

# Package ‘CoGAPS’

April 12, 2022

**Version** 3.14.0

**Date** 2020-12-30

**Title** Coordinated Gene Activity in Pattern Sets

**Author** Thomas Sherman, Wai-shing Lee, Conor Kelton, Ondrej Maxian, Jacob Carey, Genevieve Stein-O'Brien, Michael Considine, Maggie Wodicka, John Stansfield, Shawn Sivy, Carlo Colantuoni, Alexander Favorov, Mike Ochs, Elana Fertig

**Description** Coordinated Gene Activity in Pattern Sets (CoGAPS) implements a Bayesian MCMC matrix factorization algorithm, GAPS, and links it to gene set statistic methods to infer biological process activity. It can be used to perform sparse matrix factorization on any data, and when this data represents biomolecules, to do gene set analysis.

**Maintainer** Elana J. Fertig <ejfertig@jhmi.edu>, Thomas D. Sherman <tomsherman159@gmail.com>, Melanie L. Loth <mloth1@jhmi.edu>

**Depends** R (>= 3.5.0)

**Imports** BiocParallel, cluster, methods, gplots, graphics, grDevices, RColorBrewer, Rcpp, S4Vectors, SingleCellExperiment, stats, SummarizedExperiment, tools, utils, rhdf5

**Suggests** testthat, knitr, rmarkdown, BiocStyle

**LinkingTo** Rcpp, BH

**VignetteBuilder** knitr

**LazyLoad** true

**License** BSD\_3\_clause + file LICENSE

**biocViews** GeneExpression, Transcription, GeneSetEnrichment, DifferentialExpression, Bayesian, Clustering, TimeCourse, RNASeq, Microarray, MultipleComparison, DimensionReduction, ImmunoOncology

**NeedsCompilation** yes

**RoxygenNote** 7.1.0

**Encoding** UTF-8

**Collate** 'class-CogapsParams.R' 'CoGAPS.R' 'DistributedCogaps.R'  
 'HelperFunctions.R' 'Package.R' 'RcppExports.R' 'SubsetData.R'  
 'class-CogapsResult.R' 'methods-CogapsParams.R'  
 'methods-CogapsResult.R'

**git\_url** <https://git.bioconductor.org/packages/CoGAPS>

**git\_branch** RELEASE\_3\_14

**git\_last\_commit** f937a7c

**git\_last\_commit\_date** 2021-10-26

**Date/Publication** 2022-04-12

## R topics documented:

CoGAPS-package . . . . .	3
binaryA . . . . .	3
buildReport . . . . .	4
calcCoGAPSStat . . . . .	5
calcGeneGSStat . . . . .	6
calcZ . . . . .	7
checkpointsEnabled . . . . .	7
CoGAPS . . . . .	8
CogapsParams . . . . .	9
CogapsParams-class . . . . .	10
CogapsResult-class . . . . .	11
compiledWithOpenMPsupport . . . . .	11
computeGeneGSProb . . . . .	12
findConsensusMatrix . . . . .	13
getAmplitudeMatrix . . . . .	13
getClusteredPatterns . . . . .	14
getCorrelationToMeanPattern . . . . .	14
getFeatureLoadings . . . . .	15
getMeanChiSq . . . . .	16
getOriginalParameters . . . . .	16
getParam . . . . .	17
getPatternMatrix . . . . .	18
getRetinaSubset . . . . .	18
getSampleFactors . . . . .	19
getSubsets . . . . .	19
getUnmatchedPatterns . . . . .	20
getVersion . . . . .	21
GIST.data_frame . . . . .	21
GIST.matrix . . . . .	21
GIST.result . . . . .	22
GIST.uncertainty . . . . .	22
GWCoGAPS . . . . .	22
initialize,CogapsParams-method . . . . .	24
initialize,CogapsResult-method . . . . .	24

*binaryA* 3

patternMarkers . . . . .	25
plotPatternMarkers . . . . .	26
plotResiduals . . . . .	27
reconstructGene . . . . .	27
scCoGAPS . . . . .	28
setAnnotationWeights . . . . .	29
setDistributedParams . . . . .	30
setFixedPatterns . . . . .	31
setParam . . . . .	32

**Index** 33

---

CoGAPS-package                      *CoGAPS: Coordinated Gene Activity in Pattern Sets*

---

### Description

CoGAPS implements a Bayesian MCMC matrix factorization algorithm, GAPS, and links it to gene set statistic methods to infer biological process activity. It can be used to perform sparse matrix factorization on any data, and when this data represents biomolecules, to do gene set analysis.

Package: CoGAPS  
Type: Package  
Version: 2.99.0  
Date: 2018-01-24  
License: LGPL

### Author(s)

Maintainer: Elana J. Fertig <ejfertig@jhmi.edu>, Michael F. Ochs <ochsm@tcnj.edu>

### References

Fertig EJ, Ding J, Favorov AV, Parmigiani G, Ochs MF. CoGAPS: an R/C++ package to identify patterns and biological process activity in transcriptomic data. *Bioinformatics*. 2010 Nov 1;26(21):2792-3

---

*binaryA*                                      *binary heatmap for standardized feature matrix*

---

### Description

creates a binarized heatmap of the A matrix in which the value is 1 if the value in Amean is greater than threshold \* Asd and 0 otherwise

**Usage**

```
binaryA(object, threshold = 3)

## S4 method for signature 'CogapsResult'
binaryA(object, threshold = 3)
```

**Arguments**

object	an object of type CogapsResult
threshold	the number of standard deviations above zero that an element of Amean must be to get a value of 1

**Value**

plots a heatmap of the A Matrix

**Examples**

```
data(GIST)
# to expensive to call since it plots
# binaryA(GIST.result, threshold=3)
```

---

buildReport

*Information About Package Compilation*

---

**Description**

Information About Package Compilation

**Usage**

```
buildReport()
```

**Details**

returns information about how the package was compiled, i.e. which compiler/version was used, which compile time options were enabled, etc...

**Value**

string containing build report

**Examples**

```
CoGAPS::buildReport()
```

---

calcCoGAPSStat      *calculate statistic on sets of measurements (genes) or samples*

---

## Description

calculates a statistic to determine if a pattern is enriched in a particular set of measurements or samples.

## Usage

```
calcCoGAPSStat(  
  object,  
  sets = NULL,  
  whichMatrix = "featureLoadings",  
  numPerm = 1000,  
  ...  
)  
  
## S4 method for signature 'CogapsResult'  
calcCoGAPSStat(  
  object,  
  sets = NULL,  
  whichMatrix = "featureLoadings",  
  numPerm = 1000,  
  ...  
)
```

## Arguments

object	an object of type CogapsResult
sets	list of sets of measurements/samples
whichMatrix	either "featureLoadings" or "sampleFactors" indicating which matrix to calculate the statistics for
numPerm	number of permutations to use when calculating p-value
...	handles old arguments for backwards compatibility

## Value

gene set statistics for each column of A

---

calcGeneGSStat      *probability gene belongs in gene set*

---

### Description

calculates the probability that a gene listed in a gene set behaves like other genes in the set within the given data set

### Usage

```
calcGeneGSStat(  
  object,  
  GStoGenes,  
  numPerm,  
  Pw = rep(1, ncol(object@featureLoadings)),  
  nullGenes = FALSE  
)  
  
## S4 method for signature 'CogapsResult'  
calcGeneGSStat(  
  object,  
  GStoGenes,  
  numPerm,  
  Pw = rep(1, ncol(object@featureLoadings)),  
  nullGenes = FALSE  
)
```

### Arguments

object	an object of type CogapsResult
GStoGenes	data.frame or list with gene sets
numPerm	number of permutations for null
Pw	weight on genes
nullGenes	logical indicating gene adjustment

### Value

gene similarity statistic

---

calcZ                    *compute z-score matrix*

---

**Description**

calculates the Z-score for each element based on input mean and standard deviation matrices

**Usage**

```
calcZ(object, whichMatrix)

## S4 method for signature 'CogapsResult'
calcZ(object, whichMatrix)
```

**Arguments**

object                an object of type CogapsResult  
whichMatrix          either "featureLoadings" or "sampleFactors" indicating which matrix to calculate the z-score for

**Value**

matrix of z-scores

**Examples**

```
data(GIST)
featureZScore <- calcZ(GIST.result, "featureLoadings")
```

---

checkpointsEnabled    *Check if package was built with checkpoints enabled*

---

**Description**

Check if package was built with checkpoints enabled

**Usage**

```
checkpointsEnabled()
```

**Value**

true/false if checkpoints are enabled

**Examples**

```
CoGAPS::checkpointsEnabled()
```

---

 CoGAPS

*CoGAPS Matrix Factorization Algorithm*


---

### Description

calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix

### Usage

```
CoGAPS(
  data,
  params = new("CogapsParams"),
  nThreads = 1,
  messages = TRUE,
  outputFrequency = 1000,
  uncertainty = NULL,
  checkpointOutFile = "gaps_checkpoint.out",
  checkpointInterval = 0,
  checkpointInFile = NULL,
  transposeData = FALSE,
  BPPARAM = NULL,
  workerID = 1,
  asynchronousUpdates = TRUE,
  nSnapshots = 0,
  snapshotPhase = "sampling",
  ...
)
```

### Arguments

data	File name or R object (see details for supported types)
params	CogapsParams object
nThreads	maximum number of threads to run on
messages	T/F for displaying output
outputFrequency	number of iterations between each output (set to 0 to disable status updates, other output is controlled by @code messages)
uncertainty	uncertainty matrix - either a matrix or a supported file type
checkpointOutFile	name of the checkpoint file to create
checkpointInterval	number of iterations between each checkpoint (set to 0 to disable checkpoints)
checkpointInFile	if this is provided, CoGAPS runs from the checkpoint contained in this file



transposeData	T/F for transposing data while reading it in - useful for data that is stored as samples x genes since CoGAPS requires data to be genes x samples
BPPARAM	BiocParallel backend
workerID	if calling CoGAPS in parallel the worker ID can be specified, only worker 1 prints output and each worker outputs when it finishes, this is not necessary when using the default parallel methods (i.e. distributed CoGAPS) but only when the user is manually calling CoGAPS in parallel
asynchronousUpdates	enable asynchronous updating which allows for multi-threaded runs
nSnapshots	how many snapshots to take in each phase, setting this to 0 disables snapshots
snapshotPhase	which phase to take snapshots in e.g. "equilibration", "sampling", "all"
...	allows for overwriting parameters in params

### Details

The supported R types are: matrix, data.frame, SummarizedExperiment, SingleCellExperiment. The supported file types are csv, tsv, and mtx.

### Value

CogapsResult object

### Examples

```
# Running from R object
data(GIST)
resultA <- CoGAPS(GIST.data_frame, nIterations=25)

# Running from file name
gist_path <- system.file("extdata/GIST.mtx", package="CoGAPS")
resultB <- CoGAPS(gist_path, nIterations=25)

# Setting Parameters
params <- new("CogapsParams")
params <- setParam(params, "nPatterns", 3)
resultC <- CoGAPS(GIST.data_frame, params, nIterations=25)
```

---

CogapsParams

*CogapsParams constructor*

---

### Description

create a CogapsParams object

### Usage

```
CogapsParams(...)
```

**Arguments**

... parameters for the initialization method

**Value**

CogapsParams object

**Examples**

```
params <- CogapsParams(nPatterns=10)
params
```

---

CogapsParams-class      *CogapsParams*

---

**Description**

Encapsulates all parameters for the CoGAPS algorithm

**Slots**

nPatterns number of patterns CoGAPS will learn  
 nIterations number of iterations for each phase of the algorithm  
 alphaA sparsity parameter for feature matrix  
 alphaP sparsity parameter for sample matrix  
 maxGibbsMassA atomic mass restriction for feature matrix  
 maxGibbsMassP atomic mass restriction for sample matrix  
 seed random number generator seed  
 sparseOptimization speeds up performance with sparse data (roughly >80 default uncertainty distributed either "genome-wide" or "single-cell" indicating which distributed algorithm should be used  
 nSets [distributed parameter] number of sets to break data into  
 cut [distributed parameter] number of branches at which to cut dendrogram used in pattern matching  
 minNS [distributed parameter] minimum of individual set contributions a cluster must contain  
 maxNS [distributed parameter] maximum of individual set contributions a cluster can contain  
 explicitSets [distributed parameter] specify subsets by index or name  
 samplingAnnotation [distributed parameter] specify categories along the rows (cols) to use for weighted sampling  
 samplingWeight [distributed parameter] weights associated with samplingAnnotation  
 subsetIndices set of indices to use from the data  
 subsetDim which dimension (1=rows, 2=cols) to subset



---

computeGeneGSProb      *compute gene probability*

---

### Description

Computes the p-value for gene set membership using the CoGAPS-based statistics developed in Fertig et al. (2012). This statistic refines set membership for each candidate gene in a set specified in GSGenes by comparing the inferred activity of that gene to the average activity of the set.

### Usage

```
computeGeneGSProb(  
  object,  
  GStoGenes,  
  numPerm = 500,  
  Pw = rep(1, ncol(object@featureLoadings)),  
  PwNull = FALSE  
)  
  
## S4 method for signature 'CogapsResult'  
computeGeneGSProb(  
  object,  
  GStoGenes,  
  numPerm = 500,  
  Pw = rep(1, ncol(object@featureLoadings)),  
  PwNull = FALSE  
)
```

### Arguments

object	an object of type CogapsResult
GStoGenes	data.frame or list with gene sets
numPerm	number of permutations for null
Pw	weight on genes
PwNull	- logical indicating gene adjustment

### Value

A vector of length GSGenes containing the p-values of set membership for each gene contained in the set specified in GSGenes.

---

findConsensusMatrix *find the consensus pattern matrix across all subsets*

---

**Description**

find the consensus pattern matrix across all subsets

**Usage**

```
findConsensusMatrix(unmatchedPatterns, gapsParams)
```

**Arguments**

unmatchedPatterns  
list of all unmatched pattern matrices from initial run of CoGAPS

gapsParams  
list of all CoGAPS parameters

**Value**

matrix of consensus patterns

---

getAmplitudeMatrix *return Amplitude matrix from CogapsResult object*

---

**Description**

return Amplitude matrix from CogapsResult object

**Usage**

```
getAmplitudeMatrix(object)  
  
## S4 method for signature 'CogapsResult'  
getAmplitudeMatrix(object)
```

**Arguments**

object  
an object of type CogapsResult

**Value**

amplitude matrix

**Examples**

```
data(GIST)  
amplitudeMatrix <- getAmplitudeMatrix(GIST.result)
```

---

getClusteredPatterns *return clustered patterns from set of all patterns across all subsets*

---

**Description**

return clustered patterns from set of all patterns across all subsets

**Usage**

```
getClusteredPatterns(object)

## S4 method for signature 'CogapsResult'
getClusteredPatterns(object)
```

**Arguments**

object            an object of type CogapsResult

**Value**

CogapsParams object

**Examples**

```
data(GIST)
clusteredPatterns <- getClusteredPatterns(GIST.result)
```

---

getCorrelationToMeanPattern  
*return correlation between each pattern and the cluster mean*

---

**Description**

return correlation between each pattern and the cluster mean

**Usage**

```
getCorrelationToMeanPattern(object)

## S4 method for signature 'CogapsResult'
getCorrelationToMeanPattern(object)
```

**Arguments**

object            an object of type CogapsResult

**Value**

CogapsParams object

**Examples**

```
data(GIST)
corrToMeanPattern <- getCorrelationToMeanPattern(GIST.result)
```

---

`getFeatureLoadings`      *return featureLoadings matrix from CogapsResult object*

---

**Description**

return featureLoadings matrix from CogapsResult object

**Usage**

```
getFeatureLoadings(object)

## S4 method for signature 'CogapsResult'
getFeatureLoadings(object)
```

**Arguments**

object                  an object of type CogapsResult

**Value**

featureLoadings matrix

**Examples**

```
data(GIST)
fLoadings <- getFeatureLoadings(GIST.result)
```

---

`getMeanChiSq`                    *return chi-sq of final matrices*

---

**Description**

return chi-sq of final matrices

**Usage**

```
getMeanChiSq(object)

## S4 method for signature 'CogapsResult'
getMeanChiSq(object)
```

**Arguments**

`object`                    an object of type `CogapsResult`

**Value**

chi-sq error

**Examples**

```
data(GIST)
getMeanChiSq(GIST.result)
```

---

`getOriginalParameters`    *return original parameters used to generate this result*

---

**Description**

return original parameters used to generate this result

**Usage**

```
getOriginalParameters(object)

## S4 method for signature 'CogapsResult'
getOriginalParameters(object)
```

**Arguments**

`object`                    an object of type `CogapsResult`



**Value**

CogapsParams object

**Examples**

```
data(GIST)
params <- getOriginalParameters(GIST.result)
```

---

getParam	<i>get the value of a parameter</i>
----------	-------------------------------------

---

**Description**

get the value of a parameter

**Usage**

```
getParam(object, whichParam)

## S4 method for signature 'CogapsParams'
getParam(object, whichParam)
```

**Arguments**

- object            an object of type CogapsParams
- whichParam      a string with the name of the requested parameter

**Value**

the value of the parameter

**Examples**

```
params <- new("CogapsParams")
getParam(params, "seed")
```

---

getPatternMatrix      *return pattern matrix from CogapsResult object*

---

**Description**

return pattern matrix from CogapsResult object

**Usage**

```
getPatternMatrix(object)

## S4 method for signature 'CogapsResult'
getPatternMatrix(object)
```

**Arguments**

object                  an object of type CogapsResult

**Value**

pattern matrix

**Examples**

```
data(GIST)
patternMatrix <- getPatternMatrix(GIST.result)
```

---

getRetinaSubset      *get specified number of retina subsets*

---

**Description**

combines retina subsets from extdata directory

**Usage**

```
getRetinaSubset(n = 1)
```

**Arguments**

n                        number of subsets to use

**Value**

matrix of RNA counts

**Examples**

```
retSubset <- getRetinaSubset()  
dim(retSubset)
```

---

getSampleFactors      *return sampleFactors matrix from CogapsResult object*

---

**Description**

return sampleFactors matrix from CogapsResult object

**Usage**

```
getSampleFactors(object)  
  
## S4 method for signature 'CogapsResult'  
getSampleFactors(object)
```

**Arguments**

object              an object of type CogapsResult

**Value**

sampleFactors matrix

**Examples**

```
data(GIST)  
sFactors <- getSampleFactors(GIST.result)
```

---

getSubsets              *return the names of the genes (samples) in each subset*

---

**Description**

return the names of the genes (samples) in each subset

**Usage**

```
getSubsets(object)  
  
## S4 method for signature 'CogapsResult'  
getSubsets(object)
```

**Arguments**

object            an object of type CogapsResult

**Value**

CogapsParams object

**Examples**

```
data(GIST)
subsets <- getSubsets(GIST.result)
```

---

getUnmatchedPatterns    *return unmatched patterns from each subset*

---

**Description**

return unmatched patterns from each subset

**Usage**

```
getUnmatchedPatterns(object)

## S4 method for signature 'CogapsResult'
getUnmatchedPatterns(object)
```

**Arguments**

object            an object of type CogapsResult

**Value**

CogapsParams object

**Examples**

```
data(GIST)
unmatchedPatterns <- getUnmatchedPatterns(GIST.result)
```

---

getVersion	<i>return version number used to generate this result</i>
------------	---

---

**Description**

return version number used to generate this result

**Usage**

```
getVersion(object)
```

```
## S4 method for signature 'CogapsResult'  
getVersion(object)
```

**Arguments**

object            an object of type CogapsResult

**Value**

version number

**Examples**

```
data(GIST)  
getVersion(GIST.result)
```

---

GIST.data_frame	<i>GIST gene expression data from Ochs et al. (2009)</i>
-----------------	--

---

**Description**

GIST gene expression data from Ochs et al. (2009)

---

GIST.matrix	<i>GIST gene expression data from Ochs et al. (2009)</i>
-------------	--

---

**Description**

GIST gene expression data from Ochs et al. (2009)

---

GIST.result	<i>CoGAPS result from running on GIST dataset</i>
-------------	---

---

**Description**

CoGAPS result from running on GIST dataset

---

GIST.uncertainty	<i>GIST gene expression uncertainty matrix from Ochs et al. (2009)</i>
------------------	--

---

**Description**

GIST gene expression uncertainty matrix from Ochs et al. (2009)

---

GWCoGAPS	<i>Genome Wide CoGAPS</i>
----------	---------------------------

---

**Description**

wrapper around genome-wide distributed algorithm for CoGAPS

**Usage**

```
GWCoGAPS(
  data,
  params = new("CogapsParams"),
  nThreads = 1,
  messages = TRUE,
  outputFrequency = 500,
  uncertainty = NULL,
  checkpointOutFile = "gaps_checkpoint.out",
  checkpointInterval = 1000,
  checkpointInFile = NULL,
  transposeData = FALSE,
  BPPARAM = NULL,
  workerID = 1,
  asynchronousUpdates = FALSE,
  ...
)
```

**Arguments**

data	File name or R object (see details for supported types)
params	CogapsParams object
nThreads	maximum number of threads to run on
messages	T/F for displaying output
outputFrequency	number of iterations between each output (set to 0 to disable status updates, other output is controlled by @code messages)
uncertainty	uncertainty matrix - either a matrix or a supported file type
checkpointOutFile	name of the checkpoint file to create
checkpointInterval	number of iterations between each checkpoint (set to 0 to disable checkpoints)
checkpointInFile	if this is provided, CoGAPS runs from the checkpoint contained in this file
transposeData	T/F for transposing data while reading it in - useful for data that is stored as samples x genes since CoGAPS requires data to be genes x samples
BPPARAM	BiocParallel backend
workerID	if calling CoGAPS in parallel the worker ID can be specified, only worker 1 prints output and each worker outputs when it finishes, this is not necessary when using the default parallel methods (i.e. distributed CoGAPS) but only when the user is manually calling CoGAPS in parallel
asynchronousUpdates	enable asynchronous updating which allows for multi-threaded runs
...	allows for overwriting parameters in params

**Value**

CogapsResult object

**Examples**

```
## Not run:
data(GIST)
params <- new("CogapsParams")
params <- setDistributedParams(params, nSets=2)
params <- setParam(params, "nIterations", 100)
params <- setParam(params, "nPatterns", 3)
result <- GWCoGAPS(GIST.matrix, params, BPPARAM=BiocParallel::SerialParam())

## End(Not run)
```

---

```
initialize,CogapsParams-method
      constructor for CogapsParams
```

---

**Description**

constructor for CogapsParams

**Usage**

```
## S4 method for signature 'CogapsParams'
initialize(.Object, distributed = NULL, ...)
```

**Arguments**

.Object	CogapsParams object
distributed	either "genome-wide" or "single-cell" indicating which distributed algorithm should be used
...	initial values for slots

**Value**

initialized CogapsParams object

---

```
initialize,CogapsResult-method
      Constructor for CogapsResult
```

---

**Description**

Constructor for CogapsResult

**Usage**

```
## S4 method for signature 'CogapsResult'
initialize(
  .Object,
  Amean,
  Pmean,
  Asd,
  Psd,
  meanChiSq,
  geneNames,
  sampleNames,
  diagnostics = NULL,
  ...
)
```



**Arguments**

.Object	CogapsResult object
Amean	mean of sampled A matrices
Pmean	mean of sampled P matrices
Asd	std dev of sampled A matrices
Psd	std dev of sampled P matrices
meanChiSq	mean value of ChiSq statistic
geneNames	names of genes in data
sampleNames	names of samples in data
diagnostics	assorted diagnostic reports from the run
...	initial values for slots

**Value**

initialized CogapsResult object

---

patternMarkers	<i>compute pattern markers statistic</i>
----------------	--

---

**Description**

calculate the most associated pattern for each gene

**Usage**

```
patternMarkers(object, threshold = "all", lp = NA, axis = 1)
```

```
## S4 method for signature 'CogapsResult'
patternMarkers(object, threshold = "all", lp = NA, axis = 1)
```

**Arguments**

object	an object of type CogapsResult
threshold	the type of threshold to be used. The default "all" will distribute genes into pattern with the lowest ranking. The "cut" thresholds by the first gene to have a lower ranking, i.e. better fit to, a pattern.
lp	a vector of weights for each pattern to be used for finding markers. If NA markers for each pattern of the A matrix will be used.
axis	either 1 or 2, specifying if pattern markers should be calculated using the rows of the data (1) or the columns of the data (2)

**Value**

By default a non-overlapping list of genes associated with each lp.

**Examples**

```
data(GIST)
pm <- patternMarkers(GIST.result)
```

---

```
plotPatternMarkers      heatmap of original data clustered by pattern markers statistic
```

---

**Description**

heatmap of original data clustered by pattern markers statistic

**Usage**

```
plotPatternMarkers(
  object,
  data,
  patternPalette,
  sampleNames,
  samplePalette = NULL,
  heatmapCol = bluered,
  colDenogram = TRUE,
  scale = "row",
  ...
)
```

**Arguments**

object	an object of type CogapsResult
data	the original data as a matrix
patternPalette	a vector indicating what color should be used for each pattern
sampleNames	names of the samples to use for labeling
samplePalette	a vector indicating what color should be used for each sample
heatmapCol	pallelet giving color scheme for heatmap
colDenogram	logical indicating whether to display sample denogram
scale	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default is "row".
...	additional graphical parameters to be passed to heatmap. 2

**Value**

heatmap of the data values for the patternMarkers

**See Also**

[heatmap.2](#)

---

plotResiduals	<i>plot of residuals</i>
---------------	--------------------------

---

**Description**

calculate residuals and produce heatmap

**Usage**

```
plotResiduals(object, data, uncertainty = NULL)

## S4 method for signature 'CogapsResult'
plotResiduals(object, data, uncertainty = NULL)
```

**Arguments**

object	an object of type CogapsResult
data	original data matrix run through GAPS
uncertainty	original standard deviation matrix run through GAPS

**Value**

creates a residual plot

**Examples**

```
data(GIST)
# to expensive to call since it plots
# plotResiduals(GIST.result, GIST.matrix)
```

---

reconstructGene	<i>reconstruct gene</i>
-----------------	-------------------------

---

**Description**

reconstruct gene

**Usage**

```
reconstructGene(object, genes = NULL)

## S4 method for signature 'CogapsResult'
reconstructGene(object, genes = NULL)
```

**Arguments**

object            an object of type CogapsResult  
 genes            an index of the gene or genes of interest

**Value**

the D' estimate of a gene or set of genes

**Examples**

```
data(GIST)
estimatedD <- reconstructGene(GIST.result)
```

---

 scCoGAPS

*Single Cell CoGAPS*


---

**Description**

wrapper around single-cell distributed algorithm for CoGAPS

**Usage**

```
scCoGAPS(
  data,
  params = new("CogapsParams"),
  nThreads = 1,
  messages = TRUE,
  outputFrequency = 500,
  uncertainty = NULL,
  checkpointOutFile = "gaps_checkpoint.out",
  checkpointInterval = 1000,
  checkpointInFile = NULL,
  transposeData = FALSE,
  BPPARAM = NULL,
  workerID = 1,
  asynchronousUpdates = FALSE,
  ...
)
```

**Arguments**

data            File name or R object (see details for supported types)  
 params        CogapsParams object  
 nThreads      maximum number of threads to run on  
 messages      T/F for displaying output

outputFrequency	number of iterations between each output (set to 0 to disable status updates, other output is controlled by @code messages)
uncertainty	uncertainty matrix - either a matrix or a supported file type
checkpointOutFile	name of the checkpoint file to create
checkpointInterval	number of iterations between each checkpoint (set to 0 to disable checkpoints)
checkpointInFile	if this is provided, CoGAPS runs from the checkpoint contained in this file
transposeData	T/F for transposing data while reading it in - useful for data that is stored as samples x genes since CoGAPS requires data to be genes x samples
BPPARAM	BiocParallel backend
workerID	if calling CoGAPS in parallel the worker ID can be specified, only worker 1 prints output and each worker outputs when it finishes, this is not necessary when using the default parallel methods (i.e. distributed CoGAPS) but only when the user is manually calling CoGAPS in parallel
asynchronousUpdates	enable asynchronous updating which allows for multi-threaded runs
...	allows for overwriting parameters in params

**Value**

CogapsResult object

**Examples**

```
## Not run:
data(GIST)
params <- new("CogapsParams")
params <- setDistributedParams(params, nSets=2)
params <- setParam(params, "nIterations", 100)
params <- setParam(params, "nPatterns", 3)
result <- scCoGAPS(t(GIST.matrix), params, BPPARAM=BiocParallel::SerialParam())

## End(Not run)
```

---

setAnnotationWeights *set the annotation labels and weights for subsetting the data*

---

**Description**

these parameters are interrelated so they must be set together

**Usage**

```
setAnnotationWeights(object, annotation, weights)
```

```
## S4 method for signature 'CogapsParams'
setAnnotationWeights(object, annotation, weights)
```

**Arguments**

object	an object of type CogapsParams
annotation	vector of labels
weights	vector of weights

**Value**

the modified params object

**Examples**

```
params <- new("CogapsParams")
params <- setAnnotationWeights(params, c('a', 'b', 'c'), c(1,2,1))
```

---

setDistributedParams    *set the value of parameters for distributed CoGAPS*

---

**Description**

these parameters are interrelated so they must be set together

**Usage**

```
setDistributedParams(
  object,
  nSets = NULL,
  cut = NULL,
  minNS = NULL,
  maxNS = NULL
)
```

```
## S4 method for signature 'CogapsParams'
setDistributedParams(
  object,
  nSets = NULL,
  cut = NULL,
  minNS = NULL,
  maxNS = NULL
)
```

**Arguments**

object	an object of type CogapsParams
nSets	number of sets to break data into
cut	number of branches at which to cut dendrogram used in pattern matching
minNS	minimum of individual set contributions a cluster must contain
maxNS	maximum of individual set contributions a cluster can contain

**Value**

the modified params object

**Examples**

```
params <- new("CogapsParams")
params <- setDistributedParams(params, 5)
```

---

setFixedPatterns      *set the fixed patterns for either the A or the P matrix*

---

**Description**

these parameters are interrelated so they must be set together

**Usage**

```
setFixedPatterns(object, fixedPatterns, whichMatrixFixed)
```

```
## S4 method for signature 'CogapsParams'
setFixedPatterns(object, fixedPatterns, whichMatrixFixed)
```

**Arguments**

object	an object of type CogapsParams
fixedPatterns	values for either the A or P matrix
whichMatrixFixed	either 'A' or 'P' indicating which matrix is fixed

**Value**

the modified params object

**Examples**

```
params <- new("CogapsParams")
data(GIST)
params <- setFixedPatterns(params, getSampleFactors(GIST.result), 'P')
```

---

setParam	<i>set the value of a parameter</i>
----------	-------------------------------------

---

**Description**

set the value of a parameter

**Usage**

```
setParam(object, whichParam, value)
```

```
## S4 method for signature 'CogapsParams'  
setParam(object, whichParam, value)
```

**Arguments**

object	an object of type CogapsParams
whichParam	a string with the name of the parameter to be changed
value	the value to set the parameter to

**Value**

the modified params object

**Examples**

```
params <- new("CogapsParams")  
params <- setParam(params, "seed", 123)
```



# Index

binaryA, [3](#)  
binaryA, CogapsResult-method (binaryA), [3](#)  
buildReport, [4](#)

calcCoGAPSSStat, [5](#)  
calcCoGAPSSStat, CogapsResult-method  
(calcCoGAPSSStat), [5](#)  
calcGeneGSStat, [6](#)  
calcGeneGSStat, CogapsResult-method  
(calcGeneGSStat), [6](#)  
calcZ, [7](#)  
calcZ, CogapsResult-method (calcZ), [7](#)  
checkpointsEnabled, [7](#)  
CoGAPS, [8](#)  
CoGAPS-package, [3](#)  
CogapsParams, [9](#)  
CogapsParams-class, [10](#)  
CogapsResult-class, [11](#)  
compiledWithOpenMPsupport, [11](#)  
computeGeneGSProb, [12](#)  
computeGeneGSProb, CogapsResult-method  
(computeGeneGSProb), [12](#)

findConsensusMatrix, [13](#)

getAmplitudeMatrix, [13](#)  
getAmplitudeMatrix, CogapsResult-method  
(getAmplitudeMatrix), [13](#)  
getClusteredPatterns, [14](#)  
getClusteredPatterns, CogapsResult-method  
(getClusteredPatterns), [14](#)  
getCorrelationToMeanPattern, [14](#)  
getCorrelationToMeanPattern, CogapsResult-method  
(getCorrelationToMeanPattern),  
[14](#)  
getFeatureLoadings, [15](#)  
getFeatureLoadings, CogapsResult-method  
(getFeatureLoadings), [15](#)  
getMeanChiSq, [16](#)  
getMeanChiSq, CogapsResult-method  
(getMeanChiSq), [16](#)  
getOriginalParameters, [16](#)  
getOriginalParameters, CogapsResult-method  
(getOriginalParameters), [16](#)  
getParam, [17](#)  
getParam, CogapsParams-method  
(getParam), [17](#)  
getPatternMatrix, [18](#)  
getPatternMatrix, CogapsResult-method  
(getPatternMatrix), [18](#)  
getRetinaSubset, [18](#)  
getSampleFactors, [19](#)  
getSampleFactors, CogapsResult-method  
(getSampleFactors), [19](#)  
getSubsets, [19](#)  
getSubsets, CogapsResult-method  
(getSubsets), [19](#)  
getUnmatchedPatterns, [20](#)  
getUnmatchedPatterns, CogapsResult-method  
(getUnmatchedPatterns), [20](#)  
getVersion, [21](#)  
getVersion, CogapsResult-method  
(getVersion), [21](#)  
GIST.data\_frame, [21](#)  
GIST.matrix, [21](#)  
GIST.result, [22](#)  
GIST.uncertainty, [22](#)  
GWCoGAPS, [22](#)

heatmap.2, [26](#)

initialize, CogapsParams-method, [24](#)  
initialize, CogapsResult-method, [24](#)

patternMarkers, [25](#)  
patternMarkers, CogapsResult-method  
(patternMarkers), [25](#)  
plotPatternMarkers, [26](#)  
plotResiduals, [27](#)

plotResiduals, CogapsResult-method  
(plotResiduals), [27](#)

reconstructGene, [27](#)  
reconstructGene, CogapsResult-method  
(reconstructGene), [27](#)

scCoGAPS, [28](#)  
setAnnotationWeights, [29](#)  
setAnnotationWeights, CogapsParams-method  
(setAnnotationWeights), [29](#)  
setDistributedParams, [30](#)  
setDistributedParams, CogapsParams-method  
(setDistributedParams), [30](#)  
setFixedPatterns, [31](#)  
setFixedPatterns, CogapsParams-method  
(setFixedPatterns), [31](#)  
setParam, [32](#)  
setParam, CogapsParams-method  
(setParam), [32](#)