Package 'FREddyPro'

September 11, 2016

Type Package

Title Post-Processing EddyPro Full Output File

Version 1.0

Date 2016-09-10

Author Georgios Xenakis

Maintainer Georgios Xenakis <georgios.xenakis@forestry.gsi.gov.uk>

Depends R (>= 3.3.1)

Description Despike, ustar filtering, plotting, footprint modelling and general post-processing of a standard EddyPro full output file (LI-COR Inc 2011-2015 <https://www.licor.com/env/products/eddy_covariance/eddypro.html>).

License GPL-3

Imports lubridate, ggplot2, raster, dismo, sp, RColorBrewer

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-11 19:14:19

R topics documented:

`FREddyPro-package` *Post-Processing EddyPro Full Output File*

Description

Despike, ustar filtering, plotting, footprint modelling and general post-processing of a standard EddyPro full output file (LI-COR Inc 2011-2015 <https://www.licor.com/env/products/eddy_covariance/eddypro.html>).

Author(s)

Georgios Xenakis

Maintainer: Georgios Xenakis <georgios.xenakis@forestry.gsi.gov.uk>

Examples

```
library(FREddyPro)
```

addSiteName *Create an extra column in a data frame with the name of the site*

Description

Adds an extra column with the given string for each row in the data frame

Usage

```
addSiteName(data, name)
```

Arguments

data	The data frame
name	A string to write in the extra column, usually the name of the site e.g., *"Harwood"*

Details

The function is useful if you need an identifier when combine data with different characteristics

Value

Returns a column with character elements

Author(s)

Georgios Xenakis

Examples

```
##Load the data
data(fluxes)

## Clean data
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Add the name of the site
fluxes=addSiteName(fluxes,"Harwood")
```

`Average` *Average flux footprint*

Description

Calculate the average flux footprint from an array of half hourly turbulance data using Kormann and Meixner 2001 model.

Usage

```
Average(fetch = 500, height = 3, grid = 200, speed, direction, uStar,
zol, sigmaV, weights = NULL)
```

Arguments

`fetch`	The upwind distance over which you want the calculation domain to extend. Note that the footprint will be normalized to that domain so footprints that extend well beyond the domain will be overrepresented in
`height`	The z-d height of the flux system
`grid`	The total calculation grid size. The default is 200 which would mean the footprint would be represented in a 200x200 array corresponding to dimensions of two times the fetch on each side of the grid.
`speed`	An N length array of wind speeds at the flux system
`direction`	An N length array of wind directions at the flux system
`uStar`	An N length array of friction velocities
`zol`	An N length array of Monin Ohbukov stability parameters
`sigmaV`	An N length array of standard deviations of the cross-stream wind component
`weights`	A dictionary of N length arrays footprint weighting values. This is an optional input

Value

Output is a list object with a 3 elements including:

Probability: the array holding the footprint probabilities Fpn: the array holding the grid point distance north of the flux system Fpe: the array holding the grid point distance east of the flux system WeightedProbability: the array holding the weighted footprint probabilities

Note

This code impliments the footprint model of Kormann and Meixner 'An Analytical Footprint Model For Non-Neutral Stratification' , Boundary-Layer Meteorology, 2001, v99, Issue 2, pp 207-224

Modifications have been added to determine the footprint for a two-dimensional area surrounding the flux tower. Calculations can be done for a single flux footprint, or an average of weighted or non-weighted footprints.

Author(s)

Original author: Robert Clement, University of Edinburgh. R adaptation: Georgios Xenakis, Forest Research

References

Kormann, R., and Meixner, F.X. (2001). An analytical footprint model for non-neutral stratification, Boundary-Layer Meteorology. 99, 207-224.

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Use data without NA values
fluxes_noNA<-fluxes[which(!is.na(fluxes$L)),]

## Displacement height
d=17.42

## Calculate z
fluxes_noNA$z=(fluxes_noNA$X.z.d..L*fluxes_noNA$L)+d

## Calculate zol
fluxes_noNA$zol=fluxes_noNA$z/fluxes_noNA$L

## Calculate average footprint
footprint=Average(fetch=500,height=33,grid=200,fluxes_noNA$wind_speed[180:183],
```

```
fluxes_noNA$wind_dir[180:183],fluxes_noNA$u.[180:183],fluxes_noNA$zol[180:183],
sqrt(fluxes_noNA$v_var[180:183]))

## You can also use the plot.footprint function of FREddyPro to plot the result
plotFootprint(footprint)
```

`buildTimestamp` *Build a timestamp*

Description

Use a day of year to build a timestamp column with the format used by EddyPro

Usage

```
buildTimestamp(data, doy)
```

Arguments

`data`	The data frame to use
`doy`	A string with the name for the day of year column

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Remove some line as an example
fluxes=fluxes[-c(832,833,834,835,840,953),]

## Create timestamp from date and time
fluxes=createTimestamp(fluxes)

## Now fill the gap timestmaps with NA's
fluxes=fillTimestamp(fluxes,flux=TRUE)
```

`Calculate` *Kormann and Meixner 2001 flux footprint model*

Description

Performs the footprint calculation for a single set of inputs

Usage

```
Calculate(fetch = 500, height = 3, grid = 200, speed, direction, uStar,
zol, sigmaV)
```

Arguments

`fetch`	The upwind distance over which you want the calculation domain to extend Note that the footprint will be normalized to that domain so footprints that extend well beyond the domain will be overrepresented in
`height`	The z-d height of the flux system
`grid`	The total calculation grid size. The default is 200 which would mean the footprint would be represented in a 200x200 array corresponding to dimensions of two times the fetch on each side of the grid.
`speed`	Single values of wind speed at the flux system
`direction`	Single values of the direction wind is coming from at the flux system
`uStar`	Single values of friction velocity
`zol`	Single values of Monin Ohbukov stability parameter
`sigmaV`	Single values of the standard deviation of the cross-stream wind component

Value

Output is a list object with a 3 elements including:

footprint: the array holding the footprint probabilities FPn: the array holding the grid point distance north of the flux system FPe the array holding the grid point distance east of the flux system

Note

This code impliments the footprint model of Kormann and Meixner 'An Analytical Footprint Model For Non-Neutral Stratification' , Boundary-Layer Meteorology, 2001, v99, Issue 2, pp 207-224

Modifications have been added to determine the footprint for a two-dimensional area surrounding the flux tower. Calculations can be done for a single flux footprint, or an average of weighted or non-weighted footprints.

This code was written by Robert Clement, (robert.clement@ed.ac.uk)

Author(s)

Original author: Robert Clement, University of Edinburgh. R adaptation: Georgios Xenakis, Forest Research

References

Kormann, R., and Meixner, F.X. (2001). An analytical footprint model for non-neutral stratification, Boundary-Layer Meteorology. 99, 207-224.

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Use data without NA values
fluxes_noNA<-fluxes[which(!is.na(fluxes$L)),]

## Displacement height
d=17.42

## Calculate z
fluxes_noNA$z=(fluxes_noNA$X.z.d..L*fluxes_noNA$L)+d

## Calculate zol
fluxes_noNA$zol=fluxes_noNA$z/fluxes_noNA$L

## Calculate the footprint
footprint=Calculate(fetch=500,height=33,grid=200,fluxes_noNA$wind_speed[853],
fluxes_noNA$wind_dir[853],fluxes_noNA$u.[853],fluxes_noNA$zol[853],
sqrt(fluxes_noNA$u_var[853]))

## You can also use the plot.footprint function of FREddyPro to plot the result
plotFootprint(footprint)
```

`calculateFootprint` *Minimum argument footprint calculation*

Description

A wrapper function for quick calculation of the average footprint for a specific period of time with minimum required arguments

Usage

```
calculateFootprint(df, displacement, stability = NULL, fetch = 500,
grid = 200, height, lowerDay = NULL, upperDay = NULL)
```

Arguments

`df`	The data frame to use
`displacement`	Displacement height in meters.
`stability`	Integer used to mask the data for stability. 1=Unstable, 2=neutral, 3=stable conditions.
`fetch`	The upwind distance over which you want the calculation domain to extend. Note that the footprint will be normalized to that domain so footprints that extend well beyond the domain will be overrepresented in
`grid`	The total calculation grid size. The default is 200 which would mean the footprint would be represented in a 200x200 array corresponding to dimensions of two times the fetch on each side of the grid.
`height`	The z-d height of the flux system
`lowerDay`	The lower limit of the period to calculate the footprint
`upperDay`	The upper limit of the period to calculate the footprint

Value

Returns a list object of a footprint similar to that of the function Average

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))
```

```
## Quick calculation of footprint
ftp=calculateFootprint(fluxes,17.42,stability=1,fetch=500,grid=200,height=33,
lowerDay=150,upperDay=151)

## You can also use the plot.footprint function of FREddyPro to plot the result
plotFootprint(ftp)
```

`calculatePercentFootprint` *Calculate Percent Footprint*

Description

Function which calculates the accumulated percentage footpring contribution

Usage

```
calculatePercentFootprint(ftp, percent = 99.99)
```

Arguments

`ftp`	A footprint object
`percent`	The cut-off percentage point for visual representation

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Quick calculation of footprint
ftp=calculateFootprint(fluxes,17.42,stability=1,fetch=500,grid=200,height=33,
lowerDay=150,upperDay=151)

## Calculate the footprint as an accumulated percentage and mask it for
## up to 60%
ftpP=calculatePercentFootprint(ftp,percent=60)

## You can also use the plot.footprint function of FREddyPro to plot the result
plotFootprint(ftpP)
```

cleanFluxes *Cleaning and de-spiking fluxes*

Description

The main function for cleaning and de-spiking post-processed fluxes. There are five ways to clean/de-spike gas fluxes based on 1) QC flags, 2) standard deviation of negative and positive fluxes, 2) flux distribution for each hour of the day, 4) mean AGC value of the IRGA and 5) u* filtering. There are also three ways to clean heat and water fluxes based on 1) QC flags, 2) a threshold value and on 3) standard deviation of negative and positive fluxes.

Usage

```
cleanFluxes(data, gas = "co2_flux", qcFlag = 2, sdCor = FALSE, sdTimes =
1, distCor = FALSE, agcCor = FALSE, agcVal = NULL, ustar = NULL, plot
= FALSE, write = FALSE, outputFile, thresholdList = list(H = NULL, LE =
NULL, Tau = NULL, h2o = NULL), timesList = list(H = NULL, LE = NULL, Tau
= NULL, h2o = NULL), sunset = 19, sunrise = 6, na.value = "NaN")
```

Arguments

data	The name of the data frame to clean and de-spike.
gas	A character input giving the name of the gas to clean. The default value is for CO2.
qcFlag	The QC flag to clean. Default value is 2 using the Mauder and Foken (2004) flagging system. qcFlag=NULL will bypass the flag removal system and no data will be removed. Function can be used with other flagging systems (e.g., 1-9 by Foken 2003) but an array must be given of values to remove e.g., qcFlag=c(7,8,9).
sdCor	Logical. If TRUE de-spiking based on standard deviation is applied.
sdTimes	A number showing how many times the gas flux data have to be greater than the SD to remove them as spikes. The default value is 1 SD.
distCor	Logical. If TRUE gas flux data are de-spiked using a distribution calculated for every half hour.
agcCor	Logical. If TRUE data with a mean value between 50-60% are removed.
agcVal	A character input giving the name of the mean AGC in the data.
ustar	This input can be either logical or numeric. If logical and TRUE then ustar filtering is applied using the Papale et al. 2006 method. Sunset and sunrise times should also be provided for more accurate definition of the night time data.
plot	Logical. If TRUE two multiplot outputs are produced showing all major variables before and after cleaning and de-spiking.
write	Logical. If TRUE the new clean data frame will be written into a new csv file. The output file name should also be given.

outputFile	A character input giving the name of the output file to write the clean data. String can include full path name.
thresholdList	A list giving the threshold for which if greater and lower data will be removed. The list can include sensible heat (H), latent heat (LE), momentum flux (Tau), and water flux (h2o).
timesList	A list giving how many times a flux should be greater than the SD to be a spike. The list can include sensible heat (H), latent heat (LE), momentum flux (Tau), and water flux (h2o).
sunset	The time of sunset as a real number (0-23)
sunrise	The time of sunrise as a real number (0-23)
na.value	A string or a number showing NA values in the data set

Details

By default the function removes QC flag 2, assuming data were flagged for quality using Mauden and Foken (2004) system.

To clean data using standard deviation, the mean and standard deviation of negative and positive values is calculated separately. Data greater than a user-defined times of the standard deviation are removed

For distributional cleaning, the half-hourly destribution using the complete dataset and the 5th and 95th quantiles are calculated. Values outside these quantiles are removed.

Mean AGC is calculated by EddyPro if a diagnostics value is given during processing of raw data. AGC usually should be between 50 - 60%. Values outside the threshold will be removed. Use with caution as it may remove a large number of data

U* filtering is standard procedure. For the FREddyPro function we used the procedure described by Papale et al. 2006 . Air temperature is classes and within each temperature class u* is grouped in 20 classes. For more details see Papale et al. 2006.

Author(s)

Georgios Xenakis

References

Papale D, Reichstein M, Aubinet M, Canfora E, Bernhofer C, Kutsch W, Longdoz B, Rambal S, Valentini R, Vesala T & et al. (2006) Towards a standardized processing of Net Ecosystem Exchange measured with eddy covariance technique: algorithms and uncertainty estimation. Biogeosciences, Copernicus GmbH, 3, 571 - 583

Examples

```
## Load the data
data(fluxes)

## Clean data using 3 times the SD for both gas and heat fluxes.
## Also use some thresholds for head fluxes.
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,distCor=TRUE,timesList=3,
```

```
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)),plot=TRUE)
```

`cleanSecondVar` *Clean a second variable*

Description

It cleans a second variable giving another variable so the NA correspond row by row

Usage

```
cleanSecondVar(x, y, data)
```

Arguments

`x`	A character giving the name of the first variable
`y`	A charcter giving the name of the second variable for which we want values to be removed
`data`	The data frame

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean a second variable based on the first
fluxes<-cleanSecondVar(x="H",y="qc_H",data=fluxes)
```

`cleanVar` *Clean a variable*

Description

Remove values based if they are greter or less than the given thresholds

Usage

```
cleanVar(x, data, lessThan = NULL, greaterThan = NULL)
```

Arguments

x	A character giving the name of the variable to clean
data	The data frame
lessThan	A number giving the "less than" threshold for removing values from the variable
greaterThan	A number giving the "greater than" threshold for removing values from the variable

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean a variable
fluxes<-cleanVar("H",fluxes,lessThan=-200,greaterThan=1500)
```

---

cleanVarG *Clean a variable above a threshold*

Description

Remove values based if they are greter than the given thresholds

Usage

```
cleanVarG(data, x, y = NULL, greaterThan = NULL)
```

Arguments

data	The data frame
x	A character giving the name of the variable to clean
y	A second character giving the name of the variable to clean based on the selection of x variable
greaterThan	A number giving the "greater than" threshold for removing values from the variable

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean a variable
fluxes<-cleanVarG(data=fluxes,x="H",greaterThan=1500)
fluxes<-cleanVarG(data=fluxes,x="DOY",y="H",greaterThan=175)
```

---

cleanVarL *Clean a variable below a threshold*

Description

Remove values based if they are greter or less than the given thresholds

Usage

```
cleanVarL(data, x, y = NULL, lessThan = NULL)
```

Arguments

data	The data frame
x	A character giving the name of the variable to clean
y	A second character giving the name of the variable to clean based on the selection of x variable
lessThan	A number giving the "less than" threshold for removing values from the variable

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean a variable
fluxes<-cleanVarL(data=fluxes,x="H",lessThan=-200)
fluxes<-cleanVarL(data=fluxes,x="DOY",y="H",lessThan=155)
```

`continuity` *Check timeseries continuity*

Description

Function based on the timestamp to identify if there is a discontinuity in the timeseries

Usage

```
continuity(data, timestamp = NULL, timediff = 30)
```

Arguments

`data`	The data frame
`timestamp`	A string giving the name of the timestamp.
`timediff`	

Details

If no timestamp is given then the function assumes the standard EddyPro output (which originaly does not include a timestamp) and uses internally the createTimestamp function to create one.

Value

Returns a data frame

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Remove some line as an example
fluxes=fluxes[-c(832,833,834,835,840,953),]

## Create timestamp from date and time
continuity(fluxes)
```

`createTimestamp` *Create a time stamp*

Description

It creates a new timestam colume using the date and time output of EddyPro. It then converts the column to a POSIXct class and extracts to separate columns the year, month, day and hour. It requires the *lubridate* package.

Usage

```
createTimestamp(data, timestamp = NULL)
```

Arguments

`data`	The data frame
`timestamp`	A character giving the name of the timestamp column

Details

For the function to work properly, two columns need to exist. The data and time, each one with a specific format. The date column should be formated as **YYYY-MM-DD** and the time as **HH:MM**. The functions can be used with an existing timestamp column too and that should be formated as **YYYY-MM-DD HH:MM:SS**.

Author(s)

Georgios Xenakis

Examples

```
##Load the data
data(fluxes)

## Create the time stamp
fluxes=createTimestamp(fluxes)
```

`distClean` *Distributional cleaning*

Description

Function to clean and de-spike gas flux data based on a distribution produced for each half hour and removing values lower and higher than the 5th and 95th percentile respectively

Usage

```
distClean(var, hour, df)
```

Arguments

var	The variable to clean/de-spike
hour	The hour variable created by the createTimestamp function
df	The data frame

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean data
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Create timestamp
fluxes<-createTimestamp(fluxes)

## Find which rows have positive and which negative CO2 flux values
positive<-which(fluxes$co2_flux>0)
negative<-which(fluxes$co2_flux<0)

## Clean data for positive and negative separately
fluxes$co2_flux[positive]<-distClean(fluxes$co2_flux[positive],
                                     fluxes$hour[positive],fluxes[positive,])
fluxes$co2_flux[negative]<-distClean(fluxes$co2_flux[negative],
                                     fluxes$hour[negative],fluxes[negative,])
```

---

exportFootprintPoints *Write footprint as points*

Description

Converts a two-dimensional footprint model to a data frame with X and Y coordinates

Usage

```
exportFootprintPoints(ftp, xcoord, ycoord)
```

Arguments

ftp	The footprint model
xcoord	The x coordinate of the tower
ycoord	The y coordinate of the tower

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Quick calculation of footprint
ftp=calculateFootprint(fluxes,17.42,stability=1,fetch=500,grid=200,height=33,
lowerDay=150,upperDay=151)

## Export the footprint as points
ftp_points=exportFootprintPoints(ftp, 0, 0)
```

---

fillTimestamp *Fill timestamp gaps*

Description

Identify where there is a discontinuity in the timestamp and fill it with NA values

Usage

```
fillTimestamp(data, timestamp = "timestamp", flux = FALSE)
```

Arguments

data	The data frame to use
timestamp	A string with the timestamp column name
flux	Logical. If TRUE then the data frame is a standard EddyPro full output.

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Remove some line as an example
fluxes=fluxes[-c(832,833,834,835,840,953),]

## Create timestamp from date and time
fluxes=createTimestamp(fluxes)

## Now fill the gap timestmaps with NA's
fluxes=fillTimestamp(fluxes,flux=TRUE)
```

---

fluxes *Fluxes from Harwood forest*

Description

Standard EddyPro full output for Harwood forest, Northumberland, Great Britain. The eddy covariance tower is managed by Forest Research

Usage

```
data(fluxes)
```

Examples

```
## Load the data
data(fluxes)

## Explore the variables in the data frame
str(fluxes)
## Quickly view some variables
plot(fluxes[,c(8,10,12,14)])
```

---

harwood *CSV EddyPro output file for Harwood forest*

Description

Standard EddyPro full output for Harwood forest, Northumberland, Great Britain. The eddy covariance tower is managed by Forest Research

Usage

```
data(harwood)
```

Examples

```
## Load the data
data(harwood)

## Write the data as csv
write.table(harwood,file="harwood.csv",sep=",",quote=FALSE,row.names=FALSE)
```

---

lue_model *Light use efficiency model*

Description

A simple light use efficiency model for fitting NEE data.

Usage

```
lue_model(temp, par, vpd, swc, Topt, Tmin, Tmax, StandAge, fullCanAge,
k, lai, gamma, kappa, thetaWP, thetaFC, alpha, v, A, epsilon)
```

Arguments

temp	Air temperature (K)
par	Photosynthetic active radiation (umol/m2/s)
vpd	Vapour pressure deficit (Pa)
swc	Soil water content (unitless)
Topt	Optimal temperature for growth (oC)
Tmin	Minimum temperature for growth (oC)
Tmax	Maximum temperature for growth (oC)
StandAge	Age of the stand (years)
fullCanAge	Age at full canopy coluser (years)
k	Light extinction coefficient (unitless)
lai	Leaf area index (unitless)
gamma	Empirical parameter for light modifiers
kappa	Empirical parameter for vapour pressure modifiers
thetaWP	Volumetric water content at wilting point
thetaFC	Volumetric water content at field capacity
alpha	Empirical parameter for soil water
v	Empirical parameter for soil water
A	Empirical parameters for ecosystem respiration
epsilon	Empirical potential light use efficiency

Details

The model is based on the simple LUE model by Makela et al. 2008 with some small modification including functions for light interception, temperature modifier, and ecosystem respiration. Light interception and temperature modifiers are thosed used in the popular ecophysiological model 3-PG by Landsberg and Sands 1997. Light interception is based on Beer's law using a light extinction coefficient, leaf area index and canopy cover. Canopy cover is calculated from current stand age, and the age at full canopy cover. Ecosystem respiration is estimated using Arrhenius type function (Lloyd and Taylor, 1994) dependent on temperature.

Value

The output of the function is a list with three elements including gross primary production (GPP), ecosystem respiration (Re) and net ecosystem exchange (NEE).

Author(s)

Georgios Xenakis

References

Landsberg, J.J. and Waring, R.H. (1997). A generalised model of forest productivity using simplified concepts of radiationuse efficiency, carbon balance and partitioning. Forest Ecology and Management. 95, 209-228

Lloyd, J. and Taylor, A. (1994). On the temperature dependence of soil respiration. Function Ecology. 8, 315-323.

Makela, A., Pulkkinen, M., Kolari, P., Lagergren, F., Berbigier, P., Lindroth, A., Loustau, D., Nikinmaa, E., Vesala, T. and Hari, P. (2008). Developing an empirical model of stand GPP with the LUE approach: analysis of eddy covariance data at five ontrasting conifer sites in Europe. Global Change Biology. 14, 92-108.

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(fluxes)

## Clean the fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Input
PPFD=2500
swc=0.35

## Parameters
k = 0.5
Topt = 12
Tmin = -2
```

```
Tmax = 35
lai = 6
StandAge = 41
fullCanAge = 15

gamma = 0.0003
kappa = -0.0006
alpha = 0.3
v = 0.5
thetaWP = 0.4
thetaFC = 0.7
A = 200
epsilon = 0.0164 ## umol C / umol APAR

model<-lue_model( fluxes$air_temperature, 2500, fluxes$VPD, swc=0.35,
Topt, Tmin, Tmax, StandAge, fullCanAge, k, lai, gamma, kappa, thetaWP,
thetaFC, alpha, v, A, epsilon )

par(mfrow=c(3,1))
plot(model$NEE,type='l')
plot(model$GPP,type='l')
plot(model$Re,type='l')
```

`midpoints` *Mid-point of a class*

Description

Calculate the mid-point of a class produced by the cut() command

Usage

```
midpoints(x, dp = 2)
```

Arguments

x	The classes variable
dp	Integer giving the decimal digits of the output

Examples

```
## Load the data
data(fluxes)

## Split the wind data into 20 classes
fluxes$wind_class<-cut(fluxes$wind_dir,breaks=seq(0,365,20))

## Find the midpoint of each wind class
fluxes$wind_mid_points<-midpoints(fluxes$wind_class)
```

percentNA *Percent of NA values*

Description

Gives a percent of NA values for a specific column in a data frame

Usage

```
percentNA(var)
```

Arguments

var	The variable to calculate the percentage of NAs

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Calculate percentage of NAs in the CO2 flux before cleaning data
percentNA(fluxes$co2_flux)

## Clean data
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Calculate percentage of NAs after cleaning
percentNA(fluxes$co2_flux)
```

percentQC *Percent of each QC*

Description

Gives the percentage of a each QC flag for a variable and writes the result either on the standard output or a text file.

Usage

```
  percentQC(data, var = "co2_flux", qc_var = "qc_co2_flux",
write = FALSE, dir = "./")
```

Arguments

data	A data frame
var	A string giving the name of the variable.
qc_var	A string giving the name of the QC variable
write	Logical. If true the output is written on a text file rather the standard output. The name of the output file is standard and is *percent_qc_flags.txt*.
dir	The directory to write the output file. If none given then the working directory will be used.

Details

By default the function will give the percentage of each QC flag for CO2 flux. Other variables which have QC can be defined using the var option e.g., sensible heat, latent heat and momentum flux. The variable should have a QC flag for the function to work. If there is no QC flag associated with the variable, the result will be zero.

Value

Text with output percentages

Note

Variable needs to have an associate QC flag. Function currently works for the three flag system i.e., 0, 1, 2.

Author(s)

Georgios Xenakis

Examples

```
##Load the data
data(fluxes)

## Find the percent of each QC flag for CO2 before cleaning
percentQC(fluxes)

## Do the same for sensible heat
percentQC(fluxes,var="H")

## Clean data
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Find the percent of each QC flag for CO2 after cleaning
percentQC(fluxes)

## Do the same for latent heat
percentQC(fluxes,var="H")
```

plotDaily *Plot average daily patterns*

Description

Calculating and plotting the daily pattern of any variable with the option to use mean or both mean and median. There is also an option to plot quantiles.

Usage

```
plotDaily(x, day.x, y = NULL, day.y = NULL, median = FALSE,
quantiles = FALSE, probs = c(0.05, 0.95), legend = FALSE, legendSide =
NULL, legendText = NULL, type = "l", lty = c(1, 4), col = c(1, 2),
cex.legend = 1, horiz.legend = TRUE, ylab = "Mean and median", xlab =
"Day", ...)
```

Arguments

x	The variable to plot
day.x	The day variable associated with the X variable
y	A second optional variable to plot.
day.y	The day variable associated with the Y variable, also optional.
median	Logical. If TRUE the median is also plotted with the mean.
quantiles	Logical. If TRUE the quantiles will be plotted as a colour band
probs	An array of the quantiles to use. The default is 0.05 and 0.95 (5th and 95th).
legend	Logical. If TRUE a legend will be plotted.
legendSide	The side of the plot to place the legend.
legendText	The text to use for legend.
type	The type of plot. The default is a line plot.
lty	The type of line when a line plot is used.
col	The colour to use for the lines or points. It can be a single values or an array.
cex.legend	The size of the legend labels.
horiz.legend	A logical parameter. If TRUE the legend is will be placed horizontally.
ylab	The y-axis label.
xlab	The x-axis label.
...	Further graphical options.

Details

The function first aggregates the given variable daily and plots by default the mean. The median can also be plotted with the mean. There is an optional graphical output for quantiles. By default is the 5th and 95th but other quantiles can be used. There are two separate daily inputs to the functions to allow plotting variables from different data frames.

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean and despike the fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,distCor=TRUE,timesList=3,
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Plot the daily patterns

plotDaily(x=fluxes$co2_flux,day.x=fluxes$yday,type='o',
median=TRUE,ylim=c(-30,20),lty=c(1,2),col=c(1,2),quantiles=TRUE,
legend=TRUE,legendSide='topleft',
lwd=2,ylab="Fc (umol m"^2~" s"^-1~")",main="Harwood forest",cex.legend=0.9,
horiz.legend=FALSE)
abline(h=0,lty=2)
```

---

plotDiurnal *Plot average diurnal patterns*

Description

Calculating and plotting the diurnal pattern of any variable with the option to use mean or both mean and median. There is also an option to plot quantiles.

Usage

```
plotDiurnal(data, xVar, hourX, dataY = NULL, yVar = NULL, hourY = NULL,
median = FALSE, quantiles = FALSE, probs = c(0.05, 0.95),
stdev = FALSE, sterr = FALSE, legend = FALSE, legendSide = NULL,
legendText = NULL, type = "l", lty = c(1, 4), col = c(1, 2),
cex.legend = 1, horiz.legend = TRUE, ylab = "Mean", xlab = "Hour", ...)
```

Arguments

data	The data frame to plot
xVar	The variable to plot
hourX	The hour variable associated with the X variable
dataY	A second data frame to plot (optional)
yVar	A second optional variable to plot.
hourY	The hour variable associated with the Y variable, also optional.
median	Logical. If TRUE the median is also plotted with the mean.

quantiles	Logical. If TRUE the quantiles will be plotted as a colour band
probs	An array of the quantiles to use. The default is 0.05 and 0.95 (5th and 95th).
stdev	Logical. If True the standard deviation is plotted as a colour band
sterr	Logical. If True the standard error is plotted as a colour band
legend	Logical. If TRUE a legend will be plotted.
legendSide	The side of the plot to place the legend.
legendText	The text to use for legend.
type	The type of plot. The default is a line plot.
lty	The type of line when a line plot is used.
col	The colour to use for the lines or points. It can be a single values or an array.
cex.legend	The size of the legend labels.
horiz.legend	A logical parameter. If TRUE the legend is will be placed horizontally.
ylab	The y-axis label.
xlab	The x-axis label.
...	Further graphical options.

Details

The function first aggregates the given variable to half hourly and plots by default the mean. The median can also be plotted with the mean. There is an optional graphical output for quantiles. By default is the 5th and 95th but other quantiles can be used. There are two separate hourly inputs to the functions to allow plotting variables from different data frames.

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean and despike the fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,distCor=TRUE,timesList=3,
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Create timestamp
fluxes=createTimestamp(fluxes)

## Plot the diurnal patterns
plotDiurnal(fluxes, xVar="H", hourX="hour",yVar="LE",hourY="hour",
median=TRUE,stdev=TRUE,ylim=c(-100,400),lty=c(1,2,3,4),col=c(1,2,3,4),
legend=TRUE,legendSide='topleft',legendText=c("sensible heat","latent heat"),
lwd=2,ylab="Heat (W/m"^2~")",main="Harwood forest",cex.legend=0.9,
horiz.legend=FALSE)

abline(h=0,lty=2)
```

plotFingerprint *Plot a fingerprint graph*

Description

Plots a variable with day of year on the y-axis and the hour of the day on the x-axis. It produces the commonly used fingerprint plot.

Usage

```
plotFingerprint(var, doy, hour, step = 2, xlab = "Hour",
ylab="Day or Year", ...)
```

Arguments

var	Variable to be plotted.
doy	The day of the year variable.
hour	The hour of the day variable.
step	The length between y-axis tics. Defaule is 2 days.
xlab	The label of the x-axis
ylab	The label of the y-axis
...	Further graphical parameters for the filled.contour() plot.

Author(s)

Georgios Xenakis

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(fluxes)

## Clean the fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Plot the fingerprint plot
plotFingerprint(fluxes$co2_flux,fluxes$yday,fluxes$hour,step=1,main='CO'[2]~' flux',
key.title=title(main="umol m"^2~"s"^-1~"",adj=0.2))
```

plotFootprint *Plot a tower footprint*

Description

Plots the output of a Kormann and Meixner footprint model.

Usage

```
plotFootprint(footprint, main.title = NULL, key.title =
NULL, color.palette = NULL, ...)
```

Arguments

footprint	The footprint output produced by either *Calculate* or *Average* function
main.title	The title of the plot. Default is "Footprint"
key.title	The title over the key bar. Default is "Probability"
color.palette	The color palette to use.
...	Further graphical arguments for the filled.contour() function

Author(s)

Georgios Xenakis

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load data
data(fluxes)

## Clean and de-spike the fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Use only non-NA data
fluxes_noNA<-fluxes[which(!is.na(fluxes$L)),]

## Displacement height
d=17.42 ## Displacement height

## Calculate input variables for the footprint model
fluxes_noNA$z=(fluxes_noNA$X.z.d..L*fluxes_noNA$L)+d
fluxes_noNA$zol=fluxes_noNA$z/fluxes_noNA$L

## Calculate the footprint for day 853
footprint=Calculate(fetch=500,height=33,grid=200,fluxes_noNA$wind_speed[853],
```

```
fluxes_noNA$wind_dir[853],fluxes_noNA$u.[853],fluxes_noNA$zol[853],
sqrt(fluxes_noNA$u_var[853]))

## Plot the footprint
plotFootprint(footprint)
```

---

plotMonthly *Plot monthly averages*

Description

A function to plot diurnal patterns for each individual month.

Usage

```
plotMonthly(data, var, legend = FALSE, legendSide = NULL, type = "o",
col = 1, lty = 1, yaxt.out = NULL, yaxt.in = NULL, xaxt.out = NULL,
xaxt.in = NULL, axis1.in = FALSE, at1.in = NULL, axis2.in = FALSE,
at2.in = NULL, axis1.out = FALSE, at1.out = NULL, axis2.out = FALSE,
at2.out = NULL, ...)
```

Arguments

data	The data frame.
var	The variable to plot. The default is CO2 flux.
legend	Logical. If TRUE a legend will be plotted for each subplot.
legendSide	The side at which to plot the legend.
type	The type of plot. The default is a point-line.
col	The color of the line/points.
lty	The type of line.
yaxt.out	Drawing the outside y ticks and labels. If 'n' then they are not drawn.
yaxt.in	Drawing the inside y ticks and labels. If 'n' then they are not drawn.
xaxt.out	Drawing the outside x ticks and labels. If 'n' then they are not drawn.
xaxt.in	Drawing the inside x ticks and labels. If 'n' then they are not drawn.
axis1.in	Drawing the inside x ticks and labels. If 'n' then they are not drawn.
at1.in	Variable to show where to draw the new ticks for the bottom axis of the inside plots.
axis2.in	Logical. If TRUE the left axis will be drawn for the inside plots.
at2.in	Variable to show where to draw the new ticks for the left axis of the outside plots.
axis1.out	Logical. If TRUE the bottom axis will be drawn for the outside plots.
at1.out	Variable to show where to draw the new ticks for the bottom axis of the outside plots.

axis2.out	Logical. If TRUE the left axis will be drawn for the outside plots.
at2.out	Variable to show where to draw the new ticks for the left axis of the outside plots.
...	Further graphical parameters.

Author(s)

Georgios Xenakis

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(fluxes)

## Clean the fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,distCor=TRUE,timesList=3,
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Plot monthly
plotMonthly(fluxes,"co2_flux",legend=TRUE,legendSide='bottomleft',type='o',
lty=2,ylim=c(-50,20),col=1,yaxt.in='n',yaxt.out='n',xaxt.in='n',xaxt.out='n',
at2.out=seq(-50,10,10),axis2.out=TRUE,at1.out=seq(0,23,2),axis1.out=TRUE,
at1.in=seq(0,23,2),axis1.in=TRUE)

## Put some text as title and axis labels
mtext(side=3,"Harwood forest",outer=TRUE,line=2,cex=1.2)
mtext(side=2,"Fc (umol m"^-2~" s"^-1~")",outer=TRUE,line=2,cex=1.2)
mtext(side=1,"Hour",outer=TRUE,line=3,cex=1.2)
```

---

plotQC *Plot using QC flags*

Description

Plot fluxes using different colour and points for each QC.

Usage

```
plotQC(data, var, qc_var=NULL, xvar, lines = FALSE, legendSide = NULL,
ylab = var, xlab = "DOY", col = c(1, 2, 3, 4, 5, 6, 7, 8, 9),
pch = c(16, 17, 18, 19, 20, 21, 22, 23, 24), ...)
```

Arguments

data	The EddyPro full output data frame
var	Character. The name of the variable to plot. Default variable is CO2 flux.
qc_var	Character. The name of the qc variable to plot.
xvar	Character. The name of the X variable to plot. Default variable is DOY.
lines	Logical. If TRUE a line will be drawn between each point
legendSide	Side where to place the legend of the plot. Options include 'topright', 'topleft', 'bottomright', 'bottomleft', 'center', 'bottom', 'left', 'right', 'top'. For more details see *legend* help file
ylab	y axis label
xlab	x axis label
col	Colour of the points. This can either be a single value or a vector for each of the flag
pch	Integer or single character. The type of points
...	Any extra plotting arguments

Note

The variable should have a QC flag. Currently the function works on the three flag system i.e., 0, 1 and 2.

Author(s)

Georgios Xenakis

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Plot CO2 fluxes based on the QC flag
plotQC(fluxes,"co2_flux",legendSide='bottomleft',xlab="Day of year",
        lines=TRUE,lty=2,ylab="F"[c]~" (umol/m"^2~"/s)",col=c(1,1,1),pch=c(1,2,8),
        xlim=c(150,155),xaxt='n',main='QC flags')

## Draw a new X-axis
axis(1,at=seq(150,155,0.5),labels=TRUE)

## A horizontal line
abline(h=0,lty=2)
```

plotSpectra *Plot spectra*

Description

Function to plot the ensemble spectra output from EddyPro.

Usage

```
plotSpectra(data, gas, xlab = "Frequency [Hz]", ylab = "Spectrum", avgT
= FALSE, predicted = FALSE, type = c("o", "o", "o"), col = c(1, 2, 4),
pch=c(1,1,1), na.value = "NaN", ...)
```

Arguments

data	The data frame.
gas	A character giving the name of the gas which spectra will be plotted.
xlab	The label for the x axis.
ylab	The label for the y axis.
avgT	Logical. If TRUE the avgT will be plotted.
predicted	Logical. If TRUE the predicted spectra will be plotted.
type	The type of plot. Default is point-line. It can be a single value or an array.
col	The color of the points/lines. It can be a single value or an array.
pch	The type of points to use. It can be a single value or an array.
na.value	The number/string to use for identifying NA values.
...	Further graphical parameters.

Author(s)

Georgios Xenakis

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(spectra)

## Plot the spectra
plotSpectra(spectra,"co2",predict=TRUE,avgT=TRUE,type=c('p','l','l'),pch=c(16,-1,-1))
```

plotTimeseries *Plot a time series plot*

Description

A function to plot some of the most commonly and important variables given in a standard EddyPro output file as a time series plot. The variables which will be plotted include 1) CO2 flux, sensible heat, latent heat, air temperature, vapour pressure deficit, and wind speed. Net radiation is optional.

Usage

```
plotTimeseries(data1, limList = NULL, data2 = NULL, step = 1,
legendText = NULL, ...)
```

Arguments

data1	The data frame to use for plotting.
limList	A list giving the ylim for the plotted variables. If a variable is missing from the list, ylim is derived from the range of the data. Not all variables are necessary to appear in the list.
data2	A second data frame to use for plotting and compare with the first. This is optional.
step	The length between tics in the x-axis. Default is 1 day
legendText	The legend to appear in each subplot. This is usually necessary when two data frames are used which might be from different sites or treatments.
...	Further graphical paramters.

Note

Net radiation is an optional output and will only be plotted if it exists in the data frame. If not, will be ignored. The name of the net radiation variable should be the same with the one used by EddyPro's biomet format i.e., Rn_1_1_1. In fact, all variables should have the original naming otherwise it will give an error.

Author(s)

Georgios Xenakis

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(fluxes)

## Clean the fluxes
```

```
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,distCor=TRUE,timesList=3,
thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Plot all major data in as a timeseries.
plotTimeseries(fluxes,limList=list(DOY=c(146,183)),step=2,type='o',
cex=0.6,pch=1,lwd=0.5,lty=2)
```

`plotWindrose` *Plot a wind rose*

Description

Create a wind rose plot showing the direction and speed of wind

Usage

```
plotWindrose(data, spd, dir, spdres = 2, dirres = 30, spdmin = 2,
spdmax = 20, spdseq = NULL, palette = "YlGnBu", countmax = NA, debug =
0, title = NULL)
```

Arguments

`data`	The data frame.
`spd`	Wind speed (m/s).
`dir`	Wind direction (degrees).
`spdres`	Wind speed resolution (m/s).
`dirres`	Wind direction resolution (degrees).
`spdmin`	Minimum wind speed (m/s).
`spdmax`	Maximum wind speed (m/s).
`spdseq`	The wind speed bins.
`palette`	Colour palette to be used.
`countmax`	Maximum for ylim.
`debug`	An integer. If greater than zero then the function will run in debug mode.
`title`	The title of the plot.

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(fluxes)

## Plot the wind rose
plotWindrose(data=fluxes,spd="wind_speed",dir="wind_dir")
```

polar2cart *Polar to cartesian*

Description

Converts polar co-ordinates to the cartesian system.

Usage

```
polar2cart(x, y, dist, bearing, as.deg = FALSE)
```

Arguments

x	The X cartesian co-ordinate of the point of origin.
y	The y cartesian co-ordinate of the point of origin.
dist	The distance (m) of each point from the point of origin.
bearing	The polar bearing of each point to convert.
as.deg	Logical. If true the polar bearing is in degrees.

Examples

```
## Close any previously open graphic devices
graphics.off()

## Load the data
data(fluxes)

## Set the point of origin
x=0;y=0

## Use wind direction, distance and distance from the tower to convert
## to cartesian co-ordinates
pol<-polar2cart(x,y,fluxes$x_90.,fluxes$wind_dir,as.deg=TRUE)

## Plot the distance from the tower
plot(pol,xlab='X (m)',ylab='Y (m)',pch=16,cex=0.6)

## Add also the tower point
points(0,0,col=2,pch=16,cex=1.2)
```

qcClean *Clean using QC flag*

Description

Function cleaning data based on QC flags.

Usage

```
qcClean(var, qcVar, qc)
```

Arguments

var	The variable to clean.
qcVar	The associated QC flags variable.
qc	An integer indicating the QC flags to remove. It can be a single value or an array of values. Default value is 2.

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Remove QC flags 1 and 2 for CO2
fluxes$co2_flux<-qcClean(fluxes$co2_flux,fluxes$qc_co2_flux,c(1,2))
```

readEddyPro *Read EddyPro full output*

Description

A function to read the standard full output produced by EddyPro.

Usage

```
readEddyPro(dataFile, na = "NaN")
```

Arguments

dataFile	A character with the name of the output CSV file. String can also include full path.
na	The value used in the file to represent NAs (e.g., -9999). The default is "NaN"

Details

The EddyPro full output file has a standardise format with three header rows. The function calls the readFile function of the package with this standard number of headers.

Value

Returns a data frame

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(harwood)

## Write the data as csv
write.table(harwood,file="harwood.csv",sep=",",quote=FALSE,row.names=FALSE)

## Now read the file as a new data frame
harwood=readEddyPro('harwood.csv')
```

---

readFile *Read a file*

Description

A wrapper function for read.table for easy reading files and ignoring a given number of header rows

Usage

```
readFile(dataFile, nSkip = 0, nSkipNames = 0, header = FALSE, reverse =
FALSE, sep = ",", na = "NaN")
```

Arguments

dataFile	A character with the name of the file to read. It can include a full length path.
nSkip	The number of rows to skip to read the data
nSkipNames	The number of rows to skip to read the column names
header	Logical, for indicating if the file has a header or not. Default is TRUE.
reverse	Logical. If true then first reads a header and then skips a number or rows to read the data. Works in conjunction with nSkip option
sep	The field separator. Default is comma delimited
na	The value used in the file to represent NAs (e.g., -9999). The default is *NaN*

Details

It read a file by skiping first a number of rows to read the data and then reads the file again skiping a second number of rows to read the row with the variable names.

Value

Returns a data frame

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(harwood)

## Write the data as csv
write.table(harwood,file="harwood.csv",sep=",",quote=FALSE,row.names=FALSE)

## Now read the file as a new data frame
harwood=readFile('harwood.csv',3,1,na='NaN')
```

---

removeDuplicates *Remove duplicates*

Description

The function looks at the timestamp of the data and identifies rows with same timestamp and removes one of the two

Usage

```
removeDuplicates(data)
```

Arguments

data	Data frame

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Remove duplicates
fluxes<-removeDuplicates(fluxes)
```

removeOutliers *Remove outliers*

Description

Function to remove statistical outliers

Usage

```
removeOutliers(x, na.rm = TRUE, ...)
```

Arguments

x	Variable to remove outliers
na.rm	Logical option for NA values. If TRUE NA value will be ignored.
...	Further arguments

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Remove outliers for evapotranspiration
fluxes$E<-removeOutliers(fluxes$E)
```

rotateMatrix *Rotate a matrix*

Description

Rotates a matrix

Usage

```
rotateMatrix(x)
```

Arguments

x	The matrix to rotate

Note

This is a function needed for internal purposes

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean fluxes
fluxes=cleanFluxes(fluxes,sdCor=TRUE,sdTimes=3,timesList=3,distCor=TRUE,
                   thresholdList=list(H=c(-100,1000),LE=c(-100,1000)))

## Quick calculation of footprint
ftp=calculateFootprint(fluxes,17.42,stability=1,fetch=500,grid=200,height=33,
lowerDay=150,upperDay=151)

## The footprint as a matrix
m=ftp[[1]]

## Rotate matrix
rot.m=rotateMatrix(m)
```

---

sdClean *Clean using standard deviation*

Description

Functions which splits the fluxes to positive and negative, estimates mean and standard deviation and removes values which are greater than a predefined times the standard deviation

Usage

```
sdClean(var, p)
```

Arguments

var	The variable to clean
p	A number giving the times of the standard deviation

Author(s)

Georgios Xenakis

Examples

```
## Load the data
data(fluxes)

## Clean CO2 fluxes using 3 times the SD for negative and positive
## values separately
fluxes$co2_flux<-sdClean(fluxes$co2_flux,3)
```

`spectra` *Ensemble spectra from semi-enclosed IRGA at Harwood forest*

Description

Standard EddyPro ensemble spectra for Harwood forest, Northumberland, Great Bridtain. The eddy covariance tower is managed by Forest Research

Usage

```
data("spectra")
```

Examples

```
## Load the data
data(spectra)

## Explore the variables in the data frame
str(spectra)
```

`ustarThreshold` *Find the u* threshold*

Description

Function implementing the methodology by Papale et al. 2006

Usage

```
ustarThreshold(data, sunset = 19, sunrise = 6)
```

Arguments

`data`	The data frame
`sunset`	Time of sunset as a real number (0-23)
`sunrise`	Time of sunrise as a real number (0-23)

Details

Air temperature is split into 7 classes and u* within each class is split into 20 classes (see Papale et al. 2006). Sunrise and sunset time is used to define nightime data

Author(s)

Georgios Xenakis

References

Papale D, Reichstein M, Aubinet M, Canfora E, Bernhofer C, Kutsch W, Longdoz B, Rambal S, Valentini R, Vesala T & et al. (2006) Towards a standardized processing of Net Ecosystem Exchange measured with eddy covariance technique: algorithms and uncertainty estimation. Biogeosciences, Copernicus GmbH, 3, 571 - 583

Examples

```
## Load the data
data(fluxes)

## Calculate U* and print on screen
ustar=ustarThreshold(fluxes,sunset=19,sunrise=6)
print(ustar)
```

Index