# SOMSerraDourada

Paper: Unveiling Geological Complexity in the Serra Dourada Granite Using Self-Organizing Maps and Hierarchical Clustering: Insights for REE Prospecting in the Goiás Tin Province, Central Brazil

Authors: Ferreira da Silva, G.; Ferreira, M.V., Chemale, L.T.; Santana, I.V.; Botelho, N.F. 2024.

Abstract: This study explores the use of Self-Organizing Maps (SOM) combined with hierarchical clustering to provide insights into the geological differentiation and mineral prospecting in the Serra Dourada Granite (SDG), part of the Goiás Tin Province. After some issues on the geological cartography of the SDG based on traditional approaches, such as the interpretation of outcrops and the limited geochemistry data, often struggle to capture the complexity of high-dimensional geophysical datasets. To address this, we apply unsupervised machine learning techniques to segment airborne geophysical data, providing a more nuanced understanding of the SDG internal structure. Using airborne gamma-ray data, we employed SOM for dimensionality reduction and data segmentation, supported by hierarchical clustering. This methodology enabled us to identify distinct geological units with greater accuracy and resolution than traditional methods such as Principal Component Analysis (PCA). The SOM-based approach retained the data's original topology and revealed fine-scale patterns within the dataset, distinguishing between areas affected by magmatic processes and those influenced by post-magmatic hydrothermalism and supergene leaching. The results indicate that some clusters are mainly associated with magmatic differentiation, characterized by average concentrations of potassium (K), equivalent thorium (eTh), and equivalent uranium (eU) and others show evidence of secondary processes, including hydrothermal alteration and weathering. Notably, Cluster 4 is spatially linked to REE-enriched plateaus and the Serra Verde Mine, reinforcing its significance for mineral exploration. The SOM model proved more effective than PCA at capturing non-linear relationships within the data. While PCA provided insights into the primary variance, it did not fully account for the complex geological processes at play. In contrast, the SOM model segmented the data into clusters that reflected both broad geophysical trends and localized variations, particularly in areas influenced by hydrothermalism and supergene processes. Our findings underscore the value of machine learning techniques, particularly SOM, in geoscientific data analysis. This approach provides a robust framework for integrating multivariate geophysical data, offering valuable insights for geological mapping and mineral exploration, especially in regions with complex geological histories. The methodology presented here can be adapted to other geological settings, enhancing the accuracy of subsurface mapping and identifying areas of economic interest, such as Rare Earth Element (REE) and other critical mineral deposits.

Keywords: Unsupervised Segmentation; Compositional Data Analysis; Clustering Algorithms; Machine Learning Applications in Geoscience; Dimensionality Reduction

```
# Setting up the enviroment
setwd('~/GitHub/GoiasTinProvince/data') # Working direction
set.seed(0) # Random State
```

Predifined Functions

```
# Color Pallette
coolBlueHotRed <- function(n, alpha = 1) {
  rainbow(n, end=4/6, alpha=alpha)[n:1]
}

# Traning SOM for iterations
```

```
training_som <- function(xdim = 10,ydim = 15,rlen = 100,k = 8){
  som_grid <- kohonen::somgrid(xdim = xdim,ydim = ydim,
                               topo = 'hexagonal',
                               toroidal = TRUE,
                               neighbourhood.fct = 'gaussian')
  set.seed(0)
  print(paste('xdim: ',xdim,', ydim: ',ydim,' rlen: ',rlen,' Time: ',system.time(som_model <- kohonen::s



  ))

  mydata <- kohonen::getCodes(som_model) # Assigning data to closest neuron
  cut_avg <- cutree(hclust(dist(mydata), method = 'average'), k = k)

  cluster_assignment <- cut_avg[som_model$unit.classif]

  df$hc_avg <- cluster_assignment

  (p1 <- ggplot(df, aes(x = Longitude,
                        y = Latitude,
                        fill = as.factor(hc_avg))) +
    geom_raster() +
    coord_equal() +
    scale_fill_viridis_d(option = 'B') +
    theme_classic() +
    labs(title = paste0('xdim: ',xdim,', ydim: ',ydim,', rlen: ',rlen,' and k: ',k),
         fill = 'Cluster') +
    theme(axis.text.x = element_text(angle = 90,
                                     vjust = .5))
    )
  ggsave(filename = paste('figure/som',xdim,ydim,rlen,'.pdf',sep = '_'),width = 8,height = 8,device = '
}

# Radar plot
coord_radar <- function (theta = "x", start = 0, direction = 1) {
  theta <- match.arg(theta, c("x", "y"))
  r <- if (theta == "x") "y" else "x"
  ggproto("CordRadar", CoordPolar, theta = theta, r = r, start = start,
          direction = sign(direction),
          is_linear = function(coord) TRUE)
}
```

Importing libraries

```
library(tidyverse) # ggplot, dplyr, readr, tibble, readr
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(factoextra) # Cluster Vis and PCA
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(geoquimica) # Data transformation
library(kohonen) # Self Organizing Maps
```

```
##
## Attaching package: 'kohonen'
##
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(doParallel) # Parallel Computing
```

```
## Carregando pacotes exigidos: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## Carregando pacotes exigidos: iterators
## Carregando pacotes exigidos: parallel
```

```
library(foreach) # For loops
library(circlize) # Data vis
```

```
## ========================================
## circlize version 0.4.16
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```r
library(ggpubr) # Data vis
library(dendextend) # Dendograms and HClustering
```

```
##
## ---------------------
## Welcome to dendextend version 1.17.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
##
##
## Attaching package: 'dendextend'
##
## The following object is masked from 'package:ggpubr':
##
##     rotate
##
## The following object is masked from 'package:stats':
##
##     cutree
```

```r
library(pheatmap) # Dissimilarity matrix
```

Importing and previewing data:

```r
df_raw <- readRDS('~/GitHub/GoiasTinProvince/data/gamma_GSD.RDS') %>%
  filter(!is.na(X)) %>%
  select(c('X','Y','KPERC','eTH','eU','CT')) %>%
  rename(Longitude = X,
         Latitude = Y,
         TC = CT,
         K = KPERC) %>%
  mutate(Ki = eTH*(mean(K, na.rm = TRUE)/mean(eTH, na.rm = TRUE)),
         Ui = eTH*mean(eU, na.rm = TRUE)/mean(eTH, na.rm = TRUE)) %>%
  mutate(Kd = K - Ki,
         Ud = eU - Ui,
         K_TH = K/eTH
  )

head(df_raw)
```

```
##   Longitude  Latitude        K      eTH       eU       TC       Ki       Ui
## 1 -48.49275 -13.57551 1.585320 37.14201 3.180177 15.16076 1.885156 2.802171
## 2 -48.49152 -13.57551 1.639656 39.07323 3.394328 15.97142 1.983176 2.947871
```

```
## 3 -48.49029 -13.57551 1.697323 40.15634 3.482608 16.42931 2.038149 3.029586
## 4 -48.48906 -13.57551 1.747037 40.76690 3.468431 16.66500 2.069138 3.075649
## 5 -48.48783 -13.57551 1.775707 41.19197 3.378610 16.76237 2.090713 3.107719
## 6 -48.48660 -13.57551 1.774680 41.54077 3.233844 16.75653 2.108416 3.134034
##           Kd         Ud       K_TH
## 1 -0.2998354 0.37800656 0.04268267
## 2 -0.3435193 0.44645693 0.04196367
## 3 -0.3408263 0.45302194 0.04226787
## 4 -0.3221012 0.39278219 0.04285430
## 5 -0.3150061 0.27089136 0.04310808
## 6 -0.3337361 0.09981013 0.04272141
```

Data Preparation:

```
# Processing ----

# Min-max feature scaling
df_norm <- df_raw %>%
  elem_norm(method = 'minmax',
            keep = c('Longitude', 'Latitude')) %>%
  select(-Ki,-Ui,-TC)

summary(df_norm)
```

```
##    Longitude         Latitude            K               eTH
##  Min.   :-48.58   Min.   :-13.58   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:-48.53   1st Qu.:-13.48   1st Qu.:0.2789   1st Qu.:0.1762
##  Median :-48.51   Median :-13.34   Median :0.4022   Median :0.2626
##  Mean   :-48.51   Mean   :-13.34   Mean   :0.4128   Mean   :0.2769
##  3rd Qu.:-48.49   3rd Qu.:-13.21   3rd Qu.:0.5411   3rd Qu.:0.3405
##  Max.   :-48.43   Max.   :-13.07   Max.   :1.0000   Max.   :1.0000
##        eU               Kd               Ud             K_TH
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2090   1st Qu.:0.6451   1st Qu.:0.6111   1st Qu.:0.1102
##  Median :0.2846   Median :0.7208   Median :0.6711   Median :0.1608
##  Mean   :0.3053   Mean   :0.7030   Mean   :0.6654   Mean   :0.1786
##  3rd Qu.:0.3828   3rd Qu.:0.7766   3rd Qu.:0.7272   3rd Qu.:0.2219
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
# CLR transformation
df_clr <- df_raw %>%
  elem_norm(method = 'clr',
            keep = c('Longitude', 'Latitude','Kd','Ud')) %>%
  elem_norm(method = 'minmax', keep = c('Longitude', 'Latitude')) %>%
  dplyr::select(-Ki,-Ui,-TC)

summary(df_clr)
```

```
##    Longitude         Latitude            K               eTH
##  Min.   :-48.58   Min.   :-13.58   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:-48.53   1st Qu.:-13.48   1st Qu.:0.5902   1st Qu.:0.5603
##  Median :-48.51   Median :-13.34   Median :0.7009   Median :0.6567
##  Mean   :-48.51   Mean   :-13.34   Mean   :0.6827   Mean   :0.6317
```

```
##  3rd Qu.:-48.49   3rd Qu.:-13.21   3rd Qu.:0.7953   3rd Qu.:0.7214
##  Max.   :-48.43   Max.   :-13.07   Max.   :1.0000   Max.   :1.0000
##        eU               Kd               Ud             K_TH
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.6916   1st Qu.:0.6451   1st Qu.:0.6111   1st Qu.:0.4570
##  Median :0.7518   Median :0.7208   Median :0.6711   Median :0.5424
##  Mean   :0.7451   Mean   :0.7030   Mean   :0.6654   Mean   :0.5385
##  3rd Qu.:0.8099   3rd Qu.:0.7766   3rd Qu.:0.7272   3rd Qu.:0.6186
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Principal Components Analysis

```
# PCA
pca <- prcomp(x = df_clr[3:8],center = TRUE,scale. = TRUE)

summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4   PC5       PC6
## Standard deviation     1.8251 1.2498 0.9913 0.31667 0.155 3.652e-15
## Proportion of Variance 0.5552 0.2603 0.1638 0.01671 0.004 0.000e+00
## Cumulative Proportion  0.5552 0.8155 0.9793 0.99600 1.000 1.000e+00
```

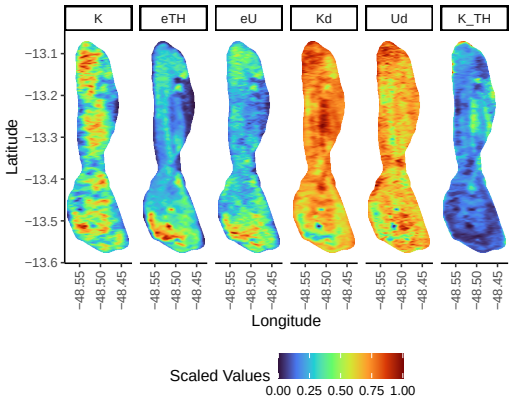
```
df <- bind_cols(df_clr, as_tibble(pca$x[,1:6])) # apending results
```

View Features:

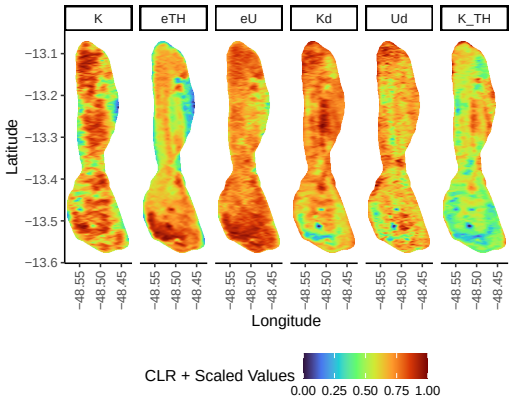
```
# Plot Feature Maps ----
## Min-Max
raw.plot <- ggplot(df_norm %>%
                     gather(key = 'Variables',value = 'Value', 3:8) %>%
                     mutate(Variables = factor(Variables,levels = c('K',
                                                                    'eTH',
                                                                    'eU',
                                                                    'Kd',
                                                                    'Ud',
                                                                    'K_TH'
                     ))),
                   aes(x = Longitude, y = Latitude, fill = Value)) +
  geom_raster() +
  coord_equal() +
  scale_fill_gradientn(colors = pals::turbo(8)) +
  facet_wrap(. ~ Variables,nrow = 1) +
  theme_classic() +
  theme(axis.text.x = element_text(
    angle = 90,
    vjust = 0.5),
    legend.direction = 'horizontal',
    legend.position = 'bottom')  +
  labs(fill = 'Scaled Values')

## CLR
```

```
clr.plot <- ggplot(df_clr %>%
                     gather(key = 'Variables',value = 'Value', 3:8) %>%
                     mutate(Variables = factor(Variables,levels = c('K',
                                                                    'eTH',
                                                                    'eU',
                                                                    'Kd',
                                                                    'Ud',
                                                                    'K_TH'
                     ))),
                   aes(x = Longitude, y = Latitude, fill = Value)) +
    geom_raster() +
    coord_equal() +
    scale_fill_gradientn(colors = pals::turbo(8)) +
    facet_wrap(. ~ Variables,nrow = 1) +
    theme_classic() +
    theme(axis.text.x = element_text(
      angle = 90,
      vjust = 0.5),
      legend.direction = 'horizontal',
      legend.position = 'bottom')   +
    labs(fill = 'CLR + Scaled Values')

## Principal Components maps
pca.plot <- ggplot(df %>%
                     geoquimica::elem_norm(keep = c('Longitude', 'Latitude')) %>%
                     gather(key = 'Components',value = 'PCA', 9:14),
                   aes(x = Longitude, y = Latitude, fill = PCA)) +
    geom_raster() +
    coord_equal() +
    scale_fill_gradientn(colors = pals::turbo(8)) +
    # scale_fill_viridis_c() +
    facet_wrap(. ~ Components,nrow = 1) +
    theme_classic() +
    theme(axis.text.x = element_text(angle = 90,vjust = 0.5),
          legend.direction = 'horizontal',legend.position = 'bottom')

ggarrange(raw.plot,clr.plot,pca.plot,ncol = 1,labels = c('a)','b)','c)'))
```

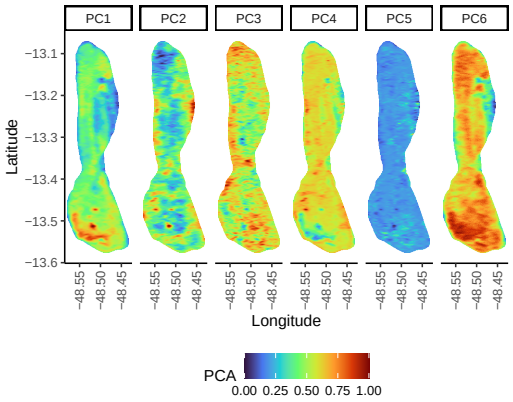
**a)**



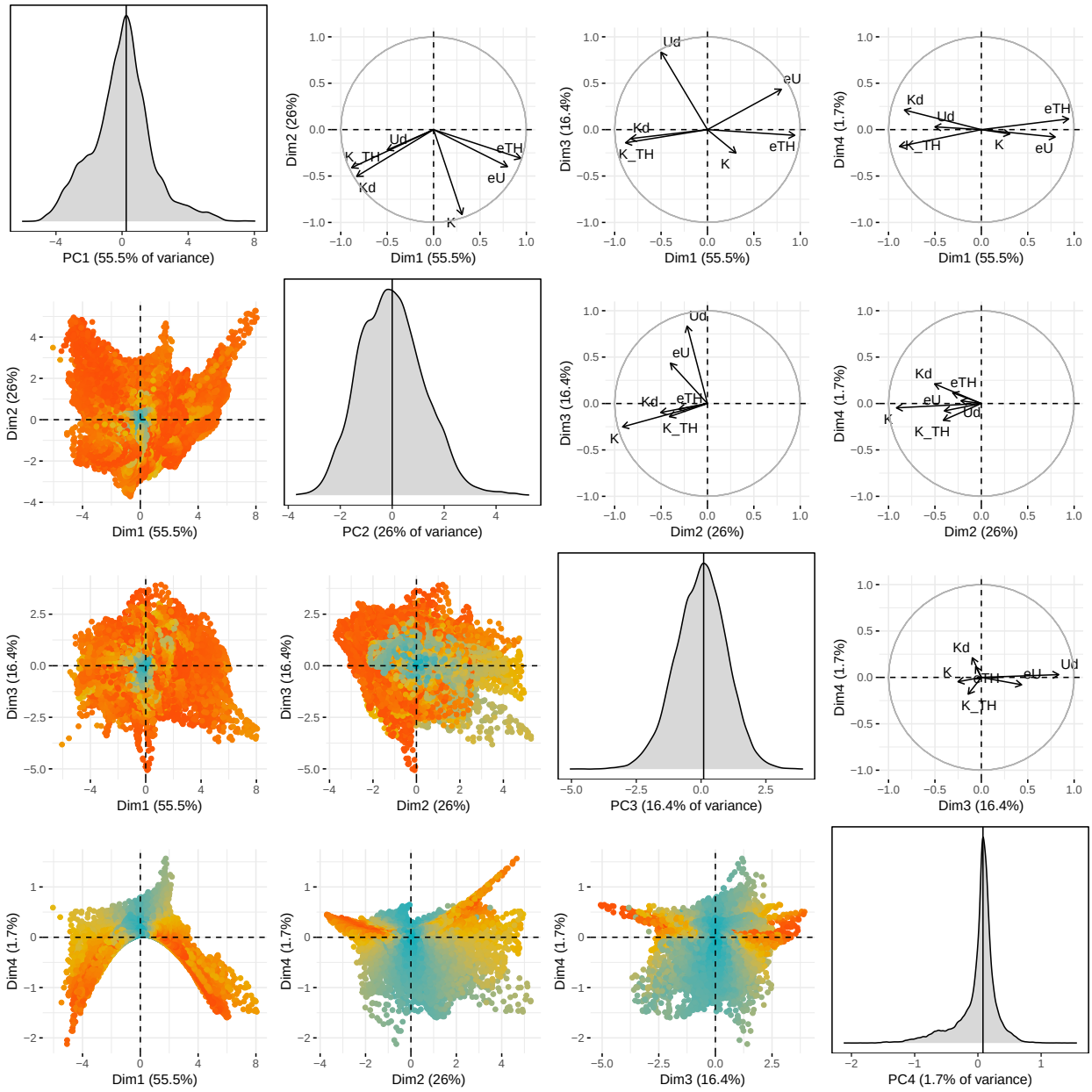

**b)**




**c)**




View PCA Matrix




Training Self Organizing Maps

```
# SOM

data_train <- as.matrix(df[,c('K',
                              'eTH',
                              'eU',
                              'Kd',
                              'Ud',
                              'K_TH'
)])

xdim <- c(15,24,30)
ydim <- c(20,32,40)
rlen <- c(10,100,300,500,1000)
```

In this session we iterated between grid arrange and rlen on the SOM training:

```
# registerDoParallel(cores = parallel::detectCores()-1)
# foreach(xdim = xdim) %do% {
#   foreach(ydim = ydim) %do% {
#     foreach(rlen = rlen) %do% {
#       training_som(xdim = xdim,ydim = ydim,rlen = rlen)
#     }
#   }
# }
#
# plot(p1)
```

Training Selected Model

```
# ## Defining the configuration of Neurons
som_grid <- kohonen::somgrid(xdim = 24,ydim = 32,
                             topo = 'hexagonal',
                             toroidal = TRUE,
                             neighbourhood.fct = 'gaussian')
# ## Training Kohonen Maps
set.seed(0)
som_model <-
  kohonen::som(data_train,
                                      grid = som_grid,
                                      rlen = 300,
                                      keep.data = TRUE)

summary(som_model) # Summary
```

```
## SOM of size 24x32 with a hexagonaltoroidal topology and a gaussian neighbourhood function.
## The number of data layers is 1.
## Distance measure(s) used: sumofsquares.
## Training data included: 29675 objects.
## Mean distance to the closest unit in the map: 0.002.
```

View changes during different epochs for SOM model

```
plot(som_model, type = 'changes') # Average distance of neurons by epochs
```


Hierarchical Clustering

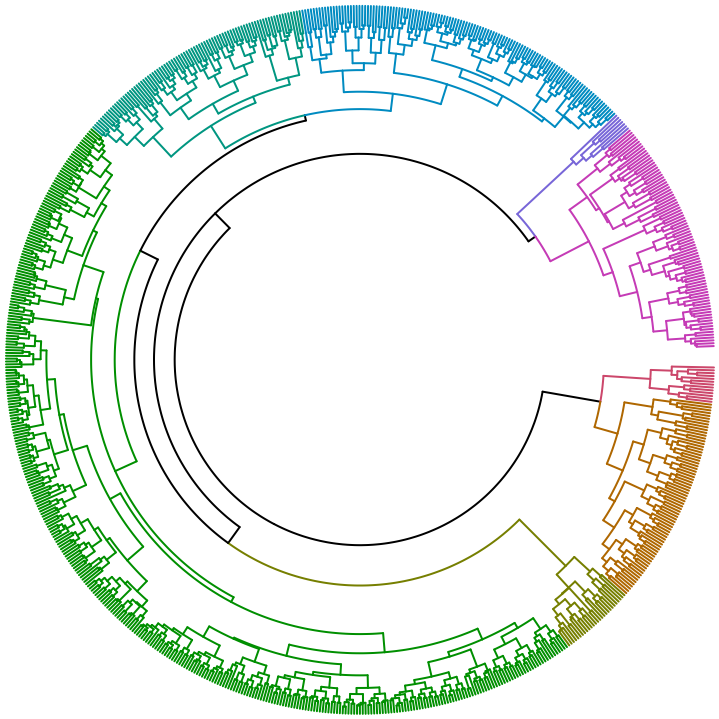
```
mydata <- kohonen::getCodes(som_model) # Assigning data to closest neuron

dist_mat <- dist(mydata,method = 'euclidean') # Building distance matrix with euclidean distances

hclust_avg <- hclust(dist_mat, method = 'average')

cut_avg <- cutree(hclust_avg, k = 8) # determining the number of clusters
# get vector with cluster value for each original data sample
cluster_assignment <- cut_avg[som_model$unit.classif]
# for each of analysis, add the assignment as a column in the original data:
df$hc_avg <- cluster_assignment

glimpse(df)
```

```
## Rows: 29,675
## Columns: 15
## $ Longitude <dbl> -48.49275, -48.49152, -48.49029, -48.48906, -48.48783, -48.4~
## $ Latitude  <dbl> -13.57551, -13.57551, -13.57551, -13.57551, -13.57551, -13.5~
## $ K         <dbl> 0.5758427, 0.5881451, 0.6007633, 0.6113020, 0.6172441, 0.617~
## $ eTH       <dbl> 0.6179474, 0.6316509, 0.6390428, 0.6431223, 0.6459266, 0.648~
## $ eU        <dbl> 0.7524258, 0.7654732, 0.7706137, 0.7697970, 0.7645439, 0.755~
## $ Kd        <dbl> 0.6709311, 0.6662524, 0.6665408, 0.6685463, 0.6693063, 0.667~
## $ Ud        <dbl> 0.7066949, 0.7141806, 0.7148985, 0.7083107, 0.6949809, 0.676~
## $ K_TH      <dbl> 0.4738289, 0.4692837, 0.4712161, 0.4749025, 0.4764822, 0.474~
```

```
## $ PC1       <dbl> 0.13177958, 0.26887556, 0.32169438, 0.33930403, 0.35793598, ~
## $ PC2       <dbl> 0.7483693, 0.6354386, 0.5350726, 0.4726522, 0.4720059, 0.544~
## $ PC3       <dbl> 0.66950851, 0.77414668, 0.77647019, 0.69232151, 0.54398430, ~
## $ PC4       <dbl> 0.2070352, 0.1931986, 0.1776903, 0.1674058, 0.1669094, 0.178~
## $ PC5       <dbl> -0.0264645970, -0.0163788120, -0.0082897127, -0.0054216179, ~
## $ PC6       <dbl> -1.498801e-15, -1.387779e-15, -6.106227e-16, -1.110223e-15, ~
## $ hc_avg    <int> 5, 5, 5, 5, 5, 5, 1, 1, 3, 3, 3, 5, 5, 5, 1, 1, 1, 1, 3, 3, ~
```

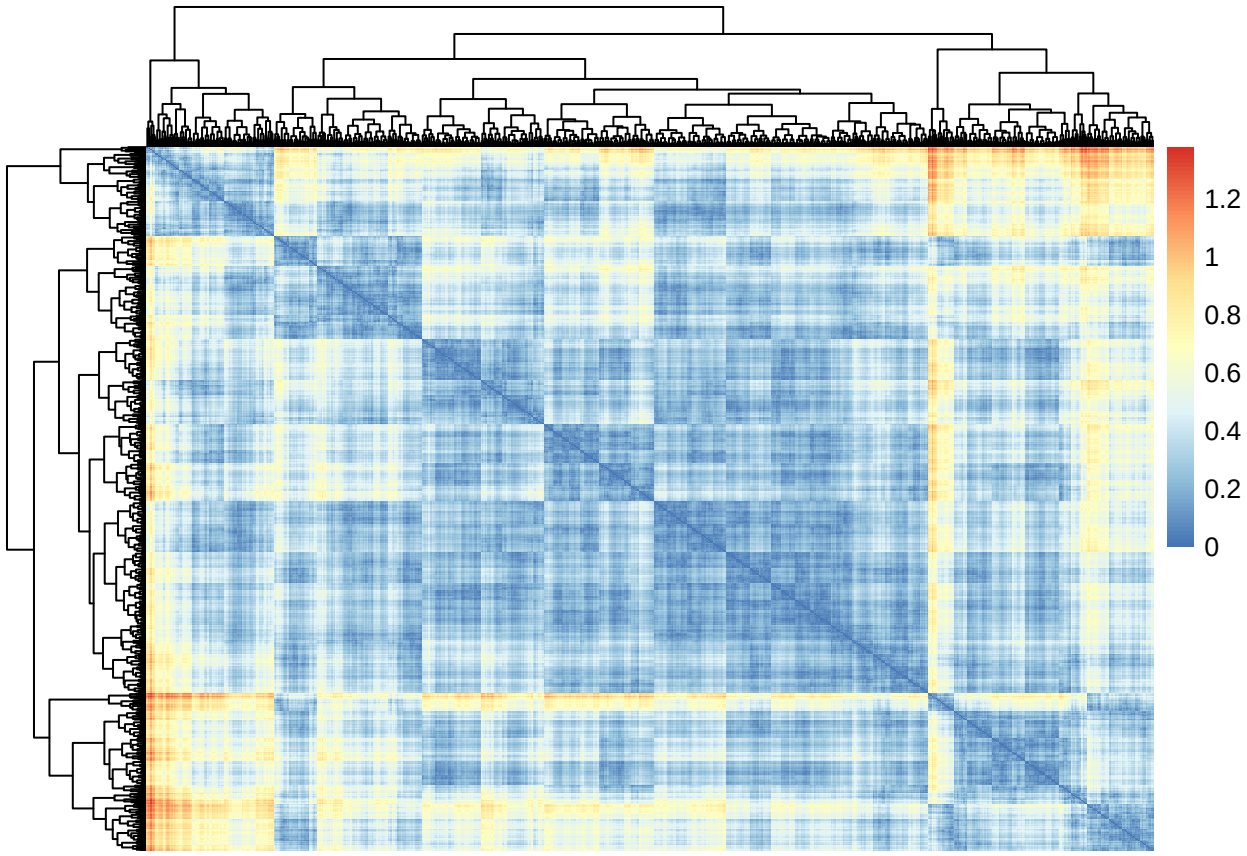
Plotting the Dendogram

```
avg_dend_obj <- as.dendrogram(hclust_avg)

avg_col_dend <- color_branches(avg_dend_obj,h=.325)

circlize_dendrogram(avg_col_dend,
                    labels_track_height = NA,
                    dend_track_height = 0.5,labels = FALSE)
```



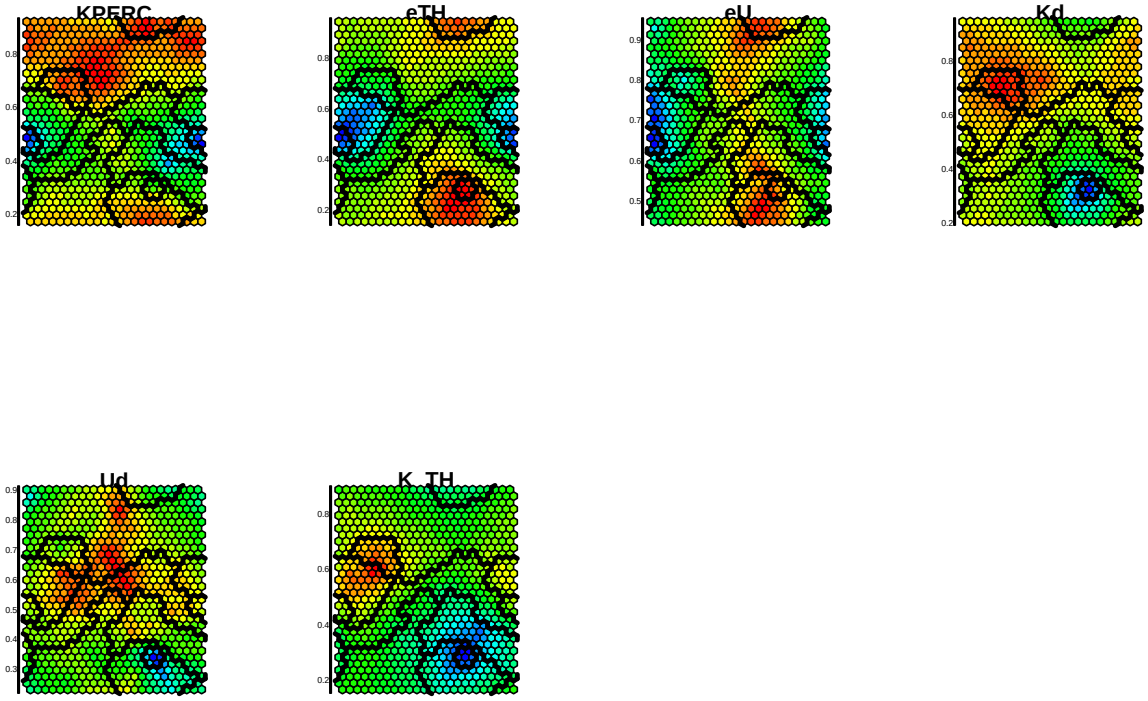
Plotting the Dissimilarity Matrix

```
pheatmap(as.matrix(dist_mat),
         clustering_distance_rows = dist_mat,
         clustering_distance_cols = dist_mat,
         show_rownames = FALSE,
         show_colnames = FALSE)
```



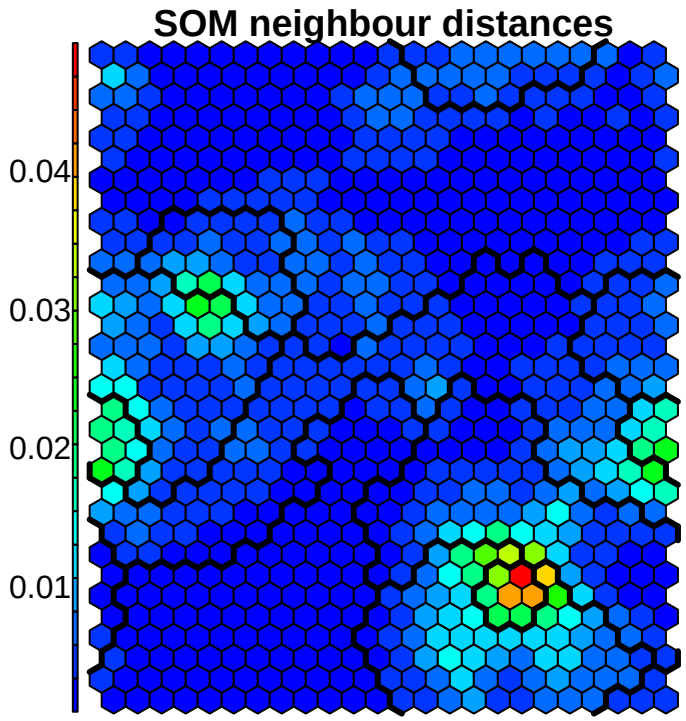

Clustered Kohonen Maps for each feature

```
nome <- c('KPERC','eTH','eU','Kd','Ud','K_TH')

par(mfrow=c(2,4), mai = c(5, 5, 5, 5)
)
for (j in 1:6){
  plot(som_model,type="property",
       shape = 'straight',
       property = getCodes(som_model)[,j],
       palette.name=coolBlueHotRed,
       main=nome[j],
       cex=0.5)
  add.cluster.boundaries(som_model, lwd = 3,
                         cut_avg)
}
par(mfrow=c(1,1))
```



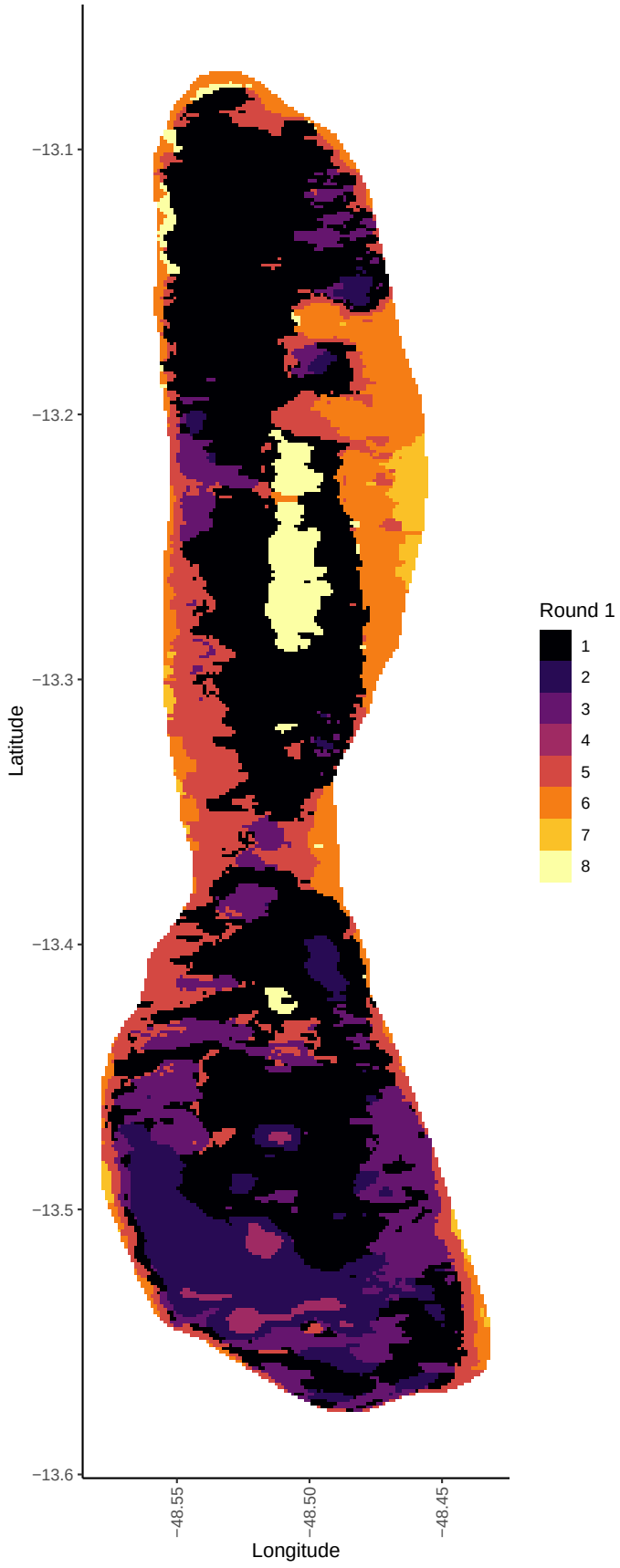

Distance plot for Kohonen Maps

```
plot(som_model, type="dist.neighbours",
     main = "SOM neighbour distances",
     palette.name=coolBlueHotRed, shape = 'straight'
)
add.cluster.boundaries(som_model, lwd = 3,
                       cut_avg)
```




Map of Clustered Serra Dourada Granite

```
# DATA VIS ----
## Mapa de Cluster
ggplot(df, aes(x = Longitude,
               y = Latitude,
               fill = as.factor(hc_avg))) +
  geom_raster() +
  coord_equal() +
  scale_fill_viridis_d(option = 'B') +
  theme_classic() +
  labs(fill = 'Round 1') +
  theme(axis.text.x = element_text(angle = 90,
                                   vjust = .5))
```


Round 1
1
2
3
4
5
6
7
8
Latitude
−13.1
−13.2
−13.3
−13.4
−13.5
−13.6
−48.55
−48.50
−48.45
Longitude

Radar

plot for each Cluster

```
df %>%
  group_by(hc_avg) %>%
  summarize(KPERC = median(K),
            eTH = median(eTH),
            eU = median(eU),
            Kd = median(Kd),
            Ud = median(Ud),
            K_TH = median(K_TH)) %>%
  ungroup() %>%
  gather(value = "value", key = 'key',2:7) %>%
  ggplot(aes(x = key, y = value, col = as.factor(hc_avg),
             group = hc_avg)) +
  geom_line() +
  geom_point() +
  coord_radar() +
  scale_color_viridis_d(option = 'B') +
  scale_fill_viridis_d(option = 'B') +
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.y = element_blank()) +
  facet_wrap(. ~ hc_avg)
```