Examen 1. Computo Estadistico

Radel Rojas B

12 de octubre de 2017

Instrucciones Los ejercicio se entregaran con el script de `R` y con un reporte que incluya los resultados obrtenidos y las respuestas solicitadas. El script y el reporte se subirán al drive en la carpeta marcada con su nombre.

Ejercicios

1) Considerando el conjunto de datos de **carSeats** de la libreria *ISLR*, Contruye un modelo para predecir las ventas considerando las restantes variables como preditoras

Solución Utilizando `R` se cargan las librerias y datos necesarios

```
library(ISLR) #Contiene la base de datos a utilizar
library(leaps)
data = Carseats #base de datos
dim(Carseats) # matriz de 777 observaciones(universidades) - 18 variables
```

```
## [1] 400  11
```

```
colnames(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

```
set.seed(1) # usamos esta función para fijar una semilla
#para los generadores de numeros aleaorios en R
```

Se Divide el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba

```
train =sample(400,200) # Se escoge un conjunto de entrenamiento de 200 datos aprox la mitad
test=-train #datos de prueba

attach(Carseats)
```

Se ajusta el modelo de regresión de Ridge sobre el conjunto de entrenamiento, con λ elegido por validacion cruzada. se Reporta el error de prueba obtenido.

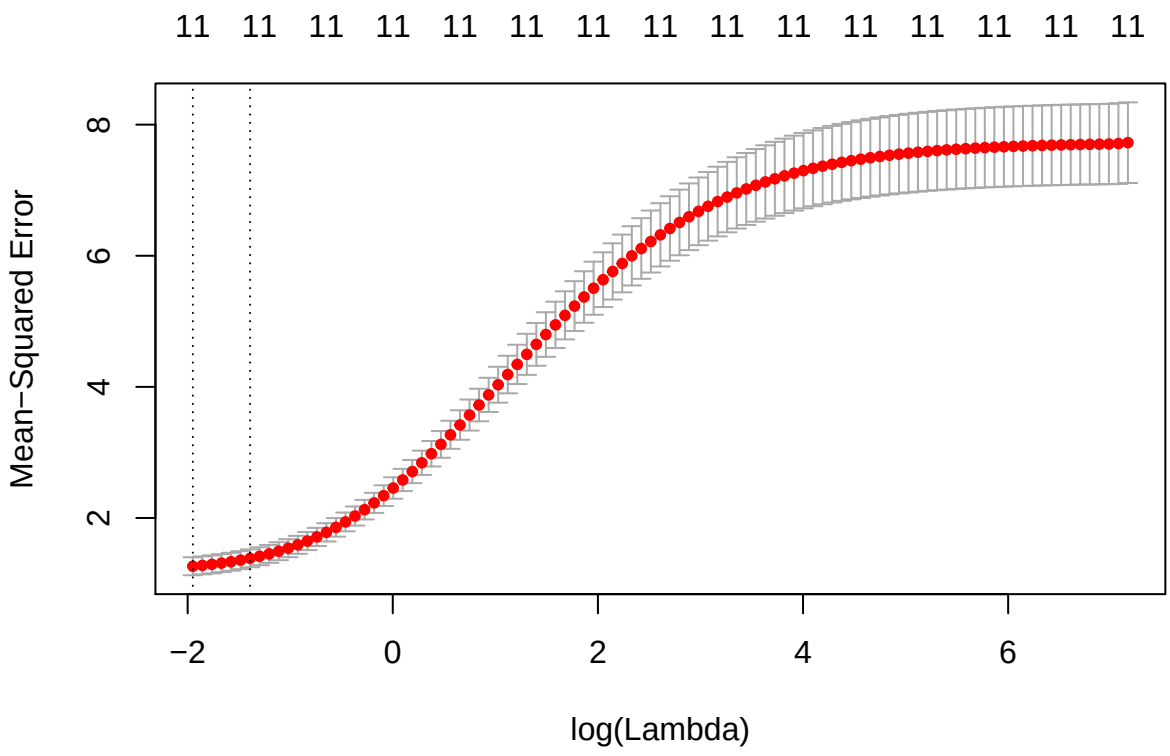
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-12
```

```
#contiene los procedimientos de regresion Ridge
x=model.matrix (Sales~.,Carseats )[,-1]
cv.out=cv.glmnet(x[train,],Sales[train],alpha =0)
#funcion que realiza la validacion cruzada k-fold, por default usa k=10
plot(cv.out)
```




```
#grafica los MSE para cada valor de lamda
bestlam=cv.out$lambda.min
#elige el valor de lamda que tiene el MSE mas pequeño
bestlam
```

```
## [1] 0.1423627
```

```
ridge.mod=glmnet(x[train,],Sales[train],lambda = bestlam)
ridge=predict(ridge.mod,s=bestlam,newx=x[test,])
error.ridge=mean((ridge-Sales[test])^2)
error.ridge
```

```
## [1] 1.327669
```

```
#calcula el MSE asociado al mejor valor de lamda

out=glmnet(x,Sales,alpha=0)
predict(out,type="coefficients",s=bestlam)
```

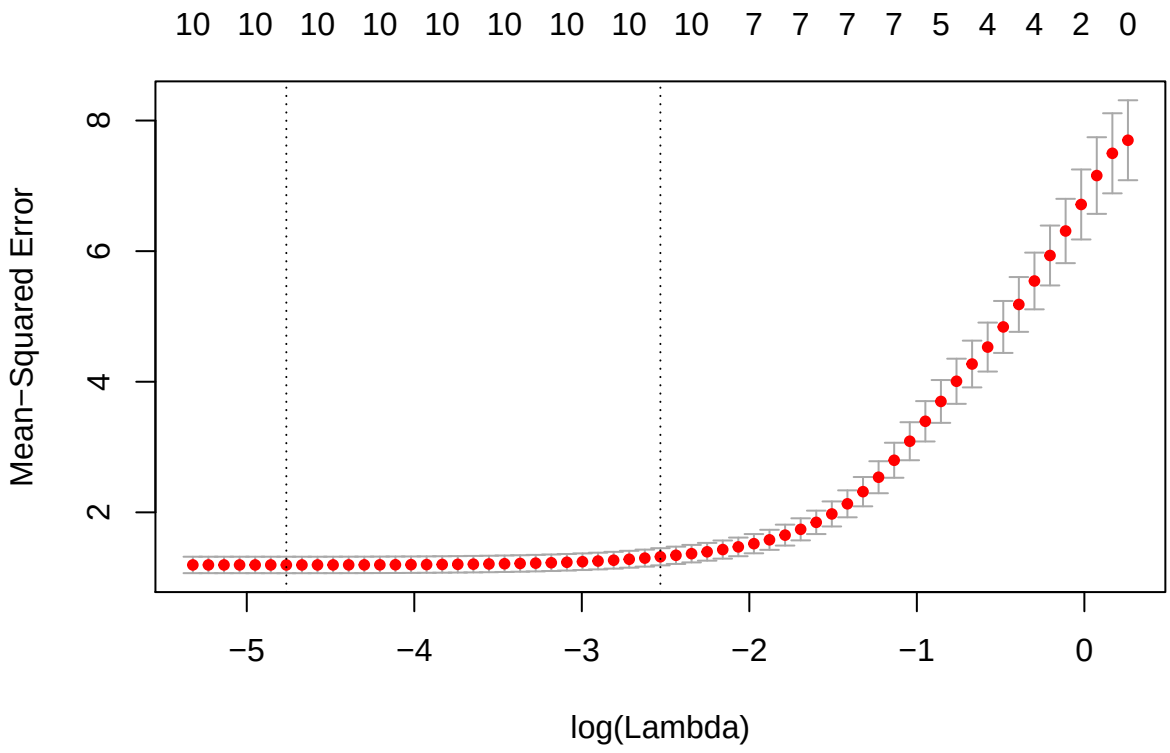
```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)      6.3131495430
## CompPrice        0.0805889042
## Income           0.0146489990
## Advertising      0.1115192505
## Population       0.0001899445
## Price           -0.0859391726
## ShelveLocGood    4.4155940236
```

```
## ShelveLocMedium  1.6697637915
## Age             -0.0434043360
## Education       -0.0209757603
## UrbanYes         0.0967790955
## USYes           -0.0741752218
```

```
#calcula los coeficientes
```

Se ajusta un modelo de LASSO en el conjunto de entrenamiento, con λ elegido por validacion cruzada. se reporta el error de prueba obtenido, junto con el número de estimación de coeficientes no nulos.

```
cv.out=cv.glmnet(x[train,],Sales[train],alpha =1)
#funcion que realiza la validacion cruzada k-fold, por default usa k=10
plot(cv.out)
```




```
#grafica los MSE para cada valor de lamda
bestlam=cv.out$lambda.min
#elige el valor de lamda que tiene el MSE mas pequeño
bestlam
```

```
## [1] 0.008534418
```

```
lasso.mod=glmnet(x[train,],Sales[train],lambda = bestlam)
lasso=predict(lasso.mod,s=bestlam,newx=x[test,])
error.lasso=mean((lasso-Sales[test])^2)
error.lasso
```

```
## [1] 1.03146
```

```r
#calcula el MSE asociado al mejor valor de lamda

out=glmnet(x,Sales,alpha=1)
predict(out,type="coefficients",s=bestlam)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)      5.7318441089
## CompPrice        0.0913604642
## Income           0.0154261754
## Advertising      0.1192249821
## Population       0.0001721316
## Price           -0.0944033173
## ShelveLocGood    4.7967775109
## ShelveLocMedium  1.9145728770
## Age             -0.0454957307
## Education       -0.0179439191
## UrbanYes         0.1012877251
## USYes           -0.1259129485
```

```r
#determina los coeficientes de lasso
```

Se ajusta modelo PCR en el conjunto de entrenamiento, con M elegido por validación cruzada. Reporta el error de prueba obtenido, junto con el valor de M seleccionado.

```r
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.4.2
```

```
##
## Attaching package: 'pls'
```

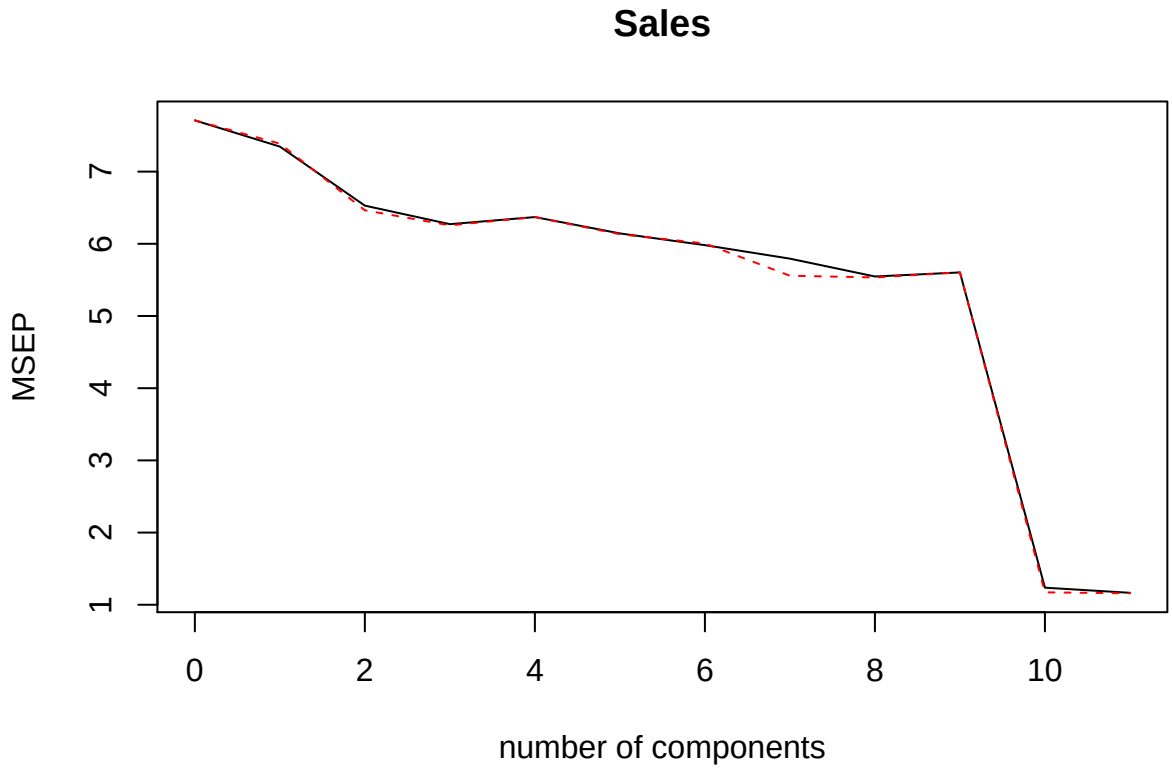
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
#contiene  las funciones que realizann PCR y PLS
pcr.fit=pcr(Sales~.,data=Carseats,subset=train,scale=TRUE,validation ="CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 200 11
##  Y dimension: 200 1
## Fit method: svdpc
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           2.777    2.710    2.555    2.504    2.524    2.479    2.446
## adjCV        2.777    2.718    2.543    2.501    2.524    2.476    2.450
##        7 comps  8 comps  9 comps  10 comps  11 comps
## CV       2.407    2.355    2.367     1.112     1.080
## adjCV    2.358    2.352    2.367     1.083     1.077
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       17.827    32.95    46.94    57.36    66.38    74.58    82.46
```

```
## Sales    4.925    17.15    20.42    20.53    23.37    26.33    33.71
##        8 comps  9 comps  10 comps  11 comps
## X        90.21    94.39     97.48    100.00
## Sales    34.41    34.80     86.18     86.26
```

```r
validationplot(pcr.fit,val.type="MSEP")
```

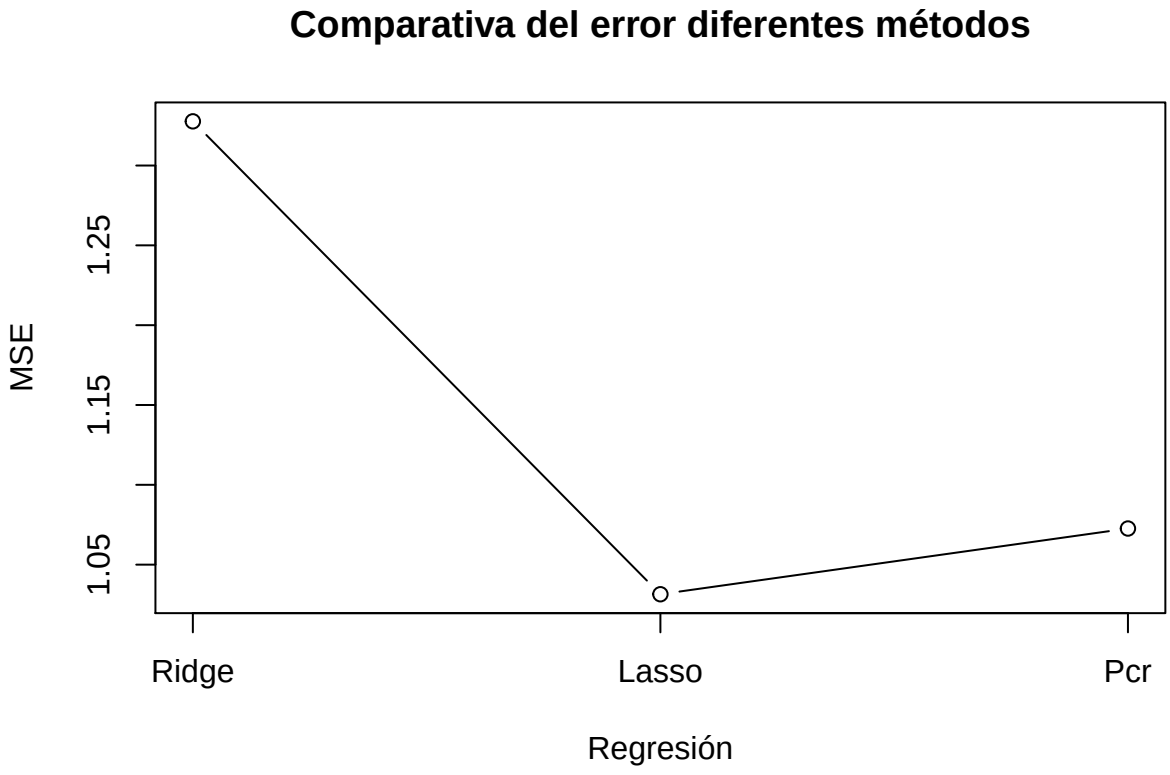



```r
# grafica los valores de los errores de VC para cada M, componentes principales.

pcr.pred=predict(pcr.fit,x[test,],ncomp =10) # esta funcion predice
#los valores de prueba de acuerdo al modelo ajustado con  10
#componentes
error.pcr=mean((pcr.pred-Sales[test])^2) # se calcula el error de prueba
error.pcr
```

```
## [1] 1.072627
```

De los modelos anteriores, se observa en la siguiente grafica que la regresion Lasso, logra disminuir el error cuadratico al minimo, en contraste con los otros modelos




```
error.ridge
```

```
## [1] 1.327669
```

```
error.lasso
```

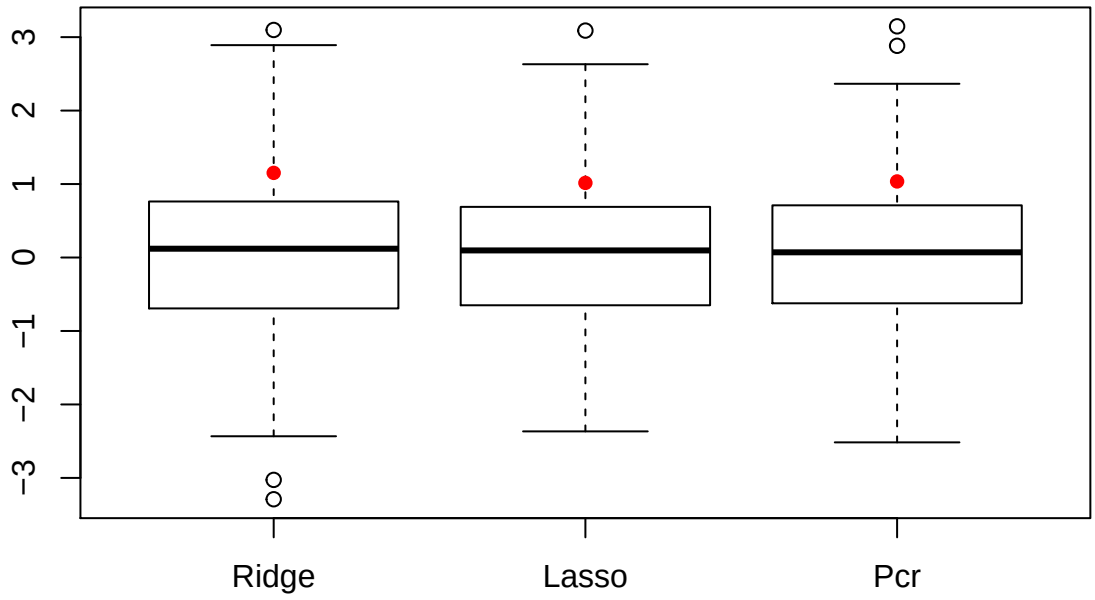
```
## [1] 1.03146
```

```
error.pcr
```

```
## [1] 1.072627
```

Lo anterior, aunado a una menor dispersión de los datos resultantes (como se aprecia en el boxplot)

```
boxplot(data.frame(Sales[test]-ridge,Sales[test]-lasso,Sales[test]-pcr.pred),xaxt="n")
axis(1, at=c(1:3),labels = c("Ridge","Lasso","Pcr"))
points(1:3,c(error.ridge,error.lasso,error.pcr)^(0.5),cex=1,pch=16,col="red")
```



invita a utilizar regresión Lasson $\sqrt{(Errorcuadraticomedio)}$. Considerando aquellos coeficentes mas significativos para nuestra regresión

```
predict(out,type="coefficients",s=bestlam)
```

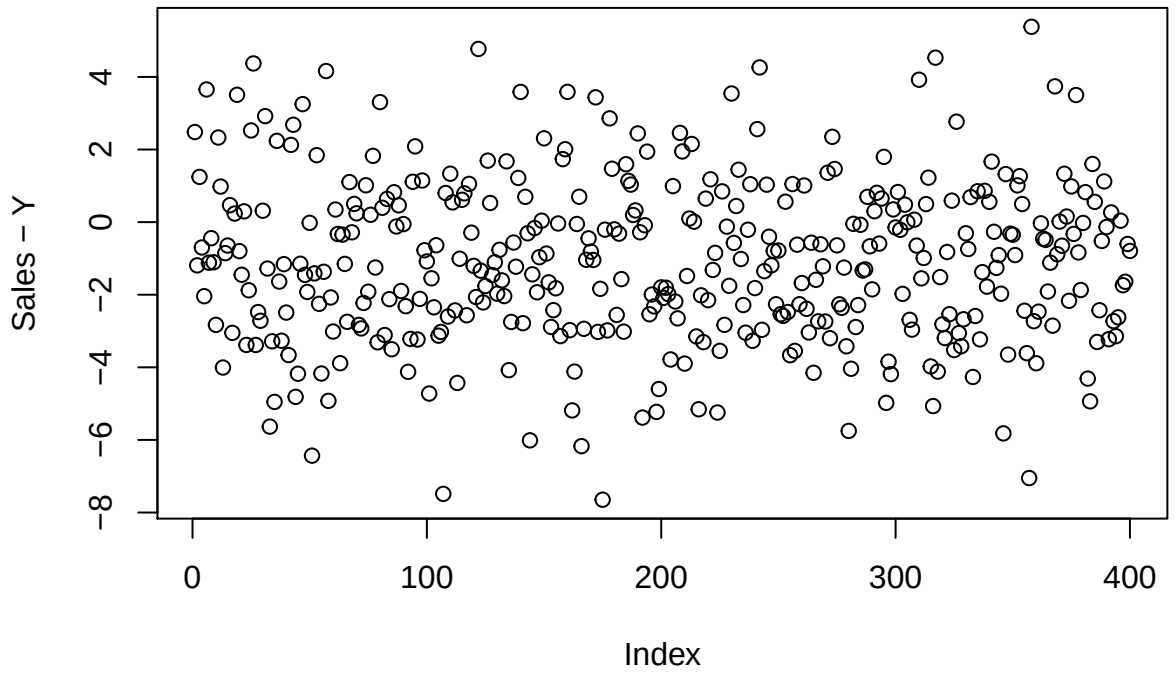
```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)      5.7318441089
## CompPrice        0.0913604642
## Income           0.0154261754
## Advertising      0.1192249821
## Population       0.0001721316
## Price           -0.0944033173
## ShelveLocGood    4.7967775109
## ShelveLocMedium  1.9145728770
## Age             -0.0454957307
## Education       -0.0179439191
## UrbanYes         0.1012877251
## USYes           -0.1259129485
```

Escogemos: Advertising, ShelveLocGood,ShelveLocMedium,UrbanYes. Las cuales corresponden a los mayores coeficientes. Se propone un modelo de tipo $sales = 5.73 + 4.79 * ShelveLocGood + 1.91 * ShelveLocMedium + 0.11 * Advertising + 0.10 * UrbanYes + USyes * -0.12$ Se observa que no se incluyen todas las carateristicas, dado que de acuerdo a nuestros metodos, estas no estan directamente relacionadas con la variable respuesta y por tanto no explican de forma adecuada las ventas. Aplicando regresión con las variables determinandas se tiene el modelo

```
betas= c(5.7318441089,0.1192249821,4.7967775109,1.9145728770,0.1012877251,-0.1259129485)
Y=as.matrix(data.frame(1,x[,3],x[,6],x[,7],x[,10],x[,11]))%*%(as.matrix(betas)) #vector respuestas
```

Luego se determina la aproximacion de las respuestas

```
plot(Sales-Y) #Grafico de errores
```




```
1-mean((Sales-Y)[test]^2)/mean((Sales-Y)^2) #no te que se tiene un coeficiente R^2 cercano a cero, de d
```

```
## [1] 0.0783195
```

```
cor(Sales,Y) # Como era de esperarse se tiene una correlacion de al menos un 60 % entre los datos origi
```

```
##           [,1]
## [1,] 0.6111092
```