Package 'pesco'

July 9, 2015

Type Package

Title Data fusion for air quality data

Version 0.3.0

Date 2015-07-09

Author Lucia Paci, Giovanni Bonafe'

Maintainer G.Bonafe' <gbonafe@arpa.emr.it>

Imports geoR, akima, fields, caTools, ncdf, timeDate, chron

Description PESCO stands for Post-processing and Evaluation with Statistical methods of a Chemistry-transport-model Output. The package provides functions to perform data fusion for air quality data, correcting the output of a deterministic CTM with observed data, with a Trans-Gaussian Kriging approach.

License GPL-2

R topics documented:

`pesco-package` *Data fusion for air quality data*

Description

PESCO stands for "Post-processing and Evaluation with Statistical methods of a Chemistry-transport-model Output". The package provides functions to perform data fusion for air quality data, correcting the output of a deterministic CTM with observed data, with a Trans-Gaussian Kriging approach.

Details

Package:	pesco
Type:	Package
Version:	0.2.2
Date:	2015-03-27
License:	GPL-2

Author(s)

Lucia Paci, Giovanni Bonafe'

Maintainer: G.Bonafe' <gbonafe@arpa.emr.it>

`aqstat.functions` *Functions to calculate Air Quality indicators*

Description

Functions to calculate simple statistical Air Quality indicators, also for legal purposes.

Usage

```
stat.period (x, period, necess,       FUN = mean)
stat.period2(x, period, nmax.missing, FUN = mean)
which.period(x, period, necess,       FUN = which.max)
exc.period  (x, period, necess,       threshold)
```

```
stat.window(x, window, necess, FUN = mean)
mean.window(x, k, necess)

detect.event(x, threshold)
aot(x, hr, threshold = 80, estimate = T,
    hr.min = 8, hr.max = 19)

shift(x, k)
```

Arguments

`x` vector of the concentration values

`period` vector, with the same length as `x`, to distinguish different periods (e.g. days, months)

`window` numerical vectors with two elements; defines the running window, e.g. `c(-7,0)` is the 8 hours window for 8hr running mean of ozone or carbon monoxide

`necess` if >1, number of valid data needed in each time period. If <1, fraction of data needed in each time period.

`nmax.missing` number of missing data accepted in each time period

`FUN` the function to be applied

`threshold` threshold

`k` in `shift`, the number of timesteps you want to shift `x`; in `mean.window`, the width of the window

`hr` numerical vector of the hours (with the same length as `x`)

`estimate` logical. IF `TRUE` the AOT is corrected according to the EU legislation in order to take into account the number of missing values

`hr.min` first hour of the timerange over which AOT is calculated

`hr.max` last hour of the timerange over which AOT is calculated

Details

The functions `stat.period` and `stat.period2` apply the function `FUN` over defined time periods, with different approaches in handling missing data. The function `which.period` is similar to `stat.period`, but you can use it for functions (such as `which.min` or `which.max`) which do not accept the argument `na.rm`.

Instead `stat.window` operates on a floating window, and calls `shift` that moves the time series forward or backward in time. The function `mean.window` do the same and is more efficient, but limited to the moving average.

The function `exc.period` counts exceedances of a given threshold. Instead `detect.event` returns an array containing the date and time of the exceedances and their duration (expressed in number of timestep).

`aot` calculates Accumulated exposure Over Threshold

boxcox *Box-Cox transformation*

Description

One parameter Box-Cox transformation

Usage

```
boxcox(x, lambda)
```

Arguments

x	numeric vector to be trasformed
lambda	lambda parameter, as numeric scalar

Details

If $\lambda = 0$, then $log(x)$.

If $\lambda = 1$, then x.

Otherwise, $(x^{\lambda} - 1)/\lambda$.

References

Box, George EP, and David R. Cox. "An analysis of transformations." Journal of the Royal Statistical Society. Series B (Methodological) (1964): 211-252.

char.functions *Functions to manage strings*

Description

Functions to manage strings.

Usage

```
xgrep (pattern,string,where=FALSE)
subwrd(string,pos)
small (string)

capital      (strings)
Capital      (strings)
trim.leading (strings)
trim.trailing(strings)
trim         (strings)
```

Arguments

`string`	string
`strings`	vector of strings
`pattern`	pattern to be found
`pos`	position of the word in the string
`where`	logical. If `TRUE`, the first position of the `pattern` in `string` is returned. If `FALSE`, it is only checked if the `pattern` is included in `string`

`daily_synthesis` *Daily AQ indicators*

Description

Functions to calculate daily Air Quality indicators

Usage

```
dailyCtm(data, statistic)
dailyObs(data, statistic, pollutant,
         Time="Time", Code="Code",
         others=c("Name","Municipality",
                  "Lat","Lon","Elev","Type"))

dailyStat(x, time, statistic, necess=0.75)
```

Arguments

`data`	input hourly data. For `dailyObs`, a data frame; for `dailyCtm`, a list of 3 elements: **coords** coordinates, in a list of 2 **x** numeric matrix `[nx,ny]` **y** numeric matrix `[nx,ny]` **time** vector of `nt` POSIXct **data** concentration values in a 3 dimensions array `[nx,ny,nt]`
`statistic`	daily statistic to be used; possible values are "mean", "max" and "max8h" (daily maximum of the 8hr running mean)
`pollutant`	name of the column with pollutant concentations
`Time`	name of the column with time
`Code`	name of the column with station's code
`others`	vector of the names of the columns with station's static attributes
`x`	numeric vector of hourly data
`time`	vector of date-times as POSIXct
`necess`	fraction of valid hourly data needed in a day

date.functions *Functions to manage dates*

Description

Functions to manage dates, useful to read GrADS ctl+bin files.

Usage

```
it2en.date(date)
en2it.date(date)
date.lang()

date2Date(date)
Date2date(Date)

seq.date(from,to,by="1 day")
format.dates(dates)

ITholidays(years)
```

Arguments

date	date string in the format used in GrADS ctl files ('1jan2001')
Date	object of class Date
from	first day, in the format used in GrADS ctl files ('1jan2001')
to	last day, in the format used in GrADS ctl files ('1jan2001')
by	time step
dates	character vector of dates in the format 'yyyymmdd'
years	numerical vector of years

Details

The functions it2en.date and en2it.date translate a date, formatted as in GrADS ctl files, from Italian to English and vice versa, respectively, while date.lang checks if the system language is Italian or English.

The functions date2Date and Date2date convert a string in the format used in GrADS ctl files ('1jan2001') to an object of class Date and vice versa, respectively.

seq.date builds a sequence of dates in the format used in GrADS ctl files ('1jan2001').

format.dates prepares dates in a way which can be useful for an axis.

ITholidays returns Italian holidays for given years.

See Also

holiday, seq.dates, Date,

`elevation` *Emilia-Romagna topography*

Description

Emilia-Romagna topography on a grid with 1km resolution

Usage

```
data("elevation")
```

Format

List of 2

`coords` Coordinates of the 47817 grid cells (UTM 32N WGS84, in meters), as a list of x and y

`data` Elevation (meters above mean sea level). `NA` outside Emilia-Romagna region.

Examples

```
data(elevation)
str(elevation)
```

`emissions` *Emissions of PM10 and NO2 in Emilia-Romagna*

Description

Proxies of the emission densities of PM10 and NO2 in Emilia-Romagna (in year 2010) disaggregated on a grid with 1km resolution

Usage

```
data("emissions")
```

Format

List of 6 elements:

`PM10.summer` Emissions of PM10 in summer

`PM10.winter` Emissions of PM10 in winter

`PM10.annual` Annual emissions of PM10

`NO2.summer` Emissions of NO2 in summer

`NO2.winter` Emissions of NO2 in winter

`NO2.annual` Annual emissions of NO2

Each of the 6 elements is a list of 2:

`coords` Coordinates of the 47817 grid cells (UTM 32N WGS84, in meters), as a list of x and y

`data` Emissions data. Zero outside Emilia-Romagna region.

Examples

```
data(emissions)
str(emissions)
```

`geo.functions` *Geographical functions*

Description

Some useful geographical functions

Usage

```
ll2utm     (rlat, rlon, iz=32)
ll2utm.grid(lat, lon, round=-2, iz=32)

Dist(x,y,xi,yi)
which.nearest(xo,yo,xi,yi)
```

Arguments

`rlat`	latitude
`rlon`	longitude
`lat`	latitude (numeric vector)
`lon`	longitude (numeric vector)
`iz`	UTM zone
`round`	rounding on the UTM target coordinates
`x`	numeric x coordinate
`y`	numeric y coordinate
`xi, xo`	numeric vector of x coordinates
`yi, yo`	numeric vector of y coordinates

Details

`ll2utm` converts geographical coordinates (latitude/longitude) in UTM WGS84, while `ll2utm.grid` does the same on vectors of coordinates, applying some rounding, supposed they are on a regular grid in the target UTM system.

Interp *Interpolates spatial data*

Description

Interpolates data from regular grid or sparse points to (another) regular grid.

Usage

```
Interp(x, y, z, xp, yp, method = "linear", type = "points")
```

Arguments

x	vector of x coordinates (origin)
y	vector of y coordinates (origin)
z	numeric vector of values to be interpolated
xp	vector of x coordinates (target)
yp	vector of y coordinates (target)
method	interpolation method; can be "linear", "spline" or "nearest"
type	interpolation type; can be "points" or "grid"

kriging *Trans-Gaussian kriging*

Description

Trans-Gaussian kriging

Usage

```
kriging(x.pnt, y.pnt, obs, model,
        proxy.1, proxy.2 = NULL, lambda,
        K.max.dist = 2e+05, K.min.dist = 1000, K.pairs.min = 1)
```

Arguments

x.pnt	vector of x coordinates of the stations
y.pnt	vector of y coordinates of the stations
obs	numeric vector of daily values observed at the stations
model	daily values provided by a chemistry-transport model, interpolated over the target grid
proxy.1	external variable, defined over the target grid

`proxy.2` another external variable (optional), defined over the target grid

`lambda` parameter for the Box-Cox transformation

`K.max.dist` a numerical value defining the maximum distance for the variogram; pairs of locations separated for distance larger than this value are ignored for the variogram calculation

`K.min.dist` a numeric value; points which are separated by a distance less than this value are considered co-located

`K.pairs.min` an integer number defining the minimum numbers of pairs for the bins

See Also

To properly format the input data: `prepare.day`

`NO2.obs` *Observed NO2 data in Emilia-Romagna*

Description

Hourly concentrations of NO2 measured by the monitoring stations in Emilia-Romagna

Usage

```
data("NO2.obs")
```

Format

A data frame in the long-table format with 236520 hourly observations on the following 9 variables.

`Time` sampling time as POSIXct

`NO2` a numeric vector with NO2 concentrations in microgram per cubic meter

`Name` a factor with the names of the monitoring stations

`Municipality` a factor with the names of the municipalities

`Code` a factor with the codes of the stations

`Lat` latitudes as numeric vector

`Lon` longitudes as numeric vector

`Elev` elevations as numeric vector

`Type` a numeric vector identifying the station type

Examples

```
data(NO2.obs)
str(NO2.obs)
```

PM10.ctm *Concentrations of PM10 simulated by a Chemistry-Transport Model*

Description

Hourly concentrations of PM10 simulated by the Chemistry-Transport Model CHIMERE at the ground level

Usage

```
data("PM10.ctm")
```

Format

List of 3

- `coords` Coordinates of the grid cells (UTM 32N WGS84, in meters), as a list of two numeric matrices [1:128, 1:82], x and y
- `time` Time, vector of 24 POSIXct
- `data` Concentration in microgram per cubic meter, numeric array [1:128, 1:82, 1:24]

References

Stortini, M., et al. "Long-term simulation and validation of ozone and aerosol in the Po Valley." Developments in Environmental Science 6 (2007): 768-770.

Bessagnet, B., et al. "Aerosol modeling with CHIMERE - preliminary evaluation at the continental scale." Atmospheric Environment 38.18 (2004): 2803-2817.

Examples

```
data(PM10.ctm)
str(PM10.ctm)
```

PM10.obs *Observed PM10 data in Emilia-Romagna*

Description

Daily concentrations of PM10 measured by the monitoring stations in Emilia-Romagna

Usage

```
data("PM10.obs")
```

Format

A data frame in the long-table format with 760 daily observations on the following 9 variables.

`Time` sampling time as POSIXct

`PM10` a numeric vector with PM10 concentrations in microgram per cubic meter

`Name` a factor with the names of the monitoring stations

`Municipality` a factor with the names of the municipalities

`Code` a factor with the codes of the stations

`Lat` latitudes as numeric vector

`Lon` longitudes as numeric vector

`Elev` elevations as numeric vector

`Type` a numeric vector identifying the station type

Examples

```
data(PM10.obs)
str(PM10.obs)
```

population *Emilia-Romagna population*

Description

Emilia-Romagna population (in year 2010) on a grid with 1km resolution

Usage

```
data("population")
```

Format

List of 2

`coords` Coordinates of the 47817 grid cells (UTM 32N WGS84, in meters), as a list of x and y

`data` Population density (people per sq.km). Zero outside Emilia-Romagna region.

Examples

```
data(population)
str(population)
```

`prepare.functions` *Functions to prepare input for* `kriging`

Description

Functions to prepare input for `kriging`

Usage

```
prepare.day(day, obs.daily, ctm.daily,
            pollutant,
            emis.winter, emis.summer,
            elev = NULL, verbose = FALSE)

prepare.ctm(ctm.daily, day,
            x.pnt, y.pnt, x.grd, y.grd,
            conc.min = 10^-6)
prepare.emis(emis.winter, emis.summer, day,
             x.pnt, y.pnt, x.grd = NULL, y.grd = NULL)
prepare.elev(elev, x.pnt, y.pnt,
             z.pnt = rep(NA,length(x.pnt)),
             x.grd = NULL, y.grd = NULL)
prepare.obs(obs.daily, day, pollutant,
             conc.min = 10^-6)
```

Arguments

`day` required day, in the format "YYYY-MM-DD"

`obs.daily` data frame with daily observations, as returned by `dailyObs`

`ctm.daily` CTM output, aggregated on a daily basis, as returned by `dailyCtm`

`emis.winter` winter emissions, in a list of 2 elements:

- **coords** coordinates, a list of 2 numeric vectors x and y
- **data** numeric vector

`emis.summer` summer emissions, in a list of 2 elements:

- **coords** coordinates, a list of 2 numeric vectors x and y
- **data** numeric vector

`elev` elevation, in a list of 2 elements:

- **coords** coordinates, a list of 2 numeric vectors x and y
- **data** numeric vector

`pollutant` name of the column with pollutant concentations

`verbose` logical; if `TRUE` some messages are given

`x.pnt` numeric vector of x coordinates of the stations

`y.pnt` numeric vector of y coordinates of the stations `z.pnt` numeric vector of elevation of the stations

`x.grd` numeric vector of x coordinates of the grid cells

`y.grd` numeric vector of y coordinates of the grid cells

`conc.min` minimum concentration; if a concentration is less than this value, then it is set equal to it

Details

To prepare the required input for the function `kriging`, you need only the function `prepare.day`, which includes all the functions `prepare.ctm`, `prepare.emis`, `prepare.elev` and `prepare.obs`.

`read.functions` *Read data as provided by Arpa-ER*

Description

Functions to read some data as provided by Arpa Emilia-Romagna

Usage

```
read.qaria (file)
read.sql   (file)
qaria2long (datafiles, anafile, codes=NULL)

read.ncdf.arpaer(con=NULL, pollutant="pm10", lev=1,
                 tz.in="UTC", tz.out="Africa/Algiers")
read.grads(filectl)

read.field(file, coords.col=1:2, data.col=3, coords.fact=1, ...)
```

Arguments

`file` name of the input file

`datafiles` vector of names of files (format 'estra_qaria', an internal standard for some people in Arpa-ER)

`anafile` file with stations metadata

`codes` vector with stations codes, if `NULL` it is argued from the elements of `datafiles`, if they are in the form /path/to/files/pollutant_stationcode.asc

`con` name of a NetCDF file, or an already open connection to NetCDF

`pollutant` pollutant code used in the NetCDF

`lev` level

`tz.in` timezone used by the CTM

`tz.out` timezone used for the observed data

`filectl` name of the GrADS ctl file

`coords.col`	a vector with the numbers of the columns containing the coordinates
`data.col`	a scalar with the number of the column containing the data
`coords.fact`	multiplication factor for coordinates (e.g. 1000 if you need to convert kilometers to meters)
`...`	optional arguments; will be passed to `read.table`, called by `read.field`

Note

If your input data are not provided following internal 'standard' formats of Arpa-ER, you don't need these functions, except maybe `read.field`, which is quite general. Then, you'll need to write your own reading functions, suitable for your data format.

The function `read.grads` is deprecated, since I wrote it when I was young, and it needs external executables.

`time.functions` *Functions to manage time*

Description

Functions to manage time (objects of class `POSIXct`).

Usage

```
Hour   (x, tz = "Africa/Algiers")
Month  (x, tz = "Africa/Algiers")
Year   (x, tz = "Africa/Algiers")
Ymd    (x, tz = "Africa/Algiers")
Ym     (x, tz = "Africa/Algiers")
YQ     (x, tz = "Africa/Algiers")
Ndays  (x, tz = "Africa/Algiers")
Nmonths(x, tz = "Africa/Algiers")

Ndays.in.year(year, tz = "Africa/Algiers")

tz.change(x, tz.in="UTC", tz.out="Africa/Algiers")
```

Arguments

`x`	vector of class `POSIXct`
`year`	year (numeric)
`tz`	timezone
`tz.in`	original timezone of `x`
`tz.out`	target timezone

See Also

`POSIXct`

Index