

# Package ‘otpr’

January 31, 2020

**Title** An R Wrapper for the 'OpenTripPlanner' REST API

**Version** 0.3.0

**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

**Language** en-GB

**Encoding** UTF-8

**LazyData** true

**Imports** checkmate, httr, janitor, jsonlite, geojsonsf

**RoxygenNote** 6.1.1

**Suggests** testthat

**NeedsCompilation** no

**Author** Marcus Young [aut, cre]

**Maintainer** Marcus Young <[M.A.Young@soton.ac.uk](mailto:M.A.Young@soton.ac.uk)>

**Repository** CRAN

**Date/Publication** 2020-01-31 21:00:03 UTC

## R topics documented:

otp_connect . . . . .	2
otp_get_distance . . . . .	3
otp_get_isochrone . . . . .	3
otp_get_times . . . . .	5
<b>Index</b>	<b>7</b>

---

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".
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	Logical. If TRUE connection object is only returned if OTP instance and router are confirmed reachable. Optional, default is TRUE.

### Value

Returns S3 object of class otpconnect. If check is TRUE and the router is not reachable the object is not returned.

### Examples

```
## Not run:
otpcon <- otp_connect()
otpcon <- otp_connect(router = "UK2018",
                     ssl = TRUE)
otpcon <- otp_connect(hostname = "ec2.us-west-2.compute.amazonaws.com",
                     router = "UK2018",
                     port = 8888,
                     ssl = TRUE)

## End(Not run)
```

---

otp_get_distance	<i>Finds the distance in metres between supplied origin and destination</i>
------------------	---

---

### 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 <a href="#">otp_connect</a> .
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.

### 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	<i>Returns one or more travel time isochrones</i>
-------------------	---

---

### Description

Returns one or more travel time isochrone 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, time, cutoffs,
  batch = TRUE, arriveBy = FALSE, maxWalkDistance = 800,
  walkReluctance = 2, transferPenalty = 0, minTransferTime = 0)
```

**Arguments**

otpcon	An OTP connection object produced by <a href="#">otp_connect</a> .
location	Numeric vector, Latitude/Longitude pair, e.g. 'c(53.48805, -2.24258)'
fromLocation	Logical. If TRUE (default) the isochrone will be generated <i>from</i> the location. If FALSE the isochrone will be generated <i>to</i> 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, mode of travel. Valid values are: WALK, TRANSIT, BUS, or RAIL. Note that WALK mode is automatically included for TRANSIT, BUS and RAIL. TRANSIT will use all available transit modes. Default is TRANSIT.
date	Character, must be in the format mm-dd-yyyy. This is the desired date of travel. Only relevant if mode includes public transport. Default is 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. Default is 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 the specified date and time is for departure (FALSE) or arrival (TRUE). Default is FALSE.
maxWalkDistance	Numeric. The maximum distance (in meters) the user is willing to walk. Default = 800.
walkReluctance	Integer. A multiplier for how bad walking is, compared to being in transit for equal lengths of time. Default = 2.
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.

**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.

**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	<i>Finds the time in minutes between supplied origin and destination</i>
---------------	--

---

**Description**

Finds the time in minutes between supplied origin and destination by specified mode(s). If detail is set to TRUE returns time for each mode, waiting time and number of transfers.

**Usage**

```
otp_get_times(otpcon, fromPlace, toPlace, mode = "CAR", date, time,
maxWalkDistance = 800, walkReluctance = 2, arriveBy = FALSE,
transferPenalty = 0, minTransferTime = 0, detail = FALSE,
includeLegs = FALSE)
```

**Arguments**

otpcon	An OTP connection object produced by <code>otp_connect</code> .
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: TRANSIT, WALK, BICYCLE, CAR, BUS, RAIL, OR 'c("TRANSIT", "BICYCLE)". Note that WALK mode is automatically included for TRANSIT, BUS, and RAIL. TRANSIT will use all available transit modes. Default is CAR.
date	Character, must be in the format mm-dd-yyyy. This is the desired date of travel. Only relevant if mode includes public transport. Default is 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 if mode includes public transport. Default is current system time.

maxWalkDistance	Numeric. The maximum distance (in meters) the user is willing to walk. Default = 800.
walkReluctance	Integer. A multiplier for how bad walking is, compared to being in transit for equal lengths of time. Default = 2.
arriveBy	Logical. Whether 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.
detail	Logical. Default is FALSE.
includeLegs	Logical. Default is FALSE. Determines whether or not details of each journey leg are returned. If TRUE then a dataframe of journeys legs will be returned but only when detail is TRUE and mode contains transit modes (Legs are not relevant for CAR, BICYCLE or WALK modes).

### Value

Returns a list. First element in the list is `errorId`. This is "OK" if OTP has not returned an error. Otherwise it is the OTP error code. 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 `duration` in minutes is returned as integer.
- If there is no error and `detail` is TRUE then `itineraries` as a dataframe.
- If there is no error and `detail` and `legs` are both TRUE then `itineraries` as a dataframe and `legs` as a dataframe.

### 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

`otp_connect`, 2, 3–5  
`otp_get_distance`, 3  
`otp_get_isochrone`, 3  
`otp_get_times`, 5