

Package ‘PFExpTools’

March 17, 2020

Type Package

Title Tools for analysis of Plasmodium falciparum gene expression data

Version 0.1.0

Imports bnstruct,
stringr

Author Gabe Foster<foster.gabe@gmail.com>

Maintainer Gabe Foster<foster.gabe@gmail.com>

Description This package contains a large number of miscellaneous functions for the processing and analysis of Plasmodium falciparum gene expression data. This functionality is largely for use on transcriptional time courses; however, there is quite a bit of functionality for single time point studies as well. This contains the necessary code to determine transcriptional staging to a reference time course, and generate cyclical regression covariates for further analysis. This package encompasses multiple workflows. One is the processing, imputation and smoothing of transcriptional time courses, and the other is the processing, staging and covariate generation for single time point transcriptional analysis.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

aliasChange	2
blastChange	2
CRCgeneration	3
curateExpdata	3
fftimpute	4
FFTphase	4
flattenAlias	5
getAverages	5
logtransform	6
lSmooth	6
nameChange	7

peakPhase	7
polyPhase	8
remLowCoverage	8
sinPhase	9
splineCourse	9
splinePhase	10
stagingByTranscription	10
swapChange	11
tcImpute	12

Index	13
--------------	-----------

aliasChange	<i>aliasChange</i>
-------------	--------------------

Description

This function performs the alias based gene renaming; called by nameChange

Usage

```
aliasChange(platdata, aliases)
```

Arguments

platdata	see nameChange
aliases	see nameChange

Value

see nameChange

blastChange	<i>blastChange</i>
-------------	--------------------

Description

This function is...elaborate. It requires blast be installed and be in your PATH. This function takes your platform data file and your annotated transcriptome file and actually creates a blast database, searches all probes against it, and identifies hits that are unique and maximal (depending on your match and secondary match parameters.) It's called from nameChange.

Usage

```
blastChange(platdata, platform, transcripts, match, secmatch)
```

Arguments

platdata	see nameChange
platform	curated data from nameChange
transcripts	see nameChange
match	see nameChange
secmatch	see nameChange

Value

see nameChange

CRCgeneration	<i>CRCgeneration</i>
---------------	----------------------

Description

CRCgeneration

Usage

```
CRCgeneration(stages)
```

Arguments

stages	This is a single column matrix; row names are sample names, and the column contains the calculated IDC location for each sample (in hpi)
--------	--

Value

This function returns a 3 column matrix; the first column contains the provided location in hpi, and the additional columns contain the calculated x and y covariates, respectively.

curateExpdata	<i>curateExpdata</i>
---------------	----------------------

Description

This function will take the name lists generated by nameChange and apply them to an expression data set. The function will process locally downloaded GEO expression files, or even just accept a GEO accession number and do the work for you.

Usage

```
curateExpdata(expname, namelist, local = T, pct = 0.75)
```

Arguments

expname	This is the name of your data set; if local is T, it is a GEO file (any type containing exp data). If local is F, this can be a GEO accession number.
namelist	This is a name list file as generated by nameChange.
local	This is a logical variable; if T, the function looks for a local file. If F, the function uses the expname as a GEO accession number and attempts to download the data.
pct	This is the cutoff for gene data; if a gene has expression values for less than this percentage of samples, it is excluded.

Value

Returns an expression frame containing genes as rows and samples as columns.

fftimpute	<i>fftimpute</i>
-----------	------------------

Description

This function imputes timepoints using the Fourier Transform; strongly recommend using another option here, as this is INCREDIBLY specific to a 2 hr timecourse over 56 hr

Usage

```
fftimpute(expdata, times)
```

Arguments

expdata	An expression matrix with genes as rows and samples as columns
times	A matrix with row names as samples and one column containing sampling times

Value

This function returns an expression matrix containing both the measured and imputed timepoints

FFTphase	<i>FFTphase</i>
----------	-----------------

Description

This function calculates peak phase for each gene in a time course using the Fourier Transform.

Usage

```
FFTphase(expdata, times)
```

Arguments

expdata	see peakPhase
times	see peakPhase

Value

see peakPhase

flattenAlias	<i>flattenAlias</i>
--------------	---------------------

Description

The alias file from PlasmoDB is unwieldy and awful. Here is a function for flattening it for easier find and replace.

Usage

```
flattenAlias(aliasfilename)
```

Arguments

aliasfilename File name of alias file obtained from PlasmoDB.

Value

Data frame with two columns; first column is old identifiers, and second column is their mapping to the most recent name (per the alias file)

getAverages	<i>GetAverages This function will get the average expression of all genes across a reference time course. Note: the reference time course should be evenly samples across a single IDC.</i>
-------------	---

Description

GetAverages This function will get the average expression of all genes across a reference time course. Note: the reference time course should be evenly samples across a single IDC.

Usage

```
getAverages(reftc)
```

Arguments

reftc This is a reference time course to average; this must be a numeric matrix with genes as rows and samples as columns.

Value

This function returns a matrix with row names as gene names and a single column containing the average expression across the reference time course.

logtransform	<i>LogTransform</i>
--------------	---------------------

Description

If your microarray data isn't log2 ratio transformed, use this. This will take your expression data and log2 ratio normalize it to a set of gene averages you provide.

Usage

```
logtransform(newdata, refavg)
```

Arguments

newdata	This is the data to be normalized; this must be a numeric matrix with rows as genes and columns as samples.
refavg	This is the list of gene averages across a time course; must be a data frame with 1 column, with row names as genes.

Value

This function returns a matrix in an identical format to newdata, with all cells being log2 ratio transformed.

lSmooth	<i>lSmooth</i>
---------	----------------

Description

This function will loess smooth a time course, first demonstrated on P. fal time courses by Bozdech et al 2003

Usage

```
lSmooth(expdata, time, smoothvalue = 0.3)
```

Arguments

expdata	Expression data matrix, with columns as samples and rows as genes
time	matrix with one column; names are samples and column contains sample time
smoothvalue	loess smoothing value; default is 0.3

nameChange	<i>nameChange</i>
------------	-------------------

Description

This function takes a platform file from GEO (a cut and paste from the full table view, really), and identifies the new genome assembly name for each probe using one of a few methods. This function is a wrapper for subfunctions that carry out each method.

Usage

```
nameChange(
  platform,
  method = "swap",
  aliases = NA,
  transcripts = NA,
  match = 130,
  secmatch = 60
)
```

Arguments

platform	This is the filename for the platform from GEO. A simple cut/paste in to a txt file will work here (it's tab delimited by default).
method	There are 3 methods here; "swap" just takes the new gene name from the platform file (least accurate), "alias" will find new names based on the current PlasmoDB assembly file (not bad), "blast" will take a provided Annotated Transcripts file from PlasmoDB, blast each probe against it, and only keep probes that meet the bitscore criteria (default is 130 for perfect hit and 60 for minimal secondary hits).
aliases	This is the flat alias file created by the function "flattenAlias"; only necessary if selected method is "alias".
transcripts	The fasta file containing the annotated transcripts; this can be obtained from PlasmoDB.
match	The bitscore for a perfect primary match; only necessary if "blast" is selected, default is 130.
secmatch	The maximum bitscore for secondary probe alignments; only necessary if "blast" is selected, default is 60.

peakPhase	<i>peakPhase</i>
-----------	------------------

Description

peakPhase

Usage

```
peakPhase(expdata, times, method = "poly")
```

Arguments

expdata	This must be an expression matrix, with rows as genes and columns as samples. Sample names (column names) must match sample names in the "times" parameter.
times	This must be a matrix with one column; the column is actual sampling time, and the row names are the sample names.
method	This allows the user to choose the method for peak time determination; current options are "poly"(default), "sin", "spline", and "FFT". Note, FFT will only work on evenly sampled time courses (or evenly imputed time courses).

Value

This function returns a data frame with a single column; the row names are the gene names, and the column contains the time at which the gene is calculated to express maximally.

polyPhase	<i>polyPhase</i>
-----------	------------------

Description

This is a sub function that calculates peak expression by 6th order polynomial fit. Accessed through peakPhase.

Usage

```
polyPhase(expdata, times)
```

Arguments

expdata	see peakPhase
times	see peakPhase

Value

see peakPhase

remLowCoverage	<i>remLowCoverage</i>
----------------	-----------------------

Description

Removes any gene from the expression dataset that does not have values in a minimum percentage of samples.

Usage

```
remLowCoverage(expdata, mincov, faillist = F)
```


Arguments

expdata	This is a matrix containing expression values' rows must be genes and columns must be samples.
mincov	This is the minimum percent coverage; probes must appear in at least this percentage of samples to pass QC.
faillist	Default is FALSE; if TRUE, function returns a list containing both the curated expression matrix and a list of genes that failed QC.

Value

This function returns an expression matrix with the genes that do not pass QC removed.

sinPhase	<i>sinPhase</i>
----------	-----------------

Description

This is a sub function that calculates peak expression by sin fitting. Accessed through peakPhase.

Usage

```
sinPhase(expdata, times)
```

Arguments

expdata	see peakPhase
times	see peakPhase

Value

see peakPhase

splineCourse	<i>splineCourse</i>
--------------	---------------------

Description

This function takes an existing time course and uses spline interpolation to create an imputed time course with the desired resolution.

Usage

```
splineCourse(expdata, times, fulltimes = 1:48)
```

Arguments

expdata	Actual expression data; a matrix with genes as rows and time course samples as columns. In order from earliest to latest.
times	The actual sample times for the expression matrix.
fulltimes	A vector containing the times desired; for example, an hourly time course over a single IDC would be 1:48.

Value

Returns an expression matrix expanded with additional columns to contain samples at the desired times.

splinePhase	<i>splinePhase</i>
-------------	--------------------

Description

This function fits a spline to the expression data and determines the time at which each gene is maximally expressed. Called by peakPhase.

Usage

```
splinePhase(expdata, times)
```

Arguments

expdata	see peakPhase
times	see peakPhase

Value

see peakPhase

stagingByTranscription	<i>stagingByTranscription</i>
------------------------	-------------------------------

Description

This function compares each whole transcriptome sample to every time point in the provided reference time course, and identifies the best time point fit. Correlation is determined by the Pearson method.

Usage

```
stagingByTranscription(
  samples,
  reference,
  subset = NA,
  cortable = F,
  heatmap = T,
  start = 1
)
```

Arguments

samples	A matrix of expression samples, with rows as genes and columns as samples. This is the set that will be tested.
reference	A reference expression time course, with rows as genes and columns as samples.
subset	The default is to use all genes in common between the two sets; if a subset is desired, provide this as vector of gene names in the subset.
cortable	If the full matrix of correlation values is desired, set to TRUE; default is FALSE.
heatmap	A simple correlation heatmap will be generated as default. Set to false to skip.
start	What hour the reference course starts at- default is 1hpi

Value

Default return is a matrix with one column; rows are samples and values are the time point of maximum correlation. The function will also plot a simple heatmap by default. If cortable is set to TRUE, the function will return a list; the first entry is the simple time point matrix, and the second entry is the full correlation matrix.

swapChange

swapChange

Description

This function simply pulls the new ORF and probe ID and creates a renaming file from that; I swear this wasn't there before, but now that we have the new GeneIDs listed might as well build a function to rename based on them. This function is called by nameChange.

Usage

```
swapChange(platdata)
```

Arguments

platdata	see nameChange
----------	----------------

Value

see nameChange

`tcImpute`*tcImpute*

Description

This function calls the bnstruct package "knn.impute" function to perform k nearest neighbor imputation on a time course to fill in missing data.

Usage

```
tcImpute(expdata, neighbors)
```

Arguments

<code>expdata</code>	This is an expression matrix with rows as genes and columns as samples in a time course. Time points missing data should be inserted as columns in the correct place containing NAs.
<code>neighbors</code>	This is the number of neighbors used in determining value.

Value

This function returns the expression matrix with blank columns filled with imputed values.

Index

aliasChange, [2](#)

blastChange, [2](#)

CRCgeneration, [3](#)
curateExpdata, [3](#)

fftimpute, [4](#)
FFTphase, [4](#)
flattenAlias, [5](#)

getAverages, [5](#)

logtransform, [6](#)
lSmooth, [6](#)

nameChange, [7](#)

peakPhase, [7](#)
polyPhase, [8](#)

remLowCoverage, [8](#)

sinPhase, [9](#)
splineCourse, [9](#)
splinePhase, [10](#)
stagingByTranscription, [10](#)
swapChange, [11](#)

tcImpute, [12](#)