Homework 3

Authors: Adrienn Timea Aszalos - Joelelle Coley - Giovanni Giunta - Daniel Jiménez

Date: 2021/02/22

Subject: Statistics for Data Science

Data Science Master's programme

La Sapienza University of Rome

In this document, you can find a solution to the third homework proposed.

Setting environment up

First things first, for you to be able to run this file from R Studio is necessary to have the next packages installed on your PC:

- `igraph` : Used to plot the graph desired
- `ggplot2` : Used to provide statistical and other plots
- `tictoc` : Used to measure the running time of the executions
- `reshape` : Used to wrangle the data
- `dplyr` : Used to aggregate the data
- `RColorBrewer` : Used to get colors to plot
- `grid` : Used to define function `multiplot` to plot grid of line charts

Question 1

In the first place, we load both the libraries and the data. The latter was composed of two lists containing neuroimaging data of patients suffering from Autism Spectrum Disorder (ASD) and Typically Developed (TD) for 12 subjects.

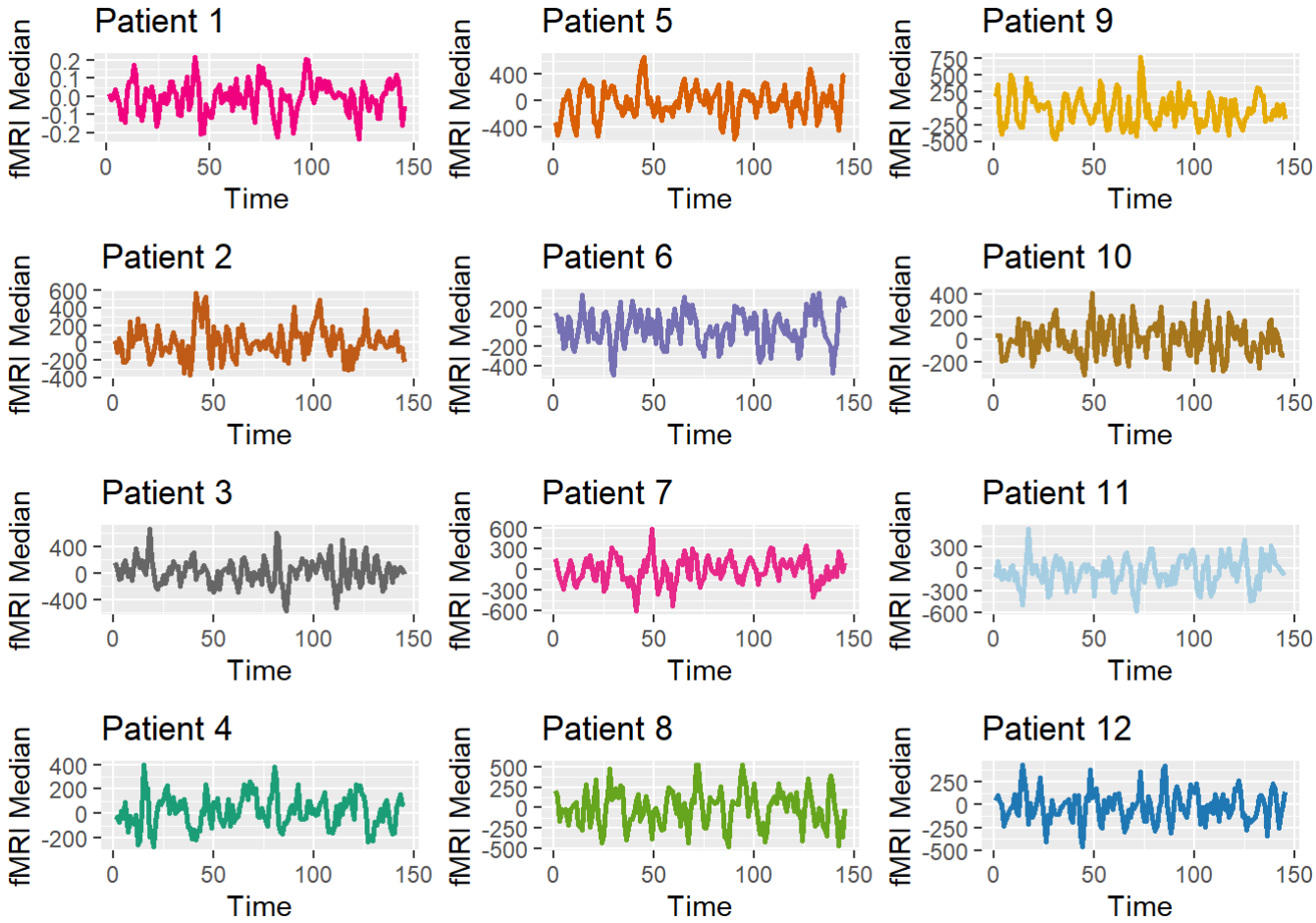
```
suppressWarnings(suppressMessages(require(igraph, quietly = T)))
suppressWarnings(suppressMessages(require(ggplot2, quietly = T)))
suppressWarnings(suppressMessages(require(tictoc, quietly = T)))
suppressWarnings(suppressMessages(require(reshape, quietly = T)))
suppressWarnings(suppressMessages(require(dplyr, quietly = T)))
suppressWarnings(suppressMessages(require(RColorBrewer, quietly = T)))
suppressWarnings(suppressMessages(require(grid, quietly = T)))
load("hw3_data.RData")
```

To pool the data together, we decided to use the `median` as a centrality measure since it will not be affected by possible outliers that can be found when measuring the Functional magnetic resonance imaging (fMRI). Besides, the aggregation was performed by each of the patients to compare how their fMRI looks like across the RoI. First, we analyzed the ASD group.

```
# Append the 12 dataframes in one
df_asd_sel <- do.call(rbind, asd_sel)

# Add an identification for the patient
df_asd_sel$Patient <- paste("Patient ",sort(rep(seq(1,dim(df_asd_sel)[1] / dim(asd_sel[[1]])[
1]),dim(asd_sel[[1]])[1])),sep="")

# Add an identification for time where the fMRI was measured
df_asd_sel$time_point_measure <- rep(seq(1,dim(asd_sel[[1]])[1]),dim(df_asd_sel)[1]/dim(asd_s
el[[1]])[1])

# Reshape the data to gruup it easily
df_asd_sel <- melt(df_asd_sel, id=c("Patient","time_point_measure"))

# Group data to get info of each individual along the time
asd_grouped <- df_asd_sel %>% group_by(Patient,time_point_measure)%>%
  summarise(median = median(value))
```

```
## `summarise()` regrouping output by 'Patient' (override with `.groups` argument)
```

```
# Define list to allocate plots
plots <- list()

# Define color to be used
qual_col_pals = brewer.pal.info[brewer.pal.info$category == 'qual',]
colors <- unique(unlist(mapply(brewer.pal, qual_col_pals$maxcolors, rownames(qual_col_pal
s))))

# Loop over each graph
for (i in 1:length(asd_sel)) {
  # Filter the dataframe to plot by each patient
  df_to_plot <- asd_grouped[asd_grouped$Patient == paste("Patient ",i,sep = ""),]

  # Get the desired line plot
  p1 = ggplot(df_to_plot, aes(x=time_point_measure, y=median)) +
    geom_line(color = colors[i+5],size = 1) +  ggtitle(paste("Patient ",i,sep = "")) +
    xlab("Time") + ylab("fMRI Median")

  # Store plot in list
  plots[[i]] <- p1  # add each plot into plot list
}

# PLot information by each patient
multiplot(plotlist = plots, cols = 3)
```



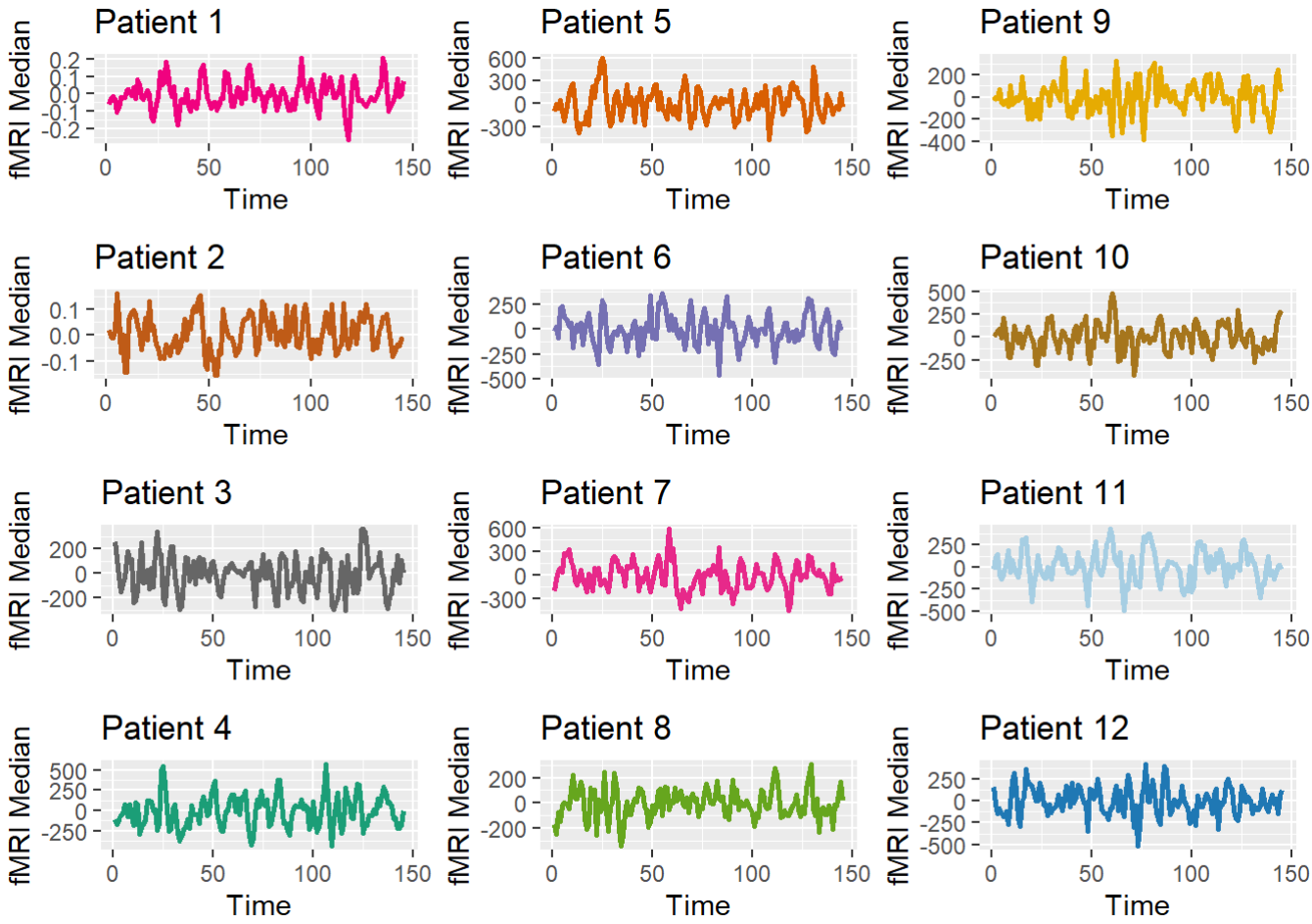
From the previous charts, we saw that patient 1 has low measures compared with the others. Besides, patient 9 has a high range of measurements since the minimum and maximum median values are between -500 and 750. That interval has a high amplitude compared with the remaining 11 patients. Finally, all the medians of the measurements tend to be around 0 with no remarkable tendency.

Later we did the same process for the TD group to get some analysis of their behaviour.

```
# Append the 12 dataframes in one
df_td_sel <- do.call(rbind, td_sel)

# Add an identification for the patient
df_td_sel$Patient <- paste("Patient ",sort(rep(seq(1,dim(df_td_sel)[1] / dim(td_sel[[1]])[1
]),dim(td_sel[[1]])[1])),sep="")

# Add an identification for time where the fMRI was measured
df_td_sel$time_point_measure <- rep(seq(1,dim(td_sel[[1]])[1]),dim(df_td_sel)[1]/dim(td_sel[[
1]])[1])

# Reshape the data to gruup it easily
df_td_sel <- melt(df_td_sel, id=c("Patient","time_point_measure"))

# Group data to get info of each individual along the time
td_grouped <- df_td_sel %>% group_by(Patient,time_point_measure)%>%
  summarise(median = median(value))
```

```
## `summarise()` regrouping output by 'Patient' (override with `.groups` argument)
```

```
# Define list to allocate plots
plots <- list()

# Define color to be used
qual_col_pals = brewer.pal.info[brewer.pal.info$category == 'qual',]
colors <- unique(unlist(mapply(brewer.pal, qual_col_pals$maxcolors, rownames(qual_col_pal
s))))

# Loop over each graph
for (i in 1:length(td_sel)) {
  # Filter the dataframe to plot by each patient
  df_to_plot <- td_grouped[td_grouped$Patient == paste("Patient ",i,sep = ""),]

  # Get the desired line plot
  p1 = ggplot(df_to_plot, aes(x=time_point_measure, y=median)) +
    geom_line(color = colors[i+5],size = 1) +  ggtitle(paste("Patient ",i,sep = "")) +
    xlab("Time") + ylab("fMRI Median")

  # Store plot in list
  plots[[i]] <- p1  # add each plot into plot list
}

# PLot information by each patient
multiplot(plotlist = plots, cols = 3)
```



From the previous charts, we saw that patients 1 and 2 have low measures compared with the others. Furthermore, patients 5 and 7 have a high range of measurements since the minimum and maximum median values are between -300 and 600. That interval has a high amplitude compared with the remaining 10 patients.

Finally, all the medians of the measurements (again) tend to be around 0 with no remarkable tendency.

Question 2

Now, we defined a function that receives the next parameters:

data: Initial list of raw data to process **alpha:** Significance level to calculate confidence intervals **appl_bonf:** Boolean to select if the Bonferroni correction should be applied or not **categories:** Number of categories considered for the Bonferroni correction

```
get_estimates <-function(data,alpha = 0.05, appl_bonf = T, categories = 116){
  # data: Initial list of raw data to process
  # alpha: Significance level to calculate confidence intervals
  # appl_bonf: Boolean to select if the Bonferroni correction should be applied or not
  # categories: Number of categories considered for the Bonferroni correction

  # Calculate correlation matrix inside of each element of the list of data
  correlations <- lapply(data, cor)

  # Create matrix to store in each position a list with the correlations of each individual
  aux <- matrix(data = list(),nrow = dim(correlations[[1]])[1],ncol = dim(correlations[[1]])[
2])

  # Iterate over each element of the list
  for (i in 1:length(data)){
    # Iterate over each dimension of the matrix of correlation
    for (j in 1:dim(correlations[[i]])[1]){
      for (k in 1:dim(correlations[[i]])[2]){

        # Store in the aux matrix 12 values (as a list) corresponding to 12 individuals
        aux[j, k] <- list(append(unlist(aux[j, k]),correlations[[i]][j, k],after = length(aux
[j, k])))

      }
    }

  }

  # Initialize matrix of 80th percentiles
  percentiles <- matrix(data = NA,nrow = dim(correlations[[1]])[1],ncol = dim(correlations[[1
]])[2])

  # Initialize matrix of confidence intervals
  conf_int <- matrix(data = list(),nrow = dim(correlations[[1]])[1],ncol = dim(correlations[[
1]])[2])

  # Initialize matrix of adjacency for the final graph
  adj_mat <- matrix(data = NA,nrow = dim(correlations[[1]])[1],ncol = dim(correlations[[1]])[
2])

  # Iterate over each row and column (each positon) of the matrix created before
  for (i in 1:dim(aux)[1]){
    for (j in 1:dim(aux)[2]){

      # Calculate the 80th percentile of the 12 individuals
      percentiles[i, j] <-  quantile(unlist(aux[i, j]), p = 0.8)

      # Perform Fisher Z-transformation for normality
      aux_fisher <-  0.5*(log(1 + unlist(aux[i, j])) - log(1 - unlist(aux[i, j])))

      # Correct infinite values caused by transformation
      aux_fisher[is.infinite(aux_fisher)] <- 1

      # Check if Bonferroni correction is declared
      if(appl_bonf == T){
```

```
        # Calculate the alpha corrected by Bonferroni
        alpha_bonf <- alpha / categories

        # Calculate the quantile of the normal distribution with the previous corrected signi
ficance level
        z_bonf <- qnorm(1 - alpha_bonf / 2)

        # Calculate the sample size
        n <- length(data)

        # Compute the confidence intervals intervals and revert the Fisher Z-transformation
        conf_int[i, j] <- list(tanh(mean(aux_fisher) + c(-1,1)*z_bonf*sqrt((1/(n - 3))/n)))

      } else{
        # Calculate the quantile of the normal distribution with the original significance le
vel
        z <- qnorm(1 - alpha / 2)

        # Calculate the sample size
        n <- length(data)

        # Compute the confidence intervals intervals and revert the Fisher Z-transformation
        conf_int[i, j] <- list(tanh(mean(aux_fisher) + c(-1,1)*z*sqrt((1/(n - 3))/n)))

      }

      # Extract the current lower bound of the confidence interval
      low <- unlist(conf_int[i, j])[1]

      # Extract the current upper bound of the confidence interval
      upp <- unlist(conf_int[i, j])[2]

      # Extract the current threshold
      t <- percentiles[i, j]

      # Evaluate whether the threshold is contained on the confidence intervals or not
      if((t >= low & t <= upp) & (-t >= low & -t <= upp)){
        # Do not place an edge
        adj_mat[i, j] <- 0
      } else {
        # Place an edge
        adj_mat[i, j] <- 1
      }

    }

  }

  # Update the main diagonal of the adjacency matrix with 0 to avoid nodes connected to itsel
f
  diag(adj_mat) <- 0

  return(adj_mat)
}
```

Within `get_estimates` function, we followed the next procedure to estimate the adjacency matrix:

1. Calculate the correlation matrix inside each one of the 12 data frames of individuals. For this, we used the `lapply` and the `cor` functions.
2. With the previous, we had 12 correlations matrix.
3. Then, we reorganized the data to have a matrix of size 116x116. In each one of the positions of that matrix, we will have 12 values that represent the 12 individuals.
4. Then, we took the 80th percentile inside each position of the previous matrix. In this step, we finish with a single matrix of size 116x116.
5. Later, using each position of the matrix generated in 2 (and applying Fisher Z-transformation), we calculate a confidence interval for the mean value of the correlations that we have. The latter leads to a matrix of 116x116, wherein each position of that matrix we have 2 values: the upper and lower bounds of the confidence interval. Besides, we apply the Fisher Z-transform to those values. In the end, we reversed the Fisher transformation to get the initial units for the correlations.
6. Finally, we checked whether the 80th percentile of each value of that matrix is outside the confidence interval to place an edge. With this step, we return the adjacency matrix to create the graph.

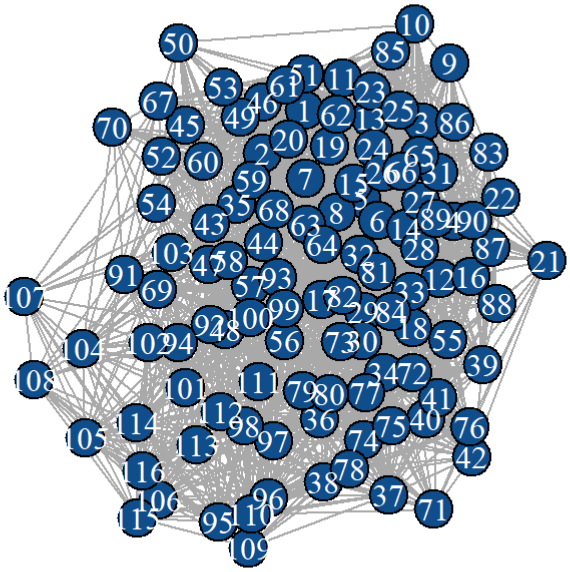
It is relevant to highlight that we decided to work with each group of individuals as a single graph because they have the same disorder. With this choice, we generated 2 graphs, one for each group, ASD and TD.

In addition, both graphs used the mean of each group of correlations (over the 12 individuals) to get the confidence intervals since it makes the results more readable and accurate.

Question 3

The first graph that we created was the one referred to as ASD patients. We used 116 categories to make the Bonferroni correction since that is the number of different regions of interest (RoI). It's important to remark that to create the whole confidence intervals, we used a significance level (α) of 0.05.

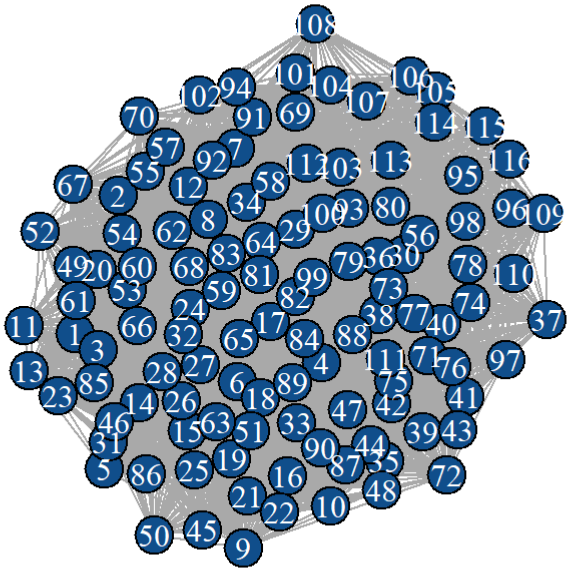
```
# Calculate adjacency matrix applying Bonferroni correction
adjacency_matrix <- get_estimates(asd_sel, categories = 116)

# Build the graph
G <- graph_from_adjacency_matrix(adjacency_matrix, mode = "undirected")
# Calculate number of edges
E <- length(E(G))
# Calculate number of vertexes
V <- length(V(G))
# Plot graph
plot(G, vertex.color="dodgerblue4", main=paste("ASD Graph with ", V," vertexes and ", E, " ed
ges",sep=""), vertex.label.color="white")
```



Examining the previous graph, we see the 116 edges generated from the RoI, and we have about 2000 edges that turned out to be statistically significative, concerning the correlation among the regions.

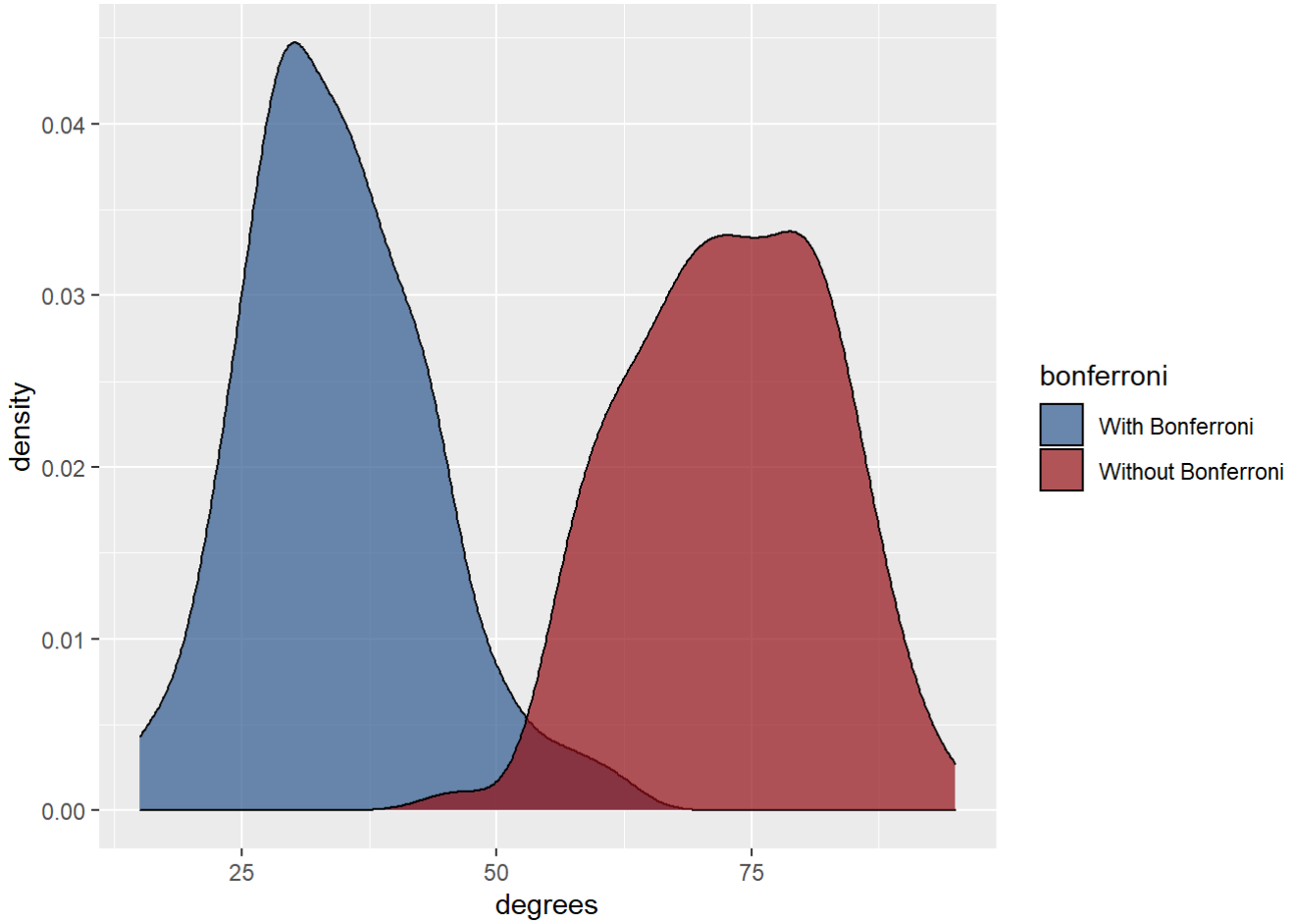
The next graph that was referred to ASD patients, but this time without using the Bonferroni correction.

```
# Store the degrees of the graph when appyting Bonferroni
degrees_with_bonf <- data.frame(bonferroni = "With Bonferroni", degrees = degree(G))

# Calculate density of the graph
D_asd_with_bonf <- 2*E / (V*(V-1))

# Calculate adjacency matrix without applying Bonferroni correction
adjacency_matrix <- get_estimates(asd_sel, categories = 116, appl_bonf = F)

# Build the graph
G <- graph_from_adjacency_matrix(adjacency_matrix, mode = "undirected")
# Calculate number of edges
E <- length(E(G))
# Calculate number of vertexes
V <- length(V(G))
# Plot graph
plot(G, vertex.color="dodgerblue4", main=paste("ASD Graph with ", V," vertexes and ", E, " ed
ges",sep=""), vertex.label.color="white")
```



In the preceding graph again, we see the 116 edges generated from the RoI. But this time (as expected) the number of edges increased at most the double. It reflects that if we don't use the Bonferroni correction, we will be creating edges that are not significative since we are inferring multiple conclusions based only on one data set.

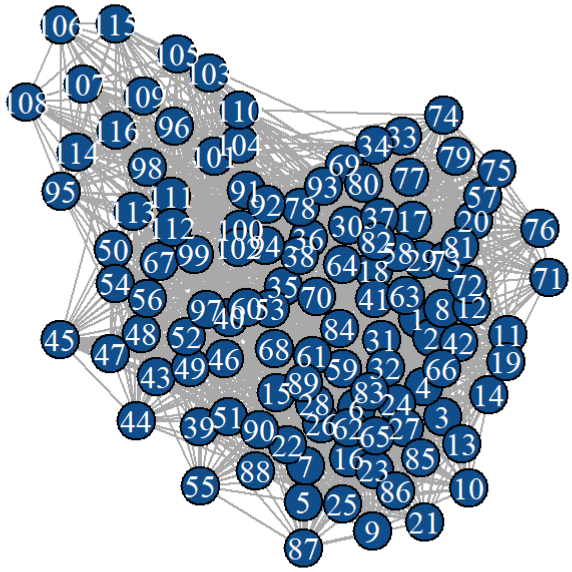
```
# Store the degrees of the graph when avoiding Bonferroni
degrees_without_bonf <- data.frame(bonferroni = "Without Bonferroni", degrees = degree(G))

# Calculate density of the graph
D_asd_without_bonf <- 2*E / (V*(V-1))

# Plot densities of the degrees when applying/avoiding Bonferroni correction
df_to_plot <- rbind(degrees_with_bonf,degrees_without_bonf)
ggplot(df_to_plot,aes(x=degrees, fill=bonferroni)) + geom_density(alpha=0.7) + scale_fill_man
ual(values=c("#2e598f","#940f15"))
```



Also, in the graph above, if we consider the nodes distribution for both graphs (with and without Bonferroni), we can see a remarkable difference from one to another. When we don't apply the correction, the degrees tend to be around 30 on average. On the other hand, when skipping Bonferroni, the degrees of the graph is more around 75.

Later we proceeded to do a similar analysis for the TD group. Similarly to ASD, we used 116 categories to make the Bonferroni correction, since that is the number of different regions of interest (RoI).

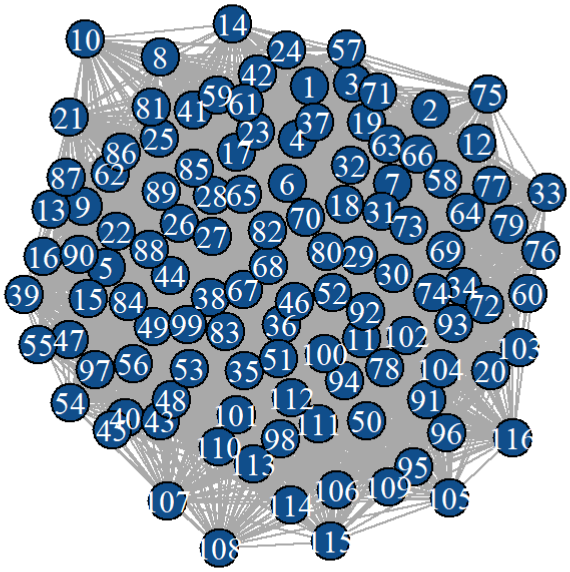
```
# Calculate adjacency matrix applying Bonferroni correction
adjacency_matrix <- get_estimates(td_sel, categories = 116)

# Build the graph
G <- graph_from_adjacency_matrix(adjacency_matrix, mode = "undirected")
# Calculate number of edges
E <- length(E(G))
# Calculate number of vertexes
V <- length(V(G))
# Plot graph
plot(G, vertex.color="dodgerblue4", main=paste("TD Graph with ", V," vertexes and ", E, " edg
es",sep=""), vertex.label.color="white")
```



Checking out this graph, we see the 116 edges generated from the RoI, and it has about 2100 edges that turned out to be statistically significative, to the correlation among the regions.

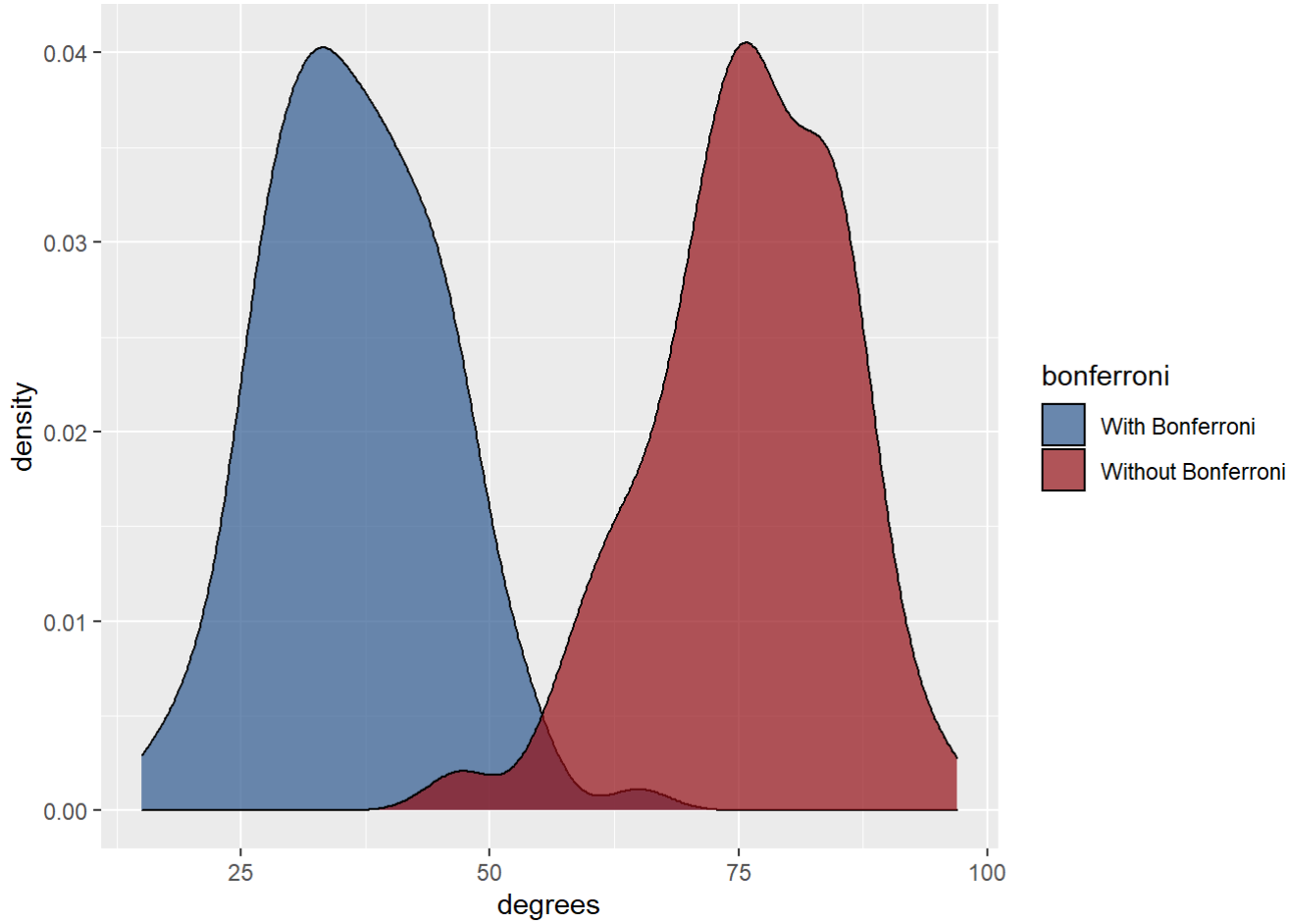
The next graph was referred to TD patients, but this time without using the Bonferroni correction.

```
# Store the degrees of the graph when appyting Bonferroni
degrees_with_bonf <- data.frame(bonferroni = "With Bonferroni", degrees = degree(G))

# Calculate density of the graph
D_td_with_bonf <- 2*E / (V*(V-1))

# Calculate adjacency matrix without applying Bonferroni correction
adjacency_matrix <- get_estimates(td_sel, categories = 116, appl_bonf = F)

# Build the graph
G <- graph_from_adjacency_matrix(adjacency_matrix, mode = "undirected")
# Calculate number of edges
E <- length(E(G))
# Calculate number of vertexes
V <- length(V(G))
# Plot graph
plot(G, vertex.color="dodgerblue4", main=paste("TD Graph with ", V," vertexes and ", E, " edg
es",sep=""), vertex.label.color="white")
```



One more time, we see the 116 edges generated from the RoI. But the number of edges again was doubled. The previous means that whether we don't use the Bonferroni correction, we will be creating accurate edges, since we are inferring multiple conclusions based only in one data set.

```
# Store the degrees of the graph when avoiding Bonferroni
degrees_without_bonf <- data.frame(bonferroni = "Without Bonferroni", degrees = degree(G))

# Calculate density of the graph
D_td_without_bonf <- 2*E / (V*(V-1))

# Plot densities of the degrees when applying/avoiding Bonferroni correction
df_to_plot <- rbind(degrees_with_bonf,degrees_without_bonf)
ggplot(df_to_plot,aes(x=degrees, fill=bonferroni)) + geom_density(alpha=0.7) + scale_fill_man
ual(values=c("#2e598f","#940f15"))
```



In the upper graph, when we consider the distribution of the nodes for both graphs (with and without Bonferroni) we can precise a significative difference from one to another. If we don't apply the correction, the degrees tend to be around 35 on average. On the other hand, when skipping Bonferroni, the degrees of the graph are more around 75.

Finally, we decided to compare the differences of both groups (ASD and TD) by calculating the density of each graph.

```
print(paste("Density of ASD graph with Bonferroni: ",D_asd_with_bonf, sep = ""))
```

```
## [1] "Density of ASD graph with Bonferroni: 0.29880059970015"
```

```
print(paste("Density of TD graph with Bonferroni: ",D_td_with_bonf, sep = ""))
```

```
## [1] "Density of TD graph with Bonferroni: 0.316341829085457"
```

```
print(paste("Density of ASD graph without Bonferroni: ",D_asd_without_bonf, sep = ""))
```

```
## [1] "Density of ASD graph without Bonferroni: 0.632983508245877"
```

```
print(paste("Density of TD graph without Bonferroni: ",D_td_without_bonf, sep = ""))
```

```
## [1] "Density of TD graph without Bonferroni: 0.659070464767616"
```

With the previous information we can see that (for both corrected and uncorrected graphs) there are no clear co-activation differences between the two groups. The later based on that the density of each couple of graphs is almost the same.

Bonus Question

To make things more readable, we created the function `reshape_data``that receives the list of correlation matrices to be reshaped. We reshaped it as a matrix of size 116x116. Each position corresponds to a list of each of the association metrics taken for the individuals and its RoI. We picked Pearson correlation as an association measure since it correctly represents possible correlations, as shown in the exercises above.

```
reshape_data <- function(corr_mat){
  # corr_mat: List of correlation matrix to reshape

  # Create matrix to store in each position a list with the correlations of each individual
  aux <- matrix(data = list(),nrow = dim(corr_mat[[1]])[1],ncol = dim(corr_mat[[1]])[2])

  # Iterate over each element of the list
  for (i in 1:length(corr_mat)){
    # Iterate over each dimension of the matrix of correlation
    for (j in 1:dim(corr_mat[[i]])[1]){
      for (k in 1:dim(corr_mat[[i]])[2]){

        # Store in the aux matrix n values (as a list) corresponding to n individuals
        aux[j, k] <- list(append(unlist(aux[j, k]),corr_mat[[i]][j, k],after = length(aux[j,
 k])))

      }
    }

  }

  return(aux)
}
```

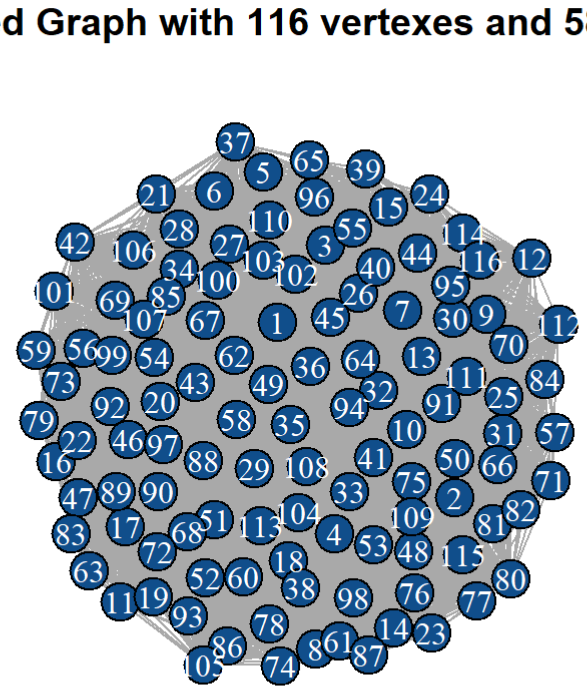
To execute the Bootstrap procedure, we used 1000 iterations and with a sample size of 50 individuals for each iteration. Furthermore, we set a threshold of $t = 0.1$ representing that we admit differences up to 10 points from the difference in terms of the correlation between ASD and TD, to place an edge in the difference graph. Again, to apply the Bonferroni correction, we used 116 as represented by the RoI that we are dealing with.

```
tic()
# Bootstrap
# Set number of iterations

M <- 1000

# Set sample size in each iteration
n <- 50

# Define the seed for replicability
seed <- 1234

# Set threshold
t <- 0.1

# Set number of categories for Bonferroni correction
categories <- 116

# Significance level to calculate confidence intervals
alpha <- 0.05

# Pre-allocate matrix to store mean of the correlations
mean_corr_boot <- vector(mode = "list", length = M)

# Loop over desired Bootstrap size
for(l in 1:M){
  # Get sample from ASD
  set.seed(seed)
  sample_asd <-sample(asd_sel, n, replace=TRUE)

  # Get sample from TD
  set.seed(seed)
  sample_td <-sample(td_sel, n, replace=TRUE)

  # Calculate correlations inside of list for each sample
  correlations_asd <- lapply(sample_asd, cor)
  correlations_td  <- lapply(sample_td, cor)

  # Calculate difference of correlation in absolute value
  delta_correlation <- lapply(mapply('-', correlations_asd, correlations_td, SIMPLIFY = FALSE
),abs)

  # Reshape data to get a matrix with list inside of each position
  aux <- reshape_data(delta_correlation)

  # Calculate the mean of the correlation values for each position
  mean_corr <- matrix(data = NA,nrow = dim(delta_correlation[[1]])[1],ncol = dim(delta_correl
ation[[1]])[2])

  # Loop over the matrix of means of correlation
  for (i in 1:dim(aux)[1]){
    for (j in 1:dim(aux)[2]){

      # Calculate the mean over correlations and store in matrix of Boostrap means
      mean_corr[i, j] <-  mean(unlist(aux[i, j]), na.rm = T)
    }
```

```
  }

  # Save the matrix of means
  mean_corr_boot[[1]] <- mean_corr


  # Modify seed
  seed <- seed + 100
}

# Reshape data to get a matrix with list of means inside of each position
aux_means_boot <- reshape_data(mean_corr_boot)

# Pre-allocate matrix to store confidence intervals
conf_int_boot <- matrix(data = list(),nrow = dim(mean_corr_boot[[1]])[1],ncol = dim(mean_corr
_boot[[1]])[2])
# Pre-allocate matrix of adjacency
adj_mat_boot <- matrix(data = NA,nrow = dim(mean_corr_boot[[1]])[1],ncol = dim(mean_corr_boot
[[1]])[2])

# Lopp over means of Bootstrap
for (i in 1:dim(aux_means_boot)[1]){
  for (j in 1:dim(aux_means_boot)[2]){
    # Calculate the alpha corrected by Bonferroni
    alpha_bonf <- alpha / categories

    # Calculate the quantile of the normal distribution with the previous corrected significa
nce level
    z_bonf <- qnorm(1 - alpha_bonf / 2)

    # Compute the confidence intervals intervals
    conf_int_boot[i, j] <- list(mean(unlist(aux_means_boot[i, j])) + c(-1,1)*z_bonf*sd(unlist
(aux_means_boot[i, j])))

    # Extract the current lower bound of the confidence interval
    low <- unlist(conf_int_boot[i, j])[1]

    # Extract the current lower bound of the confidence interval
    upp <- unlist(conf_int_boot[i, j])[2]

    # Evaluate whether the threshold is contained on the confidence intervals or not
    if((t >= low & t <= upp)){
      # Do not place an edge
      adj_mat_boot[i, j] <- 0
    } else {
      # Place an edge
      adj_mat_boot[i, j] <- 1
    }

  }

}

# Update the main diagonal of the adjacency matrix with 0 to avoid nodes connected to itself
diag(adj_mat_boot) <- 0
toc()
```

```
## 5825.61 sec elapsed
```

```
# Plot graph
G <- graph_from_adjacency_matrix(adj_mat_boot, mode = "undirected")
# Calculate number of edges
E <- length(E(G))
# Calculate number of vertexes
V <- length(V(G))
# Plot graph
plot(G, vertex.color="dodgerblue4", main=paste("Undirected Graph with ", V," vertexes and ",
 E, " edges",sep=""), vertex.label.color="white")
```



Checking out this graph, we observe 116 edges generated from the RoI. It has about 5900 edges that turned out to be statistically significative with respect to the difference of the correlation between ASD and TD groups.

```
# Calculate density of the graph
D <- 2*E / (V*(V-1))
print(paste("Density of difference graph with Bonferroni: ",D, sep = ""))
```

```
## [1] "Density of difference graph with Bonferroni: 0.883208395802099"
```

Regarding the density of the graph, we can see that it's about 90%. It's a dense graph which reflects that there are important differences in the correlations of ASD and TD patients. The previous since statistically speaking, plenty of them overpass the threshold of 0.1 regarding the dissimilarity of the association measure chosen.

Main conclusions

To sum up the main results of our experimentation, we conclude that:

- It's recommended to use the Bonferroni correction to achieve correct conclusions regarding the association graph for both ASD and TD.
- Analyzing separately ASD and TD graph doesn't lead to finding relevant differences in their behaviour.
- Analyzing the difference graph (using Pearson correlation) leads to finding important differences in the association of fMRI scan for the RoI used in the study.

Bibliography

To deal with the homework's challenges, we looked to the next sources, to understand certain topics to solve each question.

- https://support.sas.com/resources/papers/proceedings/proceedings/sugi31/170-31.pdf (https://support.sas.com/resources/papers/proceedings/proceedings/sugi31/170-31.pdf)
- https://en.wikipedia.org/wiki/Dense_graph (https://en.wikipedia.org/wiki/Dense_graph)

We hope that the way to solve this challenge has been clear and concise. See you in one next course (perhaps)!