# Package 'gasfluxes'

May 29, 2019

**Type** Package

**Title** Greenhouse Gas Flux Calculation from Chamber Measurements

**Version** 0.4-3

**Date** 2019-05-29

**Maintainer** Roland Fuss <roland.fuss@thuenen.de>

**BugReports** https://bitbucket.org/ecoRoland/gasfluxes/issues

**Description** Functions for greenhouse gas flux calculation from chamber measurements.

**License** GPL (>= 2)

**Depends** R (>= 3.5.0)

**Imports** sfsmisc (>= 1.0), data.table (>= 1.9.4), MASS (>= 7.3), AICcmodavg (>= 2.0), stats, graphics, grDevices

**Suggests** testthat, knitr, rmarkdown

**LazyData** true

**VignetteBuilder** knitr, rmarkdown

**RoxygenNote** 6.1.1

**NeedsCompilation** no

**Author** Roland Fuss [aut, cre],
Roman Hueppi [ctb],
Asger R. Pedersen [cph] (for code copied from the not-exported HMR:::.HMR.fit1 function (version 0.3.1))

**Repository** CRAN

**Date/Publication** 2019-05-29 10:40:02 UTC

## R topics documented:

gasfluxes-package . . . . . 2
agg.fluxes . . . . . 2
erfc . . . . . 3
fluxMeas . . . . . 4

gasfluxes . . . . . 4
HMR.fit . . . . . 7
HMR.orig . . . . . 9
lin.fit . . . . . 10
NDFE.fit . . . . . 12
rlin.fit . . . . . 13
selectfluxes . . . . . 15

**Index** **18**

---

`gasfluxes-package` *Calculate greenhouse gas flux calculation from chamber measurements.*

---

## Description

Gasfluxes provides functions for fitting non-linear concentration - time models as well as convenience functions for checking data and combining different calculation methods.

## Details

The wrapper function for convenient flux calculation is `gasfluxes`. Several concentration - time models are implemented

- `HMR.orig`: The original implementation of HMR.
- `HMR.fit`: A new implementation of HMR using partially linear least-squares. This is recommended over `HMR.orig`.
- `NDFE.fit`: An implementation of the NDFE model using partially linear least-squares.
- `lin.fit`: A simple linear model.
- `rlin.fit`: A simple linear model fit using robust regression.

---

`agg.fluxes` *Accumulation of fluxes*

---

## Description

Aggregate a time series of fluxes to a cummulative flux value.

## Usage

```
agg.fluxes(fluxes, datetimes, timeunit = "hours")
```

### Arguments

| | |
|---|---|
| fluxes | flux values |
| datetimes | datetime values (POSIXct or POSIXlt) |
| timeunit | the unit of time (denominator of the flux unit), supported are the explicit units supported by difftime |

### Details

The function uses linear interpolation. The unit of the cumulative flux is [fluxes] * timeunit. NA values are removed and values sorted according to time order. If less then two non-NA value pairs are provided, NA is returned for the cumlative flux.

### Value

A one-row data.frame with columns

| | |
|---|---|
| flux | the cumulative flux |
| from | the start of the cumulation period |
| to | the end of the cumulation period |

The return value being a data.frame is useful, when the function is used for "split-apply-combine" type operations to calculate groupwise cumulated values, e.g., using package data.table.

### Examples

```
#Some random example data
datetimes <- Sys.time() + (1:20)/2*24*3600
set.seed(42)
fluxes <- rlnorm(20, 5)
agg.fluxes(fluxes, datetimes)
```

---

## erfc *erfc*

---

### Description

This is the complementary error function.

### Usage

```
erfc(x)
```

### Arguments

| | |
|---|---|
| x | a numeric vector |

## Value

A numeric vector, i.e., the erfc values.

---

`fluxMeas` *Data from chamber N2O flux measurements.*

---

## Description

A dataset containing data from 1329 chamber N2O flux measurements.

## Format

A data.table with 5300 rows and 5 variables:

- serie: ID of flux measurement
- V: Volume (normalized by area, i.e., the height in m)
- A: Area (always 1)
- time: closing time in h
- C: N2O concentration in mg N / m^3

## Source

own data (anonymized by not including site and treatment information)

---

`gasfluxes` *Flux calculation*

---

## Description

A wrapper function for convenient flux calculation.

## Usage

```
gasfluxes(dat, .id = "ID", .V = "V", .A = "A", .times = "time",
  .C = "C", methods = c("linear", "robust linear", "HMR", "NDFE"),
  k_HMR = log(1.5), k_NDFE = log(0.01), verbose = TRUE,
  plot = TRUE, select, maxiter = 100, ...)
```

## Arguments

| | |
|---|---|
| dat | a data.frame or data.table with data from flux measurements. |
| .id | character vector specifying the columns to be used as ID, multiple ID columns are possible. |
| .V | character specifying the column containing chamber volume values. |
| .A | character specifying the column containing chamber area values. |
| .times | character specifying the column containing chamber closing time values. |
| .C | character specifying the column containing concentration values. |
| methods | character; which methods to use for flux estimation. See details for available methods. |
| k_HMR | starting value for HMR.fit. |
| k_NDFE | starting value for NDFE.fit. |
| verbose | logical; print progress messages? |
| plot | create plots if TRUE (the default). The IDs are used as file names and should thus not include characters that are not allowed in file names, such as / or =. A directory "pics" is created in the working directory if it doesn't exist. The plots are only intended to facilitate quick checking, not for publication quality graphs. |
| select | deprecated; please use function selectfluxes. |
| maxiter | see nls.control |
| ... | further parameters |

## Details

Available methods are

| | |
|---|---|
| "linear": | lin.fit |
| "robust linear": | rlin.fit |
| "HMR": | HMR.fit |
| "original HMR": | HMR.orig |
| "NDFE": | NDFE.fit |

Specifying other methods results in an error.

The default starting values for "HMR" and "NDFE", $k = log(\kappa)$ and $k = log(\tau)$, resp., assume that time is in hours. If you use a different time unit, you should adjust them accordingly. Note that nls is used internally by these functions and thus they should not be used with artificial "zero-residual" data.

The input data.frame or data.table should be in the following format:

```
   serie        V A      time         C
1:    ID1 0.522625 1 0.0000000 0.3317823
2:    ID1 0.522625 1 0.3333333 0.3304053
3:    ID1 0.522625 1 0.6666667 0.3394311
```

```
4:    ID1 0.522625 1 1.0000000 0.4469102
5:    ID2 0.523625 1 0.0000000 0.4572708
```

However, more than one ID column are possible. E.g., the first ID column could be the plot and a second ID column could be the date. Keep in mind that the combination of IDs must be a unique identifier for each flux measurement.

Units of the output depend on input units. It's recommended to use [V] = m^3, [A] = m^2, [time] = h, [C] = [mass or mol]/m^3, which results in [f0] = [mass or mol]/m^2/h. Since all algorithms use V/A, A can be input as 1 and V as the chamber height.

## Value

A data.table with the results of the flux calculation. See the documentation of the fitting functions for details. If a selection algorithm has been specified, the last columns are the selected flux estimate, the corresponding standard error and p-value and the method with which the selected flux was estimated.

## See Also

`selectfluxes` for flux selection

## Examples

```
## Not run:
#compare result of original HMR with plinear HMR
data(fluxMeas)
res <- gasfluxes(fluxMeas[1:400,],
                 .id = "serie", .V = "V", .A = "A",
                 .times = "time", .C = "C",
                 methods = c("HMR", "original HMR"), verbose = TRUE)

#number of successful fits
res[, sum(!is.na(HMR.kappa))]
res[, sum(!is.na(original.HMR.kappa))]

#Do the results differ?
plot(res[["HMR.f0"]], res[["original.HMR.f0"]])
abline(0, 1)


res <- gasfluxes(fluxMeas,
                 .id = "serie", .V = "V", .A = "A",
                 .times = "time", .C = "C",
                 methods = "HMR", verbose = TRUE)
# Error: time not sorted in flux ID ID556.
# Investigate the problem:
fluxMeas[serie %in% c("ID555", "ID556", "ID557")]
#    serie        V A      time         C
# 1: ID555 0.551625 1 0.0000000 0.3884388
# 2: ID555 0.551625 1 0.3333333 0.4125270
# 3: ID555 0.551625 1 0.6666667 0.3714207
```

```
# 4: ID555 0.551625 1 1.0000000 0.3735092
# 5: ID556 0.524250 1 0.0000000 0.3638239
# 6: ID556 0.524250 1 0.3333333 0.3520481
# 7: ID556 0.524250 1 0.6666667 0.3551644
# 8: ID557 0.528375 1 0.0500000 0.3954601
# 9: ID556 0.524250 1 0.0000000 0.3839834
#10: ID557 0.528375 1 0.3333333 0.3967269
#11: ID557 0.528375 1 0.6666667 0.3764967
#12: ID557 0.528375 1 1.0000000 0.3973055

# some mixup of IDs and times
# usually an input or Excel error during data preparation
# investigate and fix

## End(Not run)
```

---

`HMR.fit` *HMR fit*

---

## Description

Fit the HMR model using the Golub-Pereyra algorithm for partially linear least-squares models.

## Usage

```
HMR.fit(t, C, A = 1, V, serie = "", k = log(1.5), verbose = TRUE,
  plot = FALSE, maxiter = 100, ...)
```

## Arguments

| | |
|---|---|
| `t` | time values (usually in hours) |
| `C` | concentration values |
| `A` | area covered by the chamber |
| `V` | effective volume of the chamber |
| `serie` | id of the flux measurement |
| `k` | starting value for nls function |
| `verbose` | logical, TRUE prints message after each flux calculation |
| `plot` | logical, mainly intended for use in `gasfluxes` |
| `maxiter` | see `nls.control` |
| `...` | further parameters, currently none |

## Details

The HMR model (Pedersen et al., 2010) is $C(t) = \phi + f_0 \frac{e^{-\kappa t}}{-\kappa V/A}$. To ensure the lower bound $\kappa > 0$, the substitution $\kappa = e^k$ is used. The resulting reparameterized model is then fit using `nls` with `algorithm = "plinear"`. This is computationally more efficient than the manual implementation in the HMR package and results in almost identical flux values. Flux standard errors and p-values differ strongly from those reported by the HMR package <= version 0.3.1, but are equal to those reported by later versions.

The default starting value $k = log(\kappa)$ assumes that time is in hours. If you use a different time unit, you should adjust it accordingly.

There have been demands to return the initial concentration as predicted by the model as this is useful for checking plausibility. However, this can be easily calculated from the parameters and the equation of the model by setting t = 0, i.e., $C_0 = \phi + f_0$.

Note that `nls` is used internally and thus this function should not be used with artificial "zero-residual" data.

## Value

A list of

| | |
|---|---|
| `f0` | flux estimate |
| `f0.se` | standard error of flux estimate |
| `f0.p` | p-value of flux estimate |
| `kappa, phi` | other parameters of the HMR model |
| `AIC` | Akaike information criterion |
| `AICc` | Akaike information criterion with small sample correction |
| `RSE` | residual standard error (sigma from summary.nls) |
| `diagnostics` | error or warning messages |

## References

Pedersen, A.R., Petersen, S.O., Schelde, K., 2010. A comprehensive approach to soil-atmosphere trace-gas flux estimation with static chambers. European Journal of Soil Science 61(6), 888-902.

## Examples

```
#a single fit
t <- c(0, 1/3, 2/3, 1)
C <- c(320, 341, 352, 359)
print(fit <- HMR.fit(t, C, 1, 0.3, "a"))
plot(C ~ t)
curve({fit$phi + fit$f0 * exp(-fit$kappa * x)/(-fit$kappa*0.3)},
      from = 0, to = 1, add = TRUE)
#compare with fitting function from HMR package 0.3.1
  gasfluxes:::.HMR.fit1(t, C,
                 1, 0.3, "a",
                 ngrid = 1000, LR.always = FALSE, FollowHMR = TRUE,
                 JPG = FALSE, PS = FALSE, PHMR = FALSE, npred = 500,
```

```
                  xtxt = "", ytxt = "", pcttxt = "",
                  MSE.zero = 10 * max(.Machine$double.eps, .Machine$double.neg.eps),
                  bracketing.tol = 1e-07, bracketing.maxiter = 1000)[2:4]

## Not run:
#a dataset of 1329 chamber N2O flux measurements
data(fluxMeas)
fluxMeas[, n := length(time), by=serie]
print(fluxMeas)
fluxes <- fluxMeas[n > 3, HMR.fit(time, C, A, V, serie), by=serie]
print(fluxes)
plot(f0.se ~ f0, data = fluxes)
#one very large f0.se value (and several infinite ones not shown in the plot)
fluxes[is.finite(f0.se),][which.max(f0.se),]
plot(C~time, data=fluxMeas[serie=="ID940",])
print(tmp <- fluxes[is.finite(f0.se),][which.max(f0.se),])
curve({tmp[, phi] + tmp[, f0] * exp(-tmp[, kappa] * x)/
      (-tmp[, kappa]*fluxMeas[serie=="ID940", V[1]]/
      fluxMeas[serie=="ID940",A[1]])},
      from = 0, to = 1, add = TRUE)
plot(f0.se ~ f0, data = fluxes[f0.se < 1e4,], pch = 16)
boxplot(fluxes[f0.se < 1e4, sqrt(f0.se)])

## End(Not run)
```

---

HMR.orig *HMR fit using the Pedersen fitting algorithm*

---

## Description

Fit the HMR model using the algorithm from the HMR package.

## Usage

```
HMR.orig(t, C, A = 1, V, serie = "", verbose = TRUE, ngrid = 1000,
  plot = FALSE, ...)
```

## Arguments

| | |
|---|---|
| t | time values (usually in hours) |
| C | concentration values |
| A | area covered by the chamber |
| V | effective volume of the chamber |
| serie | id of the flux measurement |
| verbose | logical, TRUE prints message after each flux calculation |
| ngrid | see the HMR documentation |
| plot | logical, mainly intended for use in gasfluxes |
| ... | further parameters, currently none |

## Details

The HMR model (Pedersen et al., 2010) is $C(t) = \phi + f_0 \frac{e^{-\kappa t}}{-\kappa V/A}$. The algorithm from the HMR package version 0.3.1 is used for fitting. Note that this is very inefficient and standard errors and p-values are over-estimated. `HMR.fit` is recommended instead and this function is only provided to be able to reproduce results obtained with older versions of the HMR package.

## Value

A list of

| | |
|---|---|
| `f0` | flux estimate |
| `f0.se` | standard error of flux estimate |
| `f0.p` | p-value of flux estimate |
| `kappa, phi` | other parameters of the HMR model |
| `AIC` | Akaike information criterion |
| `AICc` | Akaike information criterion with small sample correction |
| `diagnostics` | error or warning messages |

## Author(s)

Asger R. Pedersen for code copied from the not-exported HMR:::.HMR.fit1 function, Roland Fuss

## References

Pedersen, A.R., Petersen, S.O., Schelde, K., 2010. A comprehensive approach to soil-atmosphere trace-gas flux estimation with static chambers. European Journal of Soil Science 61(6), 888-902.

## Examples

```
#a single fit
t <- c(0, 1/3, 2/3, 1)
C <- c(320, 341, 352, 359)
print(fit <- HMR.orig(t, C, 1, 0.3, "a"))
plot(C ~ t)
curve({fit$phi + fit$f0 * exp(-fit$kappa * x)/(-fit$kappa*0.3)},
      from = 0, to = 1, add = TRUE)
```

---

# lin.fit *Linear concentration - time model*

---

## Description

Fit a linear model to concentration - time data.

## Usage

```
lin.fit(t, C, A = 1, V, serie = "", verbose = TRUE, plot = FALSE,
  ...)
```

## Arguments

| | |
|---|---|
| t | time values (usually in hours) |
| C | concentration values |
| A | area covered by the chamber |
| V | effective volume of the chamber |
| serie | id of the flux measurement |
| verbose | logical, TRUE prints message after each flux calculation |
| plot | logical, mainly intended for use in gasfluxes |
| ... | further parameters, currently none |

## Details

This is basically a wrapper of R's OLS fitting facilities. For now lm (and methods for objects of class "lm") is used, but this may change to more efficient alternatives in later versions.

## Value

A list of

| | |
|---|---|
| f0 | flux estimate |
| f0.se | standard error of flux estimate |
| f0.p | p-value of flux estimate |
| C0 | estimated concentration at t = 0 (intercept) |
| AIC | Akaike information criterion |
| AICc | Akaike information criterion with small sample correction |
| RSE | residual standard error (sigma from summary.nls) |
| r | Pearson's correlation coefficient |
| diagnostics | error or warning messages |

## Examples

```
#a single fit
t <- c(0, 1/3, 2/3, 1)
C <- c(320, 341, 352, 359)
print(fit <- lin.fit(t, C, 1, 0.3, "a"))
plot(C ~ t)
curve({fit$f0/0.3 * x + fit$C0}, from = 0, to = 1, add = TRUE)
```

NDFE.fit *NDFE fit*

## Description

Fit the the non-steady-state diffusive flux extimator model using the Golub-Pereyra algorithm for partially linear least-squares models.

## Usage

```
NDFE.fit(t, C, A = 1, V, serie = "", k = log(0.01), verbose = TRUE,
  plot = FALSE, maxiter = 100, ...)
```

## Arguments

| | |
|---|---|
| t | time values (usually in hours) |
| C | concentration values |
| A | area covered by the chamber |
| V | effective volume of the chamber |
| serie | id of the flux measurement |
| k | starting value for nls function |
| verbose | logical, TRUE prints message after each flux calculation |
| plot | logical, mainly intended for use in gasfluxes |
| maxiter | see nls.control |
| ... | further parameters, currently none |

## Details

The NDFE model (Livingston et al., 2006) is $C(t) = C_0 + f_0 \tau \frac{A}{V} \left[ \frac{2}{\sqrt{\mathrm{pi}}} \sqrt{t/\tau} + e^{t/\tau} \mathrm{erfc}(\sqrt{t/\tau}) - 1 \right]$.

To ensure the lower bound $\tau > 0$, the substituion $\tau = e^k$ is used. The resulting reparameterized model is then fit using nls with algorithm = "plinear".

Note that according to the reference the model is not valid for negative fluxes. Warning: This function does not check if fluxes are positive. It's left to the user to handle negative fluxes.

The default starting value $k = log(\tau)$ assumes that time is in hours. If you use a different time unit, you should adjust it accordingly.

Note that nls is used internally and thus this function should not be used with artificial "zero-residual" data.

## Value

A list of

| | |
|---|---|
| f0 | flux estimate |
| f0.se | standard error of flux estimate |
| f0.p | p-value of flux estimate |
| C0, tau | other parameters of the NDFE model |
| AIC | Akaike information criterion |
| AICc | Akaike information criterion with small sample correction |
| RSE | residual standard error (sigma from summary.nls) |
| diagnostics | error or warning messages |

## References

Livingston, G.P., Hutchinson, G.L., Spartalian, K., 2006. Trace gas emission in chambers: A non-steady-state diffusion model. Soil Sci. Soc. Am. J. 70(5), 1459-1469.

## Examples

```
#a single fit
t <- c(0, 1/3, 2/3, 1)
C <- c(320, 340, 355, 362)
print(fit <- NDFE.fit(t, C, 1, 0.3, "a"))
plot(C ~ t)
curve({fit$C0+fit$f0*fit$tau*1/0.3*(2/sqrt(pi)*sqrt(x/fit$tau)+
      exp(x/fit$tau)*erfc(sqrt(x/fit$tau))-1)},
      from = 0, to = 1, add = TRUE)
#note that the flux estimate is very uncertain because
#there are no data points in the region of high curvature
```

---

# rlin.fit *Robust linear concentration - time model*

---

## Description

Fit a linear model to concentration - time data using robust methods.

## Usage

```
rlin.fit(t, C, A = 1, V, serie = "", verbose = TRUE, plot = FALSE,
  ...)
```

## Arguments

| | |
|---|---|
| t | time values (usually in hours) |
| C | concentration values |
| A | area covered by the chamber |
| V | effective volume of the chamber |
| serie | id of the flux measurement |
| verbose | logical, TRUE prints message after each flux calculation |
| plot | logical, mainly intended for use in gasfluxes |
| ... | further parameters, currently none |

## Details

This is basically a wrapper of rlm using the Huber M estimator. This function never weights the first or last time point with zero with very few data points. However, there might exist "better" robust regression methods for flux estimation.

## Value

A list of

| | |
|---|---|
| f0 | flux estimate |
| f0.se | standard error of flux estimate |
| f0.p | p-value of flux estimate |
| C0 | estimated concentration at t = 0 (intercept) |
| weights | robustness weights |
| diagnostics | error or warning messages |

## Examples

```
#a single fit
t <- c(0, 1/3, 2/3, 1)
C <- c(320, 330, 315, 351)
print(fit <- rlin.fit(t, C, 1, 0.3, "a"))
plot(C ~ t)
curve({fit$f0/0.3 * x + fit$C0}, from = 0, to = 1, add = TRUE)
```

`selectfluxes` *Select a flux estimate*

## Description

Selects the appropriate flux estimate from linear, robust linear and non-linear calculated fluxes.

## Usage

```
selectfluxes(dat, select, f.detect = NULL, t.meas = NULL,
  tol = 5e-05, ...)
```

## Arguments

| | |
|---|---|
| `dat` | a data.table as returned by `gasfluxes`. The function modifies it by reference. |
| `select` | character; specify a ruleset for selection of the final flux value, see details. |
| `f.detect` | detection limit for HMR method. This can be determined by a simple simulation (see examples) or for four data points the approximation in Parkin et al. (2012) can be used. |
| `t.meas` | a vector or single value giving the measurement time factor that relates to kappa.max. It is suggested to use the time difference between the first and last sample taken from the closed chamber. The unit should be consistent with the units of `f.detect` and kappa (e.g., h if kappa is in 1/h). |
| `tol` | the relative tolerance `abs((linear.f0 - HMR.f0)/HMR.f0)` below which the linear flux estimate and the HMR flux estimate are considered equal in the "kappa.max" algorithm. This is to protect against HMR fits that equal the linear fit and have extremely high standard errors. Defaults to `tol = 5e-5`. |
| `...` | further parameters |

## Details

Available selection algorithms currently are

**"RF2011"** The algorithm used, e.g., in Leiber-Sauheitl 2014 (doi:10.5194/bg-11-749-2014). This overwrites the methods parameter. The factor guarding against degenerate HMR fits can be set via the ellipsis as `gfactor`. Default is `gfactor = 4`. This method is not recommended any more and only provided for reproducibility of old results.

**"RF2011new"** The same rules as "RF2011", but using the improved fitting function for HMR, which results in larger SE and p-values. Thus, it is less likely to select the HMR result. This method is not recommended any more and only provided for reproducibility of old results.

**"kappa.max"** The selection algorithm restricts the use of HMR by imposing a maximal value for kappa "kappa.max", depending on the quotient of the linear flux estimate and the minimal detectable flux (f.detect), as well as the chamber closure time (t.meas). kappa.max = f.lin/f.detect/t.meas. This is currently the recommended algorithm.

Other selection algorithms could be implemented, but selection can always be done as a postprocessing step. E.g., if many data points are available for each flux measurement it is probably most sensible to use AICc.

## Value

A data.table with the with following columns added to the function input: selected flux estimate, the corresponding standard error and p-value and the method with which the selected flux was estimated. For the "kappa.max" method the "kappa.max" values are included. These columns are also added to the input data.table by reference.

## References

Parkin, T.B., Venterea, R.T., Hargreaves, S.K., 2012. Calculating the Detection Limits of Chamber-based Soil Greenhouse Gas Flux Measurements. Journal of Environmental Quality 41, 705-715.

Hueppi, R., Felber, R., Krauss, M., Six, J., Leifeld, J., Fuss, R., 2018. Restricting the nonlinearity parameter in soil greenhouse gas flux calculation for more reliable flux estimates. PLOS ONE 13(7): e0200876. https://doi.org/10.1371/journal.pone.0200876

## Examples

```
## Not run:
res <- gasfluxes(fluxMeas[1:499],
                 .id = "serie", .V = "V", .A = "A",
                 .times = "time", .C = "C",
              methods = c("linear", "robust linear", "HMR"), verbose = FALSE, plot = FALSE)
selectfluxes(res, "RF2011new")
res[method == "HMR", .N] #2

### estimate f.detect by simulation ###
#ambient concentration:
C0 <- 320/1000 * 28 * 273.15 / 22.4 / (273.15 + 15) #mg N / m^3
#uncertainty of GC measurement:
sdGC <- 5/1000 * 28 * 273.15 / 22.4 / (273.15 + 15) #mg N / m^3
#create simulated concentrations corresponding to 1 hour flux measurements with zero fluxes:
set.seed(42)
sim <- data.frame(t = seq(0, 1, length.out = 4), C = rnorm(4e3, mean = C0, sd = sdGC),
                  id = rep(1:1e3, each = 4), A = 1, V = 0.52)
#fit HMR model:
simflux <- gasfluxes(sim, .id = "id", .times = "t", methods = c("HMR", "linear"), plot = FALSE)
simflux[, f0 := HMR.f0]
simflux[is.na(f0), f0 := linear.f0]
#dection limit as 97.5 % quantile (95 % confidence):
f.detect <- simflux[, quantile(f0, 0.975)] #0.03 mg N / m^2 / h

# example using the kappa.max (ref. Hueppi et al., 2018) with a single t.meas value
t.meas <- max(fluxMeas$time[1:499]) #1
selectfluxes(res, "kappa.max", f.detect = f.detect, t.meas = t.meas)
res[method == "HMR", .N] # 11

# example using the kappa.max with a vector for t.meas
t.meas <- fluxMeas[1:499][, max(time), by = serie][["V1"]]
selectfluxes(res, "kappa.max", f.detect = f.detect, t.meas = t.meas)
res[method == "HMR", .N] # 10

## End(Not run)
```

# Index

agg.fluxes, 2

erfc, 3

fluxMeas, 4

gasfluxes, *2*, 4, *7*, *9*, *11*, *12*, *14*, *15*
gasfluxes-package, 2

HMR.fit, *2*, *5*, 7, *10*
HMR.orig, *2*, *5*, 9

lin.fit, *2*, *5*, 10
lm, *11*

NDFE.fit, *2*, *5*, 12
nls, *8*, *12*
nls.control, *5*, *7*, *12*

rlin.fit, *2*, *5*, 13
rlm, *14*

selectfluxes, *5*, *6*, 15