Package 'stdbscan'

March 14, 2026

Title Spatio-Temporal DBSCAN Clustering

Version 0.2.0

Description Implements the ST-DBSCAN (spatio-temporal density-based spatial clustering of applications with noise) clustering algorithm for detecting spatially and temporally dense regions in point data, with a fast C++ backend via 'Rcpp'. Birant and Kut (2007) <doi:10.1016/j.datak.2006.01.013>.

License GPL (>= 3)

Depends R (>= 3.5.0)

Encoding UTF-8

RoxygenNote 7.3.3

URL https://github.com/MiboraMinima/stdbscan/,
https://miboraminima.github.io/stdbscan/

BugReports https://github.com/MiboraMinima/stdbscan/issues/

LinkingTo Rcpp

Imports Rcpp, dbscan

Suggests knitr, rmarkdown, readr, testthat, ggplot2, plotly, covr, MetBrewer

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

Author Antoine Le Doeuff [aut, cre] (ORCID: <https://orcid.org/0009-0008-8807-3816>)

Maintainer Antoine Le Doeuff <antoine.ldoeuff@gmail.com>

Repository CRAN

Date/Publication 2026-03-14 15:40:02 UTC

Contents

`geolife_traj` *GPS pings from the GeoLife GPS Trajectories dataset*

Description

Extraction of the GeoLife GPS Trajectories dataset. The selected trajectory id is 000-20081023025304.

Data manipulation applied to the raw data :

- Conversion to EPSG:4586
- Manual selection of the pings
- Selection of relevant variables

Usage

```
geolife_traj
```

Format

A `data.frame` with one row per ping and the following columns:

- `date` (chr): The date
- `time` (chr): The time
- `x` (dbl): Longitude (EPSG:4586)
- `y` (dbl): Latitude (EPSG:4586)

Source

https://www.microsoft.com/en-us/download/details.aspx?id=52367

Examples

```
data(geolife_traj)
head(geolife_traj)
```

predict.stdbscan *Predict newdata*

Description

Assigns each new observation to an existing cluster from a fitted stdbscan object, or marks it as noise if it falls outside any cluster.

Usage

```
## S3 method for class 'stdbscan'
predict(object, data, newdata, ...)
```

Arguments

object	An object of class stdbscan.
data	matrix. The data set used to create the clustering object.
newdata	matrix. New data points for which the cluster membership should be predicted. The data must be in the same format as the input data.
...	Additional arguments are passed on to dbscan::frNN().

Value

An integer vector of cluster labels, matching the labels of the input stdbscan object.

Examples

```
data(geolife_traj)

geolife_traj$date_time <- as.POSIXct(
  paste(geolife_traj$date, geolife_traj$time),
  format = "%Y-%m-%d %H:%M:%S",
  tz = "GMT"
)
geolife_traj$t <- as.numeric(
  geolife_traj$date_time - min(geolife_traj$date_time)
)
data <- cbind(geolife_traj$x, geolife_traj$y, geolife_traj$t)

res <- st_dbscan(
  data = data,
  eps_spatial = 3,
  eps_temporal = 30,
  min_pts = 5
)

newdata <- cbind(
  c(440160,  440165,  440144,  440130,  440160),
```

```
  c(4428129, 4428135, 4428120, 4428123, 4428122),
  c(4617,    4620,    4629,    4635,    4640)
)
predict(res, data, newdata)
```

`st_dbscan` *Spatio-Temporal DBSCAN*

Description

Perform **ST-DBSCAN** clustering on points with spatial and temporal coordinates. This algorithm identifies clusters of points that are close both in space and time.

Usage

```
st_dbscan(data, eps_spatial, eps_temporal, min_pts, ...)
```

Arguments

`data`	matrix. A matrix containing, **in that order**, `x`, `y` and `t`. `x` (longitude) and `y` (latitude) are the spatial coordinates and `t` is the cumulative time since a common origin (*e.g.* `c(0, 6, 10)`). `t` must be sorted.
`eps_spatial`	Numeric. The spatial radius threshold. Points closer than this in space may belong to the same cluster.
`eps_temporal`	Numeric. The temporal threshold. Points closer than this in time may belong to the same cluster.
`min_pts`	Integer. Minimum number of points required to form a core point.
`...`	Additional arguments are passed on to `dbscan::frNN()` and `dbscan::dbscan()`.

Details

ST-DBSCAN extends classical DBSCAN by incorporating a temporal constraint. Two points are considered neighbors if they are within `eps_spatial` in space **and** within `eps_temporal` in time. Clusters are expanded from core points recursively following the DBSCAN algorithm.

ST-DBSCAN is implemented using the following approach:

1. Find the spatial neighbors using Fixed Radius Nearest Neighbors (`dbscan::frNN()`)
2. Filter the spatial neighbors by the temporal constraint
3. Apply DBSCAN on the filtered neighbors using `dbscan::dbscan()`

Value

`st_dbscan()` returns an object of class `stdbscan` with the following components:

`cluster`	Integer vector with cluster assignments. Zero indicates noise points.
`eps`	Value of the `eps_spatial` parameter.
`minPts`	Value of the `minPts` parameter.
`metric`	Used distance metric.
`borderPoints`	Whether border points are considered as noise (`FALSE`) or not (`TRUE`).
`eps_temporal`	Value of the `eps_temporal` parameter.

This class is a simple extension of the `dbscan` class. For more details, see `dbscan` documentation.

References

Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. Data & Knowledge Engineering, 60(1), 208–221. https://doi.org/10.1016/j.datak.2006.01.013

Examples

```
data(geolife_traj)

geolife_traj$date_time <- as.POSIXct(
  paste(geolife_traj$date, geolife_traj$time),
  format = "%Y-%m-%d %H:%M:%S",
  tz = "GMT"
)
geolife_traj$t <- as.numeric(
  geolife_traj$date_time - min(geolife_traj$date_time)
)
data <- cbind(geolife_traj$x, geolife_traj$y, geolife_traj$t)

st_dbscan(
  data = data,
  eps_spatial = 3,
  eps_temporal = 30,
  min_pts = 3,
  # Extra arguments
  splitRule = "STD",
  search = "kdtree",
  approx = 1
)
```

`st_dbscan_corepoint` *Check if points are core points*

Description

Check if data points are core points. A core point is a point with more than `min_pts` points in its neighborhood.

Usage

```
st_dbscan_corepoint(data, eps_spatial, eps_temporal, min_pts, ...)
```

Arguments

`data`	matrix. A matrix containing, **in that order**, `x`, `y` and `t`. `x` (longitude) and `y` (latitude) are the spatial coordinates and `t` is the cumulative time since a common origin (*e.g.* `c(0, 6, 10)`).
`eps_spatial`	Numeric. The spatial radius threshold. Points closer than this in space may belong to the same cluster.
`eps_temporal`	Numeric. The temporal threshold. Points closer than this in time may belong to the same cluster.
`min_pts`	Integer. Minimum number of points required to form a core point.
`...`	Additional arguments are passed on to `dbscan::frNN()`.

Value

A boolean vector indicating if data points are core points.

Examples

```
data(geolife_traj)

geolife_traj$date_time <- as.POSIXct(
  paste(geolife_traj$date, geolife_traj$time),
  format = "%Y-%m-%d %H:%M:%S",
  tz = "GMT"
)
geolife_traj$t <- as.numeric(
  geolife_traj$date_time - min(geolife_traj$date_time)
)
data <- cbind(geolife_traj$x, geolife_traj$y, geolife_traj$t)

res <- st_dbscan_corepoint(
  data = data,
  eps_spatial = 3,
  eps_temporal = 30,
  min_pts = 3
)
```

```
head(res)
```

Index