Package 'simulariatools'

September 1, 2025

Type Package

Title Simularia Tools for the Analysis of Air Pollution Data

Version 3.0.0

Maintainer Giuseppe Carlino <g.carlino@simularia.it>

Description A set of tools developed at Simularia for Simularia, to help preprocessing and post-processing of meteorological and air quality data.

Depends R (>= 3.3)

Imports ggplot2 (>= 3.3), lubridate, reshape2, reticulate, scales, terra

Suggests magick, testthat (>= 3.0.0), openair, sf

License GPL (>= 2)

URL https://www.simularia.it/simulariatools/, https://github.com/Simularia/simulariatools

BugReports https://github.com/Simularia/simulariatools/issues

LazyLoad yes

LazyData yes

Encoding UTF-8

Language en

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Giuseppe Carlino [aut, cre], Matteo Paolo Costa [ctb], Simularia [cph, fnd]

Repository CRAN

Date/Publication 2025-09-01 17:00:02 UTC

Contents

`contourPlot2` *Contour Plot of pollutant concentration field*

Description

The function `contourPlot2` generates a contour plot of a given quantity, such as the ground concentration of an airborne pollutant or odour, defined on a regular grid.

Usage

```
contourPlot2(
  data,
  x = "x",
  y = "y",
  z = "z",
  domain = NULL,
  background = NULL,
  underlayer = NULL,
  overlayer = NULL,
  legend = NULL,
  levels = NULL,
  size = 0,
  fill = TRUE,
  tile = FALSE,
  transparency = 0.75,
  colors = NULL,
  mask = NULL,
  inverse_mask = FALSE,
  bare = FALSE
)
```

Arguments

data	A dataframe in long format with three columns for Easting, Northing and values to be plotted.
x	charactrer. Name of the column containing Easting (longitude) coordinates (default "x").
y	character. Name of the column containing Northing (latitude) coordinates (default "y").
z	character. Name of the column containing concentration values (default "z").
domain	optional list of six numeric values defining the boundaries of the domain to be plotted (minimum X, maximum X, minimum Y, maximum Y) and the number of ticks on X & Y axis. Example: c(340000, 346000, 4989500, 4995500, 5, 5). If missing, all the full domain of the input data is considered, with 5 ticks.
background	filename. Optional path to a raster file to be plotted as the basemap.
underlayer	optional list of layers to be plotted between base map and contour plot. See Details
overlayer	optional list of layers to be plotted on top of the contour plot. See Details.
legend	character. Optional title of the legend.
levels	numeric vector of levels for contour plot. If not set, automatic pretty levels are computed. If `-Inf` and `Inf` are used as the lowest and highest limits of the array, the lowest and highest bands are unbounded and the legend shows < and >= symbols.
size	numeric. Width of the contour line.
fill	logical. If TRUE the contour plot is filled with colour (default = TRUE).
tile	logical. If TRUE rectangular tiles are plotted (default = FALSE).
transparency	transparency level of the contour plot between 0.0 (fully transparent) and 1.0 (fully opaque). Default = 0.75).
colors	colour palette for contour plot, as an array of colours.
mask	character. Path to *shp* file used as a mask. It must be a closed polygon.
inverse_mask	logical. If `TRUE` areas on mask are masked. Default is to mask areas outside the polygon defined in the *shp* file.
bare	boolean (default FALSE). If TRUE only the bare plot is shown: axis, legend, titles and any other graphical element of the plot are removed.

Details

This is a convenience function to plot contour levels of a scalar quantity such as pollutants computed by a dispersion model, with `ggplot2` version >= 3.3.0.

Data are required to be on a regular grid, typically (but not necessarily) in UTM coordinates. The input dataframe has to be in long format, i.e. one line per value to be plotted. The names of the columns corresponding to `x`, `y` and `z` can be specified in the input parameters.

If `tile = TRUE` data are shown as they are, without any graphical interpolation required for contour plots. This is helpful when you want to visualise the raw data. Since version 2.4.0, when `tile =`

TRUE the intervals include the lowest bound and exclude the highest bound: [min, max). Note: In previous version it was the opposite.

underlayer and overlayer layers are ggplot2 objects to be shown at different levels of the vertical stack of the plot. These are useful to show topographical information related to the plot, such as sources or receptors locations.

When a *shp* file is given to the mask argument the plot is drawn only inside the polygon. In order to avoid boundary artifacts due to reduced resolution, original data are resampled to higher resolution (currently set to 10 times the original one.) Ifinverse_mask is set to TRUE, the plot is drawn outside the polygon. The *mask* feature is based on the terra::mask() function. The CRS of the *shp* file is applied to the data in the data.frame. Please, keep in mind this feature is still experimental.

Value

A ggplot2 object.

Examples

```
# Load example data in long format
data(volcano)
volcano3d <- reshape2::melt(volcano)
names(volcano3d) <- c("x", "y", "z")
# Contour plot with default options
v <- contourPlot2(volcano3d)
v

# Set levels, and properly format the legend title:
contourPlot2(
    volcano3d,
    levels = c(-Inf, seq(100, 200, 20), Inf),
    legend = expression("PM"[10] ~ "[" * mu * "g m"^-3 * "]")
)

# Sometimes, instead of a contour plot it is better to plot the original
# raster data, without any interpolation:
contourPlot2(
    volcano3d,
    levels = c(-Inf, seq(100, 200, 20), Inf),
    tile = TRUE
)

# Since contourPlot2 returns a `ggplot2` object, you can add instructions as:
library(ggplot2)
v +
    ggtitle("Example volcano data") +
    labs(x = NULL, y = NULL)
```

`downloadBasemap` *Download basemap from Italian National Geoportal*

Description

Download the aerial orthophoto of the requested domain from the Italian National Geoportal.

Usage

```
downloadBasemap(
  file = file,
  xSW = NA,
  ySW = NA,
  xExt = NA,
  yExt = NA,
  crs = 32,
  width = 1024,
  height = 1024,
  units = "px",
  res = 72
)
```

Arguments

`file`	Path to output file. If file exists, it will be overwritten.
`xSW`	South West Easting UTM coordinate of the basemap (in metres).
`ySW`	South West Northing UTM coordinate of the basemap (in metres).
`xExt`	Easting extension in metres.
`yExt`	Northing extension in metres.
`crs`	Coordinate Reference System as UTM zone: either 32 (default) or 33.
`width`	The basemap width (default = 1024).
`height`	The basemap height (default = 1024).
`units`	The unit of measure of width and height. It can be `px` (pixels, the default), `in` (inches), `cm` or `mm`
`res`	The resolution in dpi (default = 72).

Details

The domain is specified by the South-West point coordinates, and its extension in the `x` and `y` directions. The Coordinate Reference System (CRS) is in UTM 32 or 33.

Note that, even if the downloading is successful the file might be empty due to some weird behaviour of the remote server from the PCN.

Value

The output is a *tiff* encoded with GeoTIFF metadata at the path provided. No value is returned.

Examples

```
## Not run:
# Download a basemap of a domain with SW coordinates (410000, 5000500)
# in the UTM32 CRS and extension 5000m in both directions.

downloadBasemap(
    file = "./basemap.tif",
    xSW = 410000, ySW = 5000500, xExt = 5000, yExt = 5000
)

# Download a basemap of a domain with SW coordinates (410000, 5000500)
# in the UTM32 CRS and extension 5000m in both directions.
# The file has to be 2048 x 2048 pixels.

downloadBasemap(
    file = "./basemap.tif",
    xSW = 410000, ySW = 5000500, xExt = 5000, yExt = 5000,
    width = 2048, height = 2048
)

# Download a basemap of a domain with SW coordinates (410000, 5000500)
# in the UTM32 CRS and extension 5000m in both directions.
# The file has to be 10cm x 10cm with a resolution of 150 dpi.

downloadBasemap(
    file = "./basemap.tif",
    xSW = 410000, ySW = 5000500, xExt = 5000, yExt = 5000,
    width = 10, height = 10, units = "cm", res = 144
)

## End(Not run)
```

---

importADSOBIN *ADSO/BIN data import function*

Description

Import data from ADSO/BIN binary file. It requires an active Python installation with the arinfopy library.

Usage

```
importADSOBIN(
  file = file.choose(),
  variable = NULL,
```

```
  slice = 1,
  deadline = 1,
  k = 1,
  kz = 1,
  dx = 0,
  dy = 0,
  destaggering = FALSE,
  raster.object = FALSE,
  verbose = FALSE
)
```

Arguments

`file`	Character. The path to the ADSO/BIN file to be imported.
`variable`	Character. A string with the name of the variable to be imported.
`slice`	An integer corresponding to the horizontal slice (vertical level) of 3D variables (default = 1). In the case of a 2D variable, it is ignored.
`deadline`	An integer representing the temporal deadline (default = 1). It can optionally be a string with date time (see examples).
`k`	A numeric factor to be applied to x and y coordinates (default = 1).
`kz`	A numeric factor to be applied to z values to rescale them (default = 1).
`dx`	A number to shift x coordinates by dx (default = 0).
`dy`	A number to shift y coordinates by dy (default = 0).
`destaggering`	Use `TRUE` to apply destaggering to X and Y coordinates (default = `FALSE`).
`raster.object`	Use `TRUE` to return a `raster` object instead of a dataframe with (X, Y, Z) columns (default = `FALSE`).
`verbose`	Use `TRUE` to print out basic statistics (default = `FALSE`).

Details

The `importADSIOBIN()` function imports data from an ADSO/BIN binary file. It relies on the 'arinfopy' (version >= 2.2.0) python library. For more information on the library see the GitHub repository.

For more information on the active `python` installation, check the documentation of **reticulate**.

Value

In standard use, `importADSOBIN()` return a data frame with `(X, Y, Z)` columns. Column Z contains the values of the requested variable. If the `raster.object` option is set, it returns a RasterLayer object.

See Also

`importRaster()`, `importSurferGrd()`

Examples

```
## Not run:
# Read ground level (slice = 1) value of variable M001S001.
pm10 <- importADSOBIN(
    file = "average_2018.bin",
    variable = "M001S001",
    slice = 1
)

# Read deadline 12 of the second vertical level of temperature:
temperature <- importADSOBIN(
    file = "swift_surfpro_01-10_01_2018",
    variable = "TEMPK",
    slice = 2,
    deadline = 12
)

# Read varibale M001S001 at ground level, at given date and time,
# and print basic information:
nox <- importADSOBIN(
    file = "conc_01-10_07_2018",
    variable = "M001S001",
    slice = 1,
    deadline = "2018/07/02 12:00",
    verbose = TRUE
)

## End(Not run)
```

---

importRaster *Import generic raster file*

Description

A function to import the first layer of a generic raster file.

Usage

```
importRaster(
  file = file.choose(),
  k = 1,
  kz = 1,
  dx = 0,
  dy = 0,
  destaggering = FALSE,
  variable = NULL,
  verbose = FALSE
)
```

Arguments

file	character. Path to the raster file.
k	numeric. Factor applied to x and y coordinates (default = 1). For example, it can be used to convert the grid coordinates from km to m (k = 1000).
kz	numeric. Factor applied to the variable values (default = 1).
dx	numeric. Constant to shift x coordinates (default = 0).
dy	numeric. Constant to shift y coordinates (default = 0).
destaggering	Use TRUE to apply destaggering to X and Y coordinates (default = FALSE). See the Details section.
variable	character. The name of the variable to be imported.
verbose	logical. If TRUE, prints out basic statistics (default = FALSE).

Details

This function is based on the **terra** package and it can import any format managed by it as NetCDF.

Destaggering applies a shift equal to half grid size in both horizontal directions. It is useful for importing data from the SPRAY air quality dispoersion model and it is not applied by default.

An optional summary output can be printed out by setting the verbose parameter to TRUE.

Value

A data.frame with x, y and z columns for the grid cells coordiantes and the variable value.

See Also

importADSOBIN(), importSurferGrd()

Examples

```
## Not run:
# Import binary (netcdf) file and convert coordinates from km to m,
# without destaggering. Variable name is "NOx".
mydata <- importRaster(
    file = "/path_to_file/filename.nc",
    variable = "NOx",
    k = 1000,
    destaggering = FALSE
)

# Import binary (netcdf) file and convert coordinates from km to m,
# with shift of 100 m in both directions:
mydata <- importRaster(
    file = "/path_to_file/filename.nc",
    variable = "pm10",
    k = 1000,
    dx = 100,
    dy = 100
)
```

```
## End(Not run)
```

---

importSurferGrd *Import Grid file*

Description

A function to import data from Surfer text grid file.

Usage

```
importSurferGrd(fname, k = 1000, destaggering = FALSE)
```

Arguments

fname	Surfer grd file to be imported
k	Factor to apply to x and y coordinates
destaggering	Boolean variable to apply or not destaggering.

Details

Surfer grd file is imported and an array of x, y, z columns is returned X and y coordinates can be converted from km to m (default k=1000) and vice versa. Destaggering is applied by default.

Value

A dataset with x, y and z columns is returned.

See Also

importRaster(), importADSOBIN()

Examples

```
## Not run:
# Import Surfer Grd file and convert coordinates from km to m,
# with destaggering
mydata <- importSurferGrd("/path_to_file/filename.grd", k = 1000)

# Import Surfer Grd file and do not convert coordinates, without destaggering
mydata <- importSurferGrd(
    "path_to_file/filename.grd",
    k = 1,
    destaggering = FALSE
)

## End(Not run)
```

plotAvgRad *Plot hourly average radiation*

Description

Plot a histogram with hourly average of solar radiation, together with hourly maxima for June and December.

Usage

```
plotAvgRad(
  mydata,
  date = "date",
  rad = "radg",
  ylabel = NULL,
  title = "",
  locale = NULL
)
```

Arguments

mydata	A data frame containing data to plot.
date	The name of the column representing date and time. Data must be of class POSIXlt or POSIXct (default = "date"). If timezone is unspecified, it is set to GMT.
rad	Name of the column representing radiation (default = "radg").
ylabel	The label along the y axis. If missing a default label is plotted.
title	Optional plot title
locale	Locale to use for legend. Default is English, the only other one currently supported is Italian.

Value

A ggplot2 plot.

See Also

plotStabilityClass(), plotAvgTemp()

Examples

```
data(stMeteo)
plotAvgRad(stMeteo, date = "date", rad = "radg")
```

plotAvgTemp *Plot average temperature*

Description

plotAvgTemp builds a bar plot of time average temperature and two line plots with maximum and minimum temperature.

Usage

```
plotAvgTemp(
  mydata,
  date = "date",
  temp = "temp",
  avg.time = "1 month",
  ylabel = NULL,
  title = "",
  locale = NULL
)
```

Arguments

mydata	A dataframe containing data to plot.
date	The name of the column representing date and time. Data must be of class POSIXlt or POSIXct (default = "date"). If timezone is unspecified, it is set to GMT.
temp	Name of the column representing temperature (default = "temp")
avg.time	Defines the time period to average to. Currently the only supported period is "1 month" (default).
ylabel	The label along the y axis. If missing a default label is plotted.
title	Optional plot title
locale	Locale to use for day and month names. Default is current locale. Supported locales are listed in stringi::stri_locale_list(). All other labels are in English by default or in Italian if its locale is specified.

Value

A plot with average, min and max temperature in a given range of time.

See Also

plotStabilityClass(), plotAvgRad()

Examples

```
# Plot average monthly temperature and curves with monthly maximum and minimum
data(stMeteo)
str(stMeteo)
plotAvgTemp(stMeteo)
# Add a custom title
plotAvgTemp(stMeteo, title = "Monthly temperature")

# Override default locale
plotAvgTemp(stMeteo, avg.time = "1 month", locale = "it_IT")
```

---

plotStabilityClass *Plot stability class*

Description

Histogram plot of stability classes by season or hour.

Usage

```
plotStabilityClass(
  mydata,
  date = "date",
  sc = "sc",
  type = "season",
  locale = NULL
)
```

Arguments

mydata	A dataframe containing data to plot.
date	The name of the column representing date and time. Data must be of class POSIXlt or POSIXct (default = "date"). If the timezone is unspecified, it is set to GMT.
sc	The name of the column that represents the stability class (default = "sc").
type	Specify how the data are to be split and plotted. Accepted values are "season" (default) and "hour".
locale	Set the locale for day and month names. The system locale is used by default, but you can specify a different one from the supported ones listed in stringi::stri_locale_list(). All other labels are in English by default or in Italian if its locale is specified.

Details

Numerical values of stability classes are mapped as: 1 = A, 2 = B, ..., 6 = F.

Value

A ggplot2 plot.

See Also

stabilityClass(), plotAvgRad(), plotAvgTemp()

Examples

```
data("stMeteo")

# Season plot of stability class pgt
plotStabilityClass(stMeteo, date = "date", sc = "pgt", type = "season")

# Hourly plot of stability class pgt
plotStabilityClass(stMeteo, date = "date", sc = "pgt", type = "hour")

# Override default locale
plotStabilityClass(
    stMeteo,
    date = "date",
    sc = "pgt",
    type = "season",
    locale = "it_IT"
)
```

---

removeOutliers *Remove data outliers*

Description

Remove data outliers based on the interquartile range.

Usage

```
removeOutliers(x, k = 1.5)
```

Arguments

x	vector of data.
k	factor to applied to the interquartile range (default = 1.5).

Details

The interquartile range IQR is computed from input dataset as IQR = Q3 - Q1, where Q1 is 25th percentile and Q3 is the 75th percentile. Values larger than Q3 + k * IQR and smaller than Q1 - k * IQR are deemed as outliers and substituted with NA's.

The default value of k is 1.5.

Value

A numeric vector with the same length as input vector.

Examples

```
mydata <- c(-10 * runif(10), runif(10))
removeOutliers(mydata)
```

`rollingMax` *Compute Rolling Max*

Description

The function computes the rolling maximum value along a time series.

Usage

```
rollingMax(mydata, length = 24)
```

Arguments

`mydata`	A numeric vector of data values
`length`	An integer specifying the window size (number of observations) to consider. Must be at least 3 (default = 24).

Details

It calculates the maximum over consecutive elements centered within a specified window.

For each index i, it considers a window of `length` points centered around i. When `length` is odd, the center falls exactly on i and the window extends equally to both sides. When `length` is even, the window extends one less point to the left than to the right and the rolling max is not exactly centered.

Values near the start of the series use windows with fewer than `length` data points if there are not enough preceding elements to form a full window. Similarly for values at the end.

Value

A numeric vector containing rolling maximum values, with same dimensions as `mydata`.

Examples

```
# Compute rolling max over a 24-hour period on hourly time series data
data(stMeteo)
ws_24h <- rollingMax(mydata = stMeteo$ws, length = 24)
```

`stabilityClass` *Stability class.*

Description

Computes stability class given net radiation, total cloud cover and wind speed.

Usage

```
stabilityClass(rad, tcc, ws, option = "iaea")
```

Arguments

`rad`	The net radiation in W/m^2
`tcc`	The total cloud cover in a range from 1 to 8
`ws`	wind speed in m/s
`option`	The method used to determine the stability class. It can be `iaea` (default), `pasquill` or `custom`.

Details

`stabilityClass()` computes stability class according to IAEA method based on net radiation, total cloud cover tcc and wind speed. Net radiation and wind are used by day; tcc and wind are used by night.

Three different alogorithms are implemented, selected by the `option` argument.

`iaea` option implements the *radiation-wind method recommended by the International Atomic Energy Agency (IAEA) and it is based on the net radiation during the day and cloud cover by night.

`pasquill` option is based on the original Pasquill formulation and lacks the "very weak" solar insolation present in the modified `iaea` version.

Eventually, the `custom` options is similar to `iaea`, with slightly different set of parameters for net radiation, wind speed and cloud cover.

Previously used option `impact` is the same as `iaea` and it is now deprecated.

Value

`stabilityClass` returns a numeric vector with Pasquill stability classes coded as: A = 1, B = 2, ... , F = 6 ranging from "very unstable" to "very stable".

See Also

`plotStabilityClass()`

Examples

```
# Compute stability class with custom algorithm
stMeteo$cst <- stabilityClass(
    rad = stMeteo$rad,
    tcc = stMeteo$tcc,
    ws = stMeteo$ws,
    option = "custom"
)
```

`stMeteo` *Meteorological dataset with hourly values*

Description

A dataset containing 8760 hourly values of some meteorological variables corresponding to a full solar year.

Usage

```
stMeteo
```

Format

A data frame with 8760 rows and 7 variables:

date date time in yyyy-mm-hh HH:MM:SS

ws wind speed in m/s

wd wind direction in deg.

temp air temperature in C

radg Global solar radiation in W/m^2

tcc Total cloud cover in integers ranging from 0 to 8

pgt Pasquill-Gifford-Turner stability class

Source

Self derived dataset.

`vectorField` *Vector field plot*

Description

Simple function to plot a vector field given two components.

Usage

```
vectorField(
  data,
  scale = 1,
  everyx = 1,
  everyy = 1,
  size = 0.25,
  preview = TRUE
)
```

Arguments

`data`	A dataframe containing data to be plotted in the form of: *(x, y, u, v)*.
`scale`	length factor of vector components
`everyx`	keep one out of every *everyx* values, along *x* direction.
`everyy`	keep one out of every *everyy* values, along *y* direction.
`size`	arrow size.
`preview`	(default = TRUE) create a plot. If FALSE it only creates the ggplot2 directive to be added to another plot.

Details

This function plots a vector field given a data.frame with coordinates (x, y) and corresponding velocity components (u, v). Vectors are coloured by magnitude (speed). The coordinates are assumed to be on a regular rectangular grid in the UTM reference system.

This function is heavily inspired by snippets of code in *R Graphics Cookbook* by Winston Chang (https://r-graphics.org/index.html).

Value

A `ggplot2` object if `preview = TRUE`. A `ggplot2` a plot, as a `contourPlot2()` and the vector field will be overlapped.

Examples

```
## Not run:
metU <- importADSOBIN(
    "/path/to/meteofile",
    variable = 'U',
    slice = 2,
    k = 1000,
    verbose = TRUE
)
metU <- as.data.frame(metU)
metU <- metU %>%
    mutate(u = z, z = NULL)

metV <- importADSOBIN(
    "/path/to/meteofile",
    variable = 'V',
    slice = 2,
    k = 1000,
    verbose = TRUE
 )
metV <- as.data.frame(metV)
metV <- metV |>
    mutate(v = z, z = NULL)

met <- merge(metU, metV, by = c("x", "y"))

vectorField(
    met,
    everyx = 2,
    everyy = 2,
    scale = 10
) +
    coord_fixed(ratio = 1, xlim = c(0, 1000), ylim = c(0, 1000)) +
    scale_color_viridis_c()

# Overlap the vector field to a contour plot and set vector colours to black
met$ws <- sqrt(met$u^2 + met$v^2)
contourPlot2(met, z = "ws") +
    vectorField(
        met,
        everyx = 2,
        everyy = 2,
        scale = 10,
        preview = FALSE
    ) +
    scale_colour_gradient(low = "black", high = "black", guide = NULL)

## End(Not run)
```

Index