muy buenos resultados incrementando las ventas. También están presentes en muchos otros ámbitos, por ejemplo, el de las noticias, mostrando al usuario información que le interesa de forma rápida. La mayoría de sistemas de recomendación se pueden clasificar en tres grupos: basados en contenido, filtrado colaborativos y mixtos (combinación de los dos anteriores).

El objetivo de los ejemplos mostrados en este documento es facilitar la comprensión de las ideas que hay detrás de algunos de estos sistemas, no persiguen ser una implementación óptima y sofisticada, sino intuitiva. Para sistemas más optimizados pueden emplearse librerías como recommenderlab.

Basados en contenido (*content based filtering*)

Se fundamentan en la idea de *muéstrame más cosas como las que me han gustado*. Es de esperar que, productos similares a otros que han sido valorados positivamente por el usuario, también sean bien valorados.

Para que este tipo de sistemas de recomendación funcionen, se necesitan dos elementos: por un lado, disponer de un perfil de valoraciones del usuario en cuestión y por otro, una cuantificación de las características de los productos disponibles. A partir de los productos que han sido comprados o valorados positivamente por el usuario, se pueden recomendar otros cuyas características se asemejen a los anteriores. Un ejemplo de plataforma que emplea está aproximación es Pandora, una radio online que genera listas de reproducción adaptadas a cada usuario. Para lograrlo, dispone de una base de datos con una descripción detallada sobre el ritmo, estilo de música, instrumentos, temática... de cada canción. A partir de que el usuario escoge una canción inicial, se empiezan a sugerir canciones similares. Este tipo de sistemas de recomendación tiene una serie de limitaciones:

- Requiere establecer cuáles son los atributos/características con las que se describe a los productos, para a continuación, catalogar cada uno de ellos. La calidad de esta catalogación es clave en la identificación de productos similares, por lo que requiere de una persona experta en el tema.
- Si con el tiempo, se descubre que hay una nueva característica que es importante para los potenciales usuarios, se tiene que evaluar dicha característica en todos los productos. Para algunos modelos de negocio con cientos o miles de artículos, esta aproximación no es viable.
- Poca originalidad de las recomendaciones. Las recomendaciones suelen ser productos muy similares a los ya consumidos por el usuario.
- Los valores de los atributos no aportan información acerca de la calidad del producto.

A pesar de ello, en algunos escenarios, por ejemplo: canciones, artículos científicos…, donde los ítems se pueden describir de forma precisa con atributos (palabras clave, autor, temática…) consiguen dar muy buenos resultados.

Filtrado colaborativos (collaborative filtering)

Se fundamentan en la idea de *muéstrame cosas que le hayan gustado a gente parecida a mí*. Es de esperar que, usuarios con un perfil similar, tiendan a hacer valoraciones similares. Para que este tipo de sistemas de recomendación funcionen se necesita disponer de las valoraciones de los usuarios, así como información sobre sus perfiles (actividades, comportamiento, búsquedas, valoraciones previas…) que permita agruparlos por similitud. Sin embargo, no se necesita conocer la descripción detallada del producto en sí. En la actualidad, estos sistemas están muy extendidos por la facilidad con la que se puede recopilar información sobre las valoraciones y perfiles de los usuarios de la red. Es el método adecuado para modelos de negocio con miles de artículos, en los que no es factible describir con detalle los atributos de cada uno, pero puede recopilarse información de muchos clientes, un ejemplo es *Amazon*. La limitación de estos sistemas aparece en el momento que se une un nuevo usuario, ya que, hasta que no se dispone de suficiente información sobre su comportamiento, no se pueden hacer recomendaciones (*cold start*).

Aunque la idea de estudiar la similitud entre usuarios abarca el empleo de cualquier tipo de información (nivel económico, trabajo, hobby, estado familiar, búsquedas por internet…) en la práctica, una empresa que vende determinados productos suele tener acceso únicamente a las valoraciones que los usuarios han hecho de ellos, por lo que, generalmente, el perfil de un usuario se describe en función de las valoraciones que hace. Esta es la aproximación que se va a emplear en los siguientes ejemplos, pero, si se dispusiera de más información, podría añadirse y se estaría enriqueciendo el sistema.

Dentro de los filtrados colaborativos se diferencian dos tipos:

Basados en usuario (*user-based*)

Para predecir la valoración que un usuario *A* hará de un ítem *X* que todavía no ha visto, se buscan usuarios con perfiles similares a *A* y se utilizan las valoraciones de estos otros usuarios sobre el ítem *X* como estimación de la valoración de *A*.

Basados en ítems (item-based)

Para predecir la valoración que un usuario *A* hará de un ítem *X* que todavía no ha visto, se buscan otros ítems similares (en función del perfil de valoraciones que han recibido) y que el usuario *A* también haya valorado. Se utilizan las valoraciones que el propio usuario *A* ha hecho de los ítems similares como predicción de su valoración sobre el ítem *X*. Este sistema puede parecer similar al basado en contenido, la diferencia se encuentra en que cada ítem no está definido por sus atributos sino por el perfil de valoraciones que ha recibido.

Las diferencias entre los tres sistemas descritos pueden parecer confusas, en los siguientes apartados se muestran ejemplos muy simplificados que facilitan su comprensión.

Medidas de valoración

A la hora de entrenar un sistema de recomendación, es crítico entender de qué forma se están haciendo las valoraciones, dependiendo de ello, se pueden emplear unos algoritmos u otros. En la actualidad, las principales formas en las que se mide la valoración que los usuarios tienen sobre un producto o información son las siguientes:

- Valoración binaria: son del tipo comprado *vs* no comprado, visto *vs* no visto, me gusta *vs* no me gusta.

- Numérica: las valoraciones se hacen empleando una escala numérica o que puede ser traducida a números, por ejemplo, las valoraciones del 1 al 5 o de 1 a 5 estrellas.

- Otras medidas: como pueden ser el número de *clicks*, número de reproducciones…

Medidas de similitud o distancia

Todos los sistemas de recomendación tienen una cosa en común, para poder llevar a cabo las predicciones, necesitan definir y cuantificar la similitud entre ítems o usuarios. El término distancia se emplea dentro del contexto de recomendadores como cuantificación de la similitud o diferencia entre observaciones. Si se representan las observaciones en un espacio p dimensional, siendo p el número de variables asociadas a cada observación (ítem o usuario), cuando más se asemejen dos observaciones, más próximas estarán, de ahí que se emplee el término distancia. La característica que hace de los sistemas de recomendación un método adaptable a escenarios muy diversos es que pueden emplear cualquier tipo de distancia, lo que permite al investigador escoger la más adecuada para el estudio en cuestión. Algunas de las empleadas con más frecuencia son: distancia euclídea, correlación Pearson, correlación Spearman, distancia coseno e índice *Jaccard*.

Distancia euclídea

La distancia euclídea entre dos puntos p y q se define como la longitud del segmento que une ambos puntos. En coordenadas cartesianas, la distancia euclídea se calcula empleando el teorema de Pitágoras. Por ejemplo, en un espacio de dos dimensiones, en el que cada punto está definido por las coordenadas (x, y), la distancia euclídea entre p y q viene dada por la ecuación:

$$d_{euc}(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

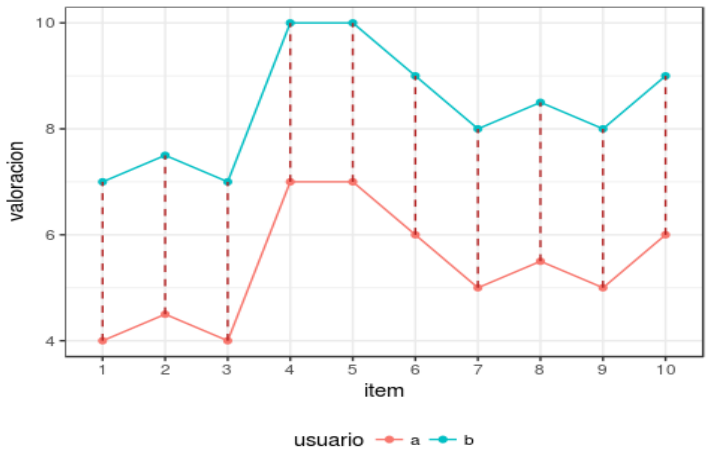
Esta ecuación puede generalizarse para un espacio euclídeo n-dimensional donde cada punto está definido por un vector de n coordenadas: $p = (p_1, p_2, p_3, \ldots, p_n)$ y $q = (q_1, q_2, q_3, \ldots, q_n)$.

$$d_{euc}(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} =$$

$$\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

La siguiente imagen muestra el perfil de dos usuarios definidos por las valoraciones que han hecho de 10 ítems (espacio con 10 dimensiones).

```
library(tidyverse)
usuario_a <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6)
usuario_b <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6) + 3
datos <- data.frame(usuario = rep(c("a", "b"), each = 10),
                    valoracion = c(usuario_a, usuario_b),
                    item = 1:10)
ggplot(data = datos, aes(x = as.factor(item), y = valoracion,
                         colour = usuario)) +
  geom_path(aes(group = usuario)) +
  geom_point() +
  geom_line(aes(group = item), colour = "firebrick", linetype = "dashed") +
  labs(x = "item") +
  theme_bw() + theme(legend.position = "bottom")
```



La distancia euclídea entre las dos observaciones equivale a la raíz cuadrada de la suma de las longitudes de los segmentos rojos que unen cada par de puntos. Tiene en cuenta, por lo tanto, el desplazamiento individual de cada una de las variables.

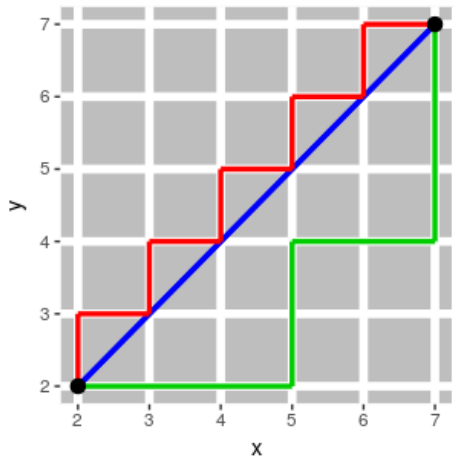
Distancia de Manhattan

La distancia de Manhattan, también conocida como *taxicab metric*, *rectilinear distance* o *L1 distance*, define la distancia entre dos puntos p y q como el sumatorio de las diferencias absolutas entre cada dimensión. Esta medida se ve menos afectada por *outliers* (es más robusta) que la distancia euclídea debido a que no eleva al cuadrado las diferencias.

$$d_{man}(p,q) = \sum_{i=1}^{n} |\,(p_i - q_i)|$$

La siguiente imagen muestra una comparación entre la distancia euclídea (segmento azul) y la distancia de manhattan (segmento rojo y verde) en un espacio bidimensional. Existen múltiples caminos para unir dos puntos con el mismo valor de distancia de manhattan, ya que su valor es igual al desplazamiento total en cada una de las dimensiones.

```
datos <- data.frame(observacion = c("a", "b"), x = c(2,7), y = c(2,7))
manhattan <- data.frame(
              x = rep(2:6, each = 2),
              y = rep(2:6, each = 2) + rep(c(0,1), 5),
              xend = rep(2:6, each = 2) + rep(c(0,1), 5),
              yend = rep(3:7, each = 2))
manhattan_2 <- data.frame(x = c(2, 5, 5, 7), y = c(2, 2, 4, 4),
                xend = c(5, 5, 7, 7), yend = c(2, 4, 4, 7))


ggplot(data = datos, aes(x = x, y = y)) +
geom_segment(aes(x = 2, y = 2, xend = 7, yend = 7), color = "blue", size = 1.2) +
geom_segment(data = manhattan, aes(x = x, y = y, xend = xend, yend = yend),
             color = "red", size = 1.2) +
geom_segment(data = manhattan_2, aes(x = x, y = y, xend = xend, yend = yend),
             color = "green3", size = 1.2) +
geom_point(size = 3) +
theme(panel.grid.minor = element_blank(),
      panel.grid.major = element_line(size = 2),
      panel.background = element_rect(fill = "gray",c olour = "white",
                                      size = 0.5, linetype = "solid"))
```



Correlación

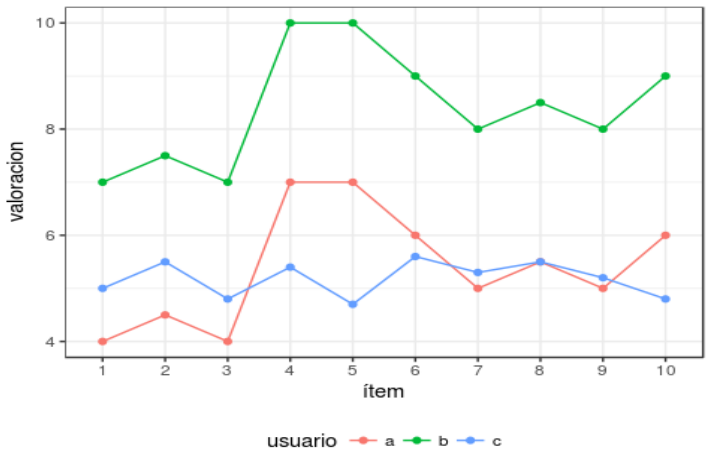
La correlación es una medida de distancia muy útil cuando la definición de similitud se hace en términos de patrón o forma y no de desplazamiento o magnitud. ¿Qué quiere decir esto? En la imagen del apartado de la distancia euclídea, los dos usuarios tienen exactamente el mismo patrón, la única diferencia es que uno de ellos está desplazado 3 unidades por encima del otro. Si se emplea como medida de similitud 1 menos el valor de la correlación, ambas observaciones se consideran idénticas (su distancia es 0).

$$d_{cor}(p,q) = 1 - \text{correlacion}(p,q)$$

donde la correlación puede ser de distintos tipos (*Pearson*, *Spearman*, *Kendall*...) Correlación lineal.

En la siguiente imagen se muestra el perfil de 3 usuarios. Acorde a la distancia euclídea, las observaciones *b* y *c* son las más similares, mientras que acorde a la correlación de Pearson, las observaciones más similares son *a* y *b*.

```
library(ggplot2)
usuario_a <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6)
usuario_b <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6) + 3
usuario_c <- c(5, 5.5, 4.8, 5.4, 4.7, 5.6, 5.3, 5.5, 5.2, 4.8)
datos <- data.frame(usuario = rep(c("a", "b", "c"), each = 10),
                    valoracion = c(usuario_a, usuario_b, usuario_c),
                    item = 1:10)
ggplot(data = datos, aes(x = as.factor(item), y = valoracion, colour = usuario)) +
  geom_path(aes(group = usuario)) +
  geom_point() +
  labs(x = "ítem") +
  theme_bw() +
  theme(legend.position = "bottom")
```



```
dist(x = rbind(usuario_a, usuario_b, usuario_c), method = "euclidean")
```

```
##           usuario_a usuario_b
## usuario_b  9.486833
## usuario_c  3.495712 10.743370
```

```
1 - cor(x = cbind(usuario_a, usuario_b, usuario_c), method = "pearson")
```

```
##           usuario_a usuario_b usuario_c
## usuario_a 0.0000000 0.0000000 0.9757201
## usuario_b 0.0000000 0.0000000 0.9757201
## usuario_c 0.9757201 0.9757201 0.0000000
```

Este ejemplo pone de manifiesto que no existe una única medida de distancia que sea mejor que las demás, sino que, dependiendo del contexto, una es más adecuada que otra.

Jackknife correlation

El coeficiente de correlación de *Pearson* resulta efectivo en ámbitos muy diversos. Sin embargo, tiene la desventaja de no ser robusto frente a *outliers* a pesar de que se cumpla la condición de normalidad. Si dos variables tienen un pico o un valle común en una única observación, por ejemplo, por un error de lectura, la correlación va a estar dominada por este registro a pesar de que entre las dos variables no haya correlación real alguna. Lo mismo puede ocurrir en la dirección opuesta. Si dos variables están altamente correlacionadas excepto para una observación, en la que los valores son muy dispares, entonces la correlación existente quedará enmascarada. Una forma de evitarlo es recurrir a la *Jackknife correlation*, que consiste en calcular todos los posibles coeficientes de correlación entre dos variables si se excluye cada vez una de las observaciones. El promedio de todas las *Jackknife correlations* calculadas atenúa en cierta medida el efecto del *outlier*.

$$\overline{\theta}_{(A,B)} = \text{Promedio Jackknife correlation (A, B)} = \frac{1}{n}\sum_{i=1}^{n}\hat{r}_i$$

donde n es el número de observaciones y $\hat{r}_i$ es el coeficiente de correlación entre las variables A y B, habiendo excluido la observación i.

Además del promedio, se puede estimar su error estándar (SE) y así obtener intervalos de confianza para la *Jackknife correlation* y su correspondiente *p-value*.

$$SE = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{r}_i - \overline{\theta})^2}$$

Intervalo de confianza del 95% ($Z = 1.96$):

$$\text{Promedio Jackknife correlation (A, B)} \pm 1.96 * SE$$

$$\overline{\theta} - 1.96 \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{r}_i - \overline{\theta})^2}\ ,\ \ \overline{\theta} + 1.96 \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{r}_i - \overline{\theta})^2}$$

P-value para la hipótesis nula de que $\overline{\theta} = 0$:

$$Z_{calculada} = \frac{\overline{\theta} - H_0}{SE} = \frac{\overline{\theta} - 0}{\sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{r}_i - \overline{\theta})^2}}$$

$$p_{value} = P(Z > Z_{calculada})$$

Cuando se emplea este método, es conveniente calcular la diferencia entre el valor de correlación obtenido por *Jackknife correlation* ($\overline{\theta}$) y el que se obtiene si se emplean todas las observaciones ($\overline{r}$). A esta diferencia se le conoce como *Bias*. Su magnitud es un indicativo de cuanto está influenciada la estimación de la correlación entre dos variables debido a un valor atípico u *outlier*.

$$Bias = (n-1) * (\overline{\theta} - \hat{r})$$

Si se calcula la diferencia entre cada correlación ($\hat{r}_i$) estimada en el proceso de *Jackknife* y el valor de correlación ($\hat{r}$) obtenido si se emplean todas las observaciones, se puede identificar que observaciones son más influyentes.

Cuando el estudio requiere minimizar al máximo la presencia de falsos positivos, a pesar de que se incremente la de falsos negativos, se puede seleccionar como valor de correlación el menor de entre todos los calculados en el proceso de *Jackknife*.

$$Correlacion = min\{\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_n\}$$

A pesar de que el método de *Jackknife* permite aumentar la robustez de la correlación de *Pearson*, si los *outliers* son muy extremos su influencia seguirá siendo notable.

Simple matching coefficient

Cuando las variables con las que se pretende determinar la similitud entre observaciones son de tipo binario, a pesar de que es posible codificarlas de forma numérica como 1 o 0, no tiene sentido aplicar operaciones aritméticas sobre ellas (media, suma…). Por ejemplo, si la variable sexo se codifica como 1 para mujer y 0 para hombre, carece de significado decir que la media de la variable sexo en un determinado set de datos es 0.5. En situaciones como esta, no se pueden emplear medidas de similitud basadas en distancia euclídea, manhattan, correlación…

Dado dos objetos A y B, cada uno con n atributos binarios, el *simple matching coefficient (SMC)* define la similitud entre ellos como:

$$SMC = \frac{\text{número coincidencias}}{\text{número total de atributos}} = \frac{M_{00} + M11}{M_{00} + M_{01} + M_{10} + M_{11}}$$

donde M_{00} y M_{11} son el número de variables para las que ambas observaciones tienen el mismo valor (ambas 0 o ambas 1), y M_{01} y M_{10} el número de variables que no coinciden. El valor de distancia *simple matching distance (SMD)* se corresponde con $1 - SMC$.

Índice Jaccard

El índice *Jaccard* o coeficiente de correlación *Jaccard* es similar al *simple matching coefficient (SMC)*. La diferencia radica en que el *SMC* tiene el término M_{00} en el numerador y denominador, mientras que el índice de *Jaccard* no. Esto significa que, *SMC*, considera como coincidencias tanto si el atributo está presente en ambos sets como si el atributo no está en ninguno de los sets, mientras que *Jaccard* solo cuenta como coincidencias cuando el atributo está presente en ambos sets.

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

o en términos matemáticos de sets:

$$J(A, B) = \frac{A \cap B}{A \cup B}$$

La distancia de *Jaccard* $(1 - J)$ supera a la *simple matching distance* en aquellas situaciones en las que la coincidencia de ausencia no aporta información. Para ilustrar este hecho, supóngase que se quiere cuantificar la similitud entre dos clientes de un supermercado en base a los artículos comprados. Es de esperar que cada cliente solo adquiera unos pocos artículos de los muchos disponibles, por lo que el número de artículos no comprados por

ninguno (M_{00}) será muy alto. Como la distancia de *Jaccard* ignora las coincidencias de tipo M_{00}, el grado de similitud dependerá únicamente de las coincidencias entre los artículos comprados.

Distancia coseno

El coseno del ángulo que forman dos vectores puede interpretarse como una medida de similitud de sus orientaciones, independientemente de sus magnitudes. Si dos vectores tienen exactamente la misma orientación (el ángulo que forman es 0º) su coseno toma el valor de 1, si son perpendiculares (forman un ángulo de 90º) su coseno es 0 y si tienen orientaciones opuestas (ángulo de 180º) su coseno es de -1.

$$\cos(\alpha) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||\ ||\mathbf{y}||}$$

Si los vectores se normalizan restándoles la media ($X - \overline{X}$), la medida recibe el nombre de coseno centrado y es equivalente a la correlación de Pearson.

```
a <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6)
b <- c(5, 5.5, 4.8, 5.4, 4.7, 5.6, 5.3, 5.5, 5.2, 4.8)

# Correlación de Pearson
cor(x = a, y = b, method = "pearson")
```

```
## [1] 0.02427991
```

```
# Coseno
coseno <- function(x, y){
  resultado <- x%*%y / (sqrt(x %*% x) * sqrt(y %*%y ))
  return(as.numeric(resultado))
}

coseno(a,b)
```

```
## [1] 0.9802813
```

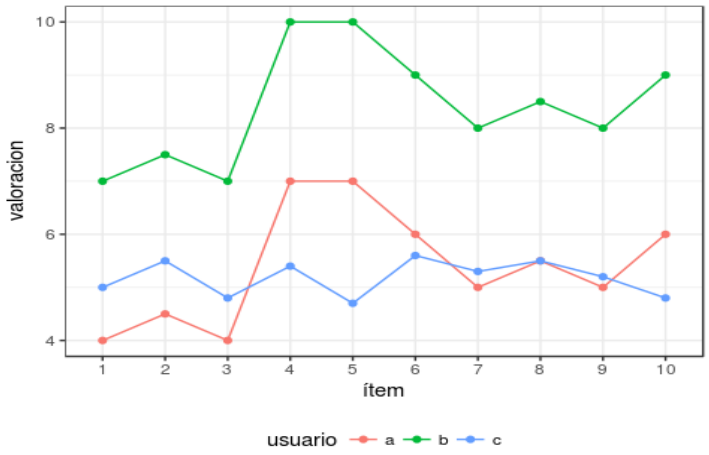
```
# Coseno tras centrar los vectores
a <- a - mean(a)
b <- b - mean(b)
coseno(a,b)
```

```
## [1] 0.02427991
```

Estandarización de valoraciones

En el ámbito de los recomendadores, cuando se habla de dos usuarios similares, suele hacerse referencia al hecho de que ambos usuarios tienen un perfil de valoraciones similar, es decir, que coinciden en los ítems que valoran de forma positiva y los que valoran de forma negativa. Sin embargo, más que el valor exacto de las valoraciones, importa el patrón o tendencia de las mismas. En la siguiente imagen pueden verse las valoraciones que 3 usuarios han hecho de 10 ítems.

```
usuario_a <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6)
usuario_b <- c(4, 4.5, 4, 7, 7, 6, 5, 5.5, 5, 6) + 3
usuario_c <- c(5, 5.5, 4.8, 5.4, 4.7, 5.6, 5.3, 5.5, 5.2, 4.8)

datos <- data.frame(usuario = rep(c("a", "b", "c"), each = 10),
                    valoracion = c(usuario_a, usuario_b, usuario_c),
                    item = 1:10)

ggplot(data = datos, aes(x = as.factor(item), y = valoracion,
                         colour = usuario)) +
  geom_path(aes(group = usuario)) +
  geom_point() +
  labs(x = "ítem") +
  theme_bw() +
  theme(legend.position = "bottom")
```

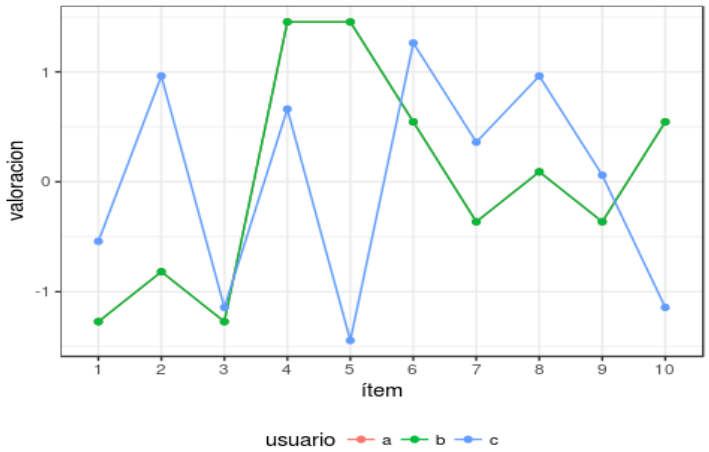


Los usuarios *a* y *b* tienen exactamente el mismo patrón, la única diferencia es que las valoraciones del usuario *b* están desplazadas por encima de las del usuario *a* 3 unidades. Acorde a la distancia euclídea, los usuarios *b* y *c* son los más similares a pesar de que sus patrones son mucho más dispares. Existen dos formas de evitar que variaciones en las escalas

ocullen patrones: utilizar medidas de distancia independientes de la escala, por ejemplo, la correlación de *Pearson* o *Kendall*, o bien estandarizar/centrar los datos.

Véase el resultado al repetir la misma representación habiendo estandarizado previamente los datos (restar la media y dividir por la desviación típica).

```
datos <- datos %>% group_by(usuario) %>%
         mutate(valoracion = scale(valoracion)) %>% ungroup()

ggplot(data = datos, aes(x = as.factor(item), y = valoracion,
                         colour = usuario)) +
  geom_path(aes(group = usuario)) +
  geom_point() +
  labs(x = "ítem") +
  theme_bw() +
  theme(legend.position = "bottom")
```



El perfil de los usuarios *a* y *b* pasa a ser idéntico, por lo que se superponen en la imagen.

Cabe destacar que, si se aplica la estandarización descrita, existe una relación entre la distancia euclídea y la correlación de Pearson que hace que los resultados de similitud obtenidos sean proporcionales.

$$d_{euc}(p, q\ estandarizados) = \sqrt{2n(1 - cor(p, q))}$$

A modo general, suele ser recomendable estandarizar o centrar las variables antes de calcular las similitudes.

Ejemplo filtrado colaborativo

Para aplicar sistemas de recomendación basados en filtrado colaborativo es necesario definir una serie de elementos.

- Set de usuarios $U = u_1, u_2, \ldots, u_m$
- Set de ítems $I = i_1, i_2, \ldots, i_n$
- Matriz de valoración R de dimensiones $m\ x\ n$, donde cada fila representa a un usuario y cada columna un ítem. El valor R_{jk} es la valoración del usuario u_j sobre el ítem i_k. R puede ser una matriz dispersa, no todos los usuarios tienen que haber valorado todos los ítems.
- Usuario sobre el que se quiere hacer una predicción (usuario activo) u_x.

Supóngase que se dispone del historial de valoraciones que 4 usuarios $(u_1, u_2, \ldots, u_4)$ han hecho sobre 5 ítems $(i_1, i_2, \ldots, i_5)$. Un nuevo usuario (u_x) no ha valorado el ítem (i_5). Se pretende aplicar un sistema de recomendación colaborativo para predecir la valoración del usuario (u_x) sobre el ítem i_5.

```
datos <- matrix(c(5, 3, 4, 4, NA, 3, 1, 2, 3, 3, 4, 3, 4, 3, 5, 3, 3, 1, 5, 4, 1,
                  5, 5, 2, 1),  nrow = 5, byrow = TRUE)
colnames(datos) <- c("i_1", "i_2", "i_3", "i_4", "i_5")
rownames(datos) <- c("u_x", "u_1", "u_2", "u_3", "u_4")
datos
```

```
##     i_1 i_2 i_3 i_4 i_5
## u_x   5   3   4   4  NA
## u_1   3   1   2   3   3
## u_2   4   3   4   3   5
## u_3   3   3   1   5   4
## u_4   1   5   5   2   1
```

Sistema basado en usuarios

Los sistemas de filtrado colaborativo basado en usuarios predicen la valoración que un determinado usuario hará sobre un producto utilizando las valoraciones que han hecho sobre ese mismo producto los n usuarios más parecidos a él. La similitud entre usuarios se mide acorde al patrón de valoraciones que tiene cada uno, en este caso, las filas de la matriz.

En primer lugar, se calcula la similitud entre el usuario u_x y el resto de usuarios. En este caso, se emplea como medida de similitud el coeficiente de correlación de Pearson. Como los usuarios están definidos por las filas, hay que transponer la matriz para realizar los cálculos.

```
library(tidyverse)
matriz_dist <- datos %>% t() %>% cor(method = "pearson", use = "complete.obs")
matriz_dist %>%  as.data.frame %>% rownames_to_column(var = "Usuario_A") %>%
                gather(key = "Usuario_B", value = "corr", -Usuario_A) %>%
                filter(corr != 1 & Usuario_A == "u_x") %>%
                arrange(desc(corr))
```

```
##   Usuario_A Usuario_B       corr
## 1       u_x       u_1  0.8528029
## 2       u_x       u_2  0.7071068
## 3       u_x       u_3  0.0000000
## 4       u_x       u_4 -0.7921180
```

Una vez ordenados los usuarios de mayor a menor similitud respecto al usuario u_x, se procede a calcular la predicción de la valoración. Existen varias formas de hacerlo:

- Promedio las valoraciones (R_{jk}) de los n usuarios más cercanos. Con esto, se evita tener en cuenta la valoración de usuarios que tienen un perfil muy distinto del usuario de interés. n se debe considerar como un hiperparámetro cuyo valor óptimo se identifica, por ejemplo, mediante validación cruzada. Véase el resultado de este problema si se emplea $n = 3$. Los 3 usuarios más similares a u_x son : u_1, u_2, u_3. La predicción de la valoración que hace el usuario u_x sobre el ítem i_5 se obtiene como la media de las valoraciones que cada uno de los usuarios seleccionados tiene sobre el ítem 5 ($R_{usuario,item_5}$).

$$\text{Predicción}(u_x, i_5) = \frac{R_{u_1,i_5} + R_{u_2,i_5} + R_{u_3,i_5}}{3}$$

```
prediccion <- mean(3, 5, 4)
prediccion
```

```
## [1] 3
```

- El inconveniente de la aproximación anterior es que los n usuarios seleccionados tienen el mismo peso en la predicción, sin embargo, no todos se parecen en la misma medida al usuario de interés. Una forma de compensar esta influencia es ponderando la media con los valores de similitud, de esta forma, la valoración de los usuarios pesa más cuanto más se parecen al usuario estudiado. Esta estrategia solo puede aplicarse cuando la similitud toma valores en el rango [0, número positivo], ya que, la media aritmética ponderada, no está definida para pesos negativos y, al menos uno de los pesos, debe ser mayor de cero (wikipedia). Otra opción es considerar las similitudes negativas como 0 de forma que no contribuyen en el cálculo.

$$\text{Predicción}(u_x, i_5) = \frac{\text{corr}(u_x, u_1) \times R_{u_1,i_5} + \text{corr}(u_x, u_2) \times R_{u_2,i_5} + \text{corr}(u_x, u_3) \times R_{u_3,i_5}}{\text{corr}(u_x, u_1) + \text{corr}(u_x, u_2) + \text{corr}(u_x, u_3)}$$

```
prediccion <- (0.85 * 3 + 0.71 * 5 + 0 * 4) / (0.85 + 0.71 + 0)
prediccion
```

```
## [1] 3.910256
```

- Los ítems que tienen siempre valoraciones muy positivas aportan poca información sobre el perfil de usuarios. Por ejemplo, un reloj que esté de oferta al precio de 1 euro cuando realmente vale 100, posiblemente sea bien valorado por todos los usuarios independientemente de su perfil de preferencias. Para evitar este problema se pueden pesar los ítems en función de su varianza en el momento que se calculan las similitudes entre perfiles, o eliminar aquellos ítems con varianza próxima a cero.

Sistema basado en ítems

La idea es muy similar al método basado en usuarios, pero en este caso, se identifican ítems similares (empleando el perfil de valoraciones que han recibido) en lugar de usuarios similares. Además, los ítems que participan en el proceso tienen que haber sido valorados por el usuario de interés u_x.

En primer lugar, se calcula la similitud entre el ítem i_5 y el resto de ítems. En la matriz *datos*, se corresponde con la similitud entre columnas. En este ejemplo, se emplea como medida de similitud el coeficiente de correlación de Pearson.

```
library(tidyverse)
matriz_dist <- datos %>% cor(method = "pearson", use = "complete.obs")
matriz_dist %>% as.data.frame %>% rownames_to_column(var = "Item_A") %>%
                gather(key = "Item_B", value = "corr", -Item_A) %>%
                filter(corr != 1 & Item_A == "i_5") %>%
                arrange(desc(corr))
```

```
##   Item_A Item_B       corr
## 1    i_5    i_1  0.9694584
## 2    i_5    i_4  0.5816751
## 3    i_5    i_3 -0.4276180
## 4    i_5    i_2 -0.4780914
```

Una vez calculadas las similitudes entre el ítem i_5 y el resto, se seleccionan los n ítems más parecidos y se obtiene la predicción a partir de las valoraciones que el usuario u_x ha hecho de esos n ítems.

- Predicción basada en el promedio de los $n = 3$ ítems más parecidos (i_1, i_4, i_3:

$$\text{Predicción}(u_x, i_5) = \frac{R_{u_x,i_1} + R_{u_x,i_4} + R_{u_x,i_3}}{3}$$

```
prediccion <- mean(c(3, 1, 4))
prediccion
```

```
## [1] 2.666667
```

- Predicción basada en el promedio ponderado por similitud:

$$\text{Predicción}(u_x, i_5) = \frac{\text{corr}(i_5, i_1) \times R_{u_x,i_1} + \text{corr}(i_5, i_4) \times R_{u_x,i_4} + \text{corr}(i_5, i_3) \times R_{u_x,i_3}}{\text{corr}(u_x, i_1) + \text{corr}(u_x, i_4) + \text{corr}(u_x, i_3)}$$

```
# Los valores de correlación negativos se sustituyen por 0 para poder calcular
# una media ponderada.
prediccion <- (3 * 0.97 + 1 * 0.58 + 4 * 0) / (0.97 + 0.58 + 0)
prediccion
```

```
## [1] 2.251613
```

Ejemplo recomendador de películas

El set de datos MovieLense del paquete recommenderlab, contiene información sobre más de 1000 películas, tanto variables descriptivas de cada largometraje como las valoraciones de más de 900 usuarios. Empleando este set de datos, se generan 3 tipos de sistemas de recomendación con el objetivo de recomendar 10 nuevas películas al usuario 329.

Se ha elegido el usuario 329 porque el número de películas que ha visto se corresponde a la mediana de películas vistas por los usuarios.

Carga y exploración de los datos

Las valoraciones de los usuarios se encuentran almacenadas en un objeto de tipo realRatingMatrix llamado *MovieLense* y la descripción de las películas en un *dataframe* llamado *MovieLenseMeta*. Para facilitar su manejo, se almacenan ambos sets de datos en formato de *dataframe* y se renombran como *valoraciones* y *atributos*.

Nota: En pasos posteriores, en concreto para cálculos entre vectores, el formato de matriz es más adecuado. Sin embargo, como el volumen de datos no supone un problema de memoria, se volverán a crear las matrices cuando se necesiten.

```
library(tidyverse)
library(recommenderlab)
data("MovieLense")

# No es posible pasar de realRatingMatrix a dataframe directamente.
# Se extraen las valoraciones de las películas y se almacenan en formato matriz.
# Cada fila de la matriz contiene la información de un usuario y cada columna la
# información de una película.
valoraciones <- as(MovieLense, "matrix")

# Se convierte la matriz a dataframe
valoraciones <- as.data.frame(valoraciones)
valoraciones <- valoraciones %>% rownames_to_column(var = "usuario")

# Datos descriptivos de las películas
atributos <- MovieLenseMeta
```

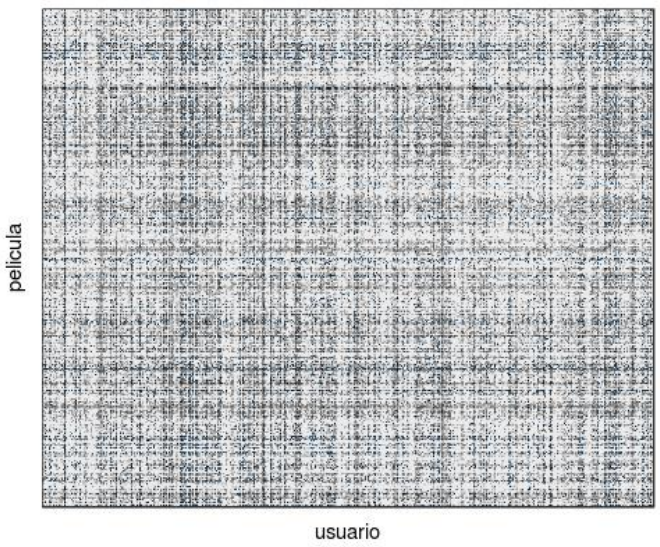
Valoraciones de los usuarios

Visualización de la matriz de datos

```
# Se reestructuran los datos para que tengan un formato tidy
valoraciones_tidy <- valoraciones %>% gather(key = "pelicula",
                                             value = "valoracion",
                                             -usuario)
head(valoraciones_tidy)
```

```
##   usuario         pelicula valoracion
## 1       1 Toy Story (1995)          5
## 2       2 Toy Story (1995)          4
## 3       3 Toy Story (1995)         NA
## 4       4 Toy Story (1995)         NA
## 5       5 Toy Story (1995)          4
## 6       6 Toy Story (1995)          4
```

```
valoraciones_tidy %>% filter(!is.na(valoracion)) %>%
ggplot(aes(x = usuario, y = pelicula, fill = valoracion)) +
  geom_tile(color = "black") +
  theme_bw() +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "none")
```



Porcentaje de valores NA

```
# Se cuenta el número de NA por columna del dataframe y con la función reduce
# se suman todos los resultados. La columna usuario se excluye del contaje.
total_NA <- valoraciones %>% select(-usuario) %>%
            map_dbl(.f = function(x){ sum(is.na(x))}) %>%
            reduce(.f = sum)
total_elementos <- (ncol(valoraciones) - 1) * (nrow(valoraciones))
porcentaje_NA   <- 100 * (total_NA / total_elementos)
porcentaje_NA
```

```
## [1] 93.66588
```

Número de valoraciones por usuario

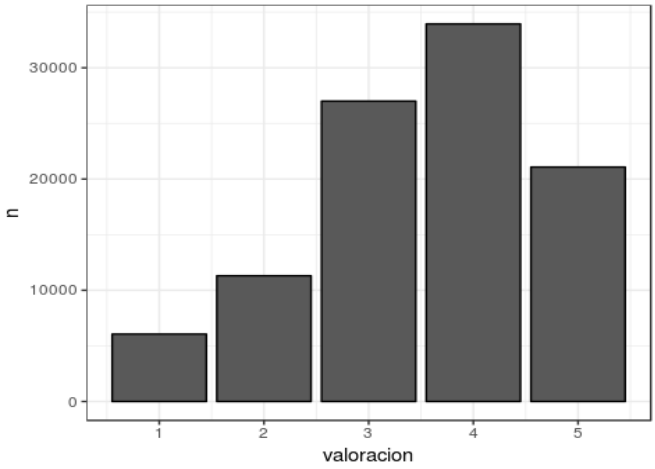
```
valoraciones_tidy %>% filter(!is.na(valoracion)) %>% group_by(usuario) %>%
                      count() %>% pull(n) %>% median()
```

```
## [1] 64
```

El set de datos contiene las valoraciones de 943 usuarios sobre un total de 1664 películas. Sin embargo, hay que tener en cuenta que se trata de una matriz incompleta (94% de valores ausentes), cada película ha sido valorada únicamente por una pequeña fracción de los usuarios. La mediana de valoraciones por usuario es de 64 películas.

Distribución de las valoraciones

```
valoraciones_tidy %>% filter(!is.na(valoracion)) %>% select(valoracion) %>%
group_by(valoracion) %>% count() %>%
  ggplot(aes(x = valoracion, y = n)) + geom_col(color = "black") + theme_bw()
```




```
# Media y mediana de las valoraciones
valoraciones_tidy %>% pull(valoracion) %>% median(na.rm = TRUE)
```

```
## [1] 4
```

```
valoraciones_tidy %>% pull(valoracion) %>% mean(na.rm = TRUE)
```

```
## [1] 3.529982
```

El valor medio y mediana de las valoraciones muestra que los usuarios tienden a valorar positivamente las películas (la media esperada de una distribución uniforme de 1 a 5 es 3).

Estandarización de las valoraciones por usuario

```
valoraciones_tidy <- valoraciones_tidy %>% group_by(usuario) %>%
                     mutate(valoracion = scale(valoracion)) %>% ungroup()
```

Atributos de las películas

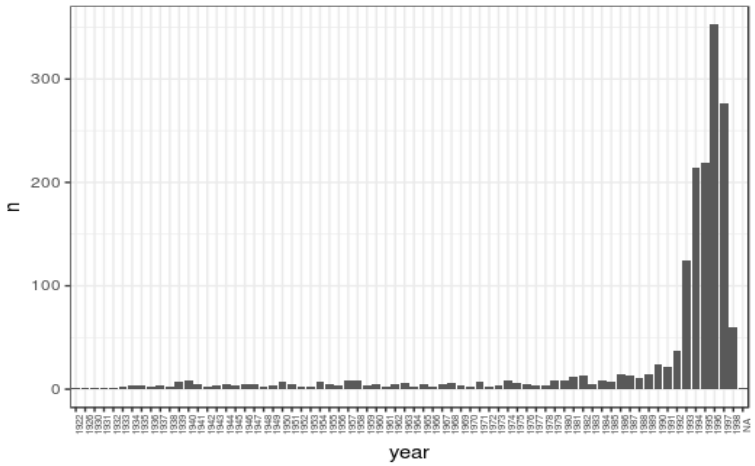
```
glimpse(atributos)
```

```
## Observations: 1,664
## Variables: 22
## $ title       <chr> "Toy Story (1995)", "GoldenEye (1995)", "Four Roo...
## $ year        <dbl> 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1...
## $ url         <chr> "http://us.imdb.com/M/title-exact?Toy%20Story%20(...
## $ unknown     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Action      <int> 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1...
## $ Adventure   <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Animation   <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

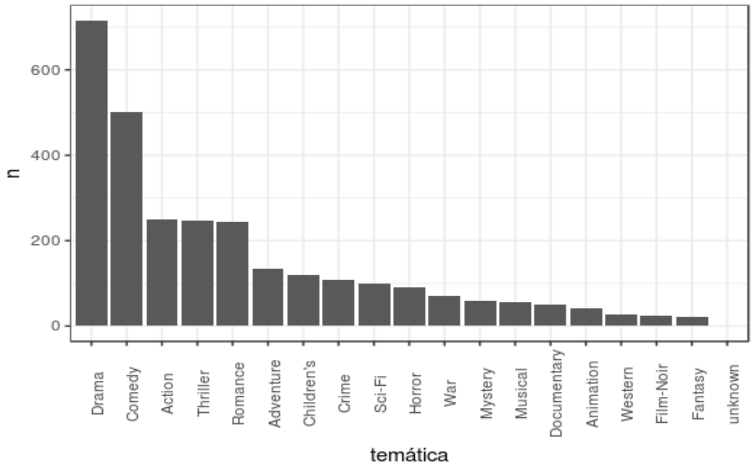
```
## $ `Children's` <int> 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Comedy       <int> 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1...
## $ Crime        <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1...
## $ Documentary  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Drama        <int> 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0...
## $ Fantasy      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ `Film-Noir`  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Horror       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1...
## $ Musical      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Mystery      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Romance      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0...
## $ `Sci-Fi`     <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Thriller     <int> 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1...
## $ War          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0...
## $ Western      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

Entre los atributos descriptivos de cada película se encuentran el título, el año, una dirección web y 19 posibles temáticas.

```
atributos %>% select(year) %>% group_by(year) %>% count() %>%
  ggplot(aes(x =  as.factor(year), y = n)) +
    geom_col() +
    theme_bw() +
    labs(x = "year") +
    theme(axis.text.x = element_text(angle = 90, size = 6))
```




```
atributos %>% select(-title, -url, -year) %>%
  gather(key = "variable", value = "valor") %>%
  filter(valor != 0) %>% group_by(variable) %>%
  count() %>%
  ggplot(aes(x =  reorder(variable, desc(n)), y = n)) + geom_col() + theme_bw() +
 labs(x = "temática") + theme(axis.text.x = element_text(angle = 90))
```



Un análisis sencillo de los datos muestra que la mayoría de las películas disponibles en el set de datos son de 1990 o posteriores, y que las temáticas más frecuentes son drama y comedia. Para los siguientes análisis, las variables *url* y *year* no se emplean, por lo que se excluyen.

```
atributos <- atributos %>% select(-url, -year)
```

Sistema de recomendación basado en contenido

La estrategia seguida para recomendar 10 películas al usuario *329* mediante un sistema basado en contenido es la siguiente:

---

1. Identificar todas las películas que el usuario no ha visto. Se asume que son aquellas para las que el usuario no ha dado su valoración.

2. Para cada una de las *p* películas seleccionadas en el paso 1:
   - Calcular su similitud con las películas vistas por el usuario. En este caso, dado que los atributos son binarios, se emplea como medida de similitud el índice de *Jaccard*.
   - Seleccionar las *n=15* películas más parecidas. En la práctica, el número óptimo de películas debería identificarse mediante validación cruzada, sin embargo, para no añadir una capa de complejidad extra al ejemplo, se emplea este valor.

- Se calcula la media ponderada de las valoraciones que el usuario *329* ha dado de las *n=15* películas más parecidas. Este valor se almacena como el valor predicho para la película *p*.

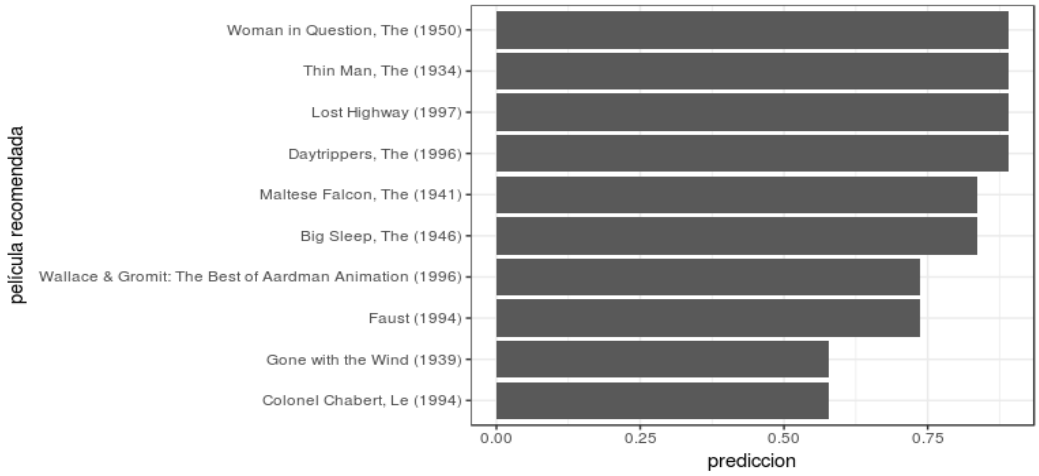
3. Se muestran como recomendaciones las 10 películas con mayor valor predicho.

```
# Identificación de las películas vistas y no vistas por el usuario 329.
# Se asume que si la película no ha sido valorada es que no ha sido vista.
peliculas_vistas <- valoraciones_tidy %>%
                    filter(usuario == 329 & !is.na(valoracion)) %>%
                    pull(pelicula)
peliculas_no_vistas <- valoraciones_tidy %>%
                       filter(usuario == 329 & is.na(valoracion)) %>%
                       pull(pelicula)
# Se calcula la similitud entre cada película no valorada y las sí valoradas.
# Se genera un grid con todas las comparaciones que se tienen que realizar
comparaciones <- expand.grid(peliculas_no_vistas, peliculas_vistas,
                             stringsAsFactors = FALSE)
colnames(comparaciones) <- c("pelicula_no_vista", "pelicula_vista")

# Cuando un cálculo implica múltiples pares de vectores, suele ser práctico
# almacenar los datos en forma de matriz o dataframe donde cada vector es una
# columna. Se crea un dataframe en el que cada columna es una película.
atributos <- atributos %>% gather(key = "atributo", value = "valor", -title) %>%
             spread(key = title, value = valor)

# Se define la función que calcula la similitud
indice_jaccard <- function(pelicula1, pelicula2, datos) {
  # Esta función calcula el índice jaccard entre dos columnas de un dataframe.
  # El valor 1 indica presencia y el valor 0 ausencia.
  m11 <- sum(datos[, pelicula1] == 1 & datos[, pelicula2] == 1)
  m10 <- sum(datos[, pelicula1] == 1 & datos[, pelicula2] == 0)
  m01 <- sum(datos[, pelicula1] == 0 & datos[, pelicula2] == 1)
  indice <- m11 / sum(m01 + m10 + m11)
  return(indice)
}

# Con la función map2 del paquete purrr, se aplica la función indice_jaccard
# empleando las columnas del grid comparaciones como valores de los argumentos
# pelicula1 y pelicula2.
recomendaciones <- comparaciones[] %>%
                   mutate(similitud = map2_dbl(.x = pelicula_no_vista,
                                               .y = pelicula_vista,
                                               .f = indice_jaccard,
                                                datos = atributos))

# Para cada película no vista, se filtran las 15 películas más parecidas
```

```
recomendaciones <- recomendaciones %>% group_by(pelicula_no_vista) %>%
                                       top_n(n = 15, wt = similitud) %>%
                                       arrange(pelicula_no_vista, desc(similitud))

# Se añade la valoración que el usuario 329 ha hecho de cada una de las películas
valoraciones_u329 <- valoraciones_tidy %>%
                     filter(usuario == 329 & !is.na(valoracion))
recomendaciones <- recomendaciones %>%
                   left_join(y = valoraciones_u329,
                             by = c("pelicula_vista"  = "pelicula"))

# Media ponderada de las valoraciones por película
media_ponderada <- function(df){
  resultado <- sum(df$valoracion * df$similitud) / sum(df$similitud)
  return(resultado)
}
top10_recomendaciones <- recomendaciones %>% group_by(pelicula_no_vista) %>%
                         nest() %>%
                         mutate(prediccion = map_dbl(.x = data,
                                                     .f = media_ponderada)) %>%
                         select(-data) %>% arrange(desc(prediccion)) %>% head(10)
top10_recomendaciones
```

```
## # A tibble: 10 x 2
##    pelicula_no_vista                                          prediccion
##    <chr>                                                           <dbl>
##  1 Daytrippers, The (1996)                                         0.890
##  2 Lost Highway (1997)                                             0.890
##  3 Thin Man, The (1934)                                            0.890
##  4 Woman in Question, The (1950)                                   0.890
##  5 Big Sleep, The (1946)                                           0.836
##  6 Maltese Falcon, The (1941)                                      0.836
##  7 Faust (1994)                                                    0.736
##  8 Wallace & Gromit: The Best of Aardman Animation (1996)          0.736
##  9 Colonel Chabert, Le (1994)                                      0.578
## 10 Gone with the Wind (1939)                                       0.578
```

```
ggplot(data = top10_recomendaciones,
       aes(x = reorder(pelicula_no_vista, prediccion), y = prediccion)) +
  geom_col() +
  coord_flip() +
  labs(x = "película recomendada") +
  theme_bw()
```



Es importante recordar que las valoraciones han sido estandarizadas y por lo tanto, también lo están las predicciones.

Filtrado colaborativo basado en usuarios

La estrategia seguida para recomendar 10 películas al usuario *329* mediante un sistema colaborativo basado en usuarios es la siguiente:

---

1. Calcular la similitud entre el usuario *329* y el resto de usuarios en base a sus perfiles de valoración, es decir, utilizando los vectores formados por sus valoraciones. Para este ejemplo se emplea la correlación de Pearson como medida de similitud. *nota 1*

2. Identificar todas las películas que el usuario *329* no ha visto. Se asume que son aquellas para las que el usuario *329* no ha dado su valoración.

3. Para cada una de las *p* películas seleccionadas en el paso 2:
   - Seleccionar los *n=15* usuarios más parecidos al usuario *329*, cuyo valor de similitud es positivo *nota 2* *nota 3*, y que sí han visto la película *p*. En la práctica, el número óptimo de usuarios debería identificarse mediante validación cruzada, sin embargo, para no añadir una capa de complejidad extra al ejemplo, se emplea este valor.
   - Se calcula la media ponderada de las valoraciones que los *n=15* usuarios han dado de la película. Este valor se almacena como el valor predicho para la película *p*.

4. Se muestran como recomendaciones las 10 películas con mayor valor predicho.

Nota1: En el paso 1 del algoritmo, se calcula la similitud entre usuarios. Para que esta estimación sea mínimamente realista, conviene incluir únicamente aquellos usuarios que hayan valorado un mínimo de películas. El valor límite se determina en función de los datos disponibles y de la robustez que se necesite en las estimaciones.

Nota2: Dado que se emplea la media ponderada como estimación final, no se pueden incluir pesos negativos. Como la correlación de Pearson toma valores en el rango [-1, +1], se emplean únicamente aquellas observaciones con valores mayores o iguales a cero. A efectos prácticos, equivale a decir que no se tienen en cuenta las valoraciones de los usuarios que tienen un perfil opuesto.

Nota3: Aunque se establezca que se tienen que emplear los n usuarios más similares para predecir la valoración, puede ocurrir que, para algunas películas, no haya suficientes usuarios que las hayan valorado. Es conveniente recomendar únicamente películas cuya predicción esté basada en un mínimo de usuarios, de lo contrario la estimación puede ser muy mala.

Se consideran únicamente aquellos usuarios que han valorado al menos 30 películas.

```
usuarios_excluidos <- valoraciones_tidy %>% filter(!is.na(valoracion)) %>%
                      group_by(usuario) %>% count() %>% filter(n < 30) %>%
                      pull(usuario)
valoraciones_tidy <- valoraciones_tidy %>% filter(!usuario %in% usuarios_excluidos)

# Se crea un dataframe en el que cada columna representa las valoraciones de
# un usuario.
valoraciones_usuarios <- valoraciones_tidy %>%
                         spread(key = usuario, value = valoracion, fill = NA)

# Función que calcula la similitud entre dos columnas
funcion_correlacion <- function(x, y){
  correlacion <- cor(x, y, use = "na.or.complete", method = "pearson")
  return(correlacion)
}

# Se aplica la función de correlación a cada columna de valoraciones_usuarios,
# empelando como argumento "y" la columna del usuario "329"
similitud_usuarios <- map_dbl(.x = valoraciones_usuarios[, -1],
                              .f = funcion_correlacion,
                              y = valoraciones_usuarios[, "329"])
similitud_usuarios <- data_frame(usuario   = names(similitud_usuarios),
                                 similitud = similitud_usuarios) %>%
                      arrange(desc(similitud))
head(similitud_usuarios)
```

```
## # A tibble: 6 x 2
##   usuario similitud
##   <chr>       <dbl>
## 1 329         1.00
## 2 861         1.00
## 3 544         0.969
## 4 122         0.968
## 5 338         0.928
## 6 306         0.927
```

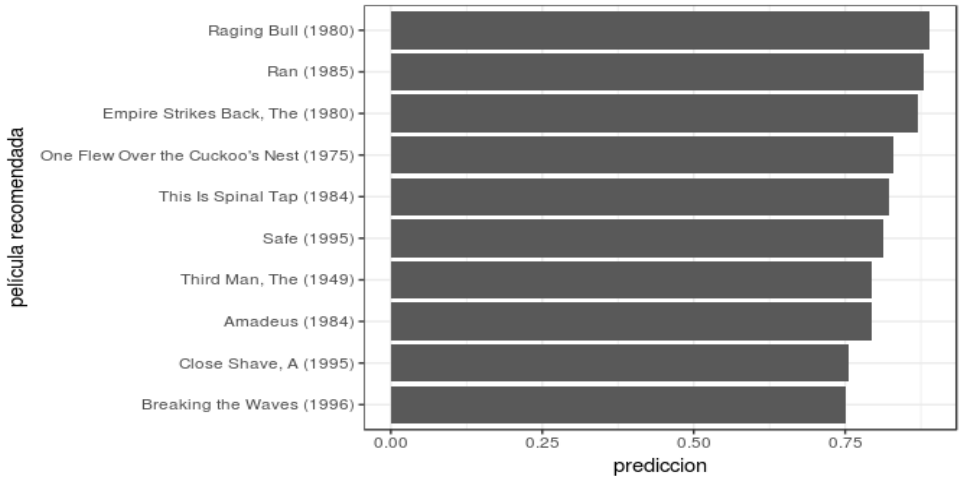
```r
# Identificación de las películas vistas y no vistas por el usuario 329.
# Se asume que si la película no ha sido valorada por el usuario 329 es que no
# ha sido vista.
peliculas_vistas <- valoraciones_tidy %>%
                    filter(usuario == 329 & !is.na(valoracion)) %>%
                    pull(pelicula)
peliculas_no_vistas <- valoraciones_tidy %>%
                       filter(usuario == 329 & is.na(valoracion)) %>%
                       pull(pelicula)

# Se inicia un bucle para predecir la valoración que el usuario 329 hará de cada
# una de las películas no vistas.
prediccion <- rep(NA, length(peliculas_no_vistas))
pelicula   <- rep(NA, length(peliculas_no_vistas))
n_obs_prediccion <- rep(NA, length(peliculas_no_vistas))

for(i in seq_along(peliculas_no_vistas)){
  # Usuarios que han visto la película i
  usuarios_pelicula_i <- valoraciones_tidy %>%
                         filter(pelicula == peliculas_no_vistas[i] &
                                !is.na(valoracion)) %>% pull(usuario)


  # Si no hay un mínimo de usuarios que han visto la película, no se considera una
  # estimación suficientemente buena por lo que se pasa a la siguiente película.
  if (length(usuarios_pelicula_i) < 10){
    next()
  }
  # Los 15 usuarios más parecidos de entre los que han visto la película i, cuya
  # similitud es >= 0.
  top_15_usuarios <- similitud_usuarios %>%
                     filter(similitud>=0 & (usuario %in% usuarios_pelicula_i)) %>%
                     arrange(desc(similitud)) %>%
                     head(15)
  # Si no hay un mínimo de usuarios con valoraciones válidas, no se considera una
  # estimación suficientemente buena por lo que se pasa a la siguiente película.
  if (nrow(top_15_usuarios) < 10){
    next()
  }
```

```
  # Valoraciones de esos 15 usuarios sobre la película i
  valoraciones_top_15 <- valoraciones_tidy %>%
                         filter(pelicula == peliculas_no_vistas[i] &
                                usuario %in% top_15_usuarios$usuario)

  # Media ponderada de las valoraciones de los top_15_usuarios
  top_15_usuarios <- top_15_usuarios %>% left_join(valoraciones_top_15,
                                                   by = "usuario")
  prediccion[i] <- sum(top_15_usuarios$similitud * top_15_usuarios$valoracion) /
                   sum(top_15_usuarios$similitud)
  pelicula[i] <- peliculas_no_vistas[i]
  n_obs_prediccion[i] <- nrow(top_15_usuarios)
}

top10_recomendaciones <- data.frame(pelicula, prediccion, n_obs_prediccion) %>%
                         arrange(desc(prediccion)) %>%
                         head(10)
top10_recomendaciones
```

```
##                                 pelicula prediccion n_obs_prediccion
## 1                     Raging Bull (1980)  0.8892095               15
## 2                           Ran (1985)  0.8797687               15
## 3         Empire Strikes Back, The (1980)  0.8711033               15
## 4  One Flew Over the Cuckoo's Nest (1975)  0.8301386               15
## 5               This Is Spinal Tap (1984)  0.8228707               15
## 6                            Safe (1995)  0.8122988               10
## 7                   Third Man, The (1949)  0.7935308               15
## 8                         Amadeus (1984)  0.7928368               15
## 9                   Close Shave, A (1995)  0.7551640               15
## 10              Breaking the Waves (1996)  0.7512220               15
```

La columna *n_obs_prediccion* contiene el número de usuarios que se han empleado para estimar la valoración de la película. Es importante tenerlo en cuenta ya que, aunque por defecto son 15, puede ocurrir que para algunas películas no haya tantos usuarios que las hayan valorado.

```
ggplot(data = top10_recomendaciones,
       aes(x = reorder(pelicula, prediccion), y = prediccion)) +
  geom_col() +
  coord_flip() +
  labs(x = "película recomendada") +
  theme_bw()
```



Filtrado colaborativo basado en ítems

La estrategia seguida para recomendar 10 películas al usuario *329* mediante un sistema colaborativo basado en ítems es la siguiente:

---

1. Identificar todas las películas que el usuario *329* no ha visto. Se asume que son aquellas para las que el usuario *329* no ha dado su valoración.

2. Para cada una de las *p* películas seleccionadas en el paso 1:
   - Calcular la similitud con las películas que el usuario *329* sí ha visto, en base al perfil de valoración que han recibido, es decir, utilizando los vectores formados por sus valoraciones. Para este ejemplo se emplea la correlación de Pearson como medida de similitud. *nota 1*
   - Seleccionar los *n=15* películas más parecidas. En la práctica, el número óptimo de películas debería identificarse mediante validación cruzada, sin embargo, para no añadir una capa de complejidad extra al ejemplo, se emplea este valor. *nota 2*
   - Se calcula la media ponderada de las valoraciones que el usuario *329* ha hecho de las *n=15* películas más parecidas. Este valor se almacena como el valor predicho para la película *p*. *nota 3*

3. Se muestran como sugerencias las 10 películas con mayor valor de predicción.

Nota1: En el paso 1 del algoritmo, se calcula la similitud entre películas. Para que esta estimación sea mínimamente realista, conviene incluir únicamente aquellas películas que hayan sido valoradas por un mínimo de usuarios. El valor límite se determina en función de los datos disponibles y de la robustez que se necesite en las estimaciones.

Nota2: Dado que se emplea la media ponderada como estimación final, no se pueden incluir pesos negativos. Como la correlación de Pearson toma valores en el rango [-1, +1], se emplean únicamente aquellas observaciones con valores mayores o iguales a cero. A efectos prácticos, equivale a decir que no se tienen en cuenta las valoraciones de los usuarios que tienen un perfil opuesto.

Nota3: Aunque se establezca que se tienen que emplear las n películas más similares para predecir la valoración, puede ocurrir que no haya suficientes. Es conveniente hacer recomendaciones basadas en un mínimo de observaciones, de lo contrario la estimación puede ser muy mala.

Para que el cálculo de similitudes entre películas sea válido, se emplean únicamente películas que hayan recibido un mínimo de 10 valoraciones.

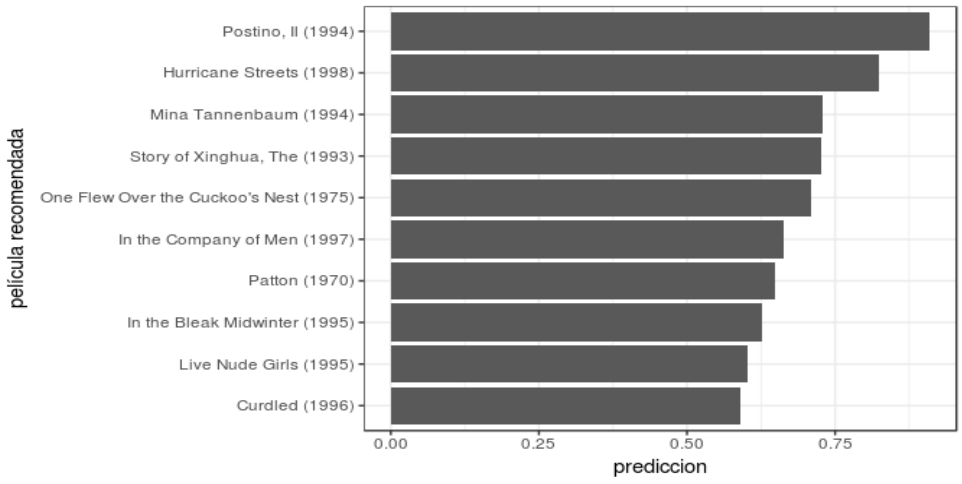
```
valoraciones_tidy <- valoraciones %>% gather(key = "pelicula",
                                             value = "valoracion",
                                             -usuario) %>%
                     group_by(usuario) %>%
                     mutate(valoracion = scale(valoracion)) %>%
                     ungroup()
peliculas_excluidas <- valoraciones_tidy %>% filter(!is.na(valoracion)) %>%
                       group_by(pelicula) %>% count() %>% filter(n < 5) %>%
                       pull(pelicula)
valoraciones_tidy <- valoraciones_tidy %>%
                     filter(!pelicula %in% peliculas_excluidas)

# Identificación de las películas vistas y no vistas por el usuario 329.
# Se asume que si la película no ha sido valorada es que no ha sido vista.
peliculas_vistas <- valoraciones_tidy %>%
                    filter(usuario == 329 & !is.na(valoracion)) %>%
                    pull(pelicula)

peliculas_no_vistas <- valoraciones_tidy %>%
                       filter(usuario == 329 & is.na(valoracion)) %>%
                       pull(pelicula)

# Se genera un grid con todas las comparaciones que se tienen que realizar
comparaciones <- expand.grid(peliculas_no_vistas, peliculas_vistas,
                             stringsAsFactors = FALSE)
colnames(comparaciones) <- c("pelicula_no_vista", "pelicula_vista")
```

```
# Se crea un dataframe en el que cada columna es una película
valoraciones <- valoraciones_tidy %>%
                      spread(key = pelicula, value = valoracion, fill = NA)

# Se define la función que calcula la similitud
correlacion <- function(pelicula1, pelicula2, datos) {
  # Esta función calcula la correlación entre dos columnas de un dataframe.
  similitud <- cor(x = datos[, pelicula1], y = datos[, pelicula2],
                   method = "pearson", use = "na.or.complete")
  return(similitud)
}

# Con la función map2 del paquete purrr, se aplica la función correlación empleando
# las columnas del grid comparaciones como valores de los argumentos pelicula1 y
# pelicula2.
comparaciones <- comparaciones %>%
                 mutate(similitud = map2_dbl(.x = pelicula_no_vista,
                                             .y = pelicula_vista,
                                             .f = correlacion,
                                             datos = valoraciones))

# Para cada película no vista, se filtran las 15 películas más parecidas y cuyo
# valor de similitud es mayor o igual a cero.
comparaciones <- comparaciones %>% filter(similitud >= 0) %>%
                 group_by(pelicula_no_vista) %>%
                 top_n(n = 15, wt = similitud) %>%
                 arrange(pelicula_no_vista, desc(similitud))

# Se eliminan aquellas películas para las que no haya un mínimo de películas
# similares con valores positivos.
exclusion <- comparaciones %>%
             group_by(pelicula_no_vista) %>%
             count() %>%
             filter(n < 10) %>%
             pull(pelicula_no_vista)
comparaciones <- comparaciones %>% filter(!pelicula_no_vista %in% exclusion)

# Se añade la valoración que el usuario 329 ha hecho de cada una de las películas.
valoraciones_u329 <- valoraciones_tidy %>%
                     filter(usuario == 329 & !is.na(valoracion))
comparaciones <-  comparaciones %>%
                  left_join(y = valoraciones_u329,
                            by = c("pelicula_vista"  = "pelicula"))

# Media ponderada de las valoraciones por película
media_ponderada <- function(df){
  resultado <- sum(df$valoracion * df$similitud) / sum(df$similitud)
  return(resultado)
}
```

```
top10_recomendaciones <- comparaciones %>% group_by(pelicula_no_vista) %>%
                         nest() %>%
                         mutate(prediccion = map_dbl(.x = data,
                                                     .f = media_ponderada)) %>%
                         select(-data) %>% arrange(desc(prediccion)) %>% head(10)
top10_recomendaciones
```

```
## # A tibble: 10 x 2
##    pelicula_no_vista                          prediccion
##    <chr>                                           <dbl>
##  1 Postino, Il (1994)                              0.909
##  2 Hurricane Streets (1998)                        0.824
##  3 Mina Tannenbaum (1994)                          0.729
##  4 Story of Xinghua, The (1993)                    0.726
##  5 One Flew Over the Cuckoo's Nest (1975)          0.710
##  6 In the Company of Men (1997)                    0.663
##  7 Patton (1970)                                   0.649
##  8 In the Bleak Midwinter (1995)                   0.627
##  9 Live Nude Girls (1995)                          0.602
## 10 Curdled (1996)                                  0.591
```

```
ggplot(data = top10_recomendaciones,
       aes(x = reorder(pelicula_no_vista, prediccion), y = prediccion)) +
  geom_col() +
  coord_flip() +
  labs(x = "película recomendada") +
  theme_bw()
```



Bibliografía

recommenderlab: A Framework for Developing and Testing Recommendation Algorithms by Michael Hahsler

Introduction to Data Science A Python Approach to Concepts, Techniques and Applications by by Laura Igual, Santi Seguí