Package: otpr (via r-universe)

November 5, 2024

Title An R Wrapper for the 'OpenTripPlanner' REST API

Version 0.5.1

Description A wrapper for the 'OpenTripPlanner' <http://www.opentripplanner.org/> REST API. Queries are submitted to the relevant 'OpenTripPlanner' API resource, the response is parsed and useful R objects are returned.

License MIT + file LICENSE

URL https://github.com/marcusyoung/otpr

BugReports https://github.com/marcusyoung/otpr/issues

Language en-GB

Encoding UTF-8

Imports checkmate, httr, geojsonsf, janitor, jsonlite, sf, urltools, dplyr, rrapply, rlang

RoxygenNote 7.1.2

Suggests testthat, covr

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

RemoteUrl https://github.com/marcusyoung/otpr

RemoteRef HEAD

RemoteSha bbb4e2a0c58ce75889bf96191e35344eab0215d1

Contents

`otp_connect` *Set up and confirm a connection to an OTP instance.*

Description

Defines the parameters required to connect to a router on an OTP instance and, if required, confirms that the instance and router are queryable.

Usage

```
otp_connect(
  hostname = "localhost",
  router = "default",
  port = 8080,
  tz = Sys.timezone(),
  ssl = FALSE,
  check = TRUE
)
```

Arguments

`hostname`	A string, e.g. "ec2-34-217-73-26.us-west-2.compute.amazonaws.com". Optional, default is "localhost".
`router`	A string, e.g. "UK2018". Optional, default is "default". Do not specify for OTPv2 which does not support named routers.
`port`	A positive integer. Optional, default is 8080.
`tz`	A string, containing the time zone of the router's graph. Optional. This should be a valid time zone (checked against vector returned by 'OlsonNames()'). For example: "Europe/Berlin". Default is the timezone of the current system (obtained from `Sys.timezone()`). Using the default will be ok if the current system time zone is the same as the time zone of the OTP graph.
`ssl`	Logical, indicates whether to use https. Optional, default is FALSE.
`check`	Deprecated and has no effect.

Value

Returns S3 object of class otpconnect if reachable.

Examples

```
## Not run:
otpcon <- otpr_connect()
otpcon <- otpr_connect(router = "UK2018",
                       ssl = TRUE)
otpcon <- otpr_connect(hostname = "ec2.us-west-2.compute.amazonaws.com",
                       router = "UK2018",
```

```
                    port = 8888,
                    ssl = TRUE)

## End(Not run)
```

`otp_create_surface` *Creates a travel time surface (OTPv1 only).*

Description

Creates a travel time surface for an origin point. A surface contains the travel time to every geographic coordinate that can be reached from that origin (up to a hard coded limit in OTP of 120 minutes). Optionally, the surface can be saved as a raster file (GeoTIFF) to a designated directory.

Usage

```
otp_create_surface(
  otpcon,
  getRaster = FALSE,
  rasterPath = tempdir(),
  fromPlace,
  mode = "TRANSIT",
  date = format(Sys.Date(), "%m-%d-%Y"),
  time = format(Sys.time(), "%H:%M:%S"),
  maxWalkDistance = NULL,
  walkReluctance = 2,
  waitReluctance = 1,
  transferPenalty = 0,
  minTransferTime = 0,
  batch = TRUE,
  arriveBy = TRUE,
  extra.params = list()
)
```

Arguments

`otpcon`	An OTP connection object produced by `otp_connect`.
`getRaster`	Logical. Whether or not to download a raster (geoTIFF) of the generated surface. Default FALSE.
`rasterPath`	Character. Path of a directory where the the surface raster should be saved if `getRaster` is TRUE. Default is `tempdir()`. Use forward slashes on Windows. The file will be named surface_id.tiff, with id replaced by the OTP id assigned to the surface.
`fromPlace`	Numeric vector, Latitude/Longitude pair, e.g. 'c(53.48805, -2.24258)'. This is the origin of the surface to be created.

mode	Character vector, mode(s) of travel. Valid values are: WALK, BICYCLE, CAR, TRANSIT, BUS, RAIL, TRAM, SUBWAY OR 'c("TRANSIT", "BICYCLE")'. TRANSIT will use all available transit modes. Default is CAR. WALK mode is automatically added for TRANSIT, BUS, RAIL, TRAM, and SUBWAY.
date	Character, must be in the format mm-dd-yyyy. This is the desired date of travel. Only relevant for transit modes. Default is the current system date.
time	Character, must be in the format hh:mm:ss. If arriveBy is FALSE (the default) this is the desired departure time, otherwise the desired arrival time. Only relevant for transit modes. Default is the current system time.
maxWalkDistance	Numeric. The maximum distance (in meters) that the user is willing to walk. Default is NULL (the parameter is not passed to the API and the OTP default of unlimited takes effect). This is a soft limit in OTPv1 and is ignored if the mode is WALK only. In OTPv2 this parameter imposes a hard limit on WALK, CAR and BICYCLE modes (see: http://docs.opentripplanner.org/en/latest/OTP2-MigrationGuide/#router-config).
walkReluctance	A single numeric value. A multiplier for how bad walking is compared to being in transit for equal lengths of time. Default = 2.
waitReluctance	A single numeric value. A multiplier for how bad waiting for a transit vehicle is compared to being on a transit vehicle. This should be greater than 1 and less than walkReluctance (see API docs). Default = 1.
transferPenalty	Integer. An additional penalty added to boardings after the first. The value is in OTP's internal weight units, which are roughly equivalent to seconds. Set this to a high value to discourage transfers. Default is 0.
minTransferTime	Integer. The minimum time, in seconds, between successive trips on different vehicles. This is designed to allow for imperfect schedule adherence. This is a minimum; transfers over longer distances might use a longer time. Default is 0.
batch	Logical. Set to TRUE by default. This is required to tell OTP to allow a query without the toPlace parameter. This is necessary as we want to build paths to all destinations from one origin.
arriveBy	Logical. Whether a trip should depart (FALSE) or arrive (TRUE) at the specified date and time. Default is FALSE.
extra.params	A list of any other parameters accepted by the OTP API SurfaceResource entry point. For advanced users. Be aware that otpr will carry out no validation of these additional parameters. They will be passed directly to the API.

Details

There are a few things to note regarding the raster image that OTP creates:

- The travel time cutoff for a surface is hard-coded within OTP at 120 minutes. Every grid cell within the extent of the graph that is 120 minutes travel time or beyond, or not accessible, is given the value of 120.
- Any grid cell outside of the extent of the network (i.e. unreachable) is given the value 128.

- It is advisable to interpret the raster of a surface in conjunction with results from evaluating the surface.
- OTP can take a while the first time a raster of a surface is generated after starting up. Subsequent rasters (even for different origins) are much faster to generate.

Value

Assuming no error, returns a list of 5 elements:

- `errorId` Will be "OK" if no error condition.
- `surfaceId` The id of the surface that was evaluated.
- `surfaceRecord` Details of the parameters used to create the surface.
- `rasterDownload` The path to the saved raster file (if `getRaster` was set to TRUE and a valid path was provided via `rasterPath`.)
- `query` The URL that was submitted to the OTP API.

If there is an error, a list containing 3 elements is returned:

- `errorId` The id code of the error.
- `errorMessage` The error message.
- `query` The URL that was submitted to the OTP API.

Examples

```
## Not run:
otp_create_surface(otpcon, fromPlace = c(53.43329,-2.13357), mode = "TRANSIT",
maxWalkDistance = 1600, getRaster = TRUE)

otp_create_surface(otpcon, fromPlace = c(53.43329,-2.13357), date = "03-26-2019",
time = "08:00:00", mode = "BUS", maxWalkDistance = 1600, getRaster = TRUE,
rasterPath = "C:/temp")

## End(Not run)
```

---

otp_evaluate_surface *Evaluates an existing travel time surface (OTPv1 only).*

Description

Evaluates an existing travel time surface. Using a pointset from a specified CSV file, the travel time to each point is obtained from the specified surface. Accessibility indicators are then generated for one or more 'opportunity' columns in the pointset. For example, you might have the number of jobs available at each location, or the number of hospital beds.

Usage

```
otp_evaluate_surface(otpcon, surfaceId, pointset, detail = FALSE)
```

Arguments

otpcon	An OTP connection object produced by otp_connect.
surfaceId	Integer, the id number of an existing surface created using otp_create_surface().
pointset	Character string, the name of a pointset known to OTP. A pointset is contained in a CSV file present in the pointset directory location passed to OTP at startup. The name of the pointset is the name of the file (without extension).
detail	logical, whether the travel time from the surface origin to each location in the pointset should be returned. Default is FALSE.

Details

This function requires OTP to have been started with the --analyst switch and the --pointset parameter set to the path of a directory containing the pointset file(s).

Value

Assuming no error, returns a list containing 4 or more elements:

- errorId Will be "OK" if no error condition.
- surfaceId The id of the surface that was evaluated.
- One or more dataframes for each of the 'opportunity' columns in the pointset CSV file. Each dataframe contains four columns:
  - minutes. The time from the surface origin in one-minute increments.
  - counts. The number of the opportunity locations reached within each minute interval.
  - sum. The sum of the opportunities at each of the locations reached within each minute interval.
  - cumsums. A cumulative sum of the opportunities reached.
- If detail was set to TRUE then an additional dataframe containing the time taken (in seconds) to reach each point in the pointset CSV file. If a point was not reachable the time will be recorded as NA.
- query The URL that was submitted to the OTP API.

If there is an error, a list containing 3 elements is returned:

- errorId The id code of the error.
- errorMessage The error message.
- query The URL that was submitted to the OTP API.

Examples

```
## Not run:
otp_evaluate_surface(otpcon, surfaceId = 0, pointset = "jobs", detail = TRUE)

## End(Not run)
```

`otp_get_distance` *Finds the distance in metres between supplied origin and destination*

Description

Finds the distance in metres between supplied origin and destination. Only makes sense for walk, cycle or car modes (not transit)

Usage

```
otp_get_distance(otpcon, fromPlace, toPlace, mode = "CAR")
```

Arguments

`otpcon`	An OTP connection object produced by `otp_connect`.
`fromPlace`	Numeric vector, Latitude/Longitude pair, e.g. ‘c(53.48805, -2.24258)‘
`toPlace`	Numeric vector, Latitude/Longitude pair, e.g. ‘c(53.36484, -2.27108)‘
`mode`	Character vector, single mode of travel. Valid values are WALK, BICYCLE, or CAR. Default is CAR.

Value

If OTP has not returned an error then a list containing `errorId` with the value "OK", and the `distance` in metres. If OTP has returned an error then a list containing `errorId` with the OTP error code and `errorMessage` with the error message returned by OTP. In both cases there will be a third element named `query` which is a character string containing the URL that was submitted to the OTP API.

Examples

```
## Not run:
otp_get_distance(otpcon, fromPlace = c(53.48805, -2.24258), toPlace = c(53.36484, -2.27108))

otp_get_distance(otpcon, fromPlace = c(53.48805, -2.24258), toPlace = c(53.36484, -2.27108),
mode = "WALK")

## End(Not run)
```

`otp_get_isochrone` *Returns one or more travel time isochrones (OTPv1 only)*

Description

Returns one or more travel time isochrones in either GeoJSON format or as an **sf** object. Only works correctly for walk and/or transit modes - a limitation of OTP. Isochrones can be generated either *from* a location or *to* a location.

Usage

```
otp_get_isochrone(
  otpcon,
  location,
  fromLocation = TRUE,
  format = "JSON",
  mode = "TRANSIT",
  date = format(Sys.Date(), "%m-%d-%Y"),
  time = format(Sys.time(), "%H:%M:%S"),
  cutoffs,
  batch = TRUE,
  arriveBy = FALSE,
  maxWalkDistance = NULL,
  walkReluctance = 2,
  waitReluctance = 1,
  transferPenalty = 0,
  minTransferTime = 0,
  extra.params = list()
)
```

Arguments

`otpcon`	An OTP connection object produced by `otp_connect`.
`location`	Numeric vector, Latitude/Longitude pair, e.g. ‘c(53.48805, -2.24258)‘
`fromLocation`	Logical. If TRUE (default) the isochrone will be generated *from* the `location`. If FALSE the isochrone will be generated *to* the `location`.
`format`	Character, required format of returned isochrone(s). Either JSON (returns GeoJSON) or SF (returns simple feature collection). Default is JSON.
`mode`	Character vector, mode(s) of travel. Valid values are: WALK, TRANSIT, BUS, RAIL, TRAM, SUBWAY. TRANSIT will use all available transit modes. Default is TRANSIT. WALK mode is automatically added to TRANSIT, BUS, RAIL, TRAM, and SUBWAY. Due to an OTP limitation this function is *not* suitable for CAR or BICYCLE modes.
`date`	Character, must be in the format mm-dd-yyyy. This is the desired date of travel. Only relevant for transit modes. Default is the current system date.

`time` Character, must be in the format hh:mm:ss. If `arriveBy` is FALSE (the default) this is the desired departure time, otherwise the desired arrival time. Only relevant for transit modes. Default is the current system time.

`cutoffs` Numeric vector, containing the cutoff times in seconds. for example: 'c(900, 1800, 2700)' would request 15, 30 and 60 minute isochrones. Can be a single value.

`batch` Logical. If true, goal direction is turned off and a full path tree is built

`arriveBy` Logical. Whether a trip should depart (FALSE) or arrive (TRUE) at the specified date and time. Default is FALSE.

`maxWalkDistance` Numeric. The maximum distance (in meters) that the user is willing to walk. Default is NULL (the parameter is not passed to the API and the OTP default of unlimited takes effect). This is a soft limit in OTPv1 and is ignored if the mode is WALK only. In OTPv2 this parameter imposes a hard limit on WALK, CAR and BICYCLE modes (see: http://docs.opentripplanner.org/en/latest/OTP2-MigrationGuide/#router-config).

`walkReluctance` A single numeric value. A multiplier for how bad walking is compared to being in transit for equal lengths of time. Default = 2.

`waitReluctance` A single numeric value. A multiplier for how bad waiting for a transit vehicle is compared to being on a transit vehicle. This should be greater than 1 and less than `walkReluctance` (see API docs). Default = 1.

`transferPenalty` Integer. An additional penalty added to boardings after the first. The value is in OTP's internal weight units, which are roughly equivalent to seconds. Set this to a high value to discourage transfers. Default is 0.

`minTransferTime` Integer. The minimum time, in seconds, between successive trips on different vehicles. This is designed to allow for imperfect schedule adherence. This is a minimum; transfers over longer distances might use a longer time. Default is 0.

`extra.params` A list of any other parameters accepted by the OTP API LIsochrone entry point. For advanced users. Be aware that otpr will carry out no validation of these additional parameters. They will be passed directly to the API. Do not pass 'fromPlace' or 'toPlace' to this function. These parameters are handled internally based on the values of `location` and `fromLocation`.

Value

Returns a list. First element in the list is `errorId`. This is "OK" if OTP successfully returned the isochrone(s), otherwise it is "ERROR". The second element of list varies:

- If `errorId` is "ERROR" then `response` contains the OTP error message.
- If `errorId` is "OK" then `response` contains the the isochrone(s) in either GeoJSON format or as an **sf** object, depending on the value of the `format` argument.

The third element of the list is `query` which is a character string containing the URL that was submitted to the OTP API.

Examples

```
## Not run:
otp_get_isochrone(otpcon, location = c(53.48805, -2.24258), cutoffs = c(900, 1800, 2700))

otp_get_isochrone(otpcon, location = c(53.48805, -2.24258), fromLocation = FALSE,
cutoffs = c(900, 1800, 2700), mode = "BUS")

## End(Not run)
```

`otp_get_times` *Queries OTP for the time or detailed itinerary for a trip between an origin and destination*

Description

In its simplest use case the function returns the time in minutes between an origin and destination by the specified mode(s) for the top itinerary returned by OTP. If `detail` is set to TRUE one or more detailed trip itineraries are returned, including the time by each mode (if a multimodal trip), waiting time and the number of transfers. Optionally, the details of each journey leg for each itinerary can also be returned.

Usage

```
otp_get_times(
  otpcon,
  fromPlace,
  toPlace,
  mode = "CAR",
  date = format(Sys.Date(), "%m-%d-%Y"),
  time = format(Sys.time(), "%H:%M:%S"),
  maxWalkDistance = NULL,
  walkReluctance = 2,
  waitReluctance = 1,
  arriveBy = FALSE,
  transferPenalty = 0,
  minTransferTime = 0,
  maxItineraries = 1,
  detail = FALSE,
  includeLegs = FALSE,
  extra.params = list()
)
```

Arguments

`otpcon`	An OTP connection object produced by `otp_connect`.
`fromPlace`	Numeric vector, Latitude/Longitude pair, e.g. ‘c(53.48805, -2.24258)‘
`toPlace`	Numeric vector, Latitude/Longitude pair, e.g. ‘c(53.36484, -2.27108)‘

`mode` Character vector, mode(s) of travel. Valid values are: WALK, BICYCLE, CAR, TRANSIT, BUS, RAIL, TRAM, SUBWAY OR 'c("TRANSIT", "BICYCLE")'. TRANSIT will use all available transit modes. Default is CAR. WALK mode is automatically added for TRANSIT, BUS, RAIL, TRAM, and SUBWAY.

`date` Character, must be in the format mm-dd-yyyy. This is the desired date of travel. Only relevant for transit modes. Default is the current system date.

`time` Character, must be in the format hh:mm:ss. If `arriveBy` is FALSE (the default) this is the desired departure time, otherwise the desired arrival time. Only relevant for transit modes. Default is the current system time.

`maxWalkDistance` Numeric. The maximum distance (in meters) that the user is willing to walk. Default is NULL (the parameter is not passed to the API and the OTP default of unlimited takes effect). This is a soft limit in OTPv1 and is ignored if the mode is WALK only. In OTPv2 this parameter imposes a hard limit on WALK, CAR and BICYCLE modes (see: http://docs.opentripplanner.org/en/latest/OTP2-MigrationGuide/#router-config).

`walkReluctance` A single numeric value. A multiplier for how bad walking is compared to being in transit for equal lengths of time. Default = 2.

`waitReluctance` A single numeric value. A multiplier for how bad waiting for a transit vehicle is compared to being on a transit vehicle. This should be greater than 1 and less than `walkReluctance` (see API docs). Default = 1.

`arriveBy` Logical. Whether a trip should depart (FALSE) or arrive (TRUE) at the specified date and time. Default is FALSE.

`transferPenalty` Integer. An additional penalty added to boardings after the first. The value is in OTP's internal weight units, which are roughly equivalent to seconds. Set this to a high value to discourage transfers. Default is 0.

`minTransferTime` Integer. The minimum time, in seconds, between successive trips on different vehicles. This is designed to allow for imperfect schedule adherence. This is a minimum; transfers over longer distances might use a longer time. Default is 0.

`maxItineraries` Integer. Controls the number of trip itineraries that are returned when `detail` is set to TRUE. This is not an OTP parameter. All suggested itineraries are allowed to be returned by the OTP server. The function will return them to the user in the order they were provided by OTP up to the maximum specified by this parameter. Default is 1. This is an alternative to using the OTP `maxNumItineraries` parameter which has problematic behaviour.

`detail` Logical. When set to FALSE a single trip time is returned. When set to TRUE one or more detailed trip itineraries are returned (dependent on `maxItineraries`). Default is FALSE.

`includeLegs` Logical. Determines whether or not details of each journey leg are returned. If TRUE then a nested dataframe of journeys legs will be returned for each itinerary if `detail` is also TRUE. Default is FALSE.

`extra.params` A list of any other parameters accepted by the OTP API PlannerResource entry point. For advanced users. Be aware that otpr will carry out no validation of these additional parameters. They will be passed directly to the API.

Details

If you plan to use the function in simple-mode - where just the duration of the top itinerary is returned - it is advisable to first review several detailed itineraries to ensure that the parameters you have set are producing sensible results.

If requested using `includeLegs`, the itineraries dataframe will contain a column called 'legs' which has a nested legs dataframe for each itinerary. Each legs dataframe will contain a set of core columns that are consistent across all queries. However, as the OTP API does not consistently return the same attributes for legs, there will be some variation in columns returned. You should bare this in mind if your post processing uses these columns (e.g. by checking for column existence).

Value

Returns a list of three or four elements. The first element in the list is `errorId`. This is "OK" if OTP has not returned an error. Otherwise it is the OTP error code. The second element of list varies:

- If OTP has returned an error then `errorMessage` contains the OTP error message.
- If there is no error and `detail` is FALSE then the `duration` in minutes is returned as an integer. This is the duration of the top itinerary returned by the OTP server.
- If there is no error and `detail` is TRUE then `itineraries` as a dataframe.

The third element of the list is `query`. This is a character string containing the URL that was submitted to the OTP API.

Examples

```
## Not run:
otp_get_times(otpcon, fromPlace = c(53.48805, -2.24258), toPlace = c(53.36484, -2.27108))

otp_get_times(otpcon, fromPlace = c(53.48805, -2.24258), toPlace = c(53.36484, -2.27108),
mode = "BUS", date = "03-26-2019", time = "08:00:00")

otp_get_times(otpcon, fromPlace = c(53.48805, -2.24258), toPlace = c(53.36484, -2.27108),
mode = "BUS", date = "03-26-2019", time = "08:00:00", detail = TRUE)

## End(Not run)
```

Index