m565A4

Vinay Vaida

2023-10-19

options(repos = c(CRAN = "https://cran.r-project.org/"))  
#chooseCRANmirror(ind = 1) # Choose the first mirror

* 1. In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

1. Use the rnorm() function to generate a predictor of length , as well as a noise vector of length .

# Set a seed for reproducibility (for easy debugging of the code and long-term use)  
set.seed(26)  
# Generate a predictor variable X of length n = 100  
X <- rnorm(n = 100, mean = 0, sd = 1)  
head(X)

## [1] -2.1298417 1.1478961 -0.4895019 0.8263438 -0.4099352 0.1487879

# Generate a noise vector epsilon of length n = 100  
e <- rnorm(n = 100, mean = 0, sd = 1)  
head(e)

## [1] -0.44047513 0.89066641 -0.06381945 -0.50242994 -0.45432984 -0.58985570

1. Generate a response vector of length according to the model , where , and are constants of your choice.

# Define your chosen constants  
beta\_0 <- 3 # beta\_0 value  
beta\_1 <- 6 #beta\_1 value  
beta\_2 <- 9 #beta\_2 value  
beta\_3 <- 12 #beta\_3 value  
  
  
# Calculate the response vector Y using the specified model  
Y <- beta\_0 + beta\_1 \* X + beta\_2 \* X^2 + beta\_3 \* X^3 + e  
  
#First few values of Y  
head(Y)

## [1] -85.3308101 40.7875449 0.7481914 20.3723970 0.7718208 3.5416382

1. Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors .

Note that you will need to use the data.frame() function to create a single data set containing both and all the powers in consideration, and . Example: for use, in the data.frame, .

Use regsubsets() with nbest = 3,really.big = T, nvmax = 10

# Create a data frame with X, X^2, ..., X^10 and Y  
#mydata <- data.frame(X, X2 = X^2, X3 = X^3, X4 = X^4, X5 = X^5, X6 = X^6, X7 = X^7, X8 = X^8, X9 = X^9, X10 = X^10, Y)  
  
  
X\_df <- data.frame(  
 X = X,  
 X2 = X^2,  
 X3 = X^3,  
 X4 = X^4,  
 X5 = X^5,  
 X6 = X^6,  
 X7 = X^7,  
 X8 = X^8,  
 X9 = X^9,  
 X10 = X^10  
)  
  
Y\_df <- data.frame(Y)  
  
mydata<-cbind(Y\_df, X\_df)  
  
# Load the 'leaps' library  
install.packages("leaps")

## Installing package into 'C:/Users/vinay/AppData/Local/R/win-library/4.3'  
## (as 'lib' is unspecified)

## package 'leaps' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'leaps'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying  
## C:\Users\vinay\AppData\Local\R\win-library\4.3\00LOCK\leaps\libs\x64\leaps.dll  
## to C:\Users\vinay\AppData\Local\R\win-library\4.3\leaps\libs\x64\leaps.dll:  
## Permission denied

## Warning: restored 'leaps'

##   
## The downloaded binary packages are in  
## C:\Users\vinay\AppData\Local\Temp\RtmpQNXMyM\downloaded\_packages

library(leaps)  
  
# Use regsubsets() to perform best subset selection  
best <- regsubsets(Y ~ ., data = mydata, nbest = 3, really.big = TRUE, nvmax = 10)  
  
head(mydata)

## Y X X2 X3 X4 X5  
## 1 -85.3308101 -2.1298417 4.53622582 -9.661443080 2.057734e+01 -4.382649e+01  
## 2 40.7875449 1.1478961 1.31766534 1.512542842 1.736242e+00 1.993025e+00  
## 3 0.7481914 -0.4895019 0.23961207 -0.117290554 5.741394e-02 -2.810423e-02  
## 4 20.3723970 0.8263438 0.68284407 0.564263960 4.662760e-01 3.853043e-01  
## 5 0.7718208 -0.4099352 0.16804688 -0.068888337 2.823976e-02 -1.157647e-02  
## 6 3.5416382 0.1487879 0.02213783 0.003293841 4.900836e-04 7.291850e-05  
## X6 X7 X8 X9 X10  
## 1 9.334348e+01 -1.988068e+02 4.234271e+02 -9.018327e+02 1.920761e+03  
## 2 2.287786e+00 2.626140e+00 3.014536e+00 3.460374e+00 3.972150e+00  
## 3 1.375707e-02 -6.734113e-03 3.296361e-03 -1.613575e-03 7.898479e-04  
## 4 3.183938e-01 2.631028e-01 2.174133e-01 1.796582e-01 1.484594e-01  
## 5 4.745603e-03 -1.945390e-03 7.974838e-04 -3.269167e-04 1.340147e-04  
## 6 1.084939e-05 1.614258e-06 2.401820e-07 3.573616e-08 5.317108e-09

Before proceeding, follow structure in file modelselection Example: Heights. The goal is to find the best model obtained according to Cp, BIC, and adjusted .

Report the coefficients of the best model obtained.

See the results:

# Get the summary of the best models  
sbest <- summary(best)  
sbest

## Subset selection object  
## Call: regsubsets.formula(Y ~ ., data = mydata, nbest = 3, really.big = TRUE,   
## nvmax = 10)  
## 10 Variables (and intercept)  
## Forced in Forced out  
## X FALSE FALSE  
## X2 FALSE FALSE  
## X3 FALSE FALSE  
## X4 FALSE FALSE  
## X5 FALSE FALSE  
## X6 FALSE FALSE  
## X7 FALSE FALSE  
## X8 FALSE FALSE  
## X9 FALSE FALSE  
## X10 FALSE FALSE  
## 3 subsets of each size up to 10  
## Selection Algorithm: exhaustive  
## X X2 X3 X4 X5 X6 X7 X8 X9 X10  
## 1 ( 1 ) " " " " "\*" " " " " " " " " " " " " " "  
## 1 ( 2 ) " " " " " " " " "\*" " " " " " " " " " "  
## 1 ( 3 ) "\*" " " " " " " " " " " " " " " " " " "  
## 2 ( 1 ) " " "\*" "\*" " " " " " " " " " " " " " "  
## 2 ( 2 ) " " " " "\*" "\*" " " " " " " " " " " " "  
## 2 ( 3 ) " " " " "\*" " " " " "\*" " " " " " " " "  
## 3 ( 1 ) "\*" "\*" "\*" " " " " " " " " " " " " " "  
## 3 ( 2 ) " " "\*" "\*" " " "\*" " " " " " " " " " "  
## 3 ( 3 ) " " "\*" "\*" " " " " " " "\*" " " " " " "  
## 4 ( 1 ) "\*" "\*" "\*" " " "\*" " " " " " " " " " "  
## 4 ( 2 ) "\*" "\*" "\*" " " " " " " "\*" " " " " " "  
## 4 ( 3 ) "\*" "\*" "\*" " " " " " " " " " " "\*" " "  
## 5 ( 1 ) "\*" "\*" "\*" " " "\*" "\*" " " " " " " " "  
## 5 ( 2 ) "\*" "\*" "\*" "\*" "\*" " " " " " " " " " "  
## 5 ( 3 ) "\*" "\*" "\*" " " "\*" " " " " "\*" " " " "  
## 6 ( 1 ) "\*" "\*" "\*" " " "\*" " " " " "\*" " " "\*"  
## 6 ( 2 ) "\*" "\*" "\*" " " " " " " "\*" "\*" " " "\*"  
## 6 ( 3 ) "\*" "\*" "\*" " " " " " " " " "\*" "\*" "\*"  
## 7 ( 1 ) "\*" "\*" "\*" " " " " "\*" "\*" "\*" " " "\*"  
## 7 ( 2 ) "\*" "\*" "\*" " " " " "\*" " " "\*" "\*" "\*"  
## 7 ( 3 ) "\*" "\*" "\*" " " "\*" "\*" " " "\*" " " "\*"  
## 8 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 8 ( 2 ) "\*" "\*" "\*" "\*" " " "\*" "\*" "\*" " " "\*"  
## 8 ( 3 ) "\*" "\*" "\*" " " "\*" "\*" " " "\*" "\*" "\*"  
## 9 ( 1 ) "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*"  
## 9 ( 2 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" "\*"  
## 9 ( 3 ) "\*" "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"  
## 10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

See what is inside sbest:

names(sbest)

## [1] "which" "rsq" "rss" "adjr2" "cp" "bic" "outmat" "obj"

To see the models and their Cp: (you can do the same with adjr2 and bic)

cbind(sbest$which ,sbest$cp)

## (Intercept) X X2 X3 X4 X5 X6 X7 X8 X9 X10   
## 1 1 0 0 1 0 0 0 0 0 0 0 2.400749e+04  
## 1 1 0 0 0 0 1 0 0 0 0 0 7.555906e+04  
## 1 1 1 0 0 0 0 0 0 0 0 0 1.079891e+05  
## 2 1 0 1 1 0 0 0 0 0 0 0 1.493029e+03  
## 2 1 0 0 1 1 0 0 0 0 0 0 3.773410e+03  
## 2 1 0 0 1 0 0 1 0 0 0 0 7.501201e+03  
## 3 1 1 1 1 0 0 0 0 0 0 0 4.858075e+00  
## 3 1 0 1 1 0 1 0 0 0 0 0 5.873263e+02  
## 3 1 0 1 1 0 0 0 1 0 0 0 7.797694e+02  
## 4 1 1 1 1 0 1 0 0 0 0 0 4.250565e+00  
## 4 1 1 1 1 0 0 0 1 0 0 0 4.256760e+00  
## 4 1 1 1 1 0 0 0 0 0 1 0 4.259202e+00  
## 5 1 1 1 1 0 1 1 0 0 0 0 5.772062e+00  
## 5 1 1 1 1 1 1 0 0 0 0 0 5.796139e+00  
## 5 1 1 1 1 0 1 0 0 1 0 0 5.868293e+00  
## 6 1 1 1 1 0 1 0 0 1 0 1 7.002594e+00  
## 6 1 1 1 1 0 0 0 1 1 0 1 7.097361e+00  
## 6 1 1 1 1 0 0 0 0 1 1 1 7.194418e+00  
## 7 1 1 1 1 0 0 1 1 1 0 1 5.472460e+00  
## 7 1 1 1 1 0 0 1 0 1 1 1 5.497651e+00  
## 7 1 1 1 1 0 1 1 0 1 0 1 5.802417e+00  
## 8 1 1 1 1 1 0 1 0 1 1 1 7.322956e+00  
## 8 1 1 1 1 1 0 1 1 1 0 1 7.361392e+00  
## 8 1 1 1 1 0 1 1 0 1 1 1 7.407971e+00  
## 9 1 1 1 1 0 1 1 1 1 1 1 9.145029e+00  
## 9 1 1 1 1 1 1 1 0 1 1 1 9.258747e+00  
## 9 1 1 1 1 1 0 1 1 1 1 1 9.290266e+00  
## 10 1 1 1 1 1 1 1 1 1 1 1 1.100000e+01

Follow the notes in modelselection Example: Heights to see how to select the best model according to the various metrics.

mybestmodel<-function(Xnames,Yname,dataset,p,crit="bic"){  
 if(crit=="Cp"){  
 n<-dim(dataset)[1]  
 fullMSE=summary(lm(as.formula(paste(Yname,"~.")),data=dataset))$sigma^2  
 }  
 varsel<-lapply(0:p, function(x) combn(p,x))   
   
 modcrit<-numeric(p); form<-character(p)  
 for(k in 1:p){  
 s<-dim(varsel[[k+1]])[2]   
 tempform<-character(s); tempcrit<-numeric(s)  
 for(j in 1:s){  
 temp <- Xnames[varsel[[k+1]][,j]]  
 tempform[j]<- ifelse(length(temp)>1,  
 paste(temp, collapse = " + "), temp)  
 tempform[j] <- paste(Yname, tempform[j],sep='~')  
 tempmod<-lm(as.formula(tempform[j]),data=dataset)  
 if(crit=="aic"){  
 tempcrit[j] <- AIC(tempmod)  
 }  
 if(crit=="bic"){  
 tempcrit[j] <- BIC(tempmod)  
 }  
 if(crit=="r2"){  
 tempcrit[j] <- summary(tempmod)$adj  
 }  
 if(crit=="Cp"){  
 tempcrit[j]<-sum(tempmod$resˆ2)/fullMSE+2\*(k+1)-n  
 }  
 }  
 # best model of size k  
 if(crit %in% c("aic", "bic")){  
 best<-which.min(tempcrit)  
 }  
 if(crit == "r2"){  
 best<-which.max(tempcrit)  
 }  
 if(crit=="Cp"){  
 best<-which.min(abs(tempcrit[j]-(k+1)))  
 }  
 form[k]<-tempform[best]  
 modcrit[k]<-tempcrit[best]  
 }  
 if(crit %in% c("aic", "bic")){  
 out<-form[which.min(modcrit)]  
 }  
 if(crit == "r2"){  
 out<-form[which.max(modcrit)]  
 }  
 if(crit=="Cp"){  
 out<-form[which.min(abs(modcrit[-p]-(2:p)))]  
 }  
 return(out)  
}  
  
p<-length(names(mydata))-1   
Xnames<-names(mydata)[-1]  
Yname<-"Y"  
dataset<-mydata  
bicform<-mybestmodel(Xnames, Yname, mydata, p, crit="bic")  
bicform

## [1] "Y~X + X2 + X3"

Criteria<-function(model){  
 out<-data.frame(`p+1`=length(model$coef),  
 R2adj=summary(model)$adj,  
 AIC=AIC(model),  
 BIC=BIC(model))  
 return(out)  
}  
  
modbic<-lm(as.formula(bicform), data=mydata)  
modbic

##   
## Call:  
## lm(formula = as.formula(bicform), data = mydata)  
##   
## Coefficients:  
## (Intercept) X X2 X3   
## 3.036 6.003 8.976 12.024

aicform<-mybestmodel(Xnames, Yname, mydata, p, crit="aic")  
aicform

## [1] "Y~X + X2 + X3 + X5"

modaic<-lm(as.formula(aicform), data=mydata)  
modaic

##   
## Call:  
## lm(formula = as.formula(aicform), data = mydata)  
##   
## Coefficients:  
## (Intercept) X X2 X3 X5   
## 3.05232 5.71570 8.96151 12.24614 -0.02839

cpform<-mybestmodel(Xnames, Yname, mydata, p, crit="Cp")  
cpform

## [1] "Y~X + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9"

modCp<-lm(as.formula(cpform), data=mydata)  
modCp

##   
## Call:  
## lm(formula = as.formula(cpform), data = mydata)  
##   
## Coefficients:  
## (Intercept) X X2 X3 X4 X5   
## 2.96265 5.53876 9.88416 12.74037 -0.93010 -0.34488   
## X6 X7 X8 X9   
## 0.25157 0.06833 -0.01958 -0.00484

r2form<-mybestmodel(Xnames, Yname, mydata, p, crit="r2")  
r2form

## [1] "Y~X + X2 + X3 + X6 + X7 + X8 + X10"

modr2<-lm(as.formula(r2form), data=mydata)  
modr2

##   
## Call:  
## lm(formula = as.formula(r2form), data = mydata)  
##   
## Coefficients:  
## (Intercept) X X2 X3 X6 X7   
## 3.000075 5.739624 9.325135 12.212488 -0.239550 -0.007475   
## X8 X10   
## 0.078506 -0.006353

1. Now find the best model using backward stepwise selection.

#backwards  
modback<-step(lm(Y~.,data=mydata),trace=0, direction = "backward")

1. Now fit a lasso model to the simulated data, again using as predictors. Use cross-validation to select the optimal value of . Create plots of the cross-validation error as a function of . Report the resulting coefficient estimates, and discuss the results obtained.

install.packages("glmnet")

## Installing package into 'C:/Users/vinay/AppData/Local/R/win-library/4.3'  
## (as 'lib' is unspecified)

## package 'glmnet' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'glmnet'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying  
## C:\Users\vinay\AppData\Local\R\win-library\4.3\00LOCK\glmnet\libs\x64\glmnet.dll  
## to C:\Users\vinay\AppData\Local\R\win-library\4.3\glmnet\libs\x64\glmnet.dll:  
## Permission denied

## Warning: restored 'glmnet'

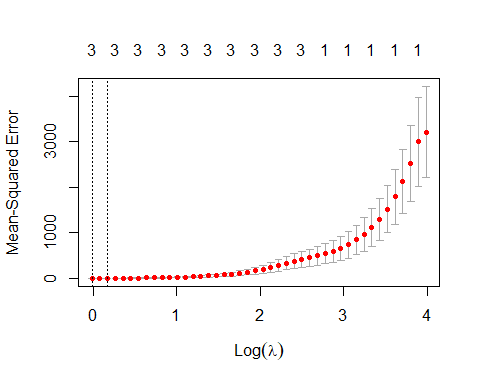
##   
## The downloaded binary packages are in  
## C:\Users\vinay\AppData\Local\Temp\RtmpQNXMyM\downloaded\_packages

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

X\_matrix <- as.matrix(X\_df)  
Y\_vector <- as.vector(Y\_df$Y)  
  
lasso.mod <- cv.glmnet(X\_matrix, Y\_vector, alpha = 1)  
plot(lasso.mod)



best\_lambda <- lasso.mod$lambda.min  
cat("Optimal lambda:", best\_lambda, "\n")

## Optimal lambda: 0.9871121

lasso\_coef <- coef(lasso.mod, s = best\_lambda)  
cat("Coefficient estimates for the optimal lambda:\n")

## Coefficient estimates for the optimal lambda:

print(lasso\_coef)

## 11 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 3.757670  
## X 5.484885  
## X2 8.270673  
## X3 11.870624  
## X4 .   
## X5 .   
## X6 .   
## X7 .   
## X8 .   
## X9 .   
## X10 .

1. Display the adjR^2, BIC and AIC for all the models

rbind(bicm=Criteria(modbic),  
 aicm=Criteria(modaic),  
 adjr2m=Criteria(modr2),  
 cpm=Criteria(modCp),  
 bsel=Criteria(modback)  
)

## p.1 R2adj AIC BIC  
## bicm 4 0.9997263 274.7041 287.7300  
## aicm 5 0.9997308 273.9751 289.6061  
## adjr2m 8 0.9997361 274.7725 298.2190  
## cpm 10 0.9997277 279.7123 308.3691  
## bsel 8 0.9997361 274.8007 298.2472

1. Now use cross validation to select the final model.

cv.lm <- function(data, formulae, nfolds = 5) {  
 data <- na.omit(data) # remove missing values  
 formulae <- sapply(formulae, as.formula)  
 n <- nrow(data)  
 fold.labels <- sample(rep(1:nfolds, length.out = n))  
 mses <- matrix(NA, nrow = nfolds, ncol = length(formulae))  
 colnames <- as.character(formulae)  
 for (fold in 1:nfolds) {  
 test.rows <- which(fold.labels == fold)  
 train <- data[-test.rows, ]  
 test <- data[test.rows, ]  
 for (form in 1:length(formulae)) {  
 current.model <- lm(formula = formulae[[form]], data = train)  
 predictions <- predict(current.model, newdata = test)  
 test.responses <- eval(formulae[[form]][[2]], envir = test)  
 test.errors <- test.responses - predictions  
 mses[fold, form] <- mean(test.errors^2)  
 }  
 }  
 return(colMeans(mses))  
}  
  
set.seed(1999)  
formulae<-c(formula(modbic),  
 formula(modaic),  
 formula(modr2),  
 formula(modCp),  
 formula(modback))  
  
mse<-cv.lm(data=mydata, formulae, nfolds = 5)  
print(mse)

## [1] 0.9271315 0.9098586 5.5689550 193.7407091 5.0457759

From the above obtained values, we observe that the model with AIC metric has the least error value. Therefore, the model with AIC is the final model.

Following are the Error values in order:

modelAIC > modelBIC > modback > modr2 > modCp

* 1. The perils of post-selection inference, and data splitting to the rescue.

1. Generate a 1000 x 101 array, where all the entries are IID standard Gaussian variables.

We’ll call the first column the response variable , and the others the predictors .

mydata<-as.data.frame(matrix(rnorm(1000\*101),ncol=101))  
names(mydata)<-c("Y",paste("X",1:100,sep=""))  
mydata[1:3,1:5]

## Y X1 X2 X3 X4  
## 1 -0.2642307 1.7613338 -1.0921775 0.01325845 0.2512596  
## 2 -0.8245672 -1.1632197 -1.0310507 -0.44688678 -0.2315223  
## 3 0.1615689 0.3593122 -0.5022663 -0.84728363 -0.5650197

By design, there is no true relationship between the response and the predictors (but all the usual linear-Gaussian-modeling assumptions hold).

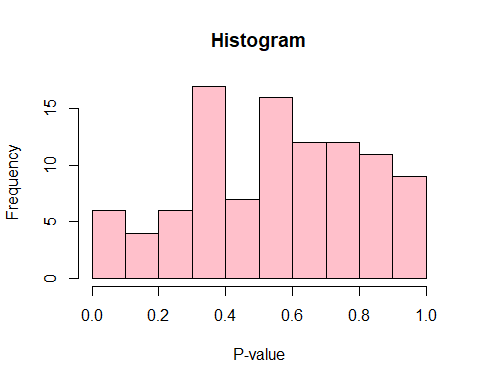
1. Estimate the model Extract the p-value for the **F test** of the whole model.

getpvalue <- function(model, mydata) {  
 model <- lm(model, data = mydata)  
 sum<-summary(model)  
 pvalue<-pf(sum$fstatistic[1], sum$fstatistic[2], sum$fstatistic[3], lower.tail=FALSE)  
 return (pvalue)  
}  
  
getpvalue(Y~X1+X50, mydata)

## value   
## 0.7639463

1. Repeat the simulation (steps a and b), estimation and testing 100 times, and plot the histogram of the p-values. What does it look like? What should it look like?

pvalue\_list <- numeric(100)  
for (i in 1:100){  
 gendata<-as.data.frame(matrix(rnorm(1000\*101),ncol=101))  
 names(gendata)<-c("Y",paste("X",1:100,sep=""))  
 pvalue\_list[i]<-getpvalue(Y~X1+X50, gendata)  
}  
  
# Create a histogram  
 hist(pvalue\_list, main = "Histogram", xlab = "P-value", ylab = "Frequency", col = "pink", border = "black")



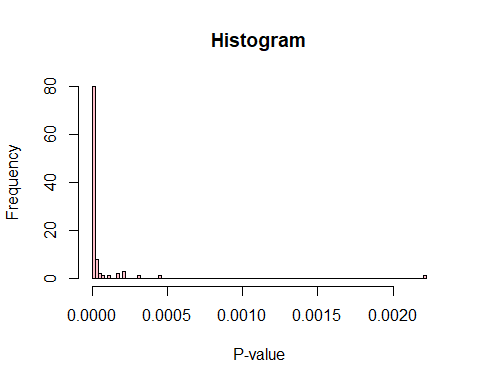
1. From step (a), use the step function to select a linear model by backward stepwise selection.

final\_model <- step(lm(Y~., mydata), direction = "backward", trace = 0)  
model\_backward <- function(model, data){  
 final\_model <- step(model, direction = "backward", trace = 0)  
 sum<-summary(final\_model)  
 pvalue<-pf(sum$fstatistic[1], sum$fstatistic[2], sum$fstatistic[3], lower.tail=FALSE)  
 return (pvalue)  
  
}  
model <- lm(Y~., data = mydata)  
model\_backward(model, mydata)

## value   
## 7.796943e-09

1. Extract the p-value for the **F-test** of the selected model. Repeat 100 times and plot the histogram of p-values. Explain what’s going on.

pvalue\_list\_back <- list()  
for (i in 1:100){  
 gendata<-as.data.frame(matrix(rnorm(1000\*101),ncol=101))  
 names(gendata)<-c("Y",paste("X",1:100,sep=""))  
 model <- lm(Y~., data = gendata)  
 pvalue\_list\_back[i]<-model\_backward(model, gendata)  
}  
combined\_data <- unlist(pvalue\_list\_back)  
# Create a histogram  
hist(combined\_data, main = "Histogram", xlab = "P-value", ylab = "Frequency", breaks = 100, col = "pink", border = "black")



1. Again,
2. use step() to select a model based on one random 1000x101 array. This acts as a training dataset. Compute the in-sample error.

traindata<-as.data.frame(matrix(rnorm(1000\*101),ncol=101))  
names(traindata)<-c("Y",paste("X",1:100,sep=""))  
  
trainmodel <- lm(Y~., data = traindata)  
step\_model <- step(trainmodel, trace = 0)  
  
summary(step\_model)

##   
## Call:  
## lm(formula = Y ~ X9 + X13 + X31 + X33 + X40 + X44 + X47 + X51 +   
## X53 + X54 + X55 + X61 + X72 + X73 + X94, data = traindata)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.7151 -0.6693 0.0057 0.6588 4.6065   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.0003809 0.0317421 0.012 0.990428   
## X9 -0.0622354 0.0326191 -1.908 0.056690 .   
## X13 0.0661169 0.0308867 2.141 0.032549 \*   
## X31 -0.0493883 0.0322023 -1.534 0.125427   
## X33 -0.0473256 0.0321273 -1.473 0.141053   
## X40 -0.0477077 0.0323105 -1.477 0.140120   
## X44 0.0687462 0.0311637 2.206 0.027617 \*   
## X47 -0.0496685 0.0324589 -1.530 0.126290   
## X51 -0.0531202 0.0320632 -1.657 0.097892 .   
## X53 -0.1089305 0.0326937 -3.332 0.000895 \*\*\*  
## X54 0.0552399 0.0320559 1.723 0.085160 .   
## X55 0.0590463 0.0327000 1.806 0.071271 .   
## X61 -0.0521269 0.0334262 -1.559 0.119208   
## X72 0.0491468 0.0308289 1.594 0.111217   
## X73 -0.0459464 0.0325121 -1.413 0.157909   
## X94 -0.0569298 0.0318553 -1.787 0.074223 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9955 on 984 degrees of freedom  
## Multiple R-squared: 0.04752, Adjusted R-squared: 0.033   
## F-statistic: 3.273 on 15 and 984 DF, p-value: 2.338e-05

in\_sample\_error <- mean(residuals(step\_model)^2)  
  
cat("In-sample Mean Squared Error:", in\_sample\_error, "\n")

## In-sample Mean Squared Error: 0.9751785

1. Now re-estimate the selected model on a new 1000 x 101 array, and extract the new p-value. This acts as a testing dataset. Compute the generalization (prediction) error.

testdata<-as.data.frame(matrix(rnorm(1000\*101),ncol=101))  
names(testdata)<-c("Y",paste("X",1:100,sep=""))  
  
sum<-summary(step\_model)  
pvalue<-pf(sum$fstatistic[1], sum$fstatistic[2], sum$fstatistic[3], lower.tail=FALSE)  
  
pvalue

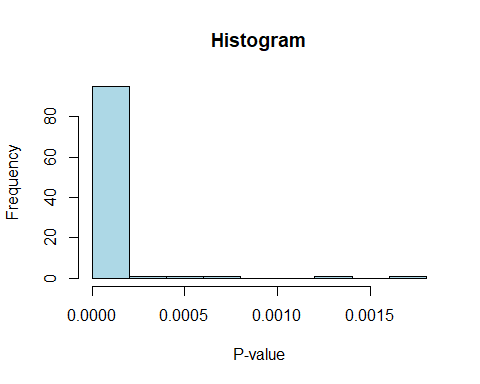
## value   
## 2.337796e-05

predictions\_test <- predict(step\_model, newdata = testdata)  
  
generalization\_error <- mean((testdata$Y - predictions\_test)^2)  
  
cat("Generalization Error:", generalization\_error, "\n")

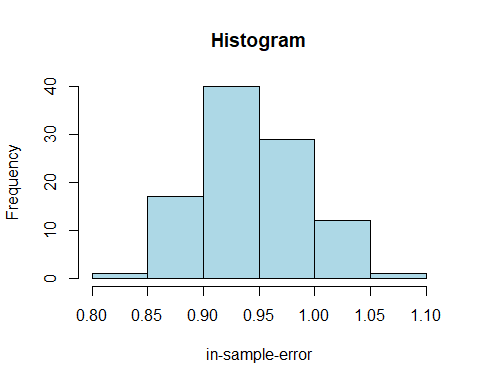
## Generalization Error: 1.123454

1. Repeat 100 times, with new selection and inference sets each time, and plot the histogram of p-values, the histograms of in-sample errors and the histogram of generalization errors.

isr\_list <- numeric(100)  
gen\_list <- numeric(100)  
pvalue\_list\_step <- numeric(100)  
  
for(i in 1:100){  
 train\_gen\_data<-as.data.frame(matrix(rnorm(1000\*101),ncol=101))  
 names(train\_gen\_data)<-c("Y",paste("X",1:100,sep=""))  
   
 test\_gen\_data<-as.data.frame(matrix(rnorm(1000\*101),ncol=101))  
 names(test\_gen\_data)<-c("Y",paste("X",1:100,sep=""))  
   
 genmodel <- lm(Y~., data = train\_gen\_data)  
 final\_gen\_model <- step(genmodel, trace = 0)  
   
 sum<-summary(final\_gen\_model)  
   
 predictions\_test <- predict(final\_gen\_model, newdata = test\_gen\_data)  
   
 pvalue\_list\_step[i]<-pf(sum$fstatistic[1], sum$fstatistic[2], sum$fstatistic[3], lower.tail=FALSE)  
   
 isr\_list[i] <- mean(residuals(final\_gen\_model)^2)  
   
 gen\_list[i] <- mean((test\_gen\_data$Y - predictions\_test)^2)  
   
}  
  
hist(pvalue\_list\_step, main = "Histogram", xlab = "P-value", ylab = "Frequency", col = "lightblue", border = "black")



hist(isr\_list, main = "Histogram", xlab = "in-sample-error", ylab = "Frequency", col = "lightblue", border = "black")



hist(gen\_list, main = "Histogram", xlab = "generalization error", ylab = "Frequency", col = "lightblue", border = "black")

