Package ‘capiu’
April 25, 2006

Version 0.1
Date 2006-04-10
Author Henning Redestig, Florian Sohler
Maintainer Henning Redestig <redestig@mpimp-golm.mpg.de>
Title Clustering with A-priori Information using Unsupervised decision trees
Depends R, ellipse, Biobase, mclust, MASS, GO, hu6800, cluster, e1071
Description Main function treeGen divides an expression matrix into different feature categories depending on (possibly) functional context and by evaluating each of these categories computes a sample wise clustering in tree like fashion.
License GPL version 2 (or later).
SaveImage no

R topics documented:

batchPca .......................... 2
capiu .................................. 2
Batched PCA ........................ 4
modelScores ........................ 4
node .................................. 5
pcaRes ................................ 6
udt .................................. 7
clusterGeneClass ................... 7
GOhelpers .......................... 8
golub ................................. 9
Lazy Model Bootstrap ................ 10
makeMap ................................ 11
nodeGen .............................. 12
pca .................................. 13
PlotPcs .............................. 14
scoreModels ........................ 15
slplot ............................... 16
toDot ................................ 17
Utilities for CAPIU ................... 18
batchPca  

Compute Batched PCA

Description

Provided a matrix with variables as rows, samples as columns and a map that defines which variables belong to a common class, batchPca can be used for computing a PCA for each of these classes. pca is used for the PCA analysis.

Usage

batchPca(object, map, verbose=TRUE,...)

Arguments

object  
Either a matrix or an exprSet
map  
A list corresponding to the different subsets of indices
verbose  
If TRUE, messages are printed at start and end of calculation
...  
Further arguments to pca

Value

A batchPcaRes object

Author(s)

Henning Redestig

See Also

cpa,prcomp,princomp

capiu  

Clustering using A Priori Information via Unsupervised decision trees.

Description

A method for generating unsupervised decision trees for e.g. gene expression or similar data that provide variables which can be grouped in classes for separate analysis.

Usage

capiu(x, map, labels=NULL,...)
Arguments

- x: A numerical matrix or exprSet with rows as variables and samples as columns.
- map: A list that specifies which class each variable belongs to.
- labels: Optional labels for the samples to be used in visualizations.
- ...: Further arguments to batchPca, scoreModels, nodeGen

Details

Consider a gene expression data matrix with expected subgroups in the samples and a mapping that defines classes of genes that alone can be assumed to provide more information about some biological process (or similar) than all genes do together. Then PCA models can be built for each of the gene classes alone and these models can then be examined to see if they provide a 'natural' partitioning of the samples such as 'treated' versus 'non-treated', the label classes that do so can then be hypothesised to be related to the biological reasons for the groupings. By then splitting the data using only this class and the re-partition the data in each node, the subgroups of samples can be (hopefully) be recovered in the leaves of the obtained tree.

capiu is a wrap-up function that combines batchPca, scoreModels and nodeGen to one interface for computing unsupervised decision trees. The result can be visualised using dot and toDot.

Unfortunately, capiu is a very slow function as it is 100% R and relies on a slow bootstrapping test. Setting K to a low value, correction to "none" and increasing minClustersize obviously makes it quicker but then no real checking for significance is done.

The extra parameters to give as ... are typically:

For **batchPca**: If the PCA preprocessing step of each gene class should scale the data or not by specifying scale. How many PC’s that should be extracted by eg specifying varLimit.

For **scoreModels**: How the gene classes should be scored by specifying scoretype

For **nodeGen**: How small the smallest cluster is expected to be be specifying minClusterSize, how similar to clusterings must be to be considered the same by specifying minSimilarity, how many bootstrap replicates should be calculated for the the significance test by specifying K

Value

A udt object.

Author(s)

Henning Redestig

See Also

nodeGen, batchPca, scoreModels

Examples

data(golub)
golubMap <- makeMap("hu6800", geneNames(golub), upperLimit=200)
#Warning, this takes a long time to execute!
## Not run:
golubTree <- capiu(golub,
modelScores

golubMap,
varLimit=0.3,
K=2000,
labels=paste(golub["ALL.AML"], golub["T.B.cell"]))

## End(Not run)
#You need dot installed to compile script generated by toDot
## Not run:
toDot(golubTree, "golubTree.dot")
## End(Not run)

Batched PCA

Description
This is a class representation of a batched PCA result

Creating Objects

new("pcaRes", pcaModels=[list of pca models])

Slots

pcaModels "list", list of pca models, each element is of class pcaRes.

Methods

print Print simple information about the batch result, how many there are and information about
the first model.
summary Print summary information about batch result
"[" Subsetting extracts one or a set of pca models from the batch. Result is either a new batchP-
cRes or a pcaRes if single element.
length Amount of elements in this batch
slplot Plot a scores and loadings plot for this object, see slplot

modelScores

Class for representing score object from a batchPcaRes

Description
This is a class representation of scores from a batched PCA result

Creating Objects

new("modelScores", scores=[the scores for each model], clusterings=[the
clusterings computed based on each model alone], sizes=[the sizes of
the gene classes used], certainty=[how certain the classification of
each sample was], scoretype=[the scoretype used])
node

Slots

scores "numeric", vector with one score for each gene class

clusterings "matrix", each column in this matrix contains the clustering vector from the corresponding gene class.

sizes "numeric", vector showing the sizes of the different gene classes (amount of genes in them)

certainty "matrix", each column in this matrix contains the certainty levels for the clustering of each sample. 1 is completely sure (default for e.g. K-means)

scortype "character", the scoretype used to calculate this model score

Methods

names Extract the names of the gene classes used

"[" Subsetting extracts one or a set of scores from the batch.

node

Class for representing a node in a CAPIU tree

Description

This is a class representation of a node in a CAPIU tree

Creating Objects

new("node", ranks=[the ranks of the gene classes used], classes=[identifiers for the gene classes used], clustering=[the consensus clustering], children=[amount of children to this node], scores=[the scores of the used gene classes], sizes=[the sizes of the used gene classes], members=[the samples belonging to this node], type=[what kind of node this is])

Slots

ranks "numeric", the ranks of the gene classes in this node. Rank one means the gene class got the highest score of all

classes "character", identifiers of the gene classes in this node

clustering "numeric", the consensus clustering vector calculated from all gene classes in this node

children "numeric", the amount of children this node has. Zero if the node is a leaf

scores "numeric", the scores of the gene classes in this node

sizes "numeric", the sizes of the gene classes in this node

members "list", of length one if node otherwise same length as amount of children indicating which samples goes where in the tree

type "character", Either "node" or "leaf"
pcaRes

Class for representing a PCA result

Description

This is a class representation of a PCA result.

Creating Objects

new("pcaRes", scores=[the scores], loadings=[the loadings], k=[amount of PCs], R2cum=[cumulative R2], nobs=[amount of observations], nvar=[amount of variables], R2=[R2 for each individual PC], sdev=[stdev for each individual PC], scaled=[which method was used to scale data], centered=[was data centered], varLimit=[what variance limit was exceeded], method=[method used to calculate PCA], subset=[subset of variables of data used], missing=[amount of NAs], center=[original means])

Slots

scores "matrix", the calculated scores
loadings "matrix", the calculated loadings
R2cum "numeric", the cumulative R2 values
sdev "numeric", the individual standard deviations
R2 "numeric", the individual R2 values
nobs "numeric", amount of observations
nvar "numeric", amount of variables
centered "logical", data was centered or not
center "numeric", the original variable centers
subset "numeric", the subset of variables used
varLimit "numeric", the exceeded variance limit
scaled "character", the scaled method used
k "numeric", the amount of calculated PCs
method "character", the method used to perform PCA
missing "numeric", the total amount of missing values in original data

Methods

print Print function
summary Extract information about PC relevance
screeplot Plot a barplot of standard deviations for PCs
slplot Make a side by side score and loadings plot
Class for representing of a CAPIU tree

Description
This is a class representation of a CAPIU tree. It used a perhaps bit weird way of representing a
graph where the first node points to the next 'children' nodes. The next node points to its children
calculated with an offset depending on the amount of children the previous node had and so on.

Creating Objects
new("node", nodes=[list of node], labels=[optional labels of the samples
in the dataset])

Slots
nodes "list", each element is of class "node", together they make up a tree
labels "character", optional labels for each sample used to generate the tree. Should be identifiable
by their names.

Methods
getSizes(object, i) Get sizes for node i
getScores(object, i) Get scores for node i
getGeneClasses(object, i) Get gene classes for node i
getRanks(object, i) Get ranks for node i
getChildren(object, i) Get amount of children for node i
getMembers(object, i, j=NULL, translate=TRUE) Get members for node i, branch j and possibly translate the using the labels slot
type(object, i) Get type of node i
addNodes(object, node) Add node to object to get a larger tree
length(x) Get length of this tree

clusterGeneClass Cluster the samples using information from a gene class.

Description
This function provides functionality for clustering the samples microarray experiment using information from one gene class only. The gene class should be pre-processed with PCA.

Usage
clusterGeneClass(object, clNum=2, scoretype=c("mclust", "weightedsilhouette", "score")
Arguments

object A matrix containing raw gene expression measurements or a pcaRes object.
clNum The amount of clusters to search for. More than two is experimental.
scoretype The scoretype to use:

mclust Use mixture model clustering to fit Gaussian mixture model. Score is the log-likelihood ratio between uni-modal and clNum-model clusterings
silhouette Clustering is done by PAM from the cluster package and the score is the average silhouette width from the clusters.
weightedsilhouette Same as silhouette but final score is weighted by the entropy of the cluster sizes to give preference for similar sized clusters

... Pass through arguments

Value

A list with clustering, certainty, score, clNum and size of the geneclass

Author(s)

Henning Redestig

See Also

pca

Examples

data(golubMergeSub)
myRandomMapping <- sapply(1:10, function(x) sample(1:1000, sample(5:200)))
models <- batchPca(golubMergeSub, myRandomMapping)
score <- clusterGeneClass(models[1], 2, "mclust")

GO helpers

Small convenience functions for dealing with GO

Description

getFlavorSubmap extracts a submap of a large GO map. Selects all items whose names are GO identifiers that belong to the <flavor> part of the GO hierarchy. isFlavor checks if a GO class is of requested flavor and getGOTerm extracts the term corresponding to a given GO identifier.

Usage

getFlavorSubmap(map, flavor=c("CC", "MF", "BP"))
isFlavor(goclass, flavor)
getGOTerm(goclass)
Arguments

- **map**: Parent mapping, typically obtained from the GO package.
- **flavor**: One of CC, MF or BP for cellular component, molecular function or biological process.
- **goclass**: A GO identifier (or vector thereof), e.g. "GO:0006412"

Value

- **getFlavorSubmap**: Returns the submap, a list where each component is a character vector containing the identifiers of the members in that class.
- **isFlavor**: logical
- **getGOTerm**: Character vector with corresponding terms

Author(s)

Henning Redestig

See Also

GO

Examples

```r
require("GO") || stop("GO unavailable")
# Convert a GO environment object to a list
map <- as.list(GOLOCUSID)
submap <- getFlavorSubmap(map, "BP")
isFlavor("GO:0006412", "BP")
getGOTerm("GO:0006412")
```

---

**golub**  
*The famous Leukemia data set*

**Description**

This is a shortened version of the training data set used by Golub et al. taken from the golubEsets package.

**Usage**

data(golub)

**Format**

An exprSet object.

**Source**

golubEsets
Lazy Model Bootstrap

References


Lazy Model Bootstrap

Perform a simplified version of the bootstrap test for modelScores

Description

Can be used to test how much better a score for a gene class is than one would expect if the data came from a zero mean multivariate normal distribution with Sigma equal to that of the gene class (or the by PCA summarized gene class)

Usage

lazyModelBootstrap(models, scores, K=2000, null=1, alt=2, correction="BY", verbose=TRUE, ...)

Arguments

models A batchPcaRes object
scores A modelScores object
K The amount of replica to consider
null The modality of the null distribution
alt The modalitve of the alternative hypothesis
 correction The P-value correction to use see p.adjust
verbose Print information about progress
... Passes through arguments

Details

This method computes a set of samples, for each cardinality of the batchPcaRes object, from a multivariate distribution under the null-hypothesis with mean zero and Sigma sampled from the set of Sigmas in the batchPcaRes object. So if the elements of batchPcaRes has either 1 or 2 PCs, 2 * K replicates will be computed. These are then scored the same way the models were scored and a p-value is calculated as the amount of samples greater than the observed under null hypothesis divided by K.

Value

A list of corrected and uncorrected p-values.

Author(s)

Henning Redestig
Examples

data(golubMergeSub)
myRandomMapping <- sapply(1:10, function(x) sample(1:1000, sample(5:200)))
models <- batchPca(golubMergeSub, myRandomMapping)
scores <- scoreModels(models)
pvals <- lazyModelBootstrap(models, scores, K=100)

Description

Makes a mapping given a chip name and a set of probes to use. Parents identical to any of their children are removed. `getGOTerm` extracts the term corresponding to a given GO identifier.

Usage

makeMap(chip, probes, ontology="BP", upperLimit=300, lowerLimit=5)

Arguments

- **chip**: The used chip
- **probes**: The set of probes that were used on that chip.
- **ontology**: The ontology from GO that should be used
- **upperLimit**: The largest allowed gene class
- **lowerLimit**: The smallest allowed gene class

Value

A list with elements corresponding to the indices in the probes vector.

Author(s)

Henning Redestig

See Also

GO

Examples

```r
require("GO") || stop("GO unavailable")
# Convert a GO environment object to a list
data(golubMergeSub)
map <- makeMap("hu6800", geneNames(golubMergeSub))
```
nodeGen

Generate a node in a CAPIU tree

Description
A method for creating a node in a decision tree based on a modelScore object. This method is normally not used directly but called from within capiu.

Usage
nodeGen(scores, models=NULL, map=NULL, leaf=FALSE, minSimilarity=0.80, eset=NULL, minClusterSize=4, alfa=0.05, verbose=interactive(), ...)

Arguments
scores  a modelScores object
models  The batchPcaRes object that was used to calculate x. Needed if the node should be a leaf or if cross-validation is desired.
map    The mapping from genes to gene classes that was used
leaf   logical, should this node be a leaf
minSimilarity
     Minimum corrected Rand index defining how similar to clusterings must be in order to get merged in to the same node
eset   The original data
minClusterSize
    How small a final cluster is allowed to be
alfa   Significance level for the bootstrap test
verbose Print some messages to indicate progress
...    Further arguments passed on to lazyModelBootstrap

Value
A node object

Author(s)
Henning Redestig

See Also
modelScores, treeGen

Examples
Perform principal component analysis

Description

Can be used for computing PCA on a numeric matrix using either the standard prcomp method or NIPALS (Nonlinear Iterative Partial Least Squares) algorithm which is an iterative approach for estimating the principal components extracting them one at a time.

Usage

pca(object, k=2, scale=c("none", "pareto", "vector", "UV"), center=TRUE, limit=1e-6, maxiterations=5000, varLimit=1, subset=numeric(), method=c("svd", "nipals"), verbose=FALSE, ...)

Arguments

- **object**: Numerical matrix with (or on object convertible to such) with samples in rows and variables as columns. Also takes exprSet in which case the transposed exprs slot is used.
- **k**: Number of components that should be extracted.
- **center**: Indicates if the matrix should be mean centered or not.
- **limit**: The limit condition for judging if the algorithm has converged or not, specifically if a new iteration is done if \((T_{old} - T)^T (T_{old} - T) > limit\).
- **maxiterations**: Defines how many iterations can be done before the algorithm should abort (happens almost exclusively when there were some wrong in the input data).
- **verbose**: If TRUE a simple progress bar is displayed
- **varLimit**: If defined, \(k\) is ignored and the algorithm continues until the explained variance is greater than this number (<1).
- **scale**: One of "UV" (unit variance \(a = a/\sigma_a\)) "vector" (vector normalization \(b = b/||b||\)), "pareto" or "none" to indicate which scaling should be used to scale the matrix with \(a\) variables and \(b\) samples.
- **subset**: For convenience one can pass a large matrix but only use the variable specified as subset. Can be colnames or indices.
- **method**: One of "svd" or "nipals". "svd" makes pca use prcomp
- **...**: Pass through arguments

Details

NIPALS is capable of handling missing values (by simply leaving them out of the dot-product) provided they are well scattered in the matrix and relatively few. The NIPALS and SVD should (except for rounding errors) yield identical result but SVD is much faster and should be used as default. pca therefore does not add much to prcomp except a maybe more convenient way of representing the result.

Value

A pcaRes object.
Author(s)
Henning Redestig

References

See Also
prcomp, princomp

Examples

data(iris)
## Usually some kind of scaling is appropriate
pcIr <- pca(iris[,1:4], scale="UV", method="nipals")
pcIr <- pca(iris[,1:4], scale="UV", method="svd")
## Get a short summary on the calculated model
summary(pcIr)
## Scores and loadings plot
slplot(pcIr, sl=as.character(iris[,5]))

Description
A function that can be used to visualize many PCs plotted against each other

Usage

plotPcs(object, pc=1:object@k, scoresLoadings=c(TRUE, FALSE),...)

Arguments

object
a pcaRes object

pc
which pcs to plot

scoresLoadings
If scores XOR loadings should be plotted

... Further arguments to slplot

Details
Uses par to provide side-by-side plots so it does not work with Sweave.

Value
None, used for side effect.
scoreModels

Author(s)
Henning Redestig

See Also
prcomp, pca, princomp, slplot

Examples

data(iris)
pcIr <- pca(iris[,1:4], k=3, scale="UV", method="svd")
plotPcs(pcIr)

scoreModels

Compute scores for the computed models in a batchPca object

Description
This method computes the clusterings and scores of choice for each element (model) in a batchPca object. These scores can then be used to find the best model according to the measure of choice. In an unsupervised decision tree setting this model is used for splitting the samples.

Usage
scoreModels(x, scoretype=c("mclust", "silhouette", "weightedsilhouette"), verbose=interactive(), ...)

Arguments
x A batchPca object
scoretype Which score to use; implemented scores are defined as such:
mclust Fit uni-modal and bi-modal distributions to a uni-variate distribution using the mclust package and compute score by log-likelihood between the two obtained models
silhouette PAM is used to cluster data and average Silhouette Index of all clusters is the total score.
weightedsilhouette Same as silhouette but multiplied by the relative entropy of the cluster sizes to give small preference for equal sized clusters.
verbose If TRUE a message is printed a start and end of calculation
... Only used for passing through arguments

Details
The clusterings are done with pam from the package cluster except when modelSelect is used in which case the EM-algorithm EMclust from the mclust package is used.

Value
A modelScores object
Author(s)
Henning Redestig

See Also
silhouette, pam, batchPca, pca

Examples

data(golubMergeSub)
myRandomMapping <- sapply(1:10, function(x) sample(1:1000, sample(5:200)))
models <- batchPca(golubMergeSub, myRandomMapping)
scores <- scoreModels(models)

slplot

Plot a side by side scores and loadings plot

Description
A common way of representing PCA result for two component

Usage

slplot(object, pcs=c(1,2), scoresLoadings=c(TRUE, TRUE),
sl="def", ll="def", hotelling=0.95, rug=TRUE,...)

Arguments

object  a pcaRes object
pcs     which two pcs to plot
scoresLoadings
    Which should be shown scores and or loadings
sl      labels to plot in the scores plot
ll      labels to plot in the loadings plot
hotelling  confidence interval for ellipse
rug      logical, rug x axis or not
...     Further arguments to plot functions

Details
Uses layout instead of par to provide side-by-side so it works with Sweave.

Value
None, used for side effect.

Author(s)
Henning Redestig
See Also

prcomp, pca, princomp

Examples

data(iris)
pcIr <- pca(iris[,1:4], scale="UV", method="svd")
slplot(pcIr, sl=NULL, pch=5, col=as.integer(iris[,5]))

toDot

Write dot scripts for an unsupervised decision tree.

Description

A convenience function for generating dot scripts for easy visualization of unsupervised decision trees.

Usage

toDot(x, filename, graphname="G", goTerm=TRUE, goId=TRUE,
compile=TRUE, useLabels=TRUE)

Arguments

x The tree to use.
filename The filename of the script (should end with ".dot")
graphname What to call this graph in the script, usually irrelevant.
goTerm Indicates if the name of each class should be added in each node, e.g. 'protein biosynthesis' if the map used was a mapping to GO.
goId Indicates if the identifier of each class should added in each node, e.g. 'GO:0006412' if the map used was a mapping to GO.
compile Logical, indicates if dot shall be called using system to compile the generated script to a PNG file.
useLabels Logical, indicates if the labels in the udt object should be used for the leaves or not.

Value

None, used for the side effect.

Author(s)

Henning Redestig

References

Description

A progress bar to typically be used in for-loop control statements

Usage

progBar(i, to, start=1, dots=50)

Arguments

- **i**: Present iteration
- **to**: Maximum of this loop
- **start**: Where loop started
- **dots**: Desired length of progress bar (only approximate)

Value

None, used for its side effect.

Author(s)

Henning Redestig

Examples

```r
for(i in 1:1000) {
  progBar(i, 1000)
}
```
Index

*Topic classes
  Batched PCA, 4
  modelScores, 4
  node, 5
  pcaRes, 5

*Topic datasets
  golub, 9

*Topic list
  GOhelpers, 8
  makeMap, 10

*Topic multivariate
  batchPca, 1
  Lazy Model Bootstrap, 9
  pca, 12
  PlotPcs, 13
  slplot, 15

*Topic tree
  capiu, 2
  clusterGeneClass, 7
  nodeGen, 11
  scoreModels, 14
  toDot, 16
  udt, 6

*Topic utilities
  Utilities for CAPIU, 17
  [,batchPcaRes-method (Batched PCA), 4
  [,modelScores-method (modelScores), 4
  addNodes (udt), 6
  addNodes, udt-method (udt), 6
  Batched PCA, 4
  batchPca, 1
  batchPca, exprSet-method (batchPca), 1
  batchPca, matrix-method (batchPca), 1
  batchPcaRes (Batched PCA), 4
  batchPcaRes-class (Batched PCA), 4
  capiu, 2
  clusterGeneClass, 7
  clusterGeneClass, matrix-method (clusterGeneClass), 7
  clusterGeneClass, pcaRes-method (clusterGeneClass), 7
  getChildren (udt), 6
  getChildren, udt-method (udt), 6
  getFlavorSubmap (GOhelpers), 8
  getGeneClasses (udt), 6
  getGeneClasses, udt-method (udt), 6
  getGOTerm (GOhelpers), 8
  getMembers (udt), 6
  getMembers, udt-method (udt), 6
  getRanks (udt), 6
  getRanks, udt-method (udt), 6
  getScores (udt), 6
  getScores, udt-method (udt), 6
  getSizes (udt), 6
  getSizes, udt-method (udt), 6
  getType (udt), 6
  getType, udt-method (udt), 6
  GOhelpers, 8
  golub, 9
  isFlavor (GOhelpers), 8
  Lazy Model Bootstrap, 9
  lazyModelBootstrap (Lazy Model Bootstrap), 9
  length, batchPcaRes-method (Batched PCA), 4
  length, udt-method (udt), 6
  makeMap, 10
  modelScores, 4
  modelScores-class (modelScores), 4
  names, modelScores-method (modelScores), 4
  node, 5
  node-class (node), 5
  nodeGen, 11
  pca, 12
  pcaRes, 5
pcaRes-class (pcaRes), 5
PlotPcs, 13
plotPcs (PlotPcs), 13
print, batchPcaRes-method
   (Batched PCA), 4
print, pcaRes-method (pcaRes), 5
progBar (Utilities for CAPIU), 17
scoreModels, 14
slplot, 15
slplot, batchPcaRes-method
   (Batched PCA), 4
slplot, pcaRes-method (pcaRes), 5
summary, batchPcaRes-method
   (Batched PCA), 4
summary, pcaRes-method (pcaRes), 5
toDot, 16
udt, 6
udt-class (udt), 6
Utilities for CAPIU, 17