Package ‘rMR’

October 6, 2016

Type Package

Title Importing data from Loligo Systems softwares, calculating metabolic rates, and critical tensions

Version 1.0

Date 2016-09-07

Author Tyler L. Moulton

Maintainer Tyler L. Moulton <tyler.moulton@mail.mcgill.ca>

Description This package is written to accompany Loligo Systems respirometry equipment. The package includes a function for loading data output by 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), easy functions for converting between different units of barometric pressure, etc. Requires ``biglm'' in order reset RAM memory during computations, allowing for the analysis of large datasets.

License GPL-3

Imports biglm

R topics documented:

`rMR-package` *Importing data from Loligo Systems softwares, calculating metabolic rates, and critical tensions*

Description

This package is written to accompany Loligo Systems respirometry equipment. The package includes a function for loading data output by 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), easy functions for converting between different units of barometric pressure, etc. Requires "biglm" in order reset RAM memory during computations, allowing for the analysis of large datasets.

Details

Package:	rMR
Type:	Package
Version:	1.0.1
Date:	2016-09-07
License:	3

Author(s)

Tyler L. Moulton

Maintainer: Tyler L. Moulton <tyler.moulton@mail.mcgill.ca>

References

Gnaiger, Erich, and Hellmuth Forstner, eds. Polarographic oxygen sensors: Aquatic and physiological applications. Springer Science & Business Media, 2012.

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

McDonnell, Laura H., and Lauren J. Chapman. "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 (2016): 62-70.

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.

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

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.

https://en.wikipedia.org/wiki/Barometric_formula

http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

See Also

biglm

`background.resp` *A function for determining the background respiration in a respirometer*

Description

background.resp 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, ...)
```

Arguments

`data`	Dataframe 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".
`...`	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. http://CRAN.R-project.org/package=biglm

See Also

as.POSIXct, strptime, biglm

Examples

```
data(fishMR)

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)")
```

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://en.wikipedia.org/wiki/Barometric_formula

http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

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)
```

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"

Value

Returns numeric value of dissolved oxygen saturation.

Author(s)

Tyler L. Moulton

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

This function converts between different different units of DO concentration. Takes into account ambient temperature and pressure.

Usage

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

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"

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://en.wikipedia.org/wiki/Barometric_formula

http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

See Also

plot, plot.raw, 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)

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 <- cbind(fishMR, DO.pct.col.a)

par(mfrow = c(1,2))
plot.raw(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)")


plot.raw(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 = NULL,
out.DO.meas = "mg/L")
```

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

Value

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

Author(s)

Tyler L. Moulton

References

Gnaiger, Erich, and Hellmuth Forstner, eds. Polarographic oxygen sensors: Aquatic and physiological applications. Springer Science & Business Media, 2012.

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://en.wikipedia.org/wiki/Barometric_formula

http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

See Also

`DO.saturation` `DO.unit.convert` `Barom.Press`

Examples

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

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 'close' 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

`timestamp` a numeric vector

`Bar.Pressure.hpa` a numeric vector

`Phase` a numeric vector

`temp.C` a numeric vector

`DO.mgL` a numeric vector

`std.time` a POSIXct

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.

Usage

```
get.pcrit(data, DO = NULL, MR = NULL,
Pcrit.below, idx.interval = NULL,
index.var = "std.time", start.idx = NULL,
stop.idx = NULL, time.units = "sec"...)
```

Arguments

`data`	Data to be used.
`DO`	Variable name of dissolved oxygen variable, formatted as character string. To be used for determination of critical point. Default = 'NULL'. Cannot have value unless MR = 'NULL'. Requires 'idx.interval' to be specified.
`MR`	Metabolic rate variable name, formatted as character. Default = 'NULL'. Cannot have value unless DO = 'NULL'.
`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.
`idx.interval`	If MR = NULL, specify interval in seconds over which to calculate instantaneous MR.
`index.var`	Column name for indexing(time) variable used to calculate instantaneous MR.
`start.idx`	Beginning of time interval over which to evaluate data for Pcrit. Default = 'NULL'. If calculating Pcrit directly from MR (i.e. DO = 'NULL') start.idx will not be evaluated by the function.
`stop.idx`	End of time interval over which to evaluate data for Pcrit. Default = 'NULL'. If calculating Pcrit directly from MR (i.e. DO = 'NULL') stop.idx will not be evaluated by the function.
`time.units`	Units of time in MR calculation. Defaults to "sec", must be "sec", "min", or "hr".

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 < 'Pcrit.below'. 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. The Pcrit is the DO value at the intersection point of the broken stick model. This is indicated by a blue circle on the plot.

Value

Returns a list of 27. $Pcrit is the Pcrit given by the intersection of the two best fit lines. $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 24 list elements are from the two linear models (denoted as above and below the pcrit) in the broken stick model.

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, 888-9

See Also

tot.RSS, tot.RSS, strptime, as.POSIXct,

Examples

```
## set data ##

data(fishMR)

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

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

Pcrit3 <-get.pcrit(data = fishMR, DO = "DO.mgL",
                    Pcrit.below = 2,
                    idx.interval = 60,
                    start.idx = "2015-07-03 06:15:00",
                    stop.idx = "2015-07-03 08:05:00",
                    time.units = "hr",
                    ylab = "Met Rate (mg O2 / hour)")
```

get.witrox.data *Load data from AutoResp software generated txt files*

Description

Allows user to import data from loligo autoresp software 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`	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 as.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.
# https://github.com/tyler-l-moulton/rMR
#

#specify chosen.names for header
data.names <- c("date.time","timestamp", "pres.hpa", "CH1.phase",
                "CH1.temp", "CH1.DO.mgl")

##Un-comment the following 4 lines
# d <- get.witrox.data ("fish_MR.txt", lines.skip = 10,
#                     choose.names = TRUE,
#                     chosen.names = data.names,
#                     format = "%d/%m/%Y %I:%M:%S %p")
```

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, system.vol = 1,
              background.consumption = 0,
              background.indices = NULL,
              temp.C, elevation.m = NULL,
              bar.press = NULL, bar.units = "atm",
              PP.units, time.units = "sec", ...)
```

Arguments

data	'data' must include a time variable in standard as.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 as.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 as.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 as.POSIXct object coding for date time. Each element of the vector represents the stop time of a new 'loop' for calculation of metabolic rates.

`system.vol` System volume in Liters (defaults to 1 L).

`background.consumption`
Default is 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 as.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 as.POSIXct format 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".

`...` arguments passed on to internal functions

Value

Returns a list of 2. $MR.summary is a '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. "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 (2016): 62-70.

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

See Also

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

Examples

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


## 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))
```

`names.vec` *header for fishMR data frame*

Description

Vector of column names for a data frame imported with 'get.witrox.data' function from AutoResp

Usage

```
data("names.vec")
```

Format

The format is: chr [1:6] "Date.time" "timestamp" "Bar.Pressure.hpa" "Phase" ...

Details

Example header for a text file from Auto Resp.

Examples

```
data(names.vec)

str(names.vec)
```

---

plot.raw *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

```
plot.raw(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

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

codeplot, codestrptime, codeget.pcrit, codeMR.loops,

Examples

```
## load data ##

data(fishMR)

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

plot.raw(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

Examples

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

## subset data to appropriate region ##

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

## calculate total RSS for different breakpoints ##

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

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

Index