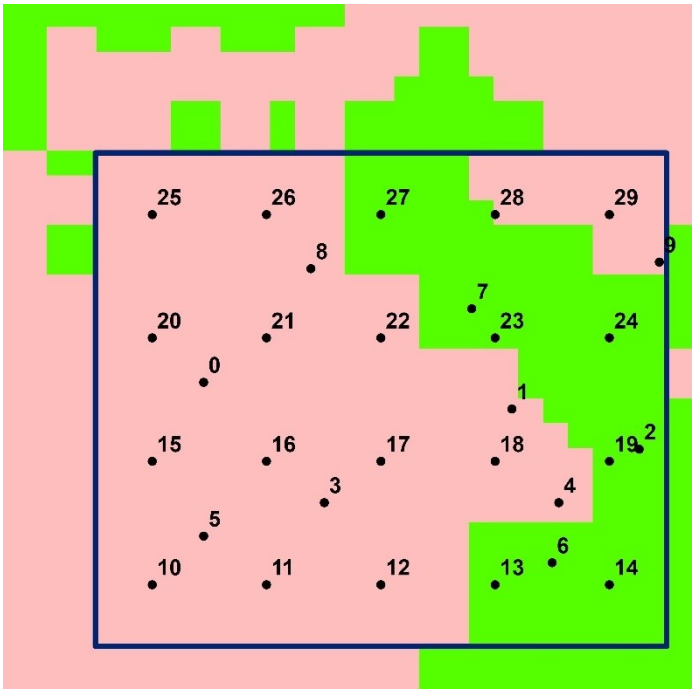
Optimal spatial and ecological sampling design tool: OSSA (Optimal Spatial Sampling Algorithm)



Developer: Luigi Sedda

(l.sedda@lancaster.ac.uk)

Contents

Introduction

OSSA is a geographic and statistical framework. In the same folder you will find OSSAtools.RData which contains a set of codes and data that you will need for masking land from sea (**landsea**), MODIS extraction code (**pextract**), area ecological classification (**lqdaloop**), spatial sampling design algorithm (**lcp**) and adaptive spatial sampling design algorithm (**asd**). These codes are written in R statistical language (https://cran.r-project.org), therefore you will need to install R in your machine before using the codes and datasets. In addition, OSSAtools.RData contains a table with mosquito collection from the same region (**benin**) and a grid for predictions in an extended area (**beningrid**).

All codes in **bold** (both black and red) can be copied and pasted in R console.

The algorithms are applied to an area of 50 by 50 km in South West Benin which encompasses the provinces of Athiémé, Bopa, Comè, Grand Popo, Houeyogbé, Kpomassè, Ouidah and Sè in the region of Atlantique.

Data download

Due to the right to attribution, environmental data files are not provided within this folder. These need to be download separately. Repositories are described in Box 1 below.

These are the files used in this example:

Global land 30: *n31_05_2010lc030.tif*

SRTM elevation: *N06E001.hgt, N06E002.hgt, N06E003.hgt, N07E001.hgt, N07E002.hgt, N07E003.hgt.*

FAO soil: *hwsd_soil1.tif*

BIOCLIM precipitation: *wc2.1_30s_bio_12.tif*

MODIS folders: *evi.zip, temp.zip and aetpet.zip*.

Box 1. Some open access data repositories.

GlobeLand30

(http://www.globallandcover.com/defaults_en.html?src=/Scripts/map/defaults/En/download_en.html&head=download&type=data) **.** Land cover mapping with 30 m resolution produced by the National Geomatics Center of China, and released for 2000, 2010 and 2020 contains 10 classes. The images utilized for GlobeLand30 classification are multispectral images with 30 meters, including the TM5 and ETM + of America Land Resources Satellite (Landsat) and the multispectral images of China Environmental Disaster Alleviation Satellite (HJ-1). GlobeLand30 data adopts WGS84 coordinate system, UTM projection, 6-degree zoning and the reference ellipsoid is WGS 84 ellipsoid.

MODIS (satellite)

MODIS Enhanced Vegetation Index (**EVI**) from the MOD13C2 product comprises monthly, global EVI at 0.05 degree resolution. This product provides consistent spatial and temporal comparisons of vegetation canopy greenness, a composite property of leaf area, chlorophyll and canopy structure. EVI minimizes canopy-soil variations and improves sensitivity over dense vegetation conditions when compared to NDVI. MODIS Air Temperature (**Temp**) from the MOD07_L2 Atmospheric Profile product comprises monthly, global temperature at 0.05 degree resolution and at the closest level to the surface. MODIS Evapotranspiration (**ET**) from the MOD16 Global Evapotranspiration product is calculated monthly at 0.05 degree as the ratio of Actual to Potential Evapotranspiration (AET/PET). All the MODIS products are provided in monthly time-series at 0.05 degree (~5km) resolution from observations by the MODIS sensor on Terra (AM) for the period February 2000-December 2013 inclusive (https://ora.ox.ac.uk/objects/uuid:896bf37f-a56b-4bc0-9595-8c9201161973).

Recent daily data can be downloaded from USGS (https://earthexplorer.usgs.gov) or https://lpdaac.usgs.gov/tools/data-pool/.

Precipitation

Can be obtained from WorldClim Version 2.1 (2020 release) as average annual precipitation from 1970 to 2000 at 30s (around 1km^2) (https://biogeo.ucdavis.edu/data/worldclim/v2.1/base/wc2.1_30s_bio.zip).

Elevation

The NASA Shuttle Radar Topography Mission (SRTM) Version 3.0 Global 1 arc second product is void-filled using elevation data from ASTER, GDEM2, GMTED and NED. The data are available for download via Earthdata Search (https://search.earthdata.nasa.gov/search).

Human density

Constrained total number of people per grid-cell is provided by WORLDPOP at a resolution of 3 arc (approximately 100m at the equator) with projection WGS84 Geographic Coordinate System. https://www.worldpop.org/geodata/listing?id=78

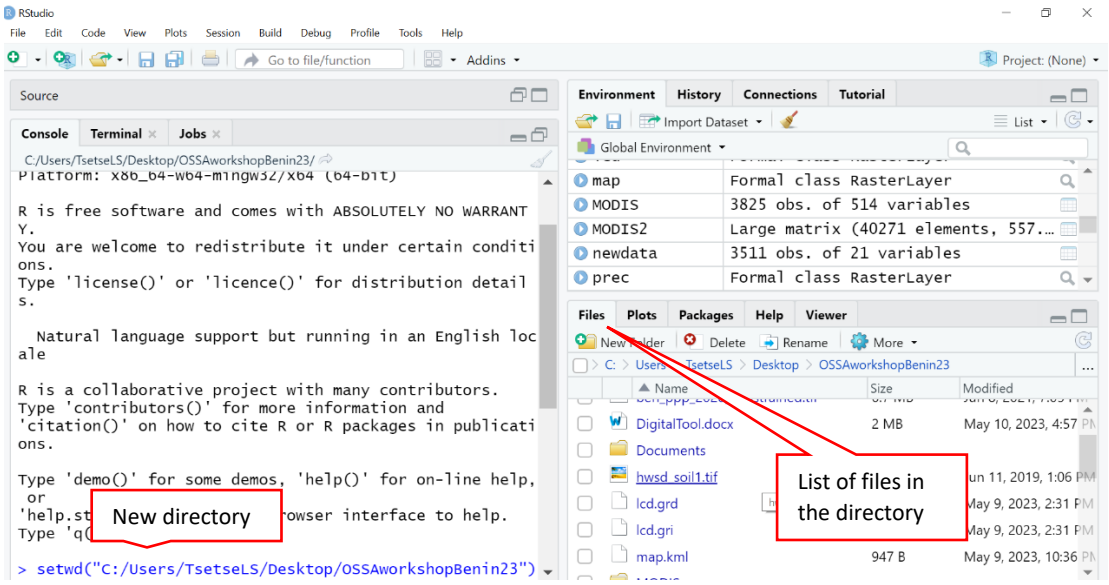
Soil data

The FAO Harmonized World Soil Database v 1.2 raster contains more than 16,000 soil units at approximately 1km resolution.

http://www.fao.org/soils-portal/data-hub/soil-maps-and-databases/harmonized-world-soil-database-v12/en/

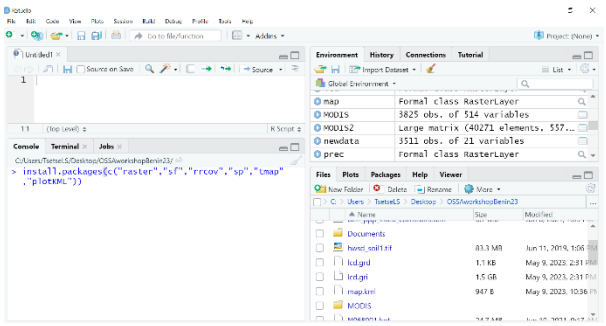
R preparation

Open RStudio. Set the working directory (the folder containing the data and codes): from the top bar of the main RStudio Session-> Set Working Directory-> Choose directory and then select *your folder*. If you want to set *your folder* as default, go to Tools-> Global Options and browse the default working directory to select it and press OK. You should be able to see the files contained in the folder in the Files tab in the bottom-right panel.



If have not done it yet, please install the R packages needed to run the OSSA functions from the lower left panel by typing (or copy and paste in R-studio console):

install.packages(c("raster","sf","rrcov","sp","tmap","plotKML"))



Alternatively they can be installed from the lower right panel and by clicking on 'Packages' and click on install and type raster, sf etc...

raster allows to work with raster or grid file format. A raster is a common data format for environmental variables (as those listed in Box 1 above). In simple word, a raster is just a matrix or table of values (elevation for example) with associated information about its geographical position and pixel (or cell) dimension. To import/work with rasters we will use the package **raster**.

To activate the packages in the workspace we need to upload them:

library(raster)

library(sf)

library(rrcov) #for luqdaloop

library(sp)

library(tmap)

library(plotKML)

or tick in the box next to **raster, sf etc…** in the list shown under the tab Packages in the lower right panel (this does the same as the function **library**).

Optimal Spatial Sampling Algorithm (OSSA)

The use of # after a command allow to comment without affecting the command itself (therefore can be included in your copy and paste).

OSSA: Step 1. Importing geographic rasters

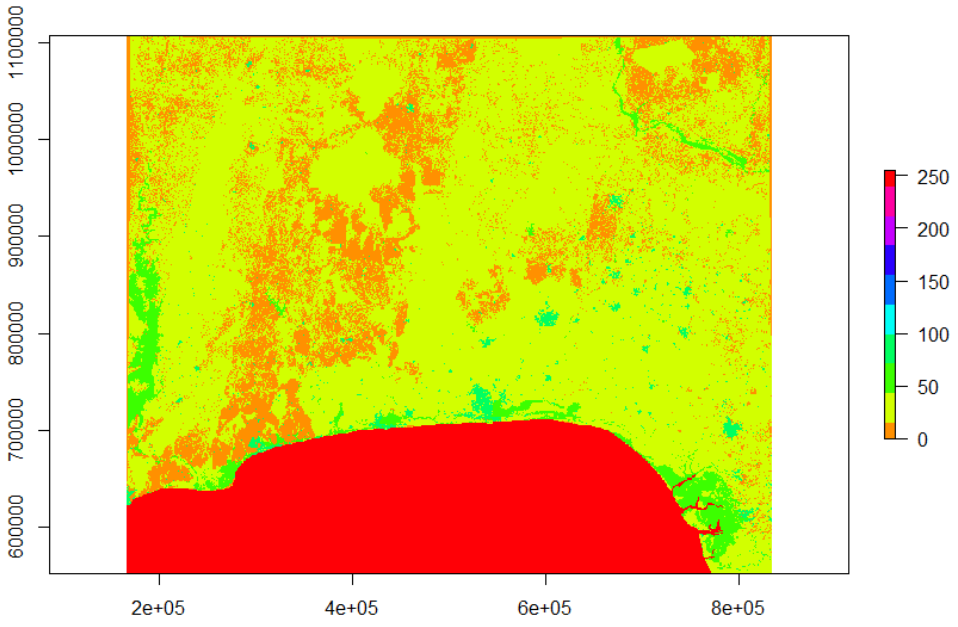
Let's start importing the land cover for the area of interest.

lc=raster("n31_05_2010lc030.tif")

lc #and press enter

This gives you useful information about the raster: resolution (30 by 30 meters), area covered in number of pixels (dimensions and extent), projection (crs), directory where the original file was (source, this is likely to be different in your computer) and range of the values (values). The raster is "connected" to the original source (in order to reduce the amount of memory in use), therefore if you move the folder where the raster is, the object **lc** will not work. You can visualize the imported land cover by using the function **plot()**.

plot(lc)



To get familiar with the **plot()** options, especially on how to customize the location of the legend see: https://biologyforfun.wordpress.com/2013/03/11/taking-control-of-the-legend-in-raster-in-r/

Here an example of customisation:

```
x=c(seq(10,100,10),255)
plot(lc,col=rainbow(255)[x],legend=FALSE)
plot(lc, legend.only=TRUE, col=rainbow(255)[x],
    legend.width=1, legend.shrink=0.75,
    axis.args=list(at=x,labels=x,cex.axis=0.6),
    legend.args=list(text='Land cover', side=4, font=2, line=2.5, cex=0.8))
```

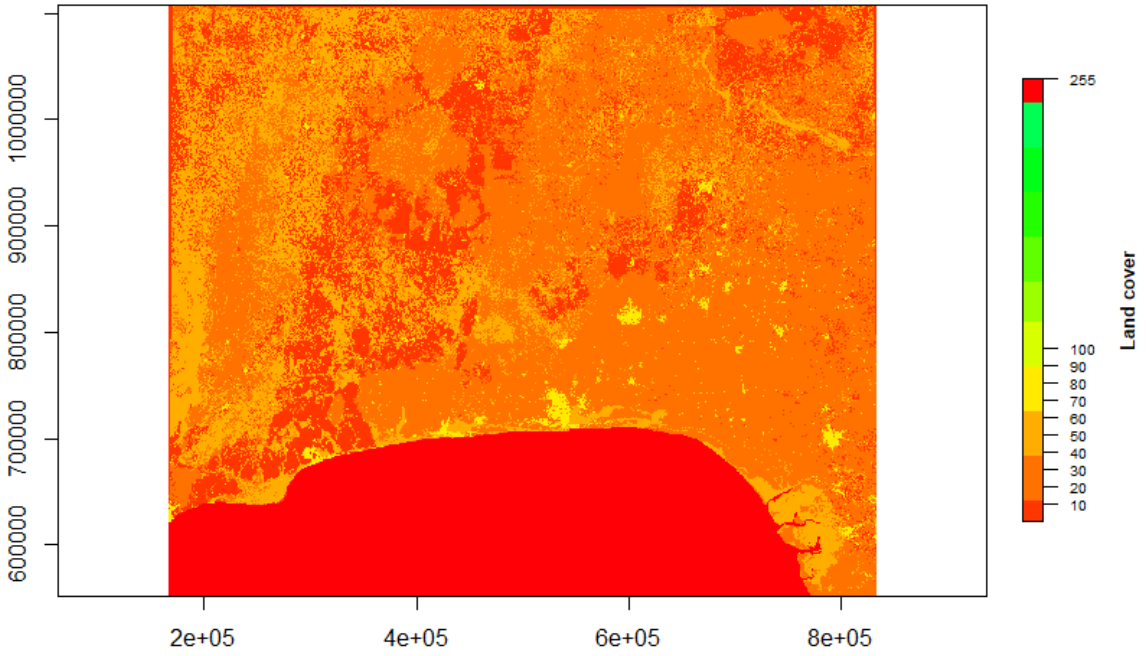


Table 1. Legend for land cover.

10	Cultivated land
20	Forest
30	Grassland
40	Shrubland
50	Wetland
60	Water bodies in the land area
70	Tundra
80	Artificial surfaces
90	Bareland
100	Permanent snow and ice
255	Sea

lc units are in meters. To convert the raster's units in degrees (since all the rest of the rasters are in degrees) use the function **projectRaster** from **sf** package.

Mind that this step can take up to two hours to complete (depending on the computer – but usually 15-30min)!

lcd=projectRaster(lc,crs="+init=epsg:4326",method="ngb",filename="lcd.grd")

You must define a filename (here we used lcd.grd, which will be saved in your working directory). More information on the function and in particular on the method "ngb" (used for categorical variables) can be found here: https://www.rdocumentation.org/packages/raster/versions/3.4-10/topics/projectRaster

Elevation is provided in multiple files (N06E001.hgt, N06E002.hgt, N06E003.hgt, N07E001.hgt, N07E002.hgt, N07E003.hgt). We will need to mosaic the files in order to work on a single raster instead of six. The format '.hgt' is recognised by the package **raster**. In order to speed up the conversion of the rasters and merge them we will first list them (incorporate multiple files in a single object). First create a vector of raster names:

f = list.files(pattern = ".hgt", full.names = TRUE)

This is looking at your directory and listing all files you have there with extension '.hgt'. Now import all the '.hgt' files in a single object (**elevation_list**)

elevation_list <- lapply(f, raster)

lapply simply apply the function raster to each source listed in the object '**f**' and create another list with all the raster imported.

Mosaic the six rasters in a single one:

elevation <- do.call(merge, elevation_list)

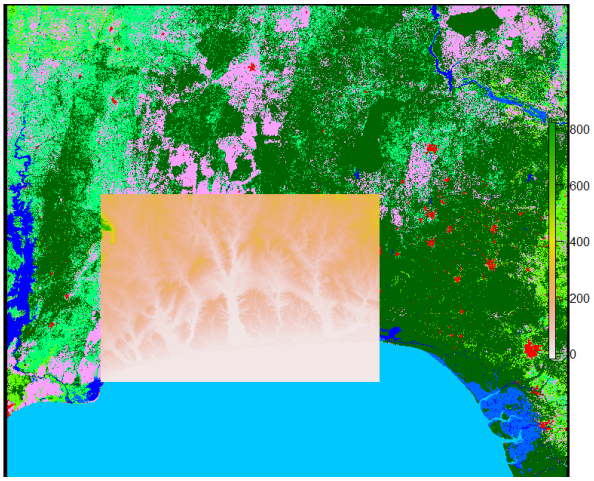
again, **do.call** allows you to work with list (**elevation_list**) to produce a single object (this is the reason why we don't use **lapply** again).

To evaluate the success of the operation, let's plot elevation over the land cover:

plot(lcd)

plot(elevation,add=TRUE)

Make sure you have 'plot' open in the lower right box of RStudio.



(this may look different in your computer)

For precipitation we will import the BIO12 bioclim raster (annual precipitation). However, there are other precipitation variables that can be considered: precipitation of wettest month (BIO13), precipitation of dries month (BIO14), precipitation seasonality (BIO15) etc… . If you want to import all the precipitation products you will need to get the other raster from the bioclim project (see Box 1), extract the raster in your folder and repeat the step below bearing in mind to change the name of the raster you upload and create.

prec=raster("wc2.1_30s_bio_12.tif")

for a quick evaluation that **prec** has the same projection of the others, we can use the function **st_crs** (from package **sf**, already uploaded):

st_crs(prec) == st_crs(elevation)

which will return '[1] TRUE', i.e. they have the same projection.

For the soil the procedure is identical to the one above:

soil=raster("hwsd_soil1.tif")

st_crs(soil) == st_crs(elevation) #as a check for projection

For the MODIS satellite data, we will extract three parameters: Temperature (Temp), Evapotranspiration (aetpet) and Enhanced Vegetation Index (EVI). They are provided monthly from 2000 to 2013 at a 5 km by 5 km for the entire globe. Therefore, we need to extract these values to a grid of the region we are

interested in. Once the values are extracted, we are going to produce the following variables: Temp mean, Temp range (amplitude), Temp standard deviation; EVI mean, EVI amplitude, EVI standard deviation, aetpet mean, aetpet amplitude, aetpet standard deviation.

Please be prepared, the computation will take 2 to 6 hours. In order to extract the MODIS data and convert in the variables described above, you will need the '**pextract**' function saved in the R file 'OSSAtools.Rdata'.

load("OSSAtools.RData")

args(pextract) #to see what is required to run the function

To run **pextract** we need 3 elements: **landsea**, **xyz** and the MODIS directories storying the monthly files (for each parameter: Temp, EVI and aetpet).

landsea, this object is already provided within the 'OSSAtools.RData' and allows to remove places that follow in the sea.

xyz is the matrix of coordinates from which we want to extract the MODIS data. We can create this easily, but first we need to define the area we want to survey. For the area of Ouidah, Kpomassè and Comè we can define a square with the upper/left corner coordinates as 7N/1.5E and lower/right corner are 6N/2.1E. Given these coordinates, to create a 70 by 70 grid (4900 nodes) we need:

Longitudes=seq(1.5,2.1,l=70)

Latitudes=seq(6,7,l=70)

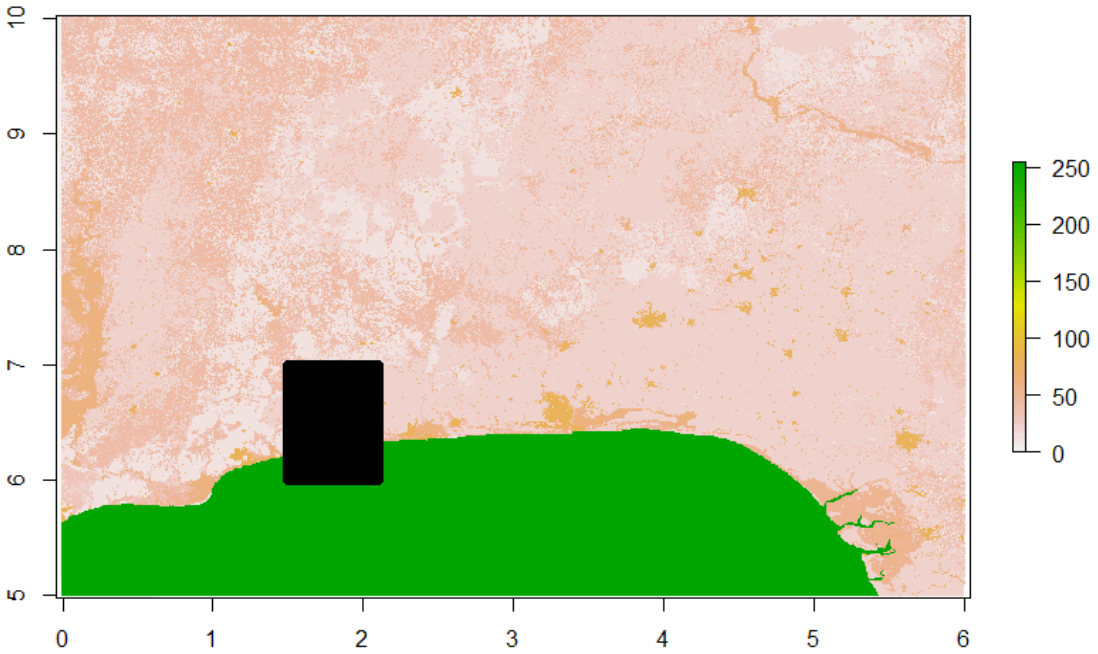
xyz=expand.grid(Longitudes,Latitudes)

seq create a set of number starting from the first argument (1.5 in Longitudes) to the second (2.1), with 70 breaks (therefore the range is broken in 70 intervals). **expand.grid** create all the possible combinations between **Longitudes** and **Latitudes**. To see the area covered by the points:

plot(lcd,useRaster=FALSE)#useRaster=False allows a better zoom of the area

points(xyz[,1],xyz[,2])

With xyz[,1] you are selecting the first (longitude) column of xyz, and with xyz[,2] the second (latitude) column of xyz.



Black square shows the area interested by the grid.

Finally, please keep the three directories inside a folder called MODIS. Each of them should have 167 files. In this way you can run the following (substitute 'your directory' with the path of your computer where the MODIS folder is – also notice that temperature MODIS files must be stored in temp folder, evapotraspiration in aetpet folder and enhanced vegetation index in the evi folder, if you keep the same names as in the code below).

MODIS=pextract(landsea,xyz,dire=c("C:*yourdirectory*\\MODIS\\temp","C:\\ *yourdirectory*\\MODIS\\aetpet","C:\\ *yourdirectory*\\MODIS\\ evi"),stat="T",varn=c("Temp","ET","EVI"))

MODIS is a table with 3825 rows (representing a grid of the area) and 514 columns, e.g. monthly averaged of the 3 parameters from February 2000 to December 2013 (167 values for each parameter) plus the mean, amplitude and standard deviation for the entire time series (one value each, for each parameter).

Instead of using all the 514 columns, we will use the total mean, amplitude and standard deviation for each parameter (temp, evi and aet), including the longitude and latitude (columns 1 and 2):

MODIS2=MODIS[,c(1:2,172:174,342:344,512:514)]

To look at the first 6 rows: **head(MODIS2).** MODIS and MODIS2 contain some NA values or infinite. To remove the rows with NA:

MODIS2=na.omit(MODIS2)

xyz=MODIS2[,1:2] #this is needed for later

Let's check if we have any infinite value.

MODIS2=as.matrix(MODIS2) #this helps all the next calculations

length(which(is.finite(MODIS2)==FALSE))

As you can see the length is 0, we don't have infinite values and neither NAs.

Finally combine the MODIS data with land cover, elevation, precipitation and soil information in a new table called **data**:

data=cbind(MODIS2,extract(lcd,xyz),extract(elevation, xyz),extract(prec, xyz),extract(soil, xyz))

colnames(data)=c("x","y","temM","temV","temA","etM","etV","etA","eviM","eviV","eviA","LC","Elev","Prec","Soil")

data=na.omit(data) #for the NAs in lcd, elvation, prec and soil

data=as.data.frame(data)

We have removed other NAs using again na.omit. Now **data** table is ready to be used for step 2 of OSSA (classification).

Box 2. Discriminant analysis.

The theory on discriminant analysis, a multivariate method for clustering, is based on the multivariate normal distribution, which parameters are the mean (of the environmental variables at the class-point locations) and the variance-covariance matrix (of the environmental variables at the class-point locations).

When the variance-covariance matrices are not homogeneous (essentially, are not the same) for two or more classes, linear discriminant analysis cannot be used (just to remind you that linear discriminant analysis assumes that all the classes have the same variance-covariance matrix but not the mean). Instead a non-linear (quadratic) Discriminant Analysis (QDA) can be applied, with discriminant function as:

$$f_i^Q = -\frac{1}{2}\log|\mathbf{\Sigma}_i| - \frac{1}{2}\left((\mathbf{X}-\mu_i)^T\mathbf{\Sigma}_i^{-1}(\mathbf{X}-\mu_i)\right) + \log(p_i)$$

Where **X** is the matrix of variables, μ the vector containing the mean of each variable and **Σ** is the variance-covariance matrix, and *i* is the class. The larger is the *f* value, the higher is the probability that the point belong to the *i* group.

As you can notice, apart from the last term ($\log(p_i)$), the rest is exactly the equation of a multivariate normal distribution. The p_i (the "prior" probability of each point to belong to the class *i*) can be calculated in a number of ways, usually by "equal priors" method: each class has a prior probability equal to 1/number of classes. In this analysis, and in order to take into account the spatial proximity of the classes, a local frequency prior methods was used. It estimates the class p_i prior probability as the relative frequency of points belonging to *i* class in the (pre-defined) neighbourhood.

For classification, a new data point can be assigned to the class at which the f_i is at a maximum. By calculating the position of a point with respect to all available class centroids, it is possible to calculate the probability of membership to each class, or in other words, a point is allocated to the class for which f^Q is maximum.

The ecological classification is based on a locally (or spatially) adjusted quadratic discriminant analysis (Box 2). Before running this algorithm, we need three preliminary steps.

The first is to define the training classes. Assuming that land cover (in our data is the column 'LC') is the strongest local factor, then we can decide that land cover defines (or drives) the ecological classes. For simplicity, at the beginning we can consider the number of ecological classes equal to the number of land cover classes in the area. Given land cover = class, the second step is to check

the size of each class. Classes with a small amount of points can produce errors (technically a singular matrix). As a role of tomb, we expect to have a number of records for each class at least 5-10 times more than the number of predictors. To see how many records we have for each class simply use the function **table()**:

table(data$LC) #Note the use of $ to select the column we want to use

In our data there are 7 land use cover classes (see Table 1 above and this link for additional explanations: http://www.globallandcover.com/Page/EN_sysFrame/dataIntroduce.html?columnID=81&head=product&para=product&type=data).

Three of them show a small number of records: 30 (Grassland), 50 (Wetland), 60 (Water bodies). To simplify, we will merge 30 (Grassland) with 40 (Shrubland) and 50 with 60. To do this we create a new column LC2 identical to LC:

data$LC2=data$LC

and replace LC 30 and LC 50 with 40 and 60 respectively:

data$LC2[data$LC==30]=40 #replacing 30 with 40

data$LC2[data$LC==50]=60 #replacing 50 with 60

Now LC2 contains 5 classes instead of 7:

table(data$LC2)

The third step (although not necessary) is to remove variables with low correlation with land cover (LC2), i.e. lower than |0.01| (absolute value of 0.01).

If you want to proceed with this, you can look at the correlations by using the function **cor()**. We don't need correlation between all the variables, therefore we will only focus on the correlations with LC2:

cor(data)["LC2",] #in this way it returns only theLC2 correlations

We remove temA (-0.005), eviM (0.002), and Elev (-0.003):

datared=data[,-c(9,12,13)] #removing the fifth, ninth, … columns

If you use datared, make sure to substitute datared with data in the luqdaloop command below.

Finally, the data is ready to go through the local quadratic discriminant analysis. The function to use is **luqdaloop** (already available since contained in OSSAtools.RData that you have previously loaded) that calculates the quadratic discriminant analysis for the given dataset, plus collapse or merge the classes to a given fixed maximum number of classes, and perform cross validation.

classanalysis=luqdaloop(X=data[,c(3:11,13:15)],y=data$LC2,grid=data[,1:2],n=0.1,nx=7)

X is the matrix of variables used to classify the ecology of the area (excluding x – the longitude - and y – the latitude - because they are considered in the grid; and LC – column 12 - because we are using LC2), y is the initial or prior classes (here assumed coincident with land cover), grid is the two columns matrix with coordinates of each record. To calculate the local frequencies (prior probabilities) we used a circular neighbourhood of 0.1 degrees (almost 12 km), quite a large one (18 points in each direction). Finally, nx is the maximum number of classes allowed (from splitting existing ones).

luqdaloop selects the best classification as the one with a relative minimisation of the Wilks' lambda value, equivalent to a minimisation of the within class variance. We talk about 'relative minimisation' because the algorithm search to the largest decrease in the Wilks' lambda value from a classification with n classes to a classification with n+1 classes.

The result **classanalysis** is a list containing the following values (you can see the list of objected contained in **classanalysis** by typing **names(classanalysis)** and press enter):

1. **WilksSummary**, a table showing the Wilks value for each number of clusters (first row), the prediction error (second row), the Wilks value for the test data (third row), and the prediction error for the test data (forth row). If no number of records was defined for the argument test in **luqdaloop** the last two rows will not show any values.
2. **Cluster classifications**. For nx clusters, **classanalysis** will show 2cluster, 3cluster… nx cluster. Each of this element is itself a list containing 11 elements (see below).
3. **ExcludedData**. For some classes it is not possible to evaluate the discriminant value (often because the predictors tend to not vary much within this class) and therefore they are excluded or simply considered as a cluster that cannot be split or merged. This value is present only if data is excluded.
4. **NewData**. This is the same data in input with the priors and new classification. This is what will be used at the next OSSA step.

Each cluster classification contains:

1. **WMqr**, a list of pooled variance/covariance matrix for each class.
2. **GM**, a list of predictors means for each class.
3. **ldet**, a numeric vector containing the determinants for each class.
4. **prior**, a matrix where for each record (row) the probabilities to belong to the different classes (columns) are provided.
5. **scores**, a numeric vector of discriminant values.
6. **classification**, a vector of new classes attributed at each record.
7. **confusion**, confusion matrix.
8. **error_rate**, probability to not belong to the cluster attributed in classification.
9. **Wlambda**, Wilks' lambda statistic for the classification.
10. **Nclasses**, classes and classes size of the classification.
11. **Classes**, a numeric vector with original classes.

To print the Wilks' summary, simply:

classanalysis$WilksSummary

or

classanalysis[[1]] #show the first element of the list

To print the confusion matrix from classification with 4 classes:

classanalysis[[4]][[7]]

4 is the slot for 4 classes classification and within it 7 is the slot containing the confusion matrix.

When **luqdaloop** finishes it prints the number of optimal classes found. You can also retrieve it by first extracting the new data and applying **table** to the new classification:

newdata=classanalysis$NewData

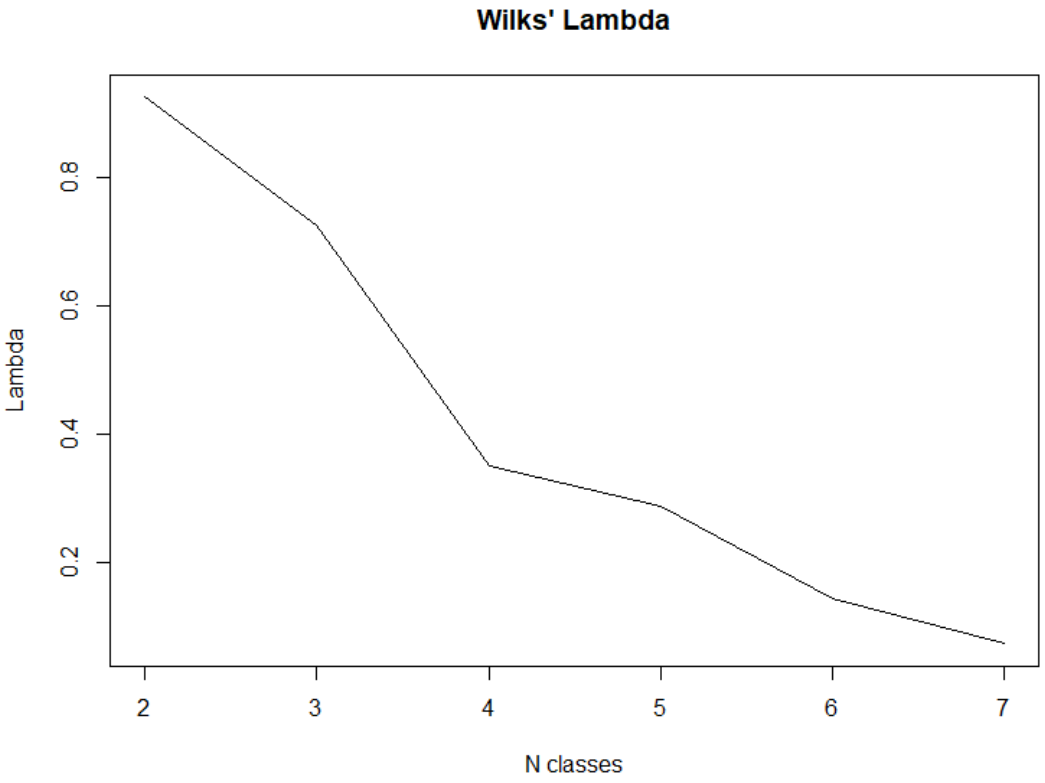
table(newdata$BestClass)

It is clear that the algorithm found 4 best classes in which it merged class 60 with 80 (water bodies with artificial surfaces). This is not surprising since large urbanisations are close to wetlands and water bodies.

IMPORTANT: some of the original records may be re-classified. For example, compare the results from **table(newdata$BestClass)** with **table(data$LC2)**, you will notice a decrease/increase in some land cover. This is because once, luqdaloop calculates the probability for each record to be in a specific cluster, it will assign the cluster associated with the largest probability to the record.

Now if you want to plot the Wilk's over the number of classes:

plot(2:7,classanalysis[[1]][1,2:7],main="Wilks' Lambda",ylab="Lambda",xlab="N classes",type="l")



7 was the number of classes explored (see **luqdaloop** command above).

From the plot above the largest difference (or drop) in the lambda is when passing from 3 to 4 classes. Therefore 4 classes are the best classification.

Finally, let's plot the new classification:

map=newdata[,c("x","y","BestClass")] #creating an object with x,y and class

coordinates(map)=~x+y # defining the coordinates' columns

gridded(map)=TRUE

proj4string(map)=(CRS("+init=epsg:4326")) #defining the projection

map=raster(map) #converting in raster

tm_shape(map)+tm_raster() #plot

or to plot onto a map:

tmap_leaflet(tm_shape(map)+tm_raster())

To plot on google earth (make sure you have launched google earth first):

plotKML(map)

If not lunched, go in your working directory and double click on the file 'map.kml'

A useful tutorial on plotKML can be found at the following URL:

http://gsif.isric.org/doku.php/wiki:tutorial_plotkml

OSSA: Step 3. Lattice with close pairs design for the spatial allocation of sampling sites

We will use the object **map** created in the previous step as the area where to optimise the sampling design based on the ecological classes identified above. See the properties of this raster by typing map and pressing enter:

```
> map
class       : RasterLayer
dimensions  : 16, 13, 208  (nrow, ncol, ncell)
resolution  : 0.05, 0.05  (x, y)
extent      : 1.475, 2.125, 6.225, 7.025  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
source      : memory
names       : BestClass
values      : 1, 4  (min, max)
attributes :
 ID  levels
  1      10
  2      20
  3      40
  4 MC60.80
```

We have already described these properties in step 1. As shown in attributes, classes are converted in numeric values. In fact, the best classes (10, 20, 40, MC60.80 called 'levels') are converted into classes 1, 2, 3 and 4 respectively (see 'ID'). This is important for interpreting maps and running the spatial sampling design algorithm.

The allocation of the sampling locations in each of the classes of **map** area is through a non-adaptive sampling design: sampling locations are chosen in advance of any data collection.

Our optimisation of the spatial sampling design follows the theory of the mixture design known as lattice with close pairs which itself can be considered as an extension of an inhibitory designs adapted for inclusion of random close pairs (Box 3). In fact, this design combines random close pairs design (good for parameter estimation – i.e. reduces model uncertainty) with lattice design (good for estimation when parameters are unknown – i.e. increases accuracy in predictions).

Box 3. Spatial sampling designs.

If sampling sites are preferentially chosen to capture larger (or smaller) than average values of a response, e.g., areas with abundant mosquitoes or with higher malaria transmission, then subsequent estimation and prediction of the exposure surface using standard geostatistical methods may be misleading due to the selective sampling. Practical needs or deliberate actions often lead to preferential sampling.

A combination of theoretical and empirical work has led to general acceptance that lattice designs should lead to efficient spatial prediction provided model parameters are known. If model parameters are unknown, a completely random design has the advantage that it will include a wider range of inter-point distances, and in particular some small inter-point distances, and so provides more information on the shape of the spatial covariance function. However, the resulting uneven spatial distribution makes prediction less efficient, given the model parameters, and leaves open the possibility of systematic bias. In inhibitory design parameters are unknown. Points are chosen at random but with the constrain that the distance between points are not less than a certain value. We included the conditions that random points are close pairs of a random selection of nodes from a sampling grid. The term 'close pairs' here is used loosely, since not all the points in the grid will have a close pair, and some close pairs may be shared between points.

Other randomised designs include: complete randomised design stratified or clustered; completely regular lattice design (square, equilateral, triangular or hexagonal); random-tessellation stratified design which guarantees that no point from the target population is too far from a sampling point; and that few points are very close together.

The function '**lcp**'(lattice with close pairs - already available since contained in **OSSAtools.RData** that you have previously loaded) distributes M (G+R) sampling points under three conditions: (i) G % of sampling points are in grid (lattice), (ii) R % are close-pairs randomly allocated; and (iii) each stratum (or class) must contain a number of points proportional to the stratum size.

To run **lcp** we need the **map** object created above, e.g. a raster with classes, and the following set of parameters:

delta inhibition distance or minimum distance between any two locations.

zeta radius around each grid location where to allocate the close pair(s)

total total number of locations to be optimised

grid proportion of locations in grid, the rest (1-grid) will be allocated as close pairs.

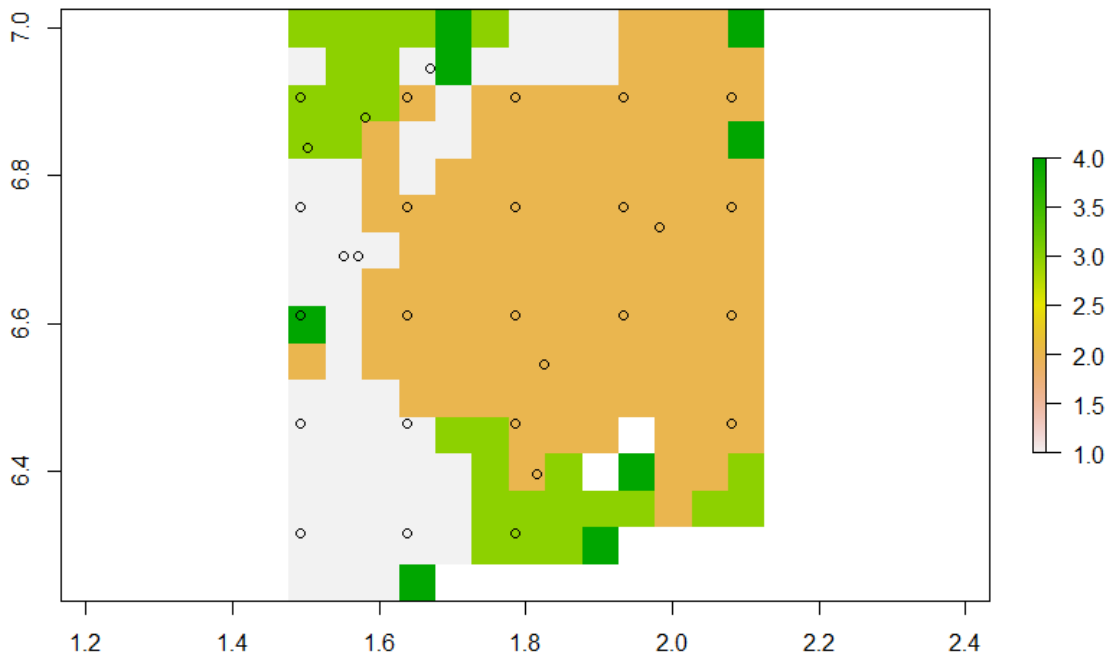
delta and **zeta** can be chosen based on the spatial process (e.g. **zeta** equal to the spatial range of the process under study, e.g. dispersion of a mosquito) while **delta** can be based on the desired spatial coverage (a larger **delta** allows to increase the probability to have sparse close pairs within the grid, and improve homogenous coverage of the area). For this example we have chosen a **delta** of 1km (0.01 degrees), **zeta** of 0.16 degrees, **total** of 30 locations and **grid** =0.7 (equivalent to 70% - 20 locations allocated in grid).

sites=lcp(map,delta=0.01,zeta=0.16,total=30,grid=0.7)

The optimised locations are shown in the figure below. The object **sites** is a matrix with coordinates, class and type columns. 'type' is coded 'G' if the location is allocated in the grid or 'I' if it is allocated as close pair.

plot(map)

points(sites[,1],sites[,2])



OSSA: Step 4. Adaptive sampling design

The 3 first steps of OSSA were designed to optimise the spatial allocation of survey locations based on the ecology and the spatial characteristics of the process under study (malaria distribution, mosquito presence/absence etc…). The assumption is that the process is dependent on the ecology of the area under study and its variation is spatially dependent. We have not used any available data that can describe the process since we assumed operating in absence of any prior information.

But when previous data is available, we may want to conditionate the new sampling to existing information in what is called an 'Adaptive sampling design': sampling locations are chosen sequentially (singly or in batches) during the data collection or during the design of the sampling campaign based on historical data. The adaptive approach is preferential since guided by a predictive target (e.g. where the prevalence is above a certain threshold or for error minimisation). These sampling designs are becoming more important in poor resource settings where uniformly precise mapping may be unrealistically costly and the priority is often to identify critical areas where interventions can have the most health impact. Two constructions are often applied: singleton and batch adaptive sampling. In most settings, batch sampling is more convenient than singleton sampling (one new location is added at each sampling iteration).

Here we provide a simplified adaptive sampling design function (**asd**) with the choice of two targets: 'U' minimise uncertainty and 'H' maximising accuracy in hotspot (large value of the process – probability of presence of mosquitoes for example) definition.

In the following example **asd** is applied to previous Benin mosquito collection carried in the same area in 2018. The table is called **benin** and contains the longitude (x), latitude (y), number of *Anopheles gambiae* collected overnight (AnGam), the week of collection (Week), the village (Village), land cover class (LCD), elevation (Elev) and soil type (Soil). The adaptive sampling will be targeting the hotspot of *An. gambiae* – e.g. where the mosquito is more prevalent with less certainty (target 'H' as described in the paragraph above). In order to identify the adaptive locations we need to consider the continuous area around the **benin** locations. This area is described by the **beningrid** table

which contains the same elements of **benin** that will be used in the model apart from the outcome (AnGam).

benin, **beningrid** and **asd** are already available since contained in OSSAtools.Rdata that you have previously loaded.

The arguments in the **asd** function are:

Data	the data to be modelled (including outcome, predictors and random effects);
area	the area to be considered for sampling; must include the same predictors and random effects as in Data;
formulaf	formula of the model for the fixed effects;
formular	formula of the model for the random effects;
total	batch of adaptive sampling locations to be allocated
delta	inhibition distance or minimum distance between any two locations.

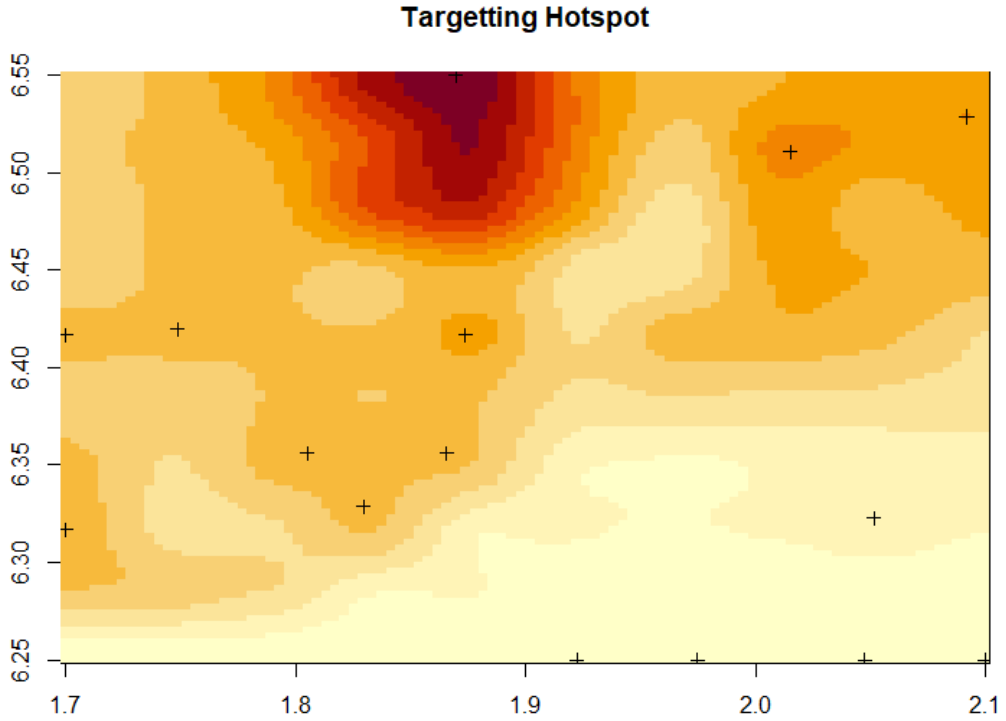
The modelling is done via glmmPQL (http://127.0.0.1:25126/library/MASS/html/glmmPQL.html) that allows for random effects to control for 'grouped variability'; and for explicit definition of the correlation function (exponential).

Let's assume we want to allocate 15 trapping locations at a minimum distance of 0.01 degrees (max undefined), then we simply run the following function:

adaptiveSites=asd(Data=benin[,-5], area=beningrid, formulaf = as.formula("AnGam~Week+Elev+Soil"), formular = as.formula("~1|LCD"), target="H", total=15, delta=0.01)

Note **benin[,-5]** it means that we are not using the fifth column (Village) in **benin** table. The used predictors are Week, Elevation and Soil, while the grouping variable is the land cover.

The visual results are shown in the figure below (the crosses are the adaptive locations).



To view the adaptive location information saved by the **asd** function, simply digit **adaptiveSites** and press enter:

```
> adaptiveSites
            x        y         Fit Uncertainty
9943 1.869697 6.550000 6.284772955   1.0400181
8679 2.015152 6.510606 3.314937844   0.6551063
9298 2.091919 6.528788 3.121836797   0.5982997
5544 1.873737 6.416667 2.877953801   0.4234016
2201 1.700000 6.316667 2.416944586   0.3982157
5501 1.700000 6.416667 2.311557624   0.3739736
5613 1.748485 6.419697 2.295409399   0.4093024
3527 1.805051 6.356061 2.290641413   0.4239954
2633 1.829293 6.328788 2.244973376   0.4219271
3542 1.865657 6.356061 2.206260474   0.4275912
2488 2.051515 6.322727 0.906078091   0.3178085
69   1.974747 6.250000 0.007630050   0.4114378
87   2.047475 6.250000 0.006950127   0.4110779
56   1.922222 6.250000 0.005254357   0.4115585
100  2.100000 6.250000 0.005254284   0.4115585
```

'Fit' are the values predicted by the glmmPQL model and 'Uncertainty' is the standard errors around these predictions.

Acknowledgments and contact

This work is funded by the Academy of Medical Sciences GCRF Networking Grant Scheme (GCRFNGR7/1329).

A preliminary version of the same algorithm (without adaptive component) was developed during the Medical Research Council grant GAARDIAN (grant no. MR/P02520X/1) which was part of the EDCTP2 programme supported by the European Union.

We thanks all the teams that used the preliminary version of OSSA (Eric R. Lucas, Luc S. Djogbénou, Ako V. C. Edi, Alexander Egyir-Yawson, Bilali I. Kabula, Janet Midega, Eric Ochomo, David Weetman, Martin J. Donnelly, Daniel McDermott, Joshua Longbottom and El Hadji Amadou Niang) for sampling campaigns in Benin, Democratic Republic for Congo, Ghana, Ivory Coast, Kenya, Malawi, Tanzania, Senegal and UK.

We are planning to improve this stand-alone tool for optimal spatial sampling design. It is essential for us to receive your feedback on these tools and manual. Please send your feedback to:

Dr Luigi Sedda l.sedda@lancaster.ac.uk

We hope you have enjoyed OSSA!