CSSS 569 Visualizing Data and Models

Lab 7: Visualizing Network/Relational Data

Brian Leung

Department of Political Science, UW

December 29, 2021

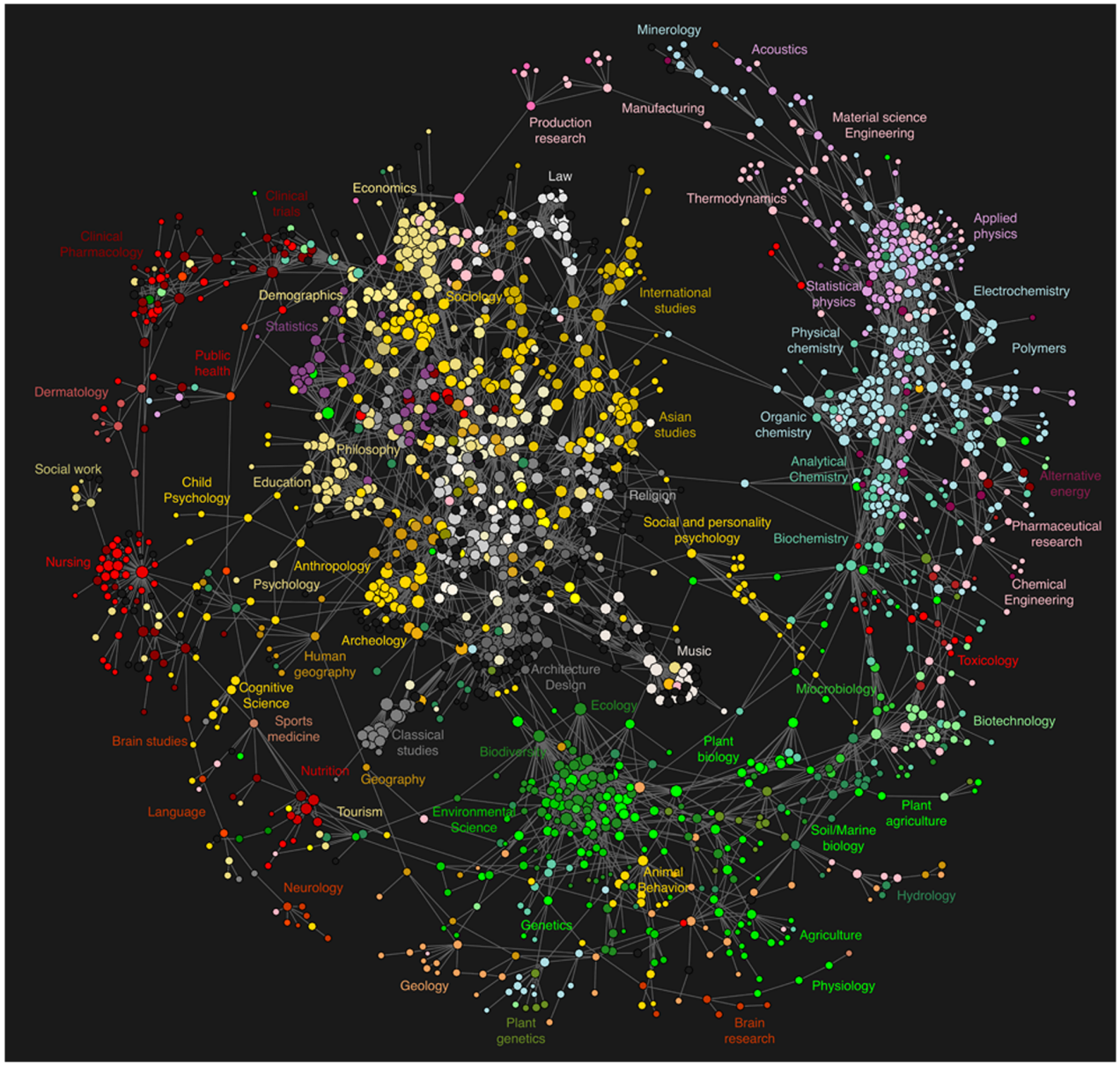
Prerequisite

- The following packages are required for this lab:

```
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

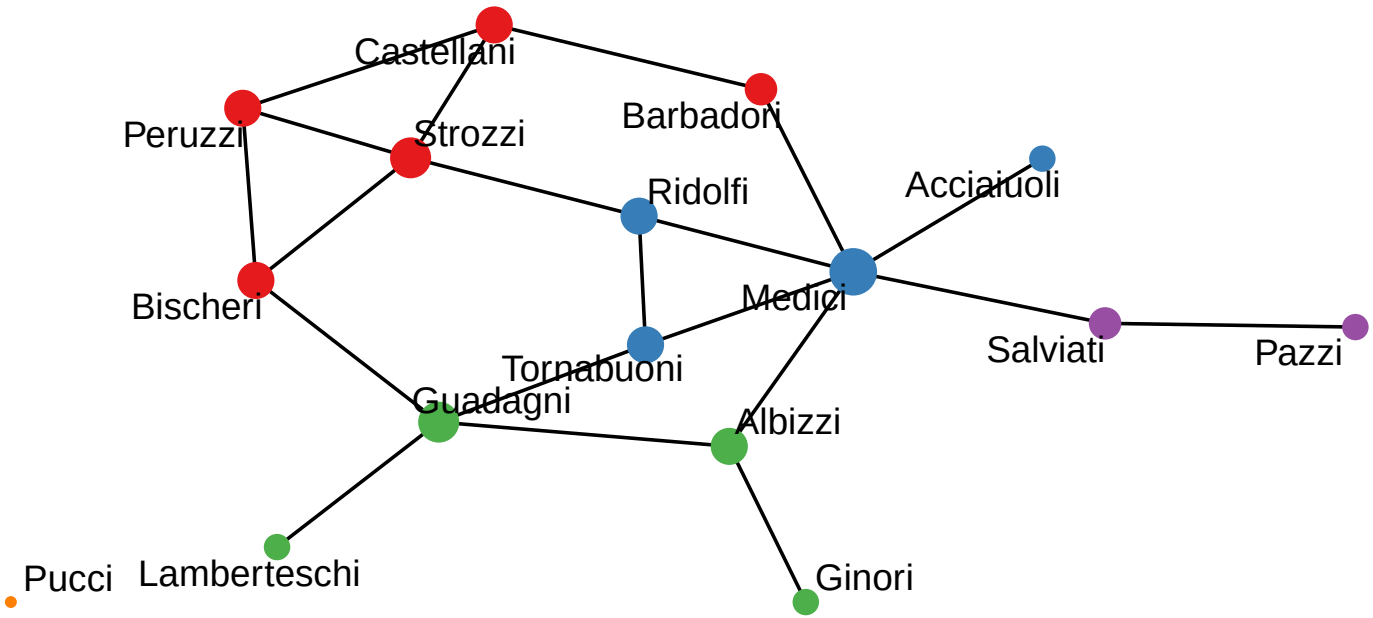
- Network data create many challenges for visualization
  - Cursed by high dimensionality
  - Network diagrams usually result in hairballs or spaghetti balls...

Introduction: Relational data

- 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

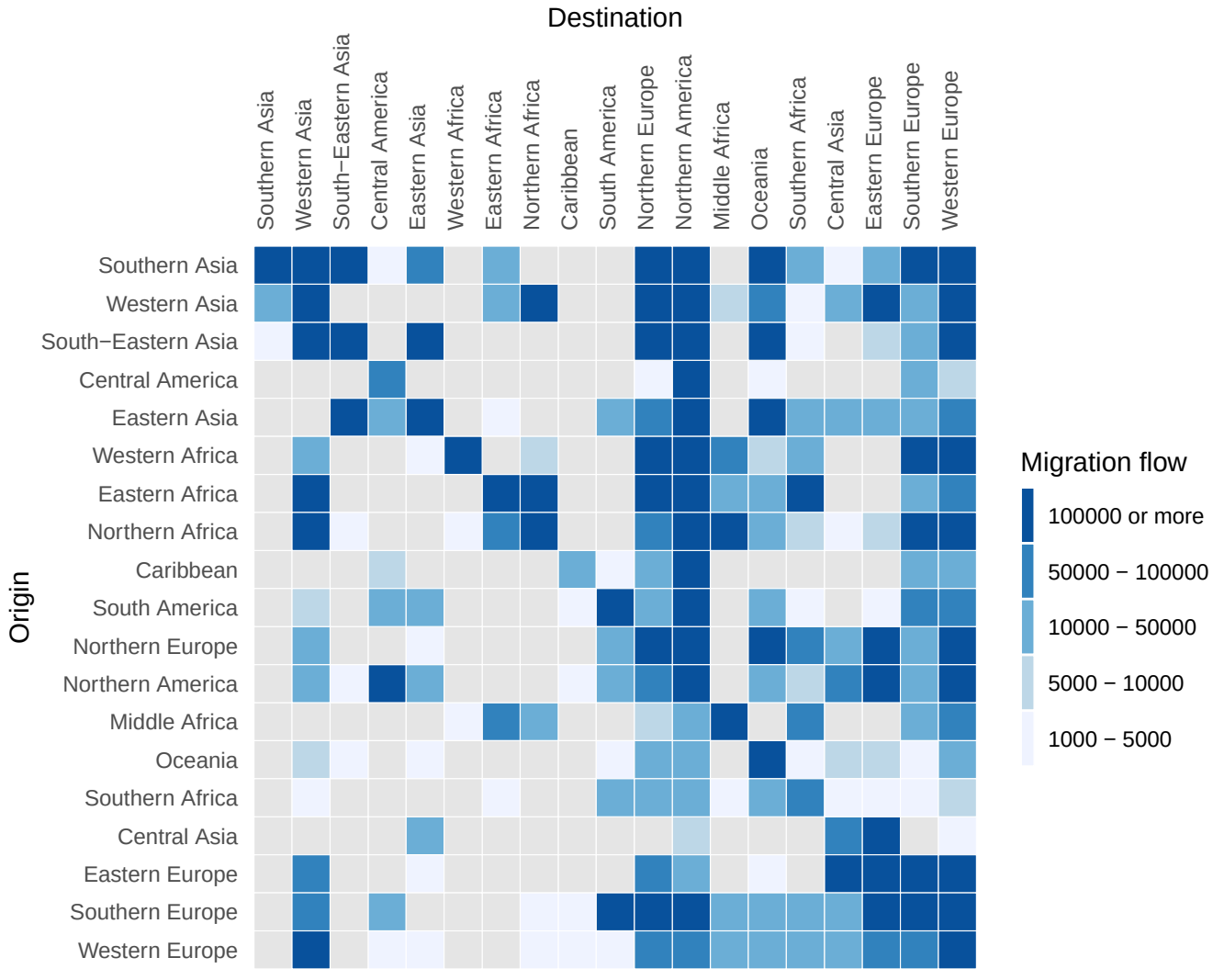
Examples in today's lab

- Florentine families and the rise of Medici: network diagram



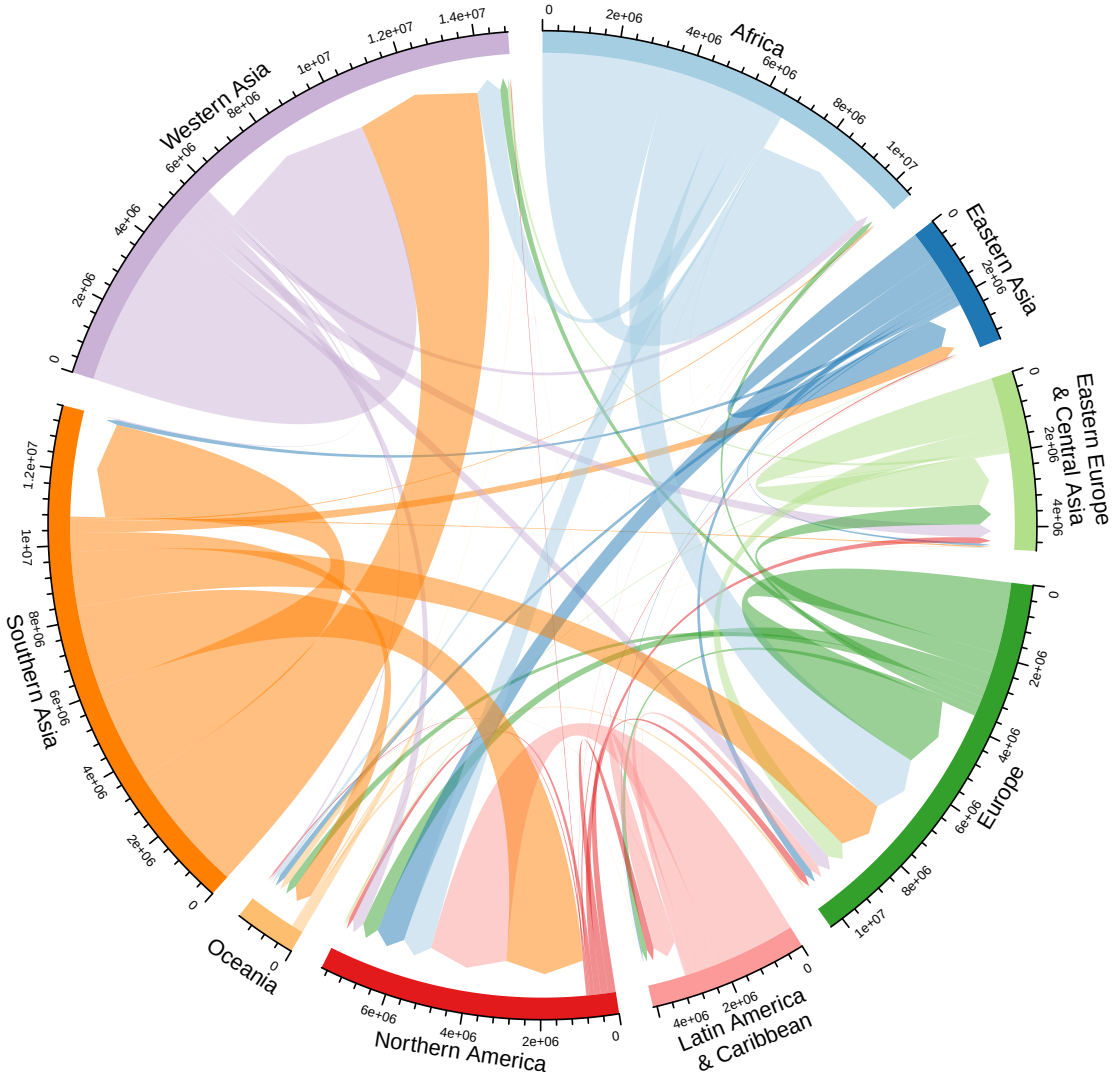
Examples in today's lab

- Global migration data: heat map
  - Additional tricks: making `NA`s explicit; cluster analysis



Examples in today's lab

- Global migration data: chord diagram



Introduction: Relational data

- The science of networks is incredibly interdisciplinary:

Introduction: Relational data

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

Introduction: Relational data

- 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)

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 genericalyl 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 genericalyl 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 genericalyl 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

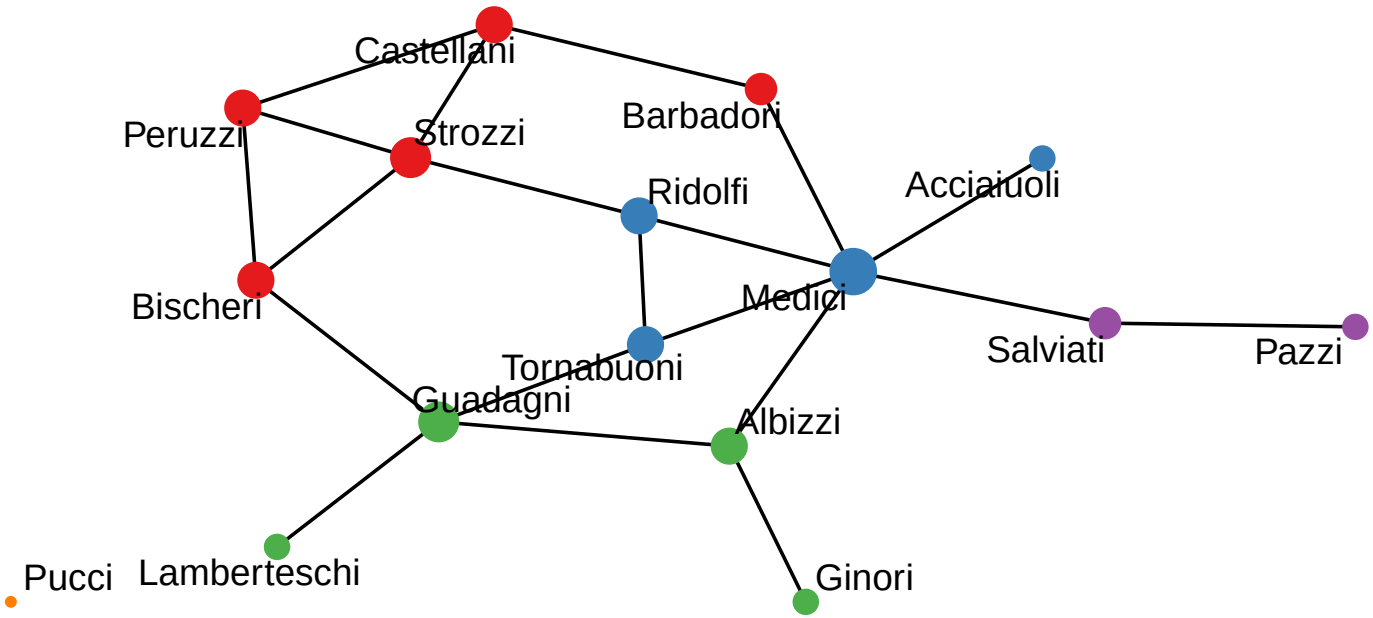
- Two basic elements:
  - Nodes (or vertices)
  - Links (or edges)
- Two ways to represent relational data:
  - Matrix (or adjacency matrix)
  - Long data frame(or edge list)

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

Example 1: Florentine families and the rise of Medici

- Marriage ties between Florentine familes in early 15th century
  - From Padget & Ansell (1993)



Example 1: Florentine families and the rise of Medici

- Represent relational data with matrix (or adjacency matrix)

```
##              Acciaiuoli Albizzi Barbadori Bischeri Castellani Ginori Guadagni
## Acciaiuoli            0       0          0        0          0      0        0
## Albizzi               0       0          0        0          0      1        1
## Barbadori             0       0          0        0          1      0        0
## Bischeri              0       0          0        0          0      0        1
## Castellani            0       0          1        0          0      0        0
## Ginori                0       1          0        0          0      0        0
## Guadagni              0       1          0        1          0      0        0
## Lamberteschi          0       0          0        0          0      0        1
## Medici                1       1          1        0          0      0        0
## Pazzi                 0       0          0        0          0      0        0
## Peruzzi               0       0          0        1          1      0        0
## Pucci                 0       0          0        0          0      0        0
## Ridolfi               0       0          0        0          0      0        0
## Salviati              0       0          0        0          0      0        0
## Strozzi               0       0          0        1          1      0        0
## Tornabuoni            0       0          0        0          0      0        1
```

Example 1: Florentine families and the rise of Medici

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

```
##       [,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

```
medici_graph <- as_tbl_graph(medici, directed = FALSE)
```

Example 1: Florentine families and the rise of Medici

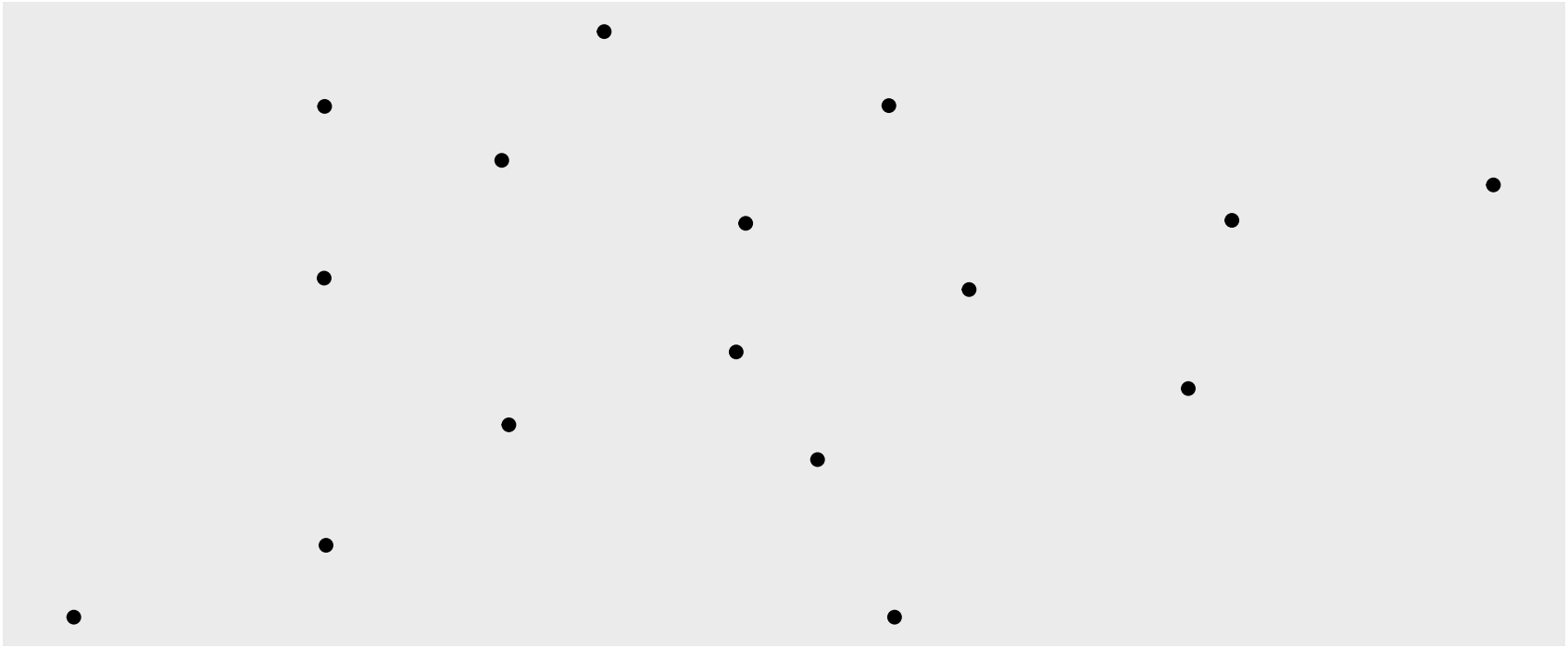
- First, we have to turn our matrix into a `tidygraph` object

```
## # 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

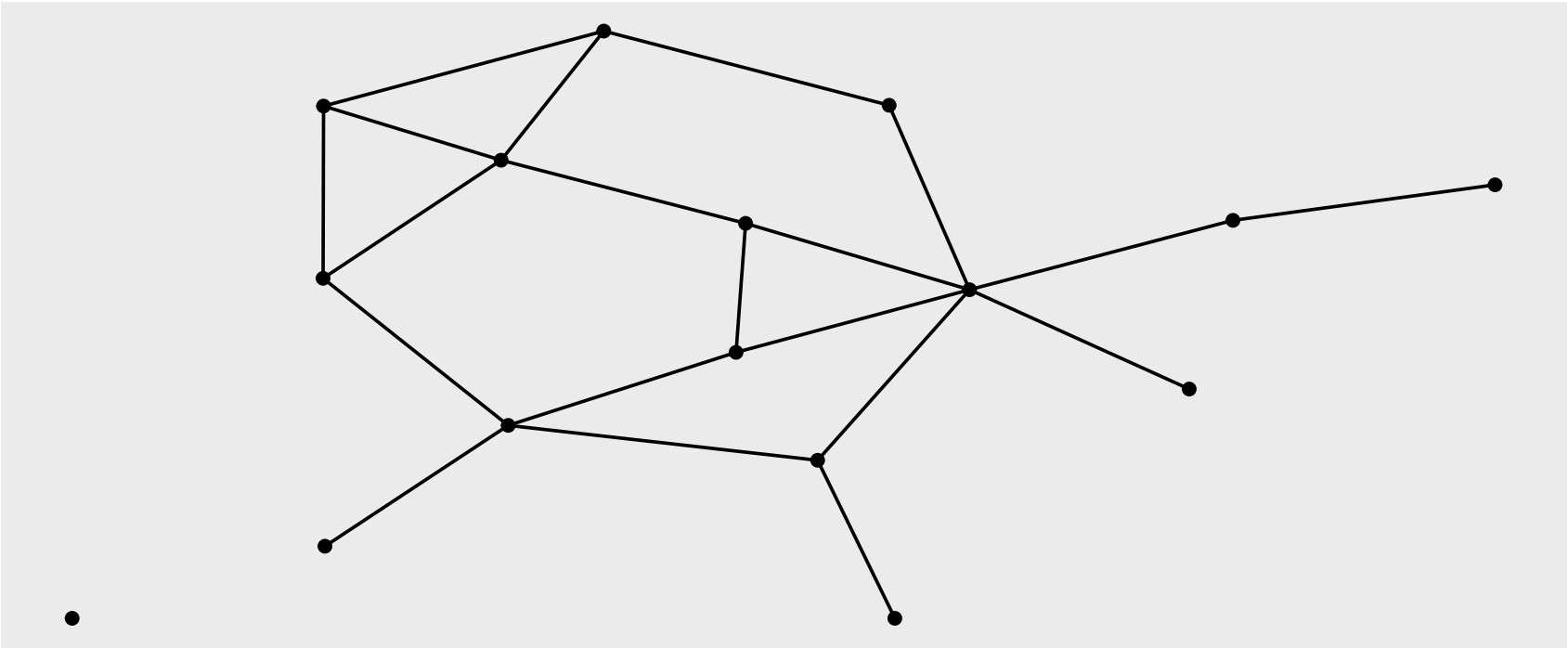
```
ggraph(medici_graph) +
  geom_node_point()
```



Example 1: Florentine families and the rise of Medici

- Visualize network data using ggraph package

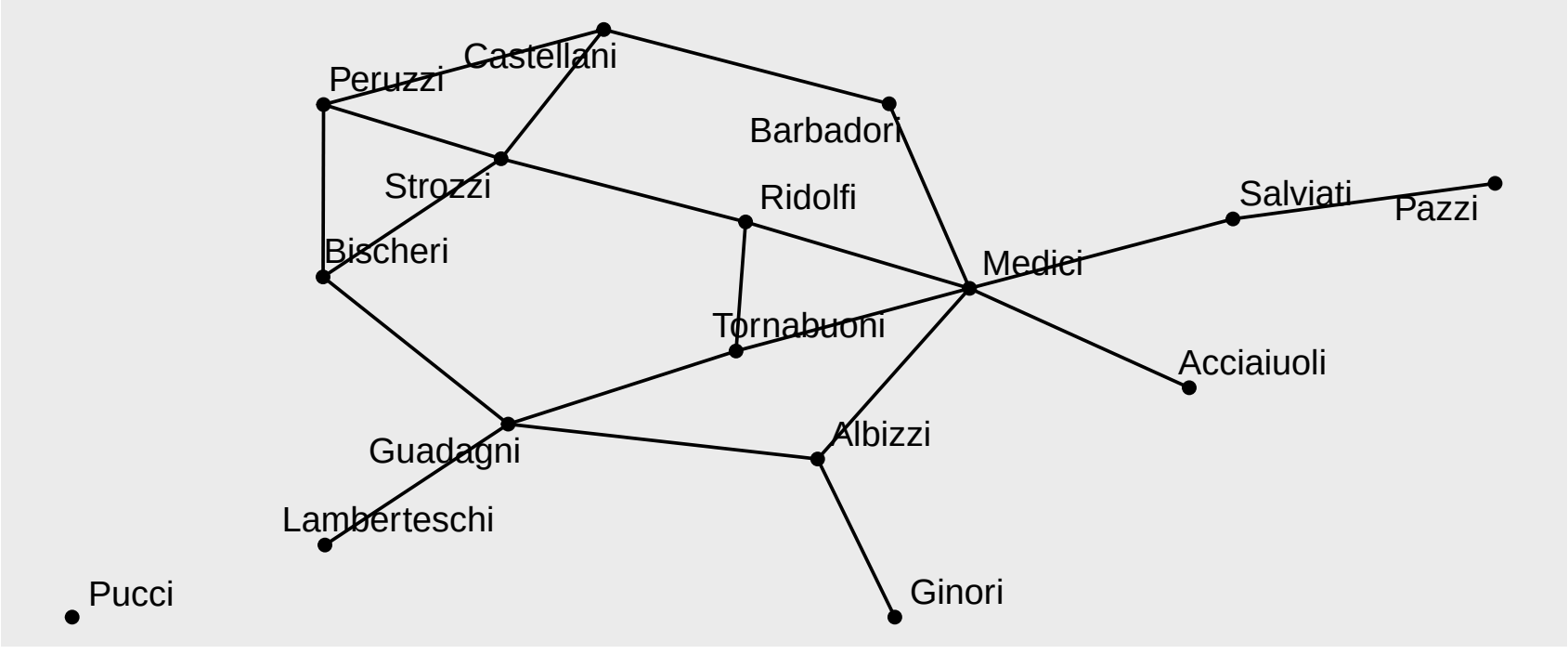
```
ggraph(medici_graph) +
  geom_node_point() +
  geom_edge_link()
```



Example 1: Florentine families and the rise of Medici

- Visualize network data using `ggraph` package

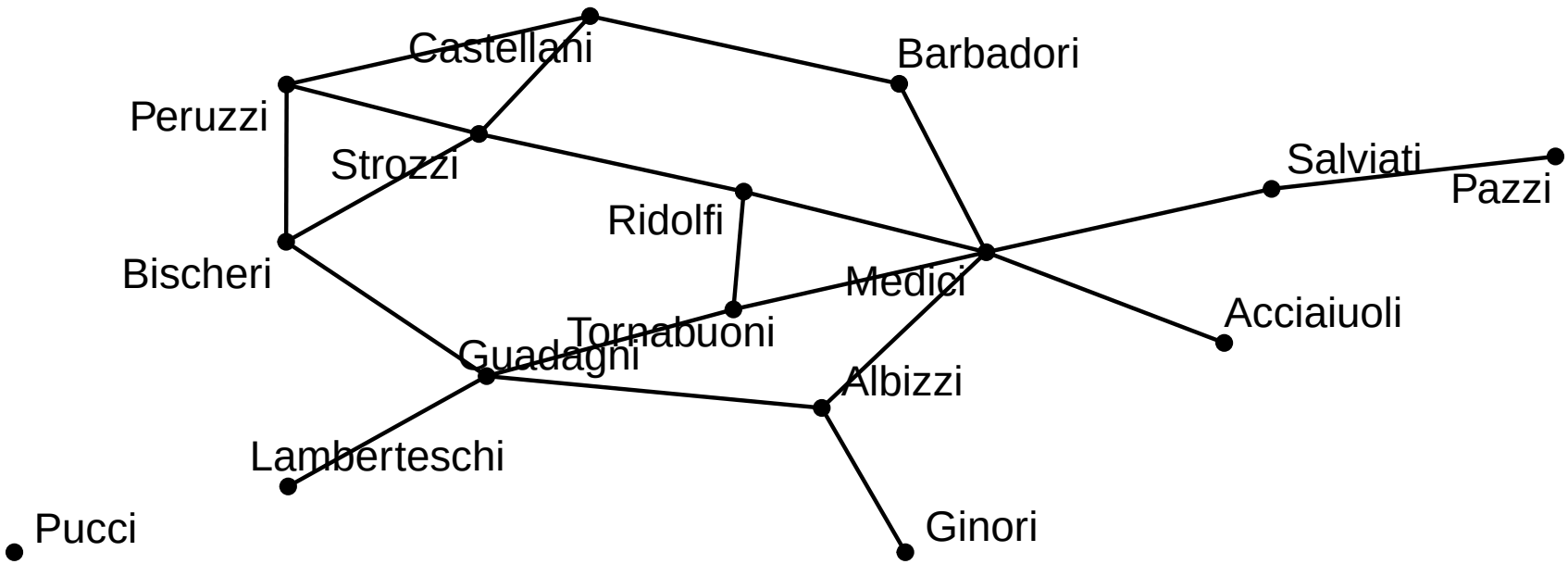
```
ggraph(medici_graph) +
  geom_node_point() +
  geom_edge_link() +
  geom_node_text(aes(label = name), repel = TRUE)
```



Example 1: Florentine families and the rise of Medici

- Visualize network data using ggraph package

```
ggraph(medici_graph) +
  geom_node_point() +
  geom_edge_link() +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_graph()
```



Example 1: Florentine families and the rise of Medici

- Create new network measures using `tidygraph`

```
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

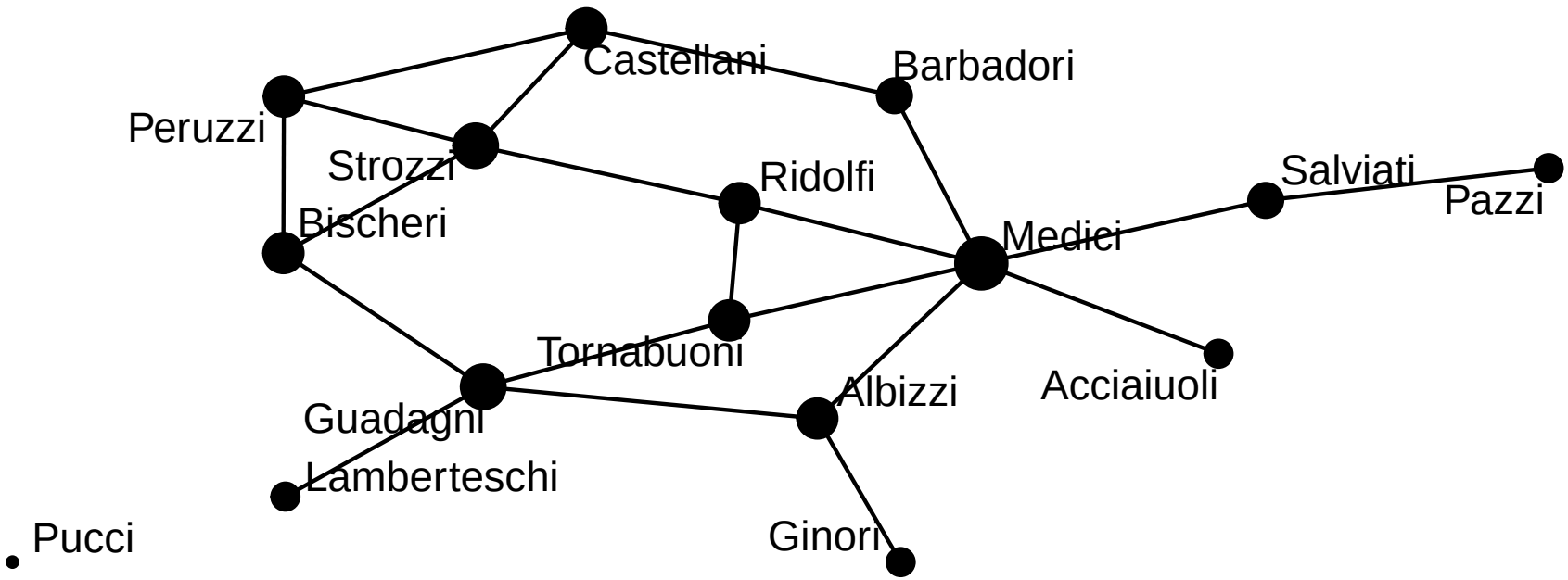
- Create new network measures using `tidygraph`

```
## # 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

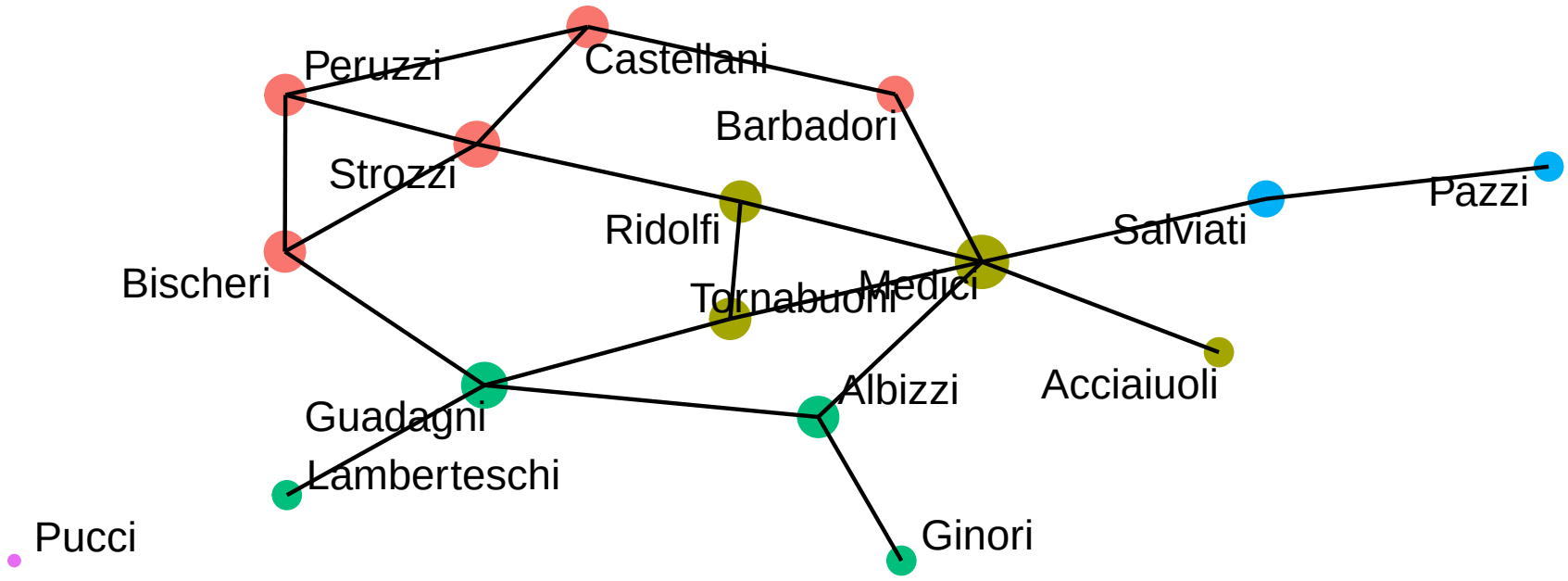
```
ggraph(medici_graph) +
  geom_node_point(aes(size = degree), show.legend = FALSE) +
  geom_edge_link() +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_graph()
```



Example 1: Florentine families and the rise of Medici

- Incorporate new network measures into our visualization

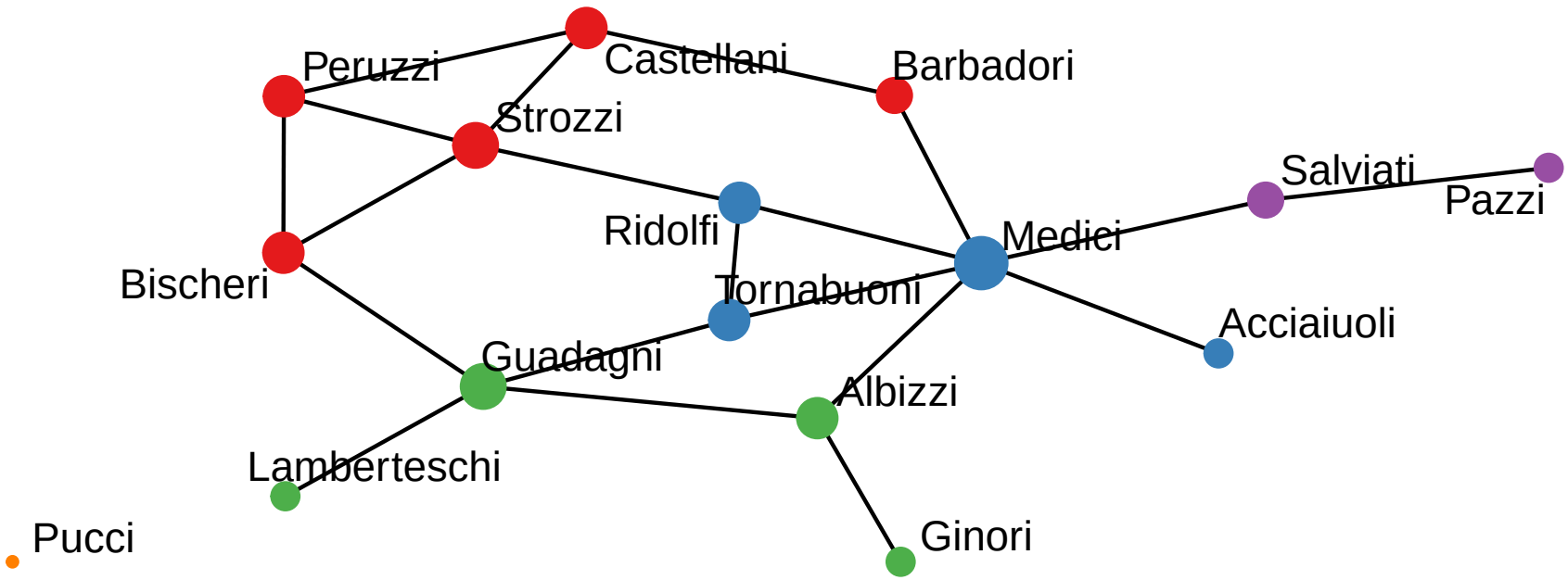
```
ggraph(medici_graph) +
  geom_node_point(aes(size = degree, color = factor(community)),
                  show.legend = FALSE) +
  geom_edge_link() +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_graph()
```



Example 1: Florentine families and the rise of Medici

- Incorporate new network measures into our visualization

```
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()
```



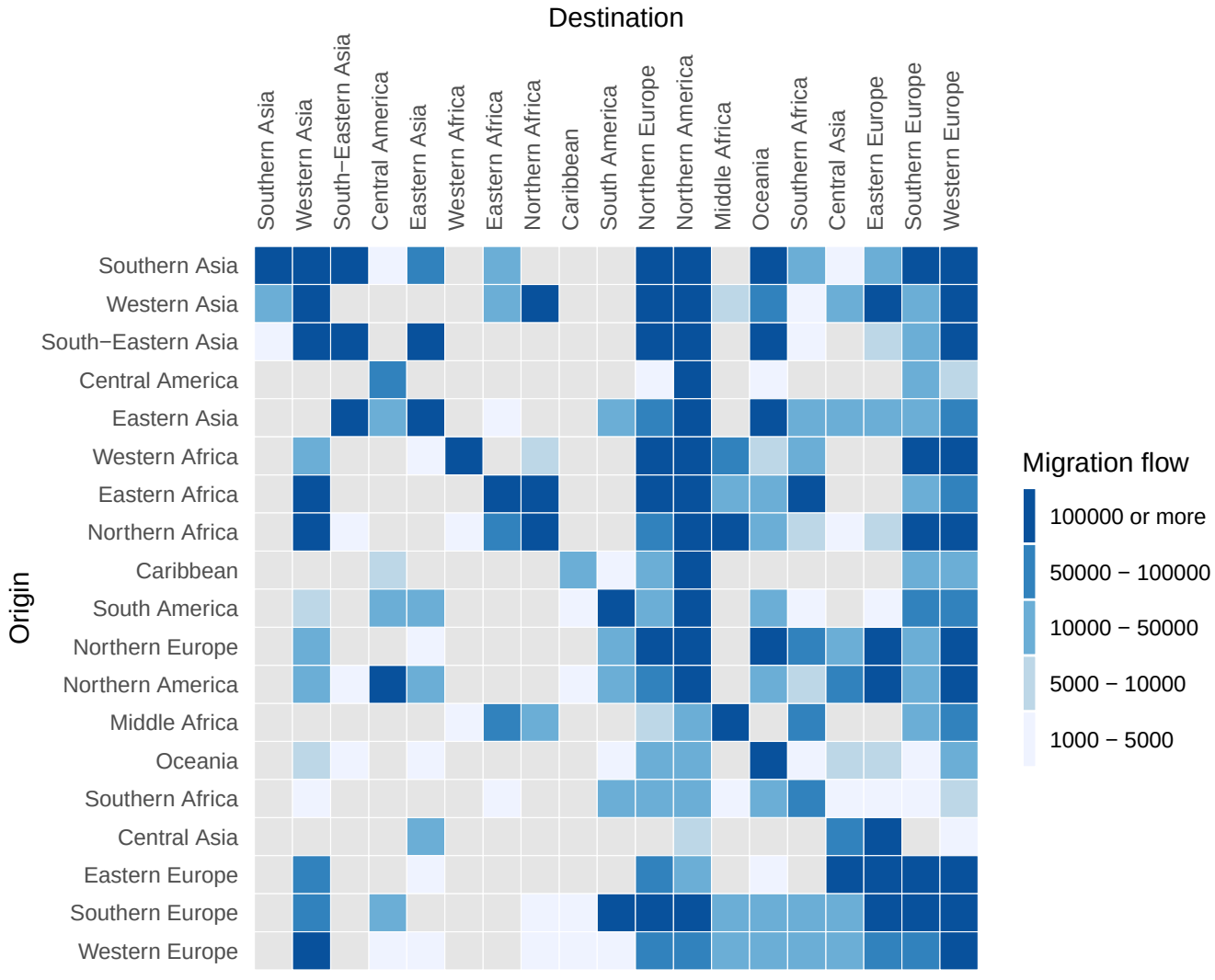
Example 1: Florentine families and the rise of Medici

- Save the output

```
width = 7
ggsave("output/medici.pdf", width = width, height = width/2)
```

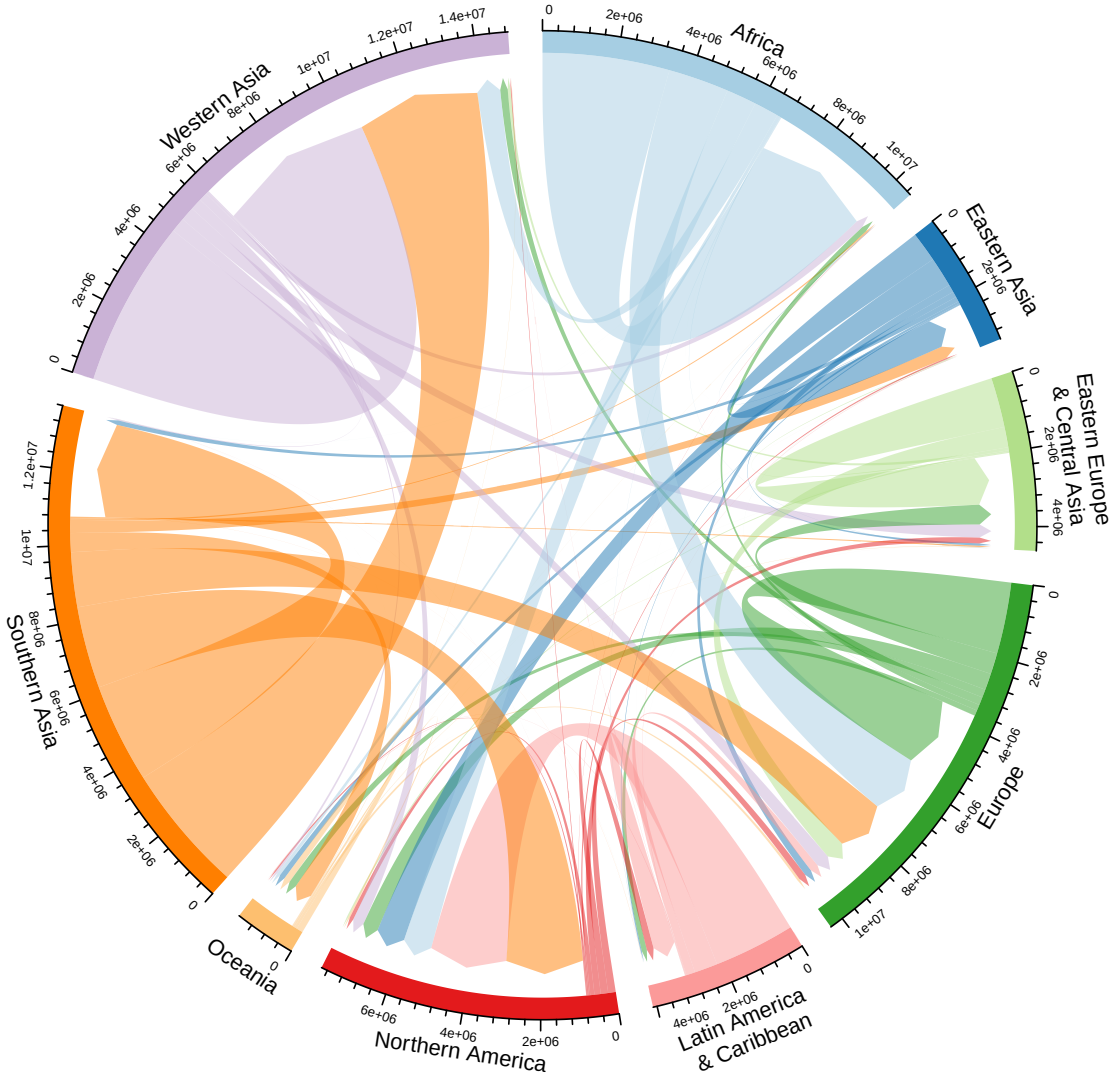
Example 2: Global migration flow data

- Heat map



Example 2: Global migration flow data

- Chord diagram



Example 2: Global migration flow data

- Original data are from Abel (2018)

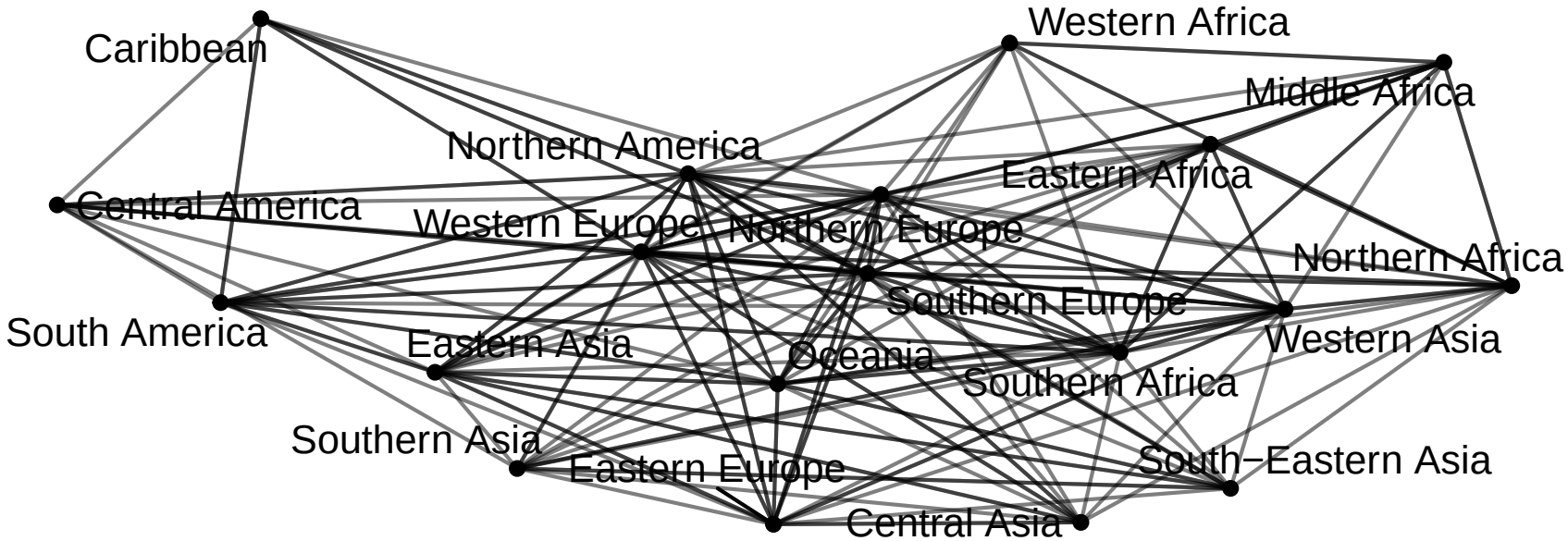
```
migrat2010 <- read_csv("data/migrat2010.csv")

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...

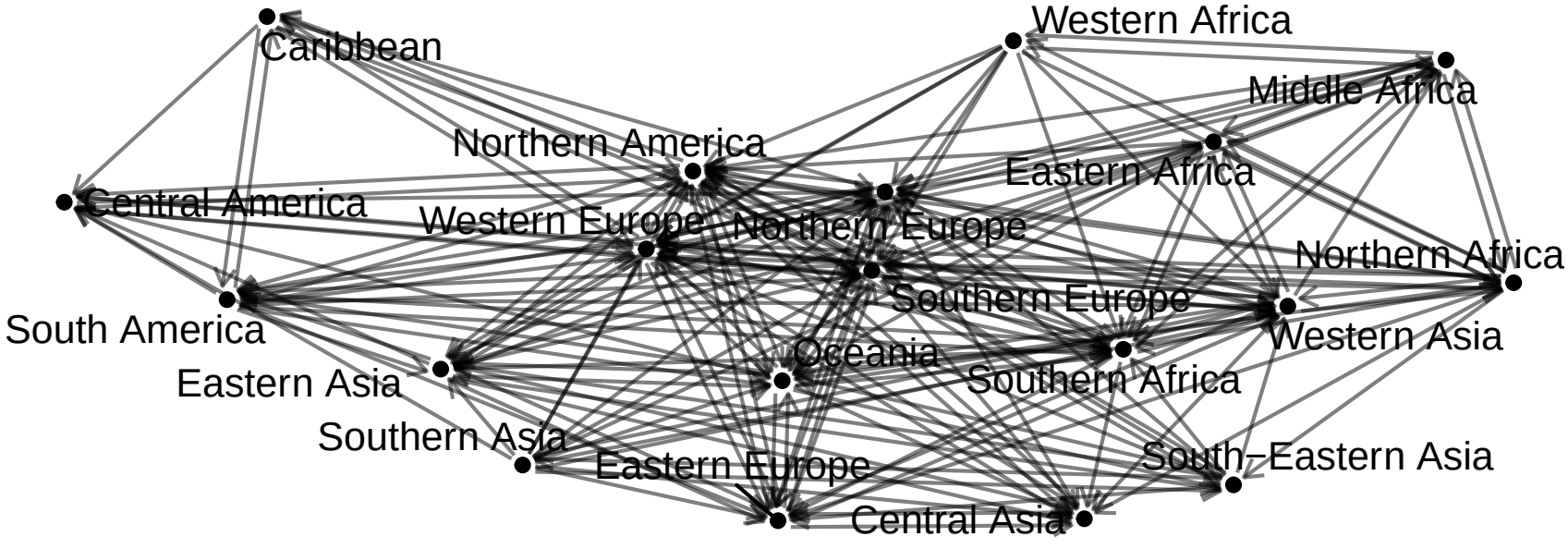
```
migrat2010_graph <- as_tbl_graph(migrat2010)

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

```
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

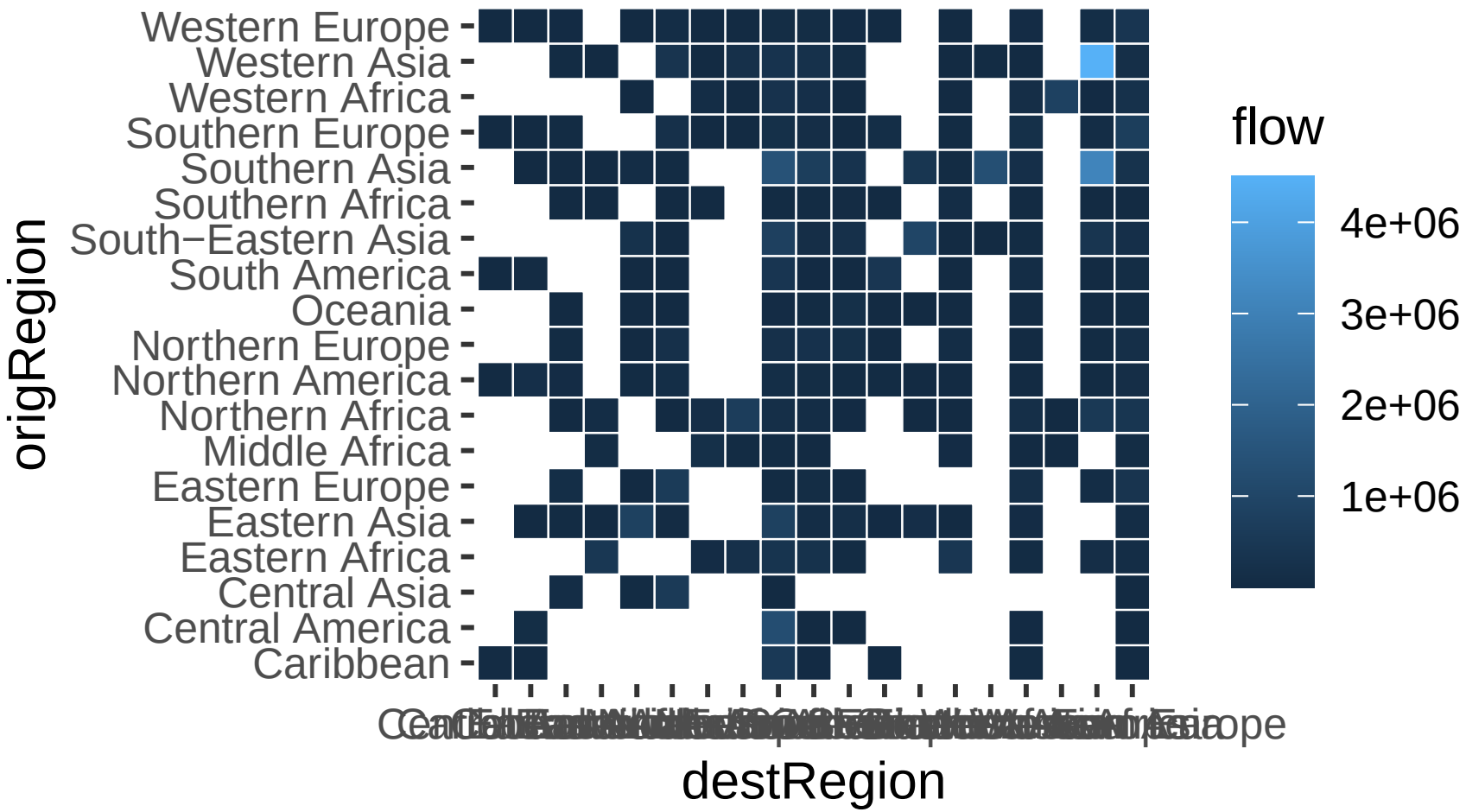
Example 2: Global migration flow data

- Two alternative visualization methods:
  - Heatmap
  - Chord diagram

Example 2: Global migration flow data :: Heatmap

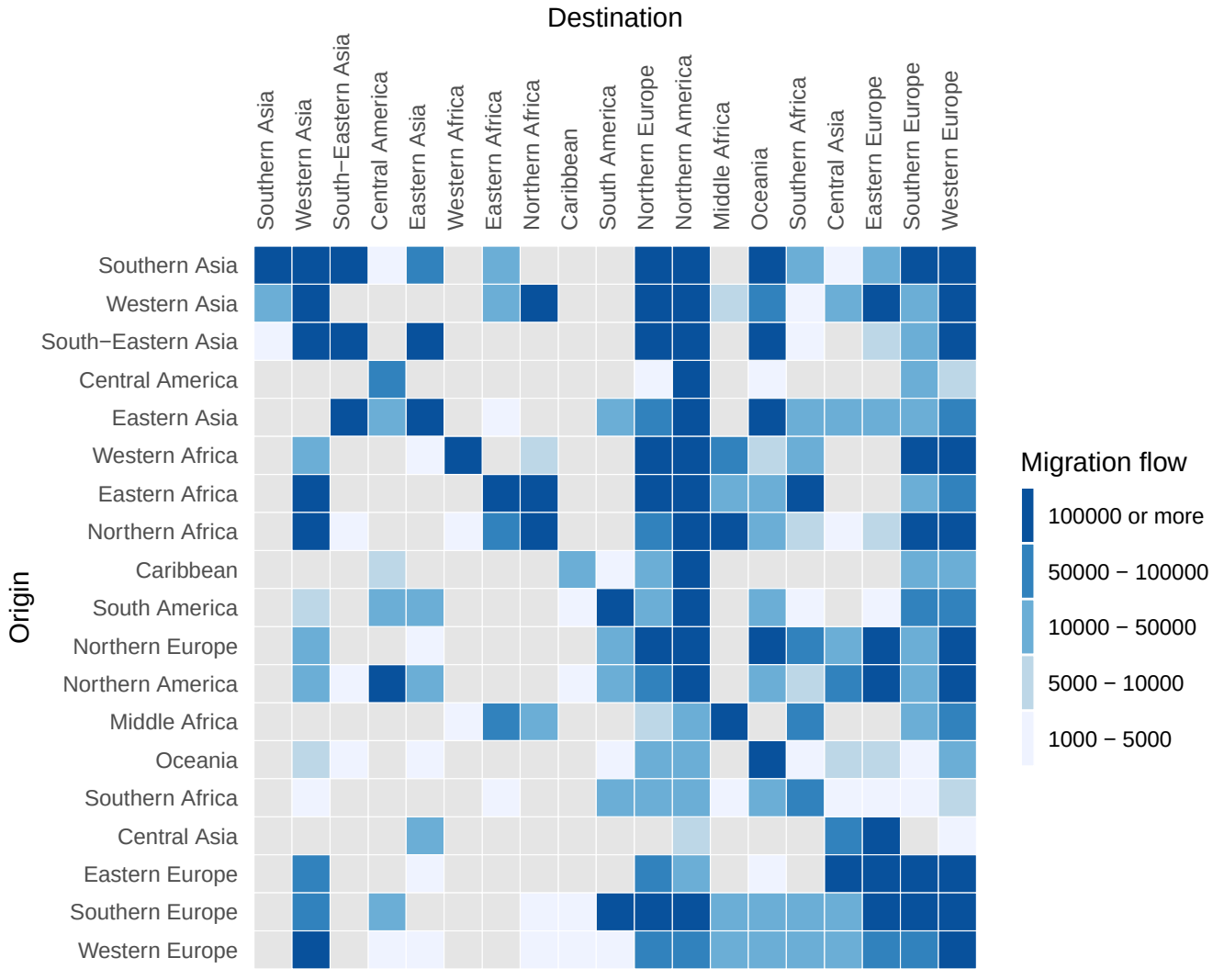
- Like what we did in lab 4 using `geom_tile()`

```
migrat2010 %>%
  ggplot(aes(y = origRegion, x = destRegion, fill = flow)) +
  geom_tile(color = "white", size = 0.2) +
  coord_equal() +
  theme(panel.background = element_blank())
```



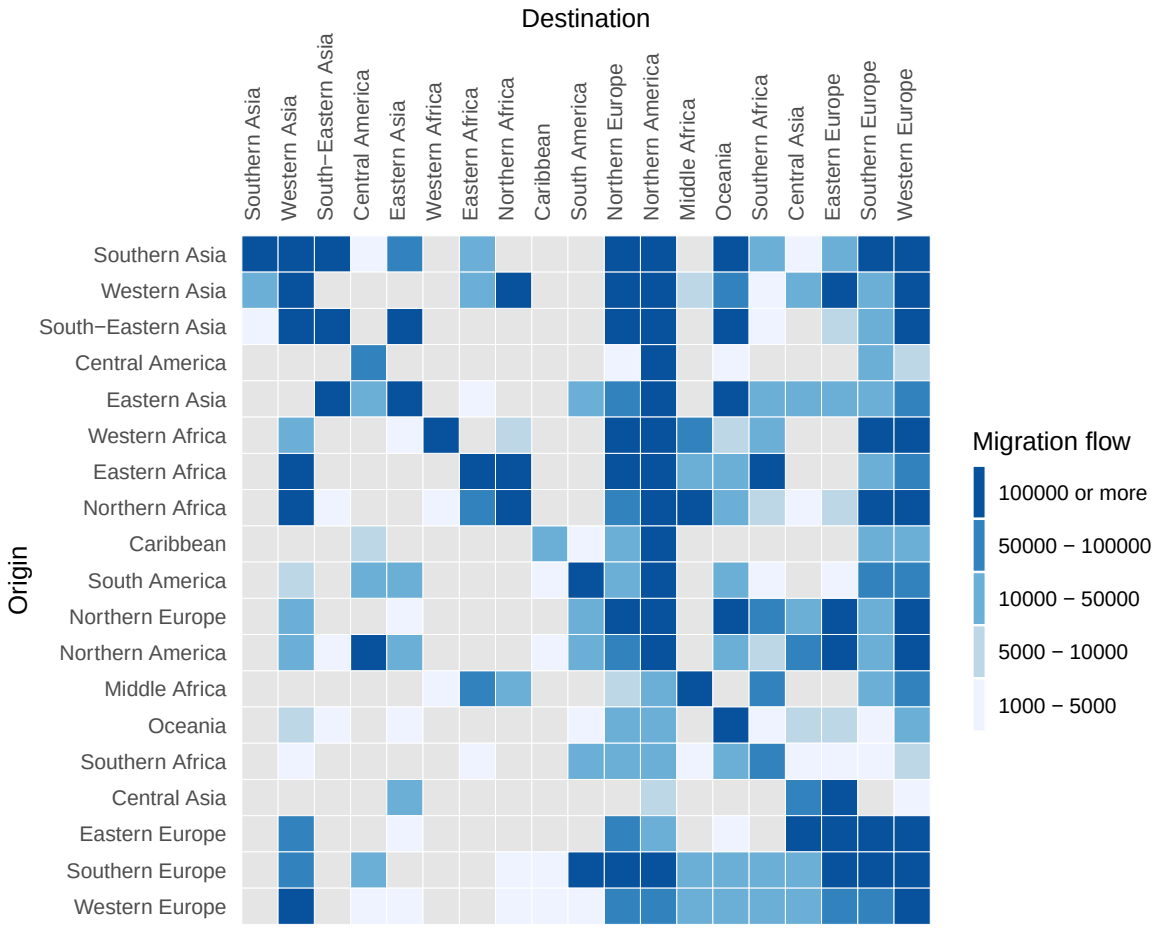
Example 2: Global migration flow data :: Heatmap

- 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



Example 2: Global migration flow data :: Heatmap

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

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

```
## # 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()
```

```
## # 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`

```
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

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

```
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

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

```
##          0%         25%         50%         75%        100%
##    1058.00    9147.75   36104.00  153035.50 4497527.00
```

```
# 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

```
head(migrat2010)
```

```
## # 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

```
# 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

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

```
##                  Caribbean Central America Central Asia Eastern Africa
## Caribbean            40506            8183            0              0
## Central America          0           99171            0              0
## Central Asia             0               0        77252              0
## Eastern Africa           0               0            0         444352
```

Example 2: Global migration flow data :: Heatmap

- Cluster analysis and sorting

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

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

```
##  [1] 15 18 13  2  5 17  4  8  1 12 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)
```

```
##  [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

```
# Re-level `origRegion` and `destRegion` according to the level
migrat2010 <- migrat2010 %>%
  mutate(
    origRegion = factor(origRegion, levels = rev(countryLevels)),
    destRegion = factor(destRegion, levels = countryLevels)
  )
```

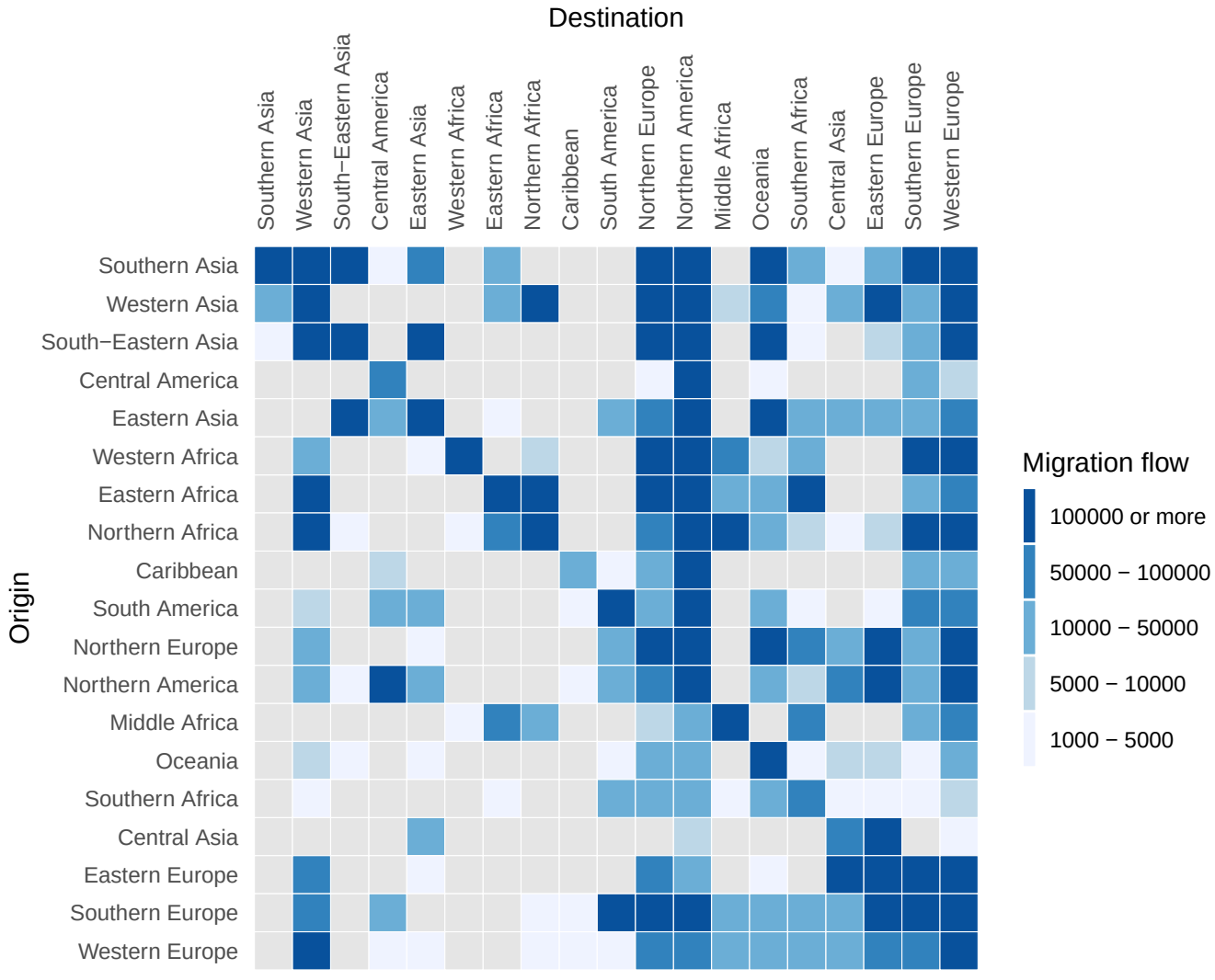
Example 2: Global migration flow data :: Heatmap

- Visualize the heat map again:

```
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")
```

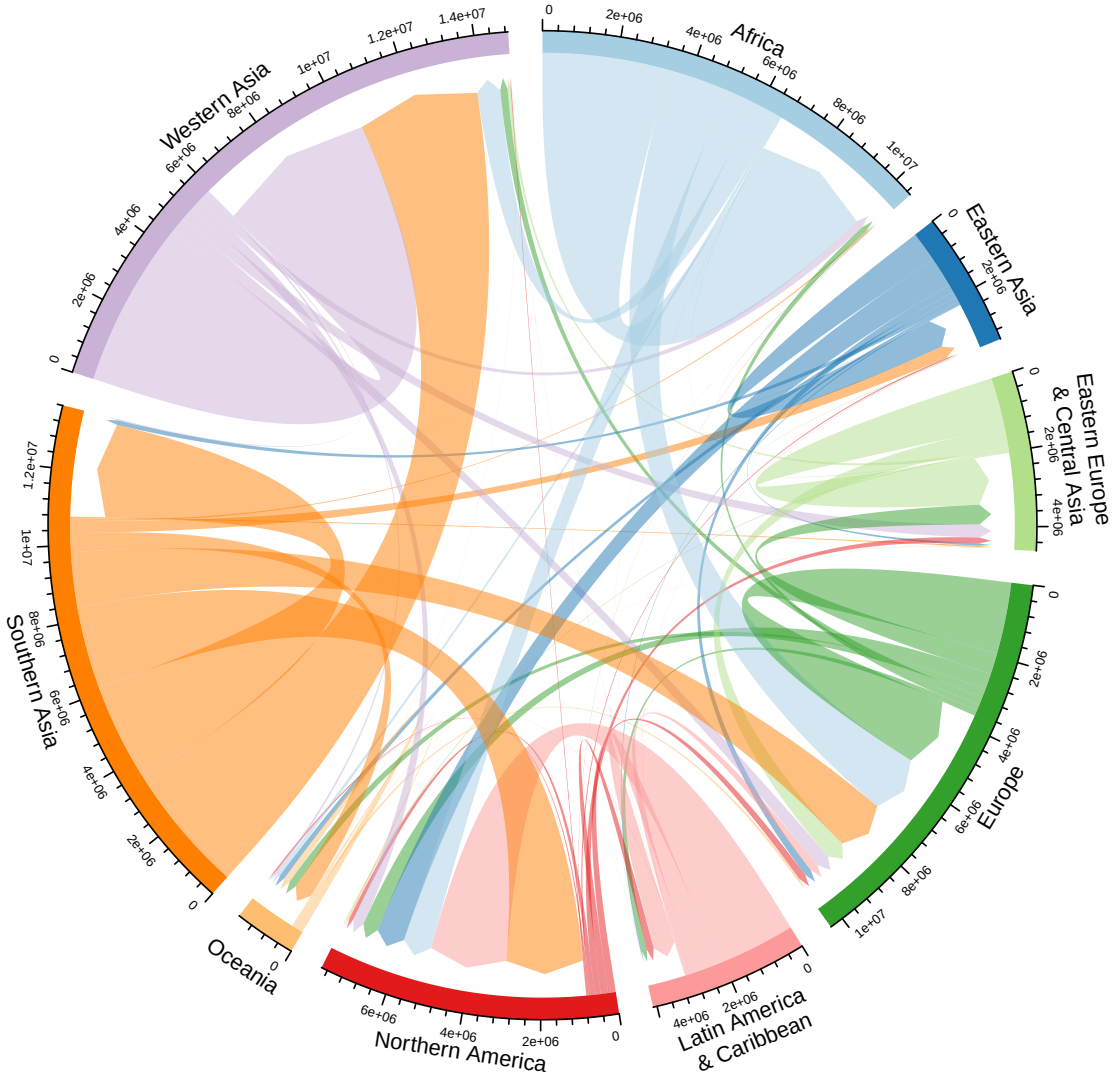
Example 2: Global migration flow data :: Heatmap

- Visualize the heat map again:



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

```
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

```
# 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
```

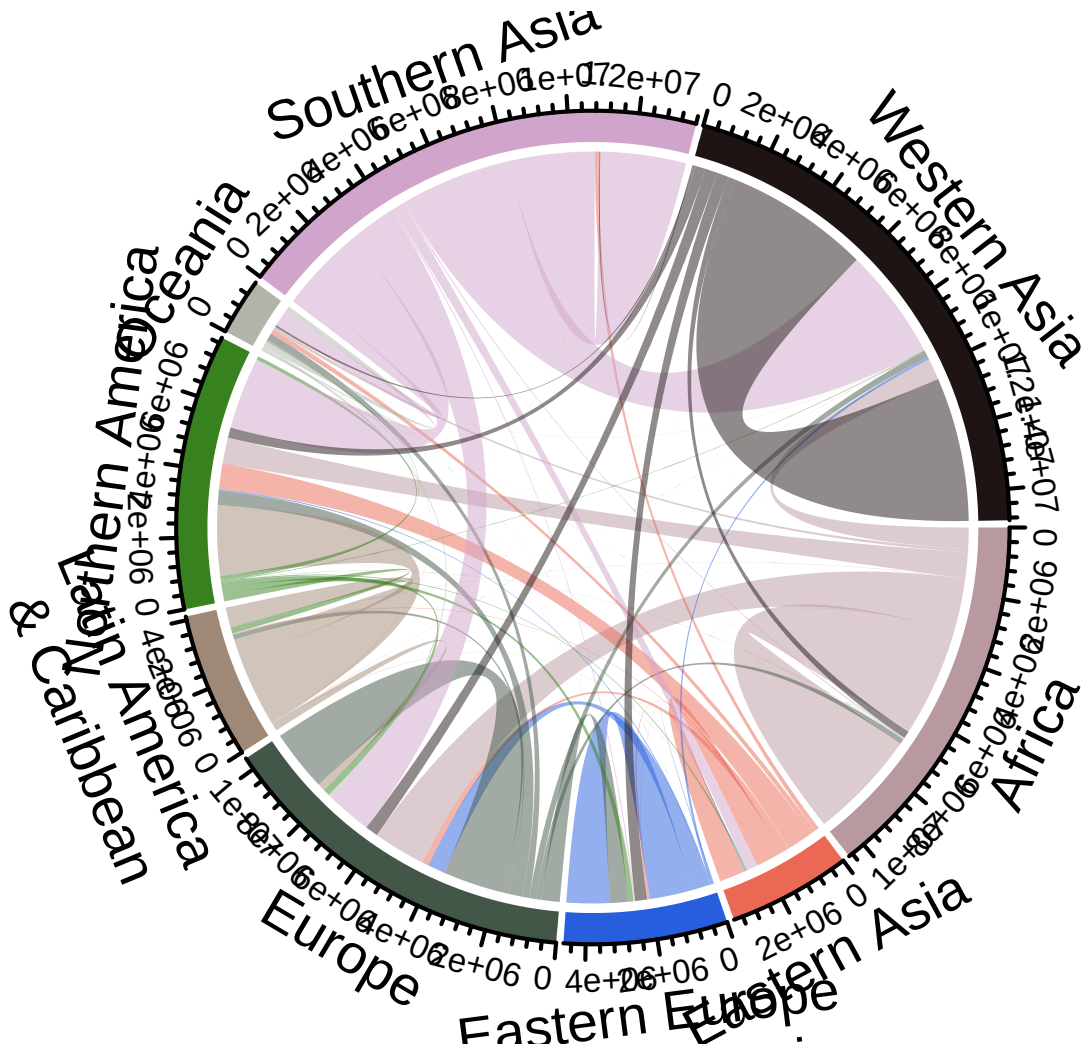
```
head(migrat2010)
```

```
## # 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)

```
# 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)

```
# 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