Package 'hbm'

February 20, 2015

Type Package

Title Hierarchical Block Matrix Analysis

Version 1.0

Date 2015-01-25

Author Yoli Shavit

Maintainer Yoli Shavit <ys388@cam.ac.uk>

Description A package for building hierarchical block matrices from association matrices and for performing multi-scale analysis. It specifically targets chromatin contact maps, generated from high-throughput chromosome conformation capture data, such as 5C and Hi-C, and provides methods for detecting movements and for computing chain hierarchy and region communicability across scales.

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports Matrix, foreach, doParallel

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-01 19:50:29

R topics documented:

hbm-package 2
add.noise 2
communicability 3
detect.movement 5
generate.random.conf 7
get.movements 8
hbm 9
hbm.features 11
hierarchy 12
mcl 14

Index **16**

hbm-package *Hierarchical Block Matrix Analysis*

Description

A package for building hierarchical block matrices from association matrices and for performing multi-scale analysis. It specifically targets chromatin contact maps, generated from high-throughput chromosome conformation capture data, such as 5C and Hi-C, and provides methods for detecting movements and for computing chain hierarchy and region communicability across scales.

Details

Package:	hbm
Type:	Package
Version:	1.0
Date:	2015-01-25
License:	GPL (>=2)

Get started with hbm's tutorials at: http://www.cl.cam.ac.uk/~ys388/hbm/

Author(s)

Yoli Shavit
Maintainer: <ys388@cam.ac.uk>

References

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

add.noise *Add Noise to a Symmetric Association Matrix*

Description

add.noise adds noise to a symmetric association matrix, typically a chromatin contact map.

Usage

```
add.noise(m, ...)
```

Arguments

m	a symmetric numeric association matrix, typically a chromatin contact map.
...	additional parameters for jitter.

Value

add.noise returns a matrix of the same dimension as m but with noise added (see additional parameters for setting noise amount in jitter).

Author(s)

Yoli Shavit

References

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

hbm.features to see how add.noise is used to estimate feature robustness in hierarchical block matrices
hbm to learn how to build hierarchical block matrices
hbm's tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Examples

```
set.seed(2)
n = 200 # chain size
conf = generate.random.conf(n, sd = 0.5, scale = FALSE)
# generate a contact map -like matrix using the model c ~ exp(-d)
control = exp(-1*as.matrix(dist(conf)))
# add noise
control.noisy = add.noise(control, factor = 5)
```

---

communicability *Compute Communicability between Nodes in an HBM*

Description

communicability computes the scale communicability between nodes (rows) in a hierarchical block matrix hm, generated with hbm.

Usage

```
communicability(hm)
```

Arguments

hm	a hierarchical block matrix computed with hbm, with: $hm_{i,j}$ = the minimal scale (iteration) at which i and j were clustered together, or 0 if i=j

Details

The communicability of an adjacency matrix A can be expressed as e^A (Estrada et al., 2012), so that the i,j-th entry is a weighted sum of all paths from i to j, where shortest paths are assigned with larger weights. For a hierarchical block matrix, computed with hbm, each scale s defines its own adjacency matrix, where all entries with values larger than s are set to 0. The scale-communicability between 2 nodes i and j in this matrix is defined here as the mean communicability across scales (excluding the largest scale).

Value

communicability returns a matrix of the same dimensions as hm, where the (i,j)-th entry gives the scale-communicability between i and j in hm.

Author(s)

Yoli Shavit

References

Estrada, E., Hatano, N. and Benzi, M. The physics of communicability in complex networks. Physics Reports 514, 89-119 (2012).

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

hbm to learn how to build hierarchical block matrices
hbm's tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Examples

```
set.seed(2)
n = 100 # chain size
conf = generate.random.conf(n, scale = FALSE)

# compute the HBM
hm.control = hbm(exp(-1*as.matrix(dist(conf))), 2)$hm

# compute scale communicability
comm = communicability(hm.control)

# explore for position 50
plot(1:n, comm[50,], xlab = "Position", ylab = "Communicability",
main = "Communicability for Position 50", pch=16)

# plot in original configuration
cols = rep("black", n)
cols[which(comm[50,] > 100)] = "blue"
plot(conf, xlab = "X", ylab = "Y", type = "n")
text(conf, labels = 1:n, col = cols)
```

`detect.movement` *Detect Movements Between Association Matrices using their HBMs*

Description

`detect.movement` takes a reference and a target association matrices, typically contact maps of chromosomes, and their hierarchical block matrices (computed with `hbm`) and detects movements between the reference and the target. A detected movement represents a disposition of node in a the reference, that gave rise to its new position in the target.

Usage

```
detect.movement(ref, target, ref.res, target.res,
            motion.prop.thresh = 0.75, siglevel = 0.05, verbose = FALSE)
```

Arguments

`ref`	the reference matrix. A numeric association matrix, typically a symmetric chromatin contact map.
`target`	the target matrix to be compared to `ref`. A numeric association matrix, typically a symmetric chromatin contact map.
`ref.res`	the result of calling `hbm` with `ref`.
`target.res`	the result of calling `hbm` with `target`.
`motion.prop.thresh`	numeric giving the threshold for detecting whether a region has moved based on the proportions of changed neighbors in its cluster.
`siglevel`	numeric giving the threshold for detecting a significant compactness/unfolding in a given cluster, set to 0.05 by default.
`verbose`	boolean indicating whether to print intermediate results.

Details

`detect.movement` iterates through the scales of the reference matrix (using its HBM) and through the clusters at each scale and test for movements in the target. For each cluster it further tests for significant differences in association probabilities and for changes in the neighbors of each node.

Value

`detect.movement` returns a numeric matrix, with the i,j-th entry taking one of the following values:
-1 if i has moved away from j,
1 if i has moved towards j,
0 if there was no movement, and
-0.5 and 0.5 for implicated away and towards movements, respectively.

Author(s)

Yoli Shavit

References

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

`get.movements` to summarize the results of `detect.movement`
`hbm` to learn how to build hierarchical block matrices
hbm's tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Examples

```
set.seed(2)
n = 200 # chain size

# control configuration
conf = generate.random.conf(n, sd = 0.5, scale = FALSE)
# condition-1
conf.tr.1 = conf
conf.tr.1[99,] = conf.tr.1[77,]-0.5

# generate contact map-like matrices
control = exp(-1*as.matrix(dist(conf)))
tr.1 = exp(-1*as.matrix(dist(conf.tr.1)))

control.res = hbm(control, 2)
tr.1.res = hbm(tr.1, 2)
m1 = detect.movement(control, tr.1, control.res, tr.1.res)
resm = get.movements(m1, control.res$hm)
resm

# compare with configuration
par(mfrow = c(1,2))
cols = rep("black", n)
cols[unique(resm$from)] = "green"
plot(conf, xlab = "X", ylab = "Y", type = 'n', main = "Control")
text(conf[,1:2], labels = 1:n, cex = 0.75, col = cols)
cols = rep("black", n)
cols[unique(resm$from)] = "green"
cols[resm$to[which(resm$type == 0.5)]] = "pink"
cols[resm$to[which(resm$type == 1)]] = "red"
cols[resm$to[which(resm$type == -0.5)]] = "cyan"
cols[resm$to[which(resm$type == -1)]] = "blue"
plot(conf.tr.1, xlab = "X", ylab = "Y", type = 'n', main = "Condition-1", col = cols)
text(conf.tr.1[,1:2], labels = 1:n, cex = 0.75, col = cols)
```

`generate.random.conf` *Generate a Random Chain Configuration*

Description

`generate.random.conf` generates a random chain configuration following a random walk/giant loop model (Sachs et al., 1995).

Usage

```
generate.random.conf(n, k = 3, perturb = NULL, scale = T, mean = 0, sd = 1)
```

Arguments

`n`	integer giving the chain length (number of beads in the chain).
`k`	integer giving the space dimension, set to 3 by default.
`perturb`	integer vector of nodes (indices) to perturb. This argument can be used to generate a configuration that deviates from the chain constraints of successive beads (nodes). Perturbation is achieved by sampling a new order for the beads to perturb and exchanging their coordinates in the original configuration accordingly. By default `perturb` is set to NULL indicating no perturbation should not be applied.
`scale`	boolean indicating whether or not to scale the generated configuration, set to `TRUE` by default.
`mean`	numeric giving the mean of differences distribution along each axis, set to 0 by default.
`sd`	numeric giving the standard deviation of differences distribution along each axis, set to 1 by default.

Details

`generate.random.conf` aims to generate a chromosome-like chain of n beads (nodes), in a k-D Euclidean space (k=3 by default) that follows a random walk/giant loop model (Sachs et al., 1995). This is achieved by sampling the differences between successive beads' coordinates from a normal distribution $N(\mu, \sigma)$ ($\mu = 0$, $\sigma = 1$, by default), across each axis (see examples in Hu et al., 2013 and Shavit et al., 2014). The configuration is scaled by default so that the distance between the first and last beads is approximately one unit. `generate.random.conf` can also be used to generate configurations that deviate from the chain constraints by perturbing beads.

Value

`generate.random.conf` returns a n x k matrix, giving the coordinates of n beads (nodes) in a k-d space.

Author(s)

Yoli Shavit

References

Sachs, R. K., van den Engh, G., Trask, B., Yokota, H. and Hearst, J. E. A random-walk/giant-loop model for interphase chromosomes. Proceedings of the National Academy of Sciences of the United States of America, 92, 2710-4 (1995).

Hu, M. et al. Bayesian Inference of Spatial Organizations of Chromosomes. PLoS Computational Biology. 9, e1002893 (2013).

Shavit, Y., Hamey, F. K. and Lio, P. FisHiCal: an R package for iterative FISH-based calibration of Hi-C data. Bioinformatics, 30, 3120-3122 (2014).

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

hbm to learn how to build a hierarchical block matrix from a contact map of a random configuration. hbm's tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Examples

```
set.seed(2)
n = 100
conf = generate.random.conf(n, k = 2)
plot(conf, xlab = "x", ylab = "y")

conf = generate.random.conf(n, k = 2, scale = FALSE)
plot(conf, xlab = "x", ylab = "y")
```

---

get.movements *Summarize Detected Movements*

Description

get.movements summarizes the results of detect.movement.

Usage

```
get.movements(movement, hm, features = NULL)
```

Arguments

movement	numeric matrix computed with detect.movement.
hm	a hierarchical block matrix computed with hbm, with: $hm_{i,j}$ = the minimal scale (iteration) at which i and j were clustered together, or 0 if i=j
features	one or more feature matrices computed with hbm.features, used to indicate if the movement is detected within a robust feature. Set to NULL by default. When more than one matrix is provided features should be a list of matrices.

Value

`get.movements` returns a data frame with the following columns:
`from`: the moving node,
`to`: the node that `from` has moved towards or away from,
`type`: one of the following values: (-1.0, 0.5, 0.5, 1) where -1 indicates `from` moved away from `to`, 1 indicates `from` moved towards `to`, and -0.5 and 0.5 indicate possible/implicated movements correspondingly,
`scale`: the scale at which the movement was detected, and
`robust` an optional column (when `features` is not `NULL`) with 1 for non-`NA` elements in the `features` matrix, and 0 otherwise.

Author(s)

Yoli Shavit

References

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

`detect.movement` to see how changes between chains are detected
`hbm` to learn how to build hierarchical block matrices
`hbm`'s tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

`hbm` *Build a Hierarchical Block Matrix (HBM)*

Description

`hbm` builds a hierarchical block matrix from an association matrix, typically a symmetric chromatin contact map, by iteratively aggregating clusters.

Usage

```
hbm(m, infl=2, ...)
```

Arguments

`m`	a numeric association matrix, typically a chromatin contact map.
`infl`	numeric giving the inflation parameter for `mcl`, set to 2 by default.
`...`	additional parameters for `mcl`.

Details

hbm iteratively applies Markov Clustering (by calling mcl). In the first iteration, clustering is applied on the input association matrix. The resulting clusters are used to generate a new association matrix to cluster, whose i,j-th entry gives the mean association between all the nodes in the i-th and j-th clusters found in the previous iteration. This is repeated until all clusters are aggregated to a single cluster or when clusters can no longer be aggregated together.

Value

hbm returns a list with the following objects:

hm	The hierarchical block matrix, defined as: $hm_{i,j}$ = the minimal scale (iteration) at which i and j were clustered together, or 0 if i=j
scales	a list of length max(hm)-1 whose k-th entry gives the list of clusters found at k-th scale (iteration).

Author(s)

Yoli Shavit

References

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

mcl for the implementation of Markov Clustering
detect.movement to see how hbm's results are used to detect movements
communicability to see how hbm's results are used to compute the communicability between different locations.
hierarchy to see how hbm's results are used to compute the hierarchy of the association matrix.
hbm's tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Examples

```
set.seed(2)
n = 200 # chain size
# generate chain configuration (random walk/giant loop model)
conf = generate.random.conf(n, sd = 0.5, scale = FALSE)
# generate a contact map like matrix using the model c ~ exp(-d)
control = exp(-1*as.matrix(dist(conf)))
res = hbm(control)
m = res$hm
image(t(m)[,nrow(m):1], axes = FALSE)
ats = seq(0,1,0.2)
lbls = as.character(n*ats)
axis(1, at= ats, labels = lbls, cex.axis = 0.8)
ats = seq(1,0,-1*0.2)
```

```
lbls = as.character(n*seq(0,1,0.2))
axis(2, at= ats, labels = lbls, cex.axis = 0.8)

res$scales
```

`hbm.features` *Compute Robust Features in an HBM*

Description

`hbm.features` computes the main features of a hierarchical block matrix.

Usage

```
hbm.features(m, noise.factor, ncores = 1, ref = NULL, ...)
```

Arguments

`m`	a numeric association matrix, typically a chromatin contact map.
`noise.factor`	numeric vector giving the noise factor to add with `add.noise` at each iteration. The length of this vector will determine the number of iterations.
`ncores`	integer giving the number of cores to register and use. If this is larger than one, iterations will be executed in parallel.
`ref`	hierarchical block matrix computed with `hbm` from `m`. If set to `NULL` this matrix will be computed as part of the execution of `hbm.features`.
`...`	additional parameters for `hbm`.

Details

`hbm.features` adds noise to the given association matrix and executes `hbm` to generate a hierarchical block matrix. Repeating this for multiple iterations (with the same or different noise factor values) gives a mean hierarchical block matrix that can be compared with the matrix computed from the non noisy association matrix.

Value

`hbm.features` returns a list with the following objects:

`noisy.hm`	The average hierarchical block matrix. A numeric matrix whose i,j-th entry gives the mean scale at which i and j were found in the same cluster across `hbm` iterations.
`features`	`noisy.hm` with entries set to `NA` when different from the entries in the original non-noisy hierarchical block matrix.

Author(s)

Yoli Shavit

References

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

add.noise to see how noise is added to matrices
hbm to learn how to build hierarchical block matrices
hbm's tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Examples

```
set.seed(2)
n = 100 # chain size
# generate chain configuration (random walk/giant loop model)
conf = generate.random.conf(n, sd = 0.5, scale = FALSE)
# generate a contact map -like matrix using the model c ~ exp(-d)
control = exp(-1*as.matrix(dist(conf)))
noise = rep(10, 10)
res = hbm.features(control, noise, prune = TRUE, pruning.prob = 0.01)

m = res$features
image(t(m)[,nrow(m):1], axes = FALSE)
ats = seq(0,1,0.2)
lbls = as.character(n*ats)
axis(1, at= ats, labels = lbls, cex.axis = 0.8)
ats = seq(1,0,-1*0.2)
lbls = as.character(n*seq(0,1,0.2))
axis(2, at= ats, labels = lbls, cex.axis = 0.8)
```

---

hierarchy *Compute Hierarchy of an HBM*

Description

hierarchy computes the hierarchy of a hierarchical block matrix computed with hbm.

Usage

```
hierarchy(hm)
```

Arguments

hm	a hierarchical block matrix computed with hbm, with: $hm_{i,j}$ = the minimal scale (iteration) at which i and j were clustered together, or 0 if i=j

Details

In a hierarchical matrix, computed with hbm, the behavior around the diagonal reflects the hierarchy of the association matrix. Specifically, for a hierarchical fractal-like structure we expect a non-decreasing series in the upper triangle of the matrix and a non-increasing series in the lower triangle. hierarchy counts the number of deviations from this behavior for each node: number of negative successive differences up to the diagonal and number of positive successive changes after the diagonal, and returns the negation of the mean number of changes across nodes.

Value

hierarchy returns a numeric value giving the hierarchy of the matrix.

Author(s)

Yoli Shavit

References

hbm's website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

generate.random.conf to see how to generate interesting chains
hbm learn how to build hierarchical block matrices
hbm's tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Examples

```
set.seed(2)

n = 100 # chain size
#generate configurations
conf = generate.random.conf(n, sd = 0.5, scale = FALSE)
#perturb the chain
conf.perturb.all = generate.random.conf(n, perturb = 1:n, sd = 0.5, scale = FALSE)
# and again with less perturbration
conf.perturb = generate.random.conf(n, perturb = 10:50, sd = 0.5, scale = FALSE)

# compute the HBMs
hm.control = hbm(exp(-1*as.matrix(dist(conf))), 2)$hm
hm.perturb.all = hbm(exp(-1*as.matrix(dist(conf.perturb.all))), 2)$hm
```

```
hm.perturb = hbm(exp(-1*as.matrix(dist(conf.perturb))), 2)$hm

h.control = hierarchy(hm.control)
h.perturb = hierarchy(hm.perturb)
h.perturb.all = hierarchy(hm.perturb.all)
h = c(h.control, h.perturb, h.perturb.all)

# plot
plot(1:3, h, pch = 19, cex = 2, axes = FALSE, ylab = "Chain Hierarchy", xlab = "Condition")
axis(1, at = 1:3, labels = c("Control", "Perturbed-Partial", "Perturbed-All"))
axis(2)
```

`mcl` *Markov Clustering*

Description

`mcl` implements the Markov Clustering algorithm (van Dongen, 2000) with a fixed expansion parameter (=2).

Usage

```
mcl(m, infl, iter = 1000, remove.self.loops = FALSE, prune = FALSE,
     thresh = 1e-06, pruning.prob = 1e-06, use.sparse = NULL, verbose = FALSE)
```

Arguments

`m`	A numeric matrix, given as input to the Markov Clustering algorithm.
`infl`	numeric. The inflation parameter for the Markov Clustering algorithm.
`iter`	integer giving the maximal number of iterations for the Markov Clustering algorithm, set to 1000 be default (in practice the algorithm is shown to converge after 10-100 iterations).
`remove.self.loops`	boolean indicating whether to remove self loops (i.e. set diagonal entries to 0), set to FALSE by default.
`prune`	boolean indicating whether to prune small probabilities (i.e. set to 0) in the transition matrix, set to FALSE by default.
`thresh`	a numeric giving the difference threshold below which the transition matrix is considered to have converged.
`pruning.prob`	numeric giving the threshold below which pruning should be applied, when `prune` is `TRUE`. Set to `1e-06` be default.
`use.sparse`	a boolean indicating whether to use sparse matrices. By default this value is set to NULL, so that sparse matrices are used only if the transition matrix is sparse enough to justify this representation (50% sparsity and above). When set to `TRUE` (`FALSE`), `use.sparse` will force (disable) the use of sparse matrices.

`verbose` boolean indicating whether to print the number of iterations before convergence was achieved.

Details

`mcl` is called from `hbm` to build a hierarchical block matrix from an association matrix, typically a chromatin contact map.

Value

`mcl` returns a vector whose i-th entry is the cluster identifier of the i-th node, and two nodes are in the same cluster iff they have the same cluster identifier.

Author(s)

Yoli Shavit

References

Stijn van Dongen. A cluster algorithm for graphs. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, May 2000.

`hbm`'s website: http://www.cl.cam.ac.uk/~ys388/hbm/

See Also

`hbm` to learn how to build hierarchical block matrices
`hbm`'s tutorials at http://www.cl.cam.ac.uk/~ys388/hbm/

Index

add.noise, 2, *11, 12*

communicability, 3, *10*

detect.movement, 5, *8–10*

generate.random.conf, 7, *13*
get.movements, *6*, 8

hbm, *3–6*, *8, 9*, 9, *11–13*, *15*
hbm-package, 2
hbm.features, *3*, *8*, 11
hierarchy, *10*, 12

jitter, *2, 3*

mcl, *9, 10*, 14