Package ‘geodist’

October 19, 2022

Title Fast, Dependency-Free Geodesic Distance Calculations

Version 0.0.8

Description Dependency-free, ultra fast calculation of geodesic
distances. Includes the reference nanometre-accuracy geodesic
distances of Karney (2013) <doi:10.1007/s00190-012-0578-z>, as used by
the 'sf' package, as well as Haversine and Vincenty distances. Default
distance measure is the "Mapbox cheap ruler" which is generally more
accurate than Haversine or Vincenty for distances out to a few hundred
kilometres, and is considerably faster. The main function accepts one
or two inputs in almost any generic rectangular form, and returns
either matrices of pairwise distances, or vectors of sequential
distances.

License MIT + file LICENSE

URL https://github.com/hypertidy/geodist

BugReports https://github.com/hypertidy/geodist/issues

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.2.1

Config/testthat/edition 3

Config/testthat/parallel true

Author Mark Padgham [aut, cre],
Michael D. Sumner [aut],
Charles F.F Karney [cph] (Original author of included code for geodesic
distances)

Maintainer Mark Padgham <mark.padgham@email.com>

Repository CRAN

Date/Publication 2022-10-19 11:15:07 UTC
Description

Convert one or two rectangular objects containing lon-lat coordinates into vector or matrix of geodesic distances in metres.

Usage

```r
geodist(
  x,
  y,
  paired = FALSE,
  sequential = FALSE,
  pad = FALSE,
  measure = "cheap",
  quiet = FALSE
)
```

Arguments

- **x** Rectangular object (matrix, `data.frame`, `tibble`, whatever) containing longitude and latitude coordinates.
- **y** Optional second object which, if passed, results in distances calculated between each object in `x` and each in `y`.
- **paired** If TRUE, calculate paired distances between each entry in `x` and `y`, returning a single vector.
- **sequential** If TRUE, calculate (vector of) distances sequentially along `x` (when no `y` is passed), otherwise calculate matrix of pairwise distances between all points.
- **pad** If `sequential = TRUE` values are padded with initial `NA` to return `n` values for input with `n` rows, otherwise return `n - 1` values.
- **measure** One of "haversine", "vincenty", "geodesic", or "cheap" specifying desired method of geodesic distance calculation; see Notes.
- **quiet** If FALSE, check whether max of calculated distances is greater than accuracy threshold and warn.
**Value**

If only `x` passed and `sequential = FALSE`, a square symmetric matrix containing distances between all items in `x`; If only `x` passed and `sequential = TRUE`, a vector of sequential distances between rows of `x`; otherwise if `y` is passed, a matrix of `nrow(x)` rows and `nrow(y)` columns. All return values are distances in metres.

**Note**

`measure = "cheap"` denotes the mapbox cheap ruler [https://github.com/mapbox/cheap-ruler-cpp](https://github.com/mapbox/cheap-ruler-cpp); `measure = "geodesic"` denotes the very accurate geodesic methods given in Karney (2013) "Algorithms for geodesics" J Geod 87:43-55, and as provided by the codesf::st_dist() function.

**Examples**

```r
n <- 50
# Default "cheap" distance measure is only accurate for short distances:
x <- cbind (runif (n, -0.1, 0.1), runif (n, -0.1, 0.1))
y <- cbind (runif (2 * n, -0.1, 0.1), runif (2 * n, -0.1, 0.1))
colnames (x) <- colnames (y) <- c ("x", "y")
d0 <- geodist (x) # A 50-by-50 matrix
d1 <- geodist (x, y) # A 50-by-100 matrix
d2 <- geodist (x, sequential = TRUE) # Vector of length 49
d3 <- geodist (x, sequential = TRUE, pad = TRUE) # Vector of length 50
d0_2 <- geodist (x, measure = "geodesic") # nanometre-accurate version of d0

# Input data can also be 'data.frame' objects:
xy <- data.frame (x = runif (n, -0.1, 0.1), y = runif (n, -0.1, 0.1))
d <- geodist (xy)
```

---

**Description**

Benchmark errors for different geodist measures

**Usage**

```r
geodist_benchmark(lat = 0, d = 1, n = 100L)
```

**Arguments**

- `lat` : Central latitude where errors should be measured
- `d` : Distance in metres over which errors should be measured
- `n` : Number of random values used to generate estimates
Value

A `data.frame` with three columns respectively comparing the accuracy of the [Haversine, Vincenty, cheap] metrics against geodesic measures in both absolute and relative terms (as two rows of the table).

Examples

```r
geodist_benchmark (0.0, 1.0, 100L)
```

Description

An alternative interface to the main `geodist` function that directly accepts inputs as individual vectors of coordinates, rather than the matrix or `data.frame` inputs of the main function. This interface is provided for cases where computational efficiency is important, and will generally provide faster results than the main function.

Usage

```r
geodist_vec(  
x1,  
y1,  
x2,  
y2,  
paired = FALSE,  
sequential = FALSE,  
pad = FALSE,  
measure = "cheap",  
quiet = FALSE  
)
```

Arguments

- **x1**: Numeric vector of longitude coordinates
- **y1**: Numeric vector of latitude coordinates
- **x2**: Optional second numeric vector of longitude coordinates
- **y2**: Optional second numeric vector of latitude coordinates
- **paired**: If TRUE, calculate paired distances between each entry in `(x1, y1)` and `(x2, y2)`, returning a single vector.
- **sequential**: If TRUE, calculate (vector of) distances sequentially along `(x1, y1)` (when no `(x2, y2)` are passed), otherwise calculate matrix of pairwise distances between all points.
georange

pad
If sequential = TRUE values are padded with initial NA to return n values for inputs of length n, otherwise return n - 1 values.

measure
One of "haversine", "vincenty", "geodesic", or "cheap" specifying desired method of geodesic distance calculation; see Notes.

quiet
If FALSE, check whether max of calculated distances is greater than accuracy threshold and warn.

Value
If only (x1, y1) are passed and sequential = FALSE, a square symmetric matrix containing distances between all items in (x1, y1); If only (x1, y1) are passed and sequential = TRUE, a vector of sequential distances between matching elements of (x1, y1); otherwise if (x2, y2) are passed, a matrix of length(x1) == length(y1) rows and length(x2) == length(y2) columns.

Note
measure = "cheap" denotes the mapbox cheap ruler https://github.com/mapbox/cheap-ruler-cpp; measure = "geodesic" denotes the very accurate geodesic methods given in Karney (2013) "Algorithms for geodesics" J Geod 87:43-55, and as provided by the codesf::st_dist() function.

Examples
n <- 50
# Default "cheap" distance measure is only accurate for short distances:
x1 <- -1 + 2 * runif (n, -0.1, 0.1)
y1 <- -1 + 2 * runif (n, -0.1, 0.1)
d0 <- geodist_vec (x1, y1) # A 50-by-50 matrix
d2 <- geodist_vec (x1, y1, sequential = TRUE) # Vector of length 49
d2 <- geodist_vec (x1, y1, sequential = TRUE, pad = TRUE) # length 50
x2 <- -10 + 20 * runif (2 * n, -0.1, 0.1)
y2 <- -10 + 20 * runif (2 * n, -0.1, 0.1)
d1 <- geodist_vec (x1, y1, x2, y2) # A 50-by-100 matrix

Description
Calculate range of distances (min-max) between all points in one or two rectangular objects containing lon-lat coordinates.

Usage
georange(x, y, sequential = FALSE, measure = "cheap")
Arguments

x  Rectangular object (matrix, data.frame, tibble, whatever) containing longitude and latitude coordinates.
y  Optional second object which, if passed, results in distances calculated between each object in x and each in y.
sequential  If TRUE, calculate (vector of) distances sequentially along x (when no y is passed), otherwise calculate matrix of pairwise distances between all points.
measure  One of "haversine", "vincenty", "geodesic", or "cheap" specifying desired method of geodesic distance calculation; see Notes.

Value

A named vector of two numeric values: minimum and maximum, giving the respective distances in metres.

Note

measure = "cheap" denotes the mapbox cheap ruler https://github.com/mapbox/cheap-ruler-cpp; measure = "geodesic" denotes the very accurate geodesic methods given in Karney (2013) "Algorithms for geodesics" J Geod 87:43-55, and as provided by the codesf::st_dist() function.

Examples

n <- 50
x <- cbind (-10 + 20 * runif (n), -10 + 20 * runif (n))
y <- cbind (-10 + 20 * runif (2 * n), -10 + 20 * runif (2 * n))
colnames (x) <- colnames (y) <- c ("x", "y")
# All of the following returns vector of two values: minimum and maximum:
d0 <- georange (x)
d1 <- georange (x, y)
d2 <- georange (x, sequential = TRUE)
d0_2 <- georange (x, measure = "geodesic")  # nanometre-accurate version of d0
Index

geodist. 2, 4
geodist_benchmark. 3
geodist_vec. 4
georange. 5