Proyecto de Aplicación:

Análisis de Componentes Principales

Anthony Javier Sanchez Romero No. Cuenta 20181900484
Andrew Sebastian Castejón Rodriguez No. Cuenta 20191000729
Centro de Innovación en Cómputo Científico
Universidad Nacional Autónoma de Honduras

Curso: *Metódos Lineales* – Profesor: Lic. Devis Alvarado

CONTENTS

LIST OF FIGURES

Abstract

This document presents the method to perform a principal component analysis on a set of variables in order to optimize information.

I. Introducción

[3]*Principal Component Analysis* (**PCA**) es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información. Supóngase que existe una muestra con n individuos cada uno con p variables $(X_1, X_2, \ldots, X_p)$, es decir, el espacio muestral tiene p dimensiones. **PCA** permite encontrar un número de factores subyacentes $(z < p)$ que explican aproximadamente lo mismo que las p variables originales. Donde antes se necesitaban p valores para caracterizar a cada individuo, ahora bastan z valores. Cada una de estas z nuevas variables recibe el nombre de componente principal.

Principal Component Analysis pertenece a la familia de técnicas conocida como *unsupervised learning*. Los métodos de [4]*supervised learning* tienen el objetivo de predecir una variable respuesta Y a partir de una serie de predictores. Para ello, se dispone de p características $(X_1, X_2, \ldots, X_p)$ y de la variable respuesta Y medidas en n observaciones. En el caso de *unsupervised learning*, la variable respuesta Y no se tiene en cuenta ya que el objetivo no es predecir Y sino extraer información empleando los predictores, por ejemplo, para identificar subgrupos. El principal problema al que se enfrentan los métodos de *unsupervised learning* es la dificultad para validar los resultados dado que no se dispone de una variable respuesta que permita contrastarlos.

El método de **PCA** permite por lo tanto "condensar" la información aportada por múltiples variables en solo unas pocas componentes. Esto lo convierte en un método muy útil de aplicar previa utilización de otras técnicas estadísticas tales como regresión, *clustering*... Aun así no hay que olvidar que sigue siendo necesario disponer del valor de las variables originales para calcular las componentes.

II. Cálculo de las componentes principales

Es importante resaltar el hecho de que el concepto de mayor información se relaciona con el de mayor variabilidad o varianza. Cuanto mayor sea la variabilidad de los datos (varianza) se considera que existe mayor información, lo cual está relacionado con el concepto de entropía.

Se considera una serie de variables $X = (x_1, x_2, ..., x_p)$ sobre un grupo de objetos o individuos y se trata de calcular, a partir de ellas, un nuevo conjunto de variables $Y = (y_1, y_2, ..., y_p)$, incorreladas entre sí, cuyas varianzas vayan decreciendo progresivamente. Pasaremos $X \longrightarrow Y$, mediante una combinación lineal, de manera que:

$$y_j = a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jp}x_p = a_j'X \text{ con } (j = 1, \cdots p).$$

Denominaremos al vector de constantes o vector de pesos $a_j = (a_{j1}, \cdots, a_{jp})$ como vector de Loadings o vector de pesos, el proposito de este sera máximizar la varianza de cada componente por el concepto entrópico visto anteriormente, De modo ideal, se buscan m ¡ p variables que sean combinaciones lineales de las p originales y que estén incorreladas, recogiendo la mayor parte de la información o variabilidad de los datos. Si las variables originales están incorreladas de partida, entonces no tiene sentido realizar un análisis de componentes principales.

A. Proceso extracción de loadings

Se elige a_1 de modo que se maximice la varianza de y_1 sujeta a la restricción de que $a_j'a_j = 1$, ademas nuestra función es $var(y_1) = var(a_1'X) = a_1'\Sigma a_1$. El método habitual para maximizar una función de varias variables sujeta a restricciones el método de los multiplicadores de Lagrange; nuestra incognita es a_1, por lo que obtenemos:

$$\begin{gathered}\text{Sea } L : L(a_1) = a_1'\Sigma a_1 - \lambda(a_1'a_1 - 1) \\ \implies \tfrac{\partial L}{\partial a_1} = 2\Sigma a_1 - 2\lambda a_1 = 0 \\ = 2(\Sigma - \lambda I)a_1 = 0\end{gathered}$$

Obtenemos entonces un sistema lineal, pero para que este tenga solución distinta de 0, $(\Sigma - \lambda I)$, Por el teorema de Roché-Frobenius, está matriz debe de ser singular

$$\begin{gathered}\implies |\Sigma - \lambda I| = 0 \\ \text{es decir que } \lambda \text{ es un autovalor de } \Sigma \\ \implies var(y_1) = var(a_1'X) = a_1'\Sigma a_1 = \lambda a_1'a_1 = \lambda(1) = \lambda.\end{gathered}$$

Suponiendo que Σ es definida positiva, disponemos de: $\lambda_1 > \lambda_2 > ... > \lambda_p$. Para maximizar la varianza de y_1 tomamos el mayor, λ_1, con lo que a_1 es t.q, $\Sigma a_1 = \lambda_1 a_1$, a_1 es el autovector asociado a λ_1. Sea $y_2 = a_2'X$, este componente se obtiene mediante un argumento parecido. Además, se quiere que y2 esté incorrelado con el anterior componente y_1.

$$\begin{gathered}\implies cov(y_2, y_1) = a_2'\Sigma a_1 = 0 \\ \text{como } \Sigma a_1 = \lambda_1 a_1 \\ \implies a_2'a_1 = 0 \text{ (ortogonales)}\end{gathered}$$

esto se suma a las otra restricción que teniamos antes:

$$\begin{gathered}\implies L(a_2) = a_2'\Sigma a_2 - \lambda(a_2'a_2 - 1) - \delta(a_2'a_1) \\ \text{con lo que} \\ \tfrac{\partial L}{\partial a_2} = 2\Sigma a_2 - 2\lambda a_2 - \delta a_1 = 0\end{gathered}$$

Multiplicamos $\frac{\partial L}{\partial a_2}$ por a_1':

$$\begin{gathered}a_1'\tfrac{\partial L}{\partial a_2} = 2a_1'\Sigma a_2 - 2\lambda a_1'a_2 - \delta a_1'a_1 = 0 \\ = 2a_1'\Sigma a_2 - \delta = 0 \\ \implies \delta = 2a_1'\Sigma a_2 = 2cov(y_2, y_1) = 0 \\ \implies \tfrac{\partial L}{\partial a_2} = 2\Sigma a_2 - 2\lambda a_2 - \delta a_1 = 2(\Sigma - \lambda I)a_2 = 0\end{gathered}$$

Análogamente al caso anterior, elegimos λ como el segundo mayor autovalor de la matriz Σ con su autovector asociado a_2. Podemos llegar hasta el j-ésimo componente y le correspondera el j-ésimo autovector; sus restricciones nuevas están dadas por:

$$\begin{gathered}a_j'a_j = 1 \\ cov(y_1, y_j) = 0 \\ cov(y_2, y_j) = 0 \\ \vdots \\ cov(y_p, y_j) = 0\end{gathered}$$

Obtendriamos:

$$Y = AX = \begin{pmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{p1} & \cdots & a_{pp} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix}$$

con A matriz ortogonal $\implies A^{-1} = A'$, y ademas:

$$\Delta = var(Y) = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_p \end{pmatrix}$$

B. Proporción de varianza

Asumiendo que las variables se han normalizado para tener media cero, la varianza total presente en el set de datos se define como:

$$\sum_{j=1}^{p} Var(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

y la varianza explicada por la componente m es

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2$$

Por lo tanto, la proporción de varianza explicada por la componente m viene dada por el ratio

$$\frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}$$

Tanto la proporción de varianza explicada como la proporción de varianza explicada acumulada son dos valores de gran utilidad a la hora de decidir el número de componentes principales a utilizar en los análisis posteriores.

C. Número óptimo de PC

Por lo general, dada una matriz de datos de dimensiones $n \times p$, el número de componentes principales que se pueden calcular es como máximo de $n - 1$ o p (el menor de los dos valores es el limitante).
Sin embargo, siendo el objetivo del **PCA** reducir la dimensionalidad, suelen ser de interés utilizar el número mínimo de componentes que resultan suficientes para explicar los datos.

D. Método para la solución del Problema

El método [5] *Principal Components Regression* **PCR** consiste en ajustar un modelo de regresión lineal por mínimos cuadrados empleando como predictores las componentes generadas a partir de un *Principal Component Analysis* (**PCA**). De esta forma, con un número reducido de componentes se puede explicar la mayor parte de la varianza de los datos.

En los estudios observacionales, es frecuente disponer de un número elevado de variables que se pueden emplear como predictores. A pesar de ello, un alto número de predictores no implica necesariamente mucha información. Si las variables están correlacionadas entre ellas, la información que aportan es redundante y además viola la condición de no colinealidad necesaria en la regresión por mínimos cuadrados. Dado que el **PCA** es útil eliminando información redundante, si se emplean como predictores las componentes principales se puede mejorar el modelo de regresión. Es importante tener en cuenta que, si bien el *Principal Components Regression* reduce el número de predictores del modelo, no se puede considerar como un método de selección de variables ya que todas ellas se necesitan para el cálculo de las componentes. La identificación del número óptimo de componentes principales que se emplean como predictores en **PCR** puede identificarse por *cross validation*.

El método **PCR** no deja de ser un ajuste lineal por mínimos cuadrados que emplea componentes principales como predictores, para que sea válido se tienen que cumplir las condiciones requeridas para la regresión por mínimos cuadrados.

1) Descripción de los experimentos:

La cuantificación del contenido en grasa de la carne pude hacerse mediante técnicas de analítica química, sin embargo, este proceso es costoso en tiempo y recursos. Una posible alternativa para reducir costes y optimizar tiempo es emplear un espectrofotómetro (instrumento capaz de detectar la absorbancia que tiene un material a diferentes tipos de luz en función de sus características). Para comprobar su efectividad se mide el espectro de absorbancia de 100 longitudes de onda en 215 muestras de carne, cuyo contenido en grasa se obtiene también por análisis químico para poder comparar los resultados. El set de datos **meatspec** del paquete **faraway** contiene toda la información.

El set de datos contiene 101 columnas. Las 100 primeras, nombradas como $v_1, ..., v_{100}$ recogen el valor de absorbancia para cada una de las 100 longitudes de onda analizadas, y la columna fat el contenido en grasa medido por técnicas químicas.

```
library(faraway)
data(meatspec)
dim(meatspec)
```

```
## [1] 215 101
```

Para poder evaluar la capacidad predictiva del modelo, se dividen las observaciones disponibles en dos grupos: uno de entrenamiento para ajustar el modelo (80% de los datos) y uno de test (20% de los datos).

```
training <- meatspec[1:172, ]
test     <- meatspec[173:215, ]
```

En primer lugar se ajusta un modelo incluyendo todas las longitudes de onda como predictores.

```
modelo <- lm(fat ~ ., data = training)
summary(modelo)
```

```
##
## Call:
## lm(formula = fat ~ ., data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09837 -0.35779  0.04555  0.38080  2.33860
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.324      2.012   3.143 0.002439 **
## V1           12134.077   3659.798   3.316 0.001443 **
```

```
## V2          -12585.857   5971.891  -2.108 0.038605 *
## V3           -5107.556   9390.265  -0.544 0.588200
## V4           23880.493  17143.644   1.393 0.167977
## V5          -40509.555  22129.359  -1.831 0.071360 .
## V6           28469.416  19569.400   1.455 0.150134
## V7          -20901.082  12501.639  -1.672 0.098952 .
## V8            8369.465   7515.467   1.114 0.269193
## V9           -1539.328   5397.505  -0.285 0.776327
## V10           4706.267   7406.895   0.635 0.527217
## V11           7012.943  11720.620   0.598 0.551516
## V12          14891.444  20169.170   0.738 0.462749
## V13         -30963.902  26186.839  -1.182 0.240983
## V14          34338.612  22323.830   1.538 0.128444
## V15         -22235.237  13842.268  -1.606 0.112640
## V16          -7466.797   8558.172  -0.872 0.385890
## V17           6716.653   6561.805   1.024 0.309500
## V18          -2033.071   6741.330  -0.302 0.763851
## V19           8541.212   9419.998   0.907 0.367627
## V20          -1667.207  17300.433  -0.096 0.923500
## V21         -31972.494  24622.615  -1.299 0.198317
## V22          59526.389  27730.712   2.147 0.035244 *
## V23         -49241.388  23117.226  -2.130 0.036632 *
## V24          16184.597  16679.609   0.970 0.335180
## V25          12077.951  10751.912   1.123 0.265081
## V26         -12632.330   6774.573  -1.865 0.066361 .
## V27          -6298.837   7032.334  -0.896 0.373442
## V28          29625.988   9011.227   3.288 0.001573 **
## V29         -39374.835  13561.228  -2.903 0.004914 **
## V30          31251.427  18742.000   1.667 0.099829 .
## V31         -27238.189  21335.756  -1.277 0.205887
## V32          23009.543  19776.156   1.163 0.248522
## V33          -4584.373  14572.471  -0.315 0.753995
## V34          -5437.943  10344.728  -0.526 0.600754
## V35          -6128.931   8762.663  -0.699 0.486564
## V36           5599.605   6652.640   0.842 0.402776
## V37          -5569.160   6670.198  -0.835 0.406557
## V38             97.451   9291.480   0.010 0.991661
## V39          36021.407  12574.711   2.865 0.005488 **
## V40         -54273.400  17144.384  -3.166 0.002280 **
## V41          52084.876  21758.024   2.394 0.019318 *
## V42         -48458.089  23950.549  -2.023 0.046813 *
## V43          29334.488  20232.617   1.450 0.151500
## V44         -18282.834  13508.157  -1.353 0.180200
## V45          22110.934   9725.348   2.274 0.026020 *
## V46         -11735.692   6631.245  -1.770 0.081061 .
## V47           -514.521   3800.612  -0.135 0.892696
## V48           2551.480   6131.893   0.416 0.678592
## V49           3707.639   8970.401   0.413 0.680618
## V50         -25762.703  10934.783  -2.356 0.021236 *
## V51          46844.468  15367.852   3.048 0.003233 **
## V52         -47783.626  18069.344  -2.644 0.010065 *
## V53          26233.604  18822.491   1.394 0.167744
## V54             87.825  17403.836   0.005 0.995988
## V55          -8475.119  13232.005  -0.641 0.523908
## V56           3488.507   7228.428   0.483 0.630858
## V57          -1520.733   4988.093  -0.305 0.761355
## V58           2275.175   5495.630   0.414 0.680124
```

```
## V59          -5415.427   5721.475  -0.947 0.347099
## V60           7152.015   4754.317   1.504 0.136935
## V61          -4494.234   4512.937  -0.996 0.322702
## V62           3662.045   4811.634   0.761 0.449129
## V63          13993.987   7098.106   1.972 0.052563 .
## V64         -23252.133   8973.839  -2.591 0.011604 *
## V65           4373.731  10048.591   0.435 0.664695
## V66           4580.913  10146.146   0.451 0.653011
## V67           -837.676  10747.974  -0.078 0.938097
## V68          -7074.425  10852.430  -0.652 0.516587
## V69           9506.571   9739.256   0.976 0.332325
## V70          -2765.100   9519.031  -0.290 0.772295
## V71          -1125.135   8586.061  -0.131 0.896113
## V72          -7295.096   7489.488  -0.974 0.333341
## V73          17059.811   6522.093   2.616 0.010870 *
## V74          -9889.553   6543.945  -1.511 0.135162
## V75           -325.615   6125.973  -0.053 0.957759
## V76            782.219   5421.002   0.144 0.885677
## V77           8058.935   5793.416   1.391 0.168554
## V78         -15869.978   6448.208  -2.461 0.016282 *
## V79          21768.619   6435.678   3.382 0.001172 **
## V80         -28338.145   8180.874  -3.464 0.000906 ***
## V81           8523.317  10053.153   0.848 0.399384
## V82          22319.451  12098.046   1.845 0.069226 .
## V83         -17244.722  13991.685  -1.232 0.221829
## V84         -18325.836  14959.964  -1.225 0.224627
## V85          33345.457  13868.197   2.404 0.018808 *
## V86          -7955.157  14571.278  -0.546 0.586813
## V87          -7837.966  16141.553  -0.486 0.628762
## V88          -1815.552  17261.928  -0.105 0.916532
## V89            631.595  15684.751   0.040 0.967992
## V90          -2701.955  16187.612  -0.167 0.867911
## V91           4375.678  19400.005   0.226 0.822199
## V92          12925.188  16456.244   0.785 0.434816
## V93          -7441.235  12417.883  -0.599 0.550923
## V94          -2464.532  11815.234  -0.209 0.835366
## V95          -2090.635   9666.576  -0.216 0.829394
## V96          10912.352   9950.716   1.097 0.276505
## V97         -20331.405  11022.234  -1.845 0.069270 .
## V98           3948.443   8227.133   0.480 0.632753
## V99           6358.930   8652.372   0.735 0.464800
## V100          -263.365   4104.463  -0.064 0.949019
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.074 on 71 degrees of freedom
## Multiple R-squared:  0.997,Adjusted R-squared:  0.9928
## F-statistic: 237.5 on 100 and 71 DF,  p-value: < 2.2e-16
```

El valor $R^2_{ajustado}$ obtenido es muy alto (0.9928) lo que indica que el modelo es capaz de predecir con gran exactitud el contenido en grasa de las observaciones con las que se ha entrenado. El hecho de que el modelo en conjunto sea significativo (p-value: $< 2.2e^{-16}$), pero que muy pocos de los predictores lo sean a nivel individual, es un indicativo de una posible redundancia entre los predictores (colinealidad).

¿Cómo de bueno es el modelo prediciendo nuevas observaciones que no han participado en ajuste? Al tratarse de un modelo de regresión, la estimación del error de predicción se obtiene mediante el $Mean\ Square\ Error\ (MSE)$.

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

```
# MSE empleando las observaciones de entrenamiento
training_mse <- mean((modelo$fitted.values - training$fat)^2)
training_mse
```

```
## [1] 0.4765372
```

```
# MSE empleando nuevas observaciones
predicciones <- predict(modelo, newdata = test)
test_mse <- mean((predicciones - test$fat)^2)
test_mse
```

```
## [1] 14.54659
```

Se observa que el modelo tiene un MSE muy bajo (0.48) cuando predice las mismas observaciones con las que se ha entrenado, pero 30 veces más alto (14.54) al predecir nuevas observaciones. Esto significa que el modelo no es útil, ya que el objetivo es aplicarlo para predecir el contenido en grasa de futuras muestras de carne. A este problema se le conoce como $overfitting$. Una de las causas por las que un modelo puede sufrir $overfitting$ es la incorporación de predictores innecesarios, que no aportan información o que la información que aportan es redundante.

Se recurre en primer lugar a la selección de predictores mediante $stepwise$ selection empleando el AIC como criterio de evaluación:

```
modelo_step_selection <- step(object = modelo, trace = FALSE)

# Número de predictores del modelo resultante
length(modelo_step_selection$coefficients)
```

```
## [1] 73
```

```
# Training-MSE
training_mse <- mean((modelo_step_selection$fitted.values - training$fat)^2)
training_mse
```

```
## [1] 0.5034001
```

El proceso de $stepwise$ selection devuelve como mejor modelo el formado por 73 de los 100 predictores disponibles. Al haber eliminado predictores del modelo, el $training\ MSE$ siempre aumenta, en este caso de 0.48 a 0.05, pero el $test - MSE$ se ha reducido a 12.88986.

2) Resultados:

Véase ahora el resultado si se ajusta el modelo empleando las componentes principales:

```
# Cálculo de componentes principales. Se excluye la columna con la variable
# respuesta *fat*
pca <- prcomp(training[, -101], scale. = TRUE)

# Se muestra la proporción de varianza explicada y acumulada de las 9 primeras
# componentes
summary(pca)$importance[, 1:9]
```

```
##                             PC1      PC2       PC3       PC4         PC5
## Standard deviation     9.92492 1.043606 0.5357885 0.3312792 0.07898436
## Proportion of Variance 0.98504 0.010890 0.0028700 0.0011000 0.00006000
## Cumulative Proportion  0.98504 0.995930 0.9988000 0.9999000 0.99996000
##                               PC6         PC7         PC8          PC9
## Standard deviation     0.04974461 0.02700194 0.02059129 0.008603878
## Proportion of Variance 0.00002000 0.00001000 0.00000000 0.000000000
## Cumulative Proportion  0.99999000 0.99999000 1.00000000 1.000000000
```

El estudio de la proporción de varianza explicada muestra que la primera componente recoge la mayor parte de la información (98.5%), decayendo drásticamente la varianza en las sucesivas componentes.

Una vez obtenido el valor de las componentes para cada observación (*principal component scores*), puede ajustarse el modelo lineal empleando dichos valores junto con la variable respuesta que le corresponde a cada observación. Con la función 'pcr()' del paquete 'pls' se evita tener que codificar cada uno de los pasos intermedios.

Acorde a la proporción de varianza acumulada, emplear las 4 primeras componentes podría ser una buena elección, ya que en conjunto explican el 99.99100 % de varianza.

```
library(pls)
```

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

```
modelo_pcr <- pcr(formula = fat ~ ., data = training, scale. = TRUE, ncomp = 4)

# Test-MSE
predicciones <- predict(modelo_pcr, newdata = test, ncomp = 4)
test_mse <- mean((predicciones - test$fat)^2)
test_mse
```

```
## [1] 20.55699
```

El $test - MSE$ obtenido (20.56) para el modelo que emplea como predictores las 4 primeras componentes es mucho mayor que el obtenido con el modelo generado por *stepwise* selection (12.89) e incluso que el obtenido incluyendo todos los predictores (14.54659). Esto significa que, o bien el hecho de emplear componentes principales como predictores no es útil para este caso, o que el número de componentes incluido no es el adecuado.

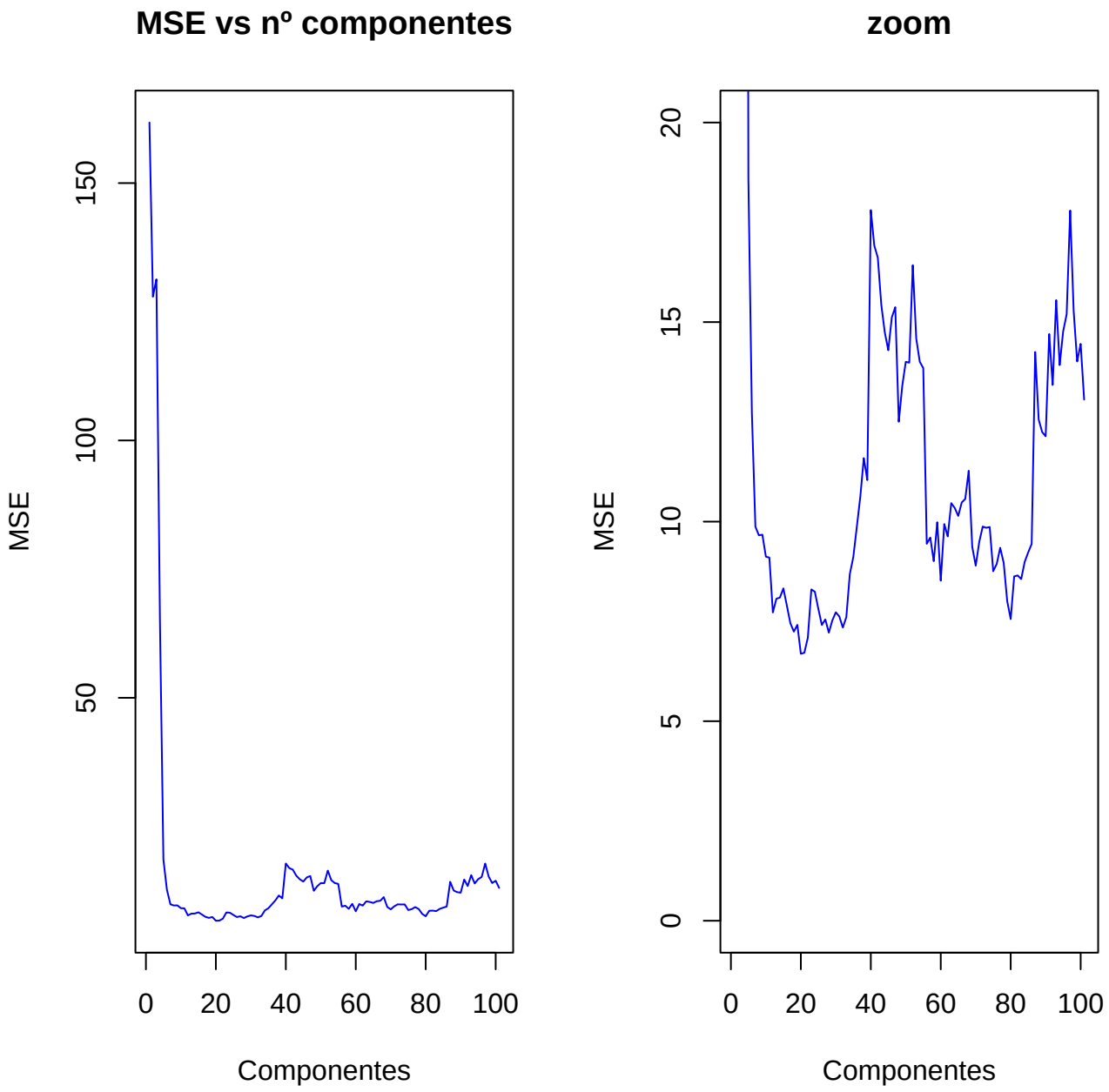
La función 'pcr()' incluye la posibilidad de recurrir a *cross validation* para identificar el número óptimo de componentes con el que se minimiza el MSE.

```
set.seed(123)
modelo_pcr <- pcr(formula = fat ~ ., data = training, scale. = TRUE,
                  validation = "CV")
```

```
modelo_pcr_CV <- MSEP(modelo_pcr, estimate = "CV")
which.min(modelo_pcr_CV$val)
```

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

```
par(mfrow = c(1,2))
plot(modelo_pcr_CV$val, main = "MSE vs nº componentes", type = "l",
     ylab = "MSE",
     col = "blue", xlab = "Componentes")
plot(modelo_pcr_CV$val, main = "zoom", type = "l", ylab = "MSE",
     xlab = "Componentes", col = "blue", ylim = c(0,20))
```



```r
# Test-MSE
predicciones <- predict(modelo_pcr, newdata = test, ncomp = 18)
test_mse <- mean((predicciones - test$fat)^2)
test_mse
```

```
## [1] 4.524698
```

El número óptimo de componentes principales identificado por $cross\ validation$ es de 18. Empleando este número en la PCR se consigue reducir el $test - MSE$ a 4.52, un valor muy por debajo del conseguido con los otros modelos.

REFERENCES

[1] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, *Introduction to Statistical Learning*.
[2] Julian J.Faraway, *Linear Models with R*.
[3] Principal Component Analysis
[4] Supervised
[5] Principal Components Regression
[6] Repositorio