Untitled

Jhonatan Smith Garcia

22/12/2021

```
require(ISLR)
```

```
## Loading required package: ISLR
```

La base de datos a trabajar "College" recopila informacion de un gran numero de universidades del año 1995.

```
college= College
datos = na.omit(college)
names(datos)
```

```
##  [1] "Private"     "Apps"        "Accept"      "Enroll"      "Top10perc"
##  [6] "Top25perc"   "F.Undergrad" "P.Undergrad" "Outstate"    "Room.Board"
## [11] "Books"       "Personal"    "PhD"         "Terminal"    "S.F.Ratio"
## [16] "perc.alumni" "Expend"      "Grad.Rate"
```

La base de datos tiene un total de 777 observaciones con 18 variables.

A continuacion se presenta un breve resumen de las 18 variables:

1) Private: Una variable tipo factor que toma dos posibles valores; "YES" y "NO" indicando si la universidad es de caracter privado.
2) Apps: Numero de aplicantes postulados a la universidad
3) Accept: Numero de aplicantes recibidos a la universidad
4) Enroll: Numero de estudiantes nuevos matriculados
5) Top10perc: Porcentaje de estudiantes por encima del 10% de la clasificacion H.S [1]
6) Top25perc: Porcentaje estudiantes por encima del 25% de la clasificacion H.S
7) F.Undergrad: Numero de estudiantes a tiempo completo de pregrado
8) P.Undergradd: Numero de estudiantes a tiempo parcial de pregrado
9) Outstate: Matricula fuera del estado.
10) Room.Board: Costos de la habitacion y sostenimiento
11) Books: Costo estimado en libros.
12) Personal: Costo estimado gastos personales
13) PhD: Porcentaje de profesores con Doctorados.
14) Terminal: porcentaje de profesors con "terminal degree"[2]
15) S.F.Ratio: Proporcion Estudiantes/docentes
16) perc.alumni: Porcentajes de ex alumnos que donan a la universidad
17) Expend: gastos de educacion por estudiantes

18) grad.rate: tasa de graduacion

1 Una manera de evaluacion rendimiento academico en estados unidos. https://blog.prepscholar.com/what-is-class-rank-why-is-it-important

2 Un terminal degree es una forma de evaluar el rendimiento en la carrera como docente. Es, generalmente otorgado por la universidad como un titulo cuando el profesional no decide comenzar su trabajo doctoral.https://en.wikipedia.org/wiki/Terminal_degree

Division de los datos:

Se procede a particionar los datos entre conjunto de entrenamiento y prueba.

Tenga presente que los metodos no utilizan variables categoricas, por tanto se elmminará la variable *Private* para aplicar los metodos PCR y PLS.

```
datos = na.omit(college[-1])
set.seed(1998)
proporcion = 0.7
train = sample(1:nrow(datos), size = nrow(datos)*proporcion)
test = -train
```

La idea detras de esta division es usar validacion cruzada (CV) para aplicar los siguientes metodos.

Modelo PCR

Se busca predecir el numero de solicitues recibidas en funcion de las otras variables del conjunto de datos. Es decir:

$$Y_i = \theta_0 + \theta_1 Z_{i1} + \theta_2 Z_{i2} + \cdots + \theta_M Z_{iM} + \epsilon_i$$

Donde M son el numero de componentes principales seleccionando a travez de validacion cruzada. Entonces:

```
set.seed(1998)
require(pls)
```

```
## Loading required package: pls
```

```
##
## Attaching package: 'pls'
```

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

```
attach(datos)
pcr.fit = pcr(Apps~., data = datos, subset= train, scale=TRUE, validation = "CV")# PCR de train
summary(pcr.fit)
```

```
## Data:    X dimension: 543 16
##  Y dimension: 543 1
## Fit method: svdpc
## Number of components considered: 16
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
```

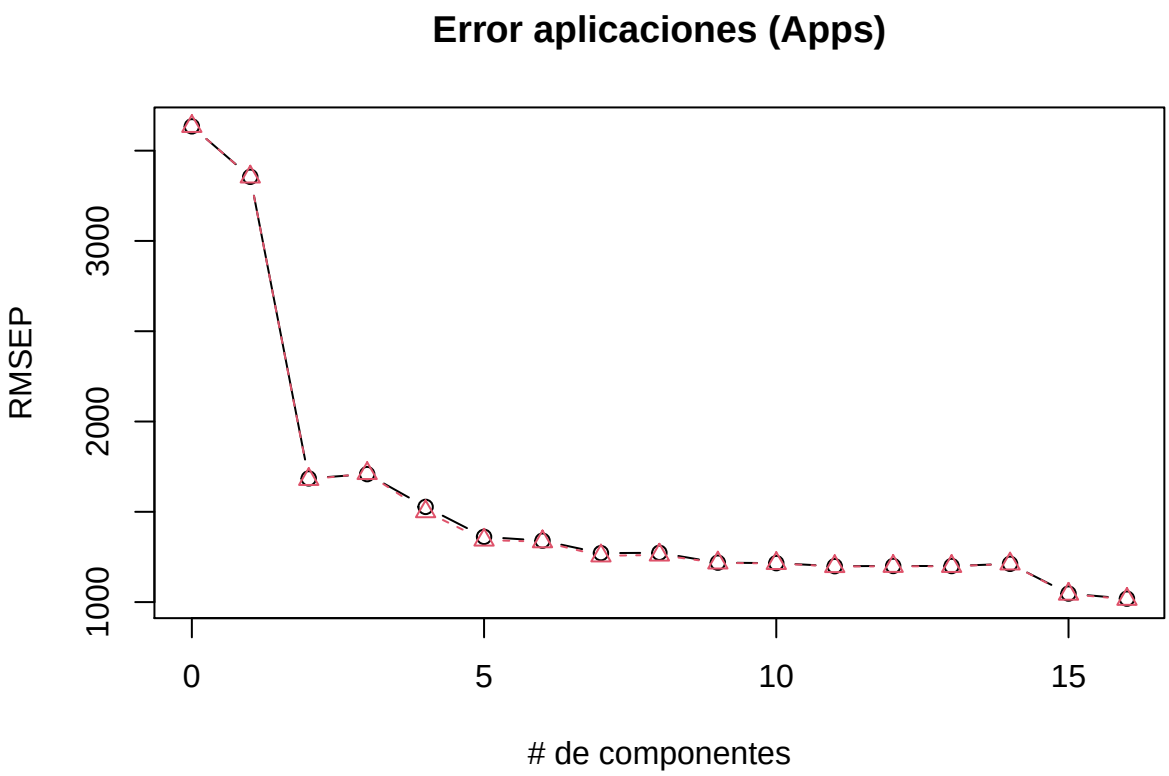
```
## CV            3635     3355     1685     1709     1527     1361     1340
## adjCV         3635     3355     1680     1712     1501     1344     1334
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1271     1273     1218      1216      1199      1200      1200
## adjCV     1256     1261     1217      1214      1197      1198      1198
##        14 comps  15 comps  16 comps
## CV         1212      1047      1019
## adjCV      1212      1044      1016
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       33.52    57.41    64.53    70.48    76.12    81.41    85.21    88.80
## Apps    15.79    79.61    79.62    84.81    87.50    87.52    89.06    89.08
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       92.20     94.72     96.50     97.85     98.81     99.34     99.80
## Apps    89.26     89.35     89.65     89.72     89.75     89.75     92.47
##       16 comps
## X          100
## Apps        93
```

Al observar el % de varianza explicada con los datos de entrenamiento, se observa que los componentes 7,8,9,10,11,12,14 y 14 son practicamente identicos salvo algunas diferencias muy sutiles. Por ejemplo, la diferencia entre el componente 7 y el 14 es de un porcentaje de varianza explicada de 0.69 la pregunta es, ¿esta ganancia es suficiente? ¿es mucha o poca esa ganancia? Dependiendo de la naturaleza del problema se decide, sin embargo; se opta por seleccionar en este caso la componente 7.

Seleccionando en un principio la componente 7, se tendra que la variabza explicada para las X's es de 85.21 y para la Y es de 89.06

Para la correcta seleccion, se hace un analisis grafico: Esta es la grafica de la raiz cuadrada de la media cuadratica del error de entrenamiento

```
set.seed(1998)
validationplot(pcr.fit,val.type = "RMSEP", type = "b", xlab = "# de componentes", main = "Error aplicac
```



Segun esto, se podria concluir que el mas optimo seria aquel que suelte un error mas bajo. En este caso, seria con 15/16 componentes (son muy parecidos).

Pero entonces, no serviria de mucho pues el proceso de reduccion de dimensionalidad seria nulo. Ademas, ¿Es mucha la diferencia entre los errores entre 7 y 15/16 componentes? Esto habria que constatarlo con un experto pero en un principio, no se ve mucha diferencia. Por lo anterior se decide trabajar con 7 componentes, intentando llevar siempre un principio de parsimonia.

```
X = model.matrix(Apps~., datos)[,-1]
Y = datos$Apps
y.test = Y[test]
set.seed(1998)
pcr.pred.1 = predict(pcr.fit, X[test,], ncomp = 7)
mean((pcr.pred.1-y.test)^2) # Prediccion error datos de prueba
```

```
## [1] 3964517
```

Este es el error estimado con los datos de prueba. Ahora, se procede a ajustar nuevamente el modelo PCR con ayuda del M (7) hallado por CV.

```
pcr.fit.2 = pcr(Y~X,scale=T, ncomp = 7 )
summary(pcr.fit.2)
```

```
## Data:    X dimension: 777 16
##  Y dimension: 777 1
## Fit method: svdpc
## Number of components considered: 7
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
```

```
## X    32.77    57.08    64.39    70.22    75.96    81.25    85.03
## Y    10.83    74.28    74.53    74.62    83.94    84.06    84.13
```

Y este seria finalmente el modelo con los datos completos, usando 7 componentes y obtener un porcentaje de varianza explicado en X de 85.03% y en Y de 84.13%

Modelo PLS:

```
set.seed(1998)
pls.fit = plsr(Apps~., data=datos, subset = train, scale =T, validation = "CV") # PCL en datos de entre
summary(pls.fit)
```

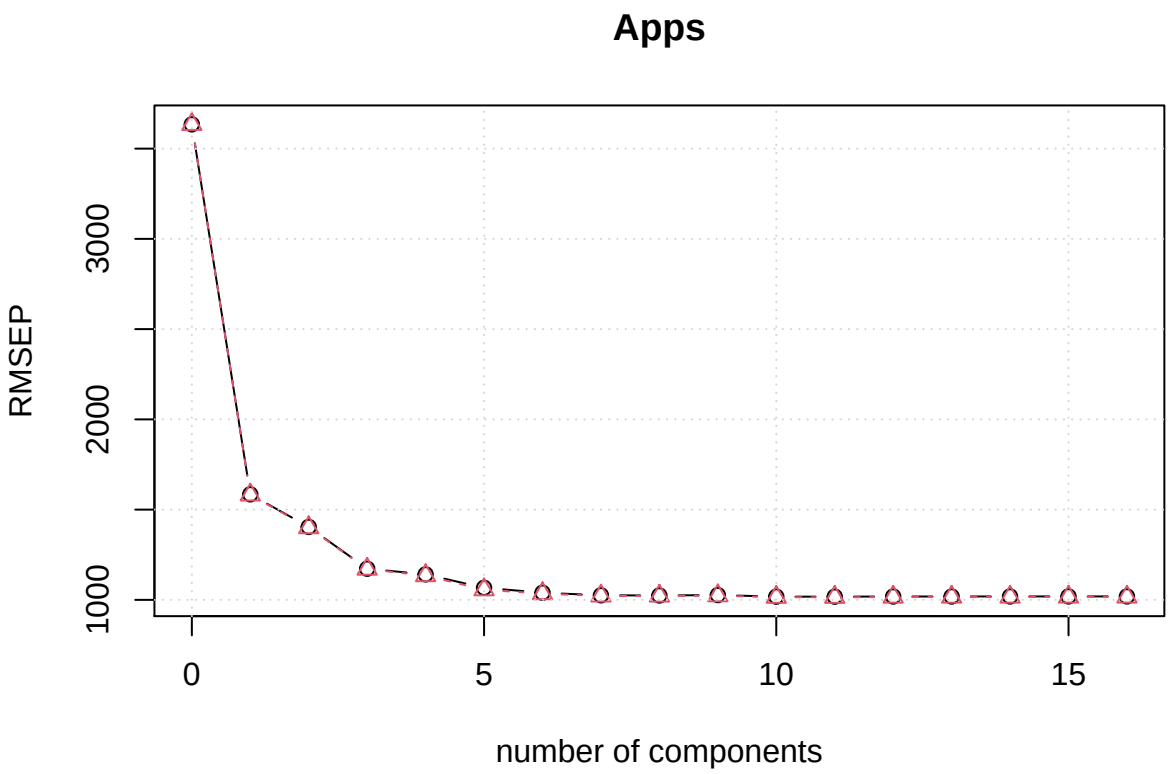
```
## Data:    X dimension: 543 16
##  Y dimension: 543 1
## Fit method: kernelpls
## Number of components considered: 16
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3635     1585     1404     1173     1141     1067     1040
## adjCV         3635     1582     1400     1170     1134     1056     1034
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1025     1024     1026      1018      1018      1019      1019
## adjCV     1022     1021     1023      1015      1014      1015      1015
##        14 comps  15 comps  16 comps
## CV         1019      1019      1019
## adjCV      1016      1016      1016
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       26.93    54.81    62.95    66.10    68.18    72.52    75.84    81.38
## Apps    81.67    86.02    90.13    91.32    92.51    92.76    92.89    92.91
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.98     85.73     88.57     92.13     94.73     96.05     99.47
## Apps    92.95     92.99     93.00     93.00     93.00     93.00     93.00
##       16 comps
## X          100
## Apps        93
```

Observando lo anterior, se puede observar que, la diferencia entre los 16 componentes y solo 3 componentes es de cerca del 2%. Se recuerda que, dependiendo del contexto del problema, ese 2% de diferencia podria o no ser sustancial. Pero para efectos practicos, se supone en este caso irrelevante esa diferencia.

Ahora, teng apresente algo; el interes del problema radica en explicar la variabilidad de "Apps", la variable problema, no de las X's entre si.

Con 3 componentes, la variabilidad de interes ya es explicada de manera eficiente. ¿Importa explicar la variabilidad de las X's? Bueno, dado el caso que si, entonces no se escogeria el modelo con 3 componentes, los candidatos serian entonces 10,11,12 componentes. Esto asignando una variabilidad explicada mayor al 85% en las X's.

```
set.seed(1998)
validationplot(pls.fit, val.type = "RMSEP", type = "b")
grid()
```



Segun este grafico, lo recomndable seria tomar 10 componentes.

Al analizar nuevamente los resultados anteriores se tiene que con 10 componentes el eeror obtenido es de 1018, la variabilidad de las X's es de 85.73% y de la Y es de 92.99%

Aun asi, esto no queda muy claro, debido a que entre el 8 y el 16 casi que toman valores iguales entonces al analizaar los valores de los RSMEP y los porcentajes de variabilidad explicada para las componentes de las 16 se tiene que:

a) El RSMEP se "estabiliza" a partir de 10 componentes (1016).

b) El RSEMO no es muy diferente a partir de las componentes 5 en adelante. Aparenta ser "poco" dicho error. Ahora, ¿Es poco? Acorde a la naturaleza del problema y nuevamente, con ayuda de un experto, se podria dictaminar si es o no alta esa diferencia. En este caso, se opta por considerar que el RSMEP no difiere mucho a partir de 5 componentes

c) Suponga que se desea tener alta interpretabilidad de la varianza para las X's y para la Y. Suponga tambien que alta interpretabilidad es a partir del 85% (podria ser 90, 95, 80, depende del contexto).

d) Con lo anterior se establece que el numero optimo de componentes serán 10, como ya se habia previamente sospechado.

Para comparar rendimiento, se procede a realizar error de prueba o test.

```
set.seed(1998)
pls.pred = predict(pls.fit, X[test,], ncomp = 10) # Prediccion con 10 componentes
mean((pls.pred- y.test)^2)
```

```
## [1] 1693891
```

Este es el error obtenido.

```r
pcl.fit.3 = plsr(Y~X, scale= T, ncomp = 10)
summary(pcl.fit.3)
```

```
## Data:    X dimension: 777 16
##  Y dimension: 777 1
## Fit method: kernelpls
## Number of components considered: 10
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    26.16    52.39    62.56    65.36    69.34    74.09    76.80    79.73
## Y    77.97    82.66    87.65    90.56    92.12    92.57    92.64    92.70
##    9 comps  10 comps
## X    82.60     85.35
## Y    92.75     92.77
```

Planteando el modelo con todos los datos y 10 componentes se obtiene una variabilidad de las X's de 85.35 y de la Y de un 92.77.

Note que si el interes fuese solo eplicar Y, un modelo con 3 componentes bastaria para explicar mas del 85% de variabilidad.