

Package ‘RNAdecay’

April 15, 2020

Date 2018-01-22

Title Maximum Likelihood Decay Modeling of RNA Degradation Data

Version 1.6.0

Description RNA degradation is monitored through measurement of RNA abundance after inhibiting RNA synthesis. This package has functions and example scripts to facilitate (1) data normalization, (2) data modeling using constant decay rate or time-dependent decay rate models, (3) the evaluation of treatment or genotype effects, and (4) plotting of the data and models. Data Normalization: functions and scripts make easy the normalization to the initial (T0) RNA abundance, as well as a method to correct for artificial inflation of Reads per Million (RPM) abundance in global assessments as the total size of the RNA pool decreases. Modeling: Normalized data is then modeled using maximum likelihood to fit parameters. For making treatment or genotype comparisons (up to four), the modeling step models all possible treatment effects on each gene by repeating the modeling with constraints on the model parameters (i.e., the decay rate of treatments A and B are modeled once with them being equal and again allowing them to both vary independently). Model Selection: The AICc value is calculated for each model, and the model with the lowest AICc is chosen. Modeling results of selected models are then compiled into a single data frame. Graphical Plotting: a function is provided to easily visualize the data and the selected model using ggplot2 package functions.

Depends R (>= 3.5)

Imports stats, grDevices, grid, ggplot2, gplots, utils, TMB, nloptr

Suggests parallel, knitr, reshape2, rmarkdown

biocViews ImmunoOncology, Software, GeneExpression, GeneRegulation, DifferentialExpression, Transcription, Transcriptomics, TimeCourse, Regression, RNASeq, Normalization, WorkflowStep

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1.9000

VignetteBuilder knitr

git_url <https://git.bioconductor.org/packages/RNAdecay>

git_branch RELEASE_3_10

git_last_commit 32e147e

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

Author Reed Sorenson [aut, cre],
 Katrina Johnson [aut],
 Frederick Adler [aut],
 Leslie Sieburth [aut]

Maintainer Reed Sorenson <reedssorenson@gmail.com>

R topics documented:

a_high	2
cols	3
const_decay	4
decay_data	4
decay_plot	5
groupings	6
group_map	7
models	8
mod_optimization	9
plain_theme	10
results	11
RPMs	12

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

a_high	<i>calculates bounds for modeled parameters</i>
--------	---

Description

Calculates maximum and minimum bounds for parameter alpha based on experimental time points ($t_0, t_1, t_2, t_3, \dots, t_{\max}$). If RNA level is too low at t_1 , then the decay has happened before our observations began - there is an upper bound to the decay rate we can detect (`a_high`). If RNA level is too high at t_{\max} , then relatively little decay has happened and we can not distinguish the decay rate and the decay of the decay rate - there is a lower bound to the base decay rate of the decaying decay model (`a_low`).

Usage

```
a_high(t_min)
```

```
a_low(t_max)
```

```
b_low(t_max)
```

Arguments

t_min time of first experiemtal time point after inhibition of transcription (not T0)

t_max time of last experimental time point

Details

Similarly, limits on beta are required to prevent precude ranges in which the decay rate and decaing decay are indistinguishable. See vignette "RNAdecay_workflow" for more information.

Value

returns the lowest/highest parameter values to be used as bounds on modeled parameters

Examples

```
a_high(7.5)
a_low(480)
b_low(480)
```

cols

Indexes column names with multiple patterns (i.e., multigrep)

Description

Identifies dataframe column names that have all of the pattern arguments (up to 4).

Usage

```
cols(df, w, x = NA, y = NA, z = NA)
```

Arguments

df	a dataframe with column names to parse
w, x, y, z	character string or regular expression passed to grep pattern argument

Details

Be aware that column data labels that are part of another data label are not advisable (e.g. mut1, mut2, mut1.mut2; cols(df,'mut1') will return indices for both 'mut1' and 'mut1.mut2' labeled columns

Value

returns a vector of integer indices of the column names of df that match to all of w, x, y, and z

Examples

```
cols(data.frame(xyz=1:5,zay=6:10,ybz=11:15,tuv=16:20),'y','z') ## returns 1 2 3
```

const_decay *exponential decay functions*

Description

Constant decay rate function (const_decay(), case when betas=0) e^{-a*t} ; decaying decay rate function (decaying_decay()) $e^{-(a/b)*(1-e^{-b*t})}$. Functions are normalized so at $t=0$ the function is 1.

Usage

```
const_decay(t, a)

decaying_decay(t, par)
```

Arguments

t	time (in minutes)
a	alpha (in per time, thus in per minute when time is in minutes)
par	vector of length 2 containing alpha (par[1]) and beta (par[2]) values; alpha=initial decay rate, beta=decay of decay rate (both in per time, thus in per minute when time is in minutes)

Value

returns abundance after time t at alpha initial decay rate and beta decay of decay rate relative to an initial abundance of 1

Examples

```
const_decay(10,log(2)/10) ## returns 0.5
decaying_decay(10,c(log(2)/10,0.01)) ##returns 0.5170495
```

decay_data *Normalized RNA abundance RNA decay timecourse*

Description

A long form dataset of RNA abundance of 118 genes in four Arabidopsis thaliana genotypes (WT, sov, vcs, vcs sov). Four biological replicates were collected 0, 7.5, 15, 30, 60, 120, 240, 480 min after blocking transcription. RNA was extracted, subjected to ribodepletion, and sequenced by RNA-seq (Illumina 50 nt single end reads). RPM values were normalized to mean T0 abundance and corrected by a decay factor.

Usage

```
decay_data
```

Format

a data frame with 5 columns and 15104 rows.

geneID gene identifier; AGI

treatment Arabidopsis genotype

t.decay time of decay, in minutes

rep replicate number

value RPM value normalized to the replicate samples' mean T0 abundance and decay factor corrected

Source

Sorenson et al. (2017) Submitted; <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE86361>

decay_plot	<i>decay_plot() function</i>
------------	------------------------------

Description

Plots RNA decay data and/or decay models using the ggplot2 package.

Usage

```
decay_plot(geneID, xlim = c(0, 500), ylim = c(0, 1.25), xticks = NA,
  yticks = 0:5/4, alphaSZ = 8, what = c("Desc", "models", "reps",
    "meanSE", "alphas&betas"), DATA, treatments = NA, colors = NA,
  mod.results = NA, gdesc = NA)
```

Arguments

geneID	single gene ID from data set (e.g. "AT1G00100") for which to plot data/model
xlim, ylim	vector of length 2 defining the limits of the plot (zooms in on data)
xticks, yticks	vectors specifying tick marks for the x and y axes
alphaSZ	text size of alpha and beta parameter labels if plotted
what	character vector specifying what to plot; any or all (default) of "Desc","models","reps","meanSE","alphas&betas" "Desc" - plots gene descriptions behind data "models" - plots the selected fit model "reps" - plots individual replicate data as distinct shapes "meanSE" - plots the replicate means and standard errors "alphas&betas" - plots the values of the alphas and betas for each model below the model at the greatest x position
DATA	(required) normalized abundance decay data with column names: "geneID", "treatment", "t.decay", "rep", "value"
treatments	what treatments/genotypes to plot from the supplied data
colors	vector of R recognized colors (e.g. "red", "darkblue")
mod.results	(optional; required for plotting models) data.frame of the model results as output from the modeling (e.g. "alphas+betas+mods+grps+patterns+relABs.txt")
gdesc	(optional; required for plotting gene descriptions) gene descriptions (geneID-named vector of gene descriptions geneID must match those of data)

Value

returns a ggplot to be used with print; could also be modified using the syntax of ggplot2 e.g. '+geom_XXXX(...)'

Examples

```
p<-decay_plot("Gene_BooFu",
  mod.results = data.frame(alpha_WT = 0.0830195, beta_WT = 0.04998945,
    model = 1, alpha_grp = 1, beta_grp = 1, alpha_subgroup = 1.1,
    row.names = "Gene_BooFu"),
  what = c("meanSE", "alphas&betas", "models"),
  treatments = "WT",
  colors = "black",
  DATA = data.frame(geneID=rep("Gene_BooFu",15),
    treatment=rep("WT",15),
    t.decay=rep(c(0,7.5,15,30,60),3),
    rep=paste0("rep",c(rep(1,5),rep(2,5),rep(3,5))),
    value= c(0.9173587, 0.4798672, 0.3327807, 0.1990708, 0.1656554,
      0.9407511, 0.7062988, 0.3450886, 0.3176824, 0.2749946,
      1.1026497, 0.6156978, 0.4563346, 0.2865779, 0.1680075)),
  xlim = c(0, 65),
  alphaSZ = 10)
print(p)
```

groupings

Combinatorial groups matrix generator

Description

Generates a combinatorial grouping matrix based on the decaydata data.frame.

Usage

```
groupings(decaydata)
```

Arguments

decaydata	a data.frame with column names: 'geneID', 'treatment', 't.decay', 'rep', 'value' with classes factor, factor, numeric, factor, numeric
-----------	--

Details

The resulting matrix of indices is used to constrain treatment alphas or treatment betas in combination. For example, in one model, treatment alphas might be allowed to vary independently (gp1), but the beta models might be constrained to be equal for some treatments indicated by having the same index number (other gp).

Value

returns a matrix of equivalence group indices based on the number of levels in the 'treatment' column (max of 4).

Examples

```
groupings(data.frame(geneID=paste0('gene',1:4),treatment=as.factor(paste0('treat',1:4)),
                    t.decay=0:3,rep=rep('rep1'),value=c(1,0.5,0.25,0.12)))
```

group_map	<i>model color map</i>
-----------	------------------------

Description

group_map makes a color map of alpha and beta equivalence groups by model. Similar colors in a row indicate constrained parameter equivalence between treatments. Gray indicates values of 0.

Usage

```
group_map(decaydata, path, nEquivGrp = nEquivGrp, groups = groups,
          mods = mods)
```

Arguments

decaydata	5 column data.frame with colnames "geneID","treatment","t.decay","rep","value"
path	write path and file name, must end in ".pdf"
nEquivGrp	number of equivalence groups based on number of treatments
groups	equivalence group matrix
mods	alpha beta equivalence group usage index (matrix)

Value

creates a model colormap and writes it to a pdf file named path

Examples

```
group_map(decaydata=data.frame(geneID=paste0("gene",1:4),
                              treatment=as.factor(rep(paste0("treat",1:2),2)),
                              t.decay=0:3,
                              rep=rep("rep1"),
                              value=c(1,0.5,0.25,0.12)),
          path=paste0(tempdir(),"/parameter equivalence colormap.pdf"),
          nEquivGrp = 2,
          groups = t(matrix(c(1,2,1,1,NA,NA),nrow=2,
                            dimnames=list(c("treat1","treat2"),c("grp1","grp2","grp3")))),
          mods = t(matrix(c(1,1,1,2,1,3,2,1,2,2,2,3),nrow=2,
                            dimnames=list(c("a","b"),paste0("mod",1:6))))))
```

models

*Example double exponential decay modeling results***Description**

Example results from maximum likelihood modeling of double exponential RNA decay of 118 genes.

Usage

models

Format

a list of data frames, each with 240 rows (1/model) with 22 columns and 240 rows.

geneID gene identifier

mod model names as factors

alpha_XXX decay rate estimate of genotype XXX, in per time (min^{-1})

beta_XXX decay of decay rate estimate of genotype XXX, in per time (min^{-1})

sigma2 variance estimate

logLik maximum log likelihood

nPar number of parameters in the given model

nStarts number of parameter starting value sets (of 50) that converged on a maximum likelihood peak

J number of parameter starting value sets that converged on the highest - within $1e-4$ - maximum likelihood of all parameter starting value sets

range.LL range of maximum likelihoods values reached by algorithm convergence from all parameter starting value sets

nUnique.LL number of unique maximum likelihoods values reached by algorithm convergence from all parameter starting value sets

C.alpha sum of all coefficients of variation for each column of alpha estimates

C.beta sum of all coefficients of variation for each column of beta estimates

C.tot C.alpha+C.beta

AICc calculated from the single highest maximum likelihood of all parameter starting value sets

AICc_est calculated from the log likelihood value computed by using the mean of each parameter from all optimizations that converged on the highest maximum likelihood of all starting parameter value sets

Source

Sorenson et al. (2017) Submitted; <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE86361>

mod_optimization	<i>model optimization for fitting exponential decay models to normalized data</i>
------------------	---

Description

The `mod_optimization` function finds the estimates of model parameters by maximum likelihood, for a single gene on a specified list of models, and saves a tab delimited text file of the results named, '[geneID]_results.txt'. The function does the following for each gene: (1) it calculates log likelihood for each point in a 2 dimensional grid of evenly spaced alpha and beta values within the alpha and beta bounds specified using the null model (in which all treatment alphas are equivalent and all betas are equivalent). (2) it calculates log likelihood for each point in a 1 dimensional range of evenly spaced alpha values within the alpha bounds using the single exponential null model (in which all treatment alphas are equivalent). (3) For each of the grid points with the highest log likelihood from steps (1) and (2) 25 starting parameter value sets that are normally distributed around these points are generated. (4) Parameter values are optimized for maximum likelihood using each of these 50 starting parameter sets using pre-compiled C++ functions loaded from dynamically linked libraries stored in the package on all models specified in the `models` argument. (5) evaluates parameter estimates of all 50 optimizations based on the reported maximum likelihood upon convergence. Only parameter estimates that converged on the same and highest maximum likelihood are returned. (6) returns the optimized parameter estimates, with model selection statistics.

Usage

```
mod_optimization(gene, data, alpha_bounds, beta_bounds, models, group, mod,
  file_only = TRUE, path = "modeling_results")
```

Arguments

<code>gene</code>	geneID from data to be modeled
<code>data</code>	decay data data.frame with columns named 'geneID', 'treatment', 't.decay', 'rep', 'value.'
<code>alpha_bounds</code>	vector of length 2 with lower and upper bounds for alpha
<code>beta_bounds</code>	vector of length 2 with lower and upper bounds for beta
<code>models</code>	vector specifying which models to run optimization on (e.g., c('mod1', 'mod239'))
<code>group</code>	grouping matrix for alphas or betas
<code>mod</code>	data.frame specifying alpha and beta group pairs for each model
<code>file_only</code>	logical; should output only be written to file (TRUE) or also return a data.frame of the results (FALSE)
<code>path</code>	specify folder for output to be written

Value

returns (if `file_only = FALSE`) and writes to `path` a data frame of model optimization results for models one row for each for gene using values for it found in `data`, the columns of the data frame are: `geneID`, `mod` (model), model estimates [`alpha_treatment1`, ..., `alpha_treatmentn`, `beta_treatment1`, ..., `beta_treatmentn`, `sigma2`], `logLik` (maximum log likelihood), `nPar` (number of parameters in the model), `nStarts` (number of parameter starting value sets (of 50) that converged on a maximum likelihood peak), `J` (number of parameter starting value sets that converged on the

highest - within $1e-4$ - maximum likelihood of all parameter starting value sets), range.LL (range of maximum likelihoods values reached by algorithm convergence from all parameter starting value sets), nUnique.LL (number of unique maximum likelihoods values reached by algorithm convergence from all parameter starting value sets), C.alpha (sum of all coefficients of variation for each column of alpha estimates), C.beta (sum of all coefficients of variation for each column of beta estimates), C.tot (C.alpha+C.beta), AICc (calculated from the single highest maximum likelihood of all parameter starting value sets), AICc_est (calculated from the log likelihood value computed by using the mean of each parameter from all optimizations that converged on the highest maximum likelihood of all starting parameter value sets.)

Examples

```
mod_optimization(gene = 'Gene_BooFu',
  data = data.frame(geneID=rep('Gene_BooFu',30),
    treatment=c(rep('WT',15),rep('mut',15)),
    t.decay=rep(c(0,7.5,15,30,60),6),
    rep=rep(paste0('rep',c(rep(1,5),rep(2,5),rep(3,5))),2),
    value= c(0.9173587, 0.4798672, 0.3327807, 0.1990708, 0.1656554,
      0.9407511, 0.7062988, 0.3450886, 0.3176824, 0.2749946,
      1.1026497, 0.6156978, 0.4563346, 0.2865779, 0.1680075,
      0.8679866, 0.6798788, 0.2683555, 0.5120951, 0.2593122,
      1.1348219, 0.8535835, 0.6423996, 0.5308946, 0.4592902,
      1.1104068, 0.5966838, 0.3949790, 0.3742632, 0.2613560)),
  alpha_bounds = c(1e-4,0.75),
  beta_bounds = c(1e-3,0.075),
  models = 'mod1',
  group = t(matrix(c(1,2,1,1,NA,NA),nrow=2,
    dimnames=list(c('treat1','treat2'),c('mod1','mod2','mod3')))),
  mod = as.data.frame(t(matrix(c(1,1,1,2,1,3,2,1,2,2,2,3),nrow=2,
    dimnames=list(c('a','b'),paste0('mod',1:6))))),
  file_only = FALSE,
  path = paste0(tempdir(),"/modeling results"))
```

plain_theme

a custom ggplot2 theme

Description

A custom ggplot2 theme generating function for ggplot2 plots; can be further manipulated using standard ggplot2 syntax.

Usage

```
plain_theme(bigFont = 30, smFont = 0.85, x.ang = 0,
  leg.pos = c(0.85, 0.85))
```

Arguments

bigFont	larger font size of axis labels in points (used for plot title, axis titles, facet titles)
smFont	fractional multiplier of bigFont (used for axis text)
x.ang	x-axis label angle

`leg.pos` legend position on plot as relative coordinates $c(x,y)$ (i.e., range is [0,1]) or 'right', 'left', 'above', 'below'

Value

returns a ggplot2 theme of class "theme" "gg"

Examples

```
plain_theme(10)
```

results

Example double exponential decay modeling results

Description

Example results from maximum likelihood modeling of double exponential RNA decay of 118 genes. Results include parameter estimates, selected model, and alpha and beta groupings.

Usage

```
results
```

Format

a data frame with 18 columns and 118 rows.

alpha_XXX decay rate estimate of genotype XXX, in per time (min^{-1})

beta_XXX decay of decay rate estimate of genotype XXX, in per time (min^{-1})

sigma2 variance estimate

model selected model number

alpha_grp model alpha grouping number

beta_grp model beta grouping number

alpha_subgroup model alpha subgroup number

alphaPattern model alpha subgroup pattern; i.e. order of genotypes of increasing decay rate

betaPattern model beta subgroup pattern; i.e. order of genotypes of increasing decay of decay rate

rA_XXX relative alpha value of genotype XXX compared to WT

nEqMods number of models that were not different than the selected model based on a AICc difference < 2

nEqAgp number of alpha groups represented in nEqMods

Source

Sorenson et al. (2017) Submitted; <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE86361>

RPMs

RNA abundance reads per million over RNA decay timecourse

Description

A dataset of RNA abundance of 118 genes in four *Arabidopsis thaliana* genotypes (WT, sov, vcs, vcs sov). Four biological replicates were collected 0, 7.5, 15, 30, 60, 120, 240, 480 min after blocking transcription. RNA was extracted, subjected to ribodepletion, and sequenced by RNA-seq (Illumina 50 nt single end reads).

Usage

RPMs

Format

a data frame with 118 rows and 128 columns; data are all RNA abundance values presented as reads per million. Column names indicate genotype, time point, and replicate number separated by underscores.

Source

Sorenson et al. (2017) Submitted; <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE86361>

Index

*Topic **datasets**

decay_data, [4](#)

models, [8](#)

results, [11](#)

RPMs, [12](#)

a_high, [2](#)

a_low (a_high), [2](#)

b_low (a_high), [2](#)

cols, [3](#)

const_decay, [4](#)

decay_data, [4](#)

decay_plot, [5](#)

decaying_decay (const_decay), [4](#)

group_map, [7](#)

groupings, [6](#)

mod_optimization, [9](#)

models, [8](#)

plain_theme, [10](#)

results, [11](#)

RPMs, [12](#)