Soil Sampling Design

Technical Manual

Rodríguez Lado, L., Angelini, M.E, Naypewe, N., Luotto, I., Yigini, Y.

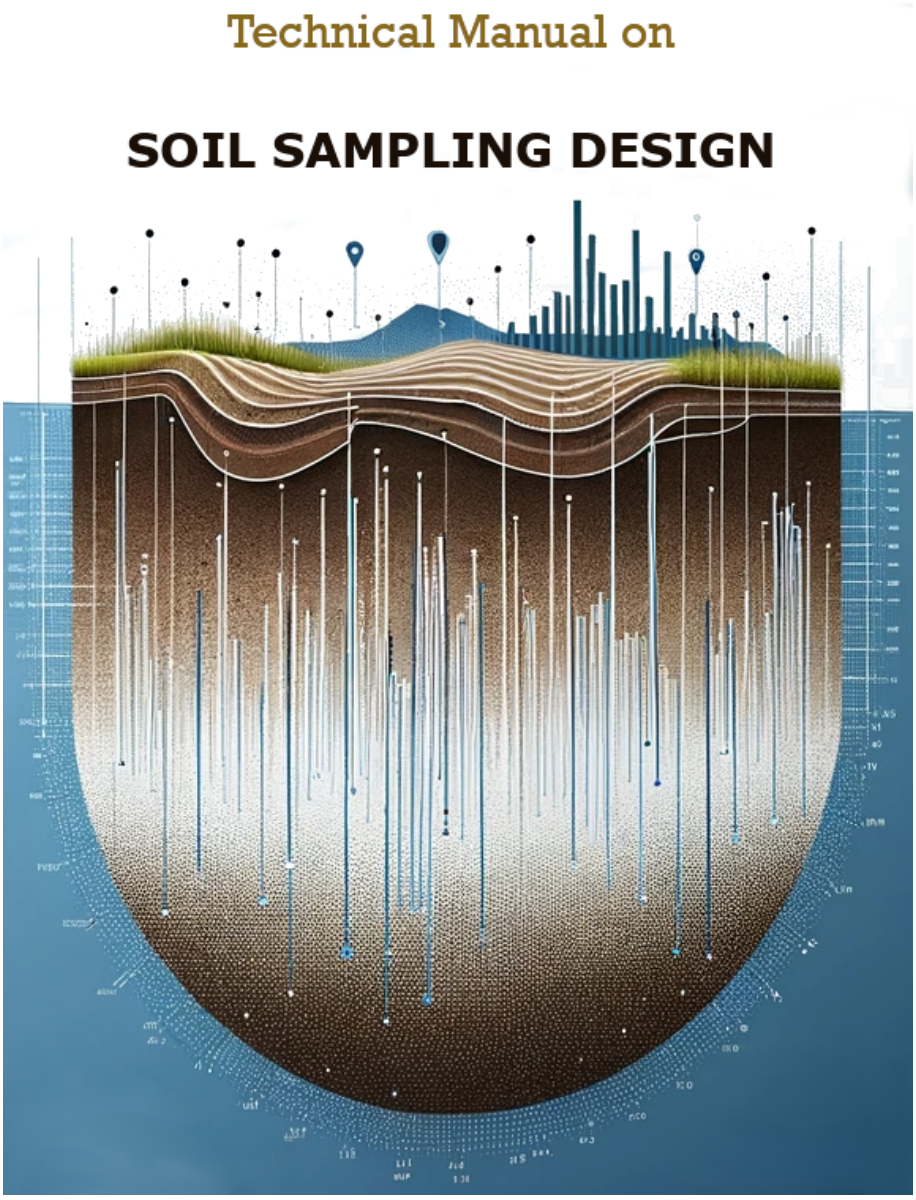
2023-12-22

Contents

Contents

List of Figures

List of Tables


Technical Manual on
SOIL SAMPLING DESIGN

Licence

Abbreviations and acronyms

BD Bulk density

CEC Cation exchange capacity

CRAN Comprehensive R archive network

Contributors and reviewers

International Network of Soil Information Institutions

GSP - Soil Information and Data

Introduction

The success of soil mapping activities relies on the existence of proper data collated through detailed soil sampling protocols ensuring representative and reliable soil data collection. This publication does not intend to be an intensive compilation which cover all the intricate possibilities in soil sampling design. In turn, we show a number of different soil sampling protocols through examples, covering both the most common approaches that can be used for the design of field soil sampling and present methodologies to evaluate their accuracy and effectiveness. The last part of the manual is particularly focused in the use of conditional Latin Hypercube Sampling (cLHS), a statistical methodology developed specifically for soil sampling from a Digital Soil Mapping perspective (Minasny and McBratney, 2006).

The manual is structured in two parts. `'Part One'` presents a methodology to evaluate the capacity of an existing soil legacy data to represent the potential soil diversity within a certain study area and determine whether it is a valid set for Digital Soil Mapping purposes. We use the Kullback-Leibler divergence (KL) measurement to quantify the difference between the probability distributions of covariate values in the legacy samples set and in the whole area and determine how much information is lost when the sample set is used to approximate the diversity in the existing environmental conditions in the whole area.

In `'Part Two'` we present several methods for creating soil sampling designs. We start with the determination of the optimal sample size for describing most of the environmental diversity in the area to the creation of sampling designs. We present examples of various sampling methods, ranging from traditional grid-based approaches to advanced statistical sampling strategies. We include methods for systematic, random and stratified sampling, evaluating their strengths and weaknesses in the context of DSM.

0.1 Training material

The manual exercises are written in the statistical environment {R} and running in the integrated development environment (IDE) `RStudio` for simplicity. Some scripts are based on the work and scripts from (Malone, Minansy and Brungard, 2019), which can be found at their repository. The raster data in some examples comes also from the github repository from Dave White (USDA-NRCS).

The train material of this book is located in the Sampling-Design-TM GitHub repository. To download the input files and **R** scripts, clone the repository or click on this link, save the ZIP file and extract its content in a folder, preferable close to the root of your system, such as `"C:/GIT/"`.

We have used a common structure for file paths in the exercises. By default, the `'RStudio'` console points to the folder where the active file is located (defined by `setwd(dirname(rstudioapi::getActiveDocumentContext()$path))` in the code). With this structure, R scripts appear in the root of the working directory and data files are in a `'data/'` directory within the root, with **.shp** and **.tif** files located within the sub-folders `'data/shapes'` and `'data/rasters'` respectively. Following this recommendation simplifies the definition of paths and execution of the scripts. If users desire to change their storage paths, they have to properly adjust data paths in the R scripts.

Part one – Soil Legacy Data

Evaluating Soil Legacy Data Sampling for DSM

Modelling techniques in Digital Soil Mapping involve the use of sampling point soil data, with its associated soil properties database, and a number of environmental covariates that will be used to ascertain the relationships of soil properties and the environment to then generalize the findings to locations where no samples have been compiled.

In soil sampling design, a crucial issue is to determine both the locations and the number of the samples to be compiled. In an optimal situation, soil sample database should adequately cover all the environmental diversity space in the study area with a frequency relative to the extent of the diversity in the environmental covariates.

When dealing with legacy soil data, a question that arises is if the data is representative of the environmental diversity within the study area. In this Chapter we present a method to answer this question and to build an alternative how many samples can be retrieved to cover the same environmental space as the existing soil data. The method follows the main findings in (Malone, Minansy and Brungard, 2019) and developed as {R} scripts.

We adapted the original scripts to make use of vector `'.shp'` and raster `'.tif'` files, as these are data formats commonly used by GIS analysts and in which both soil and environmental data is often stored. We also made some changes in order to simplify the number of R packages and to avoid the use of deprecated packages as it appears in the original code.

0.2 Data Preparation

We must load the required packages and data for the analyses. We make use of the packages `sp` and `terra` to manipulate spatial data, `clhs` for Conditioned Latin Hypercube Sampling, `entropy` to compute Kullback–Leibler (KL) divergence indexes, `tripack` for Delaunay triangulation and `manipulate` for interactive plotting within RStudio. Ensure that all these packages are installed in your system before the execution of the script.

We define the working directory to the directory in which the actual file is located and load the soil legacy sampling points and the environmental rasters from the `data` folder. To avoid the definition of each environmental covariate, we first retrieve all files with the `.tif` extension and then create a `SpatRaster` object with all of them in a row.

```
## Set working directory to source file location
  setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
```

Here we define a number of variables that will be used during the exercises in this manual. They include the path to raster and shp files, aggregation and disagregation factors, and buffer distances to define potential sampling areas from sampling points. These variables are later described at the appropriate section in the manual.

```
## Load soil legacy point data and environmental covariates

# Load soil data
  p.dat <- terra::vect(file.path(paste0(shp.path,"/legacy_soils.shp")))
# Load raster covariate data
  # Read Spatial data covariates as rasters with terra
  cov.dat <-  list.files(raster.path, pattern = "tif$",  recursive = TRUE, full
  cov.dat <- terra::rast(cov.dat)

  # Aggregate to simplify data rasters for calculations (optional)
  cov.dat <- aggregate(cov.dat, fact=agg.factor, fun="mean") # Now you can see
```

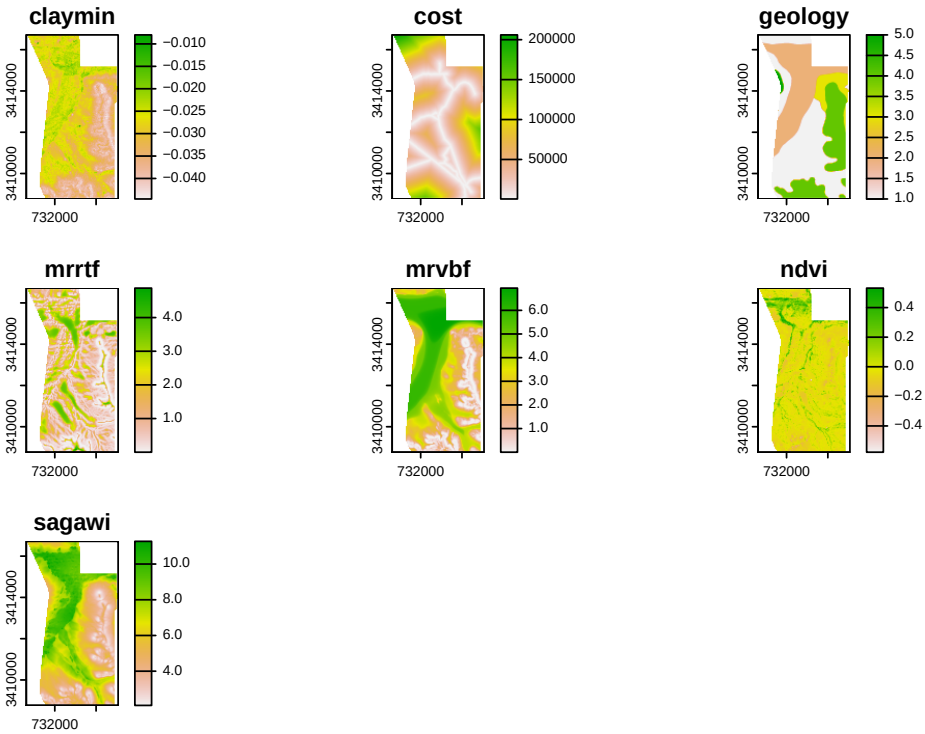


Figure 1: Covariates

0.3 Representativeness of the Legacy Soil Data

The next step involves the determination of the distributions of environmental values in the soil samples data and its comparison with the existing distributions of each environmental variable to determine the representativeness of the soil points in the environmental space.

The comparison of distributions is performed through the Kullback–Leibler divergence (KL) distance, a measure used to quantify the difference between two probability distributions. KL–divergence compares an 'objective' or reference probability distribution (here, the distribution of covariates in the complete covariate space – P) with a 'model' or approximate probability distribution (the space of covariates in the soil samples – Q). The main idea is to determine how much information is lost when Q is used to approximate P. In other words, KL–divergence measures how much the Q distribution deviates from the P distribution. KL–divergence equals approaches to 0 as the two distributions have identical quantities of information.

We cross soil and environmental data to create a dataset with the values of the environmental parameters at the locations of the soil samples.

We first calculate a `'n-matrix'` with the values of the covariates dividing their distributions into `'n'` equally–spaced bins. Each bin captures the environmental variability within its interval in the total distribution. In this exercise, `'n'` equals to 25. The result is a 26×4 matrix, where the rows represent the upper and lower limit of the bin and (thus, 26 rows are required to represent 25 bins), and 4 correspond to the number of variables used as environmental proxies.

```
## Variability matrix in the covariates
  # Define Number of bins
    nb <- 25
    #quantile matrix (of the covariate data)
    q.mat <- matrix(NA, nrow=(nb+1), ncol= nlyr(cov.dat))
    j=1
    for (i in 1:nlyr(cov.dat)){ #note the index start here
    #get a quantile matrix together of the covariates
      ran1 <- minmax(cov.dat[[i]])[2] - minmax(cov.dat[[i]])[1]
      step1 <- ran1/nb
      q.mat[,j] <- seq(minmax(cov.dat[[i]])[1], to = minmax(cov.dat[[i]])[2], b
      j<- j+1}
```

From this matrix, we compute the hypercube matrix of covariates in the whole covariate space.

```
## Hypercube of "objective" distribution (P) - covariates
  # Convert SpatRaster to dataframe for calculations
    cov.dat.df <- as.data.frame(cov.dat)
    cov.mat <- matrix(1, nrow=nb, ncol=ncol(q.mat))
      for (i in 1:nrow(cov.dat.df)){ # the number of pixels
        cntj <- 1
      for (j in 1:ncol(cov.dat.df)){ #for each column
        dd<- cov.dat.df[i,j]
        for (k in 1:nb){  #for each quantile
          kl <- q.mat[k, cntj]
          ku <- q.mat[k+1, cntj]
          if (is.na(dd)) {
            print('Missing')
          }
          else if (dd >= kl & dd <= ku){cov.mat[k, cntj]<- cov.mat[k, cntj] + 1}
        }
        cntj <- cntj+1
      }
    }
```

We then calculate the hypercube matrix of covariates in the sample space.

```
## Sample data hypercube
  h.mat <- matrix(1, nrow=nb, ncol=ncol(q.mat))

  for (ii in 1:nrow(p.dat_I)){ # the number of observations
    cntj <- 1
    for (jj in 2:ncol(p.dat_I)){ #for each column
      dd <- p.dat_I[ii,jj]
      for (kk in 1:nb){  #for each bin
        kl <- q.mat[kk, cntj]
        ku <- q.mat[kk+1, cntj]
        if (dd >= kl & dd <= ku){h.mat[kk, cntj] <- h.mat[kk, cntj] + 1}
      }
      cntj <- cntj+1
```

```
  }
}
```

- **KL–divergence**

We calculate the KL–divergence to measure how much the distribution of covariates in tbe sample space (Q) deviates from the distribution of covariates in the complete study area space (P).

```
## Compare covariate distributions in P and Q with Kullback-Leibler (KL) diverg
    kl.index <-c()
    for(i in 1:ncol(cov.dat.df)){
      kl <-    KL.empirical(c(cov.mat[,i]), c(h.mat[,i]))
      kl.index <- c(kl.index,kl)
      klo <-  mean(kl.index)
    }
    #print(kl.index) # KL divergences of each covariate
    #print(klo) # KL divergence in the existing soil samples
```

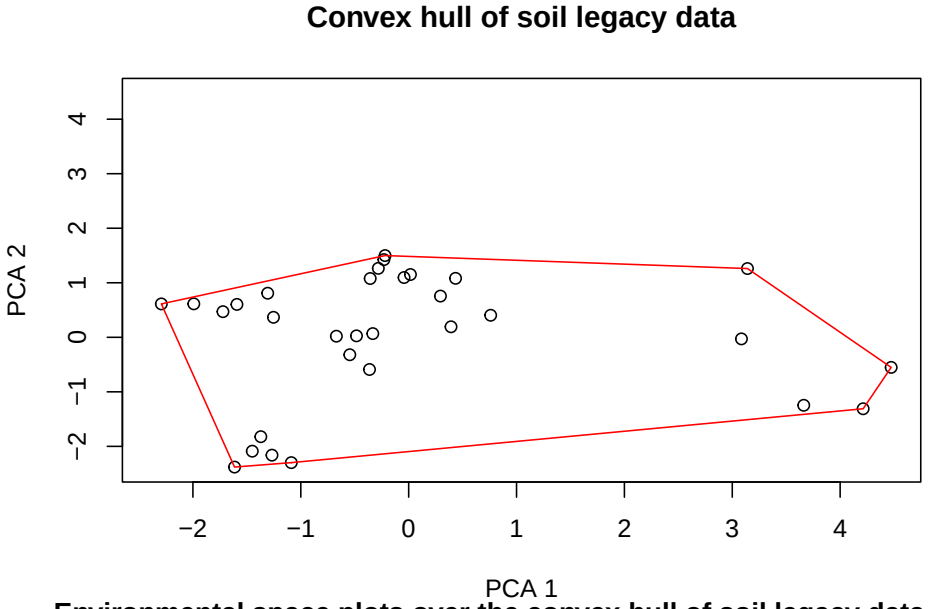
The KL–divergence is always greater than or equal to zero, and reaches its minimum value (zero) only when P and Q are identical. Thus, lower values of KL–divergence are indicative of a good match between both the sample and the study area spaces, indicating that the sample space is a fair representation of the environmental conditions in the study area.

In this case, the KL–divergence value is 0.273, indicating that the legacy samples capture most of the environmental variability in the study area.

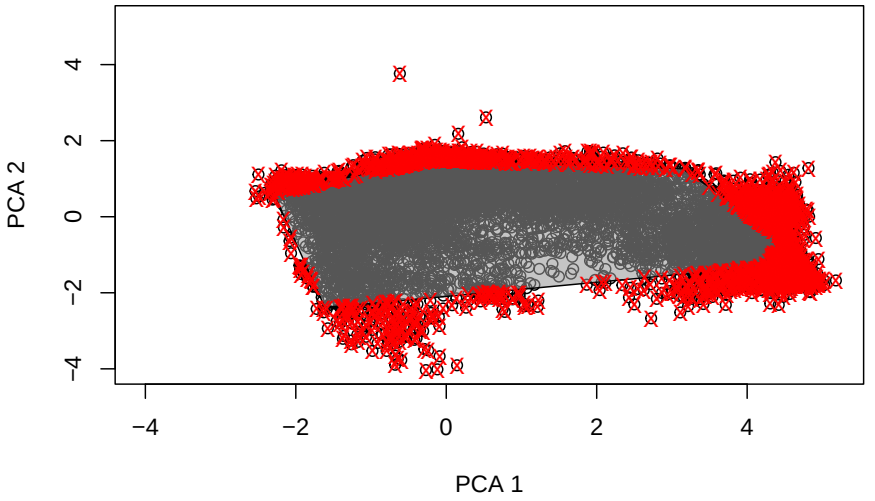
- **Percent of representativeness in relation to the overall environmental conditions**

Finally, we can also determine the degree in which our legacy soil dataset is representative of the existing environmental conditions in the study area. For that, we calculate the proportion of pixels in the study area that would fall within the convex hull polygon delineated upon the environmental conditions found at the soil legacy data locations only. The convex hull polygon is created upon a Principal Component transformation of the covariate data in the soil

legacy data and using the outter limits of the scores of the points projected on the two main Components.



Environmental space plots over the convex hull of soil legacy data



This indicates that 83.8% of the existing conditions in the study area fall within the convex hull delineated with the data in the soil samples, showing the level of adequacy of the proposed legacy data for DSM in the area.

Part I

Part two – Soil Sampling Design

Determining the optimal sampling size

Several strategies exist for designing soil sampling, including regular, random, and stratified sampling. Each strategy comes with its own set of advantages and limitations, which must be carefully considered before commencing a soil sampling campaign. Regular sampling, also called grid sampling, is straightforward and ensures uniform coverage, making it suitable for spatial analysis and detecting trends. However, it may introduce bias and miss small–scale variability. Generally, random sampling may require a larger number of samples to accurately capture soil variability compared to stratified sampling, which is more targeted. Nonetheless, from a statistical standpoint, random sampling is often preferred. It effectively minimizes selection bias by giving every part of the study area an equal chance of being selected. This approach yields a sample that is truly representative of the entire population, leading to more accurate, broadly applicable conclusions. Random sampling also supports valid statistical inferences, ensures reliability of results, and simplifies the estimation of errors, thereby facilitating a broad spectrum of statistical analyses.

The determination of both the number and locations of soil samples is an important element in the success of any sampling campaign. The chosen strategy directly influences the representativeness and accuracy of the soil data collected, which in turn impacts the quality of the conclusions drawn from the study.

In this exercise, we make use of the data provided by (Malone, Minansy and Brungard, 2019) with 4 raster covariates in a 100 has area. We want to determine the minimal number of soil samples that must be collated to capture at least the 95% of variability within the environmental covariates. The pro-

cedure start with random distribution of a low number of samples in the area, determine the values of the spatial covariates, and compare them with those representing the whole diversity in the area at pixel scale. The comparisons are made using the `'Kullback-Leibler divergence (KL)'` – a measure of how the probability distribution of the information in the samples is different from that of the Population, i.e. the covariate space. We also calculate the `'% of representativeness'` as the percent of variability in the covariate information for the complete area related to the variability of covariate information in the sample dataset. Further information can be found in the original work from (Malone, Minansy and Brungard, 2019).

The initial section of the script is related to set–up options in the methodology. We load of R packages, define the working directory, load covariate data, and store it as `SpatRaster` object. Variables related to several aspects of the analyses, such as the aggregation factor of covariates (optional), the creation of a raster stack object(required in the `clhs` function), the initial and final number of samples in the trials, the increment step between trials, and the number of iterations within each trial, are also defined.

```
## Load raster covariate data
  # Read Spatial data covariates as rasters with terra
  cov.dat <-  list.files(raster.path, pattern = "tif$",  recursive = TRUE, full
  cov.dat <- terra::rast(cov.dat)

  # Aggregate raster pixels to simplify data for calculations and build layer s
  cov.dat <- aggregate(cov.dat, fact=agg.factor, fun="mean")

  # Create a raster stack
  cov.dat.ras <- raster::stack(cov.dat)
```

We can see the covariates in a plot.

```
 plot(cov.dat)
```

```
## Define the number of samples to be tested in a loop (from initial to final)
initial.n <- 10
final.n <- 150
by.n <- 20
iters <- 2
```

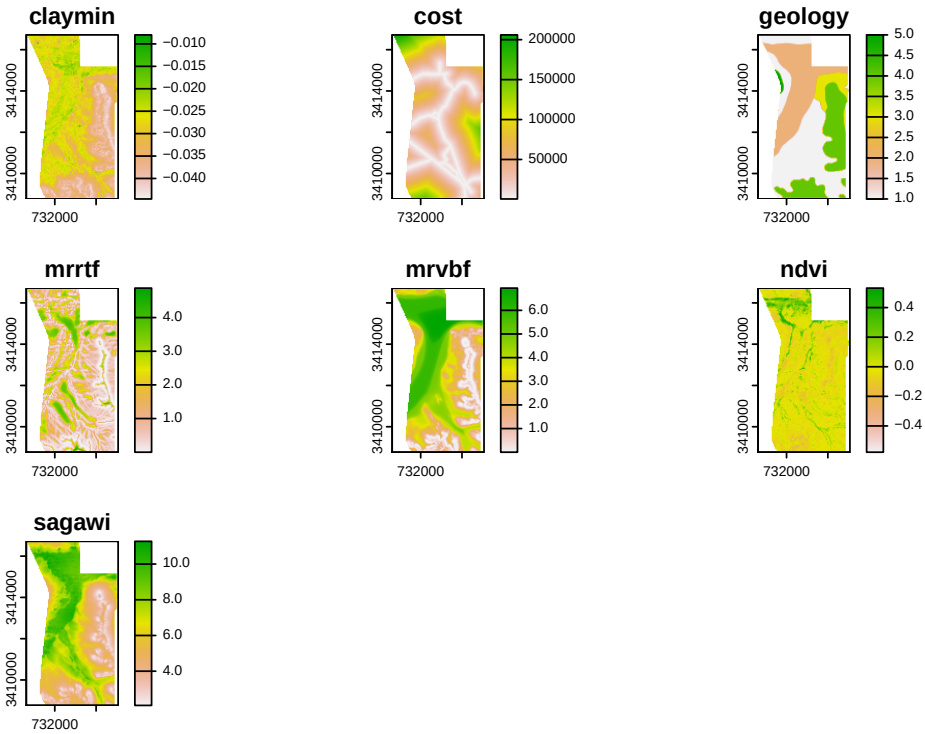


Figure 2: Plot of the covariates

The second section is where the analyses of divergence and representativeness of the sampling scheme are calculated.

The analyses are performed in a loop using growing numbers of samples at each trial. Some empty vectors are defined to store the output results at each loop. At each trial of sample size `'N'`, soil samples are located at locations where the amount of information in the covariates is maximized according to the conditioned Latin Hypercube sampling method in the `'clhs'` package (Roudier *et al.*, 2011). A number of 2 replicates are calculated to determine the amount inter–variability in KL divergence and representativeness in the trial. The final results for each sample size correspond to the mean results obtained from each iteration at the corresponding sample size. The optimal sample size selected correspond to the minimum sample size that accounts for at least 95% of the variability of information in the covariates within the area. The optimal sampling schema proposed correspond to the random scheme at the optimal sample size with higher value of representativeness.

```
  # Define empty vectors to store results
  number_of_samples <- c()
  prop_explained <- c()
  klo_samples <-c()
  samples_storage <- list()

for (trial in seq(initial.n, final.n, by=by.n)){
 for (iteration in 1:iters){
    p.dat_I <- clhs(cov.dat.ras, size = trial, iter = 5000, progress = FALSE, sim

    # Get covariate values for each point
    p.dat_I <- p.dat_I$sampled_data

  # Store samples in list
  samples_storage[[paste0("N", trial,"_",iteration)]] <- p.dat_I

  ## Comparison of population and sample distributions - Kullback-Leibler (KL) di

    # Quantiles of the study area: Number of bins
    nb<- 25
    #quantile matrix (of the covariate data)
    q.mat<- matrix(NA, nrow=(nb+1), ncol= nlyr(cov.dat))
```

```
j=1
for (i in 1:nlyr(cov.dat)){ #note the index start here
  #get a quantile matrix together of the covariates
  ran1 <- minmax(cov.dat[[i]])[2] - minmax(cov.dat[[i]])[1]
  step1 <- ran1/nb
  q.mat[,j] <- seq(minmax(cov.dat[[i]])[1], to = minmax(cov.dat[[i]])[2], by
  j <- j+1}
q.mat

# Hypercube of covariates in study area
cov.dat.df <- as.data.frame(cov.dat) # convert SpatRaster to dataframe
cov.mat <- matrix(1, nrow=nb, ncol=ncol(q.mat))
for (i in 1:nrow(cov.dat.df)){ # the number of pixels
  cntj <- 1
  for (j in 1:ncol(cov.dat.df)){ #for each column
    dd <- cov.dat.df[i,j]
    for (k in 1:nb){  #for each quantile
      kl <- q.mat[k, cntj]
      ku <- q.mat[k+1, cntj]
      if (is.na(dd)) {
        print(paste('N_',trial,'_',iteration,'Missing'))
      }
      else if (dd >= kl & dd <= ku){cov.mat[k, cntj]<- cov.mat[k, cntj] + 1}
    }
    cntj <- cntj+1
  }
}
cov.mat

# Compare whole study area covariate space with the selected sample
# Sample data hypercube (essentially the same script as for the grid data but
h.mat <- matrix(1, nrow=nb, ncol=ncol(q.mat))

for (ii in 1:nrow(p.dat_I)){ # the number of observations
  cntj <- 1
  for (jj in 1:ncol(p.dat_I)){ #for each column
    dd <- as.data.frame(p.dat_I)[ii,jj]
    for (kk in 1:nb){  #for each quantile
```

```
      kl <- q.mat[kk, cntj]
      ku <- q.mat[kk+1, cntj]
      if (dd >= kl & dd <= ku){h.mat[kk, cntj]<- h.mat[kk, cntj] + 1}
    }
    cntj <- cntj+1
  }
}
h.mat


## Compute Kullback-Leibler (KL) divergence
kl.index <- c()
for(i in 1:ncol(cov.dat.df)){
  kl <- KL.empirical(c(cov.mat[,i]), c(h.mat[,i]))
  kl.index <- c(kl.index,kl)
  klo <-  mean(kl.index)
}

## Similarity of the Legacy Dataset: ----
## Calculate the proportion of "env. variables" in the covariate spectra that
# Principal component of the legacy data sample
pca.s = prcomp(data.frame(p.dat_I@data),scale=TRUE, center=TRUE)
scores_pca1 = as.data.frame(pca.s$x)
# Plot the first 2 principal components and convex hull
rand.tr <- tri.mesh(scores_pca1[,1],scores_pca1[,2],"remove") # Delaunay tria
rand.ch <- convex.hull(rand.tr, plot.it=F) # convex hull
pr_poly <- cbind(x=c(rand.ch$x),y=c(rand.ch$y)) # save the convex hull vertic
# plot(scores_pca1[,1], scores_pca1[,2], xlab="PCA 1", ylab="PCA 2", xlim=c(m
# lines(c(rand.ch$x,rand.ch$x[1]), c(rand.ch$y,rand.ch$y[1]),col="red",lwd=1)

# PCA projection of study area population onto the principal components
PCA_projection <- predict(pca.s, cov.dat.df) # Project study area population
newScores = cbind(x=PCA_projection[,1],y=PCA_projection[,2]) # PC scores of p
# Check which points fall within the polygon
pip <- point.in.polygon(newScores[,2], newScores[,1], pr_poly[,2],pr_poly[,1]
newScores <- data.frame(cbind(newScores, pip))
# Plot the polygon and all points to be checked
# if(trial == final.n){
```

```
    # plot(newScores, xlab="PCA 1", ylab="PCA 2", xlim=c(min(newScores[,1:2]), ma
    # polygon(pr_poly,col='#99999990')
    # # Plot points outside convex hull
    # points(newScores[which(newScores$pip==0),1:2],pch='X', col='red')
    # }
    # Proportion of the conditions in the study area that fall within the convex
    #sum(newScores$pip)/nrow(newScores)*100
  klo_samples <- c(klo_samples,klo)
  prop_explained <- c(prop_explained,sum(newScores$pip)/nrow(newScores)*100)
number_of_samples <- c(number_of_samples,trial)
#print(paste("N samples = ",trial, " out of ",final.n, "; iteration = ",iteration
  }
}
```

Figure 3 shows the distribution of covariates in the sample space, and Figure 4 indicates the variability in the estimations of KL divergence and representativeness percent in the 2 within each sample size.

```
  # Plot the polygon and all points to be checked
    plot(newScores[,1:2], xlab="PCA 1", ylab="PCA 2", xlim=c(min(newScores[,1:2]
         col='black', main='Environmental space plots over the convex hull of so
    polygon(pr_poly,col='#99999990')
  # # Plot points outside convex hull
    points(newScores[which(newScores$pip==0),1:2], col='red', pch=12, cex =1)
```

```
  ## Plot dispersion on KL and % by N
  par(mar=c(5, 4, 1, 6))
  boxplot(Perc ~ N, data=results, col = rgb(1, 0.1, 0, alpha = 0.5),ylab = "%")
  mtext("KL divergence",side=4,line=3)
  # Add new plot
  par(new = TRUE,mar=c(5, 4, 1, 6))
  # Box plot
  boxplot(KL ~ N, data=results, axes = FALSE,outline = FALSE,
          col = rgb(0, 0.8, 1, alpha = 0.5), ylab = "")
  axis(4, at=seq(0.02, 0.36, by=.06),  label=seq(0.02, 0.36, by=.06), las=3)
```

We determine the optimal sample size and plot the evaluation results.

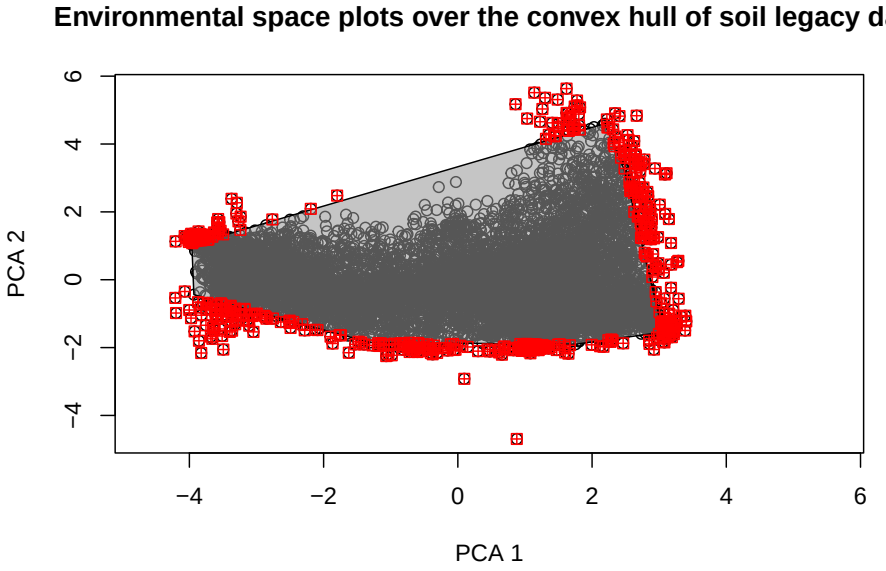


Figure 3: Distribution of covariates in the sample space

The following figure shows the cumulative distribution function (cdf) of the KL divergence and the % of representativeness with growing sample sizes. Representativeness increases with the increasing sample size, while KL divergence decreases as expected. The red dot identifies the trial with the optimal sample size for the area in relation to the covariates analysed.

```
## Plot cdf and optimal sampling point
x <- xx
y <- normalized

mydata <- data.frame(x,y)
opti <- mydata[mydata$x==minimum_n,]

plot_ly(mydata,
        x = ~x,
        y = ~normalized,
        mode = "lines+markers",
        type = "scatter",
```

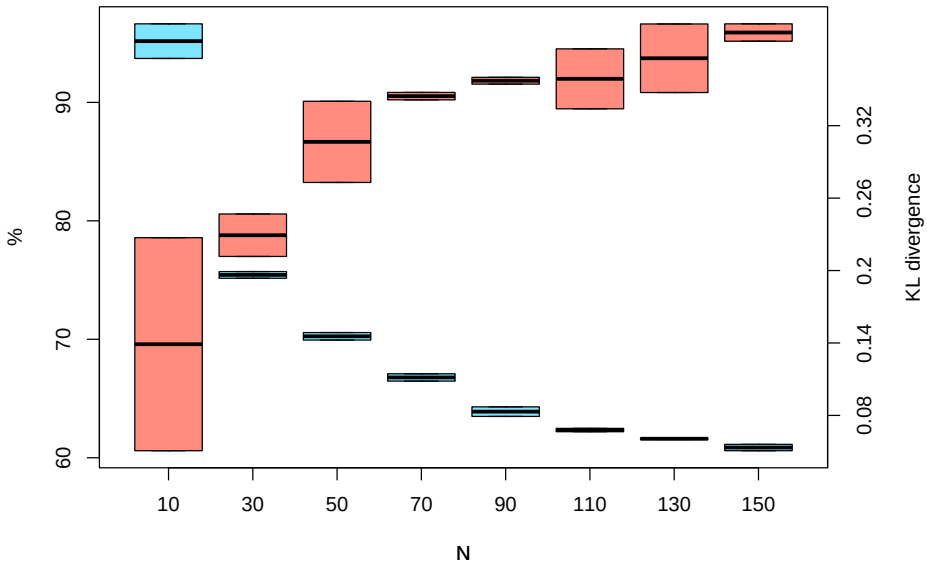


Figure 4: Boxplot of the dispersion in KLO and % representativeness in the iteration trials for each sample size

```
          name = "CDF (1-KL divergence)") %>%
    add_trace(x = ~x,
              y = ~jj,
              mode = "lines+markers",
              type = "scatter",
              yaxis = "y2",
              name = "KL divergence")  %>%
    add_trace(x = ~opti$x,
              y = ~opti$y,
              yaxis = "y",
              mode = "markers",
              name = "Optimal N",
              marker = list(size = 8, color = '#d62728',line = list(color = 'blac
    layout(xaxis = list(title = "N",
                        showgrid = T,
                        dtick = 50,
                        tickfont = list(size = 11)),
           yaxis = list(title = "KL divergence", showgrid = F ),
           yaxis2 = list(title = "1-KL divergence (% CDF)",
                         overlaying = "y", side = "right"),
           legend = list(orientation = "h", y = 1.2, x = 0.1,
                         traceorder = "normal"),
           margin = list(t = 50, b = 50, r = 100, l = 80),
           hovermode = 'x')  %>%
    config(displayModeBar = FALSE)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If
```

KL Divergence and Proportion of Representativeness as function of sample size

According to Figure I, the minimum sampling size for the area, which captures at least 95% of the environmental variability of covariates is N = 81.

Finally, we can determine the optimal distribution of samples over the study area according to these specific results, taking into account the minimum sampling size and the increasing interval in the sample size. The results are shown in Figure 5.

```
## Determine the optimal iteration according to the optimal N size
  optimal_iteration <- results[which(abs(results$N - minimum_n) == min(abs(result
    mutate(IDX = 1:n()) %>%
    filter(Perc==max(Perc))

  # Plot best iteration points
  N_final <- samples_storage[paste0("N",optimal_iteration$N,"_", optimal_iteratio
  plot(cov.dat[[1]])
  points(N_final)
```

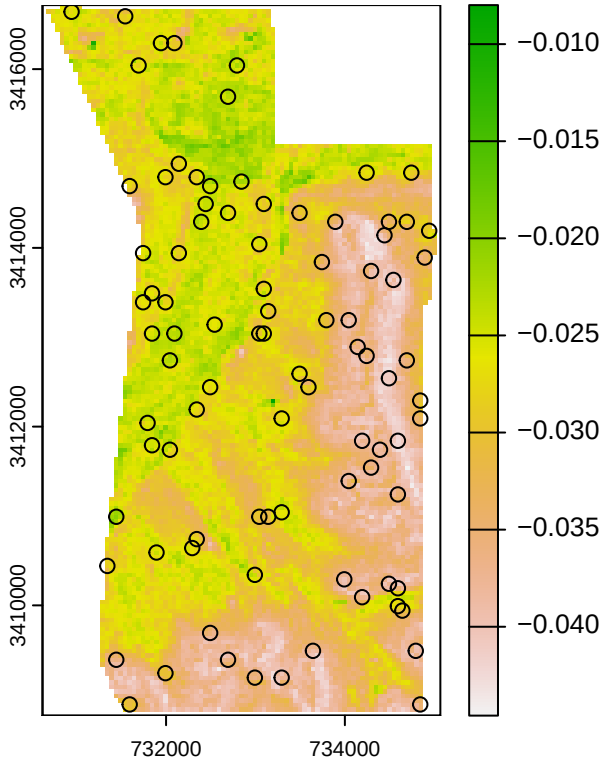


Figure 5: Covariates and optimal number and distribution of samples

In summary, we utilize the variability within the covariate data to ascertain the

minimum number of samples required to capture a minimum of 95% of this variability. Our approach involves assessing the similarities in variability between the sample space and the population space (study area) through calculations of the Kullback–Leibler (KL) divergence and the percentage of similarity at various stages of increasing sample sizes. These results are then utilized to fit a model representing the expected distribution of representativeness as a function of sample size. This model guides us in determining the minimum sample size necessary to achieve a representation of at least 95% of the environmental diversity within the area

Stratified Sampling Design

Stratified random sampling is a technique where the study area is divided into different `groups` or `strata` based on certain environmental traits and a number of random samples are taken from within each group. One of the primary advantages of stratified sampling is its ability to capture the diversity within a population by making sure each group is represented. It can provide a more accurate reflection of the entire population compared to random sampling, especially when the groups are distinct and have unique qualities. This approach is particularly beneficial when certain subgroups within the population are specifically noteworthy. It also allows for more precise estimates with a smaller total sample size compared to simple random choice. Stratified sampling presents some disadvantages. Achieving effective categories requires a proper definition and delineation of the initial information to create the `strata`. The classification of the environmental information into categories and ensuring fair portrayal of each can be intricate and time–taking and mislabeling elements into an improper group can lead to skewed outcomes.

0.4 General Procedure

The creation of a stratified random sampling design involves the identification of relevant features describing the environmental diversity in the area (soil and land use are the environmental variables generally used to define strata), delineation of the strata, determination of the number of samples to distribute to each stratum, followed by random sampling within it. By identifying relevant classes, combining them to define strata and allocating an appropriate number of samples to each stratum, a representative sample can be obtained. Random

sampling within each stratum helps to ensure that the sample is unbiased and provides a fair representation of the overall conditions in the area.

The first question is about how many samples must be retrieved from each strata. The sampling scheme starts with the definition of the total number of samples to collect. In this case, the determination of the sample size is a complex and highly variable process based, among others, on the specific goals of the study, the variability of environmental proxies, the statistical requirements for accuracy and confidence, as well as additional considerations such as accessibility, costs and available resources. The optimal number of samples can be determined following the method proposed in Chapter 2 of this manual. The number of samples within each stratum is calculated using an area–weighted approach taking into account the relative area of each stratum. The sampling design in this section must also comply with the following requirements:

- All sampling strata must have a minimum size of 100 hectares.
- All sampling strata must be represented by at least 2 samples.

This sampling process ensures the representativeness of the environmental combinations present across the area while maintaining an efficient and feasible field sampling campaign.

0.4.1 Strata creation

We must determine the kind of information that will be used to construct the `strata`. In this manual, we present a simple procedure to build strata based on data from two environmental layers: soil groups and land use classification data. The information should be provided in the form of vector shapefiles with associated information databases. The data on both sets often comprises a large number of categories, that would lead to a very large number of `strata`. Thus, it is desirable to make an effort of aggregating similar categories within each input data set, to reduce, as much as possible, the number of categories while still capturing the most of the valuable variability in the area.

The fist step is to set–up the RStudio environment and load the required packages:

We must define the number of samples to distribute in the sampling design and the soil and land use information layers to build the strata. We also define a

REPLACEMENT parameter to account for a reduction of the sampling area according to a certain area using predefined bounding–box, that can be also here defined.

We proceed with the calculation of soil groups. In this example, soil information is stored in the field `TYPES`. We have analysed the extent to which the information in this field can be synthesized to eliminate redundancy when creating the `strata`. [1]. The results are shown in 0.4.1

```
## Plot the map of the aggregated soil classes

  map = leaflet(options = leafletOptions(minZoom = 11.4)) %>%
    addTiles()
  mv <- mapview(soil["USDA_CLASS"], alpha=0, homebutton=T, layer.name = "Soils"
  mv@map
```

Plot of the soil classes

A similar procedure is performed on the land use dataset.

Figure 0.4.1 shows the landuse classes to build the strata.

```
# Plot map with the aggregated land use information

  map = leaflet(options = leafletOptions(minZoom = 11.4)) %>%
  addTiles()
  mv <- mapview(lc["LU"], alpha=0, homebutton=T, layer.name = "Landuse", map=map)
  mv@map
```

Plot of the land use classes

To create the soil–land use `strata` we must combine both classified datasets.

```
  # Combine soil and land use layers
  soil_lc <- st_intersection(soil, lc)
  soil_lc$soil_lc <- paste0(soil_lc$USDA_CLASS, "_", soil_lc$LU)
  soil_lc <- soil_lc %>% dplyr::select(soil_lc, geometry)
```

[1] This exploratory work is a prerequisite and must be adapted specifically to each soil and land use dataset

Finally, to comply with the initial requirements of the sampling design, we calculate the areas of each polygon, delete all features with extent lesser than 100 has.

The final strata map is shown in Figure 0.4.1.

```
  # Plot final map of stratum
map = leaflet(options = leafletOptions(minZoom = 11.4)) %>%
addTiles()
mv <- mapview(soil_lc["soil_lc"], alpha=0, homebutton=T, layer.name = "Strata",
mv@map
```

Plot of strata

0.5 Stratified random sampling

This example demonstrates how to establish a stratified random sampling approach within the previously defined strata polygons. The allocation of sample points is proportionate to the stratum areas, with the condition that each stratum must contain a minimum of 2 samples. The determination of sampling points, referred to as `'target points'`, is made during the initial phase of the sampling design and takes into consideration factors such as the area to be sampled, budget constraints and available personnel. Additionally, a set number of `'replacement points'` must be designated to act as substitutes for 'target points' in cases where some of the original target points cannot be accessed or sampled. These 'replacement points' are systematically indexed, with each index indicating which 'target point' it serves as a substitute for.

Results are shown in Figure 0.5.

```
map = leaflet(options = leafletOptions(minZoom = 11.4)) %>%
      addTiles()
mv <- mapview(soil_lc["soil_lc"], alpha=0, homebutton=T, layer.name = "Strata")
      mapview(sf::st_as_sf(z), zcol = 'type', color = "white", col.regions = c(
mv@map
```

Plot of strata and random target and replacement points

0.6 Stratified random sampling for large areas

The implementation of a stratified random sampling, along with target and replacement points, can present operating difficulties when dealing with areas of significant size and with locations that are hard to reach. To address this issue, the sampling approach can be modified by excluding areas with limited accessibility.

This modification can streamline fieldwork operations and establish a feasible sampling method while still retaining the essence of the stratified random sampling framework. By excluding areas with limited accessibility, the sampling design can be adjusted to ensure a more practical and effective approach to data collection.

- **Delineation of sampling accessibility:** The sampling area can be further limited based on accessibility considerations. Areas with very limited accessibility, defined as regions located more than 1 kilometre away from a main road or access path, may be excluded from sampling areas. To accomplish this, a map of main roads and paths can be used to establish a sampling buffer that includes areas within a 1–kilometre buffer around the road infrastructures. This exclusion helps to eliminate the most remote and challenging–to–access areas. An additional layer of accessibility information can be incorporated based on population distribution in the country, considering that, if population is present, there is a high change that points in the surroundings can be accessible for sampling. In this case, populated nuclei are vectorized into points and a 250–meter buffer is then generated around each point. These resulting areas can be then added to the 1–kilometre buffer around the roads, which collectively defined the final sampling area.
- **Substitution of replacement points with replacement areas in close proximity to the target points:** The sampling design presented before included designated replacement points to serve as substitutes for each target point in the case that it would be inaccessible during fieldwork. However, this approach presented challenges, particularly for large areas, as the replacement point could be located far from the target point, resulting in significant logistical efforts. This limitation posed a risk of delays in completing the sampling campaign within the allocated time frame. To address this challenge, an alternative strategy is to replace the idea

of replacement points with replacement areas situated in the immediate vicinity of the target point. The replacement area for each target point is now confined within a 500–meter buffer surrounding the target and falls within the same sampling stratum. This approach concentrates sampling and replacement activities within a specific geographic area, streamlining the overall process. By reducing the need for extensive travel, this method enhances efficiency and facilitates sample collection. Figure 2 illustrates the distribution of sampling points and replacement areas for visualization.

- **Additional area exclusion:** Some areas can be identified as not suitable for sampling purposes. This is the case of certain natural protected areas, conflict regions presenting risks for field operators, etc. These areas must be identified masked at an initial stage of the design to exclude them from the sampling strata.

The procedure is the same as that previously presented, with the difference that buffers and exclusion areas must be masked–out from the strata map before performing the random sampling.

```
# Compute sampling areas WITH REPLACEMENT -----
 if(REPLACEMENT){
     # Load strata
     soil_lc <- st_read("../soil_sampling/JAM/strata.shp")

     # Read sampling. points from previous step
     z <- st_read("../soil_sampling/JAM/sampling_points.shp")

     # Define buffer of 500 meters (coordinate system must be in metric base)
     samples.buffer = 500
     buf.samples <- st_buffer(z, dist=samples.buffer)

     # Intersect buffers
     samples_buffer = st_intersection(soil_lc, buf.samples)
     samples_buffer <- samples_buffer[samples_buffer$type=="Target",]
     samples_buffer <- samples_buffer[samples_buffer$soil_lc==samples_buffer$grc
     # Save Sampling areas
     #st_write(samples_buffer, paste0('../soil_sampling/JAM/replacement_areas_',

     # Write target points only
```

```
    targets <- z[z$type=="Target",]
    #st_write(targets, '../soil_sampling/JAM/sampling_points_TAR.shp', delete_d
}
```

0.7 Stratified regular sampling

The procedure for creating a stratified regular sampling design is identical to that presented for stratified random sampling, with the only distinction that the locations of the sampling points are distributed in a regular spatial grid. This transformation is achieved by changing the method from 'random' to 'regular' in the spatSample functions within the script above.

```
 map = leaflet(options = leafletOptions(minZoom = 11.4)) %>%
      addTiles()
mv <- mapview(soil_lc["soil_lc"], alpha=0, homebutton=T, layer.name = "Strata")
      mapview(sf::st_as_sf(z), zcol = 'type', color = "white", col.regions = c(
mv@map
```

Plot of strata and regular sampling points

Conditioned Latin Hypercube Sampling

Conditioned Latin Hypercube Sampling (cLHS) is an advanced statistical method used for sampling multidimensional data developed within the context of digital Soil Mapping. It's an extension of the basic Latin Hypercube Sampling (LHS) technique, a statistical method for generating a distribution of samples of a random variable. The main advantage of LHS over simple random sampling is its ability to ensure that the entire range of the auxiliary variables are explored. It divides the range of each variable into intervals of equal probability and samples each interval.

The term 'conditioned' refers to the way the sampling is adapted or conditioned based on specific requirements or constraints. It often involves conditioning the sampling process on one or more additional variables or criteria. This helps in generating samples that are not just representative in terms of the range of values, but also in terms of their relationships or distributions. cLHS is particularly useful for sampling from multivariate data, where there are multiple interrelated variables as it occurs in soil surveys. The main advantage of cLHS is its efficiency in sampling and its ability to better capture the structure and relationships within the data, compared to simpler sampling methods and ensures that the samples are representative not just of the range of each variable, but also of their interrelations. Detailed information on cLHS can be found in (Minasny and McBratney, 2006).

In this manual, we use the R implementation of cLHS by (Roudier *et al.*, 2011) and available as an R package. Additionally, we also included the CLHS analyses using the package `'sgsR'`(https://cran.r-project.org/web/packages/sgsR/)

from (Goodbody *et al.*, 2023), since it provides options to include buffering distance constraints within the cLHS approach.

0.8 cLHS Design

As for stratified sampling, the creation target points from a conditioned Latin Hypercube Sampling design involves the identification of the relevant features describing the environmental diversity in the area. In this case, the environmental parameters are incorporated in the form of raster covariates. The determination of the number of samples in the design is also required. This step can be calculated following the information already provided in this manual.

With the minimum sampling size of 81 calculated before, we can conduct conditioned Latin Hypercube Sampling design for the area in the example using the R package `'cLHS'` available at CRAN.

We use the rasters of 'claymin', 'cost', 'geology', 'mrrtf', 'mrvbf', 'ndvi', 'sagawi' as covariates in the exercise, which we convert to a raster.

```
  # Read Spatial data covariates as rasters with terra
  cov.dat <-  list.files(raster.path, pattern = "tif$",  recursive = TRUE, full.n
  cov.dat <- terra::rast(cov.dat) # SpatRaster from terra
  # Isolate factor raster, in this case layer 3 - geology
  r.factor <- cov.dat[[3]]
  # Aggregate stack to simplify data rasters for calculations
  # 1st aggregate quantitative layers
    cov.dat <- aggregate(cov.dat[[-3]], fact=agg.factor, fun="mean")
  # 2nd aggregate factor layers and add to stack
    cov.dat$geology <- aggregate(r.factor, fact=agg.factor, fun="max")
  # Create a raster::raster stack to be used as input in the clhs::clhs function
    cov.dat.ras <- raster::stack(cov.dat)
```

```
# Plot of covariates
  plot(cov.dat)
```

The distribution of the sampling points is obtained using the `'cLHS'`function together with the stack of raster covariates and the minimum number of samples calculated in the previous Section. The function uses a number of iterations for

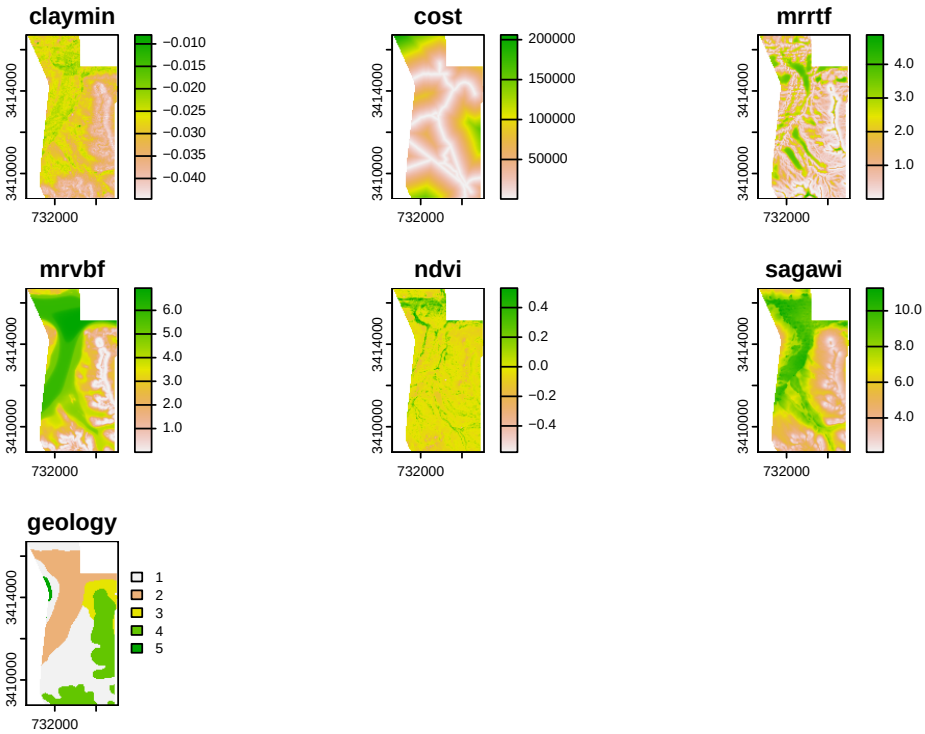


Figure 6: Covariates

the Metropolis–Hastings annealing process, with a default of 10000, to determine the optimal location of samples that account for a maximum of information on the raster covariates (Fig. 7. .

```
# Distribute sampling points with clhs
pts <- clhs(cov.dat.ras, size = minimum_n, iter = 5000, progress = FALSE, simpl
# Plot of objective function
plot(pts, c('obj'))
```

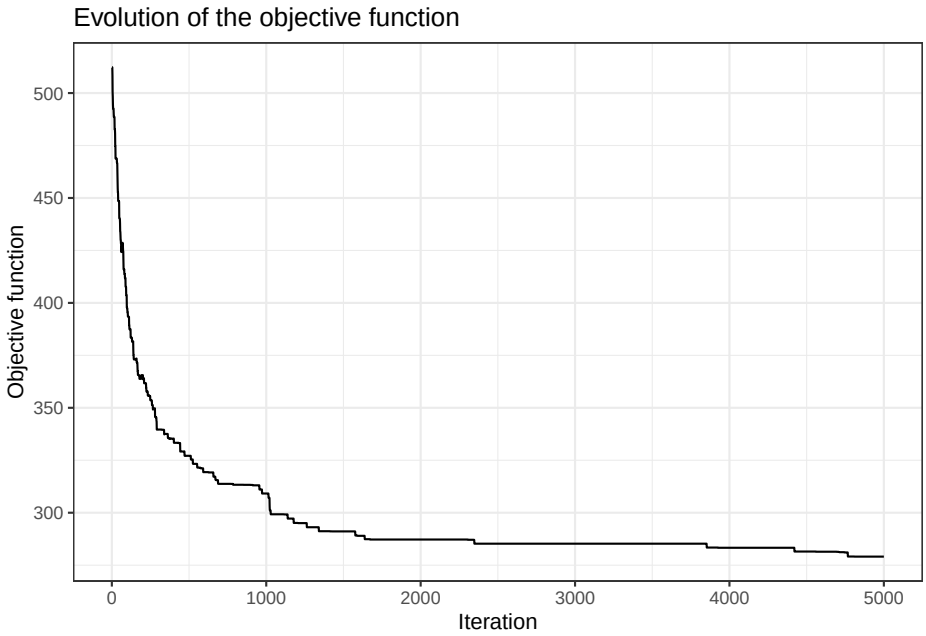


Figure 7: Evolution of the objective function

The distribution of points is shown in Figure 8.

```
## Create a cLHS sampling point set----
  plot(cov.dat[[1]], main="cLHS samples")
  points(pts$sampled_data, col="red", pch = 1)
```

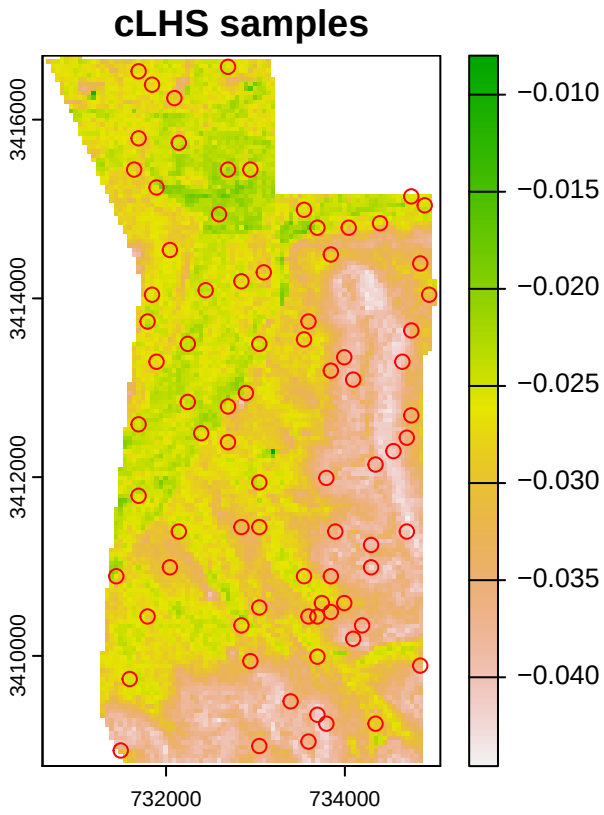


Figure 8: Distribution of cLHS sampling points in the study area

0.9 Including existing legacy data in a cHLS sampling design

In situations where there are legacy soil data samples available, it would be interesting to include them in the cLHS design to increase the diversity of covariates and avoid oversampling for some conditions. In this cases, the ancillary data can be included in the design as additional points to the `'clhs'` function.

```
# We create an artificial legacy dataset of 20 samples over the study area
  legacy.data <- spatSample(cov.dat,20, na.rm=TRUE,xy=TRUE,method="random", as.po

  # Get covariates data as a points
  cov.df<- as.points(cov.dat)

# Merge legacy and covariate information
  leg.new <-   rbind(legacy.data, cov.df)
  leg.new <- as.data.frame(leg.new,geom='XY')
  # Delete data from pixels outside the study area
  leg.new <- na.omit(leg.new)

# Calculate clhs 100 points plus locations of legacy data
    res <- clhs(x = leg.new, size = 100+length(legacy.data),  iter = 10000,simple
            must.include = c(1:nrow(legacy.data)))

# Get sampling points
  points <- res$sampled_data
```

Figure 9 shows the distribution of the created cLHS samples, which also include the position of the original legacy soil data points.

```
# Plot points
  plot(cov.dat[[1]], main="cLHS samples (blue circles) and legacy samples (red di
  points(points[,c("x","y")], col="dodgerblue", pch = 1)
  points(legacy.data, col="red", pch = 5, cex=2)
```

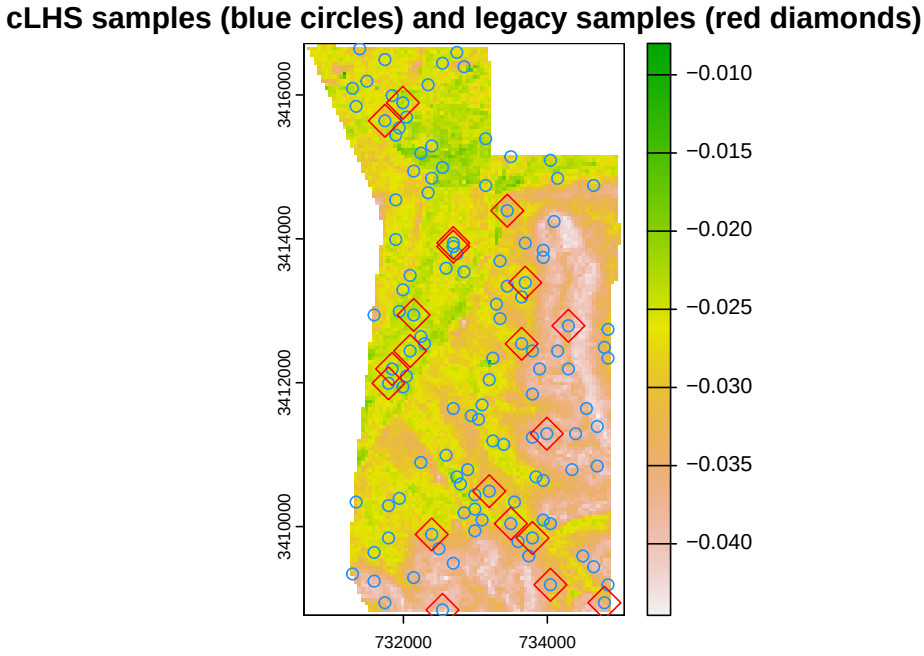


Figure 9: cLHS sampling points with legacy data

0.10 Working with large raster data

The cLHS function samples the covariates in the raster stack in order to determine the optimal location of samples that best represent the environmental conditions in the area. In the case of working with large raster sets, the process can be highly computing demanding since all pixels in the raster stack are used in the process. There are two simple methods to avoid this constraint:

- **Aggregation of covariates:** The quickest solution is to aggregate the covariates in the raster stack to a lower pixel resolution. This is directly performed using the `'aggregate'` function from the `'terra'`package. In case that the raster stack has discrete layers (factor data), the corresponding layers has to be aggregated separately using either the 'min' or 'max' functions to avoid corruption of the data and the results added later to the data of continuous raster layers.

```
## Aggregation of raster stack by a factor of 2.
## The original 10x10m grid resolution is resampled to 20x20m using the mean va
  cov.dat <- aggregate(cov.dat, fact=2, fun="mean")
```

- **Sampling covariate data:** Other method that can be used is to sample the stack (extract the covariates information at point scale) on a regular grid at a lower resolution than the raster grid and use this information as input within the cLHS function. The creation of a regular point grid on the raster stack is straightforward through the function `spatSample` from the `'terra'` package. In this case we create a regular grid of 1000 points.

```
# Create a regular grid of 1000 points on the covariate space
  regular.sample <- spatSample(cov.dat, size = 1000, xy=TRUE, method="regular",
# plot the points over the 1st raster
  plot(cov.dat[[1]], main="Regular resampled data")
  points(regular.sample, col="red", pch = 1)
```

This `dataframe` can be directly used as input in the cLHS function to get locations that best represent the covariate space in the area.

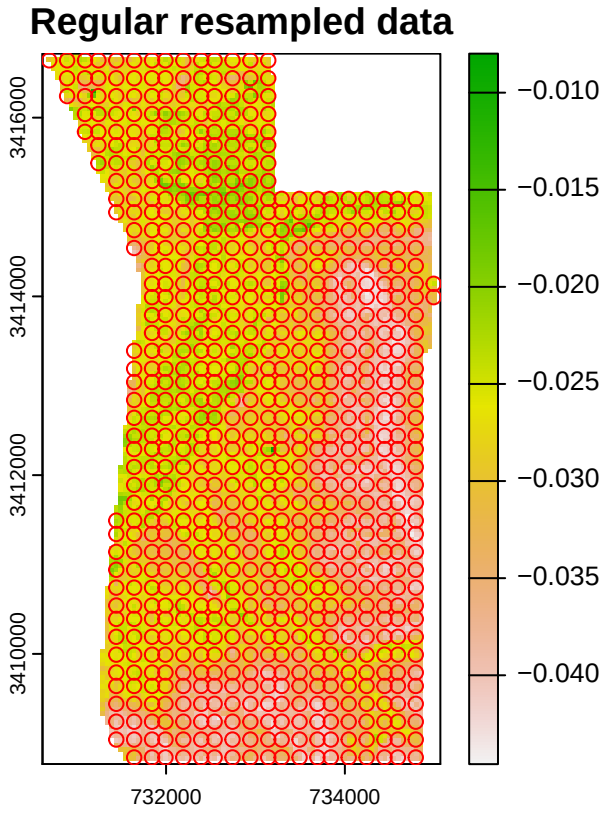


Figure 10: Low resolution points of covariate data

```
# Create clhs samples upon the regular grid
 regular.sample.clhs <- clhs(regular.sample, size = 100, progress = FALSE, iter
# Plot points of clhs samples
  points <- regular.sample.clhs$sampled_data # Get point coordinates of clhs sa
  plot(cov.dat[[1]], main="cLHS samples (red) and covariated resampled points (
  points(regular.sample, col="dodgerblue", pch = 1)
  points(points, col="red", cex=1)
```

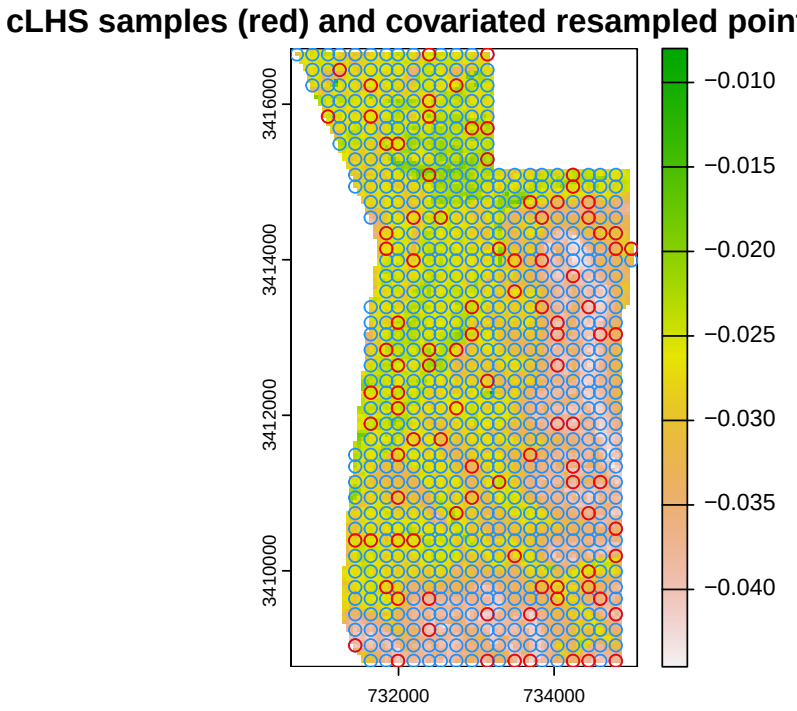


Figure 11: cLHS sampling points on point-grid transformed raster covariate data

Note that the sampling design follows the regular pattern of the regular grid extracted from the raster covariates

0.11 Implemenentation of cost–constrained sampling

There are situation in which the accessibility to some locations is totally or partially restricted such as areas with steep slopes, remote areas, or areas with forbidden access, which highly compromises the sampling process. For these cases, the sampling design can constrain the points to particular locations by defining environmental layers that cause an increment in the cost efficiency of the sampling. This is done with the `cost` attribute in the main `'clhs'` function. The following example uses the raster layer "distance to roads" as a cost layer to avoid low accessible points located at large distance from roads while optimizing the representativeness of the remaining environmental covariates.

```
# Create a cLHS sampling point set with
cost.clhs <- clhs(cov.dat.ras, size = minimum_n, iter = 5000, progress = FALSE,
# Plot objective function
plot(pts, c('obj', 'cost'))
```

Figure 13 shows the distribution of the cost constrained `'clhs'` sampling over the `'cost'` surface. The sampling procedure concentrates, as much as possible, sampling sites in locations with lower costs.

```
# Get and plot the point of samples
points <- cost.clhs$sampled_data  # Get point coordinates of clhs sampling
plot(cov.dat[[2]], main="cLHS samples with 'cost' constraints")
points(points, col="red", cex=1)
```

Cost surfaces can be defined by other parameters than distances to roads. They can represent private property boundaries, slopes, presence of wetlands, etc. The package `'sgsR'` implements functions to define both cost surfaces and distances to roads simultaneously. In this case, it is possible to define an inner buffer distance – i.e. the distance from the roads that should be avoided for sampling and an outer buffer – i.e. the maximum sampling distance) from roads to maximize the variability of the sampling point while considering these limits. The `'sample_clhs'` function in this package also includes options to include existing legacy data in the process of clhs sampling.

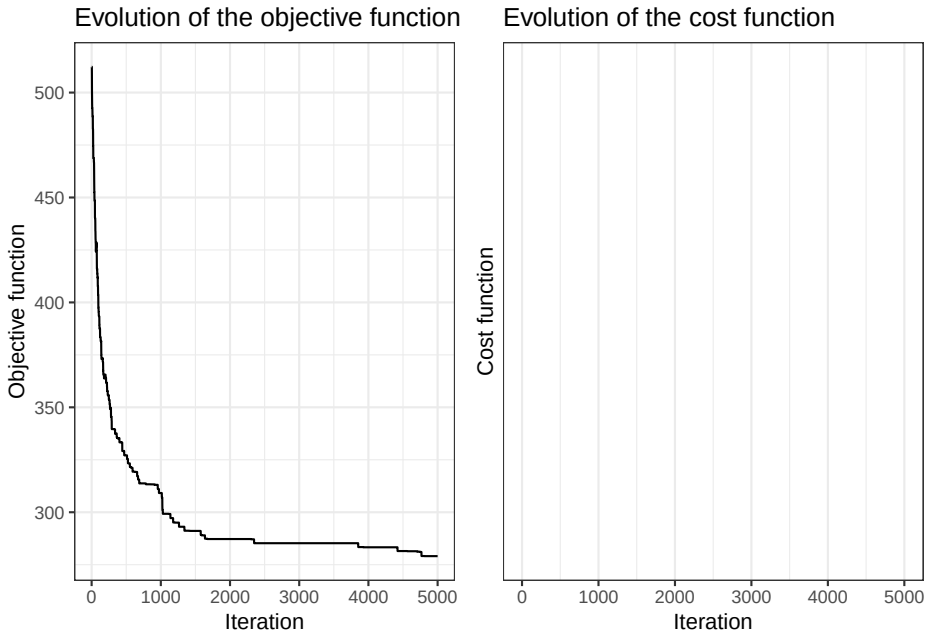


Figure 12: Objective and cost funtions

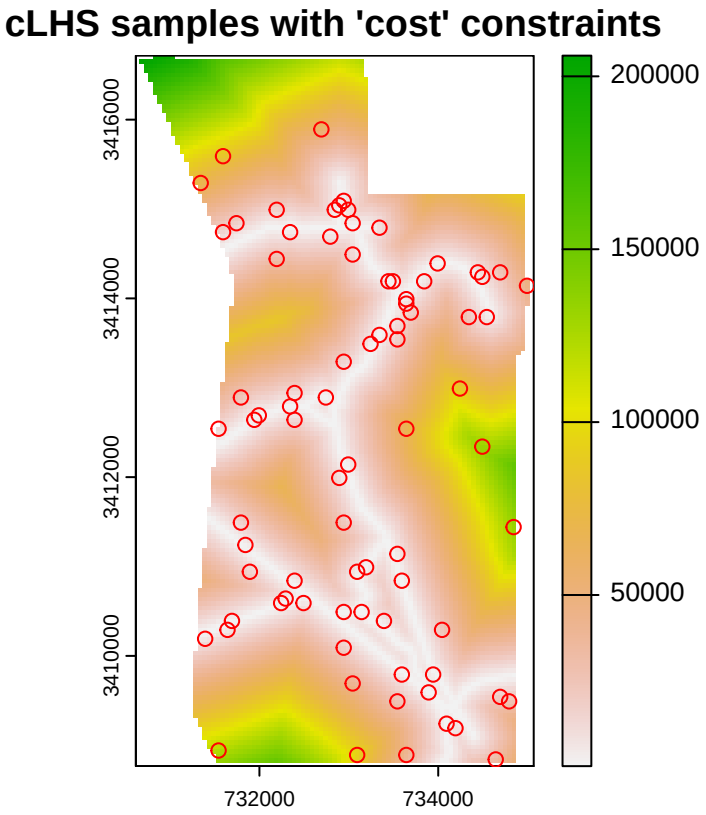


Figure 13: cLHS sampling with cost layers

```
# Load, roads and legacy data
  roads <- sf::st_read(file.path(paste0(shp.path,"/road.shp")),quiet=TRUE)
  legacy <- sf::st_read(file.path(paste0(shp.path,"/legacy_soils.shp")),quiet=TRU

  # Calculate distance to roads and delete NA in outputs
  cost <- cov.dat
  dist2access <- terra::distance(cov.dat[[1]], roads, progress=F)
  cost$dist2access<- dist2access*cost$claymin/cost$claymin

# Calculate clhs points with legacy, cost and buffer to roads
  buff_inner=20;
  buff_outer=300
  aa <- sample_clhs(mraster = cost, nSamp = 300, existing = legacy, iter = 250, d
```

```
## Plot distances, roads, clhs points and legacy data
  plot(cost$dist2access)
  plot(roads,add=TRUE)
  plot(aa$samples[aa$samples$type=="new",], col= "tomato",add=TRUE)
  plot(aa$samples[aa$samples$type=="existing",], col= "navy",add=TRUE)
```

Legacy data is represented as blue dots while new samples from cLHS analyses are in red colour (Fig.13). Note that the new sampling points are located within a distance buffer of 20-300 meters from roads. In addition, a cost surface has also been included in the analyses.

0.12 Replacement areas in cLHS design

The `'cLHS'` package incorporates methods for the delineation of replacement locations that could be utilized in the case any sampling point is unreachable. In this case, the function determines the probability of similarity to each point in an area determined by a buffer distance around the points.

```
  ## Determine the similarity to points in a buffer of distance D
  # Compute the buffers around points
    gw <- similarity_buffer(cov.dat.ras, pts$sampled_data, buffer = D)
```

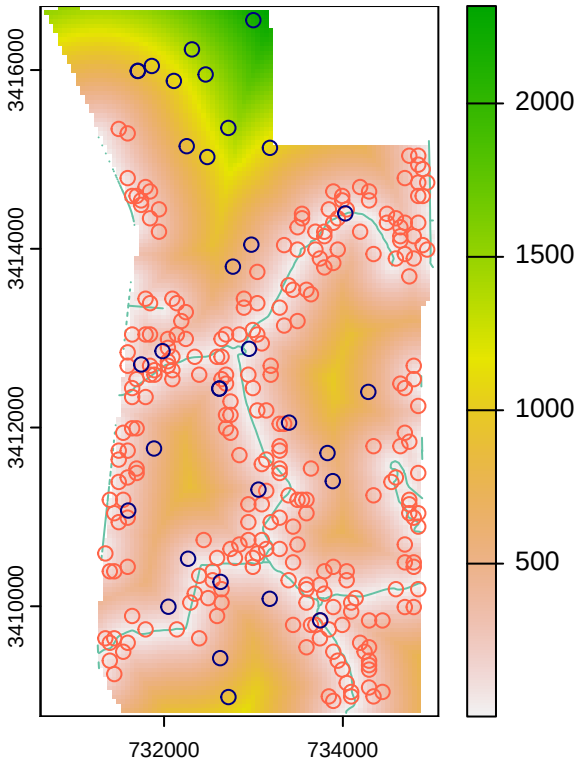


Figure 14: cLHS sampling with legacy data, cost surface and distance buffers around roads

The similarity probabilities for the first cLHS point is presented on Figure 15 over the elevation layer.

```
# Plot elevation
  plot(cov.dat[[3]], legend=TRUE,main=paste("Similarity Probability over elevat
## Overlay points
  points(pts$sampled_data[1], col = "dodgerblue", pch = 3)
## Overlay probability stack for point 1
  colors <- c((RColorBrewer::brewer.pal(9, "YlOrRd")))
  terra::plot(gw[[1]], add=TRUE ,  legend=FALSE, col=colors)
## Overlay 1st cLHS point
  points(pts$sampled_data[1,1], col = "black", pch = 3,cex=1)
```

The probabilities can then be reclassified using a threshold value to delineate the areas with higher similarity to each central target point.

```
# Determine the threshold break to determine if the surrounding area can be a r
  similarity_threshold <- 0.90
# Reclassify buffer raster data according to the threshold break of probability
# 1 = similarity >= similarity_break; NA =  similarity <  similarity_break
  # Define a vector with the break intervals and the output values (NA,1)
  breaks <- c(0, similarity_threshold, NA, similarity_threshold, 1, 1)
  # Convert to a matrix
  breaks <- matrix(breaks, ncol=3, byrow=TRUE)
  # Reclassify the data in the layers from probabilities to (NA,)
  s = stack(lapply(1:raster::nlayers(gw), function(i){raster::reclassify(gw[[i]
```

The reclassified raster stack is then converted to an object of `'SpatialPolygonsDataFrame'` class.

```
## Polygonize replacement areas
 s = lapply(as.list(s), rasterToPolygons, dissolve=TRUE)
 s <- bind(s,keepnames=TRUE)

 # Add the identifier of the corresponding target point
 for(i in 1: length(s)){
   s@data$ID[i] <- as.integer(stringr::str_replace(s@polygons[[i]]@ID,"1.","")
 }
```

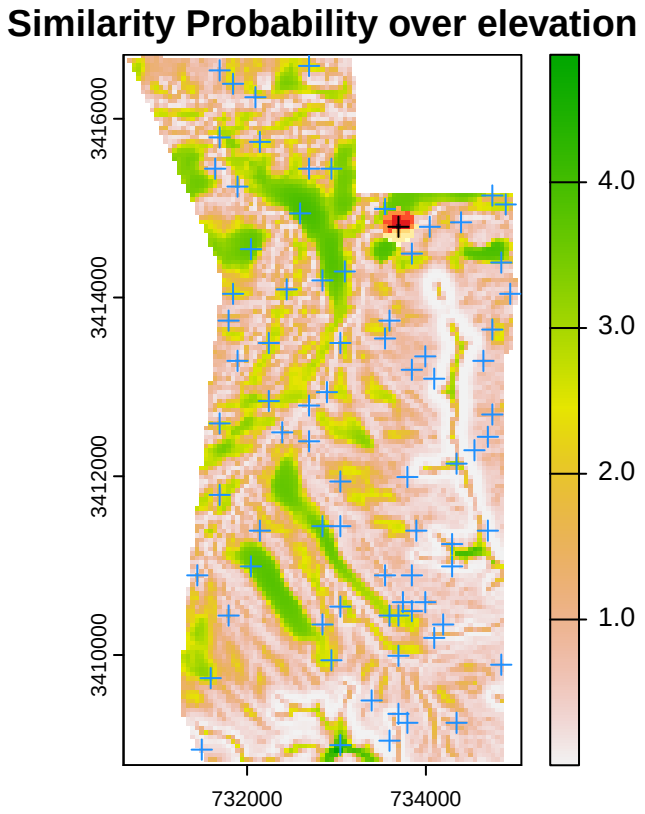


Figure 15: Probability of similarity in the buffer for the first cLHS point (in black) over elevation. The blue crosses represent the location of the remaining cLHS points from the analysis.

```
    # Clean the data by storing target ID data only
    s@data <- s@data["ID"]
```

The results are shown in Figure 16.

```
    plot(cov.dat[[1]], main=paste("cLHS samples and replacement areas for thresho
    plot(s,add=TRUE, col=NA, border="gray40")
    points(pts$sampled_data, col="red", pch = 3)
```

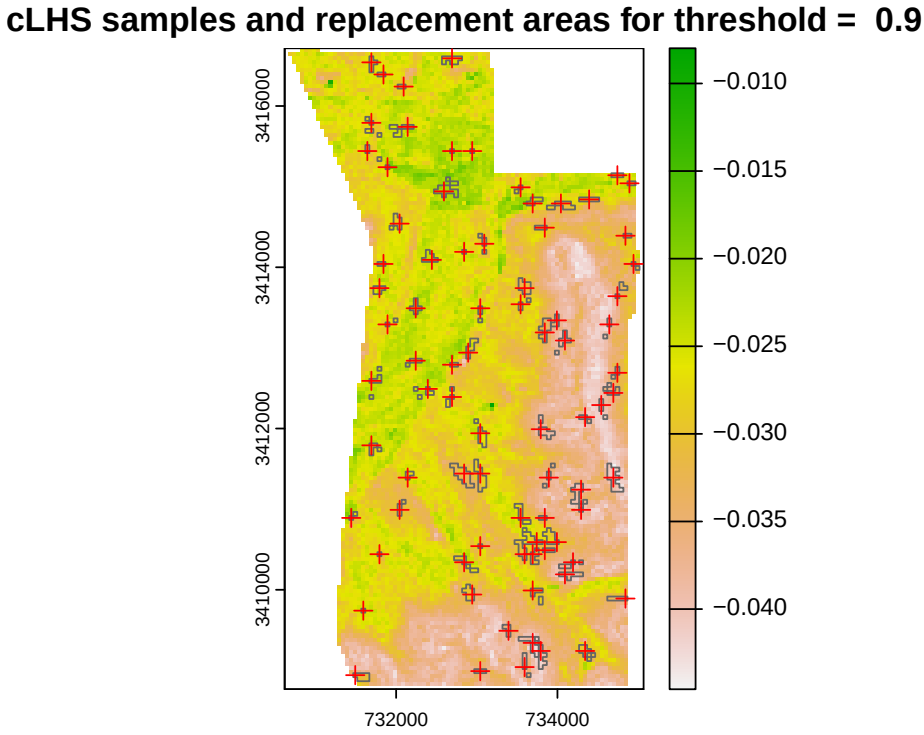


Figure 16: Distribution of cLHS sampling points in the study area

The layer of replacement areas can finally be exported to a `'shapefile'` to be used in any other software.

```
  # Write replacement areas to polygon shape
   s <- st_as_sf(s)
   st_write(s, file.path(paste0(results.path,'replacement_areas_', D, '.shp')),
# Write cLHS sampling points to shapefile
   out.pts <- st_as_sf(pts$sampled_data)
   st_write(out.pts, paste0(results.path,'target_clhs.shp'), delete_dsn = TRUE)
```

References

Goodbody, T.R.H., Coops, N.C., Queinnec, M., White, J.C., Tompalski, P., Hudak, A.T., Auty, D., Valbuena, R., LeBoeuf, A., Sinclair, I., McCartney, G., Prieur, J.-F. & Woods, M.E. 2023. *sgsR: A structurally guided sampling toolbox for LiDAR-based forest inventories.*

Malone, B.P., Minansy, B. & Brungard, C. 2019. Some methods to improve the utility of conditioned latin hypercube sampling. *PeerJ*, 7: e6451. https://doi.org/10.7717/peerj.6451

Minasny, B. & McBratney, A. 2006. A conditioned latin hypercube method for sampling in the presence of ancillary information. *Computers & Geosciences*, 32: 1378–1388. https://doi.org/10.1016/j.cageo.2005.12.009

Roudier, P., Brugnard, C., Beaudette, D., Louis, B., Daust, K. & Clifford, D. 2011. *Clhs: A r package for conditioned latin hypercube sampling.* (also available at https://cran.r-project.org/web/packages/clhs/).