Package 'bReeze'

February 5, 2024

Type Package

Title Functions for Wind Resource Assessment

Version 0.4-4

Date 2024-02-05

Author Christian Graul and Carsten Poppinga

Maintainer Christian Graul <christian.graul@gmail.com>

Description A collection of functions to analyse, visualize and interpret wind data and to calculate the potential energy production of wind turbines.

License MIT + file LICENSE

URL https://github.com/chgrl/bReeze

Depends R (>= 2.14.2)

Imports graphics, grDevices, lubridate, stats, utils

Suggests RColorBrewer, RgoogleMaps, XML

NeedsCompilation no

Repository CRAN

Date/Publication 2024-02-05 20:40:09 UTC

R topics documented:

`bReeze-package` *Functions for Wind Resource Assessment*

Description

A collection of functions to analyse, visualize and interpret wind data and to calculate the potential energy production of wind turbines.

Details

Wind power is a global source of energy which attracts a large share of investments in renewables. Project sites with high wind potential do not only minimise the financial risk but increase the efficiency in carbon footprint reduction. Hence a thorough productivity analysis of a wind project is essential, regarding both economic and environmental aspects. The evaluation of the potential of a site requires a wind resource assessment, which is best based on data gained by a local measurement campaign. A methodology of processing the measured data has been established, resulting in a better understanding of the wind conditions of a site and therefore a more reliable estimation of energy production.

`bReeze` is a collection of widely used methods to analyse, visualise and interpret wind data. Wind resource analyses can subsequently be combined with characteristics of wind turbines to estimate the potential energy production on the investigated site.

Usually the data to be analysed are collected by meteorological masts (met masts) and averaged over ten minutes, but other time intervals are also processable.

`bReeze` suggests three packages, written by other developers: `RColorBrewer` (by Erich Neuwirth) provides nice colours for the graphics, `XML` (by Duncan Temple Lang) provides xml parsing for power curve import and `RgoogleMaps` (by Markus Loecher) provides simple site maps.

Try the examples below to check if `bReeze` has been correctly installed. Any question and feedback is welcome via email to <christian.graul@gmail.com> or on GitHub.

Author(s)

Christian Graul and Carsten Poppinga

Maintainer: Christian Graul <christian.graul@gmail.com>

References

The following handbook gives a detailed thematic overview and is available online:

Brower, M., Marcus, M., Taylor, M., Bernadett, D., Filippelli, M., Beaucage, P., Hale, E., Elsholz, K., Doane, J., Eberhard, M., Tensen, J., Ryan, D. (2010) Wind Resource Assessment Handbook. http://www.renewablenrgsystems.com/TechSupport/~/media/Files/PDFs/wind_resource_handbook.ashx

Further references are given under the specific functions of the package.

Examples

```
## Not run:
# load example data
data("winddata", package="bReeze")

# create two datasets
set40 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])

# format time stamp
ts <- timestamp(timestamp=winddata[,1])

# create met mast object
metmast <- mast(timestamp=ts, set40=set40, set30=set30)

# plot time series of met mast signals
plot(metmast)

# calculate frequency and mean wind speed per wind direction sector
freq <- frequency(mast=metmast, v.set=1)

# plot frequency
plot(freq)

# calculate availability of pairs of wind speed and direction
availability(mast=metmast)

# calculate monthly means of wind speed
month.stats(mast=metmast)

# calculate turbulence intensity
turbulence(mast=metmast, turb.set=1)

# calculate weibull parameters
wb <- weibull(mast=metmast, v.set=1)

# calculate total wind energy content
energy(wb=wb)

# calculate wind profile
pf <- windprofile(mast=metmast, v.set=c(1,2), dir.set=1)
```

```
# import power curve
pc <- pc("Enercon_E126_7.5MW.pow")

# calculate annual energy production
aep <- aep(profile=pf, pc=pc, hub.h=135)

# plot AEP
plot(aep)

## End(Not run)
```

---

aep *Calculation of annual energy production*

Description

Calculates annual energy production (AEP) from a site's wind profile and wind turbine characteristics.

Usage

```
aep(profile, pc, hub.h, rho=1.225, avail=1, bins=c(5,10,15,20),
  sectoral=FALSE, digits=c(3,0,0,3), print=TRUE)

## S3 method for class 'aep'
plot(x, show.total=TRUE, ...)
```

Arguments

`profile`	Wind profile object created by `profile`.
`pc`	Power curve object created by `pc`.
`hub.h`	Hub heigth of wind turbine as numeric value.
`rho`	Air density as numeric value. Default is `1.225` kg/m3 according to the International Standard Atmosphere (ISA) at sea level and 15°C.
`avail`	Availability of turbine as numeric value between `0` (not available at all) and `1` (100% available).
`bins`	Edges of wind speed bins as numeric vector or `NULL` if only total AEP is desired. Default values are `c(5, 10, 15, 20)`.
`sectoral`	If `TRUE`, wind speeds are extrapolated to hub height using the wind profiles of each direction sector. Otherwise the general profile (`"all"`) is used for extrapolation (default).
`digits`	Number of decimal places to be used for results as numeric vector. The first value is used for `wind.speed`, the second for `operation`, the third for `aep` and the fourth for `capacity` results. Default is `c(3,0,0,3)`.

print	If TRUE, results are printed directly.
x	AEP object, created by aep.
show.total	If TRUE (the default) the total AEP is added to the plot.
...	Arguments to be passed to methods. For optional graphical parameters see below.

Details

For a wind turbine the mean energy production can be expressed by

$$E = T \int_{v_{in}}^{v_{out}} f(v)\, p(v)$$

where $f(v)$ is the probability density function of the wind speed v, $p(v)$ is the power curve of the turbine and T is the production time period. Energy production starts at the turbine's cut-in wind speed v_{in} and stops at cut-out wind speed v_{out}.

Based on this fundamental expression, aep calculates the annual energy production as follows:

$$AEP = A_{turb}\, \frac{\rho}{\rho_{pc}}\, H \sum_{b=1}^{n} W(v_b)\, P(v_b)$$

where A_{turb} is the average availability of the turbine, ρ is the air density of the site and ρ_{pc} is the air density, the power curve is defined for. $W(v_b)$ is the probability of the wind speed bin v_b, estimated by the Weibull distribution and $P(v_b)$ is the power output for that wind speed bin. H is the number of operational hours – the production time period of the AEP is per definition 8760 hours.

The wind speed v_h at hub height h of the turbine is extrapolated from the measured wind speed v_{ref} at reference height h_{ref} using the Hellman exponential law (see profile):

$$v_h = v_{ref}\, \left(\frac{h}{h_{ref}}\right)^{\alpha}$$

The productive suitability of a wind turbine for a site can be evaluated by the capacity factor CF. This factor is defined as the ratio of average power output of a turbine to the theoretical maximum power output. Using the AEP as the average power output, the rated power P_{rated} of a turbine and the maximum operational hours of a year we get:

$$CF = \frac{AEP}{P_{rated}\, 8760}$$

Value

Returns a list containing:

wind.speed	Mean wind speed for each direction sector.
operation	Operational hours per year for each direction sector.
total	Total annual energy production for each direction sector.
...	Annual energy production per wind speed bin for each direction sector.
capacity	Capacity factor of the wind turbine.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- `border.leg`: Border colour(s) for the legend. One colour for each wind speed bin or a single colour – default is same as `col`.
- `bty.leg`: Type of box to be drawn around the legend. Allowed values are `"n"` (no box, the default) and `"o"`.
- `cex`: Amount by which text on the plot should be scaled relative to the default (which is `1`), as numeric. To be used for scaling of all texts at once.
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `cex.leg`: Amount by which legend text should be scaled, as numeric value.
- `circles`: Manual definition of circles to be drawn, as numeric vector of the form c(inner circle, outer circle, interval between the circles).
- `col`: Vector of colours – one colour for each wind speed bin or a single colour if `aep` only contains the total AEP.
- `col.axis`: Colour to be used for axis annotations – default is `"gray45"`.
- `col.border`: Colour to be used for sector borders – default is `NULL` (no border is drawn).
- `col.circle`: Colour to be used for circles – default is `"gray45"`.
- `col.cross`: Colour to be used for axis lines – default is `"gray45"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `col.leg`: Colour to be used for legend text – default is `"black"`.
- `fg`: If `TRUE`, sectors are plotted in foreground (on top of axis lines and circles) – default is `FALSE`.
- `lty.circle`: Line type of circles – default is `"dashed"`. See `par` for available line types.
- `lty.cross`: Line type of axis lines – default is `"solid"`. See `par` for available line types.
- `lwd.border`: Line width of the sector borders – default is `0.5`. Only used if `col.border` is set.
- `lwd.circle`: Line width of circles, as numeric value – default is `0.7`.
- `lwd.cross`: Line width of axis lines, as numeric value – default is `0.7`.
- `pos.axis`: Position of axis labels in degree, as numeric value – default is `60`.
- `sec.space`: Space between plotted sectors, as numeric value between `0` and `1` – default is `0.2`.
- `title.leg`: Alternative legend title, as string.
- `type`: If `"n"` nothing is plotted.
- `width.leg`: Widths of legend space relative to plot space, as numeric value between `0` and `1`. If `0`, the legend is omitted, default value is `0.2`.
- `x.intersp`: Horizontal interspacing factor for legend text, as numeric – default is `0.4`.
- `y.intersp`: Vertical line distance for legend text, as numeric – default is `0.4`.

Note

Sectoral extrapolation should be used carefully. Some sector's profile might be abnormal – particularly in case of short measurement periods (<= one year) and/or few samples per sector – causing biased results. Always check the profiles and set `sectoral` to `FALSE` to get more robust results.

Author(s)

Christian Graul

References

Burton, T., Sharpe, D., Jenkins, N., Bossanyi, E. (2001) *Wind Energy Handbook*. New York: Wiley

Fördergesellschaft Windenergie e.V. (2007) Technical Guidelines for Wind Turbines, Part 6: Determination of Wind Potential and Energy Yields, Revision 7

International Organisation for Standardization (1975) ISO 2533:1975 Standard Atmosphere. ISO Standard

Jangamshetti, S.H., Rau, V.G. (1999) Site Matching of Wind Turbine Generators: A Case Study. *IEEE Transaction on Energy Conversion* **14**(4), 1537–1543

See Also

`windprofile`

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set1 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set2 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set1, set2)
neubuerg <- clean(mast=neubuerg)


## calculate AEP
# calculate wind profile
neubuerg.wp <- profile(mast=neubuerg, v.set=c(1,2), dir.set=1,
  print=FALSE)

# load power curve
pw.56 <- pc("PowerWind_56_900kW.wtg")

# calculate AEP
aep(profile=neubuerg.wp, pc=pw.56, hub.h=71)

# calculate AEP with site specific air density and availability of 97
aep(profile=neubuerg.wp, pc=pw.56, hub.h=71, rho=1.195, avail=0.97)
```

```
# calculate total AEP using sectoral profiles
aep(profile=neubuerg.wp, pc=pw.56, hub.h=71, sectoral=TRUE)

# calculate AEP for 1 m/s speed bins and without binning
aep(profile=neubuerg.wp, pc=pw.56, hub.h=71, bins=seq(0,25))
aep(profile=neubuerg.wp, pc=pw.56, hub.h=71, bins=NULL)

# change number of digits and hide results
aep(profile=neubuerg.wp, pc=pw.56, hub.h=71, digits=c(1,1,1,1))
neubuerg.aep <- aep(profile=neubuerg.wp, pc=pw.56, hub.h=71, print=FALSE)
neubuerg.aep


## plot AEP objects
# default
plot(neubuerg.aep)

# omit total AEP
plot(neubuerg.aep, show.total=FALSE)

# change colours and text sizes
plot(neubuerg.aep, col=gray(5:0 / 5), cex=0.8)

# manual definition of circles
plot(neubuerg.aep, circles=c(250, 750, 250))

# plot sectors in foreground
plot(neubuerg.aep, fg=TRUE)

# change position of axis labels
plot(neubuerg.aep, pos.axis=135)

# no legend
plot(neubuerg.aep, width.leg=0)

# freaky
plot(neubuerg.aep, border.leg=heat.colors(5), bty.leg="o",
  cex.axis=0.5, cex.lab=2, cex.leg=0.5, circles=c(80, 800, 80),
  col=rainbow(5), col.axis="green", col.border="orange",
  col.circle="purple", col.cross="yellow", col.lab="pink",
  col.leg="lightblue", fg=TRUE, lwd.border=2, lwd.circle=3,
  lwd.cross=4, lty.circle="12345678", lty.cross="87654321",
  sec.space=0.6, title.leg="* WiNd SpEeD *", x.intersp=2, y.intersp=5)

## End(Not run)
```

---

availability *Calculate availability for pairs of wind speed and direction*

Description

Calculates the availability for pairs of wind speed and wind direction of a met mast.

Usage

```
availability(mast, v.set, dir.set, subset, digits=1, print=TRUE)
avail(mast, v.set, dir.set, subset, digits=1, print=TRUE)

## S3 method for class 'availability'
plot(x, set, ...)
```

Arguments

`mast`	Met mast object created by `mast`.
`v.set`	Set(s) to be used for wind speed, specified as set number(s) or set name(s). Argument is optional – if missing, all sets containing wind speed and wind direction data are used.
`dir.set`	Set(s) to be used for wind direction, specified as set number(s) or set name(s). Argument is optional – if missing, the set(s) used for wind speed data will also be used for wind direction.
`subset`	Optional start and end time stamp for a data subset, as string vector `c(start, end)`. The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
`digits`	Number of decimal places to be used for results as numeric value. Default is `1`.
`print`	If `TRUE` (the default), results are printed directly.
`x`	Availability object created by `availability`.
`set`	Number of dataset to be plotted, specified as set number or set name (optional).
`...`	Arguments to be passed to methods. For optional graphical parameters see below.

Details

The availability is the number of pairs of valid wind speed and valid wind direction data samples as a percentage of the total possible for the measurement period, i.e.:

$$Availability = \frac{N(v_{valid} \wedge dir_{valid})}{N}$$

where N is the total possible number of samples and $v_{valid} \wedge dir_{valid}$ is a pair of valid wind speed and direction data.

Causes of invalid or missing data are manifold. Typical causes are icing and defects of sensors or other measurement equipment.

Value

Returns a list of sub lists for each pair of `v.set`/`dir.set` containing:

`total`	Total availability (%), effective period (days) and total period (days) of a set/pair of wind speed and direction.
`daily`	Availability per day, i.e. number of samples per day.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- `border`: Colour of the cell borders – default value is `"black"`.
- `cex`: Numeric value, giving the amount by which text on the plot should be scaled relative to the default (which is 1).
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `col`: Vector of three text colours (or just one colour), where the first indicates 'full availability', the second 'partial availability' and the third 'not available' – default is `"black"`.
- `col.axis`: Colour to be used for axis – default is `"gray45"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `fill`: Vector of three fill colours (or just one colour), where the first indicates 'full availability', the second 'partial availability' and the third 'not available'.
- `lwd`: Line widths of the cell borders. See `par` for usage.
- `mar`: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot – default is `c(0, 0, 0, 0)`.
- `plot.names`: If `TRUE` (the default), the names of the datasets is plotted as second label of the y axis.
- `xlab`: Alternative label for the x axis.
- `ylab`: Alternative label for the y axis.

Author(s)

Christian Graul

References

Brower, M., Marcus, M., Taylor, M., Bernadett, D., Filippelli, M., Beaucage, P., Hale, E., Elsholz, K., Doane, J., Eberhard, M., Tensen, J., Ryan, D. (2010) Wind Resource Assessment Handbook. http://www.renewablenrgsystems.com/TechSupport/~/media/Files/PDFs/wind_resource_handbook.ashx

See Also

`mast`, `clean`

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10], v.std=winddata[,13])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40=set40, set30=set30,
  set20=set20)
neubuerg <- clean(mast=neubuerg)


## calculate availability
neubuerg.avail <- availability(mast=neubuerg)

# compare the total availability of 'set40' and 'set30'
neubuerg.avail$set40$total
neubuerg.avail$set30$total

# check the daily availability of 'set40'
neubuerg.avail$set40$daily

# note: availability for 'set20' is missing - its availability is NULL,
# since it does not contain wind direction data

# calculate availability for 'set20' using wind direction data from 'set40'
set20.avail <- availability(mast=neubuerg, v.set=3, dir.set=1)
# same as above
set20.avail <- availability(mast=neubuerg, v.set="set20", dir.set="set40")

# calculate availability for all sets using wind direction data from 'set40'
neubuerg.avail <- availability(mast=neubuerg, v.set=c(1,2,3), dir.set=1)

# data subsets
availability(mast=neubuerg, v.set=1,
  subset=c("2009-12-01 00:10:00", "2009-12-31 23:50:00"))
availability(mast=neubuerg, v.set=1,
  subset=c("2010-01-01 00:10:00", NA)) # just 'start' time stamp
availability(mast=neubuerg, v.set=1,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

# change number of digits and hide results
neubuerg.avail <- availability(mast=neubuerg, digits=0)
neubuerg.avail <- availability(mast=neubuerg, print=FALSE)
neubuerg.avail


## plot availability objects
plot(neubuerg.avail)  # default
```

```
plot(neubuerg.avail, set=2)  # one dataset
plot(neubuerg.avail, set="set30")  # same as above
plot(neubuerg.avail, set=c(1,2))  # several datasets

# customize plot
plot(neubuerg.avail, border="darkgray", cex.axis=0.7,
  cex.lab=0.9, col=c("darkgreen", "orange", "red4"), col.axis="blue",
  col.lab="blue", fill=c("lightgreen", "yellow", "red"), lwd=0.5,
  mar=c(1,1,1,1), plot.names=FALSE, xlab="jour", ylab="mois")

## End(Not run)
```

---

clean *Clean faulty values*

Description

Cleans faulty values of a met mast object, set or specified set of a met mast. Faulty values are replaced by NA.

Usage

```
clean(mast, set, v.avg.min=0.4, v.avg.max=50, dir.clean=TRUE,
turb.clean=4, icing=FALSE, rep=NULL, n.rep=5, ts=FALSE)
cln(mast, set, v.avg.min=0.4, v.avg.max=50, dir.clean=TRUE,
turb.clean=4, icing=FALSE, rep=NULL, n.rep=5, ts=FALSE)
```

Arguments

mast	Met mast object created by mast. To be ignored, if a single dataset shall be cleaned.
set	Set object created by set (if no mast is given) or set of met mast specified as set number or set name. To be ignored, if all datasets of mast shall be cleaned.
v.avg.min	Lower limit for wind speeds as numeric value. Default is 0.4 m/s. Set to NULL to omit minimum wind speed.
v.avg.max	Upper limit for wind speeds as numeric value. Default is 50 m/s. Set to NULL to omit maximum wind speed.
dir.clean	If TRUE (default), faulty wind direction values are excluded. Faulty values are dir.avg<0, dir.avg>360 and dir.avg, where the wind speed is lower than the v.avg.min specified.
turb.clean	Wind speed limit for turbulence intensity as numeric value. Turbulence intesity values are excluded for wind speeds lower then this limit. Default is 4 m/s.
icing	If TRUE, wind direction values are excluded, where standard deviation of wind direction is 0, assuming icing. Default is FALSE.
rep	Signal (or a vector of signals), for which repetitions shall be cleaned – default is NULL.

n.rep	Minimum number of repetitions that shall be cleaned, as integer value – default is 5. Only used if rep is not NULL.
ts	If TRUE, uneven time intervals are corrected. Can only be used with mast objects. Default is FALSE.

Details

Turbulence can be ignored for low wind speeds. Use turb.clean to clean the respective turbulence intensity values. See turbulence for more details.

If icing is detected using icing, the time stamp should be checked to exclude implausible assumptions, e.g. in summer.

Repetitions are often generated by a corrupted data stream between sensor and data logger. Some sensors also repeat their last captured value during calm conditions. Although they are unlikely for averaged time intervals, repetitions are no faulty values by default. Do only clean repetitions if you know your data. Note: the number of repetitions n.rep means n.rep+1 consecutive values in a dataset are identical. The default of 5 repetitions corresponds to one hour in case of a ten minutes interval.

Uneven time intervals might come up due to rounding errors. ts corrects them to the median time interval.

Value

Returns the input met mast or dataset object with cleaned data.

Author(s)

Christian Graul

See Also

mast, set

Examples

```
## Not run:
# load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10], v.std=winddata[,13])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40=set40, set30=set30,
  set20=set20)

# clean faulty values of a met mast
neubuerg.clean <- clean(mast=neubuerg)

# compare a subset of the original and cleaned data
```

```
neubuerg$sets$set40$data$v.avg[660:670]
neubuerg.clean$sets$set40$data$v.avg[660:670]


# clean faulty values of a dataset
set40.clean <- clean(set=set40)

# clean just one dataset of a met mast
neubuerg.clean.2 <- clean(mast=neubuerg, set=1)
neubuerg.clean.2 <- clean(mast=neubuerg, set="set40") # same as above

# change lower wind speed limit
neubuerg.clean.3 <- clean(mast=neubuerg, v.avg.min=0.3)

# compare number of samples set to 'NA', due to lowered limit
length(which(is.na(neubuerg.clean$sets$set40$data$v.avg)==TRUE))
length(which(is.na(neubuerg.clean.3$sets$set40$data$v.avg)==TRUE))


# change wind speed limit for cleaning of turbulence intensity
neubuerg.clean.4 <- clean(mast=neubuerg, turb.clean=3)

# compare number of samples set to 'NA', due to turb.clean
neubuerg.clean$sets$set40$data$turb.int[75:100]
neubuerg.clean.4$sets$set40$data$turb.int[75:100]


# check whether icing is assumed for any samples
neubuerg.clean.5 <- clean(mast=neubuerg, set=1, v.avg.min=0,
  v.avg.max=100, dir.clean=FALSE, turb.clean=0, icing=TRUE)
not.cleaned <- which(is.na(neubuerg$sets$set40$data$dir.avg)==TRUE)
cleaned <- which(is.na(neubuerg.clean.5$sets$set40$data$dir.avg)==TRUE)
length(cleaned)-length(cleaned) # no icing here

# checked time stamp to exclude implausible icing assumptions (e.g. in summer)
# (which makes no sense here, since cleaned is empty)
neubuerg.clean.5$timestamp[cleaned]


# clean repetitions
neubuerg.clean.6 <- clean(mast=neubuerg, rep=c("v.avg", "dir.avg"))
neubuerg.clean.7 <- clean(mast=neubuerg, rep="v.avg", n.rep=3)

## End(Not run)
```

`day.plot` *Plot diurnal wind speed*

Description

Plots the diurnal variation of wind speed or wind direction.

Usage

```
day.plot(mast, set, dir.set=set, signal, num.sectors=NULL, subset, ...)
day(mast, set, dir.set=set, signal, num.sectors=NULL, subset, ...)
```

Arguments

mast	Met mast object created by mast.
set	Set used for plotting, specified as set number or set name. Argument is optional – if missing, all sets containing the choosen signal are used.
dir.set	Direction set used for sectoral splitting of the data, specified as set number or set name. Argument is used for sectoral plotting only (see num.sectors).
signal	Signal to be plotted as string.
num.sectors	Number of wind direction sectors as integer value greater 1. Argument is optional – if not NULL, data is splitted into directional sectors for plotting. Sectoral plots are available only for single sets.
subset	Optional start and end time stamp for a data subset, as string vector c(start, end). The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
...	Optional graphical parameters, see below for details.

Details

plotDay reveals diurnal variations of a signal. The plot may show outliers that indicate data inconsistancy.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- bty: Type of box to be drawn around the plot region. Allowed values are "o" (the default), "l", "7", "c", "u", or "]". The resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box.
- bty.leg: Type of box to be drawn around the legend. Allowed values are "n" (no box, the default) and "o".
- cex: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- cex.axis: Amount by which axis annotations should be scaled, as numeric value.
- cex.lab: Amount by which axis labels should be scaled, as numeric value.
- cex.leg: Amount by which legend text should be scaled, as numeric value.
- col: Vector of colours, one for each set plotted.
- col.axis: Colour to be used for axis annotations – default is "black".
- col.box: Colour to be used for the box around the plot region (if bty) – default is "black".
- col.lab: Colour to be used for axis labels – default is "black".
- col.leg: Colour to be used for legend text – default is "black".

- `col.ticks`: Colours for the axis line and the tick marks respectively – default is `"black"`.
- `las`: Style of axis labels. One of `0` (always parallel to the axis, default), `1` (always horizontal), `2` (always perpendicular to the axis), `3` (always vertical).
- `lty`: Vector of line types, one for each set plotted. See `par` for available line types.
- `lwd`: Vector of line widths, one for each set plotted. See `par` for usage.
- `mar`: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot – default is `c(4.5, 5, 1.5, 1)`.
- `mgp`: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is `c(2, 0.7, 0)`.
- `pos.leg`: Position of legend – one of `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` or `"center"`. Use `NULL` to hide the legend.
- `xlab`: Alternative label for the x axis.
- `ylab`: Alternative label for the y axis.
- `ylim`: Limits of the y axis, as vector of two values.
- `x.intersp`: Horizontal interspacing factor for legend text, as numeric – default is `0.4`.
- `y.intersp`: Vertical line distance for legend text, as numeric – default is `0.8`.

Author(s)

Christian Graul

See Also

`mast`

Examples

```
## Not run:
# load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)

# plot all datasets
day.plot(mast=neubuerg, signal="v.avg")
day.plot(mast=neubuerg, signal="dir.avg")

# plot one dataset
day.plot(mast=neubuerg, set=1, signal="v.avg")
day.plot(mast=neubuerg, set="set1", signal="v.avg") # same as above
day.plot(mast=neubuerg, set=2, signal="dir.avg")

# sectoral plot
```

```
day.plot(mast=neubuerg, set=1, signal="v.avg", num.sectors=8)

# data subsets
day.plot(mast=neubuerg, signal="v.avg",
  subset=c("2009-12-01 00:00:00", "2009-12-31 23:50:00"))
day.plot(mast=neubuerg, signal="v.avg",
  subset=c("2010-01-01 00:00:00", NA)) # just 'start' time stamp
day.plot(mast=neubuerg, set=1, signal="v.avg", num.sectors=4,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

# customize plot
day.plot(mast=neubuerg, signal="v.avg",
  bty="l", cex.axis=0.8, cex.lab=0.9, cex.leg=0.7,
  col=c("darkgreen", "royalblue", "purple"), col.axis="darkgray",
  col.box="darkgray", col.lab="darkgray", col.leg="darkgray",
  col.ticks="darkgray", las=2, lty=c(2,3,4), lwd=1.5, mar=c(3, 3, 0.5, 0.5),
  mgp=c(1.5, 0.5, 0), pos.leg="topleft", xlab="hour", ylab="velocity",
  ylim=c(3.5,6), x.intersp=1, y.intersp=1)

## End(Not run)
```

`energy` *Calculation of total wind energy content*

Description

Calculates the total wind energy content per direction sector from Weibull data.

Usage

```
energy(wb, rho=1.225, bins=c(5, 10, 15, 20),
  digits=0, print=TRUE)
en(wb, rho=1.225, bins=c(5, 10, 15, 20),
  digits=0, print=TRUE)

## S3 method for class 'energy'
plot(x, show.total=TRUE, ...)
```

Arguments

`wb`	Weibull object created by `weibull`.
`rho`	Air density as numeric value. Default is `1.225` kg/m3 according to the International Standard Atmosphere (ISA) at sea level and 15°C.
`bins`	Wind speed bins as numeric vector or `NULL` if no classification is desired. Default is `c(5, 10, 15, 20)`.
`digits`	Number of decimal places to be used for results as numeric value. Default is `0`.
`print`	If `TRUE` (the default), results are printed directly.

x	Energy object created by energy.
show.total	If TRUE (the default), the total amount of wind energy per square meter is shown.
...	Arguments to be passed to methods. For optional graphical parameters see below.

Details

The total wind energy content can be perceived as the theoretic energy potential of a particular site. Therefore it is usefull for a resource assessment, independent of the wind turbine.

The power density function

$$E(v) = \frac{1}{2}\,\rho\,v^3\,f(v)$$

where ρ is the air density, v is the wind speed and $f(v)$ is its probability density function, leads to an analytical solution using wind speed bins as:

$$E(v) = \frac{1}{2}\,\rho\,H\,\sum_{b=1}^{n} v_b^3\,W(v_b)$$

where H is the number of hours of the desired period, v_b is the wind speed bin and $W(v_b)$ is the probability of that bin estimated by the Weibull distribution. The result for $H = 8760$ is the available wind energy per square meter and year.

Value

Returns a data frame containing:

total	Total wind energy content per direction sector.
...	Wind energy content per direction sector for each given wind speed bin.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- border.leg: Border colour(s) for the legend. One colour for each wind speed bin or a single colour – default is same as col.
- bty.leg: Type of box to be drawn around the legend. Allowed values are "n" (no box, the default) and "o".
- cex: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- cex.axis: Amount by which axis annotations should be scaled, as numeric value.
- cex.lab: Amount by which axis labels should be scaled, as numeric value.
- cex.leg: Amount by which legend text should be scaled, as numeric value.
- circles: Manual definition of circles to be drawn, as numeric vector of the form c(inner circle, outer circle, interval between the circles).
- col: Vector of colours – one colour for each wind speed bin or a single colour if energy only contains the total energy per direction sector.

- `col.axis`: Colour to be used for axis annotations – default is `"gray45"`.
- `col.border`: Colour to be used for sector borders – default is `NULL` (no border is drawn).
- `col.circle`: Colour to be used for circles – default is `"gray45"`.
- `col.cross`: Colour to be used for axis lines – default is `"gray45"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `col.leg`: Colour to be used for legend text – default is `"black"`.
- `fg`: If `TRUE`, sectors are plotted in foreground (on top of axis lines and circles) – default is `FALSE`.
- `lty.circle`: Line type of circles – default is `"dashed"`. See `par` for available line types.
- `lty.cross`: Line type of axis lines – default is `"solid"`. See `par` for available line types.
- `lwd.border`: Line width of the sector borders – default is `0.5`. Only used if `col.border` is set.
- `lwd.circle`: Line width of circles, as numeric value – default is `0.7`.
- `lwd.cross`: Line width of axis lines, as numeric value – default is `0.7`.
- `pos.axis`: Position of axis labels in degree, as numeric value – default is `60`.
- `sec.space`: Space between plotted sectors, as numeric value between `0` and `1` – default is `0.2`.
- `title.leg`: Alternative legend title, as string.
- `width.leg`: Widths of legend space relative to plot space, as numeric value between `0` and `1`. If `0`, the legend is omitted, default value is `0.2`.
- `x.intersp`: Horizontal interspacing factor for legend text, as numeric – default is `0.4`.
- `y.intersp`: Vertical line distance for legend text, as numeric – default is `0.4`.

Author(s)

Christian Graul

References

Fördergesellschaft Windenergie e.V. (2007) Technical Guidelines for Wind Turbines, Part 6: Determination of Wind Potential and Energy Yields, Revision 7

International Organisation for Standardization (1975) ISO 2533:1975 Standard Atmosphere. ISO Standard

Troen, I., Petersen, E.L. (1989) *European Wind Atlas*. Tønder: Laursen

See Also

`weibull`

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set1 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set2 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])
set3 <- set(height=20, v.avg=winddata[,10], v.std=winddata[,13])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set1, set2, set3)
neubuerg <- clean(mast=neubuerg)


## calculate Energy object
# calculate Weibull object
neubuerg.wb <- weibull(mast=neubuerg, v.set=1, print=FALSE)

# calculate energy
neubuerg.e <- energy(wb=neubuerg.wb)

# calculate energy for 1 m/s speed bins and without binning
energy(wb=neubuerg.wb, bins=0:25)
energy(wb=neubuerg.wb, bins=NULL)

# calculate energy with site specific air density
energy(wb=neubuerg.wb, rho=1.115, bins=NULL)

# change number of digits and hide results
energy(wb=neubuerg.wb, digits=2)
energy(wb=neubuerg.wb, print=FALSE)


## plot energy objects
plot(neubuerg.e)  # default
plot(neubuerg.e, show.total=FALSE)  # omit total amount
plot(neubuerg.e, col=gray(5:0/5.5), cex=0.8)  # change colours/text sizes
plot(neubuerg.e, circles=c(100, 500, 100))  # manual definition of circles
plot(neubuerg.e, fg=TRUE)  # plot sectors in foreground
plot(neubuerg.e, pos.axis=135)  # change axis label position
plot(neubuerg.e, width.leg=0)  # no legend

# freaky
plot(neubuerg.e, border.leg=heat.colors(5), bty.leg="o",
  cex.axis=0.5, cex.lab=2, cex.leg=0.5, circles=c(80, 800, 80),
  col=rainbow(5), col.axis="green", col.border="orange",
  col.circle="purple", col.cross="yellow", col.lab="pink",
  col.leg="lightblue", fg=TRUE, lwd.border=2, lwd.circle=3, lwd.cross=4,
  lty.circle="52168319", lty.cross="12223242", sec.space=0.6,
  title.leg="* WiNd SpEeD *", x.intersp=2, y.intersp=5)

## End(Not run)
```

`frequency` *Calculation of frequency and mean wind speed*

Description

Calculates the frequency of occurrence and mean wind speed per wind direction sector.

Usage

```
frequency(mast, v.set, dir.set, num.sectors=12,
  bins=c(5, 10, 15, 20), subset, digits=3, print=TRUE)
freq(mast, v.set, dir.set, num.sectors=12,
  bins=c(5, 10, 15, 20), subset, digits=3, print=TRUE)
```

Arguments

`mast`	Met mast object created by `mast`.
`v.set`	Set used for wind speed, specified as set number or set name (optional, if `dir.set` is given).
`dir.set`	Set used for wind direction, specified as set number or set name (optional, if `v.set` is given).
`num.sectors`	Number of wind direction sectors as integer value greater 1. Default is `12`.
`bins`	Wind speed bins as numeric vector or `NULL` if no classification is desired. Default is `c(5, 10, 15, 20)`.
`subset`	Optional start and end time stamp for a data subset, as string vector `c(start, end)`. The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
`digits`	Number of decimal places to be used for results as numeric value. Default is `3`.
`print`	If `TRUE` (the default), results are printed directly.

Details

The directional frequency of occurrence is an important factor for the design of a wind project. The influence is clear for the arrangement of turbines in a wind farm, that should be perpendicular to the most frequent wind direction. But also single turbines are affected, e.g. by fatigue tower loads in most frequent directions or in directions with highest wind speeds.

Value

Returns a data frame containing:

`wind.speed`	Mean wind speed per direction sector.
`total`	Frequency per direction sector.
`...`	Frequencies per direction sector, for each given wind speed bin.

Optional graphical parameters for plotting

The following graphical parameters can optionally be added to customize the plot:

- `border.leg`: Border colour(s) for the legend. One colour for each wind speed bin or a single colour – default is same as `col`.
- `bty.leg`: Type of box to be drawn around the legend. Allowed values are `"n"` (no box, the default) and `"o"`.
- `cex`: Amount by which text on the plot should be scaled relative to the default (which is `1`), as numeric. To be used for scaling of all texts at once.
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `cex.leg`: Amount by which legend text should be scaled, as numeric value.
- `circles`: Manual definition of circles to be drawn, as numeric vector of the form c(inner circle, outer circle, interval between the circles).
- `col`: Vector of colours – one colour for each wind speed bin or a single colour if `frequency` only contains the total frequency per direction sector.
- `col.axis`: Colour to be used for axis annotations – default is `"gray45"`.
- `col.border`: Colour to be used for sector borders – default is `NULL` (no border is drawn).
- `col.circle`: Colour to be used for circles – default is `"gray45"`.
- `col.cross`: Colour to be used for axis lines – default is `"gray45"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `col.leg`: Colour to be used for legend text – default is `"black"`.
- `fg`: If `TRUE`, sectors are plotted in foreground (on top of axis lines and circles) – default is `FALSE`.
- `lty.circle`: Line type of circles – default is `"dashed"`. See `par` for available line types.
- `lty.cross`: Line type of axis lines – default is `"solid"`. See `par` for available line types.
- `lwd.border`: Line width of the sector borders – default is `0.5`. Only used if `col.border` is set.
- `lwd.circle`: Line width of circles, as numeric value – default is `0.7`.
- `lwd.cross`: Line width of axis lines, as numeric value – default is `0.7`.
- `pos.axis`: Position of axis labels in degree, as numeric value – default is `60`.
- `sec.space`: Space between plotted sectors, as numeric value between `0` and `1` – default is `0.2`.
- `title.leg`: Alternative legend title, as string.
- `width.leg`: Widths of legend space relative to plot space, as numeric value between `0` and `1`. If `0`, the legend is omitted, default value is `0.2`.
- `x.intersp`: Horizontal interspacing factor for legend text, as numeric – default is `0.4`.
- `y.intersp`: Vertical line distance for legend text, as numeric – default is `0.4`.

Author(s)

Christian Graul

References

Brower, M., Marcus, M., Taylor, M., Bernadett, D., Filippelli, M., Beaucage, P., Hale, E., Elsholz, K., Doane, J., Eberhard, M., Tensen, J., Ryan, D. (2010) Wind Resource Assessment Handbook. http://www.renewablenrgsystems.com/TechSupport/~/media/Files/PDFs/wind_resource_handbook.ashx

See Also

mast

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)

## calculate frequency
frequency(mast=neubuerg, v.set=1)

# if only one of v.set and dir.set is given,
# the dataset is assigned to both
frequency(mast=neubuerg, v.set=1)
frequency(mast=neubuerg, dir.set=1)

# use different datasets for wind speed and direction
frequency(mast=neubuerg, v.set=3) # no direction in dataset
frequency(mast=neubuerg, v.set=3, dir.set=2)
frequency(mast=neubuerg, v.set="set3", dir.set="set2") # same as above

# change number of direction sectors
frequency(mast=neubuerg, v.set=1, num.sectors=4)
frequency(mast=neubuerg, v.set=1, num.sectors=16)

# calculate frequency for 1 m/s speed bins and without binning
frequency(mast=neubuerg, v.set=1, bins=1:25)
frequency(mast=neubuerg, v.set=1, bins=0:25) # same as above
frequency(mast=neubuerg, v.set=1, bins=NULL)

# data subsets
frequency(mast=neubuerg, v.set=1,
  subset=c("2009-12-01 00:00:00", "2009-12-31 23:50:00"))
frequency(mast=neubuerg, v.set=1,
  subset=c("2010-01-01 00:00:00", NA)) # just 'start' time stamp
frequency(mast=neubuerg, v.set=1,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

# change number of digits and hide results
```

```
frequency(mast=neubuerg, v.set=1, digits=2)
neubuerg.freq <- frequency(mast=neubuerg, v.set=1, print=FALSE)
neubuerg.freq

## plot frequency objects
plot(neubuerg.freq)  # default
plot(neubuerg.freq, col=gray(5:0 / 5.5), cex=0.8)  # change colours/text sizes
plot(neubuerg.freq, circles=c(10, 30, 10))  # manual definition of circles
plot(neubuerg.freq, fg=TRUE)  # plot sectors in foreground
plot(neubuerg.freq, pos.axis=135) # change axis label position
plot(neubuerg.freq, width.leg=0)  # no legend

# freaky
plot(neubuerg.freq, border.leg=heat.colors(5), bty.leg="o",
  cex.axis=0.5, cex.lab=2, cex.leg=0.5, circles=c(5, 30, 5),
  col=rainbow(5), col.axis="green", col.border="orange",
  col.circle="purple", col.cross="yellow", col.lab="pink",
  col.leg="lightblue", fg=TRUE, lwd.border=2, lwd.circle=3, lwd.cross=3,
  lty.circle="12345678", lty.cross="87654321", sec.space=0.6,
  title.leg="* WiNd SpEeD *", x.intersp=2, y.intersp=5)

## End(Not run)
```

`map.plot` *Plot map or satellite image*

Description

Plots a map or satellite image of the met mast location.

Usage

```
map.plot(mast, type=c("satellite", "terrain", "hybrid", "roadmap"),
  zoom, label, ...)
map(mast, type=c("satellite", "terrain", "hybrid", "roadmap"),
  zoom, label, ...)
```

Arguments

`mast`	Met mast object created by `mast`.
`type`	Type of the map as string. One of `"satellite"` (satellite image), `"terrain"` (map with terrain information), `"hybrid"` (satellite image with map overlay) or `"roadmap"` (map).
`zoom`	Zoom level as integer from 1 (small scale) up to 18 (large scale) – default is 15.
`label`	Label to be placed next to the location symbol as string. Set to `NA` to leave the label blank. If ignored, the location coordinates are used as label default.
`...`	Optional graphical parameters, see below for details.

Details

This function is based on the RgoogleMaps package by Markus Loecher, which uses the Google Static Maps API.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- cex: Numeric value, giving the amount by which location symbol should be scaled relative to the default (which is 1.5).
- col: The colour of the symbol.
- pch: Location symbol, either as integer or as single character. See points for possible values and their interpretation – default is 8.
- cex.lab: Numeric value, giving the amount by which the label should be scaled relative to the default (which is 1).
- col.lab: The colour of the label – default is same as col.
- pos.lab: Position specifier for the label. One of 1 (below symbol), 2 (left of symbol), 3 (above symbol) or 4 (right of symbol) – default is 4.

Author(s)

Christian Graul

See Also

mast

Examples

```
## Not run:
# load and prepare data
data("winddata", package="bReeze")
set1 <- set(height=40, v.avg=winddata[,2])
ts <- timestamp(winddata[,1])
neubuerg <- mast(timestamp=ts, set1, loc=c(49.8909,11.4017))

# plot satellite image
map.plot(neubuerg)

# plot terrain map
map.plot(neubuerg, type="terrain")

# change zoom level
map.plot(neubuerg, zoom=1)
map.plot(neubuerg, zoom=18)

# change symbol (and label)
map.plot(neubuerg, col="white", pch="+", cex=2)
```

```
# change label
map.plot(neubuerg, col.lab=3, cex.lab=1.5)
map.plot(neubuerg, pos.lab=1)
map.plot(neubuerg, label="Site #247 - Neubuerg")  # custom label
map.plot(neubuerg, label=NA)  # no label

## End(Not run)
```

`mast` *Creation of met mast objects*

Description

Creates met mast objects from one or more datasets (measurement heights). All datasets are sorted by height in descending order.

Usage

```
mast(timestamp, ..., loc=NULL, desc=NULL)

## S3 method for class 'mast'
plot(x, set, signal=c("v.avg", "dir.avg", "turb.int"), subset, ...)
```

Arguments

`timestamp`	Time stamp as `POSIXlt` vector, e.g. created by `timestamp`.
`...`	At least one dataset created by `set`. For plotting function: optional graphical parameters (see below).
`loc`	Lat/lon coordinates of the site in decimal degrees as vector of two numeric values (optional).
`desc`	Plain text information about the site, measurement, met mast condition, etc. as string (optional).
`x`	Met mast object, created by `mast`.
`set`	Set used for plotting specified as set number or set name. Argument is optional – if missing, all sets containing the given `signal`(s) are used.
`signal`	Signal(s) to be plotted as vector of strings giving the signal names.
`subset`	Optional start and end time stamp for a data subset, as string vector `c(start, end)`. The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".

Details

Valuable information about a met mast is usually provided by an installation protocol.

Measurements of wind speed in several heights is required for the computation of the site's wind profile. Met masts might have mounted more than one sensor of the same type at the same height. Thus, the data of a broken sensor can later be substituted by the 'backup-sensor'.

Value

Returns a met mast object, which is necessary for all data analyses. A met mast object is a list of:

`timestamp`	Time stamp of the observations.
`location`	Lat/lon coordinates of the site (optional).
`description`	Mast information (optional).
`sets`	List of one or more datasets (measurement heights) consisting of height information and the data.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- `bty`: Type of box to be drawn around the plot region. Allowed values are `"o"` (the default), `"l"`, `"7"`, `"c"`, `"u"`, or `"]"`. The resulting box resembles the corresponding upper case letter. A value of `"n"` suppresses the box.
- `bty.leg`: Type of box to be drawn around the legend. Allowed values are `"n"` (no box, the default) and `"o"`.
- `cex`: Amount by which text on the plot should be scaled relative to the default (which is `1`), as numeric. To be used for scaling of all texts at once.
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `cex.leg`: Amount by which legend text should be scaled, as numeric value.
- `col`: Vector of colours, one for each set plotted.
- `col.axis`: Colour to be used for axis annotations – default is `"black"`.
- `col.box`: Colour to be used for the box around the plot region (if `bty`) – default is `"black"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `col.leg`: Colour to be used for legend text – default is `"black"`.
- `col.ticks`: Colours for the axis line and the tick marks respectively – default is `"black"`.
- `las`: Style of axis labels. One of `0` (always parallel to the axis, default), `1` (always horizontal), `2` (always perpendicular to the axis), `3` (always vertical).
- `legend`: If `TRUE` (the default) a legend is drawn.
- `lty`: Vector of line types – one for each set plotted. See `par` for available line types.
- `mar`: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot (only for plots with one dataset) – default is `c(1, 4.5, 0, 1)`.
- `mgp`: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is `c(2.5, 0.7, 0)`.
- `ylab`: String vector of labels for the y axis.
- `x.intersp`: Horizontal interspacing factor for legend text, as numeric – default is `0.4`.

Author(s)

Christian Graul

See Also

POSIXlt, set, timestamp

Examples

```
## Not run:
## load data, prepare sets and time stamp
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10], v.std=winddata[,13])
ts <- timestamp(timestamp=winddata[,1])

## create met mast object
neubuerg <- mast(timestamp=ts, set40, set30, set20)  # default

# add location and description
neubuerg.2 <- mast(timestamp=ts, set40, set30, set20,
  loc=c(49.8909,11.4017), desc="Site #247 - Neubuerg")

# name sets
neubuerg.3 <- mast(timestamp=ts, C1.A1=set40, C2.A2=set30,
  C3=set20, loc=c(49.8909,11.4017), desc="Site #247 - Neubuerg")


## plot met mast (time series)
plot(neubuerg)  # default
plot(neubuerg, set=1, legend=FALSE)  # only one set
plot(neubuerg, set="set1", legend=FALSE)  # same as above
plot(neubuerg, signal=c("v.avg", "dir.avg"))  # change signals

# change time scale
plot(neubuerg, subset=c("2010-01-01 00:10:00", NA))
plot(neubuerg, subset=c("2009-10-11 00:10:00", "2009-10-11 23:50:00"))
plot(neubuerg, set=1, signal="dir.avg", subset=c(NA, "2009-12-27 18:30:00"))

# customize plot
plot(neubuerg, bty="n", bty.leg="o", cex.axis=1.2, cex.lab=1.4, cex.leg=1.2,
  col=c("darkblue", "red2", "darkgreen"), col.axis="darkgray",
  col.lab="darkgray", col.leg="darkgray", col.ticks="darkgray",
  las=0, lty=rep(1.5,3), mar=c(0.5,4,0,0.5), mgp=c(2,0.5,0),
  ylab=c("v [m/s]","dir [deg]","ti [-]"), x.intersp=1)

## End(Not run)
```

`month.stats` *Calculation of monthly statistics*

Description

Calculates monthly statistics.

Usage

```
month.stats(mast, set, signal="v.avg",
  fun=c("mean", "median", "min", "max", "sd"), subset,
  digits=3, print=TRUE)
ms(mast, set, signal="v.avg",
  fun=c("mean", "median", "min", "max", "sd"), subset,
  digits=3, print=TRUE)

## S3 method for class 'month.stats'
plot(x, set, ...)
```

Arguments

mast	Met mast object created by mast.
set	Set used for calculation or plotting specified as set number or set name. If missing, the calculation or plotting is carried out for all datasets that contain the specified signal.
signal	The signal to be used, as string value. Default is "v.avg".
fun	Statistical function, as string value – one of "mean" (arithmetic mean), "median", "min" (minimum), "max" (maximum) or "sd" (standard deviation).
subset	Optional start and end time stamp for a data subset, as string vector c(start, end). The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
digits	Number of decimal places to be used for results as numeric value. Default is 3.
print	If TRUE (the default), results are printed directly.
x	Monthly statistics object created by month.stats.
...	Arguments to be passed to methods. For optional graphical parameters see below.

Details

month.stats calculates statistics of valid data for each month and year in the measurement period. Usually this function is used for the calculation of average wind speeds. Means strongly depend on the measurement period and the number of samples. One important requirement for a reliable wind assessment is a measurement period covering the full seasonal cycle of variations. A typical bias is a measurement limited to winter months, which usually results in overestimated wind speeds.

Value

Returns a list of data frames (one for each dataset) containing monthly, annual and total statistics of the specified signal.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- `border`: Colour to be used for the border of the bars – default value is `"black"`. Use `border=NA` to omit borders.
- `bty`: Type of box to be drawn around the plot region. Allowed values are `"o"` (the default), `"l"`, `"7"`, `"c"`, `"u"`, or `"]"`. The resulting box resembles the corresponding upper case letter. A value of `"n"` suppresses the box.
- `bty.leg`: Type of box to be drawn around the legend. Allowed values are `"n"` (no box, the default) and `"o"`.
- `cex`: Amount by which text on the plot should be scaled relative to the default (which is `1`), as numeric. To be used for scaling of all texts at once.
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `cex.leg`: Amount by which legend text should be scaled, as numeric value.
- `col`: Vector of colours, one for each year in the measurement period.
- `col.axis`: Colour to be used for axis annotations – default is `"black"`.
- `col.box`: Colour to be used for the box around the plot region (if `bty`) – default is `"black"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `col.leg`: Colour to be used for legend text – default is `"black"`.
- `col.ticks`: Colours for the axis line and the tick marks respectively – default is `"black"`.
- `las`: Style of axis labels. One of `0` (always parallel to the axis, default), `1` (always horizontal), `2` (always perpendicular to the axis), `3` (always vertical).
- `legend`: If `TRUE` (the default) a legend is drawn.
- `mar`: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot (only for plots with one dataset) – default is `c(4, 5, 1, 1)`.
- `mgp`: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is `c(2.5, 1, 0)`.
- `plot.names`: If `TRUE` (the default), the names of the datasets is plotted as second label of the y axis.
- `pos.leg`: Position of legend – one of `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `top`, `"topright"`, `"right"` or `"center"`.
- `xlab`: Alternative label for the x axis.
- `ylab`: Alternative label for the y axis.
- `ylim`: Limits of the y axis, as vector of two values.
- `x.intersp`: Horizontal interspacing factor for legend text, as numeric – default is `0.4`.

Author(s)

Christian Graul

References

Brower, M., Marcus, M., Taylor, M., Bernadett, D., Filippelli, M., Beaucage, P., Hale, E., Elsholz, K., Doane, J., Eberhard, M., Tensen, J., Ryan, D. (2010) Wind Resource Assessment Handbook. http://www.renewablenrgsystems.com/TechSupport/~/media/Files/PDFs/wind_resource_handbook.ashx

See Also

mast

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], v.max=winddata[,3])
set30 <- set(height=30, v.avg=winddata[,6], v.max=winddata[,7])
set20 <- set(height=20, v.avg=winddata[,10])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)

## calculate monthly means
neubuerg.stats <- month.stats(mast=neubuerg)  # default
month.stats(mast=neubuerg, set=1)  # one dataset
month.stats(mast=neubuerg, set="set1")  # same as above
month.stats(mast=neubuerg, signal="v.max")  # change signal

# calculate monthly median, min, max and standard deviation
month.stats(mast=neubuerg, fun="median")
month.stats(mast=neubuerg, fun="min")
month.stats(mast=neubuerg, fun="max")
month.stats(mast=neubuerg, fun="sd")

# data subsets
month.stats(mast=neubuerg,
  subset=c("2009-12-01 00:10:00", "2009-12-31 23:50:00"))
month.stats(mast=neubuerg,
  subset=c("2010-01-01 00:10:00", NA)) # just 'start' time stamp
month.stats(mast=neubuerg,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

month.stats(mast=neubuerg, digits=2)  # change number of digits
neubuerg.ms <- month.stats(mast=neubuerg, print=FALSE)  # hide results
neubuerg.ms


## plot month stats object
plot(neubuerg.ms)  # default
plot(neubuerg.ms, set=1)  # one dataset
plot(neubuerg.ms, set="set1")  # same as above

# customize plot
```

```
plot(neubuerg.ms, border="darkgray", bty="l", cex.axis=0.7,
  cex.lab=0.9, col=c(gray(0.3), gray(0.7)), col.axis="darkgray",
  col.box="darkgray", col.lab="darkgray", col.ticks="darkgray", las=0,
  legend=FALSE, mgp=c(2, 0.7, 0), plot.names=FALSE, ylim=c(0,7),
  ylab="Wind speed [m/s]")

## End(Not run)
```

---

pc *Import power curve from file*

Description

Imports a power curve from a WAsP 'wgt' file or a WindPower program 'pow' file.

Usage

```
pc(pc, ...)

## Default S3 method:
pc(pc, rho=1.225, rated.p, desc, ...)
## S3 method for class 'read'
pc(pc, ...)
## S3 method for class 'pc'
plot(x, cp=TRUE, ct=TRUE, ...)
```

Arguments

pc	`list` or `data.frame` of power curve variables – v (wind speed in m/s), p (corresponding power output in kW), cp (power coefficient, optional), ct (thrust coefficient, optional), all as numeric vectors of same length. Or the name of, or the path to a 'wgt' or 'pow' file containing power curve data.
rho	Air density as numeric value. Default is `1.225` kg/m3 according to the International Standard Atmosphere (ISA) at sea level and 15 degrees Celsius.
rated.p	Rated power of wind turbine in kW as numeric value. If not given, the rated power is set to the maximum value of `p`.
desc	Plain text information about the wind turbine as string (optional).
x	Power curve object created by `pc`.
cp	If `TRUE` (the default), the power coefficient (cp) is added to the plot (if available).
ct	If `TRUE` (the default), the thrust coefficient (ct) is added to the plot (if available).
...	Arguments to be passed to methods. For optional graphical parameters see below.

Details

Power curve

A power curve characterizes the power production of a wind turbine and gives the amount of generated electrical power output as a function of wind speed. The theoretical power curve of a turbine is defined as:

$$\begin{array}{ll} P \propto v^3 & \text{for } v < v_{rated} \\ P = P_{rated} & \text{for } v > v_{rated} \end{array}$$

Hence the generated power is proportional to the wind speed cubed, for wind speeds lower than rated wind speed. For higher wind speeds the generated power is equal to the rated power of the turbine.

Conventionally a power curve consists of pairs of wind speed and power in 0.5 or 1 m/s wind speed bins, starting at 0 m/s or the cut-in wind speed of the turbine and ending with the cut-out wind speed, e.g. at about 25 m/s.

Coefficients

The power coefficient c_p is defined as:

$$c_p = \frac{P_t}{P}$$

where P_t is the ratio of the electrical power extracted by the wind turbine and P is the energy available in the wind stream. According to Betz's law, the theoretically achievable power coefficient is approximately 0.59. However, no wind turbine will obtain this value, due to inefficiencies and various losses of the machine.

The thrust coefficient is a turbine specific characteristic and used for the modelling of wake effects. Therefore it is an important parameter for wind farm configuration.

`bReeze` provides several power curves of common manufacturers, that can be read from the package directory. See examples for usage.

Available 'wtg' files

Nordex_N80_2.5MW	Nordex_N90_2.5MW_HS	Nordex_N90_2.5MW_LS
Nordex_N100_2.5MW	PowerWind_56_900kW	PowerWind_90_2.5MW
Vestas_V52_850kW	Vestas_V60_850kW	Vestas_V80_2.0MW_os
Vestas_V80_2.0MW	Vestas_V82_1650kW	Vestas_V90_1.8MW
Vestas_V90_2.0MW	Vestas_V90_3.0MW	Vestas_V100_1.8MW_50Hz
Vestas_V100_1.8MW_60Hz	Vestas_V112_3.0MW	

Available 'pow' files

Bonus_82.4m_2.3MW	Bonus_MKIV_600kW	Clipper_LibertyC89_2.5MW
Clipper_LibertyC93_2.5MW	Clipper_LibertyC96_2.5MW	Clipper_LibertyC100_2.5MW
Enercon_E33_330kW	Enercon_E40_500kW	Enercon_E44_900kW
Enercon_E48_800kW	Enercon_E53_800kW	Enercon_E66_1870kW
Enercon_E66_2000kW	Enercon_E70_2.3MW	Enercon_E82_2.0MW
Enercon_E82_2.3MW	Enercon_E82_3.0MW	Enercon_E101_3.0MW
Enercon_E126_7.5MW	EWT_DW52_500kW	EWT_DW54_500kW

EWT_DW90_2MW	EWT_DW96_2MW	Gamesa_G52_850kW
Gamesa_G58_850kW	Gamesa_G80_2.0MW	Gamesa_G83_2.0MW
Gamesa_G87_2.0MW	Gamesa_G90_2.0MW	GE_1.5sl_1.5MW
GE_1.5sle_1.5MW	GE_1.5xle_1.5MW	GE_1.6MW
GE_2.5xl_2.5MW	GE_3.6sl_3.6MW	Leitwind_LTW70_1.7MW
Leitwind_LTW70_2.0MW	Leitwind_LTW77_1.5MW	Leitwind_LTW80_1.5MW
Leitwind_LTW80_1.8MW	Leitwind_LTW101_3.0MW	Nordex_N60_1.3MW
Nordex_N70_1.5MW	Nordex_N90_2.5MW	Nordex_N100_2.5MW
Nordex_S70_1.5MW	Nordex_S77_1.5MW	Nordic_1000_1.0MW
PowerWind_56_500kW	Repower_5M_5.0MW	Repower_MM82_2.0MW
Repower_MM92_2.0MW	Siemens_SWT-2.3MW-93m	Siemens_SWT-2.3MW-101m
Siemens_SWT-3.6MW-107m	Siemens_SWT-3.6MW-120m	Suzlon_S64_1.25MW
Suzlon_S64_950kW	Suzlon_S66_1.25MW	Suzlon_S88_2.1MW
VensysEnergy_77_1.5MW	Vensys_82_1.5MW	Vensys_100_2.5MW
Vensys_109_2.5MW	Vensys_112_2.5MW	Vestas_V27_225kW
Vestas_V39_500kW	Vestas_V52_850kW	Vestas_V80_2.0MW_os
Vestas_V80_2.0MW	Vestas_V82_1.65MW	Vestas_V90_2.0MW
Vestas_V90_3.0MW	Vestas_V112_3MW	Vestas_V164_7.0MW_os

Value

Returns a data frame binding the given data

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- `bty`: Type of box to be drawn around the plot region. Allowed values are `"o"` (the default), `"l"`, `"7"`, `"c"`, `"u"`, or `"]"`. The resulting box resembles the corresponding upper case letter. A value of `"n"` suppresses the box.
- `bty.leg`: Type of box to be drawn around the legend. Allowed values are `"n"` (no box, the default) and `"o"`.
- `cex`: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `cex.leg`: Amount by which legend text should be scaled, as numeric value.
- `col`: Vector of colours. The first colour is used for the power curve. If only one coefficient is available, the second colour is used for this coefficient, if both coefficients are available, the second colour is used for `pc` and the third for `ct`.
- `col.axis`: Colour to be used for axis annotations – default is `"black"`.
- `col.box`: Colour to be used for the box around the plot region (if `bty`) – default is `"black"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `col.leg`: Colour to be used for legend text – default is `"black"`.
- `col.ticks`: Colours for the axis line and the tick marks respectively – default is `"black"`.

- las: Style of axis labels. One of 0 (always parallel to the axis, default), 1 (always horizontal), 2 (always perpendicular to the axis), 3 (always vertical).
- legend: If TRUE (the default) a legend is drawn.
- leg.text: A character or expression vector to appear in the legend.
- lty: Vector of line types, assigned like col. See par for available line types.
- lwd: Vector of line widths, assigned like col. See par for usage.
- mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot – default is c(5, 5, 1, 5) for one y axis and c(5, 5, 1, 1) for two y axes.
- mgp: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is c(3, 1, 0).
- pos.leg: Position of legend – one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Use NULL to hide the legend.
- xlab: Alternative label for the x axis.
- ylab: Alternative labels for the y axis, as vector of the form c(left axis label, right axis label).
- ylim: Limits of the y axis, as vector of two values.
- x.intersp: Horizontal interspacing factor for legend text, as numeric – default is 0.4.
- y.intersp: Vertical line distance for legend text, as numeric – default is 0.4.

Note

All power curves are provided without any warranty of accuracy and timeliness. Reliable data can only be received from the respective manufacturer directly.

Author(s)

Christian Graul

Source

Wind turbine generator files (*.wtg) were collected from the WAsP website:

http://www.wasp.dk/Download/PowerCurves.aspx

Power curve files (*.pow) were collected from the WindPower Program website:

http://www.wind-power-program.com/download.htm

Both links are not active anymore and the turbine list is now a bit outdated.

References

Betz, A. (1966) *Introduction to the Theory of Flow Machines*. Oxford: Pergamon Press

Burton, T., Sharpe, D., Jenkins, N., Bossanyi, E. (2001) *Wind Energy Handbook*. New York: Wiley

International Electrotechnical Commission (2005) IEC 61400-12 Wind Turbines – Part 12-1: Power Performance Measurements of Electricity Producing Wind Turbines. IEC Standard

Milan, P., Wächter, M., Barth, S., Peinke, J. (2010) Power Curves for Wind Turbines. In: Wei Tong (Ed.), Wind Power Generation and Wind Turbine Design, Chapter 18, p. 595–612, Southampton: WIT Press

Ragheb, M., Ragheb, A.M. (2011) Wind Turbines Theory – The Betz Equation and Optimal Rotor Tip Speed Ratio. In: Rupp Carriveau (Ed.), Fundamental and Advanced Topics in Wind Power, Chapter 2, p. 19–38, InTech

Examples

```
## Not run:
## create power curve
# minimal theoretic power curve
pc.1 <- pc(list(1:25, c(0, 0, seq(0,1000,length.out=8), rep(1000,15))))
pc.1

# detailed power curve
v <- seq(3, 25, 0.5)
p <- c(5, 15.5, 32, 52, 71, 98, 136, 182, 230, 285, 345, 419, 497, 594,
  687, 760, 815, 860, 886, rep(900, 26))
cp <- c(0.263, NA, 0.352, NA, 0.423, NA, 0.453, NA, 0.470, NA, 0.478,
  NA, 0.480, NA, 0.483, NA, 0.470, NA, 0.429, NA, 0.381, NA, 0.329,
  NA, 0.281, NA, 0.236, NA, 0.199, NA, 0.168, NA, 0.142, NA, 0.122,
  NA, 0.105, NA, 0.092, NA, 0.080, NA, 0.071, NA, 0.063)
ct <- c(0.653, NA, 0.698, NA, 0.705, NA, 0.713, NA, 0.720, NA, 0.723, NA,
  0.724, NA, 0.727, NA, 0.730, NA, 0.732, NA, 0.385, NA, 0.301, NA, 0.242,
  NA, 0.199, NA, 0.168, NA, 0.146, NA, 0.128, NA, 0.115, NA, 0.103, NA,
  0.094, NA, 0.086, NA, 0.079, NA, 0.073)

# variables as list
pc.2 <- pc(list(v=v, p=p, cp=cp, ct=ct),
  rho=1.195, rated.p=900, desc="PowerWind 56")
pc.2

# variables as data frame
pc.3 <- pc(data.frame(v=v, p=p, cp=cp, ct=ct),
  rho=1.195, rated.p=900, desc="PowerWind 56")
pc.3

## import power curve
## note: XML package required for WAsP .wtg files
vestas.v90 <- pc("Vestas_V90_2.0MW.wtg")  # bReeze wtg file
repower.mm92 <- pc("Repower_MM92_2.0MW.pow")  # bReeze pow file
#my.pc <- pc("~/Projects/bReeze/Sandbox/myPC.wtg")  # user file


## plot power curve
plot(pc.2)  # default
plot(pc.2, cp=FALSE, ct=FALSE)  # drop coefficients

# customize plot
plot(pc.2, bty="u", bty.leg="o", cex.axis=0.8, cex.lab=0.9,
  cex.leg=0.7, col=c("red", gray(0.4), gray(0.4)), col.axis=gray(0.2),
```

```
  col.box=gray(0.5), col.lab=gray(0.2), col.leg=gray(0.2),
  col.ticks=gray(0.5), las=2, leg.text=c("electric Power",
  "power coefficient", "thrust coefficient"), lty=2:4, lwd=c(2,1,1),
  mar=c(3.5,3.5,0.5,3.5), mgp=c(1.8,0.6,0), pos.leg="top",
  xlab="velocity [m/s]", ylab=c("electric power", "coefficients"),
  ylim=c(0,1100), x.intersp=1, y.intersp=1)

## End(Not run)
```

`polar.plot` *Plot wind speed vs. direction*

Description

Plots wind speeds against directions in a polar plot.

Usage

```
polar.plot(mast, v.set=1, dir.set=1, subset, ...)
pol(mast, v.set=1, dir.set=1, subset, ...)
```

Arguments

`mast`	Met mast object created by `mast`.
`v.set`	Set used for wind speed values, specified as set number or set name.
`dir.set`	Set used for wind direction values, specified as set number or set name.
`subset`	Optional start and end time stamp for a data subset, as string vector `c(start, end)`. The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
`...`	Optional graphical parameters, see below for details.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- `cex`: Numeric value, giving the amount by which text on the plot should be scaled relative to the default (which is 1).
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `cex.pts`: Amount by which the plot symbols should be scaled, as numeric value.
- `circles`: Manual definition of circles to be drawn, as numeric vector of the form c(inner circle, outer circle, interval between the circles).
- `col`: The colour of the symbols plotted.
- `col.axis`: Colour to be used for axis annotations – default is `"gray45"`.
- `col.circle`: Colour to be used for circles – default is `"gray45"`.

- col.cross: Colour to be used for axis lines – default is "gray45".
- col.lab: Colour to be used for axis labels – default is "black".
- fg: If TRUE, sectors are plotted in foreground (on top of axis lines and circles) – default is FALSE.
- lty.circle: Line type of circles – default is "dashed". See par for available line types.
- lty.cross: Line type of axis lines – default is "solid". See par for available line types.
- lwd.circle: Line width of circles, as numeric value – default is 0.7.
- lwd.cross: Line width of axis lines, as numeric value – default is 0.7.
- pch: Either an integer specifying a symbol or a single character to be used as symbol – default is ".", which is drawn much faster than other symbols. See points for possible values and their interpretation.
- pos.axis: Position of axis labels in degree, as numeric value – default is 60.

Author(s)

Christian Graul

See Also

mast

Examples

```
## Not run:
# load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)

# plot v vs. dir
polar.plot(mast=neubuerg)
polar.plot(mast=neubuerg, v.set=3, dir.set=2)
polar.plot(mast=neubuerg, v.set="set3", dir.set="set2") # same as above

# data subsets
polar.plot(mast=neubuerg,
  subset=c("2009-12-01 00:00:00", "2009-12-31 23:50:00"))
polar.plot(mast=neubuerg,
  subset=c("2010-01-01 00:00:00", NA)) # just 'start' time stamp
polar.plot(mast=neubuerg,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

# customize plot
polar.plot(mast=neubuerg, cex.axis=1.2, cex.lab=1.5, cex.pts=0.8,
  circles=c(10,20,5), col="red3", col.axis=gray(0.2), col.circle=gray(0.3),
```

```
  col.cross=gray(0.3), col.lab=gray(0.2), fg=TRUE, lty.circle="dotted",
  lty.cross="longdash", lwd.circle=1.2, lwd.cross=1.2, pch=1, pos.axis=135)

## End(Not run)
```

`set` *Creation of datasets*

Description

Creates a dataset object, by combining all signals of one height of measurement.

Usage

```
set(height, desc, v.avg, v.max, v.min, v.std,
  dir.avg, dir.std, tmp, ...)
```

Arguments

`height`	Height of measurement in m as numeric value.
`desc`	Plain text information about the set, signals, instruments, etc. as string (optional).
`v.avg`	Average of wind speed within interval in m/s as numeric vector (optional, if at least one other signal is given).
`v.max`	Maximum of wind speed within interval in m/s as numeric vector (optional, if at least one other signal is given).
`v.min`	Minimum of wind speed within interval in m/s as numeric vector (optional, if at least one other signal is given).
`v.std`	Standard deviation of wind speed within interval in m/s as numeric vector (optional, if at least one other signal is given).
`dir.avg`	Average of wind direction within interval in degrees from north as numeric vector (optional, if at least one other signal is given).
`dir.std`	Standard deviation of wind direction within interval in degrees from north as numeric vector (optional, if at least one other signal is given).
`tmp`	Temperature in °C as numeric vector (optional, if at least one other signal is given).
`...`	Further signals, e.g. air pressure, humidity, etc. as numeric vector (optional).

Details

Anemometer and wind vanes are usually mounted as pairs at same or similar heights of the met mast. Signals with marginal differences in height, about 1 or 2 m, may or may not be combined to one dataset. In case of a combination of two heights the height of the anemometer should be used for the whole dataset, as in subsequent analyses wind speed might be extrapolated to other

heights. A dataset shall not contain data from more than one sensor of the same type, say data of two anemometers.

A typical interval of wind measurments is 10 minutes, but also other intervals are applicable.

For datasets containing mean wind speed `v.avg` and its standard deviation within the time interval `v.std`, the turbulence intensity is calculated and added to the data, named `turb.int`. See `turbulence` for more details about turbulence intensity.

Value

Returns a dataset object which is a list of:

`height`	Height of measurement.
`data`	Data of measured signals.

Author(s)

Christian Graul

See Also

`mast`

Examples

```
## Not run:
# load data
data("winddata", package="bReeze")

# minimal dataset
s <- set(height=40, v.avg=winddata[,2])

# detailed dataset
s2 <- set(height=40, desc=
  "C1: cup anemometer (SN: 4.3250.128), A1: wind vane (SN: 4.2800.205)",
  v.avg=winddata[,2], v.max=winddata[,3], v.min=winddata[,4],
  v.std=winddata[,5], dir.avg=winddata[,14], dir.std=winddata[,15])

## End(Not run)
```

---

timestamp *Format time stamps*

Description

Converts time stamps from string to `POSIXlt`. The conversion specification (pattern) is looked up if not given as argument.

Usage

```
timestamp(timestamp, pattern, tz)
ts(timestamp, pattern, tz)
```

Arguments

timestamp	Time stamp as string vector.
pattern	Conversion specification of time stamp as string (optional). See Details for usage.
tz	Optional character string specifying the time zone to be used for the conversion. System-specific (see as.POSIXlt), but "" is the current time zone (used as default). Use "?" to check timestamp for time zone abbreviation.

Details

If the time stamp is already formatted as POSIXlt, the usage of timestamp is not necessary. strptime can also be used to create an applicable time stamp. Usage of timestamp is recommeded, since it checks the created time stamp, thus faulty time stamps are avoided.

Pattern

A conversion specification is introduced by "%", usually followed by a single letter. Any character in the format string not part of a conversion specification is interpreted literally. Widely implemented conversion specifications include:

- %d: day of the month as decimal number (01–31)
- %m: month as decimal number (01–12)
- %y: year without century (00–99), where values 00–68 are prefixed by 20 and 69–99 by 19
- %Y: year with century
- %H: hour as decimal number (00–23)
- %M: minute as decimal number (00–59)
- %S: second as decimal number (00–61)

For details see strptime.

Value

Returns a POSIXlt vector.

Author(s)

Christian Graul

See Also

POSIXlt, strptime, mast

Examples

```
## Not run:
# load and prepare data
data("winddata", package="bReeze")

# format time stamp
timestamp <- timestamp(timestamp=winddata[,1])

# format time stamp with given pattern
timestamp.2 <- timestamp(timestamp=winddata[,1], "%d.%m.%Y %H:%M")

# wrong pattern (
timestamp.2 <- timestamp(timestamp=winddata[,1], "%d.%m.%y %H:%M")

# strange time stamp pattern
ts <- c("TS 08/2012-10 8h10m30s", "TS 08/2012-10 8h20m30s",
  "TS 08/2012-10 8h30m30s")
timestamp.3 <- timestamp(timestamp=ts) # pattern not found
timestamp.3 <- timestamp(timestamp=ts, "TS %m/%Y-%d %Hh%Mm%Ss")

# time zones
# manually define time zone
timestamp.4 <- timestamp(timestamp=winddata[,1], tz="CET")

# get time zone from timestamp
timestamp.5 <- timestamp(timestamp="2012-08-08 22:55 GMT", tz="?")

## End(Not run)
```

`turb.iec.plot` *Plot turbulence intensity site classification*

Description

Plots the turbulence intensity and site classification after IEC.

Usage

```
turb.iec.plot(mast, set, subset, ...)
iec(mast, set, subset, ...)
```

Arguments

`mast`	Met mast object created by `mast`.
`set`	Set used for plotting specified as set number or set name.
`subset`	Optional start and end time stamp for a data subset, as string vector `c(start, end)`. The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
`...`	Optional graphical parameters, see below for details.

Details

The IEC defines wind turbine classes by wind speed and turbulence characteristics. In terms of turbulence intensity three reference values (at 15 m/s) are defined:

Turbulence class	*Reference value*
A	0.16
B	0.14
C	0.12

`plotTurbIEC` plots these IEC references together with the sites values to allow for a classification.

See `turbulence` for a definition of turbulence intensity.

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- `border`: Colour, used for the border around the bars – default is `"white"`.
- `bty`: Type of box to be drawn around the plot region. Allowed values are `"o"` (the default), `"l"`, `"7"`, `"c"`, `"u"`, or `"]"`. The resulting box resembles the corresponding upper case letter. A value of `"n"` suppresses the box.
- `bty.leg`: Type of box to be drawn around the legend. Allowed values are `"n"` (no box, the default) and `"o"`.
- `cex`: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- `cex.axis`: Amount by which axis annotations should be scaled, as numeric value.
- `cex.lab`: Amount by which axis labels should be scaled, as numeric value.
- `cex.leg`: Amount by which legend text should be scaled, as numeric value.
- `col`: Colour, used to fill the bars.
- `col.axis`: Colour to be used for axis annotations – default is `"black"`.
- `col.box`: Colour to be used for the box around the plot region (if `bty`) – default is `"black"`.
- `col.lab`: Colour to be used for axis labels – default is `"black"`.
- `col.leg`: Colour to be used for legend text – default is `"black"`.
- `col.ticks`: Colours for the axis line and the tick marks respectively – default is `"black"`.
- `las`: Style of axis labels. One of `0` (always parallel to the axis, default), `1` (always horizontal), `2` (always perpendicular to the axis), `3` (always vertical).
- `legend`: If `TRUE` (the default) a legend is drawn.
- `leg.text`: A character or `expression` vector to appear in the legend.
- `line`: Vector of three colours – one for each IEC turbulence class.
- `lty`: Vector of three line types – one for each IEC turbulence class. See `par` for available line types.
- `lwd`: Vector of three line widths – one for each IEC turbulence class. See `par` for usage.

- mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot (only for plots with one dataset) – default is c(4.5, 4.5, 1, 1).
- mgp: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is c(2.2, 0.7, 0).
- pos.leg: Position of legend – one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Use NULL to hide the legend.
- space: Numeric value between 0 and 1, giving the space left before each bar. Default space is 0.2.
- xlab: Alternative label for the x axis.
- ylab: Alternative label for the y axis.
- xlim: Limits of the x axis, as vector of two values.
- ylim: Limits of the y axis, as vector of two values.
- x.intersp: Horizontal interspacing factor for legend text, as numeric – default is 0.4.
- y.intersp: Vertical interspacing factor for legend text, as numeric – default is 0.8.

Author(s)

Christian Graul

References

International Electrotechnical Commission (2005) IEC 61400-1 Wind Turbines – Part 1: Design Requirements. IEC Standard

See Also

mast

Examples

```
## Not run:
# load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5])
set30 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9])
set20 <- set(height=20, v.avg=winddata[,10], v.std=winddata[,13])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)

# plot
turb.iec.plot(mast=neubuerg, set=1)
turb.iec.plot(mast=neubuerg, set="set1") # same as above

# data subsets
turb.iec.plot(mast=neubuerg, set=1,
  subset=c("2009-12-01 00:00:00", "2009-12-31 23:50:00"))
```

```
turb.iec.plot(mast=neubuerg, set=1,
  subset=c("2010-01-01 00:00:00", NA)) # just 'start' time stamp
turb.iec.plot(mast=neubuerg, set=1,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

# customize plot
turb.iec.plot(mast=neubuerg, set=1, bty="l", cex.axis=0.8, cex.lab=0.9,
  cex.leg=0.7, col.axis="darkblue", col.box="lightblue", col.lab=
  "darkblue", col.leg="darkblue", col.ticks="darkblue", las=0,
  leg.text=c("IEC class A", "IEC class B", "IEC class C", "measured"),
  mar=c(3,3,0.5,0.5), mgp=c(1.8,0.5,0), pos.leg="top", xlab="v [m/s]",
  ylab="ti [-]", xlim=c(0,25), ylim=c(0,0.5), x.intersp=1, y.intersp=1)

# customize bars
turb.iec.plot(mast=neubuerg, set=1, col="gray", border="black", space=0.6)

# customize lines
turb.iec.plot(mast=neubuerg, set=1, line=gray(1:3 / 10), lty=2:4,
  lwd=0.5:2.5)

## End(Not run)
```

`turbulence` *Calculation of turbulence intensity*

Description

Calculates turbulence intensity and mean wind speed for each given direction sector.

Usage

```
turbulence(mast, turb.set, dir.set, num.sectors=12,
  bins=c(5, 10, 15, 20), subset, digits=3, print=TRUE)
turb(mast, turb.set, dir.set, num.sectors=12,
  bins=c(5, 10, 15, 20), subset, digits=3, print=TRUE)
```

Arguments

`mast`	Met mast object created by `mast`.
`turb.set`	Set used for turbulence intensity, specified as set number or set name (optional, if `dir.set` is given).
`dir.set`	Set used for wind direction, specified as set number or set name (optional, if `turb.set` is given).
`num.sectors`	Number of wind direction sectors as integer value greater 1. Default is `12`.
`bins`	Wind speed bins as numeric vector or `NULL` if no classification is desired. Default is `c(5, 10, 15, 20)`.

subset	Optional start and end time stamp for a data subset, as string vector c(start, end). The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
digits	Number of decimal places to be used for results as numeric value. Default is 3.
print	If TRUE (the default), results are printed directly.

Details

Turbulence can be perceived as wind speed fluctuations on a relatively short time scale and it strongly depends on surface roughness, terrain features, as well as thermal effects. High turbulence should be avoided, since it is a main driver of fatigue loads and might decrease energy output. A measure of the overall level of turbulence, is the turbulence intensity I, which is defined as:

$$I = \frac{\sigma}{\bar{v}}$$

where σ is the standard deviation of wind speed – usually measured over a 10-minutes period – and $\bar{v}$ is the mean wind speed over this period.

Value

Returns a data frame containing:

wind.speed	Mean wind speed for each direction sector.
total	Total turbulence intensity for each direction sector.
...	Turbulence intensities per direction sector for each given wind speed bin.

Optional graphical parameters for plotting

The following graphical parameters can optionally be added to customize the plot:

- cex: Numeric value, giving the amount by which text on the plot should be scaled relative to the default (which is 1).
- cex.axis: Amount by which axis annotations should be scaled, as numeric value.
- cex.lab: Amount by which axis labels should be scaled, as numeric value.
- circles: Manual definition of circles to be drawn, as numeric vector of the form c(inner circle, outer circle, interval between the circles).
- col: Colour to be used for the sectors.
- col.axis: Colour to be used for axis annotations – default is "gray45".
- col.border: Colour to be used for sector borders – default is NULL (no border is drawn).
- col.circle: Colour to be used for circles – default is "gray45".
- col.cross: Colour to be used for axis lines – default is "gray45".
- col.lab: Colour to be used for axis labels – default is "black".
- fg: If TRUE, sectors are plotted in foreground (on top of axis lines and circles) – default is FALSE.
- lty.circle: Line type of circles – default is "dashed". See par for available line types.

- `lty.cross`: Line type of axis lines – default is `"solid"`. See `par` for available line types.
- `lwd.border`: Line width of the sector borders – default is `0.5`. Only used if `col.border` is set.
- `lwd.circle`: Line width of circles, as numeric value – default is `0.7`.
- `lwd.cross`: Line width of axis lines, as numeric value – default is `0.7`.
- `pos.axis`: Position of axis labels in degree, as numeric value – default is `60`.
- `sec.space`: Space between plotted sectors, as numeric value between `0` and `1` – default is `0.2`.

Author(s)

Christian Graul

References

Albers, A. (2010) Turbulence and Shear Normalisation of Wind Turbine Power Curve. Proceedings of the EWEC 2010, Warsaw, Poland

Burton, T., Sharpe, D., Jenkins, N., Bossanyi, E. (2001) *Wind Energy Handbook*. New York: Wiley

Langreder, W. (2010) Wind Resource and Site Assessment. In: Wei Tong (Ed.), Wind Power Generation and Wind Turbine Design, Chapter 2, p. 49–87, Southampton: WIT Press

See Also

`mast`

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10], v.std=winddata[,13])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)


## calculate turbulence intensity
turbulence(mast=neubuerg, turb.set=1)  # default
turbulence(mast=neubuerg, turb.set=1, dir.set=2)  # use different datasets
turbulence(mast=neubuerg, turb.set="set1", dir.set="set2")  # same as above
turbulence(mast=neubuerg, turb.set=1, num.sectors=4)  # change sector number

# calculate turbulence intensity for 1 m/s speed bins and without binning
turbulence(mast=neubuerg, turb.set=1, bins=1:25)
turbulence(mast=neubuerg, turb.set=1, bins=NULL)
```

```
# data subset
turbulence(mast=neubuerg, turb.set=1,
  subset=c(NA, "2010-01-01 00:00:00"))

# change number of digits and hide results
turbulence(mast=neubuerg, turb.set=1, digits=2)
neubuerg.ti <- turbulence(mast=neubuerg, turb.set=1, print=FALSE)
neubuerg.ti


## plot turbulence intensity object
plot(neubuerg.ti)  # default

# change colour, text size etc.
plot(neubuerg.ti, cex.axis=0.7, cex.lab=0.9, circles=c(0.05,0.20,0.05),
  col="lightgray", col.axis="darkgray", col.border="gray", col.circle="darkgray",
  col.cross="darkgray", col.lab="darkgray", fg=TRUE, lty.circle="dotdash",
  lty.cross="longdash", lwd.border=1.2, lwd.circle=1.2, lwd.cross=1.2,
  pos.axis=135, sec.space=0.6)

## End(Not run)
```

`uncertainty` *Uncertainty assessment*

Description

Calculates probability of exceedance based of AEP data and given uncertainties of applied methods.

Usage

```
uncertainty(aep, uc.values, uc.names, prob=seq(5,95,5),
  digits=c(0,0), print=TRUE)
uc(aep, uc.values, uc.names, prob=seq(5,95,5),
  digits=c(0,0), print=TRUE)

## S3 method for class 'uncertainty'
plot(x, type=c("prob", "uncert"), p.values=c(50,75,90), ...)
```

Arguments

`aep`	AEP object created by `aep`.
`uc.values`	Uncertainty values (in percent) for applied methods as numeric vector.
`uc.names`	Names of the uncertainty components. Normally a string vector of the same length as `uc.values`. If `uc.names` is a string vector with the length of `uc.values` + 1, the calculated total uncertainty is named after the additional name (default is "total"). If `uc.names` is `NULL`, the methods are numbered consecutively.

prob	Probability values used for the probability of exceedance analysis. Default is seq(5,95,5).
digits	Number of decimal places to be used for results as numeric vector. The first value is used for uncertainties of applied methods, the second for probability of exceedance results. Default is c(0,0).
print	If TRUE, results are printed directly.
x	Uncertainty object created by uncertainty.
type	Type of plot as string. One of "prob" (probability of exceedance plot) or "uncert" (uncertainties overview plot).
p.values	The P-values highlighted in the plot as numeric vector – default is P50, P75 and P90.
...	Arguments to be passed to methods. For optional graphical parameters see below.

Details

A wind resource assessment, like every statistical analysis, is only complete with an accompanying uncertainty assessment. uncertainty provides a simple tool to arrange uncertainties of the methods applied and analyse their approximate effects to the energy output of a turbine. The total uncertainty arises from many uncertainty components of different type and impact. Common components are wind measurement (sensor quality/calibration, mast influences, etc.), data analysis (missing data, data selection, simplifications/assumptions etc.), long term data (reference data, length of measuring period, etc.), flow modelling (horizontal and vertical extrapolations, etc.) or power curve (measurement quality, assumptions, etc.).

Assuming all uncertainty components to be independent from each other, the combined standard uncertainty is calculated as follows:

$$U = \sqrt{u_1 + u_2 + ... + u_n}$$

where U is the total uncertainty and u_1 untill u_n are the uncertainty components.

Value

Returns a list containing:

uncertainty.meth
: Table of uncertainty components and their uncertainty value.

prob.exceedance
: Table of probability values and the related annual energy production.

Optional graphical parameters

*The following graphical parameters can optionally be added to customize the probability of exceedance plot (*type="prob"*):*

- bty: Type of box to be drawn around the plot region. Allowed values are "o" (the default), "l", "7", "c", "u", or "]". The resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box.

- bty.leg: Type of box to be drawn around the legend. Allowed values are "n" (no box, the default) and "o".
- cex: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- cex.axis: Amount by which axis annotations should be scaled, as numeric value.
- cex.lab: Amount by which axis labels should be scaled, as numeric value.
- cex.leg: Amount by which legend text should be scaled, as numeric value.
- col: Vector of n colours – [1] for the probability curve, [2:n] for the highlighted P-values (p.values).
- col.axis: Colour to be used for axis annotations – default is "black".
- col.box: Colour to be used for the box around the plot region (bty) – default is "black".
- col.lab: Colour to be used for axis labels – default is "black".
- col.leg: Colour to be used for legend text – default is "black".
- col.ticks: Colours for the axis line and the tick marks respectively – default is "black".
- las: Style of axis labels. One of 0 (always parallel to the axis, default), 1 (always horizontal), 2 (always perpendicular to the axis), 3 (always vertical).
- legend: If TRUE (the default) a legend is drawn.
- leg.text: Vector of strings used as alternative legend text.
- lty: Vector of line types – usage like col. See par for available line types.
- lwd: Vector of line widths – usage like col. See par for usage.
- mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot – default is c(4.5, 4.5, 1, 1).
- mgp: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is c(2.7, 0.7, 0).
- pos.leg: Position of legend – one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center".
- xlab: Alternative label for the x axis.
- ylab: Alternative label for the y axis.
- xlim: Limits of the x axis, as vector of two values.
- ylim: Limits of the y axis, as vector of two values.
- x.intersp: Horizontal interspacing factor for legend text, as numeric – default is 0.4.
- y.intersp: Vertical line distance for legend text, as numeric – default is 0.8.

*Optional graphical parameters for the uncertainty overview plot (*type="uncert"*):*

- border: Colour to be used for the border of the bars – usage like col, as default no border is drawn.
- cex: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- cex.axis: Amount by which the names of the uncertainty components should be scaled, as numeric value.

- cex.text: Amount by which the uncertainty values inside the bars should be scaled, as numeric value.
- col: One or a vector of colours. If one colour is specified, this colour is used for all bars. If two colours are specified, the first colour is used for uncertainty component bars and the second for the total uncertainty bar. Separate colours for each bar can also be specified.
- col.axis: Colour to be used for the names of the uncertainty components – default is "black".
- col.text: Colour to be used for the uncertainty values inside the bars – default is "white".
- mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot.
- mgp: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is c(3, 1, 0).
- space: Numeric value between 0 and 1, giving the space left before each bar. Default space is 0.2.

Author(s)

Christian Graul

References

Measnet (2009) MEASNET Procedure: Evaluation of Site Specific Wind Conditions, Version 1

See Also

aep

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set1 <- set(height=40, v.avg=winddata[,2], v.std=winddata[,5],
  dir.avg=winddata[,14])
set2 <- set(height=30, v.avg=winddata[,6], v.std=winddata[,9],
  dir.avg=winddata[,16])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set1, set2)
neubuerg <- clean(mast=neubuerg)


## calculate uncertainty
# calculate AEP
nb.wp <- profile(mast=neubuerg, v.set=c(1,2), dir.set=1,
  print=FALSE)
pw.56 <- pc("PowerWind_56_900kW.wtg")
nb.aep <- aep(profile=nb.wp, pc=pw.56, hub.h=71, print=FALSE)

# calculate uncertainty
uncertainty(nb.aep, uc.values=c(5,10,5,5),
```

```
  uc.names=c("Wind Measurement Mast", "Long Term Correlation",
  "Flow Model", "Power Curve"))

# unnamed uncertainty components
uncertainty(nb.aep, uc.values=c(5,10,5,5),
  uc.names=NULL)

# new name for combined uncertainty
uncertainty(nb.aep, uc.values=c(5,10,5,5),
  uc.names=c("Wind Measurement Mast", "Long Term Correlation",
  "Flow Model", "Power Curve", "Total Uncertainty"))

# changed probability values
uncertainty(nb.aep, uc.values=c(5,10,5,5),
  uc.names=c("Wind Measurement Mast", "Long Term Correlation",
  "Flow Model", "Power Curve"), prob=seq(50,90,10))

# change number of digits and hide results
neubuerg.uc <- uncertainty(nb.aep, uc.values=c(5,10,5,5),
  uc.names=c("Wind Measurement Mast", "Long Term Correlation",
  "Flow Model", "Power Curve"), digits=c(1,2), print=FALSE)
neubuerg.uc


## plot uncertainty objects - probability of exceedance plot
plot(neubuerg.uc)  # default
plot(neubuerg.uc, p.values=c(50, 95))  # change highlighted P-values

# change colours, line types, line width and text size
plot(neubuerg.uc, col="blue", lty=c(1, 2, 3, 4), lwd=2, cex=1.2)

# freaky
plot(neubuerg.uc, bty="l", bty.leg="o", cex.axis=2,
  cex.lab=0.5, cex.leg=0.8, col=c(5, 10, 15, 20), col.axis="sienna",
  col.box="purple", col.lab="plum", col.leg="orchid", col.ticks="gold",
  las=0, lty=c(8, 7, 6, 5), lwd=c(5, 3, 1, 0.5), mar=c(6, 5, 4, 3),
  mgp=c(4, 2, 1), pos.leg="bottomleft", xlim=c(0.1, 0.9), ylim=c(1000, 2000),
  x.intersp=2, y.intersp=1.5)


## plot uncertainty objects - uncertainty overview plot
plot(neubuerg.uc, type="uncert")  # default

# change colours and border
plot(neubuerg.uc, type="uncert", col="red", border="red4")
plot(neubuerg.uc, type="uncert", col=c(gray(0.7), gray(0.5)),
  border=c(gray(0.6), gray(0.4)))
plot(neubuerg.uc, type="uncert", col=c(5:1), border=c(1:5))

# change text size, space and margin
plot(neubuerg.uc, type="uncert", cex=1.5, space=0.1, mar=c(1, 13, 1, 1))

# freaky
```

```
plot(neubuerg.uc, type="uncert", border=c(11, 22, 33, 44, 55), cex.axis=0.7,
  cex.text=2, col=c("maroon", "navy", "thistle", "rosybrown", "papayawhip"),
  col.axis="pink3", col.text="seagreen", mar=c(3, 8, 2, 1), mgp=c(0, 1, 2), space=1)

## End(Not run)
```

`weibull` *Calculation of Weibull parameters*

Description

Calculates the shape parameters (k) and scale parameters (A) of the Weibull distribution per direction sector.

Usage

```
weibull(mast, v.set, dir.set, num.sectors=12,
  subset, digits=3, print=TRUE)
wb(mast, v.set, dir.set, num.sectors=12,
  subset, digits=3, print=TRUE)

## S3 method for class 'weibull'
plot(x, type=c("hist", "dir"), show.ak=FALSE, ...)
```

Arguments

`mast`	Met mast object created by `mast`.
`v.set`	Set used for wind speed, specified as set number or set name (optional, if dir.set is given).
`dir.set`	Set used for wind direction, specified as set number or set name (optional, if `v.set` is given).
`num.sectors`	Number of wind direction sectors as integer value greater 1. Default is `12`.
`subset`	Optional start and end time stamp for a data subset, as string vector `c(start, end)`. The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
`digits`	Number of decimal places to be used for results as numeric value. Default is `3`.
`print`	If `TRUE`, results are printed directly.
`x`	Weibull object, created by `weibull`.
`type`	Plot type - one of `"hist"` (histogram plot) or `"dir"` (directional plot).
`show.ak`	If `TRUE` (the default), the Weibull parameters A and k are added to the legend.
`...`	Arguments to be passed to methods. For optional graphical parameters see below.

Details

To evaluate the potential energy production of a site the observed data of a particular measurement period must be generalized to a wind speed distribution. This is commonly done by fitting the Weibull function to the data. The two-parametered Weibull distribution is expressed mathematically as:

$$f(v) = \frac{k}{A}\left(\frac{v}{A}\right)^{k-1} e^{-\left(\frac{v}{A}\right)^{k}}$$

where $f(v)$ is the frequency of occurence of wind speed v, A is the scale parameter (measure for the wind speed) and k is the shape parameter (description of the shape of the distribution).

The resulting Weibull distribution characterizes the wind regime on the site and can directly be used for the calculation of the potential energy production of a wind turbine (see aep).

Value

Returns a data frame containing:

k	Shape parameter of the Weibull distribution for each direction sector.
A	scale parameter of the Weibull distribution for each direction sector.
wind.speed	Mean wind speed of the Weibull distribution for each direction sector.
frequency	Frequency of the Weibull distribution for each direction sector.

Optional graphical parameters

Graphical parameters to customize the histogram plot:

- border: Colour, used for the border around the bars – default is "white".
- breaks: A numeric vector giving the breakpoints between histogram cells.
- bty: Type of box to be drawn around the plot region. Allowed values are "o" (the default), "l", "7", "c", "u", or "]". The resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box.
- bty.leg: Type of box to be drawn around the legend. Allowed values are "n" (no box, the default) and "o".
- cex: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- cex.axis: Amount by which axis annotations should be scaled, as numeric value.
- cex.lab: Amount by which axis labels should be scaled, as numeric value.
- cex.leg: Amount by which legend text should be scaled, as numeric value.
- col: Colour, used to fill the bars.
- col.axis: Colour to be used for axis annotations – default is "black".
- col.box: Colour to be used for the box around the plot region (if bty) – default is "black".
- col.lab: Colour to be used for axis labels – default is "black".
- col.leg: Colour to be used for legend text – default is "black".
- col.ticks: Colours for the axis line and the tick marks respectively – default is "black".

- las: Style of axis labels. One of 0 (always parallel to the axis, default), 1 (always horizontal), 2 (always perpendicular to the axis), 3 (always vertical).
- legend: If TRUE (the default) a legend is drawn.
- leg.text: A character or expression vector to appear in the legend.
- line: Colour, used for the Weibull fit line.
- lty: Line type of the Weibull fit line – see par for available line types.
- lwd: Line width for the Weibull fit line – see par for usage.
- mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot (only for plots with one dataset) – default is c(4.5, 4.5, 1, 1).
- mgp: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is c(2.2, 0.7, 0).
- pos.leg: Position of legend – one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Use NULL to hide the legend.
- xlab: Alternative label for the x axis.
- ylab: Alternative label for the y axis.
- xlim: Limits of the x axis, as vector of two values.
- ylim: Limits of the y axis, as vector of two values.
- x.intersp: Horizontal interspacing factor for legend text, as numeric – default is 0.4.
- y.intersp: Vertical interspacing factor for legend text, as numeric – default is 0.8.

Graphical parameters to customize the directional plot:

- bty: Type of box to be drawn around the plot region. Allowed values are "o" (the default), "l", "7", "c", "u", or "]". The resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box.
- bty.leg: Type of box to be drawn around the legend. Allowed values are "n" (no box, the default) and "o".
- cex: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- cex.axis: Amount by which axis annotations should be scaled, as numeric value.
- cex.lab: Amount by which axis labels should be scaled, as numeric value.
- cex.leg: Amount by which legend text should be scaled, as numeric value.
- col: Vector of colours for the sectoral distributions, with the length of sector number + 1, where the last colour is used for the general profile.
- col.axis: Colour to be used for axis annotations – default is "black".
- col.box: Colour to be used for the box around the plot region (if bty) – default is "black".
- col.lab: Colour to be used for axis labels – default is "black".
- col.leg: Colour to be used for legend text – default is "black".
- col.ticks: Colours for the axis line and the tick marks respectively – default is "black".
- las: Style of axis labels. One of 0 (always parallel to the axis, default), 1 (always horizontal), 2 (always perpendicular to the axis), 3 (always vertical).

- `lty`: Vector of line types, assigned like `col`. See `par` for usage.
- `lwd`: Vector of line widths, assigned like `col`. See `par` for usage.
- `mar`: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot – default is `c(4.5, 4.5, 1, 1)`.
- `mgp`: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is `c(2.5, 0.7, 0)`.
- `pos.leg`: Position of legend – one of `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` or `"center"`. Use `NULL` to hide the legend.
- `xlab`: Alternative label for the x axis.
- `ylab`: Alternative label for the y axis.
- `xlim`: Numeric vector of the x limits of the plot.
- `ylim`: Numeric vector of the y limits.
- `x.intersp`: Horizontal interspacing factor for legend text, as numeric – default is `0.4`.
- `y.intersp`: Vertical line distance for legend text, as numeric – default is `0.8`.

Author(s)

Christian Graul

References

Brower, M., Marcus, M., Taylor, M., Bernadett, D., Filippelli, M., Beaucage, P., Hale, E., Elsholz, K., Doane, J., Eberhard, M., Tensen, J., Ryan, D. (2010) Wind Resource Assessment Handbook. http://www.renewablenrgsystems.com/TechSupport/~/media/Files/PDFs/wind_resource_handbook.ashx

Langreder, W. (2010) Wind Resource and Site Assessment. In: Wei Tong (Ed.), Wind Power Generation and Wind Turbine Design, Chapter 2, p. 49–87, Southampton: WIT Press

Weibull, W. (1951) A Statistical Distribution Function of Wide Applicability. *Journal of Applied Mechanics – Trans. ASME* **18**(3), 293–297

See Also

`mast`

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)


## calculate Weibull parameters
```

```
weibull(mast=neubuerg, v.set=1)  # default

# if only one of v.set and dir.set is given,
# the dataset is assigned to both
weibull(mast=neubuerg, v.set=1)
weibull(mast=neubuerg, dir.set=1)
weibull(mast=neubuerg, dir.set="set1")

# change number of direction sectors
weibull(mast=neubuerg, v.set=1, num.sectors=16)

# data subsets
weibull(mast=neubuerg, v.set=1,
  subset=c("2009-12-01 00:00:00", "2009-12-31 23:50:00"))
weibull(mast=neubuerg, v.set=1,
  subset=c("2010-01-01 00:00:00", NA)) # just 'start' time stamp
weibull(mast=neubuerg, v.set=1,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

# change number of digits and hide results
neubuerg.wb <- weibull(mast=neubuerg, v.set=1, digits=2, print=FALSE)


## plot weibull objects - histogram plot
plot(neubuerg.wb)  # default
plot(neubuerg.wb, type="dir")  # same as above
plot(neubuerg.wb, show.ak=TRUE)  # show parameters in legend

# customize plot
plot(neubuerg.wb, bty="l", bty.leg="l", cex.axis=1.2, cex.lab=1.4, cex.leg=0.9,
  col.axis="darkgray", col.box="darkgray", col.lab="darkgray", col.leg="darkgray",
  col.ticks="darkgray", las=0, leg.text=c("measured", "calculated"),
  mar=c(3,3,0.5,0.5), mgp=c(1.8,0.5,0), pos.leg="right", xlab="velocity [m/s]",
  ylab="frequency [%]", xlim=c(0,20), ylim=c(0,15), x.intersp=1, y.intersp=1)

# customize bars
plot(neubuerg.wb, border="darkgray", breaks=seq(0,21,0.5),
  col="lightgray")

# customize line
plot(neubuerg.wb, line="black", lty="dotdash", lwd=2)


## plot weibull objects - directional plot
plot(neubuerg.wb, type="dir")

# show parameters in legend
plot(neubuerg.wb, type="dir", show.ak=TRUE)

# customize plot
plot(neubuerg.wb, type="dir", bty="l", bty.leg="o", cex.axis=0.8, cex.lab=0.9,
  cex.leg=0.7, col=c(rainbow(12), gray(0.4)), col.axis=gray(0.2), col.box=gray(0.4),
  col.lab=gray(0.2), col.leg=gray(0.2), col.ticks=gray(0.2), las=0,
```

```
  lty=c(rep(3, 12), 1), lwd=c(rep(1, 12), 2), mar=c(3,3,0.5,0.5), mgp=c(1.5,0.5,0),
  pos.leg="right", xlab="velocity [m/s]", ylab="frequency [m/s]", xlim=c(0,15),
  ylim=c(0,25), x.intersp=1, y.intersp=1)

## End(Not run)
```

`winddata` *Example data for bReeze*

Description

This dataset is provided as part of the `bReeze` package for use with the examples in the documantation. It contains measured wind speed and wind direction in a 10 minute interval, collected by a meteorological mast.

Format

Data frame with 36548 observations on the following 17 variables:

`date_time` Date and time of observation as character vector.

`v1_40m_avg` Average wind speed in m/s at a height of 40 m as numeric vector.

`v1_40m_max` Maximum wind speed in m/s at a height of 40 m as numeric vector.

`v1_40m_min` Minimum wind speed in m/s at a height of 40 m as numeric vector.

`v1_40m_std` Standard deviation of wind speed in m/s at a height of 40 m as numeric vector.

`v2_30m_avg` Average wind speed in m/s at a height of 30 m as numeric vector.

`v2_30m_max` Maximum wind speed in m/s at a height of 30 m as numeric vector.

`v2_30m_min` Minimum wind speed in m/s at a height of 30 m as numeric vector.

`v2_30m_std` Standard deviation of wind speed in m/s at a height of 30 m as numeric vector.

`v3_20m_avg` Average wind speed in m/s at a height of 20 m as numeric vector.

`v3_20m_max` Maximum wind speed in m/s at a height of 20 m as numeric vector.

`v3_20m_min` Minimum wind speed in m/s at a height of 20 m as numeric vector.

`v3_20m_std` Standard deviation of wind speed in m/s at a height of 20 m as numeric vector.

`dir1_40m_avg` Average wind direction in degrees from north at a height of 40 m as numeric vector.

`dir1_40m_std` Standard deviation of wind direction in degrees from north at a height of 40 m as numeric vector.

`dir2_30m_avg` Average wind direction in degrees from north at a height of 30 m as numeric vector.

`dir2_30m_std` Standard deviation of wind direction in degrees from north at a height of 30 m as numeric vector.

Examples

```
## Not run:
# load example data
data("winddata", package="bReeze")

# display the structure of the data
str(winddata)

## End(Not run)
```

`windprofile` *Calculate wind profile*

Description

Calculates a wind profile, using on the Hellman exponential law.

Usage

```
windprofile(mast, v.set, dir.set, num.sectors=12,
  method=c("hellman", "loglm", "fixed"), alpha=NULL,
  subset, digits=3, print=TRUE)
pro(mast, v.set, dir.set, num.sectors=12,
  method=c("hellman", "loglm", "fixed"), alpha=NULL,
  subset, digits=3, print=TRUE)

## S3 method for class 'windprofile'
plot(x, sector, measured=TRUE, ...)
```

Arguments

`mast`	Met mast object created by `mast`.
`v.set`	Set(s) to be used for wind speed, specified as set number or set name. The first given dataset is used as reference height. If one single dataset is given, the same Hellman exponent is assumed for each sector and can be specified using `alpha` (see Details for empirical values). If two or more datasets are given, the Hellman exponent is calculated for each sector. If more than two datasets are given, currently only the first two datasets are used.
`dir.set`	Set to be used for wind direction, specified as set number or set name.
`num.sectors`	Number of wind direction sectors as integer value greater 1. Default is `12`.
`method`	Method to be used for the calculation. One of `"hellman"`, `"loglm"` or `"fixed"` (see Details section). Optional argument: `"fixed"` is used if `v.set` is a single dataset, `"hellman"` is used if `v.set` specifies two, `"loglm"` if `v.set` specifies three datasets.

alpha	Hellman exponent – one general exponent or a vector of exponents (one per sector). Optional and only used if the choosen method is "fixed". If alpha is NULL (the default), 0.2 is used as default.
subset	Optional start and end time stamp for a data subset, as string vector c(start, end). The time stamps format shall follow the rules of ISO 8601 international standard, e.g. "2012-08-08 22:55:00".
digits	Number of decimal places to be used for results as numeric value. Default is 3.
print	If TRUE (the default), results are printed directly.
x	Wind profile object created by profile.
sector	Direction sector as integer (sector number) or string (sector code). If missing or NULL, all sectors are plotted. For plotting the general profile only use "all".
measured	If TRUE (the default), measured sector mean wind speeds are added to the plot.
...	Arguments to be passed to methods. For optional graphical parameters see below.

Details

The average wind speed as a function of height above ground gives a site's wind profile. For reasons of cost-efficiency met mast heights are usually below hub heights of modern wind turbines, thus measured wind speeds must be extrapolated by based wind profile. A common method is the Hellman exponential law or power law ("hellman"), defined as:

$$\frac{v_2}{v_1} = \left(\frac{h_2}{h_1}\right)^{\alpha}$$

where v_2 is the wind speed at height h_2, v_1 is the wind speed at height h_1 and α is the Hellman exponent (also power law exponent or shear exponent).

To calculate α, profile uses the inverted equation as:

$$\alpha = \frac{\log\left(\frac{v_2}{v_1}\right)}{\log\left(\frac{h_2}{h_1}\right)}$$

If data is available for two or more heights, a log-linear model fit can be used for the computation of α. In this case the data is logarithmized to allow for fitting a linear model. The models slope is then used as α. Please note: depending on the data this method might result in negative α.

If the wind speed is only known for one height, α must be estimated. α depends on various issues, including roughness and terrain of the site. Some empirical values for temperate climates are:

Site conditions	α
Open water	0.08–0.15
Flat terrain, open land cover	0.16–0.22
Complex terrain with mixed or continuous forest	0.25–0.40
Exposed ridgetops, open land cover	0.10–0.14
Sloping terrain with drainage flows	0.10–0.15

Value

Returns a list of:

profile	Data frame containing alpha and reference wind speed for each direction sector.
h.ref	Reference height of the profile (the height of the first given dataset in v.set).

Optional graphical parameters

The following graphical parameters can optionally be added to customize the plot:

- bty: Type of box to be drawn around the plot region. Allowed values are "o" (the default), "l", "7", "c", "u", or "]". The resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box.
- bty.leg: Type of box to be drawn around the legend. Allowed values are "n" (no box, the default) and "o".
- cex: Amount by which text on the plot should be scaled relative to the default (which is 1), as numeric. To be used for scaling of all texts at once.
- cex.axis: Amount by which axis annotations should be scaled, as numeric value.
- cex.lab: Amount by which axis labels should be scaled, as numeric value.
- cex.leg: Amount by which legend text should be scaled, as numeric value.
- col: Vector of colours, one for each set plotted.
- col.axis: Colour to be used for axis annotations – default is "black".
- col.box: Colour to be used for the box around the plot region (if bty) – default is "black".
- col.lab: Colour to be used for axis labels – default is "black".
- col.leg: Colour to be used for legend text – default is "black".
- col.ticks: Colours for the axis line and the tick marks respectively – default is "black".
- las: Style of axis labels. One of 0 (always parallel to the axis, default), 1 (always horizontal), 2 (always perpendicular to the axis), 3 (always vertical).
- lty: Line type(s) of the profile lines – assigned like col. See par for available line types.
- lwd: Line width(s) of the profile lines – assigned like col. See par for usage.
- mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot – default is c(4, 4, 1, 1).
- mgp: A numerical vector of the form c(label, annotation, line), which gives the margin line for the axis label, axis annotation and axis line. The default is c(2.5, 0.7, 0).
- pos.leg: Position of legend – one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Use NULL to hide the legend.
- xlab: Alternative label for the x axis.
- ylab: Alternative label for the y axis.
- xlim: Numeric vector of the x limits of the plot.
- ylim: Numeric vector of the y limits.
- x.intersp: Horizontal interspacing factor for legend text, as numeric – default is 0.4.
- y.intersp: Vertical line distance for legend text, as numeric – default is 0.8.

Author(s)

Christian Graul

References

Bañuelos-Ruedas, F., Camacho, C.A., Rios-Marcuello, S. (2011) Methodologies Used in the Extrapolation of Wind Speed Data at Different Heights and Its Impact in the Wind Energy Resource Assessment in a Region. In: Gastón O. Suvire (Ed.), Wind Farm – Technical Regulations, Potential Estimation and Siting Assessment, Chapter 4, p. 97–114, InTech

Brower, M., Marcus, M., Taylor, M., Bernadett, D., Filippelli, M., Beaucage, P., Hale, E., Elsholz, K., Doane, J., Eberhard, M., Tensen, J., Ryan, D. (2010) Wind Resource Assessment Handbook. http://www.renewablenrgsystems.com/TechSupport/~/media/Files/PDFs/wind_resource_handbook.ashx

International Electrotechnical Commission (2005) IEC 61400-12 Wind Turbines – Part 12-1: Power Performance Measurements of Electricity Producing Wind Turbines. IEC Standard

See Also

mast

Examples

```
## Not run:
## load and prepare data
data("winddata", package="bReeze")
set40 <- set(height=40, v.avg=winddata[,2], dir.avg=winddata[,14])
set30 <- set(height=30, v.avg=winddata[,6], dir.avg=winddata[,16])
set20 <- set(height=20, v.avg=winddata[,10], v.std=winddata[,13])
ts <- timestamp(timestamp=winddata[,1])
neubuerg <- mast(timestamp=ts, set40, set30, set20)
neubuerg <- clean(mast=neubuerg)


## calculate profile
# create profile based on one height
windprofile(mast=neubuerg, v.set=1, dir.set=1) # default alpha=0.2
windprofile(mast=neubuerg, v.set=1, dir.set=1, alpha=0.15)

# calculate profile based on two heights
windprofile(mast=neubuerg, v.set=c(1,2), dir.set=1)
windprofile(mast=neubuerg, v.set=c(1,3), dir.set=1)
# same as above
windprofile(mast=neubuerg, v.set=c("set1", "set3"), dir.set="set1")

# calculate profile based on three heights
windprofile(mast=neubuerg, v.set=c(1,2,3), dir.set=1)

# change the method used for computation
# note: negative alphas!
windprofile(mast=neubuerg, v.set=c(1,2), dir.set=1, method="loglm")
```

```
# change number of direction sectors
windprofile(mast=neubuerg, v.set=c(1,2), dir.set=1, num.sectors=8)

# data subsets
windprofile(mast=neubuerg, v.set=1, dir.set=1,
  subset=c("2009-12-01 00:00:00", "2009-12-31 23:50:00"))
windprofile(mast=neubuerg, v.set=c(1,2), dir.set=1,
  subset=c("2010-01-01 00:00:00", NA)) # just 'start' time stamp
windprofile(mast=neubuerg, v.set=c(1:3), dir.set=1,
  subset=c(NA, "2009-12-31 23:50:00")) # just 'end' time stamp

# change number of digits and hide results
windprofile(mast=neubuerg, v.set=1, dir.set=1, digits=2)
neubuerg.wp <- windprofile(mast=neubuerg, v.set=1, dir.set=1, print=FALSE)
neubuerg.wp


## plot profile objects
plot(neubuerg.wp)  # default
plot(neubuerg.wp, measured=FALSE)  # omit 'measured' points

# plot only one sector
plot(neubuerg.wp, sector=3) # ENE by sector number
plot(neubuerg.wp, sector="ene") # ENE by sector code
plot(neubuerg.wp, sector="all") # general profile

# customize plot
plot(neubuerg.wp, bty="l", bty.leg="o", cex.axis=0.8,
  cex.lab=0.9, cex.leg=0.7, col=rainbow(13), col.axis=gray(0.2),
  col.box=gray(0.2), col.lab=gray(0.2), col.leg=gray(0.2),
  col.ticks=gray(0.2), las=0, lty=c(rep(3,12),1),
  lwd=c(rep(1.2,12), 1.7), mar=c(3,3,0.5,0.5), mgp=c(2,0.7,0),
  pos.leg="right", xlab="velocity [m/s]", ylab="height [m]",
  xlim=c(0,11), ylim=c(0,150), x.intersp=1, y.intersp=1.2)

## End(Not run)
```

Index