compartment model

July 6, 2022

Type Package

Title Evaluate kinetic parameters which explain RNA Pol II density in nascent transcription data

Version 0.1.0

Author Michael Guertin, Rudradeep Mukherjee

Maintainer Rudradeep Mukherjee <rmukherjee@uchc.edu>, Michael Guertin <guertin@uchc.edu>

Description Implements compartment model to describe RNA Pol II density as described in Sathyan et. al. 2019, DOI: 10.1101/gad.328237.119.

License NA

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

Imports NMF,
pracma,
RColorBrewer,
bigWig,
primaryTranscriptAnnotation
dplyr,
deSolve,
pksensi,
lattice,
zoo,
parallel,
stringr

R topics documented:

`change.two.parameters` *Takes sets of two params to be changed and two param sets to be held constant, and selects all the param sets (being changed) which satisfy the target pause and body levels*

Description

Takes sets of two params to be changed and two param sets to be held constant, and selects all the param sets (being changed) which satisfy the target pause and body levels

Usage

```
change.two.parameters(
  func.density = density.prime.kinit.krel,
  p.change = pause.change,
  b.change = body.change,
  baseline.pause = out.pause,
  baseline.body = out.body,
  param1 = kinit.vec,
  param2 = krel.vec,
  params = c("kinit", "krel"),
  constant.param1.vec = kpre.vec,
  constant.param2.vec = kelong.vec,
  perc = 0.05,
  fold.query = 5,
  sample.size = 10000,
  ...
)
```

Arguments

func.density	a function which defines the compartment model and accomodates two of the param sets held constant and two params which are changed.
p.change	change in pause density
b.change	change in body density
baseline.pause	a vector containing baseline pause densities
baseline.body	a vector containing baseline body densities
param1	a vector containing individual realisations of first param to be changed
param2	a vector containing individual realisations of second param to be changed
params	a string vector naming the two params being changed
constant.param1.vec	a vector containing realisations of the first param to be kept constant
constant.param2.vec	a vector containing realisations of the second param to be kept constant
perc	percent threshold for matching targeted and realized values
fold.query	the params to be changed will be sweeped from [1 / fold.query, fold.query] * original param vals
sample.size	the number of samples randomly choosen from the sample space

Value

a list containing two sets of modified parameters, i.e; param1 and param2 sets which satisify change in pause region (baseline.pause*p.change) and gene body (baseline.body*b.change) in the perc threshold provided

Examples

```
#this function can be called separately. It is invoked within parallel.change.two.param.helper() where details
change.two.parameters(func.density = density.prime.kpre.kelong,
                                p.change = 1.15,
                                b.change = 0.95,
                                baseline.pause = out.pause,
                                baseline.body = out.body,
                                param1 = kpre.vec,
                                param2 = kelong.vec,
                                params = c("kpre", "kelong"),
                                constant.param1.vec = kinit.vec,
                                constant.param2.vec = krel.vec,
                                perc = 0.05, fold.query = 3, sample.size = 10000)
```

`density.prime` *Compartment model adapted from our G&D paper (doi:10.1101/gad.328237.119): Declare differential equations as function dP/dt = kinit - (kpre +krel)p dB/dt = krel * p - kelong * b P is the first dependent variable, promoter density; dP is its derivative wrt time B is the second dependent variable, body density; dB is its derivative wrt time*

Description

Compartment model adapted from our G&D paper (doi:10.1101/gad.328237.119): Declare differential equations as function dP/dt = kinit - (kpre +krel)p dB/dt = krel * p - kelong * b P is the first dependent variable, promoter density; dP is its derivative wrt time B is the second dependent variable, body density; dB is its derivative wrt time

Usage

```
density.prime(t, initial.state, params = params)
```

Arguments

t	time points to run the ODE model
initial.state	a list containing initial values of the variables being evolved.
params	list of other variables (rates) being used in the model.

Value

this function is called within ode function, and returns pause body with values in three body compartments.

Examples

```
library(deSolve)
initial.state = c(P = 1, B1 = 0.01)
parms = c(kinit = 0.1, krel = 0.1, kpre = 0.1, kelong = 50)
t <- seq(from = 0.01, to = 100.01, by = 10)
y <- ode(y = initital.state, times = t, func = density.prime, parms = parms)
```

---

density.prime.kinit.kelong

an implementation of compartment model where kinit and kelong are being modified, and kpre and krel are held constant. (To be passed to ode() solver in package deSolve().)

Description

an implementation of compartment model where kinit and kelong are being modified, and kpre and krel are held constant. (To be passed to ode() solver in package deSolve().)

Usage

```
density.prime.kinit.kelong(
  t,
  initial.state,
  params = params,
  constant.param1,
  constant.param2
)
```

Arguments

t	series of time points where this function is used to evaluate the change in variables
initial.state	the values of variables being used to start the model evolution
params	containing values of rates being changed, i.e; kinit and kelong
constant.param1	the first param not being modified, here kpre
constant.param2	the second param not being modified, here krel

Value

two values corresponding to pause and body densities

---

density.prime.kinit.kpre

an implementation of compartment model where kinit and kpre are being modified, and krel and kelong are held constant. (To be passed to ode() solver in package deSolve().)

Description

an implementation of compartment model where kinit and kpre are being modified, and krel and kelong are held constant. (To be passed to ode() solver in package deSolve().)

Usage

```
density.prime.kinit.kpre(
  t,
  initial.state,
  params = params,
  constant.param1,
  constant.param2
)
```

Arguments

t	series of time points where this function is used to evaluate the change in variables
initial.state	the values of variables being used to start the model evolution
params	containing values of rates being changed, i.e; kinit and kpre
constant.param1	the first param not being modified, here krel
constant.param2	the second param not being modified, here kelong

Value

two values corresponding to pause and body densities

`density.prime.kinit.krel` *an implementation of compartment model where kinit and krel are being modified, and kpre and kelong are held constant. (To be passed to ode() solver in package deSolve().)*

Description

an implementation of compartment model where kinit and krel are being modified, and kpre and kelong are held constant. (To be passed to ode() solver in package deSolve().)

Usage

```
density.prime.kinit.krel(
  t,
  initial.state,
  params = params,
  constant.param1,
  constant.param2
)
```

Arguments

`t`	series of time points where this function is used to evaluate the change in variables
`initial.state`	the values of variables being used to start the model evolution
`params`	containing values of rates being changed, i.e; kinit and krel
`constant.param1`	the first param not being modified, here kpre
`constant.param2`	the second param not being modified, here kelong

Value

two values corresponding to pause and body densities

`density.prime.kpre.kelong` *an implementation of compartment model where kpre and kelong are being modified, and kinit and krel are held constant. (To be passed to ode() solver in package deSolve().)*

Description

an implementation of compartment model where kpre and kelong are being modified, and kinit and krel are held constant. (To be passed to ode() solver in package deSolve().)

Usage

```
density.prime.kpre.kelong(
  t,
  initial.state,
  params = params,
  constant.param1,
  constant.param2
)
```

Arguments

t	series of time points where this function is used to evaluate the change in variables
initial.state	the values of variables being used to start the model evolution
params	containing values of rates being changed, i.e; kinit and krel
constant.param1	the first param not being modified, here kinit
constant.param2	the second param not being modified, here krel

Value

two values corresponding to pause and body densities

`density.prime.kpre.krel` *an implementation of compartment model where kpre and krel are being modified, and kinit and kelong are held constant. (To be passed to ode() solver in package deSolve().)*

Description

an implementation of compartment model where kpre and krel are being modified, and kinit and kelong are held constant. (To be passed to ode() solver in package deSolve().)

Usage

```
density.prime.kpre.krel(
  t,
  initial.state,
  params = params,
  constant.param1,
  constant.param2
)
```

Arguments

t	series of time points where this function is used to evaluate the change in variables
initial.state	the values of variables being used to start the model evolution
params	containing values of rates being changed, i.e; kpre and krel
constant.param1	the first param not being modified, here kinit
constant.param2	the second param not being modified, here kelong

Value

two values corresponding to pause and body densities

---

density.prime.krel.kelong

an implementation of compartment model where krel and kelong are being modified, and kinit and kpre are held constant. (To be passed to ode() solver in package deSolve().)

Description

an implementation of compartment model where krel and kelong are being modified, and kinit and kpre are held constant. (To be passed to ode() solver in package deSolve().)

Usage

```
density.prime.krel.kelong(
  t,
  initial.state,
  params = params,
  constant.param1,
  constant.param2
)
```

Arguments

t	series of time points where this function is used to evaluate the change in variables
initial.state	the values of variables being used to start the model evolution
params	containing values of rates being changed, i.e; krel and kelong
constant.param1	the first param not being modified, here kinit
constant.param2	the second param not being modified, here kpre

Value

two values corresponding to pause and body densities

`dynamic.pro.profile` *uses [find.body.param()] and [get.pro.waveform()] to generate PRO-seq waveform based on pause and gene body values provided in the input.*

Description

uses [find.body.param()] and [get.pro.waveform()] to generate PRO-seq waveform based on pause and gene body values provided in the input.

Usage

```
dynamic.pro.profile(
  input,
  tau = 20,
  min.pk = 0,
  max.pk = 1,
  dpk = 0.001,
  gene.len = 1000,
  filename = "dynamic_pro_model_density"
)
```

Arguments

`input`	a data farme containing Body and Pause columns with corresponding values
`tau`	a parameter which acts as a "time constant" for the exponential function fitting in find.body.param
`min.pk`	minimum peak value - to be passed to [find.body.param()]
`dpk`	threshold for implicit solution - to be passed to [find.body.param()]
`filename`	name of the file to which the plot is saved
`max.px`	maximum peak value - to be passed to [find.body.param()]
`gen.len`	length of gene i.e; x-range for the desired PRO-seq profile

`find.body.param` *code adapted from our G&D paper (doi:10.1101/gad.328237.119): to visualize it in a composite profile form function for finding the gene body parameter*

Description

code adapted from our G&D paper (doi:10.1101/gad.328237.119): to visualize it in a composite profile form function for finding the gene body parameter

Usage

```
find.body.param(
  bpeak = NULL,
  tau = NULL,
  pausepeak = NULL,
  min.pk = 0,
  max.pk = 1,
  dpk = 0.001
)
```

Arguments

bpeak	desired gene body level
tau	exponential decay constant
pausepeak	desired paused region level
min.pk	minimal level for gene body peak
max.pk	maximal level for gene body peak
dpk	resolution for the implicit solution

Value

the desired body parameter

Examples

```
body.param <- find.body.param(bpeak=0.08, tau=20, pausepeak=0.25, min.pk=0, max.pk=1, dpk=.001)
```

---

find.pause.regions *Get start, end coordinates of pause window and gene body for each gene using the bigWig files representing all conditions x replicates. The start and end coordinates are treated as transcription start site (TSS) and transcription termination site (TTS) respectively. The pause window has a 50bp size around the peak signal. The gene body starts from pause window end and ends at TTS.*

Description

Get start, end coordinates of pause window and gene body for each gene using the bigWig files representing all conditions x replicates. The start and end coordinates are treated as transcription start site (TSS) and transcription termination site (TTS) respectively. The pause window has a 50bp size around the peak signal. The gene body starts from pause window end and ends at TTS.

Usage

```
find.pause.regions(bed.input, combined.plus.bw, combined.minus.bw)
```

Arguments

bed.input	A bed6 file containing genes of interest, where start and end are transcription start site and transciption termination site respectively.
combined.plus.bw	bigWig file for the plus strand data
combined.minus.bw	bigWig file for the minus strand data

Value

A bed6 file containing start, end coordinates of pause window (50bp in size), gene body () starting from seventh column.

Examples

```
master.pTA.coords.file = 'mastercoords.bed' # bed file created from primary Transcript Annotation.
pTA <- read.table(master.pTA.coords.file)
df.pause.body <- find.pause.regions(pTA, bw.plus, bw.minus) # bw.* files are corresponding bigWig files.
```

---

generate.composite.df *creates composite signal starting from -upstream to +downstream based upon bigWig files passed per condtion.*

Description

creates composite signal starting from -upstream to +downstream based upon bigWig files passed per condtion.

Usage

```
generate.composite.df(
  df.input,
  cond1.plus.bw,
  cond1.minus.bw,
  cond1 = "cond1",
  cond2.plus.bw,
  cond2.minus.bw,
  cond2 = "cond2",
  upstream = 350,
  downstream = 1400,
  roll.avg = 10,
  step = 5
)
```

Arguments

df.input	a bed6 input file - simular to output of [find.pause.regions()] - containing pause.start and pause.end coordinates at seventh and eighth columns.
cond1.plus.bw	bigWig file for plus strand data in cond1 (baseline).
cond1.minus.bw	bigWig file for plus strand data in cond1 (baseline).

cond1	name of cond1 (baseline) to be used as to fill cell values under ‘cond‘ column in the returned object,
cond2.plus.bw	bigWig file for plus strand data in cond2 (cond2 being the condition of interest being compared against baseline)
cond2.minus.bw	bigWig file for minus strand data in cond2 (cond2 being the condition of interest being compared against baseline)
cond2	name of the condition which is being compared against baseline
upstream	the number of bases upstream from pause summit (default 350)
downstream	the number of bases downstream from pause summit (default 1400)
step	a factor used to define the ends of genes by (step * roll.avg)/2
roll, avg	the rolling size used in rollmean() to calculate the composites (default 10)

`get.pro.waveform` *function to get the PRO waveform*

Description

function to get the PRO waveform

Usage

```
get.pro.waveform(
  bpeak = NULL,
  pausepeak = NULL,
  bodypeak = NULL,
  bp.seq = NULL,
  tau = NULL
)
```

Arguments

bpeak	desired gene body level
pausepeak	desired pause region level
bodypeak	body peak value to obtain a level of bpeak
bp.seq	base pair sequence
tau	exponential decay constant

Value

a list containing a vector with simulated PRO-seq singal and an object containing the simulated and requested signal

Examples

```
x <- get.pro.waveform(pausepeak=0.25, bpeak=0.02, bp.seq=seq(0,999),tau=20)
```

`merge.pbody.deseq2` *a helper function to merge two data frames on gene names. ASSUMES that both data frames have rownames set to gene names.*

Description

a helper function to merge two data frames on gene names. ASSUMES that both data frames have rownames set to gene names.

Usage

```
## S3 method for class 'pbody.deseq2'
merge(df.pause.body, deseq2.df)
```

Arguments

`df.pause.body`	a data frame containing start and end coordinate for pause windows and gene body. Ideally, an output of [find.pause.regions()]
`deseq2.df`	a data object created from running DESeq2 workflow and containing response status of genes.

Value

an object merged on gene names having columns in order of df.pause.body followed by deseq2.df

---

`parallel.change.two.param.helper` *a general function to call [change.two.parameters()] given the condition name and the objects associated with scanned parameters*

Description

a general function to call [change.two.parameters()] given the condition name and the objects associated with scanned parameters

Usage

```
## S3 method for class 'change.two.param.helper'
parallel(
  cond1 = "cond1",
  cond2 = "cond2",
  direction = "activated",
  output.param.scan,
  x.fast99.obj,
  perc = 0.02,
  pause.index.baseline,
  pause.sum.change,
  body.change,
  foldquery = 2,
  samplesize = 1000
)
```

Arguments

cond1	name of baseline conditon
cond2	name of condition which is being compared with the baseline
direction	nature of the change in the data being compared between condition of interest and the baseline. Say, "activated"
output.param.scan	an object containing the model output from param scans.
x.fast99.obj	an object created from using rfast99() function from pksensi package
perc	percentage threshold used while matching simulated values with the target values
pause.index.baseline	the pause index in the baseline condition
pause.sum.change	the change in pause sum between conditon of interest and baseline
body.change	the change in body signal between condition of interest and baseline
foldquery	the param scan will be done with values ranging between [1/foldquery, foldquery] * original param values
samplesize	the number of samples randomly selected from the param space defined by foldquery.

Examples

```
# see paramFittingCompartmentModel.R for a description of calling this function
```

---

plot.bw.plots *creates a plot containing box and whisker plot, violin plot and scatter plot of the change in pause indices between a condition of interest and a baseline.*

Description

creates a plot containing box and whisker plot, violin plot and scatter plot of the change in pause indices between a condition of interest and a baseline.

Usage

```
## S3 method for class 'bw.plots'
plot(input.df, cond1 = "cond1", cond2 = "cond2", color = "lightblue")
```

Arguments

input.df	an object containing pause indices for each gene of interest (oer condition,) Ideally, an output of [total.region.density()]
cond1	name of the baseline condition
cond2	nqme of the condition which is being compared against baseline
color	color to be used in the violin plot

`plot.changes.wrt` *plot gene body and pause region density based on the input*

Description

plot gene body and pause region density based on the input

Usage

```
## S3 method for class 'changes.wrt'
plot(input, filename = "dynamic_pro_model")
```

Arguments

`input`	contains iteration, body and pause columns for values for plotting
`filename`	name of the file to which the plot is saved

`plot.composites` *uses output of [generate.composite.df()] to generate plot of composite signal.*

Description

uses output of [generate.composite.df()] to generate plot of composite signal.

Usage

```
## S3 method for class 'composites'
plot(
  dat,
  fact = "GR",
  comp = "20.v.0",
  summit = "TSS",
  y.low = 0,
  y.high = NULL,
 col.lines = c(rgb(0, 0, 1, 1/2), rgb(1, 0, 0, 1/2), rgb(0.1, 0.5, 0.05, 1/2), rgb(0,
   0, 0, 1/2), rgb(1/2, 0, 1/2, 1/2), rgb(0, 1/2, 1/2, 1/2), rgb(1/2, 1/2, 0, 1/2))
)
```

Arguments

`dat`	output of [generate.composite.df()]
`fact`	name of the factor in the condition being compared against
`comp`	a small string to describe the condition being made
`summit`	a string which will be used as a xlabel - distance from <summit>
`y.low`	the lower limit of the y axis
`y.high`	the max limit of the y axis. If not provided, it estimates it using the maximum estimate present in input data
`col.lines`	colors to be used for drawing the composites

Value

NULl

`plot.pro.simulation.composites`
create a plot of (simulated) composites given input containing y-values against x-values

---

Description

create a plot of (simulated) composites given input containing y-values against x-values

Usage

```
## S3 method for class 'pro.simulation.composites'
plot(input, filename = "dynamic_pro_model")
```

Arguments

`input`	a data frame with three columns in this order - index, y.val and x.val. y.val is plotted against x.val within the function
`filename`	this name is used to save the plot

`plot.pro.simulation.composites.2`
plots PRO-seq waveform of two conditions present in the input.

Description

plots PRO-seq waveform of two conditions present in the input.

Usage

```
## S3 method for class 'pro.simulation.composites.2'
plot(
  input,
  pause.offset = 0,
  filename = "dynamic_pro_model",
  ylim = c(0, 0.0082),
  trace.col = c("grey50", "#6aa3ce")
)
```

Arguments

input	a dataframe where first column is condition (group), second column is y.val, third column is x.val.
pause.offset	an offset to place vertical guide lines at [0 - pause.offset, 50 - pause.offset]
filename	name of the file to which the plot will be saved
ylim	two values to set limits of the y.axis
trace.col	two set of colors used for plotting PRO-seq signal. Color indices will be mapped in alphabetical order of condition names.

`plot.simulated.composites.helper`

a helper function to plot PRO-seq waveform derived after simulations. Used in the script plot_simulation_composites.R. Uses data from changing knit and krel and arranges code present from page 40 of compartment modeling vignette.

Description

a helper function to plot PRO-seq waveform derived after simulations. Used in the script plot_simulation_composites.R. Uses data from changing knit and krel and arranges code present from page 40 of compartment modeling vignette.

Usage

```
## S3 method for class 'simulated.composites.helper'
plot(
  input.data.identifier = "activated.condnvsbaseline",
  output.plot.name = "model_condnvsbaseline_composite",
  baseline.condn.name = "control",
  condn.name = "condn",
  seq.end = 0.1,
  by.step = 1e-04,
  tau = 10,
  dpk = 1e-05,
  dataFolder = "../data"
)
```

Arguments

input.data.identifier	a string which is used to identify data objects that are read. Ideally, the data objects were generated using [parallel.change.two.param.helper()] called in paramFittingCompartmentModel.R
output.plot.name	name of the file used for saving
condn.name	name of the condition which is being compared against baseline
seq.end	end of the sequence or range of the x-axis
by.step	step size of going from 0 to seq.end

tau	a "time-constant" like parameter passed to [find.body.param()]
dpk	threshold for the implicit solution
baseline.cond.name	name of the baseline condition

`plot.two.parameter.bw` *plots fold change for new parameters (obtained from param scan) w.r.t to original parameters.*

Description

plots fold change for new parameters (obtained from param scan) w.r.t to original parameters.

Usage

```
## S3 method for class 'two.parameter.bw'
plot(factor.param1.param2, factor = "GR", y.lab, y.lim = NULL)
```

Arguments

factor.param1.param2	an object created from [two.parameter.bw.plot.lattice()] containing three columns fold.change, (modified) rates and description of constant params.
factor	name of the factor/experiment which will be used in the filnename as <factor>_change_two_parameters.pdf
y.lab	label for y-axis
y.lim	two values giving the lower and upper limit of y-axis. If not provided, the upper limit is calculated using the data and lower limit is set to zero.

`pro.integrated.peak` *takes an input dataframe with Pause and Body signals and returns a PRO-seq waveform with the x-coordinate.*

Description

takes an input dataframe with Pause and Body signals and returns a PRO-seq waveform with the x-coordinate.

Usage

```
pro.integrated.peak(
  input,
  tau = 20,
  min.pk = 0,
  max.pk = 1,
  dpk = 0.001,
  gene.len = 2000,
  pause.height,
  time.char = "0 min"
)
```

Arguments

input	a dataframe containing iteration, pause and body signal
tau	a "time constant" parameter passed to [find.body.param()]
min.pk	minimum of the peak value - to be passed to [find.body.param()]
max.pk	maximum of the peak value - to be passed to [find.body.param()]
dpk	threshold for implicit solution - to be passed to [find.body.param()]
gene.len	range of the x-axis to be used for plotting.
pause.height	height of the pause summit.
time.char	character which describes the condition

Value

a dataframe with columns iteration, PRO-seq signal, x-coordinate

---

run.plotting.steps *run the plotting steps given a pause.body object (with desired genes) and pair of bigWigs from two conditions being compared*

Description

run the plotting steps given a pause.body object (with desired genes) and pair of bigWigs from two conditions being compared

Usage

```
run.plotting.steps(
  df.pausebody,
  cond1.name,
  bw.cond1.plus,
  bw.cond1.minus,
  cond2.name,
  bw.cond2.plus,
  bw.cond2.minus,
  factor.name = "efferocytosis",
  color.names = c(rgb(0, 0, 1, 1/2), rgb(1, 0, 0, 1/2))
)
```

Arguments

df.pausebody	an object containing start, end coordinates of pause window and gene body.
cond1.name	name of baseline condition
bw.cond1.plus	bigWig file containing plus strand data from baseline condition
cond2.name	name of the condition which is being compared against the baseline condition
bw.cond2.plus	bigWig file containing plus strand data from the condition which is being compared against baseline condition
bw.cond2.minus	bigWig file containing minus strand data from the condition which is being compared against baseline condition'

factor.name	descriptive name of the factor involved in the comparison of condition to baseline
color.names	to be used as colors for plotting. The order of colors chosen will be same as the alphabetical order of names of two conditions
bw, cond1.minus	bigWig file containing minus strand data from baseline condition

---

`total.region.density` *appends "pause sum" and "body average" columns to a bed6 object created by [find.pause.regions()]. It also appends "pause index" column which is the ratio of "pause sum" and "body average".*

Description

appends "pause sum" and "body average" columns to a bed6 object created by [find.pause.regions()]. It also appends "pause index" column which is the ratio of "pause sum" and "body average".

Usage

```
total.region.density(
  df.input,
  cond1.plus.bw,
  cond1.minus.bw,
  cond1 = "cond1",
  cond2.plus.bw,
  cond2.minus.bw,
  cond2 = "cond2"
)
```

Arguments

df.input	bed6 file created by [find.pause.regions()] containing the start and end of pause window and gene body for genes of interest.
cond1.plus.bw	bigWig file containing data from plus strand in cond1 ("cond1" is baseline.)
cond1.minus.bw	bigWig file containing data from minus strand in cond1 ("cond1" is baseline.)
cond1	name of the condition (baseline)
cond2.plus.bw	bigWig file containing data from plus strand in cond2 ("cond1" is the condition of interest being compared against the baseline.)
cond2.minus.bw	bigWig file containing data from minus strand in cond2 ("cond1" is the condition of interest being compared against the baseline.)
cond2	name of the condition of interest

Value

the input bed6 object appended with columns containing pause sum, body average, pause index for each gene present in the input file.

Examples

```
see documentation of [find.pause.regions()] to see how bed.object was created.
df.input = find.pause.regions(bed.object, bw.plus, bw.minus)
df.region.density <- total.region.density(df.input, cond1.plus.bw, cond1.minus.bw, "cond1",
                                          cond2.plus.bw, cond2.minus.bw, "cond2")
```

`two.parameter.bw.plot.lattice` *calculates fold change between given param list and original values and returns a data frame*

Description

calculates fold change between given param list and original values and returns a data frame

Usage

```
two.parameter.bw.plot.lattice(
  param1.param2.lists,
  param1.vec,
  param2.vec,
  params = c("Kinit", "Krel"),
  constant.params = c("Kelong", "Kpre")
)
```

Arguments

`param1.vec`	original values of param1
`param2.vec`	original values of param2
`params`	names of params which were modified i.e; the identity of param1, param2
`constant.params`	names of parameters which were held constant
`param1.param2.list`	a list containing the derived param values from running [change.two.parameters()]

Value

data frame containing three columns - fold.change, (modified) rates and description of constant params. The description in 3rd column is constant_<param1>_<param2>

Examples

```
#the function was taken from page 28 of compartment modeling vignette
#see ./callers/plotParamSets.R for how these functions were called. The "two param" lists were created by paral
```

Index