

Package ‘idopNetwork’

July 22, 2025

Type Package

Title A Network Tool to Dissect Spatial Community Ecology

Version 0.1.2

Author Ang Dong

Maintainer Ang Dong <fantasys05227@gmail.com>

Description Most existing approaches for network reconstruction can only infer an overall network and, also, fail to capture a complete set of network properties. To address these issues, a new model has been developed, which converts static data into their 'dynamic' form. 'idopNetwork' is an 'R' interface to this model, it can inferring informative, dynamic, omnidirectional and personalized networks. For more information on functional clustering part, see Kim et al. (2008) <[doi:10.1534/genetics.108.093690](https://doi.org/10.1534/genetics.108.093690)>, Wang et al. (2011) <[doi:10.1093/bib/bbr032](https://doi.org/10.1093/bib/bbr032)>. For more information on our model, see Chen et al. (2019) <[doi:10.1038/s41540-019-0116-1](https://doi.org/10.1038/s41540-019-0116-1)>, and Cao et al. (2022) <[doi:10.1080/19490976.2022.2106103](https://doi.org/10.1080/19490976.2022.2106103)>.

License GPL (>= 3)

Imports grDevices, stats, mvtnorm, orthopolynom, parallel, deSolve, ggplot2, reshape2, glmnet, igraph, scales, patchwork

Encoding UTF-8

LazyData true

Depends R (>= 3.50)

RoxygenNote 7.2.3

Suggests covr, knitr, rmarkdown

VignetteBuilder knitr

URL <https://github.com/cxzdsa2332/idopNetwork>

BugReports <https://github.com/cxzdsa2332/idopNetwork/issues>

NeedsCompilation no

Repository CRAN

Date/Publication 2023-04-18 06:50:02 UTC

Contents

bifun_clu	3
bifun_clu_convert	4
bifun_clu_parallel	4
bifun_clu_plot	5
biget_par_int	6
bipower_equation_plot	6
biqdODE_plot_all	7
biqdODE_plot_base	8
biQ_function	8
darken	9
data_cleaning	10
data_match	10
fun_clu	11
fun_clu_BIC	12
fun_clu_convert	12
fun_clu_parallel	13
fun_clu_plot	13
fun_clu_select	14
get_biSAD1	14
get_interaction	15
get_legendre_matrix	16
get_legendre_par	16
get_mu	17
get_mu2	17
get_par_int	18
get_SAD1_covmatrix	18
gut_microbe	19
legendre_fit	19
logsumexp	20
mustard_microbe	20
network_conversion	21
network_maxeffect	21
network_plot	22
normalization	22
power_equation	23
power_equation_all	23
power_equation_base	24
power_equation_fit	24
power_equation_plot	25
qdODEmod	25
qdODEplot_convert	26
qdODE_all	26
qdODE_fit	27
qdODE_ls	28
qdODE_parallel	28
qdODE_plot_all	29

<i>bifun_clu</i>	3
qdODE_plot_base	29
Q_function	30
Index	31

bifun_clu	<i>main function for bifunctional clustering</i>
-----------	--

Description

main function for bifunctional clustering

Usage

```
bifun_clu(
  data1,
  data2,
  k,
  Time1 = NULL,
  Time2 = NULL,
  trans = log10,
  inv.cov = NULL,
  initial.pars = NULL,
  iter.max = 100,
  parscale = 0.001
)
```

Arguments

data1	matrix or data for cluster
data2	matrix or data for cluster
k	vector for the cluster number
Time1	vector for the time point
Time2	vector for the time point
trans	indicate log/log2/log10 transform dataset
inv.cov	matrix for directly solve cov matrix, default not given(currently not available)
initial.pars	vector for manual give initial parameters, default not given
iter.max	scales control iteration for EM algorithm
parscale	scales control parameters scales for cov pars

Value

the initial parameters for functional clustering

bifun_clu_convert	<i>convert result of bifunctional clustering result</i>
-------------------	---

Description

convert result of bifunctional clustering result

Usage

```
bifun_clu_convert(result, best.k)
```

Arguments

result	list directly from bifun_clu_parallel function
best.k	scale of BIC-determined cluster number

Value

list contain module data and fitted data

bifun_clu_parallel	<i>parallel version for functional clustering</i>
--------------------	---

Description

parallel version for functional clustering

Usage

```
bifun_clu_parallel(
  data1,
  data2,
  Time1 = NULL,
  Time2 = NULL,
  trans = log10,
  start,
  end,
  iter.max = 100,
  thread = 2
)
```

Arguments

data1	data for cluster
data2	data for cluster
Time1	vector for the time point
Time2	vector for the time point
trans	indicate log/log2/log10 transform dataset
start	vector for the minimum cluster number
end	vector for the maximum cluster number
iter.max	scales control iteration for EM algorithm
thread	scales for how many thread used

Value

the initial parameters for functional clustering

bifun_clu_plot	<i>bifunctional clustering plot</i>
----------------	-------------------------------------

Description

bifunctional clustering plot

Usage

```
bifun_clu_plot(
  result,
  best.k,
  label = 10,
  degree = 1/4,
  show.legend = FALSE,
  color1 = "#38E54D",
  color2 = "#FF8787"
)
```

Arguments

result	list directly from bifun_clu_parallel function
best.k	scale of BIC-determined cluster number
label	relabel x and y label due to log-transform, set 10 as default
degree	scalar control transparency degree
show.legend	show legend or not
color1	Hex Color Codes for first data
color2	Hex Color Codes for second data

Value

functional clustering plot

biget_par_int	<i>acquire initial parameters for functional clustering</i>
---------------	---

Description

acquire initial parameters for functional clustering

Usage

```
biget_par_int(X, k, times1, times2, n1, n2)
```

Arguments

X	matrix for cluster
k	vector for the cluster number
times1	vector for the x values or time points
times2	vector for the x values or time points
n1	scalar for number of column contain first trait/location etc
n2	scalar for number of column contain second trait/location etc

Value

the initial parameters for functional clustering

bipower_equation_plot	<i>plot power equation fitting results for bi-variate model</i>
-----------------------	---

Description

plot power equation fitting results for bi-variate model

Usage

```
bipower_equation_plot(
  result,
  label = 10,
  n = 9,
  show.legend = FALSE,
  color1 = "#38E54D",
  color2 = "#FF8787"
)
```

Arguments

result	list object returned from data_match
label	relabel x and y label due to log-transform, set 10 as default
n	scales for how many subplots needed
show.legend	show legend or not
color1	Hex Color Codes for first data
color2	Hex Color Codes for second data

Value

plot show power curve fitting result

biqdODE_plot_all	<i>plot all decompose plot for two data</i>
------------------	---

Description

plot all decompose plot for two data

Usage

```
biqdODE_plot_all(
  result1,
  result2,
  label = 10,
  show.legend = FALSE,
  remove.label = TRUE,
  nrow = NULL,
  ncol = NULL
)
```

Arguments

result1	list of qdODE all for first data
result2	list of qdODE all for second data
label	relabel x and y label due to log-transform, set 10 as default
show.legend	to show legend
remove.label	to remove x and y label
nrow	scalar for subplot row number
ncol	scalar for subplot column number

biqdODE_plot_base	<i>plot single decompose plot for two data</i>
-------------------	--

Description

plot single decompose plot for two data

Usage

```
biqdODE_plot_base(
  result1,
  result2,
  label = 10,
  show.legend = FALSE,
  remove.label = FALSE
)
```

Arguments

result1	list of qdODE all for first data
result2	list of qdODE all for second data
label	relabel x and y label due to log-transform, set 10 as default
show.legend	to show legend
remove.label	to remove x and y label

biQ_function	<i>Q-function to replace log-likelihood function</i>
--------------	--

Description

Q-function to replace log-likelihood function

Usage

```
biQ_function(par, prob_log, omega_log, X, k, n1, n2, times1, times2)
```

Arguments

par	numeric vector for parameters need to be estimated
prob_log	mixture component weights(log)
omega_log	latent variables(log)
X	matrix for cluster
k	vector for the cluster number

n1	scalar for number of column contain first trait/location etc
n2	scalar for number of column contain second trait/location etc
times1	vector for the x values or time points
times2	vector for the x values or time points

Value

the Loglikelihood value

darken	<i>make color more dark</i>
--------	-----------------------------

Description

make color more dark

Usage

```
darken(color, factor = 1.2)
```

Arguments

color	hex color code
factor	scalar for darken level

Value

darkened hex color code

Examples

```
darken("#FF0000")
```

data_cleaning	<i>remove observation with too many 0 values</i>
---------------	--

Description

remove observation with too many 0 values

Usage

```
data_cleaning(data, x = round(ncol(data) * 0.3))
```

Arguments

data	dataframe of imported dataset, must have first column as ID
x	scales indicate how many 0 to remove

Value

a dataframe without too many 0 observations

Examples

```
data_cleaning(matrix(c(c(0,1,1,0,0,1,1), c(2,1,0,3,5,2,2), c(1,1,3,2,4,5,1)), 3, 7), 2)
```

data_match	<i>match power_equation fit result for bi-variate model</i>
------------	---

Description

match power_equation fit result for bi-variate model

Usage

```
data_match(result1, result2)
```

Arguments

result1	list object from power_equation fit
result2	list object from power_equation fit

Value

a id match list for input dataset

fun_clu	<i>main function for functional clustering</i>
---------	--

Description

main function for functional clustering

Usage

```
fun_clu(  
  data,  
  k,  
  Time = NULL,  
  trans = log10,  
  inv.cov = NULL,  
  initial.pars = NULL,  
  iter.max = 100,  
  parscale = 0.1  
)
```

Arguments

data	matrix or data for cluster
k	vector for the cluster number
Time	vector for the time point
trans	indicate log/log2/log10 transform dataset
inv.cov	matrix for directly solve cov matrix, default not given(currently not available)
initial.pars	vector for manual give initial parameters, default not given
iter.max	scales control iteration for EM algorithm
parscale	scales control parameters scales for cov pars

Value

the initial parameters for functional clustering

fun_clu_BIC	<i>plot BIC results for functional clustering</i>
-------------	---

Description

plot BIC results for functional clustering

Usage

```
fun_clu_BIC(result, crit = "BIC", title = NULL)
```

Arguments

result	list directly from fun_clu_parallel function
crit	either BIC or AIC for module selection
title	title for the plot

Value

the BIC plot

fun_clu_convert	<i>convert result of functional clustering result</i>
-----------------	---

Description

convert result of functional clustering result

Usage

```
fun_clu_convert(result, best.k)
```

Arguments

result	list directly from fun_clu_parallel function
best.k	scale of BIC-determined cluster number

Value

list contain module data and fitted data

fun_clu_parallel	<i>parallel version for functional clustering</i>
------------------	---

Description

parallel version for functional clustering

Usage

```
fun_clu_parallel(  
  data,  
  Time = NULL,  
  trans = log10,  
  start,  
  end,  
  iter.max = 100,  
  thread = 2  
)
```

Arguments

data	data for cluster
Time	vector for the time point
trans	indicate log/log2/log10 transform dataset
start	vector for the minimum cluster number
end	vector for the maximum cluster number
iter.max	scales control iteration for EM algorithm
thread	scales for how many threads used

Value

the initial parameters for functional clustering

fun_clu_plot	<i>functional clustering plot</i>
--------------	-----------------------------------

Description

functional clustering plot

Usage

```
fun_clu_plot(result, best.k, label = 10, degree = 1)
```

Arguments

- result list directly from fun_clu_parallel function
- best.k scalar of BIC-determined cluster number
- label relabel x and y label due to log-transform, set 10 as default
- degree scalar control transparency degree

Value

functional clustering plot

fun_clu_select	<i>select result of functional clustering result</i>
----------------	--

Description

select result of functional clustering result

Usage

```
fun_clu_select(result_fit, result_funclu, i)
```

Arguments

- result_fit list directly from power_equation_fit
- result_funclu list from fun_clu_convert
- i scale of which cluster selected

Value

list contain microbe data and fitted data

get_biSAD1	<i>generate biSAD1 covariance matrix</i>
------------	--

Description

generate biSAD1 covariance matrix

Usage

```
get_biSAD1(par, n1, n2)
```

Arguments

par	vector with four number, first two for ck and the rest for stress
n1	scalar indicate length of time1
n2	scalar indicate length of time2

Value

biSAD1 covariance matrix

Examples

```
get_biSAD1(par=c(2,0.5,2,0.1),n1=4, n2 = 5)
```

get_interaction	<i>Lasso-based variable selection</i>
-----------------	---------------------------------------

Description

Lasso-based variable selection

Usage

```
get_interaction(data, col, reduction = FALSE)
```

Arguments

data	data of clustered results, do not contain cluster column
col	scalar of which row number selected
reduction	use $n/\log(n)$ dimension reduction

Value

list contain relationship of each row

get_legendre_matrix	<i>generate legendre matrix</i>
---------------------	---------------------------------

Description

generate legendre matrix

Usage

```
get_legendre_matrix(x, legendre_order)
```

Arguments

x vector equal to the x value for legendre polynomials(in this case times)
 legendre_order the order of legendre polynomials

Value

the polynomials value of each order

Examples

```
get_legendre_matrix(1:14,4)
```

get_legendre_par	<i>use legendre polynomials to fit a given data</i>
------------------	---

Description

use legendre polynomials to fit a given data

Usage

```
get_legendre_par(y, legendre_order, x)
```

Arguments

y vector equal to the y observed data(in this case generic effect)
 legendre_order scalar of legendre polynomials
 x vector equal to the x value for legendre polynomials(in this case times)

Value

the polynomials coefficients

Examples

```
get_legendre_par(14:1,4,1:14)
```

get_mu	<i>curve fit with modified logistic function</i>
--------	--

Description

curve fit with modified logistic function

Usage

```
get_mu(mu_par, times)
```

Arguments

mu_par	vector with five number
times	vector of time point

Value

numeric vector with the same length to times

Examples

```
get_mu(mu_par = 1:5, times = 1:14)
```

get_mu2	<i>generate mean vectors with ck and stress condition</i>
---------	---

Description

generate mean vectors with ck and stress condition

Usage

```
get_mu2(par, times)
```

Arguments

par	vector with ten number, first five for ck and the rest for stress
times	vector of time point

Value

numeric vector with the double length to times

Examples

```
get_mu2(par = 1:10, times = 1:14)
```

get_par_int	<i>acquire initial parameters for functional clustering</i>
-------------	---

Description

acquire initial parameters for functional clustering

Usage

```
get_par_int(X, k, times)
```

Arguments

X	matrix for cluster
k	vector for the cluster number
times	vector for the x values or time points

Value

the initial parameters for functional clustering

get_SAD1_covmatrix	<i>generate standard SAD1 covariance matrix</i>
--------------------	---

Description

generate standard SAD1 covariance matrix

Usage

```
get_SAD1_covmatrix(par, n)
```

Arguments

par	vector with two number for SAD1 covariance matrix
n	scalar indicate length of time d

Value

SAD1 covariance matrix

Examples

```
get_SAD1_covmatrix(par = c(2,0.5), n = 14)
```

gut_microbe	<i>gut microbe OTU data (species level)</i>
-------------	---

Description

The original nucleotide sequences of this study were deposited to the NCBI Sequence Read Archive under accession number SRP128619.

Usage

```
data(gut_microbe)
```

Format

A data frame with 65 rows and 21 column, contain first column as microbe ID:

legendre_fit	<i>generate curve based on legendre polynomials</i>
--------------	---

Description

generate curve based on legendre polynomials

Usage

```
legendre_fit(par, x)
```

Arguments

par	vector of legendre polynomials coefficients
x	vector equal to the x value for legendre polynomials(in this case times)

Value

the polynomials value

Examples

```
legendre_fit(rep(1,5),1:14)
```

logsumexp	<i>calculate log-sum-exp values</i>
-----------	-------------------------------------

Description

calculate log-sum-exp values

Usage

```
logsumexp(v)
```

Arguments

v	numeric vector
---	----------------

Value

log-sum-exp values

Examples

```
logsumexp(c(100, 1000, 10000))
```

mustard_microbe	<i>mustard microbe OTU data</i>
-----------------	---------------------------------

Description

Wagner, M. R. et al. Host genotype and age shape the leaf and root microbiomes of a wild perennial plant. Nat. Commun. 7:12151 doi: 10.1038/ncomms12151 (2016) This dataset is a subset of otuTable97, we select location = JAM, keep samples with both root and leaf data, and then run data_cleaning first (set x = 50) to reduce size of this data. Moreover, sample 8_1382 is removed for the outlier reason.

Usage

```
data(mustard_microbe)
```

Format

A data frame with 1557 rows and 176 column, contain first column as OTU ID:

network_conversion	<i>convert ODE results(ODE_solving2) to basic network plot table</i>
--------------------	--

Description

convert ODE results(ODE_solving2) to basic network plot table

Usage

```
network_conversion(result)
```

Arguments

result	list result from qsODE_parallel
--------	---------------------------------

Value

a list with basic information to plot network

network_maxeffect	<i>convert ODE results(ODE_solving2) to basic network plot table</i>
-------------------	--

Description

convert ODE results(ODE_solving2) to basic network plot table

Usage

```
network_maxeffect(result)
```

Arguments

result	list result from qsODE_parallel
--------	---------------------------------

Value

a list with basic information to plot network

network_plot	<i>generate network plot</i>
--------------	------------------------------

Description

generate network plot

Usage

```
network_plot(result, title = NULL, maxeffect = NULL, type = NULL)
```

Arguments

result	list result from network_conversion
title	text for plot title
maxeffect	control edge size when compare networks
type	select module effect or microbe effect

Value

network plot

normalization	<i>min-max normalization</i>
---------------	------------------------------

Description

min-max normalization

Usage

```
normalization(x, z = 0.2)
```

Arguments

x	numeric vector
z	scalar add minimum value to avoid 0

Value

normalized vector

Examples

```
normalization(runif(100,min = -100, max = 100))
```

power_equation	<i>use power equation parameters to generate y values</i>
----------------	---

Description

use power equation parameters to generate y values

Usage

```
power_equation(x, power_par)
```

Arguments

x	vector for x values
power_par	matrix contain parameters for power equation

Value

y values for given power equation parameters

Examples

```
power_equation(c(1,2,3,5,7), matrix(c(2,1,1,2),2,2))
```

power_equation_all	<i>use power equation to fit observed values</i>
--------------------	--

Description

use power equation to fit observed values

Usage

```
power_equation_all(x, y, maxit = 100)
```

Arguments

x	vector for x values
y	vector for y values
maxit	numeric value for maximum initial pars try

Value

nls model

Examples

```
power_equation_all(c(1,2,3,5,7), c(5,10,15,17,20))
```

power_equation_base *use power equation to fit observed values*

Description

use power equation to fit observed values

Usage

```
power_equation_base(x, y)
```

Arguments

x	vector for x values
y	vector for y values

Value

nls model

Examples

```
power_equation_base(c(1,2,3,5,7), c(5,10,15,17,20))
```

power_equation_fit *use power equation to fit given dataset*

Description

use power equation to fit given dataset

Usage

```
power_equation_fit(data, n = 30, trans = log10, thread = 2)
```

Arguments

data	cleaned dataframe
n	scales for how many interpolation needed
trans	indicate log/log2/log10 transform dataset
thread	scales for how many thread used

Value

list contain power equation parameters and fitted data

power_equation_plot	<i>plot power equation fitting results</i>
---------------------	--

Description

plot power equation fitting results

Usage

```
power_equation_plot(result, label = 10, n = 9)
```

Arguments

result	list object returned from power_equation_fit
label	relabel x and y label due to log-transform, set 10 as default
n	scales for how many subplots needed

Value

plot show power curve fitting result

qdODEmod	<i>quasi-dynamic lotka volterra model</i>
----------	---

Description

quasi-dynamic lotka volterra model

Usage

```
qdODEmod(Time, State, Pars, power_par)
```

Arguments

Time	vector of time point
State	vector of ODE initial state
Pars	vector for unknown ODE parameters
power_par	matrix of power equation parameters for dependent effect

Value

list used in ode function

qdODEplot_convert	<i>convert qdODE results to plot data</i>
-------------------	---

Description

convert qdODE results to plot data

Usage

```
qdODEplot_convert(result)
```

Arguments

result	list of qdODE all
--------	-------------------

qdODE_all	<i>wrapper for qdODE model</i>
-----------	--------------------------------

Description

wrapper for qdODE model

Usage

```
qdODE_all(
  result,
  relationship,
  i,
  init_pars = 1,
  LOP_order = 6,
  method = "ls",
  new_time = NULL,
  n_expand = 100,
  maxit = 1000
)
```

Arguments

result	result from power_equation_fit
relationship	list contain variable selection results
i	scalar for which id used for qdODE solving, must <= nrow
init_pars	scalar for initial parameters
LOP_order	scalar of LOP order
method	scalar of qdODE solving methodm, cuurent only support least square

new_time	vector produce new defined time point
n_expand	scalar for how many interpolation needed
maxit	scalar of Optim iteration setting

Value

list contain variable selection results and LOP parameters for every row

qdODE_fit	<i>legendre polynomials fit to qdODE model</i>
-----------	--

Description

legendre polynomials fit to qdODE model

Usage

```
qdODE_fit(  
  pars,  
  data,  
  Time,  
  power_par,  
  LOP_order = 6,  
  new_time = NULL,  
  n_expand = 100  
)
```

Arguments

pars	vector of qdODE parameters
data	dataframe of observed data
Time	vector of time point
power_par	matrix of power equation parameters for dependent effect
LOP_order	scalar of LOP order
new_time	vector produce new defined time point
n_expand	scalar for how many interpolation needed

Value

list contain legendre polynomials parameters, qdODE values and LOP fitted values

qdODE_ls	<i>least-square fit for qdODE model</i>
----------	---

Description

least-square fit for qdODE model

Usage

```
qdODE_ls(pars, data, Time, power_par)
```

Arguments

pars	vector for unknown ODE parameters
data	data contain independent effect as first row and dependent effect
Time	vector of time point
power_par	matrix of power equation parameters for dependent effect

Value

mean-square error

qdODE_parallel	<i>wrapper for qdODE_all in parallel version</i>
----------------	--

Description

wrapper for qdODE_all in parallel version

Usage

```
qdODE_parallel(result, reduction = FALSE, thread = 2, maxit = 1000)
```

Arguments

result	result from power_equation_fit
reduction	use n/log(n) dimension reduction
thread	scales for how many threads used
maxit	scalar of Optim iteration setting

Value

list contain variable selection results and LOP parameters for every row

qdODE_plot_all	<i>plot all decompose plot</i>
----------------	--------------------------------

Description

plot all decompose plot

Usage

```
qdODE_plot_all(
  result,
  label = 10,
  show.legend = TRUE,
  nrow = NULL,
  ncol = NULL
)
```

Arguments

result	list of qdODE parallel
label	relabel x and y label due to log-transform, set 10 as default
show.legend	to show legend
nrow	scalar for subplot row number
ncol	scalar for subplot column number

Value

all effect curve decompose plot

qdODE_plot_base	<i>plot single decompose plot</i>
-----------------	-----------------------------------

Description

plot single decompose plot

Usage

```
qdODE_plot_base(result, label = 10, show.legend = TRUE)
```

Arguments

result	list of qdODE all
label	relabel x and y label due to log-transform, set 10 as default
show.legend	to show legend

Q_function	<i>Q-function to replace log-likelihood function</i>
------------	--

Description

Q-function to replace log-likelihood function

Usage

```
Q_function(par, prob_log, omega_log, X, k, times)
```

Arguments

par	numeric vector for parameters need to be estimated
prob_log	mixture component weights(log)
omega_log	latent variables(log)
X	matrix for cluster
k	vector for the cluster number
times	vector for the x values or time points

Value

the Loglikelihood value

Index

* datasets

gut_microbe, [19](#)
mustard_microbe, [20](#)

bifun_clu, [3](#)
bifun_clu_convert, [4](#)
bifun_clu_parallel, [4](#)
bifun_clu_plot, [5](#)
biget_par_int, [6](#)
bipower_equation_plot, [6](#)
biQ_function, [8](#)
biqdODE_plot_all, [7](#)
biqdODE_plot_base, [8](#)

darken, [9](#)
data_cleaning, [10](#)
data_match, [10](#)

fun_clu, [11](#)
fun_clu_BIC, [12](#)
fun_clu_convert, [12](#)
fun_clu_parallel, [13](#)
fun_clu_plot, [13](#)
fun_clu_select, [14](#)

get_biSAD1, [14](#)
get_interaction, [15](#)
get_legendre_matrix, [16](#)
get_legendre_par, [16](#)
get_mu, [17](#)
get_mu2, [17](#)
get_par_int, [18](#)
get_SAD1_covmatrix, [18](#)
gut_microbe, [19](#)

legendre_fit, [19](#)
logsumexp, [20](#)

mustard_microbe, [20](#)

network_conversion, [21](#)

network_maxeffect, [21](#)
network_plot, [22](#)
normalization, [22](#)

power_equation, [23](#)
power_equation_all, [23](#)
power_equation_base, [24](#)
power_equation_fit, [24](#)
power_equation_plot, [25](#)

Q_function, [30](#)
qdODE_all, [26](#)
qdODE_fit, [27](#)
qdODE_ls, [28](#)
qdODE_parallel, [28](#)
qdODE_plot_all, [29](#)
qdODE_plot_base, [29](#)
qdODEmod, [25](#)
qdODEplot_convert, [26](#)