Supporting Information Appendix S1 - GDM Examples

Contents

1 SECTION 1 - GDM basics

The gdm package is available on CRAN, development versions are available on GitHub. This vignette covers only those functions available from the CRAN version as of 16 November 2021 (1.5.0-1).

```
# installation of the package from CRAN
#install.packages("gdm")

# installation from GitHub
#library(devtools)
#install_github("fitzLab-AL/GDM")

# load the gdm package
library(gdm)
```

1.1 Example data

The biological data provided in the **gdm** package are occurrence data for plants from southwest Australia (Fitzpatrick et al. 2013). Of the original data, a subset of 26 species were selected to be included with the package. The full datasets are available from Dryad. The environmental data include both climatic and soils variables, with the climate data being supplied as both tabular (at sites only) and raster formats (all of southwest Australia).

GDM can use several data formats as input. Most common are site-by-species tables (sites in rows, species across columns) for the response and site-by-environment tables (sites in rows, predictors across columns) as the predictors, though distance matrices and rasters are also accommodated. For example purposes, a biological dissimilarity matrix is provided to showcase the use of a pre-formatted distance matrix (`bioFormat = 3`) as the response variable, though note that distance matrices can also be used as predictors (e.g., to model compositional variation in one group as a function of compositional variation in another group (Jones et al. 2013)).

The first example uses an x-y species list where there is one row *per species record rather than per site* - similar to what would be obtained from online databases such as GBIF. Note that the rows and their order must match in the biological and environmental data frames and must not include NAs. In this example both the species and environmental data are provided in the same table, which are then indexed to create two tables, one for the species data and the other for the environmental data.

```
# reads in example input data
load(system.file("./data/gdm.RData", package="gdm"))

# columns 3-7 are soils variables, remainder are climate
gdmExpData[1:3,]
```

```
##   species site    awcA  phTotal   sandA    shcA solumDepth     bio5     bio6
## 1    spp1 1066 14.4725 546.1800 71.3250 178.865   875.1725 31.43824 5.058823
## 2    spp1 1026 16.2575 470.9950 68.8975 105.840   928.4925 33.14412 4.852941
## 3    spp1 1025 23.1375 459.7425 71.4700  88.355   892.2275 32.84000 4.817143
##      bio15 bio18    bio19       Lat     Long
## 1 40.38235     0 132.6471 -32.99425 118.7573
## 2 48.20588     0 140.2941 -32.04285 118.3495
## 3 53.88571    43 145.0571 -31.99067 117.8260
```

```
# get columns with xy, site ID, and species data
sppTab <- gdmExpData[, c("species", "site", "Lat", "Long")]

# get columns with environment data and xy-coordinates
envTab <- gdmExpData[, c(2:ncol(gdmExpData))]
```

1.2 Preparing site-pair tables

The initial step in fitting a generalized dissimilarity model is to combine the biological and environmental data into "site-pair" format. This can be accomplished using the `formatsitepair` function. Each row in the resulting site-pair table contains a biological distance measure in the first column (the default is Bray-Curtis distance though any measure scaled between 0-1 will work). The second column contains the weight to be assigned to each data point in model fitting (defaults to 1, but can be customized by the user or can be scaled to site richness, see below). The remaining columns are the environmental values at a site (s1) and those at a second site (s2) making up a site pair. Subsequent rows repeat this pattern until all possible site pairs are represented and such that pairwise distances between all sites can be calculated and used as predictors. While the site-pair table format can produce extremely large data frames and contain numerous repeat values, it also allows great flexibility. Most notably, individual site pairs easily can be excluded from model fitting.

A properly formatted site-pair table will have at least six columns (distance, weights, s1.xCoord, s1.yCoord, s2.xCoord, s2.yCoord) and possibly more depending upon how many predictor variables are included. See `?formatsitepair` and `?gdm` for more details.

```
# Example where the biological data is a list of species observed in specific locations
gdmTab <- formatsitepair(bioData=sppTab,
                         bioFormat=2,
                         XColumn="Long",
                         YColumn="Lat",
                         sppColumn="species",
                         siteColumn="site",
                         predData=envTab)
gdmTab[1:3,]
```

```
##        distance weights s1.xCoord s1.yCoord s2.xCoord s2.yCoord s1.awcA
## 132   0.4485981       1   115.057 -29.40472  115.5677 -29.46599 23.0101
## 132.1 0.7575758       1   115.057 -29.40472  116.0789 -29.52556 23.0101
## 132.2 0.8939394       1   115.057 -29.40472  116.5907 -29.58342 23.0101
##       s1.phTotal s1.sandA  s1.shcA s1.solumDepth s1.bio5 s1.bio6 s1.bio15
## 132     480.3266 83.99326 477.5656      1129.933  34.668   8.908    86.64
## 132.1   480.3266 83.99326 477.5656      1129.933  34.668   8.908    86.64
## 132.2   480.3266 83.99326 477.5656      1129.933  34.668   8.908    86.64
##       s1.bio18 s1.bio19 s2.awcA s2.phTotal s2.sandA  s2.shcA s2.solumDepth
## 132          0   267.44 22.3925   494.1225  76.6900 357.7225     1183.9025
## 132.1        0   267.44 17.0975   415.1275  70.0175 112.4800      985.5300
## 132.2        0   267.44 17.0300   333.4400  71.5950 165.7250      956.5425
##        s2.bio5  s2.bio6 s2.bio15 s2.bio18 s2.bio19
## 132   35.50571 7.448572 75.37143        0 228.6572
## 132.1 36.05000 6.605882 64.52941        0 168.8824
## 132.2 36.18750 6.131250 58.75000        0 141.1250
```

```
# Example where the biological data is a distance matrix of pre-computed dissimilarities
dim(gdmDissim)
```

```
## [1] 94 94
```

```
gdmDissim[1:5, 1:5]
```

```
##          V1        V2        V3        V4        V5
## 1 0.0000000 0.8181818 1.0000000 0.5000000 0.7500000
## 2 0.8181818 0.0000000 0.9000000 0.7777778 0.6551724
## 3 1.0000000 0.9000000 0.0000000 1.0000000 0.5757576
## 4 0.5000000 0.7777778 1.0000000 0.0000000 0.9090909
## 5 0.7500000 0.6551724 0.5757576 0.9090909 0.0000000
```

```
site <- unique(sppTab$site)
gdmDissim <- cbind(site, gdmDissim)
gdmTab.dis <- formatsitepair(bioData=gdmDissim,
                             bioFormat=3,
                             XColumn="Long",
                             YColumn="Lat",
                             predData=envTab,
                             siteColumn="site")
```

Environmental data can be extracted directly from rasters, assuming x-y coordinates of sites are provided in either a site-species table (`bioFormat = 1`) or as a x-y species list (`bioFormat = 2`). The `formatsitepair` function assumes that the coordinates of the sites are in the same coordinate system as the raster layers.

```
# load the raster package (install first if necessary)
library(raster)

# environmental raster data
rastFile <- system.file("./extdata/stackedVars.grd", package="gdm")
envRast <- stack(rastFile)
gdmTab.rast <- formatsitepair(bioData=sppTab,
                              bioFormat=2,
                              XColumn="Long",
                              YColumn="Lat",
                              sppColumn="species",
                              siteColumn="site",
                              predData=envRast)
# make sure there are no NA values
# e.g., if some sites do not intersect the rasters
sum(is.na(gdmTab.rast))
```

```
## [1] 465
```

```
gdmTab.rast <- na.omit(gdmTab.rast)
```

1.3 Dealing with biases associated with presence-only data

The ideal biological data for fitting a GDM are occurrence records (presence-absence or abundance) from a network of sites where all species (from one or more taxonomic groups) have been intensively sampled such that compositional dissimilarity can be reliably estimated between sites. However most species data are collected as part of ad hoc surveys and are presence-only. Under these circumstances, there is no systematic surveying and no sites per se, but rather grid cells with some number of occurrence records depending on the number of species observed, with many grid cells having none, a few, or even a single species record. When under-sampled sites are used to calculate compositional dissimilarity, erroneously high values will result, which will bias the model.

The `formatsitepair` function provides a few options for dealing with this potential bias, including (i) weighting sites relative to the number of species observed (`weightType="richness"`), (ii) removing sites with few species (e.g., `speciesFilter=10`) or (iii) both. Decisions regarding which approach to use will depend on the nature of the data and study system. See Ferrier et al. (2007) for further discussion.

```
# weight by site richness
gdmTab.rw <- formatsitepair(bioData=sppTab,
                            bioFormat=2,
                            XColumn="Long",
                            YColumn="Lat",
                            sppColumn="species",
```

```
                              siteColumn="site",
                              predData=envTab,
                              weightType="richness")

# weights based on richness (number of species records)
gdmTab.rw$weights[1:5]
```

```
## [1] NA NA NA NA NA
```

```
# remove sites with < 10 species records
gdmTab.sf <- formatsitepair(bioData=sppTab,
                              bioFormat=2,
                              XColumn="Long",
                              YColumn="Lat",
                              sppColumn="species",
                              siteColumn="site",
                              predData=envTab,
                              sppFilter=10)
```

1.4 GDM model fitting

GDM is a nonlinear extension of permutational matrix regression that uses flexible splines and a GLM to accommodate two types of nonlinearity common in ecological datasets: (1) variation in the rate of compositional turnover (non-stationarity) along environmental gradients, and (2) the curvilinear relationship between biological distance and environmental and geographical distance.

The function `gdm` fits generalized dissimilarity models and is simple to use once the biological and predictor data have been formatted to a site-pair table. In addition to specifying whether or not the model should be fit with geographical distance as a predictor variable, the user can also specify (i) the number of I-spline basis functions (the default is three, with larger values producing more complex splines) and (ii) the locations of "knots" along the splines (defaults 0 (minimum), 50 (median), and 100 (maximum) quantiles when three I-spline basis functions are used). The effects of altering the number of splines and knot locations has not been systematically explored.

```
gdm.1 <- gdm(data=gdmTab,
             geo=TRUE)
```

The `summary` function provides an overview of the model, including deviance explained and the values of the coefficients for the I-spline for each predictor variable. Variables with all coefficients = 0 have no relationship with the biological pattern. A shorter summary can be obtained using `str`.

```
#summary(gdm.1)
str(gdm.1)
```

```
## List of 15
##  $ dataname    : symbol gdmTab
##  $ geo         : logi TRUE
##  $ sample      : int 4371
##  $ gdmdeviance : num 129
##  $ nulldeviance: num 652
##  $ explained   : num 80.2
##  $ intercept   : num 0.277
##  $ predictors  : chr [1:11] "Geographic" "awcA" "phTotal" "sandA" ...
##  $ coefficients: num [1:33] 0.014 0.372 0 0 0 ...
##  $ knots       : num [1:33] 0.452 2.46 6.532 12.975 22.186 ...
##  $ splines     : num [1:11] 3 3 3 3 3 3 3 3 3 ...
```

```
##  $ creationdate: chr "Thu Nov 25 15:24:32 2021"
##  $ observed    : num [1:4371] 0.449 0.758 0.894 0.918 0.979 ...
##  $ predicted   : num [1:4371] 0.472 0.713 0.871 0.853 0.978 ...
##  $ ecological  : num [1:4371] 0.639 1.25 2.048 1.921 3.804 ...
##  - attr(*, "class")= chr [1:2] "gdm" "list"
```

1.5 GDM model plots

The fitted splines represent one of the most informative outputs from `gdm`, which also can be used to transform and map environmental variables such that they best represent biological patterns. The fitted model and I-splines can be viewed using the `plot` function, which produces a multi-panel plot that includes: (i) the fitted relationship between predicted ecological distance and observed compositional dissimilarity; (ii) predicted versus observed biological distance, and (iii) each I-spline with at least one non-zero coefficient (in the provided example bio18 is not plotted because all three coefficients equaled zero).

The maximum height of each spline indicates the magnitude of total biological change along that gradient and thereby corresponds to the relative importance of that predictor in contributing to biological turnover while holding all other variables constant (i.e., is a partial ecological distance). The spline's shape indicates how the rate of biological change varies with position along that gradient. Thus, the splines provide insight into the total magnitude of biological change as a function of each gradient and where along each gradient those changes are most pronounced. In this example, compositional turnover is greatest along gradients of bio19 (winter precipitation) and phTotal (soil phosphorus) and most rapid near the low ends of these gradients.

```
length(gdm.1$predictors) # get ideal of number of panels
```

```
## [1] 11
```

```
plot(gdm.1, plot.layout=c(4,3))
```

To allow easy customization of I-spline plots, the `isplineExtract` function will extract the plotted values for each I-spline.

```
gdm.1.splineDat <- isplineExtract(gdm.1)
str(gdm.1.splineDat)
```

```
## List of 2
##  $ x: num [1:200, 1:11] 0.452 0.483 0.513 0.544 0.574 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:11] "Geographic" "awcA" "phTotal" "sandA" ...
##  $ y: num [1:200, 1:11] 0 0.00045 0.00095 0.0015 0.0021 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:11] "Geographic" "awcA" "phTotal" "sandA" ...
```

```
plot(gdm.1.splineDat$x[,"Geographic"],
     gdm.1.splineDat$y[,"Geographic"],
     lwd=3,
     type="l",
     xlab="Geographic distance",
     ylab="Partial ecological distance")
```

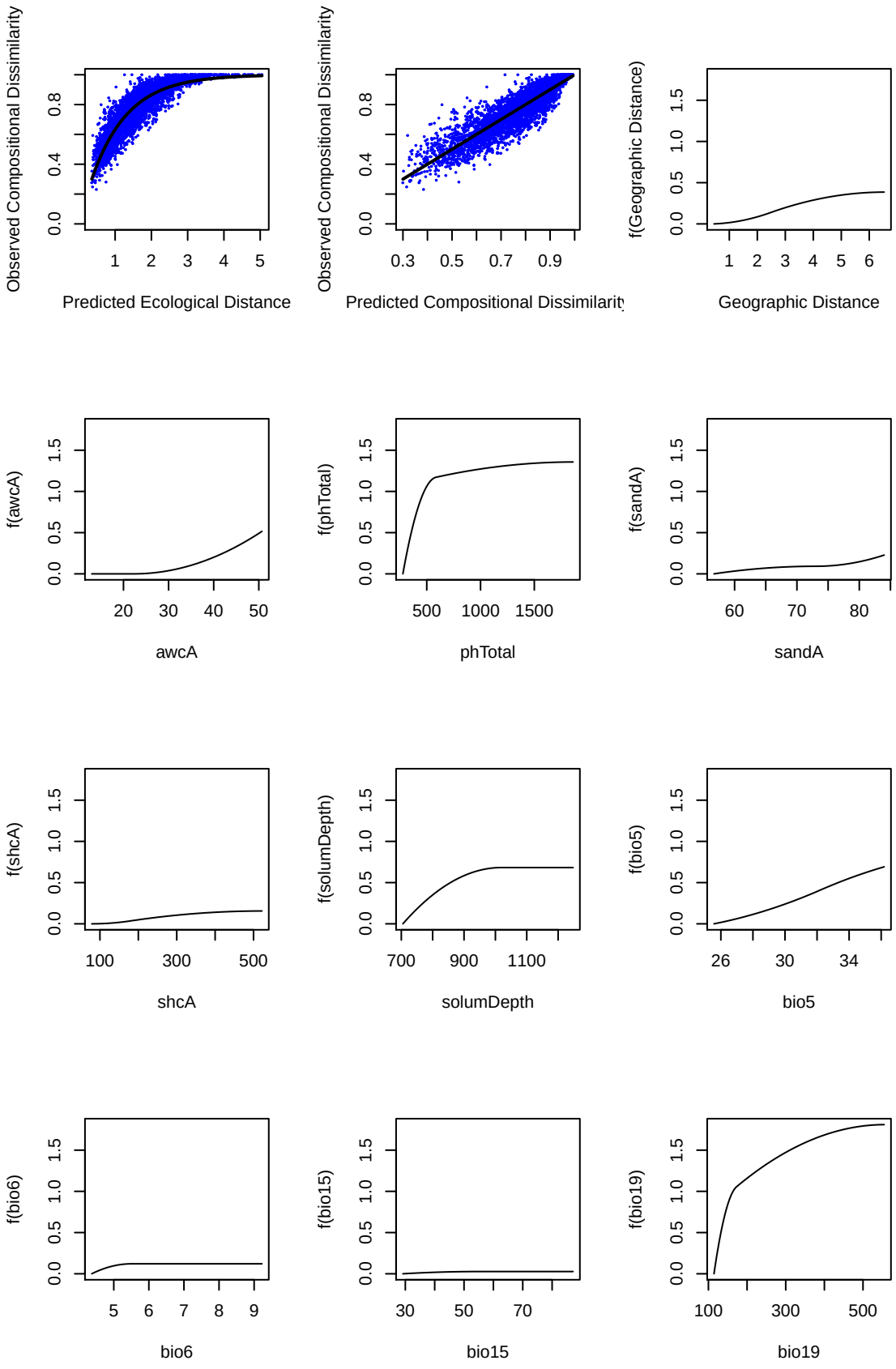


Figure 1: The fitted model (first two panels) and I-splines (remaining panels).

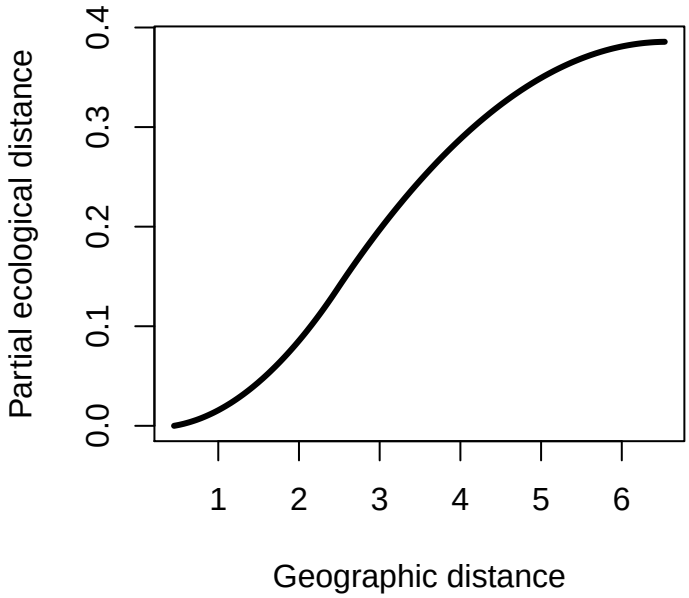


Figure 2: Custom I-spline plot for geographic distance.

1.6 GDM predictions

The I-splines provide an indication of how species composition (or other biological measure) changes along each environmental gradient. Beyond these insights, a fitted model also can be used to (i) predict biological dissimilarity between site pairs in space or between times using the `predict` function and (ii) transform the predictor variables from their arbitrary environmental scales to a common biological importance scale using the `transform` function.

The following examples show predictions between site pairs and through time, and transformation of both tabular and raster data. For the raster example, the transformed layers are used to map spatial patterns of biodiversity.

1.7 Using a fitted GDM to predict biological dissimilarity between sites

The `predict` function requires a site-pair table in the same format as that used to fit the model. For demonstration purposes, we use the same table as that used to fit the model, though predictions to new sites (or times) can be made as well assuming the same set of environmental/spatial predictors are available at those locations (or times).

```
gdm.1.pred <- predict(object=gdm.1,
                      data=gdmTab)

head(gdm.1.pred)
```

```
## [1] 0.4720423 0.7133571 0.8710175 0.8534788 0.9777208 0.3996694
```

```
plot(gdmTab$distance,
     gdm.1.pred,
```

```
     xlab="Observed dissimilarity",
     ylab="Predicted dissimilarity",
     xlim=c(0,1),
     ylim=c(0,1),
     pch=20,
     col=rgb(0,0,1,0.5))
lines(c(-1,2), c(-1,2))
```

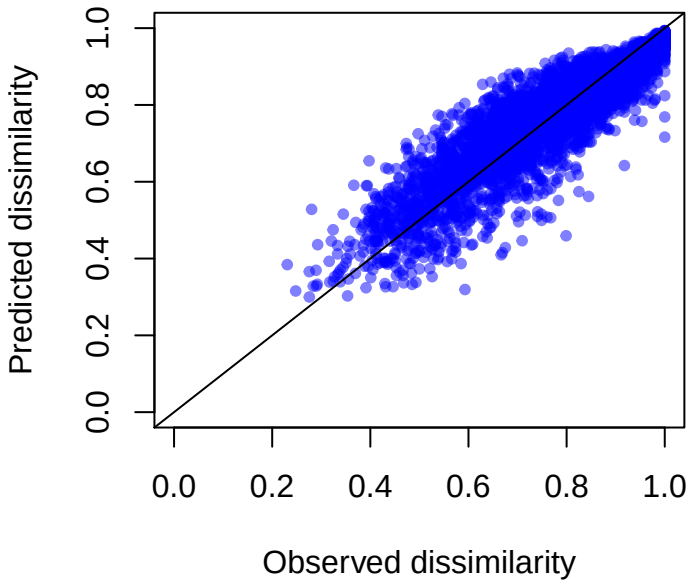


Figure 3: Predicted vs. observed compositional dissimilarity.

2 SECTION 2 - GDM-based spatial analyses

2.1 Transforming spatial predictor layers using a fitted GDM

Spatially explicit predictor data to be transformed can be a raster stack or brick with one layer per predictor. If the model was fit with geographical distance and raster data are provided to the `transform` function, there is no need to provide x- or y-raster layers as these will be generated automatically . However, the character names of the x- and y-coordinates (e.g., "Lat" and "Long") used to fit the model need to be provided.

```
# As in Section 1, we first fit a gdm using raster layers as predictors

# Load data from the gdm package
load(system.file("./data/gdm.RData", package="gdm"))
rastFile <- system.file("./extdata/stackedVars.grd", package="gdm")

# create a raster stack
envRast <- stack(rastFile)

# Create a 'species list' data input using the gdm package data
sppTab <- gdmExpData[, c("species", "site", "Lat", "Long")]

# prepare the gdm input table, using rasters as predictors
sitePairRast <- formatsitepair(bioData=sppTab,
                               bioFormat=2,
                               XColumn="Long",
                               YColumn="Lat",
                               sppColumn="species",
                               siteColumn="site",
                               predData=envRast)

# Remove any site-pairs containing NAs for the extracted raster-based predictors
sitePairRast <- na.omit(sitePairRast)

# fit the GDM
gdmRastMod <- gdm(data=sitePairRast,
                  geo=TRUE)

# Generate transformed predictor rasters, based on the raw raster predictor layers
#  and the fitted gdm
transRasts <- gdm.transform(model=gdmRastMod,
                            data=envRast)
```

2.2 Visualizing multi-dimensional biological patterns

Site-pair based biological distances are difficult to visualize. However, if the `transform` function is applied to rasters, the resulting multi-dimensional biological space can be mapped to reveal biological patterns in geographic space. Alternatively, a biplot can be used to depict where sites fall relative to each other in biological space and therefore how sites differ in predicted biological composition. In either case, the multi-dimensional biological space can be most effectively visualized by taking a PCA to reduce dimensionality and assigning the first three components to an RGB color palette.

```
# Get the data from the gdm transformed rasters as a table
rastDat <- na.omit(getValues(transRasts))

# The PCA can be fit on a sample of grid cells if the rasters are large
```

```
rastDat <- sampleRandom(transRasts, 50000)

# perform the principle components analysis
pcaSamp <- prcomp(rastDat)

# Predict the first three principle components for every cell in the rasters
# note the use of the 'index' argument
pcaRast <- predict(transRasts, pcaSamp, index=1:3)

# scale the PCA rasters to make full use of the colour spectrum
pcaRast[[1]] <- (pcaRast[[1]]-pcaRast[[1]]@data@min) /
  (pcaRast[[1]]@data@max-pcaRast[[1]]@data@min)*255
pcaRast[[2]] <- (pcaRast[[2]]-pcaRast[[2]]@data@min) /
  (pcaRast[[2]]@data@max-pcaRast[[2]]@data@min)*255
pcaRast[[3]] <- (pcaRast[[3]]-pcaRast[[3]]@data@min) /
  (pcaRast[[3]]@data@max-pcaRast[[3]]@data@min)*255

# Plot the three PCA rasters simultaneously, each representing a different colour
#  (red, green, blue)
plotRGB(pcaRast, r=1, g=2, b=3)
```

Figure 4: Predicted spatial variation in plant species composition. Colors represent gradients in species composition derived from transformed environmental predictors. Locations with similar colors are expected to contain similar plant communities.

2.3 Using GDM transformed grids to predict the similarity between two locations

This example demonstrates how the GDM predicted dissimilarity between two locations can be obtained from the spatial model-transformed predictor grids.

```
# Choose two locations of interest
focal.pt.1 <- c(122.0, -31.0) # semiarid woodland
focal.pt.2 <- c(116.0, -34.0) # coastal temperate forest
focal.pts <- rbind(focal.pt.1, focal.pt.2)
```

```
# Extract the transformed environmental values for these two focal locations
focal.trans <- extract(transRasts, focal.pts)

# Calculate the predicted similarity between them
ecol.dist <- sum(abs(focal.trans[1,] - focal.trans[2,]))
similarity.1.2 <- (exp(-1 * (gdmRastMod$intercept + ecol.dist)))

# The predicted similarity between the two locations is:
similarity.1.2
```

```
## [1] 0.01778892
```

2.4 Predicting the similarity of all locations to a specified location

This example shows how the similarity of all locations to a specified focal location of interest can be predicted and mapped.

```
# Choose some locations of interest
focal.pt.1 <- c(122.0, -31.0) # semiarid woodland
focal.pt.2 <- c(116.0, -34.0) # coastal temperate forest
focal.pts <- rbind(focal.pt.1, focal.pt.2)

# Extract the transformed environmental values for these focal locations
focal.trans <- extract(transRasts, focal.pts)

# put the values from the transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]

# now calculate the similarity of all other grid cells to each of these focal locations
similarity.focal.pt.1 <- rep(0, length=nrow(Trans.env.table))
similarity.focal.pt.2 <- rep(0, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  ecol.dist.1 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] - focal.trans[1,]))
  similarity.focal.pt.1[i.cell] <- exp(-1 * (gdmRastMod$intercept + ecol.dist.1))
  ecol.dist.2 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] - focal.trans[2,]))
  similarity.focal.pt.2[i.cell] <- exp(-1 * (gdmRastMod$intercept + ecol.dist.2))
} # end for i.cell

# Format the similarities into a raster an plot them
# First location
focal.pt.1.ras <- raster(transRasts,layer=1)
focal.pt.1.ras <- rasterize(Trans.env.table[,c(1:2)], focal.pt.1.ras,
                            field=similarity.focal.pt.1)
plot(focal.pt.1.ras,
     col = colorRampPalette(c("azure2", "darkgreen"))(100),
     zlim=c(0,1),
     legend.args = list(text = 'Similarity'))
```

```
points(x=focal.pts[1,1],y=focal.pts[1,2])
```

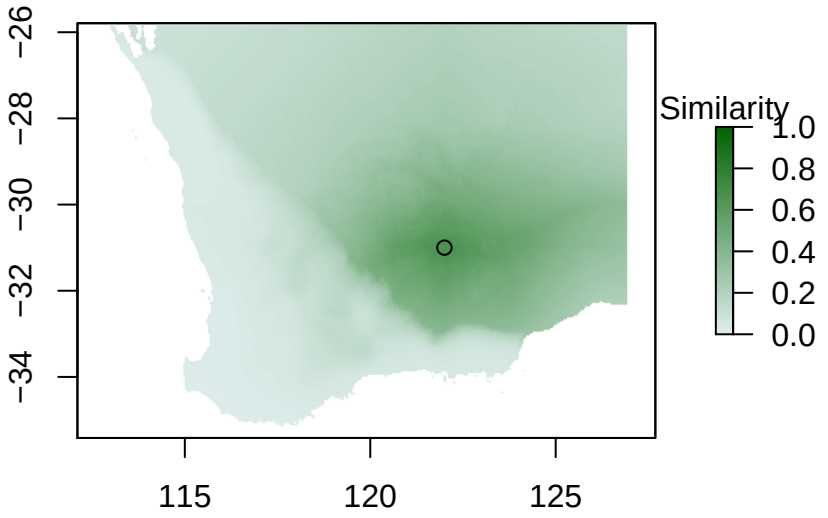


Figure 5: Predicted similarity to the focal location: darker green areas are more similar.

Then for the second location:

```
# Then for the second location
focal.pt.2.ras <- raster(transRasts,layer=1)
focal.pt.2.ras <- rasterize(Trans.env.table[,c(1:2)],
                            focal.pt.2.ras,
                            field=similarity.focal.pt.2)
plot(focal.pt.2.ras,
     col = colorRampPalette(c("azure2", "darkgreen"))(100),
     zlim=c(0,1),
     legend.args = list(text = 'Similarity'))
points(x=focal.pts[2,1], y=focal.pts[2,2])
```

2.5 Predicting the mean similarity within the neighbourhood around each location

This example demonstrates how the average similarity in the neighbourhood around each location can be predicted and mapped. Note that different outcomes will be achieved when considering neighbourhoods of different size (here we use 15km radius).

```
# specify the radius
rad <- 0.10 # this distance is in geographic units (degrees) and equates to about 15km

# put the values from the transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
```

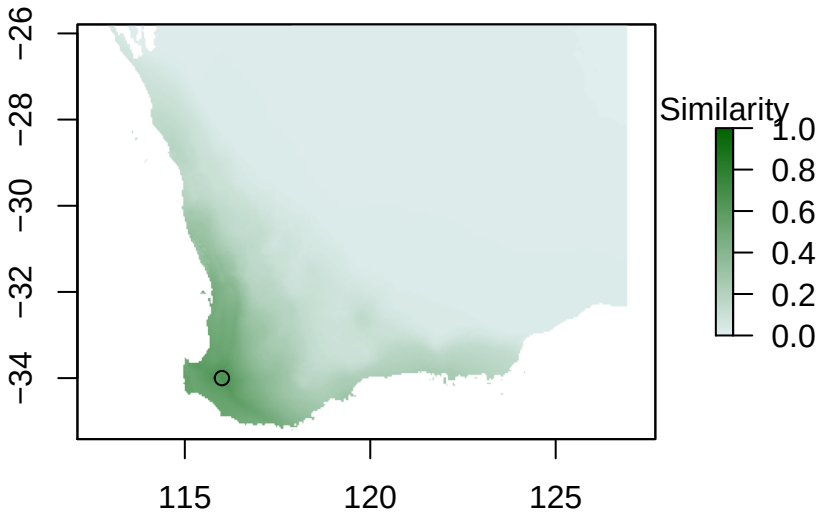


Figure 6: Predicted similarity to the focal location: darker green areas are more similar.

```
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]

# Calculate the similarity of all other grid cells to each of these focal locations
# NB - This loop takes a couple of minutes
mean.similarity.radius <- rep(0, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  # Check if this cell has data
  if(!is.na(Trans.env.table[i.cell,ncol(Trans.env.table)]))
    {
    # calculate the distance of all the cells to the focal cell
    cells.dist <- sqrt(((Trans.env.table[i.cell,1] - Trans.env.table[,1])^2) +
                         ((Trans.env.table[i.cell,2] - Trans.env.table[,2])^2))
    # Grab the cells within the specified radius of the focal cell
    rad.cells <- which(cells.dist <= rad)
    # loop through the neighbouring cells, calculate similarity, add it to the tally
    for(j.cell in 1:length(rad.cells))
    {
      ecol.dist.1 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                               Trans.env.table[rad.cells[j.cell],c(3:ncol(Trans.env.table))]))
      mean.similarity.radius[i.cell] <- mean.similarity.radius[i.cell] +
```

```
                                        (exp(-1 * (gdmRastMod$intercept + ecol.dist.1)))
    } # end for j.cells
    # Finish by dividing by the number of neighbouring cells to get the mean
    mean.similarity.radius[i.cell] <- mean.similarity.radius[i.cell] / length(rad.cells)
  } # end if(!is.na())
} # end for i.cell

# Format the similarities into a raster an plot them
mnsim.ras <- raster(transRasts,layer=1)
mnsim.ras <- rasterize(Trans.env.table[,c(1:2)], mnsim.ras, field=mean.similarity.radius)
plot(mnsim.ras,
     col = colorRampPalette(c("red", "blue"))(100),
     zlim=c(0.5,0.7),
     legend.args = list(text = 'Similarity'))
```

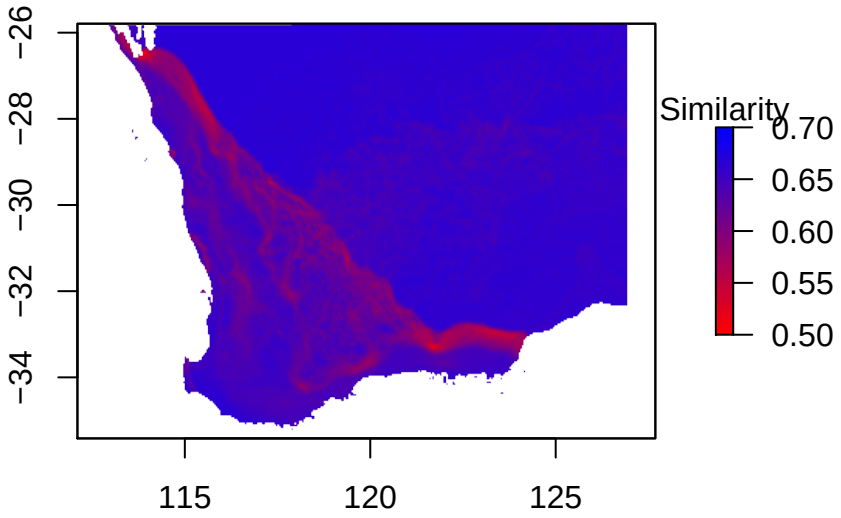


Figure 7: The predicted similarity within the neighbourhood around each location. Areas on the map that have lower similarity to the neighbouring locations are in places where there is predicted to be greater biological turnover.

2.6 Predicting the uniqueness of each location (similarity to the region)

The biological uniqueness of each location can be predicted by calculating the mean similarity between that location and a random sample of all the locations across the region.

```
# put the values from the transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
```

```
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]

# specify the number of randomly selected reference cells (0.5% of the region)
n.ref <- floor(0.005 * length(!is.na(Trans.env.table[,ncol(Trans.env.table)])))

# randomly select this number of reference cells
ref.cells <- sample.int(length(!is.na(Trans.env.table[,ncol(Trans.env.table)])),
                        size = n.ref, replace = FALSE)

# Calculate the similarity of each grid cell to the randomly selected reference cells
# NB - This loop takes a couple of minutes
mean.similarity.region <- rep(0, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  # Check if this cell has data
  if(!is.na(Trans.env.table[i.cell,ncol(Trans.env.table)]))
    {
     # loop through the reference cells, calculate similarity, add it to the tally
    for(j.cell in 1:length(ref.cells))
    {
      ecol.dist.1 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                             Trans.env.table[ref.cells[j.cell],c(3:ncol(Trans.env.table))]))
      mean.similarity.region[i.cell] <- mean.similarity.region[i.cell] +
                                          (exp(-1 * (gdmRastMod$intercept + ecol.dist.1)))
    } # end for j.cells
    # Finish by dividing by the number of neighbouring cells to get the mean
    mean.similarity.region[i.cell] <- mean.similarity.region[i.cell] / length(ref.cells)
  } # end if(!is.na())
} # end for i.cell

# Format the similarities into a raster an plot them
mnsim.ras <- raster(transRasts,layer=1)
mnsim.ras <- rasterize(Trans.env.table[,c(1:2)],
                       mnsim.ras,
                       field=mean.similarity.region)
plot(mnsim.ras,
     col = colorRampPalette(c("skyblue", "black"))(100),
     zlim=c(0,0.5),
     legend.args = list(text = 'Similarity'))
```

2.7 Survey gap anlysis - predicted similarity to survey locations

This example demonstrates a survey gap analysis, where we predict the average similarity of each location to the set of locations that have been surveyed. Here survey gap assessment is demonstrated using the locations of the data used to fit the model.

```
# put the values from the transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
```

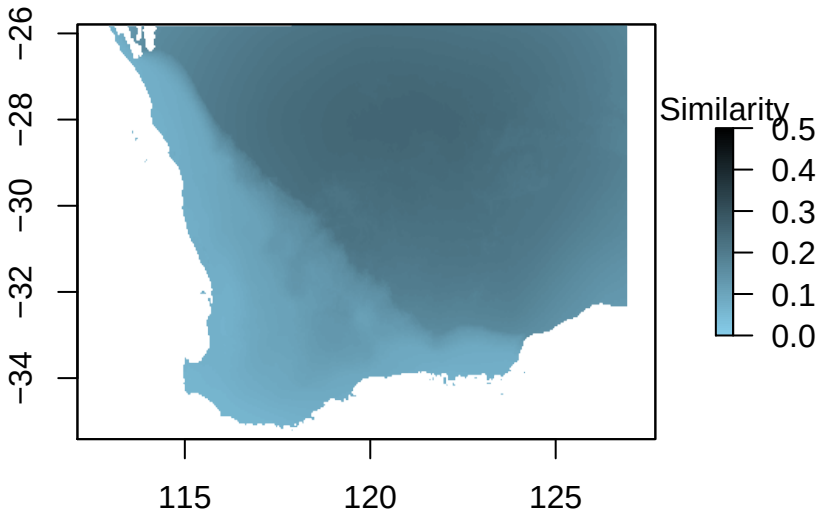


Figure 8: The predicted uniqueness of each location. Areas with lower average similarity to the region are more unique.

```
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]

# specify the locations used to fit the model
ref.coords <- unique(cbind(sppTab$Long, sppTab$Lat))
# extract the gdm transformed predictor values for those locations
ref.Trans.env.table <- extract(transRasts,ref.coords)
ref.Trans.env.table <- ref.Trans.env.table[complete.cases(ref.Trans.env.table),]

# Calculate the similarity of each grid cell to the survey cells
# NB - This loop takes a couple of minutes
mean.similarity.region <- rep(0, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  # Check if this cell has data
  if(!is.na(Trans.env.table[i.cell,ncol(Trans.env.table)]))
  {
    # loop through the reference cells, calculate similarity, add it to the tally
    for(j.cell in 1:nrow(ref.Trans.env.table))
    {
      ecol.dist.1 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                             ref.Trans.env.table[j.cell,]))
      mean.similarity.region[i.cell] <- mean.similarity.region[i.cell] +
                                        (exp(-1 * (gdmRastMod$intercept + ecol.dist.1)))
```

```
    } # end for j.cells
    # Finish by dividing by the number of neighbouring cells to get the mean
    mean.similarity.region[i.cell] <- mean.similarity.region[i.cell] / nrow(ref.Trans.env.table)
  } # end if(!is.na())
} # end for i.cell

# Format the similarities into a raster an plot them
mnsim.ras <- raster(transRasts,layer=1)
mnsim.ras <- rasterize(Trans.env.table[,c(1:2)],
                       mnsim.ras,
                       field=mean.similarity.region)
plot(mnsim.ras,
     col = heat.colors(100),
     zlim=c(0,0.3),
     legend.args = list(text = 'Similarity'))
points(x=ref.coords[,1], y=ref.coords[,2])
```

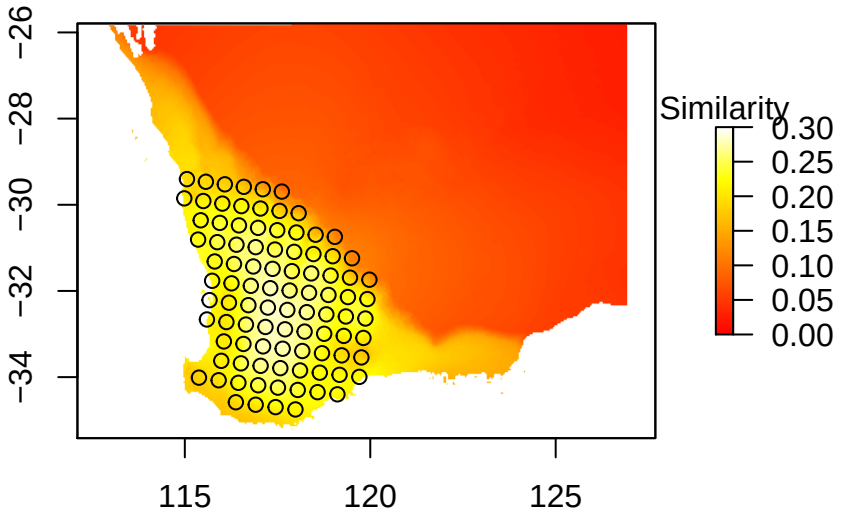


Figure 9: Areas with lower similarity are not as well represented by the surveyed locations.

2.8 Protected area representativeness - predicted similarity to protected areas

This example demonstrates how GDM predictions can be used in assessing how well the protected areas in a region represent the biological diversity. It calculates the similarity of each location to a sample of locations that are protected, accounting for the natural uniqueness of each location.

```
library(sp)

# put the values from the transformed layers in a table for easy analysis
```

```
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]

# specify some simple polygons approximating some of the larger protected areas in the region
pa.1 <- rbind(c(116.542, -34.876), c(117.410, -34.876), c(117.410, -34.569),
              c(116.542, -34.569), c(116.542, -34.876))
pa.2 <- rbind(c(117.700, -34.468), c(118.381, -34.468), c(118.381, -34.321),
              c(117.700, -34.321), c(117.700, -34.468))
pa.3 <- rbind(c(119.950, -33.841), c(119.950, -33.970), c(119.232, -33.970),
              c(119.232, -33.841), c(119.950, -33.841))
pa.4 <- rbind(c(118.862, -33.701), c(118.862, -33.445), c(119.200, -33.445),
              c(119.200, -33.701), c(118.862, -33.701))
pa.5 <- rbind(c(121.911, -32.288), c(121.911, -32.740), c(123.485, -32.740),
              c(123.485, -32.288), c(121.911, -32.288))
pa.6 <- rbind(c(123.244, -33.931), c(123.244, -33.287), c(124.063, -33.287),
              c(124.063, -33.931), c(123.244, -33.931))
pa.7 <- rbind(c(119.637, -31.485), c(119.637, -32.038), c(120.042, -32.038),
              c(120.042, -31.485), c(119.637, -31.485))
pa.8 <- rbind(c(119.405, -29.536), c(119.405, -30.488), c(119.909, -30.488),
              c(119.909, -29.536), c(119.405, -29.536))
pa.9 <- rbind(c(117.834, -29.791), c(117.834, -30.228), c(118.496, -30.228),
              c(118.496, -29.791), c(117.834, -29.791))
pa.10 <- rbind(c(116.143, -31.558), c(116.143, -31.661), c(116.280, -31.661),
               c(116.280, -31.558), c(116.143, -31.558))
pa.11 <- rbind(c(115.661, -31.366), c(115.661, -31.483), c(115.832, -31.483),
               c(115.832, -31.366), c(115.661, -31.366))
pa.12 <- rbind(c(115.603, -31.039), c(115.603, -31.173), c(115.729, -31.173),
               c(115.729, -31.039), c(115.603, -31.039))
pa.13 <- rbind(c(114.975, -29.556), c(114.975, -30.016), c(115.080, -30.016),
               c(115.080, -29.556), c(114.975, -29.556))
pa.14 <- rbind(c(114.711, -26.652), c(114.711, -27.289), c(115.535, -27.289),
               c(115.535, -26.652), c(114.711, -26.652))
pa.15 <- rbind(c(123.136, -30.236), c(123.136, -30.564), c(123.890, -30.564),
               c(123.890, -30.236), c(123.136, -30.236))
pa.16 <- rbind(c(124.671, -29.338), c(124.671, -29.713), c(125.413, -29.713),
               c(125.413, -29.338), c(124.671, -29.338))
pa.17 <- rbind(c(124.014, -27.789), c(124.014, -28.225), c(124.659, -28.225),
               c(124.659, -27.789), c(124.014, -27.789))
pa.18 <- rbind(c(115.425, -34.073), c(115.425, -34.259), c(115.767, -34.259),
               c(115.767, -34.073), c(115.425, -34.073))

# Create a multipolygon object using the sp library
pa.sply = SpatialPolygons(list(Polygons(list(Polygon(pa.1)), ID="a"),
                               Polygons(list(Polygon(pa.2)), ID="b"),
                               Polygons(list(Polygon(pa.3)), ID="c"),
                               Polygons(list(Polygon(pa.4)), ID="d"),
                               Polygons(list(Polygon(pa.5)), ID="e"),
                               Polygons(list(Polygon(pa.6)), ID="f"),
```

```
                              Polygons(list(Polygon(pa.7)), ID="g"),
                              Polygons(list(Polygon(pa.8)), ID="h"),
                              Polygons(list(Polygon(pa.9)), ID="i"),
                              Polygons(list(Polygon(pa.10)), ID="j"),
                              Polygons(list(Polygon(pa.11)), ID="k"),
                              Polygons(list(Polygon(pa.12)), ID="l"),
                              Polygons(list(Polygon(pa.13)), ID="m"),
                              Polygons(list(Polygon(pa.14)), ID="n"),
                              Polygons(list(Polygon(pa.15)), ID="o"),
                              Polygons(list(Polygon(pa.16)), ID="p"),
                              Polygons(list(Polygon(pa.17)), ID="q"),
                              Polygons(list(Polygon(pa.18)), ID="r")),
                          proj4string=crs(transRasts))

# extract values for the cells in the protected area polygons
ref.Trans.env.table <- extract(transRasts, pa.sply)
ref.Trans.env.table <- do.call(rbind, ref.Trans.env.table)
ref.Trans.env.table <- ref.Trans.env.table[complete.cases(ref.Trans.env.table),]
# randomly subsample these locations within the protected areas, to make the analysis faster
ref.Trans.env.table <- ref.Trans.env.table[sample(nrow(ref.Trans.env.table), 300),]

# Select a random sample of cells across the region, to use in standardising
# mean similarity to protected areas
# Specify the number of randomly selcted reference cells
ref.cells <- sample.int(nrow(Trans.env.table), size = nrow(ref.Trans.env.table),
                        replace = FALSE)

# Calculate the similarity of each grid cell to the cells in the protected areas as well as
# the mean similarity of each cell to the whole region, to achieve a suitable estimate of
# representativenes accounting for uniqueness.
# NB - This loop takes a couple of minutes
mean.similarity.pa <- rep(0, length=nrow(Trans.env.table))
mean.similarity.region <- rep(0, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  # Check if this cell has data
  if(!is.na(Trans.env.table[i.cell,ncol(Trans.env.table)]))
  {
    # Protected areas
    # loop through the reference cells, calculate similarity, add it to the tally
    for(j.cell in 1:nrow(ref.Trans.env.table))
      {
      ecol.dist.1 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                             ref.Trans.env.table[j.cell,]))
      mean.similarity.pa[i.cell] <- mean.similarity.pa[i.cell] +
                                    (exp(-1 * (gdmRastMod$intercept + ecol.dist.1)))
      } # end for j.cells
    # Whole Region
    # loop through the reference cells, calculate similarity, add it to the tally
    for(j.cell in 1:length(ref.cells))
      {
      ecol.dist.2 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                             Trans.env.table[ref.cells[j.cell],c(3:ncol(Trans.env.table))]))
```

```
      mean.similarity.region[i.cell] <- mean.similarity.region[i.cell] +
                                        (exp(-1 * (gdmRastMod$intercept + ecol.dist.2)))
      } # end for j.cells
    # Finish by dividing by the number of sample cells to get the mean
    mean.similarity.pa[i.cell] <- mean.similarity.pa[i.cell] / nrow(ref.Trans.env.table)
    mean.similarity.region[i.cell] <- mean.similarity.region[i.cell] / length(ref.cells)
  } # end if(!is.na())
} # end for i.cell

# Format the similarities into a raster an plot them
# the similarity to the protected areas
pasim.ras <- raster(transRasts,layer=1)
pasim.ras <- rasterize(Trans.env.table[,c(1:2)], pasim.ras, field=mean.similarity.pa)
# the similarity to the region
mnsim.ras <- raster(transRasts,layer=1)
mnsim.ras <- rasterize(Trans.env.table[,c(1:2)], mnsim.ras, field=mean.similarity.region)
# Calculate relative similarity
pa.representativeness.ras <- pasim.ras / mnsim.ras
# plot it
plot(pa.representativeness.ras,
     col = colorRampPalette(c("red", "yellow","blue"))(100),
     zlim=c(0,2),
     legend.args = list(text = 'Representativenss'))
plot(pa.sply, add=TRUE, border='black', lwd=2)
```

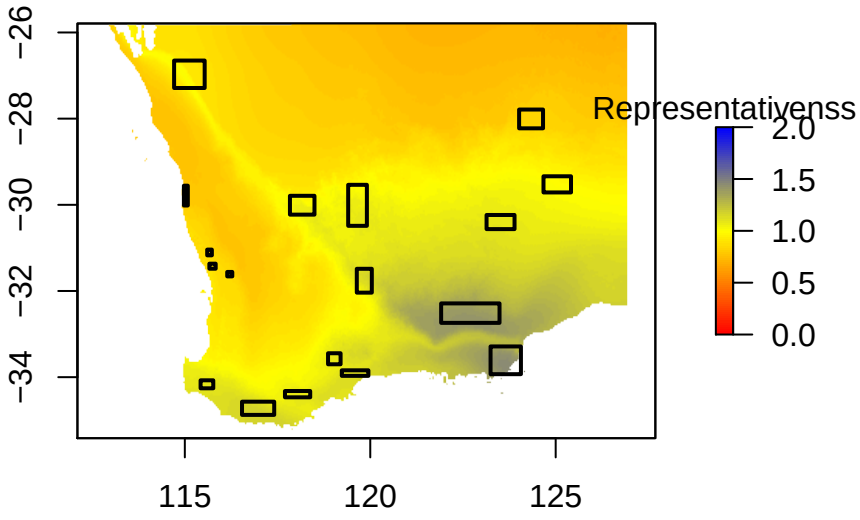


Figure 10: Protected area representativeness. Areas with higher values are better represented by the protected areas (shown as polygons) than areas with lower values.

2.9 Classifying and mapping community types

This example shows how the GDM predicted spatial layers can be used to generate ecological classifications.

```
# Put the values from the transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]

# specify the number of random samples of grid cells to use in the clustering proceedure
n.sub <- 500
# specify the number of community types to derive
n.cat <- 100

# Then take a random sample of grid cells from the transformed environment data
sub.Trans.env <- Trans.env.table[sample(nrow(Trans.env.table), n.sub),]

# Then loop through and determine the predicted dissimilarity between each pair of
# cells in the random set
sub.dissimilarity <- matrix(0, n.sub, n.sub)
colnames(sub.dissimilarity)<-c(1:n.sub)
rownames(sub.dissimilarity)<-c(1:n.sub)
for(i.col in 1:(n.sub-1))
{
  for(i.row in (i.col+1):n.sub)
  {
    ecol.dist <- sum(abs(sub.Trans.env[i.col,c(3:ncol(sub.Trans.env))] -
                         sub.Trans.env[i.row,c(3:ncol(sub.Trans.env))]))
    sub.dissimilarity[i.row,i.col] <- 1 - exp(-1 * (gdmRastMod$intercept + ecol.dist))
    sub.dissimilarity[i.col,i.row] <- sub.dissimilarity[i.row,i.col]
  } # end for i.row
} # end for i.col

# Now apply heirachical clustering to the subsample dissimilarity matrix
sub.dissimilarity<-as.dist(sub.dissimilarity)
class.results<-hclust(sub.dissimilarity, method = "ward.D")
class.membership <- cutree(class.results, k = n.cat)

# Now run through all grid cells, and allocate them to the class of the
# most similar cell in the training set
# takes 5 mins with 500 samples
cell.class <- rep(1, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  max.similarity <- 0
  i.cell.class <- 1
  for(i.sub in 1:n.sub)
  {
    ecol.dist <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                         sub.Trans.env[i.sub,c(3:ncol(sub.Trans.env))]))
    similarity <- exp(-1 * (gdmRastMod$intercept + ecol.dist))
```

```
    if(similarity > max.similarity)
    {
      max.similarity <- similarity
      i.cell.class <- class.membership[i.sub]
    } # end if
  } # end for i.sub
  cell.class[i.cell] <- i.cell.class
} # end for i.cell


# Convert the results to a raster
gdm.class.ras <- raster(transRasts,layer=1)
gdm.class.ras <- rasterize(Trans.env.table[,c(1:2)],
                           gdm.class.ras,
                           field=cell.class)

# Plot the community classes ~~~~~~~~~~~~~
plot(gdm.class.ras)
```

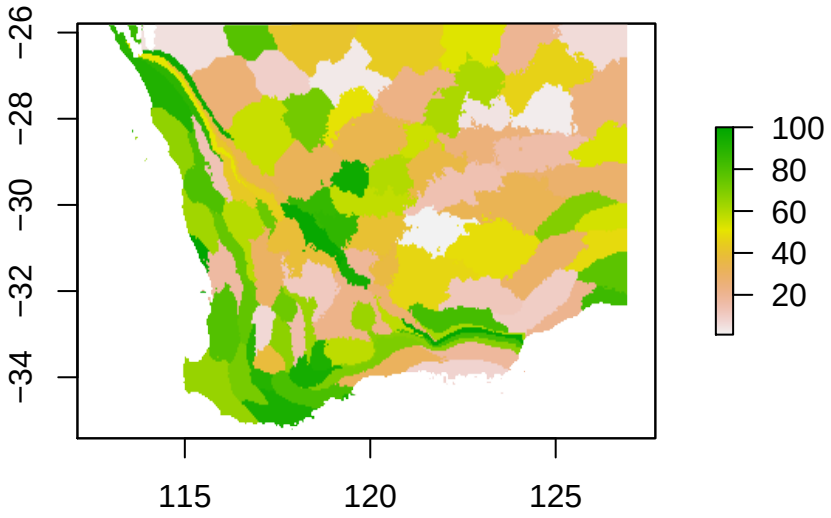


Figure 11: Predicted ecological types - noting that the colour of each class has no meaning in this case.

Next we will consider the similarity of classes to each other in allocating colours on the map.

```
# Generate a matrix of dissimilarities between classes
class.Trans<-as.data.frame(cbind(class.membership,sub.Trans.env))
class.mean <- aggregate(class.Trans, list(Class=class.membership),mean)
class.mean <- class.mean[,-c(1:4)]
class.dissimilarity <- matrix(0, n.cat, n.cat)
colnames(class.dissimilarity)<-c(1:n.cat)
```

```
rownames(class.dissimilarity)<-c(1:n.cat)
for(i.col in 1:(n.cat-1))
{
  for(i.row in (i.col+1):n.cat)
  {
    ecol.dist <- sum(abs(class.mean[i.col,]-class.mean[i.row,]))
    class.dissimilarity[i.row,i.col] <- 1 - exp(-1 * (gdmRastMod$intercept + ecol.dist))
    class.dissimilarity[i.col,i.row] <- class.dissimilarity[i.row,i.col]
  } # end for i.row
} # end for i.col

# ordinate the class-class dissimilarity matrix using multi-dimensional scaling
Class.MDS <- cmdscale(class.dissimilarity, eig = TRUE, k = 3)

# allocate each grid cell with the scale value for the three dimensions of its class
cell.Scale <- matrix(NA, nrow(Trans.env.table), 3)
for(i.cell in 1:nrow(Trans.env.table))
{
  cell.Scale[i.cell,] <- Class.MDS$points[cell.class[i.cell],]
} # end for i.cell

# Create a table with coordinates
cell.Scale.norm <- (cell.Scale-min(cell.Scale))/(max(cell.Scale)-min(cell.Scale))
ras.dat <- cbind(Trans.env.table[,c(1,2)],cell.Scale.norm)
colnames(ras.dat) <- c('x','y','r', 'g', 'b')
ras.dat <- ras.dat[complete.cases(ras.dat),]
ras.dat <- as.data.frame(ras.dat)

# convert the data to raster
gdm.cls.1.ras <- raster(transRasts,layer=1)
gdm.cls.1.ras <- rasterize(Trans.env.table[,c(1:2)],gdm.cls.1.ras,field=ras.dat$r)
gdm.cls.2.ras <- raster(transRasts,layer=1)
gdm.cls.2.ras <- rasterize(Trans.env.table[,c(1:2)],gdm.cls.2.ras,field=ras.dat$g)
gdm.cls.3.ras <- raster(transRasts,layer=1)
gdm.cls.3.ras <- rasterize(Trans.env.table[,c(1:2)],gdm.cls.3.ras,field=ras.dat$b)
gdm.cls.stack <- stack(gdm.cls.1.ras, gdm.cls.2.ras, gdm.cls.3.ras)

# plot the data
plotRGB(gdm.cls.stack, stretch="lin")
```

2.10 Expected species persistence given changes in habitat condition

Here we demonstrate how the GDM spatial predictions can be combined with information on the habitat condition of each location to estimate the expected level of species persistence, given habitat loss and degradation.

```
# put the values from the transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]
```

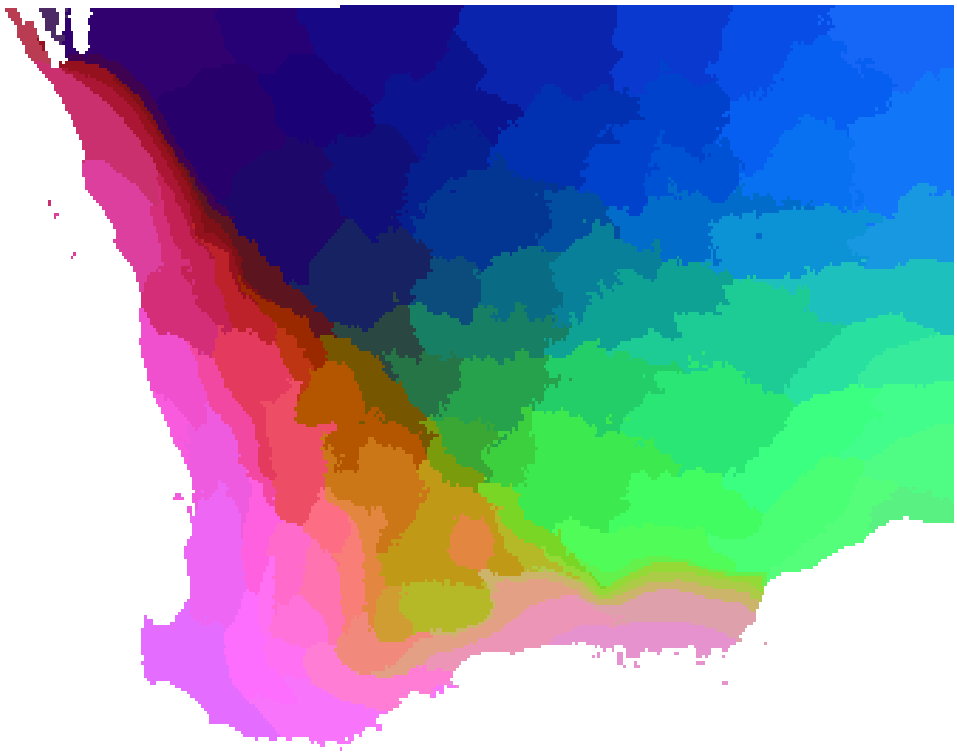


Figure 12: Predicted ecological types: classes that are more similar have more similar colour.

```
# Create data on habitat condition for the region
# specify a simple polygon indicating where habitat loss has occurred
losthab.1 <- rbind(c(116.6, -33.9), c(120.2, -33.4), c(116.4, -29.5), c(116.6, -33.9))
losthab.2 <- rbind(c(120.4, -33.7), c(121.8, -32.8), c(122.4, -33.7), c(120.4, -33.7))
losthab.3 <- rbind(c(116.0, -32.2), c(115.0, -29.9), c(116.0, -29.8), c(116.0, -32.2))
losthab.sply <- SpatialPolygons(list(Polygons(list(Polygon(losthab.1)), ID="a"),
                                     Polygons(list(Polygon(losthab.2)), ID="b"),
                                     Polygons(list(Polygon(losthab.3)), ID="c")),
                                proj4string=crs(transRasts))
# Create a habitat condition raster based on the polygons: 1=intact, 0=lost
cond.ras <- transRasts[[1]]
cond.ras[!is.na(cond.ras)] <- 1
cond.ras <- rasterize(losthab.sply, cond.ras, field=0, update=TRUE)
# Extract the habitat condition data to a table
cond.table <- extract(cond.ras, Trans.env.table[,c(1,2)])
cond.table <- data.frame('x' = Trans.env.table[,1],
                         'y' = Trans.env.table[,2],
                         'cond' = cond.table)

# Select a random sample of cells across the region, to use in comparing each cell to
ref.cells <- sample.int(nrow(Trans.env.table), size = 500, replace = FALSE)

# Calculate the expected species persistence
# NB - This loop takes a couple of minutes
proportion.persisting <- rep(0, length=nrow(Trans.env.table))
```

```
Numerator <- 0
Denominator <- 0
for(i.cell in 1:nrow(Trans.env.table))
{
  # Check if this cell has data
  if(!is.na(Trans.env.table[i.cell,ncol(Trans.env.table)]))
  {
    # initialise the similarity aggregators
    sum.similarity.condition <- 0
    sum.similarity.pristine <- 0
    # loop through the reference cells, calculate similarity, add it to the tally
    for(j.cell in 1:length(ref.cells))
    {
      ecol.dist.1 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                               Trans.env.table[ref.cells[j.cell],c(3:ncol(Trans.env.table))]))
      similarity.ij <- (exp(-1 * (gdmRastMod$intercept + ecol.dist.1)))
      sum.similarity.condition <- sum.similarity.condition +
                                    (cond.table$cond[ref.cells[j.cell]] * similarity.ij)
      sum.similarity.pristine <- sum.similarity.pristine + similarity.ij
    } # end for j.cells
    # Finish by dividing the sum similarity condition by the sum similarity pristine
    # and take to the power of z (here = 0.25) for species-area conversion
    proportion.persisting[i.cell] <- (sum.similarity.condition / sum.similarity.pristine) ^ 0.25
    cell.weight <- 1 / sum.similarity.pristine
    Numerator <-  Numerator + (proportion.persisting[i.cell] * cell.weight)
    Denominator <- Denominator + cell.weight
  } # end if(!is.na())
} # end for i.cell

# Calculate the expected proportion of species persisting across the whole region
Regional.Proportion.Persistence <- Numerator / Denominator

# The proportion of species originally occuring in the region that are expected to
# persist indefinitely given habitat loss =
round(Regional.Proportion.Persistence, digits=3)
```

```
## [1] 0.962
```

```
## Format the cell-based persistence values into a raster and plot them
pers.ras <- raster(transRasts,layer=1)
pers.ras <- rasterize(Trans.env.table[,c(1:2)], pers.ras, field=proportion.persisting)
plot(pers.ras,
     col = colorRampPalette(c("red", "yellow","green"))(100),
     zlim=c(0.9,1.0),
     legend.args = list(text = 'Persistence'))
```

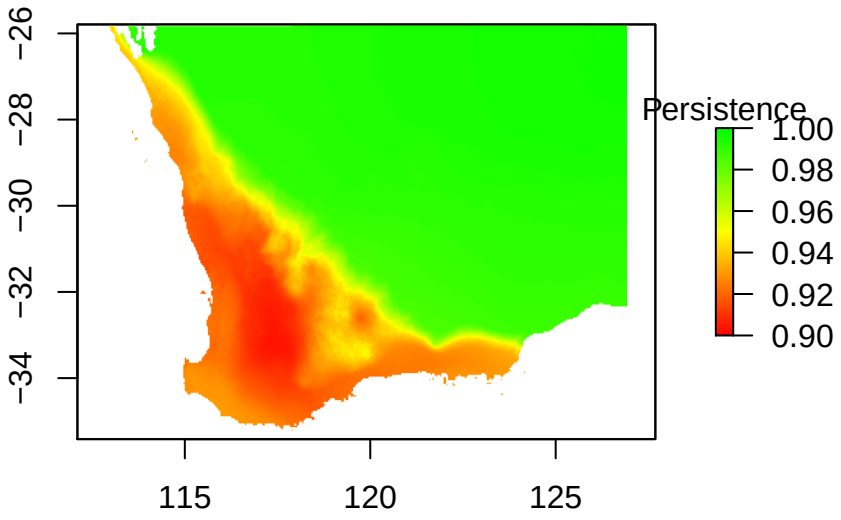


Figure 13: Expected species persistence. Locations with lower values have a lower proportion of species originally occuring there that are expected to persist anywhere in the region, given habitat loss and degradation.

2.11 The marginal biodiversity benefit of habitat restoration and marginal biodiversity loss from habitat loss

This example assess two related metrics. The first is the 'marginal loss' which is the expected loss of biodiversity persistence following complete loss of habitat from each location, individually. The second metric is 'marginal gain' which is the expected gain in biodiversity persistence following complete restoration of the habitat condition in each location, individually.

```
# put the values from the transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]

# Create data on habitat condition for the region
# specify a simple polygon indicating where habitat loss has occurred
losthab.1 <- rbind(c(116.6, -33.9), c(120.2, -33.4), c(116.4, -29.5), c(116.6, -33.9))
losthab.2 <- rbind(c(120.4, -33.7), c(121.8, -32.8), c(122.4, -33.7), c(120.4, -33.7))
losthab.3 <- rbind(c(116.0, -32.2), c(115.0, -29.9), c(116.0, -29.8), c(116.0, -32.2))
losthab.sply <- SpatialPolygons(list(Polygons(list(Polygon(losthab.1)), ID="a"),
                                     Polygons(list(Polygon(losthab.2)), ID="b"),
                                     Polygons(list(Polygon(losthab.3)), ID="c")),
                                proj4string=crs(transRasts))
```

```
# Create a habitat condition raster based on the polygons: 1=intact, 0=lost
cond.ras <- transRasts[[1]]
cond.ras[!is.na(cond.ras)] <- 1
cond.ras <- rasterize(losthab.sply, cond.ras, field=0, update=TRUE)
# Extract the habitat condition data to a table
cond.table <- extract(cond.ras, Trans.env.table[,c(1,2)])
cond.table <- data.frame('x' = Trans.env.table[,1],
                         'y' = Trans.env.table[,2],
                         'cond' = cond.table)

# Select a random sample of cells across the region, to use in standardising mean
# similarity to survey sites.
# specify the number of randomly selcted reference cells
ref.cells <- sample.int(nrow(Trans.env.table), size = 500, replace = FALSE)

# Calculate the expected species persistence
# NB - This loop takes a couple of minutes
marginal.loss <- rep(0, length=nrow(Trans.env.table))
marginal.gain <- rep(0, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  # Check if this cell has data
  if(!is.na(Trans.env.table[i.cell,ncol(Trans.env.table)]))
  {
    # initialise the similarity aggregators
    sum.similarity.condition <- 0
    sum.similarity.pristine <- 0
    # loop through the reference cells, calculate similarity, add it to the tally
    for(j.cell in 1:length(ref.cells))
    {
      ecol.dist.1 <- sum(abs(Trans.env.table[i.cell,c(3:ncol(Trans.env.table))] -
                             Trans.env.table[ref.cells[j.cell],c(3:ncol(Trans.env.table))]))
      similarity.ij <- (exp(-1 * (gdmRastMod$intercept + ecol.dist.1)))
      sum.similarity.condition <- sum.similarity.condition + (cond.table$cond[ref.cells[j.cell]]
                                                              * similarity.ij)
      sum.similarity.pristine <- sum.similarity.pristine + similarity.ij
    } # end for j.cells
    # And now include the focal cell, both with it's actual condition, and if its condition
    # was 0 (loss) or 1 (gain)
    sum.similarity.condition <- sum.similarity.condition + cond.table$cond[i.cell]
    sum.similarity.pristine <- sum.similarity.pristine + 1
    sum.similarity.loss <- sum.similarity.condition - cond.table$cond[i.cell]
    sum.similarity.gain <- sum.similarity.condition + (1 - cond.table$cond[i.cell])
    # Finish by dividing the sum similarity condition by the sum similarity pristine
    # and take to the power of z (here = 0.25) for species-area conversion
    proportion.persisting <- (sum.similarity.condition / sum.similarity.pristine) ^ 0.25
    proportion.persisting.loss <- (sum.similarity.loss / sum.similarity.pristine) ^ 0.25
    proportion.persisting.gain <- (sum.similarity.gain / sum.similarity.pristine) ^ 0.25
    marginal.loss[i.cell] <- proportion.persisting - proportion.persisting.loss
    marginal.gain[i.cell] <- proportion.persisting.gain - proportion.persisting
  } # end if(!is.na())
} # end for i.cell
```

```
# Format the cell-based marginal loss values into a raster and plot them
loss.ras <- raster(transRasts,layer=1)
loss.ras <- rasterize(Trans.env.table[,c(1:2)], loss.ras, field=marginal.loss)
plot(loss.ras,
     col = colorRampPalette(c("bisque", "yellow", "red"))(100),
     zlim=c(0.0,0.01),
     legend.args = list(text = 'Marginal Loss'))
```

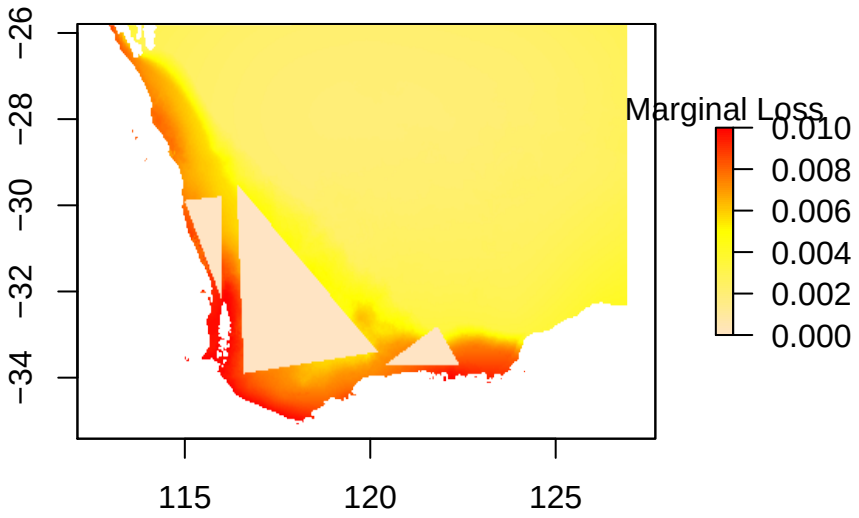


Figure 14: Marginal loss of biodiversity following future habitat loss. The locations with the highest values would see the greatest loss of biodiversity persistence for the species originally occuring there if the remaining habitat in that location was lost. Habitat already lost (here as simplified polygons) has marginal loss value of zero.

Now plot marginal gain...

```
# Format the cell-based marginal gain values into a raster and plot them
gain.ras <- raster(transRasts,layer=1)
gain.ras <- rasterize(Trans.env.table[,c(1:2)], gain.ras, field=marginal.gain)
plot(gain.ras,
     col = colorRampPalette(c("bisque", "green","darkblue"))(100),
     zlim=c(0.0,0.01),
     legend.args = list(text = 'Marginal Gain'))
```

2.12 Considering climate change

The next set of examples demonstrate use of GDM to undertake biodiversity assessments associated with potential future climate change. To implement these analyses, we first need to generate GDM predictions (transformed layers) for future climates.

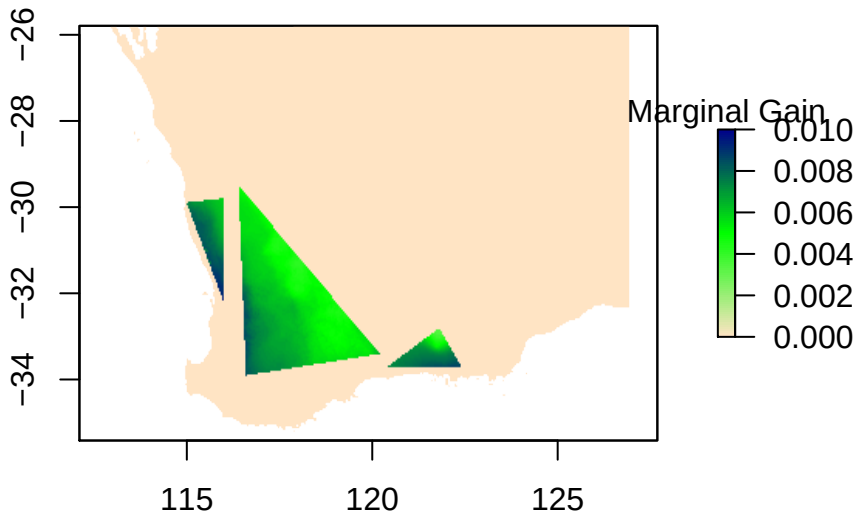


Figure 15: Marginal gain of biodiversity following future habitat restoration. The locations with the highest values would see the greatest gain in biodiversity persistence for the species originally occuring there if the habitat in that location was fully restored. Habitat already intact (here represented with simplified polygons) has marginal gain value of zero.

```
# Make some hypothetical climate change data
futRasts <- envRast

##reduce winter precipitation by 25% & increase temps
futRasts[[3]] <- futRasts[[3]]*0.75
futRasts[[4]] <- futRasts[[4]]+2
futRasts[[5]] <- futRasts[[5]]+3

# create model transformed env layers (predictions) for the climate change env layers
transRasts.fut <- gdm.transform(model=gdmRastMod,
                                data=futRasts)
```

2.13 Predicting the potential change for each location

The simplest climate change analysis with GDM is to predict the expected change (turnover) for each location, from current climate to a future climate.

```
# For this analysis, we simply need to use the 'predict' function in the gdm package,
# specifying the 'predRasts' argument, and the 'time' argument
timePred <- predict(object=gdmRastMod,
                    data=envRast,
                    time=T,
                    predRasts=futRasts)
plot(timePred)
```

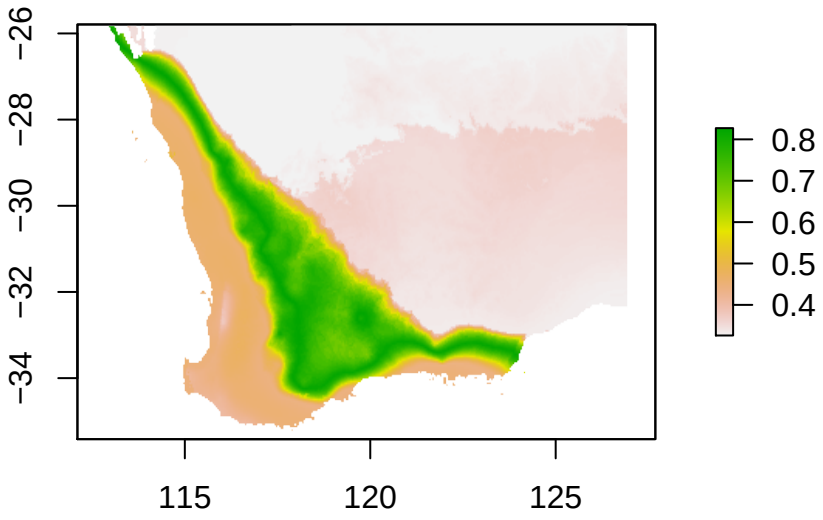


Figure 16: Predicted magnitude of biological change through time

2.14 Identifying potential climate change refugia

This example uses GDM predictions for both the current and future climate to assess possible areas of refugia under climate change.

```
# put the values from the current climate transformed layers in a table for easy analysis
Trans.env.table <- as.matrix(transRasts)
col.longs<-xFromCol(transRasts)
row.lats<-yFromRow(transRasts)
Cell_Long<-rep(col.longs, times=nrow(transRasts))
Cell_Lat<-rep(row.lats, each=ncol(transRasts), times=1)
Trans.env.table<-cbind(Cell_Long, Cell_Lat, Trans.env.table)
Trans.env.table <- Trans.env.table[complete.cases(Trans.env.table),]


# put the values from the future transformed layers in a table for easy analysis
Trans.env.table.fut <- as.matrix(transRasts.fut)
col.longs<-xFromCol(transRasts.fut)
row.lats<-yFromRow(transRasts.fut)
Cell_Long<-rep(col.longs, times=nrow(transRasts.fut))
Cell_Lat<-rep(row.lats, each=ncol(transRasts.fut), times=1)
Trans.env.table.fut<-cbind(Cell_Long, Cell_Lat, Trans.env.table.fut)
Trans.env.table.fut <- Trans.env.table.fut[complete.cases(Trans.env.table.fut),]


# Run the refugia analysis
# specify the radius to use in assessing refugia
rad <- 0.15 # about 15km

# Calculate the similarity of all other grid cells to each of these focal locations
# NB - This loop takes a couple of minutes
refugia <- rep(0, length=nrow(Trans.env.table))
for(i.cell in 1:nrow(Trans.env.table))
{
  # Check if this cell has data
  if(!is.na(Trans.env.table[i.cell,ncol(Trans.env.table)]))
  {
    # calculate the distance of all the cells to the focal cell
    cells.dist <- sqrt(((Trans.env.table[i.cell,1] - Trans.env.table[,1])^2) +
                         ((Trans.env.table[i.cell,2] - Trans.env.table[,2])^2))
    # Grab the cells within the specified radius of the focal cell
    rad.cells <- which(cells.dist <= rad)
    # set up the catching objects
    sum.s.iFut.jPres <- 0
    sum.s.iFut.jFut <- 0
    # loop through the neighbouring cells, calculate the two similarities, add them to the tally
    for(j.cell in 1:length(rad.cells))
      {
      ecol.dist.1 <- sum(abs(Trans.env.table.fut[i.cell,c(3:ncol(Trans.env.table.fut))] - Trans.env.tabl
      sum.s.iFut.jPres <- sum.s.iFut.jPres + (exp(-1 * ecol.dist.1))
      ecol.dist.2 <- sum(abs(Trans.env.table.fut[i.cell,c(3:ncol(Trans.env.table.fut))] - Trans.env.tabl
      sum.s.iFut.jFut <- sum.s.iFut.jFut + (exp(-1 * ecol.dist.2))
    } # end for j.cells
    # Finish by dividing by the number of neighbouring cells to get the mean
    refugia[i.cell] <- sum.s.iFut.jPres / sum.s.iFut.jFut
```

```
  } # end if(!is.na())
} # end for i.cell

# Format the cell-based refugia values into a raster and plot them
refugia.ras <- raster(transRasts,layer=1)
refugia.ras <- rasterize(Trans.env.table[,c(1:2)], refugia.ras, field=refugia)
plot(refugia.ras,
     col = colorRampPalette(c("bisque", "green","darkblue"))(100),
     zlim=c(0.25,1.06),
     legend.args = list(text = 'refugia'))
```

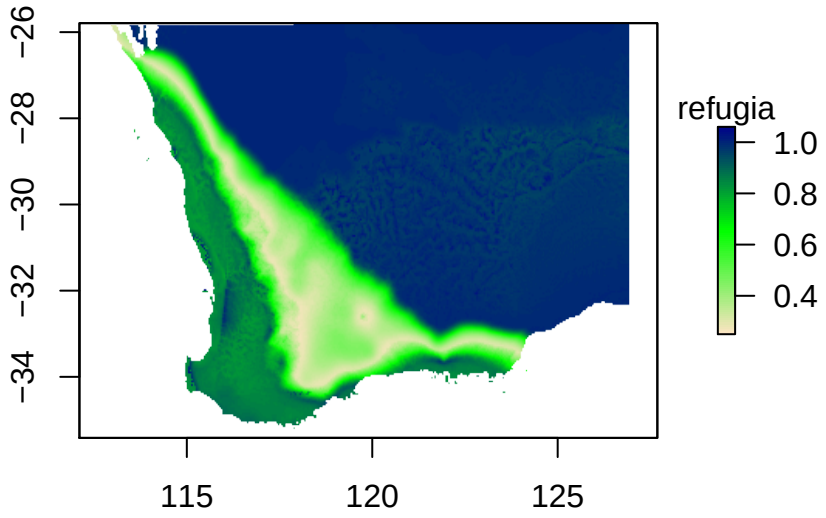


Figure 17: Potential climate change refugia. Locations with higher values have greater expected potential to act as refugia under climate change.

2.15 Identifying potential novel and disappearing habitats under climate change

This example uses GDM spatial predictions under both current and future climate to estimate two related metrics: novel habitats expected under climate change, and; dissapearing current habitats expected under climate change.

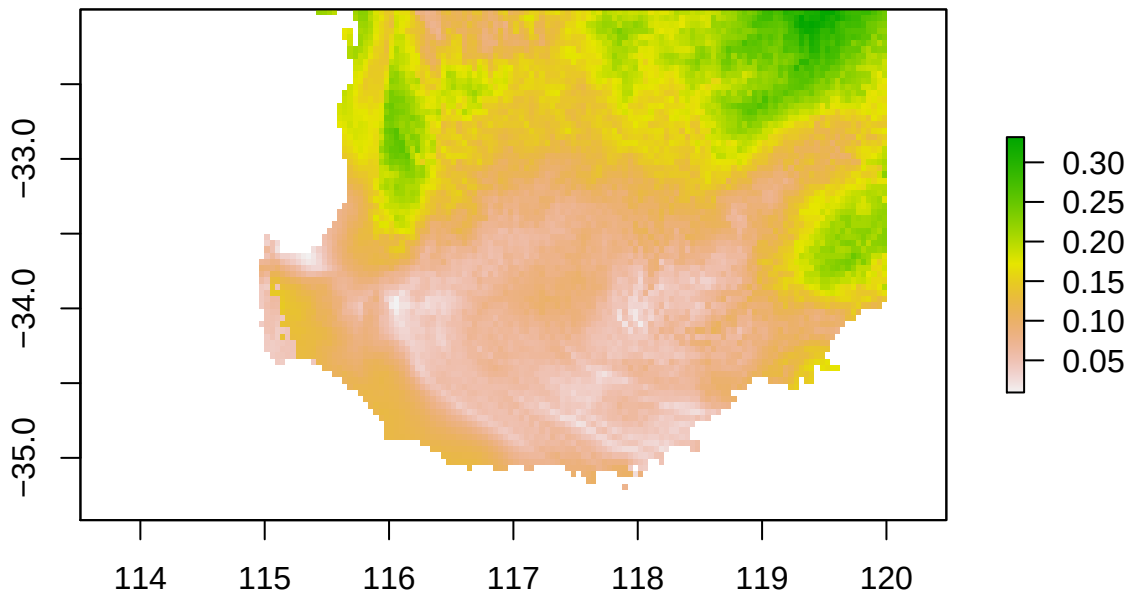
Generic data preparation

```
## load libraries
library(gdm)
library(raster)
library(sp)

## Load data from the gdm package
load(system.file("./data/gdm.RData", package="gdm"))
rastFile <- system.file("./extdata/stackedVars.grd", package="gdm")

## grab the columns with xy, site ID, and species data
sppTab <- gdmExpData[, c("species", "site", "Lat", "Long")]

## prepare the gdm input table, using rasters as predictors
envRast <- stack(rastFile)
sitePairRast <- formatsitepair(bioData=sppTab,
                               bioFormat=2,
                               XColumn="Long",
                               YColumn="Lat",
                               sppColumn="species",
                               siteColumn="site",
                               predData=envRast)
sitePairRast <- na.omit(sitePairRast)

## fit the GDM
gdmRastMod <- gdm(data=sitePairRast,
                  geo=TRUE)

## Generate transformed predictor rasters, based on the raw raster predictor layers
transPresRast <- gdm.transform(model=gdmRastMod,
                               data=envRast)

# crop to smaller area to speed computation
envRast <- crop(envRast, extent(114, 120, -35.41667, -32))

## Make some fake climate change rasters
futRast <- envRast
futRast$bio19 <- futRast$bio19*0.8 # reduce winter precip
futRast$bio5 <- futRast$bio5*1.1 # increase temperature
futRast$bio6 <- futRast$bio6*1.18 # increase temperature
futClimDat <- as.data.frame(futRast, xy=TRUE, na.rm=TRUE)

#Getting all coordinates in range map
gridDat <- na.omit(as.data.frame(envRast, xy=TRUE))
gridDat <- data.frame(distance=1, weight=1, gridDat)
gridDat <- split(gridDat, seq(nrow(gridDat)))

predNames <- names(envRast)
```

```
#function to remove intercept from gdm predictions
removeIntercept <- function(mod,pred){
  adjust <- 0 - log(1-pred) - mod$intercept
  adjustDissim <- 1-exp(0-adjust)
  return(adjustDissim)
}
```

Disappearing habitats calculation.

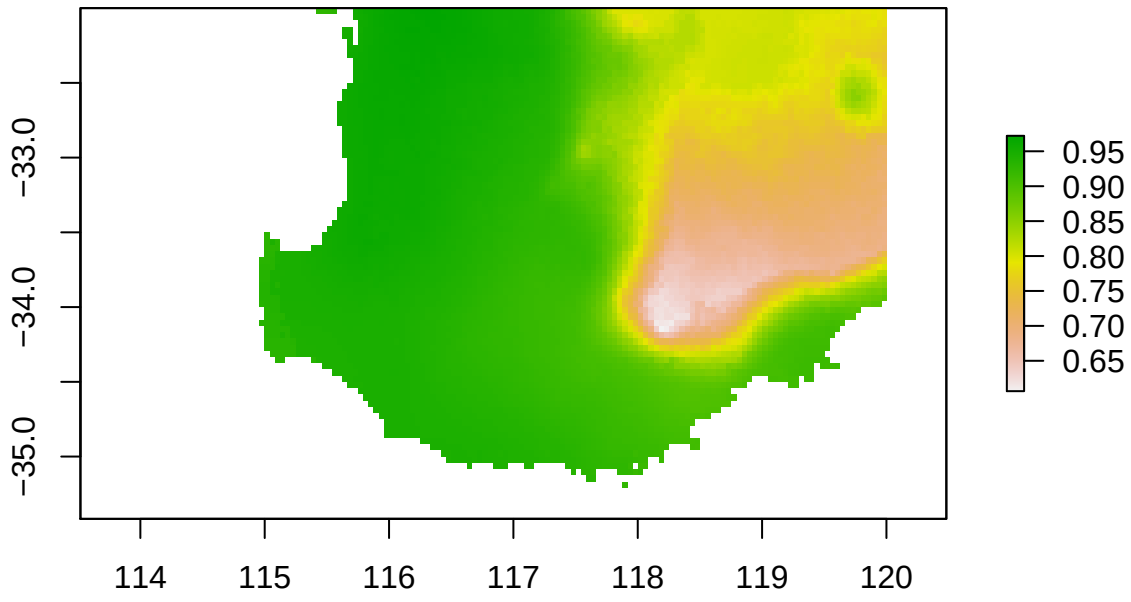
```
cl <- parallel::makeCluster(10, setup_strategy = "sequential") #ideally should be run in parallel to re
registerDoParallel(cl)
disappearGDM <- foreach(i = 1:length(gridDat), .packages=c("gdm")) %dopar%{

  #get the focal cell
  oneCell <- gridDat[[i]]

  #set up a dataframe where the first site is the focal cell, and the other locations are cells across
  setUp <- cbind(oneCell, futClimDat)
  colnames(setUp) <- c("distance","weights",
                       "s1.xCoord", "s1.yCoord",paste("s1.", predNames, sep=""),
                       "s2.xCoord", "s2.yCoord",paste("s2.", predNames, sep=""))

  #rearrange the colums for the gdm prediction
  dat <- setUp[,c("distance","weights","s1.xCoord", "s1.yCoord","s2.xCoord", "s2.yCoord",
                  paste("s1.", predNames, sep=""),
                  paste("s2.", predNames, sep=""))]

  #do the prediction and set up a dataframe with second sites x/y and predicted Fst
  predDiss <- predict(object=gdmRastMod, dat, time=FALSE)
  predDiss <- min(removeIntercept(gdmRastMod, predDiss))
}

stopCluster(cl)

mask <- envRast[[1]]
mask[which(!is.na(mask[]))] <- unlist(disappearGDM)
plot(mask)
```



Novel habitats calculation

```
#set up for prediction
futClimDat <- data.frame(distance=1, weight=1, futClimDat)
futClimDat <- split(futClimDat, seq(nrow(futClimDat)))

#Getting all coordinates in range map
gridDat <- na.omit(as.data.frame(envRast, xy=TRUE))
gridDat <- data.frame(distance=1, weight=1, gridDat)

###############
#Reverse offset calculation
###############
cl <- parallel::makeCluster(10, setup_strategy = "sequential")
registerDoParallel(cl)
novelGDM <- foreach(i = 1:length(futClimDat), .packages=c("fields","gdm","geosphere")) %dopar%{

  ##get the focal cell
  oneCell <- futClimDat[[i]]

  #set up a dataframe where the first site is the focal cell (at time=2), and the other locations are c
  setUp <- cbind(oneCell, gridDat)
  colnames(setUp) <- c("distance","weights",
                       "s1.xCoord", "s1.yCoord",paste("s1.", predNames, sep=""),
                       "s2.xCoord", "s2.yCoord",paste("s2.", predNames, sep=""))

  #rearrange the colums for the gdm prediction
```

```
  dat <- setUp[,c("distance","weights","s1.xCoord", "s1.yCoord","s2.xCoord", "s2.yCoord",
                  paste("s1.", predNames, sep=""),
                  paste("s2.", predNames, sep=""))]

  #do the prediction and set up a dataframe with second sites x/y and predicted Fst
  predDiss <- predict(object=gdmRastMod, dat, time=FALSE)
  predDiss <- min(removeIntercept(gdmRastMod, predDiss))
}

stopCluster(cl)


mask <- envRast[[1]]
mask[which(!is.na(mask[]))] <- unlist(novelGDM)
plot(mask)
```



3 SECTION 3 - Further exploring GDM model fitting

3.1 Transforming predictor variables can influence model performance

This example demonstrates the potential for improved model performance through transforming highly skewed predictor data.

```
# Load libraries
library(moments) # to calculate skewness

# Set up site-pair table, environmental tabular data
sppData <- gdmExpData[c(1,2,13,14)]
envTab <- gdmExpData[c(2:ncol(gdmExpData))]
sitePairTab <- formatsitepair(bioData=sppData,
                              bioFormat=2,
                              XColumn="Long",
                              YColumn="Lat",
                              sppColumn="species",
                              siteColumn="site",
                              predData=envTab)

# Using the data from the gdm package, we will demonstrate the effect of transforming a
#  predictor (phTotal), when applying the default model fitting settings
# First create an environment table with just the 'phTotal' variable
envTab <- gdmExpData[c(2,4)]
# then create several alternative transformations of this variable
envTab$phTotal_cube <- envTab$phTotal^3
envTab$phTotal_cuberoot <- envTab$phTotal^(1/3)
envTab$phTotal_log10 <- log10(envTab$phTotal)
envTab$phTotal_scale <- scale(envTab$phTotal, center = TRUE, scale = TRUE)
envTab$phTotal_exp <- exp(envTab$phTotal/100)

# Have a look at how these transformations have changed the distribution of values
par(mfrow=c(3,2))
hist(envTab$phTotal, main = "Untransformed")
hist(envTab$phTotal_cube, main = "Cubed")
hist(envTab$phTotal_cuberoot, main = "Cube-root")
hist(envTab$phTotal_log10, main = "Log10")
hist(envTab$phTotal_scale, main = "Scaled")
hist(envTab$phTotal_exp, main = "Exponential")
```

Now compare the performance of GDMs fit using these different distributions of the predictor variable.

```
# Fit GDMs to each data distribution
variable.name <- colnames(envTab)[2:7]
variable.skewness <- c()
deviance.explained <- c()
for(i.var in 2:7)
  {
  sitePairTab <- formatsitepair(bioData=sppData,
                                bioFormat=2,
                                XColumn="Long",
                                YColumn="Lat",
                                sppColumn="species",
                                siteColumn="site",
                                predData=envTab[,c(1,i.var)])
```

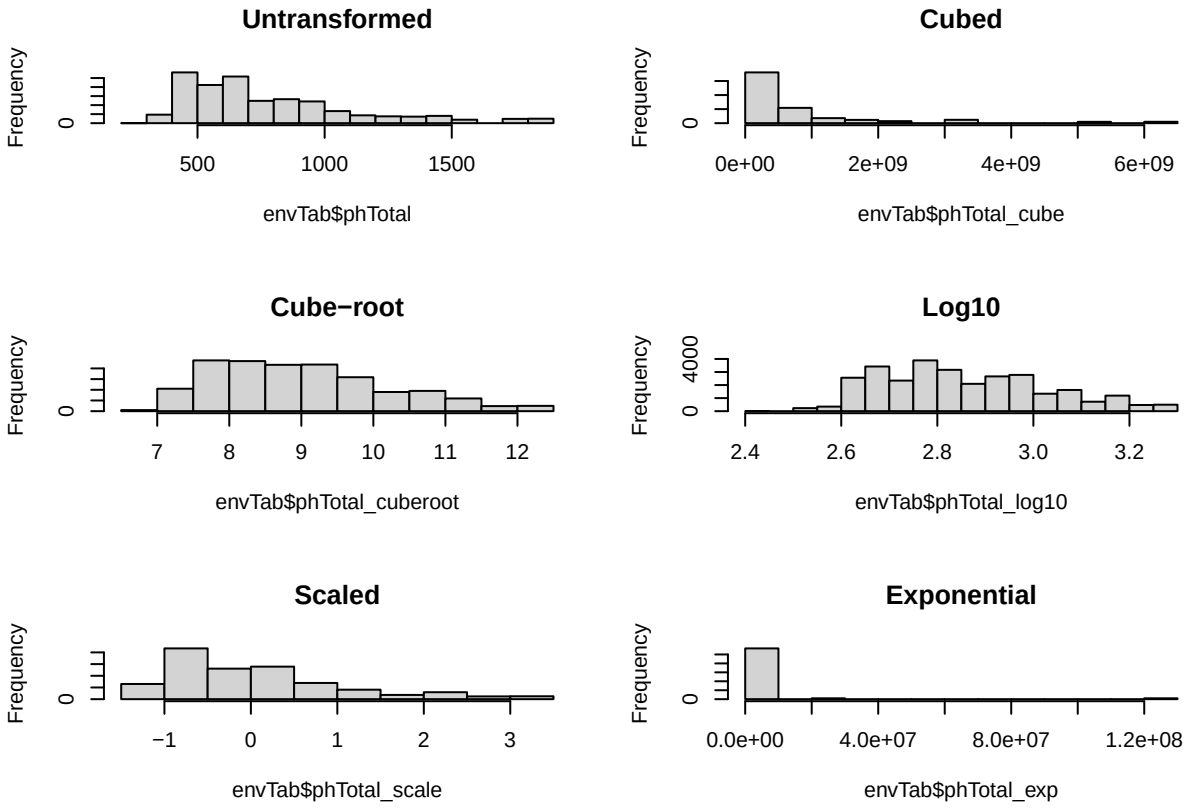


Figure 18: Alterative distributions (from transformations) of the predictor variable

```
  gdmTabMod <- gdm(data=sitePairTab,
                   geo=FALSE)
  deviance.explained <- c(deviance.explained, gdmTabMod$explained)
  variable.skewness <- c(variable.skewness, skewness(envTab[,i.var]))
  }# end for i.var

# And lets see how model performance changes with the skewness of the variable
par(mfrow=c(1,1))
plot(variable.skewness, deviance.explained)
```

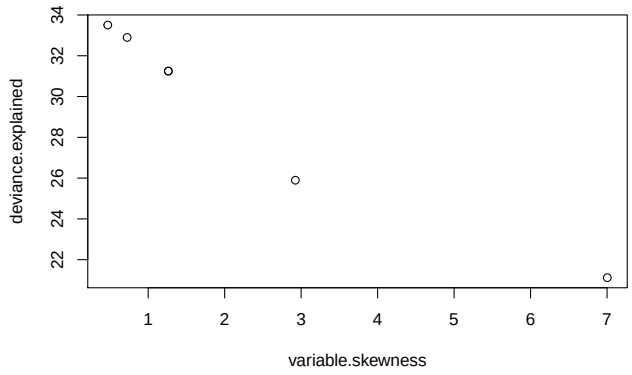


Figure 19: This example suggests that reducing the skewness of the predictor data in GDM will improve model performance, at least when using the default number of splines and position of knots

3.2 Sub-sampling site-pairs can influence model performance

This example demonstrates the implications of subsampling site-pairs on model performance, both in terms of deviance explained in the training data, and mean absolute error for independent cross-validation data.

```
library(gdm)

# load example data
load(system.file("./data/gdm.RData", package="gdm"))

# Set up site-pair table, environmental tabular data
sppData <- gdmExpData[c(1,2,13,14)]
envTab <- gdmExpData[c(2:ncol(gdmExpData))]
asp <- formatsitepair(bioData=sppData,
                      bioFormat=2,
                      XColumn="Long",
                      YColumn="Lat",
                      sppColumn="species",
                      siteColumn="site",
                      predData=envTab,
                      sampleSites = 1) # use all (100%) data

# using all sites returns 4371 site pairs
dim(asp)
```

```
## [1] 4371   26
```

```
# set up and run subsampling of site-pair table
subSamps <- replicate(99, c(seq(0.05, 0.25, by=0.025), seq(0.3, 0.95, by=0.15)))

subSampGDMs <- apply(subSamps, c(1,2), function(x){
  # spt for modeling TRAINING
  sitePairTab.train <- formatsitepair(bioData=sppData,
                                      bioFormat=2,
                                      XColumn="Long",
                                      YColumn="Lat",
                                      sppColumn="species",
                                      siteColumn="site",
                                      predData=envTab,
                                      sampleSites = x)

  # spt for model EVALUATION
  # start with full spt, then subsample by removing rows
  # that are already in the training spt
  sitePairTab.test <- formatsitepair(bioData=sppData,
                                     bioFormat=2,
                                     XColumn="Long",
                                     YColumn="Lat",
                                     sppColumn="species",
                                     siteColumn="site",
                                     predData=envTab,
                                     sampleSites = 1)
  # spt for model EVALUATION
  # remove rows that are in the training spt
  sitePairTab.test <- sitePairTab.test[which((rownames(sitePairTab.test) %in%
                                                rownames(sitePairTab.train))==FALSE),]
```

```
  # model fit to TRAINING spt
  modTrain <- gdm(sitePairTab.train)

  # predict model to EVALUATION data
  predTest <- predict(modTrain, sitePairTab.test)

  # mean absolute error (mae)
  mae <- mean(abs(sitePairTab.test$distance - predTest))

  return(c(modTrain$explained, mae))
})
```

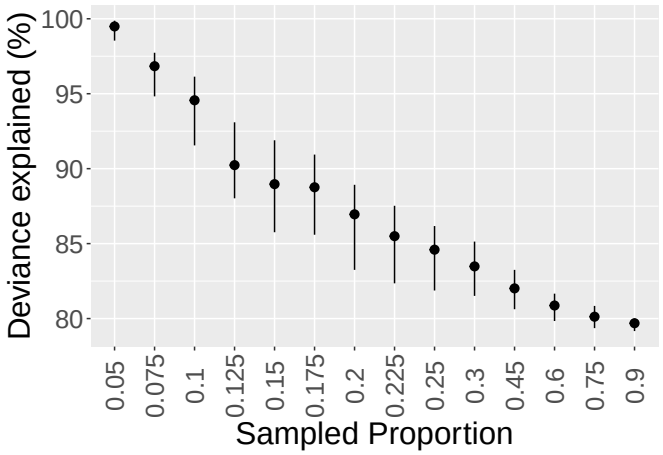


Figure 20: Influence of random site-pair subsampling on model performance - Deviance Explained

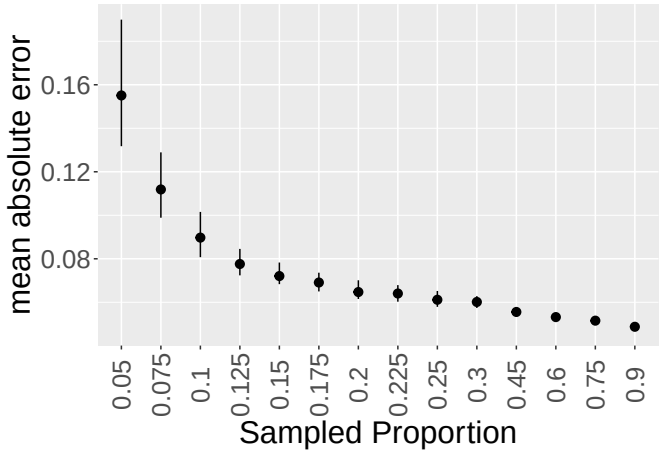


Figure 21: Influence of random site-pair subsampling on model performance - Mean Absolute Error

This example shows that while models appear to perform better in terms of deviance explained in the training data as fewer site-pairs are used, independent tests of model accuracy show increasing error as fewer site-pairs are used.

3.3 Including a higher proportion of low dissimilarity site-pairs can influence model performance

This example explores the implications of subsampling site-pairs for use in a GDM based on the spatial distance between site-pairs. This approach assumes site-pairs that are closer together with have lower dissimilarity, and so using a larger proportion of site-pairs that are nearer to each other will improve the predictive accuracy for lower values of dissimilarity.

```
# Load the gdm library
library(gdm)

# read in example input data
load(system.file("./data/gdm.RData", package="gdm"))

# point to the data
sppData <- gdmExpData[c(1,2,13,14)]
sppData <- unique(sppData)
envTab <- gdmExpData[c(2:ncol(gdmExpData))]
envTab <- unique(envTab)

# Separate the data into training (85%) and testing sites (15%)
test.sites <- envTab$site[sample.int(n=length(envTab$site), size=floor(0.15*length(envTab$site)))]
sppData.train <- sppData[which(!sppData$site %in% test.sites),]
sppData.test <- sppData[which(sppData$site %in% test.sites),]

# create the gdm input tables
sitePairTab.train <- formatsitepair(bioData=sppData.train,
                                    bioFormat=2,
                                    XColumn="Long",
                                    YColumn="Lat",
                                    sppColumn="species",
                                    siteColumn="site",
                                    predData=envTab)
sitePairTab.test <- formatsitepair(bioData=sppData.test,
                                   bioFormat=2,
                                   XColumn="Long",
                                   YColumn="Lat",
                                   sppColumn="species",
                                   siteColumn="site",
                                   predData=envTab)

# create a version of the training table where the site-pairs are sampled based on their
#  geographic separation (less sitepairs that are further apart)
# First generate a geographic weighting
geodist <- sqrt(((sitePairTab.train$s1.xCoord - sitePairTab.train$s2.xCoord)^2) +
                  ((sitePairTab.train$s1.yCoord - sitePairTab.train$s2.yCoord)^2))
# linear weighting
weighting <- 1-(geodist/max(geodist))

# Now subsample the site pairs
# purely randomly for the unweighted data
sitePairTab.train.unwt <- sitePairTab.train[sample.int(n=nrow(sitePairTab.train),
                                                       size=floor(nrow(sitePairTab.train)/2)),]
# and alternatively, using the geographic weighting to inform random selection of site pairs
sitePairTab.train.wt <- sitePairTab.train[sample.int(n=nrow(sitePairTab.train),
```

```
                                                    size=floor(nrow(sitePairTab.train)/2),
                                                    prob=weighting),]

# Fit a  gdm to both the unweighted similarities and the geographically weighted similarities
# Here we're not using geographic distance as a predictor, given we've used it to weight the
#  similarities
mod.unwt <- gdm(sitePairTab.train.unwt, geo=FALSE)
mod.wt <- gdm(sitePairTab.train.wt, geo=FALSE)

# In terms of deviance explained in the training data, compare the model using random
#  site-pairs with that using more nearby site-pairs

# random sitepairs - explained dissimilarity for the training data
mod.unwt$explained
```

```
## [1] 77.19531
```

```
# nearby sitepairs - explained dissimilarity for the training data
mod.wt$explained
```

```
## [1] 77.28542
```

```
# But let's assess how each model performs in predicting the more similar sitepairs, using
#  the testing data, let's calculate the mean absolute error for the sitepairs in the test
#  set that are <= 0.5 dissimilarity
sitePairTab.test.lowest <- sitePairTab.test[which(sitePairTab.test$distance <= 0.5),]
predicted.dissim.unwt <- predict(mod.unwt, sitePairTab.test.lowest)
predicted.dissim.wt <- predict(mod.wt, sitePairTab.test.lowest)
mae.unwt <- mean(abs(sitePairTab.test.lowest$distance - predicted.dissim.unwt))
mae.wt <- mean(abs(sitePairTab.test.lowest$distance - predicted.dissim.wt))

# In terms of mean absolute error in predicting lower levels of dissimilarity, using
#  independent cross-validation data:

# random sitepairs - mean absolute error for dissimilarities < 0.5
mae.unwt
```

```
## [1] 0.08837678
```

```
# nearby sitepairs - mean absolute error for dissimilarities < 0.5
mae.wt
```

```
## [1] 0.1055081
```

```
# Also view as a boxplot
err.unwt <- predicted.dissim.unwt - sitePairTab.test.lowest$distance
err.wt <- predicted.dissim.wt - sitePairTab.test.lowest$distance
error.dat <- data.frame('error' = c(err.unwt, err.wt),
                        'data.type' = c(rep('random_sitepairs',
                                            times=length(err.unwt)),
                                        rep('closer_sitepairs',
                                            times=length(err.wt))))
boxplot(error~data.type,
        error.dat,
        ylim=c(min(error.dat$error)-0.01, max(error.dat$error)+0.01))
```

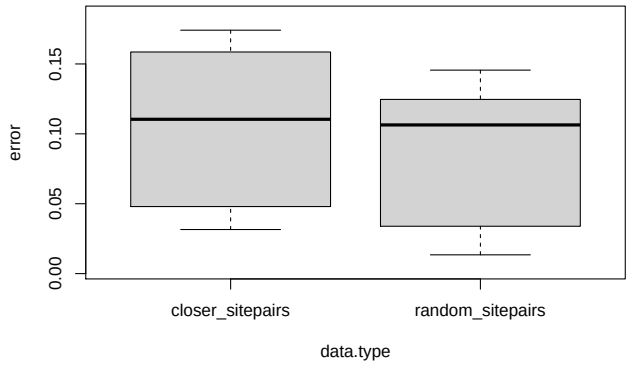


Figure 22: Error in predicted dissimilarities where observed dissimilarity <0.5, using either randomly slected site-pairs, or sitepairs that are closer together geographically.

3.4 Increasing the number of splines can influence model performance

This example examines the implications of increasing the number of splines for a predictor when fitting a GDM. The outcomes are compared in terms of deviance explained for the data used to fit the model, and mean absolute error in predictions for independent cross-validation data.

```
library(ggplot2)

# load example data
load(system.file("./data/gdm.RData", package="gdm"))

## Set up site-pair table, environmental tabular data
sppData <- gdmExpData[, c(1,2,13,14)]
envTab <- gdmExpData[, c(2:ncol(gdmExpData))]

# spt for modeling training
spt.Train <- replicate(99, formatsitepair(sppData,
                                          2,
                                          XColumn="Long",
                                          YColumn="Lat",
                                          sppColumn="species",
                                          siteColumn="site",
                                          predData=envTab,
                                          sampleSites = 0.75), simplify = F)

sitePairTab.test <- formatsitepair(sppData,
                                   2,
                                   XColumn="Long",
                                   YColumn="Lat",
                                   sppColumn="species",
                                   siteColumn="site",
                                   predData=envTab,
                                   sampleSites = 1) # use all (100%) data

spt.Test <- lapply(spt.Train, function(x, sptIn){
  sitePairTab.test[which((rownames(sitePairTab.test) %in% rownames(x))==FALSE),]
```

```
}, sptIn = sitePairTab.test)

# setup splines
nSplines <- seq(3, 10, by=1)
#nPreds <- (ncol(sitePairTab.train)-6)/2
nPreds <- (ncol(spt.Train[[1]])-6)/2

modMetrics <- list()
for(i in 1:length(nSplines)){
  #print(i)
  gdmSplines <- lapply(1:length(spt.Train), function(x,
                                                     trainDat,
                                                     testDat,
                                                     splineI){
    # model fit to training spt
    modTrain <- gdm(trainDat[[x]],
                    splines = rep(splineI, times=10))
    # predict model to testing data
    predTest <- predict(modTrain, testDat[[x]])

    # mean absolute error
    mae <- mean(abs(testDat[[x]]$distance - predTest))

    return(c(modTrain$explained, mae))
  }, trainDat=spt.Train, testDat=spt.Test, splineI=nSplines[i])

  modMetrics[[i]] <- data.frame(nSplines=nSplines[i], do.call(rbind, gdmSplines))
}

modMetrics <- do.call(rbind, modMetrics)
names(modMetrics) <- c("nSplines", "devExp", "mae")

devExpTab <- aggregate(devExp~nSplines, data=modMetrics, FUN=quantile, prob=c(0.25, 0.5, 0.75))
devExpTab <- data.frame(nSplines=devExpTab$nSplines, devExp=devExpTab$devExp)

maeTab <- aggregate(mae~nSplines, data=modMetrics, FUN=quantile, prob=c(0.25, 0.5, 0.75))
maeTab <- data.frame(nSplines=maeTab$nSplines, mae=maeTab$mae)
```

This example shows that while model fit can appear to improve for the training data when more splines are used, the accuracy of the model in predicting dissimilarity for independent cross-validation data may sometimes decrease, due to over-fitting to the training data.

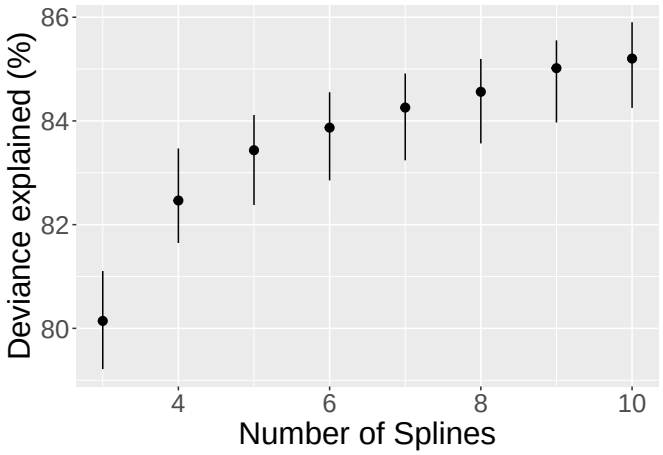


Figure 23: Influence of Number of Splines - Deviance Explained

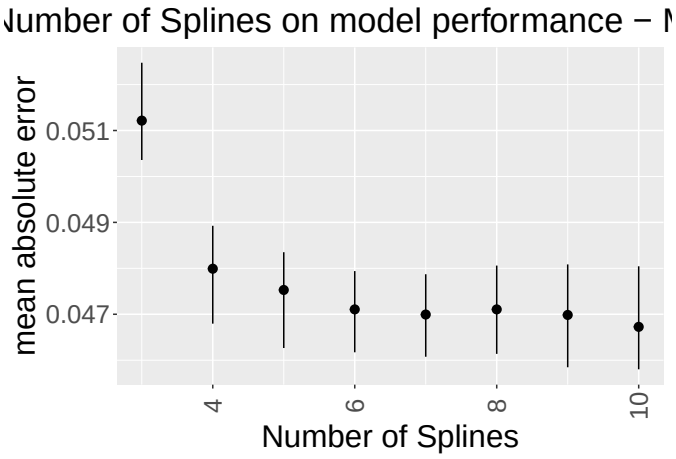


Figure 24: Influence of Number of Splines - - Mean Absolute Error

3.5 Correlation between predictor variables and between site-pair predictors

This example considers how the correlation between two predictor variables is related to the correlation between the site-pair differences for those two predictors, which is what is considered in GDM.

```
# Set up site-pair table, environmental tabular data
sppData <- gdmExpData[c(1,2,13,14)]
envTab <- gdmExpData[c(2:ncol(gdmExpData))]
# create sitepair table
sitePairTab <- formatsitepair(bioData=sppData,
                              bioFormat=2,
                              XColumn="Long",
                              YColumn="Lat",
                              sppColumn="species",
                              siteColumn="site",
                              predData=envTab)

# Calculate correlation coefficient for predictor variables
env.preds <- c("awcA","phTotal","sandA","shcA","solumDepth","bio5","bio6","bio15","bio18","bio19")
env.cor <- cor(envTab[,which(colnames(envTab) %in% env.preds)], method = "pearson")

# Calculate correlation coefficients for site-pairs
env.pair.dif <- matrix(0, ncol = length(env.preds), nrow = nrow(sitePairTab))
colnames(env.pair.dif) <- paste0(env.preds,"_diff")
for(i in 1:length(env.preds))
{
  env.pair.dif[,i] <- abs(sitePairTab[,which(colnames(sitePairTab) == paste0("s1.",env.preds[i]))] -
                            sitePairTab[,which(colnames(sitePairTab) == paste0("s2.",env.preds[i]))])
} # end for i
env.dif.cor <- cor(env.pair.dif, method = "pearson")

# compare the correlation coefficients for the predictors vs the differnece between the predictors
plot(abs(env.cor),abs(env.dif.cor), xlab="Predictor correlation", ylab="Predictor-pair correlation")
lines(c(0,1),c(0,1),lty=3)
```

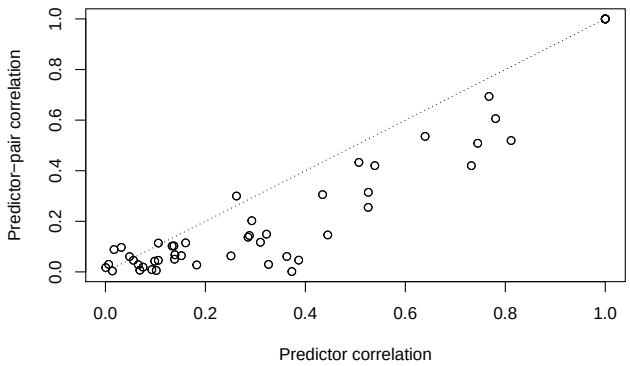


Figure 25: Correlation between the site-pair differences in the values of two predictors v.s. correlation between the predictor values of sites for two predictors.

```
# calculate the average site-pair differnce correlation relative to the site predictor correlation
mean(env.cor)
```

```
## [1] 0.2577607
```

```
mean(env.dif.cor)
```

```
## [1] 0.2084643
```

```
cor.ratio <- mean(env.dif.cor/env.cor)
cor.ratio # so site-pair differnces are about 3/4 those of the environmental predictor correlations
```

```
## [1] 0.778917
```

This example shows that there is not a simple relationship between the level of correlation in predictor variables, and the level of correlation in the site-pair difference between predictor variables. The level of correlation is typically lower when considering site-pairs than sites.

3.6 Calculate AIC for a GDM

Calculating AIC () may be useful in refining a GDM, helping to select a parsimonious set of predictors and associated number of splines. This example creates a function to calculate AIC for a GDM, then demonstrates the use of the function.

```
# Establish the AIC function
AICFxn<-function(model){
  mod<-glm((1-model$observed)~model$ecological,
           family=binomial(link=log))
  k<-length(which(model$coefficients>0))+1  # Number of coefficients, plus 1 for the model intercept
  AIC<-(2*k)-(2*logLik(mod))
  dev<-((mod$null.deviance-mod$deviance)/mod$null.deviance)*100
  return(list(AIC,dev))
} # end AICFxn

# Use the AIC function on a gdm
# reads in example input data
load(system.file("./data/gdm.RData", package="gdm"))

# Prepare the biological data
sppTab <- gdmExpData[, c("species", "site", "Lat", "Long")]

# Prepare the predictor data
envTab <- gdmExpData[, c(2:ncol(gdmExpData))]

# Prepare the site-pair table
gdmTab <- formatsitepair(bioData=sppTab,
                         bioFormat=2,
                         XColumn="Long",
                         YColumn="Lat",
                         sppColumn="species",
                         siteColumn="site",
                         predData=envTab)

# Fit a GDM
gdm.1 <- gdm(data=gdmTab,
             geo=TRUE)
```

```r
# Get the AIC for the GDM
gdm.1.aic <- AICFxn(gdm.1)
gdm.1.aic[1]
```

```
## [[1]]
## 'log Lik.' 2428.358 (df=2)
```

4 References

Ferrier, S., Manion, G., Elith, J., & Richardson, K. (2007). Using generalized dissimilarity modelling to analyse and predict patterns of beta diversity in regional biodiversity assessment. Diversity and Distributions, 13, 252-264. doi:10.1111/j.1472-4642.2007.00341.x

Fitzpatrick, M. C., Sanders, N. J., Normand, S., Svenning, J.-C., Ferrier, S., Gove, A. D., & Dunn, R. R. (2013). Environmental and historical imprints on beta diversity: insights from variation in rates of species turnover along gradients. Proceedings of the Royal Society B: Biological Sciences, 280(1768). doi:10.1098/rspb.2013.1201

Jones, M. M., Ferrier, S., Condit, R., Manion, G., Aguilar, S., & Pérez, R. (2013). Strong congruence in tree and fern community turnover in response to soils and climate in central Panama. Journal of Ecology, 101(2), 506-516. doi:10.1111/1365-2745.12053