Package 'gtfstools'

November 16, 2021

Type Package

Title General Transit Feed Specification (GTFS) Editing and Analysing Tools

Version 1.0.0

Description Utility functions to read, manipulate, analyse and write transit feeds in the General Transit Feed Specification (GTFS) data format.

License MIT + file LICENSE

URL https://ipeagit.github.io/gtfstools/, https://github.com/ipeaGIT/gtfstools

BugReports https://github.com/ipeaGIT/gtfstools/issues

Depends R (>= 2.10)

Imports checkmate, data.table, gtfsio (>= 1.0.0), sf, sfheaders, units, utils

Suggests covr, ggplot2, knitr, lwgeom, rmarkdown, testthat, zip

LinkingTo cpp11

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.1.1

Author Daniel Herszenhut [aut, cre] (<https://orcid.org/0000-0001-8066-1105>), Rafael H. M. Pereira [aut] (<https://orcid.org/0000-0003-2125-7465>), Pedro R. Andrade [aut] (<https://orcid.org/0000-0001-8675-4046>), Joao Bazzo [aut] (<https://orcid.org/0000-0003-4536-5006>), Mark Padgham [ctb], Ipea - Institute for Applied Economic Research [cph, fnd]

Maintainer Daniel Herszenhut <dhersz@gmail.com>

Repository CRAN

Date/Publication 2021-11-16 07:50:11 UTC

R topics documented:

convert_shapes_to_sf . . . 2
convert_stops_to_sf . . . 3
filter_by_route_id . . . 4
filter_by_route_type . . . 5
filter_by_sf . . . 6
filter_by_shape_id . . . 7
filter_by_stop_id . . . 8
filter_by_trip_id . . . 9
get_parent_station . . . 10
get_trip_duration . . . 11
get_trip_geometry . . . 12
get_trip_segment_duration . . . 13
get_trip_speed . . . 14
gtfstools . . . 15
merge_gtfs . . . 16
read_gtfs . . . 17
remove_duplicates . . . 19
set_trip_speed . . . 19
validate_gtfs . . . 21
write_gtfs . . . 22

Index **24**

`convert_shapes_to_sf` *Convert* `shapes` *table to simple feature object*

Description

Converts the `shapes` table to a LINESTRING sf object.

Usage

```
convert_shapes_to_sf(gtfs, shape_id = NULL, crs = 4326)
```

Arguments

`gtfs`	A GTFS object.
`shape_id`	A character vector including the `shape_id`s to be converted. If `NULL` (the default), all shapes are converted.
`crs`	The CRS of the resulting object, either as an EPSG code or as an `crs` object. Defaults to 4326 (WGS 84).

Value

A LINESTRING sf object.

Examples

```
# read gtfs
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)

shapes_sf <- convert_shapes_to_sf(gtfs)
head(shapes_sf)

shapes_sf <- convert_shapes_to_sf(gtfs, shape_id = "17846")
shapes_sf
```

---

convert_stops_to_sf *Convert* stops *table to simple feature object*

Description

Converts the stops table to a POINT sf object.

Usage

```
convert_stops_to_sf(gtfs, stop_id = NULL, crs = 4326)
```

Arguments

gtfs	A GTFS object.
stop_id	A character vector including the stop_ids to be converted. If NULL (the default), all stops are converted.
crs	The CRS of the resulting object, either as an EPSG code or as an crs object. Defaults to 4326 (WGS 84).

Value

A POINT sf object.

Examples

```
# read gtfs
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)

stops_sf <- convert_stops_to_sf(gtfs)
head(stops_sf)

stops_sf <- convert_stops_to_sf(gtfs, stop_id = "18848")
stops_sf
```

filter_by_route_id *Filter GTFS object by* route_id

Description

Filters a GTFS object by route_ids, keeping (or dropping) the relevant entries in each file.

Usage

```
filter_by_route_id(gtfs, route_id, keep = TRUE)
```

Arguments

gtfs	A GTFS object.
route_id	A character vector. The route_ids used to filter the data.
keep	A logical. Whether the entries related to the specified route_ids should be kept or dropped (defaults to TRUE, which keeps the entries).

Value

The GTFS object passed to the gtfs parameter, after the filtering process.

See Also

Other filtering functions: filter_by_route_type(), filter_by_sf(), filter_by_shape_id(), filter_by_stop_id(), filter_by_trip_id()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)
route_ids <- c("6450-51", "CPTM L11")

object.size(gtfs)

# keeps entries related to passed route_ids
smaller_gtfs <- filter_by_route_id(gtfs, route_ids)
object.size(smaller_gtfs)

# drops entries related to passed route_ids
smaller_gtfs <- filter_by_route_id(gtfs, route_ids, keep = FALSE)
object.size(smaller_gtfs)
```

filter_by_route_type *Filter GTFS object by* route_type *(transport mode)*

Description

Filters a GTFS object by route_types, keeping (or dropping) the relevant entries in each file.

Usage

```
filter_by_route_type(gtfs, route_type, keep = TRUE)
```

Arguments

gtfs	A GTFS object.
route_type	An integer vector. The route_types used to filter the data.
keep	A logical. Whether the entries related to the specified route_types should be kept or dropped (defaults to TRUE, which keeps the entries).

Value

The GTFS object passed to the gtfs parameter, after the filtering process.

Route types

Valid options are:

- 0 - Tram, Streetcar, Light rail. Any light rail or street level system within a metropolitan area.
- 1 - Subway, Metro. Any underground rail system within a metropolitan area.
- 2 - Rail. Used for intercity or long-distance travel.
- 3 - Bus. Used for short- and long-distance bus routes.
- 4 - Ferry. Used for short- and long-distance boat service.
- 5 - Cable tram. Used for street-level rail cars where the cable runs beneath the vehicle, e.g., cable car in San Francisco.
- 6 - Aerial lift, suspended cable car (e.g., gondola lift, aerial tramway). Cable transport where cabins, cars, gondolas or open chairs are suspended by means of one or more cables.
- 7 - Funicular. Any rail system designed for steep inclines.
- 11 - Trolleybus. Electric buses that draw power from overhead wires using poles.
- 12 - Monorail. Railway in which the track consists of a single rail or a beam.

See Also

Other filtering functions: filter_by_route_id(), filter_by_sf(), filter_by_shape_id(), filter_by_stop_id(), filter_by_trip_id()

Examples

```
# read gtfs
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)

object.size(gtfs)

# keeps entries related to passed route_types
smaller_gtfs <- filter_by_route_type(gtfs, route_type = 1)
object.size(smaller_gtfs)

# drops entries related to passed route_types
smaller_gtfs <- filter_by_route_type(gtfs, route_type = 1, keep = FALSE)
object.size(smaller_gtfs)
```

---

filter_by_sf *Filter a GTFS object using a* simple features *object*

Description

Filters a GTFS object using the geometry of an `sf` object, keeping (or dropping) entries related to shapes and trips selected through a spatial operation.

Usage

```
filter_by_sf(gtfs, geom, spatial_operation = sf::st_intersects, keep = TRUE)
```

Arguments

`gtfs`	A GTFS object.
`geom`	An `sf` object. Describes the geometry used to filter the data.
`spatial_operation`	A spatial operation function from the set of options listed in geos_binary_pred (check the DE-I9M Wikipedia entry for the definition of each function). Defaults to `sf::st_intersects`, which tests if the shapes and trips have ANY intersection with the object specified in `geom`. Please note that `geom` is passed as the `x` argument of these functions.
`keep`	A logical. Whether the entries related to the shapes and trips that cross through the given geometry should be kept or dropped (defaults to `TRUE`, which keeps the entries).

Value

The GTFS object passed to the `gtfs` parameter, after the filtering process.

See Also

Other filtering functions: filter_by_route_id(), filter_by_route_type(), filter_by_shape_id(), filter_by_stop_id(), filter_by_trip_id()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)

shape_id <- "68962"
shape_sf <- convert_shapes_to_sf(gtfs, shape_id)
bbox <- sf::st_bbox(shape_sf)
object.size(gtfs)

# keeps entries that intersect with the specified polygon
smaller_gtfs <- filter_by_sf(gtfs, bbox)
object.size(smaller_gtfs)

# drops entries that intersect with the specified polygon
smaller_gtfs <- filter_by_sf(gtfs, bbox, keep = FALSE)
object.size(smaller_gtfs)

# uses a different function to filter the gtfs
smaller_gtfs <- filter_by_sf(gtfs, bbox, spatial_operation = sf::st_contains)
object.size(smaller_gtfs)
```

---

filter_by_shape_id *Filter GTFS object by* shape_id

Description

Filters a GTFS object by shape_ids, keeping (or dropping) the relevant entries in each file.

Usage

```
filter_by_shape_id(gtfs, shape_id, keep = TRUE)
```

Arguments

gtfs	A GTFS object.
shape_id	A character vector. The shape_ids used to filter the data.
keep	A logical. Whether the entries related to the specified shape_ids should be kept or dropped (defaults to TRUE, which keeps the entries).

Value

The GTFS object passed to the gtfs parameter, after the filtering process.

See Also

Other filtering functions: filter_by_route_id(), filter_by_route_type(), filter_by_sf(), filter_by_stop_id(), filter_by_trip_id()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)
shape_ids <- c("17846", "68962")

object.size(gtfs)

# keeps entries related to passed shape_ids
smaller_gtfs <- filter_by_shape_id(gtfs, shape_ids)
object.size(smaller_gtfs)

# drops entries related to passed shape_ids
smaller_gtfs <- filter_by_shape_id(gtfs, shape_ids, keep = FALSE)
object.size(smaller_gtfs)
```

---

filter_by_stop_id *Filter GTFS object by* stop_id

Description

Filters a GTFS object by stop_ids, keeping (or dropping) relevant entries in each file. In order to keep the integrity of trips as described in the stop_times table, the stop_ids are actually used to filter trip_ids, which are then used to filter the rest of the GTFS object.

Usage

```
filter_by_stop_id(gtfs, stop_id, keep = TRUE)
```

Arguments

gtfs	A GTFS object.
stop_id	A character vector. The stop_ids used to filter the data.
keep	A logical. Whether the entries related to the trip_ids that passes through the specified stop_ids should be kept or dropped (defaults to TRUE, which keeps the entries).

Value

The GTFS object passed to the gtfs parameter, after the filtering process.

See Also

Other filtering functions: filter_by_route_id(), filter_by_route_type(), filter_by_sf(), filter_by_shape_id(), filter_by_trip_id()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)
stop_ids <- c("18848", "940004157")

object.size(gtfs)

# keeps entries related to trips that pass through specified stop_ids
smaller_gtfs <- filter_by_stop_id(gtfs, stop_ids)
object.size(smaller_gtfs)

# drops entries related to trips that pass through specified stop_ids
smaller_gtfs <- filter_by_stop_id(gtfs, stop_ids, keep = FALSE)
object.size(smaller_gtfs)
```

---

filter_by_trip_id *Filter GTFS object by* trip_id

Description

Filters a GTFS object by trip_ids, keeping (or dropping) the relevant entries in each file.

Usage

```
filter_by_trip_id(gtfs, trip_id, keep = TRUE)
```

Arguments

gtfs	A GTFS object.
trip_id	A character vector. The trip_ids used to filter the data.
keep	A logical. Whether the entries related to the specified trip_ids should be kept or dropped (defaults to TRUE, which keeps the entries).

Value

The GTFS object passed to the gtfs parameter, after the filtering process.

See Also

Other filtering functions: filter_by_route_id(), filter_by_route_type(), filter_by_sf(), filter_by_shape_id(), filter_by_stop_id()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)
trip_ids <- c("CPTM L07-0", "2002-10-0")

object.size(gtfs)

# keeps entries related to passed trip_ids
smaller_gtfs <- filter_by_trip_id(gtfs, trip_ids)
object.size(smaller_gtfs)

# drops entries related to passed trip_ids
smaller_gtfs <- filter_by_trip_id(gtfs, trip_ids, keep = FALSE)
object.size(smaller_gtfs)
```

---

get_parent_station *Get parent stations recursively*

Description

Returns the (recursive) parent stations of each specified `stop_id`. Recursive in this context means it returns all parents' parents (i.e. first parents, then parents' parents, and then their parents, and so on).

Usage

```
get_parent_station(gtfs, stop_id)
```

Arguments

`gtfs`	A GTFS object as created by `read_gtfs`.
`stop_id`	A string vector including the `stop_id`s to have their parents returned.

Value

A `data.table` containing the `stop_id`s and their `parent_station`s. If a stop doesn't have a parent, its `parent_station` is `""`.

Examples

```
data_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)

parents <- get_parent_station(gtfs, "N3")
parents

parents <- get_parent_station(gtfs, c("B1", "B2"))
```

parents

---

get_trip_duration *Get trip duration*

Description

Returns the duration of each specified trip_id.

Usage

```
get_trip_duration(gtfs, trip_id = NULL, unit = "min")
```

Arguments

gtfs	A GTFS object as created by read_gtfs.
trip_id	A string vector including the trip_ids to have their duration calculated. If NULL (the default) the function calculates the duration of every trip_id in the GTFS.
unit	A string representing the time unit in which the duration is desired. One of "s" (seconds), "min" (minutes, the default), "h" (hours) or "d" (days).

Value

A data.table containing the duration of each specified trip.

Details

The duration of a trip is defined as the time difference between its last arrival time and its first departure time, as specified in the stop_times file.

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)

trip_duration <- get_trip_duration(gtfs)
head(trip_duration)

trip_ids <- c("CPTM L07-0", "2002-10-0")
trip_duration <- get_trip_duration(gtfs, trip_id = trip_ids)
trip_duration

trip_duration <- get_trip_duration(gtfs, trip_id = trip_ids, unit = "h")
trip_duration
```

get_trip_geometry *Get trip geometry*

Description

Returns the geometry of each specified `trip_id`, based either on the `shapes` or the `stop_times` file (or both).

Usage

```
get_trip_geometry(gtfs, trip_id = NULL, file = NULL, crs = 4326)
```

Arguments

`gtfs`	A GTFS object.
`trip_id`	A character vector including the `trip_id`s to have their geometries generated. If `NULL` (the default), the function generates geometries for every `trip_id` in the GTFS.
`file`	A character vector specifying the file from which geometries should be generated (either one of or both `shapes` and `stop_times`). If `NULL` (the default), the function attemps to generate the geometries from both files, but only raises an error if none of the files exist.
`crs`	The CRS of the resulting object, either as an EPSG code or as an `crs` object. Defaults to 4326 (WGS 84).

Value

A LINESTRING sf.

Details

The geometry generation works differently for the two files. In the case of `shapes`, the shape as described in the text file is converted to an `sf` object. For `stop_times` the geometry is the result of linking subsequent stops along a straight line (stops' coordinates are retrieved from the `stops` file). Thus, the resolution of the geometry when generated with `shapes` tends to be much higher than when created with `stop_times`.

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)

trip_geometry <- get_trip_geometry(gtfs)
head(trip_geometry)

# the above is identical to
```

```
trip_geometry <- get_trip_geometry(gtfs, file = c("shapes", "stop_times"))
head(trip_geometry)

trip_ids <- c("CPTM L07-0", "2002-10-0")
trip_geometry <- get_trip_geometry(gtfs, trip_id = trip_ids)
trip_geometry
plot(trip_geometry["origin_file"])
```

`get_trip_segment_duration` *Get trip segments' duration*

Description

Returns the duration of segments between stops of each specified `trip_id`.

Usage

```
get_trip_segment_duration(gtfs, trip_id = NULL, unit = "min")
```

Arguments

`gtfs`	A GTFS object as created by `read_gtfs`.
`trip_id`	A string vector including the `trip_id`s to have their segments' duration calculated. If `NULL` (the default) the function calculates the segments' duration of every `trip_id` in the GTFS.
`unit`	A string representing the time unit in which the duration is desired. One of `"s"` (seconds), `"min"` (minutes, the default), `"h"` (hours) or `"d"` (days).

Value

A `data.table` containing the segments' duration of each specified trip.

Details

A trip segment is defined as the path between two subsequent stops in the same trip. The duration of a segment is defined as the time difference between its arrival time and its departure time, as specified in the `stop_times` file.

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)

trip_segment_dur <- get_trip_segment_duration(gtfs)
head(trip_segment_dur)
```

```
trip_segment_dur <- get_trip_segment_duration(gtfs, trip_id = "CPTM L07-0")
trip_segment_dur

trip_segment_dur <- get_trip_segment_duration(gtfs, "CPTM L07-0", unit = "s")
trip_segment_dur
```

`get_trip_speed` *Get trip speed*

Description

Returns the speed of each specified `trip_id`, based on the geometry created from either the `shapes` or the `stop_times` file (or both).

Usage

```
get_trip_speed(gtfs, trip_id = NULL, file = "shapes", unit = "km/h")
```

Arguments

`gtfs`	A GTFS object.
`trip_id`	A character vector including the `trip_id`s to have their speeds calculated. If `NULL` (the default), the function calculates the speed of every `trip_id` in the GTFS.
`file`	The file from which geometries should be generated, either `shapes` or `stop_times` (geometries are used to calculate the length of a trip). Defaults to `shapes`.
`unit`	A string representing the unit in which the speeds are desired. Either `"km/h"` (the default) or `"m/s"`.

Value

A `data.table` containing the duration of each specified trip and the file from which geometries were generated.

Details

Please check `get_trip_geometry()` documentation to understand how geometry generation differs depending on the chosen file.

See Also

`get_trip_geometry()`

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)

# the examples below require the 'lwgeom' package to be installed
if (requireNamespace("lwgeom", quietly = TRUE)) {

  trip_speed <- get_trip_speed(gtfs)
  head(trip_speed)

  trip_ids <- c("CPTM L07-0", "2002-10-0")
  trip_speed <- get_trip_speed(gtfs, trip_ids)
  trip_speed

  trip_speed <- get_trip_speed(
    gtfs,
    trip_ids,
    file = c("shapes", "stop_times")
  )
  trip_speed

  trip_speed <- get_trip_speed(gtfs, trip_ids, unit = "m/s")
  trip_speed

}
```

`gtfstools` *gtfstools: General Transit Feed Specification (GTFS) Editing and Analysing Tools*

Description

Utility functions to read, manipulate, analyse and write transit feeds in the General Transit Feed Specification (GTFS) data format.

Usage

Please check the vignettes for more on the package usage:

- Basic usage: reading, analysing, manipulating and writing feeds. Run `vignette("gtfstools")` or check it on the website.
- Filtering GTFS feeds. Run `vignette("filtering", package = "gtfstools")` or check it on the website.

Author(s)

Maintainer: Daniel Herszenhut <dhersz@gmail.com> (ORCID)

Authors:

- Rafael H. M. Pereira <rafa.pereira.br@gmail.com> (ORCID)
- Pedro R. Andrade <pedro.andrade@inpe.br> (ORCID)
- Joao Bazzo (ORCID)

Other contributors:

- Mark Padgham <mark.padgham@email.com> [contributor]
- Ipea - Institute for Applied Economic Research [copyright holder, funder]

See Also

Useful links:

- https://ipeagit.github.io/gtfstools/
- https://github.com/ipeaGIT/gtfstools
- Report bugs at https://github.com/ipeaGIT/gtfstools/issues

`merge_gtfs` *Merge GTFS files*

Description

Combines many GTFS file into a single one.

Usage

```
merge_gtfs(..., files = NULL, quiet = TRUE, warnings = TRUE)
```

Arguments

`...`	GTFS objects, as created by `read_gtfs`, to be merged. Each argument can either be a GTFS or a list of GTFS objects.
`files`	A character vector listing the GTFS text files (i.e. the ones represented by `data.tables`) to be merged. If `NULL` (the default) all files are merged.
`quiet`	Whether to hide log messages (defaults to TRUE).
`warnings`	Whether to display warning messages (defaults to TRUE).

Value

Returns a GTFS object, with an updated `validation_result` attribute, in which each `data.table` is the combination (by row) of `data.tables` with the same name from the GTFS objects passed in `...`.

Details

Please note that this function does not disambiguate `ids` that may be repeated within different GTFS objects. Please let us know if you'd like to see this feature implemented.

Examples

```
spo_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
ggl_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfstools")

spo_gtfs <- read_gtfs(spo_path)
names(spo_gtfs)

ggl_gtfs <- read_gtfs(ggl_path)
names(ggl_gtfs)

merged_gtfs <- merge_gtfs(spo_gtfs, ggl_gtfs)
names(merged_gtfs)

# use a list() to programatically merge many GTFS objects
merged_gtfs <- merge_gtfs(list(spo_gtfs, ggl_gtfs))
```

`read_gtfs` *Read GTFS files*

Description

Reads GTFS text files from either a local `.zip` file or an URL.

Usage

```
read_gtfs(
  path,
  files = NULL,
  fields = NULL,
  skip = NULL,
  quiet = TRUE,
  encoding = "unknown",
  warnings
)
```

Arguments

`path`	The path to a GTFS `.zip` file.
`files`	A character vector containing the text files to be read from the GTFS (without the `.txt` extension). If `NULL` (the default) all existing files are read.

fields	A named list containing the fields to be read from each text file, in the format list(file = c("field1","field2")). If NULL (the default), all fields from the files specified in files are read. If a file is specified in files but not in fields, all fields from that file will be read (i.e. you may specify in fields only files whose fields you want to subset).
skip	A character vector containing the text files that should not be read from the GTFS, without the .txt extension. If NULL (the default), no files are skipped. Cannot be used if files is already set.
quiet	Whether to hide log messages and progress bars (defaults to TRUE).
encoding	A string, ultimately passed to \link[data.table]{fread}. Defaults to "unknown". Other possible options are "UTF-8" and "Latin-1". Please note that this is not used to re-encode the input, but to enable handling encoded strings in their native encoding.
warnings	DEPRECATED. Whether to display warning messages.

Value

A data.table-based GTFS object: a list of data.tables in which each table represents a GTFS text file.

Details

The column types of each data.table in the final GTFS object conform as closely as possible to the Google's Static GTFS Reference. Exceptions are date-related columns (such as calendar.txt's start_date and end_date, for example), which are converted to Date objects, instead of being kept as integers, allowing for easier data manipulation. These columns are converted back to integers when writing the GTFS object to a .zip file using write_gtfs.

See Also

Other io functions: write_gtfs()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)
names(gtfs)

gtfs <- read_gtfs(data_path, files = c("trips", "stop_times"))
names(gtfs)

gtfs <- read_gtfs(data_path, skip = "trips")
names(gtfs)

gtfs <- read_gtfs(data_path, fields = list(agency = "agency_id"))
names(gtfs)
names(gtfs$agency)
```

remove_duplicates *Remove duplicated entries*

Description

Removes duplicated entries from GTFS objects tables.

Usage

```
remove_duplicates(gtfs)
```

Arguments

gtfs	A GTFS object.

Value

A GTFS object containing only unique entries.

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)

# this gtfs includes some duplicated entries
gtfs$agency

gtfs <- remove_duplicates(gtfs)
gtfs$agency
```

set_trip_speed *Set trip average speed*

Description

Sets the average speed of each specified `trip_id` by changing the `arrival_time` and `departure_time` columns in `stop_times`.

Usage

```
set_trip_speed(gtfs, trip_id, speed, unit = "km/h", by_reference = FALSE)
```

Arguments

gtfs	A GTFS object as created by read_gtfs.
trip_id	A string vector including the trip_ids to have their average speed set.
speed	A numeric representing the speed to be set. Its length must either equal 1, in which case the value is recycled for all trip_ids, or equal trip_id's length.
unit	A string representing the unit in which the speed is given. One of "km/h" (the default) or "m/s".
by_reference	Whether to update stop_times' data.table by reference. Defaults to FALSE.

Value

If by_reference is set to FALSE, returns a GTFS object with the time columns of its stop_times adjusted. Else, returns a GTFS object invisibly (note that in this case the original GTFS object is altered).

Details

The average speed is calculated as the difference between the arrival time at the last stop minus the departure time at the first top, over the trip's length (as calculated via get_trip_geometry, based on the shapes file). The arrival and departure times at all other stops (i.e. not the first neither the last) are set as "", which is written as NA with write_gtfs. Some transport routing software, such as OpenTripPlanner, support specifying stop times like so. In such cases, they estimate arrival/departure times at the others stops based on the average speed as well. We plan to add that feature to this function in the future.

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)

# the examples below require the 'lwgeom' package to be installed
if (requireNamespace("lwgeom", quietly = TRUE)) {

  gtfs_new_speed <- set_trip_speed(gtfs, trip_id = "CPTM L07-0", 50)
  gtfs_new_speed$stop_times[trip_id == "CPTM L07-0"]

  # original gtfs remains unchanged
  gtfs$stop_times[trip_id == "CPTM L07-0"]

  # now do it by reference
  set_trip_speed(gtfs, trip_id = "CPTM L07-0", 50, by_reference = TRUE)
  gtfs$stop_times[trip_id == "CPTM L07-0"]

}
```

validate_gtfs *Validate GTFS file*

Description

Validates the GTFS object against GTFS specifications and raises warnings if required files/fields are not found.

Important note: this function is considered deprecated. Use it with caution, and note that its usage and output may heavily change in future versions of `gtfstools`.

Usage

```
validate_gtfs(gtfs, files = NULL, quiet = TRUE, warnings = TRUE)
```

Arguments

`gtfs`	A GTFS object as created by `read_gtfs`.
`files`	A character vector containing the text files to be validated against the GTFS specification (without the `.txt` extension). If `NULL` (the default) the provided GTFS is validated against all possible GTFS text files.
`quiet`	Whether to hide log messages (defaults to TRUE).
`warnings`	Whether to display warning messages (defaults to TRUE).

Value

A GTFS object with a `validation_result` attribute. This attribute is a `data.table` containing the validation summary of all possible fields from the specified files.

Details

GTFS object's files and fields are validated against the GTFS specifications as documented in Google's Static GTFS Reference:

- GTFS feeds are considered valid if they include all required files and fields. If a required file/field is missing the function (optionally) raises a warning.
- Optional files/fields are listed in the reference above but are not required, thus no warning is raised if they are missing.
- Extra files/fields are those who are not listed in the reference above (either because they refer to a specific GTFS extension or due to any other reason).

Note that some files (`calendar.txt`, `calendar_dates.txt` and `feed_info.txt`) are conditionally required. This means that:

- `calendar.txt` is initially set as a required file. If it's not present, however, it becomes optional and `calendar_dates.txt` (originally set as optional) becomes required.
- `feed_info.txt` is initially set as an optional file. If `translations.txt` is present, however, it becomes required.

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")

gtfs <- read_gtfs(data_path)
attr(gtfs, "validation_result")

# should not raise a warning, because 'shapes' is not a required file
gtfs$shapes <- NULL
validation_result <- validate_gtfs(gtfs)

# should raise a warning, because 'stop_times' is a required file
gtfs$stop_times <- NULL
validation_result <- validate_gtfs(gtfs)
```

---

write_gtfs *Write GTFS files*

Description

Writes GTFS objects as GTFS `.zip` files.

Usage

```
write_gtfs(
  gtfs,
  path,
  files = NULL,
  standard_only = FALSE,
  as_dir = FALSE,
  overwrite = TRUE,
  quiet = TRUE,
  optional,
  extra,
  warnings
)
```

Arguments

`gtfs`	A GTFS object as created by `read_gtfs`.
`path`	The path to the `.zip` file in which the feed should be written to.
`files`	A character vector containing the name of the elements to be written to the feed. If `NULL` (the default), all elements inside the GTFS object are written.
`standard_only`	Whether to write only standard files and fields (defaults to `FALSE`, which doesn't drop extra files and fields).
`as_dir`	Whether to write the feed as a directory, instead of a `.zip` file (defaults to `FALSE`, which means that the field is written as a zip file).

overwrite	Whether to overwrite existing .zip file (defaults to TRUE).
quiet	Whether to hide log messages and progress bars (defaults to TRUE).
optional	DEPRECATED, use files instead. Whether to write optional .txt.
extra	DEPRECATED, use files and standard_only instead. Whether to write extra .txt.
warnings	DEPRECATED. Whether to display warning messages.

Value

Invisibly returns the same GTFS object passed to the gtfs parameter.

See Also

Other io functions: read_gtfs()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)

tmp_dir <- file.path(tempdir(), "tmpdir")
dir.create(tmp_dir)
list.files(tmp_dir) #'
tmp_file <- tempfile(pattern = "gtfs", tmpdir = tmp_dir, fileext = ".zip")
write_gtfs(gtfs, tmp_file)
list.files(tmp_dir)

gtfs_all_files <- read_gtfs(tmp_file)
names(gtfs_all_files)

write_gtfs(gtfs, tmp_file, files = "stop_times")
gtfs_stop_times <- read_gtfs(tmp_file)
names(gtfs_stop_times)
```

Index

∗ **filtering functions**
filter_by_route_id, 4
filter_by_route_type, 5
filter_by_sf, 6
filter_by_shape_id, 7
filter_by_stop_id, 8
filter_by_trip_id, 9
∗ **io functions**
read_gtfs, 17
write_gtfs, 22
_PACKAGE *(*gtfstools*)*, 15

convert_shapes_to_sf, 2
convert_stops_to_sf, 3

filter_by_route_id, 4, *5*, *7–9*
filter_by_route_type, *4*, 5, *7–9*
filter_by_sf, *4, 5*, 6, *8, 9*
filter_by_shape_id, *4, 5*, *7*, 7, *9*
filter_by_stop_id, *4, 5*, *7, 8*, 8, *9*
filter_by_trip_id, *4, 5*, *7–9*, 9

geos_binary_pred, *6*
get_parent_station, 10
get_trip_duration, 11
get_trip_geometry, 12, *20*
get_trip_geometry(), *14*
get_trip_segment_duration, 13
get_trip_speed, 14
gtfstools, 15
gtfstools-package *(*gtfstools*)*, 15

merge_gtfs, 16

read_gtfs, *10, 11*, *13*, *16*, 17, *20–23*
remove_duplicates, 19

set_trip_speed, 19

validate_gtfs, 21

write_gtfs, *18*, *20*, 22