Facultad de Matemáticas

Departamento de Estadística e Investigación Operativa

Máster Universitario en Matemáticas

TRABAJO FIN DE MÁSTER

MODELO PLS

Alumna: Lucía Espejo Alonso

Tutor: Rafael Pino Mejías

20 de junio de 2017

ABSTRACT

The Partial Least Squares approach (PLS) is a multivariate technique which was originated around 1975 by Herman Wold for the modelling of complicated data sets in terms of chains of matrices (blocks), the so-called path model. This included a simple but efficient way to estimate the parameters in these models called NIPALS (non-linear iterative partial least squares).

Around 1980, the simplest PLS model with two blocks (**X**, set of predictor variables, and **Y**, set of response variables) was slightly modified by his son Svante Wold and Hararld Martes to better suit to data from science and technology, and it was proved to be useful to deal with complicated data sets where ordinary regression was difficult or impossible to apply.

Partial Least Squares Regression (PLSR) solves the problem that arises when there are many predictor variables with an extreme dependency relation (multicollinearity problem). For this, PLSR finds a set of new variables which are created as a linear combination of the original variables, taking into account the response variables, so that the multicollinearity problem is eliminated. The Principal Component Regression (PCR) also solves the problem of multicollinearity, but, unlike PLSR, it only takes into account the predictor variables to create the new variables.

Nowaday, PLSR has great utility to model problems associated with market research, economics, chemistry, biology, communication or medicine, among others.

This work is structured as follows: in chapter 2 a review of the principal component regression is made. In Chapter 3 the partial least squares regression is introduced. Chapter 4 describes the NIPALS algorithm as well as the properties that are deduced from it. In addition, the interpretations are shown and the regression coefficients are deduced. In chapter 5 other alternative algorithms are detailed. Finally, Chapter 6 describes the R package which implements both PCR and PLSR, and several examples are shown.

Índice general

CAPÍTULO 1

Introducción

El enfoque por mínimos cuadrados parciales o partial least squares (PLS), por sus siglas en inglés, es una técnica multivariante que fue introducida en 1975 por el estadístico sueco Herman Wold para el modelado de un conjunto de datos complicados en términos de cadenas de matrices (bloques), los llamados modelos de dependencias (path models). Esto incluyó una manera simple pero eficiente de estimar los parámetros en estos modelos llamada algoritmo NIPALS (mínimos cuadrados parciales iterativos no lineales). Esto condujo a su vez al acrónimo PLS para estos modelos.

Alrededor de 1980, el modelo PLS más simple con dos bloques (**X**, conjunto de variables predictoras, e **Y**, conjunto de variables dependientes o respuesta) fue ligeramente modificado por su hijo Svante Wold y por Harald Martens para adaptarse mejor a los datos de la ciencia y la tecnología y demostró ser útil para tratar conjuntos de datos complejos donde la regresión ordinaria era difícil o imposible de aplicar debido al problema de la multicolinealidad.

El problema de la multicolinealidad se presenta cuando los coeficientes de un modelo de regresión son estimados y hay un número relativamente grande de variables predictoras con una relación de extrema dependencia entre ellas. Para resolver el problema de la multicolinealidad existen algunas metodologías como son: regresión por componentes principales (PCR), regresión por mínimos cuadrados parciales (PLSR), método de correlación canónica, métodos de regresión de Ridge y Lasso, entre otras. Las tres primeras son técnicas de reducción de dimensionalidad, es decir, encuentran un conjunto de nuevas variables que se crean como una combinación lineal de las originales de tal manera que el problema de

multicolinealidad se elimine. Sin embargo la regresión de Ridge y Lasso no reducen la dimensionalidad sino que resuelven un problema de optimización para que el nuevo problema sea lo más próximo a la no presencia de multicolinealidad.

La principal diferencia entre la regresión por componentes principales y la regresión por mínimos cuadrados parciales es que la primera solo tiene en cuenta las variables explicativas para construir las nuevas variables (componentes principales), mientras que la segunda además de considerar las variables explicativas, también tiene en cuenta la variable o variables respuesta para construnir las componentes PLS. Ambas técnicas adoptan diferentes enfoques y por lo tanto se obtienen resultados diferentes.

La regresión por mínimos cuadrados parciales también resuelve el problema que surge cuando el número de observaciones es menor que el número de variables predictoras y el efecto que esto tiene sobre la estimación de coeficientes de regresión. Esto da una idea del potencial de este método en situaciones con muestras pequeñas. Esta es una situación común en muchos laboratorios ya que puede llevar mucho tiempo obtener una nueva muestra pero cada muestra puede dar una gran cantidad de información (variables). Una de las grandes ventajas de esta regresión, es que no necesita de datos provenientes de distribuciones normales o conocidas.

En la actualidad, la regresión por mínimos cuadrados parciales tiene gran utilidad para modelar problemas asociados a la investigación de mercados, a la economía, quiometría, biología, comunicación, medicina, análisis de imagen, análisis sensorial, diseño de experimientos, entre otros.

Este trabajo está estructurado de la siguiente manera: en el capítulo 2 se hace un repaso de la regresión por componentes principales. En el capítulo 3 se introduce la regresión por mínimos cuadrados parciales. En el capítulo 4 se describe el algortimo clásico de la regresión por mínimos cuadrados parciales así como las propiedades que se deducen del mismo. Además se muestran las interpretaciones y se deducen los coeficientes de regresión. En el capítulo 5 se detallan otros algoritmos alternativos al algoritmo clásico. Finalmente, en el capítulo 6 se describe el paquete *pls* de R que implementa tanto PCR como PLSR y se muestran varios ejemplos.

CAPÍTULO 2

Regresión por Componentes Principales (PCR)

Sean $X_1, \ldots, X_M$ variables predictoras e $Y_1, \ldots, Y_K$ variables dependientes o respuesta de N observaciones. Estos datos forman dos matrices

$$\mathbf{X}_{N\times M} = \begin{pmatrix} x_{11} & \cdots & x_{1M} \\ \vdots & & \vdots \\ x_{N1} & \cdots & x_{NM} \end{pmatrix} \text{ matriz de variables predictoras}$$

$$\mathbf{Y}_{N\times K} = \begin{pmatrix} y_{11} & \cdots & y_{1K} \\ \vdots & & \vdots \\ y_{N1} & \cdots & y_{NK} \end{pmatrix} \text{ matriz de variables respuesta}$$

El análisis de componentes principales y el modelo PLS son métodos de proyección por lo que los resultados dependen de las escalas de los datos. Cuando las escalas de medida de las variables son muy distintas, las variables con valores más grandes tendrán más peso en el análisis y por tanto más importancia. Para evitar ese problema se estandarizan las variables para que las magnitudes de dichas variables sean parecidas y por lo tanto les correspondan unos pesos e importancias similares:

$$\mathbf{X}_{N\times M} = \begin{pmatrix} \dfrac{x_{11}-\overline{x}_1}{\widehat{\sigma}_{x_1}} & \cdots & \dfrac{x_{1M}-\overline{x}_M}{\widehat{\sigma}_{x_M}} \\ \vdots & & \vdots \\ \dfrac{x_{N1}-\overline{x}_1}{\widehat{\sigma}_{x_1}} & \cdots & \dfrac{x_{NM}-\overline{x}_M}{\widehat{\sigma}_{x_M}} \end{pmatrix}$$

$$\mathbf{Y}_{N\times K} = \begin{pmatrix} \frac{y_{11}-\overline{y}_1}{\widehat{\sigma}_{y_1}} & \cdots & \frac{y_{1K}-\overline{y}_K}{\widehat{\sigma}_{y_K}} \\ \vdots & & \vdots \\ \frac{y_{N1}-\overline{y}_1}{\widehat{\sigma}_{y_1}} & \cdots & \frac{y_{NK}-\overline{y}_K}{\widehat{\sigma}_{y_K}} \end{pmatrix}$$

El objetivo de la regresión lineal múltiple es determinar la relación

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E}. \tag{2.1}$$

Para estimar **B** por el método de mínimos cuadrados, se minimiza el error residual $\|\mathbf{E}\|^2$ definido por la relación

$$\mathbf{E} = \mathbf{Y} - \mathbf{XB}.$$

Esto conduce a las ecuaciones normales

$$\mathbf{X}'\mathbf{XB} = \mathbf{X}'\mathbf{Y}$$

y por tanto, la solución de mínimos cuadrados es

$$\mathbf{B} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{XY}.$$

Esta ecuación da una idea del problema más frecuente en regresión lineal múltiple: la inversa de $\mathbf{X}'\mathbf{X}$ puede no existir. Esto puede ser debido a:

- Por un lado, la presencia de multicolinealidad ya que $|\mathbf{X}'\mathbf{X}| = 0$ si hay colinealidad exacta o $|\mathbf{X}'\mathbf{X}| \simeq 0$ si hay colinealidad aproximada.
- Por otro lado, **X** no sea de rango máximo, es decir $N < M$ (número de observaciones menor que el número de variables predictoras).

Como solución al problema que presenta la regresión lineal múltiple debido a la multicolinealidad surge la regresión por componentes principales y la regresión por mínimos cuadrados parciales.

La idea básica detrás de la regresión por componentes principales es realizar un análisis de componentes principales, es decir, extraer un número de componentes principales inferior al de variables predictoras de manera que se elimine la multicolinealidad y usar dichas componentes para calcular los coeficientes de regresión.

2.1. Análisis de Componentes Principales (PCA)

El objetivo de PCA es determinar $k < M$ componentes principales (*nuevas variables*) que no presenten multicolinealidad, que sean combinaciones lineales de las variables predictoras originales y que recojan la mayor parte de la información o variabilidad de los datos.

Sean las componentes principales (también denominadas puntuaciones de **X**) las columnas de la siguiente matriz

$$\mathbf{T} = \mathbf{XA} \tag{2.2}$$

donde **A** es una matriz de constantes, denominada matriz de cargas de **X**.

1ª Componente Principal

La primera componente principal, $\mathbf{t}_1$, se calcula de modo que recoja la mayor varianza posible sujeto a que $\mathbf{a}_1$ sea ortogonal:

$$\begin{aligned} \text{máx} \quad & Var(\mathbf{t}_1) = Var(\mathbf{X}\mathbf{a}_1) = \mathbf{a}_1'\Sigma\mathbf{a}_1 \\ \text{sujeto a:} \quad & \mathbf{a}_1'\mathbf{a}_1 = 1 \end{aligned}$$

donde $\Sigma = \mathbf{X}'\mathbf{X}$ (por estar los datos estandarizados) es la matriz de covarianza. Sean $\lambda_1, \lambda_2, \ldots, \lambda_M$ los autovalores de Σ. Se cumple que Σ es simétrica y semidefinida positiva por lo que los autovalores son reales y positivos

$$\lambda_1 \geq \lambda_2 \geq \cdots \lambda_M \geq 0.$$

Además por ser simétrica, los autovectores son ortogonales:

$$\mathbf{a}_i'\mathbf{a}_j = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Para resolver el problema anterior de maximización cuya función objetivo es una función de varias variables sujeto a restricciones, se utiliza el método de los multiplicadores de Lagrange. En este caso, la función Lagrangiana es:

$$L(\mathbf{a}_1) = \mathbf{a}_1'\Sigma\mathbf{a}_1 - \lambda(\mathbf{a}_1'\mathbf{a}_1 - 1).$$

El máximo de esta función se encuentra derivando con respecto a $\mathbf{a}_1$ e igualando a cero:

$$\frac{\partial L}{\partial \mathbf{a}_1} = 2\Sigma\mathbf{a}_1 - 2\lambda I\mathbf{a}_1 = 0 \Leftrightarrow (\Sigma - \lambda I)\mathbf{a}_1 = 0.$$

Esto último, es decir $(\Sigma - \lambda I)a_1 = 0$, es un sitema de ecuaciones lineales, que por el teorema de Rouché-Frobenius, tiene solución no nula siempre que $\Sigma - \lambda I$ sea no invertible, es decir, $|\Sigma - \lambda I| = 0$. Luego λ es un autovalor de Σ.

A partir de $(\Sigma - \lambda I)\mathbf{a}_1 = 0$ se tiene que $\Sigma\mathbf{a}_1 = \lambda I\mathbf{a}_1$, por lo tanto $Var(\mathbf{t}_1) = \mathbf{a}_1'\lambda I\mathbf{a}_1 = \lambda$. Luego para maximizar la varianza de la componente $\mathbf{t}_1$ se tiene que tomar el mayor autovalor $\lambda = \lambda_1$ y el autovector $\mathbf{a}_1$ asociado a λ_1.

2ª Componente Principal

La segunda componente principal es solución de un problema de maximización similar al anterior. En este caso las componentes $\mathbf{t}_1$ y $\mathbf{t}_2$ tienen que ser incorreladas:

$$\begin{aligned} \text{máx} \quad & Var(\mathbf{t}_2) = Var(\mathbf{X}\mathbf{a}_2) = \mathbf{a}_2'\Sigma\mathbf{a}_2 \\ \text{sujeto a:} \quad & \mathbf{a}_1'\mathbf{a}_1 = 1 \\ & \mathbf{a}_2'\mathbf{a}_2 = 1 \\ & Cov(\mathbf{t}_1, \mathbf{t}_2) = 0 \end{aligned}$$

Desarrollando la última restricción:

$$Cov(\mathbf{t}_1, \mathbf{t}_2) = Cov(\mathbf{X}\mathbf{a}_1, \mathbf{X}\mathbf{a}_2) = \mathbf{a}_2'E[(\mathbf{X} - \mu)(\mathbf{X} - \mu)']\mathbf{a}_1 = \mathbf{a}_2'\Sigma\mathbf{a}_1 = 0$$

pero $\Sigma\mathbf{a}_1 = \lambda\mathbf{a}_1$, por tanto $Cov(\mathbf{t}_1, \mathbf{t}_2) = \lambda\mathbf{a}_2'\mathbf{a}_1 = 0$, es decir, $\mathbf{a}_1$ y $\mathbf{a}_2$ son ortogonales.

Razonando igual que para una componente, se tiene que $\mathbf{a}_2$ es el autovector asociado al segundo autovalor mayor de Σ, λ_2.

i-ésima Componente principal

La i-ésima componente principal es solución del problema:

$$\begin{aligned} \text{máx} \quad & Var(\mathbf{t}_i) = Var(\mathbf{X}\mathbf{a}_i) = \mathbf{a}_i'\Sigma\mathbf{a}_i \\ \text{sujeto a:} \quad & \mathbf{a}_1'\mathbf{a}_1 = 1 \\ & \vdots \\ & \mathbf{a}_i'\mathbf{a}_i = 1 \\ & Cov(\mathbf{t}_1, \mathbf{t}_2) = \mathbf{a}_2'\mathbf{a}_1 = 0 \\ & \vdots \\ & Cov(\mathbf{t}_{i-1}, \mathbf{t}_i) = \mathbf{a}_i'\mathbf{a}_{i-1} = 0 \end{aligned}$$

Luego $\mathbf{a}_i$ es el autovector asociado a i-ésimo autovalor mayor de Σ.

2.1.1. Número de Componentes Principales

La matriz de covarianza de **T**, Λ, es diagonal:

$$\Lambda = Var(\mathbf{T}) = \mathbf{A}'Var(\mathbf{X})\mathbf{A} = \mathbf{A}'\Sigma\mathbf{A} = diag\{\lambda_1, \lambda_2, \ldots, \lambda_M\}$$

o equivalentemente

$$\Sigma = \mathbf{A}\Lambda\mathbf{A}'$$

ya que **A** es ortogonal, es decir, $\mathbf{A}'\mathbf{A} = \mathbf{A}\mathbf{A}' = \mathbf{I}$.

Teorema 2.1.1.

$$\sum_{i=1}^{M} Var(X_i) = \sum_{i=1}^{M} Var(\boldsymbol{t}_i)$$

Demostración. Por un lado como $Var(\mathbf{t}_i) = \lambda_i$, siendo λ_i el i-ésimo autovalor asociado a la matriz Σ, se tiene que $\sum_{i=1}^{M} Var(\mathbf{t}_i) = \sum_{i=1}^{M} \lambda_i = tr(\Lambda)$. Además teniendo en cuenta las propiedades del operador traza y que **A** es ortogonal, se tiene:

$$tr(\Lambda) = tr(\mathbf{A}'\Sigma\mathbf{A}) = tr(\mathbf{A}'\mathbf{A}\Sigma) = tr(\Sigma)$$

Entonces:

$$\sum_{i=1}^{M} Var(\mathbf{t}_i) = tr(\Lambda) = tr(\Sigma) = \sum_{i=1}^{M} Var(X_i).$$

□

Como consecuencia de este teorema, la variabilidad explicada por la i-ésima componente principal es:

$$\frac{\lambda_i}{\sum_{s=1}^{M} \lambda_s} = \frac{\lambda_i}{\sum_{s=1}^{M} Var(X_s)}.$$

Además la proporción de la variabilidad total explicada por las k primeras componentes principales es:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{s=1}^{M} \lambda_s}$$

y la no explicada

$$\frac{\sum_{i=k+1}^{M} \lambda_i}{\sum_{s=1}^{M} \lambda_s}.$$

Existen varios criterios para determinar el número de componentes principales a seleccionar:

- Se pueden elegir las k primeras componentes principales si

$$\frac{\lambda_{k+1}}{\sum_{s=1}^{M}\lambda_s} < \delta_1 \quad \text{o} \quad \frac{\sum_{i=k+1}^{M}\lambda_i}{\sum_{s=1}^{M}\lambda_s} < \delta_2$$

  donde δ_1 y δ_2 son elegidos convenientemente.

- Retener componentes suficientes hasta que expliquen un determinado porcentaje de la variabilidad total (normalmente un $80\,\%$).

- Retener todas las componentes cuyos autovalores sean mayores que la media $\frac{\sum_{i=1}^{M}\lambda_i}{M}$.

- Hacer un gráfico de i frente a λ_i, y ver si se observa una división entre *autovalores grandes* y *pequeños*.

- Hacer un test de hipótesis (requiere normalidad multivariante).

2.1.2. Interpretación geométrica

Las componentes principales representan geométricamente la selección de un nuevo sistema de coordenadas obtenido por rotación del sistema original ($\mathbf{T} = \mathbf{XA}$), donde los nuevos ejes representan las direcciones con variabilidad máxima (ortogonales entre sí), proporcionando así una descripción más detallada de la estructura de covarianzas.

2.2. Coeficientes de regresión

Las variables predictoras originales $\mathbf{X}$ se sustituyen por las componentes principales $\mathbf{T}$ en (2.1)obteniendo:

$$\mathbf{Y} = \mathbf{TB}^{*'} + \mathbf{E}^*$$

donde $\mathbf{B}^*$ es la solución por el método de mínimos cuadrados ordinario que minimiza $\|\mathbf{E}^*\|^2$

$$\mathbf{B}^* = (\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{Y}.$$

Teniendo en cuenta que

$$\mathbf{T}'\mathbf{T} = \mathbf{A}'\mathbf{X}'\mathbf{XA} = \mathbf{A}'\Sigma\mathbf{A} = \mathbf{A}'\mathbf{A}\Lambda\mathbf{A}'\mathbf{A} = \lambda,$$

entonces

$$\mathbf{B}^* = \Lambda^{-1}\mathbf{T}'\mathbf{Y}.$$

Los coeficientes de regresión $\mathbf{B}^*$ y $\mathbf{B}$ están relacionados de la siguiente manera:

$$\mathbf{B}^* = \mathbf{B}'\mathbf{A}$$

o equivalentemente

$$\mathbf{B} = \mathbf{A}\mathbf{B}^{*'}.$$

CAPÍTULO 3

Regresión por Mínimos Cuadrados Parciales (PLSR)

El objetivo de PLSR también es determinar $k < M$ componentes PLS (*nuevas variables*) que no presenten multicolinealidad y que sean combinaciones lineales de las variables predictoras. Además en este caso, las componentes PLS recogen la mayor variación entre las variables predictoras y respuestas.

Para determinar las componentes PLS, las matrices de datos **X** e **Y** son modeladas mediante nuevas variables en base a modelos de regresión

$$\begin{aligned} \mathbf{X} &= \mathbf{T}\mathbf{P}' + \mathbf{E}_X \\ \mathbf{Y} &= \mathbf{U}\mathbf{Q}' + \mathbf{E}_Y \end{aligned}$$

Sea $k \leq \text{mín}\{N, M, K\}$ el número de componentes PLS a determinar.

- Las matrices $\mathbf{T}_{N\times k}$ y $\mathbf{U}_{N\times k}$ son, respectivamente, puntuaciones de **X** (X scores o también llamadas componentes PLS) y puntuaciones de **Y** (Y scores). Además **T** y **U** son combinaciones lineales, respectivamente, de **X** e **Y** y contienen información sobre las observaciones y sus similitudes o disimilitudes con respecto al problema y modelo dado.
- Las matrices $\mathbf{P}_{M\times k}$ y $\mathbf{Q}_{K\times k}$ son, respectivamente, las matrices de cargas de **X** (X loading) e **Y** (Y loading).
- $\mathbf{E}_X$ y $\mathbf{E}_Y$ son, respectivamente, las matrices residuales de **X** e **Y**.

Las matrices de puntuaciones **T** y **U** están relacionadas mediante la siguiente relación interna:

$$\mathbf{U} = \mathbf{T}\mathbf{D} + \mathbf{H}$$

donde $\mathbf{D}_{k\times k}$ es una matriz diagonal en cuya diagonal principal están los elementos $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k$ y $\mathbf{H}_{N\times k}$ es la matriz residual. Por tanto

$$\mathbf{Y} = \mathbf{TDQ}' + \underbrace{\mathbf{HQ}' + \mathbf{E}_Y}_{\mathbf{E}_Y^*} = \mathbf{TC}' + \mathbf{E}_Y^*. \tag{3.1}$$

donde $\mathbf{C}_{K\times k}$ es la matriz de pesos de $\mathbf{Y}$.

Sea $\mathbf{W}_{M\times k}$ la matriz de pesos de $\mathbf{X}$ que cumple que es ortogonal $\mathbf{W}'\mathbf{W} = 1$. Los pesos $\mathbf{w}$ y $\mathbf{c}$ dan información sobre cómo las variables se combinan para formar la relación cuantitativa entre $\mathbf{X}$ e $\mathbf{Y}$, proporcionando así una interpretación de las proyecciones $\mathbf{t}$ y $\mathbf{u}$. Por lo tanto, estos pesos son esenciales para la comprensión de qué variables de $\mathbf{X}$ son importantes (valores de $\mathbf{w}$ numéricamente grandes) y qué variables de $\mathbf{X}$ proporcionan la misma información (valores de $\mathbf{w}$ similares).

CAPÍTULO 4

Algoritmo clásico de PLSR: NIPALS

Existen diversos algoritmos para determinar las matrices **T**, **P**, **U**, **Q**, **W** y **C** del modelo PLSR. El primer algoritmo que se utilizó fue el NIPALS, desarrollado por Wold.

Algoritmo 1 (NIPALS). *1. Sea* $\boldsymbol{u}$ *la primera columna de* $\boldsymbol{Y}$

2. $\boldsymbol{w} = \dfrac{\boldsymbol{X}'\boldsymbol{u}}{\boldsymbol{u}'\boldsymbol{u}}$ *(pesos de* $\boldsymbol{X}$*)*
3. $\boldsymbol{w} = \dfrac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ *(pesos de* $\boldsymbol{X}$ *normalizados)*
4. $\boldsymbol{t} = \boldsymbol{X}\boldsymbol{w}$ *(puntuaciones de* $\boldsymbol{X}$*)*
5. $\boldsymbol{c} = \dfrac{\boldsymbol{Y}'\boldsymbol{t}}{\boldsymbol{t}'\boldsymbol{t}}$ *(pesos de* $\boldsymbol{Y}$*)*
6. $\boldsymbol{c} = \dfrac{\boldsymbol{c}}{\|\boldsymbol{c}\|}$ *(pesos normalizados de* $\boldsymbol{Y}$*)*
7. *Actualizar* $\boldsymbol{u}$*:* $\boldsymbol{u}^* = \frac{\boldsymbol{Y}'\boldsymbol{c}}{\boldsymbol{c}'\boldsymbol{c}}$ *(puntuaciones de* $\boldsymbol{Y}$*)*
8. $\boldsymbol{u}_\triangle = \boldsymbol{u}^* - \boldsymbol{u}$
9. $\triangle\boldsymbol{u} = \boldsymbol{u}'_\triangle\boldsymbol{u}_\triangle$
10. *Si* $\triangle\boldsymbol{u} < \varepsilon$ *continuar con (11). En caso contrario, hacer* $\boldsymbol{u} = \boldsymbol{u}^*$ *y volver a (2).*

11. $\boldsymbol{p} = \dfrac{\boldsymbol{X}'\boldsymbol{t}}{\boldsymbol{t}'\boldsymbol{t}}$ *(cargas de* $\boldsymbol{X}$*)*

12. $\boldsymbol{q} = \dfrac{\boldsymbol{Y}'\boldsymbol{u}}{\boldsymbol{u}'\boldsymbol{u}}$ *(cargas de* $\boldsymbol{Y}$*)*

13. $d = \dfrac{\boldsymbol{u}'\boldsymbol{t}}{\boldsymbol{t}'\boldsymbol{t}}$ *(factor de escala)*

14. *Eliminar el efecto de* $\boldsymbol{t}$ *sobre* $\boldsymbol{X}$ *e* $\boldsymbol{Y}$ *obteniendo las siguientes matrices residuales*

$$\begin{aligned} \boldsymbol{X} &\rightarrow \boldsymbol{X} - \boldsymbol{t}\boldsymbol{p}' \\ \boldsymbol{Y} &\rightarrow \boldsymbol{Y} - d\boldsymbol{t}\boldsymbol{c}' \end{aligned}$$

Para las siguientes componentes las matrices $\mathbf{X}$ e $\mathbf{Y}$ son sustituidas por las matrices residuales de la iteración anterior. Las iteraciones pueden continuar hasta que se cumpla un criterio de parada o $\mathbf{X}$ se convierta en la matriz nula.

El factor de escala d está relacionado con $\mathbf{c}$. Si $\mathbf{c}$ ha sido normalizado (Paso 6)

$$\mathbf{u}'\mathbf{t} \overset{(7)}{=} \frac{\mathbf{c}'\mathbf{Y}'\mathbf{t}}{\mathbf{c}'\mathbf{c}} = \frac{\mathbf{c}'(\mathbf{Y}'\mathbf{t})}{\mathbf{c}'\mathbf{c}} \overset{\mathbf{c}=\frac{\mathbf{Y}'\mathbf{t}}{\|\mathbf{c}\|\mathbf{t}'\mathbf{t}}}{=} \frac{\|\mathbf{c}\|\mathbf{c}'\mathbf{c}\mathbf{t}'\mathbf{t}}{\mathbf{c}'\mathbf{c}} = \|\mathbf{c}\|\mathbf{t}'\mathbf{t} \Rightarrow d = \frac{\mathbf{u}'\mathbf{t}}{\mathbf{t}'\mathbf{t}} = \|\mathbf{c}\|.$$

Si $\mathbf{c}$ no ha sido normalizado:

$$\mathbf{u}'\mathbf{t} \overset{(7)}{=} \frac{\mathbf{c}'\mathbf{Y}'\mathbf{t}}{\mathbf{c}'\mathbf{c}} = \frac{\mathbf{c}'(\mathbf{Y}'\mathbf{t})}{\mathbf{c}'\mathbf{c}} \overset{\mathbf{c}=\frac{\mathbf{Y}'\mathbf{t}}{\mathbf{t}'\mathbf{t}}}{=} \frac{\mathbf{c}'\mathbf{c}\mathbf{t}'\mathbf{t}}{\mathbf{c}'\mathbf{c}} = \mathbf{t}'\mathbf{t} \Rightarrow d = \frac{\mathbf{u}'\mathbf{t}}{\mathbf{t}'\mathbf{t}} = 1.$$

Conclusión:

$$d = \begin{cases} \|\mathbf{c}\| & \text{si } \mathbf{c} \text{ ha sido normalizado en el algoritmo} \\ 1 & \text{si } \mathbf{c} \text{ no ha sido normalizado en el algoritmo} \end{cases}$$

Las propiedades numéricas de este algoritmo son difíciles de deducir a simple vista pero relacionando los vectores del paso n con los del paso $n-1$ para una dimensión dada, la situación se vuelve más fácil:

$$\begin{aligned} \mathbf{u}_n &= \frac{\mathbf{Y}\mathbf{c}_n}{\mathbf{c}_n'\mathbf{c}_n} = \frac{\mathbf{Y}\mathbf{Y}'\mathbf{t}_n}{(\mathbf{c}_n'\mathbf{c}_n)(\mathbf{t}_n'\mathbf{t}_n)} = \frac{\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{w}_n}{(\mathbf{c}_n'\mathbf{c}_n)(\mathbf{t}_n'\mathbf{t}_n)(\mathbf{w}_n'\mathbf{w}_n)} \\ &= \frac{\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{u}_{n-1}}{(\mathbf{c}_n'\mathbf{c}_n)(\mathbf{t}_n'\mathbf{t}_n)(\mathbf{w}_n'\mathbf{w}_n)(\mathbf{u}_{n-1}'\mathbf{u}_{n-1})}. \end{aligned}$$

Análogamente para los vectores $\mathbf{c}_n$, $\mathbf{t}_n$ y $\mathbf{w}_n$:

$$\begin{aligned}
\mathbf{c}_n &= \frac{\mathbf{Y}'\mathbf{X}\mathbf{X}\mathbf{Y}\mathbf{c}_{n-1}}{(\mathbf{t}'_n\mathbf{t}_n)(\mathbf{w}'_n\mathbf{w}_n)(\mathbf{u}'_{n-1}\mathbf{u}_{n-1})(\mathbf{c}'_{n-1}\mathbf{c}_{n-1})} \\
\mathbf{t}_n &= \frac{\mathbf{X}\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{t}_{n-1}}{(\mathbf{w}'_n\mathbf{w}_n)(\mathbf{u}'_{n-1}\mathbf{u}_{n-1})(\mathbf{c}'_{n-1}\mathbf{c}_{n-1})(\mathbf{t}'_{n-1}\mathbf{t}_{n-1})} \\
\mathbf{w}_n &= \frac{\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{w}_{n-1}}{(\mathbf{u}'_{n-1}\mathbf{u}_{n-1})(\mathbf{c}'_{n-1}\mathbf{c}_{n-1})(\mathbf{t}'_{n-1}\mathbf{t}_{n-1})(\mathbf{w}'_{n-1}\mathbf{w}_{n-1})}
\end{aligned}$$

Estas ecuaciones muestran que el algoritmo funciona de manera similar al método de la potencia para determinar el mayor autovalor de una matriz. El algoritmo, al igual que el método de la potencia, converge rápidamente en casi todos los casos prácticos. La única complicación que puede surgir es cuando hay autovalores dobles que son los más grandes. Pero esta situación no contiene ningún problema de convergencia, sólo que la componente seleccionada no es única. La experiencia indica que en la mayoría de los casos prácticos la convergencia se alcanza en menos de diez iteraciones.

En la convergencia podemos escribir

$$\begin{aligned}
\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{u} &= a\mathbf{u} \\
\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y}\mathbf{c} &= a\mathbf{c} \\
\mathbf{X}\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{t} &= a\mathbf{t} \\
\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{w} &= a\mathbf{w}
\end{aligned}$$

El método de la potencia muestra que a es el mayor autovalor del problema de autovalores. Los vectores **u**, **c**, **t** y **w** son, por lo tanto, los autovectores de las matrices apropiadas correspondientes al mayor autovalor.

El algoritmo de PLSR puede ser visto así: para cada conjunto de matrices residuales **X** e **Y** se calcula el mayor autovalor y los autovectores asociados de las matrices

$$\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X} \quad \mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y} \quad \mathbf{X}\mathbf{X}'\mathbf{Y}\mathbf{Y}' \quad \mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{X}'$$

y los autovectores se utilizan para calcular nuevas matrices residuales. En la práctica solo se necesitan los autovectores de **X**'**YY**'**X**. Los otros se pueden calcular como en el algoritmo.

4.1. Propiedades

En esta sección se describen las propiedades de los vectores y matrices implicadas en el algoritmo NIPALS. Dichas propiedades se derivan de la forma en que la matriz residual $\mathbf{X}_i$ se calcula a partir de las matrices residuales anteriores:

$$\begin{aligned}\mathbf{X}_i &= \mathbf{X}_{i-1} - \mathbf{t}_{i-1}\mathbf{p}_{i-1}^{'} \overset{\mathbf{p}=\frac{\mathbf{X}'\mathbf{t}}{\mathbf{t}'\mathbf{t}}}{=} \mathbf{X}_{i-1} - \frac{\mathbf{t}_{i-1}\mathbf{t}_{i-1}^{'}\mathbf{X}_{i-1}}{\mathbf{t}_{i-1}^{'}\mathbf{t}_{i-1}} = \left[\mathbf{I} - \frac{\mathbf{t}_{i-1}\mathbf{t}_{i-1}^{'}}{\mathbf{t}_{i-1}^{'}\mathbf{t}_{i-1}}\right]\mathbf{X}_{i-1} \\ &= \left[\mathbf{I} - \frac{\mathbf{t}_{i-1}\mathbf{t}_{i-1}^{'}}{\mathbf{t}_{i-1}^{'}\mathbf{t}_{i-1}}\right]\left[\mathbf{X}_{i-2} - \frac{\mathbf{t}_{i-2}\mathbf{t}_{i-2}^{'}\mathbf{X}_{i-2}}{\mathbf{t}_{i-2}^{'}\mathbf{t}_{i-2}}\right].\end{aligned}$$

Propiedad 4.1.1. *Los vectores* $\boldsymbol{w}$ *son ortogonales entre sí*

$$(\boldsymbol{w}_i, \boldsymbol{w}_j) = \boldsymbol{w}_i^{'}\boldsymbol{w}_j = 0 \quad \textit{para } i \neq j.$$

Demostración. Suponemos $i < j$. Escribimos como antes

$$\mathbf{X}_j = \mathbf{Z}\left[\mathbf{X}_i - \frac{\mathbf{t}_i\mathbf{t}_i^{'}\mathbf{X}_i}{\mathbf{t}_i^{'}\mathbf{t}_i}\right]$$

donde $\mathbf{Z}$ es una cierta matriz. Demostraremos $\mathbf{X}_j\mathbf{w}_i = 0$ para $i < j$:

$$\mathbf{X}_j\mathbf{w}_i = \mathbf{Z}\left[\mathbf{X}_i - \frac{\mathbf{t}_i\mathbf{t}_i^{'}\mathbf{X}_i}{\mathbf{t}_i^{'}\mathbf{t}_i}\right]\mathbf{w}_i \overset{\mathbf{t}=\mathbf{X}\mathbf{w}}{=} \mathbf{Z}\left[\mathbf{t}_i - \frac{\mathbf{t}_i\mathbf{t}_i^{'}\mathbf{t}_i}{\mathbf{t}_i^{'}\mathbf{t}_i}\right] = \mathbf{Z}(\mathbf{t}_i - \mathbf{t}_i) = 0.$$

Por tanto

$$\mathbf{w}_j^{'}\mathbf{w}_i = \frac{\mathbf{w}_j^{'}\mathbf{X}_j^{'}\mathbf{Y}_j\mathbf{Y}_j^{'}\overbrace{\mathbf{X}_j\mathbf{w}_i}^{=0}}{a_j} = 0.$$

□

Propiedad 4.1.2. *Los vectores* $\boldsymbol{t}_i$ *son ortogonales entre sí*

$$(\boldsymbol{t}_i, \boldsymbol{t}_j) = \boldsymbol{t}_i^{'}\boldsymbol{t}_j = 0 \quad \textit{para } i \neq j.$$

Demostración. Escribimos como antes

$$\begin{aligned}\mathbf{X}_j &= \mathbf{X}_{j-1} - \frac{\mathbf{t}_{j-1}\mathbf{t}_{j-1}^{'}\mathbf{X}_{j-1}}{\mathbf{t}_{j-1}^{'}\mathbf{t}_{j-1}} \overset{\mathbf{t}=\mathbf{X}\mathbf{w}}{=} \mathbf{X}_{j-1} - \frac{\mathbf{X}_{j-1}\mathbf{w}_{j-1}\mathbf{t}_{j-1}^{'}\mathbf{X}_{j-1}}{\mathbf{t}_{j-1}^{'}\mathbf{t}_{j-1}} \\ &= \mathbf{X}_{j-1}\left[\mathbf{I} - \frac{\mathbf{w}_{j-1}\mathbf{t}_{j-1}^{'}\mathbf{X}_{j-1}}{\mathbf{t}_{j-1}^{'}\mathbf{t}_{j-1}}\right] = \mathbf{X}_{i+1}\mathbf{Z} = \left[\mathbf{X}_i - \frac{\mathbf{t}_i\mathbf{t}_i^{'}\mathbf{X}_i}{\mathbf{t}_i^{'}\mathbf{t}_i}\right]\mathbf{Z}\end{aligned}$$

donde $\mathbf{Z}$ es un producto de matrices. Luego

$$\mathbf{t}_i'\mathbf{X}_j = \left[\mathbf{t}_i'\mathbf{X}_i - \frac{\mathbf{t}_i'\mathbf{t}_i\mathbf{t}_i'\mathbf{X}_i}{\mathbf{t}_i'\mathbf{t}_i}\right]\mathbf{Z} = (\mathbf{t}_i'\mathbf{X}_i - \mathbf{t}_i'\mathbf{X}_i)\mathbf{Z} = 0 \quad \text{para } i < j. \tag{4.1}$$

Por tanto

$$\mathbf{t}_i'\mathbf{t}_j \overset{\mathbf{t}=\mathbf{X}\mathbf{w}}{=} \overbrace{\mathbf{t}_i'\mathbf{X}_j}^{=0}\mathbf{w}_j = 0 \quad \text{para } i < j.$$

□

Los vectores $\mathbf{t}_i$ forman una base ortogonal en el espacio generado por las columnas de $\mathbf{X}$. Si suponemos que el rango de $\mathbf{X}$ es k, podemos escribir la matriz $\mathbf{X}$ como

$$\mathbf{X} = \sum_{i=1}^{k} \mathbf{t}_i\mathbf{p}_i' + \mathbf{X}_0 \tag{4.2}$$

donde $\mathbf{X}_0$ es una matriz ortogonal a $\mathbf{Y}$. En lo que sigue supongamos que $\mathbf{X}_0$ es cero, porque $\mathbf{X}_0$ no aporta nada a $\mathbf{Y}$.

Propiedad 4.1.3. *Los vectores $\boldsymbol{w}_i$ son ortogonales a los vectores $\boldsymbol{p}_j$ para $i < j$*

$$(\boldsymbol{w}_i, \boldsymbol{p}_j) = \boldsymbol{w}_i'\boldsymbol{p}_j = 0 \quad \textit{para } i < j.$$

Demostración. De la ecuación (4.1) obtenemos

$$\mathbf{w}_i'\mathbf{p}_j \overset{\mathbf{p}=\frac{\mathbf{X}'\mathbf{t}}{\mathbf{t}'\mathbf{t}}}{=} \frac{\mathbf{w}_i'\mathbf{X}_j'\mathbf{t}_j}{\mathbf{t}_j'\mathbf{t}_j} = \frac{\mathbf{w}_i'(\overbrace{\mathbf{t}_j'\mathbf{X}_j}^{=0})'}{\mathbf{t}_j'\mathbf{t}_j} = 0 \quad \text{para } i < j.$$

□

Propiedad 4.1.4. *Los vectores $\boldsymbol{p}_i$ son ortogonales en el espacion de kernel de $\boldsymbol{X}$:*

$$(\boldsymbol{p}_i, \boldsymbol{p}_j)_X = \boldsymbol{p}_i'(\boldsymbol{X}'\boldsymbol{X})^{-}\boldsymbol{p}_j = 0 \quad \textit{para } i \neq j$$

siendo $(\boldsymbol{X}'\boldsymbol{X})^{-}$ la inversa generalizada de $\boldsymbol{X}'\boldsymbol{X}$.

Demostración. A partir de (4.2) obtenemos

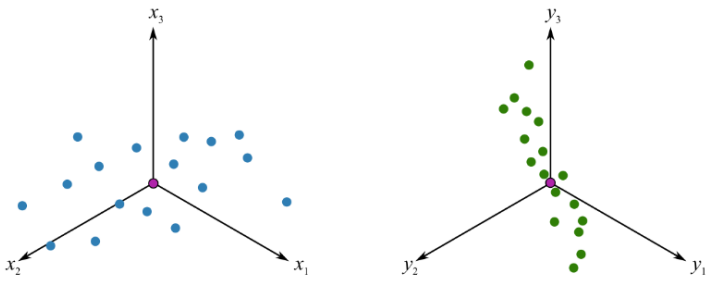
$$\mathbf{X}'\mathbf{X} = \sum_{i=1}^{k}(\mathbf{t}_i, \mathbf{t}_i)\mathbf{p}_i\mathbf{p}_i' = \mathbf{PLP}'$$

donde $\mathbf{L}$ es una matriz diagonal que contiene los coeficientes $(\mathbf{t}_i, \mathbf{t}_i)$. De la teoría de los espacios de kernel se deduce esta propiedad. □

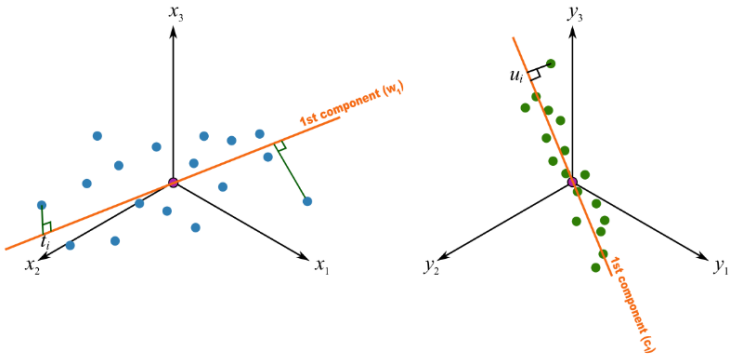
Estas cuatro propiedades se derivan de la forma en que se construyen las matrices residuales $\mathbf{X}_i$ y no dependen de la forma en que se construye un nuevo vector $\mathbf{t}$.

4.2. Interpretación geométrica

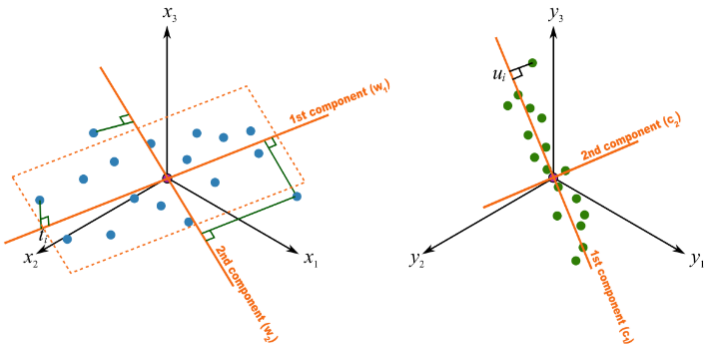
Para visualizar mejor la interpretación geométrica, suponemos que tenemos tres variables predictoras y tres variables respuesta. Partimos de unos datos estandarizados, es decir, el sistema de coordenadas está centrado en el origen. Hay un punto en **X** para cada punto en **Y** y cada punto representa una fila de la correspondiente matriz **X** e **Y**.



Las puntuaciones son la proyección perpendicular de cada punto de datos en cada vector de dirección **w** o **c**. El punto en que cada observación aterriza se le llama proyección en el espacio de **X**, $\mathbf{t}_i$, o proyección en el espacio de **Y**, $\mathbf{u}_i$



La segunda componente es perpendicular a la primera en el espacio de **X**. Sin embargo, la segunda componente no es necesariamente ortogonal en el espacio de **Y**, aunque a menudo está cerca de ser ortogonal.



4.3. Interpretación

Sea **f** una puntuación de **X** y **g** una puntuación de **Y**:

$$\begin{aligned} \mathbf{f} &= \mathbf{Xd} \\ \mathbf{g} &= \mathbf{Ye} \end{aligned}$$

La covarianza muestral entre las dos puntuaciones es dada por

$$Cov(\mathbf{f}, \mathbf{g}) = \frac{\mathbf{f}'\mathbf{g}}{N}.$$

El objetivo de PLSR es maximizar la covarianza entre las puntuaciones de **X** e **Y**. Como el problema de maximización no es único entonces es necesaria una restricción, es usual la restricción $\|\mathbf{d}\| = \|\mathbf{e}\| = 1$. El problema de maximización entonces es el siguiente:

$$\begin{aligned} \text{máx} \quad & [Cov(\mathbf{f}, \mathbf{g})]^2 = [Cov(\mathbf{Xd}, \mathbf{Ye})]^2 \\ s.a. \quad & \|\mathbf{d}\| = 1 \\ & \|\mathbf{e}\| = 1 \end{aligned}$$

Los vectores **w** y **c** del algoritmo NIPALS son la solución al problema.

Demostración. Consideramos la descomposición en valores singulares de **X**'**Y**

$$\mathbf{X}'\mathbf{Y} = \sum a_i \mathbf{f}_i \mathbf{g}_i'.$$

El valor singular a_1 tiene la interpretación

$$(a_1)^2 = \text{máx}(\mathbf{d}'\mathbf{X}'\mathbf{Ye})^2 \quad \text{para } \|\mathbf{d}\| = \|\mathbf{e}\| = 1$$

con el máximo alcanzado en $\mathbf{d} = \mathbf{f}_1$ y $\mathbf{e} = \mathbf{g}_1$. En el algoritmo PLSR tenemos $\mathbf{w} = \mathbf{f}_1$ y $\mathbf{c} = \mathbf{g}_1$, lo que demuestra el problema de maximización. □

Las puntuaciones **t** y **u** en el algoritmo NIPALS tienen la interpretación de que son las componentes en el espacio de **X** e **Y** que hacen máxima la covarianza entre todas las componentes en el espacio de **X** e **Y**.

4.4. Coeficientes de regresión

Los vectores de pesos **w** se aplican a matrices residuales diferentes, es decir

$$\mathbf{t} = \mathbf{X}\mathbf{w}$$

donde **X** es la matriz residual correspondiente a la iteración anterior del algoritmo. Esto obscurece la interpretación de las componentes PLS **t**, principalmente porque se va perdiendo visión de lo que está en las matrices residuales cuando se va aumentando de dimensión. Algunas variables de **X** se usan en las primeras componentes, otras mucho más tarde. Esta relación entre componentes y variables se muestra mejor por las cargas **p**. De hecho, los vectores de pesos **w** se usan menos en la interpretación de modelos de regresión PLS que los vectores de carga. Por lo tanto, una alternativa es reexpresar las componentes en términos de los datos originales

$$\mathbf{T} = \mathbf{X}\mathbf{W}^*.$$

Reemplanzado esta expresión de **T** en (3.1) se obtiene

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^*\mathbf{C}' + \mathbf{E}_Y^* = \mathbf{X}\mathbf{B} + \mathbf{E}_Y^*. \tag{4.3}$$

Para estimar los coeficientes de regresión **B**, se expresa **X** de la siguiente manera:

$$\mathbf{X} = \mathbf{T}\mathbf{R}\mathbf{W}' \tag{4.4}$$

donde $\mathbf{R}_{k\times k} = \mathbf{P}'\mathbf{W}$ es una matriz triangular superior y por tanto invertible (por la propiedad 4.1.3).

Una inversa generalizada de **X** puede escribirse de la forma

$$\mathbf{X}^+ = \mathbf{W}\mathbf{R}^{-1}\mathbf{T}'$$

es decir, $\mathbf{X}\mathbf{X}^+\mathbf{X} = \mathbf{X}$. Además $\mathbf{X}^+$ satisface

$$\mathbf{X}^+\mathbf{X}\mathbf{X}^+ = \mathbf{X}^+ \quad \mathbf{X}\mathbf{X}^+ = (\mathbf{X}\mathbf{X}^+)' \quad (\mathbf{X}^+\mathbf{X})' = \mathbf{X}^+\mathbf{X}$$

que define la *inversa generalizada de Moore-Penrose*.

Se puede demostrar [8] que la inversa generalizada de Moore-Penrose da la solución de mínima norma al problema de mínimos cuadrados, es decir,

$$\mathbf{B} = \mathbf{X}^{+}\mathbf{Y} = \mathbf{W}\mathbf{R}^{-1}\mathbf{T}'\mathbf{Y} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{T}'\mathbf{Y} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}'$$

es la solución que minimiza $\mathbf{B}'\mathbf{B}$ en el caso de que $\mathbf{X}'\mathbf{X}$ sea singular.

La expresión de los pesos transformados $\mathbf{W}^{*}$ se obtiene insertando la expresión anterior de los coeficientes de regresión en (4.3)

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^{*}\mathbf{C}' + \mathbf{E}_{Y}^{*} = \mathbf{X}\mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}' + \mathbf{E}_{Y}^{*}.$$

Por tanto

$$\mathbf{W}^{*} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}. \tag{4.5}$$

CAPÍTULO 5

Otros algoritmos

Otros algoritmos utilizados en PLSR son el algoritmo Kernel y SIMPLS.

5.1. Algoritmo Kernel (Núcleo)

El algoritmo Kernel fue propuesto por Lindgren en 1993. El nombre resulta de usar la descomposición espectral de las llamadas matrices kernel, de productos de **X** e **Y**.

Algoritmo 2 (Kernel)**.** 1. $\boldsymbol{w}$ *es el autovector asociado al mayor valor singular de* $\boldsymbol{X'YY'X}$

2. $\boldsymbol{w} = \dfrac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$

3. $\boldsymbol{t}=\boldsymbol{Xw}$ *con* $\|\boldsymbol{t}\| = 1$

4. $\boldsymbol{c}$ *es el autovector asociado al mayor valor singular de* $\boldsymbol{Y'XX'Y}$

5. $\boldsymbol{c} = \dfrac{\boldsymbol{c}}{\|\boldsymbol{c}\|}$

6. $\boldsymbol{u}=\boldsymbol{Yc}$ *con* $\|\boldsymbol{u}\| = 1$

7. $\boldsymbol{p}$ *se obtiene mediante mínimos cuadrados ordinarios en relación al modelo* $\boldsymbol{X} = \boldsymbol{TP}' + \boldsymbol{E}_X$

$$\boldsymbol{p}' = (\boldsymbol{t}'\boldsymbol{t})^{-1}\boldsymbol{t}'\boldsymbol{X} = \boldsymbol{t}'\boldsymbol{X} = \boldsymbol{w}'\boldsymbol{X}'\boldsymbol{X}$$

8. ***q*** *se obtiene a partir del modelo de regresión* $\boldsymbol{Y} = \boldsymbol{UQ}' + \boldsymbol{E}_Y$

$$\boldsymbol{q} = (\boldsymbol{u}'\boldsymbol{u})^{-1}\boldsymbol{u}\boldsymbol{Y}$$

9. *Eliminar el efecto de la componente* ***t*** *de* ***X*** *obteniendo la siguiente matriz residual*

$$\boldsymbol{X} \rightarrow \boldsymbol{X} - \boldsymbol{t}\boldsymbol{p}'$$

No es necesario eliminar el efecto de la componente ***t*** *de* ***Y****:*

$$\begin{aligned} \boldsymbol{X}^* &= \boldsymbol{X} - \boldsymbol{t}\boldsymbol{p}' = \boldsymbol{X} - \boldsymbol{t}\boldsymbol{t}'\boldsymbol{X} = \underbrace{(\boldsymbol{I} - \boldsymbol{t}\boldsymbol{t}')}_{\boldsymbol{G}}\boldsymbol{X} \\ \boldsymbol{Y}^* &= \boldsymbol{G}\boldsymbol{Y} \end{aligned}$$

donde ***G*** *es una matriz simétrica (****G=G'****) e idempotente (*$\boldsymbol{G}^2 = \boldsymbol{G}$*). Por tanto:*

$$\begin{aligned} \boldsymbol{X}^{*'}\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{X}^* &= \boldsymbol{X}'\boldsymbol{G}'\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{G}\boldsymbol{X} = \boldsymbol{X}'\boldsymbol{G}\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{G}\boldsymbol{X} = \boldsymbol{X}'\boldsymbol{G}^2\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{G}^{\,2}\boldsymbol{X} = \\ \boldsymbol{X}'\boldsymbol{G}\boldsymbol{G}\boldsymbol{Y}\boldsymbol{Y}'\boldsymbol{G}\boldsymbol{G}\boldsymbol{X} &= (\boldsymbol{X}'\boldsymbol{G}')(\boldsymbol{G}\boldsymbol{Y})(\boldsymbol{Y}'\boldsymbol{G}')(\boldsymbol{G}\boldsymbol{X}) = \boldsymbol{X}^{*'}\boldsymbol{Y}^*\boldsymbol{Y}^{*'}\boldsymbol{X}^* \end{aligned}$$

$$\begin{aligned} \boldsymbol{Y}'\boldsymbol{X}^*\boldsymbol{X}^{*'}\boldsymbol{Y} &= \boldsymbol{Y}'\boldsymbol{G}\boldsymbol{X}\boldsymbol{X}'\boldsymbol{G}'\boldsymbol{Y} = \boldsymbol{Y}'\boldsymbol{G}\boldsymbol{X}\boldsymbol{X}'\boldsymbol{G}\boldsymbol{Y} = \boldsymbol{Y}'\boldsymbol{G}^2\boldsymbol{X}\boldsymbol{X}'\boldsymbol{G}^2\boldsymbol{Y} = \\ \boldsymbol{Y}'\boldsymbol{G}\boldsymbol{G}\boldsymbol{X}\boldsymbol{X}'\boldsymbol{G}\boldsymbol{G}\boldsymbol{Y} &= (\boldsymbol{Y}'\boldsymbol{G}')(\boldsymbol{G}\boldsymbol{X})(\boldsymbol{X}'\boldsymbol{G}')(\boldsymbol{G}\boldsymbol{Y}) = \boldsymbol{Y}^{*'}\boldsymbol{X}^*\boldsymbol{X}^{*'}\boldsymbol{Y}^* \end{aligned}$$

Es decir, para hallar los vectores ***w*** *y* ***c*** *de la siguiente iteración no es necesario eliminar el efecto de la componente* ***t*** *de* ***Y****.*

Las siguientes componentes se obtienen de la misma manera usando la matriz residual **X** obtenidaa después del cálculo de la componente anterior.

Los resultados de este algoritmo son los mismos que el algoritmo NIPALS ya que el efecto de las componentes se elimina de la misma manera, luego solo las componentes se calculan de manera diferente pero con el mismo resultado numérico.

La interpretación de PLSR usando este algoritmo es la misma qua utilizando el algoritmo NIPALS ya que resuelve el mismo problema de optimización . Además los coeficientes de regresión también son los mismos que los obtenidos para el algortimo NIPALS, es decir, $\mathbf{B} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}'$.

Esta versión del algoritmo Kernel está especialmente diseñada para matrices de datos con un gran número de observaciones N. En este caso -y si las dimensiones de las matrices de datos no son demasiado grandes- las matrices kernel tienen

una dimensión mucho más pequeña que N y la descomposición espectral es rápida de calcular. También existe una versión alternativa al algoritmo Kernel para un gran número de variables predictoras y respuesta, y para un número moderado de observaciones N, llamado **algoritmo Wide Kernel**.

Algoritmo 3 (Wide Kernel)**.**

1. ***w*** *es el autovector asociado al mayoyr valor singular de* ***XX'YY'***
2. $\boldsymbol{w} = \dfrac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$
3. ***t=Xw*** *con* $\|\boldsymbol{t}\| = 1$
4. ***c*** *es el autovector asociado al mayor valor singular de* ***YY'XX'***
5. $\boldsymbol{c} = \frac{\boldsymbol{c}}{\|\boldsymbol{c}\|}$
6. ***u=Yc*** *con* $\|\boldsymbol{u}\| = 1$
7. $\boldsymbol{p}' = \boldsymbol{w}'\boldsymbol{X}'\boldsymbol{X}$
8. $\boldsymbol{q} = (\boldsymbol{u}'\boldsymbol{u})^{-1}\boldsymbol{u}\boldsymbol{Y}$
9. *Eliminar el efecto de la componente* ***t*** *de* ***X*** *obteniendo la siguiente matriz residual*

$$\boldsymbol{X} \;\rightarrow\; \boldsymbol{X} - \boldsymbol{t}\boldsymbol{p}^{'}$$

5.2. Algoritmo SIMPLS

El algoritmo SIMPLS es una implementación algorítmica modificada del NIPALS y dicha modificación conduce al cálculo directo de $\mathbf{W}^*$. De esta manera se evita la construcción de las matrices residuales, que resultan de eliminar el efecto de las componentes, y el cálculo de los pesos $\mathbf{W}$. Por tanto el cálculo explícito de la matriz inversa (4.5) también se elude. La nueva definición de $\mathbf{W}^*$ es similar, pero no idéntica, a la introducida en $\mathbf{T} = \mathbf{X}\mathbf{W}^*$. De hecho la nueva $\mathbf{W}^*$ contiene vectores de pesos normalizados como $\mathbf{W}$ en el algoritmo NIPALS.

Sean dos puntuaciones $\mathbf{t} = \mathbf{X}\mathbf{w}^*$ y $\mathbf{u} = \mathbf{Y}\mathbf{c}$, los vectores de pesos $\mathbf{w}^*$ y $\mathbf{u}$ satisfacen el problema de optimización siguiente:

$$\begin{aligned} \text{máx} \quad & Cov(\mathbf{t}, \mathbf{u}) = Cov(\mathbf{X}\mathbf{w}^*, \mathbf{Y}\mathbf{c}) \\ s.a. \quad & \mathbf{w}^{*'}\mathbf{w}^* = 1 \\ & \mathbf{c}'\mathbf{c} = 1 \\ & \mathbf{t}_j'\mathbf{t}_i = 0 \text{ para } i > j \end{aligned}$$

Sin la última restricción sólo hay una solución directa: $\mathbf{w}^*$ y $\mathbf{c}$ son los primeros vectores singulares izquierdo y derecho de la matriz de productos cruzados $\mathbf{S} = \mathbf{X}'\mathbf{Y}$. Sin embargo, con el fin de obtener más de una solución y generar un conjunto de puntuaciones ortogonales de $\mathbf{X}$, se tiene que añadir la última restricción. Por lo tanto se requiere

$$\mathbf{t}_j'\mathbf{t}_i = \mathbf{t}_j'\mathbf{X}\mathbf{w}_i^* = (\mathbf{t}_j'\mathbf{t}_j)\mathbf{p}_j'\mathbf{w}_i^* = 0 \quad \text{para } i > j$$

donde

$$\mathbf{p} = \frac{\mathbf{X}\mathbf{t}}{\mathbf{t}'\mathbf{t}}.$$

Como $\mathbf{t}_j'\mathbf{t}_j \neq 0$ entonces $\mathbf{p}_j'\mathbf{w}_i^* = 0$, cualquier vector de pesos nuevo $\mathbf{w}_i^*$ ($1 < i \leq k$) debe ser ortogonal a todos los vectores de carga $\mathbf{p}_j$ ($i > j$) precedentes, es decir, a las columnas de $\mathbf{P}_{i-1} \equiv [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{i-1}]$. Sea $\mathbf{P}_{i-1}^{\perp}$ el proyector ortogonal requerido

$$\mathbf{P}_{i-1}^{\perp} = \mathbf{I} - \mathbf{P}_{i-1}(\mathbf{P}_{i-1}'\mathbf{P}_{i-1})^{-1}\mathbf{P}_{i-1}'$$

por lo tanto

$$\mathbf{w}_i^* = \mathbf{P}_{i-1}^{\perp}\mathbf{w}_i^* \quad \text{para } 1 < i \leq k$$

Estas dos últimas ecuaciones explican concisamente la tercera restricción del problema de optimización. La solución para $\mathbf{c}$ y $\mathbf{w}^*$ es dada ahora por el primer par de vectores singulares de la descomposición de valores singulares de $\mathbf{S}$ proyectada en un subespacio ortogonal a $\mathbf{P}_{i-1}$, es decir, la descomposición de valores singulares de $\mathbf{P}_{i-1}^{\perp}\mathbf{S}$.

Algoritmo 4 (SIMPLS). *1. Calcular* $\boldsymbol{S} = \boldsymbol{X}'\boldsymbol{Y}$

2. *Para* $i = 1$, $\boldsymbol{S}_i = \boldsymbol{S}$
3. *Para* $1 < i \leq k$, $\boldsymbol{S}_i = \boldsymbol{S}_{i-1} - \boldsymbol{P}_{i-1}(\boldsymbol{P}_{i-1}'\boldsymbol{P}_{i-1})^{-1}\boldsymbol{P}_{i-1}'\boldsymbol{S}_{i-1}$
4. *Calcular* $\boldsymbol{w}_i^*$ *como el primer vector singular izquierdo de* $\boldsymbol{S}_i$
5. $\boldsymbol{w}_i^* = \dfrac{\boldsymbol{w}_i^*}{\|\boldsymbol{w}_i^*\|}$
6. $\boldsymbol{t}_i = \boldsymbol{X}\boldsymbol{w}_i^*$
7. $\boldsymbol{t}_i = \dfrac{\boldsymbol{t}_i}{\|\boldsymbol{t}_i\|}$
8. $\boldsymbol{p}_i = \dfrac{\boldsymbol{X}\boldsymbol{t}_i}{\boldsymbol{t}_i'\boldsymbol{t}_i}$

Este algoritmo coincide con el NIPALS y Kernel en el caso de una sola variable respuesta.

Las predicciones con este algoritmo se obtendría de la siguiente manera:

$$\hat{\mathbf{Y}} = \mathbf{T}\mathbf{T}'\mathbf{Y} = \mathbf{X}\mathbf{W}^*\mathbf{W}^{*'}\mathbf{X}'\mathbf{Y} = \mathbf{X}\mathbf{W}\mathbf{W}^{*'}\mathbf{S}$$

Por tanto los coeficientes de regresión son:

$$\mathbf{B} = \mathbf{W}^*\mathbf{W}^{*'}\mathbf{S} = \mathbf{W}^*\mathbf{T}'\mathbf{Y} = \mathbf{W}^*\mathbf{C}'.$$

CAPÍTULO 6

PCR Y PLSR con el paquete pls de R

El **paquete pls** implementa las técnicas de regresión por componentes principales y regresión por mínimos cuadrados parciales.

```
#Cargar la librería
library(pls)
```

Las principales funciones del paquete *pls* para ajustar modelos son **pcr** y **plsr**. Los argumentos de dichas funciones son:

- **formula**: fórmula describiendo el modelo. Es de la forma *respuesta*∼*términos* donde *respuesta* es el nombre del vector o matriz de variables respuesta (para modelos de respuesta múltiple) y *términos* es el nombre de una o más matrices de variables predictoras separadas por +.
- **ncomp**: número de componentes a incluir en el modelo (es opcional, si no se indica se utiliza el número posible de componentes).
- **data**: data frame opcional con los datos para ajustar el modelo. Si no se indica, las variables especificadas en la fórmula se buscan en el entorno global (el espacio de trabajo del usuario).
- **na.action**: función que indica cómo tratar los valores perdidos. Los datos perdidos pueden ser un problema pues los algoritmos implementados en *pcr* y *pls* no manejan valores perdidos intrínsecamente por lo que deben ser eliminados. Pueden ser tratados de la siguiente manera:

  -**na.action=na.omit**: cualquier observación con valores perdidos se elimina completamente del modelo.

-**na.action=na.exclude**: los valores perdidos se eliminan del proceso de ajuste, pero se incluyen como NAs (valores perdidos) en los valores residuales y ajustados.

- **method**: método de regresión multivariante a utilizar. El paquete *pls* implementa cuatro algoritmos PLSR: Kernel, Wide Wernel, NIPALS, SIMPLS. El algoritmo que usa por defecto R es el Kernel, para el resto de algoritmos se escribe:

  - **method=widekernelpls**
  - **method=oscorespls**
  - **method=simpls**

  Para PCR solo hay un algoritmo disponible que usa la descomposición en valores singulares ("svdpc").

- **scale**: lógico. Si scale=TRUE los datos se tipifican.

- **validation**: carácter. Argumento que controla la validación cruzada que comúnmente se utiliza para determinar el número óptimo de componentes a tener en cuenta. (Ver en la siguiente sección)

  -**validation=CV**: validación cruzada ordinaria.

  -**validation=LOO**: validación cruzada dejando uno fuera.

- **model**: lógico. Si es TRUE, devuelve el frame del modelo.

- **x**: lógico. Si es TRUE, devuelve la matriz del modelo.

- **y**: lógica. Si es TRUE, devuelve la respuesta.

6.1. Selección del número de componentes

En cualquier modelo empírico, es esencial determinar la complejidad (en este caso número de componentes) correcta de un modelo. Con numerosas variables predictoras que además están muy correlacionadas existe un riesgo sustancial de ajuste excesivo, es decir, obtener un modelo bien ajustado con poco o ninguna potencia significativa. Por lo tanto, es necesaria una prueba estricta de la significación predictiva de cada componente y luego deternerse cuando las componentes comienzan a no ser significativas.

La validación cruzada (CV) es una herramienta empleada habitualmente para configurar la complejidad de un modelo de predicción.

Supongamos que los datos están separados en un conjunto de entrenamiento y un conjunto test. El conjunto de entrenamiento se utiliza para ajustar los modelos y determinar la complejidad del modelo mientras que el conjunto test se deja aparte y solo se utiliza para estimar el error de generalización del modelo finalmente seleccionado. Los tamaños usuales de estos son conjuntos son $75\,\%$ (entrenamiento) y $25\,\%$ (test).

Con la validación cruzada se trata de obtener k' divisiones aleatorias del conjunto de entrenamiento en dos grupos: conjunto de estimación y conjunto de validación. Pues bien, dado un modelo con una determinada complejidad, para cada división se ajusta el modelo sobre el conjunto de estimación y se mide el error sobre el conjunto de validación ($E_i \;\; 1 \leq i \leq k'$). La media de estos k' errores de validación proporcionan una estimación del error de validación para la complejidad considerada

$$ECV = \frac{1}{k'} \sum_{i=1}^{k'} E_i.$$

Un caso particular de la validación cruzada es la validación cruzada dejando uno fuera (leave-one-out o también conocida como Jackknife). En este caso se hacen tantas divisiones como observaciones, es decir, $k' = N$. Esto se vuelve computacionalmente complicado cuando se tienen muchas observaciones. Además cuanto mayor sea k', menor es el sesgo del estimador, pero a cambio mayor será su varianza y el contrario. Se recomienda que $k' = 10$, de hecho este es el valor por defecto que utiliza R.

La elección del mejor número de componentes es difícil y generalmente se hace visualizando el **gráfico de validación** en el que se representan los errores por validación cruzada en función del número de componentes. En los casos en los que se construye un gran número de modelos, esta opción debe ser automatizada. El paquete *pls* contiene una función llamada **selectNcomp** que implementa dos propuestas. En ambas propuestas se utilizan los resultados de validación cruzada.

- Una de ellas se denomina *regla de una sigma* (**method=onesigma**). Para cada número de componentes i se obtiene un error por validación cruzada (ECV_i) y una desviación típica (σ_i). Se selecciona el error más pequeño por validación cruzada ($E_j = \min E_i$) y se le suma su correspondiente desviación típica, es decir, $E_j + \sigma_j = E_j^*$. Se selecciona el número más pequeño de componentes cuyo error E_i sea inferior a E_j^*

- La otra se conoce como *prueba de aleatorización* (**method=randomization**). En primer lugar se calcula el error por validación cruzada más pequeño lo que conduce al modelo de referencia. El método de la prueba de aleatorización verifica si los errores de predicción al cuadrado de los modelos con menos componentes son significativamente mayores que en el modelo de referencia ($H_1 : MSEP_A > MSEP_B$):

  1. Calcular $d_i = e_{Ai}^2 - e_{Bi}^2$, $i = 1, \ldots, N$ donde $e_{Ai} = y_i - \hat{y}_{Ai}$ y $e_{Bi} = y_i - \hat{y}_{Bi}$
  2. Calcular $T_{obs} = \hat{MSEP}_A - \hat{MSEP}_B = \overline{d} = \frac{1}{N}\sum d_i$
  3. Iterar los pasos 3a-3b m veces:
     a. Adjuntar signos aleatorios a d_i, $i = 1, \ldots, N$
     b. Calcular $T^* = \overline{d}^*$
  4. Calcular el nivel de significación $p = \frac{l}{m+1}$ donde l es el número de veces que $T^* > T_{obs}$

  Luego cada modelo considerado lleva asociado un valor de p. El modelo más pequeño no significativamente peor que el modelo de referencia es devuelto como el seleccionado.

Los dos métodos no siempre coinciden pero por lo general están muy cerca.

La validación cruzada es un procedimiento exigente desde el punto de vista computacional. Hay que ajustar un nuevo modelo por cada división. Las funciones de ajuste subyacentes se han optimizado y la implementación de la validación cruzada que se utiliza al especificar el argumento *validation* evita cualquier cálculo innecesario. Aún así, la validación cruzada puede llevar mucho tiempo, para modelos con matrices grandes, muchas componentes o muchas divisiones.

Por defecto, los cálculos de validación cruzada en *pls* se realizan en serie, en un CPU (núcleo). Sin embargo hay disponible un paquete llamado **parallel** para ejecutar los cálculos en paralelo, en máquinas con varios CPU o en varias máquinas. Este paquete tiene varias formas de ejecutar los cálculos en paralelo, y no todos están disponibles en todos los sistemas.

La opción en paralelo se puede especificar como un objeto de clúster creado por el **makeCluster** desde el paquete *parallel*. Hay varios tipos de clústers disponibles pero en este caso se usará **PSOCK** ya que es compatible con Windows.

6.2. Ejemplo de PCR con una variable respuesta

La libería *pls* incluye el conjunto de datos **gasoline**:

```
data(gasoline)
help(gasoline)
```

Se trata de 60 muestras de gasolina en las que se han estudiado 401 mediciones de reflectancia difusa de 900 a 1700 nm en intervalos de 2 nm (espectro NIR) y número de octano, o a veces denominado octanaje, que mide la capacidad de antidetonante del carburante (como la gasolina) cuando se comprime dentro del cilindro de un motor.

- $\mathbf{X}_{60\times 401}$: matriz de variables predictoras.
- $\mathbf{Y}_{60\times 1}$: vector de la variable respuesta.

Partición de los datos en entrenamiento y test

Los datos de *gasoline* se dividen en un conjunto de entrenamiento formado por el 75 % de los datos y en un conjunto test compuesto por el 25 % restante:

```
set.seed(1)
n<-nrow(gasoline)
indices<-1:n
indientr=sample(indices,floor(n*0.75))
gasTrain<-gasoline[indientr,]#conjunto de entrenamiento
dim(gasTrain)
```

```
## [1] 45  2
```

Por tanto, el conjunto de entrenamiento está formado por 45 muestas.

```
inditest<-setdiff(indices,indientr)
gasTest<-gasoline[inditest,]#conjunto test
dim(gasTest)
```

```
## [1] 15  2
```

Y el conjunto test por 15 muestras.

Determinar el número de componentes principales

En primer lugar se ajusta el modelo con 10 componentes como máximo haciendo uso de la validación cruzada ordinaria y se visualiza de manera general los resultados de ajuste y validación:

```
gas<-pcr(octane ~ NIR,ncomp=10,data=gasTrain,
            scale=TRUE,validation="CV")
summary(gas)
```

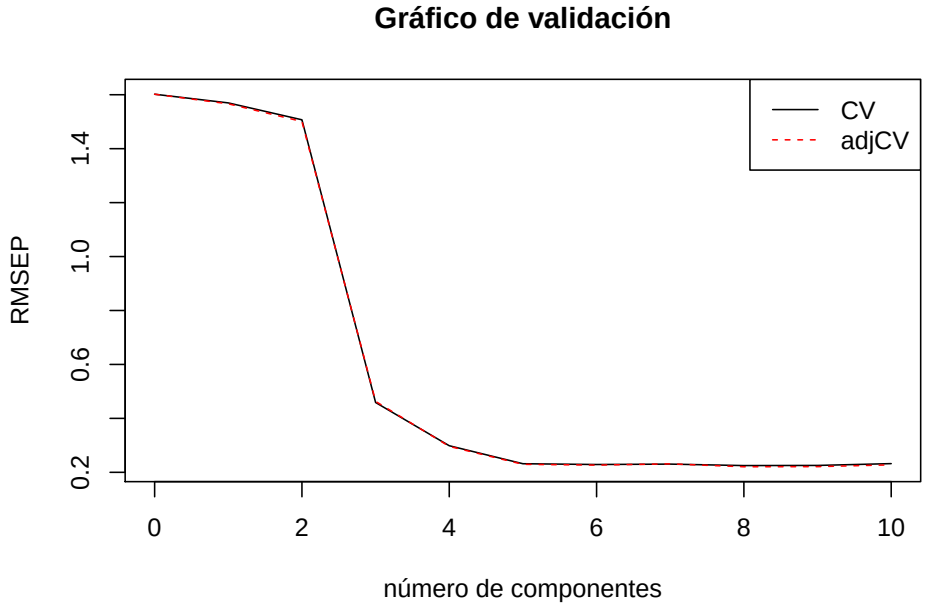
```
## Data:    X dimension: 45 401
##  Y dimension: 45 1
## Fit method: svdpc
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps   2 comps  3 comps
## CV           1.602    1.570     1.507   0.4589
## adjCV        1.602    1.566     1.501   0.4630
##            4 comps  5 comps   6 comps  7 comps
## CV          0.2986   0.2318    0.2289   0.2304
## adjCV       0.2961   0.2297    0.2267   0.2314
##            8 comps  9 comps  10 comps
## CV          0.2250   0.2255    0.2322
## adjCV       0.2206   0.2210    0.2277
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps   5 comps
## X         71.417    88.71    94.25    97.74     98.58
## octane     7.318    22.46    92.91    97.18     98.38
##         6 comps  7 comps  8 comps  9 comps  10 comps
## X         98.98    99.21    99.39    99.50     99.60
## octane    98.59    98.62    98.97    98.97     98.98
```

Los resultados de validación son la raíz cuadrada de los errores cuadráticos medios de predicción (RMSEP). Hay dos estimaciones por validación cruzada:

- *CV* es la estimación por validación cruzada ordinaria
- *adjCV* que es la estimación por validación cruzada con sesgo corregido.

1. Una forma de seleccionar el número óptimo de componentes principales es visualizando el gráfico de validación, como se ha comentado anteriormente.

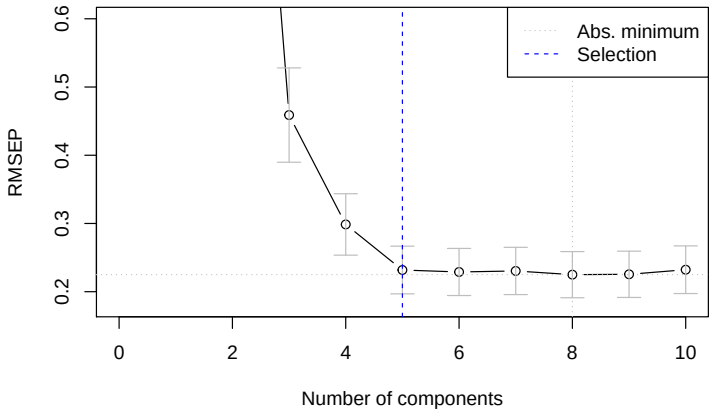
```
plot(RMSEP(gas),main="Gráfico de validación",xlab=
     "número de componentes",legendpos="topright")
#también se puede representar así:
#validationplot(gas,val.type="RMSEP")
#plot(gas,plottype="validation")
```



Cuatro componentes parecen ser suficientes ya que el RMSEP no muestra una disminución significativa para más de 4 componentes. En este caso, RMSEP=0.2986.

2. Otra opción es con la función *selecNcomp*. Usando la regla de una sigma:

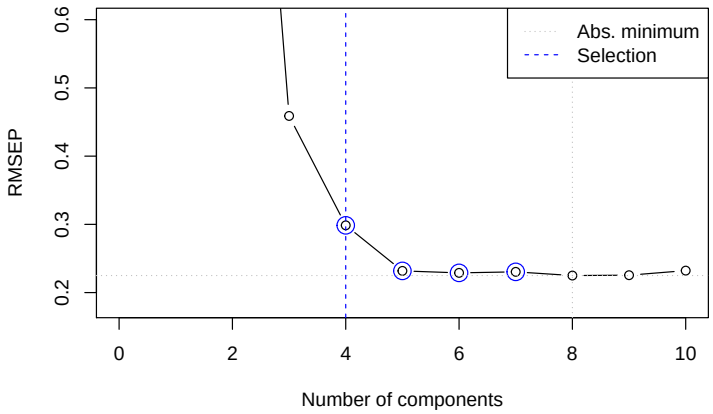
```
ncomp.onesigma<-selectNcomp(gas,method="onesigma",
                            plot=TRUE,ylim=c(.18, .6))
```



La línea punteada gris indica el mínimo global de la curva de validación cruzada y la línea discontinua azul sugiere el número óptimo de componentes. Además se muestra la anchura de los intervalos en los que se basa la regla, es decir, los errores E_i son los puntos y los intervalos son $(E_i + \sigma_i, E_i - \sigma_i)$. Luego según este método, se seleccionarían 5 componentes.

Por otro lado, usando la prueba prueba de aleatorización:

```
ncomp.per<-selectNcomp(gas,method="randomization",
                       plot=TRUE,ylim=c(.18, .6))
```



Las líneas azul y gris tienen el mismo significado de antes. En este caso, los círculos más grandes azules indican los modelos que han sido probados en el método de aleatorización. Luego según esta prueba, se seleccionarían 4 componentes.

3. La alternativa a la validación cruzada para determinar el número de componentes, es decir la validación cruzada paralela, se utiliza de la siguiente manera pero en este caso no es necesario ya que no disponemos de numesoras muestras.

```
library(parallel)
pls.options(parallel=makeCluster(4,type="PSOCK"))
#con 4CPUs
gas.cvparalelo<-gas<-pcr(octane ~NIR,ncomp=10,
     data=gasTrain,scale=TRUE,validation="CV")
stopCluster(pls.options()$parallel)
```

Por tanto si se seleccionan 4 componentes, el modelo ajustado sería:

```
gaspcr<-pcr(octane ~ NIR,ncomp=4,data=gasTrain,
            scale=TRUE, validation="CV")
```

Valores ajustados y residuos

Del modelo ajustado se pueden extraer los valores ajustados así como los residuos:

```
fitted.values(gaspcr)
```

```
residuals(gaspcr)
```

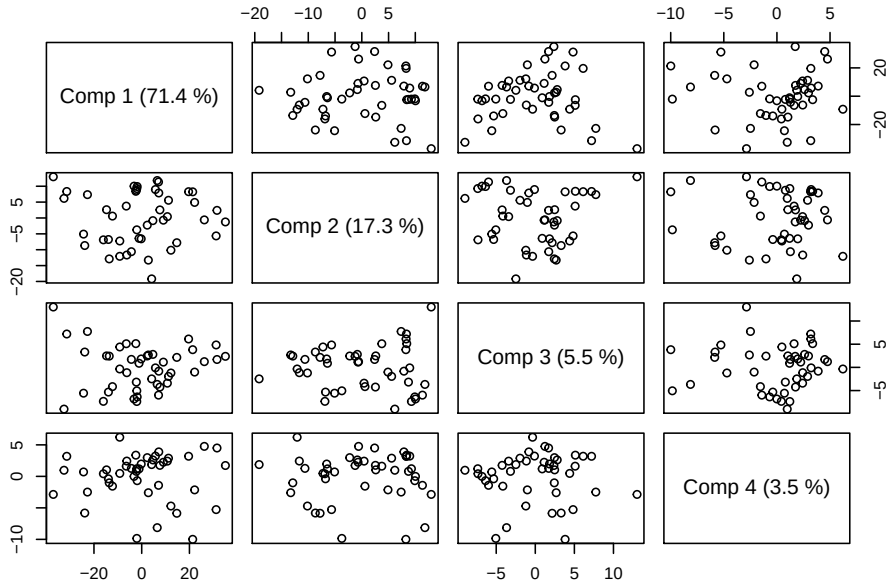
Puntuaciones de X (Componentes): T

```
dim(gaspcr$scores)
```

```
## [1] 45  4
```

Las puntuaciones de **X** para el número de componentes seleccionadas se representan en el siguiente gráfico y éste se usa para buscar patrones, grupos o valores atípicos en los datos.

```
plot(gaspcr,plottype="scores",comps=1:4)
#otra forma
#scoreplot(gas,comps=1:4)
```



En este ejemplo no hay indicación clara de agrupamiento o outliers. Los números entre paréntesis son la cantidad relativa de la varianza total de **X** explicada por cada componente. Estas varianzas se pueden extraer de la siguiente manera:

```
explvar(gaspcr)
```

```
##    Comp 1    Comp 2    Comp 3    Comp 4
## 71.417135 17.297191  5.535876  3.493021
```

Por tanto, la primera componente explica un 71,42 % de variabilidad de los datos, la segunda un 17,3 % mientras que la tercera y cuarta explican solo un 5.54 % y 3.49 % respectivamente.

Las varianzas acumuladas son

```
cumsum(explvar(gaspcr))
```

```
##   Comp 1   Comp 2   Comp 3   Comp 4
## 71.41713 88.71433 94.25020 97.74322
```

Es decir que con 4 componentes se explica un 97.74 % de variabilidad de los datos.

Cargas de X (Coeficientes de las combinaciones lineales de las variables originales): A

```
dim(gaspcr$loadings)
```

```
## [1] 401   4
```

La representación gráfica de las cargas de **X** se utiliza mucho para propósitos de interpretación, por ejemplo para buscar picos o perfiles espectrales conocidos.

```
plot(gaspcr,"loadings",comps=1:4,legendpos="topleft",
     labels="numbers",xlab="nm")
abline(h=0)
grid()
#otra forma
#loadingplot(gas,comps=1:4,legendpos="topleft",
#            labels="numbers",xlab="nm")
#abline(h=0)
#grid()
```

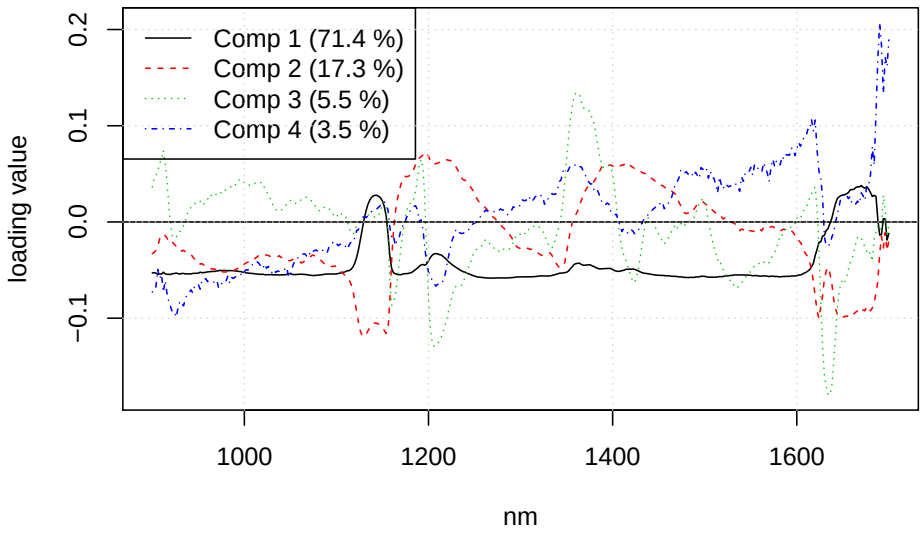
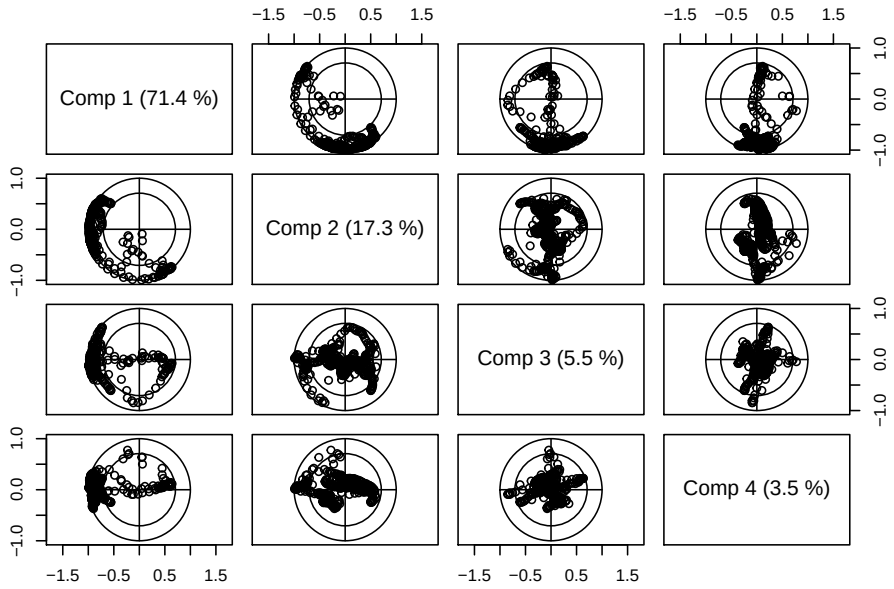


Gráfico de correlaciones

```
plot(gaspcr,plottype="correlation",comps=1:4)
#otra forma
#corrplot(gaspcr,comps=1:4)
```



Este gráfico muestra las correlaciones entre cada variable y las componentes seleccionadas. Se trata de un diagrama de dispersión de puntuaciones con círculos concéntricos de radios dados por *radii*. Cada punto corresponde a una variable de **X**. La distancia al cuadrado entre el punto y el origen es igual a la fracción de la varianza de la variables explicada por las componentes del gráfico. Los valores por defecto de *radii* corresponden al 50 % y al 100 % de la varianza explicada, respectivamente.

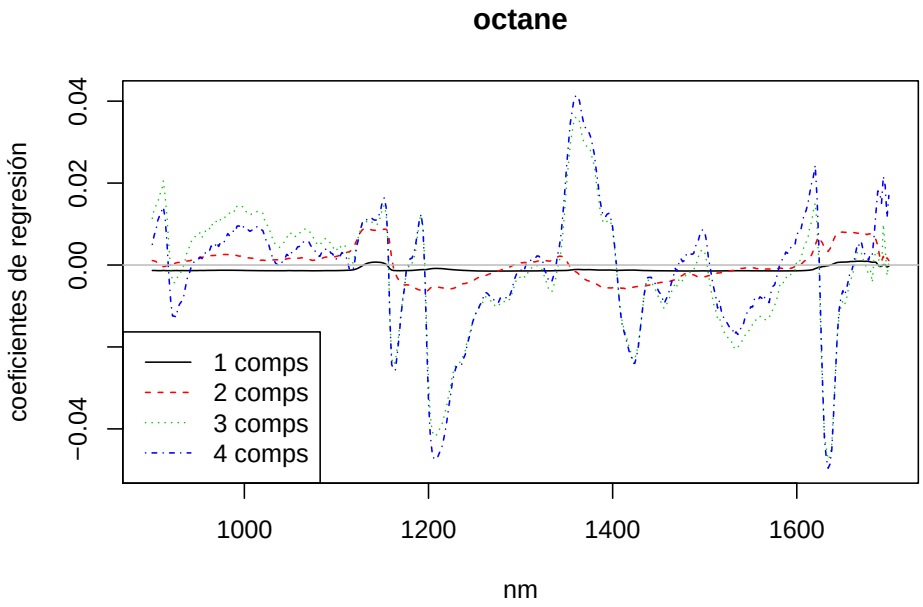
Coeficientes de regresión: B

```
dim(coef(gaspcr))
```

```
## [1] 401   1   1
```

Los coeficientes de regresión para las cuatro primeras componentes se representan de la siguiente manera

```
plot(gaspcr,plottype="coef",ncomp=1:4,legendpos=
       "bottomleft",labels="numbers",ylab=
       "coeficientes de regresión",xlab="nm")
#también se puede representar así:
#coefplot(gaspcr,ncomp=1:4,legendpos="bottomleft",
#         labels = "numbers", ylab="coeficientes
#         de regresiín", xlab = "nm")
```



Los coeficientes para 3 y 4 componentes son similares. Esto es porque la 4 componente contribuye poco a las predicciones. Además este gráfico nos indica qué variables de **X** son más y menos importantes, esto corresponde a valores altos y bajos, respectivamente, de los coeficientes.

Coeficientes de regresión sobre las componentes principales: B^*

```
dim(gaspcr$Yloadings)
```

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

Predicciones

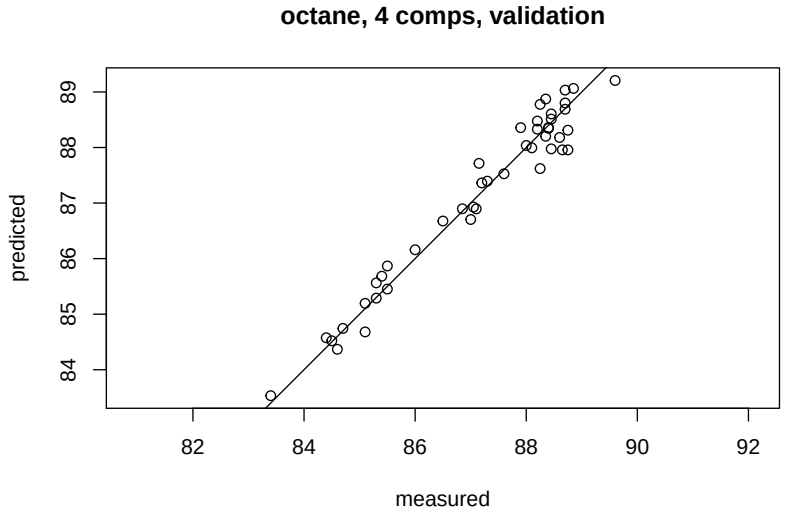
En general el modelo construido sirve para obtener predicciones sobre nuevas observaciones. Considerando estas nuevas observaciones las del conjunto test, se obtienen las siguientes predicciones

```
predict(gaspcr,ncomp=4,newdata=gasTest)
```

```
## , , 4 comps
##
##      octane
## 2  85.06993
## 7  88.84896
## 8  88.42161
## 24 87.42503
## 26 88.60426
## 27 86.48190
## 29 86.49757
## 31 86.58607
## 37 85.51656
## 41 88.87765
## 45 88.63074
## 47 88.54428
## 51 87.96054
## 56 84.82166
## 58 86.93988
```

Dichas predicciones se pueden representar gráficamente

```
plot(gaspcr,ncomp=4,asp=1,line=TRUE)
#otra forma
#predplot(gaspcr,ncomp=4,asp=1,line=TRUE)
```



El ajuste es bastante bueno y el RMSEP sobre el conjunto test es

```
RMSEP(gaspcr,newdata=gasTest)
```

```
## (Intercept)      1 comps      2 comps      3 comps
##      1.3601       1.2596       1.1745       0.3903
##     4 comps
##      0.2553
```

6.3. Ejemplo de PLSR con una variable respuesta

En esta sección se va a construir un modelo PLS sobre el mismo conjunto de datos de la sección anterior.

Determinar el número de componentes PLS

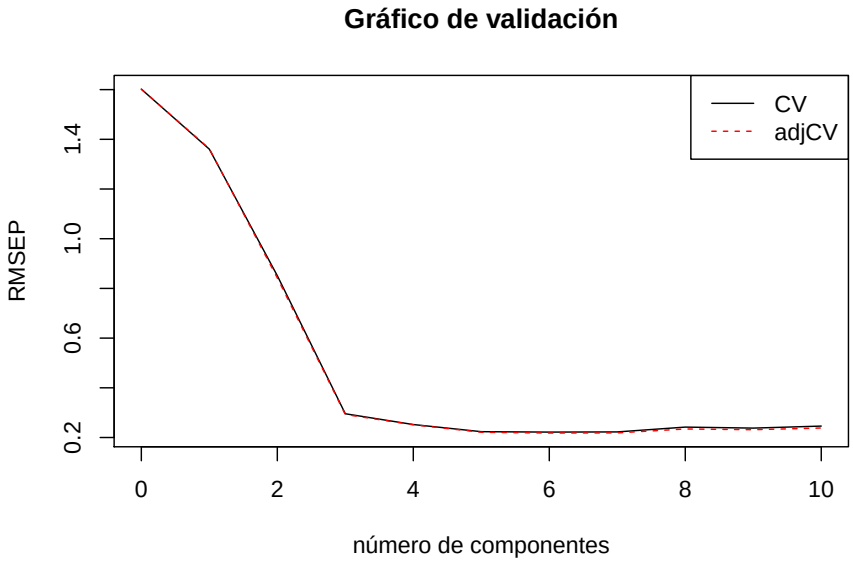
En este caso también se va a ajustar el modelo con 10 componentes como máximo haciendo uso de la validación cruzada ordinaria. El método de regresión utilizado es el Wide Kernel ya que hay muchas más variables (401) que observaciones (60)

```
gas1<-plsr(octane~NIR,ncomp=10,data=gasTrain,scale=TRUE,
       method="widekernelpls",validation="CV")
summary(gas1)
```

```
## Data:    X dimension: 45 401
##  Y dimension: 45 1
## Fit method: widekernelpls
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps   2 comps  3 comps
## CV           1.602    1.360    0.8512   0.2955
## adjCV        1.602    1.362    0.8422   0.2926
##            4 comps  5 comps   6 comps  7 comps
## CV          0.2521   0.2235    0.2217   0.2225
## adjCV       0.2512   0.2212    0.2180   0.2185
##            8 comps  9 comps  10 comps
## CV          0.2418   0.2378    0.2459
## adjCV       0.2343   0.2314    0.2382
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps   5 comps
## X         62.54    83.65    94.17    97.41     98.53
## octane    32.17    79.77    97.38    98.18     98.81
##         6 comps  7 comps  8 comps  9 comps  10 comps
## X         98.86    99.14    99.21    99.39     99.48
## octane    99.10    99.23    99.48    99.53     99.63
```

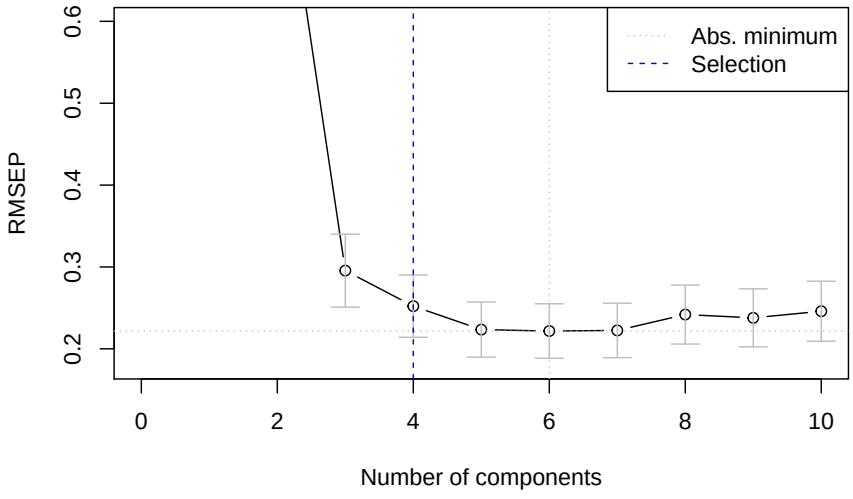
1. Visualizando el siguiente gráfico de validación, 3 componentes parecen ser suficientes ya que el RMSEP no muestra una disminución significativa para más de 3 componentes. En estE caso RMSEP=0.2955.

```
plot(RMSEP(gas1),main="Gráfico de validación",xlab=
     "número de componentes",legendpos="topright")
```



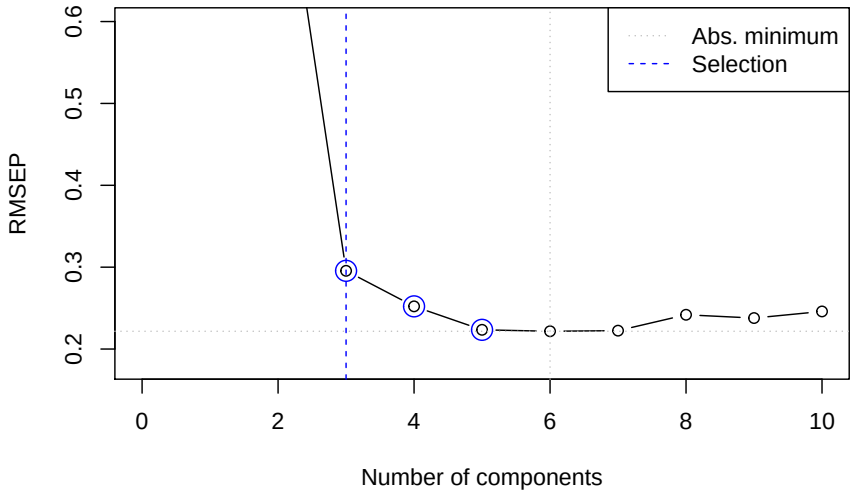
2. Con la función *selectNcomp* y usando la regla de una sigma, se seleccionarían 4 componentes

```
ncomp.onesigma<-selectNcomp(gas1,method="onesigma",
                            plot=TRUE,ylim=c(.18, .6))
```



Y con la prueba de aleatorización, se seleccionarían 3 componentes

```
ncomp.permut<-selectNcomp(gas1,method="randomization",
                          plot=TRUE,ylim=c(.18, .6))
```



Por tanto si seleccionan 3 componentes el modelo ajustado sería:

```
gaspls<-plsr(octane~NIR,ncomp=3,data=gasTrain,
    scale=TRUE, method="widekernelpls" ,validation="CV")
```

La principal diferencia práctica entre PCR y PLSR es que PCR necesita más componentes que PLSR para lograr aproximadamente el mismo error de predicción.

	RMSEP	**Nº Componentes**
PCR	0.2986	4
PLSR	0.2955	3

Valores ajustados y residuos

```
fitted.values(gaspls)
```

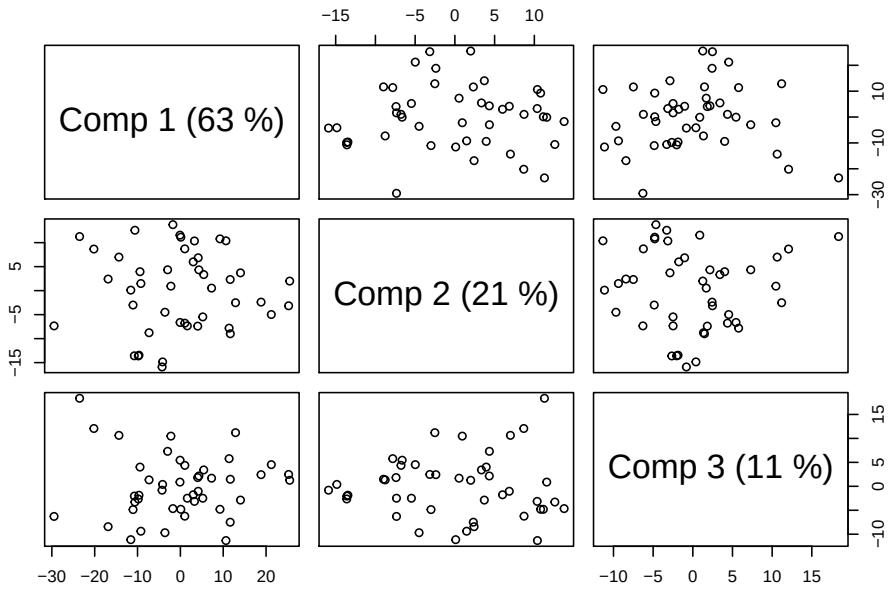
```
residuals(gaspls)
```

Puntuaciones de X (Componentes): T

```
dim(gaspls$scores)
```

```
## [1] 45  3
```

```
plot(gaspls,plottype="scores",comps=1:3)
```



En este ejemplo no hay indicación clara de agrupamiento o outliers. Y la cantidad relativa de la varianza total de **X** explicada por cada componente es

```
explvar(gaspls)
```

```
##    Comp 1    Comp 2    Comp 3
## 62.54337 21.10728 10.51748
```

Por tanto, la primera componente explica un 62.54 % de variabilidad de los datos, la segunda un 21.10 % y la tercera un 10.52 %.

Las varianzas acumuladas son

```
cumsum(explvar(gaspls))
```

```
##    Comp 1    Comp 2    Comp 3
## 62.54337 83.65065 94.16813
```

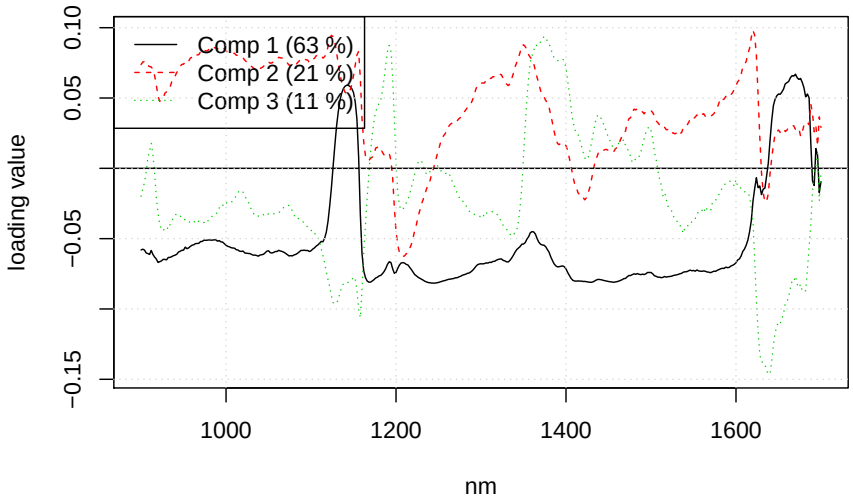
Es decir que con 3 componentes se explica un 94.17 % de variabilidad de los datos.

Cargas de X: P

```
dim(gaspls$loadings)
```

```
## [1] 401   3
```

```
plot(gaspls,"loadings",comps=1:3,legendpos="topleft",
     labels="numbers",xlab="nm")
abline(h=0)
grid()
```



Puntuaciones y cargas de Y: U, Q

En el modelo PLS, como se ha descrito, también hay puntuaciones y cargas para la variable respuesta.

```
dim(gaspls$Yscores)
```

```
## [1] 45  3
```

```
dim(gaspls$Yloadings)
```

```
## [1] 1 3
```

Pesos: W

Los pesos de **X** también se pueden extaer del modelo ajustado

```
dim(gaspls$projection)
```

```
## [1] 401   3
```

Pesos transformados: W^*

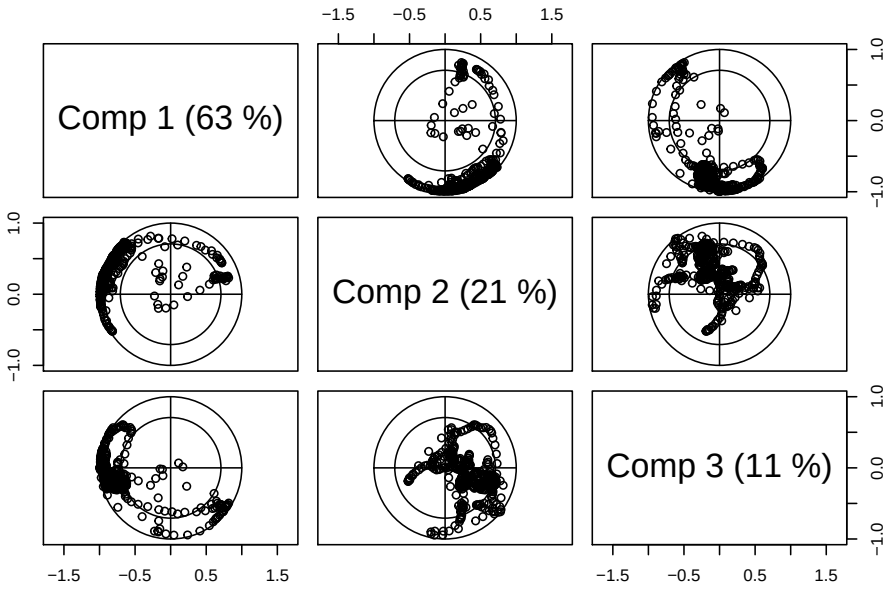
Para los algoritmos Kernel, Wide Kernel y NIPALS, los pesos anteriores se transforman y son los coeficientes de las combinaciones lineales de las variables originales

```
dim(gaspls$loading.weights)
```

```
## [1] 401   3
```

Gráfico de correlaciones

```
plot(gaspls,plottype="correlation",comps=1:3)
```



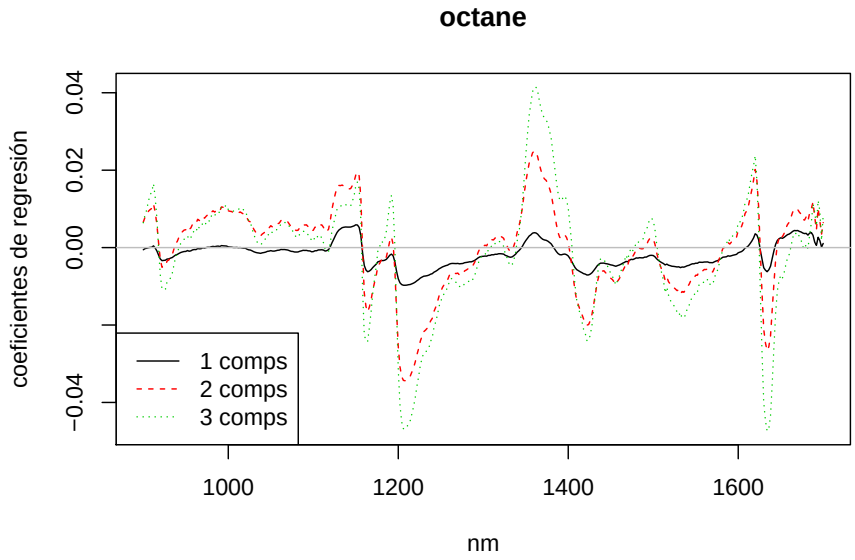
Coeficientes de regresión: B

```
dim(coef(gaspls))
```

```
## [1] 401   1   1
```

El gráfico de los coeficientes de regresión para las tres primeras componentes es

```
plot(gaspls,plottype="coef",ncomp=1:3,legendpos=
       "bottomleft",labels="numbers",ylab=
       "coeficientes de regresión",xlab="nm")
```



Las variables más importantes están entorno a los 1350 nm, mientras que las menos importantes en 1200 nm y 1630 nm.

Predicciones

Las predicciones obtenidas el modelo construido son

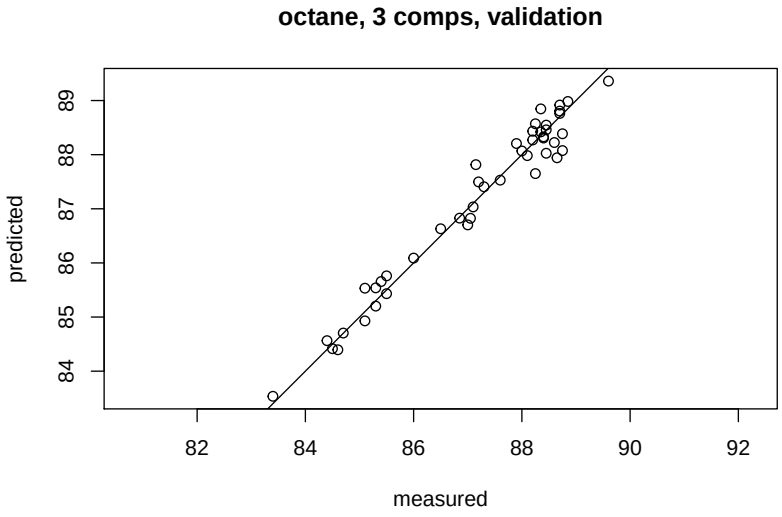
```
predict(gaspls,ncomp=3,newdata=gasTest)
```

```
## , , 3 comps
##
##       octane
## 2   85.10344
## 7   88.84391
## 8   88.44032
## 24  87.35993
## 26  88.56769
## 27  86.49248
## 29  86.45325
## 31  86.51506
## 37  85.45410
## 41  88.85347
## 45  88.59758
```

```
## 47 88.36606
## 51 88.12639
## 56 84.89044
## 58 87.11218
```

Y su representación gráfica

```
plot(gaspls,ncomp=3,asp=1,line=TRUE)
```



El ajuste es bastante bueno y el RMSEP sobre el conjunto test es

```
RMSEP(gaspls,newdata=gasTest)
```

```
## (Intercept)      1 comps      2 comps      3 comps
##      1.3601       1.0826       0.6799       0.2341
```

Los resultados aquí expuestos coinciden para el resto de algoritmos.

6.4. Ejemplo de PLSR con más de una variable respuesta

Otro conjunto de datos que incluye la librería *pls* es **oliveoil**:

```
data(oliveoil)
help(oliveoil)
```

Se trata de 16 muestras de aceite de oliva, las cinco primeras corresponden a aceites griegos, las cinco siguientes son italianos y las seis últimas son españoles. En ellas se han medido 5 parámetros de calidad físico-química: acidez, peróxido, K232, K270 y DK. Además cada muestra tiene puntuaciones de 6 atributos de un panel sensorial: 'yellow', 'green', 'brown', 'glossy', 'transp', and 'syrup'.

- $X_{16\times5}$: matriz de variables predictoras.
- $Y_{16\times6}$: matriz de variables respuestas.

Determinar el número de componentes PLS

Al disponer de pocas observaciones no es recomendable, en este caso, particionar los datos en entrenamiento y test. Por tanto, el modelo se va a ajustar sobre todo el conjunto de datos. Por otro lado, no se va a indica un número máximo de componentes ya que solo hay 5 variables predictoras y de nuevo se hace uso de la validación cruzada ordinaria. Además al disponer casi de tantas observaciones como variables el método de regresión utilizado va a ser el NIPALS:

```
olive1<-plsr(sensory~chemical,data=oliveoil,method=
          "oscorespls",scale=TRUE,validation="CV")
summary(olive1)
```

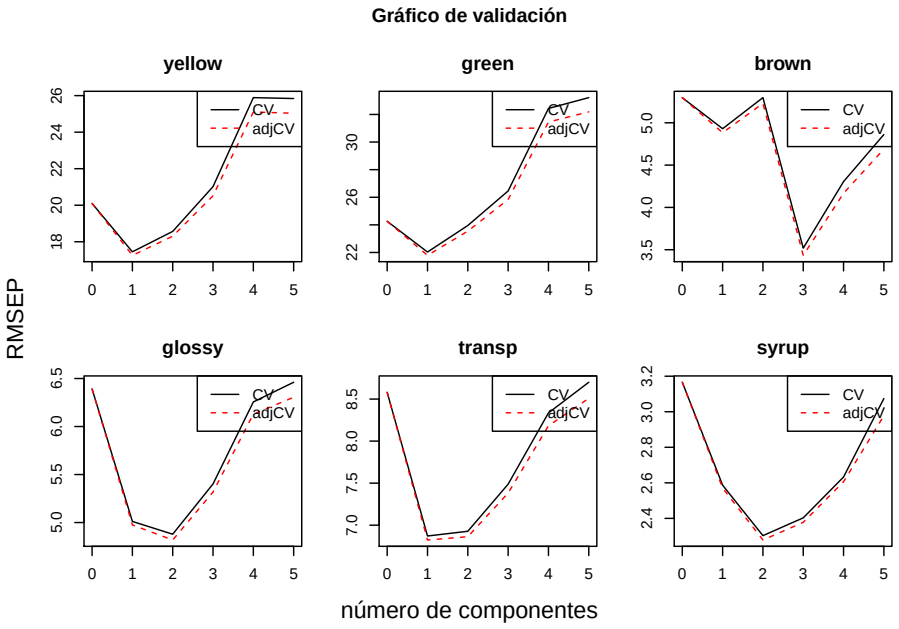
```
## Data:    X dimension: 16 5
##  Y dimension: 16 6
## Fit method: oscorespls
## Number of components considered: 5
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##
## Response: yellow
##        (Intercept)  1 comps  2 comps  3 comps
## CV            20.1    17.45    18.57    21.02
## adjCV         20.1    17.26    18.30    20.52
##            4 comps  5 comps
## CV           25.89    25.84
## adjCV        25.10    25.04
```

```
##
## Response: green
##        (Intercept)  1 comps  2 comps  3 comps
## CV           24.26    22.03    23.95    26.46
## adjCV        24.26    21.79    23.55    25.84
##             4 comps  5 comps
## CV            32.43    33.22
## adjCV         31.45    32.19
##
## Response: brown
##        (Intercept)  1 comps  2 comps  3 comps
## CV           5.297    4.930    5.298    3.519
## adjCV        5.297    4.883    5.229    3.433
##            4 comps  5 comps
## CV           4.305    4.861
## adjCV        4.167    4.689
##
## Response: glossy
##        (Intercept)  1 comps  2 comps  3 comps
## CV           6.391    5.012    4.878    5.402
## adjCV        6.391    4.975    4.818    5.317
##            4 comps  5 comps
## CV           6.256    6.461
## adjCV        6.129    6.305
##
## Response: transp
##        (Intercept)  1 comps  2 comps  3 comps
## CV            8.58    6.871    6.928    7.488
## adjCV         8.58    6.824    6.863    7.382
##            4 comps  5 comps
## CV           8.335    8.700
## adjCV        8.182    8.508
##
## Response: syrup
##        (Intercept)  1 comps  2 comps  3 comps
## CV           3.166    2.587    2.303    2.403
## adjCV        3.166    2.571    2.278    2.378
##            4 comps  5 comps
## CV           2.633    3.074
## adjCV        2.606    2.978
##
```

```
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps
## X         57.79    79.09    95.56    98.25   100.00
## yellow    45.88    50.53    53.15    53.74    54.74
## green     40.33    47.21    47.89    48.52    48.66
## brown     29.80    52.90    77.10    78.39    78.49
## glossy    47.14    52.22    52.27    52.27    53.11
## transp    43.57    44.98    44.98    45.15    45.97
## syrup     40.15    50.35    52.29    55.50    62.39
```

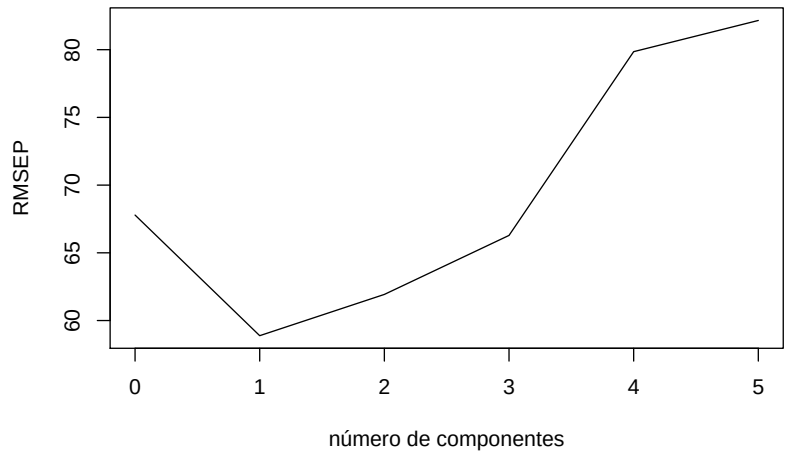
Si se representa el gráfico de validación, se obtiene en este caso uno para cada componente

```
plot(RMSEP(olive1),main="Gráfico de validación",xlab=
       "número de componentes",legendpos="topright")
```



Para visualizarlo en una sola gráfica, se suman los RMSEP para las diferentes componentes

```
VC=RMSEP(olive1)$val[1,,]
plot(0:5,apply(VC,2,sum),type="l")
```



Por tanto, el menor error se produce con una sola componente y el modelo ajustado sería

```
olivepls<-plsr(sensory~chemical,ncomp=1,data=oliveoil,
        method="oscorespls",scale=TRUE,validation="CV")
```

La varianza explicada por esta componente es

```
explvar(olivepls)
```

```
##    Comp 1
## 57.78977
```

Es decir, solo explica un 57.8 % de la variabilidad de los datos. Sin embargo, si se eligen 2

```
explvar(olivepls)
```

```
##    Comp 1    Comp 2
## 57.78977 21.29851
```

```
cumsum(explvar(olivepls))
```

```
##    Comp 1    Comp 2
## 57.78977 79.08828
```

Luego se explicaría un 79.09 % de la variabilidad de los datos. Como el error entre elegir 1 o 2 no difiere mucho (59 frente a 62), entonces se eligirían 2 componentes ya que se explica mucho más la variabilidad de los datos. La primera componente explicaría un 57.8 % de la variabilidad de los datos mientras que la segunda un 21.3 %.

```
olivepls<-plsr(sensory~chemical,ncomp=2,data=oliveoil,
        method="oscorespls",scale=TRUE,validation="CV")
```

Valores ajustados y residuos

```
fitted.values(olivepls)
residuals(olivepls)
```

Puntuaciones de X (Componentes): T

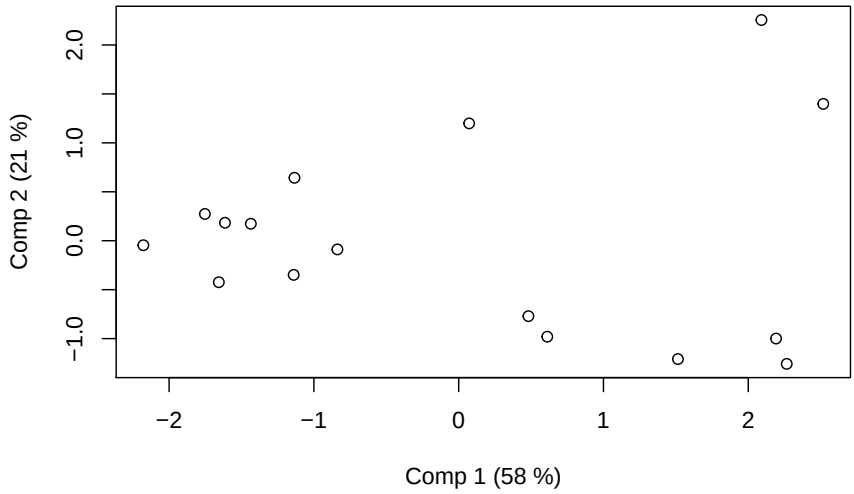
```
olivepls$scores
```

```
##          Comp 1      Comp 2
## G1   2.51832234  1.39820454
## G2  -0.83764624 -0.08876161
## G3  -1.13410737  0.64280200
## G4   2.09116266  2.25494470
## G5   0.07232963  1.19868808
## I1   2.26589702 -1.25809411
## I2   0.61188989 -0.98056994
## I3   1.51443626 -1.20906408
## I4   0.48177431 -0.77059026
## I5   2.19222202 -0.99933742
## S1  -1.43557766  0.17344539
## S2  -1.61406246  0.18336064
## S3  -1.13954217 -0.34879220
## S4  -1.65648557 -0.42393119
## S5  -2.17781235 -0.04589859
## S6  -1.75280031  0.27359406
## attr(,"class")
## [1] "scores"
```

```
dim(olivepls$scores)
```

```
## [1] 16  2
```

```
plot(olivepls,plottype="scores",comps=1:2)
```



No hay indicación cada de agrupamiento o outliers.

Cargas de X: P

```
olivepls$loadings
```

```
##
## Loadings:
##           Comp 1 Comp 2
## Acidity    0.337  0.715
## Peroxide   0.462 -0.556
## K232       0.515 -0.418
## K270       0.513  0.247
## DK         0.418 -0.116
##
##                Comp 1 Comp 2
## SS loadings     1.030  1.069
```

```
## Proportion Var  0.206  0.214
## Cumulative Var  0.206  0.420
```

```
dim(olivepls$loadings)
```

```
## [1] 5 2
```

Puntuaciones y cargas de Y: U, Q

Las puntuaciones serían

```
olivepls$Yscores
```

```
##         Comp 1     Comp 2
## G1   3.6837048  2.6972408
## G2   3.3653654  5.6340333
## G3   1.7952495  4.6294661
## G4   2.3739904  1.2280706
## G5   0.2356869 -0.4252670
## I1   1.6176090 -1.9411819
## I2  -0.2754720 -1.1909931
## I3   3.1441515  2.6837511
## I4  -1.9500934 -3.4045140
## I5   0.5821973 -3.1728104
## S1  -2.4847601 -1.2165263
## S2  -2.8555805 -1.2135996
## S3  -1.9395967 -1.8194597
## S4  -2.0570029 -0.8221403
## S5  -2.5952325 -0.5591660
## S6  -2.6402166 -1.1069038
## attr(,"class")
## [1] "scores"
```

```
dim(olivepls$Yscores)
```

```
## [1] 16  2
```

Y las cargas serían

```
olivepls$Yloadings
```

```
##
## Loadings:
##        Comp 1 Comp 2
## yellow -7.870 -4.204
## green   8.906  6.174
## brown   1.672 -2.470
## glossy -2.537  1.398
## transp -3.274  0.987
## syrup   1.160 -0.981
##
##                 Comp 1 Comp 2
## SS loadings    162.555 65.786
## Proportion Var  27.093 10.964
## Cumulative Var  27.093 38.057
```

```
dim(olivepls$Yloadings)
```

```
## [1] 6 2
```

Pesos: W

```
olivepls$projection
```

```
##             Comp 1     Comp 2
## Acidity  0.4222728  0.6764360
## Peroxide 0.3735538 -0.5249847
## K232     0.4803406 -0.2119901
## K270     0.5873493  0.4375073
## DK       0.3262467 -0.2409384
```

```
dim(olivepls$projection)
```

```
## [1] 5 2
```

Pesos transformados: W^*

```
olivepls$loading.weights
```

```
##
## Loadings:
##           Comp 1 Comp 2
## Acidity    0.422  0.606
## Peroxide   0.374 -0.587
## K232       0.480 -0.292
## K270       0.587  0.340
## DK         0.326 -0.295
##
##                 Comp 1 Comp 2
## SS loadings        1.0    1.0
## Proportion Var     0.2    0.2
## Cumulative Var     0.2    0.4
```

```
dim(olivepls$loading.weights)
```

```
## [1] 5 2
```

Coeficientes de regresión: B

```
coef(olivepls)
```

```
## , , 2 comps
##
##               yellow      green       brown     glossy
## Acidity  -6.1670268 7.93725612 -0.96496512 -0.1258532
## Peroxide -0.7329325 0.08554506  1.92130813 -1.6814244
## K232     -2.8891619 2.96909032  1.32668118 -1.5148771
## K270     -6.4617672 7.93224946 -0.09879616 -0.8785791
## DK       -1.5547249 1.41798282  1.14057629 -1.1644121
##               transp      syrup
## Acidity  -0.7150432 -0.1740505
## Peroxide -1.7413211  0.9484015
## K232     -1.7820567  0.7651071
## K270     -1.4913906  0.2518598
## DK       -1.3060655  0.6148018
```

6.4.1. Comparación de resultados

En esta sección se muestra que los resultados con el algoritmo SIMPLS son ligeramente diferentes para modelos con más de una variable respuesta. De hecho esta algoritmo es más rápido que el NIPALS.

Determinar el número de componentes PLS

Se procede igual que antes solo que con el algoritmo SIMPLS

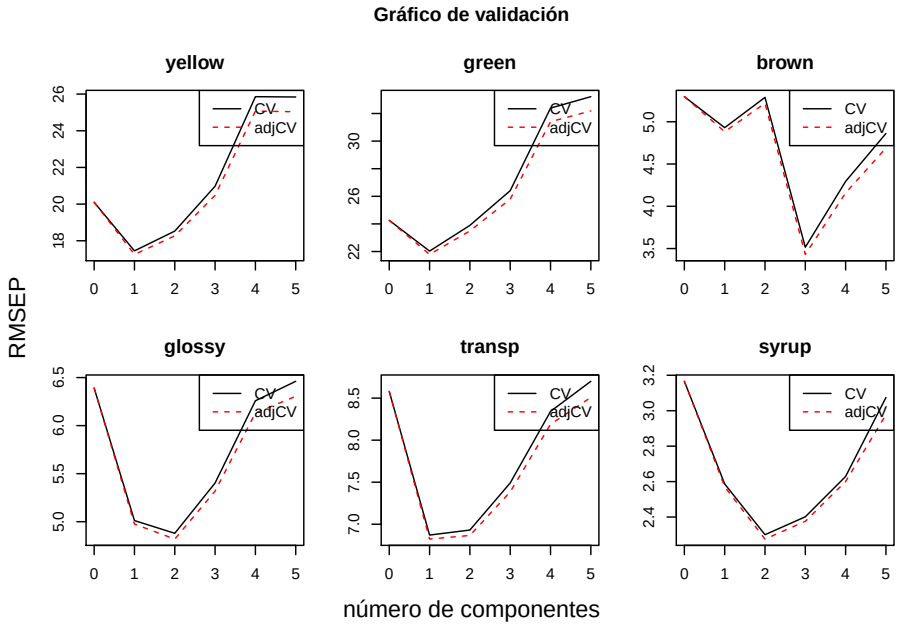
```
olive1<-plsr(sensory~chemical,data=oliveoil,method=
          "simpls",scale=TRUE,validation="CV")
summary(olive1)
```

```
## Data:    X dimension: 16 5
##  Y dimension: 16 6
## Fit method: simpls
## Number of components considered: 5
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##
## Response: yellow
##        (Intercept)  1 comps  2 comps  3 comps
## CV            20.1    17.45    18.53    20.97
## adjCV         20.1    17.26    18.27    20.48
##            4 comps  5 comps
## CV           25.86    25.84
## adjCV        25.07    25.04
##
## Response: green
##        (Intercept)  1 comps  2 comps  3 comps
## CV            24.26    22.03    23.90    26.41
## adjCV         24.26    21.79    23.49    25.79
##            4 comps  5 comps
## CV           32.39    33.22
## adjCV        31.41    32.19
##
## Response: brown
##        (Intercept)  1 comps  2 comps  3 comps
## CV            5.297    4.930    5.288    3.515
```

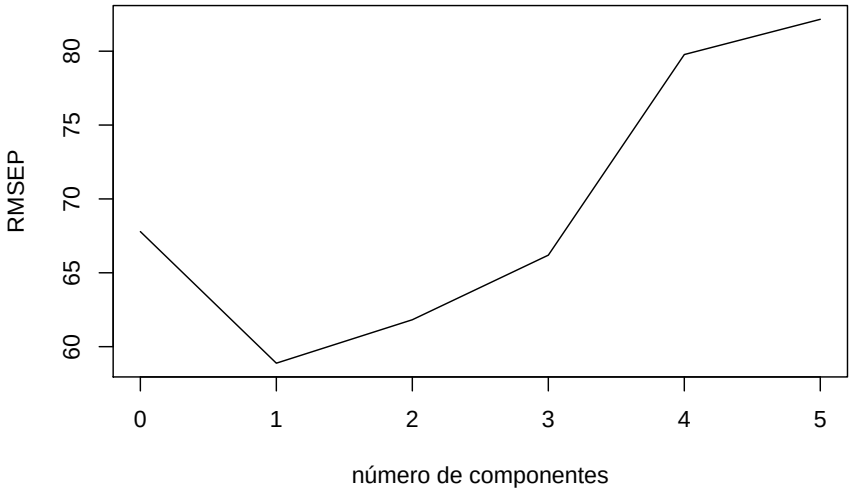
```
## adjCV        5.297    4.883    5.219    3.430
##            4 comps  5 comps
## CV           4.294    4.861
## adjCV        4.158    4.689
##
## Response: glossy
##        (Intercept)  1 comps  2 comps  3 comps
## CV           6.391    5.012    4.879    5.403
## adjCV        6.391    4.975    4.819    5.318
##            4 comps  5 comps
## CV           6.259    6.461
## adjCV        6.131    6.305
##
## Response: transp
##        (Intercept)  1 comps  2 comps  3 comps
## CV            8.58    6.871    6.931    7.491
## adjCV         8.58    6.824    6.867    7.385
##            4 comps  5 comps
## CV           8.342    8.700
## adjCV        8.188    8.508
##
## Response: syrup
##        (Intercept)  1 comps  2 comps  3 comps
## CV           3.166    2.587    2.300    2.401
## adjCV        3.166    2.571    2.276    2.376
##            4 comps  5 comps
## CV           2.629    3.074
## adjCV        2.601    2.978
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps
## X         57.79    79.17    95.57    98.28   100.00
## yellow    45.88    50.46    53.14    53.72    54.74
## green     40.33    47.17    47.89    48.52    48.66
## brown     29.80    53.35    77.19    78.40    78.49
## glossy    47.14    52.23    52.27    52.27    53.11
## transp    43.57    44.98    44.98    45.16    45.97
## syrup     40.15    50.45    52.35    55.66    62.39
```

Se puede observar que la varianza explicada en el conjunto de entrenamiento es distinta a la obtenida con el algoritmo NIPALS para más de una componente.

```
plot(RMSEP(olive1),main="Gráfico de validación",xlab=
       "número de componentes",legendpos="topright")
```



Visto en un único gráfico



Por el mismo razonamiento anterior, se seleccionan dos variables. Por tanto, el modelo ajustado para 2 componentes es

```
olivepls<-plsr(sensory~chemical,ncomp=2,data=oliveoil,
        method="simpls",scale=TRUE,validation="CV")
```

Valores ajustados y residuos

```
fitted.values(olivepls)
residuals(olivepls)
```

Puntuaciones de X (Componentes): T

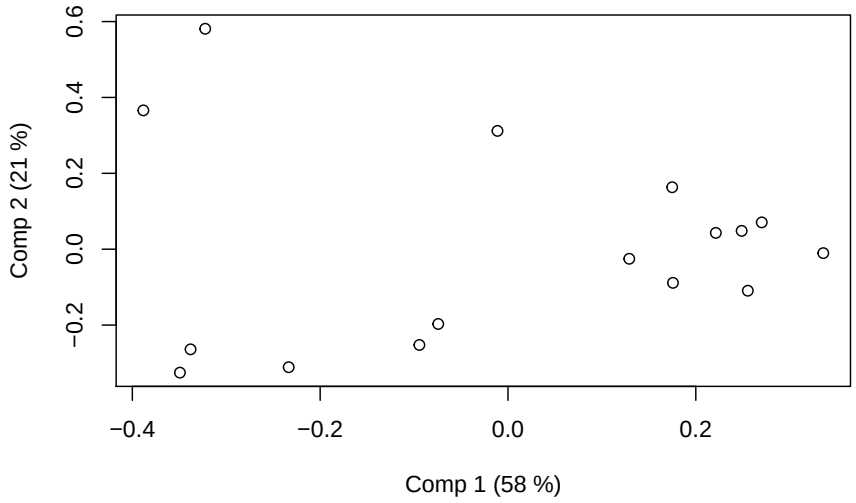
```
olivepls$scores
```

```
##          Comp 1      Comp 2
## G1 -0.38825335  0.36586876
## G2  0.12914112 -0.02531117
## G3  0.17484695  0.16326589
## G4 -0.32239753  0.58101921
## G5 -0.01115116  0.31180087
## I1 -0.34933658 -0.32537977
## I2 -0.09433594 -0.25245197
## I3 -0.23348280 -0.31110352
## I4 -0.07427583 -0.19707984
## I5 -0.33797800 -0.26396856
## S1  0.22132506  0.04294562
## S2  0.24884232  0.04826719
## S3  0.17568484 -0.08873074
## S4  0.25538275 -0.10946286
## S5  0.33575644 -0.01035290
## S6  0.27023173  0.07067379
## attr(,"class")
## [1] "scores"
```

```
dim(olivepls$scores)
```

```
## [1] 16  2
```

```
plot(olivepls,plottype="scores",comps=1:2)
```



La varianza explicada por cada componente es

```
explvar(olivepls)
```

```
##    Comp 1    Comp 2
## 57.78977 21.38046
```

Se puede observar que el tanto por ciento de varibilidad explicada por la primera componente coincide con la de NIPALS. La segunda difiere un poco, es decir, la segunda componente explica un 21.38 % de la variabilidad de los datos.

```
cumsum(explvar(olivepls))
```

```
##    Comp 1    Comp 2
## 57.78977 79.17024
```

Luego con este algortimo se explicaría un 79.17 % de la variabilidad de los datos, un poco más que para el NIPALS.

Cargas de X: P

```
olivepls$loadings
```

```
##
## Loadings:
##          Comp 1 Comp 2
## Acidity  -2.186  2.775
## Peroxide -2.999 -2.154
## K232     -3.339 -1.624
## K270     -3.329  0.939
## DK       -2.709 -0.425
##
##                Comp 1 Comp 2
## SS loadings    43.342 16.035
## Proportion Var  8.668  3.207
## Cumulative Var  8.668 11.876
```

```
dim(olivepls$loadings)
```

```
## [1] 5 2
```

Puntuaciones y cargas de Y: U, Q

Las puntuaciones son

```
olivepls$Yscores
```

```
##        Comp 1    Comp 2
## G1 -3884.0259  684.1275
## G2 -3548.3751 1425.8315
## G3 -1892.8757 1172.0876
## G4 -2503.0888  311.7526
## G5  -248.5037 -107.4485
## I1 -1705.5751 -492.3803
## I2   290.4523 -301.1215
## I3 -3315.1315  679.6246
## I4  2056.1402 -861.4519
## I5  -613.8574 -805.1457
## S1  2619.8822 -307.7590
## S2  3010.8680 -306.5677
## S3  2045.0726 -460.9465
```

```
## S4  2168.8634 -208.5412
## S5  2736.3621 -141.7510
## S6  2783.7925 -280.3105
## attr(,"class")
## [1] "scores"
```

```
dim(olivepls$Yscores)
```

```
## [1] 16  2
```

Y las cargas son

```
olivepls$Yloadings
```

```
##
## Loadings:
##        Comp 1  Comp 2
## yellow  51.048 -16.133
## green  -57.768  23.794
## brown  -10.844  -9.640
## glossy  16.455   5.407
## transp  21.239   3.815
## syrup   -7.523  -3.810
##
##                  Comp 1   Comp 2
## SS loadings    6839.013  977.679
## Proportion Var 1139.835  162.947
## Cumulative Var 1139.835 1302.782
```

```
dim(olivepls$Yloadings)
```

```
## [1] 6 2
```

Pesos

El algoritmo SIMPLS conduce al cálculo directo de $\mathbf{W}^*$. Por tanto, los pesos son

```
olivepls$projection
```

```
##                Comp 1      Comp 2
## Acidity  -0.06510241  0.17534403
## Peroxide -0.05759132 -0.13597887
## K232     -0.07405479 -0.05550221
## K270     -0.09055248  0.11186096
## DK       -0.05029793 -0.05999148
```

```
dim(olivepls$projection)
```

```
## [1] 5 2
```

Y en este caso no hay pesos transformados

```
olivepls$loading.weights
```

```
## NULL
```

Coeficientes de regresión: B

```
coef(olivepls)
```

```
## , , 2 comps
##
##                yellow      green       brown     glossy
## Acidity  -6.1522482 7.93296610 -0.98432154 -0.1231696
## Peroxide -0.7461179 0.09140489  1.93530561 -1.6829321
## K232     -2.8849093 2.95733221  1.33805560 -1.5186956
## K270     -6.4272245 7.89262727 -0.09638751 -0.8852168
## DK       -1.5997412 1.47813910  1.12371870 -1.1520449
##                transp      syrup
## Acidity  -0.7137795 -0.1783105
## Peroxide -1.7418962  0.9513025
## K232     -1.7845534  0.7685401
## K270     -1.4964808  0.2550045
## DK       -1.2971160  0.6069316
```

```
dim(coef(olivepls))
```

```
## [1] 5 6 1
```

Bibliografía

[1] Svante Wold, Michael Sjöström, Lennart Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, *Vol*. 58, 109-130, 2001.

[2] Carlos Gaviria Peña, *Regresión por Mínimos Cuadrados Parciales PLS Aplicada a Datos Variedad Valuados*. Tesis 2016.

[3] NCSS Statistical Software. P*rincipal Components Regression*. Capter 340.

[4] Agnar Höskuldsson, PLS regression methods. *Journal of Chemometrics*, *Vol*. 2, 211-228, 1988.

[5] Kurt Varmuza, Peter Filzmoser. *Introduction to Multivariate Statistical Analysis in Chemometrics*. 2009

[6] Kevin Dunn. *Process Improvement Using Data*. 2016

[7] Rolf Manne, Analysis of two partial least squares algorithms for multivariate regression. *Chemometrics and Intelligen Laboratory Systems*, *Vol*. 2, 187-197, 1987.

[8] C.R. Rao and S.K. Mitra. *Generalized Inverse of Matrices and its Applications*. Wiley, New York, 1971

[9] B. S. Dayal, J. F. MacGregor. Improved pls algorithms. *Journal of Chemometrics*, *Vol*. 11, 73-85, 1997.

[10] Sijmen de Jong. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, *Vol* 18, 251-263, 1993.

[11] Rännar, S., Lindgren, F., Geladi, P. and Wold, S. (1994) A PLS Kernel Algorithm for Data Sets with Many Variables and Fewer Objects. Part 1: Theory and Algorithm. *Journal of Chemometrics*, 8, 111–125.

[12] Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Springer Second Edition.

[13] Hilko van der Voet. Comparing the predictive accurary of models using a simple randomization test. *Chemometrics and Intelligent Laboratory*, Vol 25, 313-323, 1994.

[14] Bjorn-Helge Mevik, Ron Wehrens, *Introduction to the pls Package*. Cran R (2016).