Package 'opentripplanner'

November 9, 2020

Title Setup and connect to 'OpenTripPlanner'

Version 0.3.1

Maintainer Malcolm Morgan <m.morgan1@leeds.ac.uk>

Description Setup and connect to 'OpenTripPlanner' (OTP) <http://www.opentripplanner.org/>. OTP is an open source platform for multi-modal and multi-agency journey planning written in 'Java'. The package allows you to manage a local version or connect to remote OTP server to find walking, cycling, driving, or transit routes. This package has been peer-reviewed by rOpenSci (v. 0.2.0.0).

Language EN-GB

License GPL-3

URL https://github.com/ropensci/opentripplanner, https://docs.ropensci.org/opentripplanner/

BugReports https://github.com/ropensci/opentripplanner/issues

Encoding UTF-8

LazyData true

Imports checkmate, data.table, geodist, googlePolylines, curl, rjson, pbapply, lubridate, sf (>= 0.9.3), sfheaders, tibble

RoxygenNote 7.1.1

Suggests RcppSimdJson (>= 0.1.2), covr, knitr, rmarkdown, testthat

VignetteBuilder knitr

Depends R (>= 3.2)

NeedsCompilation no

Author Malcolm Morgan [aut, cre] (<https://orcid.org/0000-0002-9488-9183>), Marcus Young [aut] (<https://orcid.org/0000-0003-4627-1116>), Robin Lovelace [aut] (<https://orcid.org/0000-0001-5679-6536>), Layik Hama [ctb] (<https://orcid.org/0000-0003-1912-4890>)

Repository CRAN

Date/Publication 2020-11-09 22:00:03 UTC

R topics documented:

opentripplanner-package . . . 2
json_example_drive . . . 3
json_example_long_drive . . . 3
json_example_transit . . . 4
otp_build_graph . . . 4
otp_check_java . . . 5
otp_connect . . . 6
otp_dl_demo . . . 7
otp_dl_jar . . . 8
otp_geocode . . . 9
otp_isochrone . . . 10
otp_make_config . . . 11
otp_plan . . . 12
otp_routing_options . . . 14
otp_setup . . . 15
otp_stop . . . 17
otp_validate_config . . . 18
otp_validate_routing_options . . . 18
otp_write_config . . . 19

Index **21**

`opentripplanner-package`
OpenTripPlanner of R

Description

The goal of OpenTripPlanner for R is to provide a simple R interface to OpenTripPlanner (OTP). The OTP is a multimodal trip planning service.

Author(s)

Maintainer: Malcolm Morgan <m.morgan1@leeds.ac.uk> (ORCID)

Authors:

- Marcus Young <M.A.Young@soton.ac.uk> (ORCID)
- Robin Lovelace <rob00x@gmail.com> (ORCID)

Other contributors:

- Layik Hama <layik.hama@gmail.com> (ORCID) [contributor]

See Also

Useful links:

- https://github.com/ropensci/opentripplanner
- https://docs.ropensci.org/opentripplanner/
- Report bugs at https://github.com/ropensci/opentripplanner/issues

---

`json_example_drive` *Example JSON for driving*

Description

Example JSON response from OTP This is used for internal testing and has no use

Usage

```
json_example_drive
```

Format

json

`json_example_long_drive`
Example JSON for driving long distance

Description

Example JSON response from OTP This is used for internal testing and has no use

Usage

```
json_example_long_drive
```

Format

json

json_example_transit *Example JSON for transit*

Description

Example JSON response from OTP This is used for internal testing and has no use

Usage

```
json_example_transit
```

Format

json

otp_build_graph *Build an OTP Graph*

Description

OTP is run in Java and requires Java commands to be typed into the command line. The function allows the parameters to be defined in R and automatically passed to Java. This function builds a OTP graph from the Open Street Map and other files.

Usage

```
otp_build_graph(
  otp = NULL,
  dir = NULL,
  memory = 2048,
  router = "default",
  analyst = FALSE
)
```

Arguments

otp	A character string, path to the OTP .jar file
dir	A character string, path to a directory containing the necessary files, see details
memory	A positive integer. Amount of memory to assign to the OTP in MB, default is 2048
router	A character string for the name of the router, must subfolder of dir/graphs, default "default". See vignettes for details.
analyst	Logical, should analyst feature be built, default FALSE. See advanced vignette for details.

Details

The OTP .jar file can be downloaded from https://repo1.maven.org/maven2/org/opentripplanner/otp/

To build an OTP graph requires the following files to be in the directory specified by the dir variable.

/graphs - A sub-directory

/default - A sub-directory with the name of the OTP router used in router' variable

osm.pbf - Required, pbf file containing the Open Street Map

router-config.json - Required, json file containing configurations settings for the OTP

gtfs.zip - Optional, and number of GTFS files with transit timetables

terrain.tif - Optional, GeoTiff image of terrain map

The function will accept any file name for the .jar file, but it must be the only .jar file in that directory OTP can support multiple routers (e.g. different regions), each router must have its own sub-directory in the graphs directory

Value

Character vector of messages produced by OTP, and will return the message "Graph built" if successful

See Also

Other setup: `otp_check_java()`, `otp_dl_demo()`, `otp_dl_jar()`, `otp_make_config()`, `otp_setup()`, `otp_stop()`, `otp_validate_config()`, `otp_write_config()`

Examples

```
## Not run:
log <- otp_build_graph(otp = "C:/otp/otp.jar", dir = "C:/data")

## End(Not run)
```

---

otp_check_java *Check Java version*

Description

Check if you have the correct version of Java for running OTP locally

Usage

```
otp_check_java()
```

See Also

Other setup: `otp_build_graph()`, `otp_dl_demo()`, `otp_dl_jar()`, `otp_make_config()`, `otp_setup()`, `otp_stop()`, `otp_validate_config()`, `otp_write_config()`

`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 query-able.

Usage

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

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". OTP can support multiple routers see advanced vignette for details.
`url`	If a non-standard URL structure is used provide a full url, default is NULL
`port`	A positive integer. Optional, default is 8080.
`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.
`timezone`	Character, timezone, defaults to local timezone

Details

The default URL structure for the OTP API is: http://<hostname>:<port>/otp/routers/<router> For example: http://localhost:8080/otp/routers/default

Functions construct the URL from the parameters provided in otpconnect objects. However some websites hosting OTP have modified the default URL structure. If this is the case you can use the `url` parameter to bypass the URL construction and provide a fully formed URL. In this case the `hostname`, `router`, `port`, and `ssl` are ignored.

Value

Returns an 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
)
otpcon <- otp_connect(
  url = "https://api.digitransit.fi:443/routing/v1/routers/hsl"
)

## End(Not run)
```

---

otp_dl_demo *Download Demo Data*

Description

Download the demonstration data for the Isle of Wight

Usage

```
otp_dl_demo(
  path_data = NULL,
  url = paste0("https://github.com/ropensci/opentripplanner/",
    "releases/download/0.1/isle-of-wight-demo.zip"),
  quiet = FALSE
)
```

Arguments

path_data	path to folder where data for OTP is to be stored
url	URL to data
quiet	logical, passed to download.file, default FALSE

See Also

Other setup: otp_build_graph(), otp_check_java(), otp_dl_jar(), otp_make_config(), otp_setup(), otp_stop(), otp_validate_config(), otp_write_config()

Examples

```
## Not run:
otp_dl_demo(tempdir())

## End(Not run)
```

`otp_dl_jar` *Download OTP Jar File*

Description

Download the OTP jar file from maven.org

Usage

```
otp_dl_jar(
  path = NULL,
  version = "1.4.0",
  file_name = paste0("otp-", version, "-shaded.jar"),
  url = "https://repo1.maven.org/maven2/org/opentripplanner/otp",
  quiet = FALSE,
  cache = TRUE
)
```

Arguments

`path`	path to folder where OTP is to be stored
`version`	a character string of the version number default is "1.4.0"
`file_name`	file name to give the otp default "otp.jar"
`url`	URL to the download server
`quiet`	logical, passed to download.file, default FALSE
`cache`	logical, default TRUE, see details

Details

As of version 0.3.0.0 ‘otp_dl_jar‘ will cache the JAR file within the package and ignore the ‘path‘ argument. You can force a new download to be saved in the ‘path‘ location by setting ‘cache = FALSE‘.

Value

The path to the OTP file

See Also

Other setup: `otp_build_graph()`, `otp_check_java()`, `otp_dl_demo()`, `otp_make_config()`, `otp_setup()`, `otp_stop()`, `otp_validate_config()`, `otp_write_config()`

Examples

```
## Not run:
otp_dl_jar(tempdir())

## End(Not run)
```

---

otp_geocode *Use OTP Geo-coder to find a location*

Description

Geo-coding converts a named place, such as a street name into a lng/lat pair.

Usage

```
otp_geocode(
  otpcon = NULL,
  query = NULL,
  autocomplete = FALSE,
  stops = TRUE,
  clusters = FALSE,
  corners = TRUE,
  type = "SF"
)
```

Arguments

otpcon	OTP connection object produced by otp_connect()
query	Character, The query string we want to geocode
autocomplete	logical Whether we should use the query string to do a prefix match, default FALSE
stops	Logical, Search for stops, either by name or stop code, default TRUE
clusters	Logical, Search for clusters by their name, default FALSE
corners	Logical, Search for street corners using at least one of the street names, default TRUE
type	Character, How should results be returned can be "SF" or "Coordinates" or "Both", Default "SF"

Details

OTP will return a maximum of 10 results

Value

Returns a data.frame of SF POINTS or Coordinates of all the locations that match ‘query‘

See Also

Other routing: otp_isochrone(), otp_plan(), otp_routing_options(), otp_validate_routing_options()

Examples

```
## Not run:
locations <- otp_geocode(otpcon, "High Street")

## End(Not run)
```

---

otp_isochrone *Get the Isochrones from a location*

Description

Get the Isochrones from a location

Usage

```
otp_isochrone(
  otpcon = NA,
  fromPlace = NA,
  fromID = NULL,
  mode = "TRANSIT",
  date_time = Sys.time(),
  arriveBy = FALSE,
  maxWalkDistance = 1000,
  routingOptions = NULL,
  cutoffSec = c(600, 1200, 1800, 2400, 3000, 3600),
  ncores = 1,
  timezone = otpcon$timezone
)
```

Arguments

otpcon	OTP connection object produced by otp_connect()
fromPlace	Numeric vector, Longitude/Latitude pair, e.g. 'c(-0.134649,51.529258)', or 2 column matrix of Longitude/Latitude pairs, or sf data frame of POINTS
fromID	character vector same length as fromPlace
mode	character vector of one or more modes of travel valid values TRANSIT, WALK, BICYCLE, CAR, BUS, RAIL, default CAR. Not all combinations are valid e.g. c("WALK","BUS") is valid but c("WALK","CAR") is not.
date_time	POSIXct, a date and time, defaults to current date and time
arriveBy	Logical, Whether the trip should depart or arrive at the specified date and time, default FALSE

maxWalkDistance	maximum distance to walk in metres
routingOptions	named list passed to OTP see 'otp_routing_options()'
cutoffSec	Numeric vector, number of seconds to define the break points of each Isochrone
ncores	number of cores to use in parallel processing
timezone	character, timezone to use, default from otpcon

Details

Isochrones are maps of equal travel time, for a given location a map is produced showing how long it takes to reach each location.

Value

Returns a SF data.frame of POLYGONs

See Also

Other routing: otp_geocode(), otp_plan(), otp_routing_options(), otp_validate_routing_options()

Examples

```
## Not run:
isochrone1 <- otp_isochrone(otpcon, fromPlace = c(-0.1346, 51.5292))
isochrone2 <- otp_isochrone(otpcon,
  fromPlace = c(-0.1346, 51.5292),
  mode = c("WALK", "TRANSIT"), cutoffSec = c(600, 1200, 1800)
)

## End(Not run)
```

---

otp_make_config *Make Config Object*

Description

OTP can be configured using three json files 'otp-config.json', 'build-config.json', and 'router-config.json'. This function creates a named list for each config file and populates the defaults values.

Usage

```
otp_make_config(type)
```

Arguments

type	Which type of config file to create, "otp", "build", "router"

Details

For more details see: http://docs.opentripplanner.org/en/latest/Configuration

See Also

Other setup: otp_build_graph(), otp_check_java(), otp_dl_demo(), otp_dl_jar(), otp_setup(), otp_stop(), otp_validate_config(), otp_write_config()

Examples

```
{
  conf <- otp_make_config("build")
  conf <- otp_make_config("router")
}
```

---

otp_plan *Get get a route or routes from the OTP*

Description

This is the main routing function for OTP and can find single or multiple routes between 'fromPlace' and 'toPlace'.

Usage

```
otp_plan(
  otpcon = NA,
  fromPlace = NA,
  toPlace = NA,
  fromID = NULL,
  toID = NULL,
  mode = "CAR",
  date_time = Sys.time(),
  arriveBy = FALSE,
  maxWalkDistance = 1000,
  numItineraries = 3,
  routeOptions = NULL,
  full_elevation = FALSE,
  get_geometry = TRUE,
  ncores = 1,
  timezone = otpcon$timezone,
  distance_balance = FALSE,
  get_elevation = FALSE
)
```

Arguments

otpcon	OTP connection object produced by otp_connect()
fromPlace	Numeric vector, Longitude/Latitude pair, e.g. 'c(-0.134649,51.529258)', or 2 column matrix of Longitude/Latitude pairs, or sf data frame of POINTS with CRS 4326
toPlace	Numeric vector, Longitude/Latitude pair, e.g. 'c(-0.088780,51.506383)', or 2 column matrix of Longitude/Latitude pairs, or sf data frame of POINTS with CRS 4326
fromID	character vector same length as fromPlace
toID	character vector same length as toPlace
mode	character vector of one or more modes of travel valid values TRANSIT, WALK, BICYCLE, CAR, BUS, RAIL, default CAR. Not all combinations are valid e.g. c("WALK","BUS") is valid but c("WALK","CAR") is not.
date_time	POSIXct, a date and time, defaults to current date and time
arriveBy	Logical, Whether the trip should depart or arrive at the specified date and time, default FALSE
maxWalkDistance	Numeric passed to OTP in metres
numItineraries	The maximum number of possible itineraries to return
routeOptions	Named list of values passed to OTP use 'otp_route_options()' to make template object.
full_elevation	Logical, should the full elevation profile be returned, default FALSE
get_geometry	Logical, should the route geometry be returned, default TRUE, see details
ncores	Numeric, number of cores to use when batch processing, default 1, see details
timezone	Character, what timezone to use, see as.POSIXct, default is local timezone
distance_balance	Logical, use distance balancing, default false, see details
get_elevation	Logical, default FALSE, if true XYZ coordinates returned else XY coordinates returned.

Details

This function returns a SF data.frame with one row for each leg of the journey (a leg is defined by a change in mode). For transit, more than one route option may be returned and is indicated by the 'route_option' column. The number of different itineraries can be set with the 'numItineraries' variable.

Batch Routing

When passing a matrix or SF data frame object to fromPlace and toPlace 'otp_plan' will route in batch mode. In this case the 'ncores' variable will be used. Increasing 'ncores' will enable multicore routing, the max 'ncores' should be the number of cores on your system - 1.

Distance Balancing

When using multicore routing each task does not take the same amount of time. This can result in wasted time between batches. Distance Balancing sorts the routing by the euclidean distance

between fromPlace and toPlace before routing. This offers a small performance improvement of around five percent. As the original order of the inputs is lost fromID and toID must be provided.

Elevation

OTP supports elevation data and can return the elevation profile of the route if available. OTP returns the elevation profile separately from the XY coordinates, this means there is not direct match between the number of XY points and the number of Z points. OTP also only returns the elevation profile for the first leg of the route (this appears to be a bug). If 'get_elevation' is TRUE the otp_plan function matches the elevation profile to the XY coordinates to return an SF linestring with XYZ coordinates. If you require a more detailed elevation profile, the full_elevation parameter will return a nested data.frame with three columns. first and second are returned from OTP, while distance is the cumulative distance along the route and is derived from First.

Route Geometry

The 'get_geometry' provides the option to not return the route geometry, and just return the meta-data (e.g. journey time). This may be useful when creating an Origin:Destination matrix and also provides a small performance boost by reduced processing of geometries.

Value

Returns an SF data frame of LINESTRINGs

See Also

Other routing: `otp_geocode()`, `otp_isochrone()`, `otp_routing_options()`, `otp_validate_routing_options()`

Examples

```
## Not run:
otpcon <- otp_connect()
otp_plan(otpcon, c(0.1, 55.3), c(0.6, 52.1))
otp_plan(otpcon, c(0.1, 55.3), c(0.6, 52.1),
  mode = c("WALK", "TRANSIT")
)
otp_plan(otpcon, c(0.1, 55.3), c(0.6, 52.1),
  mode = "BICYCLE", arriveBy = TRUE,
  date_time = as.POSIXct(strptime("2018-06-03 13:30", "%Y-%m-%d %H:%M"))
)

## End(Not run)
```

---

otp_routing_options *Make routingOptions object*

Description

OTP supports a wide selection of routing options 'otp_plan()' accepts a named list of these options. This function produces an empty named list of valid options supported by both this package and OTP.

Usage

```
otp_routing_options()
```

Details

Supports almost all of the possible options in OTP 1.4. Note that some of the most popular option (mode, date, time, etc.) are set directly in 'otp_plan()'. If you want to permenaty set an option many are supported in the config files, see help on 'otp_make_config()'.

http://dev.opentripplanner.org/apidoc/1.4.0/resource_PlannerResource.html

See Also

Other routing: `otp_geocode()`, `otp_isochrone()`, `otp_plan()`, `otp_validate_routing_options()`

Examples

```
## Not run:
routingOptions <- otp_routing_options()
routingOptions$walkSpeed <- 1.5
routingOptions <- otp_validate_routing_options(routingOptions)

## End(Not run)
```

---

otp_setup *Set up an OTP instance.*

Description

OTP is run in Java and requires Java commands to be typed into the command line. The function allows the parameters to be defined in R and automatically passed to Java. This function sets up a local instance of OTP, for remote versions see documentation.

The function assumes you have run otp_build_graph()

Usage

```
otp_setup(
  otp = NULL,
  dir = NULL,
  memory = 2048,
  router = "default",
  port = 8080,
  securePort = 8081,
  analyst = FALSE,
  wait = TRUE
)
```

Arguments

otp	A character string, path to the OTP .jar file
dir	A character string, path to a directory containing the necessary files, see details
memory	A positive integer. Amount of memory to assign to the OTP in MB, default is 2048
router	A character for the name of the router to use, must be subfolder of dir/graphs, default "default". See vignettes for details.
port	A positive integer. Optional, default is 8080.
securePort	A positive integer. Optional, default is 8081.
analyst	Logical. Should the analyst features be loaded? Default FALSE
wait	Logical, Should R wait until OTP has loaded before running next line of code, default TRUE

Details

To run an OTP graph must have been created using otp_build_graph and the following files to be in the directory specified by the dir variable.

/graphs - A sub-directory

/default - A sub-directory with the name of the OTP router used in 'router' variable

graph.obj OTP graph

Value

This function does not return a value to R. If wait is TRUE R will wait until OTP is running (maximum of 5 minutes). After 5 minutes (or if wait is FALSE) the function will return R to your control, but the OTP will keep loading.

See Also

Other setup: otp_build_graph(), otp_check_java(), otp_dl_demo(), otp_dl_jar(), otp_make_config(), otp_stop(), otp_validate_config(), otp_write_config()

Examples

```
## Not run:
otp_setup(
  otp = "C:/otp/otp.jar",
  dir = "C:/data"
)
otp_setup(
  otp = "C:/otp/otp.jar",
  dir = "C:/data",
  memory = 5000,
  analyst = TRUE
)

## End(Not run)
```

`otp_stop` *Stop and OTP Instance*

Description

OTP is run in Java and requires Java commands to be typed into the command line. The function allows the parameters to be defined in R and automatically passed to Java. This function stops an already running OTP instance

Usage

```
otp_stop(warn = TRUE, kill_all = TRUE)
```

Arguments

`warn`	Logical, should you get a warning message
`kill_all`	Logical, should all Java instances be killed?

Details

The function assumes you have run otp_setup()

Value

This function return a message but no object

See Also

Other setup: `otp_build_graph()`, `otp_check_java()`, `otp_dl_demo()`, `otp_dl_jar()`, `otp_make_config()`, `otp_setup()`, `otp_validate_config()`, `otp_write_config()`

Examples

```
## Not run:
otp_stop(kill_all = FALSE)

## End(Not run)
```

otp_validate_config *Validate Config Object*

Description

Checks if the list of OTP configuration options is valid

Usage

```
otp_validate_config(config, type = attributes(config)$config_type)
```

Arguments

config	A named list made/modified from 'otp_make_config()'
type	type of config file

Details

Performs basic validity checks on class, max/min values etc as appropriate, some of more complex parameters are not checked. For more details see:

http://docs.opentripplanner.org/en/latest/Configuration http://dev.opentripplanner.org/javadoc/1.3.0/org/opentripplanner/rout

See Also

Other setup: otp_build_graph(), otp_check_java(), otp_dl_demo(), otp_dl_jar(), otp_make_config(), otp_setup(), otp_stop(), otp_write_config()

Examples

```
## Not run:
conf <- otp_make_config("build")
otp_validate_config(conf)

## End(Not run)
```

otp_validate_routing_options *Validate routingOptions object*

Description

OTP supports a wide selection of routing options 'otp_plan()' accepts a named list of these options. This function validates a named list of inputs and removes any empty inputs.

Usage

```
otp_validate_routing_options(opts)
```

Arguments

opts	a named list of options possibly from 'otp_routing_options()'

Details

Supports almost all of the possible options in OTP 1.4. Note that some of the most popular option (mode, date, time, etc.) are set directly in 'otp_plan()'. If you want to permenaty set an option many are supported in the config files, see help on 'otp_make_config()'. http://dev.opentripplanner.org/apidoc/1.4.0/resource_Plann

See Also

Other routing: otp_geocode(), otp_isochrone(), otp_plan(), otp_routing_options()

Examples

```
## Not run:
routingOptions <- otp_routing_options()
routingOptions$walkSpeed <- 1.5
routingOptions <- otp_validate_routing_options(routingOptions)

## End(Not run)
```

---

otp_write_config *Write config object as json file*

Description

Takes a config list produced by 'otp_make_config()' and saves it as json file for OTP

Usage

```
otp_write_config(config, dir = NULL, router = "default")
```

Arguments

config	A named list made/modified from 'otp_make_config()'
dir	Path to folder where data for OTP is to be stored
router	name of the router, default is "default", must be a subfolder of dir/graphs

See Also

Other setup: otp_build_graph(), otp_check_java(), otp_dl_demo(), otp_dl_jar(), otp_make_config(), otp_setup(), otp_stop(), otp_validate_config()

Examples

```
## Not run:
conf <- otp_make_config("build")
otp_write_config(conf, dir = tempdir())

## End(Not run)
```

Index

∗ **connect**
    otp_connect, 6
∗ **datasets**
    json_example_drive, 3
    json_example_long_drive, 3
    json_example_transit, 4
∗ **mulitmodal**
    opentripplanner-package, 2
∗ **opentripplanner**
    opentripplanner-package, 2
∗ **routing**
    opentripplanner-package, 2
    otp_geocode, 9
    otp_isochrone, 10
    otp_plan, 12
    otp_routing_options, 14
    otp_validate_routing_options, 18
∗ **setup**
    otp_build_graph, 4
    otp_check_java, 5
    otp_dl_demo, 7
    otp_dl_jar, 8
    otp_make_config, 11
    otp_setup, 15
    otp_stop, 17
    otp_validate_config, 18
    otp_write_config, 19
∗ **transport**
    opentripplanner-package, 2

json_example_drive, 3
json_example_long_drive, 3
json_example_transit, 4

opentripplanner
        *(*opentripplanner-package*)*, 2
opentripplanner-package, 2
otp_build_graph, 4, *5*, *7, 8*, *12*, *16–19*
otp_check_java, *5*, 5, *7, 8*, *12*, *16–19*
otp_connect, 6
otp_dl_demo, *5*, 7, *8*, *12*, *16–19*
otp_dl_jar, *5*, *7*, 8, *12*, *16–19*
otp_geocode, 9, *11*, *14, 15*, *19*
otp_isochrone, *10*, 10, *14, 15*, *19*
otp_make_config, *5*, *7, 8*, 11, *16–19*
otp_plan, *10, 11*, 12, *15*, *19*
otp_routing_options, *10, 11*, *14*, 14, *19*
otp_setup, *5*, *7, 8*, *12*, 15, *17–19*
otp_stop, *5*, *7, 8*, *12*, *16*, 17, *18, 19*
otp_validate_config, *5*, *7, 8*, *12*, *16, 17*, 18, *19*
otp_validate_routing_options, *10, 11*, *14, 15*, 18
otp_write_config, *5*, *7, 8*, *12*, *16–18*, 19