stam

April 19, 2009

`golubTrain.cv` *Examplar StAM Cross-Validation Results*

Description

This data set has been generated by `stam.cv`. It has been computed on the Golub data set on leukemia, the classification task being to separate AML from ALL patients.

Usage

```
data(golubTrain.cv)
```

Format

This is a `stamCV` object

Details

The original data set is drawn from the `golubEsets` library. The samples 1 to 38 have been fed to `stam.cv` to generate the data set at hand.

See Also

`stamCV-class`, `stam.cv`, `golubTrain`

`golubTrain.fit` *Examplar StAM Model Fit*

Description

This data set has been generated by `stam.fit`. It has been computed on the Golub data set on leukemia, the classification task being to separate AML from ALL patients.

Usage

```
data(golubTrain.cv)
```

Format

This is a stamFit object

Details

The original data set is drawn from the golubEsets library. The samples 1 to 38 have been fed to stam.cv to generate the golubTrain.cv. This set in turn was given to stam.fit to generate the data set at hand.

See Also

stamFit-class, stam.fit, golubTrain.cv

---

image.stamPrediction

Molecular Symptoms Image on StAM Prediction

Description

Shows prediction results for each sample of the class of interest and for all nodes of a StAM model fit as a color coded image.

Usage

```
## S3 method for class 'stamPrediction':
image(x, aclass = NULL, main = NULL,
                    xlab = NULL, show.graph = TRUE,
                    max.label.length = 40,
                    sample.labels = FALSE, full.names = TRUE,
                    outfile = NULL, ps = FALSE, res = 72,
                    pointsize = 10, width = 11,
                    minspec = NULL, minsens = 0.1,
                    maxsens = 1, invert = FALSE,
                    col = stam.rgb.colors(r0 = 0),
                    bin.thresh = NULL, ...)
```

Arguments

x	stamPrediction object to be illustrated
aclass	the name of the phenotype class of interest, usually the disease class. If set to NULL the first class in lexicographical order is chosen
main	the main title of the plot, generated automatically if left blank
xlab	the label of the x-axis
show.graph	(default is T) whether or not to show GO relations between the nodes
max.label.length	the maximum string length for a GO term
sample.labels	whether or not to show sample names on the x-axis
full.names	whether or not to show GO terms instead of GO IDs

outfile	name of output file if postscript or PNG graphics is to be generated. The extension of the file is chosen automatically according to the ps argument. If no outfile is specified, an interactive plot is attempted
res	resolution in points per inch
ps	if set to TRUE postscript output is generated
pointsize	the standard fontsize
width	width of image in inches (the height is computed according to the number of nodes to be shown and the pointsize)
minspec	nodes to be shown must be at least this specific. If set to NULL this is chosen such that no more than 50 nodes are shown.
minsens	nodes to be shown must be at least this sensitive
maxsens	nodes to be shown must be at most this sensitive
invert	whether or not to invert background and foreground colors
col	the color gradient to code classifier output
bin.thresh	threshold for binary color coding, if left at the default NULL, classifier outputs are coded on a continuous scale.
...	additional options passed to image

Details

This image illustrates classifier outputs generated during a structured analysis of microarrays (StAM). The centralpart of the image shows the color coded classifier outputs for each sample in the class of interest (columns) and each nodes from StAM's model fit (rows). Nodes can be restricted to those of minimal sensitivity or specificity. Also nodes with particularly high sensitivity can be excluded. The color code for the classifier outputs is shown in a color bar on the right hand side if a continuous is used. If a testset is specified only these samples are used to compute sensitivity and specificity. If the sample names are not displayed on the x-axis, the test samples are marked with capital letters or just vertical bars (bars if there are too many test samples). Sensitivity and specificity is shown to the left of the figure together with the relations between the GO nodes from the Gene Ontology. GO terms are printed on the right of the image and may be used for a clickable map in HTML.

Value

A string to be used on an HTML page to provide a clickable map for the GO terms.

Author(s)

Claudio Lottaz

See Also

stam.predict

`plot.stamCV` *Plots for StAM Cross Validation*

Description

Plots performance and redundancy as well as number of remaining nodes and genes for a cross validation in structured analysis of microarray data.

Usage

```
## S3 method for class 'stamCV':
plot(x, outfile = NULL, aclass = NULL, delta = NULL,
            main = NULL, which = 0, res = 72, ps = FALSE,
            pointsize = 16, ...)
```

Arguments

`x`	the object of type stamCV for which the plots are to be drawn
`outfile`	name of output file if postscript or PNG graphics is to be generated. The extension of the file is chosen automatically according to the `ps` argument. If no outfile is specified, an interactive plot is attempted
`aclass`	the name of the phenotype class of interest, usually the disease class
`which`	choose the plot to be generated. 1:performance plot, 2:genes/nodes plot, 0:both plots in interactive mode, non otherwise.
`delta`	if a delta is provided, vertical lines are added accordingly
`main`	the main title of the plot, generated automatically if left blank
`res`	resolution in points per inch
`ps`	if set to TRUE postscript output is generated
`pointsize`	the standard fontsize
`...`	additional arguments to be passed to plot

Details

This function generates two plots. The first one plots the root error rate, its performance and the graph redundancy versus the shrinkage level. The second plot depicts the number of remaining nodes and remaining accessible probesets for each shrinkage candidate.

Author(s)

Claudio Lottaz

See Also

`stam.cv`

`plot.stamFit` *Plots for StAM Model Fit*

Description

Overall alpha vs. delta plot as well as nodewise scatter plots on performance/redundancy and specificity/sensitivity.

Usage

```
## S3 method for class 'stamFit':
plot(x, outfile = NULL, aclass = NULL, main = NULL,
             which = 0, res = 72, ps = FALSE, pointsize = 12, ...)
```

Arguments

`x`	the object of type stamFit for which the plots are to be drawn
`outfile`	name of output file if postscript or PNG graphics is to be generated. The extension of the file is chosen automatically according to the `ps` argument. The filename is also augmented by suffixses to distinguish the plots generated. If no outfile is specified, an interactive plot is attempted
`aclass`	the name of the phenotype class of interest, usually the disease class
`which`	choose the plot to be generated. 1:alpha vs. delta, 2:nodewise evaluation, 0:both plots in interactive mode, non otherwise.
`main`	the main title of the plot, generated automatically if left blank
`res`	resolution in points per inch
`ps`	if set to TRUE postscript output is generated
`pointsize`	the standard fontsize
`...`	additional arguments to be passed to plot

Details

If several values for alpha are provided to stam.fit plot.stamFit generates a compound score vs. shrinkage level plot. For each alpha one line is drawn and the best shrinkage (where the minimum is achieved) is marked. In a second pane of the same plot alpha is plotted vs. these best shrinkage levels.

Additionally nodewise scatter plots comparing performance vs. redundancy and sensitivity vs. specificity are generated.

Author(s)

Claudio Lottaz

See Also

`stam.fit`

`plot.stamPrediction` *Scatter Plot on Node Results*

Description

Plots nodewise performance vs. redundancy as well as nodewise sensitivity vs. specificity side by side.

Usage

```
## S3 method for class 'stamPrediction':
plot(x, outfile = NULL, aclass = NULL,
                    main = NULL, minspec = 0.9,
                    minsens = 0.1, maxsens = 1,
                    res = 72, ps = FALSE, pointsize = 12,
                    ...)
```

Arguments

`x`	object of class stamPrediction to be plotted
`outfile`	name of output file if postscript or PNG graphics is to be generated. The extension of the file is chosen automatically according to the `ps` argument. If no outfile is specified, an interactive plot is attempted
`aclass`	the name of the phenotype class of interest, usually the disease class. If set to NULL the first class in lexicographical order is chosen.
`main`	the main title of the plot, generated automatically if left blank
`minspec`	if given the corresponding horizontal line is drawn in the sensitivity vs. specificity plot
`minsens`	if given the corresponding vertical line is drawn in the sensitivity vs. specificity plot
`maxsens`	if given the corresponding vertical line is drawn in the sensitivity vs. specificity plot
`res`	resolution in points per inch
`ps`	if set to TRUE postscript output is generated
`pointsize`	the standard fontsize
`...`	additional arguments to be passed to plot

Author(s)

Claudio Lottaz

See Also

`stam.predict`

stam.cv *Cross Validated Training for StAM*

Description

Determine classifiers in leaf nodes and weights in inner nodes as well as best graph shrinkage by cross validated model fitting.

Usage

```
stam.cv(expression.matrix, classifications,
        chip = "hgu95av2", root = "GO:0008150",
        beta = NULL, deltas = NULL, ndeltas = 10,
        results.per.node = FALSE, old.cv = NULL,
        pamimagefile = NULL, verbose = FALSE)
```

Arguments

expression.matrix	holds the expression levels. It may be of classes exprSet or ExpressionSet, or a plain numeric matrix. In the first case exprs is used to extract the expression levels. The matrix is expected to hold one column per sample and one row per probeset.
classifications	This character vector must contain one entry per sample identifying the group it belongs to. Alternatively, if expression.matrix is an exprSet or ExpressionSet, this may be the name of a phenoData variable.
chip	the name of the microarray chip. A meta data package is expected to be found holding the needed annotation, namely the links between probesets and Gene Ontology nodes.
root	the GO node used as root of the classifier graph. Only successors of this node are considered during construction of the graph.
beta	holds class weights used when judging classifier quality. The default is to set class weights to the corresponding prevalence.
deltas	numeric vector holding graph shrinkage candidates. Default is to determine ndelta candidates between 0 and the lowest shrinkage level which removes all leaf nodes.
ndeltas	number of automatically determined graph shrinkage candidates determined if deltas is not defined.
results.per.node	whether results for each node should be returned
old.cv	stamCV object used to modify when PAM fits need not to be recomputed. E.g. used when only beta is adapted.
pamimagefile	When this parameter is specified stam.cv tries to read this file and extract a stamCV object to be used as old.cv. If the file does not yet exist, PAM fits are stored there after computation.
verbose	when set to TRUE reports summary on each leaf training, otherwise shows a progress bar.

Details

`stam.cv` uses `stam.net` to generate a classifier graph for the microarray chip at hand. It then fits a PAM classifier for each leaf node only considering the probesets annotated to the node. Afterwards, in each inner node, weights are attributed to each child according to the childs classification performance. Finally, the weights are shrunken such that most of them become zero. In fact, the best shrinkage level is chosen in a cross validation setting.

Classification performance is evaluated using an inverted deviance like measure which uses weights to overstate specificity of a classifier. Weights for nodes are chosen according to this measure and shrunken by an absolute shrinkage level. For each shrinkage candidate cross validated performance results in terms of graph heterogeneity and classification performance are stored.

Value

An object of class stamCV is returned. Use the methods print and plot to extract information about the cross validation.

Author(s)

Claudio Lottaz

See Also

`stamCV-class`, `plot.stamCV`, `stam.writeHTML`

Examples

```
## Not run:
# load and prepare some data
library(golubEsets)
data(Golub_Merge)
golubTrain <- Golub_Merge[,1:38]

# classify into ALL and AML
# (root is chosen to yield results reasonably fast,
#  consider GO:0008150 (biological process) to obtain
#  meaningful results)
golubTrain.cv <- stam.cv(golubTrain, "ALL.AML", chip="hu6800",
                         root="GO:0005576", ndeltas=10)

# get further information
print(golubTrain.cv)
plot(golubTrain.cv, delta=0.6)
## End(Not run)
```

`stam.evaluate` *StAM Evaluation Procedure*

Description

This performs a structured analysis of microarrays (StAM) from scratch to the end. It starts with a cross-validation, performas a model fit, predicts phenotypes and writes complete HTML code with images.

Usage

```
stam.evaluate(expression.matrix, classifications,
              report.dir = getwd(), aclass =
              names(table(classifications))[1],
              titlestem = NULL, testset =
              stam.balanced.folds(classifications, 3)[[1]],
              chip = "hgu95av2", root = "GO:0008150",
              no.output = FALSE, alpha = seq(0, 1, 0.1),
              beta = NULL, deltas = NULL, ndeltas = 30,
              minspec = NULL, minsens = 0.1, maxsens = 1,
              pamimagefile = NULL)
```

Arguments

`expression.matrix`	holds the expression levels. It may be of class `exprSet` or `ExpressionSet`, or a plain numeric matrix. In the first case `exprs` is used to extract the expression levels. The matrix is expected to hold one column per sample and one row per probeset.
`classifications`	This character vector must contain one entry per sample identifying the group it belongs to.
`aclass`	the name of the phenotype class of interest, usually the disease class. If set to NULL the first class in lexicographical order is chosen.
`testset`	indeces of the columns in the expression.matrix representing test samples.
`chip`	the name of the microarray chip. A meta data package of the same name is expected to be found holding the needed annotation, namely the links between probesets and Gene Ontology nodes.
`root`	the GO node used as root of the classifier graph. Only successors of this node are considered during construction of the graph/model.
`alpha`	root performance vs. mean redundancy weight. If set to NULL the root error rate is used exclusively to determine the best shrinkage level. If a numeric vector is provided, all alternatives are computed and the user is given an interactive choice. Values between 0 and 1 are valid, 0 meaning exclusive weight on redundancy and 1 putting exclusive weight on performance.
`beta`	holds class weights used when judging classifier quality. The default is to set class weights to the corresponding prevalence. Several combinations of class weights may be provided for testing one after the other. To do so a matrix is expected to hold one combination of weights per row and must thus have one column per class.
`deltas`	numeric vector holding graph shrinkage candidates. Default is to determine `ndelta` candidates between 0 and the lowest shrinkage level which removes all leaf nodes.
`ndeltas`	number of automatically determined graph shrinkage candidates determined if `deltas` is not defined.
`titlestem`	the first part of the title of the HTML page to be written, is complemented by some of the parameters.
`report.dir`	the directory where the HTML pages are to be written
`no.output`	do not generate any HTML or images

minspec	nodes to be shown in molecular symptoms image must be at least this specific
minsens	nodes to be shown in molecular symptoms image must be at least this sensitive
maxsens	nodes to be shown in molecular symptoms image must be at most this sensitive
pamimagefile	When this parameter is specified stam.cv tries to read this file and extract a stamCV object to avoid recomputing PAM fits. If the file does not yet exist, PAM fits are stored there after computation.

Details

stam.evaluate exexutes all steps needed in a structured analysis of a microarray study and coherently generates HTML output including plots and images. In Firstly, a 10 fold cross validation is performed with the data not identified as test set. Secondly, using an adequate graph shrinkage level, a model fit is computed. Finally, all data is used for prediction to illustrate the performance.

Furthermore, this method generates a set of HTML pages. One page reports on the analysis as a whole, while additional interlinked pages, one for each node in the model fit, contain information on the fit and results of each node. On the main page plots and images illustrate and summarize the analysis. Clickable maps make the exploration of the results convenient. All files are stored together with an R data containing the returned R object in the user specified report directory.

Value

Returns an object of class stamEval containing all results generated during the above described procedure. Use the methods defined on the class corresponding the slot you want to investigat further.

Author(s)

Claudio Lottaz

See Also

stamEval-class, stam.cv, stam.fit, stam.predict, stam.writeHTML

Examples

```
# load and normalize some data
## Not run:
library(golubEsets)
data(Golub_Merge)

# (root is chosen to yield results reasonably fast,
#  consider GO:0008150 (biological process) to obtain
#  meaningful results)

# demonstrate the use of several combinations of class weights
betas <- cbind(c(0.5, 0.8, 0.9), c(0.5, 0.2, 0.1))
golubNorm.eval.explore <- stam.evaluate(Golub_Merge, "ALL.AML",
                                        chip="hu6800", root="GO:0005576",
                                        alpha=seq(0, 1, 0.1), beta=betas, ndelta=10)

# demonstrate the use of testsets
golubNorm.eval.predict <- stam.evaluate(Golub_Merge, "ALL.AML", testset=39:72,
                                        chip="hu6800", root="GO:0005576", ndelta=10)
## End(Not run)
```

stam.fit *Fit StAM Model to Training Data*

Description

Using the whole expression data provided fit one StAM model according to the chosen shrinkage level.

Usage

```
stam.fit(cv, expression.matrix, collapse.scnodes = FALSE,
         alpha = 0.5, delta = NULL, max.nodes = 100)
```

Arguments

cv	result stam.cv on the same data, must be of class stamCV
expression.matrix	holds the expression levels. It may be of class exprSet or ExpressionSet, or a plain numeric matrix. In the first case exprs is used to extract the expression levels. The matrix is expected to hold one column per sample and one row per probeset.
collapse.scnodes	if set to TRUE replace single children nodes after shrinkage
alpha	root performance vs. mean redundancy weight. If set to NULL the root error rate is used exclusively to determine the best shrinkage level. If a numeric vector is provided, all alternatives are computed and the user is given an interactive choice. Values between 0 and 1 are valid, 0 meaning exclusive weight on redundancy and 1 putting exclusive weight on performance.
delta	overrule alpha and set shrinkage level explicitely.
max.nodes	choose default shrinkage level such that no more than this number of nodes remain after shrinkage.

Details

In a first step stam.fit must choose a shrinkage level. In order to do so it uses results stored in the cv. If the user provides a shrinkage level explicitly this delta is used. If he specifies a single weighting factor alpha the corresponding weighted score is used to determine the best shrinkage level. If alpha is set to a vector of values, the corresponding scores are computed and a default delta is suggested using the median value of the alphas. If this shrinkage level leads to more than max.nodes nodes remaining the shrinkage level is increase until no more than max.nodes remain after shrinkage.

Using the thus determined shrinkage level a weighting of nodes is computed using the leaf node results from cv. Thereby, the whole dataset supplied is used.

Value

An object of class stamFit is returned. You may use the print and plot methods to further investigate the returned value.

Author(s)

Claudio Lottaz

See Also

stam.cv, stamFit-class, plot.stamFit, stam.graph.plot, stam.writeHTML

Examples

```
## Not run:
# prepare data
library(golubEsets)
data(Golub_Merge)

# load and prepare some data
golubTrain <- Golub_Merge[,1:38]
data(golubTrain.cv)

# compute fit
golubTrain.fit <- stam.fit(golubTrain.cv, golubTrain, alpha=seq(0, 1, 0.1))

# investigate
print(golubTrain.fit)
plot(golubTrain.fit)
## End(Not run)

# show clickable web-page
## Not run:
map <- stam.graph.plot(golubTrain.fit, outfile="golubTrain")
cat("<HTML><BODY><MAP NAME='graph_map'>", map, "</MAP>",
    "<IMG SRC='golubTrain_graph_plot.png' USEMAP='#graph_map'></BODY></HTML>\n",
    file="graph_plot.html")
browseURL(paste("file://", getwd(), "/graph_plot.html", sep=""))
## End(Not run)
```

---

stam.graph.plot *Draw StAM Model Fit Graph*

Description

This function uses graphviz to layout a graph plot of a model fit. In addition a client side clickable map is returned to added to an HTML page.

Usage

```
stam.graph.plot(x, outfile = "", pointsize = 10,
                width = 9, height = 6)
```

Arguments

x	the stamFit object holding the model to be drawn
outfile	name of output file without extension.
pointsize	the standard font size

width	width of plot in inches
height	height of plot in inches

Details

This function generates a file in the dot language for graphviz. It uses the dot program to produce the layout of the graph and png as well as postscript files of this layout. Moreover, a client-side clickable map is generated which can be included in an HTML page.

Value

A character string containing HTNL code for a clickable map.

Note

This function only works on unix systems with graphviz installed.

Author(s)

Claudio Lottaz

References

Gansner ER, North SC. "An open graph visualization system and its applications to software engineering". Software Practice and Experience, 1999, pp. 1–5.

See Also

stam.fit

---

stam.net *Generate a Classifier Graph for StAM*

Description

Generates a classifier graph for structured analysis of microarray data based on the Gene Ontology.

Usage

```
stam.net(chip = "hgu95av2", root = "GO:0008150", probes = character(0))
```

Arguments

chip	A character string representing the microarray chip from which data has been generated. A meta data packages of the same name is used to obtain the annotation data, namely for the Gene Ontology.
root	The identifier of the GO node to be used as the root of the classifier graph. Only successors of this node are considered to generate the graph.
probes	The probe names in the order as they occur in the expression matrix. Indeces according to this character array are used to index probes/probesets.

Details

stam.net crawls through the Gene Ontology starting with the root node specified. It collects all successors of the root into a classifier graph according to the parent-children relations defined in the Gene Ontology. Probesets of the microarray chip at hand are attributed to GO nodes according to the Bioconductor annotation meta data package for the chip.

For any node i which has GO annotations AND successors an additional node i' is introduced to the classifier graph. The new node is added as an additional child to i and all probesets annotated to i are moved to i', such that only leaf nodes hold probesets. Nodes which neither themselves nor any of their successors hold probesets are discarded. stam.net also replaces inner nodes with only one child by their successor.

Value

An object of class stamNet is returned. You may use the print method to obtain detailed information about the classifier graph. You may further investigate any element of the 'nodes' list using the print method.

Author(s)

Claudio Lottaz

See Also

stamNet-class, stam.writeHTML

Examples

```
## Not run:
# load some data
library(golubEsets)
data(Golub_Merge)
emat <- exprs(Golub_Merge)

# determine classifier graph for chip "hu6800" on
# biological processes, taking the positions of
# probesets in Golub's expression matrix into account
net <- stam.net(chip="hu6800", root="GO:0003674", probes=rownames(emat))

# have a look
print(net)
print(net@nodes[[16]])
print(net@nodes$"GO:0007638")
## End(Not run)
```

---

stam.predict *Predict Classifications of New Data*

Description

StAM analysis on new data using a given model fit.

Usage

```
stam.predict(fit, expression.matrix, classifications = NULL,
             testset = NULL)
```

Arguments

`fit`	stamFit object containing a trained model
`expression.matrix`	matrix or `exprSet` or `ExpressionSet`, containing new data
`classifications`	character vector specifying class names per sample. You may either specify one class per sample in the expression.matrix, or one class per training sample (all but the testset).
`testset`	indeces of samples not yued in training

Details

stam.predict uses an object returned by stam.fit to perform a structured analysis of the new expression data provided. Thereby, it uses all classifiers in the leaf nodes to provide classification results in these for each sample. In addition, weighted sums in inner nodes are computed to provide classification results for the whole graph.

Value

An object of type stamPrediction is returned. You may use print, plot and image functions to further investigate the results. Information on node classifiers are obtained through the print methods on elements of the `nodes` slot.

Author(s)

Claudio Lottaz

See Also

`stamPrediction-class`, `plot.stamPrediction`, `image.stamPrediction`, `stam.fit`, `stam.writeHTML`

Examples

```
## Not run:
# load and prepare data
library(golubEsets)
data(Golub_Merge)
golubTest <- Golub_Merge[,39:72]
data(golubTrain.fit)

# compute predictions
golubTest.pred <- stam.predict(golubTrain.fit, golubTest,
                               pData(golubTest)[,"ALL.AML"])
golubMerge.pred <- stam.predict(golubTrain.fit, Golub_Merge,
                                pData(Golub_Merge)[,"ALL.AML"], testset=39:72)

# further investigate
print(golubTest.pred)
```

```
plot(golubTest.pred)
## End(Not run)
## Not run:
map <- image(golubMerge.pred, outfile="golubMerge")
cat("<HTML><BODY><MAP NAME='image_map'>", map, "</MAP>",
    "<IMG SRC='golubMerge_pred_img.png' USEMAP='#image_map'></BODY></HTML>\n",
    file="pred_img.html")
browseURL(paste("file://", getwd(), "/pred_img.html", sep=""))
## End(Not run)
```

`stam.serve` *StAM server launch and installation*

Description

stam.serve installs StAM's server feature if it has not been installed before. Moreover it launches the StAM server needed to use StAM output interactively through the internet.

Usage

```
stam.serve(tmp.path = NULL, cgi.path = NULL, cgi.url = NULL)
```

Arguments

`tmp.path`	the path to the directory where files are stored for communication between your WWW-server and the StAM-server.
`cgi.path`	the path to the directory in which the StAM-related CGI scripts are to be stored. Make sure that your WWW-server can execute CGI scripts from here and that access rights are set correctly.
`cgi.url`	the URL prefix needed to access the CGI scripts, i.e. the directory specified in cgi.path.

Details

The stam package provides a feature to manipulate some parameters interactively using HTML forms. This feature needs a WWW browser which is able to execute CGI scripts. The HTML output used to work with interactively must be generated with the stam.write.forms option turned on.

StAM provides a set of CGI scripts which are called from the forms written into the HTML code when the above mentioned option is turned on. This scripts write task files into the directory given by tmp.path. This directory must have write permission for the WWW server. The StAM server regularly checks this directory for such tasks and executes them.

stam.serve installs the StAM server feature when it is called the first time after the installation of the stam package. Firstly, this consists of registering the three parameters of stam.serve into the package installation such that they can be reloaded when the packages is loaded into R the next time. For this purpose a dataset is written into the installation, thus you need write permissions in the corresponding directory. Secondly, the cgi.url is written into StAM's CGI scripts and these scripts are written into the cgi.path.

When the installation of StAM's server feature is complete, stam.serve starts checking the directory where it expects the tasks written by the CGI scripts and is thus ready for operation. After the server feature has been installed the server can be launched simply by calling stam.serve() without parameters. A second call with parameters modifies the StAM server installation accordingly.

Note

You need write permission in the stam installation directory in order to install StAM's server feature.

You need write permission in the directory where the CGI scripts are to be deposited in order to install StAM's server feature.

You must regenerate all HTML you want to work with through the stam server after installation of the server feature.

You must turn on the stam.write.forms option when generating HTML for use with the server feature.

Author(s)

Claudio Lottaz

See Also

stam.writeHTML

Examples

```
## Not run:
# make sure subsequent calls to stam.writeHTML generate forms
options(stam.write.forms=TRUE)

# first call to stam.serve after installation of the stam package
stam.serve(tmp.path = "/home/myhome/upload",
           cgi.path = "/home/myhome/cgi-bin/stam",
           cgi.url  = "http://www.myserver.com/cgi-bin/stam")

# subsequent calls to launch StAM server without modifying the installation
stam.serve()
## End(Not run)
```

---

stam.writeHTML *Write StAM Output in HTML*

Description

Write HTML output for various pieces of a structured analysis of microarray data for further interactive exploration.

Usage

```
stam.writeHTML(x, title = NULL, align = "left",
               outfile = "index.html", nonodes = FALSE, ...)
```

Arguments

x	the piece of the analysis for which HTML is to be generated. It may be of any of the following classes: stamNode, stamNet, stamCV, stamFit, stamPrediction or stamEval
title	the title to be used on the generated main page. A default is generated according to a few important parameters of x
align	alignment of the tatle (left, right, center)
outfile	the file where to store the generated HTML code unless x is of class stamEval. In the latter case, outfile is a directory where the the collection of files is stored.
nonodes	whether to generate (possibly lengthy) output for all nodes
...	further arguments passed to writeHTML calls.

Details

Use this function to generate HTML pages for further investigation of StAM results. The pages are interlinked and contain links to external resources such as the Gene Ontology and the Affymetrix website. Clickable maps are generated for the illustration of the model fit as well as the molecular symptoms image.

Author(s)

Claudio Lottaz

Examples

```
## Not run:
data(golubTrain.cv)
data(golubTrain.fit)

stam.writeHTML(golubTrain.cv)
stam.writeHTML(golubTrain.fit, nonodes=TRUE)
## End(Not run)
```

---

stamCV-class *Cross Validation Information Generated by StAM*

Description

Objects if this class are generated by stam.cv. It contains results of cross validated model fits generated in structured analysis of microarrays in order to choose graph shrinkage levels.

Objects from the Class

Objects can be created by calls of the form new("stamCV", exprs, classifications, beta, chip, root), but it is recommended the use the function stam.cv.

Slots

sample.labels: Object of class "character", names of samples

sample.classes: Object of class "character", class names for each sample

class.labels: Object of class "character", one name for each class

prior: Object of class "numeric", prior class probabilities according to prevalence

beta: Object of class "numeric", class weights, one per class

full.pamfit: Object of class "nsc", PAM fit on all probesets

probs: Object of class "array", matrix of cross validated prediction probabilities [samples x classes x nodes]

folds: Object of class "list", buckets used in cross validation

results: Object of class "data.frame", cross-validated root error rate, root performance and mean redundancy as well as remaining nodes and the accessible probesets for each delta

node.results: Object of class "list", performance, redundancy, sensitivity and specificity per node

max.leafdev: Object of class "numeric", performance of worst leaf node

deltas: Object of class "numeric", shrinkage candidates

See stamNet-class for slots chip, root, chippkg, GOpkg, nodes, leafs, inodes and probes.

Extends

Class "stamNet", directly.

Methods

print signature(x = "stamCV"): print information on cross validation

writeHTML signature(x = "stamCV"): generate HTML information on cross validation. However, using stam.writeHTML is recommended.

Author(s)

Claudio Lottaz

See Also

stam.cv, stamNet-class

---

stamEval-class *Results of a Complete StAM Analysis*

Description

Objects of this class are returned by ctam.evaluate. Results of all steps in a structured analysis of microarray data are stored.

Objects from the Class

Objects can be created by calls of the form `new("stamEval", exprs, cv, fit, pred, testset)`, but using `stam.evaluate` is recommended.

Slots

chip: Object of class `"character"`, the name of the chip for which the classifier net is generated.

exprs: Object of class `"matrix"`, the plain matrix of expression levels [probesets x samples], rownames and colnames are expected to be defined

cv: Object of class `"stamCV"`, store cross validation results

fit: Object of class `"stamFit"`, store model fit

pred: Object of class `"stamPrediction"`, store prediction results

testset: Object of class `"numeric"`, indices of samples treated as test set. The others are used for trainig.

Methods

writeHTML `signature(x = "stamEval")`: generate HTML information on a complete StAM analysis, but using `stam.writeHTML` is recommended.

Author(s)

Claudio Lottaz

See Also

`stam.evaluate`, `stamCV-class`, `stamFit-class`, `stamPrediction-class`, `stam.writeHTML`

---

stamFit-class *Model Fit Generated by StAM*

Description

Objects of this class hold a model fit as it is generated by structured analysis of microarray data. The function `stam.fit` returns such objects. They are handed on to `stam.predict` for predictions.

Objects from the Class

Objects can be created by calls of the form `new("stamFit", cv, exprs, alpha, delta, max.nodes, collapse.scnodes)`, but it is recommended to use the function `stam.fit`.

Slots

sample.classes: Object of class "character", class names for each sample

class.labels: Object of class "character", one name for each class

prior: Object of class "numeric", prior class probabilities according to prevalence

full.pamfit: Object of class "nsc", PAM fit on all probesets

alpha: Object of class "numeric", performance vs. redundancy weight(s)

beta: Object of class "numeric", class weights, one per class

delta: Object of class "numeric", shrinkage level given by the user

best.delta: Object of class "numeric", shrinkage level used for computing

default.delta: Object of class "numeric", default shrinkage level suggested by stam.fit

scores: Object of class "matrix", compound scores weighted using the provided alpha(s)

alpha.tab: Object of class "matrix", results comparing alphas

node.results: Object of class "list", performance, redundancy, sensitivity and specificity per node

collapse.scnodes: Object of class "logical", whether single children nodes are removed after shrinkage

See stamNet-class for slots chip, root, chippkg, GOpkg, nodes, leafs, inodes and probes.

Extends

Class "stamNet", directly.

Methods

print signature(x = "stamFit"): print information on the model fit.

writeHTML signature(x = "stamFit"): generate HTML information on the model fit, but using stam.writeHTML is recommended.

Author(s)

Claudio Lottaz

See Also

stam.fit, stamNet-class

`stamINode-class` *Inner Nodes in Classifier Nets by StAM*

Description

Objects of this class represent inner nodes of classifier nets as they are generated by structured analysis of microarray data. These nodes only contain children but never hold direct annotations of genes.

Objects from the Class

Objects can be created by calls of the form `new("stamINode", ID, GOidx, children)`.

ID is the GO identifier as character string

GOidx environment attributing indices to all GO indentifiers

children indeces of all children

Exactly one of `GOidx` and `children` must be defined, the other set to NULL (default)

Slots

`children:` Object of class `"numeric"`, indeces of the node's direct children in the Gene Ontology.

`weights:` Object of class `"numeric"`, weights attributed to the node's direct children.

See `stamNode-class` fot `ID`, `category`, `replacedParents` and `supNode`.

Extends

Class `"stamNode"`, directly.

Methods

print `signature(x = "stamINode")`: print information on the inner node.

writeHTML `signature(x = "stamINode")`: generate HTML information on the inner node.

Author(s)

Claudio Lottaz

See Also

`stamNode-class`, `stamLeaf-class`

stamLeaf-class *Leaf Nodes in Classifier Nets by StAM*

Description

Objects of this class represent leaf nodes in a classifier net as it is used by structured analysis of microarray data. These are the only nodes which have genes annotated.

Objects from the Class

Objects can be created by calls of the form `new("stamLeaf", ID, chip, probesidx)`.

`ID` is the GO identifier as character string

`chip` the name of the chip for which the classifier net is generated.

`probes` a character vector holding the identifiers of the probesets in the order as they occur in the expression matrix

Slots

`chip:` Object of class `"character"`, the name of the chip for which the classifier net is generated.

`probes:` Object of class `"numeric"`, for each probeset holds its position in the expression metrices to be analyzed

`pamfit:` Object of class `"nsc"`, holds the results of pamr.train restricted to the genes annotated to the current leaf node.

`delta:` Object of class `"numeric"`, stores the best delty for the local pamfit as determined by pamr.cv.

See `stamNode-class` fot `ID`, `category`, `replacedParents` and `supNode`.

Extends

Class `"stamNode"`, directly.

Methods

print `signature(x = "stamLeaf")`: print information on the leaf node.

writeHTML `signature(x = "stamLeaf")`: generate HTML information on the leaf node.

Author(s)

Claudio Lottaz

See Also

`stamNode-class`, `stamINode-class`

`stamNet-class` *Classifier Net for StAM*

Description

Objects of this class describe a network of classifiers as it is used by structured analysis of microarray data.

Objects from the Class

Objects can be created by calls of the form `new("stamNet", chip, root, probes)`, or by a call to `stam.net`.

`chip` the name of the chip for which the classifier net is generated.

`root` the GO identifier of the node where the generation of the classifier net is started.

`probes` a character vector holding the identifiers of the probesets in the order as they occur in the expression matrix

Slots

`chip`: Object of class `"character"`, the name of the chip for which the classifier net is generated.

`root`: Object of class `"character"`, the GO identifier of the node where the generation of the classifier net is started.

`chippkg`: Object of class `"character"`, information on the version of the meta data package for the chip

`GOpkg`: Object of class `"character"`, information on the version of the meata data package on the Gene Ontology

`nodes`: Object of class `"list"`, elements are of class `stamINode` or `stamLeaf`, one for each node in the classifier net.

`leafs`: Object of class `"numeric"`, indices of all leaf nodes in slot `nodes`

`inodes`: Object of class `"numeric"`, indices of all inner nodes in slot `nodes`

`probes`: Object of class `"environment"`, the corresponding index for each probeset in the expression matrices to be analyzed

Methods

print `signature(x = "stamNet")`: print information on the classifier net

writeHTML `signature(x = "stamNet")`: generate HTML information on the classifier net. However, using `stam.writeHTML` is recommended.

Author(s)

Claudio Lottaz

See Also

`stam.net`, `stamLeaf-class`, `stamINode-class`

stamNode-class *Nodes in a Classifier Net by StAM*

Description

Objects of this class and its subclasses represent single nodes in a classifier net as it is used by structured analysis of microarray data.

Objects from the Classes

Objects can be created by calls of the form `new("stamNode", ID, supNode)`.

ID is the GO identifier as character string

supNode indicates whether the node is a supplementary node to avoid nodes which have both, genes directly annotated and children in the stamNet.

Slots

`ID:` Object of class `"character"`, holds the GO identifier as character string

`category:` Object of class `"character"`, represents the GO ontology the node belongs (MF: molecular function, BP: biological process, CC: cellular component).

`replacedParents:` Object of class `"character"`, holds all GO identifiers of nodes which have been removed, because the had only one child.

`supNode:` Object of class `"logical"`, indicates whether the nodes has been added to avoid nodes which have both directly annotated genes as well as children in the classifier graph

Methods

getGOchildren `signature(x = "stamNode")`: returns the identifiers of the direct children of node x in the Gene Ontology.

getGOchildren `signature(x = "character")`: returns the identifiers of the direct children of the node with identifier x in the Gene Ontology.

getGOparents `signature(x = "stamNode")`: returns the identifiers of the direct parents of node x in the Gene Ontology.

getGOparents `signature(x = "character")`: returns the identifiers of the direct parents of the node with identifier x in the Gene Ontology.

getGOterm `signature(x = "stamNode")`: returns the GO term represented by node x.

getGOterm `signature(x = "character")`: returns the GO term represented by the node with identifier x.

print `signature(x = "stamNode")` print information on the node.

writeHTML `signature(x = "stamNode")` generate HTML information on the node.

Author(s)

Claudio Lottaz

See Also

`stamLeaf-class`, `stamINode-class`

stamPrediction-class

Results of Predictions by StAM

Description

Object of this class are returned by `stam.predict` and contain prediction results as they are computed by structured analysis of microarray data.

Objects from the Class

Objects can be created by calls of the form `new("stamPrediction", fit, expr.mat, cls, testset)`, but using `stam.predict` is recommended.

Slots

`chip`: Object of class `"character"`, the name of the chip for which the classifier net is generated.

`nodes`: Object of class `"list"`, elements are of class `stamINode` or `stamLeaf`, one for each remaining node in the classifier net.

`class.labels`: Object of class `"character"`, one name for each class

`best.delta`: Object of class `"numeric"`, shrinkage level used for computing

`cls`: Object of class `"character"`, class names for each sample

`probs`: Object of class `"array"`, matrix of prediction probabilities [samples x classes x nodes]

`predicts`: Object of class `"character"`, overall prediction for each sample

`testset`: Object of class `"numeric"`, indeces of samples which belong to the test set. The other samples are assumed to be the traiing set.

`node.results`: Object of class `"list"`, performance, redundancy, sensitivity and specificity per node

Methods

image `signature(x = "stamPrediction")`: molecular symptoms image, see `image.stamPrediction` for details

print `signature(x = "stamPrediction")`: print information on prediction

writeHTML `signature(x = "stamPrediction")`: generate HTML information on prediction, but using `stam.writeHTML` is recommended

Author(s)

Claudio Lottaz

See Also

`stam.predict`, `image.stamPrediction`, `stam.writeHTML`

Index