SDS - Homework 3

Group 00: Michele Meo, Lorenzo Ceccomancini

Connect your brain

Graph estimates

Given data from 12 different patients for each group of autism (ASD or TD), we want to estimate the brain connectivity among specific regions of interest. In particular, we have 116 ROIs and 145 observations in time for each ROI that we can interpret as a signal related to the neural activity in that specific ROI.

Assuming we know the correlation coefficient among all the couples of ROIs, we have to choose a method to pool together all the correlations related to different patients belonging to the same group (ASD or TD): we decided to pool together correlations using *mean* and *median*, then in the following we will analyze the results obtained with both. We decided to use just *Pearson correlation* as association measure in order to compare different pooling methods and visualizations on the same correlation coefficient.

Pooling with Mean and Z-transform correlation

First of all we tried to apply an empirical average to pool together the data. The idea is to evaluate the correlation matrix on each patient of both groups in order to have a quantity related to the connectivity among different ROI on the same patient set of observations. The second step is to compute the empirical average on the 12 correlation matrices we obtain, so that we end up to have a mean correlation measure for each possible couple of ROIs: at the end we will have 2 mean correlation matrix, one for each type of autism.

In the following function we compute the average among the 12 correlation matrices related to the 12 subjects.

input: *data* dataframe of our data, *cor_method* measure of association or correlation

output: the mean correlation matrix

```
average_cor_matrix = function(data, cor_method = 'pearson'){

  D = 116

  average_matrix = matrix(0, D, D)

  for(patient in data){
    cor_matrix = cor(patient, method=cor_method)
    average_matrix = average_matrix + cor_matrix
  }

  diag(average_matrix) = 0   # set auto-correlation to 0

  return(average_matrix/length(data))
}
```

Now that we have the correlation matrices for both groups, we don't work directly with those correlations but with the Fisher Z-transform of them: this is useful because through this transformation we map the correlation random variable in the normal support and, without assumption at the population level, we have that $h(\hat{\rho}) \sim N(h(\rho), \frac{1}{n-3})$, so knowing n we already have the variance of this asymptotically normal distribution.

Once we transform the correlations using Fisher, we can then build the graph computing a confidence interval around each component of the transformed correlation matrix and defining a link when this confidence interval has an empty intersection with the $[-t, t]$, where t is the 80^{th} percentile.

Given the pooled correlation matrix, we build the graph using the following function.

input: *cor_matrix* pooled correlation matrix of a specific group, *correction* a flag to include or not the Bonferroni correction for multiplicity

output: *graph* the adjacency matrix associated to the estimated graph

```
graph_estimate = function(cor_matrix, correction = TRUE){

  D = 116
  n = 145

  z_cor = atanh(cor_matrix)   # compute Fisher on correlations

  t = quantile(z_cor, probs=0.8)
  alpha = 0.05

  m = choose((D-1), 2)
  a = alpha
  if(correction) a = alpha/m   # apply or not Bonferroni correction

  # Build the lower and upper bound for each estimated correlation
  low = z_cor - (1/sqrt(n-3))*qnorm(1-a/2)
  up = z_cor + (1/sqrt(n-3))*qnorm(1-a/2)

  # Build the matrix to compare CI with [-t,t] interval easier
  t_up = matrix(t, D, D)
  t_low = matrix(-t, D, D)

  # Apply the reasoning proposed in the text to set connections
  graph = apply(((up < t_low) | (low > t_up)), FUN = as.numeric,
                MARGIN = c(1, 2))

  return(graph)
}
```

For the visualization we decided to take the edges that are unique in one kind of graph or the other, so in other words the edges not in common between the two graphs related to the 2 different autism forms: this is done in the following function.

input: *matrix1* and *matrix2* the 2 adjacency matrix

output: *matrix* the adjacency matrix containing all the edges that are unique for *matrix1* (so note that here the inputs order is important)

```
delta_matrix = function(matrix1, matrix2){

  matrix = matrix1
  mask = xor(matrix1, matrix2)   # compute edges not in common between ASD-TD
  n = length(matrix1[1, ])

  #Take the edges not in common just belonging to matrix1
  for(i in 1:n){
    for(j in 1:n){
      if(!mask[i, j]) matrix[i, j] = 0
    }
  }

  return(matrix)
}
```

The visualization is performed by the following function, where we use the *ggraph* library to plot the graphs. We plot the two graphs related to ASD and TD autism, then we plot a graph representing the edges not in common between the two previous graphs and, in the end, a graph representing the common edges: this is done in order to emphasize differences and similarities.

input: *g1* and *g2* the 2 adjacency matrix

output: no output, just a sequence of plots

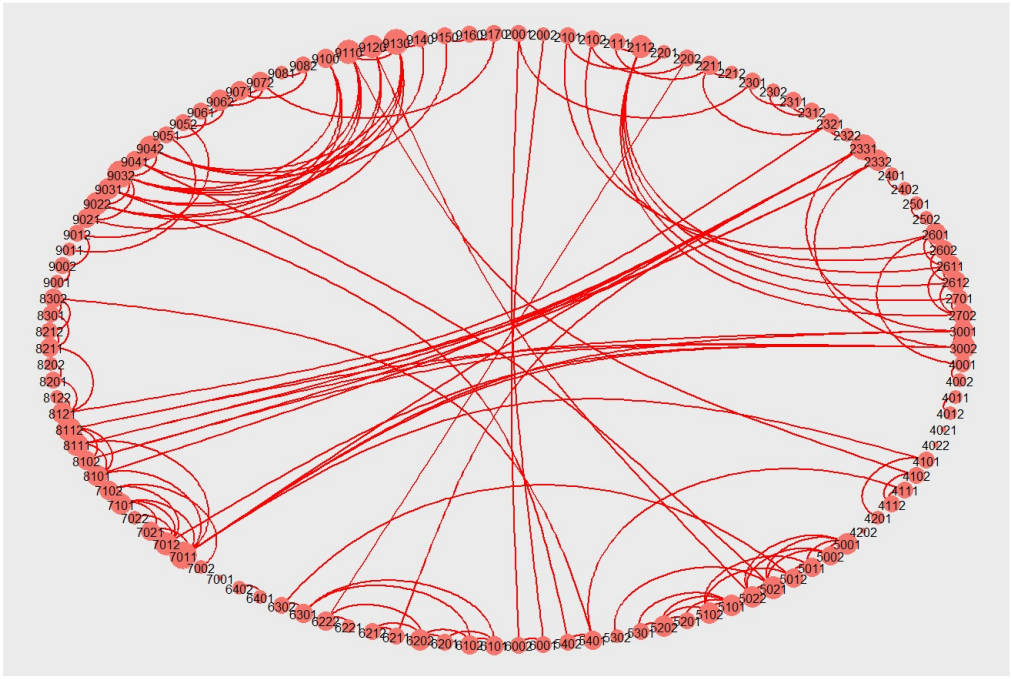
```
graph_visual = function(g1, g2){

  d1 = delta_matrix(g1, g2)
  d2 = delta_matrix(g2, g1)
  asd_diff = graph_from_adjacency_matrix(d1, diag = F)
  td_diff = graph_from_adjacency_matrix(d2, diag = F)

  edges_asd = extract_edges(asd_diff)
  edges_td = extract_edges(td_diff)

  g = union(asd_diff, td_diff)
  edges_g = extract_edges(g)

  mask = rep(FALSE, length(E(g)))

  for(i in 1:length(edges_g)){
    for(j in 1:length(edges_td)){
      if(edges_g[i]==edges_td[j]) mask[i] = TRUE
    }
  }

  E(g)$type[!mask] = 'ASD'
  E(g)$type[mask] = 'TD'

  g_asd = graph_from_adjacency_matrix(g1, diag = F)
  g_td = graph_from_adjacency_matrix(g2, diag = F)

  asd_graph = ggraph(g_asd, layout='linear', circular=T) +
    geom_edge_arc(color='red') +
    geom_node_point(show.legend=F, aes(size=degree(g_asd),
                                       color='orange')) +
    geom_node_text(show.legend=F, aes(label=name, size=0.8)) +
    ggtitle("ASD graph")

  plot(asd_graph)

  td_graph = ggraph(g_td, layout='linear', circular=T) +
    geom_edge_arc(color='skyblue') +
    geom_node_point(show.legend=F, aes(size=degree(g_td),
                                       color='orange')) +
    geom_node_text(show.legend=F, aes(label=name, size=0.8)) +
    ggtitle("TD graph")

  plot(td_graph)

  delta_graph = ggraph(g, layout='linear', circular=T) +
    geom_edge_arc(aes(color=type)) +
    geom_node_point(show.legend=F, aes(size=degree(g),
                                       color='orange')) +
    geom_node_text(show.legend=F, aes(label=name, size=0.8)) +
    ggtitle("Different edges")

  plot(delta_graph)

  common = g1 & g2
  g = graph_from_adjacency_matrix(common, diag = F)

  common_graph = ggraph(g, layout='linear', circular=T) +
    geom_edge_arc(color='black') +
    geom_node_point(show.legend=F, aes(size=degree(g),
                                       color='orange')) +
    geom_node_text(show.legend=F, aes(label=name, size=0.8)) +
    ggtitle("Common edges")

  plot(common_graph)
}
```

Now we call the above functions in order to plot our resulting graphs: the graph for the *Autism Spectrum Disord* group, the graph for the *Typically Develop* group, the graph with the edges not in common between ASD and TD, the graph with the common edges between ASD and TD.

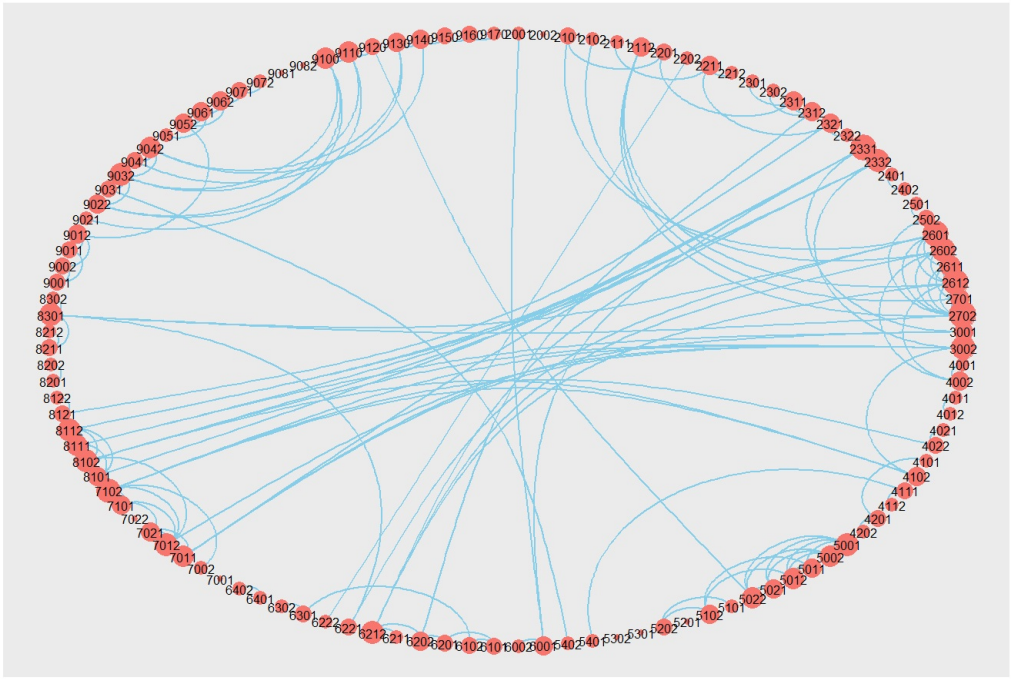
From the first 2 plots we can observe that the graph related to ASD seems to have more connections with respect to the TD graph, this can be seen also from the size of the node representation that is proportional to their degree: however, later we will show a more detailed analysis of the degrees. The most relevant observation comes from the third plot, where we can see that the TD graph has not unique edges that connect Region of Interest labeled in the interval $\sim \{9000, \ldots, 9170\}$, where with “unique edges” we mean edges that are not in common with the ASD graph. Instead, in the fourth plot we show the graph defined by common edges between the ASD and TD graphs.

```
ASD_cor = average_cor_matrix(asd_sel)
TD_cor = average_cor_matrix(td_sel)

ASD_graph_a = graph_estimate(ASD_cor)
TD_graph_a = graph_estimate(TD_cor)

graph_visual(ASD_graph_a, TD_graph_a)
```

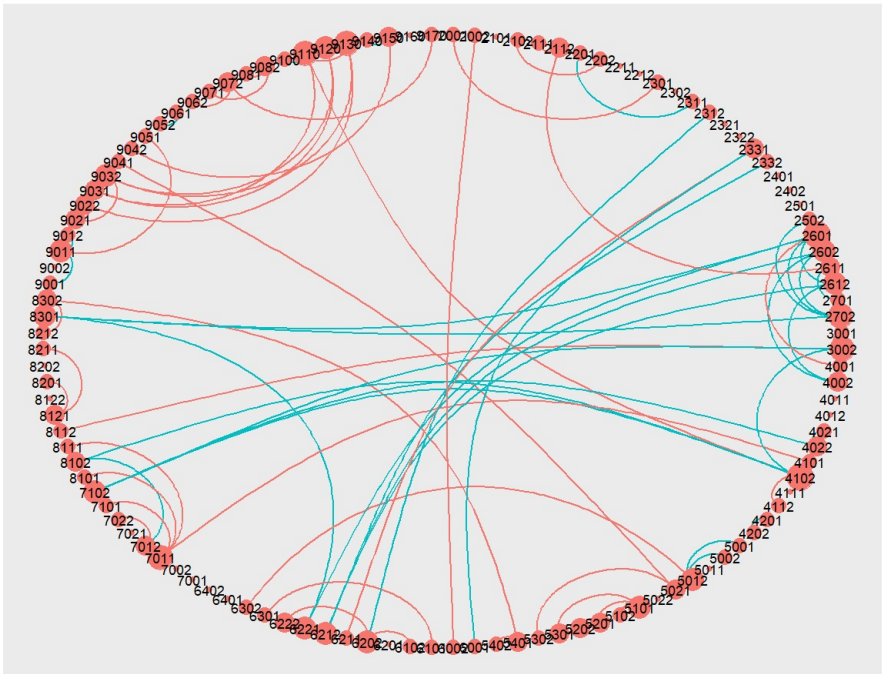
ASD graph



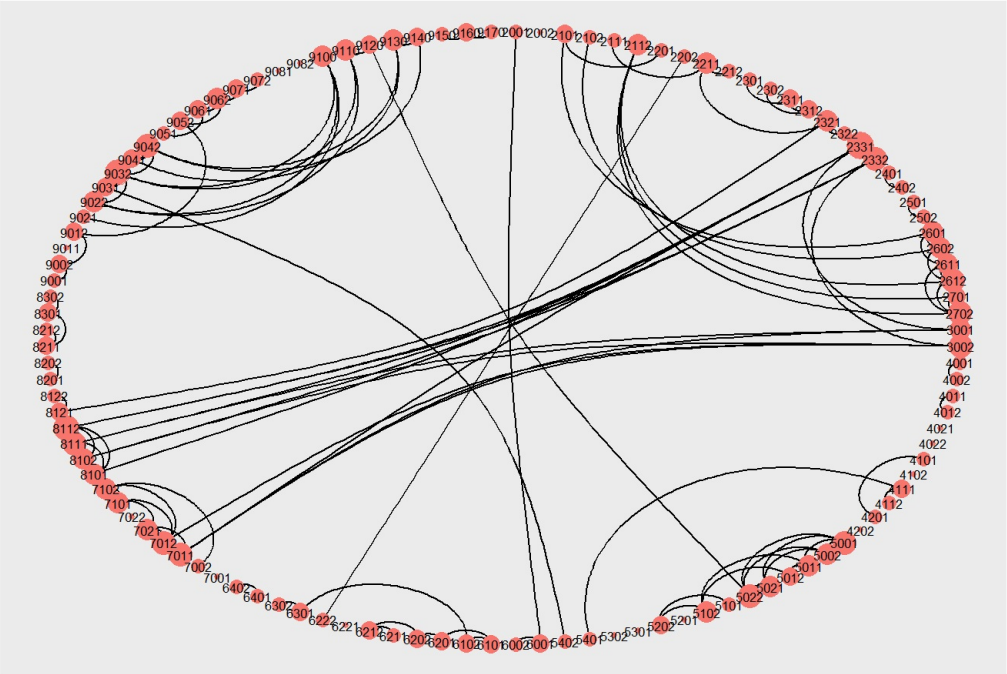
TD graph



Different edges



Common edges



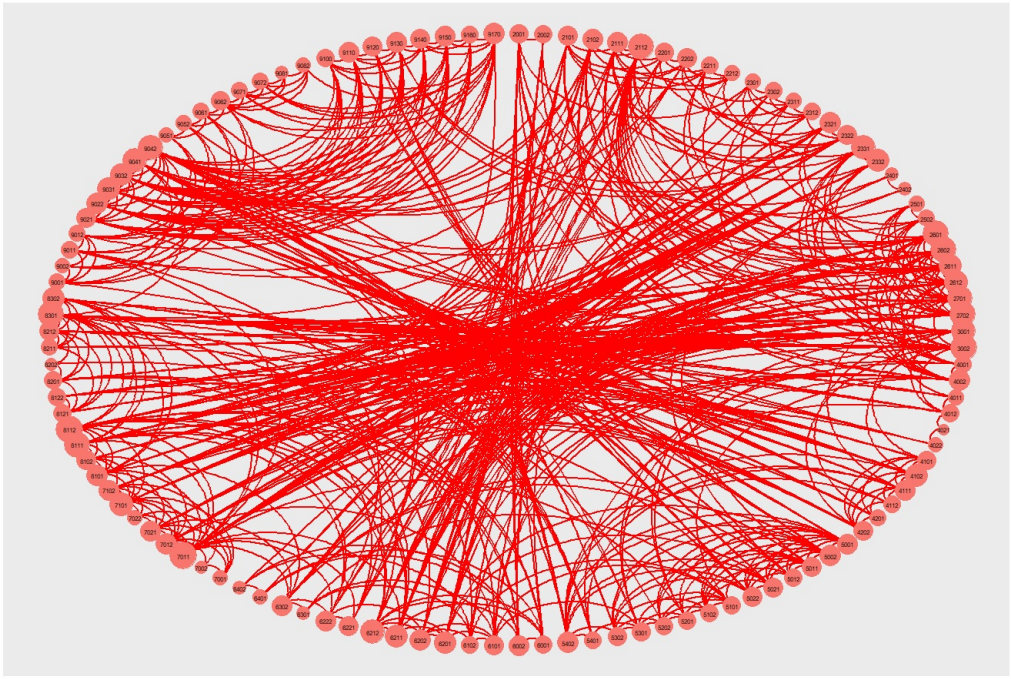
Without Bonferroni correction

In the following code we show the same sequence of plots but in the case where we are not using the Bonferroni correction in the graph estimate process, in order to consider the multiplicity.

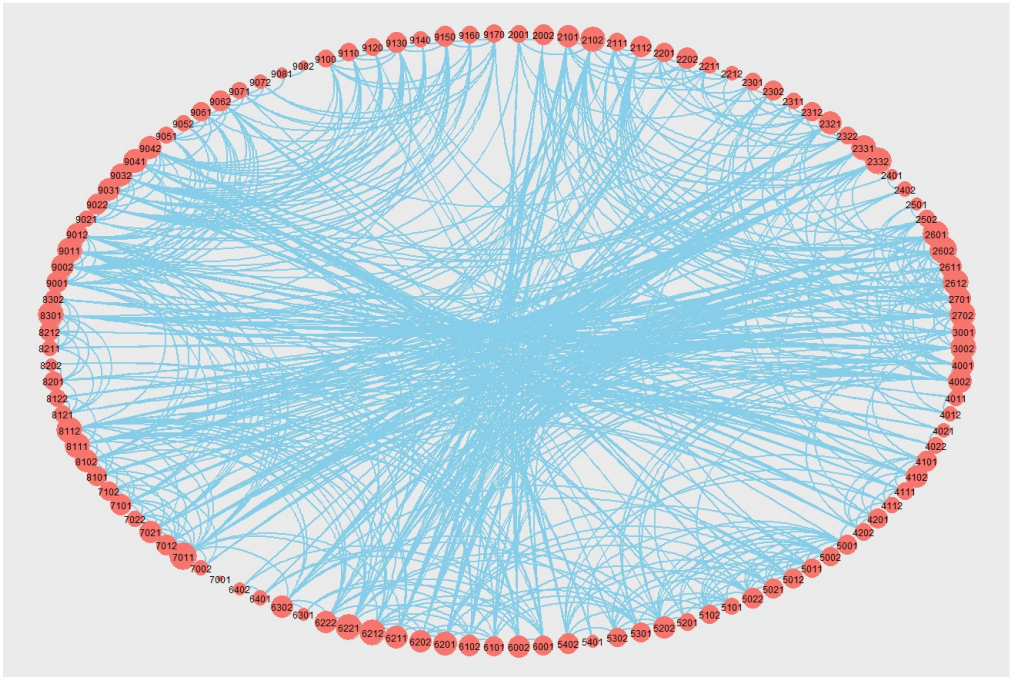
From the plots we can see how, without the control on multiplicity, we obtain an overflow of false connections between ROIs: this shows how without taking into account multiplicity, we get something that is meaningless or, however, not interpretable.

```
ASD_graph = graph_estimate(ASD_cor, correction=FALSE)
TD_graph = graph_estimate(TD_cor, correction=FALSE)

graph_visual(ASD_graph, TD_graph)
```

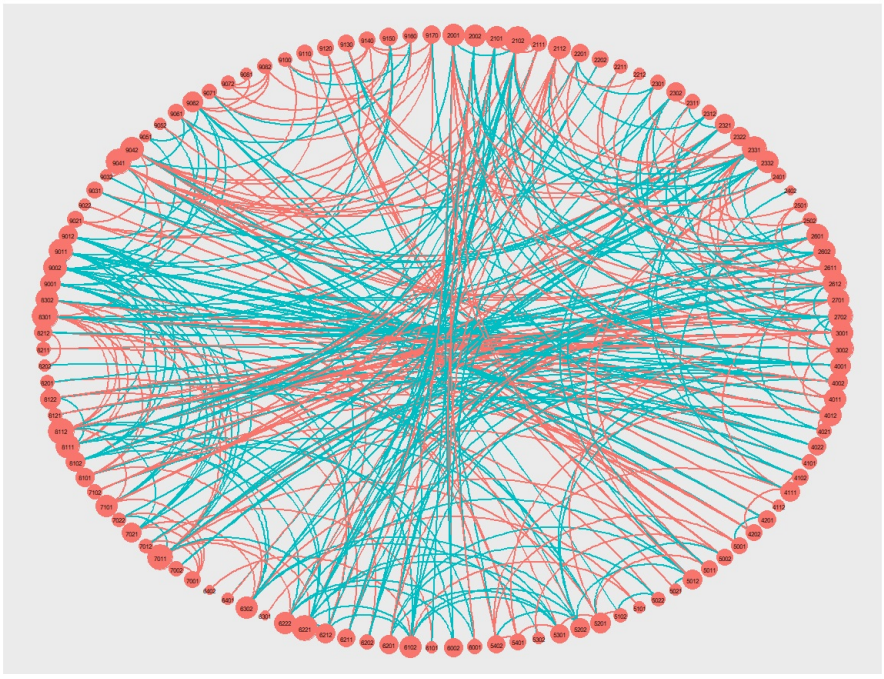
ASD graph



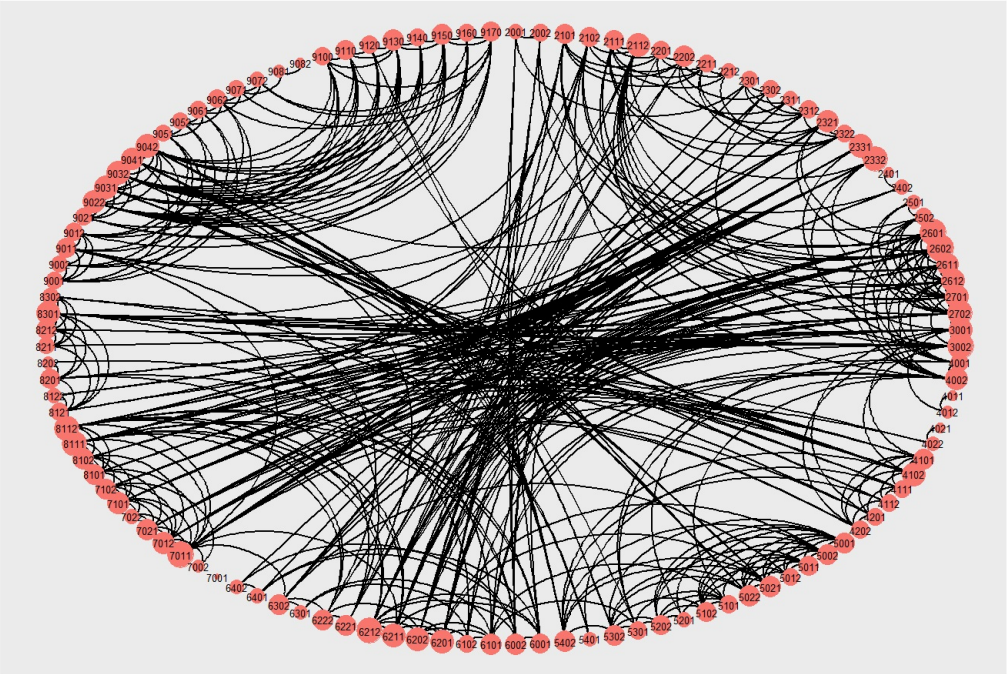
TD graph



Different edges



Common edges



Pooling with Median and Z-transform correlation

As second pooling method, we thought to apply the median instead of the empirical average.

The application of the median is performed by the following function, that return as output the matrix containing the median of all the correlation matrices for the given group.

input: *data* dataframe of our data, *cor_method* measure of association or correlation

output: the median correlation matrix

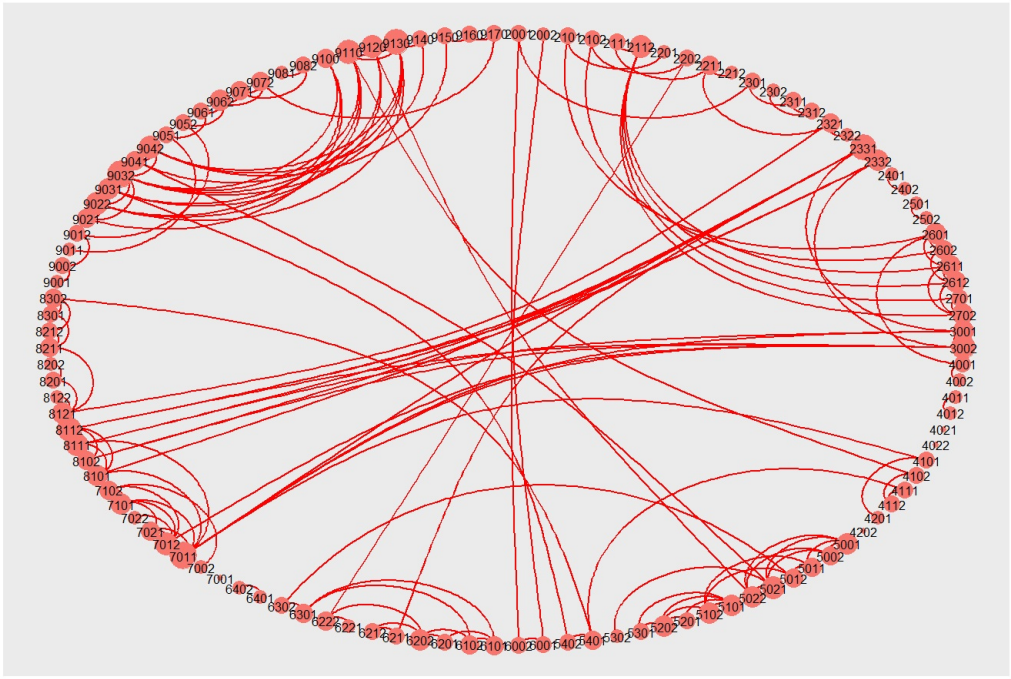
```
median_cor_matrix = function(data, cor_method = 'pearson'){

  tmp_matrix = matrix(NA, nrow=12, ncol=116*116)
  i = 1

  for(patient in data){
    cor_matrix = cor(patient, method=cor_method)
    tmp_matrix[i, ] = flatten(cor_matrix)
    i = i+1
  }

  region_labels = rownames(cor_matrix)   # just names of ROIs

  # Evaluate the median along patients of same group
  medians_matrix = colMedians(tmp_matrix)
  medians_matrix = resize(medians_matrix, 116, 116)

  # Re-define the labels of ROIs in the matrix
  colnames(medians_matrix) = region_labels
  rownames(medians_matrix) = region_labels

  return(medians_matrix)
}
```

We show the usual sequence of plots but, this time, in the case in which we are using the median as estimator of the correlation on the ROIs couples and for the graph construction itself.

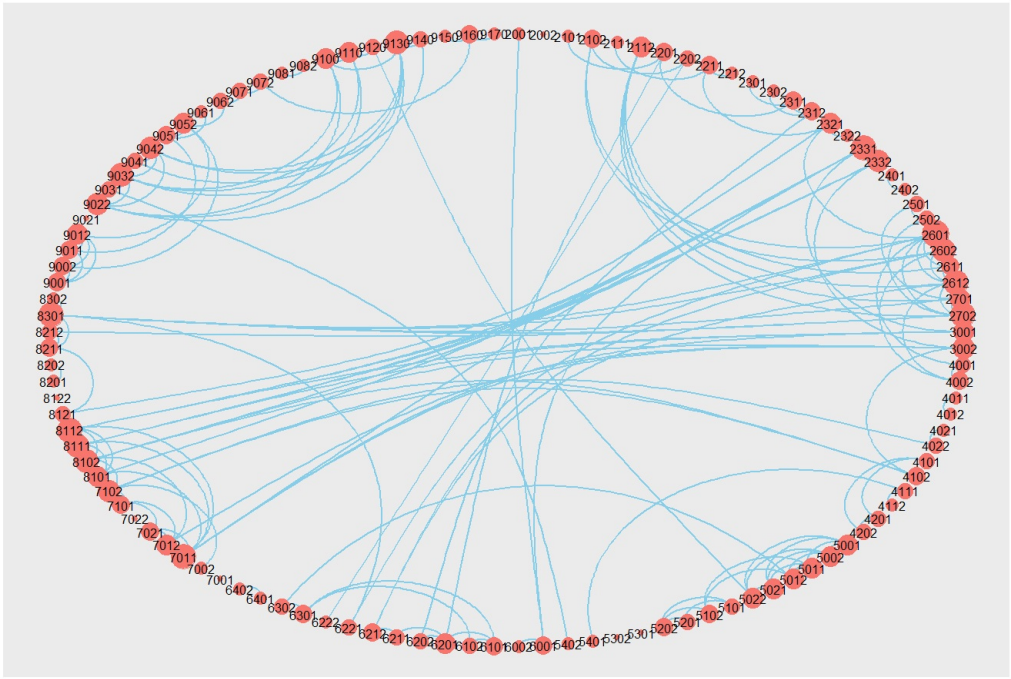
The plots are not so different from plots obtained by using the average as estimator of the correlations, especially for the first 2 representation of the ASD and TD graph. About the third plot, the observation done on the third plot of the average application seems to be not valid here, because unique connections of the TD between ROIs in $\sim \{9000, \ldots, 9170\}$ appear. With the last consideration and considering that the median is less affected by the presence of outliers, we think that the application of the median is able to capture more information about the connections among the ROIs, for this reason we will use the median as method of pooling data within the same group.

```
ASD_cor_median = median_cor_matrix(asd_sel)
TD_cor = median_cor_matrix(td_sel)

ASD_graph_m = graph_estimate(ASD_cor)
TD_graph_m = graph_estimate(TD_cor)

graph_visual(ASD_graph_m, TD_graph_m)
```

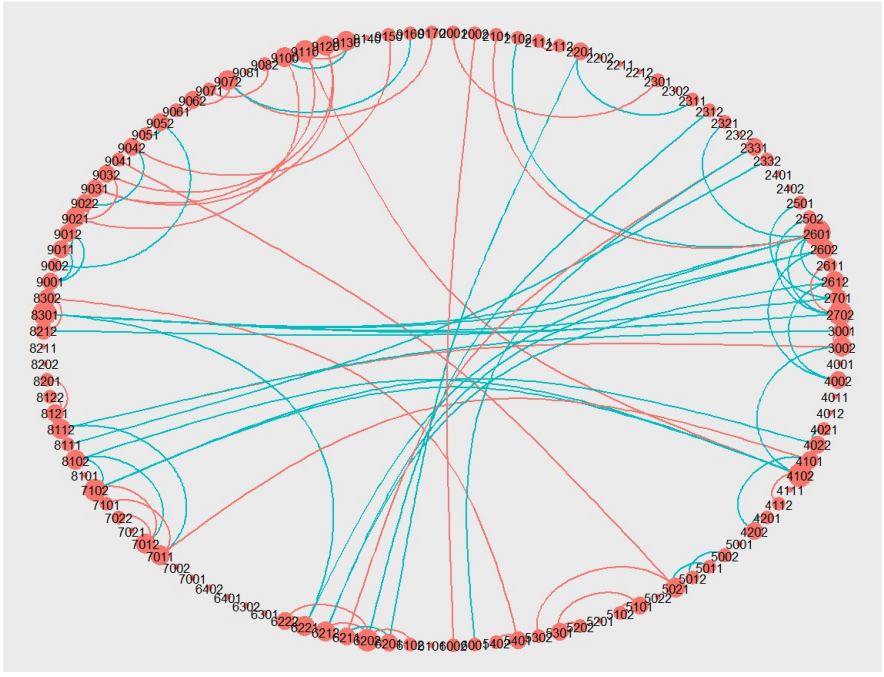
ASD graph



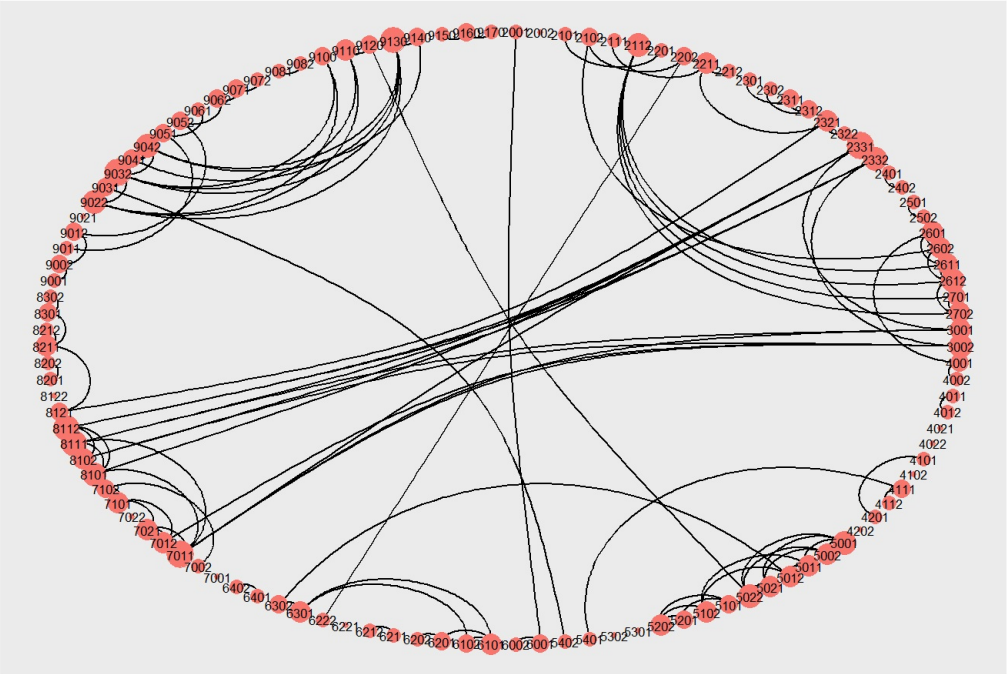
TD graph



Different edges



Common edges

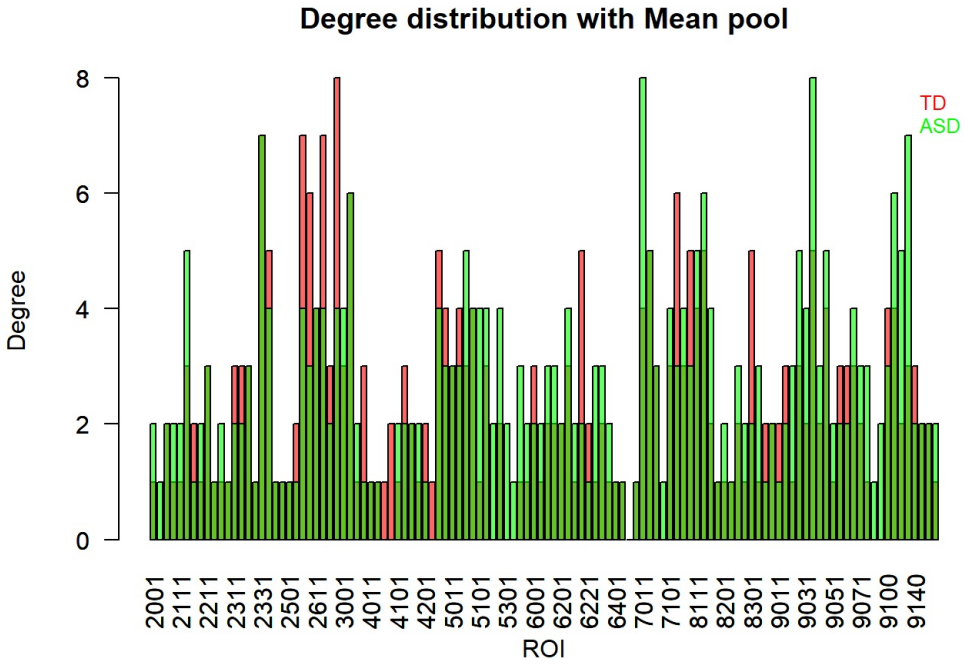


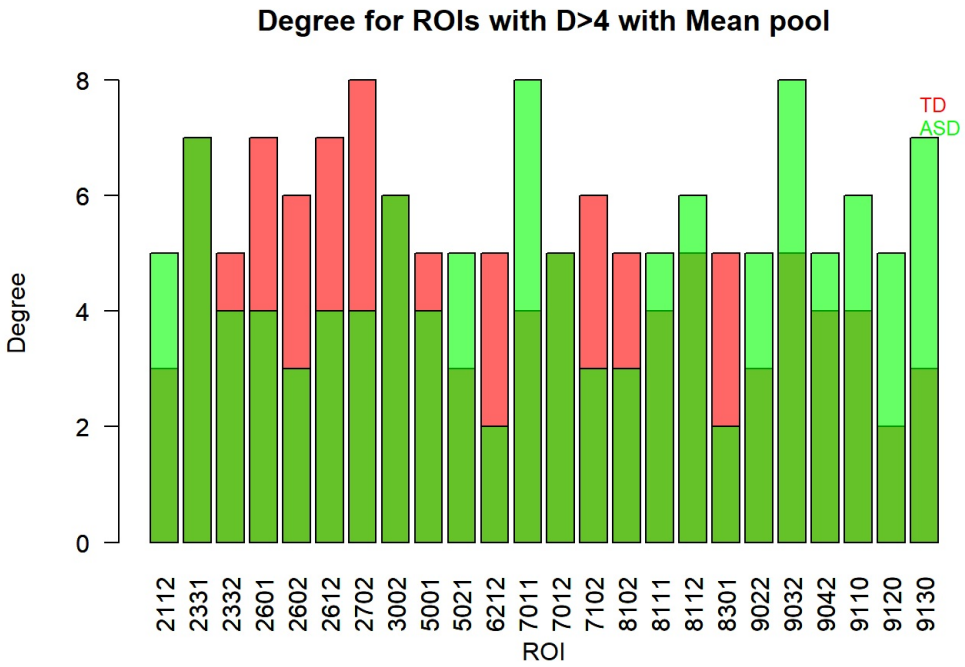
Degrees visualization

Moreover, we decided to visualize the degree distribution for the ASD and TD graph in both the case of mean and median pooling that we have implemented.

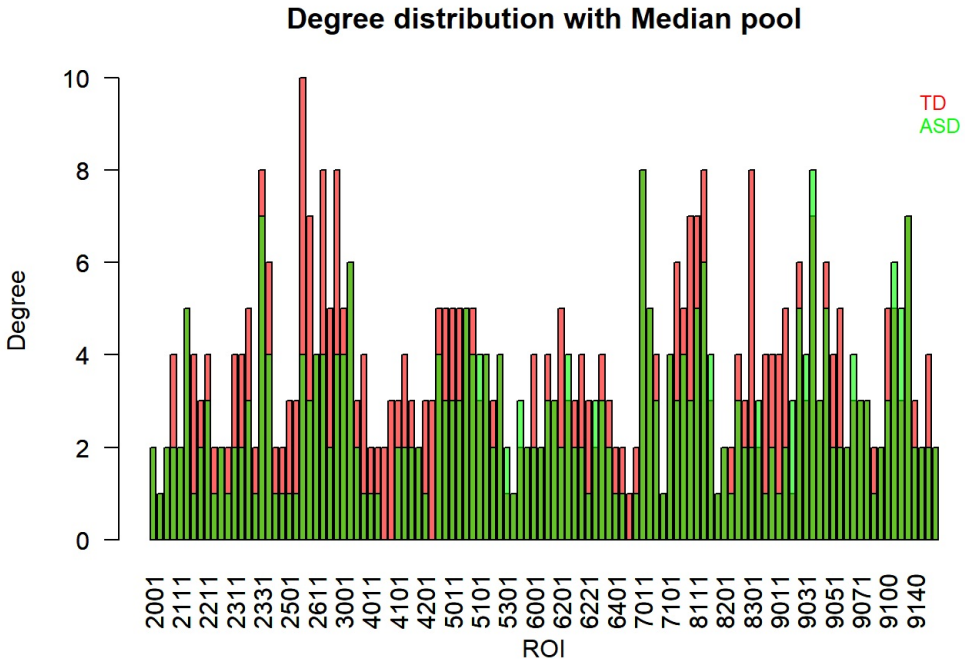
In the first plot of each pooling method we show the whole degree distribution, while in the second plot we represent just the distribution for the ROIs that have degree greater than 4 in one graph estimate, ASD, or the other, TD.

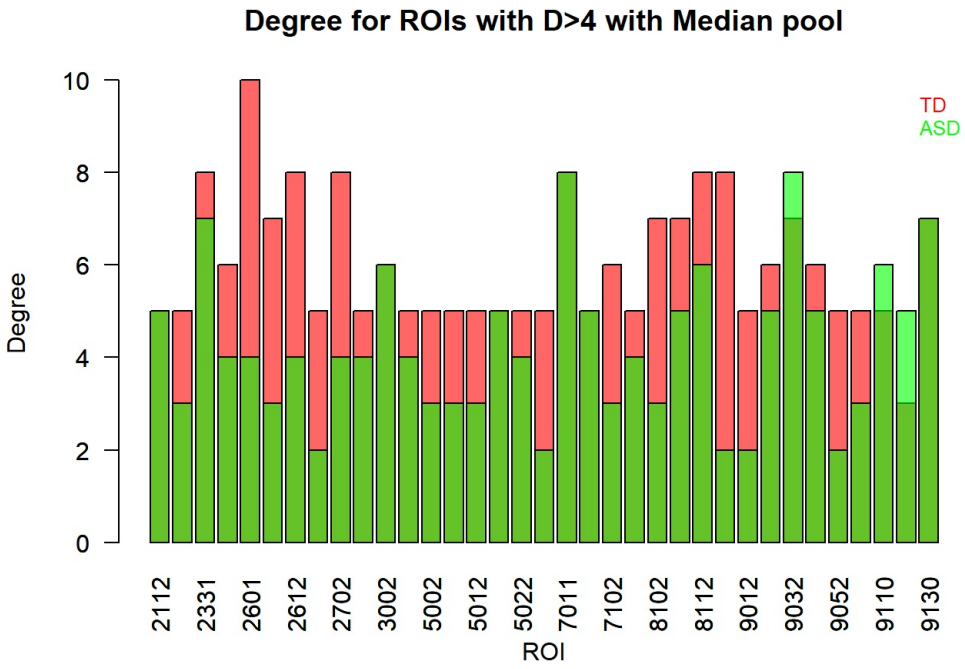
With this degree distributions analysis we observe that with the use of the pooling method based on the mean or on the median, we are introducing a substantial difference in the estimate of the degrees, consequently also of the graph, only in the TD case. So we confirm that we prefer to work with the median pooling method because it manage to capture more information about the TD graph structure.






```
## [1] "The total number of connections in the ASD graph with Mean pool is:"
## [1] 155
## [1] "The total number of connections in the TD graph with Mean pool is:"
## [1] 139
```






```
## [1] "The total number of connections in the ASD graph with Median pool is:"
## [1] 155
## [1] "The total number of connections in the TD graph with Median pool is:"
## [1] 217
```

Bootstrap time!

Now we want to study differences and similarities about the neural connectivity between the ASD and TD patients trying to estimate directly the graph induced by the difference between the correlation matrices for the 2 cases of study. In order to build this difference graph we perform a non-parametric bootstrap because we need more observations and this re-sampling technique allow us to do it.

The first step is to gather together all our observations related to a group and perform the non-parametric bootstrap on them. In order to build the single sample for each bootstrap iteration, we implement a random selection of the patients and a random selection of the time indices (145 time observations per patient) bpth with replacement; once we build the sample for the single bootstrap iteration, we evaluate the correlation matrix among ROIs on this particular sample and this matrix correlation will be a single row of our final matrix that will contain all the bootstrapped correlation matrices.

Notice the advantage to have observations in a single sample that are selected on different patients of the same autism group, while before we had to implement a specific pooling method in order to gather correlation matrices from different patients.

input: *data* data from a specific group, *patients* list of patients in the group, *B* total bootstrap number of iteration

ouput: *tmp_matrix* matrix containing all the boostrapped correlations

```
boot_correlation = function(data, patients, B){

  size = 145*12   # size of sample in each boot iteration

  region_labels = colnames(data[["caltech_0051472"]])

  tmp_matrix = matrix(NA, nrow=B, ncol=116*116)  # final matrix with boot. corr.

  for(b in 1:B){

    boot_matrix = matrix(NA, nrow=size, ncol=116)

    # Random selection of patients and times labels/indeces
    boot_patients = sample(patients, size=size, replace=TRUE)
    boot_times = sample(1:145, size=size, replace=TRUE)

    for(i in 1:size){
      boot_matrix[i, ] = as.double(data[[boot_patients[i]]][boot_times[i], ])
    }

    colnames(boot_matrix) = region_labels

    boot_cor = cor(boot_matrix)

    # Store boot. correlations in one row of the matrix with flatten() function
    tmp_matrix[b, ] = flatten(boot_cor)
  }

  return(tmp_matrix)
}
```

Once we have the bootstrapped correlations for both the groups, we need to build the difference graph. First of all we need to evaluate the differences between the two bootstrap matrices, then we implement the median along the columns of the difference matrix, this time not to pool observations from different patients but to build a summary of our bootstrap iterations.

Given a level $1 - \alpha$ and a small threshold t, we can build the difference graph with the same algorithm that we implemented in the previous analysis. We have chosen to work with a fixed $1 - \alpha = 0.95$ level and a tunable parameter t, in order to show how the graph structure varies as the parameter t varies.

input: *boot_matrix1* and *boot_matrix2* the bootstrap matrices for the 2 groups, *t* a threshold, *pool_method* median as default or mean available, *region_names* labels for ROIs

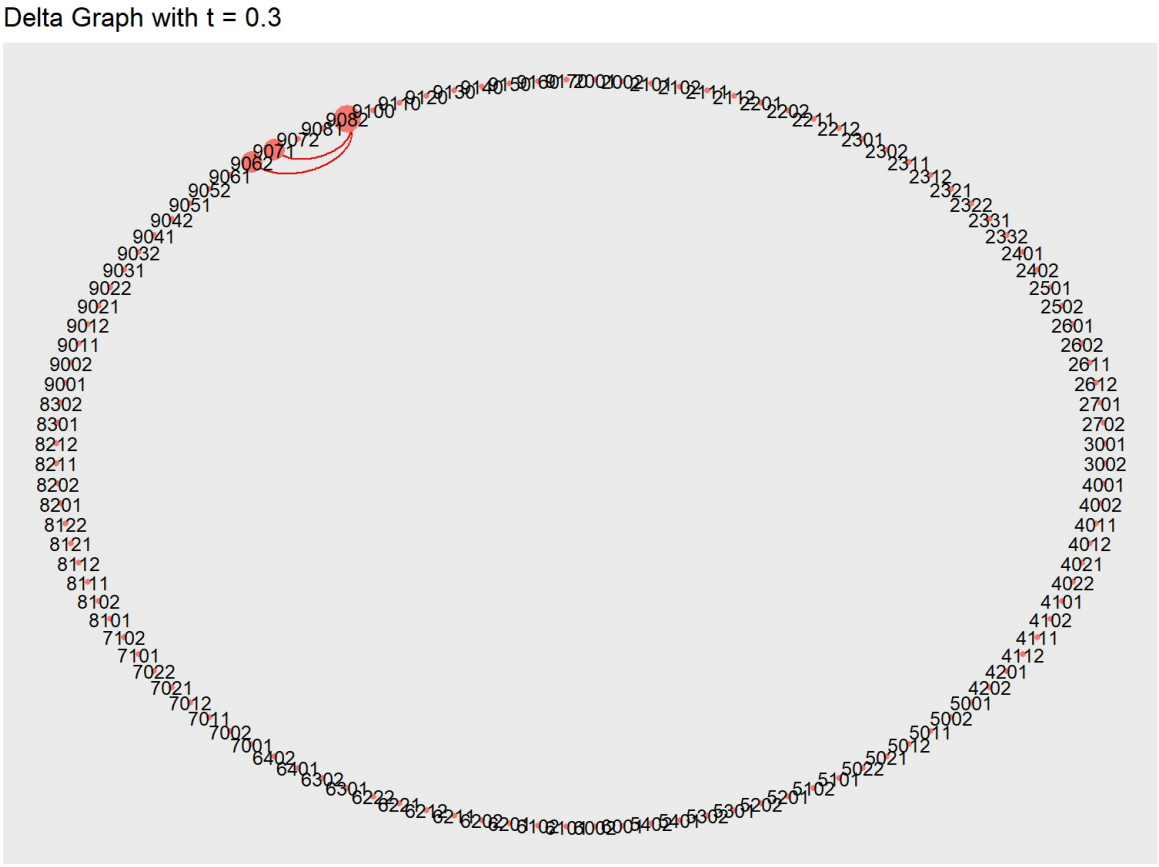
output: *graph* the adjacency matrix for the difference graph

```
boot_graph = function(boot_matrix1, boot_matrix2, t, pool_method='median', region_names = colnames(asd_sel[["calt
ech_0051472"]])){

  D = 116

  diff_boot = boot_matrix1-boot_matrix2  # delta bootstrapped correlations coef.

  # If we want use mean or median summary... we will use just median
  if(pool_method=='mean'){
    cor_hat = resize(colMeans2(diff_boot), D, D)
  } else{
    cor_hat = resize(colMedians(diff_boot), D, D)
  }
  colnames(cor_hat) = region_names
  rownames(cor_hat) = region_names

  # Evaluate standard deviation on bootstrap iterations
  sd_cor = resize(colSds(diff_boot), D, D)
  colnames(sd_cor) = region_names
  rownames(sd_cor) = region_names

  # Now we can build the graph as in the previous function... same reasoning
  # just different standard errors that now are defined by bootstrap procedure
  alpha = 0.05

  m = choose((D-1), 2)
  a = alpha/m

  low = cor_hat - sd_cor*qnorm(1-a/2)
  up = cor_hat + sd_cor*qnorm(1-a/2)

  t_up = matrix(t, D, D)
  t_low = matrix(-t, D, D)

  graph = apply(((up < t_low) | (low > t_up)), FUN = as.numeric,
                MARGIN = c(1, 2))

  return(graph)
}
```

In the followind code cell we implement the functions we have defined above in order to evaluate the difference graph for each threshold t that we have chosen, in particular we have seen that the interesting t-range is around $[0.01, 0.3]$ since above 0.3 essentially we don't have connections and below 0.01 we have too many connections.

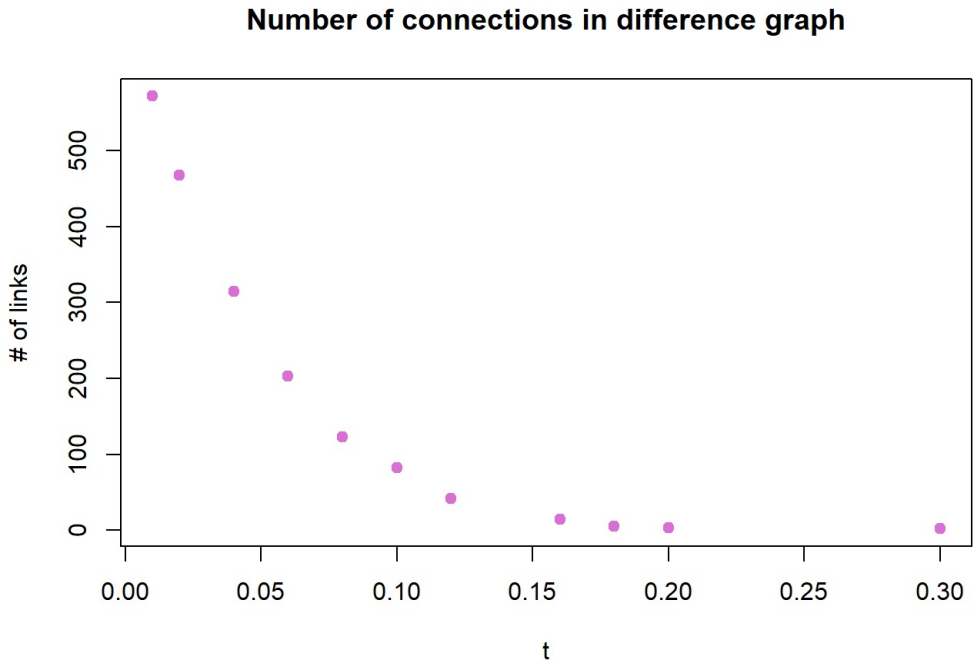
For each threshold t that we are analyzing, we produce a plot of the corresponding graph and, at the end, we plot them all together in a nice gift.

```
t_values = c(0.3, 0.2, 0.18, 0.16, 0.12, 0.1, 0.08, 0.06, 0.04, 0.02, 0.01)
links = c(rep(NA, length(t_values)))
i = 1

for(t in t_values){
  delta_g = boot_graph(asd_medians, td_medians, t=t)
  delta_graph = graph_from_adjacency_matrix(delta_g, diag = F)

  title = paste('t =', as.character(t))
  plot( ggraph(delta_graph, layout='linear', circular=T) +
        geom_edge_arc(color='red') +
        geom_node_point(show.legend=F, aes(size=degree(delta_graph),
                                           color='orange')) +
        geom_node_text(show.legend=F, aes(label=name, size=0.8)) +
    ggtitle(paste('Delta Graph with', title)))

  links[i] = sum(delta_g)/2
  i = i+1
}
```



For each analyzed threshold t we compute also the total number of connections between different ROIs and we plot its behavior as t varies in the range of interest.

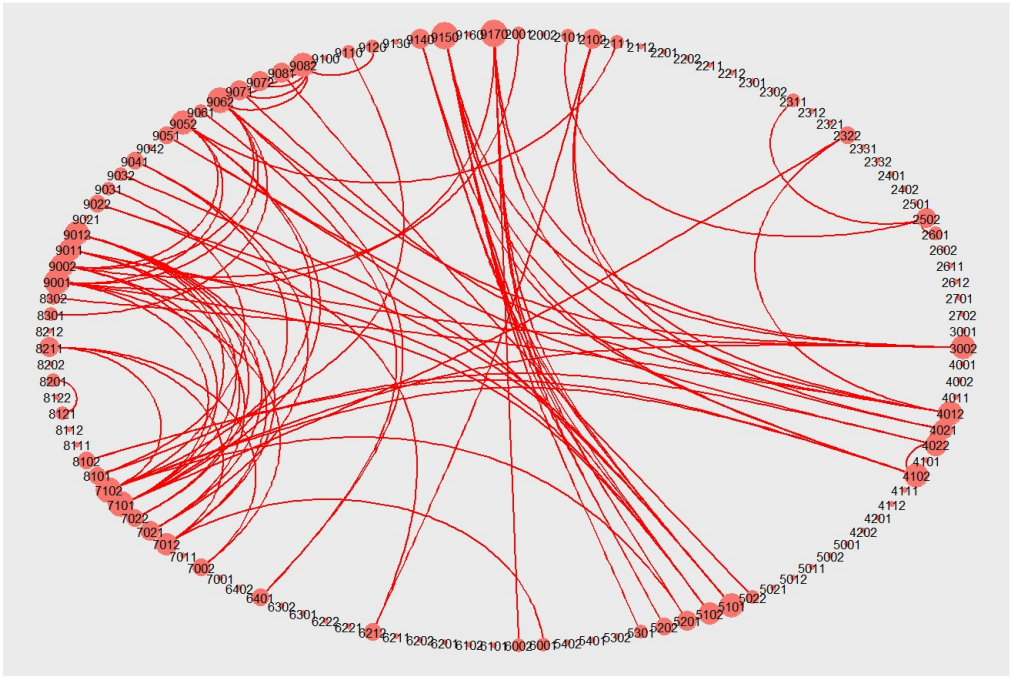
We can see how there is an significant increment or decrease in the number of connections around the interval $[0.05, 0.012]$.



Finally, we decide to show also a plot of the difference graph obtained for $t = 0.1$, that is the elbow of the curve.

It is interesting to notice how the area whose ROIs are labeled {7000, …, 9000} is dense in connections among themselves: especially this very connected part was not captured very well by the method used before the bootstrap, in which we reported the edges not in common between the two types of graph ASD and TD.

Delta Graph



Thanks for reading!

i. C. & M.M.