CMI_ABCD_JIVE

Lucinda Sisk

February 28, 2020

Drop empty columns, combine data to ensure same IDs

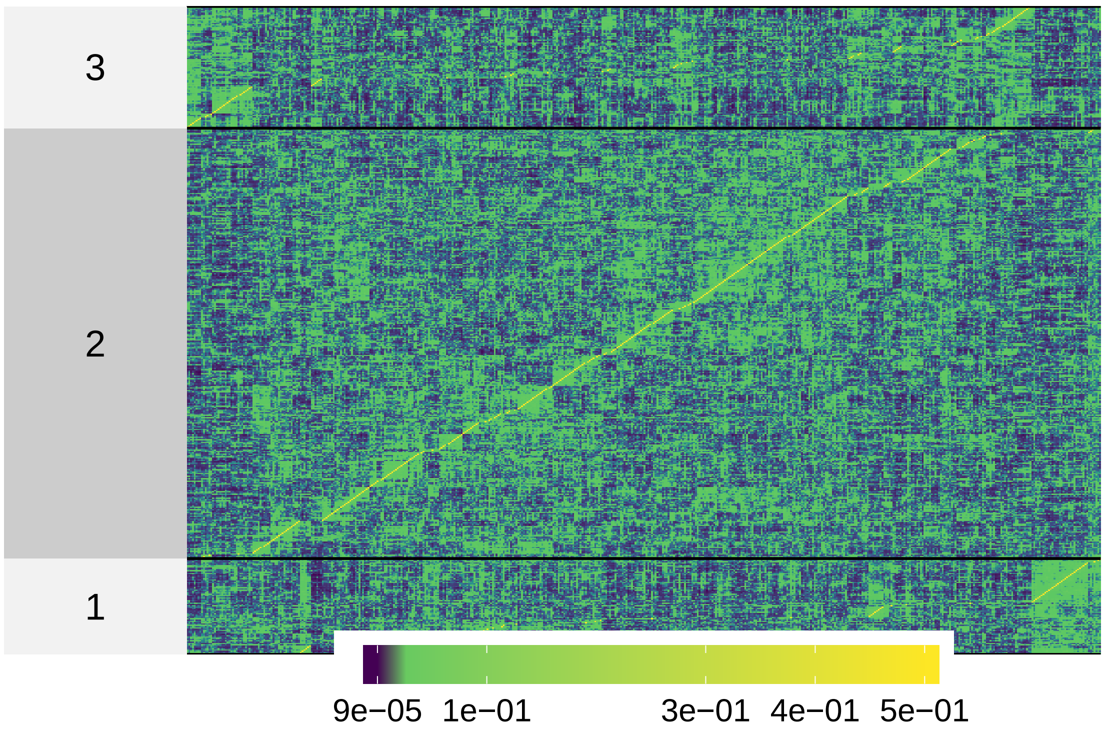
```
not_all_na <- function(x) any(!is.na(x))

# Set local vars to speed computaitons
thick_data <- thick_data  #[1:50]
myelin_data <- myelin_data  #[1:50]

# Drop ID columns; drop all columns that sum to 0
myelin_data_clean <- myelin_data %>% select_if(not_all_na)

# Drop ID columns; drop all columns that sum to 0
thick_data_clean <- thick_data %>% select_if(not_all_na)

# Add in group info merged_group <- full_join(corrected_groups,
# uncorrected_groups, by='subid')

# Merge data frames to ensure they are in same order
full_df <- myelin_data_clean %>% left_join(thick_data_clean, by = "subid") %>% left_join(pheno_data,
    by = c(subid = "subjectkey"))

# DFs with no group info
myelin <- full_df %>% select("subid", "zmyelin_set1":"zmyelin_set48")

thick <- full_df %>% select("subid", "zthickness_set1":"zthickness_set49")

pheno <- full_df %>% select("subid", "physical_abuse_sum":"fes_p_ss_fc")

# Convert to matrices and drop subject ID column; normalize the features with SNF
# method

myelin_mat <- myelin %>% dplyr::select(-c("subid")) %>% data.matrix() %>% standardNormalization()

thick_mat <- thick %>% dplyr::select(-c("subid")) %>% data.matrix() %>% standardNormalization()

pheno_mat <- pheno %>% dplyr::select(-c("subid")) %>% data.matrix() %>% standardNormalization()
```

SNF Analysis

- Documentation: https://cran.r-project.org/web/packages/SNFtool/SNFtool.pdf

```
# SNF analysis based on documentation
```

```r
data = list(pheno_mat, thick_mat)  #, myelin_mat)
truelabel = c(matrix(1, 500, 1))

# Set the other parameters
K = 12  # number of neighbours --> set at 20 in example; get error when using this with our data
alpha = 0.5  # hyperparameter in affinityMatrix
T = 20  # number of iterations of SNF
C = 3  #Number of clusters

# Calculate the distances for each view
data_dist = lapply(data, function(x) (dist2(x, x))^(1/2))

# Construct the similarity graphs
affinityL = lapply(data_dist, function(x) affinityMatrix(x, K, alpha))

# Example of how to use SNF to perform subtyping Construct the fused network
W = SNF(affinityL, K, T)
# Perform clustering on the fused network.
cluster_groups = spectralClustering(W, C)
# Use NMI to measure the goodness of the obtained labels.
NMI = calNMI(cluster_groups, truelabel)  #THIS IS GIVING AN ERROR BUT STILL RUNS
# #Display clusters generated by SNF displayClusters(W, cluster_groups)
# displayClustersWithHeatmap(W, cluster_groups,
# M_label_colors[,'spectralClustering']) ## Get a matrix containing the group
# information ## for the samples such as the SpectralClustering result and the
# True label M_label=cbind(cluster_groups,truelabel)
# colnames(M_label)=c('spectralClustering','TrueLabel') ## Use the
# getColorsForGroups function to assign a color to each group ## NB is more than
# 8 groups, you will have to input a vector ## of colors into the
# getColorsForGroups function
# M_label_colors=t(apply(M_label,1,getColorsForGroups)) ## or choose you own
# colors for each label, for example:
# M_label_colors=cbind('spectralClustering'=getColorsForGroups(M_label[,'spectralClustering'],
# colors=c('blue','green',
# 'red')),'TrueLabel'=getColorsForGroups(M_label[,'TrueLabel'],
# colors=c('cyan')))


# concordanceNetworkNMI(affinityL, 3)
superheat(W, scale = FALSE, pretty.order.rows = TRUE, pretty.order.cols = TRUE, membership.rows = clust
```



Prepare matrices, run jive

- Data requirements: A list of two or more linked data matrices on which to perform the JIVE decomposition. These matrices must have the same column dimension, which is assumed to be common.

```
# Combine measures into list

# data <- list(pheno_mat, myelin_mat, thick_mat)
```

JIVE for all data

```
# #Run JIVE analysis for ALL DATA #Estimate JIVE ranks based on permutation
# testing (best validated) #Row-orthogonality enforced between the joint and
# individual estimates and also between each individual estimate.  #Compute ranks
# try(cmi_jive_result <- jive(data, scale = TRUE, conv='default', method='perm'))
# #Get Results result_joint_rank <- cmi_jive_result$rankJ result_individ_rank <-
# cmi_jive_result$rankA #Plot variance explained by individual and joint ranks,
# and noise try(cmi_jive_var <- showVarExplained(cmi_jive_result, col =
# c('#811c4e', '#fdb663', '#37486d'))) #Plot heatmaps of results
# #try(cmi_jive_heatmaps <- showHeatmaps(cmi_jive_result, order_by = 0, show_all
# = TRUE)) #Plot PCA try(cmi_jive_pca <- showPCA(cmi_jive_result, n_joint =
# result_joint_rank, n_indiv = c(1,1), Colors = c('#811c4e', '#fdb663')))
```

```
### All Data Results: Joint `r result_joint_rank`, Individual `r
### result_individ_rank` cmi_jive_result$individual
```