An Introduction to `extenedglmnet`

Weiwei Tao

May 03, 2022

Contents

Introduction

extendedglmnet package is an extension of glmnet package. Users can either fit a linear model or generalized linear model depending the response variable types. It can support ridge, lasso and random lasso methods. User can choose to use cross validation to find the optimal regularization parameter lambda or use user specified lambda. It can deal with both continuous and binary responses. The package includes methods for prediction and print, and evalution.

The authors of extendedglmnet is Weiwei Tao. This vignette describes basic usage of extendedglmnet in R.

Lasso and ridge regulation are known to be suffering from that it can select only one or a few of a set of highly correlated important variables. If several independent data sets were generated from the same distribution, then we would expect lasso to select nonidentical subsets of those highly correlated important variables from different data sets. Random Lasso proposed by Sijian Wang et al. solved this issue by using bootstrap, which will provide more stable results than lasso and ridge when number of predictors are greater than number of observations. Random lasso algorithm is a two-step approach including: - step1: the lasso method is applied to many bootstrap samples, each using a set of randomly selected covariates. A measure of importance is yielded from this step for each covariate. - step2: a similar procedure to the first step is implemented with the exception that for each bootstrap sample, a subset of covariates is randomly selected with unequal selection probabil- ities determined by the covariates' importance. The final set of covariates and their coefficients are determined by averaging boot- strap results obtained from step 2.

Installation

Like many other R packages, the simplest way to obtain `extendedglmnet` is to install it directly from CRAN. Type the following command in R console:

```
install.packages("extendedglmnet")
```

Quick Start

The purpose of this section is to give users a general sense of the package. We will briefly go over the main functions, basic operations and outputs. After this section, users may have a better idea of what functions are available, which ones to use, or at least where to seek help.

First, we load the `extendedglmnet` package:

```
library(extendedglmnet)
```

The default family used in the package is the Guassian linear model or "least squares", which we will demonstrate in this section. We generate data sample as following:

```
size = 50
x = matrix(runif(55*size),ncol=55)
y = 10*x[,1] + 3*x[,2] + 20*(x[,3]-0.5)**2 + 10*x[,4] + 5*x[,35] + runif(1,0,1)
```

We split the data into training set and testing set with ratio of 7:3.

```
index = sample(1:size, 0.7*size)
trainx = x[index,]
testx = x[-index,]
trainy = y[index]
testy = y[-index]
```

Linear Regression: `family = "gaussian", type = "regression"` (default)

`"gaussian"` is the default `family` argument for the function `extendedglmnet`. We can fit the model using:

```
fit.lr<- extendedglmnet(trainx, trainy, family = "gaussian", type = "regression")
```

```
##
## Call:
## lm(formula = y ~ ., data = dat)
##
## Coefficients:
## (Intercept)           x1           x2           x3           x4           x5
##     -37.955        1.238      118.850     -122.506       -9.495       22.998
##          x6           x7           x8           x9          x10          x11
##     -34.582       96.766     -115.606      107.184       81.370       54.965
##         x12          x13          x14          x15          x16          x17
##      90.528      -50.504       79.555     -142.622      -11.926       -9.268
##         x18          x19          x20          x21          x22          x23
```

```
##    -137.990      149.882      -48.687      -51.938      -43.667      -23.584
##         x24          x25          x26          x27          x28          x29
##      -8.707      -79.047       15.819       23.178      110.623      -98.141
##         x30          x31          x32          x33          x34          x35
##      75.289      176.242       74.118      -97.418        3.107           NA
##         x36          x37          x38          x39          x40          x41
##          NA           NA           NA           NA           NA           NA
##         x42          x43          x44          x45          x46          x47
##          NA           NA           NA           NA           NA           NA
##         x48          x49          x50          x51          x52          x53
##          NA           NA           NA           NA           NA           NA
##         x54          x55
##          NA           NA
```

`fit` is a list that contains all the relevant information of the fitted model for further use including type of the fit, family, coefficient of the model, lambda (0 for regression without regulations), degree of freedom (number of non-zero coefficients), number of observations and number of covariates in the input matrix x. Two methods are provided for the object such as `print.extendglmnet, and`, `predict.extendglmnet` that enable us to explore the results elegantly.

We can visualize the fitted results by `print.extendglmnet` method:

```
print.extendglmnet(fit.lr)
```

```
##
## Call: linear regression for gaussian reponses
##
## (Intercept)          x1          x2          x3          x4          x5
##  -37.955085    1.238222  118.849870 -122.506452   -9.494525   22.998475
##          x6          x7          x8          x9         x10         x11
##  -34.582482   96.765723 -115.605619  107.184372   81.369538   54.965317
##         x12         x13         x14         x15         x16         x17
##   90.527593  -50.503838   79.555466 -142.622201  -11.925832   -9.268153
##         x18         x19         x20         x21         x22         x23
## -137.990204  149.881988  -48.687164  -51.938098  -43.667041  -23.583528
##         x24         x25         x26         x27         x28         x29
##   -8.706952  -79.046892   15.819482   23.177586  110.622600  -98.141226
##         x30         x31         x32         x33         x34         x35
##   75.289263  176.242365   74.117612  -97.418208    3.107115          NA
##         x36         x37         x38         x39         x40         x41
##          NA          NA          NA          NA          NA          NA
##         x42         x43         x44         x45         x46         x47
##          NA          NA          NA          NA          NA          NA
##         x48         x49         x50         x51         x52         x53
##          NA          NA          NA          NA          NA          NA
##         x54         x55
##          NA          NA
## Degrees of Freedom: 34 Total (i.e. Null); 0 Residual
```

The training dataset contains 35 observations and 55 potential predictors. The number of predictors are greater than number of obervations. Traditional linear regression will automatically assign NA values to the last 20 covariates.

We can predict responses based upon new data we have.

```r
ypred <- predict.extendglmnet(fit.lr, newx = testx, type = "response")
```

```
## Warning in predict.extendglmnet(fit.lr, newx = testx, type = "response"):
## prediction from a rank-deficient fit may be misleading
```

The results may be misleading due to rank-deficient issue. We can also perform model evaluation by calculating MSE, RMSE, MAE for continuous responses.

```r
evaluation(testy, ypred, "MSE")
```

```
## [1] 20486.68
```

```r
evaluation(testy, ypred, "RMSE")
```

```
## [1] 143.1317
```

```r
evaluation(testy, ypred, "MAE")
```

```
## [1] 100.0276
```

Ridge/Lasso Regression: `family = "gaussian", type = "ridge" or "lasso"`

We can also choose to fit our model with either ridge or lasso regulations by changing type assigned.

```r
fit.ridge<- extendedglmnet(trainx, trainy, family = "gaussian", type = "ridge")
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```r
print.extendglmnet(fit.ridge)
```

```
##
## Call: ridge for gaussian reponses
##
## 56 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) 14.295658851
## x1           1.070789201
## x2           0.952331045
## x3           0.113624335
## x4           1.034586331
## x5          -0.348833284
## x6          -0.201037585
## x7          -0.327345237
## x8           0.232384457
## x9          -0.188865329
## x10          0.323344362
## x11         -0.052110709
## x12         -0.576715792
```

```
## x13          0.147739111
## x14         -0.165024095
## x15          0.388248957
## x16         -0.169200098
## x17          0.024729106
## x18         -0.622600888
## x19          0.062159754
## x20          0.120605993
## x21         -0.283741539
## x22          0.009952784
## x23         -0.248815035
## x24         -0.167245765
## x25          0.104888877
## x26         -0.161617984
## x27          0.523101128
## x28         -0.115528250
## x29          0.239511895
## x30         -0.458894645
## x31          0.358419659
## x32         -0.493709521
## x33         -0.491038003
## x34         -0.125852884
## x35          0.883575639
## x36          0.244912683
## x37          0.630440405
## x38          0.164663877
## x39          0.052937361
## x40         -0.009862988
## x41          0.010904017
## x42         -0.099150185
## x43          0.155156534
## x44         -0.042208190
## x45          0.018702362
## x46          0.068465562
## x47          0.261085446
## x48          0.318050146
## x49          0.593909615
## x50         -0.154751664
## x51         -0.124350423
## x52         -0.154104714
## x53         -0.191581231
## x54          0.186233070
## x55          0.368092304
## Degrees of Freedom: 34 Total (i.e. Null); -21 Residual
```

Similarly, we can predict responses based upon new testing data we have.

```
ypred <- predict.extendglmnet(fit.ridge, newx = testx, type = "response")
```

The MSE by using ridge regression is much smaller than that using linear regression without regulation.

```
evaluation(testy, ypred, "MSE")
```

```
## [1] 19.49856
```

```
evaluation(testy, ypred, "RMSE")
```

```
## [1] 4.415717
```

```
evaluation(testy, ypred, "MAE")
```

```
## [1] 3.26723
```

The MSE with lasso is the smallest as comparing to Ridge and linear regression.

```
fit.lasso<- extendedglmnet(trainx, trainy, family = "gaussian", type = "lasso")
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
ypred <- predict.extendglmnet(fit.lasso, newx = testx, type = "response")
evaluation(testy, ypred, "MSE")
```

```
## [1] 5.002703
```

```
evaluation(testy, ypred, "RMSE")
```

```
## [1] 2.236672
```

```
evaluation(testy, ypred, "MAE")
```

```
## [1] 1.835713
```

function arguments for choice of lambdas

`extendedglmnet` provides various arguments for users to customize the cross validation process during the fit.

- `cv` is an indicator variable to specify whether cross validation should be used to pick optimal lambda value. Default is TRUE.
- `lambda` A user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence with default values in glmnet package. Supplying a value of lambda overrides this.
- `nfolds` number of folds - default is 5. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds = 3.

As an example, we can specify user defined range of lambdas during the cross validation process of the fit.

```
grid=10^seq(6,-2,length=10)
fit.lasso<- extendedglmnet(trainx, trainy, family = "gaussian", type = "lasso", lambda = grid)
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```r
ypred <- predict.extendglmnet(fit.lasso, newx = testx, type = "response")
evaluation(testy, ypred, "MSE")
```

```
## [1] 4.612114
```

Random Lasso Method: `family = "gaussian", type = "random lasso"`

So far, what we have introduced were the same as what provided in the glmnet package. An important feature in extendedglmnet package is that it provide use the choice of random lasso method which can provide more stable fitting results.

For random lasso method, 3 additional parameters need to specified: * `B` number of bootstrap samples in random lasso. One can take B = 500 or B = 1000. The default is 200. A large B can make computation time much longer.

- `q1` and `q2` number of candidate variables to be included in each bootstrap sample during generating importance measures in random lasso. This value need to be tuned to ensure best performance. The value should be positive and no greater than number of predictors in x.

We leave B as the default of 200 and choose q1 and q2 to be 30.

```r
fit.rl<- extendedglmnet(trainx, trainy, family = "gaussian", type = "random lasso", q1 = 30, q2 = 30)
print.extendglmnet(fit.rl)
```

```
##
## Call: random lasso for gaussian reponses
##
##   (Intercept)            x1            x2            x3            x4
##  3.8172527738  7.7487574249  3.2468858109 -0.1377096380  8.1621008508
##            x5            x6            x7            x8            x9
## -0.3562675041  0.0160993094 -0.1830059197  0.0510967099 -0.0015133674
##           x10           x11           x12           x13           x14
##  0.1566633630 -0.0028762499 -0.1624204036 -0.0362693828 -0.0390804245
##           x15           x16           x17           x18           x19
##  0.2245372071  0.0161876408 -0.0616911309 -0.2348667533  0.1239332739
##           x20           x21           x22           x23           x24
##  0.0593786023 -0.1512250221  0.1419985095  0.0211719788 -0.0342233294
##           x25           x26           x27           x28           x29
##  0.0000000000 -0.1737380230  0.5888846589 -0.0611605978  0.1481412725
##           x30           x31           x32           x33           x34
## -0.3969996985  0.0373118907  0.0003646149 -0.3593316037  0.0687764832
##           x35           x36           x37           x38           x39
##  4.1161806946  0.1580327448  0.6895255911 -0.0219759430  0.1742213955
##           x40           x41           x42           x43           x44
## -0.0150866980  0.0319384293 -0.0208836154  0.0545422994 -0.0491624666
##           x45           x46           x47           x48           x49
## -0.0028814032  0.0111292741 -0.0391602313  0.1828010481  0.0639549080
##           x50           x51           x52           x53           x54
##  0.1269437911  0.0043059173 -0.0004242010 -0.0527697343  0.6012171162
##           x55
##  0.4487400386
## Degrees of Freedom: 34 Total (i.e. Null); -20 Residual
```

The MSE with random lasso is comparable to that of lasso.

```
ypred <- predict.extendglmnet(fit.rl, newx = testx, type = "response")
evaluation(testy, ypred, "MSE")
```

```
## [1] 4.101258
```

```
evaluation(testy, ypred, "RMSE")
```

```
## [1] 2.025156
```

```
evaluation(testy, ypred, "MAE")
```

```
## [1] 1.637545
```

Logistic Regression: `family = "binomial", type = "regression"`

extendedglmnet package supports binomial responses as well. Similarly we generated a simulated dataset and first apply logistic regression to that dataset.

```
size = 200
x = matrix(rnorm(210*size),ncol=210)
z = 1 + 2*x[,1] + 3*x[,40] - 4*x[,50] + 3*x[,210]
pr = 1/(1+exp(-z))
y = rbinom(size,1,pr)

index = sample(1:size, 0.7*size)
trainx = x[index,]
testx = x[-index,]
trainy = y[index]
testy = y[-index]
```

For binary responses, we need simply change family argument from "gaussian" to "binomial".

```
fit.logistic<- extendedglmnet(trainx, trainy, family = "binomial", type = "regression")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
print.extendglmnet(fit.logistic)
```

```
##
## Call: logistic regression for binomial reponses
##
## (Intercept)          x1          x2          x3          x4          x5
## 102.1968683  -0.8848632 -79.9909796  14.4607602 -37.0768962 -54.1391906
##          x6          x7          x8          x9         x10         x11
##  46.5825988 -38.2960689 -62.9257544  18.4584896  34.5917733  35.2767924
##         x12         x13         x14         x15         x16         x17
##  43.3243087  22.8861457 -31.9150639  79.2167712  -9.8181146 -53.1417141
```

```
##         x18         x19         x20         x21         x22         x23 
## -56.6227046  72.8810744 -20.1420558  63.3757357  90.3816582  -5.7623812 
##         x24         x25         x26         x27         x28         x29 
##  13.1139870 -13.7604254 -18.5873808  -3.5868768 -86.9981346 -93.4252389 
##         x30         x31         x32         x33         x34         x35 
## -53.9270350  -0.2607302  37.2378852 -31.0568648  76.7881058 -20.1807277 
##         x36         x37         x38         x39         x40         x41 
## -24.7432991  -1.1539312 -64.0471040 -39.4898811  11.8937655  12.8817907 
##         x42         x43         x44         x45         x46         x47 
##  45.8417642   2.9049976  35.9500452   5.0145353   5.4189297 -15.7475237 
##         x48         x49         x50         x51         x52         x53 
##  42.3171404  -9.9672188 -23.2810273 -20.0187980  25.6596439 -10.4072370 
##         x54         x55         x56         x57         x58         x59 
##  22.3702132  92.9697830  32.7299540  45.6856519  40.5952378  20.6105462 
##         x60         x61         x62         x63         x64         x65 
## -22.3067086   2.9213384  64.3909626  30.0546029  49.2745782 -54.9179146 
##         x66         x67         x68         x69         x70         x71 
##  18.5312273  14.6629000 -98.4046052  33.1898084 -27.4824580 -17.2834407 
##         x72         x73         x74         x75         x76         x77 
##  44.1054128 -30.6301024 -24.6771787 -18.4573297  26.9893772   9.5284000 
##         x78         x79         x80         x81         x82         x83 
##  11.9345856 -41.0937222  11.3024631 -49.3396452  57.2012167 -53.9602809 
##         x84         x85         x86         x87         x88         x89 
##  58.9431004  97.9406481 -26.0184592  43.1244601  29.0727807  -3.6716322 
##         x90         x91         x92         x93         x94         x95 
##  63.9849794  33.7779954  19.4079093  11.9904775 -34.9545750 -33.3508651 
##         x96         x97         x98         x99        x100        x101 
## -26.3209718  63.8038093 -33.0351215 -58.7444246 -96.2578603 -10.9738672 
##        x102        x103        x104        x105        x106        x107 
## -22.0528061 -65.2893154   4.6244911  36.6632337   6.3923739  78.8178094 
##        x108        x109        x110        x111        x112        x113 
##  21.5880250 -58.1887422 -25.8867080  45.5533803 -11.9183376   9.0574742 
##        x114        x115        x116        x117        x118        x119 
##  47.3839862  -6.0088098  28.4289124 -29.3909355 -38.9766652  16.8365755 
##        x120        x121        x122        x123        x124        x125 
##   9.8403896 -74.5365844  73.6744629 -13.7666430 -37.4649604  62.5503701 
##        x126        x127        x128        x129        x130        x131 
##  37.5594634  35.4039516 -27.9126678 -12.8459048  56.9942713   9.0782037 
##        x132        x133        x134        x135        x136        x137 
## -30.0429152  30.9478071 -16.1730499  14.7107869 -12.5234679  49.2772683 
##        x138        x139        x140        x141        x142        x143 
## -17.5242693 -13.0017961          NA          NA          NA          NA 
##        x144        x145        x146        x147        x148        x149 
##          NA          NA          NA          NA          NA          NA 
##        x150        x151        x152        x153        x154        x155 
##          NA          NA          NA          NA          NA          NA 
##        x156        x157        x158        x159        x160        x161 
##          NA          NA          NA          NA          NA          NA 
##        x162        x163        x164        x165        x166        x167 
##          NA          NA          NA          NA          NA          NA 
##        x168        x169        x170        x171        x172        x173 
##          NA          NA          NA          NA          NA          NA 
##        x174        x175        x176        x177        x178        x179 
##          NA          NA          NA          NA          NA          NA 
```

```
##        x180        x181        x182        x183        x184        x185
##          NA          NA          NA          NA          NA          NA
##        x186        x187        x188        x189        x190        x191
##          NA          NA          NA          NA          NA          NA
##        x192        x193        x194        x195        x196        x197
##          NA          NA          NA          NA          NA          NA
##        x198        x199        x200        x201        x202        x203
##          NA          NA          NA          NA          NA          NA
##        x204        x205        x206        x207        x208        x209
##          NA          NA          NA          NA          NA          NA
##        x210
##          NA
## Degrees of Freedom: 139 Total (i.e. Null); 0 Residual
```

```
ypred <- predict.extendglmnet(fit.logistic, newx = testx, family = "binomial", type = "class")
```

```
## Warning in predict.extendglmnet(fit.logistic, newx = testx, family =
## "binomial", : prediction from a rank-deficient fit may be misleading
```

```
evaluation(testy, ypred, "accuracy")
```

```
## [1] 0.45
```

Predicting with extendedglmnet objects for binary responses

For binary responses, three types of predictions are available. Type "link" gives the linear predictors for "binomial" models. Type "response" gives the fitted probabilities. Type "class" applies only to "binomial" models, and produces the class label corresponding to the maximum probability. If no levels argument were specifies, the class will be labeled as 0 or 1 classes.

```
ypred <- predict.extendglmnet(fit.logistic, newx = testx, family = "binomial", type = "link")
```

```
## Warning in predict.extendglmnet(fit.logistic, newx = testx, family =
## "binomial", : prediction from a rank-deficient fit may be misleading
```

```
t(ypred)
```

```
##               [,1]          [,2]          [,3]          [,4]         [,5]
## [1,] 1.462429e-207 3.308667e+155 4.796829e-247 1.465324e-309 4.042111e+42
##               [,6]          [,7]          [,8]          [,9]        [,10] [,11]
## [1,] 3.513117e+119 1.573149e-151 6.295256e-116 8.545733e+136 3.691463e+13   Inf
##              [,12]         [,13]         [,14] [,15]        [,16] [,17]
## [1,] 2.40776e-310 1.015158e+139 1.887134e-218     0 5.409654e+30   Inf
##              [,18]        [,19]         [,20] [,21]         [,22]        [,23]
## [1,] 1.728859e+143 2.422424e-95 3.064792e+217     0 1.834202e-179 2.436051e+63
##              [,24]         [,25]         [,26]         [,27]         [,28]
## [1,] 9.531474e+252 2.137773e+143 9.781443e+265 1.032252e-210 3.452678e+108
##             [,29]        [,30]         [,31]         [,32] [,33]        [,34]
## [1,] 3.041048e-64 1.032382e+66 1.807103e+105 1.141086e+258   Inf 3.448341e-45
##             [,35]        [,36]        [,37]         [,38]        [,39] [,40]
```

```
## [1,] 6.629514e+84 1.608787e-138 3.288886e-103 5.443472e-161 4.295217e-29     0
##              [,41]        [,42]        [,43] [,44]         [,45]         [,46]
## [1,] 4.405457e-180 1.732937e+43 2.216706e+79     0 2.432248e-150 1.016516e-239
##              [,47]        [,48]        [,49] [,50]        [,51]        [,52]
## [1,] 8.561753e-142 1.224095e-77 7.503855e+15     0 1.380286e+51 3.161221e-45
##              [,53] [,54]         [,55]         [,56] [,57]        [,58] [,59]
## [1,] 1.299577e+183     0 2.674293e-229 2.824117e-206     0 9.044215e+24     0
##              [,60]
## [1,] 2.600173e-100
```

```
ypred <- predict.extendglmnet(fit.logistic, newx = testx, family = "binomial", type = "response")
```

```
## Warning in predict.extendglmnet(fit.logistic, newx = testx, family =
## "binomial", : prediction from a rank-deficient fit may be misleading
```

```
t(ypred)
```

```
##      [,1]          [,2] [,3] [,4]         [,5]          [,6] [,7] [,8]
## [1,]    1 3.022365e-156    1    1 2.473955e-43 2.846475e-120    1    1
##               [,9]        [,10] [,11] [,12]         [,13] [,14] [,15]
## [1,] 1.170175e-137 2.708953e-14     0     1 9.850683e-140     1     1
##              [,16] [,17]         [,18] [,19]         [,20] [,21] [,22]
## [1,] 1.848547e-31     0 5.784162e-144     1 3.262864e-218     1     1
##              [,23]         [,24]         [,25]         [,26] [,27]         [,28]
## [1,] 4.105005e-64 1.049156e-253 4.677764e-144 1.022344e-266     1 2.896303e-109
##      [,29]        [,30]        [,31]         [,32] [,33] [,34]        [,35]
## [1,]     1 9.686338e-67 5.53372e-106 8.763585e-259     0     1 1.508406e-85
##      [,36] [,37] [,38] [,39] [,40] [,41]        [,42]        [,43] [,44] [,45]
## [1,]     1     1     1     1     1     1 5.770549e-44 4.511199e-80     1     1
##      [,46] [,47] [,48]        [,49] [,50]        [,51] [,52]         [,53]
## [1,]     1     1     1 1.332648e-16     1 7.244875e-52     1 7.694813e-184
##      [,54] [,55] [,56] [,57]        [,58] [,59] [,60]
## [1,]     1     1     1     1 1.105679e-25     1     1
```

Regression with regulations: family = "binomial", type = "lasso" or "ridge"

Similarly, we can apply logistic regression with ridge or lasso regulations to the same dataset. The prediciton accuracy is the highest with lasso (0.87), following that with ridge regulation (0.75) and is the lowest for model without regulation (0.38).

```
fit.ridge<- extendedglmnet(trainx, trainy, family = "binomial", type = "ridge")
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
ypred <- predict.extendglmnet(fit.ridge, newx = testx, family = "binomial", type = "class")
evaluation(testy, ypred, "accuracy")
```

```
## [1] 0.5833333
```

```
fit.lasso<- extendedglmnet(trainx, trainy, family = "binomial", type = "lasso")
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
ypred <- predict.extendglmnet(fit.lasso, newx = testx, family = "binomial", type = "class")
evaluation(testy, ypred, "accuracy")
```

```
## [1] 0.8333333
```

Random Lasso: `family = "binomial", type = "random lasso"`

We tested random lasso model to the same dataset as well. q1 and q2 are set to be 150. The prediction is the highest with random lasso which is 0.9.

```
fit.rl<- extendedglmnet(trainx, trainy, family = "binomial", type = "random lasso")
ypred <- predict.extendglmnet(fit.rl, newx = testx, family = "binomial", type = "class")
evaluation(testy, ypred, "accuracy")
```

```
## [1] 0.8
```

References

- Wang, S., Nan, B., Rosset, S., & Zhu, J. (2011). Random lasso, The annals of applied statistics, 5(1), 468.
- Friedman J, Hastie T, Tibshirani R (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1–22. doi: 10.18637/jss.v033.i01, https://www.jstatsoft.org/v33/i01/.