UNIVERSIDAD NACIONAL DE COLOMBIA

Trabajo 1

Luis David Hernández Pérez

Daniel Felipe Villa Rengifo

Juan Gabriel Carvajal Negrete

Said Alejandro Duran

Introducción a análitica

Cesar Augusto Gomez Velez

Universidad Nacional de Colombia

Sede Medellín

Facultad de Ciencias

Medellín, Noviembre del 2023

Tabajo 1: Modulo 2

1. (50%) En este ejercicio, se quiere predecir el numero de aplicaciones recibidas en distintos colegios universitarios americanos. utilizando las demás variables en el conjunto de datos College.

   a) Divida el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba.

   b) Ajustar un modelo lineal usando mínimos cuadrados en el conjunto de entrenamiento, y informar el error de prueba obtenido.

   c) Ajustar un modelo de regresión **ridge** en el conjunto de entrenamiento,escogiendo un valor de λ mediante validación cruzada. Informe el error de prueba obtenido.

   d) Ajuste un modelo **lasso** en el conjunto de entrenamiento, escogiendo el valor de λ mediante validación cruzada. Informe el error de prueba obtenido, junto con el número de estimaciones de coeficientes distintos de cero.

   e) Ajuste un modelo de PCR en el conjunto de entrenamiento, escogiendo M mediante validación cruzada. Informe el error de prueba obtenido, junto con el valor. de M que fue seleccionado.

   f) Ajuste un modelo PLS en el conjunto de entrenamiento, con M escogido mediante validación cruzada. Informe el error de prueba obtenido, junto con el valor de M que se selecciono.

   g) Comente los resultados obtenidos.

      - ¿Con que precisión se puede predecir la cantidad de solicitudes universitarias recibidas?.
      - ¿Cuanta diferencia hay entre los errores de prueba resultantes de estos cinco enfoques?

Solución

Punto 1:

a) División de la base de datos

Para dividir la base de datos tomaremos el 70% de las observaciones para entrenamiento y el otro 30% para prueba.

```
set.seed(1003)
datos <- ISLR2::College
lenp <- floor(0.7*nrow(datos))
filas_train <- sample(1:nrow(datos),lenp)
datos_train <- datos[filas_train,]
datos_test <- datos[-filas_train,]
```

Se presenta los primeros y últimos registros con algunas de las variables de la base de datos de entrenamiento y de prueba, esto para intentar minimizar espacio en el documento.

Datos de entrenamiento

Universidad	Private	Apps	Accept	perc.alumni	Expend	Grad.Rate
Geneva College	Yes	668	534	26	6786	74
Central College	Yes	1283	1113	29	8444	67
University of Louisville	No	4777	3057	24	10207	31
Alaska Pacific University	Yes	193	146	2	10922	15
...		...	...	...	...	...
Cedarville College	Yes	1307	1090	34	6897	64
East Carolina University	No	9274	6362	18	9002	58
Rowan College of New Jersey	No	3820	1431	6	7252	51
Warren Wilson College	Yes	440	311	20	9430	63

Datos de prueba

Universidad	Private	Apps	Accept	perc.alumni	Expend	Grad.Rate
Abilene Christian University	Yes	1660	1232	12	7041	60
Albion College	Yes	1899	1720	37	11487	73
Allegheny College	Yes	2652	1900	41	11711	76
Alma College	Yes	1267	1080	32	9305	68
...		...	...	...	...	...
Willamette University	Yes	1658	1327	37	10779	68
Winthrop University	No	2320	1805	26	6729	59
Worcester State College	No	2197	1515	14	4469	40
Yale University	Yes	10705	2453	49	40386	99

b) Modelo con mínimos cuadrados.

```
modelo1 <- lm(Apps~.,data = datos_train)
# Coeficientes del modelo
modelo1$coefficients
```

```
  (Intercept)    PrivateYes        Accept        Enroll     Top10perc
-418.76046487 -446.10363413    1.63490765   -0.96090340   55.57436719
    Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
 -16.68524902    0.04172949    0.05572888   -0.09370301    0.16640491
       Books      Personal           PhD     Terminal     S.F.Ratio
   0.06537217    0.02817159   -6.85524641   -2.52351241   19.40756786
  perc.alumni        Expend     Grad.Rate
   0.18773458    0.05882818    6.12117852
```

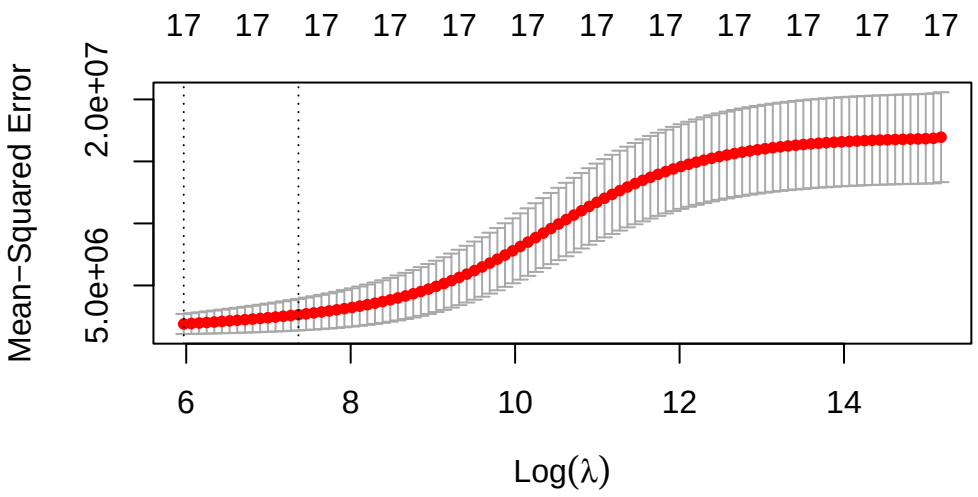
```
# Error de prueba
predicciones <- predict(modelo1,datos_test)
MSE1 <- mean((datos_test$Apps-predicciones)^2)
```

El error de prediccion es de 1.1458612×10^6

c) Modelo de regresión ridge.

Para ajustar un modelo de regresión ridge primero encontremos el λ que minimice el error cuadrático medio de la regresión, para esto vamos a usar la función cv.glmnet, esta función realiza una validación cruzada de 10-folds internamente.A continuación se muestra la gráfica para diferentes valores del λ y su efecto en el error del modelo.

```
x <- model.matrix(Apps~.,datos_train)[,-1]
y <- datos_train$Apps
xtest <- model.matrix(Apps~.,datos_test)[,-1]
# Grafica con diferentes valores de lambda
set.seed(734)
cv.out <- cv.glmnet(x,y,alpha=0)
# lambda optimo
optilamb <- cv.out$lambda.min
plot(cv.out)
```



El valor de λ que hace que el modelo resulte con el menor error es 391.4477254. Ahora ajustemos la regresión ridge con este λ.

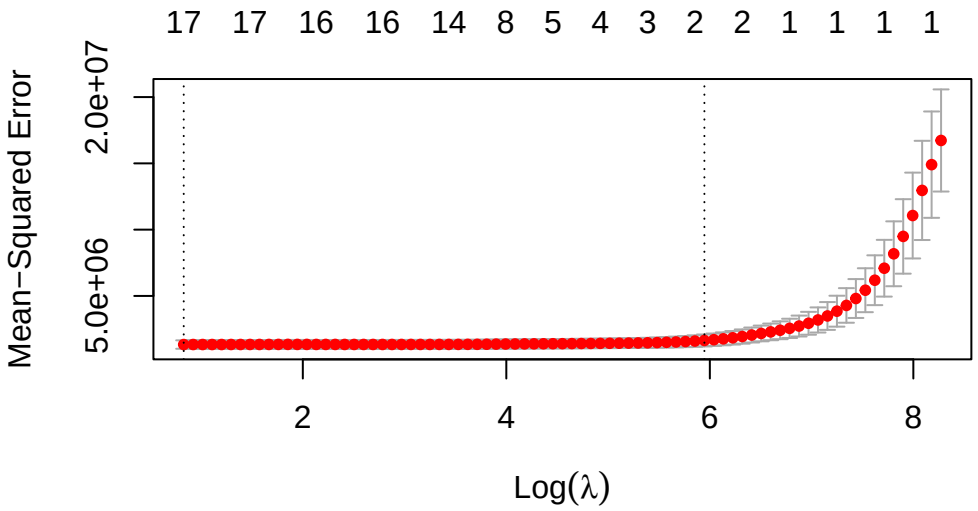
```
# Modelo ridge
mod.ridge<- glmnet(x, y, alpha = 0, lambda = optilamb)
redge.predi <- predict(mod.ridge,s=optilamb,newx = xtest)
# MSE de ridge
MSEr <- mean((redge.predi-datos_test$Apps)^2)
```

Con un λ encontrado por medio de validación cruzada, ajustando un modelo con los datos de entrenamiento y finalmente evaluando dicho modelo con los datos de prueba se obtuvo un error de test de 1.109606×10^6

d) Modelo de regresión lasso.

Para ajustar el modelo de regresión lasso se procede igual que en la regresión ridge pero, teniendo en cuenta el parámetro alpha. Realicemos validación cruzada para determinar el λ que minimice el error.

```
set.seed(189)
cv.out1 <- cv.glmnet(x,y,alpha=1)
optilamb.l <- cv.out1$lambda.min
plot(cv.out1)
```



Por medio de la validación cruzada se determino que el que minimiza el error del modelo es 2.2927175. Ahora ajustemos el modelo lasso y determinemos el error de prueba.

```
# Modelo lasso
mod.lasso<- glmnet(x, y, alpha = 1, lambda = optilamb.l)
lasso.predi <- predict(mod.lasso,s=optilamb.l,newx = xtest)
# error de prueba
MSEl <- mean((lasso.predi-datos_test$Apps)^2)
```

Con un λ encontrado por medio de validación cruzada, ajustando un modelo con los datos de entrenamiento y finalmente evaluando dicho modelo con los datos de prueba se obtuvo un error de test de 1.144627×10^6

Coeficientes del modelo

```
lasso.coeff <- predict(mod.lasso , type = "coefficients",
s = optilamb.l)[1:18,]
lasso.coeff
```

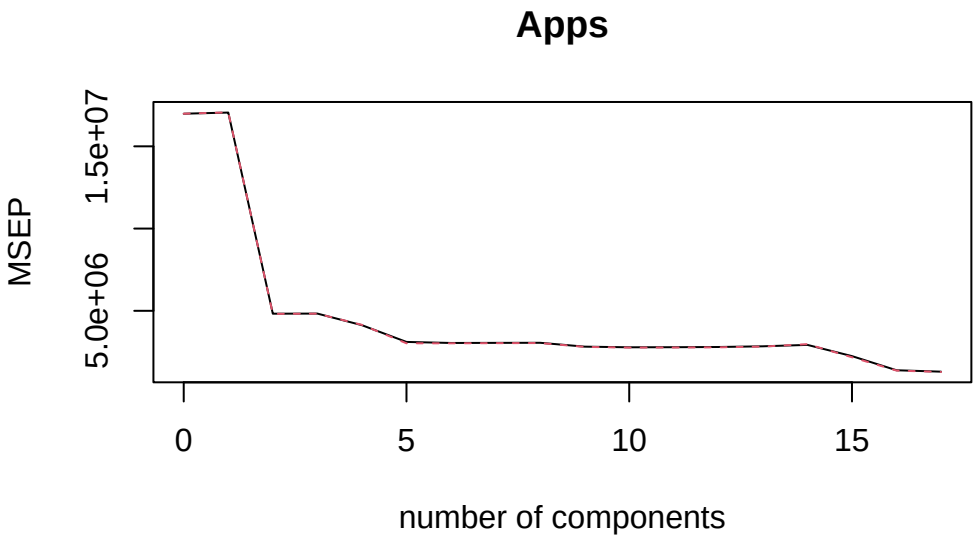
```
  (Intercept)    PrivateYes        Accept        Enroll     Top10perc
-441.97319169 -446.01011173    1.62056476   -0.83420214   53.79320846
    Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
 -15.28705010    0.02351327    0.05683887   -0.09112386    0.16502255
       Books      Personal           PhD     Terminal     S.F.Ratio
   0.06395986    0.02584610   -6.57679317   -2.38844052   18.24504172
  perc.alumni        Expend     Grad.Rate
  -0.08538603    0.05780048    5.75070467
```

Para este caso no se presentan coeficientes estimaos iguales a cero.

e) Modelo de regresión PCR.

Para ajustar la regresión PCR (Principal Components Analysis) vamos a utilizar la función pcr() que trae incorporada el parámetro validation el cual realiza validación cruzada 10-fold para determinar M (el numero de componentes) que minimizan el error del ajuste del modelo.

```
# error por cada numero de componentes posibles
pcr.model <- pcr(Apps~.,data=datos_train,scale=TRUE,validation="CV")
validationplot(pcr.model,val.type = "MSEP")
```



De la gráfica tenemos que con M = p se obtiene el menor MSE pero seria igual a ajustar un modelo con la función lm(), por esto un M adecuado es M=5 ya que a partir de ahí el MSE del modelo no varia en gran cantidad y también estaríamos evitando el sobre ajuste. Con este numero de componentes encontremos el error de prueba.

```
# Error de prueba
pcr.pred <- predict(pcr.model , datos_test, ncomp = 5)
MSEpcr <- mean((pcr.pred - datos_test$Apps)^2)
```

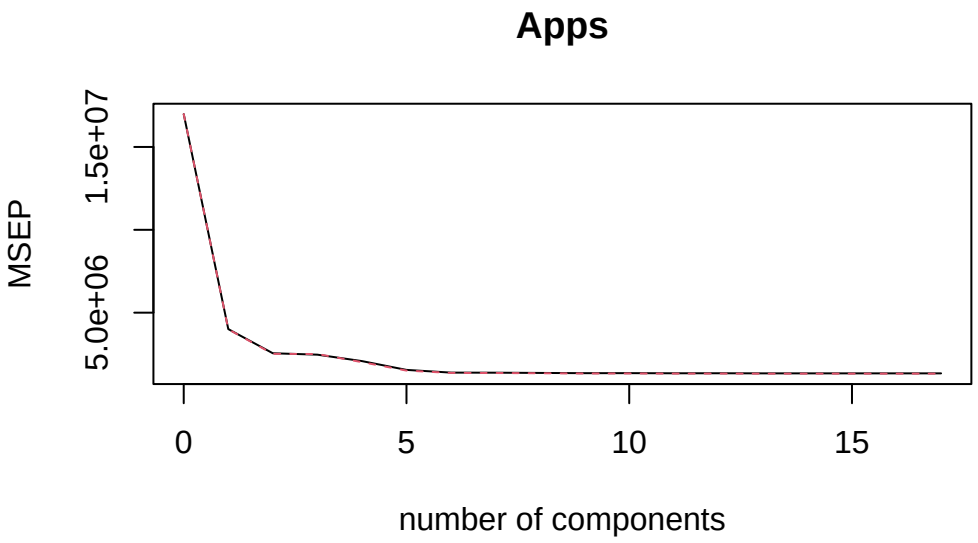
El modelo PCR presenta un error de prueba de 1.7650222×10^6

f) Modelo de regresión PLS.

Para ajustar el modelo PLS (mínimos cuadrados parciales) se utiliza la función plsr() que también hace internamente una validación de 10-fold para determinar el numero M (de direcciones) que obtenga el error mas bajo del modelo.

```
set.seed (34)
pls.model <- plsr(Apps ~ ., data = datos_train,
```

```
scale = TRUE , validation = "CV")
validationplot(pls.model , val.type = "MSEP")
```



De igual forma como lo observamos en la regresión PCR el numero de M optimo es 5, ya que de ahí en adelante no cambia significativamente el error con mas componentes. Ajustemos el modelo.

```
pls.pred <- predict(pls.model,datos_test,ncomp = 5)
MSEpls <- mean((pls.pred - datos_test$Apps)^2)
```

El modelo de PLS presenta un error de 1.2099127×10^6

g) Resultados

Modelo	Error de prueba
Modelo con mínimos cuadrados	1,145,861.24
Modelo de regresión ridge	1,109,606.14
Modelo de regresión lasso	1,144,628.00
Modelo de regresión PCR	1,765,022.18
Modelo de regresión PLS	1,209,912.69

Comparando los errores de predicción de todos los modelos, notamos que los dos modelos que presentan el mayor error de prueba son el modelo de regresión PCR y el modelo de regresión PLS. Por otro lado, el error de prueba de los modelos de regresión por mínimos cuadrados y lasso es muy similar. Si tuviéramos que elegir un modelo basándonos en el error de prueba, optaríamos por el modelo de regresión ridge, ya que con un error de prueba de 1,109,606 es el más pequeño entre todos los modelos.

Aunque el modelo de regresión ridge fue el que presento un mejor error de prueba dado que este es relativamente grande en comparación con la amplitud de los valores de la variable respuesta.En este caso la precisión del modelo es limitada y el MSE sugiere que las predicciones del modelo tienen un error promedio significativo en comparación con los valores reales.

2. (50%) Esta pregunta utiliza las variables dis (la media ponderada de distancias a cinco centros de empleo de Boston) y nox (concentracion de óxidos de nitrogeno en partes por 10 millones) del conjunto de datos Boston. Vamos atratar dis como predictor y nox como respuesta.

   a) Use la función poly() para ajustar una regresión polinomial cúbica y con esta predecir la variable nox usando dis. Reporte el resultado de la regresión, luego grafique los datos resultantes y los ajustes polinómicos.

   b) Grafique los ajustes polinómicos para un rango de polinomios de diferentes grados (digamos, de 1 a 10), y reporte la suma de cuadrados de los residuales asociada.

   c) Realice una validacion cruzada o algun otro enfoque para seleccionar el optimo grado para el polinomio y explique sus resultados.

   d) Use la función bs() para ajustar una spline de regresion para predecir nox usando dis. Reporte la salida para el ajuste usando cuatro grados de libertad. ¿Cómo ubicó los nodos?. Graique el ajuste resultante.

   e) Ahora ajuste una spline de regresión para un rango de grados de libertad,y grafique los ajustes resultantes e informe el RSS resultante. Describa los resultados obtenidos.

   f) Realice una validación cruzada o algún otro enfoque para seleccionar los mejores grados de libertad para una spline de regresión sobre estos datos. Describa sus resultados.

Solución

Punto 2:

a)

```r
datos2 <- Boston
```

```r
# Ajuste del modelo de regresión polinomial cubico
#=================================================
fit <- lm(nox ~ poly(dis, 3), data = datos2)
summary(fit)
```

```
Call:
lm(formula = nox ~ poly(dis, 3), data = datos2)

Residuals:
      Min        1Q    Median        3Q       Max
-0.121130 -0.040619 -0.009738  0.023385  0.194904

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06207 on 502 degrees of freedom
Multiple R-squared:  0.7148,    Adjusted R-squared:  0.7131
F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

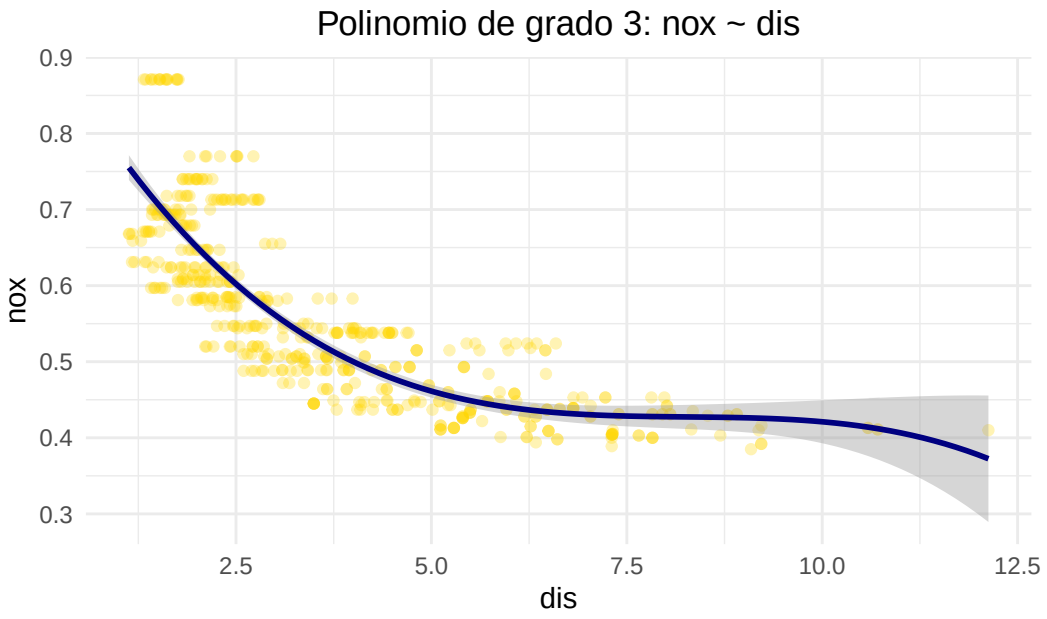


Figure 1: Gráfico del ajuste del polinomio cubico

A partir del gráfico anterior, se evidencia que el ajuste de regresión polinómica de grado 3 logra representar de manera efectiva la tendencia de los datos.

b)

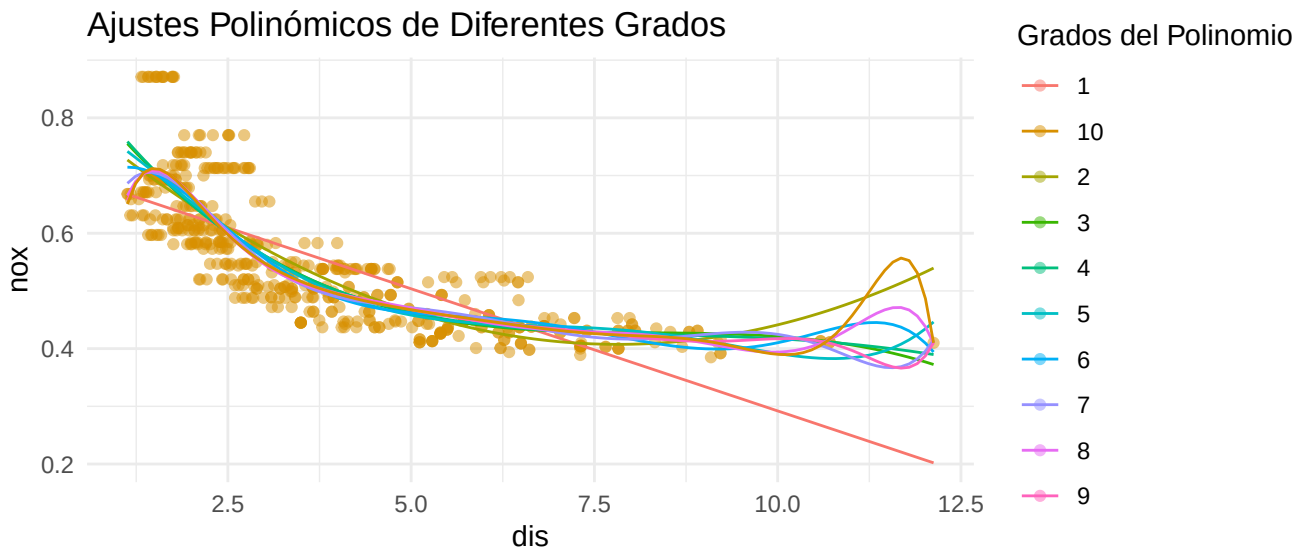


Figure 2: Gráfico de ajustes polinomicos de diferentes grados

A partir del gráfico anterior, se puede apreciar que, con la excepción del polinomio de grado 1, los polinomios de grados superiores muestran una notable capacidad de ajuste a la tendencia de los datos.

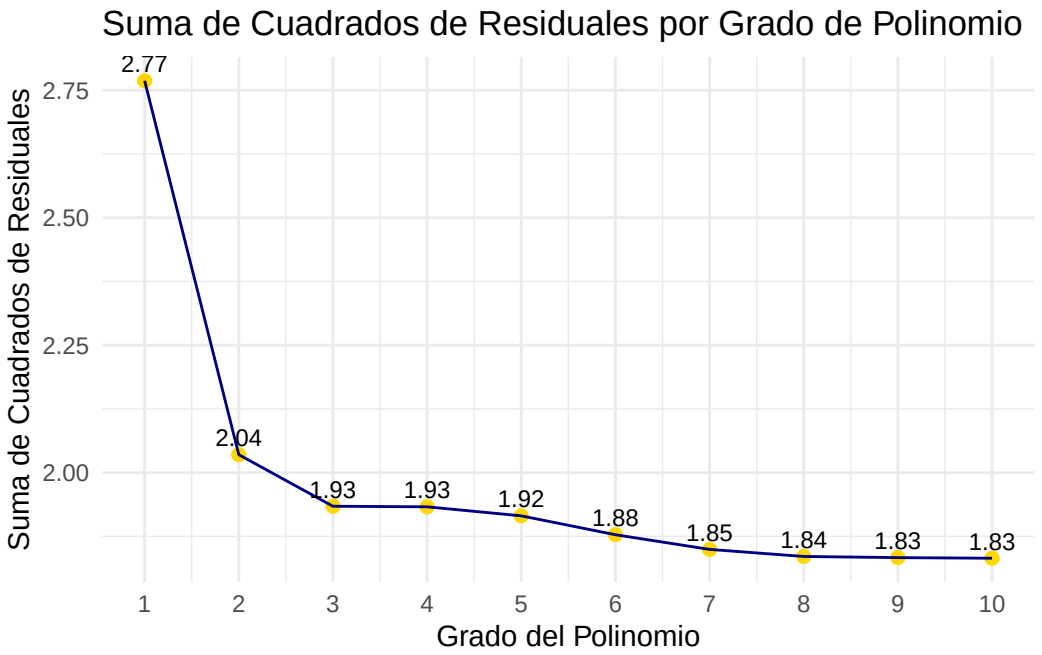
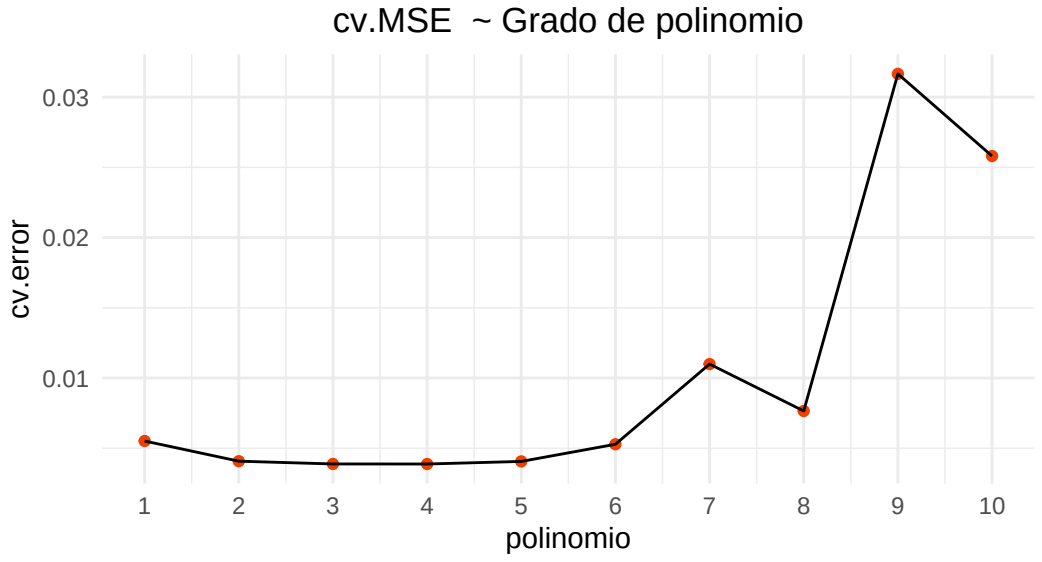


Figure 3: Gráfico de la suma de cuadrados residuales

A partir del gráfico anterior apreciamos que a medida que aumenta el grado del polinomio, la suma de cuadrados residuales tiende a disminuir. Esto significa que a medida que se consideran polinomios de mayor grado, el modelo se ajusta mejor a los datos de entrenamiento, reduciendo la cantidad de error no explicado, lo que resulta en una mejor capacidad del modelo para explicar la variabilidad en los datos. Sin embargo, es importante tener cuidado al seleccionar un grado de polinomio alto, ya que un exceso de complejidad puede llevar a un sobreajuste del modelo, lo que significa que se adaptaría demasiado a los datos de entrenamiento y no generalizaría bien a nuevos datos. Por lo tanto, se necesita un equilibrio al elegir el grado óptimo del polinomio.

c)



```r
# Polinomio con menor error de validación
which.min(cv.error)
```

```
[1] 4
```

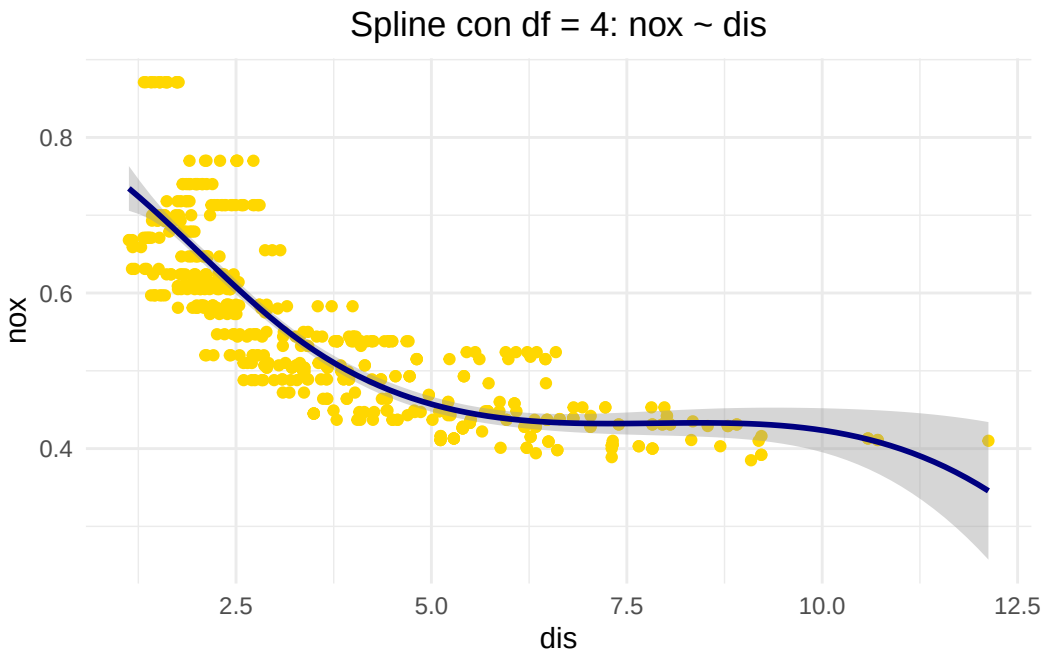
El análisis de la minimización del error de validación (MSE) indica que el polinomio de grado 4 es el que mejor se ajusta en este caso. No obstante, al considerar la tendencia en la evolución del error de validación en el gráfico, surge la preferencia por el ajuste con un polinomio de grado 3, ya que la diferencia entre este y el polinomio de grado 4 no se muestra significativa.

d)

Del literal anterior, consideramos ajustar un spline cubico, por tanto el ajuste de la regresión spline solo va a tener un nodo.

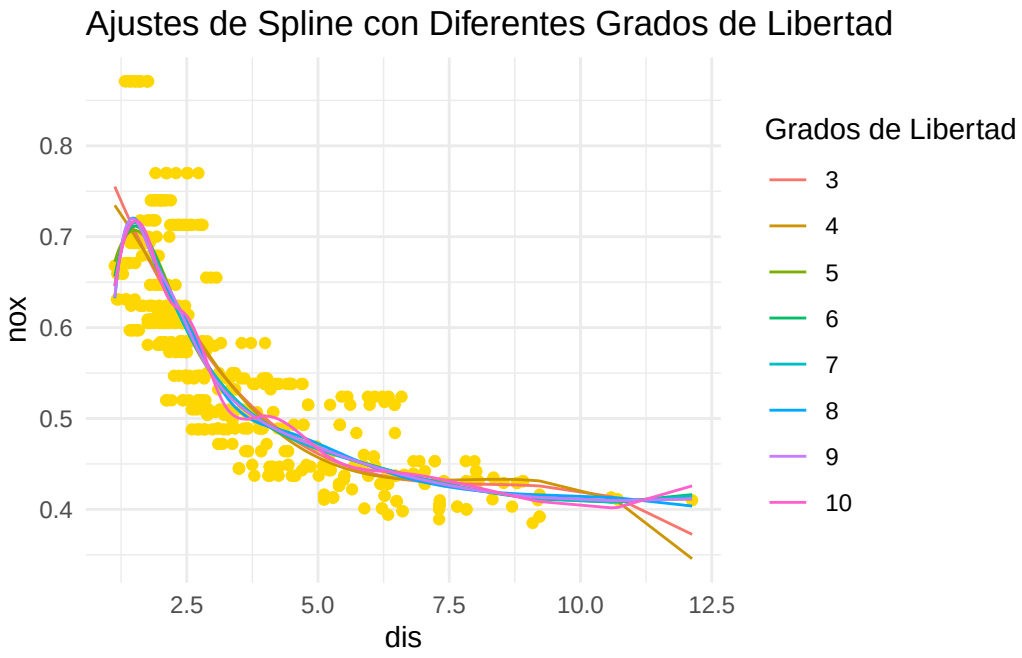
```r
modelo_splines <- lm(nox ~ bs(dis, df = 4, degree = 3), data = datos2)
attr(bs(datos2$dis, df = 4, degree = 3), which = "knots")
```

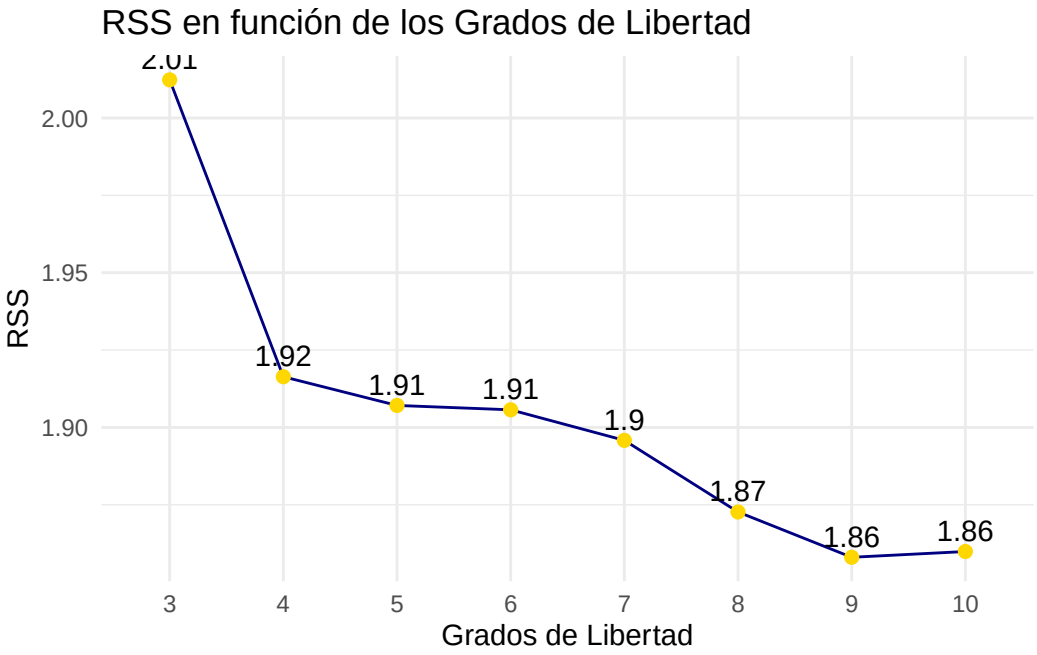
```
[1] 3.20745
```



e)

Se utilizo la función bs() para los ajustes del spline de regresión considerando un rango de 3 a 10 grados de libertad, el grado por defecto es 3 (cubic spline).



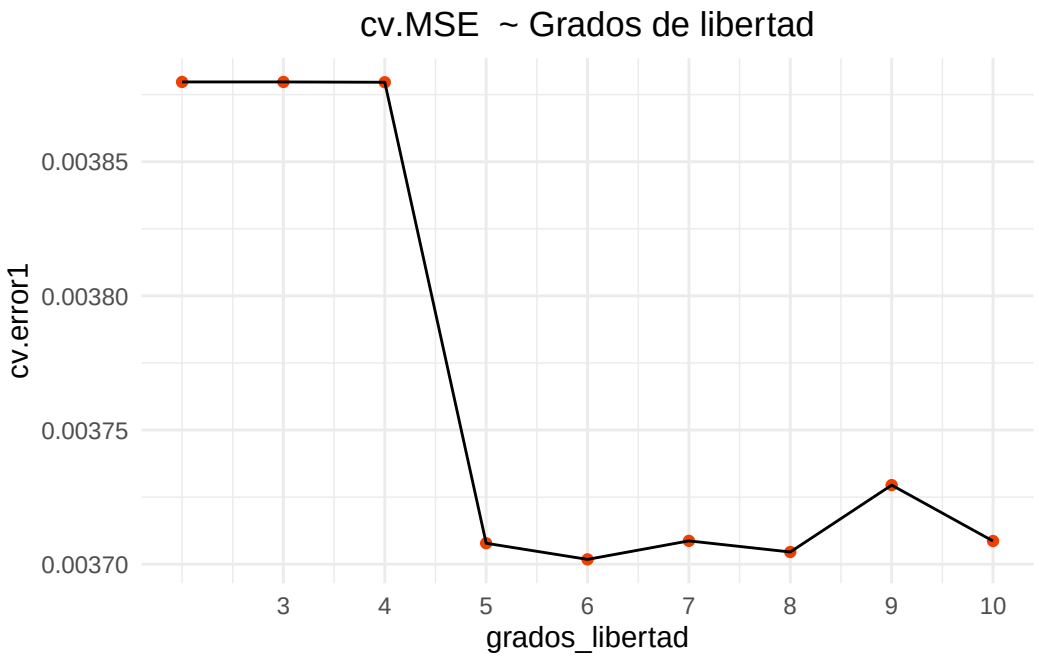


El gráfico muestra cómo el RSS disminuye a medida que aumenta el número de grados de libertad en la spline cúbica de regresión. Esto es típico en modelos más flexibles, ya que pueden adaptarse mejor a los datos, reduciendo la discrepancia entre los valores observados y los valores ajustados. Sin embargo, es importante encontrar un equilibrio, ya que demasiados grados de libertad pueden llevar a un sobreajuste del modelo. La elección del número óptimo de grados de libertad depende de consideraciones estadísticas y prácticas.

f)

Emplearemos el método de 10-fold cross validation para encontrar el valor óptimo de grados de libertad, probando valores de entre 3 y 10. Puesto que el valor óptimo de polinomio escogido fue de 3, será necesario especificar el argumento degree, ya que el spline se ajusta automáticamente con un grado de polinomio 3.

```
# Vector donde se almacenarán los errores de validación
cv.error1 <- rep(NA, 10)
# K-fold cross validation para cada valor de df
for(i in 2:10){
modelo.spline1 <- glm(nox ~ bs(dis, degree = 3, df = i),
                      data = datos2)
set.seed(123)
cv.error1[i] = cv.glm(datos2, modelo.spline1, K = 10)$delta[1]
}
```




```
which.min(cv.error1)
```

```
[1] 6
```

En este caso, se han identificado 6 grados de libertad como la elección óptima para minimizar el error de validación. Esta elección se traduce en la utilización de 3 knots, dado que el modelo de spline seleccionado es cúbico.