Package 'GTFSwizard'

June 22, 2026

Type Package

Title Creating, Exploring and Manipulating 'GTFS' Files

Version 1.2.0

Author Nelson de Oliveira Quesado Filho [aut, cre],
Caio Gustavo Coelho Guimarães [aut],
Francisco Moraes de Oliveira Neto [aut]

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

Date 2026-06-16

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

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

Description Creating, 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, service dates, and time, generate spatial visualizations, and perform detailed analyses of transit networks, including headway, dwell times, route frequencies, service span, scheduled vehicle-hours, and trip duration. Methods follow common public transport planning and operation concepts described in Ceder (2007, ISBN:978-0-7506-6166-6), Vuchic (2005, ISBN:978-0-471-63265-8), and Vuchic (2007, ISBN:978-0-471-75823-5).

Imports sf, tidyr, checkmate, dplyr, ggplot2, gtfsio, rlang, tibble

Suggests data.table, gtfstools, hms, leaflet, shiny, testthat, tidytransit

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Repository CRAN

Date/Publication 2026-06-22 15:20:02 UTC

Contents

as_wizardgtfs . . . 3
create_gtfs . . . 4
delay_trip . . . 5
edit_dwelltime . . . 6
edit_speed . . . 7
explore_gtfs . . . 8
filter_servicepattern . . . 8
for_bus_gtfs . . . 10
for_rail_gtfs . . . 11
get_1stdeparture . . . 12
get_corridor . . . 13
get_distances . . . 13
get_durations . . . 15
get_dwelltimes . . . 16
get_fleet . . . 17
get_frequency . . . 18
get_headways . . . 19
get_hubs . . . 20
get_servicepattern . . . 20
get_shapes . . . 21
get_shapes_df . . . 22
get_shapes_sf . . . 23
get_speeds . . . 23
get_stops_sf . . . 24
latlon2epsg . . . 25
merge_gtfs . . . 25
plot_calendar . . . 26
plot_corridor . . . 27
plot_frequency . . . 27
plot_headways . . . 28
plot_hubs . . . 29
plot_routeduration . . . 29
plot_routefrequency . . . 30
plot_serviceheatmap . . . 31
plot_servicespan . . . 31
plot_servicesupply . . . 32
read_gtfs . . . 33
selection . . . 33
set_dwelltime . . . 35
split_trip . . . 35
tidy_raptor . . . 36

wizardgtfs-methods 38
write_gtfs 39

Index **40**

`as_wizardgtfs` *Convert a GTFS Feed to 'wizardgtfs'*

Description

Converts a list-like GTFS feed to the class used by GTFSwizard. Character GTFS dates are converted to [Date] values, a date-to-service lookup is created, and missing shapes can optionally be inferred from stop locations.

Usage

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

Arguments

`gtfs_list`	A named list of GTFS tables or a 'tidygtfs' object.
`build_shapes`	Logical. If 'TRUE', infer 'shapes' when the table is absent. Inferred shapes connect stops with straight line segments and are intended for analysis and visualization, not map-matched routing.

Details

The input is checked for required tables, required fields, primary-key duplication, foreign-key consistency, increasing stop and shape sequences, valid service dates, and valid GTFS time strings. Times remain character values because GTFS permits hours greater than 24.

Value

A 'wizardgtfs' object.

See Also

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

Examples

```
gtfs_wizard <- as_wizardgtfs(for_rail_gtfs, build_shapes = TRUE)
```

`create_gtfs` *Create a GTFS Feed from Tables*

Description

Creates and validates a GTFS Schedule feed from data frames. The required files and fields follow the current GTFS Schedule reference, including its conditional rules for route names, stop locations, and service calendars.

Usage

```
create_gtfs(
  agency,
  routes,
  trips,
  stop_times,
  stops,
  calendar = NULL,
  calendar_dates = NULL,
  shapes = NULL,
  frequencies = NULL,
  transfers = NULL,
  fare_attributes = NULL,
  fare_rules = NULL,
  build_shapes = TRUE,
  zipfile = NULL,
  ...
)
```

Arguments

`agency`, `routes`, `trips`, `stop_times`, `stops`	Required GTFS tables supplied as data frames. ‘agency_id‘ may be omitted when the feed contains one agency.
`calendar`, `calendar_dates`	Service calendar tables. At least one must define every ‘service_id‘ used by ‘trips‘.
`shapes`, `frequencies`, `transfers`, `fare_attributes`, `fare_rules`	Optional GTFS tables.
`build_shapes`	Logical. If ‘TRUE‘, infer straight-line shapes when ‘shapes‘ is not supplied.
`zipfile`	Optional output path ending in ‘.zip‘.
`...`	Additional named GTFS tables, such as ‘pathways‘, ‘levels‘, ‘feed_info‘, or GTFS Fares v2 tables.

Value

A validated ‘wizardgtfs‘ object, invisibly written as well when ‘zipfile‘ is supplied.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/)

Examples

```
feed <- create_gtfs(
  agency = data.frame(
    agency_id = "A", agency_name = "Demo Transit",
    agency_url = "https://example.com",
    agency_timezone = "America/Fortaleza"
  ),
  routes = data.frame(
    route_id = "R1", agency_id = "A", route_short_name = "1",
    route_long_name = "Central", route_type = 3
  ),
  trips = data.frame(route_id = "R1", service_id = "WK", trip_id = "T1"),
  stop_times = data.frame(
    trip_id = "T1", arrival_time = c("08:00:00", "08:10:00"),
    departure_time = c("08:00:00", "08:10:00"),
    stop_id = c("S1", "S2"), stop_sequence = c(1, 2)
  ),
  stops = data.frame(
    stop_id = c("S1", "S2"), stop_name = c("First", "Second"),
    stop_lat = c(-3.73, -3.74), stop_lon = c(-38.52, -38.53)
  ),
  calendar = data.frame(
    service_id = "WK", monday = 1, tuesday = 1, wednesday = 1,
    thursday = 1, friday = 1, saturday = 0, sunday = 0,
    start_date = "20260101", end_date = "20261231"
  )
)
```

---

delay_trip *Shift Trips in Time*

Description

Adds a number of seconds to every non-empty arrival and departure time for selected trips. GTFS times above 24 hours are preserved.

Usage

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

Arguments

gtfs	A GTFS object.
trip	Character vector of ‘trip_id‘ values.
duration	Numeric seconds or an object coercible to seconds, such as a ‘difftime‘.

Value

A modified ‘wizardgtfs‘ object.

See Also

[GTFSwizard::edit_speed()], [GTFSwizard::set_dwelltime()]

Examples

```
delayed <- delay_trip(
  for_rail_gtfs,
  trip = for_rail_gtfs$trips$trip_id[1:2],
  duration = 300
)
```

`edit_dwelltime` *Scale Dwell Times*

Description

Multiplies dwell time at selected trip-stop calls and propagates each change to all later times in the same trip. Arrival at the edited stop is retained; departure and subsequent calls move by the dwell-time difference.

Usage

```
edit_dwelltime(gtfs, trips = "all", stops = "all", factor)
```

Arguments

`gtfs`	A GTFS object.
`trips`, `stops`	Character ID vectors or ‘"all"‘.
`factor`	One non-negative numeric multiplier.

Value

A modified ‘wizardgtfs‘ object.

See Also

[GTFSwizard::set_dwelltime()], [GTFSwizard::get_dwelltimes()]

Examples

```
edited <- edit_dwelltime(
  for_rail_gtfs,
  trips = for_rail_gtfs$trips$trip_id[1:2],
  stops = for_rail_gtfs$stops$stop_id[1:2],
  factor = 1.5
)
```

`edit_speed` *Scale In-Vehicle Travel Speed*

Description

Changes travel time between consecutive stops by dividing it by a speed multiplier. Dwell times are preserved and all downstream times are shifted.

Usage

```
edit_speed(gtfs, trips = "all", stops = "all", factor)
```

Arguments

`gtfs`	A GTFS object.
`trips`	Character trip IDs or ‘"all"‘.
`stops`	Character stop IDs or ‘"all"‘. A segment is edited when either endpoint is selected.
`factor`	One positive speed multiplier. For example, ‘2‘ halves segment travel times.

Value

A modified ‘wizardgtfs‘ object.

See Also

[GTFSwizard::get_speeds()], [GTFSwizard::get_durations()]

Examples

```
edited <- edit_speed(
  for_rail_gtfs,
  trips = for_rail_gtfs$trips$trip_id[1:2],
  stops = "all",
  factor = 1.25
)
```

explore_gtfs *Explore GTFS Data in an Interactive Shiny Dashboard*

Description

Opens a lightweight Shiny dashboard for exploring a GTFS feed. The dashboard shows summary cards, route and stop maps, service calendar, and key operational charts. Route, service, service-pattern, stop, date, and time filters update the dashboard without changing the original object.

Usage

```
explore_gtfs(gtfs = NULL)
```

Arguments

gtfs	A GTFS object, preferably of class ‘wizardgtfs‘. When omitted or ‘NULL‘ in an interactive session, a file-selection window opens so the user can choose a GTFS ‘.zip‘ archive.

Value

A Shiny app object.

See Also

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

Examples

```
if (interactive()) {
  explore_gtfs()
  explore_gtfs(GTFSwizard::for_rail_gtfs)
}
```

filter_servicepattern *Filter a GTFS Feed*

Description

Filters a GTFS feed while preserving referential integrity across routes, trips, stops, shapes, calendars, frequencies, fares, and transfers.

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 = "00:00:00", to = "48:00:00")
```

Arguments

`gtfs`	A GTFS object.
`servicepattern`	Character vector of service-pattern IDs returned by [get_servicepattern()]. When ‘NULL‘, the most frequent pattern is used.
`dates`	Dates to retain. Values accepted by [as.Date()] are supported. When ‘NULL‘, the latest available service date is used.
`service`, `route`, `trip`, `stop`	Character vectors of IDs to retain.
`keep`	Logical. If ‘FALSE‘, the specified route or trip IDs are excluded.
`from`, `to`	Inclusive GTFS time bounds in ‘"HH:MM:SS"‘ form. Hours may exceed 24.

Details

‘filter_stop()‘ retains the requested stop calls and therefore may return partial trips. ‘filter_time()‘ retains individual stop calls whose arrival or departure falls inside the inclusive time interval and may also return partial trips. This behavior is useful for network experiments. Route, trip, service, service-pattern, and date filters retain complete trips.

‘filter_date()‘ rewrites service availability as ‘calendar_dates‘ additions for exactly the selected dates. Other filters preserve the selected services’ original date ranges and exceptions.

Value

A ‘wizardgtfs‘ object.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/)

See Also

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

Examples

```
typical <- filter_servicepattern(for_rail_gtfs)
one_day <- filter_date(for_rail_gtfs, "2021-02-10")
one_route <- filter_route(for_rail_gtfs, for_rail_gtfs$routes$route_id[1])
two_trips <- filter_trip(for_rail_gtfs, for_rail_gtfs$trips$trip_id[1:2])
serving_stop <- filter_stop(for_rail_gtfs, for_rail_gtfs$stops$stop_id[1])
morning <- filter_time(for_rail_gtfs, from = "06:30:00", to = "10:00:00")
```

---

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 345 rows and 5 columns, linking fare IDs to routes, along with optional restrictions on origins, destinations, and zones.

routes Data frame with 345 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 125,776 rows and 5 columns, representing the spatial paths of routes with latitude, longitude, point sequence, and cumulative distance traveled.

stop_times Data frame with 2,659,737 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 4,676 rows and 12 columns, containing information about each stop, including stop ID, name, location (latitude and longitude), and accessibility.

trips Data frame with 85,410 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_1stdeparture *Get the First Departure of Each Trip*

Description

Returns the departure at the lowest ‘stop_sequence‘ for every trip.

Usage

```
get_1stdeparture(gtfs)
```

Arguments

gtfs	A GTFS object.

Value

A tibble with ‘route_id‘, ‘trip_id‘, ‘departure_time‘, and ‘stop_id‘.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/#stop_timestxt)

Examples

```
head(get_1stdeparture(for_rail_gtfs))
```

get_corridor *Identify High-Frequency Transit Corridors*

Description

Finds frequently served adjacent stop pairs, joins connected pairs into corridors, and measures them in a local metric CRS.

Usage

```
get_corridor(gtfs, i = 0.01, min.length = 1500)
```

Arguments

gtfs	A GTFS object.
i	Proportion of the highest-frequency stop pairs to retain. Must be greater than 0 and no greater than 1.
min.length	Minimum corridor length in meters.

Value

An ‘sf‘ object with ‘corridor‘, list-columns ‘stop_id‘ and ‘trip_id‘, numeric ‘length‘ in meters, and WGS84 geometry. Segments are straight lines between stop coordinates, not shape geometry.

See Also

[GTFSwizard::plot_corridor()]

Examples

```
corridors <- get_corridor(for_rail_gtfs, i = 0.2, min.length = 100)
```

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", trips = "all")
```

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.
trips	A character vector of trip IDs to consider. When set to ‘all‘, includes all trips.

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", trips = 'all')

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


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

`get_durations` *Calculate Trip Durations in GTFS Data*

Description

Calculates scheduled trip and segment durations in seconds.

Usage

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

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.
`trips`	A character vector of trip IDs to consider. When set to ‘all‘, includes all trips.

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, excluding dwell times.

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"‘ Includes dwell from first departure to final arrival.

If ‘method = "by.trip"‘ Includes dwell from first departure to final arrival.

If ‘method = "detailed"‘ It does not include dwell times. 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", trips = 'all')

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

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

`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 mean dwell time for each trip.

- "detailed": Calculates departure minus arrival at each stop call.

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 *Estimate Simultaneous Vehicles*

Description

Estimates the number of active trip instances from their first departure to final arrival. Frequency-based services are expanded from ‘frequencies.txt‘.

Usage

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

Arguments

gtfs	A GTFS object.
method	One of ‘"by.route"‘, ‘"by.hour"‘, ‘"peak"‘, or ‘"detailed"‘.

Value

A tibble containing fleet estimates by the selected grouping and service pattern. GTFS times beyond 24 hours remain in the following service day rather than being wrapped.

Examples

```
get_fleet(for_rail_gtfs, "by.route")
get_fleet(for_rail_gtfs, "by.hour")
```

`get_frequency` *Calculate Scheduled Service Frequency*

Description

Counts trip departures by route, shape, stop, or hour. Trips referenced by ‘frequencies.txt‘ are expanded using the period’s inclusive ‘start_time‘, exclusive ‘end_time‘, and ‘headway_secs‘, as required by GTFS.

Usage

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

Arguments

`gtfs`	A GTFS object.
`method`	One of ‘"by.route"‘, ‘"by.shape"‘, ‘"by.stop"‘, or ‘"detailed"‘.

Value

A tibble containing the selected identifiers, frequency, service pattern, and number of dates represented by the pattern.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/#frequenciestxt)

See Also

[GTFSwizard::get_headways()]

Examples

```
get_frequency(for_rail_gtfs, "by.route")
get_frequency(for_rail_gtfs, "detailed")
```

`get_headways` *Calculate Scheduled Headways*

Description

Calculates elapsed minutes between successive service instances. Frequency based trips are expanded according to ‘frequencies.txt‘.

Usage

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

Arguments

`gtfs`	A GTFS object.
`method`	One of ‘"by.route"‘, ‘"by.hour"‘, ‘"by.trip"‘, ‘"by.stop"‘, ‘"by.shape"‘, or ‘"detailed"‘.

Details

Route, hour, and trip methods compare first departures. Stop and detailed methods compare arrivals at the same stop, route, direction, and service pattern. The first service instance in each group has no headway and is omitted.

Value

A tibble. Headway values are always returned in ‘headway_minutes‘; ‘valid_trips‘ is the number of intervals summarized.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/#frequenciestxt)

See Also

[GTFSwizard::get_frequency()]

Examples

```
get_headways(for_rail_gtfs, "by.route")
get_headways(for_rail_gtfs, "by.hour")
```

get_hubs *Identify Stops Serving Multiple Routes*

Description

Identify Stops Serving Multiple Routes

Usage

```
get_hubs(gtfs)
```

Arguments

gtfs	A GTFS object.

Value

A WGS84 ‘sf‘ object with one row per served stop, list-columns ‘trip_id‘ and ‘route_id‘, counts ‘n_trip‘ and ‘n_routes‘, and point geometry.

See Also

[GTFSwizard::plot_hubs()], [GTFSwizard::get_stops_sf()]

Examples

```
get_hubs(for_rail_gtfs)
```

get_servicepattern *Identify Services with the Same Operating Dates*

Description

Groups ‘service_id‘ values that operate on exactly the same set of dates.

Usage

```
get_servicepattern(gtfs)
```

Arguments

gtfs	A GTFS object.

Value

A tibble with one row per ‘service_id‘, its ‘service_pattern‘, and ‘pattern_frequency‘, the number of active dates in that pattern. Pattern numbers are ordered from most to least frequent.

See Also

[GTFSwizard::filter_servicepattern()]

Examples

```
get_servicepattern(for_rail_gtfs)
```

`get_shapes` *Infer Straight-Line Shapes from Stop Sequences*

Description

Builds one shape for each unique ordered stop pattern and assigns it to the corresponding trips.

Usage

```
get_shapes(gtfs)
```

Arguments

`gtfs`	A GTFS object.

Details

Consecutive stop coordinates are connected directly. The result is useful for visualization and approximate distance analysis, but it is not a map-matched representation of the vehicle path.

Value

A ‘wizardgtfs‘ object with a new ‘shapes‘ table and updated ‘trips$shape_id‘.

See Also

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

Examples

```
gtfs_with_shapes <- get_shapes(for_rail_gtfs)
```

`get_shapes_df` *Convert Shape Geometries to a GTFS Shapes Table*

Description

Convert Shape Geometries to a GTFS Shapes Table

Usage

```
get_shapes_df(shape)
```

Arguments

shape	An ‘sf‘ object with a ‘shape_id‘ field and line geometries.

Details

Distances are geodesic approximations between consecutive coordinates. GTFS permits any consistent distance unit; this function uses meters.

Value

A tibble containing ‘shape_id‘, WGS84 point coordinates, ‘shape_pt_sequence‘, and cumulative ‘shape_dist_traveled‘ in meters.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/#shapestxt)

See Also

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

Examples

```
shapes_sf <- get_shapes_sf(for_rail_gtfs)$shapes
shapes_df <- get_shapes_df(shapes_sf)
```

get_shapes_sf *Convert GTFS Shapes to Simple Features*

Description

Convert GTFS Shapes to Simple Features

Usage

```
get_shapes_sf(gtfs)
```

Arguments

gtfs	A GTFS object or a ‘shapes‘ data frame.

Value

For a data frame, one WGS84 ‘LINESTRING‘ feature per ‘shape_id‘. For a GTFS object, the same object with its ‘shapes‘ table replaced by that ‘sf‘ object. If present, ‘shape_dist_traveled‘ is summarized as its maximum value for each shape.

See Also

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

Examples

```
gtfs_sf <- get_shapes_sf(for_rail_gtfs)
shapes_sf <- get_shapes_sf(for_rail_gtfs$shapes)
```

get_speeds *Calculate Scheduled Speeds*

Description

Combines distance in meters and duration in seconds to calculate kilometers per hour.

Usage

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

Arguments

gtfs	A GTFS object.
method	One of ‘"by.route"‘, ‘"by.trip"‘, or ‘"detailed"‘.
trips	Character trip IDs or ‘"all"‘.

Details

Detailed distances are straight-line geodesic distances between stops. Route and trip distances follow ‘shapes.txt‘, or inferred straight-line shapes if the feed has none.

Value

A tibble with ‘average.speed‘ for route and trip methods, or segment-level ‘speed‘ for the detailed method.

See Also

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

Examples

```
get_speeds(for_rail_gtfs, "by.route")
get_speeds(for_rail_gtfs, "by.trip")
```

---

get_stops_sf *Convert GTFS Stops to Simple Features*

Description

Convert GTFS Stops to Simple Features

Usage

```
get_stops_sf(gtfs)
```

Arguments

gtfs	A GTFS object or a ‘stops‘ data frame.

Value

For a data frame, a WGS84 point ‘sf‘ object. For a GTFS object, the same object with its ‘stops‘ table converted to ‘sf‘.

Examples

```
gtfs_sf <- get_stops_sf(for_rail_gtfs)
stops_sf <- get_stops_sf(for_rail_gtfs$stops)
```

latlon2epsg *Transform Spatial Data to a Local Metric CRS*

Description

Selects a UTM zone from the geographic centroid, or a polar stereographic CRS outside UTM's latitude range.

Usage

```
latlon2epsg(sf_obj)
```

Arguments

sf_obj	An 'sf' or 'sfc' object with a defined CRS.

Value

The input transformed to EPSG 326xx/327xx, EPSG 3413, or EPSG 3031.

See Also

[sf::st_transform()]

Examples

```
shapes <- get_shapes_sf(for_rail_gtfs)$shapes
metric_shapes <- latlon2epsg(shapes)
```

merge_gtfs *Merge Two GTFS Feeds*

Description

Combines two feeds table by table. By default, all identifiers and their foreign-key references receive feed-specific suffixes, preventing accidental collisions.

Usage

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

Arguments

gtfs.x, gtfs.y	GTFS objects.
suffix	Logical. Append '.x' and '.y' to identifiers when 'TRUE'.

Details

Standard references in routes, trips, stop times, shapes, calendars, frequencies, transfers, pathways, fare tables, networks, and booking rules are updated together. Non-standard tables are row-bound without identifier rewriting.

Value

A validated ‘wizardgtfs‘ object.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/)

Examples

```
merged <- merge_gtfs(for_rail_gtfs, for_rail_gtfs, suffix = TRUE)
```

---

plot_calendar *Plot the GTFS Service Calendar*

Description

Creates a calendar heatmap of scheduled trips by service date.

Usage

```
plot_calendar(gtfs, ncol = 4, facet_by_year = FALSE)
```

Arguments

gtfs	A GTFS object.
ncol	Number of facet columns when ‘facet_by_year = FALSE‘.
facet_by_year	Logical. Arrange years in rows and months in columns.

Value

A ‘ggplot‘ object.

See Also

[GTFSwizard::get_servicepattern()]

Examples

```
plot_calendar(for_rail_gtfs, ncol = 4)
```

plot_corridor *Plot High-Frequency Corridors*

Description

Plot High-Frequency Corridors

Usage

```
plot_corridor(gtfs, i = 0.01, min.length = 1500)
```

Arguments

gtfs	A GTFS object.
i	Proportion of highest-frequency segments to retain.
min.length	Minimum corridor length in meters.

Value

A ‘ggplot‘ map.

See Also

[GTFSwizard::get_corridor()]

plot_frequency *Plot System Frequency by Hour*

Description

Shows the distribution and weighted mean of scheduled trip departures by hour. Service-pattern date counts provide the weights.

Usage

```
plot_frequency(gtfs)
```

Arguments

gtfs	A GTFS object.

Value

A ‘ggplot‘ object.

See Also

[GTFSwizard::get_frequency()]

Examples

```
plot_frequency(for_rail_gtfs)
```

`plot_headways` *Plot System Headway by Hour*

Description

Plot System Headway by Hour

Usage

```
plot_headways(gtfs)
```

Arguments

`gtfs`	A GTFS object.

Value

A ‘ggplot‘ object with headway in minutes.

See Also

[GTFSwizard::get_headways()]

Examples

```
plot_headways(for_rail_gtfs)
```

plot_hubs *Plot Transit Hubs*

Description

Plot Transit Hubs

Usage

```
plot_hubs(gtfs, i = 0.05)
```

Arguments

gtfs	A GTFS object.
i	Proportion of stops with the most routes to retain.

Details

To keep dense networks readable, at most 40 of the highest-ranked stops are drawn. Ranking uses distinct route count and then trip count.

Value

A ‘ggplot‘ map.

See Also

[GTFSwizard::get_hubs()]

plot_routeduration *Plot Scheduled Trip Duration by Route*

Description

Shows the distribution of scheduled end-to-end trip duration for the busiest routes.

Usage

```
plot_routeduration(gtfs, top_n = 20L)
```

Arguments

gtfs	A GTFS object.
top_n	Maximum number of routes to display.

Value

A ‘ggplot‘ object. Each observation is one scheduled trip; boxes summarize duration in minutes by route.

References

[FTA Evaluation Introduction](https://www.transit.dot.gov/research-innovation/evaluation-introduction)

Examples

```
plot_routeduration(for_rail_gtfs)
```

---

plot_routefrequency *Plot Route Frequency by Hour*

Description

Creates a heatmap of scheduled departures by route, hour, and service pattern. The tile fill is the number of scheduled trips.

Usage

```
plot_routefrequency(gtfs, route = NULL, top_n = 25L)
```

Arguments

gtfs	A GTFS object.
route	Optional character vector of route IDs. ‘NULL‘ includes all routes after applying ‘top_n‘.
top_n	Maximum number of routes to display when ‘route‘ is ‘NULL‘.

Value

A ‘ggplot‘ object.

See Also

[GTFSwizard::get_frequency()]

Examples

```
plot_routefrequency(
  for_rail_gtfs,
  route = for_rail_gtfs$routes$route_id[1:2]
)
```

plot_serviceheatmap *Plot Average Scheduled Departures by Weekday and Hour*

Description

Creates a heatmap of average scheduled departures on active service dates.

Usage

```
plot_serviceheatmap(gtfs)
```

Arguments

gtfs	A GTFS object.

Value

A 'ggplot' object. Each tile is a weekday-hour combination and its fill is the mean number of scheduled trip departures per active date.

References

[GTFS Schedule Reference](https://gtfs.org/documentation/schedule/reference/)

Examples

```
plot_serviceheatmap(for_rail_gtfs)
```

plot_servicespan *Plot Scheduled Service Span by Route*

Description

Shows the first departure and final arrival for route and service-pattern combinations. The longest and most frequently operated route-pattern combinations are retained when the feed is large.

Usage

```
plot_servicespan(gtfs, top_n = 30L)
```

Arguments

gtfs	A GTFS object.
top_n	Maximum number of route-pattern combinations to display.

Value

A ‘ggplot‘ object. Each horizontal line is one route and service pattern; its endpoints are scheduled clock hours.

References

[GTFS Routes, Stops, and Trips](https://gtfs.org/documentation/schedule/examples/routes-stops-trips/)

Examples

```
plot_servicespan(for_rail_gtfs)
```

`plot_servicesupply` *Plot Scheduled Vehicle-Hours by Route*

Description

Sums scheduled end-to-end trip durations for each route and service pattern.

Usage

```
plot_servicesupply(gtfs, top_n = 20L)
```

Arguments

`gtfs`	A GTFS object.
`top_n`	Maximum number of routes to display.

Value

A ‘ggplot‘ object. Each bar is scheduled vehicle-hours for one route on one representative service-pattern day.

References

[FTA National Transit Database](https://www.transit.dot.gov/ntd)

Examples

```
plot_servicesupply(for_rail_gtfs)
```

read_gtfs *Read a GTFS Feed*

Description

Imports a GTFS zip archive and converts it to ‘wizardgtfs‘.

Usage

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

Arguments

file.path	Path to a GTFS ‘.zip‘ archive.
files	Optional character vector of table names without ‘.txt‘.
quiet	Logical. Suppress importer messages.
...	Additional arguments passed to [gtfsio::import_gtfs()].

Value

A validated ‘wizardgtfs‘ object.

See Also

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

Examples

```
path <- tempfile(fileext = ".zip")
write_gtfs(for_rail_gtfs, path)
gtfs <- read_gtfs(path)
```

selection *Group or Select GTFS Records Without Filtering the Feed*

Description

Creates persistent selection metadata while leaving every GTFS table unchanged. Bare columns and computed values define groups, similarly to [dplyr::group_by()]. Logical expressions restrict the records represented by those groups.

Usage

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

unselection(gtfs)
```

Arguments

gtfs	A GTFS object.
...	Unquoted grouping columns, computed grouping expressions, or logical selection expressions. Columns from ‘stop_times‘, ‘trips‘, ‘routes‘, and ‘stops‘ are available. A ‘geometry‘ column is available for stop-based spatial predicates.
add	Logical. When ‘TRUE‘, add the new grouping or selection expressions to an existing selection. When ‘FALSE‘, replace it.

Details

A bare expression such as ‘route_id‘ creates one group per route. Multiple grouping expressions create combinations, for example ‘selection(gtfs, route_id, direction_id)‘. Logical expressions such as ‘route_id selection but do not remove rows from the GTFS feed.

Spatial expressions may use ‘geometry‘ with ‘ ‘

Value

The unchanged feed with class ‘wizardgtfs_selected‘ and a ‘selection‘ attribute. The attribute contains ‘groups‘, ‘group_vars‘, selected route, trip, and stop IDs, and the original expressions.

See Also

[GTFSwizard::unselection()], [dplyr::group_by()]

Examples

```
grouped <- selection(for_rail_gtfs, route_id, direction_id)
attr(grouped, "selection")$groups

selected <- selection(
  for_rail_gtfs,
  route_id,
  stop_id %in% for_rail_gtfs$stops$stop_id[1:3]
)

bbox <- sf::st_bbox(
  c(xmin = -38.58, ymin = -3.81, xmax = -38.50, ymax = -3.75),
  crs = sf::st_crs(4326)
)
spatial <- selection(
  for_rail_gtfs,
  geometry %intersects% sf::st_as_sfc(bbox)
)
```

set_dwelltime *Set Dwell Times*

Description

Sets dwell time at selected trip-stop calls and propagates each change to all later times in the same trip. Arrival at the edited stop is retained.

Usage

```
set_dwelltime(gtfs, duration = 30, trips = "all", stops = "all")
```

Arguments

gtfs	A GTFS object.
duration	One non-negative dwell time in seconds.
trips, stops	Character ID vectors or '"all"'.

Value

A modified 'wizardgtfs' object.

See Also

[GTFSwizard::edit_dwelltime()], [GTFSwizard::get_dwelltimes()]

Examples

```
edited <- set_dwelltime(
  for_rail_gtfs,
  duration = 30,
  trips = for_rail_gtfs$trips$trip_id[1:2],
  stops = for_rail_gtfs$stops$stop_id[1:2]
)
```

split_trip *Split Trips into Consecutive Parts*

Description

Splits each selected trip into approximately equal consecutive parts. The boundary stop is included at the end of one part and the start of the next, producing valid complete stop sequences.

Usage

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

Arguments

gtfs	A GTFS object.
trip	Character vector of ‘trip_id‘ values.
split	Positive integer number of split points. ‘split = 1‘ creates two parts.

Details

New IDs use ‘.part1‘, ‘.part2‘, and so on. New straight-line shapes are inferred from stop coordinates for the split parts. Frequency periods are shifted by each part's offset from the original first departure. Trip-level transfers are reassigned to the part containing their transfer stop.

Value

A modified ‘wizardgtfs‘ object.

See Also

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

Examples

```
gtfs_split <- split_trip(
  for_rail_gtfs,
  trip = for_rail_gtfs$trips$trip_id[1],
  split = 2
)
```

---

tidy_raptor *Calculate Travel Times with RAPTOR Algorithm*

Description

The ‘tidy_raptor‘ function calculates travel times from a set of origin stops to all reachable stops within a GTFS dataset. It uses the RAPTOR (Round-Based Public Transit Routing) algorithm from the ‘tidytransit‘ package and integrates it with the GTFSwizard framework.

Usage

```
tidy_raptor(
  gtfs,
  min_departure = "0:0:0",
  max_arrival = "23:59:59",
  dates = NULL,
  stop_ids,
  arrival = FALSE,
  time_range = 3600,
```

```
  max_transfers = NULL,
  keep = "all",
  filter = TRUE
)
```

Arguments

gtfs	A GTFS object, preferably of class 'wizardgtfs'. If not, the function will attempt to convert it using 'GTFSwizard::as_wizardgtfs()'.
min_departure	A string representing the earliest departure time, in "HH:MM:SS" format. Defaults to '"0:0:0"'.
max_arrival	A string representing the latest arrival time, in "HH:MM:SS" format. Defaults to '"23:59:59"'.
dates	A date (in '"YYYY-MM-DD"' format) to filter the GTFS dataset to specific calendar days. Defaults to 'NULL', meaning the furthest date.
stop_ids	A character vector of stop IDs from where journeys should start (or end, if 'arrival = TRUE').
arrival	Logical. If 'FALSE' (default), journeys start from 'stop_ids'. If 'TRUE', journeys end at 'stop_ids'.
time_range	Either a range in seconds (numeric) or a vector with the minimal and maximal departure time (e.g., 'c(0, 3600)' or '"HH:MM:SS"') describing the journey window.
max_transfers	Maximum number of transfers allowed. Defaults to 'NULL' (no limit).
keep	One of '"all"', '"shortest"', '"earliest"', or '"latest"'. Determines which journeys to retain: - '"all"': All journeys are returned (default). - '"shortest"': Only journeys with the shortest travel time. - '"earliest"': Journeys arriving at stops the earliest. - '"latest"': Journeys arriving at stops the latest.
filter	A logical to filter for min_departure, max_arrivel, and dates. Defaults to 'TRUE'.

Value

A tibble containing the RAPTOR algorithm results, including:

from_stop_id The ID of the stop where the journey starts.

to_stop_id The ID of the stop where the journey ends.

departure_time Departure time from the origin stop.

arrival_time Arrival time at the destination stop.

travel_time Total travel time in seconds.

Note

Ensure that the 'stop_times' is present and correctly structured in the GTFS dataset. Time values in 'min_departure', 'max_arrival', and 'time_range' should be correctly formatted to avoid errors. 'max_arrival' must be 23:59:59 or earlier.

See Also

[tidytransit::raptor()], [GTFSwizard::as_wizardgtfs()], [GTFSwizard::filter_time()]

Examples

```
tidy_raptor(for_rail_gtfs,
   min_departure = '06:20:00',
   max_arrival = '09:40:00',
   dates = "2021-12-13",
   max_transfers = 2,
   keep = "all",
   stop_ids = '66')
```

---

wizardgtfs-methods *Print, Summarize, and Plot 'wizardgtfs' Objects*

Description

Print, Summarize, and Plot 'wizardgtfs' Objects

Usage

```
## S3 method for class 'wizardgtfs'
print(x, ..., n = 5L)

## S3 method for class 'wizardgtfs'
summary(object, ...)

## S3 method for class 'summary.wizardgtfs'
print(x, ...)

## S3 method for class 'wizardgtfs'
plot(x, ...)
```

Arguments

x, object	A 'wizardgtfs' object or its summary.
...	Additional arguments passed to print methods.
n	Number of rows shown per GTFS table.

Value

'print()' returns 'x' invisibly; 'summary()' returns a 'summary.wizardgtfs' object; 'plot()' returns a 'ggplot' object.

`write_gtfs` *Write a GTFS Feed*

Description

Exports a GTFS object to a zip archive. Internal ‘dates_services‘ data are omitted, GTFS dates use ‘YYYYMMDD‘, and spatial stops/shapes are converted back to standard text-table columns.

Usage

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

Arguments

`gtfs`	A GTFS object.
`zipfile`	Output ‘.zip‘ path.
`...`	Additional arguments passed to the format-specific writer.

Value

The normalized output path, invisibly.

See Also

[GTFSwizard::read_gtfs()]

Examples

```
path <- tempfile(fileext = ".zip")
write_gtfs(for_rail_gtfs, path)
```

Index

as_wizardgtfs, 3

create_gtfs, 4

delay_trip, 5

edit_dwelltime, 6
edit_speed, 7
explore_gtfs, 8

filter_date *(*filter_servicepattern*)*, 8
filter_route *(*filter_servicepattern*)*, 8
filter_service *(*filter_servicepattern*)*, 8
filter_servicepattern, 8
filter_stop *(*filter_servicepattern*)*, 8
filter_time *(*filter_servicepattern*)*, 8
filter_trip *(*filter_servicepattern*)*, 8
for_bus_gtfs, 10
for_rail_gtfs, 11

get_1stdeparture, 12
get_corridor, 13
get_distances, 13
get_durations, 15
get_dwelltimes, 16
get_fleet, 17
get_frequency, 18
get_headways, 19
get_hubs, 20
get_servicepattern, 20
get_shapes, 21
get_shapes_df, 22
get_shapes_sf, 23
get_speeds, 23
get_stops_sf, 24

latlon2epsg, 25

merge_gtfs, 25

plot.wizardgtfs *(*wizardgtfs-methods*)*, 38
plot_calendar, 26
plot_corridor, 27
plot_frequency, 27
plot_headways, 28
plot_hubs, 29
plot_routeduration, 29
plot_routefrequency, 30
plot_serviceheatmap, 31
plot_servicespan, 31
plot_servicesupply, 32
print.summary.wizardgtfs *(*wizardgtfs-methods*)*, 38
print.wizardgtfs *(*wizardgtfs-methods*)*, 38

read_gtfs, 33

selection, 33
set_dwelltime, 35
split_trip, 35
summary.wizardgtfs *(*wizardgtfs-methods*)*, 38

tidy_raptor, 36

unselection *(*selection*)*, 33

wizardgtfs-methods, 38
write_gtfs, 39