Package ‘rpostgisLT’

March 2, 2018

Title Managing Animal Movement Data with 'PostGIS' and R
Version 0.6.0
Date 2018-03-02
Description Integrates R and the 'PostgreSQL/PostGIS' database
system to build and manage animal trajectory (movement) data sets.
The package relies on 'ltraj' objects from the R package 'adehabitatLT',
building the analogous 'pgtraj' data structure in 'PostGIS'. Functions
allow users to seamlessly transfer between 'ltraj' and 'pgtraj', as
well as build new 'pgtraj' directly from location data stored in the
database.
SystemRequirements PostgreSQL with PostGIS extension
Depends R (>= 3.3.0), DBI, RPostgreSQL, rpostgis (>= 1.0.3),
    adehabitatLT (>= 0.3.12)
License GPL (>= 3)
URL https://github.com/mablab/rpostgisLT
BugReports https://github.com/mablab/rpostgisLT/issues
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
Imports sp, lubridate, shiny, leaflet, htmltools, mapview,
    shinyWidgets, magrittr, sf
Suggests knitr, raster, rmarkdown, testthat
VignetteBuilder knitr
NeedsCompilation no
Author Balázs Dukai [aut, cre] (Package creator during Google Summer of Code
    2016 and 2017),
    Mathieu Basille [aut],
    David Bucklin [aut],
    Clément Calenge [ctb]
Maintainer Balázs Dukai <balazs.dukai@gmail.com>
asPgtraj

Description

asPgtraj populates a pgtraj schema from the data provided in relocations_table. If the provided schema doesn’t exist, it will be created. On successful data input, asPgtraj creates two database views for each new pgtraj. These views are named parameters_<pgtraj_name>, step_geometry_<pgtraj_name> and described in more detail in the package vignette.

Usage

asPgtraj(conn, relocations_table, schema = "traj", pgtrajs = "pgtraj", animals = "animal", bursts = NULL, relocations, timestamps = NULL, rids = "rid", srid = NULL, tzone = NULL, note = NULL, clauses = NULL, info_cols = NULL, info_table = NULL, info_rids = NULL)

Arguments

conn Connection object created with RPostgreSQL
relocations_table String. Name of the schema and table that stores the relocations, e.g. c("schema","relocations")
schema String. Name of the schema that stores or will store the pgtraj data model (Default = "traj").
pgtrajs String. Name of the pgtraj or name of the field that stores the pgtraj names.
animals String. Name of the animal or name of the field that stores the animal names.
asPgtraj

bursts String. (Optional) name of the burst or name of the field that stores the burst names. If not given, each animal will have one burst.

relocations String. Name of the field that contains the relocations in relocations_table. Relocations can be provided either as columns names containing X,Y coordinates (e.g., c("x","y")) or a PostGIS geometry (e.g., "geom"). In both cases all relocations in relocations_table must have the same projection. If provided as coordinates in two columns, projection will be undefined unless srid is defined.

timestamps String. Name of the field in relocations_table that contains the timestamps. If NULL, Type I trajectory is assumed.

rids String. Name of the field in relocations_table that contains the numeric IDs of relocations. If timestamps = NULL, relocations will be sorted by the ascending numeric IDs in this field.

srid Integer. Optional SRID (spatial reference ID) of (x,y) coordinates provided for relocations. Ignored if relocations is a geometry type.

tzone String. Time zone specification for the timestamps column. If not specified, the database server time zone will be used (usually the server's local time zone).

note String. Comment on the pgtraj. The comment is only used in the database and not transferred into the ltraj.

clauses character, additional SQL to append to modify data selected from relocations_table. Must begin with WHERE . . . , and cannot contain ORDER BY or LIMIT clauses.

info_cols String. Optional character vector of database table column names storing additional information on relocations (replicating "infolocs" from the adehabitatLT object ltraj).

info_table Character vector of c("schema","table") holding the info_cols. If info_cols are in relocations_table, leave NULL.

info_rids String. Column name of unique integer ID in info_table to join with rids from relocations_table. If info_cols are in relocations_table, leave NULL.

Details

Opening and closing PostgreSQL connections have to be done manually by the user. However, the function checks if the provided connection is still valid.

Note that the arguments pgtrajs, animals, bursts, and note can refer to either a column name in relocations_table, or a string value. If the value is a column name, the values for the corresponding attribute (e.g., animals) will be the values from that column. When providing a string value, the value will be applied to that attribute for the entire pgtraj.

Burst names must be unique across a pgtraj. If it is not desired to further subset individual animal trajectories, leave bursts = NULL, in which case burst names will be equal to the animal name.

The time zone of the pgtraj is set to the local time zone of the user.

Value

TRUE on success
createShinyViews

Description

It is expected that *all* pgtrajes are projected in the schema in order to run.

Author(s)

Balázs Dukai <balazs.dukai@gmail.com>
David Bucklin <dbucklin@ufl.edu>

References

https://CRAN.R-project.org/package=adehabitatLT/vignettes/adehabitatLT.pdf

See Also

Section on pgtraj data model in the package vignette.

Examples

```r
## Not run:
asPgtraj(conn, 
  relocations_table = c("example_data","relocations_plus"), 
  pgtrajs = "id", 
  animals = "animal", 
  bursts = "burst", 
  relocations = "geom", 
  timestamps = "time", 
  rids = "gid", 
  note = "trajectories in 2015", 
  clauses = "WHERE extract(year FROM acquisition_time) = 2015", 
  info_cols = c("dist_to_road","land_cover","error_class")
)

## End(Not run)

## Not run:
asPgtraj(conn, 
  relocations_table = c("example_data","relocations_plus"), 
  schema = "traj_t4", 
  pgtrajs = "id", 
  animals = "animal", 
  bursts = "burst", 
  relocations = c("x","y"), 
  timestamps = "time", 
  rids = "gid")

## End(Not run)
```
explorePgtraj

Usage

createShinyViews(conn, schema, pgtraj, force = FALSE)

Arguments

conn       DBI::DBIConnection
schema     String. Schema name.
pgtraj     String. Pgtraj name.
force      Boolean. Drop and recreate the views if they already exist.

Value

nothing

Examples

## Not run:
createShinyViews(conn, schema="ibex_traj", pgtraj="ibex", force=TRUE)

## End(Not run)

explorePgtraj  Explore a pgtraj interactively in a Shiny app

Description

See vignette for details.

Usage

explorePgtraj(conn, schema, pgtraj, layer_vector = NULL,
               layer_param_vector = NULL, layer_raster = NULL,
               layer_param_raster = NULL)

Arguments

conn       DBI::DBIConnection
schema     String. Schema name of the pgtraj.
pgtraj     String. Pgtraj name.
layer_vector List of character vectors. As c(schema, table).
layer_param_vector Named list of lists. Names need to map to the table names in layer_vector. Sub-lists contain parameters passed to leaflet::add*. See example.
layer_raster RasterLayer object
layer_param_raster List. Parameters passed to leaflet::addRasterImage()
ltraj2pgtraj

Export ltraj object from R to a PostGIS database.

Description

ltraj2pgtraj exports an ltraj to the a database pgtraj, creating a new pgtraj schema if it doesn’t exist. The time zone and projection information stored in the ltraj is transferred to the database.

Usage

ltraj2pgtraj(conn, ltraj, schema = "ltraj", pgtraj = NULL, note = NULL, overwrite = FALSE, infolocs = TRUE)

Arguments

c conn A connection object.
ltraj An object of class ltraj.
schema Character. Name of the schema that stores or will store the pgtraj data model.
pgtraj2ltraj

pgtraj2ltraj( conn, pgtraj, schema = "traj" )

Arguments

- conn: Connection object created with RPostgreSQL
- pgtraj: String. Name of the pgtraj
- schema: String. Name of the schema storing the pgtraj

Value

TRUE on success.

Author(s)

Balázs Dukai <balazs.dukai@gmail.com>

See Also

asPgtraj to create a pgtraj with data already stored in the database.

Examples

## Not run:
# create pgtraj from ltraj "ibex" in schema "traj_t2"
ltraj2pgtraj(conn, ibex, "traj_t2")

## End(Not run)

---

Description

pgtraj2ltraj imports a single pgtraj from a database into an ltraj object.

Usage

pgtraj2ltraj(conn, pgtraj, schema = "traj")

Arguments

- conn: Connection object created with RPostgreSQL
- pgtraj: String. Name of the pgtraj
- schema: String. Name of the schema storing the pgtraj
pgtrajDrop

**Value**

an ltraj object

**Author(s)**

Balázs Dukai <balazs.dukai@gmail.com>

**Examples**

```r
## Not run:
# create ltraj from pgtraj named "ibex" in schema "traj_t2"
ibex<-pgtraj2ltraj(conn, "ibex", "traj_t2")

## End(Not run)
```

**Description**

pgtrajDrop deletes a pgtraj and/or all unused rows from a traj schema.

**Usage**

```r
pgtrajDrop(conn, pgtraj = NULL, schema = "traj", full_clean = TRUE)
```

**Arguments**

- `conn`: Connection object created with RPostgreSQL
- `pgtraj`: String. Name of the pgtraj (can be left NULL to perform `full_clean`)
- `schema`: String. Name of the schema storing the pgtraj
- `full_clean`: String. Whether to delete all unused rows in 'relocation' table. Should be done regularly if frequently overwriting many pgtraj, but note that it can take a long time to run.

**Value**

TRUE on success

**Author(s)**

Balázs Dukai <balazs.dukai@gmail.com>
Examples

```r
## Not run:
# drop "ibex" pgtraj in schema "traj"
pgtrajDrop(conn, "ibex")

# clean "traj" schema by deleting all unused rows in "relocation" table
pgtrajDrop(conn)

## End(Not run)
```

---

**pgtrajSchema**

*Check/create pgtraj schema.*

---

**Description**

Checks if the provided schema is a valid pgtraj schema, and creates one if it does not exist.

**Usage**

```r
pgtrajSchema(conn, schema = "traj")
```

**Arguments**

- **conn**: Connection object created with RPostgreSQL.
- **schema**: Character string. Name of the schema that stores or will store the pgtraj data model.

**Details**

Creates a schema to store pgtrajs in the database by calling a SQL script from ./sql/traj_schema.sql. The schema name defaults to traj. If a schema with the provided name already exists in the database, it checks if it contains all the required tables. The function does not attempt to repair the schema if all pgtraj tables are not present (e.g. because some were manually deleted). In this case, a new pgtraj schema needs to be created, or the existing schema needs to be deleted and recreated.

The function has its own standalone transaction control.

**Value**

`TRUE` if the schema exists (whether it was already available or was successfully created).

**Author(s)**

Balázs Dukai <balazs.dukai@gmail.com>
Examples

```r
## Not run:
# Check (or create) pgtraj schema with name "traj_1"
pgtrajSchema(conn,"traj_1")

## End(Not run)
```

---

**pgtrajVacuum**  
*VACUUM a pgtraj schema.*

---

**Description**

Performs a VACUUM (garbage-collect and optionally analyze) on all the tables of a traj schema.

**Usage**

```r
pgtrajVacuum(conn, schema = "traj", full = FALSE, verbose = FALSE, analyze = TRUE)
```

**Arguments**

- `conn`  
  Connection object created with RPostgreSQL

- `schema`  
  String. Name of the schema that stores or will store the pgtraj data model.

- `full`  
  Logical. Whether to perform a "full" vacuum, which can reclaim more space, but takes much longer and exclusively locks the table.

- `verbose`  
  Logical. Whether to print a detailed vacuum activity report for each table.

- `analyze`  
  Logical. Whether to update statistics used by the planner to determine the most efficient way to execute a query (default to TRUE).

**Value**

TRUE on success.

**Author(s)**

Balázs Dukai <balazs.dukai@gmail.com>

**Examples**

```r
## Not run:
# Vacuum analyze all tables in pgtraj schema with default name "traj"
pgtrajVacuum(conn)

## End(Not run)
```
Example data from a GPS tracking project

Description

Example datasets related to a GPS tracking project for roe deer in Trentino Region, Italy. Four datasets include raw data from GPS sensors (roe_gps_data), information on animals, sensors, and sensor deployments on animals (roe_sensors_animals_tables), and ancillary vector (roe_vector_geom) and raster (roe_raster) spatial datasets.

Usage

- roe_gps_data
- roe_sensors_animals_tables
- roe_vector_geom
- roe_raster

Format

roe_gps_data: A list containing five data.frames corresponding to five GPS sensors

- GSM01438 data frame for sensor 01438
- GSM01508 data frame for sensor 01508
- GSM01511 data frame for sensor 01511
- GSM01512 data frame for sensor 01512
- GSM02927 data frame for sensor 02927

roe_sensors_animals_tables: A list containing three data.frames

- animals data frame containing basic information on animals
- gps_sensors data frame containing basic information on GPS sensors
- gps_sensors_animals data frame containing information on deployment of GPS sensors on animals

roe_vector_geom: A list containing four Spatial*DataFrames

- study_area SpatialPolygonsDataFrame containing boundary of study area
- adm_boundaries SpatialPolygonsDataFrame containing administrative boundaries in study area
- meteo_stations SpatialPointsDataFrame containing locations of weather stations in study area
- roads SpatialLinesDataFrame containing representation of roads for study area

roe_raster: A list containing two RasterLayer datasets

- corine06 RasterLayer depicting land cover classification in the study area
- srtm_dem RasterLayer digital elevation model in the study area
**Source**


**Examples**

data("roe_gps_data")
head(roe_gps_data$GSM01438)
data("roe_sensors_animals_tables")
roe_sensors_animals_tables$animals
data("roe_vector_geom")
if (require(sp, quietly = TRUE)) {
  plot(roe_vector_geom$adm_boundaries)
  plot(roe_vector_geom$roads, col = 'red', add = TRUE)
}
if (require(raster, quietly = TRUE)) {
  data("roe_raster")
  plot(roe_raster$srtm_dem)
}

---

**rpostgisLT**

Integration of ltraj (adehabitatLT) and pgtraj (PostGIS).

---

**Description**

rpostgisLT

**Details**

The ‘rpostgisLT’ package develops the integration of R and PostGIS for managing movement trajectories. The focus is on streamlining the workflow for biologists to store and process animal trajectories in PostGIS and analyze them in R, thus utilizing the strengths of both software. The package relies on ‘ltraj’ objects from the R package ‘adehabitatLT’, and provides the analogous ‘pgtraj’ data structure in PostGIS, with all functions to create and manage ‘pgtraj’ data, and convert from and to both format (‘pgtraj’ in PostGIS, ‘ltraj’ in R). For a list of documented functions, use `library(help = "rpostgisLT")`

**Author(s)**

Balázs Dukai <balazs.dukai@gmail.com>
Index

*Topic datasets
  roe_gps_data, 11

asPgtraj, 2, 7
createShinyViews, 4
explorePgtraj, 5
ltraj2pgtraj, 6
pgtraj2ltraj, 7
pgtrajDrop, 8
pgtrajSchema, 9
pgtrajVacuum, 10

readTraj(pgtraj2ltraj), 7
roe_gps_data, 11
roe_raster (roe_gps_data), 11
roe_sensors_animals_tables
  (roe_gps_data), 11
roe_vector_geom (roe_gps_data), 11
rpostgisLT, 12
rpostgisLT-package (rpostgisLT), 12

writeTraj(ltraj2pgtraj), 6