semana 2

luis

2/7/2021

Contents

division de datos **1**

opciones de entrenamiento **3**

trazar predictores **4**

preprocesamiento básico **10**

Creacion de covariables **13**

PCA **16**
PCA y SVD 17
Reflexiones finales sobre las PC 23

Predicciones con regresion lineal **24**
Notas y lectura adicional 27

prediciendo con regresion lineal multivariable **28**

recursos **34**

division de datos

1. Submiestra aleatoria

Division 75% para conjunto de entrenamiento y 25% para conjunto de prueba

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
dim(training)
```

```
## [1] 3451   58
```

2. K-fold:

- conjunto de entrenamiento

```
set.seed(32323)
folds <- createFolds(y=spam$type,k=10,
```

```
                              list=TRUE,returnTrain=TRUE)
sapply(folds,length)
```

```
## Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
##   4140   4142   4141   4140   4141   4141   4142   4141   4141   4140
```

```
folds[[1]][1:10]
```

```
##  [1]  1  2  4  5  6  7  8  9 10 11
```

- conjunto de prueba

```
set.seed(32323)
folds <- createFolds(y=spam$type,k=10,
                              list=TRUE,returnTrain=FALSE)
sapply(folds,length)
```

```
## Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
##    461    459    460    461    460    460    459    460    460    461
```

```
folds[[1]][1:10]
```

```
##  [1]  3 19 33 38 44 51 66 67 72 83
```

3. Remuestreo

```
set.seed(32323)
folds <- createResample(y=spam$type,times=10,
                              list=TRUE)
sapply(folds,length)
```

```
## Resample01 Resample02 Resample03 Resample04 Resample05 Resample06 Resample07
##       4601       4601       4601       4601       4601       4601       4601
## Resample08 Resample09 Resample10
##       4601       4601       4601
```

```
folds[[1]][1:10]
```

```
##  [1] 1 1 2 2 3 3 4 5 5 7
```

4. Time Slices

```
set.seed(32323)
tme <- 1:1000
folds <- createTimeSlices(y=tme,initialWindow=20,
                          horizon=10)
names(folds)
```

```
## [1] "train" "test"
```

```
folds$train[[1]]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
folds$test[[1]]
```

```
##  [1] 21 22 23 24 25 26 27 28 29 30
```

```
folds$train[[2]]
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
```

```
folds$test[[2]]
```

```
##  [1] 22 23 24 25 26 27 28 29 30 31
```

```
folds$train[[971]]
```

```
##  [1] 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989
## [20] 990
```

```
folds$test[[971]]
```

```
##  [1]  991  992  993  994  995  996  997  998  999 1000
```

opciones de entrenamiento

Opciones por default

```
function (x, y, method = "rf", preProcess = NULL, ..., weights = NULL,
    metric = ifelse(is.factor(y), "Accuracy", "RMSE"), maximize = ifelse(metric ==
        "RMSE", FALSE, TRUE), trControl = trainControl(), tuneGrid = NULL,
    tuneLength = 3)
NULL
```

Opciones métricas

Resultados continuos: * *RMSE* = Error cuadrático medio de raíz * *RSquared* = R^2 de modelos de regresión

Resultados categóricos: * *Accuracy* = Fracción correcta * *Kappa* = Una medida de concordancia

Para traincontro

```
args(trainControl)
```

```
## function (method = "boot", number = ifelse(grepl("cv", method),
##     10, 25), repeats = ifelse(grepl("[d_]cv$", method), 1, NA),
##     p = 0.75, search = "grid", initialWindow = NULL, horizon = 1,
##     fixedWindow = TRUE, skip = 0, verboseIter = FALSE, returnData = TRUE,
##     returnResamp = "final", savePredictions = FALSE, classProbs = FALSE,
##     summaryFunction = defaultSummary, selectionFunction = "best",
##     preProcOptions = list(thresh = 0.95, ICAcomp = 3, k = 5,
##         freqCut = 95/5, uniqueCut = 10, cutoff = 0.9), sampling = NULL,
##     index = NULL, indexOut = NULL, indexFinal = NULL, timingSamps = 0,
##     predictionBounds = rep(FALSE, 2), seeds = NA, adaptive = list(min = 5,
##         alpha = 0.05, method = "gls", complete = TRUE), trim = FALSE,
##     allowParallel = TRUE)
## NULL
```

remuestreo trainControl

- *method*
  - *boot* = arranque
  - *boot632* = bootstrapping con ajuste
  - *cv* = validación cruzada
  - *repeatedcv* = validación cruzada repetida

  - *LOOCV* = dejar una validación cruzada
- *numbers*
  - Para validación de arranque / cruzada
  - Número de submuestras a tomar
- *repeat*
  - Número de veces que se repite el submuestreo
  - Si es grande, esto puede *ralentizar las cosas*

trazar predictores

Usaremos datos salariales

```
library(ISLR); library(ggplot2); library(caret); library(gridExtra);
data(Wage)
summary(Wage)
```

```
##       year           age                     maritl           race
##  Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480
##  1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293
##  Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190
##  Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37
##  3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55
##  Max.   :2009   Max.   :80.00
##
##                education                     region               jobclass
##  1. < HS Grad      :268   2. Middle Atlantic   :3000   1. Industrial :1544
##  2. HS Grad        :971   1. New England       :   0   2. Information:1456
##  3. Some College   :650   3. East North Central:   0
##  4. College Grad   :685   4. West North Central:   0
##  5. Advanced Degree:426   5. South Atlantic    :   0
##                           6. East South Central:   0
##                           (Other)              :   0
##             health      health_ins      logwage           wage
##  1. <=Good     : 858   1. Yes:2083   Min.   :3.000   Min.   : 20.09
##  2. >=Very Good:2142   2. No : 917   1st Qu.:4.447   1st Qu.: 85.38
##                                      Median :4.653   Median :104.92
##                                      Mean   :4.654   Mean   :111.70
##                                      3rd Qu.:4.857   3rd Qu.:128.68
##                                      Max.   :5.763   Max.   :318.34
##
```

Dividiendo los conjuntos de datos

```
inTrain <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
training <- Wage[inTrain,]
testing <- Wage[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 2102   11
```

```
## [1] 898  11
```

grafico de caracteristicas

```
featurePlot(x=training[,c("age","education","jobclass")],
            y = training$wage,
```

```
            plot="pairs")
```

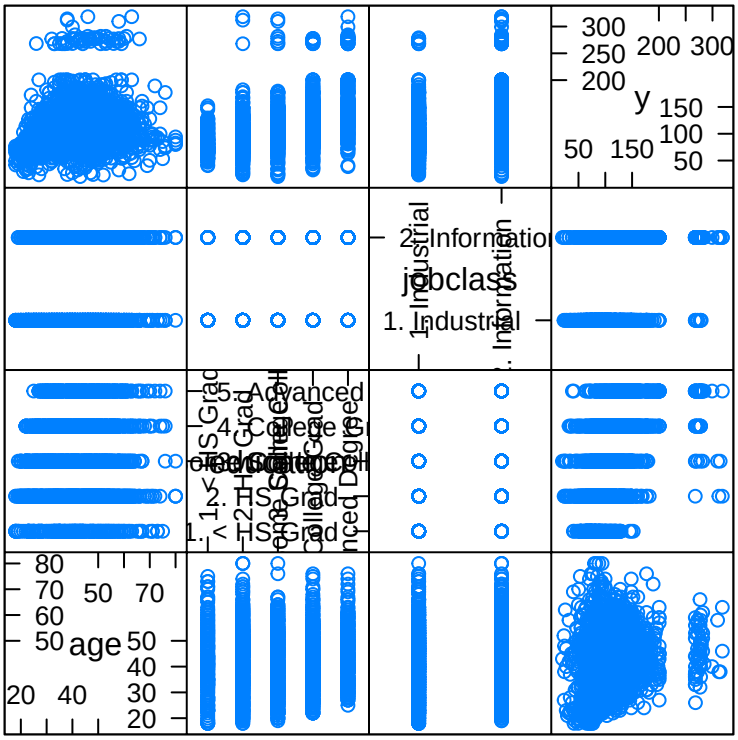


grafico de dispercion

```
library(ggplot2)
qplot(age,wage,colour=jobclass,data=training)
```

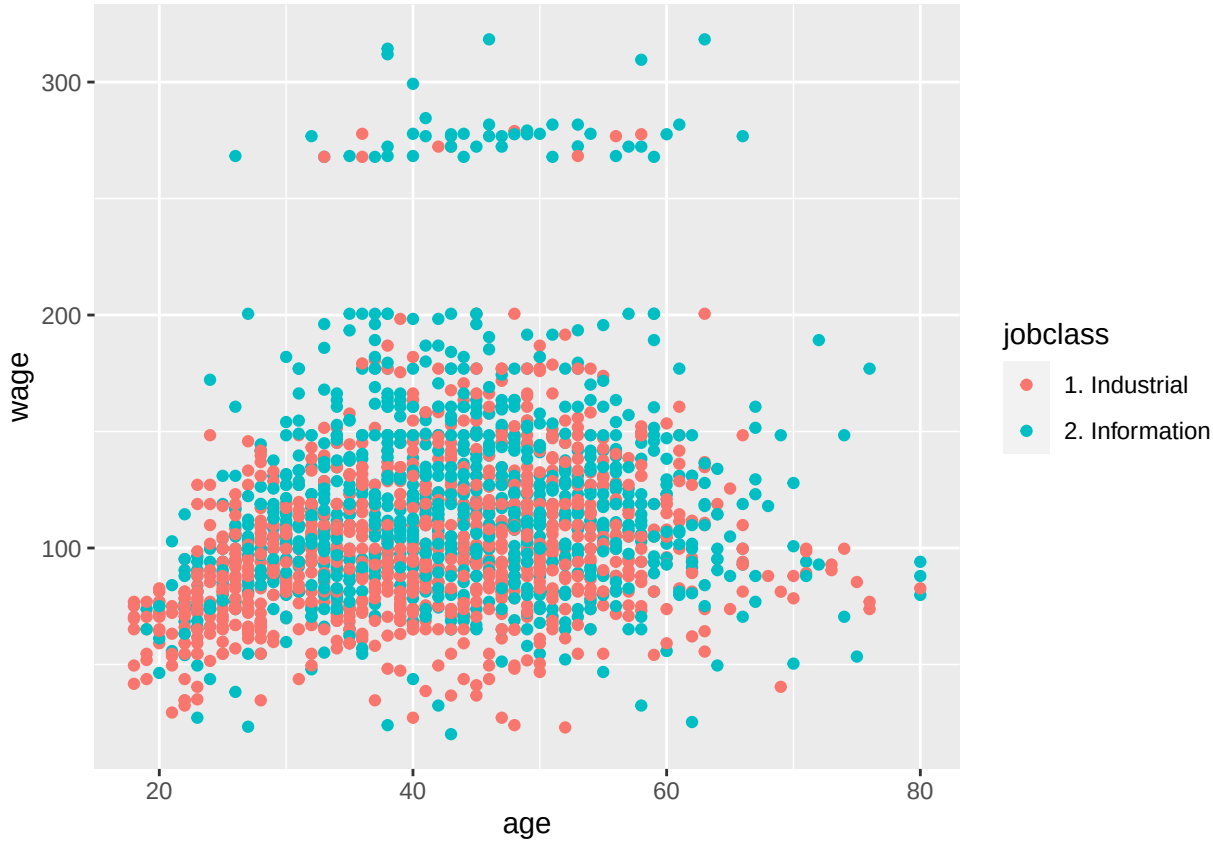


grafico de disppercion con regresion lineal

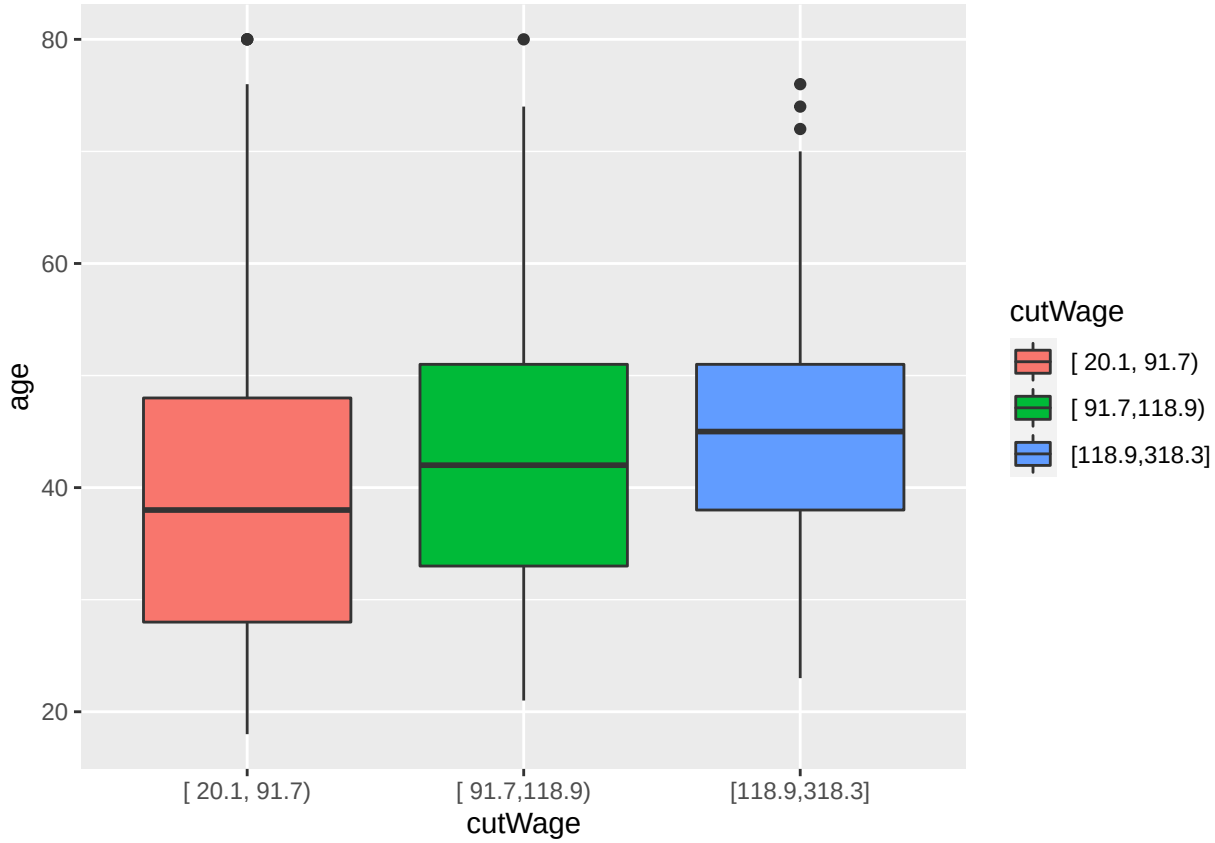
```
qq <- qplot(age,wage,colour=education,data=training)
qq +  geom_smooth(method='lm',formula=y~x)
```

crando subcategorias y haciendo boxplot

```
library(Hmisc)
cutWage <- cut2(training$wage,g=3)
table(cutWage)
```

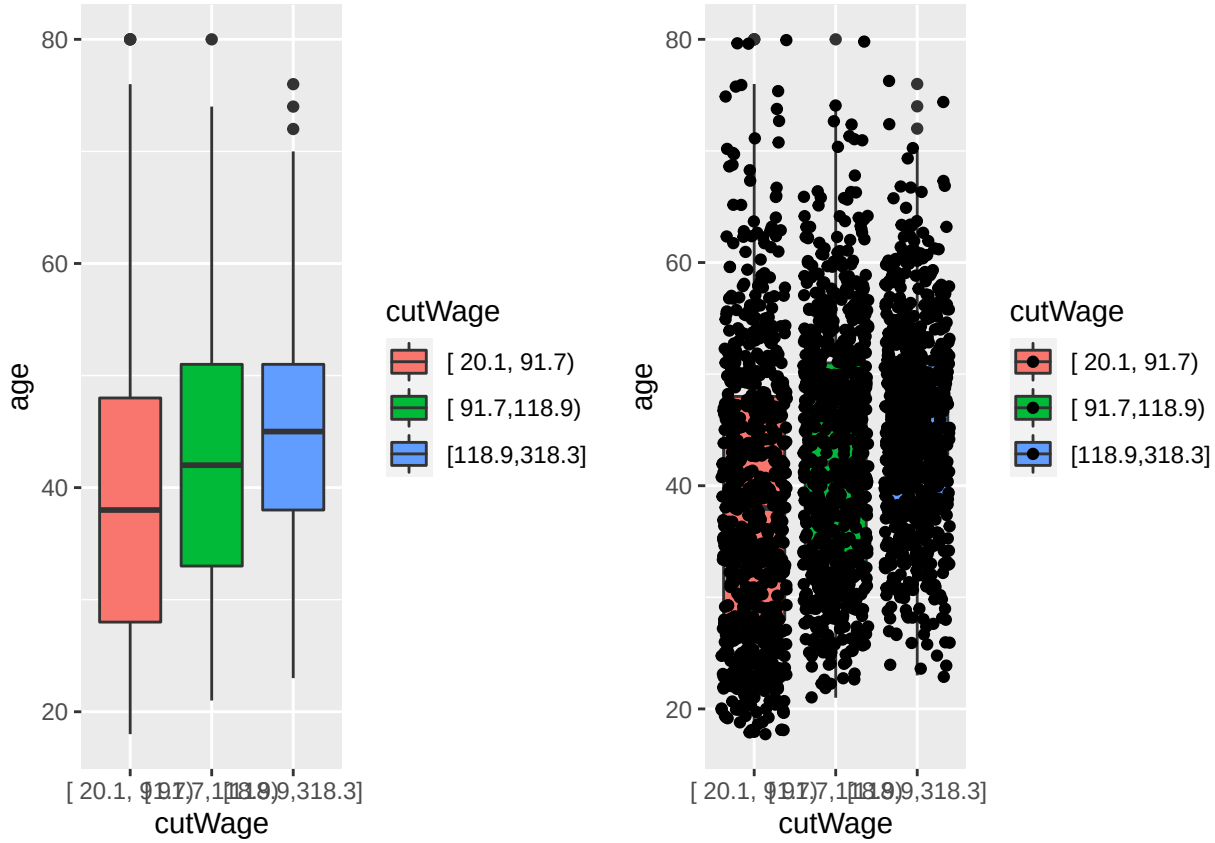
```
## cutWage
## [ 20.1, 91.7) [ 91.7,118.9) [118.9,318.3]
##           702           723           677
```

```
p1 <- qplot(cutWage,age, data=training,fill=cutWage,
      geom=c("boxplot"))
p1
```



boxplot con puntos superpuestos

```
p2 <- qplot(cutWage,age, data=training,fill=cutWage,
      geom=c("boxplot","jitter"))
grid.arrange(p1,p2,ncol=2)
```



tablas

```
t1 <- table(cutWage,training$jobclass)
t1
```

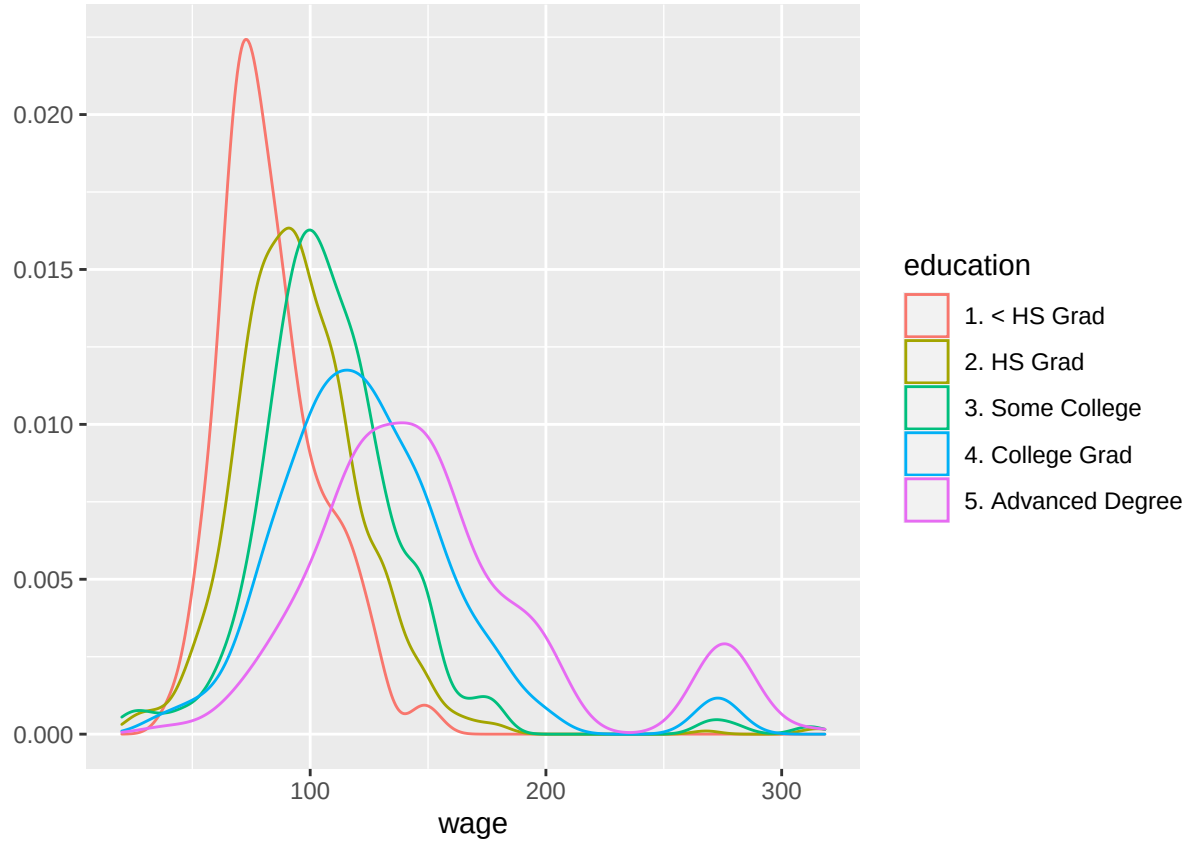
```
##
## cutWage          1. Industrial 2. Information
##    [ 20.1, 91.7)           453            249
##    [ 91.7,118.9)           372            351
##    [118.9,318.3]           255            422
```

```
prop.table(t1,1)
```

```
##
## cutWage          1. Industrial 2. Information
##    [ 20.1, 91.7)     0.6452991      0.3547009
##    [ 91.7,118.9)     0.5145228      0.4854772
##    [118.9,318.3]     0.3766617      0.6233383
```

Graficos de densidad

```
qplot(wage,colour=education,data=training,geom="density")
```



Notas y lectura adicional

- Haz tus parcelas solo en el set de entrenamiento
  - ¡No utilice el equipo de prueba para explorar!
- Cosas que deberías buscar
  - Desequilibrio en los resultados / predictores
  - Valores atípicos
  - Grupos de puntos no explicados por un predictor
  - Variables sesgadas

preprocesamiento básico

la Standarizacion

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
# entrenamiento
trainCapAve <- training$capitalAve
trainCapAveS <- (trainCapAve  - mean(trainCapAve))/sd(trainCapAve)
mean(trainCapAveS)
```

```
## [1] 1.033455e-17
```

```
sd(trainCapAveS)
```

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

```
# prueba
testCapAve <- testing$capitalAve
testCapAveS <- (testCapAve  - mean(trainCapAve))/sd(trainCapAve)
mean(testCapAveS)
```

```
## [1] -0.02143666
```

```
sd(testCapAveS)
```

```
## [1] 0.6315994
```

con *preProcess* function

```
preObj <- preProcess(training[,-58],method=c("center","scale"))
trainCapAveS <- predict(preObj,training[,-58])$capitalAve
mean(trainCapAveS)
```

```
## [1] 1.033455e-17
```

```
sd(trainCapAveS)
```

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

```
testCapAveS <- predict(preObj,testing[,-58])$capitalAve
mean(testCapAveS)
```

```
## [1] -0.02143666
```

```
sd(testCapAveS)
```

```
## [1] 0.6315994
```

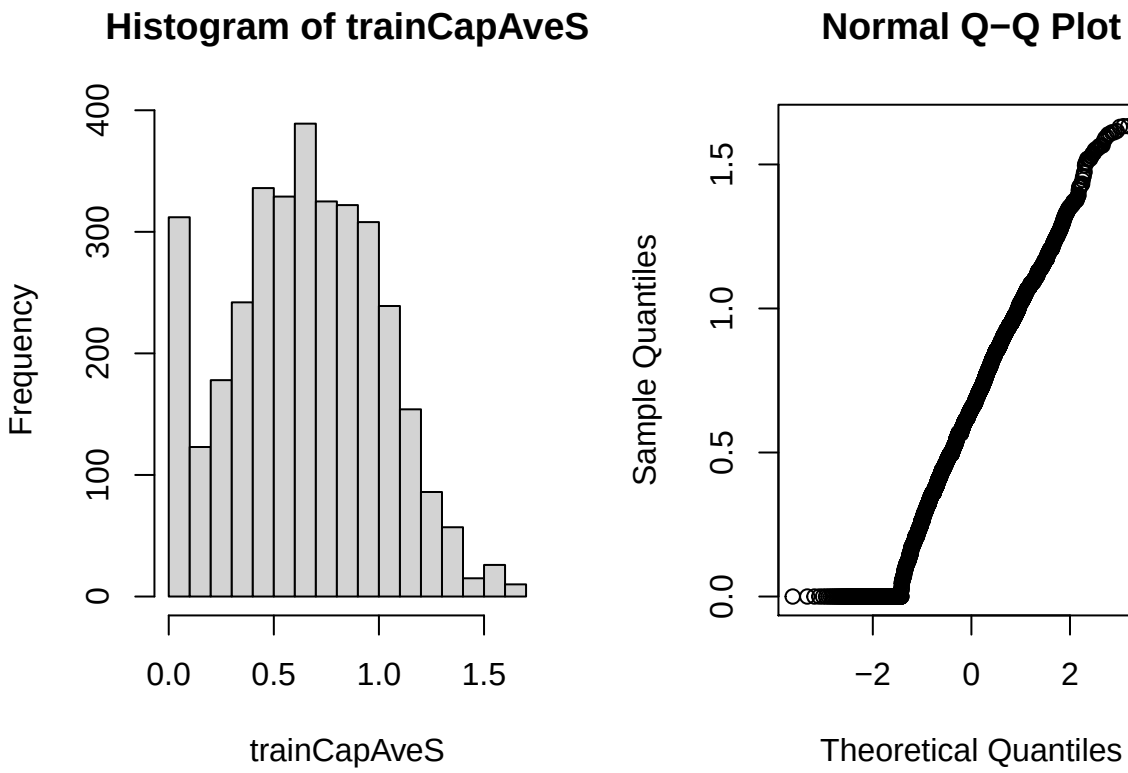
preProcess como argumento

```
set.seed(32343)
modelFit <- train(type ~.,data=training,
                  preProcess=c("center","scale"),method="glm")
modelFit
```

```
## Generalized Linear Model
##
## 3451 samples
##   57 predictor
##    2 classes: 'nonspam', 'spam'
##
## Pre-processing: centered (57), scaled (57)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9233584  0.8390136
```

transformacion box cox para hacer mas "normales" los datos

```
preObj <- preProcess(training[,-58],method=c("BoxCox"))
trainCapAveS <- predict(preObj,training[,-58])$capitalAve
par(mfrow=c(1,2)); hist(trainCapAveS); qqnorm(trainCapAveS)
```



Para eliminar NA's y estandarizar

```
set.seed(13343)

# Make some values NA
training$capAve <- training$capitalAve
selectNA <- rbinom(dim(training)[1],size=1,prob=0.05)==1
training$capAve[selectNA] <- NA

# Impute and standardize
preObj <- preProcess(training[,-58],method="knnImpute")
capAve <- predict(preObj,training[,-58])$capAve

# Standardize true values
capAveTruth <- training$capitalAve
capAveTruth <- (capAveTruth-mean(capAveTruth))/sd(capAveTruth)
```

comparando los cuantiles

```
quantile(capAve - capAveTruth)
```

```
##           0%          25%          50%          75%         100%
## -0.538671223  0.001009906  0.001681090  0.002005423  0.966747398
```

```
quantile((capAve - capAveTruth)[selectNA])
```

```
##           0%          25%          50%          75%         100%
## -0.5386712234 -0.0181903917  0.0004948413  0.0138904101  0.9667473978
```

```
quantile((capAve - capAveTruth)[!selectNA])
```

```
##           0%          25%          50%          75%         100%
## -0.474735628  0.001062302  0.001681090  0.001988968  0.002237522
```

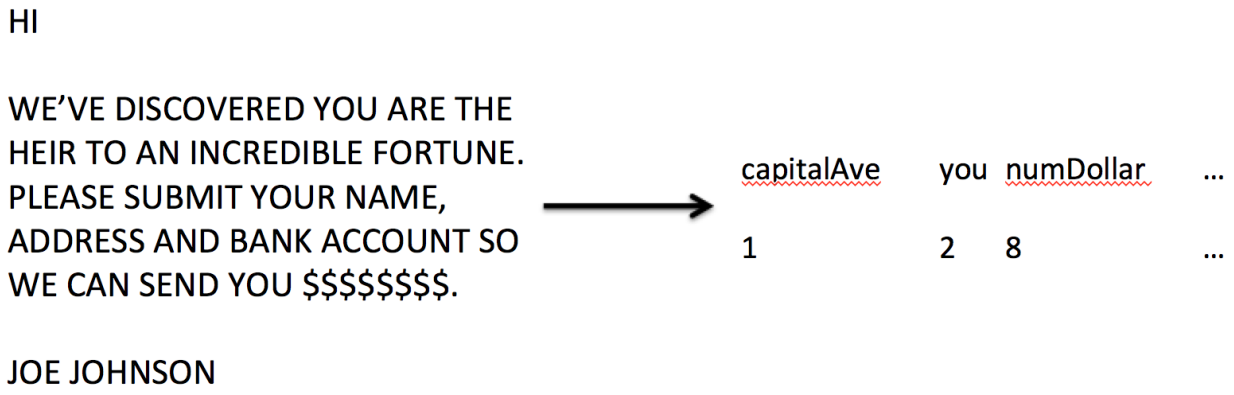
Notas y lectura adicional

- El entrenamiento y la prueba deben procesarse de la misma manera.
- Es probable que las transformaciones de prueba sean imperfectas
  - Especialmente si los conjuntos de prueba / entrenamiento se recopilaron en diferentes momentos
- ¡Cuidado al transformar variables factoriales!
- preprocesamiento con intercalación

Creacion de covariables

Hay dos niveles para crear covariables

1. De datos brutos a covariables



2. Transformando covariables

```
library(kernlab);data(spam)
spam$capitalAveSq <- spam$capitalAve^2
```

Nivel 1, datos brutos -> covariables

- Depende en gran medida de la aplicación
- El acto de equilibrio es el resumen frente a la pérdida de información.
- Ejemplos:
  - Archivos de texto: frecuencia de palabras, frecuencia de frases ([Google ngrams] (https://books.google.com/ngrams)), frecuencia de letras mayúsculas.
  - Imágenes: bordes, esquinas, manchas, crestas ([detección de características de visión por computadora] (http://en.wikipedia.org/wiki/Feature_detection_ (computer_vision)))
  - Páginas web: número y tipo de imágenes, posición de los elementos, colores, videos ([Pruebas A / B] (http://en.wikipedia.org/wiki/A/B_testing))
  - Personas: Altura, peso, color de pelo, sexo, país de origen.
- Cuanto más conocimiento tenga del sistema, mejor será el trabajo que hará.
- En caso de duda, opte por más funciones
- Se puede automatizar, ¡pero tenga cuidado!

Nivel 2, covariables ordenadas -> nuevas covariables

- Más necesario para algunos métodos (regresión, svms) que para otros (árboles de clasificación).
- Debe realizarse *solo en el conjunto de entrenamiento*
- El mejor enfoque es a través del análisis exploratorio (gráficos / tablas)
- Se deben agregar nuevas covariables a los marcos de datos

Usaremos datos de sueldo como ejemplo

```
library(ISLR); library(caret); data(Wage);
inTrain <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
training <- Wage[inTrain,]; testing <- Wage[-inTrain,]
```

1. agregar variables ficticias

Idea básica: convertir variables de factor en variables indicadoras

```
table(training$jobclass)
```

```
##
##  1. Industrial 2. Information
##           1069           1033
```

```
dummies <- dummyVars(wage ~ jobclass,data=training)
head(predict(dummies,newdata=training))
```

```
##        jobclass.1. Industrial jobclass.2. Information
## 231655                       1                       0
## 86582                        0                       1
## 161300                       1                       0
## 155159                       0                       1
## 376662                       0                       1
## 450601                       1                       0
```

2. Eliminar covariables cero

```
nsv <- nearZeroVar(training,saveMetrics=TRUE)
nsv
```

```
##            freqRatio percentUnique zeroVar   nzv
## year        1.034384    0.33301618   FALSE FALSE
## age         1.116883    2.90199810   FALSE FALSE
## maritl      3.374713    0.23786870   FALSE FALSE
## race        8.146226    0.19029496   FALSE FALSE
## education   1.438525    0.23786870   FALSE FALSE
## region      0.000000    0.04757374    TRUE  TRUE
## jobclass    1.034850    0.09514748   FALSE FALSE
## health      2.480132    0.09514748   FALSE FALSE
## health_ins  2.199391    0.09514748   FALSE FALSE
## logwage     1.133333   19.12464320   FALSE FALSE
## wage        1.133333   19.12464320   FALSE FALSE
```

3. spline

```
library(splines)
bsBasis <- bs(training$age,df=3)
head(bsBasis)
```

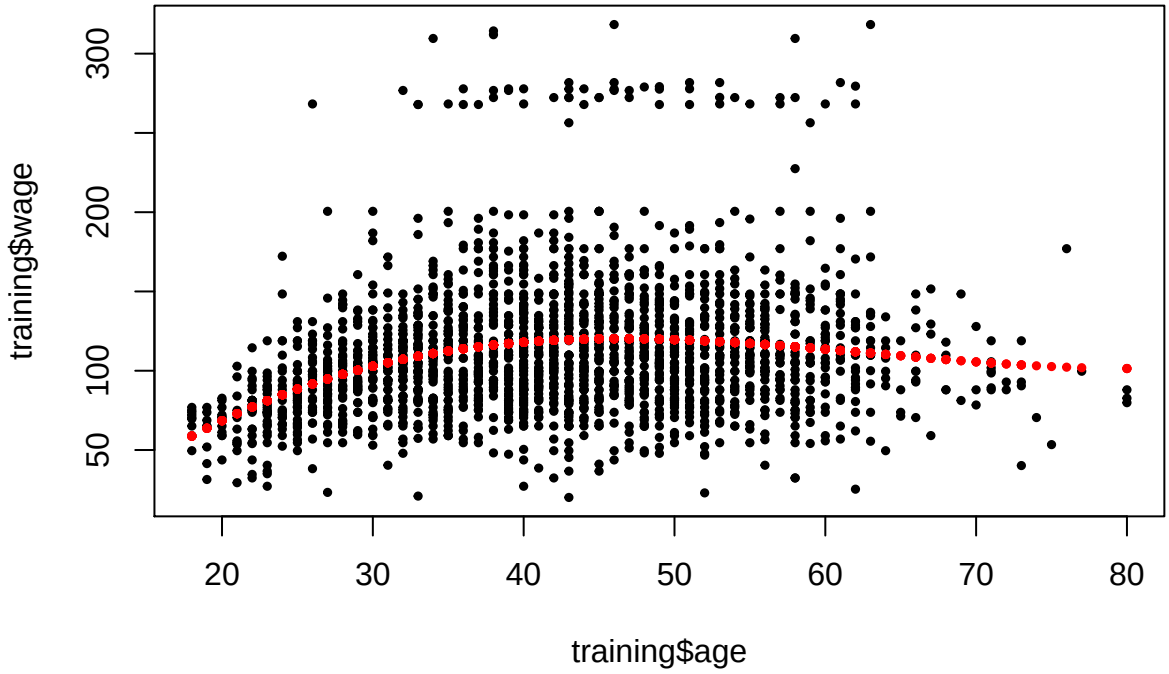
```
##              1          2          3
```

```
## [1,] 0.0000000 0.00000000 0.000000000
## [2,] 0.2368501 0.02537679 0.000906314
## [3,] 0.4163380 0.32117502 0.082587862
## [4,] 0.4308138 0.29109043 0.065560908
## [5,] 0.3063341 0.42415495 0.195763821
## [6,] 0.4241549 0.30633413 0.073747105
```

See also: ns(),poly()

ajustando curvas:

```
lm1 <- lm(wage ~ bsBasis,data=training)
plot(training$age,training$wage,pch=19,cex=0.5)
points(training$age,predict(lm1,newdata=training),col="red",pch=19,cex=0.5)
```



En el conjunto de prueba

```
head(predict(bsBasis,age=testing$age))
```

```
##             1          2           3
## [1,] 0.0000000 0.00000000 0.000000000
## [2,] 0.2368501 0.02537679 0.000906314
## [3,] 0.4163380 0.32117502 0.082587862
## [4,] 0.4308138 0.29109043 0.065560908
## [5,] 0.3063341 0.42415495 0.195763821
## [6,] 0.4241549 0.30633413 0.073747105
```

Notas y lectura adicional

- Creación de características de nivel 1 (datos sin procesar a covariables)

  - La ciencia es clave. Google "extracción de caracteristicas para [tipo de datos]"
  - Errar en la creación excesiva de caracteristicas
  - En algunas aplicaciones (imágenes, voces) la creación automática de funciones es posible / necesaria
    * http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf
- Creación de características de nivel 2 (covariables a nuevas covariables)
  - La función *preProcess* en *caret* manejará algunos preprocesos.
  - Cree nuevas covariables si cree que mejorarán el ajuste
  - Utilice un análisis exploratorio en el conjunto de entrenamiento para crearlos.
  - ¡Tenga cuidado con el sobreajuste!
- preprocesamiento con intercalación
- Si desea ajustar modelos de spline, use el método *gam* en el paquete *caret* que permite suavizar múltiples variables.
- Más información sobre la creación de características / ordenación de datos en el curso Obtención de datos de la pista del curso Ciencia de datos.

PCA

Sirve para Predictores correlacionados,y disminuye el número total de predictores

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]

# obtiene las covarianzas altas
M <- abs(cor(training[,-58]))
diag(M) <- 0
which(M > 0.8,arr.ind=T)
```

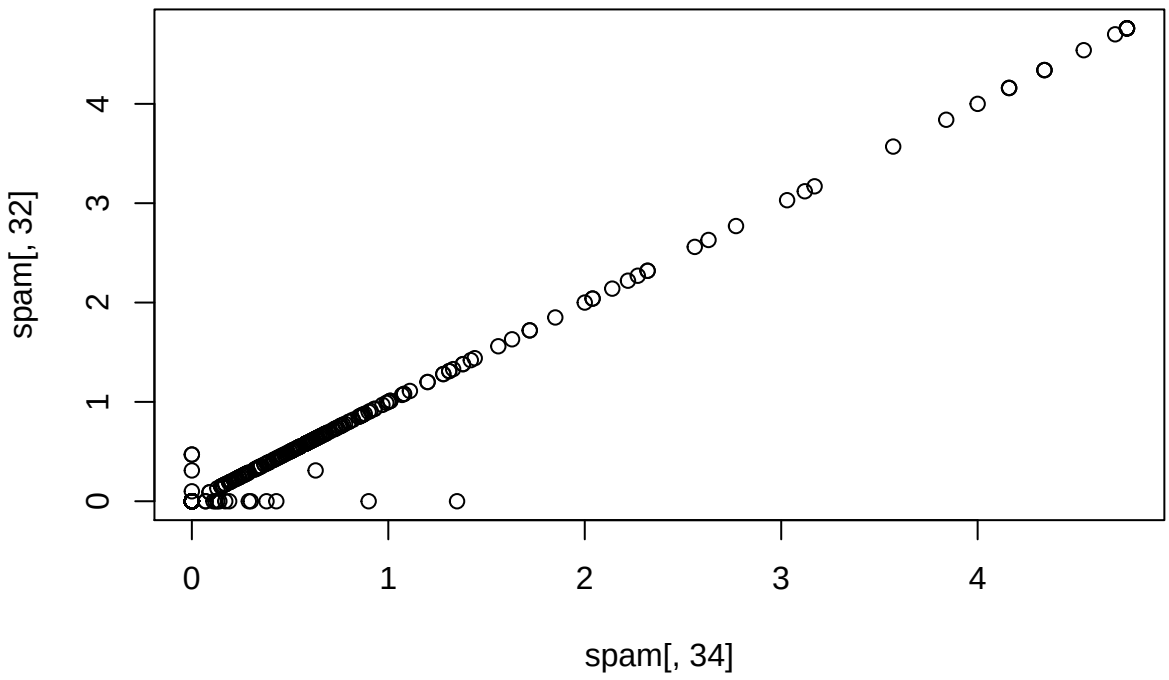
```
##        row col
## num857  32  31
## num415  34  31
## telnet  31  32
## num415  34  32
## direct  40  32
## telnet  31  34
## num857  32  34
## direct  40  34
## num857  32  40
## num415  34  40
```

tenemos 2 predictores correlacionados en la columna 32 y 34

```
names(spam)[c(34,32)]
```

```
## [1] "num415" "num857"
```

```
plot(spam[,34],spam[,32])
```



idea basica de PCA

- Puede que no necesitemos todos los predictores
- Una combinación ponderada de predictores podría ser mejor
- Debemos elegir esta combinación para capturar la "mayor información" posible
- Beneficios
  - Número reducido de predictores
  - Reducción de ruido (debido al promedio)

Problemas relacionados

Tiene variables multivariadas $X_1, \ldots, X_n$ entonces $X_1 = (X_{11}, \ldots, X_{1m})$

- Encuentre un nuevo conjunto de variables multivariadas que no estén correlacionadas y explique tanta varianza como sea posible.
- Si junta todas las variables en una matriz, busque la mejor matriz creada con menos variables (rango inferior) que explique los datos originales.

El primer objetivo es estadístico y el segundo objetivo son los datos.

PCA y SVD

SVD

Si X es una matriz con cada variable en una columna y cada observación en una fila, entonces el SVD es una "descomposición de la matriz"

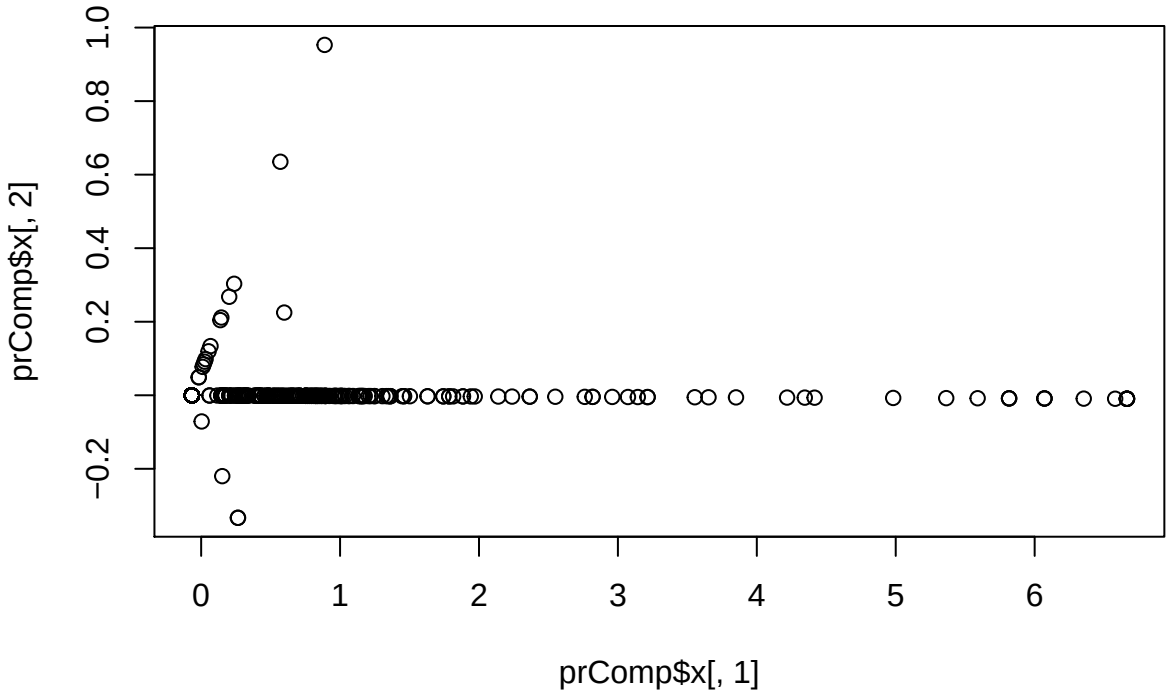
$$X = UDV^T$$

donde las columnas de U son ortogonales (vectores singulares izquierdos), las columnas de V son ortogonales (vectores singulares derechos) y D es una matriz diagonal (valores singulares).

PCA Los componentes principales son iguales a los valores singulares correctos si primero escala (resta la media, divide por la desviación estándar) las variables.

ejemplo utilizando prcomp

```
smallSpam <- spam[,c(34,32)]
prComp <- prcomp(smallSpam)
plot(prComp$x[,1],prComp$x[,2])
```

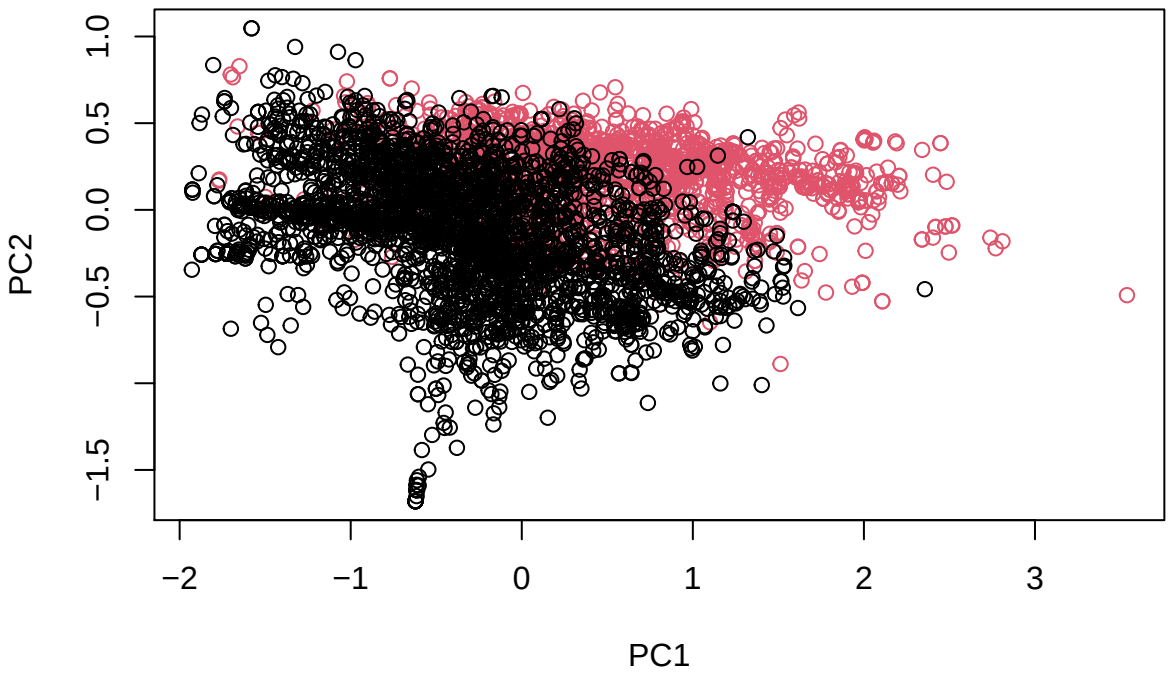



```
prComp$rotation
```

```
##              PC1        PC2
## num415 0.7080625  0.7061498
## num857 0.7061498 -0.7080625
```

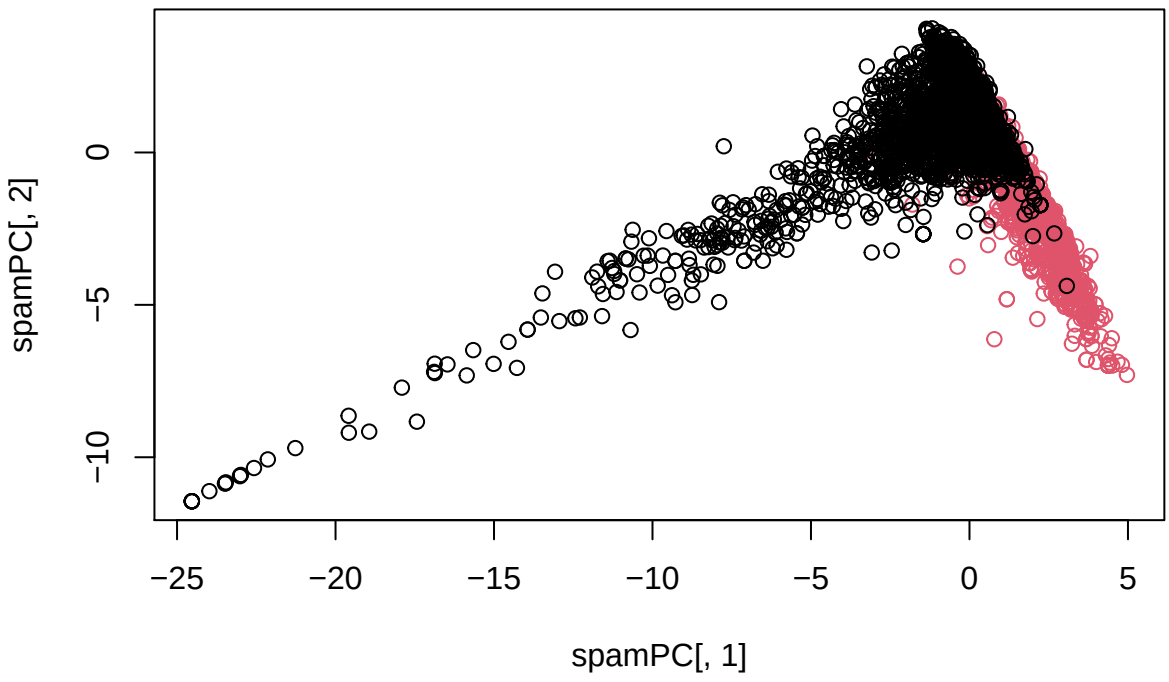
ejemplo en datos SPAM

```
typeColor <- ((spam$type=="spam")*1 + 1)
prComp <- prcomp(log10(spam[,-58]+1))
plot(prComp$x[,1],prComp$x[,2],col=typeColor,xlab="PC1",ylab="PC2")
```



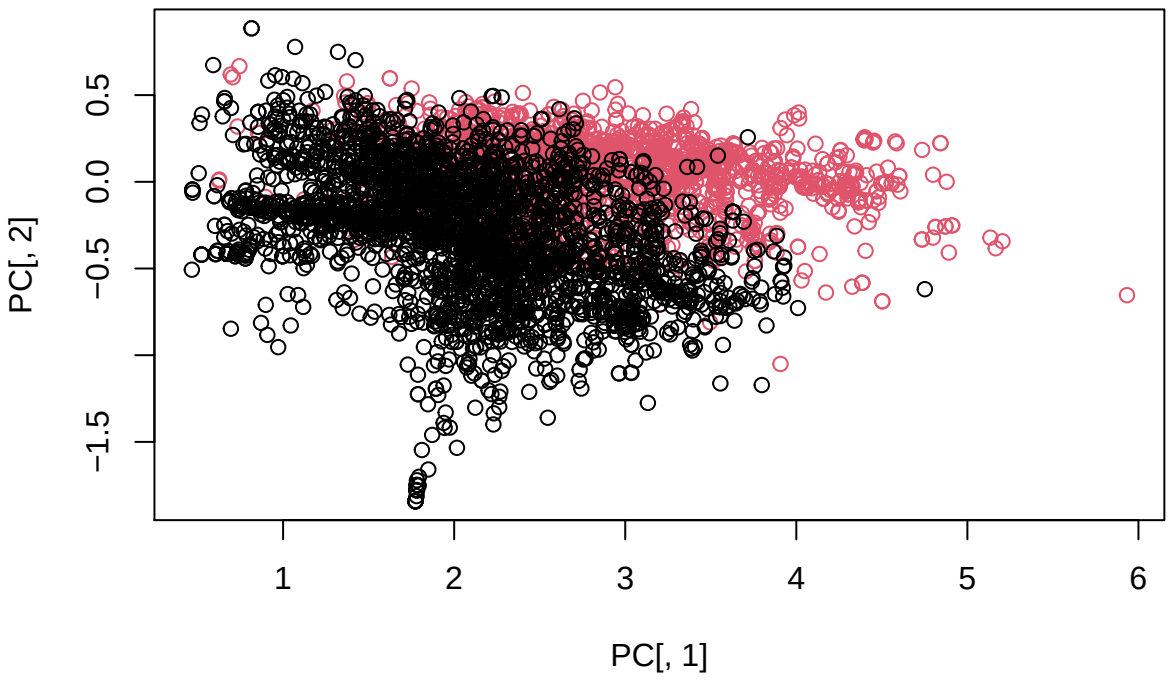
PCA con caret

```
preProc <- preProcess(log10(spam[,-58]+1),method="pca",pcaComp=2)
spamPC <- predict(preProc,log10(spam[,-58]+1))
plot(spamPC[,1],spamPC[,2],col=typeColor)
```

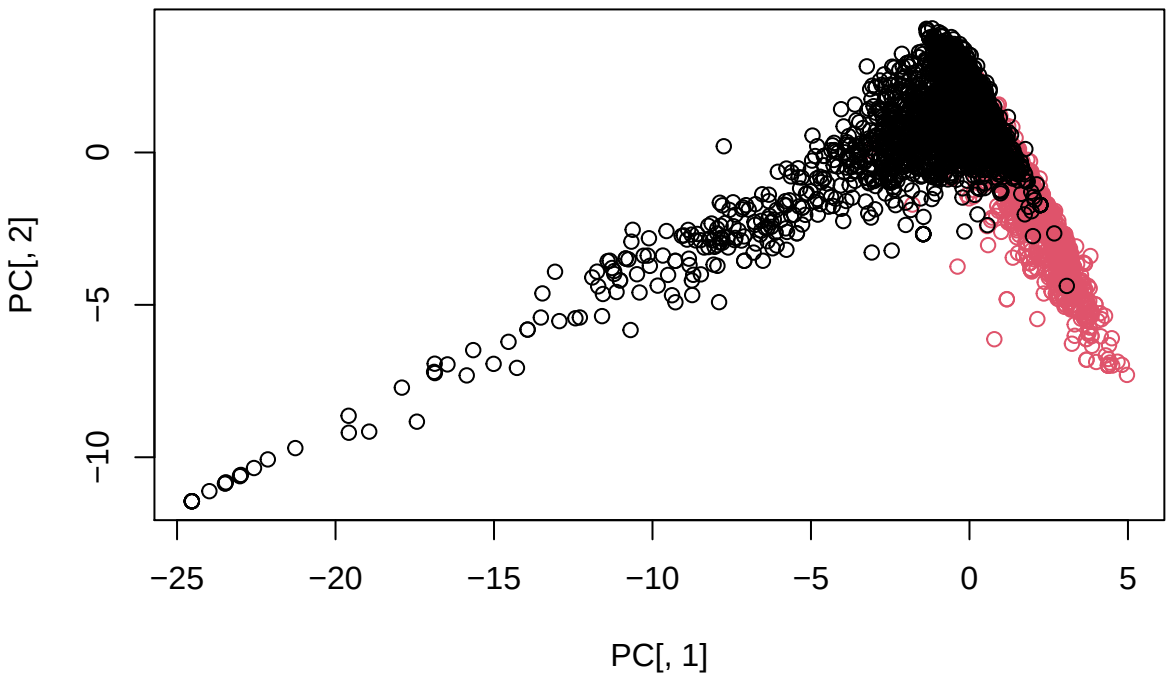


lo mismo *"a mano"*

```
dat<-log10(spam[,-58]+1)
prComp <- prcomp(dat)
pr<-prComp$rotation[,1:2]
PC<-as.matrix(dat) %*%as.matrix(pr)
plot(PC[,1],PC[,2],col=typeColor)
```




```
dat<-log10(spam[,-58]+1)
dat<-scale(dat)
prComp <- prcomp(dat)
pr<-prComp$rotation[,1:2]
PC<-as.matrix(dat) %*%as.matrix(pr)
plot(PC[,1],PC[,2],col=typeColor)
```



preprocesando con PCA pon preProcess

```
preProc <- preProcess(log10(training[,-58]+1),method="pca",pcaComp=2)
trainPC <- predict(preProc,log10(training[,-58]+1))
trainPC$y<-training$type
modelFit <- train(y ~.,method="glm",data=trainPC)

testPC <- predict(preProc,log10(testing[,-58]+1))
confusionMatrix(testing$type,predict(modelFit,testPC))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam     659   38
##    spam         69  384
##
##                Accuracy : 0.907
##                  95% CI : (0.8887, 0.9231)
##     No Information Rate : 0.633
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8028
##
##  Mcnemar's Test P-Value : 0.003729
##
##             Sensitivity : 0.9052
```

```
##             Specificity : 0.9100
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.8477
##              Prevalence : 0.6330
##          Detection Rate : 0.5730
##    Detection Prevalence : 0.6061
##       Balanced Accuracy : 0.9076
##
##        'Positive' Class : nonspam
##
```

agregados a la funcion train

```
modelFit <- train(type ~ .,method="glm",preProcess="pca",data=training)
confusionMatrix(testing$type,predict(modelFit,testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction nonspam spam
##    nonspam     664   33
##    spam         50  403
##
##                Accuracy : 0.9278
##                  95% CI : (0.9113, 0.9421)
##     No Information Rate : 0.6209
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8478
##
##  Mcnemar's Test P-Value : 0.07905
##
##             Sensitivity : 0.9300
##             Specificity : 0.9243
##          Pos Pred Value : 0.9527
##          Neg Pred Value : 0.8896
##              Prevalence : 0.6209
##          Detection Rate : 0.5774
##    Detection Prevalence : 0.6061
##       Balanced Accuracy : 0.9271
##
##        'Positive' Class : nonspam
##
```

Reflexiones finales sobre las PC

- Más útil para modelos de tipo lineal
- Puede dificultar la interpretación de los predictores.
- ¡Cuidado con los valores atípicos!
  - Transformar primero (con logs/ Box Cox)
  - Trazar predictores para identificar problemas
- Para obtener más información, consulte
  - Análisis exploratorio de datos
  - Elementos de aprendizaje estadístico

Predicciones con regresion lineal

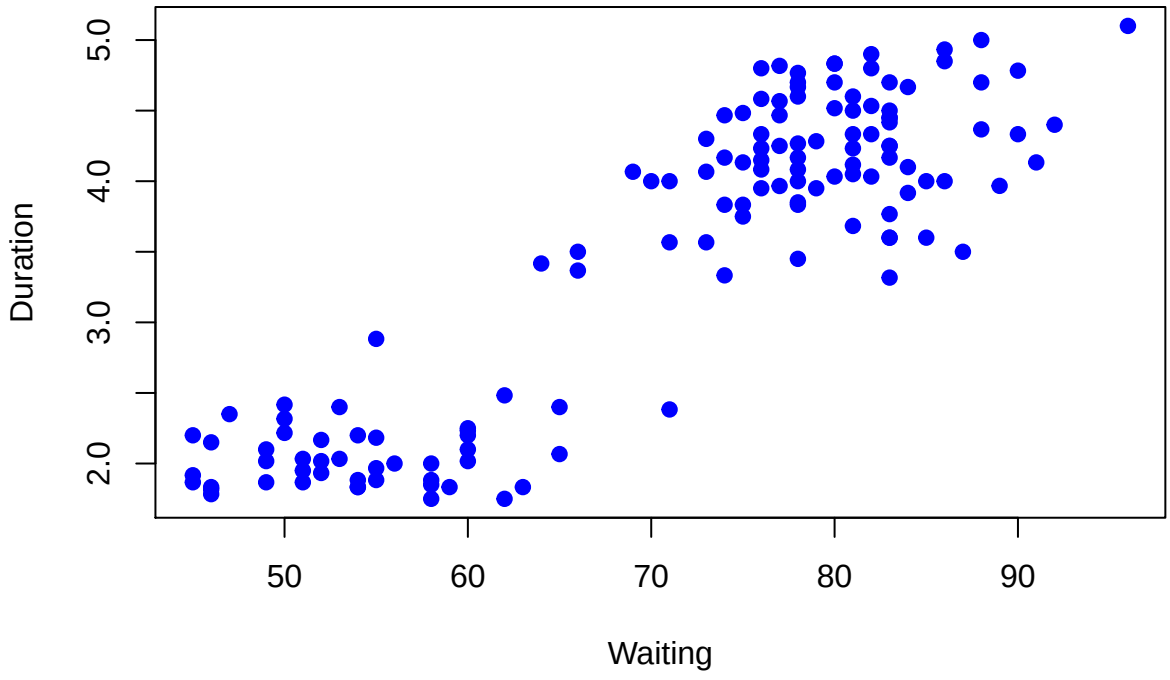
utilizando datos de Erupciones "Old faithful"

```
library(caret);data(faithful); set.seed(333)
inTrain <- createDataPartition(y=faithful$waiting,
                               p=0.5, list=FALSE)
trainFaith <- faithful[inTrain,]; testFaith <- faithful[-inTrain,]
head(trainFaith)
```

```
##    eruptions waiting
## 3      3.333      74
## 6      2.883      55
## 7      4.700      88
## 8      3.600      85
## 9      1.950      51
## 11     1.833      54
```

Eruption duration versus waiting time

```
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
```



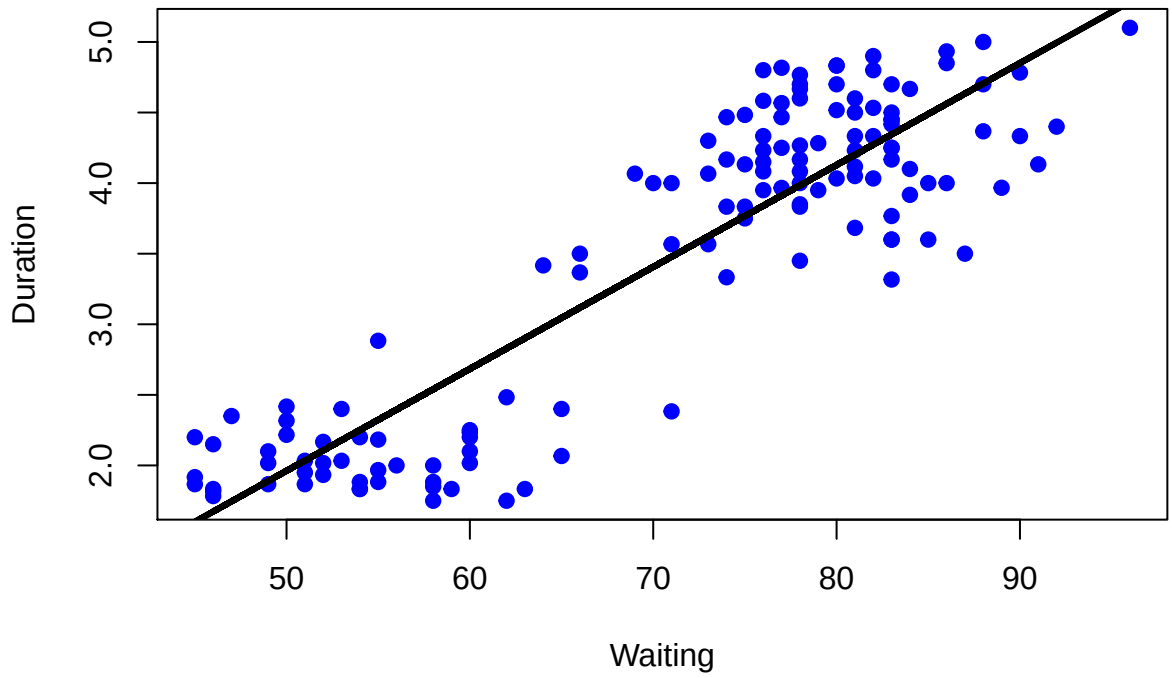
ajustando un modelo de la manera comun

```
lm1 <- lm(eruptions ~ waiting,data=trainFaith)
summary(lm1)
```

```
##
## Call:
```

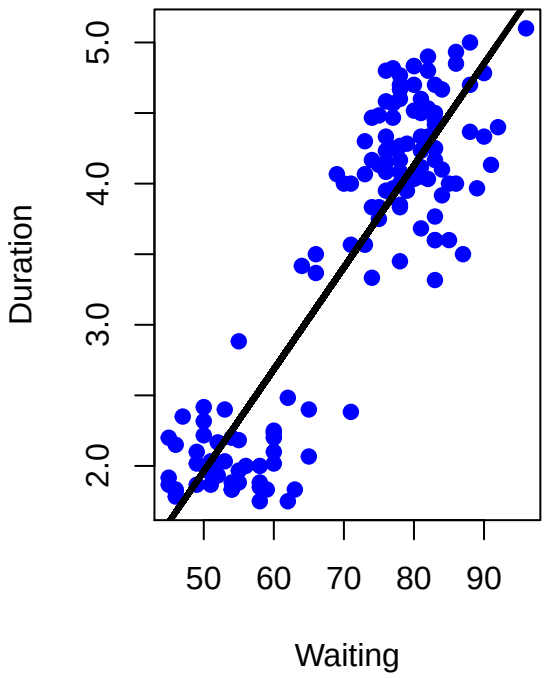
```
## lm(formula = eruptions ~ waiting, data = trainFaith)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13375 -0.36778  0.06064  0.36578  0.96057
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.648629   0.226603  -7.275 2.55e-11 ***
## waiting      0.072211   0.003136  23.026  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4941 on 135 degrees of freedom
## Multiple R-squared:  0.7971, Adjusted R-squared:  0.7956
## F-statistic: 530.2 on 1 and 135 DF,  p-value: < 2.2e-16
```

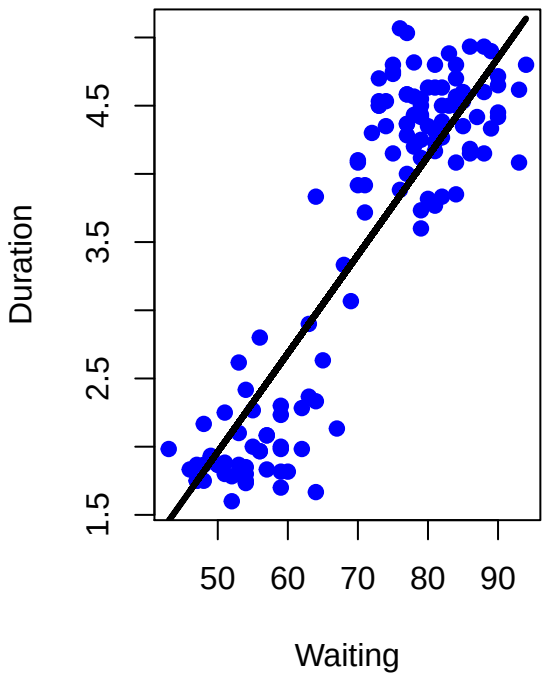
```
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(trainFaith$waiting,lm1$fitted,lwd=3)
```



modelando conjunto de entrenamiento y de prueba

```
par(mfrow=c(1,2))
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(trainFaith$waiting,predict(lm1),lwd=3)
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(testFaith$waiting,predict(lm1,newdata=testFaith),lwd=3)
```





Obtener errores de conjunto de entrenamiento / conjunto de prueba

```
# Calculate RMSE on training
sqrt(sum((lm1$fitted-trainFaith$eruptions)^2))
```

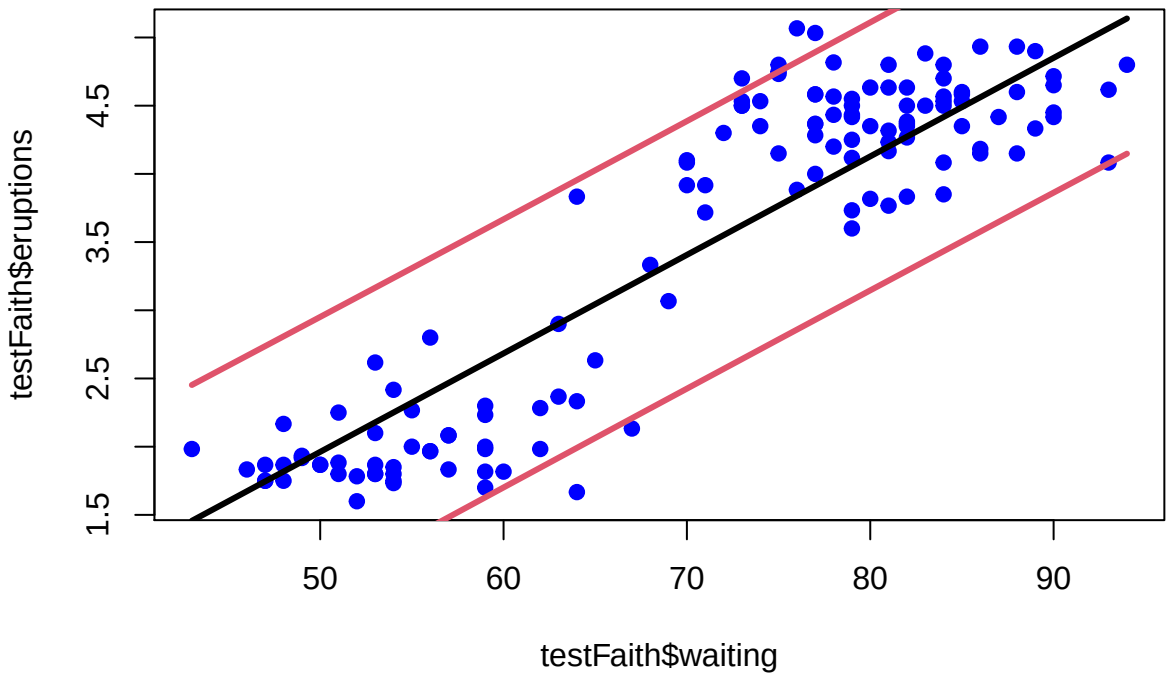
```
## [1] 5.740844
```

```
# Calculate RMSE on test
sqrt(sum((predict(lm1,newdata=testFaith)-testFaith$eruptions)^2))
```

```
## [1] 5.853745
```

creando intervalos

```
pred1 <- predict(lm1,newdata=testFaith,interval="prediction")
ord <- order(testFaith$waiting)
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue")
matlines(testFaith$waiting[ord],pred1[ord,],type="l",,col=c(1,2,2),lty = c(1,1,1), lwd=3)
```



lo mismo con caret

```
modFit <- train(eruptions ~ waiting,data=trainFaith,method="lm")
summary(modFit$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13375 -0.36778  0.06064  0.36578  0.96057
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.648629   0.226603  -7.275 2.55e-11 ***
## waiting      0.072211   0.003136  23.026  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4941 on 135 degrees of freedom
## Multiple R-squared:  0.7971, Adjusted R-squared:  0.7956
## F-statistic: 530.2 on 1 and 135 DF,  p-value: < 2.2e-16
```

Notas y lectura adicional

- Se pueden incluir modelos de regresión con múltiples covariables

- Suele ser útil en combinación con otros modelos
- [Elementos del aprendizaje estadístico] (http://www-stat.stanford.edu/~tibs/ElemStatLearn/)
- [Estadísticas aplicadas modernas con S] (http://www.amazon.com/Modern-Applied-Statistics-W-N-Venables/dp/0387954570)
- [Introducción al aprendizaje estadístico] (http://www-bcf.usc.edu/~gareth/ISL/)

prediciendo con regresion lineal multivariable

utilizaremos los datos de sueldo

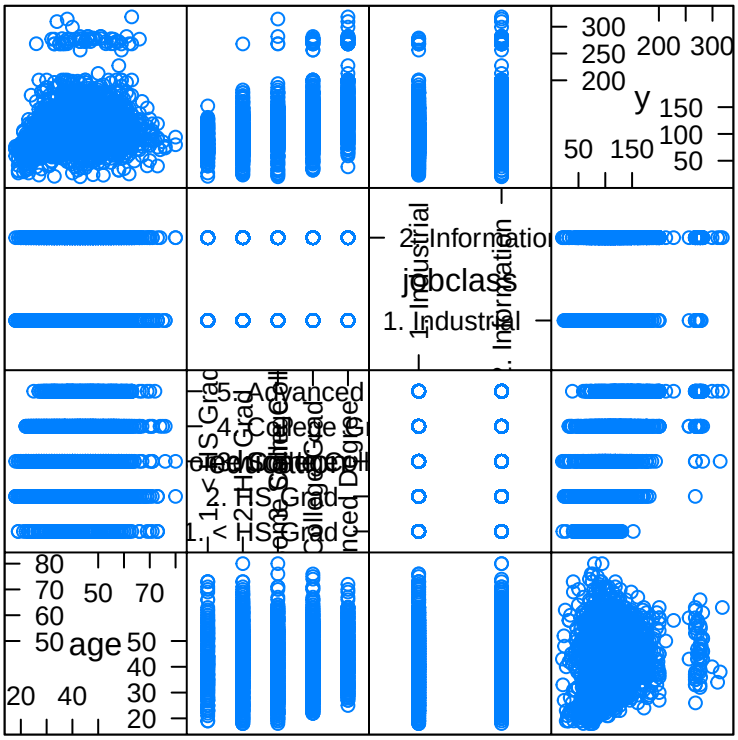
```
library(ISLR); library(ggplot2); library(caret);
data(Wage); Wage <- subset(Wage,select=-c(logwage))

inTrain <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
training <- Wage[inTrain,]; testing <- Wage[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 2102   10
```

```
## [1] 898  10
```

grafica de caracteristicas

```
featurePlot(x=training[,c("age","education","jobclass")],
            y = training$wage,
            plot="pairs")
```



```
qplot(age,wage,colour=education,data=training)
```

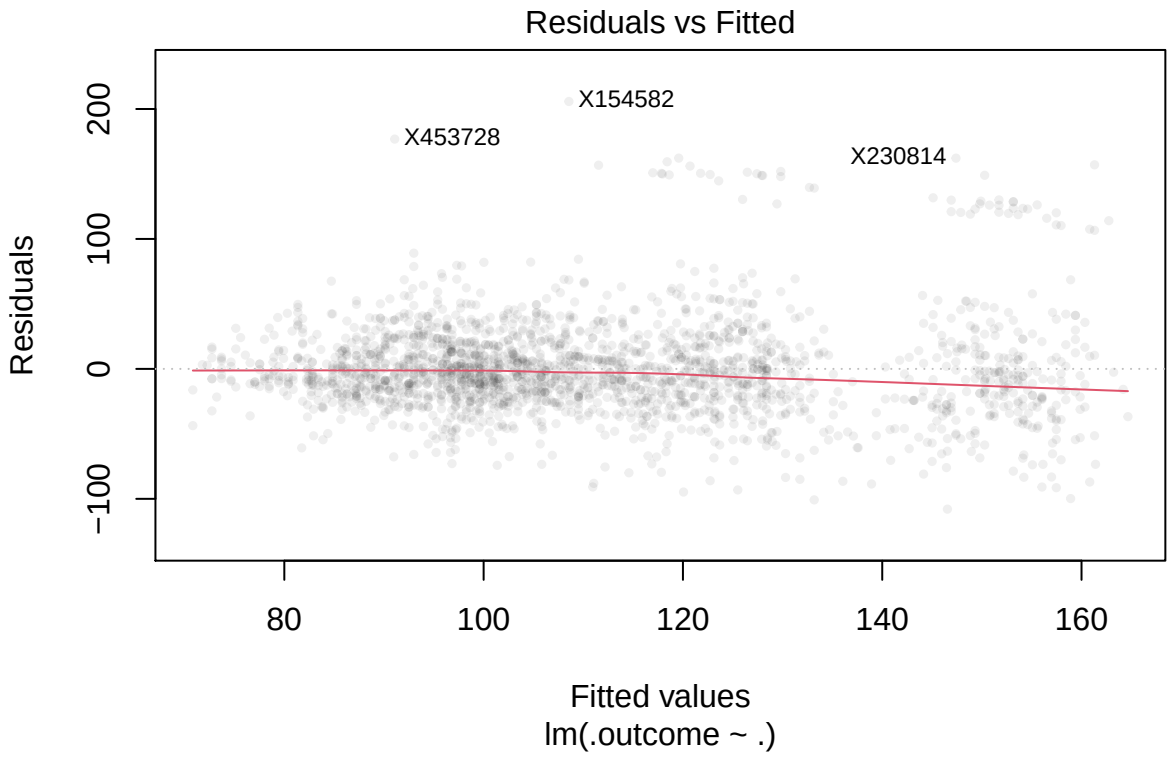
ajustando un modelo de regresion lineal con la funcion train

```
modFit<- train(wage ~ age + jobclass + education,
               method = "lm",data=training)
finMod <- modFit$finalModel
print(modFit)
```

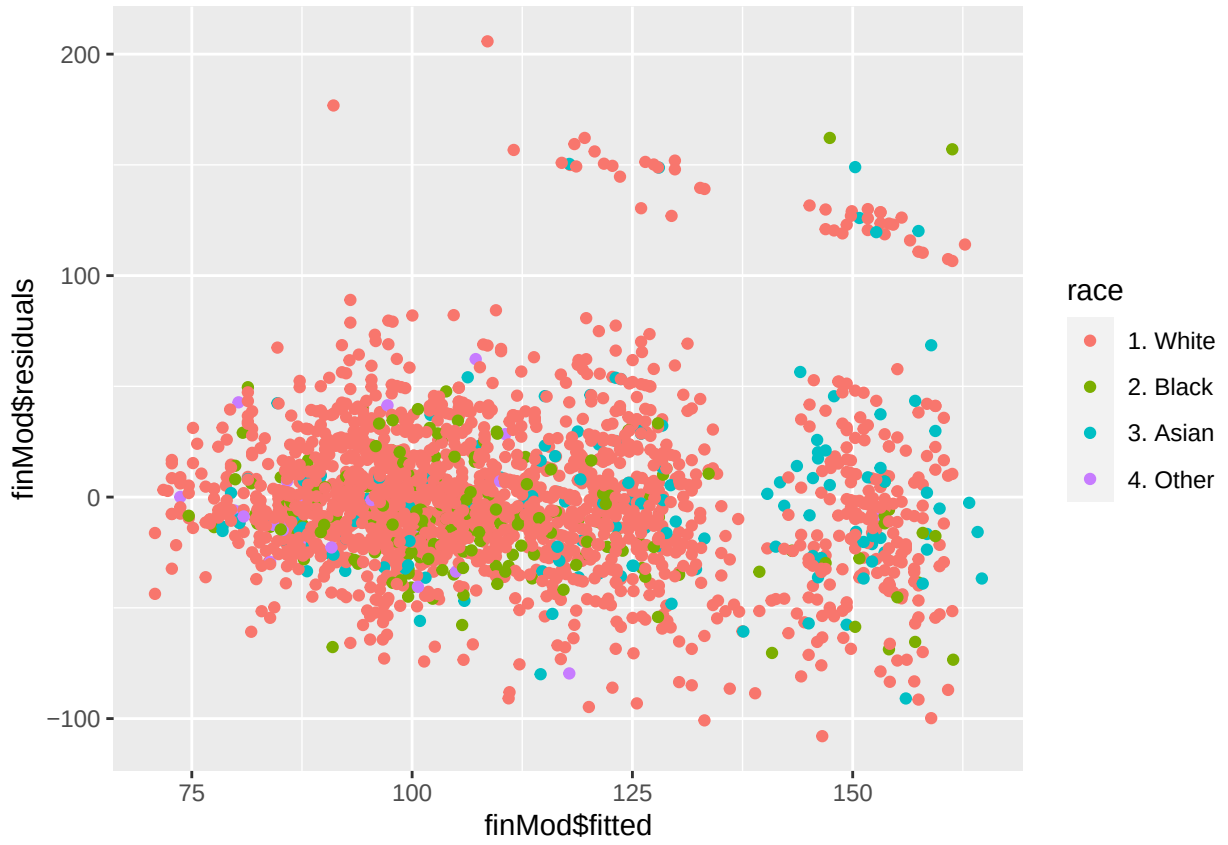
```
## Linear Regression
##
## 2102 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   35.56759  0.2589245  24.87554
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

diagnosticos

```
plot(finMod,1,pch=19,cex=0.5,col="#00000010")
```

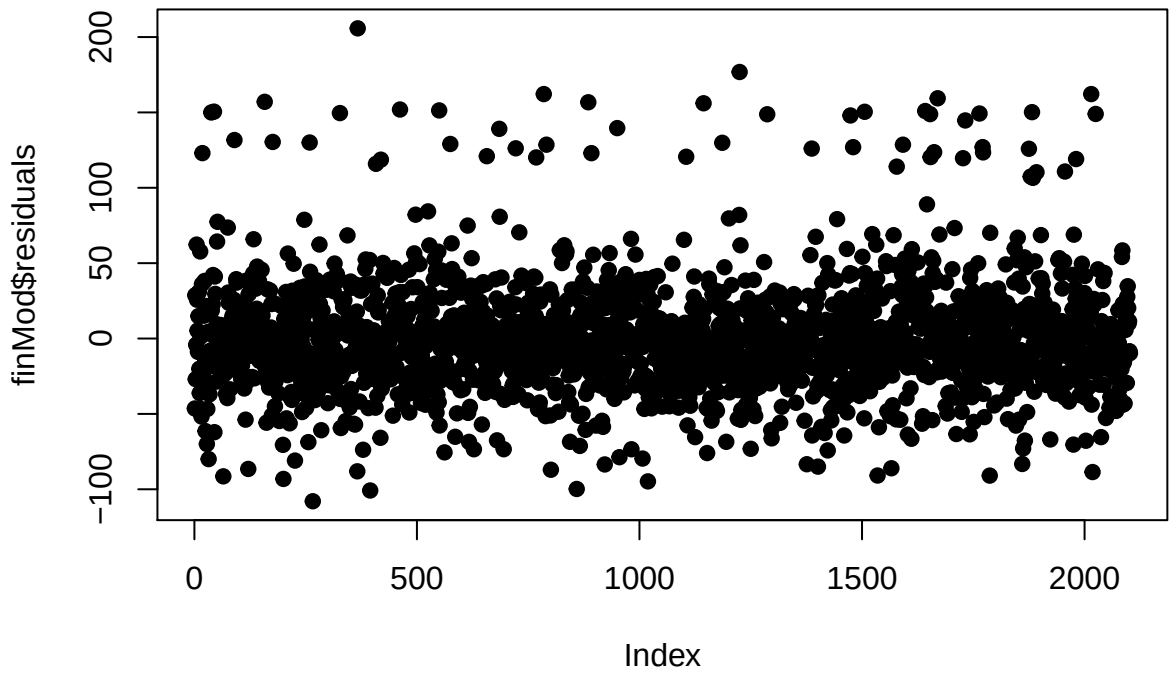



```
qplot(finMod$fitted,finMod$residuals,colour=race,data=training)
```



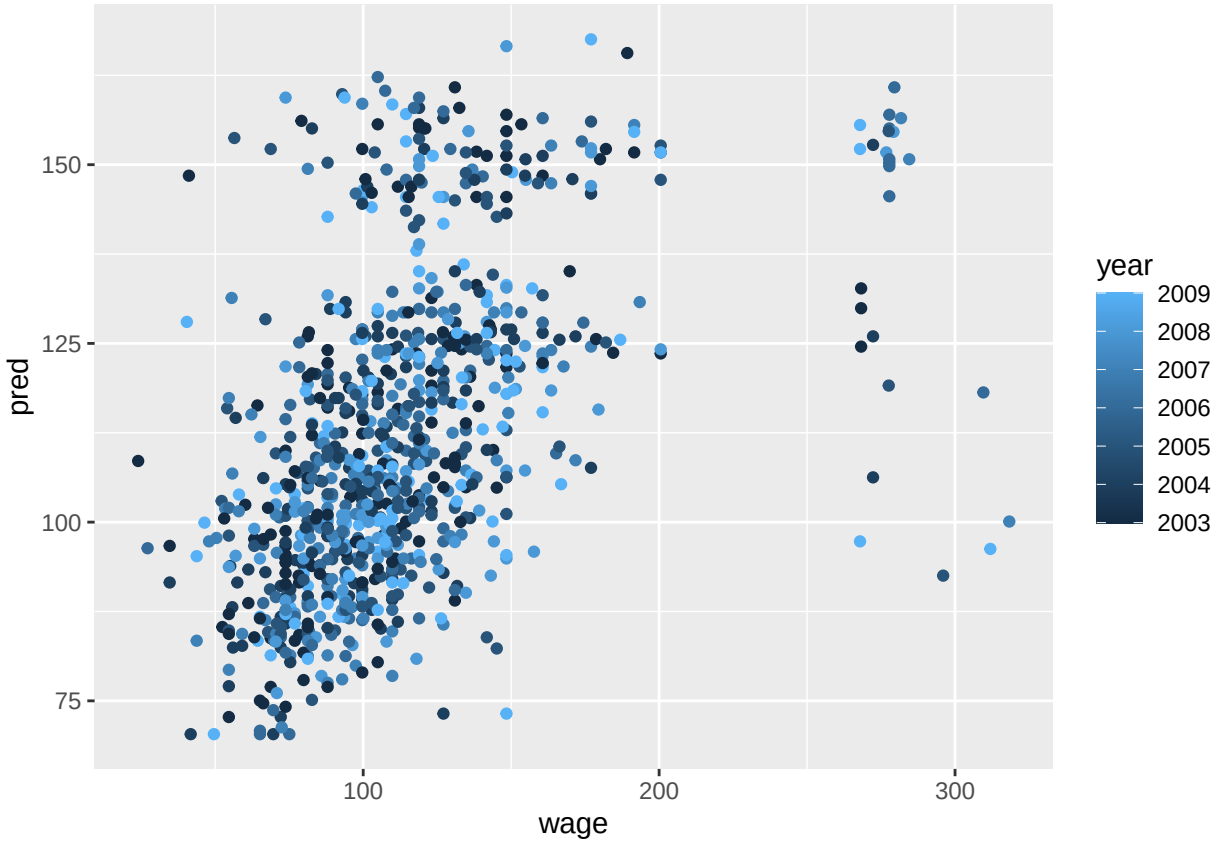
por indice

```
plot(finMod$residuals,pch=19)
```



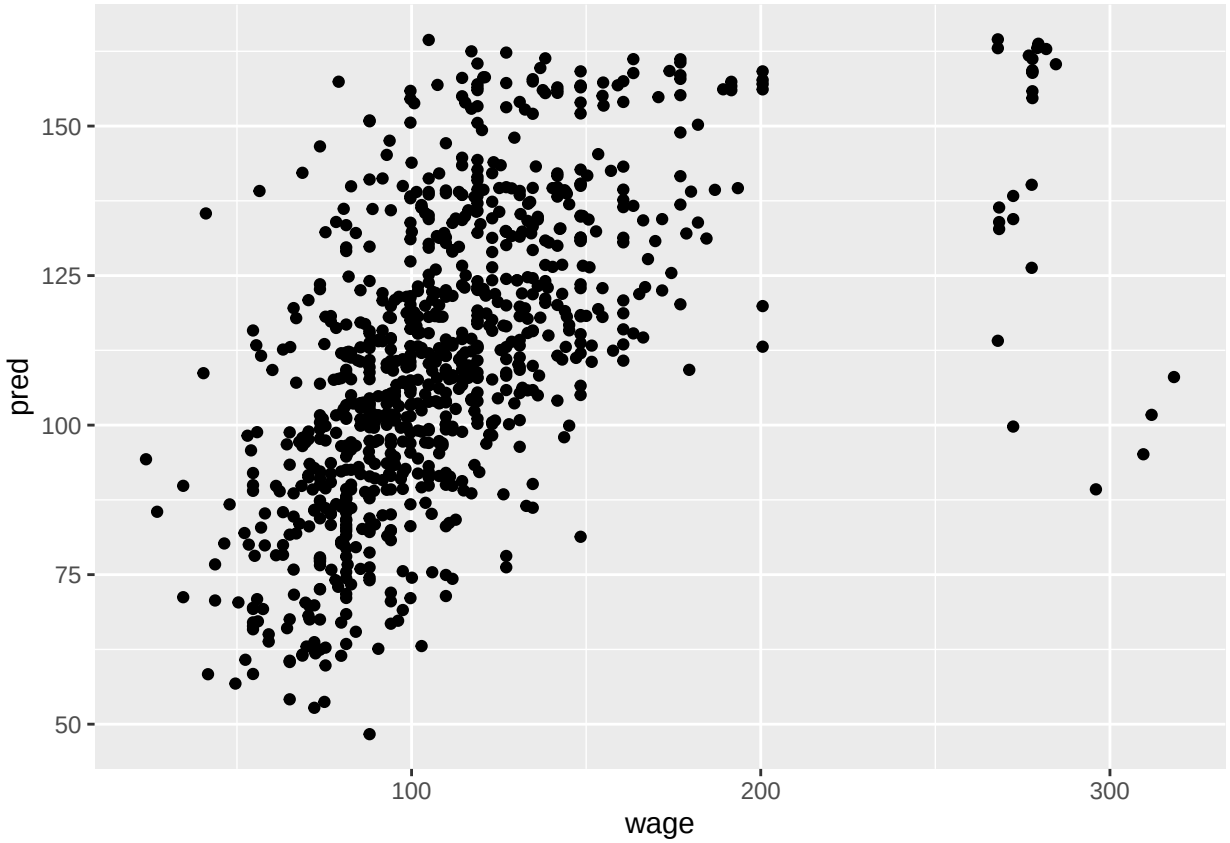
Predicción versus verdad en el conjunto de prueba

```
pred <- predict(modFit, testing)
qplot(wage,pred,colour=year,data=testing)
```



usando todas las covariables

```
modFitAll<- train(wage ~ .,data=training,method="lm")
pred <- predict(modFitAll, testing)
qplot(wage,pred,data=testing)
```



recursos

- Caret tutorials:
  - http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user_caret_2up.pdf
  - http://cran.r-project.org/web/packages/caret/vignettes/caret.pdf
- A paper introducing the caret package
  - http://www.jstatsoft.org/v28/i05/paper