

# EST-25134: Aprendizaje Estadístico

**Profesor:** Alfredo Garbuno Iñigo — Primavera, 2023 — Métodos de selección.

**Objetivo:** Detalles en métodos de selección de variables. Veremos las estrategias de regularización y penalización para ajustar modelos controlando el sesgo predictivo hacia el conjunto de entrenamiento. Utilizaremos validación cruzada para probar configuraciones y elegir la *mejor*. Hablaremos sobre reducción de dimensiones y su combinación con métodos predictivos.

**Lectura recomendada:** Capítulo 6 de [1]. Sección 6.4 de [2]. Aunque el enfoque es regresión, los principios de validación cruzada para escoger modelos penalizados son análogos en el contexto de clasificación. Puedes leer la sección 12.5 de [2].

## 1. INTRODUCCIÓN

Ya hemos visto cómo cuantificar el **error de generalización** en un proceso de aprendizaje. Es decir, cuantificar los **errores de predicción** sobre nuevas observaciones y, además, **cuantificar la variabilidad** de esta predicción. En esta parte estudiaremos cómo incorporar lo aprendido para **escoger un modelo sobre otro**.

Por ejemplo, hemos visto que es natural **extender** el modelo lineal

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon. \quad (1)$$

Veremos (mas adelante) la idea de incorporar relaciones **no lineales** manteniendo el supuesto de **aditividad**.

Incluso aunque el modelo lineal es sencillo, tiene sus ventajas pues nos ayuda a tener un modelo **interpretable** y al mismo tiempo con buena **capacidad predictiva**.

El libro de Kuhn and Johnson [2] tiene una buena discusión sobre las ventajas algorítmicas de un modelo lineal. Usualmente en la práctica queremos tener nuestro modelo en un ambiente productivo. Lo cual necesita que las predicciones sean fácilmente calculables. ¿Qué pasaría si en la plataforma de Netflix o Amazon se tarda mucho en aparecer las sugerencias? Los modelos lineales son fácilmente calculables en prácticamente cualquier ambiente productivo.

Estudiaremos estrategias para mejorar modelos lineales a través de procedimientos alternativos de ajuste.

### 1.1. Opciones para ajustar modelos

- Basados en **precisión de ajuste**, ideal cuando  $p > n$  con el objetivo de *reducir* varianza.
- Basados en **interpretabilidad**. Por ejemplo, eliminar variables que no tengan capacidad predictiva.

## 2. ESTRATEGIAS DE SELECCIÓN DE VARIABLES

- Selección por subconjuntos.
- Reducción de coeficientes (regularización).
- Reducción de dimensiones.

## 2.1. Selección por subconjuntos

El mecanismo sería el siguiente.

1. Utilizar el modelo nulo  $\mathcal{M}_0$  (sin predictores).
2. Para  $k = 1, \dots, p$ :
  - a) Ajustar todos modelos posibles con  $k$  predictores.
  - b) Elegir el *mejor* de esa colección de modelos, le pondremos  $\mathcal{M}_k$ .
3. Elegir el mejor modelo dentro de la colección  $\mathcal{M}_0, \dots, \mathcal{M}_p$  utilizando un **criterio de comparación de modelos**.

2.1.1. *Para pensar:* ¿Por qué no puedes utilizar el criterio de RSS para escoger entre las opciones  $\mathcal{M}_1, \dots, \mathcal{M}_p$ ?

## 2.2. En el contexto de clasificación

La **devianza** –el negativo de dos veces la log-verosimilitud– se utiliza como una métrica de bondad de ajuste (como el RSS) para una clase mas amplia de modelos.

## 2.3. Selección iterativa

Podemos elegir empezar con el modelo mas sencillo e ir incorporando una variable a la vez mas predictores. En cada paso podemos evaluar la **mejora adicional** de haber incorporado estas nuevas características.

## 2.4. Pseudo-código (selección hacia adelante)

El proceso sería el siguiente.

1. Denotamos por  $\mathcal{M}_0$  el modelo nulo.
2. Para  $k = 0, \dots, p - 1$ :
  - a) Considera todos los  $p - k$  modelos que aumentan el modelo en la iteración anterior  $\mathcal{M}_k$  con un predictor adicional.
  - b) Escoge el **mejor** de estos  $p - k$  modelos y llámale  $\mathcal{M}_{k+1}$ .
3. Escoge el mejor de los modelos entre  $\mathcal{M}_0, \dots, \mathcal{M}_p$  utilizando un criterio de comparación de modelos.

## 2.5. Aplicación: créditos

```

1 # A tibble: 400 × 10
2   Income Limit Rating Cards Age Education Gender Student Married Balance
3   <dbl> <int> <int> <int> <int> <int> <fct> <fct> <fct> <int>
4 1  14.9  3606  283     2   34      11 Male   No    Yes    333
5 2 106.   6645  483     3   82      15 Female Yes    Yes    903
6 3 105.   7075  514     4   71      11 Male   No    No     580
7 4 149.   9504  681     3   36      11 Female No    No     964
8 5  55.9  4897   357     2   68      16 Male   No    Yes    331
9 # ... with 395 more rows
10 # Use 'print(n = ...)' to see more rows

```

LISTING 1. Muestra de datos del conjunto *Credit*.

El objetivo es predecir **Saldo** utilizando las demás características. El ejemplo de [1] ha implementado la búsqueda por subconjuntos y la búsqueda iterativa hacia adelante. Estos son los mejores modelos encontrados.

# Variables	Best subset	Forward stepwise
One	<b>rating</b>	<b>rating</b>
Two	<b>rating, income</b>	<b>rating, income</b>
Three	<b>rating, income, student</b>	<b>rating, income, student</b>
Four	<b>cards, income</b> <b>student, limit</b>	<b>rating, income,</b> <b>student, limit</b>

FIGURA 1. Método de selección para los datos de créditos. Tomada de [1].

Nota que el mecanismo iterativo no tiene garantía de encontrar el mejor modelo dentro de las  $\binom{p}{k}$  posibilidades.

```

1 # A tibble: 2 × 6
2   estrategia  sigma r.squared adj.r.squared  AIC deviance
3   <chr>      <dbl>   <dbl>         <dbl> <dbl>   <dbl>
4 1 subconjunto  99.6     0.954         0.953 4823. 3915058.
5 2 adelante    101.     0.952         0.952 4835. 4032502.

```

LISTING 2. Métricas de bondad de ajuste para los datos de *Credit*.

2.5.1. *Para pensar:* ¿Cuántos modelos en total se ajustan con el procedimiento de búsqueda iterativa hacia adelante? Considera  $p = 20$ .

## 2.6. Selección iterativa hacia atrás

Empezamos con el **modelo completo** que contenga los  $p$  predictores. Eliminando variables, una a la vez, cuando un predictor no sea tan útil. La única restricción que necesitamos es que  $n > p$ .

## 3. MÉTRICAS DE DESEMPEÑO

Si utilizáramos el RSS para comparar entre  $\mathcal{M}_0, \dots, \mathcal{M}_k$  tendríamos un problema pues eliminar (aumentar) predictores siempre perjudicaría (beneficia) la capacidad predictiva del modelo. Necesitamos **compensar** por el sesgo de sobre-ajuste. Es decir, considerar una métrica que pueda estimar el error de **generalización**.

### 3.1. $C_p$ de Mallows

Es un criterio de bondad de ajuste (**menor mejor**) definida como

$$C_p(\mathcal{M}_d) = \frac{1}{n} (\text{RSS}(d) + 2d\hat{\sigma}^2). \quad (2)$$

Tenemos una penalización a la suma de residuales al cuadrado (RSS) que considera un aumento en predictores utilizados.

### 3.2. El criterio de información de Akaike (AIC)

Se utiliza para evaluar modelos ajustados por máxima verosimilitud (**menor mejor**)

$$\text{AIC}(\mathcal{M}_d) = -2 \log L + 2d\hat{\sigma}^2. \quad (3)$$

3.2.1. *Ejercicio:* Prueba que en el caso del modelo lineal con errores Gaussianos el criterio de mínimos cuadrados y máxima verosimilitud es el mismo. Además los criterios  $C_p$  y AIC son lo mismo.

### 3.3. $R^2$ ajustada

Se calcula como

$$R_A^2(\mathcal{M}_d) = 1 - \frac{\text{RSS}/(n - d - 1)}{\text{TSS}/(n - 1)}. \quad (4)$$

Es una métrica de correlación entre predicción ( $\hat{y}$ ) y respuesta ( $y$ ) (**mayor mejor**). Al contrario de la  $R^2$  tradicional esta métrica si se afecta por la inclusión de variables inecesarias/redundantes.

### 3.4. Objetivo

Cada uno de los procedimientos de selección de variables regresa una secuencia de modelos  $\mathcal{M}_k$ . Lo que queremos es escoger la  $k^*$  de acuerdo al **error de generalización**. El error de generalización obtenido por **validación cruzada** tiene la ventaja de no hacer la estimación de  $\sigma^2$ .

Estimar  $\sigma^2$  es una tarea complicada. Implica, bajo el modelo de regresión, estimar el mejor modelo y encontrar la precisión de la familia de modelos que estamos utilizando.

3.4.1. *Selección de modelo: Datos de crédito* El objetivo es predecir el **Saldo** en términos de los demás predictores. Se seleccionarán las variables de acuerdo a un proceso iterativo. En este caso por **búsqueda hacia adelante**.

**Funciones a utilizar:**

```
1 train <- analysis(split)
2 valid <- assessment(split)
```

LISTING 3. Separación de muestras

```
1 modelo.nulo <- lm(Balance ~ 1, train)
2 modelo.completo <- lm(Balance ~ ., train)
```

LISTING 4. Ajuste de modelos *esquina*.

```
1 adelante <- step(modelo.nulo,
2               direction='forward',
3               scope=formula(modelo.completo),
4               trace=0)
5 predictores <- attributes(adelante$terms)$term.labels
```

LISTING 5. Instrucción de ajuste iterativo.

Escogemos el modelo con el error mas pequeño. Sin embargo, validación cruzada nos puede dar una métrica de incertidumbre (¿cuál?). ¿Y si el problema de decisión lo planteamos como una prueba de hipótesis?

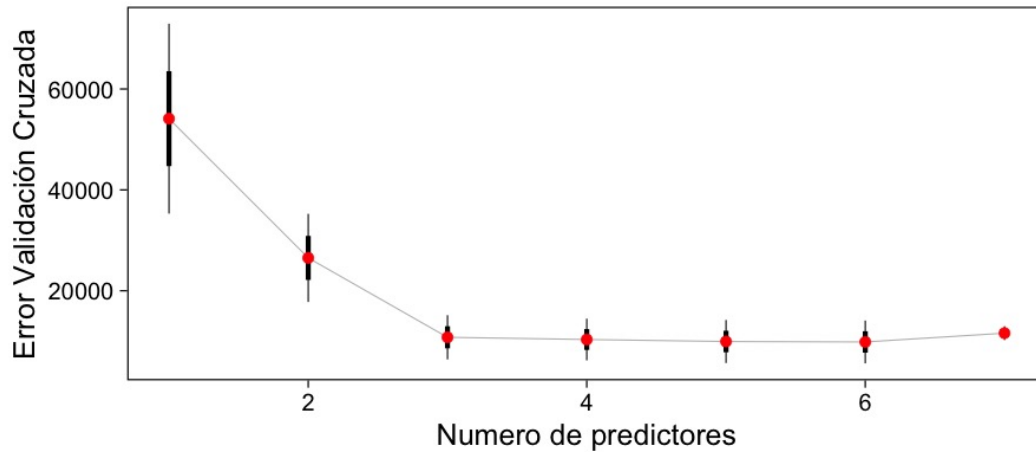


FIGURA 2. Error de generalización estimado por validación cruzada con  $K = 10$ . Para los datos de *Credit*.

### 3.5. El método de un error estándar

1. Se estima la función de pérdida generalizada parametrizada por los hiper-parámetros.

$$CV_{(K)}(\lambda) = \frac{1}{K} \sum_{k=1}^K MSE_k(\lambda). \quad (5)$$

2. Se localiza el valor de los hiper-parámetros que minimicen dicha función de pérdida.

$$\hat{\lambda} = \arg \min_{\lambda \in \Lambda} CV_{(K)}(\lambda). \quad (6)$$

3. Se escoge el modelo mas sencillo que se encuentre a un error estándar.

$$\hat{\lambda}^* \text{ tal que } CV_{(K)}(\lambda) \leq CV_{(K)}(\hat{\lambda}) + SE \left( CV_{(K)}(\hat{\lambda}) \right). \quad (7)$$

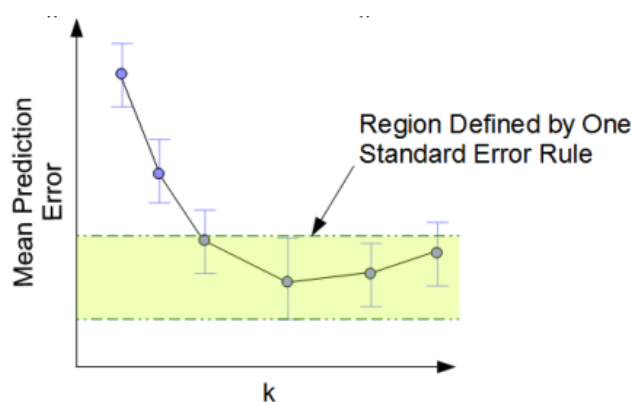


FIGURA 3. Imagen tomada de las notas del curso ofrecido en [CMU](#).

## 4. REGULARIZACIÓN

Los procedimientos selección de variables discretos/iterativos pueden generar una **varianza** muy alta en las estimaciones del error y podría no reducir el error de predicción del

modelo completo. Estudiaremos dos métodos de regularización, **Ridge** y **LASSO**, donde ajustamos un modelo con todas las características *penalizando* de alguna manera la complejidad del modelo.

#### 4.1. Regresión Ridge

Nuestra formulación anterior consideraba encontrar  $\beta_0, \beta_1, \dots, \beta_n$  minimizando

$$\text{RSS} = \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_j \right)^2. \quad (8)$$

Lo que haremos ahora será incorporar un **término de penalización** en la función objetivo

$$\text{RSS} + \lambda \sum_{j=1}^p \beta_j^2, \quad (9)$$

donde  $\lambda \geq 0$  es un **hyper-parámetro**.

El objetivo sigue siendo el mismo, ajustar el modelo lo mejor posible. El término adicional favorece soluciones con  $\beta_1, \dots, \beta_p$  pequeños. El parámetro  $\lambda$  controla qué tanto **penalizamos** el *tamaño* de los coeficientes.

*4.1.1. Para pensar:* Un valor muy pequeño para  $\lambda$  implica una penalización **pequeña**, por lo tanto la solución tenderá a ser un modelo **altamente flexible**. Por otro lado un valor de  $\lambda$  grande implica una penalización **fuerte**. Esto se traduce en un solución **poco flexible**.

¿Por qué la regresión Ridge mejora las estimaciones en contraste con mínimos cuadrados?

R: La descomposición en sesgo-varianza.

La regularización disminuye la varianza de las estimaciones, ¿por qué?

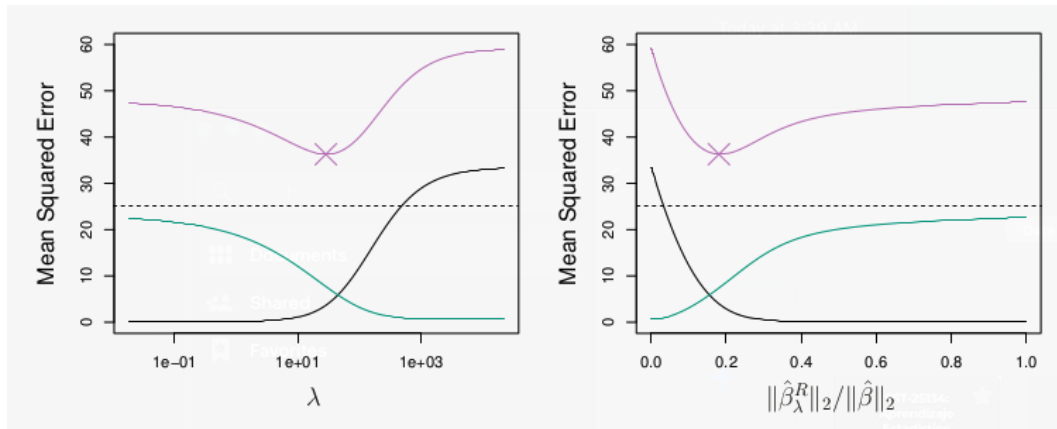
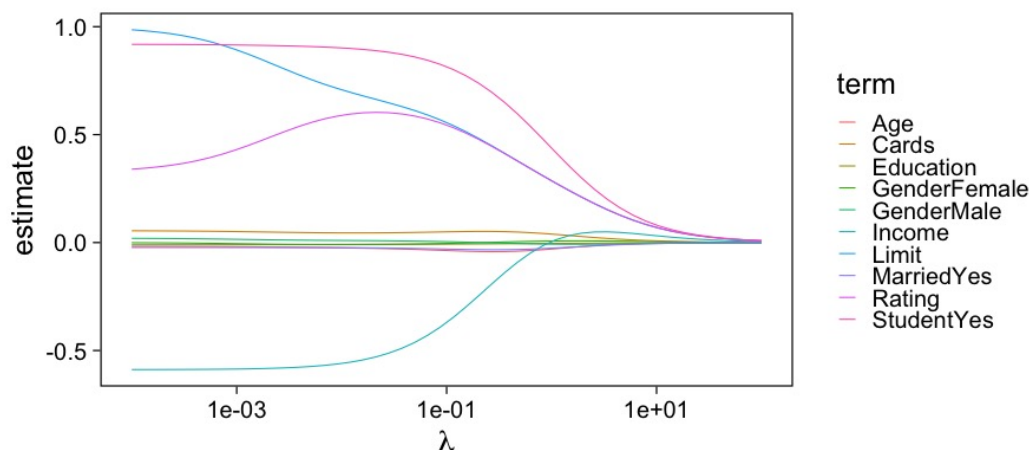


FIGURA 4. Sesgo (negro), varianza (verde) y error predictivo fuera de muestra (morado).

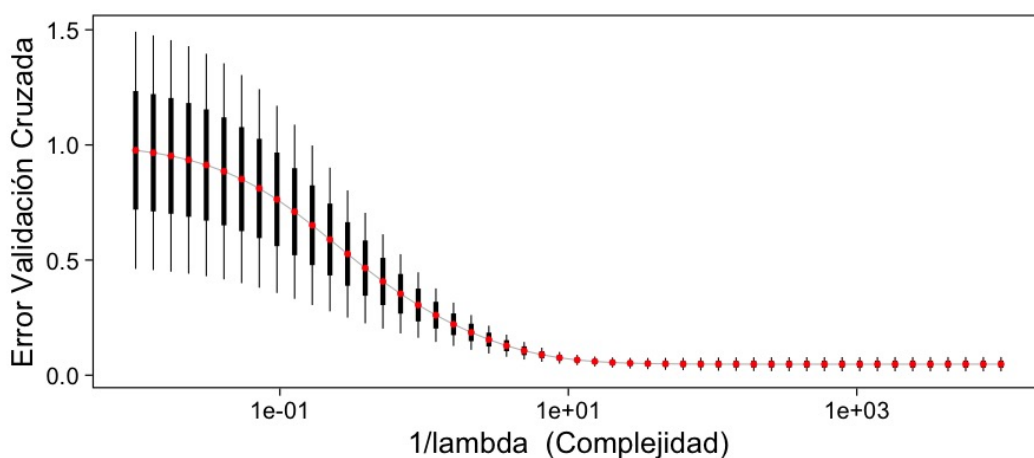
#### 4.2. Ridge: datos de crédito

Usaremos Ridge como mecanismo de reducción de coeficientes para ajustar modelos parsimoniosos.

Observa que conforme aumenta la penalización los coeficientes disminuyen gradualmente.

FIGURA 5. Trayectorias de los coeficientes al aumentar la penalización  $\lambda$ .

Al penalizar sobre los coeficientes necesitamos que todos *platicuen* en el mismo idioma. Es por esto que tenemos que estandarizar los predictores. Si queremos estimar el error de generalización métodos de separación de muestras, ¿en qué momento lo hacemos? Es decir, ¿antes de separar los datos o en cada paso del proceso de ajuste?

FIGURA 6. Error de validación calculada con  $K = 10$ . Nota que graficamos contra  $1/\lambda$ .

Con validación cruzada podemos identificar qué valor de  $\lambda$  es el adecuado para penalizar. Una vez realizada esta elección, re-entrenamos el modelo utilizando **todo** el conjunto de datos para predecir situaciones/observaciones futuras.

**4.2.1. Implementación con *tidymodels*:** Primero tenemos que realizar un *split* de los datos en entrenamiento y prueba.

```
1 data_split <- initial_split(data)
2 data_train <- training(data_split)
3 data_test <- testing(data_split)
```

Tenemos que especificar el modelo.

```

1 ridge_spec <- linear_reg(penalty = tune(), mixture = 0) ▷
2   set_engine(engine = "glmnet")
3
4 ridge_spec

```

```

1 Linear Regression Model Specification (regression)
2
3 Main Arguments:
4   penalty = tune()
5   mixture = 0
6
7 Computational engine: glmnet

```

Con el modelo especificado podemos definir el proceso de pre-procesamiento a través de un *recipe*.

```

1 ## Defino el pre-procesamiento
2 ridge_recipe <- recipe(Balance ~ ., data = data_train) ▷
3   step_dummy(all_nominal_predictors()) ▷
4   step_normalize(all_predictors())
5
6 ridge_recipe

```

LISTING 6. Definir el pre-procesamiento con *recipes*.

```

1 Recipe
2
3 Inputs:
4
5   role #variables
6   outcome      1
7   predictor      9
8
9 Operations:
10
11 Dummy variables from all_nominal_predictors()
12 Centering and scaling for all_predictors()

```

```

1 ridge_workflow <- workflow() ▷
2   add_recipe(ridge_recipe) ▷
3   add_model(ridge_spec)
4
5 ridge_workflow

```

```

1 Workflow
2 Preprocessor: Recipe
3 Model: linear_reg()
4
5 Preprocessor
6 2 Recipe Steps•
7
8   step_dummy()•

```



```

9   step_normalize()
10
11   Model
12   Linear Regression Model Specification (regression)
13
14   Main Arguments:
15     penalty = tune()
16     mixture = 0
17
18   Computational engine: glmnet

```

Definimos la partición de los datos para realizar una estimación del error de generalización.

```

1   data_folds <- vfold_cv(data_train, v = 10)
2   data_folds > print(n = 5)

```

```

1   # 10-fold cross-validation
2   # A tibble: 10 × 2
3     splits      id
4   <list>      <chr>
5 1 <split [270/30]> Fold01
6 2 <split [270/30]> Fold02
7 3 <split [270/30]> Fold03
8 4 <split [270/30]> Fold04
9 5 <split [270/30]> Fold05
10  # ... with 5 more rows
11  # Use 'print(n = ...)' to see more rows

```

Definimos el espacio de búsqueda de  $\lambda$

```

1   penalty_grid <- grid_regular(penalty(range = c(-3, 5)), levels = 50)
2   penalty_grid > print(n = 5)

```

```

1   # A tibble: 50 × 1
2     penalty
3   <dbl>
4 1 0.001
5 2 0.00146
6 3 0.00212
7 4 0.00309
8 5 0.00450
9   # ... with 45 more rows
10  # Use 'print(n = ...)' to see more rows

```

Usamos la partición de los datos en bloques para ajustar múltiples modelos con distintos conjuntos de entrenamiento y sus respectivas métricas de generalización

```

1   tune_res <- tune_grid(
2     ridge_workflow,
3     resamples = data_folds,
4     metrics = metric_set(rmse),
5     grid = penalty_grid
6   )
7   tune_res > print(n = 5)

```

```

1 # Tuning results
2 # 10-fold cross-validation
3 # A tibble: 10 × 4
4   splits      id      .metrics      .notes
5   <list>      <chr>    <list>      <list>
6 1 <split [270/30]> Fold01 <tibble [100 × 5]> <tibble [0 × 3]>
7 2 <split [270/30]> Fold02 <tibble [100 × 5]> <tibble [0 × 3]>
8 3 <split [270/30]> Fold03 <tibble [100 × 5]> <tibble [0 × 3]>
9 4 <split [270/30]> Fold04 <tibble [100 × 5]> <tibble [0 × 3]>
10 5 <split [270/30]> Fold05 <tibble [100 × 5]> <tibble [0 × 3]>
11 # ... with 5 more rows
12 # Use 'print(n = ...)' to see more rows

```

```

1 tune_res > unnest(.metrics) > print(n = 5)

```

```

1 # A tibble: 1,000 × 8
2   splits      id      penalty .metric .estimator .estimate .config .notes
3   <list>      <chr>    <dbl> <chr>    <chr>      <dbl> <chr>    <list>
4 1 <split [270/30]> Fold01 0.001  rmse     standard    131. ...Preproc <
   tibble>
5 2 <split [270/30]> Fold01 0.00146 rmse     standard    131. ...Preproc <
   tibble>
6 3 <split [270/30]> Fold01 0.00212 rmse     standard    131. ...Preproc <
   tibble>
7 4 <split [270/30]> Fold01 0.00309 rmse     standard    131. ...Preproc <
   tibble>
8 5 <split [270/30]> Fold01 0.00450 rmse     standard    131. ...Preproc <
   tibble>
9 # ... with 995 more rows
10 # Use 'print(n = ...)' to see more rows

```

```

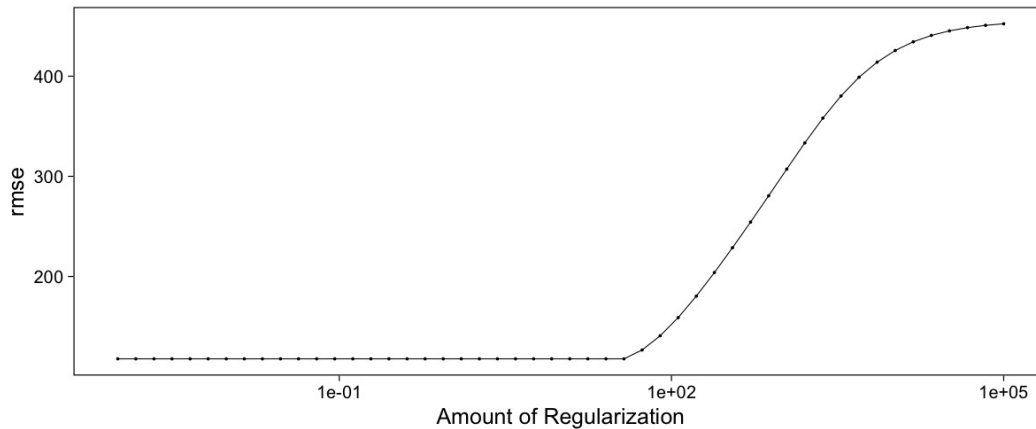
1 collect_metrics(tune_res) > print(n = 5)

```

```

1 # A tibble: 100 × 7
2   penalty .metric .estimator      mean      n std_err .config
3   <dbl> <chr>    <chr>    <dbl> <int>   <dbl> <chr>
4 1 0.001  rmse     standard  122.      10 3.87  Preprocessor1_Model01
5 2 0.001  rsq       standard  0.937     10 0.00575 Preprocessor1_Model01
6 3 0.00146 rmse     standard  122.      10 3.87  Preprocessor1_Model02
7 4 0.00146 rsq       standard  0.937     10 0.00575 Preprocessor1_Model02
8 5 0.00212 rmse     standard  122.      10 3.87  Preprocessor1_Model03
9 # ... with 95 more rows
10 # Use 'print(n = ...)' to see more rows

```



```
1 best_penalty <- select_best(tune_res, metric = "rmse")
2 best_penalty
```

```
1 # A tibble: 1 × 2
2   penalty .config
3   <dbl> <chr>
4 1 0.001 Preprocessor1_Model01
```

Podemos seleccionar el mejor modelo regularizado

```
1 ridge_final <- finalize_workflow(ridge_workflow, best_penalty)
2 ridge_final_fit <- fit(ridge_final, data = data_train)
```

y obtener su capacidad predictiva en el conjunto de prueba para reportar

```
1 augment(ridge_final_fit, new_data = data_test) >
2   rmse(truth = Balance, estimate = .pred)
```

```
1 # A tibble: 1 × 3
2   .metric .estimator .estimate
3   <chr>    <chr>        <dbl>
4 1 rmse    standard      111.
```

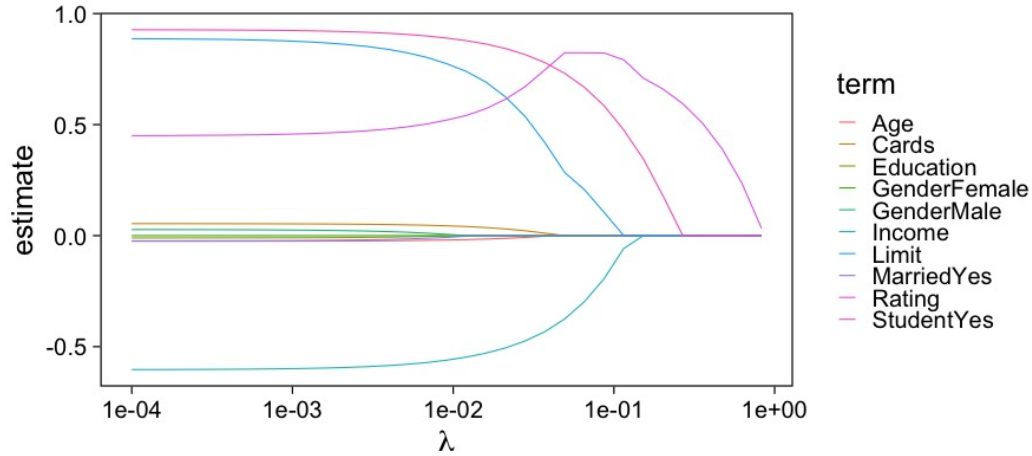
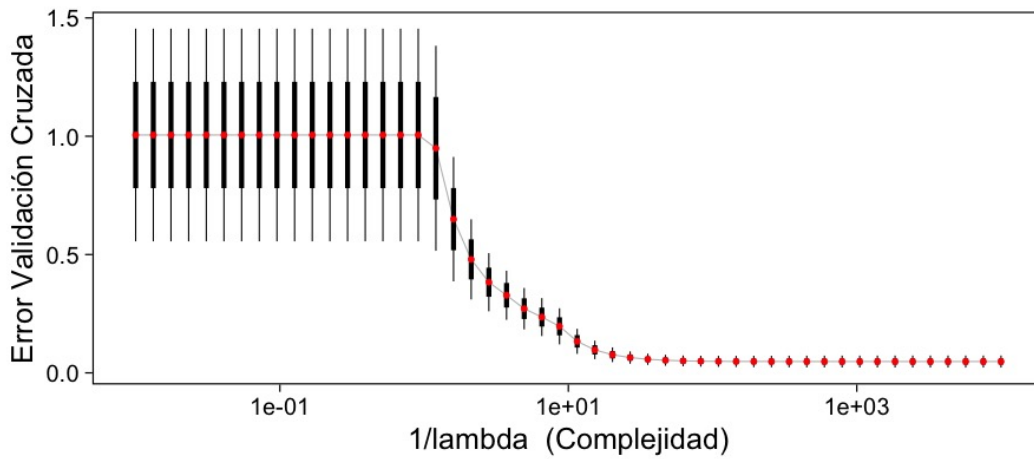
### 4.3. Regresión LASSO

En la práctica Ridge no elimina completamente los predictores. Podemos cambiar la penalización para incorporar un término de penalización en la función objetivo

$$\text{RSS} + \lambda \sum_{j=1}^p |\beta_j|, \quad (10)$$

donde  $\lambda \geq 0$  es un hiper-parámetro.

Igual que antes... el objetivo sigue siendo el mismo, ajustar el modelo lo mejor posible. El término adicional favorece soluciones con  $\beta_1, \dots, \beta_p$  pequeños. El parámetro  $\lambda$  controla qué tanto penalizamos el tamaño de los coeficientes.

FIGURA 7. Trayectorias de los coeficientes al aumentar la penalización  $\lambda$ .FIGURA 8. Error de validación calculada con  $K = 10$ . Nota que graficamos contra  $1/\lambda$ .

#### 4.4. LASSO: datos de crédito

Observa que LASSO tiene la propiedad de eliminar completamente los predictores ( $\beta = 0$ ) por lo que es un mecanismo de **selección automática de variables**.

#### 4.5. Comparación: Ridge v. LASSO

El problema de optimización (Ridge) se puede reescribir de la siguiente manera

$$\text{minimizar RSS,} \quad \text{sujeto a } \sum_{j=1}^p \beta_j^2 \leq s, \quad (11)$$

y el respectivo de LASSO

$$\text{minimizar RSS,} \quad \text{sujeto a } \sum_{j=1}^p |\beta_j| \leq s. \quad (12)$$

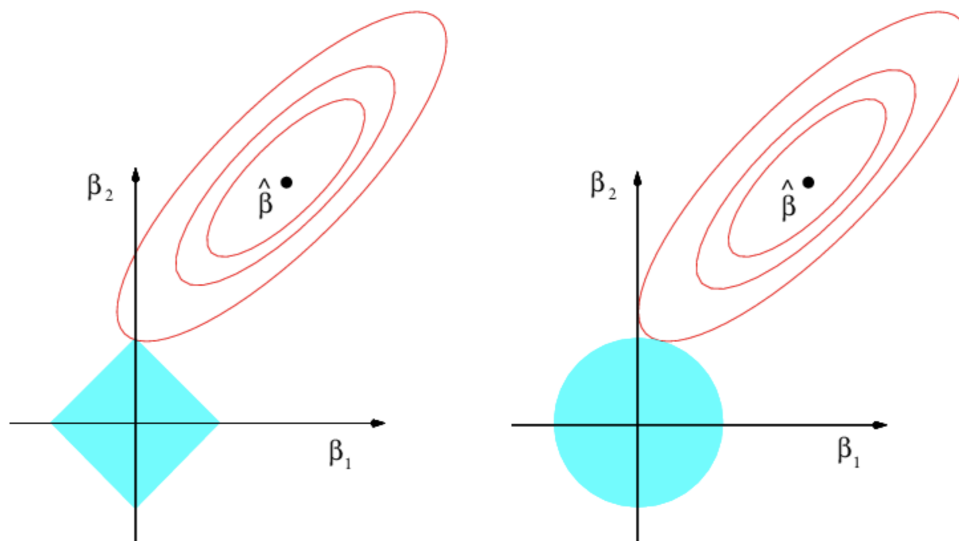


FIGURA 9. Curvas de nivel de los problemas de optimización: LASSO (izquierda) y Ridge (derecha). Tomada de [1].

## 5. APLICACIÓN: RATING DE ESPISODIOS

Queremos predecir las calificaciones en IMDB de los episodios de *The Office*. Utilizaremos información que hay sobre el *rating* y sobre los episodios (actores, directores, escritores y diálogo). Ejemplo tomado de [aquí](#).

Veamos la participación de cada personaje en los episodios

```

1 # A tibble: 185 × 16
2   episode_name  Andy Angela Darryl Dwight   Jim Kelly Kevin Michael Oscar
3   <chr>         <int>  <int>  <int>  <int> <int> <int> <int>  <int> <int> <
4   int>
5 1 a benihana ...c    28    37     3    61    44     5    14   108     1
6   57
7 2 aarm          44    39    30    87    89     0    30     0    28
8   34
9 3 after hours    20    11    14    60    55     8     4     0    10
10  15
11 4 alliance       0     7     0    47    49     0     3    68    14
12  22
13 5 angry y       53     7     5    16    19    13     9     0     7
14  29
15
16 # ... with 180 more rows, and 5 more variables: Phyllis <int>, Ryan <int>,
17 #   Toby <int>, Erin <int>, Jan <int>
18 # Use 'print(n = ...)' to see more rows, and 'colnames()' to see all variable
19 #   names

```

También veamos al equipo creativo detrás de cada episodio.

```

1 # A tibble: 135 × 14
2   episode_name Ken ...1Kw Greg ...2 B.J. ...3 Paul ..... Mindy Paul ... Gene
3   ... Lee ...E

```

```

3   <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <
4   dbl>
1 pilot      1      1      0      0      0      0      0
0
5 2 diversity day 1      0      1      0      0      0      0
0
6 3 health care  0      0      0      1      0      0      0
0
7 4 basketball   0      1      0      0      0      0      0
0
8 5 hot girl     0      0      0      0      1      0      0
0
9 # ... with 130 more rows, 5 more variables: 'Jennifer Celotta' <dbl>,
10 #   'Randall Einhorn' <dbl>, 'Brent Forrester' <dbl>, 'Jeffrey Blitz' <dbl>,
11 #   'Justin Spitzer' <dbl>, and abbreviated variable names 1'Ken Kwapis',
12 # 2 'Greg Daniels', 3'B.J. Novak', 'Paul Lieberstein', 'Mindy Kaling',
13 #   'Paul Feig', 'Gene Stupnitsky', 'Lee Eisenberg'
14 # Use 'print(n = ...)' to see more rows, and 'colnames()' to see all variable
    names

```

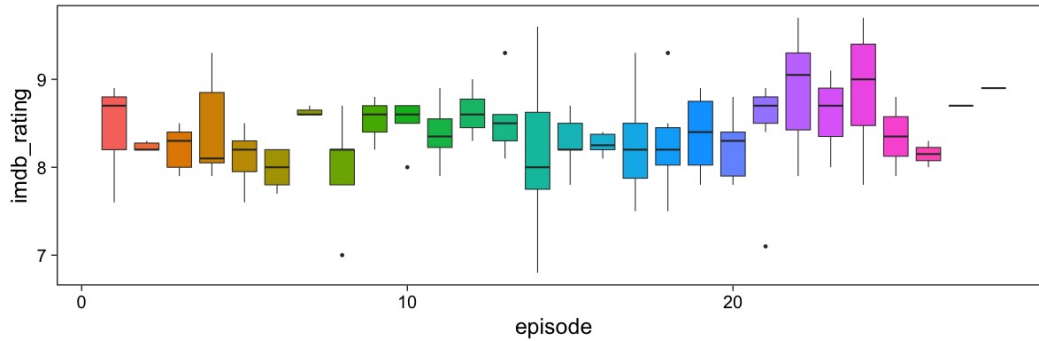
Juntamos toda la información en un sólo conjunto de datos.

```

1   Joining, by = "episode_name"
2   Joining, by = "episode_name"
3   Joining, by = "episode_name"
4   # A tibble: 136 × 32
5   season episode episode_...1 andy angela darryl dwright jim kelly kevin
6   michael
7   <dbl> <dbl> <chr>      <int> <int> <int> <int> <int> <int> <int> <
8   int>
1   1      1 pilot      0      1      0      29      36      0      1
81
2   1      2 ...diversity 0      4      0      17      25      2      8
75
3   1      3 health ...ca  0      5      0      62      42      0      6
56
4   1      5 basketball 0      3      15      25      21      0      1
104
5   1      6 hot girl   0      3      0      28      55      0      5
106
12 # ... with 131 more rows, 21 more variables: oscar <int>, pam <int>,
13 #   phyllis <int>, ryan <int>, toby <int>, erin <int>, jan <int>,
14 #   ken_kwapis <dbl>, greg_daniels <dbl>, b_j_novak <dbl>,
15 #   paul_lieberstein <dbl>, mindy_kaling <dbl>, paul_feig <dbl>,
16 #   gene_stupnitsky <dbl>, lee_eisenberg <dbl>, jennifer_celotta <dbl>,
17 #   randall_einhorn <dbl>, brent_forrester <dbl>, jeffrey_blitz <dbl>,
18 #   justin_spitzer <dbl>, imdb_rating <dbl>, and abbreviated variable name ...
19 # Use 'print(n = ...)' to see more rows, and 'colnames()' to see all variable
    names

```

Para el ajuste de un modelo hay que cuidar el proceso de exploración de datos (pueden ver el siguiente video [aquí](#)).



### 5.1. Proceso de entrenamiento

```

1 office_split <- initial_split(office, strata = season)
2 office_train <- training(office_split)
3 office_test <- testing(office_split)

```

Preparamos un procesamiento para el entrenamiento usual.

```

1 ## Con mixture = 1, pedimos lasso
2 tune_spec <- linear_reg(penalty = tune(), mixture = 1) >
3   set_engine("glmnet") >
4   set_mode("regression")
5
6 office_rec <- recipe(imdb_rating ~ ., data = office_train) >
7   update_role(episode_name, new_role = "ID") >
8   step_zv(all_numeric_predictors()) >
9   step_normalize(all_numeric_predictors())
10
11 office_prep <- office_rec >
12   prep(strings_as_factors = FALSE)

```

### 5.2. Validación cruzada

```

1 set.seed(108727)
2 office_boot <- vfold_cv(office_train, v = 10, strata = season)

```

```

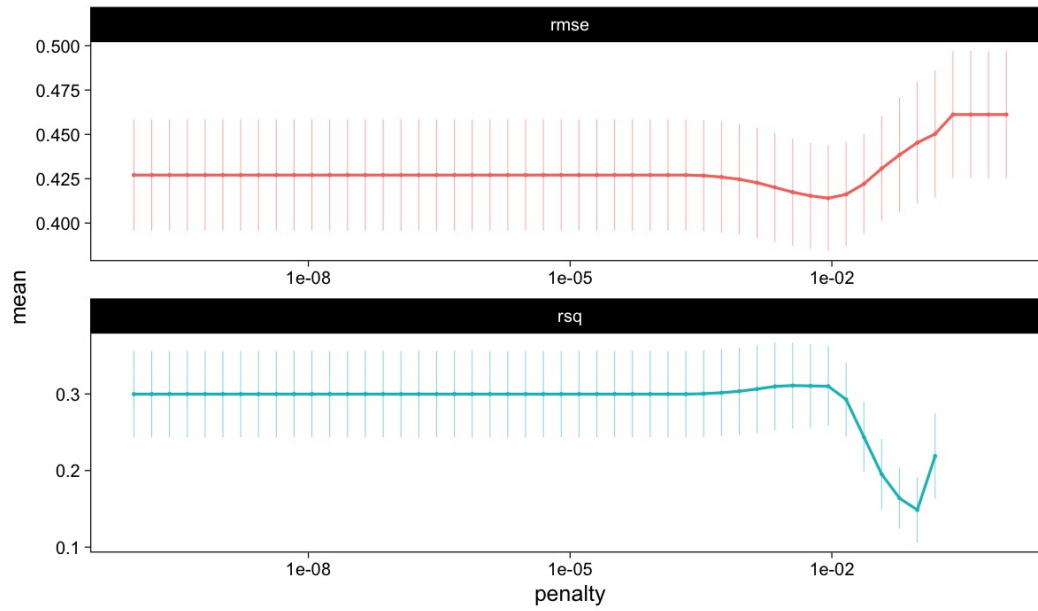
1 set.seed(2020)
2 wf <- workflow() >
3   add_recipe(office_rec) >
4   add_model(tune_spec)
5
6 lasso_grid <- wf >
7   tune_grid(
8     resamples = office_boot,
9     grid = lambda_grid,
10    control = control_grid(verbose = FALSE)
11  )

```

```

1 lasso_grid >
2   collect_metrics()

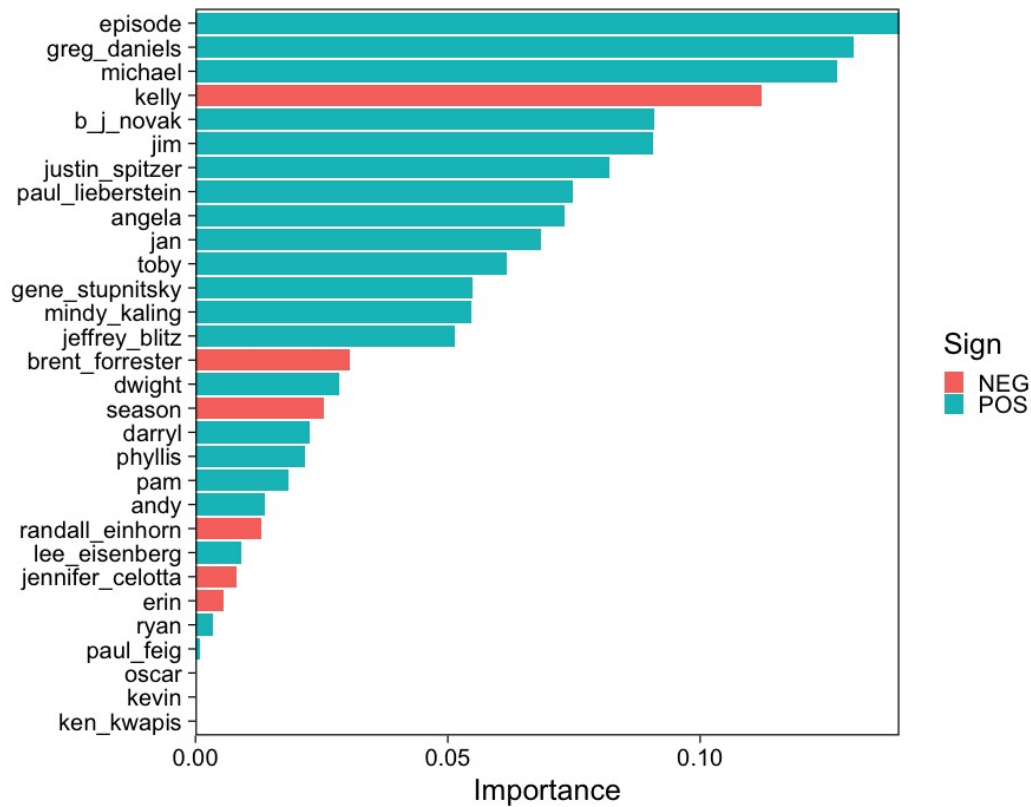
```



### 5.3. Selección de modelo

```
1 lowest_rmse <- lasso_grid >  
2   select_best("rmse")  
3  
4 final_lasso <- finalize_workflow(  
5   wf > add_model(tune_spec),  
6   lowest_rmse  
7 )
```





```

1 final_lasso >
2   fit(office_train) >
3   augment(new_data = office_test) >
4   rmse(truth = imdb_rating, estimate = .pred)

```

```

1 # A tibble: 1 × 3
2   .metric .estimator .estimate
3   <chr>   <chr>         <dbl>
4 1 rmse    standard       0.482

```

## 5.4. Conclusiones

En la práctica no hay una estrategia dominante. LASSO podría ser preferido cuando el número de parámetros es pequeño. Pero eso implica conocer *a priori* el número de predictores para usar en el modelo.

5.4.1. Para pensar: ¿cómo escogerías entre Ridge o LASSO?

La penalización tipo LASSO es indiferente a la selección de variables dentro de un conjunto de variables altamente correlacionadas. Ridge, por su parte, tiende a agrupar los coeficientes en valores similares. La penalización de Elastic Net puede lograr un compromiso entre los puntos anteriores

$$\lambda [\alpha \|\beta\|_1 + (1 - \alpha) \|\beta\|_2^2] . \quad (13)$$

El segundo término logra promediar los coeficientes de atributos altamente correlacionados, mientras que el primer término se encarga de acercarlos mas a cero (una solución *sparse*).

Extender la selección de modelos a una búsqueda de varios hiper-parámetros se puede realizar dentro de `tidymodels` especificando los parámetros que se deben de ajustar en la definición del modelo y creando la *malla* adecuada de búsqueda.

```
1 tune_spec <- linear_reg(penalty = tune(), mixture = tune()) >
2   set_engine("glmnet") >
3   set_mode("regression")
```

```
1 hyper_params <- parameters(penalty(range = c(-5, 3)), mixture())
2 hparams_grid <- grid_regular(hyper_params, levels = 10)
```

Tenemos que asegurar que la especificación se actualiza para poder hacer la búsqueda.

```
1 set.seed(2020)
2 wf <- workflow() >
3   add_recipe(office_rec) >
4   add_model(tune_spec)
5
6 lasso_grid <- wf >
7   tune_grid(
8     resamples = office_boot,
9     grid = hparams_grid,
10    control = control_grid(verbose = FALSE)
11  )
```

Existen otras estrategias de búsqueda utilizando un diseño aleatorio en el espacio de hiper-parámetros.

```
1 grid_max_entropy(hyper.params, size = 50)
2 grid_random(hyper.params, size = 50)
```

## 6. MÉTODOS DE REDUCCIÓN DE DIMENSIONES

LASSO o Ridge utilizan el concepto de regularización para restringir los modelos posibles. Una alternativa es **transformar** primero los predictores (el espacio de los predictores) y **ajustar** un modelo con ese subespacio.

### 6.1. Regresión con reducción de dimensiones

Denotemos por  $Z_1, Z_2, \dots, Z_M$  combinaciones lineales de nuestros predictores originales. Lo escribimos como

$$Z_m = \sum_{j=1}^p \phi_{mj} X_j, \quad m = 1, \dots, M, \quad (14)$$

con algunas constantes  $\phi_{mj}$  (que se escogen con alguna estrategia).

Podemos ajustar un modelo de regresión por medio de

$$y_i = \theta_0 + \sum_{m=1}^M \theta_m z_{im} + \epsilon_i, \quad (15)$$

utilizando mínimos cuadrados.

Nota que podemos rescribir

$$\sum_{m=1}^M \theta_m z_{im} = \sum_{m=1}^M \theta_m \sum_{j=1}^p \phi_{mj} x_{ij} = \sum_{j=1}^p \beta_j x_{ij}, \quad (16)$$

donde

$$\beta_j = \sum_{m=1}^M \theta_m \phi_{mj}. \quad (17)$$

El modelo restringe automáticamente las  $\beta_j$  pues tienen que tomar una forma muy particular. Si las  $\phi_{mj}$  se escogen bien, incluso pueden realizar un mejor trabajo que el modelo de mínimos cuadrados en las variables originales.

### 6.2. Otros métodos de reducción de dimensiones

- Utilizar componentes principales (**varianza máxima entre predictores**).
- Utilizar *partial least squares* (**varianza máxima entre predictores y respuesta**).
- Utilizar *least angle regression* (trayectoria de **contribución lineal predictiva** de atributos).

## REFERENCIAS

- [1] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer US, New York, NY, 2021. [1](#), [2](#), [3](#), [13](#)
- [2] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. Springer New York, New York, NY, 2013. ISBN 978-1-4614-6848-6 978-1-4614-6849-3. [1](#)