Package 'bioRad'

June 16, 2025

Title Biological Analysis and Visualization of Weather Radar Data

Version 0.10.0

Description Extract, visualize and summarize aerial movements of birds and insects from weather radar data. See Dokter, A. M. et al. (2018) ``bioRad: biological analysis and visualization of weather radar data" <doi:10.1111/ecog.04028> for a software paper describing package and methodologies.

License MIT + file LICENSE

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

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

Depends R (>= 4.0.0)

Imports assertthat, curl, dplyr (>= 1.1.0), fields, ggplot2, glue, graphics, jsonlite, lifecycle, lubridate, lutz, methods, raster, readr, rhdf5, rlang, sf, sp, stats, stringr, suntools, tidyr, tidyselect, utils, viridis, viridisLite

Suggests aws.s3, ggspatial, knitr, prettymapr, rmarkdown, rosm, testthat (>= 3.0.0), vdiffr, vol2birdR (>= 1.1.0), withr

LazyData true

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Adriaan M. Dokter [aut, cre] (ORCID: <https://orcid.org/0000-0001-6573-066X>), Peter Desmet [aut] (ORCID: <https://orcid.org/0000-0002-8442-8025>), Bart Kranstauber [aut] (ORCID: <https://orcid.org/0000-0001-8303-780X>), Cecilia Nilsson [aut] (ORCID: <https://orcid.org/0000-0001-8957-4411>), Stijn Van Hoey [aut] (ORCID: <https://orcid.org/0000-0001-6413-3185>), Bart Hoekstra [ctb] (ORCID: <https://orcid.org/0000-0002-7085-3805>),

Pieter Huybrechts [ctb] (ORCID:
<https://orcid.org/0000-0002-6658-6062>),
Hidde Leijnse [ctb] (ORCID: <https://orcid.org/0000-0001-7835-4480>),
Nicolas Noé [ctb] (ORCID: <https://orcid.org/0000-0002-9503-4750>),
Raphaël Nussbaumer [ctb] (ORCID:
<https://orcid.org/0000-0002-8185-1020>),
Jurriaan Spaaks [ctb],
Alexander Tedeschi [ctb] (ORCID:
<https://orcid.org/0000-0003-0772-6931>),
Lourens Veen [ctb],
Liesbeth Verlinden [ctb] (ORCID:
<https://orcid.org/0000-0003-1744-9325>)

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

Repository CRAN

Date/Publication 2025-06-16 19:10:02 UTC

Contents

apply_mistnet *Apply MistNet segmentation to a polar volume*

Description

Applies the MistNet segmentation model to a polar volume file on disk and loads the resultant segmentation as a polar volume (pvol) object.

Usage

```
apply_mistnet(
  file,
  pvolfile_out,
  verbose = FALSE,
  mount = dirname(file),
  load = TRUE,
  mistnet_elevations = c(0.5, 1.5, 2.5, 3.5, 4.5),
  local_install,
  local_mistnet
)
```

Arguments

file	Character. Path to a polar volume (pvol) file.
pvolfile_out	Character. (optional) File name. When provided, writes a polar volume (pvol) file to disk that includes the Mistnet segmentation results.
verbose	Logical. When TRUE, vol2bird stdout is piped to the R console.
mount	Character. Directory path of the mount point for the Docker container (deprecated).
load	Logical. When TRUE, returns a pvol object.
mistnet_elevations	Numeric vector of length 5. Elevation angles to feed to the MistNet segmentation model, which expects exactly 5 elevation scans at 0.5, 1.5, 2.5, 3.5 and 4.5 degrees. Specifying different elevation angles may compromise segmentation results.
local_install	(deprecated) Character. Path to local vol2bird installation (e.g. your/vol2bird_install_directory/vo to use instead of the Docker container.
local_mistnet	Character. Path to local MistNet segmentation model in PyTorch format (e.g. /your/path/mistnet_nexrad.pt) to use.

Details

MistNet (Lin et al. 2019) is a deep convolutional neural network that has been trained using labels derived from S-band dual-polarization data across the US NEXRAD network. Its purpose is

to screen out areas of precipitation in weather radar data, primarily legacy data for which dual-polarization data are not available. Because the network has been trained on S-band data, it may not perform as well on C-band.

MistNet requires three single-polarization parameters as input: reflectivity (DBZH), radial velocity (VRADH), and spectrum width (WRADH), at 5 specific elevation angles (0.5, 1.5, 2.5, 3.5 and 4.5 degrees). Based on these data it can estimate a segmentation mask that identifies pixels with weather that should be removed when interested in biological data only.

MistNet will calculate three class probabilities (from 0 to 1, with 1 corresponding to a 100% probability) as additional scan parameters to the polar volume:

- BACKGROUND: Class probability that no signal was detected above the noise level of the radar.
- WEATHER: Class probability that weather was detected.
- BIOLOGY: Class probability that biological scatterers were detected.

MistNet will calculate three class probabilities (from 0 to 1, with 1 corresponding to a 100% probability) as additional scan parameters to the polar volume:

- BACKGROUND: Class probability that no signal was detected above the noise level of the radar
- WEATHER: Class probability that weather was detected
- BIOLOGY: Class probability that biological scatterers were detected

These class probabilities are only available for the 5 input elevations used as input for the MistNet model. Based on all the class probabilities a final weather segmentation map is calculated, stored as scan parameter CELL, which is available for all elevation scans.

- CELL: Final weather segmentation, with values > 1 indicating pixels classified as weather and values equal to 1 indicating pixels that are located within 5 km distance of a weather pixels.

A pixel is classified as weather if the class probability WEATHER > 0.45 or when the average class probability for rain across all five MistNet elevation scans at that spatial location > 0.45.

MistNet may run more slowly on Windows than on Linux or Mac OS X.

Value

When load is TRUE, a polar volume (pvol) object with the Mistnet segmentation results. When load is FALSE, TRUE on success.

References

Please cite this publication when using MistNet:

- Lin T-Y, Winner K, Bernstein G, Mittal A, Dokter AM, Horton KG, Nilsson C, Van Doren BM, Farnsworth A, La Sorte FA, Maji S, Sheldon D (2019) MistNet: Measuring historical bird migration in the US using archived weather radar data and convolutional neural networks. Methods in Ecology and Evolution 10 (11), pp. 1908-22. doi:10.1111/2041210X.13280

See Also

- check_docker()
- calculate_vp()

Examples

```
# make sure you have installed the MistNet libraries and install if necessary:
if (requireNamespace("vol2birdR", quietly = TRUE) && vol2birdR::mistnet_installed()){
# if mistnet has not been installedi yet, run the following:
if(!vol2birdR::mistnet_exists()){
   vol2birdR::install_mistnet()
   vol2birdR::install_mistnet_model()
}
# start a temporary file to store polar volume
tempfile=tempfile("KBGM_example")
# Download a NEXRAD file and save as KBGM_example
download.file(
  "https://noaa-nexrad-level2.s3.amazonaws.com/2019/10/01/KBGM/KBGM20191001_000542_V06",
  method="libcurl", mode="wb", tempfile
)

# Calculate MistNet segmentation
mistnet_pvol <- apply_mistnet(tempfile)

# Print summary info for the segmented elevation scan at the 0.5 degree,
# verify new parameters BIOLOGY, WEATHER, BACKGROUND and CELL have been added
scan <- get_scan(mistnet_pvol, 0.5)
scan

# Project the scan as a ppi
ppi <- project_as_ppi(scan, range_max = 100000)

# Plot the reflectivity parameter
plot(ppi, param = "DBZH")

# Plot the MistNet class probability [0-1] for weather
plot(ppi, param = "WEATHER")

# Plot the MistNet class probability [0-1] for biology
plot(ppi, param = "BIOLOGY")

# Plot the final segmentation result, with values >1 indicating
# areas classified as weather, and value 1 pixels that fall within an
# additional 5 km fringe around weather areas
plot(ppi, param = "CELL")

# Remove file
file.remove(tempfile)
}
```

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

Description

Converts a vertical profile (vp) or a time series of vertical profiles (vpts) to a data frame containing all quantities per datetime and height. Has options to include latitude/longitude/antenna height (parameter geo) and day/sunrise/sunset (parameter suntime).

Usage

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

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

Arguments

x	A vp or vpts object.
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed. See base::as.data.frame().
optional	Logical. 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. See base::as.data.frame().
geo	Logical. When TRUE, adds latitude (lat), longitude (lon) and antenna height of the radar (height_antenna) to each row.
suntime	Logical. When TRUE, adds whether it is daytime (day) and the datetime of sunrise and sunset to each row.
lat	Numeric. Radar latitude in decimal degrees. When set, overrides the latitude stored in x for sunrise()/sunset() calculations.

lon	Numeric. Radar longitude in decimal degrees. When set, overrides the longitude stored in x for sunrise()/sunset() calculations.
elev	Numeric. Sun elevation in degrees, used for 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(). This is different from the default plot.vp(), plot.vpts() and get_quantity() functions, where quantities eta, dbz, ff, u, v, w, dd are all thresholded by sd_vvp_threshold().

Value

A data.frame object, containing radar, datetime and height as rows and all profile quantities as columns, complemented with some oft-used additional information (columns lat, lon, height_antenna, day, sunrise, sunset).

See Also

- summary.vpts()

Examples

```
# Convert vp object to a data.frame
vp_df <- as.data.frame(example_vp)

# Print data.frame
vp_df

# Convert vpts object to a data.frame
vpts_df <- as.data.frame(example_vpts)

# Print the first 5 rows of the data.frame
vpts_df[1:5, ]

# Do not add lat/lon/height_antenna information
vpts_df <- as.data.frame(example_vpts, geo = FALSE)

# Do not add day/sunrise/sunset information
vpts_df <- as.data.frame(example_vpts, suntime = FALSE)

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

as.vp *Convert a dataframe into a vp object*

Description

Convert a dataframe into a vp object

Usage

```
as.vp(data)
```

Arguments

data	a dataframe created from a VPTS CSV file

Value

a bioRad vp object

Examples

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

as.vpts *Convert a dataframe into a vpts object*

Description

Convert a dataframe into a vpts object

Usage

```
as.vpts(data)
```

Arguments

data	a dataframe created from a VPTS CSV file

Value

a bioRad vpts object

Examples

```
# locate example file in VPTS CSV format:
df <- read.csv(system.file("extdata", "example_vpts.csv", package = "bioRad"))
# convert the data.frame to a vpts object:
as.vpts(df)
```

`attribute_table` *Extract a volume coverage pattern table with all attributes*

Description

Extract a volume coverage pattern table with all attributes

Usage

```
attribute_table(
  x,
 select = c("how.lowprf", "how.midprf", "how.highprf", "where.elangle", "where.nbins",
    "where.nrays", "where.rscale", "how.NI"),
  ...
)
```

Arguments

`x`	Either a pvol or scan for which the table should be created.
`select`	A character vector which the column names that should be returned when NULL all attributes are to be returned
`...`	Currently not used

This function tabulates the attributes of one scan or all scans of a pvol. Attributes that have a length longer then one are presented as a list column. By default the function returns a limited set of columns to keep the output clear. It is important to note that attributes of the full polar volume can contain additional information on processing that is not included in the resulting table. This function only tabulates attributes of the scans.

Value

A data.frame with the attributes of the scan(s)

Examples

```
data(example_scan)
attribute_table(example_scan)

pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")
example_pvol <- read_pvolfile(pvolfile)
attribute_table(example_pvol)
```

beam_distance *Calculate radar beam distance*

Description

Calculates the distance as measured over the earth's surface (the down range) for a given beam elevation and slant range.

Usage

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

Arguments

range	Numeric. Slant range, i.e. the length of the skywave path between target and the radar antenna, in m.
elev	Numeric. Beam elevation, in degrees.
k	Numeric. Standard refraction coefficient.
lat	Numeric. Geodetic latitude of the radar, in degrees.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

depends on beam_height to calculate beam height.

Value

Beam distance (down range), in m.

See Also

- beam_height()

Other beam_functions: beam_height(), beam_profile(), beam_profile_overlap(), beam_range(), beam_width(), gaussian_beam_profile()

Examples

```
# Down range of the 5 degree elevation beam at a slant range of 100 km:
beam_distance(100000, 5)
```

`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. Slant range, i.e. the length of the skywave path between target and the radar antenna, in m.
`elev`	Numeric. Beam elevation, in degrees.
`k`	Numeric. Standard refraction coefficient.
`lat`	Numeric. Geodetic latitude of the radar, in degrees.
`re`	Numeric. Earth equatorial radius, in km.
`rp`	Numeric. 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 m.

See Also

- `beam_width()`

Other beam_functions: `beam_distance()`, `beam_profile()`, `beam_profile_overlap()`, `beam_range()`, `beam_width()`, `gaussian_beam_profile()`

Examples

```
# Beam height in meters at 10 km range for a 1 degree elevation beam:
beam_height(10000, 1)

# Beam height in meters at 10 km range for a 3 and 5 degree elevation beam:
beam_height(10000, c(3, 5))

# Define ranges from 0 to 1000000 m (100 km), in steps of 100 m:
range <- seq(0, 100000, 100)

# Plot the beam height of the 0.5 degree elevation beam:
plot(range, beam_height(range, 0.5), ylab = "beam height [m]", xlab = "range [m]")
```

---

beam_profile *Calculate vertical radiation profile*

Description

Calculates for a set of beam elevations (`elev`) the altitudinal normalized distribution of radiated energy by those beams. Is a function of altitude (`height`) at a given distance (`distance`) from the radar, assuming the beams are emitted at antenna level

Usage

```
beam_profile(
  height,
  distance,
  elev,
  antenna = 0,
  beam_angle = 1,
  k = 4/3,
  lat = 35,
  re = 6378,
  rp = 6357
)
```

Arguments

`height`	Numeric. Height in m.
`distance`	Numeric. Distance from the radar as measured along sea level (down range), in m.
`elev`	Numeric vector. Beam elevation(s), in degrees.
`antenna`	Numeric. Height of the centre of the radar antenna, in m.
`beam_angle`	Numeric. Beam opening angle in degrees, typically the angle between the half-power (-3 dB) points of the main lobe.
`k`	Numeric. Standard refraction coefficient.

lat	Numeric. Geodetic latitude of the radar, in degrees.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

Beam profile is calculated using beam_height and beam_width. Returns a beam profile as a function of height relative to ground level.

Returns the normalized altitudinal pattern of radiated energy as a function of altitude at a given distance from the radar, assuming the beams are emitted at antenna level.

Value

Numeric vector. Normalized radiated energy at each of the specified heights.

See Also

Other beam_functions: beam_distance(), beam_height(), beam_profile_overlap(), beam_range(), beam_width(), gaussian_beam_profile()

Examples

```
# Plot the beam profile, for a 0.5 degree elevation beam at 50 km distance
# from the radar:
plot(beam_profile(height = 0:4000, 50000, 0.5), 0:4000,
  xlab = "normalized radiated energy",
  ylab = "height [m]", main = "beam elevation: 0.5 deg, distance=50km"
)

# Plot the beam profile, for a 2 degree elevation beam at 50 km distance
# from the radar:
plot(beam_profile(height = 0:4000, 50000, 2), 0:4000,
  xlab = "normalized radiated energy",
  ylab = "height [m]", main = "beam elevation: 2 deg, distance=50km"
)

# Plot the combined beam profile for a 0.5 and 2.0 degree elevation beam
# at 50 km distance from the radar:
plot(beam_profile(height = 0:4000, 50000, c(0.5, 2)), 0:4000,
  xlab = "normalized radiated energy",
  ylab = "height [m]", main = "beam elevations: 0.5,2 deg, distance=50km"
)
```

`beam_profile_overlap` *Calculate overlap between a vertical profile ('vp') and the vertical radiation profile emitted by the radar*

Description

Calculates the distribution overlap between a vertical profile ('vp') and the vertical radiation profile of a set of emitted radar beams at various elevation angles as given by beam_profile.

Usage

```
beam_profile_overlap(
  vp,
  elev,
  distance,
  antenna,
  zlim = c(0, 4000),
  noise_floor = -Inf,
  noise_floor_ref_range = 1,
  steps = 500,
  quantity = "dens",
  normalize = TRUE,
  beam_angle = 1,
  k = 4/3,
  lat,
  re = 6378,
  rp = 6357
)
```

Arguments

`vp`	A vp object.
`elev`	Numeric vector. Beam elevation(s), in degrees.
`distance`	Numeric. The distance(s) from the radar along sea level (down range) for which to calculate the overlap, in m.
`antenna`	Numeric. Radar antenna height, in m. Default to antenna height in vp.
`zlim`	Numeric vector of length two. Altitude range, in m
`noise_floor`	Numeric. The system noise floor in dBZ. The total system noise expressed as the reflectivity factor it would represent at a distance `noise_floor_ref_range` from the radar. NOT YET IMPLEMENTED
`noise_floor_ref_range`	Numeric. The reference distance from the radar at which `noise_floor` is expressed. NOT YET IMPLEMENTED.
`steps`	Numeric. Number of integration steps over altitude range `zlim`, defining altitude grid size used for numeric integration.

quantity	Character. Profile quantity (dens or eta) to use for the altitude distribution.
normalize	Logical. If TRUE, normalize the radiation coverage pattern over the altitude range specified by zlim.
beam_angle	Numeric. Beam opening angle in degrees, typically the angle between the half-power (-3 dB) points of the main lobe.
k	Numeric. Standard refraction coefficient.
lat	Numeric. Radar latitude. Defaults to latitude in vp.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

This function also calculates the overlap quantity in the output of integrate_to_ppi.

Overlap is calculated as the Bhattacharyya coefficient (i.e. distribution overlap) between the (normalized) vertical profile (vp) and the (normalized) radiation coverage pattern as calculated by beam_profile(). In the calculation of this overlap metric, NA and NaN values in the profile quantity specified by quantity are replaced with zeros.

The current implementation does not (yet) take into account the system noise floor when calculating the overlap.

In the ODIM data model the attribute /how/NEZ or /how/NEZH specifies the system noise floor (the Noise Equivalent Z or noise equivalent reflectivity factor. the H refers to the horizontal channel of a dual-polarization radar). In addition, the attribute /how/LOG gives "security distance above mean noise level (dB) threshold value". This is equivalent to the log receiver signal-to-noise ratio, i.e. the dB above the noise floor for the signal processor to report a valid reflectivity value. We recommend using NEZH + LOG for noise_floor, as this is the effective noise floor of the system below which no data will be reported by the radar signal processor.

Typical values are NEZH = -45 to -50 dBZ at 1 km from the radar. LOG is typically around 1 dB.

Need to evaluate beam by beam the returned signal relative to a uniform beam filling of at least NEZH + LOG If returned signal is lower, the gate is below noise level.

Value

A data.frame with columns distance and overlap.

See Also

- beam_height()
- beam_width()
- beam_profile()

Other beam_functions: beam_distance(), beam_height(), beam_profile(), beam_range(), beam_width(), gaussian_beam_profile()

Examples

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

# Read the corresponding vertical profile example
pvol <- read_pvolfile(pvolfile)

# let us use this example vertical profile:
data(example_vp)
example_vp

# Calculate overlap between vertical profile of birds and the vertical
# radiation profile emitted by the radar
bpo <- beam_profile_overlap(
  example_vp,
  get_elevation_angles(pvol), seq(0, 100000, 1000)
)

# Plot the calculated overlap:
plot(bpo)
```

---

beam_range *Calculate radar beam range*

Description

Calculates the range (i.e. slant range) given a distance measured along the earth's surface (i.e. down range) and beam elevation.

Usage

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

Arguments

distance	Numeric. Distance from the radar as measured along sea level (down range), in m.
elev	Numeric. Beam elevation, in degrees.
k	Numeric. Standard refraction coefficient.
lat	Numeric. Geodetic latitude of the radar, in degrees.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

depends on beam_height to calculate beam height.

Value

Beam range (slant range), in m.

See Also

Other beam_functions: beam_distance(), beam_height(), beam_profile(), beam_profile_overlap(), beam_width(), gaussian_beam_profile()

Examples

```
# Slant range of the 5 degree elevation beam at a down range of 100 km
beam_range(100000, 5)
```

---

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, i.e. distance from the radar antenna, in m.
beam_angle	Numeric. Beam opening angle in degrees, typically the angle between the half-power (-3 dB) points of the main lobe.

Value

numeric. Beam width in m, typically the full width at half maximum (FWHM).

See Also

Other beam_functions: beam_distance(), beam_height(), beam_profile(), beam_profile_overlap(), beam_range(), gaussian_beam_profile()

Examples

```
#' # Beam width in meters at 10 km range:
beam_width(10000)

# Define ranges from 0 to 1000000 m (100 km), in steps of 100 m:
range <- seq(0, 100000, 100)

# Plot the beam width as a function of range:
plot(range, beam_width(range), ylab = "beam width [m]", xlab = "range [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 on datetime. 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 object or a vector of these.
...	A vp, vpts object or a vector of these.
attributes_from	Integer. Which vpts to copy attributes from (default: first).

Details

bind_into_vpts() currently requires profiles to have aligning altitude layers that are of equal width. Profiles are allowed to differ in the number of altitude layers, i.e. the maximum altitude.

Value

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

Methods (by class)

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

See Also

- summary.vp()
- summary.vpts()

Examples

```
# Split the example 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]

# Bind the two vpts together
vpts1_and_2 <- bind_into_vpts(vpts1, vpts2)

# Verify that the binded vpts now has 20 profiles, 10 from vpts1 and 10 from
# vpts2
summary(vpts1_and_2)

# 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

Concatenates vertical profiles (vp) into a list of vertical profiles (c(vp, vp, vp)) and warns if they are not from a single radar.

Usage

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

Arguments

...	vp objects.

Value

A list of vp objects.

See Also

bind_into_vpts()

Examples

```
# concatenate vp objects into a list:
c(example_vp, example_vp)
```

---

calculate_param *Calculate a new scan parameter*

Description

Calculate a new parameter (param) for a scan (scan) or polar volume (pvol)

Usage

```
calculate_param(x, ...)

## S3 method for class 'pvol'
calculate_param(x, ...)

## S3 method for class 'ppi'
calculate_param(x, ...)

## S3 method for class 'scan'
calculate_param(x, ...)
```

Arguments

x	A pvol or scan object.
...	An expression defining the new scan parameter in terms of existing scan parameters.

Details

Calculates a new scan parameter (param) from a combination of existing scan parameters. Useful for calculating quantities that are defined in terms of other basic radar moments, like linear reflectivity eta, depolarization ratio (Kilambi et al. 2018), or for applying clutter corrections (CCORH) to uncorrected reflectivity moments (TH) as TH + CCORH.

For the expression to work it is important that the operation can be vectorized. For example the base ifelse function is not vectorized, in these cases alternatives can be used (e.g. dplyr::if_else).

Also note that some functions do not operate on a matrix or param object. One example is the dplyr::if_else function. A workaround is calling the c() function on a parameter to convert it to a vector (e.g. c(DBZH), see examples).

Value

An object of the same class as `x`, either a `pvol` or `scan`.

Methods (by class)

- `calculate_param(pvol)`: Calculate a new parameter (`param`) for all scans in a polar volume (`pvol`).
- `calculate_param(ppi)`: Calculate a new parameter (`param`) for a plan position indicator (`ppi`).
- `calculate_param(scan)`: Calculate a new parameter (`param`) for a scan (`scan`).

References

- Kilambi A, Fabry F, Meunier V (2018) A simple and effective method for separating meteorological from nonmeteorological targets using dual-polarization data. Journal of Atmospheric and Oceanic Technology 35, pp. 1415–1424. doi:10.1175/JTECHD170175.1

See Also

- `get_param()`

Examples

```
# Locate and read the polar volume example file
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")
pvol <- read_pvolfile(pvolfile)

# Calculate linear reflectivity ETA from reflectivity factor DBZH
radar_wavelength <- pvol$attributes$how$wavelength
pvol <- calculate_param(pvol, ETA = dbz_to_eta(DBZH, radar_wavelength))

# Add depolarization ratio (DR) as a scan parameter (see Kilambi 2018)
pvol <- calculate_param(pvol, DR = 10 * log10((ZDR + 1 - 2 * ZDR^0.5 * RHOHV) /
  (ZDR + 1 + 2 * ZDR^0.5 * RHOHV)))

# The function also works on scan and ppi objects
calculate_param(example_scan, DR = 10 * log10((ZDR + 1 - 2 * ZDR^0.5 * RHOHV) /
  (ZDR + 1 + 2 * ZDR^0.5 * RHOHV)))

# set all reflectivity  values to NA when correlation coefficient > 0.95
# (indicating precipitation)
if (require(dplyr, quietly = TRUE)) {
  calculate_param(pvol,
    DBZH=if_else(c(RHOHV)>.95, NA, c(DBZH)) )
}

# it also works for ppis
ppi <- project_as_ppi(example_scan)
calculate_param(ppi, exp(DBZH))
```

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

Description

Calculates a vertical profile of biological scatterers (`vp`) from a polar volume (`pvol`) file using the algorithm vol2bird (Dokter et al. 2011 doi:10.1098/rsif.2010.0116).

Usage

```
calculate_vp(
  file,
  vpfile = "",
  pvolfile_out = "",
  autoconf = FALSE,
  verbose = FALSE,
  warnings = TRUE,
  mount,
  sd_vvp_threshold,
  rcs = 11,
  dual_pol = TRUE,
  rho_hv = 0.95,
  single_pol = TRUE,
  elev_min = 0,
  elev_max = 90,
  azim_min = 0,
  azim_max = 360,
  range_min = 5000,
  range_max = 35000,
  n_layer = 20,
  h_layer = 200,
  dealias = TRUE,
  nyquist_min = if (dealias) 5 else 25,
  dbz_quantity = "DBZH",
  mistnet = FALSE,
  mistnet_elevations = c(0.5, 1.5, 2.5, 3.5, 4.5),
  local_install,
  local_mistnet
)
```

Arguments

`file`	Character (vector). Either a path to a single radar polar volume (`pvol`) file containing multiple scans/sweeps, or multiple paths to scan files containing a single scan/sweep. Or a single `pvol` object. The file data format should be either 1) ODIM format, which is the implementation of the OPERA data information model in the HDF5 format, 2) a format supported by the RSL library or 3) Vaisala IRIS (IRIS RAW) format.

`vpfile` Character. File name. When provided, writes a vertical profile file (`vpfile`) either in the VPTS CSV or ODIM HDF5 format to disk.

`pvolfile_out` Character. File name. When provided, writes a polar volume (`pvol`) file in the ODIM HDF5 format to disk. Useful 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`, vol2bird `stdout` is piped to the R console.

`warnings` Logical. When `TRUE`, vol2bird warnings are piped to the R console.

`mount` Character. Directory path of the mount point for the Docker container (deprecated).

`sd_vvp_threshold` Numeric. Lower threshold for the radial velocity standard deviation (profile quantity `sd_vvp`) in m/s. Biological signals with `sd_vvp` < `sd_vvp_threshold` are set to zero. Defaults to 2 m/s for C-band radars and 1 m/s for S-band radars.

`rcs` Numeric. Radar cross section per bird to use, in cm^2.

`dual_pol` Logical. When `TRUE`, uses dual-pol mode, in which meteorological echoes are filtered using the correlation coefficient threshold `rho_hv`.

`rho_hv` Numeric. Lower threshold in correlation coefficient to use for filtering meteorological scattering.

`single_pol` Logical. When `TRUE`, uses precipitation filtering in single polarization mode based on reflectivity and radial velocity quantities.

`elev_min` Numeric. Minimum elevation angle to include, in degrees.

`elev_max` Numeric. Maximum elevation angle to include, in degrees.

`azim_min` Numeric. Minimum azimuth to include, in degrees clockwise from north.

`azim_max` Numeric. Maximum azimuth to include, in degrees clockwise from north.

`range_min` Numeric. Minimum range to include, in m.

`range_max` Numeric. Maximum range to include, in m.

`n_layer` Numeric. Number of altitude layers to use in generated profile.

`h_layer` Numeric. Width of altitude layers to use in generated profile, in m.

`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 to include, in m/s.

`dbz_quantity` Name of the available reflectivity factor to use if not `DBZH` (e.g. `DBZV`, `TH`, `TV`).

`mistnet` Logical. Whether to use the MistNet segmentation model.

`mistnet_elevations` Numeric vector of length 5. Elevation angles to feed to the MistNet segmentation model, which expects exactly 5 elevation scans at 0.5, 1.5, 2.5, 3.5 and 4.5 degrees. Specifying different elevation angles may compromise segmentation results.

`local_install` Character. Path to local vol2bird installation (e.g. `your/vol2bird_install_directory/vol2bird/bin`) (deprecated)

`local_mistnet` Character. Path to local MistNet segmentation model in PyTorch format (e.g. `/your/path/mistnet_nexrad.pt`).

Details

Typical use:

Common arguments set by users are `file`, `vpfile` and `autoconf`. 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 and dual polarization mode when available. The default for S-band data is to apply precipitation filtering in dual-polarization mode only.

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.

sd_vvp_threshold:

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 C-band weather radars, which is used as the C-band default. For S-band, the default threshold is 1 m/s.

rcs:

The default radar cross section (`rcs`) (11 cm^2) corresponds to the average value found by Dokter et al. (2011) in a calibration campaign of a full migration autumn season in western Europe at C-band. Its value may depend on radar wavelength. `rcs` will scale approximately $M^{2/3}$ with `M` the bird's mass.

dual_pol:

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

azim_min / azim_max:

`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_min / range_max:

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 35 km become too wide to resolve the default altitude layer width of 200 meter (see `beam_width()`). `range_max` may be extended up to 40 km (`40000`) for volumes with low elevations only, in order to extend coverage to higher altitudes.

h_layer:

The algorithm has been tested and developed for altitude layers with `h_layer = 200`m. Smaller widths than 100 m 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.

dealias:

Dealiasing uses the torus mapping method by Haase et al. (2004).

Local installation:

You may point parameter `local_mistnet` to a local download of the MistNet segmentation model in PyTorch format, e.g. `/your/path/mistnet_nexrad.pt`. The MistNet model can be downloaded at https://s3.amazonaws.com/mistnet/mistnet_nexrad.pt.

Value

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

References

Dokter et al. (2011) is the main reference for the profiling algorithm (vol2bird) underlying this function. When using the `mistnet` option, please also cite Lin et al. (2019). When dealiasing data (`dealias`), please also cite Haase et al. (2004).

- Dokter AM, Liechti F, Stark H, Delobbe L,Tabary P, Holleman I (2011) Bird migration flight altitudes studied by a network of operational weather radars, Journal of the Royal Society Interface 8 (54), pp. 30-43. doi:10.1098/rsif.2010.0116
- Haase G & Landelius T (2004) Dealiasing of Doppler radar velocities using a torus mapping. Journal of Atmospheric and Oceanic Technology 21(10), pp. 1566-1573. doi:10.1175/1520-0426(2004)021<1566:DODRVU>2.0.CO;2
- Lin T-Y, Winner K, Bernstein G, Mittal A, Dokter AM, Horton KG, Nilsson C, Van Doren BM, Farnsworth A, La Sorte FA, Maji S, Sheldon D (2019) MistNet: Measuring historical bird migration in the US using archived weather radar data and convolutional neural networks. Methods in Ecology and Evolution 10 (11), pp. 1908-22. doi:10.1111/2041210X.13280

See Also

- `summary.pvol()`
- `summary.vp()`

Examples

```
# Locate and read the polar volume example file
pvolfile_source <- system.file("extdata", "volume.h5", package = "bioRad")

# Copy the file to a temporary directory with read/write permissions
pvolfile <- paste0(tempdir(),"/volume.h5")
file.copy(pvolfile_source, pvolfile)

# Calculate the profile
if (requireNamespace("vol2birdR", quietly = TRUE)) {
vp <- calculate_vp(pvolfile)

# Get summary info
vp

# Clean up
file.remove(pvolfile)
```

```
}
```

---

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, vpi) this information is extracted from the bioRad object directly.

Usage

```
check_night(x, ..., elev = -0.268, offset = 0)

## Default S3 method:
check_night(x, lon, lat, ..., tz = "UTC", elev = -0.268, offset = 0)

## S3 method for class 'vp'
check_night(x, ..., elev = -0.268, offset = 0)

## S3 method for class 'list'
check_night(x, ..., elev = -0.268, offset = 0)

## S3 method for class 'vpts'
check_night(x, ..., elev = -0.268, offset = 0)

## S3 method for class 'vpi'
check_night(x, ..., elev = -0.268, offset = 0)

## S3 method for class 'pvol'
check_night(x, ..., elev = -0.268, offset = 0)
```

Arguments

x	A pvol, vp, vpts, vpi object, a POSIXct date or a string interpretable by base::as.POSIXct().
...	Optional lat, lon arguments.
elev	Numeric (vector). Sun elevation in degrees defining nighttime. May also be a numeric vector of length two, with first element giving sunset elevation, and second element sunrise elevation.
offset	Numeric (vector). Time duration in seconds by which to shift the start and end of nighttime. May also be a numeric vector of length two, with first element added to moment of sunset and second element added to moment of sunrise.

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

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.

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

offset can be used to shift the moment of sunset and sunrise by a temporal offset, for example, offset = c(600,-900) will assume nighttime starts 600 seconds after sunset (as defined by elev) and stops 900 seconds before sunrise.

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)

# Select nighttime profiles that are between 3 hours after sunset
# and 2 hours before sunrise:
index <- check_night(example_vpts, offset = c(3,-2)*3600)
example_vpts[index]
```

---

clean_mixture *Partition mixtures of animals using assumptions on airspeeds.*

Description

Partition mixtures of birds and insects using assumptions on their respective airspeeds, following the approach by Shi et al. (2025).

Usage

```
clean_mixture(x, ...)

## Default S3 method:
clean_mixture(
  x,
  slow = 1,
  fast = 8,
  drop_slow_component = TRUE,
  drop_missing = FALSE,
  keep_mixture = FALSE,
  u_wind,
  v_wind,
  u,
  v,
  ...
)

## S3 method for class 'vpts'
clean_mixture(
  x,
  slow = 1,
  fast = 8,
  drop_slow_component = TRUE,
  drop_missing = FALSE,
  keep_mixture = FALSE,
  u_wind = "u_wind",
  v_wind = "v_wind",
  ...
)

## S3 method for class 'vp'
clean_mixture(
  x,
  ...,
  slow = 1,
  fast = 8,
  drop_slow_component = TRUE,
  drop_missing = FALSE,
  keep_mixture = FALSE,
  u_wind = "u_wind",
  v_wind = "v_wind"
)
```

Arguments

x	a vp or vpts object, or a mixture animal density or linear reflectivity eta in cm^2/km^3.

...	eta, u, v, u_wind, v_wind arguments, taken from object for vp or vpts class.
slow	the slow component's airspeed in m/s, typically the airspeed of birds. Either a single number, or (optionally for vpts) a numeric vector equal in length to the number of profiles, or a data column name (see Details).
fast	the fast component's airspeed in m/s, typically the airspeed of insects. Either a single number, or (optionally for vpts) a numeric vector equal in length to the number of profiles, or a data column name (see Details).
drop_slow_component	when TRUE (default) output density, ground speed and heading for fast component, when FALSE for slow component.
drop_missing	Values eta without an associated ground speed and wind speed are set to NA when TRUE, or returned unaltered when FALSE (default).
keep_mixture	When TRUE store original mixture reflectivity and speeds as renamed quantities with mixture_ prefix
u_wind	the west to east wind component in m/s. In the case of vp and vpts objects the quantity name for the U-component of the wind.
v_wind	the south to north wind component in m/s. In the case of vp and vpts objects the quantity name for the V-component of the wind.
u	the mixture's ground speed u component (west to east) in m/s.
v	the mixture's ground speed v component (south to north) in m/s.

Details

For a detail description of the methodology see Shi et al. (2025). Most commonly the fast component refers to migrating birds, while the slow component refers to insects. The slow component is always oriented in the direction of the wind by definition. Note that for mixture airspeeds exceeding the airspeed of the fast component, all reflectivity is assigned to the fast component. Similarly, for mixture airspeeds below the airspeed of the slow component, all reflectivity will be assigned to the slow component.

How to use this function?:

1. To apply this function to vp or vpts data altitudinal wind data needs to be added to the vertical profile data first. This is most easily accomplished by first converting the objects to a data.frame with as.vp() or as.vpts(). Wind data can then be added as a new columns to the data.frame. By default the wind data is expected to be named u_wind for the U component and v_wind for the V component of the wind. Alternatively, arguments u_wind and v_wind can be used to specify different names.
2. Realistic assumptions for the expected airspeed for the slow (insect) and fast (bird) components need to be provided, using arguments slow and fast. See Shi et al. 2025 for recommendations in choosing these values. The parameter values for fast and slow can be specified as follows:
   - as single values applied to all heights and timestamps
   - as a numeric vector of equal length as the number of profiles in the vpts, allowing the user to specify changes in the parameter over time

- as the name of a profile data quantity, allowing the user to specify changes in the parameter over time and/or altitude. Profile quantities are most easily added by first converting the vpts object to a data.frame with as.data.frame.vpts(), adding the values, and back-converting with as.vpts

3. Use drop_slow_component to toggle between retaining the slow or fast component. When TRUE the fast (bird) component is retained. When FALSE the slow (insect) component is retained. Note that in this case the corrected ground speed direction will be identical to the wind direction, and the magnitude of the ground speed will be equal to the wind speed plus the value of slow, due to the underlying assumption of wind following by the slow component.

Value

a named list with cleaned densities and speeds. Output differs depending on whether the fast component is retained (drop_slow_component=TRUE, default) or the slow component (drop_slow_component=FALSE, default). Output quantities include:

- eta: cleaned reflectivity in cm^2/km^3. only the fast component (default) or the slow component (when drop_slow_component is TRUE).
- u: cleaned ground speed component west to east in m/s.
- v: cleaned ground speed component south to north in m/s.
- airspeed: the airspeed of the selected component in m/s.
- airspeed_u: the u-component (west to east) of the airspeed of the retained component in m/s.
- airspeed_v: the v-component (south to north) of the airspeed of the retained component in m/s.
- heading: the heading of the selected component in degrees clockwise from north.
- f: the reflectivity proportion of the slow component (0-1 range), typically the proportion of insects.

For vp and vpts objects the quantities eta,u,v will be updated, and other quantities listed above will be added.

References

- Shi X, Drucker J, Chapman JW, Sanchez Herrera M, Dokter AM Analysis of mixtures of birds and insects in weather radar data. Ornithological Applications. 2025 (in press) doi:10.1093/ornithapp/duaf020.
- Nussbaumer R, Schmid B, Bauer S, Liechti F. A Gaussian mixture model to separate birds and insects in single-polarization weather radar data. Remote Sensing. 2021 May 19;13(10):1989 doi:10.3390/rs13101989.

Examples

```
# convert profile object to data.frame
df <- as.data.frame(example_vp, suntime=FALSE)
# add wind u and v component wind data
# (here a NW wind identical at all altitudes)
df$u_wind=3
```

```
df$v_wind=-3
# convert back to vp object
my_vp <- as.vp(df)
# partition the mixture:
my_vp_clean <- clean_mixture(my_vp)

# drop the slow component (typically insects)
clean_mixture(100,u=-13,v=13,u_wind=-7,v_wind=6, fast=8, slow=1)
# drop the fast component (typically birds)
clean_mixture(100,u=-13,v=13,u_wind=-7,v_wind=6, fast=8, slow=1, drop_slow_component=FALSE)
# keep the original mixture reflectivity and speed components
clean_mixture(100,u=-13,v=13,u_wind=-7,v_wind=6, fast=8, slow=1, keep_mixture=TRUE)
# keep reflectivity unaltered when one of the speed components is not a number:
clean_mixture(100,u=-13,v=13,u_wind=NaN,v_wind=6, fast=8, slow=1)["eta"]
# set reflectivity to NaN when one of the speed components is not a number:
clean_mixture(100,u=-13,v=13,u_wind=NaN,v_wind=6, fast=8, slow=1, drop_missing=TRUE)["eta"]
```

`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 = "all",
  nx = 100,
  ny = 100,
  xlim,
  ylim,
  res,
  crs,
  raster = NA,
  method = "max",
  idp = 2,
  idw_max_distance = NA,
  coverage = FALSE
)
```

Arguments

`x`	A list of `ppi` objects.
`param`	Character (vector). One or more parameter name(s) to composite. To composite all available scan parameters use `all` (default).

nx	number of raster pixels in the x (longitude) dimension
ny	number of raster pixels in the y (latitude) dimension
xlim	x (longitude) range
ylim	y (latitude) range
res	numeric vector of length 1 or 2 to set the resolution of the raster (see res). If this argument is used, arguments nx and ny are ignored. Unit is identical to xlim and ylim.
crs	character or object of class CRS. PROJ.4 type description of a Coordinate Reference System (map projection). When 'NA' (default), an azimuthal equidistant projection with origin at the radar location is used. To use a WSG84 (lat,lon) projection, use crs="+proj=longlat +datum=WGS84"
raster	(optional) RasterLayer with a CRS. When specified this raster topology is used for the output, and nx, ny, res arguments are ignored.
method	Character (vector). Compositing method(s), either mean, min, max or idw. To apply different methods for each of the parameters, provide a vector with the same length as param.
idp	Numeric. Inverse distance weighting power.
idw_max_distance	Numeric. Maximum distance from the radar to consider in inverse distance weighting. Measurements beyond this distance will have a weighting factor of zero.
coverage	Logical. When TRUE, adds an additional coverage parameter to the ppi indicating the number of ppis covering a single composite pixel.

Details

The function can combine multiple ppis of different scan elevations of the same radar or ppis of different radars. The coordinates of the returned ppi object are in the WGS84 datum, unless a different crs is provided. If only res is provided, but no crs is set, res is in meters and the origin of the composite ppi is set to the mean(lat, lon) location.

The method parameter determines how values of different ppis at the same geographic location are combined:

- mean: Compute the average value.
- max: Compute the maximum value. If ppis are of the same radar and the same polar volume, this computes a max product, showing the maximum detected signal at that geographic location.
- min: Compute the minimum value.
- idw: This option is useful primarily when compositing ppis of multiple radars. Performs an inverse distance weighting, where values are weighted according to 1/(distance from the radar)^idp.

Argument method determines how values of different ppi's at the same geographic location are combined.

- mean: Compute the average value

- max: Compute the maximum value. If ppi's are of the same radar and the same polar volume, this computes a max product, showing the maximum detected signal at that geographic location.
- min: Compute the minimum value
- idw: This option is useful primarily when compositing ppi's of multiple radars. Performs an inverse distance weighting, where values are weighted according to 1/(distance from the radar)^idp

The coordinates system of the returned ppi is a WGS84 (lat, lon) datum, unless a different crs is provided. If only res is provided, but no crs is set, res is in meter units and the origin of the composite ppi is set to the mean (lat, lon) location.

This function is a prototype and under active development

Value

A ppi object.

Examples

```
# Locate and read the polar volume example file

pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")
pvol <- read_pvolfile(pvolfile)

# Calculate a ppi for each elevation scan, 1000m grid resolution
ppis <- lapply(pvol$scans, project_as_ppi, grid_size=1000)

# Overlay the ppis, calculating the maximum value observed
# across the available scans at each geographic location
composite <- composite_ppi(ppis, method = "max", res=1000)

# Plot the calculated max product on the basemap
if (all(sapply(c("ggspatial","prettymapr", "rosm"), requireNamespace, quietly = TRUE))) {
map(composite)
}
```

---

convert_legacy *Convert legacy bioRad objects*

Description

Convert legacy bioRad objects (vp, vpts) and make them compatible with the current bioRad version. Conversion includes renaming HGHT to height.

Usage

```
convert_legacy(x)

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

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

Arguments

x	A vp or vpts object.

Value

An updated object of the same class as the input.

See Also

- summary.vp()
- summary.vpts()

Examples

```
# Convert a vp object
vp <- convert_legacy(example_vp)

# Convert a vpts object
vpts <- convert_legacy(example_vpts)
```

---

dbz_to_eta *Convert reflectivity factor (dBZ) to reflectivity (eta)*

Description

Converts reflectivity factor (dBZ) to reflectivity (eta).

Usage

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

Arguments

dbz	Numeric. Reflectivity factor, in dBZ.
wavelength	Numeric. Radar wavelength, in cm.
K	Numeric. Norm of the complex refractive index of water.

Value

Reflectivity, in cm^2/km^3.

See Also

- eta_to_dbz()

Examples

```
# Calculate eta for a 7 dBZ reflectivity factor at C-band:
dbz_to_eta(7, 5)

# Calculate eta for a 7 dBZ reflectivity factor at S-band:
dbz_to_eta(7, 10)

# Calculate animal density for a 5 dBZ reflectivity factor, assuming a
# radar cross section of 11 cm^2 per individual
dbz_to_eta(7, 5) / 11 # C-band
dbz_to_eta(7, 10) / 11 # S-band
```

---

download_pvolfiles *Download polar volume (pvol) files from the NEXRAD archive*

Description

Download a selection of polar volume (pvol) files from the NEXRAD Level II archive data.

Usage

```
download_pvolfiles(
  date_min,
  date_max,
  radar,
  directory = ".",
  overwrite = FALSE,
  bucket = "noaa-nexrad-level2",
  directory_tree = TRUE
)
```

Arguments

date_min	POSIXct. Start date of file selection. If no timezone are provided, it will be assumed to be UTC.
date_max	POSIXct. End date of file selection.If no timezone are provided, it will be assumed to be UTC.
radar	character (vector). 4-letter radar code(s) (e.g. "KAMA")
directory	character. Path to local directory where files should be downloaded

overwrite	logical. TRUE for re-downloading and overwriting previously downloaded files of the same names.
bucket	character. Bucket name to use.
directory_tree	logical. Whether to create the yyyy/mm/dd/radar directory structure. TRUE by default.

Value

NULL. The function's primary effect is to download selected polar volume files from the NEXRAD Level II archive to a specified local directory, and to provide a message and a progress bar in the console indicating the download status.

Examples

```
# create temporary directory
if (requireNamespace("aws.s3", quietly = TRUE)) {
temp_dir <- paste0(tempdir(),"/bioRad_tmp_files")
dir.create(temp_dir)
download_pvolfiles(
  date_min = as.POSIXct("2002-10-01 00:00", tz = "UTC"),
  date_max = as.POSIXct("2002-10-01 00:05", tz = "UTC"),
  radar = "KBRO",
  directory = temp_dir,
  overwrite = TRUE
)
# Clean up
unlink(temp_dir, recursive = TRUE)
}
```

---

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

Description

[Superseded]

This function has been superseded by getRad::get_vpts().

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. Start date of file selection, in YYYY-MM-DD format. Days will be ignored.
date_max	Character. End date of file selection, in YYYY-MM-DD format. Days will be ignored.
radars	Character (vector). 5-letter country/radar code(s) to include in file selection.
directory	Character. Path to local directory where files should be downloaded and unzipped.
overwrite	Logical. When TRUE, re-download and overwrite previously downloaded files of the same names.

Value

NULL. The function's primary effect is to download selected vertical profiles files from ENRAM data repository to a specified local directory, and to provide a message and a progress bar in the console indicating the download status. Message will show a 404 error for files that are not available.

See Also

- read_vpts()
- select_vpfiles()
- read_vpfiles()

Examples

```
# Download (and overwrite) data from radars "bejab" and "bewid".
download_vpfiles(
  date_min = "2018-10-01",
  date_max = "2018-10-31",
  radars = c("bejab", "bewid"),
  directory = tempdir(),
  overwrite = TRUE
)
```

---

doy_noy *Look up day of year (doy) or night of year (noy)*

Description

Returns the day of year (doy) or night of year (noy) number for datetimes and various bioRad objects. The first night of the year is the night with datetime Jan 01 00:00:00 in the local time zone, so sunset on Jan 1 occurs on the second night of the year and New Years Eve on Dec 31 occurs on the first night of the new year.

Usage

```
doy(x, ..., method = "fast")

noy(x, ..., method = "fast")

## Default S3 method:
doy(x, lon, lat, ..., method = "fast")

## Default S3 method:
noy(x, lon, lat, ..., method = "fast")

## S3 method for class 'vp'
doy(x, ..., method = "fast")

## S3 method for class 'vp'
noy(x, ..., method = "fast")

## S3 method for class 'vpts'
doy(x, ..., method = "fast")

## S3 method for class 'vpts'
noy(x, ..., method = "fast")

## S3 method for class 'vpi'
doy(x, ..., method = "fast")

## S3 method for class 'vpi'
noy(x, ..., method = "fast")

## S3 method for class 'pvol'
doy(x, ..., method = "fast")

## S3 method for class 'pvol'
noy(x, ..., method = "fast")
```

Arguments

x	A pvol, vp, vpts, or vpi object, or a base::as.POSIXct datetime.
...	Optional lat, lon arguments.
method	Method by which to do the time zone lookup. Either fast (default) or accurate. See lutz::tz_lookup_coords].
lon	Numeric. Longitude in decimal degrees.
lat	Numeric. Latitude in decimal degrees.

Value

integer representing the ordinal day of year or night of year.

Examples

```
# Get day of year of a vp object
noy(example_vp)

# Get night of year of a vp object
noy(example_vp)

# Get night of year of a vpts object
noy(example_vpts)
```

`eta_to_dbz` *Convert reflectivity (eta) to reflectivity factor (dBZ)*

Description

Converts reflectivity (eta) to reflectivity factor (dBZ).

Usage

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

Arguments

`eta`	Numeric. Reflectivity, in cm^2/km^3.
`wavelength`	Numeric. Radar wavelength, in cm.
`K`	Numeric. Norm of the complex refractive index of water.

Value

Reflectivity factor, in dBZ.

Examples

```
# Calculate dBZ for a 10000 cm^2/km^3 eta reflectivity at C-band
eta_to_dbz(10000, 5)

# Calculate dBZ for a 10000 cm^2/km^3 eta reflectivity at S-band
eta_to_dbz(10000, 10)

# Calculate dBZ for an animal density of 1000 individuals/km^3 and a radar
# cross section of 11 cm^2 per individual
eta_to_dbz(1000 * 11, 5) # C-band
eta_to_dbz(1000 * 11, 10) # S-band
```

example_scan *Scan (scan) example*

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.

Value

An example object of type scan which represents a single scan from a weather radar.

See Also

- summary.scan()

Examples

```
# Reload example_scan from package (e.g. in case it was altered)
data(example_scan)

# Get summary info
example_scan
```

example_vp *Vertical profile (vp) example*

Description

Example of a vp object with name example_vp.

Usage

```
example_vp
```

Format

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

Value

An example object of type vp which represents a vertical profile.

See Also

- summary.vp()

Examples

```
# Reload example_vp from package (e.g. in case it was altered)
data(example_vp)

# Get summary info
example_vp
```

---

example_vpts *Time series of vertical profiles (*vpts*) example*

Description

Example of a vpts object with name example_vpts.

Usage

```
example_vpts
```

Format

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

Value

An example object of type vpts which represents a time series of vertical profiles.

See Also

- summary.vpts()

Examples

```
# Reload example_vpts from package (e.g. in case it was altered)
data(example_vpts)

# Get summary info
example_vpts
```

`filter_precip` *Posthoc precipitation filter*

Description

The posthoc precipitation filter assesses how much of the altitude column has a high total reflectivity factor (biology + meteorology) consistent with precipitation, and removes biological signals when there is evidence for the presence of precipitation. Applied to vertical profiles ('vp') or time series of vertical profiles ('vpts').

Usage

```
filter_precip(
  x,
 dbz = ifelse(x$attributes$how$wavelength < 7 || is.null(x$attributes$how$wavelength),
    7, 20),
  range = 2500,
  alt_max = 3000,
  drop = FALSE
)
```

Arguments

`x`	A `vp` or `vpts` object.
`dbz`	The minimum reflectivity factor for precipitation in units dBZ. Defaults to 7 dBZ for C-band radars and 20 dBZ for S-band radars.
`range`	The minimum altitude range with total reflectivity factor. `DBZH` > `dbz` at which the filter is triggered in m.
`alt_max`	Maximum altitude above ground level to consider in m.
`drop`	When `TRUE` the profile is removed from the

Details

During precipitation events usually a high proportion of the altitude column will show a high total reflectivity `DBZH` (which includes biology + meteorology), because precipitation usually falls from several kilometers high to the ground surface. Precipitation events are often obvious in profile plots of quantity `DBZH` as reflectivity signals extending from ground level to high altitudes far above the typical altitudes where biology is expected. This filter identifies and removes these cases.

The posthoc precipitation filter examines the total reflectivity factor `DBZH` and calculates the altitude range at which `DBZH` is larger than parameter `dbz`. Whenever this altitude range is larger than parameter `range` (and `drop` is `FALSE`), the biology is removed by setting profile quantities `dens` and `eta` to zero and profile quantity `dbz` to `-Inf`. When parameter `drop` is `TRUE`, the profile is removed from the time series altogether.

This posthoc precipitation filter is likely to remove biological scatterers that co-occur with precipitation, for example biological movements during isolated convective thunderstorms. It is more

aggressive than precipitation filters applied during vertical profile generation with `calculate_vp()` that attempt to remove precipitation and retain biology. The posthoc precipitation filter is especially useful for analyses where you want to minimize the risk of precipitation contamination, at the cost of filtering out some biology during precipitation events.

Lowering the default minimum reflectivity (`dbz`) for precipitation below 7 dBZ is not recommended, as most precipitation has a reflectivity above 7 dBZ.

Parameter `range` should be chosen carefully, and should be higher than the typical altitude where biological scatterers still reach a reflectivity factor equal to `dbz`.

Note that at S-band wavelengths bird migration occurs much more frequently in the reflectivity regime for precipitation than at C-band. Therefore, at C-band lower settings for parameter `dbz` are appropriate than at S-band.

Value

A `vpts` object or a `vp` object, depending on input `x`.

See Also

- eta_to_dbz
- dbz_to_eta

Examples

```
# rain periods are visible in quantity DBZH as dark vertical lines
# extending above 3 km height:
plot(regularize_vpts(example_vpts), quantity='DBZH')
# Apply posthoc filter to remove profiles during precipitation events:
# (regularization is applied to visualize removed profiles)
my_vpts <- regularize_vpts(filter_precip(example_vpts, drop=TRUE))
# verify that rain events have been removed:
plot(my_vpts, quantity='DBZH')
# the posthoc filter removes some biology during precipitation:
plot(my_vpts, quantity='dens')
# original retains more biology (but at the cost of a higher
# likelihood of occasional precipitation contamination):
plot(regularize_vpts(example_vpts), quantity='dens')
# filter can also be applied to single vp objects:
filter_precip(example_vp)
```

`filter_vpts` *Filter a time series of vertical profiles ('vpts').*

Description

Filter a time series of vertical profiles ('vpts') by a start and end time, for night or day time, or extract profile nearest to provided timestamp. Use argument night = TRUE to select only time stamps between sunset and sunrise, or night = FALSE to select daytime (sunrise to sunset). Selection for night and day uses `check_night()`.

Usage

```
filter_vpts(x, min, max, nearest, night, elev = -0.268, offset = 0)
```

Arguments

x	A vpts object.
min	POSIXct date or character. Minimum datetime to be included.
max	POSIXct date or character. Datetime up to this maximum included.
nearest	POSIXct date or character. If specified, min and max are ignored and the profile (vp) nearest to the specified datetime is returned that matches the day/night selection criteria.
night	When TRUE selects only nighttime profiles, when FALSE selects only daytime profiles, as classified by check_night().
elev	Numeric (vector). Sun elevation in degrees defining nighttime. May also be a numeric vector of length two, with first element giving sunset elevation, and second element sunrise elevation.
offset	Numeric (vector). Time duration in seconds by which to shift the start and end of nighttime. May also be a numeric vector of length two, with first element added to moment of sunset and second element added to moment of sunrise. See check_night() for details.

Details

Returns profiles for which min <= timestamp profile < max. Selection for night and day occurs by check_night.

Value

A vpts object, or a vp object when nearest is specified.

See Also

- summary.vpts()
- check_night()

Examples

```
# Select profiles later than 02 Sep 2016

# Select the profile nearest to 2016-09-01 03:00 UTC
filter_vpts(example_vpts, nearest = "2016-09-01 03:00")

# Select profiles between than 01:00 and 03:00 UTC on 02 Sep 2016
filter_vpts(example_vpts, min = "2016-09-02 01:00", max = "2016-09-02 03:00")

# Select daytime profiles (i.e. profiles between sunrise and sunset)
filter_vpts(example_vpts, night = FALSE)
```

```
# Select nighttime profiles, with nights starting and ending at
# civil twilight (when the sun is 6 degrees below the horizon)
filter_vpts(example_vpts, night = TRUE, elev = -6)

# Select nighttime profiles from 3h after sunset to 2h before sunrise
filter_vpts(example_vpts, night = TRUE, offset = c(3, -2)*3600)
```

`get_elevation_angles` *Get elevation angles of a polar volume (*`pvol`*), scan (*`scan`*) or parameter (*`param`*)*

Description

Returns the elevation angles in degrees of all scans within a polar volume (`pvol`) or the elevation angle of a single scan (`scan`) or scan parameter (`param`).

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`, `scan` or `param` object.

Value

The elevation angle(s) in degrees.

See Also

- `get_scan()`

Examples

```
# Locate and read the polar volume example file
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")
example_pvol <- read_pvolfile(pvolfile)

# Get the elevations angles of the scans in the pvol
```

```
get_elevation_angles(example_pvol)

# Extract the first scan
scan <- example_pvol$scans[[1]]

# Get the elevation angle of that scan
get_elevation_angles(scan)
```

---

get_iris_raw_task *Check the* task *type of an IRIS RAW file*

Description

Checks what type of task(s), i.e. polar volume types, are contained in an IRIS RAW file.

Usage

```
get_iris_raw_task(
  file,
  header_size = 50,
  task = c("WIND", "SURVEILLANCE", "VOL_A", "VOL_B")
)
```

Arguments

file	Character. Path to a polar volume file in IRIS RAW format.
header_size	Integer. Number of header bytes to search.
task	Character (vector). Task names to search for in the file header.

Value

Specified task names found in the header or NA if none of the task names were found.

---

get_odim_object_type *Check the* data *type of an ODIM HDF5 file*

Description

Checks what type of data object is contained in an ODIM HDF5 file. See ODIM specification, Table 2 for a full list of existing ODIM file object types.

Usage

```
get_odim_object_type(file)
```

Arguments

file	Character. Path of the file to check.

Value

Character. PVOL for polar volume, VP for vertical profile, otherwise NA.

See Also

- is.pvolfile()
- is.vpfile()

Examples

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

# Check the data type
get_odim_object_type(pvolfile)
```

---

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

Description

Returns the selected parameter (param) from a scan (scan).

Usage

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

Arguments

x	A scan object.
param	Character. A scan parameter, such as DBZH or VRADH. See summary.param() for commonly available parameters.

Value

A param object.

See Also

- summary.param()

Examples

```
# Get summary info for a scan (including parameters)
example_scan

# Extract the VRADH scan parameter
param <- get_param(example_scan, "VRADH")

# Get summary info for this parameter
param
```

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

Description

Returns values for the selected quantity from a vertical profile (`vp`), list, or time series of vertical profiles (`vpts`). Values are organized per height bin. Values for `eta` are set to `0`, `dbz` to `-Inf` and `ff`, `u`, `v`, `w`, `dd` to `NaN` when the `sd_vvp` for that height bin is below the `sd_vvp_threshold()`.

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`, list of `vp` or `vpts` object.
`quantity`	Character. A (case sensitive) profile quantity, one of: • `height`: Height bin (lower bound) in m above sea level. • `u`: Ground speed component west to east in m/s. • `v`: Ground speed component south to north in m/s. • `w`: Vertical speed (unreliable!) in m/s. • `ff`: Horizontal speed in m/s. • `dd`: Direction in degrees clockwise from north. • `sd_vvp`: VVP radial velocity standard deviation in m/s. • `gap`: Angular data gap detected in T/F.

- dbz: Animal reflectivity factor in dBZ.
- eta: Animal reflectivity in cm^2/km^3.
- dens: Animal density in animals/km^3.
- DBZH: Total reflectivity factor (bio + meteo scattering) in dBZ.
- n: Number of data points used for the ground speed estimates (quantities u, v, w, ff, dd).
- n_all: Number of data points used for the radial velocity standard deviation estimate (quantity sd_vvp).
- n_dbz: Number of data points used for reflectivity-based estimates (quantities dbz, eta, dens).
- n_dbz_all: Number of data points used for the total reflectivity estimate (quantity DBZH).
- attributes: List of the vertical profile's what, where and how attributes.

Value

the value of a specific profile quantity specified in quantity.

For a vp object: a named (height bin) vector with values for the selected quantity.

For a list object: a list of named (height bin) vectors with values for the selected quantity.

For a vpts object: a (height bin * datetime) matrix with values for the selected quantity.

See Also

- summary.vp()
- sd_vvp_threshold()<- for setting the sd_vvp threshold of an object.

Examples

```
# Extract the animal density (dens) quantity from a vp object
get_quantity(example_vp, "dens")

# Extract the horizontal ground speed (ff) quantity from a vpts object and show the
# first two datetimes
get_quantity(example_vpts, "ff")[,1:2]
```

---

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

Description

Returns the scan (scan) from a polar volume (pvol) with elevation angle closest to elev.

Usage

```
get_scan(x, elev, all = FALSE)
```

Arguments

x	A pvol object.
elev	Numeric. Elevation angle in degrees.
all	Logical. Return the first scan in the pvol object closest to the requested elevation (FALSE), or a list with all scans equally close to the requested elevation (TRUE).

Details

In cases where elev is exactly in between two scan elevation angles, the lower elevation angle scan is returned.

Value

A scan object when all equals FALSE (default), or a list of scan objects if all equals TRUE

See Also

- summary.scan()
- get_elevation_angles()

Examples

```
# Locate and read the polar volume example file
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")
pvol <- read_pvolfile(pvolfile)

# Get elevation angles
get_elevation_angles(pvol)

# Extract the scan closest to 3 degrees elevation (2.5 degree scan)
scan <- get_scan(pvol, 3)

# Get summary info
scan

# Extract all scans closest to 3 degrees elevation (2.5 degree scan)
# Always returns a list with scan object(s), containing multiple scans
# if the pvol contains multiple scans at the same closest elevation.
scan_list <- get_scan(pvol, 3)
scan_list
```

---

integrate_profile *Vertically integrate profiles (vp or vpts) into 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 = 3600,
  interval_replace = NA,
  height_quantile = NA
)

## S3 method for class 'vp'
integrate_profile(
  x,
  alt_min = 0,
  alt_max = Inf,
  alpha = NA,
  interval_max = 3600,
  interval_replace = NA,
  height_quantile = NA
)

## S3 method for class 'list'
integrate_profile(
  x,
  alt_min = 0,
  alt_max = Inf,
  alpha = NA,
  interval_max = 3600,
  interval_replace = NA,
  height_quantile = NA
)

## S3 method for class 'vpts'
integrate_profile(
  x,
  alt_min = 0,
  alt_max = Inf,
  alpha = NA,
  interval_max = 3600,
  interval_replace = NA,
  height_quantile = NA
)
```

Arguments

x	A vp or vpts object.
alt_min	Minimum altitude in m. "antenna" can be used to set the minimum altitude to

the height of the antenna.

`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`.

`interval_replace`
Time interval to use for any interval > interval_max. By default the mean of all intervals <= interval_max

`height_quantile`
For default `NA` the calculated height equals the mean flight altitude. Otherwise a number between 0 and 1 specifying a quantile of the height distribution.

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, cumulated from the start of the time series up to `datetime`
- `rt`: Reflectivity Traffic in cm^2/km, cumulated from the start of the time series up to `datetime`
- `ff`: Horizontal ground speed in m/s
- `dd`: Direction of the horizontal ground speed in degrees
- `u`: Ground speed component west to east in m/s
- `v`: Ground speed component south to north in m/s
- `height`: Mean flight height (height weighted by eta) in m above sea level

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:

$$mtr = 3.6 \sum_i dens_i ff_i \Delta h$$

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. The factor 3.6 refers to a unit conversion of speeds ff_i from m/s to km/h.

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:

$$mtr = 3.6 \sum_i dens_i ff_i \cos((dd_i - \alpha)\pi/180)\Delta h$$

with dd_i the migratory direction at altitude i.

Note that this equation evaluates to the previous equation when `alpha` equals dd_i. Also note we can rewrite this equation using trigonometry as:

$$mtr = 3.6 \sum_i dens_i (u_i \sin(\alpha\pi/180) + v_i \cos(\alpha\pi/180))\Delta h$$

with u_i and v_i the u and v ground speed components at altitude 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.

Note that `mtr` for a given value of `alpha` can also be calculated from the vertically integrated density `vid` and the height-integrated velocity components u and v as follows:

$$mtr = 3.6(u \sin(\alpha\pi/180) + v \cos(\alpha\pi/180))vid$$

Formula for reflectivity traffic rate `rtr` are found by replacing `dens` with `eta` and `vid` with `vir` 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. `mtr` values are conditional on settings of rcs, while `rtr` 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.

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

Columns 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.

Ground speed (ff) and ground speed components (u,v): The height-averaged ground speed is defined as:

$$ff = \sum_i dens_i ff_i / \sum_i dens_i$$

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.
the height-averaged u component (west to east) is defined as:

$$u = \sum_i dens_i u_i / \sum_i dens_i$$

the height-averaged v component (south to north) is defined as:

$$v = \sum_i dens_i v_i / \sum_i dens_i$$

Note that $ff_i = \sqrt{(u_i^2 + v_i^2)}$, but the same does not hold for the height-integrated speeds, i.e. $ff \neq \sqrt{(u^2 + v^2)}$ as soon as the ground speed directions vary with altitude.

Value

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

Methods (by class)

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

Examples

```
# Calculate migration traffic rates for a single vp
integrate_profile(example_vp)

# Calculate migration traffic rates for a list of vps
integrate_profile(c(example_vp, example_vp))

# Calculate migration traffic rates for a vpts
vpi <- integrate_profile(example_vpts)

# Plot migration traffic rate (mtr) for the full air column
plot(integrate_profile(example_vpts))

# Plot migration traffic rate (mtr) for altitudes > 1 km above sea level
plot(integrate_profile(example_vpts, alt_min = 1000))

# Plot cumulative migration traffic rates (mt)
plot(integrate_profile(example_vpts), quantity = "mt")
# calculate median flight altitude (instead of default mean)
integrate_profile(example_vp, height_quantile=.5)
# calculate the 90% percentile of the flight altitude distribution
integrate_profile(example_vpts, height_quantile=.9)
```

`integrate_to_ppi` *Calculate a plan position indicator (*`ppi`*) of vertically integrated density adjusted for range effects*

Description

Estimates a spatial image of vertically integrated density (`vid`) based on all elevation scans of the radar, while accounting for the changing overlap between the radar beams as a function of range. The resulting `ppi` is a vertical integration over the layer of biological scatterers based on all available elevation scans, corrected for range effects due to partial beam overlap with the layer of biological echoes (overshooting) at larger distances from the radar. The methodology is described in detail in Kranstauber et al. (2020).

Usage

```
integrate_to_ppi(
  pvol,
  vp,
  nx = 100,
  ny = 100,
  xlim,
  ylim,
  zlim = c(0, 4000),
  res,
  quantity = "eta",
  param = "DBZH",
  raster = NA,
  lat,
  lon,
  antenna,
  beam_angle = 1,
  crs,
  param_ppi = c("VIR", "VID", "R", "overlap", "eta_sum", "eta_sum_expected"),
  k = 4/3,
  re = 6378,
  rp = 6357
)
```

Arguments

`pvol`	A `pvol` object.
`vp`	A `vp` object
`nx`	number of raster pixels in the x (longitude) dimension
`ny`	number of raster pixels in the y (latitude) dimension
`xlim`	x (longitude) range

ylim	y (latitude) range
zlim	Numeric vector of length two. Altitude range, in m
res	numeric vector of length 1 or 2 to set the resolution of the raster (see res). If this argument is used, arguments nx and ny are ignored. Unit is identical to xlim and ylim.
quantity	Character. Profile quantity on which to base range corrections, either eta or dens.
param	reflectivity Character. Scan parameter on which to base range corrections. Typically the same parameter from which animal densities are estimated in vp. Either DBZH, DBZV, DBZ, TH, or TV.
raster	(optional) RasterLayer with a CRS. When specified this raster topology is used for the output, and nx, ny, res arguments are ignored.
lat	Latitude of the radar, in degrees. If missing taken from pvol.
lon	Latitude of the radar, in degrees. If missing taken from pvol.
antenna	Numeric. Radar antenna height, in m. Default to antenna height in vp.
beam_angle	Numeric. Beam opening angle in degrees, typically the angle between the half-power (-3 dB) points of the main lobe.
crs	character or object of class CRS. PROJ.4 type description of a Coordinate Reference System (map projection). When 'NA' (default), an azimuthal equidistant projection with origin at the radar location is used. To use a WSG84 (lat,lon) projection, use crs="+proj=longlat +datum=WGS84"
param_ppi	Character (vector). One or multiple of VIR, VID, R, overlap, eta_sum or eta_sum_expected.
k	Numeric. Standard refraction coefficient.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

The function requires:

- A polar volume, containing one or multiple scans (pvol).
- A vertical profile (of birds) calculated for that same polar volume (vp).
- A grid defined on the earth's surface, on which we will calculate the range corrected image (defined by raster, or a combination of nx, ny,res arguments).

The pixel locations on the ground are easily translated into a corresponding azimuth and range of the various scans (see beam_range()).

For each scan within the polar volume, the function calculates:

- the vertical radiation profile for each ground surface pixel for that particular scan, using beam_profile.
- the reflectivity expected for each ground surface pixel ($\eta_{expected}$), given the vertical profile (of biological scatterers) and the part of the profile radiated by the beam. This $\eta_{expected}$ is simply the average of (linear) eta in the profile, weighted by the vertical radiation profile.

- the observed eta at each pixel $\eta_{observed}$, which is converted form `DBZH` using function dbz_to_eta, with `DBZH` the reflectivity factor measured at the pixel's distance from the radar.

If one of `lat` or `lon` is missing, the extent of the `ppi` is taken equal to the extent of the data in the first scan of the polar volume.

To arrive at the final PPI image, the function calculates

- the vertically integrated density (`vid`) and vertically integrated reflectivity (`vir`) for the profile, using the function integrate_profile.
- the spatial range-corrected PPI for `VID`, defined as the adjustment factor image (`R`), multiplied by the `vid` calculated for the profile
- the spatial range-corrected PPI for `VIR`, defined as the adjustment factor `R`, multiplied by the `vir` calculated for the profile.

Scans at 90 degree beam elevation (e.g. birdbath scans) are ignored.

Value

A `ppi` object with the following parameters:

- `VIR`: the vertically integrated reflectivity in cm^2/km^2
- `VID`: the vertically integrated density in 1/km^2
- `R`: the spatial adjustment factor (unitless). See Kranstauber 2020 for details. Equal to `eta_sum/eta_sum_expected`.
- `overlap`: the distribution overlap between the vertical profile `vp` and the vertical radiation profile for the set of radar sweeps in `pvol`, as calculated with beam_profile_overlap.
- `eta_sum`: the sum of observed linear reflectivities over elevation angles. See Kranstauber 2020 for details.
- `eta_sum_expected`: the sum of expected linear reflectivities over elevation angles based on the input vertical profile `vp`. See Kranstauber 2020 for details.

References

- Kranstauber B, Bouten W, Leijnse H, Wijers B, Verlinden L, Shamoun-Baranes J, Dokter AM (2020) High-Resolution Spatial Distribution of Bird Movements Estimated from a Weather Radar Network. Remote Sensing 12 (4), 635. doi:10.3390/rs12040635
- Buler JJ & Diehl RH (2009) Quantifying bird density during migratory stopover using weather surveillance radar. IEEE Transactions on Geoscience and Remote Sensing 47: 2741-2751. doi:10.1109/TGRS.2009.2014463

- Kranstauber B, Bouten W, Leijnse H, Wijers B, Verlinden L, Shamoun-Baranes J, Dokter AM (2020) High-Resolution Spatial Distribution of Bird Movements Estimated from a Weather Radar Network. Remote Sensing 12 (4), 635. doi:10.3390/rs12040635
- Buler JJ & Diehl RH (2009) Quantifying bird density during migratory stopover using weather surveillance radar. IEEE Transactions on Geoscience and Remote Sensing 47: 2741-2751. doi:10.1109/TGRS.2009.2014463

See Also

- summary.ppi()
- beam_profile()
- beam_range()
- integrate_profile()

Examples

```
# Locate and read the polar volume example file
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")

# load polar volume
pvol <- read_pvolfile(pvolfile)

# Read the corresponding vertical profile example
data(example_vp)

# Calculate the range-corrected ppi on a 50x50 pixel raster
ppi <- integrate_to_ppi(pvol, example_vp, nx = 50, ny = 50)

# Plot the vertically integrated reflectivity (VIR) using a
# 0-2000 cm^2/km^2 color scale
plot(ppi, zlim = c(0, 2000))

# Calculate the range-corrected ppi on finer 2000m x 2000m pixel raster
ppi <- integrate_to_ppi(pvol, example_vp, res = 2000)

# Plot the vertically integrated density (VID) using a
# 0-200 birds/km^2 color scale
plot(ppi, param = "VID", zlim = c(0, 200))

# Download a basemap and map the ppi
if (all(sapply(c("ggspatial","prettymapr", "rosm"), requireNamespace, quietly = TRUE))) {
map(ppi)
}

# The ppi can also be projected on a user-defined raster, as follows:

# First define the raster
template_raster <- raster::raster(
  raster::extent(12, 13, 56, 57),
  crs = sp::CRS("+proj=longlat")
)

# Project the ppi on the defined raster
ppi <- integrate_to_ppi(pvol, example_vp, raster = template_raster)

# Extract the raster data from the ppi object
raster::brick(ppi$data)

# Calculate the range-corrected ppi on an even finer 500m x 500m pixel raster,
```

```
# cropping the area up to 50000 meter from the radar
ppi <- integrate_to_ppi(
  pvol, example_vp, res = 500,
  xlim = c(-50000, 50000), ylim = c(-50000, 50000)
)
plot(ppi, param = "VID", zlim = c(0, 200))
```

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

Description

Checks whether a file is a polar volume (`pvol`) in the ODIM HDF5 format that can be read with bioRad. Evaluates to `FALSE` for NEXRAD and IRIS RAW polar volume file (see `nexrad_to_odim()`).

Usage

```
is.pvolfile(file)
```

Arguments

`file`	Character. Path of the file to check.

Value

`TRUE` for a polar volume file in readable format, otherwise `FALSE`.

See Also

- `read_pvolfile()`
- `get_odim_object_type()`
- `is.pvol()`

Examples

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

# Check if it is a pvolfile
is.pvolfile(pvolfile)
```

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

Description

Checks whether a file is a vertical profile (vp) in the ODIM HDF5 format that can be read with bioRad.

Usage

```
is.vpfile(file)
```

Arguments

file	Character. Path of the file to check.

Value

TRUE for a vertical profile file in readable format, otherwise FALSE.

See Also

- read_vpfiles()
- get_odim_object_type()
- is.vp()

Examples

```
# Locate the vertical profile example file
vpfile <- system.file("extdata", "profile.h5", package = "bioRad")

# Check if it is a vpfile
is.vpfile(vpfile)
```

list_vpts_aloft — *List aloft urls for time series of vertical profiles (vpts) of radar stations*

Description

[Superseded]

This function has been superseded by getRad::get_vpts() and getRad::aloft_data_coverage().

Usage

```
list_vpts_aloft(
  date_min = NULL,
  date_max = NULL,
  radars = NULL,
  format = "csv",
  source = "baltrad",
  show_warnings = TRUE
)
```

Arguments

date_min	Character, the first date to return urls for. In the shape of YYYY-MM-DD.
date_max	Character, the last date to return urls for. In the shape of YYYY-MM-DD.
radars	Character vector, radar stations to return urls for.
format	Character, the format of archive urls to return, either csv or hdf5. Currently only csv urls are supported.
source	Character, either baltrad or ecog-04003
show_warnings	Logical, whether to print warnings for dates or radar stations for which no data was found.

Value

A character vector of aloft urls

Examples

```
if (requireNamespace("aws.s3", quietly = TRUE)) {
list_vpts_aloft(radars = "bejab", date_min='2018-10-01', date_max = '2018-12-31')
}
```

---

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

Description

Plots a plan position indicator (ppi) on a base layer

Usage

```
map(x, ...)

## S3 method for class 'ppi'
map(
  x,
  map = "cartolight",
```

```
  param,
  alpha = 0.7,
  xlim,
  ylim,
  zlim = c(-20, 20),
  ratio,
  radar_size = 3,
  radar_color = "#202020",
  n_color = 1000,
  palette = NA,
  zoomin = -2,
  cachedir = tools::R_user_dir("bioRad"),
  ...
)
```

Arguments

x	A ppi object.
...	Arguments passed to ggplot2::ggplot().
map	Basemap to use, one of rosm::osm.types()
param	Character. Scan parameter to plot, e.g. DBZH or VRADH. See summary.param() for commonly available parameters.
alpha	Numeric. Transparency of the data, value between 0 and 1.
xlim	Numeric vector of length 2. Range of x values (degrees longitude) to plot.
ylim	Numeric vector of length 2. Range of y values (degrees latitude) to plot.
zlim	Numeric vector of length 2. The range of values to plot.
ratio	Numeric. Aspect ratio between x and y scale, by default $1/cos(latituderadar * pi/180)$.
radar_size	Numeric. Size of the symbol indicating the radar position.
radar_color	Character. Color of the symbol indicating the radar position.
n_color	Numeric. Number of colors (>=1) to use in the palette.
palette	Character vector. Hexadecimal color values defining the plot color scale, e.g. output from viridisLite::viridis().
zoomin	Numeric. Maps to ggspatial::annotation_map_tile()
cachedir	Character. Maps to ggspatial::annotation_map_tile(), defaults to tools::R_user_dir("bioRad")

Details

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

- DBZH, DBZ: (Logged) reflectivity factor (dBZ)
- TH, T: (Logged) uncorrected 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 ggplot object

Methods (by class)

- map(ppi): Plot a ppi object on a map.

See Also

- project_as_ppi()

Examples

```
# Project a scan as a ppi
ppi <- project_as_ppi(example_scan)

if (all(sapply(c("ggspatial","prettymapr", "rosm"), requireNamespace, quietly = TRUE))) {
# Choose a basemap
basemap <- rosm::osm.types()[1]

# Map the radial velocity of the ppi onto the basemap
map(ppi, map = basemap, param = "VRADH")

# Extend the plotting range of velocities, from -50 to 50 m/s
map(ppi, map = basemap, param = "VRADH", zlim = c(-50, 50))

# Map the reflectivity
map(ppi, map = basemap, param = "DBZH")

# Change the color palette to Viridis colors
map(ppi, map = basemap, param = "DBZH", palette = viridis::viridis(100), zlim=c(-10,10))

# Give the data more transparency
map(ppi, map = basemap, param = "DBZH", alpha = 0.3)

# Change the appearance of the symbol indicating the radar location
map(ppi, map = basemap, radar_size = 5, radar_color = "blue")

# Crop the map
map(ppi, map = basemap, xlim = c(12.4, 13.2), ylim = c(56, 56.5))
}
```

`Math.scan` *Mathematical and arithmetic operations on param's, scan's and pvol's*

Description

Mathematical and arithmetic operations on param's, scan's and pvol's

Usage

```
## S3 method for class 'scan'
Math(x, ...)

## S3 method for class 'pvol'
Math(x, ...)

## S3 method for class 'param'
Ops(e1, e2)

## S3 method for class 'scan'
Ops(e1, e2)

## S3 method for class 'pvol'
Ops(e1, e2)
```

Arguments

`x`	object of class `scan`, or `pvol`
`...`	objects passed on to the Math functions
`e1`	object of class `param`, `scan`, `pvol` or a number
`e2`	object of class `param`, `scan`, `pvol` or a number

Details

Use caution when applying these manipulations, as there are no consistency checks if the operations lead to interpretable outcomes. For example, when averaging scans with logarithmic values (e.g. DBZ), it might be required to first exponentiate the data before summing.

Attributes are taken from the first object in the operation.

When a `pvol` is multiplied by a list, in which case arguments are taken from the list per scan. this requires the list to have the same length as the number of scans.

Value

an object of the input class

See Also

- `calculate_param()`

Examples

```
# Locate and read the polar volume example file
scan1 <- example_scan

#add a value of 1 to all scan parameters:
scan2 <- example_scan + 1

# average the scan parameters of two scans:
# NB: requires identical scan parameter names and order!
(scan1 + scan2)/2
```

`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)
```

Arguments

`pvolfile_nexrad`	Character (vector). Either a path to a single radar polar volume (`pvol`) file containing multiple scans/sweeps, or multiple paths to scan files containing a single scan/sweep. Or a single `pvol` object. The file data format should be either 1) ODIM format, which is the implementation of the OPERA data information model in the HDF5 format, 2) a format supported by the RSL library or 3) Vaisala IRIS (IRIS RAW) format.
`pvolfile_odim`	Filename for the polar volume in ODIM HDF5 format to be generated.
`verbose`	Logical. When `TRUE`, vol2bird `stdout` is piped to the R console.

Value

`TRUE` on success

Examples

```
# download a NEXRAD file, save as KBGM_example
path = file.path(tempdir(), "KBGM_example")

download.file(paste0("https://noaa-nexrad-level2.s3.amazonaws.com/",
  "2019/10/01/KBGM/KBGM20191001_000542_V06"), path, method="libcurl", mode="wb")

# convert to ODIM format
```

```
new_path = file.path(tempdir(), "KBGM_example.h5")

if (requireNamespace("vol2birdR", quietly = TRUE)) {
nexrad_to_odim(path, new_path)

# verify that we have generated a polar volume in ODIM HDF5 format
get_odim_object_type(new_path)

# clean up
file.remove(new_path)
}
file.remove(path)
```

`nyquist_velocity` *Calculate Nyquist velocity for a given pulse repetition frequency (PRF)*

Description

Calculates the Nyquist velocity given a radar's pulse repetition frequency (PRF) and wavelength. When specifying two PRFs, the extended Nyquist velocity is given for a radar using the dual-PRF technique.

Usage

```
nyquist_velocity(wavelength, prf1, prf2)
```

Arguments

`wavelength`	Numeric. Radar wavelength, in cm.
`prf1`	Numeric. Radar pulse repetition frequency, in Hz.
`prf2`	Numeric. Alternate radar pulse repetition frequency for a radar operating in dual-PRF mode, in Hz.

Value

Nyquist velocity, in m/s.

Examples

```
# Get Nyquist velocity at C-band (5.3 cm wavelength) and a PRF of 2000 Hz
nyquist_velocity(5.3, 2000)

# Get extended Nyquist velocity in a dual-PRF scheme using 2000 Hz and
# 1500 Hz PRFs
nyquist_velocity(5.3, 2000, 1500)
```

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

Description

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

Usage

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

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. Defaults to parameter specific limits for plotting, not full range of data.
ratio	Aspect ratio between x and y scale.
na.value	ggplot argument setting the plot color of NA values
...	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)
- TH, T: (Logged) uncorrected 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

No return value, side effect is a plot.

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 default scan parameter, which is reflectivity "DBZH":
plot(ppi)

# plot the radial velocity parameter:
plot(ppi, param = "VRADH")

# change the range of reflectivities to plot, from -10 to 10 dBZ:
plot(ppi, param = "DBZH", zlim = c(-10, 10))

# change the scale name and colour scheme, using viridis colors:
plot(ppi, param = "DBZH", zlim = c(-10, 10)) + viridis::scale_fill_viridis(name = "dBZ")
```

---

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

Description

Plots a scan (scan) in polar coordinates. To plot in Cartesian coordinates, see project_as_ppi().

Usage

```
## S3 method for class 'scan'
plot(
  x,
  param,
  xlim = c(0, 1e+05),
  ylim = c(0, 360),
  zlim = c(-20, 20),
  na.value = "transparent",
```

```
  ...
)
```

Arguments

x	A scan object.
param	Character. Scan parameter to plot, e.g. DBZH or VRADH. See summary.param() for commonly available parameters.
xlim	Numeric vector of length 2. Range of x values (range, distance to radar) to plot.
ylim	Numeric vector of length 2. Range of y values (azimuth) to plot.
zlim	Numeric vector of length 2. The range of parameter values to plot. Defaults to parameter specific limits for plotting, not full range of data.
na.value	Character. ggplot2::ggplot() parameter to set the color of NA values.
...	Arguments passed to ggplot2::ggplot().

Details

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

- DBZH, DBZ: (Logged) reflectivity factor (dBZ)
- TH, T: (Logged) uncorrected 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

No return value, side effect is a plot.

Examples

```
# Plot reflectivity
plot(example_scan, param = "DBZH")


# Change the range of reflectivities to plot, from -10 to 10 dBZ
plot(example_scan, param = "DBZH", zlim = c(-10, 10))

# Change the scale name, change the color palette to Viridis colors
plot(example_scan, param = "DBZH", zlim = c(-10, 10)) +
  viridis::scale_fill_viridis(name = "dBZ")
```

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.

Value

No return value, side effect is a plot.

Examples

```
# load example vp object:
data(example_vp)

# plot the animal density:
plot(example_vp, quantity = "dens")

# change the line color:
plot(example_vp, line_col = "blue")

# plot the ground speed:
plot(example_vp, quantity = "ff")

# plot the reflectivity factor of
# all scatterers (including precipitation):
plot(example_vp, quantity = "DBZH")
```

---

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 ground speed)
- `dd` (height-averaged direction)
- `u` (height-averaged u-component of ground speed),
- `v` (height-averaged v-component of ground 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 sunrise.
`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 south to north in m/s
- `height`: Mean flight height (height weighted by reflectivity eta) in m above sea level The height-averaged ground speed quantities (ff,dd,u,v) and height are weighted averages by reflectivity eta.

Value

No return value, side effect is a plot.

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 = NA,
  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,
  na_color = "#C8C8C8",
  nan_color = "white",
  n_color = 1000,
  palette = NA,
  ...
)
```

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.
na_color	Color to use for NA values, see class vpts() conventions.
nan_color	Color to use for NaN values, see class vpts() conventions.
n_color	The number of colors (>=1) to be in the palette.
palette	(Optional) character vector of hexadecimal color values defining the plot color scale, e.g. output from viridis
...	Additional arguments to be passed to the low level image plotting function.

Details

Aerial abundances can be visualized in four 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)

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

- ff: ground speed. The aerial velocity relative to the ground surface in m/s.
- u: eastward ground speed component in m/s.
- v: northward ground speed component in m/s.

barbs:

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

legend_ticks / zlim:

Default legend ticks and plotting range are specified based on quantity, radar wavelength (S- vs C-band), and value of log

log:

Quantities u and v cannot be plotted on a logarithmic scale, because these quantities assume negative values. For quantities DBZH and dbz log=TRUE is ignored, because these quantities are already logarithmic.

Value

No return value, side effect is a plot.

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")
# regularize the time grid, which includes empty (NA) profiles at
# time steps without data:
ts_regular <- regularize_vpts(ts)
plot(ts_regular)
# change the color of missing NA data to red
plot(ts_regular, na_color="red")
# change the color palette:
plot(ts_regular[1:1000], ylim = c(0, 3000), palette=viridis::viridis(1000))
# change and inverse the color palette:
plot(ts_regular[1:1000], ylim = c(0, 3000), palette=rev(viridis::viridis(1000, option="A")))
# plot the speed profile:
plot(ts_regular[1:1000], quantity="ff")
# plot the northward speed component:
plot(ts_regular[1:1000], quantity="v")
# plot speed profile with more legend ticks,
plot(ts_regular[1:1000], quantity="ff", legend_ticks=seq(0,20,2), zlim=c(0,20))
```

`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 = TRUE,
  ylim = NULL,
  xlim = NULL,
  raster = NA,
  k = 4/3,
  re = 6378,
  rp = 6357
)

## S3 method for class 'param'
project_as_ppi(
  x,
  grid_size = 500,
  range_max = 50000,
  project = TRUE,
  ylim = NULL,
  xlim = NULL,
  raster = NA,
  k = 4/3,
  re = 6378,
  rp = 6357
)

## S3 method for class 'scan'
project_as_ppi(
  x,
  grid_size = 500,
  range_max = 50000,
  project = TRUE,
  ylim = NULL,
  xlim = NULL,
  raster = NA,
  k = 4/3,
```

```
  re = 6378,
  rp = 6357
)
```

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.
raster	(optional) RasterLayer with a CRS. When specified this raster topology is used for the output, and grid_size, range_max, xlim, ylim are ignored.
k	Numeric. Standard refraction coefficient.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

The returned PPI is in Azimuthal Equidistant Projection.

Value

An object of class 'ppi'.

Methods (by class)

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

Examples

```
# load a polar scan example object:
data(example_scan)
example_scan

# plot the scan:
plot(example_scan)

# make PPIs for all scan parameters in the scan:
ppi <- project_as_ppi(example_scan)

# print summary info for the ppi:
ppi

# plot the ppi:
plot(ppi)
```

```
# extract the DBZH scan parameter of the volume to a new
# object 'param':
param <- get_param(example_scan, "VRADH")

# make a ppi for the new 'param' object:
ppi <- project_as_ppi(param)

# print summary info for this ppi:
ppi

# plot the ppi:
plot(ppi)
```

`rcs` *Get radar cross section*

Description

Returns the currently assumed radar cross section of an object 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`, `vpts` or `vpi` object.

Value

The radar cross section in cm^2.

See Also

- `rcs()<-` for setting the radar cross section of an object.
- `sd_vvp_threshold()`

Examples

```
# Get the radar cross section for a vp
rcs(example_vp)

# Get the radar cross section for a vpts
rcs(example_vpts)

# Get the radar cross section for a vpi
vpi <- integrate_profile(example_vpts)
rcs(vpi)
```

---

rcs<- *Set radar cross section*

Description

Sets the assumed radar cross section of an object in cm^2. This function also updates the migration densities in `x$data$dens` to `eta/rcs` when above `sd_vvp_threshold` and `0` if below.

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`, `vpts` or `vpi` object.
`value`	Numeric. The radar cross section value to assign in cm^2.

Value

The input object with updated density `x$data$dens` and updated radar cross section attribute.

See Also

- `rcs()` for getting the radar cross section of an object.
- `sd_vvp_threshold()<-`

Examples

```
# Set the radar cross section for a vp
vp <- example_vp
rcs(vp) <- 11

# Set the radar cross section for a vpts
vpts <- example_vpts
rcs(vpts) <- 11

# Set the radar cross section for a vpi
vpi <- integrate_profile(example_vpts)
rcs(vpi) <- 11
```

`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`	Character. Path to a text file containing the standard output (`stdout`) generated by UMASS Cajun pipeline.
`rcs`	Numeric. Radar cross section per bird in cm^2.
`wavelength`	Character or numeric. Radar wavelength, either `C` for C-band (5.3 cm), `S` for S-band (10.6 cm) or in cm.

Value

A vp object.

See Also

- `summary.vp()`

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

Description

Read a polar volume (`pvol`) from file

Usage

```
read_pvolfile(
  file,
  param = c("DBZH", "DBZ", "VRADH", "VRAD", "WRADH", "WRAD", "TH", "T", "RHOHV", "ZDR",
    "PHIDP", "CELL", "BIOLOGY", "WEATHER", "BACKGROUND"),
  sort = TRUE,
  lat,
  lon,
  height,
  elev_min = 0,
  elev_max = 90,
  verbose = TRUE,
  mount = dirname(file),
  local_install
)
```

Arguments

`file`	A string containing the path to a polar volume file
`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`	(deprecated) A character string with the mount point (a directory path) for the Docker container.
`local_install`	(deprecated) String with path to local vol2bird installation, to use local installation instead of 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)
- TH, T: (Logged) uncorrected 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:
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")

# print the local path of the volume file:
pvolfile

# load the file:
example_pvol <- read_pvolfile(pvolfile)

# print summary info for the loaded polar volume:
example_pvol

# print summary info for the scans in the polar volume:
example_pvol$scans

# copy the first scan to a new object 'scan'
scan <- example_pvol$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

```
# 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)

# to load multiple files at once:
read_vpfiles(c(vpfile, vpfile))
```

---

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

Description

Reads vpts data from one or more files. The following file formats are supported (but cannot be mixed):

- VPTS CSV.
- ODIM bird profile.
- vol2bird standard output (see example below).

Usage

```
read_vpts(files, data_frame = FALSE, ...)
```

Arguments

files	Path(s) to one or more files containing vpts data.
data_frame	When FALSE (default) output a vpts object, when TRUE output a data.frame
...	Additional arguments for backward compatibility, passed to read_stdout.

Value

vpts object.

Examples

```
## read a vertical profile time series in VPTS CSV format:
vptsfile <- system.file("extdata", "example_vpts.csv", package = "bioRad")
read_vpts(vptsfile)
# read a single vertical profile file in ODIM h5 format:
vpfile <- system.file("extdata", "profile.h5", package = "bioRad")
read_vpts(vpfile)
# read a vertical profile time series in `vol2bird` stdout format:
stdout_file <- system.file("extdata", "example_vpts.txt", package = "bioRad")
read_vpts(stdout_file, radar = "KBGM", wavelength = "S")
```

---

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,
  date_max,
  units = "secs",
  fill = TRUE,
  verbose = TRUE,
  keep_datetime = FALSE
)
```

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 by default'.
date_max	End time of the projected time series, as a POSIXct object. Taken from ts by default.
units	Optional units of interval and fill, one of 'secs', 'mins', 'hours','days', 'weeks'. Defaults to 'mins'.

fill	Numeric or Logical. fill each regularized timestep with the closest original profile found within a time window of +/- fill. When TRUE, fill maps to interval, filling single missing timesteps. When FALSE, fill maps to 0, disabling filling.
verbose	Logical, when TRUE prints text to console.
keep_datetime	Logical, when TRUE keep original radar acquisition timestamps.

Details

Projects objects of class vpts on a regular time grid, and fills temporal gaps by nearest neighbor interpolation.

Irregular time series of profiles are typically aligned on a regular time grid with the expected time interval at which a radar provides data. Alignment is performed using a nearest neighbor interpolation limited to neighboring profiles that fall within +/- fill (centered) of an original profile.

Remaining temporal gaps in the time series are filled with empty profiles that have values NA for all quantities, such that each timestamp of the regular grid has an associated profile.

In plots of regular time series (see plot.vpts()) temporal gaps of missing profiles (e.g. due to radar down time) become visible, as a result of the gap filling with empty profiles. 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.

When keep_datetime is TRUE the original profile timestamps are kept in ts$datetime. This may lead to duplicate timestamps when regularizing on a timegrid finer than the interval of available profiles.

Value

An object of class vpts with regular time steps.

Examples

```
# start form example vpts object:
data(example_vpts)
ts <- example_vpts
# data gaps are not visible:
plot(ts)

# regularize the time series on a 5 minute interval grid
tsRegular <- regularize_vpts(ts, interval = 300)
# data gaps are visible:
plot(tsRegular)

# regularize the time series on a 10 minute interval grid,
# and fill data gaps smaller than 1 hour by nearest neighbor interpolation
tsRegular <- regularize_vpts(ts, interval = 600, fill = 3600)
# data gaps are smaller as a result of nearest neighbor interpolation:
plot(tsRegular)
```

scan_to_raster *convert a polar scan into a raster*

Description

convert an object of class 'scan' into a raster of class 'RasterBrick'

Usage

```
scan_to_raster(
  scan,
  nx = 100,
  ny = 100,
  xlim,
  ylim,
  res = NA,
  param,
  raster = NA,
  lat,
  lon,
  crs = NA,
  k = 4/3,
  re = 6378,
  rp = 6357
)
```

Arguments

scan	a scan (sweep) of class scan
nx	number of raster pixels in the x (longitude) dimension
ny	number of raster pixels in the y (latitude) dimension
xlim	x (longitude) range
ylim	y (latitude) range
res	numeric vector of length 1 or 2 to set the resolution of the raster (see res). If this argument is used, arguments nx and ny are ignored. Unit is identical to xlim and ylim.
param	scan parameters to include. If NA include all scan parameters. Reducing the number of scan parameters speeds up evaluation.
raster	(optional) RasterLayer with a CRS. When specified this raster topology is used for the output, and nx, ny, res arguments are ignored.
lat	Geodetic latitude of the radar in degrees. If missing taken from scan.
lon	Geodetic longitude of the radar in degrees. If missing taken from scan.

crs	character or object of class CRS. PROJ.4 type description of a Coordinate Reference System (map projection). When 'NA' (default), an azimuthal equidistant projection with origin at the radar location is used. To use a WSG84 (lat,lon) projection, use crs="+proj=longlat +datum=WGS84"
k	Numeric. Standard refraction coefficient.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

uses scan_to_spatial to georeference the scan's pixels. If multiple scan pixels fall within the same raster pixel, the last added pixel is given (see rasterize for details).

Value

a RasterBrick

Examples

```
# default projects full extent on 100x100 pixel raster:
scan_to_raster(example_scan)

# crop the scan and project at a resolution of 0.1 degree:
scan_to_raster(example_scan, ylim = c(55, 57), xlim = c(12, 13), res = .1)

# using a template raster
template_raster <- raster::raster(raster::extent(12, 13, 56, 58), crs = sp::CRS("+proj=longlat"))
scan_to_raster(example_scan, raster = template_raster)
```

---

scan_to_spatial *convert a polar scan into a spatial object.*

Description

Georeferences the center of pixels for a scan into a SpatialPointsDataFrame object.

Usage

```
scan_to_spatial(scan, lat, lon, k = 4/3, re = 6378, rp = 6357)
```

Arguments

scan	a scan (sweep) of class scan
lat	Geodetic latitude of the radar in degrees. If missing taken from scan.
lon	Geodetic longitude of the radar in degrees. If missing taken from scan.
k	Numeric. Standard refraction coefficient.
re	Numeric. Earth equatorial radius, in km.
rp	Numeric. Earth polar radius, in km.

Details

Beam altitude accounts for the curvature of the earth, using beam_height. Distance from the radar over the earth's surface is calculated using beam_distance.

Value

a SpatialPointsDataFrame

Examples

```
# load example scan:
data(example_scan)

# convert to a SpatialPointsDataFrame:
scan_to_spatial(example_scan)
```

---

sd_vvp_threshold *Get threshold of the radial velocity standard deviation*

Description

Returns the current threshold of the radial velocity standard deviation (sd_vvp) of an object in m/s, retrieved by velocity volume processing (VVP).

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

The sd_vvp threshold in m/s.

See Also

- sd_vvp_threshold()<- for setting the sd_vvp threshold of an object.
- rcs()

Examples

```
# Get the sd_vvp threshold for a vp
sd_vvp_threshold(example_vp)

# Get the sd_vvp threshold for a vpts
sd_vvp_threshold(example_vpts)
```

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

Description

Sets the threshold of radial velocity standard deviation (`sd_vvp`) of an object in m/s. Altitude layers with `sd_vvp` below this threshold are assumed to have an aerial density of zero individuals. This function also updates the migration densities in `x$data$dens` to `eta/rcs` when above `sd_vvp_threshold` and `0` if below.

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`	Numeric. The `sd_vvp` threshold value to assign in m/s.

Value

The input object with updated density `x$data$dens` and `sd_vvp_thresh` attribute.

See Also

- `sd_vvp_threshold()` for getting the `sd_vvp` threshold of an object.
- `rcs()<-`

Examples

```
# Set the sd_vvp threshold for a vp
vp <- example_vp
sd_vvp_threshold(vp) <- 2

# Set the sd_vvp threshold for a vpts
vpts <- example_vpts
sd_vvp_threshold(vpts) <- 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.

See Also

download_vpfiles

Examples

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

`summary.param` *Inspect a parameter (*`param`*)*

Description

R base functions for inspecting a parameter (`param`) object.

Usage

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

is.param(x)
```

Arguments

`object`	A `param` object.
`...`	Additional arguments affecting the summary produced.
`x`	A `param` object.

Details

A parameter is a quantity/variable measured by the radar during a scan (or sweep). These are organized along radar range (bins) and azimuth (rays). 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 in dBZ.
- `TH`, `T`: (Logged) uncorrected reflectivity factor in dBZ.
- `VRADH`, `VRAD`: Radial velocity in m/s. Radial velocities towards the radar are negative, while radial velocities away from the radar are positive.
- `RHOHV`: Correlation coefficient (unitless). Correlation between the vertically and horizontally polarized reflectivity factor.
- `PHIDP`: Differential phase in degrees.
- `ZDR`: (Logged) differential reflectivity in dB.

Value

For is.param(): TRUE for an object of class param, otherwise FALSE.

See Also

- get_param()

Examples

```
# Extract the DBZH parameter from a scan
param <- get_param(example_scan, "DBZH")

# Check if it is an object of class param
is.param(param)

# Get summary info for this parameter
param # Same as summary(param) or print(param)
```

---

summary.ppi *Inspect a plan position indicator (ppi)*

Description

R base functions for inspecting a plan position indicator (ppi) object.

Usage

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

is.ppi(x)

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

Arguments

object	A ppi object.
...	Additional arguments affecting the summary produced.
x	A ppi object.

Details

A plan position indicator is a projection of radar data onto the earth's surface, generated from a single scan (scan) with project_as_ppi(), a polar volume (pvol) with integrate_to_ppi() or multiple plan position indicators (ppi) with composite_ppi(). A plan position indicator (ppi) object is a list containing:

- radar: Radar identifier.
- datetime: Nominal time of the volume to which the scan belongs in UTC.
- data: A sp::SpatialGridDataFrame containing the georeferenced data. See summary.param() for commonly available parameters, such as DBZH.
- geo: List of the scan's geographic properties (see the geo element in summary.scan()), with two additional properties:
  - bbox: Bounding box for the plan position indicator in decimal degrees.
  - merged: Logical. Flag to indicate if a plan position indicator is a composite of multiple scans. TRUE if generated with integrate_to_ppi() or composite_ppi().

Value

For is.ppi(): TRUE for an object of class ppi, otherwise FALSE.

For dim.ppi(): number of parameters (param), x and y pixels in a plan position indicator (ppi).

See Also

- project_as_ppi()
- integrate_to_ppi()
- plot.ppi()
- map()
- composite_ppi()
- [ppi()

Examples

```
# Project a scan as a ppi
ppi <- project_as_ppi(example_scan)

# Check if it is an object of class ppi
is.ppi(ppi)

# Get summary info
ppi # Same as summary(ppi) or print(ppi)

# Get dimensions
dim(ppi)
```

summary.pvol *Inspect a polar volume (*pvol*)*

Description

R base functions for inspecting a polar volume (pvol) object.

Usage

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

is.pvol(x)

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

Arguments

object	A pvol object.
...	Additional arguments affecting the summary produced.
x	A pvol object.

Details

A polar volume consists of a number of scans (or sweeps) made by the radar at different elevation angles. A polar volume (pvol) object is a list containing:

- radar: Radar identifier.
- datetime: Nominal time of the volume in UTC.
- scans: List of scans (scan) at different elevation angles.
- attributes: List of the volume's what, where and how attributes.
- geo: List of the volume's geographic properties:
  - lat: Latitude of the radar in decimal degrees.
  - lon: Longitude of the radar in decimal degrees.
  - height: Height of the radar antenna in meters above sea level.

Value

For is.pvol(): TRUE for an object of class pvol, otherwise FALSE.

For dim.pvol(): number of scans (scan) in a polar volume (pvol).

See Also

- read_pvolfile()
- get_elevation_angles()
- get_scan()

Examples

```
# Locate and read the polar volume example file
pvolfile <- system.file("extdata", "volume.h5", package = "bioRad")
pvol <- read_pvolfile(pvolfile)

# Check if it is an object of class pvol
is.pvol(pvol)

# Get summary info
pvol # Same as summary(pvol) or print(pvol)

# Get dimensions
dim(pvol)

# Get summary info for the scans in the polar volume
pvol$scans
```

---

summary.scan *Inspect a scan (*scan*)*

Description

R base functions for inspecting a scan (scan) object.

Usage

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

is.scan(x)

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

Arguments

object	A scan object.
...	Additional arguments affecting the summary produced.
x	A scan object.

Details

A scan (or sweep) is made by the radar at a certain elevation angle. The resulting parameter data (`param`) are organized along radar range (bins) and azimuth (rays). A scan (`scan`) object is a list containing:

- `radar`: Radar identifier.
- `datetime`: Nominal time of the volume to which the scan belongs in UTC.
- `params`: List of scan parameters (`param`).
- `attributes`: List of the scan's `what`, `where` and `how` attributes.
- `geo`: List of the scan's geographic properties:
  - `lat`: Latitude of the radar in decimal degrees.
  - `lon`: Longitude of the radar in decimal degrees.
  - `height`: Height of the radar antenna in meters above sea level.
  - `elange`: Elevation angle of the radar beam for that scan in degrees.
  - `rscale`: Range bin size for that scan in m (e.g. 500 m * 480 bins equals 240 km range).
  - `ascale`: Azimuth bin size for that scan in degrees (e.g. 1 degree * 360 rays equals full circle).
  - `rstart`: The range where the first range gate starts in meters (note ODIM stores it as kilometers)
  - `astart`: The start of the first ray.

Value

For `summary.scan()`: prints a summary of the `scan` object

For `is.scan()`: `TRUE` for an object of class `scan`, otherwise `FALSE`.

For `dim.scan()`: number of parameters (`param`), bins and rays in a scan (`scan`).

See Also

- `get_scan()`
- `example_scan`
- `plot.scan()`
- `get_param()`

Examples

```
# Check if an object is of class scan
is.scan(example_scan)

# Get summary info
example_scan # Same as summary(example_scan) or print(example_scan)

# Get dimensions
dim(example_scan)

# Get summary info for the parameters in the scan
example_scan$params
```

summary.vp *Inspect a vertical profile (*vp*)*

Description

R base functions for inspecting a vertical profile of biological targets (vp) object.

Usage

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

is.vp(x)

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

Arguments

object	A vp object.
...	Additional arguments affecting the summary produced.
x	A vp object.

Details

A vertical profile of biological targets contains a collection of quantities, organized in different (typically equally spaced) altitude layers (height bins) above the earth's surface. A vertical profile (vp) object is a list containing:

- radar: Radar identifier.
- datetime: Nominal time of the volume to which the scan belongs in UTC.
- data: A data.frame with the profile's quantities organized per height bin. Use get_quantity() to access these:
  - height: Height bin (lower bound) in m above sea level.
  - u: Ground speed component west to east in m/s.
  - v: Ground speed component south to north in m/s.
  - w: Vertical speed (unreliable!) in m/s.
  - ff: Horizontal ground speed in m/s.
  - dd: Ground speed direction in degrees clockwise from north.
  - sd_vvp: VVP radial velocity standard deviation in m/s.
  - gap: Angular data gap detected in T/F.
  - dbz: Animal reflectivity factor in dBZ.
  - eta: Animal reflectivity in cm^2/km^3.
  - dens: Animal density in animals/km^3.

  - DBZH: Total reflectivity factor (bio + meteo scattering) in dBZ.
  - n: Number of data points used for the ground speed estimates (quantities u, v, w, ff, dd).
  - n_all: Number of data points used for the radial velocity standard deviation estimate (quantity sd_vvp).
  - n_dbz: Number of data points used for reflectivity-based estimates (quantities dbz, eta, dens).
  - n_dbz_all: Number of data points used for the total reflectivity estimate (quantity DBZH).
- attributes: List of the vertical profile's what, where and how attributes.

Value

For summary.vp(): prints summary of the vp object.

For is.vp(): TRUE for an object of class vp, otherwise FALSE.

For dim.vp(): number of heights and quantities in a vertical profile (vp).

Conventions

- NA: Maps to nodata in the ODIM convention: value to denote areas void of data (never radiated).
- NaN: Maps to undetect in the ODIM convention: denote areas below the measurement detection threshold (radiated but nothing detected). The value is also used when there are too few datapoints to calculate a quantity.
- 0: Maps to 0 in the ODIM convention: denote areas where the quantity has a measured value of zero (radiated and value zero detected or inferred).

It depends on a radar's detection threshold or signal to noise ratio whether it safe to assume an undetect is equivalent to zero. When dealing with close range data only (within 35 km), it is typically safe to assume aerial densities (dens) and reflectivities (eta) are in fact zero in case of undetects.

See Also

- calculate_vp()
- read_vpfiles()
- example_vp
- get_quantity()
- plot.vp()
- as.data.frame.vp()
- bind_into_vpts()

Examples

```
# Check if an object is of class vp
is.vp(example_vp)

# Get summary info
```

```
example_vp # Same as summary(example_vp) or print(example_vp)

# Get dimensions
dim(example_vp)
```

---

summary.vpi *Inspect an integrated profile (*vpi*)*

Description

R base functions for inspecting an integrated profile of biological targets (vpi) object.

Usage

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

is.vpi(x)
```

Arguments

object	A vpi object.
...	Additional arguments affecting the summary produced.
x	A vpi object.

Details

A integrated profile of biological targets is a specially classed data.frame generated by the function integrate_profile() with the following quantities:

- radar: Radar identifier.
- 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, cumulated from the start of the time series up to datetime.
- rt: Reflectivity Traffic in cm^2/km, cumulated from the start of the time series up to datetime.
- 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 south to north in m/s.
- height: Mean flight height (height weighted by eta) in m above sea level.

Value

For summary.vpi(): prints summary of the vpi object.

For is.vpi(): TRUE for an object of class vpi, otherwise FALSE.

See Also

- integrate_profile()
- plot.vpi()

Examples

```
# Load the example vertical profile time series and integrate to a vpi
vpi <- integrate_profile(example_vpts)

# Check if it is an object of class vpi
is.vpi(vpi)

# Get summary info
summary(vpi)
```

---

summary.vpts *Inspect a time series of vertical profiles (*vpts*)*

Description

R base functions for inspecting a time series of vertical profiles (vpts) object.

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

Usage

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

## S3 method for class 'vpts'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.vpts(x)

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

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

Arguments

object	A vpts object.
...	Additional arguments affecting the summary produced.
x	A vpts object.
digits	The number of significant digits to use when printing. Defaults to max(3L, getOption("digits") - 3L).
i	Integer. Index/indices specifying which range of vertical profiles to extract.

Details

A time series of vertical profiles contains time-ordered vertical profiles (vp) of a single radar. This time series can be **regular** (vp are equally spaced in time) or **irregular** (time steps between vp are of unequal length), indicated in the field regular. Irregular time series can be projected onto a regular time grid with regularize_vpts(). A time series of vertical profile (vp) object is a list containing:

- radar: Radar identifier.
- datetime: Nominal times of the profiles (named dates in bioRad < 0.4.0) in UTC.
- height: Lowest height of the height bins in the profiles in m above sea level.
- daterange: Minimum and maximum nominal time of the profiles in UTC.
- timesteps: Time differences between the profiles. Element i gives the difference between profile i and i+1.
- data: A list of quantities, each containing a datetime by height matrix with the values. Use get_quantity() to access these and see summary.vp() for a description of available quantities.
- attributes: List of the vertical profile's what, where, and how attributes, copied from the first profile.
- regular: Logical indicating whether the time series is regular or not.

Value

For summary.vpts(): prints the summary of thevpts object.

For print.vpts(): prints the summary of the vpts object.

For is.vpts(): TRUE for an object of class vpts, otherwise FALSE.

For dim.vpts(): number of datetimes, heights and quantities in a time series of vertical profiles (vpts).

For [.vpts: A vpts object containing a subset of vertical profiles (vp) or a vp object when subsetting a single vertical profile (vp).

Conventions

- NA: Maps to nodata in the ODIM convention: value to denote areas void of data (never radiated).

- NaN: Maps to undetect in the ODIM convention: denote areas below the measurement detection threshold (radiated but nothing detected). The value is also used when there are too few datapoints to calculate a quantity.
- 0: Maps to 0 in the ODIM convention: denote areas where the quantity has a measured value of zero (radiated and value zero detected or inferred).

See Also

- bind_into_vpts()
- read_vpts()
- filter_vpts()
- regularize_vpts()
- example_vpts
- get_quantity()
- plot.vp()
- as.data.frame.vpts()
- [vpts()

Examples

```
# Check if an object is of class vpts
is.vpts(example_vpts)

# Get summary info
example_vpts # Same as summary(example_vpts) or print(example_vpts)

# Get dimensions
dim(example_vpts)
# The example vpts contains 1934 profiles (i.e. datetimes)
dim(example_vpts)

# Subset vpts to extract 10th profile
example_vpts[10] # A vp object

# Subset vpts to extract the 20th to 100th profile
example_vpts[20:100] # A vpts object with 81 profiles

# Subset vpts to remove the first 10 profiles
example_vpts[-1:-10] # A vpts object with 10 less profiles
```

---

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", force_tz = FALSE)

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

Arguments

date	POSIXct. Date interpretable by base::as.Date().
lon	Numeric. Longitude in decimal degrees.
lat	Numeric. Latitude in decimal degrees.
elev	Numeric. Sun elevation in degrees.
tz	Character. Time zone of date, ignored if date already has an associated time zone.
force_tz	Logical. If TRUE, the output is converted to the timezone set by tz.

Details

The day for which sunrise and sunset are calculated is given by the input date. Sunrise and sunset are calculated relative to the moment of solar noon for that date, i.e. the first sunrise before the moment of solar noon, and the first sunset after the moment of solar noon. Therefore, depending on the timezone provided, it is possible that the nearest sunrise prior to solar noon occurs a day earlier than the input date. Similarly, sunset may occur a day later than the input date. See examples for 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 suntools::crepuscule.

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

If force_tz is TRUE, the output is converted to the timezone set by tz

The day for which sunrise and sunset are calculated is given by the input date. Sunrise and sunset are calculated relative to the moment of solar noon for that date, i.e. the first sunrise before the moment of solar noon, and the first sunset after the moment of solar noon. Therefore, depending on the timezone provided, it is possible that the nearest sunrise prior to solar noon occurs a day earlier than the input date. Similarly, sunset may occur a day later than the input date. See examples for 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 suntools::crepuscule.

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

If force_tz is TRUE, the output is converted to the timezone set by tz

Value

The moment of sunrise or sunset for the date set by dateand time zone as specified (by date and tz) or in UTC if not specified.

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
sunrise("2016-01-01", -76.5, 42.4, elev = -6)

# next sunset in South Dakota, USA
sunset("2016-11-15", -98, 45)

# Beware that some days have two sunsets, or
# two sunrises! E.g. on 5 Oct (local timezone) at
# this location  sunset is actually on the 6 Oct
# in UTC time zone, i.e. the next day
sunset("2016-10-5", -98, 45)
# One day later, sunset is again on 6 Oct:
sunset("2016-10-6", -98, 45)

# working in local time zones typically avoids such ambiguities:
sunset(lubridate::as_datetime("2016-06-05",tz="America/Chicago"), -98, 45)
sunset(lubridate::as_datetime("2016-06-06",tz="America/Chicago"), -98, 45)

# use force_tz to force output to a specific time zone, by default UTC:
sunset(lubridate::as_datetime("2016-06-05",tz="America/Chicago"), -98, 45, force_tz=TRUE)
sunset(lubridate::as_datetime("2016-06-06",tz="America/Chicago"), -98, 45, force_tz=TRUE)

# Also beware of jumps in sunrise and sunset date with longitude:
sunrise("2016-11-01", 100, 45)
sunrise("2016-11-01", 102, 45)

# 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
sunrise("2016-01-01", -76.5, 42.4, elev = -6)

# Next sunset in South Dakota, USA
sunset("2016-11-15", -98, 45)

# Beware that some days have two sunsets, or two sunrises! E.g. on 5 Oct
# (local timezone) at this location sunset is actually on the 6 Oct in UTC,
```

```
# i.e. the next day
sunset("2016-10-5", -98, 45)
# One day later, sunset is again on 6 Oct
sunset("2016-10-6", -98, 45)

# Working in local time zones typically avoids such ambiguities
sunset(lubridate::as_datetime("2016-06-05", tz = "America/Chicago"), -98, 45)
sunset(lubridate::as_datetime("2016-06-06", tz = "America/Chicago"), -98, 45)

# Use force_tz to force output to a specific time zone, by default UTC
sunset(lubridate::as_datetime("2016-06-05", tz = "America/Chicago"), -98, 45, force_tz = TRUE)
sunset(lubridate::as_datetime("2016-06-06", tz = "America/Chicago"), -98, 45, force_tz = TRUE)

# Also beware of jumps in sunrise and sunset date with longitude
sunrise("2016-11-01", 100, 45)
sunrise("2016-11-01", 102, 45)
```

`write_pvolfile` *Write a polar volume (*`pvol`*) object to ODIM HDF5 file*

Description

Write a polar volume (`pvol`) object to ODIM HDF5 file

Usage

```
write_pvolfile(pvol, file, overwrite = FALSE, infer_dtype = FALSE)
```

Arguments

`pvol`	An object of class `pvol`.
`file`	string. A filepath to write the `pvol` object to.
`overwrite`	logical. Overwrites existing file when TRUE.
`infer_dtype`	logical. By default (infer_dtype = FALSE) writes 'params' back into ODIM HDF5 files with data stored in original data types. When TRUE infers data type from the R object data type, at the cost of (heavily) inflated file sizes.

Value

0 on success. A `pvol` object will be written to file in ODIM H5 format.

Examples

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

# load the file:
example_pvol <- read_pvolfile(pvolfile)
```

```
# write the file:
pvolfile_out <- paste0(tempdir(),"pvolfile_out.h5")
write_pvolfile(example_pvol, pvolfile_out)

# clean up
file.remove(pvolfile_out)
```

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

Description

Select parameters (`param`) or derived quantities by index from a plan position indicator (`ppi`).

Usage

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

Arguments

`x`	A `ppi` object.
`i`	Integer. Index/indices specifying which parameters (`param`) or derived quantities to extract.

Value

A `ppi` object containing a subset of parameters (`param`).

Examples

```
# Project a scan as a ppi
ppi <- project_as_ppi(example_scan)

# This ppi contains 5 parameters (DBZH VRADH ZDR RHOHV PHIDP)
ppi

# Subset ppi to one containing only the first parameter (DBZH)
ppi[1]

# Subset ppi to one containing the first three parameters (DBZH, VRADH, ZDR)
ppi[1:3]

# Subset ppi to one without the first 2 parameters (ZDR RHOHV PHIDP)
ppi[-1:-2]
```

Index