Package: r5r (via r-universe)

October 5, 2026

Type Package

Title Rapid Realistic Routing with 'R5'

Version 2.4.0

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 and monetary cost 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. This version of r5r depends on R5 v7.1.

License MIT + file LICENSE

URL https://github.com/ipeaGIT/r5r, https://ipeagit.github.io/r5r/

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

Depends R (>= 3.6)

Imports checkmate, cli, data.table, isoband, jsonlite, lifecycle, methods, purrr, rJava (>= 0.9-10), rlang, sf (>= 1.0-12), sfheaders, utils, zip

Suggests accessibility, covr, dplyr, fs, ggplot2 (>= 3.3.1), gtfstools, h3jsr (>= 1.3.0), interp, knitr, mapview, parallel, patchwork, rJavaEnv, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

SystemRequirements Java JDK (>= 21.0)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Marcus Saraiva [aut] (ORCID: <https://orcid.org/0000-0001-6218-2338>), Rafael H. M. Pereira [aut, cre] (ORCID: <https://orcid.org/0000-0003-2125-7465>), Daniel Herszenhut [aut] (ORCID: <https://orcid.org/0000-0001-8066-1105>), Alex Magnus [aut], Matthew Wigginton Bhagat-Conway [aut] (ORCID: <https://orcid.org/0000-0002-1210-2982>), Carlos Kaue Vieira Braga [ctb] (ORCID: <https://orcid.org/0000-0002-6104-7297>), Luyu Liu [ctb] (ORCID: <https://orcid.org/0000-0002-6684-5570>), Egor Kotov [ctb] (ORCID: <https://orcid.org/0000-0001-6690-5345>), Daniel Snow [ctb], Ipea - Institute for Applied Economic Research [cph, fnd], Department of Geography & Planning, University of Toronto [fnd]

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

Config/pak/sysreqs default-jdk

Repository https://cran.r-universe.dev

Date/Publication 2026-05-20 08:39:15 UTC

RemoteUrl https://github.com/cran/r5r

RemoteRef HEAD

RemoteSha 84a4b1cf1beee024292d79e397f19899233480b8

Contents

accessibility 3
arrival_travel_time_matrix 10
build_network 16
check_transit_availability 17
detailed_itineraries 19
download_r5 25
expanded_travel_time_matrix 26
find_snap 31
get_gtfs_errors 33
isochrone 33
pareto_frontier 39
r5r_cache 44
r5r_sitrep 45
read_fare_structure 46
setup_fare_structure 46

setup_r5 48
stop_r5 50
street_network_bbox 51
street_network_to_sf 52
transit_network_to_sf 53
travel_time_matrix 54
write_fare_structure 60

Index **62**

`accessibility` *Calculate access to opportunities*

Description

Fast computation of access to opportunities given a selected decay function.

Usage

```
accessibility(
  r5r_network,
  r5r_core = deprecated(),
  origins,
  destinations,
  opportunities_colnames = "opportunities",
  mode = "WALK",
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  time_window = 10L,
  percentiles = 50L,
  decay_function = "step",
  cutoffs = NULL,
  decay_value = NULL,
  max_walk_time = Inf,
  max_bike_time = Inf,
  max_car_time = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  fare_structure = NULL,
  max_fare = Inf,
  new_carspeeds = NULL,
  carspeed_scale = 1,
  new_lts = NULL,
  draws_per_minute = 5L,
  n_threads = Inf,
```

```
  verbose = FALSE,
  progress = FALSE,
  output_dir = NULL
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
origins, destinations	Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.
opportunities_colnames	A character vector. The names of the columns in the destinations input that tells the number of opportunities in each location. Several different column names can be passed, in which case the accessibility to each kind of opportunity will be calculated.
mode	A character vector. The transport modes allowed for access, transfer and vehicle legs of the trips. Defaults to WALK. Please see details for other options.
mode_egress	A character vector. The transport mode used after egress from the last public transport. It can be either WALK, BICYCLE or CAR. Defaults to WALK. Ignored when public transport is not used.
departure_datetime	A POSIXct object. Please note that the departure time only influences public transport legs. When working with public transport networks, please check the calendar.txt within your GTFS feeds for valid dates. Please see details for further information on how datetimes are parsed.
time_window	An integer. The time window in minutes for which r5r will calculate multiple travel time matrices departing each minute. Defaults to 10 minutes. By default, the function returns the result based on median travel times, but the user can set the percentiles parameter to extract more results. Please read the time window vignette for more details on its usage vignette("time_window", package = "r5r")
percentiles	An integer vector (max length of 5). Specifies the percentile to use when returning accessibility estimates within the given time window. Please note that this parameter is applied to the travel time estimates that generate the accessibility results, and not to the accessibility distribution itself (i.e. if the 25th percentile is specified, the accessibility is calculated from the 25th percentile travel time, which may or may not be equal to the 25th percentile of the accessibility distribution itself). Defaults to 50, returning the accessibility calculated from the median travel time. If a vector with length bigger than 1 is passed, the output contains an additional column that specifies the percentile of each accessibility estimate. Due to upstream restrictions, only 5 percentiles can be specified at a time. For more details, please see R5 documentation at https://docs.conveyal.com/analysis/methodology#accounting-for-variability.

`decay_function` A string. Which decay function to use when calculating accessibility. One of `step`, `exponential`, `fixed_exponential`, `linear` or `logistic`. Defaults to `step`, which is equivalent to a cumulative opportunities measure. Please see the details to understand how each alternative works and how they relate to the `cutoffs` and `decay_value` parameters.

`cutoffs` A numeric vector (maximum length of 12). This parameter has different effects for each decay function: it indicates the cutoff times in minutes when calculating cumulative opportunities accessibility with the `step` function, 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` A number. Extra parameter to be passed to the selected `decay_function`. Has no effects when `decay_function` is either `step` or `exponential`.

`max_walk_time` An integer. The maximum walking time (in minutes) to access and egress the transit network, to make transfers within the network or to complete walk-only trips. Defaults to no restrictions (numeric value of `Inf`), as long as `max_trip_duration` is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set `max_walk_time` to 15, you could get trips with an up to 15 minutes walk leg to reach transit and another up to 15 minutes walk leg to reach the destination after leaving transit. In walk-only trips, whenever `max_walk_time` differs from `max_trip_duration`, the lowest value is considered.

`max_bike_time` An integer. The maximum cycling time (in minutes) to access and egress the transit network, to make transfers within the network or to complete bicycle-only trips. Defaults to no restrictions (numeric value of `Inf`), as long as `max_trip_duration` is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set `max_bike_time` to 15, you could get trips with an up to 15 minutes cycle leg to reach transit and another up to 15 minutes cycle leg to reach the destination after leaving transit. In bicycle-only trips, whenever `max_bike_time` differs from `max_trip_duration`, the lowest value is considered.

`max_car_time` An integer. The maximum driving time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as `max_trip_duration` is respected. The max time is considered separately for each leg (e.g. if you set `max_car_time` to 15 minutes, you could potentially drive up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to `Inf`, no limit.

`max_trip_duration`
An integer. The maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

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

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

`max_rides` An integer. The maximum number of public transport rides allowed in the same trip. Defaults to 3.

`max_lts` An integer 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. Please see details for more information.

`fare_structure` A fare structure object, following the convention set in `setup_fare_structure()`. This object describes how transit fares should be calculated. Please see the fare structure vignette to understand how this object is structured: `vignette("fare_structure", package = "r5r")`.

`max_fare` A number. The maximum value that trips can cost when calculating the fastest journey between each origin and destination pair.

`new_carspeeds` A `data.frame` specifying the new car speed for each OSM edge id. This table must contain columns `osm_id`, `max_speed` and `speed_type`. The `"speed_type"` column is of class character and it indicates whether the values in `"max_speed"` should be interpreted as percentages of original speeds (`"scale"`) or as absolute speeds (`"km/h"`). Alternatively, the `new_carspeeds` parameter can receive an `sf data.frame` with POLYGON geometry that indicates the new car speed for all the roads that fall within each polygon. In this case, the table must contain the columns `poly_id` with a unique id for each polygon, `scale` with the new speed scaling factors and `priority`, which is a number ranking which polygon should be considered in case of overlapping polygons. See more into in the `link to congestion vignette`.

`carspeed_scale` Numeric. The default car speed to use for road segments not specified in `new_carspeeds`. By default, it is `NULL` and the speeds of the unlisted roads are kept unchanged.

`new_lts` A `data.frame` specifying the new LTS levels for each OSM edge id. The table must contain columns `osm_id` and `lts`. Alternatively, the `new_lts` parameter can receive an `sf data.frame` with LINESTRING geometry. R5 will then find the nearest road for each LINESTRING and update its LTS value accordingly.

`draws_per_minute`
An integer. The number of Monte Carlo draws to perform per time window minute when calculating travel time matrices and when estimating accessibility. Defaults to 5. This would mean 300 draws in a 60-minute time window, for example. This parameter only affects the results when the GTFS feeds contain a `frequencies.txt` table. If the GTFS feed does not have a frequency table, r5r still allows for multiple runs over the set `time_window` but in a deterministic way.

`n_threads` An integer. The number of threads to use when running the router in parallel. Defaults to use all available threads (Inf).

`verbose` A logical. Whether to show `R5` informative messages when running the function. Defaults to `FALSE` (please note that in such case `R5` error messages are still shown). Setting `verbose` to `TRUE` shows detailed output, which can be useful for debugging issues not caught by `r5r`.

`progress` A logical. Whether to show a progress counter when running the router. Defaults to `FALSE`. 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.

`output_dir` Either `NULL` or a path to an existing directory. When not `NULL` (the default), the function will write one `.csv` file with the results for each origin in the specified

directory. In such case, the function returns the path specified in this parameter. This parameter is particularly useful when running on memory-constrained settings because writing the results directly to disk prevents `r5r` from loading them to RAM memory.

Value

A `data.table` with accessibility estimates for all origin points. This `data.table` contain columns listing the origin id, the type of opportunities to which accessibility was calculated, the travel time percentile considered in the accessibility estimate and the specified cutoff values (except in when `decay_function` is `fixed_exponential`, in which case the `cutoff` parameter is not used). If `output_dir` is not `NULL`, the function returns the path specified in that parameter, in which the `.csv` files containing the results are saved.

Decay functions

`R5` allows one to use different decay functions when calculating accessibility. Please see the original `R5` documentation from Conveyal for more information on each one one (https://docs.conveyal.com/learn-more/decay-functions). A summary of each available option, as well as the value passed to `decay_function` to use it (inside parentheses) are listed below:

- **Step**, also known as cumulative opportunities (`"step"`):
  a binary decay function used to find the sum of available opportunities within a specific travel time cutoff.
- **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 that 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 `exp(-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`.

Level of Traffic Stress (LTS)

When cycling is enabled in `R5` (by passing the value `BIKE` to either `mode` or `mode_egress`), 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 and 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. Routing also reverts to walking if the street segment is tagged as non-bikable in OSM (e.g. a staircase), independently of the specified max LTS.

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 only for "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.

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.

Routing algorithm

The travel_time_matrix(), expanded_travel_time_matrix(), arrival_travel_time_matrix() and accessibility() functions use 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). However, whenever the user includes transit fares inputs to these functions, they automatically switch to use an R5-specific extension to the McRAPTOR routing algorithm.

- 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. doi:10.3141/265306
- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604. doi:10.1287/trsc.2014.0534

Examples

```
library(r5r)

data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)
points <- read.csv(file.path(data_path, "poa_hexgrid.csv"))[1:500, ]

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

access <- accessibility(
  r5r_network ,
  origins = points,
  destinations = points,
  opportunities_colnames = "schools",
  mode = "WALK",
  departure_datetime = departure_datetime,
  decay_function = "step",
  cutoffs = 30,
  max_trip_duration = 30
)
head(access)

# using a different decay function
access <- accessibility(
  r5r_network ,
  origins = points,
  destinations = points,
  opportunities_colnames = "schools",
  mode = "WALK",
  departure_datetime = departure_datetime,
  decay_function = "logistic",
  cutoffs = 30,
```

```
  decay_value = 1,
  max_trip_duration = 30
)
head(access)

# using several cutoff values
access <- accessibility(
  r5r_network ,
  origins = points,
  destinations = points,
  opportunities_colnames = "schools",
  mode = "WALK",
  departure_datetime = departure_datetime,
  decay_function = "step",
  cutoffs = c(15, 30),
  max_trip_duration = 30
)
head(access)

# calculating access to different types of opportunities
access <- accessibility(
  r5r_network ,
  origins = points,
  destinations = points,
  opportunities_colnames = c("schools", "healthcare"),
  mode = "WALK",
  departure_datetime = departure_datetime,
  decay_function = "step",
  cutoffs = 30,
  max_trip_duration = 30
)
head(access)

stop_r5(r5r_network )
```

`arrival_travel_time_matrix` *Calculate travel time matrix between origin destination pairs considering a time of arrival*

Description

Computation of travel time estimates between one or multiple origin destination pairs considering a time of arrival. This function considers a time of arrival set by the user. The function returns the travel time of the trip with the latest departure time that arrives before the arrival time set by the user. If you want to calculate travel times considering a departure time, have a' look at the `travel_time_matrix()` function. This function is a wrapper around `expanded_travel_time_matrix()`. On one hand, this means this the output of this function has more columns (more info) compared the output of `travel_time_matrix()`. On the other hand, this function can be very memory intensive if the user allows for really long max trip duration.

Usage

```
arrival_travel_time_matrix(
  r5r_network,
  r5r_core = deprecated(),
  origins,
  destinations,
  mode = "WALK",
  mode_egress = "WALK",
  arrival_datetime = Sys.time(),
  breakdown = FALSE,
  max_walk_time = Inf,
  max_bike_time = Inf,
  max_car_time = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  new_carspeeds = NULL,
  carspeed_scale = 1,
  new_lts = NULL,
  draws_per_minute = 5L,
  n_threads = Inf,
  verbose = FALSE,
  progress = FALSE,
  output_dir = NULL
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
origins, destinations	Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.
mode	A character vector. The transport modes allowed for access, transfer and vehicle legs of the trips. Defaults to WALK. Please see details for other options.
mode_egress	A character vector. The transport mode used after egress from the last public transport. It can be either WALK, BICYCLE or CAR. Defaults to WALK. Ignored when public transport is not used.
arrival_datetime	A POSIXct object.
breakdown	A logical. Whether to include detailed information about each trip in the output. If FALSE (the default), the output lists the total time between each origin-destination pair and the routes used to complete the trip for each minute of the specified time window. If TRUE, the output includes the total access, waiting,

in-vehicle and transfer time of each trip. Please note that setting this parameter to TRUE makes the function significantly slower.

max_walk_time An integer. The maximum walking time (in minutes) to access and egress the transit network, to make transfers within the network or to complete walk-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set max_walk_time to 15, you could get trips with an up to 15 minutes walk leg to reach transit and another up to 15 minutes walk leg to reach the destination after leaving transit. In walk-only trips, whenever max_walk_time differs from max_trip_duration, the lowest value is considered.

max_bike_time An integer. The maximum cycling time (in minutes) to access and egress the transit network, to make transfers within the network or to complete bicycle-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set max_bike_time to 15, you could get trips with an up to 15 minutes cycle leg to reach transit and another up to 15 minutes cycle leg to reach the destination after leaving transit. In bicycle-only trips, whenever max_bike_time differs from max_trip_duration, the lowest value is considered.

max_car_time An integer. The maximum driving time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as max_trip_duration is respected. The max time is considered separately for each leg (e.g. if you set max_car_time to 15 minutes, you could potentially drive up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to Inf, no limit.

max_trip_duration
An integer. The maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

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

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

max_rides An integer. The maximum number of public transport rides allowed in the same trip. Defaults to 3.

max_lts An integer 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. Please see details for more information.

new_carspeeds A data.frame specifying the new car speed for each OSM edge id. This table must contain columns osm_id, max_speed and speed_type. The "speed_type" column is of class character and it indicates whether the values in "max_speed" should be interpreted as percentages of original speeds ("scale") or as absolute speeds ("km/h"). Alternatively, the new_carspeeds parameter can receive an sf data.frame with POLYGON geometry that indicates the new car speed for all the roads that fall within each polygon. In this case, the table must contain the columns poly_id with a unique id for each polygon, scale with the new speed scaling factors and priority, which is a number ranking which polygon

should be considered in case of overlapping polygons. See more into in the link to congestion vignette.

carspeed_scale Numeric. The default car speed to use for road segments not specified in new_carspeeds. By default, it is NULL and the speeds of the unlisted roads are kept unchanged.

new_lts A data.frame specifying the new LTS levels for each OSM edge id. The table must contain columns osm_id and lts. Alternatively, the new_lts parameter can receive an sf data.frame with LINESTRING geometry. R5 will then find the nearest road for each LINESTRING and update its LTS value accordingly.

draws_per_minute
An integer. The number of Monte Carlo draws to perform per time window minute when calculating travel time matrices and when estimating accessibility. Defaults to 5. This would mean 300 draws in a 60-minute time window, for example. This parameter only affects the results when the GTFS feeds contain a frequencies.txt table. If the GTFS feed does not have a frequency table, r5r still allows for multiple runs over the set time_window but in a deterministic way.

n_threads An integer. The number of threads to use when running the router in parallel. Defaults to use all available threads (Inf).

verbose A logical. Whether to show R5 informative messages when running the function. Defaults to FALSE (please note that in such case R5 error messages are still shown). Setting verbose to TRUE shows detailed output, which can be useful for debugging issues not caught by r5r.

progress A logical. Whether to show a progress counter when running the router. Defaults to FALSE. 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.

output_dir Either NULL or a path to an existing directory. When not NULL (the default), the function will write one .csv file with the results for each origin in the specified directory. In such case, the function returns the path specified in this parameter. This parameter is particularly useful when running on memory-constrained settings because writing the results directly to disk prevents r5r from loading them to RAM memory.

Value

A data.table with travel time estimates (in minutes) and the routes used in each trip between origin and destination pairs, for each minute of the specified time window. Each set of origin, destination and departure minute can appear up to N times, where N is the number of Monte Carlo draws specified in the function arguments (please note that this only applies when the GTFS feeds that describe the transit network include a frequencies table, otherwise only a single draw is performed). A pair is completely absent from the final output if no trips could be completed in any of the minutes of the time window. If for a single pair trips could be completed in some of the minutes of the time window, but not for all of them, the minutes in which trips couldn't be completed will have NA travel time and routes used. If output_dir is not NULL, the function returns the path specified in that parameter, in which the .csv files containing the results are saved.

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.

Level of Traffic Stress (LTS)

When cycling is enabled in R5 (by passing the value BIKE to either mode or mode_egress), 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 and 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. Routing also reverts to walking if the street segment is tagged as non-bikable in OSM (e.g. a staircase), independently of the specified max LTS.

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 only for "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.

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.

Routing algorithm

The travel_time_matrix(), expanded_travel_time_matrix(), arrival_travel_time_matrix() and accessibility() functions use 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). However, whenever the user includes

transit fares inputs to these functions, they automatically switch to use an R5-specific extension to the McRAPTOR routing algorithm.

- 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. doi:10.3141/265306
- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604. doi:10.1287/trsc.2014.0534

See Also

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

Examples

```
library(r5r)

# build transport network
data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path )

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

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

# by default only returns the total time between each pair in each minute of
# the specified time window
arrival_ttm <- arrival_travel_time_matrix(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  arrival_datetime = arrival_datetime,
  max_trip_duration = 60
)

head(arrival_ttm)

# when breakdown = TRUE the output contains much more information
arrival_ttm2 <- arrival_travel_time_matrix(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  arrival_datetime = arrival_datetime,
  max_trip_duration = 60,
  breakdown = TRUE
```

```
)

head(arrival_ttm2)

stop_r5(r5r_network)
```

`build_network` *Build a transport network used for routing in R5*

Description

Builds a multimodal transport network used for routing in R5, combining multiple data inputs present in the directory where the network should be saved to. The directory must contain only one street network file (in `.osm.pbf` format). It may optionally contain one or more public transport GTFS feeds (in `.zip` format), when used for public transport routing, and a `.tif` file describing the elevation profile of the study area. If there is more than one GTFS feed in the directory, all feeds are automatically merged. If there is already a `'network.dat'` file in the directory, the function will simply read it and load it to memory (unless specified not to do so).

Usage

```
build_network(
  data_path,
  verbose = FALSE,
  temp_dir = FALSE,
  elevation = "TOBLER",
  overwrite = FALSE
)
```

Arguments

`data_path`	A string pointing to the directory where data inputs are stored and where the built `network.dat` will be saved.
`verbose`	A logical. Whether to show R5 informative messages when running the function. Defaults to `FALSE` (please note that in such case R5 error messages are still shown). Setting `verbose` to `TRUE` shows detailed output, which can be useful for debugging issues not caught by `r5r`.
`temp_dir`	A logical. Whether the `network.dat` file should be saved to a temporary directory. Defaults to `FALSE`.
`elevation`	A string. The name of the impedance function to be used to calculate impedance for walking and cycling based on street slopes. Available options include `TOBLER` (Default) and `MINETTI`, or `NONE` to ignore elevation. R5 loads elevation data from `.tif` files saved inside the `data_path` directory. Elevation raster must be in WGS 84 (EPSG:4326) coordinate reference system. See more info in the Details section below.
`overwrite`	A logical. Whether to overwrite an existing `network.dat` or to use a cached file. Defaults to `FALSE` (i.e. use a cached network).

Value

A r5r_network object representing the built network to connect with R5 routing engine.

Elevation

More information about the TOBLER and MINETTI options to calculate the effects of elevation on travel times can be found in the references below:

- Campbell, M. J., et al (2019). Using crowdsourced fitness tracker data to model the relationship between slope and travel rates. Applied geography, 106, 93-107. doi:10.1016/j.apgeog.2019.03.008.
- Minetti, A. E., et al (2002). Energy cost of walking and running at extreme uphill and downhill slopes. Journal of applied physiology. doi:10.1152/japplphysiol.01177.2001.
- Tobler, W. (1993). Three presentations on geographical analysis and modeling: Non-isotropic geographic modeling speculations on the geometry of geography global spatial analysis. Technical Report. National center for geographic information and analysis. 93 (1). https://escholarship.org/uc/item/05r820mz.

See Also

Other Build network: download_r5(), setup_r5()

Examples

```
library(r5r)

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

r5r_network <- build_network(data_path)
```

---

check_transit_availability

Check transit service availability by date

Description

This function checks the number and proportion of public transport services from the GTFS feeds in a r5r_network that are active on specified dates. This is useful to verify that the selected departure dates for routing analysis are valid and have adequate service levels. When routing with public transport, it is crucial to use a departure date where services are operational, as indicated in the GTFS calendar.txt file.

Usage

```
check_transit_availability(
  r5r_network,
  r5r_core = deprecated(),
  dates = NULL,
  start_date = NULL,
  end_date = NULL
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
dates	A vector of specific dates to be checked. Can be character strings in #' "YYYY-MM-DD" format, or objects of class Date. This argument cannot be used with start_date or end_date.
start_date	The start date for a continuous date range. Must be a single character string in "YYYY-MM-DD" format or a Date object. Must be used with end_date.
end_date	The end date for a continuous date range. Must be a single character string in "YYYY-MM-DD" format or a Date object. Must be used with start_date.

Details

You can specify the dates to check in two ways:

- Using the dates argument to provide a vector of specific dates.
- Using the start_date and end_date arguments to provide a continuous date range.

You must use one of these two methods, but not both in the same function call.

Value

A data.table with four columns: date, total_services, active_services, and pct_active (the proportion of active services).

Examples

```
library(r5r)
data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)

# Example 1: Check a vector of specific dates
# Let's check a regular weekday and a Sunday, where service may differ.
dates_to_check <- c("2019-05-13", "2019-05-19")
availability1 <- check_transit_availability(r5r_network, dates = dates_to_check)
availability1

# Example 2: Check a continuous date range using start_date and end_date
```

```
availability2 <- check_transit_availability(
  r5r_network,
  start_date = "2019-01-01",
  end_date = "2019-12-31"
)
availability2[121:124,]

# plot availability over the year
library(ggplot2)
ggplot(availability2, aes(x = date, y = pct_active)) +
  geom_line() +
  geom_point() +
  theme_classic(base_size = 16)

stop_r5(r5r_network)
```

`detailed_itineraries` *Detailed itineraries between origin-destination pairs*

Description

Returns detailed trip information between origin-destination pairs. The output includes the waiting and moving time in each trip leg, as well as some info such as the distance traveled, the routes used and the geometry of each leg. Please note that this function was originally conceptualized as a trip planning functionality, similar to other commercial and non-commercial APIs and apps (e.g. Moovit, Google's Directions API, OpenTripPlanning's PlannerResource API). Thus, it consumes much more time and memory than the other (more analytical) routing functions included in the package.

Usage

```
detailed_itineraries(
  r5r_network,
  r5r_core = deprecated(),
  origins,
  destinations,
  mode = "WALK",
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  time_window = 10L,
  suboptimal_minutes = 0L,
  max_walk_time = Inf,
  max_bike_time = Inf,
  max_car_time = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
```

```
  max_rides = 3,
  max_lts = 2,
  shortest_path = TRUE,
  all_to_all = FALSE,
  fare_structure = NULL,
  max_fare = Inf,
  new_carspeeds = NULL,
  carspeed_scale = 1,
  new_lts = NULL,
  n_threads = Inf,
  verbose = FALSE,
  progress = FALSE,
  drop_geometry = FALSE,
  osm_link_ids = FALSE,
  output_dir = NULL
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
origins, destinations	Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.
mode	A character vector. The transport modes allowed for access, transfer and vehicle legs of the trips. Defaults to WALK. Please see details for other options.
mode_egress	A character vector. The transport mode used after egress from the last public transport. It can be either WALK, BICYCLE or CAR. Defaults to WALK. Ignored when public transport is not used.
departure_datetime	A POSIXct object. Please note that the departure time only influences public transport legs. When working with public transport networks, please check the calendar.txt within your GTFS feeds for valid dates. Please see details for further information on how datetimes are parsed.
time_window	An integer. The time window in minutes for which r5r will calculate multiple itineraries departing each minute. Defaults to 10 minutes. If the same sequence of routes appear in different minutes of the time window, only the fastest of them will be kept in the output. This happens because the result is not aggregated by percentile, as opposed to other routing functions in the package. Because of that, the output may contain trips departing after the specified departure_datetime, but still within the time window. Please read the time window vignette for more details on how this argument affects the results of each routing function: vignette("time_window", package = "r5r").
suboptimal_minutes	A number. The difference in minutes that each non-optimal RAPTOR branch can have from the optimal branch without being disregarded by the routing al-

gorithm. If, for example, users set `suboptimal_minutes = 10`, the routing algorithm will consider sub-optimal routes that arrive up to 10 minutes after the arrival of the optimal one. This argument emulates the real-life behaviour that makes people want to take a path that is technically not optimal in terms of travel time, for example, for some practical reasons (e.g. mode preference, safety, etc). In practice, the higher this value, the more itineraries will be returned in the final result.

`max_walk_time` An integer. The maximum walking time (in minutes) to access and egress the transit network, to make transfers within the network or to complete walk-only trips. Defaults to no restrictions (numeric value of `Inf`), as long as `max_trip_duration` is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set `max_walk_time` to 15, you could get trips with an up to 15 minutes walk leg to reach transit and another up to 15 minutes walk leg to reach the destination after leaving transit. In walk-only trips, whenever `max_walk_time` differs from `max_trip_duration`, the lowest value is considered.

`max_bike_time` An integer. The maximum cycling time (in minutes) to access and egress the transit network, to make transfers within the network or to complete bicycle-only trips. Defaults to no restrictions (numeric value of `Inf`), as long as `max_trip_duration` is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set `max_bike_time` to 15, you could get trips with an up to 15 minutes cycle leg to reach transit and another up to 15 minutes cycle leg to reach the destination after leaving transit. In bicycle-only trips, whenever `max_bike_time` differs from `max_trip_duration`, the lowest value is considered.

`max_car_time` An integer. The maximum driving time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as `max_trip_duration` is respected. The max time is considered separately for each leg (e.g. if you set `max_car_time` to 15 minutes, you could potentially drive up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to `Inf`, no limit.

`max_trip_duration`
An integer. The maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

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

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

`max_rides` An integer. The maximum number of public transport rides allowed in the same trip. Defaults to 3.

`max_lts` An integer 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. Please see details for more information.

`shortest_path` A logical. Whether the function should only return the fastest itinerary between each origin and destination pair (the default) or multiple alternatives.

`all_to_all` A logical. Whether to query routes between the 1st origin to the 1st destination, then the 2nd origin to the 2nd destination, and so on (`FALSE`, the default) or to query routes between all origins to all destinations (`TRUE`).

fare_structure — A fare structure object, following the convention set in setup_fare_structure(). This object describes how transit fares should be calculated. Please see the fare structure vignette to understand how this object is structured: vignette("fare_structure", package = "r5r").

max_fare — A number. The maximum value that trips can cost when calculating the fastest journey between each origin and destination pair.

new_carspeeds — A data.frame specifying the new car speed for each OSM edge id. This table must contain columns osm_id, max_speed and speed_type. The "speed_type" column is of class character and it indicates whether the values in "max_speed" should be interpreted as percentages of original speeds ("scale") or as absolute speeds ("km/h"). Alternatively, the new_carspeeds parameter can receive an sf data.frame with POLYGON geometry that indicates the new car speed for all the roads that fall within each polygon. In this case, the table must contain the columns poly_id with a unique id for each polygon, scale with the new speed scaling factors and priority, which is a number ranking which polygon should be considered in case of overlapping polygons. See more into in the link to congestion vignette.

carspeed_scale — Numeric. The default car speed to use for road segments not specified in new_carspeeds. By default, it is NULL and the speeds of the unlisted roads are kept unchanged.

new_lts — A data.frame specifying the new LTS levels for each OSM edge id. The table must contain columns osm_id and lts. Alternatively, the new_lts parameter can receive an sf data.frame with LINESTRING geometry. R5 will then find the nearest road for each LINESTRING and update its LTS value accordingly.

n_threads — An integer. The number of threads to use when running the router in parallel. Defaults to use all available threads (Inf).

verbose — A logical. Whether to show R5 informative messages when running the function. Defaults to FALSE (please note that in such case R5 error messages are still shown). Setting verbose to TRUE shows detailed output, which can be useful for debugging issues not caught by r5r.

progress — A logical. Whether to show a progress counter when running the router. Defaults to FALSE. 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 — A logical. Whether the output should include the geometry of each trip leg or not. The default value of FALSE keeps the geometry column in the result.

osm_link_ids — A logical. Whether the output should include additional columns: osm_id_list for the OSM ids of the road segments used along the trip geometry, and board_stop_id and alight_stop_id for the OSM ids of transit boarding and alighting stops. Defaults to FALSE.

Keep in mind that osm_id_list will contain an id even if the route only uses a small stretch of the road (e.g. 5m of a 600m street segment). For more precision, use edge_id_list, which returns the exact internal edge segments used in the trip. You can inspect these edge ids and their associated properties, including OSM ids, with street_network_to_sf().

output_dir Either NULL or a path to an existing directory. When not NULL (the default), the function will write one .csv file with the results for each origin in the specified directory. In such case, the function returns the path specified in this parameter. This parameter is particularly useful when running on memory-constrained settings because writing the results directly to disk prevents r5r from loading them to RAM memory.

Value

When drop_geometry is FALSE, the function outputs a LINESTRING sf with detailed information on the itineraries between the specified origins and destinations. When TRUE, the output is a data.table. All distances are in meters and travel times are in minutes. If output_dir is not NULL, the function returns the path specified in that parameter, in which the .csv files containing the results are saved.

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.

Level of Traffic Stress (LTS)

When cycling is enabled in R5 (by passing the value BIKE to either mode or mode_egress), 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 and 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. Routing also reverts to walking if the street segment is tagged as non-bikable in OSM (e.g. a staircase), independently of the specified max LTS.

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 only for "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.

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.

Routing algorithm

The `detailed_itineraries()` and `pareto_frontier()` functions use an R5-specific extension to the McRAPTOR routing algorithm. The implementation used in `detailed_itineraries()` allows the router to find paths that are optimal and less than optimal in terms of travel time, with some heuristics around multiple access modes, riding the same patterns, etc. The specific extension to McRAPTOR to do suboptimal path routing is not documented yet, but a detailed description of base McRAPTOR can be found in Delling et al (2015). The implementation used in `pareto_frontier()`, on the other hand, returns only the fastest trip within a given monetary cutoff, ignoring slower trips that cost the same. A detailed discussion on the algorithm can be found in Conway and Stewart (2019).

- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604. doi:10.1287/trsc.2014.0534
- Conway, M. W., & Stewart, A. F. (2019). Getting Charlie off the MTA: a multiobjective optimization method to account for cost constraints in public transit accessibility metrics. International Journal of Geographical Information Science, 33(9), 1759-1787. doi:10.1080/13658816.2019.1605075

See Also

Other routing: `arrival_travel_time_matrix()`, `expanded_travel_time_matrix()`, `pareto_frontier()`, `travel_time_matrix()`

Examples

```
library(r5r)

# build transport network
data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)

# 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"
)
```

```
det <- detailed_itineraries(
  r5r_network,
  origins = points[10,],
  destinations = points[12,],
  mode = c("WALK", "TRANSIT"),
  departure_datetime = departure_datetime,
  max_trip_duration = 60
)
head(det)

stop_r5(r5r_network)
```

`download_r5` *Download* `R5.jar`

Description

Downloads `R5.jar` and saves it locally, inside the package directory.

Usage

```
download_r5(
  version = NULL,
  quiet = FALSE,
  force_update = FALSE,
  temp_dir = FALSE
)
```

Arguments

`version`	A string. The version of R5 to be downloaded. When `NULL`, it defaults to the latest version.
`quiet`	A logical. Whether to show informative messages when downloading the file. Defaults to `FALSE`.
`force_update`	A logical. Whether to overwrite a previously downloaded `R5.jar` in the local directory. Defaults to `FALSE`.
`temp_dir`	A logical. Whether the file should be saved in a temporary directory. Defaults to `FALSE`.

Value

The path to the downloaded file.

See Also

Other Build network: `build_network()`, `setup_r5()`

Examples

```
library(r5r)

download_r5(temp_dir = TRUE)
```

`expanded_travel_time_matrix` *Calculate minute-by-minute travel times between origin destination pairs*

Description

Detailed computation of travel time estimates between one or multiple origin destination pairs. Results show the travel time of the fastest route alternative departing each minute within a specified time window. Please note this function can be very memory intensive for large data sets and time windows.

Usage

```
expanded_travel_time_matrix(
  r5r_network,
  r5r_core = deprecated(),
  origins,
  destinations,
  mode = "WALK",
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  time_window = 10L,
  breakdown = FALSE,
  max_walk_time = Inf,
  max_bike_time = Inf,
  max_car_time = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  new_carspeeds = NULL,
  carspeed_scale = 1,
  new_lts = NULL,
  draws_per_minute = 5L,
  n_threads = Inf,
  verbose = FALSE,
  progress = FALSE,
  output_dir = NULL
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
origins, destinations	Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.
mode	A character vector. The transport modes allowed for access, transfer and vehicle legs of the trips. Defaults to WALK. Please see details for other options.
mode_egress	A character vector. The transport mode used after egress from the last public transport. It can be either WALK, BICYCLE or CAR. Defaults to WALK. Ignored when public transport is not used.
departure_datetime	A POSIXct object. Please note that the departure time only influences public transport legs. When working with public transport networks, please check the calendar.txt within your GTFS feeds for valid dates. Please see details for further information on how datetimes are parsed.
time_window	An integer. The time window in minutes for which r5r will calculate multiple travel time matrices departing each minute. Defaults to 10 minutes. The function returns the result based on median travel times. Please read the time window vignette for more details on its usage vignette("time_window", package = "r5r")
breakdown	A logical. Whether to include detailed information about each trip in the output. If FALSE (the default), the output lists the total time between each origin-destination pair and the routes used to complete the trip for each minute of the specified time window. If TRUE, the output includes the total access, waiting, in-vehicle and transfer time of each trip. Please note that setting this parameter to TRUE makes the function significantly slower.
max_walk_time	An integer. The maximum walking time (in minutes) to access and egress the transit network, to make transfers within the network or to complete walk-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set max_walk_time to 15, you could get trips with an up to 15 minutes walk leg to reach transit and another up to 15 minutes walk leg to reach the destination after leaving transit. In walk-only trips, whenever max_walk_time differs from max_trip_duration, the lowest value is considered.
max_bike_time	An integer. The maximum cycling time (in minutes) to access and egress the transit network, to make transfers within the network or to complete bicycle-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set max_bike_time to 15, you could get trips with an up to 15 minutes cycle leg to reach transit and another up to 15 minutes cycle leg to reach the destination after leaving transit. In bicycle-only trips, whenever max_bike_time differs from max_trip_duration, the lowest value is considered.

max_car_time An integer. The maximum driving time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as max_trip_duration is respected. The max time is considered separately for each leg (e.g. if you set max_car_time to 15 minutes, you could potentially drive up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to Inf, no limit.

max_trip_duration An integer. The maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

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

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

max_rides An integer. The maximum number of public transport rides allowed in the same trip. Defaults to 3.

max_lts An integer 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. Please see details for more information.

new_carspeeds A data.frame specifying the new car speed for each OSM edge id. This table must contain columns osm_id, max_speed and speed_type. The "speed_type" column is of class character and it indicates whether the values in "max_speed" should be interpreted as percentages of original speeds ("scale") or as absolute speeds ("km/h"). Alternatively, the new_carspeeds parameter can receive an sf data.frame with POLYGON geometry that indicates the new car speed for all the roads that fall within each polygon. In this case, the table must contain the columns poly_id with a unique id for each polygon, scale with the new speed scaling factors and priority, which is a number ranking which polygon should be considered in case of overlapping polygons. See more into in the link to congestion vignette.

carspeed_scale Numeric. The default car speed to use for road segments not specified in new_carspeeds. By default, it is NULL and the speeds of the unlisted roads are kept unchanged.

new_lts A data.frame specifying the new LTS levels for each OSM edge id. The table must contain columns osm_id and lts. Alternatively, the new_lts parameter can receive an sf data.frame with LINESTRING geometry. R5 will then find the nearest road for each LINESTRING and update its LTS value accordingly.

draws_per_minute An integer. The number of Monte Carlo draws to perform per time window minute when calculating travel time matrices and when estimating accessibility. Defaults to 5. This would mean 300 draws in a 60-minute time window, for example. This parameter only affects the results when the GTFS feeds contain a frequencies.txt table. If the GTFS feed does not have a frequency table, r5r still allows for multiple runs over the set time_window but in a deterministic way.

n_threads An integer. The number of threads to use when running the router in parallel. Defaults to use all available threads (Inf).

verbose A logical. Whether to show R5 informative messages when running the function. Defaults to FALSE (please note that in such case R5 error messages are still

shown). Setting verbose to TRUE shows detailed output, which can be useful for debugging issues not caught by r5r.

progress — A logical. Whether to show a progress counter when running the router. Defaults to FALSE. 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.

output_dir — Either NULL or a path to an existing directory. When not NULL (the default), the function will write one .csv file with the results for each origin in the specified directory. In such case, the function returns the path specified in this parameter. This parameter is particularly useful when running on memory-constrained settings because writing the results directly to disk prevents r5r from loading them to RAM memory.

Value

A data.table with travel time estimates (in minutes) and the routes used in each trip between origin and destination pairs, for each minute of the specified time window. Each set of origin, destination and departure minute can appear up to N times, where N is the number of Monte Carlo draws specified in the function arguments (please note that this only applies when the GTFS feeds that describe the transit network include a frequencies table, otherwise only a single draw is performed). A pair is completely absent from the final output if no trips could be completed in any of the minutes of the time window. If for a single pair trips could be completed in some of the minutes of the time window, but not for all of them, the minutes in which trips couldn't be completed will have NA travel time and routes used. If output_dir is not NULL, the function returns the path specified in that parameter, in which the .csv files containing the results are saved.

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.

Level of Traffic Stress (LTS)

When cycling is enabled in R5 (by passing the value BIKE to either mode or mode_egress), 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 and 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. Routing also reverts to walking if the street segment is tagged as non-bikable in OSM (e.g. a staircase), independently of the specified max LTS.

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 only for "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.

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.

Routing algorithm

The `travel_time_matrix()`, `expanded_travel_time_matrix()`, `arrival_travel_time_matrix()` and `accessibility()` functions use 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). However, whenever the user includes transit fares inputs to these functions, they automatically switch to use an R5-specific extension to the McRAPTOR routing algorithm.

- 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. doi:10.3141/265306
- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604. doi:10.1287/trsc.2014.0534

See Also

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

Examples

```
library(r5r)

# build transport network
data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)

# load origin/destination points
```

```
points <- read.csv(file.path(data_path, "poa_points_of_interest.csv"))

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

# by default only returns the total time between each pair in each minute of
# the specified time window
ettm <- expanded_travel_time_matrix(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  time_window = 20,
  departure_datetime = departure_datetime,
  max_trip_duration = 60
)
head(ettm)

# when breakdown = TRUE the output contains much more information
ettm <- expanded_travel_time_matrix(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  time_window = 20,
  departure_datetime = departure_datetime,
  max_trip_duration = 60,
  breakdown = TRUE
)
head(ettm)

stop_r5(r5r_network)
```

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

Description

Finds the snapped location of points on R5 network. Snapping is an important step of the routing process, which is when the origins and destinations specified by the user are actually positioned on the network created by R5. The snapping process in R5 is composed of two rounds. First, it tries to snap the points within a radius of 300 meters from themselves. If the first round is unsuccessful, then R5 expands the search to the radius specified (by default 1.6km). If yet again it is unsuccessful, then the unsnapped points won't be used during the routing process. The snapped location of each point depends on the transport mode set by the user, because some network edges are not available to specific modes (e.g. a pedestrian-only street cannot be used to snap car trips).

Usage

```
find_snap(
  r5r_network,
  r5r_core = deprecated(),
  points,
  radius = 1600,
  mode = "WALK"
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
points	Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.
radius	Numeric. The maximum radius in meters within which to snap. Defaults to 1600m.
mode	A string. Which mode to consider when trying to snap the points to the network. Defaults to WALK, also allows BICYCLE and CAR.

Value

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

See Also

Other network functions: street_network_bbox(), street_network_to_sf(), transit_network_to_sf()

Examples

```
library(r5r)

path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path = path)
points <- read.csv(file.path(path, "poa_hexgrid.csv"))

snap_df <- find_snap(
  r5r_network,
  points = points,
  radius = 2000,
  mode = "WALK"
  )

stop_r5(r5r_network)
```

get_gtfs_errors *Get GTFS eventual errors encountered in network building*

Description

This returns a data frame of GTFS errors R5 encountered when building the network. You can call this with the network itself as the main parameter. If network build fails, you won't have a network object, so you can also call this with the `data_path` to where the network is stored.

Usage

```
get_gtfs_errors(r5r_network)
```

Arguments

`r5r_network`	the R5R network object, or a path to the location where the network is stored (useful if network build failed).

Value

A `data.frame`

See Also

Other support functions: `exists_tiff()`, `fileurl_from_metadata()`, `start_r5r_java()`, `stop_r5()`, `tempdir_unique()`, `travel_time_surface`, `validate_bad_osm_ids()`

Examples

```
library(r5r)

# directory with street network and gtfs files
data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)

get_gtfs_errors(r5r_network)
```

isochrone *Estimate isochrones from a given location*

Description

Fast computation of isochrones from a given location. The function can return either polygon-based or line-based isochrones. Polygon-based isochrones are generated as concave polygons based on the travel times from the trip origin to all nodes in the transport network. Meanwhile, line-based isochronesare based on travel times from each origin to the centroids of all segments in the transport network.

Usage

```
isochrone(
  r5r_network,
  r5r_core = deprecated(),
  origins,
  mode = "transit",
  mode_egress = "walk",
  cutoffs = c(0, 15, 30),
  zoom = 10,
  departure_datetime = Sys.time(),
  polygon_output = TRUE,
  time_window = 10L,
  max_walk_time = Inf,
  max_bike_time = Inf,
  max_car_time = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  draws_per_minute = 5L,
  percentiles = NULL,
  n_threads = Inf,
  verbose = FALSE,
  progress = TRUE,
  sample_size = deprecated()
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
origins	Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.
mode	A character vector. The transport modes allowed for access, transfer and vehicle legs of the trips. Defaults to WALK. Please see details for other options.
mode_egress	A character vector. The transport mode used after egress from the last public transport. It can be either WALK, BICYCLE or CAR. Defaults to WALK. Ignored when public transport is not used.
cutoffs	numeric vector. Number of minutes to define the time span of each Isochrone. Defaults to c(0, 15, 30).
zoom	Resolution of the travel time surface used to create isochrones, can be between 9 and 12. The default is 10 (which uses cells of 153 meters at the Equator). More detailed isochrones will result from larger numbers, at the expense of compute time. Specifically, a raster grid of travel times in the Web Mercator projection at

this zoom level is created, and the isochrones are interpolated from this grid. For more information on how the grid cells are defined, see the R5 documentation.

`departure_datetime`
: A POSIXct object. Please note that the departure time only influences public transport legs. When working with public transport networks, please check the `calendar.txt` within your GTFS feeds for valid dates. Please see details for further information on how datetimes are parsed.

`polygon_output`
: A Logical. If `TRUE`, the function outputs polygon-based isochrones (the default) based on travel times from each origin to a sample of a random sample nodes in the transport network (see parameter `sample_size`). If `FALSE`, the function outputs line-based isochrones based on travel times from each origin to the centroids of all segments in the transport network.

`time_window`
: An integer. The time window in minutes for which `r5r` will calculate multiple travel time matrices departing each minute. Defaults to 10 minutes. The function returns the result based on median travel times. Please read the time window vignette for more details on its usage `vignette("time_window", package = "r5r")`

`max_walk_time`
: An integer. The maximum walking time (in minutes) 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 time is considered separately for each leg (e.g. if you set `max_walk_time` to 15, you could potentially walk up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to `Inf`, no limit.

`max_bike_time`
: An integer. The maximum cycling time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as `max_trip_duration` is respected. The max time is considered separately for each leg (e.g. if you set `max_bike_time` to 15 minutes, you could potentially cycle up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to `Inf`, no limit.

`max_car_time`
: An integer. The maximum driving time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as `max_trip_duration` is respected. The max time is considered separately for each leg (e.g. if you set `max_car_time` to 15 minutes, you could potentially drive up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to `Inf`, no limit.

`max_trip_duration`
: An integer. The maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

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

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

`max_rides`
: An integer. The maximum number of public transport rides allowed in the same trip. Defaults to 3.

`max_lts`
: An integer 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. Please see details for more information.

draws_per_minute
: An integer. The number of Monte Carlo draws to perform per time window minute when calculating travel time matrices and when estimating accessibility. Defaults to 5. This would mean 300 draws in a 60-minute time window, for example. This parameter only affects the results when the GTFS feeds contain a `frequencies.txt` table. If the GTFS feed does not have a frequency table, r5r still allows for multiple runs over the set `time_window` but in a deterministic way.

percentiles
: An integer vector (max length of 5). Specifies the percentile to use when returning accessibility estimates within the given time window. Please note that this parameter is applied to the travel time estimates that generate the accessibility results, and not to the accessibility distribution itself (i.e. if the 25th percentile is specified, the accessibility is calculated from the 25th percentile travel time, which may or may not be equal to the 25th percentile of the accessibility distribution itself). Defaults to 50, returning the accessibility calculated from the median travel time. If a vector with length bigger than 1 is passed, the output contains an additional column that specifies the percentile of each accessibility estimate. Due to upstream restrictions, only 5 percentiles can be specified at a time. For more details, please see R5 documentation at https://docs.conveyal.com/analysis/methodology#accounting-for-variability.

n_threads
: An integer. The number of threads to use when running the router in parallel. Defaults to use all available threads (`Inf`).

verbose
: A logical. Whether to show R5 informative messages when running the function. Defaults to `FALSE` (please note that in such case R5 error messages are still shown). Setting `verbose` to `TRUE` shows detailed output, which can be useful for debugging issues not caught by `r5r`.

progress
: A logical. Whether to show a progress counter when running the router. Defaults to `FALSE`. 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.

sample_size
: deprecated, no longer has any effect.

Value

A `"sf" "data.frame"` for each isochrone of each origin.

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`.

Level of Traffic Stress (LTS)

When cycling is enabled in R5 (by passing the value BIKE to either mode or mode_egress), 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 and 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. Routing also reverts to walking if the street segment is tagged as non-bikable in OSM (e.g. a staircase), independently of the specified max LTS.

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 only for "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.

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.

Routing algorithm

The travel_time_matrix(), expanded_travel_time_matrix(), arrival_travel_time_matrix() and accessibility() functions use 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). However, whenever the user includes transit fares inputs to these functions, they automatically switch to use an R5-specific extension to the McRAPTOR routing algorithm.

- 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. doi:10.3141/265306
- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604. doi:10.1287/trsc.2014.0534

Examples

```
options(java.parameters = "-Xmx2G")
library(r5r)
library(ggplot2)

# build transport network
data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path = data_path)

# load origin/point of interest
points <- read.csv(file.path(data_path, "poa_points_of_interest.csv"))
origin <- points[2,]

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

# estimate polygon-based isochrone from origin
iso_poly <- isochrone(
  r5r_network,
  origins = origin,
  mode = "walk",
  polygon_output = TRUE,
  departure_datetime = departure_datetime,
  cutoffs = seq(0, 120, 30)
  )

head(iso_poly)


# estimate line-based isochrone from origin
iso_lines <- isochrone(
  r5r_network,
  origins = origin,
  mode = "walk",
  polygon_output = FALSE,
  departure_datetime = departure_datetime,
  cutoffs = seq(0, 100, 25)
  )

head(iso_lines)


# plot colors
colors <- c('#ffe0a5','#ffcb69','#ffa600','#ff7c43','#f95d6a',
            '#d45087','#a05195','#665191','#2f4b7c','#003f5c')

# polygons
ggplot() +
  geom_sf(data=iso_poly, aes(fill=factor(isochrone))) +
  scale_fill_manual(values = colors) +
```

```
  theme_minimal()

# lines
ggplot() +
  geom_sf(data=iso_lines, aes(color=factor(isochrone))) +
  scale_color_manual(values = colors) +
  theme_minimal()

stop_r5(r5r_network)
```

---

pareto_frontier *Calculate travel time and monetary cost Pareto frontier*

Description

Fast computation of travel time and monetary cost Pareto frontier between origin and destination pairs.

Usage

```
pareto_frontier(
  r5r_network,
  r5r_core = deprecated(),
  origins,
  destinations,
  mode = c("WALK", "TRANSIT"),
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  time_window = 10L,
  percentiles = 50L,
  max_walk_time = Inf,
  max_bike_time = Inf,
  max_car_time = Inf,
  max_trip_duration = 120L,
  fare_structure = NULL,
  fare_cutoffs = -1L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  n_threads = Inf,
  verbose = FALSE,
  progress = FALSE,
  output_dir = NULL
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
origins, destinations	Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.
mode	A character vector. The transport modes allowed for access, transfer and vehicle legs of the trips. Defaults to WALK. Please see details for other options.
mode_egress	A character vector. The transport mode used after egress from the last public transport. It can be either WALK, BICYCLE or CAR. Defaults to WALK. Ignored when public transport is not used.
departure_datetime	A POSIXct object. Please note that the departure time only influences public transport legs. When working with public transport networks, please check the calendar.txt within your GTFS feeds for valid dates. Please see details for further information on how datetimes are parsed.
time_window	An integer. The time window in minutes for which r5r will calculate multiple travel time matrices departing each minute. Defaults to 10 minutes. By default, the function returns the result based on median travel times, but the user can set the percentiles parameter to extract more results. Please read the time window vignette for more details on its usage vignette("time_window", package = "r5r")
percentiles	An integer vector (max length of 5). Specifies the percentile to use when returning travel time estimates within the given time window. Please note that this parameter is applied to the travel time estimates only (e.g. if the 25th percentile is specified, and the output between A and B is 15 minutes and 10 dollars, 25% of all trips cheaper than 10 dollars taken between these points are shorter than 15 minutes). Defaults to 50, returning the median travel time. If a vector with length bigger than 1 is passed, the output contains an additional column that specifies the percentile of each travel time and monetary cost combination. Due to upstream restrictions, only 5 percentiles can be specified at a time. For more details, please see R5 documentation at https://docs.conveyal.com/analysis/methodology#accounting-for-variability.
max_walk_time	An integer. The maximum walking time (in minutes) to access and egress the transit network, to make transfers within the network or to complete walk-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set max_walk_time to 15, you could get trips with an up to 15 minutes walk leg to reach transit and another up to 15 minutes walk leg to reach the destination after leaving transit. In walk-only trips, whenever max_walk_time differs from max_trip_duration, the lowest value is considered.
max_bike_time	An integer. The maximum cycling time (in minutes) to access and egress the transit network, to make transfers within the network or to complete bicycle-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration

is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set `max_bike_time` to 15, you could get trips with an up to 15 minutes cycle leg to reach transit and another up to 15 minutes cycle leg to reach the destination after leaving transit. In bicycle-only trips, whenever `max_bike_time` differs from `max_trip_duration`, the lowest value is considered.

`max_car_time` An integer. The maximum driving time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as `max_trip_duration` is respected. The max time is considered separately for each leg (e.g. if you set `max_car_time` to 15 minutes, you could potentially drive up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to `Inf`, no limit.

`max_trip_duration`
An integer. The maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

`fare_structure` A fare structure object, following the convention set in `setup_fare_structure()`. This object describes how transit fares should be calculated. Please see the fare structure vignette to understand how this object is structured: `vignette("fare_structure", package = "r5r")`.

`fare_cutoffs` A numeric vector. The monetary cutoffs that should be considered when calculating the Pareto frontier. Most of the time you'll want this parameter to be the combination of all possible fares listed in you `fare_structure`. Choosing a coarse distribution of cutoffs may result in many different trips falling within the same cutoff. For example, if you have two different routes in your GTFS, one costing $3 and the other costing $4, and you set this parameter to `5`, the output will tell you the fastest trips that costed up to $5, but you won't be able to identify which route was used to complete such trips. In this case, it would be more beneficial to set the parameter as `c(3, 4)` (you could also specify combinations of such values, such as 6, 7, 8 and so on, because a transit user could hypothetically benefit from making transfers between the available routes).

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

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

`max_rides` An integer. The maximum number of public transport rides allowed in the same trip. Defaults to 3.

`max_lts` An integer 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. Please see details for more information.

`n_threads` An integer. The number of threads to use when running the router in parallel. Defaults to use all available threads (Inf).

`verbose` A logical. Whether to show `R5` informative messages when running the function. Defaults to `FALSE` (please note that in such case `R5` error messages are still shown). Setting `verbose` to `TRUE` shows detailed output, which can be useful for debugging issues not caught by `r5r`.

`progress` A logical. Whether to show a progress counter when running the router. Defaults to `FALSE`. 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.

output_dir Either NULL or a path to an existing directory. When not NULL (the default), the function will write one .csv file with the results for each origin in the specified directory. In such case, the function returns the path specified in this parameter. This parameter is particularly useful when running on memory-constrained settings because writing the results directly to disk prevents r5r from loading them to RAM memory.

Value

A data.table with the travel time and monetary cost Pareto frontier between the specified origins and destinations. An additional column identifying the travel time percentile is present if more than one value was passed to percentiles. Origin and destination pairs whose trips couldn't be completed within the maximum travel time using less money than the specified monetary cutoffs are not returned in the data.table. If output_dir is not NULL, the function returns the path specified in that parameter, in which the .csv files containing the results are saved.

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.

Level of Traffic Stress (LTS)

When cycling is enabled in R5 (by passing the value BIKE to either mode or mode_egress), 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 and 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. Routing also reverts to walking if the street segment is tagged as non-bikable in OSM (e.g. a staircase), independently of the specified max LTS.

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 only for "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.

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.

Routing algorithm

The `detailed_itineraries()` and `pareto_frontier()` functions use an R5-specific extension to the McRAPTOR routing algorithm. The implementation used in `detailed_itineraries()` allows the router to find paths that are optimal and less than optimal in terms of travel time, with some heuristics around multiple access modes, riding the same patterns, etc. The specific extension to McRAPTOR to do suboptimal path routing is not documented yet, but a detailed description of base McRAPTOR can be found in Delling et al (2015). The implementation used in `pareto_frontier()`, on the other hand, returns only the fastest trip within a given monetary cutoff, ignoring slower trips that cost the same. A detailed discussion on the algorithm can be found in Conway and Stewart (2019).

- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604. doi:10.1287/trsc.2014.0534
- Conway, M. W., & Stewart, A. F. (2019). Getting Charlie off the MTA: a multiobjective optimization method to account for cost constraints in public transit accessibility metrics. International Journal of Geographical Information Science, 33(9), 1759-1787. doi:10.1080/13658816.2019.1605075

See Also

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

Examples

```
library(r5r)

# build transport network
data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path = data_path)

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

# load fare structure object
fare_structure_path <- system.file(
  "extdata/poa/fares/fares_poa.zip",
  package = "r5r"
)
fare_structure <- read_fare_structure(fare_structure_path)
```

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

pf <- pareto_frontier(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = departure_datetime,
  fare_structure = fare_structure,
  fare_cutoffs = c(4.5, 4.8, 9, 9.3, 9.6)
)
head(pf)

stop_r5(r5r_network)
```

`r5r_cache` *Manage cached files from the r5r package*

Description

Manage cached files from the r5r package

Usage

```
r5r_cache(list_files = TRUE, delete_file = NULL)
```

Arguments

`list_files`	Logical. Whether to print a message with the address of r5r JAR files cached locally. Defaults to `TRUE`.
`delete_file`	String. The file name (basename) of a JAR file cached locally that should be deleted. Defaults to `NULL`, so that no file is deleted. If `delete_file = "all"`, then all cached files are deleted.

Value

A message indicating which file exist and/or which ones have been deleted from local cache directory.

Examples

```
# download r5 JAR
r5r::download_r5()

# list all files cached
r5r_cache(list_files = TRUE)

# delete r5 JAR
r5r_cache(delete_file = 'r5-v7.0')
```

`r5r_sitrep` *Generate an r5r situation report to help debug errors*

Description

The function reports a list with the following information:

- The package version of `{r5r}` in use.
- The installed version of `R5.jar`.
- The Java version in use.
- The amount of memory set to Java through the `java.parameters` option.
- The user's Session Info.

Usage

```
r5r_sitrep()
```

Value

A `list` with information of the versions of the r5r package, Java and R5 Jar in use, the memory set to Java and user's Session Info.

Examples

```
r5r_sitrep()
```

read_fare_structure *Read a fare structure object from a file*

Description

Read a fare structure object from a file

Usage

```
read_fare_structure(file_path, encoding = "UTF-8")
```

Arguments

file_path	A path pointing to a fare structure with a .zip extension.
encoding	A string. Passed to data.table::fread(), defaults to "UTF-8". Other possible options are "unknown" and "Latin-1". Please note that this is not used to re-encode the input, but to enable handling encoded strings in their native encoding.

Value

A fare structure object.

See Also

Other fare structure: setup_fare_structure(), write_fare_structure()

Examples

```
path <- system.file("extdata/poa/fares/fares_poa.zip", package = "r5r")
fare_structure <- read_fare_structure(path)
```

setup_fare_structure *Setup a fare structure to calculate the monetary costs of trips*

Description

Creates a basic fare structure that describes how transit fares should be calculated in travel_time_matrix(), expanded_travel_time_matrix(), accessibility() and pareto_frontier(). This fare structure can be manually edited and adjusted to the existing rules in your study area, as long as they stick to some basic premises. Please see the fare-structure vignette for more information.

Usage

```
setup_fare_structure(
  r5r_network,
  r5r_core = deprecated(),
  base_fare,
  by = "MODE",
  debug_path = NULL,
  debug_info = NULL
)
```

Arguments

r5r_network	A routable transport network created with build_network().
r5r_core	The r5r_core argument is deprecated as of r5r v2.3.0. Please use the r5r_network argument instead.
base_fare	A numeric. A base value used to populate the fare structure.
by	A string. Describes how fare_types (a classification we created to assign fares to different routes) are distributed among routes. Possible values are MODE, AGENCY and GENERIC. MODE is used when the mode is what determines the price of a route (e.g. if all the buses of a given city cost $5). AGENCY is used when the agency that operates each route is what determines its price (i.e. when two different routes/modes operated by a single agency cost the same; note that you can also use AGENCY_NAME, if the agency_ids listed in your GTFS cannot be easily interpreted). GENERIC is used when all the routes cost the same. Please note that this classification can later be edited to better suit your needs (when, for example, two types of buses cost the same, but one offers discounts after riding the subway and the other one doesn't), but this parameter may save you some work.
debug_path	Either a path to a .csv file or NULL. When NULL (the default), fare debugging capabilities are disabled - i.e. there's no way to check if the fare calculation is correct. When a path is provided, r5r saves different itineraries and their respective fares to the specified file. How each itinerary is described is controlled by debug_info.
debug_info	Either a string (when debug_path is a path) or NULL (the default). Doesn't have any effect if debug_path is NULL. When a string, accepts the values MODE, ROUTE and MODE_ROUTE. These values dictates how itinerary information is written to the output. Let's suppose we have an itinerary composed by two transit legs: first a subway leg whose route_id is 001, and then a bus legs whose route_id is 007. If debug_info is MODE, then this itinerary will be described as SUBWAY|BUS. If ROUTE, as 001|007. If MODE_ROUTE, as SUBWAY 001|BUS 007. Please note that the final debug information will contain not only the itineraries that were in fact used in the itineraries returned in travel_time_matrix(), accessibility() and pareto_frontier(), but all the itineraries that R5 checked when calculating the routes. This imposes a performance penalty when tracking debug information (but has the positive effect of returning a larger sample of itineraries, which might help finding some implementation issues on the fare structure).

Value

A fare structure object.

See Also

Other fare structure: `read_fare_structure()`, `write_fare_structure()`

Examples

```
library(r5r)

data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)

fare_structure <- setup_fare_structure(r5r_network, base_fare = 5)

# to debug fare calculation
fare_structure <- setup_fare_structure(
  r5r_network,
  base_fare = 5,
  debug_path = "fare_debug.csv",
  debug_info = "MODE"
)

fare_structure$debug_settings

# debugging can be manually turned off by setting output_file to ""
fare_structure$debug_settings <- ""
```

---

setup_r5 *Create a transport network used for routing in R5 (deprecated)*

Description

[Deprecated]

`setup_r5()` was renamed to `build_network()` to create a more consistent API. `setup_r5()` **is being deprecated** after *r5r* v2.3.0 and will be **removed in a future release**. Please switch to `build_network()`.

Usage

```
setup_r5(
  data_path,
  verbose = FALSE,
  temp_dir = FALSE,
  elevation = "TOBLER",
  overwrite = FALSE
)
```

Arguments

data_path	A string pointing to the directory where data inputs are stored and where the built network.dat will be saved.
verbose	A logical. Whether to show R5 informative messages when running the function. Defaults to FALSE (please note that in such case R5 error messages are still shown). Setting verbose to TRUE shows detailed output, which can be useful for debugging issues not caught by r5r.
temp_dir	A logical. Whether the network.dat file should be saved to a temporary directory. Defaults to FALSE.
elevation	A string. The name of the impedance function to be used to calculate impedance for walking and cycling based on street slopes. Available options include TOBLER (Default) and MINETTI, or NONE to ignore elevation. R5 loads elevation data from .tif files saved inside the data_path directory. Elevation raster must be in WGS 84 (EPSG:4326) coordinate reference system. See more info in the Details section below.
overwrite	A logical. Whether to overwrite an existing network.dat or to use a cached file. Defaults to FALSE (i.e. use a cached network).

Value

A r5r_network object representing the built network to connect with R5 routing engine.

Elevation

More information about the TOBLER and MINETTI options to calculate the effects of elevation on travel times can be found in the references below:

- Campbell, M. J., et al (2019). Using crowdsourced fitness tracker data to model the relationship between slope and travel rates. Applied geography, 106, 93-107. doi:10.1016/j.apgeog.2019.03.008.
- Minetti, A. E., et al (2002). Energy cost of walking and running at extreme uphill and downhill slopes. Journal of applied physiology. doi:10.1152/japplphysiol.01177.2001.
- Tobler, W. (1993). Three presentations on geographical analysis and modeling: Non-isotropic geographic modeling speculations on the geometry of geography global spatial analysis. Technical Report. National center for geographic information and analysis. 93 (1). https://escholarship.org/uc/item/05r820mz.

See Also

Other Build network: build_network(), download_r5()

Examples

```
library(r5r)

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

```
# `setup_r5()` has been deprecated, please switch to `build_network()`
r5r_network <- build_network(data_path)
```

`stop_r5` *Stop running r5r network*

Description

Stops running r5r network

Usage

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

Arguments

`...`	`r5r_network` objects currently running. By default, if no r5r network is supplied all running networks are stopped.

Value

No return value, called for side effects.

See Also

Other support functions: `exists_tiff()`, `fileurl_from_metadata()`, `get_gtfs_errors()`, `start_r5r_java()`, `tempdir_unique()`, `travel_time_surface`, `validate_bad_osm_ids()`

Examples

```
library(r5r)

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

r5r_network <- build_network(path)

stop_r5(r5r_network)
```

`street_network_bbox` *Extract the geographic bounding box of the transport network*

Description

Extracts the geographic bounding box of the street network layer from a routable transport network built with `build_network()`). It is a fast and memory-efficient alternative to `sf::st_bbox(street_network_to_sf(r5r_ne`

Usage

```
street_network_bbox(
  r5r_network,
  output = c("polygon", "bbox", "vector"),
  r5r_core = deprecated()
)
```

Arguments

`r5r_network`	A routable transport network created with `build_network()`.
`output`	A character string specifying the desired output format. One of `"polygon"` (the default), `"bbox"`, or `"vector"`.
`r5r_core`	The `r5r_core` argument is deprecated as of r5r v2.3.0. Please use the `r5r_network` argument instead.

Value

By default (`output = "polygon"`), an `sf` object with a single `POLYGON` geometry. If `output = "bbox"`, an `sf bbox` object. If `output = "vector"`, a named numeric vector with `xmin`, `ymin`, `xmax`, `ymax` coordinates. All outputs use the WGS84 coordinate reference system (EPSG: 4326).

See Also

Other network functions: `find_snap()`, `street_network_to_sf()`, `transit_network_to_sf()`

Examples

```
library(r5r)
library(sf)

data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)

# Get the network's bounding box as an sf polygon (default)
poly <- street_network_bbox(r5r_network, output = "polygon")
plot(poly)

# Get an sf bbox object (order is xmin, ymin, xmax, ymax)
box <- street_network_bbox(r5r_network , output = "bbox")
```

```
box

# Get a simple named vector (order now also xmin, ymin, xmax, ymax)
vec <- street_network_bbox(r5r_network , output = "vector")
vec

stop_r5(r5r_network)
```

`street_network_to_sf` *Extract OpenStreetMap network in sf format*

Description

Extracts the OpenStreetMap network in `sf` format from a routable transport network built with `build_network()`).

Usage

```
street_network_to_sf(r5r_network, r5r_core = deprecated())
```

Arguments

`r5r_network`	A routable transport network created with `build_network()`.
`r5r_core`	The `r5r_core` argument is deprecated as of r5r v2.3.0. Please use the `r5r_network` argument instead.

Value

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

See Also

Other network functions: `find_snap()`, `street_network_bbox()`, `transit_network_to_sf()`

Examples

```
library(r5r)

# build transport network
path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(path)

# extract street network from r5r_network
street_net <- street_network_to_sf(r5r_network)

stop_r5(r5r_network)
```

`transit_network_to_sf` *Extract transit network in sf format*

Description

Extracts the transit network in `sf` format from a routable transport network built with `build_network()`).

Usage

```
transit_network_to_sf(r5r_network, r5r_core = deprecated())
```

Arguments

`r5r_network`	A routable transport network created with `build_network()`.
`r5r_core`	The `r5r_core` argument is deprecated as of r5r v2.3.0. Please use the `r5r_network` argument instead.

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 the same route is associated to more than one `shape_ids` in the GTFS feed used to create the transit network. Some transit stops might be returned with geometry `POINT EMPTY` (i.e. missing spatial coordinates). This may occur when a transit stop is not snapped to the road network, possibly because the GTFS feed used to create the transit network covers an area larger than the `.osm.pbf` input data.

See Also

Other network functions: `find_snap()`, `street_network_bbox()`, `street_network_to_sf()`

Examples

```
library(r5r)

# build transport network
path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(path)

# extract transit network from r5r_network
transit_net <- transit_network_to_sf(r5r_network)

stop_r5(r5r_network)
```

`travel_time_matrix` *Calculate travel time matrix between origin destination pairs considering a departure time*

Description

Fast computation of travel time estimates between one or multiple origin destination pairs. This function considers a departure time set by the user. If you want to calculate travel times considering a time of arrival, have a look at the `arrival_travel_time_matrix()` function.

Usage

```
travel_time_matrix(
  r5r_network,
  r5r_core = deprecated(),
  origins,
  destinations,
  mode = "WALK",
  mode_egress = "WALK",
  departure_datetime = Sys.time(),
  time_window = 10L,
  percentiles = 50L,
  max_walk_time = Inf,
  max_bike_time = Inf,
  max_car_time = Inf,
  max_trip_duration = 120L,
  walk_speed = 3.6,
  bike_speed = 12,
  max_rides = 3,
  max_lts = 2,
  fare_structure = NULL,
  max_fare = Inf,
  new_carspeeds = NULL,
  carspeed_scale = 1,
  new_lts = NULL,
  draws_per_minute = 5L,
  n_threads = Inf,
  verbose = FALSE,
  progress = FALSE,
  output_dir = NULL
)
```

Arguments

`r5r_network`	A routable transport network created with `build_network()`.
`r5r_core`	The `r5r_core` argument is deprecated as of r5r v2.3.0. Please use the `r5r_network` argument instead.

origins, destinations
: Either a POINT sf object with WGS84 CRS, or a data.frame containing the columns id, lon and lat.

mode
: A character vector. The transport modes allowed for access, transfer and vehicle legs of the trips. Defaults to WALK. Please see details for other options.

mode_egress
: A character vector. The transport mode used after egress from the last public transport. It can be either WALK, BICYCLE or CAR. Defaults to WALK. Ignored when public transport is not used.

departure_datetime
: A POSIXct object. Please note that the departure time only influences public transport legs. When working with public transport networks, please check the calendar.txt within your GTFS feeds for valid dates. Please see details for further information on how datetimes are parsed.

time_window
: An integer. The time window in minutes for which r5r will calculate multiple travel time matrices departing each minute. Defaults to 10 minutes. By default, the function returns the result based on median travel times, but the user can set the percentiles parameter to extract more results. Please read the time window vignette for more details on its usage vignette("time_window", package = "r5r")

percentiles
: An integer vector (max length of 5). Specifies the percentile to use when returning travel time estimates within the given time window. For example, if the 25th travel time percentile between A and B is 15 minutes, 25% of all trips taken between these points within the specified time window are shorter than 15 minutes. Defaults to 50, returning the median travel time. If a vector with length bigger than 1 is passed, the output contains an additional column for each percentile specifying the percentile travel time estimate. each estimate. Due to upstream restrictions, only 5 percentiles can be specified at a time. For more details, please see R5 documentation at https://docs.conveyal.com/analysis/methodology#accounting-for-variability.

max_walk_time
: An integer. The maximum walking time (in minutes) to access and egress the transit network, to make transfers within the network or to complete walk-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set max_walk_time to 15, you could get trips with an up to 15 minutes walk leg to reach transit and another up to 15 minutes walk leg to reach the destination after leaving transit. In walk-only trips, whenever max_walk_time differs from max_trip_duration, the lowest value is considered.

max_bike_time
: An integer. The maximum cycling time (in minutes) to access and egress the transit network, to make transfers within the network or to complete bicycle-only trips. Defaults to no restrictions (numeric value of Inf), as long as max_trip_duration is respected. When routing transit trips, the max time is considered separately for each leg (e.g. if you set max_bike_time to 15, you could get trips with an up to 15 minutes cycle leg to reach transit and another up to 15 minutes cycle leg to reach the destination after leaving transit. In bicycle-only trips, whenever max_bike_time differs from max_trip_duration, the lowest value is considered.

max_car_time An integer. The maximum driving time (in minutes) to access and egress the transit network. Defaults to no restrictions, as long as max_trip_duration is respected. The max time is considered separately for each leg (e.g. if you set max_car_time to 15 minutes, you could potentially drive up to 15 minutes to reach transit, and up to *another* 15 minutes to reach the destination after leaving transit). Defaults to Inf, no limit.

max_trip_duration
An integer. The maximum trip duration in minutes. Defaults to 120 minutes (2 hours).

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

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

max_rides An integer. The maximum number of public transport rides allowed in the same trip. Defaults to 3.

max_lts An integer 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. Please see details for more information.

fare_structure A fare structure object, following the convention set in setup_fare_structure(). This object describes how transit fares should be calculated. Please see the fare structure vignette to understand how this object is structured: vignette("fare_structure", package = "r5r").

max_fare A number. The maximum value that trips can cost when calculating the fastest journey between each origin and destination pair.

new_carspeeds A data.frame specifying the new car speed for each OSM edge id. This table must contain columns osm_id, max_speed and speed_type. The "speed_type" column is of class character and it indicates whether the values in "max_speed" should be interpreted as percentages of original speeds ("scale") or as absolute speeds ("km/h"). Alternatively, the new_carspeeds parameter can receive an sf data.frame with POLYGON geometry that indicates the new car speed for all the roads that fall within each polygon. In this case, the table must contain the columns poly_id with a unique id for each polygon, scale with the new speed scaling factors and priority, which is a number ranking which polygon should be considered in case of overlapping polygons. See more into in the link to congestion vignette.

carspeed_scale Numeric. The default car speed to use for road segments not specified in new_carspeeds. By default, it is NULL and the speeds of the unlisted roads are kept unchanged.

new_lts A data.frame specifying the new LTS levels for each OSM edge id. The table must contain columns osm_id and lts. Alternatively, the new_lts parameter can receive an sf data.frame with LINESTRING geometry. R5 will then find the nearest road for each LINESTRING and update its LTS value accordingly.

draws_per_minute
An integer. The number of Monte Carlo draws to perform per time window minute when calculating travel time matrices and when estimating accessibility. Defaults to 5. This would mean 300 draws in a 60-minute time window, for example. This parameter only affects the results when the GTFS feeds contain

a frequencies.txt table. If the GTFS feed does not have a frequency table, r5r still allows for multiple runs over the set time_window but in a deterministic way.

n_threads — An integer. The number of threads to use when running the router in parallel. Defaults to use all available threads (Inf).

verbose — A logical. Whether to show R5 informative messages when running the function. Defaults to FALSE (please note that in such case R5 error messages are still shown). Setting verbose to TRUE shows detailed output, which can be useful for debugging issues not caught by r5r.

progress — A logical. Whether to show a progress counter when running the router. Defaults to FALSE. 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.

output_dir — Either NULL or a path to an existing directory. When not NULL (the default), the function will write one .csv file with the results for each origin in the specified directory. In such case, the function returns the path specified in this parameter. This parameter is particularly useful when running on memory-constrained settings because writing the results directly to disk prevents r5r from loading them to RAM memory.

Value

A data.table with travel time estimates (in minutes) between origin and destination pairs. Pairs whose trips couldn't be completed within the maximum travel time and/or whose origin is too far from the street network are not returned in the data.table. If output_dir is not NULL, the function returns the path specified in that parameter, in which the .csv files containing the results are saved.

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.

Level of Traffic Stress (LTS)

When cycling is enabled in R5 (by passing the value BIKE to either mode or mode_egress), 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 and 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. Routing also reverts to walking if the street segment is tagged as non-bikable in OSM (e.g. a staircase), independently of the specified max LTS.

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 only for "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.

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.

Routing algorithm

The `travel_time_matrix()`, `expanded_travel_time_matrix()`, `arrival_travel_time_matrix()` and `accessibility()` functions use 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). However, whenever the user includes transit fares inputs to these functions, they automatically switch to use an R5-specific extension to the McRAPTOR routing algorithm.

- 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. doi:10.3141/265306
- Delling, D., Pajor, T., & Werneck, R. F. (2015). Round-based public transit routing. Transportation Science, 49(3), 591-604. doi:10.1287/trsc.2014.0534

See Also

Other routing: `arrival_travel_time_matrix()`, `detailed_itineraries()`, `expanded_travel_time_matrix()`, `pareto_frontier()`

Examples

```
library(r5r)

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

```
r5r_network <- build_network(data_path)

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

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

ttm <- travel_time_matrix(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = departure_datetime,
  max_trip_duration = 60
)
head(ttm)

# using a larger time window
ttm <- travel_time_matrix(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = departure_datetime,
  time_window = 30,
  max_trip_duration = 60
)
head(ttm)

# selecting different percentiles
ttm <- travel_time_matrix(
  r5r_network,
  origins = points,
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = departure_datetime,
  time_window = 30,
  percentiles = c(25, 50, 75),
  max_trip_duration = 60
)
head(ttm)

# use a fare structure and set a max fare to take monetary constraints into
# account
fare_structure <- read_fare_structure(
  file.path(data_path, "fares/fares_poa.zip")
)
ttm <- travel_time_matrix(
  r5r_network,
  origins = points,
```

```
  destinations = points,
  mode = c("WALK", "TRANSIT"),
  departure_datetime = departure_datetime,
  fare_structure = fare_structure,
  max_fare = 5,
  max_trip_duration = 60,
)
head(ttm)

stop_r5(r5r_network)
```

`write_fare_structure` *Write a fare structure object to disk*

Description

Writes a fare structure object do disk. Fare structure is saved as a collection of `.csv` files inside a `.zip` file.

Usage

```
write_fare_structure(fare_structure, file_path)
```

Arguments

`fare_structure`	A fare structure object, following the convention set in `setup_fare_structure()`. This object describes how transit fares should be calculated. Please see the fare structure vignette to understand how this object is structured: `vignette("fare_structure", package = "r5r")`.
`file_path`	A path to a `.zip` file. Where the fare structure should be written to.

Value

The path passed to `file_path`, invisibly.

See Also

Other fare structure: `read_fare_structure()`, `setup_fare_structure()`

Examples

```
library(r5r)

data_path <- system.file("extdata/poa", package = "r5r")
r5r_network <- build_network(data_path)

fare_structure <- setup_fare_structure(r5r_network, base_fare = 5)
```

```
tmpfile <- tempfile("sample_fare_structure", fileext = ".zip")
write_fare_structure(fare_structure, tmpfile)
```

Index

∗ **Build network**
  build_network, 16
  download_r5, 25
  setup_r5, 48
∗ **Cache data**
  r5r_cache, 44
∗ **Isochrone**
  isochrone, 33
∗ **accessibility**
  accessibility, 3
∗ **fare structure**
  read_fare_structure, 46
  setup_fare_structure, 46
  write_fare_structure, 60
∗ **network functions**
  find_snap, 31
  street_network_bbox, 51
  street_network_to_sf, 52
  transit_network_to_sf, 53
∗ **routing**
  arrival_travel_time_matrix, 10
  detailed_itineraries, 19
  expanded_travel_time_matrix, 26
  pareto_frontier, 39
  travel_time_matrix, 54
∗ **support functions**
  get_gtfs_errors, 33
  stop_r5, 50

accessibility, 3
accessibility(), *9*, *14*, *30*, *37*, *46, 47*, *58*
arrival_travel_time_matrix, 10, *24*, *30*, *43*, *58*
arrival_travel_time_matrix(), *9*, *14*, *30*, *37*, *54*, *58*

build_network, 16, *25*, *49*
build_network(), *4*, *11*, *20*, *27*, *32*, *34*, *40*, *47, 48*, *51–54*

check_transit_availability, 17

data.table::fread(), *46*
detailed_itineraries, *15*, 19, *30*, *43*, *58*
detailed_itineraries(), *24*, *43*
download_r5, *17*, 25, *49*

exists_tiff, *33*, *50*
expanded_travel_time_matrix, *15*, *24*, 26, *43*, *58*
expanded_travel_time_matrix(), *9, 10*, *14*, *30*, *37*, *46*, *58*

fileurl_from_metadata, *33*, *50*
find_snap, 31, *51–53*

get_gtfs_errors, 33, *50*

isochrone, 33

pareto_frontier, *15*, *24*, *30*, 39, *58*
pareto_frontier(), *24*, *43*, *46, 47*

r5r_cache, 44
r5r_sitrep, 45
read_fare_structure, 46, *48*, *60*

setup_fare_structure, 46, *46*, *60*
setup_fare_structure(), *6*, *22*, *41*, *56*, *60*
setup_r5, *17*, *25*, 48
start_r5r_java, *33*, *50*
stop_r5, *33*, 50
street_network_bbox, *32*, 51, *52, 53*
street_network_to_sf, *32*, *51*, 52, *53*
street_network_to_sf(), *22*

tempdir_unique, *33*, *50*
transit_network_to_sf, *32*, *51, 52*, 53
travel_time_matrix, *15*, *24*, *30*, *43*, 54
travel_time_matrix(), *9, 10*, *14*, *30*, *37*, *46, 47*, *58*

travel_time_surface, *33*, *50*

validate_bad_osm_ids, *33*, *50*

write_fare_structure, *46*, *48*, 60