Package: MBHdesign (via r-universe)

August 21, 2024

Title Spatial Designs for Ecological and Environmental Surveys

Version 2.3.15

Author Scott D. Foster

Description Provides spatially survey balanced designs using the quasi-random number method described Robinson et al. (2013) <doi:10.1111/biom.12059> and adjusted in Robinson et al. (2017) <doi:10.1016/j.spl.2017.05.004>. Designs using MBHdesign can: 1) accommodate, without substantial detrimental effects on spatial balance, legacy sites (Foster et al., 2017 <doi:10.1111/2041-210X.12782>); 2) be based on points or transects (foster et al. 2020 <doi:10.1111/2041-210X.13321> and produce clustered samples (Foster et al. (in press). Additional information about the package use itself is given in Foster (2021) <doi:10.1111/2041-210X.13535>.

Maintainer Scott Foster <scott.foster@data61.csiro.au>

License GPL (>= 2)

Depends R (>= 4.1.0)

Imports mgcv, geometry, randtoolbox, mvtnorm, stats, class, parallel, grDevices, graphics, terra

Suggests spsurvey, MASS, fields, knitr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-25 09:50:13 UTC

Contents

`alterInclProbs` *Alters inclusion probabilities to accommodate legacy sites*

Description

Alters inclusion probabilities to accommodate legacy sites. Inclusion probabilities are deflated around legacy sites, and the resulting set of new sites (and legacy sites) are spatially balanced.

Usage

```
alterInclProbs( legacy.sites, potential.sites=NULL, n=NULL, inclusion.probs=NULL,
                                                    mc.cores=1, sigma=NULL)
```

Arguments

`legacy.sites`	a matix (MxD) matrix of locations of the M legacy sites as points in D dimensions. For most applications D=2. Each row gives the location (in space) of one of the legacy sites.
`potential.sites`	a matrix (NxD) of locations of the N potential sampling sites as points in D dimension. These are the locations from which n are taken as the sample. If NULL (default) a maximum of N=10000 samples are placed on a regular grid within a convex hull defined by the legacy locations. This default may or may not make sense for you (but something has to be the default).
`n`	an integer specificying the number of new sites to sample. One of n and inclusion.probs needs to be specified, but if both are then n is discarded.
`inclusion.probs`	a vector specifying the inclusion probability for each of the N sampling sites. This is the probability that each site will be included in the final sample. The sum of inclusion.probs must be the number of new sites required. Locations in inclusion.probs must be ordered the same as the potential.sites argument. If NULL (default) equal inclusion probabilities are specified and the number of new sites is taken to be n.
`mc.cores`	the number of processes to for some of the calculations on (in particular the calculation of distances to legacy sites). See parLapply(qv) in package parallel.
`sigma`	a parameter defining the spatial influence of the legacy sites. Must be positive. It is sigma in the squared exponential decay function, exp(-distanceFromLegacySite^2 / sigma^2). If NULL (default), then it is chosen so that 95 percent of the legacy sites influence is located within the average patch size around a point. This seems like a useful default. See Foster et al. (in prep) for details.

Details

The inclusion probabilities are adjusted using a variant of the spatially clustered Poisson sampling method in Grafstrom (2012), which is itself a spatially explicit version of Bondesson and Thorburn (2008). The adjustments here are given in Foster et al. (in prep). Basically, the adjustment is similar to that proposed in Section 3.2 of Grafstrom (2012), that is the inclusion probabilities are updated using the squared loss distance metric (as above). However, the weighting function here is given by the distance times the inclusion probability of the new site.

Value

The `alterInclProbs` function returns a `numeric` vector containing inclusion probabilities adjusted for legacy sites for all of the points in the `potential.sites` argument.

Author(s)

Scott D. Foster

References

Bondesson, L. and Thoburn, D. (2008) A List Sequential Sampling Method Suitable for Real-Time Sampling. Scandinavian Journal of Statistics *35*:466–483.

Foster, S.D., Hosack, G.R., Lawrence, E., Przeslawski, R., Hedge,P., Caley, M.J., Barrett, N.S., Williams, A., Li, J., Lynch, T., Dambacher, J.M., Sweatman, H.P.A, and Hayes, K.R. (2017) Spatially-Balanced Designs that Incorporate Legacy Sites. Methods in Ecology and Evolution *8*:1433–1442.

Grafstrom, A. (2012) Spatially correlated Poisson sampling. Journal of Statistical Plannind and Inference *142*:139–147.

See Also

`quasiSamp`, `modEsti`

Examples

```
#big plane today
set.seed(747)
#the number of potential sampling locations
N <- 50^2
#number of samples
n <- 27
#number of legacy sites
nLegacy <- 3
#the grid
X <- as.matrix( expand.grid( 1:sqrt( N), 1:sqrt(N)) / sqrt(N) - 1/(2*sqrt(N)))
#the inclusion probabiltiies with gradient according to non-linear function of X[,1]
p <- 1-exp(-X[,1])
#standardise to get n samples
p <- n * p / sum( p)
```

```
#randomly choose legacy sites
legacySites <- sample( 1:N, nLegacy, prob=p)
#alter inclusion probabilities
p2 <- alterInclProbs( legacy.sites=X[legacySites,], potential.sites=X, inclusion.probs=p)
if( requireNamespace( "graphics", quietly = TRUE)) {
  #plotting up old and new inclusion probabilities
  par( mfrow=c(1,2))
  graphics::image( x=unique( X[,1]), y=unique( X[,2]),
    z=matrix( p, nrow=sqrt(nrow(X)), ncol=sqrt(nrow( X))),
    main="Undadjusted Inclusion Probabilities", ylab="y", xlab="x")
  graphics::image( x=unique( X[,1]), y=unique( X[,2]),
    z=matrix( p2, nrow=sqrt(nrow(X)), ncol=sqrt(nrow( X))),
    main="Adjusted Inclusion Probabilities", ylab="y", xlab="x")
  points( X[legacySites,], pch=20, col=1)
}
#tidy
rm( N, n, nLegacy, X, p, legacySites, p2)
```

`alterInclProbs.cluster` *Alters inclusion probabilities to accommodate a 2-stage cluster sampling process*

Description

Alters inclusion probabilities so that the cluster sampling approach respects specified inclusion probabilities.

Usage

```
alterInclProbs.cluster( nCluster, clusterSize, clusterRadius, inclusion.probs,
                                    maxIter=50, tolerance=NULL,
                              mc.cores=parallel::detectCores()-1, doPlot=FALSE)
```

Arguments

`nCluster`	an integer specifying the number of clusters.
`clusterSize`	an integer specifying the number of sites within a cluster.
`clusterRadius`	a numeric scalar giving the geographical extent of each cluster (from cluster centre to most extreme). Must be specified in metres (see `extract` and `buffer`).
`inclusion.probs`	a RasterLayer specifying the inclusion probability for each of the potential sampling sites. The values contained are the probability that each site will be included in the final sample. The values of this rasterLayer will be scaled so that they sum to ncluster*clusterSize. This is due to the alteration not depending on the number of clusters (otherwise they would be scaled to clusterSize * nCLuster. Must be specified.

`maxIter` an integer giving the maximum number of iterations that the algorithm is allowed to perform before termination.

`tolerance` a scalar numeric specifying the desired tolerance between observed inclusion probabilities (from 2-stage cluster sampling process) and the specified inclusion probabilities (from user). This tolerance specifies the accuracy with which the working inclusion probabilities are calculated.

`mc.cores` an integer defining the number of processes to for some of the calculations on (in particular the calculation of distances to legacy sites). See parLapply(qv) in package parallel. Defaut is quite greedy, all but one of the cores available.

`doPlot` a boolean indicating if a set of diagnostic plots should be produced within each iteration. Can be useful to help identify problems with algorithm convergence.

Details

Using the spatially-balanced 2-stage clustering process of Foster et al (in prep) will not, without careful consideration and adjustments, respect the inclusion probabilities specified by the user. This function provides the means to create _working_ inclusion probabilities, which when used in the 2-stage process _will_ produce the specified inclusion probabilies.

Value

The output of this function is designed for direct input into `quasiSamp.cluster`. It contains a SpatRaster with four layers: the user-specified inclusion probabilities (IP.s), the observed inclusion probabilities (IP.o, those inclusion probabilities obtained when using 2-stage sampling with the working inclusion probabilities), the working inclusion probabilities and the area-summed working inclusion probabilities (IP.bar). The IP.bar are based on the working inclusion probabilities and are used to sample the centre of clusters.

Author(s)

Scott D. Foster

References

Foster, S.D., Lawrence, E., and Hoskins, A. (in prep). Spatially-Balanced Cluster Sampling.

See Also

`quasiSamp.cluster`

`ConstrainTransects` *Finds transects that are descending over complex terrain, or are stemming from a set of locations.*

Description

Creates a 'mask' for the extended sampling frame (sites by rotations) for two reasons: Some sampling gears that can only be deployed down-slope, and those designs where a 'wagon-wheel' is required from a small set of points (e.g. the top of a seamount).

Usage

```
findDescendingTrans( potential.sites, bathy, in.area, descend.cutoff=0, control=NULL)
 findTransFromPoint( potential.sites, originPoints, in.area, control = NULL)
```

Arguments

`potential.sites`	a matrix (of size Nxdimension) of the spatial coordinates of the N sampling locations, of which n«N are taken as the sample. If NULL (default) N=10000 samples are placed on a regular grid. If study.area is defined, then this grid is over the smallest bounding (hyper-)rectangle for the study.area. If study.area is NULL, the grid is over the unit (hyper-)square.
`bathy`	a vector of length N, giving the depth (or height) at each of the N potential.sites. Ordering must be the same as potential.sites.
`in.area`	a vector describing which cells are in the study region (TRUE) and which are not (FALSE). Ordering must be the same as potential.sites.
`descend.cutoff`	the value describing how down a descent actually is (in some cases slight rises are still manageable). Default is 0, meaning that strict descent is required.
`control`	A list containing details of the transects, their computational representation, and decisions about the design. See the Details Section of `transectSamp` for more information.
`originPoints`	a matrix with two columns giving the locations of the centre cells. There can be as many central cells as desired (more rows).

Details

The function findDescendingTrans aims to provide a set of transects, which all can be performed down slope. The function findTransFromPoint aims to provide a set of transects, which all start from a particular point (or set of points). The output of these functions is formatted so that it can be directly used as an argument to `transectSamp`. In this case, and when supplied to `transectSamp`, a set of descending or radially stemming, spatially-balanced transects are generated.

This function implements a type of constraint on the randomisation process. Discussed in Foster et al (2020).

Value

The findDescendingTrans function returns a matrix giving a character matrix of size N x nRotate that specifies which transects are 'descend'ing (with N centre locations and nRotate different orientations), which transects are 'upAndDown' slope, which contains only NAs (where bathymetry data is not available), which are descending over the available bathymetry data ('descendAndNA'), which are up and down over the available data ('upAndDownAndNA_NaN').

The findTransFromPoint function similarly returns a character matrix containing the strings "startsFromPoint" meaning that the transect starts form one of the specified points, and "badStart" meaning that the transect does not start from one of the specified points. The developer realises that "badStart" is not an extremely informative string – it must have seemed like a good idea late one night.

Author(s)

Scott D. Foster

References

Foster, S.D., Hosack, G.R., J. Monk, Lawrence, E., Barrett, N.S., Williams, A. and Przeslawski, P. (2020) Spatially-Balanced Designs for Transect-Based Surveys. Methods in Ecology and Evolution. emph11: 95–105

See Also

transectSamp

`modEsti` *Get a model-based estimate of mean of a sampled area*

Description

For a given survey design in any number of dimensions, calculate the mean prediction (plus SE plus 95% CI) for the area.

Usage

```
modEsti( y, locations, includeLegacyLocation=TRUE, legacyIDs=NULL, predPts=NULL,
  family=stats::gaussian(), offset=rep(0,length(y)), control=list())
```

Arguments

`y`	a numeric vector of all observations (at new sites and legacy sites) collected in the survey
`locations`	a matrix (or something that can be coerced) containing the set of locations where observations were collected. Note that nrow(locations) == length(y).

`includeLegacyLocation`
: a boolean indicating whether an extra term should be included into the model that corresponds to distance from legacy sites. See Foster et al (2017) for details. If TRUE (default) the extra term is included. If FALSE (so that model is purely spatial), then the extra term is discarded.

`legacyIDs`
: the indexes, for the rows of y and locations, that correspond to legacy sites. For example, if the first, third and sixth rows were legacy sites in y and locations, then legacyIDs would be c(1,3,6).

`predPts`
: A data.frame (or something that can be coerced to a data.frame) containing set of points to do the predictions at. Typically predPts is a dense grid (or similar) of points over the spatial area of interest. Note that the number of columns defines the number of dimensions. Do not include surplus variables in extra columns. If NULL (default), then a dense grid within the convex hull bounding "locations" will be used.

`family`
: A family object giving the distribution of the data. Default is `gaussian()` but other sensible choices in ecology include `nb(link="log")` and `binomial()`. For more details see `?family.mgcv` from the mgcv package.

`offset`
: A numeric vector of length equal to "length(y)". This gives any offset to the linear predictor of the model.

`control`
: A list of control parameters (see details)

Details

This function works by fitting a generalised additive model (see `gam()`), predicting at the points predPts, and then averaging. Well, that is the general idea. The actual implementation uses a Monte Carlo routine to account for parameter uncertainty. This is done mirroring the helpfile of predict.gam. Basically, lots of sets of parameters are drawn from the parameters (asymptotic) distribution and then predictions are made for each draw. The overall estimate is then the mean (over parameter draws) of the mean (over prediction locations) of the prediction. Standard errors and confidence intervals are likewise calculated.

The control list contains elements with names:

k the number of knot points used in each dimension of locations

N the number of prediction points (in each dimension) for the grid, argument not used if "predPts!=NULL"

B the number of bootstrap samples to take of the parameter estimates

mc.cores the number of computer cores to spread the calculation of distances over (only used if includeLegacyLocation==TRUE)

Value

A list of three elements: 1) a point prediction of mean, 2) standard error of mean (obtained by parametric bootstrap), and 3) 95% confidence interval of the mean.

Author(s)

Scott D. Foster

References

Foster, S.D., Hosack, G.R., Lawrence, E., Przeslawski, R., Hedge,P., Caley, M.J., Barrett, N.S., Williams, A., Li, J., Lynch, T., Dambacher, J.M., Sweatman, H.P.A, and Hayes, K.R. (2017) Spatially-Balanced Designs that Incorporate Legacy Sites. Methods in Ecology and Evolution *8*:1433–1442.

See Also

quasiSamp, alterInclProbs, and `total.est` from the **spsurvey** package. The total.est function provides design-based estimation of both mean and variance.

Examples

```
#set up design parameters
#taken from the example in alterInclProbs()
#big plane today
set.seed(747)
#the number of potential sampling locations
N <- 50^2
#number of samples
n <- 27
#number of legacy sites
nLegacy <- 3
#the grid
X <- as.matrix( expand.grid( 1:sqrt( N), 1:sqrt(N)) / sqrt(N) - 1/(2*sqrt(N)))
#the inclusion probabiltiies with gradient according to non-linear function of X[,1]
p <- 1-exp(-X[,1])
#standardise to get n samples
p <- n * p / sum( p)
#randomly choose legacy sites
legacySites <- sample( 1:N, nLegacy, prob=p)
#alter inclusion probabilities for legacy sites
p2 <- alterInclProbs( legacy.sites=X[legacySites,], potential.sites=X, inclusion.probs=p)
#get the sample
samp <- quasiSamp( n=n, dimension=2, potential.sites=X, inclusion.probs=p2)
samp <- rbind( cbind( X[legacySites,], inclusion.probabilities=NA, ID=NA), samp)
#generate some fake data
samp$outcomes <- rnorm( nrow( samp))
#get the estimate
esti <- modEsti( y=samp$outcomes, locations=samp[,1:2], includeLegacyLocation=TRUE,
  legacyIDs=1:3, predPts=NULL, family=gaussian(), control=list(mc.cores=1, B=100))
#in real applications the number of bootstrap samples (B), and mc.cores, should be larger
print( esti)
#tidy
rm( esti, legacySites, n, N, nLegacy, p, p2, samp, X)
```

quasiSamp *Generates a spatial design using Quasi-random numbers*

Description

Generates a spatially balanced design for given inclusion probabilities over a grid of potential sampling locations

Usage

```
quasiSamp( n, dimension=2, study.area=NULL, potential.sites=NULL, inclusion.probs=NULL,
                                randStartType=3, nSampsToConsider=25*n,
   nStartsToConsider=100*n)
quasiSamp.raster( n, inclusion.probs, randStartType=3,
   nSampsToConsider=25*n, nStartsToConsider=100*n)
quasiSamp.cluster( nCluster, clusterSize, clusterRadius, inclusion.probs=NULL,
                                working.inclusion.probs=NULL,
   nSampsToConsider=c(25*nCluster,25*clusterSize),
   nStartsToConsider=100*c(nCluster, clusterSize),
   mc.cores=parallel::detectCores()-1)
```

Arguments

n	the number of samples to take.
nCluster	the number of clusters to sample. Only used when sampling sites in cluster groups.
clusterSize	the number of sites within each cluster to sample.
clusterRadius	the radius of each individual clusters. Sites within a cluster will be chosen so that they are at most a distance of clusterRadius from the cluster centre. Must be specified in metres (see extract and buffer).
dimension	the number of dimensions that the samples are located in. Equal to 2 for areal sampling. Care should be taken with large dimensions as:1) the number of potential sampling sites needed for effective coverage starts to explode (curse of dimensionality); and 2) the well-spaced behaviour of the Halton sequence starts to deteriorate (but this requires very very many dimensions to be problematic – included as a warning here for largely academic reasons).
study.area	a numeric matrix with dimension columns. This defines the sampling area from where the sites are selected – each row defines a vertex of the sampling area and the order of rows is such that the vertices are joined in order. The last vertex is joined to the first. If NULL (default), the study.area is defined to be the smallest (hyper-)rectangle that bounds the potential.sites. If potential.sites is also NULL (default), then the study area is taken to be the unit (hyper-)square. This argument is closely related to potential.sites.

potential.sites
: a matrix (of size Nxdimension) of the spatial coordinates of the N sampling locations, of which n«N are taken as the sample. If NULL (default) N=10000 samples are placed on a regular grid. If study.area is defined, then this grid is over the smallest bounding (hyper-)rectangle for the study.area. If study.area is NULL, the grid is over the unit (hyper-)square.

inclusion.probs
: either a vector or a SpatRaster specifying the inclusion probability for each of the N potential sampling sites. For quasiSamp.raster and for quasiSamp.cluster, inclusion.probs must be a raster. The values contained are the probability that each site will be included in the final sample. Note that inclusion.probs will be scaled internally so that they sum to the number of sites to be sampled. If a vector, then the locations must be ordered the same as the potential.sites argument. If NULL and a vector (default) equal inclusion probabilities are specified. Must be specified for quasiSamp.raster. If quasiSamp is called with a SpatRaster value, then quasiSamp will internally call quasiSamp.raster and ignore the dimension, study.area and potential.sites arguments. Argument is ignored in quasiSamp.cluster if (and only if) working.inclusion.probs is also supplied. If not ignored in quasiSamp.cluster, then `alterInclProbs` is called internally.

working.inclusion.probs
: a SpatRaster of relevant inclusion probabilities. In particular, the working inclusion probabilities and their local (geogrphic) sums, which are uased in place of specified inclusion probabilities to ensure that cluster sampling respects the specified inclusion probabilities. See Foster et al (in review) for technical details. It is expected that, by far, the easiest way to generate this object is via a call to `alterInclProbs.cluster`. If working.inclusion.probs is not supplied then inclusion.probs must be. The function `alterInclProbs.cluster` internally performs the desired scaling of the inclusion probabilties, so this function (`quasiSamp.cluster`) does not perform any extra checks.

randStartType
: the type of random start Halton sequence to use. The choices are 3 (default) as recommended in Robertson et al (2017), which improves the match between observed and specified inclusion probabilities (i.e. you get closer to what you want). Other options are 2 which gives the process in Robertson et al (2013), and 1 which is a mis-interpretation of method 2 (constrained so that the size of the skip in each dimension is equal). Note that randStartType=1 is used in Foster et al (2017).

nSampsToConsider
: the total number of samples to consider in the BAS step (rejection sampling). The default is 25*n, which means that 25*n halton numbers are drawn and then thinned according to the inclusion probabilities. Users will want to increase this number if inclusion probabilities are extremely unbalanced or if the number of samples required is close to, or exceeds, 25*n. Reduce if you want the code to run quicker and are confident that a sample will be found using less. For quasiSamp.cluster, nsampsToConsider is a vector of length 2. The first element specifies the number of samples to consider for sampling clusters. The second element specifies the number of samples for sampling within each cluster.

nStartsToConsider
: (only used when randStartType=3). For quasiSamp and quasiSamp.raster: The

maximum number of times the randomisation process should be performed before giving up. Default is 100*n. If this is not enough, then consider increasing it (and probably waiting longer for your computer to finish). For quasiSamp.cluster: a two element numeric vector giving the number of starts to attempt at each level of the randomisation process.

`mc.cores` When quasiSamp.cluster is called without a working.inclusion.probs argument (NULL), then `alterInclProbs.cluster` is called with this many cores used.

Details

These function are an implementation of the balanced adaptive sampling (BAS) designs presented in Robertson et al. (2013) and Robertson et al. (2017). The former forms the basis for the methods in Foster et al (2017) and the latter is a modification of the former. The BAS approach uses Halton sequences of quasi-random numbers, which are evenly spread over space, as the basis for generating spatially balanced designs. In this implementation, we requrie that the inclusion probabilities be given as points in space and the BAS design is the set of these points that lie closest to a continuous-space Halton sequence. Computational speed has been rudimentily optimised, but (of course) it could be done better.

In an updated version of the package (Version 2.2.1 onwards) a raster can be passed to the function. Post 2.3.14, this must be a SpatRaster object from pacakge terra (prior a RasterLayer from pacakge raster). This may be both more convenient and it will be faster for very large design problems. Note though that the underlying algorithm, and much of the code, remains unchanged between the two different versions.

From version 2.3.0 onwards, the spatial cluster sampling approach of Foster et al (in review) is implemented in quasiSamp.cluster. This method proceeds in a two-staged fashion: cluster centres are chosen and then sites are chosen within clusterRadius of these centres. Both stages are chosen using quasi random numbers in BAS (Robertson et al; 2013).

In the edge case, where the number of samples is larger than the number of potential sampling points (or raster cells), the quasiSamp functions will simply sample sites multiple times. This behaviour may also be exhibited for cells with very high inclusion probabilties too, even when the sample size is larger than the number of potential sample sites. For cluster sampling, using `quasiSamp.cluster`, this also applies to within cluster sampling, as well as between cluster sampling.

Value

The quasiSamp and quasiSamp.raster functions returns a matrix of (dimension+2) columns. The first columns (of number dimension) are the sampled sites locations. The second to last column contains the inclusion probabilities for the sampled locations. The last column is the row number (of potential.sites) that corresponds to that sampled site.

The quasiSamp.cluster function returns a SpatialPointsDataFrame. It contains an identifier for cluster and site within cluster, the cellID from the original (inclusion.probs or working.inclusion.probs) raster, the specified inclusion probability for the cell, the cluster probability for a cluster centred at that cell, the conditional probability of sampling each cell within that cluster, and the working inclusion probabilities. The return object also contains a SpatialPointsDataFrame containing the design for the cluster.

Author(s)

Scott D. Foster

References

Robertson, B. L., Brown, J. A., McDonald, T. and Jaksons, P. (2013) BAS: Balanced Acceptance Sampling of Natural Resources. Biometrics *69*: 776–784.

Robertson, B.; McDonald, T.; Price, C. & Brown, J. (2017) A modification of balanced acceptance sampling Statistics and Probability Letters, *129*, 107–112

Foster, S.D., Hosack, G.R., Lawrence, E., Przeslawski, R., Hedge,P., Caley, M.J., Barrett, N.S., Williams, A., Li, J., Lynch, T., Dambacher, J.M., Sweatman, H.P.A, and Hayes, K.R. (2017) Spatially-Balanced Designs that Incorporate Legacy Sites. Methods in Ecology and Evolution *8*:1433–1442.

Foster, S.D., Lawrence, E., and Hoskins, A. (2023). Spatially Clustered Survey Designs. Journal of Agriculural, Biological and Environmental Statistics.

See Also

alterInclProbs, modEsti, alterInclProbs.cluster

Examples

```
#generate samples on a 100 x 100 grid
#Note that, although the random number is set, there may be differences between versions of R.
#In particular, post R/3.6 might be different to R/3.5 and before
#jet plane
set.seed(707)
#the number of potential sampling locations
N <- 100^2
#number of samples
n <- 10
#the grid on unit square
X <- as.matrix( expand.grid( 1:sqrt( N), 1:sqrt(N)) / sqrt(N) - 1/(2*sqrt(N)))
#the inclusion probabiltiies with gradient according to non-linear function of X[,1]
p <- 1-exp(-X[,1])
#standardise to get n samples
p <- n * p / sum( p)
#get the sample
samp <- quasiSamp( n=n, dimension=2, potential.sites=X, inclusion.probs=p)
par( mfrow=c(1,3), ask=FALSE)
plot( samp[,1:2], main="n=10")
#now let's get sillier
n <- 250
#get the sample
samp <- quasiSamp( n=n, dimension=2, potential.sites=X, inclusion.probs=p)
plot( samp[,1:2], main="n=250")
#silly or sublime?
n <- 1000
#get the sample
samp <- quasiSamp( n=n, dimension=2, potential.sites=X, inclusion.probs=p, nSampsToConsider=5000)
```

```
plot( samp[,1:2], main="n=1000")
#I'm sure that you get the idea now.

##The same for raster inclusion probabilities (just for illustration)
#Xp <- terra::rast( cbind( X,p), type='xyz')
#samp <- quasiSamp( n=10, inclusion.probs=Xp)
#plot( samp[,1:2], main="n=10 (raster)")
#samp <- quasiSamp( n=250, inclusion.probs=Xp)
#plot( samp[,1:2], main="n=250 (raster)")
#samp <- quasiSamp( n=1000, inclusion.probs=Xp, nSampsToConsider=5000)
#plot( samp[,1:2], main="n=1000 (raster)")

#tidy
rm( N, n, X, p, samp, Xp)
```

`transectSamp` *Generates a spatial design for transects*

Description

For arbitrary transect patterns, generates a spatially balanced design for given inclusion probabilities over a grid of potential sampling locations

Usage

```
transectSamp( n, study.area=NULL, potential.sites=NULL,
     inclusion.probs=NULL, control=NULL, constrainedSet=NULL)
transectSamp.internal( n, study.area=NULL, potential.sites=NULL,
     inclusion.probs=NULL, control=NULL, constrainedSet, ...)
```

Arguments

`n`	the number of transect to provide a sample for
`study.area`	a numeric matrix with dimension columns. This defines the sampling area from where the sites are selected – each row defines a vertex of the sampling area and the order of rows is such that the vertices are joined in order. The last vertex is joined to the first. If NULL (default), the study.area is defined to be the smallest (hyper-)rectangle that bounds the potential.sites. If potential.sites is also NULL (default), then the study area is taken to be the unit (hyper-)square. This argument is closely related to potential.sites.
`potential.sites`	a matrix (of size Nxdimension) of the spatial coordinates of the N sampling locations, of which n«N are taken as the sample. If NULL (default) N=10000 samples are placed on a regular grid. If study.area is defined, then this grid is over the smallest bounding (hyper-)rectangle for the study.area. If study.area is NULL, the grid is over the unit (hyper-)square.

`inclusion.probs`
: a vector specifying the inclusion probability for each of the N potential sampling sites. This is the probability that each site will be included in the final sample. Locations are ordered the same as the potential.sites argument. If NULL (default) equal inclusion probabilities are specified. Note that there is no sum constraint to this vector – inclusion probabilities are relative (this is in contrast to `quasiSamp`).

`control`
: A list containing details of the transects, their computational representation, and decisions about the design. See the Details for more information.

`constrainedSet`
: An N x nRotate Boolean matrix specifying whether the putative transect, starting at each location and orientated along each rotation, should be considered for sampling. See `findDescendingTrans` for an example.

`...`
: Further arguments mostly for reducing computation for when certain quantities have been pre-calculated. Often not useful for general purpose, unless extra constraints are required for certain types of transects. The arguments include the computationally expensive objects (where repeated survey generation may be sped up): IDs – the information about which grid cell each and every sampling location (along all transects) falls; transectsOverCells – information about which sampling locations (from any transect) falls in all cells, and; adjustedSpecified – The specified inclusion probabilities adjusted for edge effects.

Details

This function aims to provide a randomised design for transects that: 1) adheres to inclusion probabilities for each grid cell; 2) is spatially coherent/balanced so that transects are well spread throughout the study area; and 3) is not too computationally expensive (or too devilishly difficult) to use effectively. However, fine grids over large areas will be demanding nevertheless.

The methods implemented by the function are outlined in Foster et al (2020), but are based on the ideas presented in Robertson et al (2013) and as implemented for Foster et al (2017). Briefly, the locations of the centre of the transects are chosen according to BAS sampling (Robertson et al; 2013) and the direction of the sample is then taken randomly. See Foster et al (2020) for further details.

The control list contains the following elements.

- transect.pattern: a set of points describing the transect pattern (a line, or an "S" pattern, or a dense grid, or a quadrat, or ...). This is a generic pattern, which will be rotated as required by the function for the randomisation. The pattern will be centred in both the x and y directions, so this does not need to be performed by the user (unless they choose to). Please specify as a 2 column matrix full of X- and Y-coordinates. A special value, and the default, is not a matrix at all. Rather the specific character string "line", in whcih case the function will produce its own generic pattern (for a line of length control$line.length with control$transect.nPts points along it). All values that are not "line" will be interpreted as a matrix. For symmetric patterns (around the centre point), and to avoid excess computations only, please make sure that an odd number of nRotate is used (see below).
- transect.nPts: The number of points to represent the (linear) transect. If control$transect.pattern is specified to something apart from "line" then this argument is ignored. A large value of transect.nPts will lead to a more accurate representation of the transects, but it will also cost more computation. The default is the arbitrary value of 20.

- line.length: If control$transect.pattern==“line”, then this control parameter gives the length of that line. By default, it is arbitrarily taken to be 0.1 times the maximal edge of the study.area.
- nRotate: The number of different bearings (through North, East, South, West) to consider at each sampling location for directions of the transect. The default is 11 (an odd number), meaning that transects laid out at 0 degrees, 32.7 degrees, 65.5 degrees, 98.2 degrees, ... (from north) will be considered. For symmetric a transect.pattern, such as a linear transect, nRotate should be an odd number (like the default) to avoid excess computation. This is because directions around a centre point can be inadvertently repeated (e.g. a linear transect in the 10 and 190 degree directions lead to the same trasnect). Increasing the number of directions to consider also increases computational time – probably at very modest increase in efficiency or scientific validity.
- mc.cores: The number of computer cores to spread the (parallel) computation to. More decreases the real-world time that the function will use to execute, but this occurs at the expense of decreasing the performance of the computer to run other jobs. It is not recommended that the number of cores on the machine is exceeded. If NULL (the default), the function uses just 1 core. This is not generally recommend as it is the slowest choice.
- transAdjust: Should the user-specified inclusion probabilities be altered so that, in a transect design, each cell has that inclusion probability of being sampled. The default is TRUE – the inclusion probabilities should be altered. See Foster et al (2020) for details.
- edge.max.iter: The maximum number of iterations to perform whilst adjusting inclusion probabilities for transect sampling. Interger >= 1. The default is 25.
- conv.tol.diff: the tolerance for convergence for adjusting inclusion probabilities for transect sampling. Default is 0.01, meaning that successive iterations have to have proportional performance criteria less than 0.0001 apart.
- gamma: the dampening multiplier for successive iterations of the transect adjustment for inclusion probabilities. Default is 0.8. Values should be in [0,1] with larger values likely to be quicker and more unstable.
- calcObsProbs: Should the observed inclusion probabilities be calculated for the design. Default is TRUE if inclusion probabilities are being adjusted (transAdjust is TRUE), and FALSE otherwise (transAdjust is FALSE).
- return.index: Should the (computationally demanding) indexes be returned? These may be useful if multiple surveys designs are being generated; as done in a simulation study. Default is FALSE, and these large objects are not returned.
- spat.random.type: The type of randomisation used to select locations of transects. Default is “quasi”, after Robertson et al (2013) which gives spatially-balanced designs. The other option is “pseudo”, a random sample, which does no spatial balancing.
- nSampsToConsider: A parameter passed to `quasiSamp` controlling the acceptance sampling. Larger numbers are more computationally expensive, but are also more likely to find a design (important for very unequal inclusion probabilities and/or challengingly shaped survey areas). Default is 5000.
- nCells: If no potential sites are given, then some will be created on the unit square. There will be a nCells X nCells grid.

Value

The transectSamp function returns a list of two data.frames. The first data.frame (named “transect”) contains the chosen transects’ middle locations and their directions. The second data.frame (named

"points") contains the set of points that the survey's transects are planned to pass through. These are points on each of the transects and are the points used to represent the transect during the design.

Author(s)

Scott D. Foster

References

Foster, S.D., Hosack, G.R., J. Monk, Lawrence, E., Barrett, N.S., Williams, A. and Przeslawski, P. (2020) Spatially-Balanced Designs for Transect-Based Surveys. Methods in Ecology and Evolution *8*: 95–105.

Foster, S.D., Hosack, G.R., Lawrence, E., Przeslawski, R., Hedge,P., Caley, M.J., Barrett, N.S., Williams, A., Li, J., Lynch, T., Dambacher, J.M., Sweatman, H.P.A, and Hayes, K.R. (2017) Spatially-Balanced Designs that Incorporate Legacy Sites. Methods in Ecology and Evolution *8*:1433–1442.

Robertson, B. L., Brown, J. A., McDonald, T. and Jaksons, P. (2013) BAS: Balanced Acceptance Sampling of Natural Resources. Biometrics *69*: 776–784.

See Also

alterInclProbs,quasiSamp,, findDescendingTrans, findTransFromPoint

Examples

```
## Not run:
#generate samples on a 50 x 50 grid
#Note that, although the random number is set, there may be differences between versions of R.
#In particular, post R/3.6 might be different to R/3.5 and before
#jet plane
set.seed( 767)
#the number of potential sampling locations
N <- 50^2
#number of samples
n <- 10
#the grid on unit square
X <- as.matrix( expand.grid( 1:sqrt( N), 1:sqrt(N)) / sqrt(N) - 1/(2*sqrt(N)))
#the inclusion probabilities with gradient according to non-linear function of X[,1]
p <- 1-exp(-X[,1])
#standardise to get n samples
p <- n * p / sum( p)
#get the sample
#note that 5 points on the transect line and 5 directions considered is a bit thin.
#This low-definition is done to avoid trouble with CRAN's checks
#(no example should take a long time to run).
samp <- transectSamp( n, potential.sites=X, inclusion.probs=p,
      control=list( transect.patter="line", nRotate=5, transect.nPts=5, mc.cores=7))
plot( samp$points[,5:6], main="n=10 TRANSECTS")
#tidy
rm( N, n, X, p, samp)
```

```
## End(Not run)
```

Index