Análisis de regresión

Capítulo 7: Selección de variables

Mg. Sc. J. Eduardo Gamboa U.

Introducción

¿Por qué seleccionar variables?

- No todas las variables aportan información relevante, incluso podrían ser perjudiciales, al:
  - aumentar varianza de las estimaciones
  - introducir multicolinealidad
  - dificultar la interpretación del modelo
- Un modelo más complejo no es necesariamente un mejor modelo.
- Existe un compromiso entre:
  - ajuste al conjunto de datos observado
  - capacidad de generalizar a nuevos datos
- Un buen modelo explica y predice

Complejidad de un modelo

- Modelo simple: Mayor sesgo, menor varianza
- Modelo complejo: Menos sesgo, mayor varianza
- Se busca un trade - off entre sesgo y varianza
- La selección de variables es una herramienta clave para controlar este trade-off.

Partición del conjunto de datos

- Training set:
  - conjunto de datos usado para ajustar el modelo
  - permite estimar los parámetros / ajustar el modelo.
- Testing set:
  - conjunto de datos no utilizado en el ajuste
  - permite evaluar el desempeño predictivo del modelo.

Se desea modelar y predecir el desempeño de proyectos locales en distintas regiones del país, a partir de indicadores técnicos y contextuales. La variable respuesta (y) representa un índice de resultado del proyecto (mayor valor = mejor desempeño).

Variables contextuales:

- region: Zona geográfica del proyecto (costa, sierra, selva)
- chas: Indicador de condición especial del proyecto (0 = proyecto estándar, 1 = proyecto con condición especial)
- x1, x3, x8: indicadores técnicas
- x_noise_1: variable de ruido (añadida a propósito)

```r
library(dplyr)
datos = read.csv('U7_datos_1.csv')
datos |> glimpse()
```

```
Rows: 400
Columns: 27
$ id          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
$ y           <dbl> 1.9551379, 5.7849582, -2.4141802, -5.8810339, -8.4244239, --
$ x1          <dbl> 0.5583667, 2.7847571, -0.1860256, -0.5704220, -1.4857318, --
$ x2          <dbl> 0.633837114, 2.377798003, 0.316173042, -0.771308738, -1.379-
$ x3          <dbl> 0.75562271, 2.96589324, -0.09379301, -0.80933045, -1.716456-
$ x4          <dbl> 2.08763645, 0.17514025, 1.02555656, 1.31667769, 0.32808600,-
$ x5          <dbl> 1.76439554, -0.16356903, 0.76352444, 1.17462637, 0.40276336-
$ x6          <dbl> 2.06702075, 0.03355828, 1.20910906, 1.80496291, 0.44270204,-
$ x7          <dbl> 2.04089805, -1.23051249, -0.55315067, -0.11026346, -1.61679-
$ x8          <dbl> -0.84608144, -0.19442985, 1.58504729, 0.76283472, 0.1983325-
$ x9          <dbl> -1.358097684, 0.207993693, -1.314629908, 1.261792891, 1.789-
$ chas        <int> 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,-
$ region      <chr> "selva", "sierra", "sierra", "selva", "selva", "sierra", "s-
$ x1_sq       <dbl> 0.311773379, 7.754872382, 0.034605540, 0.325381252, 2.20739-
$ x4_logabs   <dbl> 1.12740590, 0.16138751, 0.70584450, 0.84013413, 0.28373880,-
$ x2_dup      <dbl> 0.66912967, 2.39808178, 0.30162115, -0.75293961, -1.3885479-
$ x5_ratio    <dbl> 0.68733201, -0.30656263, 0.44673828, 0.50960749, 0.42724354-
$ x_noise_1   <dbl> 1.676379495, -0.458178275, -1.311927281, 1.754940403, 1.925-
$ x_noise_2   <dbl> 0.091780656, 0.097275930, -0.906347140, -1.781315120, -1.18-
$ x_noise_3   <dbl> -0.547396841, 1.775555748, 1.155523847, -1.074383895, -1.36-
$ x_noise_4   <dbl> -0.13464043, -0.51677198, 0.53098968, 0.26345653, 0.0571865-
$ x_noise_5   <dbl> 0.10186177, 0.39851524, 1.89178435, 0.78017412, -1.50363223-
$ x_noise_6   <dbl> 1.25297595, -0.15159062, 0.86036105, 0.13020656, -1.1597834-
$ x_noise_7   <dbl> 0.59261813, 1.41411221, 2.19451507, 0.55508367, -0.79243619-
$ x_noise_8   <dbl> 0.46163472, -1.26705669, 0.53880475, -0.74665440, 0.9387081-
$ x_noise_9   <dbl> -0.93485797, 2.31640495, 0.66355770, 0.89710784, -0.7059756-
$ x_noise_10 <dbl> -0.48669654, 1.66628941, 0.12639955, -0.35370677, -1.152087-
```

```
datos = datos  |>
  mutate(region = as.factor(region),
         chas   = as.factor(chas)) |>
  select(id, y, region, chas, x1, x3, x8, x_noise_1)

library(rsample)
set.seed(2026)
split_obj = initial_split(datos, prop = 0.80)
train     = training(split_obj)
test      = testing(split_obj)
train |> nrow()
```

```
[1] 320
```

```
test  |> nrow()
```

```
[1] 80
```

```r
modelo    = lm(y ~ region + chas + x1 + x3 + x8 + x_noise_1,
               data = train)
modelo |> coef()
```

```
 (Intercept)  regionselva regionsierra        chas1           x1           x3
  -0.3431788    0.1524842   -0.2008598    0.6755253    2.3432076    0.7122237
          x8    x_noise_1
   0.8080033   -0.2272422
```

Sobreajuste

- Un modelo está sobreajustado cuando se adapta demasiado a los datos de entrenamiento.
- Causa: modelo excesivamente complejo.
- Consecuencia: excelente ajuste en training, mal desempeño en datos nuevos

Enfoques

- Enfoque estadístico:
  - Orientado a la inferencia, interpretación o explicación.
  - Estimación mediante mínimos cuadrados.
- Machine learning:
  - Orientado a la predicción, generalización.
  - Entrenamiento (training) minimizando una función de pérdida.

¿Cómo evaluar un modelo?

- Evaluar un modelo implica analizar dos dimensiones:
  - su capacidad de ajustarse a los datos observados
  - su capacidad de predecir datos no observados
- Un buen ajuste en el conjunto de entrenamiento no garantiza buena generalización.
- El objetivo central es estimar el error de predicción fuera de muestra.

Medidas de predicción

- PRESS: suma de los errores de predicción individuales, es una estimación del error de predicción fuera de muestra.
- R^2 predictivo: mide qué proporción de la variabilidad es explicada en predicción, no en ajuste

PRESS

Suma de cuadrados predictiva o Suma de cuadrados de los errores predictivos $(y_i - \hat{y_{i,-i}})$.

$$PRESS = \sum_{i=1}^{n} \left(\frac{e_i}{1 - h_{ii}} \right)^2$$

A menor PRESS, menor error de predicción estimado
donde:

- e_i es el i-ésimo residual
- h_{ii} es el i-ésimo hat value

```
residuals(modelo)/(1-modelo |> hatvalues()) -> pr
(sum(pr**2) -> PRESS)
```

```
[1] 3950.502
```

```
(modelo |> rstandard(type="pred"))**2 |> sum()
```

```
[1] 3950.502
```

R^2 de predicción

Se busca maximizar esta medida, que se construye de manera similar a la del coeficiente de determinación (R^2) bajo el concepto de validación cruzada LOO (leave-one-out).

$$R_p^2 = 1 - \frac{PRESS}{SCT}$$

donde SCT es la suma de cuadrados total.

```
library(broom)
library(dplyr)
modelo |> aov() |> tidy() |> pull(sumsq) |> sum() -> SCT
1-PRESS/SCT
```

```
[1] 0.4831847
```

```
library(olsrr)
modelo |> ols_pred_rsq()
```

```
[1] 0.4831847
```

```
ols_step_all_possible(modelo)$result |> data.frame() -> resultados
resultados |> psych::headTail(3)
```

```
    mindex   n                      predictors rsquare adjr rmse predrsq     cp
3        1   1                              x1    0.47 0.47 3.55    0.47  18.18
4        2   1                              x3    0.46 0.46 3.59    0.45   25.4
5        3   1                              x8    0.03 0.02 4.82    0.01 301.11
...    ... ...                             <NA>     ...  ...  ...     ...    ...
61      60   5      region x1 x3 x8 x_noise_1     0.5 0.49 3.45    0.48   8.95
60      61   5    region chas x3 x8 x_noise_1     0.5 0.49 3.46    0.48  12.36
58      62   5    region chas x1 x3 x_noise_1    0.48 0.47 3.53    0.46  23.94
63      63   6 region chas x1 x3 x8 x_noise_1    0.51  0.5 3.43    0.48      8
        aic    sbic     sbc    msep   fpe  apc  hsp
3   1724.89   816.6  1736.2 4057.23 12.76 0.53 0.04
4   1731.74  823.36 1743.04 4144.93 13.03 0.55 0.04
5   1921.17 1010.66 1932.48 7492.25 23.56 0.99 0.07
...     ...     ...     ...     ...   ...  ...  ...
61  1715.93     806 1746.08 3872.65 12.37 0.52 0.04
60  1719.37  809.31 1749.52 3914.55  12.5 0.52 0.04
58  1730.81  820.33 1760.96 4056.98 12.96 0.54 0.04
63  1714.92  805.15 1748.83 3848.64 12.33 0.51 0.04
```

```
resultados |> select(n,predrsq,predictors) |>
  arrange(-predrsq) |> head(5)
```

```
  n   predrsq           predictors
1 3 0.4909158           chas x1 x8
2 4 0.4906370 chas x1 x8 x_noise_1
3 2 0.4905717               x1 x8
4 3 0.4897158      x1 x8 x_noise_1
5 4 0.4890698        chas x1 x3 x8
```

```
modelo_r2pred <- lm(y ~ x1 + x8, data = train)
```

Criterios de información

- Todos los criterios de información parten del ajuste del modelo, incorporando una penalización por número de parámetros.
- Buscan aproximar el error de predicción sin particionar los datos.

Medidas asociadas

- AIC
- BIC
- SBIC
- Cp de Mallows

Criterio de Información de Akaike (AIC)

El modelo con menor AIC debe ser preferido. Si ambos modelos presentan AIC con diferencia menor a 2 unidades, elegir el más simple

$$AIC = 2(k+1) - 2log(L)$$

donde:

- k: número de parámetros en la ecuación de regresión
- $log(L)$: log verosimilitud del modelo

```
modelo |> logLik() |> as.numeric() -> logvero
modelo |> coef() |> length() -> k
2*(k+1)-2*logvero
```

```
[1] 1714.916
```

```
modelo |> AIC()
```

```
[1] 1714.916
```

```
resultados |> select(n,aic,predictors) |>
  arrange(aic) |> head(10)
```

```
   n      aic              predictors
1  3 1709.583              chas x1 x8
2  2 1709.888                   x1 x8
3  4 1710.150    chas x1 x8 x_noise_1
4  3 1710.777         x1 x8 x_noise_1
5  4 1710.838           chas x1 x3 x8
6  3 1711.336                x1 x3 x8
7  5 1711.409 chas x1 x3 x8 x_noise_1
8  4 1712.239      x1 x3 x8 x_noise_1
9  4 1713.098       region chas x1 x8
10 3 1713.572            region x1 x8
```

```
modelo_aic <- lm(y ~ x1 + x8, data = train)
```

Criterio de información Bayesiano de Schwarz (BIC)

Al igual que las anteriores medidas de información, el menor BIC debe ser preferido, siempre que la diferencia sea mayor a 2 unidades. Elegir el más simple entre los candidatos.

$$SBC = BIC = (k+1) \times log(n) - 2log(L)$$

donde:

- k: número de parámetros en la ecuación de regresión.
- n: tamaño de muestra utilizado para construir el modelo.
- $log(L)$: Log verosimilitud del modelo

```
modelo |> coef() |> length() -> k
modelo |> nobs() -> n
modelo |> logLik() |> as.numeric() -> logvero
(k+1)*log(n)-2*logvero
```

```
[1] 1748.831
```

```
modelo |> BIC()
```

```
[1] 1748.831
```

```
resultados |> select(n,sbc,predictors) |>
  arrange(sbc) |> head(10)
```

```
   n      sbc            predictors
1  2 1724.962                 x1 x8
2  3 1728.425            chas x1 x8
3  3 1729.618       x1 x8 x_noise_1
4  3 1730.178              x1 x3 x8
5  2 1731.452                 x3 x8
6  4 1732.760 chas x1 x8 x_noise_1
7  4 1733.448         chas x1 x3 x8
8  3 1734.138            chas x3 x8
9  4 1734.849    x1 x3 x8 x_noise_1
10 3 1736.144       x3 x8 x_noise_1
```

```
modelo_bic <- lm(y ~ x1 + x8, data = train)
```

Criterio de información Bayesiano de Sawa (SBIC)

Al igual que las anteriores medidas de información, el menor SBIC debe ser preferido, siempre que la diferencia sea mayor a 2 unidades. Elegir el más simple entre los candidatos.

$$SBIC = n \times \log\left(\frac{SCE}{n}\right) + 2(p+3)q - 2q^2$$

donde:

- n: tamaño de muestra utilizado para construir el modelo
- SCE: suma de cuadrados de los errores
- p: Número de variables (no considerar las variables indicadoras que pudieran crearse)
- $q = n\frac{CME}{SCE}$

```
modelo |> nobs() -> n
(modelo |> aov() |> tidy() |> pull(term) |> length()) - 1 -> p
modelo |> aov() |> tidy() |> filter(term=="Residuals") |> pull(sumsq) -> s
modelo |> aov() |> tidy() |> filter(term=="Residuals") |> pull(meansq) -> 
n*cme/sce -> q;n*log(sce/n) + 2*(p+3)*q -2*q**2
```

```
[1] 805.1534
```

```
modelo |> ols_sbic(modelo)
```

```
[1] 805.1534
```

Se muestra un ejemplo para un modelo reducido (retiraremos x3 y x_noise_1 para mostrar el ejemplo):

```
lm(y ~ .-x3-x_noise_1, train) -> modelo_reducido
modelo_reducido |> nobs() -> n
(modelo_reducido |> aov() |> tidy() |>
    pull(term) |> length()) - 1 -> p
modelo_reducido |> aov() |> tidy() |> filter(term=="Residuals") |>
  pull(sumsq) -> sce
modelo |> aov() |> tidy() |> filter(term=="Residuals") |> pull(meansq) -> c
n*cme/sce -> q;n*log(sce/n) + 2*(p+3)*q -2*q**2
```

```
[1] 805.0894
```

```
modelo_reducido |> ols_sbic(modelo)
```

```
[1] 805.0894
```

```
resultados |> select(n,sbic,predictors) |>
  arrange(sbic) |> head(10)
```

```
   n     sbic                  predictors
1  3 801.5988                  chas x1 x8
2  2 801.8265                      x1 x8
3  4 802.2451        chas x1 x8 x_noise_1
4  3 802.7623             x1 x8 x_noise_1
5  4 802.9117               chas x1 x3 x8
6  4 803.1636          region chas x1 x8
7  3 803.3078                   x1 x3 x8
8  3 803.5379               region x1 x8
9  5 803.5763     chas x1 x3 x8 x_noise_1
10 5 803.8037 region chas x1 x8 x_noise_1
```

```
modelo_sbic <- lm(y ~ x1 + x8, data = train)
```

Cp de Mallows

Se debe su nombre a Colin Lingwood Mallows (estadístico inglés nacido en 1930). Es un indicador que evalúa el ajuste de modelos de regresión, y se busca que su valor sea similar a la cantidad de variables

$$C_p = \frac{SCE_{subconjunto}}{CME_{completo}} - n + 2(p+1)$$

donde:

- $SCE_{subconjunto}$: Suma de cuadrados del error del modelo con un subconjunto de variables siendo evaluados
- $CME_{completo}$: Cuadrado medio del error del modelo con todas las variables
- n: Tamaño de muestra utilizado para construir el modelo
- p: Número de parámetros estimados.

Importante: En algunas referencias, p incluye al intercepto como variable, por lo que la fórmula cambiaría a:

$$C_p = \frac{SCE_{subconjunto}}{CME_{completo}} - n + 2p$$

Esta es la fórmula que emplearemos

```
modelo |> aov() |> tidy() |> filter(term=="Residuals") |>
  pull(sumsq) -> sce
modelo |> aov() |> tidy() |> filter(term=="Residuals") |>
  pull(meansq) -> cme
modelo |> nobs() -> n
modelo |> coef() |> length() -> p
sce/cme - n + 2*p
```

```
[1] 8
```

```
modelo |> ols_mallows_cp(modelo)
```

```
[1] 8
```

Para el modelo completo con un modelo que contiene solo un subconjunto de variables (retiraremos x3 y x_noise_1 para mostrar el ejemplo):

```
lm(y ~ .-x3-x_noise_1, train) -> modelo_reducido
modelo_reducido |> aov() |> tidy() |> filter(term=="Residuals") |>
  pull(sumsq) -> sce
modelo |> aov() |> tidy() |> filter(term=="Residuals") |>
  pull(meansq) -> cme
modelo |> nobs() -> n
modelo_reducido |> coef() |> length() -> p
sce/cme - n + 2*p
```

```
[1] 8.020811
```

```
modelo_reducido |> ols_mallows_cp(modelo)
```

```
[1] 8.020811
```

```
resultados |> select(n,cp,predictors) |>
  mutate(dif = n-cp,.after=cp) |> arrange(abs(dif)) |> head(10)
```

	n	cp	dif	predictors
1	4	3.879277	0.1207228	chas x1 x3 x8
2	3	2.610840	0.3891602	chas x1 x8
3	5	4.480990	0.5190099	chas x1 x3 x8 x_noise_1
4	3	3.786444	-0.7864437	x1 x8 x_noise_1
5	4	3.204809	0.7951907	chas x1 x8 x_noise_1
6	2	2.885550	-0.8855496	x1 x8
7	4	5.256072	-1.2560719	x1 x3 x8 x_noise_1
8	3	4.339169	-1.3391686	x1 x3 x8
9	5	6.711953	-1.7119532	region chas x1 x8 x_noise_1
10	6	8.000000	-2.0000000	region chas x1 x3 x8 x_noise_1

```
modelo_cpm <- lm(y ~ chas + x1 + x3 + x8, data = train)
```

Métodos paso a paso

- El número de modelos posibles crece exponencialmente con el número de variables.
- Evaluar todos los subconjuntos puede ser inviable.

Tipos de métodos

- Mejores subconjuntos: compara modelos con 1, 2, 3, ... variables
- Forward: añade variables relevantes al modelo nulo.
- Backward: elimina variables irrelevantes del modelo completo.
- Stepwise: combina ambos enfoques.

Selección a partir de los mejores subconjuntos

Este procedimiento consiste en elegir el mejor modelo con 1 variable predictora, con 2, con 3, y así sucesivamente hasta llegar al modelo completo con p variables.
Los representantes de los subconjuntos compiten para elegir el mejor modelo, utilizando los mismos criterios de la sección anterior.

```r
modelo |> ols_step_best_subset()
```

```
           Best Subsets Regression
---------------------------------------------
Model Index    Predictors
---------------------------------------------
     1         x1
     2         x1 x8
     3         chas x1 x8
     4         chas x1 x8 x_noise_1
     5         chas x1 x3 x8 x_noise_1
     6         region chas x1 x3 x8 x_noise_1
---------------------------------------------
```

Subsets Regression Summary

Model	R-Square	Adj. R-Square	Pred R-Square	C(p)	AIC	SBIC	SBC	MSEP	FPE	HSP	APC
1	0.4725	0.4709	0.4659	18.1803	1724.8932	816.5992	1736.1981	4057.2262	12.7581	0.0400	0.5341
2	0.4998	0.4967	0.4906	2.8855	1709.8884	801.8265	1724.9617	3859.4287	12.1737	0.0382	0.5096
3	0.5034	0.4987	0.4909	2.6108	1709.5830	801.5988	1728.4246	3843.8886	12.1621	0.0381	0.5091
4	0.5056	0.4994	0.4906	3.2048	1710.1497	802.2451	1732.7596	3838.8969	12.1836	0.0382	0.5100
5	0.5068	0.4989	0.4888	4.4810	1711.4093	803.5763	1737.7876	3842.2616	12.2317	0.0384	0.5121
6	0.5075	0.4965	0.4832	8.0000	1714.9164	805.1534	1748.8313	3848.6434	12.3289	0.0387	0.5145

```
AIC: Akaike Information Criteria
 SBIC: Sawa's Bayesian Information Criteria
 SBC: Schwarz Bayesian Criteria
 MSEP: Estimated error of prediction, assuming multivariate normality
 FPE: Final Prediction Error
 HSP: Hocking's Sp
 APC: Amemiya Prediction Criteria
```

Backward selection

Comienza con el modelo completo y se van retirando, una a una, variables que no contribuyan.
Una variable que fue retirada, no puede volver a entrar.
Modelo completo:

```
 (Intercept)  regionselva regionsierra         chas1            x1            x3
  -0.3431788    0.1524842   -0.2008598     0.6755253     2.3432076     0.7122237
          x8     x_noise_1
   0.8080033    -0.2272422
```

Ejecutar el siguiente código:

```
modelo |> ols_step_backward_p()
modelo |> ols_step_backward_p() |> plot()
modelo_completo |> ols_step_backward_aic()
modelo_completo |> ols_step_backward_aic() |> plot()
```

¿Cuál es el modelo resultante?

Forwad selection

Comienza con el modelo nulo y se van agregando, una a una, variables que contribuyan. Una variable que fue ingresada, no puede ser retirada.
Ejecutar el siguiente código:

```
modelo |> ols_step_forward_p()
modelo |> ols_step_forward_p() |> plot()
modelo |> ols_step_forward_aic()
modelo |> ols_step_forward_aic() |> plot()
```

¿Cuál es el modelo resultante?

Stepwise selection

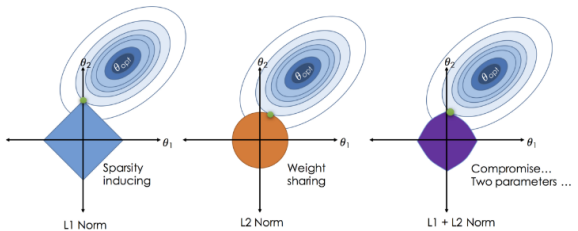
Comienza con el modelo nulo y se van agregando o retirando variables, una a una. Una variable que fue retirada, sí puede volver a entrar, así como una variable que fue ingresada, puede salir.
Ejecutar el siguiente código:

```
modelo |> ols_step_both_p()
modelo |> ols_step_both_p() |> plot()
modelo |> ols_step_both_aic()
modelo |> ols_step_both_aic() |> plot()
```

¿Cuál es el modelo resultante?

Regularización

- Controla la complejidad directamente en el proceso de estimación.



Métodos de regularización

- Ridge: reduce la varianza de los coeficientes, no los llega a eliminar.
- Lasso: induce coeficientes exactamente nulos, realizando así una selección automática.
- Elastic net: combina Ridge y Lasso mediante un coeficiente α. Es útil cuando hay alta correlación entre predictores.

Regresión Ridge

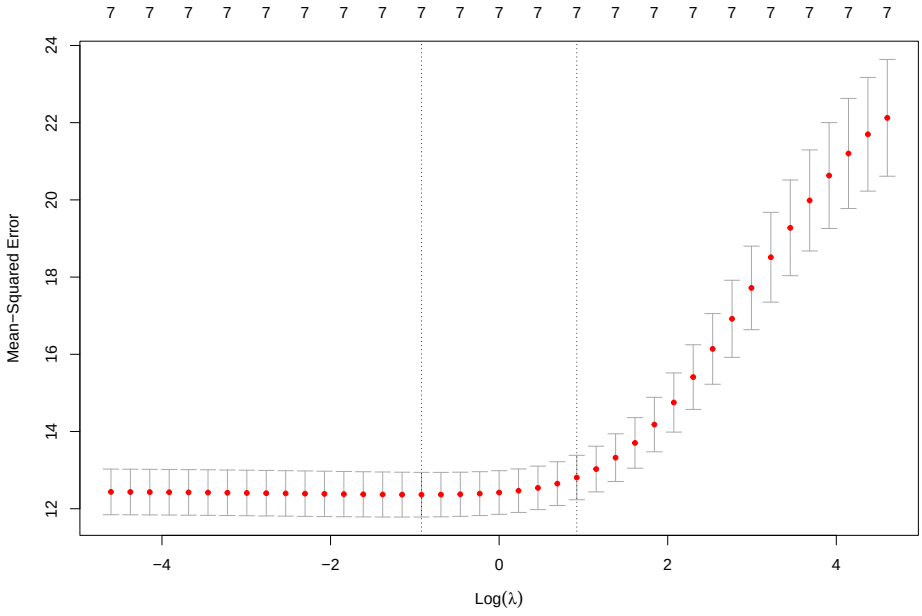
- El vector de coeficientes de regresión se estima mediante $\widehat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$
- La matriz $\mathbf{X}'\mathbf{X}$ puede presentar problemas de inversión si las variable están muy correlacionadas (y es necesario descartar alguna(s))
- Se propone el estimador $\widehat{\beta}_\lambda$, conocido como "estimador Ridge", el cual es sesgado pero menos variable que $\widehat{\beta}$.

$$\widehat{\beta}_\lambda = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y}$$

- Este estimador se obtiene al minimizar:

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

```r
modelo |> model.matrix() -> X
train  |> pull(y) -> Y
10**seq(2, -2, by = -.1) -> lambda_seq
library(glmnet)
set.seed(345)
cv.glmnet(X, Y, alpha = 0, lambda = lambda_seq) -> modelo_ridge_cv
modelo_ridge_cv |> plot()
```


7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
Mean−Squared Error
12
14
16
18
20
22
24
−4
−2
0
2
4
Log(λ)

```r
modelo_ridge_cv$lambda.1se -> best_lambda
glmnet(X, Y, alpha = 0, lambda = best_lambda) -> best_ridge
coef(best_ridge)
```

```
9 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  -0.2103338
(Intercept)   .
regionselva   0.1454745
regionsierra -0.1744242
chas1         0.4328956
x1            1.2730716
x3            1.0998468
x8            0.5288449
x_noise_1    -0.1616807
```

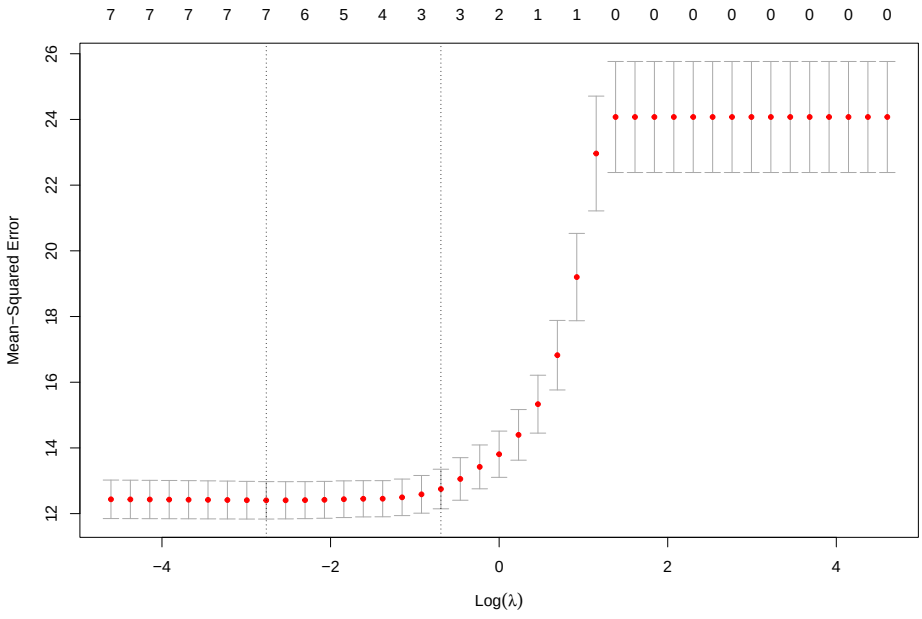
Regresión Lasso

- Se realiza un cambio en la penalidad, de modo que el estimador Lasso se obtiene al minimizar

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\beta)^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

- Tanto en la regresión Ridge como Lasso, los coeficientes de regresión no son interpretables (son sesgados) sin embargo sí es de interés obtener las predicciones.
- La regresión Lasso automáticamente realiza una selección de variables

```
set.seed(345)
cv.glmnet(X, Y, alpha = 1, lambda = lambda_seq) -> modelo_lasso_cv
modelo_lasso_cv |> plot()
```


7 7 7 7 7 6 5 4 3 3 2 1 1 0 0 0 0 0 0 0 0
Mean−Squared Error
12
14
16
18
20
22
24
26
−4
−2
0
2
4
Log(λ)

```r
modelo_lasso_cv$lambda.1se -> best_lambda
glmnet(X, Y, alpha = 1, lambda = best_lambda) -> best_lasso
coef(best_lasso)
```

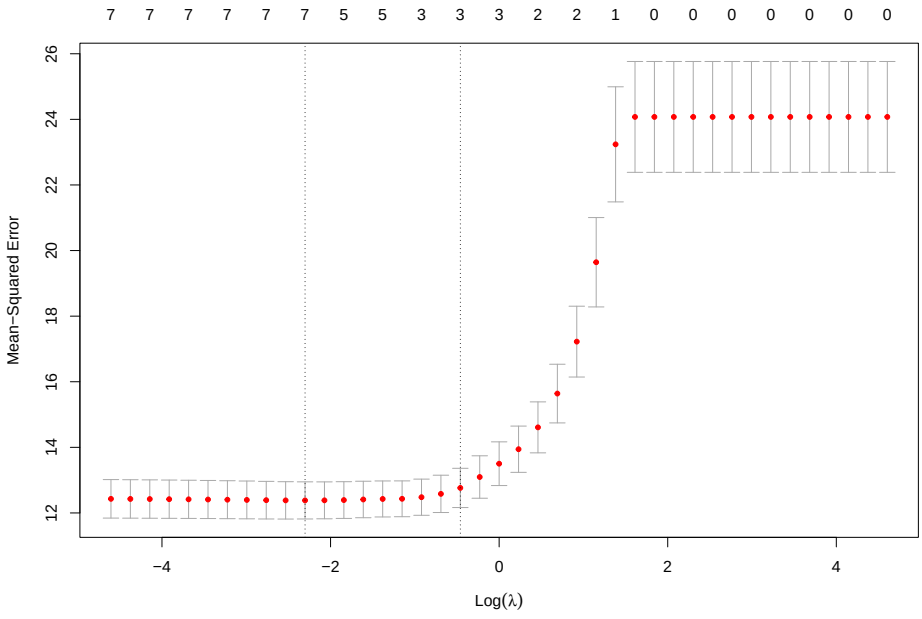
```
9 x 1 sparse Matrix of class "dgCMatrix"
                     s0
(Intercept)  0.02800487
(Intercept)  .
regionselva  .
regionsierra .
chas1        .
x1           2.31804834
x3           0.31854570
x8           0.29910056
x_noise_1    .
```

Regresión Elastic Net

- Es una mezcla de las regresiones Ridge y Lasso, ya que busca minimizar:

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\beta)^2 + \alpha\lambda_1 \sum_{j=1}^{p}|\beta_j| + (1-\alpha)\lambda_2 \sum_{j=1}^{p}\beta_j^2$$

```
set.seed(345)
cv.glmnet(X, Y, alpha = 0.8, lambda = lambda_seq) -> modelo_elastic_cv
modelo_elastic_cv |> plot()
```


7 7 7 7 7 7 5 5 3 3 3 2 2 1 0 0 0 0 0 0 0
Mean−Squared Error
12
14
16
18
20
22
24
26
−4
−2
0
2
4
Log(λ)

```r
modelo_elastic_cv$lambda.1se -> best_lambda
glmnet(X, Y, alpha = 0.8, lambda = best_lambda) -> best_elastic
best_elastic |> coef()
```

```
9 x 1 sparse Matrix of class "dgCMatrix"
                     s0
(Intercept)  0.02231246
(Intercept)  .
regionselva  .
regionsierra .
chas1        .
x1           1.73354938
x3           0.81508312
x8           0.29081796
x_noise_1    .
```

```r
X_test <- model.matrix(~ region + chas + x1 + x3 + x8 + x_noise_1,
                       data = test)
predicciones = data.frame(x = predict(best_elastic,X_test),
                          y = test$y)
predicciones |> head(10)
```

```
           s0         y
1  -3.8946527 -8.424424
2   1.4076523  4.947224
3   0.8003559  1.920726
4   1.2224520  1.888537
5   3.1463797  2.096090
6   3.2925127  9.462710
7   1.5618966  3.395282
8   2.9041331 -1.278497
9  -5.2597158 -5.993650
10  2.5428582  7.113242
```