Package ‘gtfs2gps’

June 12, 2020

Type Package

Title Converting Transport Data from GTFS Format to GPS-Like Records

Version 1.2-1

Date 2020-05-25

URL `https://github.com/ipeaGIT/gtfs2gps`

BugReports `https://github.com/ipeaGIT/gtfs2gps/issues`

Description Convert general transit feed specification (GTFS) data to global positioning system (GPS) records in 'data.table' format. It also has some functions to subset GTFS data in time and space and to convert both representations to simple feature format.

License MIT + file LICENSE

Encoding UTF-8

LazyData TRUE

Depends R (>= 3.5)

Suggests rmarkdown, knitr, testthat, dplyr

Imports data.table, furrr, future, magrittr, Rcpp, units, sf, sp, rgdal, rgeos, sfheaders, lwgeom, utils, raster, pbapply

RoxygenNote 7.1.0

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

Author Rafael H. M. Pereira [aut] (<https://orcid.org/0000-0003-2125-7465>), Pedro R. Andrade [aut, cre] (<https://orcid.org/0000-0001-8675-4046>), Joao Bazzo [aut] (<https://orcid.org/0000-0003-4536-5006>), Marcin Stepniak [ctb], Ipea - Institue for Applied Economic Research [cph, fnd]

Maintainer Pedro R. Andrade `<pedro.andrade@inpe.br>`

Repository CRAN

Date/Publication 2020-06-12 20:00:02 UTC

R topics documented:

`append_height` *Add a column with height to GPS data*

Description

Add a column named height to GPS data using a tif data as reference.

Usage

```
append_height(gps, heightfile)
```

Arguments

`gps`	A GPS data created from gtfs2gps().
`heightfile`	The pathname of a tif file with height data.

Value

The GPS data with a new column named height.

Examples

```
library(dplyr)

fortaleza <- system.file("extdata/fortaleza.zip", package = "gtfs2gps")
srtmfile <- system.file("extdata/fortaleza-srtm.tif", package = "gtfs2gps")

gtfs <- read_gtfs(fortaleza) %>%
  filter_week_days() %>%
  filter_single_trip() %>%
  remove_invalid()

fortaleza_gps <- gtfs2gps(gtfs, progress = FALSE) %>% append_height(srtmfile)
```

---

cpp_snap_points — *Snap points to the closest points from another set*

Description

Snap a set of points to the closest points available in another set of points.

Usage

```
cpp_snap_points(data, ref, spatial_resolution, id)
```

Arguments

`data`	A set of points to be snapped (a matrix). The result will have the same number of rows of this argument. Each row will return the respective snapped point.
`ref`	A set of reference points (another matrix). The result will be a subset of this parameter.
`spatial_resolution`	The spatial resolution of data, which means that from each point of data it is possible to reach at least one point within data with distance equals or less than spatial_resolution.
`id`	The id of the data to be shown in case of an error when a given point is more than [spatial_resolution ^ 4] meters away from the reference matrix.

Value

A data.frame with the snapped points.

`filter_by_agency_id` *Filter GTFS data by agency ids*

Description

Filter a GTFS data by its agency ids. It also removes the unnecessary routes, trips, frequencies, stop_times, calendars, shapes, and stops accordingly.

Usage

```
filter_by_agency_id(gtfs_data, agency_ids)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().
`agency_ids`	A vector of strings belonging to the agencies of the gtfs_data data.

Value

A filtered GTFS data.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))

result <- filter_by_agency_id(poa, "EPTC")
```

`filter_by_route_id` *Filter GTFS data by route ids*

Description

Filter a GTFS data by its route ids, subsetting routes and trips. It also removes the unnecessary stop_times, shapes, frequencies (if exist in a feed), and stops accordingly.

Usage

```
filter_by_route_id(gtfs_data, route_ids)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().
`route_ids`	A vector of route ids belonging to the routes of the gtfs_data data. Note that route_id might be loaded by gtfs2gps::read_gtfs() as a string or a number, depending on the available values.

Value

A filtered GTFS data.

Examples

```
warsaw <- read_gtfs(system.file("extdata/warsaw.zip", package="gtfs2gps"))

subset <- filter_by_route_id(warsaw, c("15", "175"))
```

`filter_by_route_type` *Filter GTFS data by transport mode (route type)*

Description

Filter a GTFS data by transport mode (coded in the column route_type in routes.txt). It also removes the unnecessary trips, stop_times, shapes, frequencies (if exist in a feed), and stops accordingly.

Usage

```
filter_by_route_type(gtfs_data, route_types)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().
`route_types`	A vector of route types belonging to the routes of the gtfs_data data. Note that route_type might be loaded by gtfs2gps::read_gtfs() as a string or a number, depending on the available values.

Value

A filtered GTFS data.

Examples

```
warsaw <- read_gtfs(system.file("extdata/warsaw.zip", package="gtfs2gps"))

subset <- filter_by_route_type(warsaw, c(0, 3))
```

filter_by_shape_id *Filter GTFS data by shape ids*

Description

Filter a GTFS data by its shape ids. It also removes the unnecessary trips, stop_times, stops, and routes accordingly.

Usage

```
filter_by_shape_id(gtfs_data, shape_ids)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().
`shape_ids`	A vector of shape_ids belonging to the shapes of the gtfs_data data. Note that shape_id might be loaded by gtfs2gps::read_gtfs() as a string or a number, depending on the available values.

Value

A filtered GTFS data.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))

subset <- filter_by_shape_id(poa, "T2-1")
```

filter_day_period *Filter GTFS data within a period of the day*

Description

Updates a GTFS feed filtering only the routes, shapes, trips, stops, agencies and services that are active within a given period of the day.

Usage

```
filter_day_period(gtfs, period_start = "00:00:01", period_end = "23:59:59")
```

Arguments

`gtfs`	A GTFS data.
`period_start`	A string of type "hh:mm" indicating start of the period (defaults to "00:00:01")
`period_end`	A string of type "hh:mm" indicating the end of the period (defaults to "23:59:59")

Value

A filtered GTFS data.

Examples

```
# read gtfs data
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))

# filter gtfs data
poa_f <- filter_day_period(poa, period_start = "10:00", period_end = "10:20")
```

`filter_single_trip` *Filter GTFS trips in order to have one trip per shape_id*

Description

Filter a GTFS data by keeping only one trip per shape_id. It also removes the unnecessary routes accordingly.

Usage

```
filter_single_trip(gtfs_data)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().

Value

A filtered GTFS data.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))

subset <- filter_single_trip(poa)
```

filter_valid_stop_times *Filter GTFS data using valid stop times*

Description

Filter a GTFS data read using gtfs2gps::read_gtfs(). It removes stop_times with NA values in arrival_time, departure_time, and arrival_time_hms. It also filters stops and routes accordingly.

Usage

```
filter_valid_stop_times(gtfs_data)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().

Value

A filtered GTFS data.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))

subset <- filter_valid_stop_times(poa)
```

filter_week_days *Filter GTFS trips operating on week days*

Description

Filter a GTFS data read using gtfs2gps::read_gtfs(). It removes the trips operating only saturday or sunday.

Usage

```
filter_week_days(gtfs_data)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().

Value

A filtered GTFS data.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))

subset <- filter_week_days(poa)
```

`gps_as_sflinestring` *Converts a GPS-like data.table to a LineString Simple Feature (sf) object*

Description

Every interval of GPS data points between stops for each trip_id is converted into a linestring segment. The output assumes constant average speed between consecutive stops.

Usage

```
gps_as_sflinestring(gps, crs = 4326)
```

Arguments

`gps`	A data.table with timestamp data.
`crs`	A Coordinate Reference System. The default value is 4326 (latlong WGS84).

Value

A simple feature (sf) object with LineString data.

Examples

```
library(gtfs2gps)
library(dplyr)

poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))
poa_subset <- filter_by_shape_id(poa, c("T2-1", "A141-1")) %>%
  filter_single_trip()

poa_gps <- gtfs2gps(poa_subset)

poa_gps_sf <- gps_as_sflinestring(poa_gps)
```

gps_as_sfpoints — *Convert GPS-like data.table to a Simple Feature points object*

Description

Convert a GPS data stored in a data.table into Simple Feature points.

Usage

```
gps_as_sfpoints(gps, crs = 4326)
```

Arguments

gps	A data.table with timestamp data.
crs	A Coordinate Reference System. The default value is 4326 (latlong WGS84).

Value

A simple feature (sf) object with point data.

Examples

```
library(gtfs2gps)
library(dplyr)

fortaleza <- read_gtfs(system.file("extdata/fortaleza.zip", package = "gtfs2gps"))
srtmfile <- system.file("extdata/fortaleza-srtm.tif", package="gtfs2gps")

subset <- fortaleza %>%
  filter_week_days() %>%
  filter_single_trip() %>%
  filter_by_shape_id(c("shape804-I", "shape806-I"))

for_gps <- gtfs2gps(subset)
for_gps_sf_points <- gps_as_sfpoints(for_gps) # without height

for_gps <- append_height(for_gps, srtmfile)
for_gps_sf_points <- gps_as_sfpoints(for_gps) # with height
```

gtfs2gps *Convert GTFS to GPS-like data given a spatial resolution*

Description

Convert GTFS data to GPS format by sampling points using a spatial resolution. This function creates additional points in order to guarantee that two points in a same trip will have at most a given distance, indicated as a spatial resolution.

Usage

```
gtfs2gps(
  gtfs_data,
  spatial_resolution = 50,
  parallel = FALSE,
  strategy = "multiprocess",
  progress = TRUE,
  filepath = NULL,
  continue = FALSE
)
```

Arguments

`gtfs_data`	A path to a GTFS file to be converted to GPS, or a GTFS data represented as a list of data.tables.
`spatial_resolution`	The spatial resolution in meters. Default is 50m.
`parallel`	Decides whether the function should run in parallel. Defaults to FALSE. When TRUE, it will use all cores available minus one.
`strategy`	Name of evaluation function to use in future parallel processing. Defaults to "multiprocess", i.e. if multicore evaluation is supported, that will be used, otherwise multisession evaluation will be used. Fore details, check ?future::plan().
`progress`	Show a progress bar. Default is TRUE.
`filepath`	Output file path. As default, the output is returned in R. When this argument is set, each route is saved into a file within filepath, with the name equals to its id. In this case, no output is returned.
`continue`	Argument that can be used only with filepath. When TRUE, it skips processing the shape identifiers that were already saved into files. It is useful to continue processing a GTFS file that was stopped for some reason. Default value is FALSE.

Value

A data.table, where each row represents a GPS point. The following columns are returned (units of measurement in parenthesis): dist and cumdist (meters), cumtime (seconds), shape_pt_lon and shape_pt_lat (degrees), speed (km/h), departure_time (hh:mm:ss), .

Examples

```
library(dplyr)
poa <- read_gtfs(system.file("extdata/poa.zip", package="gtfs2gps"))
subset <- filter_by_shape_id(poa, "T2-1") %>%
  filter_single_trip()

poa_gps <- gtfs2gps(subset)
```

---

gtfs_shapes_as_sf *Convert GTFS shapes to simple feature object*

Description

Convert a GTFS shapes data loaded using gtfs2gps::read_gtf() into a line simple feature (sf).

Usage

```
gtfs_shapes_as_sf(gtfs, crs = 4326)
```

Arguments

gtfs	A GTFS data.
crs	The coordinate reference system represented as an EPSG code. The default value is 4326 (latlong WGS84)

Value

A simple feature (sf) object.

Examples

```
poa <- read_gtfs(system.file("extdata/saopaulo.zip", package = "gtfs2gps"))
poa_sf <- gtfs_shapes_as_sf(poa)
```

---

gtfs_stops_as_sf *Convert GTFS stops to simple feature object*

Description

Convert a GTFS stops data loaded using gtfs2gps::read_gtf() into a point simple feature (sf).

Usage

```
gtfs_stops_as_sf(gtfs, crs = 4326)
```

Arguments

`gtfs`	A GTFS data.
`crs`	The coordinate reference system represented as an EPSG code. The default value is 4326 (latlong WGS84)

Value

A simple feature (sf) object.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))
poa_shapes <- gtfs_shapes_as_sf(poa)
poa_stops <- gtfs_stops_as_sf(poa)
```

`merge_gtfs_feeds` *Merge multiple GTFS feeds into a single one*

Description

Build a single GTFS by joinning together the elements of multiple GTFS feeds.

Usage

```
merge_gtfs_feeds(gtfs_list)
```

Arguments

`gtfs_list`	A list of GTFS.zip files.

Value

A single list of data.tables, where each index represents the respective GTFS file name.

Examples

```

# get a list of GTFS feeds
spo <- system.file("extdata/saopaulo.zip", package = "gtfs2gps")
poa <- system.file("extdata/poa.zip", package = "gtfs2gps")
gtfs_list <- list(spo, poa)

new_gtfs <- merge_gtfs_feeds(gtfs_list)
```

read_gtfs *Read GTFS data into a list of data.tables*

Description

Read files of a zipped GTFS feed and load them to memory as a list of data.tables. It will load the following files: "agency.txt", "calendar.txt", "shapes.txt", "routes.txt", "shapes.txt", "stop_times.txt", "stops.txt", "trips.txt", and "frequencies.txt", with this last one being optional. If one of the mandatory files does not exit, this function will stop with an error message.

Usage

```
read_gtfs(gtfszip)
```

Arguments

gtfszip	A zipped GTFS data.

Value

A list of data.tables, where each index represents the respective GTFS file name.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))
```

remove_invalid *Remove invalid objects from GTFS data*

Description

Remove all objects from GTFS data that are not used in all relations that they are required to be. That is, agency-routes relation (agency_id), routes-trips relation (route_id), trips-shapes relation (shape_id), trips-frequencies relation (trip_id), trips-stop_times relation (trip_id), stop_times-stops relation (stop_id), and trips-calendar relation (service_id), recursively, until GTFS data does not reduce its size anymore. For example, if one agency_id belongs to routes but not to agency will be removed. This might cause one cascade removal of objects in other relations that originally did not have any inconsistency.

Usage

```
remove_invalid(gtfs_data, only_essential = TRUE, prompt_invalid = FALSE)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().
`only_essential`	Remove only the essential files? The essential files are all but agency and calendar. Default is TRUE, which means that agency-routes and trips-calendar relations will not be processed as restrictions to remove objects.
`prompt_invalid`	Show the invalid objects. Default is FALSE.

Value

A subset of the input GTFS data.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))
object.size(poa)
subset <- remove_invalid(poa)
object.size(subset)
```

`simplify_shapes` *Simplify shapes of a GTFS file*

Description

Remove points from the shapes of a GTFS file in order to reduce its size. It uses Douglas-Peucker algotithm internally.

Usage

```
simplify_shapes(gtfs_data, tol = 0)
```

Arguments

`gtfs_data`	A list of data.tables read using gtfs2gps::reag_gtfs().
`tol`	Numerical tolerance value to be used by the Douglas-Peuker algorithm. The default value is 0, which means that no data will be lost.

Value

A GTFS data whose shapes is a subset of the input data.

Examples

```
poa <- read_gtfs(system.file("extdata/poa.zip", package="gtfs2gps"))

poa_simpl <- simplify_shapes(poa)
```

test_gtfs_freq *Test whether a GTFS feed is frequency based*

Description

Test whether a GTFS feed is frequency based or whether it presents detailed time table for all routes and trip ids.

Usage

```
test_gtfs_freq(gtfs)
```

Arguments

gtfs	A GTFS data set stored in memory as a list of data.tables/data.frames.

Value

A string "frequency" or "simple".

Examples

```
# read a gtfs.zip to memory
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps"))

# Test whether a GTFS feed is frequency based
test_gtfs_freq(poa)
```

write_gtfs *Write GTFS data into a zip file*

Description

Write GTFS stored in memory as a list of data.tables into a zipped GTFS feed. This function overwrites the zip file if it exists.

Usage

```
write_gtfs(gtfs, zipfile)
```

Arguments

gtfs	A GTFS data set stored in memory as a list of data.tables/data.frames.
zipfile	The pathname of a .zip file to be saved with the GTFS data.

Value

The status value returned by the external zip command, invisibly.

Examples

```
library(dplyr)

# read a gtfs.zip to memory
poa <- read_gtfs(system.file("extdata/poa.zip", package = "gtfs2gps")) %>%
  filter_by_shape_id("T2-1") %>%
  filter_single_trip()

# write GTFS data into a zip file
write_gtfs(poa, paste0(tempdir(), "/mypoa.zip"))
```

Index