Package: EBASE (via r-universe)

August 25, 2024

Title Estuarine Bayesian Single-Station Estimation Method for Ecosystem Metabolism

Version 1.0.2.9000

Date 2024-06-05

Description Estimate ecosystem metabolism in a Bayesian framework for individual water quality monitoring stations with continuous dissolved oxygen time series. A mass balance equation is used that provides estimates of parameters for gross primary production, respiration, and gas exchange. Methods adapted from Grace et al. (2015) <doi:10.1002/lom3.10011> and Wanninkhof (2014) <doi:10.4319/lom.2014.12.351>. Details in Beck et al. (2024) <doi:10.1002/lom3.10620>.

Depends R (>= 3.5)

Imports doParallel, dplyr, foreach, ggplot2 (>= 3.4.0), lubridate, R2jags (>= 0.6.1), rjags (>= 4.10), tidyr, truncnorm, zoo

Suggests testthat (>= 3.0.0), knitr, rmarkdown, covr

License CC0

SystemRequirements JAGS 4.x.y (https://mcmc-jags.sourceforge.net)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL https://fawda123.github.io/EBASE/, https://github.com/fawda123/EBASE/

BugReports https://github.com/fawda123/EBASE/issues

LazyData true

LazyDataCompression xz

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://fawda123.r-universe.dev

RemoteUrl https://github.com/fawda123/EBASE

RemoteRef HEAD

RemoteSha 92e7e688c6529fcafab7a285da5e777074ae2fb7

Contents

`credible_plot` *Plot credible intervals for a, R, and b*

Description

Plot credible intervals for a, R, and b

Usage

```
credible_plot(res, params = c("a", "R", "b"))
```

Arguments

`res`	output data frame from `ebase`
`params`	character vector indicating which parameters to plot, one to any of `a`, `R`, or `b`, (default all)

Details

This function plots 95% credible intervals (2.5th to 97.5th percentiles, approximate posterior distributions) for `a`, `R` and/or `b` using the output from `ebase`. Results in the plot are grouped by the `ndays` argument that was used in `ebase`.

Value

A ggplot object

Examples

```
# plot credible intervals
credible_plot(exres)
```

---

credible_prep *Get credible intervals for a, R, b*

Description

Get credible intervals for a, R, b

Usage

```
credible_prep(res, params = c("a", "R", "b"), labels = FALSE)
```

Arguments

res	output data frame from ebase
params	character vector indicating which parameters to plot, one to any of a, R, or b (default all)
labels	logical indicating of parameter labels are output as an expression for parsing in plot facets, default FALSE

Details

This function gets 95% credible intervals (2.5th to 97.5th percentiles, approximate posterior distributions) for a, R, and/or b using the output from ebase. The function is used in credible_plot, but is provided as a separate function for convenience.

Value

A data frame

Examples

```
# get credible intervals
credible_prep(exres)
```

ebase *Estuarine Bayesian Single-station Estimation method for ecosystem metabolism*

Description

Estuarine Bayesian Single-station Estimation method for ecosystem metabolism

Usage

```
ebase(
  dat,
  Z,
  interval,
  ndays = 1,
  aprior = c(4, 2),
  rprior = c(300, 150),
  bprior = c(0.251, 0.125),
  bmax = 0.502,
  nogas = FALSE,
  doave = TRUE,
  maxinterp = 43200/interval,
  n.iter = 10000,
  update.chains = TRUE,
  n.burnin = n.iter * 0.5,
  n.chains = 3,
  n.thin = 10,
  model_file = NULL
)
```

Arguments

dat	input data frame
Z	numeric as single value for water column depth (m) or vector equal in length to number of rows in dat
interval	timestep interval in seconds
ndays	numeric for number of days in dat for optimizing the metabolic equation, see details
aprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *a* parameter, see details
rprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *R* parameter, see details
bprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *b* parameter, see details

bmax	numeric value for the upper limit on the prior distribution for bprior, set as twice the default value of the mean
nogas	logical indicating if gas exchange is not included in the metabolic model, see details
doave	logical indicating if the average dissolved oxygen concentration is used as the starting value for the estimation (default), otherwise the first observation will be used if FALSE, see details
maxinterp	numeric value for minimum number of continuous observations that must not be interpolated within a group defined by ndays to assign as NA in output, see details
n.iter	number of MCMC iterations, passed to jags
update.chains	logical to run metab_update if chains do not converge
n.burnin	number of MCMC chains to delete, passed to jags
n.chains	number of MCMC chains to run, passed to jags
n.thin	number of nth iterations to save for each chain, passed to jags
model_file	NULL to use the model file included with the package or a path to a model text file can be used

Details

Required input data are time series for dissolved oxygen (mg L-1), water temperature (C), salinity (psu), total PAR (W m-2), and wind speed (m s-1). See the exdat example data file for a representation of the required data. Data are typically from continuously monitored water quality and weather parameters are hourly of sub-hourly time steps. Oxygen concentrations are converted to mmol/m3 prior to metabolic estimation. Water column depth is also required. This can be supplied as a single value or a vector of length equal to the number of rows in dat.

The metabolic estimates are based on a mass balance equation in Grace et al. 2015 with the gas exchange estimate from Wanninkhof 2004. It is similar to that provided by the BASEmetab R package at https://github.com/dgiling/BASEmetab, with modifications to estimate different parameters. The equation optimized in the JAGS model is:

$$Z\frac{dC_d}{dt} = aPAR - R + bU_{10}^2\left(\frac{Sc}{600}\right)^{-0.5}(C_{Sat} - C_d)$$

More simply:

$$Z\frac{dC_d}{dt} = P - R + D$$

Gross production is provided by *aPAR*, respiration is provided by *R*, and gas exchange is provided by the remainder. The likelihood of the parameters *a*, *R*, and *b* given the observed data are estimated from the JAGS model using prior distributions shown in the model file. At each time step, the change in oxygen concentration between time steps is calculated from the equation using model inputs and parameter guesses, and then a finite difference approximation is used to estimate modeled oxygen concentration. The first modeled value starts at the mean oxygen concentration for all

measurements in the optimization period. The estimated concentration at each time step is also returned for comparison to observed as one measure of model performance.

The prior distributions for the *a*, *R*, and *b* parameters are defined in the model file included with the package as normal distributions with mean and standard deviations provided by the `aprior`, `rprior`, and `bprior` arguments. The default values were chosen based on approximate values from national syntheses of metabolic estimates. An additional constraint is that all the prior distributions are truncated to be positive values as required by the core metabolism equation above. The upper limit for *b* is set as two times 0.251, as given in eqn. 4 in Wanninkhof 2014. Units for each parameter are (mmol m-2 d-1)/(W m-2) for *a*, mmol m-2 d-1 for *R*, and (cm hr-1)/(m2 s-2) for *b*.

The `ndays` argument defines the model optimization period as the number of days that are used for optimizing the above mass balance equation. By default, this is done each day, i.e., `ndays= 1` such that a loop is used that applies the model equation to observations within each day, evaluated iteratively from the first observation in a day to the last. Individual parameter estimates for *a*, *R*, and *b* are then returned for each day. However, more days can be used to estimate the unknown parameters, such that the loop can be evaluated for every `ndays` specified by the argument. The `ndays` argument will separate the input data into groups of consecutive days, where each group has a total number of days equal to `ndays`. The final block may not include the complete number of days specified by `ndays` if the number of unique dates in the input data includes a remainder when divided by `ndays`, e.g., if seven days are in the input data and `ndays = 5`, there will be two groups where the first has five days and the second has two days. The output data from `ebase` includes a column that specifies the grouping that was used based on `ndays`.

Missing values in the input data are also interpolated prior to estimating metabolism. It is the responsibility of the user to verify that these interpolated values are not wildly inaccurate. Missing values are linearly interpolated between non-missing values at the time step specified by the value in `interval`. This works well for small gaps, but can easily create inaccurate values at gaps larger than a few hours. The `interp_plot` function can be used to visually assess the interpolated values. Records at the start or end of the input time series that do not include a full day are also removed. A warning is returned to the console if gaps are found or dangling records are found.

The `maxinterp` argument specifies a minimum number of observations that must not be interpolated within groups defined by `ndays` that are assigned `NA` in the output (except `Date` and `DateTimeStamp`). Groups with continuous rows of interpolated values with length longer than this argument are assigned `NA`. The default value is half a day, i.e., 43200 seconds divided by the value in `interval`.

The `doave` argument can be used to define which dissolved oxygen value is used as the starting point in the Bayesian estimation for the optimization period. The default setting (`doave = TRUE`) will use the average of all the dissolved oxygen values in the optimization period defined by `ndays`. For example, the average of all dissolved oxygen values in each 24 hour period will be used if `doave = TRUE` and `ndays = 1`. The first dissolved oxygen observation of the time series in the optimization period will be used as the starting point if `doave = F`. In this case, the simulated dissolved oxygen time series will always start at the first observed dissolved oxygen value for each optimization period.

Value

A data frame with metabolic estimates for areal gross production (`P`, O2 mmol m-2 d-1), respiration (`R`, O2 mmol m-2 d-1), and gas exchange (`D`, O2 mmol m-2 d-1, positive values as ingassing, negative values as outgassing). Additional parameters estimated by the model that are returned include `a` and `b`. The `a` parameter is a constant that represents the primary production per quantum

of light with units of (mmol m-2 d-1)/(W m-2) and is used to estimate gross production (Grace et al., 2015). The b parameter is a constant used to estimate gas exchange in (cm hr-1)/(m2 s-2) (provided as 0.251 in eqn. 4 in Wanninkhof 2014). Observed dissolved oxygen (DO_obs, mmol m-3), modeled dissolved oxygen (DO_mod, mmol m-3), and delta dissolved oxygen of the modeled results (dDO, mmol m-3 d-1) are also returned. Note that delta dissolved oxygen is a daily rate.

95% credible intervals for a, b, and R are also returned in the corresponding columns alo, ahi, blo, bhi, Rlo, and Rhi, for the 2.5th and 97.5th percentile estimates for each parameter, respectively. These values indicate the interval within which there is a 95% probability that the true parameter is in this range. Note that all values for these parameters are repeated across rows, although only one estimate for each is returned based on the number of days defined by ndays.

Model fit can also be assessed using the converge and rsq columns. The values in these columns apply to each group in the grp column as specified with the ndays argument. The converge column indicates "Check convergence" or "Fine" if the JAGS estimate converged at that iteration (repeated across rows for the group). The n.chains argument can be increased if convergence is not achieved. Similarly, the rsq column shows the r-squared values of the linear fit between the modeled and observed dissolved oxygen (repeated across rows for the group). These values can also be viewed with fit_plot.

The nogas argument can be set to TRUE to exclude gas exchange from the metabolic estimates. This will force the prior distribution for b as mean 0 and standard deviation approximately 0.

References

Grace, M.R., Giling, D.P., Hladyz, S., Caron, V., Thompson, R.M., Nally, R.M., 2015. Fast processing of diel oxygen curves: Estimating stream metabolism with BASE (BAyesian Single-station Estimation). Limnology and Oceanography: Methods 13, e10011. https://doi.org/10.1002/lom3.10011

Wanninkhof, R., 2014. Relationship between wind speed and gas exchange over the ocean revisited. Limnology and Oceanography: Methods 12, 351–362. https://doi.org/10.4319/lom.2014.12.351

Examples

```
# get one day of data
dat <- exdat[as.Date(exdat$DateTimeStamp, tz = 'America/Jamaica') == as.Date('2012-06-01'), ]

# run ebase, use more chains and iterations for a better fit, update.chains as T
ebase(dat, interval = 900, Z = 1.85, n.chains = 2, n.iter = 50,
 update.chains = FALSE)
```

---

ebase_eqboxy *Oxygen saturation*

Description

Oxygen saturation

Usage

```
ebase_eqboxy(temp, salt)
```

Arguments

temp	numeric for temperature (C)
salt	numeric for salinity (PSU)

Details

Function to calculate equilibrium OXYGEN concentration in seawater, from water temparure (C) and salinity (PSU)

Value

oxysat (mmol/m^3)

References

Garcia, H., Gordon, L.I., 1992. Oxygen solubility in seawater: Better fitting equations. Limnology and Oceanography 37, 1307-1312. https://doi.org/10.4319/lo.1992.37.6.1307

Examples

```
temp <- c(10, 20, 30)
salt <- c(30, 35, 40)
ebase_eqboxy(temp = temp, salt = salt)
```

---

ebase_form *Format ebase output*

Description

Format ebase output

Usage

```
ebase_form(out, dat, interval, maxinterp = 43200/interval)
```

Arguments

out	data.frame for model output
dat	data.frame as returned by ebase_prep
interval	timestep interval in seconds
maxinterp	numeric value for minimum number of continuous observations that must not be interpolated within a group defined by ndays to assign as NA in output

Details

This function is used internally with ebase and should not be called by itself.

Value

Formatted output for ebase with interpolated rows as NA (except Date and DateTimeStamp as defined by maxinterp

Examples

```
library(dplyr)

# get four days of data
dat <- exdat %>%
  filter(lubridate::month(DateTimeStamp) == 6) %>%
  filter(lubridate::day(DateTimeStamp) %in% 1:4)
dat <- ebase_prep(dat, Z = 1.85, interval = 900, ndays = 1)

ebase_form(exres, dat, interval = 900, maxinterp = 48)
```

---

ebase_plot *Plot results from EBASE*

Description

Plot results from EBASE

Usage

```
ebase_plot(res, instantaneous = TRUE)
```

Arguments

res	output data frame from ebase
instantaneous	logical indicating if results are instantaneous (default) or averaged to daily

Details

All metabolic estimates are plotted as positive values (D is represented as net ingassing).

Value

A ggplot object

Examples

```
# plot instantaneous
ebase_plot(exres)

# plot daily-averaged
ebase_plot(exres, instantaneous = FALSE)
```

ebase_prep *Prepare data for ebase*

Description

Prepare data for ebase

Usage

```
ebase_prep(dat, Z, interval, ndays = 1)
```

Arguments

dat	input data frame
Z	numeric as single value for water column depth (m) or vector equal in length to number of rows in dat
interval	timestep interval in seconds
ndays	numeric for number of days in dat for optimizing the metabolic equation, see details

Details

Checks if all columns are present by matching those in exdat, checks if DateTimeStamp is in ascending order, converts dissolved oxygen from mg/L to mmol/m3, calculates the Schmidt number (unitless) from water temp (C) and salinity (psu), and calculates dissolved oxygen equilibrium concentration (mmol/m3) from salinity and temperature

The ndays argument defines the number of days that are used for optimizing the above mass balance equation. By default, this is done each day, i.e., ndays= 1 such that a loop is used that applies the model equation to observations within each day, evaluated iteratively from the first observation in a day to the last. Individual parameter estimates for a, R, and b are then returned for each day. However, more days can be used to estimate the unknown parameters, such that the loop can be evaluated for every ndays specified by the argument. The ndays argument will separate the input data into groups of consecutive days, where each group has a total number of days equal to ndays. The final block may not include the complete number of days specified by ndays if the number of unique dates in the input data includes a remainder when divided by ndays, e.g., if seven days are in the input data and ndays = 5, there will be two groups where the first has five days and the second has two days. The output data from ebase includes a column that specifies the grouping that was used based on ndays.

Missing values are interpolated at the interval specified by the interval argument for conformance with the core model equation. Records at the start or end of the input time series that do not include a full day are also removed. A warning is returned to the console if gaps are found or dangling records are found.

Value

A data frame with additional columns required for ebase. Dissolved oxygen as a volumetric concentration in dat as mg/L is returned in areal units as mmol/m2. If multiple time steps are identified, the number of rows in data frame is expanded based on the time step define by interval. Numeric values in the expanded rows will be interpolated if interp = TRUE, otherwise they will remain as NA values.

Examples

```
dat <- ebase_prep(exdat, Z = 1.85, interval = 900)
head(dat)
```

---

ebase_rho *Seawater density calculation*

Description

Seawater density calculation

Usage

```
ebase_rho(temp, salt, P)
```

Arguments

temp	numeric for temperature (C)
salt	numeric for salinity (PSU)
P	numeric for pressure above atmospheric (dbar)

Details

Density of seawater is calculated according to the internationally accepted (UNESCO) equations. The standard error of the equation is 3.6 x 10-3 kg/m-3.

Value

Rho (kg/m^3)

References

Millero, F.J., Poisson, A., 1981. International one-atmosphere equation of state of seawater. Deep Sea Research 28, 625-629. https://doi.org/10.1016/0198-0149(81)90122-9

Examples

```
temp <- c(10, 20, 30)
salt <- c(30, 35, 40)
ebase_rho(temp = temp, salt = salt, P = 0)
```

ebase_schmidt *Schmidt number calculation*

Description

Schmidt number calculation

Usage

```
ebase_schmidt(temp, salt)
```

Arguments

temp	numeric for temperature (C)
salt	numeric for salinity (PSU)

Details

The Schmidt number is calculated for the air-sea gas transfer velocity.

Value

sc (unitless)

Examples

```
temp <- c(10, 20, 30)
salt <- c(30, 35, 40)
ebase_schmidt(temp = temp, salt = salt)
```

ebase_years *Estuarine Bayesian Single-station Estimation method for ecosystem metabolism for long time series*

Description

Estuarine Bayesian Single-station Estimation method for ecosystem metabolism for long time series

Usage

```
ebase_years(
  dat,
  Z,
  interval,
  ndays = 1,
  aprior = c(4, 2),
  rprior = c(300, 150),
  bprior = c(0.251, 0.125),
  bmax = 0.502,
  nogas = FALSE,
  doave = TRUE,
  maxinterp = 43200/interval,
  n.iter = 10000,
  update.chains = TRUE,
  n.burnin = n.iter * 0.5,
  n.chains = 3,
  n.thin = 10,
  model_file = NULL,
  ncores = NULL,
  quiet = TRUE,
  maxtry = 5
)
```

Arguments

dat	input data frame
Z	numeric as single value for water column depth (m) or vector equal in length to number of rows in dat
interval	timestep interval in seconds
ndays	numeric for number of days in dat for optimizing the metabolic equation, see details
aprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *a* parameter, see details
rprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *R* parameter, see details
bprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *b* parameter, see details
bmax	numeric value for the upper limit on the prior distribution for bprior, set as twice the default value of the mean
nogas	logical indicating if gas exchange is not included in the metabolic model, see details
doave	logical indicating if the average dissolved oxygen concentration is used as the starting value for the estimation (default), otherwise the first observation will be used if FALSE, see details

maxinterp	numeric value for minimum number of continuous observations that must not be interpolated within a group defined by ndays to assign as NA in output, see details
n.iter	number of MCMC iterations, passed to jags
update.chains	logical to run metab_update if chains do not converge
n.burnin	number of MCMC chains to delete, passed to jags
n.chains	number of MCMC chains to run, passed to jags
n.thin	number of nth iterations to save for each chain, passed to jags
model_file	NULL to use the model file included with the package or a path to a model text file can be used
ncores	numeric for number of cores to use for parallel processing, use NULL to suppress
quiet	logical to suppress progress messages to the console
maxtry	numeric for maximum number of times to retry the model if it fails

Details

ebase is run for each year in the supplied data. This facilitates running ebase on long time series by running the model sequentially on each year of data, with progress messages printed to the console if quiet = FALSE. The model run for each year will restart if it fails, up to maxtry times, after which processing continues with the next year. The model is run in parallel using the number of cores used set by ncores. If ncores = NULL, sequential processing is used. All other arguments are passed to ebase.

Similar results can be obtained by running ebase on the entire data set, but this function is useful for long time series where the model may fail for some years, e.g., when weather data may be missing for some years.

Value

Output identical to that returned by ebase, where the results for each year are appended to the data frame as the function progresses through the years. Note that the grp column that specifies the optimization period defined by ndays is unique to each year, e.g., values will be repeated across years.

Examples

```
# get one day of data
dat <- exdat[as.Date(exdat$DateTimeStamp, tz = 'America/Jamaica') == as.Date('2012-06-01'), ]

# run ebase, use more chains and iterations for a better fit, update.chains as T
ebase_years(dat, Z = 1.85, interval = 900, n.chains = 2, n.iter = 50,
 update.chains = FALSE)
```

exdat *Sample data from Apalachicola NERRS*

Description

Sample data from Apalachicola NERRS

Usage

```
exdat
```

Format

A `data.frame` object with 27648 rows and 6 columns

DateTimeStamp date and time, America/Jamaica time zone, 15 minute time step

DO_obs dissolved oxygen, mg/L

Temp water temperature, C

Sal salinity, psu

PAR total PAR, W/m2

WSpd num, m/s

See Also

Other utilities: `exres`

Examples

```
head(exdat)
```

exres *Example results for four days from Apalachicola NERRS*

Description

Example results for four days from Apalachicola NERRS

Usage

```
exres
```

Format

A data frame with 384 observations and 26 variables:

DateTimeStamp POSIXct, format: "2012-06-01 00:00:00" "2012-06-01 00:15:00" ...

Date Date, format: "2012-06-01" "2012-06-01" ...

grp Numeric, grouping variable defined by `ndays` in `ebase`

Z Numeric, depth in meters

DO_obs Numeric, observed dissolved oxygen, mmol m-3

DO_mod Numeric, modelled dissolved oxygen, mmol m-3

DO_modlo Numeric, lower credible interval of modelled dissolved oxygen

DO_modhi Numeric, upper credible interval of modelled dissolved oxygen

dDO Numeric, change in dissolved oxygen, mmol m-3 d-1

converge Character, convergence status

rsq Numeric, R-squared value

a Numeric, parameter a (mmol m-2 d-1)/(W m-2)

alo Numeric, lower credible interval of parameter a

ahi Numeric, upper credible interval of parameter a

b Numeric, parameter b, (cm hr-1)/(m2 s-2)

blo Numeric, lower credible interval of parameter b

bhi Numeric, upper credible interval of parameter b

P Numeric, production, O2 mmol m-2 d-1

Plo Numeric, lower credible interval of parameter P

Phi Numeric, upper credible interval of parameter P

R Numeric, respiratoin, O2 mmol m-2 d-1

Rlo Numeric, lower credible interval of parameter R

Rhi Numeric, upper credible interval of parameter R

D Numeric, gas exchange, O2 mmol m-2 d-1)

Dlo Numeric, lower credible interval of parameter D

Dhi Numeric, upper credible interval of parameter D

See Also

Other utilities: `exdat`

Examples

```
head(exres)
```

`fit_plot` *Plot observed and modeled dissolved oxygen*

Description

Plot observed and modeled dissolved oxygen

Usage

```
fit_plot(res, bygroup = FALSE, scatter = FALSE, showfit = TRUE)
```

Arguments

`res`	output data frame from `ebase`
`bygroup`	logical indicating if the plot is faceted by group
`scatter`	logical indicating if a scatter plot of modeled versus estimated dissolved oxygen is returned
`showfit`	logical indicating if a linear fit is shown in the plot, applies only if `scatter = TRUE`

Details

Dissolved oxygen (mmol/m3) is plotted as observed from the input data (points) and modeled (lines) based on inputs to `ebase` if `scatter = FALSE`. A scatter plot of modeled versus estimated dissolved oxygen is returned if `scatter = TRUE`, including a linear fit if `showfit = TRUE`. The plot is faceted by group based on the `ndays` argument to `ebase` if `bygroup = TRUE`. The r-squared value of the fit between modeled and observed dissolved oxygen is also shown in the facet label for the group if `bygroup = TRUE`.

Value

A `ggplot` object

Examples

```
# plot observed and modeled DO
fit_plot(exres)

# plot observed and modeled DO by group
fit_plot(exres, bygroup = TRUE)

# as scatter plot
fit_plot(exres, scatter = TRUE)

# as scatter plot by group
fit_plot(exres, scatter = TRUE, bygroup = TRUE)
```

`interp_plot` *Create a diagnostic plot showing interpolated values prior to metabolism estimates*

Description

Create a diagnostic plot showing interpolated values prior to metabolism estimates

Usage

```
interp_plot(
  dat,
  param = c("DO_obs", "DO_sat", "Z", "Temp", "Sal", "PAR", "WSpd", "sc"),
  Z,
  interval,
  ndays = 1
)
```

Arguments

`dat`	input data frame
`param`	character string of the parameter to plot, one of `DO_obs`, `DO_sat`, `Z`, `Temp`, `Sal`, `PAR`, `WSpd`, or `sc`
`Z`	numeric as single value for water column depth (m) or vector equal in length to number of rows in `dat`
`interval`	timestep interval in seconds
`ndays`	numeric for number of days in `dat` for optimizing the metabolic equation, see details

Details

Missing values in the input data can also be interpolated prior to estimating metabolism. This is the default behavior and it is the responsibility of the user to verify that these interpolated values are not wildly inaccurate. Missing values are linearly interpolated between non-missing values at the time step specified by the value in `interval`. This works well for small gaps, but can easily create inaccurate values at gaps larger than a few hours. The plot from this function can be used to visually assess the interpolated gaps.

Value

A `ggplot` object

Examples

```
library(dplyr)

# get four days of data
dat <- exdat %>%
  filter(lubridate::month(DateTimeStamp) == 6) %>%
  filter(lubridate::day(DateTimeStamp) %in% 1:4)

# create missing values
set.seed(222)
dat <- dat %>%
  slice_sample(prop = 0.9) %>%
  arrange(DateTimeStamp)

interp_plot(dat, Z = 1.85, interval = 900, param = 'DO_sat')
```

`metab_update` *Update metabolism jags fit*

Description

Update metabolism jags fit

Usage

```
metab_update(metabfit, update.chains, n.iter)
```

Arguments

`metabfit`	initial jags metabolism output
`update.chains`	logical to update, only if TRUE
`n.iter`	number of iterations

Details

This function is used by `ebase` and is not to be called directly by the user. It provides additional model iterations if convergence is not achieved.

Value

Updated jags metabolism output

prior_plot *Plot prior distributions for a, R, and b*

Description

Plot prior distributions for a, R, and b

Usage

```
prior_plot(
  aprior = c(4, 2),
  rprior = c(300, 150),
  bprior = c(0.251, 0.125),
  bmax = 0.502,
  n = 1000
)
```

Arguments

aprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *a* parameter, see details
rprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *R* parameter, see details
bprior	numeric vector of length two indicating the mean and standard deviation for the prior distribution of the *b* parameter, see details
bmax	numeric value for the upper limit on the prior distribution for bprior, set as twice the default value of the mean
n	numeric indicating number of random samples to draw from prior distributions

Details

This function produces a plot of the prior distributions that are used in ebase for the *a*, *R*, and *b* parameters for the optimization equation for estimating metabolism. The ebase function uses the same default values for the arguments for aprior, rprior, and bprior as required for this function. If the default values are changed for ebase, this function can be used to assess how changing characteristics of the prior distributions could influence the resulting parameter estimates and their posterior distributions (e.g., as shown with credible_plot.

All parameters follow a normal Gaussian distribution for the priors with the means and standard deviations defined by the arguments. All distributions are truncated to include only values greater than zero as required by the core metabolism equation. The upper limit for *b* is also set as twice the default value of the mean in the bprior argument. Truncated normal distributions are obtained using the rtruncnorm function with the number of random samples defined by the n argument.

The density curves for each parameter are normalized such that the peak values are always equal to 1.

Value

A ggplot object

Examples

```
# default plot
prior_plot()

# changing the mean and standard deviation for the b parameter
prior_plot(bprior = c(0.2, 0.05))
```

Index