Package ‘lpc’

January 4, 2019

Type Package

Title Lassoed Principal Components for Testing Significance of Features

Version 1.0.2.1

Date 2013-12-15

Author Daniela M Witten and Robert Tibshirani

Maintainer Daniela M Witten <dwitten@uw.edu>

Description Implements the LPC method of Witten&Tibshirani(Annals of Applied Statistics 2008) for identification of significant genes in a microarray experiment.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2019-01-04 17:06:05 UTC

R topics documented:

`lpc-package` *This package implements the Lassoed Principal Components (LPC) method. It is used to compute LPC scores for each gene in a microrray experiment with a survival, quantitative, or two-class outcome.*

Description

LPC scores are computed for each gene; the method borrows strength across genes and can result in more accurate gene scores than simpler statistics. In this package, the LPC method is applied by regressing Cox scores (survival outcome), two-sample t-statistics (two-class outcome), or standardized regression coefficients (quantitative outcome) onto gene expression eigenarrays, with an L1 constraint.

Details

Package:	lpc
Type:	Package
Version:	1.0.2
Date:	2013-12-15
License:	GPL (>= 2)

The main function is "LPC", which computes LPC scores for each gene. The matrix of gene expression data, a vector of outcome, and the outcome type must be passed in to this function.

Author(s)

Daniela M. Witten and Robert Tibshirani

Maintainer: Daniela M. Witten <dwitten@uw.edu>

References

Witten, DM and Tibshirani R (2008) Testing significance of features by lassoed principal components. Annals of Applied Statistics.

See Also

www.biostat.washington.edu/~dwitten

Examples

```
### Uncomment to run


#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) # quantitative outcome
```

```
## make first 50 genes differentially-expressed
#x[1:25,y<(-.5)] <- x[1:25,y<(-.5)]+ 1.5
#x[26:50,y<0] <- x[26:50,y<0] - 1.5
## compute LPC and T scores for each gene
#lpc.obj <- LPC(x,y, type="regression")
#lpc.obj
## Look at plot of Predictive Advantage
#pred.adv <- PredictiveAdvantage(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh)
## Estimate FDRs for LPC and T scores
#fdr.lpc.out <- EstimateLPCFDR(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh,nreps=50)
#fdr.lpc.out
```

`EstimateLPCFDR` *Estimate LPC False Discovery Rates.*

Description

An estimated false discovery rate for each gene is computed, based on the LPC scores. Estimated false discovery rates based on T scores (Cox, regression coefficient, or two-sample t-statistic) are also given.

Usage

```
EstimateLPCFDR(x, y, type, nreps=100,soft.thresh=NULL, censoring.status=NULL)
```

Arguments

`x` The matrix of gene expression values; pxn where n is the number of observations and p is the number of genes.

`y` A vector of length n, with an outcome for each observation. For two-class outcome, y's elements are 1 or 2. For quantitative outcome, y's elements are real-valued. For survival data, y indicates the survival time. For multiclass, y is coded as 1,2,3,...

`type` One of "regression" (for a quantitative outcome), "two class", "multiclass", or "survival".

`nreps` The number of training/test set splits used to estimate LPC's false discovery rates. Default is 100.

`soft.thresh` The value of the soft threshold to be used in the L1 regression of the scores onto the eigenarrays. This can be the value of the soft-threshold chosen adaptively by LPC function when it is run on the data; if not entered by the user, then that adaptive value is computed.

`censoring.status` For survival outcome only, a vector of length n which takes on values 0 or 1 depending on whether the observation is complete or censored.

Details

Details of false discovery rate estimation for LPC can be found in the paper: http://www-stat.stanford.edu/~dwitten \

As explained in the paper, FDR of LPC is estimated by computing the FDR of simpler scores (Cox for survival outcome, standardized regression coefficients for regression outcome, and two-sample t-statistic for two-class outcome) and then estimating the difference between the FDR of LPC and the FDR of these simpler scores.

Value

`fdrlpc`	A vector of length p (equal to the number of genes), with the LPC false discovery rate for each gene.
`fdrt`	A vector of length p (equal to the number of genes), with the T false discovery rate for each gene.
`fdrdiff`	The FDR of T minus the FDR of LPC. This is approximately equal to fdrt-fdrlpc, with the caveat that the FDR of LPC (computed via this difference) must be between 0 and 1.
`pi0`	The fraction of genes that are believed to be null.
`soft.thresh`	The value of the soft threshold used in the L1 constraint for LPC.
`call`	The function call made.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten, D.M. and Tibshirani, R. (2008) Testing significance of features by lassoed principal components. Annals of Applied Statistics. http://www-stat.stanford.edu/~dwitten

Examples

```
### Uncomment to run....

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) # quantitative outcome
## make first 50 genes differentially-expressed
#x[1:25,y<(-.5)] <- x[1:25,y<(-.5)]+ 1.5
#x[26:50,y<0] <- x[26:50,y<0] - 1.5
## compute LPC and T scores for each gene
#lpc.obj <- LPC(x,y, type="regression")
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh)
## Estimate FDRs for LPC and T scores
```

```
#fdr.lpc.out <-
#EstimateLPCFDR(x,y,type=”regression”,soft.thresh=lpc.obj$soft.thresh,nreps=50)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type=”regression”)
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC on 6% of genes. In this example, LPC has
##    lower FDR.
#PlotFDRs(fdr.lpc.out,frac=.06)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)

# Now, repeating everything that we did before, but using a
#   **survival** outcome
# NOT RUNNING DUE TO TIME CONSTRAINTS -- UNCOMMENT TO RUN


#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) + 10 # survival times; must be positive
## censoring outcome: 0 or 1
#cens <- rep(1,40) # Assume all observations are complete
## make first 50 genes differentially-expressed
#x[1:25,y<9.5] <- x[1:25,y<9.5]+ 1.5
#x[26:50,y<10] <- x[26:50,y<10] - 1.5
#lpc.obj <- LPC(x,y, type=”survival”, censoring.status=cens)
## Look at plot of Predictive Advantage
#pred.adv <- PredictiveAdvantage(x,y,type=”survival”,soft.thresh=lpc.obj$soft.thresh,
#censoring.status=cens)
## Estimate FDRs for LPC scores and T scores
#fdr.lpc.out <- EstimateLPCFDR(x,y,type=”survival”,
#soft.thresh=lpc.obj$soft.thresh,nreps=20,censoring.status=cens)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type=”survival”, censoring.status=cens)
## print out results of main function
#lpc.obj
## print out info about T FDRs
```

```
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC scores on 10% of genes.
#PlotFDRs(fdr.lpc.out,frac=.1)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)
```

`EstimateTFDR` *Estimate T False Discovery Rates.*

Description

An estimated false discovery rate for each gene is computed, based on the T scores. The T scores are as follows, for two-class, survival, and quantitative outcomes: two-sample t-statistics, Cox scores, standardized regression coefficients. The output of this function is identical to the outputs "fdrt" and "pi0" of the function EstimateLPCFDR. This function should be used if only the FDR of T is desired, because computing the FDR of LPC is time-consuming.

Usage

```
EstimateTFDR(x,y, type,censoring.status=NULL)
```

Arguments

`x` The matrix of gene expression values; pxn where n is the number of observations and p is the number of genes.

`y` A vector of length n, with an outcome for each observation. For two-class outcome, y's elements are 1 or 2. For quantitative outcome, y's elements are real-valued. For survival data, y indicates the survival time. For a multiclass outcome, y is coded as 1,2,3,...

`type` One of "regression" (for a quantitative outcome), "two class", "survival", or "multiclass" (for a multiple-class outcome).

`censoring.status` For survival outcome only, a vector of length n which takes on values 0 or 1 depending on whether the observation is complete or censored.

Details

False discovery rates are estimated by permutations, as in e.g. Tusher et al (2001) and Storey & Tibshirani (2003).

Value

fdrt	A vector of length p (equal to the number of genes), with the T false discovery rate for each gene. Note that this is identical to the "fdrt" output by the function EstimateLPCFDR.
pi0	The fraction of genes that are believed to be null.
call	The function call made.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Storey, J.D. and Tibshirani, R. (2003) Statistical significance for genomewide studies. Proceedings of the National Academy of Sciences. 100(16): 9440-9445.

Tusher, V.G. and Tibshirani, R. and Chu, G. (2001) Significance analysis of microarrays applied to the ionizing radiation response. Proceedings of the National Academy of Sciences. 98(9): 5116-5121.

Witten, D.M. and Tibshirani, R. (2008) Testing significance of features by lassoed principal components. Annals of Applied Statistics. http://www-stat.stanford.edu/~dwitten

Examples

```
### not running due to timing - uncomment to run

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) # quantitative outcome
## make first 50 genes differentially-expressed
#x[1:25,y<(-.5)] <- x[1:25,y<(-.5)]+ 1.5
#x[26:50,y<0] <- x[26:50,y<0] - 1.5
## compute LPC and T scores for each gene
#lpc.obj <- LPC(x,y, type="regression")
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh)
## Estimate FDRs for LPC and T scores
#fdr.lpc.out <-
#EstimateLPCFDR(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh,nreps=50)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="regression")
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
```

```
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC on 6% of genes. In this example, LPC has
##    lower FDR.
#PlotFDRs(fdr.lpc.out,frac=.06)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)

# Now, repeating everything that we did before, but using a
#   **survival** outcome
# NOT RUNNING DUE TO TIMING -- UNCOMMENT TO RUN

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) + 10 # survival times; must be positive
## censoring outcome: 0 or 1
#cens <- rep(1,40) # Assume all observations are complete
## make first 50 genes differentially-expressed
#x[1:25,y<9.5] <- x[1:25,y<9.5]+ 1.5
#x[26:50,y<10] <- x[26:50,y<10] - 1.5
#lpc.obj <- LPC(x,y, type=”survival”, censoring.status=cens)
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type=”survival”,soft.thresh=lpc.obj$soft.thresh,
#censoring.status=cens)
## Estimate FDRs for LPC scores and T scores
#fdr.lpc.out <- EstimateLPCFDR(x,y,
#type=”survival”,soft.thresh=lpc.obj$soft.thresh,
#nreps=20,censoring.status=cens)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type=”survival”, censoring.status=cens)
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC scores on 10% of genes.
#PlotFDRs(fdr.lpc.out,frac=.1)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
```

```
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)
```

LPC *Compute LPC score for each gene*

Description

This is the main function for the Lassoed Principal Components (lpc) method. Given a matrix of gene expression data and a vector of outcome measurements (one per patient) and an outcome type, this function computes the LPC score for each gene. Simple scores are projected onto the eigenarrays (with an L1 constraint) in order to obtain LPC scores; these simple scores are Cox scores (for a survival outcome), two-sample t-statistics (for a two-class outcome) or standardized regression coefficients (for a regression/quantitative outcome).

Usage

```
LPC(x, y, type = "regression", soft.thresh = NULL, u =
NULL, censoring.status = NULL)
```

Arguments

`x` The matrix of gene expression values; pxn where n is the number of observations and p is the number of genes.

`y` A vector of length n, with an outcome for each observation. For two-class outcome, y's elements are 1 or 2. For quantitative outcome, y's elements are real-valued. For survival data, y indicates the survival time, and must be positive. For multiclass outcome, y must be coded as 1,2,3,...

`type` One of "regression" (for a quantitative outcome), "two class", "multiclass", or "survival".

`soft.thresh` The value of the soft threshold to be used in the L1 regression of the scores onto the eigenarrays. If NULL, then it will be chosen adaptively.

`u` The eigenarrays of the data matrix (useful if the data matrix is large, to avoid having to re-compute the SVD in order to run this function multiple times). If NULL, then the eigenarrays will be computed internally. Note that the eigenarrays have the same dimension as the data.

`censoring.status`
For survival outcome only, a vector of length n which takes on values 0 or 1 depending on whether the observation is complete or censored.

Details

Lassoed Principal Components (LPC) is a method for identifying features associated with an outcome, under the hypothesis that significant features occur in correlated sets. In particular, the method was designed for identifying genes in a microarray experiment that are associated with some outcome (e.g.: cancer vs. normal, survival time, tumor size). The method involves projecting simpler scores for each gene (e.g.: two-sample t-statistic, Cox score, standardized regression coefficients) onto the eigenarrays of the gene expression data matrix, subject to an L1 constraint. The fitted values are the LPC scores, which may result in a more accurate gene ranking than the initial simpler scores.

Value

`lpcscores` A vector of LPC scores, of length p (one for each gene).

`tscores` A vector of T scores (Cox, two-sample t-stats, or standardized regression coefficients), of length p (one for each gene).

`soft.thresh` The value of the soft threshold used.

`coefs` The coefficients for the L1 regression of the scores onto the eigenarrays.

`call` The call that was made to this function.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten, D.M. and Tibshirani, R. (2008) Testing significance of features by lassoed principal components. Annals of Applied Statistics. http://www-stat.stanford.edu/~dwitten

Examples

```
set.seed(2)
n <- 40 # 40 samples
p <- 1000 # 1000 genes
x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
y <-  rnorm(n) # quantitative outcome
# make first 50 genes differentially-expressed
x[1:25,y<(-.5)] <- x[1:25,y<(-.5)]+ 1.5
x[26:50,y<0] <- x[26:50,y<0] - 1.5
# compute LPC and T scores for each gene
lpc.obj <- LPC(x,y, type="regression")
# Look at plot of Predictive Advantage
pred.adv <- PredictiveAdvantage(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh)
# Estimate FDRs for LPC and T scores
fdr.lpc.out <-
EstimateLPCFDR(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh,nreps=5)
# should run more reps in practice! Running just 5 in this example
# so that it runs fast.
# Estimate FDRs for T scores only. This is quicker than computing FDRs
#    for LPC scores, and should be used when only T FDRs are needed. If we
#    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
#    would give same T FDRs.
fdr.t.out <- EstimateTFDR(x,y, type="regression")
# print out results of main function
lpc.obj
# print out info about T FDRs
```

```
fdr.t.out
# print out info about LPC FDRs
fdr.lpc.out
# Compare FDRs for T and LPC on 6% of genes. In this example, LPC has
#    lower FDR.
PlotFDRs(fdr.lpc.out,frac=.06)
# Print out names of 20 genes with highest LPC scores, along with their
#    LPC and T scores.
PrintGeneList(lpc.obj,numGenes=20)
# Print out names of 20 genes with highest LPC scores, along with their
#    LPC and T scores and their FDRs for LPC and T.
PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)




# Now, repeating everything that we did before, but using a
#   **survival** outcome
### COMMENTED TO REDUCE RUN TIME -- UNCOMMENT TO RUN

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) + 10 # survival times; must be positive
## censoring outcome: 0 or 1
#cens <- rep(1,40) # Assume all observations are complete
## make first 50 genes differentially-expressed
#x[1:25,y<9.5] <- x[1:25,y<9.5]+ 1.5
#x[26:50,y<10] <- x[26:50,y<10] - 1.5
#lpc.obj <- LPC(x,y, type="survival", censoring.status=cens)
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type="survival",soft.thresh=lpc.obj$soft.thresh,
#censoring.status=cens)
## Estimate FDRs for LPC scores and T scores
#fdr.lpc.out <-
#EstimateLPCFDR(x,y, type="survival",soft.thresh=lpc.obj$soft.thresh,
#nreps=5,censoring.status=cens)
# Running just 5 reps so that things run quickly -- please run more in practice
## Estimate FDRs for T scores only. This is quicker than computing FDRs
###    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="survival", censoring.status=cens)
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC scores on 10% of genes.
#PlotFDRs(fdr.lpc.out,frac=.1)
```

```
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)


## Now, repeating everything that we did before, but using a
##   **multiclass** outcome

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  sample(1:4, n, rep=TRUE)
## make first 50 genes differentially-expressed
#x[1:50,y==1] <- x[1:50,y==1] + 1.5
#x[1:25,y==2] <- x[1:25,y==2] + 1
#x[1:50,y==3] <- x[1:50,y==3] - 1
#lpc.obj <- LPC(x,y, type=”multiclass”)
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type=”multiclass”,soft.thresh=lpc.obj$soft.thresh)
## Estimate FDRs for LPC scores and T scores
#fdr.lpc.out <-
#EstimateLPCFDR(x,y, type=”multiclass”,soft.thresh=lpc.obj$soft.thresh,
#nreps=20)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type=”multiclass”)
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC scores on 10% of genes.
#PlotFDRs(fdr.lpc.out,frac=.1)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)
```

PlotFDRs *Plot FDRs for both T and LPC*

Description

Takes the output of a call to EstimateLPCFDR, and uses it to plot the false discovery rates of the genes with highest LPC / T scores.

Usage

```
PlotFDRs(lpcfdr.out, frac=.25)
```

Arguments

`lpcfdr.out`	Output of a call to EstimateLPCFDR.
`frac`	The fraction of genes (with highest T/LPC scores) for which the T/LPC FDRs are plotted. Default is .25 (25%).

Value

Nothing is returned.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten, D.M. and Tibshirani, R. (2008) Testing significance of features by lassoed principal components. Annals of Applied Statistics. http://www-stat.stanford.edu/~dwitten

Examples

```
###### not running due to timing; uncomment to run ####

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) # quantitative outcome
## make first 50 genes differentially-expressed
#x[1:25,y<(-.5)] <- x[1:25,y<(-.5)]+ 1.5
#x[26:50,y<0] <- x[26:50,y<0] - 1.5
## compute LPC and T scores for each gene
#lpc.obj <- LPC(x,y, type="regression")
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh)
```

```
## Estimate FDRs for LPC and T scores
#fdr.lpc.out <-
#EstimateLPCFDR(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh,nreps=50)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="regression")
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC on 6% of genes. In this example, LPC has
##    lower FDR.
#PlotFDRs(fdr.lpc.out,frac=.06)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)




## Now, repeating everything that we did before, but using a
##   **survival** outcome
## Not run due to timing

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) + 10 # survival times; must be positive
## censoring outcome: 0 or 1
#cens <- rep(1,40) # Assume all observations are complete
## make first 50 genes differentially-expressed
#x[1:25,y<9.5] <- x[1:25,y<9.5]+ 1.5
#x[26:50,y<10] <- x[26:50,y<10] - 1.5
##lpc.obj <- LPC(x,y, type="survival", censoring.status=cens)
## Look at plot of Predictive Advantage
#pred.adv <- PredictiveAdvantage(x,y,type="survival",soft.thresh=lpc.obj$soft.thresh,
#censoring.status=cens)
## Estimate FDRs for LPC scores and T scores
#fdr.lpc.out <- EstimateLPCFDR(x,y, type="survival",
#soft.thresh=lpc.obj$soft.thresh,nreps=20,censoring.status=cens)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="survival", censoring.status=cens)
```

```
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC scores on 10% of genes.
#PlotFDRs(fdr.lpc.out,frac=.1)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)
```

`PredictiveAdvantage` *Plot predictive advantage of LPC vs. T*

Description

This function plots the predictive advantage of LPC vs. T. The predictive advantage is the difference between the red and black curves. If the red curve is higher than the black curve on average, then LPC should be used instead of T on this data set.

Usage

```
PredictiveAdvantage(x,y,type,nreps=20,ngenes=100,
soft.thresh=NULL,censoring.status=NULL)
```

Arguments

`x` The matrix of gene expression values; pxn where n is the number of observations and p is the number of genes.

`y` A vector of length n, with an outcome for each observation. For two-class outcome, y's elements are 1 or 2. For quantitative outcome, y's elements are real-valued. For survival data, y indicates the survival time. For multiclass outcome, y is coded as 1,2,3,..

`type` One of "regression" (for a quantitative outcome), "two class", "multiclass", or "survival".

`nreps` Number of training/test set splits used in computing predictive advantage.

`soft.thresh` Value of soft threshold used in L1 constraint for LPC. If NULL, then it will be computed adaptively.

`ngenes` Number of genes to include in predictive advantage plot. (E.g., if ngenes=100 (default) then $E(|T_test| | |L_train|>alpha(L_train))$ and $E(|T_test| | |T_train|>alpha(T_train))$ will be plotted (see "Details"), where $alpha(L_train)$ and $alpha(T_train)$ are the 100th largest (in absolute value) T and LPC scores on the training set.)

`censoring.status`
For survival outcome only, a vector of length n which takes on values 0 or 1 depending on whether the observation is complete or censored.

Details

As explained in the paper, predictive advantage is computed by first splitting the data into a training set and a test set (each with 50% of the samples). Then, the following is computed: $E(|T_test| | |L_train|>alpha(L_train)) - E(|T_test| | |T_train|>alpha(T_train))$, where $T_test$ are the T scores on the test data, $T_train$ are the T scores on the training data, and $L_train$ are the LPC scores on the training data. $alpha(L_train)$ and $alpha(T_train)$ are the $alpha$ quantiles of the LPC and T scores on the training data. A large value of $E(|T_test| | |L_train|>alpha(L_train)) - E(|T_test| | |T_train|>alpha(T_train))$ suggests that LPC is superior to T on this data set.

Value

`lpc` A vector of numGenes elements. The ith element is of the form $E(|T_test| | |L_train|>alpha(L_train))$, where $alpha(L_train)$ is the ith smallest (in absolute value) training set LPC score. Note that $L_train$ are LPC scores on the training set, $T_test$ are T scores on the test set.

`t` A vector of numGenes elements. The ith element is of the form $E(|T_test| | |L_train|>alpha(T_train))$, where $alpha(T_train)$ is the ith smallest (in absolute value) training set T score, and where $T_train$ are T scores on the training set and $T_test$ are T scores on the test set.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten, D.M. and Tibshirani, R. (2008) Testing significance of features by lassoed principal components. Annals of Applied Statistics. http://www-stat.stanford.edu/~dwitten

Examples

```
# Not run due to timing; uncomment to run

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) # quantitative outcome
## make first 50 genes differentially-expressed
#x[1:25,y<(-.5)] <- x[1:25,y<(-.5)]+ 1.5
#x[26:50,y<0] <- x[26:50,y<0] - 1.5
## compute LPC and T scores for each gene
#lpc.obj <- LPC(x,y, type="regression")
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh)
```

```
## Estimate FDRs for LPC and T scores
#fdr.lpc.out <-
#EstimateLPCFDR(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh,nreps=50)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="regression")
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC on 6% of genes. In this example, LPC has
##    lower FDR.
#PlotFDRs(fdr.lpc.out,frac=.06)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)




## Now, repeating everything that we did before, but using a
##   **survival** outcome

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) + 10 # survival times; must be positive
## censoring outcome: 0 or 1
#cens <- rep(1,40) # Assume all observations are complete
## make first 50 genes differentially-expressed
#x[1:25,y<9.5] <- x[1:25,y<9.5]+ 1.5
#x[26:50,y<10] <- x[26:50,y<10] - 1.5
#lpc.obj <- LPC(x,y, type="survival", censoring.status=cens)
## Look at plot of Predictive Advantage
#pred.adv <-
#PredictiveAdvantage(x,y,type="survival",soft.thresh=lpc.obj$soft.thresh,
#censoring.status=cens)
## Estimate FDRs for LPC scores and T scores
#fdr.lpc.out <- EstimateLPCFDR(x,y,type="survival",
#soft.thresh=lpc.obj$soft.thresh,nreps=20,censoring.status=cens)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="survival", censoring.status=cens)
```

```
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC scores on 10% of genes.
#PlotFDRs(fdr.lpc.out,frac=.1)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)
```

PrintGeneList *Print the list of genes with highest LPC scores*

Description

Takes the output of a call to LPC function, and prints out the (T and LPC) scores and names of the top-scoring genes. It can (optionally) also print out FDRs for the T and LPC scores.

Usage

```
PrintGeneList(lpc.obj, numGenes=100,gene.names=NULL, lpcfdr.out=NULL)
```

Arguments

`lpc.obj`	Output of a call to LPC.
`numGenes`	Desired length of list of top genes.
`gene.names`	Vector containing gene names; should have length equal to the number of genes in the data set.
`lpcfdr.out`	Optional paramater, it is the output of the function EstimateLPCFDR. If this is passed in, then FDRs for LPC and T will also be printed.

Value

Nothing is returned.

Author(s)

Daniela M. Witten and Robert Tibshirani

References

Witten, D.M. and Tibshirani, R. (2008) Testing significance of features by lassoed principal components. Annals of Applied Statistics. http://www-stat.stanford.edu/~dwitten

Examples

```
# Not running due to timing

#set.seed(2)
#n <- 40 # 40 samples
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) # quantitative outcome
## make first 50 genes differentially-expressed
#x[1:25,y<(-.5)] <- x[1:25,y<(-.5)]+ 1.5
#x[26:50,y<0] <- x[26:50,y<0] - 1.5
## compute LPC and T scores for each gene
#lpc.obj <- LPC(x,y, type="regression")
## Look at plot of Predictive Advantage
#pred.adv <- PredictiveAdvantage(x,y,type="regression",soft.thresh=lpc.obj$soft.thresh)
## Estimate FDRs for LPC and T scores
#fdr.lpc.out <- EstimateLPCFDR(x,y,type="regression",
#soft.thresh=lpc.obj$soft.thresh,nreps=50)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="regression")
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC on 6% of genes. In this example, LPC has
##    lower FDR.
#PlotFDRs(fdr.lpc.out,frac=.06)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)




## Now, repeating everything that we did before, but using a
##   **survival** outcome

#set.seed(2)
#n <- 40 # 40 samples
```

```
#p <- 1000 # 1000 genes
#x <- matrix(rnorm(n*p), nrow=p) # make 40x1000 gene expression matrix
#y <-  rnorm(n) + 10 # survival times; must be positive
## censoring outcome: 0 or 1
#cens <- rep(1,40) # Assume all observations are complete
## make first 50 genes differentially-expressed
#x[1:25,y<9.5] <- x[1:25,y<9.5]+ 1.5
#x[26:50,y<10] <- x[26:50,y<10] - 1.5
#lpc.obj <- LPC(x,y, type="survival", censoring.status=cens)
## Look at plot of Predictive Advantage
#pred.adv <- PredictiveAdvantage(x,y,type="survival",
#soft.thresh=lpc.obj$soft.thresh, censoring.status=cens)
## Estimate FDRs for LPC scores and T scores
#fdr.lpc.out <- EstimateLPCFDR(x,y,type="survival",
#soft.thresh=lpc.obj$soft.thresh,nreps=20,censoring.status=cens)
## Estimate FDRs for T scores only. This is quicker than computing FDRs
##    for LPC scores, and should be used when only T FDRs are needed. If we
##    started with the same random seed, then EstimateTFDR and EstimateLPCFDR
##    would give same T FDRs.
#fdr.t.out <- EstimateTFDR(x,y, type="survival", censoring.status=cens)
## print out results of main function
#lpc.obj
## print out info about T FDRs
#fdr.t.out
## print out info about LPC FDRs
#fdr.lpc.out
## Compare FDRs for T and LPC scores on 10% of genes.
#PlotFDRs(fdr.lpc.out,frac=.1)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores.
#PrintGeneList(lpc.obj,numGenes=20)
## Print out names of 20 genes with highest LPC scores, along with their
##    LPC and T scores and their FDRs for LPC and T.
#PrintGeneList(lpc.obj,numGenes=20,lpcfdr.out=fdr.lpc.out)
```

Index