An Introduction to MBHdesign

Scott D. Foster
CSIRO, Hobart, Tasmania, Australia

Abstract

A robust scientific conclusion is the result of a rigorous scientific process. In observational ecology, this process involves making inferences about a population from a sample. The sample is crucial, and is the result of implementing a *survey design*. A good survey design ensures that the data from the survey is capable of answering the research question. Better designs, such as spatially balanced designs, will also be as precise as possible given the constraints of the budget.

The MBHdesign package is useful for creating spatially balanced designs. There are three tasks that it is intended to address: 1) designing spatially-balanced surveys using Balanced Adaptive Sampling (BAS Robertson *et al.* 2013); 2) designing spatially-balanced transect-based surveys using the methods described in Foster *et al.* (in prep), and; 3) designing and analysing spatially-balanced surveys that incorporate existing legacy sites, using Foster *et al.* (2017). The first example in this tutorial generates a point-based design and shows how legacy sites can be incorporated. There are three steps to this process:

1. Altering inclusion probabilities for spatial balance, taking into account the location of legacy sites. This is done using the function `alterInclProbs`;
2. Generating spatially balanced designs for a given set of inclusion probabilities, through the function `quasiSamp`; and
3. Analysing some (made up) data using model-based methods (using `modEsti`).

The second example in this tutorial generates a transect-based design over the same inclusion probabilities. This consists of just one substantive step: calling the `transectSamp` function.

Keywords: Spatially-Balanced Survey Design, Balanced Adaptive Sampling, Transect, Spatially Correlated Poisson Sampling, GRTS, R.

This is an introduction to the MBHdesign R-pacakge (available from https://cran.r-project.org/package=MBHdesign). It has been developed as part of a NESP Marine Biodiversity Hub Project https://www.nespmarine.edu.au/project/project-d2-standard-operating-procedures-survey-design-condition-assessment-and-trend.

First Things First

Before starting with this introduction to `MBHdesign`, we need to make sure that everything is set up properly. Much of this will vary from computer to computer, but you must have a working version of R installed (preferably the recent release). This vignette was created using R version 4.1.1 (2021-08-10). It does not matter whether you prefer to use R through a development environment (such as RStudio) or through the command line – the results will be the same. So, start R and then:

```
install.packages( "MBHdesign")
```

You will be asked which repository you want to use. Just use one that is geographically close to where you are (or where your computer is). Next load the package.

```
library( MBHdesign)
```

For illustration is is also good to fix the random number seed, so that this document is reproducible *exactly*. However, and due to R evoloving, the results may differ between different versions of R. In particular, R/3.6 could be quite different from R/3.5.

```
set.seed( 747)  #a 747 is a big plane
```

Now, we are good to go with the rest of the introduction.

A Point-Based Design

Let's pretend that we want to generate $n = 10$ samples on a grid of points (representing the centres of a tessellation). The grid of points consists of $N = 100 \times 100 = 10000$ points in 2-dimensional space (spanning the interval $[0, 1]$ in both dimensions). Let's also pretend that there are 3 legacy sites, that have been sampled in previous survey efforts, and we wish to revisit them in the current survey. The legacy sites are located at random throughout the study area. Here, I have generated it all in R (painstakingly), but in a real application, most of this information could be read in from file.

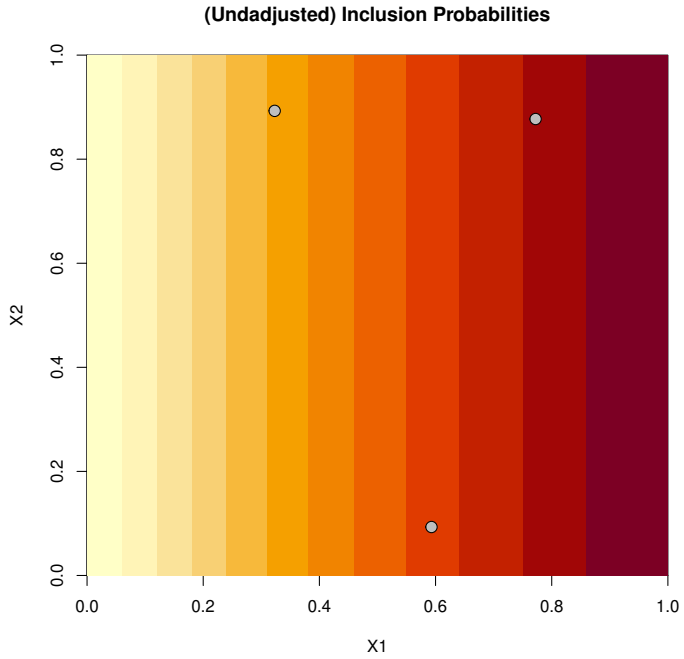
```
#number of samples
n <- 10
#number of points to sample from
N <- 100^2
#the sampling grid (offset so that the edge locations have same area)
offsetX <- 1/(2*sqrt( N))
my.seq <- seq( from=offsetX, to=1-offsetX, length=sqrt(N))
X <- expand.grid( my.seq, my.seq)
#the legacy sites (three of them)
legacySites <- matrix( runif( 6), ncol=2, byrow=TRUE)
#names can be useful
colnames( X) <- colnames( legacySites) <- c("X1","X2")
```

Inclusion Probabilities

Key to this whole design process is the concept of inclusion probabilities. Inclusion probabilities define the chance that any particular site will be part of the sample. So, if a site's inclusion probability is small, then the site is unlikely to be included into the sample. Specifying inclusion probabilities can improve efficiency of the sampling design. That is, standard errors can be reduced for a given number of samples. The 'trick' is to specify inclusion probabilities so that the sites that should have highly variable observations are sampled more often (e.g. Grafström and Tillé 2013). In ecology, variance often increases with abundance (due to Taylor's Power Law; Taylor 1961), so inclusion probabilities could be increased with abundance. If there is no knowledge about the area being sampled, then all sites should be given equal inclusion probabilities (equal to $\frac{n}{N}$). The only formal requirement, in terms of `MBHdesign`, is that the inclusion probabilities must sum to n.

Here, we are going to pretend that there is some gradient in the variance of the population under study. We stress that this is illustrative only.

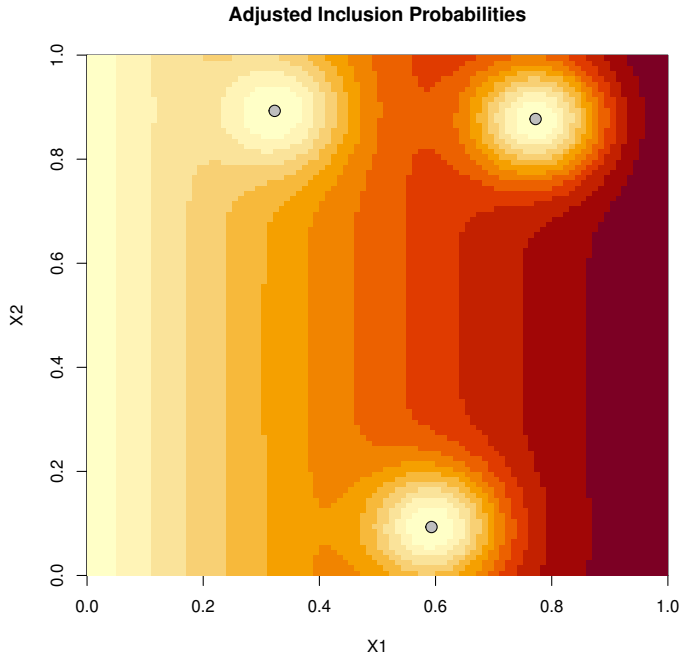
```
#non-uniform inclusion probabilities
inclProbs <- 1-exp(-X[,1])
#scaling to enforce summation to n
inclProbs <- n * inclProbs / sum( inclProbs)
#uniform inclusion probabilities would be inclProbs <- rep( n/N, times=N)
#visualise
image( x=unique( X[,1]), y=unique( X[,2]),
    z=matrix( inclProbs, nrow=sqrt(nrow(X)), ncol=sqrt(nrow( X))),
    main="(Undadjusted) Inclusion Probabilities",
    ylab=colnames( X)[2], xlab=colnames( X)[1])
#The legacy locations
points( legacySites, pch=21, bg=grey(0.75), cex=1.5)
```



Accommodating Legacy Sites

To generate a design that is spatially balanced *in both the n new sample sites and the legacy sites*, we adjust the inclusion probabilities. The adjustment (see Foster *et al.* 2017) reduces the inclusion probabilities so that sites near legacy sites are less likely to be chosen in the new sample.

```
#alter inclusion probabilities
#   so that new samples should be well-spaced from legacy
altInclProbs <- alterInclProbs( legacy.sites=legacySites,
                potential.sites=X, inclusion.probs = inclProbs)
#visualise
image( x=unique( X[,1]), y=unique( X[,2]),
    z=matrix( altInclProbs, nrow=sqrt(nrow(X)), ncol=sqrt(nrow( X))),
    main="Adjusted Inclusion Probabilities",
    ylab=colnames( X)[2], xlab=colnames( X)[1])
#The legacy locations
points( legacySites, pch=21, bg=grey(0.75), cex=1.5)
```



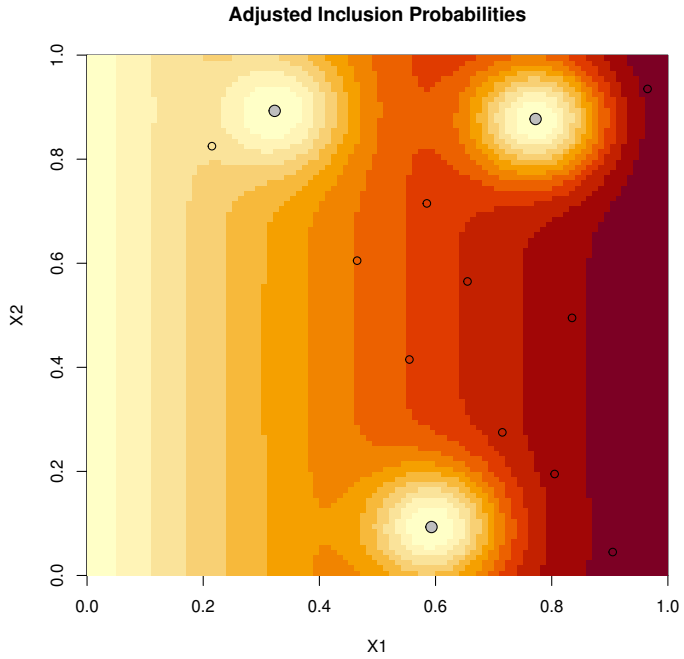
So, the inclusion probabilities have been reduced around the legacy sites. It is perhaps worth noting that the reduction in inclusion probabilities, due to the legacy sites, can be viewed as *sequential*. This means that the reduction for any legacy site is in addition to the reduction of all of the other legacy sites – there is no extra joint effect. Also, the adjustment is proportional to the original inclusion probability, so that a small inclusion probability and a large inclusion probability are both adjusted proportionally to the same amount.

There are some other arguments to the `altInclProbs()` function (omitted for clarity here). These can be seen to refine the call and/or to make the computer to do its work quicker. Type `?altInclProbs` for more details.

Generating the Design

Irrespective of how the inclusion probabilities were obtained, we can now use them to generate a spatially balanced design.

```
#generate the design according to the altered inclusion probabilities.
samp <- quasiSamp( n=n, dimension=2,
        study.area=matrix( c(0,0, 0,1, 1,1, 1,0),ncol=2,  byrow=TRUE),
        potential.sites=X, inclusion.probs=altInclProbs)
#visualise
image( x=unique( X[,1]), y=unique( X[,2]),
    z=matrix( altInclProbs, nrow=sqrt(nrow(X)), ncol=sqrt(nrow( X))),
    main="Adjusted Inclusion Probabilities",
    ylab=colnames( X)[2], xlab=colnames( X)[1])
#The legacy locations
points( legacySites, pch=21, bg=grey(0.75), cex=1.5)
points( samp[,1:2], pch=21)
```



Voilà! A spatially balanced design that incorporates legacy sites. It is contained in the object `samp`, which looks like:

```
print( samp, row.names=FALSE)

    X1    X2 inclusion.probabilities   ID
 0.655 0.565            0.0014877155 5666
 0.905 0.045            0.0018430432  491
 0.585 0.715            0.0013607580 7159
 0.835 0.495            0.0017528447 4984
 0.215 0.825            0.0004362412 8222
 0.715 0.275            0.0015475672 2772
 0.465 0.605            0.0011510759 6047
 0.965 0.935            0.0018418561 9397
 0.555 0.415            0.0013184766 4156
 0.805 0.195            0.0016915169 1981
```

The columns of `samp` are:

- The sample locations in the `X1` and `X2` dimensions;
- The inclusion probability for that sampling location; and
- The row number (ID), of the original list of potential sites (X).

Analysis

After finalising the design, time comes to go and undertake the survey. For illustration, we do

this *in silico* and generate observations according to a pre-defined function (following Foster *et al.* 2017, amongst others).

```
#generate some `observations' for the new sites
Z <- 3*( X[samp$ID,1]+X[samp$ID,2]) +
                        sin( 6*( X[samp$ID,1]+X[samp$ID,2]))
#and some for the legacy sites
Zlegacy <- 3*( legacySites[,1]+legacySites[,2]) +
                        sin( 6*( legacySites[,1]+legacySites[,2]))
```

These data can be analysed in two ways: 1) design-based, which uses minimal assumptions about the data; and 2) model-based, which attempts to describe more aspects of the data. See Foster *et al.* (2017) for a more complete description. For design-based analysis we take a weighted average of the estimator for the legacy sites and the estimator for the new sites. In both cases the estimates follow Horvitz and Thompson (1952). Please do read the section in Foster *et al.* (2017) for comments on estimation, it could save you some grief.

```
#the proportion of legacy sites in the whole sample
fracLegacy <- nrow( legacySites) / (n+nrow( legacySites))
#inclusion probabilities for legacy sites
#   (these are just made up, from uniform)
LegInclProbs <- rep( nrow( legacySites) / N, nrow( legacySites))
#estimator based on legacy sites only
legacyHT <- (1/N) * sum( Zlegacy / LegInclProbs)
#estimator based on new sites only
newHT <- (1/N) * sum( Z / inclProbs[samp$ID])
mean.estimator <- fracLegacy * legacyHT + (1-fracLegacy) * newHT
#print the mean
print( mean.estimator)

[1] 3.095743
```

This is pretty close to the true value of 2.9994. To get a standard error for this estimate, we use the `cont_analysis()` function from the `spsurvey`* (Dumelle *et al.* 2021), which implements the neighbourhood estimator of Stevens and Olsen (2003).

```
#load the spsurvey package
library( spsurvey)
#rescale the inclusion probs
#   (the sample frames are the same in legacy and new sites)
tmpInclProbs <- ( c( inclProbs[samp$ID], LegInclProbs) / n) *
                                                (n+nrow(legacySites))
#create a temporary data frame
tmpDat <- data.frame( siteID=c( samp$ID, paste0( "legacy", 1:nrow(legacySites))),
                      wgt=1/tmpInclProbs,
```

*In versions of spsurvey, prior to version 5, this was acievied through `total.est()`

```
                      xcoord=c(samp$X1,legacySites[,1]),
                      ycoord=c(samp$X2,legacySites[,2]), Z=c(Z,Zlegacy))
#calculate the standard error
se.estimator <- cont_analysis( tmpDat, vars="Z",
                      weight="wgt",
                      xcoord="xcoord",
                      ycoord="ycoord")$Mean$StdError
#print it
print( se.estimator)

[1] 0.3616072
```

For model-based mean and standard errors we follow the 'GAMdist' approach in Foster *et al.* (2017).

```
tmp <- modEsti( y=c( Z, Zlegacy), locations=rbind( X[samp$ID,], legacySites),
        includeLegacyLocation=TRUE, legacyIDs=n + 1:nrow( legacySites),
        predPts=X, control=list(B=1000))
print( tmp)

$mean
[1] 2.666453

$se
[1] 0.3124214

$CI
    2.5%    97.5%
2.035070 3.305782
```

In this case, the standard error estimates are quite different. On average, they tend to be (when there are only a few legacy sites). Even so, this level of difference is unusual.

Last Things Last

The only remaining thing to do is to tidy up our workspace. First, to export our sample locations. Second, to remove all objects for this analysis from your workspace.

```
#write csv
write.csv( samp, file="pointSample1.csv", row.names=FALSE)
#tidy
rm( list=ls())
```

Transect-Based Surveys

Let's now pretend that we want to generate $n = 10$ **transect** locations over a grid of points (representing the centres of a tessellation). We will use the methods described in Foster *et al.* (in prep). The transects are linear and are each of length 0.15, and the grid of points consists of $N = 100 \times 100 = 10000$ points in 2-dimensional space (spanning the interval $[0, 1]$ in both dimensions). For this example, we generate all information about the survey area and inclusion probabilities but in a real application, most of this information is likely to come from file (e.g. a shapefile).

A small word of warning: this example will take a little while to run. Sorry. Whilst choices have been made to reduce the computation, at the expense of accuracy, the computation is still quite intensive.

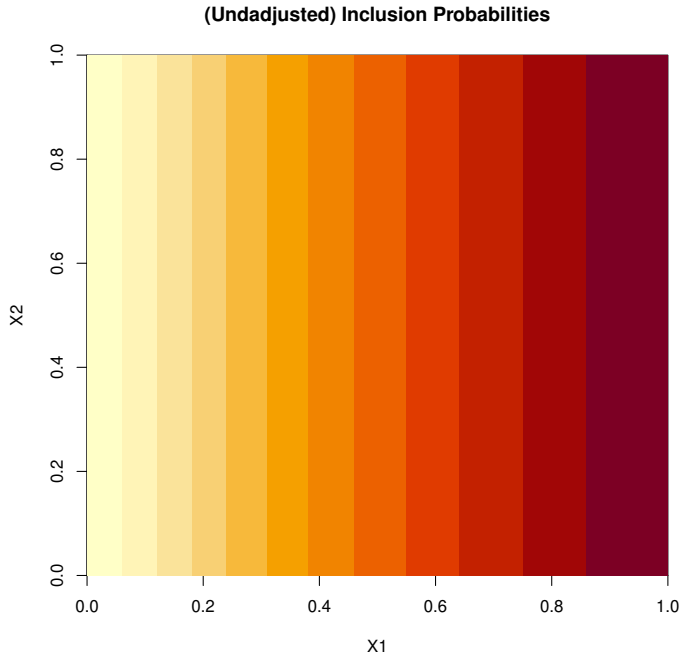
Another note of warning: while the random number seed is set, different versions of R may produce different results. This is due to R evoloving. In particular, R/3.6 could be quite different from R/3.5.

```
set.seed( 747)  #I'm currently on a 787, so it *almost* seems appropriate
#number of transects
n <- 10
#number of points to sample from
N <- 100^2
#the sampling grid (offset so that the edge locations have same area)
offsetX <- 1/(2*sqrt( N))
my.seq <- seq( from=offsetX, to=1-offsetX, length=sqrt(N))
X <- expand.grid( my.seq, my.seq)
colnames( X) <- c("X1","X2")
```

Inclusion Probabilities

Here, like in the previous example, we are going to pretend that there is some gradient in the variance of the population under study. We stress that this is illustrative only. The transect probabilities are set-up to reflect this.

```
#non-uniform inclusion probabilities
inclProbs <- 1-exp(-X[,1])
#scaling to enforce summation to n
inclProbs <- n * inclProbs / sum( inclProbs)
#uniform inclusion probabilities would be inclProbs <- rep( n/N, times=N)
#visualise
image( x=unique( X[,1]), y=unique( X[,2]),
    z=matrix( inclProbs, nrow=sqrt(nrow(X)), ncol=sqrt(nrow( X))),
    main="(Undadjusted) Inclusion Probabilities",
    ylab=colnames( X)[2], xlab=colnames( X)[1])
```



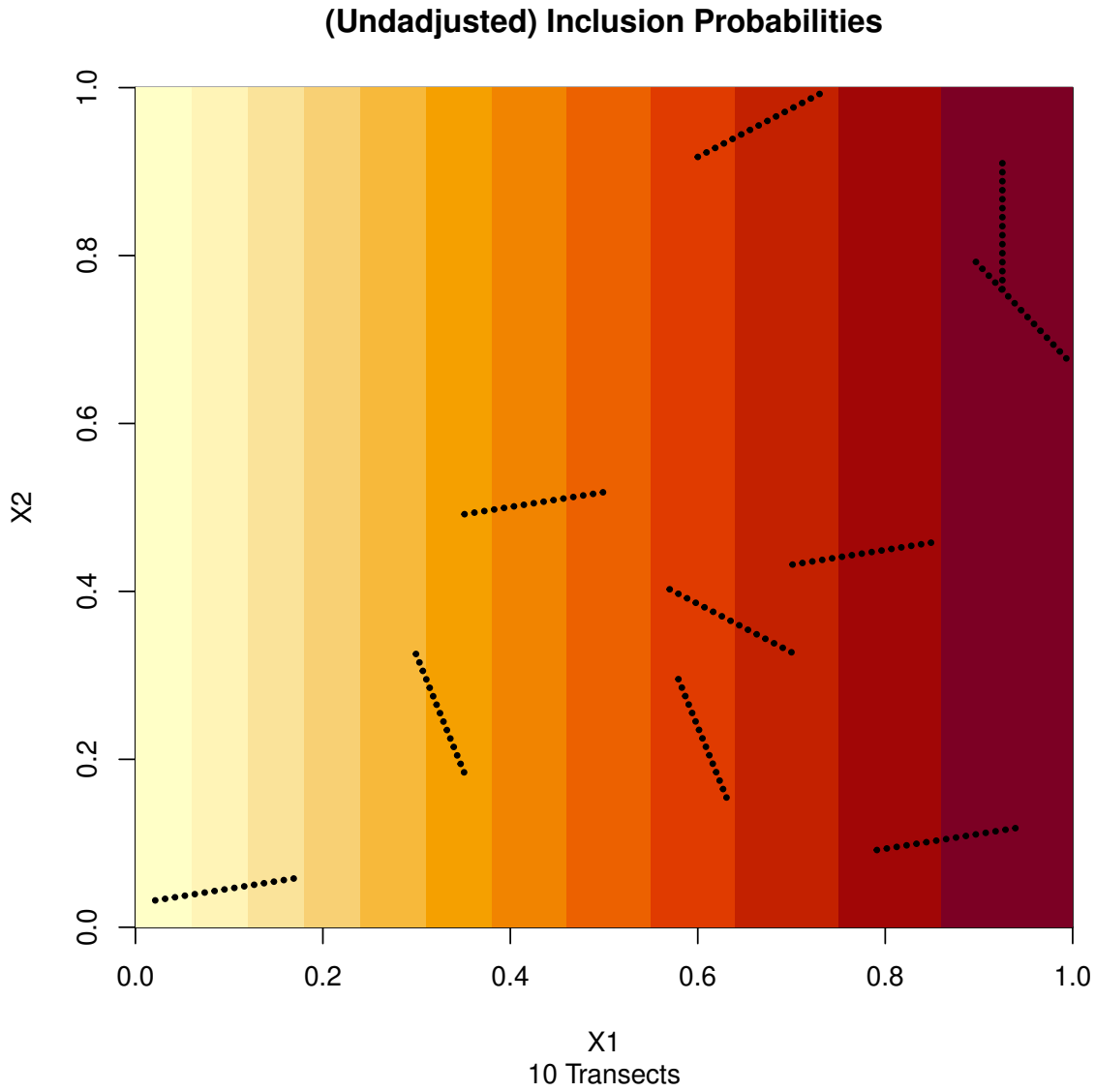
We will need to tell the transect sampling function some information about the size and shape of the transects. For this purpose, the transects should be thought of as a set of points (arranged on a line for a linear transect). For the particular case of linear transects the sampling function `transectSamp` arranges this for us, but we still need to give some information. The choices made here are compromised in favour of speed-of-execution rather than accuracy. In a real (and final) design, you may want to consider finer resolutions. These are the only control options needed in this example, but we note that there are others, which mostly control different aspects of the algorithm (not the transect setup itself).

```
#my.control is a list that contains
my.control <- list(
  #the type of transect
  transect.pattern="line",
  #the length of transect
  line.length=0.15,
  #the number of points that define the transect
  transect.nPts=15,
  #the number of putative directions that a transect can take
  nRotate=9
)
```

Take the Transect Sample

The sample is obtained by a relatively straight-forward call. They are also plotted over the inclusion probabilities.

```
#take the transect sample
samp <- transectSamp( n=n, potential.sites=X, inclusion.probs=inclProbs,
                      control=my.control)
image( x=unique( X[,1]), y=unique( X[,2]),
    z=matrix( inclProbs, nrow=sqrt(nrow(X)), ncol=sqrt(nrow( X))),
    main="(Undadjusted) Inclusion Probabilities",
    sub="10 Transects",
    ylab=colnames( X)[2], xlab=colnames( X)[1])
points( samp$points[,5:6], pch=20, cex=0.6)
```



Last Things Last

The only remaining thing to do is to tidy up our workspace. First, to export our sample

locations. Second, to remove all objects for this analysis from your workspace.

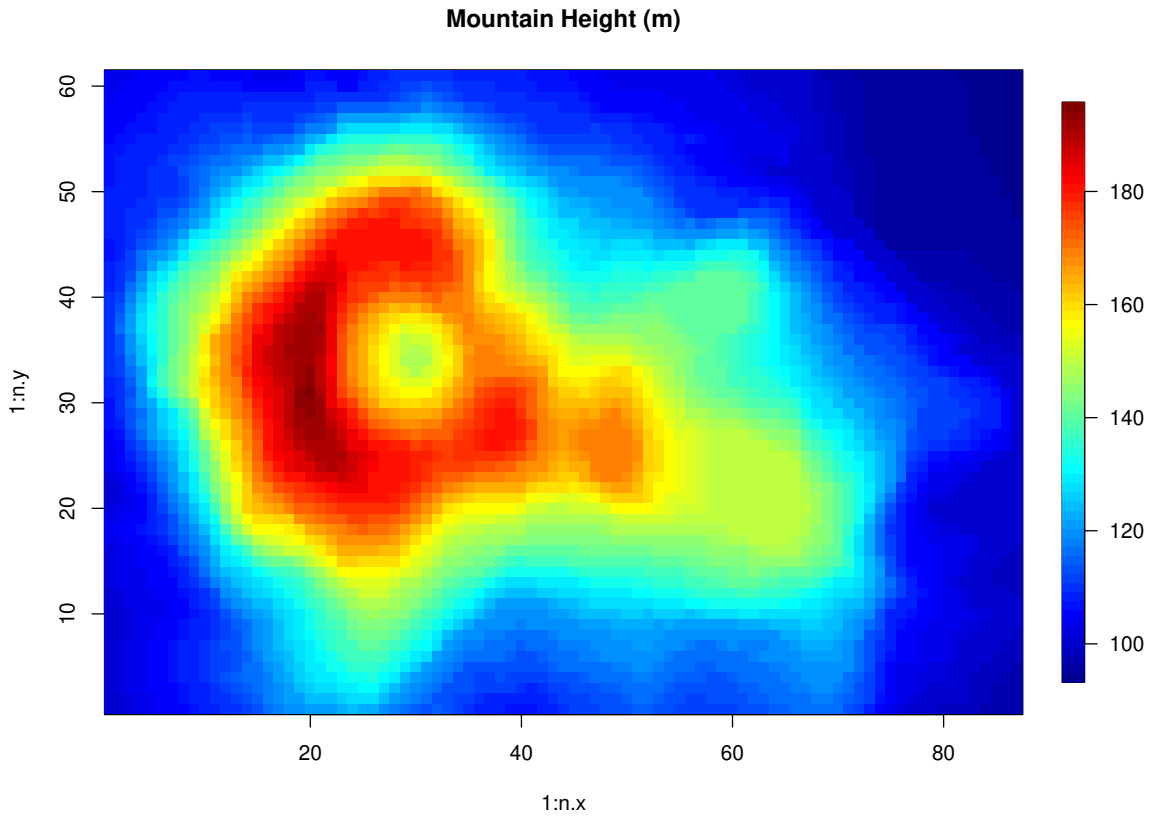
```
#write csv
write.csv( samp$transect, file="transectSample1.csv", row.names=FALSE)
#tidy
rm( list=ls())
```

A Harder/Constrained Transect-Based Survey

The design problem described in the previous section is pretty straight-forward as the sample space doesn't have any constraints upon it. Here, we make things more complicated in this respect, by forcing transects to go down-hill. This mimics towed-platforms, such as underwater image tools, and was enforced in the seamount example in Foster *et al.* (in prep). As an illustration, we utilise the volcano altitude dataset from the excellent `MASS` package (Venables and Ripley 2002). The inclusion probabilities here are assumed to be uniform, so that the transects will be spatially-balanced, and even, throughout the study area.

Forcing down-hill transects isn't the only type of constraint. However, it is illustrative. Another type of constraint that we regularly encounter is, when sampling some certain types of seamounts, the desire to run the transects from 'peak-to-base' so that all transects must start at the highest point of the seamount. This kind of constraint can be incorporated within `MBHdesign` using the function `findTransFromPoint`, which only allows transects from a specified set of points.

```
library( MASS)  #for the data
library( fields)  #for image.plot
library( MBHdesign) #for the spatial design and constraints
set.seed( 717)  #Last plan I was on
#number of transects
n <- 20
#load the altitude data
data( volcano)  #this is a matrix
n.x <- nrow( volcano)
n.y <- ncol( volcano)
image.plot( x=1:n.x, y=1:n.y, z=volcano, main="Mountain Height (m)", asp=1)
#format for MBHdesign functions
pot.sites <- expand.grid( x=1:n.x, y=1:n.y)
pot.sites$height <- as.vector( volcano)
#details of the transects (see Details section in ?transectSamp)
vol.control <- list( transect.pattern="line", transect.nPts=10,
                     line.length=7, nRotate=11, mc.cores=1)
#In a real application, transect.nPts and nRotate may need to be increased
#1 cores have been used to ensure generality for all computers. Use more to speed things u
```



So, that is the data. Note that there are locations where transects could descent and then ascend. The 'hole' (crater) in the middle of the mountain is the largest and most obvious example. To avoid transects that ascend and descend, we use an `MBHdesign` function

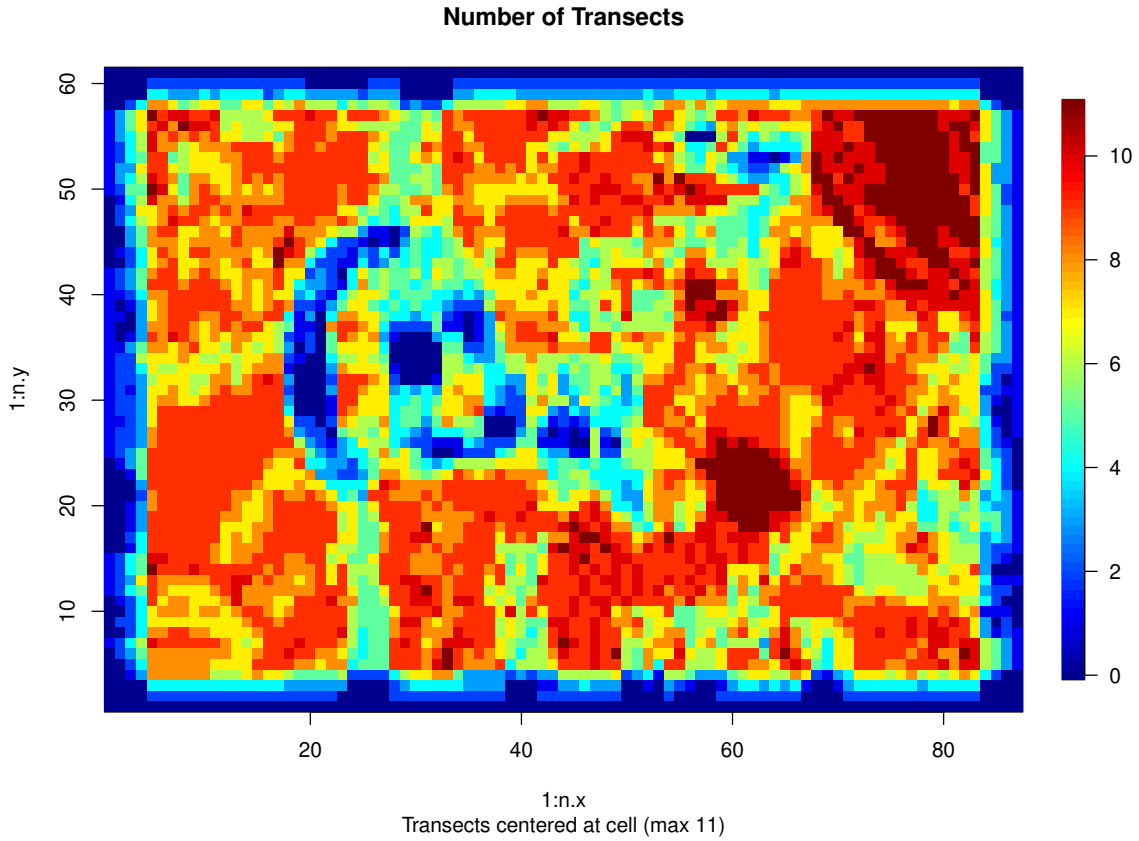
```
vol.constraints <- findDescendingTrans(
  potential.sites = pot.sites[,c("x","y")], bathy=pot.sites$height,
  in.area=rep( TRUE, nrow( pot.sites)), control=vol.control)
#this is a matrix with nrow given by the number of sites and ncol by
#   the number of rotations around each site
print( dim( vol.constraints))

[1] 5307   11

#The contents describe how the transect lays over the landscape
#So, there are 15592 putative transects that ascend and descend
#   (and can't be used in the sample)
table( as.vector( vol.constraints))


         descend descendAndNA_NaN        upAndDown
           34397             8388            15592
```

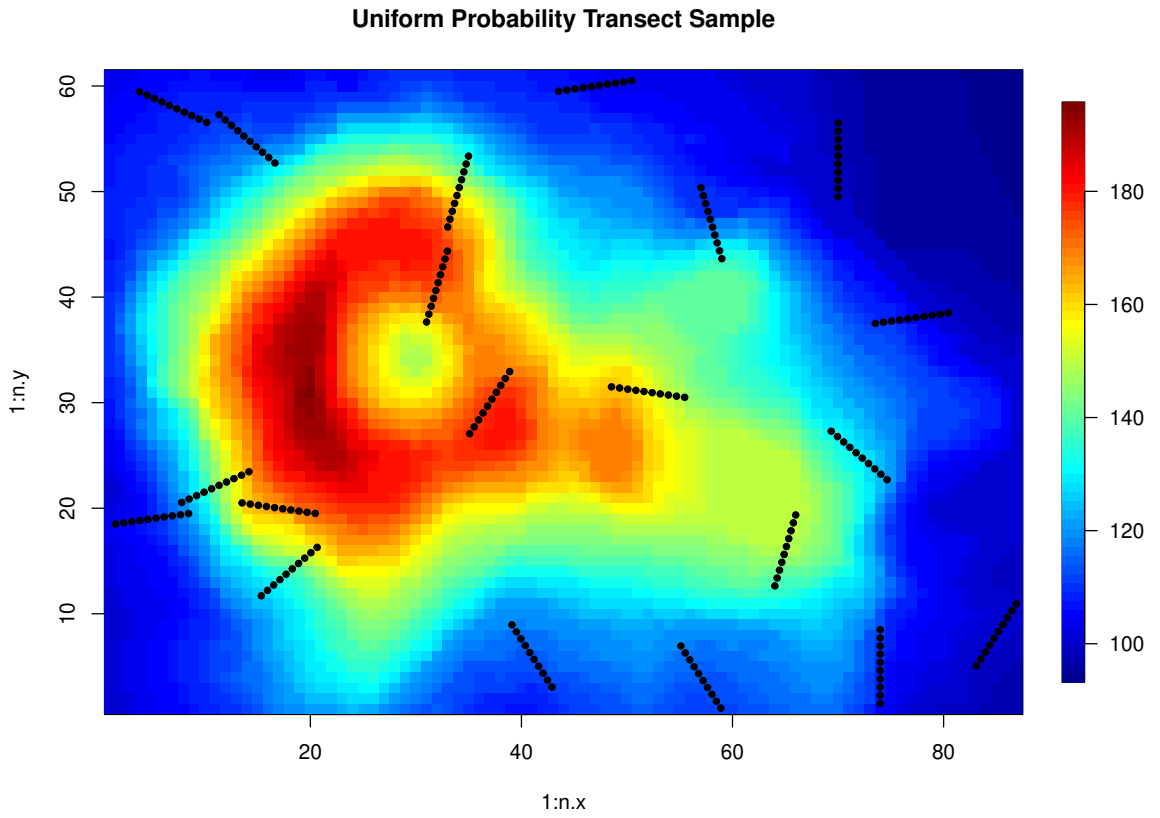
```
#convert to TRUE/FALSE
#Note that the final possible transect type ('descendAndNA') is
#   not present in these data
#If present, we would have to decide to sample these or not
vol.constraints.bool <- matrix( FALSE, nrow=nrow( vol.constraints),
                                ncol=ncol( vol.constraints))
vol.constraints.bool[vol.constraints %in% c("descend")] <- TRUE
#Let's get a visual to see what has just been done.
tmpMat <- matrix( apply( vol.constraints.bool, 1, sum), nrow=n.x, ncol=n.y)
image.plot( x=1:n.x, y=1:n.y, z=tmpMat,
            main="Number of Transects",
            sub="Transects centered at cell (max 11)", asp=1)
#There aren't any transects that are centred on ridges or depressions.
```



The task is to now sample the reduced sampling frame. Fortunately, with the constraint matrix just produced, this is pretty easy.

```
#take the sample
volSamp <- transectSamp( n=n, potential.sites=pot.sites[,c("x","y")],
                         control=vol.control,
                         constrainedSet=vol.constraints.bool)
#visualise the sample
```

```
image.plot( x=1:n.x, y=1:n.y, z=volcano,
            main="Uniform Probability Transect Sample", asp=1)
points( volSamp$points[,c("x","y")], pch=20)
```



The only remaining thing to do is to tidy up our workspace. First, to export our sample locations. Second, to remove all objects for this analysis from your workspace.

```
#write csv
write.csv( volSamp$transect, file="volcanoSample1.csv", row.names=FALSE)
#tidy
rm( list=ls())
```

1. Acknowledgements

This work was undertaken for the Marine Biodiversity Hub, a collaborative partnership supported through funding from the Australian Government's National Environmental Science Program. Our goal is to assist decision-makers to understand, manage and conserve Australia's environment by funding world-class biodiversity science. NESP Marine Biodiversity Hub partners include the Institute for Marine and Antarctic Studies, University of Tasmania; CSIRO, Geoscience Australia, Australian Institute of Marine Science, Museum Victoria, Charles Darwin University, University of Western Australia, NSW Office of Environment

and Heritage, NSW Department of Primary Industries and the Integrated Marine Observing System.

References

Dumelle M, Kincaid TM, Olsen AR, Weber MH (2021). *spsurvey: Spatial Sampling Design and Analysis*. R package version 5.0.1.

Foster S, Hosack G, Lawrence E, Przeslawski R, Hedge P, Caley M, Barrett N, Williams A, Li J, Lynch T, Dambacher J, Sweatman H, Hayes K (2017). "Spatially balanced designs that incorporate legacy sites." *Methods in Ecology and Evolution*, pp. n/a–n/a. ISSN 2041-210X. doi:10.1111/2041-210X.12782. URL http://dx.doi.org/10.1111/2041-210X.12782.

Foster S, Hosack G, Monk J, Lawrence E, Barrett N, Williams A, Przeslawski R (in prep). "Spatially-Balanced Designs for Transect-Based Surveys."

Grafström A, Tillé Y (2013). "Doubly balanced spatial sampling with spreading and restitution of auxiliary totals." *Environmetrics*, **24**(2), 120–131. ISSN 1099-095X. doi:10.1002/env.2194. URL http://dx.doi.org/10.1002/env.2194.

Horvitz D, Thompson D (1952). "A Generalization of Sampling Without Replacement From a Finite Universe." *Journal of the American Statistical Association*, **47**(260), 663–685. ISSN 01621459. URL http://www.jstor.org/stable/2280784.

Robertson BL, Brown JA, McDonald T, Jaksons P (2013). "BAS: Balanced Acceptance Sampling of Natural Resources." *Biometrics*, **69**(3), 776–784. ISSN 1541-0420. doi:10.1111/biom.12059. URL http://dx.doi.org/10.1111/biom.12059.

Stevens D, Olsen A (2003). "Variance estimation for spatially balanced samples of environmental resources." *Environmetrics*, **14**(6), 593–610. ISSN 1099-095X. doi:10.1002/env.606. URL http://dx.doi.org/10.1002/env.606.

Taylor L (1961). "Aggregation, Variance and the Mean." *Nature*, **189**(4766), 732–735. URL http://dx.doi.org/10.1038/189732a0.

Venables W, Ripley B (2002). *Modern Applied Statistics with S. Fourth Edition*. Springer.

2. Appendix

2.1. Computational details

This vignette was created using the following R and add-on package versions

- R version 4.1.1 (2021-08-10), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_AU.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_AU.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_AU.UTF-8`, `LC_MESSAGES=en_AU.UTF-8`, `LC_PAPER=en_AU.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_AU.UTF-8`, `LC_IDENTIFICATION=C`

- Running under: `Ubuntu 20.04.3 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0`
- Base packages: base, datasets, grDevices, graphics, grid, methods, stats, utils
- Other packages: MASS 7.3-54, MBHdesign 2.1.11, Matrix 1.3-4, dotCall64 1.0-1, fields 11.6, knitr 1.36, sf 0.9-8, spam 2.6-0, spsurvey 5.0.1, survey 4.1-1, survival 3.2-11
- Loaded via a namespace (and not attached): AlgDesign 1.2.0, DBI 1.1.1, KernSmooth 2.23-20, R6 2.5.0, Rcpp 1.0.6, abind 1.4-5, assertthat 0.2.1, boot 1.3-28, class 7.3-19, classInt 0.4-3, codetools 0.2-18, compiler 4.1.1, crayon 1.4.1, crossdes 1.1-1, deldir 0.2-10, digest 0.6.27, dplyr 1.0.5, e1071 1.7-6, ellipsis 0.3.2, evaluate 0.14, fansi 0.4.2, generics 0.1.0, geometry 0.4.5, glue 1.4.2, gtools 3.8.2, highr 0.8, lattice 0.20-44, lifecycle 1.0.0, lme4 1.1-27.1, magic 1.5-9, magrittr 2.0.1, maps 3.3.0, mgcv 1.8-36, minqa 1.2.4, mitools 2.4, mvtnorm 1.1-1, nlme 3.1-152, nloptr 1.2.2.2, parallel 4.1.1, pillar 1.6.1, pkgconfig 2.0.3, proxy 0.4-25, purrr 0.3.4, randtoolbox 1.30.1, rlang 0.4.10, rngWELL 0.10-6, splines 4.1.1, stringi 1.5.3, stringr 1.4.0, tibble 3.1.0, tidyselect 1.1.0, tools 4.1.1, units 0.7-1, utf8 1.2.1, vctrs 0.3.8, xfun 0.22

Affiliation:

Scott D. Foster
CSIRO
Marine Laboratories
GPObox 1538
Hobart 7001
Australia E-mail: scott.foster@csiro.au