Modelling framework for pedogenon mapping and useful functions for visualizing the classes present in aour study area

```r
##Load packages
#library(Rtools)
#library(rlang)
library(ClusterR)
library(rgdal)
library(gdalUtils)
library(raster)
library(sp)
library(sf)
library(dplyr)
library(tidyverse)
library(ggmap)
library(ggplot2)
library(viridis) # color palettes
library(scales)
library(rasterVis)
library(lattice)
library(gridExtra)
library(tmap)    # for static and interactive maps
library(leaflet) # for interactive maps
library(mapview) # for interactive maps
#library(shiny)   # for web applications
library(foreach)
library(doParallel)
library(geosphere) # calculate distances
library(dendsort)
library(gplots)
library(dendextend) # visualize dendrograms
library(colorspace)
library(slga)


### Functions for examining the study areas

# ### 1. Calculate area by Pedogenon class -----------------------------------

### Function that calculates the area of any Pedogenon map (any k) and returns a summary table
### Inputs:
### kmap - a raster with Pedogenon classes
### fname - name of the file for saving the table into a csv file
### Output: Returns a dataframe with the area of each Pedogenon, k.area.df
pedogenon.area.func <- function(kmap, fname) {

  areaPixels <-raster::area(kmap, na.rm=TRUE)
  s <- stack(kmap, areaPixels)
  k.A <-getValues(s)
  k.A <- as.data.frame(k.A)
  colnames(k.A) <- c("Pedogenon", "Area_km2")
  k.A <- k.A[!is.na(k.A$Pedogenon),]

  k.area.df <-  k.A %>%
    group_by(.,as.factor(Pedogenon), .drop=TRUE ) %>% ## Group by Pedogenon
    summarise(Pedogenon_area = sum(Area_km2, na.rm=TRUE)) ### sum area by Pedogenon class
    k.area.df <- as.data.frame(k.area.df)
```

```r
  colnames(k.area.df) <- c("Pedogenon", "Area_Km2")
  write.csv(k.area.df, file=paste0(fname,".csv")) ### Write table to csv file
  return(k.area.df) ## and return
}

### Returns a table with a row per Pedogenon indicating the closer Pedogenon class,
### the Mahalanobis distance between Pedogenons calculated with CLORPT covariates,
### and the areas that they occupy in NSW, or the study area.
### Note: the distance is calculated with the Euclidean method, but since the data of the
### training dataset was rescaled with the inverse Cholesky transformation,
### the resulting distance is the same as the Mahalanobis distance calculated on the original
data
### Inputs:
### kmodel - kmeans model from the package ClusterR
### k.area.df - is the output of the pedogenon.area.func function
### fname - name to export the table to csv
centroid.dist.func <- function(kmodel, k.area.df, fname){

  ### kmodel is a kmeans model
  ### k.area.df is the output of the Pedogenon.area.func function

  # extract the centroids
  K.centroids <- kmodel$centroids
  K.centroids <- as.data.frame(K.centroids)
  K.centroids$Pedogenon <- c(1:nrow(K.centroids))
  #rownames(K.centroids) <- c(1:nrow(K.centroids))

  ## Is any centroid NA?
  ### Extract the index of the centroids that are na/nan/Inf
  Kcent.nan <- which(apply(K.centroids, MARGIN = 1, FUN = function(x) {any(is.na(x))}))

  ### Calculate distance between all centroids
  dist.centroids <- dist(x=K.centroids[,!names(K.centroids) %in% c("Pedogenon")],
                         method = "euclidean")

  ### Create empty dataframe to store output
  outs <- data.frame(Pedogenon=rep(as.integer(NA),nrow(K.centroids)),
                     ClosestP=rep(as.integer(NA),nrow(K.centroids)),
                     Distance=rep(as.double(NA), nrow(K.centroids)))

  outs$Pedogenon <- K.centroids$Pedogenon ### Assign Pedogenon
  Gs <- as.numeric(as.character(outs$Pedogenon))
  dist.centroids <- as.matrix(dist.centroids)
  ### Calculate distance to the closest Pedogenon
  for(i in 1:nrow(outs)){
    min.dist <- sort(dist.centroids[rownames(dist.centroids)[Gs[[i]]],])[2]
    outs[i,"Distance"] <- min.dist
    outs[i,"ClosestP"] <- names(min.dist)
  }
  ### Remember that those Pedogenons that don't exist are NA
  outs$ClosestP <- ifelse(outs$Pedogenon %in% Kcent.nan, NA, outs$ClosestP )
  outs$Distance <- ifelse(outs$Pedogenon %in% Kcent.nan, NA, outs$Distance )
  colnames(outs) <- c("Pedogenon", "Closest Pedogenon", "Distance")
  outs$Distance <-round(outs$Distance, digits = 3)

  ### Join with the Pedogenon area
  outs$Pedogenon <- as.character(outs$Pedogenon)
  k.area.df$Pedogenon <- as.character(k.area.df$Pedogenon)
  outs <- left_join(outs, k.area.df, by ="Pedogenon")
```

```r
  ### Create column with area of the closest Pedogenon
  outs$Pedo2.Area <- NA
  if(length(Kcent.nan) >0) {
    G.exists <- c(1:nrow(outs))[-Kcent.nan]
  } else if(length(Kcent.nan) == 0) {
    G.exists <- c(1:nrow(outs))
  }

  for(i in 1:length(G.exists)){
    target.G <- outs[outs$Pedogenon == G.exists[[i]], ]$`Closest Pedogenon`
    target.A <- outs[outs$Pedogenon == target.G, ]$Area_Km2
    outs[outs$Pedogenon == G.exists[[i]], ]$Pedo2.Area <- target.A
  }

  colnames(outs) <- c("Pedogenon", "Closest.Pedogenon", "MahabDist", "Area_Km2",
"Closests.Pedo.Area_Km2")

  write.csv(outs, file=paste0(fname,".csv")) ### Write table to csv file
  return(as.data.frame(outs)) ## and return
}

### Function to join a table with the area per Pedogenon for a particular study area (small
study area),
### which results from applying the function pedogenon.area.func, with
### the output from centroid.dist.func for all NSW (or larger study area)
### Inputs:
### study.area.df - is the output of the pedogenon.area.func function for the study area
(small)
### LARGE.centroid.dist.area.df - is the output from the centroid.dist.func, applied to the
full (large) study area
### fname - name of the file to save the output
### Output: a dataframe with 6 columns and a row per pedogenon class.
# Pedogenon - Pedogenon class
# Study_area_km2 - area (km2) within the study area
# LARGE_area_Km2 - Total area of the pedogenon class in the whole (LARGE) area of study (e.g.,
New South Wales)
# Closest.Pedogenon - number designation of the closest pedogenon
# MahabDist - Mahalanobis distance between this centroid to the closest Pedogenon class
# Cl.Pedo.LARGE_area_Km2 - Total area of the closest pedogenon in the whole area (LARGE)
(e.g., New South Wales)

### Function to join the table for the study area and all NSW
study.pedogenon.area.func<- function(study.area.df, LARGE.centroid.dist.area.df, fname) {
  ### Change column names in study.area.df
  colnames(study.area.df) <- c("Pedogenon", "Study_area_km2")
  study.area.df$Pedogenon <- as.character(study.area.df$Pedogenon)
  colnames(LARGE.centroid.dist.area.df) <- c("Pedogenon",
"Closest.Pedogenon","MahabDist","LARGE_area_Km2","Cl.Pedo.LARGE_area_Km2")
  study.area.df <- left_join(study.area.df, LARGE.centroid.dist.area.df, by="Pedogenon")
  study.area.df <- study.area.df[,c("Pedogenon","Study_area_km2","LARGE_area_Km2",
                                    "Closest.Pedogenon","MahabDist","Cl.Pedo.LARGE_area_Km2")]
  study.area.df <- arrange(study.area.df,- Study_area_km2) ### From larger to smaller
Pedogenon class in the study area
  #head(study.area.df)
  study.area.df$Study_area_km2 <-round(study.area.df$Study_area_km2 , digits = 2)
  study.area.df$LARGE_area_Km2 <-round(study.area.df$LARGE_area_Km2 , digits = 2)
  study.area.df$Cl.Pedo.LARGE_area_Km2 <-round(study.area.df$Cl.Pedo.LARGE_area_Km2 , digits =
2)
  write.csv(study.area.df, file=paste0(fname,".csv")) ### Write table to csv file
  return(study.area.df) ## and return
```

```
}


# ### 2. Hierarchical clustering of pedogenons and color legend ------------

### First, perform the hierarchical clustering and save it to plot
### Input: kmodel - kmeans model from the package ClusterR
### Output: Hierarchical cluster (ward.D2 distance) of pedogenon centroids, hclust object
viz.map.legend.hclust <- function(kmodel) {

  ### Extract centroids from model
  centroids <- kmodel$centroids
  ### Extract the index of the centroids that are na/nan/Inf
  Kcent <- as.data.frame(centroids)
  Kcent.nan <- which(apply(Kcent, MARGIN = 1, FUN = function(x) {any(is.na(x))}))
  ### Exclude these clusters from everywhere
  if(length(Kcent.nan) >0) {
    Kcent.exist <- Kcent[-Kcent.nan,]
  } else if(length(Kcent.nan) == 0) {
    Kcent.exist <- Kcent
  }
  # Kcent.exist <- Kcent[-Kcent.nan,]
  ### Hierarchical clustering
  hc <- hclust(dist(Kcent.exist), method="ward.D2")
  #plot(dendsort(hc), main="Hierarchical clustering of kmeans centroids", sub="", xlab="")
  return(hc)

}

### function to choose the number of branches for color ramps
### Input:
### hc.object - hclust object, hierarchical cluster, output from viz.map.legend.hclust
function
### branchN - number of branches that we are considering for this kmeans model
### Output: a plot with the dendrogram and colored branches
viz.branches <- function(hc.object, branchN) {
  hc.object %>% as.dendrogram(.) %>% color_branches(., k = branchN) %>%
    plot(., main = paste0("Colored ",branchN," branches"))
}

# pal.names <- c("OrYel","PurpOr", "Dark Mint", "BurgYl","Turku",
#                "YlOrRd", "RdPu", "Peach", "GnBu","Lajolla",
#                "OrRd", "Greens", "Burg", "Heat 2","Blues",
#                "BuPu")

### Default choice of palettes, for the package colorspace
my_palette <- c("OrYel","PurpOr","TealGrn","BurgYl","RdPu",
                "GnBu","YlOrRd","Peach","Turku","Lajolla",
                "OrRd", "Greens", "Burg", "Heat 2", "Dark Mint",
                "Blues", "SunsetDark", "PuBuGn", "Viridis", "Heat")

### Function to map with the selected color palettes, based on the dedrogram, the Pedogenons
for NSW (or any study area)
### input:
### kmodel - our kmeans model from the ClusterR package
### branchN - number of branches
### pal.names - selection of palettes from colorspace
### legend.name - name for the pdf to plot the legend (dendrogram)
### kmap - raster layer with Pedogenons
```

```r
### Output:
### $hc - Hierarchical clustering of the pedogenon centroids
### $branch.centroids.ord - Table with centroid (Pedogenon number), branch code, and assigned
color
### $legend.plot - dendrogram, legend with assigned colors
### $binpal - for leaflet
### $map.out - leaflet map

viz.map.legend.pal <- function(kmodel, branchN, pal.names, legend.name, kmap, need.proj){

  ### Extract centroids from model
  centroids <- kmodel$centroids
  ### Extract the index of the centroids that are na/nan/Inf
  Kcent <- as.data.frame(centroids)
  Kcent$Pedogenon <- c(1:nrow(Kcent))
  Kcent.nan <- which(apply(Kcent, MARGIN = 1, FUN = function(x) {any(is.na(x))}))
  ### Exclude these clusters from everywhere
  if(length(Kcent.nan) >0) {
    Kcent.exist <- Kcent[-Kcent.nan,]
  } else if(length(Kcent.nan) == 0) {
    Kcent.exist <- Kcent
  }

  ### Perform hierarchical clustering on centroids, with Ward.D2 method
  hc <- hclust(dist(Kcent.exist[,!names(Kcent.exist) %in% c("Pedogenon")]), method="ward.D2")

  ### Extract labels
  hc.labels <- hc %>% as.dendrogram(.) %>% labels %>% as.numeric()

  ### Extract the membership from the tree
  dend <- hc %>% as.dendrogram(.)
  Kcent.exist$branch <- dend %>% dendextend:::cutree.dendrogram(., k = branchN)

  # branch.centroids <-
as.data.frame(cbind(c(as.numeric(as.character(rownames(Kcent.exist)))),
  #
as.numeric(as.character(dendextend:::cutree.dendrogram(dend,k = branchN)))))

  branch.centroids <- Kcent.exist[,c("Pedogenon", "branch")]
  branch.centroids$Pedogenon <- as.numeric(branch.centroids$Pedogenon)
  branch.centroids$branch <- as.numeric(branch.centroids$branch)
  colnames(branch.centroids) <- c("Centroid", "Branch")

  ### sort the dataframe of branch and Pedogenon by the dendrogram labels
  # This line, using functions from dplyr or tidyverse does not work anymore
  # branch.centroids.ord <- branch.centroids %>% left_join(tibble(Centroid = hc.labels), by =
"Centroid")

  reorder_idx <- match(hc.labels,branch.centroids$Centroid) # Saving indices for how to
reorder `branch.centroids$Centroid` to match `hc.labels`
  branch.centroids.ord <- branch.centroids[reorder_idx,]

  numbs.pal <- c((table(Kcent.exist$branch)))
  branch.count <- as.data.frame(cbind(c(1:branchN), numbs.pal))
  colnames(branch.count) <- c("Branch", "Count")
  #branch.count <- branch.count[order(- branch.count$Count),]
  branch.count <- branch.count %>% arrange(., -Count)

  ###Assign color to each
  branch.centroids.ord$colors <- NA
```

```
  for(i in 1:length(numbs.pal)){
    ## Generate as many colors for each pallete as centroids in the branch
    branch.centroids.ord[branch.centroids.ord$Branch == branch.count[i,]$Branch,]$colors <-
      sequential_hcl(pal.names[[i]], n = branch.count[i,]$Count)
  }

  ### Create legend
  ### Reorder the colors depending on the labels
  legend.plot <- dend %>%  set("labels_col", branch.centroids.ord$colors) %>%
    set("branches_k_color", branch.centroids.ord$colors)

  pdf(file = paste0("Map_legend",legend.name,".pdf"), width = 10, height = 100 )
  plot(legend.plot,
       main = "Hierarchical histogram of pedogenon centroids with the map colors",
       horiz = TRUE) # change color
  dev.off()

  ### Now, reorder by Pedogenon class
  branch.centroids.ord <- branch.centroids.ord %>% arrange(., Centroid)
  #branch.centroids.ord <- branch.centroids.ord[order(branch.centroids.ord$Centroid),]

  ### Create palette for leaflet
  #pal <- branch.centroids.ord$colors

  # binpal <- colorBin(palette = branch.centroids.ord$colors,
  #                    bins = c(as.numeric(as.character(rownames(Kcent.exist))),
  #                             tail(as.numeric(as.character(rownames(Kcent.exist))),1)+1),
  #                    na.color = "transparent")

  ### Project the map into the leaflet projection
  if (need.proj == TRUE) {
    kmap <- projectRaster(kmap, crs=CRS("+init=EPSG:3857"), method = "ngb")
  } else if (need.proj == FALSE) {
    kmap <- kmap
    }

  binpal <- colorBin(palette = branch.centroids.ord$colors,
                     bins = c(branch.centroids.ord$Centroid,
                              tail(branch.centroids.ord$Centroid,1)+1),
                     na.color = "transparent")

  map.out <- leaflet() %>%
    # Base groups
    addTiles(group="OSM (default)") %>%
    addProviderTiles("Esri.WorldImagery", group = "World Imagery") %>% # , group = "World
Imagery"
    addProviderTiles("OpenTopoMap", group = "Topo Map") %>%
    addRasterImage(kmap, opacity = 1, colors=binpal, project=need.proj,
                   maxBytes = 300000000, group = "Pedogenons") %>%
    #fitBounds(lng1=140, lat1=-38, lng2=154, lat2=-28) %>%
    leafem::addMouseCoordinates() %>%
    addLayersControl(
      baseGroups = c("OSM (default)","World Imagery", "Topo Map"),
      overlayGroups = c("Pedogenons"),
      options = layersControlOptions(collapsed = FALSE)
    )

  #mapshot(map.out, file = paste0(OutDir,"/Map_",legend.name,".pdf"), remove_url = FALSE)
  output <- list("hc"=hc, "branch.centroids.ord"=branch.centroids.ord,
```

```
                "legend.plot"=legend.plot, "map.out"=map.out)
  return(output)

}


### Function to calculate dendrogram only for the Pedogenons present in the study area
### Input: kmodel - kmeans model
### study.area.map - clip of raster Pedogenons only for the study area
### Output: Hierarchical cluster (ward.D2 distance) of centroids
viz.map.hclust.study.area <- function(kmodel, study.area.map) {
  ### Extract centroids from model
  K.centroids <- kmodel$centroids
  K.centroids <- as.data.frame(K.centroids)
  K.centroids$Pedogenon <- c(1:nrow(K.centroids))
  #rownames(K.centroids) <- c(1:nrow(K.centroids))
  ### Extract the unique values from the Pedogenon maps
  Unique.Geno.sa <- unique(getValues(study.area.map))
  ## Exclude NA
  Unique.Geno.sa <- Unique.Geno.sa[!is.na(Unique.Geno.sa)]
  ## Extract the index of the centroids that are na/nan/Inf
  ### Exclude these clusters from everywhere
  Kcent.exist <- K.centroids[K.centroids$Pedogenon %in% Unique.Geno.sa,]
  ### Hierarchical clustering
  hc <- hclust(dist(Kcent.exist[,!names(Kcent.exist) %in% c("Pedogenon")]), method="ward.D2")
  #plot(dendsort(hc), main="Hierarchical clustering of kmeans centroids", sub="", xlab="")
  return(hc)
}


### Function to map with the selected color palettes, based on the dedrogram, the Pedogenons
for the study area
### input:
### kmodel - our kmeans model
### branchN - number of branches
### pal.names - selection of palettes from colorspace
### legend.name - name for the pdf to plot the legend (dendrogram)
### study.area.map - clip of raster Pedogenons only for the study area
### Output:
### $hc - Hierarchical cluster
### $branch.centroids.ord - Table with centroid, branch, and color
### $legend.plot - dendrogram, legend with colors.
### $binpal
### $map.out - leaflet map
viz.map.legend.pal.study.area <- function(kmodel, branchN, pal.names, study.area.map,
legend.name, need.proj){

  ### Subset number of palettes
  pal.names <- pal.names[1:branchN]

  ### Extract centroids from model
  K.centroids <- kmodel$centroids
  K.centroids <- as.data.frame(K.centroids)
  K.centroids$Pedogenon <- c(1:nrow(K.centroids))
  #rownames(K.centroids) <- c(1:nrow(K.centroids))
  ### Extract the unique values from the Pedogenon maps
  Unique.Geno.sa <- unique(getValues(study.area.map))
  ## Exclude NA
  Unique.Geno.sa <- Unique.Geno.sa[!is.na(Unique.Geno.sa)]
  ## Extract the index of the centroids that are na/nan/Inf
```

```r
  ### Exclude these clusters from everywhere
  Kcent.exist <- K.centroids[K.centroids$Pedogenon %in% Unique.Geno.sa,]
  ### Hierarchical clustering
  hc <- hclust(dist(Kcent.exist[,!names(Kcent.exist) %in% c("Pedogenon")]), method="ward.D2")

  ### Extract labels
  hc.labels <- hc %>% as.dendrogram(.) %>% labels %>% as.numeric()

  ### Extract the membership from the tree
  dend <- hc %>% as.dendrogram(.)
  Kcent.exist$branch <- dend %>% dendextend:::cutree.dendrogram(., k = branchN)

  #branch.centroids <- as.data.frame(cbind(c(as.numeric(as.character(rownames(Kcent.exist)))),
  #
as.numeric(as.character(dendextend:::cutree.dendrogram(dend,k = branchN)))))
  branch.centroids <- Kcent.exist[,c("Pedogenon", "branch")]
  branch.centroids$Pedogenon <- as.numeric(branch.centroids$Pedogenon)
  branch.centroids$branch <- as.numeric(branch.centroids$branch)
  colnames(branch.centroids) <- c("Centroid", "Branch")

  ### sort them by the dendrogram labels
  #branch.centroids.ord <- branch.centroids %>% right_join(tibble(Centroid = hc.labels), by =
"Centroid")
  reorder_idx <- match(hc.labels,branch.centroids$Centroid) # Saving indices for how to
reorder `branch.centroids$Centroid` to match `hc.labels`
  branch.centroids.ord <- branch.centroids[reorder_idx,]

  numbs.pal <- c((table(Kcent.exist$branch)))
  branch.count <- as.data.frame(cbind(c(1:branchN), numbs.pal))
  colnames(branch.count) <- c("Branch", "Count")
  #branch.count <- branch.count[order(- branch.count$Count),]
  branch.count <- branch.count %>% arrange(., -Count)

  ###Assign color to each
  branch.centroids.ord$colors <- NA

  for(i in 1:length(numbs.pal)){
    ## Generate as many colors for each pallete as centroids in the branch
    branch.centroids.ord[branch.centroids.ord$Branch == branch.count[i,]$Branch,]$colors <-
      sequential_hcl(pal.names[[i]], n = branch.count[i,]$Count)
  }

  legend.plot <- dend %>%  set("labels_col", branch.centroids.ord$colors) %>%
    set("branches_k_color", branch.centroids.ord$colors)

  pdf(file = paste0("Map_legend",legend.name,".pdf"), width = 10, height = 40 )
  plot(legend.plot,
       main = "Hierarchical histogram of pedogenon centroids with the map colors",
       horiz = TRUE) # change color
  dev.off()

  ### Now, reorder by Pedogenon class
  branch.centroids.ord <- branch.centroids.ord %>% arrange(., Centroid)
  #branch.centroids.ord <- branch.centroids.ord[order(branch.centroids.ord$Centroid),]

  ### Create palette for leaflet
  #pal <- branch.centroids.ord$colors
  ### Project the map into the leaflet projection
  if (need.proj == TRUE) {
    study.area.map <- projectRaster(study.area.map, crs=CRS("+init=EPSG:3857"), method =
```

```
"ngb")
  } else if (need.proj == FALSE) {
    study.area.map <- study.area.map
    }

  #study.area.map <- projectRaster(study.area.map, crs=CRS("+init=EPSG:3857"), method = "ngb")

  binpal <- colorBin(palette = branch.centroids.ord$colors,
                     bins = c(branch.centroids.ord$Centroid,
                              tail(branch.centroids.ord$Centroid,1)+1),
                     na.color = "transparent")

  map.out <- leaflet(options = leafletOptions(zoomControl = FALSE)) %>%
     #Base groups
    addTiles(group="OSM (default)") %>%
    addProviderTiles("Esri.WorldImagery", group = "World Imagery") %>% # , group = "World
Imagery"
    addProviderTiles("OpenTopoMap", group = "Topo Map") %>%
    addRasterImage(study.area.map, opacity = 1, colors=binpal, project=FALSE,
                   layerId = "values", maxBytes = 300000000, group="Pedogenons") %>%
    #fitBounds(lng1=140, lat1=-38, lng2=154, lat2=-28) %>%
    leafem::addMouseCoordinates() %>%
    addLayersControl(
      baseGroups = c("OSM (default)","World Imagery", "Topo Map"),
      overlayGroups = c("Pedogenons"),
      options = layersControlOptions(collapsed = FALSE)
    )

  #mapshot(map.out, file = paste0(OutDir,"/Map_",legend.name,".pdf"), remove_url = FALSE)
  output <- list("hc" = hc, "branch.centroids.ord" = branch.centroids.ord,
                 "legend.plot" =legend.plot,  "map.out" = map.out)

  return(output)

}


### Function to map the Pedogenons present in a study area, their distribution across all NSW
### It works with the output from the function viz.map.legend.pal.study.area
### Input:
### study.area.Geno.out - Output from viz.map.legend.pal.study.area
### LARGE.Geno.map - Map for the large study area (e.g., NSW) with Pedogenon classes
### Output:
### $map.out - leaflet map
### $kmap - masks the Pedogenon classes not present in the study area
pedogenons.inStudy.area.func <- function(LARGE.Geno.map, study.area.Geno.out, need.proj) {

  ### Mask all Pedogenons not present in the study area
  pedogenons.present <-  study.area.Geno.out$branch.centroids.ord$Centroid ### the Pedogenon
classes present in the study area
  pedogenons.present <- as.numeric(unlist(pedogenons.present))
  kmap <- trim(calc(LARGE.Geno.map, fun= function(x) {ifelse(x %in% pedogenons.present, x,
NA)}))

  if (need.proj == TRUE) {
    kmap <- projectRaster(kmap, crs=CRS("+init=EPSG:3857"), method = "ngb")
  } else if (need.proj == FALSE) {
    kmap <- kmap
    }
```

```
  ### Create palette for leaflet
  ### Reorder by Pedogenon number
  study.area.Geno.out$branch.centroids.ord <- study.area.Geno.out$branch.centroids.ord %>%
     filter(., Centroid %in% pedogenons.present) %>% arrange(., Centroid)

  binpal <- colorBin(palette = study.area.Geno.out$branch.centroids.ord$colors,
                     bins = c(study.area.Geno.out$branch.centroids.ord$Centroid,
                              tail(study.area.Geno.out$branch.centroids.ord$Centroid,1)+1),
                     na.color = "transparent")

  map.out <- leaflet() %>%
    # Base groups
    addTiles(group="OSM (default)") %>%
    addProviderTiles("Esri.WorldImagery", group = "World Imagery") %>% # , group = "World
Imagery"
    addProviderTiles("OpenTopoMap", group = "Topo Map") %>%
    addProviderTiles("Stamen.TonerLite", group = "Stamen.TonerLite") %>%
    addRasterImage(kmap, opacity = 1, colors=binpal, project=FALSE,
                   maxBytes = 300000000, group="Pedogenons")  %>%
    #fitBounds(lng1=140, lat1=-38, lng2=154, lat2=-28) %>%
    # leafem::addMouseCoordinates() %>%
    addLayersControl(
       baseGroups = c("OSM (default)","World Imagery", "Topo Map","Stamen.TonerLite"),
       overlayGroups = c("Pedogenons"),
       options = layersControlOptions(collapsed = FALSE))

  output <- list("map.out" = map.out, "kmap" = kmap)
  return(output)

}


### Variation of previous function. Mapping only those Pedogenons in the study area, but the
surface has to be bigger than a certain value
### It works with the output from the function viz.map.legend.pal.study.area
### Input:
### study.area.Geno.out - Output from viz.map.legend.pal.study.area
### LARGE.Geno.map - Map for the whole study area (e.g., NSW) with Pedogenon classes
### study.area.df - table with the area and Pedogenons present in the study area, output from
study.pedogenon.area.func
### min.area - minimum area for a Pedogenon in order to be included in the map
### Output:
### $map.out - leaflet map
### $kmap - masks the Pedogenon classes not present in the study area
### $dendro.larger.pedogenons -  dendrogram with the larger classes in color, and the others
in grey
pedogenons.inStudy.area.bigger.func <- function(LARGE.Geno.map, study.area.Geno.out,
study.area.df, min.area,need.proj) {

  ### Subset those with an area larger than a certain value
  pedogenons.present  <- study.area.df %>% filter(., Study_area_km2 >= min.area) %>%
dplyr::select(., Pedogenon)
  pedogenons.present <- as.numeric(unlist(pedogenons.present))
  ### Mask all Pedogenons not present in the study area
  kmap <- calc(LARGE.Geno.map, fun = function(x){ifelse((x %in% pedogenons.present), yes = x,
no =NA)})
  if (need.proj == TRUE) {
    kmap <- projectRaster(kmap, crs=CRS("+init=EPSG:3857"), method = "ngb")
  } else if (need.proj == FALSE) {
    kmap <- kmap
```

```
  }
  # kmap <- projectRaster(kmap, crs=CRS("+init=EPSG:3857"), method = "ngb")

  ### Put Color only the main (larger than min.area) Pedogenons
    ### Extract the membership from the tree
  order.desired <- study.area.Geno.out$hc %>% as.dendrogram(.) %>% labels %>% as.numeric()
  reorder_idx <- match(order.desired, study.area.Geno.out$branch.centroids.ord$Centroid)
  # Saving indices for how to reorder `branch.centroids$Centroid` to match `hc.labels`
  study.area.Geno.out$branch.centroids.ord <-
study.area.Geno.out$branch.centroids.ord[reorder_idx,]

  # study.area.Geno.out$branch.centroids.ord <- study.area.Geno.out$branch.centroids.ord %>%
  #  right_join(tibble(Centroid = order.desired), by = "Centroid")

  ### To put in bold
  study.area.Geno.out$branch.centroids.ord$colors <- ifelse(
    study.area.Geno.out$branch.centroids.ord$Centroid %in% pedogenons.present,
    study.area.Geno.out$branch.centroids.ord$colors,
    "gray85")

  study.area.Geno.out$branch.centroids.ord$cex.label <- ifelse(
    study.area.Geno.out$branch.centroids.ord$Centroid %in% pedogenons.present,
    1,
    0.25)

  dendro.larger.pedogenons <- study.area.Geno.out$hc %>% as.dendrogram(.) %>%
    set("labels_col", study.area.Geno.out$branch.centroids.ord$colors) %>%
    set("branches_k_color", study.area.Geno.out$branch.centroids.ord$colors) %>%
    set("labels_cex", study.area.Geno.out$branch.centroids.ord$cex.label)

  #plot(dendro.larger.pedogenons)

  ### Create palette for leaflet

  ### Reorder by Pedogenon number
  # centroid.leaflet.pal <- study.area.Geno.out$branch.centroids.ord %>%
  #   filter(., Centroid %in% pedogenons.present)  %>% arrange(., Centroid)
  #pal <- centroid.leaflet.pal$colors ### Extract the colors from previous output
  # binpal <- colorBin(palette = centroid.leaflet.pal$colors,
  #                    bins = c(centroid.leaflet.pal$Centroid,
  #                             tail(centroid.leaflet.pal$Centroid,1)+1),
  #                    na.color = "transparent")

  study.area.Geno.out$branch.centroids.ord <- study.area.Geno.out$branch.centroids.ord %>%
    filter(., Centroid %in% pedogenons.present) %>% arrange(., Centroid)

  binpal <- colorBin(palette = study.area.Geno.out$branch.centroids.ord$colors,
                     bins = c(study.area.Geno.out$branch.centroids.ord$Centroid,
                              tail(study.area.Geno.out$branch.centroids.ord$Centroid,1)+1),
                     na.color = "transparent")

  map.out <- leaflet() %>%
    # Base groups
    addTiles(group="OSM (default)") %>%
    addProviderTiles("Esri.WorldImagery", group = "World Imagery") %>% # , group = "World
Imagery"
    addProviderTiles("OpenTopoMap", group = "Topo Map") %>%
    addRasterImage(kmap, opacity = 1,  project=FALSE, colors=binpal,
                   maxBytes = 300000000, group="Pedogenons") %>%
    #fitBounds(lng1=140, lat1=-38, lng2=154, lat2=-28) %>%
```

```
    leafem::addMouseCoordinates() %>%
    addLayersControl(
      baseGroups = c("OSM (default)","World Imagery", "Topo Map"),
      overlayGroups = c("Pedogenons"),
      options = layersControlOptions(collapsed = FALSE)
    )

  output <- list("map.out" = map.out, "kmap" = kmap,
                 "dendro.larger.pedogenons"=dendro.larger.pedogenons)
  return(output)

}
```

A. k-means on covariates representing soil forming factors

Author: Mercedes Roman Dobarco

Date: 04/09/2020

Objective: Definition of groups with similar pedogenesis (~ homogeneous soil forming factors, excluding land use and current vegetation) * Prepare covariates + Scale the numerical variables + Sample the categorical variables + Perform PCA on dummy variables created from categorical variables + Predict the dim of the PCA on for the whole raster + Scale the dims of the PCA + Regular sample of all covariates

1. Select CLORPT variables

What are the covariates that represent the soil forming factors? To make the example reproducible we will work with rasters from the Soil and Landscape Grid of Australia, which are available with the package *slga*. The covariates should also include information on parent material, vegetation (including land use), or time. Maybe also soil stable properties. Here, I define the area of interest as a rectangle, but it could be any polygon shapefile.

```
# aoi <- st_read("aoi.shp")
AOI <- c(150,-32, 151,-31)
```

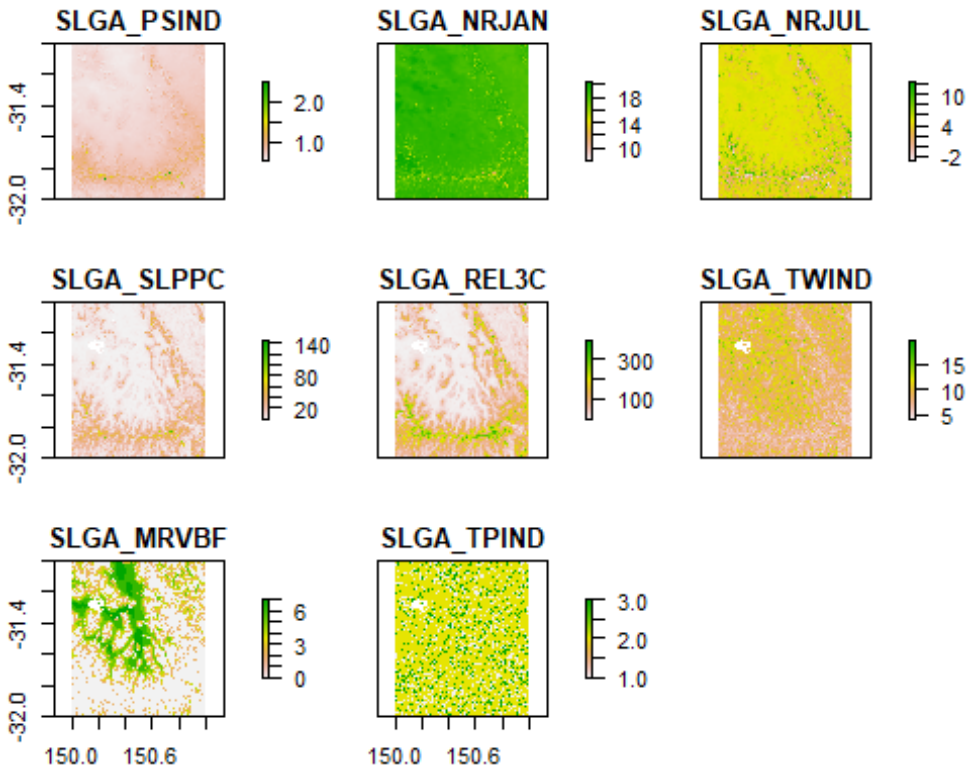
Download covariates representing the soil forming factors at the moment of chosen as reference.

Climate: 'Prescott Index', 'Net Radiation [January]', 'Net Radiation [July]'

Relief: 'Slope [percent]', 'Relief [300m radius]', 'Topographic Wetness Index', 'MrVBF', 'SRTM_TopographicPositionIndex'

Make a stack and plot it.

```
prescott <-get_lscape_data(product = 'PSIND', aoi = AOI, write_out = FALSE)
nrjan <- get_lscape_data(product = 'NRJAN' , aoi = AOI, write_out = FALSE)
nrjul <- get_lscape_data(product = 'NRJUL' , aoi = AOI, write_out = FALSE)
slope <- get_lscape_data(product = 'SLPPC' , aoi = AOI, write_out = FALSE)
rel300m <- get_lscape_data(product = 'REL3C', aoi = AOI, write_out = FALSE)
twi <- get_lscape_data(product = 'TWIND', aoi = AOI, write_out = FALSE)
mrvbf <- get_lscape_data(product = 'MRVBF', aoi = AOI, write_out = FALSE)
tpi <- get_lscape_data(product = 'TPIND', aoi = AOI, write_out = FALSE)
covariates.stack <- stack(prescott,nrjan,nrjul,slope,rel300m,twi,mrvbf,tpi)
#names(covariates.stack)
plot(covariates.stack)
```



2 Scale the numerical variables

What are the continuous covariates?

```
covariates.selection.cont <- c("SLGA_PSIND", "SLGA_NRJAN", "SLGA_NRJUL", "SLGA_SLPPC",
"SLGA_REL3C", "SLGA_TWIND", "SLGA_MRVBF")

library(doParallel)
detectCores()

## [1] 8

cl <- makeCluster(2)   ### Create cluster
registerDoParallel(cl)
getDoParWorkers()

## [1] 2

covariates.cont.out <- foreach(i=1:length(covariates.selection.cont), .packages=c("raster",
"sf")) %dopar% {
  s <- scale(x=covariates.stack[[covariates.selection.cont[[i]]]],center=TRUE, scale=TRUE) #
Scale because it is a continuous variable
  s # Return scaled raster
}
stopCluster(cl)
covariates.cont.sc <- stack(covariates.cont.out)
# plot(covariates.cont.sc)
rm(covariates.cont.out)
```

3 Process the categorical covariates

```
# 3 - Categorical variables ------------------------------------
```

```r
### Mask categorical variables if your study area is bigger
#cat <- c("SLGA_TPIND")
#for (i in 1:length(cat)){
  # setwd(paste0(HomeDir)) # Set wd
  # r <- raster(cat[[i]]) ### load raster *if from file
  # setwd(OutDir) # change wd
  # m <- mask(r,aoi, # Mask pixels outside NSW
  #          filename = paste0(cat[[i]],"_mask.tif"),
  #          format = "GTiff",na.rm=T, inf.rm=T, overwrite = T)
#}
#rm(i, cat)

### Create dummy variables
tpi.classes <- sort(unique(getValues(tpi)))
dummy.list <- list()
for(i in 1:length(tpi.classes)){
  print(i)
  # dummy.tpi <- calc(tpi, fun = function(x) {ifelse(is.na(x), NA,
ifelse(x==tpi.classes[[i]],1,0))},
  #                   filename=paste0("tpi.dummy",tpi.classes[[i]],".tif"),
  #                   overwrite=TRUE)
  dummy.tpi <- calc(tpi, fun = function(x) {ifelse(is.na(x), NA,
ifelse(x==tpi.classes[[i]],1,0))})
  dummy.list[[i]] <- dummy.tpi
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```r
dummy.stack <- stack(dummy.list)
#plot(dummy.stack)

### Perform PCA
set.seed(1946)
# Regular sampling
sampleTPI <- sampleRegular(dummy.stack, size = 600000 , xy=TRUE, na.rm = TRUE, sp = FALSE)
### select only complete cases
sampleTPI <-sampleTPI[complete.cases(sampleTPI),]
sampleTPI <- as.data.frame(sampleTPI)
### Here I take a sample, but this could be done with the whole raster converted as a
dataframe

### Apply PCA
tpi.pca <- prcomp(sampleTPI[,3:ncol(sampleTPI)], scale=TRUE)
# Eigenvalues
library(factoextra)
eig.val <- get_eigenvalue(tpi.pca)
eig.val ### 2 components have 100 % of the variability
```

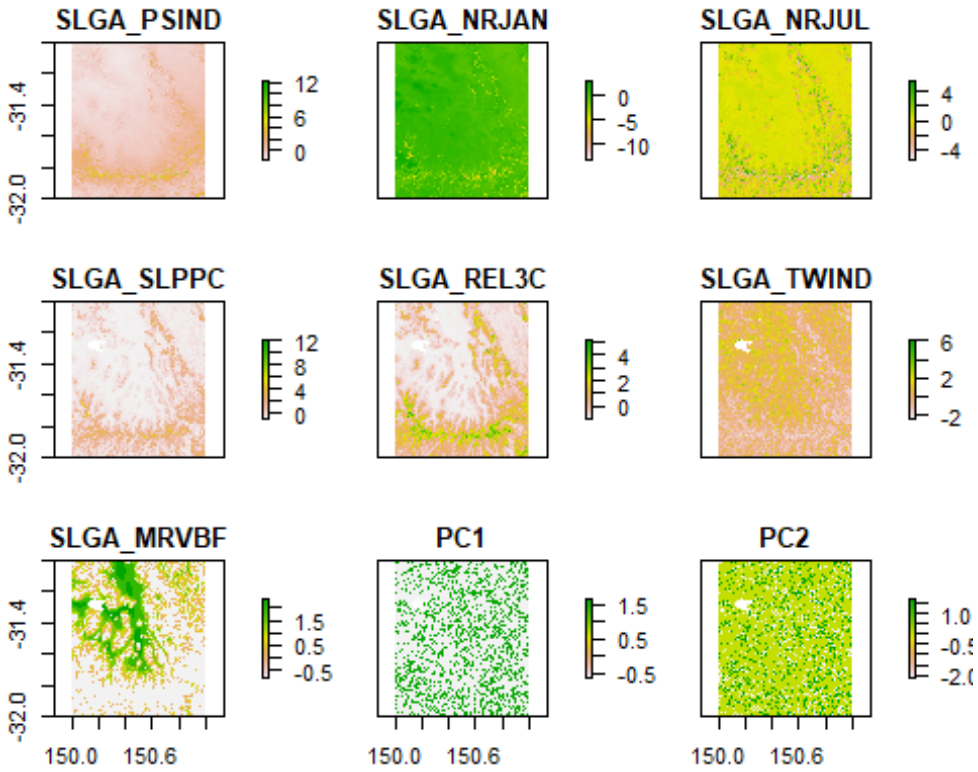
```
##         eigenvalue variance.percent cumulative.variance.percent
## Dim.1 1.839594e+00     6.131981e+01                    61.31981
## Dim.2 1.160406e+00     3.868019e+01                   100.00000
## Dim.3 5.428803e-23     1.809601e-21                   100.00000
```

```r
### If  the study area is very large this step may be done in parallel
tpi.pcs <- predict(dummy.stack, tpi.pca, index=1:2)
# plot(tpi.pcs)
### Write them to file
# for (i in 1:nlayers(tpi.pcs)){
#   writeRaster(tpi.pcs[[i]], filename = paste0("TPI_PC",i,".tif"), overwrite=TRUE, format =
```

```
"GTiff" )
# }
tpi.stack <- stack(tpi.pcs)
### You can scale them afterwards... I don't know if this step is necessary
 for (i in 1:nlayers(tpi.stack)){
   tpi.stack[[i]] <- scale(x=tpi.stack[[i]],center=TRUE, scale=TRUE)
 }
rm(tpi.pcs,tpi.pca, veg.rast,  dummy.stack, eig.val
,sampleTPI,dummy.list,tpi.classes,i,cl,dummy.tpi, covariates.selection.cont)
```

4. Regular sample of all covariates

```
# 4. Regular sample of all covariates ------------------------------------

### Make stack
covariates.stack <- stack(covariates.cont.sc,tpi.stack);plot(covariates.stack)
```




```
names(covariates.stack) <- c("SLGA_PSIND", "SLGA_NRJAN", "SLGA_NRJUL", "SLGA_SLPPC",
                             "SLGA_REL3C", "SLGA_TWIND" ,"SLGA_MRVBF", "TPI_PC1","TPI_PC2" )
# Set the random number generator to reproduce the results
set.seed(1946)
# Regular sampling
sampleCLORPT<- sampleRegular(covariates.stack, size = 50000 , xy=TRUE, na.rm = TRUE, sp =
TRUE)

# transform to sf
sampleCLORPT <- st_as_sf(sampleCLORPT)
### Transform into a dataframe
CLORPT.df <- st_drop_geometry(sampleCLORPT)
### select only complete cases
CLORPT.df <-CLORPT.df[complete.cases(CLORPT.df),]
```

```
## Clean
rm(sampleCLORPT)
```

5. Apply Cholesky decomposition to sample data

The basic Euclidean distance treats each variable as equally important in calculating the distance. An alternative approach is to scale the contribution of individual variables to the distance value according to the variability of each variable. The clustering dataset was rescaled by applying the inverse of the Cholesky transformation of the variance-covariance matrix:

$$Y = XL^{-}1$$

Where Σ_X is the variance-covariance matrix of the environmental covariates sample X, L is the Cholesky factor of Σ_X, a lower triangular matrix with positive diagonal values, and Y is the rescaled covariates dataset. The Euclidean distance calculated on the dataset Y is equal to the Mahalanobis distance calculated in the dataset X. Hence, the correlation among environmental covariates was accounted for during the clustering process.

```
### Calculate the Mahalanobis distance, as Euclidean distance after applying the Cholesky
decomposition
## Take out the coordinates
CLORPT.df.coords <- CLORPT.df[,1:2]
CLORPT.df.subset <- CLORPT.df[,3:ncol(CLORPT.df)]

## What is the average distance between pixels?
#d1 <-distm(CLORPT.df.coords[200:250,], fun = distVincentyEllipsoid)
#d1[1:10,1:10] ### the distance is 158 m and 238 m

# Rescale the data
C <- chol(var(as.matrix(CLORPT.df.subset)))
CLORPT.rs <- as.matrix(CLORPT.df.subset) %*% solve(C)

### euclidean distance of the first 10
b <- dist(CLORPT.rs[1:10,],method = "euclidean")
library(biotools)

## ---
## biotools version 3.1

a <- D2.dist(CLORPT.df.subset[1:10,], cov=var(CLORPT.df.subset));sqrt(a);b ### distances are
the same

##            1         2         3         4         5         6         7
## 2  3.5449794
## 3  3.5970798 2.0909178
## 4  1.1042278 3.4397799 3.1851493
## 5  3.6302077 1.2423931 1.2846911 3.2630681
## 6  3.1402334 1.8676785 1.2925225 3.1125491 1.5398658
## 7  3.4567773 0.3070333 2.0917494 3.4149377 1.2684838 1.7269047
## 8  3.4891635 1.0791443 2.7551362 3.6251622 1.9676154 2.0838211 0.9785830
## 9  3.7053276 0.9304228 2.9161059 3.7972985 2.0254036 2.4209254 0.8741898
## 10 3.1649160 1.9608214 1.2337056 3.1098671 1.6322348 0.5298385 1.8261235
##            8         9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
```

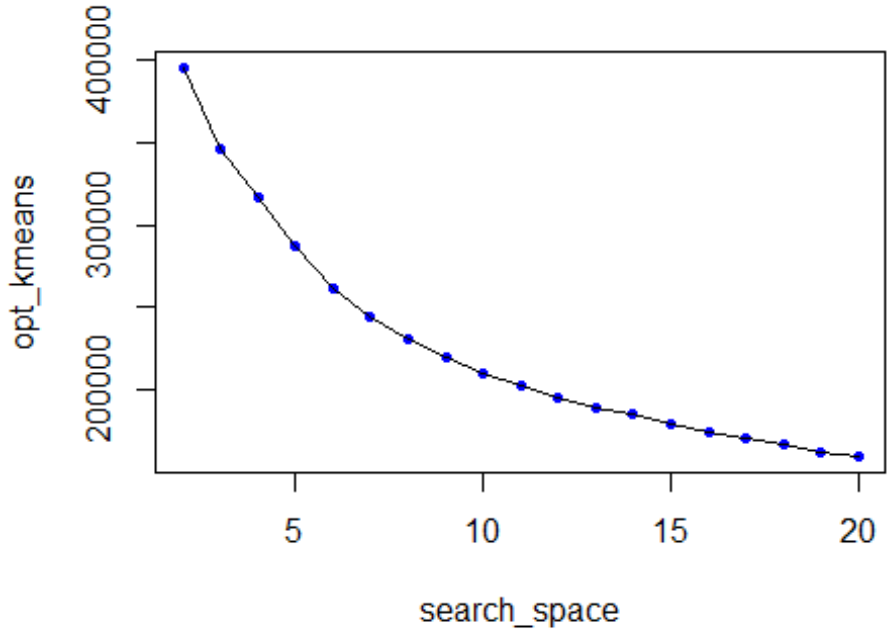
```
## 9  0.6352881
## 10 2.1479922 2.4868608

##            1         2         3         4         5         6         7
## 2  3.5449794
## 3  3.5970798 2.0909178
## 4  1.1042278 3.4397799 3.1851493
## 5  3.6302077 1.2423931 1.2846911 3.2630681
## 6  3.1402334 1.8676785 1.2925225 3.1125491 1.5398658
## 7  3.4567773 0.3070333 2.0917494 3.4149377 1.2684838 1.7269047
## 8  3.4891635 1.0791443 2.7551362 3.6251622 1.9676154 2.0838211 0.9785830
## 9  3.7053276 0.9304228 2.9161059 3.7972985 2.0254036 2.4209254 0.8741898
## 10 3.1649160 1.9608214 1.2337056 3.1098671 1.6322348 0.5298385 1.8261235
##            8         9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  0.6352881
## 10 2.1479922 2.4868608

### Clean
rm(a,b)
```

5. Optimal number of clusters

How can we calculate the optimal number of clusters? There are several methods. One is to check the sum of within-cluster distances across all clusters. Also, the package NbClust can calculate several indices to choose the optimal k.

```
### With the package ClusterR
library(ClusterR)
set.seed(1991)
search_space <- c(2:20) ### Let's say that we want to check between 2 to 10 clusters
opt_kmeans <- Optimal_Clusters_KMeans(data = CLORPT.rs, max_clusters = search_space,
                                      criterion = "WCSSE", num_init = 10,
                                      max_iters = 1000, initializer = "kmeans++",
plot_clusters = TRUE,
                                      verbose = TRUE)

##
##
##   |                                                                      |
|   0%  |                                                                      |====
|   6%  |
|========                                                              |  11%  |
|============                                                          |  17%  |
|================                                                      |  22%  |
|====================                                                  |  28%  |
|========================                                              |  33%  |
|============================                                          |  39%  |
|================================                                      |  44%  |
|====================================                                  |  50%  |
|========================================                              |  56%  |
|============================================                          |  61%  |
|================================================                      |  67%  |
|====================================================                  |  72%  |
|========================================================              |  78%  |
```

```
|=============================================================          |  83%  |
|================================================================       |  89%  |
|====================================================================   |  94%  |
|======================================================================| 100%
##

plot(search_space, opt_kmeans,pch=20, col="blue")
lines(search_space, opt_kmeans)
```

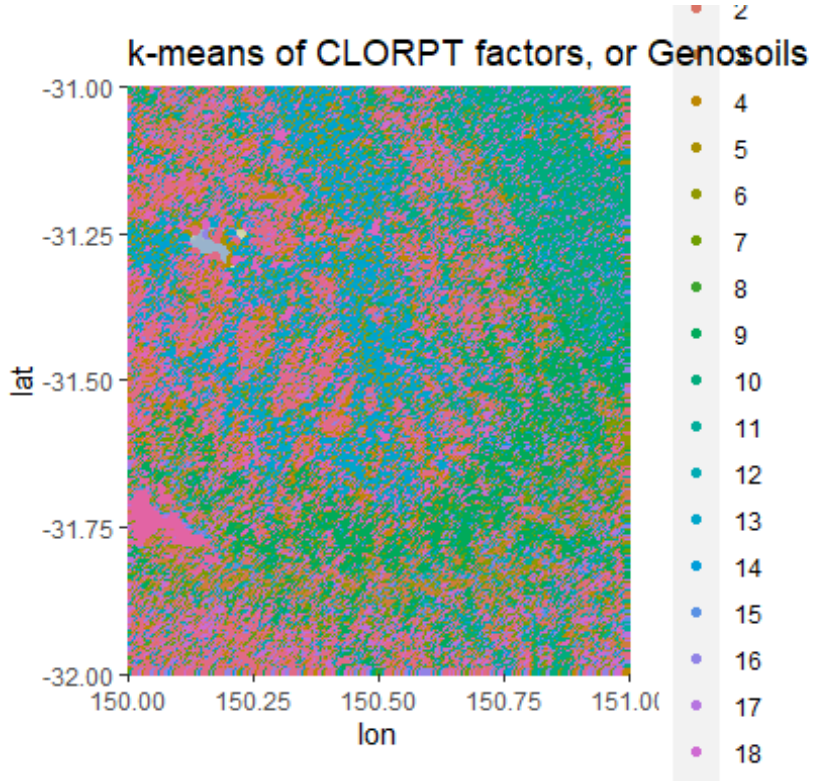



```
### An index that combines the within cluster similarity and the between cluster dissimilarity
is the Calinski-Harbasz index

### Calculate the optimal number of k-means clusters with NbClust package
# library(NbClust)
# set.seed(1984)
# system.time(opt.Clusters.CH <- NbClust(data = CLORPT.rs, diss=NULL, distance = "euclidean",
#                                        min.nc = 2, max.nc = 10,
#                                        method = "kmeans", index = "ch"))
# opt.Clusters.CH
# summary(opt.Clusters.CH)
# opt.Clusters.CH$Best.nc
# search_space <- c(2:10)
# plot(search_space, opt.Clusters.CH$All.index,pch=20, col="blue")
# lines(search_space, opt.Clusters.CH$All.index)
# abline(v=6, col="red", lty=2)
#
# ### the silhouette method is also very popular (check here)
# set.seed(1984)
# system.time(opt.Clusters.silhouette <- NbClust(data = CLORPT.rs, diss=NULL, distance =
"euclidean",
#                                                min.nc = 2, max.nc = 10,
#                                                method = "kmeans", index = "silhouette"))
# opt.Clusters.silhouette
```

```
# summary(opt.Clusters.silhouette)
# opt.Clusters.silhouette$Best.nc
# plot(search_space, opt.Clusters.silhouette$All.index,pch=10, col="blue")
# lines(search_space, opt.Clusters.silhouette$All.index)
# abline(v=6, col="red", lty=2)
```

The output of the indices calculated with the NbClust package suggest that 6 classes is the optimum number. However, for the sake of the example we will make more classes than needed. Let's say 20.

```
### In this case, we choose 6 classes as optimum
# perform KMeans_rcpp clustering
# my_seed <- 4587 ### Your set.seed() number
# km.pedogenon.rcpp <-KMeans_rcpp(data=CLORPT.rs, clusters=6,
#                                 num_init = 20, max_iters = 5000,
#                                 initializer = "kmeans++", fuzzy = TRUE, verbose = FALSE,
#                                 seed = my_seed)

### But for the sake of the example we will make more classes than needed
my_seed <- 4587 ### Your set.seed() number
km.pedogenon.rcpp <-KMeans_rcpp(data=CLORPT.rs, clusters=20,
                                num_init = 20, max_iters = 5000,
                                initializer = "kmeans++", fuzzy = TRUE, verbose = FALSE,
                                seed = my_seed)

### save the cluster number in the original dataframe
CLORPT.df$Cluster.N <- as.factor(km.pedogenon.rcpp$clusters)

### Plot the clusters
### Bounding box of your study area

baseMap <- get_stamenmap(bbox = AOI, ### Here you need to change the coordinates with that of
your study area
                         maptype="terrain-background", zoom=8,
                         source="stamen", crop=TRUE)

## Source : http://tile.stamen.com/terrain-background/8/234/151.png

## Source : http://tile.stamen.com/terrain-background/8/235/151.png

## Source : http://tile.stamen.com/terrain-background/8/234/152.png

## Source : http://tile.stamen.com/terrain-background/8/235/152.png

### Plot the cluster points
library(colorspace)
ggmap(baseMap) +
  geom_point(aes(x = x, y = y, colour = as.factor(Cluster.N)),data = CLORPT.df) +
  ggtitle("k-means of CLORPT factors, or Genosoils")+
  scale_colour_discrete_qualitative(palette = "Dark 3")
```




```
### Clean and save image
rm(covariates.cont.sc, tpi.stack,opt.Clusters.silhouette,my_seed, opt_kmeans,
   opt.Clusters.CH, tpi, mrvbf, nrjan, nrjul, rel300m, prescott,slope, search_space, baseMap,
tpi.stack, twi )

## Warning in rm(covariates.cont.sc, tpi.stack, opt.Clusters.silhouette, my_seed, :
## object 'opt.Clusters.silhouette' not found

## Warning in rm(covariates.cont.sc, tpi.stack, opt.Clusters.silhouette, my_seed, :
## object 'opt.Clusters.CH' not found

## Warning in rm(covariates.cont.sc, tpi.stack, opt.Clusters.silhouette, my_seed, :
## object 'tpi.stack' not found
```

B. Exploring the pedogenon classes

1. Mapping the clusters and write layers

```
### Load the kmeans models and raster (continue script 1.Covariates_Kmeans.R)
# InputDir <-"C:/Covariates/Output"
# setwd(InputDir)
# load("1.Covariate_kmeans.RData")

### Predict the k class with a nested foreach loop

### Extract the index of the centroids that are na/nan/Inf
Kcent.nan <- which(apply(km.pedogenon.rcpp[["centroids"]], MARGIN = 1, FUN =
function(y){any(is.na(y))}))

### Define the size of the blocks --- At each raster row do we start and finish each crop?
bs <- blockSize(covariates.stack, minblocks=2)
```

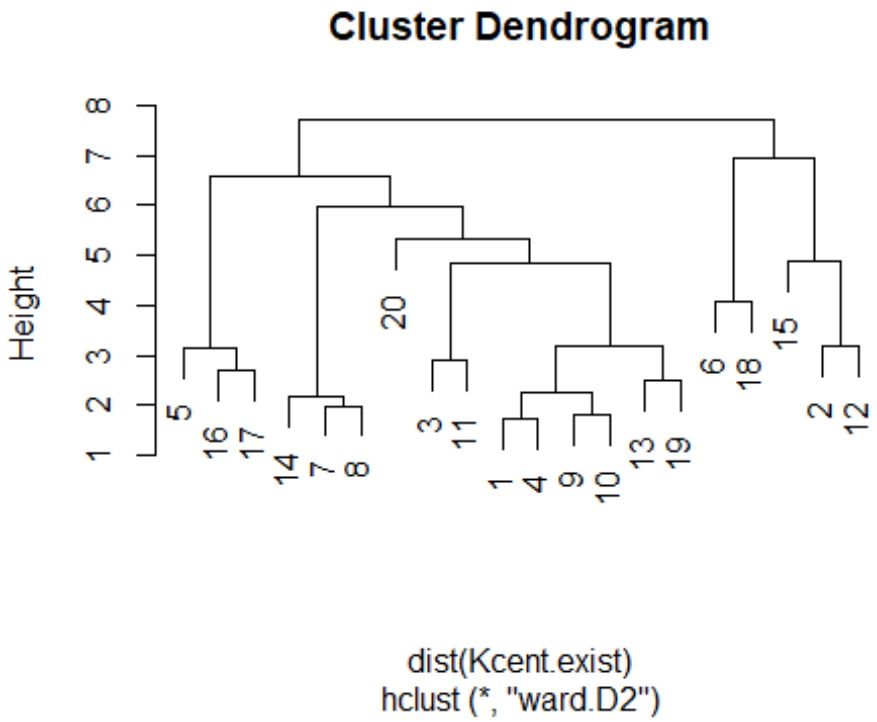
```r
### I crop all raster files (across variables stacks) in a parallel process
### with a %dopar% from the foreach package
### this is thought for a large study area, but of course for the example is not needed

## My desired cluster number
K <- 20
#K <- c(6,12)
### If K is more than one instead of km.pedogenon.rcpp being a kmeans model, it would be a
list of models



  for(m in 1:length(K)){

    cl <- makeCluster(3)
    registerDoParallel(cl)

    k_rast_list <- foreach(i=1:bs$n, .packages=c("raster", "ClusterR")) %dopar% {

      ### Get one tile of the raster stack
      tile <- crop(covariates.stack, extent(covariates.stack, bs$row[[i]],
bs$row[[i]]+bs$nrows[[i]], 1, ncol(covariates.stack)))
      ### Transform into a dataframe
      tile.df <- as.data.frame(tile, row.names=NULL, optional=FALSE, xy=TRUE, na.rm=FALSE,
long=FALSE)

      ### For each new pixel, I first have to rescale its values
      ## Take out the coordinates
      tile.df.coords <- tile.df[,1:2]
      tile.df <- tile.df[,3:ncol(tile.df)]

      # Rescale the data with CLORPT.df (sample from the stack covariates that was used to
calibrate the kmeans in the first place)
      tile.df.rs <- as.matrix(tile.df) %*% solve(C)
      tile.df.rs <- as.data.frame(tile.df.rs)

      ### Predict cluster assignment

      ### Extract the index of the dataframe rows that are na/nan/Inf
      df.na <- which(apply(tile.df.rs, MARGIN = 1, FUN = function(x) {any(is.na(x))}))

      ### Create empty prediction column
      tile.df.rs$cluster <- NA

      ### If K is more than one instead of km.pedogenon.rcpp being a kmeans model, it would be
a list of models
      ### km.pedogenon.rcpp[[m]]$centroids
      ### predict in those rows where there are not na
      tile.df.rs[-df.na, ]$cluster  <- predict_KMeans(data = tile.df.rs[-
df.na,1:(ncol(tile.df.rs)-1)],
                                                      CENTROIDS = km.pedogenon.rcpp$centroids)
      ### Assign the values to a new raster
      k.pred <- setValues(tile[[1]], tile.df.rs$cluster)
      names(k.pred) <- paste0("K",K[[m]])
      k.pred # Return this
    }

    stopCluster(cl)
```

```
    ## Assign function to mosaic
    k_rast_list$na.rm <- TRUE
    k_rast_list$fun <- min
    ## Create mosaic for whole NSW
    k.raster <- do.call(mosaic, k_rast_list)
    names(k.raster) <- paste0("K",K[[m]])

    ## Write to file
    #writeRaster(k.raster, filename= paste0("K",K[[m]],".tif"), na.rm=T,inf.rm=T,
format="GTiff", overwrite=TRUE )
    gc()

  }
```

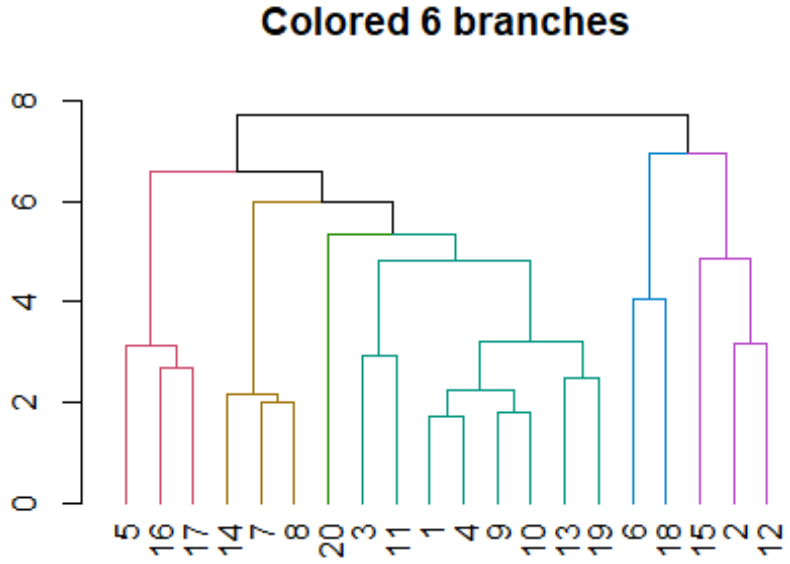
2. Visualization

For this example, we use the whole study area and a smaller section. Clip the whole map to a smaller study area.

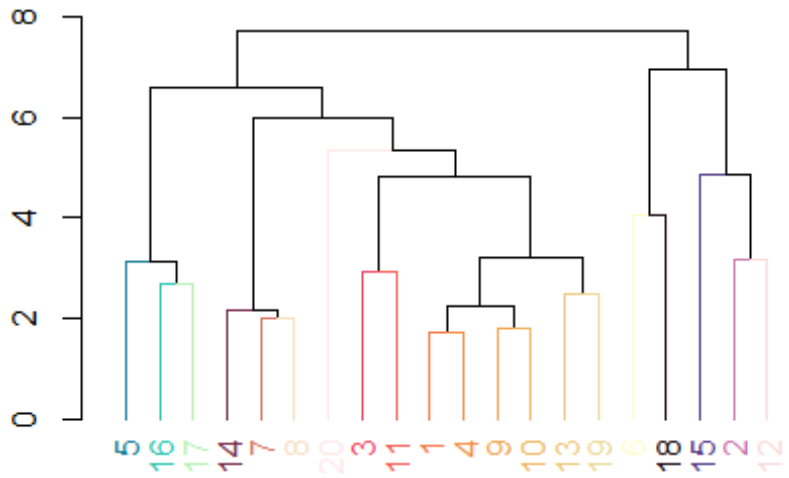
```
pedogenon.zone <- crop(k.raster, extent(150.07323,150.15760,-31.36793,-31.32397))
pedogenon <- k.raster
#plot(pedogenon)
#plot(pedogenon.zone)
pedogenon.3857 <- projectRaster(pedogenon, crs=CRS("+init=EPSG:3857"), method = "ngb")

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## ellps WGS 84 in CRS definition: +proj=merc +a=6378137 +b=6378137 +lat_ts=0
## +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null +wktext +no_defs

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## datum WGS_1984 in CRS definition

pedogenon.zone.3857 <- projectRaster(pedogenon.zone, crs=CRS("+init=EPSG:3857"), method =
"ngb")

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## ellps WGS 84 in CRS definition: +proj=merc +a=6378137 +b=6378137 +lat_ts=0
## +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null +wktext +no_defs

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## datum WGS_1984 in CRS definition

### Map the Whole study area
my_palette <- c("OrYel","PurpOr","TealGrn","BurgYl","Lajolla","Turku","RdPu",
                "GnBu","YlOrRd","Peach",
                "OrRd", "Greens", "Burg", "Heat 2", "Dark Mint",
                "Blues", "SunsetDark", "PuBuGn", "Viridis", "Heat")

hc.whole <- viz.map.legend.hclust(kmodel = km.pedogenon.rcpp)
plot(hc.whole) # 6 branches
```



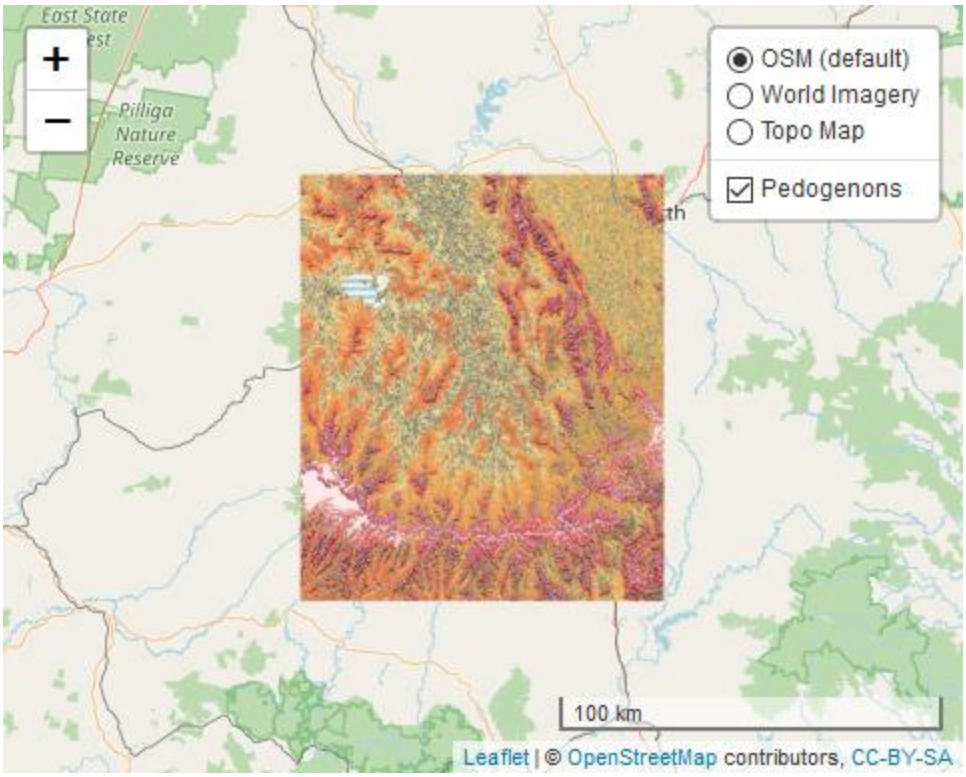

```
### Choose number of branches
viz.branches(hc.whole, 6)
```



```
hc.6.out <-viz.map.legend.pal(kmodel = km.pedogenon.rcpp,
                              branchN =6, pal.names = my_palette,
                              kmap = pedogenon.3857,
                              need.proj = FALSE,
                              legend.name = "pedogenon.6_branches")

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## ellps WGS 84 in CRS definition: +proj=merc +a=6378137 +b=6378137 +lat_ts=0
## +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null +wktext +no_defs

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## datum WGS_1984 in CRS definition

plot(hc.6.out$legend.plot)
```




```
hc.6.out$map.out %>%
  addScaleBar( position = "bottomright",
               options = scaleBarOptions(maxWidth = 200, metric = TRUE, imperial = FALSE,
updateWhenIdle = TRUE))
```



Create table that includes summary statistics on:

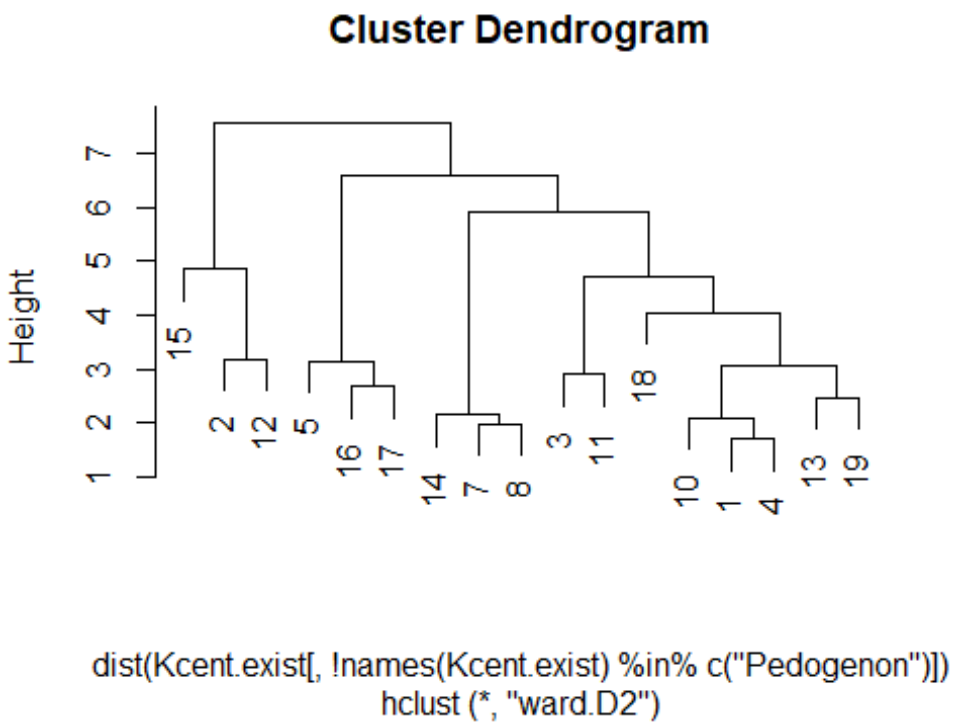
- Pedogenon class area - raster
- Average distance between pixels of that class - raster
- Mahalanobis distance across all clorpton centroids - centroid df
- Mahalanobis distance to the closest class - centroid df

```
K20.Area.df <- pedogenon.area.func(pedogenon, "K20.Area")

## `summarise()` ungrouping output (override with `.groups` argument)

K20.CentrMh <- centroid.dist.func(kmodel=km.pedogenon.rcpp, k.area.df =K20.Area.df, fname =
"K20.CentrMh")

# Study Area
K20.area.zone.df <- pedogenon.area.func(kmap=pedogenon.zone, fname="K20.area.zone")

## `summarise()` ungrouping output (override with `.groups` argument)

### Join tables with pedogenon area present in a study area and Mahalanobis distance to its
closest Genosoil (all NSW)
k20.AreaCentrMh.zone <- study.pedogenon.area.func(study.area.df= K20.area.zone.df,
                                                  LARGE.centroid.dist.area.df =  K20.CentrMh,
                                                  fname= "k20.AreaCentrMh.zone" )
# ### Calculate the % of the large study area occupied by these pedogenons
# total.area <- sum(k6.AreaCentrMh.zone$LARGE_area_Km2, na.rm=TRUE)
# k6.AreaCentrMh.zone$AreaPerc_Large <- k6.AreaCentrMh.zone$LARGE_area_Km2/total.area*100
```

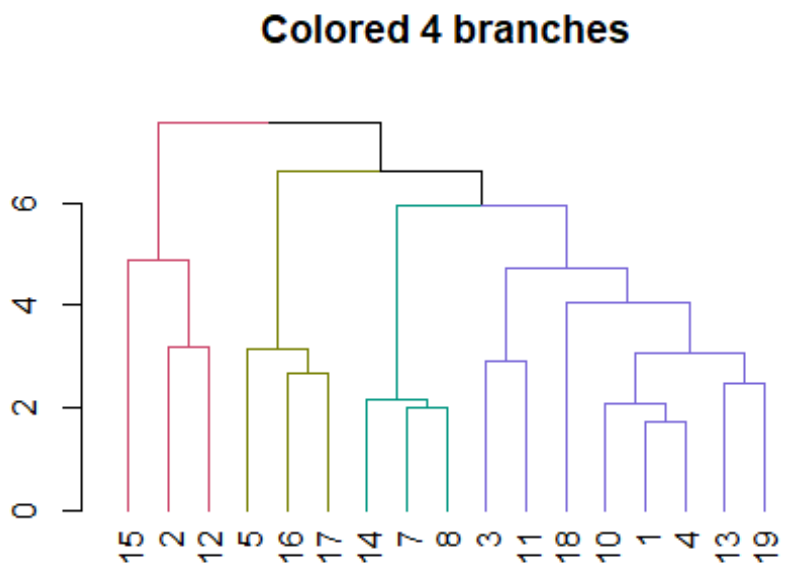
Map the study area with a custom hierarchical cluster color ramp. Apply the function to calculate a dendrogram only for the pedogenons present in the study area.

```
### Map study area with custom hierarchical cluster color ramp
### Function to calculate dendrogram  only for the pedogenons present in the study area
```

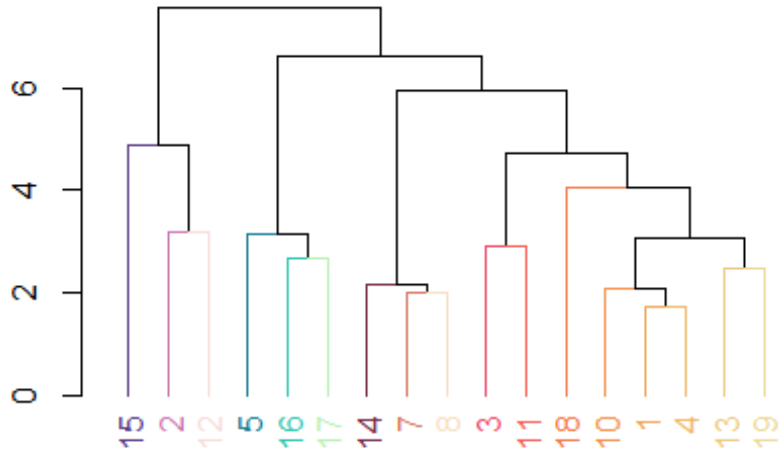
```
hc.zone <- viz.map.hclust.study.area(kmodel = km.pedogenon.rcpp, study.area.map =
pedogenon.zone)
plot(hc.zone) # 4 branches
```



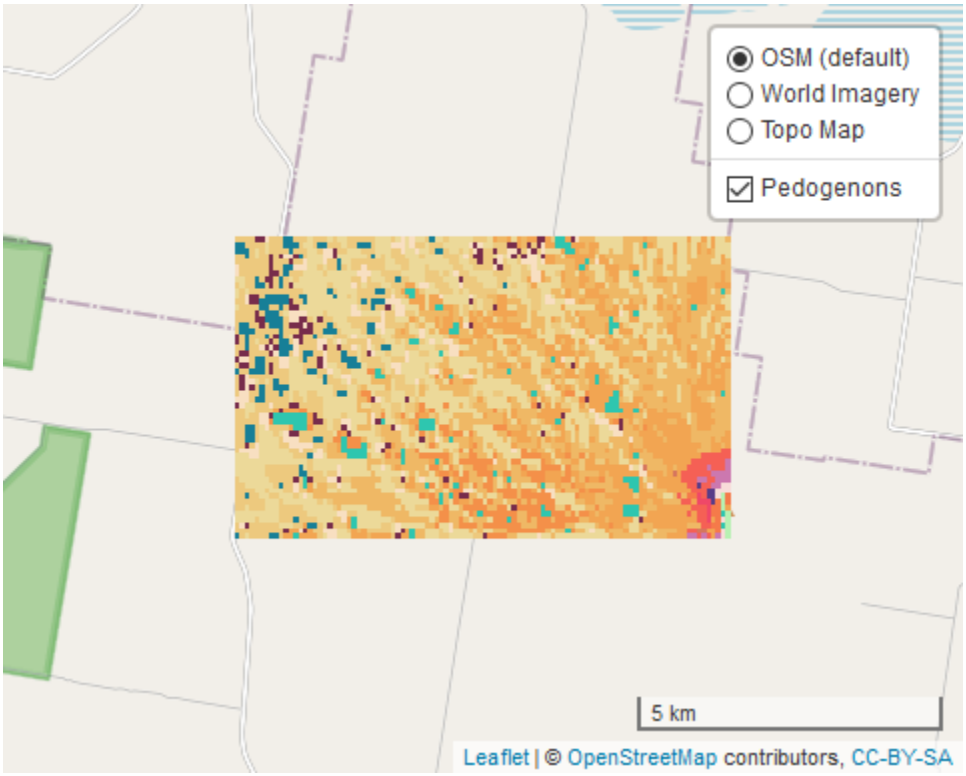

```
### Choose number of branches
viz.branches(hc.zone, 4)
```




```
hc.zone.out <-viz.map.legend.pal.study.area(kmodel = km.pedogenon.rcpp,
                                            branchN =4, pal.names = my_palette,
                                            study.area.map = pedogenon.zone.3857,
                                            need.proj = FALSE,
                                            legend.name = "k4.zone.legend")

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## ellps WGS 84 in CRS definition: +proj=merc +a=6378137 +b=6378137 +lat_ts=0
## +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null +wktext +no_defs

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## datum WGS_1984 in CRS definition

plot(hc.zone.out$legend.plot)
```

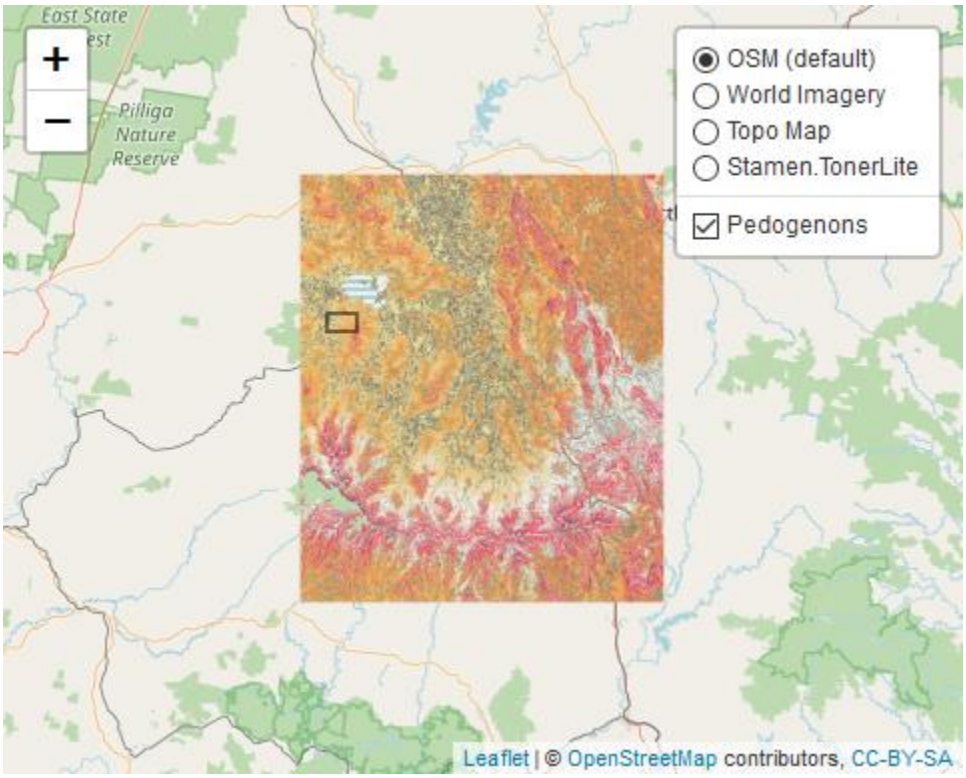



```
hc.zone.out$map.out %>%
  addScaleBar( position = "bottomright",
               options = scaleBarOptions(maxWidth = 200, metric = TRUE, imperial = FALSE,
                                         updateWhenIdle = TRUE))
```



Map the pedogenons present in the study area across the larger area and locate rectangle around the farm.

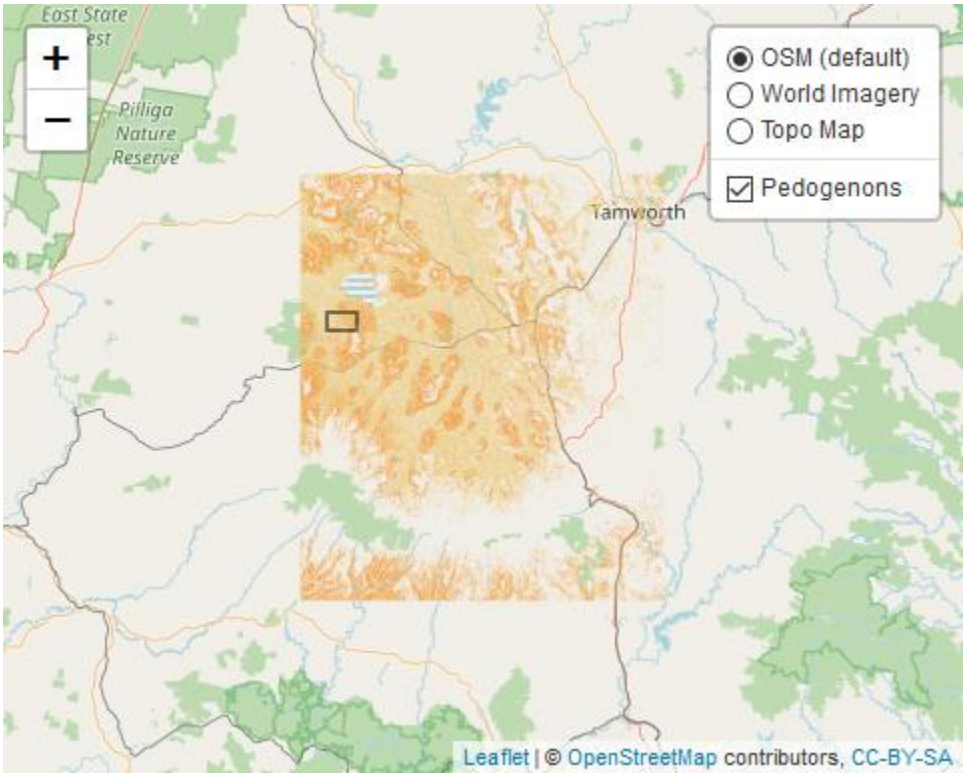
```
### Map the pedogenons present in the study area across the larger area and locate rectangle
around the farm
hc.zone.LARGE.out <- pedogenons.inStudy.area.func(LARGE.Geno.map = pedogenon.3857,
                                                  study.area.Geno.out = hc.zone.out,
                                                  need.proj = FALSE)

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## ellps WGS 84 in CRS definition: +proj=merc +a=6378137 +b=6378137 +lat_ts=0
## +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null +wktext +no_defs

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## datum WGS_1984 in CRS definition

hc.zone.LARGE.out$map.out %>%
  addRectangles(color = "black", weight = 2,
                lng1=150.1576, lat1=-31.36793,
                lng2=150.0732 , lat2=-31.32397,
                fillColor = "transparent" )
```



We may just be interested in dominant classes, for example, larger than 3 km^2. Map the pedogenons present in the farm across the larger study area and locate rectangle around the farm.

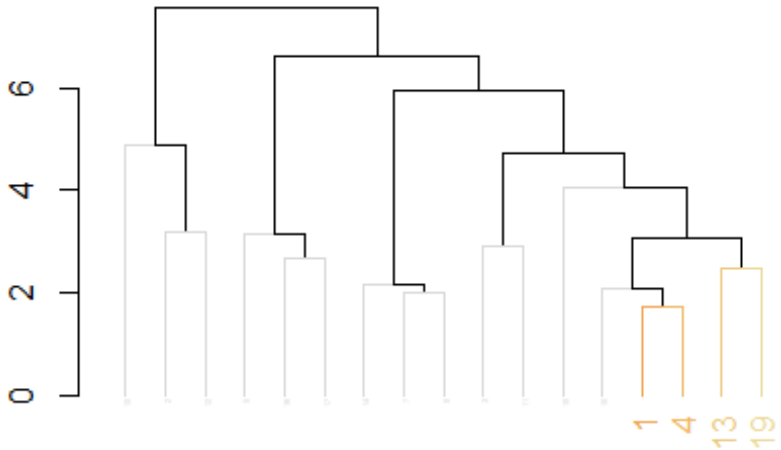
```
hc.zone.whole.out <- pedogenons.inStudy.area.bigger.func(LARGE.Geno.map = pedogenon.3857,
                                                         study.area.Geno.out = hc.zone.out,
                                                         study.area.df = k20.AreaCentrMh.zone,
                                                         min.area = 3,
                                                         need.proj = FALSE)

## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## ellps WGS 84 in CRS definition: +proj=merc +a=6378137 +b=6378137 +lat_ts=0
## +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null +wktext +no_defs
```

```
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO"): Discarded
## datum WGS_1984 in CRS definition

hc.zone.whole.out$map.out %>%
  addRectangles(color = "black", weight = 2,
                lng1=150.1576, lat1=-31.36793,
                lng2=150.0732 , lat2=-31.32397,
                fillColor = "transparent" )
```



Plot the dendrogram, only with the classes colored in the map.

```
plot(hc.zone.whole.out$dendro.larger.pedogenons)
```


0
2
4
6
1
4
13
19