Elastic Net Models

Walter K. Kremers, Mayo Clinic, Rochester MN

10 May 2025

Introduction

From our experience the relaxed lasso models involving a wieghted average of coefficients between the fully penalized lasso model and the unpenalized regression model based upon the non zero terms (from the penalized lasso model) generally provides a balance bewteen good fit and parsimony (fewer terms and so a simpler model). We understand may find advantage of the elastic net model which also involves a weighted average between a penalized and unpanalized regression models but allows for the penalty to involve a combination of L1 and L2 distance measures (See the vignette "An Introduction to glmnet" at https://CRAN.R-project.org/package=glmnet).

To allow one to investigate how the elastic net model may benefit their data the nested.glmnetr() program can fit elastic net models for multiple values of both alpha, the mixing paramater for the L1 and L2 penalty metics, and gamma, the mixing paramter between the penalzed and unpenalzed fits. As for the other models the nested.glmnetr() function performs a "simple" nested cross validation of the elastic net model in parallel with all other fitted models, even if not involving all the advantages of the nested corss valdiaiton described by Bates, Hastie and Tibshirani.

In additon to comparing the nested cross validation performances of deviances, agreement and calibration one can graphically inspect the (non nested) cross validation deviance values for the candidate models as function of each of the alpha, gamma and lambda.

An example dataset

To demonstrate usage of *cv.stepreg* we first generate a data set for analysis, run an analysis and evaluate. Following the *Using glmnetr* vignette, the code

```
# Simulate data for use in an example survival model fit
# first, optionally, assign a seed for random number generation to get applicable results
set.seed(116291950)
simdata=glmnetr.simdata(nrows=1000, ncols=100, beta=NULL)
```

generates simulated data for analysis. We extract data in the format required for input to the *nested.glmnetr* (and *glmnetr*) programs.

```
# Extract simulated survival data
xs = simdata$xs        # matrix of predictors
y_ = simdata$y_        # vector of numerical responces
```

Inspecting the predictor matrix and outcome vector we see

```r
# Check the sample size and number of predictors
print(dim(xs))
```

```
## [1] 1000  100
```

```r
# Check the rank of the design matrix, i.e. the degrees of freedom in the predictors
Matrix::rankMatrix(xs)[[1]]
```

```
## [1] 94
```

```r
# Inspect the first few rows and some select columns
print(round(xs[1:10,c(1:12,18:20)],digits=6))
```

```
##       X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12       X18       X19       X20
##  [1,]  1  0  0  0  1  0  1  0  0   0   0   0 -1.208898  0.056971 -0.565631
##  [2,]  1  1  0  0  0  0  0  0  1   0   0   0  0.395354  0.427313  0.185235
##  [3,]  1  0  0  1  0  1  0  0  0   0   0   0  1.044608 -0.746960  0.964274
##  [4,]  1  1  0  0  0  0  0  1  0   0   0   0  0.028859 -1.277651  0.203243
##  [5,]  1  0  0  1  0  1  0  0  0   0   0   0 -1.205172 -1.287454 -1.698229
##  [6,]  1  0  0  0  1  0  1  0  0   0   0   0 -1.158210 -0.068841  1.458800
##  [7,]  1  0  0  0  1  0  0  0  1   0   0   0  0.151713  1.095396  1.476831
##  [8,]  1  0  0  1  0  0  1  0  0   0   0   0 -0.139246 -0.424550  0.073340
##  [9,]  1  1  0  0  0  0  0  1  0   0   0   0 -0.069326  0.172792  1.039656
## [10,]  1  0  0  1  0  0  1  0  0   0   0   0  0.677420  1.185946 -1.473551
```

```r
summary(y_)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -8.8964 -0.5874  1.3870  1.3148  3.3802  8.2972
```

Fitting an Elastic Net model

To fit the elastic net model we call the neted.glmnetr() function line in the "An Introduction to glmnetr" while now specifying a vector of candidate alpha values using the alpha option in the function call. For comparison we are also fitting the random forest model as well as the full model including all candiate predictors.

```r
# Fit a relaxed lasso model informed by cross validation
nested_elastic_fit = nested.glmnetr(xs, start=NULL, y_, event=NULL, family="gaussian", resample=NULL, f
                                    dolasso=1, doxgb=0, dorf=1, doorf=0, doann=0, dorpart=0, dostep=0, d
                                    alpha=seq(0,1,0.2), seed=791590258, track=0)
```

```
## 
##  seed$seedr[1] =  791590258
```

When alpha is not specified the value c(1) is used to fit models based only on the L1 penalty. By default the candidate values for gamma are c(0, 0.25, 0.5, 0.75, 1), as suggested in the glmnet package.

A tabular summary of model performances

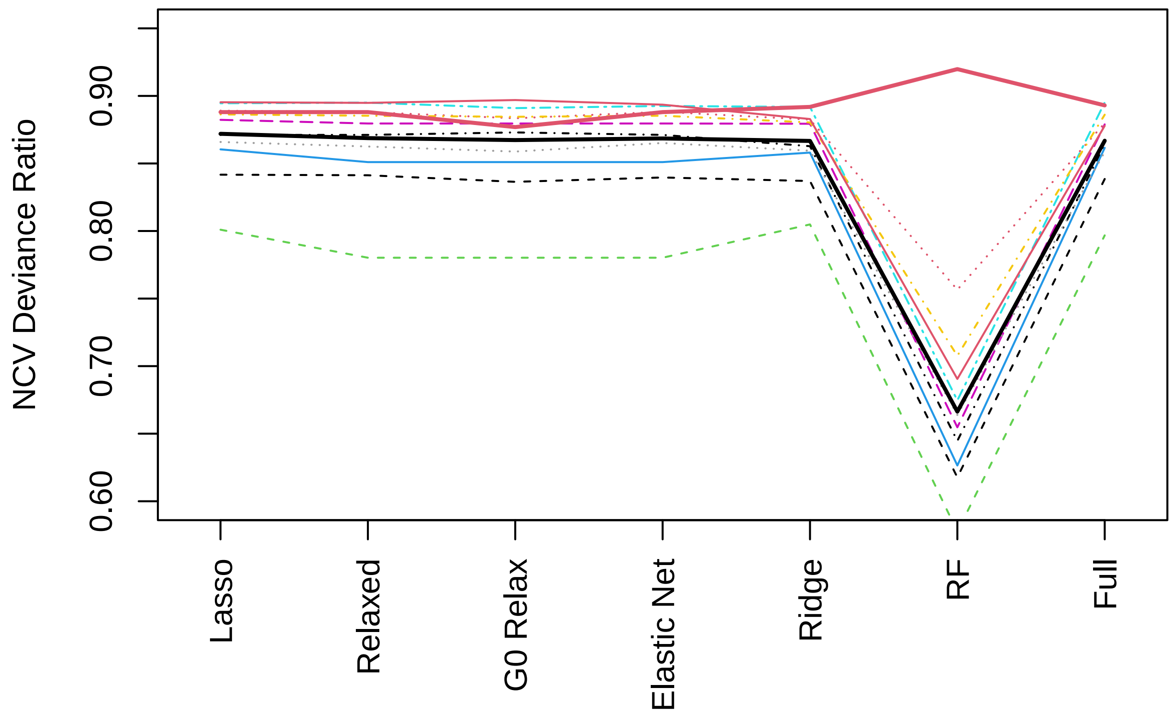
```r
# Fit a relaxed lasso model informed by cross validation
summary(nested_elastic_fit)
```

```
##   Sample information including number of records, number of columns in
##   design (predictor, X) matrix, and df (rank) of design matrix:
##     family          n xs.columns      xs.df null.dev/n
##   gaussian       1000        100         94       7.96
##
##  For LASSO, Random Forest (RF), average (Ave) model performance measures from the
##  10-fold (NESTED) Cross Validation are given together with naive summaries
##  calculated using all data without cross validation
##
##                        Ave DevRat Ave Int Ave Slope Ave R-square Ave Non Zero
## lasso                      0.8720 -0.0268    1.0239       0.8749         56.6
## lassoR                     0.8687 -0.0101    1.0097       0.8717         30.9
## lassoR0                    0.8673  0.0133    0.9915       0.8696         19.4
## elastic net                0.8685 -0.0129    1.0077       0.8714         26.1
## ridge                      0.8667 -0.0265    1.0201       0.8700         99.0
##                        Naive DevRat Naive R-square Non Zero
## lasso                        0.8880         0.9428       58
## lassoR                       0.8880         0.9428       58
## lassoR0                      0.8769         0.9364       14
## elastic net                  0.8880         0.9428       58
## ridge                        0.8919         0.9448       99
##
##                        Ave DevRat Ave Int Ave Slope Ave R-square Ave Non Zero
## RF Simple                  0.6663 -0.2819    1.2132       0.6922           56
##                        Naive DevRat Naive R-square Non Zero
## RF Simple                    0.9198          0.943       60
##
##                        Ave DevRat Ave Int Ave Slope Ave R-square Ave Non Zero
## Full regression            0.8668       0         0       0.8701           94
##                        Naive DevRat Naive R-square Non Zero
## Full regression               0.893          0.945       94
```

A graphical summary of model performances

Model deviance ratios, measures of agreement (r-square or concordance) and linear calibration coefficients obtained from nested cross validation can be displayed graphically, including values from calculated from individual fold values, the averages over the folds and the naive values obtained basing caculations on the training data are displayed. Here we present only the figure for deviance ratios.

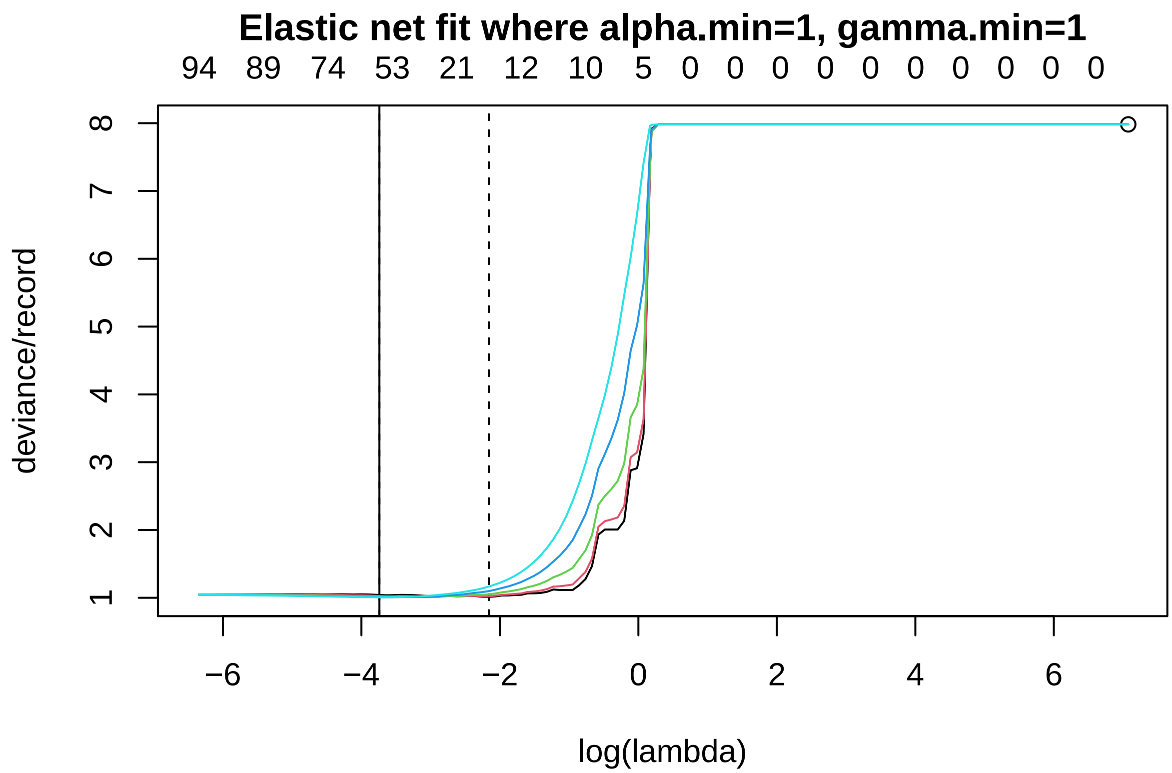
```r
# Fit a relaxed lasso model informed by cross validation
plot(nested_elastic_fit, ylim=c(0.6,0.95))
```



Graphical presentations of cross validation deviances

```
# Fit a relaxed lasso model informed by cross validation
plot(nested_elastic_fit , type="elastic")
```

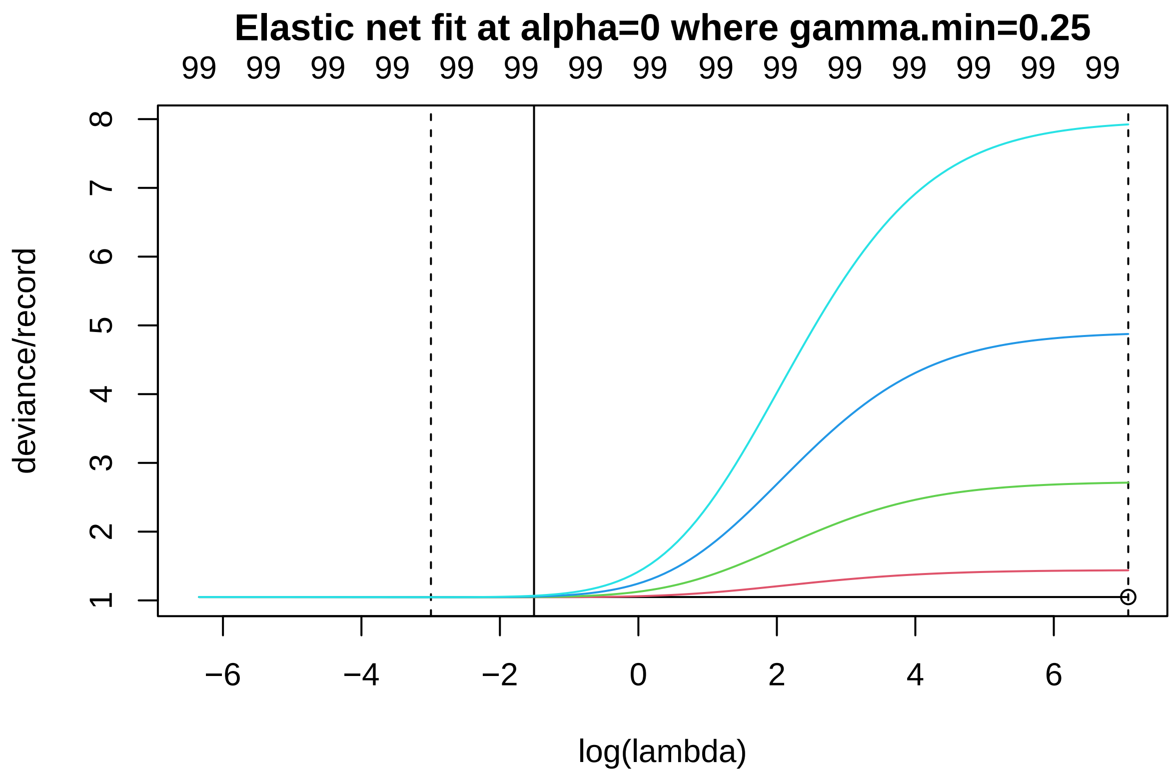
```
## Elastic Net tuned for alpha, gamma and lambda minimizing CV average deviance (maximizing log likeliho
##    alpha.min= 1, gamma.min = 1, log(lambda) = -3.74, df = 58, deviance = 1.0081
```



Here we see that all curves for differnt vlaues of gamma are for the same loss minimizing value for alpha of 1. We can also plot the curves for other values of alpha, for eample 0 correspoinding to a pure L2 penaly.

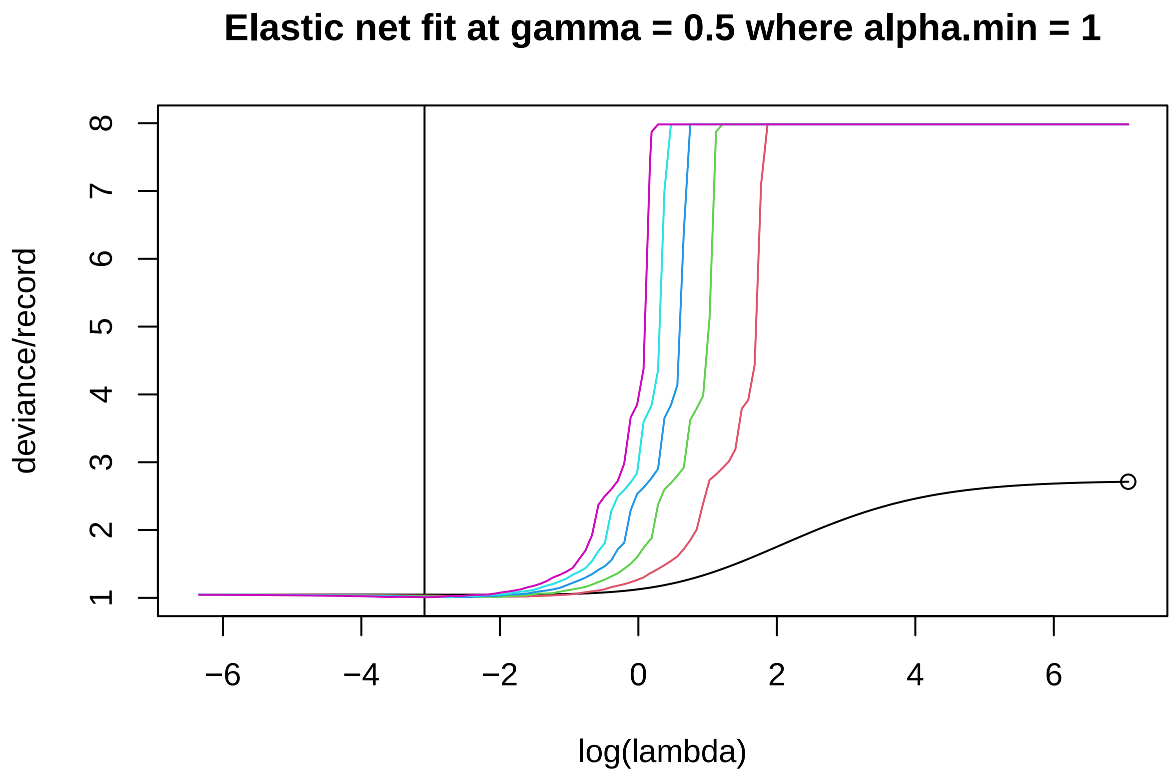
```
# Fit a relaxed lasso model informed by cross validation
plot(nested_elastic_fit , type="elastic", alpha = 0)
```

```
## Elastic Net at alpha = 0 tuned for gamma and lambda minimizing CV average deviance (maximizing log li
##    gamma.min = 0.25, log(lambda) = -1.507, df = 99, deviance = 1.0468
```



We can also plot the deviance curves for different values of alpha when specifying a single value for gamma.

```
# Fit a relaxed lasso model informed by cross validation
plot(nested_elastic_fit , type="elastic", gamma = 0.5)
```



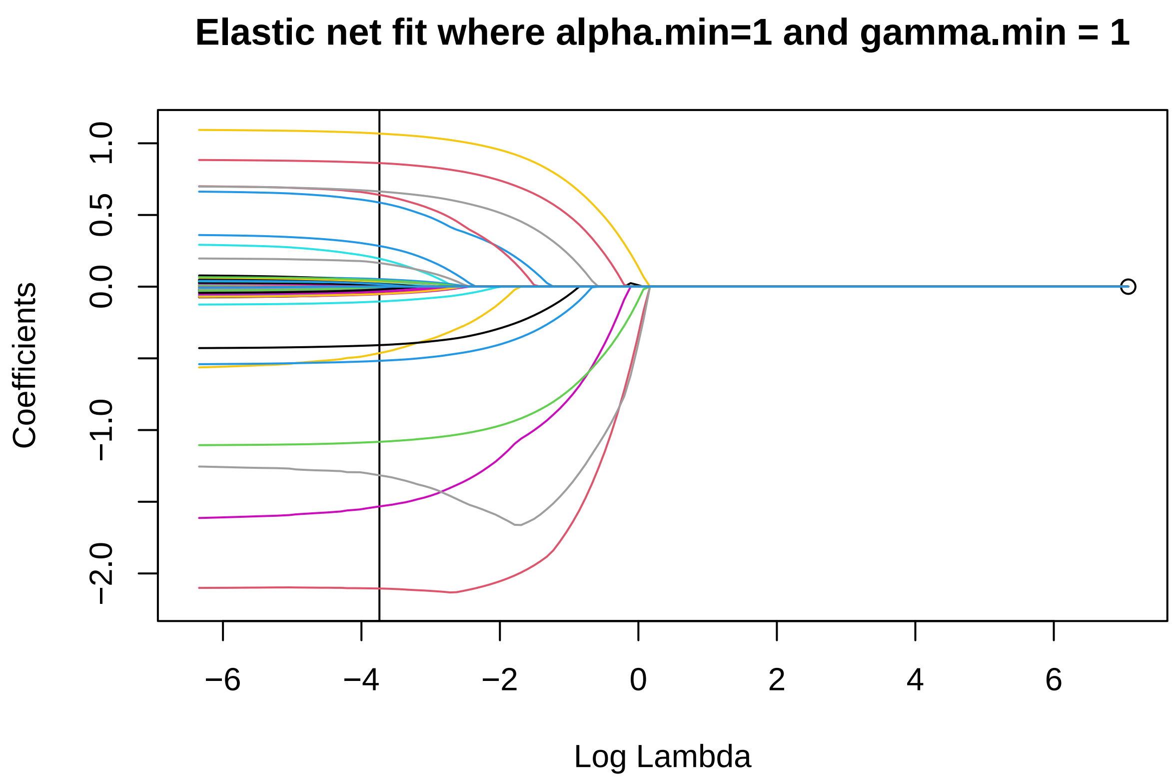

```
## Elastic Net at gamma = 0.5 tuned for alpha and lambda minimizing CV average deviance (maximizing log
##    alpha.min = 1, log(lambda) = -3.088, df = 33, deviance = 1.0097
```

Graphical presentations of beta estimates

Similary we can plot the beta estimates for the optimizing values for alpha and gamma as a function of lambda.

```
# Fit a relaxed lasso model informed by cross validation
plot(nested_elastic_fit , type="elastic", coefs=1)
```

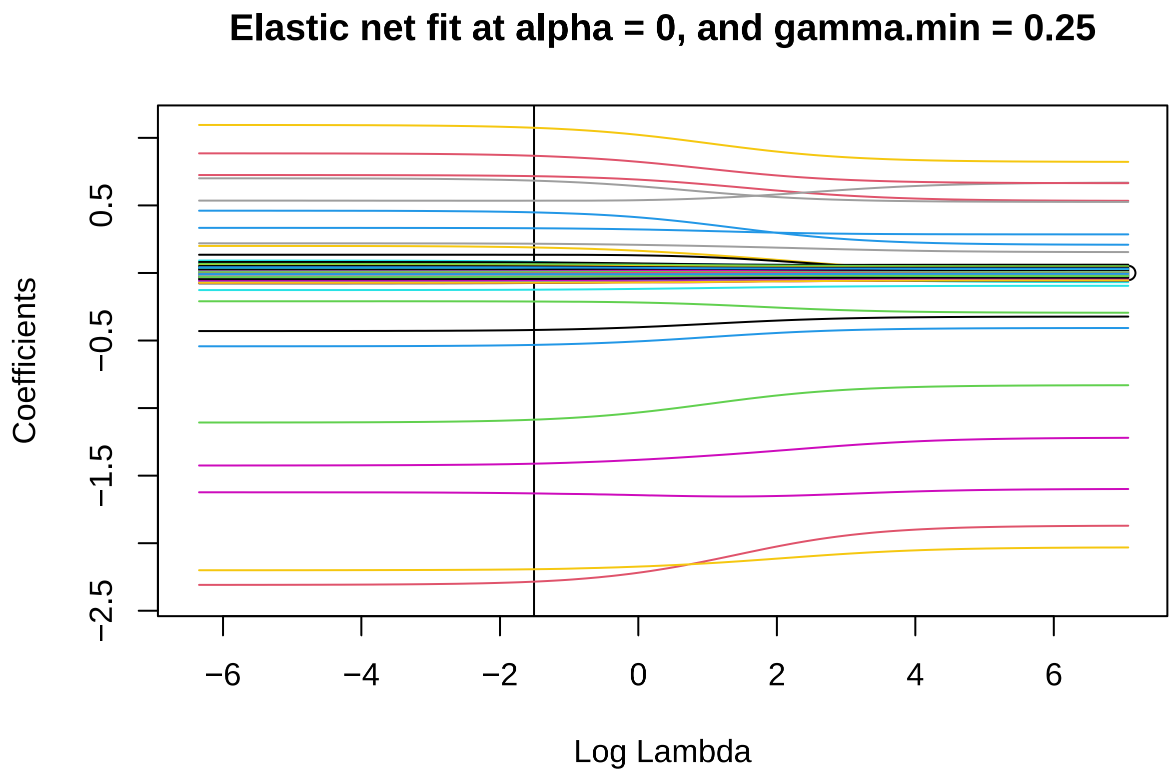
```
##   For Elastic net fit where alpha.min = 1 and gamma.min = 1,
##   log(lambda.min) = -3.74
```



Similary we can plot beta's for other values of alpha and gamma, specifying eihter alpha, gamma or both. If we spedify only one of alpha or gamma, the plot will search for the other value minimizing the cross validation deviance.

```
# Fit a relaxed lasso model informed by cross validation
plot(nested_elastic_fit , type="elastic", coefs=1, alpha=0)
```

```
##   For elastic net fit at alpha = 0
##    gamma.min = 0.25 and log(lambda.min) = -1.5072
```



corresponding to relaxed model restricing the penalty to the L2 metric.

Numerical values for beta and predicteds

To extract beta's or calculate predicteds we use the predict() function. By default predictions are given for the lasso model. Alternatively one may specify th model type as "lasso", "elastic" or "ridge".

```
# get betas  ...
betas = predict(nested_elastic_fit)
```

```
##    gamma.min = 1   lambda.min =  0.02375409   df = 59   deviance = 1.0081
```

```
 betas
```

```
## $beta_
##   (Intercept)            X1            X2            X3            X4
##  2.362494e+00  0.000000e+00 -2.105075e+00  0.000000e+00  5.865327e-01
##            X5            X6            X7            X8            X9
##  1.952103e-01  0.000000e+00 -4.634601e-01  1.650215e-01  0.000000e+00
##           X10           X11           X12           X13           X14
##  6.400439e-01  0.000000e+00  2.837661e-01 -1.182710e-02 -1.532684e+00
##           X15           X16           X17           X18           X19
##  1.348014e-13 -1.315465e+00  0.000000e+00  8.617157e-01 -1.082013e+00
##           X20           X21           X22           X23           X24
```

```
## -5.173393e-01 -1.034883e-01 -4.166578e-02  1.067433e+00  6.640276e-01
##           X25           X26           X27           X28           X29
## -4.077089e-01  0.000000e+00 -1.089376e-02  5.251897e-02  0.000000e+00
##           X30           X31           X32           X33           X34
##  0.000000e+00  9.403054e-03  1.620520e-02  3.611231e-02 -4.367165e-02
##           X35           X36           X37           X38           X39
##  0.000000e+00  0.000000e+00 -3.111932e-02  0.000000e+00  2.756439e-02
##           X40           X41           X42           X43           X44
##  3.210557e-02 -4.764198e-02  1.821047e-02  4.405780e-02  0.000000e+00
##           X45           X46           X47           X48           X49
## -8.634518e-03  0.000000e+00  0.000000e+00  0.000000e+00  9.658628e-03
##           X50           X51           X52           X53           X54
##  0.000000e+00  0.000000e+00  0.000000e+00  1.053695e-02 -3.338629e-02
##           X55           X56           X57           X58           X59
##  3.743556e-03  0.000000e+00  9.387409e-03  0.000000e+00  5.558458e-03
##           X60           X61           X62           X63           X64
##  1.970052e-02 -1.222905e-02 -5.282828e-02  3.003265e-02  0.000000e+00
##           X65           X66           X67           X68           X69
## -9.070911e-03  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
##           X70           X71           X72           X73           X74
## -4.847424e-02  0.000000e+00  0.000000e+00  0.000000e+00 -1.011700e-02
##           X75           X76           X77           X78           X79
##  0.000000e+00  3.413472e-03  0.000000e+00  3.633437e-03 -5.029406e-02
##           X80           X81           X82           X83           X84
## -1.110065e-02  1.054073e-02  0.000000e+00 -8.875924e-03 -1.246646e-02
##           X85           X86           X87           X88           X89
##  2.291093e-02  0.000000e+00  0.000000e+00  0.000000e+00  1.915288e-02
##           X90           X91           X92           X93           X94
##  0.000000e+00  0.000000e+00  1.665436e-02  0.000000e+00 -1.928135e-02
##           X95           X96           X97           X98           X99
##  0.000000e+00  0.000000e+00 -1.965333e-02  0.000000e+00 -2.165041e-03
##          X100
##  0.000000e+00
##
## $beta
##   (Intercept)            X2            X4            X5            X7
##  2.362494e+00 -2.105075e+00  5.865327e-01  1.952103e-01 -4.634601e-01
##            X8           X10           X12           X13           X14
##  1.650215e-01  6.400439e-01  2.837661e-01 -1.182710e-02 -1.532684e+00
##           X15           X16           X18           X19           X20
##  1.348014e-13 -1.315465e+00  8.617157e-01 -1.082013e+00 -5.173393e-01
##           X21           X22           X23           X24           X25
## -1.034883e-01 -4.166578e-02  1.067433e+00  6.640276e-01 -4.077089e-01
##           X27           X28           X31           X32           X33
## -1.089376e-02  5.251897e-02  9.403054e-03  1.620520e-02  3.611231e-02
##           X34           X37           X39           X40           X41
## -4.367165e-02 -3.111932e-02  2.756439e-02  3.210557e-02 -4.764198e-02
##           X42           X43           X45           X49           X53
##  1.821047e-02  4.405780e-02 -8.634518e-03  9.658628e-03  1.053695e-02
##           X54           X55           X57           X59           X60
## -3.338629e-02  3.743556e-03  9.387409e-03  5.558458e-03  1.970052e-02
##           X61           X62           X63           X65           X70
## -1.222905e-02 -5.282828e-02  3.003265e-02 -9.070911e-03 -4.847424e-02
##           X74           X76           X78           X79           X80
```

```
## -1.011700e-02  3.413472e-03  3.633437e-03 -5.029406e-02 -1.110065e-02
##           X81           X83           X84           X85           X89
##  1.054073e-02 -8.875924e-03 -1.246646e-02  2.291093e-02  1.915288e-02
##           X92           X94           X97           X99
##  1.665436e-02 -1.928135e-02 -1.965333e-02 -2.165041e-03
```

```r
# predicteds ...
preds = predict(nested_elastic_fit , xs)
```

```
##    gamma.min = 1   lambda.min =  0.02375409   deviance = 1.0081
```

```r
preds[1:10]
```

```
##  [1] -2.10717422 -0.06894307  1.47968865  2.03575200  3.31775663 -2.40491613
##  [7]  1.21467420  0.71839384  0.21358677 -0.86919865
```

For this case the best lasso model is the "fully" penalized (relaxed) lasso model.

```r
# get betas  ...
betas = predict(nested_elastic_fit, type="elastic" )
betas

# predicteds ...
preds = predict(nested_elastic_fit , xs, type="elastic" )
preds[1:10]
```

For this analysis the best elastic net model is the same as the best lasso model, so we suppressed here the printing out of the same numbers.