Package 'mixchar'

May 8, 2026

Title Mixture Model for the Deconvolution of Thermal Decay Curves

Version 0.1.0

Date 2018-08-11

Description Deconvolution of thermal decay curves allows you to quantify proportions of biomass components in plant litter. Thermal decay curves derived from thermogravimetric analysis (TGA) are imported, modified, and then modelled in a three- or four- part mixture model using the Fraser-Suzuki function. The output is estimates for weights of pseudo-components corresponding to hemicellulose, cellulose, and lignin. For more information see: Müller-Hagedorn, M. and Bockhorn, H. (2007) <doi:10.1016/j.jaap.2006.12.008>, Órfão, J. J. M. and Figueiredo, J. L. (2001) <doi:10.1016/S0040-6031(01)00634-7>, and Yang, H. and Yan, R. and Chen, H. and Zheng, C. and Lee, D. H. and Liang, D. T. (2006) <doi:10.1021/ef0580117>.

Depends R (>= 3.2.0)

Imports graphics, minpack.lm, nloptr, stats, zoo, tmvtnorm

License MIT + file LICENSE

URL http://github.com/smwindecker/mixchar

BugReports http://github.com/smwindecker/mixchar/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

Suggests knitr, rmarkdown, devtools, testthat, covr

VignetteBuilder knitr

NeedsCompilation no

Author Saras Windecker [aut, cre],
Nick Golding [aut]

Maintainer Saras Windecker <saras.windecker@gmail.com>

Repository CRAN

Date/Publication 2018-08-16 11:50:09 UTC

Contents

`component_weights` *Accessor function to extract mean weights*

Description

Accessor function to extract mean weights

Usage

```
component_weights(object)
```

Arguments

`object`	a decon object

Value

Extract mean fractions of the object

Examples

```
data(juncus)
tmp <- process(juncus, init_mass = 18.96,
               temp = 'temp_C', mass_loss = 'mass_loss')
output <- deconvolve(tmp)
component_weights(output)
```

`deconvolve` *Deconvolves Thermogravimetric Data*

Description

This function deconvolves thermogravimetric data using a Fraser-Suzuki mixture model

Usage

```
deconvolve(process_object, lower_temp = 120, upper_temp = 700,
  seed = 1, n_peaks = NULL, start_vec = NULL, lower_vec = NULL,
  upper_vec = NULL)
```

Arguments

`process_object`	process object obtained from process function
`lower_temp`	lower temperature bound to crop dataset, default to 120
`upper_temp`	upper temperature bound to crop dataset, default to 700
`seed`	random seed for nloptr optimiser
`n_peaks`	number of curves optional specification
`start_vec`	vector of starting values for nls function. Only specify this vector if you have selected the number of curves in the n_peaks parameter.
`lower_vec`	vector of lower bound values for nls. Only specify this vector if you have selected the number of curves in the n_peaks parameter.
`upper_vec`	vector of upper bound values for nls. Only specify this vector if you have selected the number of curves in the n_peaks parameter.

Value

decon list containing amended dataframe, temperature bounds, minpack.lm model fit, the number of curves fit, and estimated component weights

Examples

```
data(juncus)
tmp <- process(juncus, init_mass = 18.96,
               temp = 'temp_C', mass_loss = 'mass_loss')
output <- deconvolve(tmp)
my_starting_vec <- c(height_1 = 0.003, skew_1 = -0.15, position_1 = 250, width_1 = 50,
                     height_2 = 0.006, skew_2 = -0.15, position_2 = 320, width_2 = 30,
                     height_3 = 0.001, skew_3 = -0.15, position_3 = 390, width_3 = 200)
output <- deconvolve(tmp, n_peaks = 3, start_vec = my_starting_vec)
```

---

fs_function *Fraser-Suzuki function for a single curve*

Description

This function calculates the Fraser-Suzuki function.

Usage

```
fs_function(temp, height, skew, position, width)
```

Arguments

temp	temperature values
height	height value
skew	shape value
position	position value
width	width value

Value

Fraser-Suzuki function

Examples

```
temp <- 150:600
fs_output <- fs_function(temp, height = 0.004, skew = -.15,
position = 250, width = 50)
```

`fs_mixture` *Fraser-Suzuki mixture model*

Description

Fraser-Suzuki mixture model

Usage

```
fs_mixture(temp, height_1, skew_1, position_1, width_1, height_2, skew_2,
  position_2, width_2, height_3, skew_3, position_3, width_3,
  height_0 = NULL, skew_0 = NULL, position_0 = NULL,
  width_0 = NULL)
```

Arguments

`temp`	temperature values
`height_1`	height value for hemicellulose
`skew_1`	shape value for hemicellulose
`position_1`	position value for hemicellulose
`width_1`	width value for hemicellulose
`height_2`	height value for cellulose
`skew_2`	shape value for cellulose
`position_2`	position value for cellulose
`width_2`	width value for cellulose
`height_3`	height value for lignin
`skew_3`	shape value for lignin
`position_3`	position value for lignin
`width_3`	width value for lignin
`height_0`	height value for second hemicellulose curve, if present
`skew_0`	shape value for second hemicellulose curve, if present
`position_0`	position value for second hemicellulose curve, if present
`width_0`	width value for second hemicellulose curve, if present

Value

Fraser-Suzuki model output

Examples

```
temp <- 150:600
fs_mixture_output <- fs_mixture(temp,
height_1 = 0.003, skew_1 = -0.15, position_1 = 250, width_1 = 50,
height_2 = 0.006, skew_2 = -0.15, position_2 = 320, width_2 = 30,
height_3 = 0.001, skew_3 = -0.15, position_3 = 390, width_3 = 200)
```

fs_model *Non-linear model using Fraser-Suzuki mixture model*

Description

Non-linear model output using optimised parameter values with a three-part mixture model using Fraser-Suzuki equation

Usage

```
fs_model(dataframe, params, lb, ub)
```

Arguments

dataframe	dataframe
params	starting parameter values
lb	lower bounds for model
ub	upper bounds for model

Value

model output

get_weights *Calculate weight quantiles*

Description

Calculate weight quantiles

Usage

```
get_weights(param_vec, output)
```

Arguments

param_vec	parameter estimates from minpack model
output	deconvolve output of model

Value

weights for each component

juncus *Thermogravimetric data for Juncus amabilis*

Description

Raw thermogravimetric data from the wetland rush, J. amabilis

Usage

```
data(juncus)
```

Format

An object of class `'cross'`

Source

Saras M Windecker

Examples

```
data(juncus)
```

marsilea *Thermogravimetric data for Marsilea drumondii*

Description

Raw thermogravimetric data from the wetland forb, M. drumondii.

Usage

```
data(marsilea)
```

Format

An object of class `'cross'`

Source

Saras M Windecker

Examples

```
data(marsilea)
```

model_fit *Accessor function to extract model fit*

Description

Accessor function to extract model fit

Usage

```
model_fit(object)
```

Arguments

object	a decon object

Value

$minpack.lm of the object

Examples

```
data(juncus)
tmp <- process(juncus, init_mass = 18.96,
               temp = 'temp_C', mass_loss = 'mass_loss')
output <- deconvolve(tmp)
model_fit(output)
```

model_parameters *Accessor function to extract model parameters*

Description

Accessor function to extract model parameters

Usage

```
model_parameters(object)
```

Arguments

object	a decon object

Value

model parameters from minpack.lm::nlsLM fit

Examples

```
data(juncus)
tmp <- process(juncus, init_mass = 18.96,
               temp = 'temp_C', mass_loss = 'mass_loss')
output <- deconvolve(tmp)
model_parameters(output)
```

`plot.decon` *Default S3 plot method for decon objects (derived from 'deconvolve()')*

Description

This function sets up the default plotting method for outputs from deconvolve function

Usage

```
## S3 method for class 'decon'
plot(x, bw = TRUE, ...)
```

Arguments

`x`	decon object as generated by deconvolve
`bw`	logical argument indicating whether the plot should be in black and white or colour
`...`	other options passed to plot

Value

plot

`plot.process` *Default S3 plot method for process objects (derived from 'process()')*

Description

This function sets up the default plotting method for outputs from process function

Usage

```
## S3 method for class 'process'
plot(x, plot_type = NULL, cex = 1, ...)
```

Arguments

x	process object as generated by process
plot_type	defaults to both plots. Can specify 'mass' or 'rate' curves by themselves.
cex	size of plots features
...	other options passed to plot

Value

plot

---

print.decon *Default S3 print method for decon object (derived from 'deconvolve()')*

Description

This function sets up the default print method for outputs from deconvolve function

Usage

```
## S3 method for class 'decon'
print(x, ...)
```

Arguments

x	decon object as generated by deconvolve
...	other options passed to plot

Value

print output

---

print.process *Default S3 print method for process object (derived from 'process()')*

Description

This function sets up the default print method for outputs from process function

Usage

```
## S3 method for class 'process'
print(x, ...)
```

Arguments

x	process object as generated by deconvolve
...	other options passed to plot

Value

print output

---

process *Calculates the derivative rate of mass loss of thermogravimetric data*

Description

This function processes thermogravimetric data by calculating the derivative of mass loss

Usage

```
process(data, init_mass, temp, mass_loss = NULL, mass = NULL,
  temp_units = "C")
```

Arguments

data	dataframe
init_mass	numeric value of initial sample mass in mg
temp	column name containing temperature values
mass_loss	column name containing mass loss values in mg
mass	column name containing mass values in mg
temp_units	specify units of temperature, default = Celsius. Can specify 'K' or 'Kelvin' if in Kelvin

Value

process list containing modified dataframe, initial mass of sample, and maximum and minimum temperature values

Examples

```
data(juncus)
tmp <- process(juncus, init_mass = 18.96,
               temp = 'temp_C', mass_loss = 'mass_loss')
```

rate_data *Accessor function to extract processed dataframe*

Description

Accessor function to extract processed dataframe

Usage

```
rate_data(object)
```

Arguments

object	a process or deconvolve object

Value

Dataframe of the object

Examples

```
data(juncus)
tmp <- process(juncus, init_mass = 18.96,
               temp = 'temp_C', mass_loss = 'mass_loss')
rate_data(tmp)
```

temp_bounds *Accessor function to extract selected temperature bounds*

Description

Accessor function to extract selected temperature bounds

Usage

```
temp_bounds(object)
```

Arguments

object	the output of either the process or deconvolve functions

Value

Temperature bounds of the data in the object

Examples

```
data(juncus)
tmp <- process(juncus, init_mass = 18.96,
               temp = 'temp_C', mass_loss = 'mass_loss')
temp_bounds(tmp)
```

`weight_quantiles` *Calculate weight quantiles*

Description

Calculate weight quantiles

Usage

```
weight_quantiles(output, seed)
```

Arguments

`output`	dataframe
`seed`	seed

Value

list of means and confidence intervals of weight estimates

`wt_component` *Calculate weight single component*

Description

Calculate weight single component

Usage

```
wt_component(j, param_vec, lower_temp, upper_temp)
```

Arguments

`j`	component
`param_vec`	vector of parameters
`lower_temp`	lower temperature bound
`upper_temp`	upper temperature bound

Value

weight of component

Index