Package 'GTFSwizard'

January 20, 2025

Type Package

Title Exploring and Manipulating 'GTFS' Files

Version 1.0.0

Date/Publication 2024-11-11 20:40:05 UTC

URL https://github.com/nelsonquesado/GTFSwizard

BugReports https://github.com/nelsonquesado/GTFSwizard/issues

Description Exploring, analyzing, and manipulating General Transit Feed Specification (GTFS) files, which represent public transportation schedules and geographic data. The package allows users to filter data by routes, trips, stops, and time, generate spatial visualizations, and perform detailed analyses of transit networks, including headway, dwell times, and route frequencies. Designed for transit planners, researchers, and data analysts, 'GTFSwizard' integrates functionalities from popular packages to enable efficient GTFS data manipulation and visualization.

Imports gtfstools, tidytransit, lubridate, sf, tidyr, data.table, shiny, leaflet, checkmate, dplyr, ggplot2, gtfsio, purrr, rlang, crayon, forcats, hrbrthemes, stringr, tibble, plotly, leaflet.extras, geosphere, stplanr, glue, hms, sfnetworks

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

LazyDataCompression xz

Config/testthat/edition 3

NeedsCompilation no

Author Nelson de O. Quesado Filho [aut, cre], Caio G. C. Guimarães [aut]

Maintainer Nelson de O. Quesado Filho <nquesado@gmail.com>

Repository CRAN

Contents

`as_wizardgtfs` *Convert GTFS Object to wizardgtfs Format*

Description

'as_wizardgtfs' transforms a GTFS object into the 'wizardgtfs' format, providing enhanced functionality and compatibility with the GTFSwizard package. This function supports GTFS objects in various formats, including 'tidygtfs' and list-based structures, and can optionally create a shapes table if it is missing.

Usage

```
as_wizardgtfs(gtfs_list, build_shapes = TRUE)
```

Arguments

gtfs_list	A GTFS object in list or ‘tidygtfs‘ format.
build_shapes	Logical. If ‘TRUE‘, builds the shapes table if it is missing in the provided GTFS object. Default is ‘TRUE‘.

Details

- ‘as_wizardgtfs‘ is a generic function with S3 methods for different GTFS object formats.

- The ‘wizardgtfs‘ format includes additional processing and checks, such as validation of unique IDs and structure formatting.

Value

An object of class ‘wizardgtfs‘, which includes multiple data frames for transit data analysis.

See Also

[GTFSwizard::get_shapes()]

Examples

```
# Convert a GTFS object to wizardgtfs format
gtfs_wizard <- as_wizardgtfs(for_rail_gtfs, build_shapes = TRUE)
```

---

delay_trip *Delay Specified Trips in a ‘wizardgtfs‘ Object*

Description

This function adds a delay to the arrival and departure times of specified trips within a ‘wizardgtfs‘ object. If the input GTFS object is not of class ‘wizardgtfs‘, it will be converted.

Usage

```
delay_trip(gtfs, trip, duration)
```

Arguments

gtfs	An object representing GTFS data, preferably of class ‘wizardgtfs‘.
trip	A character vector of ‘trip_id‘s in the ‘wizardgtfs‘ object that will be delayed. Each ‘trip_id‘ must exist in ‘gtfs$trips$trip_id‘.
duration	A delay duration, either as a ‘duration‘ object or a numeric value representing seconds.

Details

This function adjusts the arrival and departure times of the specified ‘trip_id‘s in ‘gtfs$stop_times‘ by the specified ‘duration‘. If ‘gtfs‘ is not a ‘wizardgtfs‘ object, the function will attempt to convert it using ‘GTFSwizard::as_wizardgtfs()‘, and a warning will be issued. The function checks that ‘trip‘ contains valid ‘trip_id‘s and that ‘duration‘ is either a ‘duration‘ or numeric (seconds).

Value

A modified ‘wizardgtfs‘ object with updated arrival and departure times for the specified trips.

See Also

[GTFSwizard::as_wizardgtfs()] for converting GTFS objects to ‘wizardgtfs‘ class.

Examples

```
# Delay trips by 5 minutes
gtfs <- delay_trip(gtfs = for_rail_gtfs, for_rail_gtfs$trips$trip_id[1:2], duration = 300)

# Delay trips by duration
gtfs <- delay_trip(gtfs = for_rail_gtfs,
                   trip = for_rail_gtfs$trips$trip_id[1],
                   duration = lubridate::duration(10, "minutes"))
```

---

explore_gtfs *Explore GTFS Data in an Interactive Shiny Application*

Description

This function pops-up a Shiny application for exploring General Transit Feed Specification (GTFS) data. The application provides an overview of the GTFS data, visualizations of route characteristics, and detailed information on selected routes, allowing users to analyze various aspects of a GTFS feed interactively.

Usage

```
explore_gtfs(gtfs)
```

Arguments

gtfs	A GTFS object, preferably of class ‘wizardgtfs‘. If not, the function attempts to convert it to ‘wizardgtfs‘ using ‘GTFSwizard::as_wizardgtfs()‘.

Details

The Shiny application generated by this function has two main tabs: - **Overview**: Displays general GTFS information, maps, and summary charts of the transit system, including frequency, fleet, speed, and other statistics. - **By Route**: Allows users to select specific routes and view detailed maps and visualizations for each selected route.

If the provided ‘gtfs‘ object does not contain a ‘shapes‘ table, it will attempt to add it using ‘GTFSwizard::get_shapes()‘, issuing a warning

Value

A Shiny app object that, when run, opens an interactive dashboard for GTFS data exploration.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_shapes()], [GTFSwizard::plot_calendar()]

Examples

```
if (interactive()) {
  # To run the Shiny application:
  explore_gtfs(gtfs = GTFSwizard::for_rail_gtfs)
}
```

---

filter_functions *Filter GTFS Data by Service, Route, Date, Stop, Trip, and Time*

Description

The ‘filter_‘ functions allow you to selectively filter data within a ‘wizardgtfs‘ object based on criteria such as service patterns, specific dates, service IDs, route IDs, trip IDs, stop IDs, or time ranges.

Usage

```
filter_servicepattern(gtfs, servicepattern = NULL)

filter_date(gtfs, dates = NULL)

filter_service(gtfs, service)

filter_route(gtfs, route, keep = TRUE)

filter_trip(gtfs, trip, keep = TRUE)

filter_stop(gtfs, stop)

filter_time(gtfs, from = "0:0:0", to = "48:00:00")
```

Arguments

gtfs	A GTFS object, preferably of class ‘wizardgtfs‘. If not, the function will attempt to convert it using ‘GTFSwizard::as_wizardgtfs()‘.
servicepattern	(Optional) A character vector of service patterns to retain. Defaults to the most frequent pattern (typical day) if ‘NULL‘.
dates	(Optional) A date or vector of dates (as "YYYY-MM-DD" character or POSIXct) to filter services active on those dates. Defaults to the last available date if ‘NULL‘.
service	(Optional) A character vector of service IDs to retain in the ‘wizardgtfs‘ object.
route	(Optional) A character vector of route IDs to retain in the ‘wizardgtfs‘ object. When ‘keep = FALSE‘, excludes the specified routes.
keep	Logical. When ‘TRUE‘ (default), retains specified ‘route‘ or ‘trip‘ IDs; when ‘FALSE‘, excludes them.
trip	(Optional) A character vector of trip IDs to retain in the ‘wizardgtfs‘ object. When ‘keep = FALSE‘, excludes the specified trips.
stop	(Optional) A character vector of stop IDs to retain.
from	(Optional) Start time in "HH:MM:SS" format to include only trips that start after this time. Defaults to ‘0:0:0‘.
to	(Optional) End time in "HH:MM:SS" format to include only trips that end before this time. Defaults to ‘48:00:00‘.

Details

Each ‘filter_‘ function targets a specific aspect of the GTFS data, applying filters to the relevant tables:

- filter_servicepattern: Filters by specified service patterns in the GTFS data. If no pattern is provided, defaults to the most frequent one.

- filter_date: Filters data by a date or dates, returning only services active on those dates.

- filter_service: Filters by service ID, retaining data related to specified services.

- filter_route: Filters by route ID. When ‘keep = TRUE‘, only specified routes are retained; when ‘FALSE‘, the specified routes are excluded.

- filter_trip: Filters by trip ID, using ‘keep‘ to either retain or exclude specified trips.

- filter_stop: Filters by stop ID, retaining only stops and related data (trips, routes, etc.) associated with the specified stops.

- filter_time: Filters stop times within a specified time range (between ‘from‘ and ‘to‘).

These functions selectively subset the GTFS tables (‘trips‘, ‘stop_times‘, ‘routes‘, ‘agency‘, ‘shapes‘, etc.), maintaining only the records that meet the defined criteria. If a table or required column is missing from the GTFS data, the function will either attempt to infer it using available data or exclude the table as necessary.

Value

A filtered ‘wizardgtfs‘ object containing only the records that match the specified criteria.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_servicepattern()]

Examples

```
## Not run:
# Filter by service pattern
filtered_gtfs <- filter_servicepattern(gtfs = for_gtfs, servicepattern = "servicepattern-2")

# Filter by a specific date
filtered_gtfs <- filter_date(gtfs = for_gtfs, dates = "2023-01-01")

# Filter by route ID, keeping only specified routes
filtered_gtfs <- filter_route(gtfs = for_gtfs, route = for_gtfs$routes$route_id[1:2])

# Filter by trip ID, excluding specified trips
filtered_gtfs <- filter_trip(gtfs = for_gtfs, trip = for_gtfs$trips$trip_id[1:2], keep = FALSE)

# Filter by a time range
filtered_gtfs <- filter_time(gtfs = for_gtfs, from = "06:30:00", to = "10:00:00")

## End(Not run)
```

---

for_bus_gtfs *GTFS Data for Fortaleza (Bus System), Brazil.*

Description

A dataset containing GTFS (General Transit Feed Specification) data for Fortaleza's transit system by bus. The data includes information on routes, trips, stops, stop times, and other elements necessary for transit planning and analysis.

Format

An object of class `wizardgtfs`, containing multiple data frames:

agency Data frame with 1 row and 7 columns, providing information about the transit agency, including agency name, URL, timezone, and contact details.

calendar Data frame with 3 rows and 10 columns, detailing service availability by day of the week, start and end dates for each service.

fare_attributes Data frame with 2 rows and 6 columns, showing fare information, including price, currency, payment method, and transfer rules.

fare_rules Data frame with 259 rows and 5 columns, linking fare IDs to routes, along with optional restrictions on origins, destinations, and zones.

routes Data frame with 259 rows and 9 columns, listing route details such as route ID, agency ID, route short and long names, route type, and colors.

shapes Data frame with 89,846 rows and 5 columns, representing the spatial paths of routes with latitude, longitude, point sequence, and cumulative distance traveled.

stop_times Data frame with 1,719,386 rows and 9 columns, including stop times for each trip, with arrival and departure times, stop sequence, and stop ID information.

stops Data frame with 4,793 rows and 12 columns, containing information about each stop, including stop ID, name, location (latitude and longitude), and accessibility.

trips Data frame with 52,304 rows and 9 columns, detailing trips associated with routes, including trip IDs, route IDs, direction, block, and shape IDs.

Details

The GTFS data format is widely used for representing public transportation schedules and associated geographic information. This dataset follows the GTFS standard and includes elements for advanced analysis in transit planning.

Source

Fortaleza transit agency (ETUFOR).

Examples

```
# Load the dataset
data(for_bus_gtfs)

# Access trips data
head(for_bus_gtfs$trips)

# Access stops data
head(for_bus_gtfs$stops)
```

---

for_rail_gtfs *GTFS Data for Fortaleza (Rail System), Brazil*

Description

This dataset contains GTFS (General Transit Feed Specification) data for Fortaleza's rail transit system, managed by METROFOR. The data includes information on routes, trips, stops, stop times, shapes, and other necessary elements for transit analysis and planning.

Format

An object of class `wizardgtfs`, consisting of multiple data frames:

agency Data frame with 1 row and 7 columns, providing information about the transit agency, including agency name, URL, timezone, language, and contact details.

calendar Data frame with 1 row and 10 columns, detailing the service availability by day of the week, along with start and end dates for each service.

calendar_dates Data frame with 26 rows and 3 columns, listing specific dates and exceptions (e.g., holidays) that modify the usual service pattern.

routes Data frame with 3 rows and 9 columns, listing route details such as route ID, short and long names, route type, and colors associated with each route.

stops Data frame with 39 rows and 10 columns, containing information about each stop, including stop ID, name, location (latitude and longitude), and additional details.

stop_times Data frame with 3,420 rows and 10 columns, detailing arrival and departure times for each trip, along with stop sequences and stop IDs.

trips Data frame with 215 rows and 7 columns, providing trip-specific information such as trip ID, headsign, direction, associated service ID, route ID, and shape ID.

shapes Data frame with 80 rows and 5 columns, representing spatial paths of routes using latitude, longitude, point sequence, and cumulative distance traveled.

Details

The GTFS data format is widely adopted for representing public transportation schedules and spatial information. This dataset follows GTFS standards and is tailored for advanced analysis, particularly in transit planning and operations. Key tables included are ‘agency‘, ‘routes‘, ‘stops‘, ‘stop_times‘, ‘trips‘, and ‘shapes‘, each providing essential attributes for a comprehensive transit analysis.

Source

Cia Cearense de Transportes Metropolitanos (METROFOR).

Examples

```
# Load the dataset
data(for_rail_gtfs)

# Access trips data
head(for_rail_gtfs$trips)

# Access stops data
head(for_rail_gtfs$stops)
```

---

get_distances *Calculate Distances in GTFS Data*

Description

The ‘get_distances‘ function calculates distances within a ‘wizardgtfs‘ object based on various methods. Depending on the ‘method‘ chosen, it can calculate average route distances, trip-specific distances, or detailed distances between stops.

Usage

```
get_distances(gtfs, method = "by.route")
```

Arguments

gtfs	A GTFS object, ideally of class ‘wizardgtfs‘. If it is not of this class, it will be converted.
method	A character string indicating the calculation method. Choices are:
	"by.route" Calculates average distances for each route.
	"by.trip" Calculates distances for each trip, associating each trip ID with its total distance.
	"detailed" Calculates detailed distances between each consecutive stop for all trips. This is the most computationally intensive option and may take several minutes to complete.

Details

The function calls specific sub-functions based on the selected method:

- "by.route": Calculates average distances per route.

- "by.trip": Calculate distances per trip.

- "detailed": Calculates detailed stop-to-stop distances within each route. Note that this method may be slow for large datasets.

If an invalid ‘method‘ is provided, the function defaults to ‘"by.route"‘ and issues a warning.

Value

A data frame with calculated distances based on the specified method:

If ‘method = "by.route"‘ Returns a summary with columns: ‘route_id‘, ‘trips‘, ‘average.distance‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.trip"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘distance‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "detailed"‘ Returns a data frame with columns: ‘shape_id‘, ‘from_stop_id‘, ‘to_stop_id‘, and ‘distance‘.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_servicepattern()]

Examples

```
# Calculate average route distances
distances_by_route <- get_distances(gtfs = for_rail_gtfs, method = "by.route")

# Calculate distances by trip
distances_by_trip <- get_distances(gtfs = for_rail_gtfs, method = "by.trip")


# Calculate detailed distances between stops
detailed_distances <- get_distances(gtfs = for_rail_gtfs, method = "detailed")
```

get_durations *Calculate Trip Durations in GTFS Data*

Description

The ‘get_durations‘ function calculates trip durations within a ‘wizardgtfs‘ object using different methods. Depending on the selected ‘method‘, it can provide average durations per route, durations for individual trips, or detailed segment durations between stops.

Usage

```
get_durations(gtfs, method = "by.route")
```

Arguments

gtfs	A GTFS object, ideally of class ‘wizardgtfs‘. If not, it will be converted.
method	A character string specifying the calculation method. Options include:
	"by.route" Calculates the average duration for each route.
	"by.trip" Calculates the total duration for each trip.
	"detailed"‘ Calculates detailed durations for each stop-to-stop segment within a trip.

Details

This function calls specific sub-functions based on the selected method:

- "by.route": Calculates average durations for each route.

- "by.trip": Calculates the total duration of each trip.

- "detailed": Calculates detailed durations between consecutive stops within each trip.

If an invalid ‘method‘ is specified, the function defaults to ‘"by.route"‘ and provides a warning.

Value

A data frame containing trip durations based on the specified method:

If ‘method = "by.route"‘ Returns a summary data frame with columns: ‘route_id‘, ‘trips‘, ‘average.duration‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.trip"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘duration‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "detailed"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘hour‘, ‘from_stop_id‘, ‘to_stop_id‘, ‘duration‘, ‘service_pattern‘, and ‘pattern_frequency‘.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_servicepattern()]

Examples

```
# Calculate average route durations
durations_by_route <- get_durations(gtfs = for_rail_gtfs, method = "by.route")

# Calculate trip durations
durations_by_trip <- get_durations(gtfs = for_rail_gtfs, method = "by.trip")

# Calculate detailed durations between stops
detailed_durations <- get_durations(gtfs = for_rail_gtfs, method = "detailed")
```

`get_dwelltimes` *Calculate Dwell Times in GTFS Data*

Description

The ‘get_dwelltimes‘ function calculates dwell times within a ‘wizardgtfs‘ object using different methods. Depending on the selected ‘method‘, it can provide average dwell times per route, per trip, by hour, or detailed dwell times at each stop.

Usage

```
get_dwelltimes(gtfs, max.dwelltime = 90, method = "by.route")
```

Arguments

`gtfs`	A GTFS object, ideally of class ‘wizardgtfs‘. If not, it will be converted.
`max.dwelltime`	Numeric. The maximum allowable dwell time (in seconds). Dwell times exceeding this value are excluded from the calculations. Defaults to 90 seconds.
`method`	A character string specifying the calculation method. Options include:
	"by.hour" Calculates the average dwell time per hour of the day across all trips.
	"by.route" Calculates the average dwell time for each route.
	"by.trip" Calculates the average dwell time for each trip.
	"detailed" Calculates detailed dwell times at each stop within every trip.

Details

This function calls specific sub-functions based on the selected method:

- "by.hour": Calculates the average dwell time for each hour of the day.

- "by.route": Calculates average dwell times across each route.

- "by.trip": Calculates the total dwell time for each trip.

- "detailed": Calculates the dwell time between consecutive stops within each trip.

If an invalid ‘method‘ is specified, the function defaults to ‘"by.route"‘ and provides a warning.

Value

A data frame containing dwell times based on the specified method:

If ‘method = "by.hour"‘ Returns a data frame with columns: ‘hour‘, ‘trips‘, ‘average.dwelltime‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.route"‘ Returns a data frame with columns: ‘route_id‘, ‘trips‘, ‘average.dwelltime‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.trip"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘average.dwelltime‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "detailed"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘stop_id‘, ‘hour‘, ‘dwell_time‘, ‘service_pattern‘, and ‘pattern_frequency‘.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_servicepattern()]

Examples

```
# Calculate dwell times by hour
dwelltimes_by_hour <- get_dwelltimes(gtfs = for_rail_gtfs, max.dwelltime = 120, method = "by.hour")

# Calculate dwell times by route
dwelltimes_by_route <- get_dwelltimes(gtfs = for_rail_gtfs, max.dwelltime = 90, method = "by.route")

# Calculate dwell times by trip
dwelltimes_by_trip <- get_dwelltimes(gtfs = for_rail_gtfs, max.dwelltime = 45, method = "by.trip")

# Calculate detailed dwell times between stops
detailed_dwelltimes <- get_dwelltimes(gtfs = for_rail_gtfs, max.dwelltime = 60, method = "detailed")
```

---

get_fleet *Estimates Fleet from GTFS Data*

Description

The ‘get_fleet‘ function estimates the fleet from a ‘wizardgtfs‘ object using different methods. Depending on the selected ‘method‘, it can estimates fleet by route, by hour, peak times, or detailed timepoints.

Usage

```
get_fleet(gtfs, method = "by.route")
```

Arguments

gtfs	A GTFS object, ideally of class ‘wizardgtfs‘. If not, it will be converted.
method	A character string specifying the calculation method. Options include:

"by.route" Calculates the maximum number of simultaneous trips for each route.

"by.hour" Calculates the maximum number of simultaneous trips by hour of the day across all routes.

"peak" Calculates the maximum number of simultaneous trips for the three busiest hours.

"detailed" Calculates the maximum number of simultaneous trips across each timepoint within a trip.

Details

This function calls specific sub-functions based on the selected method:

- "by.route": Calculates the maximum simultaneous trips per route.

- "by.hour": Calculates the maximum simultaneous trips for each hour of the day.

- "peak": Calculates the maximum simultaneous trips for the three busiest hours.

- "detailed": Provides a timepoint-based fleet calculation, showing detailed fleet fluctutations over the course of the trip.

If an invalid ‘method‘ is specified, the function defaults to ‘"by.route"‘ and provides a warning.

Value

A data frame containing the fleet based on the specified method:

If ‘method = "by.route"‘ Returns a data frame with columns: ‘route_id‘, ‘fleet‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.hour"‘ Returns a data frame with columns: ‘hour‘, ‘fleet‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "peak"‘ Returns a data frame with columns: ‘hour‘, ‘fleet‘, ‘service_pattern‘, and ‘pattern_frequency‘ for the busiest three hours.

If ‘method = "detailed"‘ Returns a data frame with columns: ‘route_id‘, ‘net.fleet‘, ‘fleet‘, ‘time‘, ‘service_pattern‘, and ‘pattern_frequency‘ for each timepoint.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_servicepattern()]

Examples

```
# Calculate fleet requirements by route
fleet_by_route <- get_fleet(gtfs = for_rail_gtfs, method = "by.route")

# Calculate fleet requirements by hour
fleet_by_hour <- get_fleet(gtfs = for_rail_gtfs, method = "by.hour")
```

```
# Calculate fleet requirements for peak hours
fleet_peak <- get_fleet(gtfs = for_rail_gtfs, method = "peak")

# Calculate detailed fleet requirements over timepoints
fleet_detailed <- get_fleet(gtfs = for_rail_gtfs, method = "detailed")
```

`get_frequency` *Calculate Route Frequency in GTFS Data*

Description

The ‘get_frequency‘ function calculates route frequency within a ‘wizardgtfs‘ object using different methods. Depending on the selected ‘method‘, it can provide daily frequencies by route or detailed hourly frequencies.

Usage

```
get_frequency(gtfs, method = "by.route")
```

Arguments

`gtfs`	A GTFS object, ideally of class ‘wizardgtfs‘. If not, it will be converted.
`method`	A character string specifying the calculation method. Options include:
	"by.route" Calculates the total daily frequency for each route.
	"detailed" Calculates the hourly frequency for each route.

Details

This function calls specific sub-functions based on the selected method:

- "by.route": Calculates the total daily frequency for each route, summing up the number of trips for each route on a daily basis.

- "detailed": Provides an hourly breakdown of frequency, showing the number of departures per hour for each route.

If an invalid ‘method‘ is specified, the function defaults to ‘"by.route"‘ and provides a warning.

Value

A data frame containing route frequencies based on the specified method:

If ‘method = "by.route"‘ Returns a data frame with columns: ‘route_id‘, ‘daily.frequency‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "detailed"‘ Returns a data frame with columns: ‘route_id‘, ‘hour‘, ‘frequency‘, ‘service_pattern‘, and ‘pattern_frequency‘.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_servicepattern()]

Examples

```
# Calculate daily route frequency
frequency_by_route <- get_frequency(gtfs = for_rail_gtfs, method = "by.route")

# Calculate detailed hourly frequency
detailed_frequency <- get_frequency(gtfs = for_rail_gtfs, method = "detailed")
```

---

get_headways *Calculate Headways in GTFS Data*

Description

The ‘get_headways‘ function calculates headways within a ‘wizardgtfs‘ object using different methods. Depending on the selected ‘method‘, it can provide average headways by route, by trip, by hour, or detailed stop-level headways.

Usage

```
get_headways(gtfs, method = "by.route")
```

Arguments

gtfs	A GTFS object, ideally of class ‘wizardgtfs‘. If not, it will be converted.
method	A character string specifying the calculation method. Options include:
	"by.route" Calculates the average headway for each route, assuming constant headways along stops.
	"by.hour" Calculates the hourly headway for each route, assuming constant headways along stops.
	"by.trip" Calculates headways for each trip, assuming constant headways along stops.
	"detailed" Calculates detailed headways between consecutive stops within each route and trip.

Details

This function calls specific sub-functions based on the selected method:

- "by.route": Calculates the average headway for each route based on the first stop time per trip.

- "by.hour": Calculates the hourly headway for each route, grouping trips by hour.

- "by.trip": Calculates headways for each trip, considering only the first stop time.

- "detailed": Provides headway calculations for each consecutive stop within each trip.

If an invalid ‘method‘ is specified, the function defaults to ‘"by.route"‘ and provides a warning.

Value

A data frame containing service headways based on the specified method:

If ‘method = "by.route"‘ Returns a data frame with columns: ‘route_id‘, ‘trips‘, ‘average.headway‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.hour"‘ Returns a data frame with columns: ‘hour‘, ‘trips‘, ‘average.headway‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.trip"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘headway‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "detailed"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘stop_id‘, ‘hour‘, ‘headway‘, ‘service_pattern‘, and ‘pattern_frequency‘.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_servicepattern()]

Examples

```
# Calculate average route headways
headways_by_route <- get_headways(gtfs = for_rail_gtfs, method = "by.route")

# Calculate hourly headways
headways_by_hour <- get_headways(gtfs = for_rail_gtfs, method = "by.hour")

# Calculate headways for each trip
headways_by_trip <- get_headways(gtfs = for_rail_gtfs, method = "by.trip")

# Calculate detailed stop-level headways
detailed_headways <- get_headways(gtfs = for_rail_gtfs, method = "detailed")
```

---

get_servicepattern *Identify Service Patterns in GTFS Data*

Description

The ‘get_servicepattern‘ function identifies and organizes unique service patterns within a ‘wizardgtfs‘ object. It groups services by common dates of operation and assigns each a frequency-based pattern identifier.

Usage

```
get_servicepattern(gtfs)
```

Arguments

gtfs	A GTFS object, ideally of class ‘wizardgtfs‘. If not, it will be converted.

Details

The function first checks if the input ‘gtfs‘ object is of class ‘wizardgtfs‘. If not, it converts it using ‘as_wizardgtfs()‘. It then groups services by common dates of operation, assigns a frequency to each unique pattern, and organizes these into service pattern identifiers, ordered by their frequency.

Value

A data frame containing unique service patterns with the following columns:

‘service_id‘ Unique identifier(s) for each service.

‘service_pattern‘ An identifier for each distinct service pattern based on operational dates, in the format "servicepattern-N".

‘pattern_frequency‘ The frequency of each service pattern, indicating the number of dates associated with that pattern.

See Also

[GTFSwizard::as_wizardgtfs()]

Examples

```
# Generate service patterns for a GTFS object
service_patterns <- get_servicepattern(gtfs = for_rail_gtfs)
```

`get_shapes` *Generate Shapes Table for GTFS Data*

Description

The ‘get_shapes‘ function reconstructs the ‘shapes‘ table for a GTFS dataset using an approximation based on stop coordinates and sequence information. It creates geometric representations of trips by connecting stops in sequence for each trip.

Usage

```
get_shapes(gtfs)
```

Arguments

`gtfs`	A GTFS object, ideally of class ‘wizardgtfs‘. If not, it will be converted automatically.

Details

This function constructs the ‘shapes‘ table by sequentially connecting stops along each trip using a Euclidean approximation. If the GTFS object already contains a ‘shapes‘ table, it will be overwritten, and a warning will be displayed. The process involves:

- Selecting and arranging stops by trip and sequence

- Connecting stops with line segments to form a path for each trip

- Grouping unique paths into distinct shape IDs

Value

A modified GTFS object that includes a ‘shapes‘ table derived from the stops and trips information.

Note

This approximation may not perfectly represent real-world shapes, especially for complex or curved routes. ‘get_shapes()‘ uses stop sequences to recriate the shapes table; accordingly, it should not be used after ‘filter_time()‘, as this function removes invalid ‘stop_times‘.

See Also

[GTFSwizard::as_wizardgtfs()], [GTFSwizard::get_shapes_df()]

Examples

```
# Generate a shapes table for a GTFS object
gtfs_with_shapes <- get_shapes(gtfs = for_rail_gtfs)
```

---

get_shapes_df *Convert Shape Geometries to GTFS Shape Points Data Frame*

Description

The ‘get_shapes_df‘ function converts a spatial object of shapes (with geometry) into a GTFS-compliant ‘shapes‘ data frame format, detailing latitude, longitude, point sequence, and cumulative distance traveled along each shape.

Usage

```
get_shapes_df(shape)
```

Arguments

shape	A spatial (‘sf‘) object containing shapes, with ‘shape_id‘ and geometry information.

Details

The function performs the following steps:

- Validates that the ‘shape‘ object is of class ‘sf‘ and contains a ‘shape_id‘ column.

- Extracts point coordinates from each shape's geometry, creating a sequence of latitude and longitude points.

- Computes cumulative distances along the shape, using Euclidean distance between consecutive points.

The resulting data frame conforms to the GTFS ‘shapes.txt‘ format. Distances are expressed in meters.

Value

A data frame with columns:

‘shape_id‘ Unique identifier for each shape.

‘shape_pt_lon‘ Longitude coordinates of each shape point.

‘shape_pt_lat‘ Latitude coordinates of each shape point.

‘shape_pt_sequence‘ Sequence of points along each shape.

‘shape_dist_traveled‘ Cumulative distance traveled along the shape in meters.

See Also

[GTFSwizard::get_shapes()], [GTFSwizard::get_shapes_sf()]

Examples

```
# Convert a shape geometry to a GTFS-compliant shapes data frame
shape <- get_shapes_sf(for_rail_gtfs$shapes)
shapes_df <- get_shapes_df(shape = shape)
```

`get_shapes_sf` *Convert GTFS Shapes Table to Simple Features (sf) Format*

Description

‘get_shapes_sf‘ converts the shapes table in a ‘wizardgtfs‘ object into a simple features (‘sf‘) object, making it suitable for spatial analysis. This function checks and processes the ‘shapes‘ data in the provided GTFS object and structures it as ‘LINESTRING‘ features.

Usage

```
get_shapes_sf(gtfs)
```

Arguments

gtfs	A GTFS object containing the ‘shapes‘ table or the shape table itself. If the ‘shapes‘ table is missing, it will be created using ‘get_shapes()‘.

Details

- When the input ‘wizardgtfs‘ object lacks a ‘shapes‘ table, the function automatically generates one using ‘get_shapes()‘.

- The ‘shapes‘ table in the GTFS object are transformed into ‘LINESTRING‘ geometries. If ‘shape_pt_sequence‘ is absent, the points are treated as ordered by their position in the data.

- If ‘shape_dist_traveled‘ is available, cumulative distance calculations are included for each shape point.

Value

An ‘sf‘ object with shapes as ‘LINESTRING‘ geometries:

Note

If ‘shape_pt_sequence‘ is missing, the function will assume that points are ordered, constructing the shape accordingly.

See Also

[GTFSwizard::get_shapes()], [GTFSwizard::get_shapes_df()]

Examples

```
# Convert shapes data in a GTFS object to sf format
gtfs_sf <- get_shapes_sf(for_rail_gtfs)
```

---

get_speeds *Calculate Speeds for GTFS Routes and Trips*

Description

‘get_speeds‘ calculates the average speed of trips and routes within a ‘wizardgtfs‘ object. It uses distance and duration to provide speed outputs based on the specified ‘method‘.

Usage

```
get_speeds(gtfs, method = "by.route")
```

Arguments

gtfs	A GTFS object, ideally of class ‘wizardgtfs‘. If the ‘shapes‘ table is missing, it will be created automatically using ‘get_shapes()‘.
method	A character string specifying the calculation method. Options include:
	"by.route" Calculates the average speed for each route based on average distance and duration.
	"by.trip" Calculates the average speed for each trip based on total distance and duration.
	"detailed" Calculates the speed for each segment between stops within a trip.

Details

- This function calls specific sub-functions based on the selected ‘method‘:

‘by.route‘ Calculates average speed across each route.

‘by.trip‘ Calculates average speed across each trip.

‘detailed‘ Calculates speeds between consecutive stops within each trip.

- If an invalid ‘method‘ is specified, the function defaults to ‘"by.route"‘ and provides a warning.

Value

A data frame containing speed calculations, depending on the specified method:

If ‘method = "by.route"‘ Returns a data frame with columns: ‘route_id‘, ‘trips‘, ‘average.speed‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "by.trip"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘average.speed‘, ‘service_pattern‘, and ‘pattern_frequency‘.

If ‘method = "detailed"‘ Returns a data frame with columns: ‘route_id‘, ‘trip_id‘, ‘hour‘, ‘from_stop_id‘, ‘to_stop_id‘, ‘speed‘, ‘service_pattern‘, and ‘pattern_frequency‘.

See Also

[GTFSwizard::get_distances()], [GTFSwizard::get_durations()], [GTFSwizard::get_shapes()]

Examples

```
# Calculate average route speeds
speeds_by_route <- get_speeds(gtfs = for_rail_gtfs, method = "by.route")

# Calculate trip speeds
speeds_by_trip <- get_speeds(gtfs = for_rail_gtfs, method = "by.trip")


# Calculate detailed speeds between stops
detailed_speeds <- get_speeds(gtfs = for_rail_gtfs, method = "detailed")
```

`get_stops_sf` *Convert GTFS Stops Table to Simple Features (sf) Format*

Description

'get_stops_sf' converts the stops table in a 'wizardgtfs' object into a simple features ('sf') object, making it suitable for spatial analysis. This function checks the format of the 'stops' data and structures it as point geometries.

Usage

```
get_stops_sf(gtfs)
```

Arguments

`gtfs`	A 'wizardgtfs' object containing a 'stops' table or the stops table itself as a data frame.

Details

- When the input 'stops' table is not in 'sf' format, this function converts it to 'sf' by using the coordinates in the 'stop_lon' and 'stop_lat' columns.

- The resulting 'sf' object is assigned a CRS of WGS 84 (EPSG:4326) for geographic compatibility.

- If the 'stops' table is already in 'sf' format, the function simply reassigns the CRS and returns it unchanged.

Value

An 'sf' object with stops as point geometries or a 'wizardgtfs' object.

See Also

[GTFSwizard::get_shapes()], [GTFSwizard::get_shapes_sf()], [GTFSwizard::get_shapes_df()]

Examples

```
# Convert stops data in a GTFS object to sf format
gtfs_sf <- get_stops_sf(for_rail_gtfs)
```

`merge_gtfs` *Merge Two GTFS Datasets*

Description

'merge_gtfs' combines two GTFS datasets into a single 'wizardgtfs' object, with an option to append suffixes to ensure unique identifiers across tables.

Usage

```
merge_gtfs(gtfs.x, gtfs.y, suffix = TRUE)
```

Arguments

`gtfs.x`	The first GTFS dataset, ideally of class 'wizardgtfs'. If not, it will be converted.
`gtfs.y`	The second GTFS dataset, ideally of class 'wizardgtfs'. If not, it will be converted.
`suffix`	A logical value. If 'TRUE', appends '.x' and '.y' suffixes to identifier columns in 'gtfs.x' and 'gtfs.y', respectively, to prevent conflicts.

Details

- When 'suffix = TRUE', unique suffixes are appended to key identifiers in 'gtfs.x' and 'gtfs.y' (e.g., 'agency_id', 'route_id', 'trip_id').

- After suffix handling, the function merges individual tables, ensuring no duplicated entries.

- Finally, the resulting list is converted into a 'wizardgtfs' object.

Value

A merged 'wizardgtfs' object containing all records from 'gtfs.x' and 'gtfs.y' across GTFS tables.

Note

This function assumes that both input datasets follow GTFS structure. Non-standard tables or columns may be ignored or cause warnings.

See Also

[GTFSwizard::as_wizardgtfs()]

Examples

```
# Merge two GTFS datasets with suffix handling
merged_gtfs <- merge_gtfs(for_rail_gtfs, for_bus_gtfs, suffix = TRUE)
```

plot_calendar *Plot Trip Frequency Calendar for GTFS Data*

Description

'plot_calendar' creates a calendar heatmap visualization of the number of trips in a GTFS dataset for each day, with options for monthly and yearly faceting.

Usage

```
plot_calendar(gtfs, ncol = 6, facet_by_year = FALSE)
```

Arguments

gtfs	A GTFS object, ideally of class 'wizardgtfs'. If not, it will be converted.
ncol	Number of columns for monthly faceting. Ignored if 'facet_by_year = TRUE'.
facet_by_year	Logical value. If 'TRUE', plots data by year with each month in a separate column.

Details

- The function calculates daily trip frequencies from the 'service_id' and 'dates_services' tables in the GTFS object.

- Days with no trips are marked in black, while other days are shaded on a gradient from pink (low trip count) to red (high trip count).

- If 'facet_by_year = TRUE', the plot will display each year in separate rows, and 'ncol' is automatically set to zero.

Value

A 'ggplot2' object showing a calendar heatmap of the daily trip counts across the specified GTFS date range.

See Also

[GTFSwizard::as_wizardgtfs()]

Examples

```
# Plot a GTFS trip calendar with 4 columns
plot_calendar(for_rail_gtfs, ncol = 4)

# Plot a GTFS trip calendar, faceting by year
plot_calendar(for_rail_gtfs, facet_by_year = TRUE)
```

plot_frequency *Plot System Frequency by Hour*

Description

'plot_frequency' generates an interactive plot of the frequency of trips by hour across the GTFS dataset. The plot shows hourly trip distributions, hourly average frequency, and an overall average frequency for the system, providing insights into peak times and overall transit service frequency.

Usage

```
plot_frequency(gtfs)
```

Arguments

gtfs	A GTFS object. This should ideally be of the 'wizardgtfs' class, or it will be converted.

Details

The function first calculates hourly and overall average frequencies using a weighted mean based on 'pattern_frequency'. Frequencies are plotted by hour of the day to visualize the system's trip distribution patterns.

Value

A 'plotly' interactive plot displaying hourly frequency distributions, including:

- Hourly Distribution: Boxplots showing frequency distribution across hours.

- Hourly Average Frequency: A line indicating the weighted average frequency for each hour.

- Overall Average Frequency: A dashed line marking the system's overall average frequency.

See Also

[GTFSwizard::get_frequency()]

Examples

```
if (interactive()) {
# Plot the frequency of trips by hour for a GTFS object
plot_frequency(for_rail_gtfs)
}
```

`plot_headways` *Plot System Average Headway by Hour*

Description

'plot_headways' generates an interactive plot of the average headways (time between trips) by hour across the GTFS dataset. The plot displays hourly headway distributions for each service pattern and includes an overall average headway line.

Usage

```
plot_headways(gtfs)
```

Arguments

`gtfs`	A GTFS object. This should ideally be of the 'wizardgtfs' class, or it will be converted.

Details

The function calculates hourly and overall average headways by weighting 'pattern_frequency' and 'trips' for each service pattern. The plot provides a visual representation of how average headways vary by hour and across service patterns.

Value

A 'plotly' interactive plot showing the hourly average headway (in minutes) across service patterns, including:

- Service Pattern Distribution: Lines for each service pattern, showing hourly headway values.

- Overall Average Headway: A dashed line marking the weighted overall average headway.

See Also

[GTFSwizard::get_headways()]

Examples

```
if (interactive()) {
# Plot average headway by hour for a GTFS object
plot_headways(for_rail_gtfs)
}
```

plot_routefrequency *Plot Route Frequency by Hour*

Description

'plot_routefrequency' generates an interactive plot of the frequency of trips by hour for specified routes in a GTFS dataset. The plot shows the hourly frequency distribution for each route and visualizes different service patterns.

Usage

```
plot_routefrequency(gtfs, route = NULL)
```

Arguments

gtfs	A GTFS object. Ideally, this should be of the 'wizardgtfs' class, or it will be converted.
route	A character vector specifying one or more 'route_id' values to plot. If 'NULL', all routes are included.

Details

The function filters the GTFS dataset by route and computes hourly frequencies for each service pattern. The plot shows variations in service frequency across hours and highlights the primary service pattern.

Value

A 'plotly' interactive plot displaying the frequency distribution by hour for each selected route, with:

- Hourly Frequency: A line for each route, indicating its frequency distribution across the day.

- Service Patterns: Transparency levels indicate different service patterns, with the primary pattern highlighted.

See Also

[GTFSwizard::filter_route()], [GTFSwizard::get_frequency()]

Examples

```
if (interactive()) {
# Plot frequency by hour for specific routes
plot_routefrequency(for_rail_gtfs, route = for_rail_gtfs$routes$route_id[1:2])
}
```

read_gtfs *Read GTFS file*

Description

Reads GTFS files from a .zip file.

Usage

```
read_gtfs(file.path, files = NULL, quiet = TRUE, ...)
```

Arguments

file.path	A path to a .zip GTFS file.
files	A character vector containing the text files to be read from the GTFS zip (without the .txt extension). Defaults to NULL, which reads all files.
quiet	Logical. If TRUE, suppresses messages from gtfsio::import_gtfs(). Defaults to TRUE.
...	Additional arguments to pass to gtfsio::import_gtfs().

Details

If no specific files are indicated, all GTFS files within the zip archive are read. After importing, the function converts the GTFS data into a ‘wizardgtfs‘ object, which is tailored for efficient handling and analysis of transit data.

Value

A ‘wizardgtfs‘ object: a list of tibbles representing each text file in the .zip and a tibble for services by date.

Note

Additional notes can be added here if needed.

See Also

[GTFSwizard::as_wizardgtfs()]

Examples

```
## Not run:
gtfs_data <- read_gtfs("path/to/gtfs.zip")

## End(Not run)
```

`selection` *Select Subsets of GTFS Data*

Description

The "selection" function makes a selection in the GTFS file without altering or filtering the GTFS file.

Usage

```
selection(gtfs, ..., add = FALSE)

unselection(gtfs)
```

Arguments

`gtfs`	An object representing GTFS data. It can be a list or a 'wizardgtfs' class gtfsect.

`...`	Expressions used to filter the data within 'gtfs'. The expressions can operate on four GTFS variables:
	`"stop_id"` Select the GTFS by stops using a vector of stop_id, must be character.
	`"route_id"` Select the GTFS by routes using a vector of route_id, must be character.
	`"trip_id"` Select the GTFS by trip using a vector of trip_id, must be character.
	`"geometry"` Select the GTFS by stops using an 'sf', 'sfc', or 'sfg' object. The geometry predicate function is evaluated with the geometry of the GTFS stops. Available predicates are:
	`%intersects%`
	`%touches%`
	`%within%`
	`%equals%`
	`%overlaps%`
	`%contains%`.

`add`	A logical argument. If 'TRUE', appends the new selection to existing ones in the gtfsect; otherwise, creates a new selection.

Details

The function evaluates the provided expressions in an environment restricted to recognized variables ('stop_id', 'route_id', 'trip_id', 'geometry'). An error is thrown if an unrecognized variable is used, indicating that only specific variables are allowed.

Value

A 'wizardgtfs_selected' wizardgtfs, which is a modified version of the original attributes with the selections applied. If the expression yields no matches, returns the original gtfs unchanged.

Examples

```
# Apply the selection function
result <- selection(for_rail_gtfs,
 stop_id == for_rail_gtfs$stops$stop_id[1] & trip_id %in% for_rail_gtfs$trips$trip_id[1:5])

# Check the selection
class(result)
attr(result, 'selection')

# Use geometry selection
bbox <- sf::st_bbox(c(
  xmin = -38.57219059002416,
  ymin = -3.7999496173114118,
  xmax = -38.50455165901261,
  ymax = -3.756631724636505
),
crs = sf::st_crs(4326))  # Set CRS to WGS 84

# Convert the bounding box to a polygon
polygon <- sf::st_as_sfc(bbox)

result <- for_rail_gtfs |> selection(geometry %intersects% polygon)
```

---

split_trip *Split a Trip into Sub-Trips within a GTFS Object*

Description

'split_trip' divides a specified trip in a 'wizardgtfs' object into multiple sub-trips by updating the stop sequences, trip identifiers, and related data, allowing for analysis or adjustments to different segments of the original trip.

Usage

```
split_trip(gtfs, trip, split = 1)
```

Arguments

gtfs	A GTFS object, ideally of class 'wizardgtfs'. If not, it will be converted.
trip	A character vector specifying the 'trip_id' to be split.
split	An integer indicating the number of splits to apply. One split means two trip segments.

Details

- The function creates sub-trips by dividing the specified trip(s) into equal parts based on the stop sequence.

- New trip IDs are generated for each sub-trip, and ‘stop_times‘, ‘trips‘, ‘frequencies‘, and ‘transfers‘ tables are updated accordingly.

- If ‘shape_dist_traveled‘ is present, it is adjusted to reflect distances within each new sub-trip.

- After the split, the function re-generates the shapes table for the new trips using ‘get_shapes‘, and merges it back into the ‘wizardgtfs‘ object.

- Be aware: ‘get_shapes‘ reconstructs shapes using euclidean approximation and may not be accurate.

Value

A GTFS object with the specified trip split into new sub-trips.

Note

‘split_trip()‘ uses stop sequences to recriate the shapes table of split trips; accordingly, it should not be used after ‘filter_time()‘, as this function removes invalid ‘stop_times‘.

See Also

[GTFSwizard::get_shapes()], [GTFSwizard::merge_gtfs()]

Examples

```
# Split a trip into 3 segments
gtfs_split <- split_trip(for_rail_gtfs, trip = for_rail_gtfs$trips$trip_id[1:3], split = 2)
```

---

write_gtfs *Write GTFS Data to Zip File*

Description

‘write_gtfs‘ exports a GTFS object to a zip file format, suitable for use in various GTFS-compatible software. This function supports multiple GTFS object formats and ensures compatibility by converting data frames and spatial objects as needed.

Usage

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

Arguments

gtfs	A GTFS object. This can be in ‘wizardgtfs‘ or list format.
zipfile	A character string specifying the path to the output zip file.
...	Additional arguments to pass to ‘gtfsio::export_gtfs()‘.

Details

The function converts spatial data frames (e.g., shapes and stops) to standard data frames, removes additional service pattern tables, and exports.

Value

None. This function writes the GTFS data directly to the specified ‘zipfile‘.

See Also

[GTFSwizard::read_gtfs()], [GTFSwizard::as_wizardgtfs()],

Index