PCR - Selección de componentes por validación cruzada

Jerónimo Barragán - LU: 1472/21

2024-09-05

```
#install.packages("faraway)
library(faraway)
library(tidyverse)
```

```
datos <- faraway::meatspec
datos <- data.frame(datos)

semilla <- 91234

n <- nrow(datos)
train_size <- 175
test_size <- n - train_size
set.seed(semilla)
train_indices <- sample(seq_len(train_size), size = train_size)
train_set <- datos[train_indices, ]
test_set <- datos[-train_indices, ]
```

Parte 1

Ajusto un modelo lineal para fat que utiliza como predictoras todas las otras variables del dataset.

```
reg <- lm(formula = fat ~ ., data = train_set)
prediccionesT <- predict(reg, newdata = train_set)
realesT <- train_set$fat
mseT <- mean((prediccionesT - realesT)^2)
# Imprimo el MSE en el conjunto de entrenamiento.
cat("MSE en cjto. de entrenamiento: ", mseT)
```

```
## MSE en cjto. de entrenamiento:  0.4797529
```

```
prediccionesV <- predict(reg, newdata = test_set)
realesV <- test_set$fat
mseV <- mean((prediccionesV - realesV)^2)
# Imprimo el MSE en el conjunto de validación (test).
cat("MSE en cjto. de validación: ", mseV, "\n")
```

```
## MSE en cjto. de validación:  21.62494
```

```r
cat("(MSE en validación) / (MSE en entrenamiento) ", mseV/mseT)
```

```
## (MSE en validación) / (MSE en entrenamiento)  45.07515
```

El MSE en el conjunto de validacion es 45 veces el MSE en el conjunto de entrenamiento (es de esperar que el modelo performe mejor en los datos con los que se entrenó).

Parte 2

Calculo el MSE en el conjunto de validación del modelo lineal que utiliza como predictoras a las 4 primeras componentes principales.

```r
pcaT <- prcomp(x = train_set[, 1:100], center = TRUE)
proyT <- pcaT$x
pca_data <- data.frame(proyT[, 1:4], train_set[, 101])
colnames(pca_data) <- c(paste0("PC", 1:4), "fat")
reg_pcaT <- lm(fat ~ PC1 + PC2 + PC3 + PC4, data = pca_data)

eval_centrados_proy <- data.frame(as.matrix(sweep(test_set[, 1:100], 2, colMeans(train_set[,
    1:100]), FUN = "-")) %*% as.matrix(pcaT$rotation))[, 1:4]
colnames(eval_centrados_proy) <- c("PC1", "PC2", "PC3", "PC4")

predicciones_pcaV <- predict(reg_pcaT, newdata = eval_centrados_proy)

mse_pcaV <- mean((predicciones_pcaV - realesV)^2)

# Imprimo el MSE en el conjunto de validación.
cat("MSE en cjto. de validación: ", mse_pcaV)
```

```
## MSE en cjto. de validación:  21.58712
```

Comparamos el MSE del modelo que usa como predictoras a las 4 primeras componentes principales (PCR) con el MSE del modelo que usa como predictoras a todas las variables del modelo original (siempre teniendo como variable de respuesta a “fat”).

```r
cat("(MSE con PCR) / (MSE con todas las predictoras) = ", mse_pcaV/mseV)
```

```
## (MSE con PCR) / (MSE con todas las predictoras) =  0.9982514
```

Vemos que el MSE con PCR es casi igual al MSE del modelo con todas las predictoras. Quizás podrían obtenerse distintos eligiendo en lugar de las primeras 4 componentes principales, alguna otra cantidad.

Parte 3

```r
# install.packages('pls') install.packages('rsample')
library(pls)
library(rsample)
```

Voy a ir probando distintas cantidades de componentes principales para tomar como predictoras de la PCR que tiene como variable de respuesta a "fat". Fijando un numero de componentes principales M, voy a calcular el MSE por 5-folds-cross-validation de cada modelo de PCR. Es decir que voy a partir el conjunto de entrenamiento en 5 folds, y realizar 5 iteraciones, donde en la iteración i-ésima voy a:

1) Elegir el fold i-ésimo para testear la regresión.
2) Con el resto de folds, entreno una PCR.
3) Calculo el MSE del modelo entrenado en 2) sobre el fold i-ésimo, y lo guardo.

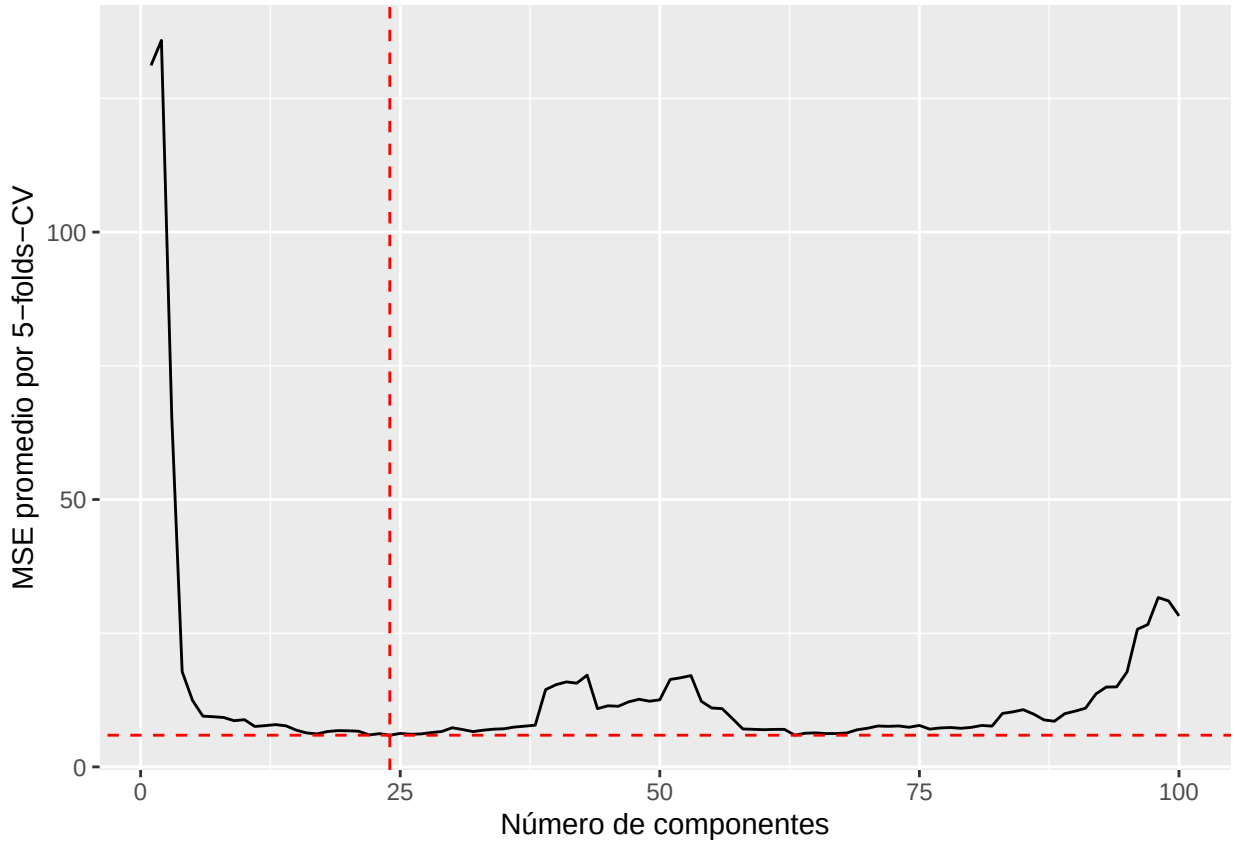
Al final, promedio todos los MSE guardados, y ese será el MSE promedio obtenido utilizando M componentes principales.

```
Ms <- c(1:100)
MSEs <- c()
set.seed(semilla)
folds <- vfold_cv(train_set, v = 5)
# Como los pcas no dependen de M, en la posición i-ésima de pcas voy a guardar
# el PCA que uso para entrenar el modelo que excluye al fold i-ésimo para
# evaluación.
pcas <- list()
for (i in c(1:5)) {
    fold <- folds$splits[[i]]
    train_folds <- analysis(fold)
    test_fold <- assessment(fold)
    train_folds_cov <- train_folds[, 1:100]
    pca <- prcomp(x = train_folds_cov)
    pcas[[i]] <- pca
}
for (M in Ms) {
    MSEs_M <- c()
    for (i in c(1:5)) {
        fold <- folds$splits[[i]]
        train_folds <- analysis(fold)
        test_fold <- assessment(fold)
        pca_mi <- pcas[[i]]
        proy_mi <- pca_mi$x
        pcr_data <- data.frame(proy_mi[, 1:M], data.frame(train_folds[, "fat"]))
        colnames(pcr_data) <- c(paste0("PC", 1:M), "fat")

        predictors <- colnames(pcr_data)[1:M]
        formula <- as.formula(paste("fat", "~", paste(predictors, collapse = "+")))
        pcr_modelo <- lm(formula, data = pcr_data)

        eval <- test_fold
        eval_cov <- eval[, 1:100]
        eval_proy <- data.frame(as.matrix(sweep(eval_cov, 2, colMeans(train_folds[,
            1:100]), FUN = "-")) %*% as.matrix(pca_mi$rotation))[, 1:M]
        eval_proy <- data.frame(eval_proy, test_fold[, 101])
        colnames(eval_proy) <- c(paste0("PC", 1:M), "fat")
        eval_pred <- predict(pcr_modelo, newdata = data.frame(eval_proy))

        eval_Y <- test_fold[, 101]
        MSEs_M <- c(MSEs_M, mean((eval_pred - eval_Y)^2))
```

```
    }
    MSEs <- c(MSEs, mean(MSEs_M))
}
```

Ahora grafico el MSE promedio obtenido para cada cantidad de componentes principales utilizadas.

```
Moptimo <- which.min(MSEs)
MSEoptimo <- min(MSEs)
ggplot(data.frame(Ms, MSEs), aes(x = Ms, y = MSEs)) + geom_line() + geom_abline(slope = 0,
    intercept = MSEoptimo, color = "red", linetype = "dashed") + geom_vline(xintercept = Moptimo,
    color = "red", linetype = "dashed") + xlab("Número de componentes") + ylab("MSE promedio por 5-folds
```




```
paste("El MSE óptimo es", paste(MSEoptimo, paste("y se alcanza con", paste(Moptimo,
    "componentes principales."))))
```

```
## [1] "El MSE óptimo es 5.92539428533629 y se alcanza con 24 componentes principales."
```

Se observa que el MSE en función del número de componentes tiende a fluctuar bastante, lo cual puede deberse al porcentaje de la relación lineal existente entre la respuesta y las predictoras que capta particularmente cada una de las componentes, independientemente de cuanta varianza total explique cada una. Además, considerando que calculamos el MSE usando validación cruzada (CV), es de esperar que al aumentar la cantidad de componentes el MSE también aumente por sobreajuste (cuantos más parámetros tiene el modelo, más puede adaptarse a cada conjunto de entrenamiento).

Ahora hago PCR con el número de componentes óptimo hallado anteriormente.

```r
predictors_opt <- colnames(proyT)[1:Moptimo]
train_pca <- data.frame(proyT[, 1:Moptimo], train_set[, 101])
colnames(train_pca) <- c(paste0("PC", 1:Moptimo), "fat")
formula_opt <- as.formula(paste("fat", "~", paste(predictors_opt, collapse = "+")))
reg_opt <- lm(formula_opt, data = train_pca)

eval_centrados_proy_opt <- data.frame(as.matrix(sweep(test_set[, 1:100], 2, colMeans(train_set[,
    1:100]), FUN = "-")) %*% as.matrix(pcaT$rotation))[, 1:Moptimo]
colnames(eval_centrados_proy_opt) <- paste0("PC", 1:Moptimo)

predicciones_pcaV_opt <- predict(reg_opt, newdata = eval_centrados_proy_opt)

mse_pcaV_opt <- mean((predicciones_pcaV_opt - realesV)^2)

# Imprimo el MSE en el conjunto de validación.
cat("MSE en cjto. de validación con PCR y usando el número óptimo de componentes hallado: ",
    mse_pcaV_opt, "\n")
```

```
## MSE en cjto. de validación con PCR y usando el número óptimo de componentes hallado:  4.711416
```

```r
cat("MSE en conjunto de evaluación con PCR y sólo 4 componentes principales: ",
    mse_pcaV, "\n")
```

```
## MSE en conjunto de evaluación con PCR y sólo 4 componentes principales:  21.58712
```

```r
cat("MSE con regresión lineal sin PCA: ", mseV)
```

```
## MSE con regresión lineal sin PCA:  21.62494
```

Claramente PCR con el número de componentes óptimo arroja muchísimos mejores resultados que los otros dos modelos, cuyos MSE son casi 4 veces más grandes. Esto puede deberse a que la variable de respuesta (fat) tenga una correlación lineal mayor con las componentes principales, que con las demás variables del dataset (los canales del espectro de absorbancia). Una de las ventajas de usar validación cruzada para hallar el número óptimo de componentes es que no influye tanto la elección de los conjuntos de entrenamiento y de validación a la hora de entrenar el modelo, reduciendo el riesgo de sobreajuste. Si hiciésemos el mismo gráfico sin hacer validación cruzada ni separar en conjuntos de entrenamiento y validación, el MSE disminuiría a medida que aumenta el número de componentes porque el modelo sobreajustaría mucho al conjunto de entrenamiento.

Parte 4

```r
pcr4 <- pcr(formula = fat ~ ., ncomp = 4, data = train_set)
predicciones_4 <- predict(pcr4, newdata = test_set, ncomp = 4)
mse_4 <- mean((predicciones_4 - realesV)^2)
cat("MSE en cjto. de validación con PCR y sólo 4 componentes principales: ", mse_4)
```

```
## MSE en cjto. de validación con PCR y sólo 4 componentes principales:  21.58712
```

Comparo el MSE obtenido con el de la parte 2.

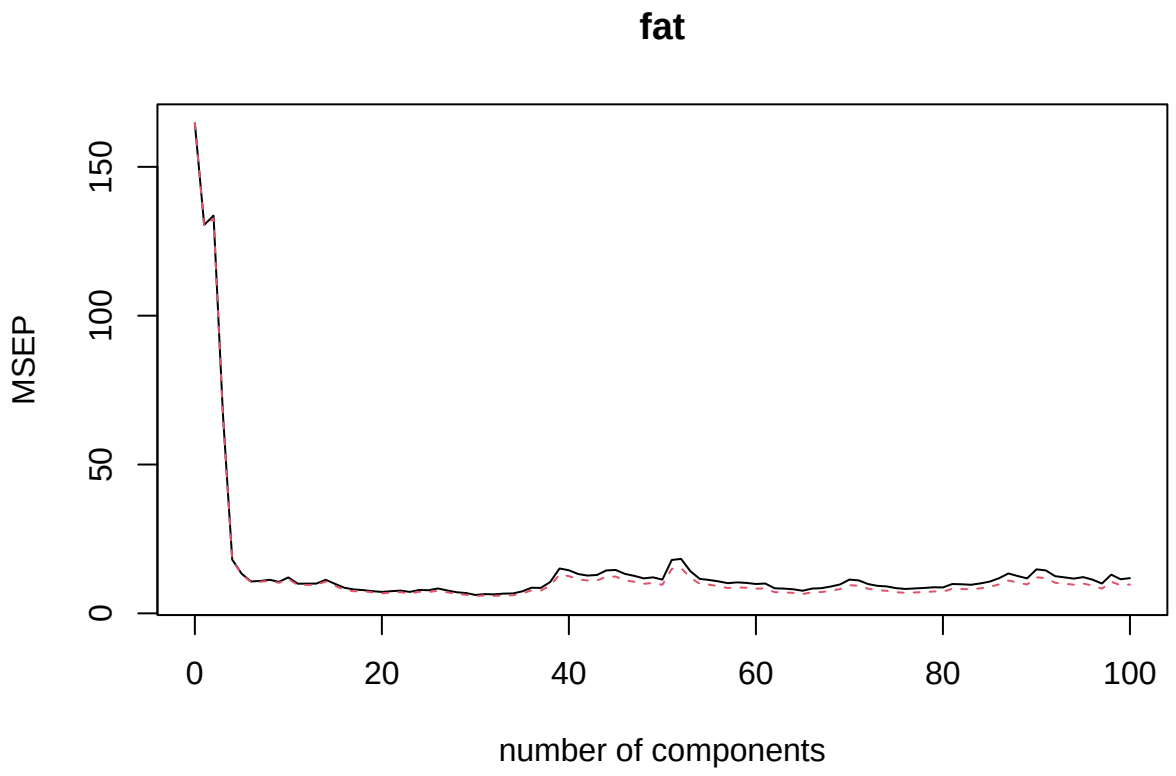
```
cat("Proporción entre los MSE: ", mse_pcaV / mse_4)
```

```
## Proporción entre los MSE:  1
```

Al hacer PCR con las primeras 4 componentes principales, y evaluar el modelo en el conjunto de evaluación, se obtuvo el mismo MSE de la parte 2. En esta parte me gustaría remarcar que si antes de proyectar, centrase los datos de evaluación calculando las medias de sus propias columnas y restándoselas a cada una, no obtendría los mismos MSEs pero no diferirían significativamente (su proporción daba aproximadamente 1). Por lo tanto, decidí evaluar cada modelo de PCR restandole a cada columna del dataset de evaluación la media de la respectiva columna del dataset de entrenamiento, y luego proyectando los datos obtenidos. Esto arrojaba resultados mejores y más similares a los de "pcr" de la librería "pls".

Ahora selecciono el número de componentes óptimo usando validación cruzada partiendo el conjunto de entrenamiento en 5 folds, como en la parte 3.

```
set.seed(semilla)
pcr_optimo <- pcr(fat ~ ., data = train_set, validation = "CV", segments = 5)
validationplot(pcr_optimo, val.type = "MSEP")
```




```
cv_msep <- MSEP(pcr_optimo)
Moptimo2 <- which.min(cv_msep$val[1, , ])
cat("El número óptimo de componentes es:", Moptimo2)
```

```
## El número óptimo de componentes es: 31
```

Observo que el número óptimo de componentes que devuelve "pcr" es mayor al calculado en la parte 3. Sin embargo, vemos que los gráficos del MSE en función del número de componentes de las partes 3 y 4 resultan bastante similares, y pueden diferir debido a detalles implementativos de la librería "pls". De todas maneras, los resultados obtenidos parecen razonables en ambos casos, en el sentido de que no difieren significativamente. Veamos esto más detalladamente, imprimiendo un ranking de las cantidades de componentes según el MSE obtenido por "pcr".

```
indices_ordenados <- order(as.vector(cv_msep$val[1, , ]))

cat("Ranking de posiciones (de menor a mayor):\n")
```

```
## Ranking de posiciones (de menor a mayor):
```

```
i <- 0
for (rank in seq_along(indices_ordenados)) {
    index <- indices_ordenados[rank]
    i <- i + 1
    cat(i, "-", sprintf("%d componentes - MSE: %.2f\n", index, cv_msep$val[1, , ][index]))
    if (i == 10)
        break
}
```

```
## 1 - 31 componentes - MSE: 6.18
## 2 - 33 componentes - MSE: 6.38
## 3 - 32 componentes - MSE: 6.47
## 4 - 34 componentes - MSE: 6.63
## 5 - 35 componentes - MSE: 6.68
## 6 - 30 componentes - MSE: 6.83
## 7 - 29 componentes - MSE: 7.10
## 8 - 21 componentes - MSE: 7.20
## 9 - 24 componentes - MSE: 7.21
## 10 - 36 componentes - MSE: 7.38
```

Se ve que la cantidad óptima obtenida en la parte 3 está dentro del top 10 por debajo de la cantidad óptima obtenida usando "pcr". Además los MSE son considerablemente similares, por lo que en este sentido es que los resultados no difieren significativamente.

```
predictors_opt2 <- colnames(proyT)[1:Moptimo2]
train_pca2 <- data.frame(proyT[, 1:Moptimo2], train_set[, 101])
colnames(train_pca2) <- c(paste0("PC", 1:Moptimo2), "fat")
formula_opt2 <- as.formula(paste("fat", "~", paste(predictors_opt2, collapse = "+")))
reg_opt2 <- lm(formula_opt2, data = train_pca2)

eval_centrados_proy_opt2 <- data.frame(as.matrix(sweep(test_set[, 1:100], 2, colMeans(train_set[,
    1:100]), FUN = "-")) %*% as.matrix(pcaT$rotation))[, 1:Moptimo2]
colnames(eval_centrados_proy_opt2) <- paste0("PC", 1:Moptimo2)

predicciones_pcaV_opt2 <- predict(reg_opt2, newdata = eval_centrados_proy_opt2)

mse_pcaV_opt2 <- mean((predicciones_pcaV_opt2 - realesV)^2)
cat("MSE en cjto. de validación con PCR y usando el número óptimo de componentes hallado: ",
    mse_pcaV_opt2, "\n")
```

```
## MSE en cjto. de validación con PCR y usando el número óptimo de componentes hallado:  3.3005
```

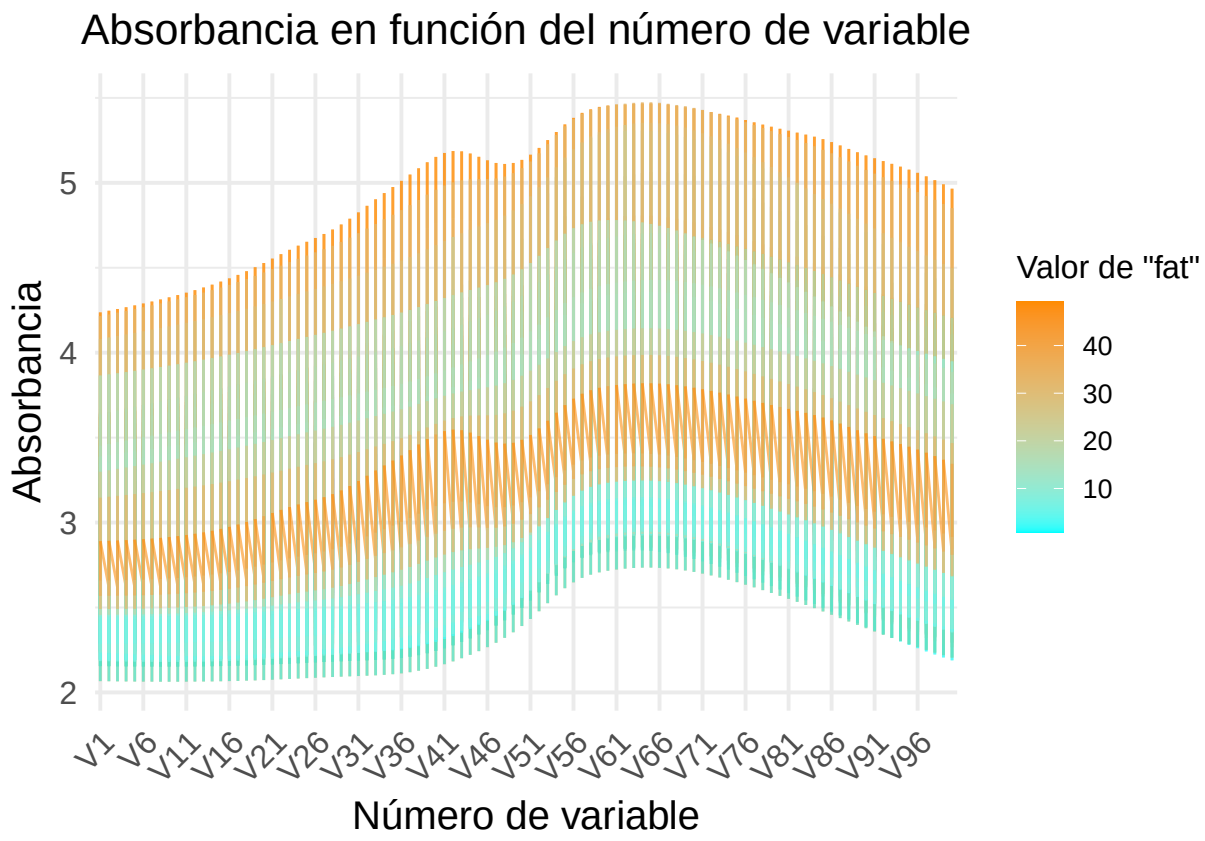
```
cat("Proporción entre el MSE de la parte 4 y el de la parte 3: ", mse_pcaV_opt2/mse_pcaV_opt)
```

```
## Proporción entre el MSE de la parte 4 y el de la parte 3:  0.7005327
```

Se obtuvo un MSE un 30% más chico que el de la parte 3, utilizando "pcr" de la librería "pls".

Extra: naturaleza del problema

```
library(reshape2)
```



A partir de este gráfico parecería ser que el comportamiento a lo largo de las variables "V1" a "V100" es idéntico para toda observación, salvo por diferencias en el valor de "fat" que trasladan las curvas que se observan hacia arriba o hacia abajo. En este comportamiento además parece tener mucha influencia el orden en que consideramos las variables, dada la forma "continua" de las curvas que se observan. Por lo tanto, pareciera ser que el orden de las variables "V1" a "V100" sí es relevante.