Package 'gtfstools'

January 9, 2025

Type Package

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

Version 1.4.0

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, cli, curl, data.table, gtfsio (>= 1.0.0), parallelly, processx, sf, sfheaders, units, utils, zip

Suggests covr, ggplot2, jsonlite, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.3.2

LinkingTo cpp11

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], Marcus Saraiva [ctb] (<https://orcid.org/0000-0001-6218-2338>), Ipea - Institute for Applied Economic Research [cph, fnd]

Maintainer Daniel Herszenhut <dhersz@gmail.com>

Repository CRAN

Date/Publication 2025-01-09 00:40:02 UTC

Contents

as_dt_gtfs *Coerce lists and GTFS objects from other packages into gtfstools-compatible GTFS objects*

Description

Coerces an existing object, such as a `list` or a GTFS object created from other packages (`{tidytransit}` and `{gtfsio}`, for example) into a gtfstools-compatible GTFS object - i.e. one whose internal tables are represented with `data.tables` and whose fields are formatted like the fields of a feed read with `read_gtfs()`.

`as_dt_gtfs()` is an S3 generic, with methods for:

- `tidygtfs`: the class of GTFS objects read with `tidytransit::read_gtfs()`. This method converts all `tibbles` to `data.tables` and convert time columns, represented as `hms` objects in a `tidygtfs`, to strings in the `"HH:MM:SS"` format.
- `gtfs`: the class of GTFS objects read with `gtfsio::import_gtfs()`. This method convert all date fields, represented as `integers` in `{gtfsio}`'s representation, to `Date` objects.
- `list`: this method tries to convert the elements of a list into `data.tables`. Please note that all list elements must inherit from `data.frame` and must be named. This method does not try not convert fields to the representation used in `{gtfstools}`, as it does not have any information on how they are formatted in the first place.

Usage

```
as_dt_gtfs(gtfs, ...)

## S3 method for class 'tidygtfs'
as_dt_gtfs(gtfs, calculate_distance = TRUE, ...)

## S3 method for class 'gtfs'
as_dt_gtfs(gtfs, ...)

## S3 method for class 'list'
as_dt_gtfs(gtfs, ...)
```

Arguments

`gtfs`	The object that should be coerced to a `dt_gtfs`.
`...`	Ignored.
`calculate_distance`	A logical. Passed to `convert_sf_to_shapes()`, which only affects the output when the object to be converted includes a `shapes` element. Controls whether this function, used to convert a `LINESTRING sf` into a GTFS `shapes` table, should calculate and populate the `shape_dist_traveled` column. This column is used to describe the distance along the shape from each one of its points to its first point. Defaults to `TRUE`.

Value

A dt_gtfs GTFS object.

Examples

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

gtfsio_gtfs <- gtfsio::import_gtfs(data_path)
class(gtfsio_gtfs)

gtfstools_gtfs <- as_dt_gtfs(gtfsio_gtfs)
class(gtfstools_gtfs)

gtfs_like_list <- unclass(gtfsio_gtfs)
class(gtfs_like_list)

gtfstools_gtfs <- as_dt_gtfs(gtfs_like_list)
class(gtfstools_gtfs)
```

---

convert_sf_to_shapes *Convert a simple feature object into a* shapes *table*

Description

Converts a LINESTRING sf object into a GTFS shapes table.

Usage

```
convert_sf_to_shapes(sf_shapes, shape_id = NULL, calculate_distance = TRUE)
```

Arguments

sf_shapes	A LINESTRING sf associating each shape_ids to a geometry. This object must use CRS WGS 84 (EPSG code 4326).
shape_id	A character vector specifying the shape_ids to be converted. If NULL (the default), all shapes are converted.
calculate_distance	A logical. Whether to calculate and populate the shape_dist_traveled column. This column is used to describe the distance along the shape from each one of its points to its first point. Defaults to TRUE.

Value

A data.table representing a GTFS shapes table.

Examples

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

# first converting existing shapes table into a sf object
shapes_sf <- convert_shapes_to_sf(gtfs)
head(shapes_sf)

# by default converts all shapes
result <- convert_sf_to_shapes(shapes_sf)
result

# shape_id argument controls which shapes are converted
result <- convert_sf_to_shapes(shapes_sf, shape_id = c("17846", "17847"))
result

# calculate_distance argument controls whether to calculate
# shape_dist_traveled or not
result <- convert_sf_to_shapes(shapes_sf, calculate_distance = TRUE)
result
```

---

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, sort_sequence = FALSE)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
shape_id	A character vector including the shape_ids 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).
sort_sequence	A logical. Whether to sort shapes by shape_pt_sequence. Defaults to FALSE, otherwise spec-compliant feeds, in which shape points are already ordered by shape_pt_sequence, would be penalized through longer processing times. Shapes generated from unordered sequences do not correctly depict the real life trip shapes.

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, as created by read_gtfs().
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
```

`convert_time_to_seconds` *Convert time fields to seconds after midnight*

Description

Converts `stop_times`' and `frequencies`' fields in the "HH:MM:SS" format to seconds after midnight. Instead of overwritting the existing fields, creates new fields with the `_secs` suffix.

Usage

```
convert_time_to_seconds(gtfs, file = NULL, by_reference = FALSE)
```

Arguments

`gtfs`	A GTFS object, as created by `read_gtfs()`.
`file`	A character vector, specifying the file whose fields should be converted. If `NULL` (the default), the function attempts to convert the times from both files, but only raises an error if none of them exist.
`by_reference`	Whether to update the tables by reference. Defaults to `FALSE`.

Value

If `by_reference` is `FALSE`, returns a GTFS object with additional time in seconds columns (identified by a `_secs` suffix). Else, returns a GTFS object invisibly (please note that in such case the original GTFS object is altered).

Examples

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

gtfs <- read_gtfs(data_path)

# by default converts both 'stop_times' and 'frequencies' times
converted_gtfs <- convert_time_to_seconds(gtfs)
head(converted_gtfs$stop_times)
head(converted_gtfs$frequencies)

# choose which table to convert with 'file'
converted_gtfs <- convert_time_to_seconds(gtfs, file = "frequencies")
head(converted_gtfs$stop_times)
head(converted_gtfs$frequencies)

# original gtfs remained unchanged, as seen with the frequencies table above
```

```
# change original object without creating a copy with 'by_reference = TRUE'
convert_time_to_seconds(gtfs, by_reference = TRUE)
head(gtfs$stop_times)
head(gtfs$frequencies)
```

`download_validator` *Download MobilityData's GTFS validator*

Description

Downloads MobilityData's command line tool to validate GTFS feeds.

Usage

```
download_validator(path, version = "latest", force = FALSE, quiet = TRUE)
```

Arguments

`path`	A string. The directory where the validator should be saved to.
`version`	A string. The version of the validator that should be downloaded. Defaults to `"latest"`, but accepts version numbers as strings (i.e. to download version v6.0.0 please enter `"6.0.0"`). Please check MobilityData/gtfs-validator releases for the full set of available versions.
`force`	A logical. Whether to overwrite a previously downloaded validator in `path`. Defaults to `FALSE`.
`quiet`	A logical. Whether to hide log messages and progress bars. Defaults to `TRUE`.

Value

Invisibly returns the normalized path to the downloaded validator.

See Also

Other validation: `validate_gtfs()`

Examples

```
path <- tempdir()

download_validator(path)

# specifying a specific version
download_validator(path, version = "6.0.0")
```

filter_by_agency_id *Filter GTFS object by* agency_id

Description

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

Usage

```
filter_by_agency_id(gtfs, agency_id, keep = TRUE)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
agency_id	A character vector. The agency_ids used to filter the data.
keep	A logical. Whether the entries related to the specified agency_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_service_id(), filter_by_sf(), filter_by_shape_id(), filter_by_spatial_extent(), filter_by_stop_id(), filter_by_time_of_day(), filter_by_trip_id(), filter_by_weekday()

Examples

```
data_path <- system.file("extdata/ber_gtfs.zip", package = "gtfstools")
gtfs <- read_gtfs(data_path)
agency_id <- "92"

object.size(gtfs)

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

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

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, as created by read_gtfs().
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_agency_id(), filter_by_route_type(), filter_by_service_id(), filter_by_sf(), filter_by_shape_id(), filter_by_spatial_extent(), filter_by_stop_id(), filter_by_time_of_day(), filter_by_trip_id(), filter_by_weekday()

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, as created by `read_gtfs()`.
`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 include the route types listed in the GTFS Schedule specification and in the Google Transit implementation. The types specified in the GTFS Schedule specification 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.

The types defined in Google Transit's extension are listed below, including some examples (not available for all types):

- 100 - Railway Service - Not applicable (N/A)
- 101 - High Speed Rail Service - TGV (FR), ICE (DE), Eurostar (GB)

- 102 - Long Distance Trains - InterCity/EuroCity
- 103 - Inter Regional Rail Service - InterRegio (DE), Cross County Rail (GB)
- 104 - Car Transport Rail Service
- 105 - Sleeper Rail Service - GNER Sleeper (GB)
- 106 - Regional Rail Service - TER (FR), Regionalzug (DE)
- 107 - Tourist Railway Service - Romney, Hythe & Dymchurch (GB)
- 108 - Rail Shuttle (Within Complex) - Gatwick Shuttle (GB), Sky Line (DE)
- 109 - Suburban Railway - S-Bahn (DE), RER (FR), S-tog (Kopenhagen)
- 110 - Replacement Rail Service
- 111 - Special Rail Service
- 112 - Lorry Transport Rail Service
- 113 - All Rail Services
- 114 - Cross-Country Rail Service
- 115 - Vehicle Transport Rail Service
- 116 - Rack and Pinion Railway - Rochers de Naye (CH), Dolderbahn (CH)
- 117 - Additional Rail Service
- 200 - Coach Service
- 201 - International Coach Service - EuroLine, Touring
- 202 - National Coach Service - National Express (GB)
- 203 - Shuttle Coach Service - Roissy Bus (FR), Reading-Heathrow (GB)
- 204 - Regional Coach Service
- 205 - Special Coach Service
- 206 - Sightseeing Coach Service
- 207 - Tourist Coach Service
- 208 - Commuter Coach Service
- 209 - All Coach Services
- 400 - Urban Railway Service
- 401 - Metro Service - Métro de Paris
- 402 - Underground Service - London Underground, U-Bahn
- 403 - Urban Railway Service
- 404 - All Urban Railway Services
- 405 - Monorail
- 700 - Bus Service
- 701 - Regional Bus Service - Eastbourne-Maidstone (GB)
- 702 - Express Bus Service - X19 Wokingham-Heathrow (GB)
- 703 - Stopping Bus Service - 38 London: Clapton Pond-Victoria (GB)
- 704 - Local Bus Service

- 705 - Night Bus Service - N prefixed buses in London (GB)
- 706 - Post Bus Service - Maidstone P4 (GB)
- 707 - Special Needs Bus
- 708 - Mobility Bus Service
- 709 - Mobility Bus for Registered Disabled
- 710 - Sightseeing Bus
- 711 - Shuttle Bus - 747 Heathrow-Gatwick Airport Service (GB)
- 712 - School Bus
- 713 - School and Public Service Bus
- 714 - Rail Replacement Bus Service
- 715 - Demand and Response Bus Service
- 716 - All Bus Services
- 800 - Trolleybus Service
- 900 - Tram Service
- 901 - City Tram Service
- 902 - Local Tram Service - Munich (DE), Brussels (BE), Croydon (GB)
- 903 - Regional Tram Service
- 904 - Sightseeing Tram Service - Blackpool Seafront (GB)
- 905 - Shuttle Tram Service
- 906 - All Tram Services
- 1000 - Water Transport Service
- 1100 - Air Service
- 1200 - Ferry Service
- 1300 - Aerial Lift Service - Telefèric de Montjuïc (ES), Saleve (CH), Roosevelt Island Tramway (US)
- 1301 - Telecabin Service
- 1302 - Cable Car Service
- 1303 - Elevator Service
- 1304 - Chair Lift Service
- 1305 - Drag Lift Service
- 1306 - Small Telecabin Service
- 1307 - All Telecabin Services
- 1400 - Funicular Service - Rigiblick (Zürich, CH)
- 1500 - Taxi Service
- 1501 - Communal Taxi Service - Marshrutka (RU), dolmuş (TR)
- 1502 - Water Taxi Service
- 1503 - Rail Taxi Service

- 1504 - Bike Taxi Service
- 1505 - Licensed Taxi Service
- 1506 - Private Hire Service Vehicle
- 1507 - All Taxi Services
- 1700 - Miscellaneous Service
- 1702 - Horse-drawn Carriage

See Also

Other filtering functions: filter_by_agency_id(), filter_by_route_id(), filter_by_service_id(), filter_by_sf(), filter_by_shape_id(), filter_by_spatial_extent(), filter_by_stop_id(), filter_by_time_of_day(), filter_by_trip_id(), filter_by_weekday()

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_service_id *Filter GTFS object by* service_id

Description

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

Usage

```
filter_by_service_id(gtfs, service_id, keep = TRUE)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
service_id	A character vector. The service_ids used to filter the data.
keep	A logical. Whether the entries related to the specified service_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_agency_id()`, `filter_by_route_id()`, `filter_by_route_type()`, `filter_by_sf()`, `filter_by_shape_id()`, `filter_by_spatial_extent()`, `filter_by_stop_id()`, `filter_by_time_of_day()`, `filter_by_trip_id()`, `filter_by_weekday()`

Examples

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

object.size(gtfs)

# keeps entries related to the specified service_ids
smaller_gtfs <- filter_by_service_id(gtfs, service_ids)
object.size(smaller_gtfs)

# drops entries related to the specified service_ids
smaller_gtfs <- filter_by_service_id(gtfs, service_ids, keep = FALSE)
object.size(smaller_gtfs)
```

---

filter_by_sf *Filter a GTFS object using a* simple features *object (deprecated)*

Description

This function has been deprecated as of the current package version and will be completely removed from version 2.0.0 onward. Please use `filter_by_spatial_extent()` instead.

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, as created by `read_gtfs()`.
`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_agency_id(), filter_by_route_id(), filter_by_route_type(), filter_by_service_id(), filter_by_shape_id(), filter_by_spatial_extent(), filter_by_stop_id(), filter_by_time_of_day(), filter_by_trip_id(), filter_by_weekday()

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, as created by read_gtfs().
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_agency_id(), filter_by_route_id(), filter_by_route_type(), filter_by_service_id(), filter_by_sf(), filter_by_spatial_extent(), filter_by_stop_id(), filter_by_time_of_day(), filter_by_trip_id(), filter_by_weekday()

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_spatial_extent

Filter a GTFS object using a spatial extent

Description

Filters a GTFS object using a spatial extent (passed as an sf object), keeping (or dropping) entries related to shapes and trips whose geometries are selected through a specified spatial operation.

Usage

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

Arguments

gtfs	A GTFS object, as created by read_gtfs().
geom	An sf object. Describes the spatial extent 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 selected by the given spatial operation 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_agency_id(), filter_by_route_id(), filter_by_route_type(), filter_by_service_id(), filter_by_sf(), filter_by_shape_id(), filter_by_stop_id(), filter_by_time_of_day(), filter_by_trip_id(), filter_by_weekday()

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_spatial_extent(gtfs, bbox)
object.size(smaller_gtfs)

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

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

Usage

```
filter_by_stop_id(
  gtfs,
  stop_id,
  keep = TRUE,
  include_children = TRUE,
  include_parents = keep,
  full_trips = TRUE
)
```

Arguments

`gtfs`	A GTFS object, as created by `read_gtfs()`.
`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).
`include_children`	A logical. Whether the filtered output should keep/drop children stops of those specified in `stop_id`. Defaults to `TRUE` - i.e. by default children stops are kept if their parents are kept and dropped if their parents are dropped.
`include_parents`	A logical. Whether the filtered output should keep/drop parent stations of those specified in `stop_id`. Defaults to the same value of `keep` - i.e. by default parent stations are kept both when their children are kept and dropped, because they can be parents of multiple stops that are not necessarily dropped, even if their sibling are.

full_trips A logical. Whether to keep all stops that compose trips that pass through the stops specified in stop_id. Defaults to TRUE, in order to preserve the behavior of the function in versions 1.2.0 and below. Please note that when TRUE, the resultant filtered feed may contain more stops than the ones specified in stop_id to preserve the integrity of the trips. IMPORTANT: using full_trips = TRUE is flagged as deprecated as of version 1.3.0 and this parameter will default to FALSE from version 2.0.0 onward.

Value

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

See Also

Other filtering functions: filter_by_agency_id(), filter_by_route_id(), filter_by_route_type(), filter_by_service_id(), filter_by_sf(), filter_by_shape_id(), filter_by_spatial_extent(), filter_by_time_of_day(), filter_by_trip_id(), filter_by_weekday()

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, full_trips = FALSE)
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,
  full_trips = FALSE
)
object.size(smaller_gtfs)

# the old behavior of filtering trips that contained the specified stops has
# been deprecated
invisible(filter_by_stop_id(gtfs, stop_ids, full_trips = TRUE))
```

---

filter_by_time_of_day *Filter GTFS object by time of day*

Description

Filters a GTFS object by time of day, keeping (or dropping) the relevant entries in each file. Please see the details section for more information on how this function filters the `frequencies` and `stop_times` tables, as well as how it handles `stop_times` tables that contain trips with some empty departure and arrival times.

Usage

```
filter_by_time_of_day(
  gtfs,
  from,
  to,
  keep = TRUE,
  full_trips = FALSE,
  update_frequencies = TRUE
)
```

Arguments

`gtfs`	A GTFS object, as created by `read_gtfs()`.
`from`	A string. The starting point of the time of day, in the "HH:MM:SS" format.
`to`	A string. The ending point of the time of day, in the "HH:MM:SS" format.
`keep`	A logical. Whether the entries related to the specified time of day should be kept or dropped (defaults to `TRUE`, which keeps the entries).
`full_trips`	A logical. Whether trips should be treated as immutable blocks or each of its stops should be considered separately when filtering the `stop_times` table (defaults to `FALSE`, which considers each stop individually). Please check the details section for more information on how this parameter changes the function behaviour.
`update_frequencies`	A logical. Whether the `frequencies` table should have its `start_time` and `end_time` fields updated to fit inside/outside the specified time of day (defaults to `FALSE`, which doesn't update the fields).

Value

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

Details

When filtering the `frequencies` table, `filter_by_time_of_day()` respects the `exact_times` field. This field indicates whether the service follows a fixed schedule throughout the day or not. If it's 0 (or if it's not present), the service does not follow a fixed schedule. Instead, the operators try to maintain the listed headways. In such cases, if `update_frequencies` is `TRUE` we just update `start_time` and `end_time` to the appropriate value of `from` or `to` (which of this value is used depends on `keep`).

If `exact_times` is 1, however, operators try to strictly adhere to the start times and headway. As a result, when updating the `start_time` field we need to follow the listed headway. For example, take

a trip that has its start time listed as 06:00:00, its end time listed as 08:00:00 and its headway listed as 300 secs (5 minutes). If you decide to filter the feed to keep the time of day between 06:32:00 and 08:00:00 while updating `frequencies`, the `start_time` field must be updated to 06:35:00 in order to preserve the correct departure times of this trips, instead of simply updating it to 06:32:00.

Another things to keep an eye on when filtering the `frequencies` table is that the corresponding `stop_times` entries of trips listed in the `frequencies` table should not be filtered, even if their departure and arrival times fall outside the specified time of day. This is because the `stop_times` entries of `frequencies`' trips are just templates that describe how long a segment between two stops takes, so the departure and arrival times listed there do not actually represent the actual departure and arrival times seen in practice. Taking the same example listed above, the corresponding `stop_times` entries of that trip could describe a departure from the first stop at 12:00:00 and an arrival at the second stop at 12:03:00. That doesn't mean the trip will actually leave and arrive at the stops at these times, but rather that it takes 3 minutes to get from the first to the second stop. So when the trip departs from the first stop at 06:35:00, it will get to the second at 06:38:00.

When filtering the `stop_times` table, a few other details should be observed. First, one could wish to filter a GTFS object in order to keep all trips that cross a given time of day, whereas others may want to keep only the specific entries that fall inside the specified time of day. For example, take a trip that leaves the first stop at 06:30:00, gets to the second at 06:35:00 and then gets to the third at 06:45:00. When filtering to keep entire trips that cross the time of day between 06:30:00 and 06:40:00, all three stops will have to be kept. If, however, you want to keep only the entries that fall within the specified time of day, only the first two should be kept. To control such behaviour you need to set the `full_trips` parameter. When it's `TRUE`, the function behaves like the first case, and when it's `FALSE`, like the second.

When using `full_trips` in conjunction with `keep`, please note how their behaviour stack. When both are `TRUE`, trips are always fully kept. When `keep` is `FALSE`, however, trips are fully dropped, even if some of their stops are visited outside the specified time of day.

Finally, please note that many GTFS feeds may contain `stop_times` entries with empty departure and arrival times. In such cases, filtering by time of day with `full_trips` as `FALSE` will drop the entries with empty times. Please set `full_trips` to `TRUE` to preserve these entries.

See Also

Other filtering functions: `filter_by_agency_id()`, `filter_by_route_id()`, `filter_by_route_type()`, `filter_by_service_id()`, `filter_by_sf()`, `filter_by_shape_id()`, `filter_by_spatial_extent()`, `filter_by_stop_id()`, `filter_by_trip_id()`, `filter_by_weekday()`

Examples

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

# taking a look at the original frequencies and stop_times
head(gtfs$frequencies)
head(gtfs$stop_times)

smaller_gtfs <- filter_by_time_of_day(gtfs, "05:00:00", "06:00:00")

# filter_by_time_of_day filters the frequencies table but doesn't filter the
# stop_times table because they're just templates
```

```
head(smaller_gtfs$frequencies)
head(smaller_gtfs$stop_times)

# frequencies entries can be adjusted using update_frequencies = TRUE
smaller_gtfs <- filter_by_time_of_day(
  gtfs,
  "05:30:00",
  "06:00:00",
  update_frequencies = TRUE
)
head(smaller_gtfs$frequencies)

# when keep = FALSE, the behaviour of the function in general, and of
# update_frequencies in particular, is a bit different
smaller_gtfs <- filter_by_time_of_day(
  gtfs,
  "05:30:00",
  "06:00:00",
  keep = FALSE,
  update_frequencies = TRUE
)
head(smaller_gtfs$frequencies)

# let's remove the frequencies table to check the behaviour of full_trips
gtfs$frequencies <- NULL
smaller_gtfs <- filter_by_time_of_day(
  gtfs,
  "05:30:00",
  "06:00:00"
)
head(smaller_gtfs$stop_times)

smaller_gtfs <- filter_by_time_of_day(
  gtfs,
  "05:30:00",
  "06:00:00",
  full_trips = TRUE
)
head(smaller_gtfs$stop_times)
```

`filter_by_trip_id` *Filter GTFS object by* `trip_id`

Description

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

Usage

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

Arguments

gtfs	A GTFS object, as created by read_gtfs().
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_agency_id(), filter_by_route_id(), filter_by_route_type(), filter_by_service_id(), filter_by_sf(), filter_by_shape_id(), filter_by_spatial_extent(), filter_by_stop_id(), filter_by_time_of_day(), filter_by_weekday()

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)
```

---

filter_by_weekday *Filter GTFS object by weekday*

Description

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

Usage

```
filter_by_weekday(gtfs, weekday, combine = "or", keep = TRUE)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
weekday	A character vector. The weekdays used to filter the data. Possible values are c("monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday").
combine	A string. Specifies which logic operation (OR or AND) should be used to filter the calendar table when multiple weekdays are specified. Defaults to "or". Please check the details and examples sections for more information on this argument usage.
keep	A logical. Whether the entries related to the specified weekdays 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.

combine usage

When filtering the calendar table using weekdays, one could reason about the process in different ways. For example, you may want to keep only services who run on mondays AND tuesdays. Or you may want to keep services that run EITHER on mondays OR on tuesdays. The first case is the equivalent of filtering using the expression monday == 1 & tuesday == 1, while the second uses monday == 1 | tuesday == 1. You can use the combine argument to control this behaviour.

Please note that combine also works together with keep. Using the same examples listed above, you could either keep the entries related to services that run on mondays and tuesdays or drop them, depending on the value you pass to keep.

See Also

Other filtering functions: filter_by_agency_id(), filter_by_route_id(), filter_by_route_type(), filter_by_service_id(), filter_by_sf(), filter_by_shape_id(), filter_by_spatial_extent(), filter_by_stop_id(), filter_by_time_of_day(), 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 services than run EITHER on monday OR on sunday
smaller_gtfs <- filter_by_weekday(gtfs, weekday = c("monday", "sunday"))
smaller_gtfs$calendar[, c("service_id", "monday", "sunday")]
object.size(smaller_gtfs)

# keeps entries related to services than run on monday AND on sunday
smaller_gtfs <- filter_by_weekday(
  gtfs,
```

```
  weekday = c("monday", "sunday"),
  combine = "and"
)
smaller_gtfs$calendar[, c("service_id", "monday", "sunday")]
object.size(smaller_gtfs)

# drops entries related to services than run EITHER on monday OR on sunday
# the resulting gtfs shouldn't include any trips running on these days
smaller_gtfs <- filter_by_weekday(
  gtfs,
  weekday = c("monday", "sunday"),
  keep = FALSE
)
smaller_gtfs$calendar[, c("service_id", "monday", "sunday")]
object.size(smaller_gtfs)

# drops entries related to services than run on monday AND on sunday
# the resulting gtfs may include trips that run on these days, but no trips
# that run on both these days
smaller_gtfs <- filter_by_weekday(
  gtfs,
  weekday = c("monday", "sunday"),
  combine = "and",
  keep = FALSE
)
smaller_gtfs$calendar[, c("service_id", "monday", "sunday")]
object.size(smaller_gtfs)
```

`frequencies_to_stop_times` *Convert frequencies to stop times*

Description

Creates `stop_times` entries based on the frequencies specified in the `frequencies` table.

Usage

```
frequencies_to_stop_times(gtfs, trip_id = NULL, force = FALSE)
```

Arguments

`gtfs`	A GTFS object, as created by `read_gtfs()`.
`trip_id`	A character vector including the `trip_id`s to have their frequencies converted to `stop_times` entries. If `NULL` (the default), the function converts all trips listed in the `frequencies` table.
`force`	Whether to convert trips specified in the `frequencies` table even if they are not described in `stop_times` (defaults to `FALSE`). When set to `TRUE`, these mismatched trip are removed from the `frequencies` table and their correspondent entries in `trips` are substituted by what would be their converted counterpart.

Value

A GTFS object with updated `frequencies`, `stop_times` and `trips` tables.

Details

A single trip described in a `frequencies` table may yield multiple trips after converting the GTFS. Let's say, for example, that the `frequencies` table describes a trip called `"example_trip"`, that starts at 08:00 and stops at 09:00, with a 30 minutes headway.

In practice, that means that one trip will depart at 08:00, another at 08:30 and yet another at 09:00. `frequencies_to_stop_times()` appends a `"_<n>"` suffix to the newly created trips to differentiate each one of them (e.g. in this case, the new trips, described in the `trips` and `stop_times` tables, would be called `"example_trip_1"`, `"example_trip_2"` and `"example_trip_3"`).

Examples

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

# converts all trips listed in the frequencies table
converted_gtfs <- frequencies_to_stop_times(gtfs)

# converts only the specified trip_id
converted_gtfs <- frequencies_to_stop_times(gtfs, trip)

# how the specified trip_id was described in the frequencies table
head(gtfs$frequencies[trip_id == trip])

# the first row of each equivalent stop_times entry in the converted gtfs
equivalent_stop_times <- converted_gtfs$stop_times[grepl(trip, trip_id)]
equivalent_stop_times[equivalent_stop_times[, .I[1], by = trip_id]$V1]
```

---

get_children_stops *Get children stops recursively*

Description

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

Usage

```
get_children_stops(gtfs, stop_id = NULL)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
stop_id	A string vector including the stop_ids to have their children returned. If NULL (the default), the function returns the children of every stop_id in the GTFS.

Value

A data.table containing the stop_ids and their children' stop_ids. If a stop doesn't have a child, its correspondent child_id entry is marked as "".

Examples

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

children <- get_children_stops(gtfs)
head(children)

# use the stop_id argument to control which stops are analyzed
children <- get_children_stops(gtfs, stop_id = c("F12S", "F12N"))
children
```

---

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 = NULL)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
stop_id	A string vector including the stop_ids to have their parents returned. If NULL (the default), the function returns the parents of every stop_id in the GTFS.

Value

A data.table containing the stop_ids and their parent_stations. If a stop doesn't have a parent, its correspondent parent_station entry is marked as "".

See Also

get_children_stops()

Examples

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

gtfs <- read_gtfs(data_path)

parents <- get_parent_station(gtfs)
head(parents)

# use the stop_id argument to control which stops are analyzed
parents <- get_parent_station(gtfs, c("B1", "B2"))
parents
```

---

get_stop_times_patterns

Get stop times patterns

Description

Identifies spatial and spatiotemporal patterns within the stop_times table. Please see the details to understand what a "pattern" means in each of these cases.

Usage

```
get_stop_times_patterns(
  gtfs,
  trip_id = NULL,
  type = "spatial",
  sort_sequence = FALSE
)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
trip_id	A character vector including the trip_ids to have their stop_times entries analyzed. If NULL (the default), the function analyses the pattern of every trip_id in the GTFS.
type	A string specifying the type of patterns to be analyzed. Either "spatial" (the default) or "spatiotemporal".
sort_sequence	A logical specifying whether to sort timetables by stop_sequence. Defaults to FALSE, otherwise spec-compliant feeds, in which timetables points are already ordered by stop_sequence, would be penalized through longer processing times. Pattern identification based on unordered timetables may result in multiple ids identifying what would be the same pattern, had the table been ordered.

Value

A data.table associating each trip_id to a pattern_id.

Details

Two trips are assigned to the same spatial pattern_id if they travel along the same sequence of stops. They are assigned to the same spatiotemporal pattern_id, on the other hand, if they travel along the same sequence of stops and they take the same time between stops. Please note that, in such case, only the time between stops is taken into account, and the time that the trip started is ignored (e.g. if two trips depart from stop A and follow the same sequence of stops to arrive at stop B, taking both 1 hour to do so, their spatiotemporal pattern will be considered the same, even if one departed at 6 am and another at 7 am). Please also note that the stop_sequence field is currently ignored - which means that two stops are considered to follow the same sequence if one is listed right below the other on the stop_times table (e.g. if trip X lists stops A followed by stop B with stop_sequences 1 and 2, and trip Y lists stops A followed by stop B with stop_sequences 1 and 3, they are assigned to the same pattern_id).

Examples

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

gtfs <- read_gtfs(data_path)

patterns <- get_stop_times_patterns(gtfs)
head(patterns)

# use the trip_id argument to control which trips are analyzed
patterns <- get_stop_times_patterns(
  gtfs,
  trip_id = c("143765658", "143765659", "143765660")
)
patterns

# use the type argument to control the type of pattern analyzed
patterns <- get_stop_times_patterns(
  gtfs,
  trip_id = c("143765658", "143765659", "143765660"),
  type = "spatiotemporal"
)
patterns
```

---

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 table.

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,
  sort_sequence = FALSE
)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
trip_id	A character vector including the trip_ids 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).
sort_sequence	A logical specifying whether to sort shapes and timetables by shape_pt_sequence and stop_sequence, respectively. Defaults to FALSE, otherwise spec-compliant feeds, in which shape/timetables points are already ordered by shape_pt_sequence/stop_sequence, would be penalized through longer processing times. Geometries generated from unordered sequences do not correctly depict the trip trajectories.

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_length` *Get trip length*

Description

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

Usage

```
get_trip_length(
  gtfs,
  trip_id = NULL,
  file = NULL,
  unit = "km",
  sort_sequence = FALSE
)
```

Arguments

`gtfs`	A GTFS object, as created by `read_gtfs()`.
`trip_id`	A character vector including the `trip_id`s to have their length calculated If `NULL` (the default), the function calculates the length of each `trip_id` in the GTFS.
`file`	A character vector specifying the file from which lengths should be calculated (either one of or both `shapes` and `stop_times`). If `NULL` (the default), the function attempts to calculate the lengths from both files, but only raises an error if none of the files exist.
`unit`	A string representing the unit in which lengths are desired. Either `"km"` (the default) or `"m"`.
`sort_sequence`	A logical specifying whether to sort shapes and timetables by `shape_pt_sequence` and `stop_sequence`, respectively. Defaults to `FALSE`, otherwise spec-compliant feeds, in which shape/timetables points are already ordered by `shape_pt_sequence`/`stop_sequence`, would be penalized through longer processing times. Lengths calculated from trip trajectories generated with unordered sequences do not correctly depict the actual trip lengths.

Value

A `data.table` containing the length of each specified trip.

Details

Please check `get_trip_geometry()` documentation to understand how geometry generation, and consequently length calculation, differs depending on the chosen file.

Examples

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

gtfs <- read_gtfs(data_path)

trip_length <- get_trip_length(gtfs)
head(trip_length)

# the above is identical to
trip_length <- get_trip_length(gtfs, file = c("shapes", "stop_times"))
head(trip_length)

trip_ids <- c("CPTM L07-0", "2002-10-0")
trip_length <- get_trip_length(gtfs, trip_id = trip_ids)
trip_length
```

---

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",
  sort_sequence = FALSE
)
```

Arguments

`gtfs`	A GTFS object, as created by `read_gtfs()`.

trip_id	A string vector including the trip_ids 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).
sort_sequence	A logical specifying whether to sort timetables by stop_sequence. Defaults to FALSE, otherwise spec-compliant feeds, in which timetables points are already ordered by stop_sequence, would be penalized through longer processing times. Durations calculated from unordered timetables do not correctly depict the real life segment durations.

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)

# use the trip_id argument to control which trips are analyzed
trip_segment_dur <- get_trip_segment_duration(gtfs, trip_id = "CPTM L07-0")
trip_segment_dur

# use the unit argument to control in which unit the durations are calculated
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",
  sort_sequence = FALSE
)
```

Arguments

gtfs	A GTFS object, as created by read_gtfs().
trip_id	A character vector including the trip_ids 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".
sort_sequence	Ultimately passed to get_trip_length(), a logical specifying whether to sort shapes and timetables by shape_pt_sequence and stop_sequence, respectively. Speeds calculated from trip trajectories generated with unordered sequences do not correctly depict the actual trip speeds. Defaults to FALSE, otherwise spec-compliant feeds, in which shape/timetables points are already ordered by shape_pt_sequence/stop_sequence, would be penalized through longer processing times.

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)

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
```

`merge_gtfs` *Merge GTFS files*

Description

Combines many GTFS objects into a single one.

Usage

```
merge_gtfs(..., files = NULL, prefix = FALSE)
```

Arguments

`...`	GTFS objects to be merged. Each argument can either be a GTFS or a list of GTFS objects.
`files`	A character vector listing the GTFS tables to be merged. If `NULL` (the default), all tables are merged.
`prefix`	Either a logical or a character vector (defaults to `FALSE`). Whether to add a prefix to the value of id fields that identify from which GTFS object the value comes from. If `TRUE`, the prefixes will range from `"1"` to `n`, where `n` is the number of objects passed to the function. If a character vector, its elements will be used to identify the GTFS objects, and the length of the vector must equal the total amount of objects passed in `...` (the first element will identify the first GTFS, the second element the second GTFS, and so on).

Value

A GTFS object in which each table is a combination (by row) of the tables from the specified GTFS objects.

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
gtfs_list <- list(spo_gtfs, ggl_gtfs)
merged_gtfs <- merge_gtfs(gtfs_list)

# 'prefix' helps disambiguating from which GTFS each id comes from.
# if TRUE, the ids range from 1:n, where n is the number of gtfs
merged_gtfs <- merge_gtfs(gtfs_list, prefix = TRUE)
merged_gtfs$agency

# if a character vector, its elements will be used to identify the each gtfs
merged_gtfs <- merge_gtfs(gtfs_list, prefix = c("spo", "ggl"))
merged_gtfs$agency
```

---

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"
)
```

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 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.

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, as created by read_gtfs().

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)

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

# use the unit argument to change the speed unit
gtfs_new_speed <- set_trip_speed(
  gtfs,
  trip_id = "CPTM L07-0",
  speed = 15,
  unit = "m/s"
)
gtfs_new_speed$stop_times[trip_id == "CPTM L07-0"]

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

# when doing by reference, original gtfs is changed
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 feed*

Description

Uses MobilityData's GTFS validator to perform a GTFS business rule validation. The results are available as an HTML report (if validator v3.1.0 or higher is used) and in JSON format. Please check the complete set of rules used in the validation here. Please note that this function requires a working installation of Java 11 or higher to work.

Usage

```
validate_gtfs(
  gtfs,
  output_path,
  validator_path,
  overwrite = TRUE,
  html_preview = TRUE,
  pretty_json = FALSE,
  quiet = TRUE,
  n_threads = 1
)
```

Arguments

gtfs	The GTFS to be validated. Can be in the format of a GTFS object, of a path to a GTFS file, of a path to a directory or an URL to a feed.
output_path	A string. The path to the directory that the validator will create and in which the results will be saved to.
validator_path	A string. The path to the GTFS validator, previously downloaded with download_validator().
overwrite	A logical. Whether to overwrite existing validation results in output_path. Defaults to TRUE.
html_preview	A logical. Whether to show HTML report in a viewer, such as RStudio or a browser. Defaults to TRUE (only works on interactive sessions).
pretty_json	A logical. Whether JSON results should be printed in a readable way, that allows it to be inspected without manually formatting. Defaults to FALSE.
quiet	A logical. Whether to hide informative messages. Defaults to TRUE.
n_threads	An integer between 1 and the number of cores in the running machine. Control how many threads are used during the validation. Defaults to using all but one of the available cores.

Value

Invisibly returns the normalized path to the directory where the validation results were saved to.

See Also

Other validation: download_validator()

Examples

```
data_path <- system.file("extdata/spo_gtfs.zip", package = "gtfstools")
output_path <- tempfile("validation_result")
validator_path <- download_validator(tempdir())
gtfs <- read_gtfs(data_path)

validate_gtfs(gtfs, output_path, validator_path)
list.files(output_path)

# works with feeds saved to disk
new_output_path <- tempfile("new_validation_result")
validate_gtfs(data_path, new_output_path, validator_path)
list.files(new_output_path)

# and with feeds pointed by an url
newer_output_path <- tempfile("newer_validation_result")
gtfs_url <- "https://github.com/ipeaGIT/gtfstools/raw/main/inst/extdata/spo_gtfs.zip"
validate_gtfs(gtfs_url, newer_output_path, validator_path)
list.files(newer_output_path)
```

---

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
)
```

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).

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