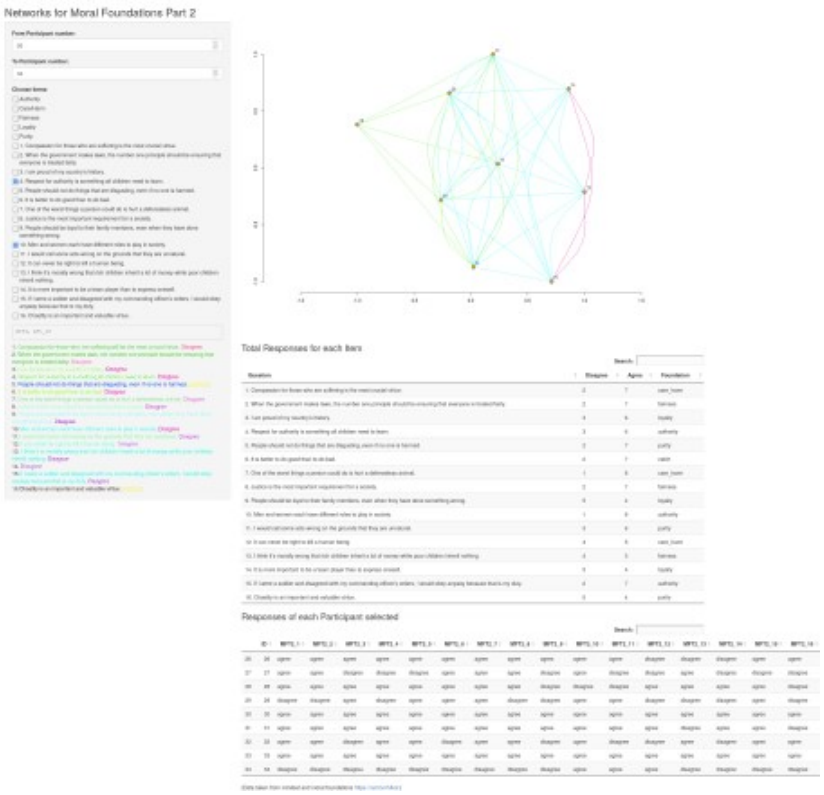
The App

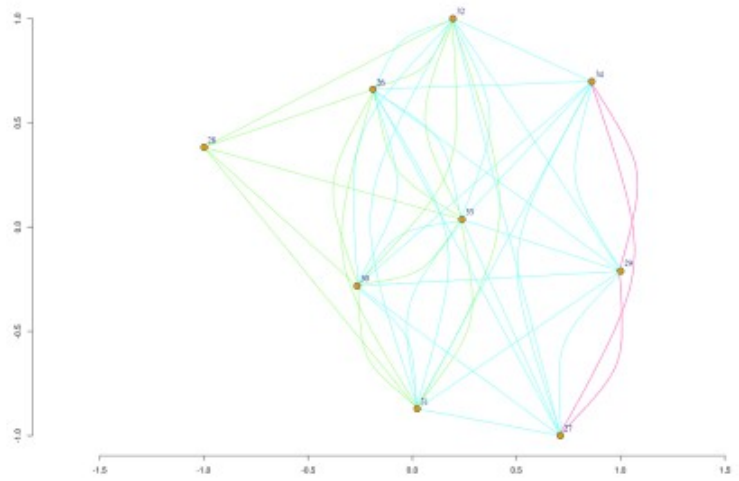
The ShinyApp is located here. It consists of a side panel with input options, and a main panel displaying the output. You can select the range of participants you wish to include, and you can select the Moral Questionnaire items you wish to look at (full list of items at the bottom of this post). The output panel displays a graph tracking the network of agreement between participants, and two tables. The first table lists the total responses for each item, and the second table displays the responses on each item for each participant. The image below displays the full ShinyApp looking at participants 26-34 on items 4 and 10 (apologies for the size).



Moral Networks ShinyApp full page view

Interpreting the output

The output contains a lot of information, but the most interesting for the current purposes is the graph. Each node (dot) represents a participant, and the edges (lines between nodes) represent agreement between participants on a particular item. The edges are colour coded according to the question and also according to the response to the question (due to the number of items, the colours are similar in some cases, so the tables may be useful in further interpreting the graph). In the graph below (and above) we can see that participants 27, 29, and 34 agree with each other on item 4; they all disagree with the statement “Respect for authority is something all children need to learn”. We can also see that participant 28 is somewhat isolated, further ivestigation reveals that they are the only participant selected who disagrees with item 10 “Men and women each have different roles to play in society”.



Moral Networks ShinyApp Graph Only

Building the App

The full code for the app is at the end of this post. There are some comments throughout (but these are far from comprehensive).

The first thing we need to do is to load the relevant libraries with:

```
library(shiny)
library(tidyverse)
library(foreign)
library(osfr)
library(igraph)
library(network)
library(igraph)
```

Next we want to download the dataset from the OSF using the `osfr` package. The code below will download a copy of the data and save it in your current working directory. I find it useful to run this first and then comment out these lines for running the app (downloading fresh each time is slow and can lead to errors).

```
osf_retrieve_file("https://osf.io/nj53t/") %>%
   osf_download(overwrite = T)
```

The UI

Once we have the libraries and data loaded we can begin to build the app. We'll start with the look and feel of it using `ui <- fluidPage()` (note that all the code in this section appears within this command).

```
ui <- fluidPage(
```

We start with a title:

```
  titlePanel("Networks for Moral Foundations Part 2"),
```

Side Panel

Then we create the sidebar for selecting our options:

```
  sidebarLayout(
      sidebarPanel(
```

Within the sidebar, we use `numericInput` to create an input options to select our range of participants. The code below will save your choices as `inNumber` and `inNumber2` for later use in the app.

```
        numericInput("inNumber", "From Participant number:", 1)
        ,
        numericInput("inNumber2", "To Participant number:", 2)
        ,
```

Having selected the participant range, we next need to select the Moral Foundations Items we wish to investigate. We do this using `checkboxGroupInput`. The code below creates an input item `MFT` that contains the selected choices, the values of the choices are in `choiceValues`. (In the server we can call `MFT` using `input$MFT`)

```
      checkboxGroupInput("MFT", "Choose Items:",
                         choiceNames =
                           c("Authority","Care/Harm","
Fairness","Loyalty","Purity",
                             "1. Compassion for those who are suffering is the
most crucial virtue."
                             ,"2. When the government makes laws, the number one
principle should be ensuring that everyone is treated fairly."
                             ,"3. I am proud of my country's history."
                             ,"4. Respect for authority is something all
children need to learn."
                             ,"5. People should not do things that are
disgusting, even if no one is harmed. "
                             ,"6. It is better to do good than to do bad."
                             ,"7. One of the worst things a person could do is
hurt a defenseless animal."
                             ,"8. Justice is the most important requirement for
a society."
                             ,"9. People should be loyal to their family
members, even when they have done something wrong.  "
                             ,"10. Men and women each have different roles to
play in society."
                             ,"11. I would call some acts wrong on the grounds
that they are unnatural."
                             ,"12. It can never be right to kill a human being."
                             ,"13. I think it's morally wrong that rich children
inherit a lot of money while poor children inherit nothing."
                             ,"14. It is more important to be a team player than
to express oneself."
                             ,"15. If I were a soldier and disagreed with my
commanding officer's orders, I would obey anyway because that is my duty."
                             ,"16. Chastity is an important and valuable
virtue."),
                         choiceValues =
                           list("mft4 mft_10 mft_15", "mft1 mft7 mft_12", "mft2
mft8 mft_13", "mft3 mft9 mft_14", "mft5 mft_11 mft_16",
                                "mft1", "mft2", "mft3", "mft4", "mft5", "mft6",
"mft7", "mft8",
                                "mft9", "mft_10", "mft_11", "mft_12", "mft_13",
"mft_14", "mft_15", "mft_16")
      )
      ,
```

Below the check box input we can include feedback confirming which items have been selected with the following:

```
      verbatimTextOutput("text")
```

And underneath this I have included the colour codes for the questions and responses using `HTML()`. I won't

include this here, refer to the full code at the end for details. This completes the side panel.

Main Panel

The output is displayed on the main panel with the following code:

```
mainPanel(
```

Display the network plot:

```
      plotOutput("netplot",width = "100%")
      ,
```

Display the table displaying total responses for each item:

```
      h3("Total Responses for each Item"),
      DT::dataTableOutput("table2")
      ,
```

And display the table displaying the responses of each participant for each item:

```
      h3("Responses of each Participant selected"),
      DT::dataTableOutput("df3_all")
    )
    )
```

(there are other bits of html that I haven't included in the above).

The Server

Now that the front end of the app has been built it's time to build the back end.

The Backend of the Shiny App

Above we have the code for setting up and modifying the look and feel of our app. Below we go through the code for making the app do what it is supposed to do. The code in full is at the bottom of this post, however I have isolated specific sections of code to describe their function. We create the back end with the following command: `server <- function(input, output) {}`; everything in the section that follow is located within the curly brackets `{}`.

```
server <- function(input, output) {
```

Before we begin working with the data, we can use the code below to generate the `text` object that we used to provide feedback on the items selected on the sidebar.

```
  output$text <- renderText({
    mft_boxes <- paste(input$MFT, collapse = ", ")
    mft_boxes
  })
```

Use `read.spss()` to create a data frame from the data file we saved above (you can identify it's name with `list.files()`). Then use `as_tibble()` to turn it into a tibble.

```
  mmfstudy1 <- read.spss("Data.sav" , to.data.frame = TRUE)
  mmfstudy1 <- as_tibble(mmfstudy1)
```

This dataframe is quite large and contains many variables we don't need. Based on the names of the variables, we can use the code below to isolate the moral foundations variables.

```
  df1 <-
    mmfstudy1 %>%
    select(matches('MFT')
```

```
            ,-matches('_DO_')
    )
```

Following this we can select the specific variables we're interested in with:

```
  df2 <- df1[17:32]
```

These variables are scale variables ranging from 0 = *Strongly disagree* to 5 = *Strongly agree*. The code below can be used to recode this to either agree or disagree:

```
  df3 <-
    df2 %>%
    mutate_all(
      funs(recode(.,
                  "0. Strongly disagree" = "disagree",
                  "1. Moderately disagree" = "disagree",
                  "2. Slightly disagree"  = "disagree",
                  "3. Slightly agree"     = "agree",
                  "4. Moderately agree"   = "agree",
                  "5. Strongly agree"   = "agree"

      ))
    )
```

Next we create a variable `ID` for participant number, and combine this with the dataframe:

```
  ID <- c(1:length(df3[[1]]))
  df3 <- cbind(ID,df3)
```

Making Networks

Going into detail on making networks is far beyond the scope of this post; (for more information on networks see this resource by Ognyanova 2016). The basic idea behind what I have done is to isolate the participants who agree with eachother on a particular item, and use the `make_full_graph()` function to create a network connecting all these participants on this item. The code below creates a custom function `net_two_a()` with arguments `x`, `a`, and `b`, where `x` is the dataframe, `a` is the location of the column containing participant ID, and `b` is the Moral Foundations Item we wish to create a network from. This function separates the participants who agree from those who disagree and creates two colour coded networks with `make_full_graph()`. These networks are then combined using `graph.disjoint.union()`.

```
  net_two_a <- function(x,a,b){

    x <- cbind.data.frame(x[a],x[b])
    colnames(x) <- c("a1","b1")

# Create a network of all participants who disagree with this item
    fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
    V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
    E(fg1)$Ju <- "Disagree"
    E(fg1)$color <- '#ff3366'           # Carmine

# Create a network of all participants who agree with this item
    fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
    V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
    E(fg7)$Ju <- "Agree"
    E(fg7)$color <- '#33cc33'           # navy
    # E(fg7)$color <- '#960018'           # Carmine
```

```
  fg <- graph.disjoint.union(fg1,fg7)
  fg
}
```

You may notice in the full code at the end of this post that I have created 16 of these functions, this is because we have 16 Moral Foundation items that we want to look at, and we want different colours for each item. As such there are 16 functions labelled `net_two_a()` to `net_two_p()`.

Selecting Participants and Items

Once we have the functions for creating networks, we need to select the participants we wish to analyse. The code below calls the numbers provided in the numeric input sidebar, and converts them into objects that we can use to generate our outputs:

```
make_a <- reactive({input$inNumber})
make_b <- reactive({input$inNumber2})
```

Similarly we can create `mft_boxes1()` which, when run, will print the MFT items that have been selected, as follows:

```
mft_boxes1 <- reactive({
  mft_boxes <- eval(parse(text = 'input$MFT'))
  mft_boxes
})
```

Generating outputs

Now that we have a way of identifying the items and the participant range selected, we can work with this to create our outputs. The most straightforward outputs are the tables. The following code creates an output `df3_all` which is displays the responses of each selected participant for each item. We use the `DT::renderDataTable({})` command to convert a dataframe into a table that can be displayed on the main panel.

```
output$df3_all <- DT::renderDataTable({
  a1 <- make_a()
  b1 <- make_b()

  a <- as.numeric(a1)
  b <- as.numeric(b1)

  df3 <- df3[a:b,]
  df3

},
options = list(bPaginate=FALSE, lengthChange = FALSE, info = FALSE))
```

Again we use the `DT::renderDataTable({})` command, however this time we create a table displaying the total responses for each item:

```
output$table2 <- DT::renderDataTable({
  a1 <- make_a()
  b1 <- make_b()

  a <- as.numeric(a1)
  b <- as.numeric(b1)

  df3 <- df3[a:b,]
```

```
    tab <-
      rbind(
        table(df3$MFT2_1),
        table(df3$MFT2_2),
        table(df3$MFT2_3),
        table(df3$MFT2_4),
        table(df3$MFT2_5),
        table(df3$MFT2_6),
        table(df3$MFT2_7),
        table(df3$MFT2_8),
        table(df3$MFT2_9),
        table(df3$MFT2_10),
        table(df3$MFT2_11),
        table(df3$MFT2_12),
        table(df3$MFT2_13),
        table(df3$MFT2_14),
        table(df3$MFT2_15),
        table(df3$MFT2_16)
      )

    tab <- cbind( c( "1. Compassion for those who are suffering is the most
crucial virtue."
                     ,"2. When the government makes laws, the number one
principle should be ensuring that everyone is treated fairly."
                     ,"3. I am proud of my country's history."
                     ,"4. Respect for authority is something all children need
to learn."
                     ,"5. People should not do things that are disgusting, even
if no one is harmed. "
                     ,"6. It is better to do good than to do bad."
                     ,"7. One of the worst things a person could do is hurt a
defenseless animal."
                     ,"8. Justice is the most important requirement for a
society."
                     ,"9. People should be loyal to their family members, even
when they have done something wrong.  "
                     ,"10. Men and women each have different roles to play in
society."
                     ,"11. I would call some acts wrong on the grounds that they
are unnatural."
                     ,"12. It can never be right to kill a human being."
                     ,"13. I think it's morally wrong that rich children inherit
a lot of money while poor children inherit nothing."
                     ,"14. It is more important to be a team player than to
express oneself."
                     ,"15. If I were a soldier and disagreed with my commanding
officer's orders, I would obey anyway because that is my duty."
                     ,"16. Chastity is an important and valuable virtue.")
                   ,tab,
                   c("care_harm","fairness","loyalty","authority","purity",
"catch","care_harm","fairness","loyalty","authority","
purity","care_harm","fairness","loyalty","authority","purity")
    )

    colnames(tab) <- c("Question","Disagree", "Agree","Foundation")
```

```
    tab
  },
  options = list(bPaginate=FALSE, lengthChange = FALSE, info = FALSE))
```

The Network

Finally, we can create the network graph (this is the point of the whole exercise). This code is repetitive (recall there are 16 items) so below I have omitted some of the repititions. The following code is all located within the `renderPlot({})` command.

First we use `renderPlot({})` to create an output object `netplot` that can be called in the `ui` to display the plot.

```
  output$netplot <- renderPlot({
```

Next we create a new dataframe that contains only the selected participants:

```
    a1 <- make_a()
    b1 <- make_b()

    a <- as.numeric(a1)
    b <- as.numeric(b1)

    df3 <- df3[a:b,]
```

Then we run our 16 custom functions to create 16 networks (one for each moral foundation item):

```
    net_1 <- net_two_a(df3,1,2)
    net_2 <- net_two_b(df3,1,3)
    net_3 <- net_two_c(df3,1,4)
    net_4 <- net_two_d(df3,1,5)
    net_5 <- net_two_e(df3,1,6)
    net_6 <- net_two_f(df3,1,7)
    net_7 <- net_two_g(df3,1,8)
    net_8 <- net_two_h(df3,1,9)
    net_9 <- net_two_i(df3,1,10)
    net_10 <- net_two_j(df3,1,11)
    net_11 <- net_two_k(df3,1,12)
    net_12 <- net_two_l(df3,1,13)
    net_13 <- net_two_m(df3,1,14)
    net_14 <- net_two_n(df3,1,15)
    net_15 <- net_two_o(df3,1,16)
    net_16 <- net_two_p(df3,1,17)
```

For each item, we can create a dataframe that includes colour attributes as a variable, and the columns are given generic names `V1`, `V2`, and `V3` (for ease of combining):

```
    df_a <- cbind.data.frame(
      get.edgelist(net_1)
      , E(net_1)$color
    )
    colnames(df_a) <- c("V1","V2","V3")
```

Repeat for each of `net_1` to `net_16` giving dataframes `df_a` to `df_p`.

Next we use `mft_boxes1()` to create an object `mft_boxes` that contains the MFT items that have been selected.

```
    mft_boxes <- paste(mft_boxes1(), sep = " " , collapse = '')
```

We then create a function `list_df_fun()` that generates a list that containing the dataframes `df_a` to

`df_p`. However, we only want to plot the items that have been selected. The code below uses `str_detect` on `mft_boxes` to identify which items should be included. First we create the function:

```
    list_df_fun <- function(){
```

Then, within the function we create an empty list called `list.df`:

```
      list.df = " "
```

The code below will add `df_a` to the list if MFT item 1 (`mft1`) has been checked:

```
      if (str_detect(mft_boxes, "mft1")) {
        list.df=c(list.df,deparse(substitute(df_a)))
      }
```

This is repeated for each item (see full code at end of post).

```
      if (str_detect(mft_boxes, "mft2")) {
        list.df=c(list.df,deparse(substitute(df_b)))
      }  # ...
```

The final line of `list_df_fun()` will print the full list:

```
      list.df
    }
```

Next we use our `list_df_fun()` function to create a list object that contains the names of the dataframes to be included in the network plot:

```
    list.df <- list_df_fun()
    list.df = list.df[2:length(list.df)] #remove first element
```

And the following code will create the object `df_combined` that contains all the networks for the items selected:

```
    expression = paste0("df_combined = rbind(",paste0( list.df, collapse = ','),
paste0(")"))

    eval(parse(text=expression))
```

Finally we can generate our plot using the code below:

```
    g2 <- df_combined
    colnames(g2) <- c("V1","V2","V3")
    g3 <- graph_from_data_frame(d=g2,directed = F)
    E(g3)$color <- E(g3)$V3

    plot(g3
         , layout_with_fr(g3)
         , vertex.label.dist=.7
         , vertex.size=3)
  },height = 800, width = 1200)
```

Conclusion

Above I have described how I went about making a ShinyApp for looking at moral networks. This post contains more complex ideas than my regular posts, I have attempted to explain the various components but it may be useful to read up on both Shiny and on Network Analysis, if there are things that aren't making sense for you.

Notes

- When it first loads it will show an error because nothing is selected;
- If there is no network to be built (e.g., 2 participants who disagree on the only item selected) it will also show up an error
- If too many participants and too many items are selected it will probably crash (I tried to create a full network of all participants and all items, and after 3 hours I gave up)
- Due to the amount of content in the App it doesn't fit too well embedded below, so it might going directly to https://cillianmacaodh.shinyapps.io/moral_networks/ to play with it.

Full Code for the App

If you copy and paste the code below into your own ShinyApp in RStudio and click run it *should* work for you.

```
library(shiny)
library(tidyverse)
library(foreign)
library(osfr)
library(igraph)
library(network)
library(igraph)


# download study 1 data
osf_retrieve_file("https://osf.io/nj53t/") %>%
  osf_download(overwrite = T)

# Define UI for application that draws a histogram
ui <- fluidPage(

  # Application title
  titlePanel("Networks for Moral Foundations Part 2"),

  # Sidebar with a slider input for number of bins
  sidebarLayout(
    sidebarPanel(
      numericInput("inNumber", "From Participant number:", 1)
      ,
      numericInput("inNumber2", "To Participant number:", 2)
      ,
      checkboxGroupInput("MFT", "Choose Items:",
                         choiceNames =
                           c("Authority","Care/Harm","
Fairness","Loyalty","Purity",
                             "1. Compassion for those who are suffering is the
most crucial virtue."
                             ,"2. When the government makes laws, the number one
principle should be ensuring that everyone is treated fairly."
                             ,"3. I am proud of my country's history."
                             ,"4. Respect for authority is something all
children need to learn."
                             ,"5. People should not do things that are
disgusting, even if no one is harmed. "
                             ,"6. It is better to do good than to do bad."
                             ,"7. One of the worst things a person could do is
hurt a defenseless animal."
                             ,"8. Justice is the most important requirement for
```

```
a society."
                            ,"9. People should be loyal to their family
members, even when they have done something wrong.  "
                            ,"10. Men and women each have different roles to
play in society."
                            ,"11. I would call some acts wrong on the grounds
that they are unnatural."
                            ,"12. It can never be right to kill a human being."
                            ,"13. I think it’s morally wrong that rich children
inherit a lot of money while poor children inherit nothing."
                            ,"14. It is more important to be a team player than
to express oneself."
                            ,"15. If I were a soldier and disagreed with my
commanding officer’s orders, I would obey anyway because that is my duty."
                            ,"16. Chastity is an important and valuable
virtue."),
                        choiceValues =
                          list("mft4 mft_10 mft_15", "mft1 mft7 mft_12", "mft2
mft8 mft_13", "mft3 mft9 mft_14", "mft5 mft_11 mft_16",
                               "mft1", "mft2", "mft3", "mft4", "mft5", "mft6",
"mft7", "mft8",
                               "mft9", "mft_10", "mft_11", "mft_12", "mft_13",
"mft_14", "mft_15", "mft_16")
      )
      ,
      verbatimTextOutput("text")
      ,
      HTML("

           1. Compassion for those who are suffering will be the most crucial
virtue.
           Disagree


           2. When the government makes laws, the number one principle should be
ensuring that everyone is treated fairly.
           Disagree


           3.  I am be proud of my country’s history.
           Disagree


           4.  Respect for authority is something all children need to learn.
           Disagree


           5. People should not do things that are disgusting, even if no one is
harmed.
           Disagree


           6.  It is better to do good than to do bad.
           Disagree


           7.  One of the worst things a person could do is hurt a defenseless
```

```
animal.
           Disagree


           8.  Justice is the most important requirement for a society.
           Disagree


           9. People are expected to be loyal to their family members, even when
they have done something wrong.
           Disagree


           10.Men and women each have different roles to play in society.
           Disagree


           11.I would call some acts wrong on the grounds that they are
unnatural.
           Disagree


           12.It can never be right to kill a human being.
           Disagree


           13. I think it is morally wrong that rich children inherit a lot of
money while poor children inherit nothing.
           Disagree


           14.
           Disagree


           15.If I were a soldier and disagreed with my commanding officer's
orders, I would obey anyway because that is my duty.
           Disagree


           16.Chastity is an important and valuable virtue.
           Disagree

           ")
      ),

    # Show a plot of the generated distribution
    mainPanel(
      #,
      plotOutput("netplot",width = "100%")
      ,
      br(),br(),br(),br(),br(),br(),br(),br(),br(),br(),br(),br(),
br(),br(),br(),br(),br(),br(),br(),br(),br(),br()
      ,
      h3("Total Responses for each Item"),
      DT::dataTableOutput("table2")
      ,
```

```
      h3("Responses of each Participant selected"),
      DT::dataTableOutput("df3_all")
      ,
      br()
      ,
      HTML('(Data taken from mindset and moral foundations https://osf.io/nh4ck/)')
    )
    )
)

server <- function(input, output) {

  # read spss file and save as tibble
  mmfstudy1 <- read.spss("Data.sav" , to.data.frame = TRUE)
  mmfstudy1 <- as_tibble(mmfstudy1)

  # select Moral Foundations Variables

  output$text <- renderText({
    mft_boxes <- paste(input$MFT, collapse = ", ")
    mft_boxes
  })

  observe({
    x <- input$MFT

    # Can use character(0) to remove all choices
    if (is.null(x))
      x <- character(0)
  })



  df1 <-
    mmfstudy1 %>%
    select(matches('MFT')
           ,-matches('_DO_')
    )


  # select agree/disagree variables
  df2 <- df1[17:32]


  levels(df2$MFT2_1)

  # bin the variables into agree vs disagree

  df3 <-
    df2 %>%
    mutate_all(
      funs(recode(.,
                  "0. Strongly disagree" = "disagree",
                  "1. Moderately disagree" = "disagree",
                  "2. Slightly disagree"  = "disagree",
                  "3. Slightly agree"     = "agree",
                  "4. Moderately agree"   = "agree",
```

```
                "5. Strongly agree"   = "agree"

    ))
  )

ID <- c(1:length(df3[[1]]))

df3 <- cbind(ID,df3)

#### make all net functions ####
net_two_a <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")

  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#ff3366'           # Carmine


  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#33cc33'           # navy
  # E(fg7)$color <- '#960018'           # Carmine

  fg <- graph.disjoint.union(fg1,fg7)
  fg
}


net_two_b <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")

  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#ff6699'           # Fandango
  # E(fg1)$color <- '#00AB6B'           # Jade


  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#33cc00'           # Sky
  # E(fg7)$color <- '#00AB6B'           # Jade

  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_c <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
```

```
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#ff00cc'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#99ff66'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg

}

net_two_d <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#ff0099'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#66ff33'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_e <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")



  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
```

```
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#fff000'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#0033ff'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_f <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#cc0099'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#99ff00'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_g <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#cc66cc'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
```

```
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#66cc99'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_h <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#cc33cc'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#99ffcc'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_i <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#9900cc'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#99ffff'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
```

```
  fg
}

net_two_j <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#cc00cc'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#33ffff'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_k <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#cc33ff'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#66ff99'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_l <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")
```

```
  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#cc66ff'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#66ffcc'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_m <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#9933cc'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#00ffcc'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_n <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")

  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#9966cc'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)
```

```
  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#00cccc'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg

}

net_two_o <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#9933ff'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#00ffff'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_p <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  # E(fg1)$color <- '#9966ff'
  E(fg1)$color <- '#ffff33'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
```

```
  # E(fg7)$color <- '#33ffcc'           # Prussian
  E(fg7)$color <- '#330000'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}

net_two_q <- function(x,a,b){

  x <- cbind.data.frame(x[a],x[b])
  colnames(x) <- c("a1","b1")


  fg1 <- make_full_graph(length(x$a1[which(x$b1=="disagree")]))
  V(fg1)$name <- c(x$a1[which(x$b1=="disagree")])
  E(fg1)$Ju <- "Disagree"
  E(fg1)$color <- '#993366'           # Mahogany (from reds)
  # E(fg1)$color <- '#FFBF00'           # Amber (from shades of yellow)



  fg7 <- make_full_graph(length(x$a1[which(x$b1=="agree")]))
  V(fg7)$name <- c(x$a1[which(x$b1=="agree")])
  E(fg7)$Ju <- "Agree"
  E(fg7)$color <- '#009999'           # Prussian
  # E(fg7)$color <- '#FFBF00'           # Amber (from shades of yellow)


  fg <- graph.disjoint.union(fg1,fg7)
  fg
}


#### select data range ####

df4 <- df3

make_a <- reactive({input$inNumber})
make_b <- reactive({input$inNumber2})

output$df3_all <- DT::renderDataTable({
  a1 <- make_a()
  b1 <- make_b()

  a <- as.numeric(a1)
  b <- as.numeric(b1)

  df3 <- df3[a:b,]
  df3

},
options = list(bPaginate=FALSE, lengthChange = FALSE, info = FALSE))


mft_boxes1 <- reactive({
```

```
  mft_boxes <- eval(parse(text = 'input$MFT'))
  mft_boxes
})

output$netplot <- renderPlot({
  a1 <- make_a()
  b1 <- make_b()

  a <- as.numeric(a1)
  b <- as.numeric(b1)

  mft_boxes <- as.character(unlist(mft_boxes1()))
  mft_boxes <- paste(mft_boxes)

  df3 <- df3[a:b,]

  net_two_a(df3,1,2)

  net_1 <- net_two_a(df3,1,2)
  net_2 <- net_two_b(df3,1,3)
  net_3 <- net_two_c(df3,1,4)
  net_4 <- net_two_d(df3,1,5)
  net_5 <- net_two_e(df3,1,6)
  net_6 <- net_two_f(df3,1,7)
  net_7 <- net_two_g(df3,1,8)
  net_8 <- net_two_h(df3,1,9)
  net_9 <- net_two_i(df3,1,10)
  net_10 <- net_two_j(df3,1,11)
  net_11 <- net_two_k(df3,1,12)
  net_12 <- net_two_l(df3,1,13)
  net_13 <- net_two_m(df3,1,14)
  net_14 <- net_two_n(df3,1,15)
  net_15 <- net_two_o(df3,1,16)
  net_16 <- net_two_p(df3,1,17)


  net_combined <- rbind.data.frame(
    cbind(
      get.edgelist(net_1)
      , E(net_1)$color
    )
    ,
    cbind(
      get.edgelist(net_2)
      , E(net_2)$color
    )
    ,
    cbind(
      get.edgelist(net_3)
      , E(net_3)$color
    )
    ,
    cbind(
      get.edgelist(net_4)
      , E(net_4)$color
    )
    ,
```

```
cbind(
  get.edgelist(net_5)
  , E(net_5)$color
)
# ,
# cbind(
#   get.edgelist(net_6)
#   , E(net_6)$color
# )
,
cbind(
  get.edgelist(net_7)
  , E(net_7)$color
)
,
cbind(
  get.edgelist(net_8)
  , E(net_8)$color
)
,
cbind(
  get.edgelist(net_9)
  , E(net_9)$color
)
,
cbind(
  get.edgelist(net_10)
  , E(net_10)$color
)
,
cbind(
  get.edgelist(net_11)
  , E(net_11)$color
)
,
cbind(
  get.edgelist(net_12)
  , E(net_12)$color
)
,
cbind(
  get.edgelist(net_13)
  , E(net_13)$color
)
,
cbind(
  get.edgelist(net_14)
  , E(net_14)$color
)
,
cbind(
  get.edgelist(net_15)
  , E(net_15)$color
)
,
cbind(
  get.edgelist(net_16)
```

```
    , E(net_16)$color
  )
)


##### create a mini dataframe for each item ####

df_a <- cbind.data.frame(
  get.edgelist(net_1)
  , E(net_1)$color
)
colnames(df_a) <- c("V1","V2","V3")

df_b <- cbind.data.frame(
  get.edgelist(net_2)
  , E(net_2)$color
)
colnames(df_b) <- c("V1","V2","V3")

df_c <- cbind.data.frame(
  get.edgelist(net_3)
  , E(net_3)$color
)
colnames(df_c) <- c("V1","V2","V3")

df_d <- cbind.data.frame(
  get.edgelist(net_4)
  , E(net_4)$color
)
colnames(df_d) <- c("V1","V2","V3")

df_e <- cbind.data.frame(
  get.edgelist(net_5)
  , E(net_5)$color
)
colnames(df_e) <- c("V1","V2","V3")

df_f <- cbind.data.frame(
  get.edgelist(net_6)
  , E(net_6)$color
)
colnames(df_f) <- c("V1","V2","V3")

df_g <- cbind.data.frame(
  get.edgelist(net_7)
  , E(net_7)$color
)
colnames(df_g) <- c("V1","V2","V3")

df_h <- cbind.data.frame(
  get.edgelist(net_8)
  , E(net_8)$color
)
colnames(df_h) <- c("V1","V2","V3")

df_i <- cbind.data.frame(
  get.edgelist(net_9)
```

```
  , E(net_9)$color
)
colnames(df_i) <- c("V1","V2","V3")

df_j <- cbind.data.frame(
  get.edgelist(net_10)
  , E(net_10)$color
)
colnames(df_j) <- c("V1","V2","V3")

df_k <- cbind.data.frame(
  get.edgelist(net_11)
  , E(net_11)$color
)
colnames(df_k) <- c("V1","V2","V3")

df_l <- cbind.data.frame(
  get.edgelist(net_12)
  , E(net_12)$color
)
colnames(df_l) <- c("V1","V2","V3")

df_m <- cbind.data.frame(
  get.edgelist(net_13)
  , E(net_13)$color
)
colnames(df_m) <- c("V1","V2","V3")

df_n <- cbind.data.frame(
  get.edgelist(net_14)
  , E(net_14)$color
)
colnames(df_n) <- c("V1","V2","V3")

df_o <- cbind.data.frame(
  get.edgelist(net_15)
  , E(net_15)$color
)
colnames(df_o) <- c("V1","V2","V3")

df_p <- cbind.data.frame(
  get.edgelist(net_16)
  , E(net_16)$color
)
colnames(df_p) <- c("V1","V2","V3")

##### include based on checkbox #####


mft_boxes <- paste(mft_boxes1(), sep = " " , collapse = '')

#    mft_boxes <- output$text

list_df_fun <- function(){

  list.df = " "
```

```
if (str_detect(mft_boxes, "mft1")) {
  list.df=c(list.df,deparse(substitute(df_a)))
}

if (str_detect(mft_boxes, "mft2")) {
  list.df=c(list.df,deparse(substitute(df_b)))
}

if (str_detect(mft_boxes, "mft3")) {
  list.df=c(list.df,deparse(substitute(df_c)))
}

if (str_detect(mft_boxes, "mft4")) {
  list.df=c(list.df,deparse(substitute(df_d)))
}

if (str_detect(mft_boxes, "mft5")) {
  list.df=c(list.df,deparse(substitute(df_e)))
}

if (str_detect(mft_boxes, "mft6")) {
  list.df=c(list.df,deparse(substitute(df_f)))
}

if (str_detect(mft_boxes, "mft7")) {
  list.df=c(list.df,deparse(substitute(df_g)))
}

if (str_detect(mft_boxes, "mft8")) {
  list.df=c(list.df,deparse(substitute(df_h)))
}

if (str_detect(mft_boxes, "mft9")) {
  list.df=c(list.df,deparse(substitute(df_i)))
}

if (str_detect(mft_boxes, "mft_10")) {
  list.df=c(list.df,deparse(substitute(df_j)))
}

if (str_detect(mft_boxes, "mft_11")) {
  list.df=c(list.df,deparse(substitute(df_k)))
}

if (str_detect(mft_boxes, "mft_12")) {
  list.df=c(list.df,deparse(substitute(df_l)))
}

if (str_detect(mft_boxes, "mft_13")) {
  list.df=c(list.df,deparse(substitute(df_m)))
}

if (str_detect(mft_boxes, "mft_14")) {
  list.df=c(list.df,deparse(substitute(df_n)))
}
```

```
      if (str_detect(mft_boxes, "mft_15")) {
        list.df=c(list.df,deparse(substitute(df_o)))
      }

      if (str_detect(mft_boxes, "mft_16")) {
        list.df=c(list.df,deparse(substitute(df_p)))
      }
      list.df
    }

    list.df <- list_df_fun()

    list.df = list.df[2:length(list.df)] #remove first element
    expression = paste0("df_combined = rbind(",paste0( list.df, collapse = ','),
paste0(")"))

    eval(parse(text=expression))


    # rename the combined object (just for recycling code)
    #g2 <- net_combined
    g2 <- df_combined
    colnames(g2) <- c("V1","V2","V3")
    g3 <- graph_from_data_frame(d=g2,directed = F)
    #V(g3)$color <- V(net_incs)$color
    E(g3)$color <- E(g3)$V3
    # g4 <- simplify(g3, remove.multiple = T, remove.loops = T)


    plot(g3
         #, vertex.label=NA
         , layout_with_fr(g3)

         #       , layout = layout.auto()# = layout_with_fr()
         , vertex.label.dist=.7
         , vertex.size=3)
  },height = 800, width = 1200)

  output$table2 <- DT::renderDataTable({
    a1 <- make_a()
    b1 <- make_b()

    a <- as.numeric(a1)
    b <- as.numeric(b1)

    df3 <- df3[a:b,]

    tab <-
      rbind(
        table(df3$MFT2_1),
        table(df3$MFT2_2),
        table(df3$MFT2_3),
        table(df3$MFT2_4),
        table(df3$MFT2_5),
        table(df3$MFT2_6),
        table(df3$MFT2_7),
        table(df3$MFT2_8),
```

```
        table(df3$MFT2_9),
        table(df3$MFT2_10),
        table(df3$MFT2_11),
        table(df3$MFT2_12),
        table(df3$MFT2_13),
        table(df3$MFT2_14),
        table(df3$MFT2_15),
        table(df3$MFT2_16)
      )

    tab <- cbind( c( "1. Compassion for those who are suffering is the most
crucial virtue."
                     ,"2. When the government makes laws, the number one
principle should be ensuring that everyone is treated fairly."
                     ,"3. I am proud of my country's history."
                     ,"4. Respect for authority is something all children need
to learn."
                     ,"5. People should not do things that are disgusting, even
if no one is harmed. "
                     ,"6. It is better to do good than to do bad."
                     ,"7. One of the worst things a person could do is hurt a
defenseless animal."
                     ,"8. Justice is the most important requirement for a
society."
                     ,"9. People should be loyal to their family members, even
when they have done something wrong.  "
                     ,"10. Men and women each have different roles to play in
society."
                     ,"11. I would call some acts wrong on the grounds that they
are unnatural."
                     ,"12. It can never be right to kill a human being."
                     ,"13. I think it's morally wrong that rich children inherit
a lot of money while poor children inherit nothing."
                     ,"14. It is more important to be a team player than to
express oneself."
                     ,"15. If I were a soldier and disagreed with my commanding
officer's orders, I would obey anyway because that is my duty."
                     ,"16. Chastity is an important and valuable virtue.")
                   ,tab,
                   c("care_harm","fairness","loyalty","authority","purity",
"catch","care_harm","fairness","loyalty","authority","
purity","care_harm","fairness","loyalty","authority","purity")
    )

    colnames(tab) <- c("Question","Disagree", "Agree","Foundation")


    tab
  },
  options = list(bPaginate=FALSE, lengthChange = FALSE, info = FALSE))

}

# Run the application
shinyApp(ui = ui, server = server)
```

MFT Items

1. Compassion for those who are suffering is the most crucial virtue.
2. When the government makes laws, the number one principle should be ensuring that everyone is treated fairly.
3. I am proud of my country's history.
4. Respect for authority is something all children need to learn.
5. People should not do things that are disgusting, even if no one is harmed.
6. It is better to do good than to do bad.
7. One of the worst things a person could do is hurt a defenseless animal.
8. Justice is the most important requirement for a society.
9. People should be loyal to their family members, even when they have done something wrong.
10. Men and women each have different roles to play in society.
11. I would call some acts wrong on the grounds that they are unnatural.
12. It can never be right to kill a human being.
13. I think it's morally wrong that rich children inherit a lot of money while poor children inherit nothing.
14. It is more important to be a team player than to express oneself.
15. If I were a soldier and disagreed with my commanding officer's orders, I would obey anyway because that is my duty.
16. Chastity is an important and valuable virtue.