Cross-Validatded Elastic-Net

The purpose of this script is to use an elsatic-net model with cross-validated test error, alpha values, and lambda values. This will be used to predict life expectancy using a cut of the HRS data set.

In [3]:
```
# Show all columns.
options(repr.matrix.max.cols=150, repr.matrix.max.rows=200)

df = read.csv('le_11142.csv')
dim(df)
head(df)
```

1. 13132
2. 54

hhidpn	wave	mstat	cendiv	gender	rahispan	racem	iwbeg	dage_m	dage_y	raedy
3010	3	1.married	9.pacific	1.male	0.not hispanic	1.white/caucasian	13345	931	77	
3010	6	1.married	9.pacific	1.male	0.not hispanic	1.white/caucasian	15445	931	77	
3010	7	1.married	9.pacific	1.male	0.not hispanic	1.white/caucasian	16267	931	77	
3010	8	1.married	9.pacific	1.male	0.not hispanic	1.white/caucasian	16875	931	77	
3010	9	1.married	9.pacific	1.male	0.not hispanic	1.white/caucasian	17577	931	77	
3010	10	1.married	9.pacific	1.male	0.not hispanic	1.white/caucasian	18520	931	77	

Create test and training sets.

In [5]:
```
# Specify data as numeric or factor
# Doesn't like dents
df = na.omit(df)
names_factor = c('gender', 'raracem', 'rahispan', 'mstat', 'shlt',
                 'depres', 'cendiv' , 'effort',
                 'sleepr', 'arthr','heart', 'strok', 'psych', 'cancr', 'wave',
                  'slfmem', 'covs',
                  'lbrf', 'smokev',  'hiltc',
                  'spcfac', 'rarelig','ravetrn')
df[,names_factor] = lapply(df[,names_factor], factor)
names_numeric = c('iearn', 'logiearn', 'bmi', 'cogtot','drinkn',
```

```
                    'dage_y','conde', 'raedyrs',
                   'logisret', 'loghatotb','loghspti')
df[,names_numeric] = lapply(df[,names_numeric], as.numeric)
dim(df)
```

1. 13132
2. 54

In [6]:
```
#This is the weak model.
#Create tests and training sets.
n = dim(df)[1]
training_ratio = .8
train_size = sort(sample(1:n, training_ratio*n))
train = na.omit(df[train_size,])
test = na.omit(df[-train_size,])
# We can also test for death age at month.
# Interact puff*puffpos, shlt*shltc
train_mat = model.matrix(dage_y ~ cogtot+ gender + raracem + rahispan + agey_m +logiea
                         shlt + shltc +raedyrs +mstat +
                        bmi +smoken + hibp + drinkn + pstmem +
                         + depres + cendiv + effort + bmi*bmi
                        +  sleepr + bmi + smoken + drinkn + arthr + heart + strok + ps
                         + diab  + lung + slfmem + pstmem  + covs
                          + lbrf + raedyrs + smokev + hibp + conde
                          + hiltc  + spcfac + loghspti + rarelig + ravetrn
                          + loghatotb + loghspti , train)
test_mat = model.matrix(dage_y ~ cogtot+ gender + raracem + rahispan + agey_m +logiear
                         shlt + shltc +raedyrs +mstat +
                        bmi +smoken + hibp + drinkn + pstmem +
                         + depres + cendiv + effort + bmi*bmi
                        +  sleepr + bmi + smoken + drinkn + arthr + heart + strok + ps
                         + diab  + lung + slfmem + pstmem  + covs
                          + lbrf + raedyrs + smokev + hibp + conde
                          + hiltc  + spcfac + loghspti + rarelig + ravetrn
                          + loghatotb + loghspti , test)
```

In [7]:
```
# Test to make sure |imensions are square.
dim(train_mat)
dim(test_mat)
dim(train)
dim(test)
sum(is.na(train_mat))
sum(is.na(test_mat))
#typeof(train_mat[1])
#train_mat
```

1. 10505
2. 94

1. 2627
2. 94

1. 10505
2. 54

1. 2627
2. 54

0

0

CV GLMNET

For variable selection.

Here I will use the glmulti() to try every combination of predictors and their pairwise interactions. \ https://www.youtube.com/watch?v=Im293ClFen4&t=342s

In [65]:
```
#help(cv.glmnet)
```

In [66]:
```
#help(predict.cv.glmnet)
```

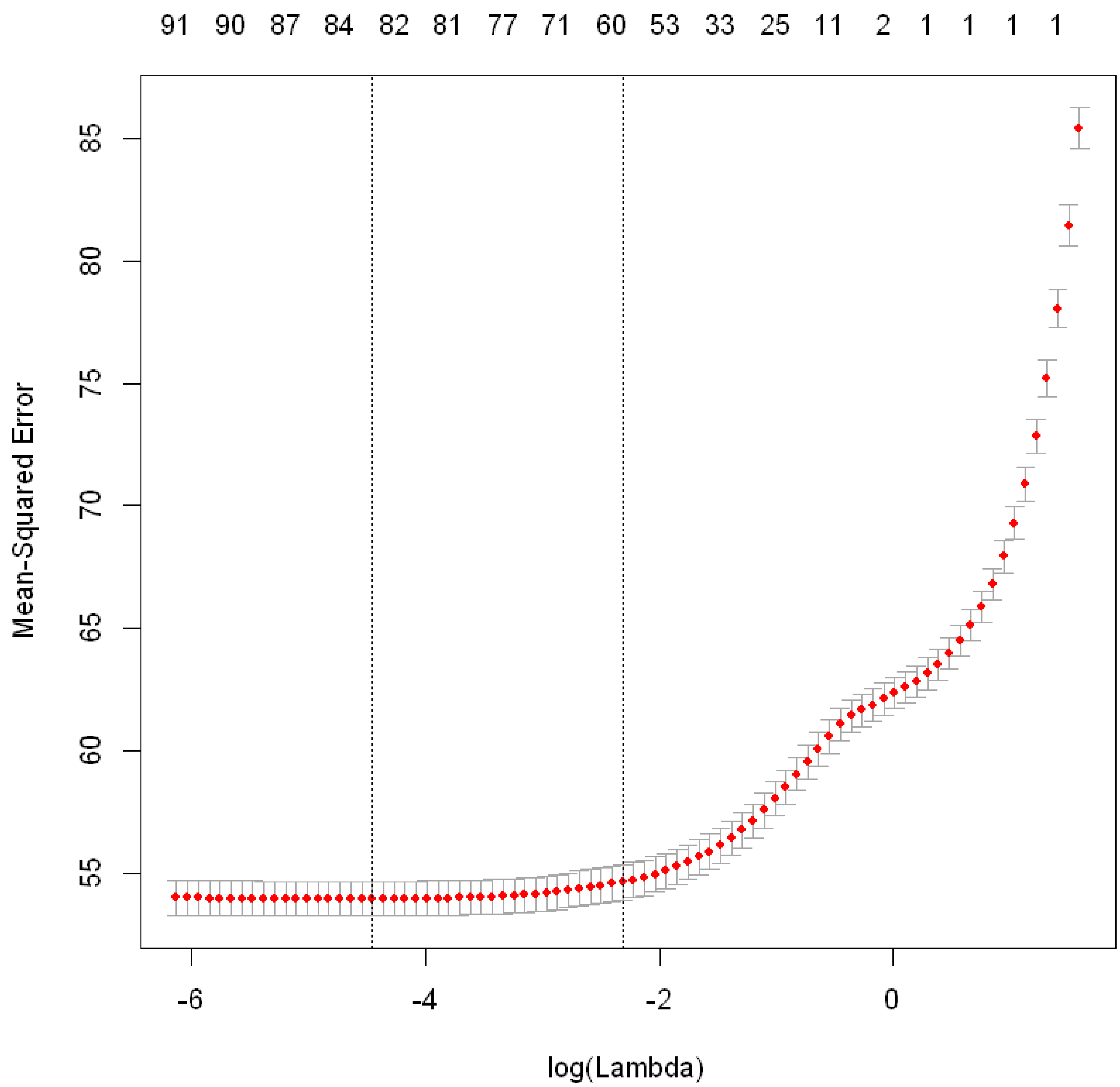
In [67]:
```
'This is one iteration of what the below function carries out.
This iteration is known as a Lasso Regression. This is becaus the alpha is set to 1.
Where the function below iterates through all alphas.
This allows us to test ridge regression(alpha = 0), lasso(alpha = 1), and inbetween at
'
#Store list
list_of_fit = list
a = cv.glmnet(train_mat, train$dage_y,
                      type.measure = 'mse', alpha = 1,
                      family = 'gaussian')

pred = predict(a, s = a$lambda.1se, test_mat, type = 'response')
rmse = sqrt(mean((pred - test$dage_y)^2))
print('RMSE')
rmse
plot(a)
```

'This is one iteration of what the below function carries out.\nThis iteration is known as a Lasso Regression. This is becaus the alpha is set to 1.\nWhere the function below iterates through all alphas.\nThis allows us to test ridge regression(alpha = 0), lasso(alpha = 1), and inbetween at various steps.\n'

```
[1] "RMSE"
```

7.6181678327018



In [68]:
```
lambda(a)
```

```
Error in lambda(a): could not find function "lambda"
Traceback:
```

In [69]:
```
# Elastic Net function.
#load the library and set seed if desired.
library(glmnet)

# alhpa_step represents how the number of steps alpha will test between 0 and 1.
# '.01' for alpha will produce 100 models and '.1' will produce 10.
elastic_net = function(df, train_mat, test_mat, y_var, alpha_step){

    # Create's list of alphas to try.
    alphas = seq(0,1, by = alpha_step)
    #Store list
    list_of_fit = list()

    # Produce models.
    # Optimizes lambda then produces a model for each alhpa.
```

```
    for(i in alphas){
        fit_name = paste0('alpha', i)

        list_of_fit[[fit_name]] = cv.glmnet(train_mat, train$dage_y,
                      type.measure = 'mse', alpha = i,
                      family = 'gaussian')
    }

    # Predict values
    results = data.frame()
    for(i in alphas){
        fit_name = paste0('alpha', i)

        predicted = predict(list_of_fit[[fit_name]],
                            s = list_of_fit[[fit_name]]$lambda.1se,
                            test_mat, type = 'response')

        rmse = sqrt(mean((predicted - test$dage_y)^2))
        mse = mean((predicted - test$dage_y)^2)

        temp = data.frame(alpha = i, rmse = rmse, mse = mse, fit.name = fit_name)
        results = rbind(results, temp)
    }
    return(results)
    }

results = elastic_net(df, train_mat, test_mat, df$dage_y, .01)
```

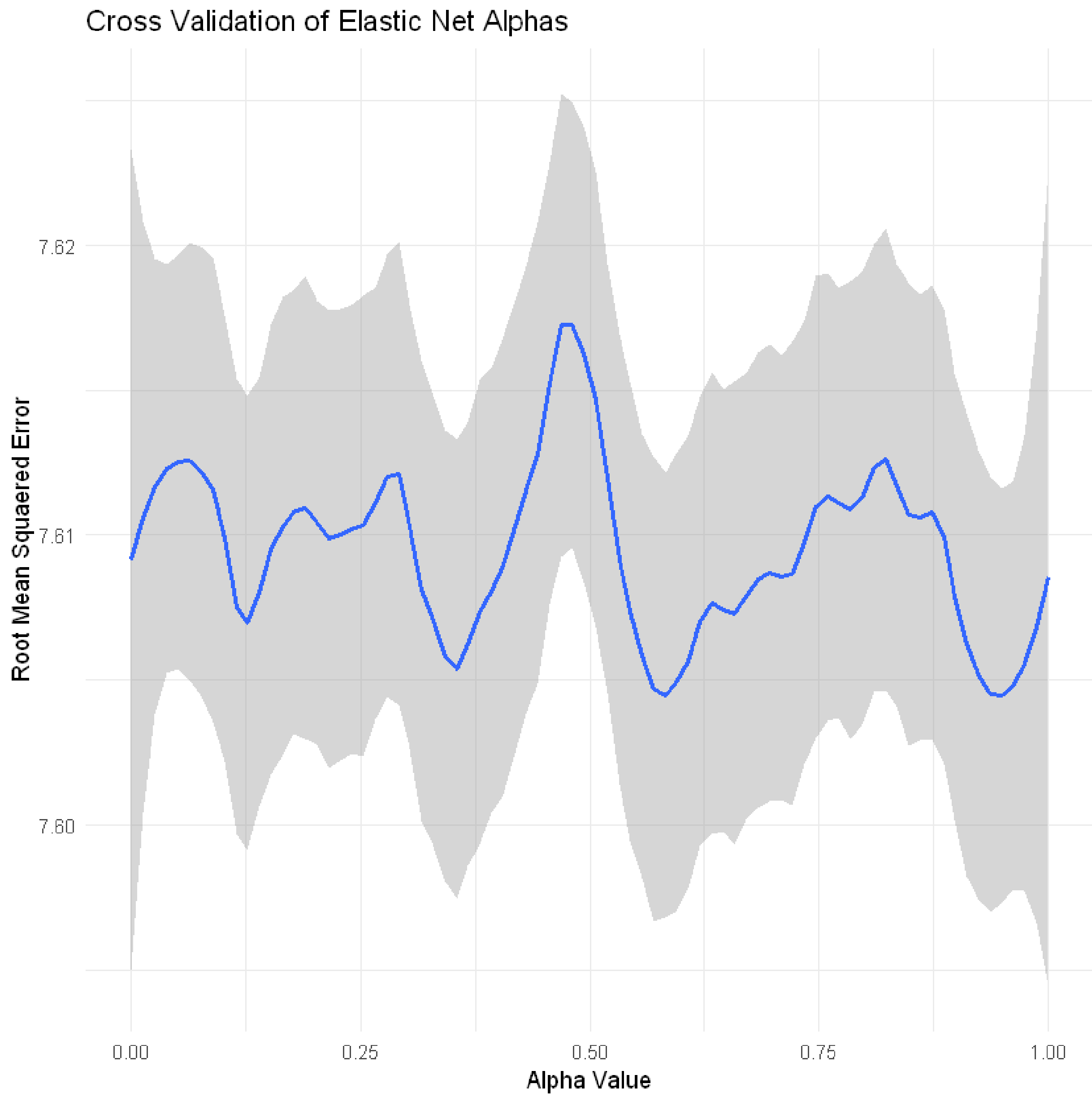
In [70]:
```
# Convert list to dataframe.
# List 10 best models.
results %>% arrange(mse) %>% head(10)
```

alpha	rmse	mse	fit.name
0.60	7.580769	57.46806	alpha0.6
0.56	7.584856	57.53004	alpha0.56
0.35	7.587662	57.57262	alpha0.35
0.69	7.588839	57.59048	alpha0.69
0.41	7.591500	57.63087	alpha0.41
0.16	7.592211	57.64167	alpha0.16
0.10	7.592228	57.64192	alpha0.1
0.95	7.593260	57.65759	alpha0.95
0.94	7.593289	57.65804	alpha0.94
0.88	7.593477	57.66090	alpha0.88

In [71]:
```
x = results$alpha
y = results$mse
y2 = results$rmse
# Below sets up function to scale variable.
scaleFUN <- function(x) sprintf("%.2f", x)
qplot(x,y2, geom = 'smooth', span = .2) +
    ggtitle("Cross Validation of Elastic Net Alphas") +
```

```
    xlab("Alpha Value") + ylab("Root Mean Squared Error")+
    scale_y_continuous(labels=scaleFUN) +
    theme_minimal()
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



In [75]:

```
m1 = cv.glmnet(train_mat, train$dage_y,
                      type.measure = 'mse', alpha = .6,
                      family = 'gaussian')
predicted2 = predict(m1,
                            s = m1$lambda.1se,
                            test_mat, type = 'response')

rmse = sqrt(mean((predicted2 - test$dage_y)^2))
mse = mean((predicted2 - test$dage_y)^2)
'MSE of best model'
mse
'RMSE of best model'
rmse
plot(m1)
coef(m1)
```

'MSE of best model'

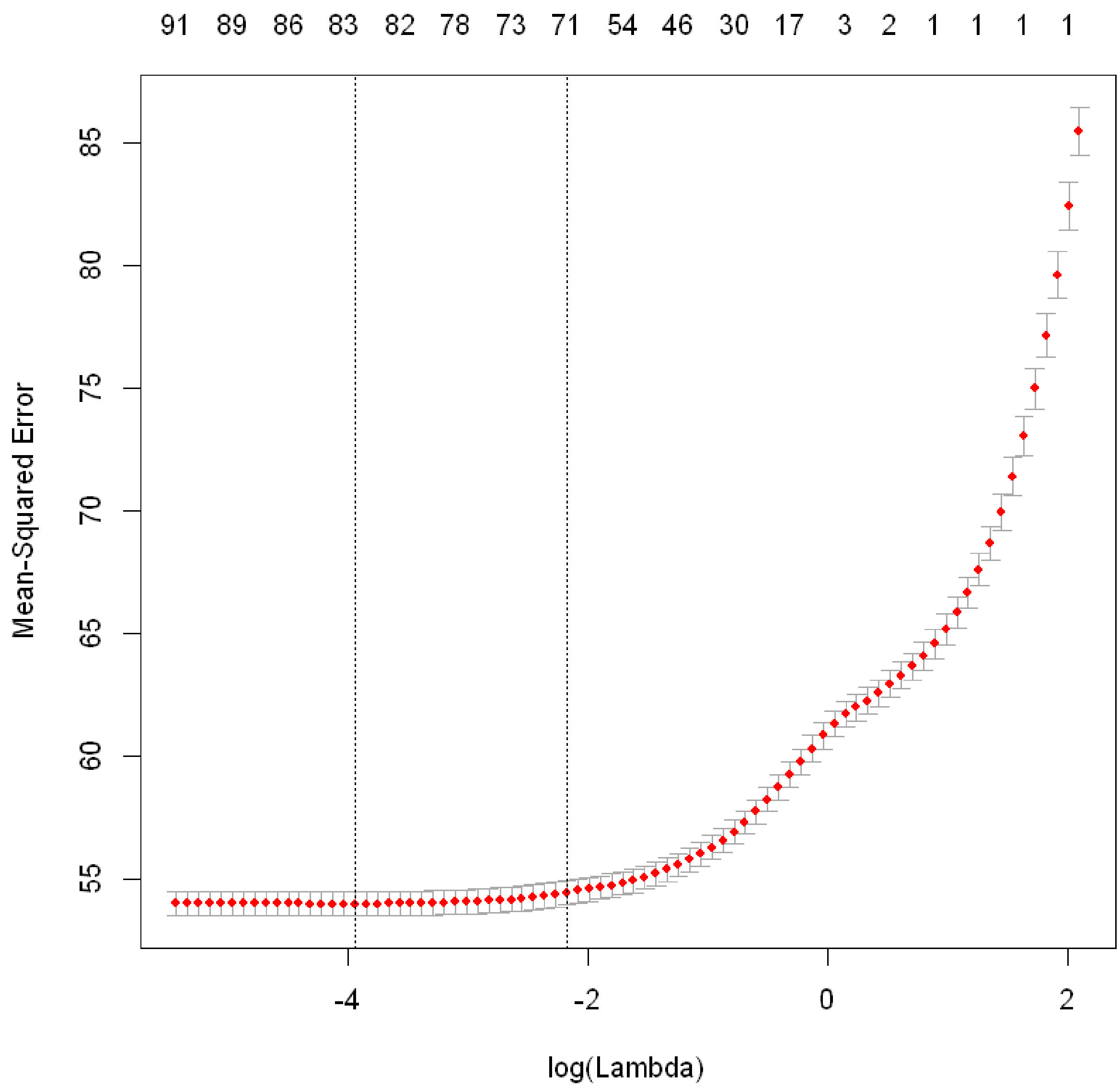
57.7901764368104

'RMSE of best model'

7.60198503266156

```
95 x 1 sparse Matrix of class "dgCMatrix"
                                                   1
(Intercept)                           -29.832706834
(Intercept)                             .
cogtot                                  .
gender2.female                          0.001589976
racem2.black/african american          -0.786579187
racem3.other                           -2.484980470
rahispan1.hispanic                     -2.499580675
agey_m                                  0.587750358
logiearn                                .
shlt2.very good                        -0.860460122
shlt3.good                              .
shlt4.fair                              0.371182036
shlt5.poor                              0.635500814
shltc-2                                 0.553256859
shltc-3                                 0.115169883
shltc-4                                 0.346869073
shltc.m                                 .
shltc.p                                 3.395510604
shltc0                                 -0.250610972
shltc1                                  .
shltc2                                  0.084143225
shltc3                                 -0.034246660
shltc4                                  2.901614193
raedyrs                                 0.062256636
mstat2.married,spouse absent            1.613768582
mstat3.partnered                       -0.813697023
mstat4.separated                       -0.790570920
mstat5.divorced                        -1.354845585
mstat6.separated/divorced               3.652765195
mstat7.widowed                          0.851274832
mstat8.never married                   -1.005610486
bmi                                    -0.093227694
smoken1.yes                             1.240609748
hibp1.yes                               0.248537213
hibp5.disp prev record (dk if cond)     4.920884297
drinkn                                 -0.321954280
pstmem2.same                            .
pstmem3.worse                           .
depres1.yes                             0.161951038
cendiv2.mid atlantic                    1.236990497
cendiv3.en central                      0.105831621
cendiv4.wn central                     -0.346408232
cendiv5.s atlantic                      .
cendiv6.es central                     -1.356827827
cendiv7.ws central                     -0.479780562
cendiv8.mountain                        .
cendiv9.pacific                         0.612656394
effort1.yes                             0.688825568
sleepr1.yes                             0.002564330
arthr1.yes                             -0.514769907
arthr4.disp prev record and no cond     .
heart1.yes                              0.090524505
heart4.disp prev record and no cond    -1.003993866
heart5.disp prev record (dk if cond)    3.440500874
heart6.preld prob:prev had/no new       7.447196140
strok1.yes                              0.722200757
strok2.tia/possible stroke              0.542524800
strok4.disp prev record and no cond     .
```

```
strok5.disp prev record (dk if cond)   0.821632831
psych1.yes                             0.229917658
psych4.disp prev record and no cond    .
psych5.disp prev record (dk if cond)   6.016290567
cancr1.yes                             0.494652648
diab1.yes                             -0.017395440
lung1.yes                              0.166199360
lung4.disp prev record and no cond    -1.479093229
lung5.disp prev record (dk if cond)    6.098700224
slfmem2.very good                      0.673802434
slfmem3.good                          -0.035688783
slfmem4.fair                           .
slfmem5.poor                           .
covs0.no                              -1.154684851
covs1.yes                             -0.099476455
lbrf2.works pt                         0.015719212
lbrf3.unemployed                      -0.139276928
lbrf4.partly retired                  -0.813634594
lbrf5.retired                          .
lbrf6.disabled                         1.455306113
lbrf7.not in lbrf                      2.245055612
smokev1.yes                            .
conde                                 -0.620457588
hiltc.m=oth missing                    .
hiltc0.no                              .
hiltc1.yes                            -0.383890107
spcfac.m=oth missing                   .
spcfac.n=in nhm                        9.639340738
spcfac0.no                             .
spcfac1.yes                           -1.638911981
loghspti                               0.608064731
rarelig2.catholic                      .
rarelig3.jewish                        .
rarelig4.none/no pref                 -0.176886995
rarelig5.other                         1.256117931
ravetrn1.yes                           1.744003720
loghatotb                             -0.047137478
```



In [76]:
```
m1$lambda.1se
```

0.113517259274811