

STAT 331 Final Project

Henry Xu, 20779975

09/04/2021

#Appendix

```
get.reduced.model = function(model, i){
  # convenient helper to return the new model with ith feature removed
  # i can be vector or number

  # first column of data will be response variable, other columns are features of original
  # model, intercept wouldn't appear here as a feature
  data = model$model
  r = nrow(data)
  c = ncol(data)

  # special case if there is only 1 feature left
  if(c==2){
    return(lm(data[1:r,1]~1))
  }

  # we shouldn't receive a model with only intercept
  if(c==1){
    stop("get.reduced.model() recieved a model with intercept only")
  }

  # explanatory variable
  names = colnames(data)[2:c]
  # response variable
  yname = colnames(data)[1]
  formu = as.formula( paste(yname, "~", paste( names[-i], collapse = "+")))
  # new model
  m = lm(formu , data=data)
  return(m)
}

removezero = function(v){
  v[v==0] = NA
  v
}

# this function is to test transformation of pollutants result on lasso result
# input the transformed data, the function does lasso on pollutants only, ignoring other features
lasso.on.pollutants =function(data_1, plot=FALSE){
  set.seed(seed)
  M = model.matrix(lm(length~., data=data_1))
  cols = colnames(M)
  # get the columns of pollutants features
  po.ind = str_detect(cols, "POP")
  y_train = data_1$length[1:700]
  X_train = M[1:700,po.ind]
  y_test= data_1$length[701:nTotal]
  X_test= M[701:nTotal,(1:ncol(M))[po.ind]]

  M_lasso <- glmnet(x=X_train,y=y_train,alpha = 1)
```

```

## plot paths

## fit with crossval
cvfit_lasso <- cv.glmnet(x=X_train,y=y_train,alpha = 1)

## plot MSPEs by lambda

## estimated betas for minimum lambda

## predictions
pred_lasso <- predict(cvfit_lasso,newx=X_test, s="lambda.min")

## MSPE in test set
MSPE_lasso <- mean((pred_lasso-y_test)^2)
print(paste("mspe",MSPE_lasso) )
if(plot){
  plot(pred_lasso, y_test, main="pollutants choosen by lasso", xlab="predicted value", ylab="actually
}

return( coef(cvfit_lasso, s = "lambda.min"))
}

## Influence
## determing outliers
plot.outliers <- function(M){
  Xmat <- model.matrix(M) ## design matrix
  H <- Xmat%*%solve(t(Xmat)%*%Xmat)%*%t(Xmat) ## Hat matrix
  diag(H)
  lev <- hatvalues(M) ## leverage (h_i)
  hbar <- mean(lev) ## \bar{h}
  c(sum(lev),ncol(model.matrix(M)))## check trace is same as rank of

  ## plot leverage
  plot(lev,ylab="Leverage", main = "Leverage Outliers")
  abline(h=2*hbar,lty=2) ## add line at 2hbar
  ids <- which(lev>2*hbar) ## x values for labelling points >2hbar
  points(lev[ids]~ids,col="red",pch=19) ## add red points >2hbar
  text(x=ids,y=lev[ids], labels=ids, cex= 0.6, pos=2) ## label points >2hbar
}

outliers <- function(M){
  Xmat <- model.matrix(M) ## design matrix
  H <- Xmat%*%solve(t(Xmat)%*%Xmat)%*%t(Xmat) ## Hat matrix
  diag(H)
  lev <- hatvalues(M) ## leverage (h_i)
  hbar <- mean(lev) ## \bar{h}
  c(sum(lev),ncol(model.matrix(M)))## check trace is same as rank of
  which(lev > 2*hbar)
}

plot.jackknife.res <- function(M){
  res <- resid(M) # raw residuals

```

```

Xmat <- model.matrix(M) ## design matrix
H <- Xmat%*%solve(t(Xmat)%*%Xmat)%*%t(Xmat) ## Hat matrix
diag(H)
lev <- hatvalues(M) ## leverage ( $h_i$ )
hbar <- mean(lev) ##  $\bar{h}$ 
ids <- which(lev>2*hbar) ## x values for labelling points  $>2\bar{h}$ 
n <- nobs(M)
p <- length(attr(terms(M),"term.labels"))
stud <- res/(sigma(M)*sqrt(1-lev)) # studentized residuals
jack <- stud*sqrt((n-p-2)/(n-p-1-stud^2))
plot(jack,ylab="Studentized Jackknife Residuals", main = "Jackknife Outliers")
points(jack[ids]~ids,col="red",pch=19) ## add high leverage points
text(ids,jack[ids], labels=ids, cex= 0.6, pos=2) ## label points  $>2\bar{h}$ 
}

jackknife.res <- function(M){
  res <- resid(M) # raw residuals

  Xmat <- model.matrix(M) ## design matrix
  H <- Xmat%*%solve(t(Xmat)%*%Xmat)%*%t(Xmat) ## Hat matrix
  diag(H)
  lev <- hatvalues(M) ## leverage ( $h_i$ )
  hbar <- mean(lev) ##  $\bar{h}$ 
  ids <- which(lev>2*hbar)
  return(ids)
}

## helpful functions for plotting influence
##-----DFFITS-----
# Calculates influential points based on DFFITS.
DFFITS <- function(M,method = 1, cutoff = 0.05){
  data <- M$model
  p <- length(attr(terms(M),"term.labels"))
  n <- nobs(M)
  ## check leverage
  h <- hatvalues(M)
  ##-----DFFITS-----
  dffits_m <- dffits(M)
  if(method == 1){
    cutoff <- 2*sqrt((p+1)/n)
  }
  which(abs(dffits_m)>cutoff)
}

plot.DFFITS <- function(M, method = 1, cutoff = 0.05){
  data <- M$model
  p <- length(attr(terms(M),"term.labels"))
  n <- nobs(M)
  ## check leverage
  h <- hatvalues(M)

  dffits_m <- dffits(M)

```

```

if(method == 1){
  cutoff <- 2*sqrt((p+1)/n)
}

## plot DFFITS
plot(dffits_m,ylab="DFFITS",main = "DFFITS Outliers")
abline(h=cutoff,lty=2, col = "red") ## add thresholds
abline(h=-cutoff,lty=2, col = "red")
## highlight influential points
dff_ind <- which(abs(dffits_m)>cutoff)
points(dffits_m[dff_ind]~dff_ind,col="red",pch=19) ## add red points
text(y=dffits_m[dff_ind],x=dff_ind, labels=dff_ind, pos=2) ## label high influence points
abline(h = cutoff, col = "red", lty = 2)
abline(h = -cutoff, col = "red", lty = 2)
}

##-----Cook's Distance-----
# Calculates influential points based on Cook's Distance
CD <- function(M, cutoff = 0.5){
  p <- length(attr(terms(M),"term.labels"))
  n <- nobs(M)
  D <- cooks.distance(M) # Cook's distance
  ## influential points
  which(pf(D,p+1,n-p-1,lower.tail=TRUE)>cutoff)
}

plot.CD <- function(M,method = 1, cutoff = 0.5){
  # method = 1 is default (may not print any influential points if cutoff is not low enough)
  # method = else <- calculate using simple R method
  if(method == 1){
    p <- length(attr(terms(M),"term.labels"))
    n <- nobs(M)
    D <- cooks.distance(M) # Cook's distance
    ## influential points
    inf_ind <- which(pf(D,p+1,n-p-1,lower.tail=TRUE)>cutoff)

    ## plot cook's Distance
    plot(D,ylab="Cook's Distance")
    points(D[inf_ind]~inf_ind,col="red",pch=19) ## add red points
    text(y=D[inf_ind],x=inf_ind, labels=inf_ind, pos=4) ## label high influence points
  }else{
    plot(M,which = 4)
  }
}

##-----DFBETAS-----
# Calculates influential points based on DFBETAS.
DFBETAS <- function(M, method = 1, cutoff = 0.05){
  DFBETAS <- dfbetas(M)
  dim(DFBETAS)
  n <- nobs(M)

```

```

# method = 1 <- default cutoff 2/sqrt(n)

if(method == 1){
  cutoff <- 2/sqrt(n)
}

vals <- list()
for(i in 2:dim(DFBETAS)[2]){
  vals[[i]] <- which(abs(DFBETAS[,i])>cutoff)
}
vals
}

plot.DFBETAS <- function(M, method = 1, cutoff = 0.05){

  n <- nobs(M)

  # method = 1 <- default cutoff 2/sqrt(n)
  if(method == 1){
    cutoff <- 2/sqrt(n)
  }

  DFBETAS <- dfbetas(M)
  dim(DFBETAS)
  ## beta1
  for(i in 2:dim(DFBETAS)[2]){
    plot(DFBETAS[,i], type="h",xlab="Obs. Number",
         ylab=bquote(beta[.(i)]), main = "DFBETAS")
    show_points <- which(abs(DFBETAS[,i])>cutoff)
    points(x=show_points,y=DFBETAS[show_points,i],pch=19,col="red")
    abline(h = cutoff, col = "red", lty = 2)
    abline(h = -cutoff, col = "red", lty = 2)
    text(x=show_points,y=DFBETAS[show_points,i],labels=show_points,pos=2)
  }
}

# Error Analysis
errorAnalysis <- function(M){
  ## residuals
  newdata <- M$model
  res1 <- resid(M) # raw residuals
  stud1 <- res1/(sigma(M)*sqrt(1-hatvalues(M))) # studentized residuals

  ## plot distribution of studentized residuals
  hist(stud1,breaks="FD",
       probability=TRUE,xlim=c(-4,4),
       xlab="Studentized Residuals",
       main="Distribution of Residuals")
  grid <- seq(-3.5,3.5,by=0.05)
  lines(x=grid,y=dnorm(grid),col="blue") # add N(0,1) pdf

  ## qqplot of studentized residuals

```

```

qqnorm(stud1)
abline(0,1) # add 45 degree line

## plot of residuals vs X
factors <- attr(terms(M),"term.labels")
for(i in 1:length(factors)){
  ind <- which(colnames(newdata)==factors[i])
  plot(res1 ~ newdata[,ind],ylab = "residuals",
       xlab = factors[i], main = paste0("Residuals vs ",factors[i]), ylim = c(-1,2))
}

## plot of studentized residuals vs fitted values
plot(stud1~fitted(M),
     xlab="Fitted Vals",
     ylab="Studentized Residuals",
     main="Residuals vs Fitted")
}

# will only be used for 10-fold CV here
kfolds.cv <- function(dat, expr){
  kfolds=10
  mspe = rep(0, kfolds)
  # labeling each data to one of then groups
  ind = rep(1:kfolds, length=nrow(dat))
  for(ii in 1:kfolds) {
    train<- which(ind!=ii) # training observations
    M.cv <- lm(expr, data=data[train,])
    # cross-validation residuals
    M.res <- dat$length[-train] - # test observations
      predict(M.cv, newdat = dat[-train,]) # prediction with training dat
    # mspe
    mspe[ii] <- mean(M.res^2)
  }
  mean(mspe)
}

# limits:
# only contains history of beta values of initial features
# forward selection won't remove already-added features

forward.change = function(data, expr, show=FALSE){
  # initial smallest model
  model = lm(expr, data=newdata)
  # initial features(removing length), we'll keep track of those
  initial.colname = names(model$coefficients)[-1]
  tempnames = colnames(data)

```

```

cv.hist=c()
aic.hist = c()
coef.hist = list()
DFFITS.hist = c()
outliers.hist = c()
j=0
models = list()
while (TRUE) {
  j=j+1
  print(paste("step", j))
  # existing features in this step's model
  cov.in.m = colnames(model$model)
  # all features
  cov.all = colnames(newdata)
  # the features that are not in this step's model, we will consider all of them
  names.to.try = cov.all[! cov.all %in% cov.in.m]
  nn = length(names.to.try)
  #update tracks of this current model
  cv.hist[j]=kfolds.cv(newdata, expr)
  aic.hist[j] = extractAIC(model)[2]
  coef.hist[[j]] = coef(model)
  DFFITS.hist[j] = length(DFFITS(model))
  outliers.hist[j] = length(outliers(model))
  models[[j]] = model
  cv.score = rep(0, nn)
  # if we have chosen all features
  if(length(names.to.try) == 0){
    print("chose all ")
    break
  }
  # we will try adding the new features one by one
  # record all thier cross vadidation score
  for (i in 1:nn) {
    name = names.to.try[i]
    newexpr = paste(expr, "+", name )
    newmodel = lm(newexpr, data=newdata)
    cv.score[i] = kfolds.cv(newdata, newexpr)
  }
  # the best model this step that has the least MSPE
  ind = which.min(cv.score)

  # if the best model is not better than our last model, we are done
  if(cv.score[ind]>cv.hist[j]){
    print ("done choosing model")
    break
  }else{
    # update our model
    print(paste("added", names.to.try[ind]))
    expr = paste(expr,"+", names.to.try[ind])
    model = lm(expr, data=newdata)
  }
}
plot(cv.hist, main = "cv")

```



```

plot(aic.hist, main = "aic")
plot(DFFITS.hist, main = "# of Influential Points - DFFITS")
plot(outliers.hist, main = "# of Outliers")
i = length(initial.colname)
j = length(coef.hist)
M = matrix(0, nrow = i, ncol = j)
# most importantly, we keep track of how the initial parameters change
# we only need to record it once as we kept track of the coefficient histories.

# the ith parameter
for (ii in 1:i){
  # at jth step
  for (jj in 1:j) {
    M[ii,jj] = coef.hist[[jj]][initial.colname[ii]]
  }
}
if(show==TRUE){
  par(cex=0.7)
  plot(M[1,], main="coefficent of pollutants", type = 'l', col=1, ylim = range(M), xlab="new feature")
  if(i!=1){
    for (a in 2:i){
      lines(1:j, M[a,], col=a)
    }
    legend("topright", legend = initial.colname, col = 1:i, pch=1)
  }
}
return(list(cv=cv.hist, coef=coef.hist, aic=aic.hist,
           outliers=outliers.hist, DFFITS = DFFITS.hist,
           models = models))
}

```

```

# understanding our polulation:
data = read.csv("pollutants.csv")

# change factor features to reasonable names
ind = data$male == 1
data$male[ind] = "M"
data$male[!ind] = "F"

# we will visualize the age groups.
# Note we will discard data$agecat latter, it won't be included in data analysis as we have ageyrs
max(data$ageyrs)

```

```

## [1] 85

# 0-25 will be labeled 1, 26-50 labeled 2, etc.
data$agecat = ceiling(data$ageyrs/25 )
agecat = c("<25", "25-50", "51-75", ">75")

# changing some labels to text descriptions.
for (i in 1:4){
  ind = data$agecat == i
  data$agecat[ind] = agecat[i]
}

```

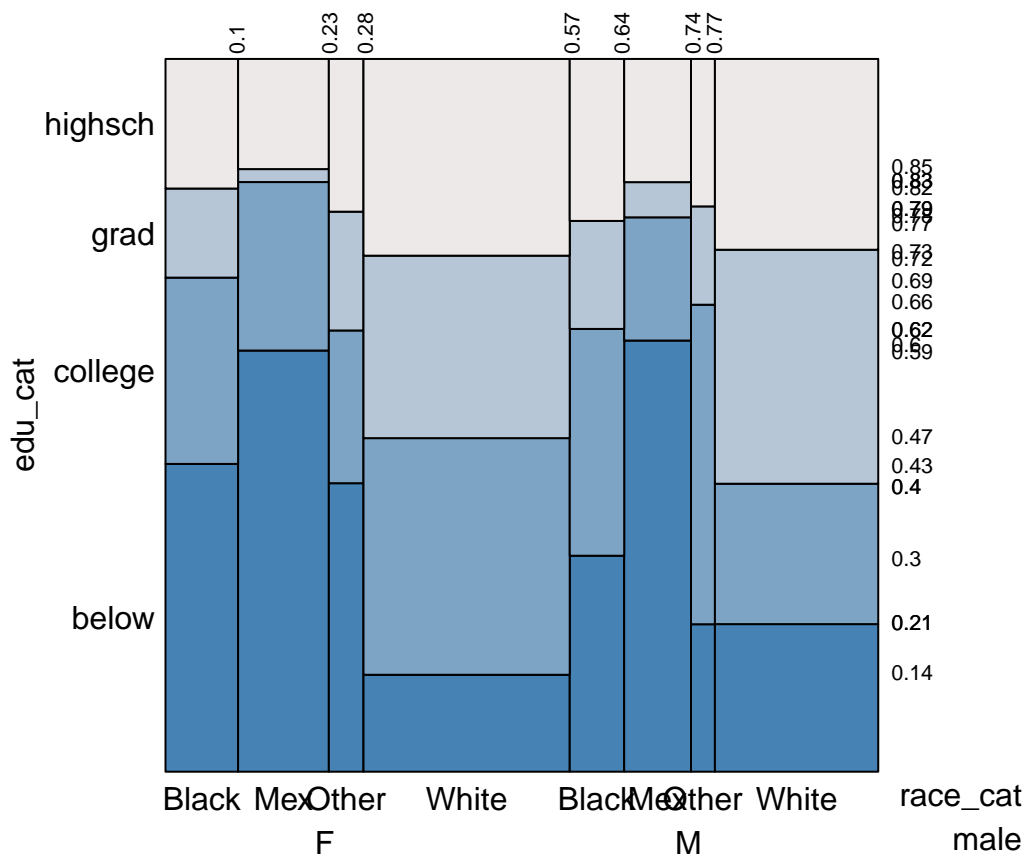
```

edu=c("below", "highsch", "college","grad")
for (i in 1:4){
  ind = data$edu_cat == i
  data$edu_cat[ind] = edu[i]
}

race=c("Other", "Mex", "Black","White")
for (i in 1:4){
  ind = data$race_cat == i
  data$race_cat[ind] = race[i]
}

eikos(edu_cat~ race_cat + male ,data=data)

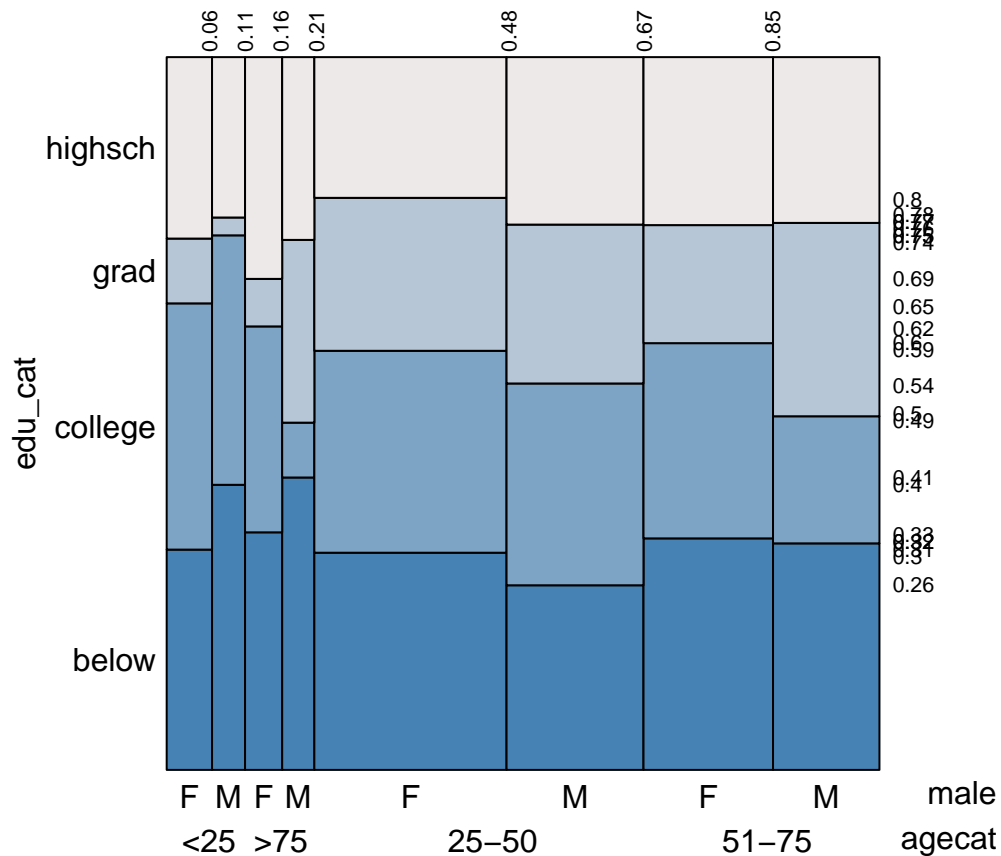
```



```

eikos(edu_cat~ male+agecat ,data=data)

```



```
# get rid of the agecat data we added
if (colnames(data)[ ncol(data)] == "agecat"){
  data = data[,-ncol(data)]
}

data = read.csv("pollutants.csv")

# the index does not really mean anything
data = data[,-1]

nTotal = nrow(data)

#change some feature to factor type
data$race_cat = factor(data$race_cat)
data$edu_cat = factor(data$edu_cat)
data$male = factor(data$male)
data$smokenow= factor(data$smokenow)

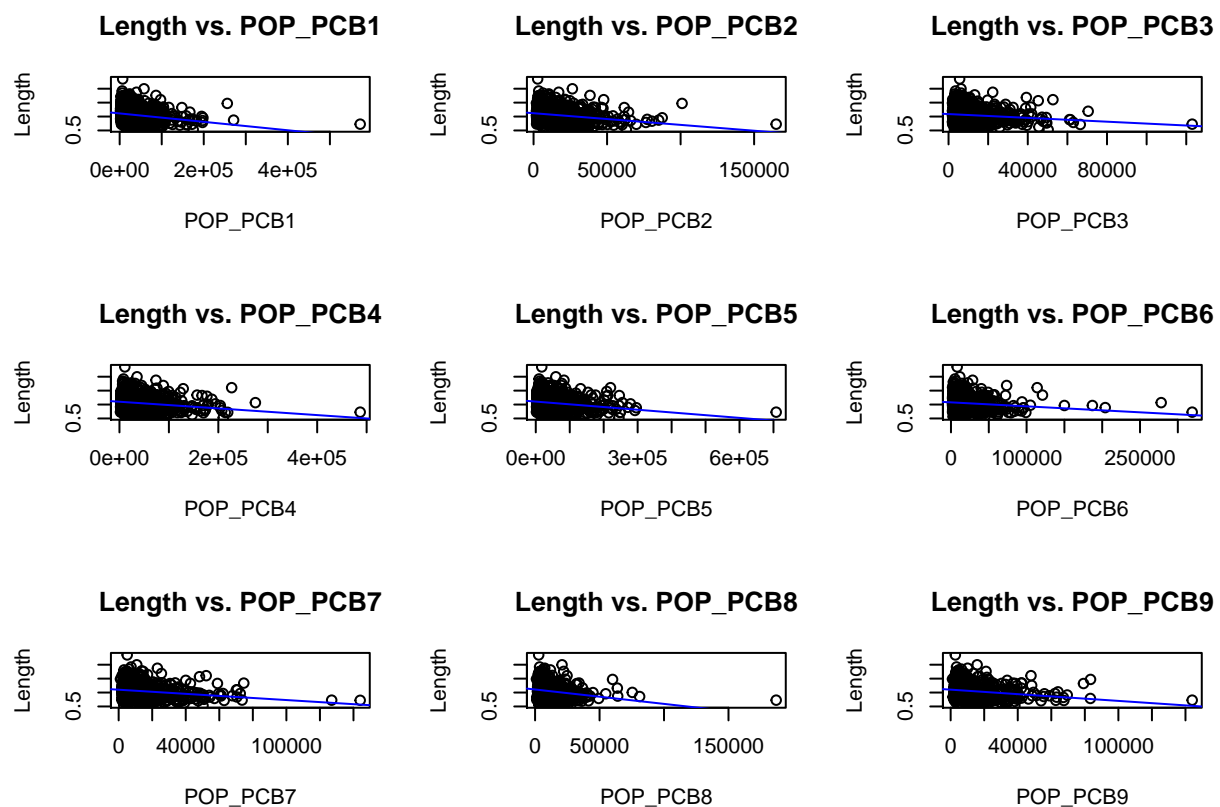
summary.stats <- matrix(NA,nrow = ncol(data),ncol = 7)
cov.names <- colnames(data)
for(i in 1:ncol(data)){
  summary.stats[i,1] <- cov.names[i]
  summary.stats[i,2:(1+length(summary(data[,i])))] <- round(summary(data[,i]),2)
}
```

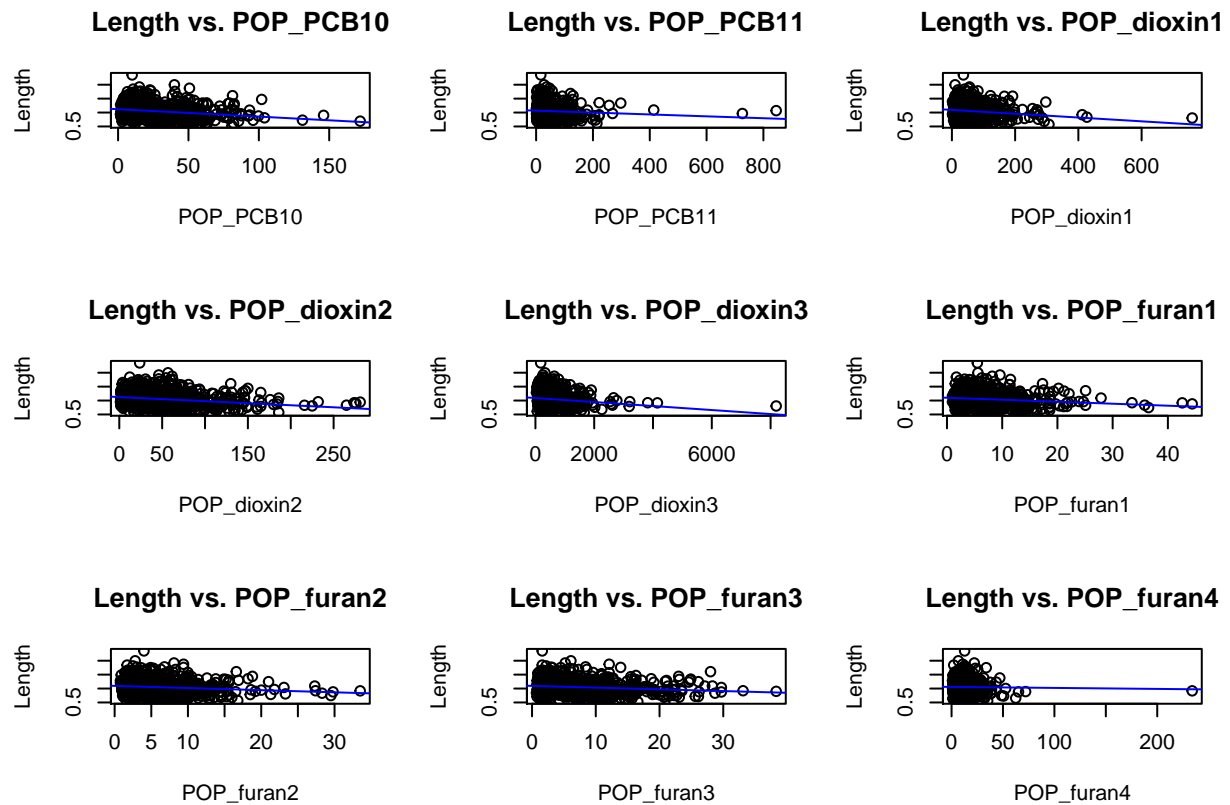
```
knitr::kable(summary.stats,caption = "Summary Statistics",
  col.names = c("Name", "Min.", "1st Qu.",
    "Median", "Mean", "3rd Qu.", "Max."))
```

Table 1: Summary Statistics

Name	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
length	0.53	0.88	1.03	1.05	1.21	2.35
POP_PCB1	2000	9975	27600	38082.18	53325	572000
POP_PCB2	2000	4800	11500	15636.81	21825	165000
POP_PCB3	2000	3700	6200	10157.75	12000	123000
POP_PCB4	2100	11475	25550	38455.79	50650	487000
POP_PCB5	2100	15600	36300	52650.23	68625	708000
POP_PCB6	2000	4400	9400	16820.02	19500	319000
POP_PCB7	1100	4000	7450	12681.94	15625	144000
POP_PCB8	1100	3800	6950	10529.75	14425	187000
POP_PCB9	1100	3900	8050	12220.25	16025	144000
POP_PCB10	1.7	9.1	18.35	24.49	34.9	172
POP_PCB11	1.3	14.8	24.5	38.15	42.95	845
POP_dioxin1	1.9	23.9	41.35	57.65	71.62	760
POP_dioxin2	1.4	21.28	37.8	47.81	62.42	281
POP_dioxin3	36.8	196.98	342.5	494.42	603	8190
POP_furan1	1	3.2	5.2	6.37	7.7	44.4
POP_furan2	0.8	2.6	4.2	5.39	6.82	33.5
POP_furan3	0.7	2.2	5.05	6.67	9.3	38.3
POP_furan4	0.9	6.4	9.65	11.54	14	234
whitecell_count	2.3	5.6	6.9	7.19	8.3	20.1
lymphocyte_pct	5.8	24	28.95	29.92	35.42	73.4
monocyte_pct	1.6	6.6	7.7	7.94	9.1	23.8
eosinophils_pct	21.6	52.35	59.3	58.62	65.23	88.1
basophils_pct	0	1.5	2.3	2.9	3.7	28.2
neutrophils_pct	0	0.4	0.6	0.67	0.8	5.5
BMI	16.16	23.88	27.38	28.09	31.17	62.99
edu_cat	270	199	228	167	NA	NA
race_cat	71	191	154	448	NA	NA
male	490	374	NA	NA	NA	NA
ageyrs	20	34	46	48.36	63	85
yrssmoke	0	0	0	10.6	20	69
smokenow	664	200	NA	NA	NA	NA
ln_lbxcot	-4.51	-4.07	-2.73	-0.98	2.8	6.58

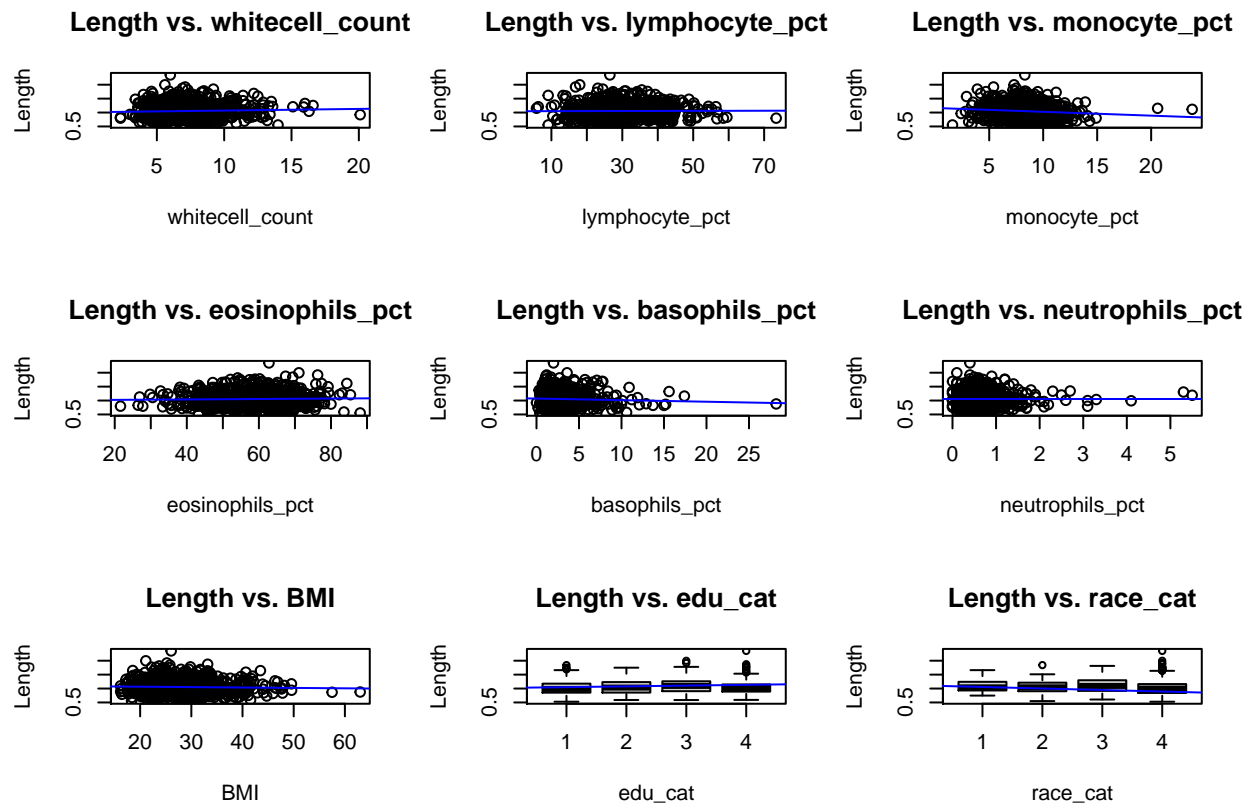
```
par(mfrow=c(3,3))
for(i in 1:length(cov.names[-1])){
  temp.model <- lm(paste0("length ~ ",cov.names[i+1]),data = data)
  plot(data[,cov.names[i+1]],data$length, main = paste0("Length vs. ",cov.names[i+1]),
    ylab = "Length", xlab = cov.names[i+1])
  abline(temp.model,col = "blue")
}
```



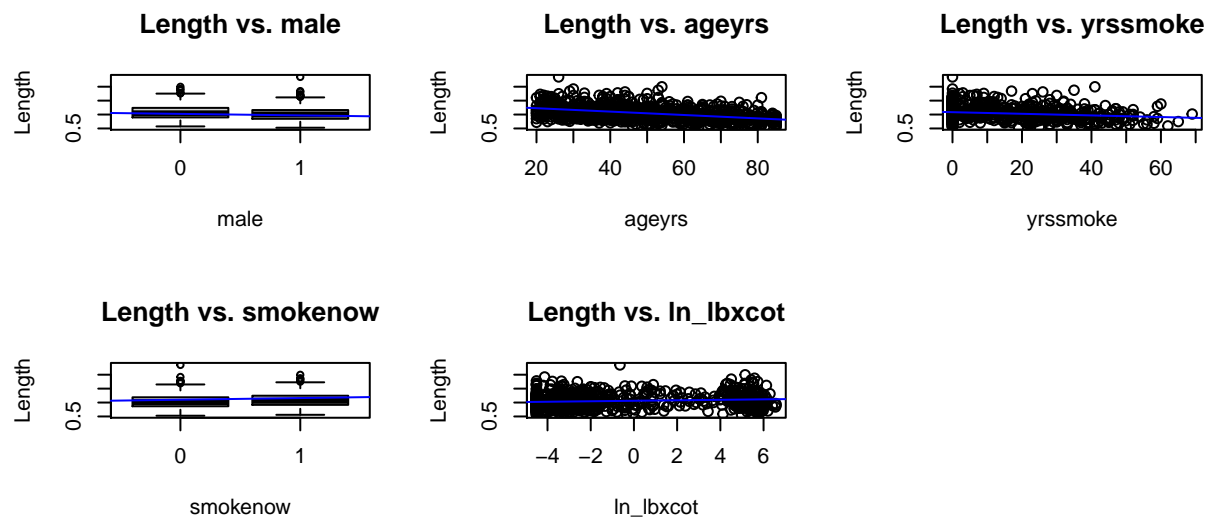


```
## Warning in abline(temp.model, col = "blue"): only using the first two of 4
## regression coefficients
```

```
## Warning in abline(temp.model, col = "blue"): only using the first two of 4
## regression coefficients
```



```
data.train = data[1:700,]
data.test = data[701:nTotal,]
```



```
# correlation between features
# high correlation -> coefficients have large variance
```

```
model = lm(length~. , data=data)
#original vif
```

```
vif(model)
```

##		GVIF	Df	GVIF^(1/(2*Df))
##	POP_PCB1	33.044120	1	5.748401
##	POP_PCB2	34.281125	1	5.855009
##	POP_PCB3	9.351143	1	3.057964
##	POP_PCB4	31.742239	1	5.634025
##	POP_PCB5	59.896895	1	7.739308
##	POP_PCB6	11.386658	1	3.374412
##	POP_PCB7	4.870075	1	2.206825
##	POP_PCB8	12.982575	1	3.603134
##	POP_PCB9	12.441595	1	3.527264
##	POP_PCB10	6.020678	1	2.453707
##	POP_PCB11	4.725769	1	2.173883
##	POP_dioxin1	5.276251	1	2.297009
##	POP_dioxin2	5.413132	1	2.326614
##	POP_dioxin3	4.398509	1	2.097262
##	POP_furan1	6.154213	1	2.480769
##	POP_furan2	6.195336	1	2.489043
##	POP_furan3	4.464346	1	2.112900
##	POP_furan4	1.821809	1	1.349744


```
## whitecell_count      1.548380  1      1.244339
## lymphocyte_pct    12250.336528  1     110.681238
## monocyte_pct       726.843372  1      26.960033
## eosinophils_pct  15071.561945  1     122.766290
## basophils_pct      867.412798  1      29.451873
## neutrophils_pct    37.984114  1       6.163125
## BMI                1.263662  1      1.124127
## edu_cat            1.543109  3      1.074978
## race_cat           2.052848  3      1.127352
## male               1.350324  1      1.162034
## ageyrs             3.238631  1      1.799620
## yrssmoke           2.204139  1      1.484634
## smokenow           4.006708  1      2.001676
## ln_lbxcot          3.963407  1      1.990831
```

```
t1=colnames( model$model)
```

```
while (TRUE) {
  score = vif(model)
  if (max(score) <10){
    break
  }
  ind = which.max(score)
  # this is safe with factor data type
  model = get.reduced.model(model, ind)
}
# reduced model vif
vif(model)
```

```
##              GVIF Df  GVIF^(1/(2*Df))
## POP_PCB3      5.310340  1      2.304417
## POP_PCB6      9.083828  1      3.013939
## POP_PCB7      4.686485  1      2.164829
## POP_PCB8      5.894052  1      2.427767
## POP_PCB9      7.640480  1      2.764142
## POP_PCB10     5.149483  1      2.269247
## POP_PCB11     4.210120  1      2.051858
## POP_dioxin1   5.184345  1      2.276916
## POP_dioxin2   5.275271  1      2.296796
## POP_dioxin3   4.311410  1      2.076394
## POP_furan1    6.000097  1      2.449509
## POP_furan2    6.154621  1      2.480851
## POP_furan3    4.412739  1      2.100652
## POP_furan4    1.812793  1      1.346400
## whitecell_count 1.533642  1      1.238403
## lymphocyte_pct 1.370966  1      1.170882
## monocyte_pct   1.255543  1      1.120510
## basophils_pct  1.097132  1      1.047441
## neutrophils_pct 1.083675  1      1.040997
## BMI            1.257562  1      1.121411
## edu_cat        1.498239  3      1.069704
## race_cat       2.012804  3      1.123657
## male           1.345703  1      1.160045
## ageyrs         3.224432  1      1.795670
```

```

## yrssmoke      2.147610  1      1.465473
## smokenow      3.967106  1      1.991759
## ln_lbxcot      3.946223  1      1.986510

t2=colnames( model$model)

# view what features got removed
setdiff(t1,t2)

## [1] "POP_PCB1"          "POP_PCB2"          "POP_PCB4"          "POP_PCB5"
## [5] "eosinophils_pct"

# does one feature alone explain the model?

# we fit length to each covariate in a linear/log/square model

Xfull = lm(length~., data=data)$model

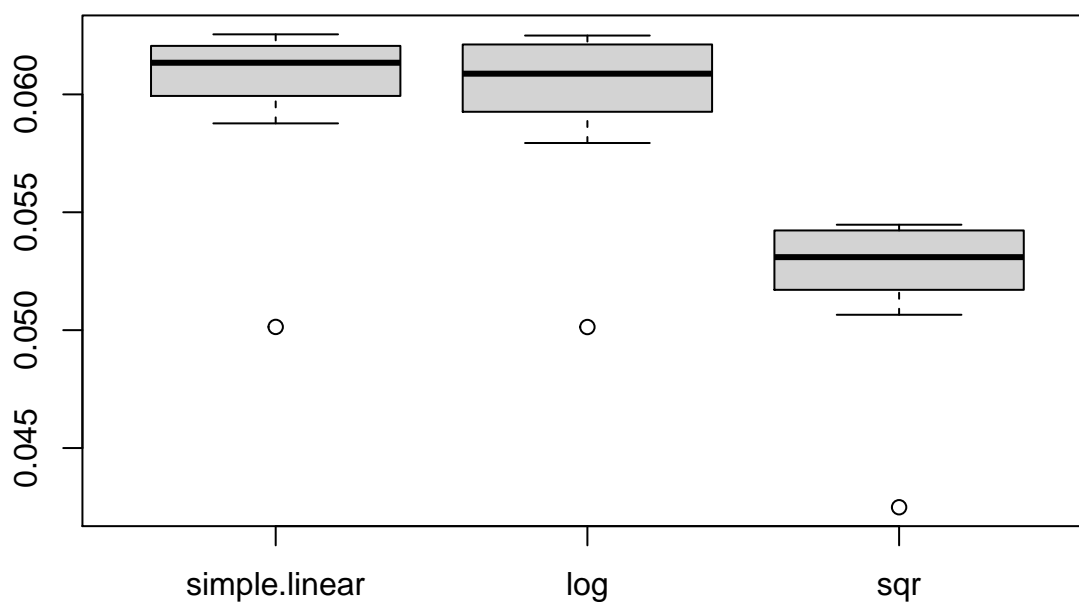
res = matrix(0, nrow = (ncol(Xfull)), ncol = 3)

for(c in 2:ncol(Xfull)){
  model = lm(data$length~Xfull[,c])
  #res[,1] is simple linear models
  #res[,2] is log linear models, we on
  #res[,3] is square models
  res[c,1] = mean(model$residuals^2)
  # we won't fit log or square model for categorical variable
  if(! is.factor(Xfull[,c])){
    modelpower2 = lm(data$length~poly( Xfull[,c], 2))
    res[c,2] = mean(modelpower2$residuals^2)
    if (! any(Xfull[,c] < 0 )){
      # we won't try to log the feature that has negative values
      modellog = lm(log(data$length)~ Xfull[,c])
      res[c,3] = mean(modellog$residuals^2)
    }
  }
}

# how do these models perform in terms of mse
box = list(simple.linear=removezero(res[,1]), log=removezero(res[,2]), sqr=removezero(res[,3]))
boxplot(box, main="single variable, MSE")

```

single variable, MSE



```
# to look at what is best single variable model  
which.min(removezero(res[,1]))
```

```
## [1] 30
```

```
which.min(removezero(res[,2]))
```

```
## [1] 30
```

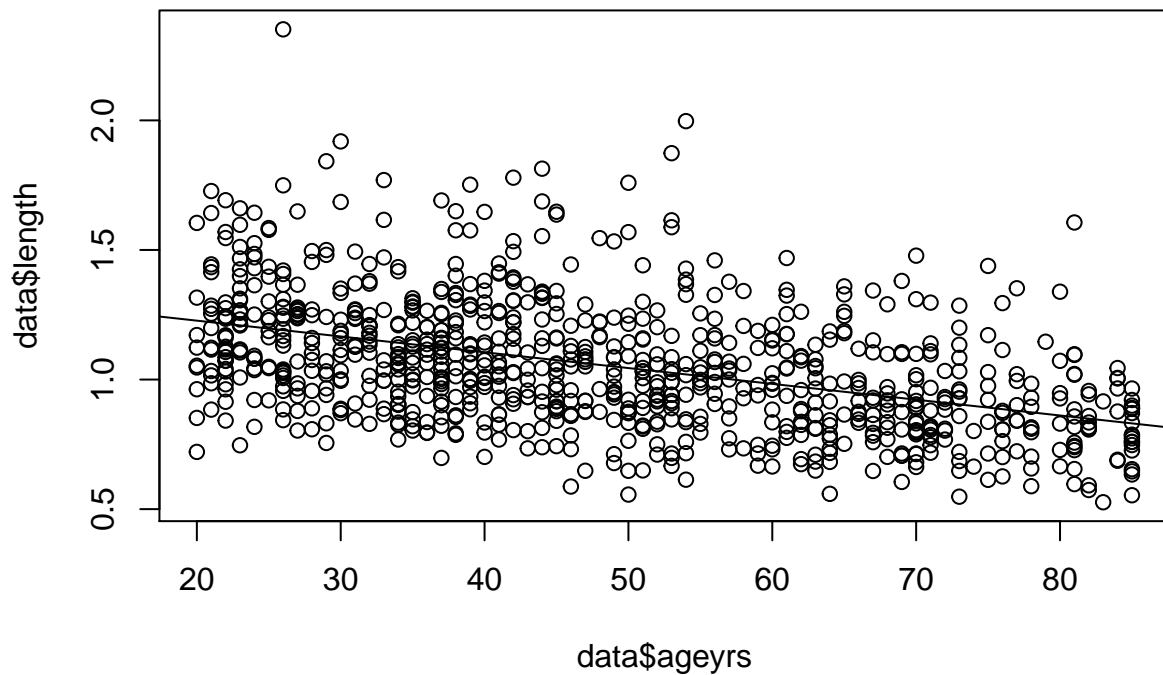
```
which.min(removezero(res[,3]))
```

```
## [1] 30
```

```
# which is the best single feature  
colnames(Xfull)[30]
```

```
## [1] "ageyrs"
```

```
# what does the best model look like  
simplelinear = lm(length~ageyrs, data=data)  
plot(data$ageyrs, data$length)  
abline(simplelinear$coefficients)
```



#seems there is a linear relationship but looks insufficient.

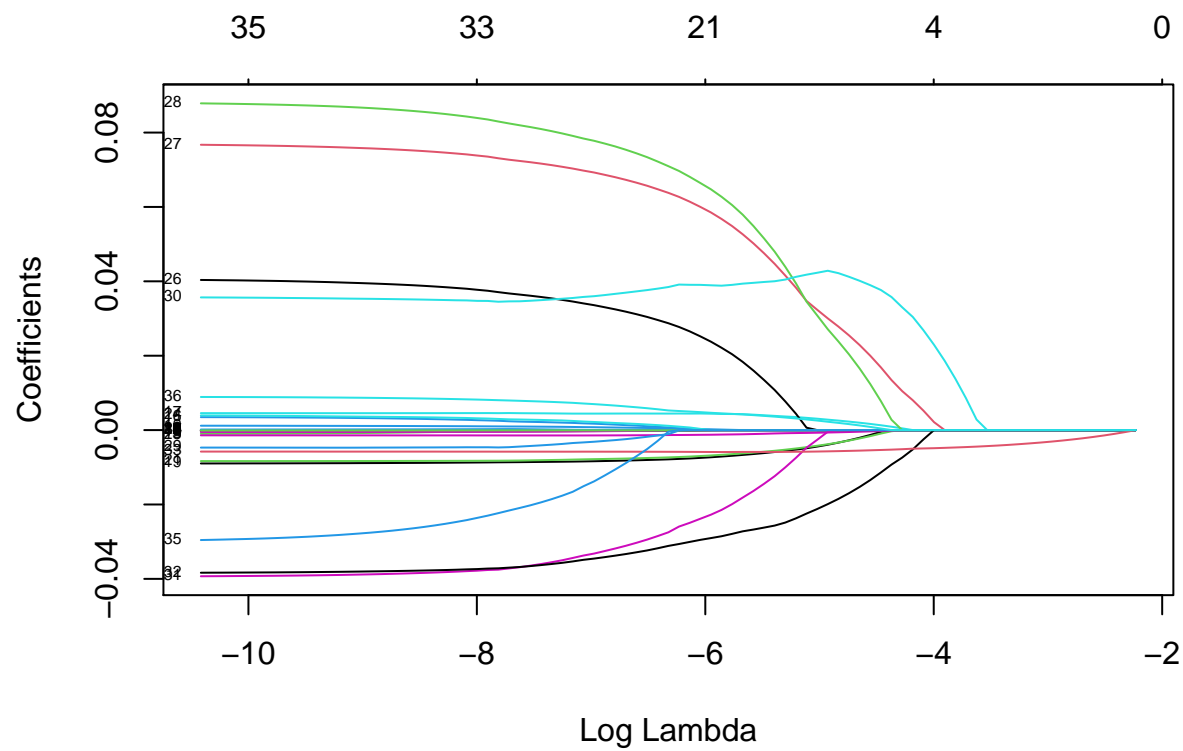
#Also seems sqr or log does not do exponentially better here

#how is model choosen by automated soluation

```
### LASSO
## fit models
M = model.matrix(lm(length~., data=data))
y_train = data$length[1:700]
X_train = M[1:700,-1]
y_test= data$length[701:nTotal]
X_test= M[701:nTotal,-1]

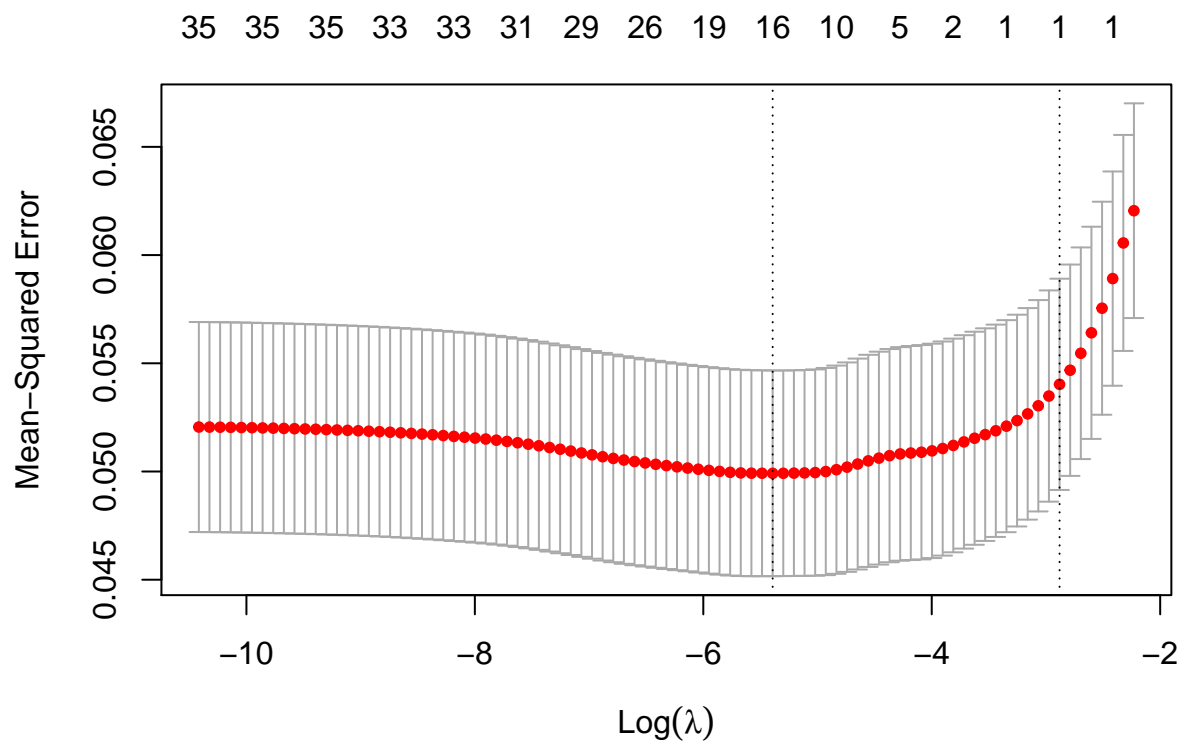
M_lasso <- glmnet(x=X_train,y=y_train,alpha = 1)
####

####
## plot paths
plot(M_lasso,xvar = "lambda",label=TRUE)
```



```
## fit with crossval
cvfit_lasso <- cv.glmnet(x=X_train,y=y_train,alpha = 1)

## plot MSPEs by lambda
plot(cvfit_lasso)
```



```
## estimated betas for minimum lambda
coef(cvfit_lasso, s = "lambda.min")
```

```
## 37 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)  1.420584e+00
## POP_PCB1     -2.566868e-08
## POP_PCB2     .
## POP_PCB3     1.713297e-07
## POP_PCB4     .
## POP_PCB5     .
## POP_PCB6     .
## POP_PCB7     .
## POP_PCB8     .
## POP_PCB9     .
## POP_PCB10    .
## POP_PCB11    1.668336e-05
## POP_dioxin1  .
## POP_dioxin2  .
## POP_dioxin3  -3.504572e-06
## POP_furan1  .
## POP_furan2  .
## POP_furan3  3.820752e-03
## POP_furan4  .
## whitecell_count -5.893220e-03
## lymphocyte_pct .
```

```

## monocyte_pct      -5.559884e-03
## eosinophils_pct   .
## basophils_pct     .
## neutrophils_pct   .
## BMI               -9.043493e-04
## edu_cat2          1.077925e-02
## edu_cat3          4.469577e-02
## edu_cat4          4.797734e-02
## race_cat2         .
## race_cat3         4.002613e-02
## race_cat4        -1.238929e-02
## male1             -2.567631e-02
## ageyrs            -5.874149e-03
## yrssmoke          .
## smokenow1         .
## ln_lbxcot         3.612243e-03

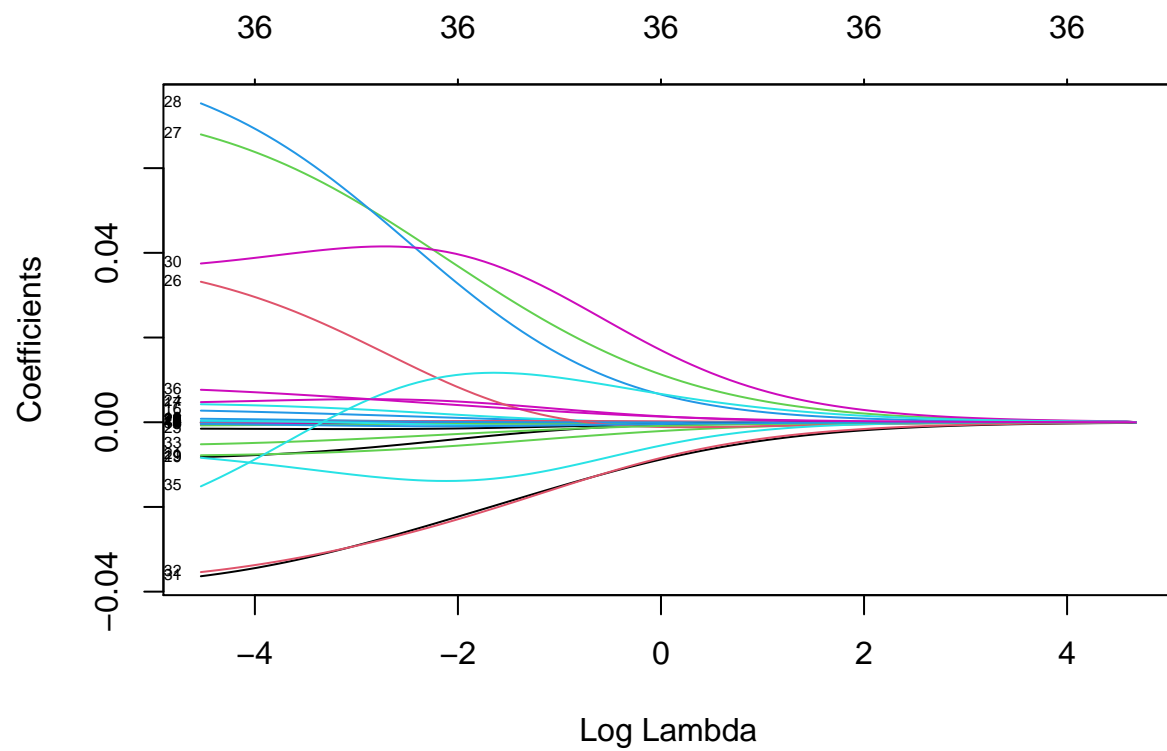
## predictions
pred_lasso <- predict(cvfit_lasso,newx=X_test, s="lambda.min")

## MSPE in test set
MSPE_lasso <- mean((pred_lasso-y_test)^2)

## RIDGE
## fit models
M_ridge <- glmnet(x=X_train,y=y_train,alpha = 0)

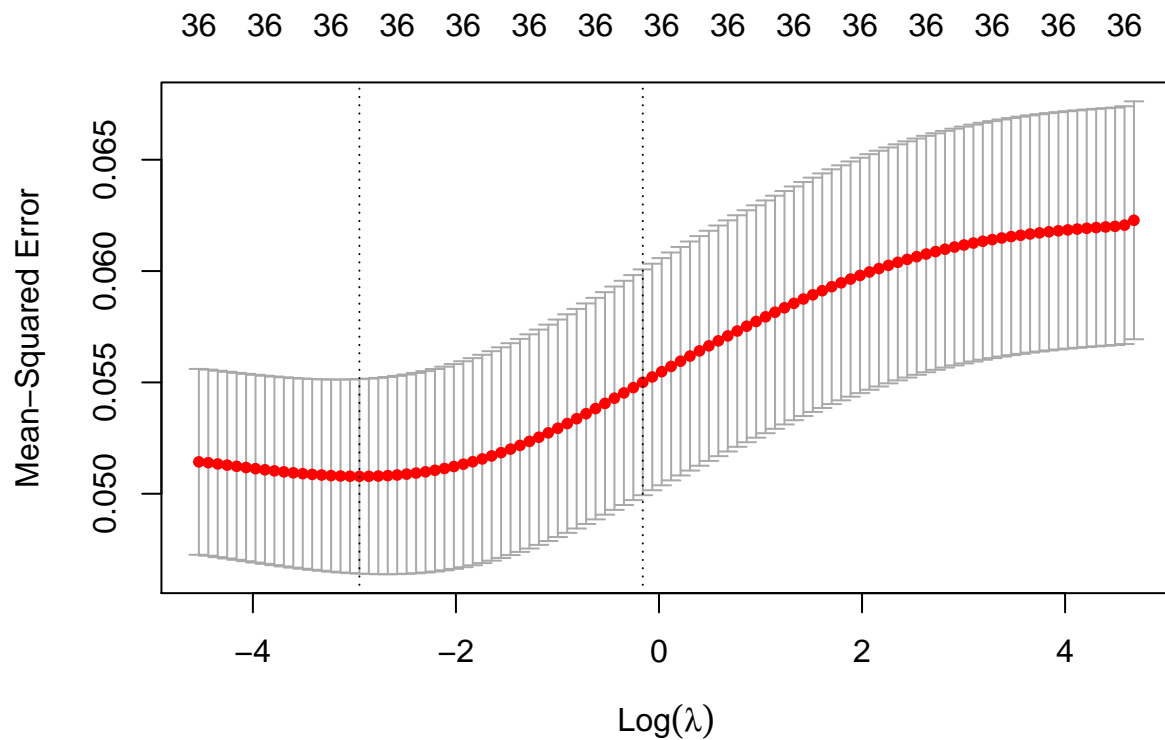
## plot paths
plot(M_ridge,xvar = "lambda",label=TRUE)

```



```
## fit with crossval
cvfit_ridge <- cv.glmnet(x=X_train,y=y_train,alpha = 0)

## plot MSPEs by lambda
plot(cvfit_ridge)
```

```
## estimated betas for minimum lambda
coef(cvfit_ridge, s = "lambda.min") ## alternatively could use "lambda.1se"
```

```
## 37 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)  1.385275e+00
## POP_PCB1     -2.772870e-07
## POP_PCB2     -2.309578e-07
## POP_PCB3      1.030760e-06
## POP_PCB4     -2.310000e-08
## POP_PCB5     -2.393277e-08
## POP_PCB6      8.648413e-08
## POP_PCB7     -5.511863e-07
## POP_PCB8     -5.058585e-07
## POP_PCB9      7.214288e-08
## POP_PCB10     3.708336e-04
## POP_PCB11     6.946198e-05
## POP_dioxin1  -1.056999e-04
## POP_dioxin2  -3.133691e-04
## POP_dioxin3  -1.054185e-05
## POP_furan1   -4.949674e-04
## POP_furan2    1.757401e-03
## POP_furan3    3.023534e-03
## POP_furan4    7.990304e-07
## whitecell_count -6.136543e-03
## lymphocyte_pct  1.995319e-04
```

```

## monocyte_pct      -6.823121e-03
## eosinophils_pct   1.602941e-04
## basophils_pct     -1.549003e-04
## neutrophils_pct   5.448093e-03
## BMI               -1.615506e-03
## edu_cat2          1.908905e-02
## edu_cat3          5.147026e-02
## edu_cat4          5.189274e-02
## race_cat2         -1.259614e-02
## race_cat3         4.138239e-02
## race_cat4         -2.879970e-02
## male1             -2.890734e-02
## ageyrs            -3.909685e-03
## yrssmoke          -8.452361e-04
## smokenow1         4.580048e-03
## ln_lbxcot         5.581195e-03

## predictions
pred_ridge <- predict(cvfit_ridge,newx=X_test, s="lambda.min")

## MSPE in test set
MSPE_ridge <- mean((pred_ridge-y_test)^2)

## stepwise
M0 = lm(length~1, data=data.train)
Mfull = lm(length~., data=data.train)
Mstep <- step(object = M0,
              scope = list(lower = M0, upper = Mfull),
              direction = "both", trace = 0, k = 2)

MSPE_step = mean(( predict(Mstep, newdata=data.test) - y_test)^2)

p = predict(Mstep, newdata=data.test)

cvfit_lasso$del

## NULL
# surprisingly, this is greater than MSE of ageyrs~length.
MSPE_lasso

## [1] 0.05125001
MSPE_ridge

## [1] 0.05277559
MSPE_step

## [1] 0.05387623
# models by automated selection makes little sense for interpretation

#pollutants and bioinfo makes little sense and there are too many covariate

```

```

#lets see if there is a smaller good model

#say we try to fit with only 2 features

# lasso choose the same single variable
min(which((M_lasso$lambda)<=exp( -2.5)))

## [1] 4

coefs = M_lasso$beta[,4]
which(coefs!=0)

## ageyrs
##      33

# what 2 features did lasso choose
i = min(which((M_lasso$lambda)<=exp( -3.96)))
coefs = M_lasso$beta[,i]
choosen=which(coefs!=0)
coefs[choosen]

##      edu_cat3      race_cat3      ageyrs
## 0.002132031 0.022925033 -0.004863833

# explore all 2-features model with best subset
models= regsubsets(length~., data=data, nvmax=2)
summary(models)

## Subset selection object
## Call: regsubsets.formula(length ~ ., data = data, nvmax = 2)
## 36 Variables (and intercept)
##              Forced in Forced out
## POP_PCB1          FALSE      FALSE
## POP_PCB2          FALSE      FALSE
## POP_PCB3          FALSE      FALSE
## POP_PCB4          FALSE      FALSE
## POP_PCB5          FALSE      FALSE
## POP_PCB6          FALSE      FALSE
## POP_PCB7          FALSE      FALSE
## POP_PCB8          FALSE      FALSE
## POP_PCB9          FALSE      FALSE
## POP_PCB10         FALSE      FALSE
## POP_PCB11         FALSE      FALSE
## POP_dioxin1        FALSE      FALSE
## POP_dioxin2        FALSE      FALSE
## POP_dioxin3        FALSE      FALSE
## POP_furan1        FALSE      FALSE
## POP_furan2        FALSE      FALSE
## POP_furan3        FALSE      FALSE
## POP_furan4        FALSE      FALSE
## whitecell_count    FALSE      FALSE
## lymphocyte_pct     FALSE      FALSE
## monocyte_pct       FALSE      FALSE
## eosinophils_pct    FALSE      FALSE
## basophils_pct      FALSE      FALSE

```

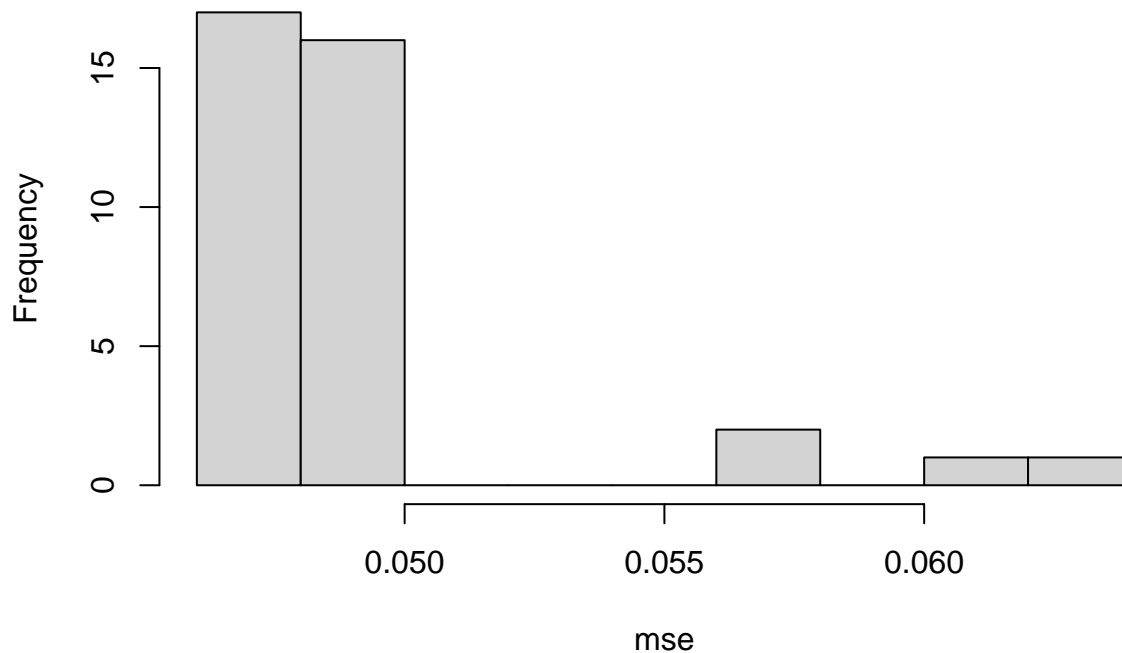
```

## neutrophils_pct      FALSE      FALSE
## BMI                  FALSE      FALSE
## edu_cat2             FALSE      FALSE
## edu_cat3             FALSE      FALSE
## edu_cat4             FALSE      FALSE
## race_cat2            FALSE      FALSE
## race_cat3            FALSE      FALSE
## race_cat4            FALSE      FALSE
## male1                FALSE      FALSE
## ageyrs               FALSE      FALSE
## yrssmoke             FALSE      FALSE
## smokenow1            FALSE      FALSE
## ln_lbxcot            FALSE      FALSE
## 1 subsets of each size up to 2
## Selection Algorithm: exhaustive
##      POP_PCB1 POP_PCB2 POP_PCB3 POP_PCB4 POP_PCB5 POP_PCB6 POP_PCB7
## 1 ( 1 ) " "      " "      " "      " "      " "      " "      " "
## 2 ( 1 ) " "      " "      " "      " "      " "      " "      " "
##      POP_PCB8 POP_PCB9 POP_PCB10 POP_PCB11 POP_dioxin1 POP_dioxin2
## 1 ( 1 ) " "      " "      " "      " "      " "      " "
## 2 ( 1 ) " "      " "      " "      " "      " "      " "
##      POP_dioxin3 POP_furan1 POP_furan2 POP_furan3 POP_furan4
## 1 ( 1 ) " "      " "      " "      " "      " "
## 2 ( 1 ) " "      " "      " "      "*"      " "
##      whitecell_count lymphocyte_pct monocyte_pct eosinophils_pct
## 1 ( 1 ) " "      " "      " "      " "
## 2 ( 1 ) " "      " "      " "      " "
##      basophils_pct neutrophils_pct BMI edu_cat2 edu_cat3 edu_cat4 race_cat2
## 1 ( 1 ) " "      " "      " " " "      " "      " "      " "
## 2 ( 1 ) " "      " "      " " " "      " "      " "      " "
##      race_cat3 race_cat4 male1 ageyrs yrssmoke smokenow1 ln_lbxcot
## 1 ( 1 ) " "      " "      " "      "*"      " "      " "      " "
## 2 ( 1 ) " "      " "      " "      "*"      " "      " "      " "

# rss of all 2 feature model, we see no magical model
mse = models$RSS/nrow(data)
hist(mse, main = "Histogram for MSE")

```

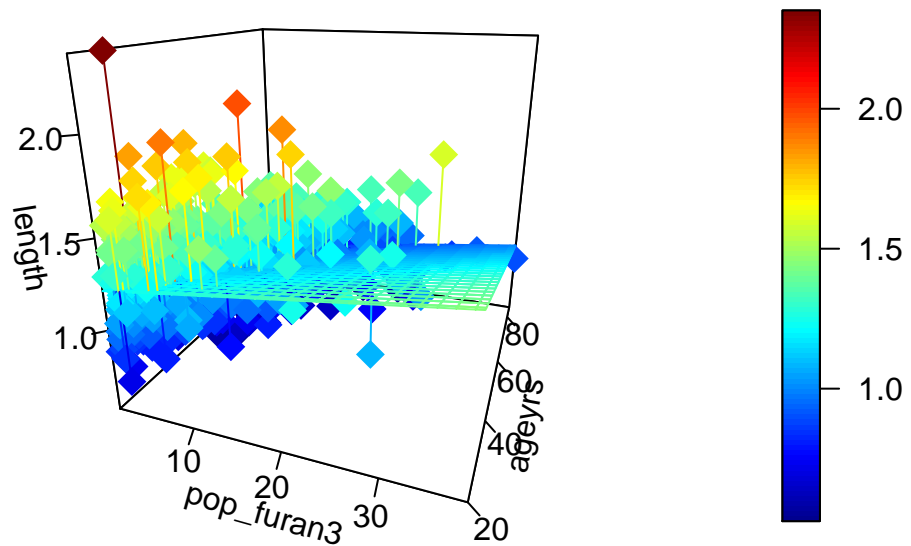
Histogram for MSE



```
# what does the best 2 feature model look like?
z=data$length
y=data$ageyrs
x=data$POP_furan3

fit <- lm(z ~ x + y)
# predict values on regular xy grid
grid.lines = 26
x.pred <- seq(min(x), max(x), length.out = grid.lines)
y.pred <- seq(min(y), max(y), length.out = grid.lines)
xy <- expand.grid( x = x.pred, y = y.pred)
z.pred <- matrix(predict(fit, newdata = xy),
                 nrow = grid.lines, ncol = grid.lines)
# fitted points for droplines to surface

fitpoints = predict(fit)
# scatter plot with regression plane
scatter3D(x, y, z, pch = 18, cex = 2,
          theta = 20, phi = 20, ticktype = "detailed",
          surf = list(x = x.pred, y = y.pred, z = z.pred,
                     facets = NA, fit = fitpoints), xlab="pop_furan3", ylab="ageyrs",zlab="length")
```



```
# perhaps length is related to organic pollutants

# pollutants values are very large, we log transform it. and erroranalysis looks better

cols = colnames(data)
po.ind = str_detect(cols, "POP")
seed <- "20779975"

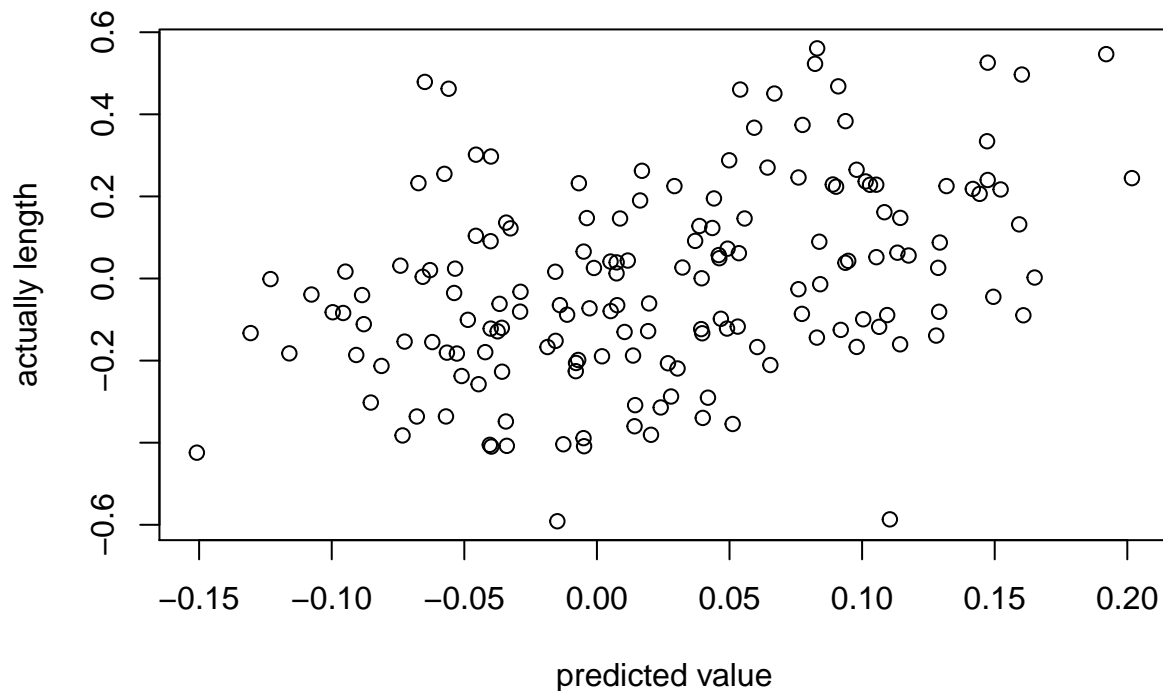
# we will analysis these
set.seed(seed)

# log transform
newdata=data
newdata$length <- log(data$length)
newdata[,po.ind] = log(newdata[,po.ind])

chosen.po.ind= which(lasso.on.pollutants(newdata,TRUE)!=0)

## [1] "mspe 0.0493594027827774"
```

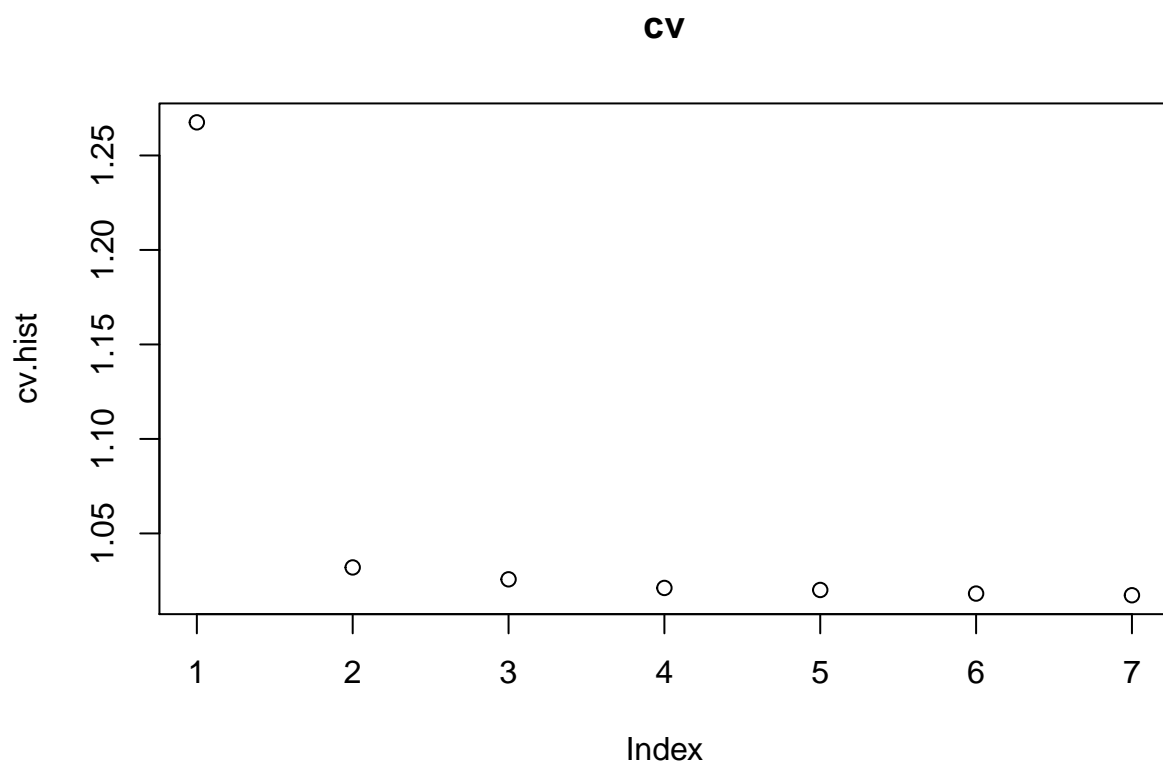
pollutants chosen by lasso

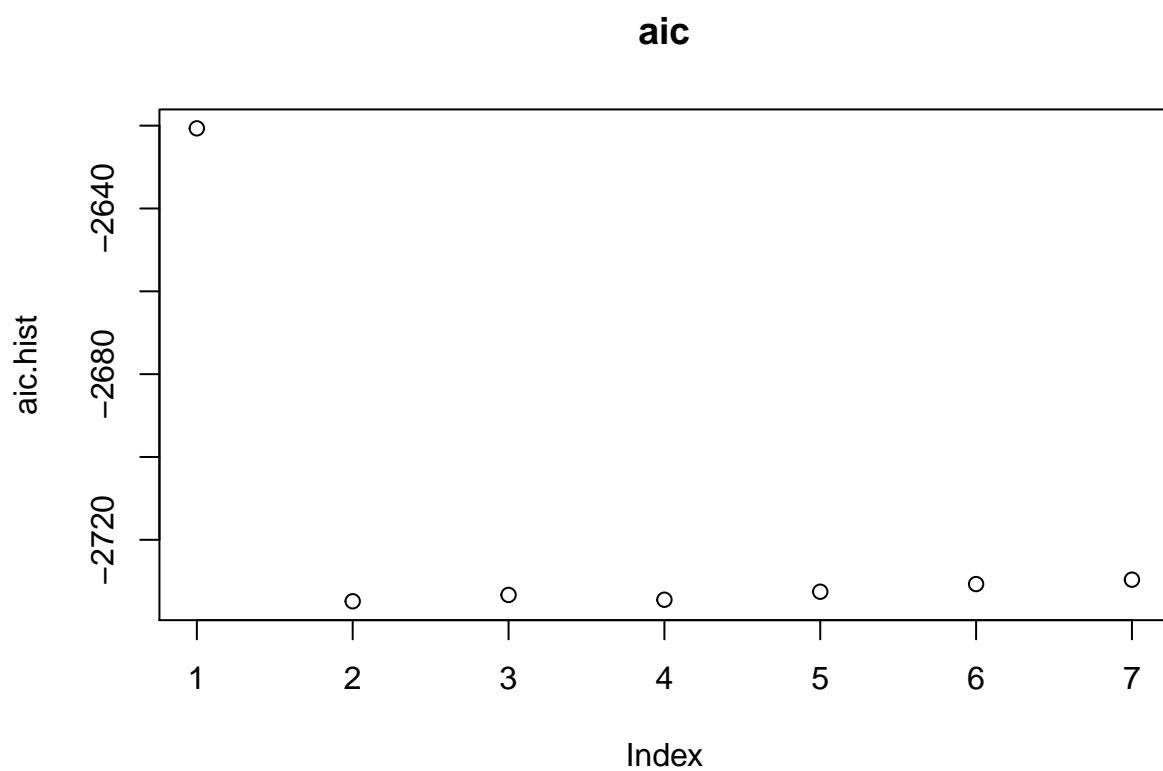


```
chosen.po.ind= chosen.po.ind[2:length(chosen.po.ind)]
chosen.pos = colnames(newdata)[chosen.po.ind]
expr = paste("length~", paste(chosen.pos, collapse = "+"))
lasso.pollu.model = (lm(expr,data=newdata))
```

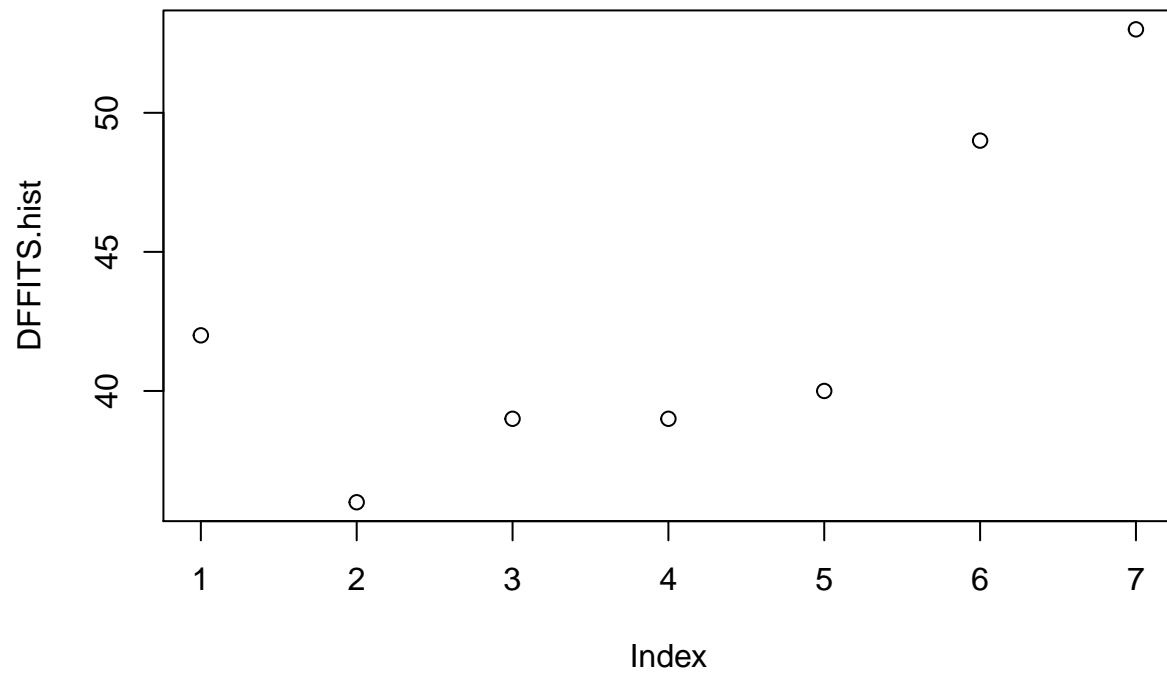
```
t=forward.change(newdata, expr, TRUE)
```

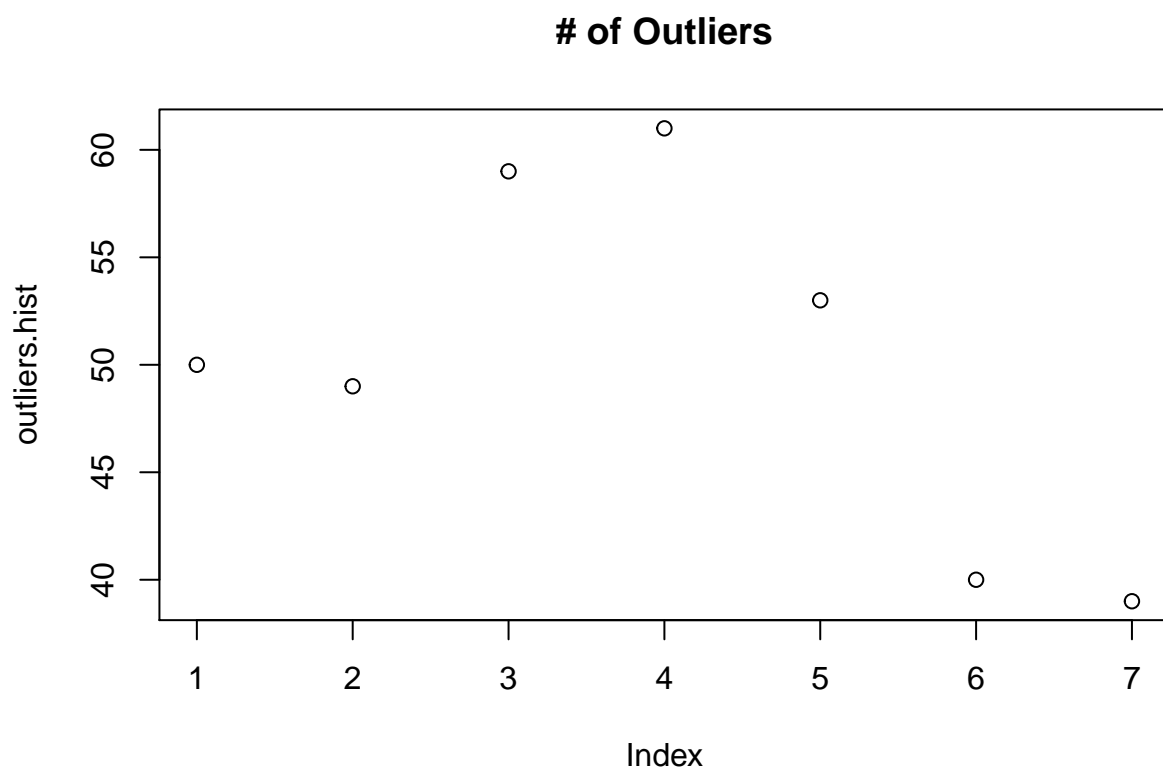
```
## [1] "step 1"
## [1] "added ageyrs"
## [1] "step 2"
## [1] "added POP_PCB10"
## [1] "step 3"
## [1] "added monocyte_pct"
## [1] "step 4"
## [1] "added POP_PCB2"
## [1] "step 5"
## [1] "added edu_cat"
## [1] "step 6"
## [1] "added smokenow"
## [1] "step 7"
## [1] "done choosing model"
```

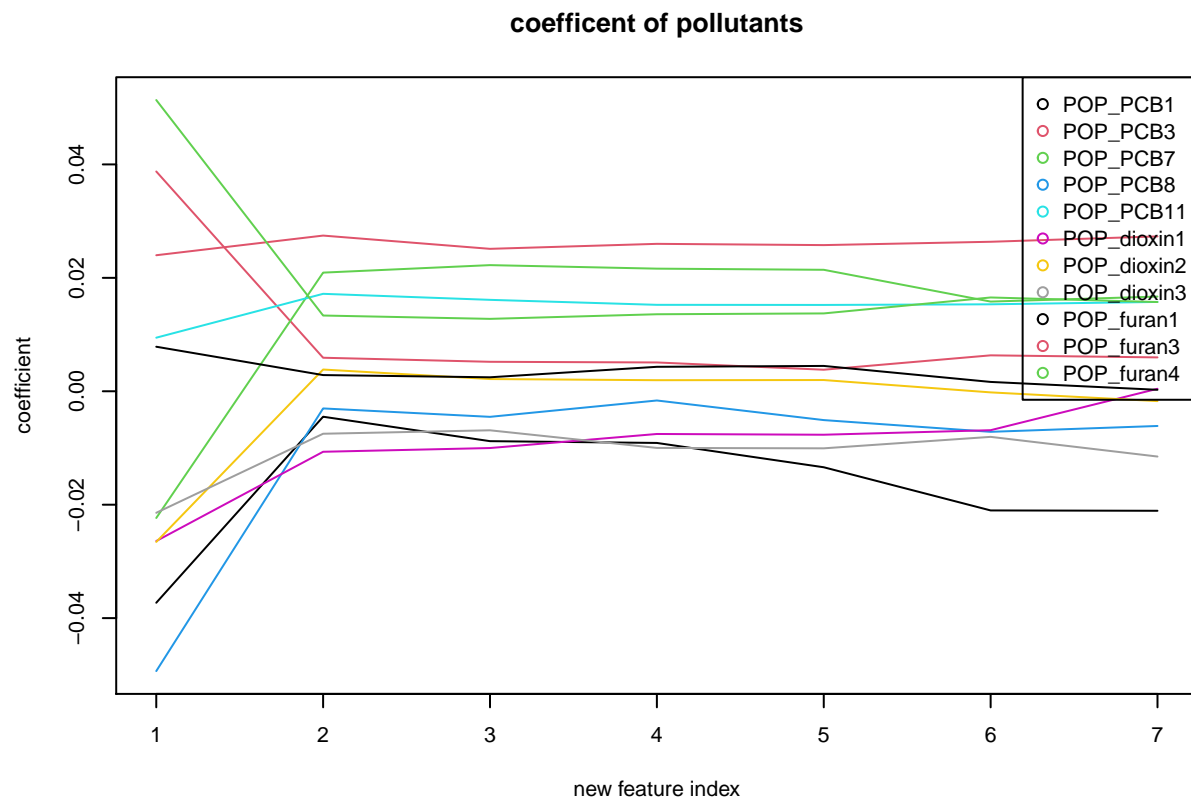




of Influential Points – DFFITS





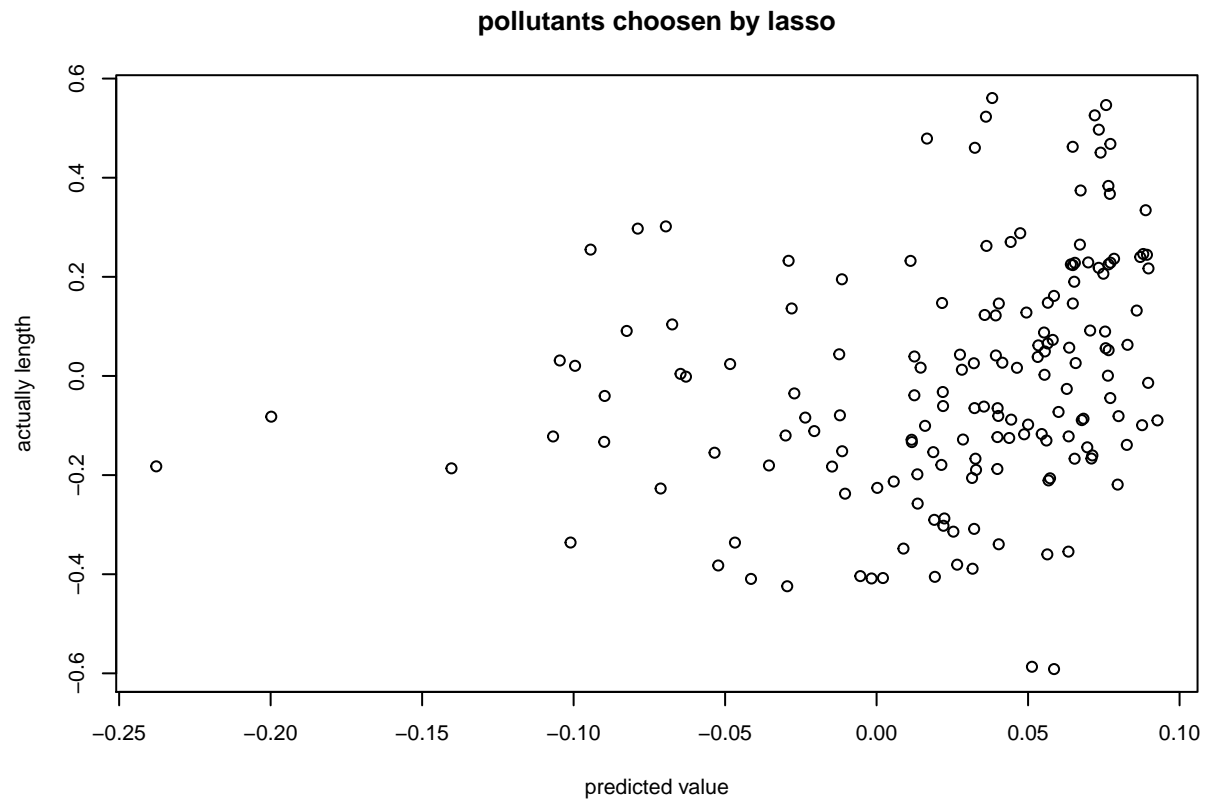


the last step vary by a lot because large aif -> large variance on beta, we shouldn't consider last s

if we only transform the pollutants

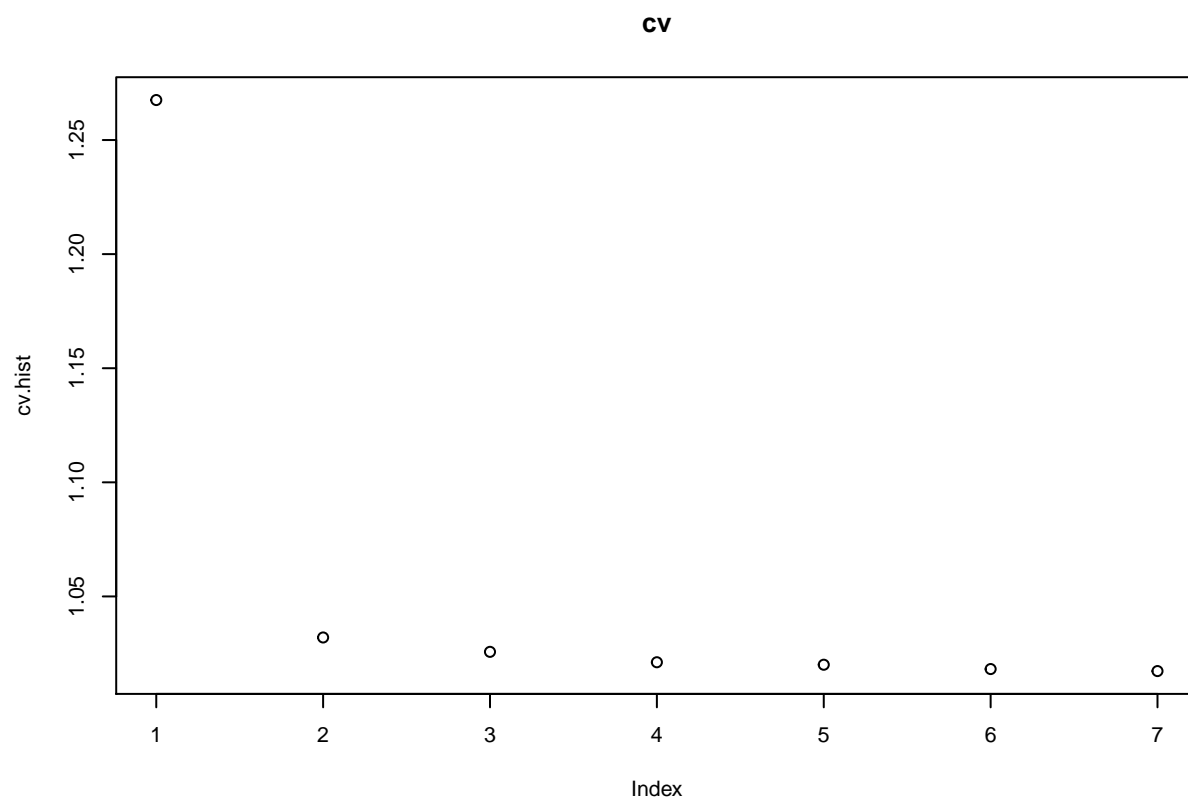
```
newdata1 = data
newdata1$length = log(data$length)
chosen.po.ind= which(lasso.on.pollutants(newdata1,TRUE)!=0)
```

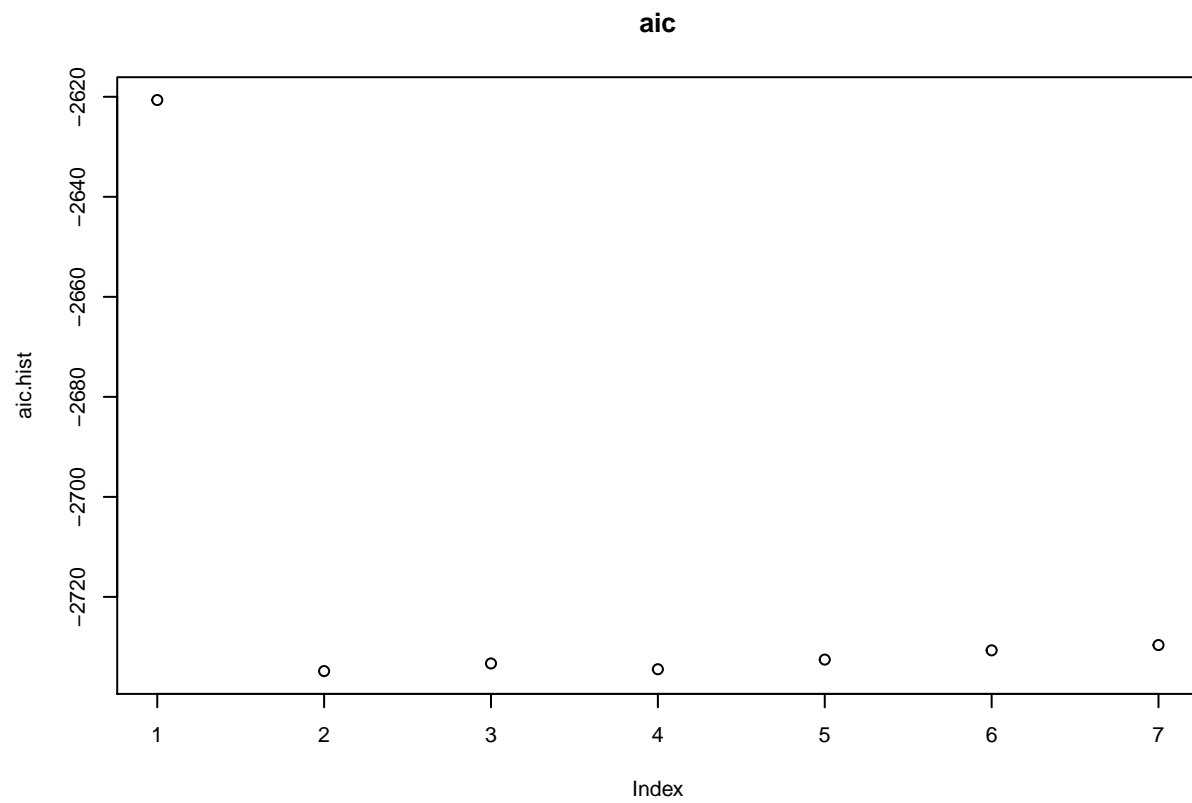
```
## [1] "mspe 0.054042181417014"
```



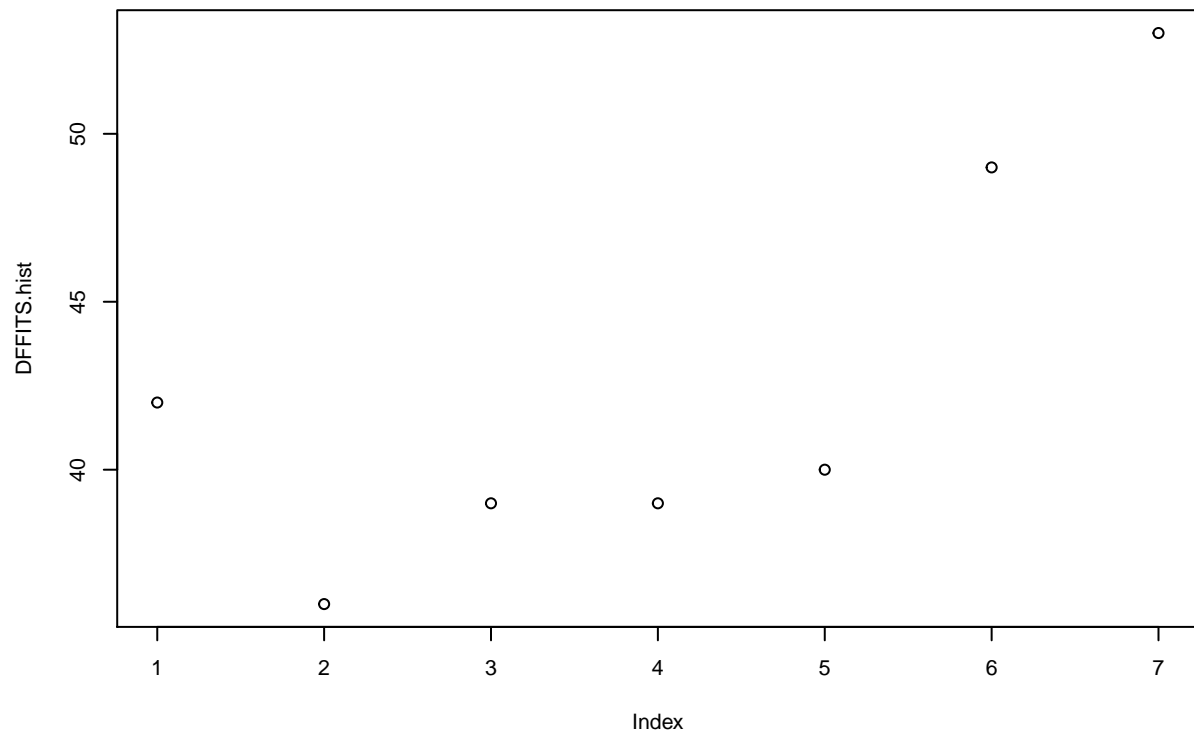
```
chosen.po.ind= chosen.po.ind[2:length(chosen.po.ind)]
chosen.pos = colnames(newdata)[chosen.po.ind]
finalModel_expr = paste("length~", paste(chosen.pos, collapse = "+"))
lasso.pollu.model1 = (lm(expr,data=newdata1))
t1=forward.change(newdata1, expr, TRUE)
```

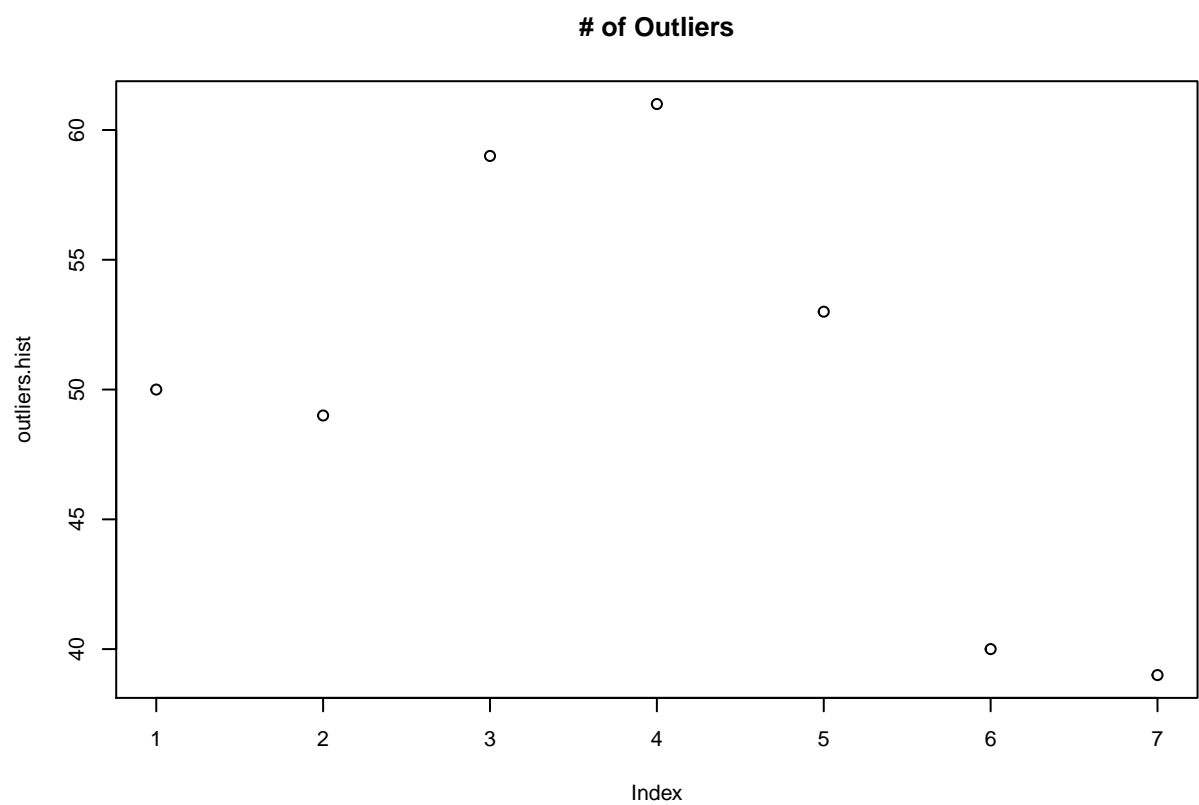
```
## [1] "step 1"
## [1] "added ageyrs"
## [1] "step 2"
## [1] "added POP_PCB10"
## [1] "step 3"
## [1] "added monocyte_pct"
## [1] "step 4"
## [1] "added POP_PCB2"
## [1] "step 5"
## [1] "added edu_cat"
## [1] "step 6"
## [1] "added smokenow"
## [1] "step 7"
## [1] "done choosing model"
```

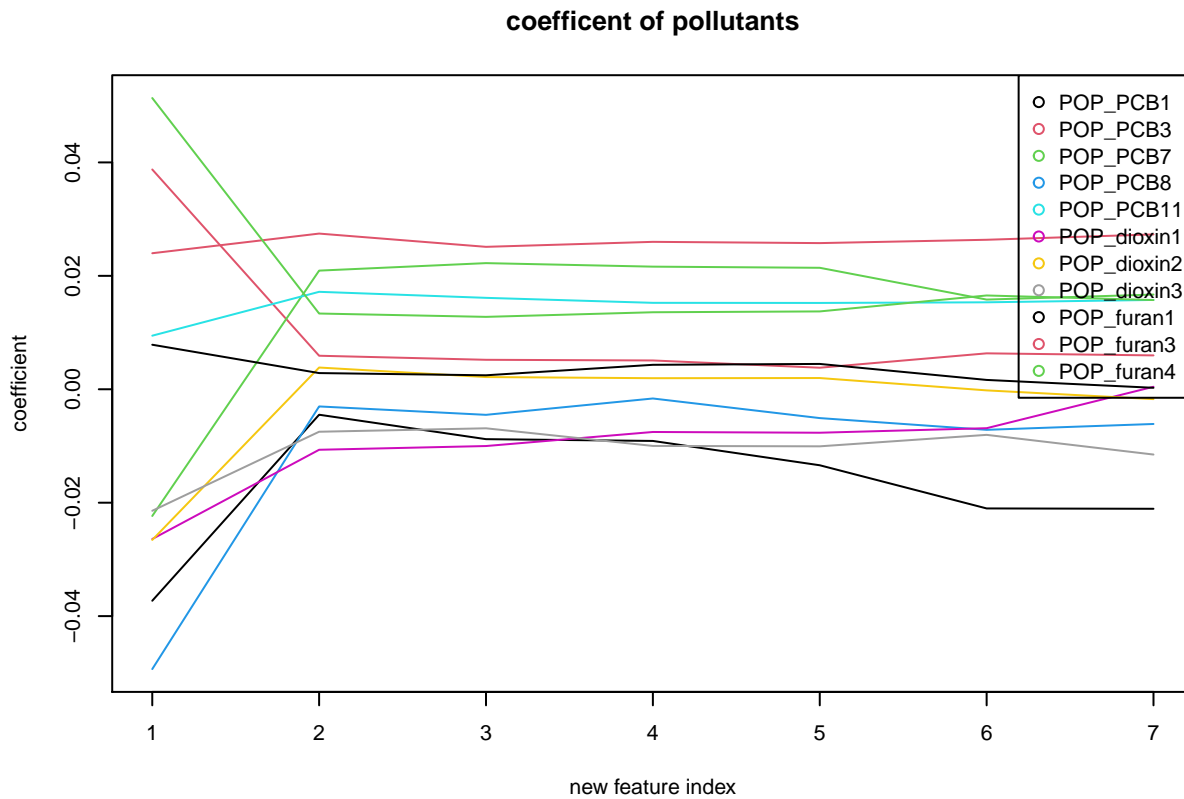




of Influential Points – DFFITS







```
finalModel <- t$models[[2]]
summary(finalModel)
```

```
##
## Call:
## lm(formula = expr, data = newdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.58353 -0.13296 -0.00239  0.13048  0.69185
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1575373  0.1315420   1.198  0.2314
## POP_PCB1     -0.0044886  0.0163771  -0.274  0.7841
## POP_PCB3      0.0059126  0.0176141   0.336  0.7372
## POP_PCB7      0.0209225  0.0181920   1.150  0.2504
## POP_PCB8     -0.0030293  0.0195994  -0.155  0.8772
## POP_PCB11     0.0171819  0.0097167   1.768  0.0774 .
## POP_dioxin1  -0.0106703  0.0143878  -0.742  0.4585
## POP_dioxin2   0.0038354  0.0128612   0.298  0.7656
## POP_dioxin3  -0.0074880  0.0161115  -0.465  0.6422
## POP_furan1   0.0028556  0.0184173   0.155  0.8768
## POP_furan3   0.0274499  0.0118849   2.310  0.0211 *
## POP_furan4   0.0133555  0.0125493   1.064  0.2875
## ageyrs       -0.0074950  0.0006772 -11.067 <2e-16 ***
```

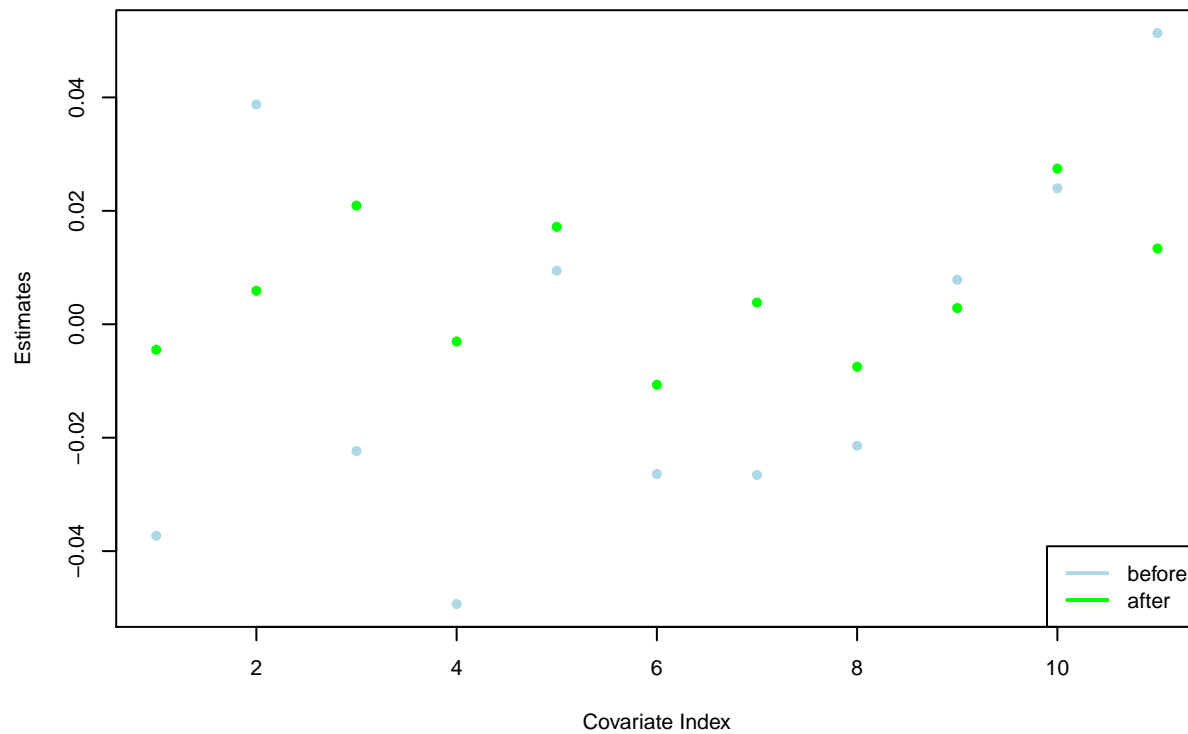
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2039 on 851 degrees of freedom
## Multiple R-squared:  0.2483, Adjusted R-squared:  0.2377
## F-statistic: 23.43 on 12 and 851 DF,  p-value: < 2.2e-16
summary(lm(data$length~data$POP_furan4))

##
## Call:
## lm(formula = data$length ~ data$POP_furan4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52661 -0.17867 -0.02668  0.15557  1.29734
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.0584473   0.0122963   86.079   <2e-16 ***
## data$POP_furan4 -0.0003581   0.0007682   -0.466    0.641
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2504 on 862 degrees of freedom
## Multiple R-squared:  0.0002521, Adjusted R-squared:  -0.0009077
## F-statistic: 0.2173 on 1 and 862 DF,  p-value: 0.6412
finalModel$coefficients[ finalModel$coefficients<0.01]

##      POP_PCB1      POP_PCB3      POP_PCB8 POP_dioxin1 POP_dioxin2 POP_dioxin3
## -0.004488595  0.005912602 -0.003029345 -0.010670329  0.003835385 -0.007487969
##      POP_furan1      ageyrs
##  0.002855554 -0.007495015
finalModel$coefficients[ finalModel$coefficients>0.01]

## (Intercept)      POP_PCB7      POP_PCB11 POP_furan3 POP_furan4
##  0.15753730  0.02092253  0.01718193  0.02744992  0.01335549
numpara= length(lasso.pollu.model$coefficients)-1
# excluding length in full model
plot(rep(1:numpara,2), c(lasso.pollu.model$coefficients[-1],finalModel$coefficients[-c(1,numpara+2)]),
     col=rep(c("lightblue","green"), each=numpara), pch = 16,
     xlab = "Covariate Index", ylab="Estimates",
     main = "pollutants coefficient before&after adding other features")
legend("bottomright", legend=c("before", "after"), col=c("lightblue","green"), lty=1, lwd = 2)
```

pollutants coefficient before&after adding other features



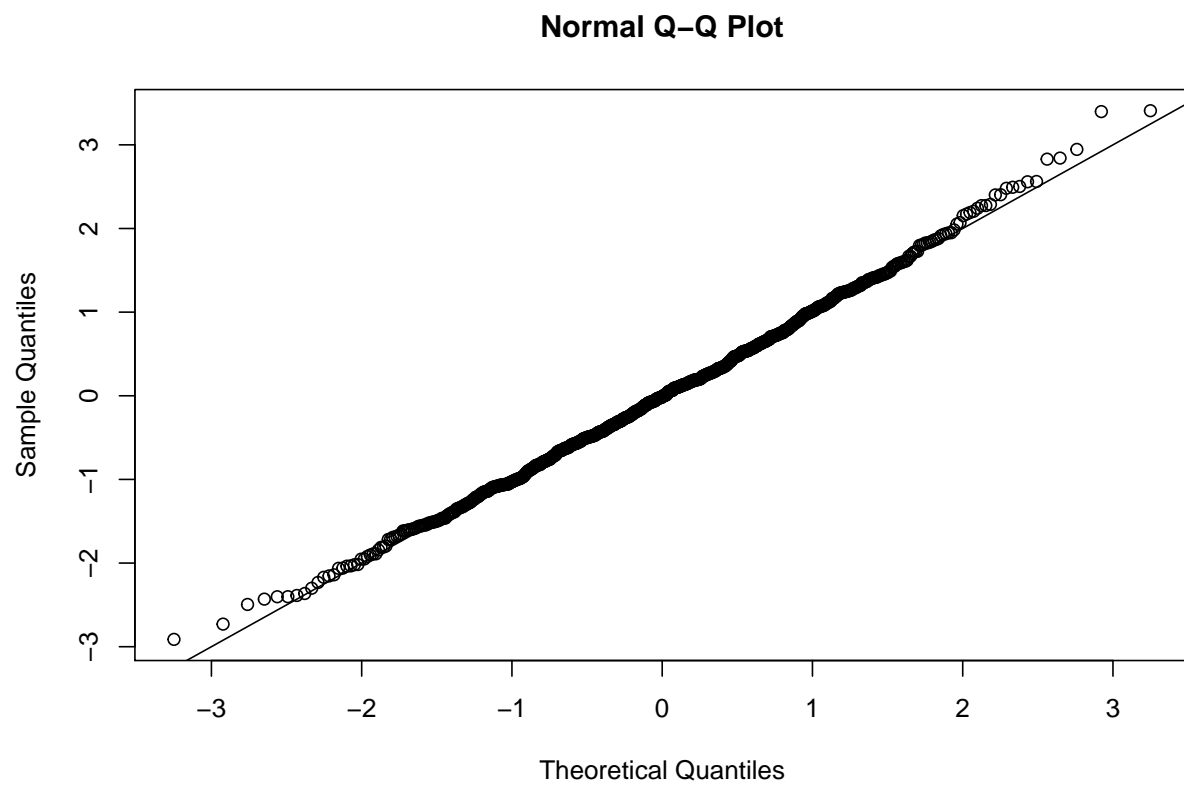
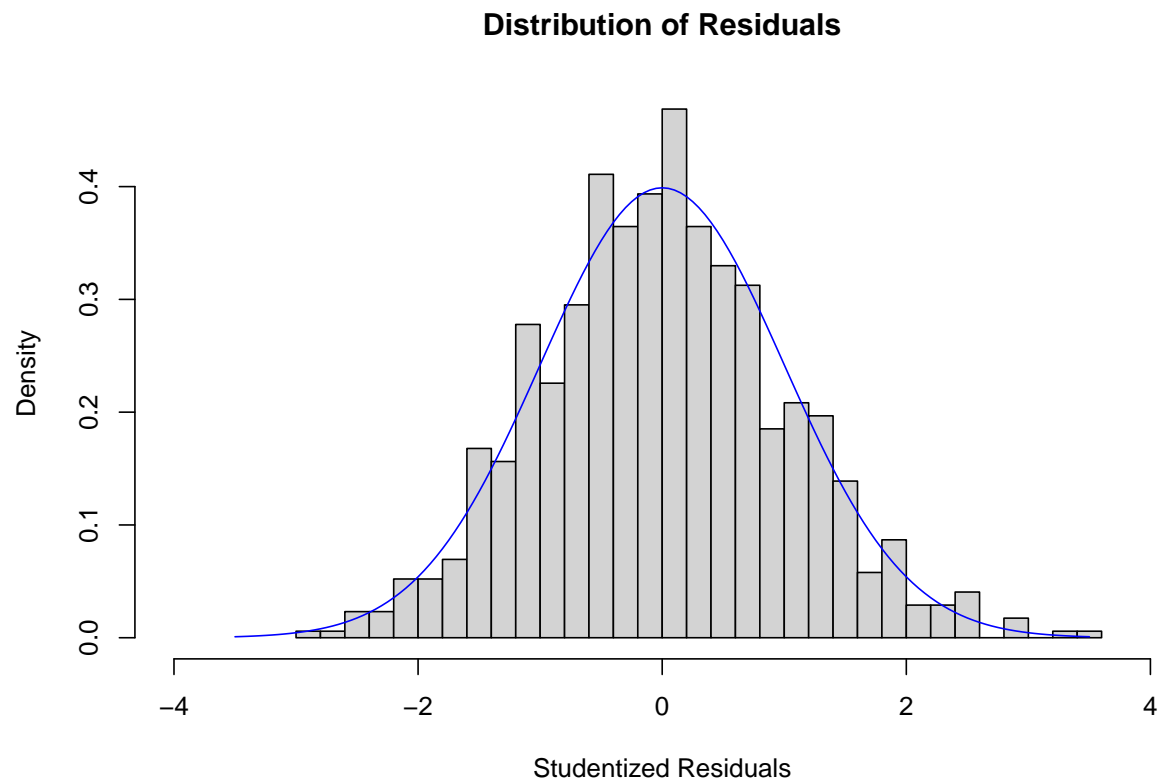
```
coef(lasso.pollu.model)
```

```
## (Intercept)    POP_PCB1    POP_PCB3    POP_PCB7    POP_PCB8    POP_PCB11
##  0.823419273 -0.037291353  0.038756534 -0.022342147 -0.049337110  0.009449720
##  POP_dioxin1 POP_dioxin2 POP_dioxin3  POP_furan1  POP_furan3  POP_furan4
## -0.026408081 -0.026566899 -0.021431406  0.007854683  0.023985959  0.051351220
```

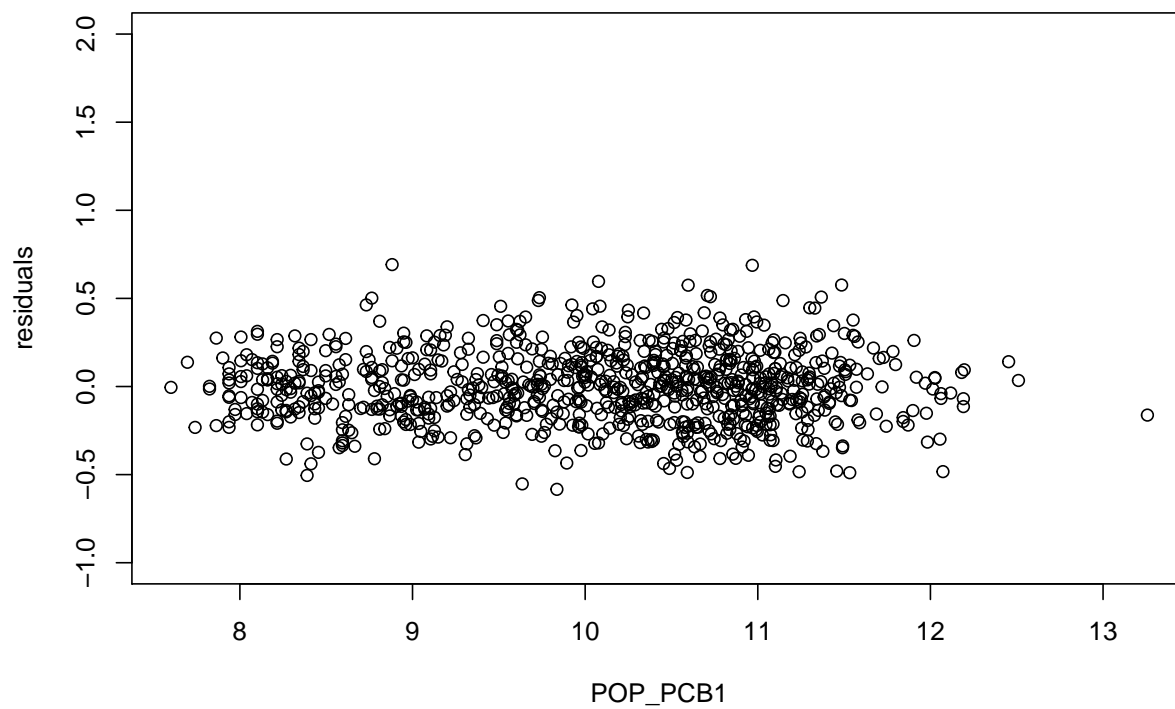
```
coef(finalModel)
```

```
## (Intercept)    POP_PCB1    POP_PCB3    POP_PCB7    POP_PCB8    POP_PCB11
##  0.157537300 -0.004488595  0.005912602  0.020922531 -0.003029345  0.017181935
##  POP_dioxin1 POP_dioxin2 POP_dioxin3  POP_furan1  POP_furan3  POP_furan4
## -0.010670329  0.003835385 -0.007487969  0.002855554  0.027449917  0.013355489
##      ageyrs
## -0.007495015
```

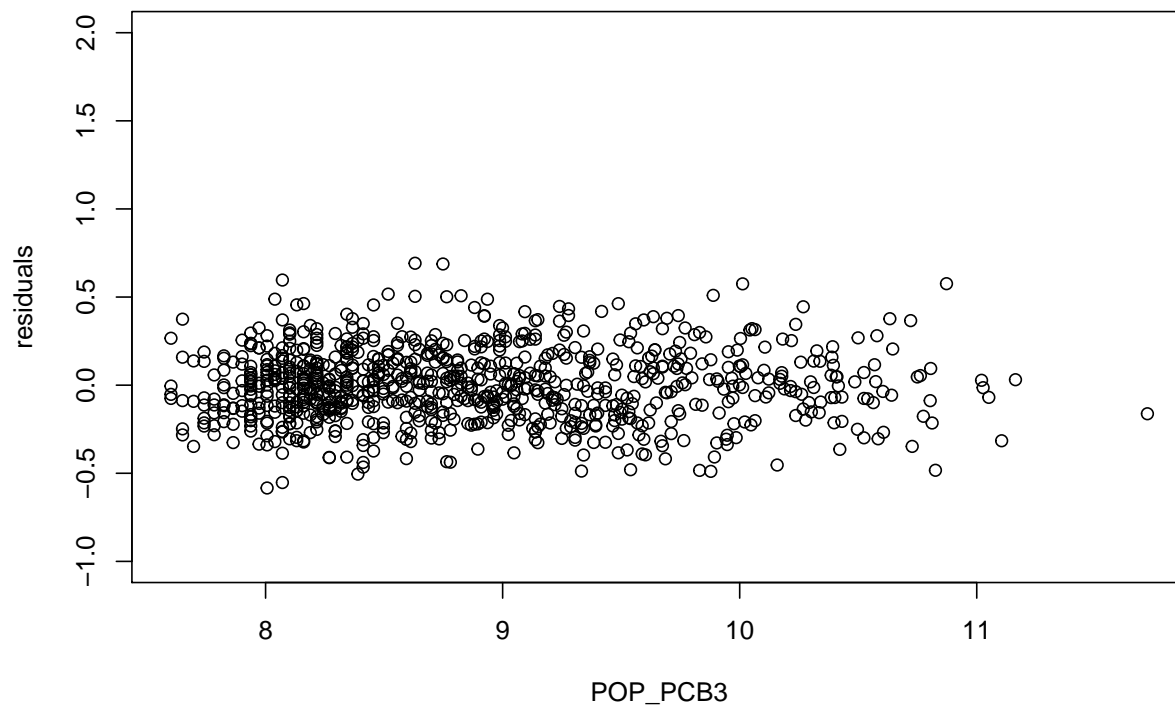
```
errorAnalysis(t$models[[2]])
```



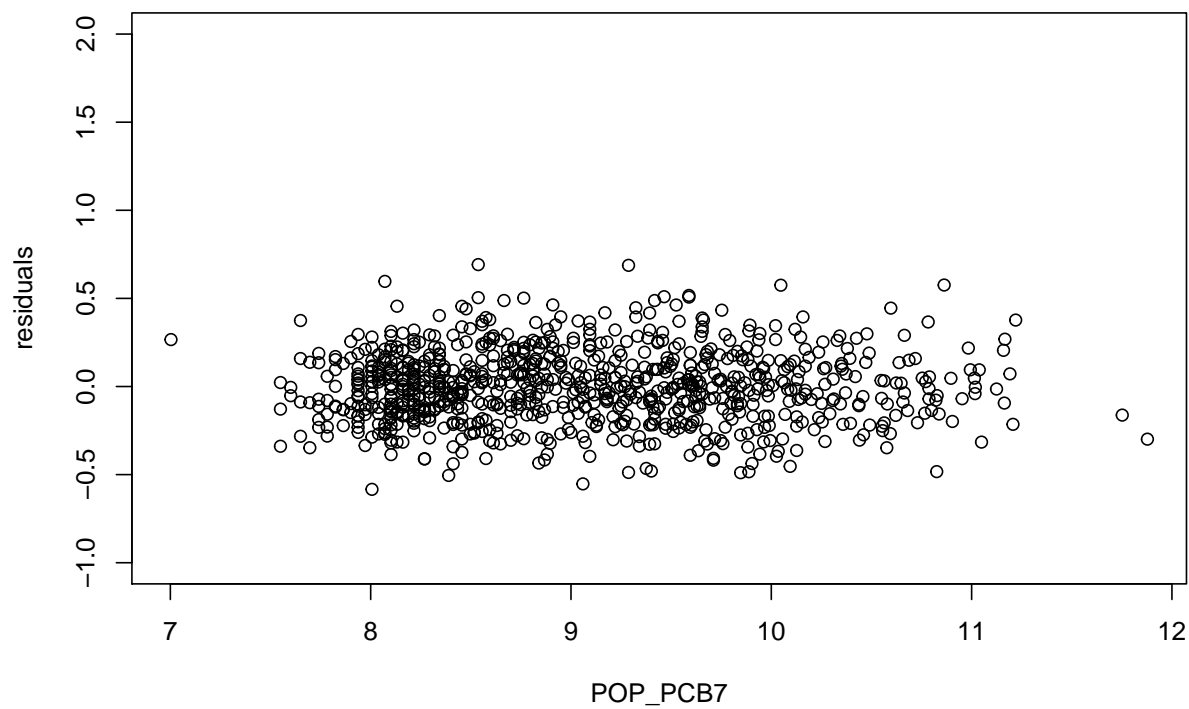
Residuals vs POP_PCB1



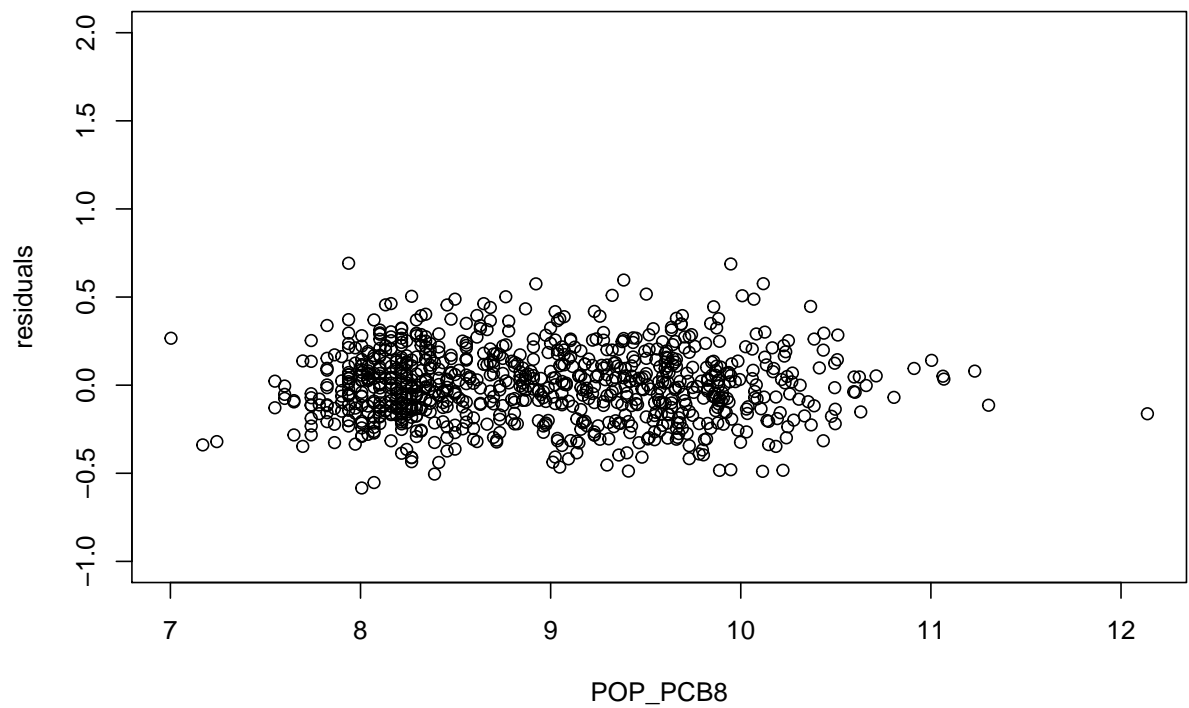
Residuals vs POP_PCB3



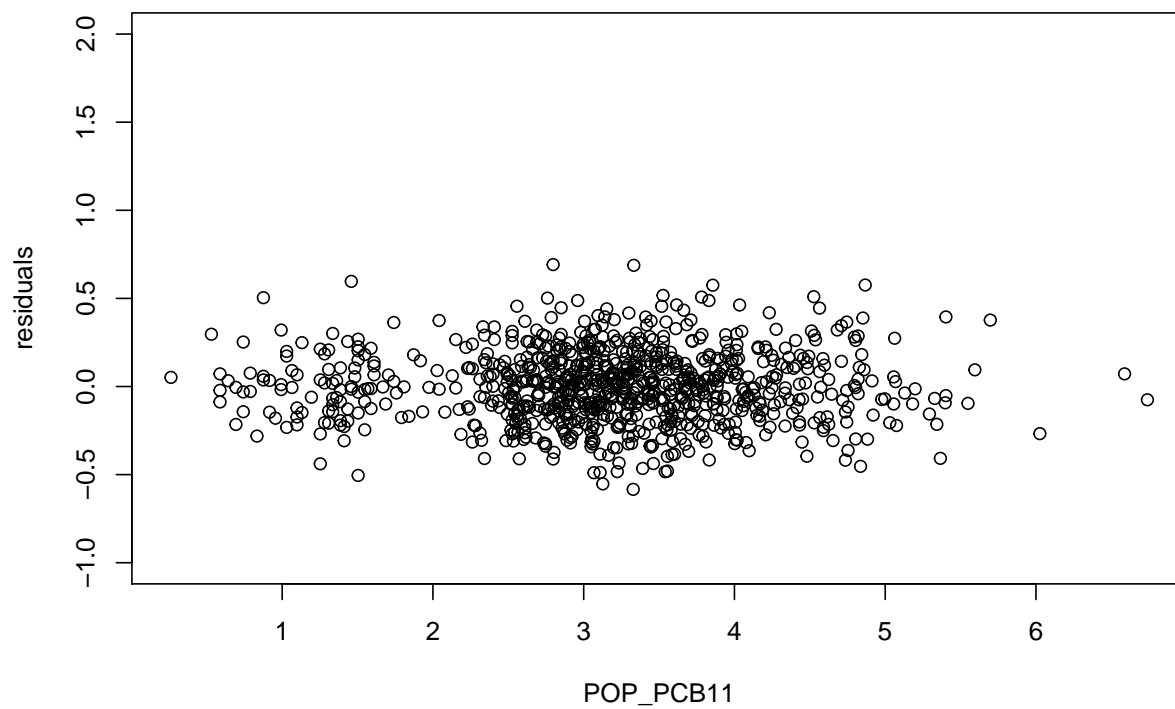
Residuals vs POP_PCB7



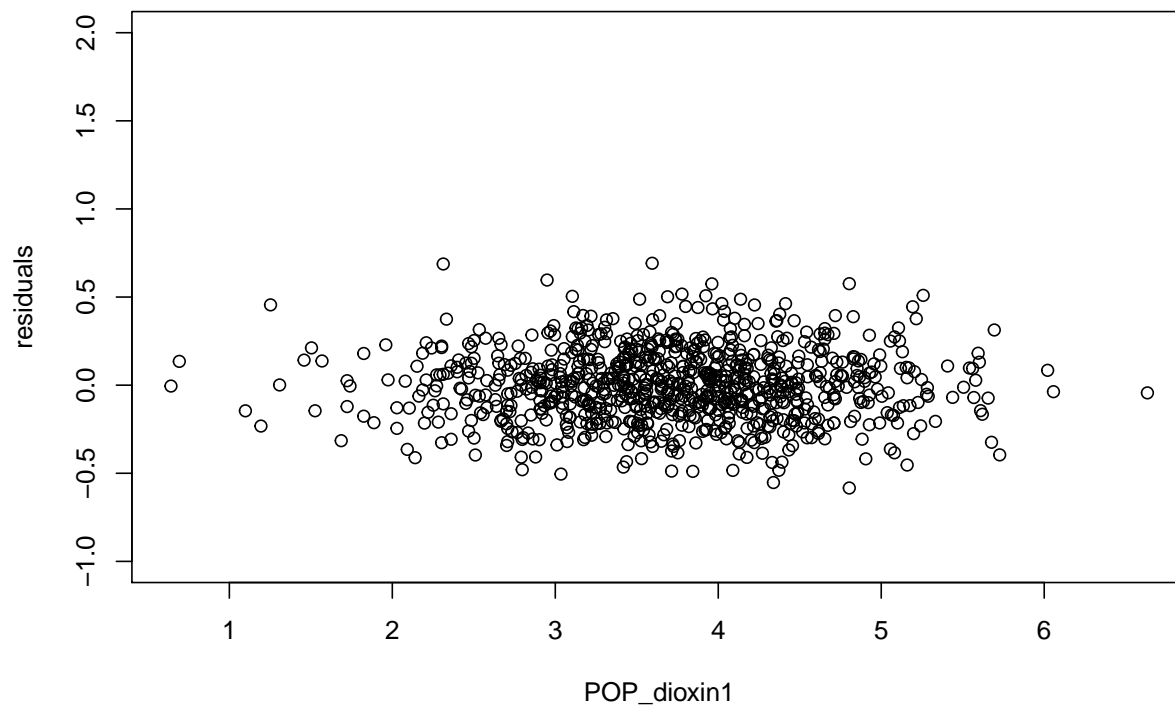
Residuals vs POP_PCB8



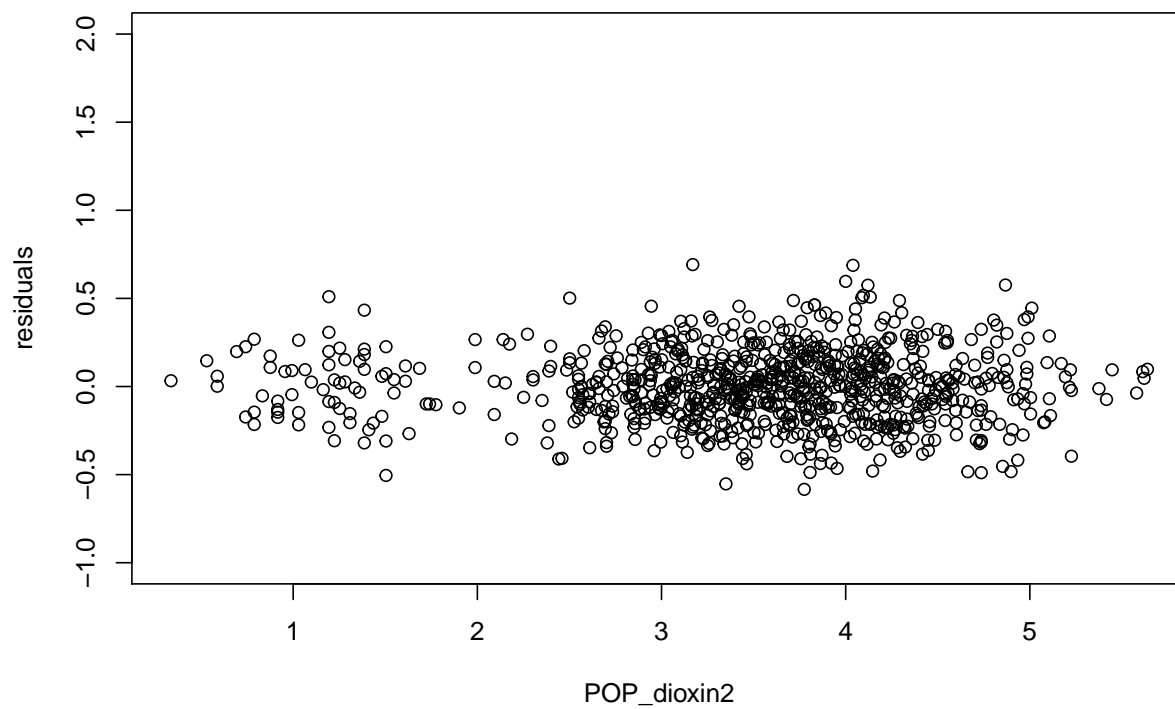
Residuals vs POP_PCB11



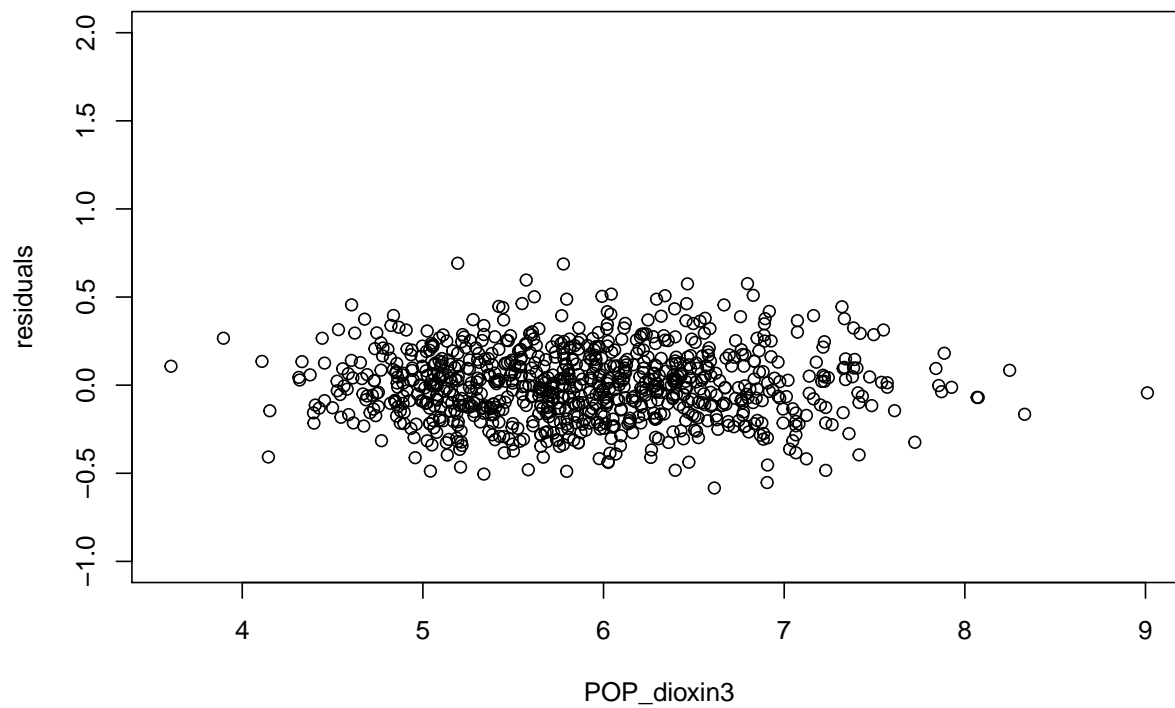
Residuals vs POP_dioxin1



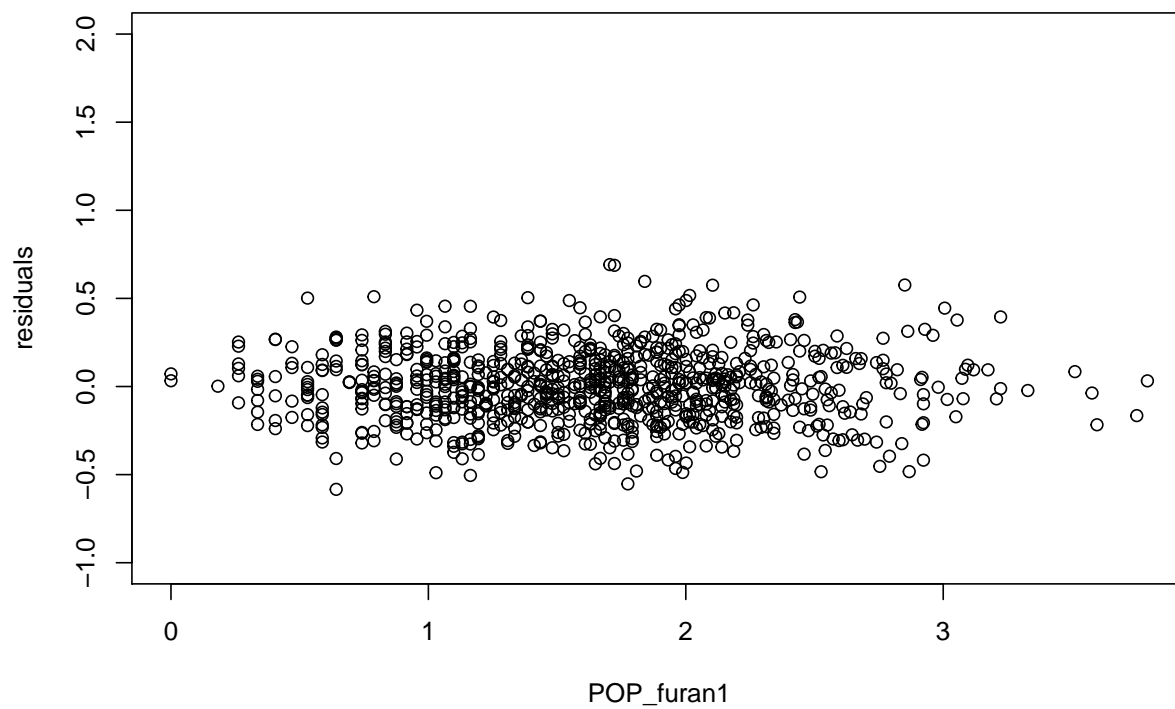
Residuals vs POP_dioxin2



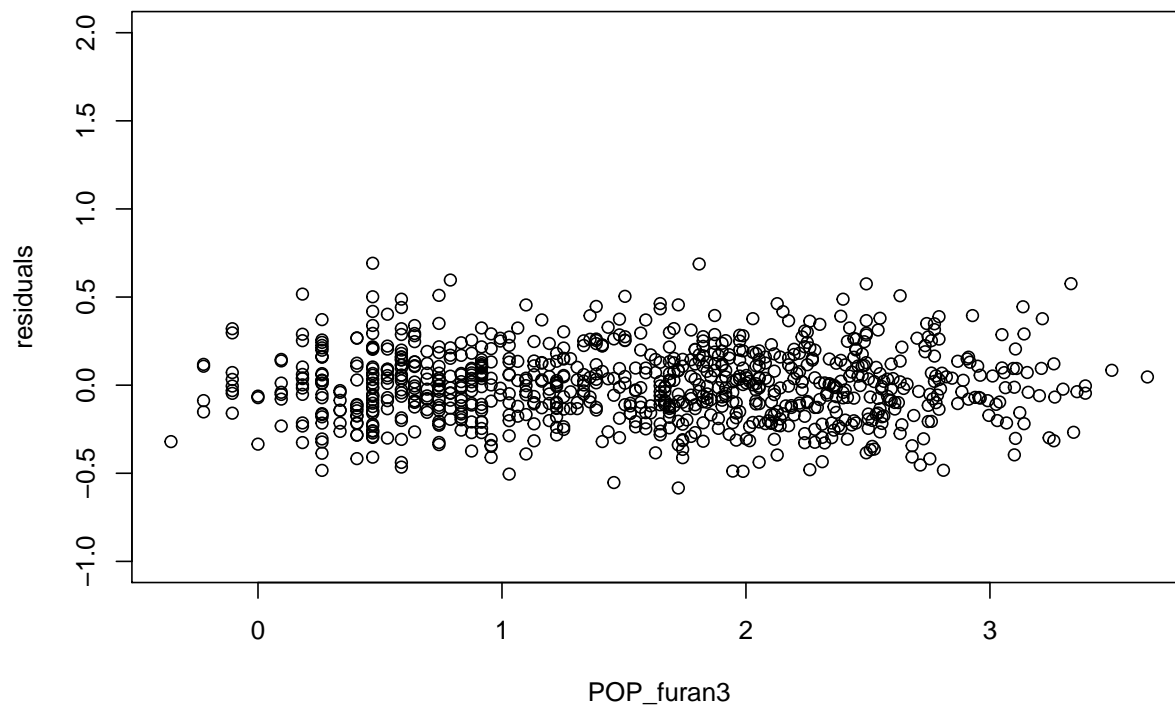
Residuals vs POP_dioxin3



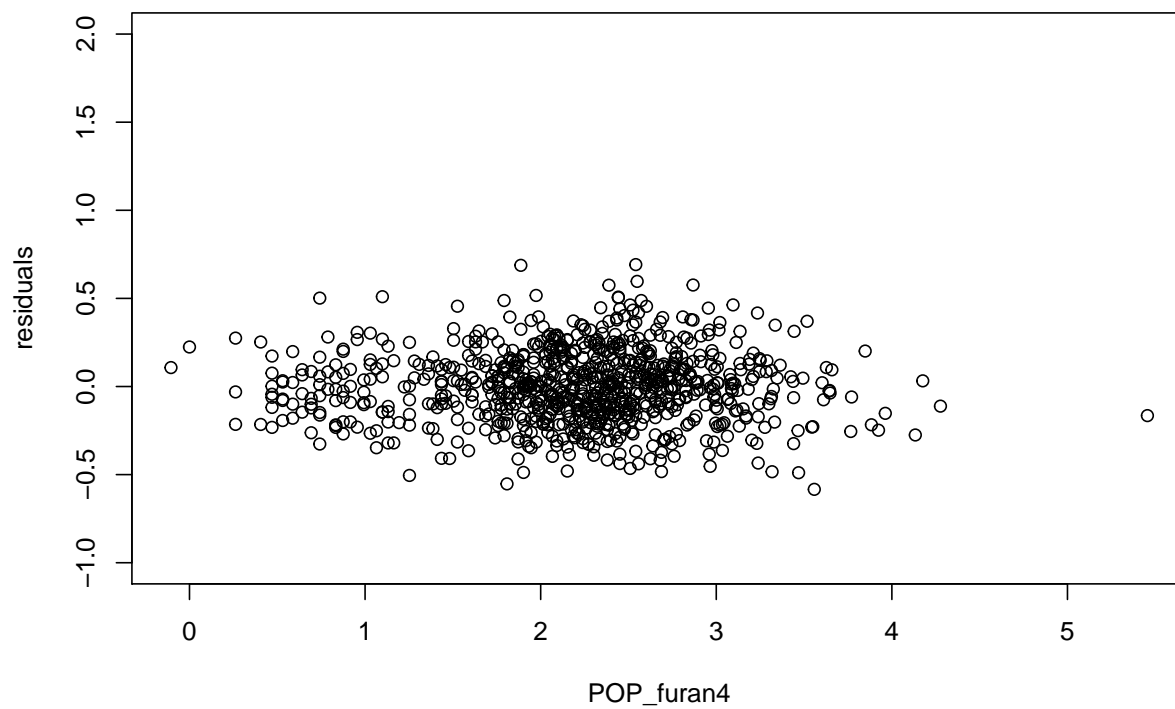
Residuals vs POP_furan1



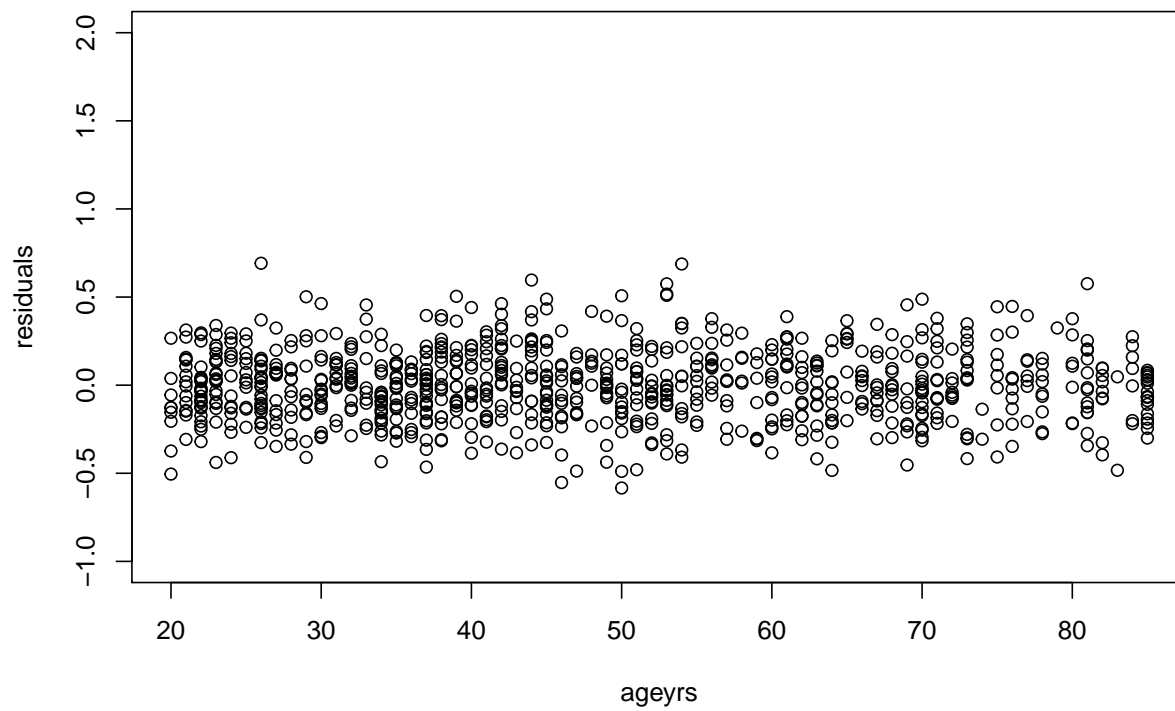
Residuals vs POP_furan3

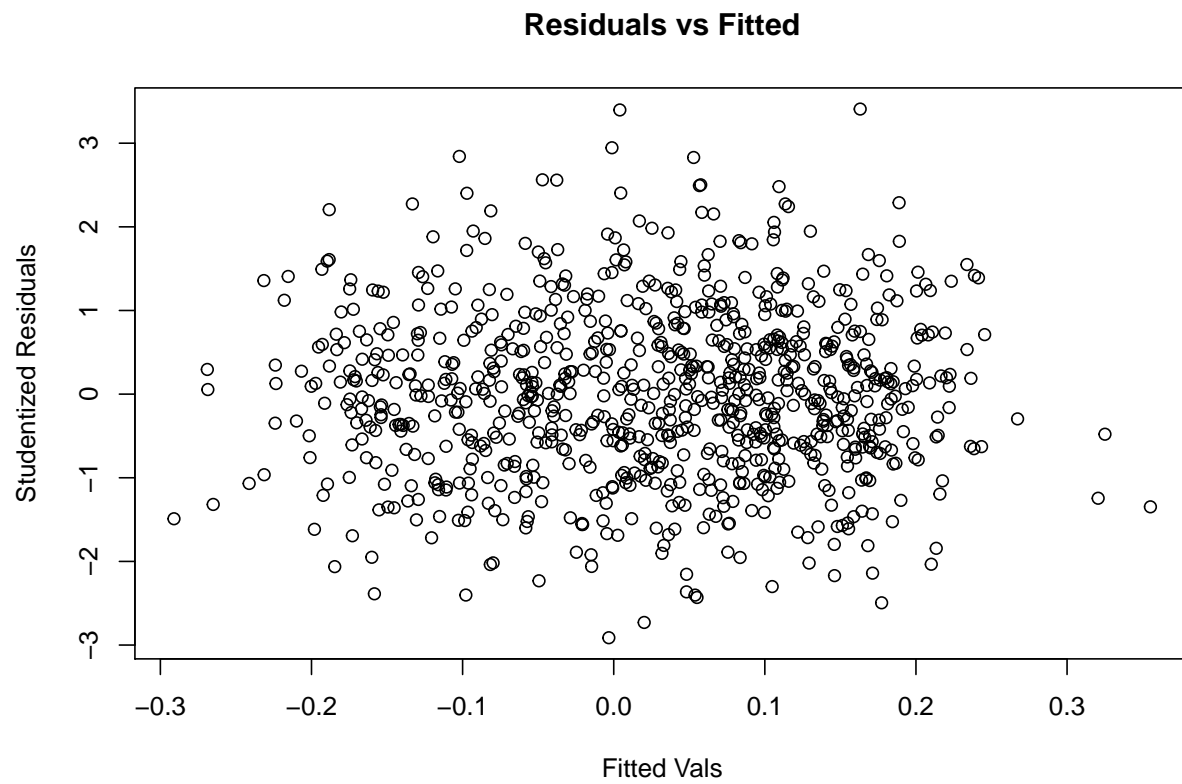


Residuals vs POP_furan4



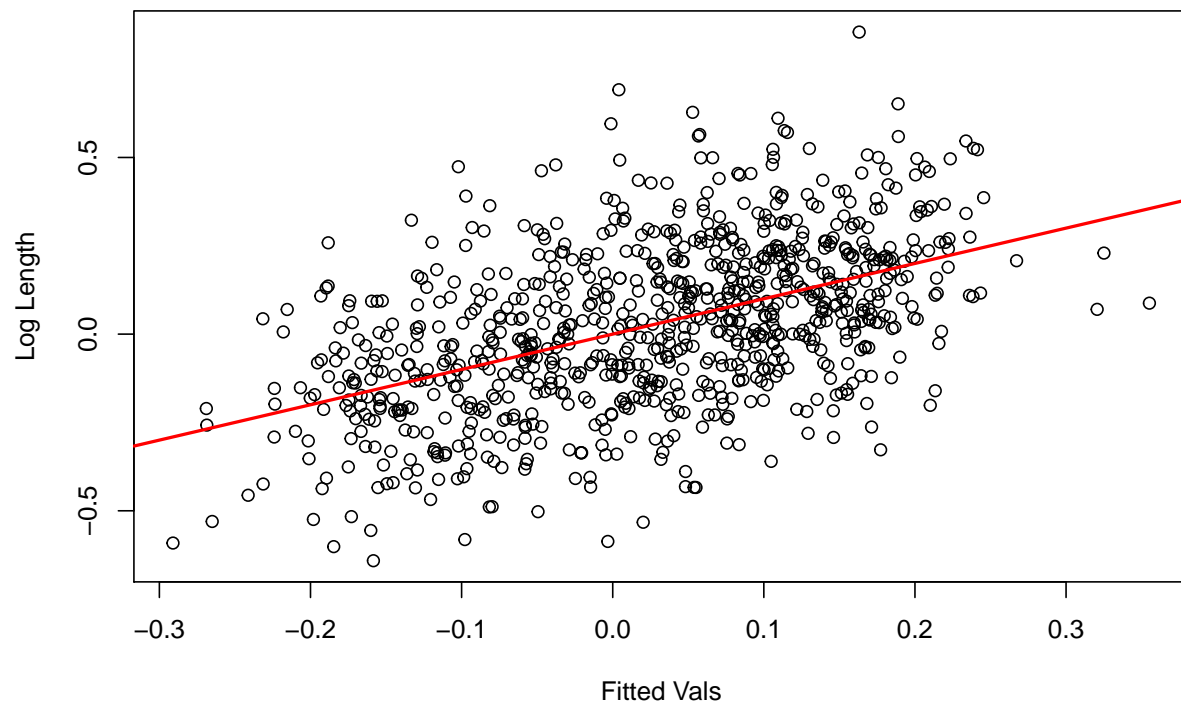
Residuals vs ageyrs



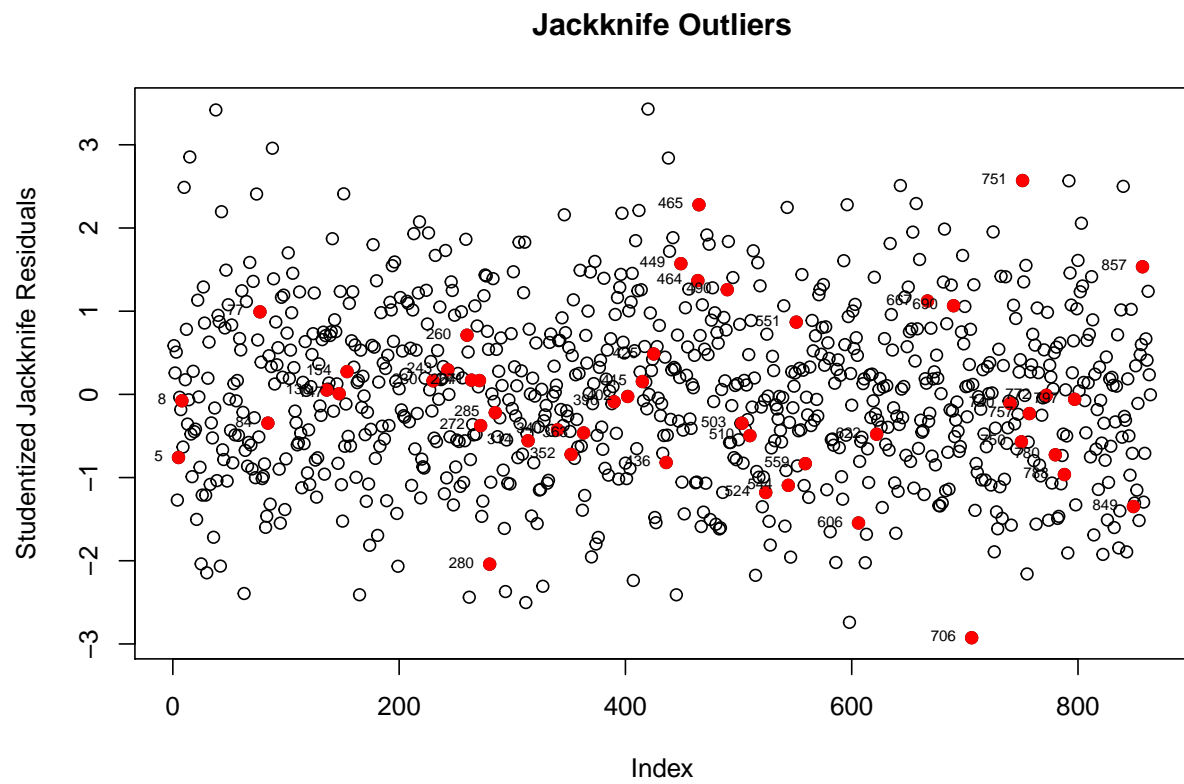


```
plot(x = finalModel$fitted.values, y = log( data$length),  
     main = "Fitted values vs. Log Length", ylab = "Log Length", xlab = "Fitted Vals")  
abline(a = 0, b = 1,col = "red", lwd= 2)
```

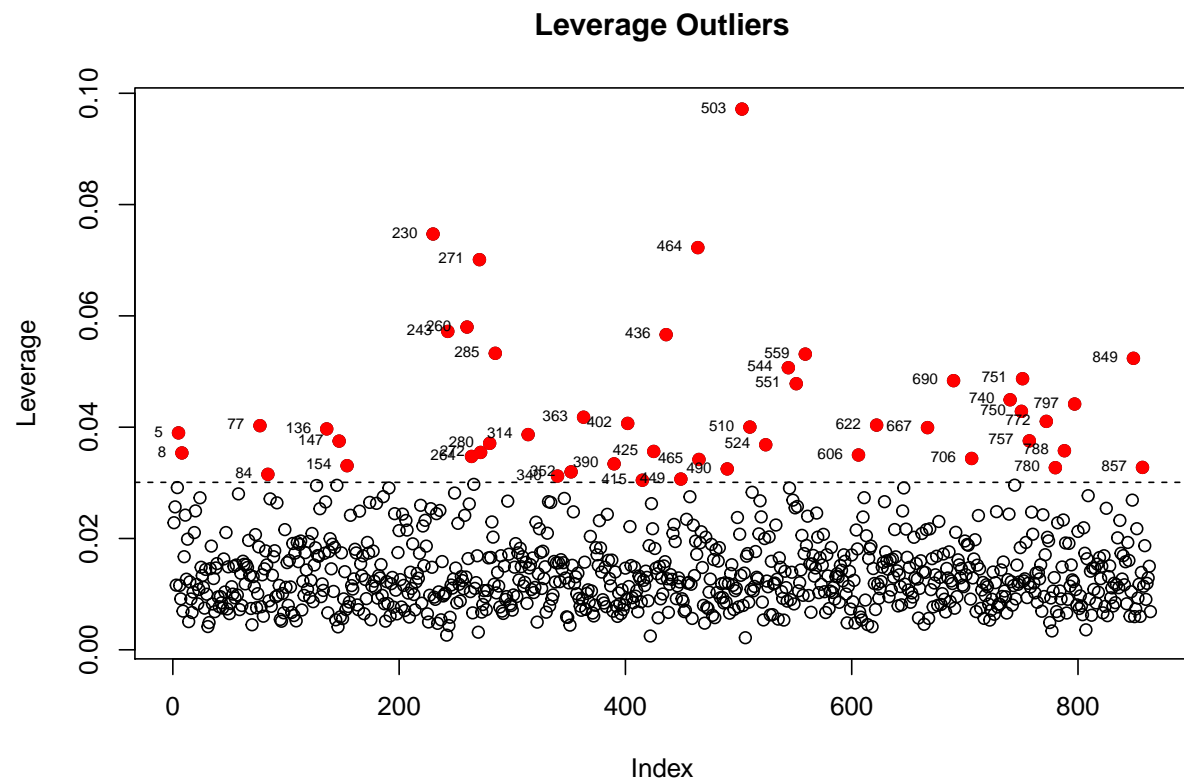
Fitted values vs. Log Length



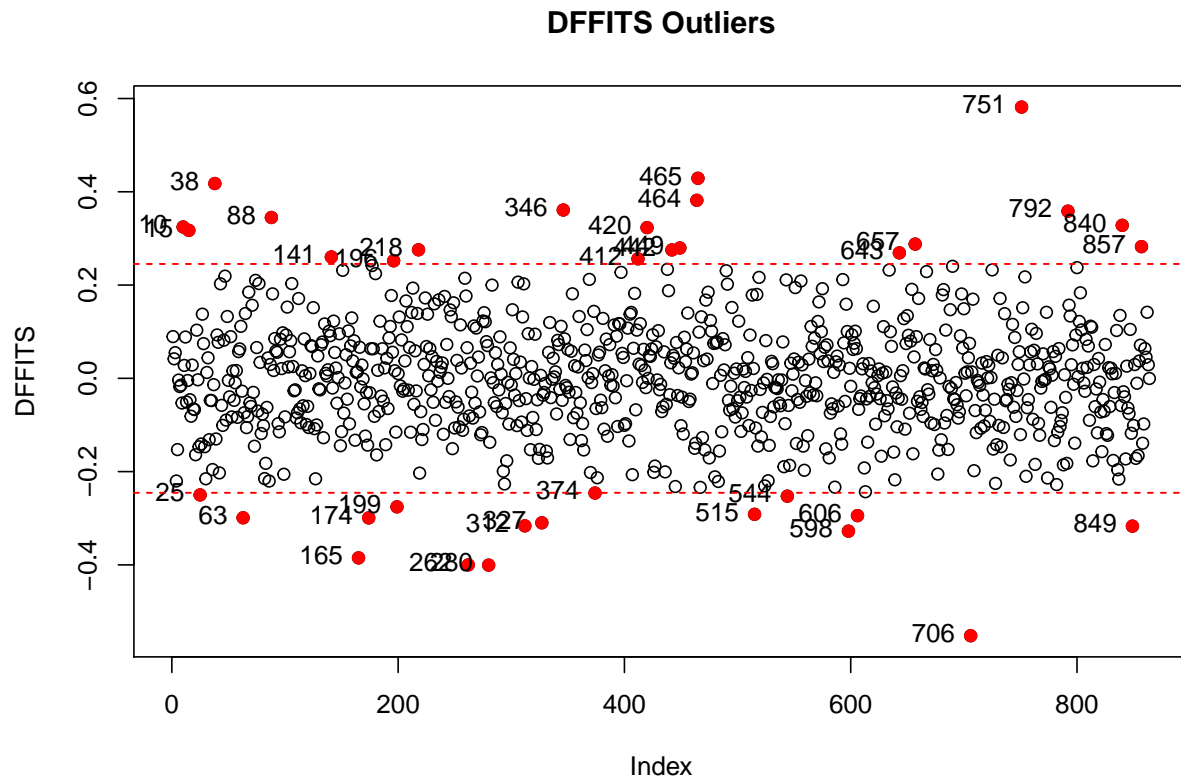
```
plot.jackknife.res(finalModel)
```



```
plot.outliers(finalModel)
```



```
plot.DFFITS(finalModel)
```



```
# Removing DFFITS outliers
```

```
DFFITS_ol <- DFFITS(finalModel)
newdata4 <- newdata[-DFFITS_ol,]
finalModel_DFFITS <- lm(finalModel_expr, data = newdata4)
summary(finalModel_DFFITS)
```

```
##
## Call:
## lm(formula = finalModel_expr, data = newdata4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.59229 -0.14882 -0.00055  0.13824  0.62111
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.807470   0.097640   8.270 5.37e-16 ***
## POP_PCB1     -0.024933   0.016397  -1.521  0.12875
## POP_PCB7       0.006905   0.013154   0.525  0.59977
## POP_PCB8     -0.056474   0.019516  -2.894  0.00391 **
## POP_dioxin1  -0.016785   0.010644  -1.577  0.11521
## POP_dioxin2  -0.008635   0.012502  -0.691  0.48998
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.206 on 822 degrees of freedom
## Multiple R-squared:  0.1221, Adjusted R-squared:  0.1168
```



```
## F-statistic: 22.87 on 5 and 822 DF, p-value: < 2.2e-16
```

```
# RMSE  
sqrt(mean(resid(finalModel_DFFITS)^2))
```

```
## [1] 0.2052092
```

```
# Removing leverage outliers  
lev_ol <- outliers(finalModel)  
newdata5 <- newdata[-lev_ol,]  
finalModel_lev <- lm(finalModel_expr, data = newdata5)  
summary(finalModel_lev)
```

```
##  
## Call:  
## lm(formula = finalModel_expr, data = newdata5)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -0.60311 -0.15760 -0.00016  0.13742  0.74083   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept)  0.809557   0.109770   7.375 4.07e-13 ***  
## POP_PCB1     -0.033307   0.019312  -1.725  0.0850 .      
## POP_PCB7      0.009009   0.014866   0.606  0.5447        
## POP_PCB8     -0.049407   0.022205  -2.225  0.0264 *      
## POP_dioxin1 -0.021646   0.012262  -1.765  0.0779 .      
## POP_dioxin2 -0.002069   0.015505  -0.133  0.8939        
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.2215 on 809 degrees of freedom  
## Multiple R-squared:  0.1019, Adjusted R-squared:  0.09636  
## F-statistic: 18.36 on 5 and 809 DF, p-value: < 2.2e-16
```

```
# RMSE  
sqrt(mean(resid(finalModel_lev)^2))
```

```
## [1] 0.220676
```

```
# Removing jackknife outliers  
newdata6 <- newdata[-c(456, 751, 280, 706),]  
finalModel_JK <- lm(finalModel_expr, data = newdata6)  
summary(finalModel_JK)
```

```
##  
## Call:  
## lm(formula = finalModel_expr, data = newdata6)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -0.60295 -0.15562 -0.00194  0.13890  0.74434   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept)  0.808306   0.100898   8.011 3.72e-15 ***
```

```

## POP_PCB1      -0.025974    0.017057   -1.523   0.12818
## POP_PCB7       0.005403    0.013631    0.396   0.69193
## POP_PCB8      -0.052655    0.020327   -2.590   0.00975 **
## POP_dioxin1   -0.020239    0.011087   -1.825   0.06828 .
## POP_dioxin2   -0.007079    0.012460   -0.568   0.57007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2197 on 854 degrees of freedom
## Multiple R-squared:  0.1071, Adjusted R-squared:  0.1019
## F-statistic: 20.49 on 5 and 854 DF,  p-value: < 2.2e-16

# RMSE
sqrt(mean(resid(finalModel_JK)^2))

## [1] 0.2189371

# F-test to test for if only POP_furan3 and ageyrs coefficients are not equal to 0
finalModel_red <- lm(length ~ POP_furan3 + ageyrs, data = newdata)
dff <- finalModel$df
dfr <- finalModel_red$df
SSRes_full <- sum(residuals(finalModel)^2)
SSRes_red <- sum(residuals(finalModel_red)^2)

Fobs <- (SSRes_red-SSRes_full)/(dfr-dff)/(SSRes_full/dff)
pf(Fobs,df1 = (dfr-dff),df2 = dff, lower.tail = FALSE)

## [1] 0.3556256

```