Package ‘CoGAPS’

January 30, 2017

Version 2.8.0
Date 2014-08-23
Title Coordinated Gene Activity in Pattern Sets
Author Elana J. Fertig, Michael F. Ochs

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>

Depends R (>= 3.0.1), Rcpp (>= 0.11.2)
Imports RColorBrewer (>= 1.0.5), gplots (>= 2.8.0), graphics,
grDevices, methods, stats, utils
Suggests testthat, lintr
LinkingTo Rcpp, BH
License GPL (==2)

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

NeedsCompilation yes
RoxygenNote 5.0.1

R topics documented:

   CoGAPS-package .........................................................  2
   binaryA .................................................................  2
   calcCoGAPSSStat ......................................................  3
   calcGeneGSStat .........................................................  3
   calcZ .................................................................  4
   CoGAPS .................................................................  4
   computeGeneGSProb ....................................................  5
   gapsMapRun .............................................................  7
   gapsMapTestRun .......................................................  8
   gapsRun ............................................................... 10
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: 1.0
Date: 2014-07-23
License: LGPL

Author(s)

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

References


binaryA

binaryA 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
**Description**

`binaryA` 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(Amean, Asd, threshold = 3)`

**Arguments**

- `Amean`: the mean estimate for the A matrix
- `Asd`: the standard deviations on Amean
- `threshold`: the number of standard deviations above zero that an element of Amean must be to get a value of 1

**Description**

`calcCoGAPSStat` calculates the gene set statistics for each column of A using a Z-score from the elements of the A matrix, the input gene set, and permutation tests.

**Usage**

`calcCoGAPSStat(Amean, Asd, GStoGenes, numPerm = 500)`

**Arguments**

- `Amean`: A matrix mean values
- `Asd`: A matrix standard deviations
- `GStoGenes`: data.frame or list with gene sets
- `numPerm`: number of permutations for null

**Description**

`calcGeneGSStat` 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(Amean, Asd, GSGenes, numPerm, Pw = rep(1, ncol(Amean)), nullGenes = F)`
Arguments

Amean  A matrix mean values
Asd    A matrix standard deviations
GSGenes  data.frame or list with gene sets
numPerm  number of permutations for null
Pw      weight on genes
nullGenes - logical indicating gene adjustment

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

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

Usage
calcZ(meanMat, sdMat)

Arguments

meanMat  matrix of mean values
sdMat    matrix of standard deviation values

CoGAPS  CoGAPS calls the C++ MCMC code through gapsRun and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix and then calls calcCoGAPSStat to estimate gene set activity with nPerm set to 500

Description
CoGAPS calls the C++ MCMC code through gapsRun and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix and then calls calcCoGAPSStat to estimate gene set activity with nPerm set to 500

Usage
CoGAPS(data, unc, ABins = data.frame(), PBins = data.frame(), GStoGenes, nFactor = 7, simulation_id = "simulation", nEquil = 1000, nSample = 1000, nOutR = 1000, output_atomic = FALSE, fixedBinProbs = FALSE, fixedDomain = "N", sampleSnapshots = TRUE, numSnapshots = 100, plot = TRUE, nPerm = 500, alphaA = 0.01, nMaxA = 1e+05, max_gibbmass_paraA = 100, alphaP = 0.01, nMaxP = 1e+05, max_gibbmass_paraP = 100)
**computeGeneGSProb**

**Arguments**

- **data**: data matrix
- **unc**: uncertainty matrix (std devs for chi-squared of Log Likelihood)
- **ABins**: a matrix of same size as A which gives relative probability of that element being non-zero
- **PBins**: a matrix of same size as P which gives relative probability of that element being non-zero
- **GStoGenes**: data.frame or list with gene sets
- **nFactor**: number of patterns (basis vectors, metagenes)
- **simulation_id**: name to attach to atoms files if created
- **nEquil**: number of iterations for burn-in
- **nSample**: number of iterations for sampling
- **nOutR**: how often to print status into R by iterations
- **output_atomic**: whether to write atom files (large)
- **fixedBinProbs**: Boolean for using relative probabilities given in Abins and Pbins
- **fixedDomain**: character to indicate whether A or P is domain for relative probabilities
- **sampleSnapshots**: Boolean to indicate whether to capture individual samples from Markov chain during sampling
- **numSnapshots**: the number of individual samples to capture
- **plot**: Boolean to indicate whether to produce output graphics
- **nPerm**: number of permutations in gene set test
- **alphaA**: sparsity parameter for A domain
- **nMaxA**: PRESENTLY UNUSED, future = limit number of atoms
- **max_gibbmass_paraA**: limit truncated normal to max size
- **alphaP**: sparsity parameter for P domain
- **nMaxP**: PRESENTLY UNUSED, future = limit number of atoms
- **max_gibbmass_paraP**: limit truncated normal to max size

---

**computeGeneGSProb**

**CoGAPS gene membership statistic**

**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. Specifically, we compute the following summary statistic for each gene $g$ that is a candidate member of gene set $G$:

$$ S_{g,G} = \frac{\sum_p -\log(P_{rG,p}P_{w[p]}(A_{gp}/\sigma_{gp}))}{\sum_p -\log(P_{rG,p}P_{w[p]}),} $$
where $p$ indexes each of the patterns, $Pr_{G,p}$ is the probability that gene set $G$ is upregulated computed with `calcCoGAPSStat`, $A_{gp}$ is the mean amplitude matrix from the GAPS matrix factorization, $Pw[p]$ is a prior weighting for each pattern based upon the context to which that pattern relates, and $\sigma_{gp}$ is the standard deviation of the amplitude matrix. P-values are formulated from a permutation test comparing the value of $S_{p,G}$ for genes in GSGenes relative to the value of $S_{p,G}$ numPerm random gene sets with the same number of targets.

### Usage

```r
computeGeneGSProb(Amean, Asd, GSGenes, Pw=rep(1,ncol(Amean)),numPerm=500,PwNull=F)
```

### Arguments

- **Amean**: Sampled mean value of the amplitude matrix $A$. `rownames(Amean)` must correspond to the gene names contained in GSGenes.
- **Asd**: Sampled standard deviation of the amplitude matrix $A$.
- **GSGenes**: Vector containing the prior estimate of members of the gene set of interest.
- **Pw**: Vector containing the weight to assign each pattern in the gene statistic assumed to be computed from the association of the pattern with samples in a given context (optional: default=1 giving all patterns equal weight).
- **numPerm**: Number of permutations used for the null distribution in the gene set statistic. (optional; default=500)
- **PwNull**: Logical value. If TRUE, use pattern weighting in Pw when computing the null distribution for the statistic. If FALSE, do not use the pattern weighting so that the null is context independent. (optional; default=F)

### Value

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

### Author(s)

Elana J. Fertig <ejfertig@jhmi.edu>

### References


### See Also

- `calcCoGAPSStat`

### Examples

```r
## Not run:
#################################################
# Results for GIST data in Fertig et al. (2012)#
#################################################
# load the data
gapsMapRun

data('GIST_TS_20084')
data('TFGSList')

# define transcription factors of interest based on Ochs et al. (2009)

# run the GAPS matrix factorization
nIter <- 10000
results <- CoGAPS(GIST.D, GIST.S, tf2ugFC,
nFactor=5, nEquil=nIter, nSample=nIter, plot=FALSE)

# set membership statistics
permTFStats <- list()
for (tf in TFs) {
  genes <- levels(tf2ugFC[,tf])
  genes <- genes[2:length(genes)]
  permTFStats[[tf]] <- computeGeneTFProb(Amean = GISTResults$Amean, Asd = GISTResults$Asd, genes)
}

## End(Not run)

gapsMapRun

gapsMapRun calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix; as opposed to gapsRun, this method takes an additional input specifying set patterns in the P matrix

Description
gapsMapRun calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix; as opposed to gapsRun, this method takes an additional input specifying set patterns in the P matrix

Usage
gapsMapRun(D, S, FP, ABins = data.frame(), PBins = data.frame(),
nFactor = 5, simulation_id = "simulation", nEquil = 1000,
nSample = 1000, nOutR = 1000, output_atoatomic = FALSE,
fixedMatrix = "P", fixedBinProbs = FALSE, fixedDomain = "N",
sampleSnapshots = TRUE, numSnapshots = 100, alphaA = 0.01,
nMaxA = 1e+05, max_gibbmass_paraA = 100, alphaP = 0.01, nMaxP = 1e+05,
max_gibbmass_paraP = 100, seed = -1)

Arguments
D data matrix
S uncertainty matrix (std devs for chi-squared of Log Likelihood)
FP data.frame with rows giving fixed patterns for P
gapsMapTestRun

gapsMapTestRun calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix; as opposed to gapsRun, this method takes an additional input specifying set patterns in the P matrix

description

gapsMapTestRun calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix; as opposed to gapsRun, this method takes an additional input specifying set patterns in the P matrix.
Usage
gapsMapTestRun(D, S, FP, ABins = data.frame(), PBins = data.frame(),
nFactor = 7, simulation_id = "simulation", nEquil = 1000,
nSample = 1000, nOutR = 1000, output_atomic = FALSE,
fixedMatrix = "P", fixedBinProbs = FALSE, fixedDomain = "N",
alphaA = 0.01, nMaxA = 1e+05, max_gibbmass_paraA = 100, alphaP = 0.01,
nMaxP = 1e+05, max_gibbmass_paraP = 100)

Arguments
D data matrix
S uncertainty matrix (std devs for chi-squared of Log Likelihood)
FP data.frame with rows giving fixed patterns for P
ABins a matrix of same size as A which gives relative probability of that element being non-zero
PBins a matrix of same size as P which gives relative probability of that element being non-zero
nFactor number of patterns (basis vectors, metagenes), which must be greater than or equal to the number of rows of FP
simulation_id name to attach to atoms files if created
nEquil number of iterations for burn-in
nSample number of iterations for sampling
nOutR how often to print status into R by iterations
output_atomic whether to write atom files (large)
fixedMatrix character indicating whether A or P matrix has fixed columns or rows respectively
fixedBinProbs Boolean for using relative probabilities given in Abins and Pbins
fixedDomain character to indicate whether A or P is domain for relative probabilities
alphaA sparsity parameter for A domain
nMaxA PRESENTLY UNUSED, future = limit number of atoms
max_gibbmass_paraA limit truncated normal to max size
alphaP sparsity parameter for P domain
nMaxP PRESENTLY UNUSED, future = limit number of atoms
max_gibbmass_paraP limit truncated normal to max size
gapsRun calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix.

**Description**

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

**Usage**

```r
gapsRun(D, S, ABins = data.frame(), PBins = data.frame(), nFactor = 7,
    simulation_id = "simulation", nEquil = 1000, nSample = 1000,
    nOutR = 1000, output_atomic = FALSE, fixedBinProbs = FALSE,
    fixedDomain = "N", sampleSnapshots = TRUE, numSnapshots = 100,
    alphaA = 0.01, nMaxA = 1e+05, max_gibbmass_paraA = 100, alphaP = 0.01,
    nMaxP = 1e+05, max_gibbmass_paraP = 100, seed = -1)
```

**Arguments**

- `D`: data matrix
- `S`: uncertainty matrix (std devs for chi-squared of Log Likelihood)
- `ABins`: a matrix of same size as A which gives relative probability of that element being non-zero
- `PBins`: a matrix of same size as P which gives relative probability of that element being non-zero
- `nFactor`: number of patterns (basis vectors, metagenes), which must be greater than or equal to the number of rows of FP
- `simulation_id`: name to attach to atoms files if created
- `nEquil`: number of iterations for burn-in
- `nSample`: number of iterations for sampling
- `nOutR`: how often to print status into R by iterations
- `output_atomic`: whether to write atom files (large)
- `fixedBinProbs`: Boolean for using relative probabilities given in Abins and Pbins
- `fixedDomain`: character to indicate whether A or P is domain for relative probabilities
- `sampleSnapshots`: Boolean to indicate whether to capture individual samples from Markov chain during sampling
- `numSnapshots`: the number of individual samples to capture
- `alphaA`: sparsity parameter for A domain
- `nMaxA`: PRESENTLY UNUSED, future = limit number of atoms
- `max_gibbmass_paraA`: limit truncated normal to max size
- `alphaP`: sparsity parameter for P domain
- `nMaxP`: PRESENTLY UNUSED, future = limit number of atoms
gapsTestRun calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix.

**Description**

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

**Usage**

```r
gapsTestRun(D, S, ABins = data.frame(), PBins = data.frame(), nFactor = 7,
  simulation_id = "simulation", nEquil = 1000, nSample = 1000,
  nOutR = 1000, output_atomic = FALSE, fixedBinProbs = FALSE,
  fixedDomain = "N", alphaA = 0.01, nMaxA = 1e+05,
  max_gibbmass_paraA = 100, alphaP = 0.01, nMaxP = 1e+05,
  max_gibbmass_paraP = 100)
```

**Arguments**

- **D**: data matrix
- **S**: uncertainty matrix (std devs for chi-squared of Log Likelihood)
- **ABins**: a matrix of same size as A which gives relative probability of that element being non-zero
- **PBins**: a matrix of same size as P which gives relative probability of that element being non-zero
- **nFactor**: number of patterns (basis vectors, metagenes), which must be greater than or equal to the number of rows of FP
- **simulation_id**: name to attach to atoms files if created
- **nEquil**: number of iterations for burn-in
- **nSample**: number of iterations for sampling
- **nOutR**: how often to print status into R by iterations
- **output_atomic**: whether to write atom files (large)
- **fixedBinProbs**: Boolean for using relative probabilities given in Abins and Pbins
- **fixedDomain**: character to indicate whether A or P is domain for relative probabilities
- **alphaA**: sparsity parameter for A domain
- **nMaxA**: PRESENTLY UNUSED, future = limit number of atoms
- **max_gibbmass_paraA**: limit truncated normal to max size
- **alphaP**: sparsity parameter for P domain
- **nMaxP**: PRESENTLY UNUSED, future = limit number of atoms
- **max_gibbmass_paraP**: limit truncated normal to max size

```r
gapsTestRun calls the C++ MCMC code and performs Bayesian matrix factorization returning the two matrices that reconstruct the data matrix.
```
**GIST.D**

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

**Description**

Gene expression data from gastrointestinal stromal tumor cell lines treated with Gleevec.

**Usage**

GIST_TS_20084

**Format**

Matrix with 1363 genes by 9 samples of mean gene expression data.

**References**


---

**GIST.S**

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

**Description**

Standard deviation of gene expression data from gastrointestinal stromal tumor cell lines treated with Gleevec.

**Usage**

GIST_TS_20084

**Format**

Matrix with 1363 genes by 9 samples containing standard deviation (GIST.S) of the gene expression data.

**References**

GSets

Simulated dataset to quantify gene set membership.

Description
Simulated gene sets used to generate amplitude matrix in SimpSim.A and corresponding data SimpSim.D.

Usage
GSets

Format
A list containing names of genes in two simulated gene sets used to generate the data in SimpSim.D.

plotAtoms

plotAtoms a simple plot of the number of atoms from one of the vectors returned with atom numbers

Description
plotAtoms a simple plot of the number of atoms from one of the vectors returned with atom numbers

Usage
plotAtoms(gapsRes, type = "sampA")

Arguments
gapsRes the list resulting from applying GAPS
type the atoms to plot, values are "sampA", "sampP", "equilA", or "equilP" to plot sampling or equilibration teop atom numbers

plotDiag

plotDiag plots a series of diagnostic plots

Description
plotDiag plots a series of diagnostic plots

Usage
plotDiag(gapsRes)

Arguments
gapsRes list returned by gapsRun, gapsMapRun, or CoGAPS
plotGAPS plots the output A and P matrices as a heatmap and line plot respectively

Description
plotGAPS plots the output A and P matrices as a heatmap and line plot respectively

Usage
plotGAPS(A, P, outputPDF = "")

Arguments
A the mean A matrix
P the mean P matrix
outputPDF optional root name for PDF output, if not specified, output goes to screen

plotP plots the P matrix in a line plot with error bars

Description
plotP plots the P matrix in a line plot with error bars

Usage
plotP(PMean_Mat, P_SD)

Arguments
PMean_Mat matrix of mean values of P
P_SD matrix of standard deviation values of P

plotSmoothPatterns plots the output A and P matrices as a heatmap and line plot respectively

Description
plotSmoothPatterns plots the output A and P matrices as a heatmap and line plot respectively

Usage
plotSmoothPatterns(P, x = NULL, breaks = NULL, breakStyle = T, orderP = !all(is.null(x)), plotPTS = F, pointCol = "black", lineCol = "grey", add = F, ...)
### Arguments

- **P**: the mean A matrix
- **x**: optional variables
- **breaks**: breaks in plots
- **breakStyle**: style of breaks
- **orderP**: whether to order patterns
- **plotPTS**: whether to plot points on lines
- **pointCol**: color of points
- **lineCol**: color of line
- **add**: logical specifying if bars should be added to an already existing plot; defaults to ‘FALSE’.

... arguments to be passed to/from other methods. For the default method these can include further arguments (such as ‘axes’, ‘asp’ and ‘main’) and graphical parameters (see ‘par’) which are passed to ‘plot.window()’, ‘title()’ and ‘axis’.

---

**reorderByPatternMatch** plots the output A and P matrices as a heatmap and line plot respectively.

### Description

*reorderByPatternMatch* plots the output A and P matrices as a heatmap and line plot respectively.

### Usage

```r
reorderByPatternMatch(P, matchTo)
```

### Arguments

- **P**: matrix to be matched
- **matchTo**: matrix to match P to

---

**residuals** calculate residuals and produce heatmap

### Description

*residuals* calculate residuals and produce heatmap.

### Usage

```r
residuals(AMean_Mat, PMean_Mat, D, S)
```

### Arguments

- **AMean_Mat**: matrix of mean values for A from GAPS
- **PMean_Mat**: matrix of mean values for P from GAPS
- **D**: original data matrix run through GAPS
- **S**: original standard deviation matrix run through GAPS
### SimpSim.A

**Description**
True amplitude matrix generated from gene sets in GSets used to generate simulated data in SimpSim.D.

**Usage**
SimpSim.A

**Format**
Matrix with 30 genes by 3 patterns of true amplitude used to generate simulated data.

### SimpSim.D

**Description**
Simulated gene expression data from true patterns in SimpSim.P and amplitude in SimpSim.A.

**Usage**
SimpSim.D

**Format**
Matrix with 30 genes by 20 samples of simulated gene expression data.

### SimpSim.P

**Description**
True pattern matrix used to generate simulated data in SimpSim.D.

**Usage**
SimpSim.P

**Format**
Matrix with 3 patterns by 20 samples of true patterns used to generate simulated data.
### SimpSim.S

**Description**

Standard deviation of simulated gene expression data from true patterns in SimpSim.P and amplitude in SimpSim.A.

**Usage**

SimpSim.S

**Format**

Matrix with 30 genes by 20 samples of containing standard deviation of simulated gene expression data.

### tf2ugFC

**Description**

List of genes contained in gastrointestinal stromal tumor cell line measurements that are regulated by transcription factors in the TRANSFAC database. Used for the gene set analysis in Ochs et al. (2009).

**Usage**

TFGSList

**Format**

Data.frame containing genes (rows) regulated by each transcription factor (columns).

### References

Index

*Topic **datasets**
- GIST.D, 12
- GIST.S, 12
- GSets, 13
- SimpSim.A, 16
- SimpSim.D, 16
- SimpSim.P, 16
- SimpSim.S, 17
- tf2ugFC, 17

*Topic **misc**
- computeGeneGSProb, 5

binaryA, 2
- calcCoGAPSStat, 3, 6
- calcGeneGSStat, 3
- calcZ, 4
- CoGAPS, 4
- CoGAPS-package, 2
- computeGeneGSProb, 5
- gapsMapRun, 7
- gapsMapTestRun, 8
- gapsRun, 10
- gapsTestRun, 11
- geneGSProb (computeGeneGSProb), 5
- GIST.D, 12
- GIST.S, 12
- GSets, 13, 16
- list, 13
- plotAtoms, 13
- plotDiag, 13
- plotGAPS, 14
- plotP, 14
- plotSmoothPatterns, 14
- reorderByPatternMatch, 15
- residuals, 15
- SimpSim.A, 13, 16, 16, 17
- SimpSim.D, 13, 16, 16
- SimpSim.P, 16, 16, 17
- SimpSim.S, 17
- tf2ugFC, 17