Package 'r5r'

October 26, 2021

Type Package

Title Rapid Realistic Routing with 'R5'

Version 0.6.0

Description Rapid realistic routing on multimodal transport networks (walk, bike, public transport and car) using 'R5', the Rapid Realistic Routing on Real-world and Reimagined networks engine <https://github.com/conveyal/r5>. The package allows users to generate detailed routing analysis or calculate travel time matrices using seamless parallel computing on top of the R5 Java machine. While R5 is developed by Conveyal, the package r5r is independently developed by a team at the Institute for Applied Economic Research (Ipea) with contributions from collaborators. Apart from the documentation in this package, users will find additional information on R5 documentation at <https://docs.conveyal.com/>. Although we try to keep new releases of r5r in synchrony with R5, the development of R5 follows Conveyal's independent update process. Hence, users should confirm the R5 version implied by the Conveyal user manual (see <https://docs.conveyal.com/changelog>) corresponds with the R5 version that r5r depends on.

Date 2021-10-25

URL https://github.com/ipeaGIT/r5r

BugReports https://github.com/ipeaGIT/r5r/issues

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.6)

Suggests akima, covr, dplyr, ggplot2 (>= 3.3.1), knitr, mapview, rmarkdown, rgdal, testthat

Imports checkmate, curl, data.table, httr, methods, raster, rJava (>= 0.9-10), sf (>= 0.9-3), sfheaders, utils

SystemRequirements Java JDK (>= 11.0)

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Marcus Saraiva [aut] (<https://orcid.org/0000-0001-6218-2338>),
Rafael H. M. Pereira [aut, cre]
(<https://orcid.org/0000-0003-2125-7465>),
Daniel Herszenhut [aut] (<https://orcid.org/0000-0001-8066-1105>),
Carlos Kaue Vieira Braga [aut]
(<https://orcid.org/0000-0002-6104-7297>),
Matthew Wigginton Conway [aut]
(<https://orcid.org/0000-0002-1210-2982>),
Ipea - Institute for Applied Economic Research [cph, fnd]

Maintainer Rafael H. M. Pereira <rafa.pereira.br@gmail.com>

Repository CRAN

Date/Publication 2021-10-26 13:20:02 UTC

R topics documented:

`accessibility` *Calculate access to opportunities*

Description

Fast computation of access to opportunities given a selected decay function. See `details` for the available decay functions.

Usage

```
accessibility(
  r5r_core,
  origins,
  destinations,
  opportunities_colname = "opportunities",
  mode = "WALK",
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  time_window = 1L,
  percentiles = 50L,
  decay_function = "step",
  cutoffs = 30L,
  decay_value = 1,
  max_walk_dist = Inf,
  max_bike_dist = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  n_threads = Inf,
  verbose = TRUE,
  progress = TRUE
)
```

Arguments

`r5r_core`	a rJava object to connect with R5 routing engine
`origins, destinations`	a spatial sf POINT object with WGS84 CRS, or a data.frame containing the columns 'id', 'lon', 'lat'.
`opportunities_colname`	string. The column name in the `destinations` input that tells the number of opportunities in each location. Defaults to "opportunities".
`mode`	string. Transport modes allowed for the trips. Defaults to "WALK". See details for other options.

mode_egress string. Transport mode used after egress from public transport. It can be either 'WALK', 'BICYCLE', or 'CAR'. Defaults to "WALK".

departure_datetime
POSIXct object. If working with public transport networks, please check `calendar.txt` within the GTFS file for valid dates. See details for further information on how datetimes are parsed.

time_window numeric. Time window in minutes for which r5r will calculate travel times departing each minute. When using frequency-based GTFS files, 5 Monte Carlo simulations will be run for each minute in the time window. See details for further information.

percentiles numeric vector. Defaults to '50', returning the accessibility value for the median travel time computed for a given time_window. If a numeric vector is passed, for example c(25, 50, 75), the function will return accessibility estimates for each percentile, by travel time cutoff. Only the first 5 cut points of the percentiles are considered. For more details, see R5 documentation at 'https://docs.conveyal.com/analysis/methodology#accounting-for-variability'

decay_function string. Choice of one of the following decay functions: 'step', 'exponential', 'fixed_exponential', 'linear', and 'logistic'. Defaults to 'step', which yields cumulative opportunities accessibility metrics. More info in `details`.

cutoffs numeric. Cutoff times in minutes for calculating cumulative opportunities accessibility when using the 'step decay function'. This parameter has different effects for each of the other decay functions: it indicates the 'median' (or inflection point) of the decay curves in the 'logistic' and 'linear' functions, and the 'half-life' in the 'exponential' function. It has no effect when using the 'fixed exponential' function.

decay_value numeric. Extra parameter to be passed to the selected `decay_function`.

max_walk_dist numeric. Maximum walking distance (in meters) to access and egress the transit network, or to make transfers within the network. Defaults to no restrictions as long as `max_trip_duration` is respected. The max distance is considered separately for each leg (e.g. if you set `max_walk_dist` to 1000, you could potentially walk up to 1 km to reach transit, and up to *another* 1 km to reach the destination after leaving transit). Obs: if you want to set the maximum walking distance considering walking-only trips you have to set the `max_trip_duration` accordingly (e.g. to set a distance of 1 km assuming a walking speed of 3.6 km/h you have to set `max_trip_duration = 1 / 3.6 * 60`).

max_bike_dist numeric. Maximum cycling distance (in meters) to access and egress the transit network. Defaults to no restrictions as long as `max_trip_duration` is respected. The max distance is considered separately for each leg (e.g. if you set `max_bike_dist` to 1000, you could potentially cycle up to 1 km to reach transit, and up to *another* 1 km to reach the destination after leaving transit). Obs: if you want to set the maximum cycling distance considering cycling-only trips you have to set the `max_trip_duration` accordingly (e.g. to set a distance of 5 km assuming a cycling speed of 12 km/h you have to set `max_trip_duration = 5 / 12 * 60`).

max_trip_duration
numeric. Maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

walk_speed	numeric. Average walk speed in km/h. Defaults to 3.6 km/h.
bike_speed	numeric. Average cycling speed in km/h. Defaults to 12 km/h.
max_rides	numeric. The max number of public transport rides allowed in the same trip. Defaults to 3.
max_lts	numeric (between 1 and 4). The maximum level of traffic stress that cyclists will tolerate. A value of 1 means cyclists will only travel through the quietest streets, while a value of 4 indicates cyclists can travel through any road. Defaults to 2. See details for more information.
n_threads	numeric. The number of threads to use in parallel computing. Defaults to use all available threads (Inf).
verbose	logical. TRUE to show detailed output messages (the default).
progress	logical. TRUE to show a progress counter. Only works when verbose is set to FALSE, so the progress counter does not interfere with R5's output messages. Setting progress to TRUE may impose a small penalty for computation efficiency, because the progress counter must be synchronized among all active threads.

Value

A data.table with accessibility estimates for all origin points, by a given transport mode, and per travel time cutoff and percentile.

Decay functions:

R5 allows for multiple decay functions. More info in the original R5 documentation from Conveyal, at https://docs.conveyal.com/learn-more/decay-functions The options include:

Step `step` **(cumulative opportunities):**

A binary decay function used to calculate cumulative opportunities metrics.

Logistic CDF `logistic`**:**

This is the logistic function, i.e. the cumulative distribution function of the logistic distribution, expressed such that its parameters are the median (inflection point) and standard deviation. This function applies a sigmoid rolloff that has a convenient relationship to discrete choice theory. Its parameters can be set to reflect a whole population's tolerance for making trips with different travel times. The function's value represents the probability that a randomly chosen member of the population would accept making a trip, given its duration. Opportunities are then weighted by how likely it is a person would consider them "reachable".

calibration:
The median parameter is controlled by the `cutoff` parameter, leaving only the standard deviation to configure through the `decay_value` parameter.

Fixed Exponential `fixed_exponential`**:**

This function is of the form e-Lt where L is a single fixed decay constant in the range (0, 1). It is constrained to be positive to ensure weights decrease (rather than grow) with increasing travel time.

calibration:
This function is controlled exclusively by the L constant, given by the `decay_value` parameter. Values provided in `cutoffs` are ignored.

Half-life Exponential Decay `exponential`**:**

This is similar to the fixed-exponential option above, but in this case the decay parameter is inferred from the `cutoffs` parameter values, which is treated as the half-life of the decay.

Linear `linear`**:**

This is a simple, vaguely sigmoid option, which may be useful when you have a sense of a maximum travel time that would be tolerated by any traveler, and a minimum time below which all travel is perceived to be equally easy.

calibration:
The transition region is transposable and symmetric around the `cutoffs` parameter values, taking `decay_value` minutes to taper down from one to zero.

Transport modes:

R5 allows for multiple combinations of transport modes. The options include:

Transit modes:

TRAM, SUBWAY, RAIL, BUS, FERRY, CABLE_CAR, GONDOLA, FUNICULAR. The option 'TRANSIT' automatically considers all public transport modes available.

Non transit modes:

WALK, BICYCLE, CAR, BICYCLE_RENT, CAR_PARK

max_lts, Maximum Level of Traffic Stress:

When cycling is enabled in R5, setting `max_lts` will allow cycling only on streets with a given level of danger/stress. Setting `max_lts` to 1, for example, will allow cycling only on separated bicycle infrastructure or low-traffic streets; routing will revert to walking when traversing any links with LTS exceeding 1. Setting `max_lts` to 3 will allow cycling on links with LTS 1, 2, or 3.

The default methodology for assigning LTS values to network edges is based on commonly tagged attributes of OSM ways. See more info about LTS in the original documentation of R5 from Conveyal at https://docs.conveyal.com/learn-more/traffic-stress. In summary:

- **LTS 1**: Tolerable for children. This includes low-speed, low-volume streets, as well as those with separated bicycle facilities (such as parking-protected lanes or cycle tracks).
- **LTS 2**: Tolerable for the mainstream adult population. This includes streets where cyclists have dedicated lanes and only have to interact with traffic at formal crossing.
- **LTS 3**: Tolerable for "enthused and confident" cyclists. This includes streets which may involve close proximity to moderate- or high-speed vehicular traffic.
- **LTS 4**: Tolerable for only "strong and fearless" cyclists. This includes streets where cyclists are required to mix with moderate- to high-speed vehicular traffic.

For advanced users, you can provide custom LTS values by adding a tag <key = "lts> to the `osm.pbf` file

Routing algorithm:

The `accessibility()` function uses an R5-specific extension to the RAPTOR routing algorithm (see Conway et al., 2017). This RAPTOR extension uses a systematic sample of one departure per minute over the time window set by the user in the 'time_window' parameter. A detailed description of base RAPTOR can be found in Delling et al (2015).

- Conway, M. W., Byrd, A., & van der Linden, M. (2017). Evidence-based transit and land use sketch planning using interactive accessibility methods on combined schedule and headway-based networks. Transportation Research Record, 2653(1), 45-53.
- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604.

Datetime parsing

`r5r` ignores the timezone attribute of datetime objects when parsing dates and times, using the study area's timezone instead. For example, let's say you are running some calculations using Rio de Janeiro, Brazil, as your study area. The datetime `as.POSIXct("13-05-2019 14:00:00",format = "%d-%m-%Y %H:%M:%S")` will be parsed as May 13th, 2019, 14:00h in Rio's local time, as expected. But `as.POSIXct("13-05-2019 14:00:00",format = "%d-%m-%Y %H:%M:%S",tz = "Europe/Paris")` will also be parsed as the exact same date and time in Rio's local time, perhaps surprisingly, ignoring the timezone attribute.

See Also

Other routing: `detailed_itineraries()`, `travel_time_matrix()`

Examples

```
if (interactive()) {
library(r5r)

# build transport network
data_path <- system.file("extdata/poa", package = "r5r")
r5r_core <- setup_r5(data_path = data_path, temp_dir = TRUE)

# load origin/destination points
points <- read.csv(file.path(data_path, "poa_hexgrid.csv"))

  access <- accessibility(r5r_core,
                          origins = points,
                          destinations = points,
                          opportunities_colname = "schools",
                          mode = "WALK",
                          cutoffs = c(25, 30),
                          max_trip_duration = 30,
                          verbose = FALSE)

stop_r5(r5r_core)

}
```

apply_elevation *Apply elevation to street network*

Description

Loads a Digital Elevation Model (DEM) from a raster file and weights the street network for walking and cycling according to the terrain's slopes

Usage

```
apply_elevation(r5r_core, raster_files)
```

Arguments

r5r_core	a rJava object to connect with R5 routing engine
raster_files	string. Path to raster files containing the study area's topography. If a list is provided, all the rasters are automatically merged.

Value

No return value, called for side effects.

See Also

Other elevation support functions: tobler_hiking()

assert_breakdown_stat *Assert travel times breakdown stat parameter value*

Description

Assert travel times breakdown stat parameter value

Usage

```
assert_breakdown_stat(breakdown_stat)
```

Arguments

breakdown_stat	Name of statistic function (minimum or average/mean).

Value

A character with the validated statistic function name.

See Also

Other support functions: assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

---

assert_decay_function *Assert decay function and parameter values*

Description

Assert decay function and parameter values

Usage

```
assert_decay_function(decay_function, decay_value)
```

Arguments

decay_function	Name of decay function.
decay_value	Value of decay parameter.

Value

A list with the validated decay function and parameter value.

See Also

Other support functions: assert_breakdown_stat(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

---

assert_points_input *Assert class of origin and destination inputs and the type of its columns*

Description

Assert class of origin and destination inputs and the type of its columns

Usage

```
assert_points_input(df, name)
```

Arguments

df	Any object.
name	Object name.

Value

A data.frame with columns id, lon and lat.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

---

check_connection *Check internet connection with Ipea server*

Description

Checks if there is internet connection to Ipea server to download geobr data.

Usage

```
check_connection(
  file_url = "https://www.ipea.gov.br/geobr/metadata/metadata_gpkg.csv"
)
```

Arguments

file_url	A string with the file_url address of an geobr dataset

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

`detailed_itineraries` *Calculate detailed itineraries between origin destination pairs*

Description

Fast computation of (multiple) detailed itineraries between one or many origin destination pairs.

Usage

```
detailed_itineraries(
  r5r_core,
  origins,
  destinations,
  mode = "WALK",
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  max_walk_dist = Inf,
  max_bike_dist = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  shortest_path = TRUE,
  n_threads = Inf,
  verbose = TRUE,
  progress = TRUE,
  drop_geometry = FALSE
)
```

Arguments

`r5r_core`	rJava object to connect with R5 routing engine
`origins, destinations`	a spatial sf POINT object with WGS84 CRS, or a data.frame containing the columns 'id', 'lon', 'lat'.
`mode`	string. Transport modes allowed for the trips. Defaults to "WALK". See details for other options.
`mode_egress`	string. Transport mode used after egress from public transport. It can be either 'WALK', 'BICYCLE', or 'CAR'. Defaults to "WALK". Ignored when public transport is not used.
`departure_datetime`	POSIXct object. If working with public transport networks, please check `calendar.txt` within the GTFS file for valid dates. See details for further information on how datetimes are parsed.

`max_walk_dist` numeric. Maximum walking distance (in meters) to access and egress the transit network, or to make transfers within the network. Defaults to no restrictions as long as `max_trip_duration` is respected. The max distance is considered separately for each leg (e.g. if you set `max_walk_dist` to 1000, you could potentially walk up to 1 km to reach transit, and up to *another* 1 km to reach the destination after leaving transit). Obs: if you want to set the maximum walking distance considering walking-only trips you have to set the `max_trip_duration` accordingly (e.g. to set a distance of 1 km assuming a walking speed of 3.6 km/h you have to set `max_trip_duration = 1 / 3.6 * 60`).

`max_bike_dist` numeric. Maximum cycling distance (in meters) to access and egress the transit network. Defaults to no restrictions as long as `max_trip_duration` is respected. The max distance is considered separately for each leg (e.g. if you set `max_bike_dist` to 1000, you could potentially cycle up to 1 km to reach transit, and up to *another* 1 km to reach the destination after leaving transit). Obs: if you want to set the maximum cycling distance considering cycling-only trips you have to set the `max_trip_duration` accordingly (e.g. to set a distance of 5 km assuming a cycling speed of 12 km/h you have to set `max_trip_duration = 5 / 12 * 60`).

`max_trip_duration` numeric. Maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

`walk_speed` numeric. Average walk speed in km/h. Defaults to 3.6 km/h.

`bike_speed` numeric. Average cycling speed in km/h. Defaults to 12 km/h.

`max_rides` numeric. The max number of public transport rides allowed in the same trip. Defaults to 3.

`max_lts` numeric (between 1 and 4). The maximum level of traffic stress that cyclists will tolerate. A value of 1 means cyclists will only travel through the quietest streets, while a value of 4 indicates cyclists can travel through any road. Defaults to 2. See details for more information.

`shortest_path` logical. Whether the function should only return the fastest route alternative (the default) or multiple alternatives.

`n_threads` numeric. The number of threads to use in parallel computing. Defaults to use all available threads (Inf).

`verbose` logical. `TRUE` to show detailed output messages (the default).

`progress` logical. `TRUE` to show a progress counter. Only works when `verbose` is set to `FALSE`, so the progress counter does not interfere with R5's output messages. Setting `progress` to `TRUE` may impose a small penalty for computation efficiency, because the progress counter must be synchronized among all active threads.

`drop_geometry` logical. Indicates whether R5 should drop segment's geometry column. It can be helpful for saving memory.

Value

A LINESTRING sf with detailed information about the itineraries between specified origins and destinations. Distances are in meters and travel times are in minutes.

Transport modes:

R5 allows for multiple combinations of transport modes. The options include:

Transit modes:
TRAM, SUBWAY, RAIL, BUS, FERRY, CABLE_CAR, GONDOLA, FUNICULAR. The option 'TRANSIT' automatically considers all public transport modes available.

Non transit modes:
WALK, BICYCLE, CAR, BICYCLE_RENT, CAR_PARK

max_lts, Maximum Level of Traffic Stress:

When cycling is enabled in R5, setting `max_lts` will allow cycling only on streets with a given level of danger/stress. Setting `max_lts` to 1, for example, will allow cycling only on separated bicycle infrastructure or low-traffic streets; routing will revert to walking when traversing any links with LTS exceeding 1. Setting `max_lts` to 3 will allow cycling on links with LTS 1, 2, or 3.

The default methodology for assigning LTS values to network edges is based on commonly tagged attributes of OSM ways. See more info about LTS in the original documentation of R5 from Conveyal at https://docs.conveyal.com/learn-more/traffic-stress. In summary:

- **LTS 1**: Tolerable for children. This includes low-speed, low-volume streets, as well as those with separated bicycle facilities (such as parking-protected lanes or cycle tracks).
- **LTS 2**: Tolerable for the mainstream adult population. This includes streets where cyclists have dedicated lanes and only have to interact with traffic at formal crossing.
- **LTS 3**: Tolerable for "enthused and confident" cyclists. This includes streets which may involve close proximity to moderate- or high-speed vehicular traffic.
- **LTS 4**: Tolerable for only "strong and fearless" cyclists. This includes streets where cyclists are required to mix with moderate- to high-speed vehicular traffic.

For advanced users, you can provide custom LTS values by adding a tag <key = "lts> to the `osm.pbf` file

Routing algorithm:

The detailed_itineraries function uses an R5-specific extension to the McRAPTOR routing algorithm to find paths that are optimal or less than optimal, with some heuristics around multiple access modes, riding the same patterns, etc. The specific extension to McRAPTOR to do suboptimal path routing are not documented yet, but a detailed description of base McRAPTOR can be found in Delling et al (2015).

- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604.

Datetime parsing

`r5r` ignores the timezone attribute of datetime objects when parsing dates and times, using the study area's timezone instead. For example, let's say you are running some calculations using Rio de Janeiro, Brazil, as your study area. The datetime `as.POSIXct("13-05-2019 14:00:00",format = "%d-%m-%Y %H:%M:%S")` will be parsed as May 13th, 2019, 14:00h in Rio's local time, as expected.

But `as.POSIXct("13-05-2019 14:00:00",format = "%d-%m-%Y %H:%M:%S",tz = "Europe/Paris")` will also be parsed as the exact same date and time in Rio's local time, perhaps surprisingly, ignoring the timezone attribute.

See Also

Other routing: `accessibility()`, `travel_time_matrix()`

Examples

```
if (interactive()) {
library(r5r)

# build transport network
data_path <- system.file("extdata/poa", package = "r5r")
r5r_core <- setup_r5(data_path = data_path, temp_dir = TRUE)

# load origin/destination points
points <- read.csv(file.path(data_path, "poa_points_of_interest.csv"))

# inputs
departure_datetime <- as.POSIXct("13-05-2019 14:00:00", format = "%d-%m-%Y %H:%M:%S")

dit <- detailed_itineraries(r5r_core,
                            origins = points[10,],
                            destinations = points[12,],
                            mode = c("WALK", "TRANSIT"),
                            departure_datetime = departure_datetime,
                            max_walk_dist = 1000,
                            max_trip_duration = 120L)

stop_r5(r5r_core)
}
```

`download_r5` *Download R5 Jar file*

Description

Download a compiled JAR file of R5 and saves it locally. The JAR file is saved within the package directory. The package uses a compilation of R5 tailored for the purposes of r5r that keeps R5's essential features. Source code available at https://github.com/ipeaGIT/r5r.

Usage

```
download_r5(
  version = "6.4.0",
  quiet = FALSE,
  force_update = FALSE,
  temp_dir = FALSE
)
```

Arguments

version	string with the version of R5 to be downloaded. Defaults to latest version '6.4'.
quiet	logical, passed to download.file. Defaults to FALSE
force_update	logical, Replaces the jar file stored locally with a new one. Defaults to FALSE.
temp_dir	logical, whether the R5 Jar file should be saved in temporary directory. Defaults to FALSE

Value

A jar file is saved locally in the r5r package directory

See Also

Other setup: `setup_r5()`

Examples

```
if (interactive()) {

library(r5r)

download_r5(version = "6.4.0", temp_dir = TRUE)
}
```

---

filename_from_metadata

Get most recent JAR filename from metadata

Description

Returns the most recent JAR filename from metadata, depending on the version.

Usage

```
filename_from_metadata(version)
```

Arguments

version	A string, the version of R5's to get the filename of.

Value

The filename as a string.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

---

find_snap *Find snapped locations of input points on street network*

Description

R5 tries to snap origin and destination points to the street network in two rounds. First, it uses a search radius of 300 meters. If the first round is unsuccessful, then R5 expands the search radius to 1.6 km. Points that aren't linked to the street network after those two rounds are returned with NA coordinates and found = FALSE. Please note that the location of the snapped points depends on the transport mode set by the user.

Usage

```
find_snap(r5r_core, points, mode = "WALK")
```

Arguments

r5r_core	a rJava object to connect with R5 routing engine
points	a spatial sf POINT object, or a data.frame containing the columns 'id', 'lon', 'lat'
mode	string. Defaults to "WALK", also allows "BICYCLE", and "CAR".

Value

A data.table with the original points as well as their respective snapped coordinates on the street network and the Euclidean distance between original points and their respective snapped location. Points that could not be snapped show NA coordinates and found = FALSE.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

Examples

```
if (interactive()) {

library(r5r)

# build transport network
path <- system.file("extdata/spo", package = "r5r")
r5r_core <- setup_r5(data_path = path, temp_dir = TRUE)

# load origin/destination points
points <- read.csv(file.path(path, "spo_hexgrid.csv"))

# find where origin or destination points are snapped
snap_df <- find_snap(r5r_core,
                     points = points,
                     mode = 'CAR')

stop_r5(r5r_core)
}
```

`java_to_dt` *Java object to data.table*

Description

Converts a Java object returned by r5r_core to an R data.table

Usage

```
java_to_dt(obj)
```

Arguments

`obj`	A Java Object reference

Value

An R data.table

posix_to_string *Generate date and departure time strings from POSIXct*

Description

Generate date and departure time strings from POSIXct

Usage

```
posix_to_string(datetime)
```

Arguments

datetime	An object of POSIXct class.

Value

A list with the date and time of the trip departure as characters.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

select_mode *Select transport mode*

Description

Select transport mode

Usage

```
select_mode(mode, mode_egress)
```

Arguments

mode	character string passed from routing functions.
mode_egress	character string passed from routing functions.

Value

A list with the transport modes used in the routing.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

---

setup_r5 *Create transport network used for routing in R5*

Description

Combine data inputs in a directory to build a multimodal transport network used for routing in R5. The directory must contain at least one street network file (in .pbf format). One or more public transport data sets (in GTFS.zip format) are optional. If there is more than one GTFS file in the directory, both files will be merged. If there is already a 'network.dat' file in the directory the function will simply read it and load it to memory.

Usage

```
setup_r5(
  data_path,
  version = "6.4.0",
  verbose = TRUE,
  temp_dir = FALSE,
  use_elevation = FALSE,
  overwrite = FALSE
)
```

Arguments

data_path	character string, the directory where data inputs are stored and where the built network.dat will be saved.
version	character string, the version of R5 to be used. Defaults to latest version '6.4.0'.
verbose	logical, TRUE to show detailed output messages (Default) or FALSE to show only eventual ERROR and WARNING messages.
temp_dir	logical, whether the R5 Jar file should be saved in temporary directory. Defaults to FALSE
use_elevation	logical. If TRUE, load any tif files containing elevation found in the data_path folder and calculate impedances for walking and cycling based on street slopes.
overwrite	logical, whether to overwrite an existing network.dat or to use a cached file. Defaults to FALSE (i.e. use a cached network).

Value

An rJava object to connect with R5 routing engine

See Also

Other setup: download_r5()

Examples

```
if (interactive()) {

library(r5r)

# directory with street network and gtfs files
path <- system.file("extdata/poa", package = "r5r")

r5r_core <- setup_r5(data_path = path, temp_dir = TRUE)

}
```

---

set_max_lts *Set max Level of Transit Stress (LTS)*

Description

Set max Level of Transit Stress (LTS)

Usage

```
set_max_lts(r5r_core, max_lts)
```

Arguments

r5r_core	rJava object to connect with R5 routing engine
max_lts	numeric (between 1 and 4). The maximum level of traffic stress that cyclists will tolerate. A value of 1 means cyclists will only travel through the quietest streets, while a value of 4 indicates cyclists can travel through any road.

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

set_max_rides *Set max number of transfers*

Description

Set maxTransfers parameter in R5.

Usage

```
set_max_rides(r5r_core, max_rides)
```

Arguments

r5r_core	rJava object to connect with R5 routing engine
max_rides	numeric. The max number of public transport rides allowed in the same trip. Passed from routing function.

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

set_max_street_time *Set max street time*

Description

Converts a time duration and speed input and converts it to distances.

Usage

```
set_max_street_time(max_walk_dist, walk_speed, max_trip_duration)
```

Arguments

max_walk_dist	numeric, Maximum walking distance (in meters) for the whole trip. Passed from routing functions.
walk_speed	numeric, Average walk speed in Km/h. Defaults to 3.6 Km/h. Passed from routing functions.
max_trip_duration	numeric, Maximum trip duration in seconds. Defaults to 120 minutes (2 hours). Passed from routing functions.

Value

An integer representing the maximum number of minutes walking

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

---

set_n_threads *Set number of threads*

Description

Sets number of threads to be used by the r5r .jar.

Usage

```
set_n_threads(r5r_core, n_threads)
```

Arguments

r5r_core	a rJava object to connect with R5 routing engine
n_threads	Any object.

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

set_progress *Set progress argument*

Description

Set progress argument

Usage

```
set_progress(r5r_core, progress)
```

Arguments

r5r_core	a rJava object to connect with R5 routing engine
progress	logical, passed from function above

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

set_speed *Set walk and bike speed*

Description

This function receives the walk and bike 'speed' inputs in Km/h from routing functions above and converts them to meters per second, which is then used to set these speed profiles in r5r JAR.

Usage

```
set_speed(r5r_core, speed, mode)
```

Arguments

r5r_core	a rJava object to connect with R5 routing engine
speed	A numeric representing the speed in km/h.
mode	Either "bike" or "walk".

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_suboptimal_minutes(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

`set_suboptimal_minutes` *Set suboptimal minutes*

Description

Set suboptimalMinutes parameter in R5.

Usage

```
set_suboptimal_minutes(r5r_core, suboptimal_minutes)
```

Arguments

`r5r_core`	rJava object to connect with R5 routing engine
`suboptimal_minutes`	numeric. The number of suboptimal minutes in a public transport point-to-point query. From R5's documentation: This parameter compensates for the fact that GTFS does not contain information about schedule deviation (lateness). The min-max travel time range for some trains is zero, since the trips are reported to always have the same timings in the schedule. Such an option does not overlap (temporally) its alternatives, and is too easily eliminated by an alternative that is only marginally better. We want to effectively push the max travel time of alternatives out a bit to account for the fact that they don't always run on schedule.

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_verbose(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

set_verbose *Set verbose argument*

Description

Set verbose argument

Usage

```
set_verbose(r5r_core, verbose)
```

Arguments

r5r_core	a rJava object to connect with R5 routing engine
verbose	logical, passed from function above

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), stop_r5(), street_network_to_sf(), transit_network_to_sf()

stop_r5 *Stop running r5r core*

Description

Stops running r5r cores.

Usage

```
stop_r5(...)
```

Arguments

...	r5r_core objects currently running. By default, if no cores are supplied all running cores are stopped.

Value

No return value, called for side effects.

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), street_network_to_sf(), transit_network_to_sf()

Examples

```
if (interactive()) {
  library(r5r)

  path <- system.file("extdata/poa", package = "r5r")

  r5r_core <- setup_r5(data_path = path, temp_dir = TRUE)

  stop_r5(r5r_core)
}
```

---

street_network_to_sf *Extract OpenStreetMap network in sf format from a network.dat built with setup_r5*

Description

Extract OpenStreetMap network in sf format from a network.dat built with setup_r5

Usage

```
street_network_to_sf(r5r_core)
```

Arguments

r5r_core	a rJava object, the output from 'r5r::setup_r5()'

Value

A list with two components of a street network in sf format: vertices (POINT) and edges (LINESTRING).

See Also

Other support functions: assert_breakdown_stat(), assert_decay_function(), assert_points_input(), check_connection(), filename_from_metadata(), find_snap(), posix_to_string(), select_mode(), set_max_lts(), set_max_rides(), set_max_street_time(), set_n_threads(), set_progress(), set_speed(), set_suboptimal_minutes(), set_verbose(), stop_r5(), transit_network_to_sf()

Examples

```
if (interactive()) {

library(r5r)

# build transport network
path <- system.file("extdata/poa", package = "r5r")
r5r_core <- setup_r5(data_path = path, temp_dir = TRUE)

# extract street network from r5r_core
street_net <- street_network_to_sf(r5r_core)

stop_r5(r5r_core)
}
```

---

tobler_hiking *Tobler's hiking function*

Description

Calculates effect of the topography on walking speeds, using Tobler's hiking function.

Usage

```
tobler_hiking(slope)
```

Arguments

slope	numeric. Terrain's slope.

Value

numeric. Tobler's weighting factor

See Also

Other elevation support functions: apply_elevation()

`transit_network_to_sf` *Extract transit network in sf format*

Description

Extract transit network in `sf` format from a `network.dat` file built with the `setup_r5` function.

Usage

```
transit_network_to_sf(r5r_core)
```

Arguments

`r5r_core`	a rJava object, the output from 'r5r::setup_r5()'

Value

A list with two components of a transit network in sf format: route shapes (LINESTRING) and transit stops (POINT). The same `route_id`/`short_name` might appear with different geometries. This occurs when a route has two different shape_ids. Some transit stops might be returned with geometry POINT EMPTY (i.e. missing `NA` spatial coordinates). This may occur when a transit stop is not snapped to the road network, possibly because the `gtfs.zip` input data covers an area larger than the `osm.pbf` input data.

See Also

Other support functions: `assert_breakdown_stat()`, `assert_decay_function()`, `assert_points_input()`, `check_connection()`, `filename_from_metadata()`, `find_snap()`, `posix_to_string()`, `select_mode()`, `set_max_lts()`, `set_max_rides()`, `set_max_street_time()`, `set_n_threads()`, `set_progress()`, `set_speed()`, `set_suboptimal_minutes()`, `set_verbose()`, `stop_r5()`, `street_network_to_sf()`

Examples

```
if (interactive()) {

library(r5r)

# build transport network
path <- system.file("extdata/poa", package = "r5r")
r5r_core <- setup_r5(data_path = path, temp_dir = TRUE)

# extract transit network from r5r_core
transit_net <- transit_network_to_sf(r5r_core)

stop_r5(r5r_core)
}
```

`travel_time_matrix` *Calculate travel time matrix between origin destination pairs*

Description

Fast computation of travel time estimates between one or multiple origin destination pairs.

Usage

```
travel_time_matrix(
  r5r_core,
  origins,
  destinations,
  mode = "WALK",
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  time_window = 1L,
  percentiles = 50L,
  breakdown = FALSE,
  breakdown_stat = "MEAN",
  max_walk_dist = Inf,
  max_bike_dist = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  n_threads = Inf,
  verbose = TRUE,
  progress = TRUE
)
```

Arguments

`r5r_core`	a rJava object to connect with R5 routing engine
`origins, destinations`	a spatial sf POINT object with WGS84 CRS, or a data.frame containing the columns 'id', 'lon', 'lat'.
`mode`	string. Transport modes allowed for the trips. Defaults to "WALK". See details for other options.
`mode_egress`	string. Transport mode used after egress from public transport. It can be either 'WALK', 'BICYCLE', or 'CAR'. Defaults to "WALK".
`departure_datetime`	POSIXct object. If working with public transport networks, please check `calendar.txt` within the GTFS file for valid dates. See details for further information on how datetimes are parsed.

`time_window` numeric. Time window in minutes for which r5r will calculate multiple travel time matrices departing each minute. By default, the number of simulations is 5 times the size of 'time_window' set by the user. Defaults window size to '1', the function only considers 5 departure times. This parameter is only used with frequency-based GTFS files. See details for further information.

`percentiles` numeric vector. Defaults to '50', returning the median travel time for a given time_window. If a numeric vector is passed, for example c(25, 50, 75), the function will return additional columns with the travel times within percentiles of trips. For example, if the 25 percentile of trips between A and B is 15 minutes, this means that 25% of all trips taken between A and B within the set time window are shorter than 15 minutes. Only the first 5 cut points of the percentiles are considered. For more details, see R5 documentation at 'https://docs.conveyal.com/analysis/methodology#for-variability'

`breakdown` logic. If `FALSE` (default), the function returns a simple output with columns origin, destination and travel time percentiles. If `TRUE`, r5r breaks down the trip information and returns more columns with estimates of `access_time`, `waiting_time`, `ride_time`, `transfer_time`, `total_time` , `n_rides` and `route`. Warning: Setting `TRUE` makes the function significantly slower.

`breakdown_stat` string. If `min`, all the brokendown trip informantion is based on the trip itinerary with the smallest waiting time in the time window. If `breakdown_stat = mean`, the information is based on the trip itinerary whose waiting time is the closest to the average waiting time in the time window.

`max_walk_dist` numeric. Maximum walking distance (in meters) to access and egress the transit network, or to make transfers within the network. Defaults to no restrictions as long as `max_trip_duration` is respected. The max distance is considered separately for each leg (e.g. if you set `max_walk_dist` to 1000, you could potentially walk up to 1 km to reach transit, and up to *another* 1 km to reach the destination after leaving transit). Obs: if you want to set the maximum walking distance considering walking-only trips you have to set the `max_trip_duration` accordingly (e.g. to set a distance of 1 km assuming a walking speed of 3.6 km/h you have to set `max_trip_duration = 1 / 3.6 * 60`).

`max_bike_dist` numeric. Maximum cycling distance (in meters) to access and egress the transit network. Defaults to no restrictions as long as `max_trip_duration` is respected. The max distance is considered separately for each leg (e.g. if you set `max_bike_dist` to 1000, you could potentially cycle up to 1 km to reach transit, and up to *another* 1 km to reach the destination after leaving transit). Obs: if you want to set the maximum cycling distance considering cycling-only trips you have to set the `max_trip_duration` accordingly (e.g. to set a distance of 5 km assuming a cycling speed of 12 km/h you have to set `max_trip_duration = 5 / 12 * 60`).

`max_trip_duration` numeric. Maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

`walk_speed` numeric. Average walk speed in km/h. Defaults to 3.6 km/h.

`bike_speed` numeric. Average cycling speed in km/h. Defaults to 12 km/h.

`max_rides` numeric. The max number of public transport rides allowed in the same trip. Defaults to 3.

max_lts	numeric (between 1 and 4). The maximum level of traffic stress that cyclists will tolerate. A value of 1 means cyclists will only travel through the quietest streets, while a value of 4 indicates cyclists can travel through any road. Defaults to 2. See details for more information.
n_threads	numeric. The number of threads to use in parallel computing. Defaults to use all available threads (Inf).
verbose	logical. TRUE to show detailed output messages (the default).
progress	logical. TRUE to show a progress counter. Only works when verbose is set to FALSE, so the progress counter does not interfere with R5's output messages. Setting progress to TRUE may impose a small penalty for computation efficiency, because the progress counter must be synchronized among all active threads.

Value

A data.table with travel time estimates (in minutes) between origin destination pairs by a given transport mode. Note that origins/destinations that were beyond the maximum travel time, and/or origins that were far from the street network are not returned in the data.table.

Transport modes:

R5 allows for multiple combinations of transport modes. The options include:

Transit modes:

TRAM, SUBWAY, RAIL, BUS, FERRY, CABLE_CAR, GONDOLA, FUNICULAR. The option 'TRANSIT' automatically considers all public transport modes available.

Non transit modes:

WALK, BICYCLE, CAR, BICYCLE_RENT, CAR_PARK

max_lts, Maximum Level of Traffic Stress:

When cycling is enabled in R5, setting max_lts will allow cycling only on streets with a given level of danger/stress. Setting max_lts to 1, for example, will allow cycling only on separated bicycle infrastructure or low-traffic streets; routing will revert to walking when traversing any links with LTS exceeding 1. Setting max_lts to 3 will allow cycling on links with LTS 1, 2, or 3.

The default methodology for assigning LTS values to network edges is based on commonly tagged attributes of OSM ways. See more info about LTS in the original documentation of R5 from Conveyal at https://docs.conveyal.com/learn-more/traffic-stress. In summary:

- **LTS 1**: Tolerable for children. This includes low-speed, low-volume streets, as well as those with separated bicycle facilities (such as parking-protected lanes or cycle tracks).
- **LTS 2**: Tolerable for the mainstream adult population. This includes streets where cyclists have dedicated lanes and only have to interact with traffic at formal crossing.
- **LTS 3**: Tolerable for "enthused and confident" cyclists. This includes streets which may involve close proximity to moderate- or high-speed vehicular traffic.
- **LTS 4**: Tolerable for only "strong and fearless" cyclists. This includes streets where cyclists are required to mix with moderate- to high-speed vehicular traffic.

For advanced users, you can provide custom LTS values by adding a tag <key = "lts> to the `osm.pbf` file

Routing algorithm:

The travel_time_matrix function uses an R5-specific extension to the RAPTOR routing algorithm (see Conway et al., 2017). This RAPTOR extension uses a systematic sample of one departure per minute over the time window set by the user in the 'time_window' parameter. A detailed description of base RAPTOR can be found in Delling et al (2015).

- Conway, M. W., Byrd, A., & van der Linden, M. (2017). Evidence-based transit and land use sketch planning using interactive accessibility methods on combined schedule and headway-based networks. Transportation Research Record, 2653(1), 45-53.
- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604.

Datetime parsing

`r5r` ignores the timezone attribute of datetime objects when parsing dates and times, using the study area's timezone instead. For example, let's say you are running some calculations using Rio de Janeiro, Brazil, as your study area. The datetime `as.POSIXct("13-05-2019 14:00:00",format = "%d-%m-%Y %H:%M:%S")` will be parsed as May 13th, 2019, 14:00h in Rio's local time, as expected. But `as.POSIXct("13-05-2019 14:00:00",format = "%d-%m-%Y %H:%M:%S",tz = "Europe/Paris")` will also be parsed as the exact same date and time in Rio's local time, perhaps surprisingly, ignoring the timezone attribute.

See Also

Other routing: `accessibility()`, `detailed_itineraries()`

Examples

```
if (interactive()) {
library(r5r)

# build transport network
data_path <- system.file("extdata/spo", package = "r5r")
r5r_core <- setup_r5(data_path = data_path, temp_dir = TRUE)

# load origin/destination points
points <- read.csv(file.path(data_path, "spo_hexgrid.csv"))[1:5,]

departure_datetime <- as.POSIXct("13-05-2019 14:00:00", format = "%d-%m-%Y %H:%M:%S")

# estimate travel time matrix
ttm <- travel_time_matrix(r5r_core,
                          origins = points,
                          destinations = points,
                          mode = c("WALK", "TRANSIT"),
                          departure_datetime = departure_datetime,
                          max_walk_dist = Inf,
```

```
                            max_trip_duration = 120L)

    stop_r5(r5r_core)

    }
```

Index