2020 Tokyo Olympic Network Analysis

Jangmin Song

2025-04-08

Libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readxl)
library(stringr)
library(ggplot2)
library(RColorBrewer)
library(stringr)
```

Figure out which of these packages is already installed

```
# Store all installed packages
ya_installed <- library()$results[,1]

# Check whether required packages are already installed and grab only those that still need installatio
need_install<-my_packages[!(my_packages %in% ya_installed)]

#install required packages
lapply({need_install}, install.packages, character.only = TRUE)
```

Now, load only unloaded packages

```
# Store all installed packages
ya_loaded <- (.packages())

# Check whether required packages are already installed and grab only those that still need installatio
need_load<-my_packages[!(my_packages %in% ya_loaded)]

# Load required packages
lapply(need_load, require, character.only = TRUE)
```

```
## Loading required package: bipartite
```

```
## Loading required package: sna
```

```
## Loading required package: statnet.common

##
## Attaching package: 'statnet.common'

## The following objects are masked from 'package:base':
##
##     attr, order

## Loading required package: network

##
## 'network' 1.19.0 (2024-12-08), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information

## sna: Tools for Social Network Analysis
## Version 2.8 created on 2024-09-07.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.

## Loading required package: vegan

## Loading required package: permute

## Loading required package: lattice

##  This is bipartite 2.21.
##  For latest changes see versionlog in ?"bipartite-package". For citation see: citation("bipartite").
##  Have a nice time plotting and analysing two-mode networks.

##
## Attaching package: 'bipartite'

## The following object is masked from 'package:vegan':
##
##     nullmodel

## Loading required package: igraph

##
## Attaching package: 'igraph'

## The following object is masked from 'package:bipartite':
##
##     strength

## The following object is masked from 'package:vegan':
##
##     diversity

## The following object is masked from 'package:permute':
##
##     permute

## The following objects are masked from 'package:sna':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census
```

```
## The following objects are masked from 'package:network':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

Read in Data

Read in the csv files

```
athletes_full<-read.csv("athletes.csv")
athletes <- athletes_full %>%
  select(name, country, disciplines, events, birth_date)

medals_full <-read.csv("medallists.csv")
medals <- medals_full %>%
  select(name, medal_type, medal_code, country, discipline, event)

athletes$disciplines <- str_replace_all(athletes$disciplines, "\\['", "")
athletes$disciplines <- str_replace_all(athletes$disciplines, "'\\]", "")
athletes$disciplines <- str_replace_all(athletes$disciplines, "\\[\"", "")
athletes$disciplines <- str_replace_all(athletes$disciplines, "\"\\]", "")

athletes$events <- str_replace_all(athletes$events, "\\['", "")
athletes$events <- str_replace_all(athletes$events, "'\\]", "")
athletes$events <- str_replace_all(athletes$events, "\\[\"", "")
athletes$events <- str_replace_all(athletes$events, "\"\\]", "")

medals$medal_type <- str_replace_all(medals$medal_type, " Medal", "")
```

Creating the bipartite graphs

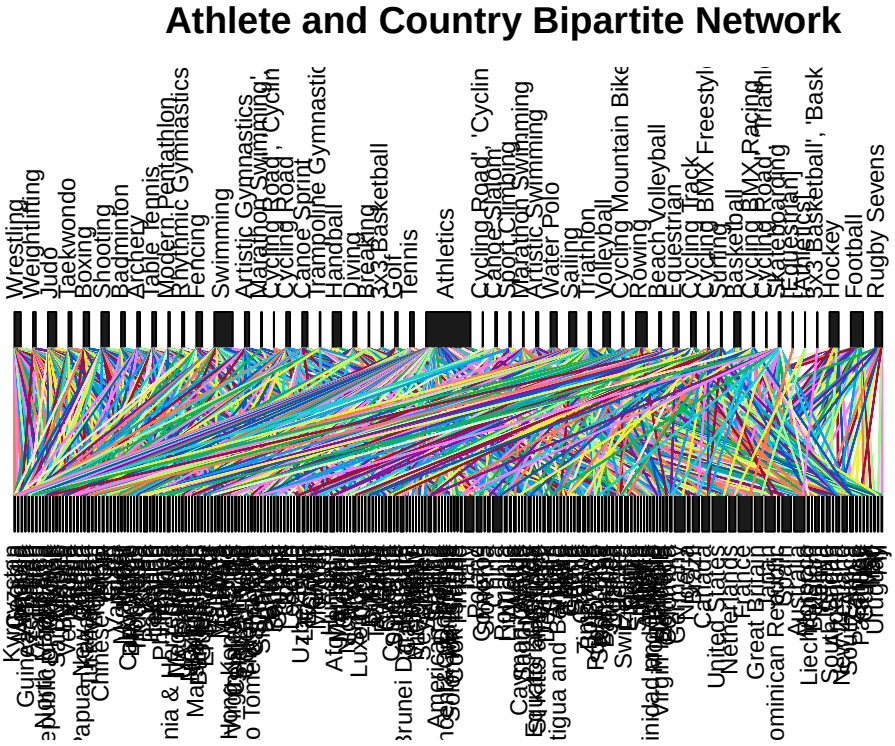
```
#Creating webID
athletes_webID <- data.frame(matrix("olympicweb", nrow = nrow(athletes), ncol = 1))
new_athletes <- cbind(athletes, athletes_webID)
colnames(new_athletes)<- c("name", "country", "disciplines", "events", "birth_date", "webID")
```

Plotting network interactions between two node types

```
#Generate the graph object using frame2webs()
web_athletes <- frame2webs(new_athletes, varnames = c("country", "disciplines", "webID"), type.out = "li
```

Bipartite Graph of new_athletes

```
#creating a color vector
cols1 <-c( '#8214a0', '#005ac8', '#00a0fa', '#fa78fa', '#14d2dc', '#aa0a3c', '#fa7850', '#0ab45a', '#f0
#plotting the two-dimensional matrix to a bipartite graph
plotweb(web_athletes$"olympicweb", method='cca', labsize=1.2, x.lim=c(0,4), y.lim=c(-0.6,2.8), text.rot=
title("Athlete and Country Bipartite Network")
```




```
#PNG
png("athlete_country_bipartite.png", width = 3000, height = 1200, res = 300)
plotweb(web_athletes$"olympicweb", method = 'cca', labsize=0.6, y.lim=c(-0.5,2), text.rot = 90,
        col.interaction = cols1, bor.col.interaction = cols1)
dev.off()
```

```
## pdf
##   2
```

```
athletes_by_country <- athletes %>%
  count(country, name = "num_athletes")

# countrys with only 1 or 2 athletes
small_country <- athletes_by_country %>%
  filter(num_athletes <= 15) %>%
  pull(country)

# Update country in original data
athletes_cleaned <- athletes %>%
  mutate(country = ifelse(country %in% small_country, "Other Countries", country))

#Creating webID
athletes_cleaned_webID <- data.frame(matrix("olympicweb", nrow = nrow(athletes_cleaned), ncol = 1))
```

```
new_athletes_cleaned <- cbind(athletes_cleaned, athletes_cleaned_webID)
colnames(new_athletes_cleaned)<- c("name", "country", "disciplines", "events", "birth_date", "webID")
```

Plotting network interactions between two node types

```
#Generate the graph object using frame2webs()
web_cleaned_athletes <- frame2webs(new_athletes_cleaned, varnames = c("country", "disciplines", "webID"
```

Bipartite Graph of new_athletes

```
#creating a color vector
cols1 <-c( '#8214a0', '#005ac8', '#00a0fa', '#fa78fa', '#14d2dc', '#aa0a3c', '#fa7850', '#0ab45a', '#f0
#plotting the two-dimensional matrix to a bipartite graph
plotweb(web_cleaned_athletes$"olympicweb", method='cca', labsize=1.2, x.lim=c(0,4), y.lim=c(-0.6,2.8),
title("Simplified Athlete and Country Bipartite Network")
```

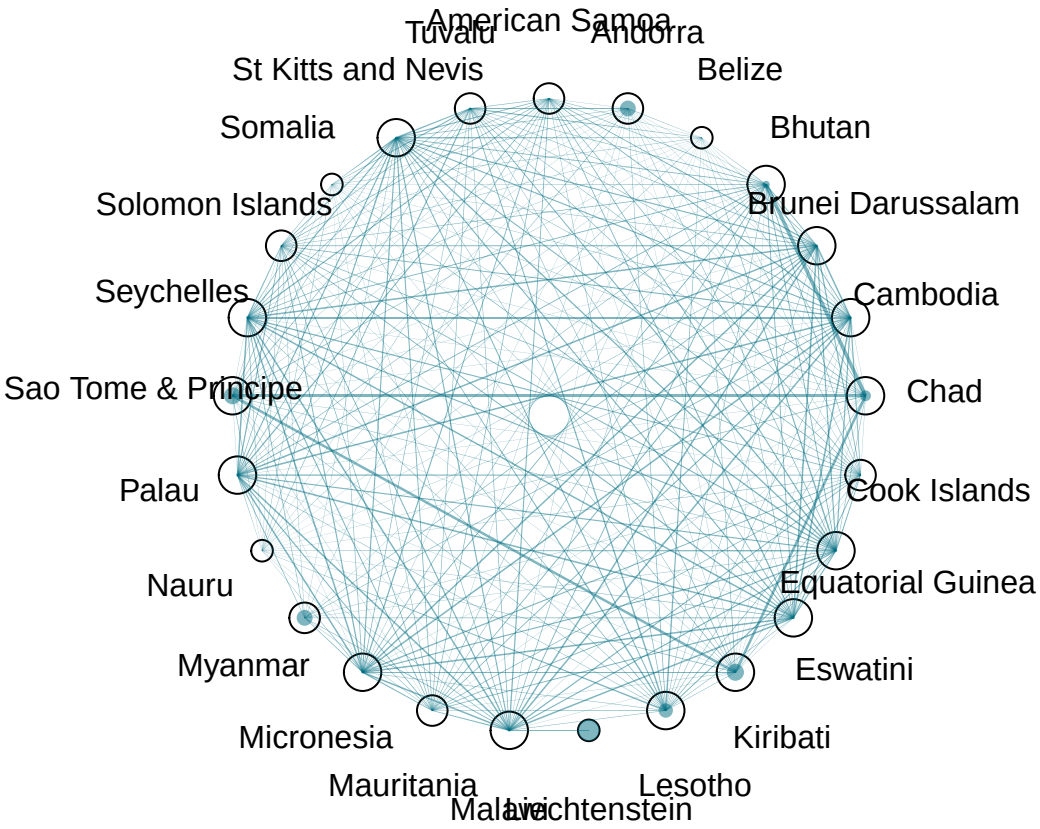


```
# Save as PNG
png("simple_athlete_country_bipartite.png", width = 3000, height = 1200, res = 300)
plotweb(web_cleaned_athletes$"olympicweb", method='cca', labsize=0.6, y.lim=c(-0.2,2), text.rot=90,
        col.interaction=cols1, bor.col.interaction=cols1)

dev.off()
```

```
## pdf
##   2
```

```
bottom_20_countrys <- athletes_by_country %>%
  slice_min(num_athletes, n = 20) %>%
  pull(country)
athletes_bottom20 <- athletes %>%
  filter(country %in% bottom_20_countrys)
```

```
athletes_bottom20_webID <- data.frame(matrix("olympicweb", nrow = nrow(athletes_bottom20), ncol = 1))
new_athletes_bottom20 <- cbind(athletes_bottom20, athletes_bottom20_webID)
colnames(new_athletes_bottom20)<- c("name", "country", "disciplines", "events", "birth_date", "webID")
web_new_athletes_bottom20 <- frame2webs(new_athletes_bottom20, varnames = c("country", "disciplines", "

web_mat <- web_new_athletes_bottom20$"olympicweb"

par(mar = c(5, 5, 5, 5))  # Extra margins for label room
plotPAC(
  web_mat,
  scaling = 2,
  fill.col = rgb(0, 110/255, 130/255, 0.5),
  arrow.col = rgb(0, 110/255, 130/255, 0.5)
)
```




```
rownames(new_athletes_bottom20$"olympicweb")
```

```
## NULL
```

Bhutan and Chad share Archery and Athletics, and Sao Tome and Principe and Chad share Athletics and Judo

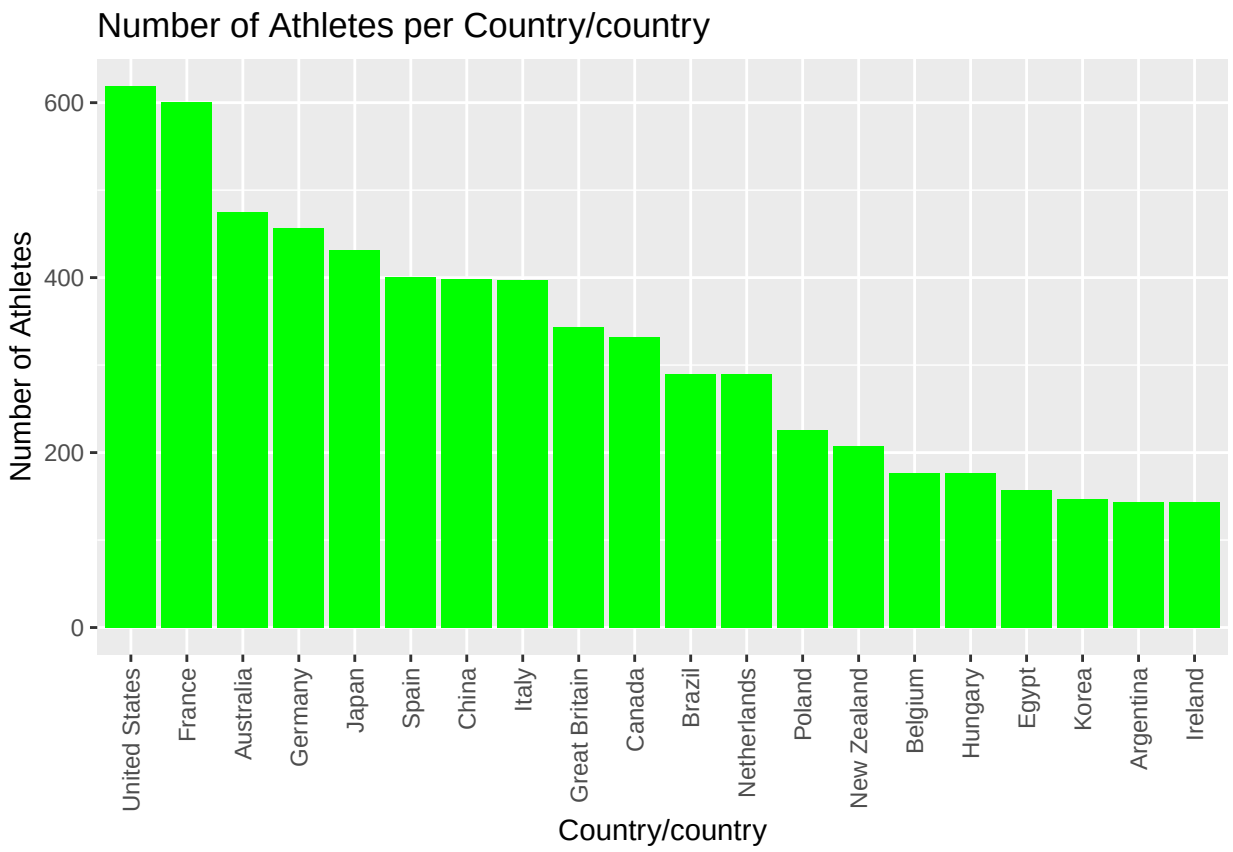
```
athletes_by_country <- athletes %>%
  count(country, name = "num_athletes") %>%
  arrange(desc(num_athletes))
```

```
theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

```
## List of 1
##  $ axis.text.x:List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi FALSE
```

```
##  - attr(*, "validate")= logi TRUE
```

```
top_countrys <- athletes_by_country %>% slice_max(num_athletes, n = 20)

ggplot(top_countrys, aes(x = reorder(country, -num_athletes), y = num_athletes)) +
  geom_bar(stat = "identity", fill = "green") +
  labs(title = "Number of Athletes per Country/country",
       x = "Country/country",
       y = "Number of Athletes") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



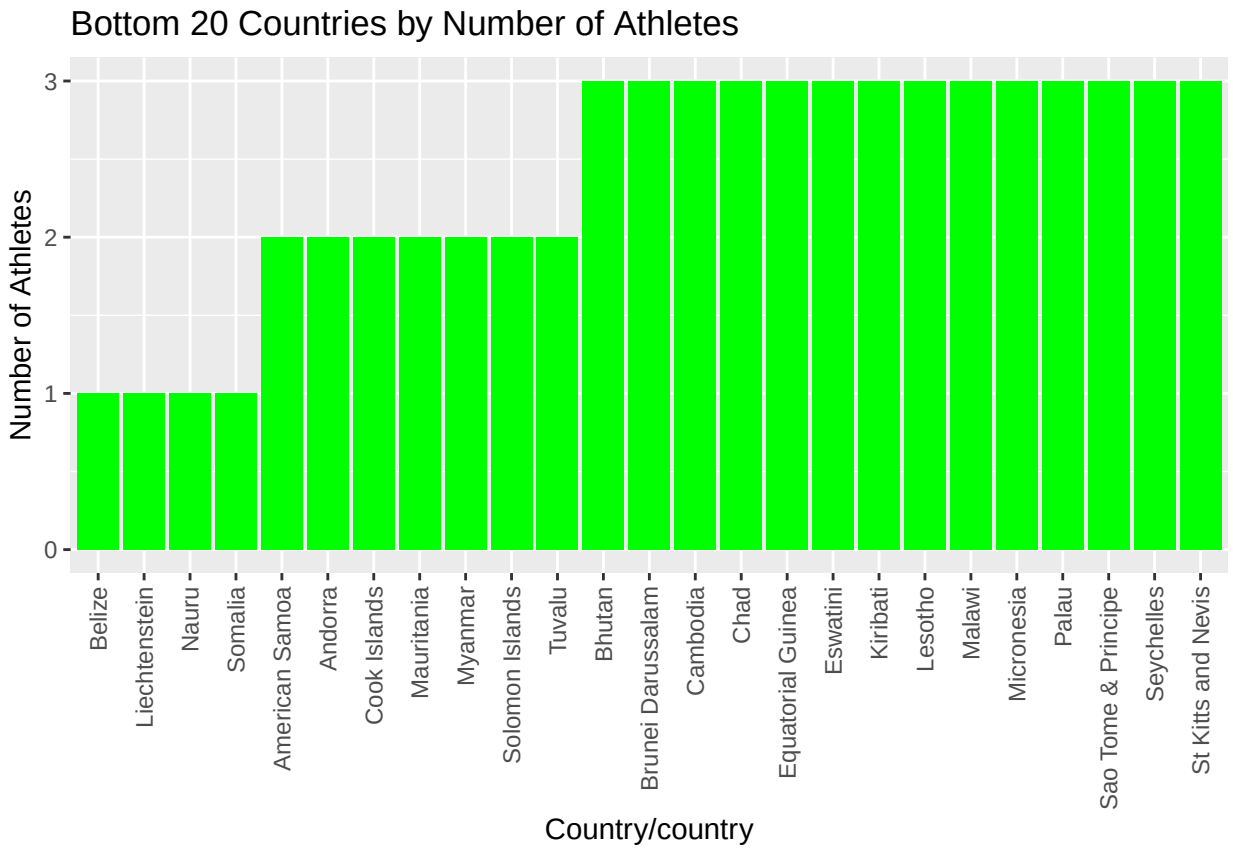

```
# Save as PNG
png("num_athletes_vs_top_country.png", width = 3000, height = 1200, res = 300)
ggplot(top_countrys, aes(x = reorder(country, -num_athletes), y = num_athletes)) +
  geom_bar(stat = "identity", fill = "green") +
  labs(title = "Number of Athletes per Country/country",
       x = "Country/country",
       y = "Number of Athletes") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
dev.off()
```

```
## pdf
##   2
```

```
bottom_countrys <- athletes_by_country %>%
  slice_min(num_athletes, n = 20)
```

```
ggplot(bottom_countrys, aes(x = reorder(country, num_athletes), y = num_athletes)) +
  geom_bar(stat = "identity", fill = "green") +
```

```
  labs(title = "Bottom 20 Countries by Number of Athletes",
       x = "Country/country",
       y = "Number of Athletes") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```




```
# Save as PNG
png("num_athletes_vs_bottom_country.png", width = 3000, height = 1200, res = 300)
ggplot(bottom_countrys, aes(x = reorder(country, num_athletes), y = num_athletes)) +
  geom_bar(stat = "identity", fill = "green") +
  labs(title = "Bottom 20 Countries by Number of Athletes",
       x = "Country/country",
       y = "Number of Athletes") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
dev.off()
```

```
## pdf
##   2
```

```
name_medal <- as.matrix(cbind(medals$name, medals$medal_type))
japan_medalist <- as.matrix(cbind(name_medal,medals$country))
japan_medalist<-as.data.frame(japan_medalist)
japan_medalist <- japan_medalist %>%
  filter(V3 == "Japan") %>%
  select(V1, V2)
japan_medalist <- as.matrix(japan_medalist)
japan_medalist.g <- graph_from_edgelist(japan_medalist, directed = FALSE)
V(japan_medalist.g)$type <- bipartite_mapping(japan_medalist.g)$type
bipart_data_japan <- as_biadjacency_matrix(japan_medalist.g)
```

```
#Creating sociomatrix for medalist
japan_medalist.mat <- bipart_data_japan %*% t(bipart_data_japan)
#show athletes_medalist.mat matrix
diag(japan_medalist.mat) <- NA
```

Creating the network graph of medalists

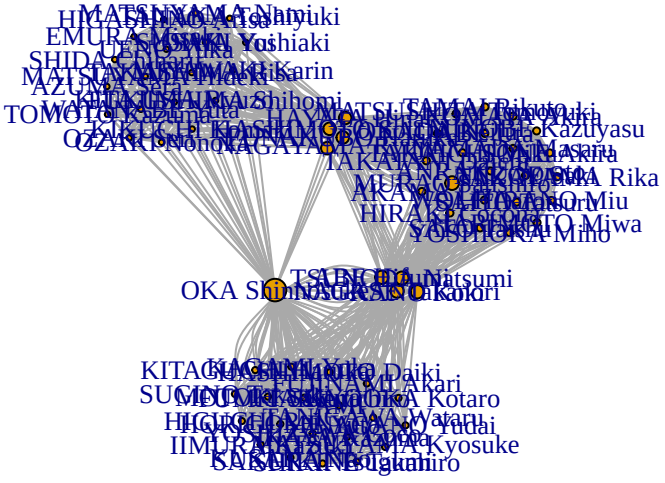
```
# Create an undirected graph from the sociomatrix of the dcs instructor network.
japan_medalist.g <- graph_from_adjacency_matrix(japan_medalist.mat, mode="undirected")
```

Plotting the network graph of courses

```
#plot features
#layout setting
la <- layout_with_fr(japan_medalist.g, niter = 1000000, area = vcount(japan_medalist.g)^8)
```

```
## Warning: The `area` argument of `layout_with_fr()` is deprecated as of igraph 0.8.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
#edge list setting
e.wt <- edge_attr(japan_medalist.g, "weight")
v.wt <- strength(japan_medalist.g, mode = "all")
v.size <- v.wt / max(v.wt) * 10  # scale node sizes between 0 and 30
#plot instructor network
plot(japan_medalist.g,
     layout=la,
     vertex.size=v.size,
     edge.width=e.wt,
     vertex.label= V(japan_medalist.g)$name,
     vertex.label.cex = 0.8
     )
```




```
name_medal <- as.matrix(cbind(medals$name, medals$medal_type))
g_medal <- graph_from_edgelist(name_medal, directed = FALSE)
V(g_medal)$type <- bipartite_mapping(g_medal)$type
```

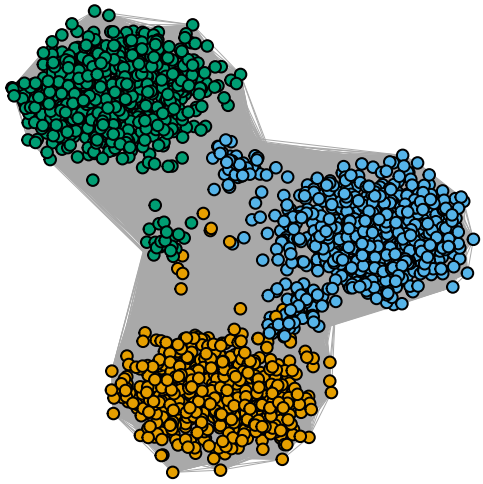
```
# Create biadjacency matrix
bipart_data <- as_biadjacency_matrix(g_medal)
```

```r
# Project to name-name matrix
medalist_mat <- bipart_data %*% t(bipart_data)
diag(medalist_mat) <- 0

# Create graph
medalist_g <- graph_from_adjacency_matrix(medalist_mat, mode = "undirected", weighted = TRUE)
```

```r
clusters <- cluster_louvain(medalist_g)
V(medalist_g)$cluster <- clusters$membership
```

```r
layout_medal <- layout_with_fr(medalist_g)
plot(medalist_g,
     layout = layout_medal,
     vertex.size = 5,
     vertex.label = NA,
     vertex.color = clusters$membership,
     edge.width = E(medalist_g)$weight / max(E(medalist_g)$weight) * 5,
     main = "Medalist Clustering Based on Shared Medal Types")
```

Medalist Clustering Based on Shared Medal Types



```r
country_discipline <- as.matrix(cbind(athletes$country, athletes$disciplines))
g_cd <- graph_from_edgelist(country_discipline, directed = FALSE)
V(g_cd)$type <- bipartite_mapping(g_cd)$type
```

```r
bipart_cd <- as_biadjacency_matrix(g_cd)
```

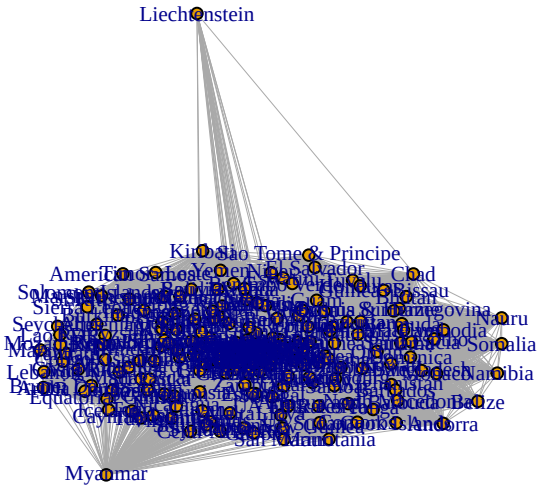
```r
country_mat <- bipart_cd %*% t(bipart_cd)
diag(country_mat) <- 0
```

```r
g_country <- graph_from_adjacency_matrix(country_mat, mode = "undirected", weighted = TRUE)

# Optional: remove 0-weight edges
g_country <- delete_edges(g_country, E(g_country)[weight == 0])
```

```r
layout_country <- layout_with_fr(g_country)

plot(g_country,
```

```
    layout = layout_country,
    vertex.size = 5,
    vertex.label.cex = 0.7,
    vertex.label = V(g_country)$name,
    edge.width = E(g_country)$weight / max(E(g_country)$weight) * 5,
    main = "Country Collaboration Network by Shared Disciplines")
```

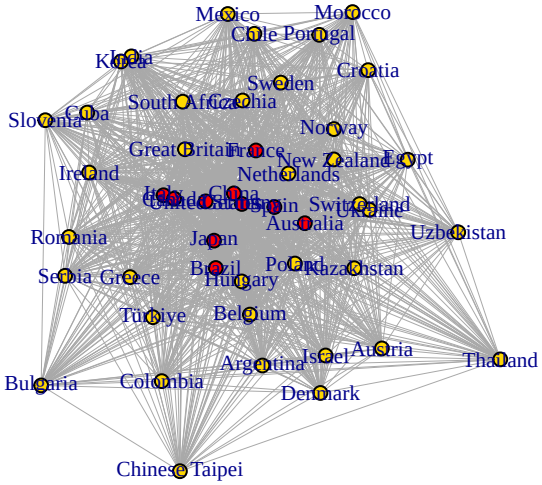
Country Collaboration Network by Shared Disciplines




```
# Step 1: Count number of disciplines per country
country_discipline_count <- athletes %>%
  distinct(country, disciplines) %>%          # each (country, discipline) pair
  count(country, name = "num_disciplines") %>%
  filter(num_disciplines > 15)                # keep only countries with >10

# Step 2: Filter original data
filtered_athletes <- athletes %>%
  filter(country %in% country_discipline_count$country)

# Step 3: Create bipartite edge list: country-discipline
country_discipline <- as.matrix(cbind(filtered_athletes$country, filtered_athletes$disciplines))
g_cd <- graph_from_edgelist(country_discipline, directed = FALSE)
V(g_cd)$type <- bipartite_mapping(g_cd)$type

# Step 4: Create biadjacency matrix and project
bipart_cd <- as_biadjacency_matrix(g_cd)
country_mat <- bipart_cd %*% t(bipart_cd)
diag(country_mat) <- 0

# Step 5: Create country-country graph
g_country <- graph_from_adjacency_matrix(country_mat, mode = "undirected", weighted = TRUE)
g_country <- delete_edges(g_country, E(g_country)[weight == 0])

top_countries <- country_discipline_count %>%
  arrange(desc(num_disciplines)) %>%
  slice(1:10) %>%
  pull(country)
```

```
V(g_country)$color <- ifelse(V(g_country)$name %in% top_countries, "red", "gold")



# Step 6: Plot
layout_country <- layout_with_fr(g_country)

plot(g_country,
     layout = layout_country,
     vertex.size = 6,
     vertex.label.cex = 0.7,
     vertex.label = V(g_country)$name,
     vertex.color = V(g_country)$color,
     edge.width = E(g_country)$weight / max(E(g_country)$weight) * 5,
     main = "Country Collaboration Network with Top 10 in Red")
```

Country Collaboration Network with Top 10 in Red



This is the network of countries where if two countries share at least one Olympic discipline, then they have the connections. Edges represent two countries share at least one Olympic discipline (e.g., Swimming, Athletics). Edge weight shows how many disciplines they have in common. Lastly, red nodes shows the top 10 countries by number of disciplines (the most sport-diverse countries)

The red nodes are the most diverse countries — they compete in the largest variety of Olympic disciplines. This suggests that those countries have greater investment in a broad Olympic program, larger delegation sizes, and Well-established athletic infrastructure because of their economic background.

If two countries are strongly connected (thick edge), they tend to send athletes to the same set of disciplines, even if they're not top-ranked countries. This may indicate that they have similar sports development priorities, regional sport trends such as Judo in Asia, and shared cultural or funding influences

```
# medals_ranked <- medals %>%
#   mutate(`Rank by Total` = as.numeric(`Rank by Total`)) %>%
#   arrange(`Rank by Total`) %>%
#   mutate(Rank_Group = ceiling(`Rank by Total` / 5))
#
# team_rank <- as.matrix(cbind(medals_ranked$`Team/country`, medals_ranked$Rank_Group))
#
```

```
# team_rank.g <- graph_from_edgelist(team_rank, directed = FALSE)
```

```
# ##############################################################################
# #I asked Chat GPT for this.
# is_rank_group <- suppressWarnings(!is.na(as.numeric(V(team_rank.g)$country)))
# # Assign color
# V(team_rank.g)$color <- ifelse(is_rank_group, "grey", "gold")
# ##############################################################################
#
# plot(team_rank.g,
#      layout = layout_with_fr(team_rank.g),
#      vertex.size = 10,
#      vertex.label.cex = 0.6,
#      vertex.color = V(team_rank.g)$color,
#      main = "Country-Level Network with Rank Groups")
#
```

```
# # Get Top 20 countrys by number of athletes
# top20_countrys <- athletes %>%
#   count(country, name = "num_athletes") %>%
#   arrange(desc(num_athletes)) %>%
#   slice(1:20) %>%
#   pull(country)
#
# # Identify which nodes are Rank Group nodes
# is_rank_group <- suppressWarnings(!is.na(as.numeric(V(team_rank.g)$country)))
#
# # Generate colors for each rank group (assuming max 6 groups)
# rank_colors <- brewer.pal(6, "Set2")
#
# # Create a vector to hold final colors
# V(team_rank.g)$color <- "gold"  # default for regular countrys
#
# # Color rank group nodes all the same (e.g., light gray)
# V(team_rank.g)$color[is_rank_group] <- "gray"
#
# # Apply red to top 20 countrys
# V(team_rank.g)$color[V(team_rank.g)$name %in% top20_countrys] <- "red"
```

```
# plot(team_rank.g,
#      layout = layout_with_fr(team_rank.g),
#      vertex.label = V(team_rank.g)$label,
#      vertex.color = V(team_rank.g)$color,
#      vertex.frame.color = "black",
#      vertex.size = 10,
#      vertex.label.cex = 0.5,
#      edge.color = "gray",
#      main = "Country-Level Network with Rank Groups and Top 20 countrys by Athlete Count Highlighte i
```

```
# bottom_countrys <- athletes_by_country %>%
#   slice_min(num_athletes, n = 20)
```

```
# #I asked Chat GPT for this.
# # Identify rank group nodes (still needed)
# is_rank_group <- suppressWarnings(!is.na(as.numeric(V(team_rank.g)$name)))
```

```
#
# # Default color: gold for countries
# V(team_rank.g)$color <- "gold"
#
# # Color rank group nodes all the same (e.g., light gray)
# V(team_rank.g)$color[is_rank_group] <- "gray"
#
# # Highlight bottom 20 countries (already shortened) in blue
# V(team_rank.g)$color[V(team_rank.g)$name %in% bottom_countrys$country] <- "red"
```

```
# plot(team_rank.g,
#      layout = layout_with_fr(team_rank.g),
#      vertex.label = V(team_rank.g)$label,
#      vertex.label.cex = 0.5,
#      vertex.color = V(team_rank.g)$color,
#      vertex.frame.color = "black",
#      vertex.size = 10,
#      edge.color = "gray",
#      main = "Country-Level Network with Rank Groups and Bottom 20 countrys by Athlete Count Highlight
```