Package ‘gtfsrouter’

March 22, 2019

Title Routing with GTFS (General Transit Feed Specification) Data

Version 0.0.1

Description Use GTFS (General Transit Feed Specification) data for routing from nominated start and end stations, and for extracting isochrones from nominated start station.

License GPL-3

Depends R (>= 2.10)

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

Suggests alphahull, geodist, here, hms, lubridate, lwgeom, knitr, mapview, rmarkdown, sf, testthat

LinkingTo Rcpp

SystemRequirements C++11

NeedsCompilation yes

VignetteBuilder knitr

Encoding UTF-8

LazyData true

URL `https://github.com/ATFutures/gtfs-router`

BugReports `https://github.com/ATFutures/gtfs-router/issues`

RoxygenNote 6.1.1

Author Mark Padgham [aut, cre]

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

Repository CRAN

Date/Publication 2019-03-22 17:20:20 UTC

R topics documented:

berlin_gtfs 2
berlin_gtfs_to_zip 3
extract_gtfs 4

go_home . . . 4
go_to_work . . . 6
gtfsrouter . . . 7
gtfs_isochrone . . . 7
gtfs_route . . . 9
gtfs_timetable . . . 10
plot.gtfs_ischrone . . . 12
process_gtfs_local . . . 12
summary.gtfs . . . 13

Index **14**

`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

square matrix of distances between nodes

Note

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

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)
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)
```

---

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

Nothing

extract_gtfs *extract_gtfs*

Description

Extract "stop_times" and "transfers" table from a GTFS `zip` archive.

Usage

```
extract_gtfs(filename = NULL)
```

Arguments

`filename`	Name of GTFS archive

Value

List of 2 **data.table** objects, one for "stop_times" and one for "transfers"

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)
```

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 locationn 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 largley unavoidable.

Value

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

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")

## End(Not run)
# 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:
## Not run:
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 locationn 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 largley unavoidable.

Value

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

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")
```

```
## End(Not run)
# 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:
## Not run:
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_isochrone()`: One-to-many routing from a nominated start station to all stations reachable within a specified travel duration.

`gtfs_isochrone` *gtfs_isochrone*

Description

Calculate a single isochrone from a given start station, returning the list of all stations reachable to the specified `end_time`.

Usage

```
gtfs_isochrone(gtfs, from, start_time, end_time, day = NULL,
  route_pattern = NULL, hull_alpha = 0.1, 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 of start 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**.
`end_time`	End time to calculate isochrone
`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. (Parameter not used at all if `gtfs` has already been prepared with gtfs_timetable.)
`hull_alpha`	alpha value of non-convex hulls returned as part of result (see ?alphashape::ashape for details).
`quiet`	Set to `TRUE` to suppress screen messages (currently just regarding timetable construction).

Value

An object of class `gtfs_isochrone`, including **sf**-formatted points representing the `from` station (`start_point`), the terminal end stations (`end_points`), and all intermediate stations (`mid_points`), along with lines representing the individual routes. A non-convex ("alpha") hull is also returned (as an **sf** `POLYGON` object), including measures of area and "elongation", which equals zero for a circle, and increases towards one for more elongated shapes.

Examples

```
berlin_gtfs_to_zip ()
f <- file.path (tempdir (), "vbb.zip")
g <- extract_gtfs (f)
g <- gtfs_timetable (g)
from <- "Alexanderplatz"
start_time <- 12 * 3600 + 600
end_time <- start_time + 600
ic <- gtfs_isochrone (g,
                      from = from,
                      start_time = start_time,
                      end_time = end_time)
## Not run:
plot (ic)
```

```
## End(Not run)
```

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,
  max_transfers = NA, 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 of start station
`to`	Name 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. (Parameter 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.
`max_transfers`	If not `NA`, specify a maximum number of transfers (including but not exceeding this number) for the route.
`quiet`	Set to `TRUE` to suppress screen messages (currently just regarding timetable construction).

Value

square matrix of distances between nodes

Note

This function will by default calculate the route that departs on the first available service after the specified `start_time`, although this may arrive later than subsequent services. If the earliest arriving route is desired, ...

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)
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)
```

`gtfs_timetable` *gtfs_timetable*

Description

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

Usage

```
gtfs_timetable(gtfs, day = 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.
`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.
`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.

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)
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)
```

plot.gtfs_ischrone *plot.gtfs_isochrone*

Description

plot.gtfs_isochrone

Usage

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

Arguments

`x`	object to be plotted
`...`	ignored here

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.

`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

Examples

```
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
```

Index

∗Topic **datasets**
    berlin_gtfs, 2

berlin_gtfs, 2, *3*
berlin_gtfs_to_zip, 3

difftime, *8, 9*

extract_gtfs, 4, *8, 9*, *11*

go_home, 4, *5, 6*, *12*
go_home(), *7*
go_to_work, *5, 6*, 6
go_to_work(), *7*
gtfs_isochrone, 7
gtfs_isochrone(), *7*
gtfs_route, 9, *11*
gtfs_route(), *7*
gtfs_timetable, *8, 9*, 10
gtfsrouter, 7
gtfsrouter-package *(*gtfsrouter*)*, 7

plot.gtfs_ischrone, 12
plot.gtfs_isochrone
        *(*plot.gtfs_ischrone*)*, 12
process_gtfs_local, *5, 6*, 12

Startup, *5, 6*
summary.gtfs, 13