Package 'MeDiChI'

February 15, 2013

Type Package

Title MeDiChI ChIP-chip deconvolution library

Version 0.4.0

Date Oct 12 2010

Author David J Reiss

Maintainer David J Reiss <dreiss@systemsbiology.org>

Description Model-based deconvolution of genome-wide binding (ChIP-chip) data

License GPL-2

Depends R (>= 2.10), lars (>= 0.9.5), quadprog (>= 1.4.10), corpcor (>= 1.4.5), Matrix (>= 0.99875.0)

Suggests zoo (>= 1.3-1), lattice, multicore (>= 0.1-3), doMC (>= 1.2.0)

URL http://baliga.systemsbiology.net/medichi

Repository CRAN

Repository/R-Forge/Project medichi

Repository/R-Forge/Revision 11

Date/Publication 2012-07-23 10:35:31

NeedsCompilation no

R topics documented:

MeDiChI-package *MeDiChI ChIP-chip deconvolution library*

Description

Deconvolution of Chromatin-IP-to-microarray (ChIP-chip) data to identify binding sites at high resolution across the genome.
To be published in Bioinformatics and submitted to Bioconductor.
This package includes all data described in the manuscript.

Details

Package:	MeDiChI
Type:	Package
Version:	0.1-4
Date:	2007-11-25
License:	GPL version 3

demo(MeDiChI)

Author(s)

David J Reiss, Institute for Systems Biology

Maintainer: <dreiss@systemsbiology.org>

References

(1). Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", Bioinformatics; doi: 10.1093/bioinformatics/btm592. http://baliga.systemsbiology.net/medichi
(2) Qi, Y and et al. (2006). "High-resolution computational models of genome binding events", Nature Biotechnol, 24(8), 963-970. http://cgs.csail.mit.edu/jbd.

See Also

chip.deconv, deconv.entire.genome, fit.peak.profile, generate.fake.data, generate.binding.profile, MeDiChI-data, <lars>, <quadprog>, <Matrix>

Examples

```
demo( MeDiChI )

## Run the demo yourself:
data( "halo.lowres", package="MeDiChI" )
```

```
fit <- chip.deconv( data.halo.lowres, where="Chr", fit.res=30,
               center=650000, wind=20000, max.steps=100, n.boot=10,
               kernel=kernel.halo.lowres, verbose=TRUE, boot.sample.opt="case" )

coef( fit ) ## Print out the coefficients
plot( fit, plot.genes=TRUE, cex=0.5, cex.lab=0.8, cex.axis=0.8 )
```

---

chip.deconv *High-resolution model-based deconvolution of normalized ChIP-chip data derived from tiling arrays.*

Description

Deconvolves a subset of data on one chromosome, including multiple replicates, and running multiple bootstraps, if desired. To deconvolve an entire data set across multiple chromosomes, see 'deconv.entire.genome'.

Usage

```
chip.deconv(data, where = NA, center = NA,
window = 30000, fit.res = 10, max.steps = 200,
post.proc.factor = 2, min.npeaks = 0, max.npeaks = 99999,
selection.method = "bic", quant.cutoff = "q0.85",
n.boot = 1, boot.sample.opt = c("residual","resample","case","wild",
"position","replicate")[1], max.peak = NA, boot.vary.res = F,
kernel = NA, tile.distance = NA, verbose = T, trace = F, ...)


plot.chip.deconv(x, boot.results = c("scaled.prob", "prob", "scale",
"conf=95", "NONE")[1], where = NA, center = NA, window = NULL, verbose = F,
plot.genes = F, org = NA, hi.res = NA, quants = c(0.95, 0.5, 0.05),
smooth = T, ... )



print.chip.deconv(x, ...)

coef.chip.deconv(object, ...)
```

Arguments

data	Input data matrix, connection, or filename. See Details.
center	Central chromosomal coordinate for the subset of data to deconvolve, or to be plotted. See Details.
where	The chromosome of the subset of data to deconvolve. See Details.

window	The window size (in base-pairs) of the subset of data to deconvolve; see Details. For 'plot.chip.deconv', limit the window size that is plotted. The default ('NULL'), is to use the 'window that was input to 'chip.deconv()'.
fit.res	Desired deconvolution resolution (base-pairs).
kernel	Required deconvolution kernel. See Details.
max.steps	Limit the number of LARS steps taken. Should opt towards higher values (above 200), as the fit from the highest step is used to estimate the noise for BIC.
post.proc.factor	Post-processing filter for combining deconvolution coefficients, in units of (n*fit.res).
min.npeaks	Optionally limit the minimum number of coefficients. Default is '0' – no lower limit.
max.npeaks	Optionally limit the maximum number of coefficients. Default is '99999' – no upper limit.
selection.method	Use argmin(method) to choose optimal model, one of c('bic','aic'). Default is 'bic'.
quant.cutoff	Intensity or quantile cutoff for data to be processed. Limits the locations of potential sites to those near probes that are above this quantile. See Details. Default is 'q0.85' – use 85th quantile.
n.boot	Number of bootstrap iterations to perform. If '0' (default) then no bootstraps are performed.
boot.sample.opt	Bootstrap resampling option. See Details. Default is 'case.
max.peak	Any coefficient with an intensity above this threshold is set to this value. Default is 'NA' – no cutoff.
boot.vary.res	If TRUE, vary the resolution (around 'fit.res') during bootstraps. May result in more realistic solutions.
tile.distance	Distance (in base pairs) between adjacent probes on the array. If 'NA' (the default) then this is computed from the data.
verbose	If TRUE, print out status messages.
trace	If TRUE, print out LARS progress.
x	Object output from 'chip.deconv()'
object	Object output from 'chip.deconv()'
boot.results	Plot bootstrap distributions, either 'prob' (posterior probability), 'scaled.prob' (intensity-scaled posterior probability), 'conf=95' (95% confidence intervals), or 'NONE' (no bootstrap results plotted). Default is 'scaled.prob'.
plot.genes	If TRUE, include gene positions in the plot. Currently only supported for Halobacterium and S. cerevisiae.
org	Organism used for 'plot.genes', one of either 'halo' or 'yeast'.
quants	Quantiles of model fits from bootstraps to include in plot (see Figure 3 in paper). Default is to plot 5th, 50th, and 95th quantiles.
hi.res	Compute high-resolution model fits at this resolution. Default is 'fit.res' as provided to 'chip.deconv()'.

smooth	If TRUE, plot kernel-smoothed bootstrap distributions, rather than simple counts.
...	Additional parameters passed to 'lars' or 'plot' or 'read.table' (if the 'data' parameter is a file name).

Details

'chip.deconv' is used to deconvolve a limited subset of the entire tiling array data set. Usually this subset is limited to a single chromosome (the 'where' parameter) and a range of coordinates on the chromosome between 'center - window / 2' and 'center + window / 2'. The method will identify potential binding sites (coefficients) across this range at a resolution given by 'fit.res', up to a maximum of n.coeffs = window / fit.res coefficients.

It is not recommended (due to memory constraints) to attempt to fit more than a few thousand potential coefficients using this function. This number may be decreased by either decreasing the 'window' size or increasing the 'fit.res'. To deconvolve larger windows at high resolution, use 'deconv.entire.genome'.

The input 'data' can be formatted as one of:
(a) a 2-column matrix or data-frame containing probe coordinates (column 1) and intensities (column 2), with optional rownames containing the chromosome identifier for each probe, or
(b) a 3-column data frame containing probe chromosome identifiers (column 1), coordinates (column 2) and intensities (column 3), or
(c) a file name or connection pointing to either a GFF file or a 3-column tab-delimited file formatted as in (b).

Replicates are included in the input data simply as additional rows with the same probe coordinate.

Note that 'intensities' in the above description refers to relative (potentially normalized) intensities, exponentiated log-ratios (e.g. versus a reference) or some such quantity. The data should not be logged, as this will not conform with the MeDiChI peak model (see the reference below for details).

If the entire input 'data' matrix is to be processed using this function (only recommended if a subset of the entire array has been pre-selected; see above), or if the data only cover one chromosome, then the chromosome identifiers are not required.

If the input 'data' contains the entire array (including multiple chromosomes) then only the contiguous subset of probes within the window described above and with the matching chromosome identifier will be deconvolved.

'quant.cutoff' can be either a character starting with "q", e.g. "q0.85" to represent a quantile cutoff, or a numeric value, representing an absolute intensity cutoff. Only positions within +/- one 'tile.size' of any probe that has an intensity greater than this cutoff are considered to contain potential binding sites. This parameter may be used to decrease the runtime of the function.

'boot.sample.opt' can be one of 'wild' (Default) – wild resampling (see http://en.wikipedia.org/wiki/Bootstrapping_(statistics)#Wild_bootstrap); 'residual' – wild resampling, only run on residuals for estimation of coefficient p-values; 'case' – case resampling; 'position' – resample the central positions of the probes; 'replicate' – resample probe intensities from the range

given by their replicates (if replicates exist); or 'resample' – case sampling of intensities only for estimation of coefficient p-values.

'kernel' is a 2-column matrix providing position (column 1) and intensity (column 2) of the deconvolution kernel (profile model) to be used. This may be computed using 'generate.binding.profile' and parameters for this model may be learned from the data using 'fit.peak.profile'.

Value

A list of class 'chip.deconv', for which 'plot', 'print', and 'coef' functions exist, and containing the following elements (repeated 'n.boot' times for each bootstrap run):

`data`	The input data subset (subset selected using 'where', 'window' and 'center'; see Description.
`fit`	Best-fit values at the locations of each probe in the input data.
`kernel`	Kernel used for deconvolution (provided by 'kernel' parameter).
`coeffs`	Non-zero coefficients (coordinate and intensity) for the chosen best-fit model.
`out.info`	Statistics on the best-fit solution including the LARS step number, BIC, RSS, etc.
`args`	All parameters input to 'chip.deconv', used for plotting and future reference.

Author(s)

David J Reiss, Institute for Systems Biology

Maintainer: <dreiss@systemsbiology.org>

References

Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", Bioinformatics; doi: 10.1093/bioinformatics/btm592.

http://baliga.systemsbiology.net/medichi

See Also

deconv.entire.genome, fit.peak.profile, generate.fake.data, generate.binding.profile, MeDiChI-data, `lars`, `quadprog`, `Matrix`

Examples

```
## see 'help(MeDiChI)', or...
## Run the demo yourself:

data( "halo.lowres", package="MeDiChI" )

fit <- chip.deconv( data.halo.lowres, where="Chr", fit.res=30,
```

```
                center=650000, wind=20000, max.steps=100, n.boot=10,
                kernel=kernel.halo.lowres, verbose=TRUE, boot.sample.opt="case" )

plot( fit, plot.genes=TRUE, cex=0.5, cex.lab=0.8, cex.axis=0.8 )
```

`deconv.entire.genome` *High-resolution model-based deconvolution of normalized ChIP-chip data derived from tiling arrays.*

Description

Deconvolves an entire ChIP-chip data set, across all chromosomes, including multiple replicates, and running multiple bootstraps, if desired, by running 'chip.deconv' on multiple, contiguous, overlapping subsets of the data and combining the results.

Usage

```
deconv.entire.genome(data, chroms=NA, window=6000, step.by=5500,
centers=NULL, quiet=F, plot=F, n.boot=1, fit.res=10, quant.cutoff="q0.85",
max.peak=NA, verbose=F, kernel=NA, no.multicore=T, ... )

print.chip.deconv.entire.genome(x, ...)

plot.chip.deconv.entire.genome(x, where = NA, center = NA, window = NULL, ...)

coef.chip.deconv.entire.genome(object, ...)
```

Arguments

`data`	Input data matrix, connection, or filename. See 'chip.deconv'.
`chroms`	Character vector listing chromosomes to include in analysis. The default ('NA') is to include all chromosomes. See 'chip.deconv' for more information.
`window`	Size (in bp) of chunks to be processed one-at-a-time (see 'chip.deconv'). For 'plot.chip.deconv.entire.genome', the coord. range that is plotted; the default ('NULL') is to use the 'window' that was input to 'deconv.entire.genome()'.
`step.by`	Increment (in bp) in increment the 'center' coordinate of each 'chip.deconv' call.
`centers`	Optional vector of 'centers' to use (overrides 'step.by' option. Vector may be optionally named by chromosome.
`quiet`	If TRUE, be very quiet.
`plot`	If TRUE, plot each chunk as it is deconvolved.
`n.boot`	Number of bootstrap iterations. See 'chip.deconv'.
`fit.res`	Deconvolution resolution. See 'chip.deconv'.
`quant.cutoff`	See 'chip.deconv'.

max.peak	See 'chip.deconv'.
verbose	If TRUE, print out status messages. See 'chip.deconv'.
kernel	Required deconvolution kernel. See 'chip.deconv'.
no.multicore	Prevent use of multiple cores, even if 'multicore' is installed.
x	Object output from 'deconv.entire.genome()'
where	The chromosome(s) to be plotted. See Details.
center	Central chromosomal coordinate(s) to be plotted. See Details.
object	Object output from 'deconv.entire.genome()'
...	Further parameters passed to 'chip.deconv' or 'plot.chip.deconv'.

Details

'deconv.entire.genome' breaks the full data set in to contiguous (in chromosomal coordinate) chunks of size 'window' and processes these using 'chip.deconv'. It increments the center of the 'window' by 'step.by' base pairs, and combines the resulting fit by averaging nearby peaks in the overlapping segments. The defaults were chosen to contain overlaps of about 10%, and constrain the required memory usage for each deconvolution.

'deconv.entire.genome' distributes the processing of multiple sections of the chromosome to multiple processor cores, unless 'no.multicore' is set to TRUE.

'plot.chip.deconv.entire.genome' will, by default, plot the entire fit across all chromosomes, via multiple plots, in chunks of size 'window'. It is recommended to send this to a postscript or pdf device (rather than the screen). See 'plot.chip.deconv' for more information.

Value

A list of class 'chip.deconv.entire.genome', for which 'plot', 'print', and 'coef'. This list is comprised of 'chip.deconv' objects (see 'chip.deconv') for each chromosome. See 'Example' for more information. The list is itemized as:

fits.fin	The fits combined into a single one for each chromosome.

Author(s)

David J Reiss, Institute for Systems Biology

Maintainer: <dreiss@systemsbiology.org>

References

Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", Bioinformatics; doi: 10.1093/bioinformatics/btm592.

http://baliga.systemsbiology.net/medichi

See Also

chip.deconv, plot.chip.deconv, fit.peak.profile, generate.fake.data, generate.binding.profile, MeDiChIdata, lars, quadprog, Matrix

Examples

```
data( "halo.lowres", package="MeDiChI" )
## Deconvolve all three replicons: "Chr", "pNRC100", "pNRC200". Note
## this will take a while to run.

## Not run:
fits <- deconv.entire.genome( data.halo.lowres, fit.res=30,
               n.boot=1, kernel=kernel.halo.lowres, verbose=FALSE, trace=FALSE )

## Plot the entire fit, across many windows:
plot( fits )

## Plot a piece of the finished product on just the chromosome:
plot( fits$fits.fin$Chr, center=20000, wind=10000 )

## End(Not run)
```

---

fit.peak.profile *Iteratively learn deconvolution kernel (peak profile) from real ChIP-chip data.*

Description

Learn the peak profile from ChIP-chip data, to be used for data deconvolution via 'chip.deconv' or 'deconv.entire.genome'.

Usage

```
fit.peak.profile(data, tile.size, n.peaks = 30, n.skip = 10, in.kernel =
NA, fits = NULL, method = "Nelder-Mead", positions=c( 0, 25, 50, 100,
150, seq( 200, (mini.window*1.5)+1, by=100 ) ), re.fit = 25, start.pars
= c(shape = 7, scale = 50, bs.size = 20, h.cutoff = 15), to.be.fit=c(
"shape", "scale", "bs.size", "h.cutoff" ), rnd = F, mini.window = max(5
* tile.size, 1300), plot = T, name = "", no.multicore=T, ...)

plot.fit.peak.profile(x, n.peak.plot = 7, plot.spline = F, ...)
```

Arguments

data	Input data matrix, connection, or file name. See 'chip.deconv' for details.
tile.size	Probe length (bp) for use in kernel generation. See 'generate.binding.profile' for details.
n.peaks	Number of (biggest) peaks to learn from

n.skip	Skip this many worst-fitting of the 'n.peaks'. Enables filtering out of peaks that don't agree with the majority of isolated peaks.
in.kernel	Input seed kernel to start with (guessed if 'NA').
fits	Input preliminary fits (via 'deconv.entire.genome'; generated from input data if 'NULL').
method	Optimization method, see 'optim'
positions	Passed directly to 'generate.binding.profile'
re.fit	Re-run 'deconv.entire.genome' on input data using current best-fit profile, every 're.fit' iterations
start.pars	Starting parameters for model profile. See Details.
to.be.fit	Names of parameters to be learned. See Details.
rnd	Add some 'jitter' to the starting parameters
plot	Plot the fit as it progresses (including data and fit to several bright peaks in the data)
mini.window	Size of window to be plotted around each of the bright peaks, if 'plot' is TRUE
no.multicore	Prevent use of multiple cores, even if 'multicore' is installed.
name	Ignored
...	Further parameters for 'deconv.entire.genome' and 'generate.binding.profile'
x	Object output from 'fit.peak.profile'
n.peak.plot	Number of bright peaks (and their fits) to include in the plot
plot.spline	Plot a 'smooth.spline' fit to the data for reference

Details

'fit.peak.profile' iteratively runs 'deconv.entire.genome' on a data set, isolates the 'n.peaks' brightest peaks, and fits parameters to the model profile use by 'generate.binding.profile', then re-fits the data using 'deconv.entire.genome'. Currently, the parameters that are fit include the shape and scale of the Gamma function used for the fragment length distribution, the binding site footprint, and the "cutoff" for hybridization. All parameters listed in 'to.be.fit' will be optimized. Parameters not listed in 'start.pars' will be set to the defaults.

Value

A list of class 'fit.peak.profile, for which a 'plot' function exists, and containing the following elements:

score	The log of the RSS of the final peak profile to the 'n.peaks - 'n.skip' brightest isolated peaks in the data (this is the measure that was optimized).
kernel	The final peak profile, generated by 'generate.binding.profile'.
new.fits	The deconvolution fits of the final peak profile to the 'n.peaks' brightest peaks in the data, generated by 'chip.deconv'.
is.bad	Logical vector telling which of the 30 'n.peaks' brightest peaks were not among the 'n.skip' poorly-fitting peaks.

par	Final best-fit model parameters for the peak profile.
peaks	Two-column matrix listing the centers and intensities of the 'n.peaks' brightest peaks in the data, that were used for the fitting.
args	All parameters input to 'fit.peak.profile', for future reference.

Author(s)

David J Reiss, Institute for Systems Biology

Maintainer: <dreiss@systemsbiology.org>

References

Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", Bioinformatics; doi: 10.1093/bioinformatics/btm592.

http://baliga.systemsbiology.net/medichi

See Also

chip.deconv, deconv.entire.genome, generate.fake.data, generate.binding.profile, MeDiChI-data, lars, quadprog, Matrix

Examples

```
  ## Fit the peak profile to the high-resolution Nimblegen data,
  ## plotting progress. Note this will take some time.
  data( "halo.hires", package="MeDiChI" )

## Not run:
   params <- fit.peak.profile( data.halo.hires, tile.size=50,
                              quant.cutoff="q0.99", chrom="Chr",
                              fit.res=30, max.steps=100, plot=TRUE )
   plot( params )

  ## Use the output kernel for deconvolution (see 'deconv.entire.genome'):
   fits <- deconv.entire.genome( data.halo.lowres, fit.res=10,
               n.boot=1, kernel=params$kernel, verbose=TRUE, trace=FALSE )

## End(Not run)
```

---

generate.binding.profile

Construct the binding profile ('kernel') required as input for 'chip.deconv' and 'deconv.entire.genome'.

Description

Construct a model-based binding profile as described in Reference (1).

Usage

```
generate.binding.profile(fragment.distrib = function(x, ...) dgamma(x,
shape = 6, scale = 50), bs.size = 1, tile.size = 50, min.frag.size = 0,
positions = seq(0, 1001, by = 50), intensity.scaling = function(x, ...)
x, hybridization.prob = function(x, ...) as.integer(x > 10), interp = T,
plot = F, verbose = F, no.multicore=T, ...)
```

Arguments

`fragment.distrib`	Distribution of DNA fragment lengths.
`bs.size`	Footprint of TF binding site on the genome.
`tile.size`	Length of the array probes (in bp).
`min.frag.size`	Cutoff for minimum DNA fragment size.
`positions`	The distances from the center of the profile for which to compute the relative intensity. A longer vector increases the running time of this procedure. Values for distances other than those listed here are computed via interpolation. See 'interp'.
`intensity.scaling`	Intensity of DNA fragments as a function of their length.
`hybridization.prob`	Hybridization probability as a function of length of complementary sequence.
`interp`	If TRUE, interpolate the values for distances other than those listed in 'positions'.
`plot`	If TRUE, plot the resulting profile.
`verbose`	If TRUE, be verbose.
`no.multicore`	Prevent use of multiple cores, even if 'multicore' is installed.
`...`	Further parameters for 'fragment.distrib', 'intensity.scaling', and 'hybridization.prob'.

Details

No details.

Value

A two-column matrix containing positions (distance from center, in bp) and relative intensities of the profile.

Author(s)

David J Reiss, Institute for Systems Biology

Maintainer: <dreiss@systemsbiology.org>

References

(1) Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", Bioinformatics; doi: 10.1093/bioinformatics/btm592.
http://baliga.systemsbiology.net/medichi
(2) Qi, Y and et al. (2006). "High-resolution computational models of genome binding events", Nature Biotechnol, 24(8), 963-970. http://cgs.csail.mit.edu/jbd.

See Also

chip.deconv, deconv.entire.genome, fit.peak.profile, generate.fake.data, MeDiChI-data

Examples

```
## Compare profiles for DNA fragment distrs. with mean=300 and 400 bp.
kern.300 <- generate.binding.profile( fragment=function(x) dgamma( x,
                    shape=6, scale=50 ), verbose=TRUE )
kern.400 <- generate.binding.profile( fragment=function(x) dgamma( x,
                    shape=8, scale=50 ), verbose=TRUE )
plot( kern.300, typ="l", col="red" )
lines( kern.400, col="green" )
```

`generate.fake.data` *Generate simulated ChIP-chip data.*

Description

Generate simulated ChIP-chip data, as described in Reference (1).

Usage

```
generate.fake.data(posns = seq(1, 6001, by = 20), n.pts = 5, noise =
0.1, min.pk = 0.1, plot = F, verbose = F, in.pts = NULL, kernel = NULL,
tile.size = 100, reps = 1,
noise.func = function(data)noise*(1+sqrt(data)), ...)
```

Arguments

`posns`	Probe centers
`n.pts`	Number of (randomly-chosen) binding sites to add
`noise`	Level of noise (as described in Reference (1))
`min.pk`	Minimum peak intensity
`plot`	Plot the generated data?
`verbose`	Be verbose?
`in.pts`	2-column matrix with positions (column 1) and intensities (column 2) of input peaks; if n.pts is 'NA'.

kernel	Input peak profile to use.
tile.size	Input probe length (used to generate peak profile kernel if 'kernel' is 'NULL')
noise.func	Function used to generate noise as a function of signal intensity
reps	Number of replicate intensities per simulated probe
...	Further parameters for 'generate.binding.profile' (if 'kernel' is 'NULL')

Details

No details.

Value

A list of class containing the following three elements:

input	A two-column matrix containing positions and intensities of input binding sites
data	A two-column matrix containing positions and intensities of simulated probes, can be passed directly to 'chip.deconv'
kernel	The peak profile kernel used to generate the data (as produced by 'generate.binding.profile')

Author(s)

David J Reiss, Institute for Systems Biology

Maintainer: <dreiss@systemsbiology.org>

References

Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", Bioinformatics; doi: 10.1093/bioinformatics/btm592.
http://baliga.systemsbiology.net/medichi

See Also

chip.deconv, generate.binding.profile

Examples

```
## Generate data with 2 peaks at positions 3000 and 4000, deconvolve the
## data, and plot the resulting data and fit.
kern.300 <- generate.binding.profile( fragment=function(x) dgamma( x,
                    shape=6, scale=50 ), verbose=TRUE )
data <- generate.fake.data( in.pts=cbind( c( 3000, 4000 ), c( 1, 0.7 ) ), reps=3,
                    kernel=kern.300 )
plot( data$data, pch=20 )
fit <- chip.deconv( data$data, center=NA, wind=NA, kernel=data$kernel,
                   fit.res=30, n.boot=10, verbose=TRUE, boot.sample.opt="case" )
plot( fit, boot="prob.scaled" )
print( fit )
```

MeDiChI-data *ChIP-chip data supporting the MeDiChI package*

Description

Three different ChIP-chip datasets of varying tiling resolution, referenced in Reference (1) below.

Usage

```
data(halo.hires)
data(halo.lowres)
data(yeast.gcn4)
```

Format

The format of "halo.lowres" is: chr [1:3] "data.halo.lowres" "kernel.halo.lowres" "gene.coords" The format of "halo.hires" is: chr [1:3] "data.halo.hires" "kernel.halo.hires" "gene.coords" The format of "yeast.gcn4" is: chr [1:4] "data.yeast.gcn4" "kernel.yeast.gcn4" "kernel.yeast.gcn4.jbd" "gene.coords"

Details

"halo.lowres" loads:
"data.halo.lowres" – low-resolution HaloSpan Halobacterium sp. TfbD ChIP-chip data, formatted for input to chip.deconv() and deconv.entire.genome(). Data is described in ref. (1) below.
"kernel.halo.lowres" – deconvolution kernel learned for this data via fit.peak.profile().
"gene.coords" – map of gene coordinates for this organism for use in plotting functions.
"halo.hires" loads:
"data.halo.hires" – high-resolution Nimblegen format Halobacterium sp. TfbD ChIP-chip data, formatted for input to chip.deconv() and deconv.entire.genome(). Data is described in ref. (1) below.
"kernel.halo.hires" – deconvolution kernel for this data via fit.peak.profile().
"gene.coords" – map of gene coordinates for this organism for use in plotting functions.
"yeast.gcn4" loads:
"data.yeast.gcn4" – 266bp-resolution S. cerevisiae Gcn4 ChIP-chip data, formatted for input to chip.deconv() and deconv.entire.genome(). Data is described in ref. (1) below.
"kernel.yeast.gcn4" – deconvolution kernel for this data via fit.peak.profile().
"kernel.yeast.gcn4.jbd" – deconvolution kernel used by ref. (2) for the JBD algorithm.
"gene.coords" – map of gene coordinates for this organism for use in plotting functions.

Source

See references below.

References

(1). Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", Bioinformatics; doi: 10.1093/bioinformatics/btm592. http://baliga.systemsbiology.net/medichi
(2) Qi, Y and et al. (2006). "High-resolution computational models of genome binding events", Nature Biotechnol, 24(8), 963-970. http://cgs.csail.mit.edu/jbd.

See Also

chip.deconv, deconv.entire.genome, fit.peak.profile, generate.fake.data, generate.binding.profile

Examples

```
data(halo.hires)

## Then try print(data.halo.lowres) to see the format.

data(halo.lowres)
data(yeast.gcn4)
```

Index