Package ‘geodist’

April 17, 2019

Title Fast, Dependency-Free Geodesic Distance Calculations
Version 0.0.3
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
Suggests knitr, testthat
NeedsCompilation yes
VignetteBuilder knitr
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
URL https://github.com/hypertidy/geodist
BugReports https://github.com/hypertidy/geodist/issues
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 2019-04-17 12:30:03 UTC
**Description**

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

**Usage**

```r
ggeodist(x, y, paired = FALSE, sequential = FALSE, pad = 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`.
- `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.

**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.

**Note**

- `measure` = "cheap" denotes the mapbox cheap ruler https://github.com/mapbox/cheap-ruler-cpp;
- `measure` = "geodetic" denotes the very accurate geodetic methods given in Kearney (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")
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

description

Benchmark errors for different geodist measures

Usage

geodist_benchmark(lat = 0, d = 1, n = 100)

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

Examples

geodist_benchmark (0, 1, 100)

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

**Examples**

```r
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
geodist-package (geodist), 2
geodist_benchmark, 3
georange, 3