Package 'spotoroo'

March 31, 2021

Title Spatiotemporal Clustering of Satellite Hot Spot Data

Version 0.1.1

Description An algorithm to cluster satellite hot spot data spatially and temporally.

License MIT + file LICENSE

URL https://tengmcing.github.io/spotoroo/,
https://github.com/TengMCing/spotoroo/

BugReports https://github.com/TengMCing/spotoroo/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 3.3.0)

Imports geodist (>= 0.0.4), progress (>= 1.2.2), dplyr (>= 1.0.0), cli (>= 2.3.0), stats, patchwork, ggrepel, ggExtra (>= 0.9), ggbeeswarm (>= 0.6.0), ggplot2 (>= 3.0.0)

Suggests sf (>= 0.7-3), testthat (>= 3.0.0), covr, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Weihao Li [aut, cre] (<https://orcid.org/0000-0003-4959-106X>), Di Cook [ctb] (<https://orcid.org/0000-0002-3813-7155>), Emily Dodwell [ctb]

Maintainer Weihao Li <llreczx@gmail.com>

Repository CRAN

Date/Publication 2021-03-31 10:00:03 UTC

R topics documented:

`dist_point_to_vector` *Calculation of the geodesic of a point to multiple points*

Description

This function calculates the geodesic of a point to multiple points given the coordinate information. It is a wrapper of `geodist::geodist_vec()`.

Usage

```
dist_point_to_vector(plon, plat, vlon, vlat)
```

Arguments

`plon`	Numeric. The longitude of a point.
`plat`	Numeric. The latitude of a point.
`vlon`	Numeric. A vector of longitude values.
`vlat`	Numeric. A vector of latitude values.

Value

Numeric. The geodesic of a point to multiple points in meters.

Examples

```
# Define vlon and vlat
vlon <- c(141.12, 141.13)
vlat <- c(-37.1, -37.0)

# Calculate the geodesic
dist_point_to_vector(141.12, -37.1, vlon, vlat)
```

`extract_fire` *Extracting fires from the spatiotemporal clustering results*

Description

This function takes a `spotoroo` object to produce a data frame which contains information about the fire.

Usage

```
extract_fire(result, cluster = "all", noise = FALSE)
```

Arguments

`result`	`spotoroo` object. A result of a call to `hotspot_cluster()`.
`cluster`	Character/Integer. If "all", extract all clusters. If an integer vector is given, extract corresponding clusters.
`noise`	Logical. Whether or not to include noise.

Value

A data.frame. The fire information

- `lon` : Longitude.
- `lat` : Latitude.
- `obsTime` : Observed time.
- `timeID` : Time indexes.
- `membership` : Membership labels.
- `noise` : Whether it is a noise point.
- `distToIgnition` : Distance to the ignition location.
- `distToIgnitionUnit` : Unit of distance to the ignition location.
- `timeFromIgnition` : Time from ignition.
- `timeFromIgnitionUnit` : Unit of time from ignition.
- `type` : Type of the entry, either "hotspot", "noise" or "ignition"
- `obsInCluster` : Number of observations in the cluster.
- `clusterTimeLen` : Length of time of the cluster.
- `clusterTimeLenUnit` : Unit of length of time of the cluster.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)


# Extract all fires
all_fires <- extract_fire(result)
head(all_fires, 3)

# Extract cluster 4
fire_4 <- extract_fire(result, 4)
head(fire_4, 3)
```

`get_fire_mov` *Calculation of the fire movement*

Description

This function calculates the movement of a single fire per `step` time indexes. It collects hot spots per `step` time indexes, then takes the mean or median of the longitude and latitude as the centre of the fire.

Usage

```
get_fire_mov(result, cluster, step = 1, method = "mean")
```

Arguments

`result`	spotoroo object. A result of a call to `hotspot_cluster()`.
`cluster`	Integer. The membership label of the cluster.
`step`	Integer (>0). Step size used in the calculation of the fire movement.
`method`	Character. Either "mean" or "median", method of the calculation of the centre of the fire.

Value

A data.frame. The fire movement.

- `membership` : Membership labels.
- `lon` : Longitude of the centre of the fire.
- `lat` : Latitude of the centre of the fire.
- `timeID` : Time indexes.
- `obsTime` : Observed time (approximated).
- `ignition` : Whether or not it is a ignition point.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)

# Get fire movement of the first cluster
mov1 <- get_fire_mov(result, cluster = 1, step = 3, method = "mean")
mov1

# Get fire movement of the second cluster
mov2 <- get_fire_mov(result, cluster = 2, step = 6, method = "median")
mov2
```

---

global_clustering *Clustering hot spots spatially and temporally*

Description

This function clusters hot spots spatially and temporally.

Usage

```
global_clustering(lon, lat, timeID, activeTime, adjDist)
```

Arguments

lon	Numeric. A vector of longitude values.
lat	Numeric. A vector of latitude values.
timeID	Integer (>=1). A vector of time indexes.
activeTime	Numeric (>=0). Time tolerance. Unit is time index.
adjDist	Numeric (>0). Distance tolerance. Unit is metre.

Details

For more details about the clustering algorithm and the arguments activeTime and adjDist, please check the documentation of hotspot_cluster(). This function performs the **first 3 steps** of the clustering algorithm.

Value

Integer. A vector of membership labels.

Examples

```
# Define lon, lat and timeID for 10 observations
lon <- c(141.1, 141.14, 141.12, 141.14, 141.16, 141.12, 141.14,
          141.16, 141.12, 141.14)
lat <- c(-37.10, -37.10, -37.12, -37.12, -37.12, -37.14, -37.14,
         -37.14, -37.16, -37.16)
timeID <- c(rep(1, 5), rep(26, 5))

# Cluster 10 hot spots with different values of activeTime and adjDist
global_clustering(lon, lat, timeID, 12, 1500)
global_clustering(lon, lat, timeID, 24, 3000)
global_clustering(lon, lat, timeID, 36, 6000)
```

---

handle_noise *Handling noise in the clustering results*

Description

This function finds noise from the clustering results and label it with -1.

Usage

```
handle_noise(global_membership, timeID, minPts, minTime)
```

Arguments

`global_membership`	Integer. A vector of membership labels.
`timeID`	Integer. A vector of time indexes.
`minPts`	Numeric (>0). Minimum number of hot spots in a cluster.
`minTime`	Numeric (>=0). Minimum length of time of a cluster. Unit is time index.

Details

For more details about the clustering algorithm and the arguments `minPts` and `minTime`, please check the documentation of `hotspot_cluster()`. This function performs the **step 4** of the clustering algorithm. It uses a given threshold (minimum number of points and minimum length of time) to find noise and label it with `-1`.

Value

Integer. A vector of membership labels.

Examples

```
# Define membership labels and timeID for 10 observations
global_membership <- c(1,1,1,2,2,2,2,2,3,3,3,3,3,3)
timeID <- c(1,2,3,2,3,3,4,5,6,3,3,3,3,3)

# Handle noise with different values of minPts and minTime
handle_noise(global_membership, timeID, 4, 0)
handle_noise(global_membership, timeID, 4, 1)
handle_noise(global_membership, timeID, 3, 3)
```

`hotspots` *1070 observations of satellite hot spots*

Description

A dataset containing the 1070 observations of satellite hot spots in Victoria, Australia during the 2019-2020 Australian bushfire season.

Usage

```
hotspots
```

Format

A data frame with 1070 rows and 3 variables:

lon longitude

lat latitude

obsTime observed time

Source

https://www.eorc.jaxa.jp/ptree/

`hotspot_cluster` *Spatiotemporal clustering of hot spot data*

Description

This is the main function of the package.
This function clusters hot spots into fires. It can be used to reconstruct fire history and detect fire ignition points.

Usage

```
hotspot_cluster(
  hotspots,
  lon = "lon",
  lat = "lat",
  obsTime = "obsTime",
  activeTime = 24,
  adjDist = 3000,
  minPts = 4,
  minTime = 3,
  ignitionCenter = "mean",
  timeUnit = "n",
  timeStep = 1
)
```

Arguments

`hotspots`	List/Data frame. A list or a data frame which contains information of hot spots.
`lon`	Character. The name of the column of the list which contains numeric longitude values.
`lat`	Character. The name of the column of the list which contains numeric latitude values.
`obsTime`	Character. The name of the column of the list which contains the observed time of hot spots. The observed time has to be in date, datetime or numeric.

activeTime	Numeric (>=0). Time tolerance. Unit is time index.
adjDist	Numeric (>0). Distance tolerance. Unit is metre.
minPts	Numeric (>0). Minimum number of hot spots in a cluster.
minTime	Numeric (>=0). Minimum length of time of a cluster. Unit is time index.
ignitionCenter	Character. Method to calculate ignition points, either "mean" or "median".
timeUnit	Character. One of "s" (seconds), "m" (minutes), "h" (hours), "d" (days) and "n" (numeric).
timeStep	Numeric (>0). Number of units of timeUnit in a time step.

Details

Arguments timeUnit and timeStep need to be specified to convert date/datetime/numeric to time index. More details can be found in transform_time_id().

This clustering algorithm consisted of **5 steps**:

In **step 1**, it defines T intervals using the time index

$$Interval(t) = [max(1, t - activeTime), t]$$

where $t = 1, 2, ..., T$, and T is the maximum time index. activeTime is an argument that needs to be specified. It represents the maximum time difference between two hot spots in the same local cluster. Please notice that a local cluster is different with a cluster in the final result. More details will be given in the next part.

In **step 2**, the algorithm performs spatial clustering on each interval. A local cluster is a cluster found in an interval. Argument adjDist is used to control the spatial clustering. If the distance between two hot spots is smaller or equal to adjDist, they are directly-connected. If hot spot A is directly-connected with hot spot B and hot spot B is directly-connected with hot spot C, hot spot A, B and C are connected. All connected hot spots become a local cluster.

In **step 3**, the algorithm starts from interval 1. It marks all hot spots in this interval and records their membership labels. Then it moves on to interval 2. Due to a hot spot could exist in multiple intervals, it checks whether any hot spot in interval 2 has been marked. If there is any, their membership labels will be carried over from the record. Unmarked hot spots in interval 2, which share the same local cluster with marked hot spots, their membership labels are carried over from marked hot spots. If a unmarked hot spot shares the same local cluster with multiple marked hot spots, the algorithm will carry over the membership label from the nearest one. All other unmarked hot spots in interval 2 that do not share the same cluster with any marked hot spot, their membership labels will be adjusted such that the clusters they belong to are considered to be new clusters. Finally, all hot spots in interval 2 are marked and their membership labels are recorded. This process continues for interval 3, 4, ..., T. After finishing step 3, all hot spots are marked and their membership labels are recorded.

In **step 4**, it checks each cluster. If there is any cluster contains less than minPts hot spots, or lasts shorter than minTime, it will not be considered to be a cluster any more, and their hot spots will be assigned with -1 as their membership labels. A hot spot with membership label -1 is noise.

Arguments `minPts` and `minTime` need to be specified.

In **step 5**, the algorithm finds the earliest observed hot spots in each cluster and records them as ignition points. If there are multiple earliest observed hot spots in a cluster, the mean or median of the longitude values and the latitude values will be used as the coordinate of the ignition point. This needs to be specified in argument `ignitionCenter`.

Value

A `spotoroo` object. The clustering results. It is also a list:

- `hotspots` : A data frame contains information of hot spots.
  - `lon` : Longitude.
  - `lat` : Latitude.
  - `obsTime` : Observed time.
  - `timeID` : Time index.
  - `membership` : Membership label.
  - `noise` : Whether it is a noise point.
  - `distToIgnition` : Distance to the ignition location.
  - `distToIgnitionUnit` : Unit of distance to the ignition location.
  - `timeFromIgnition` : Time from ignition.
  - `timeFromIgnitionUnit` : Unit of time from ignition.
- `ignition` : A data frame contains information of ignition points.
  - `lon` : Longitude.
  - `lat` : Latitude.
  - `obsTime` : Observed time.
  - `timeID` : Time index.
  - `obsInCluster` : Number of observations in the cluster.
  - `clusterTimeLen` : Length of time of the cluster.
  - `clusterTimeLenUnit` : Unit of length of time of the cluster.
- `setting` : A list contains the clustering settings.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
              lon = "lon",
              lat = "lat",
              obsTime = "obsTime",
              activeTime = 24,
              adjDist = 3000,
              minPts = 4,
```

```
                minTime = 3,
                ignitionCenter = "mean",
                timeUnit = "h",
                timeStep = 1)

# Make a summary of the clustering results
summary(result)

# Make a plot of the clustering results
plot(result, bg = plot_vic_map())
```

`ignition_point` *Calculation of the ignition points*

Description

This function calculates ignition points for all clusters.

Usage

```
ignition_point(lon, lat, obsTime, timeUnit, timeID, membership, ignitionCenter)
```

Arguments

`lon`	Numeric. A vector of longitude values.
`lat`	Numeric. A vector of latitude values.
`obsTime`	Date/Datetime/Numeric. A vector of observed time.
`timeUnit`	Character. One of "s" (seconds), "m"(minutes), "h"(hours), "d"(days) and "n"(numeric).
`timeID`	Integer (>=1). A vector of time indexes.
`membership`	Integer. A vector of membership labels.
`ignitionCenter`	Character. Method of calculating ignition points, one of "mean" and "median".

Details

For more details about the clustering algorithm and the argument `timeUnit`, `timeID` and `ignitionCenter`, please check the documentation of `hotspot_cluster()`. This function performs the **step 5** of the clustering algorithm. It calculates ignition points. For a cluster, when there are multiple earliest hot spots, if `ignitionCenter` is "mean", the centroid of these hot spots will be used as the ignition point. If `ignitionCenter` is "median", median longitude and median latitude of these hot spots will be used.

Value

A data frame of ignition points

- `membership` : Membership labels.
- `lon` : Longitude of ignition points.
- `lat` : Latitude of ignition points.
- `obsTime` : Observed time of ignition points.
- `timeID` : Time indexes.
- `obsInCluster` : Number of observations in the cluster.
- `clusterTimeLen` : Length of time of the cluster.
- `clusterTimeLenUnit` : Unit of length of time of the cluster.

Examples

```
# Define lon, lat, obsTime, timeID and membership for 10 observations
lon <- c(141.1, 141.14, 141.12, 141.14, 141.16, 141.12, 141.14,
          141.16, 141.12, 141.14)
lat <- c(-37.10, -37.10, -37.12, -37.12, -37.12, -37.14, -37.14,
         -37.14, -37.16, -37.16)
obsTime <- c(rep(1, 5), rep(26, 5))
timeUnit <- "n"
timeID <- c(rep(1, 5), rep(26, 5))
membership <- c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2)

# Calculate the ignition points using different methods
ignition_point(lon, lat, obsTime, timeUnit, timeID, membership, "mean")
ignition_point(lon, lat, obsTime, timeUnit, timeID, membership, "median")
```

---

local_clustering *Clustering hot spots spatially*

Description

This function clusters hot spots spatially.

Usage

```
local_clustering(lon, lat, adjDist)
```

Arguments

`lon`	Numeric. A vector of longitude values.
`lat`	Numeric. A vector of latitude values.
`adjDist`	Numeric (>0). Distance tolerance. Unit is metre.

Details

For more details about the clustering algorithm and the argument `adjDist`, please check the documentation of `hotspot_cluster()`. This function performs the **step 2** of the clustering algorithm. It clusters hot spots in a given interval.

Value

Integer. A vector of membership labels.

Examples

```
# Define lon and lat for 10 observations
lon <- c(141.1, 141.14, 141.12, 141.14, 141.16, 141.12, 141.14,
          141.16, 141.12, 141.14)
lat <- c(-37.10, -37.10, -37.12, -37.12, -37.12, -37.14, -37.14,
         -37.14, -37.16, -37.16)

# Cluster 10 hot spots with different values of adjDist
local_clustering(lon, lat, 2000)
local_clustering(lon, lat, 3000)
local_clustering(lon, lat, 4000)
```

`plot.spotoroo` *Plotting spatiotemporal clustering result*

Description

`plot.spotoroo()` is the `plot` method of the class `spotoroo`. It is a simple wrapper of `plot_spotoroo()`.

Usage

```
## S3 method for class 'spotoroo'
plot(x, ...)
```

Arguments

`x`	`spotoroo` object. A result of a call to `hotspot_cluster()`.
`...`	Additional arguments pass to `plot_spotoroo()`

Value

A `ggplot` object. The plot of the clustering results.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)



# Different types of plots

# Default plot
plot(result, "def", bg = plot_vic_map())

# Fire movement plot
plot(result, "mov", cluster = 1:3, step = 3, bg = plot_vic_map())
```

`plot_def` *Default method of plotting the clustering results*

Description

This function plots the clustering result spatially as a scatter plot.

Usage

```
plot_def(
  result,
  cluster = "all",
  hotspot = TRUE,
  noise = FALSE,
  ignition = TRUE,
  from = NULL,
  to = NULL,
```

```
  bg = NULL
)
```

Arguments

result	spotoroo object. A result of a call to hotspot_cluster().
cluster	Character/Integer. If "all", plot all clusters. If an integer vector is given, plot corresponding clusters.
hotspot	Logical. If TRUE, plot the hot spots.
noise	Logical. If TRUE, plot the noise points.
ignition	Logical. If TRUE, plot the ignition points.
from	**OPTIONAL**. Date/Datetime/Numeric. Start time. The data type needs to be the same as the provided observed time.
to	**OPTIONAL**. Date/Datetime/Numeric. End time. The data type needs to be the same as the provided observed time.
bg	**OPTIONAL**. ggplot object. If specified, plot onto this object.

Value

A ggplot object. The plot of the clustering results.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)

# Plot a subset of clusters
plot_def(result, cluster = 1:3)

# Plot all clusters
plot_def(result, cluster = "all")
```

plot_fire_mov *Plotting the fire movement*

Description

This function plots the fire movement. The fire movement is calculated from `get_fire_mov()`.

Usage

```
plot_fire_mov(
  result,
  cluster = "all",
  hotspot = TRUE,
  from = NULL,
  to = NULL,
  step = 1,
  bg = NULL
)
```

Arguments

result	spotoroo object. A result of a call to `hotspot_cluster()`.
cluster	Character/Integer. If "all", plot all clusters. If an integer vector is given, plot corresponding clusters.
hotspot	Logical. If `TRUE`, plot the hot spots.
from	**OPTIONAL**. Date/Datetime/Numeric. Start time. The data type needs to be the same as the provided observed time.
to	**OPTIONAL**. Date/Datetime/Numeric. End time. The data type needs to be the same as the provided observed time.
step	Integer (>0). Step size used in the calculation of the fire movement.
bg	**OPTIONAL**. `ggplot` object. If specified, plot onto this object.

Value

A `ggplot` object. The plot of the fire movements.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
```

```
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)

  # Plot cluster 1 to 4
  plot_fire_mov(result, cluster = 1:4)

  # Plot cluster 1 to 4, set step = 6
  plot_fire_mov(result, cluster = 1:4, step = 6)
```

`plot_spotoroo` *Plotting spatiotemporal clustering result*

Description

This function takes a `spotoroo` object to produce a plot of the clustering results. It can be called by `plot.spotoroo()`.

Usage

```
plot_spotoroo(
  result,
  type = "def",
  cluster = "all",
  hotspot = TRUE,
  noise = FALSE,
  ignition = TRUE,
  from = NULL,
  to = NULL,
  step = 1,
  mainBreak = NULL,
  minorBreak = NULL,
  dateLabel = NULL,
  bg = NULL
)
```

Arguments

`result`	`spotoroo` object. A result of a call to `hotspot_cluster()`.
`type`	Character. Type of the plot. One of "def" (default), "timeline" (timeline) and "mov" (fire movement).

cluster	Character/Integer. If "all", plot all clusters. If an integer vector is given, plot corresponding clusters. Unavailable in plot_timeline().
hotspot	Logical. If TRUE, plot the hot spots. Unavailable in plot_timeline().
noise	Logical. If TRUE, plot the noise. Only used in plot_def().
ignition	Logical. If TRUE, plot the ignition points. Only used in plot_def().
from	**OPTIONAL**. Date/Datetime/Numeric. Start time. The data type needs to be the same as the provided observed time.
to	**OPTIONAL**. Date/Datetime/Numeric. End time. The data type needs to be the same as the provided observed time.
step	Integer (>=0). Step size used in the calculation of the fire movement. Only used in plot_fire_mov().
mainBreak	**OPTIONAL**. Character/Numeric. A string/value giving the difference between major breaks. If the observed time is in date/datetime format, this value will be passed to ggplot2::scale_x_date() or ggplot2::scale_x_datetime() as date_breaks. Only used in plot_timeline().
minorBreak	**OPTIONAL**. Character/Numeric. A string/value giving the difference between minor breaks. If the observed time is in date/datetime format, this value will be passed to ggplot2::scale_x_date() or ggplot2::scale_x_datetime() as date_minor_breaks. Only used in plot_timeline().
dateLabel	**OPTIONAL**. Character. A string giving the formatting specification for the labels. If the observed time is in date/datetime format, this value will be passed to ggplot2::scale_x_date() or ggplot2::scale_x_datetime() as date_labels. Unavailable if the observed time is in numeric format. Only used in plot_timeline().
bg	**OPTIONAL**. ggplot object. If specified, plot onto this object. Unavailable in plot_timeline().

Details

if type is "def", the clustering results will be plotted spatially. See also plot_def(). Available arguments:

- result
- type
- cluster
- ignition
- hotspot
- noise
- from (**OPTIONAL**)
- to (**OPTIONAL**)
- bg (**OPTIONAL**)

if type is "mov", plot of the fire movement will be made. See also plot_fire_mov(). Available arguments:

- result
- type
- cluster
- hotspot
- from (**OPTIONAL**)
- to (**OPTIONAL**)
- step
- bg (**OPTIONAL**)

if type is "timeline", plot of the timeline will be made. See also plot_timeline(). Available arguments:

- result
- type
- from (**OPTIONAL**)
- to (**OPTIONAL**)
- mainBreak (**OPTIONAL**)
- minorBreak (**OPTIONAL**)
- dateLabel (**OPTIONAL**)

Value

A ggplot object. The plot of the clustering results.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering result
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)

# Different types of plots

# Default plot
```

```
plot_spotoroo(result, "def", bg = plot_vic_map())


# Fire movement plot
plot_spotoroo(result, "mov", cluster = 1:3, step = 3,
              bg = plot_vic_map())
```

`plot_timeline` *Plotting the timeline of the fire and the noise*

Description

This function plots the timeline of the fires and the noise points.

Usage

```
plot_timeline(
  result,
  from = NULL,
  to = NULL,
  mainBreak = NULL,
  minorBreak = NULL,
  dateLabel = NULL
)
```

Arguments

`result`	spotoroo object. A result of a call to `hotspot_cluster()`.
`from`	**OPTIONAL**. Date/Datetime/Numeric. Start time. The data type needs to be the same as the provided observed time.
`to`	**OPTIONAL**. Date/Datetime/Numeric. End time. The data type needs to be the same as the provided observed time.
`mainBreak`	**OPTIONAL**. Character/Numeric. A string/value giving the difference between major breaks. If the observed time is in date/datetime format, this value will be passed to `ggplot2::scale_x_date()` or `ggplot2::scale_x_datetime()` as `date_breaks`.
`minorBreak`	**OPTIONAL**. Character/Numeric. A string/value giving the difference between minor breaks. If the observed time is in date/datetime format, this value will be passed to `ggplot2::scale_x_date()` or `ggplot2::scale_x_datetime()` as `date_minor_breaks`.
`dateLabel`	**OPTIONAL**. Character. A string giving the formatting specification for the labels. If the observed time is in date/datetime format, this value will be passed to `ggplot2::scale_x_date()` or `ggplot2::scale_x_datetime()` as `date_labels`. Unavailable if the observed time is in numeric format.

Value

A ggplot object. The plot of the timeline.

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)

# Plot timeline
plot_timeline(result,
              mainBreak = "1 week",
              minorBreak = "1 day",
              dateLabel = "%b %d")
```

---

plot_vic_map *Plotting map of Victoria, Australia*

Description

This function plots the map of Victoria, Australia.

Usage

```
plot_vic_map(...)
```

Arguments

...	All arguments will be ignored.

Details

Require package sf installed.

Value

A ggplot object. The map of Victoria, Australia.

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  plot_vic_map()
}
```

---

print.spotoroo *Printing spatiotemporal clustering result*

Description

print.spotoroo() is the print method of the class spotoroo.

Usage

```
## S3 method for class 'spotoroo'
print(x, ...)
```

Arguments

x	spotoroo object. A result of a call to hotspot_cluster().
...	Additional arguments will be ignored.

Value

No return value, called for side effects

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
```

```
                          timeStep = 1)

# print the results
print(result)
```

---

spotoroo *spotoroo: spatiotemporal clustering in R of hot spot*

Description

A package for clustering satellite hot spots and detecting fire ignition points.

Authors

- Weihao Li <llreczx@gmail.com>
- Dianne Cook <dicook@monash.edu>
- Emily Dodwell <emdodwell@gmail.com>

Functions

The spotoroo package provides 13 important functions:

- hotspot_cluster()
- global_clustering()
- local_clustering()
- handle_noise()
- ignition_point()
- get_fire_mov()
- plot.spotoroo()
- plot_spotoroo()
- plot_def()
- plot_fire_mov()
- plot_timeline()
- plot_vic_map()
- summary.spotoroo()
- summary_spotoroo()
- print.spotoroo()
- transform_time_id()

- dist_point_to_vector()

The spotoroo package provides 2 external data objects:

- hotspots
- vic_map

---

summary.spotoroo *Summarizing spatiotemporal clustering result*

Description

summary.spotoroo() is the summary method of the class spotoroo. It is a simple wrapper of summary_spotoroo().

Usage

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

Arguments

object	spotoroo object. A result of a call to hotspot_cluster().
...	Additional arguments pass to summary_spotoroo()

Value

No return value, called for side effects

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)
```

```
# Make a summary
summary(result)
```

---

summary_spotoroo *Summarizing spatiotemporal clustering results*

Description

This function takes a spotoroo object to produce a summary of the clustering results. It can be called by summary.spotoroo().

Usage

```
summary_spotoroo(result, cluster = "all")
```

Arguments

result	spotoroo object. A result of a call to hotspot_cluster().
cluster	Character/Integer. If "all", summarize all clusters. If an integer vector is given, summarize corresponding clusters.

Value

No return value, called for side effects

Examples

```
# Time consuming functions (>5 seconds)


# Get clustering results
result <- hotspot_cluster(hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)
```

```
# Make a summary of all clusters
summary_spotoroo(result)

# Make a summary of cluster 1 to 3
summary_spotoroo(result, 1:3)
```

`transform_time_id` *Transforming a series of time or datetime to time indexes*

Description

This function transforms a series of time or datetime to time indexes.

Usage

```
transform_time_id(obsTime, timeUnit, timeStep)
```

Arguments

`obsTime`	Date/Datetime/Numeric. A vector of observed time of hot spots. If `timeUnit` is "n", `obsTime` needs to be a numeric vector, otherwise, it needs to be in date or datetime format.
`timeUnit`	Character. The unit of time, one of "s" (seconds), "m"(minutes), "h"(hours), "d"(days) and "n"(numeric).
`timeStep`	Numeric (>0). Number of units of `timeUnit` in a time step.

Details

The earliest time is assigned with a time index `1`. The difference between any other time to the earliest time is transformed using the `timeUnit` and divided by the `timeStep`. These differences are floored to integer and used as the time indexes.

Value

Integer. A vector of time indexes.

Examples

```
# Define obsTime
obsTime <- as.Date(c("2020-01-01",
                     "2020-01-02",
                     "2020-01-04"))

# Transform it to time index under different settings
transform_time_id(obsTime, "h", 1)
```

```
transform_time_id(obsTime, "m", 60)
transform_time_id(obsTime, "s", 3600)

# Define numeric obsTime
obsTime <- c(1,
             1.5,
             4.5,
             6)

# Transform it to time index under different settings
transform_time_id(obsTime, "n", 1)
transform_time_id(obsTime, "n", 1.5)
```

---

vic_map *simple features map of Victoria*

Description

A dataset containing the simple features of Victoria, Australia.

Usage

```
vic_map
```

Format

A "sf" object with 1 row.

Details

The dataset is obtained via the following codes:

```
library(rnaturalearth)
au_map <-ne_states(country = "Australia",returnclass = "sf")
vic_map <-au_map[7,]$geometry
```

Source

https://www.naturalearthdata.com/

Index