Regresión, modelos y métodos - PEC 2

Alejandro Keymer

Ejercicio 1 (25 pt.) Mecanismo neural de la nocicepción.

(a)

En una primera aproximación para comparar los porcentajes de reacción de los gusanos mutantes con los salvajes, nos planteamos hacer un test de comparación de dos muestras independientes y un gráfico adecuado, sin tener en cuenta los pulsos. Para ello debemos observar que la variable respuesta es un porcentaje y el test t de Student habitual no sirve. Entre las posibles soluciones tenemos las siguientes:

1. **Aplicar un test de Welch a los datos transformados con la función normalizante $\arcsin(\sqrt{p})$.**
2. **Hacer un test no paramétrico como el de Mann-Whitney-Wilcoxon.**
3. **Aplicar un test de permutaciones para comparar las dos muestras.**
4. **Realizar una regresión logística binomial.**
5. **Calcular una regresión beta que es especialmente apropiada para proporciones.**

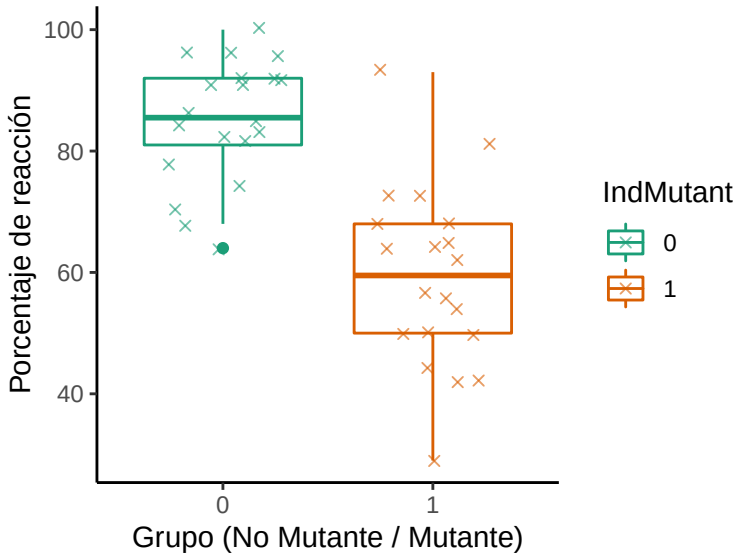
Las dos últimas propuestas son las más acertadas, ya que las otras contemplan comparar medias o medianas de porcentajes. En todo caso, realizar tres de las cinco soluciones con una de ellas entre las dos últimas y comentar su resultado.

En primer lugar cargamos la base da datos a trabajar.

```
df <-
    read_csv2("C_elegans.csv") %>%
    mutate(p_reac = Perc_Reacting/100,
           IndMutant = factor(IndMutant))
```

Una vez cargada la base de datos, hacemos un gráfico de cajas entre los dos grupos de gusanos, para valorar la diferencia entre los porcentajes de reacción.

```
# boxplot de los datos
ggplot(df, aes(x = IndMutant, y = Perc_Reacting, color = IndMutant)) +
    geom_boxplot() + labs(x="Grupo (No Mutante / Mutante)", y = "Porcentaje de reacción") +
    geom_jitter(width = .3, shape = 4, alpha = .6) +
    theme_classic() + scale_color_brewer(type = "qual", palette = 2)
```



En el gráfico se puede observar que visualmente **si** que se aprecian diferencias entre los dos grupos.

A continuación se plantean diferentes *tests* para valorar si esta diferencia es significativa o no.

i) Regresión logística binomial.

```
mod_null <-
    glm(IndMutant ~ 1, df, family = binomial(link = "logit"))
mod_log <-
    glm(IndMutant ~ Perc_Reacting, df, family = binomial(link = "logit"))

pander(mod_log)
```

Table 1: Fitting generalized (binomial/logit) linear model: IndMutant ~ Perc_Reacting

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	11.07	3.343	3.313	0.0009244
Perc_Reacting	-0.151	0.04483	-3.369	0.000755

```
anova(mod_null, mod_log, test = "Chisq") %>%
    pander(caption = "ANOVA de modelo Nulo v/s Regresión Logística")
```

Table 2: ANOVA de modelo Nulo v/s Regresión Logística

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
39	55.45	NA	NA	NA
38	28.23	1	27.22	1.812e-07

En la regresión logística se aprecia que existe una diferencia entre los dos grupos, que resulta significativo. Esto se puede observar con la prueba de χ^2 la que permite rechazar la $H0$.

ii) Test de permutaciones.

```r
mod <- coin::independence_test(p_reac ~ IndMutant, df)


tibble(
    z = coin::statistic(mod),
    p = coin::pvalue(mod))
```

z	p
4.49297	7.02367e-06

Al realizar el test de permutaciones se confirma lo anterior en cuanto a que según el resultado se puede rechazar la $H0$ de igualdad de medias entre los dos grupos.

iii) Test de Mann-Whitney-Wilcoxon.

```r
mod <- wilcox.test(Perc_Reacting ~ IndMutant, df, exact = F)
pander(mod)
```

Table 4: Wilcoxon rank sum test with continuity correction: `Perc_Reacting` by `IndMutant`

Test statistic	P value	Alternative hypothesis
368	5.733e-06 * * *	two.sided

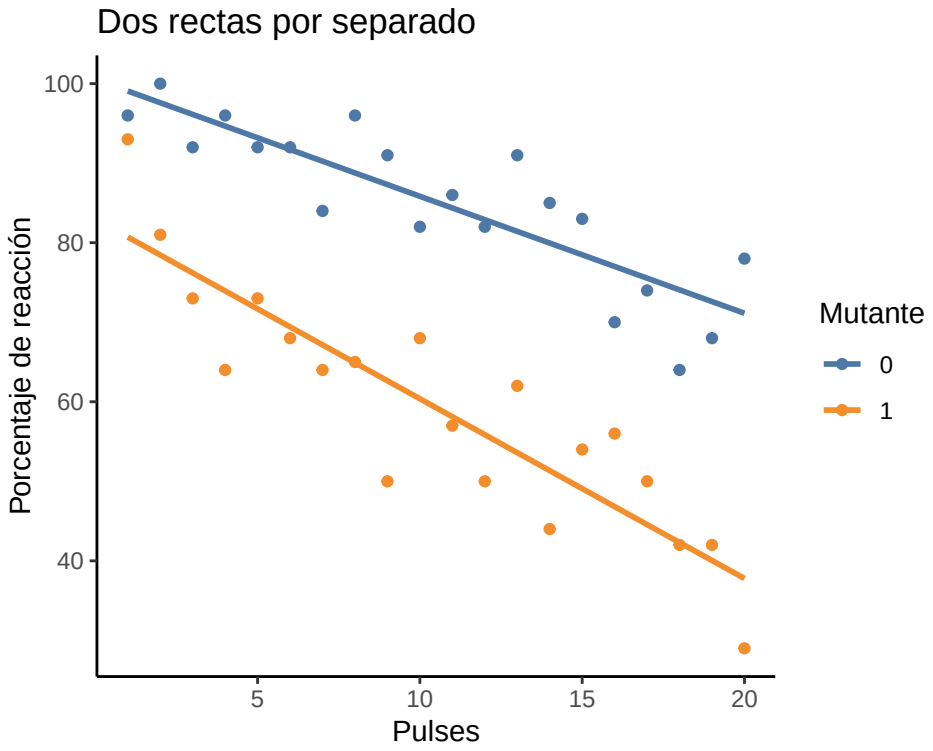
El test de rango de Mann-Whitney-Wilcoxon también permite establecer que se puede rechazar la $H0$ entre los dos grupos, con un grado de significación alto.

Las tres pruebas no paramétricas son coherentes entre si, lo que afirma la conclusión de que se puede rechazar la $H0$ de igualdad de medias entre los grupos de Mutantes y no Mutantes, de manera significativa.

(b)

Dibujar el gráfico de dispersión del porcentaje de reacción en función del número de pulsos de luz con dos rectas que representen las dos sub-poblaciones por separado, sin modelo común. Describir la forma de la relación. ¿Qué sugiere sobre el patrón de respuestas de abstinencia en las dos sub-poblaciones de *C. elegans* para un número creciente de pulsos de luz? ¿Cómo encaja su respuesta en el contexto de habituación? Explique por qué estos resultados sugieren una participación de las neuronas sensoriales de PVD en la nocicepción y la habituación. Nota: En este apartado consideramos la variable respuesta el porcentaje de reacción sin ninguna transformación.

```r
# las recta de regresión se puden graficar con la función 'geom_smooth()' y el
# método de 'lm' que corresponde al modelo lineal.
ggplot(df, aes(Pulses,Perc_Reacting,  color = IndMutant)) +
    geom_point() +
    geom_smooth(method = "lm", se= F) +
    labs(title = "Dos rectas por separado", y = "Porcentaje de reacción", color = "Mutante") +
    theme_classic() +
    scale_color_tableau()
```



En el gráfico de las dos poblaciones se puede observar que, por una parte, ambas curvas muestran una relación inversa entre el Porcentaje de reacción y la cantidad de Pulsos. Esta relación sugiere que ambas poblaciones de *C.Elegans* presentan respuestas de adaptación en la respuesta a los pulsos de luz, es decir, que frente al estímulo repetitivo, se produce una habituación al estímulo y la respuesta va siendo cada vez menor. Por otra parte es posible apreciar que la población de **mutantes** presenta un *intercepto* algo menor que la población **no mutante**, que traduce la mayor lentitud de las neuronas sensoriales en procesar el estímulo. Se observa además que la pendiente de ambas curvas es relativamente similar, lo que podría indicar que la velocidad de habituación en ambas poblaciones es muy similar.

```
mod_0 <- lm(Perc_Reacting ~ Pulses, filter(df, IndMutant == 0))
mod_1 <- lm(Perc_Reacting ~ Pulses, filter(df, IndMutant == 1))

# modelo de NO MUTANTES
pander(mod_0)
```

Table 5: Fitting linear model: Perc_Reacting ~ Pulses

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	100.5	2.491	40.37	4.135e-19
Pulses	-1.471	0.2079	-7.073	1.351e-06

```
# modelo de MUTANTES
pander(mod_1)
```

Table 6: Fitting linear model: Perc_Reacting ~ Pulses

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	82.97	3.242	25.59	1.318e-15
Pulses	-2.259	0.2707	-8.347	1.331e-07

(c)

Proponer un modelo lineal que permita estudiar la asociación de la variable respuesta *porcentaje de reacción* (sin ninguna transformación) con la variable `Pulses` y el factor `IndMutant`. ¿Cual es la ecuación del modelo final propuesto? ¿Es un modelo significativo? ¿Qué tanto por ciento de la variabilidad del porcentaje de reacción queda explicado por el modelo? ¿Todos los coeficientes de las variables explicativas del modelo son significativos? Dibujar el gráfico de dispersión del apartado anterior pero con las rectas resultantes del modelo.

```
# modelo que incluye la interacción entre la variable cualitativa y el factor.
# rectas con diferente pendiente.
mod_int <- lm(Perc_Reacting ~ Pulses * IndMutant, df)

# modelo que no incluye la interacción. Rectas paralelas
mod_no_int <- lm(Perc_Reacting ~ Pulses + IndMutant, df)

pander(mod_int)
```

Table 7: Fitting linear model: Perc_Reacting ~ Pulses * IndMutant

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	100.5	2.891	34.78	2.66e-29
Pulses	-1.471	0.2413	-6.094	5.212e-07
IndMutant1	-17.57	4.089	-4.297	0.0001257
Pulses:IndMutant1	-0.7887	0.3413	-2.311	0.02668

```
anova(mod_int, mod_no_int) %>%
    pander("modelo con interacción v/s modelo sin interaccion")
```

Table 8: modelo con interacción v/s modelo sin interaccion

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
36	1394	NA	NA	NA	NA
37	1601	-1	-206.8	5.34	0.02668

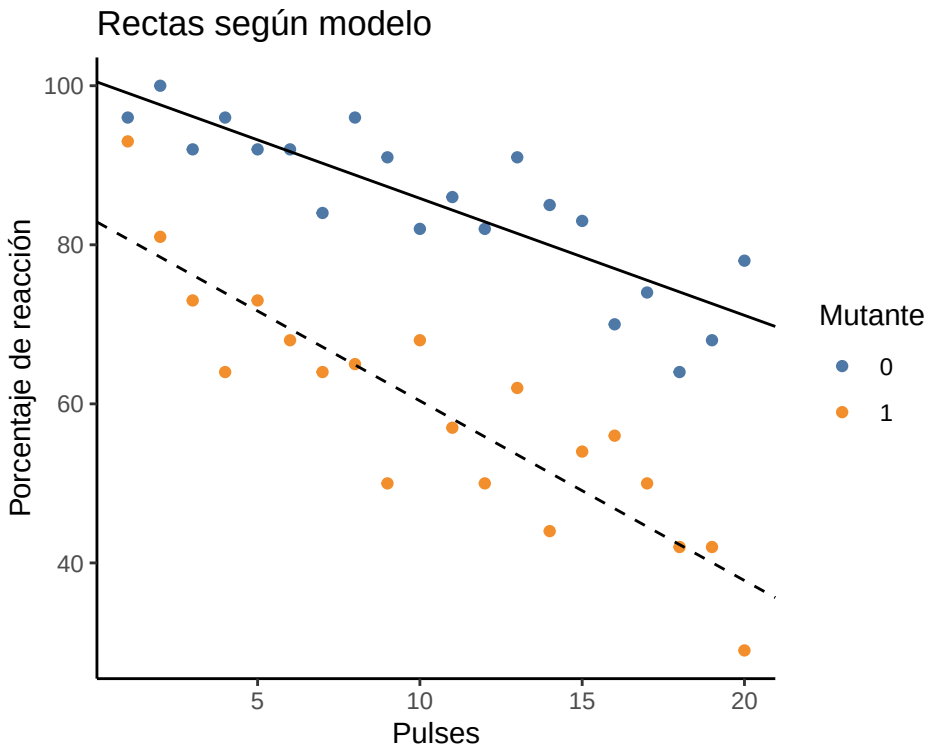
En primer lugar creamos el modelo que incluye el predictor categorial `IndMutant` y con interacción de este con `Pulses`. El modelo presenta un valor alto de F que resulta significativo lo que permite rechazar la $H0$. Al mirar los predictores es posible ver que tanto `Pulses`, `IndMutant` como la interacción de estos dos resulta significativa.

Podemos corroborar además la misma información si hacemos el modelo que NO incluye la interacción, y luego comparamos ambos modelos con un `anova()`. El valor de p del estadístico F de esta prueba, es el mismo valor de significación de la interacción del modelo original. Esto permite asumir que el modelo más válido es el que contempla la interacción del factor con la variable cualitativa.

El valor de R^2 refleja el porcentaje de variabilidad que explica el modelo. En este caso el modelo explica un 88.3 % de la variabilidad de la respuesta.

```
ggplot(df, aes(Pulses, Perc_Reacting, color = IndMutant)) +
    geom_point() +
```

```
    geom_abline(intercept = coef(mod_int)[1],
                slope = coef(mod_int)[2],
                linetype = 1) +
    geom_abline(intercept = (coef(mod_int)[1] + coef(mod_int)[3]),
                slope = (coef(mod_int)[2] + coef(mod_int)[4]),
                linetype = 2) +
    labs(title = "Rectas según modelo", y = "Porcentaje de reacción", color = "Mutante") +
    scale_color_tableau() +
    theme_classic()
```



Al graficar el modelo completo se puede observar que es similar al primer gráfico, con las rectas independientes. Esto tiene sentido en la medida que el modelo contempla el efecto de los pulsos, el del factor `IndMutant` y además la interacción entre ellos, lo que permite tener estas dos rectas con *pendientes diferentes*.

(d)

Ahora contestaremos las preguntas y dibujaremos el gráfico del apartado anterior, si realizamos la transformación normalizante $\arcsin\sqrt{p}$ **sobre la variable respuesta. Hallar el intervalo de confianza al 95 % para la media del porcentaje de mutantes que reaccionan a 10 pulsos de luz en condiciones experimentales parecidas. Nota: Habrá que deshacer la transformación para dibujar "las rectas".**

```
df_adj <-
    df %>%
    mutate(p_arcsin = asin(sqrt(p_reac)))

mod_arc_int <- lm(p_arcsin ~ Pulses * IndMutant, df_adj)
mod_arc_no_int <- lm(p_arcsin ~ Pulses + IndMutant, df_adj)
```

```
pander(mod_arc_int)
```

Table 9: Fitting linear model: p_arcsin ~ Pulses * IndMutant

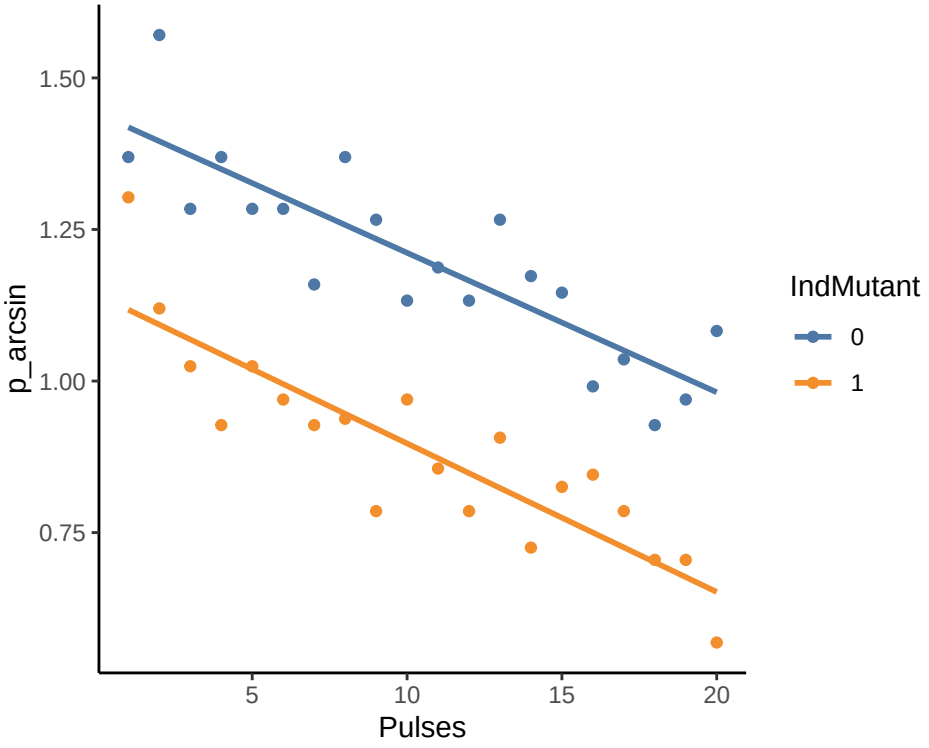
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.442	0.03831	37.63	1.677e-30
Pulses	-0.023	0.003198	-7.192	1.839e-08
IndMutant1	-0.2995	0.05418	-5.527	2.977e-06
Pulses:IndMutant1	-0.001499	0.004523	-0.3314	0.7423

```
pander(mod_arc_no_int)
```

Table 10: Fitting linear model: p_arcsin ~ Pulses + IndMutant

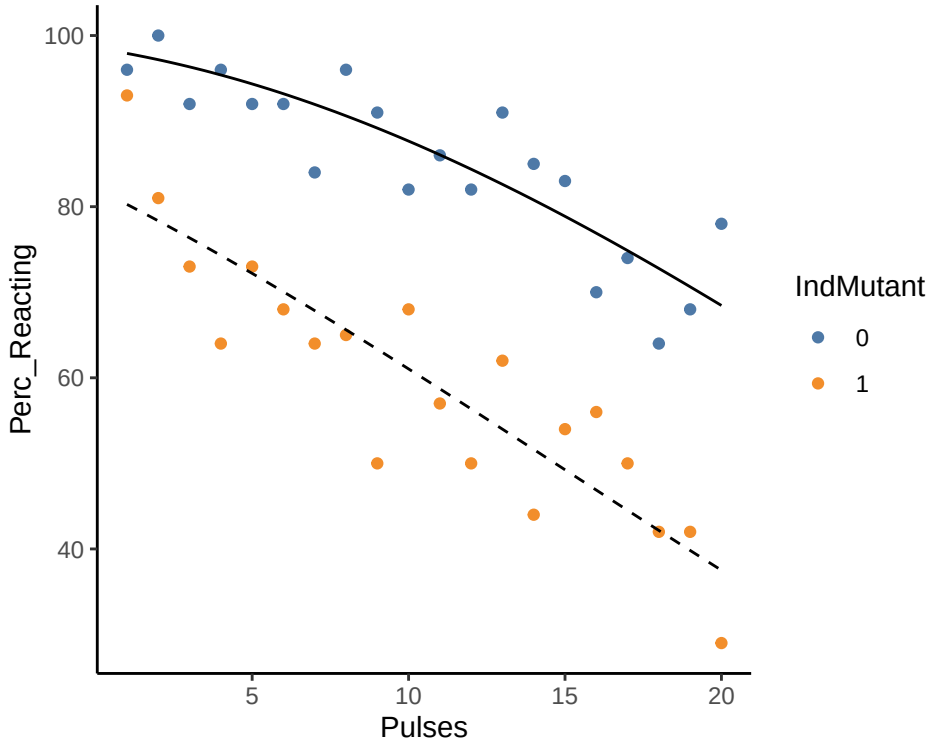
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.449	0.0297	48.8	3.421e-35
Pulses	-0.02375	0.002234	-10.63	8.417e-13
IndMutant1	-0.3152	0.02576	-12.23	1.434e-14

```
# Gráfico en dimensiones transformadas
df_adj %>%
  ggplot(aes(Pulses, p_arcsin, color = IndMutant)) +
  geom_point() +
  geom_smooth(method = "lm", fullrange=T, se = F) +
  scale_color_tableau() +
  theme_classic()
```



Como en el ejemplo anterior, comenzamos realizando dos modelos; un modelo que considere la interacción entre el factor `IndMutant` y la variable cuantitativas `Pulses`, y otro que no. En este caso se puede observar que la interacción deja de ser significativa. De esta manera, en el caso del modelo con la variable repuesta transformada, se puede simplificar quitando el término de la interacción. Lo anterior se puede ver de manera visual al realizar el gráfico de dispersión en que se puede ver como ambas rectas en este caso, tienen una pendiente muy similar, y casi paralela.

```
# gráfico en dimensiones originales.
cf <- coef(mod_arc_no_int)

df %>%
    ggplot(aes(Pulses, Perc_Reacting, color = IndMutant)) +
    geom_point() +
    stat_function(fun = ~ sin(cf[1] + cf[2] * .x)^2 * 100,
                  color = "black") +
    stat_function(fun = ~ sin(cf[1] + cf[3] + cf[2] * .x)^2 * 100,
                  color = "black", linetype=2) +
    scale_color_tableau() +
    theme_classic()
```



Al graficar las curvas en el espacio original, e posible ver que el modelo se adapta a una relación no lineal de la variable `Pulses` con la variable `Perc_Reacting`. Esta relación se vuelve lineal al utilizar la transformación.

Con todo lo anterior se podría concluir que la interacción entre el grupo con la variable de `Pulses` no tendría mayor importancia. Se podría decir que según este modelo, no hay evidencia de que los grupos de gusano difieran en la rapidez en la que se habitúan al estímulo lumínico, sino que mas bien, el grupo de mutante reacciona menos. La diferencia en al habituación, que se observa en la sección (a) sería un sesgo dado por la distribución de la variable `Perc_Reacting`, que al ser una proporción, se aleja de una distribución normal.

```
# predicción
new_df <- tibble(Pulses = 10, IndMutant = "1")
ypred <- predict.lm(mod_arc_no_int, new_df, interval = "conf")
```

```r
#predicción de la respuesta en la escala original
pander(sin(ypred)^2 * 100)
```

fit	lwr	upr
61.04	57.4	64.61

El valor que predice el modelo es de alredors de un 61 %.

Ejercicio 2 (50 pt.)

```r
# cargamos los datos
penta <-
    read.table("penta.dat", header = T, na.strings = ".") %>%
    na.omit()

# funcion para caluclo de RMSE
rmse <- function(x,y) sqrt(mean((x-y)^2))
```

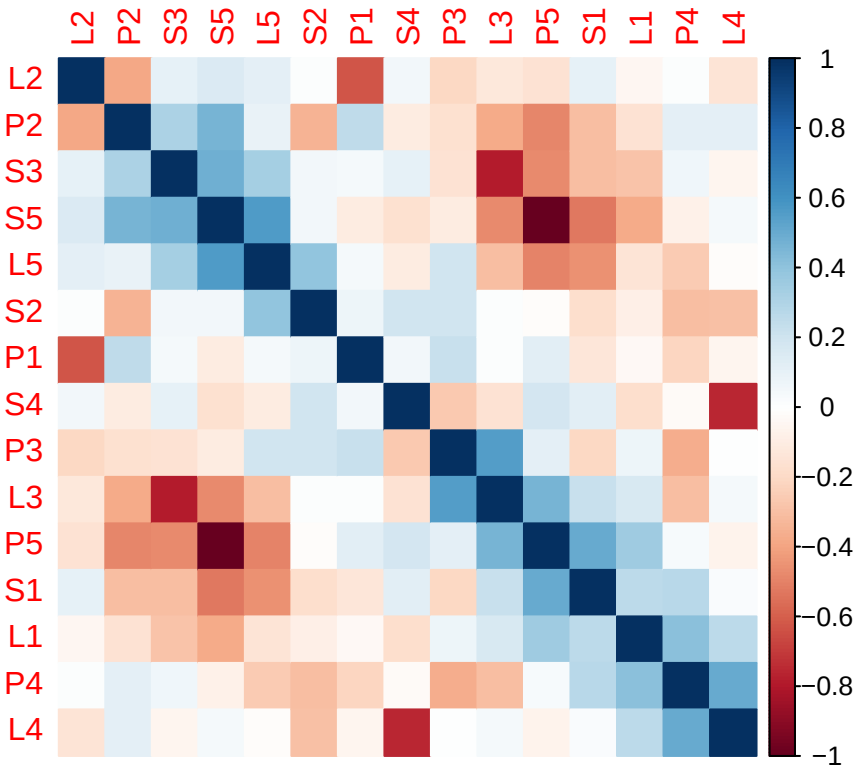
```r
set.seed(321)
idx <- sample(1:30, size = 20, replace = F)
ptrain <- penta[idx,]
ptest <- penta[-idx,]
```

(a)

Comprobar con los factores de inflación de la varianza que el modelo lineal completo y con todos los datos padece un problema grave de multi-colinealidad. Esto justifica que debamos reducir el número de variables predictoras o utilizar algún método alternativo.

```r
# creamos el modelo con todos los datos
mod_full <- lm(log.RAI ~ ., penta[-1])
mod_org <- lm(log.RAI ~ ., ptrain[-1])
```

```r
X <- model.matrix(mod_full)[,-1]

# hacemos un plot de correlación
corrplot::corrplot(cor(X), method = "color", order = 'AOE')
```




```
# Chequeamos las vifs
vifs_x <- vif(X)
pander(rbind(vifs_x, SE = sqrt(vifs_x)), digits = 3)
```

Table 12: Table continues below

	S1	L1	P1	S2	L2	P2	S3	L3	P3
vifs_x	1.87	2.06	2.26	1.83	2.84	2.97	5.94	11.6	5.35
SE	1.37	1.44	1.5	1.35	1.69	1.72	2.44	3.4	2.31

	S4	L4	P4	S5	L5	P5
vifs_x	11.9	12.1	10.6	3645	33.3	3317
SE	3.45	3.47	3.26	60.4	5.77	57.6

En primer lugar podemos evidenciar de manera visual, que las variables predictoras presentan correlación alta. Por otra parte al valorar los valore de VIF, se puede observar que hay variables con unos valores muy altos que confirman la existencia de co-linearidad que puede ser problemática al momento de validar el modelo completo.

(b)

Estudiar el modelo reducido que proporciona el método `stepwise`. Se trata de comparar los RMSE del modelo reducido y del modelo completo para el grupo de entrenamiento y para el grupo de prueba. También de ver si el modelo reducido salva el problema de multicolinealidad.

```r
mod_step <- step(lm(log.RAI ~ ., ptrain[-1]), trace = 0)
anova(mod_org, mod_step) %>%
    pander()
```

Table 14: Analysis of Variance Table

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
4	0.4863	NA	NA	NA	NA
11	0.598	-7	-0.1117	0.1312	0.9892

Una prueba de *anova* no permite establecer diferencias entre el modelo original y el modelo reducido por el método de `stepwise`, por lo que por parsimonia podemos elegir el modelo reducido.

```r
# Modelo Original
(rmse_org <- tibble(
     modelo = "original",
     entrenamiento = rmse(predict(mod_org), ptrain$log.RAI),
     prueba = rmse(predict(mod_org, newdata = ptest), ptest$log.RAI)
   ))
```

modelo	entrenamiento	prueba
original	0.1559287	0.582237

```r
# Modelo reducido `stepwise`
(rmse_step <- tibble(
    modelo = "stepwise",
    entrenamiento = rmse(predict(mod_step), ptrain$log.RAI),
    prueba = rmse(predict(mod_step, newdata = ptest), ptest$log.RAI)
))
```

modelo	entrenamiento	prueba
stepwise	0.172911	0.5183494

En cuanto al poder predictivo del modelo reducido, se puede observar que al utilizar la muestra de `test`, el valor del RMSE es menor que el del modelo original.

```r
# Chequeamos las vifs
X <- model.matrix(mod_step)[,-1]
vifs_x <- vif(X)
pander(rbind(vifs_x, SE = sqrt(vifs_x)), digits = 2)
```

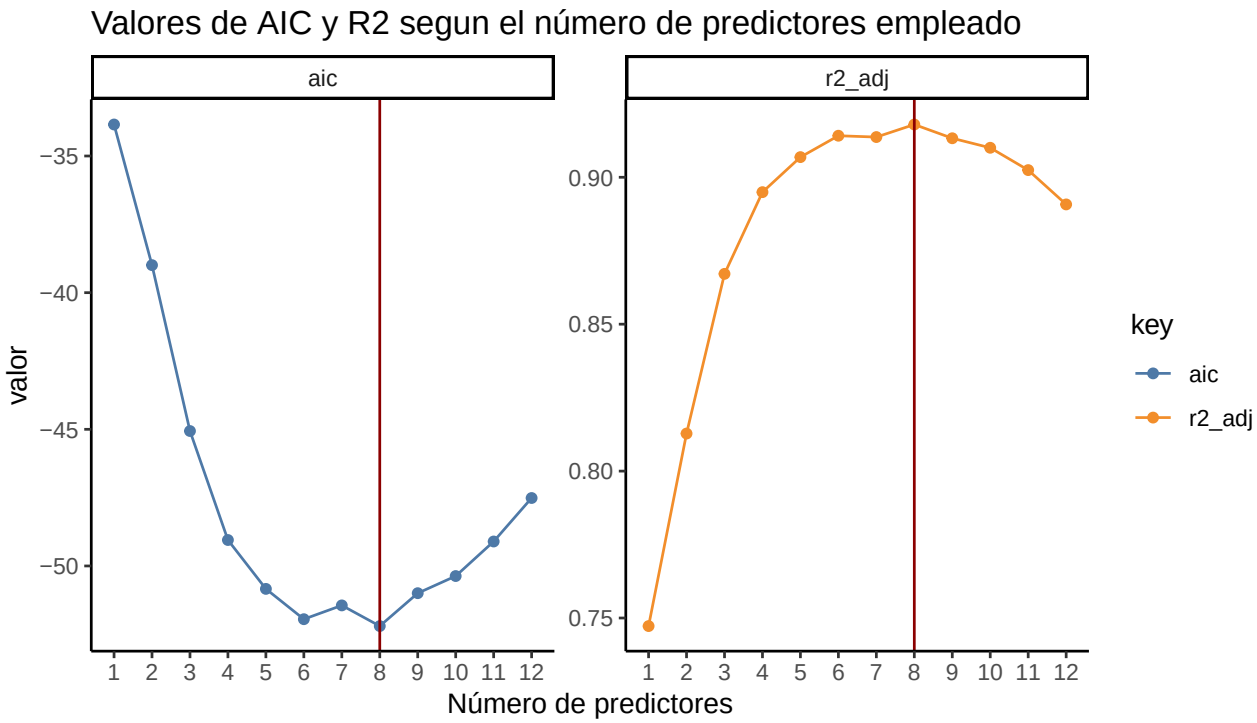
	S1	P1	S2	L2	S3	L3	L4	P4
vifs_x	1.8	2	1.2	1.6	5.1	6.1	2.2	3.4
SE	1.3	1.4	1.1	1.3	2.3	2.5	1.5	1.9

Si nos fiamos en los valores de las VIF, podemos observar que estos también son más bajos que los valores del modelo original.

(c)

Hallar el mejor modelo por el criterio del AIC y por el R^2 ajustado. ¿Coinciden? ¿Es el mismo modelo que selecciona el stepwise?

```
# Se crean las combinaciones de submodelos.
rs <-
    regsubsets(log.RAI ~ ., ptrain[-1], nvmax = 12) %>%
    summary()

n <- dim(ptrain)[1]
p <- length(rs$rss)


rs_crit <-
    rs$which %>%
    as_tibble() %>%
    rowid_to_column() %>%
    mutate(
        rowid = factor(rowid),
        rss = rs$rss,
        r2_adj = rs$adjr2,
        aic = n * log(rss/n) + (2:(p+1)) * 2)

# check minimo
mn <- ifelse(which.min(rs_crit$aic) == which.max(rs_crit$r2_adj),
             which.max(rs_crit$r2_adj), NA)

# Creamos el plot de AIC y R2
rs_crit %>%
    select(rowid, r2_adj, aic) %>%
    gather(key, value, -rowid) %>%
    ggplot(aes(rowid, value, color = key, group = key)) +
    geom_point() +
    geom_line() +
    geom_vline(xintercept = mn, color = "darkred") +
    facet_wrap(~key, scales = "free") +
    labs(title = "Valores de AIC y R2 segun el número de predictores empleado",
         x = "Número de predictores", y = "valor") +
    scale_color_tableau() +
    theme_classic()
```



En el caso de los criterios de AIC y de R^2, podemos observar que para las diferentes combinaciones, el modelo de 8 predictores es que *maximiza* el valor de R^2 y al mismo tiempo *minimiza* el AIC.

```
# modelo elegido
vars_aic_r2 <- rs$obj$xnames[rs$which[8,]]
vars_step <- colnames(model.matrix(mod_step))


pander(rbind(vars_aic_r2, vars_step))
```

vars_aic_r2	(Intercept)	S1	P1	S2	L2	S3	L3	L4	P4
vars_step	(Intercept)	S1	P1	S2	L2	S3	L3	L4	P4

El modelo elegido por le criterio $AIC/R2$, resulta igual al modelo elegido mediante el procedimiento *stepwise*

(d)

Estudiar una regresión por componentes principales con estos datos. ¿Qué tal funciona con 8 componentes para el grupo de entrenamiento? ¿Cual es el número de componentes que se recomienda si hacemos una validación cruzada? Con ese número mínimo de componentes, ¿cual es el RMSE para el conjunto de prueba?

```
pca_1 <-
    ptrain %>%
    select(S1:P5) %>%
    FactoMineR::PCA(graph = F)

pander(t(pca_1$eig)[,1:8], digits = 2)
```

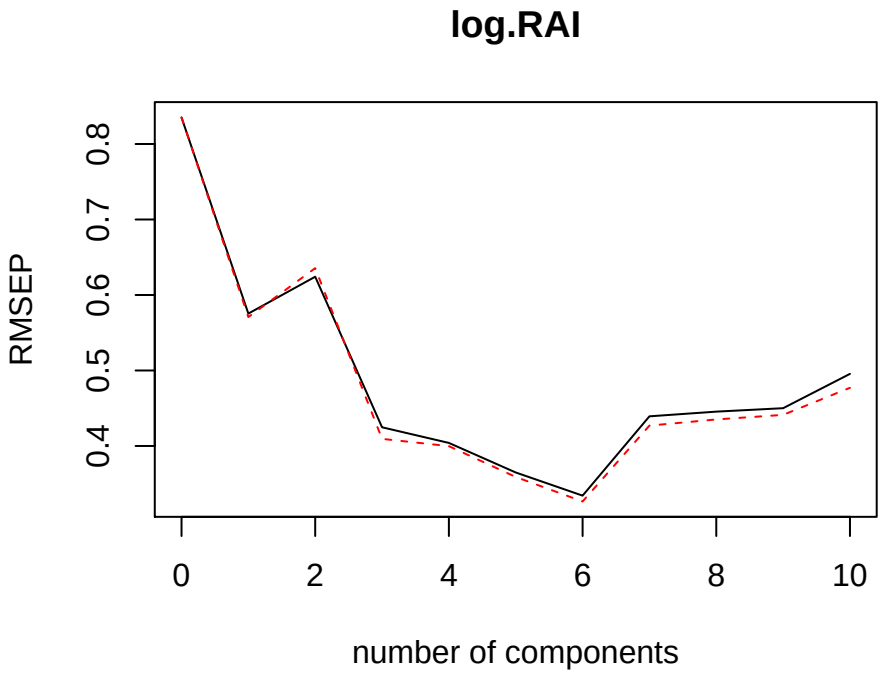
Table 19: Table continues below

	comp 1	comp 2	comp 3	comp 4	comp 5
eigenvalue	4.6	2.9	2.4	1.7	0.84
percentage of variance	31	19	16	11	5.6
cumulative percentage of variance	31	50	66	78	83

	comp 6	comp 7	comp 8
eigenvalue	0.74	0.6	0.39
percentage of variance	4.9	4	2.6
cumulative percentage of variance	88	92	95

Una primera aproximación permite observar que un modelo de 8 componentes explica hasta un 98% de la variabilidad de la muestra. Por otra parte, se puede observar que la reducción de la varianza explicada, cae bastante desde el componente 5 en adelante lo que también podría ayuda a determinar el numero ideal de componentes.

```
mod_pca <- pcr(log.RAI ~ ., data = ptrain[-1], validation = "CV", ncomp = 10)

plot(RMSEP(mod_pca, estimate = c("CV", "adjCV")))
```



Al utilizar una validación cruzada del modelo, podemos identificar que el $RMSE$ se minimiza en el modelo con 6 componentes.

```
# PCA con 8 componentes
(rmse_pca_8 <- tibble(
     modelo = "PCR 8",
     entrenamiento = rmse(predict(mod_pca, ncomp=8), ptrain$log.RAI),
```

```
    prueba = rmse(predict(mod_pca, ptest, ncomp=8), ptest$log.RAI)
  ))
```

modelo	entrenamiento	prueba
PCR 8	0.2208868	0.5548939

```
# PCA con 6 componentes
(rmse_pca_6 <- tibble(
    modelo = "PCR 6",
    entrenamiento = rmse(predict(mod_pca, ncomp=6), ptrain$log.RAI),
    prueba = rmse(predict(mod_pca, ptest, ncomp=6), ptest$log.RAI)
  ))
```

modelo	entrenamiento	prueba
PCR 6	0.2229476	0.5350163

Al utiliza los modelos con 8 y con 6 componentes para predecir la respuesta en la muestra de `test` podemos corroborar que el modelo con 6 componentes da una mejor solución.

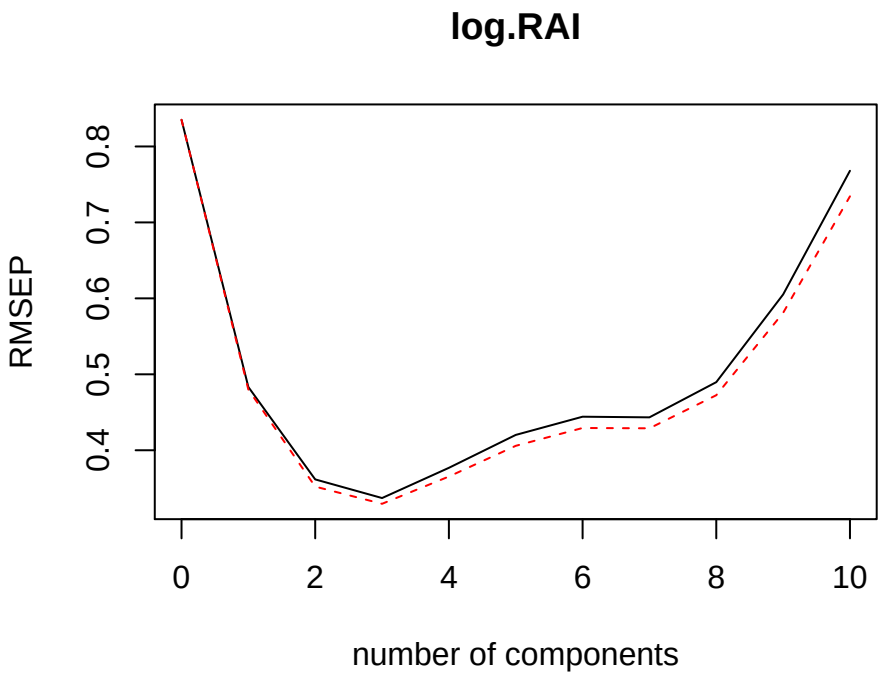
(e)

Estudiar una regresión PLS. ¿Cuantas componentes selecciona la validación cruzada? ¿Cual es la variabilidad de las variables predictoras explicada por 3 componentes? Con el paquete `plsdepot` se pueden estudiar las correlaciones entre las variables y las componentes y realizar el gráfico llamado *círculo de correlaciones*. ¿Cual es la variable predictora mejor correlacionada con la primera componente?

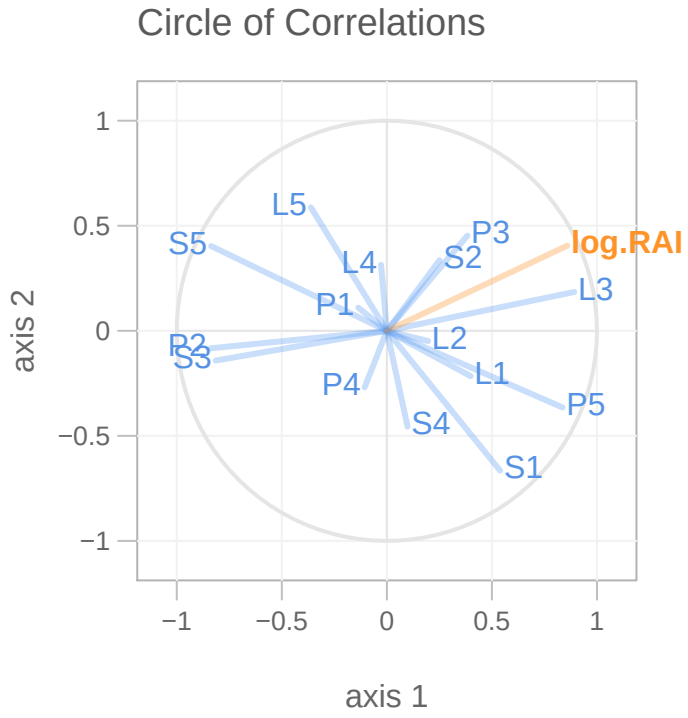
```
mod_plr <- plsr(log.RAI ~ ., data = ptrain[,-1], validation = "CV", ncomp = 10)
pls_reg <- plsdepot::plsreg1(ptrain[2:16], ptrain[17])

# VAriablidad explicada por 3 componentes
mod_plr$Xvar[3]
```

```
##   Comp 3
## 68.86143
```

```
plot(RMSEP(mod_plr))
```




```
plot(pls_reg)
```



El modelo de 3 componentes explica hasta un 68.9 de la varianza de los datos. En el gráfico se puede observar que la variable que mejor predice con el eje 1 es `P2`

```
# PLS
(rmse_pls <- tibble(
    modelo = "PLS",
    entrenamiento=rmse(predict(mod_plr, ncomp=3), ptrain$log.RAI),
    prueba=rmse(predict(mod_plr, ptest, ncomp=3), ptest$log.RAI)))
```

modelo	entrenamiento	prueba
PLS	0.2040088	0.512713

Finalmente, calculamos el valor de RMSE que resulta de la aplicación del modelo de PLS con la muestra de test.

(f)

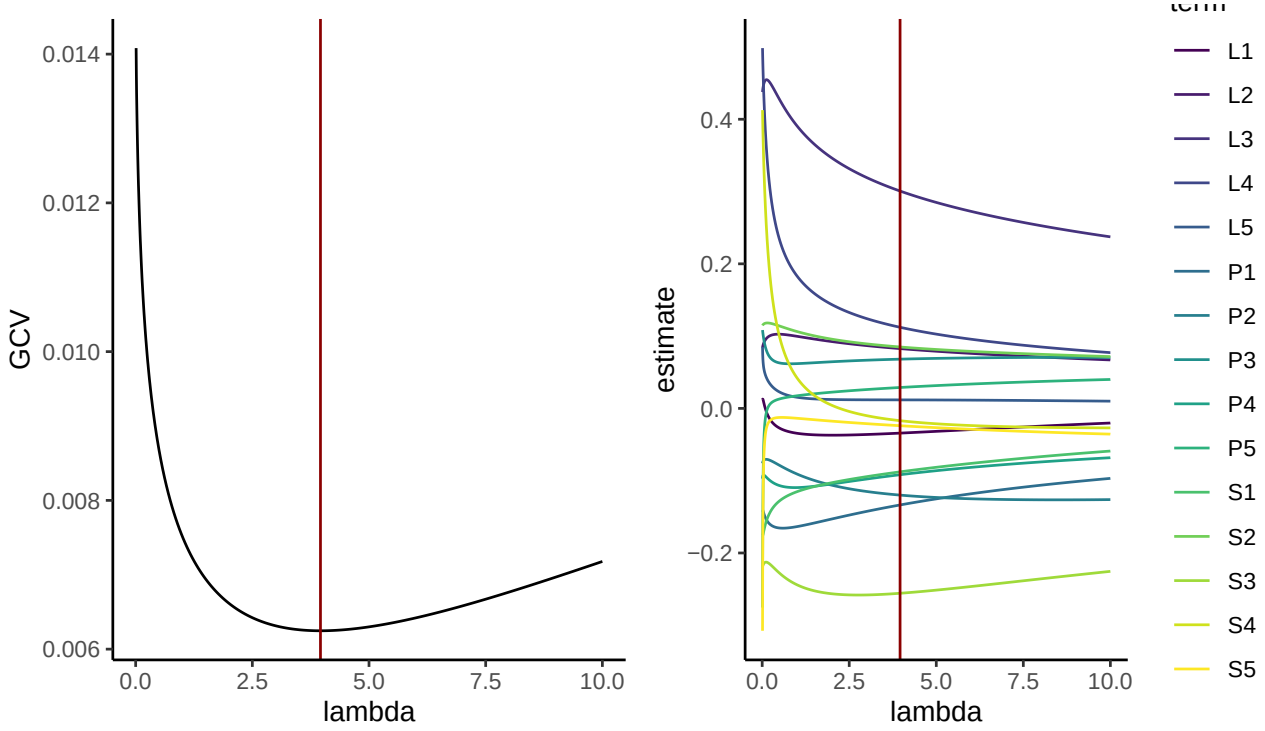
Estudiar también el método de Ridge Regression con la función `lm.ridge()`. Además de los valores RMSE para el grupo de entrenamiento y de prueba, debemos acompañar el análisis con un gráfico de los coeficientes. Para hallar el lambda óptimo podemos empezar por un límite superior bajo y aumentar sucesivamente ese límite hasta que el valor óptimo no sea el límite superior. Aunque siempre es mejor estandarizar los datos, en este ejercicio no lo haremos para no complicar más los cálculos del RMSE del conjunto de prueba. Internamente, lo hace la propia función `lm.ridge()`.

Nota: Si el resultado del ajuste de las predicciones a los datos es decepcionante podríamos optarpor utilizar la función `v.glmnet(...,alpha=0)` del paquete `glmnet`. Esta función no necesita que le indiquemos una secuencia de valores de lambda, aunque sí requiere que le pasemos los datos como objeto matrix. Además podemos utilizar la validación leave one out simplemente haciendo que el número de carpetas (folds) sea exactamente el número de observaciones.

Nota2: Aunque no hace falta, si se utilizan las funcines `cv.glmnet()` y `glmnet()` para el cálculo del modelo Ridge Regression, se puede comprobar que los resultados difieren notablemente de los que tenemos con la función `lm.ridge()`. Una sencilla explicación se puede leer en https://stats.stackexchange.com/questions/74206/ridge-regression-results-different-in-using-lm-ridge-and-glmnet

```
# Previa expermientacion definimos valores de lambda a graficar.
mod_rg <- lm.ridge(log.RAI ~ ., data = ptrain[,-1], lambda = seq(0.01, 10, .005))

lbd <-
    tidy(mod_rg) %>%
    distinct(lambda, GCV) %>%
    rowid_to_column() %>%
    top_n(1, -GCV)


# plot de los GCV

p1 <- tidy(mod_rg) %>%
    ggplot(aes(lambda, GCV)) +
    geom_line() +
    geom_vline(xintercept = lbd$lambda, color = "darkred") +
    theme_classic()


p2<- tidy(mod_rg) %>%
    ggplot(aes(lambda, estimate, color = term)) +
    geom_line() +
    geom_vline(xintercept = lbd$lambda, color = "darkred") +
    scale_colour_viridis_d() +
    theme_classic()
```

```
grid.arrange(p1,p2,nrow = 1)
```



Se utiliza el valor de lambda que minimiza el GCV. En este caso corresponde a un valor de 3.96.

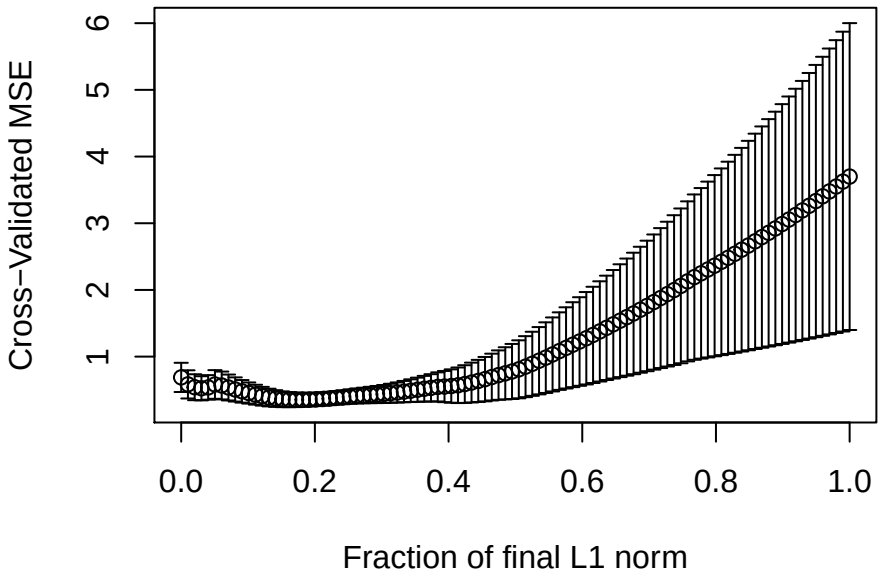
```
# hacemos la prediccion con el valor de lambda que minimiza GCV

ypred_rg_t <- cbind(1, as.matrix(ptest[2:16])) %*% coef(mod_rg)[lbd$rowid,]

ypred_rg_e <- cbind(1, as.matrix(ptrain[2:16])) %*% coef(mod_rg)[lbd$rowid,]

# rendimiento del modelo
(rmse_rg <- tibble(
    modelo = "Ridge R",
    entrenamiento = rmse(ypred_rg_e, ptrain$log.RAI),
    prueba = rmse(ypred_rg_t, ptest$log.RAI)))
```

modelo	entrenamiento	prueba
Ridge R	0.1998409	0.5967337

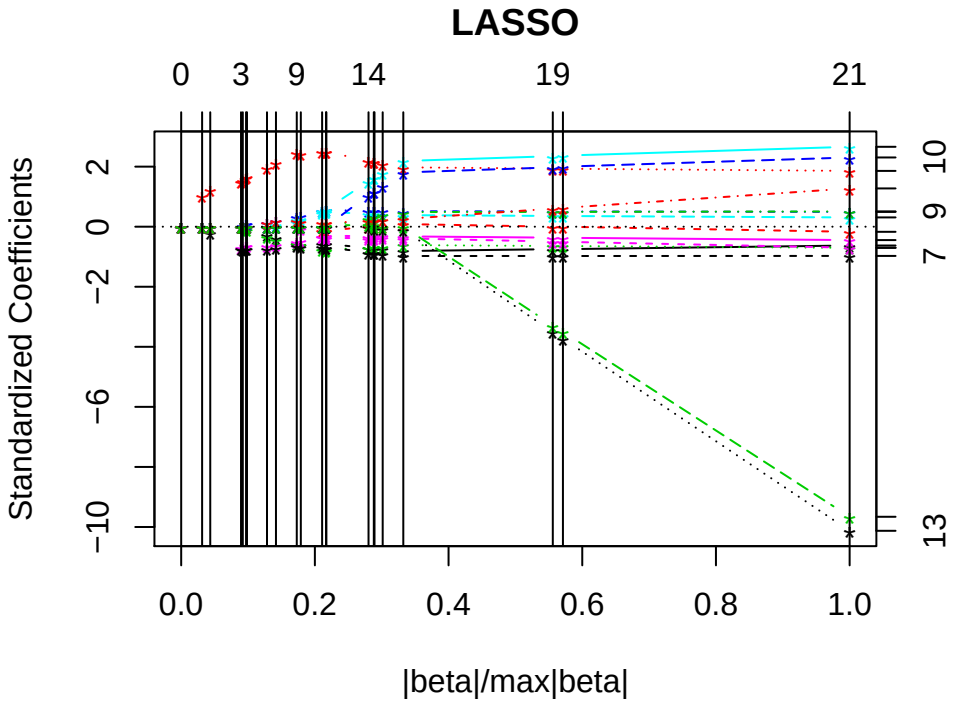
(g)

Finalmente estudiaremos el método LASSO con la función `lars()` del paquete del mismo nombre. Nota: Tal vez el resultado es demasiado restrictivo y hay que aumentar el número de variables seleccionadas, lo que equivale a aumentar ligeramente el valor de lambda.

```
mod_lar <- lars(as.matrix(ptrain[2:16]), ptrain$log.RAI)
mod_lar_cv <- cv.lars(as.matrix(ptrain[2:16]), ptrain$log.RAI)
```




```
plot(mod_lar)
```




```
# minimzamos x cv
ind <-
    as_tibble(mod_lar_cv) %>%
    top_n(1, -cv) %>%
    pull(index)
```

Creamos el modelo de LASSO y creamos el modelo para realizar una validación cruzada y optimizar el parámetro lambda.

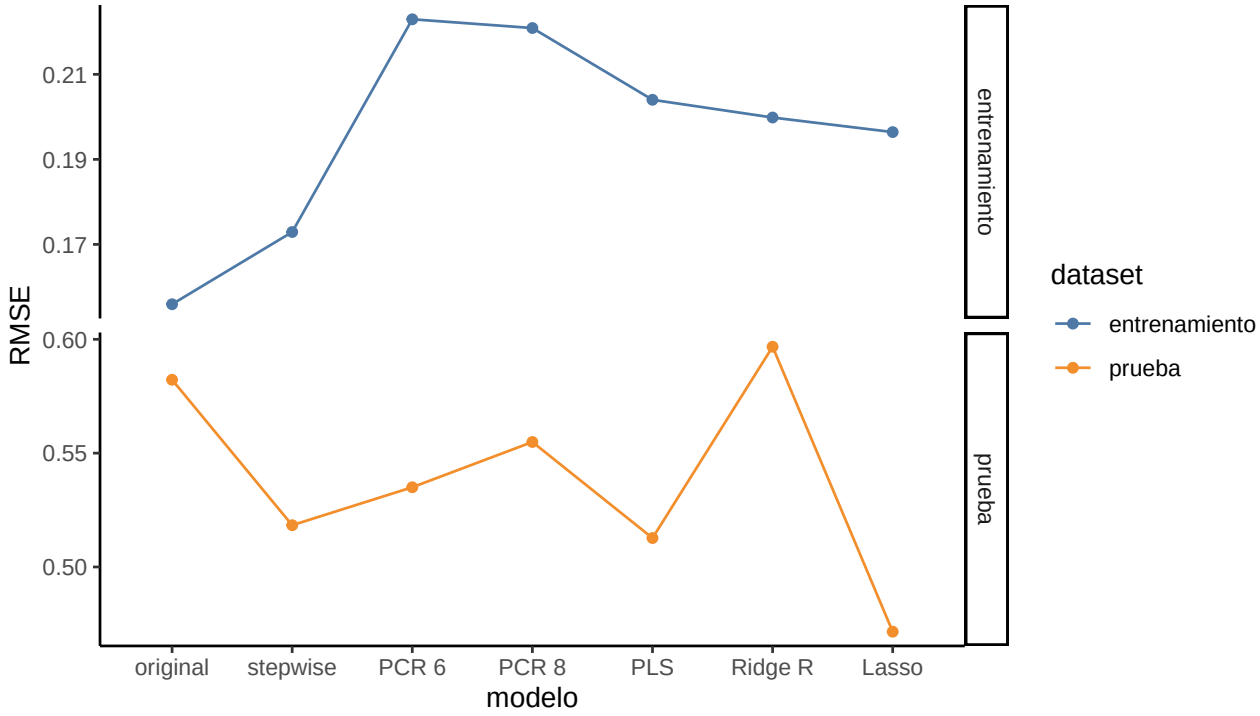
```
ypred_lasso <- predict.lars(mod_lar, ptest[2:16], s = ind, mode="fraction")$fit
ypred_lasso_e <- predict(mod_lar, ptrain[2:16], s = ind, mode="fraction")$fit
```

```r
(rmse_lasso <- tibble(
    modelo = "Lasso",
    entrenamiento = rmse(ypred_lasso_e, ptrain$log.RAI),
    prueba = rmse(ypred_lasso, ptest$log.RAI)))
```

modelo	entrenamiento	prueba
Lasso	0.1964145	0.4715754

Podemos ver que el modelo de LASSO permite obtener una respuesta optima con el dataset de prueba.

```r
bind_rows(
    rmse_org,
    rmse_step,
    rmse_pca_6,
    rmse_pca_8,
    rmse_pls,
    rmse_rg,
    rmse_lasso
) %>%
    mutate(modelo = factor(modelo, levels = c('original', 'stepwise', 'PCR 6',
                                              'PCR 8', 'PLS', 'Ridge R', 'Lasso'))) %>%
    gather(dataset, RMSE, -modelo) %>%
    ggplot(aes(modelo, RMSE, color = dataset, group = dataset)) +
    geom_point() +
    geom_line() +
    facet_grid(rows = vars(dataset), scales = "free") +
    scale_color_tableau() +
    theme_classic()
```



Podemos graficar el resultado del RMSE de todos los modelos, con los datos de *entreneamiento* y los de

prueba. De esta manera podemos ver que el modelo que aporta la mejor solución para el set de prueba, es el de `Lasso`.

Ejercicio 3 (25 pt.)

Es posible utilizar la regresión logística para estudiar la discriminación entre dos grupos dado un conjunto de variables explicativas. Para más detalles podéis consultar el apartado 8.10 del libro de Manly.

(a)

Realizar un análisis discriminante con ayuda de la regresión logística con los datos de los 49 gorriones hembra después de una tormenta. Reproducir con todo detalle la tabla 8.6 de la página 153 del libro de Manly.

```
load("gorriones.RData")
mod_lg <- glm(superviv ~ x1 + x2 + x3 + x4 + x5, gorriones,
              family = binomial(link = "logit"))
mod_NULL <- glm(superviv ~ 1, gorriones, family=binomial(link="logit"))
```

```
# Para reporducir *con todo detalle* se puede utilizar la sintaxis de dplyr
# y crear la tabla
tbl <-
    tidy(mod_lg) %>%
    mutate(
        chi_2 = (estimate / std.error) ^ 2,
        Variable = c("Constant", "Total length","Alar length", "Length beak and head",
                     "Length humerus", "Length keel of sternum"),
        chi_2 = round(chi_2, 2)) %>%
    mutate_at(vars(estimate:p.value), list( ~ round(., 3))) %>%
    select(
        Variable, '  estimate' = estimate, 'Standard error' = std.error,
        'Chi-squared' = chi_2, 'p-Value' = p.value)

tbl[1,4:5] <- "--"

pander(tbl, justify = "lcccc")
```

Variable	estimate	Standard error	Chi-squared	p-Value
Constant	13.58	15.87	–	–
Total length	-0.163	0.14	1.36	0.244
Alar length	-0.028	0.106	0.07	0.794
Length beak and head	-0.084	0.629	0.02	0.894
Length humerus	1.062	1.023	1.08	0.299
Length keel of sternum	0.072	0.417	0.03	0.864

Es posible crear una tabla muy similar, con la sintaxis de la librería `dplyr`.

(b)

Contrastar con un test χ^2 la significación de la regresión y explicar su resultado.

```
anova(mod_NULL, mod_lg, test = "Chisq") %>%
    pander()
```

Table 27: Analysis of Deviance Table

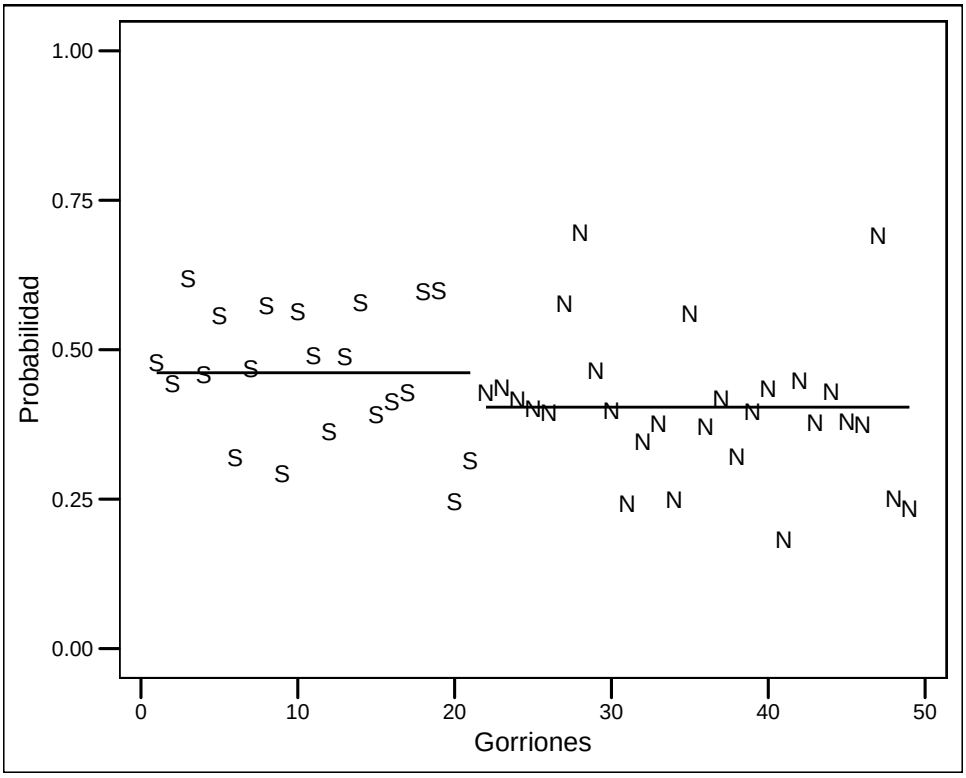
Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
48	66.92	NA	NA	NA
43	64.07	5	2.854	0.7225

En este caso hacemos un contraste del modelo creado con el modelo nulo. De esta manera lo que se evalúa como $H0$ es que el modelo creado no difiere del modelo nulo, es decir no hay diferencias entre los grupos. El resultado del estadístico es mas bien bajo, y con un valor no significativo, por lo que no se puede rechazar la $H0$ y por ende, se establece que las diferencias entre los grupos no son significativas.

(c)

Realizar un gráfico como el de la figura 8.2 de la página 156 del libro de Manly pero con los datos de este ejercicio y valorar el resultado.

```
tibble(
    fitted = mod_lg$fitted.values,
    class = augment(mod_lg)$superviv
) %>%
    rowid_to_column() %>%
    group_by(class) %>%
    mutate(mean = mean(fitted)) %>%
    ggplot(aes(rowid, fitted, shape = class)) +
    geom_point(size = 3) +
    geom_line(aes(y=mean)) +
    scale_shape_manual(values = c("N","S")) +
    theme_base(base_size = 10) +
    labs(x="Gorriones", y="Probabilidad") +
    scale_y_continuous(limits = c(0,1)) +
    guides(shape = F)
```



En el gráfico podemos observar que efectivamente, los dos grupos no presentan mayores diferencias en cuanto a lo predicho por el modelo según las variables cuantitativas que utiliza.

---

d)

Calcular un intervalo de confianza para el parámetro de la variable x4=length humerus y para su odds ratio.

```
as_tibble(c(coef(mod_lg)[5], confint(mod_lg)[5,]), rownames = "key") %>%
    mutate(odds_x4 = exp(value)) %>% rename(x4 = value) %>%
    gather(coefficients, b, -key) %>%
    spread(key, b) %>%
    select(coefficients, `2.5 %`, '-' = x4, `97.5 %`) %>%
    arrange(rev(coefficients))
```

coefficients	2.5 %	-	97.5 %
x4	-0.9118587	1.061743	3.166486
odds_x4	0.4017768	2.891405	23.723984

Podemos observar en la tabla el valor de $\beta_{lengthhumerus}$, y su intervalo de confianza para el 95%, así como el valor de la *odds ratio* y su intervalo de confianza.

(e) Calcular la tabla de confusión de la clasificación obtenida con la regresión logística.

```
ypred <-
    ifelse(
```

```
        predict.glm(mod_lg, type = "response") > 0.5,
        'S', 'N') %>%
    factor()


# para la matriz de confusión se puede utilizar la función de
# la librería `caret`
cm <- confusionMatrix(gorriones$superviv, ypred)


pander(cm$table)
```

	N	S
N	24	4
S	14	7

```
pander(cm$overall)
```

Table 30: Table continues below

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull
0.6327	0.2025	0.4829	0.7658	0.7755

AccuracyPValue	McnemarPValue
0.9926	0.03389

Para el cálculo de la matriz de confusión, en primer lugar clasificamos la predicción del modelo; el modelo entrega una predicción probabilística acerca de la categoría, pero necesitamos utilizar una categoría determinada. Una vez que tenemos la predicción categórica, podemos utilizar la función `confusionMatrix` del paquete `caret`, que permite hacer la tabla de confusión y además calcular los parámetros de exactitud de la predicción.