Regresión múltiple vía PLS1

Andrés Felipe Palomino - David Stiven Rojas

2023-07-12

1 Introducción

A continuación se presenta el desarrollo del código para realizar un ajuste de regresión múltiple vía PLS, con una aplicación a una base de datos que cuenta con multicolinealidad perfecta y que se contamina con datos faltantes.

1.1 Base de datos sin NA´S

En los datos de Cornell (1990), se registra el índice de octano de motor y en n=12 mezclas, para determinar la influencia de p=7 componentes, x1= destilación directa,.., x7 = esencia natural.

```
library(car)
library(xtable)

YX <- read.table("Cornell.txt",header=TRUE)
xtable(YX)
```

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Wed Jul 12 20:50:25 2023

	y	x1	x2	x3	x4	x5	x6	x7
1	98.70	0.00	0.23	0.00	0.00	0.00	0.74	0.03
2	97.80	0.00	0.10	0.00	0.00	0.12	0.74	0.04
3	96.60	0.00	0.00	0.00	0.10	0.12	0.74	0.04
4	92.00	0.00	0.49	0.00	0.00	0.12	0.37	0.02
5	86.60	0.00	0.00	0.00	0.62	0.12	0.18	0.08
6	91.20	0.00	0.62	0.00	0.00	0.00	0.37	0.01
7	81.90	0.17	0.27	0.10	0.38	0.00	0.00	0.08
8	83.10	0.17	0.19	0.10	0.38	0.02	0.06	0.08
9	82.40	0.17	0.21	0.10	0.38	0.00	0.06	0.08
10	83.20	0.17	0.15	0.10	0.38	0.02	0.10	0.08
11	81.40	0.21	0.36	0.12	0.25	0.00	0.00	0.06
12	88.10	0.00	0.00	0.00	0.55	0.00	0.37	0.08

2 Base de datos con 11% NA´s

Para contaminar la base de datos con el 11% de NA´s se realizará la función fmd() del “Script PLS1 tenenhaus”.

En primer lugar, la función tiene dos componentes de entrada, la base de datos y el porcentaje de NA´s que se colocaran (a), seguidamente la base de datos se transforma en una matriz y se cuenta el total de datos que hay (N), por último se realiza un sample, que extrae una muestra aleatoria de tamaño a*N de N datos, y por ultimo las posiciones que se hayan obtenido en la muestra aleatoria se reemplaza por NA en la matriz.

```
 fmd <- function(Xo,a)  # Genera a: % NAs , md: miss data
  {
    X. <- as.matrix(Xo)
    n <- nrow(X.); p <- ncol(X.); N <- n*p
    m <- sample(N,round(a*N,0))

    X.[m] <- NA

    return(X.)
  }

set.seed(1379)

YXna <- fmd(YX,0.11)
xtable(YXna)
```

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Wed Jul 12 20:50:25 2023

	y	x1	x2	x3	x4	x5	x6	x7
1		0.00		0.00	0.00	0.00	0.74	
2	97.80	0.00	0.10	0.00	0.00	0.12	0.74	0.04
3	96.60	0.00	0.00	0.00	0.10	0.12	0.74	0.04
4	92.00	0.00	0.49	0.00	0.00	0.12	0.37	0.02
5	86.60	0.00	0.00	0.00	0.62	0.12	0.18	0.08
6	91.20	0.00			0.00	0.00	0.37	0.01
7	81.90		0.27		0.38	0.00	0.00	0.08
8	83.10	0.17	0.19		0.38	0.02	0.06	0.08
9	82.40	0.17	0.21	0.10		0.00	0.06	0.08
10			0.15	0.10	0.38	0.02	0.10	0.08
11	81.40	0.21	0.36	0.12	0.25	0.00	0.00	0.06
12	88.10	0.00	0.00	0.00	0.55	0.00	0.37	0.08

3 componentes t_h no centradas y no ortogonales.

En primer lugar, se corre la función fPLS1na del “Script PLS1 tenenhaus”

```
fPLS1na <- function(YX)
    {
      Z <- as.matrix(YX)
        Xo <- scale(YX[,-1])    # matriz n.p
       p <- ncol(Xo); n <- nrow(Xo)

       if(any(!is.finite(Z)))H <- p
        else H <- qr(Xo)$rank

        Yo <- scale(YX[,1]); yz <- Yo
```

```
    cv2 <- matrix(0,1,H)

    W <- matrix(0,p,H); W. <- matrix(0,p,H)
    T <- matrix(0,n,H); C <- matrix(0,1,H)
    P <- matrix(0,p,H); B <- matrix(0,p,H)

    for(h in 1:H)
    {
       for(j in 1:p)
       {
       wh. <- na.omit(cbind(Xo[,j],Yo))
       W.[j,h] <- sum(wh.[,1]*wh.[,2])/sum(wh.[,2]^2)
       }

       nW. <- sqrt(sum(W.[,h]^2))
       W[,h] <- W.[,h]/nW.

       for(i in 1:n)
       {
       ti <- na.omit(cbind(Xo[i,],W[,h]))
       T[i,h] <- sum(ti[,1]*ti[,2])/sum(ti[,2]^2)
       }

       th <- T[,h] ; cv2[1,h] <- cov(th,yz)^2

       ch. <- na.omit(cbind(th,Yo))
       ch <- sum(ch.[,1]*ch.[,2])/sum(ch.[,1]^2)
       C[,h] <- ch

       for(j in 1:p)
       {
       ph. <- na.omit(cbind(Xo[,j],th))
       P[j,h] <- sum(ph.[,1]*ph.[,2])/sum(ph.[,2]^2)
       }
       ph <- P[,h]

       X1 <- Xo - th%*%t(ph); Xo <- X1
       Y1 <- Yo - th%*%t(ch); Yo <- Y1

      B[,h] <- W[,1:h]%*%(solve(t(P[,1:h])%*%W[,1:h]))%*%C[1:h]

    } # end h

    r.PLS1na <- list(W,T,C,P,B,cv2)
    return(r.PLS1na)

  } # end fPLS1na con y sin datos completos
```

Dicha función tiene de salida una lista con 6 resultados, en donde la segunda entrada de la lista (T) corresponde a los componentes t_h de máxima covarianza con y. Para evaluar que las componentes no son ortogonales se procede a calcular la correlación entre ellas, esto porque la ortogonalidad se refiere a la independencia lineal entre los componentes principales, lo cual se ve reflejado en la matriz de covarianzas, la cual debería de ser diagonal. También se realiza el cálculo de media de los componentes para evaluar la

centralidad de los mismos.

```
fpls1 <- fPLS1na(YXna)

T <- fpls1[[2]] # Componentes
labs <- paste(c("T"),1:7,sep=""); colnames(T) <- labs

xtable(cor(T)) # Correlacion para evaluar ortogonalidad
```

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Wed Jul 12 20:50:25 2023

	T1	T2	T3	T4	T5	T6	T7
T1	1.00	0.00	-0.01	0.10	0.14	0.34	-0.02
T2	0.00	1.00	-0.02	-0.05	-0.03	-0.07	0.04
T3	-0.01	-0.02	1.00	0.30	0.30	-0.07	-0.74
T4	0.10	-0.05	0.30	1.00	0.05	-0.04	0.19
T5	0.14	-0.03	0.30	0.05	1.00	0.01	-0.05
T6	0.34	-0.07	-0.07	-0.04	0.01	1.00	0.27
T7	-0.02	0.04	-0.74	0.19	-0.05	0.27	1.00

```
Centra <- c(mean(T[,1]),mean(T[,2]),mean(T[,3]),mean(T[,4]),mean(T[,5])
            ,mean(T[,6]),mean(T[,7])) #media para evaluar centralidad
Centra
```

[1] -0.08861845 0.10310036 -0.01208457 0.06879887 -0.01217265 0.11639402 [7] -0.02345694

Como se evidencia, la matriz de covarianzas presenta correlaciones entre los componentes, por ende estos no son ortogonales, además la media de los mismos no es cero y por ende no están centrados.

4 Función fPLS1 por partes con las respectivas propiedades

En primer lugar, se especificará partes del código de la Función fPLS1 que tienen como objetivo garantizar las propiedades descritas.

4.1 Base de Datos con NAs < 30%

Para verificar la propiedad de que la función fPLS1 funcione con NAs < 30% tanto por fila como por columna, se realizaron contadores, los contadores representan el porcentaje de NAs, tanto por filas como por columnas. Estos contadores son los siguientes:

contary: realiza el cálculo del porcentaje de NAs que hay en la variable de respuesta Y

contarcolumna: es un vector de unos y ceros, los unos se dan cuando el cálculo de porcentaje de NAs por columna es mayor al 30%, y ceros en el caso contrario, por último se realiza la suma de este vector.

contarcolumna: es un vector de unos y ceros, los unos se dan cuando el cálculo de porcentaje de NAs por fila es mayor al 30%, y ceros en el caso contrario, por último se realiza la suma de este vector.

Entonces, si no se cumple la condición de que contarfila <1 & contarcolumna<1 & contary> 0 & contary<0.3 o la condición de que contarfila <1 & contarcolumna<1 & contary==0, entonces la función fPLS1 imprime "Warning Demasiados NA'S por fila o columna o en Y".

En el siguiente código se evidencia el procedimiento a realizar, y al final se implementará a la función fPLS1na.

```
#YX Base de datos
 Z <- as.matrix(YX) # convertida a matriz
Xo <- scale(YX[,-1])    # matriz de variables predictoras
p <- ncol(Xo); n <- nrow(Xo)
Yo <- scale(YX[,1]) # vector de la variable de respuesta
contary<- sum(is.na(Yo))/length(Yo) #Contar porcentaje Na's y
ind<- which(is.na(Yo))
contarcolumna<-c(); contarfila<- c()
for(i in 1:dim(Xo)[2]){
  if(sum(is.na(Xo[,i]))/dim(Xo)[1]>=0.3){  #Contar porcentaje Na's por cada columna
      contarcolumna[i]<- 1
    }
  else{
      contarcolumna[i]<-0
    }
  }
for(i in 1:dim(Xo)[1]){
  if(sum(is.na(Xo[i,]))/dim(Xo)[2]>=0.3){   #Contar porcentaje Na's por cada fila
      contarfila[i]<- 1
    }
  else{
      contarfila[i]<-0
    }
  }
contarfila<- sum(contarfila) #Contador NA's filas
contarcolumna<- sum(contarcolumna)#Contador NA's filas
```

4.2 Número de componentes

Para determinar el número de componentes significativas en la regresión se tendrá en cuenta el R^2_{Adj}, para ello se realiza la diferencia en los R^2_{Adj} agregando cada componente, y cuando se evidencie la condición de un aumento no mayor al 5%, se toman los anteriores componentes al que cumplió con la condición.

```
  RAdj<- c() # R2 ajustado para cada componente agregada
  diff<- c() # Diferencia de los R^2 ajustados con i componentes e i-1 componentes.
  for(i in 1:ncol(T)){
    model<- lm(scale(Yo)~T[,1:ncol(T)])
    resumen <- summary(model)
    RAdj[i] <- resumen$adj.r.squared #calculo de los R^2 ajustados agregando I componentes

  }
  for(i in 2:ncol(T)){
    diff[i]<- RAdj[i]-RAdj[i-1]

  }
  ncomp<- (which(diff < 0.05)) #Numero de componentes a utilizar -1
```

4.3 Funcion del algoritmo con y sin datos faltantes

La función fPLS1na se separa en sección con condicionales, los condicionales están dados por los contadores mencionados anteriormente. si (contarfila <1 & contarcolumna<1 & contary==0) se corre el PLS teniendo en

cuenta que no hay datos faltantes en "y", si (contarfila <1 & contarcolumna<1 & contary> 0 & contary<0.3) se corre el PLS teniendo en cuenta que hay datos faltantes en "y" < 30%, por último en caso de no cumplir con las respectivas condiciones se está asumiendo que existe un NAs mayor al 30% ya sea en la matriz de covariables X o Y.

4.4 Función fPLS1 completa

```
fPLS1na <- function(YX){
  Z <- as.matrix(YX) # Base de datos convertida en Matriz
  Xo <- scale(YX[,-1])  # matriz de variables predictoras
  p <- ncol(Xo); n <- nrow(Xo) # numero de individuos y variables
  Yo <- scale(YX[,1]); yz <- Yo #variables y escalonada

##### Seccion porcentaje de datos faltantes
  contary<- sum(is.na(Yo))/length(Yo) #Contar Na's y
  ind<- which(is.na(Yo))
  contarcolumna<-c();  contarfila<- c()
  for(i in 1:dim(Xo)[2]){
    if(sum(is.na(Xo[,i]))/dim(Xo)[1]>=0.3){ #Contar porcentaje Na's por cada columna
      contarcolumna[i]<- 1
    }
    else{
      contarcolumna[i]<-0
    }
  }
  for(i in 1:dim(Xo)[1]){
    if(sum(is.na(Xo[i,]))/dim(Xo)[2]>=0.3){ #Contar porcentaje Na's por cada fila
      contarfila[i]<- 1
    }
    else{
      contarfila[i]<-0
    }
  }
  contarfila<- sum(contarfila) #Contador NA's filas
  contarcolumna<- sum(contarcolumna)#Contador NA's filas

##### Seccion calculos PLS cuando no hay datos faltantes en y

  if(contarfila <1 & contarcolumna<1 & contary==0){
  if(any(!is.finite(Z)))H <- p
  else H <- qr(Xo)$rank
  cv2 <- matrix(0,1,H)

  W <- matrix(0,p,H); W. <- matrix(0,p,H)
  T <- matrix(0,n,H); C <- matrix(0,1,H)
  P <- matrix(0,p,H); B <- matrix(0,p,H)

  for(h in 1:H)
  {
    for(j in 1:p)
    {
      wh. <- na.omit(cbind(Xo[,j],Yo))
```

```
      W.[j,h] <- sum(wh.[,1]*wh.[,2])/sum(wh.[,2]^2)
    }

    nW. <- sqrt(sum(W.[,h]^2))
    W[,h] <- W.[,h]/nW.

    for(i in 1:n)
    {
      ti <- na.omit(cbind(Xo[i,],W[,h]))
      T[i,h] <- sum(ti[,1]*ti[,2])/sum(ti[,2]^2)
    }

    th <- T[,h] ; cv2[1,h] <- cov(th,yz)^2

    ch. <- na.omit(cbind(th,Yo))
    ch <- sum(ch.[,1]*ch.[,2])/sum(ch.[,1]^2)
    C[,h] <- ch

    for(j in 1:p)
    {
      ph. <- na.omit(cbind(Xo[,j],th))
      P[j,h] <- sum(ph.[,1]*ph.[,2])/sum(ph.[,2]^2)
    }
    ph <- P[,h]

    X1 <- Xo - th%*%t(ph); Xo <- X1
    Y1 <- Yo - th%*%t(ch); Yo <- Y1

    B[,h] <- W[,1:h]%*%(solve(t(P[,1:h])%*%W[,1:h]))%*%C[1:h]

  } # end h
  RAdj<- c() # R2 ajustado para cada componente agregada
  diff<- c() # Diferencia de los R^2 ajustados con i componentes e i-1 componentes.
  for(i in 1:ncol(T)){
    model<- lm(scale(yz)~T[,1:ncol(T)])
    resumen <- summary(model)
    RAdj[i] <- resumen$adj.r.squared #calculo de los R^2 ajustados agregando I componentes

  }
  for(i in 2:ncol(T)){
    diff[i]<- RAdj[i]-RAdj[i-1]

  }
  ncomp<- (which(diff < 0.05)) #Numero de componentes a utilizar -1
  r.PLS1na <- list(W,T,C,P,B,cv2,ncomp)

  return(r.PLS1na)
  }

##### Seccion calculos PLS cuando hay datos faltantes en y

if(contarfila <1 & contarcolumna<1 & contary> 0 & contary<0.3){
      if(any(!is.finite(Z)))H <- p
```

```
else H <- qr(Xo)$rank
cv2 <- matrix(0,1,H)

W <- matrix(0,p,H); W. <- matrix(0,p,H)
T <- matrix(0,n,H); C <- matrix(0,1,H)
P <- matrix(0,p,H); B <- matrix(0,p,H)

for(h in 1:H)
{
  for(j in 1:p)
  {
    wh. <- na.omit(cbind(Xo[,j],Yo))
    W.[j,h] <- sum(wh.[,1]*wh.[,2])/sum(wh.[,2]^2)
  }

  nW. <- sqrt(sum(W.[,h]^2))
  W[,h] <- W.[,h]/nW.

  for(i in 1:n)
  {
    ti <- na.omit(cbind(Xo[i,],W[,h]))
    T[i,h] <- sum(ti[,1]*ti[,2])/sum(ti[,2]^2)
  }

  th <- T[,h] ; cv2[1,h] <- cov(th,yz)^2

  ch. <- na.omit(cbind(th,Yo))
  ch <- sum(ch.[,1]*ch.[,2])/sum(ch.[,1]^2)
  C[,h] <- ch

  for(j in 1:p)
  {
    ph. <- na.omit(cbind(Xo[,j],th))
    P[j,h] <- sum(ph.[,1]*ph.[,2])/sum(ph.[,2]^2)
  }
  ph <- P[,h]

  X1 <- Xo - th%*%t(ph); Xo <- X1
  Y1 <- Yo - th%*%t(ch); Yo <- Y1

  B[,h] <- W[,1:h]%*%(solve(t(P[,1:h])%*%W[,1:h]))%*%C[1:h]

} # end h
RAdj<- c() # R2 ajustado para cada componente agregada
diff<- c() # Diferencia de los R^2 ajustados con i componentes e i-1 componentes.
for(i in 1:ncol(T)){
  model<- lm(scale(yz)~T[,1:ncol(T)])
  resumen <- summary(model)
  RAdj[i] <- resumen$adj.r.squared #calculo de los R^2 ajustados agregando I componentes

}
for(i in 2:ncol(T)){
  diff[i]<- RAdj[i]-RAdj[i-1]
```

```
      }
      ncomp<- (which(diff < 0.05)) #Numero de componentes a utilizar -1
      r.PLS1na <- list(W,T,C,P,B,cv2,ncomp)

      return(r.PLS1na)

    }
  else{
    print("Warning Demasiados NA'S por fila o columna o en Y") #Pasa cuando no se cumplen las condicion
                                                               # del contador (30% o mas de NAS)
  }
} # end fPLS1na con y sin datos completos
```

5 Corrección NA en "y"

Para solucionar el problema de que las pruebas $F(R^2_{adj})$ son afectadas en sus gl, debido a que no participan las líneas que contienen NA en "y" se procede a estimar esos datos faltantes a partir de la regresión obtenida por fPLS1na con la siguiente ecuación:

$$\hat{Y} = T_h c_h$$

Donde t_h es el número de componentes necesarios en la regresión.

```
  yimput<- matrix(0,length(Yi),ncomp) #Matriz de las estimaciones
  for( i in 1:ncomp){
    yimput[,i]<- as.matrix(t(c[i]%*%T[,i])) #calculo de la estimacion ThCh
  }
  yimput<- rowSums(yimput)
  Yi<-scale(Yi)
  Yi[which(is.na(Yi))]<-yimput[which(is.na(Yi))] #reemplazo de la estimacion donde
                                                 # se encuentren datos faltantes
  Yinew<-scale(Yi)
  ######## Nuevamente PLS
  Xi <- YX[,-1]; Yi <- Yinew
  Xna <- fmd(Xi,0.1)
  YXna <- cbind(Yi,Xna) ; YXna
  YXna <- cbind(Yi,Xna) ; YXna
  fpls1 <- fPLS1na(YXna) #Calculo de PLS con los datos faltantes en y corregidos
```

6 Resultados aplicativos de las funciones

6.1 Resultados sin NAS

Con base en las funciones creadas anteriormente, se procede a realizar la regresión vía PLS de la base de datos Cornell con datos completos:

```
YX
```

```
##       y   x1   x2   x3   x4   x5   x6   x7
## 1  98.7 0.00 0.23 0.00 0.00 0.00 0.74 0.03
## 2  97.8 0.00 0.10 0.00 0.00 0.12 0.74 0.04
## 3  96.6 0.00 0.00 0.00 0.10 0.12 0.74 0.04
## 4  92.0 0.00 0.49 0.00 0.00 0.12 0.37 0.02
## 5  86.6 0.00 0.00 0.00 0.62 0.12 0.18 0.08
## 6  91.2 0.00 0.62 0.00 0.00 0.00 0.37 0.01
## 7  81.9 0.17 0.27 0.10 0.38 0.00 0.00 0.08
## 8  83.1 0.17 0.19 0.10 0.38 0.02 0.06 0.08
## 9  82.4 0.17 0.21 0.10 0.38 0.00 0.06 0.08
## 10 83.2 0.17 0.15 0.10 0.38 0.02 0.10 0.08
## 11 81.4 0.21 0.36 0.12 0.25 0.00 0.00 0.06
## 12 88.1 0.00 0.00 0.00 0.55 0.00 0.37 0.08
```

```
Yi <- YX[,1]
fpls0 <- fPLS1na(YX)
T <- fpls0[[2]]
ncomp<- fpls0[[7]][1]
ncomp
```

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

```
model<- lm(scale(Yi)~T[,1:ncomp]-1)
summary(model)
```

```
##
## Call:
## lm(formula = scale(Yi) ~ T[, 1:ncomp] - 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.24437 -0.05354  0.01472  0.05402  0.33808
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## T[, 1:ncomp]1  0.48204    0.02440  19.759 2.42e-09 ***
## T[, 1:ncomp]2  0.27311    0.05784   4.722 0.000814 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1613 on 10 degrees of freedom
## Multiple R-squared:  0.9763, Adjusted R-squared:  0.9716
## F-statistic: 206.4 on 2 and 10 DF,  p-value: 7.409e-09
```

Como se observa en la regresión múltiple vía PLS, se obtiene un R^2 de 0.9763, el cual se calculó con dos componentes, ya que la diferencia entre los R^2_{adj} era de 0.055, lo que indica que era necesario tener un segundo componente.

6.2 Resultados con NAS mayores al 30% por fila o columna

Con base en las funciones creadas anteriormente, se procede a realizar la regresión vía PLS de la base de datos Cornell con el 11% de NAs que no cumpla con la condición de datos faltantes menores al 30% en alguna columna o fila.

```
set.seed(1)
YXna0<- fmd (YX,0.11)
YXna0
```

```
##          y   x1   x2   x3   x4   x5   x6   x7
##  [1,]   NA 0.00 0.23 0.00 0.00 0.00 0.74   NA
##  [2,] 97.8   NA 0.10 0.00 0.00 0.12 0.74 0.04
##  [3,] 96.6 0.00 0.00   NA   NA 0.12 0.74   NA
##  [4,] 92.0 0.00 0.49 0.00 0.00 0.12 0.37 0.02
##  [5,] 86.6 0.00 0.00 0.00 0.62 0.12 0.18 0.08
##  [6,] 91.2 0.00 0.62 0.00 0.00 0.00 0.37 0.01
##  [7,] 81.9 0.17 0.27   NA 0.38 0.00 0.00 0.08
##  [8,] 83.1 0.17 0.19 0.10 0.38   NA 0.06 0.08
##  [9,] 82.4 0.17 0.21 0.10 0.38 0.00 0.06 0.08
## [10,] 83.2 0.17   NA 0.10 0.38 0.02   NA 0.08
## [11,] 81.4 0.21 0.36 0.12   NA 0.00 0.00 0.06
## [12,] 88.1 0.00 0.00 0.00 0.55 0.00 0.37 0.08
```

```
fpls1 <- fPLS1na(YXna0)
```

```
## [1] "Warning Demasiados NA'S por fila o columna o en Y"
```

Como se observa, en el individuo 3 se cuenta con un 42% de datos faltantes, por ende no corre el PLS.

6.3 Resultados con NAS menores al 30% por fila o columna

```
set.seed(1379)

YXna <- fmd(YX,0.11)
YXna
```

```
##          y   x1   x2   x3   x4   x5   x6   x7
##  [1,]   NA 0.00   NA 0.00 0.00 0.00 0.74   NA
##  [2,] 97.8 0.00 0.10 0.00 0.00 0.12 0.74 0.04
##  [3,] 96.6 0.00 0.00 0.00 0.10 0.12 0.74 0.04
##  [4,] 92.0 0.00 0.49 0.00 0.00 0.12 0.37 0.02
##  [5,] 86.6 0.00 0.00 0.00 0.62 0.12 0.18 0.08
##  [6,] 91.2 0.00   NA   NA 0.00 0.00 0.37 0.01
##  [7,] 81.9   NA 0.27   NA 0.38 0.00 0.00 0.08
##  [8,] 83.1 0.17 0.19   NA 0.38 0.02 0.06 0.08
##  [9,] 82.4 0.17 0.21 0.10   NA 0.00 0.06 0.08
## [10,]   NA   NA 0.15 0.10 0.38 0.02 0.10 0.08
## [11,] 81.4 0.21 0.36 0.12 0.25 0.00 0.00 0.06
## [12,] 88.1 0.00 0.00 0.00 0.55 0.00 0.37 0.08
```

```
Xina <- YXna[,-1]; Yina <- YXna[,1]
fpls1 <- fPLS1na(YXna)
T <- fpls1[[2]]
c <- fpls1[[3]]
```

```r
labs <- paste(c("T"),1:7,sep=""); colnames(T) <- labs
ncomp<- fpls1[[7]][1]-1
ncomp
```

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

```r
model<- lm(scale(Yina)~T[,1:ncomp]-1)
summary(model)
```

```
##
## Call:
## lm(formula = scale(Yina) ~ T[, 1:ncomp] - 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33913 -0.09847  0.08877  0.17650  0.38509
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## T[, 1:ncomp]  0.49216    0.03684   13.36 3.07e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2191 on 9 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.952,  Adjusted R-squared:  0.9467
## F-statistic: 178.5 on 1 and 9 DF,  p-value: 3.069e-07
```

Como se observa en la regresión múltiple vía PLS, se obtiene un R^2 de 0.952, el cual se calculó con un componente, ya que la diferencia entre los R^2_{adj} era de 0.038, lo que indica que no era necesario tener un segundo componente.

6.4 Resultados con imputación por el método de estimación

En primer lugar, se calcula el PLS con los datos que se tienen (1 paso), seguidamente se calcula las estimaciones de yi (2 paso), después donde están los NAs en la variable de respuesta y escalonada se cambian por los valores estimados (3 paso), se escalona de nuevo y se construye de nuevo la base de datos (4 paso). Por último se realiza de nuevo el pls (5 paso)

```r
fpls1 <- fPLS1na(YXna) #1 paso
Yina <- YXna[,1]
yimput<- matrix(0,length(Yina),ncomp)
for( i in 1:ncomp){
  yimput[,i]<- as.matrix(t(c[i]%*%T[,i])) #2 paso
}
yimput<- rowSums(yimput)
yimput
```

```
##  [1]  0.75335458  1.20500190  1.18554322  0.83637024  0.09134588  0.64006897
##  [7] -1.02421019 -0.93696894 -1.13404602 -0.85956150 -1.16948289 -0.11079080
```

```r
Y<-scale(Yina)
Y
```

```
##               [,1]
##  [1,]           NA
##  [2,]  1.590094616
##  [3,]  1.393178874
##  [4,]  0.638335197
##  [5,] -0.247785642
##  [6,]  0.507058035
##  [7,] -1.019038964
##  [8,] -0.822123222
##  [9,] -0.936990738
## [10,]           NA
## [11,] -1.101087190
## [12,] -0.001640965
## attr(,"scaled:center")
## [1] 88.11
## attr(,"scaled:scale")
## [1] 6.093977
```

```r
Y[which(is.na(Y))]<- yimput[which(is.na(Y))]#3 paso
Y
```

```
##               [,1]
##  [1,]  0.753354579
##  [2,]  1.590094616
##  [3,]  1.393178874
##  [4,]  0.638335197
##  [5,] -0.247785642
##  [6,]  0.507058035
##  [7,] -1.019038964
##  [8,] -0.822123222
##  [9,] -0.936990738
## [10,] -0.859561501
## [11,] -1.101087190
## [12,] -0.001640965
## attr(,"scaled:center")
## [1] 88.11
## attr(,"scaled:scale")
## [1] 6.093977
```

```r
Yinew<-scale(Y)#4 paso
Yinew
```

```
##               [,1]
##  [1,]  0.787471352
##  [2,]  1.651948328
##  [3,]  1.448505063
##  [4,]  0.668639214
##  [5,] -0.246855479
##  [6,]  0.533010371
```

```
##  [7,] -1.043674934
##  [8,] -0.840231669
##  [9,] -0.958906907
## [10,] -0.878910982
## [11,] -1.128442961
## [12,]  0.007448602
## attr(,"scaled:center")
## [1] -0.008850577
## attr(,"scaled:scale")
## [1] 0.9679148
```

```
######## Nuevamente PLS
YXna1 <- cbind(Yinew,Xina) ;  #4 paso
fpls2 <- fPLS1na(YXna1) #5 paso
T <- fpls2[[2]]
labs <- paste(c("T"),1:7,sep=""); colnames(T) <- labs
ncomp<- fpls1[[7]][1]-1
summary(lm(Yinew~T[,1:ncomp]))
```

```
##
## Call:
## lm(formula = Yinew ~ T[, 1:ncomp])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31795 -0.13094 -0.00573  0.11519  0.38435
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.04768    0.06187   0.771    0.459
## T[, 1:ncomp]  0.50448    0.03326  15.169 3.14e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.214 on 10 degrees of freedom
## Multiple R-squared:  0.9584, Adjusted R-squared:  0.9542
## F-statistic: 230.1 on 1 and 10 DF,  p-value: 3.139e-08
```

Como se observa en la regresión múltiple vía PLS con imputación, se obtiene un R^2 de 0.9584, el cual se calculó con un componente, ya que la diferencia entre los R^2_{adj} era de 0.0227, lo que indica que no era necesario tener un segundo componente. Además se cuenta con un mayor grado de libertad.