HW4 - Network Visualizations

Name: Anna Martirosyan

Deadline 4 December 2022

Network Visualizations

Problem 1 (70 points)

1) Load the file "transfers.csv" into a dataset called **transfers**.
Group by the variables **League_from** and **League_to**, then create new variables **Mean_Age**(Mean of the variable **Age**), **Mean_Fee** (Mean of the variable **Transfer_fee** divided by 1000000) and **Count**(number of transfers in that group) by using the summarize function and filter out the observations with less than 10 transfers in between (**Count** >= 10) (5 points)

```
library(dplyr)
transfers = read.csv("transfers.csv")
transfers_1 = transfers %>%
   group_by(League_from, League_to)%>%
   summarise(Mean_Age=mean(Age),
             Mean_Fee = (mean(Transfer_fee))/1000000,
             Count =n())%>%
  filter(Count >= 10)
```

```
## `summarise()` has grouped output by 'League_from'. You can override using the
## `.groups` argument.
```

```
head(transfers_1, n=5)
```

```
## # A tibble: 5 x 5
## # Groups:   League_from [2]
##   League_from  League_to      Mean_Age Mean_Fee Count
##   <chr>        <chr>             <dbl>    <dbl> <int>
## 1 1.Bundesliga 1.Bundesliga       24.0    10.6    106
## 2 1.Bundesliga LaLiga             23.7    16.9     18
## 3 1.Bundesliga Premier League     24.1    20.5     45
## 4 1.Bundesliga Serie A            25.8    16.4     13
## 5 Eredivisie   1.Bundesliga       23.4     7.71    17
```

2) Create an igraph using `graph_from_edgelist` for the dataset you just got.
(5 points)

```
matrix = cbind(transfers_1$League_from, transfers_1$League_to)
graph = igraph::graph_from_edgelist(matrix,directed = T)
head(graph, n=5)
```

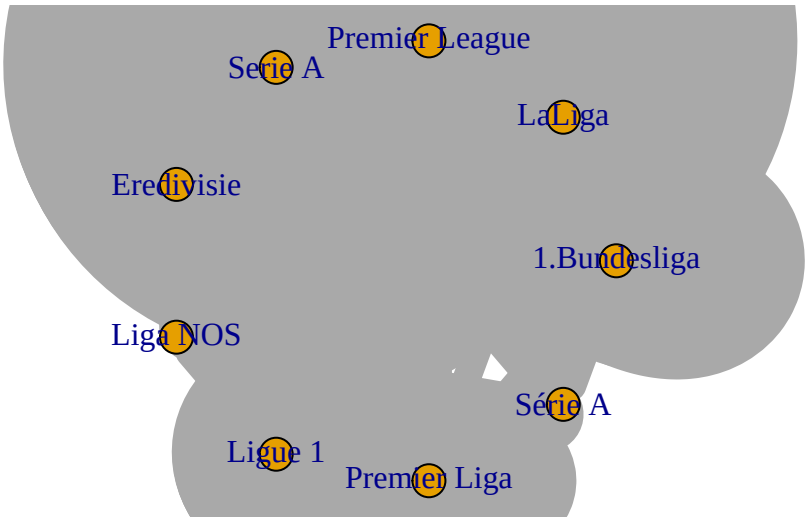
```
## 5 x 9 sparse Matrix of class "dgCMatrix"
##                1.Bundesliga LaLiga Premier League Serie A Eredivisie Liga NOS
## 1.Bundesliga              1      1              1       1          .        .
## LaLiga                    1      1              1       1          .        1
## Premier League            1      1              1       1          .        .
## Serie A                   1      1              1       1          .        .
## Eredivisie                1      1              1       .          1        .
##                Ligue 1 Premier Liga Série A
## 1.Bundesliga         .            .       .
## LaLiga               1            1       .
## Premier League       1            .       .
## Serie A              1            1       .
## Eredivisie           .            .       .
```

3) Set edge attributes for the summarized three variables and use their values. (5 points)

```
graph = set_edge_attr(graph, "Mean_Age", value = transfers_1$Mean_Age )
graph = set_edge_attr(graph, "Mean_Fee", value = transfers_1$Mean_Fee)
graph = set_edge_attr(graph, "Count", value = transfers_1$Count)
```

4) Draw the network in a circle layout, pick one of the attributes as the width of the edge. Were you able to get an interpretable graph ? If no what's the reason ? If yes interpret it :) (10 points)

```
plot(graph,
     layout = layout_in_circle(graph),
     lable = V(graph),
     mode="circle",
     main="Visualization 1",
     edge.width = E(graph)$"Count")
```



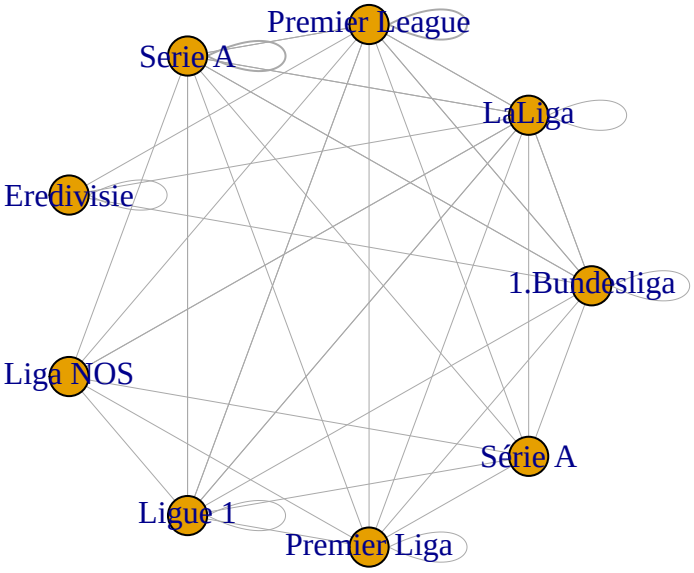

```
# Interpretation:
# The graph is impossible to interpret, because default edge width is 1, but
# in out case, as we put the edge width to be equal to the "Count", the edges
# become too thick which made impossible to understand their connections. In
# situations like this we need to transform and scale the variable for the edge.
```

5) Apply a min-max or any other reasonable transformation to the variable chosen by you and visualize again using the scaled variable for now. !Hint, if you are using min/max replace 0 with a small number. (10 points)

```
count = E(graph)$"Count"
count = (count-min(count))/(max(count)-min(count))
count[which(count ==0, arr.ind=T)] = 0.000001
graph = set_edge_attr(graph, "scaled.count", value=count)

plot(graph,
     layout = layout_in_circle(graph),
     lable = V(graph),
     mode="circle",
     main="Visualization 2",
     edge.width = E(graph)$"scaled.count",
     edge.arrow.mode =0.6)
```

Visualization 2



6) Interpret the graph (10 points)

You can write your interpretations directly here: Now, the graph is possible to interpret because, we scaled the "Count" and got "scaled.count" for the edge.width. The leagues are vertices and edges show the count of transfers from one league to another. The width of the edge is directly proportional to the count of transfers, which means if the count is higher, the edge is thicker, and vice versa. The self loops are also possible, because some transfers happen between 2 teams of the same league. The thickest edge is between Seria A and Seria A, because the number of transfers (288) is the highest between Seria A and Seria A. And the thinnest edge is between Premier Liga and 1.Bundesliga because the number of transfers is only 10.

7) Using the dataset *transfers* create another subset of data. For now, pick any of the leageus that you want from the available ones and filter out the internal transfers in that league (meaning that the variables *League_from* and *League_to* are equal to each other). Now group by the variables *Team_from* and *Team_to* and summarize the same values as in Problem 1. (5 points)

```
sub_data = transfers %>%
  filter(League_from=="Liga NOS", League_from == League_to)%>%
  group_by(Team_from, Team_to)%>%
  summarise(Mean_Age=mean(Age),
            Mean_Fee = (mean(Transfer_fee))/1000000,
            Count =n())
```

```
## `summarise()` has grouped output by 'Team_from'. You can override using the
## `.groups` argument.
```

```
head(sub_data, n=5)
```

```
## # A tibble: 5 x 5
## # Groups:   Team_from [3]
##   Team_from   Team_to     Mean_Age Mean_Fee Count
##   <chr>       <chr>          <dbl>    <dbl> <int>
## 1 Braga       Benfica           26     10       2
## 2 Braga       FC Porto          25      6.5     1
## 3 Braga       Sporting CP       24      2       1
## 4 Marítimo    FC Porto          22      2       1
## 5 Sporting CP FC Porto          23     11       1
```

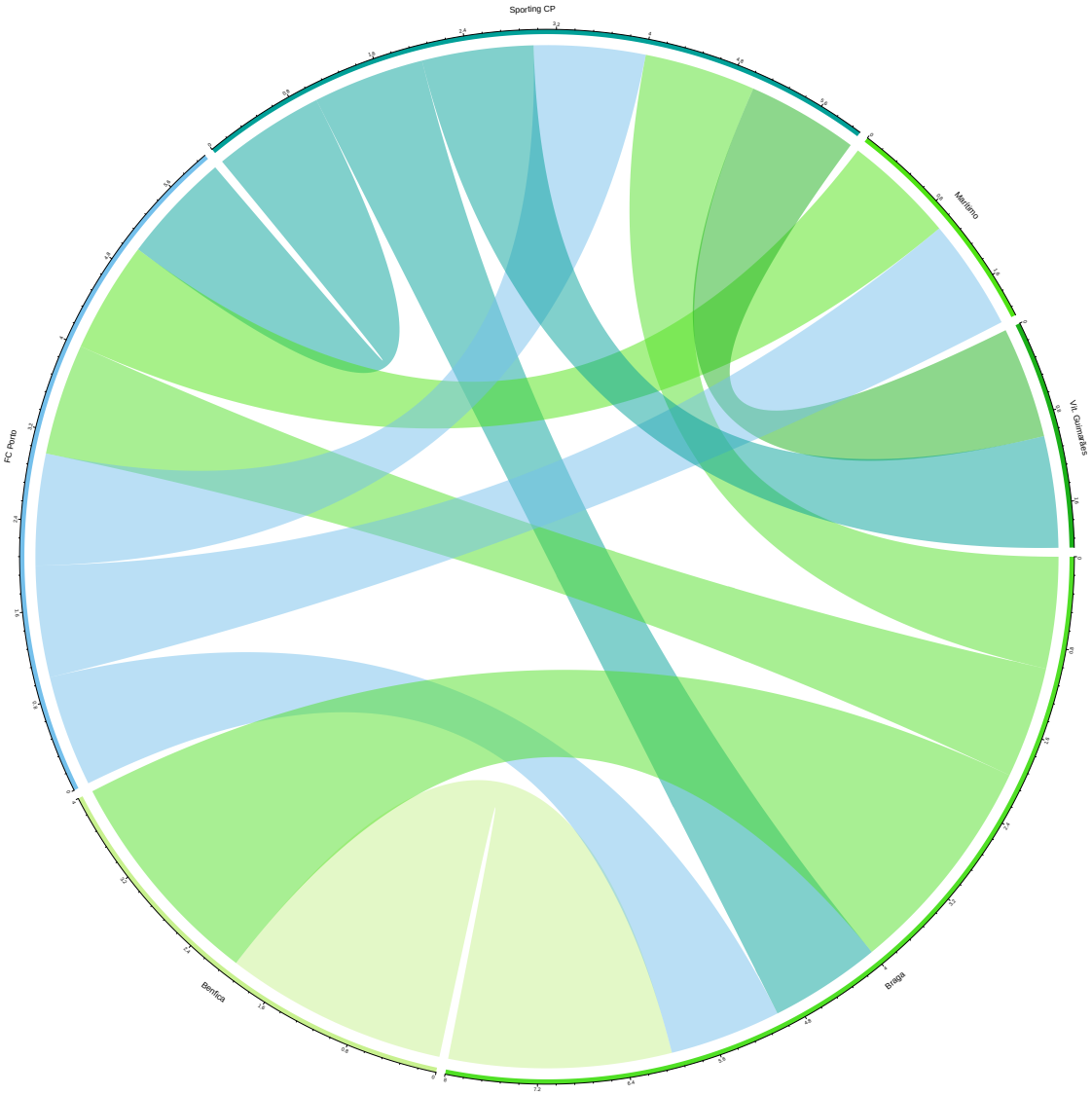
8) Create an igraph network and set the edge attributes for the graph, the same way as in problem 3. (5 points)

```
matrix_2 = cbind(sub_data$Team_from, sub_data$Team_to)
graph_2 = igraph::graph_from_edgelist(matrix_2, directed = F)
graph_2 = set_edge_attr(graph_2, "Mean_Age", value = sub_data$Mean_Age )
graph_2 = set_edge_attr(graph_2, "Mean_Fee", value = sub_data$Mean_Fee)
graph_2 = set_edge_attr(graph_2, "Count", value = sub_data$Count)
head(graph_2, n=5)
```

```
## 5 x 6 sparse Matrix of class "dgCMatrix"
##             Braga Benfica FC Porto Sporting CP Marítimo Vit. Guimarães
## Braga           .       1        1           1        .              .
## Benfica         1       .        .           .        .              .
## FC Porto        1       .        .           1        1              .
## Sporting CP     1       .        1           .        .              1
## Marítimo        .       .        1           .        .              .
```

9) Now use that graph and draw a chordDiagram and use any of the three attributes you desire, apply the transformations to the variables if needed. (10 points)

```
as_m <- as.matrix(intergraph::asNetwork(graph_2),
                  matrix.type = "adjacency",
                  attrname="Count")
chordDiagram(as_m)
```



10) Interpret the visualization. (5 points)

The above graph shows connections between teams of Liga NOS. The data is arranged radially around a circle, and the count of transfers between the vertices(teams) are drawn as asymmetric arcs connecting the vertices. The arcs show transfer counts between 2 teams. Since the count of transfers between all teams are equal to each other and is equal to 1, only except from Braga to Benfica (count is equal to 2), only the arcs from Braga to Benfica ate thicker than others.

Network Statistics (30 points)

Problem 2

1) Using the network created in **Problem 1.2** calculate the edge density of the network. Interpret the result. (5 points)

```
(density= edge_density(graph, loops=F))
```

```
## [1] 0.6527778
```

```
# The density can range from 0 to 1 and shows how interconnected are the nodes
# within the network. Self loops are counted in  the total  number of edges so
# with self loops can have density higher than 1.
```

2) Now calculate and interpret the reciprocity of the network. (5 points)

```
(reciprocity = reciprocity(graph))
```

```
## [1] 0.5
```

3) Find the most active league of the market. by calculating the degree centrality measure (5 points)

```
which.max(igraph::degree(graph, mode="out", normalized=T))
```

```
## LaLiga
##      2
```

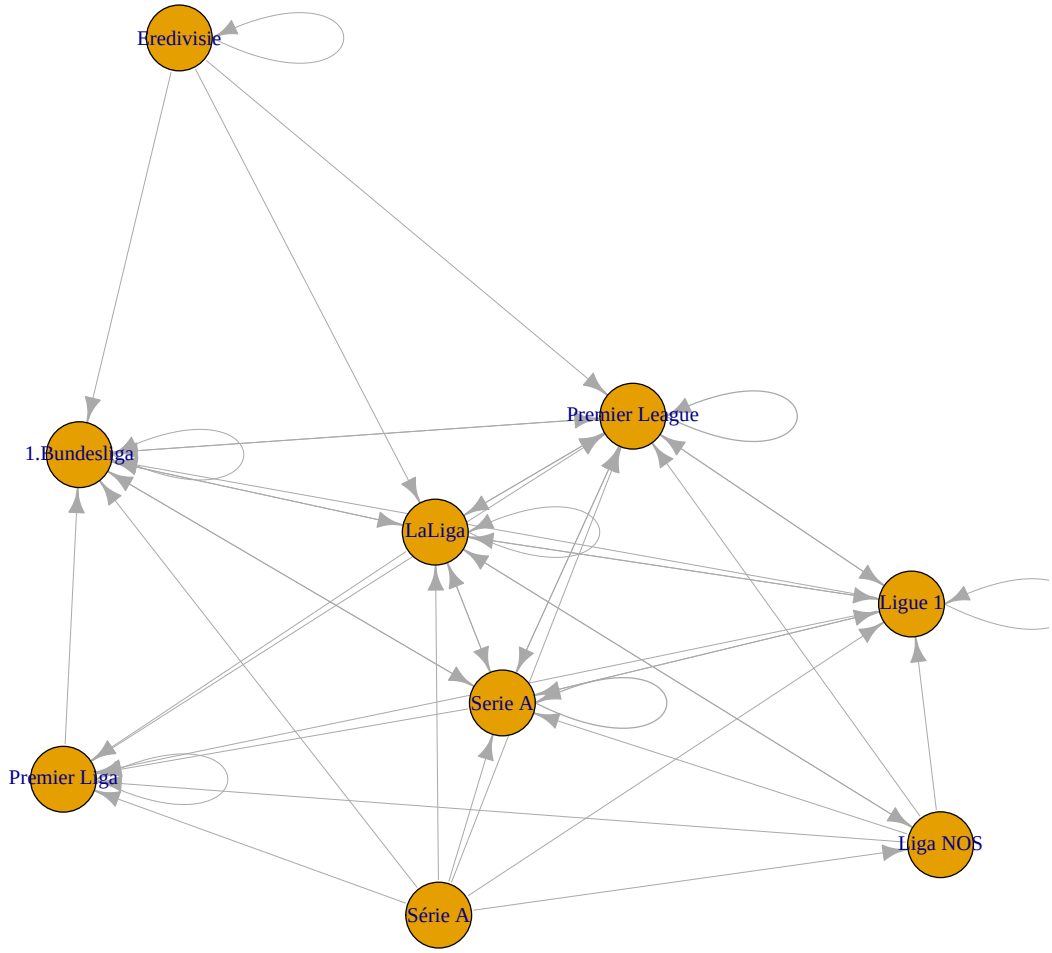
```
#LaLiga is the most active league of the market
```

4) Calculate the closeness statistics of the network, pick any of the edge attributes as a weight and interpret the results. (5 points)

```
igraph::closeness(graph)
```

```
##   1.Bundesliga         LaLiga Premier League        Serie A     Eredivisie
##     0.11111111     0.16666667     0.12500000     0.14285714     0.09090909
##       Liga NOS        Ligue 1   Premier Liga        Série A
##     0.14285714     0.14285714     0.09090909     0.14285714
```

```
plot(graph, displaylabels=T, edge.width = E(graph)$"scaled.count")
```




```
# Interpretation: Closeness statistics of my graph network show how close each
# league is to everyone else. In out case, LaLiga and Premier League are the
# closest to others since their closeness statistics is the highest (0.125),
# and the Eredivisie is the farest, since it has the lowest score
# (approximately 0.077).
```

5) Now calculate the page rank of the network, which league is the most popular ?. (5 points)

```
page_rank(graph)$vector
```

```
##   1.Bundesliga         LaLiga Premier League        Serie A     Eredivisie
##     0.18587216     0.15957614     0.19234365     0.15507879     0.02116402
##       Liga NOS        Ligue 1   Premier Liga        Série A
```

```
##      0.03806758      0.11558095      0.11565004      0.01666667
```

```
# The most popular is LaLiga since its page rank is the biggest.
```

6) Detect communities in the network. (5 points)

```
edge_b <- edge.betweenness.community(graph)
plot(edge_b, graph, main="Edge Betweenness")
```