# Package 'flux'

June 26, 2022

**Type** Package

**Title** Flux Rate Calculation from Dynamic Closed Chamber Measurements

**Version** 0.3-0.1

**Date** 2014-04-23

**Author** Gerald Jurasinski, Franziska Koebsch, Anke Guenther, Sascha Beetz

**Maintainer** Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

**Depends** R (>= 2.12.0), caTools

**Description** Functions for the calculation of greenhouse gas flux rates from closed chamber concentration measurements. The package follows a modular concept: Fluxes can be calculated in just two simple steps or in several steps if more control in details is wanted. Additionally plot and preparation functions as well as functions for modelling gpp and reco are provided.

**License** GPL-2

**LazyLoad** yes

**Encoding** UTF-8

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-06-26 06:58:11 UTC

## R topics documented:

---

`flux-package` *Flux rate estimation with dynamic closed chamber data*

---

## Description

Several functions for the estimation of greenhouse gas (GHG) flux rates using closed chamber concentration measurements. The package follows a modular concept: Fluxes can be calculated in just two simple steps or in several steps if more control is wanted. Functions for further analyses (GPP and Reco model fitting and prediction for budgets including error terms) are also available.

## Details

| | |
|---|---|
| Package: | flux |
| Type: | Package |
| Version: | 0.3-0 |
| Date: | 2014-04-23 |
| License: | GPL-2 |
| LazyLoad: | yes |

Obtain flux rates from many chamber measurements within minutes. After preparing the read in data (Field or device measured concentration data on the three most prominent greenhouse gases) with `chop` just run `flux` or `fluxx` (for medium frequency data) on the result returned by `chop` and get flux rates in an easy to interpret table including quality flags. Plot diagnostic plots as pdf per factor level to a folder or simply to the screen. Use `gpp` to model GPP and `reco` to model ecosystem respiration or use `gpp.bulk` to bulk model GPP and `reco.bulk` to bulk model $R_{eco}$

and use the resulting objects with budget.gpp and budget.reco, respectively, to predict fluxes using continuously logged data. Use budget.ie to estimate the uncertainty associated with the interpolation between models. Several helper functions for ghg analysis are also provided.

## Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>, Franziska Koebsch <franziska.koebsch@uni-rostock.de>, Ulrike Hagemann <ulrike.hagemann@zalf.de>, Anke Günther <anke.guenther@uni-rostock.de>

Maintainer: Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

## References

Nakano T (2004) A comparison of regression methods for estimating soil- atmosphere diffusion gas fluxes by a closed-chamber technique. Soil Biology and Biochemistry 36: 107-113.

Forbrich I, Kutzbach L, Hormann A, Wilmking M (2010) A comparison of linear and exponential regression for estimating diffusive CH4 fluxes by closed- chambers in peatlands. Soil Biology and Biochemistry 42: 507-515.

Beetz S, Liebersbach H, Glatzel S, Jurasinski G, Buczko U, Hoper H (2013) Effects of land use intensity on the full greenhouse gas balance in an Atlantic peat bog. Biogeosciences 10:1067-1082.

Koebsch F, Glatzel S, Jurasinski G (2013) Vegetation controls methane emissions in a coastal brackish fen. Wetlands Ecology and Management 21:323–337.

## See Also

HMR for a different approach to flux rate estimation from chamber data.

---

# amc *Climate station data from 2009 to 2011 in the Ahlenmoor peat bog, Northeast Germany*

---

## Description

Climatic variables measured from 2009 to 2011 in the Ahlenmoor peat bog, Northeast Germany as part of a closed chamber measurement study on GHG exchange.

## Usage

```
data(amc)
```

## Format

A data frame with 43197 observations on the following 8 variables.

`date` Factor giving the date of field sampling, format is "%Y-%m-%d".

`time` Factor giving the time of measurement in the field, format is "%H:%M:%S".

`t.air` Numeric. Average air temperatures in °C

`t.soil2` Numeric. Average soil temperatures in 2cm depth in °C

`t.soil5` Numeric. Average soil temperatures in 5cm depth in °C

`t.soil10` Numeric. Average soil temperatures in 10cm depth in °C

`PAR` Numeric. Average half hourly photosynthetically active radiation during the flux measurement. Actually represents PPFD (photon flux density) in micromole per sqm and second

`timestamp` POSIXlt representing the date and time for the measurements

## Source

Diss Sascha

## References

Beetz (2014) ...

## Examples

```
data(amc)
```

---

`amd` *Closed chamber fluxes from 2009 to 2011 in the Ahlenmoor peat bog, Northeast Germany*

---

## Description

CO2 exchange rates determined with closed chamber measurements and cooresponding measurements of temperatures, photosynthetically actiove radiation, and other variables in the Ahlen-Falkenberger Moor peat bog complex from 2009 to 2011 on one specific plot (3 replicates) ....

## Usage

```
data(amd)
```

## Format

A data frame with 559 observations on the following 14 variables.

`timestamp` POSIXlt representing the date and time for the measurements

`campaign` Numeric. Representing IDs of the measurement campaigns, i.e. data sharing on campaign were acquired within short time period (typically one day but sometimes also two consecutive days)

`plot` Numeric. Representing the field plot numbers

`kind` Character vector giving the kind of chamber measurements. Either "D" for dark (opaque) chamber measurements (i.e., $R_{eco}$ measurements) or "T" for transparent chamber measurements (i.e., NEE measurements)

`flux` Numeric. The estimated flux rate. CO2 exchange in micromole per sqm and second

`PAR` Numeric. Average photosynthetically active radiation during the flux measurement. Actually represents PPFD (photon flux density) in micromole per sqm and second

`t.air` Numeric. Average air temperature during chamber measurement in °C

`t.soil2` Numeric. Average soil temperature in 2cm depth during chamber measurement in °C

`t.soil5` Numeric. Average soil temperature in 5cm depth during chamber measurement in °C

`t.soil10` Numeric. Average soil temperature in 10cm depth during chamber measurement in °C

`n.meas` Numeric giving the number of concentration measurements that were available to estimate the flux

`duration` Factor with 282 levels giving the duration of measurement in the field, format is "%H:%M"

`r.squared` Numeric. The R2s of the linear regression models that were fit to the concentration data to estimate the fluxes (with `fluxx`)

`sigmif` Numeric. The significance of the linear regression models

## Source

Saschas Doktorarbeit

## References

Beetz S (2014) ???

## Examples

```
data(amd)
```

---

`append.df` *Append a data.frame to another including consistency checks*

---

## Description

Often ghg concentration data come in chunks. This function provides a wrapper for appending data.

## Usage

```
append.df(orig, add)
```

## Arguments

| | |
|---|---|
| `orig` | A data.frame |
| `add` | Another data.frame |

## Details

The two data.frames are appended based on common columns. A warning is issued if some column names do not match. New columns are silently added.

## Value

Data.frame

## Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

## Examples

```
## add later
```

---

auc *Calculate the area under a line(curve).*

---

## Description

Calculates the **a**rea **u**nder a **c**urve (integral) following the trapezoid rule. With `auc.mc` several Monte Carlo methods can be applied to obtain error terms for estimating the interpolation error for the integration.

## Usage

```
auc(x, y, thresh = NULL, dens = 100, sort.x = TRUE)

auc.mc(x, y, method = "leave out", lo = 2, it = 100, ...)
```

## Arguments

| | |
|---|---|
| `x` | Numerical vector giving the x cordinates of the points of the line (curve). |
| `y` | Numerical vector giving the y cordinates of the points of the line (curve). One can calculate the integral of a fitted line through giving a vector to `x` that spans `xlim` with small intervals and predicting the y coordinates with `predict` and that `x`-vector as `newdata`. See example. |
| `thresh` | Threshold below which area is not calculated. Can be used to deal with unrealistically low flux data. By default `thresh` is set to `NULL` and therefore the complete area below the zero line is subtracted from the area above the zero line to integrate the area under the curve. When data below a certain value make no sense for your question, you are able to set `thresh`. Then, all y-values below `thresh` are set to the value of `thresh` and the regarding areas below `thresh` are not subtracted from the total area. |
| `dens` | By default the data density is artificially increased by adding 100 data points between given adjacent data points. These additional data points are calculated by linear interpolation along x and y. When a threshold is set, this procedure increases the accuracy of the result. Setting `dens` has no effect on the result when `thresh` is set to `NULL`. |

| | |
|---|---|
| sort.x | By default the vectors in x and y are ordered along increasing x because integration makes no sense with unordered data. You can override this by setting sort.x = FALSE |
| method | Specify how interpolation error should be estimated. Available methods include "leave out", "bootstrap", "sorted bootstrap", "constrained bootstrap", "jackknife", "jack-validate". True bootstrap is only effective when sort.x = FALSE. See details. |
| lo | When estimating interpolation error with "leave out" or "jack-validate", how many data points should be left out randomly? Defaults to 2. See method and details. |
| it | How many iterations should be run when using auc.mc to estimate the interpolation error. Defaults to 100. |
| ... | Any arguments passed through to auc. |

## Details

During integration the underlying assumption is that values can be interpolated linearly between adjacent data points. In many cases this is questionable. For estimating the linear interpolation error from the data at hand one may use Monte Carlo resampling methods. In auc.mc the following approaches are available:

- leave out: In each run lo data points are randomly omitted. This is quite straightforward, but the number of data points left out (lo) is arbitrary and thus the error terms estimated with this approach may be hardly defensible.
- bootstrap: Data are bootstrapped (sampling with replacement). Thus, some data points may repeat whereas others may be omitted. Due to the random sampling the order of data points is changed which may be unwanted with times series and may produce largely exaggerated error terms. This is only effective if sort.x = FALSE.
- sorted bootstrap: Same as before but ordering along x after bootstrapping may cure some problems of changed order. However, due to repeated data points time series spreading seasons but having data showing distinct seasonality may still be misrepresented.
- constrained bootstrap: Same as before but after ordering repeated data points are omitted. Thus, this equals leaving some measurements out at each run with a random number of leave outs. Numbers of leave outs typically show normal distribution around 3/4n.
- jackknife: auc is calculated for all possible combinations of length(x)-1 data points. Depending on length(x) the number of combinations can be quite low.
- jack-validate: auc is calculated for all possible combinations of (length(x)-lo) : (length(x)-1) data points. Partly cures the "arbitrarity" problem of the leave out approach and produces stable summary statistics.

## Value

auc returns a numeric value that expresses the area under the curve. The unit depends from the input.

auc.mc returns a numeric vector containing the auc values of the it permutations. Just calculate summary statistics from this as you like. Due to the sampling approaches means and medians are not stable for most of the methods. jackknife and jack-validate produce repeatable results, in the case of leave out it depends on n (length(x)) and it.

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

## See Also

trapz, integrate

## Examples

```
## Construct a data set (Imagine 2-hourly ghg emission data
## (methane) measured during a day).
## The emission vector (data in mg CH4 / m2*h) as a time series.
ghg <- ts(c(12.3, 14.7, 17.3, 13.2, 8.5, 7.7, 6.4, 3.2, 19.8,
22.3, 24.7, 15.6, 17.4), start=0, end=24, frequency=0.5)
## Have a look at the emission development.
plot(ghg)
## Calculate what has been emitted that day
## Assuming that emissions develop linearly between
## measurements
auc(time(ghg), ghg)

## Test some of the auc.mc approaches
## "leave out" as default
auc.rep <- auc.mc(time(ghg), ghg)
## mean and median are well below the original value
summary(auc.rep)
## results for "bootstrap" are unstable (run several times)
auc.rep <- auc.mc(time(ghg), ghg, "boot")
summary(auc.rep)
## results for "jack-validate" are stable (run several times)
auc.rep <- auc.mc(time(ghg), ghg, "jack-val", lo=3)
summary(auc.rep)

## The effect of below.zero:
## Shift data, so that we have negative emissions (immissions)
ghg <- ghg-10
## See the difference
plot(ghg)
abline(h=0)
## With thresh = NULL the negative emissions are subtracted
## from the positive emissions
auc(time(ghg), ghg)
## With thresh = -0.5 the negative emissions are set to -0.5
## and only the emissions >= -0.5 count.
auc(time(ghg), ghg, thresh = -0.5)
```

budget.ie *Estimate interpolation errors for GPP and Reco budgets*

## Description

Use several MC methods to estimate the uncertainty associated with the interpolation of fluxes between models when preparing time series data for budgeting $R_{eco}$, GPP (and finally NEE) with budget.reco and budget.gpp.

## Usage

```
budget.ie(bdgt, method = "leave out", lo = 2, it = 100)
```

## Arguments

| | |
|---|---|
| bdgt | An object deriving from running budget.reco or budget.gpp. The budget should cover a reasonable time span, e.g. a year or even better, a calender year and must have been derived with return.models = TRUE and with a set.back that only defines start and end dates. See example at budget.reco. |
| method | Specify how interpolation error should be estimated. Available methods include "leave out", "bootstrap", "sorted bootstrap", "constrained bootstrap", "jackknife", "jack-validate". True bootstrap is only effective when sort.x = FALSE. See auc.mc for details. |
| lo | When estimating interpolation error with "leave out" or "jack-validate", how many data points should be left out randomly? Defaults to 2. See method and details. |
| it | How many iterations should be run? Defaults to 100. Not effective for methods "jackknife", and "jack-validate". |

## Details

ATTENTION: It takes a while. How long one budget run takes depends on the length of the bdgt but typically takes about 5 seconds. So if you run with defaults (it = 100) it may take some minutes. Progress is shown in the console with numbers representing the runs separated by colons.

The approaches are quite similar to the ones in auc.mc. However, the function randomly samples from a list of models and then runs the complete budgeting via budget.reco or budget.gpp. This is done either it times or as often as needed to get all combinations that are possible (for methods "jackknife", and "jack-validate").

## Value

A vector of budgets.

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>,

## References

Beetz S, Liebersbach H, Glatzel S, Jurasinski G, Buczko U, Hoper H (2013) Effects of land use intensity on the full greenhouse gas balance in an Atlantic peat bog. Biogeosciences 10:1067-1082

## See Also

reco.bulk, gpp.bulk, modjust, budget.reco, budget.gpp

## Examples

```
## See examples at budget.reco
```

---

budget.reco *Predict fluxes from GPP and Reco models and prepare for summing them up to budgets.*

---

## Description

The functions predict fluxes from GPP and $R_{eco}$ models and prepare the data for summing them up to budgets including feeding in set.back positions and values (e.g. to truncate the output or to acknowledge the harvesting of biomass) and several adjustments and corrections.

## Usage

```
budget.reco(models, new.data, set.back = NULL, time.unit = "extract",
adjust = TRUE, correct = list(thresh = "get", cvrm = TRUE, wndw = 12,
intvl = 0.95), return.models = FALSE)

budget.gpp(models, new.data, set.back = NULL, time.unit = "extract",
adjust = FALSE, correct = list(thresh = "get", cvrm = TRUE, wndw = 12,
intvl = 0.95), PAR.correct = 100, return.models = FALSE)
```

## Arguments

| | |
|---|---|
| models | List model objects of class "breco" or "bgpp" |
| new.data | Data from a climate station or other logging facility including timestamps. For $R_{eco}$ all temperature variables that have been used to construct the models are needed. For GPP PAR values are needed and the respective column must be named "PAR". The timestamp column (POSIXt) must be named "timestamp". |
| set.back | Data.frame with two columns, the first a timestamp and the second either a value of $R_{eco}$ or GPP, to which fluxes should be set at the date and time in timestamp. Additionally, the second column can contain a value of -999 or -9999 for the start or end of the data series during budgeting. |

| | |
|---|---|
| time.unit | By default the time intervals between data points in new.data are extracted from the timestamp vector. However, by setting time.unit you can additionally give a number that represents the interval in seconds. The function uses this information to linearly interpolate values via lips when some are missing. |
| adjust | Logical. When TRUE, predicted fluxes are adjusted by fitting a linear model to the relationship of predicted to measured fluxes and extracting the slope. This slope value is then used to adjust all predicted fluxes by calculating flux / slope. This is applied to the finished time series of fluxes. For budget.reco default is TRUE whereas for budget.gpp default is FALSE. See details. |
| correct | This triggers a further correction: Unreasonably high fluxes or spikes are identified and eliminated. This is applied to the finished time series of fluxes. The argument requires a named list with the entries thresh, cvrm (logical), wndw, and intvl. When no correction is wanted set correct = NULL. See details. |
| PAR.correct | Numeric representing the value of PAR (e.g. PPFD in micromole per squaremeter per second) below which GPP is assumed to be 0. This is applied to the finished time series of fluxes. All predicted GPP fluxes are corrected to 0 according to this value. See details. |
| return.models | Logical. Shall models be returned? |

## Details

**How does it work?** Both functions take a list of model objects (of class "breco" for $R_{eco}$ models and of class "bgpp" for GPP) and predict fluxes via predict.nls. The required variables are taken from new.data. Fluxes are predicted for each model either for the whole time interval in new.data or when set.back is given with start and end configuration (values -999 and -9999) for the defined time period. If the integration period is defined with set.back, the models that are temporally closest to the start and end times are duplicated and get the respective timestamps. With all models, fluxes are predicted forward from the date and time the models are hooked onto (argument hook in reco.bulk and gpp.bulk) up to the next model's date and time as well as backward down to the previous model's date and time. The resulting two flux vectors for the time period between two consecutive models (one based on the first, the other based on the second model) are linearly interpolated (see below). The same is done for the model errors. Errors are extracted from the model objects and assigned to each interpolated flux from one model to the next and to the previous. Then they are linearly interpolated (see below).

Further, IDs are assigned to the weighted fluxes to identify the periods between two models. This is needed later for model error propagation. The fluxes in the time series between the first and the second model in the whole list get ID = 1, the fluxes in the time series between the second and the third model in the whole list get ID = 2, and so on.

**How does interpolation work?** Linear interpolation of the two vectors (one based on the first model, the other based on the second model) is achieved by calculating weighted means of the integrated fluxes for each date and time with the weights the distances in time to the corresponding model timestamp. Thus, close to the one model timestamp the weighted mean of the two fluxes is almost entirely determined by that model, in the middle of the time series between two model dates both fluxes contribute equally to the mean and close to the other model timestamp the weighted mean of the two fluxes is almost entirely determined by that other model.

**Why adjust?** It may happen that in sum the various models in a seasonal, annual or even bigger dat set tend to over- or underestimate the measured fluxes. To correct for this, the predicted fluxes

can be adjusted as explained above. This typically leads to better overall modelling performance.

**Why** `correct`**?** Especially with locally fitted $R_{eco}$ models it happens - most often with winter data - that predicted fluxes are much higher than supported by the measurements because of the exponential element in fitting $R_{eco}$ models. The model itself is fine but may have been fitted to temperature data spanning a relatively small range. If the temperatures in `new.data` are much higher (in relative terms), then unreasonable high fluxes may result. Such fluxes are identified and eliminated by `correct`.

**How is** `correct`**ed?** `thresh` specifies a maximum predicted flux allowed. When set to "get" (default) it is determined based on the data that were used to construct the models and represents the highest flux ever measured across all campaigns of the series. `cvrm` (logical) triggers whether further despiking should be done with `wndw` the width of the moving window in which the coefficient of variation (cv) is calculated and `intvl` the probability of the `quantile` against which cv should be tested. All fluxes with corresponding cv > `quantile(cv, intvl)` are eliminated.

**Why** `PAR.correct`**?** Because the function not only predicts with all the provided models using `new.data` but also interpolates linearly between models, it happens that unreasonable GPP values are predicted reflecting photosynthesis under no or very low light conditions. These are just set to plant physiologically sensible 0. `PAR.correction` is done after inserting `set.back`(s) for cutting(s).

All data gaps resulting from corrections are then filled with linear interpolation from the values adjacent to the gap via `lips`.

**How** `set.back` **is used to factor in cut dates or the like:**. When further `set.back` values are given in addition to the definition of the start and end dates, e.g. to acknowledge for biomass removal when predicting GPP fluxes, two things happen. First, the time series of fluxes resulting from all of the above is changed like this: At the date and time of a cut the flux is set to the value given in `set.back` and then fluxes are linearly interpolated to the next proper model date by weighted means in the same manner as described above. Second, cut models are defined as linear models and integrated into the model list according to their timestamp. Thus, when run with `return.models = TRUE`, the updated model list can be used with `tbl8` to extract the relevant model parameters.

## Value

Both functions return a `data.frame` (called `tbl`) containing the predicted values, timestamps, etc. and optionally an object of class `"breco"` or `"bgpp"` (called `models`) containing the final models including the start and end models and any `set.back` models when `set.backs` were specified.

For `budget.reco` `tbl` has 4 columns

| | |
|---|---|
| `reco.flux` | Predicted and interpolated fluxes |
| `reco.se` | Model errors spawned across predicted and interpolated fluxes |
| `reco.id` | ID that identifies the model periods |
| `timestamp` | Timestamps |

For `budget.gpp` `tbl` has 4 columns

| | |
|---|---|
| `gpp.flux` | Predicted and interpolated fluxes |
| `gpp.se` | Model errors spawned across predicted and interpolated fluxes |
| `gpp.id` | ID that identifies the model periods |
| `timestamp` | Timestamps |

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>,

with ideas by Sascha Beetz, <sascha.beetz@uni-rostock.de>

## References

Beetz S, Liebersbach H, Glatzel S, Jurasinski G, Buczko U, Hoper H (2013) Effects of land use intensity on the full greenhouse gas balance in an Atlantic peat bog. Biogeosciences 10:1067-1082

## See Also

fluxx, reco, gpp, gpp2, reco.bulk, gpp.bulk, modjust

## Examples

```
### The examples are consecutive and are a suggestion
### how to run the whole process from bulk modelling
### over corrections and checks to full budgets including
### propagated model and interpolation error terms.

## The whole examples section is marked as
## not run because parts take longer than
## accepted by CRAN incoming checks.
## Remove first hash in each line to run them.

## Not run ##
## load data
#data(amd)
#data(amc)
## set global conversion factor
## All fluxes are in micromole * m-2 * s-1,
## thus they are transformed to g CO2-C * m-2 *1/2h
#uf <- 12*60*30/1000000
#
#### Reco ### (for details see reco.bulk)
## extract opaque (dark) chamber measurements
#amr <- amd[amd$kind=="D",]
## fit reco models
#r.models <- reco.bulk(flux ~ t.air + t.soil2 + t.soil5 +
#t.soil10 + timestamp, amr, amr$campaign, window=3,
#remove.outliers=TRUE, method="arr", min.dp=2)
## adjust models
#r.models <- modjust(r.models, alpha=0.1, min.dp=3)
#
### prepare Reco budget (predict half hourly values for two years)
## define set.back with start and end dates only
## (typically you would take this from read in table)
#set.back <- data.frame(timestamp = c("2010-01-01 00:30", "2011-12-31 23:30"),
#value = c(-999, -9999))
#set.back$timestamp <- strptime(set.back$timestamp, format="%Y-%m-%d %H:%M", tz="GMT")
## run budget function with defaults
```

```
#r.bdgt <- budget.reco(r.models, amc, set.back)
## prepare for quality check (global model of predicted ~ measured)
#r.check <- checkm(r.bdgt, amr)
## have a look
#par(pty="s")
#lims <- range(r.check$reco.flux, r.check$flux)
#plot(reco.flux ~ flux, data=r.check, xlim=lims, ylim=lims)
#abline(coef=c(0,1), lty=3)
#mf1 <- lm(reco.flux ~ flux, r.check)
#abline(mf1)
#summary(mf1)
## calculate daily values (better for plotting)
#r.bdgt$day <- format(r.bdgt$timestamp, format="%Y-%m-%d")
#r.daily <- data.frame(day = unique(r.bdgt$day))
#r.daily$day <- as.POSIXct(strptime(r.daily$day, format="%Y-%m-%d",
#tz="GMT"), tz="GMT")
## in addition to summing up per day, change unit
#r.daily$reco <- tapply(r.bdgt$reco.flux*uf, r.bdgt$day, sum)
## same for the model error terms
#r.daily$se <- tapply(r.bdgt$reco.se*uf, r.bdgt$day, sum)
#
#
#### GPP ### (for details see gpp.bulk)
#g.models <- gpp.bulk(flux ~ PAR + timestamp + kind, amd, amd$campaign,
#method="Falge", min.dp=5)
#
### prepare GPP budget (predict half hourly values for two years)
## define set.back with start, end, and cut dates
## (typically you would take this from read in table)
#set.back <- data.frame(timestamp = c("2010-01-01 00:30", "2011-12-31 23:30",
#"2010-07-22 12:00", "2010-09-03 12:00", "2010-10-13 12:00", "2011-06-29 12:00",
#"2011-08-11 12:00", "2011-10-21 12:00"), value = c(-999, -9999, rep(-0.0001, 6)))
#set.back$timestamp <- strptime(set.back$timestamp, format="%Y-%m-%d %H:%M", tz="GMT")
## have a look at the resulting data.frame
#set.back
## run budget function with correct = NULL and return the models
#g.bdgt <- budget.gpp(g.models, amc, set.back, correct=NULL, return.models=TRUE)
## the cut models are also returned:
#tbl8(g.bdgt$models)
## extract the half hourly values to proceed
#g.bdgt <- g.bdgt$tbl
## make daily values (better for plotting)
#g.bdgt$day <- format(g.bdgt$timestamp, format="%Y-%m-%d")
#g.daily <- data.frame(day = unique(g.bdgt$day))
#g.daily$day <- as.POSIXct(strptime(g.daily$day, format="%Y-%m-%d", tz="GMT"), tz="GMT")
## in addition to summing up per day, change unit
#g.daily$gpp <- tapply(g.bdgt$gpp.flux*uf, g.bdgt$day, sum)
## same for the model error terms
#g.daily$se <- tapply(g.bdgt$gpp.se*uf, g.bdgt$day, sum)
#
#
#### Budgets ###
### doing the actual budgeting
```

```
## first bring Reco and GPP budget data together
## because of different handling data sets
## may be of different length, therefore use merge
#r.bdgt$ts <- as.character(r.bdgt$timestamp)
#g.bdgt$ts <- as.character(g.bdgt$timestamp)
#bdgt <- merge(r.bdgt, g.bdgt[, -c(ncol(g.bdgt)-2, ncol(g.bdgt)-1)],
#by.x="ts", by.y="ts", all.x=TRUE)
## calculate NEE
#bdgt$nee.flux <- bdgt$reco.flux + bdgt$gpp.flux
## error propagation
#bdgt$nee.se <- sqrt(bdgt$reco.se^2 + bdgt$gpp.se^2)/sqrt(2)
## define unique id that spans across reco and gpp ids
#bdgt$nee.id <- paste(bdgt$reco.id, bdgt$gpp.id, sep=".")
## do budgets of fluxes (sum and use global conversion factor, see above) and error terms
## the model errors are summed up per model id and resulting
## sums are combined following error propagation
#with(bdgt, {c(
#  reco = sum(reco.flux*uf, na.rm=TRUE),
# reco.me = sqrt(sum(tapply(reco.se, reco.id, sum)^2))*uf,
# gpp = sum(gpp.flux*uf, na.rm=TRUE),
# gpp.me = sqrt(sum(tapply(gpp.se, gpp.id, sum)^2))*uf,
# nee = sum(nee.flux*uf, na.rm=TRUE),
# nee.me = sqrt(sum(tapply(nee.se, nee.id, sum, na.rm=TRUE)^2))*uf
#)})
#
### annual budget incl. interpolation error
#set.back <- data.frame(timestamp = c("2010-01-01 00:30",
#"2010-12-31 23:30"), value = c(-999, -9999))
#set.back$timestamp <- strptime(set.back$timestamp, format="%Y-%m-%d %H:%M", tz="GMT")
### reco.ie
## redoing budget with annual bounds
#r.bdgt.2010 <- budget.reco(r.models, amc, set.back, return.models=TRUE)
## run budget.ie with lo = 3 and it = 10 (default of 100 is advisable but slow)
#r.ie2010 <- budget.ie(r.bdgt.2010, lo=3, it=10)
## summary statistic and factor in the uf
#r.ie2010 <- sd(r.ie2010*uf)
## gpp.ie
## redoing budget with annual bounds
#g.bdgt.2010 <- budget.gpp(g.models, amc, set.back, correct=NULL, return.models=TRUE)
#g.ie2010 <- budget.ie(g.bdgt.2010, lo=3, it=10)
#g.ie2010 <- sd(g.ie2010*uf)
#
## do the actual budgeting
#tmp <- bdgt[(bdgt$timestamp >= set.back$timestamp[1]) &
#(bdgt$timestamp <= set.back$timestamp[2]),]
#with(tmp, {c(
#  reco = sum(reco.flux*uf, na.rm=TRUE),
# reco.me = sqrt(sum(tapply(reco.se, reco.id, sum)^2))*uf,
# reco.ie = r.ie2010,
# gpp = sum(gpp.flux*uf, na.rm=TRUE),
# gpp.me = sqrt(sum(tapply(gpp.se, gpp.id, sum)^2))*uf,
# gpp.ie = g.ie2010,
# nee = sum(nee.flux*uf, na.rm=TRUE),
```

```
# nee.me = sqrt(sum(tapply(nee.se, nee.id, sum, na.rm=TRUE)^2))*uf,
# nee.ie = sqrt(r.ie2010^2 + g.ie2010^2)
#)})
#
## End not run ##
```

---

checkm *Bring modelled and measured values together based on timestamp*

---

## Description

Trivial function that is a simple wrapper for frequent task: Bringing together the measured and modelled values, for instance to do a posteriori analyses of model performance.

## Usage

```
checkm(modelled, measured, t.unit = NULL)
```

## Arguments

| | |
|---|---|
| `modelled` | A `data.frame` with the modelled fluxes and all additional data reported. For instance resulting from doing `budget.reco` on a "`breco`" object. One variable has to be the timestamp and it should be named exactly like this. |
| `measured` | A `data.frame` with the originally measured fluxes and additional data. One variable has to be the timestamp and it should be named exactly like this. |
| `t.unit` | If NULL, data in `modelled` and `measured` are merged based on their timestamp by calculating the minimum difference in time between all entries (see Details). If !NULL, character string specifying to which time interval the timestamps in `measured` should be rounded using `round.POSIXlt`. May be one of "`mins`", "`5mins`", "`10mins`", "`15mins`", "`quarter hours`", "`30mins`", "`half hours`", "`hours`". Alternatively, a numeric specifying the minutes to round to. To go to seconds just use values < 1, to go beyond the hour just use values > 60. |

## Details

**Case 1 (t.unit = NULL)** Data are merged by calculating the difference in time between all timestamps in `modelled` and all timestamps in `measured` and identifying the minimum difference to each measured flux. If minimum difference between measured and modelled flux > 1h, no modelled flux is assigned. This approach is a bit slower but it is not necessary to give a correct `t.unit`, which makes it less error prone.

**Case 2 (t.unit != NULL)** After rounding the timestamps in `measured` according to `t.unit` and transforming both timestamnps to character vectors `modelled` and `measured` are merged based on these timestamps and only data rows that are present in both are retained. Therefore `t.unit` has to be specified according to the interval of the timestamps in `modelled`.

## Value

Data.frame containing the corresponding rows of `modelled` and `measured`

### Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

### Examples

```
## See examples at reco.bulk
```

---

chop *Prepare data for flux rate estimation with* flux *or GPP/Reco modelling.*

---

### Description

The function simply constructs a list of data.frames that each contains the data for one closed chamber measurement or for one NEE/GPP or $R_{eco}$ model.

### Usage

```
chop(dat, factors, nmes = NULL, min.cm = 3)
```

### Arguments

| | |
|---|---|
| dat | data.frame containing static closed chamber data for several chamber measurements. See columns and example for details. |
| factors | A character vector giving the names of the columns that are used to partition the data in dat into chunks that each contains the data for one chamber placement. factors $\subset$ columns! See example. |
| nmes | A character vector giving the names of the columns that are used to name the data chunks. nmes $\subset$ columns! See example. |
| min.cm | Integer giving the minimum number of concentration measurements allowed per chamber measurement. Defaults to 3 because a linear fit to 2 points does not make any sense. Attention: Chamber placements with less than min.cm measurements are quietly skipped. |

### Details

This could easily be hand scripted (e.g. with split) but the function shall provide a simple way to obtain the structure needed for flux and it also carries naming information.

### Value

Returns a list with 2 entries. The first is itself a list of data.frames containing the concentration measurements that result from the field sampling during one chamber placement (if factors was specified correctly) and the columns specified in columns. The entries in the list are named according to nmes. However, the second part of the upper level list is a table with the naming information. This is handed over to flux and plot.fluss. See example.

### Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

### See Also

flux

### Examples

```
## load example data
data(tt.pre)

## extract field concentration measurements
gcd <- tt.pre[tt.pre$sampletype_a=="P",]

## partition the data into data tables per chamber measurement
gcd.parts <- chop(gcd, factors = c("date", "spot", "veg"),
nmes = c("date", "veg", "spot"))
# have a look at the first three tables
gcd.parts$tables[1:3]
# have a look at the names part of the returned object
gcd.parts$nmes
# use inspect to have a look at (a) specific data table(s)
inspect(gcd.parts, c("2011-03-15.c.3", "2011-03-15.c.6", "2011-03-15.p.6"))
# inspect the same tables using their indices
inspect(gcd.parts, c(3,6,12))
inspect(gcd.parts, c("c.3", "c.6", "p.6"))
```

---

## export *simple export wrapper*

---

### Description

Export your flux estimations easily

### Usage

```
export(x, digits = 4, ...)
```

### Arguments

| | |
|---|---|
| x | A fluxes object. |
| digits | The number of digits that all numeric values in the output table shall have. Defaults to 4. |
| ... | Further arguments to write.table. For instance to specify another field separator (defaults to tab delimited output files) and a file where to write the results to. |

### Details

It's really very simple.

### Value

The function is called for its side effects. Nothing is returned.

### Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

### See Also

flux, chop, (also for examples)

---

## flux *Estimate gas flux rates using non-steady-state closed chamber data*

---

### Description

flux is a convenience wrapper for the other two functions that should be suitable for most users. It can be used to estimate gas fluxes for all three commonly measured greenhouse gases ($CO_2$, $CH_4$, $N_2O$) at once or separately.

### Usage

```
flux(x, var.par, co2ntrol = list(leak = TRUE, relay = FALSE),
min.allowed = 3, max.nrmse = 0.1, nrmse.lim = 0.2, r2.qual = 0.8,
range.lim = 30, out.unit = "auto", elementar = FALSE,
hardflag = list(range = TRUE), asterisks = TRUE)

flux.odae(dat, var.par, min.allowed = 3, max.nrmse = 0.1, rl = NULL)

flux.conv(fl.dat, ghg = "CH4", r2.qual = 0.8, nrmse.lim = 0.2,
out.unit = "auto", elementar = FALSE, hardflag = list(range = TRUE))
```

### Arguments

| | |
|---|---|
| x | A list of data tables as returned by chop. Each table contains data for one chamber measurement. Required are at least the columns specified in the argument var.par (see also Example). |
| var.par | A named list specifying the variables and parameters that are used in the estimation process and variables that should be handed through the function so that they are easily available for further analysis. Some of the names are obligatory (time, volume, area, t.air, p.air, and two or more that specify the gas column and the gas quality column), others are optional. See details and examples. |

ghg — Character string identifying the greenhouse gas for which concentration measurements are handled. Can be "CH4", "N2O" or "CO2". Defaults to "CH4". This argument determines the molar weight that is used in the flux calculation with gflux, the input unit (ppb or ppm), and the ambient concentration of the gas that is added to the diagnostic plots plotted with plot.fluss and used for the determination of the number of measurements below ambient (nomba).

co2ntrol — Options for estimating fluxes with $CO_2$ control. In this case outliers and the slope of the $CO_2$ concentration measurements influence the estimated flux rate of the other greenhouse gases ($N_2O$ and $CH_4$; see Details). By default, $CO_2$ correction is switched off.

min.allowed — Integer giving the minimum number of concentration measurements allowed during the estimation of one single flux. Can be any number between 3 and the number of concentration measurements during one chamber placement.

max.nrmse — Numeric giving the maximum acceptable normalized root mean square error for configurations with higher numbers of concentration measurements than specified in min.allowed. Numeric value between 0 and 1. Defaults to 0.1. Above that value lower numbers of concentration measurements down to min.allowed are considered. See details. In flux max.nrmse can also be given as a named list with three items giving the maximum acceptable nrmse per gas. See examples.

nrmse.lim — Numeric between 0 and 1 (defaults to 0.2) giving the main quality parameter for the model fit, the maximum acceptable normalized root mean square error. If the best fit for one chamber measurement exceeds this value, the function reports FALSE in the nrmse.f quality flag. See details and value. In flux nrmse.lim can also be given as a named list with three items giving the nrmse.lim per gas. See examples.

range.lim — The minimum detectable range of the concentration measurements during one chamber placement. Has to be either a single numerical value, a numeric vector with the same length as x giving different range limits for each chamber placement (for instance obtained by flux.calib) or a character string naming the column in x that contains range limit data. If this column is named "rl" (the default when the range limits are attached to the data by flux.calib) flux automatically detects it. Note, that setting range.lim != NULL overrides the auto detected range limits.

The acceptable range limit depends on the accuracy of the concentration measurements. When the range of the concentration measurements is smaller than the repeatability range of the measurement device (e.g., a gas chromatograph) one cannot tell real increase in concentration from random fluctuation. Therefore, if the range of the concentration measurements during one chamber placement is < range.lim, the range.flag is set to FALSE (0). See details. In flux range.lim can also be given as a named list with three items giving the maximum acceptable range limits per gas. See examples.

r2.qual — Numeric giving the limit of minimum acceptable r2 as an alternative quality parameter describing the model fit. Can be between 0 and 1 (0.8 by default). If a model r2 is below the setting the r2 quality flag is reported FALSE (0). In flux r2.qual can also be given as a named list with three items giving the acceptable r2.qualities per gas. See examples.

| | |
|---|---|
| out.unit | Character string determining the output unit of the flux rate mass part. Character string. The default "auto" tries to find a unit that ranges the output value between 0.01 and 10. Possible output units are "ng", "mug", "mg", or "g". "mug" stands for "$\mu$g" because non-ascii characters are not allowed in functions. |
| elementar | When the fluxes are wanted as element values set elementar = TRUE. Defaults to FALSE. |
| hardflag | Named list that controls which of the quality flags are to be hard flagged (the value is changed according to the quality flag). range.lim is hard flagged by default. So when the range of the actual concentration values for a chamber measurement is smaller than the set range limit a zero flux is returned. When a flux estimation does not meet the quality requirements of any other hard-flagged quality parameter flux returns NA. For changing a quality parameter to a hard flag just provide its name (without quotation marks) and set it to TRUE. Possible parameters to chose as hard or soft flag are nrmse, range, and r2. Further the number of measurements below ambient (nomba) can be hardflagged by setting nomba = [0. . . nc] with nc = number of concentration measurements during one chamber placement. See examples. |
| asterisks | Logical. If TRUE, p-values are given as asterisks. |
| dat | One data table for one chamber placement. See x and var.par for details. |
| rl | Specifies range.lim in the low level function flux.odae. As with range.lim several options are allowed. Defaults to NULL. In this case, the function looks for a column rl in dat. If it can't find a column rl, the value is set to 0 and a warning is returned; if it does exist dat$rl is always taken. If rl is a character string the function looks for a column of that name in dat, if rl is a numeric value, this value is taken as the range limit. See examples. |
| fl.dat | An object with the same structure as returned by flux.odae. See details and value. |

## Details

Typically it will be most convenient to use flux on objects returned by chop (i.e. on lists of data tables that contain all necessary data per chamber measurement including supporting information). flux simply wraps flux.odae and flux.conv applied on lists of chamber measurement data tables into one function. Thus, the data of a one day field campaign or a year of chamber measurements can easily be handled by simply running two functions (chop and flux) consecutively to estimate ghg fluxes for all three common ghg gases. See example.

Probably the most important argument is var.par. It specifies the variables (by referring to the names of the data columns) from x and parameters (fixed values that are constant for all chamber placements) that are used for the flux estimations. For simple handling it is expecting a named list.

For flux the obligatory list items are: One item that refers to a column in x containing ghg concentrations (see next paragraph for details); time – chamber closing time in minutes; volume – chamber volume during placement (in cbm); area – chamber area (in sqm); t.air – air temperature inside chamber during concentration measurements (in °C); p.air – air pressure during concentration measurements (in Pa).

The list items that are used to specify the ghg for which flux estimation is carried out have to be specified by using the named list items CH4 – $CH_4$ concentrations; CH4.gcq – $CH_4$ GC quality flag;

CO2 – $CO_2$ concentrations; CO2.gcq – $CO_2$ GC quality flag; N2O – $N_2O$ concentrations; N2O.gcq – $N_2O$ GC quality flag. Fluxes are estimated for all ghg for which concentration data are given. Thus, at least one ghg should be specified. GC quality flags are optional. If you don't provide a reference to a column in x the function assumes that all GC measurements were OK.

All these list items can either be given as a variable (name of a column in x) or as a fixed parameter (a numeric value). This makes no sense for the ghgs and time, but in many cases chamber volume and area will be constant across measurements. Another likely candidate for a fixed parameter is p.air because air pressure is often not logged during chamber measurements. All additional list items should be of type 'variable' and refer to further columns in x if you want those data handed through the function and be part of the result tables (for having all data in one place for further analyses). You are free to choose appropriate names. Fixed parameters will not be relayed.

If the flux estimation is carried out in two steps it will typically be carried out on a list structure as returned by chop. Therefore, it is used within a lapply call. For details see examples. However, the functions flux.odae and flux.conv are designed to be carried out on single data tables (data.frame) per chamber measurement.

First flux.odae is run. It simply tries to find the best model fit for the series of concentration measurements that are given in dat. This data.frame has to consist of five columns that give (in that order): gas concentration, closing time of the chamber in minutes, gas concentration quality flag, chamber volume, temperature within the chamber headspace during measurements (may change during chamber placement). See example data.

At the moment the optimization bases on linear regression. All possible models with n (= total number of concentration measurements per chamber placement) to min.allowed number of concentration measurements are fitted and the best fit is evaluated in a stepwise procedure. The normalized root mean square error is used as the quality criterion for the outlier detection and elimination procedure. All model fits with a nrmse <= max.nrmse are extracted and ranked according to the number of concentration measurements (decreasing) and to the nrmse (increasing). The first ranked model is stored along with the original data table and some other information. Therefore a model with e.g. a nrmse of 0.081 constructed from 5 concentration measurements wins against a model with a nrmse of 0.07 with only 4 concentration measurements. This reflects the idea that models with nrmse <= max.nrmse already represent a sufficient fit and do not have "outliers" that must be eliminated.

In case no model has a nrmse <= max.nrmse, the models are simply ranked according to their nrmse and the model with the lowest nrmse wins and is stored. In that way outliers are detected and exluded. flux.odae returns a complex object that contains most of the necessary information for the flux.conv function and also carries information that is later needed for the plot functions (plot.flux and plot.fluss).

The flux calculation is then carried out with the function flux.conv. It takes the object returned by flux.odae and additional information (chamber area, gas species, several quality settings and in- as well as output units) and calculates the flux rates. Further several quality checks (r2 check, range check, nrmse check, nomba check; for details see Value) are carried out and quality flags are reported along with the fluxes in the output. It is best when all quality flags are returned TRUE. Depending on the application quality requirements might vary. Therefore, per default the function reports soft quality flags (despite for range). However, this can be changed via hardflag.

The idea behind co2ntrol in flux is that the $CO_2$ concentration measurements might serve as a further check on the integrity of the chamber measuremnt in the field. When co2ntrol is set, the function first carries out an outlier procedure on the $CO_2$ concentration data (the respective

columns have to be in x of course). Further, the slope of the $CO_2$ concentration change over time is checked. When it is negative, chamber leakage is assumed and a respective quality flag (leak.flag) is reported FALSE. The leak.flag cannot be hard flagged.

## Value

flux returns a complex object of class fluxes that is a 3 entry list. When the object is printed to the console only the second entry is displayed in a modified form that is meant to maximize information display with small footprint for easy inspection. A table is printed to the console with three columns per gas. The first contains the quality flags (e.g. "111.02"). The order is: nrmse.f, r2.f, range.f, nomba.f, leak.f. The first three are considered more important, and if they are '1' everything is fine. The first flag behind the full stop just gives the number of measurements below ambient, while the second is '2' when co2ntrol was switched off, '0' when leaking occurred, and '1' when no leaking occurred.

The data.frame with the estimated flux rates contains all data needed for further analysis. The columns represent the entries in fluss of the single chamber measurements (including quality flags, see below) plus naming information according to the settings in the nmes argument of chop. export provides a simple way to export the results. The first entry is itself a list of lists and data tables. It is called flux.res and is comprised of objects that are returned by flux.conv per ghg. Each first level entry in these lists contains the information for one chamber measurement. It is named according to the nmes-setting in chop and contains the elements fluss (which is itself a list with the elements given below), fl.dat (equals the object returned by flux.odae; see below), and unit which provides information on the output mass unit of the flux rate that is handed over to the function plot.fluss and to the table output.

The elements of fluss:

| | |
|---|---|
| ghg | Character. The gas species for which the flux has been estimated. |
| flux | Numeric. Calculated flux rate in mass unit per m2 and hour. |
| r2 | The $R^2$ of the best fitted model that has been used for flux caclulation. |
| nrmse | The NRMSE of the best fitted model that has been used for flux caclulation. |
| r2.f | Logical. $R^2$ quality flag telling whether the $R^2$ quality setting given in r2.qual is fulfilled. |
| range.f | Logical. Range quality flag telling whether the range of the concentration measurements exceeded the quality range of the measurement device that has been specified in range.lim. |
| nrmse.f | Logical. NRMSE quality flag telling whether the NRMSE quality setting given in nrmse.lim is fulfilled (i.e. if the NRMSE of the best model <= nrmse.lim). |
| nomba.f | Integer. Reports the **n**umber **o**f **m**easurements **b**elow **a**mbient. The ambient concentrations are set to be 392.6 ppm ($CO_2$), 1874 ppb ($CH_4$), and 324 ppb ($N_2O$) (taken from Mace Head Ireland ($N_2O$, $CH_4$) and global average ($CO_2$) obtained from http://cdiac.ornl.gov/pns/current_ghg.html as of August 16th, 2013). |
| leak.f | Logical. When co2ntrol was applied with leak = TRUE, possible chamber leakage as represented by decreasing $CO_2$ concentrations over time is shown by a FALSE (0). |

The elements of fl.dat that is also the object returned by flux.odae are:

| | |
|---|---|
| lm4flux | Complex object. The best fitting model as reported by lm. It builds the basis for the calculation of the flux rate via flux.conv. |
| row.select | Integer vector giving the indices of the rows of the data table that have been used to construct the best fitting model. This information is later used in the plotting functions plot.flux and plot.fluss. |
| orig.dat | data.frame with the original data provided according to arguments x and columns. |
| out.dat | Data to be handed through. Per default area and volume of the chamber are relayed but these values are not part of the table output whereas all additionally relayed data are part of the table output. |

## Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>, Franziska Koebsch <franziska.koebsch@uni-rostock.de>, Ulrike Hagemann <ulrike.hagemann@zalf.de>, Anke Günther <anke.guenther@uni-rostock.de>

## References

Nakano T (2004) A comparison of regression methods for estimating soil-atmosphere diffusion gas fluxes by a closed-chamber technique. Soil Biology and Biochemistry 36: 107-113.

Forbrich I, Kutzbach L, Hormann A, Wilmking M (2010) A comparison of linear and exponential regression for estimating diffusive CH4 fluxes by closed-chambers in peatlands. Soil Biology and Biochemistry 42: 507-515.

## See Also

chop, flux.calib, gflux, plot.fluss

## Examples

```
## load example data
data(tt.pre)

## extract field concentration measurements
gcd <- tt.pre[tt.pre$sampletype_a=="P",]

## partition the data into data tables per chamber measurement
# then do the partitioning
gcd.parts <- chop(gcd, factors = c("date", "spot", "veg"),
nmes = c("date", "veg", "spot"))

## calculate flux rates for methane
# first define a global CH4 range limit
CH4.lim <- 30
# do the flux rate estimation (it will often be best to define
# var.par separately, note that p.air is given as a parameter)
vp.CH4 <- list(CH4 = "CH4ppb", time = "time_min", CH4.gcq = "CH4Code",
volume = "cham_vol", t.air = "temp_dC", area = "cham_area", p.air = 101325)
flux.CH4 <- flux(gcd.parts, var.par = vp.CH4)
# look at the results table
```

```
flux.CH4

# extracting range limits from the calibration gas measurements
# and attaching them to gcd.parts. first get the calibration gas
# measurements from tt.pre (changing the date because it is in
# a strange format and has to be the same as the dates in gcd.parts)
cgm <- tt.pre[tt.pre$sampletype_a=="E",c("date_gc", "CH4ppb", "CH4Code",
"CO2ppm", "CO2Code", "N2Oppb", "N2OCode")]
names(cgm)[1] <- "date"
cgm$date <- "2011-03-16"
# now we can do the flux.calib
gcd.parts.cal <- flux.calib(gcd.parts, columns = c("date", "CH4ppb"),
calib = cgm, format="%Y-%m-%d", window=48, buffer=1100, attach=TRUE)
# do the flux rate estimation (we use the same var.par as before)
flux.CH4 <- flux(gcd.parts.cal, var.par=vp.CH4, co2ntrol = NULL,
range.lim=NULL)
# look at the results table
flux.CH4
# export the results to the working directory
wd <- getwd()
export(flux.CH4, file=paste(wd, "/flux.CH4.txt", sep=""))

## plot the concentration-change-with-time-plots as kind of diagnostic
plot(flux.CH4, dims = c(3,6))

## do the flux rate estimation whilst using CO2 concentrations to
## control for possible chamber leakage
flux.CH4.b <- flux(gcd.parts, var.par=vp.CH4)
# look at the results table
flux.CH4.b
# plot the concentration-change-with-time-plots as kind of diagnostic
plot(flux.CH4.b, dims = c(3,6))

## do the flux rate estimation whilst using CO2 concentrations to
## control for outliers and possible chamber leakage
flux.CH4.c <- flux(gcd.parts, var.par=vp.CH4, co2ntrol = list(leak = TRUE,
relay = FALSE))
# look at the results table
flux.CH4.c
# plot the concentration-change-with-time-plots as kind of diagnostic
plot(flux.CH4.c, dims = c(3,6))
```

---

`flux.calib` *Determine calibration measurement ranges according to the dates of real measurements*

---

## Description

The function basically takes calibration gas measurements and extracts the calibration gas measurements that have been carried out temporally close to a real data measurement and calculates the

standard deviation of the calibration gas measurements. The obtained range limits can be used in flux as a quality parameter (via range.lim).

## Usage

```
flux.calib(dat, columns, calib, format = "%Y-%m-%d %H:%M:%S",
window = 3, buffer = 1000, n.cg = 4, rl.backup = 20, attach = FALSE)
```

## Arguments

| | |
|---|---|
| dat | Object returned by chop containing gas concentration measurements for several chamber measurements. |
| columns | Character vector giving the names of the two columns that shall be taken from dat and from calib for extracting the calibration measurements. Typically one date and one concentration column. This also implies that they carry the same names in dat and calib. |
| calib | data.frame with concentration measurements of calibration gases that have been carried out at least in part during the time the concentration measurements in dat have been achieved. |
| format | Character string specifying the format of dates in dat as well as in calib. Internally dates are converted to a date format R can handle (see strptime for details and format options). |
| window | Integer value. Hours. Window around the date and time (if available) of measurement of the field greenhouse gas concentrations at the measurement device (e.g. a GC) that shall be considered for the inclusion of calibration gas measurements. If no times are given windows = 48 includes the day after the measurement date and windows >48 include the day before and the day after the measurement date. |
| buffer | Numeric. Concentration buffer around the range of concentration measurements in dat in which the function searches for calibration gas measurements. Defaults to 1000 (ppm or ppb, depends on gas). When only the closest calibration gas concentration shall be considered one can decrease the buffer. When real concentrations are far different from available calibration gas concentrations one might need to increase the buffer to have enough data. |
| n.cg | Integer. Number of calibration gas concentrations in calib. |
| rl.backup | Numeric value. Range limit backup value that is used in situations where no range limit can be derived from the calibration measurements. See details. Defaults to a quite reasonable 20. Deprecated. |
| attach | Logical. If TRUE the range limits are attached to the original data. |

## Details

The function automatically detects the single species of calibration gases that have been measured. It calculates the standard deviations of the measurements per calibration gas species and than gives back an average of the calculated range limit values if there are more than one calibration gas concentrations covered by the range within the field concentration measurements per chamber placement. However, this is rather academic because a chamber measurement for which concentrations

develop over the range of two or more calibration concentrations will typically not have a range limit problem.

In its actual form it is possible that there are no valid calibration measurements found for certain chamber data because the range of the chamber data (even with range extension) does not cover any of the calibration gas concentrations. In this case, the minimum range limit is assigned if `rl.backup` = NULL.

## Value

Returns a named vector with the range limits of the measurement device (as needed within `flux`) per chamber measurement or attaches the range limits to the original data tables that are in `x` and returns the altered `x`.

## Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

## See Also

`chop`, `flux`

## Examples

```
## load example data
data(tt.pre)

## extract field concentration measurements
gcd <- tt.pre[tt.pre$sampletype_a=="P",]

## partition the data into data tables per chamber measurement
gcd.parts <- chop(gcd, factors = c("date", "spot", "veg"),
nmes = c("date", "veg", "spot"))

## calculate range limits according to the data and the accompanying
## calibration gas measurements
# extract and prepare calibration measurements
cal <- tt.pre[tt.pre$sampletype_a=="E",c("date_gc", "CH4ppb", "CH4Code",
"CO2ppm", "CO2Code", "N2Oppb", "N2OCode")]
names(cal)[1] <- "date"
cal$date <- "2011-03-16"
# calculate the range limits per gas (makes no real sense with such
# a small dataset).
# CH4 range limits
CH4.lims <- flux.calib(gcd.parts, columns = c("date", "CH4ppb"),
calib = cal, format="%Y-%m-%d", window=48, attach=FALSE, buffer=1100)
# N2O range limits
N2O.lims <- flux.calib(gcd.parts, columns = c("date", "N2Oppb"),
calib = cal, format="%Y-%m-%d", window=48, attach=FALSE, buffer=1100)
# CO2 range limits
CO2.lims <- flux.calib(gcd.parts, columns = c("date", "CO2ppm"),
calib = cal, format="%Y-%m-%d", window=48, attach=FALSE, buffer=1100)
```

```
## attach the range limits to the original data
gcd.parts.cal <- flux.calib(gcd.parts, columns = c("date", "CH4ppb"),
calib = cal, format = "%Y-%m-%d", attach = TRUE, window=48, buffer=1100)
```

---

`fluxx` *Estimate (ghg) flux rates from online dynamic closed chamber measurements in through-flow mode.*

---

## Description

(Bulk) estimates of (ghg) fluxes from online concentration measurements with non-steady-state closed chambers. The function tries to find stable linear conditions in concentration change by fitting many regressions to the data and automatically detects and excludes rapid concentration fluctations.

## Usage

```
fluxx(x, var.par, subset, asterisks = FALSE, loop = "auto", ...)

mf.flux(x, var.par, method = "r2", time.unit = "S", all.through = TRUE, iv = 1,
wndw = 0.1, pdk = 0.5, min.dp = 20, nrmse.lim = 0.1, r2.qual = 0.9,
range.lim = 5, out.unit = "auto", elementar = FALSE,
hardflag = list(range = TRUE), consecutive = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | A list of data tables as returned by `chop` or alternatively one data table (for `mf.flux` which will rarely be called directly by the user). Each table contains data for one chamber measurement. Minimum requirements are the columns specified in `var.par`. |
| `var.par` | A named list specifying the variables and parameters that are used in the estimation process and variables that should be handed through the function so that they are easily available for further analysis. Some of the names are obligatory (e.g. `time`, `volume`, `area`, `t.air`, `p.air`, and two or more that specify the gas column and the gas quality column), others are optional. See details and examples. |
| `subset` | An optional vector specifying a subset of concentration measurements to be used in the estimation process. |
| `asterisks` | Logical. If TRUE p-values are given as asterisks and other symbols (p<.001 = "***", .001<p<.01 = "**", .01<p<.05 = "*", .05<p<.1 = ".", p>=.1 == " "). |
| `loop` | Can be `TRUE`, `FALSE` or "auto". Determines how bulk flux estimations are done. If `TRUE` a `for`-loop is used, if `FALSE` `lapply` is used, and if "auto" the approach is switched automatically depending on the number of data tables: If `x` contains more than 100 data tables (chamber placements) the approach is switched from `lapply` to `for`-loop because `lapply` may be slow on large `x`. |

| | |
|---|---|
| ... | Further arguments passed to mf.flux. |
| method | Character string specifying the statistic used for finding the linear part. Partial match to "r2", "rmse", "AIC". Defaults to "r2". See Details. |
| time.unit | Single character giving the appropriate unit of time elapsed between two concentration measurements. Will typically be seconds, thus default is "S". Other options are "M" for minutes and "H" for hours. ATTENTION: Setting the time unit to the wrong value will result in incorrect fluxes. |
| all.through | Logical. When TRUE, all data columns in x other than the ones needed for flux calculation are also handed through the function so that they can be used in later steps of analysis. You may also specify one or several columns that are handed through using var.par. The all.through setting overruns the var.par settings. |
| iv | Numeric. Sometimes there is no time information at all but the rows in x are just numbered consecutively. The correct temporal spread is calculated inside the function when the measurement interval is specified here. Defaults to 1 which expects times to be correctly given in x. |
| wndw | Numeric between 0 and 1. Relative width of a moving window in which the standard deviation of the concentrations is calculated to identify high frequency fluctuations. See details and next. |
| pdk | Numeric between 0 and 1. Minimum proportion of data points to be kept. See details. In case one single concentration value occurs more than pdk * n times in the data (may happen under zero to very low flux conditions), all other data is assumed to represent high frequency fluctuations and flux is set to zero. |
| min.dp | Numeric. The minimum number of data points. Defaults to 20. If there are less rows the estimation is run anyway but a warning is issued and min.dp is automatically adjustet to n-1. |
| nrmse.lim | The maximum acceptable normalized root mean square error. Numeric value between 0 and 1. Defaults to 0.1. If the final best solution has a higher nrmse it is flagged accordingly. |
| r2.qual | Numeric between 0 and 1. Quality parameter for the model fit. The minimum acceptable $R^2$ of the best fitted model. Defaults to 0.8. When the value is below quality setting a quality flag is reported. |
| range.lim | Numeric. The minimum range of the concentration measurements during one chamber placement. The acceptable range limit depends on the accuracy of the concentration measurements. When the range of the concentration measurements is smaller than the repeatability range of the measurement device one cannot tell real increase in concentration from random fluctuation. Therefore, if the range of the concentration measurements during one chamber placement is < range.lim, the range.flag is set to FALSE (0). See details. |
| out.unit | Character string determining the output unit of the flux rate mass part. The default "auto" tries to find a unit that ranges the output value between 0.01 and 10. Possible output units are "ng", "mug", "mg", or "g". "mug" stands for "$\mu$g" because non-ascii characters are not allowed in functions. Beware of varying mass units in your output when running in auto mode. |
| elementar | When the fluxes are wanted as element values set elementar = TRUE. Defaults to FALSE. |

hardflag — Named list that controls which of the quality flags are to be hard flagged (the value is changed according to the quality flag). Only range.lim is hard flagged by default. So when the range of concentration values for a chamber measurement is smaller than the set range limit a zero flux is returned. When a flux estimation does not meet the quality requirements of any other hard-flagged quality parameter fluxx returns NA. For changing a quality flag to a hard flag just provide its name (without quotation marks) and set it to TRUE. Possible parameters to choose as hard or soft flag are nrmse, range, and r2. Further the number of measurements below ambient (nomba) can be hardflagged by setting nomba = [0. . . nc] with nc = number of concentration measurements for one chamber placement.

consecutive — Shall the most linear part be found by a consecutive approach starting at the first concentration reading. As soon as a stable flux is detected, it is stored. Strictly experimental.

## Details

The function is similar to flux but uses a different algorithm to identify the most linear part of the concentration development. First high frequency fluctations are omitted. Then all possible pdk * n : n consecutive concentration measurements are regressed against the corresponding times. The model with the highest r2 is chosen.

var.par specifies the variables within x and fixed parameters for all chamber placements that are used for the flux estimations. For obligatory var.par items see flux and examples. In contrast to flux there is just one workhorse function doing the actual estimation (mf.flux) per data table. Especially when there are many data tables in x and/or many data points per data table it takes some time. Progress is shown in the console. Each dot represents one finalized data table.

## Value

fluxx returns a complex object of class fluxxes that is a 2 entry list. When the object is printed to the console only the second entry is displayed in a modified form that is meant to maximize information display with small footprint for easy inspection. A table is printed to the console with three columns per gas. The first contains the quality flags (e.g. 111.9). The order is: r2.f, range.f,nrmse.f, nomba.f. The first three are considered more important, and if they are '1' everything is fine. The last number digit following the full stop gives the number of concentration readings below ambient.

The data.frame with the estimated flux rates contains all data needed for further analysis. The columns represent the entries in fluss of the single chamber measurements (including quality flags, see below) plus naming information according to the settings in the nmes argument of chop. export provides a simple way to export the results.

The first entry is itself a list of lists and data tables. It is called flux.res. The only one first level entry in this list contains the information for one gas which is itself a list. In this list each first level entry contains the information for one chamber measurement. It is named according to the nmes-setting in chop and contains the elements fluss (which is itself a list with the elements given below), mod, out (a list with hand through data, list items according to columns in x that have been handed trough via all.through or var.par), and inn - a data.frame with the input data that were relevant for estimating the flux (the obligatory part of var.par).

The elements of fluss:

| | |
|---|---|
| ghg | Character. The gas species for which the flux has been estimated. |
| flux | Numeric. Calculated flux rate in mass unit per m2 and hour. |
| r2 | r2 of the best fitted model that has been used for flux caclulation. |
| nrmse | nrmse of the best fitted model that has been used for flux calculation. |
| r2.f | Logical. r2 quality flag telling whether the r2 quality setting given in r2.qual is fulfilled. |
| range.f | Logical. Range quality flag telling whether the range of the concentration measurements exceeded the quality range of the measurement device that has been specified in range.lim. |
| nrmse.f | Logical. nrmse quality flag telling whether the nrmse quality setting given in nrmse.lim is fulfilled (i.e. if the nrmse of the best model <= nrmse.lim). |
| nomba.f | Integer. Reports the **n**umber **o**f **m**easurements **b**elow **a**mbient. When one observes concentrations below ambient that might make the measurements unstable, it is possible to filter the results later and allow only a maximum acceptable number of measurements below ambient. The ambient concentration is build into the function with data from Mace Head Ireland (N2O, CH4) and global average (CO2) obtained from http://cdiac.ornl.gov/pns/current_ghg.html as of August 1st, 2011. |
| unit | The mass unit assigned. |
| podpu | Proportion (expressed as a number between 0 and 1) of data points used for constructing the linear model for estimating the flux rate. The higher the less disturbed the measurements. |

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

## References

Nakano T (2004) A comparison of regression methods for estimating soil-atmosphere diffusion gas fluxes by a closed-chamber technique. Soil Biology and Biochemistry 36: 107-113.

Forbrich I, Kutzbach L, Hormann A, Wilmking M (2010) A comparison of linear and exponential regression for estimating diffusive CH4 fluxes by closed-chambers in peatlands. Soil Biology and Biochemistry 42: 507-515.

## See Also

gpp and reco for further processing of the results.

## Examples

```
## Not run:
## load data
data(tt.nee)

## prepare flux estimation
# make parts with chop
```

```
tt.parts <- chop(tt.nee, factors=c("session", "spot"),
nmes=c("spot", "date", "session"), min.cm=40)
# prepare var.par list (like with flux)
vp <- list(CO2 = "NEE", time = "datetime", area = "area",
volume = "volume", t.air = "t.cham", p.air = 101325)

## do the flux estimation
# run fluxx. with lots of data it may take a while
# (approx. 10 sec per chamber)
tt.flux <- fluxx(tt.parts, subset=c(1:30), vp, pdk=0.5,
range.lim=3, out.unit="mg")
# inspect results table
tt.flux
# plot diagnostic plots
plot(tt.flux, dims=c(4,4), subs="spot")
# run fluxx with alternative method
tt.fluxa <- fluxx(tt.parts, subset=c(1:30), vp, pdk=0.5,
range.lim=3, out.unit="mg", method="rmse")
# inspect results
tt.fluxa

## End(Not run)
```

---

`gflux` *Calculate gas flux rate from two concentrations*

---

## Description

Calculate gas flux rate from two concentrations using the ideal gas law to obtain a mass flow from an area per time. Therefore, besides the two concentrations ct and c0 the temperature within and the volume of the closed chamber are needed. For areal reference the area from which the gases are emitted has to be given. Without any further unit transformation the input unit directly gives the output unit: When concentration is coming in ppm the calculated flux rate is in $\mu$g/m2*h, when concentration is in ppb the flux rate will be in ng/m2*h

## Usage

```
gflux(ct, c0 = NULL, T, V, A = 1, M = 44, t = 1/60, p = 101325)
```

## Arguments

| | |
|---|---|
| `ct` | Concentration of the gas at time t. When the function is used internally the concentration is automatically derived from a regression model. May also be the concentration change in time dc/dt. In this case `c0` must not be specified. |
| `c0` | Concentration of the gas at time 0. When the function is used internally the concentration is automatically derived from a regression model. |

| | |
|---|---|
| T | Temperature within the chamber during the measurement in °C (it is converted automatically to Kelvin). When it is changing during the measurement typically the average temperature is used. However, if there is too much change in temperature during the chamber closing time (more than 5 °K) the ideal gas law might not longer be appropriate. |
| V | Chamber headspace volume. Because concentrations are typically small volume matters and should therefore be determined as exactly as possible. |
| A | Area covered by the measurement chamber. Defaults to 1. For dimensionless sampling just leave the default. As with the volume it matters a lot for the end value, therefore it should be determined as exactly as possible. |
| M | Molar weight of the gas for wich concentration data is given. Defaults to 44 because two of the three most commonly considered greenhouse gases share this molar weight ($N_2O$ and $CO_2$). When calculating the flux rate from methane concentrations change accordingly (16 g/mol). |
| t | Chamber closing time or more exactly the time span between the measurement of c0 and ct. When derived from a regression model this might not be the whole chamber closing time. |
| p | The air pressure at earth surface during measurements. Default is given by the standard value at sea level of 101325 Pa. Should be OK for most lowland measurements. However, if the measurements took place on higher altitudes, it might be reasonable to adapt the air pressure value to local conditions. |

## Details

Typically the function will not be called separately. However, for checking single chamber measurements or for testing purposes it might be useful to have this as a separate function.

The flux rate is calculated using

flux.rate = ((ct-c0) * V * M * p) / (t * R * (T + 273.15) * A)

The gas constant R is used with its standard value 8.314 Pa/K*mol.

## Value

Returns one numeric value that represents the flux rate in mass unit (depending on input concentration) per $m^2$ and hour.

## Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>, Franziska Koebsch <franziska.koebsch@uni-rostock.de>

## See Also

flux, flux.conv

## Examples

```
## see examples for function flux
```

gpp *Model GPP from CO2 closed chamber flux data*

## Description

Model GPP from $CO_2$ closed chamber flux data under consideration of ecosystem respiration. Four different methods are available: Providing one global Reco model, providing several Reco models, providing estimated Reco fluxes via function gpp or extracting Reco fluxes from real measurements via gpp2. Timestamps are used to assign Reco data to the respective NEE data. In the latter case they have to be provided alongside the Reco fluxes.

## Usage

```
gpp(NEE, PAR, ts.NEE, PAR.Temp, Reco.m, ts.Reco = NULL,
method = "Michaelis-Menten", units = "30mins", allow.offset = FALSE,
virtual = FALSE, start.par = max(PAR), ...)

gpp2(NEE, PAR, ts.NEE, oot, oot.id = c("D", "T"),
method = "Michaelis-Menten", allow.offset = FALSE,
virtual = FALSE, start.par = max(PAR), ...)
```

## Arguments

| | |
|---|---|
| NEE | Numeric vector with $CO_2$ fluxes from measurements of $CO_2$ net ecosystem exchange (NEE). |
| PAR | Numeric vector of mean irradiation during $CO_2$ flux measurements. Can be supplied as photosynthetically active radiation (PAR) or photosynthetic photon flux density (PPFD). |
| ts.NEE | POSIXlt vector holding the timestamp of the NEE values. NEE and Reco values are linked to each other based on their timestamps specified in ts.NEE and ts.Reco or elsewhere (depending on method). The two timestamps don't have to match exactly. In that case, the function links NEE and Reco values based on the time interval given in units. |
| PAR.Temp | Either numeric vector of mean recorded temperature readings during $CO_2$ flux measurements or data.frame with several temperature records (if Reco.m is provided as an object resulting from running reco.bulk). In case of the latter, appropriate temperatures are extracted based on the which.Temp parameter that is stored to the model structure that is returned by reco.bulk. Therefore names have to correspond with the particular temperature variable names used in $R_{eco}$ modeling (e.g., air, soil in -2cm/-5cm/-10cm depth). See details. |
| Reco.m | Model structure obtained from running reco.bulk or reco or vector with estimated Reco values. The latter has to contain (at least) the values that are valid at the times of the NEE measurements. Typically the data will result from estimating hourly or half-hourly $R_{eco}$ values using budget.reco on objects of class "breco" derived from running reco.bulk. See details. |

`ts.Reco` POSIXlt vector holding the timestamp of the $R_{eco}$ values. Has to be specified if $R_{eco}$ values instead of an $R_{eco}$ model are given in `Reco.m`. The function assumes that this is the case if `ts.Reco != NULL`. See details.

`method` The function knows several equations to model the relationship between gpp and irradiation. At the moment `"Michaelis-Menten"`, `"Falge"`, `"Smith"`, and `"Misterlich"` are implemented which are all discussed in Falge et al. 2001. Partial matching is applied. Defaults to `"Michaelis-Menten"`. See details for equations.

`units` Character string specifying how `ts.NEE` shall be rounded. If `Reco.m` holds values instead of an $R_{eco}$ model structure the NEE and $R_{eco}$ values are matched based on their timestamp after `ts.NEE` has been rounded according to `units`.

`allow.offset` Logical. Shall GPP values other than 0 be allowed at zero irradiation? See details.

`virtual` Logical. If `TRUE`, virtual NEE data are generated that show a typical saturation curve with saturation at mean `NEE`. Can be used in bulk gpp modeling to allow falling back to a mean model.

`start.par` Numeric between 0 and `max(PAR)`. All data points with `PAR <= start.par` are used to obtain a start value for alpha via linear regression. Defaults to `max(PAR)`. See details.

`...` Any arguments passed to `nls` which is used internally to do the model fitting.

`oot` Vector of length = `length(NEE)` specifying which of the measured fluxes derive from opaque ($R_{eco}$) and which derive from transparent (NEE) chamber measurements. `gpp2` uses this to extract corresponding $R_{eco}$ values for calculating GPP from NEE before fitting the models (Approach 1, see details.)

`oot.id` Vector of length 2 that specifies which of the flux values derive from opaque (first value, i.e. $R_{eco}$ measurements) and which derive from transparent (second value, i.e. NEE measurements) chamber measurements when `data` contains both. May be character, factor, or numeric. See details.

## Details

The function models the relationship between $CO_2$ uptake by plants (gross primary production, GPP) and irradiation using one out of 4 methods (Falge et al. 2001). Per default the Michaelis-Menten kinetic (e.g., Schmitt et al. 2010) is used. The following models can be fitted to the data:

$GPP = \frac{GPmax*alpha*PAR}{alpha*PAR+GPmax}$ (Michaelis-Menten)

$GPP = \frac{alpha*PAR}{1-\frac{PAR}{2000}+\frac{alpha*PAR}{GPmax}}$ (Falge)

$GPP = \frac{GPmax*alpha*PAR}{\sqrt{GPmax^2+(alpha*par)^2}}$ (Smith)

$GPP = GPmax * (1 - e^{\frac{alpha*PAR}{GPmax}})$ (Misterlich)

with `PAR` the incoming light (irradiation). Note, that irradiation can be given in `PAR` or in PPFD although the equation states `PAR`. `GPmax` and `alpha` are the parameters that are fitted. `GPmax` refers to the maximum gross primary production at saturating or optimum light whereas alpha refers to the ecosystem quantum yield and gives the starting slope of the model.

Transparent closed chamber measurements in the field typically capture net ecosystem exchange (NEE), which is the sum of the two opposing processes ecosystem respiration ($R_{eco}$) and GPP. Therefore, it is necessary to subtract modeled $R_{eco}$ from the measured NEE to obtain GPP that can be used for the modelling against irradiance.

Real $R_{eco}$ at the time of the NEE measurement is typically unkown because dark and light measurements cannot be taken at the same spot at the same time. Therefore, $R_{eco}$ has to be modelled based on dark chamber or nighttime measurements (see reco). For modelling GPP from NEE chamber measurements, gpp just needs measured NEE, the associated irradiance (PAR) and temperature (PAR.Temp) values and the $R_{eco}$ model(s) (Reco.m). The $R_{eco}$ model(s) can derive from a longer period of time than the NEE data, which is often better to get more reliable models. In contrast, gpp2 extracts $R_{eco}$ fluxes from actual measurements.

Approaches to assigning $R_{eco}$ values:

**Approach 1:** Extract corresponding $R_{eco}$ fluxes from the provided data that are assigned to corresponding NEE values via their timestamp: For this approach NEE has to contain both NEE and $R_{eco}$ fluxes. oot has to be specified as a vector that indicates whether the respective fluxes were measured as NEE (transparent chamber) or Reco (opaque chamber or low PAR). In addition oot.id may have to be changed accordingly. gpp2 is used for fitting the models.

**Approach 2:** If Reco.m is specified as a vector containing modelled $R_{eco}$ values these are used to calculate GPP = NEE + Reco. The correct $R_{eco}$ values are assigned to the appropriate NEE values by rounding the timestamp of the latter (given in ts.NEE) according to the time lapse of the $R_{eco}$ values and then merging both on the respective timestamps. Therefore ts.Reco has to be specified while PAR.Temp is ignored.

**Approach 3:** If just one $R_{eco}$ model is provided as an object of class "reco" resulting from running reco this is used to predict $R_{eco}$ at the times of the NEE measurements with the temperatures provided in PAR.Temp as new.data. PAR.Temp has to be specified as a vector of length = length(NEE). ts.Reco must not be specified.

**Approach 4:** If several $R_{eco}$ models are provided as an object of class "breco" resulting from running reco.bulk these are used to predict $R_{eco}$ at the times of the NEE measurements with the temperatures provided in PAR.Temp as new.data. PAR.Temp has to be provided as a data.frame with all temperature variables that were used when obtaining the $R_{eco}$ models via reco.bulk with ncol(PAR.Temp) = length(NEE). The appropriate temperatures are assigned using the parameter which.Temp that is reported with each model in an object of class "breco". ts.Reco must not be specified.

The Michaelis Menten fit to the GPP/PAR relationship presumes that plants (at least C3 plants) do not take up $CO_2$ when there is no irradiance. However, sometimes the $R_{eco}$ model gives quite unrealistic $R_{eco}$ estimates for the times of NEE measurements leading to an alleged considerable uptake of $CO_2$ under no or very low light conditions. This in turn leads to unrealistic and not well fitted GPP models. Therefore, it is possible to correct the model by not allowing an offset: allow.offset = FALSE (default). The offset is determined automatically by constructing a linear model using the data points until PAR = start.par and predicting GPP at PAR = 0. The offset is then subtracted from all GPP values and is later automatically added when doing the diagnostic plots.

The start parameters for the non-linear fit (via nls) are derived from the data itself. For alpha (initial slope of the model) the slope of the linear model of GPP against PAR constructed from the data points until PAR = start.par is used. For GPmax the mean of the five highest GPP values is taken.

It is advisable to test various configurations regarding the $R_{eco}$ model and testing the effect of allowing the offset. ATTENTION: The offset is not added back to the predicted GPP data but it is returned as part of the output (see value section). Therefore, if the model parameters and model formula are used to predict GPP fluxes, the offset has to be added manually.

## Value

The function returns an object of class gpp (for `ts.Reco != NULL`) or of class gpp2 (for `ts.Reco = NULL`). It is a list with the following components.

| | |
|---|---|
| `mg` | The gpp model. A `nls` model structure. |
| `mr` | The Reco model used. A `nls` model structure. |
| `data` | Either a three entry list (with `ts.Reco != NULL`) or a 4 entry list (with `ts.Reco = NULL`) |
| `dat` | `data.frame` (see below for more). |
| `offset` | Numeric value giving the offset. |
| `start` | List with the start values for the gpp modelling. |
| `PAR.Temp` | Numeric vector with the `PAR.Temp` values specified in the function call. Only reported if `ts.Reco = NULL`. |

The `data.frame` in `dat` contains the following columns:

| | |
|---|---|
| `NEE` | `NEE` values. |
| `GPP` | Corresponding `GPP` values. |
| `Reco` | Corresponding $R_{eco}$ values. |
| `PAR` | Corresponding `PAR` values. |
| `timestamp` | Corresponding timestamps. |
| `mins` | Temporal distance to next reco value. Always 0 but reported for consistency with `gpp2`. |
| `Reco` | Numeric vector of corresponding $R_{eco}$ values estimated with the $R_{eco}$ model (`Reco.m`). |

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

## References

Falge E, Baldocchi D, Olson R, Anthoni R, et al. 2001. Gap filling strategies for defensible annual sums of net ecosystem exchange. Agricultural and Forest Meteorology, 107:43-69.

Schmitt M, Bahn M, Wohlfahrt G, Tappeiner U, Cernusca A. 2010. Land use affects the net ecosystem CO2 exchange and its components in mountain grasslands. Biogeosciences, 7:2297-2309.

## See Also

`reco`, `fluxx`

## Examples

```
## load data
data(tt.flux)

## make timestamp
tt.flux$timestamp <- strptime(paste(tt.flux$date, tt.flux$time),
format="%Y-%m-%d %H:%M:%S")

## model reco with Arrhenius type model
# extract data and omit estimated fluxes with both the nrmse
# and the r2 flag set to 0
ttf <- tt.flux[!(tt.flux$CO2.r2.f + tt.flux$CO2.nrmse.f) == 0, ]

# extract table with flux data for reco modeling
ttf4reco <- subset(ttf, kind > 4)

# omit CO2 fluxes below zero
ttf4reco <- ttf4reco[ttf4reco$CO2.flux >= 0,]

# plot reco data
plot(CO2.flux ~ t.air, data=ttf4reco)

# check for the best temperature for reco modelling
temps <- c("t.air", "t.soil2", "t.soil5", "t.soil10")
sapply(temps, function(x) lapply(reco(ttf4reco$CO2.flux,
ttf4reco[,x], method="arr"), AIC))

# take the temperature in soil 2 cm
reco.m <- reco(ttf4reco$CO2.flux, ttf4reco$t.soil2, method="arr")

# inspect
reco.m

## model gpp
# extract table with flux data for gpp modeling
ttf4gpp <- subset(ttf, kind < 4)

# do a single gpp model for a measurement day using data of spot 2
tmp <- ttf4gpp[(ttf4gpp$date=="2011-05-11") & (ttf4gpp$spot==2),]
gpp.m1 <- gpp(tmp$CO2.flux, tmp$PAR, tmp$timestamp, tmp$t.soil2,
reco.m[[1]])
# check diagnostic plot
plot(gpp.m1)

# same for spot 3
tmp <- ttf4gpp[(ttf4gpp$date=="2011-05-11") & (ttf4gpp$spot==3),]
gpp.m2 <- gpp(tmp$CO2.flux, tmp$PAR, tmp$timestamp, tmp$t.soil2,
reco.m[[1]])
# check diagnostic plot
plot(gpp.m2)

# same with all three spots
```

```
tmp <- ttf4gpp[(ttf4gpp$date=="2011-05-11"),]
gpp.m3 <- gpp(tmp$CO2.flux, tmp$PAR, tmp$timestamp, tmp$t.soil2,
reco.m[[1]])
# check diagnostic plot
plot(gpp.m3)
```

---

`inspect` *Inspect and alter prepared ghg concentration data*

---

## Description

The function allows straightforward inspection and alteration of ghg concentration data that have been prepared using chop

## Usage

```
inspect(x, what, retain = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | Object returned by chop containing tables with chamber measurement data. |
| `what` | Specifies the concentration measurement tables in `x` that ought to be inspected/altered. For inspection either give a numeric vector with the indices of the tables or a character vector with the names of the entries in `x`. For altering data tables (i.e. deleting some values) use a named list. The names refer to the names of the tables and the entries are either numeric values or vectors that specify the concentration measurements that shall be skipped. The function allows lazy naming so it is possible to provide just the parts of the names that identify a certain measurement unambiguously and checks for consistency. |
| `retain` | Logical. When you alter `x` by giving a list structure to `what`, shall the original data tables be returned as well? Defaults to FALSE. |

## Details

The typical workflow is to look at the diagnostic plots of the results of the `flux` estimation and then turn to `inspect` for having a closer look at the data or to delete some concentration measurements for further estimations.

## Value

Either the data tables to inspect are returned in a list or the altered `x`. In case `retain = TRUE` the original tables are appended to `x`. The respective list item is `tables.orig`.

## Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

## See Also

flux, chop, (also for examples)

---

lips *Linear interpolation between data points similar to approx.*

---

## Description

Linear interpolation between data points similar to approx. x may be a time vector.

## Usage

```
lips(x, y, x.step = 1)
```

## Arguments

| | |
|---|---|
| x | Numeric vector or one that could be coerced to numeric along which interpolation shall take place. May be a time vector (POSIXlt or POSIXct). |
| y | Numeric vector of values which shall be interpolated. |
| x.step | Numeric giving at what time interval interpolation shall be done. In seconds! Thus half hourly interpolation is achieved with x.step = 1800. |

## Value

Data.frame containing the interpolated x (x.out) and y (x.out) values.

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

## See Also

approx

## Examples

```
## has to be added
```

`modjust` *Adjust Reco models*

## Description

The function allows to adjust fitted $R_{eco}$ models by eliminating the maximum $R_{eco}$ flux as long as the p.value of the linear model of the residuals regressed against original fluxes is above a given threshold. In addition models with parameters that went astray may be skipped. The default is that $R_{eco}$ models with t1 > 20 are omitted.

## Usage

```
modjust(models, alpha = 0.1, minimum = 0.8, prmtrs = list(t1 = 20), ...)
```

## Arguments

| | |
|---|---|
| `models` | Object of class "`breco`". |
| `alpha` | Alpha level against which the p.value of the linear model of the residuals against original fluxes shall be tested. |
| `minimum` | The minimum proportion of data points that should be kept. The optimisation runs in a `while` loop until the p.value is below `alpha`. It may happen - especially when the number of data points was already low from beginning - that many data points are skipped before a solution is reached. This is prevented by this argument which acts as a brute force to the process and stops it. |
| `prmtrs` | List object that allows to skip models according to thresholds set for coefficients of the fitted regression models. The list has to be set up according to the actual method used in `reco` an the names refer to the names of the corresponding coefficients. The default is that $R_{eco}$ models with t1 > 20 are omitted. |
| `...` | Arguments passed through to `reco` which is used to fit the models again based on the adjusted data. |

## Details

When fitting $R_{eco}$ models based on one or few measurement campaigns in the field it may happen that outliers in the extremes of the temperature gradient have a very high influence on the fit. Although the model could be fit in the first place this often leads to unrealistic predicted fluxes. The adjustment via `modjust` leads to better overall performance and reliability of the bulk modelling.

## Value

Returns a "`breco`" object with the possibly adjusted models. All returned models gain a list entry within the `mod` object (see `reco` and `reco.bulk`) named `n.out.adj` giving the number of omitted data points. Fall.back models (see `reco.bulk`) in `models` are left untouched.

### Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>,

based on ideas by Sascha Beetz, <sascha.beetz@uni-rostock.de>

### See Also

reco, reco.bulk

### Examples

```
## See examples at reco.bulk
```

---

## plot.fluss *Functions for bulk plotting of concentration change with time as kind of diagnostic plots for flux rate calculations*

---

### Description

Bulk plotting of concentration change with time adding color and symboling for acting as a diagnostic plot for the flux rate estimation functions (flux, flux.odae, flux.conv) in this package.

### Usage

```
## S3 method for class 'fluss'
plot(x, subs, dims, folder = getwd(), xlims = NULL, ...)

## S3 method for class 'flux'
plot(x, zero.line, note = "", margin = 0.2, xlims = NULL, ...)

## S3 method for class 'fluxes'
plot(x, dims, ghg = "all", subs = NULL, folder = getwd(),
xlims = NULL, ask = TRUE, ...)

## S3 method for class 'fluxx'
plot(x, ...)

## S3 method for class 'fluxxes'
plot(x, dims, subs = NULL, folder = getwd(), ask = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | Object of class fluss that is returned by flux or object of class flux that is returned by flux.conv. In case of the latter, the function has to be applied to a list of flux extimation results via lapply or the like. |

| | |
|---|---|
| subs | Single character value or character value specifying the factors that shall be used for subsetting the plots into plates (that are stored as pdf files to a folder specified in folder). Must be names of columns of the original data that have been used to partition the data into chunks and that are part of the naming of the data chunks (see flux for details.). When there are only few chamber measurements that shall be plotted to the screen set subs to NULL. This is default behaviour for plot.fluxes |
| dims | Integer vector with two elements that specify the mfrow setting (see par for details) during the plotting of the single plates into pdf files. For all single plots to fit on the plate the product of the two entries has to be equal or higher the number of chamber measurements that are in the data in each partition according to subs. |
| folder | Character string giving the path to the folder were the files have to be stored. The names of the pdf files are generated automatically. |
| xlims | Two entry numeric vector specifying the x-axes limits for all plots. Defaults to NULL in which case it is derived from the data itself. The y-axes limits are always set according to the range of the concentration data $\pm$ 20 |
| ... | further arguments passed through to plot.flux (see below) or to plot.default. |
| zero.line | The y-axes position of a horizontal line that reflects the ambient concentration of the plotted gas species. When using plot.fluss this is determined automatically from x. |
| note | A note that shall appear in the plots. Typically not a fixed value but a value that changes from plot to plot. See example. |
| margin | Numeric between 0 and 1. Specifies the empty space within the diagnostic plots on the y-axis. |
| ghg | Character value or an up to three entry vector specifying which ghg should be plotted. Note that only ghg fluxes that were estimated can be plotted. |
| ask | Logical; if TRUE, the user is asked before starting to plot the concentration data for the next ghg, see par(ask=.) and examples. |

## Details

Typically plot.fluss will be used. However, for lower level plotting the function plot.flux that also does the plotting within plot.fluss is provided as a separate function.

## Value

The function is invoked for its side effects and does not return anything.

## Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

## See Also

chop, flux, flux.odae, flux.conv

## Examples

```
## load example data
data(tt.pre)

## extract field concentration measurements
gcd <- tt.pre[tt.pre$sampletype_a=="P",]

## partition the data into data tables per chamber measurement
# then do the partitioning
gcd.parts <- chop(gcd, factors = c("date", "spot", "veg"),
nmes = c("date", "veg", "spot"))

## calculate flux rates for methane
# first define CH4 range limit (alternatively use flux.calib)
CH4.lim <- 30
# do the flux rate estimation
vp.CH4 <- list(CH4 = "CH4ppb", time = "time_min", CH4.gcq = "CH4Code",
volume = "cham_vol", t.air = "temp_dC", area = "cham_area", p.air = 101325)
flux.CH4 <- flux(gcd.parts, var.par = vp.CH4, co2ntrol = NULL,
range.lim=CH4.lim)

## look at the results table
flux.CH4

## plot the concentration-change-with-time-plots as kind of diagnostic
plot(flux.CH4, dims = c(3,6))
```

---

`plot.gpp` *Plot diagnostic plots for GPP (NEE) models derived with reco and gpp.*

---

## Description

Plot diagnostic plots for GPP (NEE) models derived with reco and gpp.

## Usage

```
## S3 method for class 'gpp'
plot(x, nm = "", single.pane = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object of class gpp that is returned by `gpp`. |
| `nm` | The three panels of the resulting plot are already named. However, if you'd like to add something you can do it here. |
| `single.pane` | For bulk plotting of several models to one device it is necessary to `FALSE` `single.pane`. See example at `gpp` |
| `...` | Further arguments passed to `plot.default`. |

### Details

The function produces a three panel plot representing in this order from left to right: (1) $R_{eco}$ plot and the used $R_{eco}$ model. (2) Combined plot of the NEE/GPP data with the measured NEE vs PAR, the derived GPP and the modelled $R_{eco}$. (3) Diagnostic plot of $NEE_{measured}$ vs $NEE_{predicted}$

### Value

The function is invoked for its side effects and does not return anything.

### Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

### See Also

chop, fluxx, gpp, reco

### Examples

```
## see examples at gpp
```

---

## plot.reco *Plot diagnostic plots for Reco models.*

---

### Description

Plot diagnostic plots for $R_{eco}$ models.

### Usage

```
## S3 method for class 'reco'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object of class reco that is returned by reco. |
| ... | Further arguments passed to plot.default. |

### Details

The function produces a plot of the reco model.

### Value

The function is invoked for its side effects and does not return anything.

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

## See Also

chop, fluxx, gpp, reco

## Examples

```
## see examples at gpp
```

---

reco *Model R_eco from CO2 exchange closed chamber data.*

---

## Description

Model $R_{eco}$ from CO2 exchange closed chamber data.

## Usage

```
reco(R, Temp, Tref = 10, T0 = -46.02, method = "all", min.dp = 6)
```

## Arguments

| | |
|---|---|
| R | Numeric vector with ecosystem respiration ($R_{eco}$) flux rates. |
| Temp | Numeric vector with corresponding temperature values. |
| Tref | Numeric value giving the reference temperature used in the Arrhenius type model. Defaults to 10 (°C). |
| T0 | Numeric value giving the activation temperature used in the Arrhenius type model. Defaults to -46.02 (°C). |
| method | Specifies the model to be used. Partial match is possible. One can either check all included models ("all"), select the best performing model (according to AIC; "best"), or do all models that do not fail in fitting ("not.failed") or specify one particular model: "linear", "arrhenius", "Q10", "lt" (Lloyd & Taylor), "ltr" (Lloyd & Taylor restricted), "logistic". See details. |
| min.dp | Integer. Minimum number of data points accepted. If number is below function execution is stopped and a warning is issued. |

## Details

Respiration is controlled by both biological and physical factors. Work by Arrhenius and van't Hoff in the late-19th century on the temperature dependence of chemical reactions lead to the insight that there is a certain relationship between temperature and respiration (see review by Lloyd and Taylor, 1994). The most prominent models that have been used extenively in the literature can be fitted with this function. For an in-depth review, even more models and references see Richardson et al. 2006.

Models (T = Temperature):

| | |
|---|---|
| `linear` | $R = \theta_1 + \theta_2 * T$ |
| `arrhenius` | $R = \theta_1 + exp\left[E_0\left(\frac{1}{T_{Ref}-T_0} - \frac{1}{T-T_0}\right)\right]$ |
| `Q10` | $R = \theta_1\theta_2^{(T-T_{Ref})/10}$ |
| `lt` | $R = \theta_1 exp\left(\frac{-\theta_2}{T+273.15-\theta_3}\right)$ |
| `ltr` | $R = \theta_1 exp\left(\frac{-308.56}{T+46.02}\right)$ |
| `logistic` | $R = \frac{\theta_1}{1+exp(\theta_2-\theta_3 T)}$ |

## Value

Either returns a list of $R_{eco}$ models or the specified model structure. The wanted or resultant model can be fed into gpp or used on its own to predict Reco values.

## Note

In its current implementation the `lt` and `logistic` models are easily over parameterized and therefore find singular gradients and provide no fit.

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

## References

Lloyd J, Taylor JA, 1994. On the temperature dependence of soil respiration. Functional Ecology 8:315–323.

Richardson et al. 2006. Comparing simple respiration models for eddy flux and dynamic chamber data. Agricultural and Forest Meteorology 141:219–234.

## See Also

gpp

## Examples

```
## see examples at gpp
```

---

`reco.bulk` *Bulk fitting of Reco and GPP models*

---

## Description

The function allows for bulk fitting of $R_{eco}$ and GPP models with the respective functions `reco` and gpp. This is often appropriate because data are gathered over a season, a year or longer...

## Usage

```
reco.bulk(formula, data, INDEX, window = 1, hook = "mean", remove.outliers = FALSE,
fall.back = TRUE, ...)

gpp.bulk(formula, data, INDEX, window = 1, hook = "mean", oot.id = c("D", "T"),
min.dp = 5, Reco.m = NULL, ts.Reco = NULL, fall.back = TRUE, ...)
```

## Arguments

| | |
|---|---|
| formula | An object of class "formula" (or one that can be coerced to that class): a symbolic description of the terms that are used in bulk $R_{eco}$ and GPP model fitting. Choices of terms are more restricted than typically (see details). For instance, a timestamp always has to be provided. Also, temperature variables are required for gpp.bulk if $R_{eco}$ values are predicted from models. |
| data | A data frame (or an object that can be coerced to that class by as.data.frame) containing at least all the 'model' terms specified in formula. |
| INDEX | A vector of length nrow(data) that is used to extract and compile data, for instance according to measurement campaign in the field. Internally split is used with f = INDEX to create a list of data.frames of which each contains all flux measurements for one model. |
| window | Both functions can fit the respective models across a moving window of adjacent INDEX values. Not advisable for GPP while $R_{eco}$ modelling can really profit because more data points often lead to better models. |
| hook | Character string specifiying the kind of summary statistics used to fix a date and time to which the fitted model shall refer. Up to now this is simply achieved by doing one of these summary statistics on the timestamp: mean, min, max or median. |
| remove.outliers | Logical. If TRUE the function searches for outliers in the data points of the $R_{eco}$ models and eliminates them. Per model the boxplot.stats of the residuals are obtained and if outliers are present they are eliminated and the model is fitted again. This is done twice. If the function fails in fitting the model to the new data set it falls back to the original data. |
| oot.id | Vector of length 2 that specifies which of the flux values derive from opaque (first value, i.e. $R_{eco}$ measurements) and which derive from transparent (second value, i.e. NEE measurements) chamber measurements when data contains both. This is one of several approaches to GPP modeling here. See details. |
| min.dp | Numeric. Specifies the minimum number of data points that are accepted per model. Defaults to 5 which is already quite a small number. |
| Reco.m | Either an object of class "reco" resulting from reco with one $R_{eco}$ model or an object of class "breco" resulting from reco.bulk with several $R_{eco}$ models or a vector with estimated half hourly or hourly (or whatever interval you have) $R_{eco}$ values. In case of the latter ts.Reco has to be specified as well because it is also used as a switch between internal $R_{eco}$ modeling and assigning existing $R_{eco}$ values. See details. |

| | |
|---|---|
| ts.Reco | POSIXlt or POSIXct vector with timestamps of the fluxes in Reco.m. Further, the default (ts.Reco = NULL) lets the function expect model object(s) in Reco.m. |
| fall.back | Logical. When TRUE the function falls back to linear mean models when the non linear approach did not work out (for reco.bulk: the slope of the linear relationship between Reco and temperature is < 0; for gpp.bulk: either no model could be fit or the starting slope parameter alpha is > 0). To do so a virtual data set is created with 50 random GPP values that have the same mean and sd as the original data and with a sequence of 50 PAR values spanning from 0 to 2000. A linear model is fit to these data with lm(GPP ~ PAR). |
| ... | Further arguments passed to reco or gpp e.g., the method for fitting the model when not using the respective defaults. |

## Details

Models are - comparable to regression models - specified symbolically. Accordingly, the basic form is response ~ terms with response always referring to $CO_2$ exchange rates. For terms requirements differ between the two methods. In contrast to other formulae the response and all terms have to be in data.

reco.bulk expects a formula of the form Reco ~ T1 + ... + timestamp with Reco referring to $CO_2$ fluxes estimated based on opaque chamber measurements (for instance with flux), T1 referring to temperature readings relevant for Reco (e.g. air temperature) and taken during the corresponding chamber measurements. The ... symbolizes that several more temperature readings can be specified if available (e.g. temperature in soil at 2cm), as many as you want. When more than one temperature is specified models are fit for each temperature and the best one is determined via AIC and reported together with the name of the corresponding temperature variable. Finally, timestamp is referring to the POSIXt timestamps that represent the dates and times of the corresponding measurements. timestamp always has to be specified as the last term of the formula. Models are fit using reco.

gpp.bulk expects a formula of the form NEE ~ PAR + timestamp + ... with NEE referring to $CO_2$ fluxes estimated based on transparent chamber measurements (for instance with flux), PAR referring to readings of the photosynthetically active radiation relevant for NEE and taken during the corresponding chamber measurements. The ... symbolizes that several more terms can or have to be specified. This depends on the approach to the $R_{eco}$ part of the GPP modeling (see gpp).

Approaches to estimate GPP values from measured NEE data using corresponding $R_{eco}$ values:

**Approach 1:** Extract corresponding $R_{eco}$ fluxes from the provided data that are assigned to corresponding NEE values via their timestamp: For this approach data has to contain both NEE and $R_{eco}$ fluxes and the model formula is specified as NEE ~ PAR + timestamp + oot with the latter referring to a variable that indicates whether the respective fluxes were measured as NEE (transparent chamber) or Reco (opaque chamber or low PAR). In addition oot.id may have to be changed accordingly. gpp2 is used for fitting the models.

**Approach 2:** Provide measured $R_{eco}$ fluxes that are assigned to corresponding NEE values via their timestamp: To do this set ts.Reco != NULL and Reco.m a vector of $R_{eco}$ fluxes and specifiy model with: NEE ~ PAR + timestamp. gpp is used for fitting the models.

**Approach 3:** Provide one $R_{eco}$ model to predict $R_{eco}$ fluxes at the time of the NEE measurements using the same temperature variable that was used to construct the model (with reco). Specify model with: NEE ~ PAR + timestamp + temperature. gpp is used for fitting the models.

**Approach 4:** Provide several $R_{eco}$ models to predict $R_{eco}$ fluxes at the time of the NEE measurements using the same temperature variables that were used to construct the models (with reco.bulk). The corresponding models are assigned to the NEE data via the timestamps that they carry. Specify model with: NEE ~ PAR + timestamp + temperature1 + temperature2 + temperature3 + .... All temperatures that may have been used for fitting the $R_{eco}$ models (see above) should be given. gpp is used for fitting the models.

remove.outliers may result in better $R_{eco}$ models. One should be careful with this and watch out for cases in which too many data points are eliminated. The function returns the number of skipped outliers per model to do just that.

If fall.back = TRUE no failed model fits are reported. That's quite useful when further bulk methods like budget.reco or budget.gpp are used to get annual or seasonal budgets.

## Value

Both functions return complex list structures with models.

Output of reco.bulk: Object of class "breco", a list with length(unique(INDEX)) elements, each containing 3 elements:

| | |
|---|---|
| ts | Timestamp of the model. |
| mod | Has itself two elements. The first contains the model object as returned by reco and is named according to the method used. The second, n.out, is optional (only reported when remove.outliers = TRUE and there were indeed outliers identified and skipped) and gives the number of omitted data points. |
| which.Temp | Character string that identifies the temperature variable that was finally used for constructing the best model |

Output of gpp.bulk: Object of class "bgpp", a list with length(unique(INDEX)) elements each containing itself 2 entries:

| | |
|---|---|
| ts | Timestamp of the model |
| mod | Either an object of class "gpp" or of class "gpp2" depending on the approach used. Approaches 1 and 2 return "gpp2" objects, Approaches 3 and 4 return "gpp" objects. See gpp and gpp2 for details. |

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>,

with suggestions by Sascha Beetz, <sascha.beetz@uni-rostock.de>

## References

Beetz S, Liebersbach H, Glatzel S, Jurasinski G, Buczko U, Hoper H (2013) Effects of land use intensity on the full greenhouse gas balance in an Atlantic peat bog. Biogeosciences 10:1067-1082

## See Also

reco, gpp, gpp2, fluxx, modjust

## Examples

```
## Whole example is consecutive and largely marked as
## not run because parts take longer than
## accepted by CRAN incoming checks.
## Remove first hash in each line to run them.
data(amd)
data(amc)

### Reco ###
## do reco models with 3 campaign wide window and
## outlier removal (outliers according to models)
# first extract opaque (dark) chamber measurements
amr <- amd[amd$kind=="D",]

## Nor run ##
## do bulk fitting of reco models (all specified temperatures
## are tested and the best model (per campaign) is finally stored)
#r.models <- reco.bulk(flux ~ t.air + t.soil2 + t.soil5 +
#t.soil10 + timestamp, amr, amr$campaign, window=3,
#remove.outliers=TRUE, method="arr", min.dp=2)
#
## adjust models (BEWARE: stupid models with t1 >= 20 are skipped
## within the function, this can be changed)
#r.models <- modjust(r.models, alpha=0.1, min.dp=3)
#
## make data.frame (table) for overview of model parameters
## the temperature with which the best model could be fit is reported
## this information also resides in the model objects in r.models
#tbl8(r.models)
#
#### GPP ###
### fit GPP models using method = Falge and min.dp = 5
### and take opaque (dark, i.e. reco) measurements from data
## the function issues a warning because some campaigns have
## not enough data points
#g.models <- gpp.bulk(flux ~ PAR + timestamp + kind, amd, amd$campaign,
#method="Falge", min.dp=5)
#tbl8(g.models)
#
### alternative approaches to acknowledge reco when fitting GPP models
## we need only fluxes based on transparent chamber measurements (NEE)
#amg <- amd[amd$kind=="T",]
## fit gpp models and predict reco from models
#g.models.a1 <- gpp.bulk(flux ~ PAR + timestamp + t.air + t.soil2 +
#t.soil5 + t.soil10, amg, amg$campaign, method="Falge", min.dp=5,
#Reco.m=r.models)
#tbl8(g.models.a1)
## have a look the model fits (first 10)
#par(mfrow=c(5,6))
## select only non linear fits
#sel <- sapply(g.models.a1, function(x) class(x$mod$mg)=="nls")
#lapply(g.models.a1[sel][1:10], function(x) plot(x$mod, single.pane=FALSE))
```

```
#
## fit gpp models with providing reco data
## to do so, rerun budget.reco with other start and end points
#set.back <- data.frame(timestamp = c("2009-09-01 00:30", "2011-12-31 23:30"),
#value = c(-999, -9999))
#set.back$timestamp <- strptime(set.back$timestamp, format="%Y-%m-%d %H:%M")
#r.bdgt.a2 <- budget.reco(r.models, amc, set.back)
## now fit the models
#g.models.a2 <- gpp.bulk(flux ~ PAR + timestamp, amg, amg$campaign,
#method="Falge", units = "30mins", min.dp=5, Reco.m=r.bdgt.a2$reco.flux,
#ts.Reco = r.bdgt.a2$timestamp)
#tbl8(g.models.a2)
#
## End not run ##
```

---

`round.POSIXlt` *Round times.*

---

## Description

There are `round` methods in base for objects of `DateTimeClasses`. However, they can only round to full second, minutes, hours, and days. These functions offer some more options.

## Usage

```
## S3 method for class 'POSIXlt'
round(x, digits = c("mins", "5mins", "10mins", "15mins", "quarter hours",
"30mins", "half hours", "hours"))

## S3 method for class 'POSIXct'
round(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `DateTimeClasses` object to be rounded. Needs to carry a timezone specification. |
| `digits` | Either a character string specifying the time units to round to (see choices above) or a numeric specifying the minutes to round to. To go to seconds just use values < 1, to go beyond the hour just use values > 60. |
| `...` | Further arguments to methods. |

## Value

A `POSIXct` object with rounded, not truncated date times.

### Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>,

borrowing heavily from https://stat.ethz.ch/pipermail/r-help/2012-June/315336.html

### Examples

```
# Current time in GMT and as class "POSIXlt"
zlt <- as.POSIXlt(Sys.time(), "GMT")

# Same time as class POSIXct
zct <- as.POSIXct(zlt)

# round to minute
round(zct)

# round to half hour
round(zct, "30mins")
round(zct, "half hour")
round(zct, 30)

# round to 20 minutes
round(zlt, 20)

# round to 30 seconds
round(zlt, 0.5)
```

---

## tbl8 *Extract the relevant data of a bulk model object to a data.frame*

---

### Description

Extract the relevant data of an object of class "breco" to a data.frame

### Usage

```
tbl8(models)
```

### Arguments

| | |
|---|---|
| models | An object of class "breco" returned by reco.bulk or an object of class "bgpp" returned by gpp.bulk. |

### Value

Returns a data.frame with the model coefficients, brute force R2s (not reliable because non linear responses are fitted), timestamps and the which.Temp string or the offset for $R_{eco}$ models and GPP models, respectively.

## Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>, based on an idea by Sascha Beetz, <sascha.beetz@uni-rostock.de>

## See Also

reco, reco.bulk, gpp, gpp.bulk

## Examples

```
## see examples at reco.bulk
```

---

# tt.nee *Medium frequency concentration data and fluxes from non-steady state closed chamber measurements*

---

## Description

The data comes from the Trebeltal / Northeastern Germany and has been recorded with flexible transparent and opaque non-steady state closed chambers in 2011.

## Usage

```
data(tt.nee)
data(tt.flux)
```

## Format

tt.nee is a data.frame with 18 variables representing 14388 CO2 concentration measurements from 104 chamber placements.

tt.flux is a results table representing fluxes estimated with fluxx from tt.nee with 28 columns and 104 rows (= number of chamber placements in tt.nee). Contains many variables from tt.nee.

date Factor giving the date of field sampling, format is "%Y-%m-%d".

time Factor giving the time of measurement in the field, format is "%H:%M:%S".

session (Unique) Session number identifying one chamber placement. Integer.

record Integer, running number of concentration measurement within one session.

spot Factor identifying the field measurement location.

PAR Numeric. Photosynthetic photon flux density (PPFD).

t.cham Numeric. Temperature logged inside chamber during concentration measurements

NEE Numeric. CO2 concentration in chamber headspace.

t.air Numeric. Air temperature outside chamber.

t.soil2 Numeric. Soil temperature at 2cm depth.

t.soil5 Numeric. Soil temperature at 5cm depth.

`t.soil10` Numeric. Soil temperature at 10cm depth.

`area` Numeric. Chamber area.

`height` Numeric. Chamber height.

`kind` Integer. Chamber kind. 1 = transparent chamber, 3 = transparent chamber with measurement before sun rise, 5 = opaque chamber

`volume` Numeric. Chamber volume.

`datetime` POSIXlt. Time stamp.

`plot` Factor identifying the field plot (all TY1).

`all` Factor. Combined unique identifier for chamber placement.

`CO2.pv` Numeric. p.value of the fitted regression for the flux estimation.

`CO2.r2.f` Logical numeric (0 | 1) giving the r2 flag. See `fluxx` for details.

`CO2.range.f` Logical numeric (0 | 1) giving the range flag. See `fluxx` for details.

`CO2.nrmse.f` Logical numeric (0 | 1) giving the nrmse flag. See `fluxx` for details.

`CO2.ghg` Character. Greenhouse gas as submitted to `fluxx` via `var.par`.

`CO2.unit` Character. Ouput unit of the flux as specified via `out.unit` in `fluxx`.

`CO2.flux` Numeric. Flux

`CO2.r2` Numeric. R2 of the model that has been used for flux estimation.

`CO2.nrmse` Numeric. NRMSE of the model that has been used for flux estimation.

`CO2.nomba.f` Numeric. Number of concentration measurements below ambient.

`CO2.podpu` Numeric between 0 and 1. Propórtion of data points used.

## Details

`tt.nee` contains medium frequency (measured online) CO2 concentration data from 3 spots with *Typha angustifolia* and includes data needed for modelling `gpp`/nne measured with transparent chamber and `reco` measured with opaque chamber. `tt.flux` contains fluxes estimated from `tt.nee` using `fluxx`.

## Source

unpublished preliminary data

## Examples

```
## see examples at fluxx and gpp.
```

tt.pre *One day data from closed chamber measurements in the Trebeltal*

## Description

The data comes from the Trebeltal / Northeastern Germany and has been recorded with flexible non-steady state closed chambers in March 2011. It contains concentration data from 18 chamber measurements including calibration gas measurements that have been carried out alternatingly on the GC.

## Usage

```
data(tt.pre)
```

## Format

A data frame with 118 observations on the following 17 variables.

`year` numeric vector giving the year of measurement

`date` factor giving the date of field sampling, format is "%Y-%m-%d"

`time` factor giving the time of measurement in the field, format is "%H:%M"

`veg` factor with levels `c p t`

`spot` numeric vector, but it is a factor giving the number of the field measurement location. The combination of `veg` and `spot` uniquely identifies the measurement locations in the site

`time_min` numeric vector, time in minutes during the chamber placement. starts with 0 from placing the chamber

`sampletype_a` factor with levels `E P` determining whether its a field concentration measurement or a calibration gas measurement

`temp_dC` numeric vector, air temperature within chamber during measurements, taken at the same times as the concentration samples

`cham_vol` numeric vector, chamber volume per chamber placement. Varies from chamber placement to chamber placement depending on the chamber used

`cham_area` numeric vector giving the chamber area

`date_gc` factor giving the date of the gc measurement, format is "%Y-%m-%d"

`CO2Code` numeric vector, quality parameter from the GC

`CO2ppm` numeric vector, concentration of CH4 in air sample / calibration gas sample

`N2OCode` numeric vector, quality parameter from the GC

`N2Oppb` numeric vector, concentration of N2O in air sample / calibration gas sample

`CH4Code` numeric vector, quality parameter from the GC

`CH4ppb` numeric vector, concentration of CO2 in air sample / calibration gas sample

## Details

The 18 chamber measurements are carried out on three vegetation types (*Phragmites*, *Typha*, *Carex*).

## Source

unpublished preliminary data, whole data set in

Günther A, Huth V, Jurasinski G, Glatzel S (2013a) Scale-dependent temporal variation during the determination of the methane balance of a temperate fen. Greenhouse Gas Measurement & Management DOI: 10.1080/20430779.2013.850395

Huth V, Günther A, Jurasinski G, Glatzel S (2013) The impact of an extraordinarily wet summer on methane emissions from a 15-year re-wetted fen in north-east Germany. Mires & Peat 13.2:1–7

## Examples

```
## load data
data(tt.pre)
## see their structure
str(tt.pre)
```

# Index