03-Penkrot et al., 2018 Geosphere U1419 general cluster analysis

Michelle Penkrot

3/19/2018

## Code Information

This code will import physical property data and normalized scanning XRF elemental concentrations (NMS normalized; Lyle et al., 2012) from Integrated Ocean Drilling Site U1419, and then perform both mixture-model clustering and heirarchical clustering on these data. See here for more details about this drilling location: <http://iodp.tamu.edu/scienceops/expeditions/alaska_tectonics_climate.html>.

This file will produce 2-3 principle components and one to five clusters, which is the general model output. The user should be aware that running the full model (this code) takes considerable computing time. It is optimized to use parallel processing on four cores. Running the mclust routine for five clusters with two and three principle components can take up to 24 hours.

Note that the cluster number (e.g., Cluster 1, Cluster 2, etc.) produced by he code is arbitrary, so we renamed the clusters post-modeling in rank order so that Cluster 1 indicates the most common cluster and Cluster 5 the least common.

The mclust portion of the following code is modified from R scripts that were originally written and kindly supplied by Karl Ellefsen (2015, personal communication). See Ellefsen, K. J., Smith, D. B., & Horton, J. D. (2014). A modified procedure for mixture-model clustering of regional geochemical data. Applied Geochemistry, 51, 315–326. <https://doi.org/10.1016/j.apgeochem.2014.10.011>

## Load Packages

library(knitr)  
library(rgr)  
library(ggplot2)  
library(psych)  
library(parallel)  
library(rrcov)  
library(mclust)  
library(lsr)  
library(parallel)

## Create functions

CalcSampleClusters <- function( theData, nPDFs, sampleSize, sampleSpace, nIter )  
{  
 require( mclust, quiet=TRUE )  
   
 theLogLikelihoods <- vector( mode="numeric" )  
 nErrors <- 0  
 for( i in 1:nIter ) {  
 S <- sample( sampleSpace, size = sampleSize )  
   
 clusterResult <- tryCatch( Mclust( theData, nPDFs, modelNames=c("VVV"), initialization=list(subset=S) ), error=function(e){ e } )  
   
 if( inherits( clusterResult, "error" ) ) {  
 nErrors <- nErrors + 1  
 } else {  
 theLogLikelihoods <- append( theLogLikelihoods, clusterResult$loglik )  
 if( max( theLogLikelihoods ) == clusterResult$loglik ) {  
 bestClusterResult <- clusterResult  
 }  
 if( min( theLogLikelihoods ) == clusterResult$loglik ) {  
 worstClusterResult <- clusterResult  
 }  
 }  
 }  
   
 return( list( theLogLikelihoods=theLogLikelihoods, bestClusterResult=bestClusterResult,  
 worstClusterResult=worstClusterResult, nErrors=nErrors ) )  
}

## Load Data

Loads the data from a csv file. Separates data based on type (i.e. physical property or scanning XRF elemental). The physical property data are centered and scaled (z-score), and an isometric log ratio transformation (ILR) is performed on the elemental data to open them.

NOTE: See comments below to set the code to run on the full mud-diamict dataset. It is set here to run only on the diamict-only portion of the data.

data<-read.csv("../raw\_data/2018-03-20\_U1419-Penkrot\_Geosphere-2018-data.csv")  
data<-data[509:6490,] #cuts the data down to the diamict-only portion of the core; comment this line if you want to run the analyses on the full mud-diamict data set.  
#lith<-data$Mud\_Diamict\_code  
lith<-data$Diamict\_only\_code #comment this line and uncomment previous line ifyou want to run the analyses on the full mud-diamict data set.  
depth<-data$CCFS\_A  
  
physprops<-data[c("b\_star","NGR","MS")]  
elements<-data[c("Al","Ca","Rb","Zr","K","Si")]  
  
physprops\_scaled<-scale(physprops, center=TRUE,scale=TRUE) # Centers and scales the physical property data  
elements\_ilr<-ilr(as.matrix(elements)) # Performs isometric log ratio transformation to open XRF data

## \*\* Are the data all in the same measurement units? \*\*

## Select Data for Models

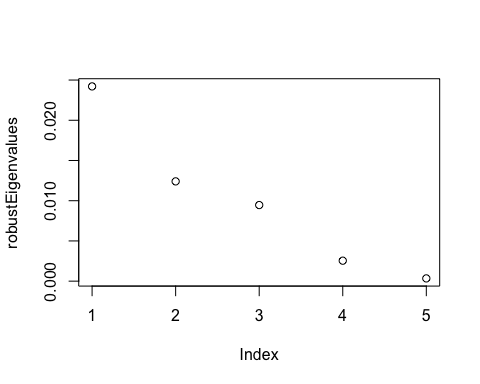
Select which data to include in cluster analysis. This study ran the model 3 times, with physical property and scanning XRF elemental data inputs (run name: G1), only physical property inputs (run name: G2) and only scanning XRF data (run name G3). The optimal data set was G3 and is used in the paper.

#run.name<-"G1" # Change for each cluster model run  
#run.name<-"G2" # Change for each cluster model run  
run.name<-"G3" # Change for each cluster model run  
  
#data\_transformed<-as.matrix(cbind(physprops\_scaled,elements\_ilr)) #phys props & scanning XRF data; G1  
#data\_transformed<-as.matrix(cbind(physprops\_scaled)) #phys props only; G2  
data\_transformed<-as.matrix(cbind(elements\_ilr)) #scanning XRF only; G3

## Data Processing

Performs a robust principle component analysis on the input data to remove outliers. These principle component results are used in the cluster analysis rather than the raw data. The alpha value lets you select which percentage of the data to be excluded as outliers. Follows this method: Martin Maechler, Peter Rousseeuw, Christophe Croux, Valentin Todorov, Andreas Ruckstuhl, Matias Salibian-Barrera, Tobias Verbeke, Manuel Koller, Eduardo L. T. Conceicao and Maria Anna di Palma (2016). robustbase: Basic Robust Statistics R package version 0.92-7. URL <http://CRAN.R-project.org/package=robustbase>

# select alpha(s): 0.98, 0.96, and 0.92, which will exclude 2%, 4%, and 8% of the data as outliers, respectively  
alpha <- c( 0.98)  
outputDirectories <- c( "../produced\_data/") # Change to appropriate output directory  
  
 mcdResult <- covMcd( data\_transformed, alpha=alpha )  
 robustIlrCenter <- mcdResult$center  
 robustIlrCov <- mcdResult$cov  
 centeredIlrCoefs <- data\_transformed - rep.int( 1, nrow( data\_transformed ) ) %o% robustIlrCenter  
 svdResult <- svd( robustIlrCov )  
 robustEigenvectors <- svdResult$u  
 robustEigenvalues <- svdResult$d # namely, robust variances  
 robustPCs <- centeredIlrCoefs %\*% robustEigenvectors  
 plot(robustEigenvalues) # Scree plot



save(robustIlrCenter, robustEigenvectors, robustEigenvalues, robustPCs,  
 file=paste( outputDirectories,run.name,"\_PrinComp.dat", sep="" ) )

## Model-based clustering (MClust)

Performs model clustering on robust principle component results. The number of principles components (nPCs) can be changed in line 121 and the number of clusters (nPDFs) can be changed in line 123. This study ran the model cluster analysis for both 2 & 3 PCs and 2-5 clusters.

nWorkers <- 4 # should be <= 4 on my machine, i.e., number of processor cores  
cl <- makeCluster( nWorkers )  
clusterSetRNGStream( cl, 123 )  
  
outputDirectories <- c("../produced\_data/") #Change to appropriate output directory  
   
 for( j in 1:length(outputDirectories) ) {  
 #for( j in 1:1 ) {  
   
 load( paste( outputDirectories[j],run.name,"\_PrinComp.dat", sep="" ) )  
   
 nRows <- nrow( robustPCs )  
 sampleSize <- as.integer( 0.75 \* nRows )  
 sampleSpace <- 1:nRows  
   
 for( nPCs in 3:3) { #uses 2 & 3 principle components  
   
 for( nPDFs in 5:5 ) { #creates 2-5 clusters  
   
 clusterOutputDirectory <- paste( outputDirectories[j], run.name,"\_modelclust\_",nPCs, "-PCs\_\_", nPDFs, "-PDFs/", sep="" )  
 dir.create(clusterOutputDirectory)  
   
 nIter <- 100 + ( nPDFs - 2 ) \* 150 + ( nPCs - 4 ) \* 30  
 nIterPerWorker <- as.integer( nIter / nWorkers )  
 cat( sprintf( "No. of principal components: %3d No. of pdfs: %3d No. of iter: %3d No. of iter per worker: %3d\n",  
 nPCs, nPDFs, nIter, nIterPerWorker ) )  
   
 tmpResult <-clusterCall( cl, CalcSampleClusters, robustPCs[,1:nPCs], nPDFs, sampleSize, sampleSpace, nIterPerWorker )  
   
 theLogLikelihoods <- tmpResult[[1]]$theLogLikelihoods  
 bestClusterResult <- tmpResult[[1]]$bestClusterResult  
 worstClusterResult <- tmpResult[[1]]$worstClusterResult  
 nErrors <- tmpResult[[1]]$nErrors  
 for( i in 2:nWorkers ) {  
   
 theLogLikelihoods <- append( theLogLikelihoods, tmpResult[[i]]$theLogLikelihoods )  
   
 if( bestClusterResult$loglik < tmpResult[[i]]$bestClusterResult$loglik )  
 bestClusterResult <- tmpResult[[i]]$bestClusterResult  
   
 if( worstClusterResult$loglik > tmpResult[[i]]$worstClusterResult$loglik )  
 worstClusterResult <- tmpResult[[i]]$worstClusterResult  
   
 nErrors <- nErrors + tmpResult[[i]]$nErrors  
 }  
   
 save( theLogLikelihoods, bestClusterResult, worstClusterResult, nErrors,  
 file=paste( clusterOutputDirectory,"data.dat", sep="" ) )  
   
 cat( sprintf( "No. of errors: %3d\n", nErrors ) )  
 }  
 }  
 }  
   
 stopCluster( cl )

### Combines model clustering results into one matrix.

pc<-c(2,2,2,2,3,3,3,3)  
pdf<-c(2,3,4,5,2,3,4,5)  
  
mclust\_results<-as.data.frame(matrix(0,nrow(data\_transformed),length(pc)))  
for(i in 1:length(pc)){  
load(paste("../produced\_data/",run.name,"\_modelclust\_",pc[1],"-PCs\_\_",pdf[1],"-PDFs/data.dat",sep=""))   
bestclusterresult<-bestClusterResult$classification  
mclust\_results[,i]<-bestclusterresult  
colnames(mclust\_results)[i]<-paste("mclust\_",pc[i],"PCs\_",pdf[i],"Clusters",sep="")  
}

## Warning in readChar(con, 5L, useBytes = TRUE): cannot open compressed file  
## '../produced\_data/G3\_modelclust\_2-PCs\_\_2-PDFs/data.dat', probable reason  
## 'No such file or directory'

## Error in readChar(con, 5L, useBytes = TRUE): cannot open the connection

load(paste("../produced\_data/G3\_modelclust\_3-PCs\_\_5-PDFs/data.dat",sep=""))   
bestclusterresult<-bestClusterResult$classification  
mclust\_results<-bestclusterresult

## Hierarchical-based clustering (hclust)

Performs Hierarchical cluster analysis using the hclust function from the R stats package. Hierarchical clustering produces a dendrogram based on similariities of the downcore data. The dendrogram is cut off at heights that produces 2-5 clusters.

distance2<-dist(as.data.frame(robustPCs[,1:2]),method="euclidean") # uses 2 principle components  
MyTree2<-hclust(distance2, method='ward.D2')  
Cut2PC\_2CL<-as.data.frame(cutree(MyTree2,k=2)) # cuts hclust tree at 2 clusters  
Cut2PC\_3CL<-as.data.frame(cutree(MyTree2, k=3))  
Cut2PC\_4CL<-as.data.frame(cutree(MyTree2, k=4))  
Cut2PC\_5CL<-as.data.frame(cutree(MyTree2, k=5)) # cuts hclust tree at 5 clusters  
   
distance3<-dist(as.data.frame(robustPCs[,1:3]),method="euclidean") # uses 3 principle components  
MyTree3<-hclust(distance3, method='ward.D2')  
Cut3PC\_2CL<-as.data.frame(cutree(MyTree3,k=2)) # cuts hclust tree at 2 clusters  
Cut3PC\_3CL<-as.data.frame(cutree(MyTree3, k=3))  
Cut3PC\_4CL<-as.data.frame(cutree(MyTree3, k=4))  
Cut3PC\_5CL<-as.data.frame(cutree(MyTree3, k=5)) # cuts hclust tree at 5 clusters  
   
hclust\_results<-cbind(Cut2PC\_2CL,Cut2PC\_3CL,Cut2PC\_4CL,Cut2PC\_5CL,Cut3PC\_2CL,Cut3PC\_3CL,Cut3PC\_4CL,Cut3PC\_5CL)  
colnames(hclust\_results)<-c("hclust\_2PCs-2Clusters","hclust\_2PCs-3Clusters","hclust\_2PCs-4Clusters","hclust\_2PCs-5Clusters","hclust\_3PCs-2Clusters","hclust\_3PCs-3Clusters","hclust\_3PCs-4Clusters","hclust\_3PCs-5Clusters")  
  
hclust\_results<-(Cut3PC\_5CL)  
colnames(hclust\_results)<-c("hclust\_3PCs-5Clusters")

## Statistical Analysis of Results

Statistical parameters used to validate cluster results through comparison with downcore changes in observed lithofacies. Chi-squared test, Cramer's V-value, F-measure and Rand Index are calculated. This study only discusses the Chi-squared test and Cramer's V-value results.

results<-cbind(depth,mclust\_results,hclust\_results)  
r<-length(results)-1   
c<-4  
statistics<-matrix(0,r,c) # r=number of models, c=number of statistical tests  
for (i in 1:r){  
 q<-results[,i+1]  
 tbl<-table(lith,q)  
 chi<-chisq.test(tbl)  
 cv<-cramersV(tbl)  
 x<-chi$statistic  
 y<-chi$parameter  
 z<-chi$p.value  
 statistics[i,]<-cbind(x,y,z,cv)  
}

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

## Warning in chisq.test(...): Chi-squared approximation may be incorrect

colnames(statistics)<-c("X-square","Df","p-value","Cramers V Value")  
nCols <- ncol(results)  
rownames(statistics)<-colnames(results[,2:nCols])

## Save Model Output

Saves clustering results from both model and hierarchical clustering for 2-3 PCs and 2-5 clusters, and statistical parameter results as .csv files.

write.csv(results,file=paste("../produced\_data/",toString(run.name),"\_clusteringresults.csv",sep="")) # Saves Hierarchical and model clustering results  
write.csv(statistics,file=paste("../produced\_data/",toString(run.name),"\_statistics.csv",sep="")) # Saves statistical validation results

## Create plots of cluster models

Plots of downcore distributions of cluster results for both model and hierarchical clustering, 2-3 PCs and 2-5 clusters. Plots go in "plot" RProject folder.

titles<-colnames(results)  
resultsd<-results[!duplicated(results$depth),]  
nColsL <- ncol(resultsd)  
for(i in 2:nColsL){  
 cluster<-factor(resultsd[,i])  
 p<-ggplot(resultsd, aes(x=resultsd$depth,y=resultsd[,i],fill=cluster))+geom\_bar(stat="identity",size=3,width=0.1) +  
 scale\_fill\_manual(values=rainbow(n=length(unique(resultsd[,i])))) + coord\_flip() + scale\_x\_reverse() +   
 labs(x="CCSF-A, meters",y=toString(colnames(resultsd)[i]))   
 print(p)  
 ggsave(filename=paste("../plot/",toString(run.name),"\_",toString(colnames(resultsd)[i]),".pdf",sep=''),height=12,width=3)  
 dev.off()  
}

## Warning: position\_stack requires non-overlapping x intervals  
  
## Warning: position\_stack requires non-overlapping x intervals  
  
## Warning: position\_stack requires non-overlapping x intervals  
  
## Warning: position\_stack requires non-overlapping x intervals