Package: traderoutes (via r-universe)

November 9, 2026

Title Compute International Trade Routes

Version 0.0.3

Description Provides tools to query border relationships between countries and to compute weighted shortest-path trade routes on a worldwide transportation network combining roads, shipping lanes, and ports. Border queries use a precomputed adjacency graph; route computation supports per-country and per-country-pair cost overrides for roads, ports, and land borders.

License MIT + file LICENSE

URL https://github.com/milonemario/traderoutes, https://milonemario.r-universe.dev/traderoutes

BugReports https://github.com/milonemario/traderoutes/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports igraph

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libglpk-dev libxml2-dev

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

Date/Publication 2026-11-09 19:10:58 UTC

RemoteUrl https://github.com/milonemario/traderoutes

RemoteRef HEAD

RemoteSha 05c6a196e6e0346635e63b240598ed3f2adb4eef

RemoteSubdir traderoutes-r

Contents

traderoutes-package . . . 2
are_neighbors . . . 3
cities_to_kml . . . 3
countries . . . 4
get_common_neighbors . . . 4
get_countries . . . 5
get_great_circle . . . 6
get_main_city . . . 6
get_neighbors . . . 7
get_path . . . 7
get_pattern . . . 9
get_second_neighbors . . . 10
get_trade_route_great_circle . . . 10
ports_to_kml . . . 11
road_distance . . . 11
roads_to_kml . . . 12
sea_distance . . . 12
seas_to_kml . . . 13
to_kml . . . 13
transits_i . . . 13
transits_j . . . 14

Index **15**

`traderoutes-package` *traderoutes: Compute International Trade Routes*

Description

Provides tools to query border relationships between countries and to compute weighted shortest-path trade routes on a worldwide transportation network combining roads, shipping lanes, and ports.

Details

Before installing or testing the package, make sure the generated RDS graph files are present under `inst/extdata/`.

Border functions

`are_neighbors()`, `get_neighbors()`, `get_second_neighbors()`, `get_common_neighbors()`

Route functions

`countries()`, `get_main_city()`, `get_great_circle()`, `get_path()`, `get_pattern()`, `get_countries()`, `road_distance()`, `sea_distance()`, `transits_i()`, `transits_j()`

Author(s)

Maintainer: Mario Milone <milonemario@gmail.com>

Authors:

- Mario Milone <milonemario@gmail.com>

---

are_neighbors *Check whether two countries share a land border*

Description

Check whether two countries share a land border

Usage

```
are_neighbors(country1, country2)
```

Arguments

country1	ISO 3-letter code of the first country (e.g. "FRA").
country2	ISO 3-letter code of the second country (e.g. "ITA").

Value

TRUE if the countries are direct neighbors, FALSE otherwise.

Examples

```
## Not run:
are_neighbors("FRA", "ITA")   # TRUE
are_neighbors("FRA", "USA")   # FALSE

## End(Not run)
```

---

cities_to_kml *Export city locations to KML*

Description

Export city locations to KML

Usage

```
cities_to_kml(filename, all_cities = FALSE)
```

Arguments

filename	Output KML path.
all_cities	If FALSE, export only main cities.

---

countries *Return all countries present in the transportation network*

Description

A country is included if it has at least one city vertex (vtype = "city") in the full graph.

Usage

```
countries()
```

Value

Sorted character vector of ISO 3-letter country codes.

Examples

```
## Not run:
cs <- countries()
"FRA" %in% cs   # TRUE

## End(Not run)
```

---

get_common_neighbors *Return countries that border both of two given countries*

Description

Return countries that border both of two given countries

Usage

```
get_common_neighbors(country1, country2)
```

Arguments

country1	ISO 3-letter code of the first country.
country2	ISO 3-letter code of the second country.

Value

Character vector of ISO3 codes of countries neighboring both inputs. Returns `character(0)` if no common neighbors exist.

Examples

```
## Not run:
get_common_neighbors("FRA", "ITA")   # "CHE"

## End(Not run)
```

`get_countries` *Return country segments traversed along a path*

Description

Each element of the returned list contains the ISO3 country code and the distance (km) traveled within that segment.

Usage

```
get_countries(path, with_sea = FALSE, clean_dist = NULL)
```

Arguments

`path`	List with elements `vpath` and `epath`, as returned by `get_path()`.
`with_sea`	Logical. If `TRUE`, sea segments are included with country code `"sea"`. Default `FALSE`.
`clean_dist`	Numeric or `NULL`. If set, iteratively removes intermediate country segments shorter than `clean_dist` km, keeping origin and destination. Default `NULL`.

Value

A list of named lists, each with elements `country` (character) and `dist` (numeric, km), in traversal order.

Examples

```
## Not run:
path <- get_path(get_main_city("FRA"), get_main_city("POL"), road_cost = 7)
sapply(get_countries(path), `[[`, "country")

## End(Not run)
```

get_great_circle *Great-circle distance between two countries' main cities*

Description

Great-circle distance between two countries' main cities

Usage

```
get_great_circle(country1, country2)
```

Arguments

country1	ISO 3-letter code of the origin country.
country2	ISO 3-letter code of the destination country.

Value

Great-circle distance in kilometres (numeric scalar).

Examples

```
## Not run:
round(get_great_circle("FRA", "ITA"), 2)   # 1109.56

## End(Not run)
```

get_main_city *Return the main city vertex index for a country*

Description

The main city is the vertex with `vtype = "city"` and `maincity = "1"` (or `1`) for the specified country. It is used as the default path endpoint and for great-circle distance calculations.

Usage

```
get_main_city(country)
```

Arguments

country	ISO 3-letter code of the country (e.g. `"FRA"`).

Value

Integer vertex index, or `NA` if no main city is found. A warning is issued when none or more than one main city is found.

Examples

```
## Not run:
v <- get_main_city("FRA")
igraph::vertex_attr(g, "name", v)

## End(Not run)
```

`get_neighbors` *Return neighboring countries of a given country*

Description

Return neighboring countries of a given country

Usage

```
get_neighbors(country)
```

Arguments

`country`	ISO 3-letter code of the country (e.g. `"FRA"`).

Value

Character vector of ISO3 codes of countries sharing a land border with `country`. Returns `character(0)` if the country is not in the graph.

Examples

```
## Not run:
get_neighbors("USA")   # c("CAN", "MEX")

## End(Not run)
```

`get_path` *Compute the weighted shortest-path trade route between two vertices*

Description

Applies configurable cost weights to graph edges and returns the shortest path. Costs are applied in this order:

Usage

```
get_path(
  v1,
  v2,
  road_cost = 1,
  road_cost_countries = list(),
  port_cost = 1,
  port_cost_countries = list(),
  border_cost = 0,
  border_cost_countries = list()
)
```

Arguments

v1	Integer vertex index of the origin (from get_main_city()).
v2	Integer vertex index of the destination.
road_cost	Multiplicative cost for road edges. Default 1.
road_cost_countries	Named list of per-country road cost overrides, e.g. list(FRA = 2, NLD = 4). Default list().
port_cost	Additive cost for land-to-sea transitions. Default 1.
port_cost_countries	Named list of per-country port cost overrides. Default list().
border_cost	Additive cost for crossing a country border. Default 0.
border_cost_countries	Named list of per-country-pair border cost overrides. Keys are "C1-C2" strings, e.g. list("FRA-DEU" = 500). Default list().

Details

1. All edges are initialized to their physical distance (wdist = dist).
2. **Port edges** (etype = "road_sea"): wdist = dist + port_cost. Per-country overrides in port_cost_countries take precedence.
3. **Border edges** (etype in c("road_border", "sea_border")): wdist = dist + border_cost. Per-pair overrides in border_cost_countries take precedence (keys are "C1-C2" strings).
4. **Road edges** (etype = "road"): wdist = dist * road_cost. Per-country overrides in road_cost_countries take precedence.

Value

A named list with:

epath	Integer vector of edge indices along the path.
vpath	Integer vector of vertex indices along the path.

Examples

```
## Not run:
v_fra <- get_main_city("FRA")
v_nld <- get_main_city("NLD")
path  <- get_path(v_fra, v_nld, road_cost = 7, port_cost = 1e4)
get_pattern(path)

## End(Not run)
```

`get_pattern` *Return the vertex-type pattern of a trade route path*

Description

Compresses consecutive vertices of the same type into a single character: `c` = city, `r` = road, `p` = port, `s` = sea. Intermediate city vertices (not the first or last vertex) are treated as road.

Usage

```
get_pattern(path)
```

Arguments

`path`	List with element `vpath` (integer vector), as returned by `get_path()`.

Value

Character string, e.g. `"crc"` (land route) or `"csrc"` (sea route).

Examples

```
## Not run:
path <- get_path(get_main_city("FRA"), get_main_city("DEU"), road_cost = 7)
get_pattern(path)   # "crc"

## End(Not run)
```

get_second_neighbors *Return countries reachable in exactly two border hops*

Description

Returns countries that are two hops away from `country` in the border graph, excluding `country` itself and its direct neighbors.

Usage

```
get_second_neighbors(country)
```

Arguments

`country`	ISO 3-letter code of the country.

Value

Character vector of ISO3 codes of second-degree neighbors, in traversal order. Returns `character(0)` if the country is not found.

Examples

```
## Not run:
"PRT" %in% get_second_neighbors("FRA")   # TRUE

## End(Not run)
```

get_trade_route_great_circle *Return a list with country codes and the great-circle distance between them*

Description

Return a list with country codes and the great-circle distance between them

Usage

```
get_trade_route_great_circle(country1, country2)
```

Arguments

`country1`	ISO 3-letter code of the origin country.
`country2`	ISO 3-letter code of the destination country.

Value

A named list with `iso_i`, `iso_j`, and `dist`.

---

ports_to_kml *Export port locations to KML*

Description

Export port locations to KML

Usage

```
ports_to_kml(filename)
```

Arguments

`filename`	Output KML path.

---

road_distance *Total road distance along a path*

Description

Sums the `dist` attribute of all edges with `etype` in `c("road", "road_sea")`.

Usage

```
road_distance(path)
```

Arguments

`path`	List with element epath, as returned by `get_path()`.

Value

Numeric scalar, distance in km.

roads_to_kml *Export the road network to KMZ*

Description

Export the road network to KMZ

Usage

```
roads_to_kml(filename)
```

Arguments

filename	Output path prefix. .kmz is appended.

sea_distance *Total sea distance along a path*

Description

Sums the dist attribute of all edges with etype in c("sea", "sea_border").

Usage

```
sea_distance(path)
```

Arguments

path	List with element epath, as returned by get_path().

Value

Numeric scalar, distance in km.

seas_to_kml *Export the sea network, including sea-border edges, to KMZ*

Description

Export the sea network, including sea-border edges, to KMZ

Usage

```
seas_to_kml(filename)
```

Arguments

filename	Output path prefix. .kmz is appended.

to_kml *Export a trade route path to KML*

Description

Export a trade route path to KML

Usage

```
to_kml(path, filename)
```

Arguments

path	Path as returned by get_path().
filename	Output KML path.

transits_i *Transit countries from the origin side of a path*

Description

For **land-only** routes: all intermediate countries between origin and destination. For routes **with sea**: countries between the origin and the first sea segment.

Usage

```
transits_i(path, clean_dist = NULL)
```

Arguments

path	List as returned by get_path().
clean_dist	Numeric or NULL. Passed to get_countries() to remove very short country segments. Default NULL.

Value

Character vector of ISO3 transit country codes (may be empty).

Examples

```
## Not run:
path <- get_path(get_main_city("CHE"), get_main_city("POL"), road_cost = 7)
transits_i(path)

## End(Not run)
```

---

transits_j *Transit countries from the destination side of a path*

Description

Equivalent to transits_i() applied to the reversed path.

Usage

```
transits_j(path, clean_dist = NULL)
```

Arguments

path	List as returned by get_path().
clean_dist	Numeric or NULL. Default NULL.

Value

Character vector of ISO3 transit country codes (may be empty).

Index

are_neighbors, 3
are_neighbors(), *2*

cities_to_kml, 3
countries, 4
countries(), *2*

get_common_neighbors, 4
get_common_neighbors(), *2*
get_countries, 5
get_countries(), *2*, *14*
get_great_circle, 6
get_great_circle(), *2*
get_main_city, 6
get_main_city(), *2*, *8*
get_neighbors, 7
get_neighbors(), *2*
get_path, 7
get_path(), *2*, *5*, *9*, *11–14*
get_pattern, 9
get_pattern(), *2*
get_second_neighbors, 10
get_second_neighbors(), *2*
get_trade_route_great_circle, 10

ports_to_kml, 11

road_distance, 11
road_distance(), *2*
roads_to_kml, 12

sea_distance, 12
sea_distance(), *2*
seas_to_kml, 13

to_kml, 13
traderoutes *(*traderoutes-package*)*, 2
traderoutes-package, 2
transits_i, 13
transits_i(), *2*, *14*
transits_j, 14
transits_j(), *2*