Package: traveltime (via r-universe)

October 31, 2026

Title Calculate Travel Times Over Space

Version 1.0.0

Description Calculate travel time across a landscape from a set of user-specified locations, over a friction surface describing the time taken to cross each cell. Implements the least-cost travel time approach of Weiss et al. (2018) <doi:10.1038/nature25181> and Weiss et al. (2020) <doi:10.1038/s41591-020-1059-1>, and provides convenient access to their global walking and motorised friction surfaces via the 'malariaAtlas' package. Travel times are computed with 'terra'. See Ryan et al. (2025) <doi:10.12688/gatesopenres.16356.1> for further details.

License MIT + file LICENSE

Copyright The 'stations' data contains information from the Land Transport Authority of Singapore, made available under the Singapore Open Data Licence version 1.0; see ?stations.

Imports cli, malariaAtlas, terra

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

Suggests geodata, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), tidyterra

Config/testthat/edition 3

Depends R (>= 4.1.0)

LazyData true

URL https://idem-lab.github.io/traveltime/, https://github.com/idem-lab/traveltime

BugReports https://github.com/idem-lab/traveltime/issues

VignetteBuilder knitr

NeedsCompilation no

Author Gerry Ryan [aut, cre] (ORCID: <https://orcid.org/0000-0003-0183-7630>), Daniel Weiss [aut] (ORCID: <https://orcid.org/0000-0002-6175-5648>), Nicholas Tierney [ctb] (ORCID: <https://orcid.org/0000-0003-1460-8722>)

Maintainer Gerry Ryan <gerry.ryan@thekids.org.au>

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

Date/Publication 2026-10-31 17:50:40 UTC

RemoteUrl https://github.com/cran/traveltime

RemoteRef HEAD

RemoteSha a9e8f75ca570d3b21135c82fdf0ccc0c466a62cc

Contents

calculate_travel_time 2
ext_from_terra 5
ext_matrix 5
ext_vect_to_matrix 6
get_friction_surface 7
stations 8

Index **10**

`calculate_travel_time` *Calculate travel time*

Description

Calculate the travel time from a set of points over a friction surface.

Usage

```
calculate_travel_time(
  friction_surface,
  points,
  filename = NULL,
  overwrite = FALSE
)
```

Arguments

`friction_surface`
: A `SpatRaster` friction surface layer in resistance units. See Details.

points	A two-column matrix, data.frame, (including tibble types) with longitude (x) in the first column and latitude (y) in the second, or a SpatVector, in the same coordinate reference system as friction_surface. Points that fall outside friction_surface are dropped with a warning; if none fall within it, an error is raised.
filename	character. Output file name with extension suitable for terra::writeRaster. If NULL (default), output will be returned in memory
overwrite	logical. If TRUE filename is overwritten. If FALSE (default) and filename already exists, the existing file is returned with a warning and no calculation is performed.

Details

Implements methods from Weiss et al. 2018, 2020 to calculate travel time from given locations over a friction surface.

Over large areas this function can require significant RAM and will be slow.

Pre-prepared walking or motorised friction surfaces can be obtained with get_friction_surface(). Users can also provide their own friction surface. This surface must be in resistance units (1/conductance), e.g. minutes/meter. See Van Etten 2017.

Note on this implementation: as of package version 1. Earlier versions of this package used the raster and gdistance packages. These dependencies have been removed and now instead rely exclusively on terra for spatial data handling. The two approaches use the same cost model (eight-direction movement, edge cost distance * mean(friction)) and produce effectively the same result. terra::costDist() computes true geodesic inter-cell distances, so the new approach is *more* accurate than the old gdistance's distance approximation away from the equator (the results are identical at the equator and for projected coordinate systems). The remaining difference is a boundary convention at the origins: gdistance includes half of the starting cell's own friction when leaving a point, whereas terra::costDist measures cost to the border of the origin cells and omits it. This appears as a small, near-constant offset (about half a cell of friction) close to the origin points. See the validation article https://idem-lab.github.io/traveltime/articles/costdist-validation.html for the full comparison.

Citations:

D. J. Weiss, A. Nelson, C. A. Vargas-Ruiz, K. Gligoric, S., Bavadekar, E. Gabrilovich, A. Bertozzi-Villa, J. Rozier, H. S. Gibson, T., Shekel, C. Kamath, A. Lieber, K. Schulman, Y. Shao, V. Qarkaxhija, A. K. Nandi, S. H. Keddie, S. Rumisha, P. Amratia, R. Arambepola, E. G. Chestnutt, J. J. Millar, T. L. Symons, E. Cameron, K. E. Battle, S. Bhatt, and P. W. Gething. Global maps of travel time to healthcare facilities. (2020) Nature Medicine. doi:10.1038/s4159102010591

D. J. Weiss, A. Nelson, H.S. Gibson, W. Temperley, S. Peedell, A. Lieber, M. Hancher, E. Poyart, S. Belchior, N. Fullman, B. Mappin, U. Dalrymple, J. Rozier, T.C.D. Lucas, R.E. Howes, L.S. Tusting, S.Y. Kang, E. Cameron, D. Bisanzio, K.E. Battle, S. Bhatt, and P.W. Gething. A global map of travel time to cities to assess inequalities in accessibility in 2015. (2018). Nature. doi:10.1038/nature25181

van Etten, J. (2017). R package gdistance: Distances and routes on geographical grids. Journal of Statistical Software doi:10.18637/jss.v076.i13

Value

A SpatRaster of travel time (in the time units of friction_surface, e.g. minutes) from each cell to the nearest of points, with a single layer named "travel_time". Cells that cannot be reached from any of points (e.g. because they are separated from them by NA cells in friction_surface) are NA.

Examples

```
# a small synthetic friction surface (minutes per metre), so this example
# runs without an internet connection
friction_surface <- terra::rast(
  nrows = 50,
  ncols = 50,
  xmin = 111,
  xmax = 112,
  ymin = 0,
  ymax = 1,
  crs = "EPSG:4326",
  vals = 0.01
)

from_here <- data.frame(
  x = c(111.2, 111.9),
  y = c(0.2, 0.35)
)

calculate_travel_time(
  friction_surface = friction_surface,
  points = from_here
)


# a real motorised friction surface, downloaded from the Malaria Atlas
# Project (requires an internet connection)
motor_surface <- get_friction_surface(
  surface = "motor2020",
  extent = c(111, 112, 0, 1)
)

if (!is.null(motor_surface)) {
  calculate_travel_time(
    friction_surface = motor_surface,
    points = from_here
  )
}
```

ext_from_terra *Extent from* SpatRaster *or* SpatVector

Description

Formats spatial extent for use in get_friction_surface.

Usage

```
ext_from_terra(r)
```

Arguments

r	terra::SpatRaster or terra::SpatVector

Value

2x2 matrix

Examples

```
library(terra)
r <- terra::rast(
    extent = terra::ext(c(111, 112, 0, 1))
  )

ext_from_terra(r)
```

ext_matrix *Convert extent to matrix*

Description

This function allows get_friction_surface() to accept the argument extent as a vector, 2x2 matrix, SpatVector, or SpatRaster and extracts and converts into the annoyingly specific format necessary to download the friction surface. See details in ?get_friction_surface

Usage

```
ext_matrix(extent)
```

Arguments

extent	vector of length 4, 2x2 matrix, SpatExtent, SpatVector, or SpatRaster

Value

2x2 `matrix` with column names "x" and "y" and row names "min" and "max"

Examples

```
# vector/double
x <- c(111, 112, 0, 1)
ext_matrix(x)

# SpatExtent
y <-  terra::ext(x)
ext_matrix(y)

# SpatRaster
r <- terra::rast(extent = y)
ext_matrix(r)

# SpatVector
v <- terra::vect(y)
ext_matrix(v)
```

`ext_vect_to_matrix` *Extent vector to matrix*

Description

Extent vector to matrix

Usage

```
ext_vect_to_matrix(x)
```

Arguments

`x`	`numeric` length 4, consisting of `c(xmin, xmax, ymin, ymax)` dimensions of extent

Value

2x2 `matrix`

Examples

```
ext_vect_to_matrix(c(111,112,0, 1))
```

`get_friction_surface` *Get friction surface*

Description

Wrapper function to download friction surfaces via `malariaAtlas::getRaster()`.

Usage

```
get_friction_surface(
  surface = c("motor2020", "walk2020"),
  filename = NULL,
  overwrite = FALSE,
  extent = NULL
)
```

Arguments

`surface`	`"motor2020"` or `"walk2020"`.
`filename`	`character`. File name for output layer.
`overwrite`	`logical`. If `TRUE` `filename` is overwritten. If `FALSE` (default) and `filename` already exists, the existing file is returned with a warning and nothing is downloaded.
`extent`	Spatial extent as one of: • a numeric vector specifying `c(xmin, xmax, ymin, ymax)`, • a `terra` `SpatExtent`, `SpatVector`, or `SpatRaster` from which the extent will be taken, or • or 2x2 `matrix` See details

Details

Convenience wrapper to `malariaAtlas::getRaster()` to access motorised and walking travel friction layers per Weiss et al. 2020, that adds safety to check existing files before download. Surfaces can be downloaded directly from: https://malariaatlas.org/project-resources/accessibility-to-healthcare/.

Here —

- `surface = "motor2020"` will download `"Accessibility__202001_Global_Motorized_Friction_Surface"`, and
- `surface = "walk2020"` will download `"Accessibility__202001_Global_Walking_Only_Friction_Surface"`.

D. J. Weiss, A. Nelson, C. A. Vargas-Ruiz, K. Gligoric, S., Bavadekar, E. Gabrilovich, A. Bertozzi-Villa, J. Rozier, H. S. Gibson, T., Shekel, C. Kamath, A. Lieber, K. Schulman, Y. Shao, V. Qarkaxhija, A. K. Nandi, S. H. Keddie, S. Rumisha, P. Amratia, R. Arambepola, E. G. Chestnutt, J. J.

Millar, T. L. Symons, E. Cameron, K. E. Battle, S. Bhatt, and P. W. Gething. Global maps of travel time to healthcare facilities. (2020) Nature Medicine. doi:10.1038/s4159102010591

`extent` is formatted and then passed through to `malariaAtlas::getRaster()` as a 2x2 matrix. It is converted into a matrix using `ext_matrix`. The `matrix` format used is as returned by `sf::st_bbox()` — the first column has the minimum, the second the maximum values; rows 1 & 2 represent the x & y dimensions respectively: `matrix(c("xmin", "ymin", "xmax", "ymax"), nrow = 2, ncol = 2, dimnames = list(c("x", "y"), c("min", "max")))`. `extent` must be supplied.

Troubleshooting: this function requires an internet connection. If the download fails (e.g. a warning `Failed to connect to MAP geoserver`), a warning is given and `NULL` is returned. This is an issue with fetching the friction surface using `malariaAtlas::getRaster()`, and not an issue with `traveltime` itself. The server may be down or there may be an issue with that package.

Value

A `SpatRaster` friction surface with a single layer named `"friction_surface"`, or `NULL` (with a warning) if the download fails.

Examples

```
# requires an internet connection
# for more examples of passing extent types see ?ext_matrix

ext <- c(111, 112, 0, 1)

get_friction_surface(
  surface = "motor2020",
  extent = ext
)
```

---

stations *Singapore MRT and LRT data*

Description

Locations of all MRT and LRT station exits in Singapore.

Usage

```
stations
```

Format

A `matrix` with 563 rows and 2 columns:

x longitude

y latitude

Licence

Contains information from the dataset "LTA MRT Station Exit (GEOJSON)" accessed on the 10th of December 2024 from data.gov.sg (https://data.gov.sg/datasets/d_b39d3a0871985372d7e1637193335da5/view), which is made available under the terms of the Singapore Open Data Licence version 1.0 https://data.gov.sg/open-data-licence.

Source

Land Transport Authority. (2019). LTA MRT Station Exit (GEOJSON) (2024) Dataset. data.gov.sg. Retrieved December 10, 2024 from https://data.gov.sg/datasets/d_b39d3a0871985372d7e1637193335da5/view

Index

∗ **datasets**
stations, 8

calculate_travel_time, 2

ext_from_terra, 5
ext_matrix, 5
ext_vect_to_matrix, 6

get_friction_surface, 7
get_friction_surface(), *3*

malariaAtlas::getRaster(), *7*, *8*

stations, 8