Package: gtfsrouter (via r-universe)

October 30, 2024

Title Routing with 'GTFS' (General Transit Feed Specification) Data

Version 0.1.3

Description Use 'GTFS' (General Transit Feed Specification) data for
routing from nominated start and end stations, for extracting
'isochrones', and travel times from any nominated start station
to all other stations.

License GPL-3

URL `https://github.com/UrbanAnalyst/gtfsrouter`

BugReports `https://github.com/UrbanAnalyst/gtfsrouter/issues`

Depends R (>= 2.10)

Imports cli, data.table, fs, geodist, methods, Rcpp (>= 0.12.6)

Suggests digest, dodgr, dplyr, ggplot2, here, hms, knitr, lubridate,
lwgeom, markdown, pbapply, rmarkdown, testthat

LinkingTo Rcpp

VignetteBuilder knitr

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 7.3.2

Author Mark Padgham [aut, cre], Marcin Stepniak [aut]
(<`https://orcid.org/0000-0002-6489-5443`>), Alexandra Kapp [ctb]

Maintainer Mark Padgham <`mark.padgham@email.com`>

Repository CRAN

Date/Publication 2024-10-29 08:40:12 UTC

Contents

`berlin_gtfs` *berlin_gtfs*

Description

Sample GTFS data from Verkehrsverbund Berlin-Brandenburg street, reduced to U and S Bahn only (underground and overground trains), and between the hours of 12:00-13:00. Only those components of the GTFS data necessary for routing have been retained. Note that non-ASCII characters have been removed from these data, so umlauts are simply removed and eszetts become "ss". The package will nevertheless work with full GTFS feeds and non-ASCII (UTF-8) characters.

Format

A list of five **data.table** items necessary for routing:

- calendar
- routes
- trips
- stop_times
- stops
- transfers

Value

For single (from, to) values, a `data.frame` describing the route, with each row representing one stop. For multiple (from, to) values, a list of `data.frames`, each of which describes one route between the i'th start and end stations (`from` and `to` values). Origin and destination stations for which no route is possible return `NULL`.

Note

Can be re-created with the script in `https://github.com/UrbanAnalyst/gtfsrouter/blob/master/data-raw/data-script.Rmd`.

Examples

```
# Examples must be run on single thread only:
nthr_dt <- data.table::setDTthreads (1)
nthr_omp <- Sys.getenv ("OMP_THREAD_LIMIT")
Sys.setenv ("OMP_THREAD_LIMIT" = 1L)

berlin_gtfs_to_zip () # Write sample feed from Berlin, Germany to tempdir
f <- file.path (tempdir (), "vbb.zip") # name of feed
gtfs <- extract_gtfs (f)
from <- "Innsbrucker Platz" # U-bahn station, not "S"
to <- "Alexanderplatz"
start_time <- 12 * 3600 + 120 # 12:02

route <- gtfs_route (gtfs, from = from, to = to, start_time = start_time)

# Specify day of week
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday"
)

# specify travel by "U" = underground only
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday",
    route_pattern = "^U"
)
# specify travel by "S" = street-level only (not underground)
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday",
    route_pattern = "^S"
)

# Route queries are generally faster if the GTFS data are pre-processed with
# `gtfs_timetable()`:
gt <- gtfs_timetable (gtfs, day = "Sunday", route_pattern = "^S")
route <- gtfs_route (gt, from = from, to = to, start_time = start_time)

data.table::setDTthreads (nthr_dt)
Sys.setenv ("OMP_THREAD_LIMIT" = nthr_omp)
```

berlin_gtfs_to_zip *berlin_gtfs_to_zip*

Description

Write a zip archive of the internal package data, berlin_gtfs to a file named "vbb.zip" to `tempdir()`.

Usage

```
berlin_gtfs_to_zip()
```

Value

Path to newly created zip file

See Also

Other extract: `extract_gtfs()`, `gtfs_timetable()`

Examples

```
path <- berlin_gtfs_to_zip ()
gtfs <- extract_gtfs (path)
gtfs <- gtfs_timetable (gtfs, day = "Wed") # A pre-processing step to speed up queries
gtfs_route (gtfs, from = "Tegel", to = "Berlin Hauptbahnhof", start_time = 12 * 3600)
```

extract_gtfs *extract_gtfs*

Description

Extract data from a GTFS `zip` archive.

Usage

```
extract_gtfs(filename = NULL, quiet = FALSE, stn_suffixes = NULL)
```

Arguments

`filename`	Name of GTFS archive
`quiet`	If `FALSE`, display progress information on screen
`stn_suffixes`	Any values provided will be removed from terminal characters of station IDs. Useful for feeds like NYC for which some stations are appended with values of "N" and "S" to indicate directions. Specifying `stn_suffixes = c ("N", "S")` will automatically remove these suffixes.

Value

List of several **data.table** objects corresponding to the tables present in the nominated GTFS data set.

Note

Column types in each table of the returned object conform to GTFS standards (`https://developers.google.com/transit/gtfs/reference`), except that "Time" fields in the "stop_times" table are converted to integer values, rather than as character or "Time" objects ("HH:MM:SS"). These can be converted back to comply with GTFS standards by applying the `hms::hms()` function to the two time columns of the "stop_times" table.

See Also

Other extract: `berlin_gtfs_to_zip()`, `gtfs_timetable()`

Examples

```
berlin_gtfs_to_zip () # Write sample feed from Berlin, Germany to tempdir
f <- file.path (tempdir (), "vbb.zip") # name of feed
gtfs <- extract_gtfs (f)
```

`frequencies_to_stop_times` *frequencies_to_stop_times*

Description

Convert a GTFS 'frequencies' table to equivalent 'stop_times' that can be used for routing.

Usage

```
frequencies_to_stop_times(gtfs)
```

Arguments

`gtfs`	A set of GTFS data returned from extract_gtfs.

Value

The input GTFS data with data from the 'frequencies' table converted to equivalent 'arrival_time' and 'departure_time' values in `stop_times`.

See Also

Other augment: `gtfs_transfer_table()`

Examples

```
## Not run:
# Presume an input feed has been created and includes a "frequencies" table:
gtfs2 <- frequencies_to_stop_times (gtfs)
# "gtfs2" will then have an expanded "stop_times" table, with all
# "frequencies" entries converted to equivalent absolute stop times.

## End(Not run)
```

---

go_home *go_home*

Description

Use local environmental variables specifying home and work stations and locations of locally-stored GTFS data to route from work to home location with next available service.

Usage

```
go_home(wait = 0, start_time)
```

Arguments

`wait`	An integer specifying the n-th next service. That is, `wait` = n will return the n-th available service after the next immediate service.
`start_time`	If given, search for connections after specified time; if not given, search for connections from current time.

Details

This function, and the complementary function go_to_work, requires three local environmental variables specifying the names of home and work stations, and the location on local storage of the GTFS data set to be used for routing. These are respectively called `gtfs_home`, `gtfs_work`, and `gtfs_data`. This data set must also be pre-processed using the process_gtfs_local function.

See Startup for details on how to set environmental variables. Briefly, this can be done in two main ways: By setting them at the start of each session, in which case the variables may be set with: `Sys.setenv ("gtfs_home" = "<my home station>") Sys.setenv ("gtfs_work" = "<my work station>") Sys.setenv ("gtfs_data" = "/full/path/to/gtfs.zip")` Alternatively, to set these automatically for each session, paste those lines into the file `~/.Renviron` - that is, a file named ".Renviron" in the user's home directory.

The process_gtfs_local function reduces the GTFS data set to the minimal possible size necessary for local routing. GTFS data are nevertheless typically quite large, and both the go_home and go_to_work functions may take some time to execute. Most of this time is devoted to loading the data in to the current workspace and as such is largely unavoidable.

Value

A `data.frame` specifying the next available route from work to home.

See Also

Other additional: `go_to_work()`, `process_gtfs_local()`, `summary.gtfs()`

Examples

```
## Not run:
# For general use, please set these three variables:
Sys.setenv ("gtfs_home" = "<my home station>")
Sys.setenv ("gtfs_work" = "<my work station>")
Sys.setenv ("gtfs_data" = "/full/path/to/gtfs.zip")

# The following illustrate use with sample data bundled with package
Sys.setenv ("gtfs_home" = "Tempelhof")
Sys.setenv ("gtfs_work" = "Alexanderplatz")
Sys.setenv ("gtfs_data" = file.path (tempdir (), "vbb.zip"))
process_gtfs_local () # If not already done
go_home (start_time = "12:00") # next available service after 12:00
go_home (3, start_time = "12:00") # Wait until third service after that

# Generally, `start_time` will not be specified, in which case `go_home` will
# return next available service from current system time, so calls will
# generally be as simple as:
go_home ()
go_home (3)

## End(Not run)
```

`go_to_work` *go_to_work*

Description

Use local environmental variables specifying home and work stations and locations of locally-stored GTFS data to route from home to work location with next available service.

Usage

```
go_to_work(wait = 0, start_time)
```

Arguments

`wait`	An integer specifying the n-th next service. That is, `wait` = n will return the n-th available service after the next immediate service.
`start_time`	If given, search for connections after specified time; if not given, search for connections from current time.

Details

This function, and the complementary function go_to_work, requires three local environmental variables specifying the names of home and work stations, and the location on local storage of the GTFS data set to be used for routing. These are respectively called `gtfs_home`, `gtfs_work`, and `gtfs_data`. This data set must also be pre-processed using the process_gtfs_local function.

See Startup for details on how to set environmental variables. Briefly, this can be done in two main ways: By setting them at the start of each session, in which case the variables may be set with: `Sys.setenv ("gtfs_home" = "<my home station>") Sys.setenv ("gtfs_work" = "<my work station>") Sys.setenv ("gtfs_data" = "/full/path/to/gtfs.zip")` Alternatively, to set these automatically for each session, paste those lines into the file `~/.Renviron` - that is, a file named ".Renviron" in the user's home directory.

The process_gtfs_local function reduces the GTFS data set to the minimal possible size necessary for local routing. GTFS data are nevertheless typically quite large, and both the go_home and go_to_work functions may take some time to execute. Most of this time is devoted to loading the data in to the current workspace and as such is largely unavoidable.

Value

A `data.frame` specifying the next available route from work to home.

See Also

Other additional: `go_home()`, `process_gtfs_local()`, `summary.gtfs()`

Examples

```
## Not run:
# For general use, please set these three variables:
Sys.setenv ("gtfs_home" = "<my home station>")
Sys.setenv ("gtfs_work" = "<my work station>")
Sys.setenv ("gtfs_data" = "/full/path/to/gtfs.zip")

# The following illustrate use with sample data bundled with package
Sys.setenv ("gtfs_home" = "Tempelhof")
Sys.setenv ("gtfs_work" = "Alexanderplatz")
Sys.setenv ("gtfs_data" = file.path (tempdir (), "vbb.zip"))
process_gtfs_local () # If not already done
go_to_work (start_time = "12:00") # next available service after 12:00
go_to_work (3, start_time = "12:00") # Wait until third service after that

# Generally, `start_time` will not be specified, in which case `go_to_work`
# will return next available service from current system time, so calls will
# generally be as simple as:
go_to_work ()
go_to_work (3)

## End(Not run)
```

gtfsrouter *gtfsrouter*

Description

Routing engine for GTFS (General Transit Feed Specification) data, including one-to-one and one-to-many routing routines.

Main Functions

- `gtfs_route()`: Route between given start and end stations using a nominated GTFS data set.
- `go_home()`: Automatic routing between work and home stations specified with local environmental variables
- `go_to_work()`: Automatic routing between work and home stations specified with local environmental variables
- `gtfs_traveltimes()`: One-to-many routing from a nominated start station to all stations reachable within a specified travel duration.

Author(s)

Maintainer: Mark Padgham <`mark.padgham@email.com`>

Authors:

- Marcin Stepniak <`marcinstepniak@ucm.es`> (ORCID)

Other contributors:

- Alexandra Kapp <`alk@mobilityinstitute.com`> [contributor]

See Also

Useful links:

- `https://github.com/UrbanAnalyst/gtfsrouter`
- Report bugs at `https://github.com/UrbanAnalyst/gtfsrouter/issues`

gtfs_route *gtfs_route*

Description

Calculate single route between a start and end station departing at or after a specified time.

Usage

```
gtfs_route(
  gtfs,
  from,
  to,
  start_time = NULL,
  day = NULL,
  route_pattern = NULL,
  earliest_arrival = TRUE,
  include_ids = FALSE,
  grep_fixed = TRUE,
  max_transfers = NA,
  from_to_are_ids = FALSE,
  quiet = FALSE
)
```

Arguments

`gtfs`	A set of GTFS data returned from extract_gtfs or, for more efficient queries, pre-processed with gtfs_timetable.
`from`	Names, IDs, or approximate (lon, lat) coordinates of start stations (as `stop_name` or `stop_id` entry in the `stops` table, or a vector of two numeric values). See Note.
`to`	Corresponding Names, IDs, or coordinates of end station.
`start_time`	Desired departure time at `from` station, either in seconds after midnight, a vector of two or three integers (hours, minutes) or (hours, minutes, seconds), an object of class difftime, **hms**, or **lubridate**. If not provided, current time is used.
`day`	Day of the week on which to calculate route, either as an unambiguous string (so "tu" and "th" for Tuesday and Thursday), or a number between 1 = Sunday and 7 = Saturday. If not given, the current day will be used. (Not used if `gtfs` has already been prepared with gtfs_timetable.)
`route_pattern`	Using only those routes matching given pattern, for example, "^U" for routes starting with "U" (as commonly used for underground or subway routes. To negate the `route_pattern` – that is, to include all routes except those matching the pattern – prepend the value with "!"; for example "!^U" will include all services except those starting with "U". (This parameter is not used at all if `gtfs` has already been prepared with gtfs_timetable.)

`earliest_arrival`	If `FALSE`, routing will be with the first-departing service, which may not provide the earliest arrival at the `to` station. This may nevertheless be useful for bulk queries, as earliest arrival searches require two routing queries, while earliest departure searches require just one, and so will be generally twice as fast.
`include_ids`	If `TRUE`, result will include columns containing GTFS-specific identifiers for routes, trips, and stops.
`grep_fixed`	If `FALSE`, match station names (when passed as character string) with `grep(..., fixed = FALSE)`, to allow use of `grep` expressions. This is useful to refine matches in cases where desired stations may match multiple entries.
`max_transfers`	If not `NA`, specify a desired maximum number of transfers for the route (including but not exceeding this number). This parameter may be used to generate alternative routes with fewer transfers, although actual numbers of transfers may still exceed this number if the value specified is less than the minimal feasible number of transfers.
`from_to_are_ids`	Set to `TRUE` to enable `from` and `to` parameter to specify entries in `stop_id` rather than `stop_name` column of the `stops` table.
`quiet`	Set to `TRUE` to suppress screen messages (currently just regarding timetable construction).

Value

For single (from, to) values, a `data.frame` describing the route, with each row representing one stop. For multiple (from, to) values, a list of `data.frames`, each of which describes one route between the i'th start and end stations (`from` and `to` values). Origin and destination stations for which no route is possible return `NULL`.

Note

This function will by default calculate the route that arrives earliest at the specified destination, although this may depart later than the earliest departing service. Routes which depart at the earliest possible time can be calculated by setting `earliest_arrival = FALSE`.

See Also

Other main: `gtfs_route_headway()`, `gtfs_traveltimes()`

Examples

```
# Examples must be run on single thread only:
nthr_dt <- data.table::setDTthreads (1)
nthr_omp <- Sys.getenv ("OMP_THREAD_LIMIT")
Sys.setenv ("OMP_THREAD_LIMIT" = 1L)

berlin_gtfs_to_zip () # Write sample feed from Berlin, Germany to tempdir
f <- file.path (tempdir (), "vbb.zip") # name of feed
gtfs <- extract_gtfs (f)
from <- "Innsbrucker Platz" # U-bahn station, not "S"
```

```
to <- "Alexanderplatz"
start_time <- 12 * 3600 + 120 # 12:02

route <- gtfs_route (gtfs, from = from, to = to, start_time = start_time)

# Specify day of week
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday"
)

# specify travel by "U" = underground only
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday",
    route_pattern = "^U"
)
# specify travel by "S" = street-level only (not underground)
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday",
    route_pattern = "^S"
)

# Route queries are generally faster if the GTFS data are pre-processed with
# `gtfs_timetable()`:
gt <- gtfs_timetable (gtfs, day = "Sunday", route_pattern = "^S")
route <- gtfs_route (gt, from = from, to = to, start_time = start_time)

data.table::setDTthreads (nthr_dt)
Sys.setenv ("OMP_THREAD_LIMIT" = nthr_omp)
```

`gtfs_route_headway` *Route headway*

Description

Calculate a vector of headway values – that is, time intervals between consecutive services – for all routes between two specified stations.

Usage

```
gtfs_route_headway(
  gtfs,
  from,
  to,
  from_to_are_ids = FALSE,
  grep_fixed = TRUE,
  quiet = FALSE
)
```

Arguments

`gtfs`	A set of GTFS data returned from extract_gtfs or, for more efficient queries, pre-processed with gtfs_timetable.
`from`	Names, IDs, or approximate (lon, lat) coordinates of start stations (as `stop_name` or `stop_id` entry in the `stops` table, or a vector of two numeric values). See Note.
`to`	Corresponding Names, IDs, or coordinates of end station.
`from_to_are_ids`	Set to `TRUE` to enable `from` and `to` parameter to specify entries in `stop_id` rather than `stop_name` column of the `stops` table.
`grep_fixed`	If `FALSE`, match station names (when passed as character string) with `grep(..., fixed = FALSE)`, to allow use of `grep` expressions. This is useful to refine matches in cases where desired stations may match multiple entries.
`quiet`	If `TRUE`, display a progress bar

Value

A single vector of integer values containing headways between all services across a single 24-hour period

See Also

Other main: `gtfs_route()`, `gtfs_traveltimes()`

Examples

```
## Not run:
path <- berlin_gtfs_to_zip ()
gtfs <- extract_gtfs (path)
gtfs_route_headway (gtfs, from = "Tegel", to = "Berlin Hauptbahnhof")

## End(Not run)
```

gtfs_timetable *gtfs_timetable*

Description

Convert GTFS data into format able to be used to calculate routes.

Usage

```
gtfs_timetable(
  gtfs,
  day = NULL,
  date = NULL,
  route_pattern = NULL,
  quiet = FALSE
)
```

Arguments

`gtfs`	A set of GTFS data returned from extract_gtfs.
`day`	Day of the week on which to calculate route, either as an unambiguous string (so "tu" and "th" for Tuesday and Thursday), or a number between 1 = Sunday and 7 = Saturday. If not given, the current day will be used - unless the following 'date' parameter is give.
`date`	Some systems do not specify days of the week within their 'calendar' table; rather they provide full timetables for specified calendar dates via a 'calendar_date' table. Providing a date here as a single 8-digit number representing 'yyyymmdd' will filter the data to the specified date. Also the 'calendar' is scanned for services that operate on the selected date. Therefore also a merge of feeds that combine 'calendar' and 'calendar_dates' options is covered.
`route_pattern`	Using only those routes matching given pattern, for example, "^U" for routes starting with "U" (as commonly used for underground or subway routes. To negative the `route_pattern` – that is, to include all routes except those matching the patter – prepend the value with "!"; for example "!^U" with include all services except those starting with "U".
`quiet`	Set to `TRUE` to suppress screen messages (currently just regarding timetable construction).

Value

The input data with an addition items, `timetable`, `stations`, and `trips`, containing data formatted for more efficient use with gtfs_route (see Note).

Note

This function is merely provided to speed up calls to the primary function, gtfs_route. If the input data to that function do not include a formatted `timetable`, it will be calculated anyway, but queries in that case will generally take longer.

See Also

Other extract: `berlin_gtfs_to_zip()`, `extract_gtfs()`

Examples

```
# Examples must be run on single thread only:
nthr_dt <- data.table::setDTthreads (1)
nthr_omp <- Sys.getenv ("OMP_THREAD_LIMIT")
Sys.setenv ("OMP_THREAD_LIMIT" = 1L)

berlin_gtfs_to_zip () # Write sample feed from Berlin, Germany to tempdir
f <- file.path (tempdir (), "vbb.zip") # name of feed
gtfs <- extract_gtfs (f)
from <- "Innsbrucker Platz" # U-bahn station, not "S"
to <- "Alexanderplatz"
start_time <- 12 * 3600 + 120 # 12:02

route <- gtfs_route (gtfs, from = from, to = to, start_time = start_time)

# Specify day of week
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday"
)

# specify travel by "U" = underground only
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday",
    route_pattern = "^U"
)
# specify travel by "S" = street-level only (not underground)
route <- gtfs_route (
    gtfs,
    from = from,
    to = to,
    start_time = start_time,
    day = "Sunday",
    route_pattern = "^S"
)
```

```
# Route queries are generally faster if the GTFS data are pre-processed with
# `gtfs_timetable()`:
gt <- gtfs_timetable (gtfs, day = "Sunday", route_pattern = "^S")
route <- gtfs_route (gt, from = from, to = to, start_time = start_time)

data.table::setDTthreads (nthr_dt)
Sys.setenv ("OMP_THREAD_LIMIT" = nthr_omp)
```

---

gtfs_transfer_table *gtfs_transfer_table*

Description

Construct a transfer table for a GTFS feed.

Usage

```
gtfs_transfer_table(
  gtfs,
  d_limit = 200,
  min_transfer_time = 120,
  network = NULL,
  network_times = FALSE,
  quiet = FALSE
)
```

Arguments

`gtfs`	A GTFS feed obtained from the extract_gtfs function.
`d_limit`	Upper straight-line distance limit in metres for transfers.
`min_transfer_time`	Minimum time in seconds for transfers; all values below this will be replaced with this value, particularly all those defining in-place transfers where stop longitudes and latitudes remain identical.
`network`	Optional Open Street Map representation of the street network encompassed by the GTFS feed (see Examples).
`network_times`	If `TRUE`, transfer times are calculated by routing throughout the underlying street network. If this is not provided as the `net` parameter, it will be automatically downloaded. If a network, is provided, this parameter is automatically set to `TRUE`.
`quiet`	Set to `TRUE` to suppress screen messages

Value

Modified version of the `gtfs` input with additional transfers table.

See Also

Other augment: frequencies_to_stop_times()

Examples

```
# Examples must be run on single thread only:
nthr <- data.table::setDTthreads (1)

berlin_gtfs_to_zip ()
f <- file.path (tempdir (), "vbb.zip")
g <- extract_gtfs (f, quiet = TRUE)
g <- gtfs_transfer_table (g, d_limit = 200)
# g$transfers then has fewer rows than original, because original transfer
# table contains duplicated rows.

data.table::setDTthreads (nthr)
```

---

gtfs_traveltimes *gtfs_traveltimes*

Description

Travel times from a nominated station departing at a nominated time to every other reachable station in a system.

Usage

```
gtfs_traveltimes(
  gtfs,
  from,
  start_time_limits,
  day = NULL,
  from_is_id = FALSE,
  grep_fixed = TRUE,
  route_pattern = NULL,
  minimise_transfers = FALSE,
  max_traveltime = 60 * 60,
  quiet = FALSE
)
```

Arguments

`gtfs`	A set of GTFS data returned from extract_gtfs or, for more efficient queries, pre-processed with gtfs_timetable.
`from`	Name, ID, or approximate (lon, lat) coordinates of start station (as `stop_name` or `stop_id` entry in the `stops` table, or a vector of two numeric values).

`start_time_limits`	A vector of two integer values denoting the earliest and latest departure times in seconds for the traveltime values.
`day`	Day of the week on which to calculate route, either as an unambiguous string (so "tu" and "th" for Tuesday and Thursday), or a number between 1 = Sunday and 7 = Saturday. If not given, the current day will be used. (Not used if `gtfs` has already been prepared with gtfs_timetable.)
`from_is_id`	Set to `TRUE` to enable `from` parameter to specify entry in `stop_id` rather than `stop_name` column of the `stops` table (same as `from_to_are_ids` parameter of gtfs_route).
`grep_fixed`	If `FALSE`, match station names (when passed as character string) with `grep(..., fixed = FALSE)`, to allow use of `grep` expressions. This is useful to refine matches in cases where desired stations may match multiple entries.
`route_pattern`	Using only those routes matching given pattern, for example, "^U" for routes starting with "U" (as commonly used for underground or subway routes. To negate the `route_pattern` – that is, to include all routes except those matching the pattern – prepend the value with "!"; for example "!^U" will include all services except those starting with "U". (This parameter is not used at all if `gtfs` has already been prepared with gtfs_timetable.)
`minimise_transfers`	If `TRUE`, isochrones are calculated with minimal-transfer connections to each end point, even if those connections are slower than alternative connections with transfers.
`max_traveltime`	The maximal traveltime to search for, specified in seconds (with default of 1 hour). See note for details.
`quiet`	Set to `TRUE` to suppress screen messages (currently just regarding timetable construction).

Value

A `data.frame` of travel times and required numbers of transfers to all stations reachable from the given `from` station. Additional columns include "start_time" of connection, and information on destination stops including "id" numbers, names, and geographical coordinates.

Note

Higher values of `max_traveltime` will return traveltimes for greater numbers of stations, but may lead to considerably longer calculation times. For repeated usage, it is recommended to first establish a value sufficient to reach all or most stations desired for a particular query, rather than set `max_traveltime` to an arbitrarily high value.

See Also

Other main: `gtfs_route()`, `gtfs_route_headway()`

Examples

```
# Examples must be run on single thread only:
nthr <- data.table::setDTthreads (1)

berlin_gtfs_to_zip ()
f <- file.path (tempdir (), "vbb.zip")
g <- extract_gtfs (f)
g <- gtfs_timetable (g)
from <- "Alexanderplatz"
start_times <- 12 * 3600 + c (0, 60) * 60 # 8:00-9:00
res <- gtfs_traveltimes (g, from, start_times)

data.table::setDTthreads (nthr)
```

`process_gtfs_local` *process_gtfs_local*

Description

Process a local GTFS data set with environmental variables described in go_home into a condensed version for use in go_home and `go_to_work` functions.

Usage

```
process_gtfs_local(expand = 2)
```

Arguments

`expand`	The data set is reduced to the bounding box defined by the home and work stations, expanded by this multiple. If the function appears to behave strangely, try re-running this function with a higher value of this parameter.

Value

No return value. The function saves processed data to a local cache.

See Also

Other additional: `go_home()`, `go_to_work()`, `summary.gtfs()`

Examples

```
## Not run:
# For general use, please set these three variables:
Sys.setenv ("gtfs_home" = "<my home station>")
Sys.setenv ("gtfs_work" = "<my work station>")
Sys.setenv ("gtfs_data" = "/full/path/to/gtfs.zip")

# The following illustrate use with sample data bundled with package
```

```
Sys.setenv ("gtfs_home" = "Tempelhof")
Sys.setenv ("gtfs_work" = "Alexanderplatz")
Sys.setenv ("gtfs_data" = file.path (tempdir (), "vbb.zip"))
process_gtfs_local ()
# next available service from current system time:
go_home ()

## End(Not run)
```

---

summary.gtfs *summary.gtfs*

Description

summary.gtfs

Usage

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

Arguments

`object`	A `gtfs` object to be summarised
`...`	ignored here

Value

Nothing; this function only prints a summary to the console.

See Also

Other additional: `go_home()`, `go_to_work()`, `process_gtfs_local()`

Examples

```
# Examples must be run on single thread only:
nthr <- data.table::setDTthreads (1)

berlin_gtfs_to_zip ()
f <- file.path (tempdir (), "vbb.zip")
g <- extract_gtfs (f)
summary (g)
g <- gtfs_timetable (g)
summary (g) # also summarizes additional timetable information

data.table::setDTthreads (nthr)
```

Index