Package ‘Reddy’

April 15, 2024

Type Package

Title Analyzing and visualizing turbulence data

Version 0.0.0.9000

Author Laura Mack

Maintainer Laura Mack <laura.mack@geo.uio.no>

Description The package Reddy provides functions from post-processing over analyzing to plotting turbulence data, e.g., eddy-covariance measurements.

Imports pracma, MASS

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

R topics documented:

`binning` *discrete binning*

Description

discrete binning of a variable var1 based on another variable var2 (e.g., the stability parameter zeta)

Usage

```
binning(var1, var2, bins)
```

Arguments

`var1`	vector, variable that should be binned
`var2`	vector, variable used for the binning
`bins`	vector, providing the intervals of the bins of var2

Value

matrix of dimension (`length(bins)-1,4`) with columns representing mean, median, 25

Examples

```
zeta_bins=c(-10^(3:-3),10^(-3:3))
zeta_vals=rnorm(1000)
vals=runif(1000)
binned=binning(vals,zeta_vals,zeta_bins)
```

`calc_anisotropy` *Invariant analysis of Reynolds stress tensor*

Description

Invariant analysis of Reynolds stress tensor, calculation of Lumley and barycentric map coordinates and anisotropy

Usage

```
calc_anisotropy(a11, a12, a13, a22, a23, a33)
```

Arguments

`a11`	R11 element of Reynolds stress tensor: u_sd^2 (scalar or vector)
`a12`	R12 element of Reynolds stress tensor: cov(u,v) (scalar or vector)
`a13`	R13 element of Reynolds stress tensor: cov(u,w) (scalar or vector)
`a22`	R22 element of Reynolds stress tensor: v_sd^2 (scalar or vector)
`a23`	R23 element of Reynolds stress tensor: cov(v,w) (scalar or vector)
`a33`	R33 element of Reynolds stress tensor: w_sd^2 (scalar or vector)

Value

list containing xb, yb, eta, xi, all eigenvalues and eigenvectors (eta, xi are the coordinates of the Lumley triangle and xb, yb the coordinates of the barycentric map)

Examples

```
calc_anisotropy(1,0,0,1,0,1) #isotropic
calc_anisotropy(1,0,1,1,0,1) #some anisotropy
```

`calc_br` *Bowen ratio BR*

Description

Calculates Bowen ratio BR := SH/LH

Usage

```
calc_br(sh, lh)
```

Arguments

`sh`	sensible heat flux [W/m^2]
`lh`	latent heat flux [W/m^2]

Value

Bowen ratio [-]

`calc_csi` *Clear Sky Index (CSI)*

Description

Calculates clear sky index

Usage

```
calc_csi(temp, lw_in, rh = NULL, e = NULL)
```

Arguments

`temp`	scalar or vector, temperature [K]
`lw_in`	scalar or vector, longwave incoming radiation [W/m^2]
`rh`	scalar or vector, relative humidity [percent]
`e`	scalar or vector, vapor pressure [Pa]

Value

CSI, clear sky index

calc_dshear *Directional Shear*

Description

Calculates a measure for directional shear alpha_uw = arctan(cov(v,w)/cov(u,w))

Usage

```
calc_dshear(cov_uw, cov_vw)
```

Arguments

`cov_uw`	covariance cov(u,w)
`cov_vw`	covariance cov(v,w)

Value

angle that describes the impact of directional shear [deg]

Examples

```
calc_dshear(-0.5,0) #no shear
calc_dshear(-0.5,-0.1)
```

calc_ef *Evaporative fraction*

Description

Calculates the evaporative fraction EF := LH/(SH+LH)

Usage

```
calc_ef(sh, lh)
```

Arguments

`sh`	sensible heat flux [W/m^2]
`lh`	latent heat flux [W/m^2]

Value

evaporative fraction [-]

`calc_flux_footprint` *Flux-Footprint Parametrization (FFP) according to Kljun et al., 2015*

Description

Calculates Flux-Footprint Parametrization (FFP) according to Kljun et al., 2015

Usage

```
calc_flux_footprint(
  zm,
  u_mean = NA,
  h,
  L,
  v_sd,
  ustar,
  z0 = NA,
  contours = seq(0.9, 0.1, -0.1),
  nres = 1000
)
```

Arguments

`zm`	measurement height [m]
`u_mean`	mean horizontal wind speed [m/s] (alternatively you can also use z0)
`h`	boundary-layer height [m]
`L`	Obukhov length [m]
`v_sd`	standard deviation of crosswind [m/s]
`ustar`	friction velocity [m/s]
`z0`	roughness length [m] (either u_mean or z0 have to be given)
`contours`	which contour lines should be calculated? default: contours=seq(0.9,0.1,-0.1)
`nres`	resolution (default is nres=1000)

`calc_gustfactor` *Gust Factor*

Description

Calculates gust factor G := ws_max/ws_mean

Usage

```
calc_gustfactor(ws_max, ws_mean)
```

Arguments

`ws_max`	wind speed [m/s]
`ws_mean`	wind speed maximum [m/s]

Value

gust factor [-]

`calc_iw` *Vertical Turbulence Intensity Iw*

Description

Calculates vertical turbulence intensity Iw := w_sd/ws_mean

Usage

```
calc_iw(w_sd, ws_mean)
```

Arguments

`w_sd`	standard deviation of vertical wind (w-wind)
`ws_mean`	horizontal wind speed

Value

vertical turbulence intensity [-]

`calc_L` *Obukhov length*

Description

Calculates Obukhov length from friction velocity, mean temperature and cov(T,w)

Usage

```
calc_L(ustar, T_mean, cov_wT)
```

Arguments

`ustar`	friction velocity (e.g., from `calc_ustar`) [m/s]
`T_mean`	mean temperature [K]
`cov_wT`	covariance cov(w,T) [m/s K]

Value

Obukhov length [m]

calc_mrd — *Multiresolution Decomposition (MRD) according to Vickers and Mahrt, 2003*

Description

Calculates multiresolution decomposition (MRD) according to Vickers and Mahrt, 2003

Usage

```
calc_mrd(var1, var2 = NULL, time_res = 0.05)
```

Arguments

`var1`	timeseries of a variable
`var2`	timeseries of another variable to calculate cospectrum of var1 and var2, optional (default is NULL)
`time_res`	time resolution of the given timeseries in seconds (e.g., 0.05 for 20 Hz)

Value

MRD in form of data frame containing the columns: index, scale, time, mean, median, q25, q75

Examples

```
series=c(1,3,2,5,1,2,1,3) #example used in Vickers and Mahrt, 2003
calc_mrd(series)
```

calc_quadrant_analysis — *Calculating Coherent Structures following Quadrant Analysis*

Description

Calculates Occurrence Fraction and Strength of the four Quadrants

Usage

```
calc_quadrant_analysis(
  xval,
  yval,
  do_normalization = TRUE,
  hole_sizes = seq(0, 10)
)
```

Arguments

`xval` values of x variable (vector)

`yval` values of y variable (vector)

`do_normalization`
should the values be normalized? i.e. (x-mean(x))/sd(x)

`hole_sizes` vector containing desired hole sizes (integers >= 0)

Value

list containing occurrence fraction and strength (calculated based on product and covariance) for all four quadrants (mathematical orientation)

Examples

```
a=rnorm(100)
b=rnorm(100)
qa_ab=calc_quadrant_analysis(a,b)
```

---

calc_satvaporpressure *Saturation vapor pressure over water*

Description

Calculates the saturation vapor pressure over water for given temperature and pressure

Usage

```
calc_satvaporpressure(temp)
```

Arguments

`temp` scalar or vector, temperature [°C]

Value

E_s, saturation vapor pressure over water [hPa]

calc_ti *Horizontal Turbulence Intensity TI*

Description

Calculates horizontal turbulence intensity TI := sqrt(u_sd^2+v_sd^2)/ws_mean

Usage

```
calc_ti(u_sd, v_sd, ws_mean)
```

Arguments

`u_sd`	standard deviation of streamwise wind (u-wind)
`v_sd`	standard deviation of crosswise wind (v-wind)
`ws_mean`	horizontal wind speed

Value

horizontal turbulence intensity [-]

calc_tke *Turbulent Kinetic Energy TKE*

Description

Calculates turbulent kinetic energy (TKE) from u_sd, v_sd and w_sd

Usage

```
calc_tke(u_sd, v_sd, w_sd)
```

Arguments

`u_sd`	standard deviation of u-wind [m/s]
`v_sd`	standard deviation of v-wind [m/s]
`w_sd`	standard deviation of w-wind [m/s]

Value

turbulent kinetic energy TKE [m^2/s^2]

calc_ustar *Friction Velocity*

Description

Calculates friction velocity from the covariances cov(u,w) and cov(v,w)

Usage

```
calc_ustar(cov_uw, cov_vw)
```

Arguments

`cov_uw`	covariance cov(u,w) [m^2/s^2]
`cov_vw`	covariance cov(v,w) [m^2/s^2]

Value

friction velocity [m/s]

calc_var *Velocity Aspect Ratio (VAR)*

Description

Calculates the velocity aspect ratio: VAR := sqrt(2)*w_sd/sqrt(u_sd^2+v_sd^2)

Usage

```
calc_var(u_sd, v_sd, w_sd)
```

Arguments

`u_sd`	standard deviation of streamwise wind (u-wind)
`v_sd`	standard deviation of crosswise wind (v-wind)
`w_sd`	standard deviation of vertical wind (w-wind)

Value

velocity aspect ratio [-]

`calc_vtke` *Turbulent Kinetic Energy Velocity Scale*

Description

Calculates the velocity scale of turbulent kinetic energy (TKE): Vtke := sqrt(TKE)

Usage

```
calc_vtke(u_sd, v_sd, w_sd)
```

Arguments

`u_sd`	standard deviation of u-wind [m/s]
`v_sd`	standard deviation of v-wind [m/s]
`w_sd`	standard deviation of w-wind [m/s]

Value

turbulent kinetic energy velocity scale [m/s]

`calc_windDirection` *Wind Direction*

Description

Calculates (horizontal) wind direction

Usage

```
calc_windDirection(u, v)
```

Arguments

`u`	u-wind [m/s]
`v`	v-wind [m/s]

Value

wind direction [deg]

calc_windSpeed2D *Horizontal Wind Speed*

Description

Calculates horizontal wind speed

Usage

```
calc_windSpeed2D(u, v)
```

Arguments

u	u-wind [m/s]
v	v-wind [m/s]

Value

wind speed [m/s]

calc_windSpeed3D *Wind Speed (3D)*

Description

Calculates wind speed (3D)

Usage

```
calc_windSpeed3D(u, v, w)
```

Arguments

u	u-wind [m/s]
v	v-wind [m/s]
w	w-wind [m/s]

Value

wind speed (3D) [m/s]

`calc_zeta` *Stability Parameter*

Description

Calculates dimensionless stability parameter from Obukhov length and measurement height, i.e. zeta = z/L

Usage

```
calc_zeta(z, L)
```

Arguments

`z`	measurement height [m]
`L`	Obukhov length [m] (e.g., from `calc_L`)

Value

stability parameter [-]

`cov2cf` *Converts cov(co2,w) to CO2 flux*

Description

Converts cov(co2,w) to CO2 flux

Usage

```
cov2cf(cov_co2w, rho = NULL)
```

Arguments

`cov_co2w`	covariance cov(co2,w) [m/s]
`rho`	density of air [kg/m^3] (optional)

Value

latent heat flux [W/m^2]

cov2lh — *Converts cov(w,q) to latent heat flux LH*

Description

Converts cov(w,q) to latent heat flux LH

Usage

```
cov2lh(cov_wq, rho = NULL)
```

Arguments

`cov_wq`	covariance cov(w,q) [m/s]
`rho`	density of air [kg/m^3] (optional)

Value

latent heat flux [W/m^2]

cov2sh — *Converts cov(w,T) to sensible heat flux SH*

Description

Converts cov(T,w) to sensible heat flux SH

Usage

```
cov2sh(cov_wT, rho = NULL)
```

Arguments

`cov_wT`	covariance cov(w,T) [K m/s]
`rho`	density of air [kg/m^3] (optional)

Value

sensible heat flux [W/m^2]

`despiking` *Despiking*

Description

Three despiking method based on 1) pre-defined thresholds, 2) median deviation (mad) test and 3) skewness and kurtosis

Usage

```
despiking(
  series,
  thresholds = c(NA, NA),
  mad_factor = 10,
  threshold_skewness = 2,
  threshold_kurtosis = 8
)
```

Arguments

`series`	timeseries that shall be despiked
`thresholds`	vector with two elements representing lower and upper bounds for despiking (pre-defined thresholds), 'NA' means that the respective bound is not used
`mad_factor`	factor for the mad test, default 'mad_factor = 10'
`threshold_skewness`	threshold for skewness test, default 'threshold_skewness = 2'
`threshold_kurtosis`	threshold for kurtosis test, default 'threshold_kurtosis = 8'

Value

despiked timeseries

Examples

```
set.seed(5)
ts1=rnorm(100)
despiking(ts1,thresholds=c(-1,1))

ts2=rexp(1000)
despiking(ts2)
```

flag_distortion — *Flow Distortion Flag and Wind Constancy Ratio*

Description

Flow Distortion Flag according to Mauder et al., 2013: Wind coming from (pre-defined) directions blocked by the measurement device is flaged with 2 (for wind speeds greater than 0.1 assuming that during calm wind the wind direction is not well-defined). The wind constancy ratio is calculated to quantify the variability of horizontal wind direction according to Mahrt, 1999.

Usage

```
flag_distortion(u, v, dir_blocked = c(30, 60), threshold_cr = 0.9)
```

Arguments

`u`	u-wind (levelled sonic)
`v`	v-wind (levelled sonic)
`dir_blocked`	vector containing the lower and upper bound of the blocked wind sector in degrees (e.g., `dir_blocked = c(30,60)`)
`threshold_cr`	threshold for constancy ratio (default `threshold_cr = 0.9`, may be adapted to used data set)

flag_most — *Integral Turbulence Characteristics Flag*

Description

Integral Turbulence Characteristics Flag: Tests the consistency with Monin-Obukhov similarity theory using the scaling functions from Panofsky and Dutton, 1984

Usage

```
flag_most(sigma_w, ustar, zeta)
```

Arguments

`sigma_w`	standard deviation of vertical velocity
`ustar`	friction velocity
`zeta`	stability parameter zeta = z/L

`flag_stationarity` *Stationarity Flag*

Description

Stationarity Flag according to Foken and Wichura, 1996 based on the assumption that the covariance of two variables ('var1' and 'var2', one usually representing vertical velocity) calculated for blocks (of length nsub) do not differ to much from the total covariance

Usage

```
flag_stationarity(var1, var2, nsub = 3000)
```

Arguments

`var1`	variable 1
`var2`	variable 2 (same length as `var1`, usually either `var1` or `var2` represent vertical velocity)
`nsub`	number of elements used for subsampling (`nsub < length(var1)`)

Examples

```
set.seed(5)
ts1=rnorm(30)
ts2=rnorm(30)
flag_stationarity(ts1,ts2,nsub=6)
```

`flag_w` *Vertical Velocity Flag*

Description

Vertical Velocity Flag according to Mauder et al., 2013: After rotation the vertical velocity should vanish, this flag flags high remaining vertical velocities.

Usage

```
flag_w(w)
```

Arguments

`w`	vertical velocity

plot_barycentric_map

Plot in barycentric map

Description

Plots (xb, yb) from invariant analysis of Reynolds stress tensor (calc_anisotropy) in barycentric map

Usage

```
plot_barycentric_map(xb, yb, contours = c(5, 10, 20))
```

Arguments

`xb`	xb coordinate (e.g., from `calc_anisotropy`)
`yb`	yb coordinate (e.g., from `calc_anisotropy`)
`contours`	vector containing levels of contour lines for 2d kernel densoty estimation, default: `contours=c(5,10,20)`

Value

plots (xb, yb) in barycentric map with 2d kernel density estimation (no return)

Examples

```
nm=100
example1=calc_anisotropy(rep(1,nm),rep(0,nm),runif(nm,0,1),rep(1,nm),rep(0,nm),runif(nm,1,1.
plot_barycentric_map(example1$xb,example1$yb)
```

plot_flux_footprint

Plot Flux-Footprint

Description

Plots Flux-Footprint Parametrization (FFP) according to Kljun et al., 2015

Usage

```
plot_flux_footprint(ffp, levels = c(0, 10^seq(-6, -3, 0.1)))
```

Arguments

`ffp`	an object returned from calc_flux_footprint
`levels`	levels used for filled.contour plot of footprint, default levels=c(0,10^seq(-6,-3,0.1))

Examples

```
ffp=calc_flux_footprint(zm=5,u_mean=5,h=700,L=-1.3,v_sd=1.2,ustar=0.35)
plot_flux_footprint(ffp)
```

---

plot_mrd — *Plotting Multiresolution Decomposition*

Description

Plots multiresolution decomposition (MRD)

Usage

```
plot_mrd(mrd_out, ...)
```

Arguments

`mrd_out`	an output object from `calc_mrd`
`...`	parameters passed to plot function

Value

creates a plot of MRD with logarithmic time scale (no return)

Examples

```
set.seed(5)
series=rnorm(2^10)
mrd_test=calc_mrd(c(series))
plot_mrd(mrd_test)
```

---

plot_quadrant_analysis — *Plotting Quadrant Analysis*

Description

Plots quadrant analysis

Usage

```
plot_quadrant_analysis(
  xval,
  yval,
  do_normalization = TRUE,
  hole_sizes = c(1, 2),
  contours = 10^(-3:3),
  print_fit = TRUE,
  ...
)
```

Arguments

`xval`	values of x variable (vector)
`yval`	values of y variable (vector)
`do_normalization`	should the values be normalized, i.e. (x-mean(x))/sd(x)? default: `TRUE`
`hole_sizes`	vector containing desired hole sizes (integers >= 0), default: `c(1,2)`
`contours`	vector containing levels of contour lines for 2d kernel densoty estimation, default: `contours=10^(-3:3)`
`print_fit`	should the fit summary from the linear regression be printed? default: `TRUE`
`...`	arguments passed to plot function

Examples

```
a=rnorm(100)
b=rnorm(100)
plot_quadrant_analysis(a,b)
```

`plot_seb` *Plottting of surface energy balance and calculate surface energy balance unclosure*

Description

Plottting of surface energy balance and calculate surface energy balance unclosure as residual flux and closure ratio

Usage

```
plot_seb(
  sw_in,
  sw_out,
  lw_in,
  lw_out,
```

```
  sh = NULL,
  lh = NULL,
  gh = NULL,
  time_vector = NULL,
  print_fit = TRUE,
  ...
)
```

Arguments

sw_in	incoming shortwave radiation [W/m^2] (as vector of time)
sw_out	outgoing shortwave radiation [W/m^2] (as vector of time)
lw_in	incoming longwave radiation [W/m^2] (as vector of time)
lw_out	outgoing longwave radiation [W/m^2] (as vector of time)
sh	sensible heat flux [W/m^2] (as vector of time) – if measured
lh	latent heat flux [W/m^2] (as vector of time) – if measured
gh	ground heat flux [W/m^2] (as vector of time) – if measured
time_vector	times used as x-axis labels (optional)
print_fit	should the fit summary be printed? default: TRUE
...	optional plot parameters

Value

no return

---

ppt2rho *Unit conversion of "parts-per" to density (for closed-path gas analyzer)*

Description

Unit conversion of "parts-per" to density (for closed-path gas analyzer)

Usage

```
ppt2rho(ppt, T_mean = 288.15, pres = 101325, e = 0, gas = "H2O")
```

Arguments

ppt	measurement in parts per thousand [ppt]
T_mean	temperature [K]
pres	pressure [Pa]
e	water vapor pressure [Pa]
gas	which gas? can be either "H2O", "CO2", "CH4" (if CO2/CH4 is selected, make sure that it's still in ppt and not ppm as usual)

Value

density of the gas [kg/m^3]

`rotate_double` *Double rotation*

Description

Double rotation

Usage

```
rotate_double(u, v, w)
```

Arguments

`u`	u-wind (levelled sonic)
`v`	v-wind (levelled sonic)
`w`	w-wind (levelled sonic)

Value

list containing the wind in a natural coordinate system (streamwise, crosswise, vertical) and the two rotation angles theta and phi

Examples

```
wind_rotated=rotate_double(4,3,1) #double rotation can be applied instantenously
```

`rotate_planar` *Planar fit rotation*

Description

Planar fit rotation

Usage

```
rotate_planar(u, v, w, bias = c(0, 0, 0))
```

Arguments

`u`	u-wind (levelled sonic)
`v`	v-wind (levelled sonic)
`w`	w-wind (levelled sonic)
`bias`	a three-dimensional correction vector containing the offset of u-, v-, w-wind

Value

list containing u, v, w after rotation as well as the rotation angles alpha, beta and gamma and the fitted offset c3

Examples

```
u=rnorm(1000)
v=rnorm(1000)
w=rnorm(1000)
wind_rotated=rotate_planar(u,v,w) #for planar fit a timeseries is required
```

`shift2maxccf` *shifting two timeseries to match maximum cross-correlation*

Description

shifts two timeseries to match their maximum cross-correlation

Usage

```
shift2maxccf(var1, var2, plot = TRUE)
```

Arguments

`var1`	vector, first timeseries
`var2`	vector, second timeseries
`plot`	logical, should the cross-correlation be plotted? default `plot = TRUE`

Value

a matrix cotaining timeseries var1 and var2 as columns after shifting to the maximum cross-correlation

Examples

```
ts1=runif(10)
ts2=c(1,1,ts1)
shifted=shift2maxccf(ts1,ts2)
```

SNDcorrection — *SND and cross-wind correction of sensible heat flux*

Description

SND and cross-wind correction of sensible heat flux: converts the buoyancy flux cov(w,Ts) (based on sonic temperature Ts) to sensible heat flux

Usage

```
SNDcorrection(u, v, w, Ts, q = NULL, A = 7/8, B = 7/8)
```

Arguments

`u`	u-wind [m/s] (levelled sonic)
`v`	v-wind [m/s] (levelled sonic)
`w`	w-wind [m/s] (levelled sonic)
`Ts`	temperature [K] (sonic temperature or corrected temperature)
`q`	specific humidity [kg/kg] (if measured by the sonic, default NULL)
`A`	constant used in cross-wind correction, default `A = 7/8` for CSAT3
`B`	constant used in cross-wind correction, default `B = 7/8` for CSAT3

Value

SND correction of sensible heat flux

WPLcorrection — *WPL correction*

Description

WPL correction: density correction for trace gas fluxes (i.e., converts volume- to mass-related quantity)

Usage

```
WPLcorrection(rho_w, rho_c = NULL, w, Ts, q)
```

Arguments

`rho_w`	measured water vapor density [kg/m^3]
`rho_c`	measured trace gas density [kg/m^3] (only if WPL-correction should be applied to another flux, e.g. CO2 flux, default `NULL`)
`w`	w-wind [m/s] (levelled sonic)
`Ts`	temperature [K] (sonic temperature or corrected temperature)
`q`	specific humidity [kg/kg] (if measured, default `NULL`)

Value

WPL correction of respective flux