Chord Diagrams for Visualising Bilateral Migration

Guy J. Abel

Background

- Visualizing bilateral migration is not straightforward
  - Difficult to represent the geographic and temporal aspect at the same time
- Many different approaches
  - Difficult to satisfy everyone's tastes
- Two non-map based approaches
  - Chord Diagrams
  - Alluvial or Sankey Plots
- Non-map based approaches
  - Provide clearer visual guide for geographically small areas that can be overwhelmed on a map
  - Include more bilateral connections

Map based migration visuals

- Houtum and Bueno Lacy (2020) *The migration map trap. On the invasion arrows in the cartography of migration* in Mobilities
  - Cartography peddles a crude distortion of migration
  - Splices into the xenophobic tradition of propaganda cartography.
  - More scientifically robust, critical and creative cartographies of migration are required.

Invasion arrows

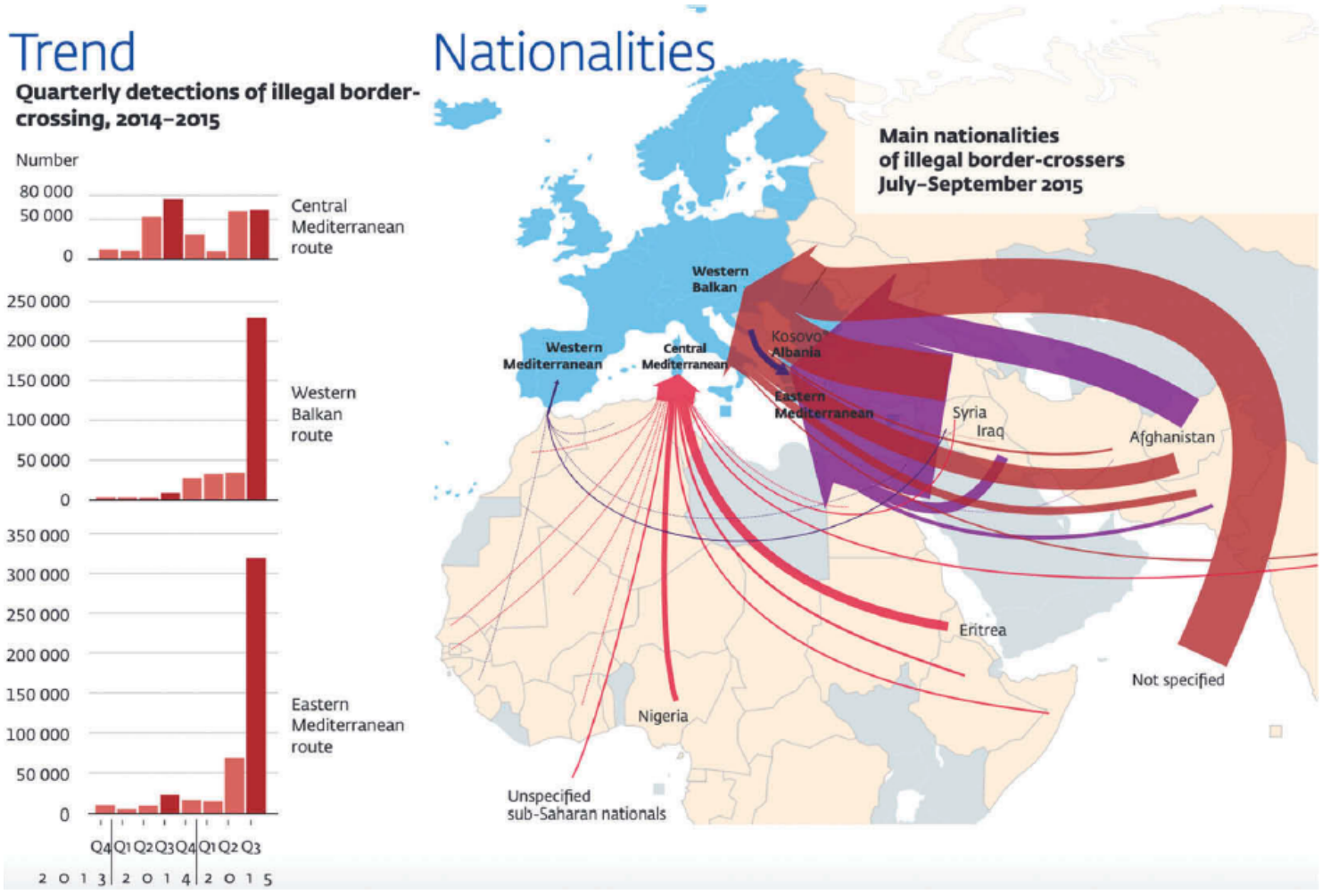


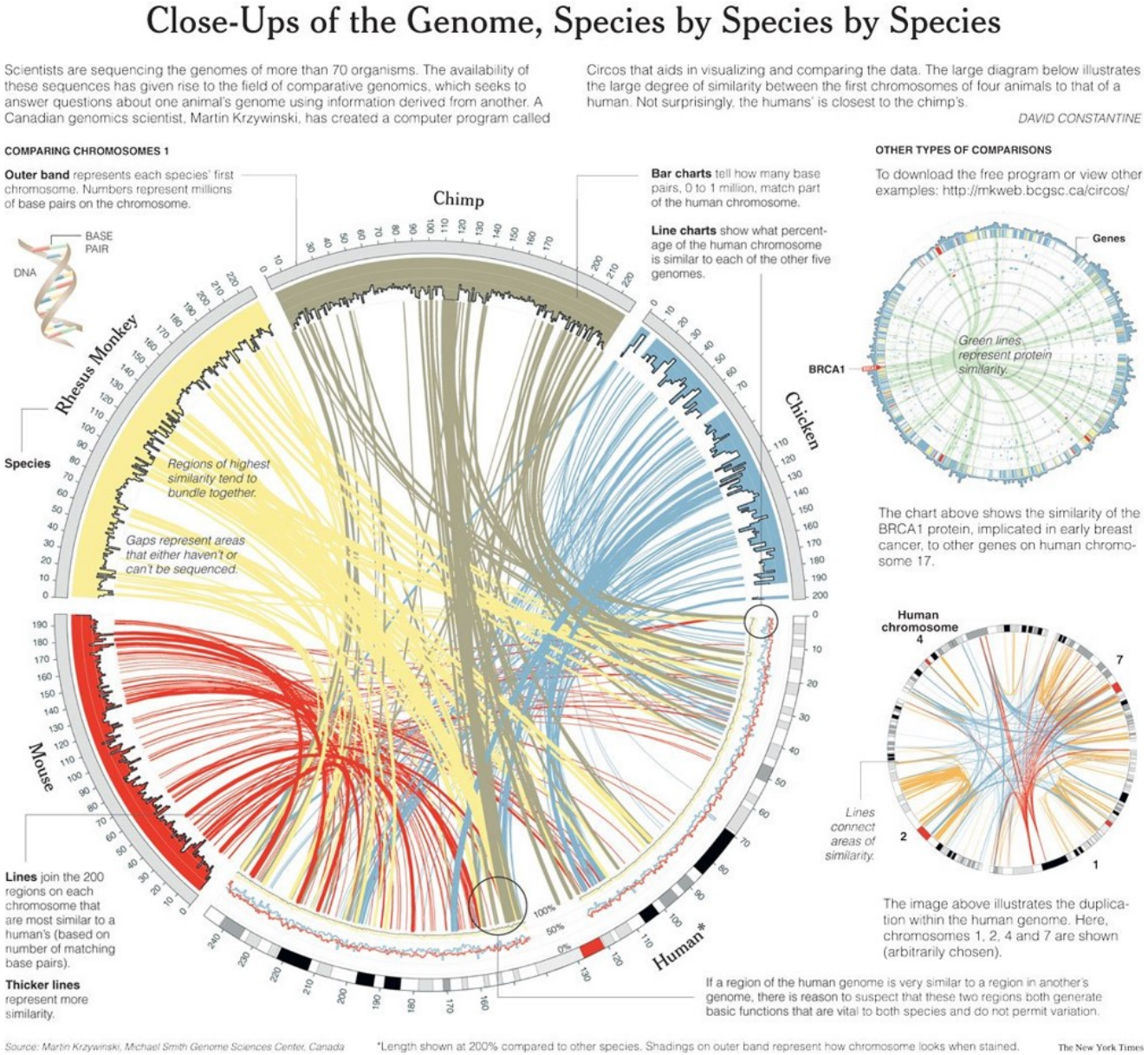
Figure 2. The Frontex map of 2015.

Source: Frontex: https://tinyurl.com/y6o38jq3

Chord Diagrams

- First chord diagrams introduced by Martin Krzywinski in 2007.
  - `https://www.nytimes.com/imagepages/2007/01/22/science/20070123_SCI`
- Used to facilitate the identification and analysis of similarities and differences arising from comparisons of genomes
- Displays relationships between pairs of positions by the use of ribbons, which encode the position, size, and orientation of related genomic elements
- Developed into Circos software in Perl by Krzywinski et al. (2009)
  - `http://circos.ca/`

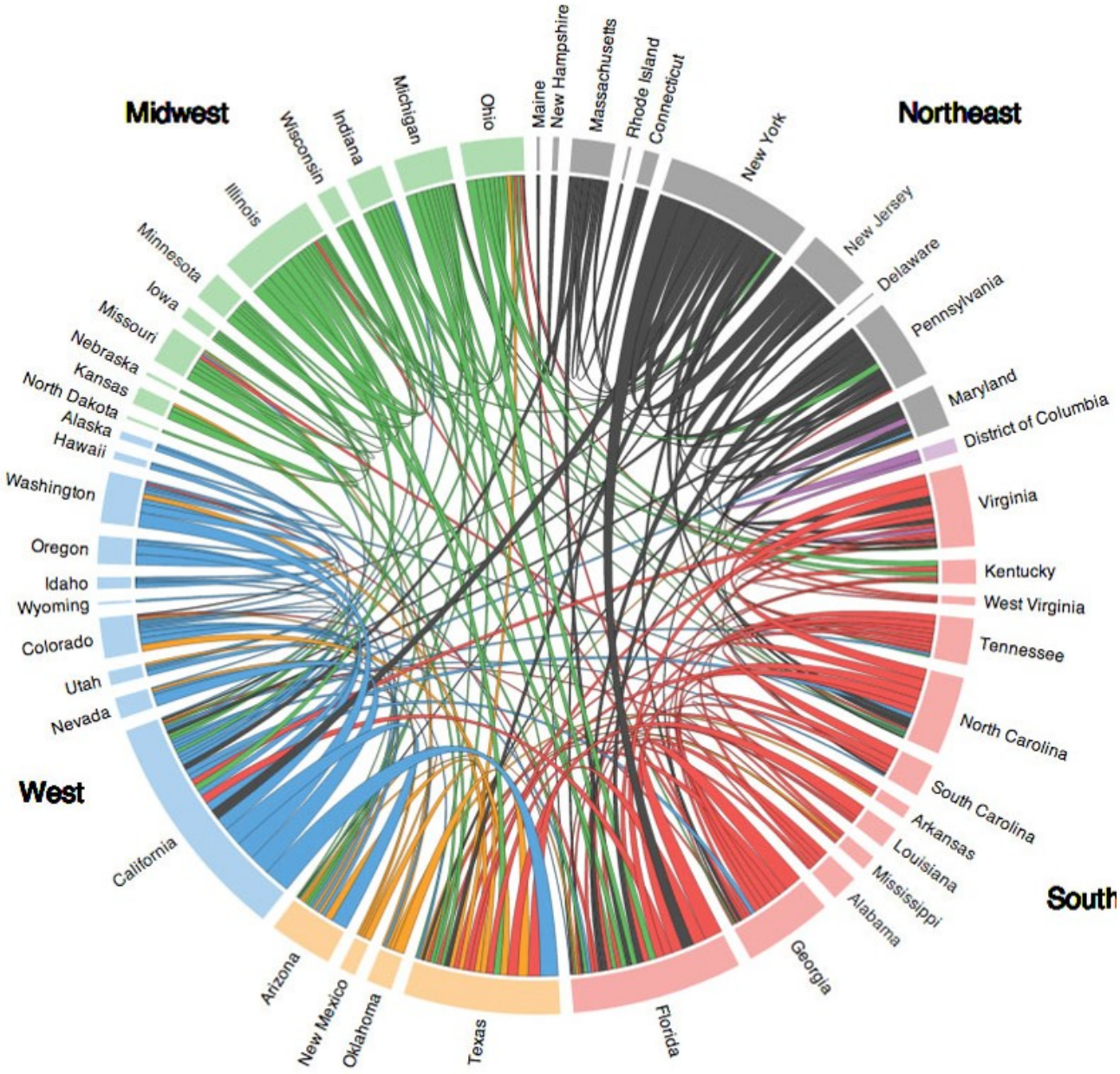
New York Times 2007



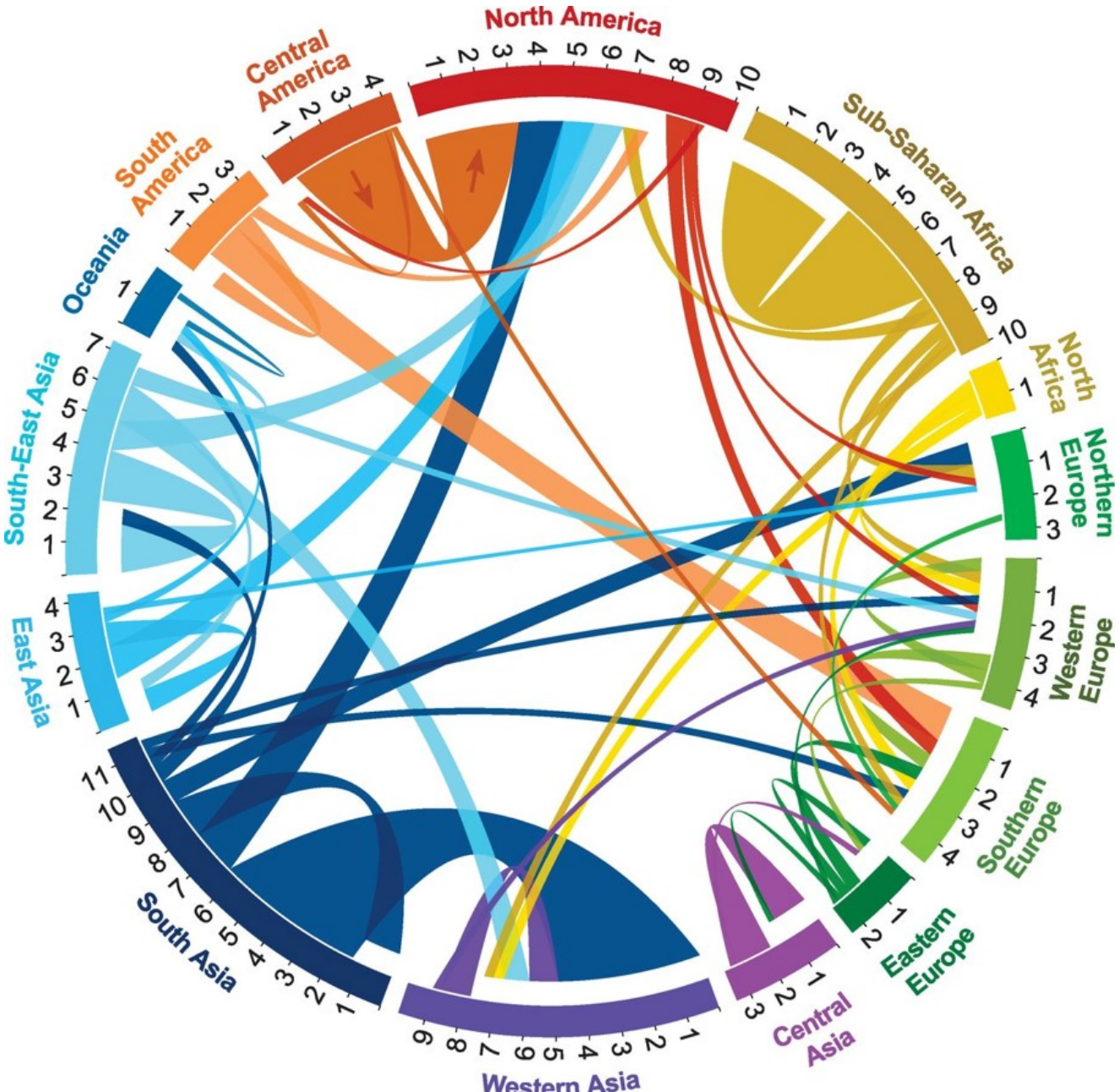
Chord Diagrams with Migration Data

- Interactive chord diagram plots introduced into D3.js library by Mike Bostock
- First used to illustrate migration patterns by data journalist Chris Walker in 2013
  - Mapping America's Restless Interstate Migration Without a Map `https://www.wired.com/2013/11/mapping-migration-without-a-map/`
- Does not show the direction of move until mouse-over.
- Nikola Sander adapted Circos software to add directional indicators for flows
  - First used in Abel and Sander (2014). *Quantifying Global International Migration Flows*. Science, 343 (6178).
  - Interactive version at `http://download.gsb.bund.de/BIB/global_flow/`

Chord Diagrams with Migration Data



Chord Diagrams with Migration Data



Chord Diagrams in R

- Some drawbacks to the Circos based plots
  - Inflows plotted first on each sector
  - Chords for smaller flows overlap larger flows
  - Hides smallest flows
  - Not easy to detect direction of flows
  - Addition of direction arrows usually require some further touch using a second piece of software, e.g. Photoshop or Illustrator
    - Problematic for replicating and/or updating

Chord Diagrams in R

- In recent years a number of R packages that implement similar plots as the Circos software have appeared on CRAN
- The *circlize* R package by Gu et al. (2014) is perhaps the most complete and accessible for non-genomic data
  - Built on base R graphics package
- Includes a `chordDiagram()` function
  - Extensive documentation of the `chordDiagram()` function in Chapters 13-15 of the circlize book.

UN international migrant stock data 2020

```
> library(tidyverse)
> un <- read_csv(file = "../data/un_desa_ims_tidy.csv")
> un
# A tibble: 259,357 x 6
    year     stock dest  dest_code orig                orig_code
   <dbl>     <dbl> <chr>     <dbl> <chr>                   <dbl>
 1  1990 152986157 WORLD       900 WORLD                     900
 2  1995 161289976 WORLD       900 WORLD                     900
 3  2000 173230585 WORLD       900 WORLD                     900
 4  2005 191446828 WORLD       900 WORLD                     900
 5  2010 220983187 WORLD       900 WORLD                     900
 6  2015 247958644 WORLD       900 WORLD                     900
 7  2020 280598105 WORLD       900 WORLD                     900
 8  1990  15334807 WORLD       900 Sub-Saharan Africa        947
 9  1995  16488973 WORLD       900 Sub-Saharan Africa        947
10  2000  15638014 WORLD       900 Sub-Saharan Africa        947
# ... with 259,347 more rows
# i Use `print(n = ...)` to see more rows
```

UN international migrant stock data 2020

- Use continent to continent flows in 2020

```
> # codes for contents
> cc <- c(903, 935, 908, 904, 905, 909)
> d <- un %>%
+   filter(orig_code %in% cc,
+          dest_code %in% cc,
+          year == 2020) %>%
+   select(orig, dest, stock)
> d
# A tibble: 36 x 3
   orig                                dest      stock
   <chr>                               <chr>     <dbl>
 1 AFRICA                              AFRICA 20917565
 2 ASIA                                AFRICA  1207631
 3 EUROPE                              AFRICA   648455
 4 LATIN AMERICA AND THE CARIBBEAN     AFRICA    32524
 5 NORTHERN AMERICA                    AFRICA    53563
 6 OCEANIA                             AFRICA    14483
 7 AFRICA                              ASIA    4720103
 8 ASIA                                ASIA   68497762
 9 EUROPE                              ASIA    7169630
10 LATIN AMERICA AND THE CARIBBEAN     ASIA     414658
# ... with 26 more rows
# i Use `print(n = ...)` to see more rows
```

Default `chordDiagram()`

- The `chordDiagram()` function can take either a `matrix` or `data.frame` object as first argument `x` for the data.
- I prefer the latter as they are much easier to create and manipulate (using *dplyr* and other *tidyverse* packages).
  - When using a `data.frame`, the first three columns should correspond to the origin, destination and size of connection.
  - Columns can take any name, but must be in that order.
  - Will also work with `tbl_df` (tibble)
- Many options in `chordDiagram()`, that by default are not ideal for displaying migration data

```
> library(circlize)
> chordDiagram(x = d)
```

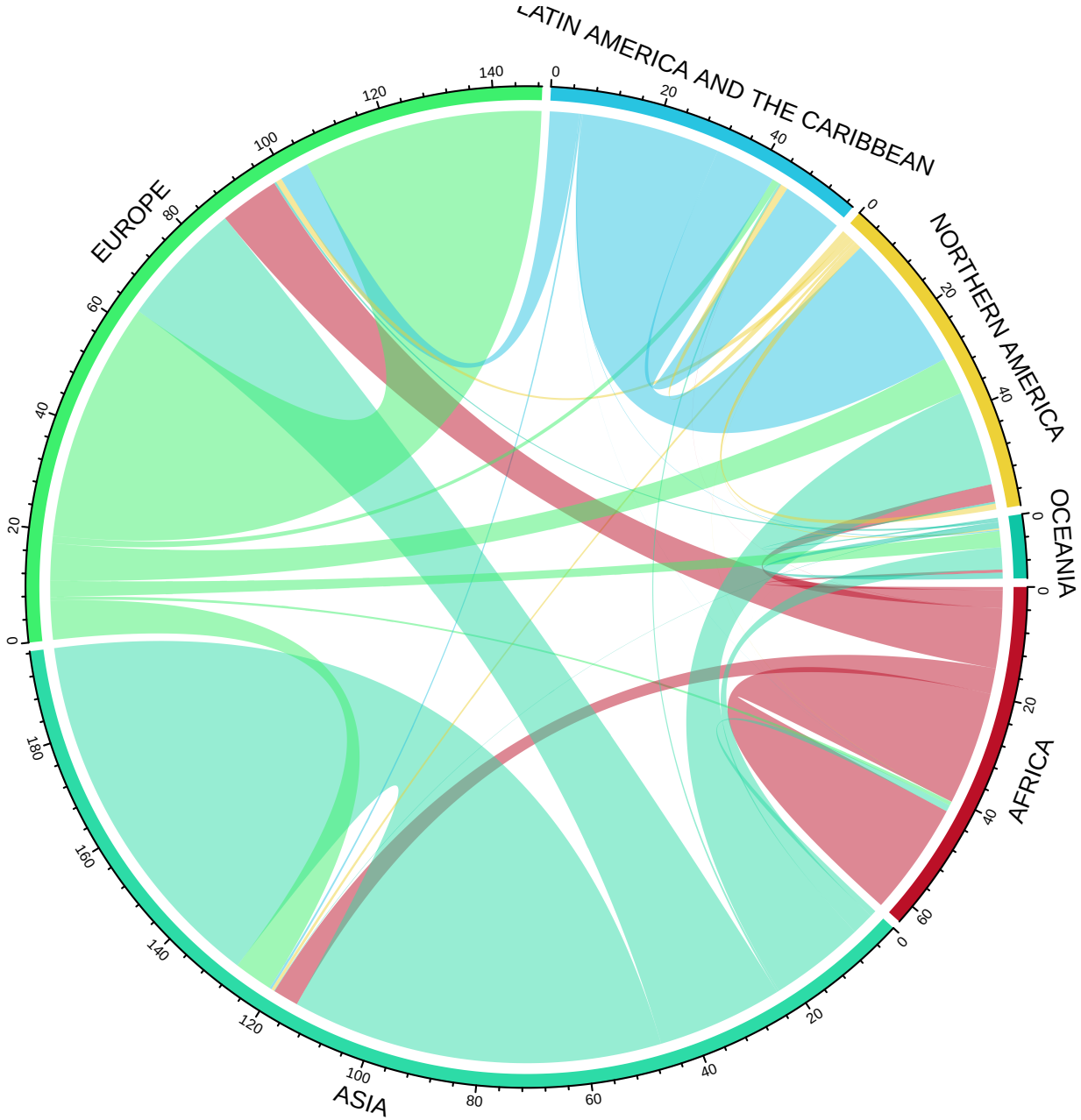
Default chordDiagram()

Scale

- Edit the bilateral counts to a sensible scale to ensure the axis labels are easily legible.

```
> d <- mutate(d, stock = stock/1e6)
> d
# A tibble: 36 x 3
   orig                              dest     stock
   <chr>                             <chr>    <dbl>
 1 AFRICA                            AFRICA 20.9
 2 ASIA                              AFRICA  1.21
 3 EUROPE                            AFRICA  0.648
 4 LATIN AMERICA AND THE CARIBBEAN   AFRICA  0.0325
 5 NORTHERN AMERICA                  AFRICA  0.0536
 6 OCEANIA                           AFRICA  0.0145
 7 AFRICA                            ASIA    4.72
 8 ASIA                              ASIA   68.5
 9 EUROPE                            ASIA    7.17
10 LATIN AMERICA AND THE CARIBBEAN   ASIA    0.415
# ... with 26 more rows
# i Use `print(n = ...)` to see more rows
> chordDiagram(x = d)
```

Scale

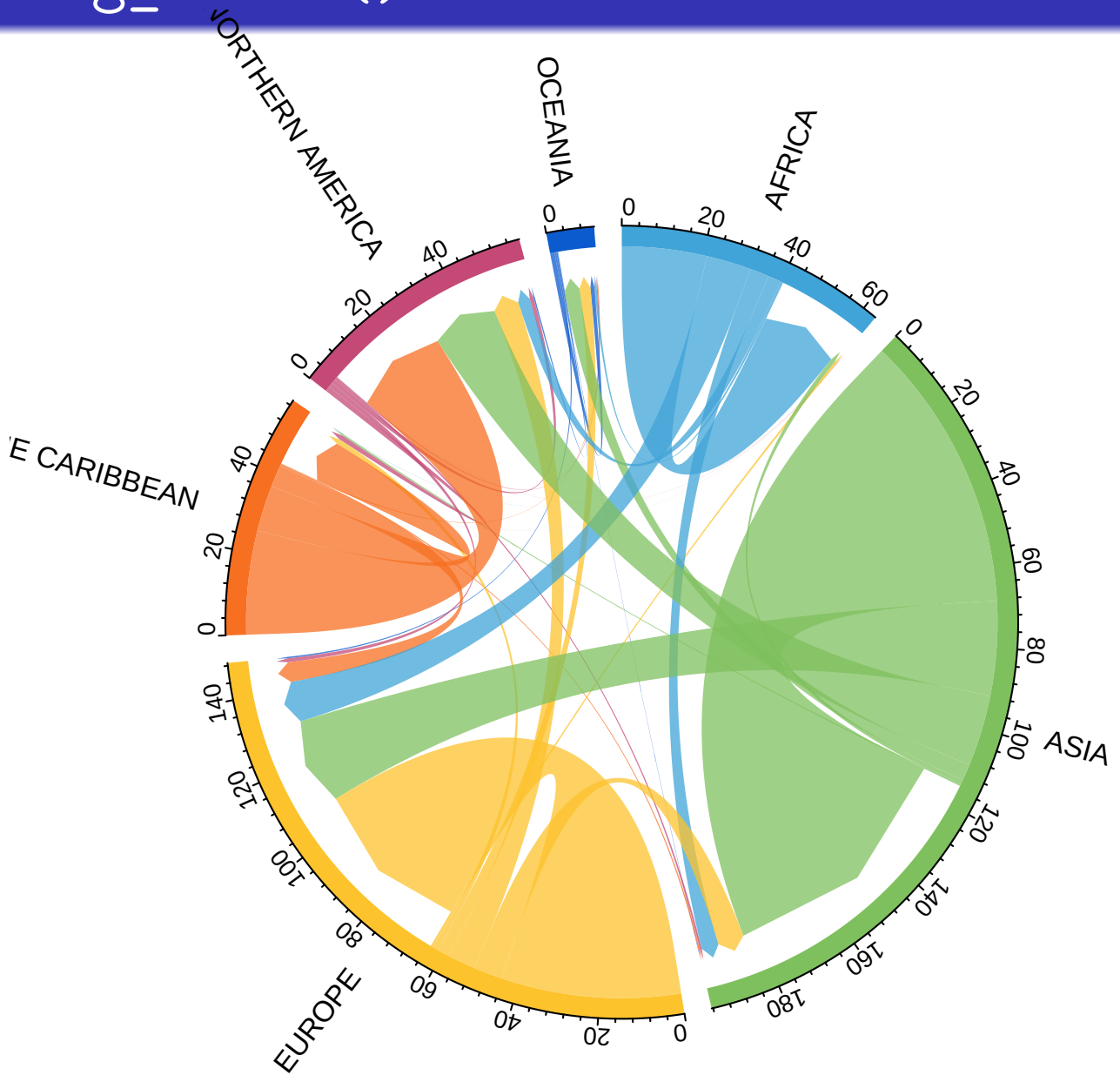

LATIN AMERICA AND THE CARIBBEAN
NORTHERN AMERICA
OCEANIA
AFRICA
ASIA
EUROPE

mig_chord()

- There currently 40 arguments in `chordDiagram()`
  - Many of the default settings are not ideal for migration data
  - Changing the labels and the axis are very complicated (done outside of `chordDiagram()`)
- The `mig_chord()` function in *migest* piggy-backs on the `chordDiagram()` diagram function, setting sensible defaults for `chordDiagram()` specific for plotting migration data
  - Sets a fixed default colour palette
  - Additional arguments to help with label placement and axis appearance

```
> library(migest)
> mig_chord(x = d)
```

Default mig_chord()



Labels

- To provide nice labels, use a data set for the regions:

```
> r <- tibble(region = union(d$orig, d$dest))
> r
# A tibble: 6 x 1
  region
  <chr>
1 AFRICA
2 ASIA
3 EUROPE
4 LATIN AMERICA AND THE CARIBBEAN
5 NORTHERN AMERICA
6 OCEANIA
```

Labels

- Use *stringr* to format labels. Provide two options
  - Labels with line break \n
  - Labels split in two columns

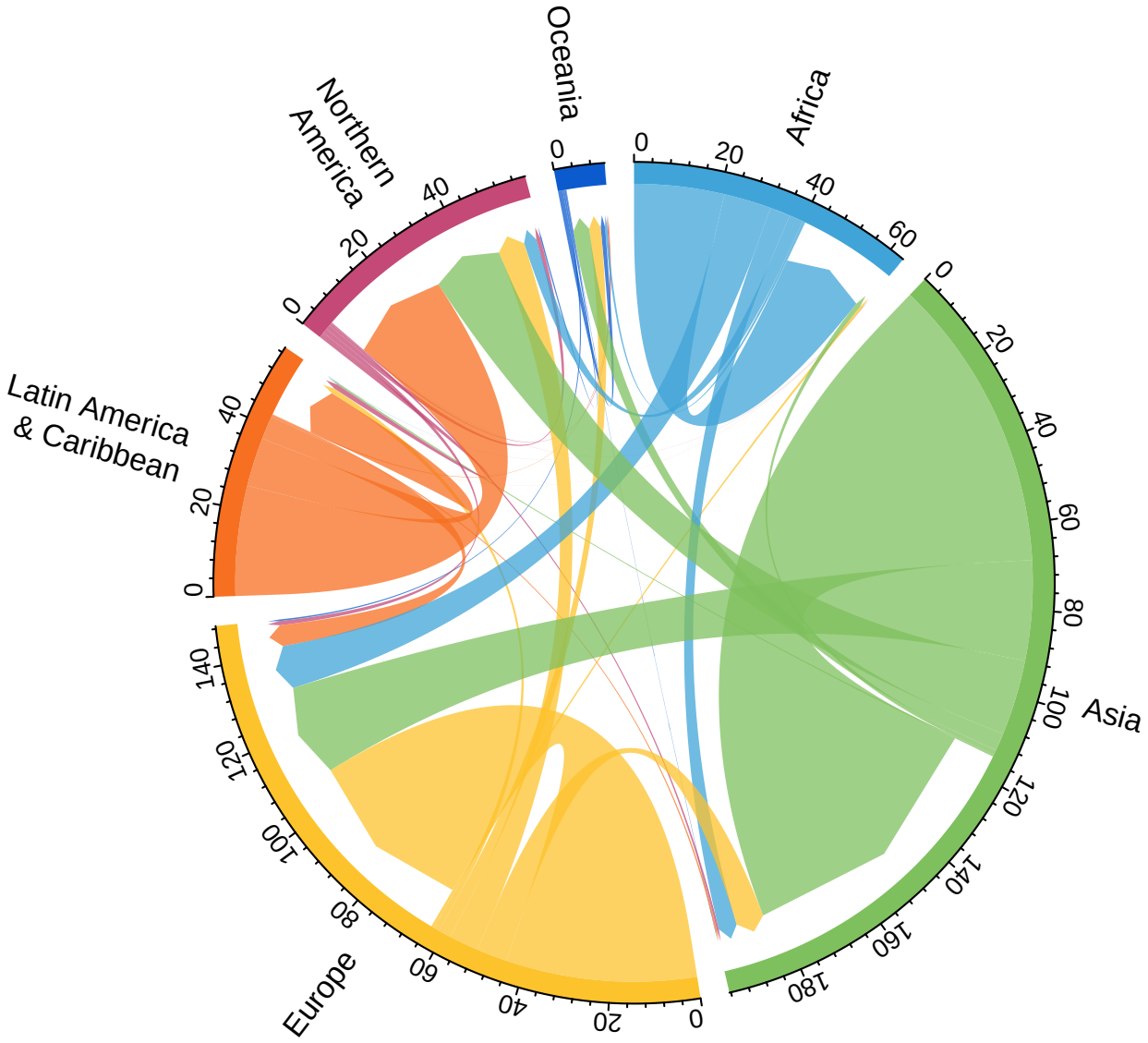
```
> r <- r %>%
+   mutate(lab = str_to_title(string = region),
+          lab = str_replace(string = lab, pattern = "And The", replacement = "&"),
+          # use str_wrap to split longer labels into two
+          lab = str_wrap(string = lab, width = 14)) %>%
+   # separate based on \n
+   separate(col = lab, into = c("lab1", "lab2"), sep = "\n", fill = "right", remov
> r
# A tibble: 6 x 4
  region                          lab                              lab1          lab2
  <chr>                           <chr>                            <chr>         <chr>
1 AFRICA                          "Africa"                         Africa        <NA>
2 ASIA                            "Asia"                           Asia          <NA>
3 EUROPE                          "Europe"                         Europe        <NA>
4 LATIN AMERICA AND THE CARIBBEAN "Latin America\n& Caribbean"     Latin Amer~   & Ca~
5 NORTHERN AMERICA                "Northern\nAmerica"              Northern      Amer~
6 OCEANIA                         "Oceania"                        Oceania       <NA>
```

Labels

- Pass a named vector to `lab` for horizontal labels

```
> reg_lab <- r %>%
+   select(region, lab) %>%
+   deframe()
> reg_lab
                         AFRICA                                   ASIA
                      "Africa"                                 "Asia"
                         EUROPE LATIN AMERICA AND THE CARIBBEAN
                      "Europe"    "Latin America\n& Caribbean"
            NORTHERN AMERICA                                  OCEANIA
         "Northern\nAmerica"                                "Oceania"
>
> mig_chord(d, lab = reg_lab)
```

Horizontal Labels

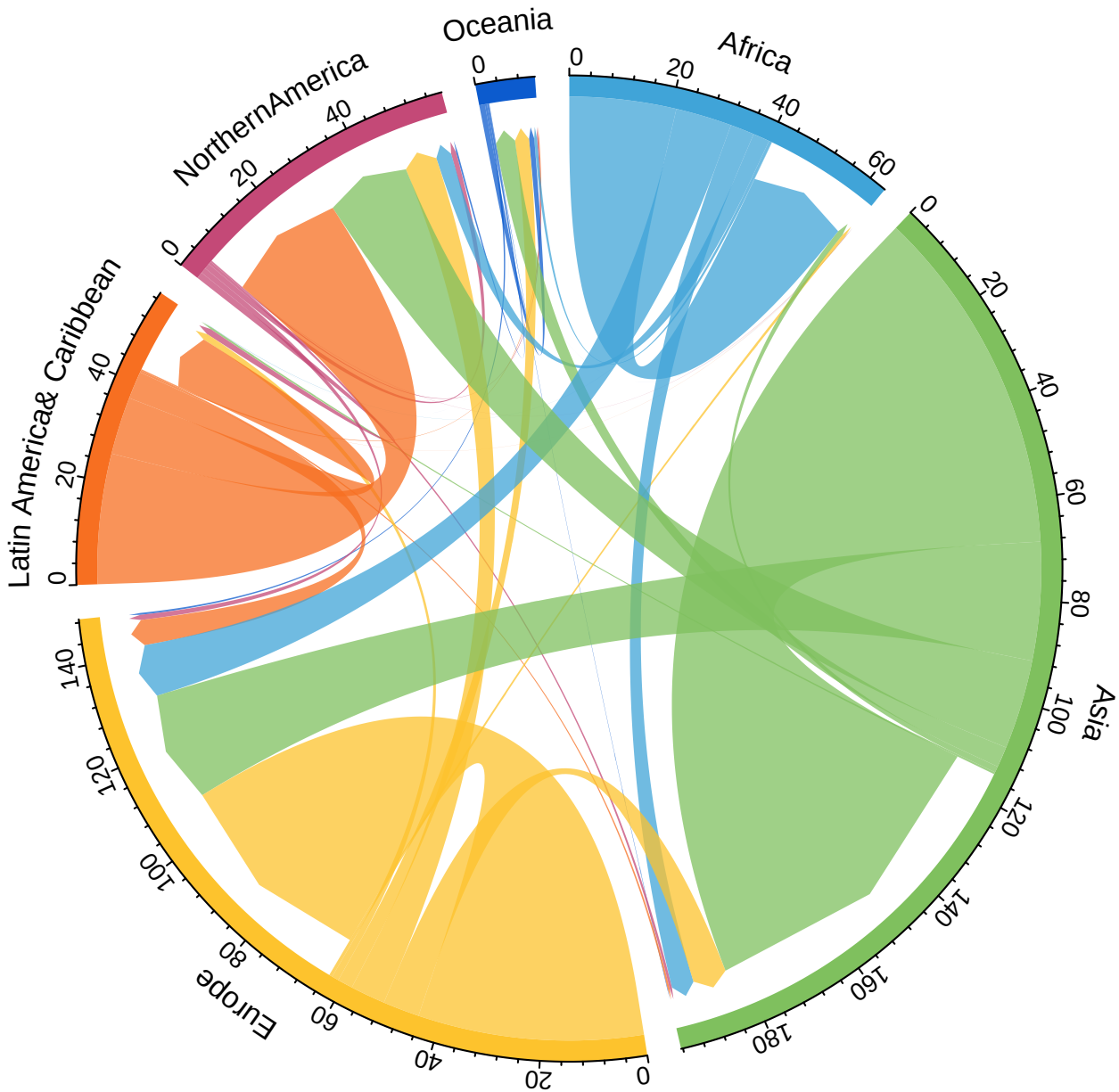


Bending Labels

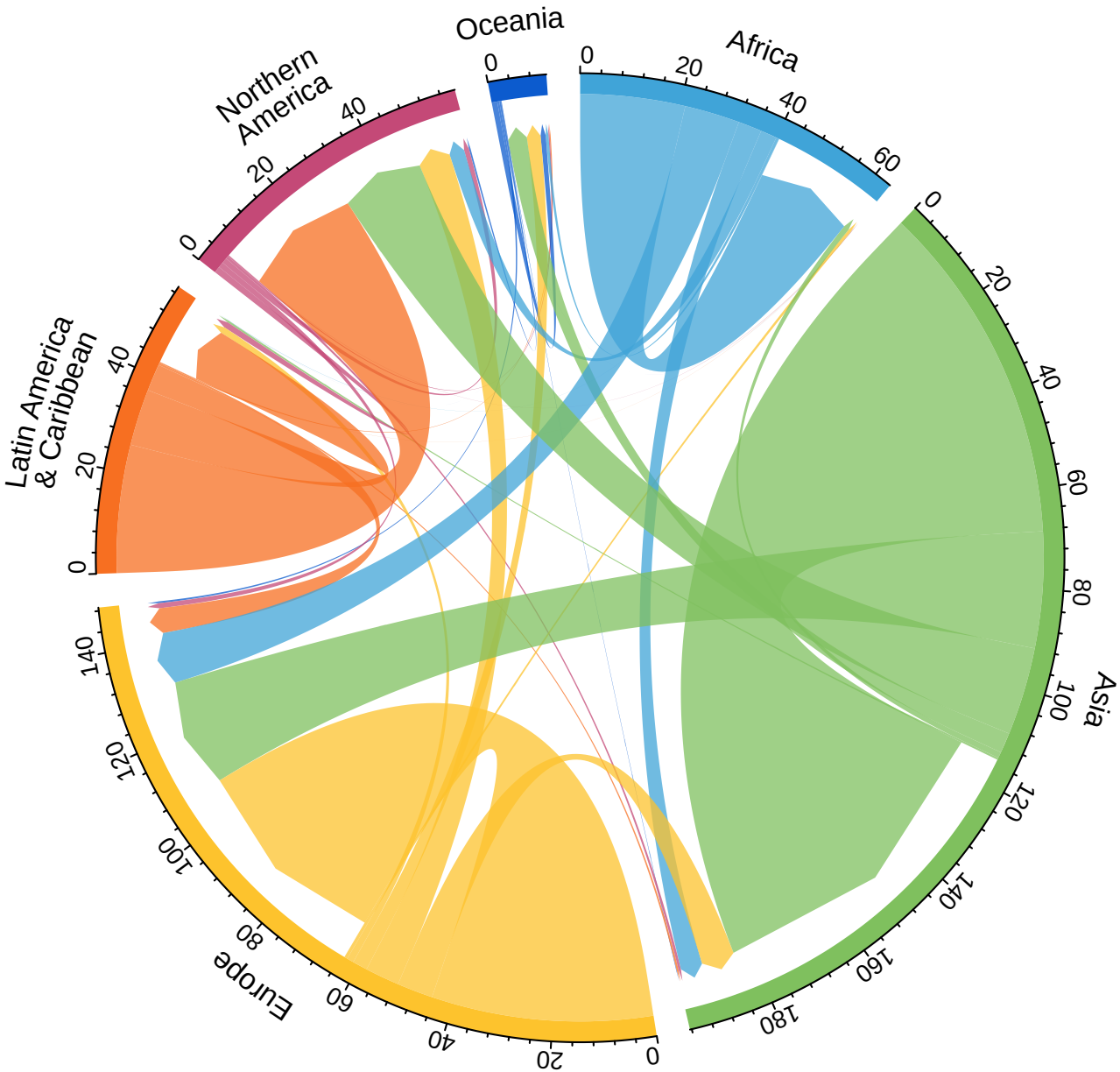
- Pass a named vector(s) to `lab_bend1` and `lab_bend2` for bending labels

```
> mig_chord(d, lab_bend1 = reg_lab)
>
> mig_chord(d,
+           lab_bend1 = r %>%
+             select(region, lab1) %>%
+             deframe(),
+           lab_bend2 = r %>%
+             select(region, lab2) %>%
+             deframe())
```

Bending Labels



Bending Labels

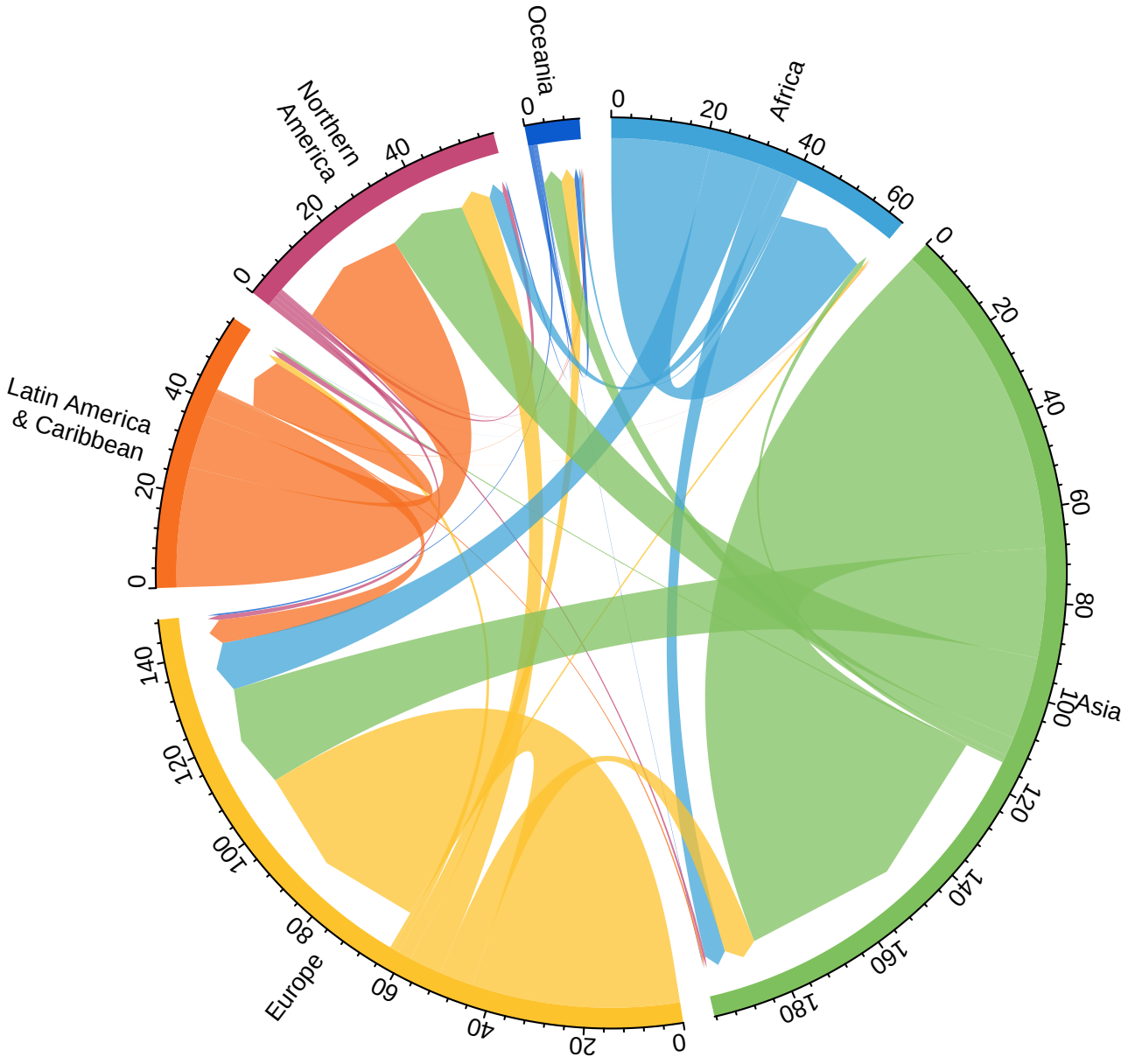


Label Spacing

- Can control the label spacing and positioning using
  - `preAllocateTracks` from `chordDiagram()`, controlling the share of the plot for label tracks
  - `label_size` the font size for the labels
  - `label_nudge` to shift labels towards (negative number) or away (positive number) the sector axis

```
> mig_chord(d, lab = reg_lab,
+           preAllocateTracks = list(track.height = 0.1),
+           label_size = 0.8,
+           label_nudge = -0.3)
```

Label Spacing

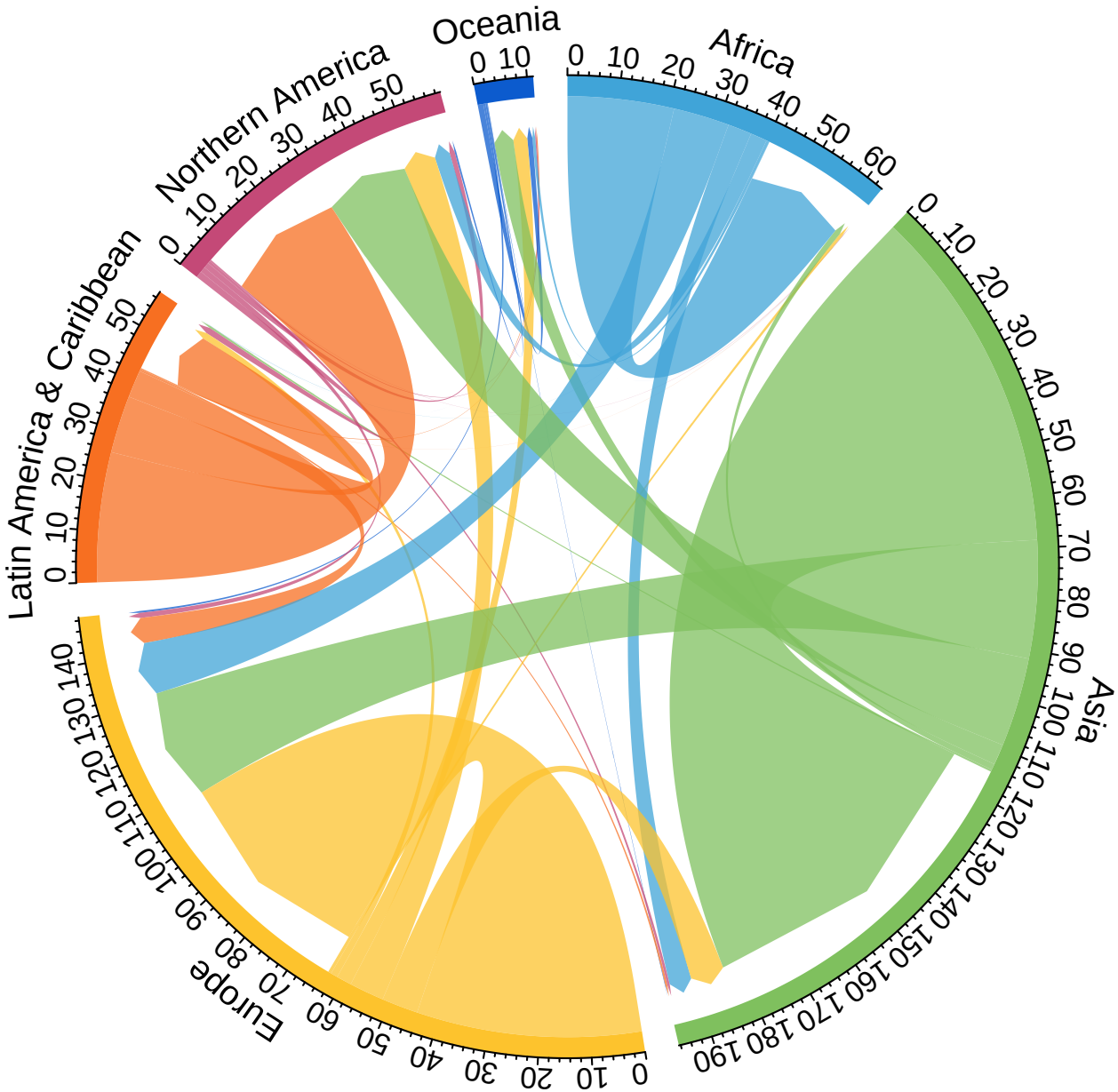


Axis

- Options in `mig_chord()` to control
  - `axis_breaks` number for breaks on axis labels
  - `axis_size` the font size for the axis labels (default 0.8)

```
> # bending labels named vector
> reg_blab <- r %>%
+   mutate(lab = str_replace(string = lab, pattern = "\n", replacement = " ")) %>%
+   select(region, lab) %>%
+   deframe()
>
> reg_blab
                        AFRICA                                  ASIA
                     "Africa"                                 "Asia"
                        EUROPE LATIN AMERICA AND THE CARIBBEAN
                     "Europe"     "Latin America & Caribbean"
              NORTHERN AMERICA                               OCEANIA
           "Northern America"                              "Oceania"
>
> mig_chord(d, lab_bend1 = reg_blab, label_size = 1.2,
+           axis_breaks = 10, axis_size = 1)
```

Label Spacing

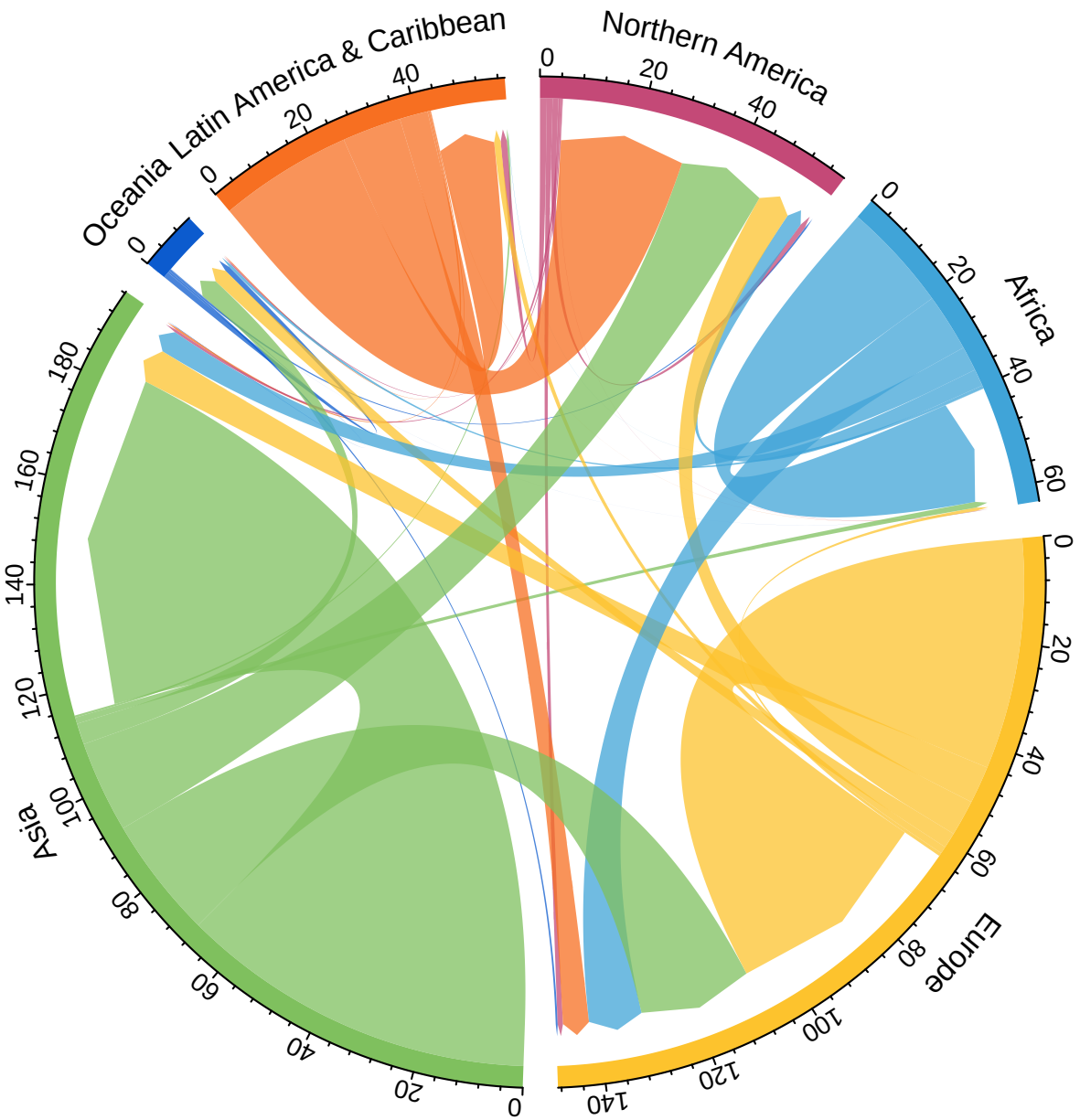


Sector Ordering

- Sector ordering is alphabetical by default
- Can specify order using `order` argument and pass a vector
- Try to order so that neighboring regions are next each other

```
> r <- r %>%
+   mutate(reg_order = c(2, 4, 3, 6, 1, 5)) %>%
+   arrange(reg_order)
> r
# A tibble: 6 x 5
  region                          lab                          lab1  lab2  reg_o~1
  <chr>                           <chr>                        <chr> <chr>   <dbl>
1 NORTHERN AMERICA                "Northern\nAmerica"          Nort~ Amer~       1
2 AFRICA                          "Africa"                     Afri~ <NA>        2
3 EUROPE                          "Europe"                     Euro~ <NA>        3
4 ASIA                            "Asia"                       Asia  <NA>        4
5 OCEANIA                         "Oceania"                    Ocea~ <NA>        5
6 LATIN AMERICA AND THE CARIBBEAN "Latin America\n& Caribbe~ Lati~ & Ca~       6
# ... with abbreviated variable name 1: reg_order
>
> # order sectors
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region)
```

Sector ordering

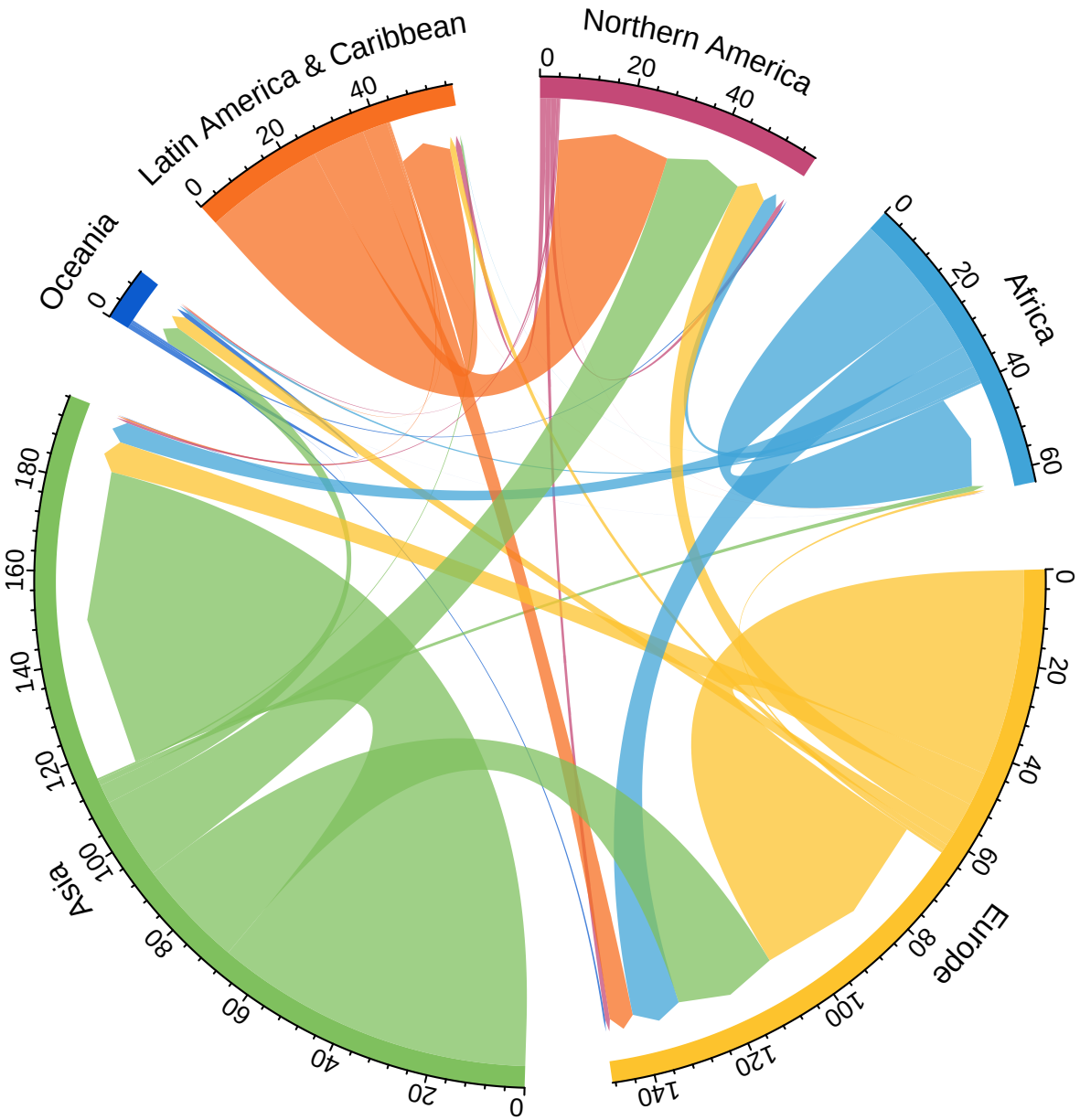


Orientation and gaps

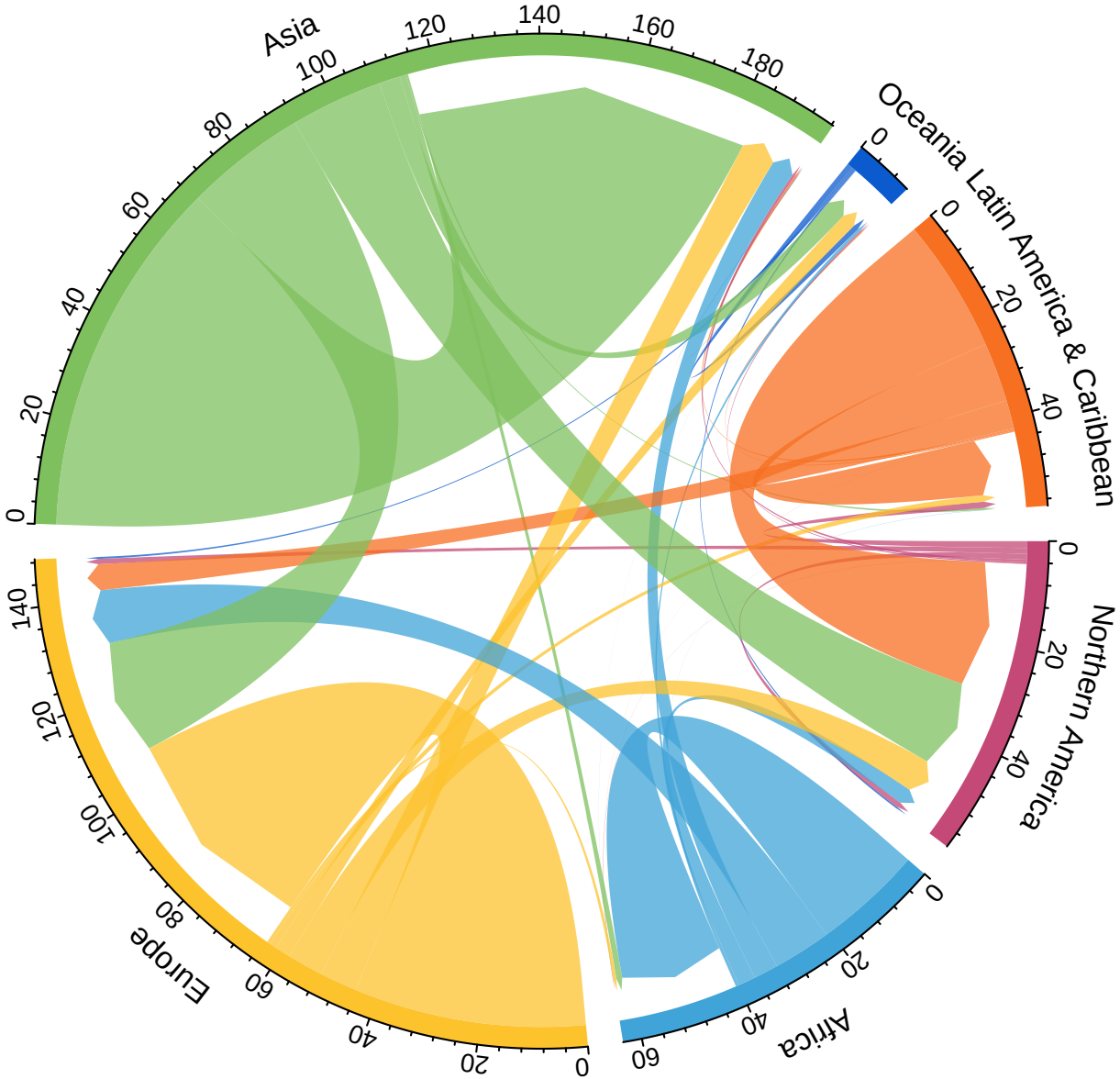
- The `circos.par()` function controls the overall layout parameters of the graphic display
- Pass settings to `circos.par()` via `mig_chord()`
  - `gap.degree` the degree of gaps between sectors are set - default `gap.degree = 1`
  - `start.degree` the degree where the first sector appears (0 corresponds to 3 o'clock, `mig_chord()` default is 90)

```
> # increase gaps
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region, gap.degree = 10)
>
> # rotate
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region, start.degree = 0)
```

Bigger gap - gap.degree = 10



Alternative orientation - `start.degree = 90`

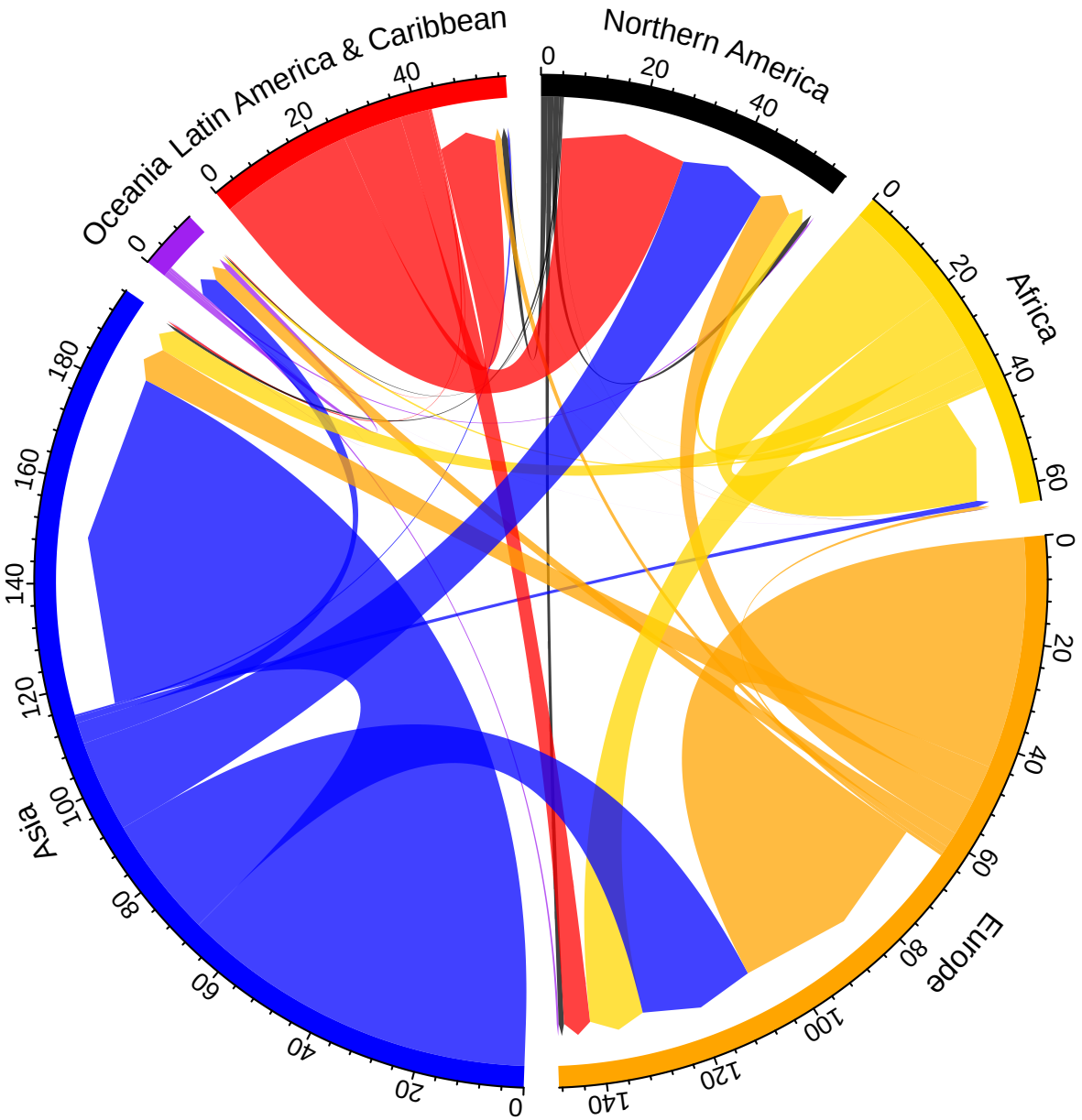


Sector colours

- The `mig_chord()` uses a umbrella colour palette
- Can set to a choice using either:
  - `grid.col` corresponding to sectors (regions/countries/areas)
  - `transparency` set by default to 0.25 in `mig_chord()`

```
> r <- r %>%
+   mutate(col1 = c("black", "gold", "orange", "blue", "purple", "red"))
> r
# A tibble: 6 x 6
  region                          lab                    lab1  lab2  reg_o~1 col1
  <chr>                           <chr>                  <chr> <chr>   <dbl> <chr>
1 NORTHERN AMERICA                "Northern\nAmerica"    Nort~ Amer~       1 black
2 AFRICA                          "Africa"               Afri~ <NA>        2 gold
3 EUROPE                          "Europe"               Euro~ <NA>        3 oran~
4 ASIA                            "Asia"                 Asia  <NA>        4 blue
5 OCEANIA                         "Oceania"              Ocea~ <NA>        5 purp~
6 LATIN AMERICA AND THE CARIBBEAN "Latin America\n& C~ Lati~ & Ca~       6 red
# ... with abbreviated variable name 1: reg_order
>
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region, grid.col = r$col1)
```

Sector colour - set `grid.col`



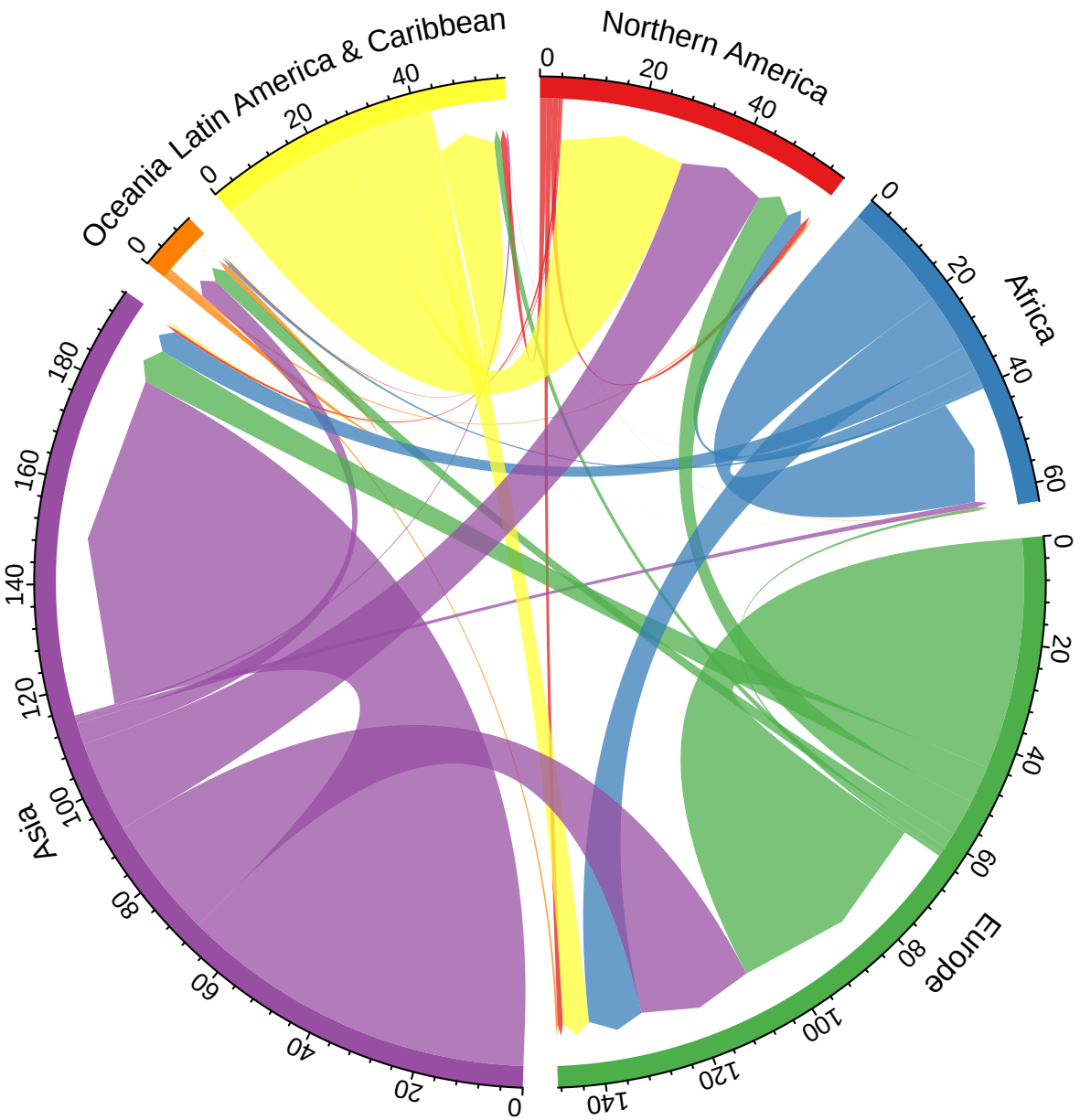
Sector colour

- Can use the *RColourBrewer* package to generate palettes (maximum of 9 colours)
  - Based on `https://colorbrewer2.org/`

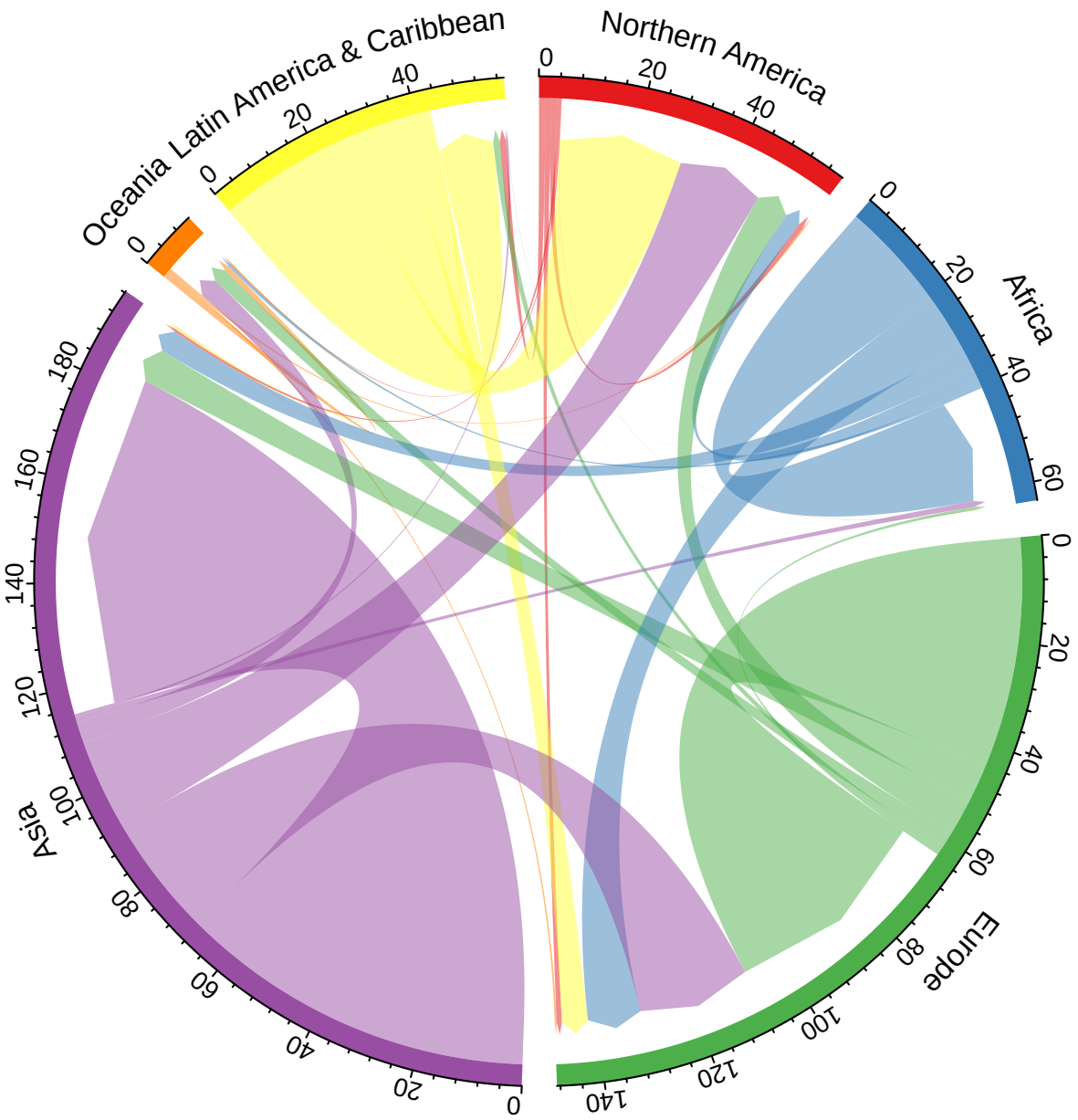
```
> library(RColorBrewer)
> r <- r %>%
+   mutate(col2 = brewer.pal(n = 6, name = "Set1"),
+          col3 = c("Red", rep("Grey", times = 5)))
> r
# A tibble: 6 x 8
  region                          lab      lab1  lab2  reg_o~1 col1  col2  col3
  <chr>                           <chr>    <chr> <chr>   <dbl> <chr> <chr> <chr>
1 NORTHERN AMERICA                "Northe~ Nort~ Amer~       1 black #E41~ Red
2 AFRICA                          "Africa" Afri~ <NA>        2 gold  #377~ Grey
3 EUROPE                          "Europe" Euro~ <NA>        3 oran~ #4DA~ Grey
4 ASIA                            "Asia"   Asia  <NA>        4 blue  #984~ Grey
5 OCEANIA                         "Oceani~ Ocea~ <NA>        5 purp~ #FF7~ Grey
6 LATIN AMERICA AND THE CARIBBEAN "Latin ~ Lati~ & Ca~       6 red   #FFF~ Grey
# ... with abbreviated variable name 1: reg_order
```

```
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region, grid.col = r$col2)
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region, grid.col = r$col2,
+           transparency = 0.5)
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region, grid.col = r$col3)
```

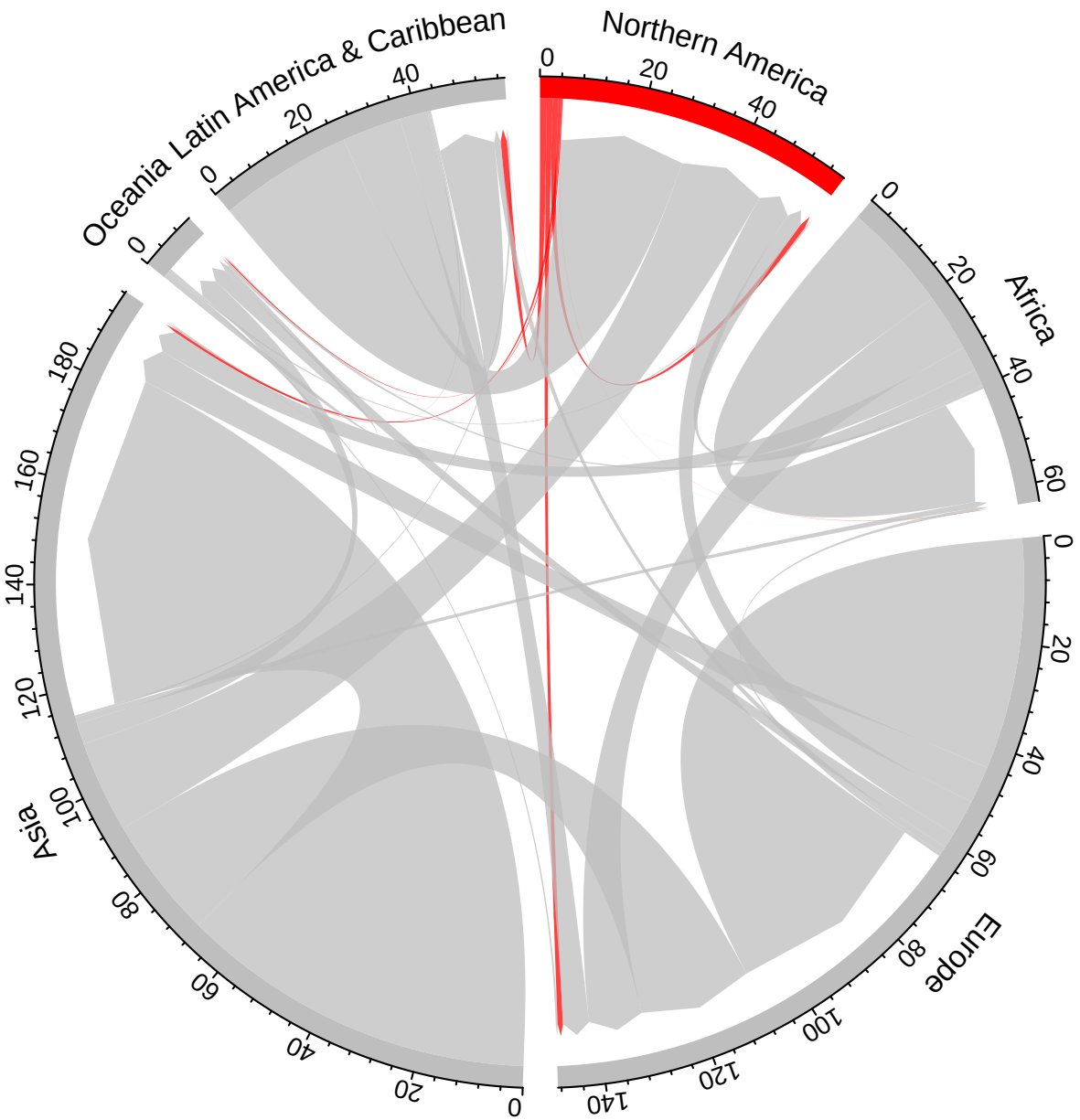
Sector colour - set `grid.col`



Sector colour - transparency = 0.25



Sector colour - set `grid.col` to highlight



Chord colours

- Chord colours follow the origin sector. We can specify different colours using
  - `col` corresponding to links (bilateral migration data)
  - `link.visible` will hide particular chords

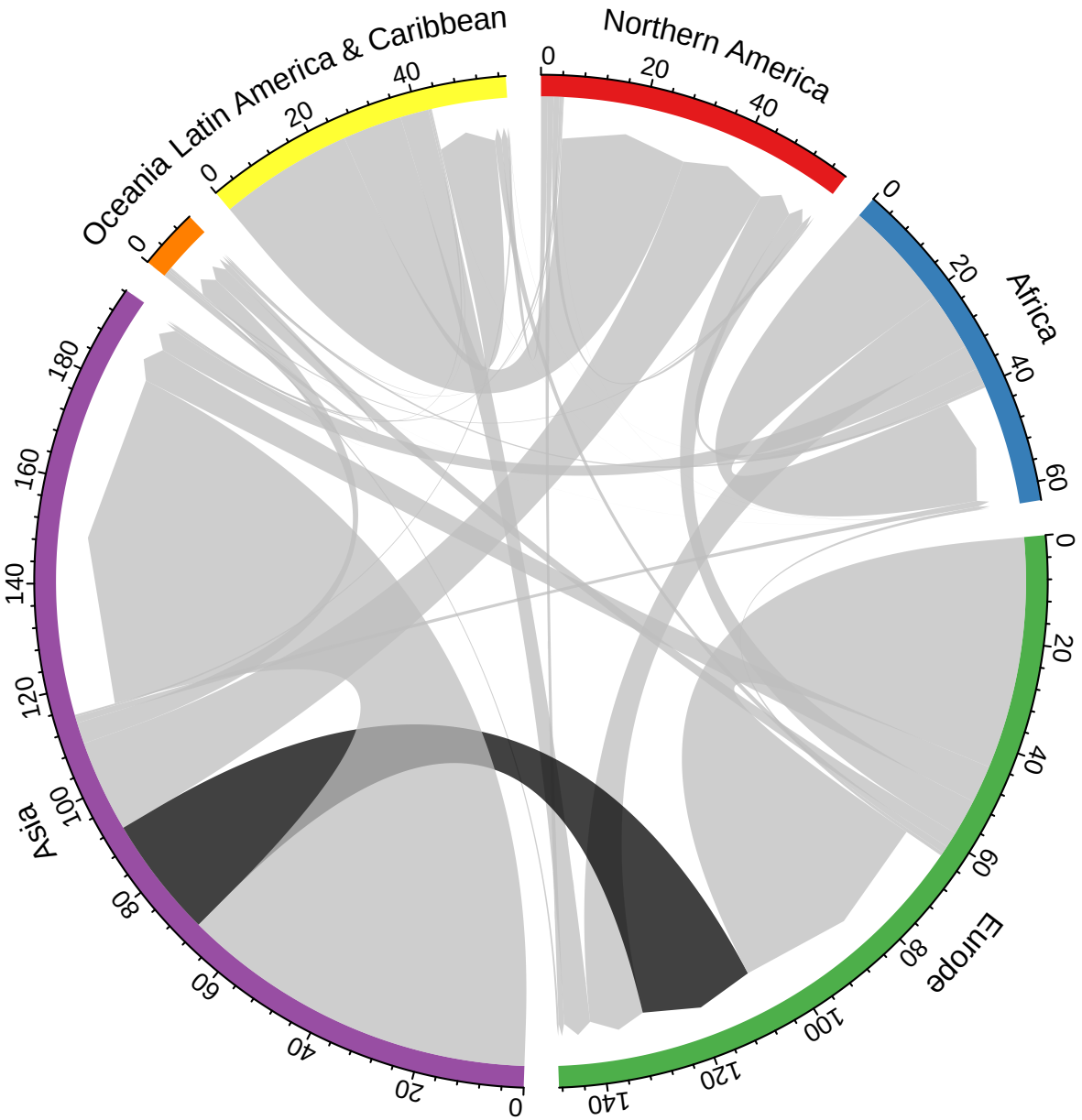
```
> d <- d %>%
+   # highlight Asia to Europe flows
+   mutate(link_col1 = ifelse(test = orig == "ASIA" & dest == "EUROPE",
+                             yes = "black", no = "grey"),
+          # show only flows out or into Asia
+          show_link = orig == "ASIA" | dest == "ASIA")
> d
# A tibble: 36 x 5
   orig                              dest     stock link_col1 show_link
   <chr>                             <chr>    <dbl> <chr>     <lgl>
 1 AFRICA                            AFRICA 20.9    grey      FALSE
 2 ASIA                              AFRICA  1.21   grey      TRUE
 3 EUROPE                            AFRICA  0.648  grey      FALSE
 4 LATIN AMERICA AND THE CARIBBEAN   AFRICA  0.0325 grey      FALSE
 5 NORTHERN AMERICA                  AFRICA  0.0536 grey      FALSE
 6 OCEANIA                           AFRICA  0.0145 grey      FALSE
 7 AFRICA                            ASIA    4.72   grey      TRUE
 8 ASIA                              ASIA   68.5    grey      TRUE
 9 EUROPE                            ASIA    7.17   grey      TRUE
10 LATIN AMERICA AND THE CARIBBEAN   ASIA    0.415  grey      TRUE
#     with 26 more rows
```

Chord colours

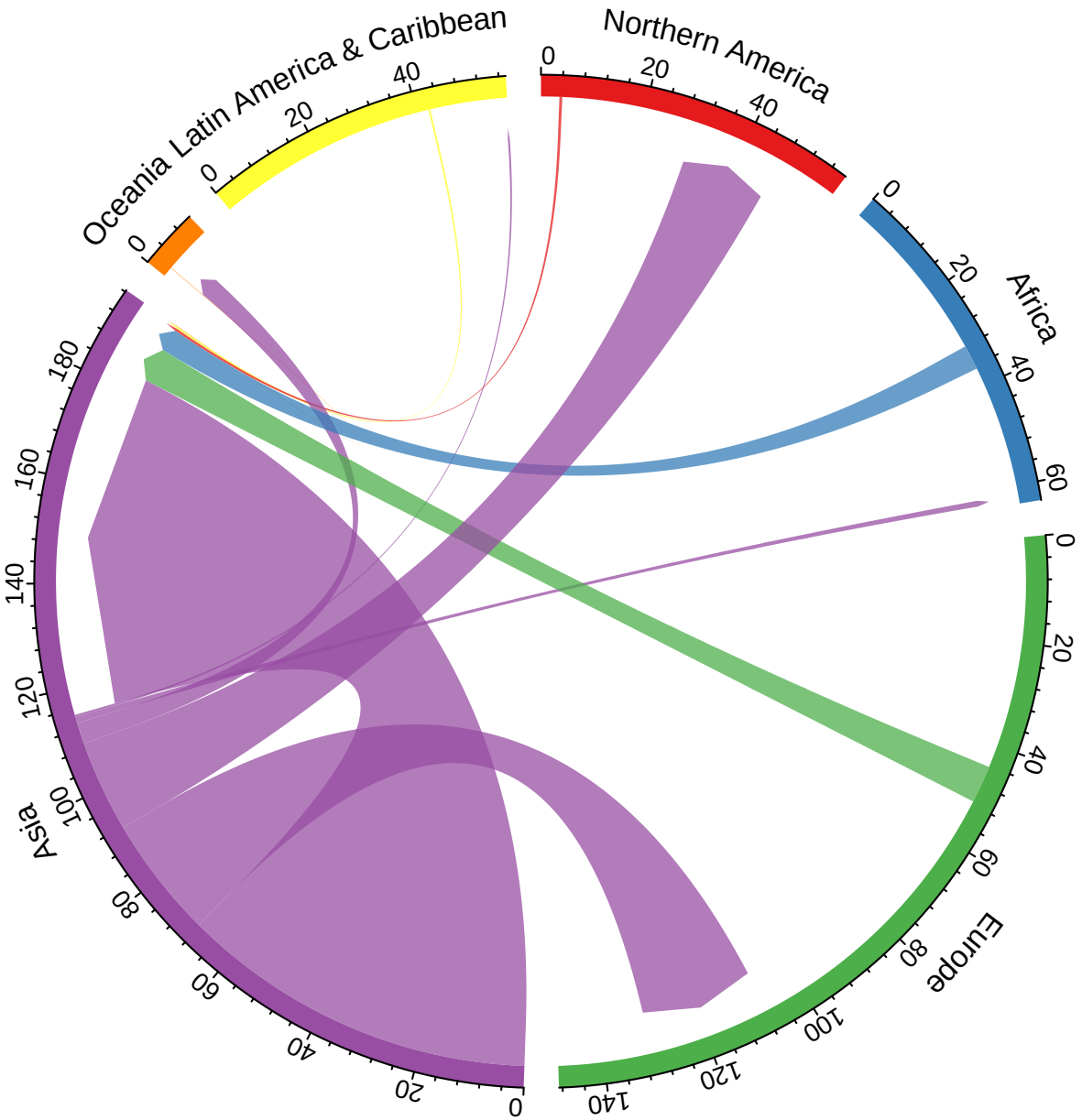
- Pass the chord specific settings to `chordDiagram()`

```
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region,
+           grid.col = r$col2, col = d$link_col1)
> mig_chord(x = d, lab_bend1 = reg_blab, order = r$region,
+           grid.col = r$col2, link.visible = d$show_link)
```

Chord colours - setting `col` to column in bilateral data



Chord colours - setting `link.visible`



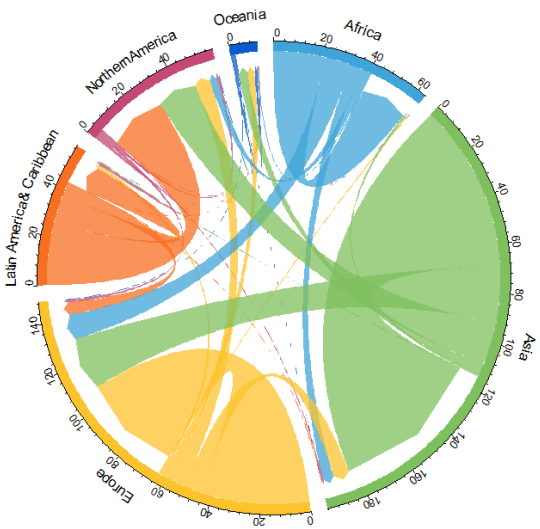
Saving

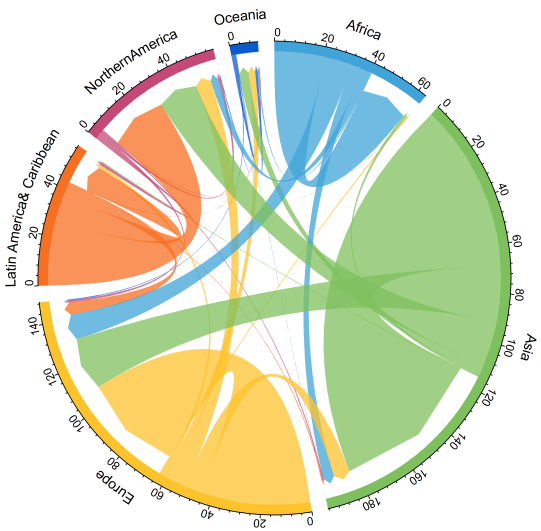
- Always save as PDF to give scalable image
  - We can zoom in very closely and we will still see the chords
  - If we save a vector graphic, e.g. PNG these details will disappear.
- Use the `pdf()` function before the plot to open a PDF
- Use `dev.off()` after the plot code to close the PDF
- The layout of the plots in `mig_chord()` are designed to specifically work on plotting images into PDF devices with widths and heights of 7 inches (the default dimension when using the `pdf()` function).

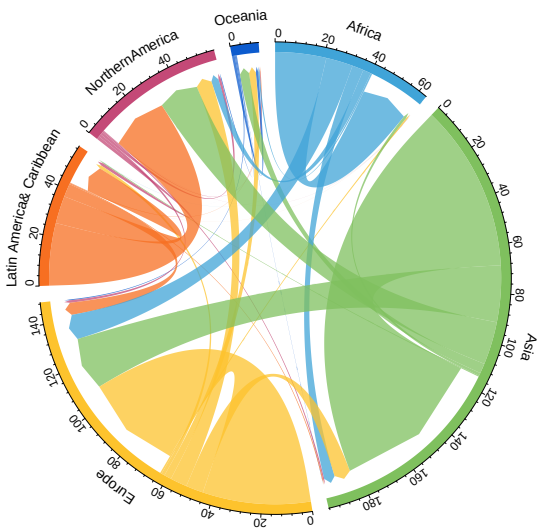
```
> pdf(file = "./slides-md/plot/un_stock_2019.pdf")
> mig_chord(x = d, lab_bend1 = reg_blab)
> dev.off()
>
> png(file = "./slides-md/plot/un_stock_2019.png", width = 7, height = 7,
+     units = "in", res = 100)
> mig_chord(x = d, lab_bend1 = reg_blab)
> dev.off()
>
> png(file = "./slides-md/plot/un_stock_2019b.png", width = 7, height = 7,
+     units = "in", res = 500)
> mig_chord(x = d, lab_bend1 = reg_blab)
> dev.off()
```

Saving

- Left: PNG with resolution 100 pixels per inch (25kb)
- Middle: PNG with resolution 500 pixels per inch (199kb)
- Right: PDF (56kb)







- Could increase resolution of PNG with larger dimensions but at the cost of very large file sizes

Exercise (ex8.R)

```
# 0.  a) Load the KOSTAT2022.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2022.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(circlize)
library(migest)
##
##
##
# 1. Run the code below to read in the label data in korea_cd_labels.csv taken
#    from https://www.tandfonline.com/doi/full/10.1080/21681376.2018.1431149
r <- read_csv("./data/korea_cd_labels.csv")
View(r)
# 2. Run the code below to select the 2020 Korean internal migration data,
#    for plotting, excluding within region movements
d <- korea_reg %>%
  filter(year == 2020,
         orig != dest)
d
# 3. Run the code below to check that all the regions in the object r are in the
#    migration data frame d
setdiff(x = union(d$orig, d$dest), y = r$region)
# 4. Modify d to enable a more sensible plot
```

References

Abel, Guy J., and Nikola Sander. 2014. "Quantifying Global International Migration Flows." *Science* 343 (6178): 1520–22. https://doi.org/10.1126/science.1248676.

Gu, Z., Lei Gu, Roland Eils, Matthias Schlesner, and Benedikt Brors. 2014. "circlize implements and enhances circular visualization in R." *Bioinformatics* 30 (19): 2811–12. https://doi.org/10.1093/bioinformatics/btu393.

Houtum, Henk van, and Rodrigo Bueno Lacy. 2020. "The migration map trap. On the invasion arrows in the cartography of migration." *Mobilities* 15 (2): 196–219. https://doi.org/10.1080/17450101.2019.1676031.

Krzywinski, Martin, Jacqueline Schein, Inanç Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J. Jones, and Marco A. Marra. 2009. "Circos: An information aesthetic for comparative genomics." *Genome Research* 19 (9): 1639–45. https://doi.org/10.1101/gr.092759.109.