Package ‘CAFE’

March 25, 2015

Type Package

Title Chromosomal Aberrations Finder in Expression data

Version 1.3.0

Date 2014-03-16

Author Sander Bollen

Maintainer Sander Bollen <sander.h.bollen@gmail.com>

Depends R (>= 2.10), biovizBase, GenomicRanges, IRanges, ggbio

Imports affy, ggplot2, annotate, grid, gridExtra, tcltk, Biobase

Suggests RUnit, BiocGenerics, BiocStyle

Description Detection and visualizations of gross chromosomal aberrations using Affymetrix expression microarrays as input

License GPL-3

ByteCompile true

biocViews GeneExpression, Microarray, OneChannel, GeneSetEnrichment


NeedsCompilation no

R topics documented:

CAFE-package .................................................. 2
armStats ......................................................... 3
bandStats ........................................................ 4
CAFE_data ........................................................ 5
chromosomeStats ................................................. 6
cliSubset ........................................................ 7
discontPlot ........................................................ 8
discontSmooth .................................................... 9
facetPlot .......................................................... 10
fisher.method .................................................... 11
guiSubset ......................................................... 12
CAFE-package

Description

CAFE attempts to find chromosomal aberrations in microarray expression (mRNA) data. It contains several plotting functions to aid in visualizing these aberrations. It generally recapitulates the workflow described by Mayshar et al (see references), and implements several algorithms described by Friedrich et al (see references).

Details

Package:     CAFE
Type:       Package
Version:    0.6.9.5
Date:       2013-05-16
License:    GPLv3

Author(s)

Sander Bollen

References


Examples

```r
## Not run:
setwd("/some/path/to/cel/files")
data <- ProcessCels()
# process cel files
samples <- c(1,2)
# select samples 1 and 2 to compare against the rest
```
armStats

Find aberrations with chromosome arm resolution

Description

Calculate significant chromosomal arms with various statistical tests

Usage

armStats(datalist, chromNum=1, arm="q", samples=NULL, select="cli", test="fisher", bonferroni = TRUE, enrichment = "greater")

Arguments

datalist The CAFE datalist to be analyzed, i.e. the output of ProcessCels.

chromNum The chromosome to be calculated. This can be "ALL" to calculate all chromosmes.

arm Select which arm - "q" or "p" - to analyse

samples A vector containing sample numbers to be analyzed

select Signifies which type of sample selection prompt will be shown, if samples=NULL. Currently supported are "cli" for a command line interface and "gui" for a tcl/tk-based graphical user interface.

test Signifies which statistical test to be used in the final calculation. Must be either "fisher" for an exact fisher test or "chisqr" for a chi square test.

bonferroni If bonferroni=TRUE, will correct the p-values of the enrichment test with a bonferroni method.

enrichment Test for over or underexpression. Can be set to "greater" or "less".
Value

A named vector containing p-values.

Note

Technically speaking, the Fisher’s exact test is better than the chi-square test; the Fisher’s exact test gives an exact p-value, whereas the chi-square test only gives an approximation. However, the Fisher’s exact test can get slow for large sample sizes, and the chi-square test becomes better with increasing sample size but does not slow down as much.

Author(s)

Sander Bollen

See Also

`chromosomeStats` `bandStats`

Examples

data("CAFE_data")
armStats(CAFE_data, chromNum="ALL", samples=c(1,3), arm="p")

---

Description

Calculate significant chromosome bands with various statistical tests

Usage

`bandStats(datalist, chromNum=1, samples=NULL, select="cli", test="fisher", bonferroni = TRUE, enrichment = "greater")`

Arguments

datalist       The CAFE datalist to be analyzed, i.e. the output of `ProcessCels`.
chromNum       The chromosome to be calculated. This can be "ALL" to calculate all chromosomes.
samples        A vector containing sample numbers to be analyzed
select          Signifies which type of sample selection prompt will be shown, if samples=NULL. Currently supported are "cli" for a command line interface and "gui" for a `tcltk`-based graphical user interface.
test            Signifies which statistical test to be used in the final calculation. Must be either "fisher" for an exact fisher test or "chisqr" for a chi square test.
bonferroni If bonferroni=TRUE, will correct the p-values of the enrichment test with a bonferroni method.
enrichment Test for over or underexpression. Can be set to "greater" or "less".

Value
A named vector containing p-values if testing a single chromosome. If chromNum="ALL", the output will be a two-column data frame, with cytoband names in the first column and p-values in the second column.

Note
Technically speaking, the Fisher’s exact test is better than the chi-square test; the Fisher’s exact test gives an exact p-value, whereas the chi-square test only gives an approximation. However, the Fisher’s exact test can get slow for large sample sizes, and the chi-square test becomes better with increasing sample size but does not slow down as much.

Author(s)
Sander Bollen

See Also

 chromosomeStats armStats

Examples

data(CAFE_data)
bandStats(CAFE_data, chromNum=17, samples=c(1,3), test="fisher")
Format

A list containing two lists

whole  A list containing a dataframe for each sample
over  A list containing a dataframe for each sample, but with only those probes that are deemed overexpressed

The dataframes inside the lists contain the following columns:

- **id**: Affymetrix probe IDs
- **sym**: Gene symbols
- **value**: Log2 transformed expression values
- **logrel**: Log2 transformed relative expression values (to the median)
- **loc**: Chromosomal locations
- **chr**: Chromosome identifiers

Source


Examples

data("CAFE_data")

---

**chromosomeStats**  Find aberrations with whole-chromosome resolution

Description

Calculate significant chromosomes with various statistical tests

Usage

chromosomeStats(datalist, chromNum=1, samples=NULL, select="cli", test="fisher", bonferroni = TRUE, enrichment = "greater")

Arguments

- **datalist**: The CAFE datalist to be analyzed, i.e. the output of ProcessCels.
- **chromNum**: The chromosome to be calculated. This can be "ALL" to calculate all chromosomes.
- **samples**: A vector containing sample numbers to be analyzed
- **select**: Signifies which type of sample selection prompt will be shown, if samples=NULL. Currently supported are "cli" for a command line interface and "gui" for a tcl/tk-based graphical user interface.
test
Signifies which statistical test to be used in the final calculation. Must be either "fisher" for an exact fisher test or "chisqr" for a chi square test.

bonferroni
If bonferroni=TRUE, will correct the p-values of the enrichment test with a bonferroni method.

enrichment
Test for over or underexpression. Can be set to "greater" or "less".

Value
A named vector containing p-values.

Note
Technically speaking, the Fisher's exact test is better than the chi-square test; the Fisher's exact test gives an exact p-value, whereas the chi-square test only gives an approximation. However, the Fisher's exact test can get slow for large sample sizes, and the chi-square test becomes better with increasing sample size but does not slow down as much.

Author(s)
Sander Bollen

See Also
bandStats armStats

Examples
```r
data("CAFE_data")
sam <- c(9,11)
chromosomeStats(CAFE_data,chromNum=17,samples=sam,test="fisher")
```

cliSubset

Subset data with a CLI

Description
Provides command line interface for subsetting input datasets

Usage
```r
cliSubset(datalist,alternative)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datalist</td>
<td>the dataset to be subsetted</td>
</tr>
<tr>
<td>alternative</td>
<td>&quot;greater&quot; or &quot;less&quot;</td>
</tr>
</tbody>
</table>
discontPlot

Value

subset of input

Author(s)

Sander Bollen

See Also

guiSubset

Examples

## Not run:
datalist <- data("CAFE_data")
sub <- cliSubset(datalist,alternative="greater")

## End(Not run)

discontPlot(datalist,samples=c(1,2),chromNum=1,gamma=300,idiogram=FALSE,
file="default")

Arguments

dataList The CAFE datalist to be analyzed, i.e. the output of ProcessCels.
samples A vector or sample numbers to be plotted
chromNum the chromosome to be plotted
gamma The gamma level can be roughly compared to the sliding window size in a normal continuous smoother. The gamma level determines how strict the algorithm functions; a higher level will correspond to fewer jumps. This can not be higher than the total number of probesets on the to-be-analyzed chromosome. Must be a positive integer.
idiogram if TRUE, will overlay a chromosome idiogram over the chromosome plot
file Specify a file name to store output png file

Description

Plots chromosome plots with a discontinuous smoother

Usage

discontPlot(datalist,samples=c(1,2),chromNum=1,gamma=300,idiogram=FALSE,
file="default")
Value

Plot to file system; Returns a ggplot2 graph if chromNum!="ALL". When chromNum="ALL", returns a list of ggplot2 graphs.

Author(s)

Sander Bollen

References


See Also

rawPlot slidPlot facetPlot

Examples

data("CAFE_data")
discontPlot(CAFE_data,samples=9,chromNum=17,gamma=300)

discontSmooth  A discontinuous smoother

Description

Calculates discontinuous smoother

Usage

discontSmooth(y,gamma)

Arguments

y            input vector

gamma            The gamma level can be roughly compared to the sliding window size in a normal continuous smoother. The gamma level determines how strict the algorithm functions; a higher level will correspond to fewer jumps. This cannot be larger than length(y). Must be a positive integer.

Details

Uses the potts filter algorithm described by Friedrich et al.

Value

Vector with same length as input y
Author(s)

Sander Bollen

References


Examples

```r
# generate piecewise vector with gaussian noise
y <- 1:450
y[1:150] <- 2
y[151:300] <- 3
y[301:450] <- 1
y <- y + rnorm(450)

# calculate smoother
y_smooth <- discontSmooth(y, 20)
```

Description

Plot all chromosomes horizontally next to each other

Usage

```r
facetPlot(datalist, samples = c(1, 2), slid = FALSE, combine = FALSE, k = 1, file = "default")
```

Arguments

datalist: The CAFE datalist to be analyzed, i.e. the output of `ProcessCels`.
samples: A vector or sample numbers to be plotted
slid: If TRUE, use moving average smoother
combine: If TRUE, will plot the unaltered raw data in the background
k: The sliding window size. Must be a positive integer, smaller than the length of Affy IDs on the chromosome
file: Specify a file name to store output png file

Value

Plot to file system. Return a ggplot2 graph
Note

Makes heavy use of the ggplot2 package

Author(s)

Sander Bollen

References


See Also

slidPlot rawPlot discontPlot

Examples

```r
data("CAFE_data")
facetPlot(CAFE_data,samples=9)
```

---

**fisher.method**  
*Combines p-values by using Fisher’s method*

### Description

Combines p-values by using Fisher’s method

### Usage

`fisher.method(pvals)`

### Arguments

- `pvals`  
  Vector of p values

### Value

Combined p value

### Author(s)

Sander Bollen

### Examples

```r
pvals <- runif(20)  
# generate 20 pvals
fisher.method(pvals)
```
guiSubset  Subset data with a GUI

Description
Provides graphical user interface for subsetting input datasets

Usage
guiSubset(datalist,alternative)

Arguments
datalist the dataset to be subsetted
alternative "greater" or "less"

Value
Subset of input to variable guiSelectedSet in working directory

Author(s)
Sander Bollen

See Also
cliSubset

Examples
## Not run:
data("CAFE_data")
guiSubset(CAFE_data,alternative="greater")
## End(Not run)

ProcessCels Processing CEL files

Description
Normalizes and computes relative expressions for all CEL files in work directory

Usage
ProcessCels(threshold.over=1.5,threshold.under=(2/3),remove_method=1,
local_file=NULL)
Arguments

threshold.over  Determines the threshold, as a multiple of median value, where probes are considered overexpressed. Default is 1.5

threshold.under Determines the threshold, as a fraction of median value, where probes are considered underexpressed. Default is 2/3

remove_method Determines which method is used to remove multiple probesets that are annotated to map to the same gene. The default option, 1, will keep 1 probeset with the following priority: 1): nnn_at; 2): nnn_a_at; 3): nnn_s_at; 4): nnn_x_at; 5): lowest nnn if multiple probes still exist. If remove_method=2, probesets will only be removed if several probesets of the same gene map to the exact same location. In the case that many probesets map to the same location, one probeset will be retained according to the priority of option 1 above. If remove_method=0, no multiple probesets will be removed

local_file Use a local - previously downloaded - UCSC file (e.g. http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/affyU133Plus2.txt.gz) instead of directly retrieving the file instead.

Details

details this function uses the RMA algorithm to normalize *.CEL files in work directory. It then computes relative expressions for every probe on every sample. Locations for probesets are downloaded from UCSC, as the standard BioConductor annotations do not map probeset location (they only map the location to the corresponding gene). Multiple probesets belonging to the same gene are removed as described above. The function then determines which probes are overexpressed and underexpressed relative to the median probeset values across all samples. Finally, the relative expressions are log2-transformed.

Value

list

$whole named list, where each element is a data.frame corresponding to a *.CEL file containing columns: 1): "ID" (Affy ID number); 2): "Sym" (gene Symbol); 3): "Value" (Expression values); 4): "LogRel" (Relative expressions); 5): "Loc" (Chromosomal locations); 6): "Chr" (Chromosome number); 7): "Band" (Cytoband); 8): "Arm" (Chromosomal arm)

$over same as $whole, but contains only those probes which are deemed overexpressed

$under same as $whole, but contains only those probes which are deemed underexpressed

Author(s)

Sander Bollen
Examples

```r
## Not run:
data <- ProcessCels()

## End(Not run)
```

---

**rawPlot**

*Plot without any smoother*

---

**Description**

Makes chromosome plot using raw data values

**Usage**

```r
rawPlot(datalist, samples=c(1,2), chromNum=1, idiogram=FALSE, file="default")
```

**Arguments**

- `datalist`: The CAFE datalist to be analyzed, i.e. the output of `ProcessCels`.
- `samples`: A vector or sample numbers to be plotted
- `chromNum`: The chromosome to be analyzed
- `idiogram`: If TRUE, will plot a chromosome idiogram over the plot
- `file`: Specify a file name to store output png file

**Value**

Plot to file system: Returns a ggplot2 graph if `chromNum!="ALL"`. When `chromNum=="ALL"`, returns a list of ggplot2 graphs.

**Author(s)**

Sander Bollen

**See Also**

- `slidPlot`  
- `facetPlot`  
- `discontPlot`

**Examples**

```r
data("CAFE_data")
rawPlot(CAFE_data, samples=8, chromNum=17)
```
slidPlot

Plot with sliding average smoother

Description
Plots chromosome plots with a moving average smoother

Usage
slidPlot(datalist, samples=c(1,2), chromNum=1, combine=FALSE, k=1, idiogram=FALSE, file="default")

Arguments
datalist The CAFE datalist to be analyzed, i.e. the output of ProcessCels.
samples A vector of sample numbers to be plotted
chromNum The chromosome to be analyzed
combine If TRUE, will plot the unaltered raw data in the background
k The sliding window size. Must be a positive integer, smaller than the total number of probesets on the chromosome
idiogram If TRUE, will plot a chromosome idiogram over the plot
file Specify a file name to store output png fileS

Value
Plot to file system; Returns a ggplot2 graph if chromNum!="ALL". When chromNum="ALL", returns a list of ggplot2 graphs.

Note
Makes heavy use of the ggplot2 package.

Author(s)
Sander Bollen

References

See Also
rawPlot facetPlot discontPlot

Examples
data("CAFE_data")
slidPlot(CAFE_data, samples=9, chromNum=17, k=50, combine=TRUE)
slidSmooth  A moving average smoother

Description

Calculates moving average smoother

Usage

slidSmooth(x,k)

Arguments

x  input vector
k  The moving average window size. Must be an integer value greater than 0, and no larger than length(y).

Value

Vector with same length as input y

Author(s)

Sander Bollen

Examples

#generate piecewise vector with gaussian noise
y <- 1:450
y[1:150] <- 2
y[151:300] <- 3
y[301:450] <- 1
y <- y + rnorm(450)

#calculate smoother
y_smooth <- slidSmooth(y,20)
Index

*Topic **datagen**
  ProcessCels, 12

*Topic **datasets**
  CAFE_data, 5

*Topic **dplot**
  discontPlot, 8
  facetPlot, 10
  rawPlot, 14
  slidPlot, 15

*Topic **hplot**
  discontPlot, 8
  facetPlot, 10
  rawPlot, 14
  slidPlot, 15

*Topic **htest**
  fisher.method, 11

*Topic **manip**
  cliSubset, 7
  guiSubset, 12

*Topic **multivariate**
  armStats, 3
  bandStats, 4
  chromosomeStats, 6

*Topic **package**
  CAFE-package, 2

*Topic **smooth**
  discontSmooth, 9
  slidSmooth, 16

armStats, 3, 5, 7
bandStats, 4, 4, 7

CAFE (CAFE-package), 2
CAFE-package, 2
CAFE_data, 5
chromosomeStats, 4, 5, 6
cliSubset, 7, 12

discontPlot, 8, 11, 14, 15