

# Package ‘breathtestcore’

May 8, 2026

**Title** Core Functions to Read and Fit 13c Time Series from Breath Tests

**Version** 0.8.10

**Description** Reads several formats of 13C data (IRIS/Wagner, BreathID) and CSV. Creates artificial sample data for testing. Fits Maes/Ghoos, Bluck-Coward self-correcting formula using 'nls', 'nlme'. Methods to fit breath test curves with Bayesian Stan methods are refactored to package 'breathteststan'. For a Shiny GUI, see package 'dmenne/breathtestshiny' on github.

**License** GPL-3

**Depends** R (>= 4.1.0)

**Imports** assertthat, dplyr, ggfittext, ggplot2, broom (>= 0.7.0), graphics, grid, MASS, methods, multcomp, nlme, purrr, readr, readxl, signal, stats, stringr, tibble (>= 3.0.0), tidyr, tools, utils, xml2

**Suggests** base, gridExtra, qpdf, knitr, rmarkdown, testthat(>= 2.99), breathteststan, covr

**URL** <https://github.com/dmenne/breathtestcore>,

**BugReports** <https://github.com/dmenne/breathtestcore/issues>

**VignetteBuilder** knitr

**Encoding** UTF-8

**SystemRequirements** pandoc

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Dieter Menne [aut, cre],  
Menne Biomed Consulting Tuebingen [cph],  
Benjamin Misselwitz [fnd],  
Mark Fox [fnd],  
Andreas Steingoetter [dtc],  
University Hospital of Zurich, Dep. Gastroenterology [fnd, dtc]

**Maintainer** Dieter Menne <dieter.menne@menne-biomed.de>

**Repository** CRAN

**Date/Publication** 2025-10-28 11:20:08 UTC

## Contents

AIC.breathtestnlmefit . . . . .	3
augment.breathtestfit . . . . .	3
breathtest_data . . . . .	4
breathtest_read_function . . . . .	6
btcore_file . . . . .	7
cleanup_data . . . . .	7
coef.breathtestfit . . . . .	9
coef_by_group . . . . .	10
coef_diff_by_group . . . . .	11
cum_exp_beta . . . . .	12
dob_to_pdr . . . . .	13
exp_beta . . . . .	14
extract_id . . . . .	17
nlme_fit . . . . .	17
nls_fit . . . . .	19
null_fit . . . . .	20
plot.breathtestfit . . . . .	21
read_any_breathtest . . . . .	22
read_breathid . . . . .	22
read_breathid_xml . . . . .	23
read_breathtest_excel . . . . .	24
read_iris . . . . .	25
read_iris_csv . . . . .	26
sigma.breathtestnlmefit . . . . .	27
simulate_breathtest_data . . . . .	27
subsample_data . . . . .	29
t50_bluck_coward . . . . .	29
t50_maes_ghoos . . . . .	31
t50_maes_ghoos_scintigraphy . . . . .	32
tidy.breathtestfit . . . . .	33
tlag_bluck_coward . . . . .	34
tlag_maes_ghoos . . . . .	34
usz_13c . . . . .	35
usz_13c_a . . . . .	36
usz_13c_d . . . . .	36

**Index**

**38**

---

AIC.breathtestnlmefit *S3 AIC method for breathtestnlmefit*

---

### Description

Extract AIC from a model fitted with `nlme_fit`

### Usage

```
## S3 method for class 'breathtestnlmefit'
AIC(object, ...)
```

### Arguments

<code>object</code>	of class <code>breathtestnlmefit</code>
<code>...</code>	not used

---

`augment.breathtestfit` *Augmented prediction for breathtest fit*

---

### Description

Broom method `augment` to compute predicted values from the results of class `breathtestfit` as generated by `nls_fit` or `nlme_fit`.

### Usage

```
## S3 method for class 'breathtestfit'
augment(x, by = NULL, minute = NULL, dose = 100, ...)
```

### Arguments

<code>x</code>	Object of class <code>breathtestfit</code>
<code>by</code>	When <code>by</code> is <code>NULL</code> , predictions for the original data values are returned. When <code>by</code> is a positive number, it is used as a step size for a sequence of minutes from 0 to the maximum value of <code>minute</code> in data set.
<code>minute</code>	When a vector is passed, this overrides settings in <code>by</code> , and predictions are calculated at the requested <code>minute</code> values.
<code>dose</code>	13C acetate or octanoate dose
<code>...</code>	other parameters passed to methods

### Value

When `by` is `NULL`, returns one row for each original observation `pdr`, and column `fitted`. If new data are given, i.e. when one of `parameter by` or `minute` is not null, only column `fitted` is added.

**See Also**[augment](#)**Examples**

```
library(broom)
# Generate simulated data
data = cleanup_data(simulate_breathtest_data(n_records = 3)$data)
# Fit using the curves individually
fit = nls_fit(data)
# Predict values at t=60 and t=120
augment(fit, minute = c(60, 120))
```

---

breathtest_data	<i>Data structure with PDR data and descriptors for breath test records</i>
-----------------	---

---

**Description**

Generates structure of class `breathtest_data` with required fields and optional fields. Optional fields by default are NA. This structure is used internally as an intermediate when reading in external file formats. All `read_XXX` functions return this structure, or a list of this structure (e.g. [read\\_breathid\\_xml](#)), and any converter to a new format should do the same to be used with [cleanup\\_data](#). To support a new format with, also update [breathtest\\_read\\_function](#) which returns the most likely function to read the file by reading a few lines in it.

**Usage**

```
breathtest_data(
  patient_id,
  name = NA,
  first_name = NA,
  initials = NA,
  dob = NA,
  birth_year = NA,
  gender = NA,
  study = NA,
  pat_study_id = NA,
  file_name,
  device = "generic",
  substrate,
  record_date,
  start_time = record_date,
  end_time = record_date,
  test_no,
  dose = 100,
  height = 180,
```

```

    weight = 75,
    t50 = NA,
    gec = NA,
    tlag = NA,
    data = data
  )

```

### Arguments

patient_id	required, string or number for unique identification
name	optional
first_name	optional
initials	optional, 2 characters, 1 number
dob	optional date of birth (not to be confused with "delta over baseline")
birth_year	optional
gender	optional m or f
study	optional name of study; can be used in population fit
pat_study_id	optional; patient number within study_ does not need to be globally unique
file_name	required; file where data were read from, or other unique string_ when data are read again, this string is tested and record is skipped when same filename is already in database, therefore uniqueness is important_ when some record does not turn up in database after repeated reading, check if a record with the same file name is already there, and rename the file to avoid collisions_
device	breath_id or iris; default "generic"
substrate	should contain string "ace" or "oct" or "okt", case insensitive_ will be replaced by "acetate" or "octanoate". If empty, "ocatanoate" is assumed.
record_date	required record date_
start_time	optional
end_time	optional
test_no	required integer; unique test number converted to integer if factor
dose	optional, default 100 mg
height	optional, in cm; when pdr must be calculated, default values are used; see <a href="#">dob_to_pdr</a>
weight	optional, in kg
t50	optional, only present if device computes this value
gec	optional, only present if device computes this value
tlag	optional, only present if device computes this value
data	data frame with at least 5 rows and columns minute or time and one or both of dob or pdr. If pdr is missing, and height, weight and substrate are given, computes pdr via function <a href="#">dob_to_pdr</a> . When height and weight are missing, defaults 180 cm and 75 kg are used instead.

**Examples**

```
# Read a file with known format
iris_csv_file = btcore_file("IrisCSV.TXT")
iris_csv_data = read_iris_csv(iris_csv_file)
# Note that many fields are NA
str(iris_csv_data)
# Convert to a format that can be fed to one of the fit functions
iris_df = cleanup_data(iris_csv_data)
# Individual curve fit
coef(nls_fit(iris_df))
```

---

```
breathtest_read_function
```

*Snoop method to read breath test file*

---

**Description**

Reads the first line of a file, and returns the best matching function to read the breath test data in it. To automatically read the file with the inferred file type, use [read\\_any\\_breathtest](#). For Excel files, only the first sheet is read.

**Usage**

```
breathtest_read_function(filename = NULL, text = NULL)
```

**Arguments**

filename	breath test data file from Iris/Wagner (extended or CSV), BreathID
text	as alternative to filename, pass the text of the file directly. This parameter is not used for Excel files.

**Value**

Function to read the file or the text; NULL if no matching function was found

**Examples**

```
file = btcore_file("IrisCSV.TXT")
# Get function to read this file. Returns \code{\link{read_iris_csv}}.
read_fun = breathtest_read_function(file)
str(read_fun(file))
# or, simple (returns a list!)
str(read_any_breathtest(file), 1 )
```

---

btcore_file	<i>Path to example breath test data file</i>
-------------	--

---

**Description**

Path to example breath test data file

**Usage**

```
btcore_file(filename = NULL, full.names = FALSE)
```

**Arguments**

filename	example file in extdata directory without path. Case sensitive on Unix. When filename is missing, returns the names of the available sample files.
full.names	When filename is NULL, return full path names

**Value**

full filename to example file to use in read\_XXX

**Examples**

```
head(btcore_file())
filename = btcore_file("IrisMulti.TXT")
data = read_iris(filename)
```

---

cleanup_data	<i>Transforms 13C breath data into a clean format for fitting</i>
--------------	---

---

**Description**

Accepts various data formats of ungrouped or grouped 13C breath test time series, and transforms these into a data frame that can be used by all fitting functions, e.g. [nls\\_fit](#). If in doubt, pass data frame through `cleanup_data` before forwarding it to a fitting function. If the function cannot repair the format, it gives better error messages than the `xxx_fit` functions.

**Usage**

```
cleanup_data(data, ...)
```

## Arguments

- `data`
- A data frame, array or tibble with at least two numeric columns with optional names `minute` and `pdr` to fit a single 13C record.
  - A data frame or tibble with three columns named `patient_id`, `minute` and `pdr`.
  - A matrix that can be converted to one of the above.
  - A list of data frames/tibbles that are concatenated. When the list has named elements, the names are converted to group labels. When the list elements are not named, group name `A` is used for all items.
  - A structure of class `breathtest_data`, as imported from a file with `read_any_breathtest`
  - A list of class `breathtest_data_list` as generated from read function such as `read_breathid_xml`
- `...` optional.
- use\_filename\_as\_patient\_id** Always use filename instead of patient name. Use this when patient id are not unique.

## Value

A tibble with 4 columns. Column `patient_id` is created with a dummy entry of `pat_a` if no `patient_id` was present in the input data set. A column `group` is required in the input data if the patients are from different treatment groups or within-subject repeats, e.g. in crossover design. A dummy group name "A" is added if no group column was available in the input data set. If group is present, this is a hint to the analysis functions to do post-hoc breakdown or use it as a grouping variable in population-based methods. A patient can have records in multiple groups, for example in a cross-over designs.

Columns `minute` and `pdr` are the same as given on input, but negative minute values are removed, and an entry at 0 minutes is shifted to 0.01 minutes because most fit methods cannot handle the singularity at  $t=0$ .

An error is raised if dummy columns `patient_id` and `group` cannot be added in a unique way, i.e. when multiple values for a given minute cannot be disambiguated.

Comments are persistent; multiple comments are concatenated with newline separators.

## Examples

```
options(digits = 4)
# Full manual
minute = seq(0,30, by = 10)
data1 = data.frame(minute,
  pdr = exp_beta(minute, dose = 100, m = 30, k = 0.01, beta = 2))
# Two columns with data at t = 0
data1
# Four columns with data at t = 0.01
cleanup_data(data1)

# Results from simulate_breathtest_data can be passed directly to cleanup_data
cleanup_data(simulate_breathtest_data(3))
# .. which implicitly does
```

```

cleanup_data(simulate_breathtest_data(3)$data)

# Use simulated data
data2 = list(
  Z = simulate_breathtest_data(seed = 10)$data,
  Y = simulate_breathtest_data(seed = 11)$data)
d = cleanup_data(data2)
str(d)
unique(d$patient_id)
unique(d$group)
# "Z" "Y"

# Mix multiple input formats
f1 = btcore_file("350_20043_0_GER.txt")
f2 = btcore_file("IrisMulti.TXT")
f3 = btcore_file("IrisCSV.TXT")
# With a named list, the name is used as a group parameter
data = list(A = read_breathid(f1), B = read_iris(f2), C = read_iris_csv(f3))
d = cleanup_data(data)
str(d)
unique(d$patient_id)
# "350_20043_0_GER" "1871960" "123456"
# File name is used as patient name if none is available
unique(d$group)
# "A" "B" "C"

```

---

coef.breathtestfit      *S3 coef and summary for breathtestfit*

---

## Description

Function `coef` extracts the estimates such as `t50`, `tlag`, from fitted 13C beta exponential models. The result is the same as `fit$coef`, but without column `stat`, which always is "estimate" for [nls\\_fit](#) and [nlme\\_fit](#).

The summary method only extracts `t50` by the Maes/Ghoos method

## Usage

```

## S3 method for class 'breathtestfit'
coef(object, ...)

```

## Arguments

<code>object</code>	of class <code>breathtestfit</code> , as returned by <a href="#">nls_fit</a> or <a href="#">nlme_fit</a>
<code>...</code>	other parameters passed to methods

**Examples**

```

# Generate simulated data
data = cleanup_data(simulate_breathtest_data())
# Fit with the population method
fit = nlme_fit(data)
# All coefficients in the long form
coef(fit)
# Access coefficients directly
fit$coef
# Only t50 by Maes/Ghoos
# Can also be used with stan fit (slow!)
## Not run:
if (require("breathteststan")) {
  fit = stan_fit(data, iter = 300, chain = 1)
  coef(fit)
  # We get quantiles here in key/value format
  unique(fit$coef$stat)
}

## End(Not run)

```

---

coef\_by\_group

*Tabulates per-group breath test parameters*


---

**Description**

Given a fit to 13C breath test curves, computes absolute values and their confidence intervals of parameters, e.g. of the half emptying time t50. Generic S3 method for class `breathtestfit`.

**Usage**

```
coef_by_group(fit, ...)
```

**Arguments**

<code>fit</code>	Object of class <code>breathtestfit</code> , for example from <code>nlme_fit</code> , <code>nls_fit</code> or <code>stan_fit</code>
<code>...</code>	Not used

**Value**

A tibble of class `coef_by_group` with columns

**parameter** Parameter of fit, e.g. beta, k, m, t50

**method** Method used to compute parameter. `exp_beta` refers to primary fit parameters beta, k, m. `maes_ghoos` uses the method from Maes B D, Ghoos Y F, Rutgeerts P J, Hiele M I, Geypens B and Vantrappen G 1994 Dig. Dis. Sci. 39 S104-6. `bluck_coward` is the self-correcting method from Bluck L J C and Coward W A 2006

**group** Grouping parameter of the fit, e.g. patient, normal, liquid, solid

**estimate** Parameter estimate

**conf.low, conf.high** Lower and upper 95 estimate.

**diff\_group** Letters a, b, c indicate that parameter would be in mutually significantly different groups. Letter combinations like ab or abc indicated that this parameter is not significantly different from the given other groups in a Tukey-corrected pairwise test.

## Examples

```
library(dplyr)
data("usz_13c")
data = usz_13c %>%
  dplyr::filter( patient_id %in%
    c("norm_001", "norm_002", "norm_003", "norm_004", "pat_001", "pat_002", "pat_003")) %>%
  cleanup_data()
fit = nls_fit(data)
coef_by_group(fit)

fit = nlme_fit(data)
coef_by_group(fit)
```

---

coef\_diff\_by\_group      *Tabulates breath test parameter differences of groups*

---

## Description

Given a fit to 13C breath test curves, computes between-group confidence intervals and p-values, for examples of the half emptying time  $t_{50}$ , with correction for multiple testing.

## Usage

```
coef_diff_by_group(fit, mcp_group = "Tukey", reference_group = NULL, ...)
```

## Arguments

fit	Object of class <code>breathtestfit</code> , for example from <code>nlme_fit</code> , <code>nls_fit</code>
mcp_group	"Tukey" (default) for all pairwise comparisons, "Dunnett" for comparisons relative to the reference group.
reference_group	Used as the first group and as reference group for <code>mcp_group == "Dunnett"</code>
...	Not used

**Value**

A tibble of class `coef_diff_by_group` with columns

**parameter** Parameter of fit, e.g. beta, k, m, t50

**method** Method used to compute parameter. `exp_beta` refers to primary fit parameters beta, k, m. `maes_ghoos` uses the method from Maes B D, Ghooos Y F, Rutgeerts P J, Hiele M I, Geypens B and Vantrappen G 1994 Dig. Dis. Sci. 39 S104-6. `bluck_coward` is the self-correcting method from Bluck L J C and Coward W A 2006

**groups** Which pairwise difference, e.g solid - liquid

**estimate** Estimate of the difference

**conf.low, conf.high** Lower and upper 95 A comparison is significantly different from zero when both estimates have the same sign.

**p.value** p-value of the difference against 0, corrected for multiple testing

**Examples**

```
library(dplyr)
data("usz_13c")
data = usz_13c %>%
  dplyr::filter( patient_id %in%
    c("norm_001", "norm_002", "norm_003", "norm_004", "pat_001", "pat_002", "pat_003")) %>%
  cleanup_data()
fit = nls_fit(data)
coef_diff_by_group(fit)

fit = nlme_fit(data)
coef_diff_by_group(fit)

# TODO: Add example for Stan fit typecast to class \code{breathtestfit} to compute
# confidence intervals instead of credible intervals
```

---

cum\_exp\_beta

*Cumulative exponential beta function*

---

**Description**

Equation (2), page 4 from Bluck, "Recent advances in the interpretation of the 13C octanoate breath test for gastric emptying"

**Usage**

```
cum_exp_beta(minute, dose, cf)
```

**Arguments**

minute	time in minutes
dose	in mg
cf	named vector of coefficients; only k and beta are required. Note that k is measured in 1/min (e.g. 0.01/min), while often it is quoted as 1/h (e.g. 0.6/h).

**Value**

Vector of predicted cumulative pdr

**See Also**

[exp\\_beta](#)

---

 dob\_to\_pdr

---

*Convert breath test DOB data to PDR data*


---

**Description**

Convert DOB (delta-over-baseline) to PDR for  $^{13}\text{C}$  breath test. This is equation (4) in Sanaka, Yamamoto, Tsutsumi, Abe, Kuyama (2005) Wagner-Nelson method for analysing the atypical double-peaked excretion curve in the [ $^{13}\text{C}$ ]-octanoate gastric emptying breath test in humans. Clinical and experimental pharmacology and physiology 32, 590-594.

**Usage**

```
dob_to_pdr(
  dob,
  weight = 75,
  height = 180,
  mw = 167,
  purity_percent = 99.1,
  mg_substrate = 100
)
```

**Arguments**

dob	Delta-over-baseline vector in 0/00
weight	Body weight in kg; assumed 75 kg if missing
height	Body height in cm; assume 180 cm if missing
mw	Molecular weight, 83.023388 g/mol for acetate, 167 g/mol for octanoate. Can also be given as string "acetate" or "octanoate".
purity_percent	Purity in percent
mg_substrate	Substrate in mg

**Value**

PDR percent dose/h

**Note**

I have no idea where the factor 10 in equation (4) comes from, possibly from percent(PDR)/and DOB(0/00). In Kim and Camillieri, Stable isotope breath test and gastric emptying, page 207, a factor of 0.1123 instead of 0.01123 is used, without the factor 10. Which one is correct?

**Examples**

```
filename = btcore_file("350_20049_0_GERWithWeight.txt")
bid = read_breathid(filename)
bid$data$pdr1 = dob_to_pdr(bid$data$dob, weight=bid$weight, height=bid$height)

plot(bid$data$minute, bid$data$pdr1, main="points: from breath_id; line: computed",
type="l")
points(bid$data$minute, bid$data$pdr, col="red", type="p", pch=16)
#
# Check how far our computed pdr is from the stored pdr
var(bid$data$pdr1-bid$data$pdr)
```

---

exp\_beta

*Exponential beta function for 13C breath data*


---

**Description**

Function to fit PDR time series data to exponential-beta function as given in:

Maes, B. D., B. J. Geypens, Y. F. Ghoois, M. I. Hiele, and P. J. Rutgeerts. 1998. 13C-Octanoic Acid Breath Test for Gastric Emptying Rate of Solids. *Gastroenterology* 114(4): 856-50

Sanaka M, Nakada K (2010) Stable isotope breath test for assessing gastric emptying: A comprehensive review. *J. Smooth Muscle Research* 46(6): 267-280

Bluck L J C and Coward W A 2006 Measurement of gastric emptying by the C-13-octanoate breath test — rationalization with scintigraphy *Physiol. Meas.* 27 279?89

For a review, see

Bluck LJC (2009) Recent advances in the interpretation of the 13C octanoate breath test for gastric emptying. *Journal of Breath Research*, 3 1-8

**Usage**

```
exp_beta(minute, dose, m, k, beta)
```

**Arguments**

minute	vector of time values in minutes
dose	in mg
m	efficiency
k	time constant
beta	form factor

**Details**

The function is defined as

```
exp_beta = function(minute,dose,m,k,beta) {
  m*dose*k*beta*(1-exp(-k*minute))^(beta-1)*exp(-k*minute)
}
```

At minute == 0, the function behaves like a polynomial with degree (beta-1).

**Value**

Values and gradients of estimated PDR for use with nls and nlme

**See Also**

In the example below, data and fit are plotted with standard R graphics. The S3 method [plot.breathtestfit](#) provides ggplot2 graphics.

**Examples**

```
start = list(m=20,k=1/100,beta=2)

# fit to real data set and show different t50 results
sample_file = btcore_file("350_20043_0_GER.txt")
# minute 0 must be removed to avoid singularity
breath_id = read_breathid(sample_file)
data = subset(breath_id$data, minute >0)
sample_nls = nls(pdr~exp_beta(minute, 100, m, k, beta), data = data, start = start)
data$pdr_fit_bluck=predict(sample_nls)
plot(data$minute, data$pdr, pch=16, cex=0.7, xlab="time (min)", ylab="PDR",
      main="t50 with different methods")
lines(data$minute,data$pdr_fit_bluck, col="blue")
t50 = t50_bluck_coward(coef(sample_nls))
t50_maes_ghoos = t50_maes_ghoos(coef(sample_nls))
t50scint = t50_maes_ghoos_scintigraphy(coef(sample_nls))
abline(v = t50, col = "red")
abline(v = t50_maes_ghoos, col = "darkgreen", lty = 2)
abline(v = breath_id$t50, col = "black", lty = 4)
abline(v = t50scint, col = "gray", lty = 3)
text(t50, 0, "Self-corrected Bluck/Coward", col = "red", adj = -0.01)
text(breath_id$t50, 0.5,"From BreathID device",col = "black", adj=-0.01)
text(t50scint, 1," Maes/Ghoos scintigraphic", col = "gray", adj = -0.01)
text(t50_maes_ghoos,1.5, "Classic Maes/Ghoos", col = "darkgreen", adj = -0.01)

# simulated data set
dose = 100
set.seed(4711)
# do not use minute 0, this gives singular gradients
# if required, shift minute = 0 by a small positive amount, e.g. 0.1
# create simulated data
pdr = data.frame(minute=seq(2, 200, by = 10))
```

```

pdr$pd_r =
  exp_beta(pdr$minute, 100, start$m, start$k, start$beta) + rnorm(nrow(pdr), 0, 1)
par(mfrow = c(1, 2))
# plot raw data
plot(pdr$minute, pdr$pd_r, pch=16, cex=0.5, xlab = "time (min)", ylab = "PDR")
# compute fit
pdr_nls = nls(pdr~exp_beta(minute, 100, m, k, beta), data = pdr, start = start)
# compute prediction
pdr$pd_rfit = predict(pdr_nls)
lines(pdr$minute, pdr$pd_rfit, col="red", lwd=2)

# plot cumulative
plot(pdr$minute, cum_exp_beta(pdr$minute, 100, coef(pdr_nls)), type="l",
      xlab = "time (min)", ylab = "cumulative PDR")
# show t50
t50 = t50_bluck_coward(coef(pdr_nls))
tlag = tlag_bluck_coward(coef(pdr_nls))
abline(v = t50, col = "gray")
abline(v = tlag, col = "green")
abline(h = 50, col = "gray")

# create simulated data from several patients
pdr1 = data.frame(patient = as.factor(letters[1:10]))
pdr1$m = start$m*(1 + rnorm(nrow(pdr1), 0, 0.1))
pdr1$k = start$k*(1 + rnorm(nrow(pdr1), 0, 0.3))
pdr1$beta = start$beta*(1 + rnorm(nrow(pdr1), 0, 0.1))
pdr1 = merge(pdr1, expand.grid(minute = seq(2, 200, by = 10),
  patient = letters[1:10]))
pdr1 = pdr1[order(pdr1$patient, pdr1$minute), ]

# simulated case: for patient a, only data up to 50 minutes are available
pdr1 = pdr1[!(pdr1$patient == "a" & pdr1$minute > 50),]
set.seed(4711)
pdr1$pd_r =
  with(pdr1, exp_beta(minute, 100, m, k, beta) + rnorm(nrow(pdr1), 0, 1))

# compute nls fit for patient a only: fails
# the following line will produce an error message

pdr_nls = try(nls(pdr~exp_beta(minute, 100, m, k, beta), data=pdr1, start=start,
  subset = patient=="a"))
stopifnot(class(pdr_nls) == "try-error")

# use nlme to fit the whole set with one truncated record
suppressPackageStartupMessages(library(nlme))
pdr_nlme = nlme(pdr~exp_beta(minute, 100, m, k, beta), data = pdr1,
  fixed = m+k+beta~1,
  random = m+k+beta~1,
  groups = ~patient,
  start = c(m = 20, k = 1/100, beta = 2))
coef(pdr_nlme)
pred_data = expand.grid(minute = seq(0, 400, 10), patient = letters[1:10])

```

```

pred_data$pdr = predict(pdr_nlme, newdata = pred_data)
suppressPackageStartupMessages(library(ggplot2))
ggplot() +
  geom_point(data = pdr1, aes(x = minute, y = pdr, color = "red")) +
  geom_line(data = pred_data, aes(x = minute, y = pdr), color = "black", linewidth = 1 ) +
  ggtitle("Short patient record 'a' gives a good fit with many missing data using nlme.\n
          Borrowing strength from nlme in action!") +
  facet_wrap(~patient) +
  theme(legend.position="none")

```

---

extract_id	<i>Extracts an ID from string IRIS CSV file</i>
------------	---

---

### Description

First tries to extract only digits, separating these by underscore when there are multiple blocks. If this give a non-valid id, returns the whole string without spaces and periods, hoping it makes sense. For internal use, but should be overridden for exotic IDs

### Usage

```
extract_id(id)
```

### Arguments

id                    One item from column Identifikation, e.g. "KEK-ZH-Nr.2013-1234"

### Examples

```
extract_id
```

---

nlme_fit	<i>Mixed-model nlme fit to 13C Breath Data</i>
----------	--

---

### Description

Fits exponential beta curves to 13C breath test series data using a mixed-model population approach. See <https://menne-biomed.de/blog/breath-test-stan/> for a comparison between single curve, mixed-model population and Bayesian methods.

### Usage

```

nlme_fit(
  data,
  dose = 100,
  start = list(m = 30, k = 1/100, beta = 2),
  sample_minutes = 15
)

```

**Arguments**

data	Data frame or tibble as created by <code>cleanup_data</code> , with mandatory columns <code>patient_id</code> , <code>group</code> , <code>minute</code> and <code>pdr</code> . It is recommended to run all data through <code>cleanup_data</code> to insert dummy columns for <code>patient_id</code> and <code>group</code> if the data are distinct, and report an error if not. At least 2 records are required for a population fit, but 10 or more are recommended to obtain a stable result.
dose	Dose of acetate or octanoate. Currently, only one common dose for all records is supported. The dose only affects parameter <code>m</code> of the fit; all important <code>t50</code> -parameters are unaffected by the dose.
start	Optional start values. In most case, the default values are good enough to achieve convergence, but slightly different values for <code>beta</code> (between 1 and 2.5) can save a non-convergent run.
sample_minutes	When the mean sampling interval is <code>&lt; sampleMinutes</code> , data are subsampled using a spline algorithm by function <code>subsample_data</code> . See the graphical output of <code>plot.breathtestfit</code> for an example where too densely sampled data of one patients were subsampled for the fit.

**Value**

A list of class ("breathtestnlmefit" "breathtestfit") with elements

**coef** Estimated parameters in a key-value format with columns `patient_id`, `group`, `parameter`, `stat`, `method` and `value`. Parameter `stat` currently always has value "estimate". Confidence intervals will be added later, so do not take for granted that all parameters are estimates. Has an attribute `AIC` which can be retrieved by the S3-function `AIC`.

**data** The data effectively fitted. If points are too closely sampled in the input, e.g. with `BreathId` devices, data are subsampled before fitting.

**See Also**

Base methods `coef`, `plot`, `print`; methods from package `broom`: `tidy`, `augment`.

**Examples**

```
d = simulate_breathtest_data(n_records = 3, noise = 0.7, seed = 4712)
data = cleanup_data(d$data)
fit = nlme_fit(data)
plot(fit) # calls plot.breathtestfit
options(digits = 3)
library(dplyr)
cf = coef(fit)
# The coefficients are in long key-value format
cf
# AIC can be extracted
AIC(fit)
# Reformat the coefficients to wide format and compare
# with the expected coefficients from the simulation
# in d$record.
cf %>%
```

```

filter(grepl("m|k|beta", parameter )) %>%
select(-method, -group) %>%
tidyr::spread(parameter, value) %>%
inner_join(d$record, by = "patient_id") %>%
select(patient_id, m_in = m.y, m_out = m.x,
        beta_in = beta.y, beta_out = beta.x,
        k_in = k.y, k_out = k.x)

```

---

nls\_fit

*Individual curve fit with nls to 13C breath test data*


---

## Description

Fits individual exponential beta curves to 13C breath test time series

## Usage

```
nls_fit(data, dose = 100, start = list(m = 50, k = 1/100, beta = 2))
```

## Arguments

data	Data frame or tibble as created by <a href="#">cleanup_data</a> , with mandatory columns <code>patient_id</code> , <code>group</code> , <code>minute</code> and <code>pdr</code> . It is recommended to run all data through <a href="#">cleanup_data</a> which will insert dummy columns for <code>patient_id</code> and <code>minute</code> if the data are distinct, and report an error if not.
dose	Dose of acetate or octanoate. Currently, only one common dose for all records is supported.
start	Optional start values <code>patient_id</code> and <code>group</code> .

## Value

A list of class ("breathtestnlsfit" "breathtestfit") with elements

**coef** Estimated parameters in a key-value format with columns `patient_id`, `group`, `parameter`, `stat`, `method` and `value`. Parameter `stat` always has value "estimate". Confidence intervals might be added later, so do not take for granted all parameters are estimates.

**data** Input data; `nls_fit` does not decimate the data. If you have large data sets where subsampling might be required to achieve faster convergence, using `nls_fit` anyway is only relevant to show how NOT to do it. Use `nlme_fit` or `stan_fit` instead.

## See Also

Base methods `coef`, `plot`, `print`; methods from package `broom`: `tidy`, `augment`.

**Examples**

```
d = simulate_breathtest_data(n_records = 3, noise = 0.2, seed = 4711)
data = cleanup_data(d$data)
fit = nls_fit(data)
plot(fit) # calls plot.breathtestfit
options(digits = 2)
cf = coef(fit)
library(dplyr)
cf %>%
  filter(grepl("m|k|beta", parameter )) %>%
  select(-method, -group) %>%
  tidyr::spread(parameter, value) %>%
  inner_join(d$record, by = "patient_id") %>%
  select(patient_id, m_in = m.y, m_out = m.x,
         beta_in = beta.y, beta_out = beta.x,
         k_in = k.y, k_out = k.x)
```

---

null\_fit

*Convert data to class breathtestfit*


---

**Description**

Does not change the data set, but returns a class suitable for plotting raw data with [plot.breathtestfit](#). See [read\\_any\\_breathtest](#) for an example.

**Usage**

```
null_fit(data, ...)
```

**Arguments**

data	Data frame or tibble as created by <a href="#">cleanup_data</a> , with mandatory columns patient_id, group, minute and pdr.
...	Not used

**Value**

A list of classes breathtestnullfit, breathtestfit with element data which contains the unmodified data.

---

plot.breathtestfit     *S3 plot method for breathtestfit*

---

### Description

Plots 13C data and fits.

### Usage

```
## S3 method for class 'breathtestfit'
plot(
  x,
  inc = 5,
  method_t50 = "maes_ghoos",
  linewidth = 1,
  point_size = NULL,
  ...
)
```

### Arguments

x	object of class <code>breathtestfit</code> , as returned by <code>nls_fit</code> , <code>nlme_fit</code> , <code>null_fit</code> or <code>stan_fit</code> ; <code>stan_fit</code> is in package <code>breathteststan</code> ,
inc	Increment for fitted curve plot in minutes
method_t50	Method for t50: "maes_ghoos", "bluck_coward" or "maes_ghoos_scintigraphy"
linewidth	optional line width; can improve look for printouts
point_size	optional point size; determined dynamically when NULL
...	other parameters passed to methods. Not used

### Examples

```
data = list(
  A = simulate_breathtest_data(n_records = 6, seed = 100),
  B = simulate_breathtest_data(n_records = 4, seed = 187)
)
# cleanup_data combines the list into a data frame
x = nls_fit(cleanup_data(data))
plot(x)
```

---

read\_any\_breathtest     *Read breathtest files of any format*

---

### Description

Uses `breathtest_read_function` to determine the file type and reads it if it has a valid format.

### Usage

```
read_any_breathtest(files)
```

### Arguments

`files`                 A single filename, a list or a character vector of filenames.

### Value

A list of `breathtest_data`, even if only one file was passed. The list can be passed to `cleanup_data` to extract one concatenated data frame for processing with `nls_fit`, `nlme_fit`, `null_fit` (no processing) or `stan_fit` in separate package `breathteststan`.

### Examples

```
files = c(
  group_a = btcore_file("IrisCSV.TXT"),
  group_a = btcore_file("350_20043_0_GER.txt"),
  group_b = btcore_file("IrisMulti.TXT"),
  group_b = btcore_file("NewBreathID_01.xml")
)
bt = read_any_breathtest(files)
str(bt, 1)
# Passing through cleanup_data gives a data frame/tibble
bt_df = cleanup_data(bt)
str(bt_df)
# If you want data only, use null_fit()
plot(null_fit(bt_df))
# Plot population fit with decimated data
plot(nlme_fit(bt_df))
```

---

read\_breathid             *Read BreathID file*

---

### Description

Reads 13c data from a BreathID file, and returns a structure of class `breathtest_data`.

**Usage**

```
read_breathid(filename = NULL, text = NULL)
```

**Arguments**

filename	name of txt-file to be read
text	alternatively, text can be given as string

**Value**

Structure of class [breathtest\\_data](#)

**Examples**

```
filename = btcore_file("350_20043_0_GER.txt")
# Show first lines
cat(readLines(filename, n = 10), sep="\n")
#
bid = read_breathid(filename)
str(bid)
```

---

read_breathid_xml	<i>Read new BreathID/Examens XML file</i>
-------------------	---

---

**Description**

Reads 13c data from an XML BreathID file, and returns a structure of class `breathtest_data_list`, which is a list with elements of class `breathtest_data`.

**Usage**

```
read_breathid_xml(filename = NULL, text = NULL)
```

**Arguments**

filename	name of xml-file to be read
text	alternatively, text can be given as string

**Value**

List of class `breathtest_data_list` of structures of class [breathtest\\_data](#); an XML file can contain multiple data sets. Errors string for individual records are returned as attribute "errors".

**Examples**

```

filename = btcore_file("NewBreathID_01.xml")
# Show first lines
cat(readLines(filename, n = 10), sep="\n")
bid = read_breathid_xml(filename)
# List with length 1
str(bid, 1)
filename = btcore_file("NewBreathID_multiple.xml")
bids = read_breathid_xml(filename)
str(bids, 1) # 3 elements - the others in the file have no data
# Create hook function to deselect first record
choose_record = function(records) {
  r = rep(TRUE, length(records))
  r[1] = FALSE
  r
}
options(breathtestcore.choose_record = choose_record)
bids = read_breathid_xml(filename)
str(bids, 1) # 2 elements, first deselected

```

---

read\_breathtest\_excel *Reads breathtest data in Excel format*

---

**Description**

Can read several formats of data sets in Excel, from 2 (minute, pdr or dob for 1 record) to 4 columns (patient\_id, group, minute, pdr or dob). Conversion from dob to pdf is done for assuming 180 cm height and 75 kg weight. See the example below with several sheets for supported formats

**Usage**

```
read_breathtest_excel(filename, sheet = 1)
```

**Arguments**

filename	Name of Excel-file to be read
sheet	Name or number of Excel file to be read. When used with <a href="#">read_any_breathtest</a> , the first sheet is always read. You must call <code>read_breathtest_excel</code> explicitly to read other worksheets, as shown in the example below.

**Value**

Different from the other readXXX function, this returns a list with a data frame, not a structure of [breathtest\\_data](#). Pass result through [cleanup\\_data](#) to make it compatible with other formats.

**Examples**

```

filename = btcore_file("ExcelSamples.xlsx")
sheets = readxl::excel_sheets(filename)
# First 4 lines of each sheet
for (sheet in sheets) {
  cat("\nSheet ", sheet, "\n")
  ex = readxl::read_excel(filename, sheet = sheet, n_max = 4)
  print(ex)
}
# To get consistently formatted data from a sheet
bt_data = read_breathtest_excel(filename, sheets[6])
# 3 columns
str(bt_data)
bt_cleaned = cleanup_data(bt_data)
# 4 columns standard format
str(bt_cleaned)

```

---

read\_iris

*Read 13C data from IRIS/Wagner Analysen*


---

**Description**

Reads composite files with 13C data from IRIS/Wagner Analysen. The composite files start as follows:

```

"Testergebnis"
"Nummer", "1330"
"Datum", "10.10.2013"
"Testart"

```

**Usage**

```
read_iris(filename = NULL, text = NULL)
```

**Arguments**

filename	name of IRIS/Wagner file in composite format
text	alternatively, text can be given as string

**Value**

List of class `breathtest_data` with `file_name`, `patient_name`, `patient_first_name`, `test`, `identifikation`, and data frame data with time and dob

## Examples

```
filename = btcore_file("IrisMulti.TXT")
cat(readLines(filename, n = 10, encoding = "latin1"), sep="\n")
#
iris_data = read_iris(filename)
str(iris_data)
```

---

read\_iris\_csv

*Read 13C data from IRIS/Wagner Analysen in CSV Format*

---

## Description

Reads 13C data from IRIS/Wagner Analysen in CSV Format The CSV files start as follows:

```
"Name", "Vorname", "Test", "Identifikation"
```

This format does not have information about the substrate (acetate, octanoate), the dose and body weight and height. The following defaults are used: substrate = acetate, dose = 100, weight = 75, height = 180.

## Usage

```
read_iris_csv(filename = NULL, text = NULL)
```

## Arguments

filename	Name of IRIS/Wagner file in CSV format
text	alternatively, text can be given as string

## Value

List of class `breath_test_data` with file name, patient name, patient first name, test, identifikation, and data frame data with time and dob

## Examples

```
filename = btcore_file("IrisCSV.TXT")
cat(readLines(filename, n = 3, encoding = "latin1"), sep="\n")
#
iris_data = read_iris_csv(filename)
str(iris_data)
```

---

`sigma.breathtestnlmefit`*S3 method to extract the fit's residual standard deviation*

---

**Description**

Functions for nls and nlme are available; additional functions for Stan-based fits are defined in package `breathteststan`.

**Usage**

```
## S3 method for class 'breathtestnlmefit'  
sigma(object, ...)
```

**Arguments**

<code>object</code>	Result of class <code>breathtestfit</code>
<code>...</code>	Not used

**Value**

A numeric value giving the standard deviation of the residuals.

---

`simulate_breathtest_data`*Simulate 13C breath time series data*

---

**Description**

Generates simulated breath test data, optionally with errors. If none of the three standard deviations `m_std`, `k_std`, `beta_std` is given, an empirical covariance matrix from USZ breath test data is used. If any of the standard deviations is given, default values for the others will be used.

**Usage**

```
simulate_breathtest_data(  
  n_records = 10,  
  m_mean = 40,  
  m_std = NULL,  
  k_mean = 0.01,  
  k_std = NULL,  
  beta_mean = 2,  
  beta_std = NULL,  
  noise = 1,  
  cov = NULL,
```

```

    student_t_df = NULL,
    missing = 0,
    seed = NULL,
    dose = 100,
    first_minute = 5,
    step_minute = 15,
    max_minute = 155
  )

```

### Arguments

n_records	Number of records
m_mean, m_std	Mean and between-record standard deviation of parameter m giving metabolized fraction.
k_mean, k_std	Mean and between-record standard deviation of parameter k, in units of 1/minutes.
beta_mean, beta_std	Mean and between-record standard deviations of lag parameter beta
noise	Standard deviation of normal noise when student_t_df = NULL; scaling of noise when student_t_df >= 2.
cov	Covariance matrix, default NULL, i.e. not used. If given, overrides standard deviation settings.
student_t_df	When NULL (default), Gaussian noise is added; when >= 2, Student_t distributed noise is added, which generates more realistic outliers. Values from 2 to 5 are useful, when higher values are used the result comes close to that of Gaussian noise. Values below 2 are truncated to 2.
missing	When 0 (default), all curves have the same number of data points. When > 0, this is the fraction of points that were removed randomly to simulate missing
seed	Optional seed; not set if seed = NULL (default)
dose	Octanoate/acetate dose, almost always 100 mg, which is also the default
first_minute	First sampling time. Do not use 0 here, some algorithms do not converge when data near 0 are passed.
step_minute	Inter-sample interval for breath test
max_minute	Maximal time in minutes.

### Value

A list of class simulated\_breathtest\_data with 2 elements:

**record** Data frame with columns patient\_id(chr), m, k, beta, t50 giving the effective parameters for the individual patient record.

**data** Data frame with columns patient\_id(chr), minute(dbl), pdr(dbl) giving the time series and grouping parameters.

A comment is attached to the return value that can be used as a title for plotting.

**Examples**

```

library(ggplot2)
pdr = simulate_breathtest_data(n_records = 4, seed = 4711, missing = 0.3,
  student_t_df = 2, noise = 1.5) # Strong outliers
#
str(pdr, 1)
#
pdr$record # The "correct" parameters
#
# Explicit plotting
ggplot(pdr$data, aes(x = minute, y = pdr)) + geom_point() +
  facet_wrap(~patient_id) + ggtitle(comment(pdr$data))
#
# Or use cleanup_data and null_fit for S3 plotting
plot(null_fit(cleanup_data(pdr$data)))

```

---

subsample_data	<i>Decimate densely sampled 13C time series</i>
----------------	---

---

**Description**

When data of a record are more closely spaced than `sample_minutes`, these are spline-subsampled to `sample_minutes`. In the region of the initial slope, i.e. the initial fifth of the time, the record is sampled more densely. Too dense sampling leads to non-convergent nlme fits and to long runs with Stan-based fits. The function is used internally by function `link{nlme_fit}` in package `breathtestcore` and is exported for use by package `breathteststan`.

**Usage**

```
subsample_data(data, sample_minutes)
```

**Arguments**

<code>data</code>	Data frame with columns <code>patient_id</code> , <code>group</code> , <code>minute</code> , <code>pdr</code> .
<code>sample_minutes</code>	Required average density. When points are more closely spaced, data are sub-sampled. No upsampling occurs when data are more sparse.

---

<code>t50_bluck_coward</code>	<i>Bluck-Coward self-corrected half-emptying time</i>
-------------------------------	---

---

**Description**

Uses Newton's method to solve the self-corrected Bluck-Coward equation for  $1/2$  to compute the half-emptying time `t_50`.

See also equation  $G(n,t)$  in

Bluck LJC, Jackson S, Vlasakakis G, Mander A (2011) Bayesian hierarchical methods to interpret the 13C-octanoic acid breath test for gastric emptying. *Digestion* 83\_96-107, page 98.

**Usage**

```
t50_bluck_coward(cf)
```

**Arguments**

`cf` Named vector of coefficients; only `k` and `beta` are required. In this package, `k` is measured in units of 1/min (e.g. 0.01/min), in publications it is often quoted as 1/h (e.g. 0.6/h).

**Value**

Time where value is 1/2 of the maximum, i.e. `t_50` or `t_1/2` in minutes; in the publication by Bluck et al, the parameter is called `t_1/2(in)`.

**See Also**

[exp\\_beta](#)

**Examples**

```
# From table 3 and 4 in Bluck et al.; values for \code{k} and \code{beta}
# (nls, bayesian) are entered and checked against the tabulated values of
# t_{1/2(in)}.
# Most errors are small, but there are some outliers; errors in paper table?
# Parameters and Bluck et al. results:
# table 3 of Bluck et al.
cf3 = data.frame(
  method = rep(c("nls", "bayesian")),
  group = rep(c("lean", "obese"), each=2),
  k = c(0.576, 0.606, 0.529, 0.608),
  beta = c(5.24, 5.79, 5.95, 7.54),
  t12 = c(3.67, 3.63, 4.23, 3.99),
  t12in = c(2.076, 2.110, 2.422, 2.466),
  tlag = c(2.88, 2.88, 3.34, 3.26),
  tlagin = c(1.632, 1.724, 1.92, 2.101)
)
cf3 = dplyr::mutate(cf3,
  t50_maes_ghoos = t50_maes_ghoos(cf3),
  t50_bluck_coward = t50_bluck_coward(cf3),
  tlag_maes_ghoos = tlag_maes_ghoos(cf3),
  tlag_bluck_coward = tlag_bluck_coward(cf3),
  err_t50_maes_ghoos = round(100*(t50_maes_ghoos-t12)/t12, 2),
  err_t50_bluck_coward =
    round(100*(t50_bluck_coward-t12in)/t12in, 2),
  err_lag_maes = round(100*(tlag_maes_ghoos-tlag)/tlag, 2),
  err_lag_bluck_coward =
    round(100*(tlag_bluck_coward-tlagin)/tlagin, 2)
)
cf3
# table 4
# there are large differences for mj3, both using the bayesian (26%)
```

```
# and the nls method (16%). The other data are within the expected limits
cf4 = data.frame(
  method = rep(c("nls", "bayesian"),each=3),
  group = rep(c("mj1", "mj2", "mj3")),
  k = c(0.585, 0.437, 0.380, 0.588, 0.418, 0.361),
  beta=c(4.35, 4.08, 4.44, 4.49, 4.30, 4.29),
  t12 = c(3.39, 4.25, 4.82, 3.40, 4.61, 5.09),
  t12in = c(1.77, 2.16, 2.19, 1.81, 2.34, 2.43),
  tlag = c(2.56, 3.17, 3.39, 2.58, 3.40, 3.62),
  tlagin = c(1.30, 1.53, 1.33, 1.35, 1.65, 1.57)
)
cf4 = dplyr::mutate(cf4,
  t50_maes_ghoos = t50_maes_ghoos(cf4),
  t50_bluck_coward = t50_bluck_coward(cf4),
  tlag_maes_ghoos = tlag_maes_ghoos(cf4),
  tlag_bluck_coward = tlag_bluck_coward(cf4),
  err_t50_maes_ghoos = unlist(round(100*(t50_maes_ghoos-t12)/t12)),
  err_t50_bluck_coward =
    round(100*(t50_bluck_coward-t12in)/t12in,2),
  err_lag_maes = round(100*(tlag_maes_ghoos-tlag)/tlag,2),
  err_lag_bluck_coward =
    round(100*(tlag_bluck_coward-tlagin)/tlagin,2)
)
cf4
```

---

t50\_maes\_ghoos

*Half-emptying time by Maes/Ghoos method*


---

## Description

Half-emptying time t50 as determined from the fit of a beta exponential function. In the Maes/Ghoos model, it is defined as the time when the area under curve (AUC) is 50% of the AUC from 0 to infinity.

Maes B D, Ghoos Y F, Rutgeerts P J, Hiele M I, Geypens B and Vantrappen G 1994 Dig. Dis. Sci. 39 S104-6.

## Usage

```
t50_maes_ghoos(cf)
```

## Arguments

cf                    named vector of coefficients; only k and beta are required note that k is measured in 1/min (e.g. 0.01/min), usually it is quoted as 1/h (e.g. 0.6/h).

## Value

Time in minutes when area under curve is 50% of the AUC to infinity. In the Maes/Ghoos model, this is used as a surrogate for gastric emptying half time t50.

**See Also**

[exp\\_beta](#), and [t50\\_bluck\\_coward](#) for an example.

**Examples**

```
# Integral from 0 to infinity is 100 at dose 100 mg
integrate(exp_beta, 0, Inf, beta = 1.5, k = 0.01, m = 1, dose = 100)
t50_mg = t50_maes_ghoos(c(beta = 1.5, k = 0.01, dose = 100))
t50_mg
# Integral to half-emptying time \code{t50_maes_ghoos} is 50
integrate(exp_beta, 0, t50_mg, beta = 1.5, k = 0.01, m = 1, dose = 100)
```

---

t50\_maes\_ghoos\_scintigraphy

*Half-emptying time t50 from Maes/Ghoos fit with scintigraphic correction*

---

**Description**

Half-emptying time t50 in minutes from beta exponential function fit, with linear and rather arbitrary correction for scintigraphic values. This is given for comparison with published data only; there is little justification to use it, even if it is closer to real gastric emptying times as determined by MRI or scintigraphy. Ghoos YF, Maes BD, Geypens BJ, Mys G, Hiele MI, Rutgeerts PJ, Vantrappen G. Measurement of gastric emptying rate of solids by means of a carbon-labeled octanoic acid breath test. *Gastroenterology*. 1993;104:1640-1647.

**Usage**

```
t50_maes_ghoos_scintigraphy(cf)
```

**Arguments**

cf                    named vector of coefficients; only k and beta are required

**Value**

Time where value is 1/2 of maximum, i.e. t50 in minutes.

**See Also**

[exp\\_beta](#), and [t50\\_bluck\\_coward](#) for an example.

---

tidy.breathtestfit      *Broom-style tidying methods for breathtestfit*

---

## Description

Broom-method `tidy` to streamline the results of class `breathtestfit` as generated by `nls_fit` or `nlme_fit`. Returns the fit coefficients and half-emptying time `t50` with the Maes/Ghoos method; additional parameters should be extracted with `coef`.

## Usage

```
## S3 method for class 'breathtestfit'  
tidy(x, ...)
```

## Arguments

<code>x</code>	Object of class <code>breathtestfit</code>
<code>...</code>	other parameters passed to methods

## Value

A tibble/data frame with columns

**patient\_id** Patient Id (character)

**group** Treatment or patient group (character)

**m** Fraction metabolized

**k** Time constant (1/minutes)

**beta** The so-called lag parameters, no dimension

**t50** Emptying half time in minutes as calculated following Maes/Ghoos

## See Also

[tidy](#)

## Examples

```
library(broom)  
# Generate simulated data  
data = cleanup_data(simulate_breathtest_data())$data  
# Fit with the population method  
fit = nlme_fit(data)  
# Output coefficients  
tidy(fit)  
# All coefficients in the long form  
coef(fit)
```

---

tlag_bluck_coward	<i>Lag phase for Bluck-Coward self-correcting fit</i>
-------------------	---

---

**Description**

This parameter is probably not very useful, as it can be negative

**Usage**

```
tlag_bluck_coward(cf)
```

**Arguments**

cf	named vector of coefficients; only k and beta are required. Note that in this package, k is measured in 1/min (e.g. 0.01/min), while in the literature is often quoted as 1/h (e.g. 0.6/h).
----	---

**Value**

Lag phase in minutes (time t at which the maximum in the rate of change of g(t) occurs)

**See Also**

[exp\\_beta](#), and [t50\\_bluck\\_coward](#) for an example.

---

tlag_maes_ghoos	<i>So-called lag time from Maes/Ghoos fit</i>
-----------------	---

---

**Description**

Computes tlag from uncorrected fit to the beta exponential function. The name tlag is a misnomer; it simply is the maximum of the PDR curve, so in papers by Bluck et al. it is renamed to t\_max.

Maes B D, Ghoos Y F, Rutgeerts P J, Hiele M I, Geypens B and Vantrappen G 1994 Dig. Dis. Sci. 39 S104-6.

**Usage**

```
tlag_maes_ghoos(cf)
```

**Arguments**

cf	named vector of coefficients; only k and beta are required k is measured in 1/min (e.g. 0.01/min).
----	--

**Value**

Lag time as defined from Maes/Ghoos fit

**See Also**

[exp\\_beta](#), and [t50\\_bluck\\_coward](#) for an example.

---

usz\_13c

*Zurich sample set of 13C breath test data*


---

**Description**

13C time series PDR data from normals and random patients from the division of [Gastroenterology and Hepatology, University Hospital Zurich](#). Most breath samples from normals were collected with bags and analyzed by [IRIS/Wagner](#) infrared spectroscopy. Patient samples were recorded with the continuous monitoring system [BreathID](#).

**patient\_id** Patient identifier, starting with norm for normals (healthy volunteers) and pat for patients. Note that for normals there are two records for each subject, so only the combination of patient\_id and group is a unique identifier of the time series record.

**group** liquid\_normal for normals and liquid meal, solid\_normal normals and solid meal, and patient for patients from the University Hospital of Zurich.

**minute** Time in minutes

**pdr** PDR as computed by breathtest device or from dob via function dob\_to\_pdr

**Usage**

```
data(usz_13c)
```

**Format**

A data frame with 15574 rows and 4 variables

**Examples**

```
data(usz_13c)
## Not run:
str(usz_13c)
# Plot all records; this needs some time
pdf(file.path(tempdir(), "usz_13c.pdf"), height= 30)
# null_fit makes data plotable without fitting a model
plot(null_fit(usz_13c))
dev.off()

## End(Not run)
# Plot a subset
suppressPackageStartupMessages(library(dplyr))
```

```
usz_part = usz_13c %>%
  filter(patient_id %in% c("norm_001", "norm_002", "pat_001", "pat_002"))
plot(null_fit(usz_part))
```

---

usz\_13c\_a

*Exotic 13C breath test data*


---

### Description

13C time series PDR data from three different groups in a randomized (= not-crossover) design. This are unpublished data from [Gastroenterology and Hepatology, University Hospital Zurich](#).

Data are formatted as described in [usz\\_13c](#). These time series present a challenge for algorithms.

### Usage

```
data(usz_13c_a)
```

### Examples

```
library(dplyr)
library(ggplot2)
data(usz_13c_a)
d = usz_13c_a %>%
  cleanup_data() %>% # recommended to test for validity
  nlme_fit()
plot(d)
```

---

usz\_13c\_d

*13C breath test data with MRI emptying for comparison*


---

### Description

13C time series PDR data from normals and three different meals in a cross-over design from the division of [Gastroenterology and Hepatology, University Hospital Zurich](#). See [Kuyumcu et al., Gastric secretion does not affect...](#)

Data are formatted as described in [usz\\_13c](#). In addition, half emptying times from MRI measurements are attached to the data as attribute `mr_i_t50`. The example below shows how to analyze the data and present half emptying times from MRI and 13C in diagrams.

### Usage

```
data(usz_13c_d)
```

**Examples**

```

library(dplyr)
library(ggplot2)
data(usz_13c_d)
mri_t50 = attr(usz_13c_d, "mri_t50")
d = usz_13c_d %>%
  cleanup_data() %>% # recommended to test for validity
  nlme_fit()
plot(d) +
  geom_vline(data = mri_t50, aes(xintercept = t50), linetype = 2)

# Maes-Ghoos t50
dd = mri_t50 %>%
  inner_join(
    coef(d) %>% filter(parameter=="t50", method == "maes_ghoos"),
    by = c("patient_id", "group")) %>%
  mutate(
    t50_maes_ghoos = value
  )

ggplot(dd, aes(x=t50, y = t50_maes_ghoos, color = group)) +
  geom_point() +
  facet_wrap(~group) +
  geom_abline(slope = 1, intercept = 0) +
  xlim(45,205) +
  ylim(45,205)

# Bluck-Coward t50
dd = mri_t50 %>%
  inner_join(
    coef(d) %>% filter(parameter=="t50", method == "bluck_coward"),
    by = c("patient_id", "group")) %>%
  mutate(
    t50_bluck_coward = value
  )

ggplot(dd, aes(x=t50, y = t50_bluck_coward, color = group)) +
  geom_point() +
  facet_wrap(~group) +
  geom_abline(slope = 1, intercept = 0) +
  xlim(0,205) +
  ylim(0,205)

```

# Index

## \* datasets

usz\_13c, [35](#)  
usz\_13c\_a, [36](#)  
usz\_13c\_d, [36](#)

AIC.breathtestnlmefit, [3](#)  
augment, [3](#), [4](#)  
augment.breathtestfit, [3](#)

breathtest\_data, [4](#), [8](#), [22–25](#)  
breathtest\_read\_function, [4](#), [6](#), [22](#)  
btcore\_file, [7](#)

cleanup\_data, [4](#), [7](#), [18–20](#), [22](#), [24](#)  
coef, [33](#)  
coef.breathtestfit, [9](#)  
coef\_by\_group, [10](#)  
coef\_diff\_by\_group, [11](#)  
cum\_exp\_beta, [12](#)

dob\_to\_pdr, [5](#), [13](#)

exp\_beta, [13](#), [14](#), [30](#), [32](#), [34](#), [35](#)  
extract\_id, [17](#)

nlme\_fit, [3](#), [9–11](#), [17](#), [21](#), [22](#)  
nls\_fit, [3](#), [7](#), [9–11](#), [19](#), [21](#), [22](#)  
null\_fit, [20](#), [21](#), [22](#)

plot.breathtestfit, [15](#), [18](#), [20](#), [21](#)

read\_any\_breathtest, [6](#), [8](#), [20](#), [22](#), [24](#)  
read\_breathid, [22](#)  
read\_breathid\_xml, [4](#), [8](#), [23](#)  
read\_breathtest\_excel, [24](#)  
read\_iris, [25](#)  
read\_iris\_csv, [26](#)

sigma.breathtestnlmefit, [27](#)  
simulate\_breathtest\_data, [27](#)  
stan\_fit, [10](#)

subsample\_data, [18](#), [29](#)

t50\_bluck\_coward, [29](#), [32](#), [34](#), [35](#)  
t50\_maes\_ghoos, [31](#)  
t50\_maes\_ghoos\_scintigraphy, [32](#)  
tidy, [33](#)  
tidy.breathtestfit, [33](#)  
tlag\_bluck\_coward, [34](#)  
tlag\_maes\_ghoos, [34](#)

usz\_13c, [35](#), [36](#)  
usz\_13c\_a, [36](#)  
usz\_13c\_d, [36](#)