Package: spreadrate (via r-universe)

October 5, 2024

Type Package

Title Estimate the spread-rate of epidemics

Version 0.1.13

Date 2019-11-13

Description Estimate the local velocity of propagation of an epidemic event, given dates and locations of observed cases. The method estimates the surface of first date of invasion by interpolation of the earliest observations in a neighbourhood and derives the local spread rate as the inverse slope of the surface. Quantify the estimation uncertainty by a Monte Carlo approach. Visualise and simulate the spatio-temporal progress of epidemics.

Depends R (>= 3.2), sf, raster

License GPL-3 | file LICENSE

Language en-GB

Encoding UTF-8

LazyData true

URL https://umr-astre.pages.mia.inra.fr/spreadrate

BugReports https://forgemia.inra.fr/umr-astre/spreadrate/issues

Imports fields, furrr, lwgeom, sp

Suggests INLA, testthat (>= 2.1.0), ggplot2

Additional_repositories https://inla.r-inla-download.org/R/testing

RoxygenNote 6.1.1

Repository https://cirad-astre.r-universe.dev

RemoteUrl https://forgemia.inra.fr/umr-astre/spreadrate

RemoteRef HEAD

RemoteSha 0fa071c495b49f43ae391485e41fc9cd624c6ff4

Contents

`dist2arc` *Approximate angle for a distance*

Description

Compute an approximate angle that correspond to a given distance in both longitude and latitude.

Usage

```
dist2arc(x, dm)
```

Arguments

`x`	A `sfc_POINT` object in longlat coordinates, indicating a position in the globe.
`dm`	A target distance in m.

Details

The distance corresponding to an angle in longitude and latitude depends on the location in the earth and is typically different for both directions.

This function computes the angle that yields approximately the target distance when added to the longitude or the latitude by minimising the squared errors.

The average error is returned as an attribute.

Value

A numeric value of angle in decimal degrees, such that a displacement in longitude and latitude by that magnitude corresponds to approximately the given distance.

Examples

```
require(sf)
## What angle yields 1 km at the equator? (and with which precision?)
x0 <- st_sfc(st_point(c(0, 0)), crs = 4326)
dist2arc(x0, 1e3)

## Note that this is independent of the longitude
x1 <- st_sfc(st_point(c(90, 0)), crs = 4326)
dist2arc(x1, 1e3)

## But is strongly dependent of the latitude
x2 <- st_sfc(st_point(c(0, 45)), crs = 4326)
x3 <- st_sfc(st_point(c(0, 75)), crs = 4326)
dist2arc(x2, 1e3)
dist2arc(x3, 1e3)
```

---

estimation_mask *Raster termplate for estimation*

Description

Given a set of observations, define a estimation template.

Usage

```
estimation_mask(x, buffer_size, res = buffer_size/8)
```

Arguments

x	An object of class sfg, sfc or sf, including particularly objects of class sr_obs. Geographical objects that determine the estimation region.
buffer_size	Numeric. Expansion of the extent around the convex-hull of x. In the units of the data (e.g. m, or arc-degrees).
res	Numeric vector of length 1 or 2. Resolution of the raster template. Size(s) of the pixel, in metres.

Examples

```
## Some random points in the unit square
pp <- st_multipoint(matrix(runif(30), ncol = 2))
(em <- estimation_mask(pp, buffer_size = .3, res = .1))
plot(em)
points(st_coordinates(pp))
```

`filter_earliest_neigh` *Earliest observations within a neighbourhood*

Description

Select a subset of the observation points at least `tol` m far apart as representative of the emphneighbourhood. Assign the earliest times observed in the neighbourhood to them.

Usage

```
filter_earliest_neigh(x)
```

Arguments

x	sr_obs object.

Details

Note that the neighbouring distance must be expressed in m, even when the coordinates are geographical.

This function reproduces the operation for all the Monte Carlo replicates if any.

For the original dataset, the neighbouring tolerance parameter is the mean value of the interval specified in its uq.

Value

Another sr_obs object with a subset of the points and the earliest times observed in the neighbourhood of each.

Examples

```
d <- data.frame(lon = runif(30), lat = runif(30), date = 1:30)

## Use between 10 and 20 % of data diameter as neighbouring
## tolerance. The main result will use exactly 15% while the
## \code{mc} replicates will use random Gaussian values from
## that interval (at 99.9%)
sruq <- sr_uq(10, 0, 1, neigh_tol = c(-10, -20))
sro <- sr_obs(d, "date", uq = sruq)
srf <- filter_earliest_neigh(sro)
```

fit_surface *Fit a Thin-plate splines model to a set of observations*

Description

Fit a Thin-plate splines model to a set of observations

Usage

```
fit_surface(x)
```

Arguments

x	A sr_obs object defined with sr_obs.

Value

An object of class Krig and Tps which includes the fitted values and the model residuals. See Tps. It also allows interpolation using interpolate.

Examples

```
d <- data.frame(lon = runif(30), lat = runif(30))
d <- transform(d, date = round(10 * sqrt(lon**2 + lat**2)))
sro <- sr_obs(d, "date")
fm <- fit_surface(sro)
summary(fm)
plot(fm)    # Some model diagnostics plots
fields::surface(fm)  # Quick image/contour plot of predicted surface.
```

get_lonlatvars *Identify variables with geographical coordinates*

Description

Given a data.frame with two variables with longitudes and latitudes, return the variable names.

Usage

```
get_lonlatvars(x)
```

Arguments

x	data.frame

Details

This function explores the names of the variables for the patterns 'lon' or 'long' and 'lat' ignoring case. If exactly one of each patterns is detected, it verifies that their magnitudes are within -365 and 365 for the longitude and within -90.1 and 90.1 for the latitude.

Value

A character vector with the variable names for longitude and latitude, or an error.

Examples

```
d <- data.frame(lon = 1:10, lat = 1:10)
get_lonlatvars(d)
```

---

invslope *Compute the inverse slope of the input surface*

Description

This function is used to derive local spread rate from a first-date-of-invasion surface

Usage

```
invslope(x, rmtop, bnd)
```

Arguments

x	Raster* with estimated first-arrival date. Needs to be projected.
rmtop	Numeric, between 0 and 100. Discard the rmtop top percent of values.
bnd	Numeric vector of length 2. Expected range of values. Everything beyond this range is filtered.

Details

To control for spurious numerical divergences, the user can remove the largest rmtop values beyond a reasonable interval bnd.

Value

An object of the same class Raster* as the input.

Examples

```
## A flat surface with constant slope of 1
r_values <- replicate(100, seq(0.01, 1, length = 100))
r <- raster(r_values, crs = sp::CRS("+init=epsg:3857"))
plot(r)
invslope(r)
```

mc_sample *Monte Carlo samples from a dataset*

Description

Shuffle observations in space and time.

Usage

```
mc_sample(x)
```

Arguments

x	An object of class sr_obs from function sr_obs

Details

Add an attribute mc with a list of datasets with some noise added to the original coordinates and dates. The number of Monte Carlo samples and the variablity of the noise is given by the sr_uq attribute in the object.

Examples

```
d <- data.frame(lon = 1, lat = 1, date = 1)
sr_obs(d, "date", uq = sr_uq(nsim = 3, space = 1, time = 1))
```

neigh_tol *Neighbourhood Tolerance Parameter*

Description

Get the value or interval of the Neighbourhood Tolerance Parameter of a sr_obs object.

Usage

```
neigh_tol(x)
```

Arguments

x	A sr_obs object.

Details

This function retrieves the values set during creation with sr_obs in absolute terms, performing the convertion from values relative to the diameter of the dataset if necessary. See sr_uq.

Value

A numeric vector of size 1 or 2. Either a constant value for this parameter or a interval with distance values in m.

Examples

```
d <- data.frame(lon = 1:3, lat = 1:3, date = 1:3)

## Here, the neighbourhood tolerance parameter is set between
## 2.5 % of the dataset diameter and 1 km.
## Note that the resulting distances are in m, even if the
## original coordinates are geographic.
obs <- sr_obs(d, "date", uq = sr_uq(neigh_tol = c(-2.5, 1e4)))
neigh_tol(obs)
```

---

representative_points *Choose representative points*

Description

Select a subset of points at a minimum tolerance distance from each other.

Usage

```
representative_points(x, dTolerance)
```

Arguments

x	Object of class sf
dTolerance	Numeric. Distance tolerance in m.

Value

Object of class sf with a subset of the original locations in x.

Examples

```
## 50 points in the unit square
require(sf)
x <- st_sfc(st_multipoint(matrix(runif(100), ncol = 2)), crs = 3857)
representative_points(x, .5)
```

sr *Spread rate*

Description

Estimate the local spread rate of an epidemiological invasion.

Usage

```
sr(x, r = estimation_mask(x, buffer_size = max(st_distance(x))/10))
```

Arguments

x	A sr_obs object defined with sr_obs.
r	A raster template for results.

Details

This function will compute spread-rate estimates for all of the Monte Carlo samples in the dataset. It uses internally the function future_map which will take advantage of multiple processors or cluster access if you set a proper plan beforehand. See examples.

Examples

```
require(sf)
## Randomly sample 100 point in the unit square
x_mat <- matrix(runif(100 * 2), ncol = 2)

## Make it spatial point in a projected CRS
x_sfc <- st_cast(st_sfc(st_multipoint(x_mat), crs = 3857), "POINT")

## Assign dates proportional to the squared distance to the origin
## So that spread-rate decreases linearly
obs_dates <- sqrt(rowSums(x_mat**2))
sro <- sr_obs(st_sf(x_sfc, date = obs_dates), "date")

## Alternative setting with uncertainty quantification and an
## absolute specification of the neighbouring tolerance parameter.
# sro2 <- sr_obs(st_sf(x_sfc, date = obs_dates), "date",
# uq = sr_uq(6, .2, .2, .3))

## Estimate local spread-rate
## in the units of the coordinates divided by the units of time
sre <- sr(sro)
plot(sre, col = hcl.colors(12))
points(st_coordinates(sro), pch = 19)
```

sr_fdoi *First-date of invasion*

Description

Estimate a surface of first-date of invasion, given a sparse set of observation locations and times.

Usage

```
sr_fdoi(x, estimation_mask)
```

Arguments

x	A sr_obs object defined with sr_obs.
estimation_mask	A raster template for results.

Details

This function first takes the earliest observed case in a neighbourhood, as defined by the parameter neigh_tol in the definition of x. Next, it interpolates the earliest observations with a smooth surface, using a Thin-Plate splines model.

Value

A RasterLayer with estimated dates of invasion at each pixel.

Examples

```
d <- data.frame(lon = runif(30), lat = runif(30))
d <- transform(d, date = round(10 * sqrt(lon**2 + lat**2)))
sro <- sr_obs(d, "date")
r <- raster(st_sf(sro), vals = 1)
fdoi <- sr_fdoi(sro, r)
plot(fdoi, col = hcl.colors(12))
points(d)
```

sr_obs *Define a observations dataset*

Description

Provide a dataset with the necessary meta-data for spread-rate estimation.

Usage

```
sr_obs(x, timevar, uq)
```

Arguments

x	data frame with geographical coordinates or sf object of type POINT, or another sr_obs object.
timevar	Character. Variable name with observation times or dates.
uq	Object created with sr_uq() with parameters of Uncertainty Quantification. See Details for default behaviour.

Details

If x is a data.frame with geographical coordinates (longitude and latitude) these are automatically recognised and validated. See get_lonlatvars,

If your dataset is projected (no lon/lat variables) then first build the sf object yourself using the corresponding Coordinate Reference System. See example.

If uq is missing and x is a data.frame or sf, a default value is generated with sr_uq() which does not perform Uncertainty Quantification. See sr_uq. If uq is missing and x is a sr_obs object,

This function will produce the Monte Carlo samples from the dataset and the uq object, if necessary. It uses internally the function future_map which will take advantage of multiple processors or cluster access if you set a proper plan beforehand. See examples.

Value

Object of class sr_obs, which is a sf of type POINT with complementary meta-data.

Examples

```
  d_geo <- data.frame(lon = 1, lat = 1, date = 1)
  sr_obs(d_geo, "date")

  ## Projected coordinates
  d_prj <- data.frame(x = 1, y = 1, date = 1)
  d.sf <- st_as_sf(d_prj, coords = c("x", "y"), crs = 27561)
  sr_obs(d.sf, "date")

## Not run:
  ## Perform Monte Carlo samples in parallel
  library(furrr)
  plan(multiprocess)
  uq <- sr_uq(nsim = 3, space = 1, time = 1)
  sr_obs(d_geo, "date", uq = uq)

## End(Not run)
```

sr_uq *Parameters for Uncertainty Quantification*

Description

Define the parameters for quantifying the uncertainty in the estimation of the spread-rate.

Usage

```
sr_uq(nsim = 1L, space = 0, time = 0, neigh_tol = -4.5)
```

Arguments

nsim	Integer > 0. Number of Monte Carlo replicates.
space	Numeric, non-negative. Uncertainty of spatial coordinates.
time	Numeric, non-negative. Uncertainty of temporal values.
neigh_tol	Number or interval (numeric vector of length 2). Neighouring-tolerance parameter in the units of the dataset coordinates if positive, or relative to the dataset diameter if negative.

Details

The actual observations will be The number of Monte Carlo replicates The space and time arguments are expressed in the data spatial and temporal units.

For each one of the nsim replicates, the observations will be randomly shifted in space within a circle of radious space, while the dates will be shifted within the interval +- time with a discrete approximation to a Gaussian curve.

The neighbouring-tolerance parameter neigh_tol is used to filter the earliest observed cases in a *neighbourhood*. If two points are closer than this value, they are considered roughly in the same neighbourhood. The date of invasion of a location is the earliest observed case in the neighbourhood. It can be defined as a numeric value in the units of the coordinates, or as a range (i.e. a numeric vector of length 2). Negative values will be interpreted as a percentage of the diameter of the dataset.

Value

An object of class sr_uq which is a list with the given or default values, after some sanity checks.

Examples

```
## Use the default values: work with current observations
## without quantifying uncertainty
uq_def <- sr_uq()

## Set the neighbouring-tolerance parameter only
uq1 <- sr_uq(neigh_tol = 800)
```

```
## 10 MC replicates, shift locations within a circle of radious
## 1 km, shift dates by zero/one day up or down, consider points
## closer than 5% of the diameter of the dataset as in the same
## neighbourhood
uq <- sr_uq(nsim = 10, space = 1e3, time = 1, neigh_tol = -5)
```

`st_transform.sr_obs` *Transform coordinates of a sr_obs object*

Description

Transform the underlying `sf` object and all the MC replicas, if any.

Usage

```
## S3 method for class 'sr_obs'
st_transform(x, crs, ...)
```

Arguments

`x`	A `sr_obs` object created with `sr_obs`.
`crs`	Coordinate reference system: inteer with the EPSG code or character with proj4string
`...`	Ignored

Value

A `sr_obs` object in the target projection.

Examples

```
## Some observed data and MC replicates in longitude/latitude
x <- sr_obs(data.frame(lat = 1, lon = 1, day = 1), "day", uq = sr_uq(2, 1, 1))
st_crs(x)
lapply(attr(x, "mc"), st_crs)

xt <- st_transform(x, 3857)
st_crs(xt)
lapply(attr(xt, "mc"), st_crs)
```

`time_jitter` *Jitter a vector of 1D values*

Description

Modify a vector of dates by a random quantity in [-e, e]. The distribution decreases exponentially, with rate 1/2

Usage

```
time_jitter(x, e)
```

Arguments

`x`	Numeric or Date vector.
`e`	Numeric. Semi-width of the jittering interval.

Value

Vector of the same size and type as `x` with jittered values.

Examples

```
spreadrate:::time_jitter(1:10, 1)  # Numeric input
spreadrate:::time_jitter(Sys.Date() + 1:10, 1)  # Date input

## Distribution of jittering
x <- seq.int(1e3)
xj <- spreadrate:::time_jitter(x, 5)
barplot(table(x - xj))
```

Index