Package 'bioRad'

December 14, 2018

Title Biological Analysis and Visualization of Weather Radar Data

Version 0.4.0

Description Extract, visualize and summarize aerial movements of birds and insects from weather radar data.

License MIT + file LICENSE

URL https://github.com/adokter/bioRad, https://adokter.github.io/bioRad

BugReports https://github.com/adokter/bioRad/issues

biocViews

Depends R (>= 2.10)

Imports curl, fields, ggmap, ggplot2, graphics, methods, raster, rgdal, rhdf5, sp, stats, utils, maptools

Suggests knitr, testthat

LazyData true

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Adriaan M. Dokter [aut, cre] (<https://orcid.org/0000-0001-6573-066X>), Peter Desmet [aut] (<https://orcid.org/0000-0002-8442-8025>), Stijn Van Hoey [aut] (<https://orcid.org/0000-0001-6413-3185>), Jurriaan Spaaks [ctb], Lourens Veen [ctb], Liesbeth Verlinden [ctb] (<https://orcid.org/0000-0003-1744-9325>), Hidde Leijnse [ctb] (<https://orcid.org/0000-0001-7835-4480>)

Maintainer Adriaan M. Dokter <amd427@cornell.edu>

Repository CRAN

Date/Publication 2018-12-14 21:30:04 UTC

R topics documented:

`as.data.frame.vp` *Convert a vertical profile (*`vp`*) to a Data Frame*

Description

Converts a vertical profile to a Data Frame, and optionally adds information on sunrise/sunset, day/night and derived quantities like migration traffic rates.

Usage

```
## S3 method for class 'vp'
as.data.frame(x, row.names = NULL, optional = FALSE,
  quantities = names(x$data), suntime = TRUE, geo = TRUE,
  elev = -0.268, lat = NULL, lon = NULL, ...)
```

Arguments

`x`	An object of class `vp`.
`row.names`	`NULL` or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	If `FALSE` then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names and are not duplicated.
`quantities`	An optional character vector with the names of the quantities to include as columns in the data frame.
`suntime`	Logical, when `TRUE`, adds sunrise/sunset and day/night information to each row.
`geo`	Logical, when `TRUE`, adds latitude, longitude and antenna height of the radar to each row.
`elev`	Sun elevation in degrees, see sunrise/sunset.
`lat`	Radar latitude in decimal degrees. When set, overrides the latitude stored in `x` in sunrise/sunset calculations
`lon`	Radar longitude in decimal degrees. When set, overrides the longitude stored in `x` in sunrise/sunset calculations.
`...`	Additional arguments to be passed to or from methods.

Details

Note that only the "dens" quantity is thresholded for radial velocity standard deviation by sd_vvp_threshold. Note that this is different from the default plot.vp, plot.vpts and get_quantity.vp functions, where quantities "eta", "dbz", "ff", "u", "v", "w", "dd" are all thresholded by sd_vvp_threshold

Value

An object of class `data.frame`.

Examples

```
# load an example vertical profile time series object
data(example_vp)

# convert the object to a data.frame
df <- as.data.frame(example_vp)

# do not compute sunrise/sunset information
df <- as.data.frame(example_vp, suntime = FALSE)

# override the latitude/longitude information stored in the object
# when calculating sunrise / sunset
df <- as.data.frame(example_vp, suntime = TRUE, lat = 50, lon = 4)
```

`as.data.frame.vpts` *Convert a time series of vertical profiles (*`vpts`*) to a data frame*

Description

Converts vertical profile time series (objects of class `vpts`) to a data Frame, and optionally adds information on sunrise/sunset, day/night and derived quantities like migration traffic rates.

Usage

```
## S3 method for class 'vpts'
as.data.frame(x, row.names = NULL, optional = FALSE,
  quantities = names(x$data), suntime = TRUE, geo = TRUE,
  elev = -0.268, lat = NULL, lon = NULL, ...)
```

Arguments

`x`	An object of class `vpts`.
`row.names`	`NULL` or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	If `FALSE` then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names and are not duplicated.

quantities	An optional character vector with the names of the quantities to include as columns in the data frame.
suntime	Logical, when TRUE, adds sunrise/sunset and day/night information to each row.
geo	Logical, when TRUE, adds latitude, longitude and antenna height of the radar to each row.
elev	Sun elevation in degrees, see sunrise/sunset.
lat	Radar latitude in decimal degrees. When set, overrides the latitude stored in x in sunrise/sunset calculations.
lon	Radar longitude in decimal degrees. When set, overrides the longitude stored in x in sunrise/sunset calculations.
...	Additional arguments to be passed to or from methods.

Details

Note that only the 'dens' quantity is thresholded for radial velocity standard deviation by sd_vvp_threshold. Note that this is different from the default plot.vp, plot.vpts and get_quantity.vp functions, where quantities "eta", "dbz", "ff", "u", "v", "w", "dd" are all thresholded by sd_vvp_threshold.

Value

An object of class data.frame.

Examples

```
# load an example vertical profile time series object
data(example_vpts)

# convert the object to a data.frame
df <- as.data.frame(example_vpts)

# do not compute sunrise/sunset information
df <- as.data.frame(example_vpts, suntime = FALSE)

# override the latitude/longitude information stored in the object
# when calculating sunrise / sunset
df <- as.data.frame(example_vpts, suntime = TRUE, lat = 50, lon = 4)
```

---

beam_height *Calculate radar beam height*

Description

Calculates the height of a radar beam as a function of elevation and range, assuming the beam is emitted at surface level.

Usage

```
beam_height(range, elev, k = 4/3, lat = 35, re = 6378, rp = 6357)
```

Arguments

range	numeric. Range (distance from the radar antenna) in km.
elev	numeric. Elevation in degrees.
k	Standard refraction coefficient.
lat	Geodetic latitude in degrees.
re	Earth equatorial radius in km.
rp	Earth polar radius in km.

Details

To account for refraction of the beam towards the earth's surface, an effective earth's radius of k * (true radius) is assumed, with k = 4/3.

The earth's radius is approximated as a point on a spheroid surface, with `re` the longer equatorial radius, and `rp` the shorter polar radius. Typically uncertainties in refraction coefficient are relatively large, making oblateness of the earth and the dependence of earth radius with latitude only a small correction. Using default values assumes an average earth's radius of 6371 km.

Value

numeric. Beam height in km.

---

beam_width *Calculate radar beam width*

Description

Calculates the width of a radar beam as a function of range and beam angle.

Usage

```
beam_width(range, beam_angle = 1)
```

Arguments

range	numeric. Range (distance from the radar antenna) in km.
beam_angle	numeric. Beam opening angle in degrees, typically the the angle between the half-power (-3 dB) points of the main lobe

Value

numeric. Beam width in m.

bind_into_vpts *Bind vertical profiles (vp) into time series (vpts)*

Description

Binds vertical profiles (vp) into a vertical profile time series (vpts), sorted in time. Can also bind multiple vpts of a single radar into one vpts.

Usage

```
bind_into_vpts(x, ...)

## S3 method for class 'vp'
bind_into_vpts(...)

## S3 method for class 'list'
bind_into_vpts(x, ...)

## S3 method for class 'vpts'
bind_into_vpts(..., attributes_from = 1)
```

Arguments

x	A vp, vpts or a vector of these.
...	A vp, vpts or a vector of these.
attributes_from	integer. Which vpts to copy attributes from (default: first).

Value

A vpts for a single radar or a list of vpts for multiple radars. Input vp are sorted in time in the output vpts.

Methods (by class)

- vp: Bind multiple vp into a vpts. If vp for multiple radars are provided, a list is returned containing a vpts for each radar.
- list: Bind multiple vp objects into a vpts. If data for multiple radars is provided, a list is returned containing a vpts for each radar.
- vpts: Bind multiple vpts into a single vpts. Requires the input vpts to be from the same radar.

Examples

```
# load example time series of vertical profiles:
data(example_vpts)

# split the vpts into two separate time series, one containing profile 1-10,
# and a second containing profile 11-20:
vpts1 <- example_vpts[1:10]
vpts2 <- example_vpts[11:20]

# use bind_into_vpts to bind the two together:
vpts1and2 <- bind_into_vpts(vpts1, vpts2)

# verify that the binded vpts now has 20 profiles, 10 from vpts1 and 10 from
# vpts2:
summary(vpts1and2)

# extract two profiles:
vp1 <- example_vpts[1]
vp1
vp2 <- example_vpts[2]
vp2

# bind the two profiles back into a vpts:
bind_into_vpts(vp1, vp2)
```

---

c.vp *Concatenate vertical profiles (vp) into a list of vertical profiles*

Description

Concatenate vertical profiles (vp) into a list of vertical profiles

Usage

```
## S3 method for class 'vp'
c(...)
```

Arguments

...	objects of class vp

Value

an object of class list

calculate_vp *Calculate a vertical profile (*vp*) from a polar volume (*pvol*)*

Description

Calculates a vertical profile of biological scatterers (vp) from a polar volume (pvol) using the algorithm vol2bird (Dokter et al. 2011).

Usage

```
calculate_vp(pvolfile, vpfile = "", pvolfile_out = "",
  autoconf = FALSE, verbose = FALSE, mount = dirname(pvolfile),
  sd_vvp_threshold = 2, rcs = 11, dual_pol = FALSE, rho_hv = 0.95,
  elev_min = 0, elev_max = 90, azim_min = 0, azim_max = 360,
  range_min = 5000, range_max = 25000, n_layer = 20L,
  h_layer = 200, dealias = TRUE, nyquist_min = if (dealias) 5 else
  25, dbz_quantity = "DBZH")
```

Arguments

pvolfile	A radar file containing a radar polar volume, either in ODIM format, which is the implementation of the OPERA data information model in HDF5 format, or a format supported by the RSL library.
vpfile	character. Filename for the vertical profile to be generated in ODIM HDF5 format (optional).
pvolfile_out	character. Filename for the polar volume to be generated in ODIM HDF5 format (optional, e.g. for converting RSL formats to ODIM).
autoconf	logical. When TRUE, default optimal configuration settings are selected automatically, and other user settings are ignored.
verbose	logical. When TRUE, pipe Docker stdout to R console. On Windows always TRUE.
mount	character. String with the mount point (a directory path) for the Docker container.
sd_vvp_threshold	numeric. Lower threshold in radial velocity standard deviation (profile quantity sd_vvp) in m/s. Biological signals with sd_vvp < sd_vvp_threshold are set to zero.
rcs	numeric. Radar cross section per bird in cm^2.
dual_pol	logical. When TRUE use dual-pol mode, in which meteorological echoes are filtered using the correlation coefficient rho_hv. When FALSE use single polarization mode based only on reflectivity and radial velocity quantities.
rho_hv	numeric. Lower threshold in correlation coefficient used to filter meteorological scattering.
elev_min	numeric. Minimum scan elevation in degrees.

elev_max	numeric. Maximum scan elevation in degrees.
azim_min	numeric. Minimum azimuth in degrees clockwise from north.
azim_max	numeric. Maximum azimuth in degrees clockwise from north.
range_min	numeric. Minimum range in km.
range_max	numeric. Maximum range in km.
n_layer	numeric. Number of altitude layers in the profile.
h_layer	numeric. Width of altitude layers in meter.
dealias	logical. Whether to dealias radial velocities; this should typically be done when the scans in the polar volume have low Nyquist velocities (below 25 m/s).
nyquist_min	numeric. Minimum Nyquist velocity of scans in m/s for scans to be included in the analysis.
dbz_quantity	character. One of the available reflectivity factor quantities in the ODIM radar data format, e.g. DBZH, DBZV, TH, TV.

Details

Requires a running Docker daemon.

Common arguments set by users are `pvolfile`, `vpfile`, `autoconf` and `mount`.

Turn on `autoconf` to automatically select the optimal parameters for a given radar file. The default for C-band data is to apply rain-filtering in single polarization mode, as well as dual polarization mode when available.

The default for S-band data is to apply precipitation filtering in dual-polarization mode.

Arguments that sometimes require non-default values are: `rcs`, `sd_vvp_threshold`, `range_max`, `dual_pol`, `dealias`.

Other arguments are typically left at their defaults.

`azim_min` and `azim_max` only affects reflectivity-derived estimates in the profile (DBZH,eta,dens), not radial-velocity derived estimates (u, v, w, ff, dd, sd_vvp), which are estimated on all azimuths at all times. `azim_min`, `azim_max` may be set to exclude an angular sector with high ground clutter.

`range_max` may be extended up to 40,000 m for volumes with low elevations only, in order to extend coverage to higher altitudes.

For altitude layers with a VVP-retrieved radial velocity standard deviation value below the threshold `sd_vvp_threshold`, the bird density `dens` is set to zero (see vertical profile vp class). This threshold might be dependent on radar processing settings. Results from validation campaigns so far indicate that 2 m/s is the best choice for this parameter for most weather radars. The algorithm has been tested and developed for altitude layers with `h_layer` = 200 m. Smaller widths are not recommended as they may cause instabilities of the volume velocity profiling (VVP) and dealiasing routines, and effectively lead to pseudo-replicated altitude data, since altitudinal patterns smaller than the beam width cannot be resolved.

The default radar cross section (11 cm^2) corresponds to the average value found by Dokter et al. in a calibration campaign of a full migration autumn season in western Europe at C-band. It's value may depend on radar wavelength. `rcs` will scale approximately $M^{2/3}$ with `M` the bird's mass.

Using default values of `range_min` and `range_max` is recommended. Ranges closer than 5 km tend to be contaminated by ground clutter, while range gates beyond 25 km become too wide to resolve the default altitude layer width of 200 meter (see beam_width).

For dealiasing, the torus mapping method by Haase et al. is used.

At S-band (radar wavelength ~ 10 cm), currently only `dual_pol=TRUE` mode is recommended.

On repeated calls of `calculate_vp`, the Docker container mount can be recycled from one call to the next if subsequent calls share the same `mount` argument. Re-mounting a Docker container takes time, therefore it is advised to choose a mountpoint that is a parent directory of all volume files to be processed, such that `calculate_vp` calls are as fast as possible.

Value

A vertical profile object of class vp. When defined, output files `vpfile` and `pvolfile_out` are saved to disk.

References

- Haase, G. and Landelius, T., 2004. Dealiasing of Doppler radar velocities using a torus mapping. Journal of Atmospheric and Oceanic Technology, 21(10), pp.1566-1573.
- Bird migration flight altitudes studied by a network of operational weather radars, Dokter et al., J. R. Soc. Interface 8 (54), pp. 30–43, 2011. https://doi.org/10.1098/rsif.2010.0116

Examples

```
# locate example polar volume file:
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")

# copy to a home directory with read/write permissions:
file.copy(pvolfile, "~/volume.h5")

# calculate the profile:
## Not run:
profile <- calculate_vp("~/volume.h5")

## End(Not run)

# clean up:
file.remove("~/volume.h5")
```

---

check_docker *Check if Docker is running*

Description

Checks that Docker daemon is running correctly on the local system

Usage

```
check_docker(verbose = TRUE)
```

Arguments

verbose	logical which indicates whether to print test results to R console. On Windows always TRUE.

Value

0 upon success, otherwise an error code.

---

check_night *Check if it is night at a given time and place*

Description

Checks if it is night (TRUE/FALSE) for a combination of latitude, longitude, date and sun elevation. When used on a bioRad object (pvol, vp, vpts) this information is extracted from the bioRad object directly.

Usage

```
check_night(x, ..., elev = -0.268)

## Default S3 method:
check_night(x, lon, lat, ..., tz = "UTC",
  elev = -0.268)

## S3 method for class 'vp'
check_night(x, ..., elev = -0.268)

## S3 method for class 'list'
check_night(x, ..., elev = -0.268)

## S3 method for class 'vpts'
check_night(x, ..., elev = -0.268)

## S3 method for class 'pvol'
check_night(x, ..., elev = -0.268)
```

Arguments

x	pvol, vp or vpts, or a date inheriting from class POSIXct or a string interpretable by as.POSIXct.
...	optional lat,lon arguments.
elev	numeric. Sun elevation in degrees.
lon	numeric. Longitude in decimal degrees.
lat	numeric. Latitude in decimal degrees.
tz	character. Time zone. Ignored when date already has an associated time zone

Details

The angular diameter of the sun is about 0.536 degrees, therefore the moment of sunrise/sunset corresponds to half that elevation at -0.268 degrees.

`check_night()` evaluates to `FALSE` when the sun has a higher elevation than parameter `elev`, otherwise `TRUE`.

Approximate astronomical formula are used, therefore the day/night transition may be off by a few minutes.

Value

`TRUE` when night, `FALSE` when day, `NA` if unknown (either datetime or geographic location missing). For `vpts` a vector of `TRUE`/`FALSE` values is returned.

Examples

```
# check if it is night at UTC midnight in the Netherlands on January 1st:
check_night("2016-01-01 00:00", 5, 53)

# check on bioRad objects directly:
check_night(example_vp)
check_night(example_vpts)
```

`composite_ppi` *Create a composite of multiple plan position indicators (`ppi`)*

Description

Combines multiple plan position indicators (`ppi`) into a single `ppi`. Can be used to make a composite of `ppi`'s from multiple radars.

Usage

```
composite_ppi(x, param = "DBZH", dim = c(100, 100))
```

Arguments

`x`	A list of `ppi`.
`param`	Scan parameter to composite.
`dim`	integer. Vector with number of cells in each spatial dimension.

Details

The latitude/longitude of the returned `ppi` use the WGS84 datum.

Value

A `ppi`.

Examples

```
# load the example polar scan:
data(example_scan)
# to be written ...
```

`dbz_to_eta` *Convert reflectivity factor to reflectivity*

Description

Convert reflectivity factor to reflectivity

Usage

```
dbz_to_eta(dbz, wavelength, K = 0.93)
```

Arguments

`dbz`	reflectivity factor in dBZ
`wavelength`	radar wavelength in cm
`K`	norm of the complex refractive index of water

Value

reflectivity in cm^2/km^3

`download_basemap` *Download a basemap for* `map(ppi)`

Description

Downloads a Google Maps, OpenStreetMap, Stamen Maps or Naver Map base layer map using get_map.

Usage

```
download_basemap(x, verbose = TRUE, zoom, alpha = 1,
  source = "stamen", ...)
```

Arguments

x	An object of class ppi.
verbose	Logical, whether to print information to console.
zoom	Zoom level (optional), see get_map. An integer from 3 (continent) to 21 (building). By default the zoom level matching the ppi extent is selected automatically.
alpha	Transparency of the basemap (0-1).
source	String identifying which map service should be used: "google", "osm" or "stamen"
...	Arguments to pass to get_map function. Note arguments maptype and source for selection of different types of basemaps.

Examples

```
# load an example scan:
data(example_scan)
# print summary info for the scan:
example_scan
# make ppi for the scan
ppi <- project_as_ppi(example_scan)
# grab a basemap that matches the extent of the ppi:
## Not run:
basemap <- download_basemap(ppi)
# map the reflectivity quantity of the ppi onto the basemap:
map(ppi, map = basemap, param = "DBZH")
# download a different type of basemap, e.g. satellite imagery:
# see get_map() in ggmap library for full documentation of options
basemap <- download_basemap(ppi, maptype = "satellite")
# map the radial velocities onto the satellite imagery:
map(ppi, map = basemap, param = "VRADH")

## End(Not run)
```

---

download_vpfiles *Download vertical profile (vp) files from the ENRAM data repository*

Description

Download and unzip a selection of vertical profile (vp) files from the ENRAM data repository, where these are stored as monthly zips per radar.

Usage

```
download_vpfiles(date_min, date_max, radars, directory = ".",
  overwrite = FALSE)
```

Arguments

date_min	character. YYYY-MM-DD start date of file selection. Days will be ignored.
date_max	character. YYYY-MM-DD end date of file selection. Days will be ignored.
radars	character (vector). 5-letter country/radar code(s) (e.g. "bejab") of radars to include in file selection.
directory	character. Path to local directory where files should be downloaded and unzipped.
overwrite	logical. TRUE for re-downloading and overwriting previously downloaded files of the same names.

Examples

```
# Download data from radars "bejab" and "bewid", even if previously
# downloaded (overwrite = TRUE). Will successfully download 2016-10 files,
# but show 404 error for 2016-11 files (as these are not available).
## Not run:
download_vpfiles(
  date_min = "2016-10-01",
  date_max = "2016-11-30",
  radar = c("bejab", "bewid"),
  directory = "my_data",
  overwrite = TRUE
)

## End(Not run)
```

---

eta_to_dbz *Convert reflectivity to reflectivity factor*

Description

Convert reflectivity to reflectivity factor

Usage

```
eta_to_dbz(eta, wavelength, K = 0.93)
```

Arguments

eta	reflectivity in cm^2/km^3
wavelength	radar wavelength in cm
K	norm of the complex refractive index of water

Value

reflectivity factor in dBZ

example_scan *Example object of class* scan

Description

Example of a scan object with name example_scan.

Usage

```
example_scan
```

Format

An object of class scan of dimension 5 x 480 x 360.

Examples

```
# get summary of example scan:
summary(example_scan)

# example_scan was created with:
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")
pvol <- read_pvolfile(pvolfile)
example_scan <- pvol$scans[[1]]
## Not run:
save(example_scan, file = "data/example_scan.rda")

## End(Not run)
```

example_vp *Example object of class* vp

Description

Example of a vp object with name example_vp. Can be created with calculate_vp or read from file with read_vpfiles.

Usage

```
example_vp
```

Format

An object of class vp with 25 rows and 16 columns.

Examples

```
# get summary of example vp:
summary(example_vp)

# example_vp was created with:
vpfile <- system.file("extdata", "profile.h5", package = "bioRad")
example_vp <- read_vpfiles(vpfile)
## Not run:
save(example_vp, file = "data/example_vp.rda")

## End(Not run)
```

`example_vpts` *Example object of class* `vpts`

Description

Example of a `vpts` object (a time series of vertical profiles) with name `example_vpts`.

Usage

```
example_vpts
```

Format

An object of class `vpts` of dimension 25 x 1934 x 15.

Examples

```
# get summary of example vpts:
summary(example_vpts)

# example_vpts was created with:
## Not run:
vptsfile <- system.file("extdata", "vpts.txt.zip", package = "bioRad")
unzip(vptsfile, exdir = (dirname(vptsfile)), junkpaths = T)
vptsfile <- substr(vptsfile, 1, nchar(vptsfile) - 4)
example_vpts <- read_vpts(vptsfile, radar = "KBGM", wavelength = "S")
rcs(example_vpts) <- 11
sd_vvp_threshold(example_vpts) <- 2
example_vpts$attributes$where$lat <- 42.2
example_vpts$attributes$where$lon <- -75.98
save(example_vpts, file = "data/example_vpts.rda", compress = "xz")

## End(Not run)
```

get_elevation_angles *Get elevation angles of a polar volume (*pvol*) or scan (*scan*)*

Description

Gives the elevation angle of a single scan, or the elevation angles of all scans within a polar volume

Usage

```
get_elevation_angles(x)

## S3 method for class 'pvol'
get_elevation_angles(x)

## S3 method for class 'scan'
get_elevation_angles(x)

## S3 method for class 'param'
get_elevation_angles(x)
```

Arguments

x	A pvol or scan object.

Value

elevation in degrees

Methods (by class)

- pvol: Elevation angles of all scans in a polar volume.
- scan: Elevation angle of a scan.
- param: Elevation angle of a scan parameter.

Examples

```
# load a polar volume
pvol <- system.file("extdata", "volume.h5", package = "bioRad")
vol <- read_pvolfile(pvol)
# elevations for the scans in the volume
get_elevation_angles(vol)
# extract the first scan:
scan <- vol$scans[[1]]
# elevation angle of the scan:
get_elevation_angles(scan)
```

get_odim_object_type *Check the ODIM data class of a polar volume file*

Description

Checks which data class is contained in ODIM HDF5 file

Usage

```
get_odim_object_type(file)
```

Arguments

file	A string containing a file name.

Value

character string pvol for polar volume, vp for vertical profile, otherwise NA

Examples

```
# locate a polar volume file
pvol <- system.file("extdata", "volume.h5", package = "bioRad")
get_odim_object_type(pvol) # > "pvol"
```

get_param *Get a scan parameter (*param*) from a scan (*scan*)*

Description

Get a scan parameter (param) from a scan (scan)

Usage

```
get_param(x, param)
```

Arguments

x	An object of class scan.
param	a scan parameter

Value

An object of class 'param'.

Examples

```
# we will extract a scan parameter from the example scan object:
example_scan
# extract the VRADH scan parameter
myparam <- get_param(example_scan, "VRADH")
```

`get_quantity` *Get a quantity of a vertical profile (*`vp`*) or time series of vertical profiles (*`vpts`*)*

Description

Get a quantity of a vertical profile (`vp`) or time series of vertical profiles (`vpts`)

Usage

```
get_quantity(x, quantity)

## S3 method for class 'vp'
get_quantity(x, quantity = "dens")

## S3 method for class 'list'
get_quantity(x, quantity = "dens")

## S3 method for class 'vpts'
get_quantity(x, quantity = "dens")
```

Arguments

`x`	A vp or vpts object.
`quantity`	A profile quantity, one of: • `"HGHT"` • `"u"` • `"v"` • `"w"` • `"ff"` • `"dd"` • `"sd_vvp"` • `"gap"` • `"dbz"` • `"eta"` • `"dens"` • `"DBZH"` • `"n"` • `"n_all"` • `"n_dbz"` • `"n_dbz_all"`

Details

This function grabs any of the data quantities stored in vp or vpts objects. See the documentation of the vertical profile vp class for a description of each of these quantities.

Value

class vp: a named vector for the requested quantity.

class list: a list of a named vectors for the requested quantity.

class vpts: a (height x time) matrix of the requested quantity.

---

get_scan *Get a scan (scan) from a polar volume (pvol)*

Description

Get a scan (scan) from a polar volume (pvol)

Usage

```
get_scan(x, elev)
```

Arguments

x	An object of class pvol.
elev	Elevation angle.

Details

The function returns the scan with elevation angle closest to elev.

Value

An object of class 'scan'.

Examples

```
# locate example volume file:
pvol <- system.file("extdata", "volume.h5", package = "bioRad")
# load the file:
vol <- read_pvolfile(pvol)
# extract the scan at 3 degree elevation:
myscan <- get_scan(vol, 3)
```

integrate_profile *Vertically integrate profiles (*vp *or* vpts*) to an integrated profile (*vpi*)*

Description

Performs a vertical integration of density, reflectivity and migration traffic rate, and a vertical averaging of ground speed and direction weighted by density.

Usage

```
integrate_profile(x, alt_min, alt_max, alpha = NA, interval_max = Inf)

## S3 method for class 'vp'
integrate_profile(x, alt_min = 0, alt_max = Inf,
  alpha = NA, interval_max = Inf)

## S3 method for class 'list'
integrate_profile(x, alt_min = 0, alt_max = Inf,
  alpha = NA, interval_max = Inf)

## S3 method for class 'vpts'
integrate_profile(x, alt_min = 0, alt_max = Inf,
  alpha = NA, interval_max = Inf)
```

Arguments

x	A vp or vpts object.
alt_min	Minimum altitude in m.
alt_max	Maximum altitude in m.
alpha	Migratory direction in clockwise degrees from north.
interval_max	Maximum time interval belonging to a single profile in seconds. Traffic rates are set to zero at times t for which no profiles can be found within the period t-interval_max/2 to t+interval_max/2. Ignored for single profiles of class vp.

Details

Available quantities: The function generates a specially classed data frame with the following quantities:

datetime POSIXct date of each profile in UTC

vid Vertically Integrated Density in individuals/km^2. vid is a surface density, whereas dens in vp objects is a volume density.

vir Vertically Integrated Reflectivity in cm^2/km^2

mtr Migration Traffic Rate in individuals/km/h

rtr Reflectivity Traffic Rate in cm^2/km/h

`mt` Migration Traffic in individuals/km

`rt` Reflectivity Traffic in cm^2/km

`ff` Horizontal ground speed in m/s

`dd` Horizontal ground speed direction in degrees

`u` Ground speed component west to east in m/s

`v` Ground speed component north to south in m/s

`HGHT` Height above sea level in m

Vertically integrated density and reflectivity are related according to $vid = vir/rcs(x)$, with rcs the assumed radar cross section per individual. Similarly, migration traffic rate and reflectivity traffic rate are related according to $mtr = rtr/rcs(x)$

Migration traffic rate (mtr) and reflectivity traffic rate (rtr): Migration traffic rate (mtr) for an altitude layer is a flux measure, defined as the number of targets crossing a unit of transect per hour.

Column mtr of the output dataframe gives migration traffic rates in individuals/km/hour.

The transect direction is set by the angle `alpha`. When `alpha=NA`, the transect runs perpendicular to the measured migratory direction. `mtr` then equals the number of crossing targets per km transect per hour, for a transect kept perpendicular to the measured migratory movement at all times and altitudes. In this case `mtr` is always a positive quantity, defined as:

$$mtr = \sum_i dens_i ff_i \Delta h$$

with the sum running over all altitude layers between `alt_min` and `alt_max`, $dens_i$ the bird density, ff_i the ground speed at altitude layer i, and Δh the altitude layer width.

If `alpha` is given a numeric value, the transect is taken perpendicular to the direction `alpha`, and the number of crossing targets per hour per km transect is calculated as:

$$mtr = \sum_i dens_i ff_i \cos(dd_i - alpha) \Delta h$$

with dd_i the migratory direction at altitude i.

Note that this equation evaluates to the previous equation when `alpha` equals dd_i.

In this definition `mtr` is a traditional flux into a direction of interest. Targets moving into the direction `alpha` contribute positively to `mtr`, while targets moving in the opposite direction contribute negatively to `mtr`. Therefore `mtr` can be both positive or negative, depending on the definition of alpha.

Formula for reflectivity traffic rate `rtr` are found by replacing `dens` with `eta` in the formula for `mtr`. Reflectivity traffic rate gives the cross-sectional area passing the radar per km transect perpendicular to the migratory direction per hour. `rtr` values are conditional on settings of rcs, while `mtr` values are not.

Migration traffic (mt) and reflectivity traffic (rt): Migration traffic is calculated by time-integration of migration traffic rates. Migration traffic gives the number of individuals that have passed per km perpendicular to the migratory direction at the position of the radar for the full period of the time series within the specified altitude band.

Reflectivity traffic is calculated by time-integration of reflectivity traffic rates. Reflectivity traffic gives the total cross-sectional area that has passed per km perpendicular to the migratory direction at the position of the radar for the full period of the time series within the specified altitude band.

`rt` values are conditional on settings of rcs, while `mt` values are not.

Columnns mt and rt in the output dataframe provides migration traffic as a numeric value equal to migration traffic and reflectivity traffic from the start of the time series up till the moment of the time stamp of the respective row.

Value

an object of class `vpi`, a data frame with vertically integrated profile quantities

Methods (by class)

- `vp`: Vertically integrate a vertical profile.
- `list`: Vertically integrate a list of vertical profiles.
- `vpts`: Vertically integrate a time series of vertical profiles.

Examples

```
# MTR for a single vertical profile
integrate_profile(example_vp)

# MTRs for a list of vertical profiles
integrate_profile(c(example_vp, example_vp))

# MTRs for a time series of vertical profiles
# load example data:
data(example_vpts)
example_vpts
# print migration traffic rates
vpi <- integrate_profile(example_vpts)
# plot migration traffic rates for the full air column
plot(example_vpts)
# plot migration traffic rates for altitudes > 1 km above sea level
plot(integrate_profile(example_vpts, alt_min = 1000))
# plot the (cumulative) migration traffic
plot(integrate_profile(example_vpts), quantity = "mt")
```

`is.pvolfile` *Check if a local file is a polar volume (*`pvol`*)*

Description

Checker whether a file is a polar volume that can be read with package **bioRad**

Usage

```
is.pvolfile(file, filename = NULL)
```

Arguments

file	A string containing a file name.
filename	Deprecated argument, use file instead.

Value

TRUE when file is a polar volume in readable format, otherwise FALSE

Examples

```
volume <- system.file("extdata", "volume.h5", package = "bioRad")
is.pvolfile(volume) # > TRUE
```

---

is.vpfile *Check if a local file is a vertical profile (*vp*)*

Description

Checker whether a file is a vertical profile that can be read with package **bioRad**

Usage

```
is.vpfile(file, filename = NULL)
```

Arguments

file	A string containing a filename
filename	Deprecated argument, use file instead.

Value

TRUE when filename is a vertical profile, otherwise FALSE

Examples

```
profile <- system.file("extdata", "profile.h5", package = "bioRad")
is.vpfile(profile) # > TRUE
```

map *Map a plan position indicator (*ppi*)*

Description

Plot a ppi on a Google Maps, OpenStreetMap, Stamen Maps or Naver Map base layer map using ggmap

Usage

```
map(x, ...)

## S3 method for class 'ppi'
map(x, map, param, alpha = 0.7, xlim, ylim, zlim = c(-20,
  20), ratio, radar_size = 3, radar_color = "red", n_color = 1000,
  radar.size = 3, radar.color = "red", n.color = 1000, ...)
```

Arguments

x	An object of class ppi.
...	Arguments passed to low level ggmap function.
map	The basemap to use, result of a call to download_basemap.
param	The scan parameter to plot.
alpha	Transparency of the data, value between 0 and 1.
xlim	Range of x values to plot (degrees longitude), as atomic vector of length 2.
ylim	Range of y values to plot (degrees latitude), as an atomic vector of length 2.
zlim	The range of values to plot.
ratio	Aspect ratio between x and y scale, by default $1/cos(latituderadar * pi/180)$.
radar_size	Size of the symbol indicating the radar position.
radar_color	Color of the symbol indicating the radar position.
n_color	The number of colors (>=1) to be in the palette.
radar.size	Deprecated argument, use radar_size instead.
radar.color	Deprecated argument, use radar_color instead.
n.color	Deprecated argument, use n_color instead.

Details

Available scan parameters for mapping can by printed to screen by summary(x). Commonly available parameters are:

"DBZH", "DBZ" (Logged) reflectivity factor [dBZ]

"VRADH", "VRAD" Radial velocity [m/s]. Radial velocities towards the radar are negative, while radial velocities away from the radar are positive

"RHOHV" Correlation coefficient [unitless]. Correlation between vertically polarized and horizontally polarized reflectivity factor

"PHIDP" Differential phase [degrees]

"ZDR" (Logged) differential reflectivity [dB]

The scan parameters are named according to the OPERA data information model (ODIM), see Table 16 in the ODIM specification.

Value

A ggmap object (a classed raster object with a bounding box attribute).

Methods (by class)

- ppi: plot a 'ppi' object on a map

Examples

```
# load an example scan:
data(example_scan)
# make ppi's for all scan parameters in the scan
ppi <- project_as_ppi(example_scan)
# grab a basemap that matches the extent of the ppi:
## Not run:
basemap <- download_basemap(ppi)

## End(Not run)
# map the radial velocity scan parameter onto the basemap:
## Not run:
map(ppi, map = basemap, param = "VRADH")

## End(Not run)
# extend the plotting range of velocities, from -50 to 50 m/s:
## Not run:
map(ppi, map = basemap, param = "VRADH", zlim = c(-50, 50))

## End(Not run)
# give the data less transparency:
## Not run:
map(ppi, map = basemap, alpha = 0.9)

## End(Not run)
# change the appearance of the symbol indicating the radar location:
## Not run:
map(ppi, map = basemap, radar_size = 5, radar_color = "green")

## End(Not run)
# crop the map:
## Not run:
map(ppi, map = basemap, xlim = c(12.4, 13.2), ylim = c(56, 56.5))

## End(Not run)
```

nexrad_to_odim *Convert a NEXRAD polar volume file to an ODIM polar volume file*

Description

Convert a NEXRAD polar volume file to an ODIM polar volume file

Usage

```
nexrad_to_odim(pvolfile_nexrad, pvolfile_odim, verbose = FALSE,
  mount = dirname(pvolfile_nexrad))
```

Arguments

pvolfile_nexrad	Polar volume input file in RSL format.
pvolfile_odim	Filename for the polar volume in ODIM hdf5 format to be generated.
verbose	logical. When TRUE, pipe Docker stdout to R console. On Windows always TRUE.
mount	character. String with the mount point (a directory path) for the Docker container.

Value

TRUE on success

plot.ppi *Plot a plan position indicator (*ppi*)*

Description

Plot a plan position indicator (PPI) generated with ppi using ggplot

Usage

```
## S3 method for class 'ppi'
plot(x, param, xlim, ylim, zlim = c(-20, 20), ratio = 1,
  ...)
```

Arguments

x	An object of class ppi.
param	The scan parameter to plot, see details below.
xlim	Range of x values to plot.
ylim	Range of y values to plot.
zlim	The range of parameter values to plot.
ratio	Aspect ratio between x and y scale.
...	Arguments passed to low level ggplot function.

Details

Available scan paramaters for plotting can by printed to screen by summary(x). Commonly available parameters are:

"DBZH", "DBZ" (Logged) reflectivity factor [dBZ]

"VRADH", "VRAD" Radial velocity [m/s]. Radial velocities towards the radar are negative, while radial velocities away from the radar are positive

"RHOHV" Correlation coefficient [unitless]. Correlation between vertically polarized and horizontally polarized reflectivity factor

"PHIDP" Differential phase [degrees]

"ZDR" (Logged) differential reflectivity [dB]

The scan parameters are named according to the OPERA data information model (ODIM), see Table 16 in the ODIM specification.

Examples

```
# load an example scan:
data(example_scan)
# print to screen the available scan parameters:
summary(example_scan)
# make ppi for the scan
ppi <- project_as_ppi(example_scan)
# plot the first scan parameter, which in this case is "VRADH":
plot(ppi)
# plot the reflectivity parameter:
plot(ppi, param = "DBZH")
# change the range of reflectivities to plot to -30 to 50 dBZ:
plot(ppi, param = "DBZH", zlim = c(-30, 50))
```

plot.scan *Plot a scan (*scan*) in polar coordinates*

Description

Plots a scan in polar coordinates. For plots in Cartesian coordinates, see ppi

Usage

```
## S3 method for class 'scan'
plot(x, param, xlim = c(0, 100), ylim = c(0, 360),
  zlim = c(-20, 20), ...)
```

Arguments

x	An object of class scan.
param	The scan parameter to plot, see details below.
xlim	Range of x (range, distance from radar) values to plot.
ylim	Range of y (azimuth) values to plot.
zlim	The range of parameter values to plot.
...	Arguments passed to low level ggplot function.

Details

Available scan parameters for plotting can by printed to screen by summary(x). Commonly available parameters are:

"DBZH", "DBZ" (Logged) reflectivity factor [dBZ]

"VRADH", "VRAD" Radial velocity [m/s]. Radial velocities towards the radar are negative, while radial velocities away from the radar are positive

"RHOHV" Correlation coefficient [unitless]. Correlation between vertically polarized and horizontally polarized reflectivity factor

"PHIDP" Differential phase [degrees]

"ZDR" (Logged) differential reflectivity [dB]

The scan parameters are named according to the OPERA data information model (ODIM), see Table 16 in the ODIM specification.

Examples

```
# load an example scan:
data(example_scan)
# print to screen the available scan parameters
summary(example_scan)
# make ppi for the scan
# plot the reflectivity param:
```

```
plot(example_scan, param = "DBZH")
# change the range of reflectivities to plot to -30 to 50 dBZ:
plot(example_scan, param = "DBZH", zlim = c(-30, 50))
```

`plot.vp` *Plot a vertical profile (*`vp`*)*

Description

Plot a vertical profile (vp)

Usage

```
## S3 method for class 'vp'
plot(x, quantity = "dens",
  xlab = expression("volume density [#/km"^3 * "]"),
  ylab = "height [km]", line_col = "red", line_lwd = 1,
  line.col = "red", line.lwd = 1, ...)
```

Arguments

`x`	A vp class object.
`quantity`	Character string with the quantity to plot. See vp for list of available quantities. • Aerial density related:'`dens`', '`eta`', '`dbz`', '`DBZH`' for density, reflectivity, reflectivity factor and total reflectivity factor, respectively. • Ground speed related:'`ff`','`dd`', for ground speed and direction, respectively.
`xlab`	A title for the x axis.
`ylab`	A title for the y axis.
`line_col`	Color of the plotted curve.
`line_lwd`	Line width of the plotted curve.
`line.col`	Deprecated argument, use line_col instead.
`line.lwd`	Deprecated argument, use line_lwd instead.
`...`	Additional arguments to be passed to the low level plot plotting function.

Examples

```
data(example_vp)
plot(example_vp)
plot(example_vp, line_col = "blue")
```

plot.vpi *Plot an integrated profile (*vpi*)*

Description

Plot an object of class vpi.

Usage

```
## S3 method for class 'vpi'
plot(x, quantity = "mtr", xlab = "time",
  ylab = "migration traffic rate [#/km/h]", main = "MTR",
  night_shade = TRUE, elev = -0.268, lat = NULL, lon = NULL,
  ylim = NULL, nightshade = TRUE, ...)
```

Arguments

x	1 class object inheriting from class vpi, typically a call to integrate_profile.
quantity	Character string with the quantity to plot, one of 'vid' (vertically integrated density), 'vir' (vertically integrated reflectivity), 'mtr' (migration traffic rate), 'rtr' (reflectivity traffic rate), 'mt' ((cumulative) migration traffic), 'rt' ((cumulative) reflectivity traffic), 'ff', (height-averaged speed) 'dd', (height-averaged direction) 'u', (height-averaged u-component of speed), 'v', (height-averaged v-component of speed).
xlab	A title for the x-axis.
ylab	A title for the y-axis.
main	A title for the plot.
night_shade	Logical, whether to plot night time shading.
elev	Numeric, sun elevation to use for day/night transition, see suntime.
lat	(optional) Latitude in decimal degrees. Overrides the lat attribute of x.
lon	(optional) Longitude in decimal degrees. Overrides the lon attribute of x.
ylim	y-axis plot range, numeric atomic vector of length 2.
nightshade	Deprecated argument, use night_shade instead.
...	Additional arguments to be passed to the low level plot plotting function.

Details

The integrated profiles can be visualized in various related quantities, as specified by argument quantity:

"vid" Vertically Integrated Density, i.e. the aerial surface density of individuals. This quantity is dependent on the assumed radar cross section per individual (RCS)

"vir" Vertically Integrated Reflectivity. This quantity is independent of the value of individual's radar cross section

"mtr" Migration Traffic Rate. This quantity is dependent on the assumed radar cross section (RCS)

"rtr" Reflectivity Traffic Rate. This quantity is independent on the assumed radar cross section (RCS)

"mt" Migration Traffic. This quantity is dependent on the assumed radar cross section (RCS)

"rt" Reflectivity Traffic. This quantity is independent on the assumed radar cross section (RCS)

ff Horizontal ground speed in m/s

dd Horizontal ground speed direction in degrees

u Ground speed component west to east in m/s

v Ground speed component north to south in m/s

HGHT Height above sea level in m

The height-averaged speed quantities (ff,dd,u,v) and HGHT are weighted averages by reflectivity eta.

Examples

```
# vertically integrate a vpts object:
vpi <- integrate_profile(example_vpts)
# plot the migration traffic rates
plot(vpi)
# plot the vertically integrated densities, without night shading:
plot(vpi, quantity = "vid", night_shade = FALSE)
```

---

plot.vpts *Plot a time series of vertical profiles (*vpts*)*

Description

Plot a time series of vertical profiles of class vpts.

Usage

```
## S3 method for class 'vpts'
plot(x, xlab = "time", ylab = "height [m]",
  quantity = "dens", log = TRUE, barbs = TRUE, barbs_height = 10,
  barbs_time = 20, barbs_dens_min = 5, zlim, legend_ticks,
  legend.ticks, main, barbs.h = 10, barbs.t = 20, barbs.dens = 5,
  ...)
```

Arguments

x	A vp class object inheriting from class vpts.
xlab	A title for the x-axis.
ylab	A title for the y-axis.

quantity	Character string with the quantity to plot, one of 'dens','eta','dbz','DBZH' for density, reflectivity, reflectivity factor and total reflectivity factor, respectively.
log	Logical, whether to display quantity data on a logarithmic scale.
barbs	Logical, whether to overlay speed barbs.
barbs_height	Integer, number of barbs to plot in altitudinal dimension.
barbs_time	Integer, number of barbs to plot in temporal dimension.
barbs_dens_min	Numeric, lower threshold in aerial density of individuals for plotting speed barbs in individuals/km^3.
zlim	Optional numerical atomic vector of length 2, specifying the range of quantity values to plot.
legend_ticks	Numeric atomic vector specifying the ticks on the color bar.
legend.ticks	Deprecated argument, use legend_ticks instead.
main	A title for the plot.
barbs.h	Deprecated argument, use barbs_height instead.
barbs.t	Deprecated argument, use barbs_time instead.
barbs.dens	Deprecated argument, use barbs_dens_min instead.
...	Additional arguments to be passed to the low level image plotting function.

Details

Profile can be visualized in three related quantities, as specified by argument quantity:

"dens" the aerial density of individuals. This quantity is dependent on the assumed radar cross section (RCS) in the x$attributes$how$rcs_bird attribute

"eta" reflectivity. This quantity is independent of the value of the rcs_bird attribute

"dbz" reflectivity factor. This quantity is independent of the value of the rcs_bird attribute, and corresponds to the dBZ scale commonly used in weather radar meteorology. Bioscatter by birds tends to occur at much higher reflectivity factors at S-band than at C-band

"DBZH" total reflectivity factor. This quantity equals the reflectivity factor of all scatterers (biological and meteorological scattering combined)

In the speed barbs, each half flag represents 2.5 m/s, each full flag 5 m/s, each pennant (triangle) 25 m/s

Examples

```
# locate example file:
ts <- example_vpts
# plot density of individuals for the first 500 time steps, in the altitude
# layer 0-3000 m.
plot(ts[1:500], ylim = c(0, 3000))
# plot total reflectivity factor (rain, birds, insects together):
plot(ts[1:500], ylim = c(0, 3000), quantity = "DBZH")
```

`project_as_ppi` *Project a scan (`scan`) or parameter (`param`) to a plan position indicator (`ppi`)*

Description

Make a plan position indicator (ppi)

Usage

```
project_as_ppi(x, grid_size = 500, range_max = 50000,
  project = FALSE, ylim = NULL, xlim = NULL)

## S3 method for class 'param'
project_as_ppi(x, grid_size = 500, range_max = 50000,
  project = FALSE, ylim = NULL, xlim = NULL)

## S3 method for class 'scan'
project_as_ppi(x, grid_size = 500, range_max = 50000,
  project = FALSE, ylim = NULL, xlim = NULL)
```

Arguments

`x`	An object of class `param` or `scan`.
`grid_size`	Cartesian grid size in m.
`range_max`	Maximum range in m.
`project`	Whether to vertically project onto earth's surface.
`ylim`	The range of latitudes to include.
`xlim`	The range of longitudes to include.
`...`	Arguments passed to methods.

Details

The returned PPI is in Azimuthal Equidistant Projection.

Value

An object of class 'ppi'.

Methods (by class)

- `param`: Project as `ppi` for a single scan parameter.
- `scan`: Project multiple `ppi`'s for all scan parameters in a scan

Examples

```
# load a polar scan example object
data(example_scan)
example_scan
# make PPIs for all scan parameters in the scan:
ppi <- project_as_ppi(example_scan)
# print summary info for the ppi:
ppi
# copy the first scan parameter of the first scan in the volume to a new
# object 'param':
param <- example_scan$params[[1]]
# make a ppi for the new 'param' object:
ppi <- project_as_ppi(param)
# print summary info for this ppi:
ppi
```

---

rcs *Get radar cross section*

Description

Gives the currently assumed radar cross section in cm^2.

Usage

```
rcs(x)

## S3 method for class 'vp'
rcs(x)

## S3 method for class 'list'
rcs(x)

## S3 method for class 'vpts'
rcs(x)

## S3 method for class 'vpi'
rcs(x)
```

Arguments

x	A vp, list of vp or vpts object.

Value

a radar cross section in cm^2

Methods (by class)

- vp: radar cross section of a vertical profile
- list: radar cross sections for a list of vertical profiles
- vpts: radar cross section of a time series of vertical profile
- vpi: radar cross section of a time series of vertically integrated vertical profile(s)

Examples

```
# extract RCS for a single vertical profile:
rcs(example_vp)
```

---

rcs<- *Set radar cross section*

Description

Sets the assumed radar cross section in cm^2. This method also updates the migration densities in x$data$dens

Usage

```
rcs(x) <- value

## S3 replacement method for class 'vp'
rcs(x) <- value

## S3 replacement method for class 'list'
rcs(x) <- value

## S3 replacement method for class 'vpts'
rcs(x) <- value

## S3 replacement method for class 'vpi'
rcs(x) <- value
```

Arguments

x	a vp, list of vp or vpts object
value	the cross section value to assign

Examples

```
# change RCS for a single vertical profile:
rcs(example_vp) <- 20
```

read_cajun *Read a vertical profile (vp) from UMASS Cajun text file*

Description

Read a vertical profile (vp) from UMASS Cajun text file

Usage

```
read_cajun(file, rcs = 11, wavelength = "S")
```

Arguments

file	A text file containing the standard output (stdout) generated by UMASS Cajun pipeline
rcs	numeric. Radar cross section per bird in cm^2.
wavelength	Radar wavelength in cm, or one of 'C' or 'S' for C-band and S-band radar, respectively, in which case C-band wavelength is assumed to be 5.3 cm and S-band wavelength 10.6 cm

Value

An object inheriting from class vp, see vp for details.

read_pvolfile *Read a polar volume (pvol) from file*

Description

Read a polar volume (pvol) from file

Usage

```
read_pvolfile(file, param = c("DBZH", "VRADH", "VRAD", "RHOHV", "ZDR",
  "PHIDP", "CELL"), sort = TRUE, lat, lon, height, elev_min = 0,
  elev_max = 90, verbose = TRUE, mount = dirname(file))
```

Arguments

file	A string containing the path to a vertical profile generated by calculate_vp.
param	An atomic vector of character strings, containing the names of scan parameters to read. To read all scan parameters use 'all'.
sort	A logical value, when TRUE sort scans ascending by elevation.

lat	Latitude in decimal degrees of the radar position. If not specified, value stored in file is used. If specified, value stored in file is overwritten.
lon	Longitude in decimal degrees of the radar position. If not specified, value stored in file is used. If specified, value stored in file is overwritten.
height	Height of the center of the antenna in meters above sea level. If not specified, value stored in file is used. If specified, value stored in file is overwritten.
elev_min	Minimum scan elevation to read in degrees.
elev_max	Maximum scan elevation to read in degrees.
verbose	A logical value, whether to print messages (TRUE) to console.
mount	A character string with the mount point (a directory path) for the Docker container.

Details

Scan parameters are named according to the OPERA data information model (ODIM), see Table 16 in the ODIM specification. Commonly available parameters are:

"DBZH", "DBZ" (Logged) reflectivity factor [dBZ]

"VRADH", "VRAD" Radial velocity [m/s]. Radial velocities towards the radar are negative, while radial velocities away from the radar are positive

"RHOHV" Correlation coefficient [unitless]. Correlation between vertically polarized and horizontally polarized reflectivity factor

"PHIDP" Differential phase [degrees]

"ZDR" (Logged) differential reflectivity [dB]

Value

An object of class pvol, which is a list containing polar scans, i.e. objects of class scan

Examples

```
# locate example volume file:
pvol <- system.file("extdata", "volume.h5", package = "bioRad")
# print the local path of the volume file:
pvol
# load the file:
vol <- read_pvolfile(pvol)
# print summary info for the loaded polar volume:
vol
# print summary info for the scans in the polar volume:
vol$scans
# copy the first scan to a new object 'scan'
scan <- vol$scans[[1]]
# print summary info for the new object:
scan
```

read_vpfiles *Read a vertical profile (vp) or a list of vertical profiles (vp) from files*

Description

Read a vertical profile (vp) or a list of vertical profiles (vp) from files

Usage

```
read_vpfiles(files)
```

Arguments

files	A character vector containing the file names of vertical profiles in ODIM HDF5 format generated by calculate_vp.

Value

A single vp object or a list of vp objects.

Examples

```
## Not run:
read_vpfiles("my/path/profile1.h5")

## End(Not run)
## Not run:
read_vpfiles(c("my/path/profile1.h5", "my/path/profile2.h5", ...))

## End(Not run)

# locate example profile file:
vpfile <- system.file("extdata", "profile.h5", package = "bioRad")

# print the local path of the profile file:
vpfile

# load the file:
read_vpfiles(vpfile)
```

read_vpts — *Read a time series of vertical profiles (*vpts*) from file*

Description

Read a time series of vertical profiles (vpts) from file

Usage

```
read_vpts(file, radar, wavelength = "C")
```

Arguments

file	A text file containing the standard output (stdout) generated by vol2bird (or the package function calculate_vp).
radar	A string containing a radar identifier.
wavelength	Radar wavelength in cm, or one of 'C' or 'S' for C-band and S-band radar, respectively, in which case C-band wavelength is assumed to be 5.3 cm and S-band wavelength 10.6 cm

Value

An object inheriting from class vpts, see vpts for details.

Examples

```
# locate example file:
vptszipfile <- system.file("extdata", "vpts.txt.zip", package = "bioRad")
## Not run:
unzip(vptszipfile, "your/directory/and/file/name.txt")
# load time series:
ts <- read_vpts("your/directory/and/file/name.txt", radar = "KBGM", wavelength = "S")
ts

## End(Not run)
```

regularize_vpts — *Regularize a time series of vertical profiles (*vpts*) on a regular time grid*

Description

Projects objects of class vpts on a regular time grid

Usage

```
regularize_vpts(ts, interval = "auto", date_min = ts$daterange[1],
  date_max = ts$daterange[2], units = "mins", fill = FALSE,
  verbose = TRUE)
```

Arguments

ts	An object inheriting from class vpts, see vpts for details.
interval	Time interval grid to project on. When 'auto' the median interval in the time series is used.
date_min	Start time of the projected time series, as a POSIXct object. Taken from ts when 'auto'.
date_max	End time of the projected time series, as a POSIXct object. Taken from ts when 'auto'.
units	Optional units of interval, one of 'secs', 'mins', 'hours','days', 'weeks'. Defaults to 'mins'.
fill	Logical, whether to fill missing timesteps with the values of the closest neighboring profile.
verbose	Logical, when TRUE prints text to console.

Details

Irregular time series of profiles are typically aligned on a regular time grid with the expected time interval at which a radar provides data. Empty profiles with only missing data values will be inserted at time stamps of the regular time grid that have no matching profile in the irregular time series.

In plots of regular time series (see plot.vpts) temporal gaps of missing profiles (e.g. due to radar down time) become visible. In irregular time series data points in the plot are carried through until the time series continues, and temporal data gaps are filled up visually.

Value

An object of class vpts with regular time steps.

Examples

```
# start form example vpts object:
ts <- example_vpts
# regularize the time series on a 5 minute interval grid
tsRegular <- regularize_vpts(ts, interval = 5)
```

`sd_vvp_threshold` *Get threshold of the VVP-retrieved radial velocity standard deviation*

Description

Gives the current threshold in VVP-retrieved radial velocity standard deviation in m/s.

Usage

```
sd_vvp_threshold(x)

## S3 method for class 'vp'
sd_vvp_threshold(x)

## S3 method for class 'list'
sd_vvp_threshold(x)

## S3 method for class 'vpts'
sd_vvp_threshold(x)
```

Arguments

`x`	A `vp`, list of `vp` or `vpts` object.

Value

threshold for `sd_vvp` in m/s.

Methods (by class)

- `vp`: threshold in VVP-retrieved radial velocity standard deviation of a vertical profile
- `list`: threshold in VVP-retrieved radial velocity standard deviation of a list of vertical profiles
- `vpts`: threshold in VVP-retrieved radial velocity standard deviation of a time series of vertical profiles

Examples

```
# extract threshold for a single vertical profile:
sd_vvp_threshold(example_vp)
```

sd_vvp_threshold<- *Set threshold of the VVP-retrieved radial velocity standard deviation*

Description

Sets the threshold for sd_vvp. Altitude layers with sd_vvp below this threshold are assumed to have an aerial density of zero individuals. This method updates the migration densities in x$data$dens

Usage

```
sd_vvp_threshold(x) <- value

## S3 replacement method for class 'vp'
sd_vvp_threshold(x) <- value

## S3 replacement method for class 'list'
sd_vvp_threshold(x) <- value

## S3 replacement method for class 'vpts'
sd_vvp_threshold(x) <- value
```

Arguments

x	a vp, list of vp or vpts object
value	the value to assign

Examples

```
# change threshold for a single vertical profile:
sd_vvp_threshold(example_vp) <- 2
```

select_vpfiles *Select vertical profile (*vp*) files from computer*

Description

Create a list of vertical profile (vp) files from a local directory that match a specific date and radar range. Files are selected based on their file name (not directory structure), which should be of format radar_vp_yyyymmdd*.*, such as bewid_vp_20171123T1900Z_0x5.h5.

Usage

```
select_vpfiles(date_min = NULL, date_max = NULL, radars = NULL,
  directory = ".")
```

Arguments

date_min	character. YYYY-MM-DD start date of file selection.
date_max	character. YYYY-MM-DD end date of file selection.
radars	character (vector). 5-letter country/radar code(s) (e.g. bejab) of radars to include in file selection.
directory	character. Path to local directory where files should be looked for.

Value

Character vector of file paths that comply to the given date and radar range.

Examples

```
select_vpfiles(
  date_min = "2016-10-03",
  date_max = "2016-10-05",
  radars = "bejab",
  directory = "my_data"
)
```

---

summary.param *Class* param*: a parameter of a scan of a polar volume*

Description

Class param for a parameter of a scan of a polar volume, and its associated R base functions.

Usage

```
## S3 method for class 'param'
summary(object, ...)

is.param(x)
```

Arguments

object	Object of class param.
...	Additional arguments affecting the summary produced.
x	Object of class param.

Details

An object of class scan is a simple matrix, with the following specific attributes:

radar character string with the radar identifier

datetime nominal time of the volume to which this scan belongs [UTC]

lat latitude of the radar [decimal degrees]

lon longitude of the radar [decimal degrees]

height height of the radar antenna [meters above sea level]

get_elevation_angles radar beam elevation [degrees]

param string with the name of the polar scan parameter

Scan parameters are named according to the OPERA data information model (ODIM), see Table 16 in the ODIM specification. Commonly available parameters are:

"DBZH", "DBZ" (Logged) reflectivity factor [dBZ]

"VRADH", "VRAD" Radial velocity [m/s]. Radial velocities towards the radar are negative, while radial velocities away from the radar are positive

"RHOHV" Correlation coefficient [unitless]. Correlation between vertically polarized and horizontally polarized reflectivity factor

"PHIDP" Differential phase [degrees]

"ZDR" (Logged) differential reflectivity [dB]

Value

for is.scan: TRUE if its argument is of class "param"

---

summary.ppi *Class* ppi*: a plan position indicator*

Description

Class ppi for a plan position indicator, and its associated R base functions.

Usage

```
## S3 method for class 'ppi'
summary(object, ...)

is.ppi(x)

## S3 method for class 'ppi'
dim(x)
```

Arguments

object	Object of class ppi.
...	Additional arguments affecting the summary produced.
x	Object of class ppi.

Details

An object of class ppi is a list containing:

data an object of class SpatialGridDataFrame containing the georeferenced data. Commonly available parameters are:

"DBZH", "DBZ" (Logged) reflectivity factor [dBZ]

"VRADH", "VRAD" Radial velocity [m/s]. Radial velocities towards the radar are negative, while radial velocities away from the radar are positive

"RHOHV" Correlation coefficient [unitless]. Correlation between vertically polarized and horizontally polarized reflectivity factor

"PHIDP" Differential phase [degrees]

"ZDR" (Logged) differential reflectivity [dB]

geo geographic data, a list with:

lat latitude of the radar [decimal degrees]

lon longitude of the radar [decimal degrees]

height height of the radar antenna [meters above sea level]

elangle radar beam elevation [degrees]

rscale range bin size [m]

ascale azimuth bin size [deg]

The geo element of a 'scan' object is a copy of the geo element of its parent scan or scan parameter.

Value

For is.ppi: TRUE if its argument is of class ppi.

For dim.ppi: dimensions of the ppi.

---

summary.pvol *Class* pvol*: a polar volume*

Description

Class pvol for a polar volume, and its associated R base functions.

Usage

```
## S3 method for class 'pvol'
summary(object, ...)

is.pvol(x)
```

Arguments

object	Object of class pvol.
...	Additional arguments affecting the summary produced.
x	Object of class pvol.

Details

An object of class pvol is a list containing:

radar character string with the radar identifier

datetime nominal time of the volume [UTC]

scans a list with scan objects of class 'scan'

attributes list with the volume's \what, \where and \how attributes

geo geographic data, a list with:

- lat latitude of the radar [decimal degrees]
- lon longitude of the radar [decimal degrees]
- height height of the radar antenna [meters above sea level]

Value

for is.pvol: TRUE if its argument is of class pvol

Examples

```
# locate example volume file:
pvol <- system.file("extdata", "volume.h5", package = "bioRad")
# print the local path of the volume file:
pvol
# load the file:
vol <- read_pvolfile(pvol)
# print summary info for the loaded polar volume:
vol
# print summary info for the scans in the polar volume:
vol$scans
# copy the first scan to a new object 'scan'
scan <- vol$scans[[1]]
is.pvol("this is not a polar volume but a string") # > FALSE
```

summary.scan *Class* scan*: a scan of a polar volume*

Description

Class scan for a scan of a polar volume, and its associated R base functions.

Usage

```
## S3 method for class 'scan'
summary(object, ...)

is.scan(x)

## S3 method for class 'scan'
dim(x)
```

Arguments

object	Object of class scan
...	Additional arguments affecting the summary produced.
x	Object of class scan

Details

A object of class scan is a list containing:

radar character string with the radar identifier

datetime nominal time of the volume to which this scan belongs [UTC]

params a list with scan parameters

attributes list with the scan's \what, \where and \how attributes

geo geographic data, a list with:

- lat latitude of the radar [decimal degrees]
- lon longitude of the radar [decimal degrees]
- height height of the radar antenna [meters above sea level]
- elangle radar beam elevation [degrees]
- rscale range bin size [m]
- ascale azimuth bin size [deg]

The geo element of a scan object is a copy of the geo element of its parent polar volume of class pvol.

Value

For is.scan: TRUE if its argument is of class scan.

For dim.scan: dimensions of the scan.

Examples

```
# load example scan object
data(example_scan)
# print the scan parameters contained in the scan:
example_scan$params
# extract the first scan parameter:
param <- example_scan$params[1]
is.scan("this is not a polar scan but a string") # > FALSE
```

---

summary.vp *Class* vp*: a vertical profile of birds*

Description

Class vp for a vertical profile of birds, and its associated R base functions.

Usage

```
## S3 method for class 'vp'
summary(object, ...)

is.vp(x)

## S3 method for class 'vp'
dim(x)
```

Arguments

object	An object of class vp.
...	Additional arguments affecting the summary produced.
x	An object of class vp.

Details

An object of class vp contains a vertical profile. A vertical profile contains a collection of quantities, with each quantity having values at different altitude layers above the earth's surface, typically equally spaced altitudinal layers.

Data contained in this class object should be accessed with the get_quantity function. Information stored under attributes (see below) can be accessed directly.

A vp object is a list containing

radar the radar identifier

datetime the nominal time of the profile

data the profile data, a list containing:

HGHT height above mean sea level [m]. Alt. bin from HGHT to HGHT+interval)

u speed component west to east [m/s]
v speed component north to south [m/s]
w vertical speed (unreliable!) [m/s]
ff horizontal speed [m/s]
dd direction [degrees, clockwise from north]
sd_vvp VVP radial velocity standard deviation [m/s]
gap Angular data gap detected [T/F]
dbz Bird reflectivity factor [dBZ]
eta Bird reflectivity [cm^2/km^3]
dens Bird density [birds/km^3]
DBZH Total reflectivity factor (bio+meteo scattering) [dBZ]
n number of points VVP bird velocity analysis (u,v,w,ff,dd)
n_all number of points VVP st.dev. estimate (sd_vvp)
n_dbz number of points bird density estimate (dbz,eta,dens)
n_dbz_all number of points total reflectivity estimate (DBZH)

attributes list with the profile's \what, \where and \how attributes

Value

For is.vp: TRUE if its argument is of class vp.

For dim.vp: dimensions of the profile data.

---

summary.vpts *Class* vpts*: a time series of vertical profiles*

Description

Class vpts for a time series of vertical profiles, and its associated R base functions.

Usage

```
## S3 method for class 'vpts'
summary(object, ...)

is.vpts(x)

## S3 method for class 'vpts'
dim(x)
```

Arguments

object	An object of class vpts.
...	Additional arguments affecting the summary produced.
x	An object of class vpts.

Details

An object of class `vpts` contains time-ordered profiles of a single radar station.

The time series can be regular or irregular, indicated by the `regular` field

In a regular `vpts` object the profiles are equally spaced in time. In an irregular `vpts` object the time steps between profiles are of unequal length.

Irregular time series can be projected onto a regular time grid using the regularize_vpts function.

By contrast, vp objects can be concatenated in a list to combine profiles without time ordering, and profiles of multiple radars.

Data contained in this class object should be accessed with the get_quantity function. Information stored under `attributes` (see below) can be accessed directly.

An object of class `vpts` is a list containing

`radar` string containing the radar identifier

`datetime` the `N` nominal times of the profiles (named `dates` in bioRad versions < 0.4.0)

`heights` the `M` heights of the layers in the profile

`daterange` the minimum and maximum nominal time of the profiles in the list

`timesteps` time differences between the profiles. Element `i` gives the time difference between profile `i` and `i+1`

`data` list of `N` by `M` matrices containing the vertical profiles for each quantity. For a description of available quantities, see the `data` element of the `vp` class in read_vpfiles

`attributes` profile attributes, copied from the first profile contained in `x`

`regular` logical indicating whether the time series is regular or not

Value

For `is.vpts`: `TRUE` if its argument is of class `vpts`.

For `dim.vpts`: dimensions of the time series.

`sunrise_sunset` *Calculate sunrise or sunset for a time and place*

Description

Calculate sunrise or sunset for a time and place

Usage

```
sunrise(date, lon, lat, elev = -0.268, tz = "UTC")

sunset(date, lon, lat, elev = -0.268, tz = "UTC")
```

Arguments

date	Date inheriting from class POSIXt or a string interpretable by as.Date.
lon	Longitude in decimal degrees.
lat	Latitude in decimal degrees.
elev	Sun elevation in degrees.
tz	output time zone. Ignored if date has an associated time zone already

Details

The angular diameter of the sun is about 0.536 degrees, therefore the moment of sunrise/sunset corresponds to half that elevation at -0.268 degrees.

This is a convenience function mapping to crepuscule.

Approximate astronomical formula are used, therefore the moment of sunrise / sunset may be off by a few minutes

Value

The moment of sunrise or sunset in UTC time.

Examples

```
# sunrise in the Netherlands
sunrise("2016-01-01", 5, 53)
# sunset in the Netherlands
sunset("2016-01-01", 5, 53)
# civil twilight in Ithaca, NY, today
sunrise(Sys.time(), -76.5, 42.4, elev = -6)
```

---

update_docker *Update Docker image from Docker hub*

Description

Pulls and installs the latest Docker image used by bioRad from Docker hub

Usage

```
update_docker()
```

Details

This command pulls the latest vol2bird Docker image from Docker hub. Run this command to ensure all Docker functionality (e.g. the vol2bird function) runs at the latest available version.

Value

the POSIXct creation date of the installed Docker image

[.ppi *Subset a plan position indicator (*ppi*)*

Description

Select quantities by index from a ppi

Usage

```
## S3 method for class 'ppi'
x[i]
```

Arguments

x	An object of class param or scan.
i	Indices specifying elements to extract.

Examples

```
# make a ppi:
my_ppi <- project_as_ppi(example_scan)
# this ppi contains 5 quantities (VRADH DBZH ZDR RHOHV PHIDP):
my_ppi
#
# This ppi only contains the first quantity (VRADH):
my_ppi[1]
# This ppi contains the first three quantities (VRADH, DBZH, ZDR):
my_ppi[1:3]
```

[.vpts *Subset a time series of vertical profiles (*vpts*)*

Description

Select a vertical profile (vp) or a time series of vertical profiles (vpts) by index from a vpts

Usage

```
## S3 method for class 'vpts'
x[i]
```

Arguments

x	Object of class vpts.
i	Indices specifying elements to extract.

Examples

```
# we start with the example vertical profile time series:
example_vpts
# extract the 10th profile in the time series (returns a vp object)
example_vpts[10]
# extract the 20th to 100th profile form the time series (returns a vpts object)
example_vpts[20:100]
```

Index