Package ‘rMR’

October 14, 2022

Type Package

Title Importing Data from Loligo Systems Software, Calculating Metabolic Rates and Critical Tensions

Version 1.1.0

Date 2018-01-20

Author Tyler L. Moulton

Maintainer Tyler L. Moulton <tyler.moulton@cortland.edu>

Description Analysis of oxygen consumption data generated by Loligo (R) Systems respirometry equipment. The package includes a function for loading data output by Loligo's 'AutoResp' software (get.witrox.data()), functions for calculating metabolic rates over user-specified time intervals, extracting critical points from data using broken stick regressions based on Yeager and Ultsch (<DOI:10.1086/physzool.62.4.30157935>), and easy functions for converting between different units of barometric pressure.

License GPL-3

Imports

Depends R (>= 2.10), biglm

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-21 16:47:01 UTC

R topics documented:

`rMR-package` *Importing Data from Loligo Systems Software Output, Calculating Metabolic Rates and Critical Tensions*

Description

Analysis of oxygen consumption data generated by Loligo (R) Systems respirometry equipment. The package includes a function for loading data output by Loligo's 'AutoResp' software (get.witrox.data()), functions for calculating metabolic rates over user-specified time intervals, extracting critical points from data using broken stick regressions based on Yeager and Ultsch (1989), and easy functions for converting between different units of barometric pressure.

Details

Package:	rMR
Type:	Package
Version:	1.0.5
Date:	2017-09-07
License:	3

Author(s)

Tyler L. Moulton

Maintainer: Tyler L. Moulton <tyler.moulton@cortland.edu>

References

Benson, B.B., and Daniel Krause, Jr (1980). The concentration and isotopic fractionation of gases dissolved in freshwater in equilibrium with the atmosphere. 1. Oxygen: Limnology and Oceanography, vol. 25, no. 4, p. 662-671. doi: 10.4319/lo.1980.25.4.0662.

Gnaiger, Erich, and Hellmuth Forstner, eds. (2012). Polarographic oxygen sensors: Aquatic and physiological applications. Springer Science & Business Media. doi: 10.1007/978-3-642-81863-9.

Lumley, Thomas (2013). "biglm: bounded memory linear and generalized linear models". 0.9-1. `https://CRAN.R-project.org/package=biglm`.

McDonnell, Laura H., and Lauren J. Chapman (2016). "Effects of thermal increase on aerobic capacity and swim performance in a tropical inland fish." Comparative Biochemistry and Physiology Part A: Molecular & Integrative Physiology 199: 62-70. doi: 10.1016/j.cbpa.2016.05.018.

Mechtly, E. A. (1973). The International System of Units, Physical Constants and Conversion Factors. NASA SP-7012, Second Revision, National Aeronautics and Space Administration, Washington, D.C. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19730018242.pdf.

Roche, Dominique G., et al. (2013). "Finding the best estimates of metabolic rates in a coral reef fish." Journal of Experimental Biology 216.11: 2103-2110. doi: 10.1242/jeb.082925.

U.S. Geological Survey (2011). Change to solubility equations for oxygen in water: Office of Water Quality Technical Memorandum 2011.03, accessed July 15, 2011, at http://water.usgs.gov/admin/memo/QW/qw11.03.pdf.

Yeager, D. P. and Ultsch, G. R. (1989). Physiological regulation and conformation: a BASIC program for the determination of critical points. Physiological Zoology, 888-907. doi: 10.1086/physzool.62.4.30157935.

http://www.loligosystems.com/

See Also

`biglm`

`background.resp` *Determine the Background Respiration in a Respirometer*

Description

Takes user-defined start and end times to calculate the backround respiration rate in a respirometer.

Usage

```
background.resp(data, DO.var.name,
time.var.name = "std.time",
start.time, end.time, col.vec = c("black","red"),...)
```

Arguments

`data`	`data.frame` object to be used, best if formatted by `get.witrox.data()`.
`DO.var.name`	Column name of DO variable, formatted as a character string.
`time.var.name`	Column name of time variable as character string. Time column must be formatted as default class for datetime: `class = "POSIXct" "POSIXt"`, `strptime` format = `"%Y-%m-%d %H:%M:%S"`.
`start.time`	Input start time as character string of `strptime` format = `"%Y-%m-%d %H:%M:%S"`.
`end.time`	Input endtime as character string of `strptime` format = `"%Y-%m-%d %H:%M:%S"`.
`col.vec`	Specifies colors on plot in the following order: 1) scatterplot points, 2) regression color.
`...`	Passes on arguments to internal functions.

Value

Returns an object of method biglm. The slope of this funtion is the metabolic rate in input units/(default time).

Author(s)

Tyler L. Moulton

References

Thomas Lumley (2013). biglm: bounded memory linear and generalized linear models. R package version 0.9-1. https://CRAN.R-project.org/package=biglm.

See Also

as.POSIXct, strptime, biglm

Examples

```
##load data##
data(fishMR)

## create time variable in POSIXct format ##
fishMR$std.time <- as.POSIXct(fishMR$Date.time,
                    format = "%d/%m/%Y %I:%M:%S %p")

bgd.resp <-
background.resp(fishMR, "DO.mgL",
           start.time = "2015-07-02 16:05:00",
           end.time = "2015-07-02 16:35:00")
```

Barom.Press *Estimate Barometric Pressure*

Description

Function to estimate barometric pressure based on altitude.

Usage

```
Barom.Press(elevation.m, units = "atm")
```

Arguments

elevation.m	Elevation in meters above sea level.
units	Output units for barometric pressure must be one of: "atm", "kpa", or "mmHg".

Details

This is just a simple conversion function. Plug and chug, as they say...

Value

Returns numeric object of barometric pressure in specified units.

Author(s)

Tyler L. Moulton

References

Mechtly, E. A., 1973: The International System of Units, Physical Constants and Conversion Factors. NASA SP-7012, Second Revision, National Aeronautics and Space Administration, Washington, D.C. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19730018242.pdf.

U.S. Geological Survey (2011). Change to solubility equations for oxygen in water: Office of Water Quality Technical Memorandum 2011.03, accessed July 15, 2011, at http://water.usgs.gov/admin/memo/QW/qw11.03.pdf.

Examples

```
bar.pressure1 <- Barom.Press(elevation.m = 1000) # returns "atm"
bar.pressure2 <- Barom.Press(elevation.m = 1000, "kpa")
bar.pressure3 <- Barom.Press(elevation.m = 1000, "mmHg")
```

DO.saturation *Calculate Oxygen Saturation of Water*

Description

Calculate the percent saturation of oxygen in water given external temperature, barometric pressure, and recorded DO concentration in mg/L.

Usage

```
DO.saturation(DO.mgl, temp.C,
elevation.m = NULL, bar.press = NULL,
bar.units = NULL,
salinity, salinity.units)
```

Arguments

`DO.mgl`	Recorded DO concentration in mg/L.
`temp.C`	Temperature in degrees C.
`elevation.m`	Elevation in meters above sea level. EITHER `elevation.m` or `bar.press` must be specified.
`bar.press`	Barometric pressure in user defined units (bar.units)–defaults to `NULL`. EITHER `elevation.m` or `bar.press` must be specified.
`bar.units`	Units of barometric pressure, defaults to `NULL`. must be `"atm"`, `"kpa"` or `"mmHg"`
`salinity`	Salinity, either reported in parts per thousand (`"pp.thou"`) or microsiemens/cm (`"us"`).
`salinity.units`	Salinity units, must be `"pp.thou"` or `"us"`

Value

Returns numeric value of dissolved oxygen saturation.

Author(s)

Tyler L. Moulton

References

Mechtly, E. A., 1973: The International System of Units, Physical Constants and Conversion Factors. NASA SP-7012, Second Revision, National Aeronautics and Space Administration, Washington, D.C. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19730018242.pdf.

U.S. Geological Survey (2011). Change to solubility equations for oxygen in water: Office of Water Quality Technical Memorandum 2011.03, accessed July 15, 2011, at http://water.usgs.gov/admin/memo/QW/qw11.03.pdf.

See Also

`Eq.Ox.conc`, `DO.unit.convert`,

Examples

```
DO.sat1 <- DO.saturation(DO.mgl = 5.5,
temp.C = 20, elevation.m = 1000)

DO.sat2 <- DO.saturation(DO.mgl = 5.5,
temp.C = 20, bar.press = 674.1, bar.units = "mmHg")

DO.sat1
DO.sat2

# Will ya look at that...
```

DO.unit.convert — *Convert Between Different Common Units of DO Concentration*

Description

Converts between different different units of DO concentration. Takes into account ambient temperature, pressure and salinity.

Usage

```
DO.unit.convert(x, DO.units.in, DO.units.out, bar.units.in,
bar.press, temp.C, bar.units.out = "mmHg",
salinity = 0, salinity.units = "pp.thou")
```

Arguments

`x`	Value or object of class numeric to be converted.
`DO.units.in`	Units of dissolved oxygen concentration measured, i.e. to be converted from. Must be `"mg/L"`, `"PP"` (partial pressure), or `"pct"` (percent). If `"PP"`, the units of partial pressure must be equal to `bar.units.in`.
`DO.units.out`	Units of dissolved oxygen concentration desired, i.e. to be converted to. Must be `"mg/L"`, `"PP"`, or "pct".
`bar.units.in`	Units of barometric pressure of user specified barometric pressure measurement. Must take value of `"atm"`, `"kpa"`, or `"mmHg"`.
`bar.press`	Ambient barometric pressure measurement
`temp.C`	Water temperature measured in degrees C
`bar.units.out`	Used in internal calculation, only visible if output DO.units.out = `"PP"`. Must take value of `"atm"`, `"kpa"`, or `"mmHg"`.
`salinity`	Salinity, either reported in parts per thousand (`"pp.thou"`) or microsiemens/cm (`"us"`)
`salinity.units`	Salinity units, must be `"pp.thou"` or `"us"`

Value

Numeric object representing dissolved oxygen concentration in the units specified by `DO.units.out`.

Note

Use this function on entire data columns to convert them to desired units before analysing with functions like `MR.loops` and `get.pcrit`.

Author(s)

Tyler L. Moulton

References

Mechtly, E. A., 1973: The International System of Units, Physical Constants and Conversion Factors. NASA SP-7012, Second Revision, National Aeronautics and Space Administration, Washington, D.C. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19730018242.pdf.

U.S. Geological Survey (2011). Change to solubility equations for oxygen in water: Office of Water Quality Technical Memorandum 2011.03, accessed July 15, 2011, at http://water.usgs.gov/admin/memo/QW/qw11.03.pdf.

See Also

plot, plotRaw, cbind, Eq.Ox.conc, DO.saturation,

Examples

```
## on a single value ##

DO.pct<- DO.unit.convert(x= 125.6863, DO.units.in = "PP",
                DO.units.out = "pct",
                bar.units.in = "mmHg", bar.press = 750, temp.C =15)

## Apply to a column in a 'data.frame' class object ##

## load data ##
data(fishMR)

## create time variable in POSIXct format ##
fishMR$std.time <- as.POSIXct(fishMR$Date.time,
                    format = "%d/%m/%Y %I:%M:%S %p")

head(fishMR)

#note that DO data are in mg/L (DO.mgL) and
#that there is an instantaneous temperature column
#(temp.C) and a pressure column (Bar.Pressure.hpa)

DO.pct.col.a <- DO.unit.convert(fishMR$DO.mgL, DO.units.in = "mg/L",
                DO.units.out = "pct",
                bar.units.in = "kpa", bar.press = 101.3,
                temp.C = fishMR$temp.C,
                bar.units.out = "kpa")
DO.pct.col.b<- DO.unit.convert(fishMR$DO.mgL, DO.units.in = "mg/L",
                DO.units.out = "pct",
                bar.units.in = "kpa", bar.press = 101.3,
                temp.C = fishMR$temp.C)
head(DO.pct.col.a)
head(DO.pct.col.b)

# Now with df #

fishMR2 <- as.data.frame(cbind(fishMR, DO.pct.col.a))
```

```
par(mfrow = c(1,2))
plotRaw(data = fishMR, DO.var.name = "DO.mgL",
          start.time = "2015-07-03 06:15:00",
          end.time = "2015-07-03 08:05:00",
          main = "DO (mg/L) vs time",
          xlab = "time",
          ylab = "DO (mg/L)")


plotRaw(data = fishMR2, DO.var.name = "DO.pct.col.a",
          start.time = "2015-07-03 06:15:00",
          end.time = "2015-07-03 08:05:00",
          main = "DO (percent saturation) vs time",
          xlab = "time",
          ylab = "DO (percent saturation)")
```

`Eq.Ox.conc` *Equilibrium Concentration of Dissolved Oxygen in Water*

Description

Determines equilibrium dissolved oxygen concentration in water based on pressure and temperature. An estimate for barometric pressure can be generated by supplying the temperature and elevation (calculation by `Barom.Press()`)

Usage

```
Eq.Ox.conc(temp.C, elevation.m = NULL,
bar.press = NULL, bar.units = "mmHg",
out.DO.meas = "mg/L",
salinity = 0, salinity.units = "pp.thou")
```

Arguments

`temp.C`	Water temperature in degrees C
`elevation.m`	Elevation in meters. Default = `NULL`. Must be `NULL` if `bar.press` takes a value.
`bar.press`	Barometric pressure. Default = `NULL`. Must be `NULL` if `elevation.m` takes a value.
`bar.units`	Units of barometric pressure for value supplied in bar.press. Must be `NULL`, `"atm"`, `"kpa"`, or`"mmHg"`.
`out.DO.meas`	Units of dissolved oxygen concentration
`salinity`	Salinity, either reported in parts per thousand (`"pp.thou"`) or microsiemens/cm (`"us"`)
`salinity.units`	Salinity units, must be `"pp.thou"` or `"us"`.

Value

Returns object of class numberic of full equilibrium dissolved oxygen concentration.

Author(s)

Tyler L. Moulton

References

Benson, B.B., and Daniel Krause, Jr (1980). The concentration and isotopic fractionation of gases dissolved in freshwater in equilibrium with the atmosphere. 1. Oxygen: Limnology and Oceanography, vol. 25, no. 4, p. 662-671. doi: 10.4319/lo.1980.25.4.0662.

Gnaiger, Erich, and Hellmuth Forstner, eds. Polarographic oxygen sensors: Aquatic and physiological applications. Springer Science & Business Media, 2012. doi: 10.1007/978-3-642-81863-9.

Mechtly, E. A., 1973: The International System of Units, Physical Constants and Conversion Factors. NASA SP-7012, Second Revision, National Aeronautics and Space Administration, Washington, D.C. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19730018242.pdf.

U.S. Geological Survey (2011). Change to solubility equations for oxygen in water: Office of Water Quality Technical Memorandum 2011.03, accessed July 15, 2011, at http://water.usgs.gov/admin/memo/QW/qw11.03.pdf.

See Also

DO.saturation DO.unit.convert Barom.Press

Examples

```
eqO2.1 <- Eq.Ox.conc(temp.C = 20, elevation.m = 1000, bar.units = NULL)

eqO2.2 <- Eq.Ox.conc(temp.C = 20,
bar.press = 674.1, bar.units = "mmHg")

eqO2.1
eqO2.2
```

---

fishMR — *Gnathonemus Respirometry Trial Data*

Description

This is a dataset acquired duirng a respirometry trial on a mormyrid of the species Gnathonemus victoriae. It went great. There are several "loops" (open/close the respirometer) to establish routine metabolic rate, as well as an extended "closed" period to capture the "P.crit", the point at which the linear relationship between metabolic rate and ambient dissolved oxygen changed.

Usage

```
data(fishMR)
```

Format

A data frame with 64239 observations on the following 7 variables.

`Date.time` a character vector

`times` a numeric vector

`Bar.Pressure.hpa` a numeric vector

`Phase` a numeric vector

`temp.C` a numeric vector

`DO.mgL` a numeric vector

References

Moulton Tyler L., Chapman Lauren J., Krahe Rudiger. Manuscript in Prep.

Examples

```
data(fishMR)
str(fishMR)
head(fishMR)
```

`get.pcrit` *Calculate Critical Tension for Rate Processes*

Description

Determines the critical point of a rate process based on the broken stick model featured in Yeager and Ultsch (1989). The two regressions are selected based on the break point which minimizes the total residual sum of squares. The rate process that this package is designed for is metabolic rate (`MR.var.name`), and it is regressed on ambient dissolved oxygen concentration (`DO.var.name`). However, the same function can be used for other processes.

If metabolic rate has not already been calculated and the user wishes to calculate metabolic rates directly from oxygen concentration measurements, let `MR.var.name= NULL`. Then indicate the name of the time variable AND the time interval (i.e. the width of the time window over which you will estimate instantaneous metabolic rates).

Usage

```
get.pcrit(data, DO.var.name, MR.var.name = NULL, Pcrit.below,
             time.interval, time.var = NULL,
             start.time, stop.time, time.units = "sec",
             Pcrit.type = "both", syst.vol = NULL,
             col.vec = c("black", "gray60", "red", "blue"),...)
```

Arguments

`data`	Data to be used.
`DO.var.name`	Variable name of oxygen concentration variable, formatted as character string. To be used for determination of critical point. If using pre-calculated instantaneous metabolic rates (MR) to conduct Pcrit, this should be specified by `MR.var.name`. If using raw oxygen values with time stamps to compute instantaneous MR within the function, `get.pcrit` requires `time.interval` to be specified.
`MR.var.name`	Metabolic rate variable name, formatted as character. Default = `NULL`. If this argument takes a value, Pcrit will be calculated by regressing `MR.var.name` on `DO.var.name`. `time.var` and time.interval should, in this case, take no value. If `MR.var.name` is left as`NULL`, then instantaneous metabolic rates (MR) at specified time intervals (see `time.interval`) from `DO.var.name` and `time.var`.
`Pcrit.below`	DO concentration below which you are confident that Pcrit occurs. Accelerates process by reducing the number of iterations required to find Pcrit. Data points featuring DO conc > `Pcrit.below` are still used to calculate regressions for model.
`time.interval`	If `MR.var.name = NULL`, specify interval in seconds over which to calculate instantaneous MR.
`time.var`	Column name for indexing (time) variable used to calculate instantaneous Metabolic Rate (MR) from oxygen concentration data (specified by `DO.var.name`). Must be a character string.
`start.time`	Beginning of time interval over which to evaluate data for Pcrit. Required if `MR.var.name = NULL`. Must be character string in the default `POSIXct` format (matches format of `time.var`).
`stop.time`	End of time interval over which to evaluate data for Pcrit. Required if `MR.var.name = NULL`. Must be character string in the default `POSIXct` format (matches format of `time.var`).
`time.units`	Units of time in MR calculation. Defaults to `"sec"`, must be `"sec"`, `"min"`, or `"hr"`. Required if `MR.var.name = NULL`.
`Pcrit.type`	Either `"lm"` to draw a vertical line at the Pcrit as determined by the intersection point of the best fit lines(Yeager and Ultsch 1989) or `"midpoint"` as determined by the midpoint between the two points on either side of the Pcrit (Yeager and Ultsch 1989). `"both"` will plot both as vertical lines on the plot. NULL will plot neither. Both values are returned in the output.
`syst.vol`	Enter the system volume in Liters. If `syst.vol` takes a value, then an additional column, `MR.vol.adj` will be added to `$DATA`. The values in this colum will reflect mg of oxygen consumed. Otherwise, the returned MR variable (and in `data`) on the y axis of the plot will represent `DO.var.name` units/time.
`...`	Arguments passed on to internal functions.
`col.vec`	Specifies colors on plot in the following order: 1) scatterplot points representing instantaneous MR, 2) regression lines color, 3) vertical line representing Pcrit using the intersect method (`Pcrit.type = "lm"`), 4)vertical line representing Pcrit using the midpoint method (`Pcrit.type = "midpoint"`).

Details

This calculates the critical oxygen tension for a change in metabolic rate. It is a simple broken stick model which evaluates the data at dissolved oxygen values recorded within the specified time frame. The data of MR and DO are ordered by decreasing DO value. Then, the function iteratively calculates the total residual sum of squares (using `tot.rss`) of two linear models, one spanning from `Pcrit.below` to `Pcrit.below` - i, the other with a range from the minimum DO value to `Pcrit.below` - (i + 1). The broken stick model resulting in the lowest total residual sum of squares is selected.

Value

A scatterplot of MR ~ DO is generated with the two regression lines (in gray). Also returns a list of 6. `$Pcrit.lm` is the Pcrit given by the intersection of the two best fit lines (vertical red dashed line). `$Pcrit.mp` is the Pcrit using the midpoint method (vertical blue dotted vertical line). `$P$Adj.r2.above` gives the adjusted R2 value of the relationship between MR~DO above the critical point, and likewise, `$Adj.r2.below` gives the R2 below the critical point. The other two elements are `lm` class objects calculated from the regression slopes above and below the break point in the broken stick model (which is not necessarily the same point as where the regression lines intersect!).

Author(s)

Tyler L. Moulton

References

Yeager, D. P. and Ultsch, G. R. (1989). Physiological regulation and conformation: a BASIC program for the determination of critical points. Physiological Zoology, 888-907. doi: 10.1086/physzool.62.4.30157935.

See Also

`tot.rss`, `strptime`, `as.POSIXct`,

Examples

```
## set data ##
data(fishMR)

## create time variable in POSIXct format ##
fishMR$std.time <- as.POSIXct(fishMR$Date.time,
                    format = "%d/%m/%Y %I:%M:%S %p")

Pcrit1 <-get.pcrit(data = fishMR, DO.var.name = "DO.mgL",
                    Pcrit.below = 2,
                    time.var = "std.time",
                    time.interval = 120,
                    start.time = "2015-07-03 06:15:00",
                    stop.time = "2015-07-03 08:05:00")
## MR units in mgO2 / sec
```

```
## Change time interval ##
Pcrit2 <-get.pcrit(data = fishMR, DO.var.name = "DO.mgL",
                    Pcrit.below = 2,
                    time.var = "std.time",
                    time.interval = 60,
                    start.time = "2015-07-03 06:15:00",
                    stop.time = "2015-07-03 08:05:00",
                    time.units = "min")
## MR units in mgO2 / min


Pcrit3 <-get.pcrit(data = fishMR, DO.var.name = "DO.mgL",
                    Pcrit.below = 2,
                    time.var = "std.time",
                    time.interval = 60,
                    start.time = "2015-07-03 06:15:00",
                    stop.time = "2015-07-03 08:05:00",
                    time.units = "hr",
                    ylab = "Met Rate (mg O2 L-1 hr-1)")
## MR units in mgO2 / hr

## syst vol specified at 0.75 L ##

Pcrit3a <-get.pcrit(data = fishMR, DO.var.name = "DO.mgL",
                    Pcrit.below = 2,
                    time.var = "std.time",
                    time.interval = 60,
                    start.time = "2015-07-03 06:15:00",
                    stop.time = "2015-07-03 08:05:00",
                    time.units = "hr",
                    syst.vol = 0.75,
                    ylab = "Met Rate (mg O2 / hr)")
## MR units in mgO2 / hr

## No vertical lines on plot

Pcrit4 <-get.pcrit(data = fishMR, DO.var.name = "DO.mgL",
                    Pcrit.below = 2,
                    time.var = "std.time",
                    time.interval = 60,
                    start.time = "2015-07-03 06:15:00",
                    stop.time = "2015-07-03 08:05:00",
                    time.units = "hr",
                    ylab = "Met Rate (mg O2 L-1 hr-1)",
                    Pcrit.type = "")
```

get.witrox.data *Load Data from 'AutoResp' Software Generated .txt Files*

Description

Allows user to import data from Loligo (R) Systems' 'Autoresp' software-generated text files into a R data.frame (class `data.frame`)

Usage

```
get.witrox.data(data.name, lines.skip, delimit = "tab",
choose.names = F, chosen.names = NULL,
format)
```

Arguments

`data.name`	Full data file name as character string.
`lines.skip`	The lines in the header to be skipped. If `choose.names = FALSE`, then skip all lines up to the column names. If `choose.names =TRUE`, skip all lines including column names.
`delimit`	Choose the delimiter. Defaults to tab delimited. Can take values of `"tab"`, `"space"`, or `"comma"`. If importing from an excel file, save the file as a .csv file, then use the delimiter argument `"comma"`
`choose.names`	logical: if `FALSE`, then names are automatically derived from the names of the text file. Sometimes, this can be a problem if there are tabs or commas included in odd places in the column name line of the text file. If `TRUE`, user must specify a vector of column names–see `lines.skip` and `chosen.names`.
`chosen.names`	If `choose.names = TRUE`, chosen.names must be a vector of character strings for use as `column.names`.
`format`	This is the format that the date-time column is formatted in by auto resp–This must be the FIRST COLUMN. The default format is `"%d/%m/%Y %I:%M:%S %p"`. Another common format is `"%d/%m/%Y/%I:%M:%S %p"`. See `strptime` for more directions on formatting the date and time.

Value

Returns an object of class data.frame, with `std.time` as the last column, which is in the default standard `POSIXct` date-time format.

Author(s)

Tyler L. Moulton

See Also

`strptime`, `as.POSIXct`,

Examples

```
# Requires a text file. Download fish_MR.txt from github repository and
# accompanying readme file at:
# https://github.com/tyler-l-moulton/rMR
```

MR.loops — *Calculate Metabolic Rates from Multiple Closed Respirometry Loops*

Description

This function calculates the metabolic rates from multiple closed respirometry loops simultaneously. Requires lots of user input, but is easy to manipulate. Returns list of metabolic rates, as well as the average metabolic rate and the standard deviation of the sample of metabolic rates, as well as biglm objects for each section of data used to calculate MRs.

Usage

```
MR.loops(data, DO.var.name, time.var.name = "std.time",
             in.DO.meas = "mg/L", out.DO.meas = "mg/L",
             start.idx, stop.idx, syst.vol = 1,
             background.consumption = 0,
             background.indices = NULL,
             temp.C, elevation.m = NULL,
             bar.press = NULL, bar.units = "atm",
             PP.units, time.units = "sec",
             col.vec = c("black","red"),...)
```

Arguments

`data`	Must include a time variable in standard `POSIXct` format. eg `"2016-09-25 15:30:00 EST"`.
`DO.var.name`	Column name of DO variable, must be entered as character string.
`time.var.name`	Column name of time variable (which is in `POSIXct` format) as character. defaults to "std.time" as generated from `get.witrox.data()`.
`in.DO.meas`	Units of DO measurement entered in the DO variable column: must be one of `"mg/L"` for miligrams/liter, `"PP"` for partial pressure, `"pct"` for saturation percent.
`out.DO.meas`	Units of DO measurement returned for metabolic rate: must be one of `"mg/L"` for miligrams/liter, `"PP"` for partial pressure (units determined by `bar.units`), `"pct"` for saturation percent.
`start.idx`	Character class value or vector matching `POSIXct` object coding for date time. Each element of the vector represents the start time of a new loop for calculation of metabolic rates.
`stop.idx`	Character class value or vector matching `POSIXct` object coding for date time. Each element of the vector represents the stop time of a new loop for calculation of metabolic rates.
`syst.vol`	System volume in Liters (defaults to 1 L).

background.consumption
: Default = 0. If using a one point calibration for background, simply set background.consumption equal to the value of the calculated respiration rate. If using a multi-point calibration, enter a vector of background respiration rates, and enter a corresponding vector for background.indices. CAUTION: The slope must be entered in raw units (i.e. those specified in the input data.frame in the data argument). For example, if the DO.var.name column is recorded in "mg/L", and the POSIXct format goes to the resolution of seconds, background consumption units would need to be entered in mgO2 / sec.

background.indices
: If using a multi-point calibration to set the background respiration rate, enter a vector of times for when the respiration rates were calculated. There should be one time point per corresponding value in background.consumption. The background respiration rate is continually factored into all calculations of metabolic rate. The elements of the vector must entered as character strings conforming to the POSIXctformat specified in the time.var.name column.

temp.C
: Water temperature in degrees C.

elevation.m
: Elevation in m. Only required if bar.press = NULL.

bar.press
: barometric pressure in units defined by bar.units argument. Only required if elevation.m = NULL.

bar.units
: Units of barometric pressure used as input and in output if DO.meas.out = "PP". Acceptable arguments: "mmHg", "atm", "kpa".

PP.units
: Units of barometric pressure used for "PP".

time.units
: Denominator for metabolic rate, also displayed as units on X-axis. Acceptable arguments: "hr", "min", "sec".

col.vec
: Specifies colors on plot in the following order: 1) scatterplot points, 2) regression lines color.

...
: Arguments passed on to internal functions

Value

Returns a list of 2. $MR.summary is of class data.frame with 3 columns: $MR (metabolic rate in user specified units, this is the same as the slope in each linear model), $sd.slope (standard deviation of slopes calculation), $r.square (adjusted r square value from each model). This second object is a list of biglm objects, each one representing a metabolic loop (see McDonnell and Chapman 2016).

Author(s)

Tyler L. Moulton

References

McDonnell, Laura H., and Lauren J. Chapman (2016). "Effects of thermal increase on aerobic capacity and swim performance in a tropical inland fish." Comparative Biochemistry and Physiology Part A: Molecular & Integrative Physiology 199: 62-70. doi: 10.1016/j.cbpa.2016.05.018.

Roche, Dominique G., et al. (2013). "Finding the best estimates of metabolic rates in a coral reef fish." Journal of Experimental Biology 216.11: 2103-2110. doi: 10.1242/jeb.082925.

See Also

as.POSIXct, strptime, background.resp, Barom.Press, Eq.Ox.conc, biglm,

Examples

```
## load data ##
data(fishMR)

## create time variable in POSIXct format ##
fishMR$std.time <- as.POSIXct(fishMR$Date.time,
                    format = "%d/%m/%Y %I:%M:%S %p")

## calc background resp rate
bgd.resp <-
    background.resp(fishMR, "DO.mgL",
                    start.time = "2015-07-02 16:05:00",
                    end.time = "2015-07-02 16:35:00",
                    ylab = "DO (mg/L)", xlab = "time (min)")

bg.slope.a <- bgd.resp$mat[2]


starts <- c("2015-07-03 01:15:00", "2015-07-03 02:13:00",
            "2015-07-03 03:02:00", "2015-07-03 03:50:00",
            "2015-07-03 04:50:00")

stops <- c("2015-07-03 01:44:00", "2015-07-03 02:35:30",
           "2015-07-03 03:25:00", "2015-07-03 04:16:00",
           "2015-07-03 05:12:00")

metR <- MR.loops(data = fishMR, DO.var.name ="DO.mgL",
                 start.idx = starts, time.units = "hr",
                 stop.idx = stops, time.var.name = "std.time",
                 temp.C = "temp.C", elevation.m = 1180,
                 bar.press = NULL, in.DO.meas = "mg/L",
                 background.consumption = bg.slope.a,
                 ylim=c(6, 8))

metR$MR.summary


## now lets assume we ran a control loop for background rate
## before and after we ran the MR loops
## let:

bg.slope.b <-bg.slope.a -0.0001
metRa <- MR.loops(data = fishMR, DO.var.name ="DO.mgL",
                 start.idx = starts, time.units = "hr",
                 stop.idx = stops, time.var.name = "std.time",
                 temp.C = "temp.C", elevation.m = 1180,
                 bar.press = NULL, in.DO.meas = "mg/L",
                 background.consumption = c(bg.slope.a, bg.slope.b),
```

```
                  background.indices = c("2015-07-02 16:20:00",
                                         "2015-07-03 06:00:00"),
                  ylim=c(6, 8))


metRa$MR.summary

# note that the calculated slopes
# diverge as time increases. This is
# because the background respiration
# rate is increasing.

metR$MR.summary-metRa$MR.summary

## This looks great, but you need to check your start and
## stop vectors, otherwise, you could end up with some
## atrocious loops, e.g.:

starts <- c("2015-07-03 01:15:00", "2015-07-03 02:13:00",
            "2015-07-03 03:02:00", "2015-07-03 03:50:00",
            "2015-07-03 04:50:00")

stops <- c("2015-07-03 01:50:00", "2015-07-03 02:35:30",
           "2015-07-03 03:25:00", "2015-07-03 04:16:00",
           "2015-07-03 05:12:00")

metRb <- MR.loops(data = fishMR, DO.var.name ="DO.mgL",
         start.idx = starts,
         stop.idx = stops, time.var.name = "std.time",
         temp.C = "temp.C", elevation.m = 1180,
         bar.press = NULL, in.DO.meas = "mg/L",
         background.consumption = bg.slope.a,
         ylim=c(6,8))
```

`plotRaw` *Plotting Data from Witrox*

Description

A good way to visualize your respiro data to get an idea of where to set up the time intervals in functions like `MR.loops()` or `get.pcrit()`.

Usage

```
plotRaw(data, DO.var.name, time.var.name = "std.time",
              start.time = data$x[1],
              end.time = data$x[length(data$x)],...)
```

Arguments

`data`	data object for plotting
`DO.var.name`	A character string matching the column header for the DO variable column.
`time.var.name`	Column name of time (or x) axis in character class.
`start.time`	Character string specifying left bound x limit in `strptime` compatable format.
`end.time`	Character string specifying right bound x limit in `strptime` compatable format.
`...`	Arguments passed on to internal functions.

Details

`start.time` and `end.time` arguments must match the `time.var.name` column's format for date time.

Value

Plot showing the overall metabolic data

Author(s)

Tyler L. Moulton

See Also

`plot`, `strptime`, `get.pcrit`, `MR.loops`,

Examples

```
## load data ##

data(fishMR)
## create time variable in POSIXct format ##
fishMR$std.time <- as.POSIXct(fishMR$Date.time,
                    format = "%d/%m/%Y %I:%M:%S %p")

plotRaw(data = fishMR, DO.var.name = "DO.mgL",
         start.time = "2015-07-03 06:15:00",
         end.time = "2015-07-03 08:05:00")

plotRaw(fishMR, DO.var.name = "DO.mgL",
         start.time = "2015-07-03 01:00:00",
         end.time = "2015-07-03 05:12:00")
```

sumsq — *Sum of squares*

Description

Internal Function for Use in Package for Calculating Sum of Squares of a Vector

Usage

```
sumsq(x)
```

Arguments

x	Numeric vector to be evaluated

Details

Internal function for package

Value

The sum of squares of the vector

Author(s)

Tyler L. Moulton

Examples

```
vec <- sample(c(100:120), 50, replace = TRUE)
sumsq(vec)
```

tot.rss — *Total Residual Sum of Squares for Broken Stick Model*

Description

Calculates the total residual sum of squares for broken stick model (2 part)

Usage

```
tot.rss(data, break.pt, xvar, yvar)
```

Arguments

`data`	data frame for calculating total residual sum of squares.
`break.pt`	This is the data point at which the data are split for a broken stick model.
`xvar`	The x-variable in the data frame for broken stick model.
`yvar`	The y-variable in the data frame for broken stick model.

Value

The residual sum of squares of a broken stick model with a specified break point.

Author(s)

Tyler L. Moulton

See Also

codesumsq codeget.pcrit

Examples

```
## load data ##
data(fishMR)

## subset data to appropriate region ##

data<-fishMR[fishMR$DO.mgL < 4,]
data$times <- data$times-min(data$times)
data<-data[data$times< 6800,]

## calculate total RSS for different breakpoints ##

a1 <- tot.rss(data, break.pt = 4000,
xvar = "times", yvar = "DO.mgL")
a2 <- tot.rss(data, break.pt = 4250,
xvar = "times", yvar = "DO.mgL")
a3 <- tot.rss(data, break.pt = 4500,
xvar = "times", yvar = "DO.mgL")
a4 <- tot.rss(data, break.pt = 4750,
xvar = "times", yvar = "DO.mgL")
a5 <- tot.rss(data, break.pt = 5000,
xvar = "times", yvar = "DO.mgL")
a6 <- tot.rss(data, break.pt = 5250,
xvar = "times", yvar = "DO.mgL")

# a5 represents the break point for the
# best broken stick linear model of the
# above 6 options.
```

Index