Package 'meteoland'

February 17, 2023

Type Package

Title Landscape Meteorology Tools

Version 2.0.0

Date 2023-02-10

Description Functions to estimate weather variables at any position of a landscape [De Caceres et al. (2018) <doi:10.1016/j.envsoft.2018.08.003>].

License GPL (>= 2)

URL https://emf-creaf.github.io/meteoland/index.html

LazyData TRUE

Depends R (>= 3.5.0), sp, sf, stars

Imports ncdf4, methods, stats, meteospain, Rcpp, units, lifecycle, cli, dplyr, tidyr, rlang, assertthat, purrr, ncdfgeom, ncmeta, lubridate, cubelyr, grDevices

Suggests knitr, forecast, spacetime, rmarkdown, testthat (>= 3.0.0), worldmet, rgdal, keyring, tibble

LinkingTo Rcpp

Encoding UTF-8

NeedsCompilation yes

VignetteBuilder knitr

RoxygenNote 7.2.3

Config/testthat/edition 3

Config/testthat/parallel true

Author Miquel De Cáceres [aut, cre] (<https://orcid.org/0000-0001-7132-2080>), Víctor Granda [aut] (<https://orcid.org/0000-0002-0469-1991>), Nicolas Martin [aut] (<https://orcid.org/0000-0001-7574-0108>), Antoine Cabon [aut] (<https://orcid.org/0000-0001-6426-1726>)

Maintainer Miquel De Cáceres <miquelcaceres@gmail.com>

Repository CRAN

Date/Publication 2023-02-17 22:20:02 UTC

R topics documented:

`add_topo` *Add topography data to meteo object*

Description

Add topography data to meteo object

Usage

```
add_topo(meteo, topo, verbose = getOption("meteoland_verbosity", TRUE))
```

Arguments

meteo	meteo object
topo	topo object
verbose	Logical indicating if the function must show messages and info. Default value checks "meteoland_verbosity" option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with options(meteoland_verbosity = FALSE)

Details

When using meteo data without topography info to create an interpolator, topography must be added

Value

meteo with the topography info added

See Also

Other interpolator functions: create_meteo_interpolator(), get_interpolation_params(), read_interpolator(), set_interpolation_params(), with_meteo(), write_interpolator()

Examples

```
# example meteo
data(meteoland_meteo_no_topo_example)
# example topo
data(meteoland_topo_example)
# add topo
with_meteo(meteoland_meteo_no_topo_example) |>
  add_topo(meteoland_topo_example)
```

---

averagearea *Averages area weather*

Description

[Deprecated]

Averages the weather data series of points or grid pixels.

Usage

```
averagearea(object, na.rm = TRUE)
```

Arguments

object	An object of class SpatialPointsMeteorology-class, SpatialGridMeteorology-class or SpatialPixelsMeteorology-class.
na.rm	Boolean flag to indicate that missing values should be excluded from averages.

Details

Assumes that all points/pixels represent the same area.

Value

An object of class as the input SpatialPointsMeteorology-class with weather series corresponding to the spatial average of the input.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

weathergeneration

Examples

```
data(examplegridtopography)
data(exampleinterpolationdata)

#Interpolation of meteorology over a grid for two days
ml = interpolationgrid(exampleinterpolationdata, examplegridtopography,
                       as.Date(c("2001-02-03", "2001-06-03")))

#Call averaging function
pa = averagearea(ml)

#Spatial information
pa

#Weather data
pa@data[[1]]
```

`complete_meteo` *Complete missing meteo variables*

Description

Calculates missing meteo variables

Usage

```
complete_meteo(meteo, verbose = getOption("meteoland_verbosity", TRUE))
```

Arguments

`meteo`	meteoland meteo data
`verbose`	Logical indicating if the function must show messages and info. Default value checks `"meteoland_verbosity"` option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with `options(meteoland_verbosity = FALSE)`

Details

This function takes a meteo object (with meteoland names) and complete any missing variable if it is possible

Value

the same `meteo` data provided with the the variables completed

Examples

```
# example data
data("meteoland_meteo_example")

# remove MinRelativeHumidity
meteoland_meteo_example$MinRelativeHumidity <- NULL
# complete vars
completed_meteo <- complete_meteo(meteoland_meteo_example)
# check MinRelativeHumidity
completed_meteo$MinRelativeHumidity
```

`correctionpoint` *Statistical correction of meteorological variables for a set of points*

Description

[Deprecated]

Functions `correctionpoint` and `correctionpoints` perform correction of predicted climatic data by applying statistical correction methods (unbiasing, scaling, or quantile mapping) to meteorological variables. Function `correctionpoints.errors` allows evaluating, for each point, the bias and mean absolute error (MAE) obtained before and after correcting the climate model for the historical period.

Usage

```
correctionpoint(
  obs,
  mod,
  proj,
  dates = NULL,
  params = defaultCorrectionParams(),
  verbose = TRUE
)

correctionpoints(
  object,
  points,
  topodata = NULL,
  dates = NULL,
  export = FALSE,
  exportDir = getwd(),
  exportFile = NULL,
  exportFormat = "meteoland/txt",
  metadataFile = "MP.txt",
  corrOut = FALSE,
  verbose = TRUE
)

correctionpoints.errors(
  object,
  points,
  topodata = NULL,
  error.type = "residuals.cv",
  keep.data = FALSE,
  verbose = FALSE
)
```

Arguments

obs	A data frame with observed meteorology.
mod, proj	Data frame with predicted meteorology for the reference and projection periods, respectively.
dates	An object of class Date with a subset of dates of the projection period to be corrected. If dates = NULL then all dates in proj or the projection data of object are processed.
params	A list with correction params (see defaultCorrectionParams).
verbose	Boolean flag to print process information.
object	An object of class MeteorologyUncorrectedData-class containing the meteorology of more than one point.
points	An object of class SpatialPointsMeteorology-class with the coordinates and historical meteorological data of the locations for which correction of predicted climatic data has to be done. Alternatively, an object of class SpatialPointsDataFrame-class containing the meta data (columns dir, filename and possibly format) of meteorological files that will be read from the disk.
topodata	A data frame with topographic data for each point (i.e. three columns named elevation, slope and aspect). If topodata = NULL then Penman's potential evapotranspiration is not calculated.
export	If export = FALSE the result of correction is stored in memory. Otherwise the result is written in the disk (using the format specified in exportFormat).
exportDir	Output directory for corrected meteorology data (txt/rds format).
exportFile	Output file for corrected meteorology data (netCDF format).
exportFormat	Export format for meteorological data (see writemeteorologypoint). If format is "meteoland/txt", "meteoland/rds", "castanea/txt" or "castanea/rds" the function tries to write one file per point in exportDir. If format is "netCDF" the function will write data to a single file specified by exportFile.
metadataFile	The name of the file that will store the meta data describing all written files.
corrOut	Boolean flag to indicate that correction parameters (i.e. calculated biases) should be included with the output. Setting corrOut = TRUE changes the returned value.
error.type	String to specify the error to be evaluated, either "before" (before correction), "residual" (after correction) or "residual.cv" (after correction, but using cross-validation).
keep.data	Boolean flag to return the uncorrected/corrected data for the historical period.

Details

Function correctionpoints performs statistical correction of predicted climatic data for all points supplied in points whereas correctionpoint performs statistical correction of one single point. Observed meteorological data for each point typically comes from a nearby meteorological station, but they can be the result of interpolating the meteorology of several stations (see MeteorologyInterpolationData) or they can be extracted from reanalyzed meteorology (e.g. EU-WATCH) (see extractNetCDF).

For each target point, correctionpoints function first determines the predicted cell where the point falls according to the euclidean distance in the geographic space of object. Then it calls

correctionpoint. In turn, correctionpoint determines the dates that are shared in observed and predicted data for the historical period. These meteorological data of dates are used to conduct the correction of predicted climatic data for the future period. Corrections biases are calculated and applied for the twelve months separately. The user can control the methods used for correction of each meteorological variable by changing the slot params in object (see class MeteorologyUncorrectedData-class) or the parameter params to correctionpoint. Three options are allowed (see defaultCorrectionParams): (a) 'unbias' for shifting the mean; (b) 'scaling' for multiplication by a factor; and (c) 'quantmap' for empirical quantile mapping between observed and modelled data (Déqué 2007).

A difficulty arises for quantile mapping when the variables bounded by zero, such as precipitation. As the models tend to drizzle (or may have lower frequency of precipitation events), the probability of precipitation in the model may be greater or lower than that observed. To correct this, when model precipitation is zero an observed value is randomly chosen in the interval where the observed cumulative frequency is less than or equal to the probability of no precipitation in the model. This procedure ensures that the probability of precipitation after correction is equal to that observed (Boé 2007).

Value

- Function correctionpoint returns a data frame.
- If export = FALSE, the function correctionpoints returns an object of class SpatialPointsMeteorology-class with the bias-corrected meteorology for each point. If export=TRUE then bias-corrected data is written into the disk. For txt/rds export formats, the function returns an object of class SpatialPointsDataFrame-class containing the meta data of the files written in the disk. For netCDF export format the function returns NULL. If corrOut = TRUE the function returns a list which contains any previous output and an object with the calculated correction factors (biases, mappings) for each point and month.
- Function correctionpoints.errors (keep.data = FALSE) returns a data frame with the mean absolute error (MAE) and bias for each variable and point. If keep.data = TRUE then the function also returns a list of data frames with the uncorrected/corrected series used in the comparisons with observations.

Functions

- correctionpoint(): **[Deprecated]**
- correctionpoints.errors(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

Nicolas Martin, INRA-Avignon

References

Boé J, Terray L, Habets F, Martin E (2007) Statistical and dynamical downscaling of the Seine basin climate for hydro-meteorological studies. Int J Climatol 27:1643–1655. doi: 10.1002/joc.1602

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

Déqué M (2007) Frequency of precipitation and temperature extremes over France in an anthropogenic scenario: Model results and statistical correction according to observed values. Glob Planet Change 57:16–26. doi: 10.1016/j.gloplacha.2006.11.030

See Also

penman, SpatialPointsMeteorology-class, writemeteorologypointfiles, MeteorologyUncorrectedData, MeteorologyInterpolationData

Examples

```
data(examplegridtopography)
data(exampleinterpolationdata)
data(examplecorrectiondata)

#Creates spatial topography points from the grid
p = 1:2
spt = as(examplegridtopography, "SpatialPointsTopography")[p]


#Interpolation of two points for the whole time period (2000-2003)
historical = interpolationpoints(exampleinterpolationdata, spt)

#Downscaling of future predictions (RCM models, year 2023)
predicted = correctionpoints(examplecorrectiondata, historical, spt@data)

#Plot predicted mean temperature for point 1
meteoplot(predicted, 1, "MeanTemperature", ylab="Temperature (Celsius)", ylim=c(-5,40))
meteoplot(predicted, 1, "MinTemperature", add=TRUE, col="blue")
meteoplot(predicted, 1, "MaxTemperature", add=TRUE, col="red")
#Add uncorrected mean temperature data (cell #3)
lines(examplecorrectiondata@dates,
      examplecorrectiondata@projection_data[[3]]$MeanTemperature,
      lty=3)
lines(examplecorrectiondata@dates,
      examplecorrectiondata@projection_data[[3]]$MinTemperature,
      col="blue", lty=3)
lines(examplecorrectiondata@dates,
      examplecorrectiondata@projection_data[[3]]$MaxTemperature,
      col="red", lty=3)
legend("topright", legend=c("corrected","uncorrected", "Maximum", "Mean", "Minimum"),
       col=c("black","black", "red","black","blue"), lty=c(1,3,1,1,1), bty="n")

#Scatter plot
plot(examplecorrectiondata@projection_data[[3]]$MeanTemperature,
     predicted@data[[1]]$MeanTemperature, cex=0.1, asp=1,
     ylab="Corrected mean temperature", xlab="Uncorrected mean temperature")
```

```
abline(a=0,b=1,col="gray")

#Plot predicted precipitation for point 1
meteoplot(predicted, 1, "Precipitation", ylab="Precipitation (mm)", ylim=c(0,120))
#Add uncorrected mean temperature data (cell #3)
lines(examplecorrectiondata@dates,
      examplecorrectiondata@projection_data[[3]]$Precipitation,
      col="red", lty=3)
legend("topleft", legend=c("corrected","uncorrected"), col=c("black","red"), lty=c(1,3), bty="n")

#Scatter plot
plot(examplecorrectiondata@projection_data[[3]]$Precipitation,
     predicted@data[[1]]$Precipitation, cex=0.1, asp=1,
     ylab="Corrected precipitation (mm)", xlab="Uncorrected precipitation (mm)")
abline(a=0,b=1,col="gray")
```

`correction_series` *Low-level correction functions*

Description

[Deprecated]

Low-level function to perform bias correction.

Usage

```
correction_series(
  obs,
  mod,
  proj = NULL,
  method = "unbias",
  isPrec = TRUE,
  qstep = 0.01
)
```

Arguments

`obs`	Observed series for the reference (historical) period.
`mod`	Modelled series for the reference (historical) period.
`proj`	Modelled series for the projected period. If missing, the reference (historical) period is corrected.
`method`	Correction method, either `"unbias"`, `"scaling"`, `"quantmap"`
`isPrec`	A flag to indicate that variable is precipitation (only relevant for quantile mapping).
`qstep`	Probability step for quantile mapping (see `defaultCorrectionParams`).

Value

Returns a vector with corrected values.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

See Also

correctionpoints, defaultCorrectionParams

---

create_meteo_interpolator

Meteoland interpolator creation

Description

Function to create the meteoland interpolator

Usage

```
create_meteo_interpolator(
  meteo_with_topo,
  params = NULL,
  verbose = getOption("meteoland_verbosity", TRUE)
)
```

Arguments

meteo_with_topo	Meteo object, as returned by with_meteo
params	Interpolation parameters as a list. Typically the result of defaultInterpolationParams.
verbose	Logical indicating if the function must show messages and info. Default value checks "meteoland_verbosity" option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with options(meteoland_verbosity = FALSE)

Details

This function takes meteorology information and a list of interpolation parameters and creates the interpolator object to be ready to use.

Value

an interpolator object (stars)

See Also

Other interpolator functions: `add_topo()`, `get_interpolation_params()`, `read_interpolator()`, `set_interpolation_params()`, `with_meteo()`, `write_interpolator()`

Examples

```
# example meteo data
data(meteoland_meteo_example)

# create the interpolator with default params
with_meteo(meteoland_meteo_example) |>
  create_meteo_interpolator()

# create the interpolator with some params changed
with_meteo(meteoland_meteo_example) |>
  create_meteo_interpolator(params = list(debug = TRUE))
```

---

defaultCorrectionParams

Default correction parameters

Description

[Deprecated]

Returns a list with the default parameterization for statistical correction.

Usage

```
defaultCorrectionParams()
```

Value

A list with the following items (default values in brackets):

- `methods`: A list with the correction method for each variable (options are "unbias" (shift of the mean), "scaling" (factor multiplication), "quantmap" (empirical quantile mapping) or "none" (for no correction)). Defaults are:
  - `MeanTemperature` = `"unbias"`
  - `MinTemperature` = `"quantmap"`
  - `MaxTemperature` = `"quantmap"`
  - `Precipitation` = `"quantmap"`

  - MeanRelativeHumidity = "unbias"
  - Radiation = "unbias"
  - WindSpeed = "quantmap"
- fill_wind [= TRUE]: A logical flag to fill wind speed values with uncorrected values when reference data is missing.
- allow_saturated [= FALSE]: A logical flag to indicate whether relative humidity values above saturation (>100%) are permitted (bias correction is performed on specific humidity).
- wind_height [= 10]: Wind measurement height (in m).
- qstep [= 0.01]: a numeric value between 0 and 1. Quantile mapping is fitted only for the quantiles defined by quantile(0,1,probs=seq(0,1,by=qstep)).

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

See Also

MeteorologyInterpolationData, MeteorologyUncorrectedData

---

defaultGenerationParams

Default generation parameters

Description

[Deprecated]

Returns a list with the default parameterization for weather generation.

Usage

```
defaultGenerationParams()
```

Value

A list with the following items (default values in brackets):

- conditional [= "none"]: A string to indicate whether multi-year weather should be conditioned or not. If conditional = "arima", annual precipitation is conditioned on a stationary auto-regressive model. If conditional = "window", a moving-window is used to subset the years used to parametrize the weather generation algorithm for each target year (see parameter range_size_years). In this last case, annual precipitation is conditioned to a log-normal variate with parameters fitted from the selected years.

- `dry_wet_threshold` [= `0.3`]: Precipitation threshold (mm) for separating dry from wet days.
- `wet_extreme_quantile_threshold` [= `0.8`]: Quantile for separating wet from extremely wet days.
- `range_size_days` [= `5`]: Minimum half range size to select the subset of dates with DOY similar to the currently simulated.
- `range_size_years` [= `12`]: Half range size to select the subset of years in a moving-window around the current year (if `conditional = "window"`).
- `n_knn_annual` [= `100`]: Number of years to be re-sampled using K-nearest neighbour for annual precipitation (if `conditional = "arima"` or `conditional = "window"`).
- `adjust_annual_precip` [= `TRUE`]: A logical flag to indicate that annual precipitation generated by the algorithm should be adjusted to fit either overall input annual precipitation or simulated annual precipitation.
- `min_ratio` [= `0.9`]: Minimum adjustment ratio for precipitation.
- `max_ratio` [= `1.2`]: Minimum adjustment ratio for precipitation.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Apipattanavis, S., G. Podesta, B. Rajagopalan, and R. W. Katz (2007), A semiparametric multivariate and multisite weather generator, Water Resour. Res., 43, W11401, doi:10.1029/2006WR005714.

Steinschneider S. & Brown C. (2013) A semiparametric multivariate, multisite weather generator with low-frequency variability for use in climate risk assessments. Water Resour. Res. 49, 7205-7220. doi:10.1002/wrcr.20528.

See Also

`weathergeneration`

`defaultInterpolationParams` *Default interpolation parameters*

Description

Returns a list with the default parameterization for interpolation. Most parameter values are set according to Thornton et al. (1997).

Usage

```
defaultInterpolationParams()
```

Value

A list with the following items (default values in brackets):

- `initial_Rp` [= `140000`]: Initial truncation radius.
- `iterations` [= `3`]: Number of station density iterations.
- `alpha_MinTemperature` [= `3.0`]: Gaussian shape parameter for minimum temperature.
- `alpha_MaxTemperature` [= `3.0`]: Gaussian shape parameter for maximum temperature.
- `alpha_DewTemperature` [= `3.0`]: Gaussian shape parameter for dew-point temperature.
- `alpha_PrecipitationEvent` [= `5.0`]: Gaussian shape parameter for precipitation events.
- `alpha_PrecipitationAmount` [= `5.0`]: Gaussian shape parameter for the regression of precipitation amounts.
- `alpha_Wind` [= `3.0`]: Gaussian shape parameter for wind.
- `N_MinTemperature` [= `30`]: Average number of stations with non-zero weights for minimum temperature.
- `N_MaxTemperature` [= `30`]: Average number of stations with non-zero weights for maximum temperature.
- `N_DewTemperature` [= `30`]: Average number of stations with non-zero weights for dew-point temperature.
- `N_PrecipitationEvent` [= `5`]: Average number of stations with non-zero weights for precipitation events.
- `N_PrecipitationAmount` [= `20`]: Average number of stations with non-zero weights for the regression of precipitation amounts.
- `N_Wind` [= `2`]: Average number of stations with non-zero weights for wind.
- `St_Precipitation` [= `5`]: Number of days for the temporal smoothing of precipitation.
- `St_TemperatureRange` [= `15`]: Number of days for the temporal smoothing of temperature range.
- `pop_crit` [= `0.50`]: Critical precipitation occurrence parameter.
- `f_max` [= `0.6`]: Maximum value for precipitation regression extrapolations (0.6 equals to a maximum of 4 times extrapolation).
- `wind_height` [= `10`]: Wind measurement height (in m).
- `debug` [= `FALSE`]: Boolean flag to show extra console output.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Thornton, P.E., Running, S.W., White, M. A., 1997. Generating surfaces of daily meteorological variables over large regions of complex terrain. J. Hydrol. 190, 214–251. doi:10.1016/S0022-1694(96)03128-9.

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

See Also

MeteorologyInterpolationData

`downloadAEMETcurrentday` *Download data from AEMET*

Description

[Deprecated]

Download data from the Spanish National Meterology Agency (AEMET)

Usage

```
downloadAEMETcurrentday(api, daily = TRUE, verbose = TRUE)

downloadAEMEThistorical(
  api,
  dates,
  station_id = NULL,
  export = FALSE,
  exportDir = getwd(),
  exportFormat = "meteoland/txt",
  metadataFile = "MP.txt",
  verbose = TRUE
)

downloadAEMETstationlist(api)
```

Arguments

`api`	String with the AEMET API key (see https://opendata.aemet.es/).
`daily`	Boolean flag. Are data to be returned at a daily or hourly scale?
`verbose`	Boolean flag to print process information.
`dates`	An object of class `Date`.
`station_id`	A string vector containing station ids (the list of stations for which historical climatic series are available is given by `downloadAEMETstationlist`).
`export`	If `export = FALSE` the downloaded data is stored in memory. Otherwise the result is written on the disk (using the format specified in `exportFormat`).
`exportDir`	Output directory for downloaded meteorology.
`exportFormat`	Format of meteorological data. Current accepted formats are `"castanea"` and `"meteoland"`.
`metadataFile`	The name of the file that will store the meta data describing all written files.

Details

API key needs to be acquired from AEMET (https://opendata.aemet.es/)

Value

Function `downloadAEMETstationlist` returns a `SpatialPointsDataFrame-class` object containing the list of AEMET weather stations for which historical climatic series are available and can be retrieved using `downloadAEMEThistorical`.

Function `downloadAEMEThistorical` downloads data for the specified AEMET stations and dates. Data is available for dates up to 4 days before current date. If `export = FALSE`, function `downloadAEMEThistorical` returns a `SpatialPointsMeteorology-class` object with the downloaded meteorology for each station (point). Otherwise the function writes on the disk at the location specified by `exportDir` and solely returns a `SpatialPointsDataFrame-class` object containing the files metadata.

Function `downloadAEMETcurrentday` downloads recent weather (the last 24h) from all currently available stations and returns data frame if `daily = FALSE` or a `SpatialPointsDataFrame-class` object with observations aggregated at the daily scale if else.

Functions

- `downloadAEMEThistorical()`: **[Deprecated]**
- `downloadAEMETstationlist()`: **[Deprecated]**

Note

Since ver. 1.0.1, weather data download functions included in `meteoland` make internal calls to functions in package `meteospain`. For an enhanced flexibility, users are recommended to call functions in `meteospain` themselves, and then to use function `reshapemeteospain` to generate data suitable for `meteoland`.

The list of stations available in `downloadAEMETcurrentday` (current observations) may be different from the list given by `downloadAEMETstationlist` and available in `downloadAEMEThistorical` (stations with historical climate series).

Author(s)

Antoine Cabon, CTFC

Miquel De Cáceres Ainsa, CREAF

References

AEMET should be acknowledged as author of information when using this data.

See Also

`SpatialPointsMeteorology-class`

downloadMETEOCLIMATICcurrentday *Download data from Meteoclimatic network*

Description

[Deprecated]

Download data from the Spanish Automatic Stations Network (non-professional)

Usage

```
downloadMETEOCLIMATICcurrentday(station_id = "ESCAT")

downloadMETEOCLIMATICstationlist(station_id = "ESCAT")
```

Arguments

station_id	A string vector containing station ids (the list of stations for which current day climatic data is available is given by downloadMETEOCLIMATICstationlist). By default, returns the list of stations in Catalonia.

Details

Meteoclimatic is a non-professional automatic stations network, maintained by volunteers that share the data from their climatic stations. Data offered by these stations has not passed any quality control.

Value

Function downloadMETEOCLIMATICstationlist returns a SpatialPointsDataFrame-class object containing the list of Meteoclimatic weather stations for which data is available based on the station_id codes provided.

Function downloadMETEOCLIMATICcurrentday downloads recent weather (for the current day) from all currently available stations and returns a SpatialPointsDataFrame-class object with observations. Only accumulated precipitation, maximum and minimum temperature and relative humidity are returned.

Functions

- downloadMETEOCLIMATICstationlist(): **[Deprecated]**

Note

Since ver. 1.0.1, weather data download functions included in meteoland make internal calls to functions in package meteospain. For an enhanced flexibility, users are recommended to call functions in meteospain themselves, and then to use function reshapemeteospain to generate data suitable for meteoland.

Author(s)

Víctor Granda, EMF-CREAF

Miquel De Cáceres Ainsa, EMF-CREAF

References

Meteoclimatic should be acknowledged as author of information when using this data.

See Also

SpatialPointsMeteorology-class

`downloadMGcurrentday` *Download data from MeteoGalicia*

Description

[Deprecated]

Download data from the Galician Meterology Agency (MeteoGalicia)

Usage

```
downloadMGcurrentday(station_id = NULL, daily = TRUE, verbose = TRUE)

downloadMGhistorical(date_from, date_to, station_id = NULL, verbose = TRUE)

downloadMGstationlist()
```

Arguments

`station_id`	A string vector containing station ids (the list of stations presently operative is given by `downloadMGstationlist`). If NULL all stations with available data are returned.
`daily`	Boolean flag. Are data to be returned at a daily or hourly scale?
`verbose`	Boolean flag to print process information.
`date_from, date_to`	Strings or objects of class `Date` specifying first and last date of the desired period.

Details

See available data services of MeteoGalicia at https://www.meteogalicia.gal/web/RSS/rssIndex.action?request_locale=es.

Value

Function downloadMGstationlist returns a SpatialPointsDataFrame-class object containing the list of MeteoGalicia weather stations currently operative. Function downloadMGhistorical downloads data for the specified MG weather stations (or all) and dates and returns a SpatialPointsMeteorology-class object with the downloaded meteorology for each station (point).

Function downloadMGcurrentday downloads recent weather data (the last 24h) from all currently available stations and returns data frame if daily = FALSE or a SpatialPointsDataFrame-class object with observations aggregated at the daily scale otherwise.

Functions

- downloadMGhistorical(): **[Deprecated]**
- downloadMGstationlist(): **[Deprecated]**

Note

Since ver. 1.0.1, weather data download functions included in meteoland make internal calls to functions in package meteospain. For an enhanced flexibility, users are recommended to call functions in meteospain themselves, and then to use function reshapemeteospain to generate data suitable for meteoland.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

MeteoGalicia (from the Conselleria de Medio Ambiente, Territorio e Vivenda of Xunta de Galicia) should be acknowledged as source of information when using this data.

See Also

SpatialPointsMeteorology-class

---

downloadSMCcurrentday *Download data from SMC*

Description

[Deprecated]

Download data from the Catalan automatic weather station network (XEMA from Servei Meteorològic de Catalunya)

Usage

```
downloadSMCcurrentday(
  api,
  daily = TRUE,
  station_id = NULL,
  date = Sys.Date(),
  verbose = TRUE
)

downloadSMChistorical(
  api,
  date,
  station_id = NULL,
  export = FALSE,
  exportDir = getwd(),
  exportFormat = "meteoland/txt",
  metadataFile = "MP.txt",
  verbose = TRUE
)

downloadSMCstationlist(api, date = NULL)
```

Arguments

api	String with the SMC API key (the procedure to apply for an api key is explained in https://apidocs.meteocat.gencat.cat/).
daily	Boolean flag. Are data to be returned at a daily or hourly scale?
station_id	A string vector containing station ids (the list of stations for which climatic series are available is given by `downloadSMCstationlist()`). If `NULL`, all available stations are queried. Otherwise, only the data corresponding to the specified stations will be returned.
date	An object of class `Date`. By default the current day in the case of `downloadSMCcurrentday()`. In the case of `downloadSMCstationlist()` a date for which operational stations are queried. In the case of `downloadSMChistorical`, the function returns the whole month the first date selected is in (implementation in package `meteospain`).
verbose	Boolean flag to print process information.
export	If `export = FALSE` the downloaded data is stored in memory. Otherwise the result is written on the disk (using the format specified in `exportFormat`).
exportDir	Output directory for downloaded meteorology.
exportFormat	Format of meteorological data. Current accepted formats are `"castanea"` and `"meteoland"`.
metadataFile	The name of the file that will store the meta data describing all written files.

Details

API key needs to be requested from SMC (https://apidocs.meteocat.gencat.cat/).

Value

Function downloadSMCstationlist returns a SpatialPointsDataFrame-class object containing the list of SMC operational weather stations for the date given.

Function downloadSMCcurrentday downloads recent weather (the last 24h or the weather for a given date) from all currently available stations and returns data frame if daily_meteoland = FALSE or a SpatialPointsDataFrame-class object with observations aggregated at the daily scale otherwise.

Function downloadSMChistorical downloads historical daily weather corresponding to a given time period from a set (or all currently available) stations. Results are returned (or exported) after formatting data as a SpatialPointsMeteorology-class if variable_code = NULL, or as a data frame otherwise.

Functions

- downloadSMChistorical(): **[Deprecated]**
- downloadSMCstationlist(): **[Deprecated]**

Note

Since ver. 1.0.1, weather data download functions included in meteoland make internal calls to functions in package meteospain. For an enhanced flexibility, users are recommended to call functions in meteospain themselves, and then to use function reshapemeteospain to generate data suitable for meteoland.

Author(s)

Antoine Cabon, CTFC

Miquel De Cáceres Ainsa, CREAF

References

Servei Meteorològic de Catalunya (SMC) should be acknowledged as author of information when accessing weather data with these functions.

See Also

SpatialPointsMeteorology-class

examplecorrectiondata *Example data set for statistical correction of RCM predictions*

Description

[Deprecated]

Example data set including the predictions of Regional Climate Model (CCLM4-8-17; driving global model CNRM-CERFACS-CNRM-CM5) for 3 model cells in a small area in Catalonia (NE Spain). Meteorological data covers an historical (reference) period (2000-2003) and a future (projection) period (2020-2023), the latter simulated under rcp4.5 scenario.

Format

Formal class 'MeteorologyUncorrectedData-class'

Source

ESFG web site (http://esgf.llnl.gov/) that centralizes climate data from GCM and RCM uploaded in the frame of different international consortium, including the EURO-CORDEX regionalisation project.

Examples

```
data(examplecorrectiondata)
```

examplegridtopography *Example spatial grid topography*

Description

[Deprecated]

'SpatialGridTopography' object describing topographic features for a grid of 5 km x 5 km and cell size of 100 m in Catalonia (NE Spain).

Format

Formal class 'SpatialGridTopography'

Source

'Institut Cartogràfic de Catalunya' (ICC)

Examples

```
data(examplegridtopography)
```

`exampleinterpolationdata` *Example data set for interpolation from weather stations*

Description

[Deprecated]

Example data set of spatial location, topography and daily meteorological records from 38 weather stations in Catalonia (NE Spain) corresponding to years 2000-2003.

Format

Formal class 'MeteorologyInterpolationData-class'

Source

'Servei Meteorològic de Catalunya' (SMC) and 'Agencia Española de Meteorología' (AEMET)

Examples

```
data(exampleinterpolationdata)
```

`extractNetCDF` *Extraction of climatic data from NetCDF files (deprecated)*

Description

[Deprecated]

This function reads a set of NetCDF files (one per variable) and extracts data for a set of NetCDF cells that are specified using a boundary box (in lon/lat format) or a set of (x,y) grid indices.

Usage

```
extractNetCDF(
  ncdf_files,
  bbox = NULL,
  offset = 0,
  cells = NULL,
  export = TRUE,
  exportDir = getwd(),
  exportFormat = "meteoland/txt",
  mpfilename = "MP.txt"
)
```

Arguments

ncdf_files	Character vector containing files to read
bbox	Boundary box (2 x 2 matrix) specifying the limit coordinates of a study area (in lon/lat format).
offset	A buffer to include NetCDF cells that are at a certain distance around the boundary box.
cells	A (n x 2) matrix specifying the x and y indices of n cells in a grid.
export	If export = FALSE the extracted data is stored in memory. Otherwise the result is written in the disk (using the format specified in exportFormat).
exportDir	Output directory for extracted meteorology.
exportFormat	Export format for meteorological data (see writemeteorologypoint).
mpfilename	The name of the file that will store the meta data describing all written files.

Details

Function extractNetCDF first identifies which cells in NetCDF data should be extracted according to bbox (or the cells are indicated by the user using cells), and the overall period (days). If neither bbox or cells is supplied, then all NetCDF cells will be processed. For each cell to be processed, the function loops over all files (which can describe different variables and time periods) and extracts the corresponding data. The function transforms units to the units used in meteoland. If specific humidity and mean temperature are available, the function calculates mean relative humidity.

Extracted meteorological data (a data frame with days in rows and meteorological variables in columns) can be stored in an object SpatialPointsMeteorology-class or it can be written in the disk (one file per cell). In the latter case, the output format can be chosen and the function also writes a supplementary file containing the meta data (i.e. the coordinates and filename of each file).

Humidity in climate model files is given as specific humidity. This is converted to relative humidity and the conversion may produce values above saturation (>100%) (see also defaultCorrectionParams for the same issue when performing bias correction).

Value

If `export = FALSE`, the function returns an object of class `SpatialPointsMeteorology-class` with the meteorological series for each cell (represented by a spatial point). Otherwise the function returns an object of class `SpatialPointsDataFrame-class` containing the meta data of the files written in the disk.

Author(s)

Miquel De Cáceres Ainsa, CREAF

Nicolas Martin, INRA-Avignon

See Also

`correctionpoints`, `writemeteorologypointfiles`, `SpatialPointsMeteorology-class`

`extractvars` *Extracts meteorological data*

Description

[Deprecated]

Extracts meteorological data from an object.

Usage

```
extractvars(object, vars, verbose = FALSE)

extractdates(object, dates = NULL, verbose = FALSE)

extractgridindex(grid, index)

extractgridpoints(grid, points, verbose = FALSE)
```

Arguments

`object`	An object of class `SpatialPointsMeteorology`, `SpatialGridMeteorology` or `SpatialPixelsMeteorology`.
`vars`	A character vector with the set of variables to be extracted.
`verbose`	Boolean flag to print process information.
`dates`	A vector of `Date` with a (subset) of dates to be extracted. If `NULL` all dates will be returned.
`grid`	An object of class `SpatialGridMeteorology-class` or `SpatialPixelsMeteorology-class` with the meteorological data for a full grid or a subset of grid cells, respectively. Alternatively, a string specifying a NetCDF to be read from the disk.
`index`	An integer with a grid index.
`points`	An object of class `SpatialPoints`.

Details

Function `extractpoints` is deprecated, because its functionality can be achieved using subsetting of spatial classes `SpatialGridMeteorology` and `SpatialPixelsMeteorology`.

Value

- Function `extractdates()`, returns a `list` with the same length as `dates`. Each element of the list is a spatial object (`SpatialPointsDataFrame`, `SpatialGridDataFrame` or `SpatialPixelsDataFrame`, depending on the input) with the meteorological data for all the spatial elements. If only one date is asked, the function returns directly the spatial object, without embedding it into a list.
- Function `extractvars()`, returns a `list` with the same length as `vars`. Each element of the list is a spatial object (`SpatialPointsDataFrame`, `SpatialGridDataFrame` or `SpatialPixelsDataFrame`, depending on the input) with the meteorological data for all the spatial elements. If only one variable is asked, the function returns directly the spatial object, without embedding it into a list.
- Function `extractgridindex()` returns a data frame.
- Function `extractgridpoints()` returns an object of class `SpatialPointsMeteorology`.

Functions

- `extractvars()`: **[Deprecated]**
- `extractgridindex()`: **[Deprecated]**
- `extractgridpoints()`: **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

`get_interpolation_params` *Retrieving interpolation parameters from interpolator object*

Description

Retrieve the parameter list from and interpolator object

Usage

```
get_interpolation_params(interpolator)
```

Arguments

`interpolator`	interpolator object as returned by `create_meteo_interpolator`

Value

The complete parameter list from the interpolator object

See Also

Other interpolator functions: `add_topo()`, `create_meteo_interpolator()`, `read_interpolator()`, `set_interpolation_params()`, `with_meteo()`, `write_interpolator()`

Examples

```
# example interpolator
data(meteoland_interpolator_example)
# get the params from the interpolator
get_interpolation_params(meteoland_interpolator_example)
```

---

humidity_relative2dewtemperature

Humidity conversion tools

Description

Functions to transform relative humidity to specific humidity or dew point temperature and viceversa.

Usage

```
humidity_relative2dewtemperature(Tc, HR)

humidity_dewtemperature2relative(Tc, Td, allowSaturated = FALSE)

humidity_specific2relative(Tc, HS, allowSaturated = FALSE)

humidity_relative2specific(Tc, HR)
```

Arguments

`Tc`	A numeric vector of temperature in degrees Celsius.
`HR`	A numeric vector of relative Humidity (in %).
`Td`	A numeric vector of dew temperature in degrees Celsius.
`allowSaturated`	Logical flag to allow values over 100%
`HS`	A numeric vector of specific humidity (unitless).

Value

A numeric vector with specific or relative humidity.

Author(s)

Nicholas Martin-StPaul, INRA

Miquel De Cáceres Ainsa, CREAF

See Also

meteocomplete

---

interpolate_data *Interpolation process for spatial data*

Description

Interpolate spatial data to obtain downscaled meteorologic variables

Usage

```
interpolate_data(
  spatial_data,
  interpolator,
  dates = NULL,
  variables = NULL,
  verbose = getOption("meteoland_verbosity", TRUE)
)
```

Arguments

spatial_data	An sf or stars raster object to interpolate
interpolator	A meteoland interpolator object, as created by create_meteo_interpolator
dates	vector with dates to interpolate (must be within the interpolator date range). Default to NULL (all dates present in the interpolator object)
variables	vector with variable names to be interpolated. NULL (default), will interpolate all variables. Accepted names are "Temperature", "Precipitation", "RelativeHumidity", "Radiation" and "Wind"
verbose	Logical indicating if the function must show messages and info. Default value checks "meteoland_verbosity" option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with options(meteoland_verbosity = FALSE)

Details

This function takes a spatial data object (sf or stars raster), an interpolator object (create_meteo_interpolator) and a vector of dates to perform the interpolation of the meteorologic variables for the spatial locations present in the spatial_data object.

Value

an object with the same class and structure as the provided spatial data with the results of the interpolation joined. In the case of spatial data being an sf, the results are added as a list-type column that can be unnested with unnest. In the case of a stars raster object, interpolation results are added as attributes (variables)

Spatial data

The spatial data provided must be of two types. (I) A sf object containing POINT for each location to interpolate or (II) a stars raster object for which the interpolation should be done. Independently of the class of `spatial_data` it has to have some mandatory variables, namely `elevation`. It should also contain `aspect` and `slope` for a better interpolation process, though this two variables are not mandatory.

Examples

```
# example of data to interpolate and example interpolator
data("points_to_interpolate_example")
data("meteoland_interpolator_example")

# interpolate data
res <- interpolate_data(points_to_interpolate_example, meteoland_interpolator_example)

# check result
# same class as input data
class(res)
# data
res
# results for the first location
res[["interpolated_data"]][1]
# unnest results
tidyr::unnest(res, cols = "interpolated_data")
```

`interpolation.calibration` *Calibration and validation of interpolation procedures*

Description

[Deprecated]

Function `interpolation.calibration` determines optimal interpolation parameters 'N' and 'alpha' for a given meteorological variable. Optimization is done by minimizing mean absolute error (MAE) (Thornton et al. 1997). Function `interpolation.cv` calculates average mean absolute errors (MAE) for the prediction period of an object of class '`MeteorologyInterpolationData`'. Function `summary` returns a data.frame with cross-validation summaries and `plot` plots cross-validation results. In both calibration and validation procedures, predictions for each weather station are made using a leave-one-out procedure (i.e. after excluding the station from the predictive set).

Usage

```
interpolation.calibration(
  object,
  stations = NULL,
  variable = "Tmin",
  N_seq = seq(5, 30, by = 5),
  alpha_seq = seq(0.25, 10, by = 0.25),
  verbose = FALSE
)

interpolation.calibration.fmax(
  object,
  stations = NULL,
  fmax_seq = seq(0.05, 0.95, by = 0.05),
  verbose = FALSE
)

interpolation.cv(object, stations = NULL, verbose = FALSE)

## S3 method for class 'interpolation.cv'
plot(x, type = "stations", ...)

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

Arguments

object	In the case of function interpolation.cv, an object of class MeteorologyInterpolationData-class. In the case of function summary, an object of class interpolation.cv
stations	A numeric vector containing the indices of stations to be used to calculate mean absolute errors (MAE) in the calibration or cross-validation analysis. All the stations with data are included in the training set but predictive MAE are calculated for the 'stations' subset only.
variable	A string indicating the meteorological variable for which interpolation parameters 'N' and 'alpha' will be calibrated. Accepted values are 'Tmin' (for minimum temperature), 'Tmax' (for maximum temperature), 'Tdew' (for dew-point temperature), 'PrecEvent' (for precipitation events),'PrecAmount' (for regression of precipitation amounts),'Prec' (for precipitation with the same values for precipitation events and regression of precipitation amounts).
N_seq	Set of average number of points to be tested.
alpha_seq	Set of alpha values to be tested.
verbose	A logical flag to generate additional console output.
fmax_seq	Set of f_max values to be tested.
x	A S3 object of class interpolation.cv with cross-validation results.
type	A string of the plot type to be produced (either "stations" or "dates").
...	Additional parameters passed to summary and plot functions.

Value

Function `interpolation.calibration` returns an object of class `'interpolation.calibration'` with the following items:

- `MAE`: A numeric matrix with the mean absolute error values (averaged across stations) for each combination of parameters 'N' and 'alpha'.
- `minMAE`: Minimum MAE value.
- `N`: Value of parameter 'N' corresponding to the minimum MAE.
- `alpha`: Value of parameter 'alpha' corresponding to the minimum MAE.
- `Observed`: A matrix with observed values.
- `Predicted`: A matrix with predicted values for the optimum parameter combination.

Function `interpolation.cv` returns a list of class `'interpolation.cv'` with the following items:

- `stations`: A data frame with as many rows as weather stations and the following columns:
  - `MinTemperature-Bias`: Bias (in degrees), calculated over the prediction period, of minimum temperature estimations in weather stations.
  - `MinTemperature-MAE`: Mean absolute errors (in degrees), averaged over the prediction period, of minimum temperature estimations in weather stations.
  - `MaxTemperature-Bias`: Bias (in degrees), calculated over the prediction period, of maximum temperature estimations in weather stations.
  - `MaxTemperature-MAE`: Mean absolute errors (in degrees), averaged over the prediction period, of maximum temperature estimations in weather stations.
  - `Precipitation-Total`: Difference in the total precipitation of the studied period.
  - `Precipitation-DPD`: Difference in the proportion of days with precipitation.
  - `Precipitation-Bias`: Bias (in mm), calculated over the days with precipitation, of precipitation amount estimations in weather stations.
  - `Precipitation-MAE`: Mean absolute errors (in mm), averaged over the days with precipitation, of precipitation amount estimations in weather stations.
  - `RelativeHumidity-Bias`: Bias (in percent), calculated over the prediction period, of relative humidity estimations in weather stations.
  - `RelativeHumidity-MAE`: Mean absolute errors (in percent), averaged over the prediction period, of relative humidity estimations in weather stations.
  - `Radiation-Bias`: Bias (in MJ/m2), calculated over the prediction period, of incoming radiation estimations in weather stations.
  - `Radiation-MAE`: Mean absolute errors (in MJ/m2), averaged over the prediction period, of incoming radiation estimations in weather stations.
- `dates`: A data frame with as many rows as weather stations and the following columns:
  - `MinTemperature-Bias`: Daily bias (in degrees), averaged over the stations, of minimum temperature estimations.
  - `MinTemperature-MAE`: Daily mean absolute error (in degrees), averaged over the stations, of minimum temperature estimations.
  - `MaxTemperature-Bias`: Daily bias (in degrees), averaged over the stations, of maximum temperature estimations.

  - `MaxTemperature-MAE`: Daily mean absolute error (in degrees), averaged over the stations, of maximum temperature estimations.
  - `Precipitation-Bias`: Daily bias (in mm), averaged over the stations, of precipitation amount estimations.
  - `Precipitation-MAE`: Daily mean absolute error (in mm), averaged over the stations, of precipitation amount estimations.
  - `RelativeHumidity-Bias`: Daily bias (in percent), averaged over the stations, of relative humidity estimations.
  - `RelativeHumidity-MAE`: Daily mean absolute error (in percent), averaged over the stations, of relative humidity estimations.
  - `Radiation-Bias`: Daily bias (in MJ/m2), averaged over the stations, of incoming radiation estimations.
  - `Radiation-MAE`: Daily mean absolute errors (in MJ/m2), averaged over the stations, of incoming radiation estimations.
- `MinTemperature`: A data frame with predicted minimum temperature values.
- `MinTemperatureError`: A matrix with predicted minimum temperature errors.
- `MaxTemperature`: A data frame with predicted maximum temperature values.
- `MaxTemperatureError`: A matrix with predicted maximum temperature errors.
- `Precipitation`: A data frame with predicted precipitation values.
- `PrecipitationError`: A matrix with predicted precipitation errors.
- `RelativeHumidity`: A data frame with predicted relative humidity values.
- `RelativeHumidityError`: A matrix with predicted relative humidity errors.
- `Radiation`: A data frame with predicted radiation values.
- `RadiationError`: A matrix with predicted radiation errors.

Methods (by generic)

- `plot(interpolation.cv)`: **[Deprecated]**
- `summary(interpolation.cv)`: **[Deprecated]**

Functions

- `interpolation.calibration()`: **[Deprecated]**
- `interpolation.calibration.fmax()`: **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Thornton, P.E., Running, S.W., 1999. An improved algorithm for estimating incident daily solar radiation from measurements of temperature, humidity, and precipitation. Agric. For. Meteorol. 93, 211–228. doi:10.1016/S0168-1923(98)00126-9.

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

See Also

MeteorologyInterpolationData

Examples

```
data(exampleinterpolationdata)

#Calibration procedure
precEv_cal = interpolation.calibration(exampleinterpolationdata, variable="PrecEvent",
                                       stations = 1:5,
                                     N_seq=c(5,10,15), alpha_seq=seq(0.25,1.0, by=0.25),
                                       verbose = TRUE)

precAm_cal = interpolation.calibration(exampleinterpolationdata, variable="PrecAmount",
                                       stations = 1:5,
                                     N_seq=c(5,10,15), alpha_seq=seq(0.25,1.0, by=0.25),
                                       verbose = TRUE)

#Set 'alpha' and 'N' parameters to values found in calibration
exampleinterpolationdata@params$N_PrecipitationEvent = precEv_cal$N
exampleinterpolationdata@params$alpha_PrecipitationEvent = precEv_cal$alpha

exampleinterpolationdata@params$N_PrecipitationAmount = precAm_cal$N
exampleinterpolationdata@params$alpha_PrecipitationAmount = precAm_cal$alpha

#Run cross validation
cv = interpolation.cv(exampleinterpolationdata, stations = 1:5, verbose = TRUE)

#Print cross validation summaries
summary(cv)

#Plot results
plot(cv)
```

---

interpolation.coverage

Spatial and temporal coverage of interpolation data

Description

[Deprecated]

Function interpolation.coverage calculates, for each meteorological variable, the number of stations with data per date or the number of dates with data per station in an object of class MeteorologyInterpolationData-class.

Usage

```
interpolation.coverage(object, type = "spatial", percent = FALSE)
```

Arguments

object	An object of class MeteorologyInterpolationData-class.
type	A string with the coverage summary to be produced (either "spatial" or "temporal").
percent	A boolean flag to indicate that percentages should be returned instead of counts.

Value

If type = "spatial" the function returns an object of class SpatialPointsDataFrame with the number (or percentage) of dates with data per station and meteorological variable. If type = "temporal" the function returns an object of class data.frame with the number (or percentage) of stations with data per day and meteorological variable.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

See Also

MeteorologyInterpolationData

Examples

```
data(exampleinterpolationdata)

#Number of days with data per station
head(interpolation.coverage(exampleinterpolationdata))

#Number of stations with data per day
head(interpolation.coverage(exampleinterpolationdata, type = "temporal"))
```

`interpolationgrid` *Interpolate daily meteorology over a landscape*

Description

[Deprecated]

Functions to interpolate meteorological data for spatial locations (at points, grid pixels or full grids) using an object of class `MeteorologyInterpolationData-class`.

Usage

```
interpolationgrid(
  object,
  grid,
  dates = NULL,
  exportFile = NULL,
  exportFormat = "netCDF",
  add = FALSE,
  overwrite = FALSE,
  verbose = TRUE
)

interpolationpixels(
  object,
  pixels,
  dates = NULL,
  exportFile = NULL,
  exportFormat = "netCDF",
  add = FALSE,
  overwrite = FALSE,
  verbose = TRUE
)

interpolationpoints(
  object,
  points,
  dates = NULL,
  export = FALSE,
  exportDir = getwd(),
  exportFile = NULL,
  exportFormat = "meteoland/txt",
  metadataFile = "MP.txt",
  verbose = TRUE
)
```

Arguments

object	An object of class MeteorologyInterpolationData-class.
grid	An object of class SpatialGridTopography-class representing the target landscape.
dates	An object of class Date. If this is NULL then all dates in object are processed.
exportFile	Output file for interpolated meteorology data (netCDF format).
exportFormat	Export format for meteorological data (see writemeteorologypoint). If format is "meteoland/txt", "meteoland/rds", "castanea/txt" or "castanea/rds" the function tries to write one file per point in exportDir. If format is "netCDF" the function will write data to a single file specified by exportFile.
add	Boolean flag to indicate that NetCDF exists and data should be added/replaced.
overwrite	Boolean flag to force overwriting an existing NetCDF.
verbose	Boolean flag to print process information.
pixels	An object of class SpatialPixelsTopography-class representing the target landscape.
points	An object of class SpatialPointsTopography-class.
export	If export = FALSE the result of interpolation is stored in memory. Otherwise the result is written in the disk (using the format specified in exportFormat).
exportDir	Output directory for interpolated meteorology data files (txt/rds format).
metadataFile	The name of the ascii text file that will store the meta data describing all written files.

Details

CRS projection needs to be defined for both object and points/pixels/grid. If CRS projection is different between object and points/pixels/grid, the function transforms the coordinates of points/pixels/grid to adapt them to the CRS of object.

Value

If export = FALSE, function interpolationpoints returns an object of SpatialPointsMeteorology-class. If export = TRUE files and written in the disk. For text/rds format the function returns an object of class SpatialPointsDataFrame-class containing point meta data.

If export = FALSE, function interpolationpixels returns an object of SpatialPixelsMeteorology-class, or an object of SpatialPixelsDataFrame-class if a single date is interpolated. If export = TRUE, the function writes the results in a NetCDF.

If export = FALSE, function interpolationgrid returns an object of SpatialGridMeteorology-class, or an object of SpatialGridDataFrame-class if a single date is interpolated. If export = TRUE, the function writes the results in files and a data.frame with columns 'dir' and 'filename' is returned.

Functions

- interpolationgrid(): **[Deprecated]**
- interpolationpixels(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Thornton, P.E., Running, S.W., White, M. A., 1997. Generating surfaces of daily meteorological variables over large regions of complex terrain. J. Hydrol. 190, 214–251. doi:10.1016/S0022-1694(96)03128-9.

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

See Also

penman, SpatialPointsTopography-class, SpatialGridTopography, SpatialPixelsTopography, MeteorologyInterpolationData

Examples

```
data(examplegridtopography)
data(exampleinterpolationdata)

####### INTERPOLATION on particular POINTS

#Creates spatial topography points from the grid
p = 1:2
spt = as(examplegridtopography, "SpatialPointsTopography")[p]

#Interpolation of two points for the whole time period (2000-2003)
mp = interpolationpoints(exampleinterpolationdata, spt)

#Plot interpolated meteorological series
meteoplot(mp,1, ylab="Mean temperature")

####### INTERPOLATION on PIXELS
# Creates spatial topography pixels as a subset of grid pixels
# and select pixels at maximum distance of 2km from center
spt = as(examplegridtopography,"SpatialPointsTopography")
cc = spt@coords
center = 5160
d = sqrt((cc[,1]-cc[center,1])^2+(cc[,2]-cc[center,2])^2)
spxt = as(spt[which(d<2000)], "SpatialPixelsTopography")

# Interpolation of meteorology over pixels for two days
ml = interpolationpixels(exampleinterpolationdata, spxt,
                         as.Date(c("2001-02-03", "2001-06-03")))

#Plot PET corresponding to 2001-06-03
spplot(ml,2,"PET")
```

```
####### INTERPOLATION over a complete GRID
#Interpolation of meteorology over a grid for two days
ml = interpolationgrid(exampleinterpolationdata, examplegridtopography,
                       as.Date(c("2001-02-03", "2001-06-03")))
#Plot PET corresponding to 2001-06-03
spplot(ml,2,"PET")
```

`interpolation_cross_validation`
Calibration and validation of interpolation procedures

Description

Calibration and validation of interpolation procedures

Usage

```
interpolation_cross_validation(
  interpolator,
  stations = NULL,
  verbose = getOption("meteoland_verbosity", TRUE)
)

interpolator_calibration(
  interpolator,
  stations = NULL,
  update_interpolation_params = FALSE,
 variable = c("MinTemperature", "MaxTemperature", "DewTemperature", "Precipitation",
    "PrecipitationAmount", "PrecipitationEvent"),
  N_seq = seq(5, 30, by = 5),
  alpha_seq = seq(0.25, 10, by = 0.25),
  verbose = getOption("meteoland_verbosity", TRUE)
)
```

Arguments

`interpolator`	A meteoland interpolator object, as created by `create_meteo_interpolator`
`stations`	A vector with the stations (numeric for station indexes or character for stations id) to be used to calculate `"MAE"`. All stations with data are included in the training set but predictive `"MAE"` are calculated for the stations subset indicated in `stations` param only. If `NULL` all stations are used in the predictive `"MAE"` calculation.

verbose	Logical indicating if the function must show messages and info. Default value checks "meteoland_verbosity" option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with options(meteoland_verbosity = FALSE)
update_interpolation_params	Logical indicating if the interpolator object must be updated with the calculated parameters. Default to FALSE
variable	A string indicating the meteorological variable for which interpolation parameters "N" and "alpha" will be calibrated. Accepted values are MinTemperature, MaxTemperature, DewTemperature, Precipitation (for precipitation with the same values for precipitation events an regression of precipitation amounts), PrecipitationAmount (for regression of precipitation amounts) and PrecipitationEvent (for precipitation events).
N_seq	Numeric vector with "N" values to be tested
alpha_seq	Numeric vector with "alpha"

Details

Function interpolator_calibration determines optimal interpolation parameters "N" and "alpha" for a given meteorological variable. Optimization is done by minimizing mean absolute error ("MAE") (Thornton *et al.* 1997). Function interpolation_cross_validation calculates average mean absolute errors ("MAE") for the prediction period of the interpolator object. In both calibration and cross validation procedures, predictions for each meteorological station are made using a *leave-one-out* procedure (i.e. after excluding the station from the predictive set).

Value

interpolation_cross_validation returns a list with the following items

- errors: Data frame with each combination of station and date with observed variables, predicated variables and the total error (predicted - observed) calculated for each variable
- station_stats: Data frame with error and bias statistics aggregated by station
- dates_stats: Data frame with error and bias statistics aggregated by date
- r2: correlation indexes between observed and predicted values for each meteorological variable

If update_interpolation_params is FALSE (default), interpolator_calibration returns a list with the following items

- MAE: A numeric matrix with the mean absolute error values, averaged across stations, for each combination of parameters "N" and "alpha"
- minMAE: Minimum MAE value
- N: Value of parameter "N" corresponding to the minimum MAE
- alpha: Value of parameter "alpha" corresponding the the minimum MAE
- observed: matrix with observed values (meteorological measured values)
- predicted: matrix with interpolated values for the optimum parameter combination

If update_interpolation_params is FALSE, interpolator_calibration returns the interpolator provided with the parameters updated

Functions

- interpolation_cross_validation():

Examples

```
# example interpolator
data("meteoland_interpolator_example")

# As the cross validation for all stations can be time consuming, we are
# gonna use only for the first 5 stations of the 198
cv <- interpolation_cross_validation(meteoland_interpolator_example, stations = 1:5)

# Inspect the results
cv$errors
cv$station_stats
cv$dates_stats
cv$r2
```

```
# example interpolator
data("meteoland_interpolator_example")

# As the calibration for all stations can be time consuming, we are gonna
# interpolate only for the first 5 stations of the 198 and only a handful
# of parameter combinations
calibration <- interpolator_calibration(
  meteoland_interpolator_example,
  stations = 1:5,
  variable = "MaxTemperature",
  N_seq = seq(10, 20, by = 5),
  alpha_seq = seq(8, 9, by = 0.25)
)

# we can update the interpolator params directly:
updated_interpolator <- interpolator_calibration(
  meteoland_interpolator_example,
  stations = 1:5,
  update_interpolation_params = TRUE,
  variable = "MaxTemperature",
  N_seq = seq(10, 20, by = 5),
  alpha_seq = seq(8, 9, by = 0.25)
)

# check the new interpolator have the parameters updated
get_interpolation_params(updated_interpolator)$N_MaxTemperature
get_interpolation_params(updated_interpolator)$alpha_MaxTemperature
```

`interpolation_precipitation` *Low-level interpolation functions*

Description

[Deprecated]

Low-level functions to interpolate meteorology (one day) on a set of points.

Usage

```
interpolation_precipitation(
  Xp,
  Yp,
  Zp,
  X,
  Y,
  Z,
  P,
  Psmooth,
  iniRp = 140000,
  alpha_event = 6.25,
  alpha_amount = 6.25,
  N_event = 20L,
  N_amount = 20L,
  iterations = 3L,
  popcrit = 0.5,
  fmax = 0.95,
  debug = FALSE
)

interpolation_dewtemperature(
  Xp,
  Yp,
  Zp,
  X,
  Y,
  Z,
  T,
  iniRp = 140000,
  alpha = 3,
  N = 30L,
  iterations = 3L,
  debug = FALSE
)
```

```
interpolation_temperature(
  Xp,
  Yp,
  Zp,
  X,
  Y,
  Z,
  T,
  iniRp = 140000,
  alpha = 3,
  N = 30L,
  iterations = 3L,
  debug = FALSE
)

interpolation_wind(
  Xp,
  Yp,
  WS,
  WD,
  X,
  Y,
  iniRp = 140000,
  alpha = 2,
  N = 1L,
  iterations = 3L,
  directionsAvailable = TRUE
)
```

Arguments

`Xp, Yp, Zp`	Spatial coordinates and elevation (Zp; in m.a.s.l) of target points.
`X, Y, Z`	Spatial coordinates and elevation (Zp; in m.a.s.l) of reference locations (e.g. meteorological stations).
`P`	Precipitation at the reference locations (in mm).
`Psmooth`	Temporally-smoothed precipitation at the reference locations (in mm).
`iniRp`	Initial truncation radius.
`iterations`	Number of station density iterations.
`popcrit`	Critical precipitation occurrence parameter.
`fmax`	Maximum value for precipitation regression extrapolations (0.6 equals to a maximum of 4 times extrapolation).
`debug`	Boolean flag to show extra console output.
`T`	Temperature (e.g., minimum, maximum or dew temperature) at the reference locations (in degrees).
`alpha, alpha_amount, alpha_event`	Gaussian shape parameter.

`N, N_event, N_amount`
: Average number of stations with non-zero weights.

`WS, WD`
: Wind speed (in m/s) and wind direction (in degrees from north clock-wise) at the reference locations.

`directionsAvailable`
: A flag to indicate that wind directions are available (i.e. non-missing) at the reference locations.

Details

This functions exposes internal low-level interpolation functions written in C++ not intended to be used directly in any script or function. The are maintained for compatibility with older versions of the package and future versions of meteoland will remove this functions (they will be still accessible through the triple colon notation (`:::`), but their use is not recommended)

Value

All functions return a vector with interpolated values for the target points.

Functions

- `interpolation_precipitation()`: Precipitation
- `interpolation_dewtemperature()`: Dew temperature
- `interpolation_wind()`: Wind

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Thornton, P.E., Running, S.W., White, M. A., 1997. Generating surfaces of daily meteorological variables over large regions of complex terrain. J. Hydrol. 190, 214–251. doi:10.1016/S0022-1694(96)03128-9.

De Caceres M, Martin-StPaul N, Turco M, Cabon A, Granda V (2018) Estimating daily meteorological data and downscaling climate models over landscapes. Environmental Modelling and Software 108: 186-196.

See Also

`defaultInterpolationParams`

Examples

```
Xp <- as.numeric(sf::st_coordinates(points_to_interpolate_example)[,1])
Yp <- as.numeric(sf::st_coordinates(points_to_interpolate_example)[,2])
Zp <- points_to_interpolate_example$elevation
X <- as.numeric(
```

```
 sf::st_coordinates(stars::st_get_dimension_values(meteoland_interpolator_example, "station"))[,1]
)
Y <- as.numeric(
 sf::st_coordinates(stars::st_get_dimension_values(meteoland_interpolator_example, "station"))[,2]
)
Z <- as.numeric(meteoland_interpolator_example[["elevation"]][1,])
Temp <- as.numeric(meteoland_interpolator_example[["MinTemperature"]][1,])
P <- as.numeric(meteoland_interpolator_example[["Precipitation"]][1,])
Psmooth <- as.numeric(meteoland_interpolator_example[["SmoothedPrecipitation"]][1,])
WS <- as.numeric(meteoland_interpolator_example[["WindSpeed"]][1,])
WD <- as.numeric(meteoland_interpolator_example[["WindDirection"]][1,])
iniRp <- get_interpolation_params(meteoland_interpolator_example)$initial_Rp
alpha <- get_interpolation_params(meteoland_interpolator_example)$alpha_MinTemperature
N <- get_interpolation_params(meteoland_interpolator_example)$N_MinTemperature
alpha_event <- get_interpolation_params(meteoland_interpolator_example)$alpha_PrecipitationEvent
N_event <- get_interpolation_params(meteoland_interpolator_example)$N_PrecipitationEvent
alpha_amount <- get_interpolation_params(meteoland_interpolator_example)$alpha_PrecipitationAmount
N_amount <- get_interpolation_params(meteoland_interpolator_example)$N_PrecipitationAmount
alpha_wind <- get_interpolation_params(meteoland_interpolator_example)$alpha_Wind
N_wind <- get_interpolation_params(meteoland_interpolator_example)$N_Wind
iterations <- get_interpolation_params(meteoland_interpolator_example)$iterations
popcrit <- get_interpolation_params(meteoland_interpolator_example)$pop_crit
fmax <- get_interpolation_params(meteoland_interpolator_example)$f_max
debug <- get_interpolation_params(meteoland_interpolator_example)$debug

interpolation_temperature(
  Xp, Yp, Zp,
  X[!is.na(Temp)], Y[!is.na(Temp)], Z[!is.na(Temp)],
  Temp[!is.na(Temp)],
  iniRp, alpha, N, iterations, debug
)

interpolation_wind(
  Xp, Yp,
  WS[!is.na(WD)], WD[!is.na(WD)],
  X[!is.na(WD)], Y[!is.na(WD)],
  iniRp, alpha_wind, N_wind, iterations, directionsAvailable = FALSE
)

interpolation_precipitation(
  Xp, Yp, Zp,
  X[!is.na(P)], Y[!is.na(P)], Z[!is.na(P)],
  P[!is.na(P)], Psmooth[!is.na(P)],
  iniRp, alpha_event, alpha_amount, N_event, N_amount,
  iterations, popcrit, fmax, debug
)

data("exampleinterpolationdata")
mxt100 = exampleinterpolationdata@MaxTemperature[,100]
Psmooth100 = exampleinterpolationdata@SmoothedPrecipitation[,100]
P100 = exampleinterpolationdata@Precipitation[,100]
mismxt = is.na(mxt100)
misP = is.na(P100)
```

```
Z = exampleinterpolationdata@elevation
X = exampleinterpolationdata@coords[,1]
Y = exampleinterpolationdata@coords[,2]
Zpv = seq(0,1000, by=100)
xp = 360000
yp = 4640000
xpv = rep(xp, 11)
ypv = rep(yp, 11)

interpolation_temperature(xpv, ypv, Zpv,
                          X[!mismxt], Y[!mismxt], Z[!mismxt],
                          mxt100[!mismxt])
interpolation_precipitation(xpv, ypv, Zpv,
                           X[!misP], Y[!misP], Z[!misP],
                           P100[!misP], Psmooth100[!misP])
```

`mergegrids` *Merges meteorological data*

Description

[Deprecated]

Merges point or gridded meteorological data into a single object

Usage

```
mergegrids(..., verbose = TRUE)

mergepoints(..., verbose = TRUE)
```

Arguments

`...`	Objects to be merged, either of class `SpatialPointsMeteorology`, `SpatialGridMeteorology` or `SpatialPixelsMeteorology`. All objects of the same class.
`verbose`	A logical flag to indicate console output.

Details

Function `mergepoints` requires all coordinate reference systems to the same. The function allows merging data from the same points (i.e. if they have the same coordinates). Function `mergegrids` pools dates and variables, but spatial structures (i.e. grid topology, pixel indices, reference system, ...) should be the same for all objects to be merged.

Value

Function codemergepoints returns an object `SpatialPointsMeteorology-class`. Function `mergegrids` returns an object `SpatialGridMeteorology-class` or an object `SpatialPixelsMeteorology-class`, depending on the input.

Functions

- `mergepoints()`: **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

SpatialPointsMeteorology-class, SpatialGridMeteorology-class, SpatialPixelsMeteorology-class

---

meteocomplete *Complete daily meteorological variables*

Description

[Deprecated]

Fills missing values of relative humidity, radiation and potential evapotranspiration from a data frame with daily values of minimum/maximum/mean temperature and precipitation.

Usage

```
meteocomplete(x, latitude, elevation, slope, aspect)
```

Arguments

`x`	A data frame with dates as row names and columns named 'MeanTemperature', 'MaxTemperature', 'MinTemperature' and 'Precipitation'
`latitude`	Latitude in degrees North.
`elevation`	Elevation in m.a.s.l.
`slope`	Slope in degrees.
`aspect`	Aspect in degrees from North.

Details

The function fills values for humidity, radiation and PET only if they are missing in the input data frame. If a column 'SpecificHumidity' is present in the input data, relative humidity is calculated from it. Otherwise, relative humidity is calculated assuming that dew point temperature equals the minimum temperature. Potential solar radiation is calculated from latitude, slope and aspect. Incoming solar radiation is then corrected following Thornton & Running (1999) and potential evapotranspiration following Penman (1948).

Value

A data frame copied from `x` but with filled values for variables:

- `MeanRelativeHumidity`: Mean daily relative humidity (in percent).
- `MinRelativeHumidity`: Minimum daily relative humidity (in percent).
- `MaxRelativeHumidity`: Maximum daily relative humidity (in percent).
- `Radiation`: Incoming solar radiation (MJ/m2).
- `PET`: Potential evapotranspiration (in mm of water).

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Thornton, P.E., Running, S.W., 1999. An improved algorithm for estimating incident daily solar radiation from measurements of temperature, humidity, and precipitation. Agric. For. Meteorol. 93, 211-228.

Penman, H. L. 1948. Natural evaporation from open water, bare soil and grass. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 193, 120-145.

See Also

`penman`, `radiation_solarRadiation`

`meteoland_interpolator_example` *Example interpolator object*

Description

[Experimental]

Example interpolator with daily meteorological records from 189 weather stations in Catalonia (NE Spain) corresponding to April 2022.

Format

stars data cube object

Source

Spanish National Forest Inventory

Examples

```
data(meteoland_interpolator_example)
```

`meteoland_meteo_example`

Example data set for meteo data from weather stations

Description

[Experimental]

Example data set of spatial location, topography and daily meteorological records from 189 weather stations in Catalonia (NE Spain) corresponding to April 2022.

Format

sf object

Source

'Servei Meteorològic de Catalunya' (SMC)

Examples

```
data(meteoland_meteo_example)
```

`meteoland_meteo_no_topo_example`

Example data set for meteo data from weather stations, without topography

---

Description

[Experimental]

Example data set of spatial location and daily meteorological records from 189 weather stations in Catalonia (NE Spain) corresponding to April 2022.

Format

sf object

Source

'Servei Meteorològic de Catalunya' (SMC)

Examples

```
data(meteoland_meteo_no_topo_example)
```

meteoland_topo_example

Example data set for topography data from weather stations, without meteo

Description

[Experimental]

Example data set of spatial location and topography records from 189 weather stations in Catalonia (NE Spain).

Format

sf object

Source

'Servei Meteorològic de Catalunya' (SMC)

Examples

```
data(meteoland_topo_example)
```

meteoplot

Plots point meteorological series

Description

[Deprecated]

Simple plotting of a meteorological series for a given point.

Usage

```
meteoplot(
  object,
  index = 1,
  var = "MeanTemperature",
  fun = NULL,
  freq = NULL,
  dates = NULL,
  months = NULL,
  add = FALSE,
  ...
)
```

Arguments

object	A data frame with daily meteorological data (in this case index is not used) or an object of class SpatialPointsMeteorology. Alternatively, an object of class SpatialPointsDataFrame containing the meta data (columns dir, filename and possibly format) of meteorological files.
index	An integer to indicate the point in the SpatialPointsMeteorology object (or the SpatialPointsDataFrame object).
var	The meteorological variable to be plotted.
fun	The name of a function to be calculated for summaries (only valid if freq is specified).
freq	A string giving an interval specification for summaries (e.g., "week", "month", "quarter" or "year").
dates	An object of class Date to define the period to be plotted. If dates = NULL then all dates in object are processed.
months	A numeric vector to indicate the subset of months for which plotting is desired (e.g. c(7,8) for July and August). When combined with fun and freq, this parameter allows plotting summaries for particular seasons. For example fun = "sum" freq = "years" and months = 6:8 leads to plotting the sum over summer months of each year.
add	A flag to indicate wether drawing should be done on the current plot (using function lines).
...	Additional parameters for functions plot or lines.

Details

Daily precipitation is plotted using bars (i.e. type = "h" when calling plot). Otherwise the function draws lines (i.e. type = "l" when calling plot). If object is of class SpatialPointsDataFrame-class then the function reads the meteorological data to be plotted from the disk.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

summarypoints

Examples

```
data(examplegridtopography)
data(exampleinterpolationdata)

#Creates spatial topography points from the grid
p = 1:2
spt = as(examplegridtopography, "SpatialPointsTopography")[p]
```

```
#Interpolation of two points for the whole time period (2000-2003)
mp = interpolationpoints(exampleinterpolationdata, spt)

#Plot interpolated meteorological series
meteoplot(mp,1, ylab="Daily mean temperature")

meteoplot(mp,1, ylab="Monthly mean temperature", fun=mean, freq="months")
```

`MeteorologyInterpolationData` *Creates an object of class 'MeteorologyInterpolationData'*

Description

[Deprecated]

Initializes an object for meteorology interpolation over landscapes using weather station data and the methods described in Thornton et al. (1997) and Thornton & Running (1999).

Usage

```
MeteorologyInterpolationData(
  points,
  elevation = NULL,
  slope = NULL,
  aspect = NULL,
  MinTemperature = NULL,
  MaxTemperature = NULL,
  Precipitation = NULL,
  RelativeHumidity = NULL,
  Radiation = NULL,
  WindSpeed = NULL,
  WindDirection = NULL,
  WindFields = NULL,
  params = defaultInterpolationParams()
)
```

Arguments

`points`	An object of class `SpatialPointsMeteorology`, an object of `SpatialPointsTopography` or an object of class `SpatialPoints` (see 'Details").
`elevation`	A numeric vector with elevation values of weather stations (in meters).
`slope`	A numeric vector with slope values of weather stations (in degrees). Needed for cross-validation of interpolation routines.

aspect	A numeric vector with aspect values of weather stations (in degrees from North). Needed for cross-validation of interpolation routines.
MinTemperature	A matrix with minimum temperature recordings (in degrees Celsius) for all weather stations (in rows) and all days (in columns).
MaxTemperature	A matrix with maximum temperature recordings (in degrees Celsius) for all weather stations (in rows) and all days (in columns).
Precipitation	A matrix with precipitation recordings (in mm of water) for all weather stations (in rows) and all days (in columns).
RelativeHumidity	A matrix with (mean) relative humidity recordings (in percent) for all weather stations (in rows) and all days (in columns).
Radiation	A matrix with relative radiation recordings (in MJ/m2) for all weather stations (in rows) and all days (in columns). Needed for cross-validation only.
WindSpeed	A matrix with wind speed recordings (in m/s) for all weather stations (in rows) and all days (in columns).
WindDirection	A matrix with wind direction recordings (in degrees from North) for all weather stations (in rows) and all days (in columns).
WindFields	Object of class "list". See function readWindNinjaWindFields.
params	A list containing interpolation parameters.

Details

There are three ways of building an object of MeteorologyInterpolationData:

1. The first way is using an object of SpatialPointsMeteorology containing both the coordinates and meteorological series of stations. In this case elevation has to be provided, but aspect and slope may be omitted. Parameters MinTemperature to WindDirection can be left as NULL.
2. The second way is using an object of class of SpatialPointsTopography containing the coordinates of stations and topographic variables. In this case parameters MinTemperature and MaxTemperature will need to be supplied, each being a matrix with weather stations in rows and days in columns, but Precipitation, RelativeHumidity, Radiation, WindSpeed and WindDirection may be left as NULL.
3. The third way is using an object of SpatialPoints containing the coordinates of stations only. In this case elevation has to be provided, but aspect and slope may be omitted. As in the second case, parameters MinTemperature and MaxTemperature will need to be supplied, each being a matrix with weather stations in rows and days in columns, but other variables may be left as NULL.

Value

An object of class MeteorologyInterpolationData.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Thornton, P.E., Running, S.W., 1999. An improved algorithm for estimating incident daily solar radiation from measurements of temperature, humidity, and precipitation. Agric. For. Meteorol. 93, 211–228. doi:10.1016/S0168-1923(98)00126-9.

Thornton, P.E., Running, S.W., White, M. a., 1997. Generating surfaces of daily meteorological variables over large regions of complex terrain. J. Hydrol. 190, 214–251. doi:10.1016/S0022-1694(96)03128-9.

See Also

MeteorologyInterpolationData, defaultInterpolationParams.

MeteorologyInterpolationData-class

Class "MeteorologyInterpolationData"

Description

[Deprecated]

An S4 class to interpolate meteorology over a landscape.

Objects from the Class

Objects can be created by calls of the form `new("MeteorologyInterpolationData", ...)`, or by calls to the function MeteorologyInterpolationData.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

MeteorologyInterpolationData, MeteorologyProcedureData-class, subsample

Examples

```
#Structure of the S4 object
showClass("MeteorologyInterpolationData")
```

MeteorologyProcedureData-class

Class "MeteorologyProcedureData"

Description

[Deprecated]

A virtual class for estimating meteorology over landscapes

Objects from the Class

A virtual Class: No objects may be created from it.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

MeteorologyInterpolationData-class, MeteorologyUncorrectedData-class

Examples

```
showClass("MeteorologyProcedureData")
```

MeteorologyUncorrectedData

Creates an object of class 'MeteorologyUncorrectedData'

Description

[Deprecated]

Initializes an object for statistical correction of meteorological data over landscapes.

Usage

```
MeteorologyUncorrectedData(
  points,
  reference_data,
  projection_data,
  dates,
  params = defaultCorrectionParams()
)
```

Arguments

points	An object of class SpatialPoints.
reference_data	Reference (historic) meteorological data used to calibrate correction factors when compared with observations. A vector of data frames (one per point) or a single data frame containing the meta data (columns dir and filename) of meteorological files that will be read from the disk. Alternatively, a NetCDF file name where points should be read.
projection_data	Projected meteorological data to be corrected. A vector of data frames (one per point) or a single data frame containing the meta data (columns dir and filename) of meteorological files that will be read from the disk. Alternatively, a NetCDF file name where points should be read.
dates	Object of class "Date" describing the time period for which meteorological correction is possible (corresponding to projection_data).
params	A "list" containing correction parameters.

Details

See correction details in vignettes or in correctionpoints.

Value

An object of class MeteorologyUncorrectedData.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

MeteorologyUncorrectedData, examplecorrectiondata, defaultCorrectionParams.

MeteorologyUncorrectedData-class

Class "MeteorologyUncorrectedData"

Description

[Deprecated]

An S4 class to conduct statistical correction of meteorology over a landscape.

Objects from the Class

Objects can be created by calls of the form new("MeteorologyUncorrectedData", ...), or by calls to the function MeteorologyUncorrectedData.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

MeteorologyUncorrectedData, MeteorologyProcedureData-class, examplecorrectiondata, subsample

Examples

```
#Structure of the S4 object
showClass("MeteorologyUncorrectedData")
```

---

meteospain2meteoland *From meteospain to meteoland meteo objects*

Description

Adapting meteospain meteo objects to meteoland meteo objects

Usage

```
meteospain2meteoland(meteo, complete = FALSE)
```

Arguments

meteo	meteospain meteo object.
complete	logical indicating if the meteo data missing variables should be calculated (if possible). Default to FALSE.

Details

This function converts meteospain R package meteo objects to compatible meteoland meteo objects by selecting the needed variables and adapting the names to comply with meteoland requirements.

Value

a compatible meteo object to use with meteoland.

Examples

```
if (interactive()) {
  # meteospain data
  library(meteospain)
  mg_april_2022_data <- get_meteo_from(
    "meteogalicia",
    meteogalicia_options("daily", as.Date("2022-04-01"), as.Date("2022-04-30"))
  )

  # just convert
  meteospain2meteoland(mg_april_2022_data)
  # convert and complete
  meteospain2meteoland(mg_april_2022_data, complete = TRUE)

}
```

---

penman *Potential evapotranspiration*

Description

Functions to calculate potential evapotranspiration using Penman or Penman-Monteith.

Usage

```
penman(
  latrad,
  elevation,
  slorad,
  asprad,
  J,
  Tmin,
  Tmax,
  RHmin,
  RHmax,
  R_s,
  u,
  z = 2,
  z0 = 0.001,
  alpha = 0.08,
  windfun = "1956"
)

penmanmonteith(rc, elevation, Tmin, Tmax, RHmin, RHmax, Rn, u = NA_real_)
```

Arguments

latrad	Latitude in radians.
elevation	Elevation (in m).
slorad	Slope (in radians).
asprad	Aspect (in radians from North).
J	Julian day, number of days since January 1, 4713 BCE at noon UTC.
Tmin	Minimum temperature (degrees Celsius).
Tmax	Maximum temperature (degrees Celsius).
RHmin	Minimum relative humidity (percent).
RHmax	Maximum relative humidity (percent).
R_s	Solar radiation (MJ/m2).
u	With wind speed (m/s).
z	Wind measuring height (m).
z0	Roughness height (m).
alpha	Albedo.
windfun	Wind speed function version, either "1948" or "1956".
rc	Canopy vapour flux (stomatal) resistance (s·m-1).
Rn	Daily net radiation (MJ·m-2·day-1).

Details

The code was adapted from package 'Evapotranspiration', which follows McMahon et al. (2013). If wind speed is not available, an alternative formulation for potential evapotranspiration is used as an approximation (Valiantzas 2006)

Value

Potential evapotranspiration (in mm of water).

Functions

- `penmanmonteith()`: Penman Monteith method

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Penman, H. L. 1948. Natural evaporation from open water, bare soil and grass. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 193, 120-145.

Penman, H. L. 1956. Evaporation: An introductory survey. Netherlands Journal of Agricultural Science, 4, 9-29.

McMahon, T.A., Peel, M.C., Lowe, L., Srikanthan, R., McVicar, T.R. 2013. Estimating actual, potential, reference crop and pan evaporation using standard meteorological data: a pragmatic synthesis. Hydrology & Earth System Sciences 17, 1331–1363. doi:10.5194/hess-17-1331-2013.

See Also

interpolationpoints

`points_to_interpolate_example` *Example data set of points for interpolation of weather variables*

Description

[Experimental]

Example data set of spatial location and topography records from 15 experimental plots in Catalonia (NE Spain).

Format

sf object

Source

Spanish National Forest Inventory

Examples

```
data(points_to_interpolate_example)
```

`precipitation_concentration` *Precipitation daily concentration*

Description

[Deprecated]

Function `precipitation_concentration()` calculates daily precipitation concentration (Martin-Vide et al. 2004).

Usage

```
precipitation_concentration(p)
```

Arguments

p	A numeric vector with daily precipitation values.

Value

Function `precipitation_concentration()` returns a value between 0 (equal distribution of rainfall) and 1 (one day concentrates all rainfall).

Author(s)

Miquel De Cáceres Ainsa, CREAF.

References

Martin-Vide J (2004) Spatial distribution of a daily precipitation concentration index in peninsular Spain. International Journal of Climatology 24, 959–971. doi:10.1002/joc.1030.

`precipitation_rainfallErosivity`

Precipitation rainfall erosivity

Description

Function `precipitation_rainfallErosivity()` calculates a multi-year average of monthly rainfall erosivity using the MedREM model proposed by Diodato and Bellochi (2010) for the Mediterranean area (see also Guerra et al. 2016).

Usage

```
precipitation_rainfallErosivity(x, long, scale = "month", average = TRUE)
```

Arguments

`x`	A data frame of meteorological variables (dates as row names and variables as columns).
`long`	Longitude in degrees.
`scale`	Either 'month' or 'year'.
`average`	Boolean flag to calculate multi-year averages before applying MedREM's formula.

Details

MedREM model is: Rm = b0·P·sqrt(d)·(alpha + b1*longitude), where P is accumulated precipitation and d is maximum daily precipitation. Parameters used for the MedREM model are b0 = 0.117, b1 = -0.015, alpha = 2. Note that there is a mistake in Guerra et al. (2016) regarding parameters b1 and a.

Value

Function `precipitation_rainfallErosivity()` returns a vector of twelve values (one for each month) (in MJ·mm·ha-1·h-1·month-1) or one value (in in MJ·mm·ha-1·h-1·yr-1) depending on the scale.

Author(s)

Miquel De Cáceres Ainsa, CREAF.

References

Diodato, N., Bellocchi, G., 2010. MedREM, a rainfall erosivity model for the Mediter-ranean region. J. Hydrol. 387, 119–127, doi:10.1016/j.jhydrol.2010.04.003.

Guerra CA, Maes J, Geijzendorffer I, Metzger MJ (2016) An assessment of soil erosion prevention by vegetation in Mediterranean Europe: Current trends of ecosystem service provision. Ecol Indic 60:213–222. doi: 10.1016/j.ecolind.2015.06.043.

---

precipitation_rainfall_erosivity

Precipitation rainfall erosivity

Description

[Experimental]

Function `precipitation_rainfall_erosivity()` calculates a multi-year average of monthly rainfall erosivity using the MedREM model proposed by Diodato and Bellochi (2010) for the Mediterranean area (see also Guerra et al. 2016).

Usage

```
precipitation_rainfall_erosivity(
  meteo_data,
  longitude,
  scale = c("month", "year"),
  average = TRUE
)
```

Arguments

`meteo_data`	A meteo tibble as with the dates and meteorological variables as returned by `interpolate_data` in the "interpolated_data" column.
`longitude`	Longitude in degrees.
`scale`	Character, either 'month' or 'year'. Default to 'month'
`average`	Boolean flag to calculate multi-year averages before applying MedREM's formula.

Details

MedREM model is: Rm = b0·P·sqrt(d)·(alpha + b1*longitude), where P is accumulated precipitation and d is maximum daily precipitation. Parameters used for the MedREM model are b0 = 0.117, b1 = -0.015, alpha = 2. Note that there is a mistake in Guerra et al. (2016) regarding parameters b1 and a.

Value

A vector of values for each month (in MJ·mm·ha-1·h-1·month-1) or each year (in MJ·mm·ha-1·h-1·yr-1), depending on the scale

Author(s)

Miquel De Cáceres Ainsa, CREAF.

Víctor Granda García, CREAF.

References

Diodato, N., Bellocchi, G., 2010. MedREM, a rainfall erosivity model for the Mediterranean region. J. Hydrol. 387, 119–127, doi:10.1016/j.jhydrol.2010.04.003.

Guerra CA, Maes J, Geijzendorffer I, Metzger MJ (2016) An assessment of soil erosion prevention by vegetation in Mediterranean Europe: Current trends of ecosystem service provision. Ecol Indic 60:213–222. doi: 10.1016/j.ecolind.2015.06.043.

Examples

```
interpolated_example <-
  interpolate_data(points_to_interpolate_example, meteoland_interpolator_example)

precipitation_rainfall_erosivity(
  meteo_data = interpolated_example$interpolated_data[[1]],
  longitude = 2.32,
  scale = "month",
  average = TRUE
)
```

`radiation_julianDay` *Solar radiation utility functions*

Description

Set of functions used in the calculation of incoming solar radiation and net radiation.

Usage

```
radiation_julianDay(year, month, day)

radiation_dateStringToJulianDays(dateStrings)

radiation_solarDeclination(J)
```

```
radiation_solarConstant(J)

radiation_sunRiseSet(latrad, slorad, asprad, delta)

radiation_solarElevation(latrad, delta, hrad)

radiation_daylength(latrad, slorad, asprad, delta)

radiation_daylengthseconds(latrad, slorad, asprad, delta)

radiation_potentialRadiation(solarConstant, latrad, slorad, asprad, delta)

radiation_solarRadiation(
  solarConstant,
  latrad,
  elevation,
  slorad,
  asprad,
  delta,
  diffTemp,
  diffTempMonth,
  vpa,
  precipitation
)

radiation_directDiffuseInstant(
  solarConstant,
  latrad,
  slorad,
  asprad,
  delta,
  hrad,
  R_s,
  clearday
)

radiation_directDiffuseDay(
  solarConstant,
  latrad,
  slorad,
  asprad,
  delta,
  R_s,
  clearday,
  nsteps = 24L
)

radiation_skyLongwaveRadiation(Tair, vpa, c = 0)
```

```
radiation_outgoingLongwaveRadiation(
  solarConstant,
  latrad,
  elevation,
  slorad,
  asprad,
  delta,
  vpa,
  tmin,
  tmax,
  R_s
)

radiation_netRadiation(
  solarConstant,
  latrad,
  elevation,
  slorad,
  asprad,
  delta,
  vpa,
  tmin,
  tmax,
  R_s,
  alpha = 0.08
)
```

Arguments

`year, month, day`	Year, month and day as integers.
`dateStrings`	A character vector with dates in format "YYYY-MM-DD".
`J`	Julian day (integer), number of days since January 1, 4713 BCE at noon UTC.
`latrad`	Latitude (in radians North).
`slorad`	Slope (in radians).
`asprad`	Aspect (in radians from North).
`delta`	Solar declination (in radians).
`hrad`	Solar hour (in radians).
`solarConstant`	Solar constant (in kW·m-2).
`elevation`	Elevation above sea level (in m).
`diffTemp`	Difference between maximum and minimum temperature (ºC).
`diffTempMonth`	Difference between maximum and minimum temperature, averaged over 30 days (ºC).
`vpa`	Average daily vapor pressure (kPa).

precipitation	Precipitation (in mm).
R_s	Daily incident solar radiation (MJ·m-2).
clearday	Boolean flag to indicate a clearsky day (vs. overcast).
nsteps	Number of daily substeps.
Tair	Air temperature (in degrees Celsius).
c	Proportion of sky covered by clouds (0-1).
tmin, tmax	Minimum and maximum daily temperature (ºC).
alpha	Surface albedo (from 0 to 1).

Value

Values returned for each function are:

- radiation_dateStringToJulianDays: A vector of Julian days (i.e. number of days since January 1, 4713 BCE at noon UTC).
- radiation_daylength: Day length (in hours).
- radiation_daylengthseconds: Day length (in seconds).
- radiation_directDiffuseInstant: A vector with instantaneous direct and diffusive radiation rates (for both SWR and PAR).
- radiation_directDiffuseDay: A data frame with instantaneous direct and diffusive radiation rates (for both SWR and PAR) for each subdaily time step.
- radiation_potentialRadiation: Daily (potential) solar radiation (in MJ·m-2).
- radiation_julianDay: Number of days since January 1, 4713 BCE at noon UTC.
- radiation_skyLongwaveRadiation: Instantaneous incoming (sky) longwave radiation (W·m-2).
- radiation_outgoingLongwaveRadiation: Daily outgoing longwave radiation (MJ·m-2·day-1).
- radiation_netRadiation: Daily net solar radiation (MJ·m-2·day-1).
- radiation_solarConstant: Solar constant (in kW·m-2).
- radiation_solarDeclination: Solar declination (in radians).
- radiation_solarElevation: Angle of elevation of the sun with respect to the horizon (in radians).
- radiation_solarRadiation: Daily incident solar radiation (MJ·m-2·day-1).
- radiation_sunRiseSet: Sunrise and sunset hours in hour angle (radians).

Functions

- radiation_dateStringToJulianDays(): Date string to julian days
- radiation_solarDeclination(): solar declination
- radiation_solarConstant(): solar constant
- radiation_sunRiseSet(): sun rise and set
- radiation_solarElevation(): solar elevation

- `radiation_daylength()`: Day length
- `radiation_daylengthseconds()`: Day length seconds
- `radiation_potentialRadiation()`: Potential radiation
- `radiation_solarRadiation()`: solar Radiation
- `radiation_directDiffuseInstant()`: Direct diffuse instant
- `radiation_directDiffuseDay()`: Direct diffuse day
- `radiation_skyLongwaveRadiation()`: Sky longwave radiation
- `radiation_outgoingLongwaveRadiation()`: Outgoing longwave radiation
- `radiation_netRadiation()`: Net radiation

Note

Code for `radiation_julianDay()`, `radiation_solarConstant()` and `radiation_solarDeclination()` was translated to C++ from R code in package 'insol' (by J. G. Corripio).

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Danby, J. M. Eqn. 6.16.4 in Fundamentals of Celestial Mechanics, 2nd ed. Richmond, VA: Willmann-Bell, p. 207, 1988.

Garnier, B.J., Ohmura, A., 1968. A method of calculating the direct shortwave radiation income of slopes. J. Appl. Meteorol. 7: 796-800

McMahon, T. A., M. C. Peel, L. Lowe, R. Srikanthan, and T. R. McVicar. 2013. Estimating actual, potential, reference crop and pan evaporation using standard meteorological data: a pragmatic synthesis. Hydrology & Earth System Sciences 17:1331–1363. See also: http://www.fao.org/docrep/x0490e/x0490e06.htm.

Reda, I. and Andreas, A. 2003. Solar Position Algorithm for Solar Radiation Applications. 55 pp.; NREL Report No. TP-560-34302, Revised January 2008. http://www.nrel.gov/docs/fy08osti/34302.pdf

Spitters, C.J.T., Toussaint, H.A.J.M. and Goudriaan, J. (1986). Separating the diffuse and direct components of global radiation and its implications for modeling canopy photosynthesis. I. Components of incoming radiation. Agricultural and Forest Meteorology, 38, 231–242.

See Also

`interpolationpoints`

raster_to_interpolate_example *Example raster data set for interpolation of weather variables*

Description

[Experimental]

Example raster data set of spatial location and topography records from Catalonia (NE Spain). Cell size is 1km x 1km and raster size is 10x10 cells.

Format

stars object

Source

ICGC

Examples

```
data(raster_to_interpolate_example)
```

readmeteorologygrid *Reads gridded meteorology from the disk*

Description

[Deprecated]

Functions to read gridded meteorological data from the disk.

Usage

```
readmeteorologygrid(
  files,
  format = "netCDF",
  varmapping = NULL,
  dates = NULL,
  bbox = NULL,
  offset = 0,
  verbose = FALSE
)

readmeteorologypixels(
```

```
  files,
  format = "netCDF",
  varmapping = NULL,
  dates = NULL,
  bbox = NULL,
  offset = 0,
  verbose = FALSE
)

readmeteorologygridpoints(
  files,
  format = "netCDF",
  varmapping = NULL,
  dates = NULL,
  bbox = NULL,
  offset = 0,
  relativehumidity = FALSE,
  verbose = FALSE
)
```

Arguments

files	Character vector with the file names to be read.
format	Format of meteorological data. Currently, the only accepted format is "netCDF".
varmapping	Named character vector specifying a mapping of variables in the NetCDF into variables used in meteoland (e.g. c(MinTemperature = "tmn") specifies a map of variable 'tmn' to MinTemperature).
dates	A character or Date vector to specify subset of dates to be read.
bbox	Boundary box (2 x 2 matrix) specifying the minimum and maximum coordinates of a study area.
offset	A buffer to include NetCDF cells that are at a certain distance around the boundary box.
verbose	A logical flag to output process information in the console.
relativehumidity	A logical flag to indicate estimation of relative humidity from specific humidity if possible.

Details

Function readmeteorologygrid reads one or several files containing the meteorology over a grid for a set of days. Function readmeteorologypixels reads one or several file containing the meteorology over a grid for a set of days and filters those pixels with missing data. If more than one file is specified, the functions read all of them and then try to merge the data into a single meteorology object (see function mergegrids).

Function readmeteorologygridpoints is similar to the preceding ones, but is meant to extract specific grid pixels and return them as spatial points. If more than one file is specified, the function

reads all of them and then tries to merge the data into a single meteorology object (see function mergepoints).

The functions are primarily meant to read NetCDF written by package meteoland, but also to import data written by other software. In this case, a mapping can be supplied to map variable names in the netCDF to variables used in meteoland. Rotated grids should not be read using functions readmeteorologygrid or readmeteorologypixels.

Value

- Function readmeteorologygrid returns an object SpatialGridMeteorology-class.
- Function readmeteorologypixels returns an object SpatialPixelsMeteorology-class.
- Function readmeteorologygridpoints returns an object SpatialPointsMeteorology-class.

Functions

- readmeteorologypixels(): **[Deprecated]**
- readmeteorologygridpoints(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

writemeteorologygrid, writemeteorologypixels, SpatialPointsMeteorology-class, SpatialGridMeteorology-cl
SpatialPixelsMeteorology-class, mergegrids, mergepoints

---

readmeteorologypoint *Reads point meteorology from the disk*

Description

[Deprecated]

Functions to read point meteorological data from the disks in different formats.

Usage

```
readmeteorologypoint(file, dates = NULL, format = "meteoland/txt", sep = "\t")

readmeteorologypointfiles(
  points,
  files = NULL,
  dates = NULL,
  format = "meteoland/txt",
  sep = "\t"
)
```

```
readmeteorologypoints(
  files,
  dates = NULL,
  stations = NULL,
  format = "netCDF",
  varmapping = NULL,
  verbose = FALSE
)
```

Arguments

file	A string of the file to be read.
dates	Object of class "Date" describing a subset of dates to be extracted from meteorological series. If NULL the whole period read from files is kept.
format	Format of meteorological data. Current accepted formats for readmeteorologypoint and readmeteorologypointfiles are "meteoland/txt", "meteoland/rds", "castanea/txt" and "castanea/rds". The only accepted format for readmeteorologypoints is "netCDF".
sep	The field separator character for ascii text files (see read.table).
points	An object of class SpatialPoints-class (in this case files cannot be NULL) or object of class SpatialPointsDataFrame-class with two data columns: 'dir' and 'filename' (and possibly 'format').
files	A vector of strings to be read (when points is of class SpatialPoints-class). Length and order must match points.
stations	An integer vector or string vector identifying point indices or station names in the netCDF.
varmapping	Named character vector specifying a mapping of variables in the NetCDF into variables used in meteoland (e.g. c(MinTemperature = "tmn") specifies a map of variable 'tmn' to MinTemperature).
verbose	A logical flag to output process information in the console.

Details

Function readmeteorologypoint reads data series of a single location from an ascii or rds file and returns a data frame. Function readmeteorologypointfiles can be used to read multiple ascii/rds files and build an object of SpatialPointsMeteorology-class. This is done by supplying an points object of class SpatialPointsDataFrame-class with point meta data. In readmeteorologypointfiles the value of format is used as default but can be overloaded if points includes a column 'format'.

Function readmeteorologypoints is used to read multiple point data from a netCDF. In this case, a mapping can be supplied to map variable names in the netCDF to variables used in meteoland.

Functions

- readmeteorologypointfiles(): **[Deprecated]**
- readmeteorologypoints(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

Nicolas Martin, INRA-Avignon

See Also

writemeteorologypoint, read.table, SpatialPointsMeteorology-class

---

readNetCDFpoints *Utility functions for NetCDFs*

Description

[Deprecated]

Functions to read spatial and temporal coordinates from NetCDFs.

Usage

```
readNetCDFpoints(file)

readNetCDFdates(file)

readNetCDFgridtopology(file)

readNetCDFproj4string(file)
```

Arguments

file	String of the NetCDF whose spatial/temporal coordinates are desired.

Value

- Function readNetCDFdates returns a Date vector.
- Function readNetCDFpoints returns an object SpatialPoints-class.
- Function readNetCDFgridtopology returns an object GridTopology-class.
- Function readNetCDFproj4string returns an object CRS-class.

Functions

- readNetCDFdates(): **[Deprecated]**
- readNetCDFgridtopology(): **[Deprecated]**
- readNetCDFproj4string(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

readmeteorologypoints, readmeteorologygrid

`readWindNinjaWindFields` *Reads WindNinja results*

Description

[Deprecated]

Reads the wind fields generated by 'WindNinja' (http://www.firelab.org/project/windninja) for combinations of domain-level wind speed and wind direction classes.

Usage

```
readWindNinjaWindFields(
  filebase,
  resolution = "100m",
  directionClasses = c(0, 45, 90, 135, 180, 225, 270, 315),
  speedClasses = c(5, 15, 25),
  proj4string = CRS(as.character(NA))
)
```

Arguments

`filebase`	A string to indicate the template for accessing WindNinja files. Resolution, wind directions and wind speed class values are appended to this string to obtain the filename to read.
`resolution`	Resolution string.
`directionClasses`	A vector of wind speed directions (in degrees).
`speedClasses`	A vector of wind class values (in m/s).
`proj4string`	Object of class `"CRS"` with the projection string of wind field rasters.

Value

A list with the following items:

- `directionClasses`: The vector of wind direction classes.
- `speedClasses`: The vector of wind speed classes.
- `indexTable`: A numeric matrix indicating the raster index of each combination of domain-level wind directions and wind speed classes.
- `windSpeed`: An object of class SpatialGridDataFrame containing wind speed rasters (in m/s) for each combination of domain-level wind direction and wind speed.
- `windDirection`: An object of class SpatialGridDataFrame containing wind direction rasters (in degrees from North) for each combination of domain-level wind direction and wind speed.

Note

WindNinja should be run with m/s as wind speed units and for all the combinations of domain-level wind speed and wind direction required.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Forthofer, J.M., Butler, B.W., Wagenbrenner, N.S., 2014. A comparison of three approaches for simulating fine-scale surface winds in support of wildland fire management. Part I. Model formulation and comparison against measurements. Int. J. Wildl. Fire 23, 969–981.

See Also

`MeteorologyInterpolationData`

`read_interpolator` *Read interpolator files*

Description

Read interpolator files created with `write_interpolator`

Usage

```
read_interpolator(filename)
```

Arguments

`filename`	interpolator file name

Details

This function takes the file name of the nc file storing an interpolator object and load it into the work environment

Value

an interpolator (stars) object

See Also

Other interpolator functions: `add_topo()`, `create_meteo_interpolator()`, `get_interpolation_params()`, `set_interpolation_params()`, `with_meteo()`, `write_interpolator()`

Examples

```
# example interpolator
data(meteoland_interpolator_example)

# temporal folder
tmp_dir <- tempdir()

# write interpolator
write_interpolator(
  meteoland_interpolator_example,
  file.path(tmp_dir, "meteoland_interpolator_example.nc")
)

# check file exists
file.exists(file.path(tmp_dir, "meteoland_interpolator_example.nc"))

# read it again
read_interpolator(file.path(tmp_dir, "meteoland_interpolator_example.nc"))
```

---

reshapemeteospain *Reshapes weather data from 'meteospain', 'worldmet' or 'weathercan'*

Description

[Deprecated]

Reshapes weather station data acquired using the 'meteospain', 'worldmet' or 'weathercan' R packages into formats useful for meteoland

Usage

```
reshapemeteospain(
  weather_data,
  output = "SpatialPointsMeteorology",
  proj4string = NULL,
  complete = TRUE,
  verbose = TRUE
)

reshapeweathercan(
  hourly_data,
  daily_data = NULL,
  output = "SpatialPointsMeteorology",
  proj4string = NULL,
```

```
  complete = TRUE,
  verbose = TRUE
)

reshapeworldmet(
  hourly_data,
  output = "SpatialPointsMeteorology",
  proj4string = NULL,
  complete = TRUE,
  verbose = TRUE
)
```

Arguments

weather_data	Hourly or daily weather data, in form of an sf (spatial) object, obtained using function 'get_meteo_from()' in package 'meteospain'.
output	Kind of output desired. Either 'SpatialPointsTopography', 'SpatialPointsMeteorology' or 'MeteorologyInterpolationData'.
proj4string	A string or CRS to change the coordinate reference system of the output. If NULL the spatial reference will be geographic coordinates (i.e. CRS("+proj=longlat")). When reshaping to MeteorologyInterpolationData it is recommended to use a reference system with meters in units, such as UTM.
complete	A flag to indicate that missing variables should be completed using function meteocomplete
verbose	A flag to show information of the reshape process in the console output.
hourly_data	Hourly weather data. In the case of reshapeworldmet, a tibble or data frame returned by function 'importNOAA'. In the case of reshapeweathercan, a tibble or data frame returned by function 'weather_dl' with 'interval="hour"'.
daily_data	Daily weather data (only for reshapeweathercan), a tibble or data frame returned by function 'weather_dl' with 'interval="day"'.

Details

Note that to have precipitation included in downloads from 'worldmet' you should set 'precip = TRUE' when calling function 'importNOAA'. In the case of weathercan, precipitation is only provided for daily data (i.e. setting 'interval="day"' when calling 'weather_dl'), whereas wind speed and relative humidity are only available for hourly data (i.e., setting 'interval="hour"' when calling 'weather_dl'). Hence, in meteoland we recommend downloading both daily and hourly data and then calling function reshapeweathercan to merge the two sources.

Value

An object of the class indicated in output.

Functions

- reshapeweathercan(): **[Deprecated]**
- reshapeworldmet(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

meteocomplete

`set_interpolation_params` *Setting interpolation parameters in an interpolator object*

Description

Changing or updating interpolation parameters in an interpolator object

Usage

```
set_interpolation_params(
  interpolator,
  params = NULL,
  verbose = getOption("meteoland_verbosity", TRUE)
)
```

Arguments

`interpolator`	interpolator object to update
`params`	list with the parameters provided by the user
`verbose`	Logical indicating if the function must show messages and info. Default value checks `"meteoland_verbosity"` option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with `options(meteoland_verbosity = FALSE)`

Details

This function ensures that if no parameters are provided, the default ones are used (see `defaultInterpolationParams`). Also, if params are partially provided, this function ensures that the rest of the parameters are not changed.

Value

The same interpolator object provided, with the updated interpolation parameters

See Also

Other interpolator functions: `add_topo()`, `create_meteo_interpolator()`, `get_interpolation_params()`, `read_interpolator()`, `with_meteo()`, `write_interpolator()`

Examples

```
# example interpolator
data(meteoland_interpolator_example)
# store the actual parameters
old_parameters <- get_interpolation_params(meteoland_interpolator_example)
# we can provide only the parameter we want to change
meteoland_interpolator_example <- set_interpolation_params(
  meteoland_interpolator_example,
  list(debug = TRUE)
)
# check
get_interpolation_params(meteoland_interpolator_example)$debug
# compare with old
old_parameters$debug
# the rest should be the same
setdiff(old_parameters, get_interpolation_params(meteoland_interpolator_example))
```

SpatialGridMeteorology

Creates a 'SpatialGridMeteorology'

Description

[Deprecated]

Initializes an object of class `SpatialGridMeteorology-class`

Usage

```
SpatialGridMeteorology(grid, proj4string = CRS(as.character(NA)), data, dates)
```

Arguments

`grid`	An object of class `GridTopology-class`
`proj4string`	Object of class `"CRS"` with the projection string.
`data`	A vector of data frames (one per date).
`dates`	Object of class `"Date"` describing the time period of meteorological estimates.

Value

An object of class `SpatialGridMeteorology-class`

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`SpatialGridMeteorology-class`

SpatialGridMeteorology-class *Class* "SpatialGridMeteorology"

Description

[Deprecated]

An S4 class that represents a spatial grid with meteorology daily data.

Objects from the Class

Objects can be created by calls of the form `new("SpatialGridMeteorology", ...)`, or by calls to the function `SpatialGridMeteorology`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`SpatialGridTopography`, `SpatialGridDataFrame-class`

Examples

```
#Structure of the S4 object
showClass("SpatialGridMeteorology")
```

SpatialGridTopography *Creates a 'SpatialGridTopography'*

Description

[Deprecated]

Function `SpatialGridTopography` creates an object of class `SpatialGridTopography-class` containing topographic variables over a landscape.

Usage

```
SpatialGridTopography(
  grid,
  elevation,
  slope = NULL,
  aspect = NULL,
  proj4string = CRS(as.character(NA))
)
```

Arguments

grid	An object of class GridTopology-class or SpatialGrid-class.
elevation	A vector of elevation values for all cells of the grid (in m.a.s.l.).
slope	A vector of slope angles for all cells of the grid (in degrees). If slope=NULL, slope is calculated as indicated in details.
aspect	A vector of aspect angles for all cells of the grid (in degrees from North clockwise). aspect=NULL, aspect values are calculated as indicated in details.
proj4string	An object of class CRS-class.

Details

Slope and aspect calculations were adapted from functions in package 'SDMTools', which used the approach described in Burrough & McDonell (1998).

The rate of change (delta) of the surface in the horizontal (dz/dx) and vertical (dz/dy) directions from the center cell determines the slope and aspect. The values of the center cell and its eight neighbors determine the horizontal and vertical deltas. The neighbors are identified as letters from 'a' to 'i', with 'e' representing the cell for which the aspect is being calculated. The rate of change in the x direction for cell 'e' is calculated with the algorithm:

```
[dz/dx] = ((c + 2f + i) - (a + 2d + g) / (8 * x_cell_size)
```

The rate of change in the y direction for cell 'e' is calculated with the following algorithm:

```
[dz/dy] = ((g + 2h + i) - (a + 2b + c)) / (8 * y_cell_size)
```

The algorithm calculates slope as: `rise_run = sqrt ( [dz/dx]2 + [dz/dy]2 ])`.

From this value , one can calculate the slope in degrees or radians as:

```
slope_degrees = ATAN (rise_run) * 57.29578
```

```
slope_radians = ATAN (rise_run)
```

Taking the rate of change in both the x and y direction for cell 'e', aspect is calculated using:

```
aspect = 57.29578 * atan2 ([dz/dy], -[dz/dx])
```

The aspect value is then converted to compass direction values (0-360 degrees).

Value

Function SpatialGridTopography returns an object 'SpatialGridTopography-class'.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Burrough, P. A. and McDonell, R.A., 1998. Principles of Geographical Information Systems (Oxford University Press, New York), p. 190.

See Also

SpatialGridTopography-class

Examples

```
data(examplegridtopography)

#Display data
spplot(examplegridtopography, variable="elevation", scales=list(draw=TRUE))

#Grids can be subsetted
sgt = examplegridtopography[1:50, 1:50]
spplot(sgt, variable="elevation", scales=list(draw=TRUE))
```

SpatialGridTopography-class

Class "SpatialGridTopography"

Description

[Deprecated]

An S4 class that represents topography over a grid of coordinates.

Objects from the Class

Objects can be created by calls of the form `new("SpatialGridTopography", ...)`, or by calls to the function SpatialGridTopography.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

SpatialGridTopography, SpatialGridDataFrame-class

Examples

```
#Structure of the S4 object
showClass("SpatialGridTopography")
```

SpatialPixelsMeteorology *Creates a 'SpatialPixelsMeteorology'*

Description

[Deprecated]

Initializes an object of class SpatialPixelsMeteorology-class

Usage

```
SpatialPixelsMeteorology(
  points,
  data,
  dates,
  tolerance = sqrt(.Machine$double.eps),
  proj4string = CRS(as.character(NA)),
  round = NULL,
  grid = NULL
)
```

Arguments

points	An object of class SpatialPoints-class.
data	A vector of data frames (one per date).
dates	Object of class "Date" describing the time period of meteorological estimates.
tolerance	Precision up to which extent points should be exactly on a grid.
proj4string	Object of class CRS in the first form only used when points does not inherit from Spatial.
round	default NULL, otherwise a value passed to as the digits argument to round for setting cell size.
grid	Grid topology using an object of class GridTopology; a value of NULL implies that this will be derived from the point coordinates.

Value

An object of class SpatialPixelsMeteorology-class

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

SpatialPixelsMeteorology-class

SpatialPixelsMeteorology-class

Class "SpatialPixelsMeteorology"

Description

[Deprecated]

An S4 class that represents meteorology data that has locations on a regular grid.

Objects from the Class

Objects can be created by calls of the form `new("SpatialPixelsMeteorology", ...)`, or by calls to the function `SpatialPixelsMeteorology`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`SpatialPixelsTopography`, `SpatialPixelsDataFrame-class`

Examples

```
#Structure of the S4 object
showClass("SpatialPixelsMeteorology")
```

SpatialPixelsTopography

Creates a 'SpatialPixelsTopography'

Description

[Deprecated]

Function `SpatialPixelsTopography` creates an object of class `SpatialPixelsTopography-class` containing topographic variables for a set of points.

Usage

```
SpatialPixelsTopography(
  points,
  elevation,
  slope,
  aspect,
  tolerance = sqrt(.Machine$double.eps),
  proj4string = CRS(as.character(NA)),
  round = NULL,
  grid = NULL
)
```

Arguments

points	An object of class SpatialPoints-class or a numeric matrix of coordinates.
elevation	Elevation values (in m) of the points.
slope	Slope values (in degrees) of the points.
aspect	Aspect values (in degrees from North) of the points.
tolerance	Precision up to which extent points should be exactly on a grid.
proj4string	Object of class CRS in the first form only used when points does not inherit from Spatial.
round	default NULL, otherwise a value passed to as the digits argument to round for setting cell size.
grid	Grid topology using an object of class GridTopology; a value of NULL implies that this will be derived from the point coordinates.

Value

Function SpatialPixelsTopography returns an object 'SpatialPixelsTopography-class'.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

SpatialPixelsTopography-class

Examples

```
data(examplegridtopography)

#Creates spatial topography pixels as a subset of points in the grid
spt = as(examplegridtopography,"SpatialPointsTopography")
cc = spt@coords
center = 5160
```

```
d = sqrt((cc[,1]-cc[center,1])^2+(cc[,2]-cc[center,2])^2)
p = which(d<3000) #Select points at maximum distance of 3km from center
spxt = SpatialPixelsTopography(spt[p], spt$elevation[p],
                               spt$slope[p],
                               spt$aspect[p])

#Alternatively, use coercing and subsetting (drop = TRUE causes grid to be recalculated)
spxt = as(examplegridtopography, "SpatialPixelsTopography")[p, drop=TRUE]

#Display data
spplot(spxt, variable="elevation", scales=list(draw=TRUE))
spplot(spxt, variable="slope", scales=list(draw=TRUE))
spplot(spxt, variable="aspect", scales=list(draw=TRUE))
```

SpatialPixelsTopography-class
Class "SpatialPixelsTopography"

Description

[Deprecated]

An S4 class that represents topography that has locations on a regular grid.

Objects from the Class

Objects can be created by calls of the form `new("SpatialPixelsTopography", ...)`, or by calls to the function SpatialPixelsTopography.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

SpatialPixelsTopography, SpatialPixelsDataFrame-class

Examples

```
#Structure of the S4 object
showClass("SpatialPixelsTopography")
```

SpatialPointsMeteorology *Creates a 'SpatialPointsMeteorology'*

Description

[Deprecated]

Initializes an object of class SpatialPointsMeteorology-class

Usage

```
SpatialPointsMeteorology(points, data, dates, dataByDate = FALSE)
```

Arguments

points	An object of class SpatialPoints-class. Row names of point coordinates are used to identify points.
data	A list of data frames. If dataByDate = FALSE the elements of data are assumed to correspond to points. If dataByDate = TRUE the elements of data are assumed to correspond to dates (see 'Details').
dates	Object of class "Date" describing the time period of meteorological estimates.
dataByDate	A flag to indicate that elements of data correspond to dates, as opposed to the default (dataByDate = FALSE) which assumes that elements correspond to points (see 'Details').

Details

There are two ways of building an object of of class SpatialPointsMeteorology-class. The first way (dataByDate = FALSE) is to supply as value for data a vector of data frames with one data frame per spatial point, with dates as rows and meteorological variables as columns. In this case all data frames must have the same number of rows (dates) and columns (variables). The second way (if dataByDate = TRUE) is to supply as value for data a vector of data frames with one data frame per date, with points as rows and meteorological variables as columns. In this case, the data frames may have different rows and different columns. Only the information corresponding to points will be taken and some variables may be missing.

Value

An object of class SpatialPointsMeteorology-class

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

SpatialPointsMeteorology-class

SpatialPointsMeteorology-class

Class "SpatialPointsMeteorology"

Description

[Deprecated]

An S4 class that represents a set of points with meteorology data series.

Objects from the Class

Objects can be created by calls of the form `new("SpatialPointsMeteorology", ...)`, or by calls to the function `SpatialPointsMeteorology`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`SpatialPointsTopography-class`, `SpatialPoints-class`

Examples

```
#Structure of the S4 object
showClass("SpatialPointsMeteorology")
```

SpatialPointsTopography

Creates a 'SpatialPointsTopography'

Description

[Deprecated]

Function `SpatialPointsTopography` creates an object of class `SpatialPointsTopography-class` containing topographic variables for a set of points.

Usage

```
SpatialPointsTopography(
  points,
  elevation,
  slope = NULL,
  aspect = NULL,
  proj4string = CRS(as.character(NA))
)
```

Arguments

points	An object of class SpatialPoints-class.
elevation	Elevation values (in m) of the points.
slope	Slope values (in degrees) of the points.
aspect	Aspect values (in degrees from North) of the points.
proj4string	Object of class CRS in the first form only used when points does not inherit from Spatial.

Details

If either slope = NULL or aspect = NULL then when estimating weather on the object locations radiation will be calculated assuming a flat surface.

Value

Function SpatialPointsTopography returns an object 'SpatialPointsTopography-class'.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

SpatialPointsTopography-class

Examples

```
data(examplegridtopography)

#Creates spatial topography points from the grid
p = 1:2
points = as(examplegridtopography,"SpatialPoints")[p]
spt = SpatialPointsTopography(points, examplegridtopography$elevation[p],
                              examplegridtopography$slope[p],
                              examplegridtopography$aspect[p])
spt

#Alternatively, use coercing and subsetting
spt = as(examplegridtopography, "SpatialPointsTopography")[p]
spt
```

SpatialPointsTopography-class

Class "SpatialPointsTopography"

Description

[Deprecated]

An S4 class that represents topography over a grid of coordinates.

Objects from the Class

Objects can be created by calls of the form `new("SpatialPointsTopography", ...)`, or by calls to the function `SpatialPointsTopography`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`SpatialPointsTopography`, `SpatialPointsDataFrame-class`

Examples

```
#Structure of the S4 object
showClass("SpatialPointsTopography")
```

spplot,SpatialGridMeteorology-method

Spatial grid plots

Description

[Deprecated]

Function `spplot` for `SpatialGridTopography-class` and `SpatialPixelsTopography-class` objects allows drawing maps of topographic attributes. Function `spplot` for `SpatialGridMeteorology-class` and `SpatialPixelsMeteorology-class` objects allows drawing maps of meteorological variables corresponding to specific dates.

Usage

```
## S4 method for signature 'SpatialGridMeteorology'
spplot(obj, date, variable = "MeanTemperature", ...)
```

Arguments

obj	An object of class SpatialGridTopography.
date	A string or an integer for the date to be plotted.
variable	A string of the variable to be plotted (only type="elevation", type="slope", type="aspect" are allowed).
...	Additional parameters to function spplot.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

meteoplot

Examples

```
data(examplegridtopography)

#Display data
spplot(examplegridtopography, type="elevation", scales=list(draw=TRUE))
spplot(examplegridtopography, type="slope", scales=list(draw=TRUE))
spplot(examplegridtopography, type="aspect", scales=list(draw=TRUE))
```

---

subsample *Sub-sampling procedure data*

Description

[Deprecated]

Generates a spatial and/or temporal subset of procedure data

Usage

```
## S4 method for signature 'MeteorologyInterpolationData'
subsample(object, bbox = NULL, stations = NULL, dates = NULL, buffer = 0)

## S4 method for signature 'MeteorologyUncorrectedData'
subsample(object, bbox = NULL, stations = NULL, dates = NULL, buffer = 0)
```

Arguments

object	An object of a sub-class MeteorologyProcedureData-class.
bbox	A 2x2 numeric matrix with the boundaries of the target area. If NULL, the original boundary box is kept (except if stations is specified).
stations	A numeric, character or logical vector specifying a subset of weather stations. If NULL all original weather stations are kept (except if bbox is specified).
dates	A vector of Date with the subset of dates of interest. If NULL, all dates are kept.
buffer	A buffer to put around bbox for the spatial selection of data.

Value

An object of the same class as object.

Functions

- subsample(MeteorologyUncorrectedData): Signature for MeteorologyUncorrectedData

Methods

subsample signature(object = "MeteorologyUncorrectedData"): Generates a MeteorologyUncorrectedData object for a smaller area and a subset of dates.

subsample signature(object = "MeteorologyInterpolationData"): Generates a MeteorologyInterpolationData object for a smaller area and a subset of dates.

Examples

```
data(exampleinterpolationdata)

oridates = exampleinterpolationdata@dates

#Interpolation data using the first ten dates (same boundary box)
subdata = subsample(exampleinterpolationdata, dates = oridates[1:10])
```

---

summarise_interpolated_data

Summarise interpolated data by temporal dimension

Description

[Experimental]

Summarises the interpolated meteorology in one or more locations by the desired temporal scale

Usage

```
summarise_interpolated_data(
  interpolated_data,
  fun = "mean",
  frequency = NULL,
  vars_to_summary = c("MeanTemperature", "MinTemperature", "MaxTemperature",
    "Precipitation", "MeanRelativeHumidity", "MinRelativeHumidity",
    "MaxRelativeHumidity", "Radiation", "WindSpeed", "WindDirection", "PET"),
  dates_to_summary = NULL,
  months_to_summary = 1:12,
  verbose = getOption("meteoland_verbosity", TRUE),
  ...
)
```

Arguments

`interpolated_data`	An interpolated data object as returned by `interpolate_data`.
`fun`	The function to use for summarising the data.
`frequency`	A string indicating the interval specification (allowed ones are "week", "month", "quarter" and "year"). If NULL (default), aggregation is done in one interval for all the dates present.
`vars_to_summary`	A character vector with one or more variable names to summarise. By default, all interpolated variables are summarised.
`dates_to_summary`	A Date object to define the dates to be summarised. If NULL (default), all dates in the interpolated data are processed.
`months_to_summary`	A numeric vector with the month numbers to subset the data before summarising. (e.g. `c(7,8)` for July and August). This parameter allows studying particular seasons, when combined with `frequency`. For example `frequency = "years"` and `months = 6:8` leads to summarizing summer months of each year.
`verbose`	Logical indicating if the function must show messages and info. Default value checks `"meteoland_verbosity"` option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with `options(meteoland_verbosity = FALSE)`
`...`	Arguments needed for `fun`

Details

If `interpolated_data` is a nested interpolated data sf object, as returned by `interpolate_data`, temporal summary is done for each location present in the interpolated data. If `interpolated_data` is an unnested interpolated data sf object, temporal summary is done for all locations together. If `interpolated_data` is a single location data.frame containing the dates and the interpolated variables, temporal summary is done for that location. If `interpolated_data` is a stars object as returned by `interpolate_data`, temporal summary is done for all the raster.

Value

For a nested interpolated data, the same sf object with a new column with the temporal summaries. For an unnested interpolated data, a data.frame with the summarised meteo variables. For an interpolated raster (stars object), the same raster with the temporal dimension aggregated as desired.

Author(s)

Víctor Granda García, CREAF

Examples

```
# points interpolation aggregation
points_to_interpolate_example |>
  interpolate_data(meteoland_interpolator_example, verbose = FALSE) |>
  summarise_interpolated_data()

# raster interpolation aggregation
raster_to_interpolate_example |>
  interpolate_data(meteoland_interpolator_example, verbose = FALSE) |>
  summarise_interpolated_data()
```

---

summarise_interpolator

Summarise interpolator objects by temporal dimension

Description

[Experimental]

Summarises an interpolator object by the desired temporal scale.

Usage

```
summarise_interpolator(
  interpolator,
  fun = "mean",
  frequency = NULL,
  vars_to_summary = c("Temperature", "MinTemperature", "MaxTemperature", "Precipitation",
   "RelativeHumidity", "MinRelativeHumidity", "MaxRelativeHumidity", "Radiation",
    "WindSpeed", "WindDirection", "PET", "SmoothedPrecipitation",
    "SmoothedTemperatureRange", "elevation", "slope", "aspect"),
  dates_to_summary = NULL,
  months_to_summary = 1:12,
  verbose = getOption("meteoland_verbosity", TRUE),
  ...
)
```

Arguments

interpolator	An interpolator object as created by create_meteo_interpolator.
fun	The function to use for summarising the data.
frequency	A string indicating the interval specification (allowed ones are "week", "month", "quarter" and "year"). If NULL (default), aggregation is done in one interval for all the dates present.
vars_to_summary	A character vector with one or more variable names to summarise. By default, all interpolated variables are summarised.
dates_to_summary	A Date object to define the dates to be summarised. If NULL (default), all dates in the interpolated data are processed.
months_to_summary	A numeric vector with the month numbers to subset the data before summarising. (e.g. c(7,8) for July and August). This parameter allows studying particular seasons, when combined with frequency. For example frequency = "years" and months = 6:8 leads to summarizing summer months of each year.
verbose	Logical indicating if the function must show messages and info. Default value checks "meteoland_verbosity" option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with options(meteoland_verbosity = FALSE)
...	Arguments needed for fun

Value

summarise_interpolator function returns the same interpolator object provided with the temporal dimension aggregated to desired frequency.

Author(s)

Víctor Granda García, CREAF

Examples

```
# example interpolator
meteoland_interpolator_example

# aggregate all dates in the interpolator, calculating the maximum values
summarise_interpolator(meteoland_interpolator_example, fun = "max")

# aggregate weekly, calculating mean values
summarise_interpolator(meteoland_interpolator_example, frequency = "week")
```

summarypixels *Summaries of meteorological data*

Description

[Deprecated]

Summarizes the meteorology of a single location, a set of spatial points, pixels in a grid, or weather stations of interpolation data.

Usage

```
summarypixels(
  pixels,
  var,
  fun = mean,
  freq = NULL,
  dates = NULL,
  months = NULL,
  ...
)

summarygrid(
  grid,
  var,
  fun = mean,
  freq = NULL,
  dates = NULL,
  months = NULL,
  ...
)

summaryinterpolationdata(
  object,
  var,
  fun = mean,
  freq = NULL,
  dates = NULL,
  months = NULL,
  ...
)

summarypoint(
  x,
  var,
  fun = "mean",
  freq = NULL,
```

```
  dates = NULL,
  months = NULL,
  ...
)

summarypoints(
  points,
  var,
  fun = mean,
  freq = NULL,
  dates = NULL,
  months = NULL,
  ...
)
```

Arguments

pixels	An object of class SpatialPixelsMeteorology-class with the meteorological data for grid pixels, or a string pointing to a NetCDF.
var	The name of the meteorological variable to be summarized.
fun	The function to be calculated on values of each point. If freq is specified, the function will be calculated by intervals.
freq	A string giving an interval specification (e.g., "week", "month", "quarter" or "year"). If NULL then no intervals are defined.
dates	An object of class Date to define the period to be summarized. If dates = NULL then all dates in points are processed.
months	A numeric vector to indicate the subset of months for which summary is desired (e.g. c(7,8) for July and August). This parameter allows studying particular seasons, when combined with freq. For example freq = "years" and months = 6:8 leads to summarizing summer months of each year.
...	Additional parameters to fun.
grid	An object of class SpatialGridMeteorology-class with the meteorological data for a grid, or a string pointing to a NetCDF.
object	An object of class MeteorologyInterpolationData-class.
x	A data frame with dates in rows and meteorological variables in columns.
points	An object of class SpatialPointsMeteorology-class with the coordinates and meteorological data of the locations for which summaries are desired. Alternatively, an object of class SpatialPointsDataFrame-class containing the meta data (columns dir, filename and possibly format) of meteorological files that will be sequentially read from the disk. Finally, points can also be a string pointing to a netCDF.

Details

If var="ALL" then function summarypoints produces a summary of all variables with default statistics and returns an object of class SpatialPointsMeteorology.

Value

- Function summarypoint returns a named vector of values with dates as names.
- Functions summarypoints and summaryinterpolationdata return an object of class SpatialPointsDataFrame containing summaries (either one variable or several if freq is specified).
- Functions summarygrid and summarypixels return an object of class SpatialGridDataFrame and SpatialPixelsDataFrame, respectively, containing the summaries analogously to summarypoints.

Functions

- summarypixels(): **[Deprecated]**
- summarygrid(): **[Deprecated]**
- summaryinterpolationdata(): **[Deprecated]**
- summarypoint(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

Antoine Cabon, CTFC

See Also

SpatialPointsMeteorology-class

Examples

```
data(examplegridtopography)
data(exampleinterpolationdata)

#Creates spatial topography points from the grid
p = 1:2
spt = as(examplegridtopography, "SpatialPointsTopography")[p]

#Interpolation of two points for the whole time period (2000-2003)
mp = interpolationpoints(exampleinterpolationdata, spt)

#PET sums by months
mp.sum = summarypoints(mp, var="PET", freq="months", fun=sum)

mp.sum
```

`utils_saturationVP` *Physical utility functions*

Description

Set of functions used in the calculation of physical variables.

Usage

```
utils_saturationVP(temperature)

utils_averageDailyVP(Tmin, Tmax, RHmin, RHmax)

utils_atmosphericPressure(elevation)

utils_airDensity(temperature, Patm)

utils_averageDaylightTemperature(Tmin, Tmax)

utils_latentHeatVaporisation(temperature)

utils_latentHeatVaporisationMol(temperature)

utils_psychrometricConstant(temperature, Patm)

utils_saturationVaporPressureCurveSlope(temperature)
```

Arguments

`temperature`	Air temperature (ºC).
`Tmin, Tmax`	Minimum and maximum daily temperature (ºC).
`RHmin, RHmax`	Minimum and maximum relative humidity (%).
`elevation`	Elevation above sea level (in m).
`Patm`	Atmospheric air pressure (in kPa).

Value

Values returned for each function are:

- `utils_airDensity`: air density (in kg·m-3).
- `utils_atmosphericPressure`: Air atmospheric pressure (in kPa).
- `utils_averageDailyVP`: average (actual) vapour pressure (in kPa).
- `utils_averageDaylightTemperature`: average daylight air temperature (in ºC). `utils_latentHeatVaporisation`: Latent heat of vaporisation (MJ·kg-1). `utils_latentHeatVaporisationMol`: Latent heat of vaporisation (J·mol-1).

- `utils_psychrometricConstant`: Psychrometric constant (kPa·ºC-1).
- `utils_saturationVP`: saturation vapour pressure (in kPa).
- `utils_saturationVaporPressureCurveSlope`: Slope of the saturation vapor pressure curve (kPa·ºC-1).

Functions

- `utils_averageDailyVP()`: Average daily VP
- `utils_atmosphericPressure()`: Atmospheric pressure
- `utils_airDensity()`: Air density
- `utils_averageDaylightTemperature()`: Daylight temperature
- `utils_latentHeatVaporisation()`: latent heat vaporisation
- `utils_latentHeatVaporisationMol()`: Heat vaporisation mol
- `utils_psychrometricConstant()`: psychrometric constant
- `utils_saturationVaporPressureCurveSlope()`: Saturation VP curve slope

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

McMurtrie, R. E., D. A. Rook, and F. M. Kelliher. 1990. Modelling the yield of Pinus radiata on a site limited by water and nitrogen. Forest Ecology and Management 30:381–413.

McMahon, T. A., M. C. Peel, L. Lowe, R. Srikanthan, and T. R. McVicar. 2013. Estimating actual, potential, reference crop and pan evaporation using standard meteorological data: a pragmatic synthesis. Hydrology & Earth System Sciences 17:1331–1363. See also: http://www.fao.org/docrep/x0490e/x0490e06.htm

---

`weathergeneration` *Weather generation*

Description

[Deprecated]

A semiparametric multivariate, multisite weather generator. The algorithm can be interpreted as a way to resample the original data to create synthetic data sets of the same length and similar properties.

Usage

```
weathergeneration(object, params = defaultGenerationParams(), verbose = TRUE)
```

Arguments

object	An object of class SpatialPointsMeteorology-class, SpatialGridMeteorology-class or SpatialPixelsMeteorology-class.
params	A list with parameters for the weather generator (see defaultGenerationParams).
verbose	Boolean flag to print process information.

Details

The implemented algorithm is based on Apipattanavis et al. (2007) for the non-conditional version, and is similar to Steinschneider et al. (2013) for the conditional one. Part of the code was adapted from package 'weathergen' by Jeffrey D. Walker, whom we are grateful. Conditioning is controlled via the element conditional of params list, which can be:

- "none" - The non-conditional version is used, which is based on a first order Markov chain (MC) to simulate weather states (dry/wet/extreme wet) and a K-nearest neighbour (KNN) algorithm to select pairs of days with the same transition and similar weather for the initial state (as in Apipattanavis et al. 2007).
- "arima" - Annual precipitation is conditioned using a stationary auto-regressive (ARIMA) model and then a K-nearest neighbour algorithm is used to select a set of years to train the MC-KNN algorithm (similar to Steinschneider et al. 2013). Recommended if low-frequency variation of annual precipitation is to be accounted for in long series.
- "window" - The MC-KNN algorithm is trained with the subset of the input data corresponding to a window around the target year. Annual precipitation is conditioned using a lognormal random trial of the precipitation corresponding to the selected years. Recommended to generate stochastic series from climate change projections.

Value

An object of the same class as the input object. Generated meteorological series are of the same length as the input.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Apipattanavis, S., G. Podesta, B. Rajagopalan, and R. W. Katz (2007), A semiparametric multivariate and multisite weather generator, Water Resour. Res., 43, W11401, doi:10.1029/2006WR005714.

Steinschneider S., and Brown C. (2013) A semiparametric multivariate, multisite weather generator with low-frequency variability for use in climate risk assessments. Water Resour. Res., 49, 7205-7220, doi:10.1002/wrcr.20528.

See Also

defaultGenerationParams

Examples

```
data(examplegridtopography)
data(exampleinterpolationdata)

#Creates spatial topography points from the grid
p = 1:2
spt = as(examplegridtopography, "SpatialPointsTopography")[p]

#Interpolation of two points for the whole time period (2000-2003)
mp = interpolationpoints(exampleinterpolationdata, spt)

#Plot interpolated meteorological series
meteoplot(mp,1, ylab="Mean temperature")

#Generate a stochastic series using interpolated data as template
y = weathergeneration(mp)

#Plot generated meteorological series
meteoplot(y,1, ylab="Mean temperature")
```

---

with_meteo *Ensure meteo object is ready to create an interpolator object*

Description

Check integrity of meteo objects

Usage

```
with_meteo(meteo, verbose = getOption("meteoland_verbosity", TRUE))
```

Arguments

meteo	meteo object
verbose	Logical indicating if the function must show messages and info. Default value checks "meteoland_verbosity" option and if not set, defaults to TRUE. It can be turned off for the function with FALSE, or session wide with options(meteoland_verbosity = FALSE)

Details

This function is the first step in the creation of a meteoland interpolator, ensuring the meteo provided contains all the required elements

Value

invisible meteo object ready to pipe in the interpolator creation

See Also

Other interpolator functions: `add_topo()`, `create_meteo_interpolator()`, `get_interpolation_params()`, `read_interpolator()`, `set_interpolation_params()`, `write_interpolator()`

Examples

```
# example meteo
data(meteoland_meteo_example)
with_meteo(meteoland_meteo_example)
```

---

worldmet2meteoland *From worldmet to meteoland meteo objects*

Description

Adapting `importNOAA` meteo objects to meteoland meteo objects

Usage

```
worldmet2meteoland(meteo, complete = FALSE)
```

Arguments

`meteo`	worldmet meteo object.
`complete`	logical indicating if the meteo data missing variables should be calculated (if possible). Default to FALSE.

Details

This function converts `importNOAA` meteo objects to compatible meteoland meteo objects by selecting the needed variables and adapting the names to comply with meteoland requirements. Also it aggregates subdaily data as well as complete missing variables if possible (setting `complete = TRUE`)

Value

a compatible meteo object to use with meteoland.

Examples

```
if (interactive()) {
  # worldmet data
  library(worldmet)
  worldmet_stations <- worldmet::getMeta(lat = 42, lon = 0, n = 2, plot = FALSE)
  worldmet_subdaily_2022 <-
    worldmet::importNOAA(worldmet_stations$code, year = 2022, hourly = TRUE)

  # just convert
  worldmet2meteoland(worldmet_subdaily_2022)
  # convert and complete
  worldmet2meteoland(worldmet_subdaily_2022, complete = TRUE)

}
```

---

writemeteorologygrid *Writes grid meteorology to the disk*

Description

[Deprecated]

Functions to write grid meteorological data to the file system.

Usage

```
writemeteorologygrid(
  object,
  file,
  dates = NULL,
  format = "netCDF",
  byPixel = FALSE,
  chunksizes = NA,
  add = FALSE,
  overwrite = FALSE,
  verbose = FALSE
)

writemeteorologypixels(
  object,
  file,
  dates = NULL,
  format = "netCDF",
  byPixel = FALSE,
  chunksizes = NA,
  add = FALSE,
```

```
  overwrite = FALSE,
  verbose = FALSE
)

writeemptymeteorologygrid(
  file,
  grid,
  proj4string,
  dates,
  byPixel = FALSE,
  chunksizes = NA,
  overwrite = FALSE,
  verbose = FALSE
)

writemeteorologygridpixel(file, index, data, verbose = FALSE)
```

Arguments

object	An object of class SpatialGridMeteorology-class or class SpatialPixelsMeteorology-class with the meteorological data to be written.
file	A string with the file name to be written.
dates	A Date vector object or a character string indicating the dates to be written.
format	Format of meteorological data. The only accepted format is "netCDF".
byPixel	Boolean flag to specify whether file will be written/read by pixels. This forces a different (supposedly) more efficient chunking based on time series data instead of daily grids.
chunksizes	Specifies the size of data chunks to be read/written. If set, this must be a vector of three integers corresponding to XYT.
add	Boolean flag to indicate that NetCDF exists and data should be added/replaced.
overwrite	Boolean flag to force overwriting an existing NetCDF.
verbose	A logical flag to output process information in the console.
grid	An object of class GridTopology-class
proj4string	Object of class CRS.
index	Integer indicating the grid index position to be written.
data	A data frame with meteorological data corresponding to a pixel.

Details

Functions writemeteorologygrid and writemeteorologypixels writes gridded meteorological data (i.e. class SpatialGridMeteorology-class or class SpatialPixelsMeteorology-class, respectively) into a netCDF, with the possibility to overwrite existing data. Function writemeteorologygridpixel is meant to add/replace data in a netCDF corresponding to a specific pixel identified by its grid index. Function writemeteorologygrid creates an empty netCDF with the specified grid dimensions, coordinate reference system and dates.

Functions

- `writemeteorologypixels()`: **[Deprecated]**
- `writeemptymeteorologygrid()`: **[Deprecated]**
- `writemeteorologygridpixel()`: **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`readmeteorologygrid`, `SpatialGridMeteorology-class`

`writemeteorologypoint` *Writes point meteorology to the disk*

Description

[Deprecated]

Functions to write point meteorological data to disk files in different formats.

Usage

```
writemeteorologypoint(data, file, format = "meteoland/txt")

writemeteorologypointfiles(
  object,
  dir = getwd(),
  format = "meteoland/txt",
  metadataFile = "MP.txt"
)

writemeteorologypoints(
  object,
  file,
  format = "netCDF",
  add = FALSE,
  overwrite = FALSE,
  verbose = FALSE
)
```

Arguments

data	An data frame with meteorological data.
file	A string with the file name to be written.
format	Output format of meteorological data. Current accepted formats are "meteoland/txt", "meteoland/rds", "castanea/txt" and "castanea/rds".
object	An object of class SpatialPointsMeteorology-class with the meteorological data to be written.
dir	Output directory for meteorology data.
metadataFile	The name of the file that will store the meta data describing all written files.
add	Boolean flag to indicate that NetCDF exists and data should be added/replaced (see details).
overwrite	Boolean flag to force overwriting an existing NetCDF.
verbose	A logical flag to output process information in the console.

Details

Function writemeteorologypoint writes data series of a single location (i.e. a data frame) to ascii or rds files. Function writemeteorologypointfiles takes an object of SpatialPointsMeteorology-class and writes one file for each data point in the same formats as writemeteorologypoint. In addition, it writes a metadata file (see argument metadataFile) with point coordinates, station names and file names. Both functions share the same accepted formats, which are "meteoland/txt", "meteoland/rds", "castanea/txt" and "castanea/rds".

Function writemeteorologypoints takes an object of SpatialPointsMeteorology-class and writes all its content in a single file (i.e. a netCDF). The same function can be used to replace data from an existing file or to add new points to the netCDF. This is done by using add=TRUE, and profits from the fact that some netCDF dimensions (in this case the number of points) can be defined as unlimited. If data is added to an existing netCDF, the coordinate reference system and the dates of object must be the same as those in the netCDF.

Functions

- writemeteorologypointfiles(): **[Deprecated]**
- writemeteorologypoints(): **[Deprecated]**

Author(s)

Miquel De Cáceres Ainsa, CREAF

Nicolas Martin, INRA-Avignon

See Also

readmeteorologypoint, SpatialPointsMeteorology-class

`write_interpolator` *Write the interpolator object*

Description

Write the interpolator object to a file

Usage

```
write_interpolator(interpolator, filename, .overwrite = FALSE)
```

Arguments

`interpolator`	meteoland interpolator object, as created by `create_meteo_interpolator`
`filename`	file name for the interpolator nc file
`.overwrite`	logical indicating if the file should be overwritten if it already exists

Details

This function writes the interpolator object created with `create_meteo_interpolator` in a NetCDF-CF standard compliant format, as specified in https://cfconventions.org/cf-conventions/cf-conventions.html

Value

invisible interpolator object, to allow using this function as a step in a pipe

See Also

Other interpolator functions: `add_topo()`, `create_meteo_interpolator()`, `get_interpolation_params()`, `read_interpolator()`, `set_interpolation_params()`, `with_meteo()`

Examples

```
# example interpolator
data(meteoland_interpolator_example)

# temporal folder
tmp_dir <- tempdir()

# write interpolator
write_interpolator(
  meteoland_interpolator_example,
  file.path(tmp_dir, "meteoland_interpolator_example.nc"),
  .overwrite = TRUE
)
```

```
# check file exists
file.exists(file.path(tmp_dir, "meteoland_interpolator_example.nc"))

# read it again
read_interpolator(file.path(tmp_dir, "meteoland_interpolator_example.nc"))
```

Index