

```

1 '''
2 Summary: While this notebook shares the same preprocessing and WGCNA foundational code as the WGCNA_Module_Functional_Enrich
3 its primary analytical focus shifts to dimensionality reduction and feature selection. It utilizes Principal
4 Component Analysis (PCA) and a Random Forest classifier to evaluate sample segregation based on clinical traits.
5 It then relies heavily on the PCAtools package to extract the top variance-driving genes (the top 1% to 10% outliers)
6 from the leading principal components, ultimately routing these specific, variance-extracted gene sets into Reactome
7 pathway enrichment analysis.
8 '''

```

```

1 #####
2
3 # _____ ANALYSIS PART A. _____
4
5 #####

```

```

1 %load_ext rpy2.ipynon
2 from google.colab import drive
3 drive.mount('/content/drive')

```

```

1 %%R -o DER_01 -o DER_03a -o DER_03b -o DER_tmp -o capstone
2
3 if (!requireNamespace("BiocManager", quietly = TRUE))
4   install.packages("BiocManager")
5
6 BiocManager::install("WGCNA")
7 library(WGCNA)
8 #BiocManager::install("zFPKM")
9 #library(zFPKM)
10 #BiocManager::install("CancerSubtypes")
11 #library(CancerSubtypes)
12 #BiocManager::install("DESeq2")
13 #library(DESeq2)
14
15 install.packages("tidyverse")
16 library(tidyverse)
17
18 capstone_tmp <- read.csv("/content/drive/My Drive/datasets/PEC_capstone_data_map_clinical.csv")
19 capstone <- capstone_tmp[, c(1, 2, 6, 7, 8)]
20
21 #DER_tmp <- read.delim("/content/drive/My Drive/datasets/Controls_only_PEC_Gene_expression_matrix_normalized.txt")
22 DER_tmp <- read.delim("/content/drive/My Drive/datasets/DER-02_PEC_Gene_expression_matrix_TPM.txt")
23 #DER_03_tmp <- DER_tmp[,1]
24 #sql_peaks_tmp <- gsub("\\.\\d+", "", DER_03_tmp)
25 #DER_tmp[,1] <- gsub("c(", "", sql_peaks_tmp, fixed="TRUE")
26
27 DER_01 <- read.delim("/content/drive/My Drive/datasets/DER-02_PEC_Gene_expression_matrix_TPM.txt")
28 #DER_01 <- read.delim("/content/drive/My Drive/datasets/Controls_only_PEC_Gene_expression_matrix_normalized.txt")
29 DER_03 <- DER_01[,1]
30 sql_peaks <- gsub("\\.\\d+", "", DER_03)
31 DER_01[,1] <- gsub("c(", "", sql_peaks, fixed="TRUE")
32 DER_03a <- DER_01[,1] # gene names
33 DER_03b <- colnames(DER_01)[-1] # sample names
34 DER_01 <- t(scale(t(DER_01[-1]))) # scaling the data (otherwise all of the highly expressed genes
35 #will cluster together even if they have different patterns
36 #among the samples)
37

```

```

1 import numpy # _____ Divide tissues and subjects to brain/peripheral
2 import numpy as np # and CTL/SCZ, respectively.
3 import pandas as pd
4 from sklearn.preprocessing import MinMaxScaler
5 from sklearn.feature_selection import VarianceThreshold
6 from sklearn.preprocessing import StandardScaler
7 from scipy.stats import gaussian_kde
8
9 %R library(data.table)
10 %R BrainSeq <- fread('https://raw.githubusercontent.com/LieberInstitute/brainseq_phase2/master/BrainSeq_Phase2_phenotype_c
11 %R head(BrainSeq)
12 %R E <- readRDS('/content/drive/My Drive/datasets/expression.rds')
13 %R gdt <- readRDS('/content/drive/My Drive/datasets/gdt.rds')
14 %R samp <- fread('https://storage.googleapis.com/gtex_analysis_v7/annotations/GTex_v7_Annotations_SampleAttributesDS.txt')
15 %R subj <- fread('https://storage.googleapis.com/gtex_analysis_v7/annotations/GTex_v7_Annotations_SubjectPhenotypesDS.txt')

```

```

16 %R samp[, SUBJID := gsub('([^-]*)-([^-]*)-.*', '\\1-\\2', SAMPID)]
17 %R sdt <- merge(samp, subj, by='SUBJID')
18 %R sdt <- merge(samp, subj, by='SUBJID')[SAMPID %in% rownames(E)]
19 # _____
20
21 %R -o BrainSeq
22 %R -o DER_01
23 # _____ Check for bias due to Tissue/Sex/Race.
24 #1.____BrainSeq[~BrainSeq.Region.str.contains("HIPPO")] #____DLPFC/HIPPO # Replace each line to the line below "3.____
25 #2.____BrainSeq[~BrainSeq.Race.str.contains("AA")] #____CAUC/AA
26 #3.____BrainSeq[~BrainSeq.Sex.str.contains("F")] #____M/F
27 BrSeq = pd.DataFrame(BrainSeq)
28 capstone = pd.DataFrame(capstone)
29
30 DER01 = pd.DataFrame(DER_01)
31 DER01.columns = DER_03b
32 DER01 = np.transpose(DER01)
33 DER01.columns = DER_03a
34 DER01 = np.transpose(DER01)
35
36 # _____ I. BrainSeq DATASET(from Lieber Institute)
37
38 arr20 = np.array([])
39 arr20 = BrSeq['BrNum']
40 arr21 = np.array([])
41 arr21 = DER01.columns
42 xy21, x21_ind, y21_ind = np.intersect1d(arr20, arr21,
43                                     return_indices=True)
44 data1 = []
45 for i in x21_ind:
46     data1.append([BrSeq['BrNum'][i], BrSeq['Region'][i], BrSeq['Dx'][i],
47                 BrSeq['Sex'][i], BrSeq['Race'][i], BrSeq['RIN'][i], BrSeq['Age'][i]])
48 DER01_BrSeq = pd.DataFrame(data1, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G'])
49 #DER01_BrSeq = DER01_BrSeq[~DER01_BrSeq.B.str.contains("HIPPO")]
50 # _____
51 %R -o sdt
52 GTx = pd.DataFrame(sdt)
53
54 arr22 = np.array([])
55 arr22 = GTx['SUBJID']
56 xy22, x22_ind, y22_ind = np.intersect1d(arr22, arr21,
57                                     return_indices=True)
58 data2 = []
59 for i in x22_ind:
60     data2.append([GTx['SUBJID'][i], GTx['SMTSD'][i]])
61 df_GTx = pd.DataFrame(data2, columns=['A', 'B'])
62 # _____
63 #FER01 = DER01.columns.to_series().str.contains('CMC')
64 #y23_ind = np.array([i for i, x in enumerate(FER01) if x])
65 #FER02 = DER01.columns.to_series().str.contains('Br')
66 #y24_ind = np.array([i for i, x in enumerate(FER02) if x])
67 #gtx_br = [*y23_ind, *y24_ind]
68
69 #gtx_br = [*y21_ind, *y23_ind, *y22_ind]
70 gtx_br = [*y21_ind, *y22_ind]
71
72 n13_train_scz = pd.DataFrame(DER01)
73 n14_train_scz = n13_train_scz.values[:,]
74 n15_train_scz = n14_train_scz[:,gtx_br] #_____ Filter samples
75
76 # _____ II. CAPSTONE DATASET(from Kelsey)
77 '''
78 arr19a = np.array([])
79 arr19a = capstone['Synapse..individualID']
80 arr19b = np.array([])
81 arr19b = capstone['resource.psychencode.org..individualID']
82 arr21 = np.array([])
83 arr21 = DER01.columns
84 xy22, x22_ind, y22_ind = np.intersect1d(arr19b, arr21,
85                                     return_indices=True)
86 gtx_cr = [*y22_ind]
87
88 data2 = []
89 for i in x22_ind:
90     data2.append([capstone['resource.psychencode.org..individualID'][i],
91                 capstone['diagnosis'][i], capstone['sex'][i], capstone['ethnicity'][i]])
92 capstone_BrSeq2 = pd.DataFrame(data2, columns=['A', 'B', 'C', 'D'])

```

```

93
94 BipD = np.where(capstone_BrSeq2.B.str.contains("Bipolar Disorder"))
95 AutD = np.where(capstone_BrSeq2.B.str.contains("Autism Spectrum Disorder"))
96 AffD = np.where(capstone_BrSeq2.B.str.contains("Affective Disorder"))
97 BPD = np.where(capstone_BrSeq2.B.str.contains("BP"))
98
99 Dis_ind = [*BipD, *AutD, *AffD, *BPD]
100 Dis_ind_flat = [item for sublist in Dis_ind for item in sublist]
101
102 capstone_BrSeq2 = capstone_BrSeq2[~capstone_BrSeq2.B.str.contains("Bipolar Disorder")]
103 capstone_BrSeq2 = capstone_BrSeq2[~capstone_BrSeq2.B.str.contains("Autism Spectrum Disorder")]
104 capstone_BrSeq2 = capstone_BrSeq2[~capstone_BrSeq2.B.str.contains("Affective Disorder")]
105 capstone_BrSeq2 = capstone_BrSeq2[~capstone_BrSeq2.B.str.contains("BP")]
106
107 gj = []
108 for i in range(len(capstone_BrSeq2)):
109     if len(list(capstone_BrSeq2['A'])[i])==0 :
110         gj.append(i)
111     if len(list(capstone_BrSeq2['B'])[i])==0 :
112         gj.append(i)
113     if len(list(capstone_BrSeq2['C'])[i])==0 :
114         gj.append(i)
115     if len(list(capstone_BrSeq2['D'])[i])==0 :
116         gj.append(i)
117
118 res = []
119 [res.append(x) for x in gj if x not in res];
120
121 mask = np.ones(len(capstone_BrSeq2), bool)
122 mask[res] = False
123 capstone_BrSeq2_new = capstone_BrSeq2.iloc[mask]
124
125 len(capstone_BrSeq2_new)
126
127 n13_train_scz = pd.DataFrame(DER01)
128 n14_train_scz = n13_train_scz.values[:,]
129 n15_train_scz = n14_train_scz[:,gtx_cr] #_____ Filter samples
130 Dis_mask = np.ones(n15_train_scz.shape[1], bool)
131 Dis_mask[Dis_ind_flat] = False
132 n15_train_scz = n15_train_scz[:,Dis_mask]
133 n15_train_scz = n15_train_scz[:,mask]
134 '''
135 #_____
136
137 #n13_train_scz_min = n15_train_scz.astype(float) #_____1. No transformation
138
139 ###medianValue = median(logData); #_____2. Median centering data before log2-trans
140 ###medianCtrData = logData-medianValue
141
142 n13_train_scz_min = np.log(n15_train_scz.astype(float)+1) #_____3. Log2 transformation
143
144 #n13_train_scz_min <- zFPKM(n15_train_scz.astype(float)) #_____4. Z-scale transformation (in R, from zFPK
145
146 n13_train_scz_min = np.transpose(n13_train_scz_min) #_____ Shape the array as (samples, genes)
147
148 n13_train_scz_min[np.isnan(n13_train_scz_min)] = 0 #(57820, 414) #_____ Replace NAs with zeroes
149 pd.DataFrame(n13_train_scz_min).isnull().sum()
150
151 print(n13_train_scz_min.shape)
152 jdx = np.argwhere(np.all(n13_train_scz_min[... ,:] == 0, axis=0)) #_____ Remove zero columns
153 n13_train_scz_min = np.delete(n13_train_scz_min, jdx, axis=1)
154 print(n13_train_scz_min.shape)
155
156
157 f = VarianceThreshold().fit(n13_train_scz_min)
158 n13_train_scz_min = n13_train_scz_min[:, f.variances_ > 0.0] #._____
159 print(n13_train_scz_min.shape)
160
161
162 kdx = jdx.tolist() #_____ Final gene list after zero column removal
163 ldx = [item for sublist in kdx for item in sublist]
164 ER_TMM = []
165 DER_tmp.rename(columns = {'gene_id':'GeneName'}, inplace = True)
166 for i in range(len(DER_tmp['GeneName'])[ldx])):
167     ER_TMM.append(DER_tmp['GeneName'][ldx][i])
168 crr1 = np.array([])
169 crr1 = DER_tmp['GeneName']

```

```

170 crr2 = np.array([])
171 crr2 = ER_TMM
172 c22, c1_ind, c2_ind = np.intersect1d(crr1, crr2,
173                                     return_indices=True)
174 crr1[c1_ind] = 0
175 crr1 = [i for i in crr1 if i != 0]
176
177 crr1 = pd.DataFrame(crr1).iloc[f.variances_ > 0.0] #_____ 2
178 #print(pd.DataFrame(crr1).iloc[transform.get_support(indices=True)].shape) #____ Gene Names(columns)
179
180
181 '''
182 for i in range(n13_train_scz_min.shape[0]): #_____ Z-FPKM(log2) transformation (in Python)
183     n13_train_tmp = n13_train_scz_min[:,i]
184     kernel = gaussian_kde(n13_train_tmp)
185     xi = np.linspace(n13_train_tmp.min(), n13_train_tmp.max(), 100)
186     yi = kernel.evaluate(xi)
187     mu = xi[np.argmax(yi)]
188     U = n13_train_scz_min[n13_train_scz_min > mu].mean()
189     sigma = (U - mu) * np.sqrt(np.pi / 2)
190     n13_train_scz_min[:,i] = (n13_train_tmp - mu) / sigma
191 '''
192
193 %R -i crr1
194 %R -i n13_train_scz_min #_____ Remove zero columns
195 %%R n13_train_scz_min <- n13_train_scz_min[, colSums(n13_train_scz_min != 0) > 0]
196 %%R print(dim(n13_train_scz_min))
197
198 %%R -i n13_train_scz_min #_____ z-transformation (in R)
199 #zscore<- function(x){
200 #     z<- (x - mean(x)) / sd(x)
201 #     return(z)
202 #}
203 #print(dim(n13_train_scz_min))
204 #n13_train_scz_min <- zscore(n13_train_scz_min)
205 #n13_train_scz_min <- n13_train_scz_min[, colSums(n13_train_scz_min != 0) > 0]
206 #print(dim(n13_train_scz_min))
207
208 ## PS: The FPKM counts are already normalised but are absolutely not suitable for
209 ## cross-sample comparisons, i.e., not suitable for differential expression analysis.
210 #_____
211
212 #_____ re-transform from log2(FPKM + z) ->
213 # -> FPKM ->
214 # -> zFPKM scores
215 #z=0.1
216 #exp.fpkm <- 2^expr
217 #exp.fpkm.original <- exp.fpkm - z
218 #exp.zfpkm <- zFPKM(exp.fpkm.original)
219 ##filter out lowly expressed genes
220 #thres <- (ncol(exp.zfpkm) * 30) / 100
221 ##filter all expression values that have absolute zfpkm score above 3.0 in more than 70% of the samples
222 #expr.zfpkm.filtered <- exp.zfpkm[(rowSums(abs(exp.zfpkm) > 3.0)) > thres, ]
223
224 #_____ Traits (CTL/SCZ)
225
226 data4 = [] #I. BrainSeq DATASET(from Lieber Institute)
227 for i in DER01_BrSeq['B']:
228     if i == 'DLPFC':
229         k = 0
230     if i == 'HIPPO':
231         k = 1
232     data4.append(k)
233 DER01_BrSeq6 = pd.DataFrame(data4, columns=['region'])
234
235 data4 = []
236 for i in DER01_BrSeq['C']:
237     if i == 'Control':
238         k = 0
239     if i == 'Schizo':
240         k = 1
241     data4.append(k)
242 DER01_BrSeq1 = pd.DataFrame(data4, columns=['diag'])
243
244 data4 = []
245 for i in DER01_BrSeq['D']:
246     if i == 'F':

```

```

247     k = 0
248     if i == 'M':
249         k = 1
250     data4.append(k)
251     DER01_BrSeq2 = pd.DataFrame(data4, columns=['sex'])
252
253     data4 = []
254     for i in DER01_BrSeq['E']:
255         if i == 'CAUC':
256             k = 1
257         else:
258             k = 0
259     data4.append(k)
260     DER01_BrSeq3 = pd.DataFrame(data4, columns=['ethn'])
261
262     data4 = []
263     for i in DER01_BrSeq['F']:
264         data4.append(i)
265     DER01_BrSeq4 = pd.DataFrame(data4, columns=['rin'])
266
267     data4 = []
268     for i in DER01_BrSeq['G']:
269         data4.append(i)
270     DER01_BrSeq5 = pd.DataFrame(data4, columns=['age'])
271
272     DER01_BrSeq_Tot = pd.concat([DER01_BrSeq1, DER01_BrSeq2, DER01_BrSeq3,
273                                  DER01_BrSeq4, DER01_BrSeq5, DER01_BrSeq6], axis=1)
274
275     n13_trait_path1 = []
276     n13_trait_path1 = pd.DataFrame(DER01_BrSeq_Tot)
277
278     n13_trait_path = n13_trait_path1
279     %R -i n13_trait_path
280
281     #
282     '''
283
284     #pd.DataFrame(capstone_BrSeq2_new['B']).value_counts()
285     #pd.DataFrame(capstone_BrSeq2_new['C']).value_counts()
286     #pd.DataFrame(capstone_BrSeq2_new['D']).value_counts()
287
288     data4 = []
289     for i in capstone_BrSeq2_new['B']:
290         if i == 'Control':
291             k = 0
292         if i == 'Schizophrenia':
293             k = 1
294     data4.append(k)
295     DER01_BrFin1 = pd.DataFrame(data4, columns=['diag'])
296
297     data4 = []
298     for i in capstone_BrSeq2_new['C']:
299         if i == 'F':
300             k = 0
301         if i == 'M':
302             k = 1
303     data4.append(k)
304     DER01_BrFin2 = pd.DataFrame(data4, columns=['sex'])
305
306     data4 = []
307     for i in capstone_BrSeq2_new['D']:
308         if i == 'CAUC':
309             k = 1
310         else:
311             k = 0
312     data4.append(k)
313     DER01_BrFin3 = pd.DataFrame(data4, columns=['ethn'])
314
315     DER01_BrFin = pd.concat([DER01_BrFin1, DER01_BrFin2, DER01_BrFin3], axis=1)
316
317     n13_trait_path1 = []
318     n13_trait_path1 = pd.DataFrame(DER01_BrFin)
319
320     n13_trait_path = n13_trait_path1
321     %R -i n13_trait_path
322     '''
323

```

#II. CAPSTONE DATASET(from Kelsey)

```

324 del DER01_BrSeq #_____ Clear unused variables
325 #del DER01_BrFin
326 del DER01
327 del n13_train_scz
328 del n14_train_scz
329 del n15_train_scz
330

```

```

1 #_____ Defining number of CTLs and SCZs
2 # before removing outliers!!!!!!!!!!
3 # Use it as input to the PCA below
4
5
6 #I. BrainSeq DATASET(from Lieber Institute)
7 DER01_BrSeq = pd.DataFrame(data1, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G'])
8 #DER01_BrSeq = DER01_BrSeq[~DER01_BrSeq.B.str.contains("HIPPO")]
9
10 FER05 = DER01_BrSeq['C'].str.contains('Con')
11 z23_ind = np.array([i for i, x in enumerate(FER05) if x])
12
13 FER06 = DER01_BrSeq['C'].str.contains('Sc')
14 z24_ind = np.array([i for i, x in enumerate(FER06) if x])
15
16 gtx_all = [*z23_ind, *z24_ind]
17 #print(DER01_BrSeq.values[gtx_all])
18
19 n13_pca_scz_min = n13_train_scz_min[gtx_all,]
20 n13_pca_path = n13_trait_path.iloc[gtx_all]
21 #n13_pca_path = n13_trait_path['D'][gtx_all]
22 %R -i n13_pca_path
23
24 DER01_BrSeq['C'].value_counts() #_____ CTLs and SCZs
25
26 #_____
27 '''
28 #II. CAPSTONE DATASET(from Kelsey)
29 FER05 = capstone_BrSeq2_new['B'].str.contains('Con')
30 z23_ind = np.array([i for i, x in enumerate(FER05) if x])
31
32 FER06 = capstone_BrSeq2_new['B'].str.contains('Sc')
33 z24_ind = np.array([i for i, x in enumerate(FER06) if x])
34
35 gtx_all = [*z23_ind, *z24_ind]
36 #print(DER01_BrFin1.values[gtx_all])
37
38 n13_pca_scz_min = n13_train_scz_min[gtx_all,]
39 n13_pca_path = n13_trait_path.iloc[gtx_all]
40 #n13_pca_path = n13_trait_path['diag'][gtx_all]
41 %R -i n13_pca_path
42
43 #DER01_BrFin['diag'].value_counts() #_____ CTLs and SCZs
44 capstone_BrSeq2_new['B'].value_counts()
45 '''

```

```

1 %R -i n13_pca_scz_min #_____ Delete R variables, save files
2 # for WGCNA analysis
3 rm(list=ls()[! ls() %in% c("n13_pca_scz_min", "n13_pca_path", "crr1")])
4 ls()
5
6 #save(n13_pca_scz_min, file = '/content/drive/My Drive/datasets/n13_pca_scz_min.RData')
7 #save(n13_pca_path, file = '/content/drive/My Drive/datasets/n13_pca_path.RData')
8

```

```

1 %R -i n13_pca_scz_min
2 #_____ plot PCA before removing outliers!!!!!!!!!!
3
4 library(ggplot2)
5
6 n13_pca_scz_min <- n13_pca_scz_min[, which(apply(n13_pca_scz_min, 2, var) != 0)]#_Remove zero variance columns from the new
7 genes=paste("gene",seq(1:dim(n13_pca_scz_min)[2]), sep="")
8 colnames(n13_pca_scz_min)=genes
9 row.names(n13_pca_scz_min)=c(paste0("ctl_",seq(1:261)), paste0("scz",seq(1:153))) #___Python command: DER01_BrSeq['C'].val
10 condition1=rep(c("ctl"), each=261)
11 condition2=rep(c("scz"), each=153)
12 condition1 <- append(condition1, condition2)

```

```

13 length(condition1)
14
15 #pca_data=prcomp(n13_train_scz_min, center = TRUE, scale = TRUE)
16 #pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
17 #df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(n13_train_scz_min), condition1=condition1)
18 #ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
19 #   geom_point(size=8)+
20 #   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
21
22                                     #____ PCA for Z-scaled data
23 pca_data=prcomp(n13_pca_scz_min, center = TRUE, scale = TRUE)
24 pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
25 df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(n13_pca_scz_min), condition1=condition1)
26
27                                     #_____ Plot samples
28 ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
29   geom_point(size=2)+
30   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
31
32                                     #_____ Plot genes
33 #ggplot(df_pca_data, aes(PC1,PC2, color = sample))+geom_point(size=2)+
34 #   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))+
35 #   theme(legend.position = "none")
36
37 #df_pca_data[df_pca_data$PC1>300000,]                                     #____ How many patients/controls
38 #sum(df_pca_data$PC1>300000)                                           # as outliers in the PC1 axis
39
40 #_____ PCA using FactoMineR package
41
42 ##install.packages("FactoMineR")
43 ##library(FactoMineR)
44 ##install.packages("factoextra")
45 ##library(factoextra)
46 #pca_data <- as.data.frame(n13_train_scz_min)
47 #pca_data$group <- c(rep('ctl',261),rep('scz',153))
48 #pca <- PCA(pca_data[,1:(ncol(pca_data)-1)], graph = F)
49 #fviz_pca_ind(pca,
50 #   geom.ind = "point",
51 #   col.ind = pca_data$group,
52 #   palette = c("#00AFBB", "#E7B800"),
53 #   addEllipses = TRUE,
54 #   legend.title = "Groups")
55

```

```

1 %%R
2
3 datExpr = n13_pca_scz_min
4
5 nGenes <- ncol(datExpr)
6 nSamples <- nrow(datExpr)
7
8 datTraits <- n13_pca_path

```

```

1 %%R                                     #_____ A. Run the codes separately to create plots
2                                     # Add each one of these plots beneath the
3 # Setting the colors to match the regional colors
4 region_labels <- RColorBrewer::brewer.pal(6, "Set1")[1:3]
5
6 # Making sure the regional colors match what has been used throughout and making a column with the hex codes.
7
8 datTraits <- datTraits %>%
9   mutate(cell.names = row.names(datTraits)) %>%
10  mutate(x = ifelse(datTraits$sex == "1", region_labels[1],
11    ifelse(datTraits$sex == "0", region_labels[2], NA)))
12
13 # Re-performing heirarchical clustering
14 sampleTree2 <- hclust(dist(datExpr), method = "average")
15
16 # Writing out that dendrogram to a PDF
17 #pdf(file = file.path(file_dir, "sample.dendrogram_region.heatmap.pdf"), height = 8, width = 12)
18 plotDendroAndColors(sampleTree2, datTraits$x,
19   groupLabels = names(datTraits),
20   main = "Sample dendrogram and sex")
21 #dev.off()
22

```

```

1 %%R                                     # _____ B.
2
3 datTraits <- datTraits %>%
4   mutate(cell.names = row.names(datTraits)) %>%
5   mutate(x = ifelse(datTraits$ethn == "1", region_labels[1],
6                     ifelse(datTraits$ethn == "0", region_labels[2], NA)))
7
8 # Writing out that dendrogram to a PDF
9 #pdf(file = file.path(file_dir, "sample.dendrogram_region.heatmap.pdf"), height = 8, width = 12)
10 plotDendroAndColors(sampleTree2, datTraits$x,
11                      groupLabels = names(datTraits),
12                      main = "Sample dendrogram and ethnicity")
13

```

```

1 %%R                                     # _____ C.
2
3 datTraits <- datTraits %>%
4   mutate(cell.names = row.names(datTraits)) %>%
5   mutate(x = ifelse(datTraits$region == "1", region_labels[1],
6                     ifelse(datTraits$region == "0", region_labels[2], NA)))
7
8 # Writing out that dendrogram to a PDF
9 #pdf(file = file.path(file_dir, "sample.dendrogram_region.heatmap.pdf"), height = 8, width = 12)
10 plotDendroAndColors(sampleTree2, datTraits$x,
11                      groupLabels = names(datTraits),
12                      main = "Sample dendrogram and region")
13

```

```

1 %%R                                     # _____ D.
2
3 datTraits <- datTraits %>%
4   mutate(cell.names = row.names(datTraits)) %>%
5   mutate(x = ifelse(datTraits$diag == "1", region_labels[1],
6                     ifelse(datTraits$diag == "0", region_labels[2], NA)))
7
8 # Writing out that dendrogram to a PDF
9 #pdf(file = file.path(file_dir, "sample.dendrogram_region.heatmap.pdf"), height = 8, width = 12)
10 plotDendroAndColors(sampleTree2, datTraits$x,
11                      groupLabels = names(datTraits),
12                      main = "Sample dendrogram and diagnosis")
13

```

```

1 %%R                                     # _____ E.
2
3 xs=quantile(datTraits$age,c(0,1/3,2/3,1))
4 xs[1]=xs[1]-.00005
5 datTraits <- datTraits %>% mutate(category=cut(age, breaks=xs,
6 labels=c("low","middle","high"),ordered_result = TRUE))
7 boxplot(datTraits$age~datTraits$category,col=3:5)
8

```

```

1 %%R
2
3 datExpr <- datExpr[,which(apply(datExpr, 2, var) != 0)]
4 datExpr = t(datExpr)
5 colnames(datExpr) <- c(rownames(datTraits))
6
7 datTraits$diag <- as.factor(datTraits$diag)
8 datTraits$ethn <- as.factor(datTraits$ethn)
9 datTraits$sex <- as.factor(datTraits$sex)
10 datTraits$age <- as.factor(datTraits$category)
11 datTraits$region <- as.factor(datTraits$region)
12
13 datExpr = t(datExpr)
14

```

```

1 %%R                                     # _____ SVA analysis
2
3
4 if (!requireNamespace('BiocManager', quietly = TRUE))
5   install.packages('BiocManager')
6
7 BiocManager::install('sva')

```

```

8 library(sva)
9
10 mod0 = model.matrix(~as.factor(ethn)+as.factor(sex)+as.factor(age)+as.factor(region), data=datTraits)
11 mod = model.matrix(~as.factor(diag)+as.factor(ethn)+as.factor(sex)+as.factor(age)+as.factor(region), data=datTraits)
12 n.sv = num.sv(t(datExpr),mod,method="leek")
13 svobj = sva(t(datExpr),mod,mod0,n.sv=n.sv)
14
15 cleanY = function(y, mod, svobj) {
16   X = cbind(mod, svobj)
17   Hat = solve(t(X) %*% X) %*% t(X)
18   beta = (Hat %*% t(y))
19   rm(Hat)
20   gc()
21   P = ncol(mod)
22   return(y - t(as.matrix(X[,-c(1:P)])) %*% beta[-c(1:P),]))
23 }
24
25 datExpr_cln <- cleanY(t(datExpr), mod, svobj$sv)
26
27 datExpr_cln <- t(datExpr_cln)
28
29 #_____
30 #mod0 = model.matrix(~as.factor(Gender)+as.factor(Population), data=pheno)
31 #mod = model.matrix(~as.factor(inversion_genotype)+as.factor(Gender)+as.factor(Population), data=pheno)
32 #n.sv = num.sv(edata,mod,method="leek")
33 #svobj = sva(edata,mod,mod0,n.sv=n.sv)
34

```

```

1 ##R
2 #_____ plot PCA after SVA analysis!!!!!!!!!!!!
3
4 #datExpr_cln <- t(datExpr_cln)
5
6 library(ggplot2)
7
8 #crr1 <- crr1[which(apply(datExpr_cln, 2, var) != 0),]
9 #datExpr_cln <- datExpr_cln[,which(apply(datExpr_cln, 2, var) != 0)]#_Remove zero variance columns from the new dataset
10
11 genes=paste("gene",seq(1:dim(datExpr_cln)[2]), sep="")
12 colnames(datExpr_cln)=genes
13 row.names(datExpr_cln)=c(paste0("ctl_",seq(1:261)), paste0("scz",seq(1:153))) #__Python command: DER01_BrSeq['C'].value_c
14 condition1=rep(c("ctl"), each=261)
15 condition2=rep(c("scz"), each=153)
16 condition1 <- append(condition1, condition2)
17 length(condition1)
18
19 #_____ PCA for Z-scaled data
20 pca_data=prcomp(datExpr_cln, center = TRUE, scale = TRUE)
21 pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
22 df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(datExpr_cln), condition1=condition1)
23
24 #_____Plot samples
25 ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
26   geom_point(size=2)+
27   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))

```

```

1 #_____ WGCNA_____

```

```

1 #_____ WGCNA_____

```

```

1 #_____ WGCNA_____

```

```

1 ##R
2
3 sampleTree = hclust(dist(datExpr_cln), method = "average");
4 par(cex = 0.6);
5 par(mar = c(0,4,2,0))
6 plot(sampleTree, main = "Sample clustering to detect outliers",
7      sub="", xlab="", cex.lab = 1.5,cex.axis = 1.5, cex.main = 2)
8
9 abline(h = 160, col = "red");
10 clust = cutreeStatic(sampleTree, cutHeight = 160, minSize = 1)
11 table(clust)
12 keepSamples1 = (clust==1)

```

```

13 print(sum(keepSamples1))
14 keepSamples2 = (clust==2)
15 print(sum(keepSamples2))
16
17 keepSamples <- list()
18 keepSamples <- c(keepSamples, keepSamples1+keepSamples2)
19 length(keepSamples)
20
21 dim(datExpr_cln)
22 datExpr = datExpr_cln[which(keepSamples != 0),]
23 print(dim(datExpr))
24
25 nGenes <- ncol(datExpr)
26 nSamples <- nrow(datExpr)
27
28 datTraits <- n13_pca_path[which(keepSamples != 0),]
29 print(dim(datTraits))
30

```

```

1 %%R
2
3 # Setting what soft thresholds (powers) should be tested for potential use as a soft threshold
4 powers <- c(c(1:10), seq(from = 12, to=30, by=2))
5
6 # Calculating soft thresholds for a signed network
7 #sft <- pickSoftThreshold(datExpr, powerVector = powers,
8 #                          verbose = 5, corFnc="bicor",
9 #                          corOptions = list(use = 'p', maxPOutliers = 0.05),
10 #                          networkType ="signed hybrid",
11 #                          moreNetworkConcepts = F)
12
13 # Calculating soft thresholds for a signed network
14 sft <- pickSoftThreshold(datExpr, powerVector = powers,
15                           verbose = 5, corFnc="cor",
16                           corOptions = list(use = 'p'),
17                           networkType ="signed")

```

```

1 %%R
2
3 # Plotting the soft threshold diagnostics in order to choose an optimal soft threshold.
4 # Setting display margins
5 par(mfrow = c(1,1))
6 cex1 = 0.9
7
8 # Scale-free topology fit index as a function of the soft-thresholding power. We are looking to choose a soft threshold whe
9 plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
10      xlab="Soft Threshold (power)", ylab="Scale Free Topology Model Fit, signed R^2", type="n",
11      main = paste("Scale independence"))
12 text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
13      labels=powers, cex=cex1, col="red")
14 abline(h=0.9, col="red") # this line corresponds to using an R^2 cut-off of h that could be used as a guide to picking a sof

```

```

1 %%R
2
3 # Mean connectivity as a function of the soft-thresholding power. Again, we are looking at when the plot begins leveling ou
4 plot(sft$fitIndices[,1], sft$fitIndices[,5],
5      xlab="Soft Threshold (power)", ylab="Mean Connectivity", type="n",
6      main = paste("Mean connectivity"))
7 text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=cex1, col="red")

```

```

1 %%R
2
3 table(clust)
4

```

```

1 # _____ defining new number of CTLs and SCZs after
2 # Use it as input to the PCA below
3
4 #I. BrainSeq DATASET(from Lieber Institute)
5 #DER01_BrSeq = pd.DataFrame(data1, columns=['A', 'B', 'C', 'D', 'E', 'F', 'G'])
6 #%%R -i DER01_BrSeq
7 #%%R -o DER02_BrSeq DER02_BrSeq <- DER01_BrSeq[which(keepSamples != 0),]
8 #pd.DataFrame(DER02_BrSeq['C']).value_counts()
9

```

```

10 %R -o datTraits
11 print(pd.DataFrame(datTraits['diag']).value_counts())
12
13 '''
14                                     #II. CAPSTONE DATASET(from Kelsey)
15 %R -i capstone_BrSeq2_new
16 %R -o capstone_BrSeq3_new capstone_BrSeq3_new <- capstone_BrSeq2_new[which(keepSamples != 0),]
17 pd.DataFrame(capstone_BrSeq3_new['B']).value_counts()
18 '''

```

```

1 %%R
2 #_____ plot PCA after removing outliers!!!!!!!!!!
3
4 library(ggplot2)
5
6 row.names(datExpr)=c(paste0("ctl_",seq(1:255)), paste0("scz",seq(1:152))) #__Python command: DER01_BrSeq['C'].value_count
7 condition1=rep(c("ctl"), each=255)
8 condition2=rep(c("scz"), each=152)
9 condition1 <- append(condition1, condition2)
10 length(condition1)
11
12 #pca_data=prcomp(datExpr, center = TRUE, scale = TRUE)
13 #pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
14 #df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(datExpr), condition1=condition1)
15 #ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
16 #   geom_point(size=8)+
17 #   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
18
19                                     #____ PCA for Z-scaled data
20 pca_data=prcomp(datExpr, center = TRUE, scale = TRUE)
21 pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
22 df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(datExpr), condition1=condition1)
23
24                                     #_____Plot samples
25 ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
26   geom_point(size=2)+
27   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
28
29                                     #_____Plot genes
30 #ggplot(df_pca_data, aes(PC1,PC2, color = sample))+geom_point(size=2)+
31 #   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))+
32 #   theme(legend.position = "none")
33 #_____ PCA using FactoMineR package
34
35 ##install.packages("FactoMineR")
36 ##library(FactoMineR)
37 ##install.packages("factoextra")
38 ##library(factoextra)
39 #pca_data <- as.data.frame(datExpr)
40 #pca_data$group <- c(rep('ctl',250),rep('scz',139))
41 #pca <- PCA(pca_data[,1:(ncol(pca_data)-1)], graph = F)
42 #fviz_pca_ind(pca,
43 #   geom.ind = "point",
44 #   col.ind = pca_data$group,
45 #   palette = c("#00AFBB", "#E7B800"),
46 #   addEllipses = TRUE,
47 #   legend.title = "Groups")
48

```

```

1 %%R
2
3 library(devtools)
4 install_github("vqv/ggbiplot")
5 library(ggbiplot)
6 library(ggplot2)
7
8 # Plot the PCA plots
9 hvPCA1<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=as.factor(datTraits$diag), ellipse=T,var.axes=F) + scale_color_manual
10 hvPCA2<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=as.factor(datTraits$ethn), ellipse=T,var.axes=F) + scale_color_manual
11 hvPCA3<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=as.factor(datTraits$sex), ellipse=T,var.axes=F) + scale_color_manual
12 hvPCA4<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=as.factor(datTraits$region), ellipse=T,var.axes=F) + scale_color_manual
13

```

```

1 %%R
2

```

```
3 # Show the diagnosis plot
4 hvPCA1
```

```
1 %%R
2
3 # Show the ethnicity plot
4 hvPCA2
```

```
1 %%R
2
3 # Show the sex plot
4 hvPCA3
```

```
1 %%R
2
3 # Show the region plot
4 hvPCA4
```

```
1 %%R
2
3 xs=quantile(as.numeric(as.character(datTraits$age)),c(0,1/3,2/3,1))
4 xs[1]=xs[1]-.00005
5
6 datTraits <- datTraits %>% mutate(category=cut(as.numeric(as.character(datTraits$age)), breaks=xs,
7   labels=c("low","middle","high"),ordered_result = TRUE))
8
9 datExpr = t(datExpr)
10 colnames(datExpr) <- c(rownames(datTraits ))
11
12 datTraits$diag <- as.factor(datTraits$diag)
13 datTraits$ethn <- as.factor(datTraits$ethn)
14 datTraits$sex <- as.factor(datTraits$sex)
15 datTraits$age <- as.factor(datTraits$category)
16 datTraits$region <- as.factor(datTraits$region)
17
18 datExpr = t(datExpr)
19
20 hvPCA5<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=as.factor(datTraits$age), ellipse=T,var.axes=F) + scale_color_manual
21
22
23 #_____
24 # Show the age plot
25
26 hvPCA5
27
28
```

```
1
2
3
4
5 #_____ Additional PCA analysis
6
7
8
9
10
```

```
1 %%R
2 #_____ Additional PCA analysis
3
4 #load(file = '/content/drive/My Drive/datasets/m13_train_scz_min')
5 #load(file = '/content/drive/My Drive/datasets/m13_trait_path.RData')
6 #load(file = '/content/drive/My Drive/datasets/crr1.RData')
7
8 #m13_train_scz_min = t(m13_train_scz_min)
9 #colnames(m13_train_scz_min) <- c(rownames(m13_trait_path))
10 #rownames(m13_train_scz_min) <- t(crr1)
11 #m13_trait_path$diag <- as.factor(m13_trait_path$diag)
12 #m13_train_scz_min = t(m13_train_scz_min)
13 #_____ PRCOMP (& randomForest prediction)
14
15 install.packages("randomForest")
```

```

16 install.packages("caret")
17 library(caret)
18 library(randomForest)
19
20 colnames(datExpr) <- t(crr1)
21 #datTraits$diag <- as.factor(datTraits$diag)
22
23 set.seed(1234)
24 new_train_scz_min <- datExpr[sample(nrow(datExpr)),]
25 new_trait_path <- datTraits[sample(nrow(datTraits)),]
26 new_trait_path <- new_trait_path[match(rownames(new_train_scz_min), rownames(new_trait_path)),]
27 set.seed(NULL)
28
29 pca.train <- new_train_scz_min[1:(as.integer(0.7*nrow(new_train_scz_min))),]
30 pca.test <- new_train_scz_min[-(1:(as.integer(0.7*nrow(new_train_scz_min))),),]
31 pca_trait_train <- new_trait_path[1:(as.integer(0.7*nrow(new_trait_path))),]
32 pca_trait_test <- new_trait_path[-(1:(as.integer(0.7*nrow(new_trait_path))),),]
33 pca.test <- pca.test[,which(apply(pca.train, 2, var) != 0)]
34 pca.train <- pca.train[,which(apply(pca.train, 2, var) != 0)]
35 prin_comp <- prcomp(pca.train, scale. = T)
36
37 #std_dev <- prin_comp$sdev                                #_____ Plots
38 #pr_var <- std_dev^2
39 #prop_varex <- pr_var/sum(pr_var)
40
41 #plot(prop_varex, xlab = "Principal Component",
42 #      ylab = "Proportion of Variance Explained",
43 #      type = "b")
44
45 #plot(cumsum(prop_varex), xlab = "Principal Component",
46 #      ylab = "Cumulative Proportion of Variance Explained",
47 #      type = "b")
48
49 #_____randomForest
50 train.data <- data.frame(diag = pca_trait_train$diag, prin_comp$x)
51 train.data <- train.data[,1:31]
52
53 RFmodel_Fit <- randomForest(diag ~ .,data = train.data, method = "anova")
54 RFmodel_Fit
55 test.data <- predict(prin_comp, newdata = pca.test)
56 test.data <- as.data.frame(test.data)
57 test.data <- test.data[,1:30]
58 RFmodel_Fit.prediction <- predict(RFmodel_Fit, test.data)
59 print(confusionMatrix(RFmodel_Fit.prediction, pca_trait_test$diag)$overall[1])
60 print(RFmodel_Fit$confusion)
61
62 #_____ PCATOOLS
63
64 if (!requireNamespace('BiocManager', quietly = TRUE))
65   install.packages('BiocManager')
66
67 BiocManager::install('PCAtools')
68 library(PCAtools)
69
70 #_____ PCATOOLS(various Plotloadings and ENSEMBL to SYMBOL translation)
71
72 organism = "org.Hs.eg.db"
73 BiocManager::install(organism, character.only = TRUE, force = TRUE)
74 library(organism, character.only = TRUE)
75
76 require(org.Hs.eg.db)
77 mapping <- mapIds(
78   org.Hs.eg.db,
79   keys = colnames(datExpr),
80   column = 'SYMBOL',
81   keytype = 'ENSEMBL')
82 colnames(datExpr) <- make.unique(ifelse(is.na(mapping), colnames(datExpr), mapping))
83
84 p <- pca(t(datExpr), metadata = datTraits, center = TRUE,
85         scale = TRUE, removeVar = 0.0)
86

```

```

1 %R
2
3 biplot(p, showLoadings = TRUE, labSize = 5, pointSize = 5, sizeLoadingsNames = 5)    #_ 1.
4

```

```

1 %%R
2
3 pairsplot(p)                                     #_ 2.
4

```

```

1 %%R
2
3 eigencorplot(p,                                   #_ 3.
4 components = getComponents(p, 1:5),
5 metavars = c("diag", "sex", "ethn", "age"),
6 col = c('darkblue', 'blue2', 'black', 'red2', 'darkred'),
7 cexCorval = 0.7,
8 colCorval = 'white',
9 fontCorval = 2,
10 posLab = 'bottomleft',
11 rotLabX = 45,
12 posColKey = 'top',
13 cexLabColKey = 1.5,
14 scale = TRUE,
15 main = bquote(Principal ~ component ~ Pearson ~ r^2 ~ clinical ~ correlates),
16 plotRquared = TRUE,
17 corFUN = 'pearson',
18 corUSE = 'pairwise.complete.obs',
19 corMultipleTestCorrection = 'BH',
20 signifSymbols = c('****', '***', '**', '*', ''),
21 signifCutoffs = c(0, 0.0001, 0.001, 0.01, 0.05, 1))
22

```

```

1 %%R
2
3 plotloadings(p,                                   #_ 4a.
4   components = getComponents(p, c(1)),
5   rangeRetain = 0.01, absolute = F,
6   drawConnectors = TRUE, labSize = 4) + coord_flip()
7

```

```

1 %%R
2
3 plotloadings(p,                                   #_ 4b.
4   components = getComponents(p, c(2)),
5   rangeRetain = 0.01, absolute = F,
6   drawConnectors = TRUE, labSize = 4) + coord_flip()

```

```

1 %%R
2
3 plotloadings(p,                                   #_ 4c.
4   components = getComponents(p, c(3)),
5   rangeRetain = 0.01, absolute = F,
6   drawConnectors = TRUE, labSize = 4) + coord_flip()

```

```

1 %%R
2
3 plotloadings(p,                                   #_ 4d.
4   components = getComponents(p, c(4)),
5   rangeRetain = 0.01, absolute = F,
6   drawConnectors = TRUE, labSize = 4) + coord_flip()

```

```

1 %%R
2
3 plotloadings(p,                                   #_ 4e.
4   components = getComponents(p, c(5)),
5   rangeRetain = 0.01, absolute = F,
6   drawConnectors = TRUE, labSize = 4) + coord_flip()

```

```

1 %%R
2
3 plotloadings(p,                                   #_ 5.
4   components = getComponents(p, c(1,2,3,4)),
5   rangeRetain = 0.1,
6   labSize = 3.0,
7   absolute = FALSE,

```

```

8   title = 'Loadings plot',
9   subtitle = 'Misc PCs',
10  caption = 'Top 10% variables',
11  shape = 23, shapeSizeRange = c(1, 16),
12  drawConnectors = FALSE)
13

```

```

1  %%R
2
3  plotloadings(p,                                     #_ 6.
4  components = getComponents(p, c(1,2,3,4,5)),
5  rangeRetain = 0.001,
6  labSize = 3.0,
7  title = 'Loadings plot',
8  subtitle = 'PC1, PC2, PC3, PC4, PC5',
9  caption = 'Top 1% variables',
10 drawConnectors = TRUE)
11

```

```

1  %%R                                     # _____ Create dataframe with the top genes of the
2                                     # Use it as input to Reactome enrichment ana
3
4  ph_all = list()
5  pl_all = list()
6
7  p1h<-order(p$loadings[1], decreasing=T)[1:30]
8  p1l<-order(p$loadings[1], decreasing=F)[1:30]
9
10 p2h<-order(p$loadings[2], decreasing=T)[1:30]
11 p2l<-order(p$loadings[2], decreasing=F)[1:30]
12
13 p3h<-order(p$loadings[3], decreasing=T)[1:30]
14 p3l<-order(p$loadings[3], decreasing=F)[1:30]
15
16 p4h<-order(p$loadings[4], decreasing=T)[1:30]
17 p4l<-order(p$loadings[4], decreasing=F)[1:30]
18
19 p5h<-order(p$loadings[5], decreasing=T)[1:30]
20 p5l<-order(p$loadings[5], decreasing=F)[1:30]
21
22 ph_all[[1]] <- rownames(p$loadings[p1h,])
23 ph_all[[2]] <- rownames(p$loadings[p2h,])
24 ph_all[[3]] <- rownames(p$loadings[p3h,])
25 ph_all[[4]] <- rownames(p$loadings[p4h,])
26 ph_all[[5]] <- rownames(p$loadings[p5h,])
27
28 pl_all[[1]] <- rownames(p$loadings[p1l,])
29 pl_all[[2]] <- rownames(p$loadings[p2l,])
30 pl_all[[3]] <- rownames(p$loadings[p3l,])
31 pl_all[[4]] <- rownames(p$loadings[p4l,])
32 pl_all[[5]] <- rownames(p$loadings[p5l,])
33

```

```

1
2
3
4
5 # _____ Additional PCA analysis
6
7
8
9
10

```

```

1  %%R
2
3  elbow <- findElbowPoint(p$variance)
4
5  screepplot(p,components = getComponents(p, 1:20),vline = c(elbow)) +
6    geom_label(aes(x = elbow + 1, y = 50,
7      label = 'Elbow method', vjust = -1, size = 8))
8
9  #datExpr = t(datExpr)

```

```

1 %%R
2
3 install.packages("tsne")
4 ##install.packages("Rtsne")
5 library(tsne)
6 ##library(Rtsne)
7 library(ggplot2)
8
9 nComponents <- 6
10 seed <- 1234
11 set.seed(seed)
12
13 dat.filter.BCV.tsne<-tsne(pca_data$x[,1:nComponents],perplexity=30,max_iter=2000,whiten = FALSE)
14
15 tsne_plot <- data.frame(x = dat.filter.BCV.tsne[,1], y = dat.filter.BCV.tsne[,2], col = datTraits$diag)
16 ggplot(tsne_plot) + geom_point(aes(x=x, y=y, color=col))
17

```

```
1 #- - - - -
```

```
1 #- - - - -
```

```
1 #- - - - -
```

```

1 %%R
2
3 BiocManager::install("WGCNA")
4 library(WGCNA)
5
6 cor <- WGCNA::cor
7
8 net = blockwiseModules(datExpr, power = 12,
9 TOMType = "signed", minModuleSize = 50, deepSplit = 4,
10 reassignThreshold = 0, mergeCutHeight = 0.1,
11 numericLabels = TRUE, pamRespectsDendro = FALSE,
12 saveTOMs = FALSE, maxBlockSize = 16000,networkType = "signed",
13 saveTOMFileBase = "femaleMouseTOM",
14 verbose = 3)
15
16 #net = blockwiseModules(datExpr, power = 12,
17 #TOMType = "signed", minModuleSize = 30, corType = "bicor",
18 #reassignThreshold = 0, mergeCutHeight = 0.1, deepSplit = 4,
19 #numericLabels = TRUE, pamStage = FALSE, pamRespectsDendro = FALSE,
20 #saveTOMs = FALSE, maxBlockSize = 16000, maxPOutliers = 0.05,networkType = "signed hybrid",
21 #saveTOMFileBase = "femaleMouseTOM", robustY = FALSE, #consensusQuantile = 0.2,
22 #verbose = 3, pearsonFallback = "individual")
23
24 print(names(net))
25
26 print(table(net$colors))
27
28 # Convert labels to colors for plotting
29 mergedColors = labels2colors(net$colors)
30 # Plot the dendrogram and the module colors underneath
31 plotDendroAndColors(net$dendrograms[[1]], mergedColors[net$blockGenes[[1]]],
32 "Module colors",
33 dendroLabels = FALSE, hang = 0.03,
34 addGuide = TRUE, guideHang = 0.05)
35
36 moduleLabels = net$colors
37 moduleColors = labels2colors(net$colors)
38 MES = net$MEs;
39 geneTree = net$dendrograms[[1]];
40 #save(MES, moduleLabels, moduleColors, mergedColors, geneTree, file = "/content/drive/My Drive/datasets/scz_networkConstruc
41

```

```

1 %%R
2
3 #load("/content/drive/My Drive/datasets/scz_networkConstruction-auto.RData")
4
5

```

```

1 # _____ Plot one dendrogram for each block
2 %%R
3
4 # Load the results of single-block analysis
5 load(file = "/content/drive/My Drive/datasets/scz_networkConstruction-auto.RData");
6 # Relabel blockwise modules
7 bwLabels = matchLabels(net$colors, moduleLabels);
8 # Convert labels to colors for plotting
9 bwModuleColors = labels2colors(bwLabels)
10
11 # Plot the dendrogram and the module colors underneath for block 1
12 plotDendroAndColors(net$dendrograms[[1]], bwModuleColors[net$blockGenes[[1]]],
13 "Module colors", main = "Gene dendrogram and module colors in block 1",
14 dendroLabels = FALSE, hang = 0.03,
15 addGuide = TRUE, guideHang = 0.05)
16
17 # Plot the dendrogram and the module colors underneath for block 2
18 plotDendroAndColors(net$dendrograms[[2]], bwModuleColors[net$blockGenes[[2]]],
19 "Module colors", main = "Gene dendrogram and module colors in block 2",
20 dendroLabels = FALSE, hang = 0.03,
21 addGuide = TRUE, guideHang = 0.05)
22
23 # Plot the dendrogram and the module colors underneath for block 3
24 plotDendroAndColors(net$dendrograms[[3]], bwModuleColors[net$blockGenes[[3]]],
25 "Module colors", main = "Gene dendrogram and module colors in block 3",
26 dendroLabels = FALSE, hang = 0.03,
27 addGuide = TRUE, guideHang = 0.05)

```

```

1 %%R
2
3 if (!requireNamespace("BiocManager", quietly = TRUE))
4   install.packages("BiocManager")
5
6 BiocManager::install("biomaRt", force = TRUE)
7 library(biomaRt)
8
9 mart <- useMart("ENSEMBL_MART_ENSEMBL")
10 mart <- useDataset("hsapiens_gene_ensembl", mart)
11
12 devtools::install_github(c("GuangchuangYu/DOSE", "GuangchuangYu/enrichplot", "GuangchuangYu/GOSemSim", "eliocamp/ggnewscale")
13
14 BiocManager::install("clusterProfiler")
15 library(clusterProfiler)
16
17 BiocManager::install("ReactomePA")
18 library(ReactomePA)
19
20 organism = "org.Hs.eg.db"
21 BiocManager::install(organism, character.only = TRUE)
22 library(organism, character.only = TRUE)
23

```

```

1 %%R
2
3 print(table(mergedColors))
4
5 print(which(as.data.frame(table(mergedColors)) == "turquoise", arr.ind = TRUE))
6 print(which(as.data.frame(table(mergedColors)) == "grey", arr.ind = TRUE))
7
8 print(which(as.data.frame(table(mergedColors)) == "lightcyan1", arr.ind = TRUE))
9 print(which(as.data.frame(table(mergedColors)) == "red", arr.ind = TRUE))
10 print(which(as.data.frame(table(mergedColors)) == "paleturquoise", arr.ind = TRUE))
11 print(which(as.data.frame(table(mergedColors)) == "navajowhite2", arr.ind = TRUE))
12 print(which(as.data.frame(table(mergedColors)) == "lightgreen", arr.ind = TRUE))
13 print(which(as.data.frame(table(mergedColors)) == "brown", arr.ind = TRUE))
14 print(which(as.data.frame(table(mergedColors)) == "cyan", arr.ind = TRUE))
15

```

```

1 %%R
2
3 datTraits <- n13_pca_path[which(keepSamples != 0),]
4
5 diagnosis = as.data.frame(datTraits$diag)
6 names(diagnosis) = "diagnosis"
7 MEs0 = moduleEigengenes(datExpr, mergedColors)$eigengenes

```

```

8 MESFemale = orderMEs(MEs0)
9 modTraitCor = cor(MESFemale, datTraits, use = "p")
10 modTraitP = corPvalueStudent(modTraitCor, nSamples)
11 # Since we have a moderately large number of modules and traits, a
12 # suitable graphical representation will help in reading the table. We
13 # color code each association by the correlation value: Will display
14 # correlations and their p-values
15 textMatrix = paste(signif(modTraitCor, 2), "\n(", signif(modTraitP, 1), ")",
16   sep = "")
17 dim(textMatrix) = dim(modTraitCor)
18 par(mar = c(6, 8.5, 3, 3))
19 # Display the correlation values within a heatmap plot
20 labeledHeatmap(Matrix = modTraitCor, xLabels = names(datTraits), yLabels = names(MESFemale),
21   ySymbols = names(MESFemale), colorLabels = FALSE, colors = greenWhiteRed(50),
22   textMatrix = textMatrix, setStdMargins = FALSE, cex.text = 0.5, zlim = c(-1,
23     1), main = paste("Module-trait relationships"))
24
25                                     #The resulting color-coded table is shown in
26                                     #the several significant module-trait associ
27                                     #(first column) as the trait of interest.
28
29                                     #Caption: Table of module-trait correlations
30                                     #correlation (and p-value) resulting from cc
31                                     #traits (columns). The table is color-coded
32                                     #legend.

```

```

1 %R
2
3 #_____ Plot Module Significance in a Barplot
4
5 GS1=as.numeric(cor(datTraits$diag,datExpr, use="p"))
6 GeneSignificance=abs(GS1)
7 # Next module significance is defined as average gene significance.
8 ModuleSignificance=tapply(GeneSignificance, mergedColors, mean, na.rm=T)
9
10 par(mfrow = c(1,1))
11 plotModuleSignificance(GeneSignificance,mergedColors)

```

```

1 %R
2
3 library("data.table")
4
5 p.sig = signif(cor(datTraits$diag,MES0, use="p"),2)
6 #cor.test(as.numeric(datTraits$diag), MES0$MEbrown)
7 p.values = corPvalueStudent(cor(datTraits$diag,MES0, use="p"), nSamples = length(datTraits$diag))
8
9
10 p.values_tmp = as.data.frame(apply(p.values, 2, unlist))
11 colnames(p.values_tmp)[1] <- "name2"
12 p.values_tmp <- setDT(p.values_tmp, keep.rownames = TRUE)[]
13 colnames(p.values_tmp)[1] <- "name1"
14
15 data_mod1 <- p.values_tmp[order(p.values, decreasing = F)]
16 data_mod1 <- data.table(data_mod1)
17 data_mod1 <- data_mod1[ , head(.SD, 10)]
18 print(data_mod1)
19
20 print("_____")
21
22 p.sig_tmp = as.data.frame(apply(p.sig, 2, unlist))
23 colnames(p.sig_tmp)[1] <- "name2"
24 p.sig_tmp <- setDT(p.sig_tmp, keep.rownames = TRUE)[]
25 colnames(p.sig_tmp)[1] <- "name1"
26 head(p.sig_tmp)
27
28
29 data_mod2 <- p.sig_tmp[order(abs(p.sig), decreasing = T)]
30 data_mod2 <- data.table(data_mod2)
31 data_mod2 <- data_mod2[ , head(.SD, 10)]
32 print(data_mod2)

```

```

1 %R
2 #_____ calculate the module membership values (aka. module eigengene
3
4                                     #_____ Caption: Gene significance (GS.weight) ver

```

```

5                                     #_____ for the body weight related modules. GS.we
6                                     #_____ correlated reflecting the high correlator
7                                     #_____ respective module eigengenes. We find that
8                                     #_____ contain genes that have high positive and
9                                     #_____ with body weight. In contrast, the grey "t
10                                    #_____ correlations with weight.
11
12 # Next use this trait to define a gene significance variable
13 GS.diagnosis = as.numeric(cor(datExpr, diagnosis, use = "p"))
14 # This translates the numeric values into colors
15 GS.diagnosisColor = numbers2colors(GS.diagnosis, signed = T)
16 blocknumber = 1
17 #datColors = data.frame(mergedColors, GS.diagnosisColor)[net$blockGenes[[blocknumber]], ]
18
19 datKME = signedKME(datExpr, MEsFemale)
20
21 colorOfColumn = substring(names(datKME), first = 4)
22 selectModules = c("lightcyan1","red","paleturquoise","turquoise","navajowhite2","lightgreen","plum4","brown","cyan","grey")
23 for (module in selectModules) {
24     column = match(module, colorOfColumn)
25     restModule = mergedColors == module
26     verboseScatterplot(datKME[restModule, column], GS.diagnosis[restModule], xlab = paste("Module Membership ",
27         module, "module"), ylab = "diagnosis", main = paste("kME.", module,
28         "vs. diag"), col = module)
29 }
30

```

```

1 %%R
2
3 #restGenes_tmp= (mergedColors != "grey" & mergedColors != "turquoise")
4 restGenes_tmp= (mergedColors != "lightcyan1" & mergedColors != "red" & mergedColors != "paleturquoise" & mergedColors != "r
5
6 ADJ1=abs(cor(datExpr[,!restGenes_tmp],use="p"))^6
7 Alldegrees1=intramodularConnectivity(ADJ1, mergedColors[!restGenes_tmp])
8 head(Alldegrees1)
9
10 par(mar = c(4,5,3,1))
11 colorOfColumn = substring(names(datKME), first = 4)
12 selectModules = c("lightcyan1","red","paleturquoise","navajowhite2","lightgreen","brown","cyan")
13 for (module in selectModules) {
14     restModule = mergedColors == module
15     verboseScatterplot(Alldegrees1$kWithin[restModule],
16     GeneSignificance[restModule], col=mergedColors[restModule],
17     main=module,
18     xlab = "Connectivity", ylab = "Gene Significance", abline = TRUE)
19 }
20
21 #_____
22 #selectModules = c("lightpink3")
23 #for (module in selectModules) {
24 #restModule = mergedColors == module
25 #print(Alldegrees1$kWithin[restModule])
26 #meta=na.omit(Alldegrees1$kWithin[restModule])
27 #}
28 #length(meta)

```

```

1 %%R
2                                     #_____ Our previous analysis has shown that the t
3                                     #_____ "interesting" module in that its module si
4                                     #_____ Here we show how to find genes with high g
5                                     #_____ high intramodular connectivity in the brow
6 #print(names(datKME)) #_____ check names list
7
8 FilterGenes1= abs(GS1)> .2 & abs(datKME$kMEpink)>.8
9 table(FilterGenes1)
10 dimnames(data.frame(datExpr))[[2]][FilterGenes1]
11
12 FilterGenes2 = abs(GS1)> .2 & abs(datKME$kMEantiquewhite1)>.8
13 table(FilterGenes2)
14 dimnames(data.frame(datExpr))[[2]][FilterGenes2]
15
16 FilterGenes3= abs(GS1)> .2 & abs(datKME$kMERed)>.8
17 table(FilterGenes3)
18 dimnames(data.frame(datExpr))[[2]][FilterGenes3]
19
20 FilterGenes4= abs(GS1)> .2 & abs(datKME$kMEcoral4)>.8

```

```

21 table(FilterGenes4)
22 dimnames(data.frame(datExpr))[[2]][FilterGenes4]
23
24 FilterGenes5= abs(GS1)> .2 & abs(datKME$kMElightpink3)>.8
25 table(FilterGenes5)
26 dimnames(data.frame(datExpr))[[2]][FilterGenes5]
27
28 FilterGenes6= abs(GS1)> .2 & abs(datKME$kMEbrown)>.8
29 table(FilterGenes6)
30 dimnames(data.frame(datExpr))[[2]][FilterGenes6]
31
32 FilterGenes7= abs(GS1)> .2 & abs(datKME$kMEskyblue2)>.8
33 table(FilterGenes7)
34 dimnames(data.frame(datExpr))[[2]][FilterGenes7]
35

```

```

1 %%R
2
3 whichmodule = "turquoise" # The heatmap color-codes the scaled gene e
4 Eigengene = MESFemale$MEturquoise # red corresponds to over-expression
5 datExprModule = datExpr[, mergedColors == whichmodule]
6 # set the margins of the graphics window
7 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
8 # create a heatmap whose columns correspond to the arrays and whose rows
9 # correspond to genes
10 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
11         main = paste("heatmap", whichmodule, "module"))
12
13 # scatter plot between eigengene and sample network connectivity # Scatter plot of the lightgreen module eig
14 par(mfrow = c(1, 1)) #sample network connectivity Z.k, which is
15 verboseScatterplot(Eigengene, Z.k, xlab = paste("ME", whichmodule, sep = "")) #module eigengene takes on an extreme negat
16 abline(h = -2, col = "red", lwd = 2) #(corresponding to the sample where most mc
17 #The red horizontal line in the scatter plc
18
19 whichmodule = "pink"
20 Eigengene = MESFemale$MEpink
21 datExprModule = datExpr[, mergedColors == whichmodule]
22 # set the margins of the graphics window
23 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
24 # create a heatmap whose columns correspond to the arrays and whose rows
25 # correspond to genes
26 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
27         main = paste("heatmap", whichmodule, "module"))
28 # scatter plot between eigengene and sample network connectivity
29 par(mfrow = c(1, 1))
30 verboseScatterplot(Eigengene, Z.k, xlab = paste("ME", whichmodule, sep = ""))
31 abline(h = -2, col = "red", lwd = 2)
32
33 whichmodule = "antiquewhite1"
34 Eigengene = MESFemale$MEantiquewhite1
35 datExprModule = datExpr[, mergedColors == whichmodule]
36 # set the margins of the graphics window
37 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
38 # create a heatmap whose columns correspond to the arrays and whose rows
39 # correspond to genes
40 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
41         main = paste("heatmap", whichmodule, "module"))
42 # scatter plot between eigengene and sample network connectivity
43 par(mfrow = c(1, 1))
44 verboseScatterplot(Eigengene, Z.k, xlab = paste("ME", whichmodule, sep = ""))
45 abline(h = -2, col = "red", lwd = 2)
46
47 whichmodule = "red"
48 Eigengene = MESFemale$MERed
49 datExprModule = datExpr[, mergedColors == whichmodule]
50 # set the margins of the graphics window
51 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
52 # create a heatmap whose columns correspond to the arrays and whose rows
53 # correspond to genes
54 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
55         main = paste("heatmap", whichmodule, "module"))
56 # scatter plot between eigengene and sample network connectivity
57 par(mfrow = c(1, 1))
58 verboseScatterplot(Eigengene, Z.k, xlab = paste("ME", whichmodule, sep = ""))
59 abline(h = -2, col = "red", lwd = 2)
60

```

```

61 whichmodule = "coral4"
62 Eigengene = MEsFemale$MEcoral4
63 datExprModule = datExpr[, mergedColors == whichmodule]
64 # set the margins of the graphics window
65 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
66 # create a heatmap whose columns correspond to the arrays and whose rows
67 # correspond to genes
68 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
69   main = paste("heatmap", whichmodule, "module"))
70 # scatter plot between eigengene and sample network connectivity
71 par(mfrow = c(1, 1))
72 verboseScatterplot(Eigengene, Z.k, xlab = paste("ME", whichmodule, sep = ""))
73 abline(h = -2, col = "red", lwd = 2)
74
75 #
76 par(mfrow=c(3,1), mar=c(1, 2, 4, 1))
77 which.module="turquoise";
78 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),nrgcols=30,rlabels=T,
79   clabels=T,rcols=which.module,
80   title=which.module )
81 # for the second (blue) module we use
82 which.module="pink";
83 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),nrgcols=30,rlabels=T,
84   clabels=T,rcols=which.module,
85   title=which.module )
86 which.module="antiquewhite1";
87 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),nrgcols=30,rlabels=T,
88   clabels=T,rcols=which.module,
89   title=which.module )
90 which.module="red";
91 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),nrgcols=30,rlabels=T,
92   clabels=T,rcols=which.module,
93   title=which.module )
94 which.module="coral4";
95 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),nrgcols=30,rlabels=T,
96   clabels=T,rcols=which.module,
97   title=which.module )
98
99 #
100 which.module="turquoise" # Produces heatmap plot for a module.
101 ME=MEs0[, paste("ME",which.module, sep="")] # Here the rows are genes and the columns a
102 par(mfrow=c(2,1), mar=c(0.3, 5.5, 3, 2)) # Well defined modules results in character
103 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ), # structures since the corresponding genes
104   nrgcols=30,rlabels=F,rcols=which.module,
105   main=which.module, cex.main=2)
106 par(mar=c(5, 4.2, 0, 0.7))
107 barplot(ME, col=which.module, main="", cex.main=2,
108   ylab="eigengene expression",xlab="array sample")
109 #table(sign(ME))
110
111 which.module="pink"
112 ME=MEs0[, paste("ME",which.module, sep="")]
113 par(mfrow=c(2,1), mar=c(0.3, 5.5, 3, 2))
114 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),
115   nrgcols=30,rlabels=F,rcols=which.module,
116   main=which.module, cex.main=2)
117 par(mar=c(5, 4.2, 0, 0.7))
118 barplot(ME, col=which.module, main="", cex.main=2,
119   ylab="eigengene expression",xlab="array sample")
120 #table(sign(ME))
121
122 which.module="antiquewhite1"
123 ME=MEs0[, paste("ME",which.module, sep="")]
124 par(mfrow=c(2,1), mar=c(0.3, 5.5, 3, 2))
125 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),
126   nrgcols=30,rlabels=F,rcols=which.module,
127   main=which.module, cex.main=2)
128 par(mar=c(5, 4.2, 0, 0.7))
129 barplot(ME, col=which.module, main="", cex.main=2,
130   ylab="eigengene expression",xlab="array sample")
131 #table(sign(ME))
132
133 which.module="red"
134 ME=MEs0[, paste("ME",which.module, sep="")]
135 par(mfrow=c(2,1), mar=c(0.3, 5.5, 3, 2))
136 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),
137   nrgcols=30,rlabels=F,rcols=which.module,

```

```

138 main=which.module, cex.main=2)
139 par(mar=c(5, 4.2, 0, 0.7))
140 barplot(ME, col=which.module, main="", cex.main=2,
141 ylab="eigengene expression", xlab="array sample")
142 #table(sign(ME))
143
144 which.module="coral4"
145 ME=MES0[, paste("ME", which.module, sep="")]
146 par(mfrow=c(2,1), mar=c(0.3, 5.5, 3, 2))
147 plotMat(t(scale(datExpr[,mergedColors==which.module ] ) ),
148 nrgcols=30, rlabels=F, rcols=which.module,
149 main=which.module, cex.main=2)
150 par(mar=c(5, 4.2, 0, 0.7))
151 barplot(ME, col=which.module, main="", cex.main=2,
152 ylab="eigengene expression", xlab="array sample")
153 #table(sign(ME))
154
155

```

```

1 %%R
2
3 whichmodule = "turquoise" # The heatmap color-codes the scaled gene ex
4 Eigengene = MESFemale$METurquoise # red corresponds to over-expression
5 datExprModule = datExpr[, mergedColors == whichmodule]
6 # set the margins of the graphics window
7 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
8 # create a heatmap whose columns correspond to the arrays and whose rows
9 # correspond to genes
10 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
11 main = paste("heatmap", whichmodule, "module"))
12

```

```

1 %%R
2
3 # _____ Correlating tSNE position with eigenvector
4 install.packages("tsne")
5 install.packages("ggplot2")
6 library(tsne)
7 library(ggplot2)
8
9 seed <- 1234
10 set.seed(seed)
11
12 whichmodule = "lightcyan1"
13 Eigengene = MESFemale$MElightcyan1
14 datExprModule = datExpr[, mergedColors == whichmodule]
15
16 dat.filter.DCV.tsne<-tsne(datExprModule,perplexity=30,max_iter=2000,whiten = FALSE)
17
18 eigentSNE<-cbind(dat.filter.DCV.tsne[,1],dat.filter.DCV.tsne[,2],Eigengene)
19 colnames(eigentSNE)<-c("tSNE1_pos", "tSNE2_pos", whichmodule)
20
21 p2.subset<-ggplot(data.frame(eigentSNE)) + geom_point(aes(x=tSNE1_pos,y=tSNE2_pos,color = lightcyan1),size=4) + scale_color
22 p2.subset.all <- p2.subset + ggtitle("WGCNA module eigengene expression") + coord_fixed(ratio = 1) + xlab("t-SNE 1") + ylat
23 p2.subset.all
24
25

```

```

1 %%R
2
3 BiocManager::install(c("dynamicTreeCut", "cluster", "flashClust", "Hmisc", "reshape", "foreach", "doParallel") )
4 library(dynamicTreeCut)
5 library(flashClust)
6 library(Hmisc)
7 library(reshape)
8 library(foreach)
9 library(doParallel)

```

```

1 %%R
2 # _____ Remove overrepresented modules (42->grey,1
3 # Calculate eigengenes for each module
4 MES <- moduleEigengenes(datExpr, colors = mergedColors)$eigengenes
5
6 #print(names(MES[120])) # _____ how to remove genes

```

```

7 #print(names(MEs[42]))
8 #MEs <- MEs[,c(1:41,43:119,121:length(table(mergedColors)))]
9
10 print(names(MEs[55]))    #_____ how to add genes
11 print(names(MEs[96]))
12 print(names(MEs[83]))
13 print(names(MEs[77]))
14 print(names(MEs[56]))
15 print(names(MEs[10]))
16 print(names(MEs[19]))
17 MEs <- MEs[,c(55,96,83,77,56,10,19)]
18
19
20

```

```

1 %%R
2 #_____ Remove grey and turquoise for ahead of attempting to look
3 modNames <- substring(names(MEs), 3)
4 #restGenes= (mergedColors != "grey" & mergedColors != "turquoise")
5 restGenes= (mergedColors != "lightcyan1" & mergedColors != "red" & mergedColors != "paleturquoise" & mergedColors != "navaj
6
7 table(!restGenes)
8
9 softPower = 12
10 diss1=1-TOMsimilarityFromExpr(datExpr[,!restGenes], power = softPower, TOMType = "signed")
11
12 colnames(diss1) =rownames(diss1) = modNames[!restGenes]
13 hier1=flashClust(as.dist(diss1), method="average" )
14 plotDendroAndColors(hier1, mergedColors[!restGenes], "Dynamic Tree Cut", dendroLabels = FALSE, hang = 0.03, addGuide = TRUE
15
16

```

```

1 %%R
2
3 install.packages("reshape2")
4 library(ggplot2)
5 library(reshape2)
6
7 if (!requireNamespace('ComplexHeatmap', quietly = TRUE))
8   BiocManager::install('ComplexHeatmap')
9 require(ComplexHeatmap)
10
11 if (!requireNamespace('circlize', quietly = TRUE))
12   BiocManager::install('circlize')
13 require(circlize)
14
15 df <- data.frame(a = mergedColors[!restGenes], c = c(1:length(mergedColors[!restGenes])))
16
17 df.newer <- df %>%
18   group_by(a) %>%
19   summarise(new_strs = c(c))
20
21 heat = datExpr[,df.newer$new_strs]
22 heat <- scale(heat)
23 column_order <- MEs
24
25 l = list()
26 k = 0
27 l[1] = 0
28
29 for(i in 1:length(selectModules)+1){
30 k = k + table(df.newer$a)[[i-1]]
31 l[i] <- k
32 }
33
34 #print(l[120])
35 #print(length(mergedColors[restGenes]))
36 #print("_____")
37
38 #for(i in 1:119){
39 #print(dim(heat[, (l[[i]]+1):l[[i+1]])))
40 #}
41 #print("_____")
42 #print(table(mergedColors[restGenes]))
43 #print("_____")
44 #print(table(df.newer$a))

```

```

45 #print("_____")
46 #print(dim(heat[, (1[[1]]+1):1[[1+1]]]))
47 #print(length(column_order[,1]))
48 #print(dim(cor(heat[, (1[[1]]+1):1[[1+1]]], column_order[,1])))
49 #print("_____")
50 #for (i in 1:length(colnames(COR3))){
51 #print(dim(datExpr[, mergedColors == substring(colnames(COR3)[[i]], first = 3)]))
52 #}
53 #print("_____")
54 #for (i in 1:119){
55 #   print(dim(heat[, (1[[i]]+1):1[[i+1]]]))
56 #}
57
58 COR3 <- array(numeric(), c(length(mergedColors[!restGenes]), length(selectModules)))
59 COR4 <- array(numeric(), c(length(selectModules), length(selectModules)))
60
61 for(i in 1:length(selectModules)){
62   for (j in 1:length(selectModules)){
63     COR3[(1[[i]]+1):1[[i+1]], j] <- cor(heat[, (1[[i]]+1):1[[i+1]]], column_order[, j], method = "spearman")
64     COR4[i, j] <- mean(COR3[(1[[i]]+1):1[[i+1]], j])
65   }
66 }
67 dim(COR3)
68

```

```

1 %%R                                     #_____ Correlation VIIc.
2
3 incq <- which(names(datExpr[1,!restGenes][df.newer$new_strs]) %in% names(datExpr[1, !restGenes]))
4 #print(length(incq))
5
6 hgt <- scale(datExpr[,!restGenes])
7 hgt <- datExpr[,!restGenes]
8 hgt_col <- mergedColors[!restGenes]
9
10 MESFemale = orderMEs(moduleEigengenes(hgt[,df.newer$new_strs], hgt_col[df.newer$new_strs])$eigengenes)
11 datKME1 = signedKME(hgt[,df.newer$new_strs], MESFemale, outputColumnName="MM.")
12
13 hmap <- ComplexHeatmap::Heatmap(datKME1, name = "CORRELATION", row_order = rownames(datKME1),
14   column_order = colnames(datKME1),
15   column_names_gp = grid::gpar(fontsize = 8),
16   column_title = "modules")
17
18 genelabels <- rowAnnotation(
19   Genes = anno_mark(
20     at = seq(1, nrow(datKME1), 1000),
21     labels = rownames(datKME1)[seq(1, nrow(datKME1), 1000)],
22     labels_gp = gpar(fontsize = 10, fontface = 'bold'),
23     padding = 0.75),
24     width = unit(2.0, 'cm') +
25     max_text_width(
26       rownames(datKME1)[seq(1, nrow(datKME1), 1000)],
27       gp = gpar(fontsize = 7, fontface = 'bold'))))
28
29 draw(hmap + genelabels)

```

```

1 '''
2 %%R                                     #_____ Correlation VIIIc.
3
4 install.packages("reshape2")
5 library(ggplot2)
6 library(reshape2)
7
8 if (!requireNamespace('ComplexHeatmap', quietly = TRUE))
9   BiocManager::install('ComplexHeatmap')
10 require(ComplexHeatmap)
11
12 if (!requireNamespace('circlize', quietly = TRUE))
13   BiocManager::install('circlize')
14 require(circlize)
15
16 sp <- c("-")
17 dspf <- data.frame(tmp = rep(sp, each = 119))                                     # reduce it by the number of modules you have excluded from analy
18 dspf
19
20 dspf$tmp[99] <- c("orange")
21 #dspf$tmp[100] <- c("purple")

```



```

1 %R                                     # _____ MDS plot of modules
2
3                                     # _____ Multidimensional scaling (MDS) is an appr
4                                     # _____ relationships specified by a dissimilarity
5                                     # _____ dissimilarity matrix is visualized by a pc
6
7                                     # _____ The Euclidean distances between a pair of
8                                     # _____ corresponding pairwise dissimilarity. An M
9                                     # _____ dissimilarity matrix as input and assigns
10                                    # _____ in d-dimensional space, where d is specifi
11                                    # _____ point locations can be displayed in the Eu
12
13                                    # _____ There are two major types of MDS: classica
14                                    # _____ function cmdscale) and non-metric MDS (R f
15                                    # _____ The following R code shows how to create a
16                                    # _____ d=2 scaling dimensions.
17
18 #restGenes_new= (mergedColors != "grey" & mergedColors != "turquoise")
19
20 adjac1 = adjacency(datExpr[, !restGenes], power = softPower, type='signed');
21 dissTOM = TOMdist(adjac1)
22 cmd1 = cmdscale(as.dist(dissTOM), 2)
23 par(mfrow = c(1, 1))
24 plot(cmd1, col = mergedColors[!restGenes], main = "MDS plot", xlab = "Dim 1",
25       ylab = "Dim2")
26

```

```

1 %R
2
3 datExpr_tmp = datExpr[, restGenes]
4 module.order <- unlist(tapply(1:ncol(datExpr_tmp),as.factor(mergedColors[restGenes]),I))
5 m<-t(t(datExpr_tmp[,module.order])/apply(datExpr_tmp[,module.order],2,max))
6 heatmap(t(m),zlim=c(0,1),col=gray.colors(100),Rowv=NA,Colv=NA,labRow=NA,scale="none",RowSideColors=mergedColors[module.order])
7

```

```

1 %R
2
3 # Making a list of module names
4 modNames <- substring(names(MEs), first = 3)
5
6 # Correlating each genes expression profile with the module eigengenes in order to create module gene sets
7 geneModuleMembership <- as.data.frame(cor(datExpr, MEs, use = "p"))
8 # "For each module, we also define a quantitative measure of module membership MM as the correlation of the
9 #module eigengene and the gene expression profile."
10
11 # Iteratively creating a list of module genesets to test. These are in ensembl ids
12 moduleGeneSets<-lapply(modNames,function(module){
13   column = match(module, modNames)
14   moduleGenes = moduleColors==module
15   rownames(geneModuleMembership[moduleGenes,])
16 })
17
18 names(moduleGeneSets)<-modNames
19
20 # Trimming the module gene sets so that the final two digits after the "." are removed
21 moduleGeneSets.trimmed<-lapply(moduleGeneSets,function(x){
22   str_split_fixed(x,"\\.",2)[,1]
23 })
24
25 # Looking up the ENTREZ id for each gene
26 moduleGeneSets.Entrez<-lapply(moduleGeneSets.trimmed,function(x){
27   bitr(x,fromType="ENSEMBL",toType="ENTREZID",OrgDb="org.Hs.eg.db")$ENTREZID
28 })
29

```

```

1 %R
2
3 moduleGeneSets_tmp <- list(moduleGeneSets$antiquewhite1,moduleGeneSets$red,moduleGeneSets$coral4,
4 moduleGeneSets$lightpink3,moduleGeneSets$brown,moduleGeneSets$skyblue2)
5
6 df = as.data.frame(org.Hs.egGO)
7 go_gene_list = unique(sort(df$gene_id))
8 go_gene_list.df <- bitr(go_gene_list, fromType = "ENTREZID",
9   toType = c("ENSEMBL", "SYMBOL"),
10   OrgDb = org.Hs.eg.db)

```

```

11
12 dfk = as.data.frame(org.Hs.egPATH)
13 kegg_gene_list = unique(sort(dfk$gene_id))
14
15 moduleGeneSets_entrez_tmp <- list(moduleGeneSets.Entrez$antiquewhite1,moduleGeneSets.Entrez$red,moduleGeneSets.Entrez$coral
16 moduleGeneSets.Entrez$lightpink3,moduleGeneSets.Entrez$brown,moduleGeneSets.Entrez$skyblue2)
17
18 ego1 = list()
19 kk = list()
20
21 for(i in 1:length(moduleGeneSets_tmp)) {
22
23   print("_____Ia. enrichGO (BP)_____")
24   l1 <- enrichGO(gene      = unique(sort(moduleGeneSets_tmp[[i]])),
25                  universe   = go_gene_list.df$ENSEMBL,                #org.Hs.egGO
26                  #universe   = geneList.df$ENSEMBL,                  #background genes
27                  OrgDb       = org.Hs.eg.db,
28                  #keyType     = "ENTREZID",
29                  keyType     = "ENSEMBL",
30                  ont          = "BP",
31                  pAdjustMethod = "BH",                                #pAdjustMethod = "fdr"
32                  pvalueCutoff = 0.05,
33                  qvalueCutoff = 0.1,
34                  readable     = TRUE)
35   if (length(l1$ID) == 0) {
36     ego1[[i]] <- "NULL"
37   }
38   else{
39     ego1[[i]] <- data.frame(
40       i,
41       l1$ID,
42       l1$Description,
43       l1$GeneRatio,
44       l1$BgRatio,
45       l1$pvalue,
46       l1$p.adjust,
47       l1$qvalue,
48       l1$geneID,
49       l1$Count
50     )
51     colnames(ego1[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
52                              "BgRatio", "pvalue", "p.adjust", "qvalue",
53                              "geneID", "Count")
54     options(scipen=999)
55   }
56
57   print("_____III. enrichKEGG_____")
58   l5 <- enrichKEGG(gene      = as.character(unique(sort(moduleGeneSets_entrez_tmp[[i]]))),
59                   organism    = 'hsa',
60                   pvalueCutoff = 0.05,
61                   universe     = kegg_gene_list,                #org.Hs.egPATH
62                   #universe     = geneList.df$ENTREZID,          #background genes
63                   qvalueCutoff = 0.1)
64   if (length(l5$ID) == 0) {
65     kk[[i]] <- "NULL"
66   }
67   else{
68     kk[[i]] <- data.frame(
69       i,
70       l5$ID,
71       l5$Description,
72       l5$GeneRatio,
73       l5$BgRatio,
74       l5$pvalue,
75       l5$p.adjust,
76       l5$qvalue,
77       l5$geneID,
78       l5$Count
79     )
80     colnames(kk[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
81                           "BgRatio", "pvalue", "p.adjust", "qvalue",
82                           "geneID", "Count")
83     options(scipen=999)
84   }
85 }
86

```

```

1 %R -o ego1,kk
2
3 for i in range(len(ego1)):
4     if (len(ego1[i])!=1):
5         df1 = pd.DataFrame(ego1[i]).T
6         df1.rename(columns={0: 'Community#', 1: 'ID',
7                             2: 'Description', 3: 'GeneRatio',
8                             4: 'BgRatio', 5: 'pvalue',
9                             6: 'p.adjust', 7: 'qvalue',
10                            8: 'geneID', 9: 'Count'},inplace=True)
11         print(df1.to_string())
12         print("#####")
13
14 print("_____")
15 print("_____")
16 print("_____")
17
18 for i in range(len(kk)):
19     if (len(kk[i])!=1):
20         df5 = pd.DataFrame(kk[i]).T
21         df5.rename(columns={0: 'Community#', 1: 'ID',
22                             2: 'Description', 3: 'GeneRatio',
23                             4: 'BgRatio', 5: 'pvalue',
24                             6: 'p.adjust', 7: 'qvalue',
25                             8: 'geneID', 9: 'Count'},inplace=True)
26         print(df5.to_string())
27         print("#####")
28

```

```

1 %%R
2
3 ck<-compareCluster(geneCluster=moduleGeneSets.Entrez,
4                   fun="enrichKEGG",organism="hsa",pvalueCutoff = 0.05,qvalueCutoff = 0.1)
5
6
7 cg.mf<-compareCluster(geneCluster=moduleGeneSets,fun="enrichGO",universe = go_gene_list.df$ENSEMBL,keyType = "ENSEMBL", pA
8                      OrgDb = "org.Hs.eg.db",pvalueCutoff = 0.05,qvalueCutoff = 0.1, ont="MF", readable = T)
9
10
11 cg.bp<-compareCluster(geneCluster=moduleGeneSets,fun="enrichGO",universe = go_gene_list.df$ENSEMBL,keyType = "ENSEMBL", pA
12                      OrgDb = "org.Hs.eg.db", pvalueCutoff = 0.05,qvalueCutoff = 0.1, ont="BP", readable = T)
13
14
15 cg.cc<-compareCluster(geneCluster=moduleGeneSets,fun="enrichGO",universe = go_gene_list.df$ENSEMBL,keyType = "ENSEMBL", pA
16                      OrgDb = "org.Hs.eg.db", pvalueCutoff = 0.05,qvalueCutoff = 0.1, ont="CC", readable = T)
17
18
19 cr<-compareCluster(geneCluster=moduleGeneSets.Entrez,fun="enrichPathway",
20                   organism="human",pvalueCutoff = 0.05, qvalueCutoff = 0.1, readable = T)
21
22
23 #file_dir = "/content/drive/My Drive/datasets/"
24 #pdf(file = file.path(file_dir,"en-tts-subset.pdf"), width = 18, height = 30, useDingbats = F)
25
26 #selected_pathways <- sample(cg.bp@compareClusterResult$Description, 10)
27 #selected_pathways <- cg.bp@compareClusterResult$Description[1:10]
28 selected_pathways <- head(cg.bp@compareClusterResult[order(cg.bp@compareClusterResult$p.adjust, decreasing = F),],10)
29 blue.bold.italic.16x.text <- element_text(face = "bold.italic", color = "blue", size = 10)
30 blue.bold.italic.16y.text <- element_text(face = "bold.italic", color = "blue", size = 10)
31
32 dotplot(ck, showCategory=selected_pathways, font.size=14)+ggtitle("KEGG enrichment") +
33   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
34   scale_x_discrete(guide = guide_axis(angle = 90))
35
36
37 dotplot(cg.mf,showCategory=selected_pathways, font.size=14)+ggtitle("GO MF enrichment") +
38   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
39   scale_x_discrete(guide = guide_axis(angle = 90))
40
41
42 dotplot(cg.bp,showCategory=selected_pathways, font.size=14)+ggtitle("GO BP enrichment") +
43   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
44   scale_x_discrete(guide = guide_axis(angle = 90))
45
46
47 dotplot(cg.cc,showCategory=selected_pathways, font.size=14)+ggtitle("GO CC enrichment") +

```

```

48 theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
49 scale_x_discrete(guide = guide_axis(angle = 90))
50
51
52 dotplot(cr, showCategory=selected_pathways, font.size=14)+ggtitle("Reactome enrichment") +
53 theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
54 scale_x_discrete(guide = guide_axis(angle = 90))
55
56 #dev.off()
57

```

```

1 %%R
2
3 # The first step to find these correlations is to make a trait table
4 # Copying over the datTraits dataframe
5
6 file_dir = "/content/drive/My Drive/datasets/"
7
8 datTraits.new <- datTraits
9
10 datTraits.new$diag <- as.factor(datTraits.new$diag)
11
12 for (l in levels(datTraits.new$diag)){
13   datTraits.new[[l]] <- datTraits.new$diag == l
14 }
15
16 # Factoring the T/F columns in to a binary 0/1 classification system
17 datTraitsFactor<-datTraits.new
18 datTraitsFactor$diag<-as.numeric(as.factor(datTraits.new$diag))
19
20 for (l in levels(datTraits.new$diag)){
21   datTraitsFactor[[l]]<-as.numeric(datTraits.new[[l]])
22 }
23
24 # Correlating the module eigengenes with subset cluster identity
25 moduleTraitCor <- cor(MEs, datTraitsFactor[, -c(6:9)])
26
27 # Getting a p-value for each correlation
28 moduleTraitPvalue <- corPvalueStudent(moduleTraitCor, nSamples)
29
30 # Plotting the relationship between learned gene co-expression modules and parameterizations
31 pdf(file = file.path(file_dir, "module-trait_relationships.pdf"), width = 20, height = 16)
32
33 par(mfrow=c(1,1))
34 textMatrix = paste(signif(moduleTraitCor, 2), "\n(",
35   signif(moduleTraitPvalue, 1), ")", sep = "")
36 dim(textMatrix) = dim(moduleTraitCor)
37 par(mar = c(6, 8.5, 3, 3))
38
39 labeledHeatmap(Matrix = t(moduleTraitCor),
40   xLabels = names(MEs), yLabels = names(datTraitsFactor[, -c(6:9)]),
41   ySymbols = names(datTraitsFactor[, -c(6:9)]),
42   colorLabels = FALSE,
43   colors = blueWhiteRed(50),
44   #textMatrix = t(textMatrix),
45   setStdMargins = T, cex.text = 0.5, zlim = c(-1,1),
46   main = paste("Module-trait relationships"))
47 dev.off()
48

```

```
1 %%R
2
3 par(mfrow=c(1,1))
4 textMatrix = paste(signif(moduleTraitCor, 2), "\n(",
5 signif(moduleTraitPvalue, 1), ")", sep = "")
6 dim(textMatrix) = dim(moduleTraitCor)
7 par(mar = c(6, 8.5, 3, 3))
8
9 labeledHeatmap(Matrix = t(moduleTraitCor),
10                xLabels = names(MEs), yLabels = names(datTraitsFactor[, -c(6:9)]),
11                ySymbols = names(datTraitsFactor[, -c(6:9)]),
12                colorLabels = FALSE,
13                colors = blueWhiteRed(50),
14                #textMatrix = t(textMatrix),
15                setStdMargins = T, cex.text = 0.5, zlim = c(-1,1),
16                main = paste("Module-trait relationships"))
```

```
1 %%R
2
3 saveRDS(moduleGeneSets$green, file = file.path(file_dir,"WGCNA.ParkGeneSet.Green.rds"))
```