finalproject

phantomOfLaMancha

3/26/2021

```
# functions we might need

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)
}

kfold.cv = function(data, M, ind, kfolds=10){
  mspe = rep(0, kfolds)
  if(length(levels(ind))!= kfolds){
    stop("given index has incorrect number of folds")
  }
  for(ii in 1:Kfolds) {
    train.ind <- which(ind!=ii) # training observations
    M.cv <- update(M, subset = train.ind)
    # cross-validation residuals
    M.res <- data$length[-train.ind] - # test observations
```

```
      predict(M.cv, newdata = data[-train.ind,]) # prediction with training data
    # mspe
    mspe[ii] <- mean(M.res^2)
  }
  mean(mspe)
}

get_col <- function(mat,i,j, breaks, cols=NULL, palette="Blues") {
    if (is.null(cols)) {
        cols <- brewer.pal(length(breaks)+1, palette)}
    val <- 1
    for (b in breaks) {
      if (is.na(mat [i,j])){
        val <- 0
      }
      else if (mat[i,j] > b) {
            val <- val + 1}
            }
    cols[val]
    }

require(RColorBrewer)
```

```
## Loading required package: RColorBrewer
```

```
col_areas <- function(matrix,
                                                breaks=NULL,
                                                cols=NULL,
                                                palette="Blues",
                                                xlab="West    <----------->    East",
                                                ylab="South   <----------->   North",
                                                ...){
    if (is.null(breaks)) {
            breaks <- unique(fivenum(matrix))}

  plot(c(0, 100*ncol(matrix)),
            c(0, 100*nrow(matrix)), frame.plot=TRUE,
            type="n",
            xlab=xlab,
            ylab=ylab, axes=FALSE, ...)

  nr <- nrow(matrix)
  nc <- ncol(matrix)
    for (i in 1:nr) {
        for (j in 1:nc) {
            rect((j-1)*100,
                 (nr-i+1)*100,
                 j*100,
                 (nr-i)*100,
                 border=NA,
                 col=get_col(matrix,i,j,breaks,cols,palette))
                 }
               }
}
```

understanding our polulation:

```
library("eikosograms")
```

```
## Warning: package 'eikosograms' was built under R version 4.0.4
```

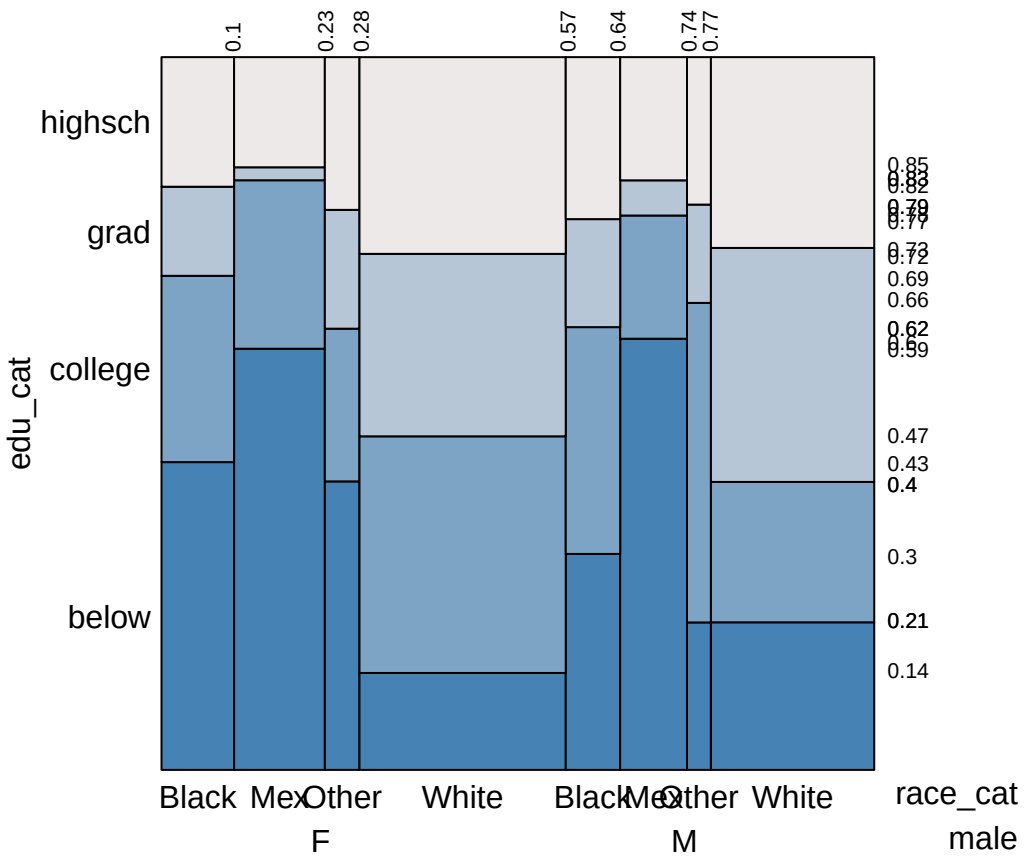
```
library("venneuler")
```

```
## Warning: package 'venneuler' was built under R version 4.0.3
```

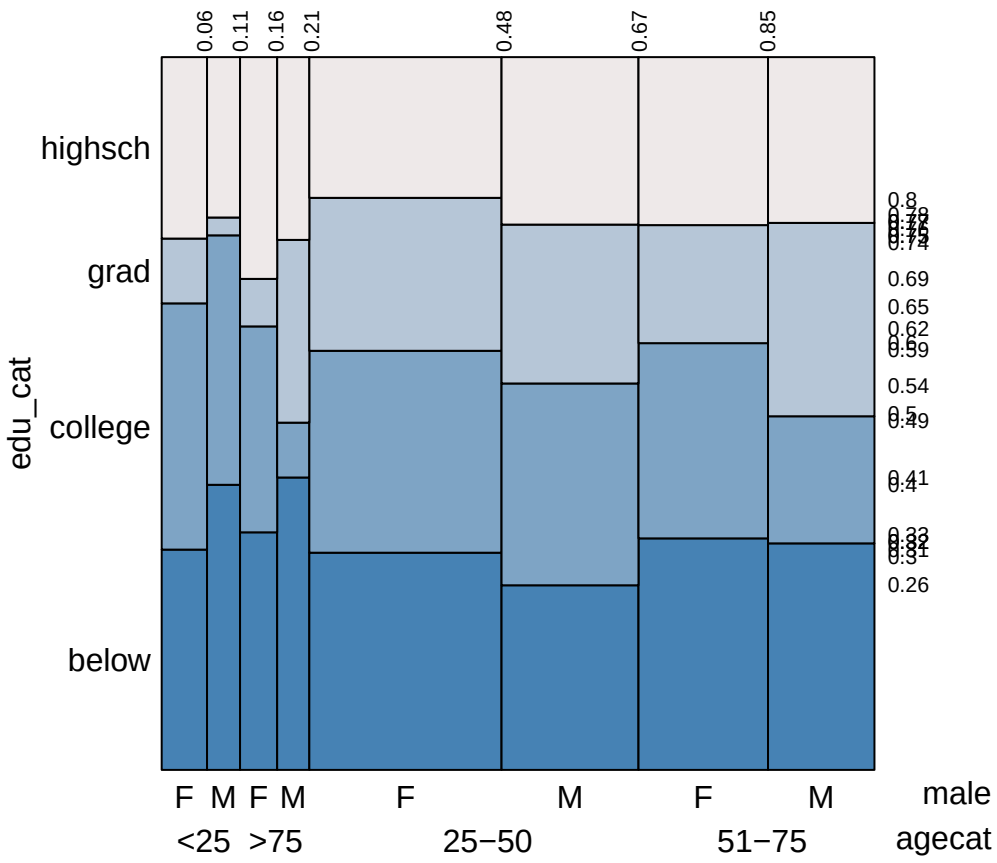
```
## Loading required package: rJava
```

```
## Warning: package 'rJava' was built under R version 4.0.3
```

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

# change factor features to reasonable names

ind = data$male == 1
data$male[ind] = "M"
data$male[!ind] = "F"
data$agecat = ceiling(data$ageyrs/25 )
agecat = c("<25","25-50","51-75",">75")

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)
```



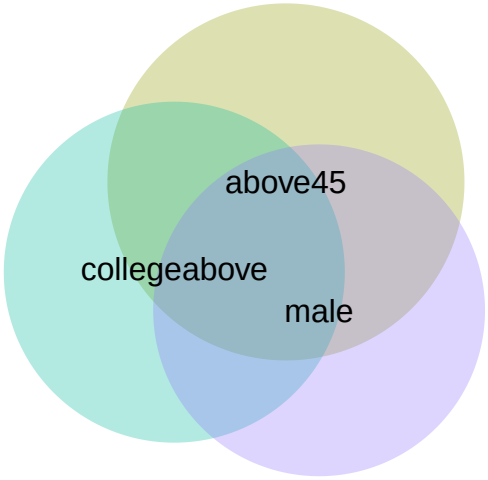

```
eikos(edu_cat~ male+agecat ,data=data)
```




```
# look at intersection

# note surface of above 45 should be approximately half of surface of total population

collegeabove = which( (data$edu_cat == "college") + (data$edu_cat == "grad") ==1 )
collegeabove.names = rep("collegeabove", length(collegeabove ))

white= which( data$race_cat == "White" )
white.names = rep("White", length(white))

median(data$ageyrs)
```

```
## [1] 46
```

```
above45 = which(data$ageyrs>45)
above45.names= rep("above45", length(above45))

male = which(data$male == "M")
male.names = rep("male", length(male))

female = which(data$male == "F")
female.names = rep("female", length(female))

subjectinfo = c(above45, collegeabove, male)
names = c(above45.names , collegeabove.names, male.names)
ven = venneuler(data.frame(elements = subjectinfo, sets=names))
plot(ven)
```




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

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.4
```

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

```
## Loaded glmnet 4.1-1
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.0.4
```

```
## Loading required package: carData
```

```
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)
```

```r
data$male = factor(data$male)
data$smokenow= factor(data$smokenow)

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

```
## [1] 0.699245
```

correlation between features

```r
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
```

```r
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
```

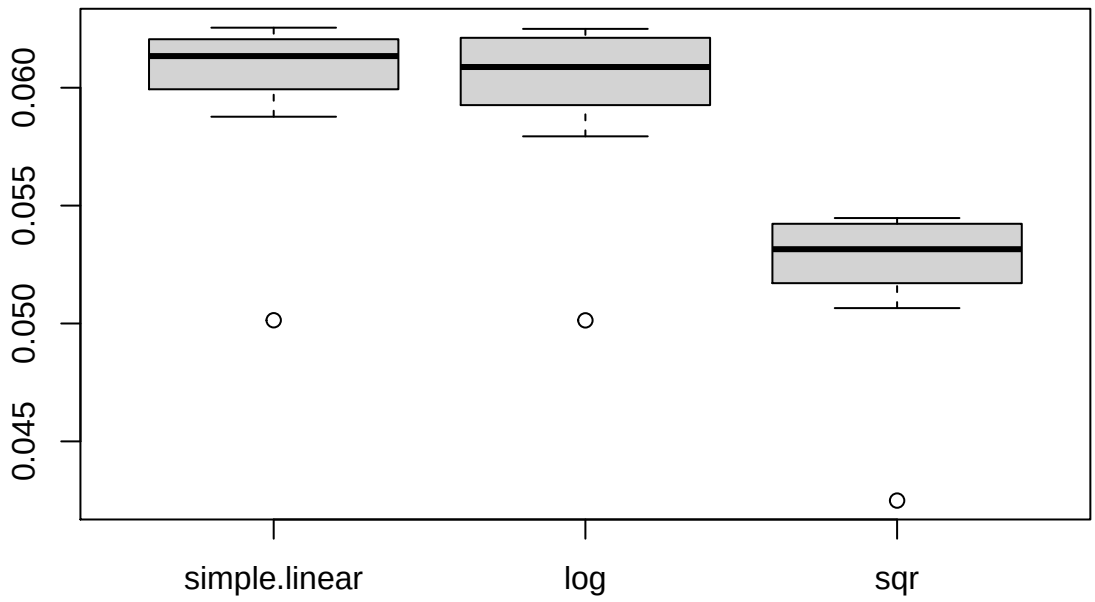
```
t2=colnames( model$model)

setdiff(t1,t2)
```

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

does one feature explain the 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[c,1] = mean(model$residuals^2)
  if(! is.factor(Xfull[,c])){
    modelpower2 = lm(data$length~poly( Xfull[,c], 2))
    modellog = lm(log(data$length)~ Xfull[,c])
    res[c,2] = mean(modelpower2$residuals^2)
    res[c,3] = mean(modellog$residuals^2)
  }
  #res[c,3] = mean(modelpower2$residuals^2)
}

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

box = list(simple.linear=removezero(res[,1]), log=removezero(res[,2]), sqr=removezero(res[,3]) )

boxplot(box)
```




```
which.min(removezero(res[,1]))
```

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

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

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

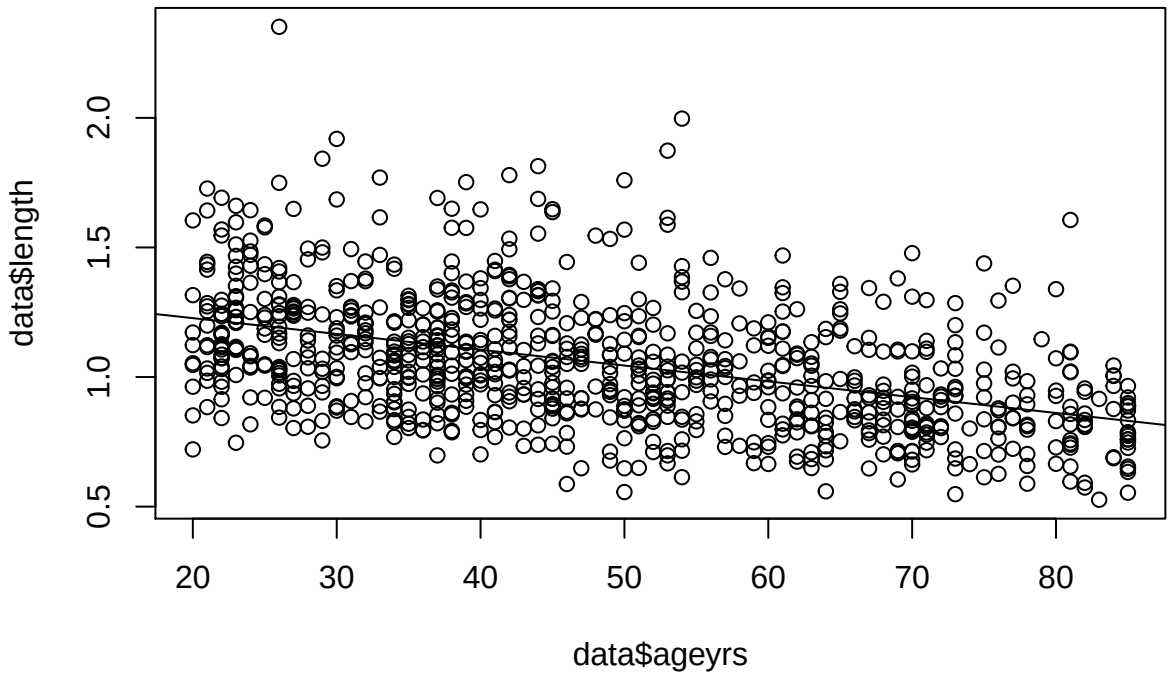
```r
which.min(removezero(res[,3]))
```

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

```r
colnames(Xfull)[30]
```

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

```r
simplelinear = lm(length~ageyrs, data=data)
plot(data$ageyrs, data$length)
abline(simplelinear$coefficients)
```



seems there is a linear relationship but looks insufficient.

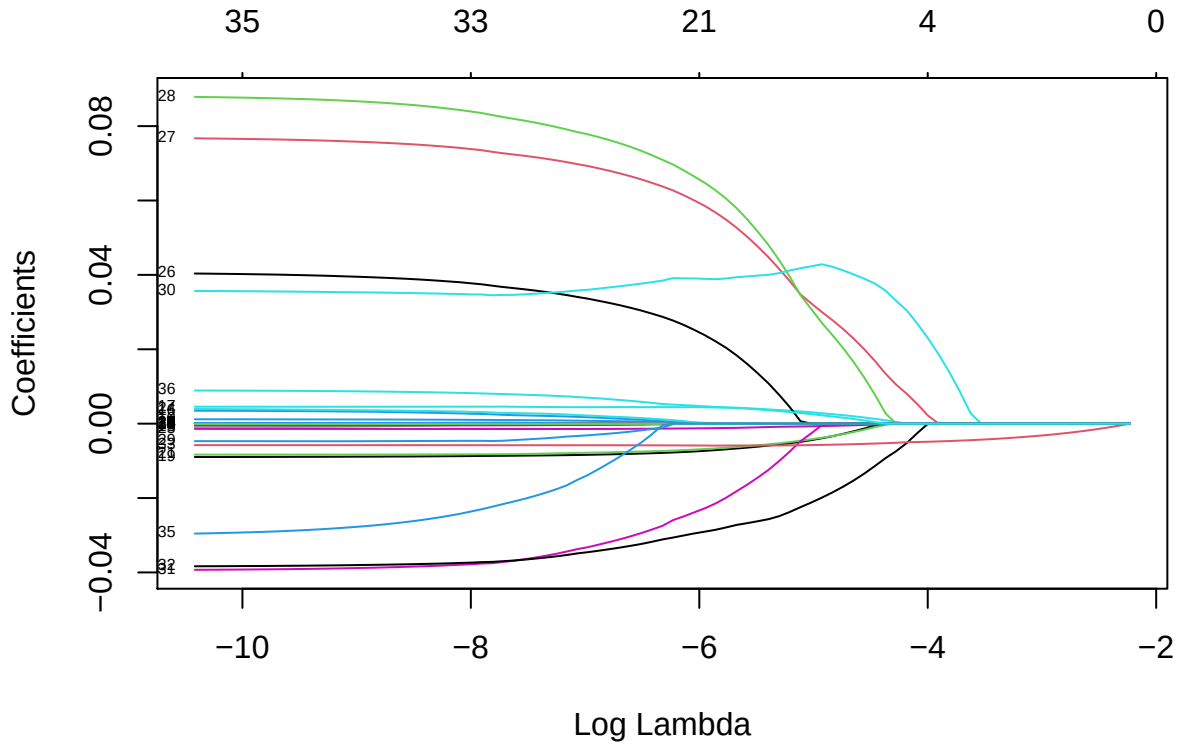
Also seems sqr or log does not do exponentially better here

what about 2 features

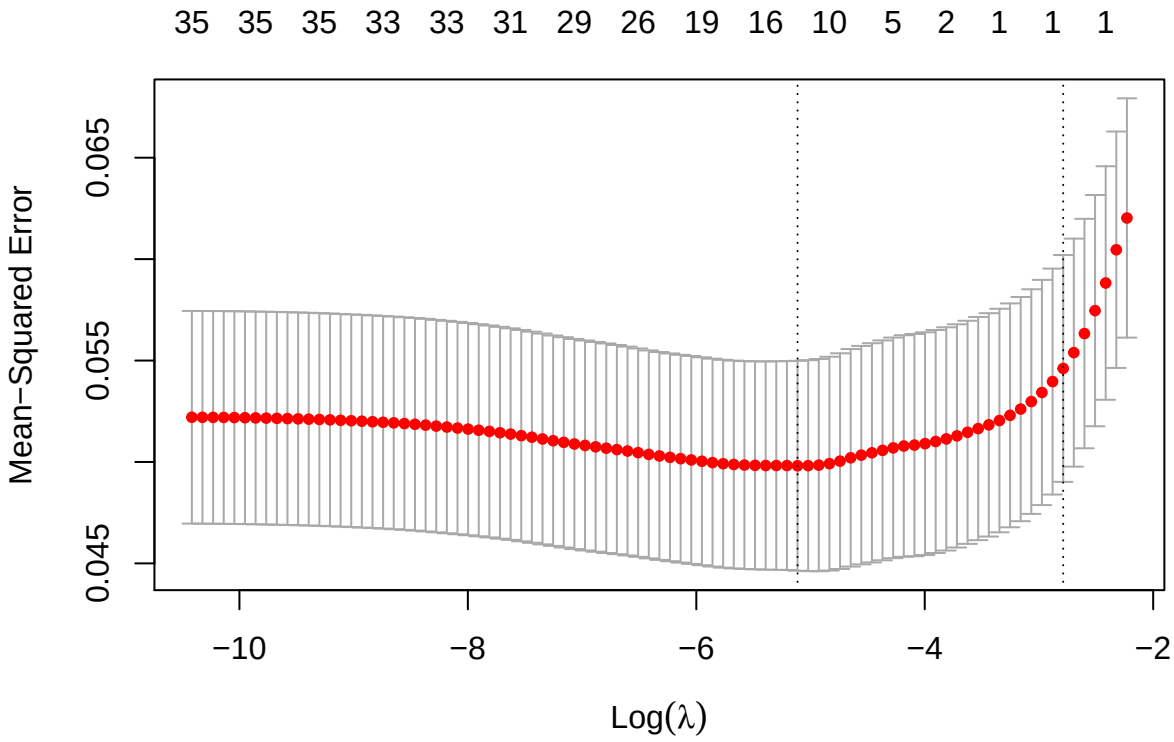
best model based on lasso/ridge

```r
### 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)
```



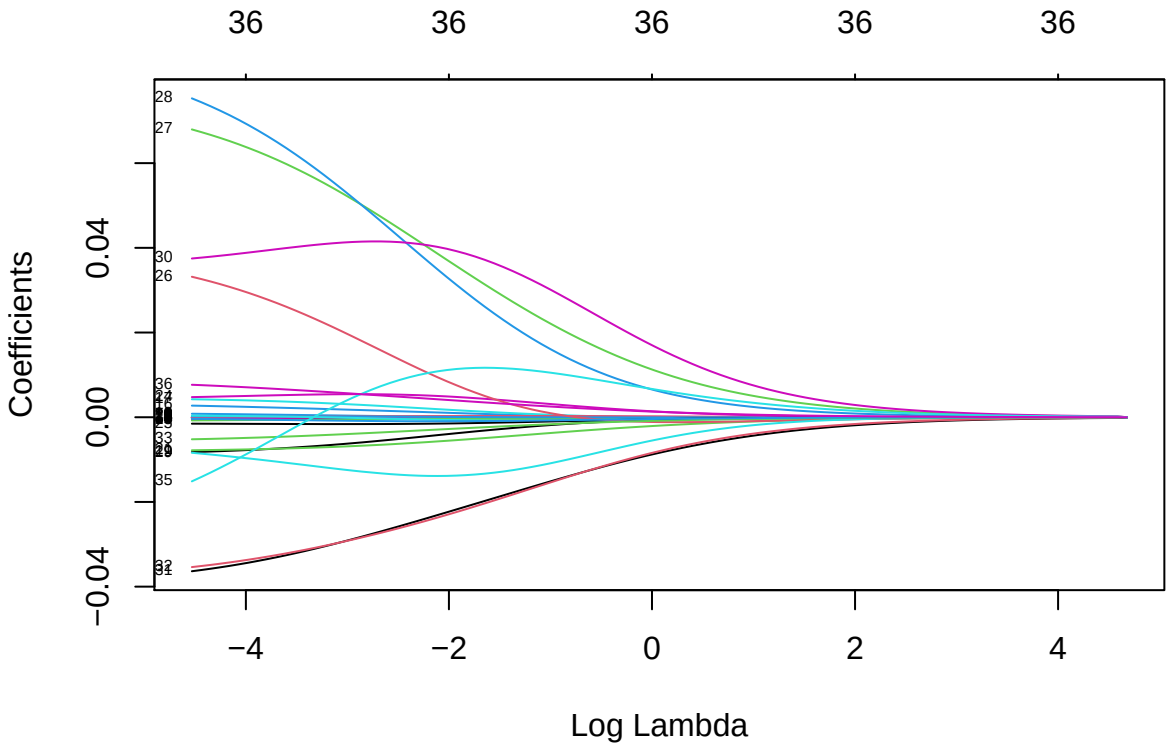

```
## estimated betas for minimum lambda
coef(cvfit_lasso, s = "lambda.min")
```

```
## 37 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)      1.4025210600
## POP_PCB1         .
## POP_PCB2         .
## POP_PCB3         .
## POP_PCB4         .
## POP_PCB5         .
## POP_PCB6         .
## POP_PCB7         .
## POP_PCB8         .
## POP_PCB9         .
## POP_PCB10        .
## POP_PCB11        .
## POP_dioxin1      .
## POP_dioxin2      .
## POP_dioxin3      .
## POP_furan1       .
## POP_furan2       .
## POP_furan3       0.0032938563
## POP_furan4       .
## whitecell_count -0.0048822505
## lymphocyte_pct   .
```

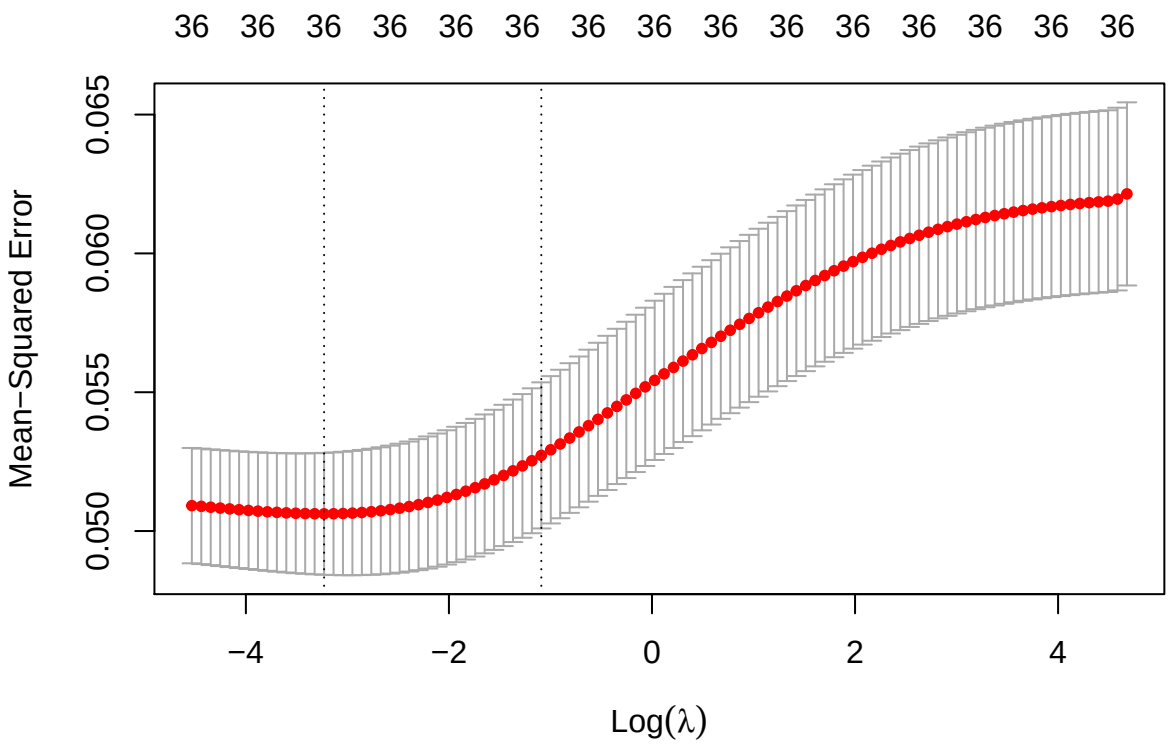
```
## monocyte_pct    -0.0045885563
## eosinophils_pct  .
## basophils_pct    .
## neutrophils_pct  .
## BMI             -0.0007588356
## edu_cat2         0.0006460338
## edu_cat3         0.0347277846
## edu_cat4         0.0345318012
## race_cat2        .
## race_cat3        0.0417894242
## race_cat4       -0.0047329660
## male1           -0.0224455671
## ageyrs          -0.0058031615
## yrssmoke         .
## smokenow1        .
## ln_lbxcot        0.0028591081
```

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

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

```r
## 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)      1.398419e+00
## POP_PCB1        -3.167862e-07
## POP_PCB2        -1.881041e-07
## POP_PCB3         1.175129e-06
## POP_PCB4        -3.170587e-08
## POP_PCB5        -3.325161e-08
## POP_PCB6         9.761732e-08
## POP_PCB7        -5.783142e-07
## POP_PCB8        -4.333730e-07
## POP_PCB9         9.992878e-08
## POP_PCB10        4.588160e-04
## POP_PCB11        6.565258e-05
## POP_dioxin1     -9.777443e-05
## POP_dioxin2     -3.212980e-04
## POP_dioxin3     -1.000259e-05
## POP_furan1      -5.382392e-04
## POP_furan2       1.958387e-03
## POP_furan3       3.337740e-03
## POP_furan4      -6.422250e-05
## whitecell_count -6.669859e-03
## lymphocyte_pct   1.822139e-04
```

```
## monocyte_pct    -7.091461e-03
## eosinophils_pct  1.796237e-04
## basophils_pct   -1.583838e-05
## neutrophils_pct  5.394501e-03
## BMI             -1.612589e-03
## edu_cat2         2.224745e-02
## edu_cat3         5.528497e-02
## edu_cat4         5.718504e-02
## race_cat2       -1.182497e-02
## race_cat3        4.090029e-02
## race_cat4       -3.051308e-02
## male1           -3.038560e-02
## ageyrs          -4.210199e-03
## yrssmoke        -7.673946e-04
## smokenow1        1.321777e-03
## ln_lbxcot        6.010704e-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
```

```
MSPE_lasso
```

```
## [1] 0.05069399
```

```
MSPE_ridge
```

```
## [1] 0.05283856
```

```
MSPE_step
```

```
## [1] 0.05387623
```

say we try to fit with only 2 features

we first see if lasso choose the same simple linear model

```
# 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
```

```
library("plot3D")
```

```
## Warning: package 'plot3D' was built under R version 4.0.4
```

```
# 2 feature lasso choose
min(which((M_lasso$lambda)<=exp( -3.8)))
```

```
## [1] 18
```

```
coefs = M_lasso$beta[,18]
choosen=which(coefs!=0)
coefs[choosen]
```

```
##    race_cat3       ageyrs
##  0.013839164 -0.004676005
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.4
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.3
```

```
## Warning: package 'tibble' was built under R version 4.0.4
```

```
## Warning: package 'tidyr' was built under R version 4.0.4
```

```
## Warning: package 'readr' was built under R version 4.0.4
```

```
## Warning: package 'dplyr' was built under R version 4.0.4
```

```
## Warning: package 'forcats' was built under R version 4.0.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::recode() masks car::recode()
## x purrr::some()   masks car::some()
## x tidyr::unpack() masks Matrix::unpack()
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.4
```

```
## Loading required package: lattice
```

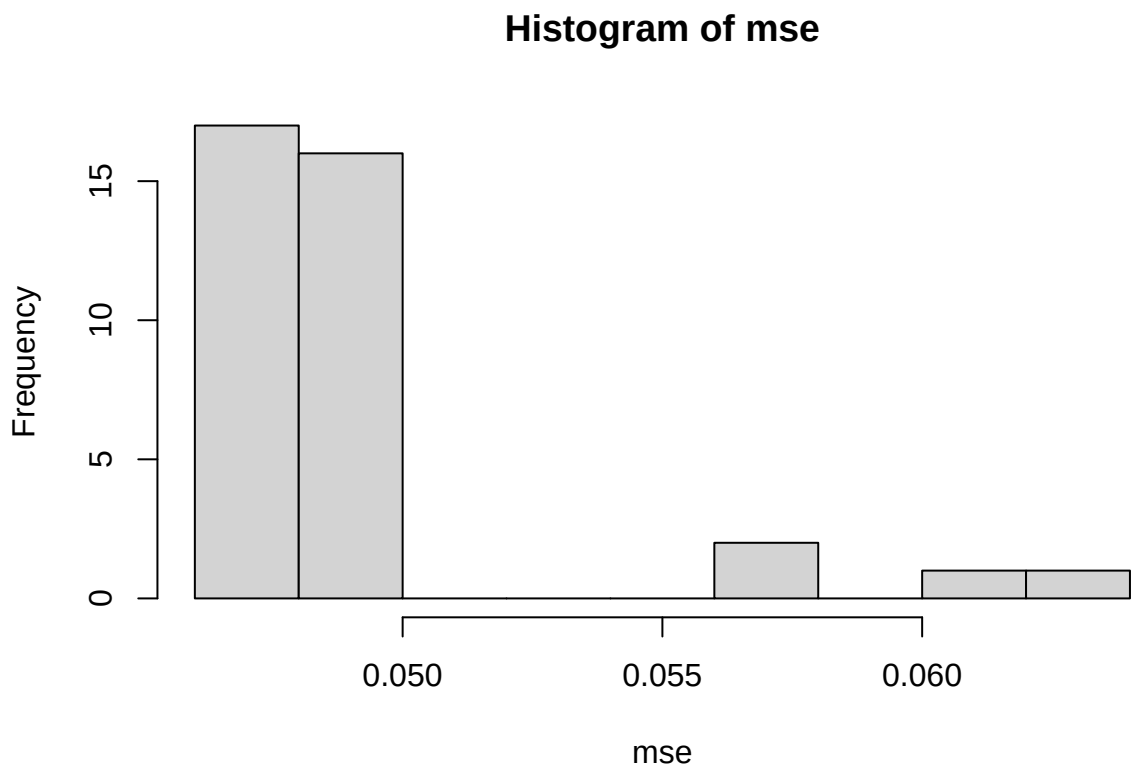
```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.0.4
```

```
models= regsubsets(length~., data=data, nvmax=2)
# rss of all 2 feature model, we see no magical model
mse = models$rss/nrow(data)
hist(mse)
```




```
library("loon")
```

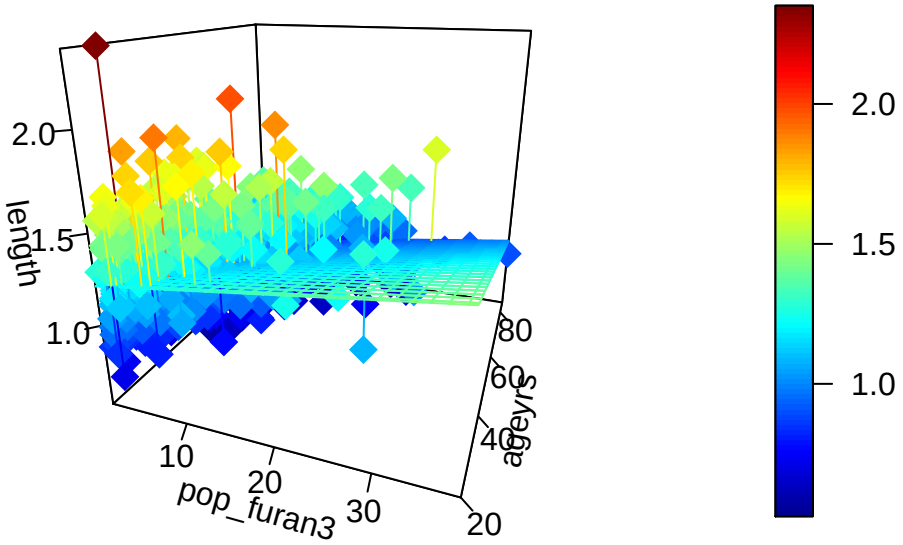
```
## Warning: package 'loon' was built under R version 4.0.4
```

```
## Loading required package: tcltk
```

```
## loon Version 1.3.4.
## To learn more about loon, see l_web().
```

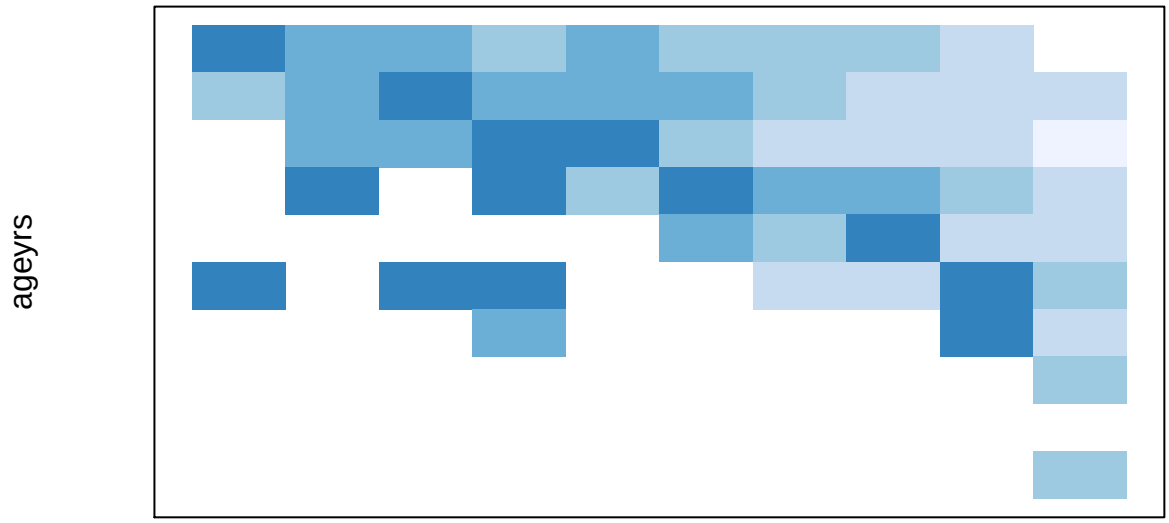
```
z=data$length
y=data$ageyrs
x=data$POP_furan3

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")
```

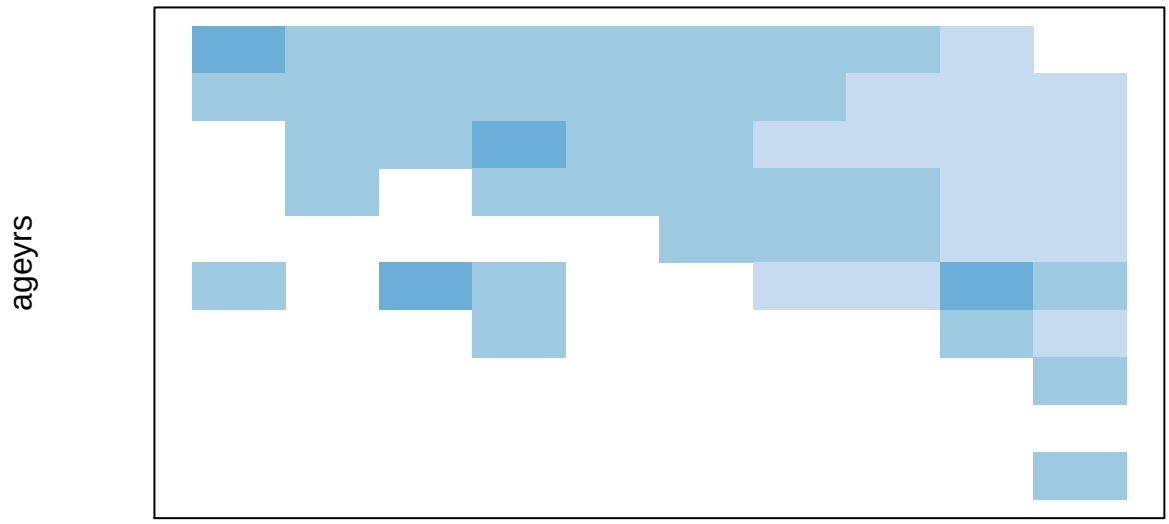



```
#turn ageyrs and pop_furan into grids

miny=min(y)
intervaly = (max(y)-miny)/10
minx=min(x)
intervalx = (max(x)-minx)/10

xy = matrix(0, nrow = 10, ncol = 10)
count = matrix(0, nrow = 10, ncol = 10)
for (i in 1:nrow(data)){
  xgrid = (x[i]-minx)/intervalx
  ygrid = (y[i]-miny)/intervaly
  count[xgrid,ygrid] = 1 + count[xgrid,ygrid]
```

```
  xy[xgrid,ygrid] = xy[xgrid, ygrid] + z[i]
}
xygrid = xy/count
col_areas(xygrid,xlab="pop_furan3", ylab="ageyrs")
```



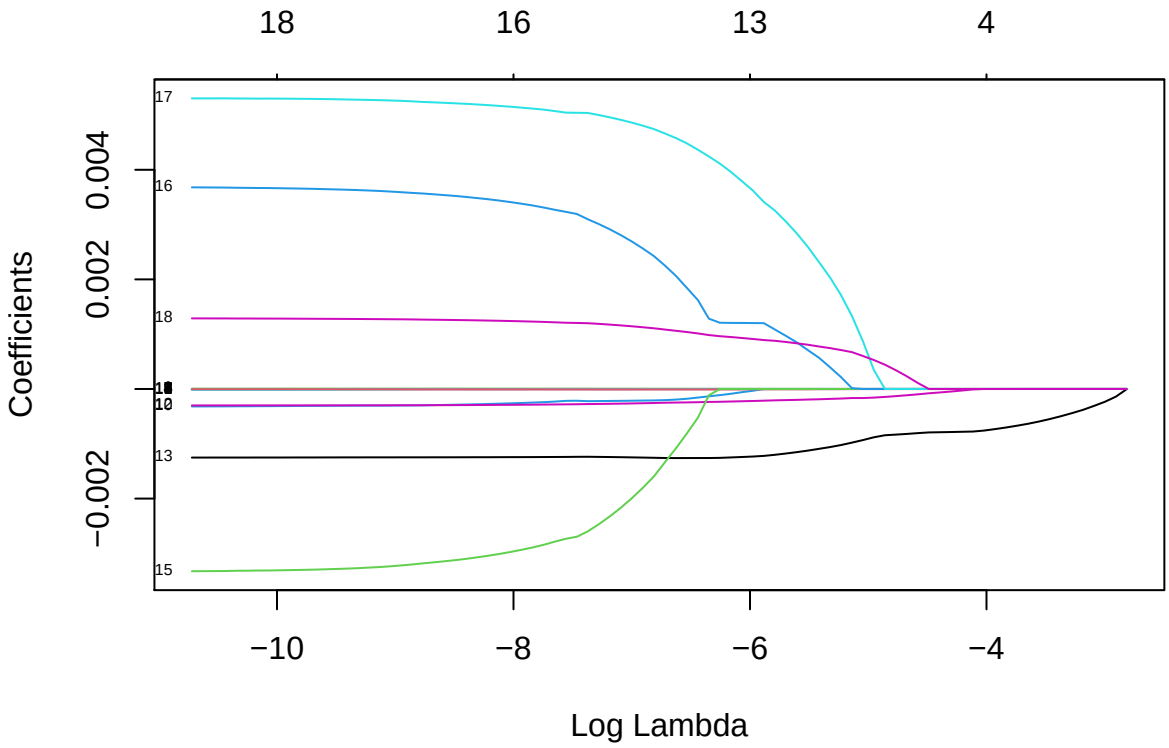

```
maxz=max(z)
minz=min(z)
breaks = seq( minz, maxz, by=(maxz-minz)/5 )
col_areas(xygrid,xlab="pop_furan3", ylab="ageyrs", breaks = breaks)
```



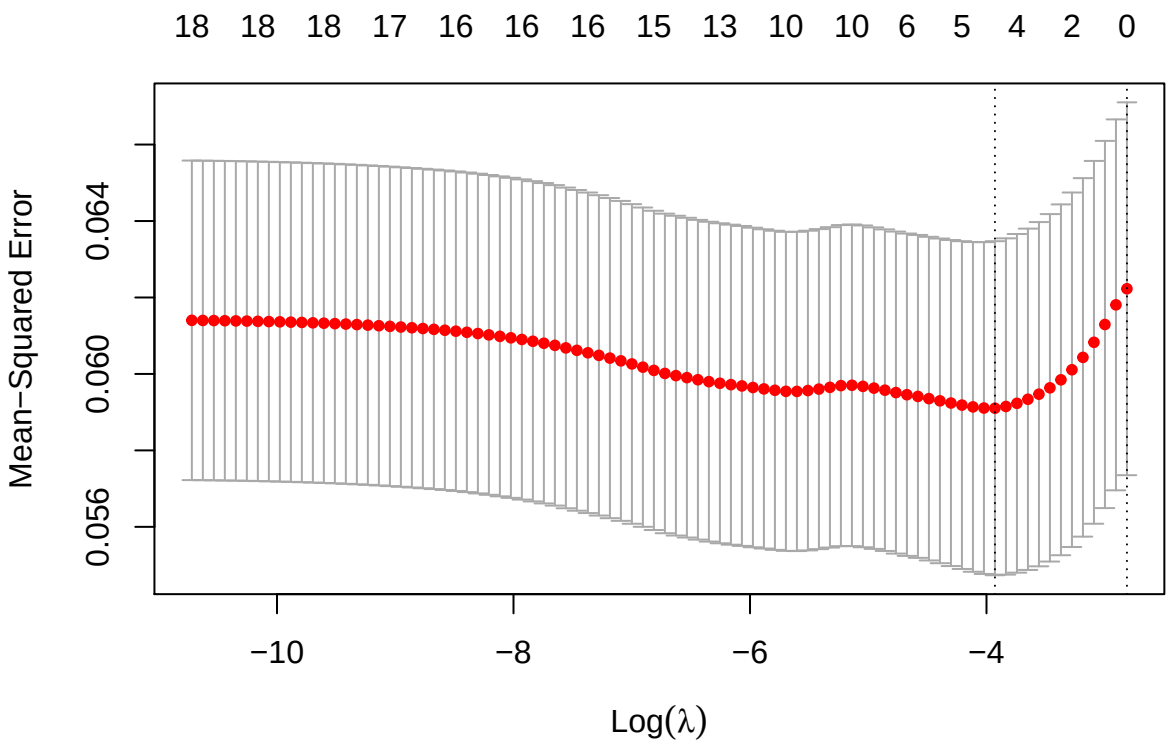

```
# anyway how does this compare to the best fit?
```

4/4

```
cols = colnames(data)
bios.ind = str_detect(cols, "POP")
bios = cols[bios.ind]
expr = paste("length~", paste(bios, collapse = "+"))
M = model.matrix(lm(expr, data=data))
y_train = data$length[1:700]
X_train = M[1:700,(1:ncol(M))[bios.ind]]
y_test= data$length[701:nTotal]
X_test= M[701:nTotal,(1:ncol(M))[bios.ind]]


M_bios.lasso <- glmnet(x=X_train,y=y_train,alpha = 1)
####

####
## plot paths
plot(M_bios.lasso,xvar = "lambda",label=TRUE)
```




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

## plot MSPEs by lambda
plot(cvfit_bios.lasso)
```




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

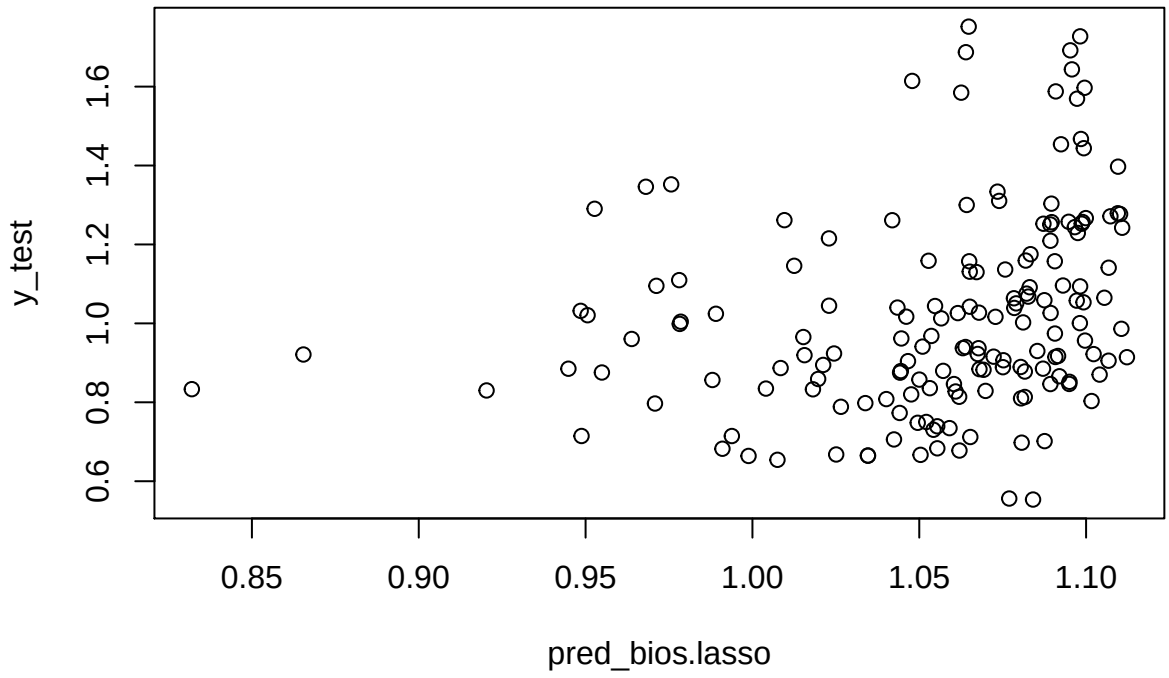
```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)  1.117266e+00
## POP_PCB1    -3.168200e-07
## POP_PCB2     .
## POP_PCB3     .
## POP_PCB4     .
## POP_PCB5     .
## POP_PCB6     .
## POP_PCB7    -2.267866e-07
## POP_PCB8    -5.895032e-07
## POP_PCB9     .
## POP_PCB10    .
## POP_PCB11    .
## POP_dioxin1  .
## POP_dioxin2 -7.323191e-04
## POP_dioxin3  .
## POP_furan1   .
## POP_furan2   .
## POP_furan3   .
## POP_furan4   .
```

```
## predictions
pred_bios.lasso <- predict(cvfit_bios.lasso,newx=X_test,  s="lambda.min")
```

```
## MSPE in test set
MSPE_bios.lasso <- mean((pred_bios.lasso-y_test)^2)
MSPE_bios.lasso
```

```
## [1] 0.06028211
```

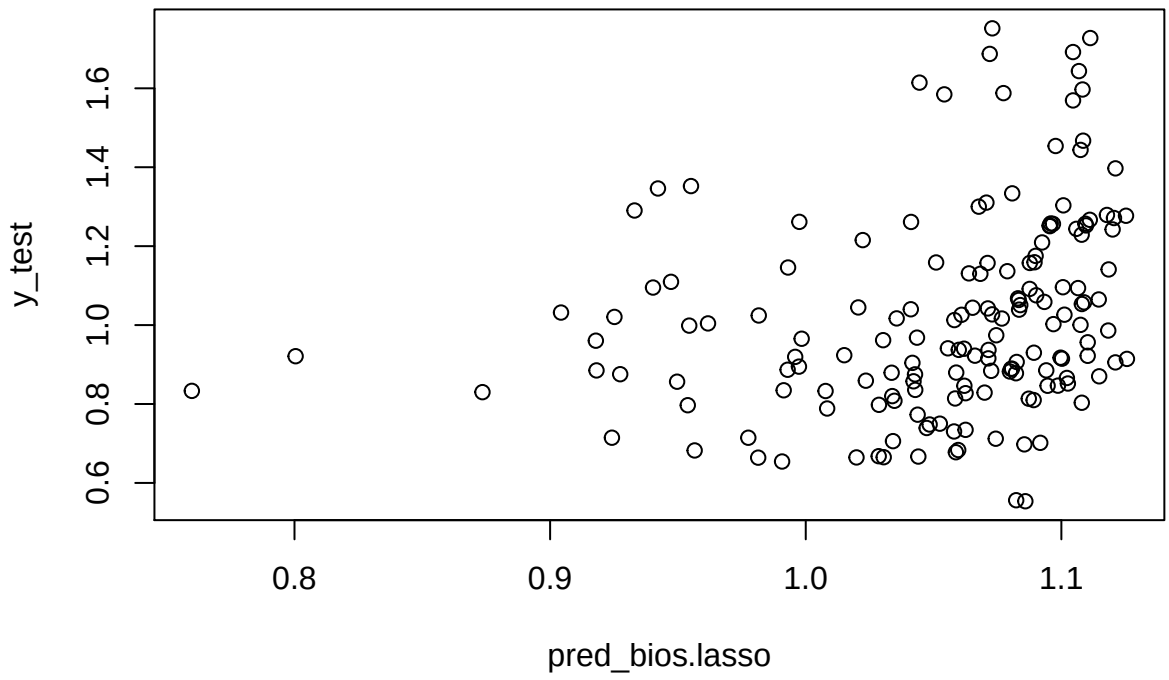
```
plot(pred_bios.lasso, y_test)
```




```
tempfunction =function(X){
  newdata = X
  M = model.matrix(lm(expr, data=newdata))
  y_train = data$length[1:700]
  X_train = M[1:700,(1:ncol(M))[bios.ind]]
  y_test= data$length[701:nTotal]
  X_test= M[701:nTotal,(1:ncol(M))[bios.ind]]


  M_bios.lasso <- glmnet(x=X_train,y=y_train,alpha = 1)
  ####

  ####
  ## plot paths

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

```
  ## plot MSPEs by lambda

  ## estimated betas for minimum lambda
  print( coef(cvfit_bios.lasso, s = "lambda.min"))

  ## predictions
  pred_bios.lasso <- predict(cvfit_bios.lasso,newx=X_test,  s="lambda.min")

  ## MSPE in test set
  MSPE_bios.lasso <- mean((pred_bios.lasso-y_test)^2)
  print( MSPE_bios.lasso)

  plot(pred_bios.lasso, y_test)

}



tempfunction(data)
```

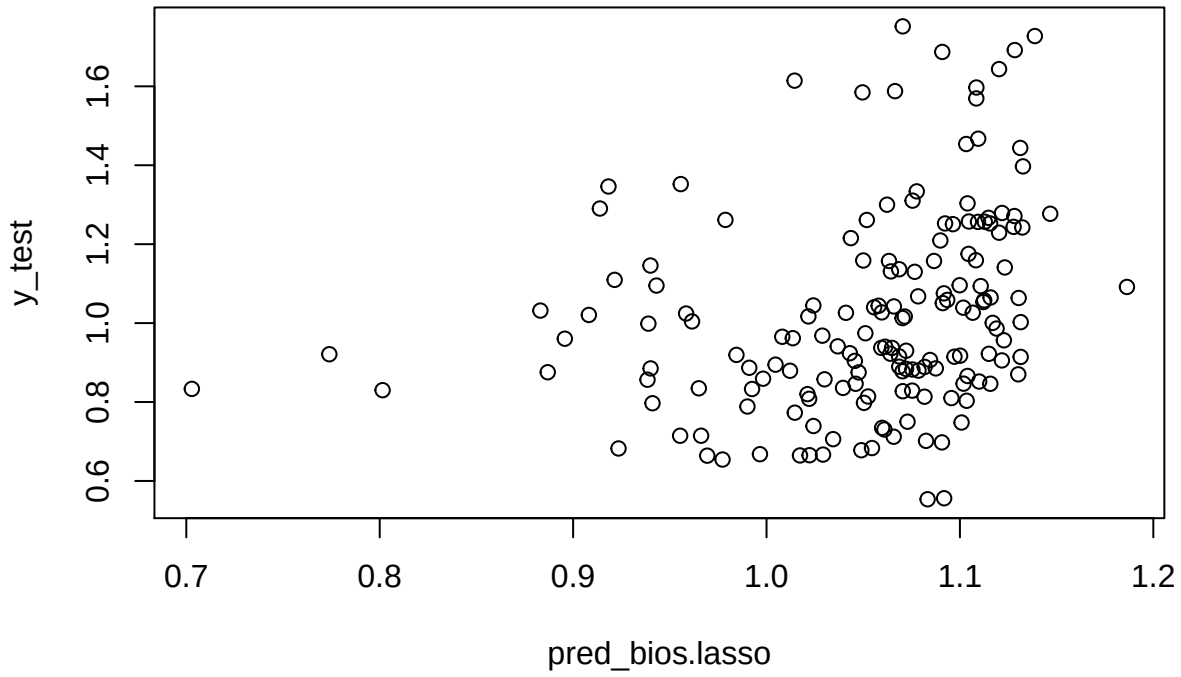
```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  1.133409e+00
## POP_PCB1    -1.724908e-07
## POP_PCB2     .
## POP_PCB3     .
## POP_PCB4     .
## POP_PCB5     .
## POP_PCB6     .
## POP_PCB7    -5.078968e-07
## POP_PCB8    -1.581107e-06
## POP_PCB9     .
## POP_PCB10    .
## POP_PCB11    .
## POP_dioxin1 -1.142741e-04
## POP_dioxin2 -8.197710e-04
## POP_dioxin3  .
## POP_furan1   .
## POP_furan2   .
## POP_furan3   .
## POP_furan4   2.306743e-04
## [1] 0.05981737
```




```
#########################
X=data
for (i in 1:ncol(M)){
  X[,i] = X[,i] / sd(X[,i])
}

tempfunction(X)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  1.131950230
## POP_PCB1    -0.021392712
## POP_PCB2     .
## POP_PCB3     0.013723297
## POP_PCB4     .
## POP_PCB5     .
## POP_PCB6     .
## POP_PCB7    -0.018910426
## POP_PCB8    -0.012977238
## POP_PCB9     .
## POP_PCB10    .
## POP_PCB11    .
## POP_dioxin1 -0.011522671
## POP_dioxin2 -0.047108495
## POP_dioxin3 -0.002478052
## POP_furan1   .
```

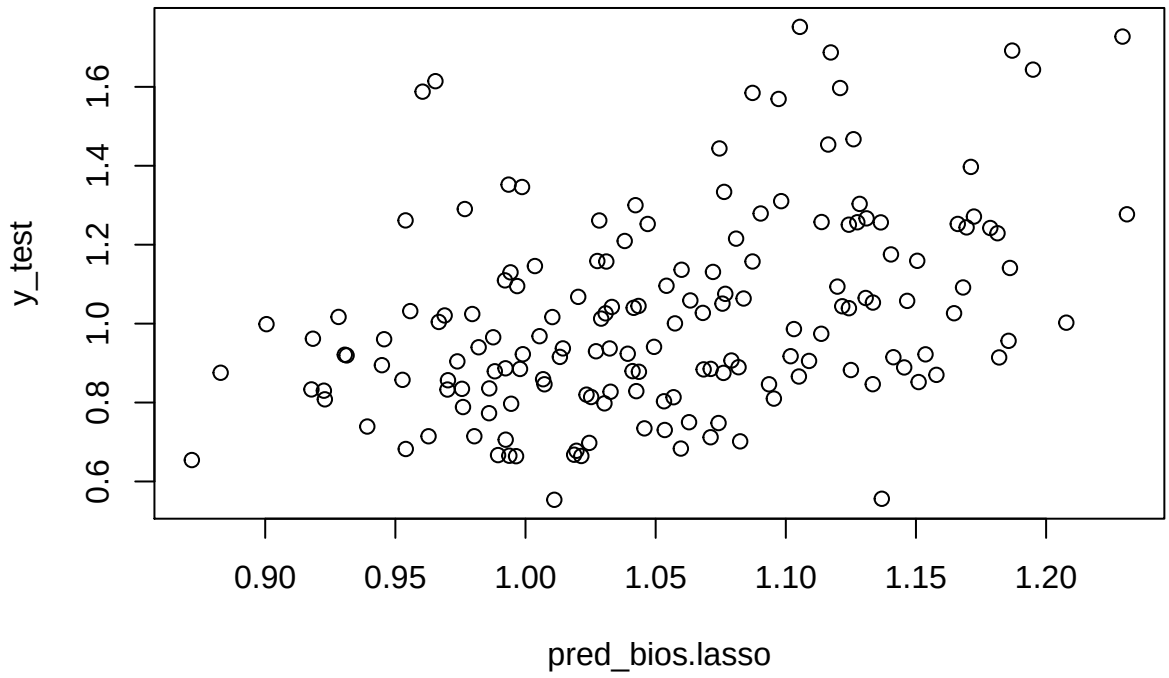
```
## POP_furan2   0.004049089
## POP_furan3   0.017679663
## POP_furan4   0.009517661
## [1] 0.0593056
```




```
#########################

newX=X
for (i in 2:ncol(M)){
  newX[,i] = log( newX[,i])
}
tempfunction(newX)
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  1.030828931
## POP_PCB1    -0.022310351
## POP_PCB2     .
## POP_PCB3     0.026367858
## POP_PCB4     .
## POP_PCB5     .
## POP_PCB6     .
## POP_PCB7    -0.012316001
## POP_PCB8    -0.055164031
## POP_PCB9     .
## POP_PCB10    .
## POP_PCB11    0.006592011
```

```
## POP_dioxin1 -0.022588973
## POP_dioxin2 -0.012560344
## POP_dioxin3 -0.023501168
## POP_furan1   .
## POP_furan2   .
## POP_furan3   0.007483030
## POP_furan4   0.054173286
## [1] 0.05491166
```




```
#TODO 4 assumption

#fit pollutants over other (only collect significant results)

#reasonable length
```

length~pollutents

good model: newvariable: predicted length (by pollutents) $\hat{y}$ not $\hat{y} + \sigma$

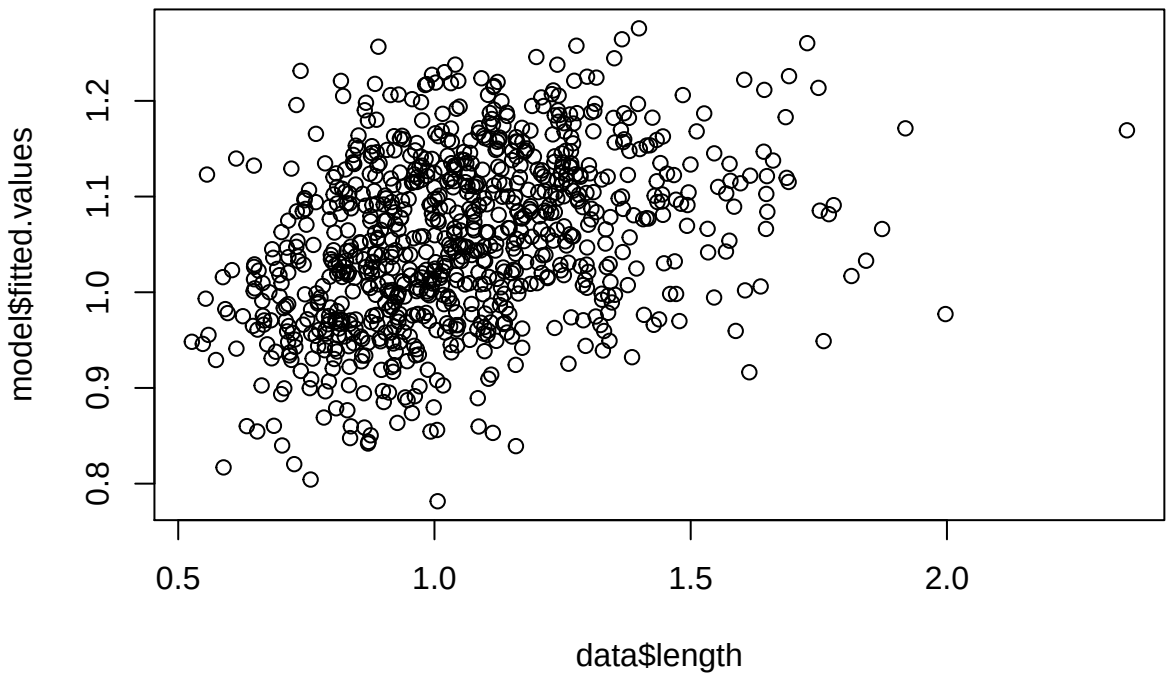
newvariable~bio/othercovariates

```
newX=data
for (i in 2:ncol(M)){
  newX[,i] = log( newX[,i])
}


  M = model.matrix(lm(expr, data=newX))
```

```r
y_train = data$length[1:700]
X_train = M[1:700,(1:ncol(M))[bios.ind]]
y_test= data$length[701:nTotal]
X_test= M[701:nTotal,(1:ncol(M))[bios.ind]]


M_bios.lasso <- glmnet(x=X_train,y=y_train,alpha = 1)
####

####
## plot paths

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

## plot MSPEs by lambda

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

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  1.758627936
## POP_PCB1    -0.020517772
## POP_PCB2     .
## POP_PCB3     0.018850756
## POP_PCB4     .
## POP_PCB5     .
## POP_PCB6     .
## POP_PCB7    -0.005922628
## POP_PCB8    -0.055169466
## POP_PCB9     .
## POP_PCB10    .
## POP_PCB11    0.004754139
## POP_dioxin1 -0.020809235
## POP_dioxin2 -0.011273053
## POP_dioxin3 -0.022857655
## POP_furan1   .
## POP_furan2   .
## POP_furan3   0.004661201
## POP_furan4   0.052762883
```

```r
ind =  which(coef(cvfit_bios.lasso, s = "lambda.min")!=0)

newdata = data[,ind]
newdata[,2:ncol(data)] = log(newdata[,2:ncol(newdata)])

model = lm(length~., data=newdata)

s =summary(model)

plot(data$length, model$fitted.values)
```




```
#### if you want to use the lasso model directly, just use
cvfit_bios.lasso
```

```
##
## Call:  cv.glmnet(x = X_train, y = y_train, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.00373    32 0.05695 0.004770      10
## 1se 0.06074     2 0.06165 0.004693       1
```