Package: valhallr (via r-universe)

September 9, 2024

Type Package

Title A Tidy Interface to the 'Valhalla' Routing Engine

Version 0.1.9000

Author Christopher Belanger

Maintainer Christopher Belanger <christopher.a.belanger@gmail.com>

Description An interface to the 'Valhalla' routing engine's application programming interfaces (APIs) for turn-by-turn routing, isochrones, and origin-destination analyses. Also includes several user-friendly functions for plotting outputs, and strives to follow ``tidy'' design principles. Please note that this package requires access to a running instance of 'Valhalla', which is open source and can be downloaded from <https://github.com/valhalla/valhalla>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports httr, purrr, jsonlite, dplyr, magrittr, tibble, tidyr, sf, leaflet, ggplot2, htmltools, stringr, ggspatial, geojsonio, rlang, Cairo

RoxygenNote 7.1.1

URL https://github.com/chris31415926535/valhallr

BugReports https://github.com/chris31415926535/valhallr/issues

Roxygen list(markdown = TRUE)

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository https://chris31415926535.r-universe.dev

RemoteUrl https://github.com/chris31415926535/valhallr

RemoteRef HEAD

RemoteSha 8b93576be15f3314b6b10d62c1e4f0f2d10526c1

Contents

`decode` *Decode Valhalla Route Shape*

Description

For point-to-point routing, Valhalla's API provides a route shapefile in a special ASCII-encoded format. This function takes an encoded string, decodes it, and returns the lat/lon coordinates as a tibble.

Usage

```
decode(encoded)
```

Arguments

`encoded`	An encoded shapefile in ASCII format from Valhalla's API.

Details

To map the results, see also `valhallr::map_trip()`.

Value

A tibble containing point locations in `lat` and `lon` columns.

`isochrone` *Generate Isochrones*

Description

An isochrone, also known as a service area, is a polygon that shows the area reachable from a starting point by traveling along a road network for a certain distance or time. This function provides an interface to the Valhalla routing engine's isochrone API. It lets you provide a starting point's latitude and longitude, a distance or time metric, and a vector of distances/times, and if it's successful it returns an sf-class tibble of polygons.

Usage

```
isochrone(
  from,
  costing = "pedestrian",
  contours = c(5, 10, 15),
  metric = "min",
  min_road_class = "residential",
  minimum_reachability = 500,
  hostname = "localhost",
  port = 8002
)
```

Arguments

`from`	A tibble containing one origin location in columns named `lat` and `lon`.
`costing`	The travel costing method: at present "auto", "bicycle", and "pedestrian" are supported.
`contours`	A numeric vector of values at which to produce the isochrones.
`metric`	Distance or time. Accepts parameters "min" and "km".
`min_road_class`	The minimum road classification Valhalla will consider. Defaults to `residential`.
`minimum_reachability`	The minimum number of nodes a candidate network needs to have before it is included.
`hostname`	Hostname or IP address of your Valhalla instance. Defaults to "localhost".
`port`	The port your Valhalla instance is monitoring. Defaults to 8002.

Details

More more information, please see Valhalla's API documentation:

- https://valhalla.readthedocs.io/en/latest/api/isochrone/api-reference/

Value

An sf/tibble object containing isochrone polygons.

Examples

```
## Not run:
library(valhallr)
# set up our departure point: the University of Ottawa
from <- test_data("uottawa")

# generate a set of isochrones for travel by bicycle
i <- valhallr::isochrone(from, costing = "bicycle")

# map the isochrones
map_isochrone(i)

## End(Not run)
```

`map_isochrone` *Generate maps of isochrones*

Description

This is a convenience function that takes the output of `valhallr::isochrone()` and generates either a static or interactive map.

Usage

```
map_isochrone(isochrone, method = "leaflet")
```

Arguments

`isochrone`	An isochrone sf object generated by `valhallr::isochrone()`.
`method`	The method used to map it. Two methods are supported: • "leaflet" produces an interactive HTML map using the Leaflet package. • "ggplot" produces a static map.

Value

A plot of the isochrones, either a a leaflet object or a ggplot object.

Examples

```
## Not run:
library(valhallr)
# set up our departure point: the University of Ottawa
from <- test_data("uottawa")

# generate a set of isochrones for travel by bicycle
i <- valhallr::isochrone(from, costing = "bicycle")

# map the isochrones
```

```
map_isochrone(i)

## End(Not run)
```

`map_trip` *Make a Map from a Trip*

Description

Make a Map from a Trip

Usage

```
map_trip(trip, method = "leaflet")
```

Arguments

`trip`	A trip response from `valhallr::route()`.
`method`	Which mapping service to use. Defaults to leaflet; also can use ggplot.

Value

A map object, either leaflet or ggplot.

Examples

```
## Not run:
  library(valhallr)
  # set up origin and destination data
  from <- test_data("uottawa")
  to <- test_data("cdntirecentre")

  # calculate the trip
  trip <- route(from = from, to = to)

  # show overall trip information
  print_trip(trip, all_details = FALSE)

  # make an interactive map of the trip using the leaflet package
  map_trip(trip, method = "leaflet")

## End(Not run)
```

od_table *Generate Tidy Origin-Destination Data using Valhalla*

Description

This function creates a tidy (i.e. long) table of origin-destination trip data using the Valhalla routing engine. For a set of o origins and d destinations, it returns a tibble with (o x d) rows with the travel distance and time between each pair. It can handle several different travel modes and routing options.

This function is a user-friendly wrapper around`valhalla::sources_to_targets()`, which calls the Valhalla API directly. `sources_to_targets()` offers finer- grained control over API options, and so this latter function may be more useful for advanced users.

Notable features of `od_matrix()`:

- You can specify human-readable indices with `from_id_col` and `to_id_col`. (Valhalla's API only returns zero-indexed integer identifiers.)
- You can specify a `batch_size` to break computation into several smaller API calls, to prevent your Valhalla instance from running out of memory. This seems especially important for pedestrian routing, where I've sometimes needed to use a batch size as small as 5.

Usage

```
od_table(
  froms,
  from_id_col,
  tos,
  to_id_col,
  costing = "auto",
  batch_size = 100,
  minimum_reachability = 500,
  verbose = FALSE,
  exclude_polygons = NA,
  hostname = "localhost",
  port = 8002
)
```

Arguments

`froms`	A tibble containing origin locations in columns named `lat` and `lon`, and an optional column with human-readable names.
`from_id_col`	The name of the column in `froms` that contains human-readable names.
`tos`	A tibble containing destination locations in columns named `lat` and `lon`, and an optional column with human-readable names.
`to_id_col`	The name of the column in `tos` that contains human-readable names.

costing	The travel costing method: at present "auto", "bicycle", and "pedestrian" are supported.
batch_size	The number of origin points to process per API call.
minimum_reachability	The minimum number of nodes a candidate network needs to have before it is included. Try increasing this value (e.g. to 500) if Valhalla is getting stuck in small disconnected road networks.
verbose	Boolean. Defaults to FALSE. If TRUE, it will provide updates on on the batching process (if applicable).
exclude_polygons	A tibble or list of tibbles with `lat` and `lon` columns defining polygons to be avoided.
hostname	Hostname or IP address of your Valhalla instance. Defaults to "localhost".
port	The port your Valhalla instance is monitoring. Defaults to 8002.

Value

A tibble showing the trip distances and times from each origin to each named destination.

Examples

```
## Not run:
library(dplyr)
library(valhallr)
# set up our inputs
origins <- bind_rows(test_data("parliament"), test_data("uottawa"), test_data("cntower"))
destinations <- bind_rows(test_data("cdntirecentre"), test_data("parliament"))

# generate a tidy origin-destination table
od <- od_table (froms = origins,
                from_id_col = "name",
                tos = destinations,
                to_id_col = "name",
                costing = "auto",
                batch_size = 100,
                minimum_reachability = 500)

## End(Not run)
```

---

print_trip *Print Trip Summary and Turn-By-Turn Directions*

Description

Print Trip Summary and Turn-By-Turn Directions

Usage

```
print_trip(trip, all_details = FALSE)
```

Arguments

trip	A trip response from valhallr::route().
all_details	Boolean. Should we print each turn-by-turn instruction along with an overall summary?

Value

The input trip object, invisibly.

Examples

```
## Not run:
  library(valhallr)
  # set up origin and destination data
  from <- test_data("uottawa")
  to <- test_data("cdntirecentre")

  # calculate the trip
  trip <- route(from = from, to = to)

  # show overall trip information
  print_trip(trip, all_details = FALSE)

  # make an interactive map of the trip using the leaflet package
  map_trip(trip, method = "leaflet")

## End(Not run)
```

---

route *Point-to-Point Routing with Valhalla*

Description

This function calls Valhalla's route API to return turn-by-turn directions from one origin to one destination. Several costing methods are supported, and there are parameters that let you give custom options to Valhalla. **Please note that this function requires access to a running instance of Valhalla.**

Usage

```
route(
  from = NA,
  to = NA,
  costing = "auto",
```

```
  unit = "kilometers",
  from_search_filter = list(max_road_class = "motorway", min_road_class =
    "residential"),
 to_search_filter = list(max_road_class = "motorway", min_road_class = "residential"),
  minimum_reachability = 50,
  costing_options = list(),
  exclude_polygons = NA,
  hostname = "localhost",
  port = 8002
)
```

Arguments

`from`	A tibble containing one origin location in columns named `lat` and `lon`.
`to`	A tibble containing one destination location in columns named `lat` and `lon`.
`costing`	The travel costing method. Values "auto", "bicycle", and "pedestrian" all work.
`unit`	Distance measurement units. Defaults to "kilometres".
`from_search_filter`	A named list of options provided to Valhalla API. Defaults set a maximum road class ("motorway", the highest) and minimum road class ("residential", which is one above the lowest, "service_other"). See API documentation for details.
`to_search_filter`	A named list of options provided to Valhalla API. Defaults set a maximum road class ("motorway", the highest) and minimum road class ("residential", which is one above the lowest, "service_other"). See API documentation for details.
`minimum_reachability`	The minimum number of nodes a candidate network needs to have before it is included. Try increasing this value (e.g. to 500) if Valhalla is getting stuck in small disconnected road networks.
`costing_options`	A named list of options provided to the Valhalla API that affect route costing, e.g. willingness to travel on highways or through alleys. See API documentation for details.
`exclude_polygons`	A tibble or list of tibbles with `lat` and `lon` columns defining polygons to be avoided.
`hostname`	Hostname or IP address of your Valhalla instance. Defaults to "localhost".
`port`	The port your Valhalla instance is monitoring. Defaults to 8002.

Details

For more details, please check the Valhalla API documentation here:

- https://valhalla.readthedocs.io/en/latest/api/turn-by-turn/api-reference/

Value

A trip object.

Examples

```
## Not run:
  library(valhallr)
  # set up origin and destination data
  from <- test_data("uottawa")
  to <- test_data("cdntirecentre")

  # calculate the trip
  trip <- route(from = from, to = to)

  # show overall trip information
  print_trip(trip, all_details = FALSE)

  # make an interactive map of the trip using the leaflet package
  map_trip(trip, method = "leaflet")

## End(Not run)
```

`sf_to_latlon` *Convert* `sf` *objects to* **valhallr***-friendly tibbles*

Description

This function converts simple feature (sf) POINT objects to tidy tibbles that you can feed to **valhallr** functions. This is handy if you're working with a lots of geospatial data and would like to run a set of locations through, for example, `valhallr::od_table()`.

Usage

```
sf_to_latlon(data, output_crs = "WGS84")
```

Arguments

`data`	A simple feature collection with geometry class POINT.
`output_crs`	The desired output coordinate reference system (CRS). Defaults to WGS84.

Value

A non-sf tibble with new columns `lat` and `lon` containing latitudes and longitudes respectively that can be fed into other **valhallr** functions like `valhallr::od_table()`.

sources_to_targets *Source-to-Targets Origin/Destination Matrices with Valhalla*

Description

This function creates a tidy (i.e. long) table of origin-destination trip data using the Valhalla routing engine. For a set of o origins and d destinations, it returns a tibble with (o x d) rows with the travel distance and time between each pair. It can handle several different travel modes and routing options. **Please note that this function requires access to a running instance of Valhalla.**

This function provides fine-grained control over Valhalla's API options.

- For a user-friendly function, see the function `valhallr::od_table()`.
- For details about the API, see Valhalla's documentation here: https://valhalla.readthedocs.io/en/latest/api/matrix/api-reference/

Usage

```
sources_to_targets(
  froms,
  tos,
  costing = "auto",
  from_search_filter = list(max_road_class = "motorway", min_road_class =
    "residential"),
 to_search_filter = list(max_road_class = "motorway", min_road_class = "residential"),
  minimum_reachability = 50,
  costing_options = list(),
  exclude_polygons = NA,
  hostname = "localhost",
  port = 8002
)
```

Arguments

`froms`	A tibble containing origin locations in columns named `lat` and `lon`.
`tos`	A tibble containing destination locations in columns named `lat` and `lon`.
`costing`	The travel costing method: at present "auto", "bicycle", and "pedestrian" are supported.
`from_search_filter`	A named list of options provided to Valhalla API. Defaults set a maximum road class ("motorway", the highest) and minimum road class ("residential", which is one above the lowest, "service_other"). See API documentation for details.
`to_search_filter`	A named list of options provided to Valhalla API. Defaults set a maximum road class ("motorway", the highest) and minimum road class ("residential", which is one above the lowest, "service_other"). See API documentation for details.

`minimum_reachability`
: The minimum number of nodes a candidate network needs to have before it is included. Try increasing this value (e.g. to 500) if Valhalla is getting stuck in small disconnected road networks.

`costing_options`
: A named list of options provided to the Valhalla API that affect route costing, e.g. willingness to travel on highways or through alleys. See API documentation for details.

`exclude_polygons`
: A tibble or list of tibbles with `lat` and `lon` columns defining polygons to be avoided.

`hostname`
: Hostname or IP address of your Valhalla instance. Defaults to "localhost".

`port`
: The port your Valhalla instance is monitoring. Defaults to 8002.

Value

A tibble showing the trip distances and times from each origin to each destination.

Examples

```
## Not run:
# NOTE: Assumes an instance of Valhalla is running on localhost:8002.
library(dplyr)
library(valhallr)
froms <- bind_rows(test_data("parliament"), test_data("uottawa"))
tos <- bind_rows(test_data("cdntirecentre"), test_data("parliament"))
st <- sources_to_targets(froms, tos)

## End(Not run)
```

---

test_data *Get Lat/Lon Coordinates for Testing*

Description

This function gives quick access to lat/lon coordinates for a few points around Ontario for testing and benchmarking purposes.

Usage

```
test_data(dataset = NA)
```

Arguments

`dataset`
: The name of a test dataset. By default, and if an unknown input is given, it returns all values.

Value

A tibble with one or more location names, latitudes, and longitudes.

Index