Package 'RPCR'

October 19, 2014

Type Package

Title High-dimensional survival prediction using RPCR.

Version 1.0

Date 2014-10-16

Author Wei Liu <freelw@gmail.com> and Chunquan Li<lcqbio@yahoo.com.cn>

Maintainer Wei Liu <freelw@gmail.com>

Depends R (>= 3.1.1), survival, igraph, Matrix

Description

This package implements the reweighted partial Cox Regression method which used for survival analysis on high-dimensional gene expression data and the directed random walk algorithm which used to evaluate the topological importances of nodes in the global pathway graph.

License GPL(>=2)

R topics documented:

dGMGraph *The global pathway graph.*

Description

The directed global pathway graph constructed by the R package `iSubpathwayMiner`.

Usage

```
data("dGMGraph")
```

Format

An `igraph` R object.

Details

There are 5746 nodes in `dGMGraph`. Each node in the graph represents a gene or a metabolite. The global pathway graph is used to evaluate the topological importances of genes by directed random walk.

Examples

```
data(dGMGraph)
```

DRW *Directed Random Walk*

Description

The directed random walk algorithm proposed by Liu et al(2013).

Usage

```
DRW(igraphM, p0, EdgeWeight = FALSE, gamma = 0.3)
```

Arguments

`igraphM`	An igraph object containing the directed global pathway graph.
`p0`	A unit vector containing the initial weights of genes in the global pathway graph.
`EdgeWeight`	Logical. Should `igraphM` be converted to a weighted matrix or an un-weighted matrix (the default)?
`gamma`	A numeric value. The restart probability in directed random walk.

Details

This function implements the directed random walk algorithm proposed by Liu et al (2013). It evaluates the topological weight of each gene according to its topological importance in the global pathway graph. The genes that close to many other genes that have large initial weights will receive larger weights. The final weights reflect the topological importances of genes in the global pathway graph.

Value

A numerical vector containing the topological weights of nodes in `igraphM`.

Author(s)

Wei Liu <freelw@gmail.com>

References

Liu, W., et al., Topologically inferring risk-active pathways toward precise cancer classification by directed random walk. Bioinformatics, 2013. 29(17): p. 2169-77.

Examples

```
data(dGMGraph)
vertexs <- V(dGMGraph)
p0 <- runif(length(vertexs), min = 0, max = 1)
names(p0) <- vertexs$name
p0 <- p0/sum(p0)
vertexWeight <- DRW(igraphM = dGMGraph, p0, EdgeWeight=FALSE, gamma = 0.3)
names(vertexWeight) <- names(p0)
```

GBM *GBM survival data set.*

Description

A GBM survival data set used to test the RPCR model.

Usage

```
data("GBM")
```

Format

A data frame with 100 observations on the following 215 variables.

Examples

```
data(GBM)
```

GBMForDRW *The GBM expression profiles.*

Description

The GBM expression profiles used to evaluate the topological importances of genes in the global pathway graph.

Usage

```
data("GBMForDRW")
```

Format

A data frame with 100 observations on the following 3916 variables.

Examples

```
data(GBMForDRW)
```

getTPWeight *Obtain topological weights of genes*

Description

Evaluate the topological weights of genes in the global pathway graph by directed random walk.

Usage

```
getTPWeight(globalGraph, data, Gamma = 0.3)
```

Arguments

globalGraph	An igraph object containing the directed global pathway graph.
data	A data frame containing the gene expression data and survival data used to initialize the weights of genes.
Gamma	A numeric value. The restart probability in directed random walk.

Details

This function evaluates the topological importance of each node in the global pathway graph `globalGraph` and returns the topological weights of genes. The argument `data` is a data frame containing the gene expression data and survival data. Each row represents a sample, and each column represents a gene (the last two columns represent "status" and "time" respectively). The rownames of data are sample names, and the colnames of data are Entrez gene IDs (the last two columns are "status" and "time" respectively). The initial weights of genes in directed random walk are initialized by assigning to each gene as its -log(P-value) from a univariate Cox regression analysis on samples in `data`, and normalized to a unit vector.

Value

A numerical vector containing the topological weights of nodes in `globalGraph`.

Author(s)

Wei Liu <freelw@gmail.com>

References

Liu, W., et al., Topologically inferring risk-active pathways toward precise cancer classification by directed random walk. Bioinformatics, 2013. 29(17): p. 2169-77.

See Also

DRW

Examples

```
# test getTPWeight
data(dGMGraph)
data(GBMForDRW)
geneTPW <- getTPWeight(globalGraph = dGMGraph, data = GBMForDRW, Gamma = 0.3)
```

`partial.coxph` *Fitting of a Partial Cox Regression model*

Description

Method for fitting a Partial Cox Regression model (Li and Gui, 2004) to survival data.

Usage

```
partial.coxph(formula, data = parent.frame(), control,
              method = c("efron", "breslow"), degree,
              min.degree, max.degree, rescale = TRUE, ...)
```

Arguments

`formula`	A formula object, with the response on the left of a '~' operator, and the covariates to the right. The response must be a survival object as returned by the `Surv(.,.)` function.
`data`	An optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model.
`control`	Fitting options used when fitting the Cox models. `control` should be created using `control.coxph`.
`method`	The method used for breaking ties. See the documentation of `coxph` for details.
`degree`	The degree (number of hidden variables) used for fitting the Partial Cox Model. You can use `min.degree` and `max.degree` to compute regression coefficients for a range of degrees.

min.degree	If coefficients for more than one degree are to be computed, the range of degrees can be specified using min.degree and max.degree.
max.degree	See max.degree.
rescale	If rescale is set to TRUE, then all covariates are standardised beforehand.
...	Additional arguments passed on to coxph.fit.

Value

partial.coxph returns an object of the class partial.coxph, which is a list containing, amongst others, the following elements:

coef	The estimated coefficient vector. If more than one degree is specified than the columns of coef correspond to each of the degrees specified.
x.centre	The mean of the training data that was used to centre the data.
x.scale	The standard deviation of the training data that was used to rescale the covariates. Note that coef is already rescaled appropriately.
degree	The range of degree for which the model has been estimated.

Author(s)

Ludger Evers <ludger@stats.gla.ac.uk>

References

Li, H., Gui, J. Partial Cox regression analysis for high-dimensional microarray gene expression data, Bioinformatics, 20, i208-i215 (2004).

See Also

coxph

---

predict.RPCR *Predicting the risk of patients.*

Description

Predict the risk of patients based on the RPCR model.

Usage

```
predict.RPCR(object, newdata, degree, ...)
```

Arguments

object	Object of class inheriting from RPCR.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
degree	An integer value. The number of PCR components that specified when fitting the RPCR model.
...	Ignored.

Value

Return the predicted scores.

Author(s)

Wei Liu <freelw@gmail.com>

References

Li, H., Gui, J. Partial Cox regression analysis for high-dimensional microarray gene expression data, Bioinformatics, 20, i208-i215 (2004).

See Also

RPCR

Examples

```
data(dGMGraph)
data(GBMForDRW)
data(GBM)
geneTPW <- getTPWeight(globalGraph = dGMGraph, data = GBMForDRW, Gamma = 0.3)
TR <- GBM[1:80, ]
TE <- GBM[81:100, 1:(ncol(GBM)-2)]
RPCRModel <- RPCR(data = TR, geneTPWeight = geneTPW, D = 3)
lp <- predict.RPCR(object = RPCRModel, D = 3)
lpnew <- predict.RPCR(object = RPCRModel, newdata = TE, D = 3)
```

RPCR *Fitting the RPCR model.*

Description

Method for fitting the Reweighted Partial Cox Regression (RPCR) Model.

Usage

```
RPCR(data, geneTPWeight, D = 3)
```

Arguments

data	A data frame containing the gene expression data and survival data used to build the RPCR model.
geneTPWeight	A numerical vector containing the topological weights of genes obtained from getTPWeight.
D	An integer value. The number of PCR components used to build the RPCR model.

Details

This function implements the fitting of the RPCR model. It integrates the topological importances of genes to reweight the coefficients of genes in the partial cox regression model. This strategy can improve the predictive accuracy and the generalization of the Cox model.

The argument `data` is a data frame containing the gene expression data and survival data. Each row represents a sample, and each column represents a gene (the last two columns represent "status" and "time" respectively). The rownames of data are sample names, and the colnames of data are Entrez gene IDs (the last two columns are "status" and "time" respectively). We suggest user uses only those genes that are significant in a univariate Cox regression analysis to build the RPCR model.

`D` is the number of PCR components that used to build the RPCR model. To determine a proper `D` value, one can test the significance of association between each PCR component and survival time using univariate Cox regression analysis, and select the top `D` significant PCR components to build the final RPCR model.

Value

Returns an object of the class `partial.coxph`, which is a list containing the following elements:

`coef`	The estimated coefficient vector.
`x.centre`	The mean of the training data that was used to centre the data.
`x.scale`	The standard deviation of the training data that was used to rescale the covariates. Note that coef is already rescaled appropriately.
`degree`	The number of PCR components used to build the RPCR model.

Author(s)

Wei Liu <freelw@gmail.com>

References

Li, H., Gui, J. Partial Cox regression analysis for high-dimensional microarray gene expression data, Bioinformatics, 20, i208-i215 (2004).

See Also

`predict.RPCR`

Examples

```
data(dGMGraph)
data(GBMForDRW)
data(GBM)
geneTPW <- getTPWeight(globalGraph = dGMGraph, data = GBMForDRW, Gamma = 0.3)
TR <- GBM[1:80, ]
TE <- GBM[81:100, 1:(ncol(GBM)-2)]
RPCRModel <- RPCR(data = TR, geneTPWeight = geneTPW, D = 3)
```

Index