paraTrabajo3.Rmd

Sylvia Acid

2 de mayo de 2017

*** El dataset Auto***

Description: Gas mileage, horsepower, and other information for cars.

```
#install.packages("ISLR")
library("ISLR", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.2")
data("Auto")   # loads the dataset
```

```
class(Auto)
```

```
## [1] "data.frame"
```

```
colnames(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"
## [5] "weight"       "acceleration" "year"         "origin"
## [9] "name"
```

```
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
```

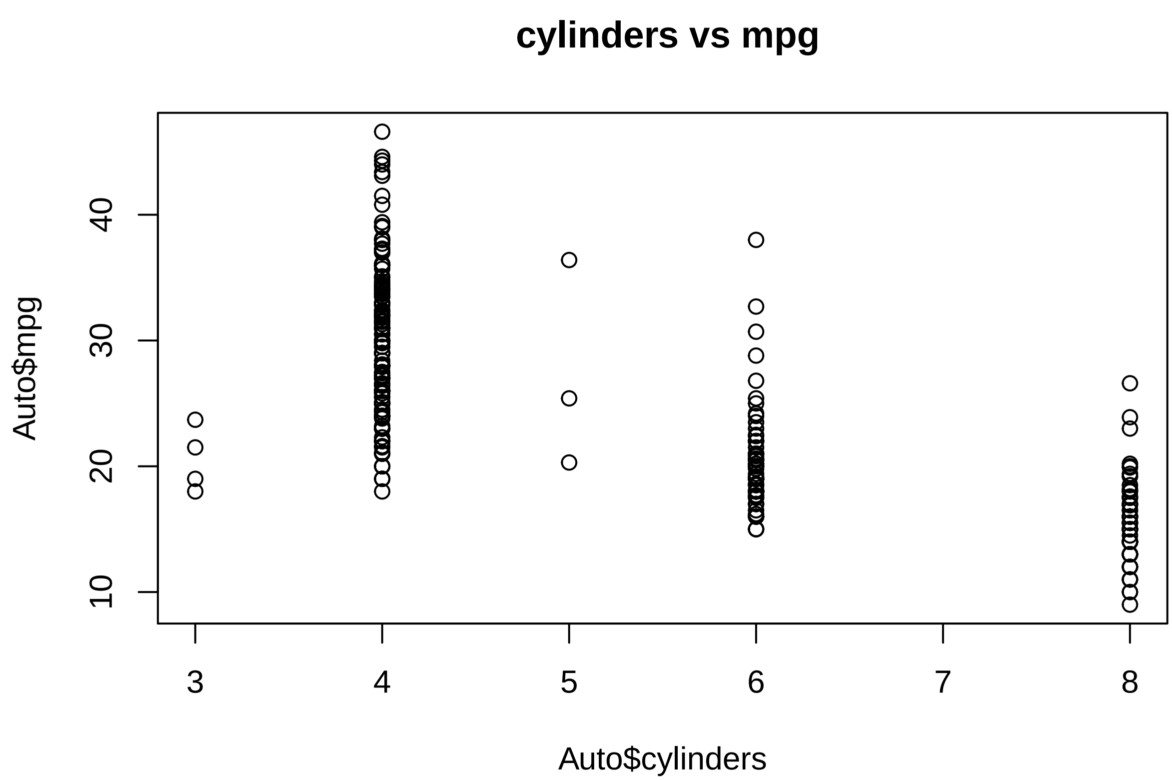
```
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##                 name
##  amc matador       :  5
##  ford pinto        :  5
##  toyota corolla    :  5
##  amc gremlin       :  4
##  amc hornet        :  4
##  chevrolet chevette:  4
##  (Other)           :365
```

```r
dim(Auto)
```

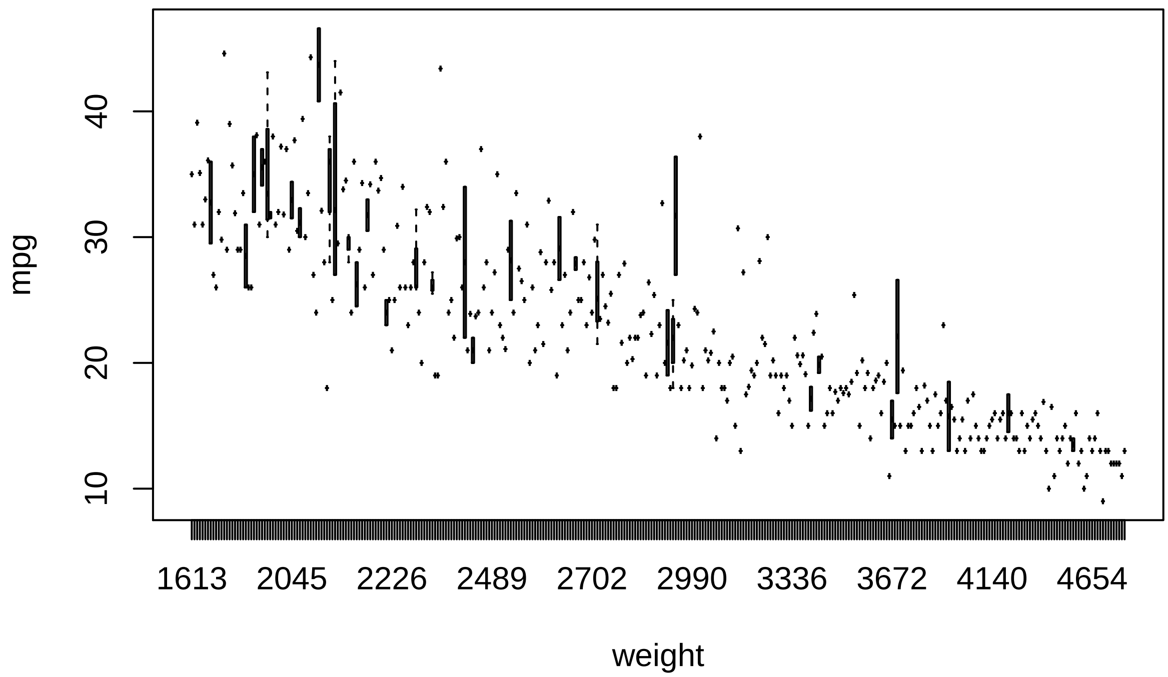
```
## [1] 392   9
```

Estamos interesados en el atributo *mpg*, vamos a tratar de visualizar por pares los atributos *mpg* y *cylinders* mediante los comandos *plot* and *boxplot*.

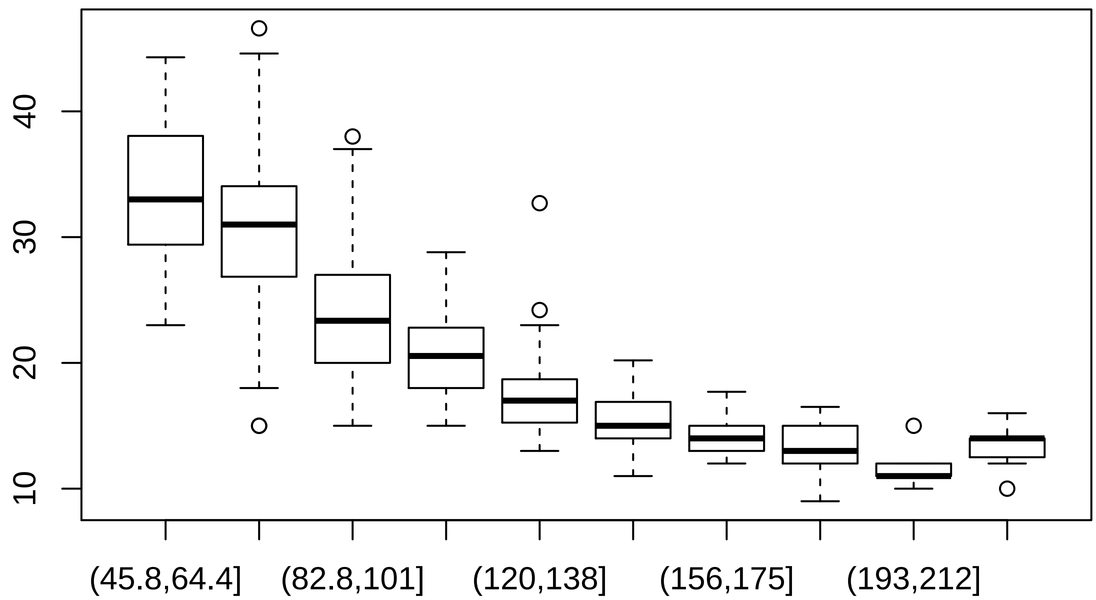
```r
plot(Auto$cylinders,Auto$mpg, main=" cylinders vs mpg")
```



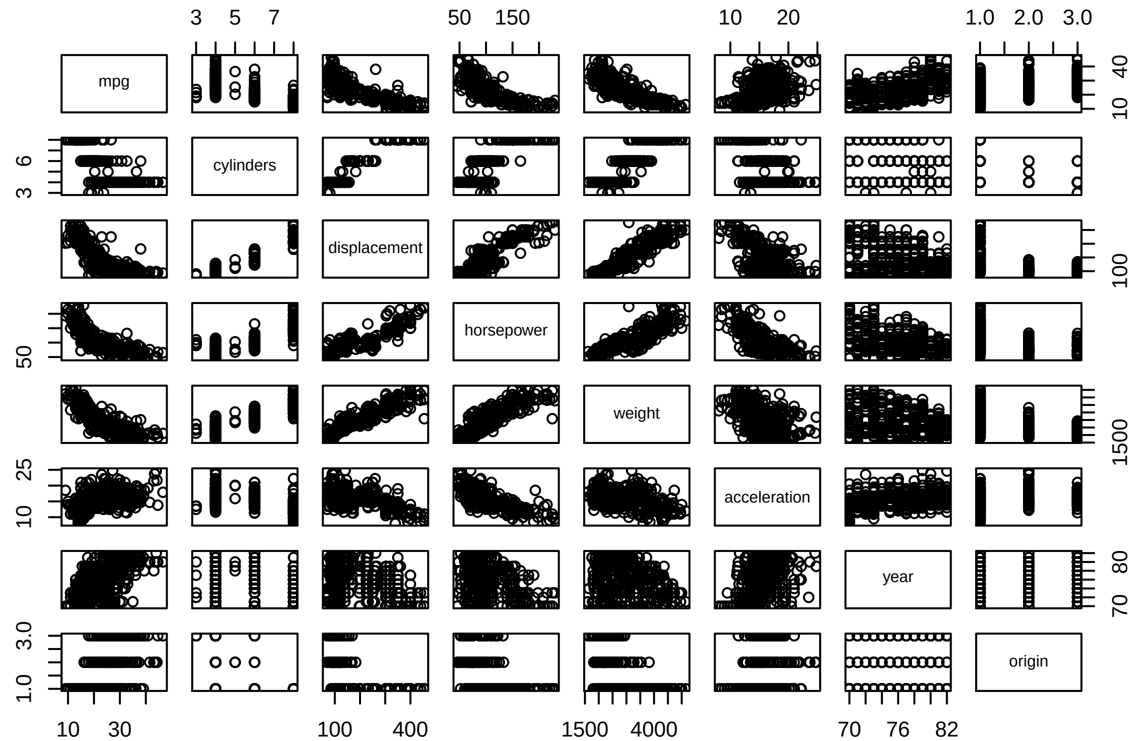

```r
boxplot(mpg~weight, data=Auto, xlab="weight", ylab = "mpg")
```




```
boxplot(mpg~cut(horsepower, breaks = 10),data = Auto)
```




```r
attach ( Auto )  # para simplificar y prescindir del prefijo Auto
#pairs(~ .,data = Auto)  # todos con todos
pairs(~ mpg + cylinders + displacement + horsepower + weight + acceleration + year + origin, data= Auto)
```




```
# solo algunas
```

Para evaluar los modelos, partimos el data.frame en training y test

```
set.seed(1)
train = sample (nrow(Auto), round(nrow(Auto)*0.7)) # nos quedamos con los indices para el training
auto.train = Auto[train,]  # podemos reservarlos aparte ... con subset no sería necesario
auto.test = Auto[-train,]
```

```
m1 = lm(mpg ~ weight, data=Auto, subset=train)
print(m1)
```

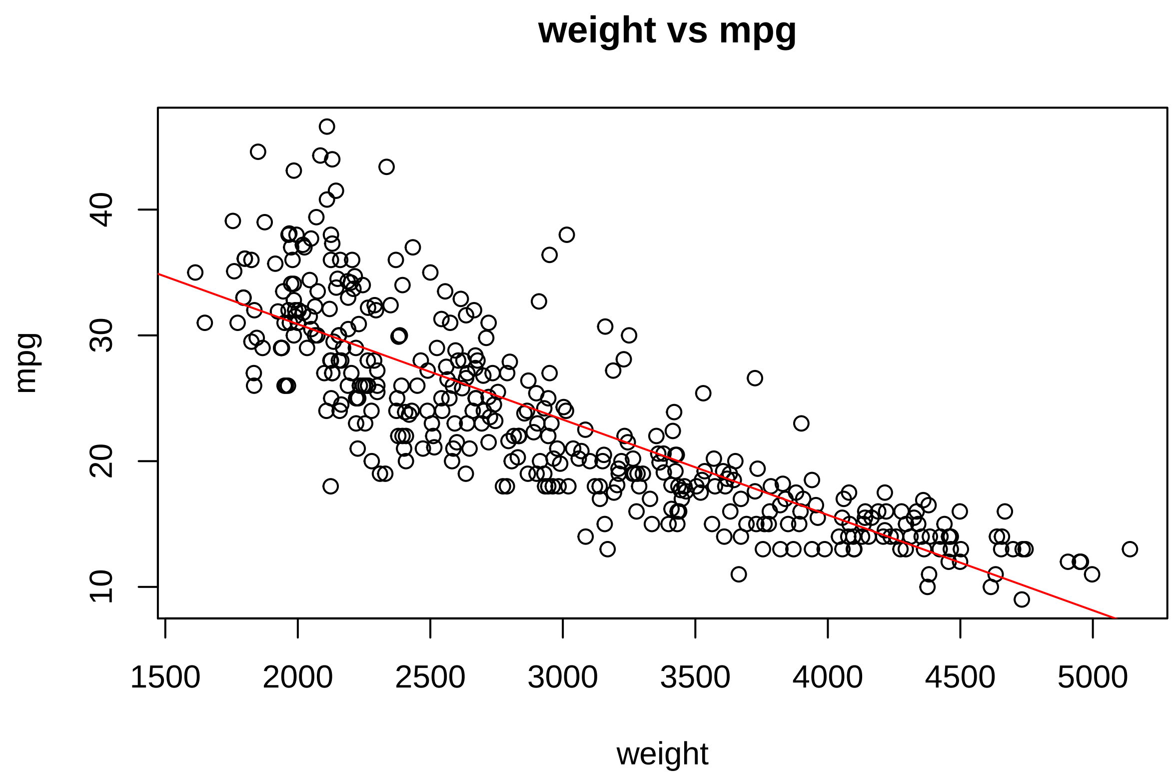
```
##
## Call:
## lm(formula = mpg ~ weight, data = Auto, subset = train)
##
## Coefficients:
## (Intercept)       weight
##   46.058884    -0.007585
```

```
summary(m1)
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = Auto, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

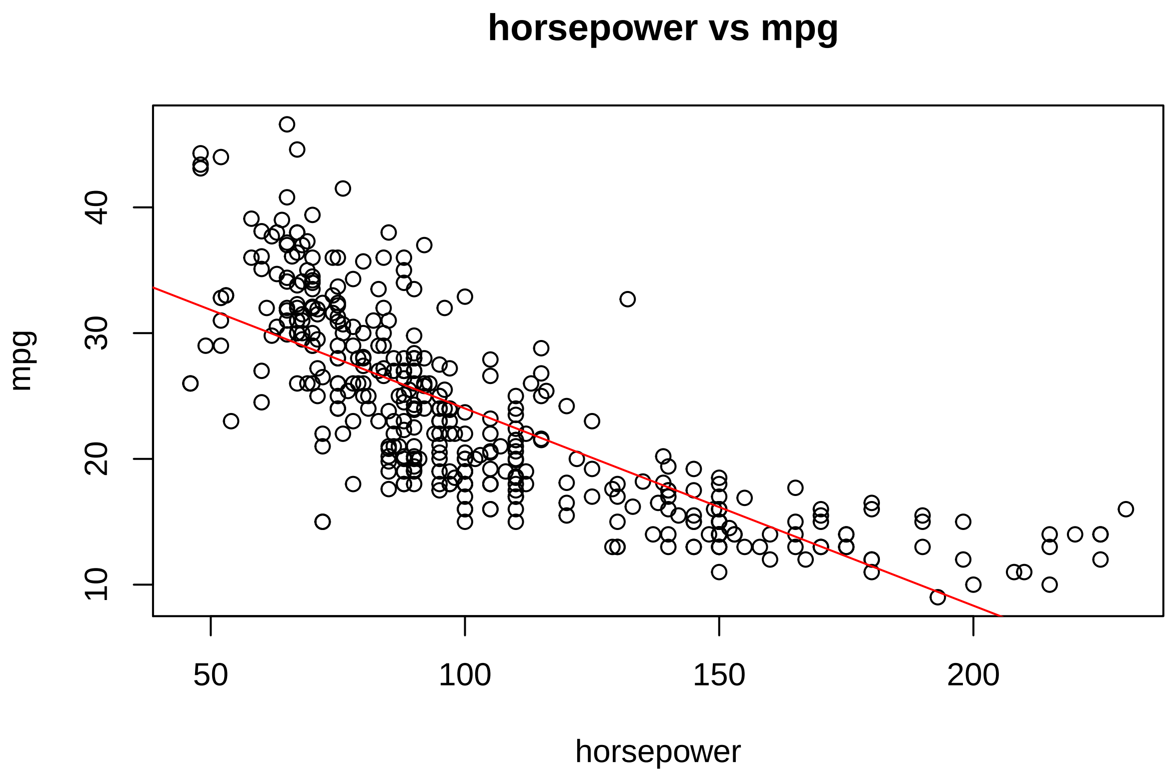
```
## -11.9477  -2.7053  -0.3457   2.2521  16.5461
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.0588840  0.9618282   47.89   <2e-16 ***
## weight      -0.0075853  0.0003078  -24.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.33 on 272 degrees of freedom
## Multiple R-squared:  0.6907, Adjusted R-squared:  0.6895
## F-statistic: 607.3 on 1 and 272 DF,  p-value: < 2.2e-16
```

```r
plot(weight, mpg, main=" weight vs mpg")
abline(m1$coefficients, col=2)
```



m1, nuestro primer modelo

```r
m2 = lm(mpg ~ horsepower, data=Auto, subset=train)
plot(horsepower, mpg, main=" horsepower vs mpg")
abline(m2$coefficients, col=2)
```




```r
summary(m2)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.3988  -3.1685  -0.1685   2.9242  17.1036
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.688412   0.849380   46.73   <2e-16 ***
## horsepower  -0.156800   0.007602  -20.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.862 on 272 degrees of freedom
## Multiple R-squared:   0.61,  Adjusted R-squared:  0.6086
## F-statistic: 425.4 on 1 and 272 DF,  p-value: < 2.2e-16
```

```r
m3 = lm(mpg ~ ., data=Auto, subset=train) # en función del resto, de TODOS
#coef(m3)


m4 = lm(mpg ~ weight + horsepower + displacement, data=Auto, subset=train)
summary(m4)
```

```
##
## Call:
## lm(formula = mpg ~ weight + horsepower + displacement, data = Auto,
##     subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.2340  -2.7069  -0.3418   2.2375  16.3002
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  44.5516734  1.4316517  31.119  < 2e-16 ***
## weight       -0.0051554  0.0008627  -5.976  7.2e-09 ***
## horsepower   -0.0437096  0.0150185  -2.910  0.00391 **
## displacement -0.0061969  0.0078486  -0.790  0.43048
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.233 on 270 degrees of freedom
## Multiple R-squared:  0.7065, Adjusted R-squared:  0.7032
## F-statistic: 216.6 on 3 and 270 DF,  p-value: < 2.2e-16
```

Qué funciones se pueden aplicar sobre un modelo, como m4?

```
methods(class=class(m4))
```

```
##  [1] add1           alias          anova          case.names
##  [5] coerce         confint        cooks.distance deviance
##  [9] dfbeta         dfbetas        drop1          dummy.coef
## [13] effects        extractAIC     family         formula
## [17] hatvalues      influence      initialize     kappa
## [21] labels         logLik         model.frame    model.matrix
## [25] nobs           plot           predict        print
## [29] proj           qr             residuals      rstandard
## [33] rstudent       show           simulate       slotsFromS3
## [37] summary        variable.names vcov
## see '?methods' for accessing help and source code
```

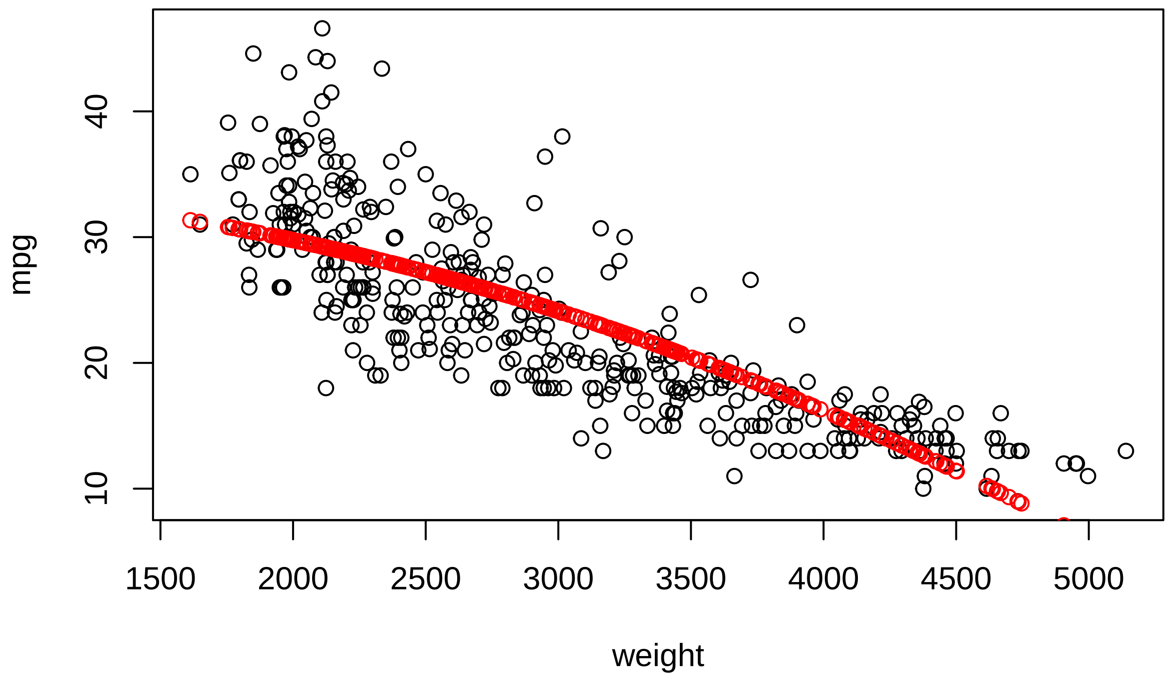
De las gráficas anteriores parece que las relaciones observadas no son lineales …

Habrá que incorporar algún tipo de transformación no lineal de los atributos … Por ejemplo, una forma cuadrática

```
m5 = lm(mpg ~ I(weight^2), data=Auto, subset=train)
coef(m5)
```

```
##   (Intercept)   I(weight^2)
##  3.428395e+01 -1.130048e-06
```

```
plot(mpg~weight)
w= m5$coefficients
x = matrix(rep(1, length(weight)),nrow= length(weight))
x= cbind (x, weight^2)
y= apply(x, 1, function(vec) w %*% vec)
points(weight, y, col=2)
```



Con los modelos, podemos obtener predicciones

```
yhatm1Tr = predict(m1) # usa el propio training
yhatm1Tst = predict(m1, auto.test, type= "response")

etr = mean((yhatm1Tr - auto.train[,1])^2)
etst = mean((yhatm1Tst - auto.test[,1])^2)
```

Para ver otras transformaciones p.ej. cúbicas etc.. consultar *poly*(), *log*()

Clasificación

Vamos a convertir el problema en un problema de clasificación binaria Se crea una variable binaria, mpg01

```
Auto2 = data.frame(mpg01 = (ifelse(mpg<median(mpg),0,1)),Auto)
```

Particionar el conjunto en training y test

Se ajusta un modelo lineal, por ejemplo de regresión logística para predecir *mpg*01. Se puede especificar de forma explícita los atributos a considerar a la hora de construir el modelo, el resto se ignoran.

```
ml1 = glm(mpg01 ~ weight + horsepower + displacement,
  family = binomial(logit), data = Auto2, subset=train)
summary(ml1)
```

```
##
## Call:
## glm(formula = mpg01 ~ weight + horsepower + displacement, family = binomial(logit),
##     data = Auto2, subset = train)
##
```

```
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.31258  -0.28359  -0.00467   0.39442   3.13796
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  11.6385137  1.8444083   6.310 2.79e-10 ***
## weight       -0.0020599  0.0008214  -2.508   0.0122 *
## horsepower   -0.0454861  0.0151603  -3.000   0.0027 **
## displacement -0.0081012  0.0060085  -1.348   0.1776
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 379.79  on 273  degrees of freedom
## Residual deviance: 158.16  on 270  degrees of freedom
## AIC: 166.16
##
## Number of Fisher Scoring iterations: 7
```

Una vez aprendido, veamos cómo predice…

```
#Calculo de probabilidades
probTr.ml1 = predict(ml1, type="response")
probTstml1 = predict(ml1, data.frame(Auto2[-train,-1]), type="response")
```

predicciones con el modelo de regresión logística

```
predTstml1 = rep(0, length(probTstml1))  # predicciones por defecto 0
predTstml1[probTstml1 >=0.5] = 1          # >= 0.5 clase 1

table(predTstml1, Auto2[-train,1])  # para el calculo del Eval
```

```
##
## predTstml1  0  1
##          0 50  3
##          1  7 58
```

```
Eval = mean(predTstml1 != Auto2[-train,1])
cat("Eval con el modelo LR "); print(ml1$call)
```

```
## Eval con el modelo LR
```

```
## glm(formula = mpg01 ~ weight + horsepower + displacement, family = binomial(logit),
##     data = Auto2, subset = train)
```

```
print(Eval)
```

```
## [1] 0.08474576
```

se obtiene el Etest, para obtener el Ein?

Otras familias de funciones …

```
ml2 = glm(mpg01 ~ weight + horsepower + displacement,
  family = gaussian(identity), data = Auto2, subset=train)
summary(ml2)
```

```
##
```

```
## Call:
## glm(formula = mpg01 ~ weight + horsepower + displacement, family = gaussian(identity),
##     data = Auto2, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9242  -0.2369   0.0808   0.2052   0.9833
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.567e+00  1.131e-01  13.847  < 2e-16 ***
## weight       -2.641e-04  6.817e-05  -3.875 0.000134 ***
## horsepower    3.488e-04  1.187e-03   0.294 0.769080
## displacement -1.609e-03  6.202e-04  -2.595 0.009977 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1119079)
##
##     Null deviance: 68.485  on 273  degrees of freedom
## Residual deviance: 30.215  on 270  degrees of freedom
## AIC: 183.47
##
## Number of Fisher Scoring iterations: 2
```

A la hora de comparar clasificadores, gráficamente se muestra por la curva ROC Es mejor clasificador, cuanto mayor sea el área debajo de la curva.

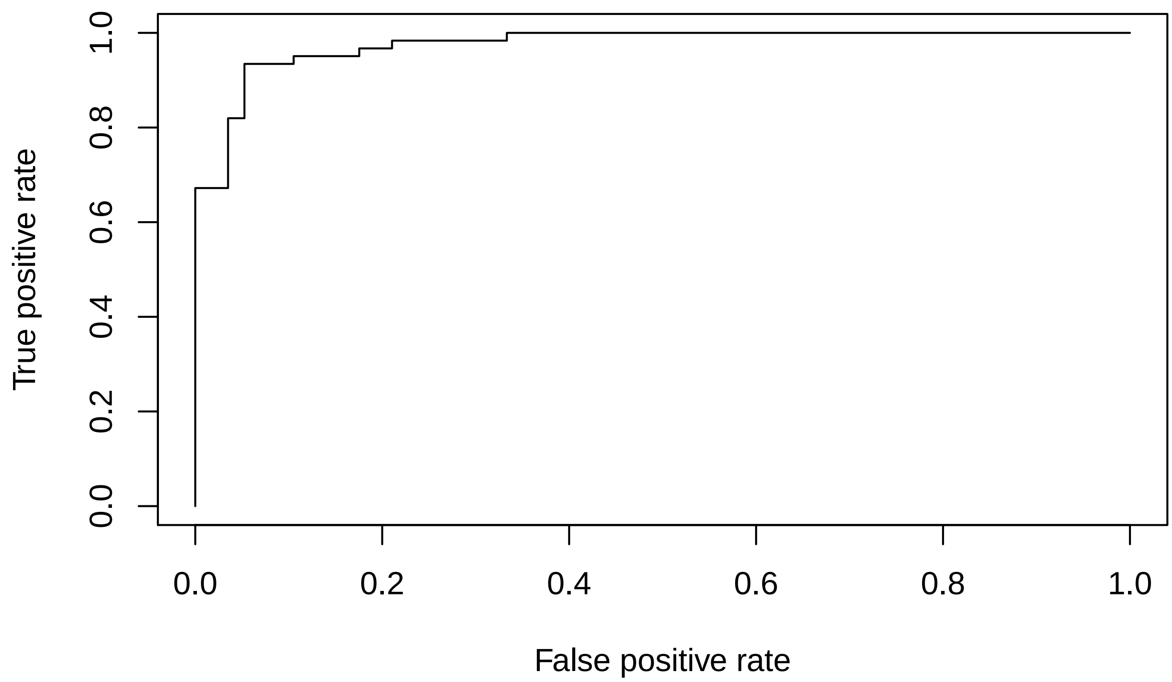
```
#install.packages("ROCR")
library("ROCR", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.2")
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
pred = prediction(probTstml1,Auto2[-train,1])
perf = performance(pred,"tpr","fpr")
plot(perf) # pinta la curva
```



Para el preprocesamiento Centrado, escalado, transformación para reducir la asimetria Vamos a trabajar con el dataset segmentationOriginal que trata de
Cell Body Segmentation problema de clasificación , células Pobremente segmentadas o Well segmentadas.

```
library("AppliedPredictiveModeling", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.3")
library(help=AppliedPredictiveModeling)
data("segmentationOriginal")
class(segmentationOriginal)
```

```
## [1] "data.frame"
```

```
names(segmentationOriginal)[1:10]
```

```
##  [1] "Cell"           "Case"           "Class"          "AngleCh1"
##  [5] "AngleStatusCh1" "AreaCh1"        "AreaStatusCh1"  "AvgIntenCh1"
##  [9] "AvgIntenCh2"    "AvgIntenCh3"
```

```
summary(segmentationOriginal[1:10])
```

```
##       Cell               Case       Class        AngleCh1
##  Min.   :207827637   Test :1010   PS:1300   Min.   :  0.03088
##  1st Qu.:208332462   Train:1009   WS: 719   1st Qu.: 53.89221
##  Median :208384321                          Median : 90.58877
##  Mean   :208402392                          Mean   : 90.49340
##  3rd Qu.:208405230                          3rd Qu.:126.68201
##  Max.   :210964110                          Max.   :179.93932
##  AngleStatusCh1      AreaCh1        AreaStatusCh1      AvgIntenCh1
##  Min.   :0.0000   Min.   : 150.0   Min.   :0.00000   Min.   :  15.16
```

```
##  1st Qu.:0.0000   1st Qu.: 193.0   1st Qu.:0.00000   1st Qu.:  35.36
##  Median :0.0000   Median : 253.0   Median :0.00000   Median :  62.34
##  Mean   :0.5686   Mean   : 320.3   Mean   :0.08024   Mean   : 126.07
##  3rd Qu.:1.0000   3rd Qu.: 362.5   3rd Qu.:0.00000   3rd Qu.: 143.19
##  Max.   :2.0000   Max.   :2186.0   Max.   :1.00000   Max.   :1418.63
##   AvgIntenCh2     AvgIntenCh3
##  Min.   :  0.0   Min.   :   0.12
##  1st Qu.: 44.0   1st Qu.:  33.50
##  Median :172.5   Median :  67.43
##  Mean   :188.1   Mean   :  96.42
##  3rd Qu.:278.3   3rd Qu.: 127.34
##  Max.   :988.5   Max.   :1205.51
```

```r
cellcase = segmentationOriginal$Case
unique(cellcase)
```

```
## [1] Test  Train
## Levels: Test Train
```

```r
segData.tr = subset(segmentationOriginal, Case == "Train")
dim(segData.tr)
```

```
## [1] 1009  119
```

```r
dim(segmentationOriginal)
```

```
## [1] 2019  119
```

```r
cellClass = segData.tr$Class
unique(cellClass)
```

```
## [1] PS WS
## Levels: PS WS
```

```r
cellID = segData.tr$Cell
length(unique(cellID))
```

```
## [1] 1009
```

```r
segData.tr = segData.tr[, -c(1:3)]  # eliminadas los 3 primeras atributos
```

Se eliminan parte de la información, columnas redundantes … Todas aquellas que contengan status …

```r
length(grep("Status", names(segData.tr)))
```

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

```r
b = (grep("Status", names(segData.tr)))
segData.tr = segData.tr[,-b]
dim(segData.tr)
```

```
## [1] 1009   58
```

```r
names(segData.tr)
```

```
##  [1] "AngleCh1"                "AreaCh1"
##  [3] "AvgIntenCh1"             "AvgIntenCh2"
##  [5] "AvgIntenCh3"             "AvgIntenCh4"
##  [7] "ConvexHullAreaRatioCh1"  "ConvexHullPerimRatioCh1"
##  [9] "DiffIntenDensityCh1"     "DiffIntenDensityCh3"
## [11] "DiffIntenDensityCh4"     "EntropyIntenCh1"
```

```
## [13] "EntropyIntenCh3"          "EntropyIntenCh4"
## [15] "EqCircDiamCh1"            "EqEllipseLWRCh1"
## [17] "EqEllipseOblateVolCh1"    "EqEllipseProlateVolCh1"
## [19] "EqSphereAreaCh1"          "EqSphereVolCh1"
## [21] "FiberAlign2Ch3"           "FiberAlign2Ch4"
## [23] "FiberLengthCh1"           "FiberWidthCh1"
## [25] "IntenCoocASMCh3"          "IntenCoocASMCh4"
## [27] "IntenCoocContrastCh3"     "IntenCoocContrastCh4"
## [29] "IntenCoocEntropyCh3"      "IntenCoocEntropyCh4"
## [31] "IntenCoocMaxCh3"          "IntenCoocMaxCh4"
## [33] "KurtIntenCh1"             "KurtIntenCh3"
## [35] "KurtIntenCh4"             "LengthCh1"
## [37] "NeighborAvgDistCh1"       "NeighborMinDistCh1"
## [39] "NeighborVarDistCh1"       "PerimCh1"
## [41] "ShapeBFRCh1"              "ShapeLWRCh1"
## [43] "ShapeP2ACh1"              "SkewIntenCh1"
## [45] "SkewIntenCh3"             "SkewIntenCh4"
## [47] "SpotFiberCountCh3"        "SpotFiberCountCh4"
## [49] "TotalIntenCh1"            "TotalIntenCh2"
## [51] "TotalIntenCh3"            "TotalIntenCh4"
## [53] "VarIntenCh1"              "VarIntenCh3"
## [55] "VarIntenCh4"              "WidthCh1"
## [57] "XCentroid"                "YCentroid"
```

Transformación de atributos asimétricos, necesarios para la aplicación de algunos métodos de aprendizaje sensibles a distancias. Se consideran asimétricos cuando o bien la ratio entre min y max de $range() > 20$ o bien el valor skewness se aleja de 0.

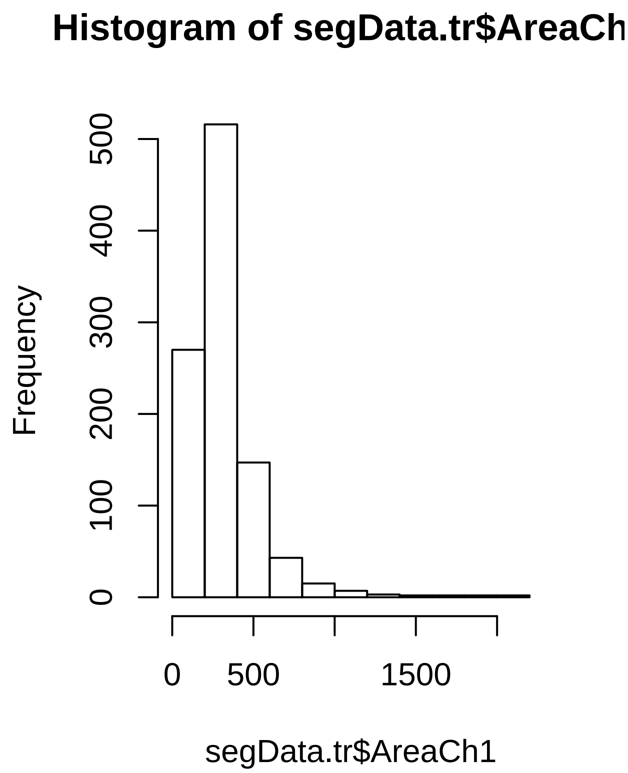
$$skewness = \frac{\sum(x_i - mean(x))^3}{(n-1)v^3/2}$$

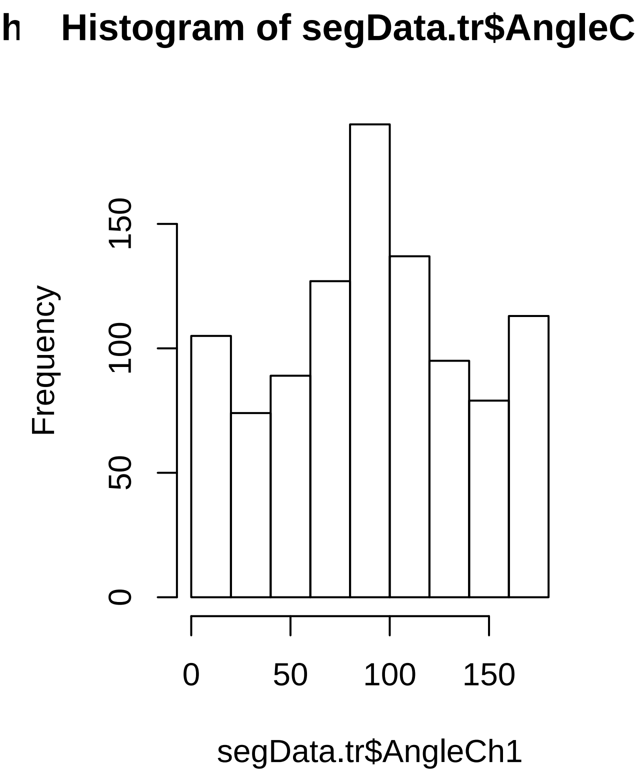
donde v es la varianza. Para verlo:

```r
par(mfrow=c(1,2))
hist (segData.tr$AreaCh1)
range(segData.tr$AreaCh1)
```

```
## [1]  150 2186
```

```r
hist(segData.tr$AngleCh1)
```






```
range(segData.tr$AngleCh1)
```

```
## [1]   0.03087639 179.93932283
```

```
par(mfrow=c(1,1))
```

Una función que la mide

```
library("e1071", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.3")
?skewness
skewness(segData.tr$AreaCh1)
```

```
## [1] 3.525107
```

```
skewness(segData.tr$AngleCh1)
```

```
## [1] -0.02426252
```

Se puede observar las que lo requieren:

```
v_asimetria = apply(segData.tr,2,skewness)
v_asimetria[1:15]
```

```
##               AngleCh1                AreaCh1             AvgIntenCh1
##            -0.02426252             3.52510745              2.95918524
##            AvgIntenCh2            AvgIntenCh3             AvgIntenCh4
##             0.84816033             2.20234214              1.90047128
##  ConvexHullAreaRatioCh1 ConvexHullPerimRatioCh1     DiffIntenDensityCh1
##             2.47658194            -1.30409896              2.76047338
##    DiffIntenDensityCh3     DiffIntenDensityCh4          EntropyIntenCh1
```

```
##              2.08518782              1.89923287              0.39789483
##          EntropyIntenCh3         EntropyIntenCh4            EqCircDiamCh1
##             -1.00295336             -0.82790492              1.95553035
```

```r
sort(abs(v_asimetria), decreasing = T)[1:10]
```

```
##            KurtIntenCh1            KurtIntenCh4 EqEllipseProlateVolCh1
##               12.859648                6.918503               6.070834
##          EqSphereVolCh1            KurtIntenCh3  EqEllipseOblateVolCh1
##                5.739502                5.505611               5.489313
##           TotalIntenCh1         EqSphereAreaCh1                AreaCh1
##                5.399604                3.525140               3.525107
##    IntenCoocContrastCh4
##                3.470305
```

Se quiere aplicar funciones sobre los datos para eliminar dicha asimetría para un trato homogéneo de todas los atributos.

Una familia de funciones para la transformación (que incluye desde cuadráticas, raíces, inversas etc.. son las propuestas por Box y Cox (1964) un parámetro como λ:

$\frac{x^{\lambda}-1}{\lambda}$ si $\lambda \neq 0$

o bien

$log(x)$ si $\lambda = 0$

Con *skewness()* lo detecta pero cuál es la transformación, para ello:

```r
#install.packages("caret")
library("caret", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.3")
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'Auto':
##
##     mpg
```

```r
BoxCoxTrans(segData.tr$AngleCh1)  # no transformacion
```

```
## Box-Cox Transformation
##
## 1009 data points used to estimate Lambda
##
## Input data summary:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##   0.03088  54.66000  90.03000  91.13000 127.90000 179.90000
##
## Largest/Smallest: 5830
## Sample Skewness: -0.0243
##
## Estimated Lambda: 0.8
## With fudge factor, no transformation is applied
```

```r
BoxCoxTrans(segData.tr$AreaCh1)
```

```
## Box-Cox Transformation
##
## 1009 data points used to estimate Lambda
##
## Input data summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   150.0   194.0   256.0   325.1   376.0  2186.0
##
## Largest/Smallest: 14.6
## Sample Skewness: 3.53
##
## Estimated Lambda: -0.9
```

```r
Ch1Area_trans = BoxCoxTrans(segData.tr$AreaCh1)

head(segData.tr$AreaCh1)
```

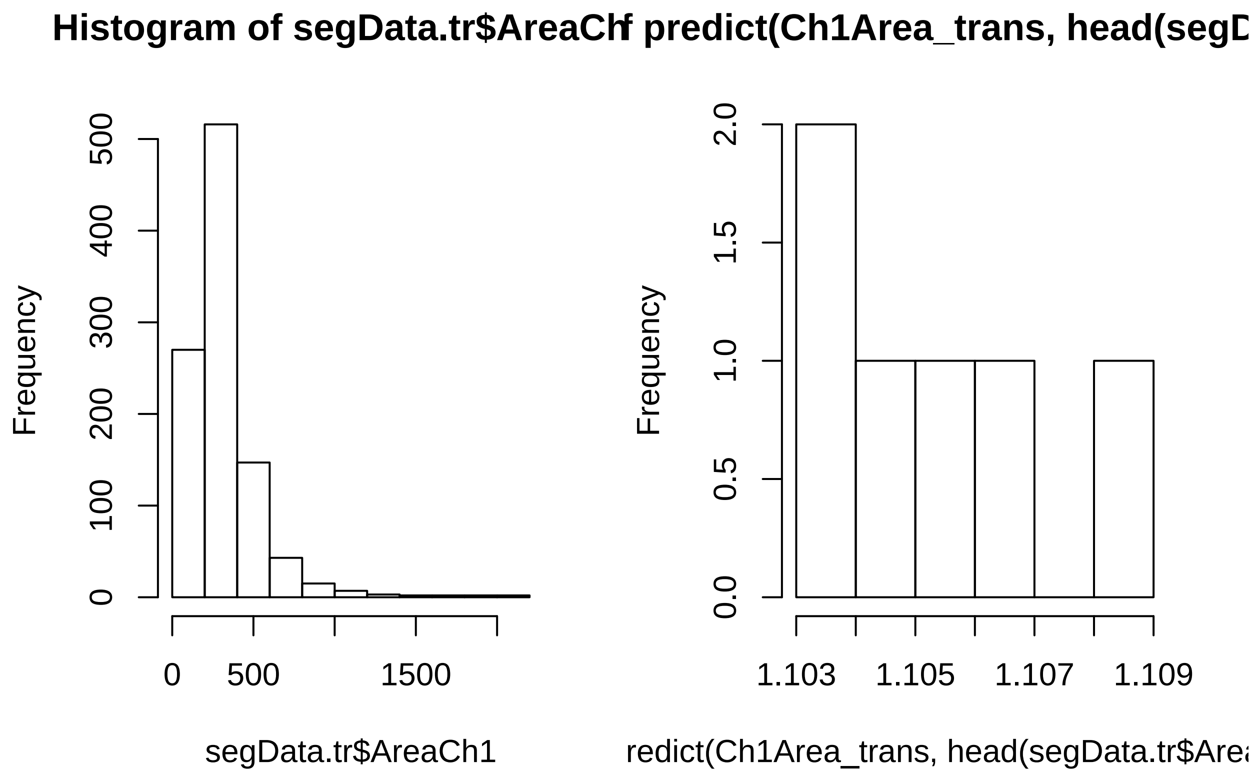
```
## [1] 819 431 298 256 258 358
```

```r
# head(Ch1Area_trans) no funciona es necesario aplicar la formula mediante predict
predict(Ch1Area_trans,head(segData.tr$AreaCh1))
```

```
## [1] 1.108458 1.106383 1.104520 1.103554 1.103607 1.105523
```

es justo,la transformación con $lambda = -0.9$ Datos transformados:

```r
par(mfrow=c(1,2))
hist (segData.tr$AreaCh1)
hist(predict(Ch1Area_trans,head(segData.tr$AreaCh1)))
```



Demasiados atributos

también en *caret* Algunos redundantes o irrelevantes, es conveniente reducir dimensionalidad. El algoritmo PCA (principal components analysis) es un filtro (selector de características) no supervisado. Aunque es sensible a escala y valores grandes, previamente se hace el centrado y la escala. Calcula el porcentaje del total de la varianza de los datos por cada atributo

```
pcaObject = prcomp(segData.tr,center = TRUE, scale. = TRUE)
attributes(pcaObject)
```

```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

```
head(pcaObject$center)
```

```
##    AngleCh1     AreaCh1 AvgIntenCh1 AvgIntenCh2 AvgIntenCh3 AvgIntenCh4
##    91.12641   325.12587   127.91503   185.19067    96.12917   140.02605
```

```
porcentVariance = pcaObject$sd^2/sum(pcaObject$sd^2)*100
porcentVariance[1:5]
```

```
## [1] 20.912359 17.013300 11.886892  7.715243  4.957698
```

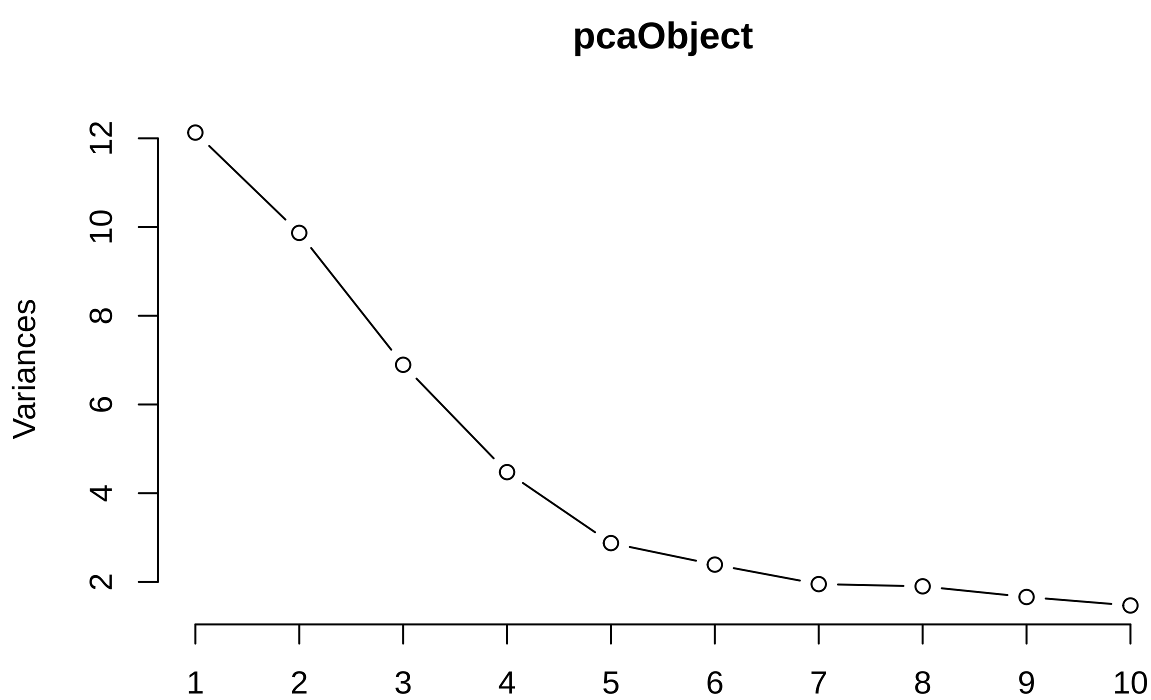
```
head(pcaObject$x[, 1:5])
```

```
##           PC1        PC2         PC3       PC4        PC5
## 2   5.0985749  4.5513804 -0.03345155 -2.640339  1.2783212
```

```
## 3  -0.2546261  1.1980326 -1.02059569 -3.731079  0.9994635
## 4   1.2928941 -1.8639348 -1.25110461 -2.414857 -1.4914838
## 12 -1.4646613 -1.5658327  0.46962088 -3.388716 -0.3302324
## 15 -0.8762771 -1.2790055 -1.33794261 -3.516794  0.3936099
## 16 -0.8615416 -0.3286842 -0.15546723 -2.206636  1.4731658
```

En X se encuentran los valores transformados ya.

```
plot(pcaObject,type="l")
```




```
head(pcaObject$rotation[, 1:5])
```

```
##                      PC1         PC2          PC3         PC4         PC5
## AngleCh1     0.001213758 -0.01284461  0.006816473 -0.02755720  0.02523673
## AreaCh1      0.229171873  0.16061734  0.089811727 -0.05523062  0.05273468
## AvgIntenCh1 -0.102708778  0.17971332  0.067696745  0.18675619  0.02401245
## AvgIntenCh2 -0.154828672  0.16376018  0.073534399  0.04145772  0.07839174
## AvgIntenCh3 -0.058042158  0.11197704 -0.185473286  0.28291291 -0.07822440
## AvgIntenCh4 -0.117343465  0.21039086 -0.105060977  0.01116373  0.04990515
```

Por filas vemos los atributos que forman parte de cada uno de los componentes y sus coeficientes. Por defecto la función selecciona aquellos componentes que explican hasta el 95% de la variabilidad de los datos… se puede cambiar con argumentos thresh.

A la hora de aplicar las transformaciones, en *caret* existe una función **preProcess()** que realiza todas transformaciones mencionadas de forma ordenada.

```
ObjetoTrans =  preProcess(segData.tr, method = c("BoxCox", "center", "scale", "pca"),thres=0.8)
ObjetoTrans
```

```
## Created from 1009 samples and 58 variables
##
## Pre-processing:
##   - Box-Cox transformation (47)
##   - centered (58)
##   - ignored (0)
##   - principal component signal extraction (58)
##   - scaled (58)
##
## Lambda estimates for Box-Cox transformation:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.00000 -0.50000 -0.10000  0.05106  0.30000  2.00000
##
## PCA needed 10 components to capture 80 percent of the variance
```

Para obtener un nuevo conjunto de datos, se aplican

```
segTrans = predict(ObjetoTrans,segData.tr)
dim(segTrans)
```

```
## [1] 1009   10
```

Eliminar las variables con varianza 0 o muy próximas, esto es muy desbalanceadas o de valor único.

```
nearZeroVar(segData.tr)
```

```
## integer(0)
```