Handling Migration Data in R

Guy J. Abel

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

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_{++}
- Will use some of this notation later on

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
```

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 26 31 78
   B  5 71 81 25
   C 20  1 16 91
   D 77 14 60 64

, , sex = male

    dest
orig  A  B  C  D
   A 10 45 29 12
   B 44  4 74 82
   C 89 35 98 92
   D 66  6 54 48
```

Show totals

- The `addmargins()` functions adds extra row, column and tables to display the dimension sums.
  - `margin` argument to specify a specific margin to sum over

```
> 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
>
> # sum over just the rows (first dimension)
> addmargins(A = m0, margin = 1)
      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
  Sum  130 160 95 115
```

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`

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
```

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
```

Convert to data frame

- 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`

```
> # 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
```

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      5
 3 C     A     female     20
 4 D     A     female     77
 5 A     B     female     26
 6 B     B     female     71
 7 C     B     female      1
 8 D     B     female     14
 9 A     C     female     31
10 B     C     female     81
# ... with 22 more rows
```

Displaying migration matrics

- Migration data in `matrix` objects can be difficult to view
  - Lengthy dimension (region) names
  - Large unit sizes
  - Large diagonal terms included - but not of interest
- For example, 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)

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
```

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())
```

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
```

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
```

Diagonal elements

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

```
> u1 <- u0
> diag(u1) <- 0
> u1
      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
```

Data scaling

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

```
> u2 <- round(x = u1/1000, digits = 1)
> u2
      dest
orig    Cair  Alxn Pr-S Isml Klyb Ghrb Menf Giza Assy Sohg Allo
  Cair   0.0  31.0  5.3  9.8 23.8 10.0  7.0 88.5  5.0  2.6 58.5
  Alxn  47.2   0.0  2.6  2.6  2.1  4.9  1.5  6.9  1.4  1.5 29.5
  Pr-S   9.5   2.6  0.0  6.5  0.5  0.8  0.3  1.5  0.3  0.5 11.2
  Isml   9.5   1.4  3.5  0.0  0.7  0.9  0.3  1.6  0.3  0.3 10.3
  Klyb  90.7   4.7  0.8  3.2  0.0  3.7  3.5 10.3  0.3  0.1 18.1
  Ghrb  99.2  40.0  1.7  3.3  7.9  0.0  6.3 14.5  0.8  0.5 64.1
  Menf 216.8  46.8  1.6  3.3  2.9 29.6  0.0 30.9  0.6  0.4 47.8
  Giza  64.6   4.9  0.5  2.0  2.9  1.5  2.2  0.0  0.5  0.4 13.5
  Assy 100.3  25.5  1.7  2.5  0.1  2.2  0.6 13.2  0.0  6.0 35.2
  Sohg 100.1  63.7 12.1  9.4  0.3  2.8  1.1 18.0 11.6  0.0 53.2
  Allo 456.5 177.5 43.9 67.0 49.8 47.3 12.2 94.6 14.7 22.4  0.0
```

Net flows and counterflows

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

```
> sum_bilat(m0)
# A tibble: 12 x 8
   orig  dest  corridor pair   flow counter_flow net_flow interchange
   <chr> <chr> <chr>    <chr> <dbl>        <dbl>    <dbl>        <dbl>
 1 B     A     B -> A   A - B    50          100      -50          150
 2 C     A     C -> A   A - C    60           30       30           90
 3 D     A     D -> A   A - D    20           70      -50           90
 4 A     B     A -> B   A - B   100           50       50          150
 5 C     B     C -> B   B - C    35           45      -10           80
 6 D     B     D -> B   B - D    25            5       20           30
 7 A     C     A -> C   A - C    30           60      -30           90
 8 B     C     B -> C   B - C    45           35       10           80
 9 D     C     D -> C   C - D    20           40      -20           60
10 A     D     A -> D   A - D    70           20       50           90
11 B     D     B -> D   B - D     5           25      -20           30
12 C     D     C -> D   C - D    40           20       20           60
```

Net flows and counterflows

```
> d1 %>%
+   group_by(sex) %>%
+   sum_bilat()
# A tibble: 24 x 9
# Groups:   sex [2]
   orig  dest  corridor pair  sex     flow counter_flow net_flow interchange
   <chr> <chr> <chr>    <chr> <fct>  <int>        <int>    <int>        <int>
 1 B     A     B -> A   A - B female     5           26      -21           31
 2 C     A     C -> A   A - C female    20           31      -11           51
 3 D     A     D -> A   A - D female    77           78       -1          155
 4 A     B     A -> B   A - B female    26            5       21           31
 5 C     B     C -> B   B - C female     1           81      -80           82
 6 D     B     D -> B   B - D female    14           25      -11           39
 7 A     C     A -> C   A - C female    31           20       11           51
 8 B     C     B -> C   B - C female    81            1       80           82
 9 D     C     D -> C   C - D female    60           91      -31          151
10 A     D     A -> D   A - D female    78           77        1          155
# ... with 14 more rows
```

Totals

- The `sum_region()` provides summary in-migration, out-migration, net-migration and turnover totals for each region
  - Accepts `matrix` or `data.frame` (or `tibble`) inputs
  - The `sum_country()` works the same as `sum_region()` but provides labels relevant to international migration.

```
> sum_region(m0)
# A tibble: 4 x 5
  region out_mig in_mig  turn   net
  <chr>    <dbl>  <dbl> <dbl> <dbl>
1 A          200    130   330   -70
2 B          100    160   260    60
3 C          135     95   230   -40
4 D           65    115   180    50
```

Totals

```
> sum_country(m = d0)
# A tibble: 4 x 5
  country   emi   imm  turn   net
  <chr>   <dbl> <dbl> <dbl> <dbl>
1 A           9    27    36    18
2 B          20    26    46     6
3 C          31    25    56    -6
4 D          42    24    66   -18
```

Totals

- The `sum_region()` or `sum_country()` functions 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>
```

Totals

```
> # single period
> f %>%
+   filter(year0 == 1990) %>%
+   sum_country(flow_col = "da_min_open")
# A tibble: 197 x 5
   country    emi     imm    turn     net
   <chr>    <dbl>   <dbl>   <dbl>   <dbl>
 1 ABW        323    7429    7752    7106
 2 AFG          8 3033115 3033123 3033107
 3 AGO          0   82264   82264   82264
 4 ALB     335374    5695  341069 -329679
 5 ARE      35922  510662  546584  474740
 6 ARG     186437  128287  314724  -58150
 7 ARM     212706  139629  352335  -73077
 8 ATG       5181    5235   10416      54
 9 AUS      71159  322450  393609  251291
10 AUT       3229  148344  151573  145115
# ... with 187 more rows
```

Totals

```
> # all periods using group_by
> f %>%
+   group_by(year0) %>%
+   sum_country(flow_col = "da_min_open") %>%
+   arrange(country)
# A tibble: 1,188 x 6
# Groups:   year0 [6]
   year0 country     emi     imm    turn     net
   <dbl> <chr>     <dbl>   <dbl>   <dbl>   <dbl>
 1  1990 ABW         323    7429    7752    7106
 2  1995 ABW        1268    7625    8893    6357
 3  2000 ABW        1070    3682    4752    2612
 4  2005 ABW        2418    3323    5741     905
 5  2010 ABW        3190    3474    6664     284
 6  2015 ABW        7596   18178   25774   10582
 7  1990 AFG           8 3033115 3033123 3033107
 8  1995 AFG      619484   10280  629764 -609204
 9  2000 AFG        2325  524656  526981  522331
10  2005 AFG     1244141   12659 1256800 -1231482
# ... with 1,178 more rows
```

Exercise (ex2.R)

```
# 0.  a) Load the KOSTAT2022.Rproj file.
#     Run the getwd() below. It should print the directory where the
#     KOSTAT2022.Rproj file is located.
getwd()
#     b) Load the packages used in this exercise
library(tidyverse)
library(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
```

References

Abel, Guy J., and Joel E. Cohen. 2019. "Bilateral international migration flow estimates for 200 countries." *Scientific Data* 6 (1): 82. https://doi.org/10.1038/s41597-019-0089-3.

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