Práctica Parte I Aprendizaje Estadístico

Antonio Galián Gálvez, Juan Luis Serradilla Tormos

2024-10-28

```
# Cargamos las librerías
library(gplots)
```

```
##
## Adjuntando el paquete: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```
library(glmnet)
```

```
## Cargando paquete requerido: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(MASS)
library(pls)
```

```
##
## Adjuntando el paquete: 'pls'
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

0. Cargamos los datos y eliminamos la columna train

```
# Cargamos los datos con separador de tabulador
datos <- read.delim("prostate.data.txt", header = TRUE, sep = "\t")

# Eliminamos la columna train
datos <- datos[, -ncol(datos)]
```

1. Exploración de datos

¿Cuántas variables hay?

```r
# Vemos las variables que hay
ncol(datos)
```

```
## [1] 10
```

Vemos que hay 10 variables.

¿De qué clase son?

```r
# Vemos la clase de las variables
str(datos)
```

```
## 'data.frame':    97 obs. of  10 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
##  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
##  $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
##  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
##  $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
##  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

Vemos que son variables numéricas.

¿Hay una variable que correspondiente al identificador de paciente? Si es así, elimínala.

```r
# Como hay una variable Id, llamada "X", la eliminamos
datos <- datos[, -1]
```

La variable X era la Id de los pacientes. Tras eliminarla quedan 9 varaibles.

¿Hay valores nulos en alguno de los ficheros?

```r
# Comprobamos si hay NA
sum(is.na(datos))
```

```
## [1] 0
```

No hay valores nulos.

¿Están estandarizadas las variables? En este punto del análisis, ¿es necesario normalizarlas?

```r
# Comprobamos si las variables estan estandarizadas
summary(datos)
```

```
##      lcavol           lweight           age             lbph        
##  Min.   :-1.3471   Min.   :2.375   Min.   :41.00   Min.   :-1.3863  
##  1st Qu.: 0.5128   1st Qu.:3.376   1st Qu.:60.00   1st Qu.:-1.3863  
##  Median : 1.4469   Median :3.623   Median :65.00   Median : 0.3001  
##  Mean   : 1.3500   Mean   :3.629   Mean   :63.87   Mean   : 0.1004  
##  3rd Qu.: 2.1270   3rd Qu.:3.876   3rd Qu.:68.00   3rd Qu.: 1.5581  
##  Max.   : 3.8210   Max.   :4.780   Max.   :79.00   Max.   : 2.3263  
##       svi              lcp             gleason          pgg45       
##  Min.   :0.0000   Min.   :-1.3863   Min.   :6.000   Min.   :  0.00  
##  1st Qu.:0.0000   1st Qu.:-1.3863   1st Qu.:6.000   1st Qu.:  0.00  
##  Median :0.0000   Median :-0.7985   Median :7.000   Median : 15.00  
##  Mean   :0.2165   Mean   :-0.1794   Mean   :6.753   Mean   : 24.38  
##  3rd Qu.:0.0000   3rd Qu.: 1.1787   3rd Qu.:7.000   3rd Qu.: 40.00  
##  Max.   :1.0000   Max.   : 2.9042   Max.   :9.000   Max.   :100.00
```

```
##       lpsa
##  Min.   :-0.4308
##  1st Qu.: 1.7317
##  Median : 2.5915
##  Mean   : 2.4784
##  3rd Qu.: 3.0564
##  Max.   : 5.5829
```

Vemos que las variables no están estandarizadas, ya que no tienen media 0 ni desviación 1. No hace falta normalizarlas en este caso.

¿Por qué crees que algunas variables están es escala logarítmica?

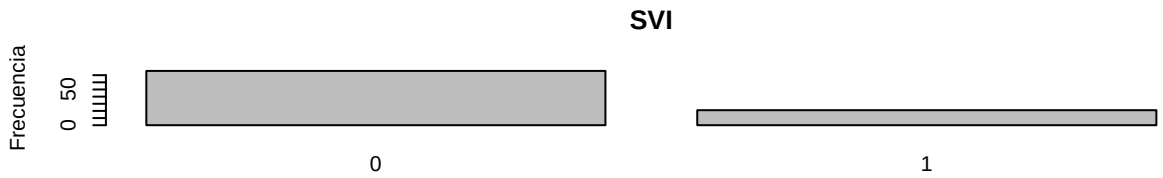
Algunas variables pueden estar en escala logarítmica para reducir relaciones de potencias y exponenciales entre dos variables a relaciones lineales, para así poder realizar regresiones lineales. Otra razón puede ser que estas variables tomen valores de órdenes de magnitud muy diferentes, simplificando su tratamiento al hacer el logaritmo.

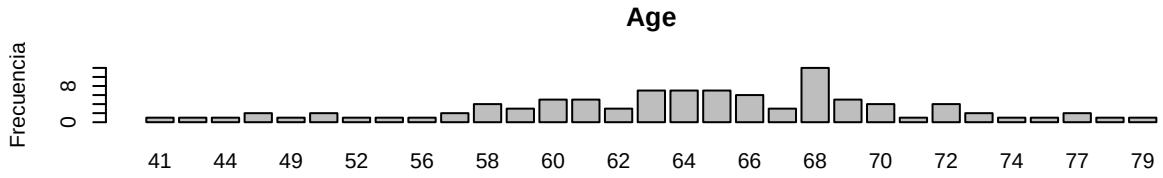
2. Análisis de variable categóricas

```r
# Convertimos las variables en factores
datos$svi <- as.factor(datos$svi)
datos$gleason <- as.factor(datos$gleason)
datos$age <- as.factor(datos$age)

# Hacemos attach a los datos
attach(datos)

# Comprobamos que las variables son categóricas
str(datos)
```

```
## 'data.frame':    97 obs. of  9 variables:
##  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
##  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
##  $ age    : Factor w/ 31 levels "41","43","44",..: 6 11 27 11 15 6 17 11 4 16 ...
##  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ svi    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ gleason: Factor w/ 4 levels "6","7","8","9": 1 1 2 1 1 1 1 1 1 1 ...
##  $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
##  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

```r
# Comprobamos la dispersión de sus valores
par(mfrow = c(3, 1))
plot(svi, main = "SVI", ylab = "Frecuencia")
plot(gleason, main = "Gleason", ylab = "Frecuencia")
plot(age, main = "Age", ylab = "Frecuencia")
```






```
par(mfrow = c(1, 1))
```

Podemos ver en los gráficos anteriores las tendencias de agrupamiento en los valores de las variables.

- En SVI la mayoría de los datos se concentran en SVI = 0.
- En Gleason la mayoría de los datos se acumulan en 7, mientras que apenas tenemos muestra en 8.
- En Age las datos se agrupan alrededor 64 y disminuyen al alejarse, aunque el valor 68 dispone de una cantidad de muestras por encima de lo normal.

3. Análisis de frecuencias

¿Qué porcentaje de pacientes con la puntuación de Gleason igual a 7, presenta índice igual svi igual a 0?

```
# Seleccionamos los pacientes con la puntuación de Gleason igual a 7 y los que tienen svi igual a 0 den
datos.gleason7 <- datos[datos$gleason == "7", ]
datos.gleason7.svi0 <- datos.gleason7[datos.gleason7$svi == "0", ]

# Vemos los pacientes que hay en datos.gleason.7.svi0 y en gleason7
patients.gleason7.svi0 <- nrow(datos.gleason7.svi0)
patients.gleason7 <- nrow(datos.gleason7)

# Dividimos la cantidad de pacientes filtrados entre el total
porcentaje <- patients.gleason7.svi0 / patients.gleason7 * 100 # creo que hay que dividir entre el los
porcentaje
```

```
## [1] 66.07143
```

Vemos que el porcentaje es del 66.07143%.

¿Qué porcentaje de pacientes con índice svi igual a 0 tiene la puntuación de Gleason igual a 7?

```
# Seleccionamos los individuos con svi igual a 0 y con gleason igual a 7 dentro de estos
datos.svi0 <- datos[datos$svi == "0", ]
datos.svi0.gleason7 <- datos.svi0[datos.svi0$gleason == "7", ]

# Vemos los pacientes que hay en datos.svi0
patients.svi0 <- nrow(datos.svi0)

# Hacemos el porcentaje
porcentaje <- patients.gleason7.svi0 / patients.svi0 * 100
porcentaje
```

```
## [1] 48.68421
```

Vemos que el porcentaje es del 48.68421%.

Estas dos variables, ¿son independientes?

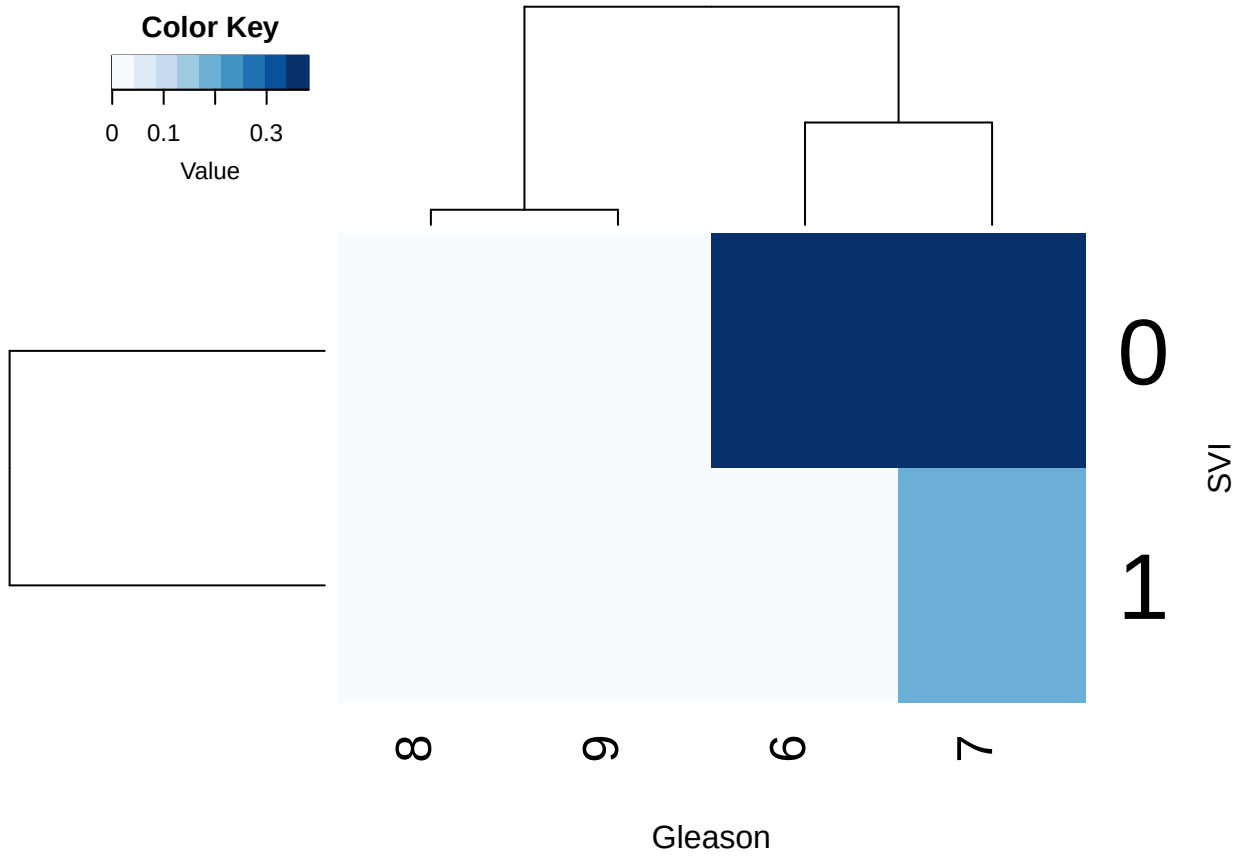
```
# Creamos una tabla con las dos variables
tabla <- table(svi, gleason)

# Creamos tablas de probabilidad por fila y por columna
addmargins(prop.table(tabla, 1), 2) * 100
```

```
##    gleason
## svi         6         7         8         9       Sum
##   0  46.052632  48.684211   1.315789   3.947368 100.000000
##   1   0.000000  90.476190   0.000000   9.523810 100.000000
```

```
addmargins(prop.table(tabla, 2), 1) * 100
```

```
##      gleason
## svi          6         7         8         9
##   0   100.00000  66.07143 100.00000  60.00000
##   1     0.00000  33.92857   0.00000  40.00000
##   Sum 100.00000 100.00000 100.00000 100.00000
```

```
# Realizamos un gráfico de la tabla para visualizar mejor la independencia
heatmap.2(
    prop.table(tabla),
    xlab = "Gleason", ylab = "SVI",
    density.info = "none",
    col = blues9,
    trace = "none"
)
```



Se puede ver en la gráfica que la mayoría de los casos se acumulan en zonas concretas:

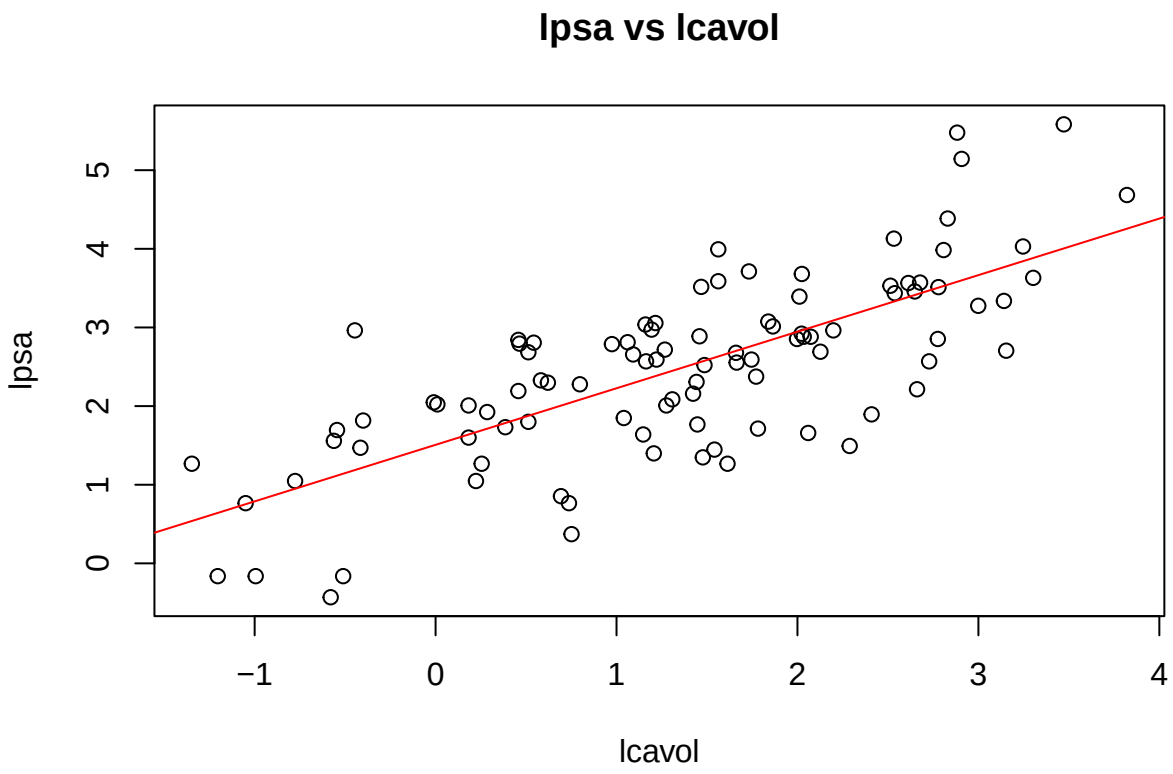
- Cuando SVI es 0, se acumulan en Gleason = 6 y 7.
- Cuando SVI es 1, se acumulan en Gleason = 7.

Por lo tanto, como los datos no se distribuyen por igual en todos los casos, las dos variables son dependientes.

4. Regresión lineal simple

El plot de los datos junto a la recta de regresión.

```
# Realizamos el modelo lineal
recta <- lm(lpsa ~ lcavol)

# Representamos el modelo sobre los datos
plot(lcavol, lpsa, main = "lpsa vs lcavol")
abline(recta, col = "red")
```



Intervalos de confianza para los coeficientes del modelo con una confianza de 0.95.

```
# Realizamos el intervalo de confianza al 95%
intervals <- confint(recta, level = 0.95)
intervals
```

```
##                 2.5 %    97.5 %
## (Intercept) 1.2652222 1.7493727
## lcavol      0.5839404 0.8547004
```

Definición de RSE y su valor.

Por definición, el RSE es la suma de los cuadrados de los residuos:

$$RSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}}$$

donde $y_i - \hat{y}_i$ son los residuos.

```
# Calculamos la suma de cuadrados de los residuos (RSE)
r1 <- residuals(recta)
RSE <- sqrt(sum(r1^2) / (dim(datos)[1] - 2))
RSE
```

```
## [1] 0.7874996
```

EL RSE vale 0.7874996.

Estudio de la eficacia del modelo.

```r
# Vemos el resumen del modelo
recta.summary <- summary(recta)
recta.summary
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.67624 -0.41648  0.09859  0.50709  1.89672
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.50730    0.12194   12.36   <2e-16 ***
## lcavol       0.71932    0.06819   10.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7875 on 95 degrees of freedom
## Multiple R-squared:  0.5394, Adjusted R-squared:  0.5346
## F-statistic: 111.3 on 1 and 95 DF,  p-value: < 2.2e-16
```

```r
# Calculamos el porcentaje de variación relativo de los intervalos de confianza respecto al valor predi
abs(intervals[1, 1] - intervals[1, 2]) / recta.summary$coefficients[1, 1] * 100
```

```
## [1] 32.12044
```

```r
abs(intervals[2, 1] - intervals[2, 2]) / recta.summary$coefficients[2, 1] * 100
```

```
## [1] 37.64109
```

```r
# Calculamos el porcentaje de error que hay respecto a la media de lpsa
RSE / mean(lpsa) * 100
```

```
## [1] 31.77468
```

Se puede ver en el modelo lineal que los p-value toman valores muy pequeños (menores a 2e-16). Además, el t-value toma valores mayores a 10 (10.55 y 12.36). Estas dos cosas parecen indicarnos que los coeficientes del modelo lineal no son nulos, por lo que las variables están relacionadas.

Sin embargo, el valor R^2 vale 0.5394, por lo que solo el 53.94% de la variación de lpsa es explicada por lcavol. El RSE es de 0.7875, que al compararlo con la media de lpsa, vemos que hay un 31.77% de error, que no es poco. Además, los intervalos de confianza son:

- (1.2652222, 1.7493727) para el valor de $\beta_0 = 1.50730$.
- (0.5839404, 0.8547004) para el valor de $\beta_1 = 0.71932$.

Si comparamos estos intervalos de confianza con los valores predichos de los coeficientes:

- Para el valor β_0 hay una variación del 32.12%.
- Para el valor β_1 hay una variación del 37.64%.

En resumen, los valores de p-value y t-value nos indican que las variables efectivamente tienen relación. Sin embargo, el resto de resultados del análisis nos dicen que el modelo lineal no es muy bueno para este caso, y que no tiene un gran desempeño.

Interpretación. El modelo lineal calculado, ¿cómo lo interpretas? Concretamente, ¿cómo a través del modelo lineal llegas a otro que relacionan las variables cavol y psa (sin los log. neperianos)?

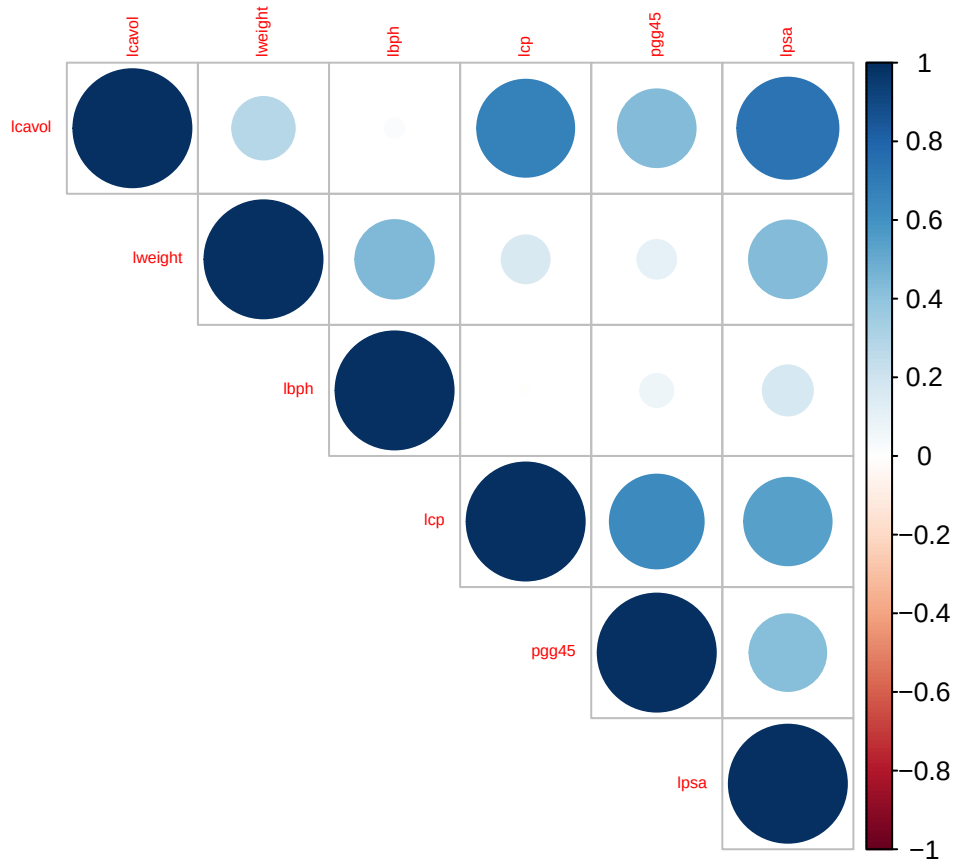
El modelo lineal tiene una pendietne positiva, lo que nos indica que la variable lpsa aumentará junto con lcavol. Dado que estas variables son logaritmos, podemos obtener una relación no lineal entre ellas haciendo la exponencial.

$$\ln(psa) = \beta_0 + \beta_1 \ln(cavol) \Rightarrow psa = e^{\beta_0 + \beta_1 \ln(cavol)} = e^{\beta_0} cavol^{\beta_1}$$

Podemos ver como a partir de la relación entre los logaritmos se ha obtenido una relación de potencias. En esta relación, la variable psa será una potencia de cavol con exponente β_1, y la variable β_0 intervendrá en la proporcionalidad de estas psa y la potencia de cavol. Como $\beta_0 > 0$, la constante de proporcionalidad será mayor que 1, y como β_1 está entre 0 y 1, la relación te potencias será como una raíz de valor $1/\beta_1 \approx 1.4$.

5. Regresión lineal multiple

```r
# Seleccionamos las columnas numéricas
num_cols <- which(sapply(datos, is.numeric))

# Hacemos un plot de la matriz de correlación
corrplot::corrplot(cor(datos[, num_cols]), type = "upper", tl.cex = 0.5)
```




```r
# Cremos un dataframe con solo los datos numéricos
datos.num <- datos[, c(-3, -7, -8)]

# Realizamos un modelo lineal entre lpsa y las variables numéricas
rectaMul <- lm(lpsa ~ ., data = datos.num)

# Vemos el modelo lineal
summary(rectaMul)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = datos.num)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.84878 -0.38372 -0.00413  0.45189  1.55468
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.36496    0.69866  -0.522  0.60269
## lcavol       0.54790    0.08637   6.343 8.56e-09 ***
## lweight      0.53036    0.19769   2.683  0.00867 **
## lbph         0.07999    0.05643   1.418  0.15971
## svi1         0.75975    0.24122   3.150  0.00221 **
## lcp         -0.03638    0.08088  -0.450  0.65391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7071 on 91 degrees of freedom
## Multiple R-squared:  0.6443, Adjusted R-squared:  0.6248
## F-statistic: 32.97 on 5 and 91 DF,  p-value: < 2.2e-16
```

Se puede ver en la matriz de correlación como algunas variables parecen tener cierta dependencia. En el caso de lpsa, parece estar relacionada con lcavol, lweight, lcp y en menor medida con pgg45. Si analizamos el p-value de los coeficientes, podemos ver que solo las variables svi1, lweight y lcavol tienen valores bajos, acompañados de t-values relativamente altos.

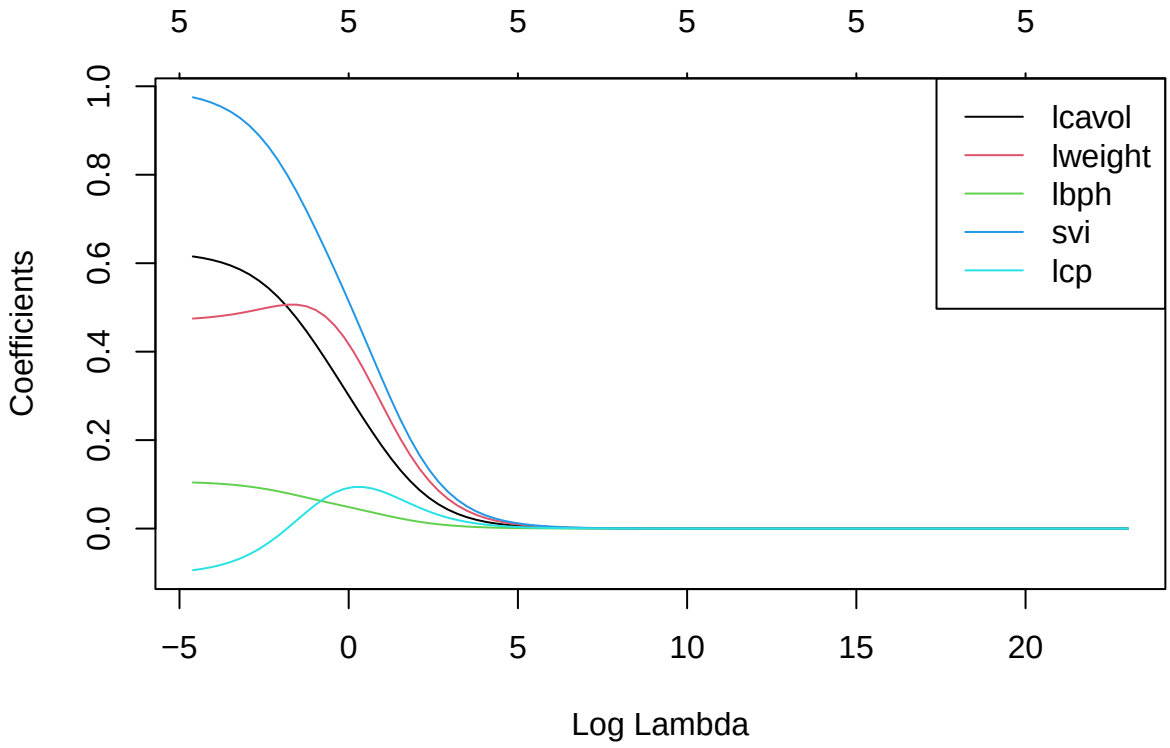
Sin embargo, los valores de R^2 y RSE son mejores que en el modelo lineal simple con lcavol, siendo de 0.6443 y 0.7071 respectivamente. Por lo tanto, podemos determinar que lpsa está relacionada con algunas variables y que el modelo lineal múltiple da mejores resultados que el simple, pero siguen siendo resultados con grandes errores. Además, se podrían eliminar algunas variables con las cuales no parece haber dependencia.

6. Modelo de Ridge y Lasso

Realizamos el modelo Ridge

```r
# Seleccionamos los datos para realizar los modelos
x <- model.matrix(lpsa ~ ., datos.num)[, -1]
y <- datos.num$lpsa

# Seleccionamos la semilla para los números aleatorios
set.seed(1)

# Seleccionamos los conjuntos de entrenamiento y test
train <- sample(seq(1, nrow(x)), nrow(x) / 2) # conjunto de entrenamiento
test <- (-train) # conjunto de testeo

# Guardamos los conjuntos de entrenamiento y test en variables
x.train <- x[train, ]
x.test <- x[test, ]
y.test <- y[test]
y.train <- y[train]

# Hacemos una malla con los valores de lambda
```

```r
malla <- 10^seq(10, -2, length = 100)
malla.ridge.train <- glmnet(x.train, y.train, alpha = 0, lambda = malla) # regresion ridge sin CV con c

# Representamos los valores de los coeficientes a lo largo del valor lambda
plot(malla.ridge.train, xvar = "lambda")
legend("topright", lty = 1, col = 2:ncol(datos.num) - 1, legend = names(datos.num[-ncol(datos.num)]))
```




```r
# Realizamos una regresión Ridge con CV
cv.out.ridge.train <- cv.glmnet(x.train, y.train, alpha = 0)

# Seleccionamos el mejor lambda
bestlam.ridge.train <- cv.out.ridge.train$lambda.min
bestlam.ridge.train
```

```
## [1] 0.1997304
```

```r
# Realizamos la regresión Ridge con CV y el mejor lambda
ridge.train <- glmnet(x.train, y.train, alpha = 0, lambda = bestlam.ridge.train)
coef(ridge.train)[, 1]
```

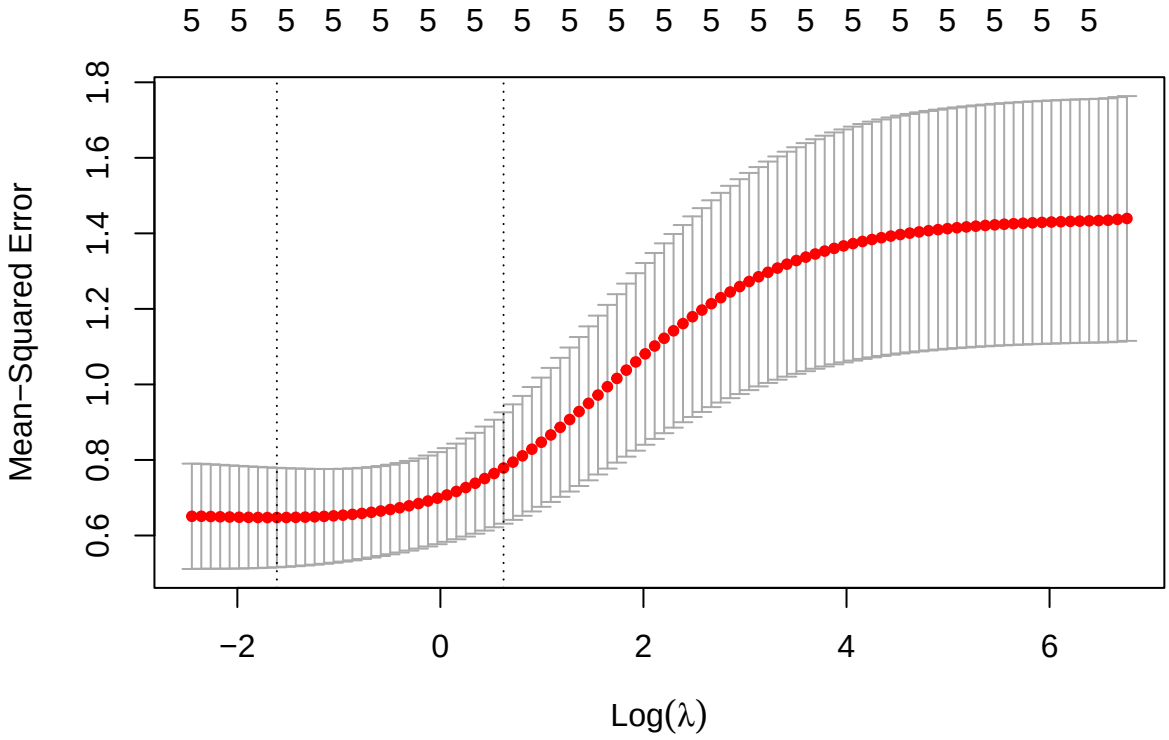
```
## (Intercept)      lcavol     lweight        lbph        svi1         lcp
## -0.15520758  0.48266384  0.50638792  0.07656639  0.77210598  0.01365333
```

```r
# Realizamos una regresión múltiple con los datos de entrenamiento
rectaMul.train <- glmnet(x.train, y.train, alpha = 0, lambda = 0)
coef(rectaMul.train)[, 1]
```

```
## (Intercept)      lcavol     lweight        lbph        svi1         lcp
```

```
##  -0.2924278   0.6263806   0.4693546   0.1068825   0.9934064  -0.1042145
```

```
# Realizamos una gráfica para ver el MSE de los ajustes para cada valor de lambda
plot(cv.out.ridge.train)
```




```
# Hacemos una predicción de Ridge para sacar los valores del MSE
ridge.pred <- predict(ridge.train, newx = x.test) # predicion de 'ridge.train' en conjunto de testeo
ridge.MSE <- mean((ridge.pred - y.test)^2) # MSE estimado de Ridge

# Hacemos una predicción el modelo de regresión lineal múltiple para sacar los valores del MSE
rectaMul.pred <- predict(rectaMul.train, newx = x[test, ]) # predicion de 'rectaMul.train' en conjunto
rectMul.MSE <- mean((rectaMul.pred - y.test)^2) # MSE estimado de rectaMul

# Comparamos los valores de los MSE con las medias de los valores de testeo
ridge.MSE / mean(y.test) * 100
```

```
## [1] 19.28006
```

```
rectMul.MSE / mean(y.test) * 100
```

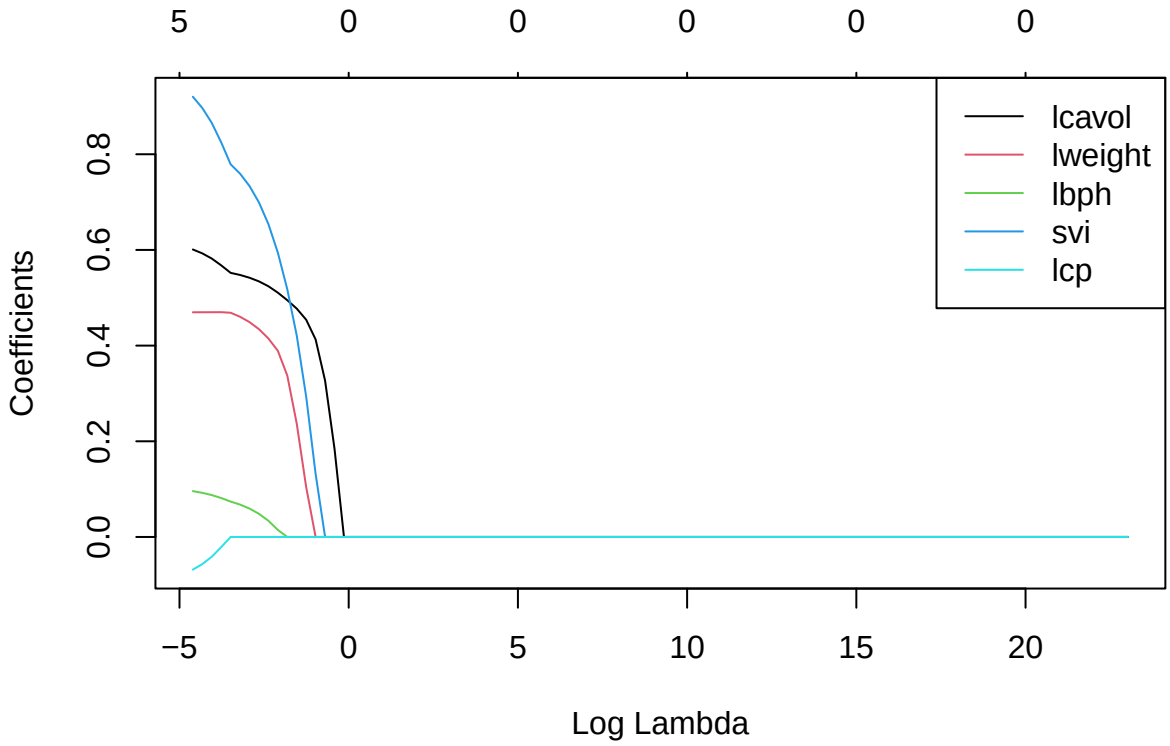
```
## [1] 20.71263
```

Tras realizar la validación cruzada, podemos ver como el mejor λ es $\lambda = 0.1997304$.

Al comparar el modelo Ridge con el λ óptimo y el modelo de regresión lineal múltiple, se observa cómo el valor de MSE es menor para el modelo de regresión Ridge. Comparando estos valores con la media de los valores del conjunto de testeo, se observa una mejora de aproximadamente un 1.4% (del 20.7% al 19.3%).

Realizamos el modelo Lasso

```r
# Realizamos una regresión LASSO sin CV para el conjunto de entrenamiento
malla.lasso.train <- glmnet(x.train, y.train, alpha = 1, lambda = malla)

# Representamos los valores de los coeficientes a lo largo del valor lambda
plot(malla.lasso.train, xvar = "lambda")
legend("topright", lty = 1, col = 2:ncol(datos.num) - 1, legend = names(datos.num[-ncol(datos.num)]))
```



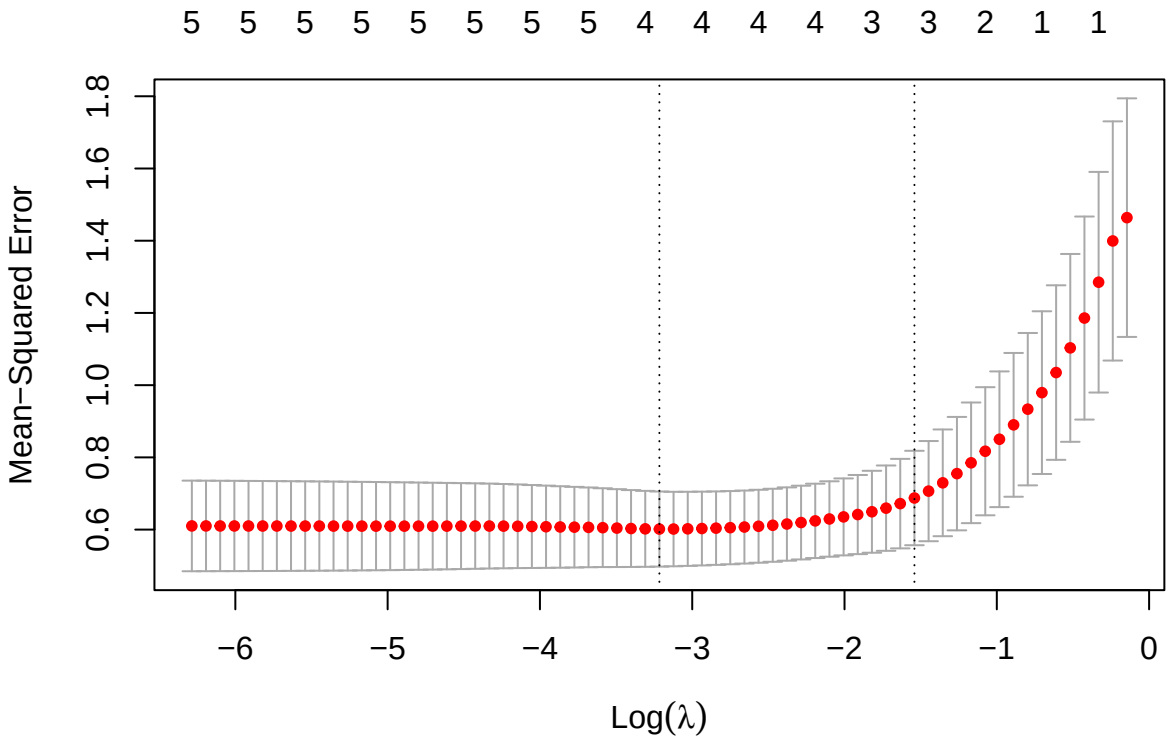

```r
# Realizamos una regresión LASSO con CV para el conjunto de entrenamiento
cv.out.lasso.train <- cv.glmnet(x[train, ], y[train], alpha = 1)

# Seleccionamos el mejor lambda (el que minimiza el MSE)
bestlam.lasso.train <- cv.out.lasso.train$lambda.min
bestlam.lasso.train
```

```
## [1] 0.0401305
```

```r
# Realizamos una regresión LASSO con el mejor lambda para el conjunto de entrenamiento
lasso.train <- glmnet(x[train, ], y[train], alpha = 1, lambda = bestlam.lasso.train)
coef(lasso.train)[, 1]
```

```
## (Intercept)      lcavol     lweight        lbph        svi1         lcp
## -0.07700425  0.54778095  0.46042266  0.06784849  0.75986294  0.00000000
```

```r
# Realizamos una gráfica para ver el MSE de los ajustes para cada valor de lambda
plot(cv.out.lasso.train) # MSE vs log(lambda)
```




```
# Hacemos una predicción del LASSO para sacar los valores del MSE
lasso.pred <- predict(lasso.train, newx = x[test, ]) # predicion de 'lasso.train' en conjunto de testeo
lasso.MSE <- mean((lasso.pred - y.test)^2) # MSE estimado de lasso
lasso.MSE
```

```
## [1] 0.4669899
```

```
lasso.MSE / mean(y.test) * 100
```

```
## [1] 19.54749
```

```
# Hacemos una predicción el modelo de regresión lineal múltiple para sacar los valores del MSE
rectaMul.pred <- predict(rectaMul.train, newx = x[test, ])
rectaMul.MSE <- mean((rectaMul.pred - y.test)^2) # MSE estimado de rectaMul
rectaMul.MSE
```

```
## [1] 0.494825
```

```
rectaMul.MSE / mean(y.test) * 100
```

```
## [1] 20.71263
```

Podemos observar que el mejor λ para realizar el ajuste es $\lambda = 0.04833733$.

Al comparar el modelo LASSO con el λ óptimo y el modelo de regresión lineal múltiple, se observa cómo el valor de MSE es menor para el modelo de regresión LASSO. Comparando estos valores con la media de los valores del conjunto de testeo, se observa una mejora de aproximadamente un 1.2% (del 20.7% al 19.5%).

Se puede apreciar como Ridge ha dado un valor de MSE un poco menor que LASSO. Sin embargo, ambos modelos tienen un buen desempeño con resultados muy similares.

7. LDA

```
# Realizamos el modelo LDA
lda <- lda(svi ~ lcavol + lcp + lpsa)
lda
```

```
## Call:
## lda(svi ~ lcavol + lcp + lpsa)
##
## Prior probabilities of groups:
##         0         1
## 0.7835052 0.2164948
##
## Group means:
##     lcavol        lcp     lpsa
## 0 1.017892 -0.6715458 2.136592
## 1 2.551959  1.6018579 3.715360
##
## Coefficients of linear discriminants:
##                LD1
## lcavol -0.08659598
## lcp     0.76640382
## lpsa    0.53153340
```

```
# Realizamos una predicción sobre el resultado de LDA
lda.pred <- predict(lda)

# Realizamos una tabla de confusión con LDA y SVI
predicted <- lda.pred$class
tabla.confusion.lda <- table(predicted, svi) # tabla de confusion
tabla.confusion.lda <- round(addmargins(prop.table(tabla.confusion.lda)) * 100, 2)
tabla.confusion.lda
```

```
##          svi
## predicted      0      1    Sum
##       0    71.13   4.12  75.26
##       1     7.22  17.53  24.74
##       Sum  78.35  21.65 100.00
```

```
# Valores acertados
tabla.confusion.lda[1, 1] / tabla.confusion.lda[3, 1] * 100 # Valores acertados de SVI = 0
```

```
## [1] 90.78494
```

```
tabla.confusion.lda[2, 2] / tabla.confusion.lda[3, 2] * 100 # Valores acertados de SVI = 1
```

```
## [1] 80.96998
```

```
# Valores errados
tabla.confusion.lda[1, 2] / tabla.confusion.lda[1, 3] * 100 # Valores errados de SVI = 0
```

```
## [1] 5.474356
```

```
tabla.confusion.lda[2, 1] / tabla.confusion.lda[2, 3] * 100 # Valores errados de SVI = 1
```

```
## [1] 29.18351
```

Se puede ver en los análisis de la tabla de confusión que el modelo LDA acierta:

- Un 90.8% de las veces cuando SVI = 0.
- Un 81.0% de las veces cuando SVI = 1.

Por otro lado, se ve que el modelo LDA falla:

- Un 5.5% de las veces cuando SVI = 0.
- Un 29.2% de las veces cuando SVI = 1.

Por lo tanto, es un buen modelo a la hora de acertar, pero tiene una gran probabilidad de falso positivo en SVI = 1, prácticamente del 30%.

8. Regresión Logística

```
# Realizamos la regresión logística
lr <- glm(svi ~ lcavol + lcp + lpsa, family = binomial)

# Calculamos las probabilidades del modelo
lr.probs <- predict(lr, type = "response")

# Calculamos las predicciones del modelo
lr.pred <- rep(0, 97)
lr.pred[lr.probs > .5] <- 1

# Hacemos las tablas
tabla.confusion.lr <- prop.table(table(lr.pred, svi)) * 100
tabla.confusion.lr <- round(addmargins(tabla.confusion.lr), 2)
tabla.confusion.lr
```

```
##        svi
## lr.pred      0      1    Sum
##     0    75.26   6.19  81.44
##     1     3.09  15.46  18.56
##     Sum  78.35  21.65 100.00
```

```
# Realizamos los análisis de la tabla de confusión
tabla.confusion.lr[1, 1] / tabla.confusion.lr[3, 1] * 100 # Valores acertados de SVI = 0
```

```
## [1] 96.05616
```

```
tabla.confusion.lr[2, 2] / tabla.confusion.lr[3, 2] * 100 # Valores acertados de SVI = 1
```

```
## [1] 71.40878
```

```
tabla.confusion.lr[1, 2] / tabla.confusion.lr[1, 3] * 100 # Valores errados de SVI = 0
```

```
## [1] 7.600688
```

```
tabla.confusion.lr[2, 1] / tabla.confusion.lr[2, 3] * 100 # Valores errados de SVI = 1
```

```
## [1] 16.64871
```

```
# Probabilidad de svi = 1 con lcabol = 2.8269, lcp = 1.843, lpsa = 3.285
predict(lr, newdata = data.frame(lcavol = 2.8269, lcp = 1.843, lpsa = 3.285), type = "response")
```

```
##         1
## 0.7710017
```

El modelo de regresión logística acierta:

- Un 96.1% de las veces cuando SVI = 0.

- Un 71.4% de las veces cuando SVI = 1.

Por otro lado, el modelo de regresión logística falla:

- Un 7.6% de las veces cuando SVI = 0.
- Un 16.6% de las veces cuando SVI = 1.

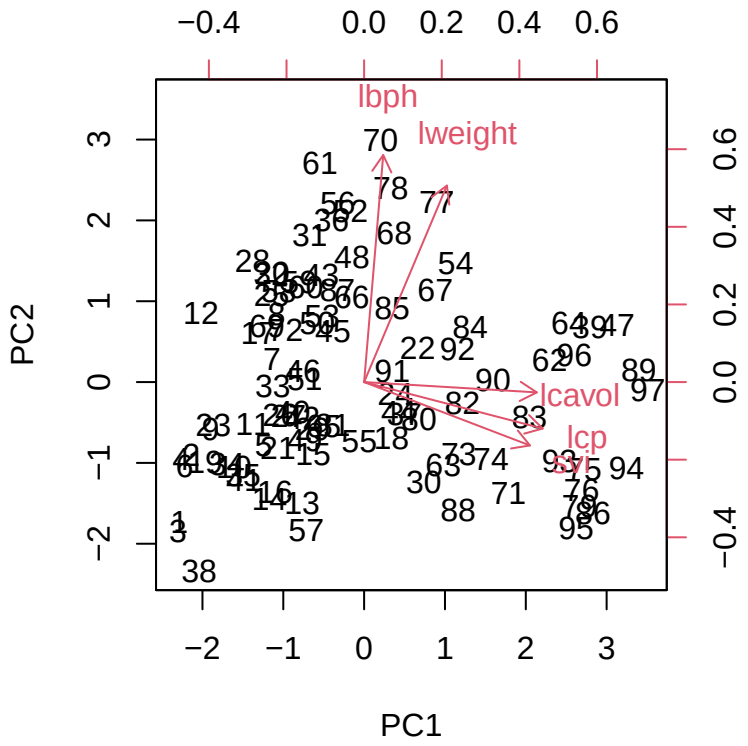
Se puede apreciar como el modelo es bueno para acertar en SVI = 0, y relativamente bueno para acertar en SVI = 1. Además, tiene mucho menos probabilidad de falso positivo que el modelo LDA, teniendo una probabilidad del 7.6% de falso positivo en SVI = 0 y 16.6% en SVI = 1.

La probabilidad de SVI = 1 con lcavol = 2.8269, lcp = 1.843 y lpsa = 3.285 es del 77.1%.

9. PCA-PCR

Un biplot y summary del modelo. Coméntalos.

```
# Convertimos los datos SVI en numéricos
datos.num$svi <- as.numeric(datos$svi)

# Seleccionamos las variables con las que hacer el PCA
variables_pca <- c("lcavol", "lweight", "lbph", "lcp", "svi")

# Seleccionamos los datos con las variables para el PCA
datos_pca <- datos.num[, variables_pca]

# Realizamos el PCA
pr.out <- prcomp(datos_pca, scale = TRUE)

# Representamos el PCA con un biplot
biplot(pr.out, scale = 0)
```




```
# Hacemos un resumen del PCA
summary(pr.out)
```

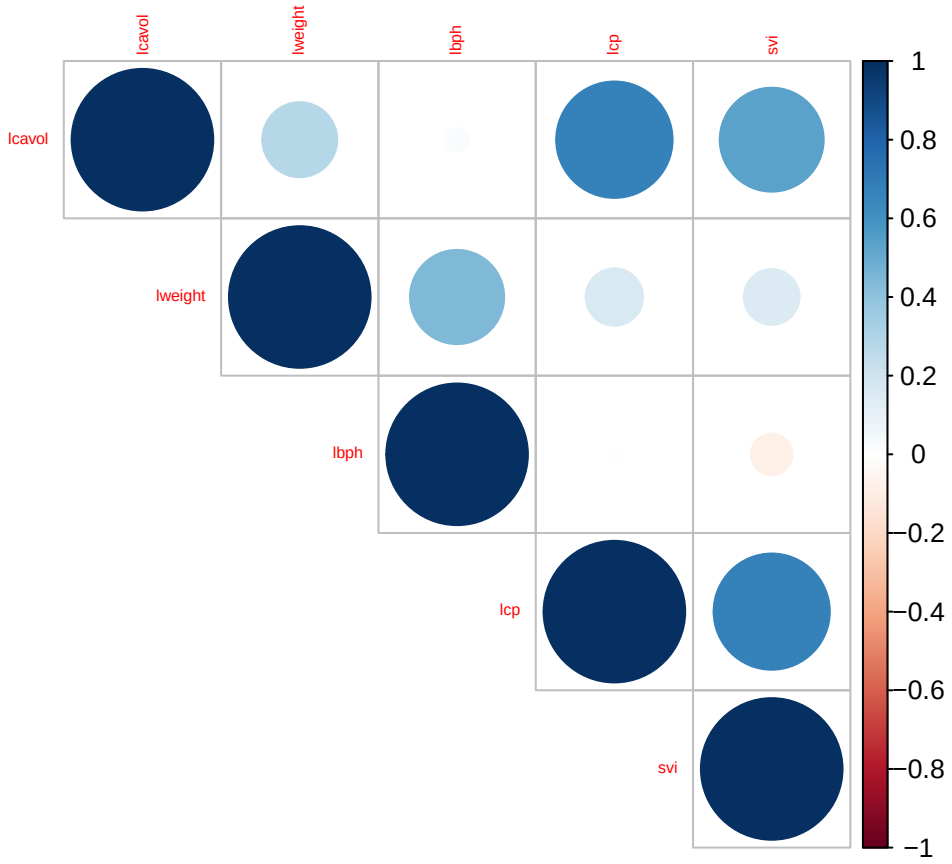
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5
## Standard deviation     1.5341 1.1862 0.7258 0.66771 0.51651
## Proportion of Variance 0.4707 0.2814 0.1054 0.08917 0.05336
## Cumulative Proportion  0.4707 0.7521 0.8575 0.94664 1.00000
```

En el biplot vemos dos grupos de flechas. Vemos que los vectores de lbph y de lweight están relativamente cerca, apuntando hacia la dirección de PC2 con unos grados de desviación en sentido horario. Por otro lado, tenemos el grupo de lcavol, lcp y svi, que estan cerca unos de otros apuntando positivamente en el eje de PC1, con unos grados de desviación en sentido horario. La variable que más alineada está con PC1 es lcavol, y la más alineada con PC2 es lbph. Que varios vectores de variables estén juntos indican que estas están correlacionadas entre sí, por lo que lpbh y lweight tinen relación entre sí, al igual que lcavol, lcp y svi, pero estos dos grupos no están correlacionados.

Si analizamos los resultados del `summary`, vemos que PC1 explica el 47% de la varianza, y que PC2 explica el 28%, siendo por tanto las variables más importantes del PCA. Con estas dos variables se puede explicar el 75% de la varianza. El hecho de que PC1 explique casi la mitad de la varianza, hace que las variables que están relacionadas con PC1 (lcavol, lcp y svi) sean las más importantes (sobre todo lcavol).

Justificar la desviación estándar del primer componente principal.

```
# Representamos la matriz de correlación con los datos del PCA
corrplot::corrplot(cor(datos_pca), type = "upper", tl.cex = 0.5)
```




```
# Mostramos la desviación estándar del PCA
pr.out$sdev
```

```
## [1] 1.5341177 1.1861896 0.7258177 0.6677143 0.5165108
```

```
# Calculamos la proporcion de la varianza de la componente PC1
1.5341^2 / (1.5341^2 + 1.1862^2 + 0.7258^2 + 0.66771^2 + 0.51651^2)
```

```
## [1] 0.4706984
```

Se puede ver como el cálculo de la desviación estándar de la componente PC1 es correcto. Como se menciona anteriormente, esta componente explica casi el 50% de toda la varianza.

Si ahora nos fijamos en la matriz de correlación, vemos como efectivamente las variables lcavol, lcp y svi tienen una correlación alta. Como estas tres variables están áltamente correlacionadas y apuntan en la dirección de PC1, no es de extrañar que la varianza de PC1 sea la más alta.

Tras la proporción de varianza acumulada, ¿cuáles son las componentes principales que reflejan el 80% de la varianza total de los datos?

```
# Hacemos un resumen del PCA
summary(pr.out)
```

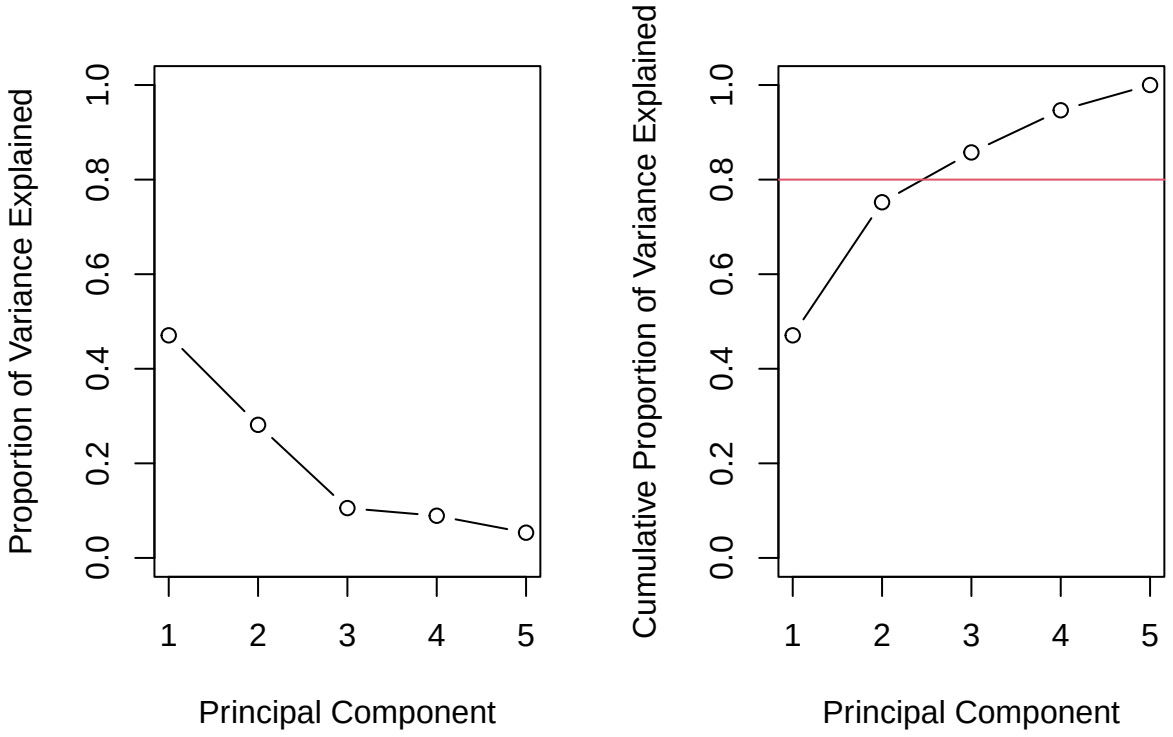
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5
## Standard deviation     1.5341 1.1862 0.7258 0.66771 0.51651
## Proportion of Variance 0.4707 0.2814 0.1054 0.08917 0.05336
## Cumulative Proportion  0.4707 0.7521 0.8575 0.94664 1.00000
```

Se puede ver en el resumen como las tres primeras componentes principales (PC1, PC2 y PC3) reflejan más

del 80% de la varianza total de los datos (el 85.75%).

También se pueden observar estas proporciones de manera más visual.

```
# Proportion of Variance Explained
pve <-  pr.out$sdev^2/sum(pr.out$sdev^2)

# Creamos dos gráficas
par(mfrow = c(1, 2))

# Gráfica con Proportion of Variance Explained
plot(pve, xlab = "Principal Component",
ylab = "Proportion of Variance Explained", ylim = c(0, 1),
type = "b")

# Gráfica con Cumulative Proportion of Variance Explained
plot(cumsum(pve), xlab = "Principal Component",
ylab = "Cumulative Proportion of Variance Explained",
ylim = c(0, 1), type = "b")
abline(h = 0.8,col = 2)
```



Se aprecia también en la segunda gráfica que son los tres primeros componentes principales los que captan más del 80% de la varianza total de los datos.

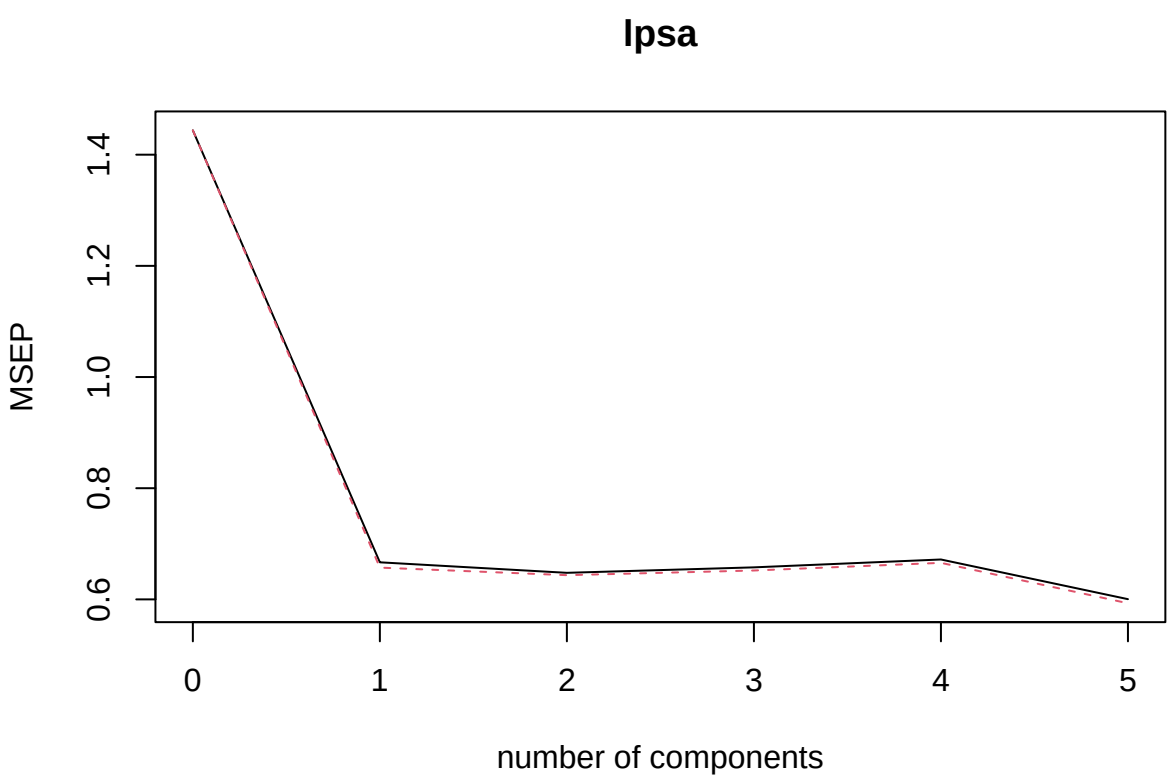
Además, aplica PCR para predecir lpsa teniendo en cuenta las variables lcavol, lweight, lbph, lcp, svi. ¿Qué conclusiones podrías sacar?¿Este modelo es mejor que el del apartado 5?

```
# Creamos variables con los datos
x.pcr <- model.matrix(lpsa ~ ., datos.num)[, -1] # datos de entrada
```

```r
y.pcr <- datos.num$lpsa # datos de salida

# Creamos un conjunto de testeo
y.pcr.test <- y.pcr[test]

# Hacemos una regresión de componentes principales (PCR) con el conjunto de entrenamiento
pcr.fit <- pcr(lpsa ~ ., data = datos.num, subset = train, scale = TRUE, validation = "CV")

# Hacemos un resumen de la regresión
summary(pcr.fit)
```

```
## Data:    X dimension: 48 5
##  Y dimension: 48 1
## Fit method: svdpc
## Number of components considered: 5
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
## CV           1.202   0.8165   0.8048   0.8109   0.8196   0.7748
## adjCV        1.202   0.8106   0.8022   0.8074   0.8159   0.7701
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps
## X       45.39    73.59    86.12    95.07   100.00
## lpsa    57.08    58.26    59.83    59.86    66.22
```

```r
# Vemos el error en funcion del numero de componentes principales
validationplot(pcr.fit, val.type = "MSEP")
```




```
# Hacemos predicciones con el conjunto de testeo y calculamos el MSE
pcr.pred <- predict(pcr.fit, x.pcr[test, ], ncomp = 5) # predecimos con conjunto de test
pcr.MSE <- mean((pcr.pred - y.pcr.test)^2) # comparamos resultado con datos de salida de test

pcr.MSE
```

```
## [1] 0.494787
```

```
ridge.MSE
```

```
## [1] 0.4606008
```

```
lasso.MSE
```

```
## [1] 0.4669899
```

```
rectaMul.MSE
```

```
## [1] 0.494825
```

Se puede ver en el gráfico como el error del PCR es menor conforme se aumenta el número de componentes, siendo su mínimo en 5 componentes.

Si comparamos el MSE de las predicciones con n = 5 con los modelos anteriores, vemos que Ridge y LASSO tienen el MSE más bajo (alrededor de 0.46). Por otro lado, la regresión múltiple y el PCR tienen valores de MSE más altos, alrededor de 0.49.

En conclusión, los modelos que mejor predicen las observaciones son Ridge y LASSO. Mientras, la regresión múltiple y el PCR tienen resultados muy parecidos, con MSE más altos.