fMRI_Analysis

Qihang Wu

Import data & processing

```r
# Change file extension
# old_names <- list.files(path)
# new_names <- gsub(".1D", "1D.csv", old_names)
# file.rename(paste0(path, old_names), paste0(path, new_names))

# Import data
pheno <- read_csv("./data/phenotypic_CMU.csv") %>% janitor::clean_names()

df_file_names <-
  inner_join(data.frame(name = all_files,
                        sub_id = as.numeric(substring(all_files, 7, 13))),
             pheno, by = "sub_id")

# Files for each group
asd_files <- df_file_names %>% filter(dx_group == 1) %>%
  select(1) %>% mutate(name = paste(path, name, sep = ""))
tc_files <- df_file_names %>% filter(dx_group == 2) %>%
  select(1) %>% mutate(name = paste(path, name, sep = ""))

# ROI
roi <- read_csv("./data/dos160_labels.csv")
colnames(roi) <- c("number", "label")

# from paper: prediction of individual brain maturity using fmri
roi_network <- data.frame(
  network = c(rep("default", 34), rep("fronto_parietal", 21), rep("cingulo_opercular", 32),
              rep("sensorimotor", 33), rep("occipital", 22), rep("cerebellum", 18)),
  number = c(1, 4, 5, 6, 7, 11, 13, 14, 15, 17, 20, 63, 72, 73, 84, 85,
             90, 91, 92, 93, 94, 105, 108, 111, 112, 115, 117, 124, 132,
             134, 136, 137, 141, 146, # 1
             2, 3, 9, 10, 12, 16, 21, 22, 23, 24, 29, 34, 36, 88, 96, 99,
             101, 104, 107, 114, 116, # 2
             8, 18, 19, 25, 26, 27, 28, 30, 31, 33, 38, 39, 40, 44, 47, 57,
             58, 59, 61, 71, 76, 78, 80, 81, 87, 89, 95, 97, 100, 102, 103, 125, # 3
             32, 35, 37, 41, 42, 43, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 60,
             62, 64, 65, 66, 67, 68, 69, 70, 74, 75, 77, 79, 82, 83, 86, # 4
             106, 118, 119, 123, 126, 129, 133, 135, 139, 142, 145, 147, 148, 149,
             152, 153, 154, 156, 157, 158, 159, 160, # 5
             98, 109, 110, 113, 120, 121, 122, 127, 128, 130, 131, 138, 140,
             143, 144, 150, 151, 155 # 6
             ))
```

```r
roi_comb <- inner_join(roi, roi_network, by = "number")
```

EDA

```r
# Demographics
tbl_summary(by = dx_group,
            data = pheno %>%
              mutate(sex = factor(sex, levels = c(1, 2), labels = c("Male", "Female")),
                     dx_group = factor(dx_group, levels = c(1, 2), labels = c("ASD", "Typical Controls")
              select(3, 5:7, 9:11),
            label = list(age_at_scan ~ "Age, yrs", sex ~ "Sex", handedness_category ~ "Handedness",
                         fiq ~ "Full-Scale Intelligence Quotient (FIQ)",
                         viq ~ "Verbal Intelligence Quotient (VIQ)",
                         piq ~ "Performance Intelligence Quotient (PIQ)")) %>%
  add_p(list(all_continuous() ~ "t.test", all_categorical() ~ "chisq.test")) %>%
  modify_table_styling(footnote = "IQ score obtained using the Wechsler Abbreviated Scale of Intelligen
  modify_caption("**Demographics for People with ASD and TC**")
```

Table 1: **Demographics for People with ASD and TC**

Characteristic	**ASD**, N = 14	**Typical Controls**, N = 13	**p-value**
Age, yrs	25.5 (21.2, 30.8)	27.0 (21.0, 30.0)	0.8
Sex			>0.9
Male	11 (79%)	10 (77%)	
Female	3 (21%)	3 (23%)	
Handedness			0.6
Ambi	1 (7.1%)	1 (7.7%)	
L	1 (7.1%)	0 (0%)	
R	12 (86%)	12 (92%)	
Full-Scale Intelligence Quotient (FIQ)	117 (105, 123)	110 (109, 124)	>0.9
Verbal Intelligence Quotient (VIQ)	111 (99, 120)	111 (108, 122)	0.7
Performance Intelligence Quotient (PIQ)	115 (108, 121)	109 (108, 123)	0.7

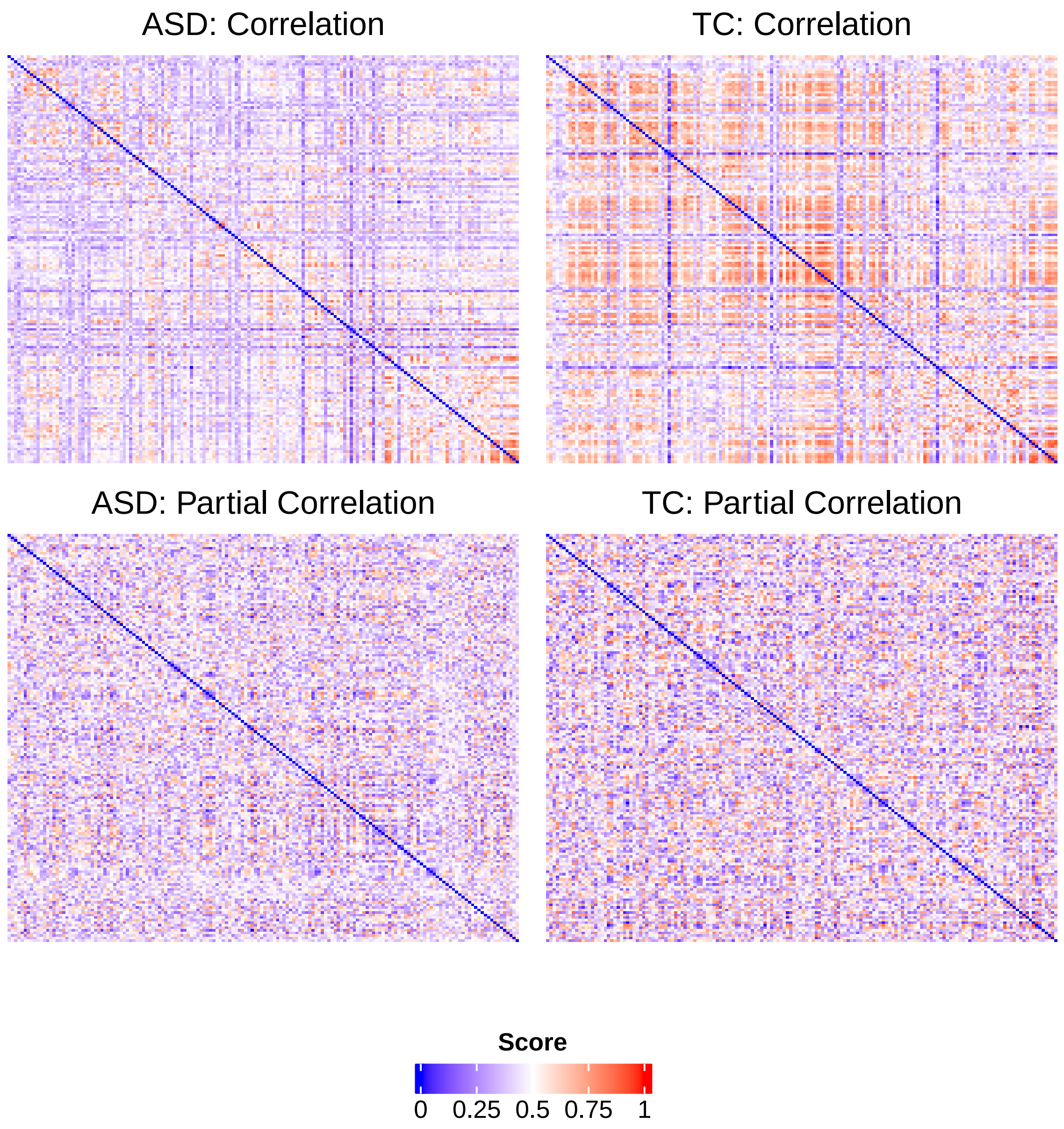
Parameter tuning

```r
# Alpha tuning
# Randomly select one sample from all files
set.seed(2022)
file_sel <- fread(paste(path, sample(df_file_names$name, 1), sep = ""),
                  select = c(1:160))

# --- PC ---
# pc_alpha_grid <- exp(seq(-2, -5, length = 20)) # make sparse graph
# suffStat <- list(C = cor(file_sel), n = nrow(file_sel))

# pc_alpha <- foreach(x = pc_alpha_grid) %do% {
#   pc_res <- pc(suffStat, indepTest, alpha = x,
#                 p = ncol(file_sel), verbose = FALSE)
#
#   pc_final <- ifelse(pc_res@pMax < x, 1, 0)
#   g <- graph.adjacency(pc_final, mode = "undirected")
#   len <- length(E(g))
#
#   data.frame(alpha = x, num_edge = len)
# }
# saveRDS(pc_alpha, file = "./data/pc_alpha.rds")
pc_alpha <- readRDS("./data/pc_alpha.rds")

pc_alpha_res <- do.call(rbind, pc_alpha)
fig_1 <- pc_alpha_res %>%
  ggplot(aes(x = log(alpha), y = num_edge)) +
  geom_point() + geom_line() + theme_bw() +
  labs(x = "log(alpha)", y = "Number",
       title = "Number of Undirected Edges by alpha in PC")
sel_alpha_pc <- pc_alpha_res$alpha[9]

# --- MMHC ---
# mmhc_alpha_grid <- exp(seq(-3, -6, length = 20)) # Try similar number of egdes
#
# mmhc_alpha <- foreach(x = mmhc_alpha_grid) %do% {
#   mmhc_res <- pchc::mmhc(as.matrix(file_sel), alpha = x)
#
#   mmhc_final <- ifelse(1 - exp(mmhc_res$ini$pvalue) < x, 1, 0)
#   g <- graph.adjacency(mmhc_final, mode = "undirected")
#   len <- length(E(g))
#
#   data.frame(alpha = x, num_edge = len)
# }
# saveRDS(mmhc_alpha, file = "./data/mmhc_alpha.rds")
mmhc_alpha <- readRDS("./data/mmhc_alpha.rds")

mmhc_alpha_res <- do.call(rbind, mmhc_alpha)
fig_2 <- mmhc_alpha_res %>%
  ggplot(aes(x = log(alpha), y = num_edge)) +
  geom_point() + geom_line() + theme_bw() +
  labs(x = "log(alpha)", y = "Number",
       title = "Number of Undirected Edges by alpha in MMHC")
sel_alpha_mmhc <- mmhc_alpha_res$alpha[12]
```

Individual-level analysis

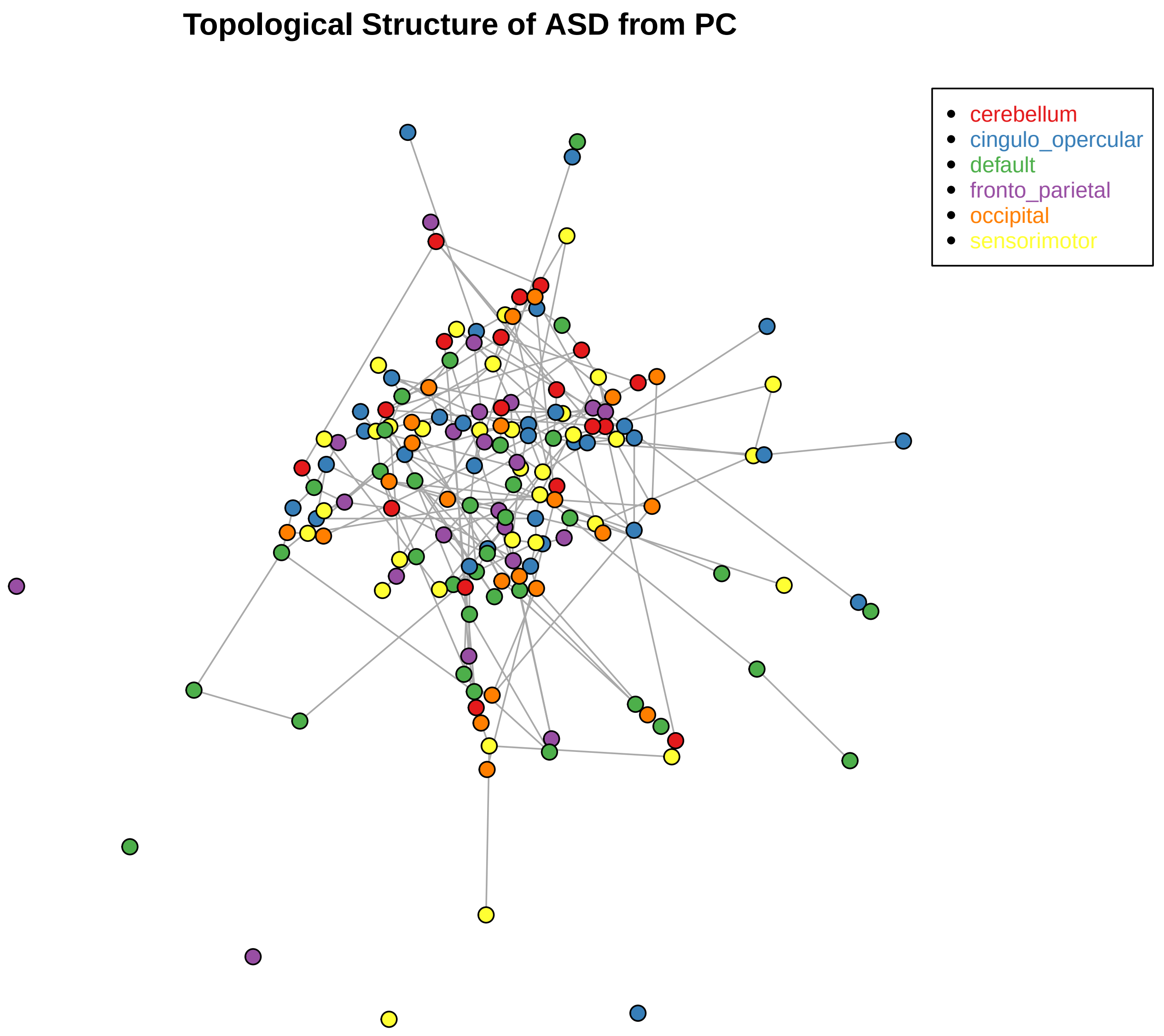
```
# Normalize for 0 to 1
normalize <- function(x){
  t_x <- (x - min(x))/(max(x) - min(x))
  return(t_x)}

indepTest <- gaussCItest # conditional independence test

# --- ASD ---
```

```r
# Randomly select one ASD sample
set.seed(123)
asd_dat <- fread(sample(as.matrix(asd_files), 1), select = c(1:160))
suffStat_asd <- list(C = cor(asd_dat), n = nrow(asd_dat))

# Correlation
asd_corr <- cor(asd_dat)
t_asd_corr <- normalize(asd_corr)
diag(t_asd_corr) <- 0

# Partial correlation
asd_pcorr <- ppcor::pcor(asd_dat)
t_asd_pcorr <- normalize(asd_pcorr$estimate)
diag(t_asd_pcorr) <- 0

# --- TC ---
# Randomly select one TC sample
set.seed(123)
tc_sel <- sample(as.matrix(tc_files), 1)
tc_dat <- fread(tc_sel, select = c(1:160))
suffStat_tc <- list(C = cor(tc_dat), n = nrow(tc_dat))

# Correlation
tc_corr <- cor(tc_dat)
t_tc_corr <- normalize(tc_corr)
diag(t_tc_corr) <- 0

# Partial Correlation
tc_pcorr <- ppcor::pcor(tc_dat)
t_tc_pcorr <- normalize(tc_pcorr$estimate)
diag(t_tc_pcorr) <- 0

# --- Make plots ---
col_fun <- colorRamp2(c(0, 0.5, 1), c("blue", "white", "red"))

grid.newpage()
pushViewport(viewport(layout = grid.layout(nr = 3, nc = 2)))

# ASD
pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 1))
draw(Heatmap(t_asd_corr, col = col_fun, cluster_rows = F, cluster_columns = F,
             show_row_names = F, show_column_names = F,
             show_heatmap_legend = F, column_title = "ASD: Correlation"), newpage = F)
upViewport()
pushViewport(viewport(layout.pos.row = 2, layout.pos.col = 1))
draw(Heatmap(t_asd_pcorr, col = col_fun, cluster_rows = F, cluster_columns = F,
             show_row_names = F, show_column_names = F,
             show_heatmap_legend = F, column_title = "ASD: Partial Correlation"), newpage = F)
upViewport()

# TC
pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 2))
draw(Heatmap(t_tc_corr, col = col_fun, cluster_rows = F, cluster_columns = F,
               show_row_names = F, show_column_names = F,
               show_heatmap_legend = F, column_title = "TC: Correlation"), newpage = F)
upViewport()
pushViewport(viewport(layout.pos.row = 2, layout.pos.col = 2))
draw(Heatmap(t_tc_pcorr, col = col_fun, cluster_rows = F, cluster_columns = F,
               show_row_names = F, show_column_names = F,
               show_heatmap_legend = F, column_title = "TC: Partial Correlation"), newpage = F)
upViewport()

lgd <- Legend(at = c(0, 0.25, 0.5, 0.75, 1), col_fun = col_fun, title = "Score",
                direction = "horizontal", title_position = "topcenter")
pushViewport(viewport(x = 0.5, y = 0.25))
grid.draw(lgd)
```

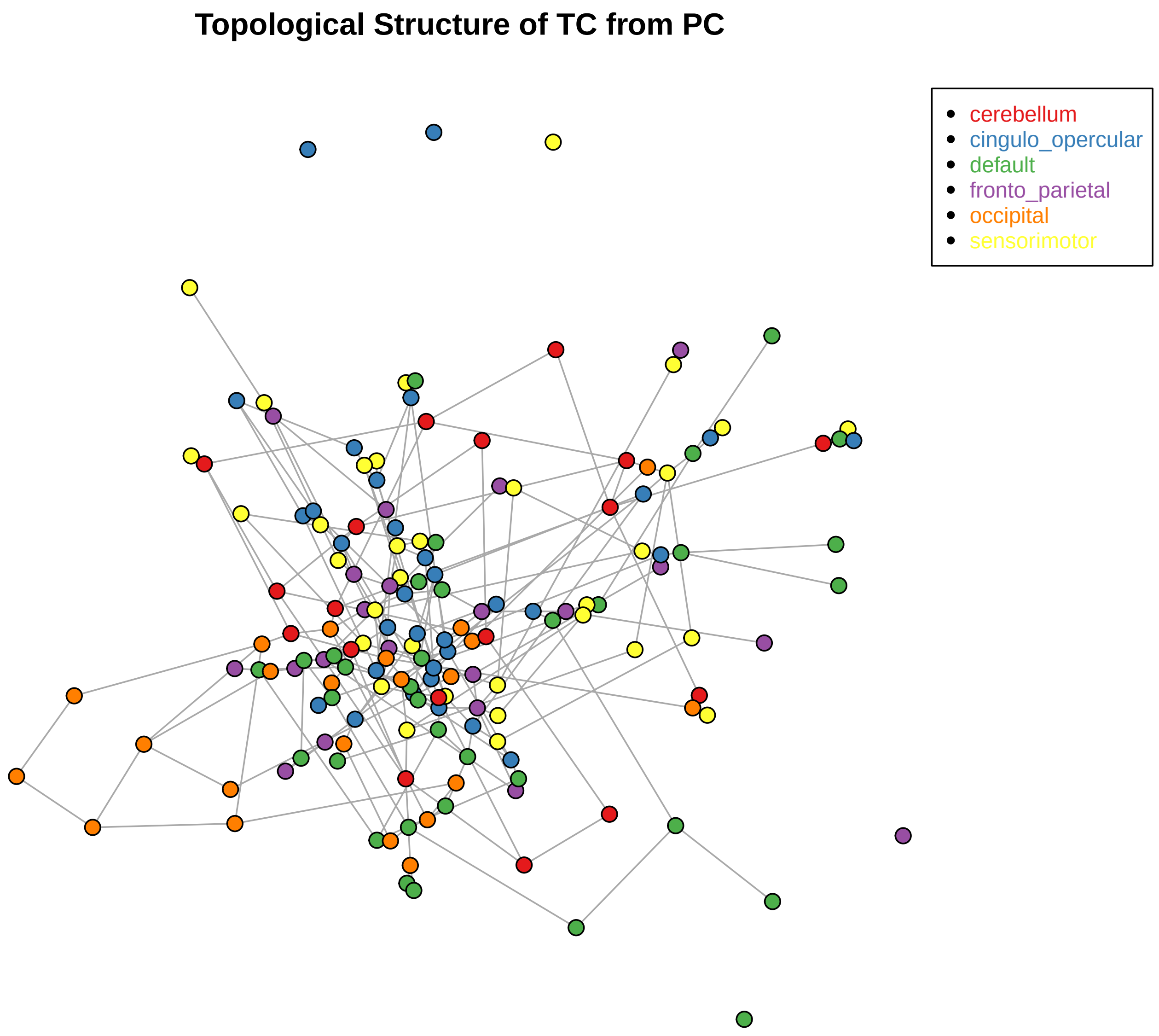


```r
# Change weights of layout
weight_community <-
  function(row, membership, weight_within = 0.5, weight_between = 100) {
    if (membership$network[which(membership$number == row[1])] ==
      membership$network[which(membership$number == row[2])])
    {
      weight = weight_within
    }
    else {
      weight = weight_between
    }
    return(weight)
  }

# Colors for 6 networks
my_color <- RColorBrewer::brewer.pal(6, "Set1")

# PC plot for ASD vs. TD
# Groups
df_grp <- roi_comb %>%
  select(1, 3)

# --- ASD ---
# PC
asd_pc <- pc(suffStat_asd, indepTest, alpha = sel_alpha_pc,
             p = ncol(asd_dat), verbose = FALSE)
asd_res <- ifelse(asd_pc@pMax < sel_alpha_pc, 1, 0)
g_asd <- graph.adjacency(asd_res, mode = "undirected") # convert to igraph
el_asd <- get.edgelist(g_asd) # get edgelist

layout_wgt <- apply(el_asd, 1, weight_community, df_grp)
g_asd$layout <- layout_with_fr(g_asd, weights = layout_wgt)

# make plot
plot(g_asd, vertex.size = 3.5, vertex.label = NA,
     vertex.color = my_color[as.factor(df_grp$network)],
     main = "Topological Structure of ASD from PC")
legend("topright", legend = levels(as.factor(df_grp$network)),
       pch = 20, text.col = my_color, cex = 0.8)
```

Topological structures from PC



```
# --- TC ---
tc_pc <- pc(suffStat_tc, indepTest, alpha = sel_alpha_pc,
            p = ncol(tc_dat), verbose = FALSE)
tc_res <- ifelse(tc_pc@pMax < sel_alpha_pc, 1, 0)
g_tc <- graph.adjacency(tc_res, mode = "undirected")
el_tc <- get.edgelist(g_tc)

layout_wgt <- apply(el_tc, 1, weight_community, df_grp)
g_tc$layout <- layout_with_fr(g_tc, weights = layout_wgt)

# make plot
plot(g_tc, vertex.size = 3.5, vertex.label = NA,
     vertex.color = my_color[as.factor(df_grp$network)],
     main = "Topological Structure of TC from PC")
legend("topright", legend = levels(as.factor(df_grp$network)),
       pch = 20, text.col = my_color, cex = 0.8)
```



Group-level predivtion

```
ctrl <- trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

# === PC ===
# Obtain graph metrics from PC
pc_res_all <- foreach(x = seq(nrow(df_file_names))) %do% {
  dat <- fread(paste(path, df_file_names$name[x], sep = ""), select = c(1:160))

  # mmhc_res <- pchc::mmhc(as.matrix(dat), alpha = 0.05)
  # mmhc_final <- ifelse(exp(mmhc_res$ini$pvalue) < 0.05, 1, 0)
  # g <- graph.adjacency(mmhc_final, mode = "undirected")

  suffStat <- list(C = cor(dat), n = nrow(dat))
```

```r
  pc_res <- pc(suffStat, indepTest, alpha = sel_alpha_pc,
               p = ncol(dat), verbose = FALSE)
  pc_final <- ifelse(pc_res@pMax < sel_alpha_pc, 1, 0)
  g <- graph.adjacency(pc_final, mode = "undirected")

  # measurements
  # distance class
  ecc <- mean(eccentricity(g)) # eccentricity
  # rad <- radius(g) # radius
  # gir <- girth(g, circle = FALSE) # girth

  # connection class
  ass <- assortativity_degree(g, directed = FALSE)
  tra <- transitivity(g, type = "global")

  # community class
  mod <- modularity(g, as.factor(df_grp$network))

  m_dist <- mean_distance(g, directed = FALSE) # mean distance
  # assor <- assortativity_degree(g, directed = FALSE) # assortivity degree
  # den <- edge_density(g, loops = FALSE) # density
  # dia <- diameter(g, directed = FALSE) # diameter

  data.frame(
    id = df_file_names$sub_id[x],
    eccentricity = ecc, assortativity = ass,
    transitivity = tra, modularity = mod, mean_distance = m_dist,
    asd = ifelse(df_file_names$dx_group[x] == 1, 1, 0))}

pc_res_all_2 <- do.call(rbind, pc_res_all) %>%
  mutate(asd = as.factor(ifelse(asd == 1, "asd", "no_asd")))

# Split train vs. Test
set.seed(1234)
tr_id <- createDataPartition(y = pc_res_all_2$asd, p = 0.75, list = FALSE)
tr_dat_pc <- pc_res_all_2[tr_id, -1]
ts_dat_pc <- pc_res_all_2[-tr_id, -1]

# glm
set.seed(1234)
glm_fit_pc <- train(x = tr_dat_pc[, 1:5], y = tr_dat_pc$asd,
                    method = "glm", metric = "ROC",
                    trControl = ctrl)
pc_test_prob <- predict(glm_fit_pc, newdata = ts_dat_pc, type = "prob")[, 1]
pc_test_pred <- ifelse(pc_test_prob > 0.5, "asd", "no_asd")

confusionMatrix(data = as.factor(pc_test_pred),
                reference = ts_dat_pc$asd,
                positive = "no_asd")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction asd no_asd
##     asd      2      0
##     no_asd   1      3
##
##                Accuracy : 0.8333
##                  95% CI : (0.3588, 0.9958)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.1094
##
##                   Kappa : 0.6667
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 1.0000
##             Specificity : 0.6667
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.8333
##
##        'Positive' Class : no_asd
##
```

```r
# === MMHC ===
# Obtain graph metrics from MMHC
mmhc_res_all <- foreach(x = seq(nrow(df_file_names))) %do% {
  dat <- fread(paste(path, df_file_names$name[x], sep = ""), select = c(1:160))

  mmhc_res <- pchc::mmhc(as.matrix(dat), alpha = sel_alpha_mmhc)
  mmhc_final <- ifelse(exp(mmhc_res$ini$pvalue) < sel_alpha_mmhc, 1, 0)
  g <- graph.adjacency(mmhc_final, mode = "undirected")

  # measurements
  # distance class
  ecc <- mean(eccentricity(g)) # eccentricity
  # rad <- radius(g) # radius
  # gir <- girth(g, circle = FALSE) # girth

  # connection class
  ass <- assortativity_degree(g, directed = FALSE)
  tra <- transitivity(g, type = "global")

  # community class
  mod <- modularity(g, as.factor(df_grp$network))

  m_dist <- mean_distance(g, directed = FALSE) # mean distance
  # assor <- assortativity_degree(g, directed = FALSE) # assortivity degree
  # den <- edge_density(g, loops = FALSE) # density
  # dia <- diameter(g, directed = FALSE) # diameter

  data.frame(
    id = df_file_names$sub_id[x],
    eccentricity = ecc,
    assortativity = ass, transitivity = tra, modularity = mod,
    asd = ifelse(df_file_names$dx_group[x] == 1, 1, 0))}

mmhc_res_all_2 <- do.call(rbind, mmhc_res_all) %>%
  mutate(asd = as.factor(ifelse(asd == 1, "asd", "no_asd")))

# Split train vs. Test
set.seed(1234)
tr_id <- createDataPartition(y = mmhc_res_all_2$asd, p = 0.75, list = FALSE)
tr_dat_mmhc <- mmhc_res_all_2[tr_id, -1]
ts_dat_mmhc <- mmhc_res_all_2[-tr_id, -1]

# glm
set.seed(1234)
glm_fit_mmhc <- train(x = tr_dat_mmhc[, 1:5], y = tr_dat_mmhc$asd,
                      method = "glm", metric = "ROC",
                      trControl = ctrl)
mmhc_test_prob <- predict(glm_fit_mmhc, newdata = ts_dat_mmhc, type = "prob")[, 1]
mmhc_test_pred <- ifelse(mmhc_test_prob > 0.5, "asd", "no_asd")

confusionMatrix(data = as.factor(mmhc_test_pred),
                reference = ts_dat_mmhc$asd,
                positive = "no_asd")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction asd no_asd
##     asd      3      0
##     no_asd   0      3
##
##                Accuracy : 1
##                  95% CI : (0.5407, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.01563
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0
##             Specificity : 1.0
##          Pos Pred Value : 1.0
##          Neg Pred Value : 1.0
##              Prevalence : 0.5
##          Detection Rate : 0.5
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 1.0
##
##        'Positive' Class : no_asd
##
```

Others

```r
# === PC ===
# ASD
run <- function(dat_files = asd_files, net = "default") {
  vars <- roi_network %>%
    filter(network == net) %>% pull(number)
  len <- length(vars)

  all_pc <- mclapply(1:nrow(dat_files), function(i){
  dat <- fread(dat_files[i, ], select = vars)

  suffStat <- list(C = cor(dat), n = nrow(dat))
  pc_res <- pc(suffStat, indepTest, alpha = 0.05,
               p = ncol(dat), verbose = FALSE)
  pc_final <- ifelse(pc_res@pMax < 0.05, 1, 0)
  adj_mat <- graph.adjacency(pc_final, mode = "max")

  # number of edges
  num <- nrow(get.edgelist(adj_mat))/((len * (len - 1))/2)
  }, mc.cores = num.cores)

  return(do.call(rbind, all_pc))
}

asd_default_pc <- run(asd_files, "cerebellum")
tc_default_pc <- run(tc_files, "cerebellum")

t.test(asd_default_pc, tc_default_pc)



asd_all_pc <- mclapply(1:nrow(asd_files), function(i){
  dat <- fread(asd_files[i, ], select = c(1:160))

  suffStat <- list(C = cor(dat), n = nrow(dat))
  pc_res <- pc(suffStat, indepTest, alpha = 0.1,
               p = ncol(dat), verbose = FALSE)
  pc_final <- ifelse(pc_res@pMax < 0.1, 1, 0)
  adj_mat <- graph.adjacency(pc_final, mode = "max")

  # number of edges
  num <- nrow(get.edgelist(adj_mat))
}, mc.cores = num.cores)

asd_res <- do.call(rbind, asd_all_pc)

# TC
tc_all_pc <- mclapply(1:nrow(tc_files), function(i){
  dat <- fread(tc_files[i, ], select = c(1:160))

  suffStat <- list(C = cor(dat), n = nrow(dat))
  pc_res <- pc(suffStat, indepTest, alpha = 0.1,
               p = ncol(dat), verbose = FALSE)
  pc_final <- ifelse(pc_res@pMax < 0.1, 1, 0)
  adj_mat <- graph.adjacency(pc_final, mode = "max")

  # number of edges
  num <- nrow(get.edgelist(adj_mat))/(160 * 159/2)
}, mc.cores = num.cores)

tc_res <- do.call(rbind, tc_all_pc)

t.test(asd_res, tc_res)

# --- Try alpha = 0.05 ---
# ASD
asd_all_res_pc <- common_conn(asd_files, alpha = 0.05, prct = 0.5)

# TC
tc_all_res_pc <- common_conn(tc_files, alpha = 0.05, prct = 0.5)

# --- Different alpha ---
alpha_ls <- seq(0.01, 0.5, length = 10)

# ASD
asd_all_res_pc_alpha <-
  mclapply(alpha_ls, function(alpha){

    # find the number of connections for each alpha
    pc_res <- common_conn_pc(dat_files = asd_files, alpha = alpha, prct = 0.5)
}, mc.cores = num.cores)
# saveRDS(asd_all_res_pc_alpha, "./data/asd_all_pc_alpha.RData")

# TC
tc_all_res_pc_alpha <-
  mclapply(seq(0.01, 1, by = 0.01), function(alpha){

    # find the number of connections for each alpha
    pc_res <- common_conn_pc(dat_files = tc_files, alpha = alpha, prct = 0.5)
}, mc.cores = num.cores)
# saveRDS(tc_all_res_pc_alpha, "./data/tc_all_pc_alpha.RData")

# Comparison
num_conn_pc <- data.frame(
  asd = lapply(asd_all_res_pc_alpha, function(x) nrow(x)) %>% unlist(),
  tc = lapply(tc_all_res_pc_alpha, function(x) nrow(x)) %>% unlist()
)
```