Práctica 7

Principal Components Regression

Extensión de Ch6-varselect-lab.R, DEL LIBRO

Continuación de la práctica anterior de Lasso y Ridge.

RECUERDA:

Algunas de las ventajas más destacables de realizar PCR son las siguientes:

– Reducción de dimensionalidad

– Evitar multicolinealidad entre predictores.

```
library(ISLR2)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
library(pls)
```

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

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

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
##
## Attaching package: 'corrplot'
```

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

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

```
##      R2
```

```
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

```
dim(Hitters)
```

```
## [1] 322  20
```

Hay que eliminar las muestras con NA en Salary, y definir x, y, como en la práctica anterior.

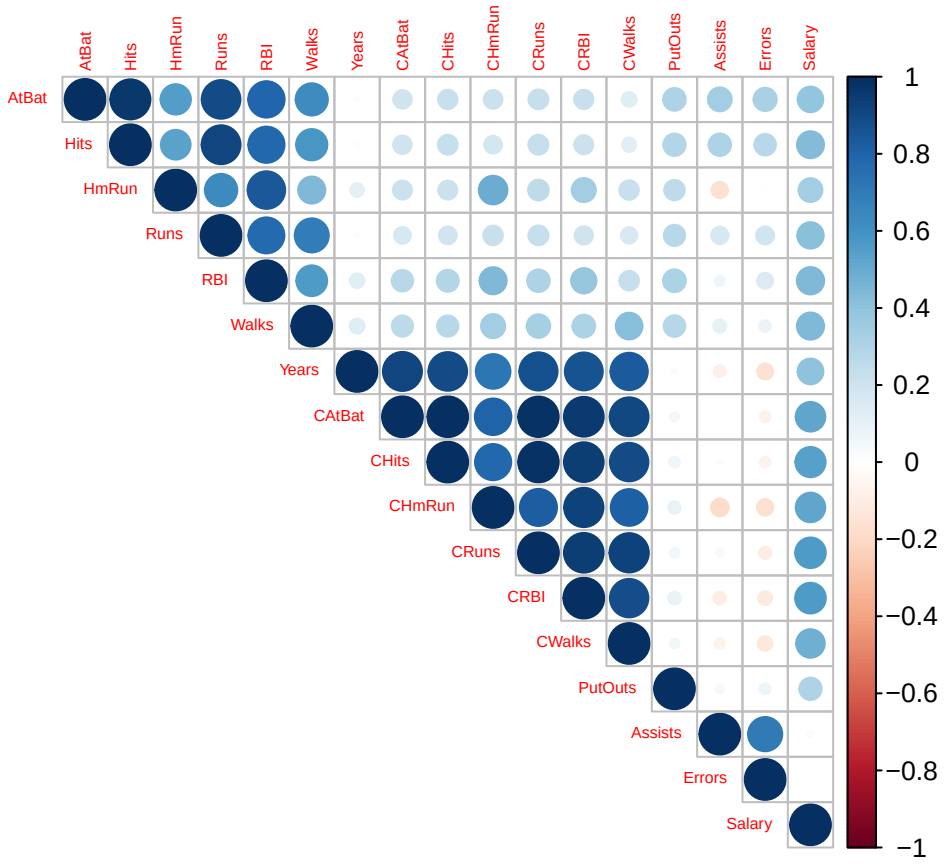
```
attach(Hitters)
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

```
Hitters <- na.omit(Hitters)
```

Sólo por curiosidad, para aquellas personas que no conozcan la función findCorrelation:

```
corrplot(cor(Hitters[, c(-14, -15, -20)]), type = "upper", tl.cex = 0.5)
```




```
findCorrelation(cor(Hitters[, c(-14, -15, -20)]), names = TRUE)
```

```
## [1] "CRBI"   "CRuns"  "CHits"  "CAtBat" "AtBat"  "Hits"
```

```
# B2=subset(Hitters,select=-findCorrelation(cor(Hitters[,c(-14,-15,-20)])))
```

La función pcr realica regresión con componentes principales.

```
?pcr
```

```
## starting httpd help server ... done
```

```
set.seed(2)
pcr.fit <- pcr(Salary ~ ., data = Hitters, scale = TRUE, validation = "CV")
```

La sintaxis de la función pcr() es similar a la de lm(), con algunas opciones adicionales.

Escala = VERDADERO: se estandarizan los predictores (vease 6.6), antes de generar los componentes principales, de modo que la escala en la que cada variable se mide no tendrá ningún efecto.

Validación = "CV" hace que pcr() calcule con 10-fold CV validation el error para cada valor posible de M, número de componentes principales utilizadas.

```
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    351.9    353.2    355.0    352.8    348.4    343.6
## adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
## adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        349.7     349.4     339.9     341.6     339.2     339.6
## adjCV     348.0     347.7     338.2     339.7     337.2     337.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

El CV score se da desde M = 0. IMPORTANTE: pcr() proporciona la raíz cuadrada del MSE Para obtener el MSE habitual, debemos elevar al cuadrado esta cantidad.

La función summary() proporciona el porcentaje de varianza explicado en los predictores y en la respuesta utilizando diferentes componentes.

Brevemente, podemos pensar en esto como la cantidad de información sobre los predictores. o la respuesta que se captura utilizando M componentes principales.

Por ejemplo, M = 1 solo captura el 38,31 % de toda la varianza o información en los predictores. En cambio, usar M = 5 aumenta el valor a 84,29 %.

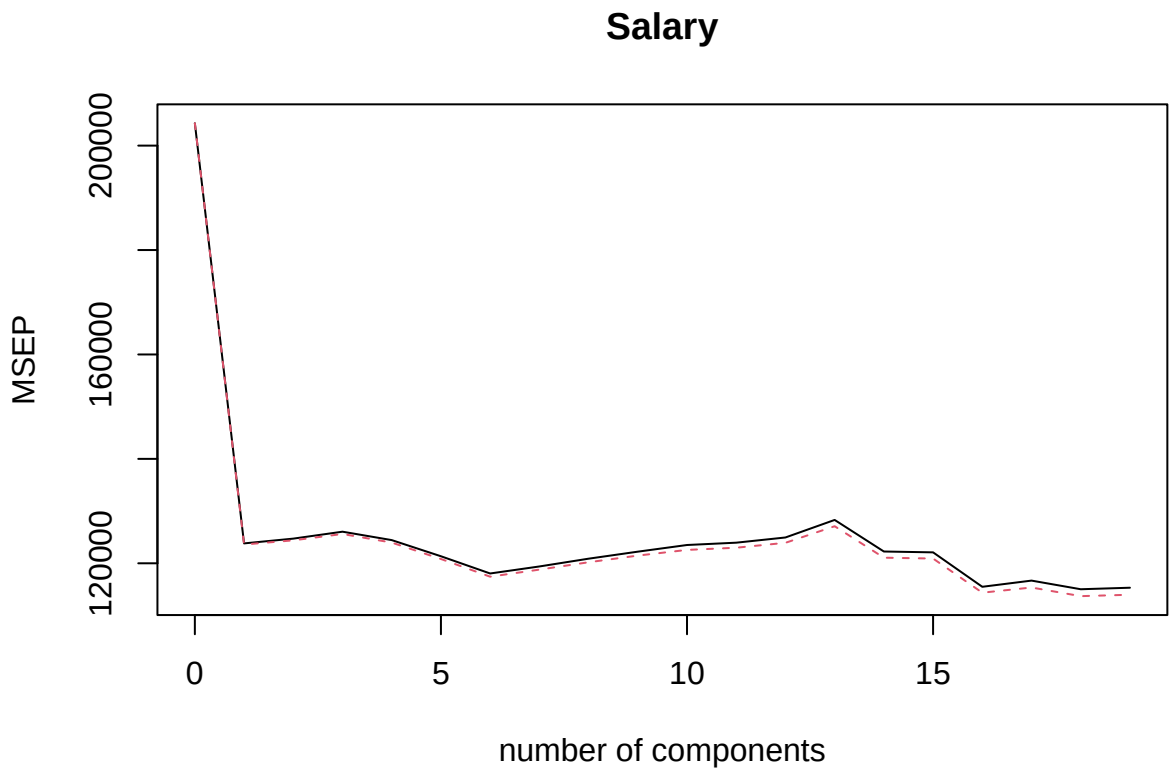
Si usáramos todos, M = p = 19, componentes, esto aumentaría al 100 %.

También se pueden trazar las puntuaciones de la validación cruzada utilizando validationplot() función. Podemos visualizar los MSE con val.type = "MSEP":

```
MSEP(pcr.fit)
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          204285   123813   124726   126055   124445   121351   118048
## adjCV       204285   123596   124400   125615   123968   120821   117436
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      119393   120887   122229    123506    123950    124945    128306
## adjCV   118789   120202   121475    122565    122985    123926    127103
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       122260    122094    115502    116686    115030    115322
## adjCV    121073    120911    114374    115364    113710    113978
```

```
validationplot(pcr.fit, val.type = "MSEP")
```



Vemos que el menor error de validación cruzada más pequeño es con M = 18 componentes, lo que equivale simplemente a realizar mínimos cuadrados,

```
which.min(MSEP(pcr.fit)$val[1, 1, ]) - 1
```

```
## 18 comps
##       18
```

Sin embargo, en el gráfico también vemos que el error de validación cruzada es aproximadamente el mismo cuando sólo se incluye pocas componentes en el modelo. Esto sugiere que un modelo que utiliza sólo una pequeña cantidad de componentes podría ser suficiente.

Más información que me da la función pcr (la he puesto en # para que el pdf generado no fuera muy extenso).

```
# pcr.fit$coefficients #coeficientes de la regresión en las variables originales
# pcr.fit$loadings #los pesos que proporcionand las componenetes en c.l. de las originales
# pcr.fit$scores #los componentes principales como variables
# pcr.fit$fitted.values #la predicción por cada observación, dependiendo del número de componentes
```

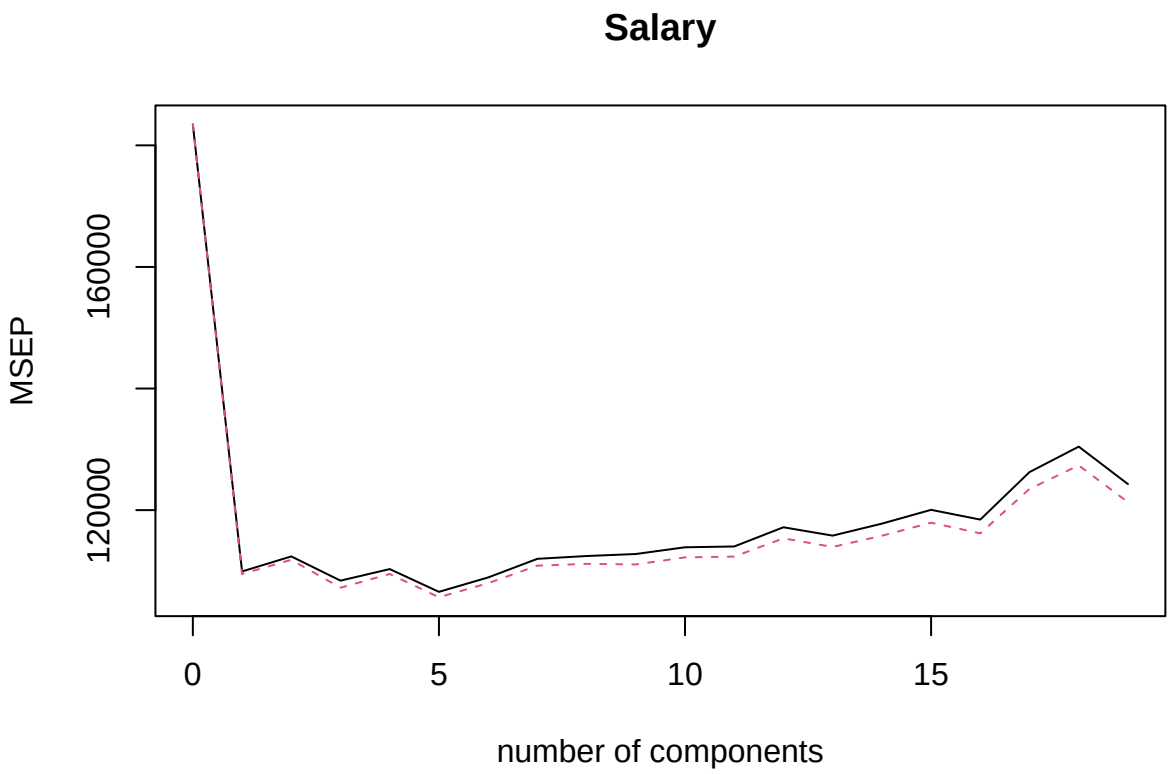
Ahora realizamos PCR en los datos de entrenamiento y evaluamos el rendimiento del conjunto de pruebas.

```
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
```

```
set.seed(1)
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]
pcr.fit2 <- pcr(Salary ~ ., data = Hitters, subset = train, scale = TRUE, validation = "CV")
summary(pcr.fit2)
```

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    331.5    335.2    329.2    332.1    326.4    330.0
## adjCV        428.3    330.9    334.4    327.5    330.9    325.1    328.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       334.6    335.3    335.8     337.5     337.7     342.3     340.3
## adjCV    333.0    333.4    333.3     335.0     335.2     339.6     337.6
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        343.2     346.5     344.2     355.3     361.2     352.5
## adjCV     340.3     343.4     340.9     351.4     356.9     348.3
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         39.32    61.57    71.96    80.83    85.95    89.99    93.25    95.34
## Salary    43.87    43.93    47.36    47.37    49.52    49.55    49.63    50.98
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.55     97.61     98.28     98.85     99.22     99.53     99.79
## Salary    53.00     53.00     53.02     53.05     53.80     53.85     54.03
##         16 comps  17 comps  18 comps  19 comps
## X          99.91     99.97     99.99    100.00
## Salary     55.85     55.89     56.21     58.62
```

```
validationplot(pcr.fit2, val.type = "MSEP")
```



Encontramos que el error de validación cruzada más bajo ocurre cuando M = 3 ó 5 (este número puede variar dependiendo del conjunto de entrenamiento) que se utilicen los componentes. Calculamos el MSE con el conjunto de validación de la siguiente manera:

```
pcr.pred2 <- predict(pcr.fit2, x[test, ], ncomp = 5)
mean((pcr.pred2 - y.test)^2)
```

```
## [1] 142811.8
```

Este conjunto de pruebas MSE es competitivo con los resultados obtenidos mediante la regresión de Ridge y Lasso. Sin embargo, como resultado de la forma en que se implementa el PCR, el modelo final es más difícil interpretar porque no realiza ningún tipo de selección de variables.

Finalmente, ajustamos la PCR al conjunto de datos completo, usando M = 5, el número de componentes identificados mediante validación cruzada.

```
pcr.fit3 <- pcr(y ~ x, scale = TRUE, ncomp = 5)
summary(pcr.fit3)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X    38.31    60.16    70.84    79.03    84.29
## y    40.63    41.58    42.17    43.22    44.90
```

```r
head(predict(pcr.fit3)) # esto me da lo mismo que pcr.fit3$fitted.values
```

```
## , , 1 comps
##
##          y
## 1 534.8996
## 2 579.6895
## 3 904.6869
## 4 263.8013
## 5 645.2411
## 6 112.5090
##
## , , 2 comps
##
##           y
## 1 494.48973
## 2 632.17353
## 3 886.84350
## 4 268.79132
## 5 679.23504
## 6  79.94653
##
## , , 3 comps
##
##           y
## 1 525.22549
## 2 610.08536
## 3 900.33822
## 4 281.41564
## 5 711.57997
## 6  76.16928
##
## , , 4 comps
##
##           y
## 1 559.82506
## 2 619.85749
## 3 960.13423
## 4 361.90615
## 5 582.35125
## 6  62.49043
##
## , , 5 comps
##
##           y
## 1  495.0068
## 2  547.8896
## 3 1010.2236
## 4  409.8232
## 5  524.9053
## 6  133.1325
```

```r
# pcr.fit3$fitted.values
```

```
coef(pcr.fit3)
```

```
## , , 5 comps
##
##                     y
## AtBat       28.766042
## Hits        30.447021
## HmRun       25.844498
## Runs        33.000876
## RBI         33.819966
## Walks       35.087794
## Years       22.351033
## CAtBat      29.014768
## CHits       29.785842
## CHmRun      30.002014
## CRuns       32.069124
## CRBI        31.112315
## CWalks      31.487349
## LeagueN     19.438996
## DivisionW  -63.203872
## PutOuts     17.360440
## Assists     -5.523264
## Errors      -6.044002
## NewLeagueN  21.742668
```

LO SIGUIENTE CORRESPONDE A Ch12-unsup-lab.R DEL LIBRO, CON APUNTES TEÓRICOS ADICIONALES

Principal Components Analysis

Consideremos otra nueva base de datos.

```
states <- row.names(USArrests)
states
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
```

```
# View(USArrests)
names(USArrests)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

Vemos que las medias de cada columna de la base de datos son muy diferente.

```r
apply(USArrests, 2, mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

También podemos ver sus varianzas.

```r
apply(USArrests, 2, var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

Observamos que son muy diferentes, entre otras cosas porque hay diferentes escalas; urbanpop es un tanto por ciento. por ello, hay que poner en el PCA, scale=TRUE, y así, además de restar la media, hacemos que todas las variables tengan varianza igual a 1. Se consigue dividiendo por la desviación típica.

```r
pr.out <- prcomp(USArrests, scale = TRUE)
```

Veamos las direcciones principales (autovectores ortonormales). ¿Los podríais interpretar?

```r
pr.out
```

```
## Standard deviations (1, .., p=4):
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
##
## Rotation (n x k) = (4 x 4):
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995 -0.4181809  0.3412327  0.64922780
## Assault  -0.5831836 -0.1879856  0.2681484 -0.74340748
## UrbanPop -0.2781909  0.8728062  0.3780158  0.13387773
## Rape     -0.5434321  0.1673186 -0.8177779  0.08902432
```

Más información:

```r
summary(pr.out)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.5749 0.9949 0.59713 0.41645
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

Veamos qué es todo esto: Empecemos por la información que da el pr.out directamente. La desviación típica es de las componentes principales Z_i (i.e. las nuevas variables). Lo verifiquemos:

```r
attach(USArrests)
sd(0.5358995 * (Murder - mean(Murder)) / sd(Murder) + 0.5831836 * (Assault - mean(Assault)) / sd(Assaul
```

```
## [1] 1.574878
```

```r
## o lo que es lo mismo
sd(pr.out$x[, 1])
```

```
## [1] 1.574878
```

x te proporciona las componentes principales como nuevas variables (scores)

```r
dim(pr.out$x)
```

```
## [1] 50  4
```

```
pr.out$x
```

```
##                       PC1         PC2         PC3          PC4
## Alabama        -0.97566045 -1.12200121  0.43980366  0.154696581
## Alaska         -1.93053788 -1.06242692 -2.01950027 -0.434175454
## Arizona        -1.74544285  0.73845954 -0.05423025 -0.826264240
## Arkansas        0.13999894 -1.10854226 -0.11342217 -0.180973554
## California     -2.49861285  1.52742672 -0.59254100 -0.338559240
## Colorado       -1.49934074  0.97762966 -1.08400162  0.001450164
## Connecticut     1.34499236  1.07798362  0.63679250 -0.117278736
## Delaware       -0.04722981  0.32208890  0.71141032 -0.873113315
## Florida        -2.98275967 -0.03883425  0.57103206 -0.095317042
## Georgia        -1.62280742 -1.26608838  0.33901818  1.065974459
## Hawaii          0.90348448  1.55467609 -0.05027151  0.893733198
## Idaho           1.62331903 -0.20885253 -0.25719021 -0.494087852
## Illinois       -1.36505197  0.67498834  0.67068647 -0.120794916
## Indiana         0.50038122  0.15003926 -0.22576277  0.420397595
## Iowa            2.23099579  0.10300828 -0.16291036  0.017379470
## Kansas          0.78887206  0.26744941 -0.02529648  0.204421034
## Kentucky        0.74331256 -0.94880748  0.02808429  0.663817237
## Louisiana      -1.54909076 -0.86230011  0.77560598  0.450157791
## Maine           2.37274014 -0.37260865  0.06502225 -0.327138529
## Maryland       -1.74564663 -0.42335704  0.15566968 -0.553450589
## Massachusetts   0.48128007  1.45967706  0.60337172 -0.177793902
## Michigan       -2.08725025  0.15383500 -0.38100046  0.101343128
## Minnesota       1.67566951  0.62590670 -0.15153200  0.066640316
## Mississippi    -0.98647919 -2.36973712  0.73336290  0.213342049
## Missouri       -0.68978426  0.26070794 -0.37365033  0.223554811
## Montana         1.17353751 -0.53147851 -0.24440796  0.122498555
## Nebraska        1.25291625  0.19200440 -0.17380930  0.015733156
## Nevada         -2.84550542  0.76780502 -1.15168793  0.311354436
## New Hampshire   2.35995585  0.01790055 -0.03648498 -0.032804291
## New Jersey     -0.17974128  1.43493745  0.75677041  0.240936580
## New Mexico     -1.96012351 -0.14141308 -0.18184598 -0.336121113
## New York       -1.66566662  0.81491072  0.63661186 -0.013348844
## North Carolina -1.11208808 -2.20561081  0.85489245 -0.944789648
## North Dakota    2.96215223 -0.59309738 -0.29824930 -0.251434626
## Ohio            0.22369436  0.73477837  0.03082616  0.469152817
## Oklahoma        0.30864928  0.28496113  0.01515592  0.010228476
## Oregon         -0.05852787  0.53596999 -0.93038718 -0.235390872
## Pennsylvania    0.87948680  0.56536050  0.39660218  0.355452378
## Rhode Island    0.85509072  1.47698328  1.35617705 -0.607402746
## South Carolina -1.30744986 -1.91397297  0.29751723 -0.130145378
## South Dakota    1.96779669 -0.81506822 -0.38538073 -0.108470512
## Tennessee      -0.98969377 -0.85160534 -0.18619262  0.646302674
## Texas          -1.34151838  0.40833518  0.48712332  0.636731051
## Utah            0.54503180  1.45671524 -0.29077592 -0.081486749
## Vermont         2.77325613 -1.38819435 -0.83280797 -0.143433697
## Virginia        0.09536670 -0.19772785 -0.01159482  0.209246429
## Washington      0.21472339  0.96037394 -0.61859067 -0.218628161
## West Virginia   2.08739306 -1.41052627 -0.10372163  0.130583080
## Wisconsin       2.05881199  0.60512507  0.13746933  0.182253407
## Wyoming         0.62310061 -0.31778662  0.23824049 -0.164976866
```

Además:

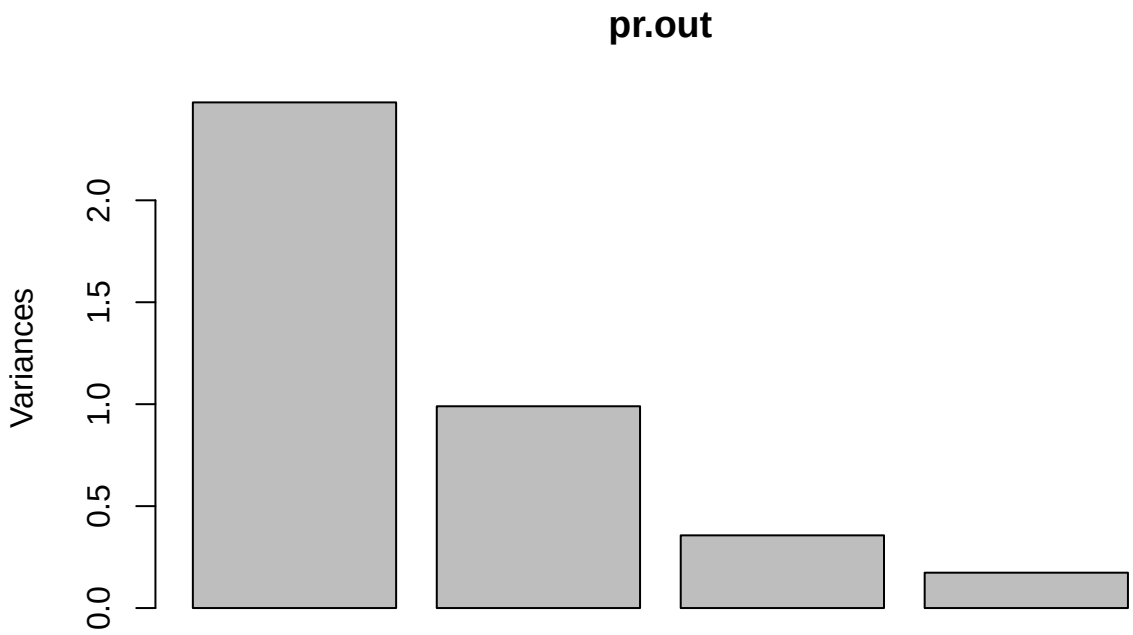
```
pr.out$sdev # sdev se corresponden a valores singulares
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
```

```
lasvar <- pr.out$sdev^2 # esto a los autovalores, es decir, las varianzas.
lasvar
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

```
screeplot(pr.out) # gráficamente
```



Proportion of Variance: PC1 expl elica 62.0% de la varianza en los datos, PC2 el 24.7 % , etc.

```
lasvar / sum(lasvar)
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

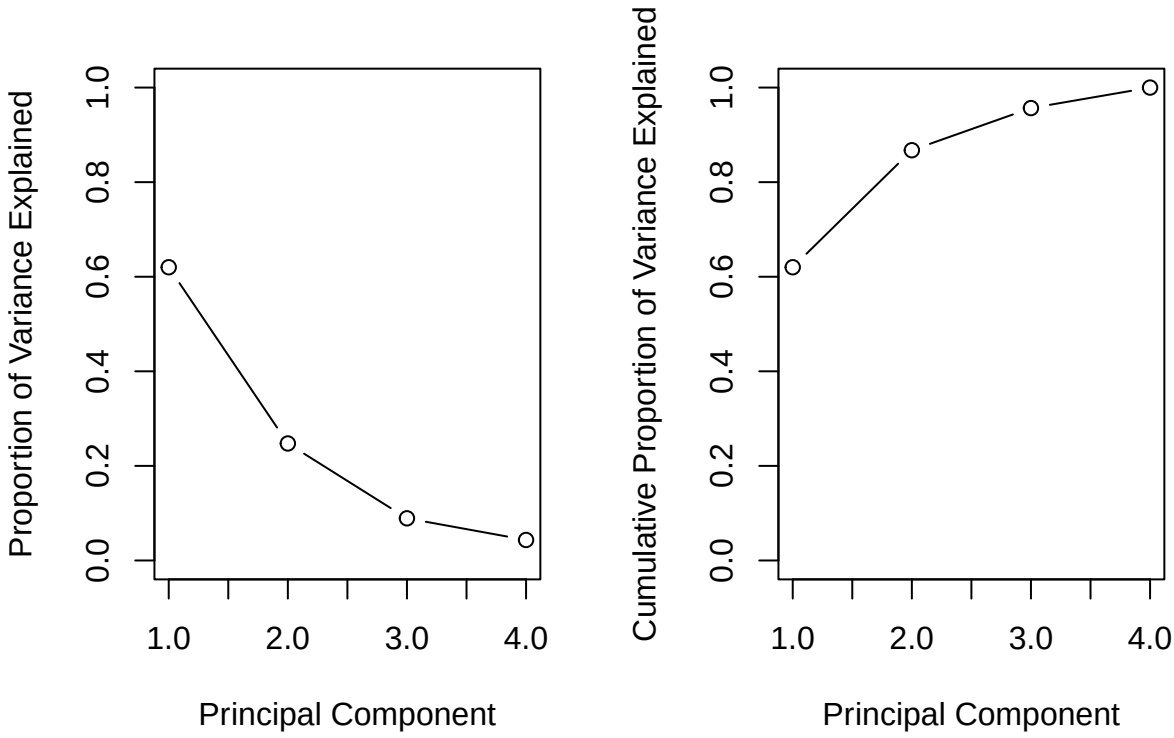
Gráficamente: PVE = proportion of variance explained

```
pve <- lasvar / sum(lasvar)
pve
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

```
par(mfrow = c(1, 2))
plot(pve,
     xlab = "Principal Component",
     ylab = "Proportion of Variance Explained", ylim = c(0, 1),
     type = "b"
```

```
)
plot(cumsum(pve),
     xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     ylim = c(0, 1), type = "b"
)
```



Sigamos: Center y Scale corresponden a media y desviación típica de las variables de partida

```
pr.out$center
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
apply(USArrests, 2, mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
pr.out$scale
```

```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

```
apply(USArrests, 2, sd)
```

```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

Rotation: proporciona los autovectores (loadings); igual que si ejecutamos sólo pr.out

```
pr.out$rotation
```

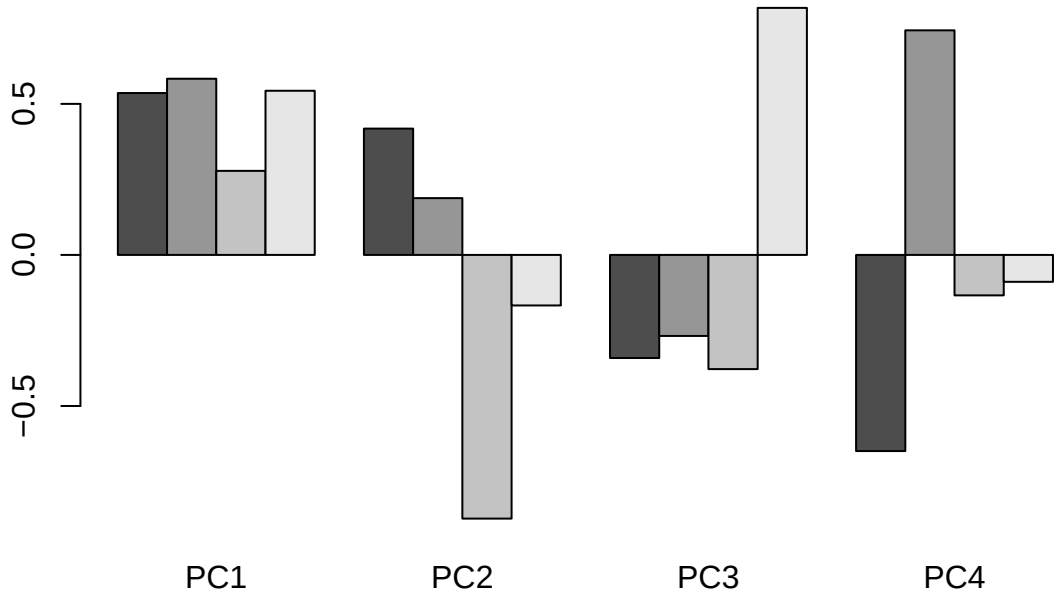
```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995 -0.4181809  0.3412327  0.64922780
## Assault  -0.5831836 -0.1879856  0.2681484 -0.74340748
## UrbanPop -0.2781909  0.8728062  0.3780158  0.13387773
## Rape     -0.5434321  0.1673186 -0.8177779  0.08902432
```

Cambiemos de signo los autovalores:

```
pr.out$rotation <- -pr.out$rotation
```

gráficamente:

```
barplot(pr.out$rotation, beside = TRUE)
```




```
pr.out$x <- -pr.out$x
```

Biplot: El biplot plotea por un lado las dos primeras componentes principales scale=0 : las flechas representa los dos primeros pesos de las componentes; es decir, Rape está en la coordenada (0.5434321 , 0.1673186).

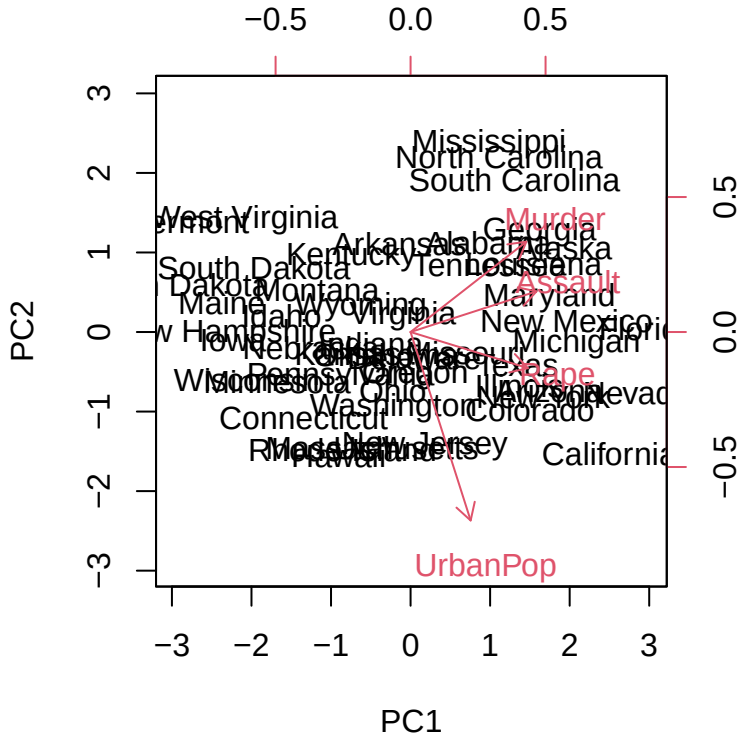
Por otro lado, en otra escala, se hace un plot de las dos primeros datos de las ciudades en las componentes principales. Permite interpretar el significado de las componentes (la primera en el eje horizontal y la segunda en el eje vertical)

Matemáticamente, las componentes principales, Z_i, son combinaciones lineales de las variables predictores iniciales X_i, y viceversa, cada predictor X_i puede ser escrita como combinación lineal de los predictores.

Esto se deduce de: X.V=Z donde X:: base de datos, en filas las observaciones; V::matriz cuyas columnas son los loadings, es decir, direcciones de componentes principales; Z:: componentes principales, es decir, las

nuevas variables que están correlacionadas.

```
biplot(pr.out, scale = 0)
```




```
pr.out
```

```
## Standard deviations (1, .., p=4):
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
##
## Rotation (n x k) = (4 x 4):
##                PC1        PC2        PC3         PC4
## Murder   0.5358995  0.4181809 -0.3412327 -0.64922780
## Assault  0.5831836  0.1879856 -0.2681484  0.74340748
## UrbanPop 0.2781909 -0.8728062 -0.3780158 -0.13387773
## Rape     0.5434321 -0.1673186  0.8177779 -0.08902432
```

```
head(pr.out$x)
```

```
##                  PC1        PC2         PC3          PC4
## Alabama     0.9756604  1.1220012 -0.43980366 -0.154696581
## Alaska      1.9305379  1.0624269  2.01950027  0.434175454
## Arizona     1.7454429 -0.7384595  0.05423025  0.826264240
## Arkansas   -0.1399989  1.1085423  0.11342217  0.180973554
## California  2.4986128 -1.5274267  0.59254100  0.338559240
## Colorado    1.4993407 -0.9776297  1.08400162 -0.001450164
```

La conclusión es que los datos podrían ser sustituidos por PC1 y PC2. O sea, la matriz de datos quedaría reducida a la mitad.

Análisis de componentes principales a través de la diagonalización de la matriz de correlación:

```
n <- nrow(USArrests)
n
```

```
## [1] 50
```

Hacer PCA con varianza igual a 1, equivale a hacer PCA con la matriz de correlación.

Podemos hacer la matriz de correlación "a mano" y así recordar qué significa:

```
a <- scale(USArrests$Murder) # escalamos las variables
b <- scale(USArrests$Assault)
c <- scale(USArrests$UrbanPop)
d <- scale(USArrests$Rape)
aver <- data.frame(a, b, c, d)
K <- as.matrix(aver)
Kt <- t(K) # matriz transpuesta de K
(Kt %*% K) / 49 # matriz de correlación
```

```
##            a         b          c         d
## a 1.00000000 0.8018733 0.06957262 0.5635788
## b 0.80187331 1.0000000 0.25887170 0.6652412
## c 0.06957262 0.2588717 1.00000000 0.4113412
## d 0.56357883 0.6652412 0.41134124 1.0000000
```

Una vez que lo hemos recordado, podemos seguir utilizando la orden cor:

```
S <- cor(USArrests)
S
```

```
##              Murder   Assault   UrbanPop      Rape
## Murder   1.00000000 0.8018733 0.06957262 0.5635788
## Assault  0.80187331 1.0000000 0.25887170 0.6652412
## UrbanPop 0.06957262 0.2588717 1.00000000 0.4113412
## Rape     0.56357883 0.6652412 0.41134124 1.0000000
```

```
auto <- eigen(S)
```

Sus autovalores son las varianzas obtenidas por el otro método:

```
auto$values
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

```
lasvar
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

Los autovectores son las direcciones de las componentes principales

```
auto$vectors
```

```
##            [,1]       [,2]       [,3]        [,4]
## [1,] -0.5358995  0.4181809 -0.3412327  0.64922780
## [2,] -0.5831836  0.1879856 -0.2681484 -0.74340748
## [3,] -0.2781909 -0.8728062 -0.3780158  0.13387773
## [4,] -0.5434321 -0.1673186  0.8177779  0.08902432
```

```
pr.out
```

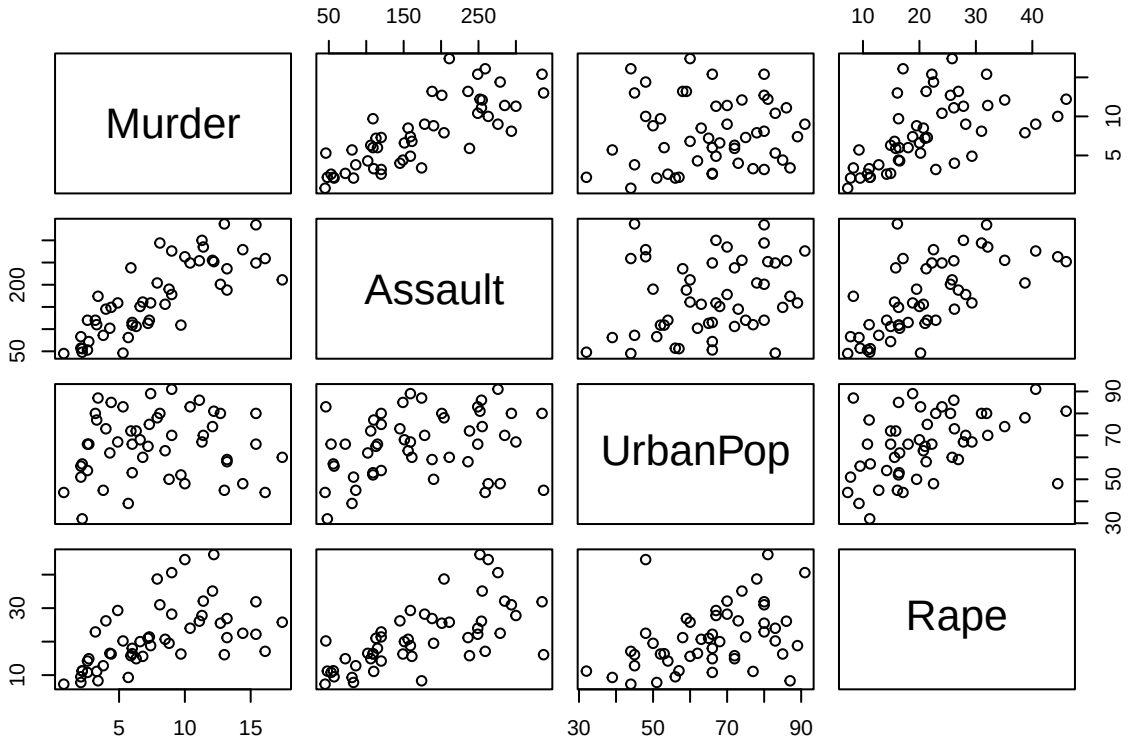
```
## Standard deviations (1, .., p=4):
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
##
## Rotation (n x k) = (4 x 4):
##                PC1        PC2        PC3         PC4
## Murder   0.5358995  0.4181809 -0.3412327 -0.64922780
## Assault  0.5831836  0.1879856 -0.2681484  0.74340748
## UrbanPop 0.2781909 -0.8728062 -0.3780158 -0.13387773
## Rape     0.5434321 -0.1673186  0.8177779 -0.08902432
```

En general, estos datos se hallan con la SVD de la matriz de correlación.

PCR - Regresión con componentes principales con la función prcomp

```
plot(USArrests)
```




```
# USArrests[,-1]
```

Primero prcomp, y a continuación lm con la dos primeras componentes principales:

```
pr.out2 <- prcomp(USArrests[, -1], scale = TRUE)
pr.out2$x[1:10, ]
```

```
##                  PC1         PC2         PC3
## Alabama    -0.2291413  0.82009892  0.39890458
## Alaska     -1.7181516  2.00498484 -1.37574835
## Arizona    -2.0322512  0.02169077  0.37669633
## Arkansas    0.4779678  1.00349690  0.06808964
```

```
## California  -2.9187200 -0.55717592 -0.36745241
## Colorado    -1.8493200 -0.22739880 -0.95209667
## Connecticut  0.7750360 -1.21810095  0.49406280
## Delaware    -0.3178373 -0.10393014  1.03666858
## Florida     -2.3916511  0.27179456  0.62072833
## Georgia     -0.4305578  0.64452877 -0.12919142
```

```
summary(pr.out2)
```

```
## Importance of components:
##                           PC1    PC2    PC3
## Standard deviation     1.3832 0.8800 0.5590
## Proportion of Variance 0.6377 0.2581 0.1042
## Cumulative Proportion  0.6377 0.8958 1.0000
```

```
regPC <- lm(USArrests$Murder ~ pr.out2$x[, -3])
regPC
```

```
##
## Call:
## lm(formula = USArrests$Murder ~ pr.out2$x[, -3])
##
## Coefficients:
##        (Intercept)  pr.out2$x[, -3]PC1  pr.out2$x[, -3]PC2
##              7.788              -2.008               2.356
```

```
summary(regPC)
```

```
##
## Call:
## lm(formula = USArrests$Murder ~ pr.out2$x[, -3])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9609 -1.5374 -0.1192  1.6443  7.2292
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          7.7880     0.3810  20.443  < 2e-16 ***
## pr.out2$x[, -3]PC1  -2.0079     0.2782  -7.217 3.83e-09 ***
## pr.out2$x[, -3]PC2   2.3557     0.4373   5.387 2.25e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.694 on 47 degrees of freedom
## Multiple R-squared:  0.6331, Adjusted R-squared:  0.6175
## F-statistic: 40.55 on 2 and 47 DF,  p-value: 5.845e-11
```

```
7.7880 - 2.008 * pr.out2$x[, 1] + 2.356 * pr.out2$x[, 2] #  que corresponde a fitted(regPC)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##      10.180269      15.961793      11.919864       9.192479      12.336083
##       Colorado    Connecticut       Delaware        Florida        Georgia
##      10.965683       3.361882       8.181358      13.230783      10.171070
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##       2.781466       5.960761       9.152018       6.171954       3.412262
##         Kansas       Kentucky      Louisiana          Maine       Maryland
```

```
##       5.602550       6.189990      10.117392       3.983399      12.492029
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##       4.794654      12.037110       3.833011      11.139101       8.926172
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##       6.132127       5.269228      13.415300       3.186898       5.227158
##     New Mexico       New York North Carolina   North Dakota           Ohio
##      12.631302       9.447693      13.060413       3.362744       5.679178
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##       6.822709       8.824877       4.326488       3.876813      12.382003
##   South Dakota      Tennessee          Texas           Utah        Vermont
##       5.439521       9.805616       8.313233       5.572910       5.074522
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##       7.471462       7.409038       5.116082       2.551356       6.906195
```

```
fitted(regPC)
```

```
##         1         2         3         4         5         6         7         8
## 10.179966 15.960936 11.919653  9.192185 12.335981 10.965575  3.362375  8.181361
##         9        10        11        12        13        14        15        16
## 13.230451 10.170807  2.782111  5.960793  9.152072  6.172102  3.412554  5.602749
##        17        18        19        20        21        22        23        24
##  6.189989 10.117178  3.983563 12.491622  4.795111 12.036846  3.833367 11.138552
##        25        26        27        28        29        30        31        32
##  8.926137  6.132141  5.269414 13.415018  3.187194  5.227624 12.630929  9.447757
##        33        34        35        36        37        38        39        40
## 13.059686  3.362896  5.679474  6.822820  8.824826  4.326856  3.877344 12.381396
##        41        42        43        44        45        46        47        48
##  5.439508  9.805382  8.313343  5.573271  5.074409  7.471465  7.409168  5.116026
##        49        50
##  2.551817  6.906201
```

```
regPCt <- lm(USArrests$Murder ~ pr.out2$x) ## regresión con las 3
regPCt
```

```
##
## Call:
## lm(formula = USArrests$Murder ~ pr.out2$x)
##
## Coefficients:
##  (Intercept)  pr.out2$xPC1  pr.out2$xPC2  pr.out2$xPC3
##        7.788        -2.008         2.356         1.538
```

```
summary(regPCt)
```

```
##
## Call:
## lm(formula = USArrests$Murder ~ pr.out2$x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3990 -1.9127 -0.3444  1.2557  7.4279
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.7880     0.3641  21.392  < 2e-16 ***
## pr.out2$xPC1  -2.0079     0.2659  -7.552 1.36e-09 ***
```

```
## pr.out2$xPC2   2.3557     0.4179   5.637 1.01e-06 ***
## pr.out2$xPC3   1.5380     0.6579   2.338   0.0238 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.574 on 46 degrees of freedom
## Multiple R-squared:  0.6721, Adjusted R-squared:  0.6507
## F-statistic: 31.42 on 3 and 46 DF,  p-value: 3.322e-11
```

Regresión con modelo lineal dependiendo de las variables originales no tendría que ser diferente a regresión con las 3 componentes principales.

```
LR <- lm(USArrests$Murder ~ USArrests$Assault + USArrests$Rape + USArrests$UrbanPop)
summary(LR)
```

```
##
## Call:
## lm(formula = USArrests$Murder ~ USArrests$Assault + USArrests$Rape +
##     USArrests$UrbanPop)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3990 -1.9127 -0.3444  1.2557  7.4279
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          3.276639   1.737997   1.885   0.0657 .
## USArrests$Assault    0.039777   0.005912   6.729 2.33e-08 ***
## USArrests$Rape       0.061399   0.055740   1.102   0.2764
## USArrests$UrbanPop -0.054694   0.027880  -1.962   0.0559 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.574 on 46 degrees of freedom
## Multiple R-squared:  0.6721, Adjusted R-squared:  0.6507
## F-statistic: 31.42 on 3 and 46 DF,  p-value: 3.322e-11
```

Con la librería pls, utilizando pcr, como al principio de la práctica:

```
library(pls)
regPCRt <- pcr(USArrests$Murder ~ ., data = USArrests, scale = TRUE)
summary(regPCRt)
```

```
## Data:    X dimension: 50 3
##  Y dimension: 50 1
## Fit method: svdpc
## Number of components considered: 3
## TRAINING: % variance explained
##                    1 comps  2 comps  3 comps
## X                    63.77    89.58   100.00
## USArrests$Murder     40.66    63.31    67.21
```

```
fitted(regPCRt)
```

```
## , , 1 comps
##
##                USArrests$Murder
```

```
## Alabama               8.248093
## Alaska               11.237876
## Arizona              11.868557
## Arkansas              6.828289
## California           13.648497
## Colorado             11.501249
## Connecticut           6.231805
## Delaware              8.426185
## Florida              12.590196
## Georgia               8.652517
## Hawaii                6.951613
## Idaho                 5.330039
## Illinois             10.433381
## Indiana               6.880696
## Iowa                  4.188512
## Kansas                6.556939
## Kentucky              5.333446
## Louisiana             9.090457
## Maine                 3.709039
## Maryland             10.675359
## Massachusetts         8.036418
## Michigan             11.485590
## Minnesota             5.500308
## Mississippi           7.112812
## Missouri              9.148843
## Montana               5.411624
## Nebraska              5.902147
## Nevada               13.407424
## New Hampshire         3.888511
## New Jersey            8.786607
## New Mexico           11.247831
## New York             10.990935
## North Carolina        8.172848
## North Dakota          2.636751
## Ohio                  7.680888
## Oklahoma              7.487327
## Oregon                8.832543
## Pennsylvania          6.380606
## Rhode Island          7.416341
## South Carolina        8.411713
## South Dakota          4.062260
## Tennessee             8.406569
## Texas                 9.750783
## Utah                  8.210820
## Vermont               2.451384
## Virginia              7.336035
## Washington            8.583467
## West Virginia         3.117226
## Wisconsin             4.653805
## Wyoming               6.506840
##
## , , 2 comps
##
##                USArrests$Murder
```

```
## Alabama              10.179966
## Alaska               15.960936
## Arizona              11.919653
## Arkansas              9.192185
## California           12.335981
## Colorado             10.965575
## Connecticut           3.362375
## Delaware              8.181361
## Florida              13.230451
## Georgia              10.170807
## Hawaii                2.782111
## Idaho                 5.960793
## Illinois              9.152072
## Indiana               6.172102
## Iowa                  3.412554
## Kansas                5.602749
## Kentucky              6.189989
## Louisiana            10.117178
## Maine                 3.983563
## Maryland             12.491622
## Massachusetts         4.795111
## Michigan             12.036846
## Minnesota             3.833367
## Mississippi          11.138552
## Missouri              8.926137
## Montana               6.132141
## Nebraska              5.269414
## Nevada               13.415018
## New Hampshire         3.187194
## New Jersey            5.227624
## New Mexico           12.630929
## New York              9.447757
## North Carolina       13.059686
## North Dakota          3.362896
## Ohio                  5.679474
## Oklahoma              6.822820
## Oregon                8.824826
## Pennsylvania          4.326856
## Rhode Island          3.877344
## South Carolina       12.381396
## South Dakota          5.439508
## Tennessee             9.805382
## Texas                 8.313343
## Utah                  5.573271
## Vermont               5.074409
## Virginia              7.471465
## Washington            7.409168
## West Virginia         5.116026
## Wisconsin             2.551817
## Wyoming               6.906201
##
## , , 3 comps
##
##                USArrests$Murder
```

```
## Alabama              10.793487
## Alaska               13.845014
## Arizona              12.499018
## Arkansas              9.296908
## California           11.770833
## Colorado              9.501235
## Connecticut           4.122251
## Delaware              9.775774
## Florida              14.185141
## Georgia               9.972108
## Hawaii                1.807086
## Idaho                 5.968315
## Illinois             10.115168
## Indiana               5.505761
## Iowa                  3.080437
## Kansas                5.346423
## Kentucky              5.769092
## Louisiana            10.934441
## Maine                 4.267684
## Maryland             13.252220
## Massachusetts         5.555289
## Michigan             11.527607
## Minnesota             3.445667
## Mississippi          12.222335
## Missouri              8.259884
## Montana               5.720538
## Nebraska              4.955995
## Nevada               11.694674
## New Hampshire         3.064389
## New Jersey            5.887785
## New Mexico           12.755499
## New York             10.278912
## North Carolina       15.208861
## North Dakota          3.108308
## Ohio                  5.261824
## Oklahoma              6.791813
## Oregon                7.735738
## Pennsylvania          4.469929
## Rhode Island          5.949135
## South Carolina       13.130661
## South Dakota          5.022175
## Tennessee             9.179467
## Texas                 8.462044
## Utah                  5.080455
## Vermont               4.123421
## Virginia              7.307146
## Washington            6.660358
## West Virginia         4.936553
## Wisconsin             2.438163
## Wyoming               7.356976
```

```
fitted(regPCt)
```

```
##         1         2         3         4         5         6         7         8
## 10.793487 13.845014 12.499018  9.296908 11.770833  9.501235  4.122251  9.775774
```

```
##         9        10        11        12        13        14        15        16
## 14.185141  9.972108  1.807086  5.968315 10.115168  5.505761  3.080437  5.346423
##        17        18        19        20        21        22        23        24
##  5.769092 10.934441  4.267684 13.252220  5.555289 11.527607  3.445667 12.222335
##        25        26        27        28        29        30        31        32
##  8.259884  5.720538  4.955995 11.694674  3.064389  5.887785 12.755499 10.278912
##        33        34        35        36        37        38        39        40
## 15.208861  3.108308  5.261824  6.791813  7.735738  4.469929  5.949135 13.130661
##        41        42        43        44        45        46        47        48
##  5.022175  9.179467  8.462044  5.080455  4.123421  7.307146  6.660358  4.936553
##        49        50
##  2.438163  7.356976
```

Observamos que regPCt y regPCRt lógicamente es similar. En cambio, pcr no te da mucha información.