CSSS 569 Visualizing Data and Models
Lab 7: Visualizing Network/Relational Data

Ramses Llobet

Department of Political Science, UW

February 18, 2022
The following packages are required for this lab:

```r
packages <- c("tidygraph", "ggraph", "reshape2", "cluster", "circlize")
install.packages(packages)
```
Introduction: Relational data

- Map of sciences (Bollen et al. 2009)
Introduction: Relational data

- Network data create many challenges for visualization.
Introduction: Relational data

- Network data create many challenges for visualization
  - Cursed by high dimensionality
Introduction: Relational data

- Network data create many challenges for visualization
  - Cursed by high dimensionality
  - Network diagrams usually result in hairballs or spaghetti balls...
Network data create many challenges for visualization

- Cursed by high dimensionality
- Network diagrams usually result in hairballs or spaghetti balls...
- The main takeaway of this lab is actually to seek alternative visualization methods whenever possible
Examples in today’s lab

- Florentine families and the rise of Medici: network diagram
Examples in today’s lab

- Global migration data: heat map
- Additional tricks: making NAs explicit; cluster analysis
Examples in today’s lab

- Global migration data: chord diagram
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
  - Epidemiology (e.g. disease transmission networks)
  - Economics (e.g. networks of interlocking directorates)
  - Sociology (e.g. networks of LGBT groups; social media)
  - Political science (e.g. political elite networks)

- In this lab, I want you to think more generically about relational data.
- More specifically, any data whose unit of observation is dyadic.
  - Examples: Migration flow data, or import/export data, between countries...
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
The science of networks is incredibly interdisciplinary:

- Computer science (e.g. World Wide Web)
- Biology (e.g. protein-protein interaction networks)
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
  - Epidemiology (e.g. disease transmission networks)
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
  - Epidemiology (e.g. disease transmission networks)
  - Economics (e.g. networks of interlocking directorates)
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
  - Epidemiology (e.g. disease transmission networks)
  - Economics (e.g. networks of interlocking directorates)
  - Sociology (e.g. networks of LGBT groups; social media)
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
  - Epidemiology (e.g. disease transmission networks)
  - Economics (e.g. networks of interlocking directorates)
  - Sociology (e.g. networks of LGBT groups; social media)
  - Political science (e.g. political elite networks)
The science of networks is incredibly interdisciplinary:

- Computer science (e.g. World Wide Web)
- Biology (e.g. protein-protein interaction networks)
- Engineering (e.g. electrical grid networks)
- Epidemiology (e.g. disease transmission networks)
- Economics (e.g. networks of interlocking directorates)
- Sociology (e.g. networks of LGBT groups; social media)
- Political science (e.g. political elite networks)

In this lab, I want you to think more generically about relational data.
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
  - Epidemiology (e.g. disease transmission networks)
  - Economics (e.g. networks of interlocking directorates)
  - Sociology (e.g. networks of LGBT groups; social media)
  - Political science (e.g. political elite networks)

- In this lab, I want you to think more generically about relational data.
  - More specifically, any data whose unit of observation is dyadic
Introduction: Relational data

- The science of networks is incredibly interdisciplinary:
  - Computer science (e.g. World Wide Web)
  - Biology (e.g. protein-protein interaction networks)
  - Engineering (e.g. electrical grid networks)
  - Epidemiology (e.g. disease transmission networks)
  - Economics (e.g. networks of interlocking directorates)
  - Sociology (e.g. networks of LGBT groups; social media)
  - Political science (e.g. political elite networks)

- In this lab, I want you to think more generically about relational data.
  - More specifically, any data whose unit of observation is dyadic
  - Examples: Migration flow data, or import/export data, between countries...
Introduction: Relational data

▶ Two basic elements:
Introduction: Relational data

- Two basic elements:
  - Nodes (or vertices)
Introduction: Relational data

- Two basic elements:
  - Nodes (or vertices)
  - Links (or edges)
Introduction: Relational data

- Two basic elements:
  - Nodes (or vertices)
  - Links (or edges)
- Two ways to represent relational data:
Introduction: Relational data

- Two basic elements:
  - Nodes (or vertices)
  - Links (or edges)
- Two ways to represent relational data:
  - Matrix (or adjacency matrix)
Introduction: Relational data

- Two basic elements:
  - Nodes (or vertices)
  - Links (or edges)

- Two ways to represent relational data:
  - Matrix (or adjacency matrix)
  - Long data frame (or edge list)
Introduction: Relational data

- Two basic elements:
  - Nodes (or vertices)
  - Links (or edges)
- Two ways to represent relational data:
  - Matrix (or adjacency matrix)
  - Long data frame (or edge list)
- Example with the marriage network of Florentine families
Undirected graph

Vertices = \{1, 2, 3\}
Edges = \{1, 2, 3, 1, 3, 2\}

Directed graph

Vertices = \{1, 2, 3\}
Edges = \{2, 1, 3, 1, 3\}
Example 1: Florentine families and the rise of Medici

- Marriage ties between Florentine families in early 15th century
  - From Padget & Ansell (1993)
Example 1: Florentine families and the rise of Medici

Represent relational data with matrix (or adjacency list)

<table>
<thead>
<tr>
<th></th>
<th>Acciaiuoli</th>
<th>Albizzi</th>
<th>Barbadori</th>
<th>Bischeri</th>
<th>Castellani</th>
<th>Ginori</th>
<th>Guadagni</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acciaiuoli</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Albizzi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Barbadori</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bischeri</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Castellani</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ginori</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Guadagni</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lamberteschi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Medici</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pazzi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Peruzzi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pucci</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ridolfi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Salviati</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Strozzi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tornabuoni</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Example 1: Florentine families and the rise of Medici

- Represent relational data with long data frame (or edge list)

```r
## [,1] [,2]
## [1,] "Acciaiuoli" "Medici"
## [2,] "Albizzi" "Ginori"
## [3,] "Albizzi" "Guadagni"
## [4,] "Albizzi" "Medici"
## [5,] "Barbadori" "Castellani"
## [6,] "Barbadori" "Medici"
## [7,] "Bischeri" "Guadagni"
## [8,] "Bischeri" "Peruzzi"
## [9,] "Bischeri" "Strozzi"
## [10,] "Castellani" "Peruzzi"
## [11,] "Castellani" "Strozzi"
## [12,] "Guadagni" "Lamberteschi"
## [13,] "Guadagni" "Tornabuoni"
## [14,] "Medici" "Ridolfi"
## [15,] "Medici" "Salviati"
## [16,] "Medici" "Tornabuoni"
## [17,] "Pazzi" "Salviati"
## [18,] "Peruzzi" "Strozzi"
## [19,] "Ridolfi" "Strozzi"
## [20,] "Ridolfi" "Tornabuoni"
```
Example 1: Florentine families and the rise of Medici

```r
# install.packages(c("tidygraph", "ggraph"))
library(tidyverse)
library(tidygraph)
library(ggraph)

# Load data (from Chris's website::lab section)
medici <- read.table("data/medici.txt")
medici <- as.matrix(medici)
```
Example 1: Florentine families and the rise of Medici

First, we have to turn our matrix into a tidygraph object

```r
medici_graph <- as_tbl_graph(medici, directed = FALSE)
```
Example 1: Florentine families and the rise of Medici

First, we have to turn our matrix into a tidygraph object

```r
## # A tbl_graph: 16 nodes and 20 edges
## #
## # An undirected simple graph with 2 components
## #
## # Node Data: 16 x 1 (active)
## # name
## # <chr>
## # 1 Acciaiuoli
## # 2 Albizzi
## # 3 Barbadori
## # 4 Bischeri
## # 5 Castellani
## # 6 Ginori
## # ... with 10 more rows
## #
## # Edge Data: 20 x 3
## # from to weight
## # <int> <int> <dbl>
## # 1  1  9  1
## # 2  2  6  1
## # 3  2  7  1
## # ... with 17 more rows
```
Example 1: Florentine families and the rise of Medici

- Visualize network data using `ggraph` package

```r
ggraph(medici_graph) + geom_node_point()
```
Example 1: Florentine families and the rise of Medici

- Visualize network data using `ggraph` package

```r
ggraph(medici_graph) + geom_node_point() + geom_edge_link()
```
Example 1: Florentine families and the rise of Medici

► Visualize network data using ggraph package

ggraph(medici_graph) +
  geom_node_point() +
  geom_edge_link() +
  geom_node_text(aes(label = name), repel = TRUE)
Example 1: Florentine families and the rise of Medici

▶ Visualize network data using ggraph package

```r
library(ggraph)

# Load data
medici_graph <- read_graph("")

# Visualize network
medici_graph <- ggraph(medici_graph) +
  geom_node_point() +
  geom_edge_link() +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_graph()
```

![Network Diagram]
Example 1: Florentine families and the rise of Medici

Create new network measures using tidygraph

```r
medici_graph <-
  medici_graph %>%
  mutate(
    # Calculate degree centrality
    degree = centrality_degree(),
    # Implement community-detection algorithm
    community = group_edge_betweenness()
  )
```
Example 1: Florentine families and the rise of Medici

▶ Create new network measures using tidygraph

```r
## # A tbl_graph: 16 nodes and 20 edges
## #
## # An undirected simple graph with 2 components
## #
## # Node Data: 16 x 3 (active)
## name       degree community
## <chr>      <dbl>     <int>
## 1 Acciaiuoli 1         2
## 2 Albizzi    3         3
## 3 Barbadori  2         1
## 4 Bischeri   3         1
## 5 Castellani 3         1
## 6 Ginori     1         3
## # ... with 10 more rows
## #
## # Edge Data: 20 x 3
## from to weight
## <int> <int>  <dbl>
## 1   1  9    1
## 2   2  6    1
## 3   2  7    1
## # ... with 17 more rows
```
Example 1: Florentine families and the rise of Medici

- Incorporate new network measures into our visualization

```r
gggraph(medici_graph) +
gem_node_point(aes(size = degree), show.legend = FALSE) +
gem_edge_link() +
gem_node_text(aes(label = name), repel = TRUE) +
theme_graph()
```
Example 1: Florentine families and the rise of Medici

Incorporate new network measures into our visualization

```r
library(ggraph)

# Load the Medici graph
medici_graph <- read_graph('medici_graph.csv')

# Incorporate new network measures

ggraph(medici_graph) +
gem_node_point(aes(size = degree, color = factor(community)),
              show.legend = FALSE) +
gem_edge_link() +
gem_node_text(aes(label = name), repel = TRUE) +
theme_graph()
```

![Graph showing Florentine families and the rise of Medici](image)
Example 1: Florentine families and the rise of Medici

Incorporate new network measures into our visualization

```r
library(ggraph)

# Load the Medici graph data
medici_graph <- read_graph(...)

# Plot the network with new measures
ggraph(medici_graph) +
  geom_edge_link() +
  geom_node_point(aes(size = degree, color = factor(community)),
                  show.legend = FALSE) +
  geom_node_text(aes(label = name), repel = TRUE) +
  scale_color_brewer(palette = "Set1") +
  theme_graph()
```

Diagram showing the connections between various Florentine families, such as Acciaiuoli, Albizzi, Bischeri, Castellani, Barbadori, Bichi, Pucci, Ridolfi, Salviati, Tornabuoni, and Strozzi.
Example 1: Florentine families and the rise of Medici

Save the output

```r
width = 7
ggsave("output/medici.pdf", width = width, height = width/2)
```
Example 2: Global migration flow data

Heat map
Example 2: Global migration flow data

- Chord diagram
Example 2: Global migration flow data

▶ Original data are from Abel (2018)

migrat2010 <- read_csv("data/migrat2010.csv")

head(migrat2010)

## # A tibble: 6 x 3
## # origRegion destRegion  flow
## # <chr>     <chr>      <dbl>
## 1 Caribbean Caribbean  40506
## 2 Caribbean Central America  8183
## 3 Caribbean Northern America  533052
## 4 Caribbean Northern Europe  15584
## 5 Caribbean South America  3264
## 6 Caribbean Southern Europe  21711
Example 2: Global migration flow data

Network diagram doesn’t work well here...

```r
migrat2010_graph <- as_tbl_graph(migrat2010)

ggraph(migrat2010_graph)+
  geom_edge_link(alpha = 0.5) +
  geom_node_point() +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_graph()
```
Example 2: Global migration flow data

- Worse still, the data is actually bidirectional, which means you need to visualize **two edges** for each dyadic pair.

```r
library(ggraph)

migrat2010_graph <-READ(GRAPHPATH)

ggraph(migrat2010_graph) +
  geom_edge_parallel(start_cap = circle(1.25, 'mm'),
                    end_cap = circle(1.25, 'mm'),
                    arrow = arrow(length = unit(2, 'mm')),
                    sep = unit(1.25, 'mm'), alpha = 0.5) +
  geom_node_point() +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_graph()
```
Example 2: Global migration flow data

- Two alternative visualization methods:
Example 2: Global migration flow data

- Two alternative visualization methods:
  - Heatmap
Example 2: Global migration flow data

- Two alternative visualization methods:
  - Heatmap
  - Chord diagram
Example 2: Global migration flow data :: Heatmap

Like what we did in lab 4 using `geom_tile()`

```r
migrat2010 %>%
ggplot(aes(y = origRegion, x = destRegion, fill = flow)) +
geom_tile(color = "white", size = 0.2) +
coord_equal() +
theme(panel.background = element_blank())
```
Example 2: Global migration flow data :: Heatmap

How can we improve the heat map?

Try to replicate the following example:
Example 2: Global migration flow data :: Heatmap

- **Main tasks:**
  - Make NA values explicit
  - Turn flow into a categorical variable
  - Cluster analysis and sorting

<table>
<thead>
<tr>
<th>Destination</th>
<th>Origin</th>
<th>Migration flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Southern Asia</td>
<td>Southern Asia</td>
<td>&gt;100000</td>
</tr>
<tr>
<td>Western Asia</td>
<td>Southern Asia</td>
<td>&gt;100000</td>
</tr>
<tr>
<td>South-Eastern Asia</td>
<td>South-Eastern Asia</td>
<td>50000−100000</td>
</tr>
<tr>
<td>Central America</td>
<td>Central America</td>
<td>10000−50000</td>
</tr>
<tr>
<td>Eastern Asia</td>
<td>Central America</td>
<td>5000−10000</td>
</tr>
<tr>
<td>Western Africa</td>
<td>Central America</td>
<td>1000−5000</td>
</tr>
<tr>
<td>Eastern Africa</td>
<td>Western Africa</td>
<td></td>
</tr>
<tr>
<td>Northern Africa</td>
<td>Northern Africa</td>
<td></td>
</tr>
<tr>
<td>Caribbean</td>
<td>Northern Africa</td>
<td></td>
</tr>
<tr>
<td>South America</td>
<td>Northern America</td>
<td></td>
</tr>
<tr>
<td>Northern Europe</td>
<td>Northern Europe</td>
<td></td>
</tr>
<tr>
<td>Northern America</td>
<td>Middle Africa</td>
<td></td>
</tr>
<tr>
<td>Middle Africa</td>
<td>Oceania</td>
<td></td>
</tr>
<tr>
<td>Oceania</td>
<td>Southern Africa</td>
<td></td>
</tr>
<tr>
<td>Southern Africa</td>
<td>Central Asia</td>
<td></td>
</tr>
<tr>
<td>Central Asia</td>
<td>Eastern Europe</td>
<td></td>
</tr>
<tr>
<td>Eastern Europe</td>
<td>Southern Europe</td>
<td></td>
</tr>
<tr>
<td>Southern Europe</td>
<td>Western Europe</td>
<td></td>
</tr>
<tr>
<td>Western Europe</td>
<td>Western Europe</td>
<td></td>
</tr>
</tbody>
</table>
Example 2: Global migration flow data :: Heatmap

To make NA values explicit, use `expand()` and `left_join()` from tidyverse

```r
migrat2010 %>%
  expand(origRegion, destRegion) %>%
  head()
```

```r
## # A tibble: 6 x 2
## #  origRegion destRegion
## <chr>        <chr>
## 1 Caribbean   Caribbean
## 2 Caribbean   Central America
## 3 Caribbean   Central Asia
## 4 Caribbean   Eastern Africa
## 5 Caribbean   Eastern Asia
## 6 Caribbean   Eastern Europe
```
Example 2: Global migration flow data :: Heatmap

▶ To make NA values explicit, use expand() and left_join() from tidyverse
  ▶ expand() creates all unique combinations of two (or more) variables

migrat2010 %>%
  expand(origRegion, destRegion) %>%
  head()

```r
## # A tibble: 6 x 2
## #  origRegion destRegion
## #  <chr>      <chr>
## 1 Caribbean  Caribbean
## 2 Caribbean  Central America
## 3 Caribbean  Central Asia
## 4 Caribbean  Eastern Africa
## 5 Caribbean  Eastern Asia
## 6 Caribbean  Eastern Europe
```
Example 2: Global migration flow data :: Heatmap

- To make NA values explicit, use `expand()` and `left_join()` from tidyverse

```r
migrat2010 <-
migrat2010 %>%
extend(OrigRegion, DestRegion) %>%
left_join(migrat2010, by = c("OrigRegion", "DestRegion"))

head(migrat2010)
```

```r
## A tibble: 6 x 3
## origRegion destRegion  flow
## <chr>      <chr>     <dbl>
## 1 Caribbean Caribbean 40506
## 2 Caribbean Central America 8183
## 3 Caribbean Central Asia  NA
## 4 Caribbean Eastern Africa  NA
## 5 Caribbean Eastern Asia  NA
## 6 Caribbean Eastern Europe  NA
```
Example 2: Global migration flow data :: Heatmap

- To make NA values explicit, use `expand()` and `left_join()` from tidyverse
- Use the result from `expand()` as a template for `left_join()`

```r
migrat2010 <-
migrat2010 %>%
  expand(origRegion, destRegion) %>%
  left_join(migrat2010, by = c("origRegion", "destRegion"))

head(migrat2010)

## # A tibble: 6 x 3
## #  origRegion destRegion flow
## # <chr>      <chr>     <dbl>
## 1 Caribbean Caribbean 40506
## 2 Caribbean Central America 8183
## 3 Caribbean Central Asia  NA
## 4 Caribbean Eastern Africa NA
## 5 Caribbean Eastern Asia  NA
## 6 Caribbean Eastern Europe NA
```
Example 2: Global migration flow data :: Heatmap

Turn flow into a categorical variable

```r
quantile(migrat2010$flow, na.rm = TRUE)

# Create breaks and labels
breaks <- c(1000, 5000, 10000, 50000, 100000, Inf)
labels <- c("1000-5000", "5000-10000", "10000-50000",
            "50000-100000", ">100000")

# Create a new variable `flowCat`
migrat2010 <-
migrat2010 %>%
  mutate(flowCat = cut(flow, breaks, labels))
```
Example 2: Global migration flow data :: Heatmap

- Turn flow into a categorical variable

```r
head(migrat2010)
```

```r
## # A tibble: 6 x 4
## #  origRegion destRegion  flow flowCat
## #  <chr>    <chr>   <dbl> <fct>
##  1 Caribbean Caribbean  40506 10000-50000
##  2 Caribbean Central America 8183 5000-10000
##  3 Caribbean Central Asia NA <NA>
##  4 Caribbean Eastern Africa NA <NA>
##  5 Caribbean Eastern Asia NA <NA>
##  6 Caribbean Eastern Europe NA <NA>
```
Example 2: Global migration flow data :: Heatmap

▶ Clustering analysis and sorting

```r
# Load packages
library(reshape2)
library(cluster)

# Convert long data frame into a full matrix
migrat2010_matrix <-
    migrat2010 %>%
    reshape2::acast(origRegion ~ destRegion, value.var = "flow", fill = 0)
```
Example 2: Global migration flow data :: Heatmap

- Clustering analysis and sorting

```r
# Convert long data frame into a full matrix
print(migrat2010_matrix[1:4, 1:4])
```

<table>
<thead>
<tr>
<th></th>
<th>Caribbean</th>
<th>Central America</th>
<th>Central Asia</th>
<th>Eastern Africa</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Caribbean</strong></td>
<td>40506</td>
<td>8183</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Central America</strong></td>
<td>0</td>
<td>99171</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Central Asia</strong></td>
<td>0</td>
<td>0</td>
<td>77252</td>
<td>0</td>
</tr>
<tr>
<td><strong>Eastern Africa</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>444352</td>
</tr>
</tbody>
</table>
Cluster analysis and sorting

```r
migrat2010_hclust <-
  dist(migrat2010_matrix) %>%
  hclust(method = "ward.D")  # Several other methods are available

countryOrder <- migrat2010_hclust$order
print(countryOrder)
```

```r
## [1] 15 18 12  2  5 17  4  8  1 13 10  9  7 11 14  3  6 16 19
```
Example 2: Global migration flow data :: Heatmap

Clustering analysis and sorting

```
# Sort the countries using the order produced by cluster analysis
countryLevels <- row.names(migrat2010_matrix)[countryOrder]

print(countryLevels)
```

```r
## [1] "Southern Asia"      "Western Asia"      "South-Eastern Asia"
## [4] "Central America"     "Eastern Asia"       "Western Africa"
## [7] "Eastern Africa"      "Northern Africa"   "Caribbean"
## [10] "South America"       "Northern Europe"   "Northern America"
## [13] "Middle Africa"       "Oceania"           "Southern Africa"
## [16] "Central Asia"        "Eastern Europe"    "Southern Europe"
## [19] "Western Europe"
```
Example 2: Global migration flow data :: Heatmap

- Clustering analysis and sorting

```r
# Re-level `origRegion` and `destRegion` according to the level
migrat2010 <- migrat2010 %>%
  mutate(
    origRegion = factor(origRegion, levels = rev(countryLevels)),
    destRegion = factor(destRegion, levels = countryLevels)
  )
```
Example 2: Global migration flow data :: Heatmap

► Visualize the heat map again:

```r
migrat2010 %>%
ggplot(aes(y = origRegion, x = destRegion, fill = flowCat)) +
geom_tile(color = "white", size = 0.2) +
# Scale fill values with "Blues" palette and "grey90" for NAs
scale_fill_brewer(palette = "Blues", na.value = "grey90",
                   breaks = rev(labels)) +
# Put x-axis labels on top
scale_x_discrete(position = "top") +
coord_equal() +
theme(panel.background = element_blank(),
      axis.ticks.x = element_blank(),
      axis.ticks.y = element_blank(),
      # Rotate and align x-axis labels
      axis.text.x.top = element_text(angle = 90, hjust = 0),
      legend.key.height = grid::unit(0.8, "cm"),
      legend.key.width = grid::unit(0.2, "cm")
    ) +
guides(fill = guide_legend(title = "Migration flow")) +
labs(y = "Origin", x = "Destination")
```
Example 2: Global migration flow data :: Heatmap

Visualize the heat map again:

Destination

<table>
<thead>
<tr>
<th>Origin</th>
<th>Southern Asia</th>
<th>Western Asia</th>
<th>South–Eastern Asia</th>
<th>Central America</th>
<th>Eastern Asia</th>
<th>Western Africa</th>
<th>Eastern Africa</th>
<th>Northern Africa</th>
<th>Caribbean</th>
<th>South America</th>
<th>Northern Europe</th>
<th>Middle Africa</th>
<th>North America</th>
<th>Northern America</th>
<th>Southern Africa</th>
<th>Central Asia</th>
<th>Eastern Europe</th>
<th>Southern Europe</th>
<th>Western Europe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Southern Asia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Western Asia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>South–Eastern Asia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Central America</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eastern Asia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Western Africa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eastern Africa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northern Africa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caribbean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>South America</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northern Europe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northern America</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middle Africa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oceania</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Southern Africa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Central Asia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eastern Europe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Southern Europe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Western Europe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Migration flow

>100000
50000–100000
10000–50000
5000–10000
1000–5000
Example 2: Global migration flow data :: Chord diagram

- Chord diagram has become growingly popular
Example 2: Global migration flow data :: Chord diagram

- We'll use the circlize package: full documentation [here](#)

```r
library(circlize)
```
Example 2: Global migration flow data :: Chord diagram

But first, we have to aggregate regions and further reduce the dimensionality

# Create vectors of countries to be aggregated
Europe <- c("Southern Europe", "Western Europe", "Northern Europe")
EECA <- c("Eastern Europe", "Central Asia")
Africa <- c("Eastern Africa", "Middle Africa", "Northern Africa", "Southern Africa", "Western Africa")
LACarib <- c("Central America", "South America", "Caribbean")
SAsia <- c("South-Eastern Asia", "Southern Asia")

# Use mutate() and across() to recode
# `origRegion` and `destRegion` simultaneously
migrat2010 <-
migrat2010 %>%
  mutate(across(c(origRegion, destRegion),
               ~ case_when(
                   .x %in% Europe ~ "Europe",
                   .x %in% EECA ~ "Eastern Europe \n& Central Asia",
                   .x %in% Africa ~ "Africa",
                   .x %in% LACarib ~ "Latin America \n& Caribbean",
                   .x %in% SAsia ~ "Southern Asia",
                   TRUE ~ as.character(.x)
               ))
)
Example 2: Global migration flow data :: Chord diagram

But first, we have to aggregate regions and further reduce the dimensionality

```r
# Collapse (sum) flow values according by newly aggregated regions
migrat2010 <- migrat2010 %>%
  group_by(origRegion, destRegion) %>%
  summarize(flowTotal = sum(flow, na.rm = TRUE)) %>%
  ungroup()

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

head(migrat2010)
```

## # A tibble: 6 x 3
## #  origRegion destRegion     flowTotal
## #  <chr>      <chr>          <dbl>
## 1 Africa     "Africa"       3412806
## 2 Africa     "Eastern Asia" 1083
## 3 Africa     "Eastern Europe \n& Central Asia" 14504
## 4 Africa     "Europe"       1634143
## 5 Africa     "Latin America \n& Caribbean"   10694
## 6 Africa     "Northern America" 813775
Example 2: Global migration flow data :: Chord diagram

▶ Basic chord diagram

chordDiagram(migrat2010)
Example 2: Global migration flow data :: Chord diagram

► Advanced chord diagram settings (based on Abel’s GitHub)

```r
# Setting parameters
circos.clear()
circos.par(
    start.degree = 90,        # Start at 12 o'clock
    gap.degree = 4,           # Increase gaps between sectors
    track.margin = c(-0.1, 0.1),  # Narrow the track margin
    points.overflow.warning = FALSE,    # Subdue warning messages
)
par(mar = rep(0, 4))       # no margins in the plot
```
Example 2: Global migration flow data :: Chord diagram

Advanced chord diagram settings (based on Abel’s GitHub)

```r
# Get nice colors
colors <- RColorBrewer::brewer.pal(9, "Paired")

# More advanced settings in `chordDiagram()`
chordDiagram(migrat2010,
    # Set colors
grid.col = colors,
    # Indicate chords are directional
directional = 1,
    # Directionality is illustrated by arrows and height differences
direction.type = c("arrows", "diffHeight"),
    # Set height difference
diffHeight = -0.04,
    # Use big arrows
link.arr.type = "big.arrow",
    # Sort the chords and plot the smallest chords first
link.sort = TRUE, link.largest.ontop = TRUE,
)

# Save the output
dev.copy2pdf(file = "output/migratChord.pdf", height = 8, width = 8)
```
Example 2: Global migration flow data :: Chord diagram

Final output

[Image of a chord diagram showing migration flows between different regions such as Africa, Eastern Asia, Eastern Europe & Central Asia, Europe, Latin America & Caribbean, Northern America, Oceania, Southern Asia, and Western Asia. The diagram uses colors and lines to represent the magnitude of migration flows.]