VIENNA INSTITUTE OF DEMOGRAPHY

Working Papers

02 / 2014

Nikola Sander, Guy J. Abel, Ramon Bauer and Johannes Schmidt

Visualising Migration Flow Data with Circular Plots

Abstract

Effective visualisations of migration flows can substantially enhance our understanding of underlying patterns and trends. However, commonly used migration maps that show place-to-place flows as stroked lines drawn atop a geographic map fall short of conveying the complexities of human movement in a clear and compelling manner.

We introduce circular migration plots, a new method for visualising and exploring migration flow tables in an intuitively graspable way. Our approach aims to provide detailed quantitative information on the intensities and patterns of migration flows around the globe by using a visualization design that is effective and visually appealing. The key elements of the design are (a) the arrangement of origins and destinations of migration flows in a circular layout, (b) the scaling of individual flows to allow the entire system to be shown simultaneously, (c) the expression of the volume of movement through the width of the flow and its direction through the colour of the origin.

Drawing on new estimates of 5-year bilateral migration flows between 196 countries, we demonstrate how to create circular migration plots at regional and country levels using three alternative software packages: Circos, R, and the JavaScript library d3.js.

Circular migration plots considerably improve our ability to graphically evaluate complex patterns and trends in migration flow data, and for communicating migration research to scientists in other disciplines and to the general public. Our visualisation method is applicable to other kinds of flow data, including trade and remittances flows.

Keywords

Migration, flow data, visualisation, D3

Authors

Nikola Sander, Research Scholar, Wittgenstein Centre (IIASA, VID/ÖAW, WU), Vienna Institute of Demography, Austria. Email: nikola.sander@oeaw.ac.at

Guy J. Abel, Research Scholar, Wittgenstein Centre (IIASA, VID/ÖAW, WU), Vienna Institute of Demography, Austria. Email: guy.abel@oeaw.ac.at

Ramon Bauer, Wittgenstein Centre (IIASA, VID/ÖAW, WU), Vienna Institute of Demography & University of Vienna, Department of Geography and Regional Research, Austria. Email: ramon.bauer@univie.ac.at

Johannes Schmidt, Null2 GmbH, Berlin, Germany. Email: johannes@null2.net

Acknowledgements

We thank Michael Holzapfel for helping with using Circos on Windows, Andi Pieper for his help with the interactive visualization, Martin Krzywinski for Circos, Mike Bostock for d3.js, and Zuguang Gu for the circlize package. We also thank Neil Bailey, Elin Charles-Edwards, Adam Dennett and Francisco Rowe Gonzales for testing the replicability of our plots.

Visualising Migration Flow Data with Circular Plots

Nikola Sander, Guy J. Abel, Ramon Bauer and Johannes Schmidt

1 Introduction

The global flow of people is a complex and dynamic phenomenon. Discourses on migration in science, politics, and the media typically focus on current volumes of migration in a single country. However, flows to and from a given country can be much better understood when placed within a global context. Using such a comparative perspective has been hindered by the lack of both adequate data and visualisation techniques. Data on international migration flows measured over a specific time period suffer from incompleteness and incompatibility, mainly because national statistical agencies measure and/or define migration in different ways. In addition, migration flows are typically visualised using migration maps, which show flows as stroked lines drawn atop a geographic map. This visualisation clearly falls short of conveying the complexities of human movement in an effective and visually appealing manner.

Abel & Sander (2014) introduced circular migration plots as a visualisation method for their estimates of the global flow of people between 196 countries for four time periods, 1990-95 to 2005-10. Circular migration plots are, as we argue, an effective and visually appealing method for visualizing flow data that enables the reader to grasp intuitively the key patterns of international migration. In addition, we have developed an interactive data visualisation "The Global Flow of People" [1] that enables us to communicate our findings to scientists in other disciplines and interested members of the general public in an engaging way. Created using the open-source JavaScript library d3.js (Bostock *et al.* , 2011), this online tool invites the user to explore global migration flows for four time periods at region and country levels.

In the present paper, we provide a detailed step by step description of how to construct circular migration plots for print and online media, so as to facilitate its use in migration studies and beyond. Drawing on new estimates of bilateral flows between 196 countries from 1990 through 2010 (see Abel, 2013; Abel & Sander, 2014, for details), we illustrate the method for producing circular plots using three alternative open-source software packages. While we mostly use Circos, software developed by Martin Krzywinski (Krzywinski *et al.* , 2009) for the visualisation of genetic data, we

[1]available at `http://www.global-migration.info`

also demonstrate how to draw circular plots using R and d3.js. The migration data and configuration files use in these examples are provided in the supplementary material.

The remainder of this paper is organised as follows. The next section summarises existing techniques for visualising migration flows including flow maps. Section 3 introduces the circular migration plot and discusses its key features. Section 4 describes how to use Circos to create custom migration plots. Section 5 outlines the steps involved in creating circular plots in R via the circlize package (Gu, 2013), and Section 6 shows how do develop interactive migration plots in d3.js. A last section offers some concluding remarks on the pros and cons of each of the alternative software packages and future avenues for visualising scientific data.

2 Existing Techniques for Visualising Migration Flows

The visualisation and communication of migration flow data is difficult because of the high degree of complexity that the data exhibit. Place-to-place flows are typically represented by an origin-destination matrix with rows for outflows from origins and columns for inflows to destinations. The diagonal element of the matrix represents flows within the same region and is thus often set to missing in datasets on international migration. Each off-diagonal cell in the matrix contains information on the volume of movement between an origin-destination pair. These three aspects of the data that need to be encoded visually (origin, destination, volume) can be extended to also include changes over time in the volume of movements. The number of spatial interactions (i.e. migration flows) contained in a dataset rapidly exceeds a size that can be analysed and visualised using standard visualisation tools, including line and bar charts, scatterplots, cartograms and choropleth maps. For instance, the dataset on global migration flows between 196 countries across four time periods contains $196 * 195 * 4 = 152,880$ interactions.

The complexity of migration flow data has led researchers to focus on net migration, other aggregate measures of movement and specific origin-destination pairs that can be shown using standard visualisation tools. Examples include the recent works by Özden *et al.* (2011) and Czaika & de Haas (2013) on global stock matrices, and various analyses of migration from Mexico to the United States (see for example Cerrutti & Massey, 2001). The static, two-dimensional representations of migration data used in the literature clearly fall short of illuminating the complex spatio-temporal patterns and trends of country-to-country migration flows.

To address this deficiency, flow maps have been proposed as a visualisation technique capable of depicting movement of people across regions and countries. The concept of creating flow maps by placing stroked lines on top of a geographic map dates back more than 150 years to the seminal visualisations of Charles Joseph Minard (Figure 1). The hand drawn map depicts the origins, destinations and volumes of migration flows in 1858, with one millimetre equalling 1,500 people. The coloured lines in this flow map represent information contained in a migration matrix on place, direction and volume. Minard's flow map is relatively effective and visually appealing because it shows only a small number of flows.

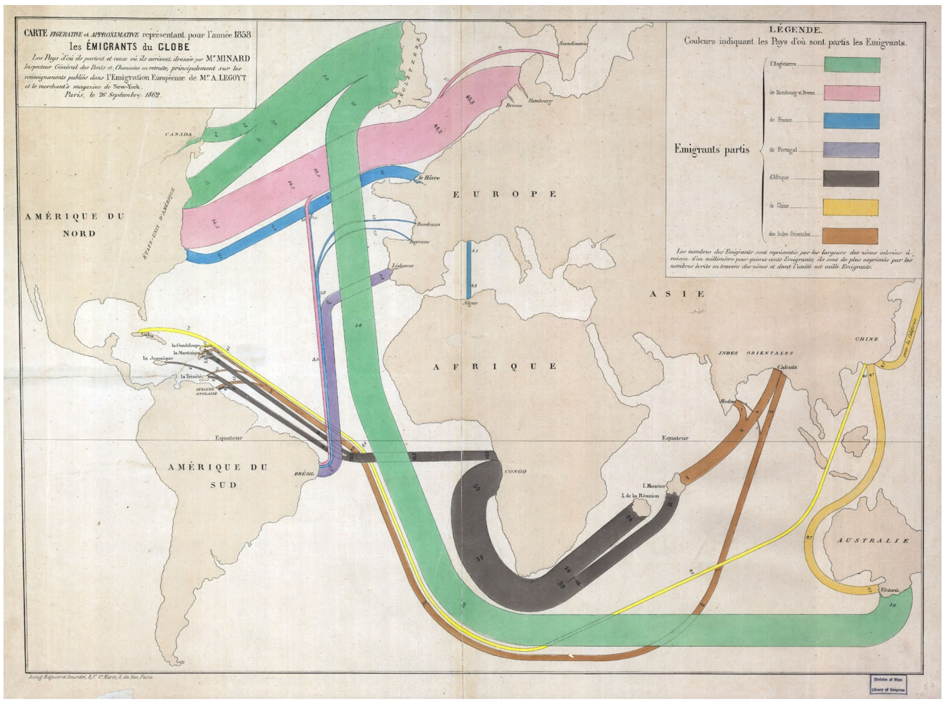
CARTE FIGURATIVE et APPROXIMATIVE représentant pour l'année 1858
les ÉMIGRANTS du GLOBE
Les Pays d'où ils partent et ceux où ils arrivent. dressée par Mr MINARD
Inspecteur Général des Ponts et Chaussées en retraite, principalement sur les renseignements publiés dans l'Emigration Européenne de Mr A. LEGOYT et le merchant's magazine de New-York.
Paris, le 26 Septembre 1862.
LÉGENDE.
Couleurs indiquant les Pays d'où sont partis les Emigrants.
Emigrants partis
d'Angleterre
de Hambourg et Breme
de France
de Portugal
d'Afrique
de Chine
des Indes Orientales
Les nombres des Emigrants sont représentés par les largeurs des zônes colorées à raison d'un millimètre pour quinze cents Emigrants; ils sont de plus exprimés par les nombres écrits en travers des zônes et dont l'unité est mille Emigrants.
CANADA
AMÉRIQUE DU NORD
EUROPE
ASIE
AFRIQUE
AMÉRIQUE DU SUD
INDES ORIENTALES
CHINE
Calcuta
pour la Californie
AUSTRALIE
Equateur

Figure 1: The Emigrants of the World, 1858. By Charles Joseph Minard.[2]

One of the main drawbacks of flow maps is that they work well only for small datasets with limited complexity. The visualisation of a larger migration matrix shown in Figure 2 provides an example of a cluttered flow map with overlaying arrows indicating the size and direction of flows. The dominance of London as an origin of elderly migration in England highlights the fallacy of using geographic maps for the encoding of spatial locations of migrant origins and destinations. Despite recent improvements in geographic information systems, the graphical features of flow maps have changed little since the seminal works by Minard and Ravenstein (1885) in the 19th century,

[2]Retrieved from `http://cartographia.wordpress.com/category/charles-joseph-minard`; last accessed 25.11.2013

notwithstanding the contributions of Beddoe (1978), Dorling (2012), Geertman *et al.* (2003), Tobler (1987), Tufte & Graves-Morris (1983) and others.

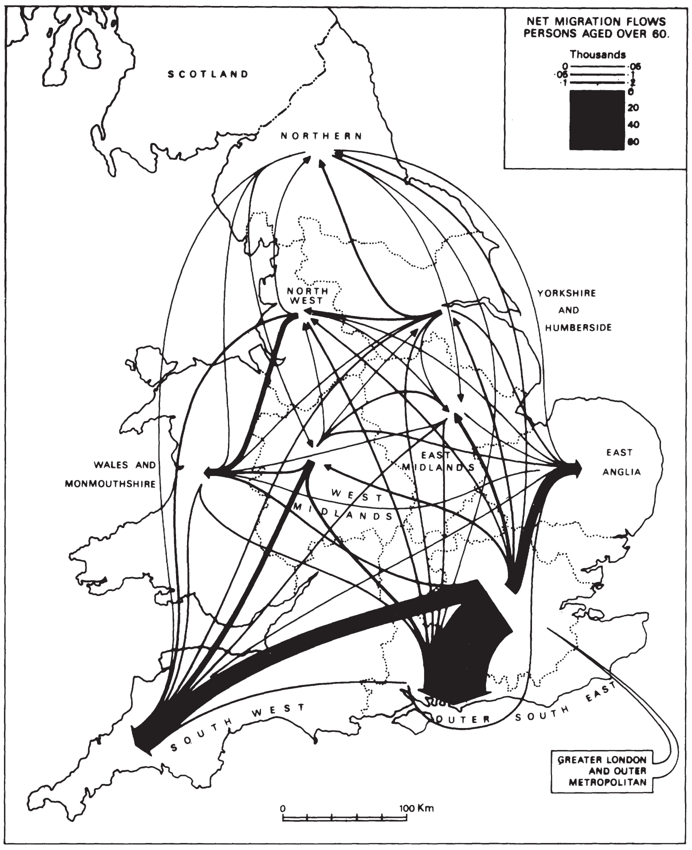

Figure 2: Net inter-Regional migration flows of those aged 60 or more, 1961-66. Source: Law & Warnes (1976, p. 464)

The challenge in the development of flow maps has been to create visualisations that are truly appropriate to the data. Migration flow data tend to be highly skewed with many cells in the matrix being close to zero and a smaller number of flows of large size. The skewed distribution of flows and the dominance of large cities as migrant hubs make it difficult or even impossible to produce an effective and engaging flow map of a migration system. Moreover, flow map tools are a surprisingly underdeveloped segment of the proprietary GIS packages (e.g. ArcGIS, Mapinfo). Consequently, recent advances in visualising spatial interaction data have been made outside the GIS community. A range of alternative designs that aim at visualising flow data more effectively have been proposed by geographers, computer scientists and demographers. Distributive flow maps, for example, display migration as flow trees that are drawn on a geographic map (see, for example, Fassmann & İçduygu, 2013; Verbeek *et al.* , 2011). The trunk of the flow tree denotes the origin of migrants and splits into as many smaller branches as there are destinations. The hierarchical structure can falsely give the impression of key migration corridors. Using flow trees to visualize migration poses an additional difficulty: scaling individual flows (i.e. branches) proportionally to its volume, and ensuring that the width of the flows adds up to that of the trunk at the origin. Other layouts that have been proposed in the literature for representing migration are corrgrams (Bryant, 2011), spatial clustering (Guo *et al.* , 2012) and raster/vector overlay (Rae, 2009). During the past few years, mobility and migration have become of increasing interest for non-scientific data visualization applications on the web. PeopleMovin[3] and MigrationsMap[4] are two examples of existing interactive visualisations of migration. To summarise the key points, PeopleMovin displays migrant stock data published by the World Bank using slopegraphs between two columns, the emigration countries on the left and the destination countries on the right. While visually appealing, the need for extensive scrolling to find the destination country of a particular flow prevents the user from obtaining a full picture of the global patterns. MigrationsMap is a somewhat old-fashioned version of a traditional flow map based on the Migration DRC database of foreign-born populations. By select a country, the user can explore the key origins of immigrants or the destinations of emigrants without getting a comprehensive view of the overall global patterns.

3 Introducing circular migration plots

Circular migration plots are, as we argue, effective, engaging and beautiful visualisations that are more appropriate to flow data than any of the existing graphical representations discussed earlier. Perhaps even more importantly, our detailed step by step descriptions of how to construct the plot allow others who may wish to visualise their data in a circular layout to do this with relatively little additional effort or cost. This section discusses the key features of this plot.

As noted by Heer *et al.* (2010, p. 1), the creation of visualisations requires the selection of "effective visual encodings" the data attributes can be linked to. For instance, time-series data are best represented as (indexed) line charts, whereas differences in GDP across European countries are best shown through a colour encoding of the geographic areas in a choropleth map.

Visualisations of migration flows have to depict a large amount of information. The most important items are the locations of origin and destination, the volume of

[3]Retrieved from `http://peoplemov.in`; last accessed 25.11.2013

[4]Retrieved from `http://migrationsmap.net`; last accessed 25.11.2013

movement, and the direction of the flow. In addition, visualisations should highlight the relations between individual place-to-place flows, the main corridors of movement, and the key senders and receivers of migration.

A circular migration plot displays a large amount of information on the flow of people in a global or regional system as representations of position, size, length and colour. Specifically, the plot uses the following visual encodings (see Figure 3):

- The origins and destinations of migrants are represented by the circle's segments. Countries or regions on the same continent are typically positioned close to each other, and each country or region is assigned a unique colour.

- The volume of movement is indicated by the width of the flow, which, for clarity are encoded as curved rather than straight lines. Consequently, the width of each flow has to be non-linearly adapted to the curvature. Hence, the width of the flow indicates the exact volume of movement only at the beginning and end points. The number of migrants that one millimetre equals to depend on the total volume across all regions and is indicated by tick marks on the outside of the circle's segments.

- The direction of the flow is encoded by the origin colour. In the example shown in Figure 3, Region A is assigned a red colour, Region B is encoded green, Region C is encoded blue and Region D is encoded orange. Hence, the flow from Region D to Region A is coloured orange, and the much smaller counter flow from Region A to Region D is coloured red. The direction of the flow is also indicated by a gap between flow and circle segment at the destination.

- Information on how individual flows relate to each other is given by visualising all flows in the global system in one plot, and by scaling the volume of each flow to double the system-wide movement, which is equal to the size of the circle (all inflows plus all outflows).

- The main corridors of movement are identified through the relative differences in the width of flows. Longer-distance flows between continents tend to go through the centre of the circle (e.g. Region C to Region A in Figure 3).

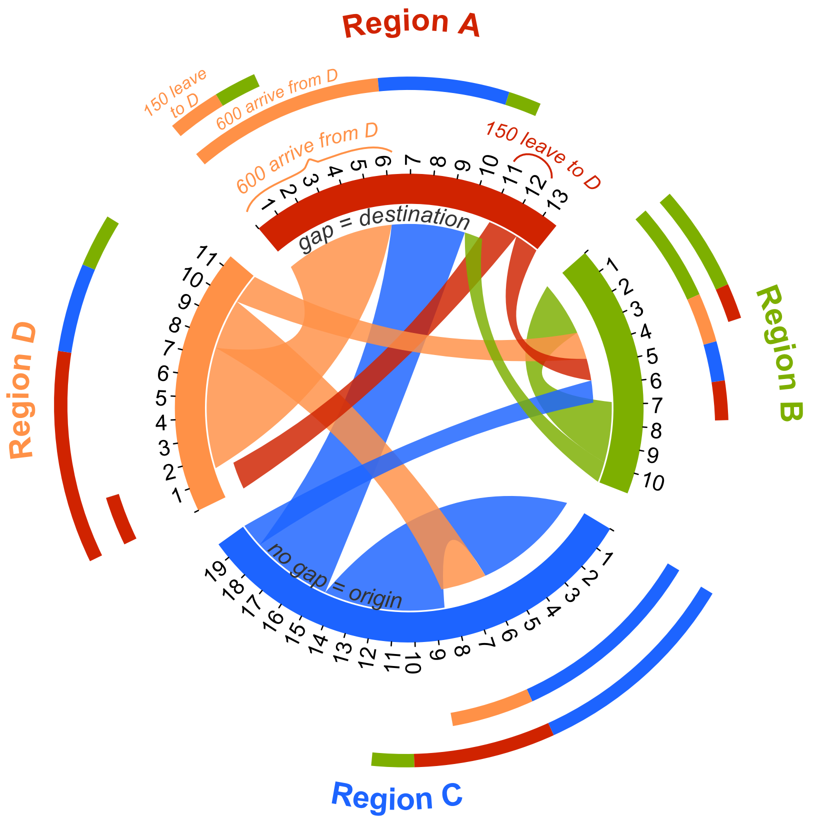

Figure 3: A simple circular migration plot using hypothetical data. Ticks indicate the number of migrants in 100s.

- The longer the circle's segments, the higher the number of migrants moving to and/or from this region. In Figure 3, Regions C sends and receives the most migrants, whereas Region B sends and receives the least number of migrants.
- The stacked bars on the outside of the circle provide additional information on the net gain or loss. The inner bar shows the total volume of immigration, subdivided by origin colour. The outer bar shows emigration by destination colour. Placing the two stacked bars on top of each other yields the total volume of migration that is encoded by the circle's segments. In Figure 3, Region A receive more migrants than it sends out, whereas Regions C and D send more migrants than they receive.

Creating an effective and visually appealing circular migration plot requires three important additional judgements. First, because of the skewness of the distribution of place-to-place flows, it is not ideal to display all flows in the matrix. The smallest flows are represented by very thin lines, making it difficult to identify origin, destination and direction of flows. Moreover, the visual clutter that is created by showing each and every flow distracts the reader from the key patterns and trends. The threshold, that is the volume of movement below which a flow is not shown, has to be decided based on the number of countries/regions that are shown in the plot and the distribution of flows. The higher the number of countries and the more skewed the distribution, the higher tends to be the threshold. Based on our experience, we suggest to show the largest 65 to 90% of flows.

Second, the ability of the reader to intuitively understand the plot is improved by allowing gaps between the circle's segments. The gaps need to be large enough to ensure that the reader can easily distinguish between flows to and from individual countries, but without unnecessarily limiting the lengths of the segments.

Third, the clarity and visual appeal of the circular plot is enhanced by showing a country's immigration and emigration flows separately, and by sorting flows by their size in descending order. This approach ensures that the largest migrant flows can be easily identified, For example, in Figure 3, each region's immigration flows are shown first, followed by emigration flows, and individual flows are sorted by their width in descending order.

Circular migration plots cannot be created using commercial off-the-shelf software. Instead, we use Circos, a software originally developed for the graphical representation of genomic data (Krzywinski *et al.* , 2009). Circos uses a special colour coding scheme to show similarities between chromosomes and genomes, which can be readily applied to visualise migration flow data. The next section provides a detailed description of how to construct circular migration plots in Circos. Input data and configuration files are provided in the supplementary material.

4 Creating circular plots using Circos

Circos consists of a suite of Perl scripts and configuration files with no user interface (Krzywinski *et al.* , 2009). Circular migration plots are created using the Circos tools add-on TABLEVIEWER.

4.1 Installing Perl, Circos and Cygwin

To run Circos on Windows 7, Perl, several Perl modules and Cygwin need to be installed.[5]

[5]For details on how to install and run Circos on other platforms see `http://circos.ca/tutorials/lessons/configuration/distribution_and_installation`; last accessed 26.11.2013) and Schenk (2012)

1. INSTALL STRAWBERRY PERL ON THE C DRIVE:

   Open command prompt (`cmd` in the Windows run dialog) and type

   ```
   perl -v
   ```

   to check if Strawberry Perl is already installed on your computer.

   If Perl has been installed properly you get a message similar to "This is perl 5, version 12, subversion 4 (v5.12.4) built for MSWin32-x64-multi-thread"

   If the installed version is 4 or older, or if you get an error message, install Strawberry Perl by first closing command prompt and then downloading the Perl Installer from www.strawberryperl.com

   Run the Strawberry Perl Installer (accept all default settings).

   To check that Perl is now installed, re-open the command prompt and type

   ```
   perl -v
   ```

2. INSTALL PERL MODULES:

   Perl modules can be installed either by using CPAN (requires internet access) or Strawberry's "Build" process.

   2.1. TO INSTALL MODULES USING CPAN:

   In the command prompt, enter the CPAN shell by typing

   ```
   perl -MCPAN -e shell
   ```

   You should get a message similar to "cpan shell - CPAN exploration and modules installation v1.9600"

   Install one by one the Perl modules listed below by typing

   ```
   install Color::Library
   ```

   Note that "Color::Library" is the first module listed below.

   You get a long message documenting the installation process. Repeat this step until all modules are installed[6].

[6] See `http://circos.ca/tutorials/lessons/configuration/perl_and_modules` for details.

- Color::Library
- Config::General
- Font::TTF
- GD
- Graphics::ColorObject
- List::MoreUtils
- Math::Bezier
- Math::Round
- Math::VecStat
- Params::Validate
- Pod::Readme
- Readonly
- Regexp::Common
- Set::IntSpan
- Statistics::Descriptive
- Test::Pod
- Test::Pod::Coverage
- Test::Portability::Files
- Text::Format
- Tie::Sub

Exit CPAN by typing

```
exit
```

in the command prompt. Close the command prompt.

2.2. To install modules using Strawberry's "Build" process:

Save the Perl modules provided in supplementary material to *C:/Strawberry/cpan/build/*

In the command prompt, move to the folder that contains the module "Color::Library" by typing

```
cd C:/Strawberry/cpan/build/Color-Library-0.021/
```

To install the module, type:

```
perl makefile.pl
dmake
dmake install
```

Repeat these commands to install the other modules listed above. Note that commands for the modules "Params::Validate" and "Readonly" differ!

To install these two modules, type:

```
perl build.pl
Build
Build install
```

Close the command prompt.

3. Install Circos:

   Download Version 0.64 of the Circos code distribution from
   www.circos.ca/software/download/circos.
   The file is called ”circos-0.64.tgz”. Do not use a later (bug) release!

   Download Version 0.16 of the Circos utility add-on script ”tools” from
   www.circos.ca/software/download/tools.
   The file is called ”circos-tools-0.16.tgz”. Again, do not use a later (bug) release.

   Unpack the archived Circos code distribution to a newly created folder called *circos* on the C drive. Within that folder, save the ”tools” add-on script in a new subfolder called *tools*. Only "tableviewer" is required for drawing circular migration plots.

   For drawing a new circular plot (see Section 4.2.4), make a copy of the entire *tableviewer* folder and re-name it to *my_first_plot*. Delete all example images in *my_first_plot/img/*

   The folder structure should look like this:

```
c:/
  circos/
    tools/
        tableviewer/
            bin/
            data/
            etc/
            img/
            lib/
        my_first_plot/
            bin/
            data/
            etc/
            img/
            lib/
```

4. Install Cygwin:

Because Circos was developed for Unix systems, the Cygwin Terminal from www.cygwin.com
needs to be installed to create a unix-like environment for Windows.[7]

Install the basic Cygwin environment by running the setup.exe program (accept all default settings).

Open the Cygwin Terminal and check that all Perl modules are installed correctly:

```
cd "C:/circos/bin"
```

```
./test.modules
```

5. THE INSTALLATION IS COMPLETED.

4.2 Creating a plot with Circos' default settings

4.2.1 The data and configuration files to be used

To create a circular plot with Circos' default settings (see Figure 4), we use the configuration files you downloaded from the Circos website and saved in *.../my_first_plot/* together with the data input file *region_default.txt* provided in the supplementary material. The data file contains global migration flow estimates aggregated to world regions (source: Abel & Sander (2014)). Save *region_default.txt* in *.../my_first_plot/*.

Before creating the plot, you must change the "housekeeping" settings in the file "circos.conf", which is saved in *.../etc/*. Open the file in a text editor (e.g. Notepad++) and change the following:

- add this code below `«include etc/ticks.conf»`

  ```
  «include etc/housekeeping.conf»
  ```

- delete this code at the bottom of the file:

  ```
  anglestep = 0.5
  minslicestep = 10
  beziersamples = 40
  debug = no
  warnings = no
  ```

[7]Instead of using Cygwin, Circos call also be called through R. To run the examples given in the tutorials on `http://circos.ca/documentation/tutorials/quick_guide/hello_world/` in R:
```
setwd("./circos/helloworld")
shell("perl C:/circos/bin/circos -conf circos.conf")
```

```
imagemap = no
units_ok = bupr
units_nounit = n
```

4.2.2 Parsing the data and drawing the plot

The input data file *region_ default.txt* has to be parsed into an intermediate state before Circos can create the actual plot. The configuration file *parse-table.conf* (saved in *.../etc/*) and the Perl script "parse-table" (saved in *.../bin/*) are used to parse the data file.

- In Cygwin, change the directory to *.../my_ first_ plot/*:
  ```
  cd /cygdrive/c/circos/tools/my_first_plot/
  ```
- Enable edits to all files in this directory:
  ```
  chmod u+x bin/*
  ```
- Parse the input data file *region_ default.txt*:
  ```
  cat region_default.txt | bin/parse-table | bin/make-conf -dir data
  ```
  If the parsing works fine, Cygwin gives a long list of messages similar to:
  ```
  Use of uninitialized value within @_ in lc at C:/strawberry/perl/site/lib/
  Graphics/ColorObject.pm line 1905, <STDIN> line 5.
  ```
  The most common reason for an error message lies in the formatting of the data input text file, such as not deleting all quotation marks if you use your own data.
- Once the data are parsed, draw the circular plot:
  ```
  ../../bin/circos -conf etc/circos.conf
  ```
  If the drawing works fine, Cygwin gives a long list of messages similar to:
  ```
  Use of uninitialized value $key in split at C:/circos/bin/../lib/Circos/
  Debug.pm line 246, <F> line 68.
  ```
  and at the end it gives a summary like:
  ```
  debuggroup summary 0.18s welcome to circos v0.63 25 Jun 2012
  debuggroup summary 0.18s loading configuration from file etc/circos.conf
  ```
  If Circos encounters an error, the message quoted above will be followed by:
  ```
  *** CIRCOS ERROR ***
   CONFIGURATION FILE ERROR
  ```
- Circos saves the newly created plot in SVG and PNG formats in the folder *../my_ first_ plot/img/*. We recommend importing the SVG vector into Inkscape, editing it for appearance, and then saving it as a PDF.

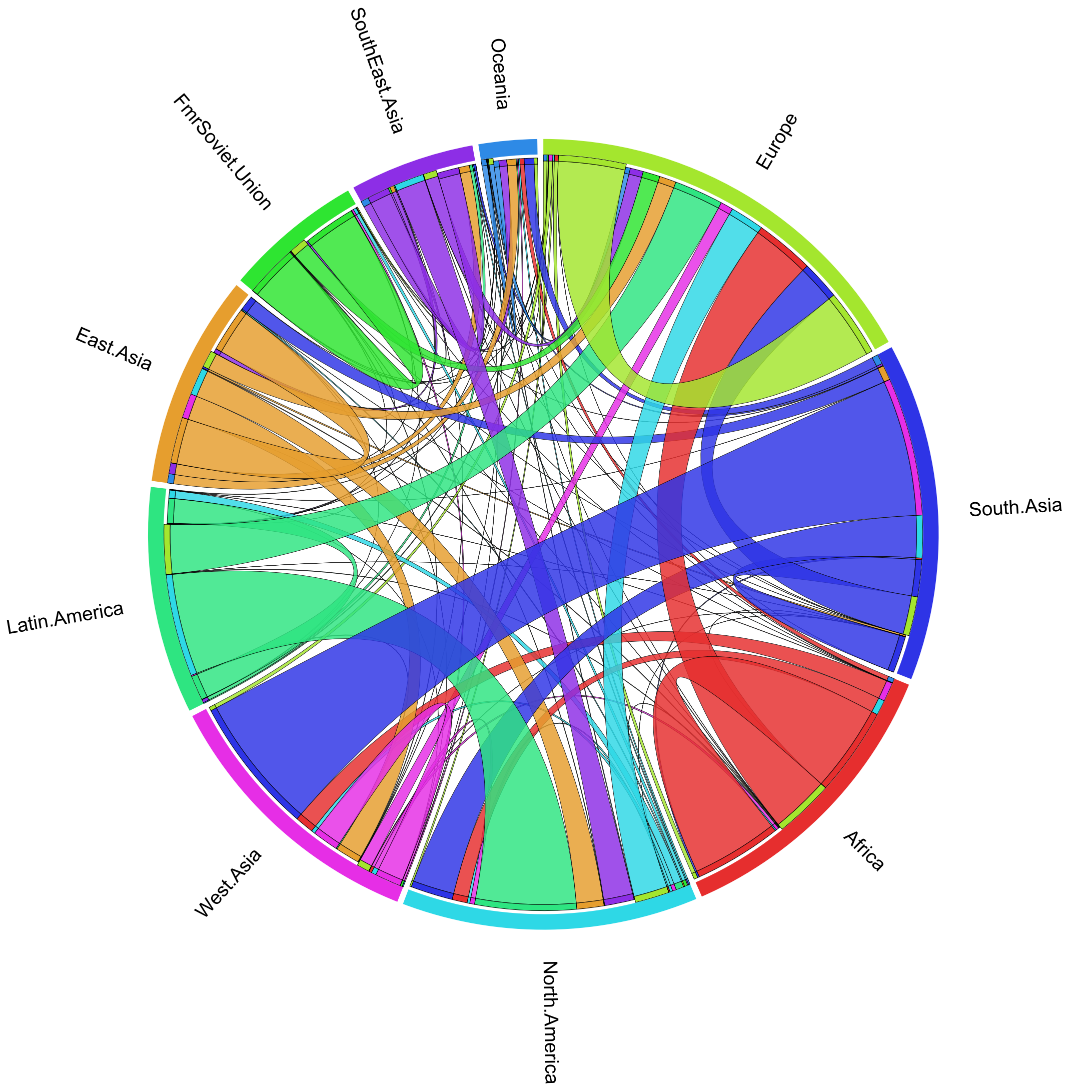

Figure 4: Circular migration plot of migration flows between world regions in 2005-10, created using Circos with default settings.

4.2.3 More on data preparation for Circos

When using your own data later on, it is recommended to use our data files as a template. This is because Circos has specific formatting requirements, including:

- The input data file must be saved as a tab-delimited text file.
- The text file with the input data must be saved in *.../my_first_plot/*.
- The input data file must contain the migration flows in matrix format, with rows corresponding to origins and columns to destinations.
- Each input data file must contain only one matrix.

- All zeros or missing values in the flow table have to be replaced with ones, because Circos cannot handle missing or zero values.
- The row and column headers (i.e. the region or country names) must be identical and must have no spaces (e.g. change North America to North.America).
- IMPORTANT: Before running Circos via Cygwin, make sure the text file does not contain any quotation marks.

4.2.4 Design specifications

When using your own data, you may want to change the design of the plot by modifying the specifications included in the configuration files saved in *.../etc/*. By default, this folder contains six files: *circos.conf* is the main configuration file; *colors.conf* defines colours in RGB format; *ideogram.conf* defines the basic layout of the plot; *make-table.conf* creates a random dataset; *parse-table.conf* defines how the input data file is parsed into an intermediate state; *ticks.conf* defines the ticks and labels positioned on the outside of the circle's segments.

The commands for Cygwin given in Section 4.2.2 call only two configuration files directly: *parse-table.conf* and *circos.conf*. The latter includes the colors, ideogram, and ticks configuration files via an `«include»` function. The file *make-table.conf* is not used at all since we use our own input data. The colors, ideogram and ticks configurations can also be included directly into *circos.conf*.

4.3 Creating plots with custom settings

The effects that the nuanced judgements discussed in Section 3 have on the visual appearance of the circular migration plot are highlighted by comparing Figures 4 and 5, which show the same dataset of migration between the worlds regions in 2005-10 (source: Abel & Sander (2014)), but use different configurations. The plot in Figure 4 uses default settings, whereas Figure 5 was created using our custom settings. This section also provides an example of a country-level plot (Figure 6.

4.3.1 Creating the plots

We provide two sets of custom configuration files and corresponding input data in the supplementary material:

- region_custom: a circular plot of migration between the world's regions using our custom settings (see Figure 5)
- country_custom: a circular plot of migration between the world's key sending and receiving countries using our custom settings (see Figure 6)

To create the circular migration plot shown in Figure 5, make a copy of the entire *tableviewer* folder and re-name it to *my_second_plot*. Save the data input file *region_custom.txt* provided in the supplementary material in *.../my_second_plot/*. Parse the data and draw the plot following the same steps as for the first plot.

- In Cygwin, change the directory to *.../my_second_plot/*:
  ```
  cd /cygdrive/c/circos/tools/my_second_plot/
  ```
- Enable edits to all files in this directory:
  ```
  chmod u+x bin/*
  ```
- Parse the input data file *region_custom.txt*:
  ```
  cat region_custom.txt | bin/parse-table | bin/make-conf -dir data
  ```
- Draw the plot:
  ```
  ../../bin/circos -conf etc/circos.conf
  ```
- The plot is saved in *../my_second_plot/img/*.

To create the circular migration plot shown in Figure 6, make another copy of the *tableviewer* folder and re-name it to *my_third_plot*. Save the data input file *country_custom.txt* provided in the supplementary material in *.../my_third_plot/*. Parse the data and draw the plot:

- In Cygwin, change the directory to *.../my_third_plot/*:
  ```
  cd /cygdrive/c/circos/tools/my_third_plot/
  ```
- Enable edits to all files in this directory:
  ```
  chmod u+x bin/*
  ```
- Parse the input data file *country_custom.txt*:
  ```
  cat country_custom.txt | bin/parse-table | bin/make-conf -dir data
  ```
- Draw the plot:
  ```
  ../../bin/circos -conf etc/circos.conf
  ```
- The plot is saved in *../my_third_plot/img/*.

4.3.2 Summary of changes made to configuration files

Several changes to the input data and configuration files were necessary before drawing the circular migration plot of flows within and between world regions shown in Figure 5. (A) The world regions were arranged based on approximate geographic proximity and a column in the data input file was included to specify the order of regions. (B) Each world region was assigned a unique RGB colour, using different shadings of the same colour for regions on the same continent (see Table 1). (C) The settings in *parse-table.conf* were modified to indicate that the input data file has a row that specifies the order of columns, a column that specifies the order of rows, and a column that specified the colour of each row. (D) Segments (i.e. regions) were ordered as specified in the extra column of the input data file, where column segments are placed before row segments and that ribbon bundles for each segment are sorted in descending order (i.e. immigration flows are plotted before emigration flows; each set of flows is sorted based on flow size). (E) Large ribbons (i.e. flows) were set to be drawn first and small ribbons to be drawn on top. This ensures that small flows remain visible. (F) Because the custom RGB colours specified in the input data file are used, the colour_remap function in *parse-table.conf* was set to "no". (G) Only flows containing >140,000 migrants are shown. (H) In *circos.conf*, the spacing between segments was enlarged in the ideogram settings; ticks and tick labels were turned on; stacked bars that are positioned outside the circle and show total inflows and outflows were set to be shown; the position and size of the world region labels was altered; and the gap between flows and region segments was changed.

Table 1: Region names and RGB colours used in regional-level circular migration plot.

Region	RGB	Colour
North America	255,0,0	🟥
Africa	210,150,12	🟧
Europe	125,175,0	🟩
Fmr Soviet Union	117,0,255	🟪
West Asia	160,0,125	🟪
South Asia	29,100,255	🟦
East Asia	0,255,233	🟦
South East Asia	73,255,100	🟩
Oceania	100,146,125	🟩
Latin America	255,219,0	🟨

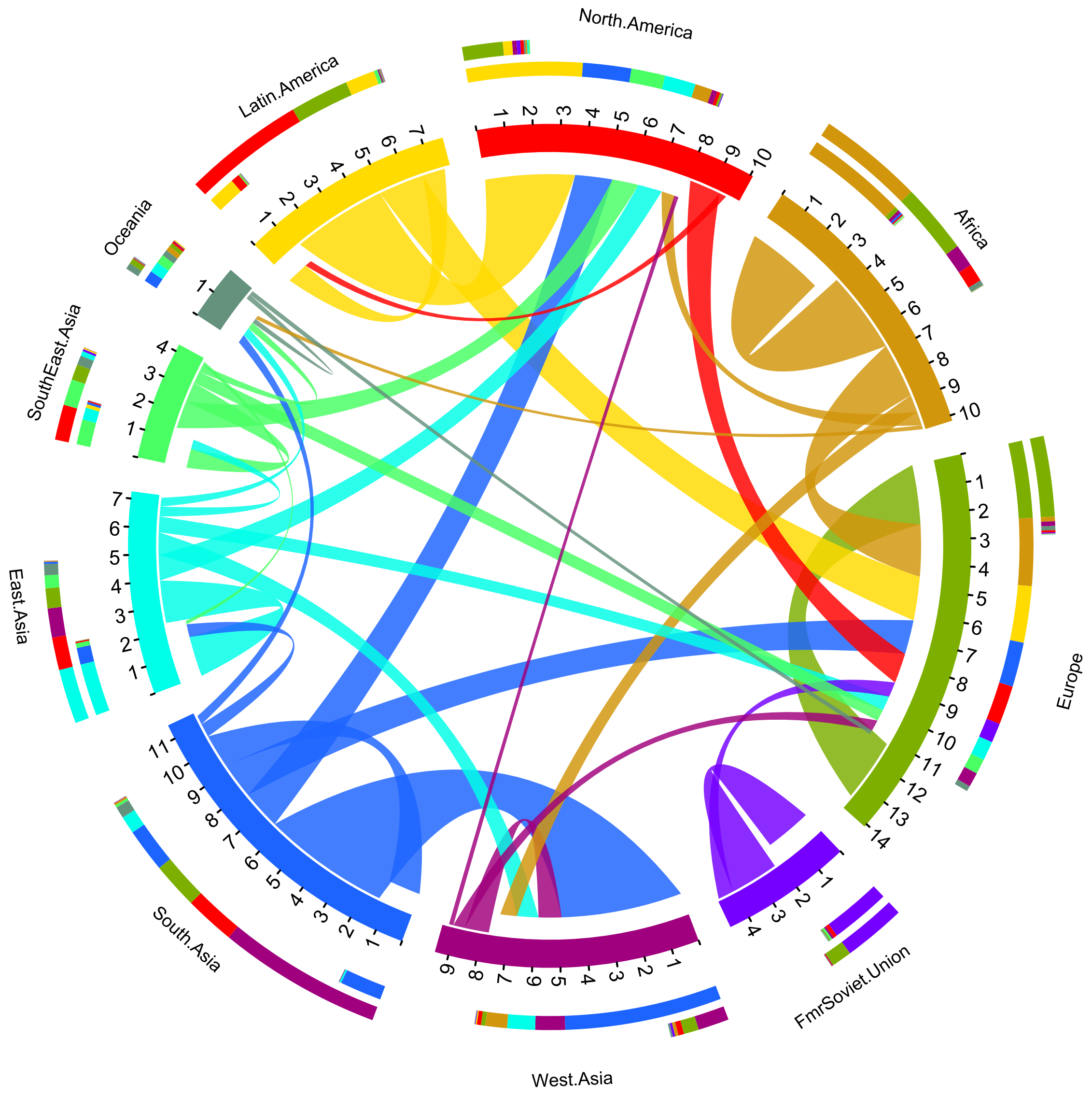

Figure 5: Circular migration plot of migration flows between world regions in 2005-10, created using Circos with custom settings. Ticks indicate the volume of migration in millions.

To draw the circular migration plot of flows between the 25 key sending and receiving countries shown in Figure 6, we made two additional key changes to the configuration files. First, we specified in *parse-table.conf* that only flows containing at least 700,000 migrants are shown (`<percentile 20>`). Second, we altered the tick settings and region/country label settings in *circos.conf*, and turned off the outer stacked bars.

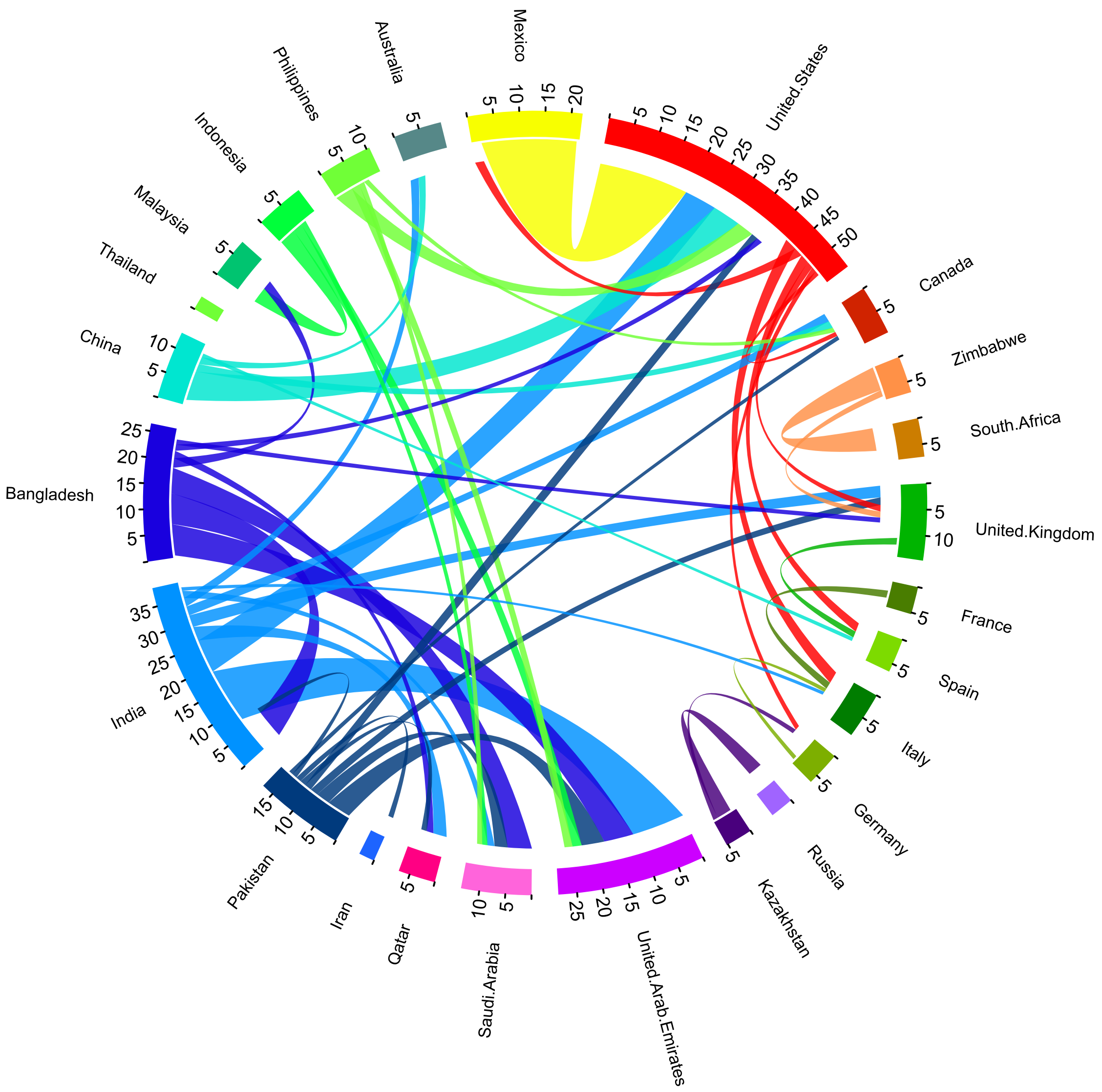

Figure 6: Circular migration plot of migration flows between world's top 25 sending and receiving countries in 2005-10, created using Circos with custom settings. Ticks indicate the volume of migration in 100,000s.

5 Creating circular plots using R

R (RDC-Team, 2011) is a free software environment for statistical computing and graphics. Since the release of the Circos software, two contributed R packages, RCiricos

(Zhang *et al.* , 2013) and circlize (Gu, 2013), have become available on the open source CRAN repository. Both allow circular migration plots, similar to those in Figures 5 and 6 to be produced with R. In this section, we focus on illustrating the reproduction of these figures using the circlize package, which we found marginally easier to work with as many of the function names and arguments are more general terms related to plotting in comparison to some of the specialist Genomics terms used in the RCircos package. In the remainder of this section, R scripts[8] used to replicate the circular migration plots of Figures 5 (without the outer stacked bars) and 6 are discussed, with a focus on the plotting functions within the circlize package.

5.1 Data preparation

In order to draw a circular migration plot using the circlize package, at least two R objects must be created. First, a `matrix` containing the flow data, which we assign to object `m` is set up directly from the input data used in Section 4.3 (region_custom.txt and country_custom.txt are provided in the supplementary material). Second a `data.frame` object, which we call `df1` is set up to store information on each segment of the circle to be plotted. For the regional data `df1` is shown below.

```
> df1
   order         rgb          region xmin      xmax   r   g   b    rcol      lcol
1      1     255,0,0   North America    0 10.095271 255   0   0 #FF0000 #FF0000C8
2      2  210,150,12          Africa    0 10.199718 210 150  12 #D2960C #D2960CC8
3      3   125,175,0          Europe    0 14.107659 125 175   0 #7DAF00 #7DAF00C8
4      4   117,0,255    Soviet Union    0  4.736899 117   0 255 #7500FF #7500FFC8
5      5   160,0,125       West Asia    0  9.492199 160   0 125 #A0007D #A0007DC8
6      6  29,100,255      South Asia    0 11.525938  29 100 255 #1D64FF #1D64FFC8
7      7   0,255,233       East Asia    0  7.256758   0 255 233 #00FFE9 #00FFE9C8
8      8  73,255,100 South East Asia    0  4.259029  73 255 100 #49FF64 #49FF64C8
9      9 100,146,125         Oceania    0  1.990567 100 146 125 #64927D #64927DC8
10    10   255,219,0   Latin America    0  7.737376 255 219   0 #FFDB00 #FFDB00C8
```

The first three rows of `df1` are provided in the input file, where the data frame is sorted using the `arrange` function in the plyr package (Wickham, 2011) based on the

[8]The R scripts are available in both the supplementary material and as demonstration scripts in the migest package Abel (2012). The later can be run in R using the commands `demo("cfplot_reg", package = "migest", ask=FALSE)` or `demo("cfplot_nat", package = "migest", ask=FALSE)` once the migest, circlize and plyr packages are installed.

order column. The xmin and xmax columns are created in order for plotting functions to determine the length of segments on the outside of the plot. The xmax values are determined by the sum of the inflows and outflows (column and row totals) of the flow table. The separate numeric r, g and b columns are used to find the colour names in the preceding two columns using the `rgb` function. The first column of colours, rcol is used for the circle's segments. The second column of colours, lcol, lists transparent versions of rcol, used in the links (i.e. flows) between the circle's segments.

5.2 Plotting circle segments

Using the circlize software, basic graphic parameters for circos plots can be set using the `circos.par` function. For our plots we found editing the `track.margin` argument was useful in controlling the amount of space for segment labels and the `gap.degree` argument for spacing between segments. We also set the `start.degree=90` to allow the first region (North America) in the circos plot to be located around the 12 o'clock position.

Before plotting the segments, users must define their names and lengths using the `circos.initialize` function. The `circos.trackPlotRegion` function can then be called to plot each segment (called sector in circlize), as such:

```
circos.trackPlotRegion(ylim = c(0, 1), factors = df1$region, track.height=0.1,
  panel.fun = function(x, y) {
    name = get.cell.meta.data("sector.index")
    i = get.cell.meta.data("sector.numeric.index")
    xlim = get.cell.meta.data("xlim")
    ylim = get.cell.meta.data("ylim")
    circos.text(x=mean(xlim), y=2.2, labels=name, direction = "arc")
    circos.rect(xleft=xlim[1], ybottom=ylim[1], xright=xlim[2], ytop=ylim[2],
                col = df1$rcol[i], border=df1$rcol[i])
    circos.rect(xleft=xlim[1], ybottom=ylim[1], xright=xlim[2]-rowSums(m)[i],
                ytop=ylim[1]+0.3, col = "white", border = "white")
    circos.rect(xleft=xlim[1], ybottom=0.3, xright=xlim[2], ytop=0.32,
                 col = "white", border = "white")
    circos.axis(labels.cex=0.8, direction = "outside",
                 major.at=seq(0,floor(df1$xmax)[i]), minor.ticks=1,
                 labels.away.percentage = 0.15)
  }
)
```

In the first three arguments of this function call, the limits on the y-axis, names and the height relative to the overall plot of the segments are given. Specifications for individual segments are determined using the `panel.fun` argument. For each segment, the name, index, x and y limits are recorded in-turn from meta data within the user's R environment. In order to display migration data, we suggest the use of five functions to build the segments similar to that of Figures 5 and 6. First, `circos.text` adds labels for given x and y co-ordinates, relative to each segment, with specifications for the label name and direction of the text. Second, `circos.rect` is used to plot the main segment given left, right (on the x-axis) top and bottom (on the y-axis) coordinates, alongside colour details taken from `df1`. Third, a white rectangle is plotted (using `circos.rect`) over part of the main segment to distinguish inflows from outflows. Its length is determined by the row sums of the migration flow matrix `m`. Its height is set as 30% of the segment height (which is 1 unit tall). Fourth, a final thin white rectangle is plotted over the entire length of each segment. Fifth, the axis are plotted using the `circos.axis` command.

5.3 Plotting links

In order to plot links between segments in order of the thickest first, we created a second `data.frame` object, `df2` containing the long form of the matrix `m`. This object is first sorted to place the thickest flow in the first row and smallest in the last row, and then sub-setted to eliminate small flows that would clutter the plot. The first six rows can be viewed in the R console:

```
> head(df2)
           orig          dest        m
1    South Asia    West Asia 4.902081
2 Latin America North America 3.627847
3        Africa        Africa 3.142471
4        Europe        Europe 2.401476
5        Africa        Europe 2.107883
6  Soviet Union  Soviet Union 1.870501
```

We also add two further columns to the `df1` object, marking the initial x-positions of each segment for links representing outflows (`sum1`) and inflows (`sum2`).

Using a `for` loop, we consider each row of `df2` in-turn. Within the loop, we first assign the `i` and `j` index for each origin and destination under consideration.

```
for(k in 1:nrow(df2)){
  i<-match(df2$orig[k],df1$region)
  j<-match(df2$dest[k],df1$region)
  circos.link(sector.index1=df1$region[i],
              point1=c(df1$sum1[i], df1$sum1[i] + abs(m[i, j])),
              sector.index2=df1$region[j],
              point2=c(df1$sum2[j], df1$sum2[j] + abs(m[i, j])),
              col = df1$lcol[i], top.ratio=0.66, top.ratio.low=0.67)
  df1$sum1[i] = df1$sum1[i] + abs(m[i, j])
  df1$sum2[j] = df1$sum2[j] + abs(m[i, j])
}
```

The links are then plotted using the `circos.link` command, connecting the origin and destination segments. The size of the base of the link is determined by the `point1` argument at the origin segment and `point2` argument at the destination segment. We set the origin segment to start at the current sum of outflows (`df1$sum1[i]`) and end at the sum of outflows with the addition of the `m[i,j]` flow under consideration in the loop. A similar specification is also set for the x-axis of the destination segment using the inflow totals (`df1$sum2[j]`). The height and thickness of the link at its mid-point is determined by the `top.ratio` and the `top.ratio.low` argument. We found setting these values to 0.66 and 0.67 (on a scale of 0 to 1) gave links similar in style to those created using Circos (see Section 4). Before ending the loop we update the `sum1` and `sum2` columns in `df1` in order for the next plotted link in the corresponding region to begin further along the x-axis of the origin and destination segments.

Plots in R graphics devices can sometimes appear skewed if the device is not perfectly square. For easier viewing, readers can output the contents of their graphics device to a PDF using:

```
dev.copy2pdf(file = "cfplot_reg.pdf", height=10, width=10)
```

The final plots of both the region- and country-level data are shown in Figure 7. A couple of adjustments were made in the code for the country-level plot. First the `track.margin` argument was increased to allow more space for country labels. Second, adjustments for the position and direction of the labels, dependent on the position of the segment in the circle were passed to the `circos.text` function.

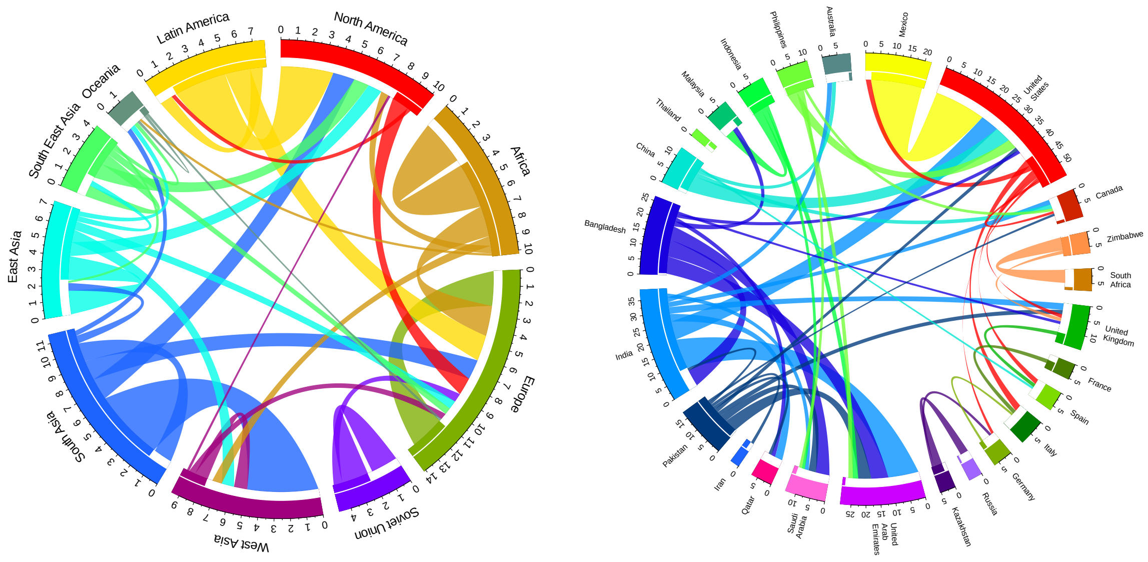

Figure 7: Circular migration plot of migration flows between regions (left) and world's top 25 sending and receiving countries (right) in 2005-10, created using the circlize R package.

6 Creating interactive circular plots using d3.js

D3 is a JavaScript library which empowers creating beautiful interactive visualisations in HTML. Although not tied to the Web per se, it is predominantly used to do data-driven manipulations of Web content. D3 is the fourth iteration of a visualization library, its precursors are Prefuse (Java, 2005), Flare (Actionscript, 2007) and Protoviz (Javascript, 2009). D3 was developed by Mike Bostock[9] and sponsored by his employer, The New York Times. Since its release in 2010, this JavaScript library has become the state-of-the-art for interactive data visualisations and is frequently used in emerging field of data journalism.

Using D3 to embedd Scalable Vector Graphics (SVGs) inside HTML documents makes it possible to dynamically create various shapes like circles, bézier curves and rectangles. We bind the migration flow estimates by (Abel & Sander, 2014) to SVGs to create a circular migration plot that can be shown on the web. Each SVG element is assigned a special set of Cascading Style Sheets (CSS) properties that define colors, and thus ensure that the plot resembles closely the static image created using Circos (see Section 4).

The main strenght of D3 is that SVG shapes can be manipulated based on the data

[9]https://github.com/mbostock/d3/commit/b2aa3c7355f46634738b46789c649c4e2da94e3e

they are bound to. This "data-driven"[10] approach to Document Object Model (DOM; the programming interface for HTML documents) manipulation facilitates powerful visualization that is not tied to a proprietary framework. With the help of D3, we generate a circular plot with smooth transitions (a) between time periods, and (b) between region- and country-level views. User interaction is further enhanced through tooltips that show the numbers of migrants for a given flow, country or region when hovering over the plot with the mouse.

The functional style of D3 encouraged us to develop a library tailored at creating circular migration plots for the web. This library is available online on GitHub[11] and makes reusing our code for other visualisations of flow data extremely easy. In the remainder of this section, we discuss the main steps for creating interactive circular migration plots using our customised D3 library.

6.1 Data preparation

To minimize computation needs on the client, raw data is preprocessed and transformed to a structure best suited for the JavaScript language which in turn generates the visible circular plot. While the CSV format is a good fit for tabular data interchange, JSON (JavaScript Object Notation)[12] has the ability to hold complex structured data and is natively parseable in Javascript programs. The output of the transform step is therefore JSON-formatted.

Preprocessing involves two steps:

1. Filter out countries with too small migration flows

2. Transform into a D3-optimized JSON structure

The first step is to remove countries which have very small migration flows. Otherwise the graphic becomes too cluttered and unresponsive. Too many fine shapes also make it difficult to read the plot and distract from the key flows. The countries to be removed are defined in a seperate CSV file.

[10]http://d3js.org/
[11]https://github.com/null2/circular-migration-plot
[12]http://json.org/

The interchange CSV looks like this:

```
originregion_name,destinationregion_name,origin_iso,origin_name,destination_iso, \
  destination_name,countryflow_1990,countryflow_1995,countryflow_2000,countryflow_2005
North America,North America,CAN,Canada,CAN,Canada,0,0,0,0
North America,North America,CAN,Canada,USA,United States,1509,190436,238,28
North America,North America,USA,United States,CAN,Canada,56108,635,84430,96074
North America,North America,USA,United States,USA,United States,0,0,0,0
```

The CSV file defining visible countries is formatted like this:

```
iso,show
USA,1
FIN,0
```

In this example, `USA` will be shown, whereas `FIN` will be hidden.

The result of the first step is exactly like the input CSV, except that the rows where the value of the origin_iso or destination_iso columns have a 0 in the countries filter CSV are removed.

The output of step one is then used as input of the second step, the compile step. This one creates a data structure which can be consumed by the JavaScript running on the client in a fast and efficient manner.

The resulting final JSON looks like this:

```
{
  "regions": [0, 3, 36, 61, 74, 88, 96, 101, 110, 113],
  "names": [
    "North America",
    "Canada",
    "United States",
    "Africa",
    "Angola",
    ...
    "Venezuela"
  ],
  "matrix": {
    "2005": [
      [ 139950, ... 8621 ],
      [ 51564, ... 458 ],
      ...
    ],
    "1990": [
      ...
    ]
  }
}
```

To reduce the amount of chords displayed at any time, data is accumulated in region flows. Only Region flows are initially displayed in the plot. The user can expand a region to see individual country flows by clicking on the region.

There are only two regions expanded to individual countries at any time, again for perfomance and focus reasons. When the user expands a third region, the first region collapses.

To achieve this, the region flows are stored in the flow `matrix` in the data structure, followed by the appropriate country flows. A `regions` index keeps track of the region flows. Expanding a region is then done by displaying all flows in the matrix between the current region index and the next region index. To display labels, region and country `names` are included. More on matrices and the data format can be found here[13].

An implementation of these tasks as well as a description and usage instructions can be found in the Circular Migration Plot Library[14].

6.2 Extending D3

While D3 provides helpful layouts[15] for generating chords, the original implementation shown in Figure 8 had to be extended to fit the requirements of circular migration plots.

In contrast to the chord diagram provided by Mike Bostock, we visualise migration flows as two directed chords, one for each directional flow between two countries/regions (see Figure 9). A chord is a shape which displays a single flow. It is a geometric shape with two arcs connected with two bezier curves. The other difference to the original example is that the chord ends on a slightly smaller radius of the main circle. The gap between bezier and circle segment indicates the destination end of a migration flow.

[13]https://github.com/mbostock/d3/wiki/Chord-Layout
[14]https://github.com/null2/circular-migration-plot
[15]https://github.com/mbostock/d3/wiki/Layouts

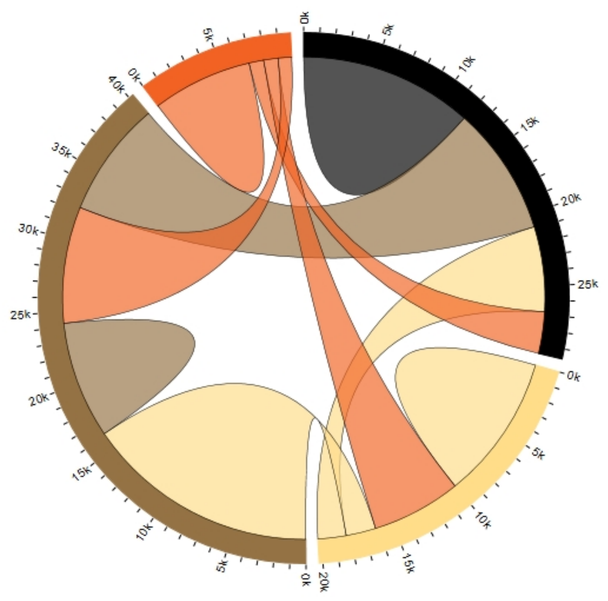

Figure 8: Chord diagram by Mike Bostock showing directed relationships among a group of entities (Source: http://bl.ocks.org/mbostock/4062006).

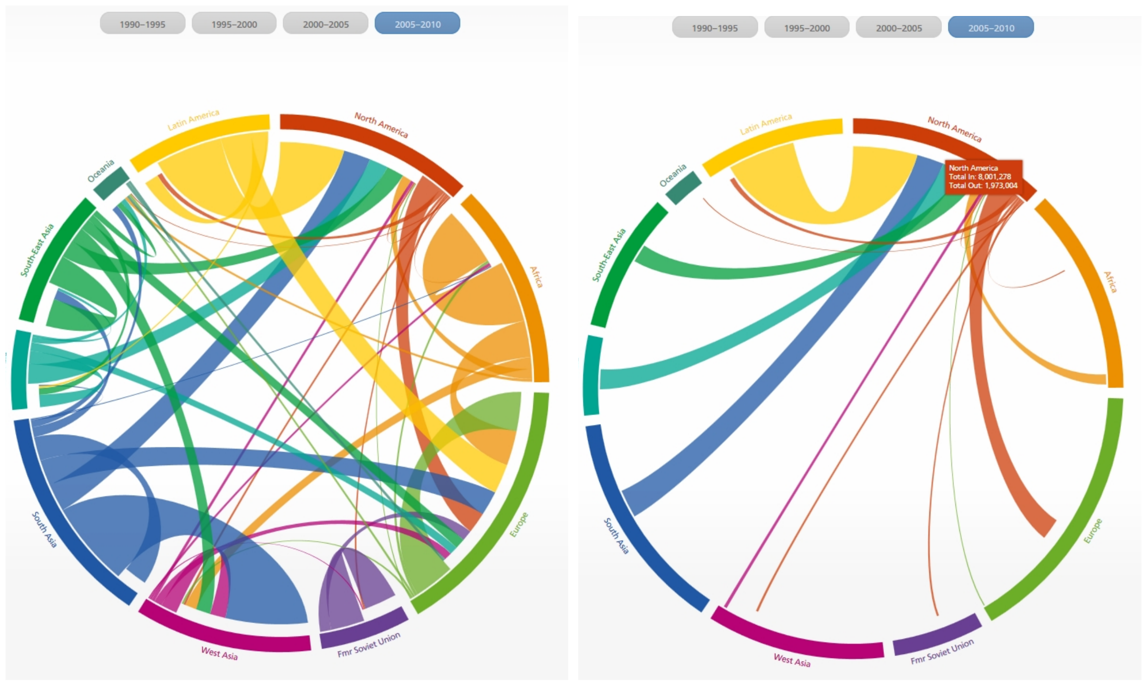

Figure 9: Interactive circular migration plot of migration flows between regions (left) and flows to and from North America (right) in 2005-10, created using D3.

The modified chord layout[16] can be found along with the extended chord shape[17] in the `lib/` folder of the Circular Migration Plot Library[18]. Figure 10 illustrates the interactive features of our visualisation, which allows the user to switch between region and country-level views by clicking on the segment of the circle. For clarity and appearance, a maximum of two regions can be split into countries.

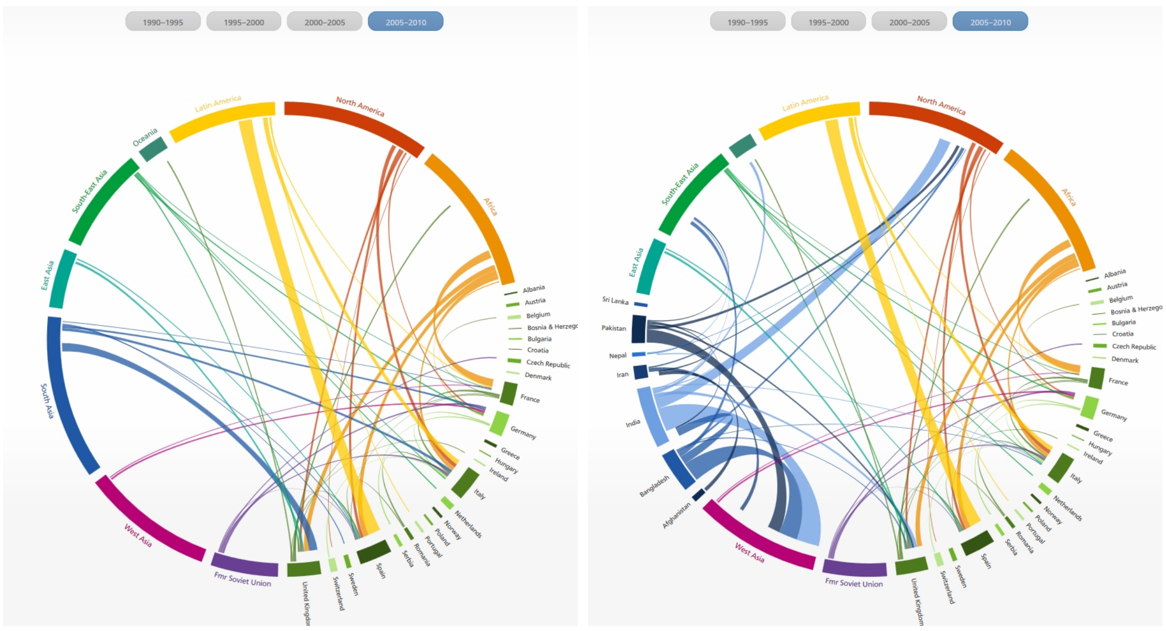

Figure 10: Interactive circular migration plot of migration flows to and from European countries (left) and flows to and from South Asian and European countries (right) in 2005-10, created using D3.

7 Conclusions

Making the wealth of information contained in migration flow data more accessible and understandable for researchers and the interested public is difficult because of the high degree of complexity that the data exhibit. Migration maps that are typically used to show the volume and direction of place-to-place flows struggle to visually convey the complexities of human movement.

The visualisation presented in this paper aimed to address this gap in the literature. We have shown in an earlier paper (Abel & Sander, 2014) that circular migration plots allow us to meaningfully explore, relate, and communicate migration data to a wider audience. In a nutshell, circular plots arrange the origins and destinations of migration

[16]https://github.com/null2/circular-migration-plot/blob/master/lib/layout.js
[17]https://github.com/null2/circular-migration-plot/blob/master/lib/chord.js
[18]https://github.com/null2/circular-migration-plot

flows in a circular layout, scale each flow to allow the entire system to be shown simultaneously, relate the volume of movement to the width of the flow, and indicate its direction through colour coding.

To encourage the wider use of circular migration plots, we have discussed the relevant steps for creating the plot using three alternative software packages: Circos, R via the circlize package, and the Java Script Library D3. The decision about which software to choose depends on the user's preferences. Circos offers the greatest flexibility in design and functionality, while R via the circlize package may be more appealing for those who are already familiar with data visualisations in R. The JavaScript library D3 is becoming an increasingly popular tool for interactively visualising data on the web. The script code can be easily embedded into any HTML5 website. The input data for the circular plots presented in this paper are available in the supplementary material, which also provides the configuration files required for creating plots with Circos. Additional information on quantifying and visualizing the global flow of people can be found at www.global-migration.info, where we also provide a link to the D3 source code on Github.

We hope that our new visualisation opens up new avenues for understanding global migration, for telling the stories hidden behind large data sets, and for more effectively communicating science to peers and the interested public. Circular migration plots not only enhance our ability to analyse, understand and communicate the complex patterns and trends in migration flow data, they can also be used to visualise other kinds of flow data, such as trade and remittances flows.

References

Abel, Guy J. 2012. *migest: Useful R Code for the Estimation of Migration.*

Abel, Guy J. 2013. Estimating global migration flow tables using place of birth data. *Demographic Research*, **28**, 505–546.

Abel, Guy J., & Sander, Nikola. 2014. Quantifying Global International Migration Flows. *Science*, **343**(6178).

Beddoe, David Paul. 1978. *An alternative cartographic method to portray Origin-Destination data.* Ph.D. thesis, University of Washington.

Bostock, M., Ogievetsky, V., & Heer, J. 2011. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, **17**(12), 2301–2309.

Bryant, John. 2011. Visualising Internal Migration Flows. *Population Association of New Zealand*, **37**, 159–171.

Cerrutti, Marcela, & Massey, Douglas S. 2001. On the auspices of female migration from Mexico to the United States. *Demography*, **38**(2), 187–200.

Czaika, Mathias, & de Haas, Hein. 2013. The Globalisation of Migration. *IMI Working Papers.*

Dorling, Danny. 2012. *The Visualisation of Spatial Social Structure.* John Wiley & Sons.

Fassmann, Heinz, & İçduygu, Ahmet. 2013. Turks in Europe: Migration Flows, Migrant Stocks and Demographic Structure. *European Review*, **21**(03), 349–361.

Geertman, Stan, Jong, Tom de, & Wessels, Coen. 2003. Flowmap: A Support Tool for Strategic Network Analysis. *Pages 155–175 of:* Geertman, Dr Stan, & Stillwell, Professor John (eds), *Planning Support Systems in Practice.* Advances in Spatial Science. Springer Berlin Heidelberg.

Gu, Zuguang. 2013. *circlize: Circular layout in R.*

Guo, Diansheng, Zhu, Xi, Jin, Hai, Gao, Peng, & Andris, Clio. 2012. Discovering Spatial Patterns in Origin-Destination Mobility Data. *Transactions in GIS*, **16**(3), 411–429.

Heer, Jeffrey, Bostock, Michael, & Ogievetsky, Vadim. 2010. A tour through the visualization zoo. *Commun. ACM*, **53**(6), 59–67.

Krzywinski, Martin, Schein, Jacqueline, Birol, İnanç, Connors, Joseph, Gascoyne, Randy, Horsman, Doug, Jones, Steven J., & Marra, Marco A. 2009. Circos: An information aesthetic for comparative genomics. *Genome Research*, **19**(9), 1639–1645.

Law, C.M., & Warnes, A.M. 1976. The Changing Geography of the Elderly in England and Wales. *Transactions of the Institute of British Geographers*, **1**(4), 453–471.

Özden, Caglar, Parsons, Christopher R., Schiff, Maurice, & Walmsley, Terrie L. 2011. Where on Earth is Everybody? The Evolution of Global Bilateral Migration 1960–2000. *The World Bank Economic Review*, **25**(1), 12–56.

Rae, Alasdair. 2009. From spatial interaction data to spatial interaction information? Geovisualisation and spatial structures of migration from the 2001 UK census. *Computers, Environment and Urban Systems*, **33**(3), 161–178.

Ravenstein, E. G. 1885. The Laws of Migration. *Journal of the Statistical Society of London*, **48**(2), 167–235.

RDC-Team. 2011. R: a language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2012. *Open access available at: http://cran. r-project. org.*

Schenk, Tom. 2012. *Circos Data Visualization How-to*. Packt Publishing Ltd.

Tobler, Waldo R. 1987. Experiments In Migration Mapping By Computer. *The American Cartographer*, **14**(2), 155–163.

Tufte, Edward R., & Graves-Morris, P. R. 1983. *The visual display of quantitative information*. Vol. 2. Graphics press Cheshire, CT.

Verbeek, Kevin, Buchin, Kevin, & Speckmann, Bettina. 2011. Flow map layout via spiral trees. *IEEE transactions on visualization and computer graphics*, **17**(12), 2536–2544. PMID: 22034375.

Wickham, Hadley. 2011. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, **40**(1), 1–29.

Zhang, Hongen, Meltzer, Paul, & Davis, Sean. 2013. RCircos: an R package for Circos 2D track plots. *BMC bioinformatics*, **14**(1), 1–5.