Trabajo Final

Reinier Mujica

13 de enero de 2019

### Inicialización

Limpiamos todas las variables del espacio de trabajo y el terminal de R.

rm(list=ls())  
cat("\014")

library(knitr)

Cargamos el paquete de wooldridge y los datos

library(wooldridge)  
attach(bwght2)

Fijamos la semilla

set.seed(100)

### Limpiar los datos

Vamos a remover los valores NA de los datos

datos = na.omit(bwght2)

### División del conjunto de datos

train.size = round(dim(datos)[1] \* 0.7)  
train = sample(1:dim(datos)[1], train.size)  
test = -train  
datos.train = datos[train, ]  
datos.test = datos[test, ]  
test.size = dim(datos.test)[1]

### Preguntas:

#### a)

El conjunto de datos es bwght2 y contiene información sobre nacimientos. Estos datos han sido obtenidos de certificados de nacimiento y desfunción y también de información de natalidad y mortalidad del Centro Nacional de Estadística para la Salud. El dataset contiene varias variables con diversa información como: edad de los padres, educación de los padres, número de visitas prenatales, peso al nacer, e información acerca de la cantidad de cigarillos y bebida que consumia la madre durante el embarazo. También se incluye información del feto.

Los nombres de las variables que contiene el dataset son:

names(bwght2)

[1] "mage" "meduc" "monpre" "npvis" "fage" "feduc" "bwght"   
 [8] "omaps" "fmaps" "cigs" "drink" "lbw" "vlbw" "male"   
[15] "mwhte" "mblck" "moth" "fwhte" "fblck" "foth" "lbwght"   
[22] "magesq" "npvissq"

La variable dependiente que va ha ser explicada es **lbwght**, que es el logaritmo de la variable **bwght** que es el peso al nacer.

Vamos a ajustar un modelo de mínimos cuadrados ordinarios (con todas las variables explicativas) en el conjunto de entrenamiento e imprimir el error de prueba obtenido.

set.seed(100)  
lm.fit = lm(lbwght~., data=datos.train)  
lm.pred = predict(lm.fit, newdata = datos.test)  
  
error.mco <- mean((datos.test[, "lbwght"] - lm.pred)^2)  
error.mco

[1] 0.000518739

Creamos una tabla para guardar los resultados de todos los modelos

results <- matrix(NA, nrow = 3, ncol = 3)  
colnames(results) <- c("None","5 Cross Validation","10 Cross Validation")  
rownames(results) <- c("MCO","MCO with Subset Selection","MCO with Forward step wise")  
results <- as.table(results)  
  
results["MCO", "None"] = error.mco

Los datos del modelo y los coeficientes B los podemos ver a continuación.

summary(lm.fit)

Call:  
lm(formula = lbwght ~ ., data = datos.train)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-0.51085 -0.00469 0.00483 0.01096 0.25896   
  
Coefficients: (2 not defined because of singularities)  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 7.033e+00 3.045e-02 230.942 < 2e-16 \*\*\*  
mage -1.245e-03 1.629e-03 -0.765 0.4446   
meduc -4.244e-04 5.006e-04 -0.848 0.3967   
monpre 5.921e-04 7.333e-04 0.807 0.4196   
npvis 1.056e-04 6.817e-04 0.155 0.8769   
fage 2.315e-04 2.013e-04 1.150 0.2504   
feduc -4.886e-06 4.527e-04 -0.011 0.9914   
bwght 2.979e-04 1.612e-06 184.811 < 2e-16 \*\*\*  
omaps 9.290e-04 8.914e-04 1.042 0.2975   
fmaps 9.733e-03 2.103e-03 4.629 4.11e-06 \*\*\*  
cigs 3.931e-04 2.164e-04 1.817 0.0696 .   
drink -6.184e-04 3.206e-03 -0.193 0.8471   
lbw -1.540e-01 9.160e-03 -16.816 < 2e-16 \*\*\*  
vlbw -3.316e-01 1.387e-02 -23.914 < 2e-16 \*\*\*  
male -7.498e-04 1.639e-03 -0.457 0.6475   
mwhte 3.539e-03 7.362e-03 0.481 0.6308   
mblck 2.381e-03 1.144e-02 0.208 0.8351   
moth NA NA NA NA   
fwhte -5.610e-03 7.984e-03 -0.703 0.4824   
fblck 2.088e-03 1.173e-02 0.178 0.8588   
foth NA NA NA NA   
magesq 1.586e-05 2.710e-05 0.585 0.5586   
npvissq -6.468e-06 2.138e-05 -0.302 0.7623   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.02708 on 1107 degrees of freedom  
Multiple R-squared: 0.9804, Adjusted R-squared: 0.98   
F-statistic: 2762 on 20 and 1107 DF, p-value: < 2.2e-16

lm.fit$coefficients

(Intercept) mage meduc monpre npvis   
 7.032634e+00 -1.245338e-03 -4.244261e-04 5.920562e-04 1.056039e-04   
 fage feduc bwght omaps fmaps   
 2.315328e-04 -4.885967e-06 2.978545e-04 9.290155e-04 9.733477e-03   
 cigs drink lbw vlbw male   
 3.931007e-04 -6.183647e-04 -1.540387e-01 -3.316268e-01 -7.498069e-04   
 mwhte mblck moth fwhte fblck   
 3.538706e-03 2.381276e-03 NA -5.610199e-03 2.087637e-03   
 foth magesq npvissq   
 NA 1.585596e-05 -6.467952e-06

Podemos concluir que dos coeficientes fueron omitidos por estar las respectivas variables muy correlacionadas(**moth** y **foth**). También como se puede ver en el resultado de la instrucción **summary** solo las variables:

**bwght** **fmaps**  
**lbw**  
**vlbw**

Son significativas para predecir **lbwght**.

#### b)

Procedemos a eliminar del Dataset las dos variables correlacionadas **moth** y **foth**

datos[, "moth"] <- NULL  
datos[, "foth"] <- NULL  
datos.train[, "moth"] <- NULL  
datos.train[, "foth"] <- NULL  
datos.test[, "moth"] <- NULL  
datos.test[, "foth"] <- NULL

Luego ajustamos un MCO utilizando la Mejor Selección de Conjuntos

library(leaps)  
  
nvariables <- as.numeric(dim(datos)[2] - 1)  
regfit.full = regsubsets(lbwght~., data=datos[train,], nvmax=nvariables)  
  
predict.regsubsets=function(object, newdata, id, ...) {  
 form=as.formula(object$call[[2]])  
 mat=model.matrix(form,newdata)  
 coefi=coef(object,id=id)   
 xvars=names(coefi)  
 mat[,xvars]%\*%coefi  
}  
  
k = 10  
set.seed(100)  
folds=sample(1:k,nrow(datos.train),replace=TRUE)  
table(folds)

folds  
 1 2 3 4 5 6 7 8 9 10   
 97 100 109 129 107 103 120 120 112 131

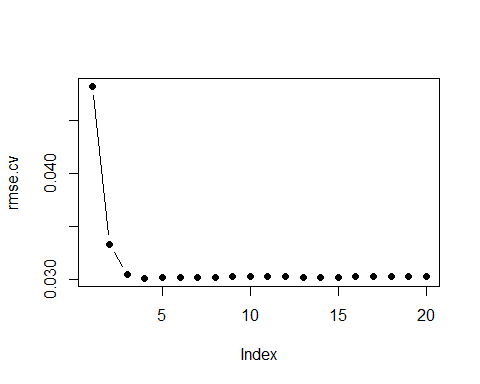
cv.errors=matrix(NA,k,nvariables, dimnames =list(NULL , paste(1:nvariables)))  
for(j in 1:k){  
 best.fit=regsubsets(lbwght~.,data=datos.train[folds!=j,],  
 nvmax=nvariables)  
 for(i in 1:nvariables){  
 pred=predict.regsubsets(best.fit,datos.train[folds==j,],id=i)  
 cv.errors[j,i]=mean( (datos.train$lbwght[folds==j]-pred)^2)  
 }  
}  
rmse.cv=sqrt(apply(cv.errors,2,mean))  
rmse.cv

1 2 3 4 5 6   
0.04821566 0.03330704 0.03041755 0.03007367 0.03013112 0.03014558   
 7 8 9 10 11 12   
0.03015778 0.03015774 0.03027918 0.03029074 0.03025820 0.03022182   
 13 14 15 16 17 18   
0.03020467 0.03019556 0.03019658 0.03021205 0.03022838 0.03022756   
 19 20   
0.03022169 0.03022142

which.min(rmse.cv)

4   
4

plot(rmse.cv,pch=19,type="b")



reg.best=regsubsets (lbwght~.,data=datos.train , nvmax=nvariables)  
coef(reg.best ,which.min(rmse.cv))

(Intercept) bwght fmaps lbw vlbw   
 7.0123631237 0.0002976461 0.0105802459 -0.1543394286 -0.3309693951

# Modelo final acorde a las la mejor selección de conjuntos  
fit.final <- lm(lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
summary(fit.final)

Call:  
lm(formula = lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-0.51461 -0.00406 0.00584 0.01070 0.26192   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 7.012e+00 1.703e-02 411.683 < 2e-16 \*\*\*  
bwght 2.976e-04 1.585e-06 187.769 < 2e-16 \*\*\*  
fmaps 1.058e-02 1.834e-03 5.768 1.04e-08 \*\*\*  
lbw -1.543e-01 9.002e-03 -17.145 < 2e-16 \*\*\*  
vlbw -3.310e-01 1.373e-02 -24.109 < 2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.02708 on 1123 degrees of freedom  
Multiple R-squared: 0.9801, Adjusted R-squared: 0.98   
F-statistic: 1.381e+04 on 4 and 1123 DF, p-value: < 2.2e-16

regfit.full=regsubsets(lbwght~.,data= datos[train,],nvmax=nvariables)  
  
lm.pred = predict.regsubsets(regfit.full, newdata = datos.test, id=which.min(rmse.cv))  
error.mss <- mean((datos.test[, "lbwght"] - lm.pred)^2)  
error.mss

[1] 0.000511914

results["MCO with Subset Selection", "10 Cross Validation"] = error.mss

El error de prueba obtenido es mas pequeño que con el MCO ordinario.

#### c)

nvariables <- as.numeric(dim(datos)[2] - 1)  
regfit.full = regsubsets(lbwght~., data=datos[train,], nvmax=nvariables, method = "forward")  
  
k = 10  
set.seed(100)  
folds=sample(1:k,nrow(datos.train),replace=TRUE)  
  
table(folds)

folds  
 1 2 3 4 5 6 7 8 9 10   
 97 100 109 129 107 103 120 120 112 131

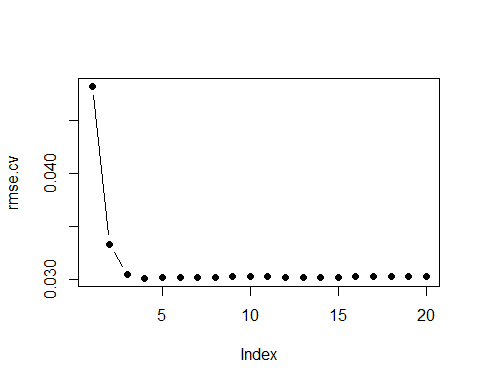
cv.errors=matrix(NA,k,nvariables, dimnames =list(NULL , paste(1:nvariables)))  
for(j in 1:k){  
 best.fit=regsubsets(lbwght~.,data=datos.train[folds!=j,],  
 nvmax=nvariables, method = "forward")  
 for(i in 1:nvariables){  
 pred=predict.regsubsets(best.fit,datos.train[folds==j,],id=i)  
 cv.errors[j,i]=mean( (datos.train$lbwght[folds==j]-pred)^2)  
 }  
}  
rmse.cv=sqrt(apply(cv.errors,2,mean))  
rmse.cv

1 2 3 4 5 6   
0.04821566 0.03330704 0.03041755 0.03007367 0.03013112 0.03015472   
 7 8 9 10 11 12   
0.03015426 0.03015431 0.03022841 0.03023027 0.03022682 0.03018473   
 13 14 15 16 17 18   
0.03019260 0.03018360 0.03018362 0.03021553 0.03022206 0.03022344   
 19 20   
0.03022860 0.03022142

which.min(rmse.cv)

4   
4

plot(rmse.cv,pch=19,type="b")



reg.best=regsubsets (lbwght~.,data=datos.train , nvmax=nvariables, method = "forward")  
coef(reg.best ,which.min(rmse.cv))

(Intercept) bwght fmaps lbw vlbw   
 7.0123631237 0.0002976461 0.0105802459 -0.1543394286 -0.3309693951

# Modelo final acorde a las la mejor selección de conjuntos  
fit.final <- lm(lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
summary(fit.final)

Call:  
lm(formula = lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-0.51461 -0.00406 0.00584 0.01070 0.26192   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 7.012e+00 1.703e-02 411.683 < 2e-16 \*\*\*  
bwght 2.976e-04 1.585e-06 187.769 < 2e-16 \*\*\*  
fmaps 1.058e-02 1.834e-03 5.768 1.04e-08 \*\*\*  
lbw -1.543e-01 9.002e-03 -17.145 < 2e-16 \*\*\*  
vlbw -3.310e-01 1.373e-02 -24.109 < 2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.02708 on 1123 degrees of freedom  
Multiple R-squared: 0.9801, Adjusted R-squared: 0.98   
F-statistic: 1.381e+04 on 4 and 1123 DF, p-value: < 2.2e-16

regfit.full=regsubsets(lbwght~.,data= datos[train,],nvmax=nvariables, method = "forward")  
  
lm.pred = predict.regsubsets(regfit.full, newdata = datos.test, id=which.min(rmse.cv))  
error.mss <- mean((datos.test[, "lbwght"] - lm.pred)^2)  
error.mss

[1] 0.000511914

results["MCO with Forward step wise", "10 Cross Validation"] = error.mss

#### d)

nvariables <- as.numeric(dim(datos)[2] - 1)  
regfit.full = regsubsets(lbwght~., data=datos[train,], nvmax=nvariables)  
  
predict.regsubsets=function(object, newdata, id, ...) {  
 form=as.formula(object$call[[2]])  
 mat=model.matrix(form,newdata)  
 coefi=coef(object,id=id)   
 xvars=names(coefi)  
 mat[,xvars]%\*%coefi  
}  
  
k = 5  
set.seed(100)  
folds=sample(1:k,nrow(datos.train),replace=TRUE)  
  
table(folds)

folds  
 1 2 3 4 5   
197 238 210 240 243

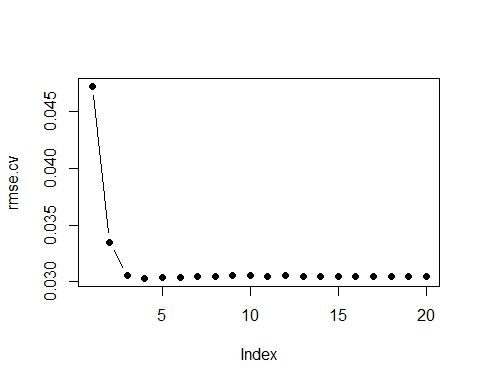
cv.errors=matrix(NA,k,nvariables, dimnames =list(NULL , paste(1:nvariables)))  
for(j in 1:k){  
 best.fit=regsubsets(lbwght~.,data=datos.train[folds!=j,],  
 nvmax=nvariables)  
 for(i in 1:nvariables){  
 pred=predict.regsubsets(best.fit,datos.train[folds==j,],id=i)  
 cv.errors[j,i]=mean( (datos.train$lbwght[folds==j]-pred)^2)  
 }  
}  
rmse.cv=sqrt(apply(cv.errors,2,mean))  
rmse.cv

1 2 3 4 5 6   
0.04724912 0.03348780 0.03056028 0.03030288 0.03039083 0.03039898   
 7 8 9 10 11 12   
0.03048605 0.03045558 0.03052369 0.03053467 0.03051360 0.03051497   
 13 14 15 16 17 18   
0.03049662 0.03048933 0.03050103 0.03051083 0.03050089 0.03050719   
 19 20   
0.03050598 0.03050462

which.min(rmse.cv)

4   
4

plot(rmse.cv,pch=19,type="b")



reg.best=regsubsets (lbwght~.,data=datos.train , nvmax=nvariables)  
coef(reg.best ,which.min(rmse.cv))

(Intercept) bwght fmaps lbw vlbw   
 7.0123631237 0.0002976461 0.0105802459 -0.1543394286 -0.3309693951

# Modelo final acorde a las la mejor selección de conjuntos  
fit.final <- lm(lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
summary(fit.final)

Call:  
lm(formula = lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-0.51461 -0.00406 0.00584 0.01070 0.26192   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 7.012e+00 1.703e-02 411.683 < 2e-16 \*\*\*  
bwght 2.976e-04 1.585e-06 187.769 < 2e-16 \*\*\*  
fmaps 1.058e-02 1.834e-03 5.768 1.04e-08 \*\*\*  
lbw -1.543e-01 9.002e-03 -17.145 < 2e-16 \*\*\*  
vlbw -3.310e-01 1.373e-02 -24.109 < 2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.02708 on 1123 degrees of freedom  
Multiple R-squared: 0.9801, Adjusted R-squared: 0.98   
F-statistic: 1.381e+04 on 4 and 1123 DF, p-value: < 2.2e-16

regfit.full=regsubsets(lbwght~.,data= datos[train,],nvmax=nvariables)  
  
lm.pred = predict.regsubsets(regfit.full, newdata = datos.test, id=which.min(rmse.cv))  
error.mss <- mean((datos.test[, "lbwght"] - lm.pred)^2)  
error.mss

[1] 0.000511914

results["MCO with Subset Selection", "5 Cross Validation"] = error.mss

nvariables <- as.numeric(dim(datos)[2] - 1)  
regfit.full = regsubsets(lbwght~., data=datos[train,], nvmax=nvariables, method = "forward")  
  
k = 5  
set.seed(100)  
folds=sample(1:k,nrow(datos.train),replace=TRUE)  
  
table(folds)

folds  
 1 2 3 4 5   
197 238 210 240 243

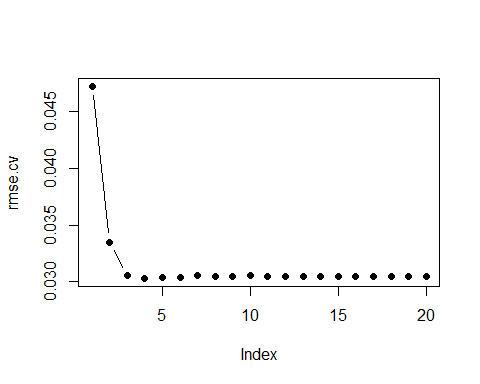
cv.errors=matrix(NA,k,nvariables, dimnames =list(NULL , paste(1:nvariables)))  
for(j in 1:k){  
 best.fit=regsubsets(lbwght~.,data=datos.train[folds!=j,],  
 nvmax=nvariables, method = "forward")  
 for(i in 1:nvariables){  
 pred=predict.regsubsets(best.fit,datos.train[folds==j,],id=i)  
 cv.errors[j,i]=mean( (datos.train$lbwght[folds==j]-pred)^2)  
 }  
}  
rmse.cv=sqrt(apply(cv.errors,2,mean))  
rmse.cv

1 2 3 4 5 6   
0.04724912 0.03348780 0.03056028 0.03030288 0.03039083 0.03039898   
 7 8 9 10 11 12   
0.03052785 0.03050632 0.03047763 0.03051874 0.03049464 0.03050382   
 13 14 15 16 17 18   
0.03047868 0.03047105 0.03047792 0.03048737 0.03048974 0.03048264   
 19 20   
0.03050674 0.03050462

which.min(rmse.cv)

4   
4

plot(rmse.cv,pch=19,type="b")



reg.best=regsubsets (lbwght~.,data=datos.train , nvmax=nvariables, method = "forward")  
coef(reg.best ,which.min(rmse.cv))

(Intercept) bwght fmaps lbw vlbw   
 7.0123631237 0.0002976461 0.0105802459 -0.1543394286 -0.3309693951

# Modelo final acorde a las la mejor selección de conjuntos  
fit.final <- lm(lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
summary(fit.final)

Call:  
lm(formula = lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-0.51461 -0.00406 0.00584 0.01070 0.26192   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 7.012e+00 1.703e-02 411.683 < 2e-16 \*\*\*  
bwght 2.976e-04 1.585e-06 187.769 < 2e-16 \*\*\*  
fmaps 1.058e-02 1.834e-03 5.768 1.04e-08 \*\*\*  
lbw -1.543e-01 9.002e-03 -17.145 < 2e-16 \*\*\*  
vlbw -3.310e-01 1.373e-02 -24.109 < 2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.02708 on 1123 degrees of freedom  
Multiple R-squared: 0.9801, Adjusted R-squared: 0.98   
F-statistic: 1.381e+04 on 4 and 1123 DF, p-value: < 2.2e-16

regfit.full=regsubsets(lbwght~.,data= datos[train,],nvmax=nvariables, method = "forward")  
  
lm.pred = predict.regsubsets(regfit.full, newdata = datos.test, id=which.min(rmse.cv))  
error.mss <- mean((datos.test[, "lbwght"] - lm.pred)^2)  
error.mss

[1] 0.000511914

results["MCO with Forward step wise", "5 Cross Validation"] = error.mss

#### e)

A continuación se muestran los modelos utilizados anteriormente con el tipo de validación cruzada y su error de prueba correspondiente.

kable(results)

|  |  |  |  |
| --- | --- | --- | --- |
|  | None | 5 Cross Validation | 10 Cross Validation |
| MCO | 0.0005187 | NA | NA |
| MCO with Subset Selection | NA | 0.0005119 | 0.0005119 |
| MCO with Forward step wise | NA | 0.0005119 | 0.0005119 |
| Como se puede observar en la | tabla no hay | mucha diferencia ent | re los errores de prueba obtenidos de estos enfoques. |

#### f)

Para seleccionar el modelo que minimiza el error, como no hay mucha diferencia entre los errores anteriores, escogimos el último modelo analizado **MCO with Forward step wise**. En el análisis de este modelo se determinaron las variables significativas al 5%, son las que se usan a continuación.

fit.final <- lm(lbwght ~ bwght + fmaps + lbw + vlbw, data = datos.train)  
  
lm.pred <- predict.regsubsets(fit.final, newdata = datos.test)  
error.new = mean((datos.test[, "lbwght"] - lm.pred)^2)  
error.new

[1] 0.000511914

El modelo con sólo cuatro variables explicativas con un nivel de significación del 5% (**bwght, fmaps, lbw, vlbw**) y un error de 5.187389710^{-4}, es un poco mas preciso respecto al error del MCO: 5.187389710^{-4}.

Los **p-valores** se muestran a continuación:

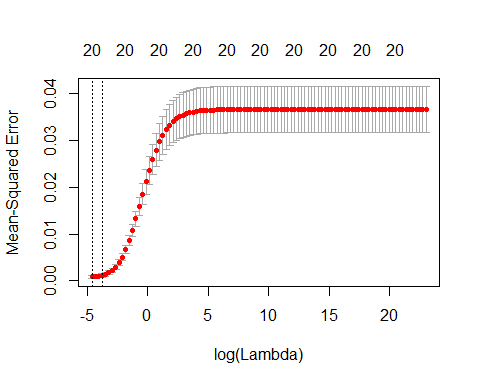
summary(fit.final)[4]

$coefficients  
 Estimate Std. Error t value Pr(>|t|)  
(Intercept) 7.0123631237 1.703342e-02 411.682667 0.000000e+00  
bwght 0.0002976461 1.585174e-06 187.768796 0.000000e+00  
fmaps 0.0105802459 1.834398e-03 5.767694 1.038118e-08  
lbw -0.1543394286 9.001872e-03 -17.145259 1.041566e-58  
vlbw -0.3309693951 1.372829e-02 -24.108564 7.864423e-104

#### g)

Regresión de Ridge

set.seed(100)  
library(glmnet)  
x=model.matrix(lbwght~.,datos)[,-1]  
y=datos$lbwght  
y.test=y[test]  
grid=10^seq(10,-2, length =100)  
cv.ridge=cv.glmnet(x[train ,],y[train],alpha=0,lambda=grid)  
plot(cv.ridge)



mejorlambda=cv.ridge$lambda.min  
mejorlambda

[1] 0.01

ridge.mod=glmnet(x[train ,],y[train],alpha=0,lambda=grid)  
ridge.pred=predict(ridge.mod,s=mejorlambda ,newx=x[test ,])  
  
error.ridge <- mean((ridge.pred-datos.test[, "lbwght"] )^2)  
error.ridge

[1] 0.0005855056

results2["RIDGE", "10 Cross Validation"] = error.ridge  
  
# Regla del "codo" de una DT del error de VC:  
lambda.codo <- cv.ridge$lambda.1se  
lambda.codo

[1] 0.0231013

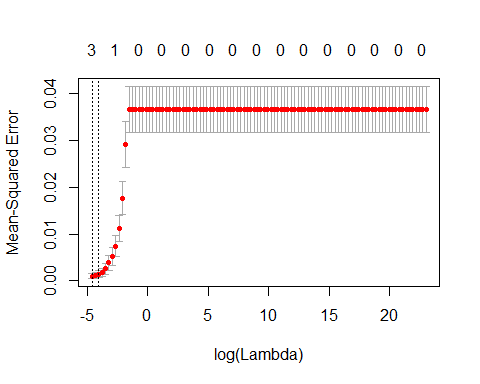
ridge.pred.2=predict(ridge.mod,s=lambda.codo,newx=x[test ,])  
error.ridge.2 <- mean((ridge.pred.2-datos.test[, "lbwght"] )^2)  
error.ridge.2

[1] 0.000850785

#### h)

Modelo de LASSO

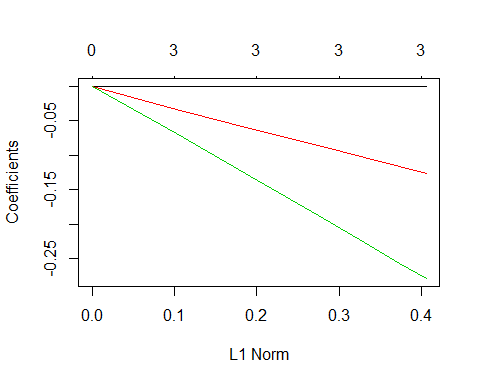
set.seed(100)  
cv.lasso=cv.glmnet(x[train ,],y[train],alpha=1, lambda = grid)  
plot(cv.lasso)



bestlam=cv.lasso$lambda.min  
bestlam

[1] 0.01

lasso.mod=glmnet(x[train ,],y[train],alpha=1,lambda=grid)  
plot(lasso.mod)



lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test ,])  
  
error.lasso <- mean((lasso.pred-datos.test[, "lbwght"] )^2)  
error.lasso

[1] 0.000435723

results2["LASSO", "10 Cross Validation"] = error.lasso  
  
# Regla del "codo" de una DT del error de VC:  
lambda.codo.l <- cv.lasso$lambda.1se  
lambda.codo.l

[1] 0.01747528

lasso.pred.2=predict(lasso.mod,s=lambda.codo.l,newx=x[test ,])  
error.lasso.2 <- mean((lasso.pred.2-datos.test[, "lbwght"] )^2)  
error.lasso.2

[1] 0.0005318692

Para obtener los coeficientes del modelo para el mejor lambda usamos la función coef

c = coef(lasso.mod, s = bestlam)  
c

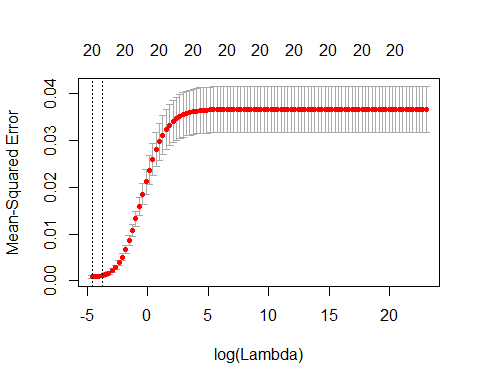
21 x 1 sparse Matrix of class "dgCMatrix"  
 1  
(Intercept) 7.1446826417  
mage .   
meduc .   
monpre .   
npvis .   
fage .   
feduc .   
bwght 0.0002865851  
omaps .   
fmaps .   
cigs .   
drink .   
lbw -0.1261450669  
vlbw -0.2789833877  
male .   
mwhte .   
mblck .   
fwhte .   
fblck .   
magesq .   
npvissq .

Los coeficientes diferentes de 0 son 3.

#### i)

Regresión de Ridge con CV 5 veces

set.seed(100)  
library(glmnet)  
x=model.matrix(lbwght~.,datos)[,-1]  
y=datos$lbwght  
y.test=y[test]  
grid=10^seq(10,-2, length =100)  
cv.ridge=cv.glmnet(x[train ,],y[train],alpha=0,lambda=grid, nfolds = 5)  
plot(cv.ridge)



mejorlambda=cv.ridge$lambda.min  
mejorlambda

[1] 0.01

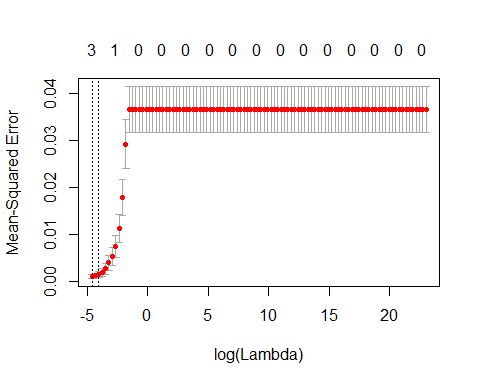
ridge.mod=glmnet(x[train ,],y[train],alpha=0,lambda=grid)  
ridge.pred=predict(ridge.mod,s=mejorlambda ,newx=x[test ,])  
  
error.ridge <- mean((ridge.pred-datos.test[, "lbwght"] )^2)  
error.ridge

[1] 0.0005855056

results2["RIDGE", "5 Cross Validation"] = error.ridge

Modelo de LASSO

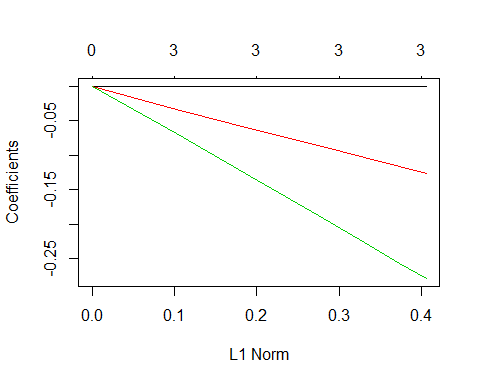
set.seed(100)  
cv.lasso=cv.glmnet(x[train ,],y[train],alpha=1, lambda = grid, nfolds = 5)  
plot(cv.lasso)



bestlam=cv.lasso$lambda.min  
bestlam

[1] 0.01

lasso.mod=glmnet(x[train ,],y[train],alpha=1,lambda=grid)  
plot(lasso.mod)



lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test ,])  
  
error.lasso <- mean((lasso.pred-datos.test[, "lbwght"] )^2)  
error.lasso

[1] 0.000435723

results2["LASSO", "5 Cross Validation"] = error.lasso

Para obtener los coeficientes del modelo para el mejor lambda usamos la función coef

c = coef(lasso.mod, s = bestlam)  
c

21 x 1 sparse Matrix of class "dgCMatrix"  
 1  
(Intercept) 7.1446826417  
mage .   
meduc .   
monpre .   
npvis .   
fage .   
feduc .   
bwght 0.0002865851  
omaps .   
fmaps .   
cigs .   
drink .   
lbw -0.1261450669  
vlbw -0.2789833877  
male .   
mwhte .   
mblck .   
fwhte .   
fblck .   
magesq .   
npvissq .

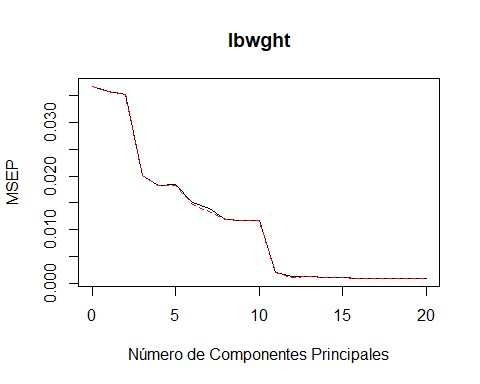
Los coeficientes diferentes de 0 son 3.

#### j)

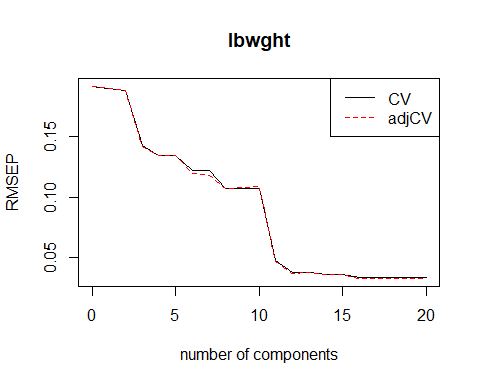
library(pls)  
set.seed(100)  
pcr.fit=pcr(lbwght~., data=datos,subset=train,scale=TRUE, validation="CV")  
summary(pcr.fit)

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: svdpc  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 10 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.1892 0.1875 0.1420 0.1352 0.1354 0.1227  
adjCV 0.1915 0.1892 0.1875 0.1415 0.1350 0.1353 0.1210  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.1179 0.1091 0.1080 0.1078 0.04523 0.03479 0.03552  
adjCV 0.1160 0.1087 0.1078 0.1081 0.04464 0.03412 0.03515  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03409 0.03372 0.03071 0.03073 0.03076 0.03077  
adjCV 0.03396 0.03368 0.03050 0.03052 0.03055 0.03056  
 20 comps  
CV 0.03085  
adjCV 0.03064  
  
TRAINING: % variance explained  
 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
X 17.186 32.312 44.07 53.67 60.79 66.51 71.64  
lbwght 3.228 6.003 47.20 52.10 52.19 66.74 69.55  
 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
X 76.75 80.83 84.77 88.28 90.92 93.30 95.43  
lbwght 70.31 70.49 70.52 94.99 97.18 97.18 97.45  
 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps  
X 97.30 98.86 99.53 99.80 99.97 100.00  
lbwght 97.52 98.03 98.03 98.03 98.03 98.04

validationplot(pcr.fit,val.type="MSEP", xlab = "Número de Componentes Principales")



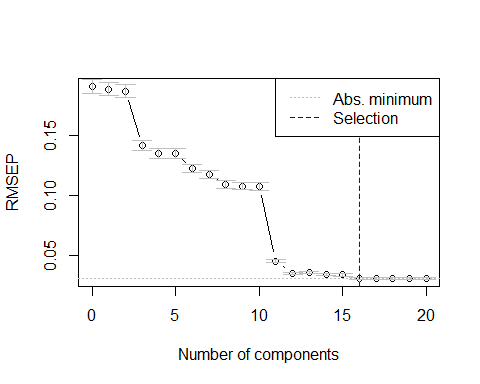
pcr.cv <- crossval(pcr.fit, segments = 10)  
plot(RMSEP(pcr.cv), legendpos="topright")



summary(pcr.cv, what = "validation")

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: svdpc  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 10 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.1896 0.1885 0.1428 0.1344 0.1344 0.1217  
adjCV 0.1915 0.1897 0.1885 0.1420 0.1342 0.1343 0.1195  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.1216 0.1076 0.107 0.1070 0.04754 0.03762 0.03783  
adjCV 0.1179 0.1073 0.108 0.1085 0.04679 0.03684 0.03738  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03610 0.03595 0.03319 0.03320 0.03325 0.03325  
adjCV 0.03582 0.03576 0.03284 0.03286 0.03291 0.03290  
 20 comps  
CV 0.03341  
adjCV 0.03306

## Selecciona el número de componentes principales  
## Regla del codo: 1 d.t.  
ncomp.1.d.t. <- selectNcomp(pcr.fit, method = "onesigma", plot = TRUE, validation = "CV",  
 segments = 10)



ncomp.1.d.t.

[1] 16

pcr.pred=predict(pcr.fit,newdata=x[test,],ncomp=ncomp.1.d.t.)  
error.pcr <- mean((pcr.pred - datos.test[, "lbwght"])^2)  
error.pcr

[1] 0.0005175971

results2["PCA", "10 Cross Validation"] = error.pcr

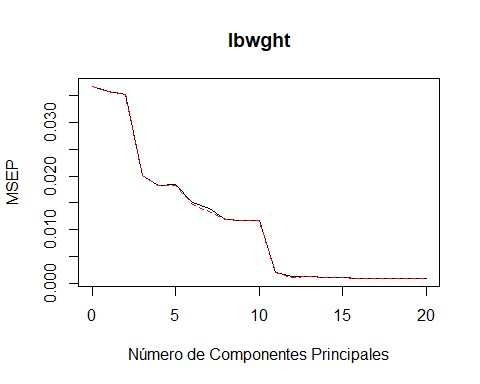
Usando el Metodo de Componentes Principales con una Validación Cruzada 10 veces obtenemos la cantidad de componentes M = 16 y un error de prueba de 5.175971410^{-4}.

El mismo método pero con la validación cruzada 5 veces seria:

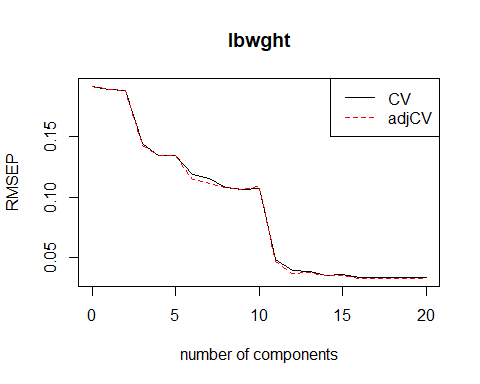
set.seed(100)  
pcr.fit=pcr(lbwght~., data=datos,subset=train,scale=TRUE, validation="CV")  
summary(pcr.fit)

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: svdpc  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 10 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.1892 0.1875 0.1420 0.1352 0.1354 0.1227  
adjCV 0.1915 0.1892 0.1875 0.1415 0.1350 0.1353 0.1210  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.1179 0.1091 0.1080 0.1078 0.04523 0.03479 0.03552  
adjCV 0.1160 0.1087 0.1078 0.1081 0.04464 0.03412 0.03515  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03409 0.03372 0.03071 0.03073 0.03076 0.03077  
adjCV 0.03396 0.03368 0.03050 0.03052 0.03055 0.03056  
 20 comps  
CV 0.03085  
adjCV 0.03064  
  
TRAINING: % variance explained  
 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
X 17.186 32.312 44.07 53.67 60.79 66.51 71.64  
lbwght 3.228 6.003 47.20 52.10 52.19 66.74 69.55  
 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
X 76.75 80.83 84.77 88.28 90.92 93.30 95.43  
lbwght 70.31 70.49 70.52 94.99 97.18 97.18 97.45  
 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps  
X 97.30 98.86 99.53 99.80 99.97 100.00  
lbwght 97.52 98.03 98.03 98.03 98.03 98.04

validationplot(pcr.fit,val.type="MSEP", xlab = "Número de Componentes Principales")



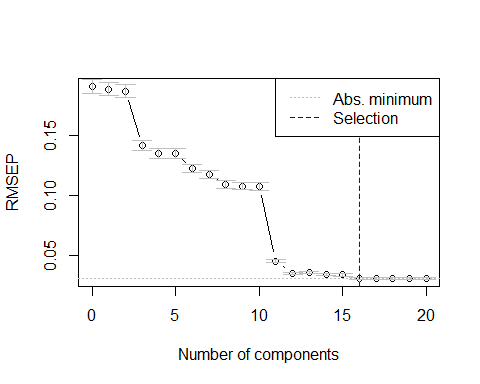
pcr.cv <- crossval(pcr.fit, segments = 5)  
plot(RMSEP(pcr.cv), legendpos="topright")



summary(pcr.cv, what = "validation")

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: svdpc  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 5 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.1890 0.1879 0.1443 0.1344 0.1345 0.1189  
adjCV 0.1915 0.1891 0.1881 0.1425 0.1342 0.1343 0.1149  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.1154 0.1082 0.1063 0.1073 0.04799 0.03897 0.03801  
adjCV 0.1109 0.1077 0.1066 0.1089 0.04688 0.03697 0.03727  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03550 0.03569 0.03342 0.03342 0.03346 0.03346  
adjCV 0.03494 0.03530 0.03270 0.03271 0.03274 0.03274  
 20 comps  
CV 0.03361  
adjCV 0.03288

## Selecciona el número de componentes principales  
## Regla del codo: 1 d.t.  
ncomp.1.d.t. <- selectNcomp(pcr.fit, method = "onesigma", plot = TRUE, validation = "CV",  
 segments = 5)



ncomp.1.d.t.

[1] 16

pcr.pred=predict(pcr.fit,newdata=x[test,],ncomp=ncomp.1.d.t.)  
error.pcr <- mean((pcr.pred - datos.test[, "lbwght"])^2)  
error.pcr

[1] 0.0005175971

results2["PCA", "5 Cross Validation"] = error.pcr

Usando el Metodo de Componentes Principales con una Validación Cruzada 5 veces obtenemos la cantidad de componentes M = 16 y un error de prueba de 5.175971410^{-4}.

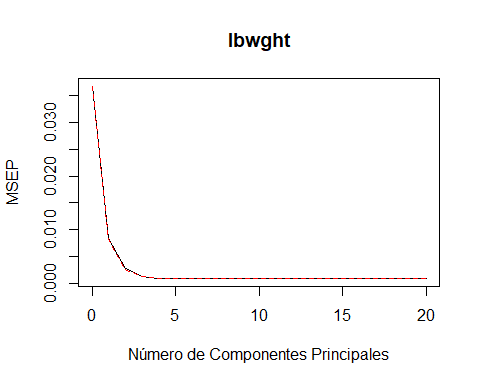
#### k)

PLS con Validación cruzada 10 veces

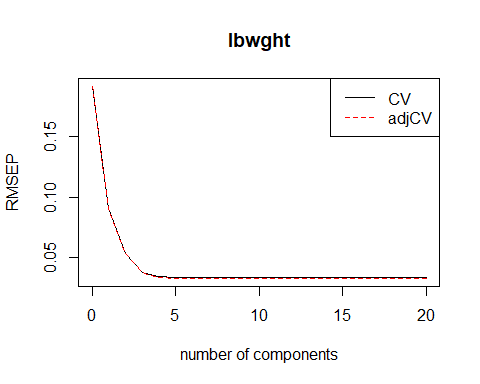
set.seed(100)  
pls.fit=plsr(lbwght~., data=datos,subset=train,scale=TRUE, validation="CV")  
summary(pls.fit)

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: kernelpls  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 10 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.09115 0.05192 0.03511 0.03159 0.03096 0.03079  
adjCV 0.1915 0.09089 0.05173 0.03493 0.03142 0.03076 0.03059  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.03085 0.03072 0.03075 0.03075 0.03078 0.03080 0.03081  
adjCV 0.03063 0.03052 0.03054 0.03054 0.03057 0.03058 0.03059  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03080 0.03080 0.03082 0.03085 0.03084 0.03085  
adjCV 0.03059 0.03059 0.03061 0.03064 0.03063 0.03064  
 20 comps  
CV 0.03085  
adjCV 0.03064  
  
TRAINING: % variance explained  
 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
X 11.77 21.71 32.67 40.98 52.12 60.42 64.80  
lbwght 78.83 93.25 97.25 97.85 97.97 98.01 98.03  
 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
X 68.80 73.23 78.72 80.34 82.46 85.54 86.74  
lbwght 98.03 98.03 98.03 98.03 98.03 98.03 98.04  
 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps  
X 90.90 93.06 94.53 97.35 98.96 100.00  
lbwght 98.04 98.04 98.04 98.04 98.04 98.04

validationplot(pls.fit,val.type="MSEP", xlab = "Número de Componentes Principales")



pls.cv <- crossval(pls.fit, segments = 10)  
plot(RMSEP(pls.cv), legendpos="topright")



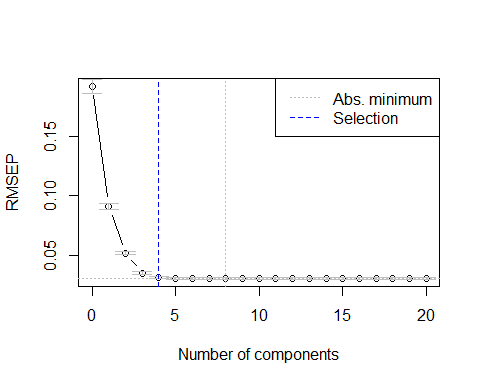
summary(pls.cv, what = "validation")

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: kernelpls  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 10 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.09097 0.05354 0.03803 0.03407 0.03345 0.03326  
adjCV 0.1915 0.09077 0.05322 0.03766 0.03377 0.03312 0.03292  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.03330 0.03322 0.03323 0.03324 0.03325 0.03330 0.03332  
adjCV 0.03296 0.03288 0.03289 0.03290 0.03291 0.03295 0.03297  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03332 0.03335 0.03337 0.03341 0.03339 0.03340  
adjCV 0.03297 0.03300 0.03302 0.03305 0.03304 0.03305  
 20 comps  
CV 0.03341  
adjCV 0.03306

## Utilizamos 4 componentes por el Mínimo Error de VC  
pls.pred=predict(pls.fit,newdata=x[test,],ncomp=4)  
error.pls <- mean((pls.pred - datos.test[, "lbwght"])^2)  
error.pls

[1] 0.0005495759

## Selecciona el número de componentes principales  
## Regla del codo: 1 d.t.  
ncomp.1.d.t. <- selectNcomp(pls.fit, method = "onesigma", plot = TRUE, validation = "CV",  
 segments = 10)



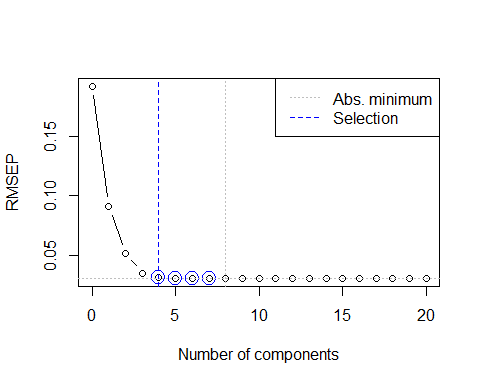
ncomp.1.d.t.

[1] 4

pls.pred.2=predict(pls.fit,newdata=x[test,],ncomp=ncomp.1.d.t.)  
error.pls.codo <- mean((pls.pred.2 - datos.test[, "lbwght"])^2)  
error.pls.codo

[1] 0.0005495759

results2["PLS", "10 Cross Validation"] = error.pls.codo  
  
## Regla de la permutación: se selecciona el ncomp que nos da el min Error de VC  
ncomp.perm <- selectNcomp(pls.fit, method = "randomization", plot = TRUE)



ncomp.perm

[1] 4

pls.pred.3=predict(pls.fit,newdata=x[test,],ncomp=ncomp.perm)  
error.pls.perm <- mean((pls.pred.3 - datos.test[, "lbwght"])^2)  
error.pls.perm

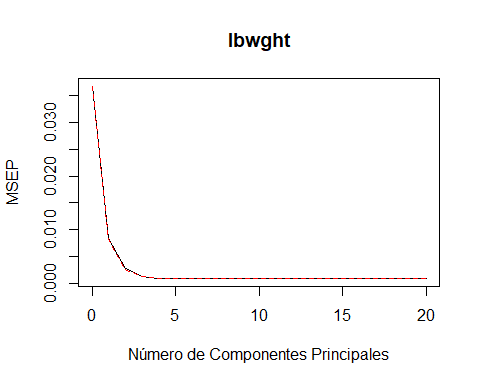
[1] 0.0005495759

PLS con Validación cruzada 5 veces

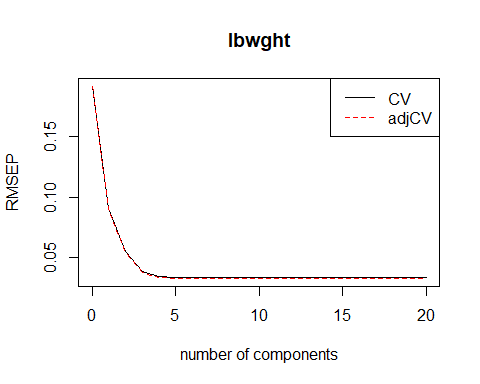
set.seed(100)  
pls.fit=plsr(lbwght~., data=datos,subset=train,scale=TRUE, validation="CV")  
summary(pls.fit)

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: kernelpls  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 10 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.09115 0.05192 0.03511 0.03159 0.03096 0.03079  
adjCV 0.1915 0.09089 0.05173 0.03493 0.03142 0.03076 0.03059  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.03085 0.03072 0.03075 0.03075 0.03078 0.03080 0.03081  
adjCV 0.03063 0.03052 0.03054 0.03054 0.03057 0.03058 0.03059  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03080 0.03080 0.03082 0.03085 0.03084 0.03085  
adjCV 0.03059 0.03059 0.03061 0.03064 0.03063 0.03064  
 20 comps  
CV 0.03085  
adjCV 0.03064  
  
TRAINING: % variance explained  
 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
X 11.77 21.71 32.67 40.98 52.12 60.42 64.80  
lbwght 78.83 93.25 97.25 97.85 97.97 98.01 98.03  
 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
X 68.80 73.23 78.72 80.34 82.46 85.54 86.74  
lbwght 98.03 98.03 98.03 98.03 98.03 98.03 98.04  
 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps  
X 90.90 93.06 94.53 97.35 98.96 100.00  
lbwght 98.04 98.04 98.04 98.04 98.04 98.04

validationplot(pls.fit,val.type="MSEP", xlab = "Número de Componentes Principales")



pls.cv <- crossval(pls.fit, segments = 5)  
plot(RMSEP(pls.cv), legendpos="topright")



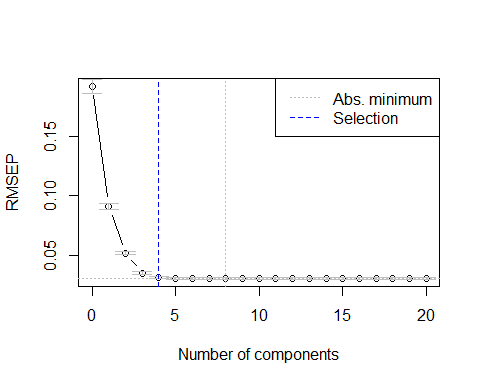
summary(pls.cv, what = "validation")

Data: X dimension: 1128 20   
 Y dimension: 1128 1  
Fit method: kernelpls  
Number of components considered: 20  
  
VALIDATION: RMSEP  
Cross-validated using 5 random segments.  
 (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
CV 0.1915 0.09076 0.05483 0.03861 0.03414 0.03360 0.03339  
adjCV 0.1915 0.09023 0.05378 0.03777 0.03355 0.03292 0.03271  
 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
CV 0.03339 0.03345 0.03345 0.03344 0.03347 0.03351 0.03352  
adjCV 0.03269 0.03273 0.03273 0.03273 0.03275 0.03279 0.03279  
 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
CV 0.03353 0.03356 0.03358 0.03361 0.03360 0.03361  
adjCV 0.03281 0.03283 0.03285 0.03287 0.03287 0.03287  
 20 comps  
CV 0.03361  
adjCV 0.03288

## Utilizamos 4 componentes por el Mínimo Error de VC  
pls.pred=predict(pls.fit,newdata=x[test,],ncomp=4)  
error.pls <- mean((pls.pred - datos.test[, "lbwght"])^2)  
error.pls

[1] 0.0005495759

## Selecciona el número de componentes principales  
## Regla del codo: 1 d.t.  
ncomp.1.d.t. <- selectNcomp(pls.fit, method = "onesigma", plot = TRUE, validation = "CV",  
 segments = 5)



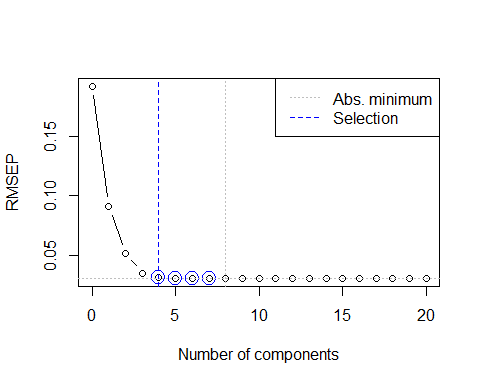
ncomp.1.d.t.

[1] 4

pls.pred.2=predict(pls.fit,newdata=x[test,],ncomp=ncomp.1.d.t.)  
error.pls.codo <- mean((pls.pred.2 - datos.test[, "lbwght"])^2)  
error.pls.codo

[1] 0.0005495759

results2["PLS", "5 Cross Validation"] = error.pls.codo  
  
## Regla de la permutación: se selecciona el ncomp que nos da el min Error de VC  
ncomp.perm <- selectNcomp(pls.fit, method = "randomization", plot = TRUE)



ncomp.perm

[1] 4

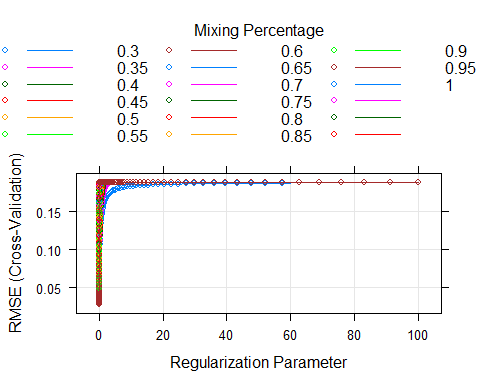
pls.pred.3=predict(pls.fit,newdata=x[test,],ncomp=ncomp.perm)  
error.pls.perm <- mean((pls.pred.3 - datos.test[, "lbwght"])^2)  
error.pls.perm

[1] 0.0005495759

#### l)

Modelo LASSO con la restricción de Red Elástica (LASSO with Elastic Net) en el conjunto de entrenamiento, con el α y el λ elegido mediante la Validación Cruzada 10-Veces.

library(glmnet)  
library(caret)  
set.seed(100)  
lambda.grid <- 10^seq(2,-2, length = 100)  
alpha.grid <- seq(0,1, by = 0.05)  
  
Control <- trainControl(method = "cv", number = 10)  
busca.grid <- expand.grid(alpha = alpha.grid, lambda = lambda.grid)  
  
set.seed(100)  
mi.entrenamiento <- train(lbwght~., data = datos.train, method = "glmnet",   
 tuneGrid = busca.grid, trControl = Control,  
 tuneLength = 10,  
 standardize = TRUE, maxit = 1000000)  
  
plot(mi.entrenamiento)



attributes(mi.entrenamiento)

$names  
 [1] "method" "modelInfo" "modelType" "results"   
 [5] "pred" "bestTune" "call" "dots"   
 [9] "metric" "control" "finalModel" "preProcess"   
[13] "trainingData" "resample" "resampledCM" "perfNames"   
[17] "maximize" "yLimits" "times" "levels"   
[21] "terms" "coefnames" "xlevels"   
  
$class  
[1] "train" "train.formula"

mi.entrenamiento$bestTune

alpha lambda  
901 0.45 0.01

mi.modelo.glmnet <- mi.entrenamiento$finalModel  
coef(mi.modelo.glmnet, s = mi.entrenamiento$bestTune$lambda)

21 x 1 sparse Matrix of class "dgCMatrix"  
 1  
(Intercept) 7.1073653340  
mage .   
meduc .   
monpre .   
npvis .   
fage .   
feduc .   
bwght 0.0002832945  
omaps .   
fmaps 0.0054557468  
cigs .   
drink .   
lbw -0.1559228772  
vlbw -0.3073639506  
male .   
mwhte .   
mblck .   
fwhte .   
fblck .   
magesq .   
npvissq .

mej.modelo <- glmnet(x[train ,],y[train], alpha=mi.entrenamiento$bestTune$alpha,  
 lambda = mi.entrenamiento$bestTune$lambda)  
c = coef(mej.modelo, s = mi.entrenamiento$bestTune$lambda)  
cbind(coef(mej.modelo, s = mi.entrenamiento$bestTune$lambda), coef(mi.modelo.glmnet, s = mi.entrenamiento$bestTune$lambda))

21 x 2 sparse Matrix of class "dgCMatrix"  
 1 1  
(Intercept) 7.1073855303 7.1073653340  
mage . .   
meduc . .   
monpre . .   
npvis . .   
fage . .   
feduc . .   
bwght 0.0002832938 0.0002832945  
omaps . .   
fmaps 0.0054537954 0.0054557468  
cigs . .   
drink . .   
lbw -0.1559405441 -0.1559228772  
vlbw -0.3073522220 -0.3073639506  
male . .   
mwhte . .   
mblck . .   
fwhte . .   
fblck . .   
magesq . .   
npvissq . .

lre.pred <- predict(mej.modelo,s=mi.entrenamiento$bestTune$lambda,newx=x[test ,])  
  
error.lassoelastic <- mean((lre.pred - datos.test[, "lbwght"])^2)  
error.lassoelastic

[1] 0.0004796094

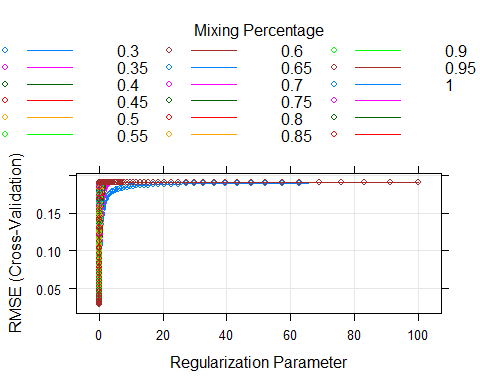
results2["LASSO with Elastic Net", "10 Cross Validation"] = error.lassoelastic

El error de prueba obtenido es 4.796094410^{-4} y la cantidad de coeficientes distintos de 0 es 4.

#### m)

Modelo LASSO con la restricción de Red Elástica (LASSO with Elastic Net) en el conjunto de entrenamiento, con el α y el λ elegido mediante la Validación Cruzada 5-Veces.

set.seed(100)  
lambda.grid <- 10^seq(2,-2, length = 100)  
alpha.grid <- seq(0,1, by = 0.05)  
Control <- trainControl(method = "cv", number = 5)  
busca.grid <- expand.grid(alpha = alpha.grid, lambda = lambda.grid)  
set.seed(100)  
mi.entrenamiento <- train(lbwght~., data = datos.train, method = "glmnet",   
 tuneGrid = busca.grid, trControl = Control,  
 tuneLength = 5,  
 standardize = TRUE, maxit = 1000000)  
  
plot(mi.entrenamiento)



attributes(mi.entrenamiento)

$names  
 [1] "method" "modelInfo" "modelType" "results"   
 [5] "pred" "bestTune" "call" "dots"   
 [9] "metric" "control" "finalModel" "preProcess"   
[13] "trainingData" "resample" "resampledCM" "perfNames"   
[17] "maximize" "yLimits" "times" "levels"   
[21] "terms" "coefnames" "xlevels"   
  
$class  
[1] "train" "train.formula"

mi.entrenamiento$bestTune

alpha lambda  
701 0.35 0.01

mi.modelo.glmnet <- mi.entrenamiento$finalModel  
coef(mi.modelo.glmnet, s = mi.entrenamiento$bestTune$lambda)

21 x 1 sparse Matrix of class "dgCMatrix"  
 1  
(Intercept) 7.0950292590  
mage .   
meduc .   
monpre .   
npvis .   
fage .   
feduc .   
bwght 0.0002826996  
omaps .   
fmaps 0.0070610993  
cigs .   
drink .   
lbw -0.1609479918  
vlbw -0.3115200778  
male .   
mwhte .   
mblck .   
fwhte .   
fblck .   
magesq .   
npvissq .

mej.modelo <- glmnet(x[train ,],y[train], alpha=mi.entrenamiento$bestTune$alpha,  
 lambda = mi.entrenamiento$bestTune$lambda)  
c = coef(mej.modelo, s = mi.entrenamiento$bestTune$lambda)  
cbind(coef(mej.modelo, s = mi.entrenamiento$bestTune$lambda), coef(mi.modelo.glmnet, s = mi.entrenamiento$bestTune$lambda))

21 x 2 sparse Matrix of class "dgCMatrix"  
 1 1  
(Intercept) 7.0950610984 7.0950292590  
mage . .   
meduc . .   
monpre . .   
npvis . .   
fage . .   
feduc . .   
bwght 0.0002826972 0.0002826996  
omaps . .   
fmaps 0.0070584853 0.0070610993  
cigs . .   
drink . .   
lbw -0.1609638819 -0.1609479918  
vlbw -0.3115141652 -0.3115200778  
male . .   
mwhte . .   
mblck . .   
fwhte . .   
fblck . .   
magesq . .   
npvissq . .

lre.pred <- predict(mej.modelo,s=mi.entrenamiento$bestTune$lambda,newx=x[test ,])  
  
error.lassoelastic <- mean((lre.pred - datos.test[, "lbwght"])^2)  
error.lassoelastic

[1] 0.0004959331

results2["LASSO with Elastic Net", "5 Cross Validation"] = error.lassoelastic

El error de prueba obtenido es 4.959330910^{-4} y la cantidad de coeficientes distintos de 0 es 4.

#### o)

Ajusta un modelo Rigorous LASSO (RLASSO) sobre el conjunto de entrenamiento con el lambda elegido mediante la penalización dependiente de los datos

Call:  
rlasso.default(x = x, y = y, post = FALSE, X.dependent.lambda = TRUE)  
  
Post-Lasso Estimation: FALSE   
  
Total number of variables: 20  
Number of selected variables: 1   
  
Residuals:   
 Min 1Q Median 3Q Max   
-1.070893 0.002102 0.011650 0.015554 0.016828   
  
 Estimate  
(Intercept) 7.154  
bwght 0.000  
  
Residual standard error: 0.05398  
Multiple R-squared: 0.9205  
Adjusted R-squared: 0.9204  
Joint significance test:  
 the sup score statistic for joint significance test is 3516 with a p-value of 0

Call:  
rlasso.default(x = x, y = y, post = FALSE, X.dependent.lambda = TRUE)  
  
Coefficients:  
(Intercept) mage meduc monpre npvis   
 7.1543986 0.0000000 0.0000000 0.0000000 0.0000000   
 fage feduc bwght omaps fmaps   
 0.0000000 0.0000000 0.0002827 0.0000000 0.0000000   
 cigs drink lbw vlbw male   
 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000   
 mwhte mblck fwhte fblck magesq   
 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000   
 npvissq   
 0.0000000

Informa el error de prueba obtenido junto con el número de coeficientes estimados diferentes de cero.

Total de variables = **20** Variables seleccionadas = **1** p-value = **0**

[1] 0.0009358474

El error de prueba de Rigorous Lasso con penalización dependiente de los datos es 9.358473810^{-4}

Con el lambda elegido mediante la penalización independiente de los datos

Call:  
rlasso.default(x = x, y = y, post = FALSE)  
  
Post-Lasso Estimation: FALSE   
  
Total number of variables: 20  
Number of selected variables: 1   
  
Residuals:   
 Min 1Q Median 3Q Max   
-1.070893 0.002102 0.011650 0.015554 0.016828   
  
 Estimate  
(Intercept) 7.154  
bwght 0.000  
  
Residual standard error: 0.05398  
Multiple R-squared: 0.9205  
Adjusted R-squared: 0.9204  
Joint significance test:  
 the sup score statistic for joint significance test is 3516 with a p-value of 0

Call:  
rlasso.default(x = x, y = y, post = FALSE)  
  
Coefficients:  
(Intercept) mage meduc monpre npvis   
 7.1543986 0.0000000 0.0000000 0.0000000 0.0000000   
 fage feduc bwght omaps fmaps   
 0.0000000 0.0000000 0.0002827 0.0000000 0.0000000   
 cigs drink lbw vlbw male   
 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000   
 mwhte mblck fwhte fblck magesq   
 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000   
 npvissq   
 0.0000000

Informa el error de prueba obtenido junto con el número de coeficientes estimados diferentes de cero.

Total de variables = **20** Variables seleccionadas = **1** p-value = **0**

[1] 0.0009358474

El error de prueba de Rigorous Lasso con penalización independiente de los datos es 9.358473810^{-4}

Calcula el error de prueba de los dos modelos mediante el ajuste de Post-LASSO.

Call:  
rlasso.default(x = x, y = y, post = TRUE)  
  
Post-Lasso Estimation: TRUE   
  
Total number of variables: 20  
Number of selected variables: 2   
  
Residuals:   
 Min 1Q Median 3Q Max   
-0.726393 -0.003614 0.006465 0.011053 0.193903   
  
 Estimate  
(Intercept) 7.094  
bwght 0.000  
lbw -0.293  
  
Residual standard error: 0.03454  
Multiple R-squared: 0.9675  
Adjusted R-squared: 0.9674  
Joint significance test:  
 the sup score statistic for joint significance test is 3516 with a p-value of 0

Call:  
rlasso.default(x = x, y = y, post = TRUE)  
  
Coefficients:  
(Intercept) mage meduc monpre npvis   
 7.0936376 0.0000000 0.0000000 0.0000000 0.0000000   
 fage feduc bwght omaps fmaps   
 0.0000000 0.0000000 0.0003018 0.0000000 0.0000000   
 cigs drink lbw vlbw male   
 0.0000000 0.0000000 -0.2934100 0.0000000 0.0000000   
 mwhte mblck fwhte fblck magesq   
 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000   
 npvissq   
 0.0000000

Call:  
rlasso.default(x = x, y = y, post = TRUE)  
  
(Intercept) bwght lbw   
 7.0936376 0.0003018 -0.2934100

Informa el error de prueba obtenido junto con el número de coeficientes estimados diferentes de cero.

Total de variables = **20** Variables seleccionadas = **1** p-value = **0**

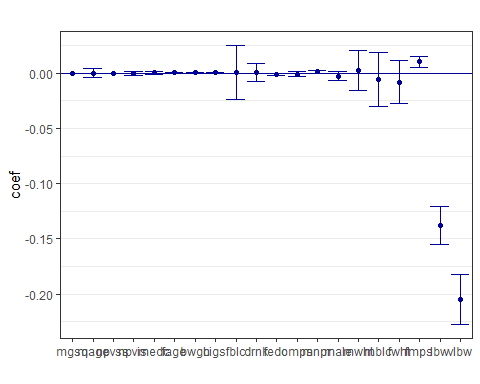
[1] 0.0005005301

El error de prueba Post-Lasso es 5.005300510^{-4}

#### p)

Contrastar la significación individual de los coeficientes estimados del modelo final. Utiliza el nivel de significación del 5%.

[1] "Estimates and significance testing of the effect of target variables"  
 Estimate. Std. Error t value Pr(>|t|)   
mage 3.843e-05 2.058e-03 0.019 0.985100   
meduc 2.686e-04 6.252e-04 0.430 0.667424   
monpre 1.284e-03 8.585e-04 1.495 0.134885   
npvis -2.455e-04 8.602e-04 -0.285 0.775333   
fage 2.723e-04 2.544e-04 1.070 0.284482   
feduc -8.788e-04 5.669e-04 -1.550 0.121109   
bwght 2.976e-04 1.583e-06 188.019 < 2e-16 \*\*\*  
omaps -9.043e-04 1.125e-03 -0.804 0.421432   
fmaps 1.023e-02 2.644e-03 3.871 0.000108 \*\*\*  
cigs 4.469e-04 2.729e-04 1.638 0.101460   
drink 7.396e-04 4.023e-03 0.184 0.854123   
lbw -1.380e-01 8.776e-03 -15.727 < 2e-16 \*\*\*  
vlbw -2.048e-01 1.159e-02 -17.672 < 2e-16 \*\*\*  
male -2.454e-03 2.057e-03 -1.193 0.232796   
mwhte 2.616e-03 9.354e-03 0.280 0.779745   
mblck -5.379e-03 1.251e-02 -0.430 0.667106   
fwhte -7.936e-03 1.010e-02 -0.786 0.431914   
fblck 6.145e-04 1.250e-02 0.049 0.960801   
magesq 6.233e-06 3.356e-05 0.186 0.852649   
npvissq -4.377e-05 2.683e-05 -1.632 0.102755   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

 Sólo las variables \***bwght, fmaps, lbw y vlbw** son significativas sobre un nivel de confianza del 1%

Ahora eliminamos las variables no significativas

[1] "Estimates and significance testing of the effect of target variables"  
 Estimate. Std. Error t value Pr(>|t|)   
bwght 2.976e-04 1.583e-06 188.019 < 2e-16 \*\*\*  
fmaps 1.058e-02 1.832e-03 5.775 7.68e-09 \*\*\*  
lbw -1.541e-01 8.984e-03 -17.148 < 2e-16 \*\*\*  
vlbw -3.310e-01 1.448e-02 -22.861 < 2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Las cuatro variables son muy significativas.

Informa el error de prueba obtenido

[1] 0.0007593627

El error de prueba obtenido es 7.593626810^{-4}

Basándose en contrastes de significación individual, selecciona modelos que contengan sólo variables significativas al 5% de significación y estima el error de prueba de estos modelos (mediante el ajuste de Post-LASSO).

Call:  
rlasso.default(x = x.nuevo.1, y = y, post = TRUE)  
  
Post-Lasso Estimation: TRUE   
  
Total number of variables: 4  
Number of selected variables: 4   
  
Residuals:   
 Min 1Q Median 3Q Max   
-0.514608 -0.004064 0.005844 0.010699 0.261923   
  
 Estimate  
(Intercept) 7.012  
bwght 0.000  
fmaps 0.011  
lbw -0.154  
vlbw -0.331  
  
Residual standard error: 0.02703  
Multiple R-squared: 0.9801  
Adjusted R-squared: 0.98  
Joint significance test:  
 the sup score statistic for joint significance test is 3516 with a p-value of 0

[1] 0.000511914

El error de prueba de estos modelos es 5.119140210^{-4}

#### q)

A continuación se muestran los modelos utilizados anteriormente con el tipo de validación cruzada y su error de prueba correspondiente.

kable(results2)

|  |  |  |  |
| --- | --- | --- | --- |
|  | None | 5 Cross Validation | 10 Cross Validation |
| RIDGE | NA | 0.0005855 | 0.0005855 |
| LASSO | NA | 0.0004357 | 0.0004357 |
| PCA | NA | 0.0005176 | 0.0005176 |
| PLS | NA | 0.0005496 | 0.0005496 |
| LASSO with Elastic Net | NA | 0.0004959 | 0.0004796 |
| Rigorous LASSO Independent L | 0.0009358 | NA | NA |
| Rigorous LASSO Dependent L | 0.0009358 | NA | NA |
| Rigorous LASSO Post-Lasso | 0.0005005 | NA | NA |
| Como sucede en el inciso **f** | no hay mucha | diferencia en el err | or de prueba de los modelos analizados. |