Tools for Estimating, Measuring and Working with Migration Data in R

Guy J. Abel, James Raymer, Ellen Kraly

2021-10-18

Contents

1 . **7**

2 Outline **9**

2.1 Workshop on Migration Estimation 9

2.2 General Points 9

3 Migration Concepts **11**

3.1 Spatial 11

3.2 Temporal 17

3.3 Measures 20

4 Handling Migration Data in R **25**

4.1 Contingency Table 25

4.2 Data Manipulation 27

4.3 Matrix Operations 31

4.4 Summaries 35

5 Summary Migration Indices **43**

5.1 Background 43

5.2 Intensity 44

5.3 Distance 50

5.4 Connectivity 57

5.5 Impact 60

6 Estimating Net Migration **65**

6.1 Net Migration 65

6.2 Vital Statstics 66

6.3 Survival Methods 69

6.4 Birthplace 76

7 Describing and Estimating Migration Age Structures **85**

7.1 Rogers Castro 85

7.2 Model schedules 89

7.3 Fitting schedules 95

7.4 Age Indices 101

7.5 Smoothing 104

7.6 Graduating 110

8 Describing Bilteral Migration Data **117**

8.1 Components 117

8.2 Log-Linear Models 128

8.3 Dimensions 130

9 Estimating Bilateral Migration **141**

9.1 IPFP 141

9.2 Net Constraints 156

10 Chord Diagrams for Visualising Bilateral Migration **161**

10.1 161

10.2 Chord Diagram 165

10.3 circlize 168

10.4 Sectors 174

10.5 Colour 178

10.6 Chords 183

10.7 Direction 185

10.8 Labels 190

11 Sankey Plots for Visualising Bilateral Migration 199

11.1 Background 199

11.2 Data Format 203

11.3 Parrellel Sets 205

11.4 Set Axes 207

11.5 Colour 209

11.6 Labels 215

11.7 Spacing 219

Chapter 1

•

Chapter 2

Outline

2.1 Workshop on Migration Estimation

- Course outline
  - Part 1: Migration data and concepts
  - Part 2: Handling migration data in R
  - Part 3: Summary migration indices
  - Part 4: Estimating net migration
  - Part 5: Describing and estimating migration age structure
  - Part 6: Describing bilateral migration data
  - Part 7: Estimating bilateral migration
  - Part 8: Chord diagrams for visualising bilateral migration
  - Part 9: Sankey plots for visualising bilateral migration
- Folder with slides, code in slides, exercises and exercise solutions on dropbox
  - `http://bit.ly/kostat2021mig`
- The `slides-code` folder contains the R functions in the PDF's of each slide
  - Ignore the first few lines (with `knitr` functions) - they are by-product of using Rmarkdown to create the slide PDFs

2.2 General Points

1. Please be patient. Teaching classes that involve R is never smooth.
   - Everyone has different computers, R versions, package versions

- Remote learning and getting used to Zoom

2. I am assuming you have some knowledge on using R, especially the *tidyverse* set of packages. If you do not, then try working your way through an online course to get up to speed:
   - `https://r-bootcamp.netlify.app/` (free)
3. Throughout the course we will work on some exercises. You might get stuck, especially if you are new to R. Be patient and remember that the frustration is a normal part of learning a programming language.
4. The exercise solutions are provided. If you do not have time to complete the exercises in class, spend some extra time to keep working on the exercises.
   - Some days we will not have time to finish all the exercises
5. If you having trouble installing R, RStudio or the packages I have set up an RStudio cloud project where everything is installed for you:
   - `https://rstudio.cloud/project/1593361`
   - Need to create a RStudio account
   - Save changes using button on top right

Chapter 3

Migration Concepts

- Migration involves a change in place of abode, or place of "usual" residence-a taking-up of life in a new or different place - United Nations Department of Economic and Social Affairs Population Division [1983]
- Could apply to a range of demographic units, such as a person, a family or a household
- Tend to excludes nomads, movement of population groups with no fixed place of residence, seasonal movements of persons who live in two or more places during the course of a year.
- Both a spatial (place) and temporal (change) dimension are required in the definition of migration

3.1 Spatial

3.1.1 Place of Residence

The *Principles and Recommendations for Population and Housing Censuses* (UN Statistics Division 2008: 102, para. 1.463) defines usual residence as follows:

"It is recommended that countries apply a threshold of 12 months when considering place of usual residence according to one of the following two criteria:

1. The place at which the person has lived continuously for most of the last 12 months (that is, for at least six months and one day), not including temporary absences for holidays or work assignments, or intends to live for at least six months

2. The place at which the person has lived continuously for at least the last 12 months, not including temporary absences for holidays or work assignments, or intends to live for at least 12 months."

3.1.2 Place of Residence

- Typically no restriction is placed on the distance involved in a relocation - see Lee [1966].
  - Could include move from one apartment to another in the same building, or it may be a move to another country.
- In the past, some researchers have drawn a distinction between moves between local communities (cities, labour markets) and moves within local communities
  - Labelled as 'migration' and 'local mobility'.
  - Many have argued that this distinction is problematic and no spatial constraints on the definition of migration should be used.

3.1.3 Place of Residence

- If address information on points of origin and points of destination, the tabulation of moves by distance covered could be obtained.
  - Usually, not possible in countries without population registers
  - Time consuming and of little policy relevance
- Census or survey results are necessarily tabulated for the administrative or political units into which the country is divided.
- A migration is then operationally defined as a change of residence from one civil division to another, and the volume of migration is to a considerable degree a function of the size of areas chosen for compilation.

3.1.4 Courgeau (1979)

Graph 2. — Annual migration rates of the different geographical subdivisions.

3.1.5 Place of Origin

- Typologies of migration data can be stratified by differing places of origin. Most commonly used:

Origin	Data Type
Previous place of residence	Migration event (movement) data
Place of residence n years ago	Migrant transition data

- Lifetime migration data could be considered as a form of transition data where n changes based on the age of each individual
  - Migrant stock data are an aggregation over all persons lifetime migration flow.
  - Given at specific point of time without an interval.
  - Migration data literature often distinguishes between stock and flow data.
- Other types of migration data occasionally collected are
  - Duration at residence (combined with place of previous residence can derive a migration transitions)
  - Number of moves over a given interval (event data)
  - Country of citizenship (sometimes used as a proxy for place of birth)

3.1.6 Migrant Transition Data

- Migrant transition data are typical collected in national censuses which identify migrants by comparing their place of usual residence at the time of enumeration (t) with that at a specified earlier date ($t - n$).
  - Time period is usually either 1 year (e.g. UK) or 5 years (e.g. USA)
  - Some countries have time periods as the interval between current and last census or significant time point in countries history.
- Transition data fail to identify multiple and return moves, and migrants who are born or who die during the measurement period.

3.1.7 Migrant Transition Data - Rees 1970

Figure 2. Two situations in which 'nonmigrants' make migrations.

3.1.8 Migrants and Migration

- Transition data are counts of migrants
- A migrant is a person who has in some specified period in the past experienced one or more migrations.
- Persons who moved during the interval and died before its end should, strictly speaking, be counted as migrants and their moves should be counted as migrations.
  - Likely to be excluded as information on migrants is usually obtained after the end of the interval and with reference to persons still living at that time.

3.1.9 Migration Event Data

- Event data record every move that is made by each individual

  - Include multiple and return migrations as well as moves by the newborn and those immediately before death.
  - Typically collected in population registers
  - Represent a more complete record of migration over time

- Geographical units for which the data are available are generally coarser and registers often fail to capture information on within-region moves.
- Less information about characteristics of migrants is usually available

  - Some groups may be omitted from altogether (e.g. prisoners, military personnel)

3.1.10 Migration Event Data

- There are important distinctions between the play (migration) and the actor (migrant).
- For a given migration interval, the number of migrants is rarely, if ever, as large as the number of migrations.

  - Unless the interval is very short (a day, or perhaps a week) some persons are certain to move more than once.
  - The longer the migration interval the more the count of migrants will understate the amount of migration.
  - Conversely, the shorter the migration interval, the count of migrants will approach the number of migrations.

- Shown by Courgeau [1979] …

3.1.11 Migration Event Data - Courgeau (1979)

Graphs 3 to 6. — Time distribution of first migrations and migrations of all ranks combined, by type of migration or move.

3.1.12 Migration Event Data

Graphs 3 to 6. — Time distribution of first migrations and migrations of all ranks combined, by type of migration or move.

3.2 Temporal

3.2.1 Migration Interval

- Migration occurs continuously over time. In order to study its incidence, data have to be compiled with reference to specified periods of time.
- The interval can be
  - Definite, e.g., one year, five years, ten years, the intercensal period
  - Indefinite, e.g., the lifetime of the population alive at a given date.
- Definite interval data typically called fixed-term or period migration (or surviving migrants in the example on the next slide)
- Lifetime migration or data based on place of last residence lack a definite time reference.

3.2.2 Migration Interval - Rees (1977)

Figure 1. Examples of the different sources for migration indicators.

3.2.3 One-Year Five-Year Problem

- Migration data is commonly collected over a one or five year interval.
- In places where both intervals are used, the number of migrants recorded over a five-year interval is far less than five times the number recorded over a one-year interval

- Seen in Courgeau (1979) plots or Rees(1970) tables - see next slides

- Rogerson [1990] no straightforward algebraic solution to comparing one-year and five-year migration probabilities.
- Event data has similar patterns to transition data for the same period, however

  - Width of the interval influences the intensity of migration and also the geographic pattern of captured migration flows.

3.2.4 One-Year Five-Year Problem - Rees 1970

Table 1. Five-year migrants, one-year migrants and five-year/one-year ratios for interregional migration, 1971 census (source: OPCS, 1974-1975; 1975).

Origin at start of period	Destination at end of period									
	1 N	2 YH	3 NW	4 EM	5 WM	6 EA	7 SE	8 SW	9 W	10 S
a. *Five-year migrants (1966-1971)*										
1 N	–	32460	21880	14460	13250	6480	47530	10660	3850	14820
2 YH	36300	–	40220	40460	18760	10810	62570	17360	7030	11270
3 NW	18740	34850	–	19060	29580	8540	84270	27100	33620	14710
4 EM	9570	35010	16280	–	25850	16180	57370	19320	6040	8450
5 WM	10230	15230	32110	35660	–	8540	76350	40720	21120	9780
6 EA	3390	7820	5220	12820	5510	–	53370	10020	3120	4030
7 SE	33720	47600	64890	73430	62750	109920	–	193920	36540	46140
8 SW	8010	10250	14930	12240	22840	9170	133950	–	14290	10800
9 W	3610	5590	17790	6740	18450	3470	42790	20400	–	4340
10 S	16110	13600	21640	14150	13820	5870	64150	13260	3990	–
11 RW	29320	60330	85110	51900	78020	51060	525190	71070	21900	62380

Key: N ≡ North, YH ≡ Yorkshire and Humberside, NW ≡ North West, EM ≡ East Midlands, WM ≡ West Midlands, EA ≡ East Anglia, SE ≡ South East, SW ≡ South West, W ≡ Wales, S ≡ Scotland, RW ≡ Rest of the World

3.2.5 One-Year Five-Year Problem - Rees 1970

Table 1. Five-year migrants, one-year migrants and five-year/one-year ratios for interregional migration, 1971 census (source: OPCS, 1974–1975; 1975).

Origin at start of period	Destination at end of period									
	1 N	2 YH	3 NW	4 EM	5 WM	6 EA	7 SE	8 SW	9 W	10 S
b. *One-year migrants (1970–1971)*										
1 N	–	9290	7370	3630	3790	2190	15040	3870	1220	4910
2 YH	12790	–	13290	13190	5970	4320	21580	5610	2570	3470
3 NW	6790	10650	–	6320	9750	2980	29210	8950	10370	5200
4 EM	4200	12170	5300	–	9610	6030	19110	6680	2330	3100
5 WM	3550	5400	9750	12600	–	3040	25630	12590	6350	3340
6 EA	1150	2510	1970	4490	2540	–	20460	3820	1170	1650
7 SE	12070	16570	22920	26390	21410	35410	–	66570	12050	7610
8 SW	2970	4600	4860	4230	8680	3390	49030	–	4730	4810
9 W	1450	2040	6390	2420	5650	1280	13160	7000	–	1650
10 S	5540	4810	7220	4910	4630	2070	23790	5330	1700	–
11 RW	14570	19360	27330	16650	23140	20280	197220	29390	8840	28160

Key: N ≡ North, YH ≡ Yorkshire and Humberside, NW ≡ North West, EM ≡ East Midlands, WM ≡ West Midlands, EA ≡ East Anglia, SE ≡ South East, SW ≡ South West, W ≡ Wales, S ≡ Scotland, RW ≡ Rest of the World

3.2.6 One-Year Five-Year Problem - Rees 1970

Table 1. Five-year migrants, one-year migrants and five-year/one-year ratios for interregional migration, 1971 census (source: OPCS, 1974–1975; 1975).

Origin at start of period	Destination at end of period									
	1 N	2 YH	3 NW	4 EM	5 WM	6 EA	7 SE	8 SW	9 W	10 S
c. *Ratio (five year/one year)*										
1 N	–	3·50	2·97	3·98	3·50	2·96	3·16	2·75	3·15	3·02
2 YH	2·84	–	3·03	3·07	3·14	2·50	2·90	3·10	2·73	3·25
3 NW	2·76	3·27	–	3·01	3·03	2·87	2·88	3·03	3·25	2·82
4 EM	2·28	2·87	3·07	–	2·69	2·68	3·00	2·89	2·59	2·72
5 WM	2·88	2·82	3·29	3·72	–	2·81	2·98	3·24	3·32	2·92
6 EA	2·95	3·12	2·65	2·86	2·17	–	2·61	2·62	2·67	2·44
7 SE	2·79	2·87	2·83	2·79	2·79	2·93	–	2·92	3·03	2·62
8 SW	2·70	2·23	3·07	2·89	2·63	2·70	2·73	–	3·02	2·25
9 W	2·49	2·74	2·79	2·79	3·27	2·71	3·25	2·92	–	2·63
10 S	2·91	2·82	2·99	2·88	2·99	2·83	2·70	2·49	2·35	–
11 RW	2·01	3·08	3·12	3·12	3·37	2·52	2·66	2·42	2·48	2·22

Key: N ≡ North, YH ≡ Yorkshire and Humberside, NW ≡ North West, EM ≡ East Midlands, WM ≡ West Midlands, EA ≡ East Anglia, SE ≡ South East, SW ≡ South West, W ≡ Wales, S ≡ Scotland, RW ≡ Rest of the World

3.3 Measures

3.3.1 Migration measures

- There are a range of migration measures over differing levels of details
  - Region to region
  - Region totals
  - System totals (or index measures)
- Measure might have multiple interchangeable names for the same terms.

3.3.2 Region to region measures

- *Streams* or *bilateral flow* or *origin-destination flow* is the total number of moves made during a given migration interval that have a common area of origin and a common area of destination.
- Data on migrations, or migrants, can be cross-classified by area of origin and area of destination.
  - Forms matrix of $n \times (n-1)$ streams, from area i to area j usually written as m_{ij}
- Contains a set of $\frac{n(n-1)}{2}$ pairs of streams, each pair representing movements in opposite directions (m_{ij} and m_{ji})

3.3.3 Region to region measures

- For a pair of streams that are of unequal size there exists a *dominant* streams and *counter-stream* or *reverse* stream
- Sum of the two members of a pair of streams is called *gross interchange*
- Differences between streams and counter-streams for individual pairs of streams, the balances are *net streams*.

3.3.4 Region totals

- Every move is an out-migration (emigration) with respect to the area of origin and an in-migration (immigration) with respect to the area of destination.
- Places of origin and destination dictate how describe migrants and migration

Scale	Area	Event Term	Migrant Term
Internal	Origin	out-migration	out-migrant
	Destination	in-migration	in-migrant
International	Origin	emigration	emigrant
	Destination	immigration	immigrant

3.3.5 Region totals

- Typically in- or out-migration are evaluated for each region
- Data collected or aggregated without reference to place of origin for in-migration totals, or destination for out-migration totals
- Beyond gross migration totals for each region other measures, other summary measures can also be derived

Term	Definition
Gross migration	All moves or all migrants
Turnover	Sum of in-migration and out-migration, or of in-migrants and out-migrants.
Net migration	Balance of movements in opposing directions from difference between in-migration and out-migration for a specific area

3.3.6 Region totals

- Never met a net-migrant
- Rogers [1990] *Requiem for net migration*
  - Net migration mis-specifies the spatial dynamics generating observed settlement patterns.
  - Obscure regularities in age profiles of migration
  - Net migration rates confound changing migration propensities with changing population stocks.

3.3.7 Rate measures

- Out-migration or emigration rates calculated by dividing events in a period by exposure population:

$$e^{[t,t+1]} = \frac{E^{[t,t+1]}}{P} k$$

where $e^{[t,t+1]}$ is the out or emigration rate, E is the number of out-migrants or emigrants during the period, P is the population exposed to the likelihood of migration during the period and k is a constant, usually 1000.

- Exposure population is typically either the
  - Population at the mid-interval, under the assumption that migration is uniformly distributed across the interval.
  - Population at the start or end of the interval under the assumption that migration has a negligible effect on population change.
- Can be decomposed by a subset of the population such age and/or sex

$$e_i^{[t,t+1]} = \frac{E_i^{[t,t+1]}}{P_i} k$$

3.3.8 Rates

- In-migration or immigration, the population exposed to the risk of migrating into a region is the entire population of the world living elsewhere.
- However, rates calculated by dividing events by the exposure time of the current residents (the population group **not** exposed to risk).

$$i^{[t,t+1]} = \frac{I^{[t,t+1]}}{P} k$$

- Net migration rates, like in-migration rates, are calculated by dividing events by the exposure time of the current residents (the population group **not** exposed to risk).

$$m^{[t,t+1]} = \frac{M^{[t,t+1]}}{P} k$$

3.3.9 Rates

- In-migration and net migration rates are unlike other demographic rates.
  - Not using the true population at risk in the denominator
- However, using the resident population satisfies the needs of the demographic balancing equation, since rates of gain and loss are measured relative to the same population.

$$\begin{aligned} P^{t+1} &= P^t + B^{[t,t+1]} - D^{[t,t+1]} + M^{[t,t+1]} \\ P^{t+1} &= P^t \left(1 + b^{[t,t+1]} - d^{[t,t+1]} + m^{[t,t+1]}\right) \\ &= P^t \left(1 + b^{[t,t+1]} - d^{[t,t+1]} + i^{[t,t+1]} - e^{[t,t+1]}\right) \end{aligned}$$

where we can substitute net migration $M^{[t,t+1]}$ with difference of in- and out-migration over the period $(I^{[t,t+1]} - O^{[t,t+1]})$

$$\begin{aligned} P^{t+1} &= P^t + B^{[t,t+1]} - D^{[t,t+1]} + I^{[t,t+1]} - O^{[t,t+1]} \\ &= P^t \left(1 + b^{[t,t+1]} - d^{[t,t+1]} + i^{[t,t+1]} - o^{[t,t+1]}\right) \end{aligned}$$

Chapter 4

Handling Migration Data in R

4.1 Contingency Table

4.1.1 Contingency Table

- Bilateral migration flow data are commonly represented in square tables.
- Values in non-diagonal cells represent a origin-destination count of migration between a specified set of regions.
- Values in diagonal cells represent some form of non-moving population, or those that move within a region, which are typically not presented.

Origin

Destination

A

B

C

D

Sum

A

100

30

70

200

B

50

45

5

100

C

60

35

40

135

D

20

25

20

65

Sum

130

160

95

115

500

4.1.2 Contingency Table

- Often denoted as m_{ij}
  - Row totals, the out-migration counts: $\sum_j m_{ij} = m_{i+}$
  - Column totals, the in-migration counts: $\sum_i m_{ij} = m_{+j}$
  - Net migration totals: $m_{i+} - m_{i+}$
  - Total migration: m_{++}

4.2 Data Manipulation

4.2.1 R `matrix` and `array`

- Some functions for describing and estimating migration in R require flow tables as `matrix` or `array` type objects
- Create a `matrix` in R using the `matrix()` function
  - Data read in by column. Change change using `byrow = FALSE`
  - Use the `dimnames` argument to supply region names

```
# create region labels
r <- LETTERS[1:4]
r
```

```
## [1] "A" "B" "C" "D"
```

```
# create matrix
m0 <- matrix(data = c(0, 100, 30, 70, 50, 0, 45, 5, 60, 35, 0, 40, 20, 25, 20, 0),
             nrow = 4, ncol = 4, byrow = TRUE,
             dimnames = list(orig = r, dest = r))
m0
```

```
##     dest
## orig  A   B  C  D
##    A  0 100 30 70
##    B 50   0 45  5
##    C 60  35  0 40
##    D 20  25 20  0
```

4.2.2 R `matrix` and `array`

- Create an `array` in R using the `array()` function

```
m1 <- array(data = sample(x = 1:100, size = 32),
            dim = c(4, 4, 2),
            dimnames = list(orig = r, dest = r, sex = c("female", "male")))
m1
```

```
## , , sex = female
##
##     dest
## orig  A  B  C  D
```

```
##    A 88 66 58 75
##    B 33 45 10  5
##    C 24 74 79 29
##    D 77 56 94 37
##
## , , sex = male
##
##     dest
## orig  A  B  C  D
##    A 86 16 87 46
##    B 65 54 14  8
##    C 95 63 43 71
##    D 38 91 68 70
```

4.2.3 Show totals

- The `addmargins()` functions adds extra row, column and tables to display the dimension sums.

```
addmargins(A = m0)
```

```
##      dest
## orig    A   B  C   D Sum
##   A     0 100 30  70 200
##   B    50   0 45   5 100
##   C    60  35  0  40 135
##   D    20  25 20   0  65
##   Sum 130 160 95 115 500
```

4.2.4 Convert to matrix

- Data will not always come as an `matrix` or an `array`.
- There a couple of useful functions in R to convert data to when working with migration tables in R
- The `xtab()` function converts data frames into a `matrix` or `array`
  - `formula` column names with
    - left hand side the column name to fill the `matrix` or `array`
    - a `~` to separate the left and right hand side
    - right hand side the columns to cross-classifying the left hand variable (separated by `+`).
  - `data` containing the variables for `formula`

- The `as.data.frame.table()` function takes a `matrix` or `array` and converts it to a `data.frame` based on the array dimension names.
  - `responseName` to set the column name of based on the cells of the `matrix` or `array`

4.2.5 Convert to matrix

```
# tidy migration data
d0
```

```
## # A tibble: 16 x 3
##    orig  dest   flow
##    <chr> <chr> <int>
##  1 A     A         1
##  2 A     B         2
##  3 A     C         3
##  4 A     D         4
##  5 B     A         5
##  6 B     B         6
##  7 B     C         7
##  8 B     D         8
##  9 C     A         9
## 10 C     B        10
## 11 C     C        11
## 12 C     D        12
## 13 D     A        13
## 14 D     B        14
## 15 D     C        15
## 16 D     D        16
```

4.2.6 Convert to matrix

```
# convert to matrix
m2 <- xtabs(formula = flow ~ orig + dest, data = d0)
m2
```

```
##     dest
## orig  A  B  C  D
##    A  1  2  3  4
##    B  5  6  7  8
##    C  9 10 11 12
##    D 13 14 15 16
```

4.2.7 Convert to data frame

```
# convert back to tibble
m2 %>%
  as.data.frame.table(responseName = "flow") %>%
  as_tibble()
```

```
## # A tibble: 16 x 3
##    orig  dest   flow
##    <fct> <fct> <int>
##  1 A     A         1
##  2 B     A         5
##  3 C     A         9
##  4 D     A        13
##  5 A     B         2
##  6 B     B         6
##  7 C     B        10
##  8 D     B        14
##  9 A     C         3
## 10 B     C         7
## 11 C     C        11
## 12 D     C        15
## 13 A     D         4
## 14 B     D         8
## 15 C     D        12
## 16 D     D        16
```

4.2.8 Convert to data frame

```
# convert array to tibble
d1 <- m1 %>%
  as.data.frame.table(responseName = "flow") %>%
  as_tibble()
d1
```

```
## # A tibble: 32 x 4
##    orig  dest  sex     flow
##    <fct> <fct> <fct>  <int>
##  1 A     A     female    88
##  2 B     A     female    33
##  3 C     A     female    24
##  4 D     A     female    77
```

```
##  5 A     B     female    66
##  6 B     B     female    45
##  7 C     B     female    74
##  8 D     B     female    56
##  9 A     C     female    58
## 10 B     C     female    10
## # ... with 22 more rows
```

4.3 Matrix Operations

4.3.1 Displaying migration matrics

- When dealing with migration `matrix` objects in R, they often are difficult to view
  - Lengthy dimension names,
  - Unit size
  - Diagonal terms included but not of interest
- Some helpful R functions to adapt objects for easier viewing
- Demonstrate with the `uar_1960` object in the *migest* package
  - Lifetime migration matrix for Governorates of United Arab Republic in 1960 used in the manual of United Nations Department of Economic and Social Affairs Population Division [1983]

4.3.2 Displaying migration matrics

```
library(migest)
uar_1960
```

```
##             dest
## orig           Cairo Alexandria Port-Said Ismailia Kalyubia Gharbia Menoufia
##   Cairo      2079434      31049      5293     9813    23837   10034     7038
##   Alexandria   47220    1085602      2641     2625     2135    4921     1505
##   Port-Said     9464       2562    168046     6461      496     817      323
##   Ismailia      9518       1395      3490   171297      718     910      306
##   Kalyubia     90668       4730       758     3182   886464    3727     3523
##   Gharbia      99179      39953      1742     3347     7870 1604851     6313
##   Menoufia    216764      46781      1640     3338     2918   29580  1308283
##   Giza         64584       4899       513     2013     2887    1503     2161
##   Assyiut     100305      25497      1738     2522      122    2245      636
##   Souhag      100100      63712     12087     9436      295    2791     1095
```

```
##   All others  456464     177476     43898    66973    49816   47315    12179
##             dest
## orig            Giza Assyiut  Souhag All others
##   Cairo        88543    4951    2569      58476
##   Alexandria    6910    1355    1467      29534
##   Port-Said     1505     326     454      11184
##   Ismailia      1593     319     263      10269
##   Kalyubia     10279     340     128      18076
##   Gharbia      14529     848     491      64140
##   Menoufia     30915     567     401      47843
##   Giza       1040179     540     433      13518
##   Assyiut      13153 1290255    5955      35157
##   Souhag       17958   11608 1540020      53224
##   All others   94577   14690   22375   11900302
```

4.3.3 Abbriviate names

- View and alter the `matrix` dimension names using `rownames()` and `colnames()` or `dimnames()`
- The `abbreviate()` function applies an algorithm to shorten names

```
dimnames(uar_1960)
```

```
## $orig
##  [1] "Cairo"      "Alexandria" "Port-Said"  "Ismailia"   "Kalyubia"
##  [6] "Gharbia"    "Menoufia"   "Giza"       "Assyiut"    "Souhag"
## [11] "All others"
##
## $dest
##  [1] "Cairo"      "Alexandria" "Port-Said"  "Ismailia"   "Kalyubia"
##  [6] "Gharbia"    "Menoufia"   "Giza"       "Assyiut"    "Souhag"
## [11] "All others"
```

```
# make a copy
u0 <- uar_1960
# new abbreviated region names
r <- list(orig = uar_1960 %>%
            rownames() %>%
            abbreviate(),
          dest = uar_1960 %>%
            colnames() %>%
            abbreviate())
```

4.3.4 Abbriviate names

```
r
```

```
## $orig
##      Cairo Alexandria  Port-Said   Ismailia   Kalyubia    Gharbia   Menoufia
##     "Cair"     "Alxn"     "Pr-S"     "Isml"     "Klyb"     "Ghrb"     "Menf"
##       Giza    Assyiut     Souhag All others
##     "Giza"     "Assy"     "Sohg"     "Allo"
##
## $dest
##      Cairo Alexandria  Port-Said   Ismailia   Kalyubia    Gharbia   Menoufia
##     "Cair"     "Alxn"     "Pr-S"     "Isml"     "Klyb"     "Ghrb"     "Menf"
##       Giza    Assyiut     Souhag All others
##     "Giza"     "Assy"     "Sohg"     "Allo"
```

```
# apply the abbreviated region names
dimnames(u0) <- r
```

4.3.5 Abbriviate names

```
u0
```

```
##       dest
## orig      Cair    Alxn   Pr-S   Isml   Klyb    Ghrb    Menf    Giza    Assy
##   Cair 2079434   31049   5293   9813  23837   10034    7038   88543    4951
##   Alxn   47220 1085602   2641   2625   2135    4921    1505    6910    1355
##   Pr-S    9464    2562 168046   6461    496     817     323    1505     326
##   Isml    9518    1395   3490 171297    718     910     306    1593     319
##   Klyb   90668    4730    758   3182 886464    3727    3523   10279     340
##   Ghrb   99179   39953   1742   3347   7870 1604851    6313   14529     848
##   Menf  216764   46781   1640   3338   2918   29580 1308283   30915     567
##   Giza   64584    4899    513   2013   2887    1503    2161 1040179     540
##   Assy  100305   25497   1738   2522    122    2245     636   13153 1290255
##   Sohg  100100   63712  12087   9436    295    2791    1095   17958   11608
##   Allo  456464  177476  43898  66973  49816   47315   12179   94577   14690
##       dest
## orig      Sohg     Allo
##   Cair    2569    58476
##   Alxn    1467    29534
##   Pr-S     454    11184
##   Isml     263    10269
```

```
##   Klyb     128    18076
##   Ghrb     491    64140
##   Menf     401    47843
##   Giza     433    13518
##   Assy    5955    35157
##   Sohg 1540020    53224
##   Allo   22375 11900302
```

4.3.6 Data scaling

- Basic arithmetic operators to scale the data to an appropriate level
- The `round()` function to specify precision of numbers

```
u1 <- round(x = u0/1000, digits = 1)
u1
```

```
##       dest
## orig     Cair   Alxn  Pr-S  Isml  Klyb   Ghrb   Menf   Giza   Assy   Sohg
##   Cair 2079.4   31.0   5.3   9.8  23.8   10.0    7.0   88.5    5.0    2.6
##   Alxn   47.2 1085.6   2.6   2.6   2.1    4.9    1.5    6.9    1.4    1.5
##   Pr-S    9.5    2.6 168.0   6.5   0.5    0.8    0.3    1.5    0.3    0.5
##   Isml    9.5    1.4   3.5 171.3   0.7    0.9    0.3    1.6    0.3    0.3
##   Klyb   90.7    4.7   0.8   3.2 886.5    3.7    3.5   10.3    0.3    0.1
##   Ghrb   99.2   40.0   1.7   3.3   7.9 1604.9    6.3   14.5    0.8    0.5
##   Menf  216.8   46.8   1.6   3.3   2.9   29.6 1308.3   30.9    0.6    0.4
##   Giza   64.6    4.9   0.5   2.0   2.9    1.5    2.2 1040.2    0.5    0.4
##   Assy  100.3   25.5   1.7   2.5   0.1    2.2    0.6   13.2 1290.3    6.0
##   Sohg  100.1   63.7  12.1   9.4   0.3    2.8    1.1   18.0   11.6 1540.0
##   Allo  456.5  177.5  43.9  67.0  49.8   47.3   12.2   94.6   14.7   22.4
##       dest
## orig      Allo
##   Cair    58.5
##   Alxn    29.5
##   Pr-S    11.2
##   Isml    10.3
##   Klyb    18.1
##   Ghrb    64.1
##   Menf    47.8
##   Giza    13.5
##   Assy    35.2
##   Sohg    53.2
##   Allo 11900.3
```

4.3.7 Diagonal elements

- Set diagonal terms (non-movers) to zero using the `diag()` function

```
u2 <- u0
diag(u2) <- 0
u2
```

```
##       dest
## orig     Cair   Alxn  Pr-S  Isml  Klyb  Ghrb  Menf  Giza  Assy  Sohg  Allo
##   Cair      0  31049  5293  9813 23837 10034  7038 88543  4951  2569 58476
##   Alxn  47220      0  2641  2625  2135  4921  1505  6910  1355  1467 29534
##   Pr-S   9464   2562     0  6461   496   817   323  1505   326   454 11184
##   Isml   9518   1395  3490     0   718   910   306  1593   319   263 10269
##   Klyb  90668   4730   758  3182     0  3727  3523 10279   340   128 18076
##   Ghrb  99179  39953  1742  3347  7870     0  6313 14529   848   491 64140
##   Menf 216764  46781  1640  3338  2918 29580     0 30915   567   401 47843
##   Giza  64584   4899   513  2013  2887  1503  2161     0   540   433 13518
##   Assy 100305  25497  1738  2522   122  2245   636 13153     0  5955 35157
##   Sohg 100100  63712 12087  9436   295  2791  1095 17958 11608     0 53224
##   Allo 456464 177476 43898 66973 49816 47315 12179 94577 14690 22375     0
```

4.4 Summaries

4.4.1 Net flows and counterflows

- The *migest* package contains a number of functions to provide summaries of origin-destination migration data
- The `counter()` function calculates the counter flow and net flow
  - Accepts `matrix` or `data.frame` (or `tibble`) inputs

```
counter(m0)
```

```
## # A tibble: 12 x 7
##    orig  dest  corridor pair   flow counter_flow net_flow
##    <chr> <chr> <chr>    <chr> <dbl>        <dbl>    <dbl>
##  1 B     A     B -> A   A - B    50          100      -50
##  2 C     A     C -> A   A - C    60           30       30
##  3 D     A     D -> A   A - D    20           70      -50
##  4 A     B     A -> B   A - B   100           50       50
##  5 C     B     C -> B   B - C    35           45      -10
##  6 D     B     D -> B   B - D    25            5       20
```

```
##  7 A     C     A -> C   A - C    30           60      -30
##  8 B     C     B -> C   B - C    45           35       10
##  9 D     C     D -> C   C - D    20           40      -20
## 10 A     D     A -> D   A - D    70           20       50
## 11 B     D     B -> D   B - D     5           25      -20
## 12 C     D     C -> D   C - D    40           20       20
```

4.4.2 Net flows and counterflows

```
d1 %>%
  group_by(sex) %>%
  counter()
```

```
## # A tibble: 24 x 8
## # Groups:   sex [2]
##    orig  dest  corridor pair  sex     flow counter_flow net_flow
##    <chr> <chr> <chr>    <chr> <fct>  <int>        <int>    <int>
##  1 B     A     B -> A   A - B female    33           66      -33
##  2 C     A     C -> A   A - C female    24           58      -34
##  3 D     A     D -> A   A - D female    77           75        2
##  4 A     B     A -> B   A - B female    66           33       33
##  5 C     B     C -> B   B - C female    74           10       64
##  6 D     B     D -> B   B - D female    56            5       51
##  7 A     C     A -> C   A - C female    58           24       34
##  8 B     C     B -> C   B - C female    10           74      -64
##  9 D     C     D -> C   C - D female    94           29       65
## 10 A     D     A -> D   A - D female    75           77       -2
## # ... with 14 more rows
```

4.4.3 Totals

- The `sum_turnover()` provides summary in-migration, out-migration, net-migration and turnover totals for each region
  - Accepts `matrix` or `data.frame` (or `tibble`) inputs
  - Setting `type = "international"` to change labels in outputs

```
sum_turnover(m0)
```

```
## # A tibble: 4 x 5
##   region in_mig out_mig  turn   net
##   <chr>   <dbl>   <dbl> <dbl> <dbl>
```

```
## 1 A         130     200   330   -70
## 2 B         160     100   260    60
## 3 C          95     135   230   -40
## 4 D         115      65   180    50
```

4.4.4 Totals

```
sum_turnover(m = d0, type = "international")
```

```
## # A tibble: 4 x 5
##   country   imm   emi  turn   net
##   <chr>   <dbl> <dbl> <dbl> <dbl>
## 1 A          27     9    36    18
## 2 B          26    20    46     6
## 3 C          25    31    56    -6
## 4 D          24    42    66   -18
```

4.4.5 Totals

- The `sum_turnover()` function can be applied with to large data sets spanning multiple years (groups)
- Demonstrate using international flow estimates of Abel and Cohen [2019]

```
# read data from web depository
f <- read_csv("https://ndownloader.figshare.com/files/26239945")
f
```

```
## # A tibble: 235,236 x 9
##    year0 orig  dest  sd_drop_neg sd_rev_neg mig_rate da_min_open da_min_closed
##    <dbl> <chr> <chr>       <dbl>      <dbl>    <dbl>       <dbl>         <dbl>
##  1  1990 BDI   BDI             0          0        0           0             0
##  2  1990 COM   BDI             0          0        0           0             0
##  3  1990 DJI   BDI             0          0        0           0             0
##  4  1990 ERI   BDI             0          0        0           0             0
##  5  1990 ETH   BDI             0          0        0           0             0
##  6  1990 KEN   BDI            30         30       69          45            29
##  7  1990 MDG   BDI             0          0        0           0             0
##  8  1990 MWI   BDI             0          0        0           0             0
##  9  1990 MUS   BDI             0          0        0           0             1
## 10  1990 MYT   BDI             0          0        0           0             0
## # ... with 235,226 more rows, and 1 more variable: da_pb_closed <dbl>
```

4.4.6 Totals

```
# single period
f %>%
  filter(year0 == 1990) %>%
  sum_turnover(flow_col = "da_pb_closed", type = "international")
```

```
## # A tibble: 197 x 5
##    country     imm    emi    turn     net
##    <chr>     <dbl>  <dbl>   <dbl>   <dbl>
##  1 BDI       61630 381611  443241 -319981
##  2 COM        9009  12011   21020   -3002
##  3 DJI       10949  55945   66894  -44996
##  4 ERI       14633 329383  344016 -314750
##  5 ETH     1635513 177334 1812847 1458179
##  6 KEN      306517  84833  391350  221684
##  7 MDG        9706  19159   28865   -9453
##  8 MWI      112416 974278 1086694 -861862
##  9 MUS       16862  22475   39337   -5613
## 10 MYT       13763   3021   16784   10742
## # ... with 187 more rows
```

4.4.7 Totals

```
# all periods using group_by
f %>%
  group_by(year0) %>%
  sum_turnover(flow_col = "da_pb_closed", type = "international") %>%
  arrange(country)
```

```
## Adding missing grouping variables: `year0`
```

```
## # A tibble: 1,188 x 6
## # Groups:   year0 [6]
##    year0 country     imm     emi    turn      net
##    <dbl> <chr>     <dbl>   <dbl>   <dbl>    <dbl>
##  1  1990 ABW       15874    1662   17536    14212
##  2  1995 ABW       10945    4007   14952     6938
##  3  2000 ABW       10064    3814   13878     6250
##  4  2005 ABW        7124    7544   14668     -420
##  5  2010 ABW        9910    8654   18564     1256
```

```
##  6  2015 ABW       17316   16306   33622     1010
##  7  1990 AFG     3421712  345255 3766967  3076457
##  8  1995 AFG      418906 1286436 1705342  -867530
##  9  2000 AFG     1178865  434706 1613571   744159
## 10  2005 AFG      457339 1500149 1957488 -1042810
## # ... with 1,178 more rows
```

4.4.8 Rest of categroies

- The `sum_lump()` function can be used to aggregate up smaller regions.
  - Specify the the desired level of small flows using the `threshold` argument
  - Specify the `lump` argument to apply the `threshold` argument to either the `flow` values or the `in` and `out` totals.

```
m0
```

```
##     dest
## orig  A   B  C  D
##    A  0 100 30 70
##    B 50   0 45  5
##    C 60  35  0 40
##    D 20  25 20  0
```

```
# threshold on flows (default)
sum_lump(m0, threshold = 50)
```

```
## # A tibble: 5 x 3
##   orig  dest   flow
##   <chr> <chr> <dbl>
## 1 A     B       100
## 2 A     D        70
## 3 B     A        50
## 4 C     A        60
## 5 other other   220
```

4.4.9 Rest of categroies

```
addmargins(m0)
```

```
##      dest
## orig    A   B  C   D Sum
##   A     0 100 30  70 200
##   B    50   0 45   5 100
##   C    60  35  0  40 135
##   D    20  25 20   0  65
##   Sum 130 160 95 115 500
```

```
# threshold on in and out totals
sum_lump(m0, threshold = 120, lump = c("in", "out"))
```

```
## # A tibble: 9 x 3
##   orig  dest   flow
##   <chr> <chr> <dbl>
## 1 A     A         0
## 2 A     C        30
## 3 A     other   170
## 4 B     A        50
## 5 B     C        45
## 6 B     other     5
## 7 other A        80
## 8 other C        20
## 9 other other   100
```

4.4.10 Rest of categroies

- Useful to reduce the number of corridors when plotting large data sets:

```
# add continental regions to the global flow data set
library(countrycode)
d <- f %>%
  filter(year0 == 2015) %>%
  mutate(
    orig_reg =
      countrycode(sourcevar = orig, origin = "iso3c", dest = "un.region.name"),
    dest_reg =
      countrycode(sourcevar = dest, origin = "iso3c", dest = "un.region.name")) %>%
  relocate(contains("orig"), contains("dest"))
d
```

```
## # A tibble: 40,000 x 11
##    orig  orig_reg dest  dest_reg year0 sd_drop_neg sd_rev_neg mig_rate
##    <chr> <chr>    <chr> <chr>    <dbl>       <dbl>      <dbl>    <dbl>
##  1 BDI   Africa   BDI   Africa    2015           0          0        0
```

```
##  2 COM   Africa   BDI   Africa    2015           0          0        0
##  3 DJI   Africa   BDI   Africa    2015           0          0        0
##  4 ERI   Africa   BDI   Africa    2015           0        131        0
##  5 ETH   Africa   BDI   Africa    2015           0         14        0
##  6 KEN   Africa   BDI   Africa    2015         194        194      211
##  7 MDG   Africa   BDI   Africa    2015           0          0        0
##  8 MWI   Africa   BDI   Africa    2015           0          0        0
##  9 MUS   Africa   BDI   Africa    2015           0          0        0
## 10 MYT   Africa   BDI   Africa    2015           0          0        0
## # ... with 39,990 more rows, and 3 more variables: da_min_open <dbl>,
## #   da_min_closed <dbl>, da_pb_closed <dbl>
```

4.4.11 Rest of categroies

- Apply the `sum_lump()` function to lump together smaller flows (less than 100,000) within and between continents.

```
d %>%
  group_by(orig_reg, dest_reg) %>%
  sum_lump(threshold = 1e5, flow_col = "da_pb_closed")
```

```
## # A tibble: 221 x 5
## # Groups:   orig_reg, dest_reg [36]
##    orig_reg dest_reg orig  dest     flow
##    <chr>    <chr>    <chr> <chr>   <dbl>
##  1 Africa   Africa   BFA   CIV    329531
##  2 Africa   Africa   CAF   COD    163440
##  3 Africa   Africa   CIV   BFA    260320
##  4 Africa   Africa   CIV   MLI    107902
##  5 Africa   Africa   COD   UGA    111439
##  6 Africa   Africa   MLI   CIV    138475
##  7 Africa   Africa   MOZ   ZAF    112554
##  8 Africa   Africa   other other 5091888
##  9 Africa   Africa   SDN   SSD    380532
## 10 Africa   Africa   SDN   TCD    121964
## # ... with 211 more rows
```

4.4.12 Exercise (ex2.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
```

```
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(migest)
##
##
##
# 1. Run the code below to read in the bilateral data in uk_census_2011_tidy.csv
#    from the ONS 2011 British Census
uk <- read_csv("./data/uk_census_2011_tidy.csv")
uk
# 2. Create a 12 by 12 origin-destination matrix m based on the bilateral flows
#    given in data frame uk
m <- #####(formula = flow ~ orig + #####, data = #####)
m
# 3. Print the matrix m again, this time include the in- and out-migration
#    sum totals
#####(m1)
# 4. Create a 12 by 12 by 2 sex-specific origin-destination array based on the
#    bilateral flows given in data frame uk
s <- #####(formula = ##### ~ ##### + dest + #####, data = uk)
s
# 5. Run the code below to check that s has 12 x 12 x 2 dimensions
dim(s)
# 6. Convert object s from above into a tibble with four columns, orig, dest,
#    sex and flow
d <- s %>%
  #####(responseName = "#####") %>%
  #####()
d
# 7. Calculate the counter-flow and net-flow for each migration pair in the
#    matrix m. Use the arrange() function to show the top 10 migration corridors
#    with biggest net losses
m %>%
  #####() %>%
  arrange(net_flow)
# 8. Calculate the sex-specific in-migration, out-migration, turnover and net
#    migration totals for each origin in s. Arrange the results by the smallest
#    turnover totals
s %>%
  #####(responseName = "flow") %>%
  group_by(#####) %>%
  #####() %>%
  arrange(turn)
```

Chapter 5

Summary Migration Indices

5.1 Background

5.1.1 Background

- Compared with fertility and mortality, little attention given to the way that internal or domestic migration varies between nations.
- Comparisons of migration over time or between spatial units is complicated by many factors including:
  - Different or changing definitions of migration
  - Different or changing collection systems for migration
  - Different sizes of regions
  - Different or changing number of regions
  - Different ways of measuring distances
  - Different and changing underlying population sizes and structures
- Bell et al. [2002] brought together and proposed a number of summary measures to enable better comparisons
  - A number of these incorporated into the IMAGE software for comparisons of internal migration between many countries; Bell et al. [2015b], Bell et al. [2015a], Stillwell et al. [2016], Rees et al. [2016], Bernard et al. [2014]

5.1.2 Background

- Bell et al. [2002] identified four main groups of migration indices:
  - Intensity of migration
  - Distance of migration

  - Migration connectivity
  - Effect of migration on the redistribution of populations

5.2 Intensity

5.2.1 Migration intensity

- Migration intensity measures attempt to capture the overall level, or incidence, of mobility.
- Provide a single measure for comparison of migration intensities over time or space
- Some indices based on age-specific migration data
  - Will discuss later in the age-schedule section

5.2.2 Crude migration probability

- Crude migration intensity is a simple measure of the overall propensity to migrate
  - Similar to crude birth or death rate
- If using migration transition data, the crude migration probability (CMP) is

$$\texttt{CMP} = 100M/P$$

  where M is the total number of migrants in a given time period and P is the population at risk

5.2.3 Migration intensity

- Courgeau [1973] discussed how the intensity of migration is directly related to the number of regions n in the country

$$\texttt{CMP} = k\log(n^2)$$

- No intrinsic meaning to a single Courgeau's k , but can be used to compare migration intensity that cannot be seen from the raw data because of differences in their zonal systems.
- Higher the value of k, greater the intensity of migration

5.2.4 Migration intensity

- The `index_intensity()` function in the *migest* package calculates both intensity measures, given a migration and population data
- The *migest* package also contains a data set on Korean internal migration and populations of first level administrative districts
  - Data originally downloaded from `https://kosis.kr/eng`

```
library(tidyverse)
library(migest)
korea_reg
```

```
## # A tibble: 2,601 x 4
##    orig  dest         year    flow
##    <fct> <fct>       <int>   <int>
##  1 Seoul Seoul        2012 1069300
##  2 Seoul Busan        2012   21437
##  3 Seoul Daegu        2012   13838
##  4 Seoul Incheon      2012   32216
##  5 Seoul Gwangju      2012   11811
##  6 Seoul Daejeon      2012   14570
##  7 Seoul Ulsan        2012    6799
##  8 Seoul Sejong       2012    1015
##  9 Seoul Gyeonggi-do  2012  254175
## 10 Seoul Gangwon-do   2012   21324
## # ... with 2,591 more rows
```

5.2.5 Migration intensity

- The `korea_pop` contains the resident population in each region and year

```
korea_pop
```

```
## # A tibble: 153 x 3
##    region  year population
##    <chr>  <int>      <dbl>
##  1 Seoul   2012   10195318
##  2 Seoul   2013   10143645
##  3 Seoul   2014   10103233
##  4 Seoul   2015   10022181
##  5 Seoul   2016    9930616
##  6 Seoul   2017    9857426
##  7 Seoul   2018    9765623
```

```
##  8 Seoul   2019    9729107
##  9 Seoul   2020    9668465
## 10 Busan   2012    3538484
## # ... with 143 more rows
```

5.2.6 Migration intensity

- Calculate migration and population totals in 2020

```
m <- korea_reg %>%
  filter(year == 2020,
         orig != dest) %>%
  pull(flow) %>%
  sum()
m
```

```
## [1] 2534114
```

```
p <- korea_pop %>%
  filter(year == 2020) %>%
  pull(population) %>%
  sum()
p
```

```
## [1] 51829023
```

```
index_intensity(mig_total = m, pop_total = p,
                n = n_distinct(korea_pop$region))
```

```
## # A tibble: 2 x 2
##   measure    value
##   <chr>      <dbl>
## 1 cmp        4.89
## 2 courgeau_k 0.863
```

5.2.7 Migration intensity

```
mm <- korea_reg %>%
  group_by(year) %>%
  filter(orig != dest) %>%
  summarise(m = sum(flow))
mm
```

```
## # A tibble: 9 x 2
##    year       m
##   <int>   <int>
## 1  2012 2512740
## 2  2013 2423429
## 3  2014 2507796
## 4  2015 2551424
## 5  2016 2453342
## 6  2017 2410930
## 7  2018 2429184
## 8  2019 2384948
## 9  2020 2534114
```

5.2.8 Migration intensity

```
pp <- korea_pop %>%
  group_by(year) %>%
  summarise(p = sum(population))
pp
```

```
## # A tibble: 9 x 2
##    year        p
##   <int>    <dbl>
## 1  2012 50948272
## 2  2013 51141463
## 3  2014 51327916
## 4  2015 51529338
## 5  2016 51696216
## 6  2017 51778544
## 7  2018 51826059
## 8  2019 51849861
## 9  2020 51829023
```

5.2.9 Migration intensity

- Passing the vectors of migration and population totals can lead to confusing output

```
d <- mm %>%
  left_join(pp)
d
```

```
## # A tibble: 9 x 3
##    year       m        p
##   <int>   <int>    <dbl>
## 1  2012 2512740 50948272
## 2  2013 2423429 51141463
## 3  2014 2507796 51327916
## 4  2015 2551424 51529338
## 5  2016 2453342 51696216
## 6  2017 2410930 51778544
## 7  2018 2429184 51826059
## 8  2019 2384948 51849861
## 9  2020 2534114 51829023
```

5.2.10 Migration intensity

```
index_intensity(mig_total = d$m, pop_total = d$p,
                n = n_distinct(korea_pop$region))
```

```
## # A tibble: 18 x 2
##    measure    value
##    <chr>      <dbl>
##  1 cmp        4.93
##  2 courgeau_k 0.870
##  3 cmp        4.74
##  4 courgeau_k 0.836
##  5 cmp        4.89
##  6 courgeau_k 0.862
##  7 cmp        4.95
##  8 courgeau_k 0.874
##  9 cmp        4.75
## 10 courgeau_k 0.838
## 11 cmp        4.66
## 12 courgeau_k 0.822
## 13 cmp        4.69
## 14 courgeau_k 0.827
## 15 cmp        4.60
## 16 courgeau_k 0.812
## 17 cmp        4.89
## 18 courgeau_k 0.863
```

5.2.11 Migration intensity

- Set `long = FALSE` to put each indicator in their own column

```
index_intensity(mig_total = d$m, pop_total = d$p,
                n = n_distinct(korea_pop$region), long = FALSE)
```

```
## # A tibble: 9 x 2
##     cmp courgeau_k
##   <dbl>      <dbl>
## 1  4.93      0.870
## 2  4.74      0.836
## 3  4.89      0.862
## 4  4.95      0.874
## 5  4.75      0.838
## 6  4.66      0.822
## 7  4.69      0.827
## 8  4.60      0.812
## 9  4.89      0.863
```

5.2.12 Migration intensity

- Use the `map2()` function in *purrr* to apply the function to subsets of the data
  - Provides results alongside the `year`
- Code below produces a nested list `i` containing the intensity measures for each year

```
d <- mm %>%
  left_join(pp) %>%
  mutate(i = map2(.x = m, .y = p,
                  .f = ~index_intensity(mig_total = .x,
                                        pop_total = .y,
                                        n = n_distinct(korea_pop$region),
                                        long = FALSE)))
d
```

```
## # A tibble: 9 x 4
##    year       m        p i
##   <int>   <int>    <dbl> <list>
## 1  2012 2512740 50948272 <tibble [1 x 2]>
## 2  2013 2423429 51141463 <tibble [1 x 2]>
## 3  2014 2507796 51327916 <tibble [1 x 2]>
## 4  2015 2551424 51529338 <tibble [1 x 2]>
## 5  2016 2453342 51696216 <tibble [1 x 2]>
## 6  2017 2410930 51778544 <tibble [1 x 2]>
```

```
## 7  2018 2429184 51826059 <tibble [1 x 2]>
## 8  2019 2384948 51849861 <tibble [1 x 2]>
## 9  2020 2534114 51829023 <tibble [1 x 2]>
```

5.2.13 Migration intensity

- The `unnest()` function in the *tidyr* package binds each component of column `i` on top of each other
  - Easier to see changes over time

```
unnest(d, cols = i)
```

```
## # A tibble: 9 x 5
##    year       m        p   cmp courgeau_k
##   <int>   <int>    <dbl> <dbl>      <dbl>
## 1  2012 2512740 50948272  4.93      0.870
## 2  2013 2423429 51141463  4.74      0.836
## 3  2014 2507796 51327916  4.89      0.862
## 4  2015 2551424 51529338  4.95      0.874
## 5  2016 2453342 51696216  4.75      0.838
## 6  2017 2410930 51778544  4.66      0.822
## 7  2018 2429184 51826059  4.69      0.827
## 8  2019 2384948 51849861  4.60      0.812
## 9  2020 2534114 51829023  4.89      0.863
```

5.3 Distance

5.3.1 Migration distance

- As migration is a spatial activity, based on movements between two locations, comparisons should take account of the way that intensities of movement vary across space.
- There are a number of measures that summarize the effects of distance across a migration system
- The distance measure between each region is not straightforward
  - Ideally measure the typical distance that migrants travel.
  - Straight line distance between population-weighted centroids of each region provide a good approximation
- The costs faced by a migrant may not be represented well by the inter-centroid distance.

  - Locations in different regions might be very close to a border, so centroids will exaggerate the distance
  - Areas can take many shapes and sizes
  - Doughnut shaped regions might have centroids not in region
  - Indented coastlines might make regions look closer than they might be (culturally, travel cost)
  - Distance measures for within region moves cause another set of problems

5.3.2 Average migration distance

- Summary of the average migration distance can be calculated by taking a weighted average of the migration counts, where the corresponding distances are the weights
- Bell et al. [2002] note a median average as clearly preferable to a mean average as the distribution of distances is negatively skewed, reflecting the strong distance decay effect which consistently occurs
- Comparison with the mean average distance provides a guide to the skewness

5.3.3 Distance decay

- A more complete method to account for the skewness in migration distances is to fit a model to predict migration counts using the distances between each regions and extract the distance parameter
- Different models could be potentially used, but tend to be based on a `log(distance)` terms with categorical control variables for the origin and destination
- The distance decay parameter (β_3) in a Poisson log-linear model;

$$\log(m_{ij}) = \beta_0 + \beta_{1i}\texttt{origin} + \beta_{2j}\texttt{destination} + \beta_3 \log(\texttt{distance})$$

- The distance decay parameter of interest (β_3 in the equation above) is almost always negative, indicating an increase in migration leads to fewer predicted migrations.
  - The set of β_{1i} and β_{2j} represent some form of push and pull factors for each region (i and j)

5.3.4 Migration distance

- The `korea_dist` matrix provides estimates of the 2020 population weighted distances in kilometers between the 17 first level administrative districts in Korea

– Data based on WorldPop estimates of Edwards et al. [2021]

```
korea_dist
```

```
##                    dest
## orig                Busan Chungcheongbuk-do Chungcheongnam-do Daegu Daejeon
##   Busan                 0               216               255    88     201
##   Chungcheongbuk-do   216                 0                58   130      45
##   Chungcheongnam-do   255                58                 0   174      54
##   Daegu                88               130               174     0     122
##   Daejeon             201                45                54   122       0
##   Gangwon-do          286               122               159   203     166
##   Gwangju             202               184               170   175     139
##   Gyeonggi-do         312                96                83   225     125
##   Gyeongsangbuk-do    107               125               176    32     128
##   Gyeongsangnam-do     43               190               222    72     168
##   Incheon             328               111                85   242     134
##   Jeju                306               381               368   333     337
##   Jeollabuk-do        201               111                96   143      66
##   Jeollanam-do        190               205               197   177     162
##   Sejong              222                37                34   141      22
##   Seoul               324               108                95   236     138
##   Ulsan                47               203               250    76     198
##                    dest
## orig                Gangwon-do Gwangju Gyeonggi-do Gyeongsangbuk-do
##   Busan                    286     202         312              107
##   Chungcheongbuk-do        122     184          96              125
##   Chungcheongnam-do        159     170          83              176
##   Daegu                    203     175         225               32
##   Daejeon                  166     139         125              128
##   Gangwon-do                 0     305         114              179
##   Gwangju                  305       0         252              202
##   Gyeonggi-do              114     252           0              215
##   Gyeongsangbuk-do         179     202         215                0
##   Gyeongsangnam-do         275     159         285              101
##   Incheon                  137     253          24              234
##   Jeju                     500     199         451              365
##   Jeollabuk-do             232      75         178              161
##   Jeollanam-do             325      31         279              207
##   Sejong                   155     155         104              143
##   Seoul                    113     265          13              226
##   Ulsan                    255     229         296               81
##                    dest
## orig                Gyeongsangnam-do Incheon Jeju Jeollabuk-do Jeollanam-do
##   Busan                           43     328  306          201          190
```

```
##   Chungcheongbuk-do              190     111  381          111          205
##   Chungcheongnam-do              222      85  368           96          197
##   Daegu                           72     242  333          143          177
##   Daejeon                        168     134  337           66          162
##   Gangwon-do                     275     137  500          232          325
##   Gwangju                        159     253  199           75           31
##   Gyeonggi-do                    285      24  451          178          279
##   Gyeongsangbuk-do               101     234  365          161          207
##   Gyeongsangnam-do                 0     299  277          160          148
##   Incheon                        299       0  450          180          281
##   Jeju                           277     450    0          275          176
##   Jeollabuk-do                   160     180  275            0          101
##   Jeollanam-do                   148     281  176          101            0
##   Sejong                         190     112  353           79          179
##   Seoul                          298      25  464          191          292
##   Ulsan                           76     314  351          212          222
##                     dest
## orig                Sejong Seoul Ulsan
##   Busan                222   324    47
##   Chungcheongbuk-do     37   108   203
##   Chungcheongnam-do     34    95   250
##   Daegu                141   236    76
##   Daejeon               22   138   198
##   Gangwon-do           155   113   255
##   Gwangju              155   265   229
##   Gyeonggi-do          104    13   296
##   Gyeongsangbuk-do     143   226    81
##   Gyeongsangnam-do     190   298    76
##   Incheon              112    25   314
##   Jeju                 353   464   351
##   Jeollabuk-do          79   191   212
##   Jeollanam-do         179   292   222
##   Sejong                 0   117   216
##   Seoul                117     0   307
##   Ulsan                216   307     0
```

5.3.5 Migration distance

- The `index_distance()` function in the *migest* package provides three summary distance measures given a set of migration and distance measures between each origin and destination.
- The origin-destination migration flows can be given as a matrix to `m` or as a data frame, where the column names are assumed to be `orig`, `dest` and `flow`.
  - Can change using `orig_col`, `dest_col` and `flow_col` arguments

- The distance values can also be given as a matrix to `d` or as a data frame, where the column names are assumed to be `orig`, `dest` and `dist`.
- Origin and destination names in `m` and `dist` must match
- Removes all within migration moves from calculations

```
# single year
index_distance(m = filter(korea_reg, year == 2020),
               d = korea_dist)
```

```
## # A tibble: 3 x 2
##   measure   value
##   <chr>     <dbl>
## 1 mean    105.
## 2 median   68.7
## 3 decay    -0.852
```

5.3.6 Migration distance

```
korea_reg %>%
  nest(m = c(orig, dest, flow)) %>%
  mutate(d = list(korea_dist)) %>%
  mutate(i = map2(.x = m, .y = d,
                  .f = ~index_distance(m = .x, d = .y, long = FALSE))) %>%
  unnest(i)
```

```
## # A tibble: 9 x 6
##    year m                  d                 mean median  decay
##   <int> <list>             <list>           <dbl>  <dbl>  <dbl>
## 1  2012 <tibble [289 x 3]> <dbl [17 x 17]>  108.   76   -0.784
## 2  2013 <tibble [289 x 3]> <dbl [17 x 17]>  107.   76   -0.795
## 3  2014 <tibble [289 x 3]> <dbl [17 x 17]>  108.   76   -0.816
## 4  2015 <tibble [289 x 3]> <dbl [17 x 17]>  108.   76   -0.828
## 5  2016 <tibble [289 x 3]> <dbl [17 x 17]>  107.   76   -0.820
## 6  2017 <tibble [289 x 3]> <dbl [17 x 17]>  108.   75.8 -0.839
## 7  2018 <tibble [289 x 3]> <dbl [17 x 17]>  107.   74.8 -0.839
## 8  2019 <tibble [289 x 3]> <dbl [17 x 17]>  107.   75.8 -0.836
## 9  2020 <tibble [289 x 3]> <dbl [17 x 17]>  105.   68.7 -0.852
```

5.3.7 Calculating distances

- There are a number of functions to a calculate distance matrices in R
- Require a set of longitude and latitudes

- If population weighted centriods are not available from national statistics offices, a number of research centers provide estimates
  - POPGRID Data Collaborative `https://www.popgrid.org/`
- Example for Ghana using WorldPop 2020 population weighted centroids
  - CSV from `https://www.worldpop.org/doi/10.5258/SOTON/WP00703`

```
g <- read_csv("data/PWD_2020_sub_national_100m.csv") %>%
  filter(ISO == "GHA")
g
```

```
## # A tibble: 10 x 25
##     year ISO   ISO_No Country_N Adm_N GID_1 HASC  PWC_Lat PWC_Lon    Pop Density
##    <dbl> <chr>  <dbl> <chr>     <chr> <chr> <chr>   <dbl>   <dbl>  <dbl>   <dbl>
##  1  2020 GHA      288 Ghana     Asha~ GHA.~ GH.AH    6.69  -1.60  6.43e6   258.
##  2  2020 GHA      288 Ghana     Bron~ GHA.~ GH.BA    7.50  -2.04  2.73e6    70.6
##  3  2020 GHA      288 Ghana     Cent~ GHA.~ GH.CP    5.47  -0.986 2.87e6   296.
##  4  2020 GHA      288 Ghana     East~ GHA.~ GH.EP    6.21  -0.517 3.12e6   188.
##  5  2020 GHA      288 Ghana     Grea~ GHA.~ GH.AA    5.64  -0.159 5.18e6  1414.
##  6  2020 GHA      288 Ghana     Nort~ GHA.~ GH.NP    9.48  -0.569 3.25e6    47
##  7  2020 GHA      288 Ghana     Uppe~ GHA.~ GH.UE   10.9   -0.680 1.11e6   129.
##  8  2020 GHA      288 Ghana     Uppe~ GHA.~ GH.UW   10.4   -2.44  8.02e5    42.2
##  9  2020 GHA      288 Ghana     Volta GHA.~ GH.TV    6.90   0.504 2.66e6   144.
## 10  2020 GHA      288 Ghana     West~ GHA.~ GH.WP    5.45  -2.16  2.92e6   118.
## # ... with 14 more variables: Area <dbl>, PWD_A <dbl>, PWD_G <dbl>,
## #   PWD_M <dbl>, PWD_D1 <dbl>, PWD_D2 <dbl>, PWD_D3 <dbl>, PWD_D4 <dbl>,
## #   PWD_D5 <dbl>, PWD_D6 <dbl>, PWD_D7 <dbl>, PWD_D8 <dbl>, PWD_D9 <dbl>,
## #   PWD_D10 <dbl>
```

5.3.8 Calculating distances

```
g <- g %>%
  filter(ISO == "GHA") %>%
  select(Adm_N, PWC_Lon, PWC_Lat)
g
```

```
## # A tibble: 10 x 3
##    Adm_N         PWC_Lon PWC_Lat
##    <chr>           <dbl>   <dbl>
##  1 Ashanti        -1.60     6.69
##  2 Brong Ahafo    -2.04     7.50
##  3 Central        -0.986    5.47
```

```
##  4 Eastern        -0.517    6.21
##  5 Greater Accra  -0.159    5.64
##  6 Northern       -0.569    9.48
##  7 Upper East     -0.680   10.9
##  8 Upper West     -2.44    10.4
##  9 Volta           0.504    6.90
## 10 Western        -2.16     5.45
```

5.3.9 Calculating distances

- The `distm()` function in the *geosphere* package provides great circle distance estimates in meters between centroids

```
library(geosphere)
ghana_dist <- g %>%
  select(PWC_Lon, PWC_Lat) %>%
  distm()

ghana_dist
```

```
##           [,1]     [,2]      [,3]      [,4]      [,5]     [,6]     [,7]
##  [1,]      0.0 102992.7 151088.53 130977.06 197276.08 329699.1 471774.1
##  [2,] 102992.7      0.0 254060.16 221397.05 293464.70 272448.3 399289.5
##  [3,] 151088.5 254060.2      0.00  97220.64  93573.16 446740.6 596619.3
##  [4,] 130977.1 221397.1  97220.64      0.00  74652.44 362233.8 513803.4
##  [5,] 197276.1 293464.7  93573.16  74652.44      0.00 427847.5 579597.5
##  [6,] 329699.1 272448.3 446740.58 362233.78 427847.51      0.0 151789.3
##  [7,] 471774.1 399289.5 596619.34 513803.44 579597.52 151789.3      0.0
##  [8,] 422969.2 325054.7 570615.97 511378.47 585433.81 229629.3 198393.6
##  [9,] 233944.5 289137.3 229081.18 136579.93 158124.96 308917.8 455693.6
## [10,] 149904.5 227534.2 130136.28 200358.81 222698.68 479309.9 619145.3
##           [,8]     [,9]    [,10]
##  [1,] 422969.2 233944.5 149904.5
##  [2,] 325054.7 289137.3 227534.2
##  [3,] 570616.0 229081.2 130136.3
##  [4,] 511378.5 136579.9 200358.8
##  [5,] 585433.8 158125.0 222698.7
##  [6,] 229629.3 308917.8 479309.9
##  [7,] 198393.6 455693.6 619145.3
##  [8,]      0.0 505963.2 550152.0
##  [9,] 505963.2      0.0 335790.5
## [10,] 550152.0 335790.5      0.0
```

5.3.10 Calculating distances

- Add the origin and destination names to the `matrix` row and columns using `dimnames()`
- Allows to combine with migration data
- Divide by 1000 to get to kilometers

```
dimnames(ghana_dist) <- list(orig = g$Adm_N, dest = g$Adm_N)
round(ghana_dist/1000)
```

```
##                dest
## orig            Ashanti Brong Ahafo Central Eastern Greater Accra Northern
##   Ashanti             0         103     151     131           197      330
##   Brong Ahafo       103           0     254     221           293      272
##   Central           151         254       0      97            94      447
##   Eastern           131         221      97       0            75      362
##   Greater Accra     197         293      94      75             0      428
##   Northern          330         272     447     362           428        0
##   Upper East        472         399     597     514           580      152
##   Upper West        423         325     571     511           585      230
##   Volta             234         289     229     137           158      309
##   Western           150         228     130     200           223      479
##                dest
## orig            Upper East Upper West Volta Western
##   Ashanti              472        423   234     150
##   Brong Ahafo          399        325   289     228
##   Central              597        571   229     130
##   Eastern              514        511   137     200
##   Greater Accra        580        585   158     223
##   Northern             152        230   309     479
##   Upper East             0        198   456     619
##   Upper West           198          0   506     550
##   Volta                456        506     0     336
##   Western              619        550   336       0
```

5.4 Connectivity

5.4.1 Migration connectivity

- The size of migration flows in different migration corridors vary due to many reasons other than population sizes and distance.
  - Also reflect the strength of many other factors linking regions such as the strength of historical, cultural, social and economic ties, between regions.

- Bell et al. [2002] note a fragmentation in the literature on measures of connections in a migration system and the use of a range of terms including *spatial connectivity*, *spatial concentration*, *spatial inequality* and *spatial focusing*
- There are many indices on migration connectivity. See the *migration.indices* package for example.
- The `index_connectivity()` function provides 12 different measures, that can broadly be placed into 5 groups.
  - Requires a matrix or data frame of migration flows
  - When providing a data frame, function is assuming flows are in a column with name `flow`; change with `flow_col` if not.

5.4.2 Migration connectivity

```
korea_reg %>%
  filter(year == 2020) %>%
  index_connectivity()
```

```
## # A tibble: 11 x 2
##    name                     value
##    <chr>                    <dbl>
##  1 connectivity            1
##  2 inequality_equal        0.541
##  3 inequality_sim          0.281
##  4 gini_total              0.709
##  5 gini_orig_standardized  0.0493
##  6 gini_dest_standardized  0.0517
##  7 mwg_orig                0.0370
##  8 mwg_dest                0.0389
##  9 mwg_mean                0.0379
## 10 cv                     17.9
## 11 acv                     3.43
```

5.4.3 Connectivity and Inequality

- The `connectivity` measure evaluates the proportion of the flows (excluding within region flows) that are non-zeros
  - More useful when many regions where populations are smaller
- Bell et al. [2002] inequality measures are based on a distributions of flows compared to distributions of expected flows

  - `inequality_equal` measures the distance of the observed flows to an expected distribution where all flows are equal
  - `inequality_sim` measures the distance of the observed flows to an expected distribution from a spatial interaction model equivalent to a Poisson regression mode for an independence fit

$$\widehat{\log(m_{ij})} = \beta_0 + \beta_{1i}\texttt{origin} + \beta_{2j}\texttt{destination}$$

- In both cases, a value of 0 shows the observed flows match the expected values (some form of equality) and 1 shows the maximum distance between the observed flows and the expected flows, i.e. maximum inequality.

5.4.4 Gini measures

- The Gini measures provide a value for the spatial focusing in a set of migration flow - i.e. how much of the migration is focused on a particular set of migration corridors
- Compares each flow with every other flow in the migration matrix.
- A `gini_total` value of zero indicates all flows are of equal size (no spatial focusing) to 1, only one single flow (maximum focusing).
- The `gini_orig_standardized` values provide a similar measure but compare every outflow from each origin with every other outflow from that origin.
  - Measures the extent to which the destination choices of out-migrants are spatially focused.
  - The `gini_dest_standardized` does the same but for the spatial focusing of origins of in-migrants.
  - The standardized values ensure a range of 0 and 1 - zero is no focusing

5.4.5 Migration totals Gini and Coefficient of Variation

- The migration weighted Gini indexes provide a measure of the focusing for the in-migration and out-migration totals (`mwg_orig` and `mwg_dest`)
- The `mwg_mean` is a simple average of `mwg_orig` and `mwg_dest` to provide a system wide measure of focusing for all migration totals.
  - As with the `gini_` measures from `index_connectivity()` values vary between zero (no focusing) and 1 (all migration goes through a single origin or destination)
- Rogers and Raymer [1998] proposed a coefficient of variation, provided by `cv` which compares the mean of the flows to the standard deviation of the flows.
  - Is not limited to 0 and 1

- The `acv` provides a similar measures of variation but based on the aggregate of coefficient of variations of in- and out-migration totals (based on the means and standard deviations of in- and out-migration totals)
  - Again, not limited to 0 and 1, but useful for comparisons across time or countries, where rising `cv` or `acv` would indicate greater inequality in migration flows or flow totals

5.5 Impact

5.5.1 Migration impact

- The impact of migration measures the extent to which migration acts to transform the pattern of human settlement
- Migration is already or becoming predominant mechanism leading to the redistribution of population in many regions of the world
- Descriptive studies tend to focus on regional net migration patterns
- Additional measures exist that summarize the overall effect of migration in redistributing a population across the entire system of regions

5.5.2 Migration effectiveness index

- The migration effectiveness index (MEI) of Shryock and Siegel [1976] compares the sum of net migration as a proportion of migration turnover, measuring the amount asymmetry or equilibrium in the migration network

$$\texttt{MEI} = 100\frac{\sum_i |\texttt{net}|}{\sum_i \texttt{turnover}_i} = 100\frac{\sum_i |m_{+i} - m_{i+}|}{\sum_i m_{+i} + m_{i+}}$$

- MEI range between 0 and 100.
- High values indicate migration is an efficient mechanism of population redistribution, based on large net totals for the given turnover.
- Values closer to zero are generated from more balanced migration systems with less population redistribution

5.5.3 Aggregate net migration rate

- The aggregate net migration rate (ANMR) of Bell et al. [2002] attempts to measure the overall effect of migration on the population settlement patterns by replacing the denominator of MEI with the each regions population P_i
- Index measures the net shift of population between regions per 100 residents in the country

  - No upper limit

$$\texttt{ANMR} = 100\frac{1}{2}\frac{\sum_i |\texttt{net}_i|}{\sum_i P_i} = 100\frac{1}{2}\frac{\sum_i |m_{+i} - m_{i+}|}{\sum_i P_i}$$

- Product of the CMI and MEI

5.5.4 Preference and velocity

- The manual by United Nations Department of Economic and Social Affairs Population Division [1983] provides two other impact measures, that seem to have fallen out of favor
- The preference index is based on an expected model of migration intestines based on population shares and the overall level of migration $M\frac{p_i}{P}\frac{p_j}{P}$, where M is the total migration flow and P is the total population based on the sum of populations in each region (p_i)
- Index compares the observed flows to an expected model where migration rates in all populations are the same
  - No upper limit

$$\texttt{preference} = \sum_{ij} \frac{m_{ij}}{M\frac{p_i}{P}\frac{p_j}{P}}$$

- The velocity index is based on a migration velocity measure $\frac{m_{ij}}{p_i p_j}$, multiplied by the total population and summed
- Index compares observed flows to an expected models where flows sizes are determined by population sizes alone
  - No upper limit

$$\texttt{velocity} = \sum_{ij} \frac{m_{ij}}{p_i p_j} P$$

5.5.5 Migration impact

- The `index_impact()` function in *migest* calculates all four measures given a set of migration flows and population sizes in each region
  - The `p` parameters assumes column names `region` and `pop` for region and population. Change from defaults using `reg_col` and `pop_col`

```
index_impact(
  m = subset(korea_reg, year == 2020),
  p = subset(korea_pop, year == 2020),
  pop_col = "population"
)
```

```
## # A tibble: 4 x 2
##   measure        value
##   <chr>          <dbl>
## 1 effectivness   7.67
## 2 anmr           0.375
## 3 preference   375.
## 4 velocity      18.3
```

5.5.6 Migration impact

- Multiple years require nesting the migration and population data bases besides each other

```
d <- korea_reg %>%
  nest(m = c(orig, dest, flow)) %>%
  left_join(korea_pop) %>%
  nest(p = c(region, population))
```

```
## Joining, by = "year"
```

```
d
```

```
## # A tibble: 9 x 3
##    year m                  p
##   <int> <list>             <list>
## 1  2012 <tibble [289 x 3]> <tibble [17 x 2]>
## 2  2013 <tibble [289 x 3]> <tibble [17 x 2]>
## 3  2014 <tibble [289 x 3]> <tibble [17 x 2]>
## 4  2015 <tibble [289 x 3]> <tibble [17 x 2]>
## 5  2016 <tibble [289 x 3]> <tibble [17 x 2]>
## 6  2017 <tibble [289 x 3]> <tibble [17 x 2]>
## 7  2018 <tibble [289 x 3]> <tibble [17 x 2]>
## 8  2019 <tibble [289 x 3]> <tibble [17 x 2]>
## 9  2020 <tibble [289 x 3]> <tibble [17 x 2]>
```

5.5.7 Migration impact

- Apply the `index_impact()` function to each row and unnest

```
d %>%
  mutate(i = map2(.x = m, .y = p,
                  .f = ~index_impact(m = .x, p = .y,
```

```
                                    pop_col = "population",
                                    long = FALSE))) %>%
  unnest(i)
```

```
## # A tibble: 9 x 7
##    year m                  p                  effectivness  anmr preference velocity
##   <int> <list>             <list>                    <dbl> <dbl>      <dbl>    <dbl>
## 1  2012 <tibble [289 x 3]> <tibble [17 x 2]>          6.07 0.300       409.     20.2
## 2  2013 <tibble [289 x 3]> <tibble [17 x 2]>          5.72 0.271       371.     17.6
## 3  2014 <tibble [289 x 3]> <tibble [17 x 2]>          5.36 0.262       434.     21.2
## 4  2015 <tibble [289 x 3]> <tibble [17 x 2]>          7.73 0.383       459.     22.7
## 5  2016 <tibble [289 x 3]> <tibble [17 x 2]>          8.47 0.402       401.     19.0
## 6  2017 <tibble [289 x 3]> <tibble [17 x 2]>          7.99 0.372       417.     19.4
## 7  2018 <tibble [289 x 3]> <tibble [17 x 2]>          9.29 0.435       398.     18.7
## 8  2019 <tibble [289 x 3]> <tibble [17 x 2]>          6.94 0.319       391.     18.0
## 9  2020 <tibble [289 x 3]> <tibble [17 x 2]>          7.67 0.375       375.     18.3
```

5.5.8 Exercise (ex3.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(migest)
library(geosphere)
##
##
##
# 1. Run the code below to read in the bilateral data in brazil_census_tidy.csv
#    from the 1991, 2000 and 2010 Brazilian censuses
br <- read_csv("./data/brazil_census_tidy.csv")
br
# 2. Run the code below to read in the WorldPop population data for Brazil in
#    2000 and check that the orig and dest names in the br migration data match
#    the region names in br_pop
br_pop <- read_csv("./data/PWD_2000_sub_national_100m.csv",
                   locale = readr::locale(encoding = "latin1")) %>%
  filter(ISO == "BRA") %>%
  select(Adm_N, contains("PWC"), Pop)
br_pop
# check names match
```

```
unique(br$orig) %in% br_pop$Adm_N
unique(br$dest) %in% br_pop$Adm_N
# 3. Calculate the migration intensity indices for Brazil in 2000
m <- br %>%
  #####(year == 2000) %>%
  pull(flow) %>%
  sum()
m
p <- br_pop %>%
  pull(Pop) %>%
  #####()
#####(mig_total = #####,
                pop_total = p,
                n = n_distinct(br_pop$#####))
# 3. Calculate the migration connectivity indices for Brazil in 1991
br %>%
  filter(year == 1991) %>%
  #####()
# 4. Create a distance matrix using the population weighted centrods in br_pop
br_dist <- br_pop %>%
  select(PWC_Lon, #####) %>%
  #####()
br_dist
# 5. Adapt the br_dist matrix object to
#    a. Include the relevant row and column names
#    b. Scaled to kilometers
#####(br_dist) <- list(orig = br_pop$Adm_N, dest = br_pop$Adm_N)
br_dist <- round(#####/1000)
# 6. Calculate the migration distance indices for Brazil in 200, using the
#    br_dist object
br %>%
  filter(year == 2000) %>%
  #####(d = br_dist)
```

Chapter 6

Estimating Net Migration

6.1 Net Migration

6.1.1 Net migration

- At the most basic level demographers are typically interested in the net balance of migration as a component of population change.
- Might not have an interest in the complexities involved in the different scales of migration to and from each region.
- Net migration tends to be used as it is readily available.
  - Data for in- and out-migration require specialized migration question in surveys or censuses.
  - Net migration does not require any questions on migration.
- Most censuses measure population changes accurately enough in order to develop a good estimate of net migration.
- Net migration has many weaknesses for the study of migration patterns, migration trends and population projections Rogers [1990]
  - Net migrants do not exist

6.1.2 Net migration estimation

- Three groups of methods to derive net migration
- First two are residual methods
  1. Vital statistics based on population change and natural increase data
  2. Survival methods based on population change data
  3. Place of birth methods based on changes in migrant stock data.

- United Nations Department of Economic and Social Affairs Population Division [1983] provides a nice discussion on the relative merits of each method

6.2 Vital Statstics

6.2.1 Vital statsitics

- The most elementary method to estimate net migration is using the demographic accounting equation

$$M = P^{t+n} - P^t - B + D$$

- Simple to calculate.
- Careful data preparation is required.
- Commonly applied to estimate net migration by sub-groups of populations where (e.g. sex)
- Less commonly applied to estimate net migration by age

$$M(x) = P^{t+n}(x+n) - P^t(x) + D(x)$$

  where parenthesis represent age groups x of size n
- Not easy to accurately estimate age-specific death counts that align to period between censuses

6.2.2 Vital statsitics

- The *migest* package has a `net_vs()` function to help obtain net migration estimates using vital statistics.
- Demonstrate using the `alabama_1970` data set in migest
  - Births are given in the under 10 age groups for `pop_1960`

```
library(tidyverse)
library(migest)
alabama_1970
```

```
## # A tibble: 68 x 6
##    age_1970 sex    race  pop_1960 pop_1970 us_census_sr
##    <fct>    <chr>  <chr>    <dbl>    <dbl>        <dbl>
##  1 0-4      female white   104556   100224        0.965
##  2 5-9      female white   119478   115269        0.956
##  3 10-14    female white   120463   121922        0.997
##  4 15-19    female white   114627   115128        1.01
```

```
##  5 20-24    female white   113551   107480        0.998
##  6 25-29    female white    93665    87706        0.989
##  7 30-34    female white    76348    77285        0.996
##  8 35-39    female white    74278    75115        0.994
##  9 40-44    female white    79572    78924        0.989
## 10 45-49    female white    80719    78284        0.968
## # ... with 58 more rows
```

6.2.3 Vital statsitics

- Obtain race and sex population totals
- Need to remove those not born in the original population `pop_1960`.

```
d <- alabama_1970 %>%
  group_by(race, sex) %>%
  summarise(births = sum(pop_1960[1:2]),
            pop_1960 = sum(pop_1960) - births,
            pop_1970 = sum(pop_1970)) %>%
  ungroup()
```

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

```
d
```

```
## # A tibble: 4 x 5
##   race      sex    births pop_1960 pop_1970
##   <chr>     <chr>   <dbl>    <dbl>    <dbl>
## 1 non-white female 126886   515483   483882
## 2 non-white male   131767   467648   426452
## 3 white     female 224034  1159548  1298342
## 4 white     male   236481  1124061  1235489
```

6.2.4 Vital statsitics

- Given the vital statistics `net_vs()` estimate net migration and returns three additional columns
  - `pop_change` for the population difference
  - `natural_inc` for the difference in births and deaths
  - `net` for the net migration based on the two previous columns
- The `net_vs()` function assumes `births_col = "births"` and `deaths_col = "deaths"`.

- Can alter from default if not the case

```
d %>%
  mutate(deaths = c(51449, 58845, 86880, 123220)) %>%
  net_vs(pop0_col = "pop_1960", pop1_col = "pop_1970")
```

```
## # A tibble: 4 x 9
##   race      sex    births pop_1960 pop_1970 deaths pop_change natural_inc     net
##   <chr>     <chr>   <dbl>    <dbl>    <dbl>  <dbl>      <dbl>       <dbl>   <dbl>
## 1 non-white female 126886   515483   483882  51449     -31601       75437 -107038
## 2 non-white male   131767   467648   426452  58845     -41196       72922 -114118
## 3 white     female 224034  1159548  1298342  86880     138794      137154    1640
## 4 white     male   236481  1124061  1235489 123220     111428      113261   -1833
```

6.2.5 Difficulties

- Strictly speaking should refer to net migration estimates as a mixture of net migration and net balance of errors from the other data sources
- Assumes international migration is nil or negligible.
- Bogue et al. [1982] list six difficulties with the vital statistics methods, most of which are due to the estimate is a residual from the combination of other data sources

1. Requires a stable administrative geography, where regions or countries do not change or at least enumerate population, births and deaths for the same units throughout the interval.

2. Adjustments will be required if there has been a big change in the method to collect census, for example switching from *de jure* to *de facto* for defining place of residence

3. Adjustments will be required if the birth and death periods do not align with the census dates. Typically vital statistics are annual measures starting from 1st January where as census dates are not usually on 1st January.

6.2.6 Difficulties

4. Births need to be tabulated or adjusted to mothers place of residence and deaths need to be tabulated or adjusted to place of residence or deceased. If place of occurrence is used for either then additional potential for error is created

5. Births and deaths need to be corrected for under-registration if it is known to exist.

6. Adjustments might be required to include/exclude population groups such as military or students - depending on how each are counted in the censuses and vital statistics registrations.

6.3 Survival Methods

6.3.1 Survival methods

- Survival ratios can be used to compute mortality over the period, to then determine net migration as a residual.
- Survival ratios are an estimate of what proportion of a hypothetically closed population would be present at the end of the period.
  - Survival measures the force of mortality, rather than an overall population change
- Methods can be applied to total population or age-specific populations
- Preferred for age-specific net migration estimates as does not require age-specific death counts.
- Three related approaches using:
  - Forward survival ratios
  - Reverse survival ratios
  - Average survival ratios

6.3.2 Forward survival ratios

- Difference between the surviving expected population and observed population at the end of the period is an estimate of net migration during the interval

$$M'(x) = P^{t+n}(x+n) - s(x)P^t(x)$$

  where:
  - $M'(x)$ is the net migration for between t and $t+n$ for age group x
  - $P^{t+n}(x+n)$ is the observed population at the end of the period $(t+n)$ for age group x
  - $s(x)$ is survival rate between t and $t+1$ for age group x
  - $P^t(x)$ is the observed population at the start of the period (t) for age group x

6.3.3 Reverse survival ratios

- An alternative method is based on the reverse of the previous method

- Estimate the number of persons that would have been x years of age at the earlier census from the number who are enumerated as $x+n$ years old in the second census by applying *reverse survival ratios*

$$M''(x) = \frac{1}{s(x)} P^{t+n}(x+n) - P^t(x)$$

6.3.4 Average survival ratios

- The average survival ratios averages the net migration estimates form the forward and reverse survival ratios

$$\bar{M}(x) = \frac{1}{2}(M'(x) + M''(x))$$

- Siegel and Hamilton [1952] found the average survival ratio method provides the most exact approximation under normal circumstances
- Summary of assumptions for deaths:
  - Forward method: all deaths of migrants are not counted as migrants, equivalent to assuming that they all died at the place of origin.
  - Reverse method: the opposite is assumed. All migrants that die are counted as migrants, as are as those that would have moved had they survived the interval.
  - Average method: only those that died after moving are counted as migrants (approximately).

6.3.5 Survival methods in R

- The *migest* package contains the `net_sr()` function to calculate all three survival ratio estimates of net migration.
- Demonstrate using the `bombay_1951` data
  - Survival ratios come from a UN model life table

```
bombay_1951
```

```
## # A tibble: 13 x 5
##    age_1941 age_1951 pop_1941 pop_1951    sr
##    <fct>    <fct>       <dbl>    <dbl> <dbl>
##  1 0-4      10-14       77135   132870 0.909
##  2 5-9      15-19       85434   170227 0.957
```

```
##  3 10-14    20-24       79185   263971 0.947
##  4 15-19    25-29       82603   253964 0.931
##  5 20-24    30-34      126247   195373 0.922
##  6 25-29    35-39      155344   151259 0.916
##  7 30-34    40-44      138843   118383 0.905
##  8 35-39    45-49      109356    76421 0.885
##  9 40-44    50-54       81626    65897 0.855
## 10 45-49    55-59       47062    32265 0.812
## 11 50-54    60-64       36908    22248 0.754
## 12 55-59    65-69       15134     9655 0.673
## 13 60+      70+         25094    10100 0.387
```

6.3.6 Survival methods in R

```
net_sr(bombay_1951, pop0_col = "pop_1941", pop1_col = "pop_1951")
```

```
## # A tibble: 13 x 10
##    age_1941 age_1951 pop_1941 pop_1951    sr net_forward net_reverse net_average
##    <fct>    <fct>       <dbl>    <dbl> <dbl>       <dbl>       <dbl>       <dbl>
##  1 0-4      10-14       77135   132870 0.909      62777.      69085.      65931.
##  2 5-9      15-19       85434   170227 0.957      88441.      92386.      90413.
##  3 10-14    20-24       79185   263971 0.947     188975.     199530.     194252.
##  4 15-19    25-29       82603   253964 0.931     177077.     190242.     183659.
##  5 20-24    30-34      126247   195373 0.922      78935.      85585.      82260.
##  6 25-29    35-39      155344   151259 0.916       8948.       9768.       9358.
##  7 30-34    40-44      138843   118383 0.905      -7228.      -7990.      -7609.
##  8 35-39    45-49      109356    76421 0.885     -20359.     -23005.     -21682.
##  9 40-44    50-54       81626    65897 0.855      -3877.      -4535.      -4206.
## 10 45-49    55-59       47062    32265 0.812      -5959.      -7337.      -6648.
## 11 50-54    60-64       36908    22248 0.754      -5562.      -7382.      -6472.
## 12 55-59    65-69       15134     9655 0.673       -524.       -779.       -652.
## 13 60+      70+         25094    10100 0.387        399.       1031.        715.
## # ... with 2 more variables: pop1_forward <dbl>, pop0_reverse <dbl>
```

6.3.7 Survival methods in R

- Second example using `manila_1970` where survivor ratios come from census life tables for all of the Philippines
- Births and survival rates of children are unknown

```
manila_1970
```

```
## # A tibble: 16 x 4
##    age_1970 pop_1960 pop_1970 phl_census_sr
##    <fct>       <dbl>    <dbl>         <dbl>
##  1 0-4            NA    85870        NA
##  2 5-9            NA    83054        NA
##  3 10-14       80275    79489         1.12
##  4 15-19       70875   101410         0.992
##  5 20-24       63250    90410         0.973
##  6 25-29       85618    56055         0.889
##  7 30-34       75793    44648         0.841
##  8 35-39       60037    36963         0.957
##  9 40-44       34813    28873         0.951
## 10 45-49       31927    23678         0.904
## 11 50-54       24297    19063         0.930
## 12 55-59       20207    14484         0.797
## 13 60-64       13714    10205         0.877
## 14 65-69        9366     6405         0.835
## 15 70-74        7921     3746         0.712
## 16 75+         11114     4779         0.562
```

6.3.8 Survival methods in R

- Estimate age-specific net migration for all ages, except children

```
net_sr(manila_1970, pop0_col = "pop_1960", pop1_col = "pop_1970",
       survival_ratio_col = "phl_census_sr")
```

```
## # A tibble: 16 x 9
##    age_1970 pop_1960 pop_1970 phl_census_sr net_forward net_reverse net_average
##    <fct>       <dbl>    <dbl>         <dbl>       <dbl>       <dbl>       <dbl>
##  1 0-4            NA    85870        NA              0           0           0
##  2 5-9            NA    83054        NA              0           0           0
##  3 10-14       80275    79489         1.12      -10196.      -9126.      -9661.
##  4 15-19       70875   101410         0.992      31134.      31400.      31267.
##  5 20-24       63250    90410         0.973      28877.      29683.      29280.
##  6 25-29       85618    56055         0.889     -20082.     -22582.     -21332.
##  7 30-34       75793    44648         0.841     -19117.     -22723.     -20920.
##  8 35-39       60037    36963         0.957     -20497.     -21416.     -20957.
##  9 40-44       34813    28873         0.951      -4244.      -4462.      -4353.
## 10 45-49       31927    23678         0.904      -5189.      -5739.      -5464.
## 11 50-54       24297    19063         0.930      -3521.      -3788.      -3655.
## 12 55-59       20207    14484         0.797      -1613.      -2025.      -1819.
## 13 60-64       13714    10205         0.877      -1822.      -2078.      -1950.
## 14 65-69        9366     6405         0.835      -1417.      -1697.      -1557.
```

```
## 15 70-74        7921     3746         0.712      -1890.      -2657.      -2274.
## 16 75+         11114     4779         0.562      -1472.      -2617.      -2045.
## # ... with 2 more variables: pop1_forward <dbl>, pop0_reverse <dbl>
```

6.3.9 Survival methods in R

- Estimate children net migration setting `net_children = TRUE`.
- Uses method of Shryock and Siegel [1976, p381]
  - Age 0-4: 1/4 (ratio of 0-4 population to 15-44 female population) times net migration for females aged 15-44
  - Age 5-9: 3/4 (ratio of 5-9 population to 20-49 female population) times net migration for females aged 20-49.
- Can alter weights in `maternal_exposure` argument
  - default is `c(0.25, 0.75)`

```
net_sr(manila_1970, pop0_col = "pop_1960", pop1_col = "pop_1970",
       survival_ratio_col = "phl_census_sr", net_children = TRUE)
```

```
## # A tibble: 16 x 9
##    age_1970 pop_1960 pop_1970 phl_census_sr net_forward net_reverse net_average
##    <fct>       <dbl>    <dbl>         <dbl>       <dbl>       <dbl>       <dbl>
##  1 0-4            NA    85870        NA           -235.       -605.       -420.
##  2 5-9            NA    83054        NA          -8935.     -10486.      -9710.
##  3 10-14       80275    79489         1.12      -10196.      -9126.      -9661.
##  4 15-19       70875   101410         0.992      31134.      31400.      31267.
##  5 20-24       63250    90410         0.973      28877.      29683.      29280.
##  6 25-29       85618    56055         0.889     -20082.     -22582.     -21332.
##  7 30-34       75793    44648         0.841     -19117.     -22723.     -20920.
##  8 35-39       60037    36963         0.957     -20497.     -21416.     -20957.
##  9 40-44       34813    28873         0.951      -4244.      -4462.      -4353.
## 10 45-49       31927    23678         0.904      -5189.      -5739.      -5464.
## 11 50-54       24297    19063         0.930      -3521.      -3788.      -3655.
## 12 55-59       20207    14484         0.797      -1613.      -2025.      -1819.
## 13 60-64       13714    10205         0.877      -1822.      -2078.      -1950.
## 14 65-69        9366     6405         0.835      -1417.      -1697.      -1557.
## 15 70-74        7921     3746         0.712      -1890.      -2657.      -2274.
## 16 75+         11114     4779         0.562      -1472.      -2617.      -2045.
## # ... with 2 more variables: pop1_forward <dbl>, pop0_reverse <dbl>
```

6.3.10 Survival ratios

- The success of the above methods depend on the survival ratios.

- Ratios can be obtained from
  - Life table survival ratios (LTSR) as in `bombay_1951` example
  - Census survival ratios (CSR) as in `manila_1970` example
- Life table survival ratios are derived from the L_x columns of the life table; the ratio of persons in stationary population at age group x that are alive in comparisons to a previous age group $x - n$.

$$s_n(x) = \frac{L(x+n)}{L(x)}$$

- Can also be derived from mortality rates, if known.

6.3.11 Life table survival ratios

- For an accurate net migration estimate, $s_n(x)$ should
  - Measure the average mortality conditions of the period
  - Reasonably applicable to the area and population for which migration estimates are required.
- Age data to derive life tables may be inaccurate. Errors will impact the net migration estimates.

6.3.12 Census survival ratios

- Where appropriate life tables are not available or not appropriate, survival ratios can be computed from census age distributions
- A census survival ratio (CSR) is the ratio of the population aged $x + n$ at a given census to the population aged x at the census n years earlier.
- Computed for a nation as a whole assuming a "closed" population.
  - Adjust data for international migration before calculating CSR if international migration is a influential part of population change for a given area or group.

6.3.13 Age-specfic birthplace data

- Example to derive the birthplace-age-specific survival ratios from the 1950 and 1960 census data, given in `usa_1960`

```
usa_1960
```

```
## # A tibble: 288 x 7
##    birthplace  race      sex    age_1950 age_1960 pop_1950 pop_1960
##    <fct>       <fct>     <fct>  <fct>    <fct>       <dbl>    <dbl>
##  1 New England white     male   0-4      10-14      465097   467291
##  2 New England white     female 0-4      10-14      445100   450248
##  3 New England non-white male   0-4      10-14        8419     8927
##  4 New England non-white female 0-4      10-14        8205     8896
##  5 New England white     male   5-9      15-19      378265   368524
##  6 New England white     female 5-9      15-19      361845   359141
##  7 New England non-white male   5-9      15-19        5421     5475
##  8 New England non-white female 5-9      15-19        5501     5977
##  9 New England white     male   10-19    20-29      606335   567349
## 10 New England white     female 10-19    20-29      591111   582993
## # ... with 278 more rows
```

6.3.14 Age-specfic birthplace data

- Focus on white males for example later on

```
s <- usa_1960 %>%
  filter(sex == "male",
         race == "white") %>%
  mutate(sr = pop_1960/pop_1950) %>%
  select(-contains("pop"))
s
```

```
## # A tibble: 72 x 6
##    birthplace      race  sex   age_1950 age_1960    sr
##    <fct>           <fct> <fct> <fct>    <fct>    <dbl>
##  1 New England     white male  0-4      10-14    1.00
##  2 New England     white male  5-9      15-19    0.974
##  3 New England     white male  10-19    20-29    0.936
##  4 New England     white male  20-29    30-39    1.00
##  5 New England     white male  30-39    40-49    0.996
##  6 New England     white male  40-49    50-59    0.946
##  7 New England     white male  50-59    60-69    0.825
##  8 New England     white male  60+      70+      0.488
##  9 Middle Atlantic white male  0-4      10-14    1.01
## 10 Middle Atlantic white male  5-9      15-19    0.975
## # ... with 62 more rows
```

6.3.15 Census survival ratios

- The CSR method tends to correct for systematic errors in the age data.

  - For example, get $s_n(x)$ in adolescent years greater than one as larger under-registration in 0-4 compared to 5-9 or 10-14 age groups.

- Systematic errors in the censuses might lead to survivor ratios to incorporate net census errors, that might lead to better estimate of net migration compared to LTSR.
- CSR tend to be less smooth than LTSR,
  - Perhaps more realistic age-patterns of net migration.

6.3.16 Limitations of census survival ratios

A number of weaknesses for CSR

- Assumes a closed population, so data must be adjusted for international migration before calculating CSR.
  - Good data on international migration data not always available
- Mortality may vary greatly in each region, so using a CSR based on national level data not always appropriate.
  - Build in regional correction factors
- Census enumeration may vary greatly in each region.
  - Build in regional correction factors
- A single census can not provide CSR for children.
  - Use birth statistics to approximate new born population for CSR calculation
  - If birth statistics are not reliable, use an approximation method using the ratio of women to children and female estimated net migration

6.4 Birthplace

6.4.1 Birthplace

- If data on lifetime migration at the start and end of the period are available, net migration can be estimated for each migrant group.
- Different procedure can be applied, depending on the availability of data
  1. Lifetime migration totals without age characteristics
  2. Lifetime migration data with age characteristics
- Both rely on a survival approach
  - Survival ratios are calculated by birthplace (and possibly other factors)

- If you view birthplace as just another dimension (such as sex) then the method is near identical to the survival ratio methods.
  - Can use the `net_sr()` function in migest once data is in correct format

6.4.2 Birthplace totals

- To demonstrate arranging birthplace totals with no age dimension and the application of `net_sv()` we use the `indian_sub` data in the *migest* package.

```
indian_sub
```

```
## # A tibble: 164 x 7
##    zone             state      sex    year in_migrants out_migrants net_migrants
##    <chr>            <chr>      <chr> <int>       <dbl>        <dbl>        <dbl>
##  1 United Provinces United Pr~ male   1901      259836       878864      -619028
##  2 East Zone        East Zone  male   1901      883052       529216       353836
##  3 East Zone        Bihar-Ori~ male   1901      466126       498082       -31956
##  4 East Zone        Assam      male   1901      416926        31134       385792
##  5 Burma            Burma      male   1901      352924         4489       348435
##  6 South Zone       South Zone male   1901      347416       509163      -161747
##  7 South Zone       Madras     male   1901      115290       450068      -334778
##  8 South Zone       Travancor~ male   1901       42927         8515        34412
##  9 South Zone       Mysore     male   1901      189199        50580       138619
## 10 Bombay           Bombay     male   1901      311720       248149        63571
## # ... with 154 more rows
```

6.4.3 Birthplace totals

- Separate columns for populations depending on birthplace
  - In state of birth or out of the state of birth.
- Rearrange data using `pivot_longer()` and `pivot_wider()` in the *tidyr* package
  - Location in its own column
  - Populations in each year in own columns
  - Work with male populations between 1921 and 1931 for those born in four selected states
  - Drop `net_migrants`, `sex` and `zone` columns

```
d <- indian_sub %>%
  filter(between(year, 1921, 1931),
                 sex == "male",
                 state %in% c("Assam", "Madras", "Mysore", "Bombay")) %>%
  select(-net_migrants, -zone, -sex) %>%
  pivot_longer(cols = contains("migrants"), names_to =  "location",
               values_to = "pop") %>%
  rename(birthplace = state)
```

6.4.4 Birthplace totals

```
d
```

```
## # A tibble: 16 x 4
##    birthplace  year location        pop
##    <chr>      <int> <chr>         <dbl>
##  1 Assam       1921 in_migrants  671195
##  2 Assam       1921 out_migrants  44136
##  3 Madras      1921 in_migrants   97105
##  4 Madras      1921 out_migrants 580136
##  5 Mysore      1921 in_migrants  187000
##  6 Mysore      1921 out_migrants  45349
##  7 Bombay      1921 in_migrants  474553
##  8 Bombay      1921 out_migrants 197593
##  9 Assam       1931 in_migrants  754821
## 10 Assam       1931 out_migrants  41785
## 11 Madras      1931 in_migrants  119621
## 12 Madras      1931 out_migrants 723755
## 13 Mysore      1931 in_migrants  204260
## 14 Mysore      1931 out_migrants  54410
## 15 Bombay      1931 in_migrants  480557
## 16 Bombay      1931 out_migrants 202197
```

6.4.5 Birthplace totals

```
d <- d %>%
  mutate(location = case_when(
    location == "in_migrants" ~ "in-state",
    location == "out_migrants" ~ "out-of-state"
  )) %>%
```

```
  pivot_wider(names_from = year, values_from = pop, names_prefix = "pop_")
d
```

```
## # A tibble: 8 x 4
##   birthplace location     pop_1921 pop_1931
##   <chr>      <chr>           <dbl>    <dbl>
## 1 Assam      in-state       671195   754821
## 2 Assam      out-of-state    44136    41785
## 3 Madras     in-state        97105   119621
## 4 Madras     out-of-state   580136   723755
## 5 Mysore     in-state       187000   204260
## 6 Mysore     out-of-state    45349    54410
## 7 Bombay     in-state       474553   480557
## 8 Bombay     out-of-state   197593   202197
```

6.4.6 Birthplace totals

- Can now apply survival ratios to estimate net migration over a period
- Use a censuses survival ratio of 0.81 for both in migrants and out migrants

```
d <- d %>%
  mutate(sr = 0.81) %>%
  net_sr(pop0_col = "pop_1921", pop1_col = "pop_1931")
d
```

```
## # A tibble: 8 x 10
##   birthplace location     pop_1921 pop_1931    sr net_forward net_reverse
##   <chr>      <chr>           <dbl>    <dbl> <dbl>       <dbl>       <dbl>
## 1 Assam      in-state       671195   754821  0.81     211153.     260683.
## 2 Assam      out-of-state    44136    41785  0.81       6035.       7450.
## 3 Madras     in-state        97105   119621  0.81      40966.      50575.
## 4 Madras     out-of-state   580136   723755  0.81     253845.     313389.
## 5 Mysore     in-state       187000   204260  0.81      52790       65173.
## 6 Mysore     out-of-state    45349    54410  0.81      17677.      21824.
## 7 Bombay     in-state       474553   480557  0.81      96169.     118727.
## 8 Bombay     out-of-state   197593   202197  0.81      42147.      52033.
## # ... with 3 more variables: net_average <dbl>, pop1_forward <dbl>,
## #   pop0_reverse <dbl>
```

6.4.7 Birthplace totals

- To derive the net migration flow estimate for each of the states we need to make one more step

  - Subtract the net migration for the out-of-state migrants from the net migration for the in-state migrants

```
d %>%
  group_by(birthplace) %>%
  summarise(net = net_forward[location == "in-state"] -
              net_forward[location == "out-of-state"])
```

```
## # A tibble: 4 x 2
##   birthplace      net
##   <chr>         <dbl>
## 1 Assam       205118.
## 2 Bombay       54022.
## 3 Madras     -212879.
## 4 Mysore       35113.
```

6.4.8 Age-specfic birthplace data

- To demonstrate arranging age-specific birthplace data and the application of `net_sv()` we use the `new_england_1960` data in the *migest* package.
  - New England population totals by birthplace for white males.

```
new_england_1960
```

```
## # A tibble: 72 x 4
##    birthplace         age_1960 pop_1950 pop_1960
##    <fct>              <fct>       <dbl>    <dbl>
##  1 New England        10-14      442577   417069
##  2 Middle Atlantic    10-14        7651    17077
##  3 East North Central 10-14        1831     4376
##  4 West North Central 10-14         719     1313
##  5 South Atlantic     10-14        3451     5578
##  6 East South Central 10-14         679      960
##  7 West South Central 10-14         830     1413
##  8 Mountain States    10-14         533      819
##  9 Pacific            10-14        1730     2687
## 10 New England        15-19      354131   314048
## # ... with 62 more rows
```

6.4.9 Age-specfic birthplace data

- Apply the age-sex-race-birthplace specific census suruviorshp rate based on the US census (see previous CSR slide)

```
d <- new_england_1960 %>%
  left_join(s)
```

```
## Joining, by = c("birthplace", "age_1960")
```

```
d
```

```
## # A tibble: 72 x 8
##    birthplace         age_1960 pop_1950 pop_1960 race  sex   age_1950    sr
##    <fct>              <fct>       <dbl>    <dbl> <fct> <fct> <fct>    <dbl>
##  1 New England        10-14      442577   417069 white male  0-4      1.00
##  2 Middle Atlantic    10-14        7651    17077 white male  0-4      1.01
##  3 East North Central 10-14        1831     4376 white male  0-4      1.01
##  4 West North Central 10-14         719     1313 white male  0-4      1.00
##  5 South Atlantic     10-14        3451     5578 white male  0-4      1.01
##  6 East South Central 10-14         679      960 white male  0-4      1.01
##  7 West South Central 10-14         830     1413 white male  0-4      1.02
##  8 Mountain States    10-14         533      819 white male  0-4      1.02
##  9 Pacific            10-14        1730     2687 white male  0-4      1.01
## 10 New England        15-19      354131   314048 white male  5-9      0.974
## # ... with 62 more rows
```

6.4.10 Age-specfic birthplace data

- Use the national age-sex-race-birthplace CSR to estimate net migration by birthplace and age in New England for white males

```
d %>%
  net_sr(pop0_col = "pop_1950", pop1_col = "pop_1960") %>%
  relocate(contains("net"))
```

```
## # A tibble: 72 x 13
##    net_forward net_reverse net_average birthplace     age_1960 pop_1950 pop_1960
##          <dbl>       <dbl>       <dbl> <fct>          <fct>       <dbl>    <dbl>
##  1     -27596.     -27466.     -27531. New England    10-14      442577   417069
##  2       9333.       9222.       9278. Middle Atlant~ 10-14        7651    17077
##  3       2531.       2511.       2521. East North Ce~ 10-14        1831     4376
##  4        594.        593.        593. West North Ce~ 10-14         719     1313
##  5       2086.       2062.       2074. South Atlantic 10-14        3451     5578
##  6        271.        267.        269. East South Ce~ 10-14         679      960
##  7        567.        556.        562. West South Ce~ 10-14         830     1413
##  8        276.        270.        273. Mountain Stat~ 10-14         533      819
##  9        932.        918.        925. Pacific        10-14        1730     2687
```

```
## 10     -30963.     -31782.     -31373. New England    15-19      354131   314048
## # ... with 62 more rows, and 6 more variables: race <fct>, sex <fct>,
## #   age_1950 <fct>, sr <dbl>, pop1_forward <dbl>, pop0_reverse <dbl>
```

6.4.11 Exercise (ex4.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(migest)
##
##
##
# 1. Run the code below to read in the population age structure data for Quebec
#    and a range of survival ratios
q <- read_csv("./data/quebec_1956.csv")
q
# 2. Estimate the age specific net migration counts based on the national census
#    survival ratio (column national_csr)
d1 <- #####(.data = q,
            p##### = "pop1951",
            pop1_col = #####,
            survival_ratio_col = #####)
d1
# 3. Find the total net migration estimates for the net_average method for the
#    estimates in the previous question
#####(d1$net_average)
# 4. Estimate the age specific net migration counts based on the Quebec life
#    tables survival ratio (column quebec_ltsr)
d2 <- net_sr(.data = #####,
            pop0_col = #####,
            pop1_col = "pop1956",
            ##### = #####)
d2
# 5. Find the total net migration estimates for the net_average method for the
#    estimates in the previous question
#    Note: the total should have the opposite sign, as the national survival
#          ratios do not account for international migration and regional
#          differences in mortality
sum(d2$#####)
# 6. Run the code below to read in the population age structure data for
```

```
#    Franklin, Ohio
f <- read_csv("./data/franklin_1960.csv")
f
# 7. Run the code below to move the births into the pop1950 column
f <- f %>%
  mutate(pop1950 = ifelse(is.na(pop1950), births, pop1950))
f
# 8. Estimate the age specific net migration counts based on the national census
#    survival ratio (column national_csr)
d3 <- net_sr(.data = #####,
             pop0_col = "pop1950",
             pop1_col = #####,
             survival_ratio_col = #####,
             net_children = #####)
# 9. Compare the total net migration estimates from each method
d3 %>%
  select(contains("#####")) %>%
  summarise_all(sum)
```

Chapter 7

Describing and Estimating Migration Age Structures

7.1 Rogers Castro

7.1.1 Rogers Castro migration age schedules

- Populations tend to experience demographic events, such fertility, mortality and migration, with persistent regularities in the age-specific rates
- Demographers have summarsied regularities in rates using mathematical expressions called model schedules.
- Rogers and Castro [1981] first proposed a migration model schedules via an analysis of over 500 age profiles of migration

7.1.2 Rogers Castro migration age schedules

Composed of curves based on migration of different life stages:

1. Pre-labor force
2. Labor force
3. Post-labor force
4. Post-retirement
5. A constant term

$$
\begin{aligned}
m(x) = & a_1 \exp(-\alpha_1 x) \\
& + a_2 \exp(-\alpha_2(x-\mu_2) - \exp(\lambda_2(x-\mu_2))) \\
& + a_3 \exp(-\alpha_3(x-\mu_3) - \exp(\lambda_3(x-\mu_3))) \\
& + a_4 \exp(\lambda_4 x) \\
& + c
\end{aligned}
$$

7.1.3 Rogers Castro migration age schedules

Fig. 2.4 The model migration schedule fitted to the observed out-migration rates of males leaving the Stockholm region, 1974.
(*Source*: Rogers & Castro, 1981)

7.1.4 Rogers Castro migration age schedules

- Most migration age patterns have a pre-labor force downward slope and labor force peak (and a constant)
  - 7-parameter model schedule
- In specific areas (in Western countries) migration age patterns have an additional retirement peak component
  - 11-parameter model schedule
- In other areas, instead of a retirement peak, age profiles have an upward slope at the end of life
  - 9-parameter model schedule
- In even fewer cases, some instances of both a retirement peak and a post-retirement upward slope Rogers and Watkins [1987]

  - 13-parameter model schedule

- Wilson [2010] introduced a 17-parameter model to incorporate a student peak before the labour force peak.

7.1.5 Rogers Castro migration age schedules

- The `mig_calculate_rc()` function in either the *DemoTools* package by Tim Riffe et. al. or the *rcbayes* package by Monica Alexander et. al. provide a quick method to calculate migration age schedules for a given parameter set
  - Same functions by same authors. Both packages currently not on CRAN. Availability might change.

```
# install from github
# install.packages("devtools")
library(devtools)

# might need to specify download.file.method
# options(download.file.method = "libcurl")

install_github("timriffe/DemoTools")
# and/or
install_github("jessieyeung/rcbayes")
```

7.1.6 Rogers Castro migration age schedules

```
library(DemoTools)
```

```
## Loading required package: Rcpp
```

```
# define 11 parameters
p <- c(a1 = 0.1, alpha1 = 0.1,
       a2 = 0.2, alpha2 = 0.1, mu2 = 20, lambda2 = 0.5,
       a3 = 0.05, alpha3 = 0.2, mu3 = 60, lambda3 = 0.1,
       c = 0.01)

# calculate model migration schedule with 11 parameters
mx <- mig_calculate_rc(ages = 1:100, pars = p)
mx
```

```
##   [1] 0.10048374 0.09187308 0.08408182 0.07703200 0.07065307 0.06488116
##   [7] 0.05965853 0.05493290 0.05065697 0.04678794 0.04328711 0.04011942
##  [13] 0.03725318 0.03465970 0.03231470 0.03037404 0.03132289 0.04264948
##  [19] 0.06746077 0.09710942 0.12091621 0.13442550 0.13855839 0.13616639
##  [25] 0.12995494 0.12185875 0.11308333 0.10431583 0.09591794 0.08806055
##  [31] 0.08080783 0.07416691 0.06811661 0.06262465 0.05765908 0.05319728
##  [37] 0.04923336 0.04578295 0.04288297 0.04058442 0.03893812 0.03797602
##  [43] 0.03769285 0.03803360 0.03889051 0.04011081 0.04151310 0.04290860
##  [49] 0.04412234 0.04501067 0.04547234 0.04545269 0.04494158 0.04396660
##  [55] 0.04258384 0.04086761 0.03890096 0.03676762 0.03454596 0.03230498
##  [61] 0.03010206 0.02798231 0.02597892 0.02411422 0.02240126 0.02084543
##  [67] 0.01944616 0.01819839 0.01709398 0.01612274 0.01527339 0.01453424
##  [73] 0.01389365 0.01334046 0.01286417 0.01245512 0.01210453 0.01180452
##  [79] 0.01154811 0.01132916 0.01114229 0.01098283 0.01084676 0.01073060
##  [85] 0.01063138 0.01054657 0.01047399 0.01041181 0.01035847 0.01031264
##  [91] 0.01027319 0.01023918 0.01020981 0.01018438 0.01016234 0.01014319
##  [97] 0.01012651 0.01011196 0.01009924 0.01008810
```

7.1.7 Rogers Castro migration age schedules

```
library(tidyverse)
tibble(x = 1:100,
       mx = mx) %>%
  ggplot(mapping = aes(x = x, y = mx)) +
  geom_line()
```

7.2 Model schedules

7.2.1 Model migration age schedules

- The *migest* package contains two sets of parameters for model migration schedules.
- The `rc_model_fund` are the set of fundamental parameters proposed by Rogers and Castro to represent a typical migration age pattern, based on their analysis of over 500 migration flows

```
library(migest)
rc_model_fund
```

```
## # A tibble: 7 x 2
##   param    value
##   <chr>    <dbl>
## 1 a1       0.02
## 2 alpha1   0.1
## 3 a2       0.06
## 4 alpha2   0.1
## 5 mu2     20
## 6 lambda2  0.4
## 7 c        0.003
```

7.2.2 Model migration age schedules

- Plot of model age schedule based on fundamental parameters

```
# convert data frame to named vector
p <- deframe(rc_model_fund)
p
```

```
##      a1  alpha1      a2  alpha2     mu2 lambda2       c
##   2e-02   1e-01   6e-02   1e-01   2e+01   4e-01   3e-03
```

```
tibble(x = 1:100,
       mx = mig_calculate_rc(ages = x, pars = p)) %>%
  ggplot(mapping = aes(x = x, y = mx)) +
  geom_line()
```

7.2.3 Model migration age schedules

- Rogers and Castro describe the nice properties in the parameters and their relationships
- Peaking: early versus late peaking (μ_2)
  - $\mu_2 = 20$ in the fundamental parameters
- Dominance: $\gamma_{12} = a_1/a_2$ as the index of child dependency, and $1/\gamma_{12}$ as index of labor dominance
  - $\gamma_{12} = 1/3$ in fundamental parameters
- Labor asymmetry: $\sigma_2 = \lambda_2/\alpha_2$
  - $\sigma_2 = 4$ in fundamental parameters
- Regularity: $\beta_{12} = \alpha_1/\alpha_2$ how the migration rates of children match to the migration rates of parents
  - $\beta_{12} = 1$ in fundamental parameters
- Users can focus on these four measures (peaking, dominance, labor asymmetry and regularity) when describing or deriving their own model schedules

7.2.4 Model migration age schedules

- The `index_age_rc()` function in the migest package returns these ratios given a named vector of the parameters

```
rc_model_fund %>%
  deframe() %>%
  index_age_rc()
```

```
## # A tibble: 5 x 2
##   measure           value
##   <chr>             <dbl>
## 1 peaking          20
## 2 child_dependency  0.333
## 3 labor_dependency  3
## 4 labor_asymmetry   4
## 5 regularity        1
```

7.2.5 Model migration age schedules

- The `rc_model_un` are the set of fundamental parameters proposed in United Nations Department of Economic and Social Affairs Population Division [1992] for estimating age-specific migration flows in different contexts

```
rc_model_un
```

```
## # A tibble: 84 x 5
##    schedule         schedule_abb sex   param     value
##    <chr>            <chr>        <chr> <chr>     <dbl>
##  1 Western standard ws           male  a1       0.0215
##  2 Western standard ws           male  alpha1   0.105
##  3 Western standard ws           male  a2       0.0694
##  4 Western standard ws           male  alpha2   0.112
##  5 Western standard ws           male  mu2     20.0
##  6 Western standard ws           male  lambda2  0.391
##  7 Western standard ws           male  c        0.0028
##  8 Low dependency   ld           male  a1       0.0128
##  9 Low dependency   ld           male  alpha1   0.105
## 10 Low dependency   ld           male  a2       0.0804
## # ... with 74 more rows
```

7.2.6 Model migration age schedules

- To calculate model schedules we can use
  - `nest()` to group together the parameters
  - `map()` to apply the parameters to the `mig_calculate_rc()` function for each group

```
d <- rc_model_un %>%
  select(-schedule_abb) %>%
```

```
  nest(rc_param = c(param, value)) %>%
  mutate(p = map(.x = rc_param, .f = ~deframe(.x)),
         mx = map(.x = p,
                  .f = ~mig_calculate_rc(ages = 1:80, pars = .x)),
         age = list(1:80))
d
```

```
## # A tibble: 12 x 6
##    schedule                              sex    rc_param    p      mx     age
##    <chr>                                 <chr>  <list>      <list> <list> <list>
##  1 Western standard                      male   <tibble [7~ <dbl ~ <dbl ~ <int ~
##  2 Low dependency                        male   <tibble [7~ <dbl ~ <dbl ~ <int ~
##  3 High dependency                       male   <tibble [7~ <dbl ~ <dbl ~ <int ~
##  4 Young labour force entry              male   <tibble [7~ <dbl ~ <dbl ~ <int ~
##  5 Old labour force entry                male   <tibble [7~ <dbl ~ <dbl ~ <int ~
##  6 Low dependency low labour force entry male   <tibble [7~ <dbl ~ <dbl ~ <int ~
##  7 Western standard                      female <tibble [7~ <dbl ~ <dbl ~ <int ~
##  8 Low dependency                        female <tibble [7~ <dbl ~ <dbl ~ <int ~
##  9 High dependency                       female <tibble [7~ <dbl ~ <dbl ~ <int ~
## 10 Young labour force entry              female <tibble [7~ <dbl ~ <dbl ~ <int ~
## 11 Old labour force entry                female <tibble [7~ <dbl ~ <dbl ~ <int ~
## 12 Low dependency low labour force entry female <tibble [7~ <dbl ~ <dbl ~ <int ~
```

7.2.7 Model migration age schedules

```
# first row parameters
d$p[[1]]
```

```
##      a1  alpha1      a2  alpha2     mu2 lambda2       c
##  0.0215  0.1050  0.0694  0.1120 20.0400  0.3910  0.0028
```

```
# data unnested
d %>%
  select(-rc_param, -p) %>%
  unnest(c(mx, age))
```

```
## # A tibble: 960 x 4
##    schedule         sex       mx   age
##    <chr>            <chr>  <dbl> <int>
##  1 Western standard male  0.0222     1
##  2 Western standard male  0.0202     2
##  3 Western standard male  0.0185     3
```

```
##  4 Western standard male  0.0169     4
##  5 Western standard male  0.0155     5
##  6 Western standard male  0.0143     6
##  7 Western standard male  0.0131     7
##  8 Western standard male  0.0121     8
##  9 Western standard male  0.0112     9
## 10 Western standard male  0.0103    10
## # ... with 950 more rows
```

7.2.8 Model migration age schedules

- Use `unnest()` to create a data base varying by age for each model schedule and sex for plotting

```
d %>%
  unnest(c(mx, age)) %>%
  mutate(schedule = str_wrap(schedule, width = 20)) %>%
  ggplot(mapping = aes(x = age, y = mx, colour = schedule)) +
  geom_line() +
  facet_wrap(facets = "sex", ncol = 1)
```

7.2.9 Model migration age schedules

- Model migration schedules are useful when we do not have any age information, but require an estimate of age specific migration

  - For example, in cohort component projections age specific migration rates are required but might not be available in any data source

- We may use an estimate or reported data on total migration to obtain age-specific migration
  - Design or select appropriate model age schedule based on existing knowledge of migration age patterns for the given flow.

```
# example for males based on young labour force entry
p <- rc_model_un %>%
  filter(sex == "male", schedule_abb == "ylfe") %>%
  select(param, value) %>%
  deframe()
p
```

```
##      a1  alpha1      a2  alpha2     mu2 lambda2       c
##  0.0215  0.1050  0.0691  0.1120 16.0900  0.3910  0.0028
```

7.2.10 Model migration age schedules

```
tibble(x = 1:90,
       mx = mig_calculate_rc(ages = x, pars = p),
       # calculate number of migrants, given a total estimate of 10,000
       Mx = 10000 * mx) %>%
  ggplot(mapping = aes(x = x, y = Mx)) +
  geom_line()
```

7.3 Fitting schedules

7.3.1 Fitting Roger Castro migration age schedules

- If we have age-specific migration data we might want to estimate the parameters of a Rogers Castro age schedule to
  - Smooth the data
  - Analyse the parameter estimates
  - Create projected age schedules based on past patterns of the age schedule parameters
- Fitting Rogers Castro migration age schedules can be difficult.
  - A number of different software has been used to fit age schedules including Rogers and Little [1994], TableCurve 2D Rogers and Raymer [1999], MATLAB Rogers et al. [2010], and Excel Wilson [2010].
- The `mig_estimate_rc()` function in DemoTools or rcbayes uses Stan, via the rstan package, a Bayesian probabilistic programming language
  - Estimation is carried out using MCMC sampling.
- Requires two arguments
  - `ages` a vector of migration ages
  - `mx` a vector of standardized migration intensities for the corresponding ages
  - Specify form of age schedule using the `pre_working_age`, `working_age`, `retirement` and `post_retirement` arguments - set to `TRUE` or `FALSE`

7.3.2 Fitting Roger Castro migration age schedules

- Demonstrate with five-year data from the `italy_area` data set in *migest*
  - Calculate the out-migration for Islands (Sicily and Sardinia) in 1970

```
# include a numeric age column for mig_estimate_rc()
i <- italy_area %>%
  filter(year == 1970) %>%
  group_by(age_grp) %>%
  sum_turnover() %>%
  filter(region == "Islands") %>%
  separate(col = age_grp, into = c("age_min", "age_max"),
           remove = FALSE, convert = TRUE)
```

```
## Adding missing grouping variables: `age_grp`
```

```
i
```

```
## # A tibble: 20 x 8
## # Groups:   age_grp [20]
##    age_grp age_min age_max region  in_mig out_mig  turn   net
##    <fct>     <int>   <int> <chr>    <dbl>   <dbl> <dbl> <dbl>
##  1 0-4           0       4 Islands   4532    7876 12408 -3344
##  2 5-9           5       9 Islands   3592    7271 10863 -3679
##  3 10-14        10      14 Islands   2228    5779  8007 -3551
##  4 15-19        15      19 Islands   3064    8526 11590 -5462
##  5 20-24        20      24 Islands   6861   15629 22490 -8768
##  6 25-29        25      29 Islands   5891   11224 17115 -5333
##  7 30-34        30      34 Islands   4042    7046 11088 -3004
##  8 35-39        35      39 Islands   2480    4612  7092 -2132
##  9 40-44        40      44 Islands   1737    3634  5371 -1897
## 10 45-49        45      49 Islands   1383    2783  4166 -1400
## 11 50-54        50      54 Islands    910    1716  2626  -806
## 12 55-59        55      59 Islands    899    1587  2486  -688
## 13 60-64        60      64 Islands    789    1217  2006  -428
## 14 65-69        65      69 Islands    602     924  1526  -322
## 15 70-74        70      74 Islands    427     702  1129  -275
## 16 75-79        75      79 Islands    311     490   801  -179
## 17 80-84        80      84 Islands    158     268   426  -110
## 18 85-89        85      89 Islands     59     116   175   -57
## 19 90-94        90      94 Islands     17      35    52   -18
## 20 95+          95      NA Islands     95     137   232   -42
```

7.3.3 Fitting Roger Castro migration age schedules

- Requires a standardized age schedule (where values sum to one)
- Will take a few minutes and print out lots of messages from Stan

```
m <- i$out_mig/sum(i$out_mig)
m
```

```
##  [1] 0.0965527387 0.0891359780 0.0708453881 0.1045211592 0.1915976070
##  [6] 0.1375962340 0.0863776786 0.0565390085 0.0445496004 0.0341170990
## [11] 0.0210366302 0.0194552052 0.0149193351 0.0113274163 0.0086058942
## [16] 0.0060069632 0.0032854411 0.0014220566 0.0004290688 0.0016794979
```

```
f <- mig_estimate_rc(ages = i$age_min + 2.5, mx = m,
                     # set model components
                     pre_working_age = TRUE, working_age = TRUE,
                     retirement = FALSE, post_retirement = FALSE)
```

```
##
## SAMPLING FOR MODEL 'f4d0f16f36ddb7179a67ef654e5d224a' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.695 seconds (Warm-up)
## Chain 1:                0.585 seconds (Sampling)
## Chain 1:                1.28 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'f4d0f16f36ddb7179a67ef654e5d224a' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
```

```
## Chain 2:  Elapsed Time: 0.754 seconds (Warm-up)
## Chain 2:                0.617 seconds (Sampling)
## Chain 2:                1.371 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'f4d0f16f36ddb7179a67ef654e5d224a' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 secon
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.771 seconds (Warm-up)
## Chain 3:                0.67 seconds (Sampling)
## Chain 3:                1.441 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'f4d0f16f36ddb7179a67ef654e5d224a' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 secon
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.716 seconds (Warm-up)
## Chain 4:                0.684 seconds (Sampling)
## Chain 4:                1.4 seconds (Total)
## Chain 4:
```

7.3.4 Fitting Roger Castro migration age schedules

The fitted object has two components

```r
# parameter estimates
f[[1]]
```

```
## # A tibble: 7 x 4
##   variable     median      lower    upper
##   <chr>         <dbl>      <dbl>    <dbl>
## 1 a1[1]       0.107    0.0943     0.118
## 2 a2[1]       0.341    0.275      0.381
## 3 alpha1[1]   0.0322   0.0270     0.0416
## 4 alpha2[1]   0.227    0.152      0.301
## 5 c           0.00157  0.0000724  0.00752
## 6 lambda2[1]  0.185    0.151      0.285
## 7 mu2[1]     24.6     20.8       27.0
```

```r
# fitted schedule
f[[2]]
```

```
## # A tibble: 20 x 6
##      age     data  median   lower  upper     diff_sq
##    <dbl>    <dbl>   <dbl>   <dbl>  <dbl>       <dbl>
##  1   2.5 0.0966   0.100   0.0902  0.111  0.0000152
##  2   7.5 0.0891   0.0857  0.0775  0.0937 0.0000121
##  3  12.5 0.0708   0.0738  0.0673  0.0804 0.00000881
##  4  17.5 0.105    0.107   0.0935  0.120  0.00000573
##  5  22.5 0.192    0.183   0.168   0.196  0.0000682
##  6  27.5 0.138    0.144   0.133   0.154  0.0000459
##  7  32.5 0.0864   0.0842  0.0725  0.0945 0.00000456
##  8  37.5 0.0565   0.0506  0.0433  0.0587 0.0000357
##  9  42.5 0.0445   0.0350  0.0299  0.0403 0.0000904
## 10  47.5 0.0341   0.0272  0.0226  0.0316 0.0000483
## 11  52.5 0.0210   0.0223  0.0178  0.0271 0.00000172
```

```
## 12  57.5 0.0195   0.0190  0.0145  0.0237 0.000000235
## 13  62.5 0.0149   0.0163  0.0122  0.0210 0.00000203
## 14  67.5 0.0113   0.0142  0.0102  0.0186 0.00000806
## 15  72.5 0.00861  0.0124  0.00868 0.0167 0.0000141
## 16  77.5 0.00601  0.0108  0.00734 0.0149 0.0000231
## 17  82.5 0.00329  0.00949 0.00628 0.0136 0.0000385
## 18  87.5 0.00142  0.00837 0.00537 0.0125 0.0000483
## 19  92.5 0.000429 0.00738 0.00462 0.0115 0.0000483
## 20  97.5 0.00168  0.00655 0.00398 0.0107 0.0000238
```

7.3.5 Fitting Roger Castro migration age schedules

```
ggplot(data = f[[2]],
       mapping = aes(x = age, y = data)) +
  geom_ribbon(mapping = aes(ymin = lower, ymax = upper), fill = "lightblue") +
  geom_line(mapping = aes(y = median), colour = "blue") +
  geom_point()
```

7.3.6 Fitting Roger Castro migration age schedules

- The *migraR* package by Ruiz-Santacruz and Garcés also has functions to estimate parameters in Rogers Castro schedule
  - Also not on CRAN

  - Uses an optimization procedure (non-Bayesian)
  - Functions to select best form schedule
- Selecting the form of the schedule usually requires some form of visual inspection

7.4 Age Indices

7.4.1 Age Indices

- Number of criticisms of model age schedules for migration (Bell et al. [2002], Bernard et al. [2014])
- Not always clear how many parameters should be included in model schedule
  - Parameter estimates sensitive to the choice of model form, making comparisons difficult
  - Use statistical accuracy measures to select best form, at the risk of over fitting
- Parameter estimates sensitive to initial values
  - Unlikely to be the case when using `mig_estimate_rc()`
- Unstable parameter estimates
  - Sensitive to measurement error in age-specific migration
- Interpretation of parameter estimates
  - The indexes in `index_age_rc()` have not been widely adopted, probably because of difficulty in fitting model schedules.

7.4.2 Age Indices

- A number of other measures of age specific migration have been proposed that do not require fitting model age schedules.
- Most a dependent on the migration intensity m_{as}, the number of migrants in a age group and given time period as a percentage of the population at risk of moving.
- Rogers [1975] proposed a Gross Migraproduction Rate (GMR) based on the sum of age-specific (and sex-specific) migration intensities

$$GMR = \sum_{as} m_{as}$$

- Bell et al. [2002] introduced
  - Peak migration intensity, the largest age-specific migration intensity of any age-group
  - Peak age, the corresponding age of the peak migration intensity

7.4.3 Age Indices

- Bell and Muhidin [2009] proposed and additional measures
  - Breadth of peak based on the sum of the peak migration intensity at the peak age and the five age-groups before and after the peak.
  - Peak share based on the percentage of the normalized migration age schedule covered by the peak age and the five age-groups before and after the peak.
- Bernard et al. [2014] provide three additioanl measures
  - The Maximum Upward Rate of Change (MURC) for the largest gradient in the slope of the labour force peak before the peak age
  - The Maximum Downward Rate of Change (MDRC) for the largest gradient in the slope of the labour force peak after the peak age
  - The asymmetry of the labour force peak based on the ratio of MURC and MDRC
- Each of these measures area calculated in the `age_index()` function in the *migest* package

7.4.4 Age Indices

- To demonstrate we use the age schedule data of Brazil 2000 and France 2006 in the `ipumsi_age` data frame of the *migest* package
  - Migration based on five-year transitions between any minor (and major) administrative units.

```
ipumsi_age %>%
  mutate(mi = migrants/population) %>%
  filter(age > 5) %>%
  ggplot(mapping = aes(x = age, y = mi)) +
  geom_line() +
  facet_wrap(facets = "sample", scales = "free")
```

7.4.5 Age Indices

- Bernard et al. [2014] recommends smoothing age schedules before calculating index values
  - Get very similar results without smoothing - at least in these examples
- `index_age()` by default ignores values above 65 (and below 5) when calculating peak index statistics
  - GMR still sensitive for outliers (e.g. oldest in Brazil)
- Index values for Brazil 2000

```
ipumsi_age %>%
  filter(sample == "BRA2000") %>%
  mutate(mi = migrants/population) %>%
  index_age()
```

```
## # A tibble: 8 x 2
##   measure        value
##   <chr>          <dbl>
## 1 gmr            7.82
## 2 peak_mi       14.3
## 3 peak_age      24
## 4 peak_breadth 147.
## 5 peak_share    18.8
## 6 murc          19
## 7 mdrc          32
## 8 asymmetry      0.594
```

7.4.6 Age Indices

- Index values are most useful for comparing age-specific migration in different countries (or regions or time periods)

```
ipumsi_age %>%
  group_by(sample) %>%
  mutate(mi = migrants/population) %>%
  index_age() %>%
  pivot_wider(names_from = sample, values_from = value)
```

```
## # A tibble: 8 x 3
##   measure      BRA2000 FRA2006
##   <chr>          <dbl>   <dbl>
## 1 gmr             7.82     9.55
## 2 peak_mi        14.3     29.5
## 3 peak_age       24       26
## 4 peak_breadth 147.     295.
## 5 peak_share     18.8     30.8
## 6 murc           19       18
## 7 mdrc           32       30
## 8 asymmetry       0.594    0.6
```

7.5 Smoothing

7.5.1 General purpose smoothing functions

- There are many non-parametric smoothing functions in R that can be used to smooth data.
- The `stats` package, which is loaded when R opens, includes
  - `ksmooth()` is a kernel regression smoother
  - `loess.smooth()` is a Local Polynomial Regression Fitting method
  - `smooth.spline` a cubic spline fit
- The DemoTools package contains a `smooth_age_5()` that is particularly useful for age-heaped data.

7.5.2 General purpose smoothing functions

- Smoothing methods perform some form of weighting data points on separate subsections (windows or bandwidths of the data)

- In a migration age schedule context, this involves some form of simple local regression or averaging of migration intensities at each age, given data from nearby ages.
- Careful consideration is usually required in choosing the bandwidth.
  - The default values are not always sensible for migration age schedules
- Might consider censoring the very oldest values where values can become volatile due to small numbers

7.5.3 General purpose smoothing functions

- Use Brazil 2000 IPUMS International sample data to demonstrate
  - Particularly rough at older age groups

```
b <- ipumsi_age %>%
  filter(sample == "BRA2000",
         age > 5) %>%
  mutate(mi = migrants/population)

ggplot(data = b, mapping = aes(x = age, y = mi)) +
  geom_point()
```

7.5.4 General purpose smoothing functions

- Most smoothing function in R require two vectors (`x` and `y`)
  - Optional arguments to control the smoothness of the fit(names differ for different smoothing functions)

- Will return a list with two components (`x` and `y`), where the length of x may differ from the original vector provided
  - Set a output length argument (names differ for different smoothing functions)
  - The `x` component will match age values
  - Can use within `mutate()`

```
k1 <- ksmooth(x = b$age, y = b$mi)
str(k1)
```

```
## List of 2
##  $ x: num [1:100] 6 6.95 7.9 8.85 9.8 ...
##  $ y: num [1:100] 0.1074 0.104 0.1004 0.0962 0.0969 ...
```

```
k2 <- ksmooth(x = b$age, y = b$mi, n.points = nrow(b))
str(k2)
```

```
## List of 2
##  $ x: num [1:95] 6 7 8 9 10 11 12 13 14 15 ...
##  $ y: num [1:95] 0.1074 0.104 0.1004 0.0962 0.0969 ...
```

7.5.5 Kernal smoothing

- The `ksmooth` function is unlikely to smooth a migration age schedule as the default bandwidth parameter is too small
  - Increase for a more suitable fit

```
b <- b %>%
  mutate(
    k_default = ksmooth(x = age, y = mi, n.points = n())$y,
    k_bw5 = ksmooth(x = age, y = mi, n.points = n(), bandwidth = 5)$y,
    k_bw10 = ksmooth(x = age, y = mi, n.points = n(), bandwidth = 10)$y
  )
b
```

```
## # A tibble: 95 x 8
##    sample    age migrants population     mi k_default  k_bw5 k_bw10
##    <chr>   <dbl>    <dbl>      <dbl>  <dbl>     <dbl>  <dbl>  <dbl>
##  1 BRA2000     6  355723.   3311728. 0.107     0.107  0.104  0.100
##  2 BRA2000     7  343852.   3307567. 0.104     0.104  0.102  0.0990
##  3 BRA2000     8  327166.   3258046. 0.100     0.100  0.101  0.0983
##  4 BRA2000     9  314905.   3272305. 0.0962    0.0962 0.0986 0.0978
```

```
##   5 BRA2000    10  324066.   3345583. 0.0969    0.0969 0.0964 0.0976
##   6 BRA2000    11  329525.   3451739. 0.0955    0.0955 0.0949 0.0978
##   7 BRA2000    12  327113.   3518160. 0.0930    0.0930 0.0944 0.0975
##   8 BRA2000    13  323180.   3473133. 0.0931    0.0931 0.0942 0.0982
##   9 BRA2000    14  334783.   3566239. 0.0939    0.0939 0.0950 0.100
## 10 BRA2000    15  337297.   3528845. 0.0956    0.0956 0.0973 0.103
## # ... with 85 more rows
```

7.5.6 Kernal smoothing

```
ggplot(data = b, mapping = aes(x = age, y = mi)) +
  geom_point(alpha = 0.5) +
  geom_line(mapping = aes(y = k_default), col = "darkgrey") +
  geom_line(mapping = aes(y = k_bw5), col = "orange") +
  geom_line(mapping = aes(y = k_bw10), col = "red")
```

7.5.7 Loess smoothing

- The `loess.smooth` function is also unlikely to smooth a migration age schedule as the default span parameter is too small

- Adjust the smoothing parameter using `spar` (between 0 and 1), default is `2/3`
- Use `evaluation` to set the number of predicted values

```
b <- b %>%
  mutate(
    lo_default = loess.smooth(x = age, y = mi, evaluation = n())$y,
    lo_sp2 = loess.smooth(x = age, y = mi, evaluation = n(), span = 0.2)$y,
    lo_sp1 = loess.smooth(x = age, y = mi, evaluation = n(), span = 0.1)$y,
  )
```

7.5.8 Loess smoothing

```
ggplot(data = b,
       mapping = aes(x = age, y = mi)) +
  geom_point(alpha = 0.5) +
  geom_line(mapping = aes(y = lo_default), col = "darkgrey") +
  geom_line(mapping = aes(y = lo_sp2), col = "orange") +
  geom_line(mapping = aes(y = lo_sp1), col = "red")
```

7.5.9 Cubic spline smoothing

- The `smooth.spline` function might have a nice smooth fit to migration age schedule
  - Adjust the smoothing parameter using `spar` (between 0 and 1)
  - Use `n` to set the number of predicted values

```
b <- b %>%
  mutate(
    s_default = smooth.spline(x = age, y = mi, n = n())$y,
    s_sp6 = smooth.spline(x = age, y = mi, n = n(), spar = 0.6)$y,
    s_sp8 = smooth.spline(x = age, y = mi, n = n(), spar = 0.8)$y)
```

7.5.10 Cubic spline smoothing

```
ggplot(data = b,
       mapping = aes(x = age, y = mi)) +
  geom_point(alpha = 0.5) +
  geom_line(mapping = aes(y = s_default), col = "darkgrey") +
  geom_line(mapping = aes(y = s_sp6), col = "orange") +
  geom_line(mapping = aes(y = s_sp8), col = "red")
```

7.6 Graduating

7.6.1 Graduating

- If you require single year migration age data, but only have data by age groups, then graduating methods can be used to estimate migration for each age that sum to the reported age group totals.
- There a multiple graduating methods available in the `graduate()` function in the DemoTools package
  - Built for interpolating population totals, but also suitable for migration flows
  - See the guide for more detail on different methods
- Requires users to provide `Value` and minimum `Age`.
- Can also specify the maximum value of final open age group, if exists, for certain methods such as `pclm`.

7.6.2 Graduating

- Using the out-migration to Italian islands area in 1970

```r
head(i)
```

```
## # A tibble: 6 x 8
## # Groups:   age_grp [6]
##   age_grp age_min age_max region  in_mig out_mig  turn   net
##   <fct>     <int>   <int> <chr>    <dbl>   <dbl> <dbl> <dbl>
## 1 0-4           0       4 Islands   4532    7876 12408 -3344
## 2 5-9           5       9 Islands   3592    7271 10863 -3679
## 3 10-14        10      14 Islands   2228    5779  8007 -3551
## 4 15-19        15      19 Islands   3064    8526 11590 -5462
## 5 20-24        20      24 Islands   6861   15629 22490 -8768
## 6 25-29        25      29 Islands   5891   11224 17115 -5333
```

```r
mx <- graduate(Value = i$out_mig, Age = i$age_min,
               method = "pclm", OAG =  TRUE, OAnew = 100)
mx
```

```
##           0           1           2           3           4           5
## 1540.822616 1563.081967 1582.487050 1594.810184 1594.859870 1576.283303
##           6           7           8           9          10          11
## 1534.233098 1470.025351 1388.679355 1301.771896 1221.108012 1158.023181
##          12          13          14          15          16          17
## 1121.942083 1119.554144 1158.435391 1247.103598 1398.284195 1623.913611
##          18          19          20          21          22          23
## 1935.092918 2321.755623 2741.539266 3112.329630 3324.796748 3324.915552
##          24          25          26          27          28          29
## 3125.320990 2812.139216 2482.216694 2189.772149 1958.656025 1781.415216
##          30          31          32          33          34          35
## 1641.408137 1521.553492 1407.170481 1293.744997 1182.104397 1077.109503
##          36          37          38          39          40          41
##  984.353885  906.668010  845.151088  798.771354  765.545510  742.411329
##          42          43          44          45          46          47
##  725.616964  710.186702  690.289756  660.651697  617.401802  562.289211
##          48          49          50          51          52          53
##  501.033932  441.541294  391.355174  354.054434  330.929904  320.470746
##          54          55          56          57          58          59
##  319.333087  323.280116  326.437894  324.340533  314.694694  298.140366
##          60          61          62          63          64          65
##  278.157544  258.006157  240.321118  226.037262  214.561457  204.846583
##          66          67          68          69          70          71
##  195.376275  185.304981  174.609045  163.809589  153.997862  145.822900
##          72          73          74          75          76          77
##  139.398446  134.182156  128.693201  121.517548  111.497757   98.816336
##          78          79          80          81          82          83
```

```
##   85.271323   72.734621   62.959243   56.411258   52.462550   49.962470
##          84          85          86          87          88          89
##   46.549791   40.446619   31.418669   21.471299   13.559411    8.503983
##          90          91          92          93          94          95
##    5.890223    4.961916    5.352577    7.481251   12.438906   21.715730
##          96          97          98          99         100
##   32.832828   36.073090   26.875823   13.368895    4.891892
```

7.6.3 Graduating

```r
# check for close match between graduate values and out_mig
# 0-4
sum(mx[1:5])
```

```
## [1] 7876.062
```

```r
# 5-9
sum(mx[6:10])
```

```
## [1] 7270.993
```

```r
select(i, age_grp, out_mig)
```

```
## # A tibble: 20 x 2
## # Groups:   age_grp [20]
##    age_grp out_mig
##    <fct>     <dbl>
##  1 0-4        7876
##  2 5-9        7271
##  3 10-14      5779
##  4 15-19      8526
##  5 20-24     15629
##  6 25-29     11224
##  7 30-34      7046
##  8 35-39      4612
##  9 40-44      3634
## 10 45-49      2783
## 11 50-54      1716
## 12 55-59      1587
## 13 60-64      1217
## 14 65-69       924
## 15 70-74       702
```

```
## 16 75-79        490
## 17 80-84        268
## 18 85-89        116
## 19 90-94         35
## 20 95+          137
```

7.6.4 Graduating

```
# different scales in y-axis
ggplot(data = i,
       mapping = aes(x = age_min + 2.5, y = out_mig)) +
  geom_point()

tibble(age = 0:100, mx = mx) %>%
  ggplot(mapping = aes(x = age, y = mx)) +
  geom_line()
```

7.6.5 Exercise 5 (ex5.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
getwd()
```

```
#     b) Load the packages used in this exercise
library(tidyverse)
library(migest)
library(DemoTools)
##
##
##
# 1. Run the code below to read in the population age structure data for flows
#    from Florida to New York based on the 2015 American Community Survey
flny <- read_csv("./data/florida_new_york_acs_2015.csv")
flny
# 2. Run the code below to plot the age schedule for migration from New York to
#    Florida. Note, the uneven spread of the age groups
ggplot(data = x, mapping = aes(x = AGE_label, y = mig_in, group = 1)) +
  geom_point() +
  geom_line() +
  theme(axis.text = element_text(angle = 45, hjust = 1))
# 3. Estimate the age schedule based on single years up to 100, based on a
#    graduation of the migration data in flny
mx <- #####(Value = flny$#####, Age = #####$age_min,
               method = "pclm", OAG =  TRUE, OAnew = #####)
mx
# 4. Build a data frame to store the graduated data frame and a migration
#    intensities (mx rescaled so that age specific intensities sum to one)
d <- tibble(
  age = 1:100,
  mx = mx,
  mi = #####/sum(mx)
)
d
# 5. Plot the graduated age schedule
d %>%
  ggplot(mapping = aes(x = #####, y = #####)) +
  geom_line()
# 6. Use the age and migration intensities in d to fit a 11 parameter Rogers-
#    Castro age schedule (with a retirement peak, but no post retirement peak)
f <- mig_estimate_rc(ages = d$#####, mx = d$mi,
                     pre_working_age = #####, working_age = TRUE,
                     retirement = #####, post_retirement = FALSE)
# 7. Run the code below to plot the fitted Rogers Casto age schedule
ggplot(data = f[[2]],
       mapping = aes(x = age, y = data)) +
  geom_ribbon(mapping = aes(ymin = lower, ymax = upper), fill = "lightblue") +
  geom_line(mapping = aes(y = median), colour = "blue") +
  geom_point()
```

```
# 8. Calculate the indices based on the median of the parameter distributions
#    for the Rogers Castro age schedule
f[[1]] %>%
  select(variable, median) %>%
  #####() %>%
  #####()
```

Chapter 8

Describing Bilteral Migration Data

8.1 Components

8.1.1 Multiplicative Component Model

- Rogers et al. [2002] proposed dis-aggregating origin-destination flow tables into separate components to allow for an easier examination of migration flows
  - Overall component - level of migration γ
  - Origin component - relative 'pushes' from each region α_i
  - Destination component - relative 'pulls' to each region β_j
  - Origin–Destination interaction component - physical or social distance between places not explained by the overall and main effects. δ_{ij}
- Simple calculations to estimate each component:

$$\gamma = m_{++} \qquad \alpha_i = \frac{m_{i+}}{m_{++}} \qquad \beta_j = \frac{m_{+j}}{m_{++}} \qquad \delta_{ij} = \frac{m_{ij}}{\gamma\alpha_i\beta_j}$$

- The interaction, δ_{ij}, is the ratio of observed flow to an expected flow (for the case of no interaction).

8.1.2 Multiplicative Component Model

- The dis-aggregation of the components is multiplicative:

$$m_{ij} = \gamma\alpha_i\beta_j\delta_{ij}$$

- Equivalent to a saturated Poisson regression model ($R^2 = 1$) where
  - γ is constant term
  - α_i is categorical term for the origin regions
  - β_j is categorical term for the destination regions
  - δ_{ij} is an interaction term between the α_i and β_j

$$\log m_{ij} = \gamma + \alpha_i ORIG_i + \beta_j DEST_j + \delta_i ORIG_i : DEST_j$$

- When data is in a tidy format with row h would be:

$$\log y_h = \beta_0 + \beta_1 ORIG_h + \beta_2 DEST_h + \beta_3 ORIG_h : DEST_h$$

- Poisson regression models such as these - where all the predictor variables are categorical - are also know as log-linear models
- Standard functions for fitting regression models, such as `glm()` in R will provide the same fitted values, but different parameter estimates
  - Use different coding system for the constraints when estimating parameters
  - Rogers' terms the parameter estimates using the equations for $\gamma, \alpha_i, \beta_j$ and δ_{ij} above the *total reference* coding system

8.1.3 Multiplicative Component Model

- The *migest* package contains a `multi_comp()` function to generate parameter estimates from an origin-destination flow matrix
  - Demonstrate with previous dummy data set

```
r <- LETTERS[1:4]
m0 <- matrix(data = c(0, 100, 30, 70,
                      50, 0, 45, 5,
                      60, 35, 0, 40,
                      20, 25, 20, 0),
             nrow = 4, ncol = 4, byrow = TRUE,
             dimnames = list(orig = r, dest = r))
addmargins(m0)
```

```
##      dest
## orig    A   B  C   D Sum
##   A     0 100 30  70 200
##   B    50   0 45   5 100
##   C    60  35  0  40 135
##   D    20  25 20   0  65
##   Sum 130 160 95 115 500
```

8.1.4 Multiplicative Component Model

```
library(tidyverse)
library(migest)
m0 %>%
  multi_comp() %>%
  round(3)
```

```
##      dest
## orig        A       B       C       D     Sum
##   A     0.000   1.563   0.789   1.522   0.400
##   B     1.923   0.000   2.368   0.217   0.200
##   C     1.709   0.810   0.000   1.288   0.270
##   D     1.183   1.202   1.619   0.000   0.130
##   Sum   0.260   0.320   0.190   0.230 500.000
```

8.1.5 Multiplicative Component Model

- As the model is saturated, the fitted values are the same as the observed values.

```
multi_comp(m = m0)
```

```
##      dest
## orig            A           B           C           D         Sum
##   A     0.0000000   1.5625000   0.7894737   1.5217391   0.4000000
##   B     1.9230769   0.0000000   2.3684211   0.2173913   0.2000000
##   C     1.7094017   0.8101852   0.0000000   1.2882448   0.2700000
##   D     1.1834320   1.2019231   1.6194332   0.0000000   0.1300000
##   Sum   0.2600000   0.3200000   0.1900000   0.2300000 500.0000000
```

```
# fitted value for A to B
500 * 0.4 * 0.32 * 1.5625
```

```
## [1] 100
```

8.1.6 Multiplicative Component Model

- The total reference coding scheme for the parameter estimates are easier to examine than parameter estimates from a Poisson model fitted using `glm()`

- More detail on `glm()` in next section

```
d0 <- as.data.frame.table(x = m0, responseName = "flow")
f0 <- glm(formula = flow ~ orig + dest + orig * dest, family = poisson(),
          data = d0)
f0
```

```
##
## Call:  glm(formula = flow ~ orig + dest + orig * dest, family = poisson(),
##     data = d0)
##
## Coefficients:
## (Intercept)        origB        origC        origD        destB        destC
##      -24.30        28.21        28.40        27.30        28.91        27.70
##       destD  origB:destB  origC:destB  origD:destB  origB:destC  origC:destC
##       28.55       -57.12       -29.45       -28.68       -27.81       -56.10
## origD:destC  origB:destD  origC:destD  origD:destD
##      -27.70       -30.85       -28.96       -55.85
##
## Degrees of Freedom: 15 Total (i.e. Null);  0 Residual
## Null Deviance:       463.7
## Residual Deviance: 2.232e-10     AIC: 96.27
```

8.1.7 Multiplicative Component Model

```
# fitted and observed values are the same
d0 %>%
  as_tibble() %>%
  mutate(fit = round(f0$fitted.values, digits = 5))
```

```
## # A tibble: 16 x 4
##    orig  dest   flow   fit
##    <fct> <fct> <dbl> <dbl>
##  1 A     A         0     0
##  2 B     A        50    50
##  3 C     A        60    60
##  4 D     A        20    20
##  5 A     B       100   100
##  6 B     B         0     0
##  7 C     B        35    35
##  8 D     B        25    25
##  9 A     C        30    30
## 10 B     C        45    45
```

```
## 11 C     C         0     0
## 12 D     C        20    20
## 13 A     D        70    70
## 14 B     D         5     5
## 15 C     D        40    40
## 16 D     D         0     0
```

8.1.8 Multiplicative Component Model

- Rogers' and colleagues have used the multiplicative component model to estimate migration flow tables
- Expand to multiple dimensions
- Rectify bumpy age schedules
  - Replace reported age parameters (proportions) in the multiplicative component model with proportions from a more regular schedule.
  - Multiply the new age parameters with the existing total, origin, destination and interaction parameters to obtain new estimated flows.

8.1.9 Multiplicative Component Model

- Italian data in *migest* package

```
italy_area
```

```
## # A tibble: 3,500 x 5
##    orig       dest        year age_grp  flow
##    <chr>      <chr>      <dbl> <fct>   <dbl>
##  1 North-West North-West  1970 0-4         0
##  2 North-East North-West  1970 0-4      2350
##  3 Center     North-West  1970 0-4      1687
##  4 South      North-West  1970 0-4      9697
##  5 Islands    North-West  1970 0-4      5139
##  6 North-West North-East  1970 0-4      2448
##  7 North-East North-East  1970 0-4         0
##  8 Center     North-East  1970 0-4      1063
##  9 South      North-East  1970 0-4      1560
## 10 Islands    North-East  1970 0-4       689
## # ... with 3,490 more rows
```

8.1.10 Multiplicative Component Model

```r
# single year, multiple age groups
c0 <- italy_area %>%
  filter(year == 2000) %>%
  multi_comp()
round(c0, 3)
```

```
## , , age_grp = 0-4
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.401      0.859      0.909      2.370      0.010
##   Islands         0.970      0.000      1.181      1.513      0.681      0.012
##   North-East      1.053      1.916      0.000      1.179      2.501      0.010
##   North-West      0.877      2.490      0.887      0.000      2.023      0.014
##   South           1.409      0.531      1.184      1.102      0.000      0.025
##   Sum             0.016      0.007      0.017      0.018      0.014      0.072
##
## , , age_grp = 5-9
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.589      0.779      0.762      2.243      0.007
##   Islands         1.166      0.000      1.393      1.707      0.562      0.010
##   North-East      0.840      1.932      0.000      0.936      2.085      0.006
##   North-West      0.877      2.714      0.844      0.000      1.963      0.010
##   South           1.387      0.507      1.283      1.151      0.000      0.018
##   Sum             0.011      0.005      0.012      0.012      0.009      0.050
##
## , , age_grp = 10-14
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.570      0.738      0.667      1.978      0.005
##   Islands         1.333      0.000      1.572      1.791      0.463      0.008
##   North-East      0.861      1.834      0.000      0.840      1.805      0.004
##   North-West      0.793      2.694      0.826      0.000      1.959      0.007
##   South           1.424      0.411      1.332      1.226      0.000      0.014
##   Sum             0.009      0.004      0.010      0.009      0.006      0.037
##
## , , age_grp = 15-19
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.358      0.732      0.668      1.673      0.005
##   Islands         1.261      0.000      1.617      2.109      0.417      0.009
```

```
##   North-East      0.677      1.769      0.000      0.847      1.697      0.004
##   North-West      0.629      2.606      0.818      0.000      1.803      0.007
##   South           1.449      0.347      1.449      1.340      0.000      0.016
##   Sum             0.009      0.004      0.011      0.011      0.006      0.041
##
## , , age_grp = 20-24
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.044      0.852      0.759      1.552      0.014
##   Islands         0.983      0.000      1.490      1.948      0.436      0.025
##   North-East      0.593      1.530      0.000      0.852      1.808      0.012
##   North-West      0.533      1.880      0.726      0.000      1.449      0.018
##   South           1.419      0.425      1.788      1.624      0.000      0.055
##   Sum             0.025      0.009      0.036      0.037      0.017      0.124
##
## , , age_grp = 25-29
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.092      0.992      0.939      2.093      0.027
##   Islands         0.915      0.000      1.221      1.599      0.544      0.034
##   North-East      0.910      1.420      0.000      1.161      1.829      0.023
##   North-West      0.795      1.652      0.947      0.000      1.719      0.034
##   South           1.473      0.482      1.457      1.373      0.000      0.079
##   Sum             0.044      0.014      0.053      0.054      0.032      0.197
##
## , , age_grp = 30-34
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.211      1.088      1.159      2.136      0.025
##   Islands         0.915      0.000      1.053      1.390      0.526      0.025
##   North-East      1.143      1.384      0.000      1.362      1.837      0.021
##   North-West      0.945      1.857      1.091      0.000      1.756      0.031
##   South           1.544      0.445      1.205      1.244      0.000      0.059
##   Sum             0.039      0.012      0.040      0.043      0.027      0.160
##
## , , age_grp = 35-39
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.439      1.175      1.245      2.126      0.016
##   Islands         0.956      0.000      1.073      1.372      0.407      0.015
##   North-East      1.278      1.396      0.000      1.484      1.719      0.013
##   North-West      1.158      2.026      1.229      0.000      1.753      0.020
```

```
##   South           1.465      0.424      1.089      1.085      0.000      0.032
##   Sum             0.024      0.008      0.024      0.025      0.015      0.096
##
## , , age_grp = 40-44
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.547      1.283      1.266      2.200      0.009
##   Islands         1.001      0.000      1.090      1.445      0.367      0.008
##   North-East      1.322      1.563      0.000      1.417      1.626      0.007
##   North-West      1.234      2.353      1.261      0.000      1.885      0.012
##   South           1.317      0.354      1.044      1.001      0.000      0.017
##   Sum             0.013      0.005      0.014      0.014      0.009      0.054
##
## , , age_grp = 45-49
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.638      1.130      1.204      2.331      0.005
##   Islands         1.076      0.000      1.100      1.372      0.400      0.005
##   North-East      1.406      1.701      0.000      1.501      1.607      0.005
##   North-West      1.320      2.600      1.354      0.000      2.007      0.008
##   South           1.286      0.408      0.919      0.912      0.000      0.010
##   Sum             0.008      0.003      0.008      0.008      0.006      0.033
##
## , , age_grp = 50-54
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.887      1.064      1.110      2.579      0.005
##   Islands         0.997      0.000      0.861      1.226      0.361      0.004
##   North-East      1.449      1.709      0.000      1.505      1.541      0.004
##   North-West      1.519      3.174      1.595      0.000      2.391      0.008
##   South           1.267      0.366      0.738      0.831      0.000      0.007
##   Sum             0.007      0.003      0.006      0.006      0.005      0.028
##
## , , age_grp = 55-59
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      2.263      1.084      1.029      2.894      0.004
##   Islands         0.845      0.000      0.643      1.027      0.343      0.003
##   North-East      1.448      1.641      0.000      1.455      1.391      0.003
##   North-West      1.724      3.929      1.892      0.000      2.921      0.008
##   South           1.160      0.398      0.544      0.722      0.000      0.005
##   Sum             0.006      0.003      0.005      0.005      0.005      0.023
```

```
##
## , , age_grp = 60-64
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      2.271      1.067      1.084      3.282      0.004
##   Islands         0.767      0.000      0.397      0.933      0.414      0.002
##   North-East      1.331      1.473      0.000      1.578      1.500      0.003
##   North-West      1.633      4.038      1.938      0.000      3.047      0.008
##   South           1.245      0.395      0.444      0.734      0.000      0.005
##   Sum             0.005      0.003      0.004      0.004      0.005      0.022
##
## , , age_grp = 65-69
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      2.383      1.159      1.030      3.451      0.003
##   Islands         0.827      0.000      0.435      0.876      0.385      0.002
##   North-East      1.222      1.237      0.000      1.629      1.439      0.002
##   North-West      1.518      3.324      1.891      0.000      2.933      0.005
##   South           1.340      0.479      0.419      0.874      0.000      0.004
##   Sum             0.004      0.002      0.003      0.004      0.004      0.017
##
## , , age_grp = 70-74
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      2.409      0.999      1.353      3.269      0.003
##   Islands         0.723      0.000      0.381      1.253      0.385      0.002
##   North-East      1.301      1.200      0.000      1.765      1.113      0.002
##   North-West      1.421      2.608      1.719      0.000      2.445      0.004
##   South           1.451      0.432      0.460      1.065      0.000      0.004
##   Sum             0.004      0.001      0.003      0.004      0.003      0.014
##
## , , age_grp = 75-79
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.926      1.174      1.311      2.957      0.003
##   Islands         0.819      0.000      0.352      1.352      0.431      0.002
##   North-East      1.395      0.840      0.000      2.114      0.929      0.002
##   North-West      1.327      2.463      1.810      0.000      1.963      0.003
##   South           1.450      0.437      0.488      1.173      0.000      0.004
##   Sum             0.003      0.001      0.003      0.004      0.002      0.013
##
## , , age_grp = 80-84
```

```
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.846      1.070      1.503      2.636      0.002
##   Islands         0.804      0.000      0.428      1.295      0.519      0.001
##   North-East      1.466      0.631      0.000      2.117      0.986      0.001
##   North-West      1.232      2.001      1.825      0.000      1.826      0.002
##   South           1.571      0.408      0.493      1.258      0.000      0.003
##   Sum             0.002      0.001      0.002      0.002      0.001      0.008
##
## , , age_grp = 85-89
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.545      1.509      1.606      2.575      0.001
##   Islands         0.739      0.000      0.383      1.345      0.414      0.001
##   North-East      1.766      1.254      0.000      2.809      0.913      0.001
##   North-West      1.090      1.667      1.944      0.000      1.395      0.002
##   South           1.410      0.301      0.415      1.240      0.000      0.002
##   Sum             0.002      0.000      0.001      0.002      0.001      0.007
##
## , , age_grp = 90-94
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      1.319      1.211      1.906      2.277      0.001
##   Islands         0.809      0.000      0.418      1.033      0.359      0.000
##   North-East      1.469      1.083      0.000      2.835      0.660      0.000
##   North-West      1.494      1.635      2.216      0.000      1.778      0.001
##   South           1.452      0.250      0.387      1.142      0.000      0.001
##   Sum             0.001      0.000      0.001      0.001      0.000      0.003
##
## , , age_grp = 95+
##
##             dest
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      0.886      1.076      1.504      2.207      0.000
##   Islands         0.847      0.000      0.521      0.835      0.523      0.000
##   North-East      1.383      1.750      0.000      2.340      0.698      0.000
##   North-West      1.707      2.593      2.149      0.000      2.263      0.000
##   South           1.394      0.485      0.523      0.965      0.000      0.000
##   Sum             0.000      0.000      0.000      0.000      0.000      0.001
##
## , , age_grp = Sum
##
##             dest
```

```
## orig             Center    Islands North-East North-West      South        Sum
##   Center          0.000      0.017      0.037      0.039      0.054      0.148
##   Islands         0.038      0.000      0.048      0.067      0.013      0.166
##   North-East      0.030      0.016      0.000      0.041      0.037      0.125
##   North-West      0.045      0.037      0.056      0.000      0.063      0.202
##   South           0.120      0.013      0.111      0.115      0.000      0.360
##   Sum             0.233      0.084      0.253      0.263      0.168 277436.000
```

8.1.11 Multiplicative Component Model

```
# origin components (shares)
c0 %>%
  as.data.frame.table(responseName = "comp") %>%
  filter(orig != "Sum", dest == "Sum", age_grp == "Sum")
```

```
##         orig dest age_grp      comp
## 1     Center  Sum     Sum 0.1477314
## 2    Islands  Sum     Sum 0.1663483
## 3 North-East  Sum     Sum 0.1245945
## 4 North-West  Sum     Sum 0.2017835
## 5      South  Sum     Sum 0.3595424
```

```
# destination components (shares)
c0 %>%
  as.data.frame.table(responseName = "comp") %>%
  filter(orig == "Sum", dest != "Sum", age_grp == "Sum")
```

```
##   orig       dest age_grp       comp
## 1  Sum     Center     Sum 0.23305555
## 2  Sum    Islands     Sum 0.08368777
## 3  Sum North-East     Sum 0.25254113
## 4  Sum North-West     Sum 0.26283900
## 5  Sum      South     Sum 0.16787656
```

8.1.12 Multiplicative Component Model

```
# age components
c0 %>%
  as.data.frame.table(responseName = "comp") %>%
  filter(orig == "Sum", dest == "Sum", age_grp != "Sum") %>%
```

```
ggplot(mapping = aes(x = age_grp, y = comp, group = 1)) +
geom_line() +
theme(axis.text = element_text(angle = 45, hjust = 1))
```

8.2 Log-Linear Models

8.2.1 Log-Linear Models

- Rogers' and collaborators like to shorten the multiplicative form of the log-linear model to use capital letters to represent parameters

$$m_{ij} = \gamma\alpha_i\beta_j\delta_{ij} = TO_iD_jOD_{ij}$$

- When there is multiple origin-destination tables, by different age groups, sex, education level, etc,... the notation can be easily used to study different log-linear models

$$m_{ij} = TO_iD_jA_xOD_{ij}OA_{ix}$$

- When data is in a tidy format with row h would be:

$$\begin{aligned}\log y_h =& \beta_0 + \beta_1 ORIG_h + \beta_2 DEST_h + \beta_3 AGE_x + \\ & \beta_4 ORIG_h : DEST_h + \beta_5 ORIG_h : AGE_h\end{aligned}$$

8.2.2 Log-Linear Models

- We can fit log-linear models in R using the `glm()` function (for generalised linear models)
- Requires a `formula`, `data` and `family` argument
- The `formula` argument is similar to that in `xtabs()`, where we use the `~` symbol to separate the the response and explanatory variables
  - For example the model in the previous slide would use `formula = flow ~ orig + dest + age + orig:dest + orig:age`
  - Use `:` or `*` to denote interactions
- The `family` argument should be set to `poisson()` for a log-linear model

8.2.3 Log-Linear Models

- Example with age-specific migration flows between Italian regions in 1970

```
d1 <- italy_area %>%
  filter(orig != dest,
         year == 1970) %>%
  # rename so later output fits on slide
  rename(age = age_grp)
d1
```

```
## # A tibble: 400 x 5
##    orig       dest        year age    flow
##    <chr>      <chr>      <dbl> <fct> <dbl>
##  1 North-East North-West  1970 0-4    2350
##  2 Center     North-West  1970 0-4    1687
##  3 South      North-West  1970 0-4    9697
##  4 Islands    North-West  1970 0-4    5139
##  5 North-West North-East  1970 0-4    2448
##  6 Center     North-East  1970 0-4    1063
##  7 South      North-East  1970 0-4    1560
##  8 Islands    North-East  1970 0-4     689
##  9 North-West Center      1970 0-4    2097
## 10 North-East Center      1970 0-4    1183
## # ... with 390 more rows
```

8.2.4 Log-Linear Models

```
glm(formula = flow ~ orig + dest, family = poisson(), data = d1)
```

```
##
## Call:  glm(formula = flow ~ orig + dest, family = poisson(), data = d1)
##
## Coefficients:
##    (Intercept)     origIslands  origNorth-East  origNorth-West       origSouth
##        6.39791         0.17515        -0.20852         0.99427         0.98847
##    destIslands  destNorth-East  destNorth-West       destSouth
##       -0.76940        -0.32536         1.08367         0.02188
##
## Degrees of Freedom: 399 Total (i.e. Null);  391 Residual
## Null Deviance:       758100
## Residual Deviance: 5e+05     AIC: 503100
```

8.3 Dimensions

8.3.1 Log-Linear Model Analysis

- As we increase the number of dimensions of the data, it might become important to understand which dimensions of the data are most important
- We can use log-linear models with detailled migration data to
  - Understand the dominate dimensions, for example van Imhoff et al. [1997] Rogers et al. [2002]
  - Predict origin-destination flows with partial data, for example de Beer et al. [2010] Rogers et al. [2003] Raymer [2007]
  - Project detailed origin-destination flows, for example Raymer et al. [2006]
- All the above examples involve fitting a number log-linear models based on different dimensions of the data frames
  - Use model fit statistics to judge the best model

8.3.2 Log-Linear Model Analysis

- One approach to choosing the most important dimensions is to fit all possible combinations of models - known as *dredging* the model space
- The `dredge()` function in the *MuMIn* package will fit all combinations of regression models given an upper limit, i.e. the most complex model.
  - The number of combinations grows exponentially with the number of predictors

  - Does not allow `na.action = "na.omit"` - set by default in `glm()` for handling missing values in regression models

8.3.3 Log-Linear Model Analysis

- Fit the most complex model using `glm()`.
  - Set `na.action = na.fail` to exclude failed models in when using the `dredge()` function later
  - Most complex model typically involves at least all two-way interactions
- The `formula` argument in `glm()` allows the use `()^2` to construct all two-way interactions, i.e. the below give the identical outputs
  - Use `()^3` for all three way interactions

```
f1 <- glm(formula = flow ~ (orig + dest + age)^2,
          family = poisson(), data = d1, na.action = na.fail)
f2 <- glm(formula = flow ~ orig * dest + orig * age + dest * age,
          family = poisson(), data = d1, na.action = na.fail)

# check terms used in models
attr(f1$terms, "term.labels")
```

```
## [1] "orig"      "dest"      "age"       "orig:dest" "orig:age"  "dest:age"
```

```
attr(f2$terms, "term.labels")
```

```
## [1] "orig"      "dest"      "age"       "orig:dest" "orig:age"  "dest:age"
```

8.3.4 Log-Linear Model Analysis

- Models will have many estimated coefficients
  - Some will be non-determinable because no observations (e.g. diagonal terms such as `origIslands:destIslands` below) as

```
f1 %>%
  coef() %>%
  length()
```

```
## [1] 196
```

```
summary(f1)
```

```
##
## Call:
## glm(formula = flow ~ (orig + dest + age)^2, family = poisson(),
##     data = d1, na.action = na.fail)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -12.1125   -1.4474    0.0186    1.5870    8.3143
##
## Coefficients: (5 not defined because of singularities)
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    6.812e+00  2.085e-02 326.747  < 2e-16 ***
## origIslands                    4.277e-01  2.236e-02  19.126  < 2e-16 ***
## origNorth-East                 1.709e-01  2.390e-02   7.151 8.64e-13 ***
## origNorth-West                 7.906e-01  1.789e-02  44.190  < 2e-16 ***
## origSouth                      1.381e+00  2.027e-02  68.123  < 2e-16 ***
## destIslands                   -1.548e-01  2.457e-02  -6.301 2.95e-10 ***
## destNorth-East                 8.107e-02  2.232e-02   3.632 0.000281 ***
## destNorth-West                 6.751e-01  1.856e-02  36.367  < 2e-16 ***
## destSouth                      8.315e-01  1.833e-02  45.350  < 2e-16 ***
## age5-9                        -1.073e-01  2.726e-02  -3.935 8.32e-05 ***
## age10-14                      -5.531e-01  3.071e-02 -18.012  < 2e-16 ***
## age15-19                      -6.345e-01  2.921e-02 -21.725  < 2e-16 ***
## age20-24                       5.581e-01  2.340e-02  23.850  < 2e-16 ***
## age25-29                       6.591e-01  2.378e-02  27.715  < 2e-16 ***
## age30-34                       4.155e-01  2.550e-02  16.296  < 2e-16 ***
## age35-39                       3.452e-02  2.846e-02   1.213 0.225246
## age40-44                      -2.132e-01  3.082e-02  -6.916 4.65e-12 ***
## age45-49                      -4.331e-01  3.333e-02 -12.995  < 2e-16 ***
## age50-54                      -8.007e-01  3.911e-02 -20.475  < 2e-16 ***
## age55-59                      -7.723e-01  3.884e-02 -19.884  < 2e-16 ***
## age60-64                      -8.835e-01  4.067e-02 -21.721  < 2e-16 ***
## age65-69                      -1.017e+00  4.508e-02 -22.565  < 2e-16 ***
## age70-74                      -1.284e+00  5.087e-02 -25.242  < 2e-16 ***
## age75-79                      -1.602e+00  6.015e-02 -26.628  < 2e-16 ***
## age80-84                      -2.099e+00  7.796e-02 -26.932  < 2e-16 ***
## age85-89                      -2.798e+00  1.164e-01 -24.039  < 2e-16 ***
## age90-94                      -4.022e+00  2.183e-01 -18.422  < 2e-16 ***
## age95+                        -3.872e+00  1.344e-01 -28.816  < 2e-16 ***
## origIslands:destIslands               NA         NA      NA       NA
## origNorth-East:destIslands    -7.535e-01  2.394e-02 -31.472  < 2e-16 ***
## origNorth-West:destIslands     4.133e-01  1.590e-02  25.998  < 2e-16 ***
## origSouth:destIslands         -1.450e+00  2.079e-02 -69.771  < 2e-16 ***
```

```
## origIslands:destNorth-East    -8.276e-01  2.016e-02 -41.062  < 2e-16 ***
## origNorth-East:destNorth-East         NA         NA      NA       NA
## origNorth-West:destNorth-East  1.723e-01  1.388e-02  12.414  < 2e-16 ***
## origSouth:destNorth-East      -9.378e-01  1.701e-02 -55.130  < 2e-16 ***
## origIslands:destNorth-West     6.005e-01  1.622e-02  37.030  < 2e-16 ***
## origNorth-East:destNorth-West  1.862e-01  1.703e-02  10.934  < 2e-16 ***
## origNorth-West:destNorth-West         NA         NA      NA       NA
## origSouth:destNorth-West       2.968e-01  1.460e-02  20.336  < 2e-16 ***
## origIslands:destSouth         -1.346e+00  1.700e-02 -79.141  < 2e-16 ***
## origNorth-East:destSouth      -1.000e+00  1.672e-02 -59.838  < 2e-16 ***
## origNorth-West:destSouth              NA         NA      NA       NA
## origSouth:destSouth                   NA         NA      NA       NA
## origIslands:age5-9             3.728e-02  2.705e-02   1.379 0.168013
## origNorth-East:age5-9         -3.398e-02  2.976e-02  -1.142 0.253461
## origNorth-West:age5-9         -2.097e-02  2.469e-02  -0.849 0.395888
## origSouth:age5-9               6.147e-02  2.489e-02   2.470 0.013522 *
## origIslands:age10-14           2.034e-01  3.009e-02   6.759 1.39e-11 ***
## origNorth-East:age10-14       -8.982e-02  3.415e-02  -2.630 0.008533 **
## origNorth-West:age10-14       -3.405e-02  2.853e-02  -1.193 0.232730
## origSouth:age10-14             2.270e-01  2.788e-02   8.145 3.81e-16 ***
## origIslands:age15-19           4.832e-01  2.807e-02  17.213  < 2e-16 ***
## origNorth-East:age15-19       -1.154e-01  3.267e-02  -3.533 0.000410 ***
## origNorth-West:age15-19       -3.695e-02  2.712e-02  -1.363 0.173004
## origSouth:age15-19             5.775e-01  2.617e-02  22.066  < 2e-16 ***
## origIslands:age20-24           6.953e-02  2.297e-02   3.027 0.002470 **
## origNorth-East:age20-24       -7.223e-02  2.531e-02  -2.853 0.004327 **
## origNorth-West:age20-24       -3.782e-01  2.141e-02 -17.664  < 2e-16 ***
## origSouth:age20-24             1.541e-01  2.112e-02   7.296 2.96e-13 ***
## origIslands:age25-29          -2.259e-01  2.368e-02  -9.536  < 2e-16 ***
## origNorth-East:age25-29       -4.052e-02  2.548e-02  -1.590 0.111742
## origNorth-West:age25-29       -3.300e-01  2.148e-02 -15.358  < 2e-16 ***
## origSouth:age25-29            -2.484e-01  2.170e-02 -11.445  < 2e-16 ***
## origIslands:age30-34          -3.437e-01  2.586e-02 -13.293  < 2e-16 ***
## origNorth-East:age30-34       -1.532e-02  2.737e-02  -0.560 0.575692
## origNorth-West:age30-34       -2.374e-01  2.295e-02 -10.342  < 2e-16 ***
## origSouth:age30-34            -3.716e-01  2.357e-02 -15.767  < 2e-16 ***
## origIslands:age35-39          -3.622e-01  2.905e-02 -12.467  < 2e-16 ***
## origNorth-East:age35-39       -6.987e-02  3.080e-02  -2.268 0.023322 *
## origNorth-West:age35-39       -2.315e-01  2.580e-02  -8.975  < 2e-16 ***
## origSouth:age35-39            -3.976e-01  2.642e-02 -15.053  < 2e-16 ***
## origIslands:age40-44          -3.607e-01  3.147e-02 -11.462  < 2e-16 ***
## origNorth-East:age40-44       -9.838e-02  3.354e-02  -2.933 0.003353 **
## origNorth-West:age40-44       -2.872e-01  2.835e-02 -10.130  < 2e-16 ***
## origSouth:age40-44            -3.450e-01  2.847e-02 -12.120  < 2e-16 ***
## origIslands:age45-49          -4.171e-01  3.414e-02 -12.215  < 2e-16 ***
## origNorth-East:age45-49       -1.373e-01  3.628e-02  -3.784 0.000155 ***
```

```
## origNorth-West:age45-49       -3.409e-01  3.073e-02 -11.093  < 2e-16 ***
## origSouth:age45-49            -3.791e-01  3.068e-02 -12.360  < 2e-16 ***
## origIslands:age50-54          -4.729e-01  4.048e-02 -11.681  < 2e-16 ***
## origNorth-East:age50-54       -1.693e-01  4.272e-02  -3.963 7.41e-05 ***
## origNorth-West:age50-54       -4.082e-01  3.613e-02 -11.299  < 2e-16 ***
## origSouth:age50-54            -4.723e-01  3.625e-02 -13.028  < 2e-16 ***
## origIslands:age55-59          -6.000e-01  4.066e-02 -14.756  < 2e-16 ***
## origNorth-East:age55-59       -1.545e-01  4.206e-02  -3.673 0.000239 ***
## origNorth-West:age55-59       -2.769e-01  3.496e-02  -7.920 2.38e-15 ***
## origSouth:age55-59            -6.989e-01  3.652e-02 -19.140  < 2e-16 ***
## origIslands:age60-64          -7.392e-01  4.388e-02 -16.847  < 2e-16 ***
## origNorth-East:age60-64       -1.382e-01  4.412e-02  -3.133 0.001731 **
## origNorth-West:age60-64       -1.846e-01  3.600e-02  -5.128 2.93e-07 ***
## origSouth:age60-64            -7.990e-01  3.874e-02 -20.624  < 2e-16 ***
## origIslands:age65-69          -7.861e-01  4.896e-02 -16.054  < 2e-16 ***
## origNorth-East:age65-69       -1.845e-01  4.884e-02  -3.777 0.000158 ***
## origNorth-West:age65-69       -3.790e-01  4.019e-02  -9.430  < 2e-16 ***
## origSouth:age65-69            -8.881e-01  4.337e-02 -20.475  < 2e-16 ***
## origIslands:age70-74          -7.729e-01  5.541e-02 -13.948  < 2e-16 ***
## origNorth-East:age70-74       -1.764e-01  5.511e-02  -3.200 0.001375 **
## origNorth-West:age70-74       -4.427e-01  4.583e-02  -9.659  < 2e-16 ***
## origSouth:age70-74            -8.297e-01  4.879e-02 -17.006  < 2e-16 ***
## origIslands:age75-79          -7.637e-01  6.539e-02 -11.680  < 2e-16 ***
## origNorth-East:age75-79       -1.946e-01  6.473e-02  -3.006 0.002648 **
## origNorth-West:age75-79       -6.148e-01  5.490e-02 -11.197  < 2e-16 ***
## origSouth:age75-79            -8.376e-01  5.775e-02 -14.503  < 2e-16 ***
## origIslands:age80-84          -8.643e-01  8.598e-02 -10.052  < 2e-16 ***
## origNorth-East:age80-84       -2.304e-01  8.370e-02  -2.752 0.005924 **
## origNorth-West:age80-84       -6.198e-01  7.212e-02  -8.594  < 2e-16 ***
## origSouth:age80-84            -9.215e-01  7.535e-02 -12.229  < 2e-16 ***
## origIslands:age85-89          -9.459e-01  1.281e-01  -7.382 1.56e-13 ***
## origNorth-East:age85-89       -3.505e-01  1.260e-01  -2.783 0.005392 **
## origNorth-West:age85-89       -7.537e-01  1.086e-01  -6.941 3.91e-12 ***
## origSouth:age85-89            -1.126e+00  1.145e-01  -9.832  < 2e-16 ***
## origIslands:age90-94          -7.778e-01  2.418e-01  -3.217 0.001296 **
## origNorth-East:age90-94       -2.249e-01  2.404e-01  -0.935 0.349581
## origNorth-West:age90-94       -6.822e-01  2.080e-01  -3.279 0.001041 **
## origSouth:age90-94            -9.923e-01  2.186e-01  -4.539 5.64e-06 ***
## origIslands:age95+            -1.078e-01  1.304e-01  -0.826 0.408661
## origNorth-East:age95+         -1.944e-01  1.469e-01  -1.324 0.185566
## origNorth-West:age95+         -2.803e-01  1.106e-01  -2.533 0.011308 *
## origSouth:age95+              -2.948e-01  1.218e-01  -2.420 0.015542 *
## destIslands:age5-9            -1.202e-01  2.898e-02  -4.149 3.33e-05 ***
## destNorth-East:age5-9          1.725e-02  2.578e-02   0.669 0.503388
## destNorth-West:age5-9          7.266e-03  1.974e-02   0.368 0.712823
## destSouth:age5-9              -1.636e-01  2.533e-02  -6.457 1.07e-10 ***
```

```
## destIslands:age10-14          -1.703e-01  3.301e-02  -5.159 2.48e-07 ***
## destNorth-East:age10-14        1.697e-02  2.858e-02   0.594 0.552693
## destNorth-West:age10-14        9.830e-02  2.152e-02   4.567 4.94e-06 ***
## destSouth:age10-14            -2.812e-01  2.911e-02  -9.660  < 2e-16 ***
## destIslands:age15-19           1.533e-01  3.050e-02   5.026 5.00e-07 ***
## destNorth-East:age15-19        1.060e-01  2.717e-02   3.903 9.52e-05 ***
## destNorth-West:age15-19        3.282e-01  2.014e-02  16.292  < 2e-16 ***
## destSouth:age15-19             3.606e-02  2.733e-02   1.319 0.187003
## destIslands:age20-24           3.125e-02  2.500e-02   1.250 0.211310
## destNorth-East:age20-24        1.057e-01  2.238e-02   4.723 2.33e-06 ***
## destNorth-West:age20-24        9.761e-02  1.695e-02   5.757 8.55e-09 ***
## destSouth:age20-24            -1.516e-01  2.221e-02  -6.828 8.63e-12 ***
## destIslands:age25-29          -1.739e-01  2.569e-02  -6.767 1.31e-11 ***
## destNorth-East:age25-29        1.137e-02  2.299e-02   0.494 0.620982
## destNorth-West:age25-29       -8.689e-02  1.763e-02  -4.928 8.32e-07 ***
## destSouth:age25-29            -2.597e-01  2.245e-02 -11.568  < 2e-16 ***
## destIslands:age30-34          -3.411e-01  2.782e-02 -12.263  < 2e-16 ***
## destNorth-East:age30-34       -7.689e-03  2.445e-02  -0.314 0.753161
## destNorth-West:age30-34       -2.411e-01  1.911e-02 -12.617  < 2e-16 ***
## destSouth:age30-34            -3.476e-01  2.397e-02 -14.500  < 2e-16 ***
## destIslands:age35-39          -4.423e-01  3.165e-02 -13.973  < 2e-16 ***
## destNorth-East:age35-39       -1.997e-02  2.713e-02  -0.736 0.461614
## destNorth-West:age35-39       -2.643e-01  2.137e-02 -12.367  < 2e-16 ***
## destSouth:age35-39            -4.538e-01  2.710e-02 -16.746  < 2e-16 ***
## destIslands:age40-44          -5.245e-01  3.517e-02 -14.913  < 2e-16 ***
## destNorth-East:age40-44       -1.014e-02  2.922e-02  -0.347 0.728693
## destNorth-West:age40-44       -2.404e-01  2.293e-02 -10.483  < 2e-16 ***
## destSouth:age40-44            -5.495e-01  3.000e-02 -18.315  < 2e-16 ***
## destIslands:age45-49          -4.948e-01  3.830e-02 -12.919  < 2e-16 ***
## destNorth-East:age45-49        9.649e-03  3.171e-02   0.304 0.760938
## destNorth-West:age45-49       -2.253e-01  2.498e-02  -9.021  < 2e-16 ***
## destSouth:age45-49            -5.734e-01  3.300e-02 -17.373  < 2e-16 ***
## destIslands:age50-54          -4.942e-01  4.522e-02 -10.927  < 2e-16 ***
## destNorth-East:age50-54        4.565e-06  3.725e-02   0.000 0.999902
## destNorth-West:age50-54       -3.163e-01  2.972e-02 -10.640  < 2e-16 ***
## destSouth:age50-54            -6.328e-01  3.937e-02 -16.074  < 2e-16 ***
## destIslands:age55-59          -5.805e-01  4.541e-02 -12.783  < 2e-16 ***
## destNorth-East:age55-59        9.658e-02  3.664e-02   2.636 0.008385 **
## destNorth-West:age55-59       -2.985e-01  3.060e-02  -9.756  < 2e-16 ***
## destSouth:age55-59            -6.443e-01  3.864e-02 -16.674  < 2e-16 ***
## destIslands:age60-64          -6.453e-01  4.780e-02 -13.500  < 2e-16 ***
## destNorth-East:age60-64        1.870e-01  3.768e-02   4.962 6.99e-07 ***
## destNorth-West:age60-64       -3.416e-01  3.289e-02 -10.386  < 2e-16 ***
## destSouth:age60-64            -6.588e-01  4.018e-02 -16.396  < 2e-16 ***
## destIslands:age65-69          -6.577e-01  5.361e-02 -12.268  < 2e-16 ***
## destNorth-East:age65-69        8.760e-02  4.258e-02   2.057 0.039641 *
```

```
## destNorth-West:age65-69       -4.820e-01  3.682e-02 -13.092  < 2e-16 ***
## destSouth:age65-69            -7.019e-01  4.519e-02 -15.533  < 2e-16 ***
## destIslands:age70-74          -7.064e-01  6.179e-02 -11.431  < 2e-16 ***
## destNorth-East:age70-74        5.675e-02  4.816e-02   1.178 0.238672
## destNorth-West:age70-74       -5.134e-01  4.125e-02 -12.447  < 2e-16 ***
## destSouth:age70-74            -7.024e-01  5.130e-02 -13.692  < 2e-16 ***
## destIslands:age75-79          -6.143e-01  7.225e-02  -8.503  < 2e-16 ***
## destNorth-East:age75-79       -2.785e-02  5.832e-02  -0.477 0.633038
## destNorth-West:age75-79       -5.902e-01  4.883e-02 -12.085  < 2e-16 ***
## destSouth:age75-79            -6.906e-01  6.135e-02 -11.257  < 2e-16 ***
## destIslands:age80-84          -7.780e-01  9.764e-02  -7.968 1.61e-15 ***
## destNorth-East:age80-84       -1.356e-01  7.696e-02  -1.762 0.078011 .
## destNorth-West:age80-84       -5.625e-01  6.335e-02  -8.879  < 2e-16 ***
## destSouth:age80-84            -7.880e-01  8.063e-02  -9.773  < 2e-16 ***
## destIslands:age85-89          -9.642e-01  1.550e-01  -6.222 4.89e-10 ***
## destNorth-East:age85-89       -1.953e-01  1.167e-01  -1.674 0.094131 .
## destNorth-West:age85-89       -6.426e-01  9.674e-02  -6.642 3.09e-11 ***
## destSouth:age85-89            -8.860e-01  1.227e-01  -7.220 5.20e-13 ***
## destIslands:age90-94          -1.047e+00  2.834e-01  -3.696 0.000219 ***
## destNorth-East:age90-94       -2.522e-01  2.087e-01  -1.209 0.226824
## destNorth-West:age90-94       -8.760e-01  1.758e-01  -4.982 6.29e-07 ***
## destSouth:age90-94            -1.128e+00  2.330e-01  -4.843 1.28e-06 ***
## destIslands:age95+             2.274e-01  1.434e-01   1.586 0.112675
## destNorth-East:age95+          4.667e-01  1.248e-01   3.740 0.000184 ***
## destNorth-West:age95+         -2.613e-01  1.108e-01  -2.359 0.018329 *
## destSouth:age95+               2.250e-01  1.262e-01   1.783 0.074616 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 758059.9  on 399  degrees of freedom
## Residual deviance:   2767.8  on 209  degrees of freedom
## AIC: 6272
##
## Number of Fisher Scoring iterations: 4
```

8.3.5 Log-Linear Model Analysis

- Pass the upper model to `dredge()`. Use `trace = TRUE` to monitor progress.

```
library(MuMIn)
mm <- dredge(global.model = f1, trace = TRUE)
```

```
## Fixed term is "(Intercept)"

## 0 : glm(formula = flow ~ 1, family = poisson(), data = d1, na.action = na.fail)
## 1 : glm(formula = flow ~ age + 1, family = poisson(), data = d1,
##     na.action = na.fail)
## 2 : glm(formula = flow ~ dest + 1, family = poisson(), data = d1,
##     na.action = na.fail)
## 3 : glm(formula = flow ~ age + dest + 1, family = poisson(), data = d1,
##     na.action = na.fail)
## 4 : glm(formula = flow ~ orig + 1, family = poisson(), data = d1,
##     na.action = na.fail)
## 5 : glm(formula = flow ~ age + orig + 1, family = poisson(), data = d1,
##     na.action = na.fail)
## 6 : glm(formula = flow ~ dest + orig + 1, family = poisson(), data = d1,
##     na.action = na.fail)
## 7 : glm(formula = flow ~ age + dest + orig + 1, family = poisson(),
##     data = d1, na.action = na.fail)
## 11 : glm(formula = flow ~ age + dest + age:dest + 1, family = poisson(),
##     data = d1, na.action = na.fail)
## 15 : glm(formula = flow ~ age + dest + orig + age:dest + 1, family = poisson(),
##     data = d1, na.action = na.fail)
## 21 : glm(formula = flow ~ age + orig + age:orig + 1, family = poisson(),
##     data = d1, na.action = na.fail)
## 23 : glm(formula = flow ~ age + dest + orig + age:orig + 1, family = poisson(),
##     data = d1, na.action = na.fail)
## 31 : glm(formula = flow ~ age + dest + orig + age:dest + age:orig +
##     1, family = poisson(), data = d1, na.action = na.fail)
## 38 : glm(formula = flow ~ dest + orig + dest:orig + 1, family = poisson(),
##     data = d1, na.action = na.fail)
## 39 : glm(formula = flow ~ age + dest + orig + dest:orig + 1, family = poisson(),
##     data = d1, na.action = na.fail)
## 47 : glm(formula = flow ~ age + dest + orig + age:dest + dest:orig +
##     1, family = poisson(), data = d1, na.action = na.fail)
## 55 : glm(formula = flow ~ age + dest + orig + age:orig + dest:orig +
##     1, family = poisson(), data = d1, na.action = na.fail)
## 63 : glm(formula = flow ~ age + dest + orig + age:dest + age:orig +
##     dest:orig + 1, family = poisson(), data = d1, na.action = na.fail)
```

8.3.6 Log-Linear Model Analysis

```
mm
```

```
## Global model call: glm(formula = flow ~ (orig + dest + age)^2, family = poisson(),
```

```
##     data = d1, na.action = na.fail)
## ---
## Model selection table
##    (Int) age dst org age:dst age:org dst:org  df      logLik     AICc     delta
## 64 6.515   +   +   +       +       +       + 191   -2944.992   6624.6      0.00
## 56 6.616   +   +   +               +       + 115   -5286.311  10896.6   4271.97
## 48 6.619   +   +   +       +               + 115   -7617.005  15558.0   8933.35
## 40 6.691   +   +   +                       +  39  -11408.460  22903.6  16278.99
## 32 6.865   +   +   +       +       +         180  -22817.598  46292.7  39668.13
## 24 6.995   +   +   +               +         104  -25545.324  51372.7  44748.08
## 16 6.997   +   +   +       +                 104  -27876.018  56034.1  49409.47
## 8  7.070   +   +   +                          28  -31667.473  63395.3  56770.72
## 12 7.612   +   +           +                 100  -82409.496 165086.5 158461.95
## 4  7.684   +   +                              24  -86200.951 172453.1 165828.50
## 22 7.250   +       +               +         100 -114058.016 228383.6 221758.99
## 6  7.325   +       +                          24 -120180.165 240411.5 233786.93
## 2  7.734   +                                  20 -160715.461 321473.1 314848.54
## 39 6.019       +   +                       +  20 -231284.060 462610.3 455985.74
## 7  6.398       +   +                           9 -251543.073 503104.6 496480.01
## 3  7.012       +                               5 -306076.551 612163.3 605538.65
## 5  6.653           +                           5 -340055.765 680121.7 673497.08
## 1  7.062                                       1 -380591.061 761184.1 754559.53
##    weight
## 64      1
## 56      0
## 48      0
## 40      0
## 32      0
## 24      0
## 16      0
## 8       0
## 12      0
## 4       0
## 22      0
## 6       0
## 2       0
## 39      0
## 7       0
## 3       0
## 5       0
## 1       0
## Models ranked by AICc(x)
```

8.3.7 Log-Linear Model Analysis

- Model comparison based on model statistics measuring the goodness of fit.
  - AIC measures a goodness of fit with a penalty for the number of predictor variables.
  - AICc has a bias correction term for small samples
- Typically the origin-destination interaction term is very important for accurately predicting the age-specific origin-destination migration flows
- The time to conduct a dredging analysis increase exponentially as the number of dimensions increases.

8.3.8 Exercise (ex6.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(migest)
library(MuMIn)
##
##
##
# 1. Run the code below to read in the migration flow data for flows within the
#    USA, decomposed by move type, from 6 censuses between 1940 and 2000.
us <- read_csv("./data/us_area_1940_2000.csv")
us
# 2. Show the multiplicative components, rounded to 3 digits, for the flows from
#    the 2000 census
us %>%
  filter(year == 2000) %>%
  #####() %>%
  round(digits = #####)
# 3. Fit a log-linear model to the entire data set using all two-way
#    interactions between the four dimensions (orig, dest, period and move_type)
f <- glm(formula = flow ~ (##### + dest + ##### + move_type) ^#####,
         family = #####(), data = us, na.action = na.fail)
# 4. View a summary of the model
summary(#####)
# 5. Use dredge() to fit all simpler models than the model saved in f
mm <- #####(global.model = f, trace = TRUE)
```

```
# 6. Use the View() function to inspect the results of the dredging of the model
#    space and identify the most important dimensions
View(mm)
```

Chapter 9

Estimating Bilateral Migration

9.1 IPFP

9.1.1 IPFP

- A common problem with bilateral migration data is that it is unavailable or outdated.
  - Often collected in censuses
- In some cases there are other data sources available that provide information on the in- and out-migration totals
  - Population registers

Origin

Destination

Origin

Destination

A

B

C

D

Sum

A

B

C

D

Sum

A

100

30

70

200

A

250

B

50

45

5

100

B

75

C

60

35

40

135

C

125

D

20

25

20

65

D

150

Sum

130

160

95

115

500

Sum

150

200

50

200

600

9.1.2 IPFP

- This provides a data estimation challenge, where the marginal tables totals are known but the cell values are known.
- Similar data estimation challenges exist for more detailed migration flow tables, for example:
  - In- and out-migration totals by age in each region are known, but the origin-destination migration flow table for each age group is missing.
  - Required by multi-regional cohort-component models
  - Estimating international migration flows from stocks (see for example Abel [2013])

9.1.3 IPFP

- A popular approach to estimate values in a contingency table based on known marginal tables and an initial contingency table is the Iterative Proportional Fitting Procedure (IPFP).
- First described by Deming and Stephan [1940], the IPFP has since been widely studied in a number of different disciplines and under a number of synonyms such as raking, matrix scaling or the RAS algorithm
  - Lovelace et al. [2015] gives a good overview of the application of IPFP in social sciences.

  - Lomax and Norman [2016] for another overview more specific to demography.

- Mathematical approach to iteratively adjust a *seed* contingency table $\mu_{ij}^{(0)} = m_{ij}$ to known row and column totals (n_{i+} and n_{+j})

$$\mu_{ij}^{(t+1)} = \frac{\mu_{ij}^{(t)}}{\mu_{i+}^{(t)}} n_{i+} \qquad \mu_{ij}^{(t+2)} = \frac{\mu_{ij}^{(t+1)}}{\mu_{+j}^{(t+1)}} n_{+j}$$

9.1.4 IPFP

Origin

Destination

Origin

Destination

A

B

C

D

Sum

A

B

C

D

Sum

A

100

30

70

200

A

102.87

11.67

135.46

250

B

50

45

5

100

B

49.03

16.73

9.25

75

C

60

35

40

135

C

43.98

25.72

55.30

125

D

20

25

20

65

D

56.99

71.41

21.60

150

Sum

130

160

95

115

500

Sum

150

200

50

200

600

9.1.5 IPFP

- Willekens [1999] calls the seed data an *auxiliary* table and notes that it should be information on a variables related to migration.
  - Typically past migration flow data
  - Distances or travel costs between the origin-destination pairs have been used where no past data exists
  - Limited testing to see which seeds work best for estimating migration
- The marginal data is then known as *primary* data.
  - Partial observations on the number of migrations

9.1.6 mipfp

- The *mipfp* package by Barthélemy and Suesse [2018] implements IPFP in R using the `Ipfp()` function
- Can be used for multi-dimensional marginal constraint problems.
- Three inputs
  - `seed` a matrix of auxiliary data to aid estimation
  - `target.list` a list of dimensions that are being targeted (see next point)
  - `target.data` a list of targets related to `target.list`
- R numbers dimension of arrays with
  - `1` row

  - 2 column
  - 3 table
  - …

- The `target.list` might involve
  - a single target, e.g. column totals `target.list = list(2)`
  - multiple targets, e.g. row and column totals `target.list = list(1, 2)`
  - sums over cells rather than margins of array, e.g. cells summed over tables `target.list = list(c(1, 2))`

9.1.7 mipfp

```
r <- LETTERS[1:4]
m0 <- matrix(data = c(0, 100, 30, 70,
                      50, 0, 45, 5,
                      60, 35, 0, 40,
                      20, 25, 20, 0),
             nrow = 4, ncol = 4, byrow = TRUE,
             dimnames = list(orig = r, dest = r))
addmargins(m0)
```

```
##      dest
## orig    A   B  C   D Sum
##   A     0 100 30  70 200
##   B    50   0 45   5 100
##   C    60  35  0  40 135
##   D    20  25 20   0  65
##   Sum 130 160 95 115 500
```

9.1.8 mipfp

```
orig_tot <- c(250, 75, 125, 150)
dest_tot <- c(150, 200, 50, 200)
names(orig_tot ) <- names(dest_tot) <- r

orig_tot
```

```
##   A   B   C   D
## 250  75 125 150
```

```r
dest_tot
```

```
##   A   B   C   D
## 150 200  50 200
```

```r
# check sums are equal
sum(orig_tot)
```

```
## [1] 600
```

```r
sum(dest_tot)
```

```
## [1] 600
```

9.1.9 mipfp

```r
library(mipfp)
Ipfp(seed = m0, target.list = list(1, 2),
     target.data = list(orig_tot, dest_tot))
```

```
##
## Call:
## Ipfp(seed = m0, target.list = list(1, 2), target.data = list(orig_tot,
##     dest_tot))
##
## Method:  ipfp - convergence:  TRUE
##
## Estimates:
##     dest
## orig        A         B        C          D
##    A  0.00000 102.87046 11.67024 135.459297
##    B 49.02778   0.00000 16.72686   9.245364
##    C 43.98433  25.72033  0.00000  55.295339
##    D 56.98789  71.40921 21.60290   0.000000
```

9.1.10 mipfp

```r
# save the result
y0 <- Ipfp(seed = m0, target.list = list(1, 2),
           target.data = list(orig_tot, dest_tot))

# view with totals
addmargins(y0$x.hat)
```

```
##      dest
## orig          A         B        C          D Sum
##   A     0.00000 102.87046 11.67024 135.459297 250
##   B    49.02778   0.00000 16.72686   9.245364  75
##   C    43.98433  25.72033  0.00000  55.295339 125
##   D    56.98789  71.40921 21.60290   0.000000 150
##   Sum 150.00000 200.00000 50.00000 200.000000 600
```

9.1.11 Three dimensions

Auxillary Data

Origin

Destination

Origin

Destination

A

B

C

D

Sum

A

B

C

D

Sum

A

80

10

55

145

A

20

20

15

55

B

30

20

0

50

B

20

25

5

50

C

50

15

10

75

C

10

20

30

60

D

5

20

10

35

D

15

5

10

30

Sum

85

115

40

65

305

Sum

45

45

55

50

195

Primary Data

Origin

Destination

A

B

C

D

Sum

A

250

B

75

C

125

D

150

Sum

150

200

50

200

600

9.1.12 IPFP More Complicated Data Situations

- The IPFP can be used for more complex data situations with more than two dimensions.
- Key to using the `mipfp()` function is setting the inputs for `target.data`.

```
library(tidyverse)
d <- expand_grid(orig = r, dest = r, sex = c("Female", "Male")) %>%
  mutate(flow = c(0, 0, 80, 20, 10, 20, 55, 15, 30, 20, 0, 0, 20, 25, 0, 5, 50, 10, 15

d
```

```
## # A tibble: 32 x 4
##    orig  dest  sex     flow
##    <chr> <chr> <chr>  <dbl>
##  1 A     A     Female     0
##  2 A     A     Male       0
##  3 A     B     Female    80
##  4 A     B     Male      20
##  5 A     C     Female    10
##  6 A     C     Male      20
##  7 A     D     Female    55
##  8 A     D     Male      15
##  9 B     A     Female    30
## 10 B     A     Male      20
## # ... with 22 more rows
```

9.1.13 Estimating Detailed Bilateral Migration

```
m1 <- xtabs(formula = flow ~ orig + dest + sex, data = d)
m1
```

```
## , , sex = Female
##
##     dest
## orig  A  B  C  D
##    A  0 80 10 55
##    B 30  0 20  0
##    C 50 15  0 10
##    D  5 20 10  0
##
## , , sex = Male
##
##     dest
## orig  A  B  C  D
##    A  0 20 20 15
##    B 20  0 25  5
##    C 10 20  0 30
##    D 15  5 10  0
```

9.1.14 mipfp

```
addmargins(m1)
```

```
## , , sex = Female
##
##      dest
## orig    A   B   C   D Sum
##   A     0  80  10  55 145
##   B    30   0  20   0  50
##   C    50  15   0  10  75
##   D     5  20  10   0  35
##   Sum  85 115  40  65 305
##
## , , sex = Male
##
##      dest
## orig    A   B   C   D Sum
##   A     0  20  20  15  55
```

```
##   B    20   0  25   5  50
##   C    10  20   0  30  60
##   D    15   5  10   0  30
##   Sum  45  45  55  50 195
##
## , , sex = Sum
##
##      dest
## orig    A   B   C   D Sum
##   A     0 100  30  70 200
##   B    50   0  45   5 100
##   C    60  35   0  40 135
##   D    20  25  20   0  65
##   Sum 130 160  95 115 500
```

9.1.15 mipfp

```
addmargins(m1)[,,sex = "Sum"]
```

```
##      dest
## orig    A   B   C   D Sum
##   A     0 100  30  70 200
##   B    50   0  45   5 100
##   C    60  35   0  40 135
##   D    20  25  20   0  65
##   Sum 130 160  95 115 500
```

9.1.16 mipfp

```
y1 <- Ipfp(seed = m1, target.list = list(1, 2),
           target.data = list(orig_tot, dest_tot))
addmargins(y1$x.hat)
```

```
## , , sex = Female
##
##      dest
## orig           A          B          C          D        Sum
##   A     0.000000  82.296369   3.890080 106.432305 192.618755
##   B    29.416668   0.000000   7.434158   0.000000  36.850826
##   C    36.653611  11.022997   0.000000  13.823835  61.500444
##   D    14.246971  57.127369  10.801451   0.000000  82.175792
```

```
##   Sum  80.317251 150.446736  22.125690 120.256140 373.145817
##
## , , sex = Male
##
##      dest
## orig           A          B          C          D        Sum
##   A     0.000000  20.574092   7.780161  29.026992  57.381245
##   B    19.611112   0.000000   9.292698   9.245364  38.149174
##   C     7.330722  14.697330   0.000000  41.471504  63.499556
##   D    42.740914  14.281842  10.801451   0.000000  67.824208
##   Sum  69.682749  49.553264  27.874310  79.743860 226.854183
##
## , , sex = Sum
##
##      dest
## orig           A          B          C          D        Sum
##   A     0.000000 102.870462  11.670241 135.459297 250.000000
##   B    49.027781   0.000000  16.726856   9.245364  75.000000
##   C    43.984334  25.720327   0.000000  55.295339 125.000000
##   D    56.987886  71.409211  21.602903   0.000000 150.000000
##   Sum 150.000000 200.000000  50.000000 200.000000 600.000000
```

9.1.17 mipfp

```
y1$x.hat %>%
  as.data.frame.table(responseName = "est") %>%
  as_tibble()
```

```
## # A tibble: 32 x 4
##    orig  dest  sex      est
##    <fct> <fct> <fct>  <dbl>
##  1 A     A     Female  0
##  2 B     A     Female 29.4
##  3 C     A     Female 36.7
##  4 D     A     Female 14.2
##  5 A     B     Female 82.3
##  6 B     B     Female  0
##  7 C     B     Female 11.0
##  8 D     B     Female 57.1
##  9 A     C     Female  3.89
## 10 B     C     Female  7.43
## # ... with 22 more rows
```

9.2 Net Constraints

9.2.1 Net constrained origin-destination flows

- Plane [1981] developed a proportional adjustment algorithm for estimating bilateral migration flows to match both
  - Constraints on the net migration of each region
  - Total sum of the bilateral migration flows
- Requires knowledge of
  - Past bilateral migration flows
  - Current (target) total migration flows (over whole system)
  - Current (target) net migration flows
  - Distance matrix to correspond
- No application of this method in R, although in *migest* package the `cm_net_tot()` function provides a similar set of estimates
  - Unable to incorporate distance matrix

9.2.2 Net constrained origin-destination flows

```
addmargins(m0)
```

```
##      dest
## orig    A   B  C   D Sum
##   A     0 100 30  70 200
##   B    50   0 45   5 100
##   C    60  35  0  40 135
##   D    20  25 20   0  65
##   Sum 130 160 95 115 500
```

```
# observed net
library(migest)
sum_turnover(m0)
```

```
## # A tibble: 4 x 5
##   region in_mig out_mig  turn   net
##   <chr>   <dbl>   <dbl> <dbl> <dbl>
## 1 A         130     200   330   -70
## 2 B         160     100   260    60
## 3 C          95     135   230   -40
## 4 D         115      65   180    50
```

9.2.3 Net constrained origin-destination flows

- Estimate migration flows to match new net migration and grand total.

```
y1 <- cm_net_tot(net_tot = c(-100, 125, -75, 50), tot = 600,
                 m = m0, verbose = FALSE)
addmargins(y1$n)
```

```
##      dest
## orig          A         B        C          D       Sum
##   A     0.00000 136.22513 32.93756  79.068944 248.23163
##   B    49.88761   0.00000 42.28296   4.833488  97.00406
##   C    74.27815  50.62851  0.00000  47.977516 172.88418
##   D    24.06590  35.15032 22.66377   0.000000  81.87999
##   Sum 148.23166 222.00396 97.88429 131.879947 599.99986
```

```
sum_turnover(y1$n)
```

```
## # A tibble: 4 x 5
##   region in_mig out_mig  turn    net
##   <chr>   <dbl>   <dbl> <dbl>  <dbl>
## 1 A       148.    248.   396. -100.
## 2 B       222.     97.0  319.  125.
## 3 C        97.9   173.   271.  -75.0
## 4 D       132.     81.9  214.   50.0
```

9.2.4 Net constrained origin-destination flows

- The requirement on the total sum of the bilateral flow for the algorithm is not realistic. -Plane [1981] method not widely adpoted
  - In many countries the overall number of migrant flows, that is demographically consistent with natural population change, is typically not known.
  - If the overall number of migrant flows is known, it is typically obtained from a comprehensive population register, and thus bilateral migration or total in- and out-migration flows already exist. If it is the later, can use IPFP approaches.
- In recent years I have been working on a method that constrains only to the net migration totals.
  - Unpublished, work in progress, use at own risk
  - Method is available in the `cm_net()` function in the *migest* package

- Potential uses
  - Update bilateral migration flows from surveys or administrative data to match known demographic consistent net migration totals
  - Estimate bilateral migration flows from known net migration totals using non-migration data as a seed (if no migration flow data available)

9.2.5 Net constrained origin-destination flows

```
y2 <- cm_net(net_tot = c(-100, 125, -75, 50), m = m0, verbose = FALSE)
addmargins(y2$n)
```

```
##      dest
## orig          A         B        C          D       Sum
##   A     0.00000 124.97056 27.96585  71.121910 224.05832
##   B    40.00942   0.00000 33.56693   4.065067  77.64142
##   C    64.36422  46.92119  0.00000  43.597199 154.88260
##   D    19.68451  30.74980 18.34980   0.000000  68.78412
##   Sum 124.05815 202.64155 79.88258 118.784175 525.36645
```

```
sum_turnover(y2$n)
```

```
## # A tibble: 4 x 5
##   region in_mig out_mig  turn    net
##   <chr>   <dbl>   <dbl> <dbl>  <dbl>
## 1 A       124.    224.   348. -100.
## 2 B       203.     77.6  280.  125.
## 3 C        79.9   155.   235.  -75.0
## 4 D       119.     68.8  188.   50.0
```

9.2.6 Exercise (ex7.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(mipfp)
##
```

```
##
##
# 1. Run the code below to read in the bilateral data in uk_census_2011_tidy.csv
#    from the ONS 2011 British Census
cen11 <- read_csv("./data/uk_census_2011_tidy.csv")
cen11
# 2. Run the code below to read in the bilateral data in
#    uk_nhs_hesa_2018.csv from the British administrative data (National
#    Health Service patient records and Higher Education Statistics Authority)
nhs18 <- read_csv("./data/uk_nhs_hesa_2018_tidy.csv")
nhs18
# 3. Run the code below to create data with abbreviated region names - to make
#    it easier to view the matrices for each time period
#    Note: the census data is more detailed (orig - dest - age - sex) than the
#          administrative data (orig - dest)
cen11 <- cen11 %>%
  mutate(orig_full = orig,
         dest_full = dest,
         orig = abbreviate(orig),
         dest = abbreviate(dest)) %>%
  mutate_if(is.character, fct_inorder)
nhs18 <- nhs18 %>%
  mutate(orig_full = orig,
         dest_full = dest,
         orig = abbreviate(orig),
         dest = abbreviate(dest)) %>%
  mutate_if(is.character, fct_inorder)
cen11
nhs18
# 4. Create an origin-destination-age-sex array object c11 from census flow data
#    in cen2011
#     (Hint: use xtabs())
c11 <- xtabs(formula = ##### ~ orig + dest + ##### + sex, data = #####)
# 5. Create a origin-destination matrix object a18 from the administrative flow
#    data in nhs18
a18 <- #####(formula = flow ~ orig + #####, data = #####)
# 6. Use the colSums() and rowSums() functions to create objects a18_tot and
#    a18_out, the targets for use later on.
a18_out <- #####(a18)
a18_in <- #####(a18)
a18_out
a18_in
# 7.  Complete the code below to estimate using IPFP the
#     origin-destination-age-sex flows in 2018 based on
#     a. seed from the 2011 census
```

```
#     (Hint: c11)
#     b. target list for the dimensions of the target totals (see c.)
#     (Hint: out total are for rows and in totals are columns)
#     c. target totals based on the a18_out and a18_in objects
e18 <- Ipfp(seed = #####,
            target.list = list(1, #####),
            target.data = list(#####, a18_in))
# 8. Run the code below to show the beginning of a data frame version of
#    the 2018 origin - destination - age - sex estimates, combining the detailed
#    2011 census estimates with the in and out migration totals in the 2018
#    administrative data
e18$x.hat %>%
  as.data.frame.table(responseName = "est") %>%
  as_tibble()
# 9. Run the code below to check the row and column totals of the detailed 2018
#    estimates, summed over age and sex, matches the observed in and out totals
#    from the administrative data
# totals outflow from estimated array
rowSums(e18$x.hat)
a18_out
# totals inflow from estimated array
colsSums(e18$x.hat)
a18_in
# Bonus - run the code below and note the lack of match in estimated origin -
#  destination totals and the observed administrative flows - did not
#  use target.list = list(c(1, 2)) and target.data = list(a18) in Ipfp()
(apply(X = e18$x.hat, MARGIN = c(1, 2), FUN = sum) - a18) %>%
  round()
```

Chapter 10

Chord Diagrams for Visualising Bilateral Migration

10.1 .

10.1.1 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
- In this class will illustrate 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

10.1.2 Map Based - The Emigrants of the World, Minard 1858.

10.1.3 Flowline Maps

Source: Bureau of the Census
Geography in the News

D. Lambert

10.1.4 Criticised New York Times refugee flow map

10.1.5 Martin Grandjean's attempt to rectify

10.2 Chord Diagram

10.2.1 Chord Diagrams

- First chord diagrams introduced by Martin Krzywinski in 2007.
  - `https://www.nytimes.com/imagepages/2007/01/22/science/20070123_SCI_ILLO.html`
- 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/`

10.2.2 New York Times 2007

10.2.3 Chord Diagrams with Migration Data

- Interactive chord diagram plots introduced into rr.js library by 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/`

10.2.4 Chord Diagrams with Migration Data

10.2.5 Chord Diagrams with Migration Data

10.3 circlize

10.3.1 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 replicability

- 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.

10.3.2 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 por_name por_code pob_name           pob_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
```

10.3.3 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(por_code %in% cc,
         pob_code %in% cc,
         year == 2020)
d
```

```
## # A tibble: 36 x 6
##     year    stock por_name por_code pob_name                        pob_code
##    <dbl>    <dbl> <chr>       <dbl> <chr>                              <dbl>
##  1  2020 20917565 AFRICA        903 AFRICA                               903
##  2  2020  1207631 AFRICA        903 ASIA                                 935
##  3  2020   648455 AFRICA        903 EUROPE                               908
##  4  2020    32524 AFRICA        903 LATIN AMERICA AND THE CARIBBEAN      904
##  5  2020    53563 AFRICA        903 NORTHERN AMERICA                     905
##  6  2020    14483 AFRICA        903 OCEANIA                              909
##  7  2020  4720103 ASIA          935 AFRICA                               903
##  8  2020 68497762 ASIA          935 ASIA                                 935
##  9  2020  7169630 ASIA          935 EUROPE                               908
## 10  2020   414658 ASIA          935 LATIN AMERICA AND THE CARIBBEAN      904
## # ... with 26 more rows
```

10.3.4 UN international migrant stock data 2020

- Remove within continent stocks (will dominate the plot) and focus on inter-continent migrants

```
d <- d %>%
  rename(orig = pob_name,
         dest = por_name) %>%
  filter(orig != dest) %>%
  select(-contains("code"))
d
```

```
## # A tibble: 30 x 4
##     year   stock dest   orig
##    <dbl>   <dbl> <chr>  <chr>
##  1  2020 1207631 AFRICA ASIA
##  2  2020  648455 AFRICA EUROPE
##  3  2020   32524 AFRICA LATIN AMERICA AND THE CARIBBEAN
##  4  2020   53563 AFRICA NORTHERN AMERICA
##  5  2020   14483 AFRICA OCEANIA
##  6  2020 4720103 ASIA   AFRICA
##  7  2020 7169630 ASIA   EUROPE
##  8  2020  414658 ASIA   LATIN AMERICA AND THE CARIBBEAN
##  9  2020  538199 ASIA   NORTHERN AMERICA
## 10  2020  101725 ASIA   OCEANIA
## # ... with 20 more rows
```

10.3.5 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)
# first three columns not origin, destination, connection (in that order)
chordDiagram(x = d)
```

10.3.6 Default `chordDiagram()`

- Move the `orig`, `dest` and `stock` columns to the left of the data frame using the `relocate()` function in the *dplyr* package

```
d <- relocate(d, orig, dest, stock)
d
```

```
## # A tibble: 30 x 4
##    orig                            dest     stock  year
##    <chr>                           <chr>    <dbl> <dbl>
##  1 ASIA                            AFRICA 1207631  2020
##  2 EUROPE                          AFRICA  648455  2020
##  3 LATIN AMERICA AND THE CARIBBEAN AFRICA   32524  2020
##  4 NORTHERN AMERICA                AFRICA   53563  2020
##  5 OCEANIA                         AFRICA   14483  2020
##  6 AFRICA                          ASIA   4720103  2020
##  7 EUROPE                          ASIA   7169630  2020
##  8 LATIN AMERICA AND THE CARIBBEAN ASIA    414658  2020
##  9 NORTHERN AMERICA                ASIA    538199  2020
## 10 OCEANIA                         ASIA    101725  2020
## # ... with 20 more rows
```

```
chordDiagram(x = d)
```

```
## There are more than one numeric columns in the data frame. Take the
## first two numeric columns and draw the link ends with unequal width.
##
## Type `circos.par$message = FALSE` to suppress the message.
```

10.3.7 Default `chordDiagram()`

- Avoid chord link ends with unequal widths at each base by using only one numeric column in `d`

```
d <- select(d, orig, dest, stock)
d
```

```
## # A tibble: 30 x 3
##    orig                            dest     stock
##    <chr>                           <chr>    <dbl>
##  1 ASIA                            AFRICA 1207631
##  2 EUROPE                          AFRICA  648455
##  3 LATIN AMERICA AND THE CARIBBEAN AFRICA   32524
##  4 NORTHERN AMERICA                AFRICA   53563
##  5 OCEANIA                         AFRICA   14483
##  6 AFRICA                          ASIA   4720103
##  7 EUROPE                          ASIA   7169630
##  8 LATIN AMERICA AND THE CARIBBEAN ASIA    414658
##  9 NORTHERN AMERICA                ASIA    538199
## 10 OCEANIA                         ASIA    101725
## # ... with 20 more rows
```

```
chordDiagram(x = d)
```

10.4 Sectors

10.4.1 Sector Axis

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

```
d <- mutate(d, stock = stock/1e6)
d
```

```
## # A tibble: 30 x 3
##    orig                            dest    stock
##    <chr>                           <chr>   <dbl>
##  1 ASIA                            AFRICA 1.21
##  2 EUROPE                          AFRICA 0.648
##  3 LATIN AMERICA AND THE CARIBBEAN AFRICA 0.0325
##  4 NORTHERN AMERICA                AFRICA 0.0536
##  5 OCEANIA                         AFRICA 0.0145
##  6 AFRICA                          ASIA   4.72
##  7 EUROPE                          ASIA   7.17
##  8 LATIN AMERICA AND THE CARIBBEAN ASIA   0.415
##  9 NORTHERN AMERICA                ASIA   0.538
## 10 OCEANIA                         ASIA   0.102
## # ... with 20 more rows
```

```
chordDiagram(x = d)
```

10.4.2 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 <- tibble(reg = union(d$orig, d$dest))
r
```

```
## # A tibble: 6 x 1
##   reg
##   <chr>
## 1 ASIA
## 2 EUROPE
## 3 LATIN AMERICA AND THE CARIBBEAN
## 4 NORTHERN AMERICA
## 5 OCEANIA
## 6 AFRICA
```

10.4.3 Sector ordering

```
r <- r %>%
  mutate(reg_order = c(4, 3, 6, 1, 5, 2)) %>%
  arrange(reg_order)
r
```

```
## # A tibble: 6 x 2
##   reg                              reg_order
##   <chr>                                <dbl>
## 1 NORTHERN AMERICA                         1
## 2 AFRICA                                   2
## 3 EUROPE                                   3
## 4 ASIA                                     4
## 5 OCEANIA                                  5
## 6 LATIN AMERICA AND THE CARIBBEAN          6
```

```
# order sectors
chordDiagram(x = d, order = r$reg)
```

10.4.4 Orientation and gaps

- The `circos.par()` function controls the overall layout parameters of the graphic display

- Use `circos.par()` to set
  - `gap.degree` the degree of gaps between sectors are set - default `gap.degree = 1`
  - `start.degree` the degree from three o'clock where the first sector appears - default `start.degree = 0`

- Anything set via `circos.par()` will be fixed for all remaining pots
- Reset to default graphic parameters using `circos.clear()` or overwrite with new `circos.par()`

```
# increase gaps
circos.par(gap.degree = 5)
chordDiagram(x = d, order = r$reg)
```

```
# rotate
circos.par(start.degree = 90)
chordDiagram(x = d, order = r$reg)
```

10.5 Colour

10.5.1 Sector colours

- Colours are randomly generated (will change every time you plot)
- Can set to a choice using either:
  - `grid.col` corresponding to sectors (regions/countries/areas)
  - `transparency` set by default to 0.5

```
r <- r %>%
  mutate(col1 = c("black", "gold", "orange", "blue", "purple", "red"))
r
```

```
## # A tibble: 6 x 3
##   reg                                reg_order col1
##   <chr>                                  <dbl> <chr>
## 1 NORTHERN AMERICA                           1 black
## 2 AFRICA                                     2 gold
## 3 EUROPE                                     3 orange
## 4 ASIA                                       4 blue
## 5 OCEANIA                                    5 purple
## 6 LATIN AMERICA AND THE CARIBBEAN            6 red
```

```
chordDiagram(x = d, order = r$reg, grid.col = r$col1)
```

10.5.2 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 5
##   reg                              reg_order col1   col2    col3
##   <chr>                                <dbl> <chr>  <chr>   <chr>
## 1 NORTHERN AMERICA                         1 black  #E41A1C Red
## 2 AFRICA                                   2 gold   #377EB8 Grey
## 3 EUROPE                                   3 orange #4DAF4A Grey
## 4 ASIA                                     4 blue   #984EA3 Grey
## 5 OCEANIA                                  5 purple #FF7F00 Grey
## 6 LATIN AMERICA AND THE CARIBBEAN          6 red    #FFFF33 Grey
```

```
chordDiagram(x = d, order = r$reg, grid.col = r$col2)
```

```
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25)
```

```
chordDiagram(x = d, order = r$reg, grid.col = r$col3)
```

10.5.3 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: 30 x 5
##    orig                              dest    stock link_col1 show_link
##    <chr>                             <chr>   <dbl> <chr>     <lgl>
##  1 ASIA                              AFRICA 1.21   grey      TRUE
##  2 EUROPE                            AFRICA 0.648  grey      FALSE
```

```
##  3 LATIN AMERICA AND THE CARIBBEAN AFRICA 0.0325 grey      FALSE
##  4 NORTHERN AMERICA                AFRICA 0.0536 grey      FALSE
##  5 OCEANIA                         AFRICA 0.0145 grey      FALSE
##  6 AFRICA                          ASIA   4.72   grey      TRUE
##  7 EUROPE                          ASIA   7.17   grey      TRUE
##  8 LATIN AMERICA AND THE CARIBBEAN ASIA   0.415  grey      TRUE
##  9 NORTHERN AMERICA                ASIA   0.538  grey      TRUE
## 10 OCEANIA                         ASIA   0.102  grey      TRUE
## # ... with 20 more rows
```

10.5.4 Chord colours

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

```
chordDiagram(x = d, order = r$reg,
             grid.col = r$col2, col = d$link_col1)
```

```
chordDiagram(x = d, order = r$reg,
             grid.col = r$col2, link.visible = d$show_link)
```

10.6 Chords

10.6.1 Direction

- Distinguish direction of bilateral link using
  - Different heights at the start and end of the chord links
  - Arrows
  - Combination of both
- Set in `chordDiagram()` using
  - `directional = 1` (from link goes from first to second column)
  - `direction.type` arguments

```
# drop link_col column
d$link_col1 <- NULL

# as used by Sander, default of direction.type = "diffHeight"
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1)
```

```
# default arrows are too much
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"))
```

```
# getting there...
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow")
```

10.7 Direction

- Connect the base of the link to the sector through combination of
  - Adjusting the difference in height between the beginning and end of chords
  - Removing padding between the axis and the grid (the inner circle where the chords are)
- Set the `diffHeight` argument to a negative number so that the start of the chord is longer than then end.
  - Removes the destination sector bars (chart junk IMO).

```
# extreme height difference
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight  = -0.2)
```

```
# height difference looks good
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight  = -0.05)
```

10.7.1 Direction

- Set in the `track.margin` option of `circos.par()` to remove the padding
  - Default of `track.margin = c(0.01, 0.01)` for chord diagrams - one percent between label names and the axis, and one percent between the axis and the grid (the chords)

```
# set second margin to zero
circos.par(track.margin = c(0.01, 0))
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05)
```

```
# set second margin to -0.01 to get seamless overlap
circos.par(track.margin = c(0.01, -0.01))
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05)
```

10.7.2 Chord ordering

- Number of options in `chordDiagram()` to control the chord link order

  - `link.sort` sort the order the links from largest to smaller as the enter and exit the plot, by default `FALSE`
  - `link.largest.ontop` sort the order of the plotting of the links so that the smallest are given less prominence. By default `FALSE`, so plots the links in the last sector last and they appear more predominant

```
# sort links on sectors
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05,
             link.sort = TRUE)
```

```
# sort link plotting order
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05,
             link.sort = TRUE, link.largest.ontop = TRUE)
```

10.8 Labels

10.8.1 Labels

- Multiple options for the orientation of labels, set via
  - `inside`, `outside`, `clockwise`, `reverse.clockwise`, `downward`, `bending.inside` and `bending.outside`
  - Cannot pass to `chordDiagram()` so we have to first use `annotationTrack` option to only plot the `grid` (the chords) and `axis` (default for `annotationTrack = c("name", "grid", "axis")`)
- To add the labels we use the `panel.fun` argument in `circos.track()`.
  - Works like a for loop, cycling through each sector of the track (the circle)
  - For each sector we use `circos.text()` to add labels at a specified `x` and `y` location
  - Can also set the `facing` orientation of the labels as well as other options such as text size (`cex`) and colour (`col`)

```
# drop the name labels
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
```

```
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05,
             link.sort = TRUE, link.largest.ontop = TRUE,
             annotationTrack = c("grid", "axis"))
```

10.8.2 Labels

- No room for labels. We can create this using the `preAllocateTracks` argument.
  - Requires a list of graphical parameters
  - Set `track.height` as a percentage of plot area.

```
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05,
             link.sort = TRUE, link.largest.ontop = TRUE,
             annotationTrack = c("grid", "axis"),
             preAllocateTracks = list(track.height = 0.1))

# add labels
circos.track(track.index = 1, bg.border = NA, panel.fun = function(x, y) {
  # create temporary objects for the sector name and x-limits
```

```
  reg_lab <- get.cell.meta.data("sector.index")
  xx <- get.cell.meta.data("xlim")
  # use the temporary objects to add text in each sector of the track
  circos.text(x = mean(xx), y = 1, labels = reg_lab, facing = "bending")
})
```

10.8.3 Labels

- Still not enough room for longer labels.
  - Increase the `track.height`
  - Create two labels for some regions
  - Reduce the font size using `cex` in `circos.text()` - default is `cex = 1`

```
str_wrap(string = r$reg, width = 14)
```

```
## [1] "NORTHERN\nAMERICA"                "AFRICA"
## [3] "EUROPE"                           "ASIA"
## [5] "OCEANIA"                          "LATIN AMERICA\nAND THE\nCARIBBEAN"
```

```
r <- r %>%
  # title case for labels
  mutate(lab = str_to_title(string = reg),
         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") %>%
  # positioning for first lab1, needs to be further out if lab2 exists
  mutate(y = ifelse(test = !is.na(lab2), yes = 1, no = 0.8))
```

10.8.4 Labels

- Still not enough room for longer labels.
  - Increase the `track.height`
  - Create two labels for some regions
  - Reduce the font size using `cex` in `circos.text()` - default is `cex = 1`

```
r
```

```
## # A tibble: 6 x 8
##   reg                              reg_order col1   col2  col3  lab1  lab2      y
##   <chr>                                <dbl> <chr>  <chr> <chr> <chr> <chr> <dbl>
## 1 NORTHERN AMERICA                         1 black  #E41~ Red   Nort~ Amer~   1
## 2 AFRICA                                   2 gold   #377~ Grey  Afri~ <NA>    0.8
## 3 EUROPE                                   3 orange #4DA~ Grey  Euro~ <NA>    0.8
## 4 ASIA                                     4 blue   #984~ Grey  Asia  <NA>    0.8
## 5 OCEANIA                                  5 purple #FF7~ Grey  Ocea~ <NA>    0.8
## 6 LATIN AMERICA AND THE CARIBBEAN          6 red    #FFF~ Grey  Lati~ & Ca~   1
```

10.8.5 Labels

```
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05,
             link.sort = TRUE, link.largest.ontop = TRUE,
             annotationTrack = c("grid", "axis"),
             # increase to 0.2 to fit two lines of labels
             preAllocateTracks = list(track.height = 0.2))
```

```
circos.track(track.index = 1, bg.border = NA, panel.fun = function(x, y) {
  s <- get.cell.meta.data("sector.index")
  # filter to row of r for the sector's region to create a temporary rr
  rr <- filter(r, reg == s)
  xx <- get.cell.meta.data("xlim")
  # use temporary rr to add text
  circos.text(x = mean(xx), y = rr$y, labels = rr$lab1, facing = "bending",
              cex = 0.8)
  circos.text(x = mean(xx), y = 0.6,  labels = rr$lab2, facing = "bending",
              cex = 0.8)
})
```

10.8.6 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

```
pdf(file = "./plot/un_stock_2019.pdf", width = 4, height = 4)

circos.par(track.margin = c(0.01, -0.01), gap.degree = 5, start.degree = 90)
chordDiagram(x = d, order = r$reg, grid.col = r$col2, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.05,
             link.sort = TRUE, link.largest.ontop = TRUE,
             annotationTrack = c("grid", "axis"),
             preAllocateTracks = list(track.height = 0.2))
circos.track(track.index = 1, bg.border = NA, panel.fun = function(x, y) {
  s <- get.cell.meta.data("sector.index")
  rr <- filter(r, reg == s)
  xx <- get.cell.meta.data("xlim")
  circos.text(x = mean(xx), y = rr$y, labels = rr$lab1, facing = "bending", cex = 0.8)
  circos.text(x = mean(xx), y = 0.6,  labels = rr$lab2, facing = "bending", cex = 0.8)
})

dev.off()
```

10.8.7 Saving

- Left: PNG with `width = 4, height = 4`
- Right: PDF with `width = 4, height = 4`

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

10.8.8 Exercise (ex8.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(migest)
library(circlize)
##
##
##
# 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$region)
# 4. Modify d to enable a more sensible plot
#    1) divide flow column by a thousand
#    2) adjust the data frame to the three relevant columns for chordDiagram()
d <- d %>%
  select(-#####) %>%
  mutate(flow = flow/#####)
# 5. Check the data is in the correct by format by plotting a chord diagram
#    using the default settings
chordDiagram(x = #####)
# 6. Plot a chord diagram using
#    a. the order of provinces from r
#    b. colours from the col column in r
#    c. transparency set to 0.25
chordDiagram(x = d, order = #####$region, grid.col = r$#####, ##### = 0.25)
# 7. Edit the code below to
#    a. add directional arrows
#    b. change the height at the start and end of the chords to -0.04
chordDiagram(x = d, order = r$#####, grid.col = r$col, transparency = 0.25,
             directional = #####, direction.type = c(#####, "arrows"),
```

```
             ##### = "big.arrow", diffHeight = #####)
# 8. Use the circos.par function to set
#    a. track margins to c(0.01, -0.01)
#    d. start degree to 90
#    c. gap degree to a the gap column in object r
#    d. plot a chord diagram with these setting based based on the code in the
#       answer above
circos.par(track.margin = c(#####, -0.01), ##### = 90, gap.degree = r$#####)
chordDiagram(x = d, order = r$region, grid.col = r$col, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.04)
# 9. Edit below to sort the chord links
#    a. into and out of each section
#    b. largest links on top
chordDiagram(x = d, order = r$region, grid.col = r$col, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.04,
             link.sort = #####, link.largest.ontop = #####)
# 10. Edit the code below to
#     a. plot only the grid and the axis
#     b. set the track height of the label area to 0.1
chordDiagram(x = d, order = r$region, grid.col = r$col, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.04,
             link.sort = TRUE, link.largest.ontop = TRUE,
             ##### = c("grid", #####),
             preAllocateTracks = list(track.height = #####))
# 11. Add the labels in the track by
#     a. setting y position of label to 1
#     b. setting text facing to bending
#     c. setting the font size to 0.7
circos.track(track.index = 1, bg.border = NA, panel.fun = function(x, y) {
  s = get.cell.meta.data("sector.index")
  xx = get.cell.meta.data("xlim")
  rr = filter(#####, region == s)
  circos.text(x = mean(xx), y = #####, labels = rr$label_en,
              facing = #####, ##### = 0.7)
})
# 12. Use the code in question 10 and 11 to create the PDF version of the plot
pdf(file = "./exercise/korea2020_en.pdf", width = 5, height = 5)
```

```
##### paste in here ...
dev.off()
# 13. Run the code below to check the PDF (might not work on Mac - if so,
#     manually open PDF file to view)
file.show("./exercise/korea2020_en.pdf")
# 14. Complete the code below to add a second set of Korean labels.
#     Note: East Asian characters require a non-standard font families - see
#           ?pdfFonts for options. Might not require to set family depending on
#           settings in your computer and/or RStudio
pdf(file = "./exercise/korea2020.pdf", width = 5, height = 5, family = "Korea1")
chordDiagram(x = d, order = r$region, grid.col = r$col, transparency = 0.25,
             directional = 1, direction.type = c("diffHeight", "arrows"),
             link.arr.type = "big.arrow", diffHeight = -0.04,
             link.sort = TRUE, link.largest.ontop = TRUE,
             annotationTrack = c("grid", "axis"),
             preAllocateTracks = list(track.height = 0.1))
circos.track(track.index = 1, bg.border = NA, panel.fun = function(x, y) {
  s = get.cell.meta.data("sector.index")
  ##### <- filter(r, region == s)
  xx = get.cell.meta.data("xlim")
  circos.text(x = mean(xx), y = 1.5, labels = rr$label_en,
              facing = "bending", cex = 0.7)
  circos.text(x = mean(xx), y = 0.9, labels = rr$#####,
              facing = "bending", cex = 0.7)
})
dev.off()
file.show("./exercise/korea2020.pdf")
# 15. Run the code below to check the PDF (might not work on Mac - if so,
#     manually open PDF file to view)
file.show("./exercise/korea2020.pdf")
```

Chapter 11

Sankey Plots for Visualising Bilateral Migration

11.1 Background

11.1.1 Background

- An alternative approach to visualize bilateral migration are Sankey or alluvial plots.
- Sankey plots feature arrows with width proportional to the flow quantity.
- Named after Irish Captain Sankey, who used to show the energy efficiency of a steam engine in 1898.
- Minard's plot of Napoleon's Russian Campaign of 1812 was made in 1869 - before Sankey
- Alluvial plots are a form of Sankey plot

  - Contain blocks at nodes (e.g. origin and destination of migraiton flows)
  - No space between blocks, implying a meaningful axis, unlike Sankey plots that do have spaces

11.1.2 Men in Napoleon's 1812 Russian Campaign

11.1.3 Sankey plot of migration in Nature by Butler [2017]

11.1.4 Sankey plots in R

- As the number of regions or countries increases the plot become more cumbersome
  - Labels for the smaller areas get too small and the plotting area becomes a very long rectangle making it awkward to fit on paper or view on the screen.
  - In such cases I prefer chord diagrams
- There are a few packages in R that have functions for Sankey plots, such as *sankey*, *PantaRhei*, *networkD3*, *sankeywheel*, *plotly*, *ggsankey*.
  - Also *ggalluvial* which produces an allivual plot, but without any spaces between each sectors.
- I am going to use *ggforce* which I think is the most flexible
  - At the cost of a new layout for the data set
  - Good labels need a some work - as in *circlize* - because Sankey plots tend to have many set axis
  - Migration data tend to have only two set axis (origin and destinations)

11.1.5 Sankey plots in R

- For Sankey plots with *ggforce* the `gather_set_data()` function formats the data so that every migration corridor has two rows for the size of the migration at the origin and destination
- Can then use standard `ggplot()` function to set up the plot format. The mapping argument includes
  - `id` the id of the ribbons
  - `value` the size of the ribbons
  - `split` categories for splitting of the ribbons
- Add on layers for the ribbons themselves using `geom_parallel_sets()`
- Add blocks at the end of the ribbons to allow for clear identification of origin and destinations using `geom_parallel_sets_axes()`
- Add labels at the start and end of the ribbons using `geom_parallel_sets_axes()`

11.2 Data Format

11.2.1 UN international migrant stock data 2020

- United Nations Department of Economic and Social Affairs Population Division [2020] stock data as before

```
library(tidyverse)
un <- read_csv(file = "data/un_desa_ims_tidy.csv")
un
```

```
## # A tibble: 259,357 x 6
##     year     stock por_name por_code pob_name           pob_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
```

11.2.2 UN international migrant stock data 2020

- Plot between World Bank income groups

```
# codes for income groups
cc <- c(1503:1500, 2003)
d <- un %>%
  filter(por_code %in% cc,
         pob_code %in% cc,
         year == 2020) %>%
  rename(orig = pob_name,
         dest = por_name) %>%
  mutate(stock = stock/1e6)
d
```

```
## # A tibble: 16 x 6
##     year  stock dest                          por_code orig             pob_code
##    <dbl>  <dbl> <chr>                            <dbl> <chr>               <dbl>
##  1  2020 45.8   High-income countries             1503 High-income cou~     1503
##  2  2020 59.9   High-income countries             1503 Upper-middle-in~     1502
##  3  2020 58.0   High-income countries             1503 Lower-middle-in~     1501
##  4  2020 10.5   High-income countries             1503 Low-income coun~     1500
##  5  2020  5.66  Upper-middle-income countries      1502 High-income cou~     1503
##  6  2020 20.6   Upper-middle-income countries      1502 Upper-middle-in~     1502
##  7  2020 18.3   Upper-middle-income countries      1502 Lower-middle-in~     1501
##  8  2020 10.8   Upper-middle-income countries      1502 Low-income coun~     1500
##  9  2020  0.961 Lower-middle-income countries      1501 High-income cou~     1503
## 10  2020  6.45  Lower-middle-income countries      1501 Upper-middle-in~     1502
## 11  2020 10.5   Lower-middle-income countries      1501 Lower-middle-in~     1501
## 12  2020  7.93  Lower-middle-income countries      1501 Low-income coun~     1500
## 13  2020  0.102 Low-income countries               1500 High-income cou~     1503
## 14  2020  0.579 Low-income countries               1500 Upper-middle-in~     1502
## 15  2020  2.90  Low-income countries               1500 Lower-middle-in~     1501
## 16  2020  8.12  Low-income countries               1500 Low-income coun~     1500
```

11.2.3 Data format

- Format data for Sankey plot using `gather_set_data()` function in *ggforce*

```
library(ggforce)

s <- d %>%
  select(orig, dest, stock) %>%
  gather_set_data(x = 1:2)
s
```

```
## # A tibble: 32 x 6
##    orig                          dest            stock    id x     y
##    <chr>                         <chr>           <dbl> <int> <chr> <chr>
##  1 High-income countries         High-income c~ 45.8       1 orig  High-income ~
##  2 Upper-middle-income countries High-income c~ 59.9       2 orig  Upper-middle~
##  3 Lower-middle-income countries High-income c~ 58.0       3 orig  Lower-middle~
##  4 Low-income countries          High-income c~ 10.5       4 orig  Low-income c~
##  5 High-income countries         Upper-middle-~  5.66      5 orig  High-income ~
##  6 Upper-middle-income countries Upper-middle-~ 20.6       6 orig  Upper-middle~
##  7 Lower-middle-income countries Upper-middle-~ 18.3       7 orig  Lower-middle~
##  8 Low-income countries          Upper-middle-~ 10.8       8 orig  Low-income c~
##  9 High-income countries         Lower-middle-~  0.961     9 orig  High-income ~
## 10 Upper-middle-income countries Lower-middle-~  6.45     10 orig  Upper-middle~
## # ... with 22 more rows
```

11.2.4 Data format

```
tail(s)
```

```
## # A tibble: 6 x 6
##   orig                          dest            stock    id x     y
##   <chr>                         <chr>           <dbl> <int> <chr> <chr>
## 1 Lower-middle-income countries Lower-middle-~ 10.5      11 dest  Lower-middle-~
## 2 Low-income countries          Lower-middle-~  7.93     12 dest  Lower-middle-~
## 3 High-income countries         Low-income co~  0.102    13 dest  Low-income co~
## 4 Upper-middle-income countries Low-income co~  0.579    14 dest  Low-income co~
## 5 Lower-middle-income countries Low-income co~  2.90     15 dest  Low-income co~
## 6 Low-income countries          Low-income co~  8.12     16 dest  Low-income co~
```

11.3 Parrellel Sets

11.3.1 Default Plot

- Pass the different columns to `ggplot()` mappings
- The `geom_parallel_sets()` plots the ribbons

```
ggplot(data = s,
       mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets()
```

11.3.2 Default Plot

- By default the x-axis goes in alphabetical order
  - Use factors to set ordering of categorical variable

```
levels(s$x)
```

```
## NULL
```

```
s <- mutate(s, x = fct_rev(x))
levels(s$x)
```

```
## [1] "orig" "dest"
```

```
ggplot(data = s,
       mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets()
```

11.4 Set Axes

11.4.1 Set Axes

- The `geom_parallel_sets_axes()` function adds blocks besides the start and end of the ribbons
  - Set the width (as a proportion) using `axis.width`

```
# default axis.width
ggplot(data = s,
       mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets() +
  geom_parallel_sets_axes()
```

```
# wider axis.width
ggplot(data = s,
       mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets() +
  geom_parallel_sets_axes(axis.width = 0.1)
```

11.5 Colour

11.5.1 Colour

- Use `mapping` in `geom_parallel_sets()` to set the colours
  - Fill the colours following the origin regions, as was the case in the chord diagrams
- The `geom_parallel_sets_axes()` cannot take a fill colour from the data frame

```
# geom_parallel_sets_axes cannot take fill colours from data
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y, fill = orig)) +
  geom_parallel_sets() +
  geom_parallel_sets_axes()
```

```
## Warning: Computation failed in `stat_parallel_sets_axes()`:
## Axis aesthetics must be constant in each split
```

```
# set fill colour for parallel_sets only
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig)) +
  geom_parallel_sets_axes()
```

11.5.2 Ribbon colour - failed axis colour

11.5.3 Ribbon transparency

- Add some transparency in the ribbons using the `alpha` argument in `geom_parallel_sets()`

```
# transparency of 0.8
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8) +
  geom_parallel_sets_axes()
```

```
# transparency of 0.2
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.2) +
  geom_parallel_sets_axes()
```

11.5.4 Axis colour

- To see the set axis colours we can draw an outline using the `colour` argument.
  - Also set `fill = "transparent"` in order to view the underlying ribbons

```
# geom_parallel_sets_axes is an axis, can provide outline colour only
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8) +
  geom_parallel_sets_axes(colour = "black")
```

```
# geom_parallel_sets_axes is an axis, can provide outline colour only
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig)) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.1)
```

11.5.5 Axis colour

- Tweak the width in `geom_parallel_sets()` so that it fills into the axis box
  - Need to set `fill = "transparent"`

```
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8, axis.width = -0.1) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.1)
```

```
# narrower set axes
ggplot(data = s,mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8, axis.width = -0.05) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.05)
```

11.6 Labels

11.6.1 Labels

- Add labels on the x-axis using `scale_x_discrete()` from *ggplot2*
- Add labels to the sets using `geom_parallel_sets_labels()` from *ggforce*
  - Terrible default positions and angles if labels are not very short.

```
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8, axis.width = -0.05) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.05) +
  guides(fill = "none") +
  geom_parallel_sets_labels() +
  scale_x_discrete(labels = c(orig = "Place of Birth",
                              dest = "Place of Residence"))
```

```
ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8, axis.width = -0.05) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.05) +
  guides(fill = "none") +
  geom_parallel_sets_labels(angle = 0) +
  scale_x_discrete(labels = c(orig = "Place of Birth",
                              dest = "Place of Residence"))
```

11.6.2 Labels

- Change order of origin and destinations by modifying the levels of the factors
  - Set levels to order they appear in the `y` column using `fct_inorder()` in the *forcats* package
  - Remove unnecessary parts in the label

```
levels(s$y)
```

```
## NULL
```

```
s <- s %>%
  mutate(y = str_remove(string = y, pattern = "-income countries"),
         y = fct_inorder(y))
levels(s$y)
```

```
## [1] "High"         "Upper-middle" "Lower-middle" "Low"
```

```
s
```

```
## # A tibble: 32 x 6
```

```
##    orig                          dest                 stock    id x     y
##    <chr>                         <chr>                <dbl> <int> <fct> <fct>
##  1 High-income countries         High-income countr~ 45.8       1 orig  High
##  2 Upper-middle-income countries High-income countr~ 59.9       2 orig  Upper-m~
##  3 Lower-middle-income countries High-income countr~ 58.0       3 orig  Lower-m~
##  4 Low-income countries          High-income countr~ 10.5       4 orig  Low
##  5 High-income countries         Upper-middle-incom~  5.66      5 orig  High
##  6 Upper-middle-income countries Upper-middle-incom~ 20.6       6 orig  Upper-m~
##  7 Lower-middle-income countries Upper-middle-incom~ 18.3       7 orig  Lower-m~
##  8 Low-income countries          Upper-middle-incom~ 10.8       8 orig  Low
##  9 High-income countries         Lower-middle-incom~  0.961     9 orig  High
## 10 Upper-middle-income countries Lower-middle-incom~  6.45     10 orig  Upper-m~
## # ... with 22 more rows
```

11.6.3 Labels

- Run same code as before, with updates `s`,…

```
ggplot(data = s,
       mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8, axis.width = -0.05) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.05) +
  guides(fill = "none") +
  geom_parallel_sets_labels(angle = 0) +
  scale_x_discrete(labels = c(orig = "Place of Birth",
                              dest = "Place of Residence"))
```

11.6.4 Labels

- Set up a label data frame to adjust position and alignment

```
p <- s %>%
  distinct(x, y) %>%
  mutate(h = as.numeric(x == "orig"),
         n = h * -0.1 + 0.05)
p
```

```
## # A tibble: 8 x 4
##   x     y                h     n
##   <fct> <fct>        <dbl> <dbl>
## 1 orig  High             1 -0.05
## 2 orig  Upper-middle     1 -0.05
## 3 orig  Lower-middle     1 -0.05
## 4 orig  Low              1 -0.05
## 5 dest  High             0  0.05
## 6 dest  Upper-middle     0  0.05
## 7 dest  Lower-middle     0  0.05
## 8 dest  Low              0  0.05
```

11.6.5 Labels

- Pass the position coordinates to the ggplot code

```
ggplot(data = s,
       mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8,
                     axis.width = -0.05) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.05) +
  guides(fill = "none") +
  geom_parallel_sets_labels(angle = 0, hjust = p$h,
                            position = position_nudge(x = p$n)) +
  scale_x_discrete(labels = c(orig = "Place of Birth",
                              dest = "Place of Residence"))
```

11.7 Spacing

11.7.1 Spacing

- We convert the Sankey plot to an alluvial plot by reducing the space separating the parallel sets to zero via the `sep` argument
  - Need to set `sep` in all the geom functions for alignment.

  - Default is `sep = 0.05` (5%)
  - Might need to reduce when have many regions

- In alluvial plots the y-axis are more meaningful

  - Add y-axis labels via `labs()` function

- Set background to white using `theme_bw()` function

```
ggplot(data = s,
       mapping = aes(x = x, id = id, value = stock, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8,
                     axis.width = -0.05, sep = 0) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = 0.05, sep = 0) +
  guides(fill = "none") +
  geom_parallel_sets_labels(angle = 0, hjust = p$h,
                            position = position_nudge(x = p$n, ), sep = 0) +
  scale_x_discrete(labels = c(orig = "Place of Birth",
                              dest = "Place of Residence")) +
  labs(y = "Migrants (millions)", x = "") +
  theme_bw()
```

11.7.2 Exercise (ex9.R)

```
# 0.  a) Load the KOSTAT2021.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2021.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(ggforce)
##
##
##
##
# 1. Run the code below to read in the migrant stock data from Gabon taken
#    from Table 21-6 in Shryock & Siegel (1979)
ga <- read_csv("./data/gabon_1961_tidy.csv")
ga
# 2. Run the code below to remove the totals groups and migrants from abroad
d <- ga %>%
  rename(orig = place_of_birth,
         dest = place_of_enumeration) %>%
  filter(sex == "total",
         !orig %in% c("Grand total", "Abroad", "Total Gabon"),
         dest != "Total") %>%
  select(-sex)
d
# 3. Create a data frame s1 using the gather_set_data() function to organise the
#    Gabon data in d ready for a Sankey plot using geom_parallel_sets
s1 <- d %>%
  select(orig, dest, #####) %>%
  #####(x = 1:#####)
s1
# 4. Run an initial plot on s1 to inspect for potential changes required to the
#    the data frame
ggplot(data = s1,
       mapping = aes(x = x, id = id, value = migrants, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig))
# 5. Create a new data s2, based on d, that
#    a. Sets migrant counts to zero for the migrant corridors for native born
#       persons, where the place of enumeration is the same as the place of
#       birth (orig == dest)
#    b. Divide the migrant counts by one thousand
#    c. Re-organises the data using the gather_set_data() function
#    d. Sets both x and y to factors based on order of appearance using the
#       fct_inorder() function (which broadly follows a north to south order)
```

```
s2 <-  d %>%
  mutate(migrants = ifelse(test = orig == #####, yes = #####, no = migrants),
         migrants = migrants/#####) %>%
  select(orig, dest, migrants) %>%
  #####(x = 1:2) %>%
  mutate(x = fct_inorder(x),
         y = #####(y))
s2
levels(s2$y)
# 6. Create an object p that sets the horizontal positioning and nudge amount
#    for each origin and destination label
p <- s2 %>%
  #####(x, y) %>%
  mutate(h = as.numeric(x == "orig"),
         n = h * -0.1 + 0.05)
p
# 7. Complete the code below for a plot of the intra-regional migrant
#    distributions for Gabon
ggplot(data = s2,
       mapping = aes(x = x, id = id, value = #####, split = y)) +
  geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8,
                     axis.width = -0.05) +
  geom_parallel_sets_axes(fill = "transparent", colour = "black",
                          axis.width = #####) +
  guides(fill = #####) +
  geom_parallel_sets_labels(angle = #####, hjust = p$h,
                            position = position_nudge(x = p$n, )) +
  scale_x_discrete(labels = c(orig = "Place of Birth",
                              dest = "Place of Residence")) +
  labs(y = "Migrants (thousands)", x = "") +
  theme_bw()
# 9. Run the code below to check the PDF (might not work on Mac - if so,
#    manually open PDF file to view)
ggsave(filename = "./exercise/gabon1961.pdf", height = 8, width = 8)
file.show("./exercise/gabon1961.pdf")
```

Bibliography

Guy J. Abel. Estimating global migration flow tables using place of birth data. *Demographic Research*, 28(March):505–546, mar 2013. ISSN 14359871. doi: 10.4054/DemRes.2013.28.18. URL http://www.demographic-research.org/volumes/vol28/18/.

Guy J. Abel and Joel E. Cohen. Bilateral international migration flow estimates for 200 countries. *Scientific Data*, 6(1):82, dec 2019. ISSN 2052-4463. doi: 10.1038/s41597-019-0089-3. URL https://www.nature.com/articles/s41597-019-0089-3.

Guy J. Abel and Nikola Sander. Quantifying Global International Migration Flows. *Science*, 343(6178):1520–1522, mar 2014. ISSN 0036-8075. doi: 10.1126/science.1248676. URL http://www.sciencemag.org/cgi/doi/10.1126/science.1248676.

Johan Barthélemy and Thomas Suesse. mipfp : An R Package for Multidimensional Array Fitting and Simulating Multivariate Bernoulli Distributions. *Journal of Statistical Software*, 86(Code Snippet 2), 2018. ISSN 1548-7660. doi: 10.18637/jss.v086.c02. URL http://www.jstatsoft.org/v86/c02/.

Martin Bell and Salut Muhidin. Cross-National Comparisons of Internal Migration. 2009.

Martin Bell, Marcus Blake, Paul Boyle, O. Duke-Williams, Philip H. Rees, John Stillwell, and Graeme John Hugo. Cross-national comparison of internal migration: issues and measures. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 165(3):435–464, oct 2002. ISSN 0964-1998. doi: 10.1111/1467-985X.00247. URL http://www.jstor.org/stable/3559697{%}5Cnhttp://www.jstor.org/stable/pdfplus/3559697.pdf?acceptTC=truehttp://doi.wiley.com/10.1111/1467-985X.00247.

Martin Bell, Elin Charles-Edwards, Dorota Kupiszewska, Marek Kupiszewski, John Stillwell, and Yu Zhu. Internal Migration Data Around the World: Assessing Contemporary Practice. *Population, Space and Place*, 21(1):1–17,

jan 2015a. ISSN 15448444. doi: 10.1002/psp.1848. URL http://doi.wiley.com/10.1002/psp.1848.

Martin Bell, Elin Charles-Edwards, Philipp Ueffing, John Stillwell, Marek Kupiszewski, and Dorota Kupiszewska. Internal Migration and Development: Comparing Migration Intensities Around the World. *Population and Development Review*, 41(1):33–58, 2015b. ISSN 17284457. doi: 10.1111/j.1728-4457.2015.00025.x.

Aude Bernard, Martin Bell, and Elin Charles-Edwards. Improved measures for the cross-national comparison of age profiles of internal migration. *Population Studies*, 68(2):179–195, may 2014. ISSN 0032-4728. doi: 10.1080/00324728.2014.890243. URL http://www.ncbi.nlm.nih.gov/pubmed/24779398http://www.tandfonline.com/doi/abs/10.1080/00324728.2014.890243.

Donald J Bogue, Kenneth Hinze, and Michael White. *Techniques of Estimating Net Migration*. Community and Family Study Center. University of Chicago, Chicago, USA, 1982. ISBN 0-89836-018-8.

Declan Butler. What the numbers say about refugees. *Nature*, 543(7643):22–23, mar 2017. ISSN 0028-0836. doi: 10.1038/543022a. URL http://www.nature.com/polopoly{_}fs/1.21548!/menu/main/topColumns/topLeftColumn/pdf/543022a.pdfhttp://www.nature.com/doifinder/10.1038/543022a.

Daniel Courgeau. Migrations et découpages du territoire. *Population (French Edition)*, 28(3):511, may 1973. ISSN 00324663. doi: 10.2307/1530704. URL https://www.jstor.org/stable/1530704?origin=crossref.

Daniel Courgeau. Migrants and migrations. *Population (English Edition)*, 3: 1—-35, 1979.

Joop de Beer, James Raymer, Rob van der Erf, and Leo van Wissen. Overcoming the Problems of Inconsistent International Migration data: A New Method Applied to Flows in Europe. *European Journal of Population / Revue européenne de Démographie*, 26(4):459–481, nov 2010. ISSN 0168-6577. doi: 10.1007/s10680-010-9220-z. URL http://link.springer.com/10.1007/s10680-010-9220-z.

W. Edwards Deming and Frederick F Stephan. On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known. *The Annals of Mathematical Statistics*, 11(4):427–444, dec 1940. ISSN 0003-4851. doi: 10.1214/aoms/1177731829. URL http://www.jstor.org/stable/2235722{%}5Cnhttp://www.jstor.org/stable/2235722?seq=1{&}cid=pdf-reference{#}references{_}tab{_}contents{%}5Cnhttp://about.jstor.org/termshttp://projecteuclid.org/euclid.aoms/1177731829.

Robin Edwards, Maksym Bondarenko, Alessandro Sorichetta, and Andrew J. Tatem. Unconstrained subnational Population Weighted Density in 2000, 2005, 2010, 2015 and 2020 (100m resolution), 2021.

Z. Gu, Lei Gu, Roland Eils, Matthias Schlesner, and Benedikt Brors. circlize implements and enhances circular visualization in R. *Bioinformatics*, 30(19): 2811–2812, oct 2014. ISSN 1367-4803. doi: 10.1093/bioinformatics/btu393. URL http://www.ncbi.nlm.nih.gov/pubmed/24930139https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu393.

Martin Krzywinski, Jacqueline Schein, Inanç Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J. Jones, and Marco A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19 (9):1639–1645, sep 2009. ISSN 1088-9051. doi: 10.1101/gr.092759.109. URL http://genome.cshlp.org/cgi/doi/10.1101/gr.092759.109.

Everett S. Lee. A Theory of Migration. *Demography*, 3(1):47, 1966. ISSN 00703370. doi: 10.2307/2060063. URL http://link.springer.com/10.2307/2060063.

Nik Lomax and Paul Norman. Estimating population attribute values in a table: "Get me started in" iterative proportional fitting. *Professional Geographer*, 68(3):451–461, 2016. ISSN 14679272. doi: 10.1080/00330124.2015.1099449.

Robin Lovelace, Mark Birkin, Dimitris Ballas, and Eveline van Leeuwen. Evaluating the Performance of Iterative Proportional Fitting for Spatial Microsimulation: New Tests for an Established Technique. *Journal of Artificial Societies and Social Simulation*, 18(2):1–15, 2015. ISSN 1460-7425. doi: 10.18564/jasss.2768. URL http://jasss.soc.surrey.ac.uk/18/2/21.html.

David A. Plane. Estimation of Place-to-Place Migration Flows from Net Migration Totals: A Minimum Information Approach. *International Regional Science Review*, 6(1):33–51, apr 1981. ISSN 0160-0176. doi: 10.1177/016001768100600103. URL http://irx.sagepub.com/cgi/doi/10.1177/016001768100600103.

James Raymer. The estimation of international migration flows: a general technique focused on the origin–destination association structure. *Environment and Planning A*, 39(4):985–995, 2007. doi: 10.1068/a38264.

James Raymer, Alberto Bonaguidi, and Alessandro Valentini. Describing and projecting the age and spatial structures of interregional migration in Italy. *Population, Space and Place*, 12(5):371–388, sep 2006. ISSN 1544-8444. doi: 10.1002/psp.414. URL http://onlinelibrary.wiley.com/doi/10.1002/psp.414/abstracthttp://doi.wiley.com/10.1002/psp.414.

Philip H. Rees, Martin Bell, Marek Kupiszewski, Dorota Kupiszewska, Philipp Ueffing, Aude Bernard, Elin Charles-Edwards, and John Stillwell. The Impact of Internal Migration on Population Redistribution: an International Comparison. *Population, Space and Place*, (April), 2016. ISSN 15448444. doi: 10.1002/psp.2036. URL http://doi.wiley.com/10.1002/psp.2036.

Andrei Rogers. *Introduction to Multiregional Mathematical Demography*. Wiley, New York, New York, USA, 1975.

Andrei Rogers. Requiem for the Net Migrant. *Geographical Analysis*, 22 (4):283–300, 1990. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.1990.tb00212.x/abstract.

Andrei Rogers and Luis J. Castro. Model Migration Schedules. Technical Report RR-81-30, International Institute for Applied Systems Analysis, Laxenburg, Austria, nov 1981. URL http://webarchive.iiasa.ac.at/Admin/PUB/Documents/RR-81-030.pdf.

Andrei Rogers and Jani S Little. An International Journal of Parameterizing age patterns of demographic rates with the multiexponential model schedule. *Mathematical Population Studies*, 4(3):175–195, 1994. doi: 10.1080/08898489409525372.

Andrei Rogers and James Raymer. The Spatial Focus of US Interstate Migration Flows. *International Journal of Population Geography*, 4(1):63–80, 1998. doi: 10.1002/(SICI)1099-1220(199803)4%3A1<63%3A%3AAID-IJPG87>3.0.CO%3B2-U.

Andrei Rogers and James Raymer. Estimating the regional migration patterns of the foreign-born population in the United States: 1950-1990. *Mathematical Population Studies*, 7(3):181–216, 307, jan 1999. ISSN 0889-8480. doi: 10.1080/08898489909525457. URL http://www.ncbi.nlm.nih.gov/pubmed/12295224.

Andrei Rogers and John Watkins. General Versus Elderly Interstate Migration and Population Redistribution in the United States. *Research on Aging*, 9(4):483–529, dec 1987. ISSN 0164-0275. doi: 10.1177/0164027587094002. URL http://roa.sagepub.com/content/9/4/483.shorthttp://roa.sagepub.com/cgi/doi/10.1177/0164027587094002.

Andrei Rogers, Frans Willekens, Jani Little, and James Raymer. Describing migration spatial structure. *Papers in Regional Science*, 81(1):29–48, jan 2002. doi: 10.1007/s101100100090.

Andrei Rogers, Frans Willekens, and James Raymer. Imposing Age and Spatial Structures on Inadequate Migration-Flow Datasets. *The Professional Geographer*, 55(1):56–69, 2003. ISSN 00330124. doi: 10.1111/0033-0124.01052. URL http://www.informaworld.com/10.1111/0033-0124.01052.

Andrei Rogers, James Raymer, and Jani Little. *The Indirect Estimation of Migration*, volume 26 of *The Springer Series on Demographic Methods and Population Analysis*. Springer Netherlands, Dordrecht, 2010. ISBN 978-90-481-8914-4. doi: 10.1007/978-90-481-8915-1. URL http://link.springer.com/10.1007/978-90-481-8915-1.

Peter A Rogerson. Migration analysis using data with time intervals of differing widths. *Papers of the Regional Science Association*, 68(1981):97–106, 1990. URL https://link.springer.com/article/10.1007/BF01933910.

Henry S. Shryock and Jacob S. Siegel. *The Methods and Materials of Demography.* Academic Press, San Diego, California, condensed edition, 1976. ISBN 0126411506.

Jacob S. Siegel and C. Horace Hamilton. Some Considerations in the Use of the Residual Method of Estimating Net Migration. *Journal of the American Statistical Association*, 47(259):475–500, 1952. ISSN 1537274X. doi: 10.1080/01621459.1952.10501186.

John Stillwell, Martin Bell, Philipp Ueffing, K. Daras, Elin Charles-Edwards, Marek Kupiszewski, and Dorota Kupiszewska. Internal migration around the world: comparing distance travelled and its frictional effect. *Environment and Planning A*, 48(8):1657–1675, aug 2016. ISSN 0308-518X. doi: 10.1177/0308518X16643963. URL http://epn.sagepub.com/lookup/doi/10.1177/0308518X16643963.

United Nations Department of Economic and Social Affairs Population Division. *Methods of measuring internal migration.* United Nations Publication, New York, New York, USA, 1983. URL https://www.un.org/en/development/desa/population/publications/manual/migration/measuring-migration.asp.

United Nations Department of Economic and Social Affairs Population Division. *Preparing Migration Data for Subnational Population Projections.* 1992. ISBN 9211512433. URL http://www.un.org/esa/population/techcoop/IntMig/migdata{_}popproj/migdata{_}popproj.html.

United Nations Department of Economic and Social Affairs Population Division. International Migrant Stock 2020 (United Nations database, POP/DB/MIG/Stock/Rev.2020), 2020.

Evert van Imhoff, Nicole van der Gaag, Leo van Wissen, and Philip H. Rees. The selection of internal migration models for European regions. *International Journal of Population Geography IJPG*, 3(2):137–59, jun 1997. ISSN 1077-3495. doi: 10.1002/(SICI)1099-1220(199706)3:2<137::AID-IJPG63>3.0.CO;2-R. URL http://www.ncbi.nlm.nih.gov/pubmed/12321164.

Frans Willekens. Modeling approaches to the indirect estimation of migration flows: from entropy to EM. *Mathematical population studies*, 7(3):239–78, 308, jan 1999. ISSN 0889-8480. doi: 10.1080/08898489909525459. URL http://www.ncbi.nlm.nih.gov/pubmed/12295226.

Tom Wilson. Model migration schedules incorporating student migration peaks. *Demographic Research*, 23(8):191–222, 2010. doi: 10.4054/DemRes.2010.23.8.