TP 2: Modelos mixtos, splines penalizados y causalidad

Florencia Fontana Walser, Jerónimo Barragán, Lorenzo Gandolfo

2025-07-09

```
library(dplyr)
library(tidyr)
library(stringr)
library(ggplot2)
library(knitr)
library(lme4)
library(ggrepel)
library(viridis)
library(mgcv)
library(caret)
library(ranger)
library(xgboost)
```

1. Análisis exploratorio

```
titles  <- read.csv("titles_train.csv",  stringsAsFactors = FALSE)
credits <- read.csv("credits_train.csv", stringsAsFactors = FALSE)

dim(titles)
dim(credits)

str(titles)
str(credits)
```

```
colSums(is.na(titles))
```

```
##                    X                   id                title
##                    0                    0                    0
##                 type          description         release_year
##                    0                    0                    0
##    age_certification              runtime               genres
##                    0                    0                    0
## production_countries              seasons              imdb_id
##                    0                 2619                    0
##           imdb_score          imdb_votes
##                  369                  382
```

```r
colSums(is.na(credits))
```

```
##         X person_id        id      name character      role
##         0         0         0         0         0         0
```

```r
sort(table(titles$production_countries), decreasing = TRUE) %>% head(10)
```

```
##
## ['US'] ['IN'] ['JP']     [] ['KR'] ['GB'] ['ES'] ['FR'] ['MX'] ['CA']
##   1358    419    186    156    144    138    110     86     69     67
```

```r
table(credits$role)
```

```
##
##    ACTOR DIRECTOR
##    50476     3167
```

Vemos que hay nulos en imdb_score y imdb_votes. Tomamos la decisión de eliminar estas filas ya que el porcentaje de faltantes no es muy alto. Además notar que existe una alta concentración de títulos en EE.UU. e India (quizás es bueno tener en cuenta esto a la hora de modelar efectos de país).

```r
titles <- titles %>%
  drop_na(imdb_score, imdb_votes)
```

```r
summary(titles$imdb_score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.600   5.800   6.600   6.532   7.400   9.600
```

```r
summary(titles$imdb_votes)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##       5.0     525.2    2287.0   25023.6   10005.0 2268288.0
```

```r
summary(titles$runtime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   46.00   88.00   79.86  106.00  235.00
```

```r
summary(titles$release_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1953    2015    2018    2016    2020    2022
```

```r
titles$comedia <- grepl("comedy", titles$genres)
prop.table(table(titles$comedia))
```

```
##
##     FALSE      TRUE
## 0.5870647 0.4129353
```

El 41% de la muestra tiene incluido el género comedia.

```
credits %>%
  filter(role == "ACTOR") %>%
  count(name, sort = TRUE) %>%
  head(10)
```

```
##                     name  n
## 1         Shah Rukh Khan 19
## 2            Anupam Kher 17
## 3            Boman Irani 17
## 4    Kareena Kapoor Khan 16
## 5       Naseeruddin Shah 16
## 6       Takahiro Sakurai 15
## 7       Amitabh Bachchan 14
## 8        Fred Tatasciore 14
## 9          Kenjiro Tsuda 14
## 10 Priyanka Chopra Jonas 14
```

```
credits %>%
  filter(role == "ACTOR") %>%
  count(name, sort = TRUE) %>%
  slice_head(n = 10)
```

```
##                     name  n
## 1         Shah Rukh Khan 19
## 2            Anupam Kher 17
## 3            Boman Irani 17
## 4    Kareena Kapoor Khan 16
## 5       Naseeruddin Shah 16
## 6       Takahiro Sakurai 15
## 7       Amitabh Bachchan 14
## 8        Fred Tatasciore 14
## 9          Kenjiro Tsuda 14
## 10 Priyanka Chopra Jonas 14
```

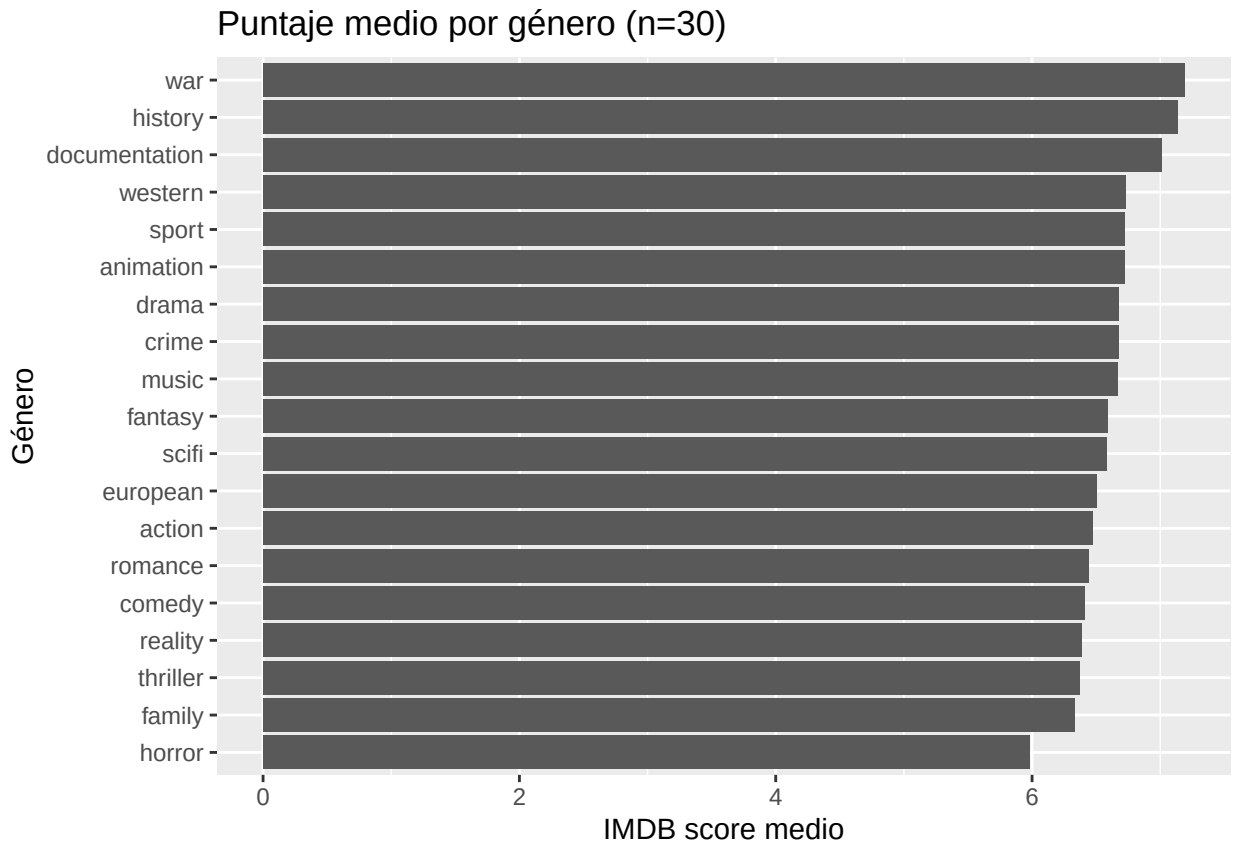
Bien, respondamos algunas preguntas que sugiere el enunciado.

¿Hay algún género que parezca estar más asociado con el puntaje del título?

```
genres_long <- titles %>%
  mutate(
    genres = str_remove_all(genres, "\\[|\\]|'")
  ) %>%
  separate_rows(genres, sep = ",\\s*") %>%
  mutate(genres = str_trim(genres))
```

```r
genre_stats <- genres_long %>%
  group_by(genres) %>%
  summarise(
    n           = n(),
    mean_score  = mean(imdb_score),
    sd_score    = sd(imdb_score)
  ) %>%
  filter(n >= 30) %>%       # umbral para tener suficiente muestra
  arrange(desc(mean_score))
```

```r
ggplot(genre_stats, aes(x = reorder(genres, mean_score), y = mean_score)) +
  geom_col() +
  coord_flip() +
  labs(title="Puntaje medio por género (n 30)",
       x="Género", y="IMDB score medio")
```



Las películas de guerra, historia y documentales suelen recibir las mejores calificaciones, mientras que el horror, el cine familiar y los thrillers tienden a quedar por debajo del promedio.

¿Hay algún actor o director asociado con mayores o menores puntajes?

```r
cred_scores <- credits %>%
  inner_join(titles, by="id")
```

```r
person_stats <- cred_scores %>%
  group_by(name, role) %>%
  summarise(
    n_titles   = n(),
    mean_score = mean(imdb_score),
    sd_score   = sd(imdb_score),
    .groups    = "drop"
  ) %>%
  filter(n_titles >= 5)

top_actors <- person_stats %>%
  filter(role == "ACTOR") %>%
  arrange(desc(mean_score)) %>%
  slice_head(n = 10)

bottom_actors <- person_stats %>%
  filter(role == "ACTOR") %>%
  arrange(mean_score) %>%
  slice_head(n = 10)
kable(top_actors,    caption = "Top 10 actores (>=5 títulos) por IMDb score medio")
```

Table 1: Top 10 actores (>=5 títulos) por IMDb score medio

name	role	n_titles	mean_score	sd_score
Yui Ishikawa	ACTOR	5	8.240000	0.5770615
Terry Jones	ACTOR	7	8.071429	0.1380131
Eric Idle	ACTOR	6	8.050000	0.1378405
Graham Chapman	ACTOR	6	8.050000	0.1378405
Terry Gilliam	ACTOR	5	8.040000	0.1516575
Yoshimasa Hosoya	ACTOR	7	8.000000	0.5477226
Michael Palin	ACTOR	7	7.914286	0.3804759
Yeom Hye-ran	ACTOR	5	7.900000	0.2449490
Tom Hanks	ACTOR	5	7.880000	0.8467585
R. Madhavan	ACTOR	7	7.857143	0.8997354

```r
kable(bottom_actors, caption = "Top 10 actores con menor score medio")
```

Table 2: Top 10 actores con menor score medio

name	role	n_titles	mean_score	sd_score
Blossom Chukwujekwu	ACTOR	6	4.750000	0.8871302
Joke Silva	ACTOR	5	4.780000	1.2316655
Bruce Willis	ACTOR	7	4.842857	1.6308047
Mike Epps	ACTOR	6	4.866667	1.4165686
Johnny Lever	ACTOR	13	4.869231	1.4665064
Prem Chopra	ACTOR	5	4.940000	1.7169741
Molly Ringwald	ACTOR	5	4.960000	1.1738824
Mushtaq Khan	ACTOR	5	5.000000	1.2767145
Jacqueline Fernandez	ACTOR	5	5.060000	1.0597169
Neil Nitin Mukesh	ACTOR	5	5.140000	0.8294577

```r
top_dirs <- person_stats %>%
  filter(role == "DIRECTOR") %>%
  arrange(desc(mean_score)) %>%
  slice_head(n = 10)

bottom_dirs <- person_stats %>%
  filter(role == "DIRECTOR") %>%
  arrange(mean_score) %>%
  slice_head(n = 10)

kable(top_dirs,    caption = "Top 10 directores (>=5 títulos) por IMDb score medio")
```

Table 3: Top 10 directores (>=5 títulos) por IMDb score medio

name	role	n_titles	mean_score	sd_score
Hong Jong-chan	DIRECTOR	5	7.900000	0.2449490
Ryan Polito	DIRECTOR	5	7.520000	0.8983318
Shannon Hartman	DIRECTOR	7	7.242857	0.4928054
Youssef Chahine	DIRECTOR	7	7.128571	0.4029652
Cathy Garcia-Molina	DIRECTOR	10	6.950000	0.3749074
Jay Karas	DIRECTOR	13	6.946154	1.0063261
Vishal Bhardwaj	DIRECTOR	5	6.940000	0.9710819
Jay Chapman	DIRECTOR	9	6.722222	0.7412452
Ashutosh Gowariker	DIRECTOR	6	6.583333	1.4824529
Lance Bangs	DIRECTOR	5	6.580000	0.5118594

```r
kable(bottom_dirs, caption = "Top 10 directores con menor score medio")
```

Table 4: Top 10 directores con menor score medio

name	role	n_titles	mean_score	sd_score
Justin G. Dyck	DIRECTOR	5	4.800000	0.8215838
Kunle Afolayan	DIRECTOR	5	4.960000	1.1886968
Umesh Mehra	DIRECTOR	7	5.242857	0.9396048
Steven Brill	DIRECTOR	6	5.900000	0.8485281
Hiroyuki Seshita	DIRECTOR	5	6.200000	0.9695360
Jan Suter	DIRECTOR	14	6.335714	0.7761910
Raúl Campos	DIRECTOR	14	6.335714	0.7761910
Marcus Raboy	DIRECTOR	12	6.458333	0.6258933
Lance Bangs	DIRECTOR	5	6.580000	0.5118594
Troy Miller	DIRECTOR	5	6.580000	0.8671793

Definitivamente hay actores y directores asociados a puntajes más altos y otros cuyo historial de títulos tiende a puntuar más bajo.

2. Predecir puntaje de IMDb en función del país de origen

Como hay películas que tienen más de un país de origen, las sacamos.

```r
titles_sc <- titles %>%
  mutate(tmp = str_remove_all(production_countries, "\\[|\\]|'"),
         tmp = str_trim(tmp),
         pcs = str_split(tmp, ",\\s*")) %>%
  filter(lengths(pcs) == 1) %>%
  unnest(pcs) %>%
  rename(country = pcs) %>%
  mutate(country = factor(country))
```

a)

```r
lm_fe_lm <- lm(imdb_score ~ country, data = titles_sc)
fe_coefs_lm <- coef(lm_fe_lm)


fixed_countries <- sub("^country", "", names(fe_coefs_lm)[-1])
fixed_vals      <- fe_coefs_lm[-1]
```

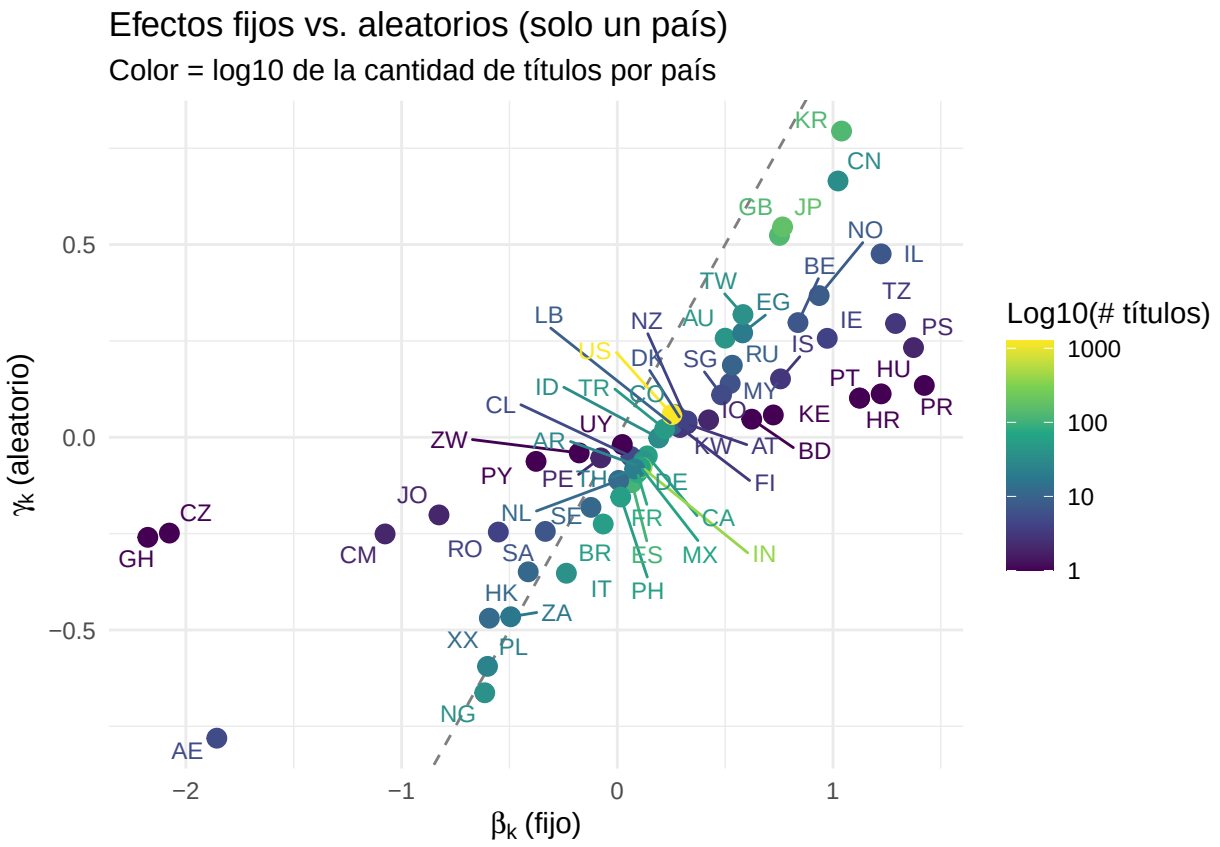
b)

```r
re_mod2   <- lmer(imdb_score ~ 1 + (1 | country), data = titles_sc)
re_df2    <- ranef(re_mod2)$country
re_coefs2 <- re_df2[ , "(Intercept)"]
names(re_coefs2) <- rownames(re_df2)
```

c)

```r
common2 <- intersect(fixed_countries, names(re_coefs2))

effects2 <- tibble(
  country = common2,
  fixed   = fixed_vals[paste0("country", country)],
  random  = re_coefs2[country]
)

counts <- titles_sc %>% count(country) %>% rename(n = n)
effects2 <- effects2 %>% left_join(counts, by = "country")

ggplot(effects2, aes(x = fixed, y = random, label = country, color = n)) +
  geom_abline(slope = 1, intercept = 0, linetype="dashed", color="gray50") +
  geom_point(size = 3) +
  geom_text_repel(size = 3, max.overlaps = Inf) +
  scale_color_viridis_c(trans="log10", name="Log10(# títulos)") +
  labs(
    title    = "Efectos fijos vs. aleatorios (solo un país)",
    subtitle = "Color = log10 de la cantidad de títulos por país",
    x        = expression(beta[k]~"(fijo)"),
```

```
    y        = expression(gamma[k]~"(aleatorio)")
  ) +
  theme_minimal()
```



En nuestro diagrama de **(efecto fijo)** vs. **(efecto aleatorio)**, con el color indicando el `log` del número de títulos por país, se ven tres comportamientos clave:

1. **Países con muy pocos títulos** (violeta oscuro): sus están muy dispersos a lo largo del eje X (efectos fijos extremos, positivos o negativos). Sin embargo, sus (efectos aleatorios) aparecen más concentrados en el eje Y. Esto ilustra el *shrinkage*: al tener poca información, el modelo mixto "aplasta" esas estimaciones hacia la media global.

2. **Países de rango medio de títulos** (tonos intermedios de violeta y verde): muestran **menos variabilidad** en que los de conteo bajo, y sin embargo **un poco más** de dispersión en . Su shrinkage es más moderado: el modelo aleatorio corrige algo, pero no tanto como en el caso de observaciones escasas.

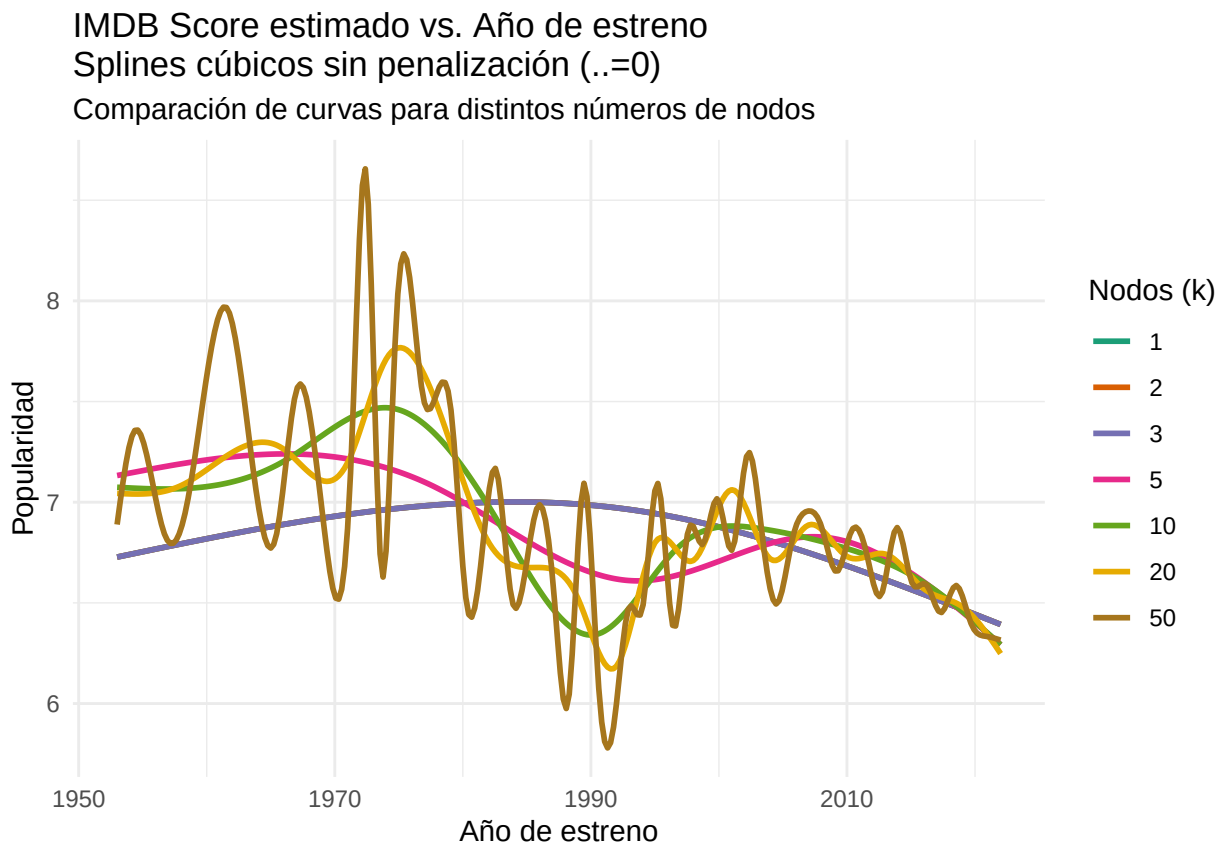
3. **Países con muchos títulos** (amarillo claro, p. ej. US, IN): sus se agrupan en una zona relativamente estrecha del eje X.

Cuanto menos datos tiene un país, más confiamos en la media global (shrinkage fuerte) y menos en su estimación puntual (extrema). A medida que aumenta el volumen de películas, el modelo mixto se comporta cada vez más como el modelo fijo, validando el uso de efectos aleatorios para regularizar estimaciones en niveles con poca información.

3. Popularidad en base a `release_year`

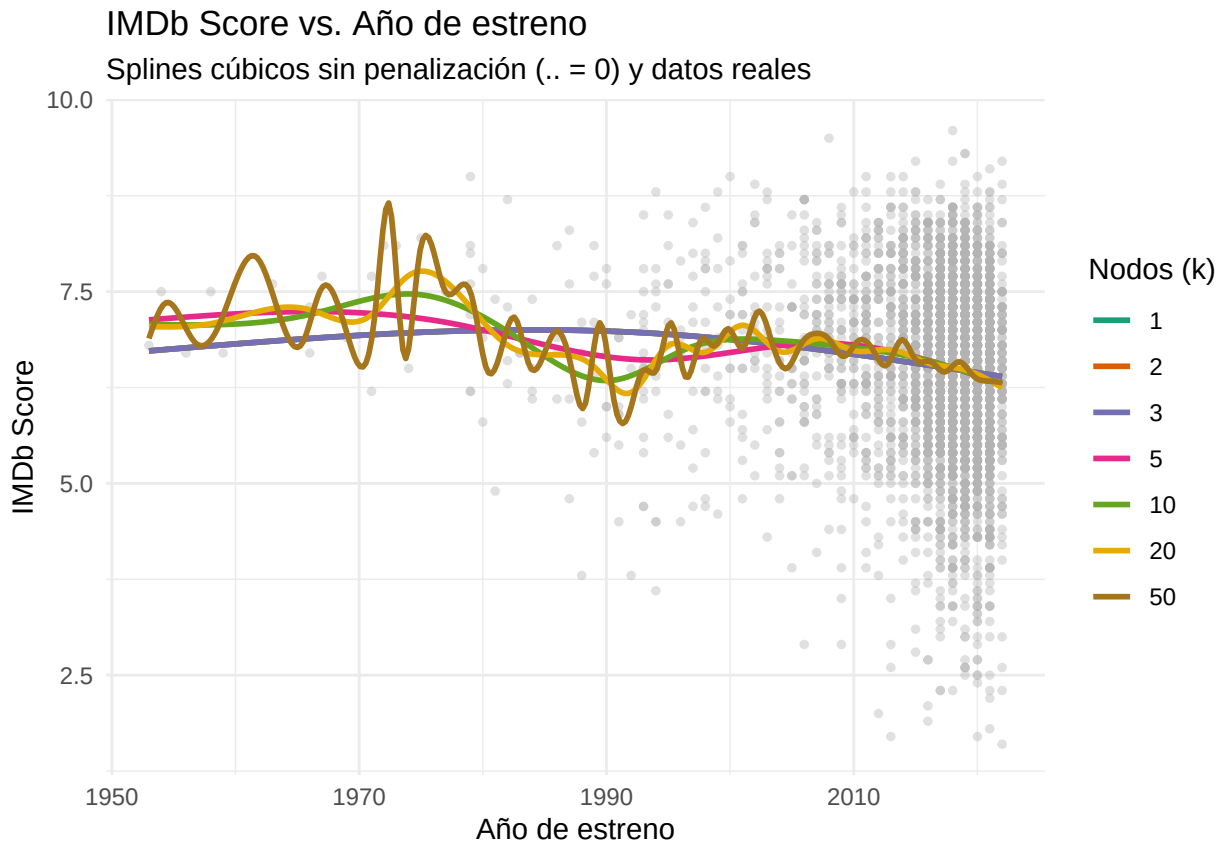
Para trazar las curvas emplearemos un modelo aditivo generalizado (GAM) con splines cúbicos de regresión (`bs = "cr"`) y penalización de rugosidad nula (`= 0`) para capturar de forma flexible la relación entre el año de estreno (`release_year`) y la calificación de IMDb (`imdb_score`). Al variar el número de nodos (`k = 1, 2, 3, 5, 10, 20, 50`), podremos comparar visualmente cómo cambia la forma de la tendencia a lo largo del tiempo según aumentemos la complejidad del spline, sin incorporar suavizado adicional. Esto nos permitirá identificar el nivel de detalle adecuado para describir la evolución de los puntajes sin sobreajustar el modelo.

```r
ks <- c(1, 2, 3, 5, 10, 20, 50)
anios   <- seq(min(titles$release_year),
               max(titles$release_year),
               length.out = 300)

newdat <- data.frame(release_year = anios)

# ajustar un GAM por cada k y sacar la curva
all_preds <- lapply(ks, function(k) {
  fit <- gam(imdb_score ~ s(release_year, bs = "cr", k = k, sp = 0),
             data = titles)
  data.frame(
    release_year = anios,
    imdb_score    = predict(fit, newdata = newdat),
    k             = factor(k, levels = ks)
  )
}) %>%
  do.call(what = rbind)

ggplot(all_preds, aes(x = release_year, y = imdb_score, color = k)) +
  geom_line(size = 1) +
  scale_color_brewer("Nodos (k)", palette = "Dark2") +
  labs(
    title    = "IMDB Score estimado vs. Año de estreno\nSplines cúbicos sin penalización ( =0)",
    x        = "Año de estreno",
    y        = "Popularidad",
    subtitle = "Comparación de curvas para distintos números de nodos"
  ) +
  theme_minimal()
```



Con muy pocos nodos (k=1–3) las curvas quedan demasiado rígidas. Con muchísimos nodos (k=20, 50) y = 0 (no hay ninguna penalización de rugosidad) vemos una oscilación extrema. Un número intermedio de nodos (por ejemplo, k=5 o k=10) parecería ofrecer un trade-off. Agreguemos los datos reales al gráfico para mejorar el análisis.

```
ggplot() +
  # datos reales
  geom_point(data = titles,
             aes(x = release_year, y = imdb_score),
             color = "gray70", alpha = 0.4, size = 1) +
  # curvas estimadas
  geom_line(data = all_preds,
            aes(x = release_year, y = imdb_score, color = k),
            size = 1) +
  scale_color_brewer("Nodos (k)", palette = "Dark2") +
  labs(
    title    = "IMDb Score vs. Año de estreno",
    subtitle = "Splines cúbicos sin penalización (  = 0) y datos reales",
    x        = "Año de estreno",
    y        = "IMDb Score"
  ) +
  theme_minimal()
```



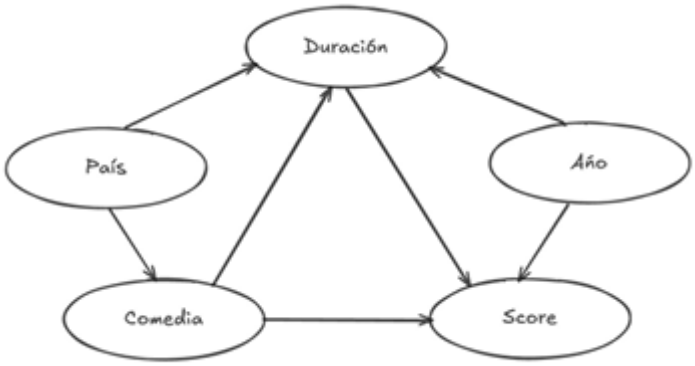
Al añadir los puntos reales al gráfico, podemos apreciar claramente la densidad y variabilidad de los `imdb_score` a lo largo del tiempo. Las curvas con muy pocos nodos (k = 1–3) quedan demasiado rígidas, pasando casi siempre por el centro de la nube de datos y perdiendo detalles de cambios reales. Por otro lado, las curvas con muchos nodos (k = 20, 50) se adaptan en exceso al ruido, mostrando oscilaciones que no se corresponden con ninguna tendencia subyacente.

Un valor intermedio de nodos (k = 5 – 10) captura las variaciones generales (picos y valles suaves) sin sobreadaptarse a cada punto aislado. Esto sugiere que, si quisiéramos un spline cúbico sin penalización que refleje de manera realista la evolución de las puntuaciones de IMDb según el año de estreno, un `k` de alrededor de 5 o 10 sería el mejor compromiso entre sesgo y varianza.

4. A qué subconjunto de las variables `Año`, `Duración` y `País` se debe condicionar para estimar el efecto causal promedio de la variable `Comedia` sobre el `Score`?

Dado el DAG:

```
knitr::include_graphics("DAG.png")
```



El único camino de "back-door" que conecta `Comedia` con `Score` es:

`Comedia` ← `País` → `Duración` → `Score`

Para bloquear ese sesgo sin tocar mediadores (`Duración`) y sin abrir otros caminos:

- Condicionar en `País` rompe el único back-door.
- **No** deben ajustarse ni `Duración` (es mediador) ni solo `Año` (no bloquea el back-door).
- Incluir `Año` además de `País` **no** introduce sesgo extra, pero tampoco es necesario.

Conjuntos de ajuste válidos

1. **Mínimo**: {`País`}
2. **Superset también válido**: {`País`, `Año`}

Cualquier otro subconjunto de {`Año`, `Duración`, `País`} introduce sesgo o deja el back-door abierto.

5. Comparación de modelos predictivos para `imdb_score`

A continuación dividimos el dataset en entrenamiento y testeo (80/20), probamos tres modelos distintos y comparamos su RMSE en el conjunto de test:

```
data_mod <- titles %>%
  mutate(
    comedia           = factor(comedia),
    type              = factor(type),
    age_certification = factor(age_certification)
  ) %>%
  select(imdb_score, release_year, runtime, imdb_votes, comedia, type, age_certification)

set.seed(123)
train_idx <- createDataPartition(data_mod$imdb_score, p = 0.8, list = FALSE)
train <- data_mod[train_idx, ]
test  <- data_mod[-train_idx, ]

# 5-fold CV
ctrl <- trainControl(method = "cv", number = 5)
```

```r
set.seed(123)
mod_lm <- train(imdb_score ~ ., data = train,
                method    = "lm",
                trControl = ctrl)
set.seed(123)
mod_rf <- train(imdb_score ~ ., data = train,
                method    = "ranger",
                trControl = ctrl,
                tuneLength = 3)
set.seed(123)
mod_xgb <- train(imdb_score ~ ., data = train,
                 method     = "xgbTree",
                 trControl  = ctrl,
                 tuneLength = 3)
```

```r
pred_lm  <- predict(mod_lm,  test)
pred_rf  <- predict(mod_rf,  test)
pred_xgb <- predict(mod_xgb, test)

rmse_lm  <- RMSE(pred_lm,  test$imdb_score)
rmse_rf  <- RMSE(pred_rf,  test$imdb_score)
rmse_xgb <- RMSE(pred_xgb, test$imdb_score)

results <- tibble(
  Modelo = c("Regresión lineal", "Random Forest", "XGBoost"),
  RMSE   = c(rmse_lm, rmse_rf, rmse_xgb)
)

print(results)
```

```
## # A tibble: 3 x 2
##   Modelo            RMSE
##   <chr>            <dbl>
## 1 Regresión lineal 1.05
## 2 Random Forest    0.995
## 3 XGBoost          1.00
```

Como el modelo de Random Forest tiene el menor RMSE, lo utilizaremos para computar nuestras predicciones del siguiente punto.

6. Predicción de score de nuevos títulos

```r
titles_test <- read.csv("titles_test.csv", stringsAsFactors = FALSE)

med_runtime <- median(train$runtime,  na.rm = TRUE)
med_votes   <- median(train$imdb_votes, na.rm = TRUE)

titles_test_mod <- titles_test %>%
  mutate(
    comedia = factor(
```

```
      grepl("comedy", genres, ignore.case = TRUE),
      levels = levels(train$comedia)
    ),
    type              = factor(type,              levels = levels(train$type)),
    age_certification = factor(age_certification, levels = levels(train$age_certification)),
    runtime    = ifelse(is.na(runtime),    med_runtime, runtime),
    imdb_votes = ifelse(is.na(imdb_votes), med_votes,   imdb_votes)
  )

titles_test_mod2 <- titles_test_mod %>% mutate(
  imdb_votes = ifelse(is.na(imdb_votes), med_votes, imdb_votes)
)

pred_rf_test <- predict(mod_rf, newdata = titles_test_mod2)

stopifnot(length(pred_rf_test) == nrow(titles_test))

write.table(
  data.frame(prediction = pred_rf_test),
  file      = "predicciones.csv",
  col.names = FALSE,
  row.names = FALSE
)
```