

# Package ‘scout’

October 13, 2009

**Type** Package

**Title** Implements the Scout method for Covariance-Regularized Regression

**Version** 1.0.1

**Date** 2008-11-20

**Author** Daniela M. Witten and Robert Tibshirani

**Maintainer** Daniela M. Witten <dwitten@stanford.edu>

**Depends** R (>= 1.9.0), glasso, lars

**Description** Implements the Scout method for regression, described in “Covariance-regularized regression and classification for high-dimensional problems”, by Witten and Tibshirani (2008), Journal of the Royal Statistical Society, Series B 71(3): 615-636.  
<<http://www-stat.stanford.edu/~dwitten>>

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2009-10-13 06:37:13

## R topics documented:

|                               |   |
|-------------------------------|---|
| scout-package . . . . .       | 2 |
| crossProdLasso . . . . .      | 3 |
| cv.scout . . . . .            | 4 |
| predict.scoutobject . . . . . | 6 |
| scout . . . . .               | 8 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>10</b> |
|--------------|-----------|

---

`scout-package`*Implements covariance-regularized regression, aka the Scout Method.*

---

**Description**

Functions for implementing covariance-regularize regression.

**Details**

Package: scout  
Type: Package  
Version: 1.0  
Date: 2008-11-20  
License: GPL (>=2)

The main function is "scout", which takes in a data matrix  $x$  and an outcome vector  $y$  and estimates regression coefficients for Scout(2,1) for a range of tuning parameter values. Alternatively one can specify other tuning parameter values and one can also perform Scout(1,1), Scout(2,.), or Scout(1,.). Cross-validation and prediction functions also are available.

**Author(s)**

Daniela Witten and Robert Tibshirani

Maintainer: Daniela Witten <dwitten@stanford.edu>

**References**

Witten and Tibshirani (2008) Covariance-regularized regression and classification for high-dimensional problems. *Journal of the Royal Statistical Society, Series B* 71(3): 615-636.

**See Also**

<<http://www-stat.stanford.edu/~dwitten>>

**Examples**

```
data(diabetes)
attach(diabetes)
cv.out <- cv.scout(x2,y,p1=1,p2=1)
print(cv.out)
out <- scout(x2,y,p1=1,p2=1,lam1=cv.out$bestlam1,lam2=cv.out$bestlam2)
coef <- out$coef[1,1,]
detach(diabetes)
```

---

crossProdLasso      *Performs the lasso on the cross product matrices  $X'X$  and  $X'y$*

---

### Description

Perform L1-regularized regression of  $y$  onto  $X$  using only the cross-product matrices  $X'X$  and  $X'y$ . In the case of covariance-regularized regression, this is useful if you would like to try out something other than L1 or L2 regularization of the inverse covariance matrix.

Suppose you use your own method to regularize  $X'X$ . Then let  $\Sigma$  denote your estimate of the population covariance matrix. Now say you want to minimize  $\beta' \Sigma \beta - 2 \beta' X'y + \lambda \|\beta\|_1$  in order to get the regression estimate  $\beta$ , which maximizes the second scout criterion when an L1 penalty is used. You can do this by calling `crossProdLasso( $\Sigma$ ,  $X'y$ ,  $\rho$ )`.

If you run `crossProdLasso( $X'X$ ,  $X'y$ ,  $\rho$ )` then it should give the same result as `lars( $X$ ,  $y$ )`

### Usage

```
crossProdLasso(xtx, xty, rho, thr=1e-4, maxit=100, beta.init=NULL)
```

### Arguments

|                        |   |
|------------------------|---|
| <code>xtx</code>       | A $p \times p$ matrix, which should be an estimate of a covariance matrix.  |
| <code>xty</code>       | A $p \times 1$ vector, which is generally obtained via $X'y$ .  |
| <code>rho</code>       | Must be non-negative; the regularization parameter you are using.   |
| <code>thr</code>       | Convergence threshold.  |
| <code>maxit</code>     | How many iterations to perform?   |
| <code>beta.init</code> | If you're running this over a range of $\rho$ values, then set <code>beta.init</code> equal to the solution you got for a previous $\rho$ value. It will speed things up. |

### Details

If your `xtx` is simply  $X'X$  for some  $X$ , and your `xty` is simple  $X'y$  with some  $y$ , then the results will be the same as running `lars` on data  $(X, y)$  for a single shrinkage parameter value.

Note that when you use the `scout` function with `p2=1`, the `crossProdLasso` function is called internally to give the regression coefficients, after the regularized inverse covariance matrix is estimated. It is provided here in case it is useful to the user in other settings.

### Value

|                   |   |
|-------------------|---|
| <code>beta</code> | A $p \times 1$ vector with the regression coefficients. |
|-------------------|---|

### Note

The FORTRAN code that this function links to was kindly written and provided by Jerry Friedman.

**Author(s)**

FORTRAN code by Jerry Friedman. R interface by Daniela M. Witten and Robert Tibshirani

**References**

Witten, DM and Tibshirani, R (2008) Covariance-regularized regression and classification for high-dimensional problems. *Journal of the Royal Statistical Society, Series B* 71(3): 615-636. <<http://www-stat.stanford.edu/~dwitten>>

**Examples**

```
data(diabetes)
attach(diabetes)
# First, let's do scout(2,1) the usual way).
scout.out <- scout(x2,y,p1=2,p2=1)
print(scout.out)

# Now, suppose I want to do develop a covariance-regularized regression
# method as in Section 3.2 of Witten and Tibshirani (2008). It will work
# like this:
# 1. Estimate Sigma by hard-thresholding X'X
# 2. Find \beta by minimize \beta^T \Sigma \beta - 2 \beta^T X^T y +
# \lambda ||\beta||_1
# 3. Re-scale \beta.

# Step 1:
regcovx <- cov(x2)*(abs(cov(x2))>.05)
# Step 2:
betahat <- crossProdLasso(regcovx, cov(x2,y), rho=.02)$beta
# Step 3:
betahat.sc <- betahat*lsfit(x2*%betahat, y, intercept=FALSE)$coef
print(betahat.sc)

# Try a different value of rho:
betahat2 <- crossProdLasso(regcovx,cov(x2,y),rho=.04,beta.init=betahat)$beta
plot(betahat,betahat2, xlab="rho=.02",ylab="rho=.04")
detach(diabetes)
```

---

cv.scout

*Perform cross-validation for covariance-regularized regression, aka the Scout.*

---

**Description**

This function returns cross-validation error rates for a range of lambda1 and lambda2 values, and also makes beautiful CV plots if plot=TRUE.

**Usage**

```
cv.scout(x, y, K= 10, lam1s=seq(0.001, .2, len=10), lam2s=seq(0.001, .2, len=10), p1=2, p2
```

**Arguments**

|         |   |
|---------|---|
| x       | A matrix of predictors, where the rows are the samples and the columns are the predictors   |
| y       | A matrix of observations, where length(y) should equal nrow(x)  |
| K       | Number of cross-validation folds to be performed; default is 10   |
| lam1s   | The (vector of) tuning parameters for regularization of the covariance matrix. Can be NULL if p1=NULL, since then no covariance regularization is taking place. If p1=1 and nrow(x)<ncol(x), then the no value in lam1s should be smaller than 1e-3, because this will cause graphical lasso to take too long. Also, if ncol(x)>500 then we really do not recommend using p1=1, as graphical lasso can be uncomfortably slow. |
| lam2s   | The (vector of) tuning parameters for the $L_1$ regularization of the regression coefficients, using the regularized covariance matrix. Can be NULL if p2=NULL. (If p2=NULL, then non-zero lam2s have no effect). A value of 0 will result in no regularization.  |
| p1      | The $L_p$ penalty for the covariance regularization. Must be one of 1, 2, or NULL. NULL corresponds to no covariance regularization.  |
| p2      | The $L_p$ penalty for the estimation of the regression coefficients based on the regularized covariance matrix. Must be one of 1 (for $L_1$ regularization) or NULL (for no regularization).  |
| trace   | Print out progress as we go? Default is TRUE.   |
| plot    | If TRUE (by default), makes beautiful CV plots.   |
| plotSE  | Should those beautiful CV plots also display std error bars for the CV? Default is FALSE  |
| rescale | Scout rescales coefficients, by default, in order to avoid over-shrinkage   |
| ...     | Additional parameters   |

**Details**

Pass in a data matrix x and a vector of outcomes y; it will perform (10-fold) cross-validation over a range of lambda1 and lambda2 values. By default, Scout(2,1) is performed.

**Value**

|          |   |
|----------|---|
| folds    | The indices of the members of the K test sets are returned.   |
| cv       | A matrix of average cross-validation errors is returned.  |
| cv.error | A matrix containing the standard errors of the elements in "cv", the matrix of average cross-validation errors. |
| bestlam1 | Best value of lam1 found via cross-validation.  |
| bestlam2 | Best value fo lam2 found via cross-validation.  |
| lam1s    | Values of lam1 considered.  |
| lam2s    | Values of lam2 considered.  |

**Author(s)**

Daniela M. Witten and Robert Tibshirani

**References**

Witten, DM and Tibshirani, R (2008) Covariance-regularized regression and classification for high-dimensional problems. *Journal of the Royal Statistical Society, Series B* 71(3): 615-636. <<http://www-stat.stanford.edu/~dwitten>>

**See Also**

[scout](#), [predict.scoutobject](#)

**Examples**

```
data(diabetes)
attach(diabetes)
par(mfrow=c(2,1))
par(mar=c(2,2,2,2))
cv.sc <- cv.scout(x2,y,p1=2,p2=1)
print(cv.sc)
cv.la <- cv.lars(x2,y)
print(c("Lars minimum CV is ", min(cv.la$cv)))
print(c("Scout (2,1) minimum CV is ", min(cv.sc$cv)))
detach(diabetes)
```

---

```
predict.scoutobject
```

*Prediction function for covariance-regularized regression, aka the Scout.*

---

**Description**

A function to perform prediction, using an x matrix and the output of the "scout" function.

**Usage**

```
predict.scoutobject(object, newx, ...)
```

**Arguments**

|        |   |
|--------|---|
| object | The results of a call to the "scout" function. The coefficients that are part of this object will be used for making predictions.                                       |
| newx   | The new x at which predictions should be made. Can be a vector of length ncol(x), where x is the data on which scout.obj was created, or a matrix with ncol(x) columns. |
| ...    | Additional arguments to predict   |

**Value**

yhats            If `newx` was a vector, then a matrix will be returned, with dimension `length(lam1s) × length(lam2s)` (where `lam1s` and `lam2s` are attributes of `scout.obj`). The  $(i,j)$  element of this matrix will correspond to tuning parameter values  $(\text{lam1s}[i], \text{lam2s}[j])$ . If `newx` is a matrix, then an array of dimension `nrow(newx) × length(lam1s) × length(lam2s)` will be returned.

**Author(s)**

Daniela M. Witten and Robert Tibshirani

**References**

Witten, DM and Tibshirani, R (2008) Covariance-regularized regression and classification for high-dimensional problems. *Journal of the Royal Statistical Society, Series B* 71(3): 615-636. <<http://www-stat.stanford.edu/~dwitten>>

**See Also**

[scout](#), [cv.scout](#)

**Examples**

```
data(diabetes)
attach(diabetes)
# Split data into training and test set
training <- sample(nrow(x2), floor(nrow(x2)/2))
xtrain <- x2[training,]
ytrain <- y[training]
xtest <- x2[-training,]
ytest <- y[-training]
# Done splitting data into training and test set
# Do cross-validation to determine best tuning parameter values for Scout(1,1)
cv.out <- cv.scout(xtrain,ytrain,p1=1,p2=1, lam1s=seq(0.001,.15,len=10))
print(cv.out)
# Done cross-validation
# Fit Model
scout.object <- scout(xtrain,ytrain,p1=1,p2=1,lam1s=cv.out$bestlam1,lam2s=cv.out$bestlam2)
print(scout.object)
# Done Fitting Model
# Predict on test data, and report MSE
yhats <- predict(scout.object,xtest)
print(mean((yhats-ytest)^2))
detach(diabetes)
```

scout

*Covariance-regularized regression, aka the Scout.***Description**

The main function of the "scout" package. Performs covariance-regularized regression. Required inputs are an  $x$  matrix of features (the columns are the features) and a  $y$  vector of observations. By default, `Scout(2,1)` is performed; however,  $p_1$  and  $p_2$  can be specified (in which case `Scout( $p_1$ ,  $p_2$ )` is performed). Also, by default `Scout` is performed over a grid of  $\lambda_1$  and  $\lambda_2$  values, but a different grid of values (or individual values, rather than an entire grid) can be specified.

**Usage**

```
scout(x, y, newx, p1=2, p2=1, lam1s=seq(.001, .2, len=10), lam2s=seq(.001, .2, len=10), rescale=TRUE)
```

**Arguments**

|                      |   |
|----------------------|---|
| <code>x</code>       | A matrix of predictors, where the rows are the samples and the columns are the predictors   |
| <code>y</code>       | A matrix of observations, where <code>length(y)</code> should equal <code>nrow(x)</code>  |
| <code>newx</code>    | An <i>optional</i> argument, consisting of a matrix with <code>ncol(x)</code> columns, at which one wishes to make predictions for each ( $\lambda_1, \lambda_2$ ) pair.  |
| <code>p1</code>      | The $L_p$ penalty for the covariance regularization. Must be one of 1, 2, or NULL. NULL corresponds to no covariance regularization. <b>WARNING:</b> When $p_1=1$ , and <code>ncol(x)&gt;500</code> , <code>Scout</code> can be SLOW. We recommend that for very large data sets, you use <code>Scout</code> with $p_1=2$ . Also, when <code>ncol(x)&gt;nrow(x)</code> and $p_1=1$ , then very small values of $\lambda_1$ ( $\lambda_1 < 1e-4$ ) will cause problems with graphical lasso, and so those values will be automatically increased to $1e-4$ . |
| <code>p2</code>      | The $L_p$ penalty for the estimation of the regression coefficients based on the regularized covariance matrix. Must be one of 1 (for $L_1$ regularization) or NULL (for no regularization).  |
| <code>lam1s</code>   | The (vector of) tuning parameters for regularization of the covariance matrix. Can be NULL if $p_1=$ NULL, since then no covariance regularization is taking place. If $p_1=1$ and <code>nrow(x)&lt;ncol(x)</code> , then the no value in <code>lam1s</code> should be smaller than $1e-3$ , because this will cause graphical lasso to take too long. Also, if <code>ncol(x)&gt;500</code> then we really do not recommend using $p_1=1$ , as graphical lasso can be uncomfortably slow.   |
| <code>lam2s</code>   | The (vector of) tuning parameters for the $L_1$ regularization of the regression coefficients, using the regularized covariance matrix. Can be NULL if $p_2=$ NULL. (If $p_2=$ NULL, then non-zero <code>lam2s</code> have no effect). A value of 0 will result in no regularization.   |
| <code>rescale</code> | Should coefficients $\beta$ obtained by covariance-regularized regression be re-scaled by a constant, given by regressing $y$ onto $x\beta$ ? This is done in Witten and Tibshirani (2008) and is important for good performance. Default is TRUE.  |

|                          |   |
|--------------------------|---|
| <code>trace</code>       | Print out progress? Prints out each time a <code>lambda1</code> is completed. This is a good idea, especially when <code>ncol(x)</code> is large.   |
| <code>standardize</code> | Should the columns of <code>x</code> be scaled to have standard deviation 1, and should <code>y</code> be scaled to have standard deviation 1, before covariance-regularized regression is performed? This affects the meaning of the penalties that are applied. In general, standardization should be performed. Default is <code>TRUE</code> . |

**Value**

|                           |   |
|---------------------------|---|
| <code>intercepts</code>   | Returns a matrix of intercepts, of dimension <code>length(lam1s)xlength(lam2s)</code>             |
| <code>coefficients</code> | Returns an array of coefficients, of dimension <code>length(lam1s)xlength(lam2s)xncol(x)</code> . |
| <code>p1</code>           | <code>p1</code> value used  |
| <code>p2</code>           | <code>p2</code> value used  |
| <code>lam1s</code>        | <code>lam1s</code> used   |
| <code>lam2s</code>        | <code>lam2s</code> used   |

**Note**

When `p1=1` and `ncol(x)>500` or so, then Scout can be very slow!! Please use `p1=2` when `ncol(x)` is large.

**Author(s)**

Daniela M. Witten and Robert Tibshirani

**References**

Witten, DM and Tibshirani, R (2008) Covariance-regularized regression and classification for high-dimensional problems. *Journal of the Royal Statistical Society, Series B* 71(3): 615-636. <<http://www-stat.stanford.edu/~dwitten>>

**See Also**

[predict.scoutobject](#), [cv.scout](#)

**Examples**

```
data(diabetes)
attach(diabetes)
scout.out <- scout(x2,y,p1=2,p2=1)
print(scout.out)
detach(diabetes)
```

# Index

\*Topic **package**

scout-package, [1](#)

crossProdLasso, [2](#)

cv.scout, [4](#), [7](#), [9](#)

predict.scoutobject, [6](#), [6](#), [9](#)

scout, [6](#), [7](#), [7](#)

scout-package, [1](#)