Clustering and Random Forests of Disease Traits

2019-2020 NSF EAGER Annual Report

Code for the hierarchical clustering approach used to determine susceptibility group identity and Random Forest Analysis to determine most influential traits in cluster identification.

Methods:

We used a Gower dissimilarity matrix that allows for mixed variables and variable weights when determining similarity of samples based on traits. The dissimilarity matrix is then used in the clustering step. Clustering was done using the function hclust() with Ward's minimum variance method of hierarchical clustering. In the resultant dendrogram, the height of the fusion provided on the vertical axis indicates the (dis)similarity between two observations. The higher the height of the fusion, the less similar the observations. We identify clusters by cutting the tree, so the height of the cut is indicative of similarity within the cluster. There is no universal method for determining the best number of clusters from a dendrogram.

We used random forest analyses to determine what traits most influence the cluster structure. Random forest analysis is a machine based learning method that estimates variable importance by combining many classification trees through bootstrap sampling and model averaging. We ran 2000 iterations of a random forest analysis that computed the increase in cluster mis-classification rate for each trait when it was excluded and all other traits held constant. Traits that resulted in high mis-class rates were determined to be the most important traits.

Code:

Packages and data load-in

```
library(FD)         # gower dissimiliarity matrix and some other
#clustering stuff
library(ggplot2)    # plotting
library(vegan)      # stats
library(igraph)     # networks
library(tidyverse)  # data manipulation
library(cluster)    # clustering algorithms
library(factoextra) # clustering visualization
library(dendextend) # for making dendograms fancy
library(clValid)    # has the dunns index function for comparing
#cluster numbers
library(mclust)     # normal mixture modeling for model based clustering,
#classification, and density estimation
library(randomForest) # Random Forests functions
```

```
#Set your directory and load the data
setwd("~/Dropbox/disease_trait_models/DiseaseTraitSpace_WP")
#Load data
trait.data<-read.csv("Clustering_and_RF/traits_loggenes_env.csv",row.names=1)
#this trait file contains all genes that are indicated to be
#environmentally sensitive and their values are log-normalized.
```

Custom functions to wrap up data analysis and visualization

Function to color each sample name by the species of the sample

```r
sp_bars_labels<-function(data){
  #Create a vector giving a color for each sample to which
  #species it belongs to
  spnames <- rep("Other", length(rownames(data)))
  is_x <- grepl("Mcav", rownames(data))
  spnames[is_x] <- "Mcav"
  is_x <- grepl("Past", rownames(data))
  spnames[is_x] <- "Past"
  is_x <- grepl("Ppor", rownames(data))
  spnames[is_x] <- "Ppor"
  is_x <- grepl("Oann", rownames(data))
  spnames[is_x] <- "Oann"
  is_x <- grepl("Ssid", rownames(data))
  spnames[is_x] <- "Ssid"
  is_x <- grepl("Cnat", rownames(data))
  spnames[is_x] <- "Cnat"
  is_x <- grepl("Ofav", rownames(data))
  spnames[is_x] <- "Ofav"
  spnames<-as.factor(spnames)
  n_sp <- length(unique(spnames))
  cols_sp <- colorspace::rainbow_hcl(n_sp, c = 70, l  = 50)
  col_sample_sp <- cols_sp[spnames]
  return(col_sample_sp)
}
```

Function to color a bar with the infected status of a sample (control, uninfected,infected)

```r
infstatus_bars_labels<-function(data){

  ### Infected status, control exposed infected
  status<-as.factor(data$Infected_Status)
  cols_status<-c("grey","black","red")
  col_sample_Infstatus<-cols_status[status]
  return(col_sample_Infstatus)
}
```

Function to color a bar by the number of days it took for a sample to become infected (0 indicates that a sample was never infected)

```r
daystoinf_bars_labels<-function(data){

  ### days to infection
  fact_daystoinf<-as.factor(data$days_to_infection)
  n_daystoinf<-length(levels(fact_daystoinf))
  if(n_daystoinf==1){
    cols_daytoinf<-"black"
  }else{
    cols_daytoinf <- colorspace::diverging_hcl(n_daystoinf)
  }
  col_sample_daystoinf <- cols_daytoinf[fact_daystoinf]
  return(col_sample_daystoinf)
}
```

Function to run clustering on dataset and then make dendrogram figure

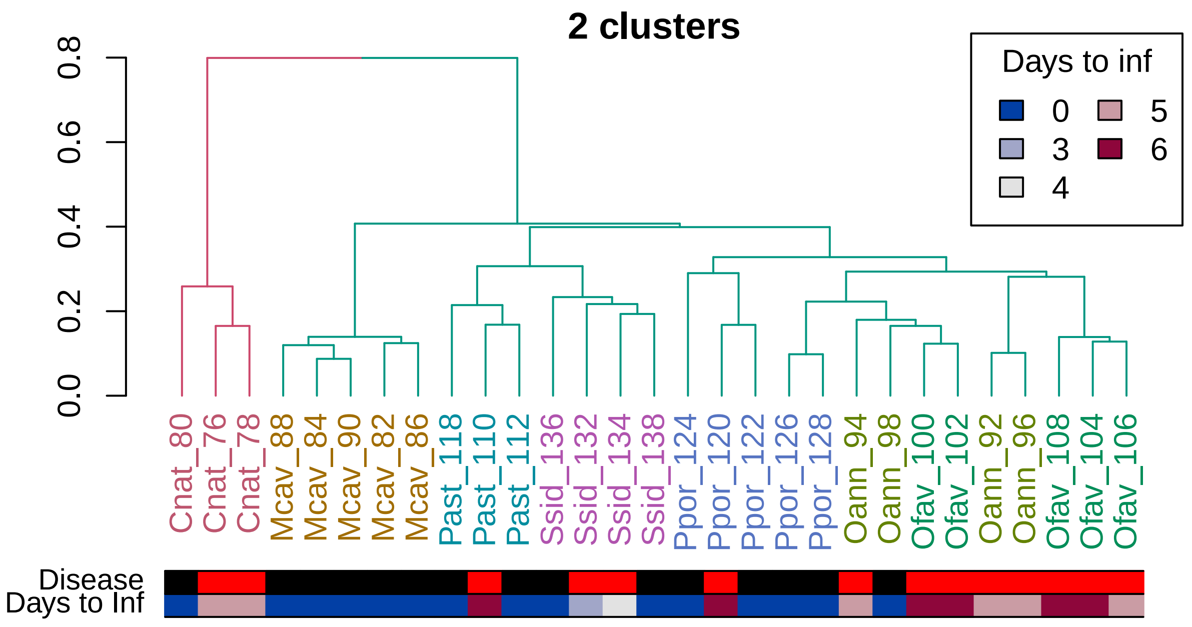
```
traits_clust<- function(data,numclustviz,colstoignore,name){
  col_sample_sp<-sp_bars_labels(data)
  fact_daystoinf<-as.factor(data$days_to_infection)
  n_daystoinf<-length(levels(fact_daystoinf))
  if(n_daystoinf==1){
    cols_daytoinf<-"black"
  }else{
    cols_daytoinf <- colorspace::diverging_hcl(n_daystoinf)
  }

  col_sample_daystoinf <- cols_daytoinf[fact_daystoinf]
  col_sample_Infstatus<-infstatus_bars_labels(data)

  data<-data[,-colstoignore]
  #make dissimiliarity matrix and run hclust
  data_gdis<-gowdis(data)
  hc_gowdis <- hclust(data_gdis, method = "ward.D2" )
  dend <- as.dendrogram(hc_gowdis)

  #customize denddrogram
  col_dend <- color_branches(dend, k = numclustviz)

  labels_colors(col_dend)<-col_sample_sp[order.dendrogram(col_dend)]

  #Make dendrogram figure
  par(mar = c(12,4,1,1))
  plot(col_dend)
  colored_bars(cbind(col_sample_daystoinf,col_sample_Infstatus),
               col_dend, rowLabels = c("Days to Inf","Disease"))
  title(name)
  legend("topright",inset = c(0,-0.03), legend = levels(fact_daystoinf),
         fill = cols_daytoinf,title="Days to inf",ncol=2,xpd=NA)
  return(hc_gowdis)
}
```

Function to run random forest analysis on data

```
traits_RF<-function(data,clusternumber,colstoignore,name){
  data<-data[,-colstoignore]
  #cut to just keep one cluster column in the dataset
  data_gdis<-gowdis(data)
  hc_gowdis <- hclust(data_gdis, method = "ward.D2" )
  Nthcol<-ncol(data)
  groups<-cutree(hc_gowdis,clusternumber)
  data[,Nthcol+1]<-as.factor(groups)
  cluster_name<-paste("cluster",clusternumber, sep="")
  colnames(data)[Nthcol+1]<-cluster_name
  cluster<-data[,Nthcol+1]
  #print(class(cluster))

  #impute to get rid of NAs
  if (any(is.na(data))==TRUE){
    mydataImpute <- rfImpute(y=cluster, x=data[,1:Nthcol])
  }else{
    mydataImpute<-data[,-ncol(data)]
  }

  data_rF <- randomForest(cluster ~ ., mydataImpute, ntree=20000)
  print(data_rF)
  varImpPlot(data_rF)
  title(name)
  return(importance(data_rF,type=2))
}
```

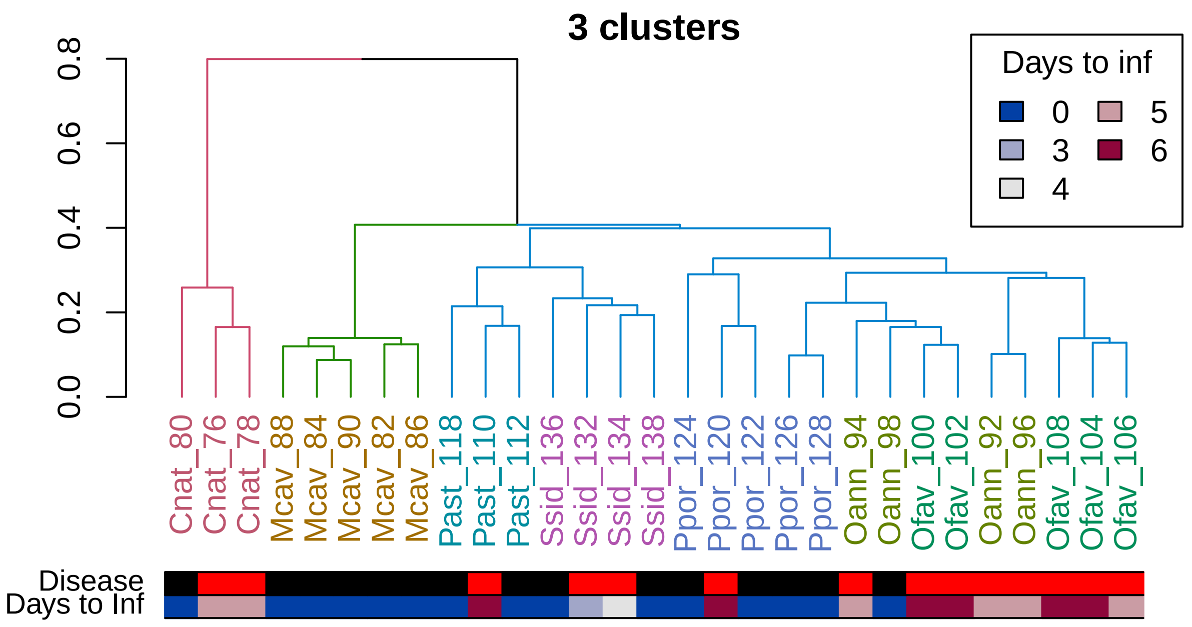
Implement Clustering Analysis and Visualization

```
#head(trait.data)
data_nocontrols<-trait.data%>%
  rownames_to_column('name') %>%
  filter(Infected_Status!="Control")%>%
  filter(ID!=72)%>% #Cnat 72 consistently clusters by itself so it is removed here
  column_to_rownames('name')
data_nocontrols<-na.omit(data_nocontrols) #removes samples with no gene expression data

colstoignore<-c(1:10,17)# all traits that are just the morphology,
#species name, disease related, and Red 660, and now genes

traits_clust(data_nocontrols,2,colstoignore,"2 clusters")
```



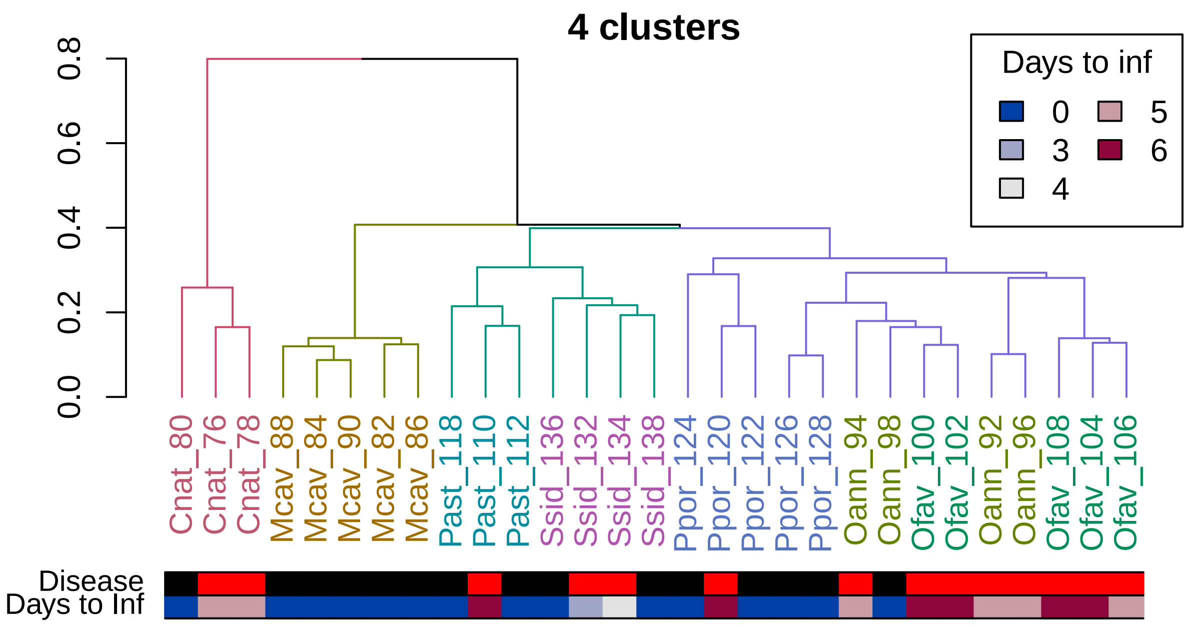
```
##
## Call:
## hclust(d = data_gdis, method = "ward.D2")
##
## Cluster method   : ward.D2
## Number of objects: 29
```

```
traits_clust(data_nocontrols,3,colstoignore, "3 clusters")
```



```
##
## Call:
## hclust(d = data_gdis, method = "ward.D2")
##
## Cluster method   : ward.D2
## Number of objects: 29
```

```
traits_clust(data_nocontrols,4,colstoignore, "4 clusters")
```

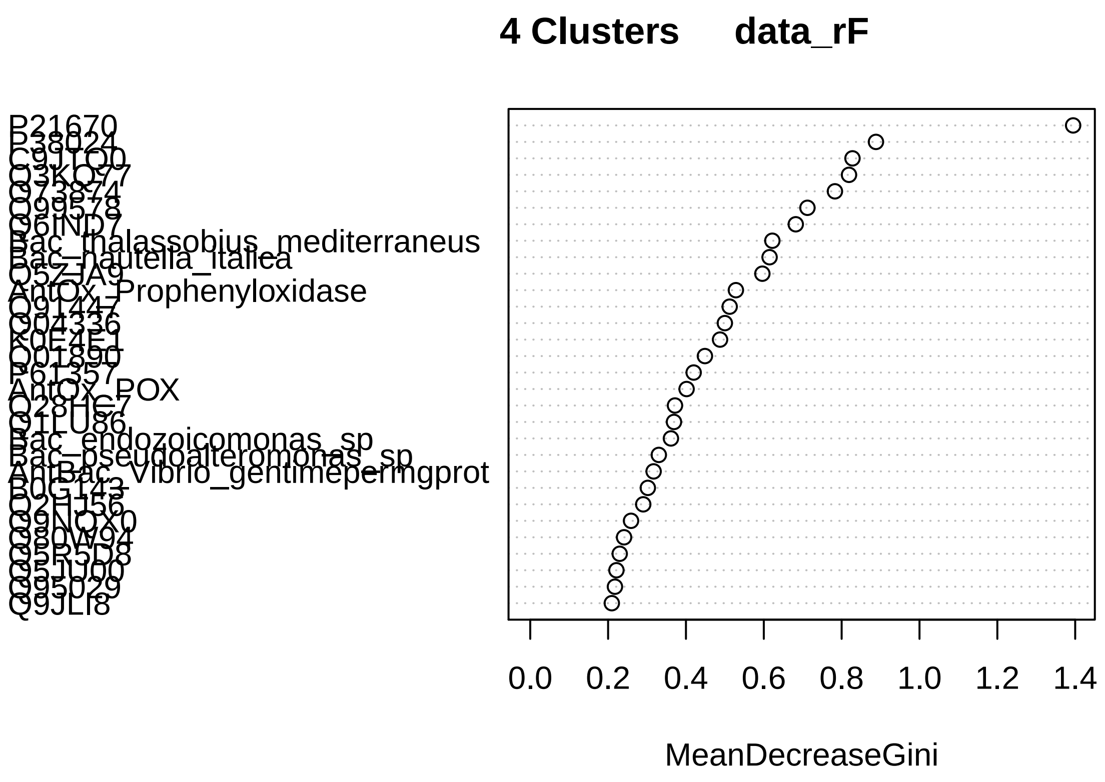


```
##
## Call:
## hclust(d = data_gdis, method = "ward.D2")
##
## Cluster method   : ward.D2
## Number of objects: 29
```

Implement Random Forests

```
#will determine important traits for determining identity within the 4 clusters above
rf_alldata_4c<-traits_RF(data_nocontrols,4,colstoignore,"4 Clusters")
```

```
##
## Call:
##  randomForest(formula = cluster ~ ., data = mydataImpute, ntree = 20000)
##                Type of random forest: classification
##                      Number of trees: 20000
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 6.9%
## Confusion matrix:
##   1 2  3 4 class.error
## 1 3 0  0 0   0.0000000
## 2 0 5  0 0   0.0000000
## 3 0 0 14 0   0.0000000
## 4 0 0  2 5   0.2857143
```



Output the RF trait rankings

```
rfoutput_to_ordereddf<-function(rf){
  RF.df<-as.data.frame(rf)
  RF.df$names<-rownames(rf)
  rf.df.ordered<-RF.df[order(RF.df$MeanDecreaseGini,decreasing=TRUE),]
  return(rf.df.ordered)
}
rfoutput_to_ordereddf(rf_alldata_4c)
```

```
##                                  MeanDecreaseGini
## P21670                              1.39469773
## P38024                              0.88814048
## C9JTQ0                              0.82777461
## Q3KQ77                              0.81889069
## O73874                              0.78270808
## Q99578                              0.71209882
## Q6IND7                              0.68209911
## Bac_thalassobius_mediterraneus      0.62192053
## Bac_nautella_italica                0.61487343
## Q5ZJA9                              0.59627550
## AntOx_Prophenyloxidase              0.52830148
## Q91447                              0.51231002
## Q04336                              0.49991719
## K0E4E1                              0.48756668
## Q01890                              0.44873858
## P61357                              0.41979231
## AntOx_POX                           0.40159574
## Q28HC7                              0.37180489
## Q1LU86                              0.36921270
## Bac_endozoicomonas_sp               0.36138110
## Bac_pseudoalteromonas_sp            0.33030098
## AntBac_Vibrio_gentimepermgprot      0.31691303
## B0G143                              0.30206587
## Q2HJ56                              0.29023499
## Q9NQX0                              0.25888578
## Q80W94                              0.24095348
## Q5R5D8                              0.22977576
## Q5JU00                              0.22137343
## Q95029                              0.21756128
## Q9JLI8                              0.20951369
## Q5JVL4                              0.19982438
## Q8N6G6                              0.19296478
## Q9SY73                              0.18715011
## Q3SZQ6                              0.18167077
## Q5ZL16                              0.17542890
## A7SDW5                              0.17511405
## A7SFB5                              0.16070721
## P34897                              0.15235155
## Bac_arthrobacter_ramosus            0.15147639
## O16025                              0.14471890
## Q6DG99                              0.14173381
## Q7Z494                              0.13914060
## A3KP77                              0.13864849
## Q08CD5                              0.13287458
## A7SE05                              0.12880928
## Q8K3Z0                              0.12284561
## Q96P65                              0.10387675
## C3YWU0                              0.10329666
## Q5ZID0                              0.10186718
## Q460N5                              0.10184923
## Q0EEE2                              0.10127273
## AntOx_Catalase                      0.10060587
## O73792                              0.09570587
## Q9R080                              0.08970481
## Q9NXG6                               0.08649249
## Q9BV90                               0.08343975
## P11029                               0.08308324
## Q9CZB9                               0.06698445
## Q5S1U6                               0.06519999
## Bac_pseudomonas_veronii              0.05604915
##                                                          names
## P21670                                                   P21670
## P38024                                                   P38024
## C9JTQ0                                                   C9JTQ0
## Q3KQ77                                                   Q3KQ77
## O73874                                                   O73874
## Q99578                                                   Q99578
## Q6IND7                                                   Q6IND7
## Bac_thalassobius_mediterraneus Bac_thalassobius_mediterraneus
## Bac_nautella_italica                     Bac_nautella_italica
## Q5ZJA9                                                   Q5ZJA9
## AntOx_Prophenyloxidase                 AntOx_Prophenyloxidase
## Q91447                                                   Q91447
## Q04336                                                   Q04336
## K0E4E1                                                   K0E4E1
## Q01890                                                   Q01890
## P61357                                                   P61357
## AntOx_POX                                             AntOx_POX
## Q28HC7                                                   Q28HC7
## Q1LU86                                                   Q1LU86
## Bac_endozoicomonas_sp                   Bac_endozoicomonas_sp
## Bac_pseudoalteromonas_sp             Bac_pseudoalteromonas_sp
## AntBac_Vibrio_gentimepermgprot AntBac_Vibrio_gentimepermgprot
## B0G143                                                   B0G143
## Q2HJ56                                                   Q2HJ56
## Q9NQX0                                                   Q9NQX0
## Q80W94                                                   Q80W94
## Q5R5D8                                                   Q5R5D8
## Q5JU00                                                   Q5JU00
## Q95029                                                   Q95029
## Q9JLI8                                                   Q9JLI8
## Q5JVL4                                                   Q5JVL4
## Q8N6G6                                                   Q8N6G6
## Q9SY73                                                   Q9SY73
## Q3SZQ6                                                   Q3SZQ6
## Q5ZL16                                                   Q5ZL16
## A7SDW5                                                   A7SDW5
## A7SFB5                                                   A7SFB5
## P34897                                                   P34897
## Bac_arthrobacter_ramosus             Bac_arthrobacter_ramosus
## O16025                                                   O16025
## Q6DG99                                                   Q6DG99
## Q7Z494                                                   Q7Z494
## A3KP77                                                   A3KP77
## Q08CD5                                                   Q08CD5
## A7SE05                                                   A7SE05
## Q8K3Z0                                                   Q8K3Z0
## Q96P65                                                   Q96P65
## C3YWU0                                                       C3YWU0
## Q5ZID0                                                       Q5ZID0
## Q460N5                                                       Q460N5
## Q0EEE2                                                       Q0EEE2
## AntOx_Catalase                                       AntOx_Catalase
## O73792                                                       O73792
## Q9R080                                                       Q9R080
## Q9NXG6                                                       Q9NXG6
## Q9BV90                                                       Q9BV90
## P11029                                                       P11029
## Q9CZB9                                                       Q9CZB9
## Q5S1U6                                                       Q5S1U6
## Bac_pseudomonas_veronii                     Bac_pseudomonas_veronii
```