


```

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]
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)
138
139 ###medianValue = median(logData);
140 ###medianCtrData = logData-medianValue
141
142 n13_train_scz_min = np.log(n15_train_scz.astype(float)+1)
143
144 #n13_train_scz_min <- zFPKM(n15_train_scz.astype(float))
145
146 n13_train_scz_min = np.transpose(n13_train_scz_min)
147
148 n13_train_scz_min[np.isnan(n13_train_scz_min)] = 0 #(57820, 414)
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))
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()
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([])

```

#_____ Filter samples

#_____ 1. No transformation

#_____ 2. Median centering data before log2-trans

#_____ 3. Log2 transformation

#_____ 4. Z-scale transformation (in R, from zFPK

#_____ Shape the array as (samples, genes)

#_____ Replace NAs with zeroes

#_____ Remove zero columns

#._____

#_____ Final gene list after zero column removal

```

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 '''

```

#II. CAPSTONE DATASET(from Kelsey)

```

323
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=condi
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=conditio
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 # _____ WGCNA _____

1 # _____ WGCNA _____

1 # _____ WGCNA _____

```

1 %%R #_____ this calculates the whole network connecti
2
3 Aadj = adjacency(t(n13_pca_scz_min), type = "distance")
4 # this calculates the whole network connectivity
5 k = as.numeric(apply(Aadj, 2, sum)) - 1
6 # standardized connectivity
7 Z.k = scale(k)

```

```

1 %%R
2
3 sampleTree = hclust(dist(n13_pca_scz_min), 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(n13_pca_scz_min)
22 datExpr = n13_pca_scz_min[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 table(clust)
4

```

```

1 %%R                                     # _____ this calculates the whole network connecti
2
3 Badj = adjacency(t(datExpr), type = "distance")
4 k = as.numeric(apply(Badj, 2, sum)) - 1
5 # standardized connectivity
6 Z.k = scale(k)
7

```

```

1 %%R                                     # _____ A. Run the codes separately to create plots
2                                     # Add each one of these plots beneath th
3 # Setting the colors to match the regional colors
4 region_labels <- RColorBrewer::brewer.pal(6, "Set1")[1:3]
5
6 datTraits <- n13_pca_path[which(keepSamples != 0),]
7
8 # Making sure the regional colors match what has been used throughtout and making a column with the hex codes.
9
10 datTraits <- datTraits %>%
11   mutate(cell.names = row.names(datTraits)) %>%
12   mutate(x = ifelse(datTraits$sex == "1", region_labels[1],
13     ifelse(datTraits$sex == "0", region_labels[2], NA)))
14
15 # Re-performing heirarchial clustering
16 sampleTree2 <- hclust(dist(datExpr), method = "average")
17
18 # Writing out that dendrogram to a PDF
19 #pdf(file = file.path(file_dir, "sample.dendrogram_region.heatmap.pdf"), height = 8, width = 12)
20 plotDendroAndColors(sampleTree2, datTraits$x,
21   groupLabels = names(datTraits),
22   main = "Sample dendrogram and sex")
23 #dev.off()
24

```

```

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 # _____ 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                                     # _____ plot PCA after removing outliers!!!!!!!!!!
2
3
4 library(ggplot2)
5
6 crr1 <- crr1[which(apply(datExpr, 2, var) != 0),]
7 datExpr <- datExpr[,which(apply(datExpr, 2, var) != 0)]# Remove zero variance columns from the new dataset
8 save(datExpr, file = '/content/drive/My Drive/datasets/datExpr.RData')
9 save(crr1, file = '/content/drive/My Drive/datasets/crr1.RData')
10
11 genes=paste("gene",seq(1:dim(datExpr)[2]), sep="")
12 colnames(datExpr)=genes
13 row.names(datExpr)=c(paste0("ctl_",seq(1:246)), paste0("scz",seq(1:138))) #___Python command: DER01_BrSeq['C'].value_count
14 condition1=rep(c("ctl"), each=246)
15 condition2=rep(c("scz"), each=138)
16 condition1 <- append(condition1, condition2)
17 length(condition1)
18
19 #pca_data=prcomp(datExpr, center = TRUE, scale = TRUE)
20 #pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
21 #df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(datExpr), condition1=condition1)
22 #ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
23 #   geom_point(size=8)+
24 #   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))

```

```

25
26 #_____ PCA for Z-scaled data
27 pca_data=prcomp(datExpr, center = TRUE, scale = TRUE)
28 pca_data_perc=round(100*pca_data$sdev^2/sum(pca_data$sdev^2),1)
29 df_pca_data=data.frame(PC1 = pca_data$x[,1], PC2 = pca_data$x[,2], sample = row.names(datExpr), condition1=condition1)
30
31 #_____Plot samples
32 ggplot(df_pca_data, aes(PC1,PC2, color = condition1))+
33   geom_point(size=2)+
34   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))
35
36 #_____Plot genes
37 #ggplot(df_pca_data, aes(PC1,PC2, color = sample))+geom_point(size=2)+
38 #   labs(x=paste0("PC1 (",pca_data_perc[1],")"), y=paste0("PC2 (",pca_data_perc[2],")"))+
39 #   theme(legend.position = "none")
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(datExpr)
47 #pca_data$group <- c(rep('ctl',250),rep('scz',139))
48 #pca <- PCA(pca_data[,1:(ncol(pca_data)-1)], graph = F)
49 #fviz_pca_ind(pca,
50 #   geom.ind = "point",
51 #   col.ind = pca_data$group,
52 #   palette = c("#00AFBB", "#E7B800"),
53 #   addEllipses = TRUE,
54 #   legend.title = "Groups")
55

```

```

1 %%R
2 #_____ Matrix factorization
3
4 #load(file = '/content/drive/My Drive/datasets/datExpr.RData')
5 #load(file = '/content/drive/My Drive/datasets/datTraits.RData')
6 #load(file = '/content/drive/My Drive/datasets/crr1.RData')
7
8 datExpr = t(datExpr)
9 colnames(datExpr) <- c(row.names(datTraits ))
10 rownames(datExpr) <- t(crr1)
11 datTraits$diag <- as.factor(datTraits$diag)
12 datTraits$ethn <- as.factor(datTraits$ethn)
13 datTraits$sex <- as.factor(datTraits$sex)
14 datTraits$age <- as.factor(datTraits$category)
15 datTraits$region <- as.factor(datTraits$region)
16 datExpr = t(datExpr)
17

```

```

1 %%R
2 #_____ SVA analysis
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 #
28 #mod0 = model.matrix(~as.factor(Gender)+as.factor(Population), data=pheno)
29 #mod = model.matrix(~as.factor(inversion_genotype)+as.factor(Gender)+as.factor(Population), data=pheno)
30 #n.sv = num.sv(edata,mod,method="leek")
31 #svobj = sva(edata,mod,mod0,n.sv=n.sv)
32

```

```

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(1:246)), paste0("scz",seq(1:138))) #__Python command: DER01_BrSeq['C'].value_c
14 condition1=rep(c("ctl"), each=246)
15 condition2=rep(c("scz"), each=138)
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 ##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=datTraits$diag, ellipse=T,var.axes=F) + scale_color_manual(values=c(
10 hvPCA2<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=datTraits$ethn, ellipse=T,var.axes=F) + scale_color_manual(values=c(
11 hvPCA3<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=datTraits$sex, ellipse=T,var.axes=F) + scale_color_manual(values=c(
12 hvPCA4<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=datTraits$region, ellipse=T,var.axes=F) + scale_color_manual(values=
13 hvPCA5<-ggbiplot(pca_data,choices=c(1,2),scale=0,groups=datTraits$age, ellipse=T,var.axes=F) + scale_color_manual(values=c(
14

```

```

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 # Show the age plot
4 hvPCA5

```

```

1 %%R
2
3 datExpr <- datExpr_cln
4 datExpr = t(datExpr)
5 colnames(datExpr) <- c(rownames(datTraits ))
6 rownames(datExpr) <- t(crr1)
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 datExpr = t(datExpr)
13

```

```

1 %%R                                     # _____ this calculates the whole network connecti
2
3 Cadj = adjacency(t(datExpr), type = "distance")
4 k = as.numeric(apply(Cadj, 2, sum)) - 1
5 # standardized connectivity
6 Z.k = scale(k)
7

```

```

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
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 plotRsquared = TRUE,
17 corFUN = 'pearson',
18 corUSE = 'pairwise.complete.obs',
19 corMultipleTestCorrection = 'BH',
20 signifSymbols = c('****', '***', '**', '*', ''),
21 signifCutpoints = 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
2
3 prin_comp <- prcomp(datExpr, scale. = F)
4 expl_var <- prin_comp$sdev^2/sum(prin_comp$sdev^2)
5 barplot(expl_var[1:50], ylab="Variance",
6         names.arg=paste0("PC",seq(1:50)), col="darkred")
7
8 N_perm <- 10
9 expl_var_perm <- matrix(NA, ncol = length(prin_comp$sdev), nrow = N_perm)
10 for(k in 1:N_perm)
11 {
12   expr_perm <- apply(datExpr,2,sample)
13   PC_perm <- prcomp(expr_perm, center=TRUE, scale=FALSE)
14   expl_var_perm[k,] <- PC_perm$sdev^2/sum(PC_perm$sdev^2)
15 }
16 plot(expl_var[1:200]~seq(1:200), ylab="Variance",
17      col="green", type='o', xlab="PC")
18 lines(colMeans(expl_var_perm)[1:200]~seq(1:200),col="red")
19 legend("topright", c("original data", "data permutation"),
20      fill=c("green","red"), inset=0.02)
21
22 pval <- apply(t(expl_var_perm) >= expl_var,1,sum) / N_perm
23 plot(pval[1:200]~seq(1:200),col="darkred",type='o',
24      xlab="PC",ylab="p-value")
25 optPC<-head(which(pval>=0.05),1)-1
26 mtext(paste0("OPTIMAL PC = ",optPC))

```

```

1 %%R
2
3 install.packages("tsne")
4 ##install.packages("Rtsne")
5 library(tsne)
6 ##library(Rtsne)
7 library(ggplot2)
8
9 nComponents <- 48

```

```

10 seed <- 1234
11 set.seed(seed)
12
13 dat.filter.BCV.tsne<-tsne(pca_data$x[,1:nComponents],perplexity=210,max_iter=2000,whiten = FALSE)
14
15 tsne_plot <- data.frame(x = dat.filter.BCV.tsne[,1], y = dat.filter.BCV.tsne[,2], col = datTraits$diag)
16 ggplot(tsne_plot) + geom_point(aes(x=x, y=y, color=col, size=1))

```

```

1 ##R
2 '''
3 if (!requireNamespace('BiocManager', quietly = TRUE))
4   install.packages('BiocManager')
5
6 BiocManager::install('PCAtools')
7 library(PCAtools)
8
9 #colnames(datExpr) <- c(rownames(datTraits ))
10 datExpr = t(datExpr)
11 p <- pca(datExpr, metadata = datTraits, center = TRUE,                #_ 1.
12         scale = TRUE)
13
14 elbow <- findElbowPoint(p$variance)
15
16 screepplot(p,components = getComponents(p, 1:20),vline = c(elbow)) +
17   geom_label(aes(x = elbow + 1, y = 50,
18     label = 'Elbow method', vjust = -1, size = 8))
19
20 datExpr = t(datExpr)
21 '''

```

```

1 ##R
2 '''
3 elbow <- findElbowPoint(p$variance)
4 screepplot(p,components = getComponents(p, 1:20),vline = c(elbow)) +
5   geom_label(aes(x = elbow + 1, y = 50,
6     label = 'Elbow method', vjust = -1, size = 8))
7 '''

```

```

1 ##R
2 '''
3 install.packages("tsne")
4 ##install.packages("Rtsne")
5 library(tsne)
6 ##library(Rtsne)
7 library(ggplot2)
8
9 nComponents <- elbow
10 seed <- 1234
11 set.seed(seed)
12
13 dat.filter.BCV.tsne<-tsne(pca_data$x[,1:nComponents],perplexity=210,max_iter=2000,whiten = FALSE)
14
15 tsne_plot <- data.frame(x = dat.filter.BCV.tsne[,1], y = dat.filter.BCV.tsne[,2], col = datTraits$diag)
16 ggplot(tsne_plot) + geom_point(aes(x=x, y=y, color=col, size=1))
17 '''

```

```

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

```

```

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(net, MEs, moduleLabels, moduleColors, mergedColors, geneTree, file = "/content/drive/My Drive/datasets/scz_networkCons
41

```

```

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 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,
23   setStdMargins = FALSE, cex.text = 0.5, zlim = c(-1,
24     1), main = paste("Module-trait relationships"))
25
26                                     #The resulting color-coded table is shown in
27                                     #the several significant module-trait associ
28                                     #(first column) as the trait of interest.
29
30                                     #Caption: Table of module-trait correlations
31                                     #correlation (and p-value) resulting from cc
32                                     #traits (columns). The table is color-coded
33                                     #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, 25)]
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, 25)]
32 print(data_mod2)

```

```

1 %%R
2
3 #print(table(mergedColors))
4
5 #print(which(as.data.frame(table(mergedColors)) == "grey", arr.ind = TRUE))
6
7 print(which(as.data.frame(table(mergedColors)) == "mistyrose", arr.ind = TRUE))
8 print(which(as.data.frame(table(mergedColors)) == "honeydew", arr.ind = TRUE))
9 print(which(as.data.frame(table(mergedColors)) == "yellow", arr.ind = TRUE))
10 print(which(as.data.frame(table(mergedColors)) == "white", arr.ind = TRUE))
11 print(which(as.data.frame(table(mergedColors)) == "blue2", arr.ind = TRUE))
12 print(which(as.data.frame(table(mergedColors)) == "black", arr.ind = TRUE))
13 print(which(as.data.frame(table(mergedColors)) == "yellow3", arr.ind = TRUE))
14 print(which(as.data.frame(table(mergedColors)) == "maroon", arr.ind = TRUE))
15 print(which(as.data.frame(table(mergedColors)) == "darkturquoise", arr.ind = TRUE))
16 print(which(as.data.frame(table(mergedColors)) == "darkred", arr.ind = TRUE))
17 print(which(as.data.frame(table(mergedColors)) == "sienna2", arr.ind = TRUE))
18 print(which(as.data.frame(table(mergedColors)) == "darkgreen", arr.ind = TRUE))
19 print(which(as.data.frame(table(mergedColors)) == "turquoise", arr.ind = TRUE))
20 print(which(as.data.frame(table(mergedColors)) == "lightpink2", arr.ind = TRUE))
21 print(which(as.data.frame(table(mergedColors)) == "blue4", arr.ind = TRUE))
22 print(which(as.data.frame(table(mergedColors)) == "lightyellow", arr.ind = TRUE))
23 print(which(as.data.frame(table(mergedColors)) == "darkgrey", arr.ind = TRUE))
24 print(which(as.data.frame(table(mergedColors)) == "sienna3", arr.ind = TRUE))
25 print(which(as.data.frame(table(mergedColors)) == "thistle3", arr.ind = TRUE))
26 print(which(as.data.frame(table(mergedColors)) == "lightgreen", arr.ind = TRUE))
27 print(which(as.data.frame(table(mergedColors)) == "pink", arr.ind = TRUE))
28 print(which(as.data.frame(table(mergedColors)) == "indianred4", arr.ind = TRUE))
29 print(which(as.data.frame(table(mergedColors)) == "green", arr.ind = TRUE))
30 print(which(as.data.frame(table(mergedColors)) == "grey60", arr.ind = TRUE))
31 print(which(as.data.frame(table(mergedColors)) == "honeydew1", arr.ind = TRUE))
32

```

```

1 %%R
2 # _____(uncorrected code from Ch.12_____ calculate the module membership values (aka. modu
3
4
5 #_____ Caption: Gene significance (GS.weight) ver
6 #_____ for the body weight related modules. GS.we
7 #_____ correlated reflecting the high correlatio
8 #_____ respective module eigengenes. We find that
9 #_____ contain genes that have high positive and
10 #_____ with body weight. In contrast, the grey "t
11 #_____ correlations with weight.
12
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("mistyrose", "honeydew", "yellow", "white", "blue2", "black", "yellow3",
23                   "maroon", "darkturquoise", "darkred", "sienna2", "darkgreen", "turquoise",
24                   "lightpink2", "blue4", "lightyellow", "darkgrey", "sienna3", "thistle3",
25                   "lightgreen", "pink", "indianred4", "green", "grey60", "honeydew1")
26 for (module in selectModules) {
27   column = match(module, colorOfColumn)
28   restModule = mergedColors == module
29   verboseScatterplot(datKME[restModule, column], GS.diagnosis[restModule], xlab = paste("Module Membership ",
30     module, "module"), ylab = "diagnosis", main = paste("kME.", module,
31     "vs. diag"), col = module)
32 }
33 '''

```

```

1 ##R
2 #_____ (corrected code from Ch.12)_____ calculate the module membership values (aka. mod
3
4 #_____ Caption: Gene significance (GS.weight) ver
5 #_____ for the body weight related modules. GS.we
6 #_____ correlated reflecting the high correlatio
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 datKME = signedKME(datExpr, MEsFemale)
13
14 colorOfColumn = substring(names(datKME), first = 4)
15 #selectModules = c("lightcyan1", "red", "paleturquoise", "navajowhite2", "lightgreen", "brown", "cyan")
16 selectModules = c("mistyrose", "honeydew", "yellow", "white", "blue2", "black", "yellow3",
17                   "maroon", "darkturquoise", "darkred", "sienna2", "darkgreen", "turquoise",
18                   "lightpink2", "blue4", "lightyellow", "darkgrey", "sienna3", "thistle3",
19                   "lightgreen", "pink", "indianred4", "green", "grey60", "honeydew1")
20
21 for (module in selectModules) {
22   restModule = mergedColors == module
23   column = match(module, colorOfColumn)
24   verboseScatterplot(datKME[restModule, column], GS1[restModule], xlab = paste("Module Membership ",
25     module, "module"), ylab = "diagnosis", main = paste("kME.", module,
26     "vs. diag"), col = module)
27 }
28
29 #_____
30 #selectModules = c("lightpink3")
31 #for (module in selectModules) {
32 #restModule = mergedColors == module
33 #print(Alldegrees1$kWithin[restModule])
34 #meta=na.omit(Alldegrees1$kWithin[restModule])
35 #}
36 #length(meta)

```

```

1 ##R
2 #_____ Our previous analysis has shown that the
3 #_____ "interesting" module in that its module s
4 #_____ Here we show how to find genes with high
5 #_____ high intramodular connectivity in the br
6 #print(names(datKME)) #_____ check names list
7
8
9 FilterGenes1= abs(GS1)> .2 & abs(datKME$kMEmistyrose)>.8
10 table(FilterGenes1)
11 dimnames(data.frame(datExpr))[[2]][FilterGenes1]
12
13 FilterGenes2= abs(GS1)> .2 & abs(datKME$kMEhoneydew)>.8
14 table(FilterGenes2)
15 dimnames(data.frame(datExpr))[[2]][FilterGenes2]
16
17 FilterGenes3= abs(GS1)> .2 & abs(datKME$kMEyellow)>.8
18 table(FilterGenes3)
19 dimnames(data.frame(datExpr))[[2]][FilterGenes3]
20
21 FilterGenes4= abs(GS1)> .2 & abs(datKME$kMEwhite)>.8
22 table(FilterGenes4)
23 dimnames(data.frame(datExpr))[[2]][FilterGenes4]
24
25 FilterGenes5= abs(GS1)> .2 & abs(datKME$kMEblue2)>.8

```

```

26 table(FilterGenes5)
27 dimnames(data.frame(datExpr))[[2]][FilterGenes5]
28
29 FilterGenes6= abs(GS1)> .2 & abs(datKME$kMEblack)>.8
30 table(FilterGenes6)
31 dimnames(data.frame(datExpr))[[2]][FilterGenes6]
32
33 FilterGenes7= abs(GS1)> .2 & abs(datKME$kMEyellow3)>.8
34 table(FilterGenes7)
35 dimnames(data.frame(datExpr))[[2]][FilterGenes7]
36
37 FilterGenes8= abs(GS1)> .2 & abs(datKME$kMEmaroon)>.8
38 table(FilterGenes8)
39 dimnames(data.frame(datExpr))[[2]][FilterGenes8]
40
41 FilterGenes9= abs(GS1)> .2 & abs(datKME$kMEdarkturquoise)>.8
42 table(FilterGenes9)
43 dimnames(data.frame(datExpr))[[2]][FilterGenes9]
44
45 FilterGenes10= abs(GS1)> .2 & abs(datKME$kMEdarkred)>.8
46 table(FilterGenes10)
47 dimnames(data.frame(datExpr))[[2]][FilterGenes10]
48
49 FilterGenes11= abs(GS1)> .2 & abs(datKME$kMEsienna2)>.8
50 table(FilterGenes11)
51 dimnames(data.frame(datExpr))[[2]][FilterGenes11]
52
53 FilterGenes12= abs(GS1)> .2 & abs(datKME$kMEdarkgreen)>.8
54 table(FilterGenes12)
55 dimnames(data.frame(datExpr))[[2]][FilterGenes12]
56
57 FilterGenes13= abs(GS1)> .2 & abs(datKME$kMETurquoise)>.8
58 table(FilterGenes13)
59 dimnames(data.frame(datExpr))[[2]][FilterGenes13]
60
61 FilterGenes14= abs(GS1)> .2 & abs(datKME$kMElightpink2)>.8
62 table(FilterGenes14)
63 dimnames(data.frame(datExpr))[[2]][FilterGenes14]
64
65 FilterGenes15= abs(GS1)> .2 & abs(datKME$kMEblue4)>.8
66 table(FilterGenes15)
67 dimnames(data.frame(datExpr))[[2]][FilterGenes15]
68
69 FilterGenes16= abs(GS1)> .2 & abs(datKME$kMElightyellow)>.8
70 table(FilterGenes16)
71 dimnames(data.frame(datExpr))[[2]][FilterGenes16]
72
73 FilterGenes17= abs(GS1)> .2 & abs(datKME$kMEdarkgrey)>.8
74 table(FilterGenes17)
75 dimnames(data.frame(datExpr))[[2]][FilterGenes17]
76
77 FilterGenes18= abs(GS1)> .2 & abs(datKME$kMEsienna3)>.8
78 table(FilterGenes18)
79 dimnames(data.frame(datExpr))[[2]][FilterGenes18]
80
81 FilterGenes19= abs(GS1)> .2 & abs(datKME$kMEthistle3)>.8
82 table(FilterGenes19)
83 dimnames(data.frame(datExpr))[[2]][FilterGenes19]
84
85 FilterGenes20= abs(GS1)> .2 & abs(datKME$kMElightgreen)>.8
86 table(FilterGenes20)
87 dimnames(data.frame(datExpr))[[2]][FilterGenes20]
88
89 FilterGenes21= abs(GS1)> .2 & abs(datKME$kMEpink)>.8
90 table(FilterGenes21)
91 dimnames(data.frame(datExpr))[[2]][FilterGenes21]
92
93 FilterGenes22= abs(GS1)> .2 & abs(datKME$kMEindianred4)>.8
94 table(FilterGenes22)
95 dimnames(data.frame(datExpr))[[2]][FilterGenes22]
96
97 FilterGenes23= abs(GS1)> .2 & abs(datKME$kMEgreen)>.8
98 table(FilterGenes23)
99 dimnames(data.frame(datExpr))[[2]][FilterGenes23]
100
101 FilterGenes24= abs(GS1)> .2 & abs(datKME$kMEgrey60)>.8
102 table(FilterGenes24)

```

```

103 dimnames(data.frame(datExpr))[[2]][FilterGenes24]
104
105 FilterGenes25= abs(GS1)> .2 & abs(datKME$kMEhoneydew1)>.8
106 table(FilterGenes25)
107 dimnames(data.frame(datExpr))[[2]][FilterGenes25]
108

```

```

1 %%R
2
3 whichmodule = "mistyrore" # The heatmap color-codes the scaled gene e
4 Eigengene = MESFemale$MEmistyrore # red corresponds to over-expression
5 datExprModule = datExpr[, mergedColors == whichmodule]
6 # set the margins of the graphics window
7 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
8 # create a heatmap whose columns correspond to the arrays and whose rows
9 # correspond to genes
10 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
11 main = paste("heatmap", whichmodule, "module"))
12
13 # scatter plot between eigengene and sample network connectivity # Scatter plot of the lightgreen module eig
14 par(mfrow = c(1, 1)) #sample network connectivity Z.k, which is
15 verboseScatterplot(Eigengene, Z.k, xlab = paste("ME", whichmodule, sep = "")) #module eigengene takes on an extreme negat
16 abline(h = -2, col = "red", lwd = 2) #(corresponding to the sample where most mc
17 #The red horizontal line in the scatter plc
18
19 whichmodule = "honeydew"
20 Eigengene = MESFemale$MEhoneydew
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 = "yellow"
34 Eigengene = MESFemale$MEyellow
35 datExprModule = datExpr[, mergedColors == whichmodule]
36 # set the margins of the graphics window
37 par(mfrow = c(1, 1), mar = c(0.3, 5.5, 3, 2))
38 # create a heatmap whose columns correspond to the arrays and whose rows
39 # correspond to genes
40 plotMat(t(scale(datExprModule)), cex.axis = 2, nrgcols = 30, rlabels = F, rcols = whichmodule,
41 main = paste("heatmap", whichmodule, "module"))
42 # scatter plot between eigengene and sample network connectivity
43 par(mfrow = c(1, 1))
44 verboseScatterplot(Eigengene, Z.k, xlab = paste("ME", whichmodule, sep = ""))
45 abline(h = -2, col = "red", lwd = 2)
46
47 whichmodule = "white"
48 Eigengene = MESFemale$MEwhite
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 = "blue2"
62 Eigengene = MESFemale$MEblue2
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="mistyrose";
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="honeydew";
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="yellow";
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="white";
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="blue2";
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="mistyrose" # 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="honeydew"
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="yellow"
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="white"
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="blue2"
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 # _____ Correlating tSNE position with eigenvector
4 install.packages("tsne")
5 install.packages("Rtsne")
6 install.packages("ggplot2")
7 library(tsne)
8 library(ggplot2)
9 library("Rtsne")
10
11 seed <- 1234
12 set.seed(seed)
13
14 # _____ 1. tsne directly on the original data with
15 whichmodule = "blue2"
16 Eigengene = MEsFemale$MEblue2
17 datExprModule = datExpr[, mergedColors == whichmodule]
18
19 dat.filter.DCV.tsne<-tsne(datExprModule,perplexity=30,max_iter=2000,whiten = FALSE)
20
21 eigentSNE<-cbind(dat.filter.DCV.tsne[,1],dat.filter.DCV.tsne[,2],Eigengene)
22 colnames(eigentSNE)<-c("tSNE1_pos","tSNE2_pos",whichmodule)
23
24 p2.subset<-ggplot(data.frame(eigentSNE)) + geom_point(aes(x=tSNE1_pos,y=tSNE2_pos,color = blue2),size=4) + scale_color_grad
25 p2.subset.all <- p2.subset + ggtitle("WGCNA module eigengene expression") + coord_fixed(ratio = 1) + xlab("t-SNE 1") + ylat
26 p2.subset.all
27
28 # _____ 2. tsne on data after PCA
29
30 #whichmodule = "blue2"
31 #datExprModule = datExpr[, mergedColors == whichmodule]
32
33 #p <- pca(t(datExprModule), metadata = datTraits, center = TRUE,
34 # scale = TRUE)
35 #elbow <- findElbowPoint(p$variance)
36 #screeplot(p,components = getComponents(p, 1:20),vline = c(elbow)) +
37 # geom_label(aes(x = elbow + 1, y = 50,
38 # label = 'Elbow method', vjust = -1, size = 8))
39
40 #dat.filter.DCV.tsne<-Rtsne(datExprModule,perplexity=30,max_iter=2000,whiten = FALSE,
41 # PCA = True, initial_dims = elbow + 1)
42 #plot(dat.filter.DCV.tsne$Y,col="blue",xlab="tSNE1",ylab="tSNE2",cex=0.5)
43 '''
44

```

```

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[78])) # _____ how to add genes

```

```

11 print(names(MEs[44]))
12 print(names(MEs[127]))
13 print(names(MEs[126]))
14 print(names(MEs[7]))
15 print(names(MEs[5]))
16 print(names(MEs[129]))
17 print(names(MEs[70]))
18 print(names(MEs[33]))
19 print(names(MEs[28]))
20 print(names(MEs[107]))
21 print(names(MEs[20]))
22 print(names(MEs[124]))
23 print(names(MEs[59]))
24 print(names(MEs[8]))
25 print(names(MEs[66]))
26 print(names(MEs[21]))
27 print(names(MEs[108]))
28 print(names(MEs[122]))
29 print(names(MEs[58]))
30 print(names(MEs[91]))
31 print(names(MEs[48]))
32 print(names(MEs[39]))
33 print(names(MEs[43]))
34 print(names(MEs[45]))
35
36 MEs <- MEs[,c(78,44,127,126,7,5,129,70,33,28,107,20,124,59,8,66,21,108,122,58,91,48,39,43,45)]
37

```

```

1 ##R
2
3 install.packages("gplots")
4 library(gplots)
5 myheatcol = colorpanel(250,'red','orange','lemonchiffon')
6
7 #_____ Remove grey and turquoise for ahead of attempting to look
8 modNames <- substr(names(MEs), 3)
9 restGenes = (mergedColors != "grey" & mergedColors != "turquoise")
10 restGenes = (mergedColors != "mistyrose" & mergedColors != "honeydew" & mergedColors != "yellow" & mergedColors != "white")
11
12 table(!restGenes)
13
14 softPower = 12
15 diss1=1-TOMsimilarityFromExpr(datExpr[,!restGenes], power = softPower, TOMType = "signed")
16
17 colnames(diss1) = rownames(diss1) = modNames[!restGenes]
18 hier1=flashClust(as.dist(diss1), method="average" )
19 plotDendroAndColors(hier1, mergedColors[!restGenes], "Dynamic Tree Cut", dendroLabels = FALSE,
20                     hang = 0.03, addGuide = TRUE, guideHang = 0.05, main = "Gene dendrogram and module colors")
21
22 #_____ The commented code below uses a lot of RAM
23
24
25
26
27 #diag(diss1) = NA;
28 #TOMplot(diss1^7, hier1, as.character(mergedColors[!restGenes]),
29 #main = "TOM heatmap plot, module genes") # or add: "module genes" , col=myheatcol"
30
31
32 #diss1 = 1-adjacency( datExpr[, !restGenes], power = softPower, type='signed' )
33 #hier1 = hclust(as.dist(diss1), method="average" )
34 #diag(diss1) = NA;
35 #TOMplot(diss1^7, hier1, as.character(mergedColors[!restGenes]),
36 #main = "Adjacency heatmap plot, module genes" # or add: "module genes" , col=myheatcol"
37

```

```

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[, (l[[1]]+1):l[[1+1]])))
47 #print(length(column_order[,1]))
48 #print(dim(cor(heat[, (l[[1]]+1):l[[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[, (l[[i]]+1):l[[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[(l[[i]]+1):l[[i+1]],j] <- cor(heat[(l[[i]]+1):l[[i+1]], column_order[,j], method = "spearman")
64   COR4[i,j] <- mean(COR3[(l[[i]]+1):l[[i+1]],j])
65 }
66 }
67 dim(COR3)
68

```

```

1 %%R
2
3 #_____ Correlating tSNE position with eigenvector
4
5 install.packages("tsne")
6 install.packages("ggplot2")
7 library(tsne)
8 library(ggplot2)
9
10 seed <- 1234
11 set.seed(seed)
12
13 Eigengene = MEsFemale
14 dat.filter.DCV.tsne<-tsne(heat,perplexity=30,max_iter=2000,whiten = FALSE)
15
16 eigentSNE<-cbind(dat.filter.DCV.tsne[,1],dat.filter.DCV.tsne[,2],MEs)
17 colnames(eigentSNE)<-c("tSNE1_pos","tSNE2_pos",modNames)

```

```

18
19 eigentSNE.melt<-melt(eigentSNE,id.vars=c("tSNE1_pos","tSNE2_pos"))
20
21 # Melting the data and selecting only the modules we want to plot
22 eigentSNE.melt.subset <- eigentSNE.melt %>%
23   filter(variable == "mistyrose" | variable == "honeydew" | variable == "yellow" | variable == "white" | variable ==
24     "blue2" | variable == "black" | variable == "yellow3" | variable == "maroon" | variable ==
25     "darkturquoise" | variable == "darkred" | variable == "sienna2" | variable ==
26     "darkgreen" | variable == "turquoise")
27
28 p2.subset<-ggplot(eigentSNE.melt.subset) + geom_point(aes(x=tSNE1_pos,y=tSNE2_pos,color=value),size=1) + scale_color_gradie
29
30 p2.subset.all <- p2.subset + facet_wrap("variable", nrow = 2) + ggtitle("WGCNA module eigengene") + coord_fixed(ratio = 1)
31
32 p2.subset.all

```

```

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), 300),
21     labels = rownames(datKME1)[seq(1, nrow(datKME1), 300)],
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), 300)],
27     gp = gpar(fontsize = 7, fontface = 'bold'))))
28
29 draw(hmap + genelabels)

```

```

1 '''
2 %%R                                     #_____ Correlation VIIc.
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")
22 dspf$tmp[101] <- c("darkturquoise")
23 #dspf$tmp[102] <- c("darkolivegreen")
24 dspf$tmp[103] <- c("coral4")
25 #dspf$tmp[104] <- c("darkmagenta")
26 dspf$tmp[105] <- c("pink")
27 #dspf$tmp[106] <- c("brown")
28 dspf$tmp[107] <- c("turquoise")
29 #dspf$tmp[108] <- c("antiquewhite1")
30 dspf$tmp[109] <- c("red")

```



```

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_cmp<-lapply(moduleGeneSets.trimmed,function(x){
27   bitr(x, fromType="SYMBOL", toType="ENTREZID", OrgDb="org.Hs.eg.db")$ENTREZID
28 })
29

```

```

1 ##R
2
3 moduleGeneSets_tmp = list()
4 moduleGeneSets.Entrez = list()
5 moduleGeneSets_entrez_tmp = list()
6 moduleGeneSets_ensembl_tmp = list()
7
8 #_____ I. Use all the genes of the most important modules
9 #moduleGeneSets_tmp <- list(moduleGeneSets$antiquewhite1,moduleGeneSets$red,moduleGeneSets$coral4,
10 #moduleGeneSets$lightpink3,moduleGeneSets$brown,moduleGeneSets$skyblue2)
11
12 #_____ II. Use only the most important genes of the most important modules
13 moduleGeneSets_tmp <- list(dimnames(data.frame(datExpr))[[2]][FilterGenes1],dimnames(data.frame(datExpr))[[2]][FilterGenes
14 dimnames(data.frame(datExpr))[[2]][FilterGenes4],dimnames(data.frame(datExpr))[[2]][FilterGenes5],dimnames(data.frame(datE
15 dimnames(data.frame(datExpr))[[2]][FilterGenes7],dimnames(data.frame(datExpr))[[2]][FilterGenes8],dimnames(data.frame(datE
16 dimnames(data.frame(datExpr))[[2]][FilterGenes10],dimnames(data.frame(datExpr))[[2]][FilterGenes11],dimnames(data.frame(d
17 dimnames(data.frame(datExpr))[[2]][FilterGenes13],dimnames(data.frame(datExpr))[[2]][FilterGenes14],dimnames(data.frame(d
18 dimnames(data.frame(datExpr))[[2]][FilterGenes16],dimnames(data.frame(datExpr))[[2]][FilterGenes17],dimnames(data.frame(d
19 dimnames(data.frame(datExpr))[[2]][FilterGenes19],dimnames(data.frame(datExpr))[[2]][FilterGenes20],dimnames(data.frame(d
20 dimnames(data.frame(datExpr))[[2]][FilterGenes22],dimnames(data.frame(datExpr))[[2]][FilterGenes23],dimnames(data.frame(d
21 dimnames(data.frame(datExpr))[[2]][FilterGenes25])
22
23 df = as.data.frame(org.Hs.egGO)
24 go_gene_list = unique(sort(df$gene_id))
25 go_gene_list.df <- bitr(go_gene_list, fromType = "ENTREZID",
26   toType = c("ENSEMBL", "SYMBOL"),
27   OrgDb = org.Hs.eg.db)
28
29 dfk = as.data.frame(org.Hs.egPATH)
30 kegg_gene_list = unique(sort(dfk$gene_id))
31
32
33 #_____ I. Use all the genes of the most important modules
34 #moduleGeneSets_entrez_tmp <- list(moduleGeneSets.Entrez$antiquewhite1,moduleGeneSets.Entrez$red,moduleGeneSets.Entrez$cor
35 #moduleGeneSets.Entrez$lightpink3,moduleGeneSets.Entrez$brown,moduleGeneSets.Entrez$skyblue2)
36
37 #_____ II. Use only the most important genes of the most important modules
38 for(i in 1:length(moduleGeneSets_tmp)) {
39   if (length(moduleGeneSets_tmp[[i]])>2){
40     print(i)

```

```

41 moduleGeneSets.Entrez[[i]] <- bitr(moduleGeneSets_tmp[[i]], fromType = "SYMBOL",
42   toType = c("ENSEMBL", "ENTREZID"),
43   OrgDb = org.Hs.eg.db)
44 }
45 }
46
47 for(i in 1:length(moduleGeneSets.Entrez)) {
48 moduleGeneSets_entrez_tmp[[i]] <- list(moduleGeneSets.Entrez[[i]]$ENTREZID)
49 }
50
51 for(i in 1:length(moduleGeneSets.Entrez)) {
52 moduleGeneSets_ensembl_tmp[[i]] <- list(moduleGeneSets.Entrez[[i]]$ENSEMBL)
53 }
54
55 ego1 = list()
56 kk = list()
57
58 for(i in 1:length(moduleGeneSets_tmp)) {
59
60 print("_____Ia. enrichGO (BP)_____")
61 l1 <- enrichGO(gene      = unique(sort(unlist(moduleGeneSets_ensembl_tmp[[i]]))),
62   universe      = go_gene_list.df$ENSEMBL,          #org.Hs.egGO
63   #universe      = geneList.df$ENSEMBL,              #background genes
64   OrgDb          = org.Hs.eg.db,
65   #keyType        = "ENTREZID",
66   keyType        = "ENSEMBL",
67   ont             = "BP",
68   pAdjustMethod  = "BH",                             #pAdjustMethod = "fdr"
69   pvalueCutoff   = 0.05,
70   qvalueCutoff   = 0.1,
71   readable       = TRUE)
72 if (length(l1$ID) == 0) {
73 ego1[[i]] <- "NULL"
74 }
75 else{
76 ego1[[i]] <- data.frame(
77   i,
78   l1$ID,
79   l1$Description,
80   l1$GeneRatio,
81   l1$BgRatio,
82   l1$pvalue,
83   l1$p.adjust,
84   l1$qvalue,
85   l1$geneID,
86   l1$Count
87 )
88 colnames(ego1[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
89   "BgRatio", "pvalue", "p.adjust", "qvalue",
90   "geneID", "Count")
91 options(scipen=999)
92 }
93
94 print("_____III. enrichKEGG_____")
95 l5 <- enrichKEGG(gene      = as.character(unique(sort(unlist(moduleGeneSets_entrez_tmp[[i]])))),
96   organism      = 'hsa',
97   pvalueCutoff  = 0.05,
98   universe      = kegg_gene_list,          #org.Hs.egPATH
99   #universe      = geneList.df$ENTREZID,    #background genes
100   qvalueCutoff  = 0.1)
101 if (length(l5$ID) == 0) {
102 kk[[i]] <- "NULL"
103 }
104 else{
105 kk[[i]] <- data.frame(
106   i,
107   l5$ID,
108   l5$Description,
109   l5$GeneRatio,
110   l5$BgRatio,
111   l5$pvalue,
112   l5$p.adjust,
113   l5$qvalue,
114   l5$geneID,
115   l5$Count
116 )
117 colnames(kk[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",

```

```

118         "BgRatio", "pvalue", "p.adjust", "qvalue",
119         "geneID", "Count")
120 options(scipen=999)
121 }
122 }
123

```

```

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_cmp,
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$SYMBOL,keyType = "SYMBOL", pAdj
8     OrgDb = "org.Hs.eg.db",pvalueCutoff = 0.05,qvalueCutoff = 0.1, ont="MF", readable = F)
9
10
11 cg.bp<-compareCluster(geneCluster=moduleGeneSets,fun="enrichGO",universe = go_gene_list.df$SYMBOL,keyType = "SYMBOL", pAdj
12     OrgDb = "org.Hs.eg.db", pvalueCutoff = 0.05,qvalueCutoff = 0.1, ont="BP", readable = F)
13
14
15 cg.cc<-compareCluster(geneCluster=moduleGeneSets,fun="enrichGO",universe = go_gene_list.df$SYMBOL,keyType = "SYMBOL", pAdj
16     OrgDb = "org.Hs.eg.db", pvalueCutoff = 0.05,qvalueCutoff = 0.1, ont="CC", readable = F)
17
18
19 cr<-compareCluster(geneCluster=moduleGeneSets.Entrez_cmp,fun="enrichPathway",
20     organism="human",pvalueCutoff = 0.05, qvalueCutoff = 0.1, readable = F)
21
22
23 #file_dir = "/content/drive/My Drive/datasets/"
24 #pdf(file = file.path(file_dir,"en-tts-subset.pdf"), width = 18, height = 30, useDingbats = F)
25
26 #selected_pathways <- sample(cg.bp@compareClusterResult$Description, 10)
27 #selected_pathways <- cg.bp@compareClusterResult$Description[1:10]
28
29 selected_pathways_ck <- head(ck@compareClusterResult[order(ck@compareClusterResult$p.adjust, decreasing = F),],10)
30 selected_pathways_cg.mf <- head(cg.mf@compareClusterResult[order(cg.mf@compareClusterResult$p.adjust, decreasing = F),],10)
31 selected_pathways_cg.bp <- head(cg.bp@compareClusterResult[order(cg.bp@compareClusterResult$p.adjust, decreasing = F),],10)
32 selected_pathways_cg.cc <- head(cg.cc@compareClusterResult[order(cg.cc@compareClusterResult$p.adjust, decreasing = F),],10)
33 selected_pathways_cr <- head(cr@compareClusterResult[order(cr@compareClusterResult$p.adjust, decreasing = F),],10)
34
35 blue.bold.italic.16x.text <- element_text(face = "bold.italic", color = "blue", size = 10)
36 blue.bold.italic.16y.text <- element_text(face = "bold.italic", color = "blue", size = 10)
37

```

```

1 %%R
2
3
4 dotplot(ck, showCategory=selected_pathways_ck$Description, font.size=14)+ggtitle("KEGG enrichment") +
5   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
6   scale_x_discrete(guide = guide_axis(angle = 90))
7

```

```

1 %%R
2
3 dotplot(cg.mf, showCategory=selected_pathways_cg.mf$Description, font.size=14)+ggtitle("GO MF enrichment") +
4   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
5   scale_x_discrete(guide = guide_axis(angle = 90))

```

```

1 %%R
2
3 dotplot(cg.bp, showCategory=selected_pathways_cg.bp$Description, font.size=14)+ggtitle("GO BP enrichment") +
4   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
5   scale_x_discrete(guide = guide_axis(angle = 90))

```

```

1 %%R
2
3 dotplot(cg.cc, showCategory=selected_pathways_cg.cc$Description, font.size=14)+ggtitle("GO CC enrichment") +
4   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
5   scale_x_discrete(guide = guide_axis(angle = 90))

```

```

1 %%R
2
3 dotplot(cr, showCategory=selected_pathways_cr$Description, font.size=14)+ggtitle("Reactome enrichment") +
4   theme_bw(base_size = 24) + theme(axis.text.x = blue.bold.italic.16x.text, axis.text.y = blue.bold.italic.16y.text) +
5   scale_x_discrete(guide = guide_axis(angle = 90))

```

```

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$loadings[1], decreasing=T)[1:5]
8 p1l<-order(p$loadings[1], decreasing=F)[1:5]
9
10 p2h<-order(p$loadings[2], decreasing=T)[1:5]
11 p2l<-order(p$loadings[2], decreasing=F)[1:5]
12
13 p3h<-order(p$loadings[3], decreasing=T)[1:5]
14 p3l<-order(p$loadings[3], decreasing=F)[1:5]
15
16 p4h<-order(p$loadings[4], decreasing=T)[1:5]
17 p4l<-order(p$loadings[4], decreasing=F)[1:5]
18
19 p5h<-order(p$loadings[5], decreasing=T)[1:5]
20 p5l<-order(p$loadings[5], decreasing=F)[1:5]
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,])

```

```

1 %%R                                     # _____ Reactome pathway gene set enrichment analy
2
3 ll = list()
4 PCAGeneSets_tmp = list()
5                                     # _____ 1.
6 PCAGeneSets_tmp <- list(ph_all[[1]], ph_all[[2]], ph_all[[3]], ph_all[[4]], ph_all[[5]],
7                           pl_all[[1]], pl_all[[2]], pl_all[[3]], pl_all[[4]], pl_all[[5]])
8

```

```

 9 #pe1 = list()
10 #pe2 = list()
11 #pe3 = list()
12 #pe4 = list()
13 #pe5 = list()
14 #pe1[[1]]<-ph_all[[1]]
15 #pe1[[2]]<-p1_all[[1]]
16 #pe2[[1]]<-ph_all[[2]]
17 #pe2[[2]]<-p1_all[[2]]
18 #pe3[[1]]<-ph_all[[3]]
19 #pe3[[2]]<-p1_all[[3]]
20 #pe4[[1]]<-ph_all[[4]]
21 #pe4[[2]]<-p1_all[[4]]
22 #pe5[[1]]<-ph_all[[5]]
23 #pe5[[2]]<-p1_all[[5]]
24 #PCAGeneSets_tmp <- list(unlist(pe1),unlist(pe2),unlist(pe3),unlist(pe4),unlist(pe5))
25
26 PCAGeneSets_entrez_tmp = list()
27 PCAGeneSets_entrez_tmp<-lapply(PCAGeneSets_tmp,function(x){
28   bitr(x,fromType="SYMBOL",toType="ENTREZID",OrgDb="org.Hs.eg.db")$ENTREZID})
29
30 for (i in 1:length(PCAGeneSets_entrez_tmp)){
31   print("_____IV. enrichPathway_____")
32   l6 <- enrichPathway(gene = as.character(unique(sort(PCAGeneSets_entrez_tmp[[i]]))),
33     pvalueCutoff = 0.05,
34     qvalueCutoff = 0.1,
35     readable = TRUE)
36   if (length(l6$ID) == 0) {
37     ll[[i]] <- "NULL"
38   }
39   else{
40     ll[[i]] <- data.frame(
41       i,
42       l6$ID,
43       l6$Description,
44       l6$GeneRatio,
45       l6$BgRatio,
46       l6$pvalue,
47       l6$p.adjust,
48       l6$qvalue,
49       l6$geneID,
50       l6$Count
51     )
52     colnames(ll[[i]]) <- c("Community#", "ID", "Description", "GeneRatio",
53       "BgRatio", "pvalue", "p.adjust", "qvalue",
54       "geneID", "Count")
55     options(scipen=999)
56     print(ll[[i]])
57     print("#####")
58   }
59 }
60

```

```

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$green, file = file.path(file_dir, "WGCNA.ParkGeneSet.Green.rds"))
4 saveRDS(moduleGeneSets$brown, file = file.path(file_dir, "WGCNA.ParkGeneSet.Brown.rds"))
5
6

```