Assignment: Predicting Employee Churn with Network Analytics

Emma Kruis

2020-01-18

Instructions

This assignment reviews the *Network Analytics* content. You will use the *network_analytics.Rmd* file I reviewed as part of the lectures for this week to complete this assignment. You will *copy and paste* relevant code from that file and update it to answer the questions in this assignment. You will respond to questions in each section after executing relevant code to answer a question. You will submit this assignment to its *Submissions* folder on *D2L*. You will submit *two* files:

1. this completed *R Markdown* script, and
2. as a first preference, a *PDF* (if you already installed `TinyTeX` properly), as a second preference, a *Microsfot Word* (if your computer has *Microsoft Word*) document, or, as a third preference, an *HTML* (if you did *not* install `TinyTeX` properly and your computer does *not* have *Microsoft Word*) file to *D2L*.

To start:

First, create a folder on your computer to save all relevant files for this course. If you did not do so already, you will want to create a folder named *mgt_592* that contains all of the materials for this course.

Second, inside of *mgt_592*, you will create a folder to host assignments. You can name that folder *assignments*.

Third, inside of *assignments*, you will create folders for each assignment. You can name the folder for this first assignment: *network_analytics*.

Fourth, create three additional folders in *network_analytics* named *scripts*, *data*, and *plots*. Store this script in the *scripts* folder and the data for this assignment in the *data* folder.

Fifth, go to the *File* menu in *RStudio*, select *New Project...*, choose *Existing Directory*, go to your *~/mgt_592/assignments/network_analytics* folder to select it as the top-level directory for this **R Project**.

Global Settings

The first code chunk sets the global settings for the remaining code chunks in the document. Do *not* change anything in this code chunk.

Load Packages

In this code chunk, we load the following packages:

1. **here**,
2. **tidyverse**,
3. **tidymodels**,

4. **corrr**,
5. **igraph**,
6. **tidygraph**, and
7. **ggraph**.

Make sure you installed these packages when you reviewed the analytical lecture.

We will use functions from these packages to examine the data. Do *not* change anything in this code chunk.

```
### load libraries for use in current working session
## here for project work flow
library(here)

## tidyverse for data manipulation and plotting
## loads eight different libraries simultaneously
library(tidyverse)

## tidymodels for modeling
library(tidymodels)

## corrr for correlation matrices
library(corrr)

## igraph for analyzing networks
library(igraph)

## tidygraph for graph data tables
library(tidygraph)

## ggraph for plotting networks
library(ggraph)
```

Task 1: Create Network

For this task, you will create a small collaboration network.

Task 1.1

Create a *data frame* named **team_collab** that represents an *edges* data table of a network. Make the following *21* connections:

1. *Pavle* is connected to *Luka*, *Sanja*, and *Nikola*;
2. *Ana* is connected to *Milan* and *Vera*;
3. *Sanja* is connected to *Jelena* and *Olga*;
4. *Lazar* is connected to *Milena*, *Vedran*, and *Sanja*;
5. *Vedran* is connected to *Olga* and *Vera*;
6. *Nikola* is connected to *Olga*, *Lazar*, and *Milan*;
7. *Jelena* is connected to *Milena*, *Nikola*, and *Vedran*;
8. *Milena* is connected to *Pavle*, *Vedran*, and *Sanja*.

Make sure the senders are listed in this order. Convert the *data table* to a *undirected* network named **team_net** using **graph_from_data_frame()**.

Create a new vertex **status** attribute in **team_net** for the nodes by setting **V(team_net)$status** to **c("R", "L", "R", "R", "U", "U", "R", "R", "L", "L", "L", "L")**. Copy the **status** attribute into a **color** vertex attribute. Then, convert an **R** into a **blue** color, an **L** into a **red** color, and an **U** into a **gray** color.

Create a vertex position matrix named **ver_pos** using **cbind()**. Set the first column equal to **c(5, 25, 13, 1, 25, 5, 8, 20, 1, 15, 25, 13)**. Set the second column equal to **c(20, 1, 23, 13, 16, 3, 10, 20, 25, 2, 10, 8)**.

Plot the updated **team_net** using **plot()**. Set the edge labels to **NA** and color to **black**. Set the layout to **ver_pos**. Set the vertex labels to **V(team_net)**$name{*}{*}andlabelcolorsto{*}{*}white{*}{*}.Setthevertexcolorsto{*} {*}V(team_net)$**color** and size to **35**.

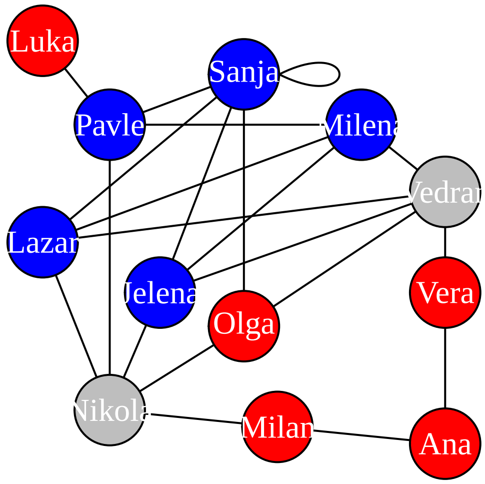
Examine the plot.

Questions 1.1: Answer these questions: (1) Did *Milena* leave? (2) How many connections does *Vedran* have to individuals who *remain* versus *left* the team? (3) How many individuals connect to *five* other teammates?

Responses 1.1: *(1) No, Milena remained; (2) 3 connections to those who remain and 2 connections to those who left the team for Vedran; (3) Vedran, Nikola, Sanja, and Milena.*

```r
team_collab <- data.frame(
## senders
  from = c(
    # first sender
    "Pavle", "Pavle", "Pavle",
    # second sender
    "Ana", "Ana",
    # third sender
    "Sanja", "Sanja",
    # fourth sender
    "Lazar", "Lazar", "Lazar",
    # fifth sender
    "Vedran", "Vedran",
    # sixth sender
    "Nikola", "Nikola", "Nikola",
    # seventh sender
    "Jelena", "Jelena", "Jelena",
    # eighth sender
    "Milena", "Milena", "Sanja"
),

to = c(
  #Receivers of first sender
  "Luka", "Sanja", "Nikola",
  #Receivers of second sender
  "Milan", "Vera",
  #Recievers of third sender
  "Jelena", "Olga",
  #Recievers of fourth sender
  "Milena", "Vedran", "Sanja",
  #Recievers of fifth sender
  "Olga", "Vera",
  #Recivers of sixth sender
  "Olga", "Lazar", "Milan",
  #Recievers of seventh sender
```

```
  "Milena", "Nikola", "Vedran",
  #Recivers of eigth sender
  "Pavle", "Vedran", "Sanja"
  )
)

#converting data table
team_net <- graph_from_data_frame(
  team_collab,
  directed = FALSE
  )


#setting up attributes
V(team_net)$status <-
  c("R", "L", "R", "R", "U", "U", "R", "R", "L", "L", "L", "L")

V(team_net)$color <- V(team_net)$status

V(team_net)$color <- str_replace(V(team_net)$color, "R", "blue")

V(team_net)$color <- str_replace(V(team_net)$color, "L", "red")

V(team_net)$color <- str_replace(V(team_net)$color, "U", "gray")

#vertex positions
ver_pos <- cbind(
  #x-coordinate
  c(5, 25, 13, 1, 25, 5, 8, 20, 1, 15, 25, 13),
  #y-coordinate
  c(20, 1, 23, 13, 16, 3, 10, 20, 25, 2, 10, 8)
)

#plotting the network
plot(
  team_net,
  edge.label = NA,
  edge.color = "black",
  layout = ver_pos,
  vertex.label = V(team_net)$name,
  vertex.label.color = "white",
  vertex.color = V(team_net)$color,
  vertex.size = 35
)
```



Task 2: Relational Neighbor Classifier

For this task, you will compute the (probabilistic) relational neighbor classifier.

Task 2.1

Compute the probability of individuals remaining in the team based on their neighbors.

First, convert **team_net** to a *long data frame* named **team_att** containing the node attributes. Second, create **node_sender_summ** from **team_att** by: grouping by **from_name**, *summarizing* the number of individuals with remain, left, and unknown **to_status**, and *renaming* the variable **from_name** to simply **name**. Third, create **node_receiver_summ** from **team_att** by: grouping by **to_name**, *summarizing* the number of individuals with remain, left, and unknown **from_status**, and *renaming* the variable **to_name** to simply **name**. Finally, create **node_summ** by row binding the two data tables **node_sender_summ** and **node_receiver_summ**, group by **name**, sum across all variables, and compute via **mutate()** (1) the total number of connected collaborators who remain or left the team and (2) the probability of connected collaborators remaining with respect to the total number who remain or left the team.

Print **node_summ**.

Questions 2.1: Answer these questions: (1) How many total collaborators with a known *status* are there for *Lazar*? (2) What is the probability of *Vedran* remaining based on the relational neighbor classifier?

Responses 2.1: *(1) 2 (2) 0.6* .

```
team_att <- as_long_data_frame(team_net)

#summarizing the sender notes
node_sender_summ <- team_att %>%
  group_by(from_name) %>%
  summarize(
    collab_remain = sum(to_status == "R"),
    collab_left = sum(to_status == "L"),
    collab_unknown = sum(to_status == "U")
    ) %>%
  rename(name = from_name)
```

```r
#summarize receiver nodes
node_receiver_summ <- team_att %>%
group_by(to_name) %>%
  summarize(
    collab_remain = sum(from_status == "R"),
    collab_left = sum(from_status == "L"),
    collab_unknown = sum(from_status == "U") ) %>%
  rename(name = to_name)


#relational neighbor classifier
node_summ <- node_sender_summ %>%
  bind_rows(node_receiver_summ) %>%
  group_by(name) %>%
  summarize(
    across(
      .cols = everything(),
      .fns = sum
      ),
    .groups = "drop" ) %>%
  mutate(
    tot_known = collab_remain + collab_left,
    remain_prob_known = collab_remain / tot_known
    )


node_summ
```

```
## # A tibble: 12 x 6
##    name   collab_remain collab_left collab_unknown tot_known remain_prob_known
##    <chr>          <int>       <int>          <int>     <int>             <dbl>
##  1 Ana                0           2              0         2             0
##  2 Jelena             2           0              2         2             1
##  3 Lazar              2           0              2         2             1
##  4 Luka               1           0              0         1             1
##  5 Milan              0           1              1         1             0
##  6 Milena             3           0              1         3             1
##  7 Nikola             3           2              0         5             0.6
##  8 Olga               1           0              2         1             1
##  9 Pavle              2           1              1         3             0.667
## 10 Sanja              5           1              0         6             0.833
## 11 Vedran             3           2              0         5             0.6
## 12 Vera               0           1              1         1             0
```

Task 2.2

You will now compute a probabilistic neighbor classifier.

Now, consider an altered collaboration network by applying **plot()** to **team_net** with the layout set to **ver_pos**, the vertex color set to **gray**, and the vertex size set to **35**.

Next, extract the adjacency matrix of **team_net** and save it as **team_adj**.
Set the seed of your computer to **1736**. Create an initial probability vector of individuals remaining on the

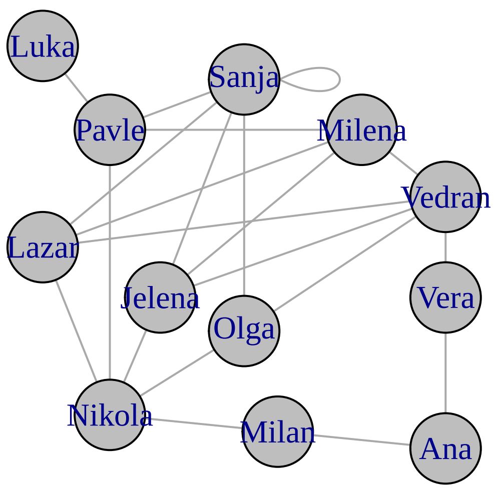
team represented by **team_net** named **remain_prob** by using **runif(12, 0, 1)**. Set the names of the elements of **remain_prob** using **names()** and **V(team_net)$name**. Print these initial probabilities.

Then, create a *for* loop to update the initial probabilities *twice*. Update the probabilities by matrix multiplying **team_adj** with **remain_prob** and dividing by the *degrees* of each node. This result should be added to **remain_prob**, and, then divided by **2**. Make sure to overwrite **remain_prob** across the loops. Print the updated probabilities.

Questions 2.2: Answer these questions: (1) Did *Pavle's* initial probability *increase* or *decrease* after the updating? (2) After updating, which individual is most likely to leave (i.e., has the smallest probability to remain)?

Responses 2.2: *(1) Increase (2) Milan.*

```
#plotting network
plot(
  team_net,
  layout = ver_pos,
  vertex.color = "gray",
  vertex.size = 35
  )
```




```
#extracting adjacency matrices
team_adj <- as_adjacency_matrix(team_net)

set.seed(1736)

remain_prob <- runif(12, 0, 1)

names(remain_prob) <- V(team_net)$name

remain_prob
```

```
##      Pavle        Ana      Sanja      Lazar     Vedran     Nikola     Jelena
## 0.15717691 0.01577420 0.25434745 0.77482946 0.60075228 0.49799156 0.52668285
##     Milena       Luka      Milan       Vera       Olga
## 0.65136751 0.62764547 0.05691282 0.67815928 0.27503613
```

```r
#for loop to iterate on intial probabilities
for (ndx in 1:2) {
  remain_prob <-
    (team_adj %*% remain_prob /
     degree(team_net) +
     remain_prob) / 2
}

remain_prob
```

```
## 12 x 1 Matrix of class "dgeMatrix"
##             [,1]
## Pavle  0.3783076
## Ana    0.2583548
## Sanja  0.3247781
## Lazar  0.5558615
## Vedran 0.5546148
## Nikola 0.4144607
## Jelena 0.4938248
## Milena 0.5509772
## Luka   0.3624593
## Milan  0.2333776
## Vera   0.4422653
## Olga   0.4001649
```

Task 3: Homophily

For this task, you will compute homophily statistics for the team network.

Task 3.1

Use **V(team_net)$status**[**c(5, 6)**] to set the *status* of the previously unknown individuals to **R**. Do the same for the *color* attribute of those two individuals; set the color attribute to **blue**.

Create an edge attribute named **label** by using this vector **c("RL", "RR", "RR", "LL", "LL", "RR", "RL", "RR", "RR", "RR", "RL", "RL", "RL", "RR", "RL", "RR", "RR", "RR", "RR", "RR", "RR")**. Break-up the vector into three lines of seven for reading clarity in your code. Copy the **label** edge attribute to a **color** edge attribute. Then, convert an **RR** into a **blue** color, an **LL** into a **red** color, and an **RL** into a **purple** color.

Plot the updated **team_net** using **plot()**. Set the edge labels to **NA** and color to **black** and width to **3**. Set the layout to **ver_pos**. Set the vertex label colors to **white** and size to **35**.

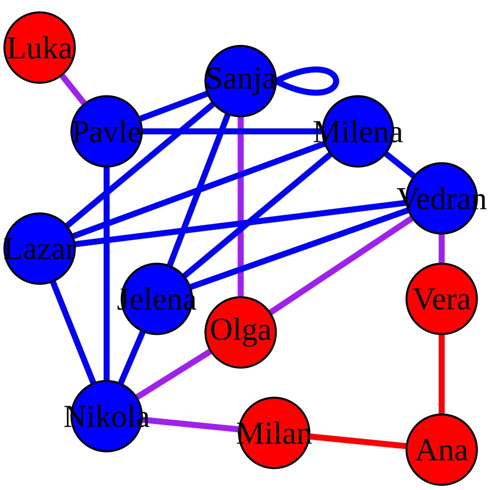
Examine the plot.

Questions 3.1: Answer these questions: (1) How many *red* edges are there in the team network? (2) How many *purple* edges are there in the team network?

Responses 3.1: *(1) 2 (2) 5* .

```r
#updating node attributes
V(team_net)$status[c(5,6)] <- "R"

V(team_net)$color[c(5,6)] <- "blue"
```

```r
#adding edge lables
E(team_net)$label <-
  c(
    "RL", "RR", "RR", "LL", "LL", "RR", "RL",
    "RR", "RR", "RR", "RL", "RL", "RL", "RR",
    "RL", "RR", "RR", "RR", "RR", "RR", "RR"
    )

#se color for remain
E(team_net)$color <- E(team_net)$label

#alter for remain
E(team_net)$color <- str_replace(E(team_net)$color, "RR", "blue")
#alter for left
E(team_net)$color <- str_replace(E(team_net)$color, "LL", "red")
#alter for unknown
E(team_net)$color <- str_replace(E(team_net)$color, "RL", "purple")

#plot updated collaboration network
plot(
  team_net,
  layout = ver_pos,
  vertex.size = 35,
  vertex.label.color = "black",
  edge.label = NA,
  edge.width = 3
)
```



Task 3.2

Compute four values for the network:

1. compute the *connectedness* of the network using **edge_density()**;
2. compute the *dyadicity* of the *remain* nodes in the network;
3. compute the *dyadicity* of the *left* nodes in the network;
4. compute the *heterophilicity* of the network;

Examine the results.

Questions 3.2: Answer these questions: (1) What proportion of total possible edges exist in the network? (2) Is there evidence for *dyadicity* of the *remain* nodes? (3) Is there evidence for *dyadicity* of the *left* nodes? (4) Is there evidence for *heterophilicity* of nodes in the network?

Responses 3.2: *(1) 31.8 percent of possible edges in this network (2) yes, 1.945578 (3) No, 0.6285714 (4) No evidence of homophilcity because below zero.*

```
#1, network conncetedness
edge_density(team_net)
```

```
## [1] 0.3181818
```

```
#2, dyadicity of remain nodes
sum(E(team_net)$label == "RR") /
  (edge_density(team_net) *
     (sum(V(team_net)$status == "R") * (sum(V(team_net)$status == "R") - 1) / 2))
```

```
## [1] 1.945578
```

```
#3, dyadicity of left nodes
sum(E(team_net)$label == "LL") /
  (edge_density(team_net) *
     (sum(V(team_net)$status == "L") * (sum(V(team_net)$status == "L") - 1) / 2))
```

```
## [1] 0.6285714
```

```
#4, heterophilicity of network
sum(E(team_net)$label == "RL") /
  (edge_density(team_net) *
     sum(V(team_net)$status == "R") *
     sum(V(team_net)$status == "L"))
```

```
## [1] 0.5387755
```

Task 4: Network Node Measures

For this task, you will compute the node measures for the team network.

Task 4.1

Compute the following node measures:

1. degree,
2. second-order neighborhood size excluding the focal node,
3. number of triangles,
4. local transitivity,
5. betweenness centrality,
6. closeness centrality,
7. eigenvector centrality,

8. PageRank,
9. the number of *remain* nodes in the first-order neighborhood, and
10. the number of *left* nodes in the first-order neighborhood.

Examine the results.

Question 4.1: Answer these questions: (1) What is the *betweenness centrality* for *Milan*? (2) What is the *second-order neighborhood size* for *Pavle*? (3) Is there evidence for *eigenvector centrality* for *Sanja*? (4) What is the *PageRank* for *Lazar*?

Response 4.1: *(1) 5.583333 (2) 9 (3) yes, she has one of the highest 0.42652666 (4) 0.09027393.*

```
#1, degree of nodes
degree(team_net)
```

```
##  Pavle    Ana  Sanja  Lazar Vedran Nikola Jelena Milena   Luka  Milan   Vera
##      4      2      6      4      5      5      4      4      1      2      2
##   Olga
##      3
```

```
#2, second-order neighborhood size excluding the focal node
neighborhood.size(team_net,order = 2) - 1
```

```
##  [1]  9  4  8  9  9 10  9  9  4  7  7  9
```

```
#3, number of triangles
count_triangles(team_net)
```

```
##  [1] 0 0 0 1 2 0 1 2 0 0 0 0
```

```
#4, local transitivity
transitivity(team_net, type = "local")
```

```
##  [1] 0.0000000 0.0000000 0.0000000 0.1666667 0.2000000 0.0000000 0.1666667
##  [8] 0.3333333       NaN 0.0000000 0.0000000 0.0000000
```

```
#5 betweenness centrality
betweenness(team_net)
```

```
##     Pavle       Ana     Sanja     Lazar    Vedran    Nikola    Jelena    Milena
## 11.642857  1.500000  3.250000  3.285714 12.595238 15.071429  3.285714  5.583333
##      Luka     Milan      Vera      Olga
##  0.000000  5.321429  4.178571  2.285714
```

```
#6, closeness centrality
closeness(team_net)
```

```
##      Pavle        Ana      Sanja      Lazar     Vedran     Nikola     Jelena
## 0.05000000 0.03448276 0.04545455 0.05000000 0.05263158 0.05555556 0.05000000
##     Milena       Luka      Milan       Vera       Olga
## 0.05000000 0.03333333 0.04166667 0.04000000 0.04761905
```

```r
#7, eigenvector centrality
eigen_centrality(team_net, scale = FALSE)
```

```
## $vector
##      Pavle        Ana      Sanja      Lazar     Vedran     Nikola     Jelena
## 0.29284498 0.04893809 0.42652666 0.36337986 0.35771490 0.34516917 0.36337986
##     Milena       Luka      Milan       Vera       Olga
## 0.34049205 0.07239524 0.09742865 0.10053012 0.27920562
##
## $value
## [1] 4.045086
##
## $options
## $options$bmat
## [1] "I"
##
## $options$n
## [1] 12
##
## $options$which
## [1] "LA"
##
## $options$nev
## [1] 1
##
## $options$tol
## [1] 0
##
## $options$ncv
## [1] 0
##
## $options$ldv
## [1] 0
##
## $options$ishift
## [1] 1
##
## $options$maxiter
## [1] 1000
##
## $options$nb
## [1] 1
##
## $options$mode
## [1] 1
##
## $options$start
## [1] 1
##
## $options$sigma
## [1] 0
##
## $options$sigmai
```

```
## [1] 0
##
## $options$info
## [1] 0
##
## $options$iter
## [1] 8
##
## $options$nconv
## [1] 1
##
## $options$numop
## [1] 27
##
## $options$numopb
## [1] 0
##
## $options$numreo
## [1] 20
```

```
#8, PageRank
page_rank(team_net, damping = 0.9, personalized = NULL)
```

```
## $vector
##      Pavle        Ana      Sanja      Lazar     Vedran     Nikola     Jelena
## 0.09657398 0.05817760 0.13085085 0.09027393 0.11531213 0.11656862 0.09027393
##     Milena       Luka      Milan       Vera       Olga
## 0.09144193 0.03006248 0.05549561 0.05526944 0.06969950
##
## $value
## [1] 1
##
## $options
## NULL
```

```
#the remaining nodes
team_adj %*%
  as.numeric(V(team_net)$status == "R")
```

```
## 12 x 1 Matrix of class "dgeMatrix"
##        [,1]
## Pavle     3
## Ana       0
## Sanja     4
## Lazar     4
## Vedran    3
## Nikola    3
## Jelena    4
## Milena    4
## Luka      1
## Milan     1
## Vera      1
## Olga      3
```

```r
#the neighbors who left
team_adj %*%
  as.numeric(V(team_net)$status == "L")
```

```
## 12 x 1 Matrix of class "dgeMatrix"
##        [,1]
## Pavle     1
## Ana       2
## Sanja     1
## Lazar     0
## Vedran    2
## Nikola    2
## Jelena    0
## Milena    0
## Luka      0
## Milan     1
## Vera      1
## Olga      0
```

Task 5: Import Data

For this task, you will import data representing an employee network.

Task 5.1

Use **load()** and **here()** to import **employee_net.rdata** from the **data** folder in the project directory. Preview the **nodes** and **edges** data tables via **glimpse()**. You will work with the complete network for this assignment.

Questions 5.1: Answer these questions: (1) What is the *churn* value of the *first* employee? (2) Which connection is listed *first*?

Responses 5.1: *(1) No (2) employee one is connected to employee 3.*

```r
load(
  here("data", "employee_net.rdata") )

glimpse(nodes)
```

```
## Rows: 1,000
## Columns: 2
## $ id    <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "~
## $ churn <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No, No, No, ~
```

```r
glimpse(edges)
```

```
## Rows: 196,966
## Columns: 2
## $ from <chr> "1", "1", "3", "3", "3", "4", "1", "3", "4", "7", "1", "3", "5", ~
## $ to   <chr> "3", "4", "4", "5", "6", "6", "7", "7", "7", "8", "9", "9", "9", ~
```

Task 6: Table Graph

For this task, you will create a *table graph* and visualize the employee network.

Task 6.1

Create a *table graph* named **employee_tg** using the **nodes** and **edges** data tables. Set the *node key* to **id** and create an *undirected* graph.

Next, update the *node* data table in **employee_tg** by computing **edge_id**, which is equivalent to the *row numbers*. Continue with a chained command to update the *edge* data table in **employee_tg** by computing **churn_type**. The **churn_type** variable should consist of three categories named **cc**, **ss**, and **cs** to indicate whether the nodes for an edge both *left*, both *remain*, or a mixture of the two, respectively.

Print the updated **employee_tg**.

Questions 6.1: Answer these questions: (1) What is the **churn_type** of the *first five edges*? (2) What is the **churn** of the *first three nodes*?

Responses 6.1: *(1) ss (2) No .*

```
#creating a table graph
employee_tg <- tbl_graph(
  nodes = nodes,
  edges = edges,
  directed = FALSE,
  node_key = "id"
)

#create data table
employee_tg <- employee_tg %>%
  ## activate nodes
activate(nodes) %>%
  mutate(
    edge_id = row_number()
    ) %>%
  activate(edges) %>%
  mutate(
      churn_type = case_when(
        .N()$churn[from] == "Yes" & .N()$churn[to] == "Yes" ~ "cc",
        .N()$churn[from] == "No" & .N()$churn[to] == "No" ~ "ss",
        .N()$churn[from] == "Yes" & .N()$churn[to] == "No" ~ "cs",
        .N()$churn[from] == "No" & .N()$churn[to] == "Yes" ~ "cs"
        )
      )

## print
employee_tg
```

```
## # A tbl_graph: 1000 nodes and 196966 edges
## #
## # An undirected simple graph with 1 component
## #
## # Edge Data: 196,966 x 3 (active)
##    from    to churn_type
```

```
##   <int> <int> <chr>
## 1     1     3 ss
## 2     1     4 ss
## 3     3     4 ss
## 4     3     5 ss
## 5     3     6 ss
## 6     4     6 ss
## # ... with 196,960 more rows
## #
## # Node Data: 1,000 x 3
##   id    churn edge_id
##   <chr> <fct>   <int>
## 1 1     No          1
## 2 2     No          2
## 3 3     No          3
## # ... with 997 more rows
```

Task 6.2

Set the random seed of your computer to **1736**. Create a new data table named **employee_nodes_samp** from **employee_tg**. The **id** variable should be removed and the **edge_id** variable should be converted to a *character* variable. The **employee_nodes_samp** data table should consist of **70** randomly sampled nodes from **employee_tg** using **slice_sample()**.

Next, create a new data table named **employee_edges_samp** from **employee_tg**. Convert **from** and **to** to *character* variables. Filter the rows by **from** and **to** being in **employee_nodes_samp$edge_id**.

Then, create a new *table graph* named **employee_tg_samp** from **employee_nodes_samp** and **employee_edges_samp**. Set the *node key* to **edge_id** and create an *undirected* graph.

Print **employee_tg_samp**.

Questions 6.2: Answer these questions: (1) How many *rows* are there in the *edge* data table of **employee_tg_samp**? (2) What is the **churn_type** of the *first three edges*? (3) What is the **churn** of the *third node*?

Responses 6.2: *(1) 969 rows (2) ss (3) Yes.*

```
set.seed(1736)

#Sample nodes
employee_nodes_samp <-employee_tg %>%
  activate(nodes) %>%
  as_tibble() %>%
  mutate(
    id = NULL,
    edge_id = as.character(edge_id)
    ) %>%
  slice_sample(n = 70)

#Sample edges
employee_edges_samp <- employee_tg %>%
  activate(edges) %>%
  as_tibble() %>%
  mutate(
    from = as.character(from),
```

```
    to = as.character(to)
    ) %>%
  filter(
    from %in% employee_nodes_samp$edge_id,
    to %in% employee_nodes_samp$edge_id
    )


#create table
employee_tg_samp <- tbl_graph(
  nodes = employee_nodes_samp,
  edges = employee_edges_samp,
  directed = FALSE,
  node_key = "edge_id"
  )


employee_tg_samp
```

```
## # A tbl_graph: 70 nodes and 969 edges
## #
## # An undirected simple graph with 1 component
## #
## # Node Data: 70 x 2 (active)
##   churn edge_id
##   <fct> <chr>
## 1 No    61
## 2 No    10
## 3 Yes   285
## 4 No    604
## 5 No    459
## 6 Yes   893
## # ... with 64 more rows
## #
## # Edge Data: 969 x 3
##    from    to churn_type
##   <int> <int> <chr>
## 1    32    37 ss
## 2    25    37 ss
## 3     2    32 ss
## # ... with 966 more rows
```

Task 6.3

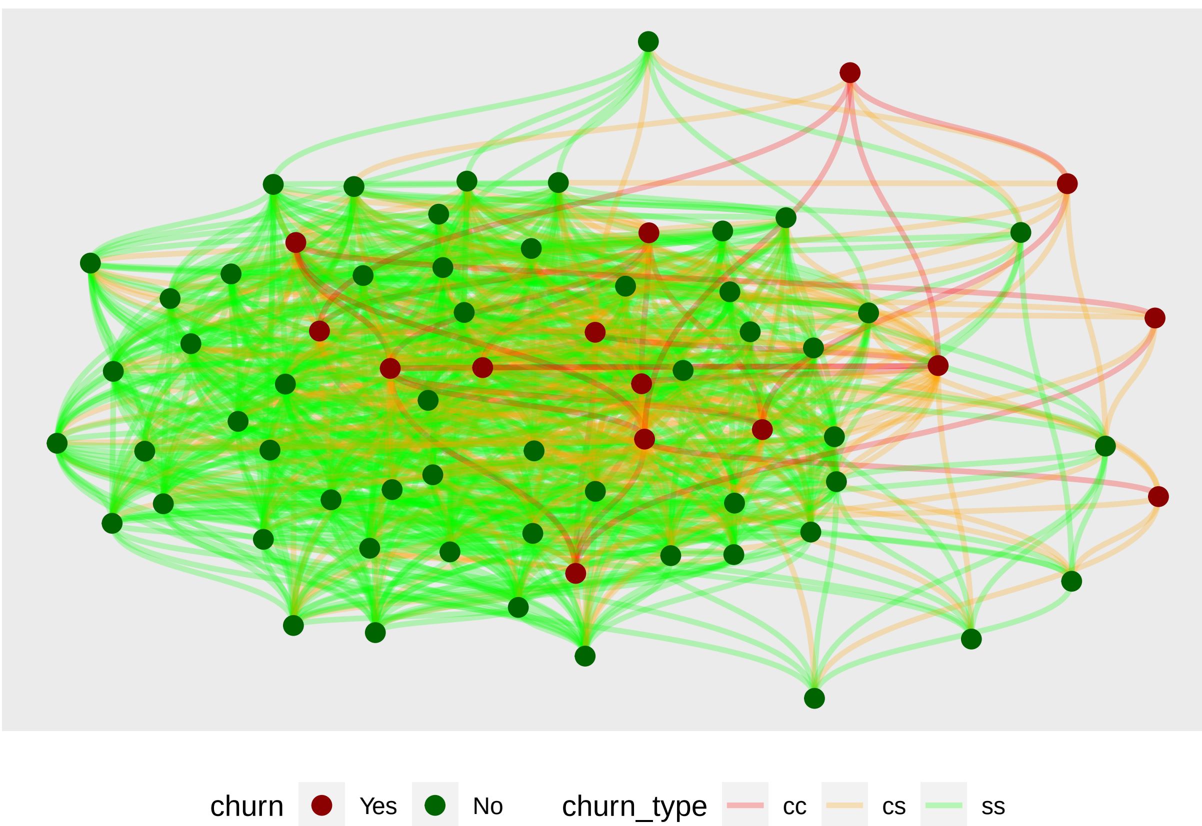
Create a plot named **employee_net_samp_plot** using **ggraph()** and **employee_tg_samp** to view a representative sample of the network. Set the *layout* to **kk**. Choose **geom_node_point()** as the node geometry and set **color** to **churn** and **size** to **3**. Choose **geom_edge_diagonal()** as the edge geometry and set **color** to **churn_type**, **alpha** to **0.25**, and **width** to **1**. Add **scale_color_manual()** to set the node colors to **darkred** and **darkgreen**. Add **scale_edge_color_manual()** to set the *churn type* colors to **red**, **orange**, and **green**. Place the legend at the bottom. Print the plot.

Create another plot named **employee_net_samp_facet_plot** using **ggraph()** and **employee_tg_samp** to view a representative sample of the network. Set the *layout* to **kk**. Choose **geom_node_point()** as the node geometry and set **color** to **churn** and **size** to **3**. Choose **geom_edge_diagonal()** as the edge geometry and set **color** to **churn_type**, **alpha** to **0.25**, and **width** to **1**. Add **scale_color_manual()** to set the node colors to **darkred** and **darkgreen**. Add **scale_edge_color_manual()** to set the *churn type* colors to **red**, **orange**, and **green**. Add **facet_edges()** to facet by **churn_type**. Place the legend at the bottom. Print the plot.

Questions 6.3: Answer these questions: (1) Which *edge color* dominates in **employee_net_samp_plot**? (2) What do you learn from **employee_net_samp_facet_plot**?

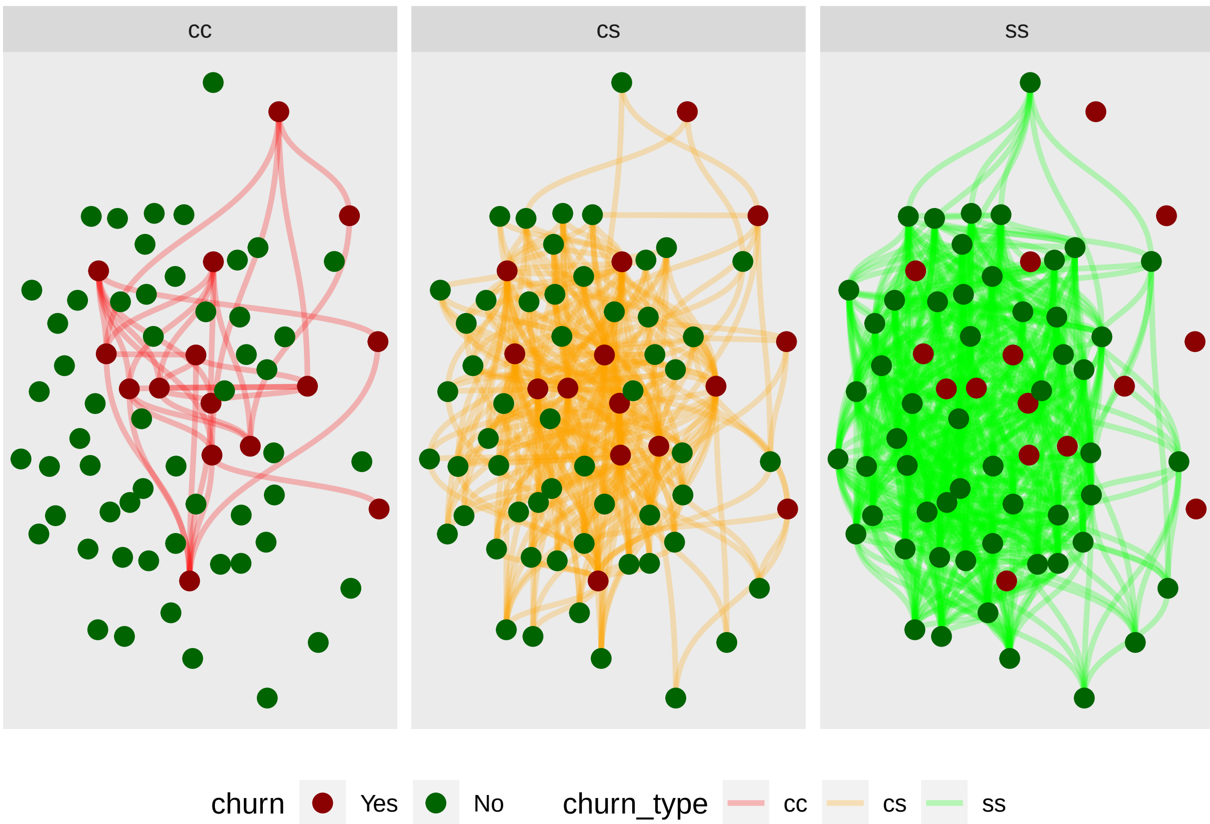
Responses 6.3: *(1) The green (2) the churners are relatively connected to themselves, and the e,mplyees who stay are in general connected to each other. however overall there are some connections between those who are churners and those who stay. Overall this is a lot of homophilicity. .*

```r
#create employee net samp plot
employee_net_samp_plot <- ggraph(
  employee_tg_samp,
  layout = "kk" )+
  geom_edge_diagonal(aes(color = churn_type), alpha = 0.25, width = 1) +
  geom_node_point(aes(color = churn), size = 3) +
  scale_color_manual(values = c("darkred", "darkgreen")) +
  scale_edge_color_manual(values = c("red", "orange", "green")) +
  theme(legend.position = "bottom")

employee_net_samp_plot
```




```r
#create employee net samp facet plot
employee_net_samp_facet_plot <- ggraph(
  employee_tg_samp,
```

```
  layout = "kk"
  )+
  geom_edge_diagonal(aes(color = churn_type), alpha = 0.25, width = 1) +
  geom_node_point(aes(color = churn), size = 3) +
  scale_color_manual(values = c("darkred", "darkgreen")) +
  scale_edge_color_manual(values = c("red", "orange", "green")) +
  facet_edges(~ churn_type) +
  theme(legend.position = "bottom")

employee_net_samp_facet_plot
```



Task 7: Calculate Measures

For this task, you will calculate network measures on the employee network.

Task 7.1

First, update **employee_tg** by computing the following network measures in the appropriate data tables using the relevant functions:

1. first-order neighborhood size,
2. second-order neighborhood size,
3. average degree of neighborhood,
4. number of triangles,
5. local transitivity,

6. betweenness centrality,
7. closeness centrality,
8. eigenvector centrality,
9. PageRank centrality,
10. edge betweenness centrality,
11. average edge PageRank centrality,

Print the updated version of **employee_tg** to confirm calculations.

Second, updated **employee_tg** by computing the following node neighborhood measures using **map_local_dbl()** with a *user-defined function* appropriately:

1. number of churners in first-order neighborhood,
2. number of non-churners in first-order neighborhood,
3. churn probability of first-order neighborhood,
4. number of churners in second-order neighborhood,
5. number of non-churners in second-order neighborhood, and
6. churn probability of second-order neighborhood.

Remain patient with the calculations. Print *wide* the updated version of **employee_tg** to examine calculations.

Questions 7.1: Answer these questions: (1) What is the *average neighborhood degree* for the *first* listed node? (2) What is the *local transitivity* for the *second* listed node? (3) What is the *number of non-churners* in the *first-order neighborhood* for the *third* listed node? (4) What is the *churn probability* in the *second-order neighborhood* for the *fourth* listed node?

Responses 7.1: *(1) 429 (2) 0.294 (3) 51 (4) 0.104.*

```
##calculating attributes
employee_tg <- employee_tg %>%
  activate(nodes) %>%
  mutate(
    degree_1 = local_size(order = 1, mindist = 1),
    degree_2 = local_size(order = 2, mindist = 2),
    neigh_avg_deg = local_ave_degree(),
    n_triangle = local_triangles(),
    loc_trans = local_transitivity(),
    between = centrality_betweenness(directed = FALSE, normalized = TRUE),
    closeness = centrality_closeness(normalized = TRUE),
    eigen = centrality_eigen(),
    page_rank = centrality_pagerank(directed = FALSE)
    ) %>%
  activate(edges) %>%
  mutate(
    edge_between = centrality_edge_betweenness(directed = FALSE),
    mean_page_rank = (.N()$page_rank[from] + .N()$page_rank[to]) / 2)


employee_tg
```

```
## # A tbl_graph: 1000 nodes and 196966 edges
## #
## # An undirected simple graph with 1 component
```

```
## #
## # Edge Data: 196,966 x 5 (active)
##    from    to churn_type edge_between mean_page_rank
##   <int> <int> <chr>             <dbl>          <dbl>
## 1     1     3 ss                 3.77        0.00112
## 2     1     4 ss                 3.51        0.00111
## 3     3     4 ss                 3.66        0.00110
## 4     3     5 ss                 3.69        0.00111
## 5     3     6 ss                 3.81        0.00108
## 6     4     6 ss                 3.59        0.00106
## # ... with 196,960 more rows
## #
## # Node Data: 1,000 x 12
##   id    churn edge_id degree_1 degree_2 neigh_avg_deg n_triangle loc_trans
##   <chr> <fct>   <int>    <dbl>    <dbl>         <dbl>      <dbl>     <dbl>
## 1 1     No          1      460      539          429.      49561     0.469
## 2 2     No          2      136      863          334.       2702     0.294
## 3 3     No          3      446      553          425.      45944     0.463
## # ... with 997 more rows, and 4 more variables: between <dbl>, closeness <dbl>,
## #   eigen <dbl>, page_rank <dbl>
```

```
##computing adjacency-based attributes
employee_tg <- employee_tg %>%
  activate(nodes) %>%
  mutate(
    churn_neigh_1 = map_local_dbl(
      order = 1,
      mindist = 1,
      .f = function(neighborhood, ...) {
        sum(
          as_tibble(neighborhood, active = "nodes")$churn == "Yes"
          )
        }
      ),
    non_churn_neigh_1 = map_local_dbl(
      order = 1,
      mindist = 1,
      .f = function(neighborhood, ...) {
        sum(
          as_tibble(neighborhood, active = "nodes")$churn == "No"
          )
        }
      ),
    neigh_churn_prob_1 = churn_neigh_1 / (churn_neigh_1 + non_churn_neigh_1),
    churn_neigh_2 = map_local_dbl(
      order = 2,
      mindist = 2,
      .f = function(neighborhood, ...) {
        sum(
          as_tibble(neighborhood, active = "nodes")$churn == "Yes"
          )
        }
      ),
    non_churn_neigh_2 = map_local_dbl(
```

```r
      order = 2,
      mindist = 2,
      .f = function(neighborhood, ...) {
        sum(
          as_tibble(neighborhood, active = "nodes")$churn == "No"
          )
        }
      ),
    neigh_churn_prob_2 = churn_neigh_2 / (churn_neigh_2 + non_churn_neigh_2)
    )
employee_tg %>%
  activate(nodes) %>%
  print(width = Inf)
```

```
## # A tbl_graph: 1000 nodes and 196966 edges
## #
## # An undirected simple graph with 1 component
## #
## # Node Data: 1,000 x 18 (active)
##   id    churn edge_id degree_1 degree_2 neigh_avg_deg n_triangle loc_trans
##   <chr> <fct>   <int>    <dbl>    <dbl>         <dbl>      <dbl>     <dbl>
## 1 1     No          1      460      539          429.      49561     0.469
## 2 2     No          2      136      863          334.       2702     0.294
## 3 3     No          3      446      553          425.      45944     0.463
## 4 4     No          4      433      566          430       44224     0.473
## 5 5     No          5      448      551          433.      48063     0.480
## 6 6     No          6      420      579          430.      41544     0.472
##    between closeness eigen page_rank churn_neigh_1 non_churn_neigh_1
##      <dbl>     <dbl> <dbl>     <dbl>         <int>             <int>
## 1 0.000771     0.650 0.940  0.00114             59               401
## 2 0.000250     0.537 0.210  0.000482            28               108
## 3 0.000785     0.644 0.904  0.00111             51               395
## 4 0.000659     0.638 0.888  0.00108             45               388
## 5 0.000647     0.645 0.925  0.00111             58               390
## 6 0.000608     0.633 0.862  0.00105             48               372
##   neigh_churn_prob_1 churn_neigh_2 non_churn_neigh_2 neigh_churn_prob_2
##                <dbl>         <int>             <int>              <dbl>
## 1              0.128            90               449              0.167
## 2              0.206           121               742              0.140
## 3              0.114            98               455              0.177
## 4              0.104           104               462              0.184
## 5              0.129            91               460              0.165
## 6              0.114           101               478              0.174
## # ... with 994 more rows
## #
## # Edge Data: 196,966 x 5
##    from    to churn_type edge_between mean_page_rank
##   <int> <int> <chr>             <dbl>          <dbl>
## 1     1     3 ss                 3.77        0.00112
## 2     1     4 ss                 3.51        0.00111
## 3     3     4 ss                 3.66        0.00110
## # ... with 196,963 more rows
```

Task 7.2

Extract the *nodes* data table from **employee_tg** and save it as **employee_nodes**. Extract the *edges* data table from **employee_tg** and save it as **employee_edges**.

Create **employee_edges_agg** from **employee_edges** by:

1. pivoting longer by **from** and **to** and setting the names to **edge_ndx** and values to **edge_id**,
2. grouping by **edge_id**, and
3. summarizing **edge_between** and **mean_page_rank** with the **mean** function.

Update **employee_nodes** by applying **left_join()** with **employee_edges_agg** and joining by **edge_id**. Replace any missing values on the network measures with *zero* values. Print *wide* the updated version of **employee_nodes** to examine calculations.

Questions 7.2: Answer these questions: (1) What is the **edge_between** for the *first* listed node? (2) What is the **mean_page_rank** for the *second* listed node?

Responses 7.2: *(1) 3.77 (2) 0.00111.*

```r
#extracting nodes as tibbles
employee_nodes <- employee_tg %>%
  activate(nodes) %>%
  as_tibble()


employee_edges <- employee_tg %>%
  activate(edges) %>%
  as_tibble()


##aggregating edges as tibble
employee_edges_agg <- employee_edges %>%
  pivot_longer(
    cols = c(from, to),
    names_to = "edge_ndx",
    values_to = "edge_id"
    ) %>%
  group_by(edge_id) %>%
  summarize(
    across(
      .cols = c(edge_between, mean_page_rank),
      .fns = mean
      )
    )


##join nodes with edges
employee_nodes <- employee_nodes %>%
  left_join(employee_edges_agg, by = "edge_id") %>%
  mutate(
    across(
      .cols = degree_1:mean_page_rank,
      .fns = ~ replace_na(.x, 0)
      )
```

```r
    )

#calling data
employee_nodes %>%
  print(width = Inf)
```

```
## # A tibble: 1,000 x 20
##    id    churn edge_id degree_1 degree_2 neigh_avg_deg n_triangle loc_trans
##    <chr> <fct>   <int>    <dbl>    <dbl>         <dbl>      <dbl>     <dbl>
##  1 1     No          1      460      539          429.      49561     0.469
##  2 2     No          2      136      863          334.       2702     0.294
##  3 3     No          3      446      553          425.      45944     0.463
##  4 4     No          4      433      566          430       44224     0.473
##  5 5     No          5      448      551          433.      48063     0.480
##  6 6     No          6      420      579          430.      41544     0.472
##  7 7     No          7      422      577          429.      41787     0.470
##  8 8     No          8      428      571          431.      43373     0.475
##  9 9     No          9      418      581          431.      41583     0.477
## 10 10    No         10      435      564          428.      44157     0.468
##     between closeness eigen page_rank churn_neigh_1 non_churn_neigh_1
##       <dbl>     <dbl> <dbl>     <dbl>         <dbl>             <dbl>
##  1 0.000771     0.650 0.940  0.00114             59               401
##  2 0.000250     0.537 0.210  0.000482            28               108
##  3 0.000785     0.644 0.904  0.00111             51               395
##  4 0.000659     0.638 0.888  0.00108             45               388
##  5 0.000647     0.645 0.925  0.00111             58               390
##  6 0.000608     0.633 0.862  0.00105             48               372
##  7 0.000635     0.634 0.863  0.00106             53               369
##  8 0.000628     0.636 0.880  0.00107             57               371
##  9 0.000584     0.632 0.860  0.00104             48               370
## 10 0.000710     0.639 0.887  0.00109             51               384
##    neigh_churn_prob_1 churn_neigh_2 non_churn_neigh_2 neigh_churn_prob_2
##                 <dbl>         <dbl>             <dbl>              <dbl>
##  1              0.128            90               449              0.167
##  2              0.206           121               742              0.140
##  3              0.114            98               455              0.177
##  4              0.104           104               462              0.184
##  5              0.129            91               460              0.165
##  6              0.114           101               478              0.174
##  7              0.126            96               481              0.166
##  8              0.133            92               479              0.161
##  9              0.115           101               480              0.174
## 10              0.117            98               466              0.174
##    edge_between mean_page_rank
##           <dbl>          <dbl>
##  1         3.84       0.00110
##  2         9.18       0.000681
##  3         3.99       0.00109
##  4         3.83       0.00108
##  5         3.67       0.00109
##  6         3.82       0.00106
##  7         3.87       0.00106
##  8         3.80       0.00107
```

```
##  9          3.78       0.00106
## 10          3.92       0.00108
## # ... with 990 more rows
```

Task 8: Predict Churn

For this task, you will fit logistic regression models to predict employee churn.

Task 8.1

Set the random seed of your computer to **1736**. Create a new object named **employee_nodes_split** using **initial_split()** applied to **employee_nodes**. Split the data into *70%* training and *30%* testing.

Plot the correlations among the node network measures on the training data. Appropriately apply **training()**, **select()**, **correlate()**, **rearrange()**, **shave()**, and **rplot()** to **employee_nodes_split**. Alter the *theme* of the plot to make the x-axis labels angled at *45* degrees.

Examine the plot.

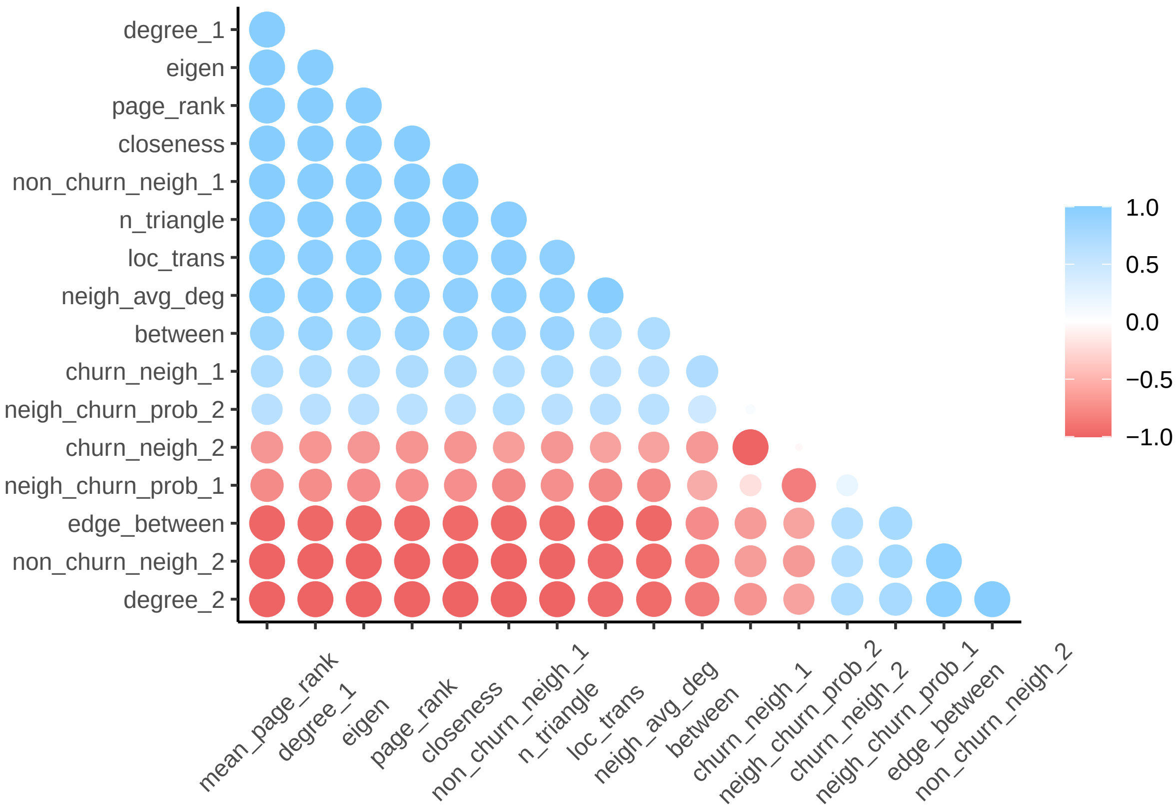
Questions 8.1: Answer these questions: (1) Is the correlation between **degree_2** and **mean_page_rank** *positive* or *negative*? (2) Is the correlation between **churn_neigh_1** and **eigen** or **closeness** and **eigen** stronger?

Responses 8.1: *(1) Negative (2) Closeness and eigen is stronger.*

```r
## training and testing split
set.seed(1736)

## create split
employee_nodes_split <- initial_split(
  employee_nodes,
  prop = 0.7 )

##examine correlations
employee_nodes_split %>%
  training() %>%
  select(degree_1:mean_page_rank) %>%
  correlate() %>%
  rearrange() %>%
  shave() %>%
  rplot() +
  theme(
    axis.text.x = element_text(angle = 45, vjust = 0.5)
    )
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
## Don't know how to automatically pick scale for object of type noquote. Defaulting to continuous.
```



Task 8.2

First, create a modeling *workflow* named **glm_wrkflw** for *logistic regression* using the **glm** engine.

Second, fit a *logistic regression* model on the *training* data of **employee_nodes_split** where **churn** is predicted by four network node measures: (1) **degree_2**, (2) **eigen**, (3) **page_rank**, (4) and **n_triangle**. Save the model as **glm_1**.

Third, apply **predict()** to **glm_1** to calculate *probabilities* of **churn** in the *testing* data of **employee_nodes_split**. Bind the predictions with the *testing* data. Save the result as **glm_1_probs**.

Fourth, apply **roc_curve()** to **glm_1_probs**. Save the result as **glm_1_roc**. Plot **glm_1_roc** using **autoplot()**.

Fifth, calculate the area under the receiver-operator characteristic (ROC) curve for **glm_1_probs** using **roc_auc()**.

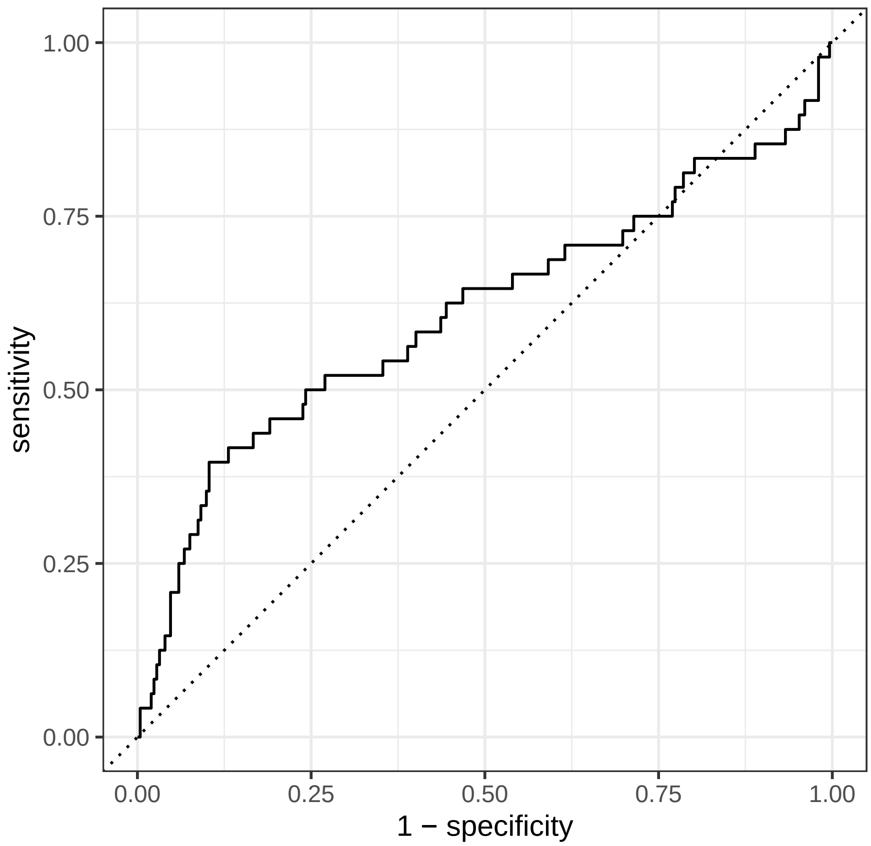
Examine the results.

Questions 8.2: What is the area under the ROC curve for this model?

Responses 8.2: *The area under the ROC curve is 0.610* .

```
##logistic regression workflow
glm_wrkflw <- workflow() %>%
  add_model(
    logistic_reg() %>%
      set_engine("glm")
    )

##first logistic regression model
```

```r
glm_1 <- glm_wrkflw %>%
  add_formula(
    churn ~ degree_2 + eigen +
      page_rank + n_triangle
    ) %>%
  fit(training(employee_nodes_split))

##evaluate predictions
glm_1_probs <- predict(
  glm_1,
  testing(employee_nodes_split),
  type = "prob"
  ) %>%
  bind_cols(testing(employee_nodes_split))


##ROC curve
glm_1_roc <- glm_1_probs %>%
  roc_curve(churn, .pred_Yes)


autoplot(glm_1_roc)
```




```r
##area under ROC curve
glm_1_probs %>%
  roc_auc(churn, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.610
```

Task 8.3

Fit a second *logistic regression* model using **glm_wrkflw** on the *training* data of **employee_nodes_split** where **churn** is predicted by two neighborhood node measures: (1) **neigh_churn_prob_1** and (2) **neigh_churn_prob_2**. Save the model as **glm_2**.

Then, apply **predict()** to **glm_2** to calculate *probabilities* of **churn** in the *testing* data of **employee_nodes_split**. Bind the predictions with the *testing* data. Save the result as **glm_2_probs**.

Then, apply **roc_curve()** to **glm_2_probs**. Save the result as **glm_2_roc**. Apply **autoplot()** to plot **glm_1_roc** and add a **geom_path()** layer using **glm_2_roc**.

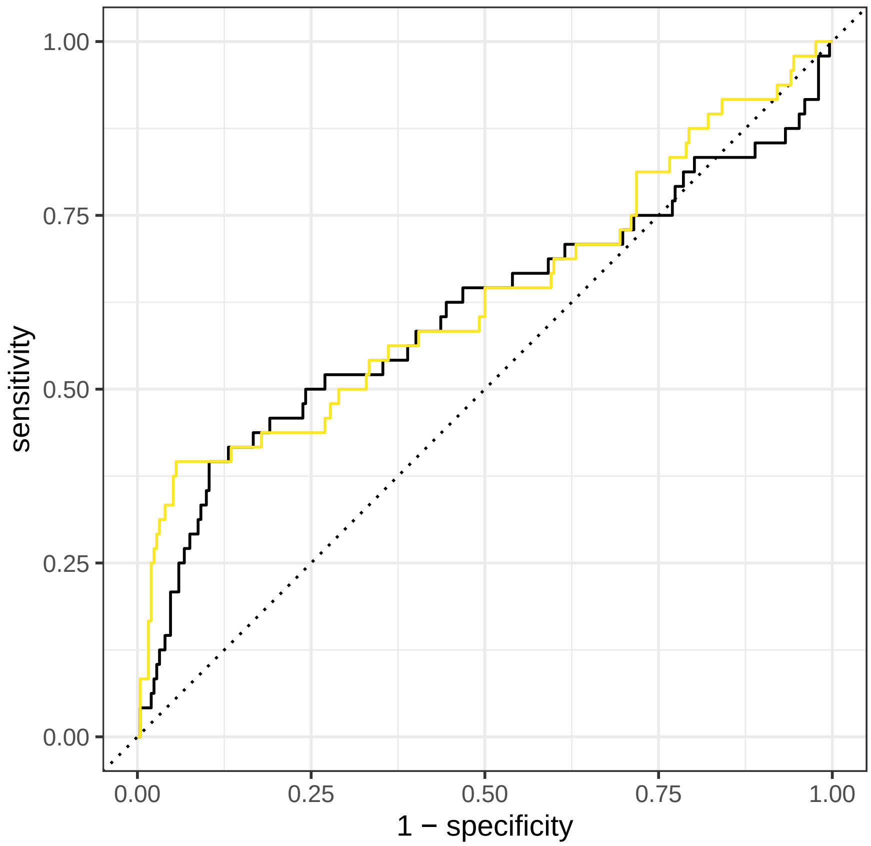
Then, calculate the area under the receiver-operator characteristic (ROC) curve for **glm_2_probs** using **roc_auc()**.

Examine the results.

Questions 8.3: Answer these questions: (1) What is the area under the ROC curve for this model? (2) Does the *first* or *second* model predict better?

Responses 8.3: *(1)0.631 (2) the second model predicted better* .

```
## second logistic regression model
glm_2 <- glm_wrkflw %>%
  add_formula(
    churn ~ neigh_churn_prob_1 + neigh_churn_prob_2
    ) %>%
  fit(training(employee_nodes_split))

##evaluate predictions
glm_2_probs <- predict(
  glm_2,
  testing(employee_nodes_split),
  type = "prob"
  ) %>%
  bind_cols(testing(employee_nodes_split))


## ROC curve
glm_2_roc <- glm_2_probs %>%
  roc_curve(churn, .pred_Yes)

##plotting first model
autoplot(glm_1_roc) +
  geom_path(
    data = glm_2_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity),
    color = "#FDE725FF"
    )
```




```
##area under ROC curve
glm_2_probs %>%
  roc_auc(churn, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.631
```

Task 8.4

Fit a third *logistic regression* model using **glm_wrkflw** on the *training* data of **employee_nodes_split** where **churn** is predicted by seven network node measures: (1) **neigh_churn_prob_1**, (2) **neigh_churn_prob_2**, (3) **eigen**, (4) **mean_page_rank**, (5) **closeness**, (6) **loc_trans**, and (7) **n_triangle**. Save the model as **glm_3**.

Then, apply **predict()** to **glm_3** to calculate *probabilities* of **churn** in the *testing* data of **employee_nodes_split**. Bind the predictions with the *testing* data. Save the result as **glm_3_probs**.

Then, apply **roc_curve()** to **glm_3_probs**. Save the result as **glm_3_roc**.

Apply **ggplot()** to start a plot, add a diagonal reference line with **geom_abline()**, add **glm_1_roc**, **glm_2_roc**, and **glm_3_roc** via **geom_path()**. Make sure to name the colors of each path inside of **geom_path()** for each model. Add appropriate labels for the axes and legend. Use **scale_color_manual()** to label the paths in the legend. Use **theme_linedraw()** as the theme. Position the legend at the bottom. Display the plot.

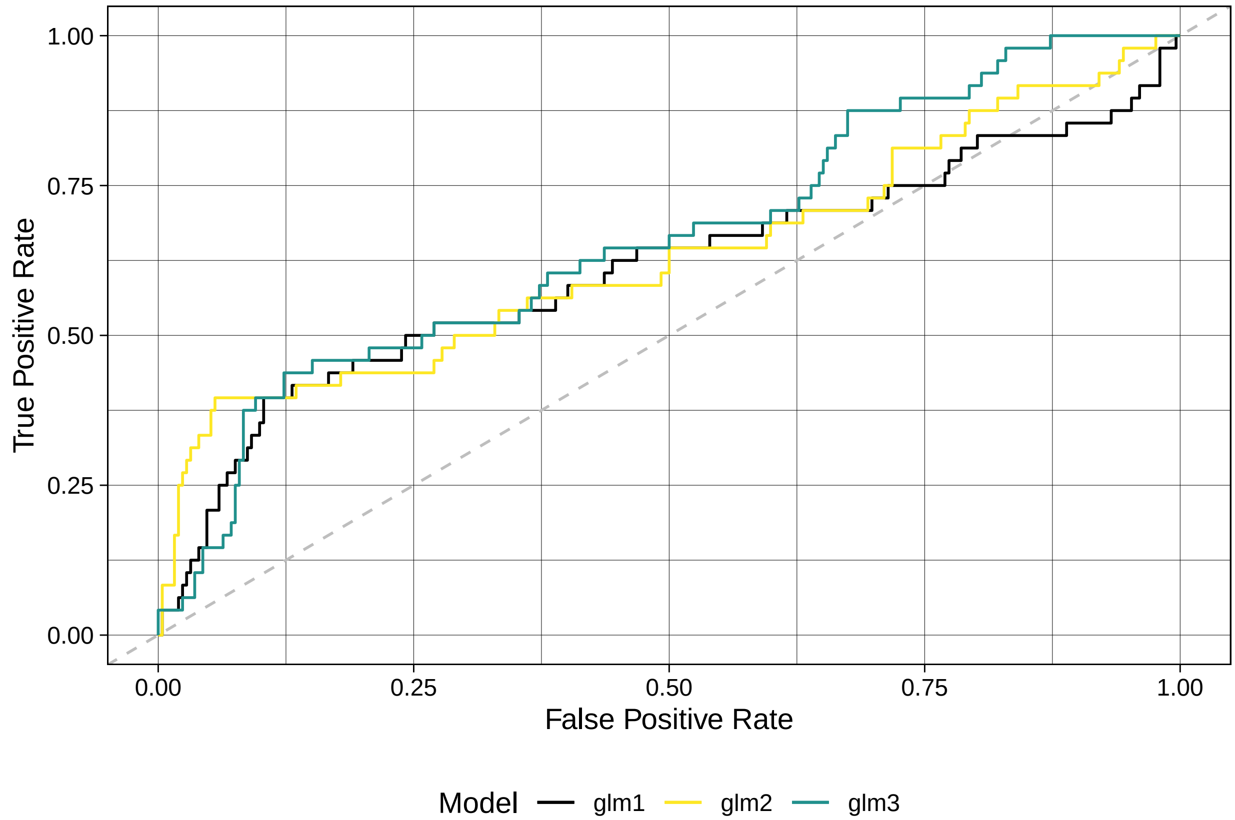
Then, calculate the area under the receiver-operator characteristic (ROC) curve for **glm_3_probs** using **roc_auc()**.

Examine the results.

Questions 8.4: Answer these questions: (1) What is the area under the ROC curve for this model? (2) Which model predicts the best?

Responses 8.4: *(1) 0.661 (2) model 3 predicts the best.*

```r
## second logistic regression model
glm_3 <- glm_wrkflw %>%
  add_formula(
    churn ~ neigh_churn_prob_1 + neigh_churn_prob_2 +
      eigen + mean_page_rank + closeness + loc_trans + n_triangle ) %>%
  fit(training(employee_nodes_split))


##evaluate predictions
glm_3_probs <- predict(
  glm_3,
  testing(employee_nodes_split),
  type = "prob"
) %>%
  bind_cols(testing(employee_nodes_split))


##ROC curve
glm_3_roc <- glm_3_probs %>%
  roc_curve(churn, .pred_Yes)


### save plot
roc_plot <- ggplot() +
  geom_abline(intercept = 0, slope = 1, linetype = 2, color = "gray") +
  geom_path(
    glm_1_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "glm1")
  )+
  geom_path(
    data = glm_2_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "glm2")
    )+
  geom_path(
    data = glm_3_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "glm3"),
    )+
  scale_color_manual(
    values = c("glm1" = "#000000", "glm2" = "#FDE725FF", "glm3" = "#21908CFF")
    )+
  labs(x = "False Positive Rate", y = "True Positive Rate", color = "Model") +
  theme_linedraw() +
  theme(legend.position = "bottom")


roc_plot
```




```
##area under ROC curve
glm_3_probs %>%
  roc_auc(churn, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.661
```

Task 9: Save Objects and Plots

For this task, you will save some of the objects and plots created in this script.

Task 9.1

Save the following objects into a single *RData* file named **employee_net_objects.rdata** in the **data** folder of the project directory:

1. **employee_tg**,
2. **employee_nodes**,
3. **employee_net_samp_plot**,
4. **glm_wrkflw**, and
5. **glm_1**, **glm_2**, and **glm_3**.

Save the three plot objects as **png** files into the **plots** folder of the project directory:

1. **employee_net_samp_plot** as **employee_net_samp.png**,
2. **employee_net_samp_facet_plot** as **employee_net_samp_facet.png**, and
3. **roc_plot** as **roc.png**.

Alter the *width* and *height* for the plots as needed.

```
##save working data
save(
  employee_tg,
  employee_nodes,
  employee_net_samp_plot,
  glm_wrkflw,
  glm_1, glm_2, glm_3,
  file = here("data", "employee_net_objects.rdata")
  )


## save plots to folder in project directory
ggsave(
  here("plots", "employee_net_samp.png"),
  plot = employee_net_samp_plot,
  units = "in", width = 9, height = 9
  )


## save a single plot to a file
ggsave(
  here("plots", "employee_net_samp_facet.png"),
  plot = employee_net_samp_facet_plot,
  units = "in", width = 12, height = 6
  )


## save a single plot to a file
ggsave(
  here("plots", "roc.png"),
  plot = roc_plot,
  units = "in", width = 12, height = 6)
```

Task 10: Conceptual Questions

For your last task, you will respond to conceptual questions based on the conceptual lectures for this week.

Question 10.1: Describe how to compute *neighbor-based* node metrics in networks.

Response 10.1: *We can compute for all the nodes how many neighbors they have by multiplying adjacency matrix by a vector that represents whether or not a node is a color. That will give you the number of neighbors they have of a color..*

Question 10.2: What is the difference between *closeness centrality* and *betweenness centrality* of nodes in networks?

Response 10.2: *Closenss centrality measures how many steps it takes to access every node from a focal node. Betweeness centrality measures the number of the shorests paths going through a node or edge. .*

Question 10.3: What is a *receiver-operator characteristic* curve?

Response 10.3: *The receiver-operator characteristic curve plots the true positive rate and false positive rate against each other for all possible decision thresholds..*