`biglasso`: extending lasso model to Big Data in R

Yaohui Zeng, Patrick Breheny

Package Version: 1.2-3

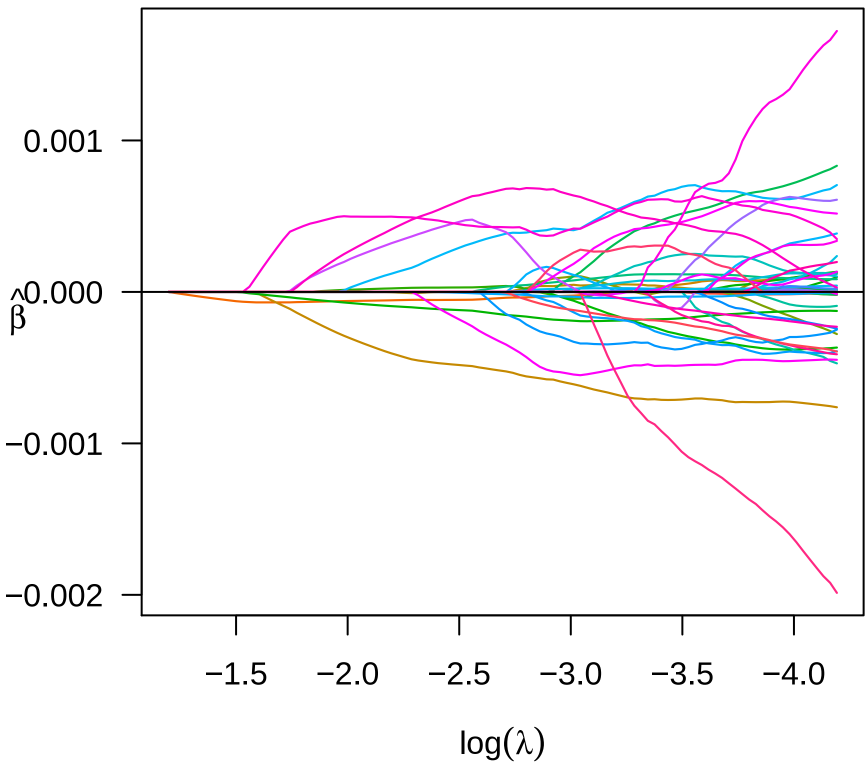
December 1, 2016

1 User guide

1.1 Small data

When the data size is small, the usage of `biglasso` package is very similar to that of `ncvreg`, except that `biglasso` requires the design matrix to be a `big.matrix` object. Below is a complete example to fit a lasso-penalized linear regression model.

```
require(biglasso)

## Loading required package:  biglasso
## Loading required package:  bigmemory
## Loading required package:  bigmemory.sri
## Loading required package:  Matrix
## Loading required package:  ncvreg

data(colon)
X <- colon$X
y <- colon$y
dim(X)

## [1]   62 2000

X[1:5, 1:5]

##   Hsa.3004 Hsa.13491 Hsa.13491.1 Hsa.37254 Hsa.541
## t  8589.42   5468.24     4263.41   4064.94 1997.89
## n  9164.25   6719.53     4883.45   3718.16 2015.22
## t  3825.71   6970.36     5369.97   4705.65 1166.55
## n  6246.45   7823.53     5955.84   3975.56 2002.61
## t  3230.33   3694.45     3400.74   3463.59 2181.42

## convert X to a big.matrix object
X.bm <- as.big.matrix(X)
str(X.bm) ## X.bm is a pointer to the data matrix

## Formal class 'big.matrix' [package "bigmemory"] with 1 slot
##   ..@ address:<externalptr>

dim(X.bm)

## [1]   62 2000

X.bm[1:5, 1:5] ## same results as X[1:5, 1:5]

##   Hsa.3004 Hsa.13491 Hsa.13491.1 Hsa.37254 Hsa.541
## t  8589.42   5468.24     4263.41   4064.94 1997.89
## n  9164.25   6719.53     4883.45   3718.16 2015.22
## t  3825.71   6970.36     5369.97   4705.65 1166.55
## n  6246.45   7823.53     5955.84   3975.56 2002.61
## t  3230.33   3694.45     3400.74   3463.59 2181.42

## fit entire solution path, using our newly proposed screening rule "SSR-BEDPP"
fit <- biglasso(X.bm, y, screen = "SSR-BEDPP")
plot(fit)
```

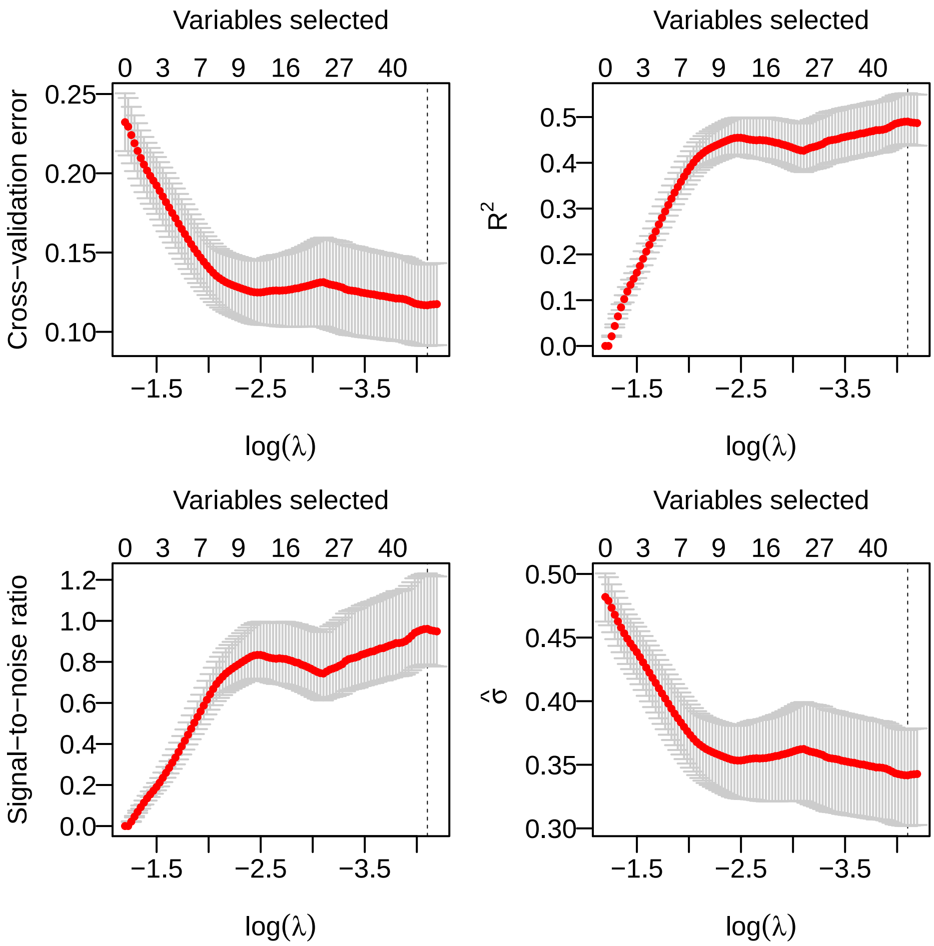



```r
## 10-fold cross-valiation in parallel
cvfit <- cv.biglasso(X.bm, y, seed = 1234, nfolds = 10, ncores = 4)
```

After cross-validation, a few things we can do:

- plot the cross-validation plots:

```r
par(mfrow = c(2, 2), mar = c(3.5, 3.5, 3, 1) ,mgp = c(2.5, 0.5, 0))
plot(cvfit, type = "all")
```



- Summarize CV object:

```
summary(cvfit)

## lasso-penalized linear regression with n=62, p=2000
## At minimum cross-validation error (lambda=0.0165):
## -------------------------------------------------
##   Nonzero coefficients: 46
##   Cross-validation error (deviance): 0.12
##   R-squared: 0.49
##   Signal-to-noise ratio: 0.96
##   Scale estimate (sigma): 0.342
```

- Extract non-zero coefficients at the optimal λ value 0.0165448:

```
coef(cvfit)[which(coef(cvfit) != 0)]

##   (Intercept)       Hsa.467      Hsa.1013.1        Hsa.832      Hsa.10358
##  7.000690e-01 -1.068388e-05 -8.774821e-06  1.469851e-05 -1.279683e-05
##      Hsa.2126    Hsa.11096.1      Hsa.36689      Hsa.16793      Hsa.10909
## -1.237284e-05  1.143071e-04 -7.435492e-04  1.141762e-04 -2.244924e-04
##      Hsa.8010     Hsa.1920.1       Hsa.9972      Hsa.692.2       Hsa.7852
##  1.905808e-05  5.447871e-05  1.168477e-04 -1.243593e-04  1.583372e-05
##      Hsa.1272       Hsa.166       Hsa.1127      Hsa.31801        Hsa.579
## -3.744562e-04  7.753436e-04 -1.255786e-05  8.553825e-05 -9.841401e-05
##     Hsa.24877      Hsa.3648       Hsa.1047      Hsa.13628       Hsa.1509
```

```
## -4.167749e-04  1.247990e-04  6.261196e-06  1.151005e-04  1.458936e-04
##      Hsa.3016      Hsa.5392     Hsa.16622      Hsa.1832     Hsa.12241
##  3.773071e-05  6.565699e-04  3.536769e-04 -7.666380e-06 -4.027543e-04
##     Hsa.44244      Hsa.9103      Hsa.2964      Hsa.1140      Hsa.9353
## -2.844216e-04 -2.123102e-04  2.624890e-05  6.996317e-06  6.055567e-04
##       Hsa.127     Hsa.41159     Hsa.33268      Hsa.2012     Hsa.34937
##  1.041546e-04  5.319337e-04 -4.489856e-04  3.107549e-04  1.578863e-03
##      Hsa.6814      Hsa.1660       Hsa.404     Hsa.36161      Hsa.1185
##  4.404346e-04  9.230497e-05 -2.166536e-04  1.749590e-04 -3.892690e-04
##     Hsa.43331   Hsa.41098.1
## -1.822198e-03 -3.690431e-04
```

1.2 Big data

When the raw data file is very large, it's better to convert the raw data file into a file-backed `big.matrix` by using a file cache. We can call function `setupX`, which reads the raw data file and creates a backing file (.bin) and a descriptor file (.desc) for the raw data matrix:

```
# Note: (1) simulated data, 1000 observations, 100,000 variables,
#       (2) the first 10 variables have non-zero coefficient 2.
xfname <- 'x_e3_e5.txt' # raw data file for design matrix, ~ 1GB
time <- system.time(
  X <- setupX(xfname, sep = '\t') # create backing files (.bin, .desc)
)

## Reading data from file, and creating file-backed big.matrix...
## This should take a while if the data is very large...
## Start time:  2016-12-16 12:49:16
## End time:  2016-12-16 12:50:51
## DONE!
##
## Note: This function needs to be called only one time to create two backing
##       files (.bin, .desc) in current dir. Once done, the data can be
##       'loaded' using function 'attach.big.matrix'. See details in doc.

print(time)

##    user  system elapsed
##  71.866   3.370  95.206

dim(X)

## [1] 1e+03 1e+05

X[1:5, 1:5]

##            [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  1.601592 -0.259093  0.174768 -1.498961 -0.302023
## [2,] -0.637744 -0.095101 -0.317369  1.248830 -0.712442
## [3,] -0.231440 -0.106024  0.799767  0.536773 -0.695111
## [4,]  0.842769  0.659977 -0.148627  0.149582  1.597956
## [5,] -0.356504 -0.718464 -0.581049  0.201162  0.392043

object.size(X) # X is merely a pointer. The data is stored on the disk!

## 664 bytes
```

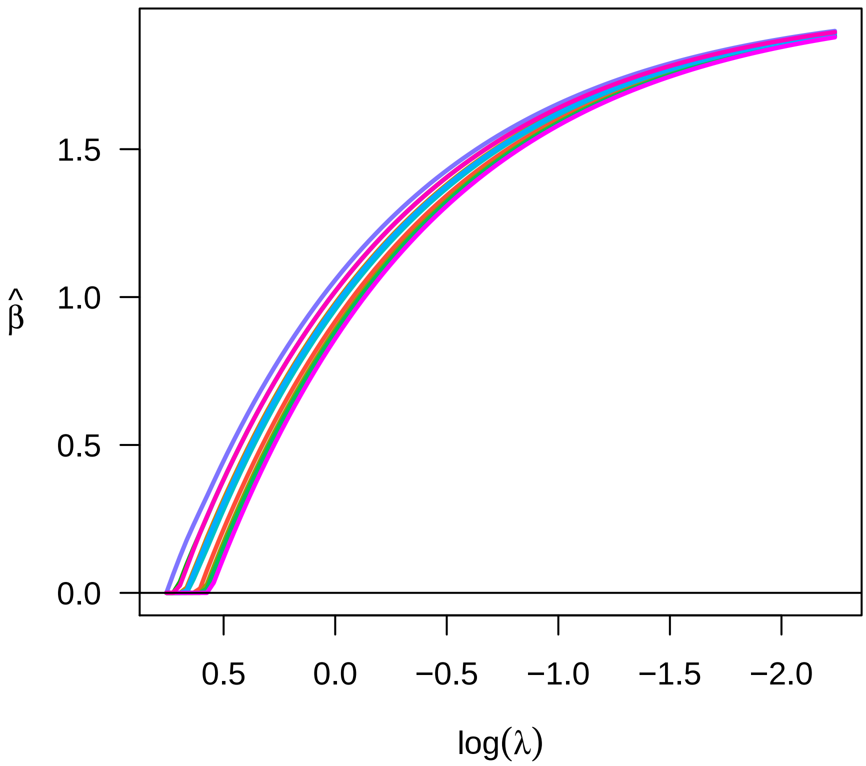
It's important to note that the above operation is just one-time execution. Once done, the data can always be retrieved seamlessly by attaching its descriptor file (.desc) in any new R session:

```r
rm(list = ls()) # start a new session
xdesc <- 'x_e3_e5.desc'
system.time(X <- attach.big.matrix(xdesc))

##    user  system elapsed
##   0.001   0.000   0.001

dim(X)

## [1] 1e+03 1e+05

X[1:5, 1:5]

##            [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  1.601592 -0.259093  0.174768 -1.498961 -0.302023
## [2,] -0.637744 -0.095101 -0.317369  1.248830 -0.712442
## [3,] -0.231440 -0.106024  0.799767  0.536773 -0.695111
## [4,]  0.842769  0.659977 -0.148627  0.149582  1.597956
## [5,] -0.356504 -0.718464 -0.581049  0.201162  0.392043

object.size(X)

## 664 bytes
```

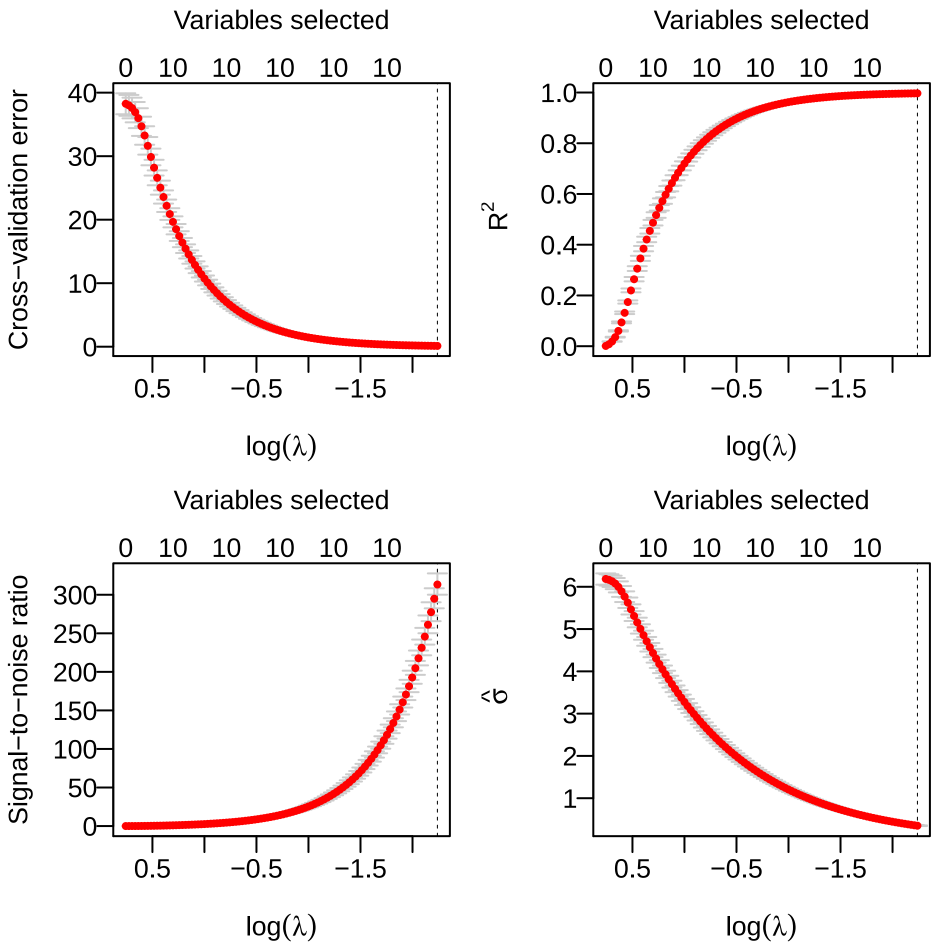
This is very appealing for big data analysis in that we don't need to "read" the raw data again in a R session, which would be very time-consuming.

The code below again fits a lasso-penalized linear model, and runs 10-fold cross-validation:

```r
yfname <- 'y_e3_e5.txt' # response vector
y <- as.matrix(read.table(yfname, header = F))
time.fit <- system.time(
  fit <- biglasso(X, y, family = 'gaussian', screen = 'SSR-BEDPP')
)
print(time.fit)

##    user  system elapsed
##   9.473   0.138   9.622

plot(fit)
```




```r
# 10-fold cross validation in parallel
time.cvfit <- system.time(
  cvfit <- cv.biglasso(X, y, screen = 'SSR-BEDPP', seed = 1234, ncores = 4, nfolds = 10)
)
print(time.cvfit)

##    user  system elapsed
##   9.602   0.040  45.574

par(mfrow = c(2, 2), mar = c(3.5, 3.5, 3, 1), mgp = c(2.5, 0.5, 0))
plot(cvfit, type = "all")
```




```
summary(cvfit)

## lasso-penalized linear regression with n=1000, p=1e+05
## At minimum cross-validation error (lambda=0.1065):
## -------------------------------------------------
##   Nonzero coefficients: 10
##   Cross-validation error (deviance): 0.12
##   R-squared: 1.00
##   Signal-to-noise ratio: 313.30
##   Scale estimate (sigma): 0.349

coef(cvfit)[which(coef(cvfit) != 0)]

## (Intercept)          V1          V2          V3          V4          V5
##   0.0284291   1.8846876   1.8912635   1.8818264   1.8955174   1.8808297
##          V6          V7          V8          V9         V10
##   1.8889880   1.8905549   1.8996113   1.8785133   1.8955111
```

2 Useful references

- `biglasso` R manual: https://cran.rstudio.com/web/packages/biglasso/biglasso.pdf
- `biglasso` on GitHub for benchmarking experiments: https://github.com/YaohuiZeng/biglasso
- `big.matrix` manipulation: https://cran.r-project.org/web/packages/bigmemory/index.html