Package ‘opentraj’

February 20, 2015

Type Package

Title Tools for Creating and Analysing Air Trajectory Data

Version 1.0

Date 2014-08-22

Author Thalles Santos Silva

Maintainer Thalles Silva <tsantossilva@algomau.ca>

Description opentraj uses the Hybrid Single Particle Lagrangian Integrated Trajectory Model (HYSPLIT) for computing simple air parcel trajectories. The functions in this package allow users to run HYSPLIT for trajectory calculations, as well as get its results, directly from R without using any GUI interface.

License GPL-2

Imports plyr, maptools, openair, raster, rgdal, reshape, doParallel, parallel, foreach, sp

LazyData yes

Depends R (>= 2.10)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-19 17:35:43

R topics documented:

`opentraj-package` *Functions to create and analyse air trajectory data.*

Description

The opentraj project was initially developed at the Natural Resources Canada labs in Sault Ste. Marie, Ontario, with the goal of analyzing the outbreak of the Spruce Budworm insects throughout Canada. The opentraj project aims to provide a collection of functions to create and analyse air trajectory data. These functions join the capabilities offered by the openair project along with the classes and methods for spatial data analysis provided by the R sp package to process air trajectory data.

This package uses the Hybrid Single Particle Lagrangian Integrated Trajectory Model (HYSPLIT) for computing simple air parcel trajectories. The functions offered by this package allow users to run HYSPLIT for trajectory calculations, as well as get its results, directly from R without using any GUI interface.

In addition, because this package bases their data format in the ones used by openair, many of the openair's functions can be used to process these results as well. One of the main advantages of this package, in relation to openair, is the possibility of running either Forward and Backward HYSPLIT trajectories. Also, this package uses methods for spatial data provided by the sp package.

Details

Package:	opentraj
Type:	Package
Version:	1.0
Date:	2014-08-21
License:	GPL-2

The most important functions are:

`ProcTraj`, `Df2SpLines`, `Df2SpLinesDf`, `PlotTraj`, `PlotTrajFreq`, `RasterizeTraj`.

Author(s)

OpenTraj is a colaborative effort of Thalles Santos Silva and Jean-Noel Candau.

`AddMetFiles` *Add meteorological file path*

Description

This Utility function is called bt ProcTraj and should not be used independetly. AddMetFiles helps in the process of creating the CONTROL files for HYSPLIT.

Usage

```
AddMetFiles(month, Year, met, script.file, control.file)
```

Arguments

`month`	Integer; month of the meteorological file to be added.
`Year`	Integer; Year of the meteorological file to be added.
`met`	String; Path to the meteorological files.
`script.file`	String; name of the script that is being created by ProcTraj.
`control.file`	String; Control file name.

Details

AddMetFiles is called by ProcTraj is a loop which ensures that always three meteorological data files will be added for each CONTROL file, one for the current month, one for the previous month and another one for the following month. This function assumes that the meteorological files were downloaded from HYSPLIT, so that, they have the following name structure: "RP<Year><Month>.gbl". For example, for Year = 2007 and Month = 06, the meteorological file name would be: "RP200706.gbl"

Author(s)

Thalles Santos Silva

See Also

`ProcTraj`

air.traj *HYSPLIT Forward air trajectories.*

Description

This data set contains forward air trajectories calculated by HYSPLIT using the function ProcTraj for the date of 2010/06/10 (YYYY,MM,DD), from the Natural Resources Canada in Sault Ste Marie, Ontario, Canada. This data set was calculated with latitude and longitude equal to 46.503784 and -84.304442 respectively, hour interval of 1 hour from 00:00 am to 23:00 pm of 2010/06/10. Each air trajectory has 12 hours of duration, and starts at a altitude of 100 meters above ground.

Usage

```
data("air.traj")
```

Format

A data frame with 312 observations on the following 12 variables.

`receptor` a numeric vector

`year` a numeric vector; calculating year.

`month` a numeric vector; calculating month.

`day` a numeric vector; calculating day.

`hour` a numeric vector; calculating hour.

`hour.inc` a numeric vector; trajectory's hour increment, i.g. age of the trajectory in hours.

`lat` a numeric vector; trajectory's latitude starting point.

`lon` a numeric vector; trajectory's longitude starting point.

`height` a numeric vector; level above ground (meters)

`pressure` a numeric vector; diagnostic output variables

`date2` a POSIXct; Year month day hour minute of the point

`date` a POSIXct; Starting year, month, day, and hour

Details

For more information regarding HYSPLIT trajectory's endpoint data, please refer to http://www.arl.noaa.gov/documents/repo

Examples

```
data(air.traj)
str(air.traj)
```

Df2SpLines *Data Frame to Spatial Lines*

Description

This function converts an object of type data.frame, calculated by the function ProcTraj, into an object of type SpatialLines-class.

Usage

```
Df2SpLines(df, crs=NA)
```

Arguments

df	data.frame Object created by the function ProcTraj.
crs	String: Valid projection string. An example would be crs="+proj=longlat +datum=NAD27"

Details

An individual line consists of a set of lines in the data frame that contains the same ID. This function identifies individual trajectories based on their length. It is assumed that all trajectories calculated by HySplit using the ProcTraj function have the same length. Thus, once known the length of the trajectories, this function splits the data frame in X different data frames where each data frame contains R rows, R being the trajectory's length and X being the number of rows in the initial data frame divided by the trajectory's length. Each of the X different data frames will be transformed into a different line.

Value

Returns an object of class SpatialLines-class.

Author(s)

Thalles Santos Silva

See Also

data.frame, ProcTraj, SpatialLines-class.

Examples

```
## load data frame of HYSPLIT trajectory calculations calculated by function ProcTraj
crs <- "+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0"
air.traj.lines <- Df2SpLines(air.traj, crs)
PlotTraj(air.traj.lines)
```

Df2SpLinesDf *Data Frame to Spatial Lines Data Frame*

Description

This function converts an object of class `SpatialLines-class`, calculated by the function `Df2SpLines`, into an Object of class SpatialLinesDataFrame.

Usage

```
Df2SpLinesDf(spLines, df, add.distance=F, add.azimuth=F)
```

Arguments

`spLines`	Object of class `SpatialLines-class` calculated by the function `Df2SpLines`.
`df`	`data.frame` Object created by the function `ProcTraj`.
`add.distance`	Logical: If True, it will calculate and include the distance in meters between the first and last point for every line.
`add.azimuth`	Logical: If True it will calculate and include the azimuth for every line.

Details

Because the additional information carried by the SpatialLinesDataFrame Object have to be a data frame with same number of lines as the number of lines in the SpatialLines Object, the additional information, which each line of the SpatialLinesDataFrame will have, concerns to the first row of an individual trajectory from the data frame calculated by the function `ProcTraj`

Value

Returns an object of class SpatialLinesDataFrame.

Author(s)

Thalles Santos Silva

See Also

`Df2SpLines`, `ProcTraj`, `SpatialLines-class`, `data.frame`.

Examples

```
crs <- "+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0"
air.traj.lines <- Df2SpLines(air.traj, crs)
air.traj.linesDf <- Df2SpLinesDf(air.traj.lines, air.traj)
PlotTraj(air.traj.linesDf)
```

`hy.traj2007` *HySplit trajectory's calculations*

Description

This data set gives HYSPLIT forward air trajectories calculations computed by the function ProcTraj for the year of 2007 with 3 hours of duration.

Usage

```
data(hy.traj2007)
```

Format

A data frame with 111100 observations on the following 12 variables.

`receptor` a numeric vector;

`year` a numeric vector; calculating year.

`month` a numeric vector; calculating month.

`day` a numeric vector; calculating day.

`hour` a numeric vector; calculating hour.

`hour.inc` a numeric vector; trajectory's hour increment, i.g. age of the trajectory in hours.

`lat` a numeric vector; trajectory's latitude starting point.

`lon` a numeric vector; trajectory's longitude starting point.

`height` a numeric vector; level above ground (meters)

`pressure` a numeric vector; diagnostic output variables

`date2` a POSIXct; Year month day hour minute of the point

`date` a POSIXct; Starting year, month, day, and hour

Details

For more information regarding HYSPLIT trajectory's endpoint data, please refer to http://www.arl.noaa.gov/documents/repo

Examples

```
data(hy.traj2007)
str(hy.traj2007)
```

`hytraj07.lines` *HYSPLIT Trajectory Lines*

Description

This object is an output example from the function `Df2SpLines` using the data set `hy.traj2007` as input, and CRS = "+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0".

Usage

```
data(hytraj07.lines)
```

Examples

```
# data(hytraj07.lines)
# str(hytraj07.lines)

# library(sp)
# plot(hytraj07.lines)
```

`hytraj07.linesDf` *HYSPLIT Trajectory Lines Data Frame*

Description

This object is an output example from the function `Df2SpLinesDf` using the object `hytraj07.lines` and the data set `hy.traj2007` as input values.

Usage

```
data(hytraj07.linesDf)
```

Examples

```
## data(hytraj07.linesDf)
## str(hytraj07.linesDf)

## library(sp)
## plot(hytraj07.linesDf)
```

`ph07.traj.freq` *Pheno 2007 trajectory frequency calculation*

Description

This Object of class RasterLayer was calculated by function `RasterizeTraj`, and it represents the trajectory frequency for the SpatialLines defined by `hy.traj2007` with resolution grid value of 10000 kilometers.

Usage

```
data(ph07.traj.freq)
```

`pheno2007` *Phenology of eastern spruce budworm adult emergence in Quebec in 2007.*

Description

This data set contains estimates of spruce budworm adult phenology in various locations in Quebec (Canada) in 2007. The phenology of adult emergence is represented by a number of adults emerging at particular dates. This number was calculated using the spruce budworm phenology model in BioSIM 10 (https://cfs.nrcan.gc.ca/projects/133) with default parameters.

Usage

```
data(pheno2007)
```

Format

A data frame with 5555 observations on the following 5 variables.

`ID` a numeric vector; Location Number

`Latitude` a numeric vector; Latitude

`Longitude` a numeric vector; Longitude

`Year.Month.Day` a factor; Date of emergence

`Adults` a numeric vector; Number of adults emerging (based on a starting value of 100 eggs).

Details

For more information regarding BioSIM, please refer to: https://cfs.nrcan.gc.ca/projects/133.

Source

Data calculated using BioSIM 10 (https://cfs.nrcan.gc.ca/projects/133).

References

BioSIM 10 - User's manual. 2013. Regniere, J.; Saint-Amant, R.; Bechard, A. Nat. Resour. Can., Can. For. Serv., Laurentian For. Cent., Quebec (Quebec). Inf. Rep. LAU-X-137E.

Examples

```
data(pheno2007)
str(pheno2007)
```

`PlotBgMap` *Plot Background Map*

Description

This function is called by functions `PlotTraj` and `PlotTrajFreq` just to add map's background.

Usage

```
PlotBgMap(traj, ...)
```

Arguments

`traj`	SpatialLines or SpatialLinesDataFrame object calculated by functions Df2SpLines or Df2SpLinesDf. PlotBgMap uses this object (traj) to get the bounding box values for drawing the map.
`...`	Further arguments to the generic plot function.

Details

This function uses the preloaded data set canada.map.

Author(s)

Thalles Santos Silva

See Also

`PlotTraj`, `PlotTrajFreq`

`PlotTraj` *Plot Trajectory*

Description

The function PlotTraj is designed to plot HySplit trajectories calculated by the function `ProcTraj`.

Usage

```
PlotTraj(traj, ...)
```

Arguments

`traj`	SpatialLines or SpatialLinesDataFrame calculated by the functions `Df2SpLines` and `Df2SpLinesDf` respectively.
`...`	Further arguments to be passed to the generic function plot.

Details

This function calls the function `PlotBgMap` to plot the background map behind the trajectories.

Author(s)

Thalles Santos Silva

See Also

`Df2SpLines`, `Df2SpLinesDf`, `ProcTraj`.

Examples

```
crs <- "+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0"
air.traj.lines <- Df2SpLines(air.traj, crs)
PlotTraj(air.traj.lines)
```

`PlotTrajFreq` *Plot Trajectory Frequency*

Description

This function is designed to display a trajectory frequency map that was output by function `RasterizeTraj`.

Usage

```
PlotTrajFreq(spGridDf, background = T, overlay = NA,
  overlay.color = "white", pdf = F, file.name = "output", ...)
```

Arguments

`spGridDf`	SpatialGridDataFrame Object obtained by the convertion of the raster Object output by the `RasterizeTraj` function.
`background`	Boolean: Indicates whether or not the Canadian's background map should be displayed.
`overlay`	[Optional] If defined, it takes a SpatialPolygonsDataFrame as input and plots it over the spGridDf.
`overlay.color`	String. sets the Polygons' color defined by the overlay argument e.g. "blue"
`pdf`	Defines whether or not the output map should be saved in a pdf file.
`file.name`	String: If the argument pdf is True, this argument defined the name of the output file.
`...`	Further arguments to be passed to the generic function plot.

Details

Since the function RasterizeTraj outputs a RasterLayer object, this Object must be converted to a SpatialGridDataFrame Object using the as(rasterObject, "SpatialGridDataFrame") for example.

Value

Trajectory Frequency Map

Author(s)

Thalles Santos Silva

See Also

`RasterizeTraj`

Examples

```
library(raster)

crs <- "+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0"
air.traj.lines <- Df2SpLines(air.traj, crs)
raster.lines <- RasterizeTraj(air.traj.lines, reduce=TRUE, resolution=15000, parallel=FALSE)

r.max.value <- maxValue(raster.lines)
v <- getValues(raster.lines)
v <- v / r.max.value
r <- setValues(raster.lines, v)

## convert raster object to SparialGridDataFrame Object
r1 <- as(r, "SpatialGridDataFrame")

PlotTrajFreq(r1, background = TRUE, main="Title", pdf=FALSE)
```

ProcTraj *Process Trajectory*

Description

The ProcTraj function is responsible for all setup and execution of the HySplit model.

Usage

```
ProcTraj(lat = 51.5, lon = -45.1,
                    hour.interval = 1, name = "london",
                    start.hour = "00:00", end.hour="23:00",
                    met,
                    out,
                    hours = 12, height = 100,
                    hy.path, ID = 1,
                    dates, script.name="test",
                    add.new.column = F, new.column.name, new.column.value,
                    tz = "GMT", clean.files = TRUE)
```

Arguments

lat	Numeric: Initial point's Latitude.
lon	Numeric: Initial point's Longitude.
hour.interval	Integer: This value specifies the hour interval when each trajectory will be calculated.
name	String: Name of the trajectory endpoints file.
start.hour	String: Specifies the START hour of the simulation. An example would be: start.hour = "12:00".
end.hour	String: Specifies the END hour of the simulation. An example would be: start.hour = "14:00".
met	String: Directory location of the meteorological file. More information concerning to meteorological files can be found in http://www.meteozone.com/home/tutorial/html/meteo_ftp.html
out	String: Directory location to which the [output.RData] trajectory end-point files will be written. Always terminate with the appropriate slash (\ or /). If this argument is omitted, the output will only be returned by the function instead of be saved on the local memory. Also, when [out] is omitted, the argument [name] will not be used.
hours	Integer: Total run time. It specifies the duration of the calculation in hours. Backward calculations are entered as negative values. A backward trajectory starts from the trajectory termination point and proceeds upwind. Meteorological data are processed in reverse-time order. Because only two additional meteorological files are loaded, one for the previous and another for the next month, it is recommended a maximum trajectory length of 24 hours.

`height`	numeric: The initial trajectories height. Height is entered as meters above ground-level.
`hy.path`	String: The local path where HySplit is located. Example, for linux/OS X Operating Systems "/home/user/Desktop/hysplit/trunk/"
`ID`	Integer: Process ID. When called in Parallel, the ID argument ensures that each process will deal with separate set of files preventing race condition problems among different processes.
`dates`	Vector containg all the dates that will be calculated by hysplit.
`script.name`	String: Name of the script file that will run HySplit, Default value: "script"
`add.new.column`	Boolean;
`new.column.name`	String:
`new.column.value`	Any Value
`tz`	String: This argument specifies the Time Zone to be applied, e.g. "GMT"
`clean.files`	Boolean: If TRUE, all the files created by HySplit will be deleted.

Details

In order to make the input files for HySplit consistent, the ProcTraj function will always load 3 meteorological files for a specific month. For example, for the month of January 2014, it will load the meteorological files from December of 2013, January of 2014, and February of 2014.

Value

ProcTraj returns a data frame with the pre-calculated HySplit forward or backward trajectories. If the [out] argument is specified with a valid path, ProcTraj will save the data frame with pre-calculated HySplit forward or backward trajectories in the system local storage.

Author(s)

Thalles Santos Silva

Examples

```
##---- For Windows system


##---- For Unix alike systems

# library("opentraj")
# library("doParallel")
#
# ########################
# # SETUP VARIABLES
# kYear <- 2007
# KHeight <- 100
#
```

```
# # path to meteorological files
# # you have to make sure this path is consistent
# # for information on how to get HySplit Meteorological data,
# # http://www.arl.noaa.gov/documents/workshop/Spring2011/HYSPLIT_Tutorial.pdf
# KMetFiles <- "/path/to/the/meteorological/files/"
#
# KOutFiles <- "/path/output/files/"
#
# # HySplit instalation path
# KHySplitPath <- "/path/to/hysplit/"
#
# # load the defoliation point file
# data(pheno2007)
#
# # convert the dates to objects of class Date
# pheno2007$Year.Month.Day <-as.Date(pheno2007$Year.Month.Day)
#
# # subset the data, in order to get only the points with ID = 1
# pointsDf<-split(pheno2007, pheno2007$ID)
#
# # get the number of phisical cores availables
# cores <- detectCores()
# #
# cl <- makeCluster(cores)
#
# registerDoParallel(cl)
#
# start.time<-Sys.time()
#
# hy.traj2007 <-
#   foreach(i = 1:length(pointsDf), .packages="opentraj", .combine = rbind) %dopar%
# {
#   points <- pointsDf[[i]]
#
#   # get the point's latitude and longitude
#   lat<-points[[2]][1]
#   long<-points[[3]][1]
#
#   dates <- points$Year.Month.Day
#
#   ########################
#   output.file.name<-""
#   output.file.name<-paste("pheno", "_", as.character(i), "_", sep="")
#
#   ProcTraj(lat = lat, lon = long, year = Year, name = output.file.name,
#            hour.interval = 1,
#            met = KMetFiles, out = KOutFiles,
#            hours = 3, height = KHeight, hy.path = KHySplitPath, ID = i, dates=dates,
#            start.hour = "19:00", end.hour="23:00",
#            tz="EST", clean.files=F)
# }
#
# end.time<-Sys.time()
# time.taken<-end.time - start.time
# time.taken
#
# stopCluster(cl)
```

RasterizeTraj *Rasterize Trajectory*

Description

This function produces a grid over an specified area and then computes the frequency of lines that cross each cells' grid.

Usage

```
RasterizeTraj(spLines, resolution=10000, reduce=TRUE, parallel=FALSE )
```

Arguments

`spLines`	An object of class SpatialLines created by the function `Df2SpLines`.
`resolution`	numeric vector of length 1 or 2 to set the resolution. If this argument is used, arguments ncols and nrows are ignored.
`reduce`	Boolean: If TRUE the result will be reduced to one raster object; if FALSE, this function will return a list of RasterLayer. The size of the list is equal to the number of available cores in the system.
`parallel`	Boolean: If TRUE, the rasterize process will be made in parallel.

Details

Because this function do all the process in parallel, it calls the function `SplitSpLines` and divides the spLines object into N sub sets of Spatial Lines objects, where N is the number of cores availables in the System.

Value

A Object of class RasterLayer or a list of objects RasterLayer.

Author(s)

Thalles Santos Silva

See Also

`SplitSpLines`, `raster`.

Examples

```
crs <- "+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0"
air.traj.lines <- Df2SpLines(air.traj, crs)
raster.lines <- RasterizeTraj(air.traj.lines, reduce=TRUE, resolution=10000, parallel=FALSE)
```

ReadFiles *Read Files*

Description

The ReadFiles function is a utility function called by function ProcTraj. This function reads all the endpoint files output by HYSPLIT, process these files, and put them all together in a single file.

Usage

```
ReadFiles(working_dir, ID, dates, tz)
```

Arguments

working_dir	String; path to HySplit working directory, this is the location of the endpoint trajectory files that will be read by ReadFiles.
ID	Integer: Process ID. When called in Parallel, this ID argument ensures that each process will deal with a separate set of files preventing race condition problems among different processes.
dates	Vector containg all the dates that will be calculated by hysplit.
tz	String; TimeZone e.g "GMT"

Details

Each HYSPLIT endpoints trajectory file has a header containing some information about the trajectory it self. In order to put all trajectories together, ReadFile take this header information out. An example of a single trajectory output by HYSPLIT could be:

3 1
CDC1 7 7 1 0 0
CDC1 7 8 1 0 0
CDC1 7 9 1 0 0
1 FORWARD OMEGA
7 8 11 22 50.185 -67.475 100.0
1 PRESSURE
1 1 7 8 11 22 0 0 0.0 50.185 -67.475 100.0 953.4
1 1 7 8 11 23 0 0 1.0 50.033 -67.312 95.3 956.9
1 1 7 8 12 0 0 0 2.0 49.871 -67.159 90.4 960.0
1 1 7 8 12 1 0 0 3.0 49.708 -67.009 85.8 962.7

In this example, the ReadFile function would take the header information out, so that, only the four last lines would be used.

Author(s)

Thalles Santos Silva

See Also

`ProcTraj`

`SplitSpLines` *Split Spatial Lines*

Description

This function divides an object of class `SpatialLines-class` defined by the argument [sp.lines] into a number of sub sets of SpatialLines defined by the argument [into].

Usage

```
# divides the SpatialLines Object into 8 sub sets of SpatialLines
SplitSpLines(sp.lines, into)
```

Arguments

`sp.lines`	Object of class `SpatialLines-class` calculated by the function `Df2SpLines`.
`into`	Number of times that the sp.lines object must be divided.

Details

If the number provided by the argument [into] is not multiple of the number of lines in the SpatialLines object, the last element of the list will contain a SpatialLines object with more lines than the first ones. Thus, the original SpatialLines object will not be equally divided.

Although this function might be used seperately, the SplitSpLines function is called by the `RasterizeTraj` function in order to split the spLines and hence, do the process in parallel.

Value

Returns a list of SpatialLines Object.

Author(s)

Thalles Santos Silva

See Also

`SpatialLines-class`, `Df2SpLines`.

Examples

```
## split the SpatialLines object in a list with 8 SpatialLines objects

crs <- "+proj=longlat +datum=NAD83 +no_defs +ellps=GRS80 +towgs84=0,0,0"
air.traj.lines <- Df2SpLines(air.traj, crs)

lines.list <- SplitSpLines(air.traj.lines, 8)
```

`worldmap` *World's Map*

Description

This dataset contains the world's map that is used by function `PlotBgMap` to display geography background.

Usage

```
data("worldmap")
```

Details

This dataset is a low resolution representation of the world's map.

Examples

```
  data(worldmap)
```

Index