Package: geosample (via r-universe)

November 5, 2026

Type Package

Title Construction of Geostatistical Sampling Designs

Version 0.2.1

Author Michael G Chipeta, Peter J Diggle

Maintainer Michael G Chipeta <mgchipeta@mlw.mw>

Imports splancs, pdist, graphics, stats

Depends R (>= 3.0.0), sf, sp

Description Functions for constructing sampling designs, including spatially random, inhibitory (simple or with close pairs), both discrete and continuous, and adaptive designs.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests geoR, dplyr, PrevMap, rmarkdown, testthat, viridisLite, raster, knitr

VignetteBuilder knitr

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2019-01-17 18:34:51 UTC

RemoteUrl https://gitlab.com/b-rowlingson/geosample

RemoteRef HEAD

RemoteSha 19728b9051c61d3b024bbbd3db5a8b0f7e729c7b

Contents

adaptive.sample 2
border 6

contin.inhibit.sample 7
discrete.inhibit.sample 9
majete 13
random.sample 14
sim.data 15

Index **17**

`adaptive.sample` *Spatially adaptive sampling*

Description

Draw an additional sample from a set of available locations in a defined geographical region, imposing a minimum distance between any two sampled units and taking into account existing data from previously sampled locations. The algorithm allows the user to specify either a *prediction variance (PV)* criterion or an *exceedance probability (EP)* criterion to choose new sampling locations. The function accepts either `sf` or `sp` objects.

Usage

```
adaptive.sample(obj1, obj2, pred.var.col = NULL, excd.prob.col = NULL,
  batch.size = 1, delta, criterion, poly = NULL, plotit = TRUE)
```

Arguments

`obj1`	a `sf` or `sp` object of **locations available for sampling**, where each line contains the coordinates of a spatial location, a **prediction variance** or an **exceedance probability** at that location and, optionally, values of one or more covariates. NOTE that only one of the two quantities (i.e. PV or EP) is required to add samples adaptively. Locations that meet the specified selection criterion are equally likely to be sampled subject to spatial contraints. See `criterion` and **Details** for more information.
`obj2`	a `sf` or `sp` object of **locations previously sampled**. Each line corresponds to one spatial location. It must contain values of 2D coordinates and may also contain the values of one or more covariates. The initial sample locations design can be generated from `random.sample`, `discrete.inhibit.sample`, `contin.inhibit.sample` or some other design.
`pred.var.col`	a scalar of length one indicating the column number corresponding to prediction variance at each spatial location in `obj1`. This is required if `criterion = "predvar"`. See `'criterion'` and **Details** for information.
`excd.prob.col`	a scalar of length one indicating the column number corresponding to exceedance probabilities at each spatial location in `obj1`. This is required if `criterion = "exceedprob"`. See `'criterion'` and **Details** for information.
`batch.size`	a non-negative integer giving the number of adaptively chosen locations to be added to the existing sample (design).
`delta`	minimum permissible distance between any two locations in the sample.

`criterion`	criterion used for choosing new locations x^*. Use `"predvar"` for **prediction variance** or `"exceedprob"` for **exceedance probablity**. See the **Details** section for more information.
`poly`	'optional', a `sf` or `sp` polygon object in which the design sits. The default is the bounding box of points given by `obj1`.
`plotit`	'logical' specifying if graphical output is required. Default is `plotit = TRUE`.

Details

For the predictive target $T = S(x)$ at a particular location `x`, given an initial set of sampling locations $X_0 = (x_1, \ldots, x_{n0})$ the available set of additional sampling locations is $A_0 = X * \backslash X_0$. To mimic spatially continuous sampling, the initial set should be a fine grid to cover the region of interest

Define the following notation:

- $\mathcal{X}^*$ is the set of all potential sampling locations, with number of elements n^*.
- X_0 is the initial sample, with number of elements n_0.
- b is the batch size.
- $n = n_0 + kb$ is the total sample size.
- $\mathcal{X}_j, j \geq 1$ is the set of locations added in the j^{th} batch, with number of elements b.
- $A_j = \mathcal{X}^* \setminus \mathcal{X}_0 \cup \ldots \cup X_j$ is the set of available locations after addition of the j^{th} batch.

1. Prediction variance criterion.

For each $x \in A_0$, denote by *PV(x)* the prediction variance, $\mathtt{Var}(T|Y_0)$. The algorithm then proceeds as follows.

- Step 1. Use a non-adaptive design to determine $\mathcal{X}_0$.
- Step 2. Set $j = 0$.
- Step 3. For each $x \in A_j$, calculate $PV(x)$.
  - Step 3.(i) choose $x^* = \mathtt{arg\,max}_{A_j} PV(x)$,
  - Step 3.(ii) if $||x^* - x_i|| > \delta, \forall i = 1, \ldots, n_0 + jb$, add x^* to the design,
- Step 4. Repeat step 3 until b locations have been added to form the set X_{j+1}.
- Step 5. Set $A_j = A_{j=1} \setminus \mathcal{X}_j$ and we update j to $j + 1$.
- Step 6. Repeat steps 3 to 5 until the total number of sampled locations is n or $A_j = \emptyset$.

2. Exceedance probability criterion.

For each $x \in A_0$, denote by *EP(x)* the exceedance probability, $P[\{T(x) > t|y_0\} - 0.5]$ for a specified threshold *t*. The algorithm proceeds as above, with changes only in step 3, as follows.

- Step 3. For each $x \in A_j$, calculate $EP(x)$.
  - Step 3.(i) choose $x^* = \mathtt{arg\,min}_{A_j} EP(x)$.

Value

A list with the following four components:

`total.size`: the total number of locations, n, sampled.

`delta`: the value of δ.

`criterion`: the sample selection criterion used for adaptive sampling.

`sample.locs`: a list of objects for sample locations. It has the following components.

`curr.sample`: a `sf` or `sp` object of dimension n by 2 containing all sampled locations, where n is the total sample size (initial plus newly added sample locations).

`prev.sample`: a `sf` or `sp` object of dimension n_i by 2 containing `initial sample` locations, where $n_i < n$.

`added.sample`: a `sf` or `sp` object of dimension n_a by 2 containing `additional sample` locations, i.e. adaptively sampled locations, where $n_a = b$, the batch size.

Note

The function can only add a single batch at a time.

Author(s)

Michael G. Chipeta <mchipeta@mlw.mw>

Peter J. Diggle <p.diggle@lancaster.ac.uk>

References

Chipeta M G, Terlouw D J, Phiri K S and Diggle P J. (2016a). Adaptive geostatistical design and analysis for prevalence surveys, *Spatial Statistics* **15**, pp. 70-84.

Giorgi E and Diggle P J. (2017). PrevMap: an R package for prevalence mapping. *Journal of Statistical Software*. **78**:1-29, doi: 10.18637/jss.v078.i08

Kabaghe A N, Chipeta M G, McCann R S, Phiri K S, Van Vugt M, Takken W, Diggle P J, and Terlouw D J. (2017). Adaptive geostatistical sampling enables efficient identification of malaria hotspots in repeated cross-sectional surveys in rural Malawi, *PLoS One* **12**(2) pp. e0172266

See Also

`discrete.inhibit.sample` and `contin.inhibit.sample`

Examples

```
## Not run:
data("sim.data")
library("PrevMap")
library("sf")

#1. Generate inhibitory design without close pairs using discrete.inhibit.sample().
set.seed(1234)
xy.sample <- discrete.inhibit.sample(obj = sim.data, size = 100, delta = 0.075,
                                     k = 0, plotit = TRUE)
```

```
names(xy.sample)
init.design <- xy.sample$sample.locs

#2. Data analysis
knots <- as.matrix(expand.grid(seq(-0.2, 1.2, length = 15),
                               seq(-0.2, 1.2, length = 15)))
lr.mcmc <- control.mcmc.MCML(n.sim = 10000, burnin = 1000, thin = 6)

par0.lr <- c(0.001, 1, 0.4)
fit.MCML.lr <- binomial.logistic.MCML(y ~ 1,
                              units.m = ~units.m, coords = ~st_coordinates(init.design),
                                     data = init.design, par0 = par0.lr,
                                     fixed.rel.nugget = 0, start.cov.pars = par0.lr[3],
                                control.mcmc = lr.mcmc, low.rank = TRUE, knots = knots,
                                     kappa = 1.5, method = "nlminb", messages = TRUE)

summary(fit.MCML.lr, log.cov.pars = FALSE)

# Note: parameter estimation above can and should be repeated several times with updated starting
# values for the covariance function.

#3. Plug-in prediction using estimated parameters
pred.MCML.lr <- spatial.pred.binomial.MCML(object = fit.MCML.lr,
                                           control.mcmc = lr.mcmc,
                                           grid.pred = st_coordinates(sim.data),
                                           type = "joint", messages = TRUE,
                                           scale.predictions = "prevalence",
                                           standard.errors = TRUE,  thresholds = 0.45,
                                           scale.thresholds = "prevalence")


#4. Visualisation of analysis from initial sample
plot(pred.MCML.lr, type = "prevalence", summary = "predictions",
     zlim = c(0, 1), main = "Prevalence - predictions")
contour(pred.MCML.lr, "prevalence", "predictions",
        zlim = c(0, 1), levels = seq(0.1,0.9, 0.1), add = TRUE)

plot(pred.MCML.lr,  summary = "exceedance.prob",
     zlim = c(0, 1), main = "Prevalence - exceedance probability")
contour(pred.MCML.lr, summary = "exceedance.prob",
        zlim = c(0, 1), levels = seq(0.1,0.3, 0.1), add = TRUE)

plot(pred.MCML.lr, type = "prevalence",  summary = "standard.errors",
     main = "Prevalence - standard errors")

#5. Adaptive sampling
#create data frame of ingredients to adaptive sampling from spatial predictions above
obj1 <- as.data.frame(cbind(pred.MCML.lr$grid,
                            c(pred.MCML.lr$prevalence$standard.errors)^2,
                            pred.MCML.lr$exceedance.prob))
colnames(obj1) <- c("x", "y", "pred.var", "exceed.prob")
obj1 <- sf::st_as_sf(obj1, coords = c('x', 'y'))
```

```
#adaptive sampling using exceedance probability criterion.
adapt.design.ep <- adaptive.sample(obj1 = obj1, obj2 = init.design,
                                   pred.var.col = 1, excd.prob.col = 2,
                                   criterion = "exceedprob", delta = 0.08,
                                   batch.size = 10, poly = NULL, plotit = TRUE)

#adaptive sampling using exceedance probability criterion.
adapt.design.pv <- adaptive.sample(obj1 = obj1, obj2 = init.design,
                                   pred.var.col = 1, excd.prob.col = 2,
                                   criterion = "predvar", delta = 0.08,
                                   batch.size = 10, poly = NULL, plotit = TRUE)


## End(Not run)
```

`border` *Majete study area borders*

Description

This data-set contains the borders for Majete ***focal area A***, relating to the study of the prevalence of malaria in Chikhwawa district, southern Malawi. The data-set contains Geometry set for 1 feature.

- `Geometry type:` Polygon.
- `dimension:` XY.
- `bbox:` xmin: 654.6224 ymin: 8243.117 xmax: 664.3984 ymax: 8253.008
- `epsg (SRID):` 32736
- `proj4string:` +proj=utm +zone=36 +south +datum=WGS84 +units=m +no_defs

Usage

```
data("border")
```

Format

Simple feature polygon

`contin.inhibit.sample` *Spatially continuous sampling*

Description

Draws a spatially continous sample of locations within a polygonal sampling region according to an **"inhibitory plus close pairs"** specification.

Usage

```
contin.inhibit.sample(poly, size, delta, delta.fix = FALSE, k = 0,
  rho = NULL, ntries = 10000, plotit = TRUE)
```

Arguments

`poly`	a `sf` or `sp` polygon in which to generate the design.
`size`	a non-negative integer giving the total number of locations to be sampled.
`delta`	minimum permissible distance between any two locations in preliminary sample. This can be allowed to vary with the number of `'close pairs'` if a **simple inhibitory** design is compared to one of the **inhibitory plus close pairs** design.
`delta.fix`	'logical' specifies whether `delta` is fixed or allowed to vary with number of close pairs k. Default is `delta.fix = FALSE`.
`k`	number of locations in preliminary sample to be replaced by near neighbours of other preliminary sample locations to form `close pairs` (integer between 0 and `size/2`). A **simple inhibitory** deisgn is generated when $k = 0$.
`rho`	maximum distance between the two locations in a `'close-pair'`.
`ntries`	number of rejected proposals after which the algorithm will terminate.
`plotit`	'logical' specifying if graphical output is required. Default is `plotit = TRUE`.

Details

To draw a simple inhibitory (**SI**) sample of size `n` from a spatially continuous region A, with the property that the distance between any two sampled locations is at least `delta`, the following algorithm is used.

- Step 1. Set $i = 1$ and generate a point x_1 uniformly distributed on $\mathcal{D}$.
- Step 2. Generate a point x uniformly distributed on $\mathcal{D}$ and calculate the minimum, $d_{\min}$, of the distances from x_i to all $x_j : j \leq i$.
- Step 3. If $d_{\min} \geq \delta$, increase i by 1, set $x_i = x$ and return to step 2 if $i \leq n$, otherwise stop;
- Step 4. If $d_{\min} < \delta$, return to step 2 without increasing i.

Sampling close pairs of points.

For some purposes, it is desirable that a spatial sampling scheme include pairs of closely spaced points, resulting in an inhibitory plus close pairs (**ICP**) design. In this case, the above algorithm

requires the following additional steps to be taken. Let k be the required number of close pairs. Choose a value `rho` such that a close pair of points will be a pair of points separated by a distance of at most `rho`.

- Step 5. Set $j = 1$ and draw a random sample of size 2 from integers $1, 2, \ldots, n$, say (i_1, i_2);
- Step 6. Replace x_{i_1} by $x_{i_2} + u$, where u is uniformly distributed on the disc with centre x_{i_2} and radius `rho`, increase i by 1 and return to step 5 if $i \leq k$, otherwise stop.

When comparing a **SI** design to one of the **ICP** designs, the inhibitory components should have the same degree of spatial regularity. This requires δ to become a function of k namely

$$\delta_k = \delta_0 \sqrt{n/(n-k)}$$

with δ_0 held fixed.

Value

a list with the following four components:

`size`: the total number of sampled locations.

`delta`: the value of δ after taking into account the number of close pairs k. If `delta.fix` = `TRUE`, this will be δ input by the user.

k : the number of close pairs included in the sample (for **inhibitory plus close pairs** design).

`sample.locs`: a `sf` or `sp` object containing coordinates of dimension `n` by 2 containing the sampled locations.

Note

If `'delta'` is set to 0, a completely random sample is generated. In this case, `'close pairs'` are not permitted and `rho` is irrelevant.

Author(s)

Michael G. Chipeta <mchipeta@mlw.mw>

Peter J. Diggle <p.diggle@lancaster.ac.uk>

References

Chipeta M G, Terlouw D J, Phiri K S and Diggle P J. (2016b). Inhibitory geostatistical designs for spatial prediction taking account of uncertain covariance structure, *Enviromentrics*, pp. 1-11.

See Also

`random.sample` and `discrete.inhibit.sample`

Examples

```
library("geoR")
library("sf")
data("parana")
poly <- parana$borders
poly <- matrix(c(poly[,1],poly[,2]),dim(poly)[1],2,byrow=FALSE)
#convert matrix to polygon
poly <- st_sf(st_sfc(st_polygon(list(as.matrix(poly)))))
#poly <- as(poly, "Spatial")
poly

# Generate spatially regular sample
set.seed(5871121)
xy.sample1 <- contin.inhibit.sample(poly=poly,size = 100, delta = 30, plotit = TRUE)


# Generate spatially regular sample with 10 close pairs
set.seed(5871122)
xy.sample2 <- contin.inhibit.sample(poly,size = 100, delta = 30,
                                    k = 5, rho = 15, plotit = TRUE)

# Generate spatially regular sample with 10 close pairs
set.seed(5871123)
xy.sample3 <- contin.inhibit.sample(poly,size = 100, delta = 30, delta.fix = TRUE,
                                    k = 10, rho = 15, plotit = TRUE)
```

`discrete.inhibit.sample`

Spatially discrete sampling

Description

Draw a spatially discrete sample from a specified set of spatial locations within a polygonal sampling region according to an **"inhibitory plus close pairs"** specification.

Usage

```
discrete.inhibit.sample(obj, size, delta, delta.fix = FALSE, k = 0,
  cp.criterion = NULL, zeta, ntries = 10000, poly = NULL,
  plotit = TRUE)
```

Arguments

`obj`	a `sf` or `sp` object where each line corresponds to a spatial location containing values of two-dimensional coordinates and, optionally, the values of one or more associated values, typically an outcome of interest and any associated covariates.
`size`	a non-negative integer giving the total number of locations to be sampled.

`delta`	minimum permissible distance between any two locations in preliminary sample. This can be allowed to vary with number of `'close pairs'` if a **simple inhibitory** design is compared to one of the **inhibitory plus close pairs** design.
`delta.fix`	'logical' specifies whether `'delta'` is fixed or allowed to vary with number of close pairs k. Default is `delta.fix = FALSE`.
`k`	number of close-pair locations in the sample. Must be an integer between 0 and `size`/2.
`cp.criterion`	criterion for choosing close pairs k. The `"cp.zeta"` criterion chooses locations not included in the initial sample, from the uniform distribution of a disk with radius `'zeta'` (NB: `zeta` argument must be provided for this criterion). The `"cp.neighb"` criterion chooses nearest neighbours amongst locations not included in the initial sample (`'zeta'` becomes trivial for `'cp.neighb'` criterion).
`zeta`	maximum permissible distance (radius of a disk with center $x_j^*, j = 1, \ldots, k$) within which a close-pair point is placed. See **Details**.
`ntries`	number of rejected proposals after which the algorithm terminates.
`poly`	'optional', a `sf` or `sp` polygon object in which the design sits. The default is the bounding box of points given by `obj`.
`plotit`	'logical' specifying if graphical output is required. Default is `plotit = TRUE`.

Details

To draw a sample of size n from a population of spatial locations $X_i : i = 1, \ldots, N$, with the property that the distance between any two sampled locations is at least δ, the function implements the following algorithm.

- Step 1. Draw an initial sample of size n completely at random and call this $x_i : i = 1, \ldots, n$.
- Step 2. Set $i = 1$.
- Step 3. Calculate the smallest distance, $d_{\min}$, from x_i to all other x_j in the initial sample.
- Step 4. If $d_{\min} \geq \delta$, increase i by 1 and return to step 2 if $i \leq n$, otherwise stop.
- Step 5. If $d_{\min} < \delta$, draw an integer j at random from $1, 2, \ldots, N$, set $x_i = X_j$ and return to step 3.

Samples generated in this way exhibit more regular spatial arrangements than would random samples of the same size. The degree of regularity achievable will be influenced by the spatial arrangement of the population $X_i : i = 1, \ldots, N$, the specified value of δ and the sample size n. For any given population, if n and/or δ is too large, a sample of the required size with the distance between any two sampled locations at least δ will not be achievable; the algorithm will then find $n_s < n$ points that can be placed for the given parameters.

Sampling close pairs of points.

For some purposes, typically when using the same sample for parameter estimation and spatial prediction, it is desirable that a spatial sampling scheme include pairs of closely spaced points x. The function offers two ways of specifying close pairs, either as the closest available unsampled point to an existing sampled point (`cp.critetrion = cp.neighb`), or as a random choice from amongst all available unsampled points within distance $zeta$ of an existing sampled point (`cp.criterion = cp.zeta`). The algorithm proceeds as follows.

Let k be the required number of close pairs.

- Step 1. Construct a simple inhibitory design **SI**$(n-k, \delta)$.
- Step 2. Sample k from $x_1, \ldots, x_{n-k}$ without replacement and call this set $x_j : j = 1, \ldots, k$.
- Step 3. For each $x_j : j = 1, \ldots, k$, select a close pair x_{n-k+j} according to the specified criterion.

Note: Depending on the spatial configuration of potential sampling locations and, when the selection criterion `cp.criterion = cp.zeta`, the specified value of $zeta$, it is possible that one or more of the selected points x_j in Step 2 will not have an eligible "close pair". In this case, the algorithm will try find an alternative x_j and report a warning if it fails to do so.

Value

a list with the following four components:

`unique.locs:` the number of unique sampled locations.

`delta:` the value of δ after taking into account the number of close pairs k. If `delta.fix = TRUE`, this will be δ input by the user.

k : the number of close pairs included in the sample (for **inhibitory plus close pairs** design).

`sample.locs:` a `sf` or `sp` object containing the final sampled locations and any associated values.

Note

If `'delta'` is set to 0, a completely random sample is generated. In this case, *'close pairs'* are not permitted and `'zeta'` becomes trivial.

Author(s)

Michael G. Chipeta <mchipeta@mlw.mw>

Peter J. Diggle <p.diggle@lancaster.ac.uk>

References

Chipeta M G, Terlouw D J, Phiri K S and Diggle P J. (2016). Inhibitory geostatistical designs for spatial prediction taking account of uncertain covariance structure, *Enviromentrics*, pp. 1-11.

Diggle P J. (2014). *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns.* 3rd ed., Boca Raton: CRC Press

Diggle P J and Lophaven S. (2006). Bayesian geostatistical design, *Scandinavian Journal of Statistics* **33**(1) pp. 53 - 64.

Examples

```
library("sf")
set.seed(1234)
x <- 0.015+0.03*(1:33)
xall <- rep(x,33)
yall <- c(t(matrix(xall,33,33)))
xy <- cbind(xall,yall)+matrix(-0.0075+0.015*runif(33*33*2),33*33,2)
```

```
# Convert to SF object
xy <- xy %>%
  as.data.frame %>%
  sf::st_as_sf(coords = c(1,2))


# Plot the points
plot(st_geometry(xy),pch=19,cex=0.25,xlab="longitude",ylab="latitude",
     cex.lab=1,cex.axis=1,cex.main=1, axes = TRUE)


# Generate spatially random sample
set.seed(15892)
xy.sample1 <- xy[sample(1:dim(xy)[1],50,replace=FALSE),]
plot(xy.sample1, pch = 19, col = 'black', add = TRUE)


set.seed(15892)
xy.sample2 <- discrete.inhibit.sample(obj=xy,size = 100,
                                      delta = 0.08,plotit = TRUE)
plot(st_geometry(xy),pch=19, cex = 0.25, col="black", add = TRUE)


# Generate spatially inhibitory sample
# with close pairs (cp.zeta criterion):
set.seed(15892)
xy.sample3 <- discrete.inhibit.sample(obj=xy, size = 100,delta = 0.065,
                                      k = 25,cp.criterion = "cp.zeta",
                                      zeta = 0.025, plotit = TRUE)
plot(st_geometry(xy),pch=19, cex = 0.25, col="black", add = TRUE)


# Generate spatially inhibitory sample
# with close pairs (cp.neighb criterion):
set.seed(15892)
xy.sample4 <- discrete.inhibit.sample(obj=xy,size = 100,
                                      delta = 0.065, k = 25,cp.criterion = "cp.neighb",
                                      plotit = TRUE)
plot(st_geometry(xy),pch=19, cex = 0.25, col="black", add = TRUE)


# Generate spatially inhibitory sample
# with close pairs (cp.zeta criterion):
set.seed(15892)
xy.sample5 <- discrete.inhibit.sample(obj=xy,size = 100,
                                      delta = 0.065, cp.criterion = "cp.zeta",
                                      zeta = 0.025, delta.fix = TRUE,
                                      k = 25, plotit = TRUE)
plot(st_geometry(xy),pch=19, cex = 0.25, col="black", add = TRUE)


# Generate simple inhibitory sample from a regular grid
library("PrevMap")
```

```
data("sim.data")
set.seed(15892)
xy.sample6 <- discrete.inhibit.sample(obj = sim.data,
                                      size = 50, delta = 0.08,plotit = TRUE)
plot(st_geometry(sim.data),pch=19,col="black", cex = 0.25, add = TRUE)


# Generate inhibitory plus close pairs sample from a regular grid
set.seed(15892)
xy.sample7 <- discrete.inhibit.sample(obj = sim.data,
                                      cp.criterion = "cp.neighb", size = 50,
                                      delta = 0.1, k = 5, plotit =TRUE)
plot(st_geometry(sim.data),pch=19,col="black", cex = 0.25, add = TRUE)
```

`majete` *Majete malaria prevalence data*

Description

This data-set relates to malaria prevalence study conduted in Majete (Chikwawa), southern Malawi. The variables are as follows:

- `rdt`: Rapid diagnostic test result; 0 = negative, 1 = positive.
- `age`: Age of the individual in months.
- `quintile`: Wealth quintile; ranging from 1 = poor to 5 = well to do.
- `itn`: Insecticide treated bed-net usage; 0 = no, 1 = yes.
- `elev`: Elevation; height above sea level in meters.
- `ndvi`: Normalised difference vegetation index (greenness).
- `agecat`: Age category; 1 = child, 2 = adult.
- `geometry`: Point or household locations (UTM).

Usage

```
data("majete")
```

Format

A data frame with 747 features and 7 variables

References

Kabaghe A N, Chipeta M G, McCann R S, Phiri K S, Van Vugt M, Takken W, Diggle P J, and Terlouw D J. (2017). Adaptive geostatistical sampling enables efficient identification of malaria hotspots in repeated cross-sectional surveys in rural Malawi, *PLoS One* **12**(2) pp. e0172266

`random.sample` *Spatially random sample*

Description

This function draws a spatially random sample from a discrete set of units located over some defined geographical region or generate completely spatially random points within a polygon.

Usage

```
random.sample(obj = NULL, poly = NULL, type, size, plotit = TRUE)
```

Arguments

`obj`	a `sf` or `sp` object (with $N \geq$ `size`) where each line corresponds to one spatial location. It should contain values of 2D coordinates, data and, optionally, covariate(s) value(s) at the locations. This argument must be provided when sampling from a `"discrete"` set of points, see `'type'` below for details.
`poly`	'optional' a `sf` or `sp` polygon in which to generate the design. The default is the bounding box of points given by `obj`. When sampling from a `"continuum"`, the argument `'poly'` must be provided.
`type`	random sampling, a choice of either `"discrete"`, from a set of N potential sampling points or `"continuum"` from independent, compeletely random points.
`size`	a non-negative integer giving the total number of locations to be sampled.
`plotit`	'logical' specifying if graphical output is required. Default is `plotit = TRUE`.

Value

a `sf` or `sp` object of dimension n by `p=dim(obj)[2]` containing the final sampled locations and any associated values, if sampling from a `"discrete"` set of points. A matrix of n by 2 containing sampled locations, if sampling from a `"continuum"`.

Author(s)

Michael G. Chipeta <mchipeta@mlw.mw>

Peter J. Diggle <p.diggle@lancaster.ac.uk>

References

Rowlingson, B. and Diggle, P. 1993 Splancs: spatial point pattern analysis code in S-Plus. Computers and Geosciences, 19, 627-655

Examples

```
# 1. Sampling from a discrete set of points.
library("dplyr")
x <- 0.015+0.03*(1:33)
xall <- rep(x,33)
yall <- c(t(matrix(xall,33,33)))
xy <- cbind(xall,yall)+matrix(-0.0075+0.015*runif(33*33*2),33*33,2)
colnames(xy) <- c('X','Y')

# Convert to SF
xy <- xy %>%
  as.data.frame %>%
  sf::st_as_sf(coords = c(1,2))
xy <- sf::st_as_sf(xy, coords = c('X', 'Y'))


# Sampling from a discrete set.
set.seed(15892)
xy.sample <- random.sample(obj = xy, size = 100, type = "discrete", plotit = TRUE)


# Sampling from a continuum.
library("geoR")
data("parana")
poly <- parana$borders
poly <- matrix(c(poly[,1],poly[,2]),dim(poly)[1],2,byrow=FALSE)
# Convert matrix to polygon
poly <- st_sf(st_sfc(st_polygon(list(as.matrix(poly)))))

set.seed(15892)
xy.sample <- random.sample(poly = poly,size = 100, type = "continuum", plotit = TRUE)
```

`sim.data` *Simulated binomial data-set over the unit square*

Description

This binomial data-set was simulated by generating a zero-mean stationary Gaussian process over a 35 by 35 grid covering the unit square with Matern correlation sturcture. The parameters used in the simulation are $\sigma^2 = 0.7$, $\phi = 0.15$, $\kappa = 1.5$ and $\tau^2 = 0$. The nugget effect was not included, hence `tau2 = 0`. The variables are as follows:

- `data` simulated values of the Gaussian process.
- `y` binomial observations.
- `units.m` binomial denominators.
- `geometry` X and Y coordinates.

Usage

```
data("sim.data")
```

Format

A data frame with 1225 rows and 5 variables

Index

∗ **datasets**
border, 6
majete, 13
sim.data, 15

adaptive.sample, 2

border, 6

contin.inhibit.sample, *2*, *4*, 7

discrete.inhibit.sample, *2*, *4*, *8*, 9

majete, 13

random.sample, *2*, *8*, 14

sim.data, 15