Replication multiregional population projection Willekens & Rogers (1978)

Central steps multiregional/state population projection model:

1. Calculation of observed transfer rates rates M_x.
2. Calculation of P_x, the per-region observed death and outmigration probabilities (transfer probabilities), based on M_x.
3. Calculation of expected number of survivors, based on P_x.
4. Calculation of duration of residence/years lived, L_x based on the number of survivors.
5. Calculation of the surivivorship proportions S_x, based on L_x.
6. Calculation of the fertility proportions B_x, based on P_x, S_x and the observed fertility rates.
7. Construction of generalized Leslie matrix G based on S_x and B_x.
8. Projection using initial population n_0 and G.
9. Deriving the stable equivalent population using eigenvalue decomposition of G.

1. Observed population characteristics

Replicate Table 1.3 "Observed rates", pg. 11-12

Observed death rate

```
round(DR, 6)
```

```
##    slovenia  r.yogos
## 0  0.006150 0.022468
## 5  0.000432 0.000669
## 10 0.000297 0.000478
## 15 0.000516 0.000865
## 20 0.000747 0.001220
## 25 0.000677 0.001585
## 30 0.000999 0.001753
## 35 0.001224 0.002073
## 40 0.001924 0.002872
## 45 0.003570 0.003889
## 50 0.005224 0.006382
## 55 0.008004 0.009615
## 60 0.012955 0.016857
## 65 0.025864 0.027784
## 70 0.047742 0.048067
```

```
## 75 0.075556 0.072084
## 80 0.146620 0.111814
## 85 0.203611 0.143486
```

Observed fertility rate

```r
round(FR, 6)
```

```
##    slovenia  r.yogos
## 0  0.000000 0.000000
## 5  0.000000 0.000000
## 10 0.000071 0.000067
## 15 0.015857 0.026458
## 20 0.070652 0.087978
## 25 0.063218 0.074260
## 30 0.041103 0.044290
## 35 0.022862 0.023532
## 40 0.007797 0.012051
## 45 0.000710 0.002151
## 50 0.000292 0.000714
## 55 0.000000 0.000000
## 60 0.000000 0.000000
## 65 0.000000 0.000000
## 70 0.000000 0.000000
## 75 0.000000 0.000000
## 80 0.000000 0.000000
## 85 0.000000 0.000000
```

Observed migration rate

```r
round(MR, 6)
```

```
##    slovenia  r.yogos
## 0  0.002832 0.000272
## 5  0.002294 0.000166
## 10 0.001485 0.000157
## 15 0.005158 0.000679
## 20 0.007170 0.000937
## 25 0.005534 0.000506
## 30 0.003756 0.000350
## 35 0.001765 0.000226
## 40 0.001013 0.000183
## 45 0.000543 0.000094
## 50 0.000663 0.000130
```

```
## 55 0.000629 0.000205
## 60 0.000884 0.000203
## 65 0.000949 0.000156
## 70 0.000876 0.000078
## 75 0.001111 0.000099
## 80 0.000704 0.000196
## 85 0.001111 0.000076
```

2. The multiregional life table

Table 2.1 (p. 22) presents P_x, the **probabilities of dying and outmigrating** (or transfer probabilities). The calculation is presented in section 2.7, based on the observed death and migration rates (section 1).

```
M <- transfer_matrix(M.obs, multiple = TRUE)
P <- transfer_prob(M, multiple = TRUE)

ages <- seq(0, 85, 5)
states <- c("slovenia", "r.yugos")

P.sl <- state_table(P, 1, ages)
P.ryu <- state_table(P, 2, ages)

# calculate per region death rate
P.sl <- cbind(1 - rowSums(P.sl), P.sl)
P.ryu <- cbind(1 - rowSums(P.ryu), P.ryu)

colnames(P.sl) <- colnames(P.ryu) <- c("death", "to slov", "to r.yug")

P.sl  # prob. of dying & outmigrating for Slovenia
```

```
##       death to slov to r.yug
## 0  0.030813  0.9561 0.013103
## 5  0.002164  0.9865 0.011370
## 10 0.001487  0.9911 0.007381
## 15 0.002598  0.9721 0.025332
## 20 0.003770  0.9613 0.034968
## 25 0.003439  0.9695 0.027105
## 30 0.005015  0.9765 0.018460
## 35 0.006121  0.9852 0.008708
## 40 0.009586  0.9854 0.004988
## 45 0.017694  0.9796 0.002660
## 50 0.025793  0.9710 0.003213
## 55 0.039248  0.9577 0.003005
```

```
## 60 0.062779  0.9331 0.004097
## 65 0.121487  0.8744 0.004157
## 70 0.213259  0.7833 0.003484
## 75 0.317728  0.6783 0.003949
## 80 0.536332  0.4617 0.002010
## 85 1.000000  0.0000 0.000000
```

```
P.ryu  # prob. of dying & outmigrating for rest of Yugoslavia
```

```
##       death   to slov to r.yug
## 0  0.106319 0.0012606   0.8924
## 5  0.003341 0.0008212   0.9958
## 10 0.002385 0.0007811   0.9968
## 15 0.004312 0.0033330   0.9924
## 20 0.006075 0.0045710   0.9894
## 25 0.007889 0.0024807   0.9896
## 30 0.008724 0.0017208   0.9896
## 35 0.010311 0.0011142   0.9886
## 40 0.014256 0.0009038   0.9848
## 45 0.019259 0.0004597   0.9803
## 50 0.031405 0.0006304   0.9680
## 55 0.046941 0.0009817   0.9521
## 60 0.080868 0.0009390   0.9182
## 65 0.129894 0.0006838   0.8694
## 70 0.214551 0.0003095   0.7851
## 75 0.305390 0.0003528   0.6943
## 80 0.436969 0.0005598   0.5625
## 85 1.000000 0.0000000   0.0000
```

2.1 Life Histories

2.2 Expected Number of Survivors at Exact Age x

The expected number of survivors is calculated based on P_x, the probability of dying and outmigrating, using

$$l_{x+5} = P_x l_x$$

where l_0 is a selected cohort size, identical for both regions, in this case 100000.

The following two table replicate tabel 2.3, pg. 30.

```
L.surv <- expected_survivors(P, radix = 1e+05)

# Exp. # of surv., initial region Slov.
round(state_table(L.surv, 1, ages, states))
```

```
##    slovenia r.yugos
## 0    100000       0
## 5     95608    1310
## 10    94316    2392
## 15    93481    3080
## 20    90880    5425
## 25    87385    8545
## 30    84737   10825
## 35    82766   12276
## 40    81552   12857
## 45    80376   13069
## 50    78746   13025
## 55    76470   12861
## 60    73251   12474
## 65    68364   11754
## 70    59783   10503
## 75    46828    8455
## 80    31768    6055
## 85    14669    3469
```

```
# Exp. # of surv., initial region r. Yugos.
round(state_table(L.surv, 2, ages, states))
```

```
##    slovenia r.yugos
## 0         0  100000
## 5       126   89242
## 10      198   88872
## 15      265   88592
## 20      553   87922
## 25      934   87005
## 30     1121   86128
## 35     1243   85249
## 40     1319   84286
## 45     1376   83015
## 50     1386   81381
## 55     1398   78779
## 60     1416   75008
## 65     1392   68877
## 70     1264   59889
## 75     1008   47026
## 80      701   32652
## 85      342   18367
```

2.3 Duration of Residence and Age Composition of the Life Table Population

The duration of residence by place of birth is calculated for every age interval using

$$L_x = \frac{5}{2}(\hat{l}_x + \hat{l}_{x+5})$$

where l_x and l_{x+5} refer to the expected number of survivors for age x (cf. section 2.2). For the last age interval, which is half-open and thus $\hat{l}_{x+5}$ not defined, we use

$$L_z = M_z^{-1}\hat{l}_z$$

where M_z is the matrix containg the observed transtion rates for the last age interval.

```
L.dur <- years_lived(L.surv, M)

# duration/number of years lived in each region, for Slov.
state_table(L.dur, 1, ages, states)
```

```
##    slovenia r.yugos
## 0    4.8902 0.03276
## 5    4.7481 0.09256
## 10   4.6949 0.13681
## 15   4.6090 0.21264
## 20   4.4566 0.34926
## 25   4.3030 0.48426
## 30   4.1876 0.57753
## 35   4.1080 0.62833
## 40   4.0482 0.64814
## 45   3.9780 0.65234
## 50   3.8804 0.64713
## 55   3.7430 0.63337
## 60   3.5404 0.60569
## 65   3.2037 0.55642
## 70   2.6653 0.47394
## 75   1.9649 0.36273
## 80   1.1609 0.23810
## 85   0.7166 0.24721
```

```r
# duration/number of years lived in each region, for r. Yugos.
state_table(L.dur, 2, ages, states)
```

```
##    slovenia r.yugos
## 0  0.003152   4.731
## 5  0.008093   4.453
## 10 0.011574   4.437
## 15 0.020462   4.413
## 20 0.037170   4.373
## 25 0.051365   4.328
## 30 0.059095   4.284
## 35 0.064056   4.238
## 40 0.067394   4.183
## 45 0.069071   4.110
## 50 0.069602   4.004
## 55 0.070336   3.845
## 60 0.070187   3.597
## 65 0.066386   3.219
## 70 0.056808   2.673
## 75 0.042728   1.992
## 80 0.026060   1.275
## 85 0.017170   1.280
```

4. Total number of Years Lived beyond Age x

5. Expectation of Life

6. Survivorship and Outmigration Proportions

For the projection matrix, we need the age-specific matrices of survivorship proportions S_x. These are based on the proportion of survivers L_x, using

$$S_x = L_{x+5} L_x^{-1}$$

These age-specific surivivorship proportions for both regions are represented in Table 2.9 (pg. 46-47), replicated below.

```r
S <- survivor_prop(L.dur)

# survivorship proportions for Slov.
state_table(S, 1, head(ages, -1), states)
```

```
##    slovenia  r.yugos
## 0    0.9709 0.012622
## 5    0.9888 0.009392
## 10   0.9816 0.016308
## 15   0.9668 0.030062
## 20   0.9653 0.031117
## 25   0.9729 0.022849
## 30   0.9808 0.013638
## 35   0.9853 0.006854
## 40   0.9826 0.003827
## 45   0.9754 0.002925
## 50   0.9645 0.003098
## 55   0.9457 0.003512
## 60   0.9048 0.004072
## 65   0.8319 0.003743
## 70   0.7372 0.003589
## 75   0.5908 0.002983
## 80   0.6171 0.007229
```

```
# survivorship proportions for r. Yugos.
state_table(S, 2, head(ages, -1), states)
```

```
##     slovenia r.yugos
## 0  0.0010638  0.9412
## 5  0.0008022  0.9963
## 10 0.0020513  0.9946
## 15 0.0039405  0.9909
## 20 0.0035411  0.9895
## 25 0.0021072  0.9896
## 30 0.0014229  0.9891
## 35 0.0010097  0.9867
## 40 0.0006821  0.9826
## 45 0.0005431  0.9742
## 50 0.0008010  0.9601
## 55 0.0009545  0.9356
## 60 0.0008019  0.8948
## 65 0.0004919  0.8302
## 70 0.0003183  0.7452
## 75 0.0004107  0.6403
## 80 0.0008529  1.0030
```

2.7 Estimation of Age-specific Outmigration and Death Probabilities

There are tree options for calculating the probabilities of dying and outmigrating (based on the observed transition rates):

- Option 1: No multiple transitions (Rogers, 1975)
- Option 2: Rogers, 1975 (mentioned, not used)
- Option 3: Allows multiple transitions, used.

We start by constructing M_x, containing observed outmigration and death rates. Based on this matrix, we can calculated P_x using

$$P_x = (I + 5/2M_x)^{-1}(I - 5/2M_x)$$

The results are presented in section 2, table 2.1 (supra).

3. Multiregional Population Projection

3.1 The Growth Matrix

The growth matrix G is a generalized Leslie matrix, constructed from submatrices S_x, containing the survivorship proportions (calculated in section 2.6) and $B_s*, containing the fertility rates. E.g, in this two-region example:

$$M_t = \begin{bmatrix} 0 & 0 & B_{\alpha-5} & B_\alpha & B_{\alpha+5} & B_{\dots} & B_{\beta-5} & 0 & 0 \\ S_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & S_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & S_{10} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{15} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{20} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{25} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{\dots} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & S_{z-5} & 0 \end{bmatrix}$$

with

$$S_x = \begin{bmatrix} s_{11}^x & s_{21}^x \\ s_{12}^x & s_{22}^x \end{bmatrix}$$

$$B_x = \begin{bmatrix} b_{11}^x & b_{21}^x \\ b_{12}^x & b_{22}^x \end{bmatrix}$$

While we have already the S_x submatrices, we still need to calculate B_x, based on the observed fertilitary rates F_x, the transition probabilites P_x and the survivorship proportions S_x, using

$$B_x = \frac{5}{4}(P_0 + I)(Fx + 5S_x)$$

```
B <- birth_prop(FR, P, S)

# birth proportions Slov.
round(state_table(B, 1, head(ages, -1), states), 6)
```

```
##    slovenia  r.yugos
## 0  0.000000 0.000000
## 5  0.000171 0.000003
## 10 0.038234 0.001277
## 15 0.205783 0.007635
## 20 0.321959 0.007623
## 25 0.252357 0.004084
## 30 0.155327 0.001800
## 35 0.074685 0.000696
## 40 0.020771 0.000159
## 45 0.002433 0.000021
## 50 0.000715 0.000005
## 55 0.000000 0.000000
## 60 0.000000 0.000000
## 65 0.000000 0.000000
## 70 0.000000 0.000000
## 75 0.000000 0.000000
## 80 0.000000 0.000000
```

```
# birth proportions r. Yugo.
round(state_table(B, 2, head(ages, -1), states), 6)
```

```
##    slovenia  r.yugos
## 0  0.000000 0.000000
## 5  0.000000 0.000157
## 10 0.000121 0.062407
## 15 0.000860 0.268805
## 20 0.000802 0.381933
## 25 0.000398 0.279343
## 30 0.000186 0.159828
## 35 0.000075 0.083796
## 40 0.000024 0.033507
```

```
## 45 0.000005 0.006732
## 50 0.000001 0.001689
## 55 0.000000 0.000000
## 60 0.000000 0.000000
## 65 0.000000 0.000000
## 70 0.000000 0.000000
## 75 0.000000 0.000000
## 80 0.000000 0.000000
```

```
# TODO: (rounding) error last two values RYU?
```

We construct G by combining B_x and S_x, resulting in a 36x36 matrix.

```
G <- projection_matrix(B, S)
dim(G)
```

```
## [1] 36 36
```

3.2 The projection process

The initial population is the observed population at t_0, formated as a vector of length 36 with 2 regions nested within 18 age-groups.

```
n0 <- ggplot2:::interleave(SL[, 1], RYU[, 1])
length(n0)
```

```
## [1] 36
```

we recursively multiply G and $n0$, projecting the population forwards for 8 steps.

```
result <- project(init = n0, pmat = G, nsteps = 8)

proj.slov <- t(result[, seq(1, ncol(result), 2)])
proj.ryog <- t(result[, seq(2, ncol(result), 2)])
rownames(proj.slov) <- rownames(proj.ryog) <- seq(0, 85, 5)
colnames(proj.slov) <- colnames(proj.ryog) <- seq(1961, 2001, 5)

proj.slov  # Projection for Solvenia, 1961-2001
```

```
##     1961  1966  1971  1976  1981  1986  1991  1996  2001
## 0  67800 69924 71442 74584 76613 77195 77880 79111 81001
## 5  74100 66731 68846 70341 73453 75468 76052 76742 77972
```

```
## 10 70700 73995 66623 68752 70245 73366 75390 75981 76680
## 15 60100 71060 74488 67035 69220 70723 73899 75967 76582
## 20 62900 60535 71869 75554 67932 70228 71753 75036 77190
## 25 66500 63284 60604 72204 76091 68363 70742 72278 75638
## 30 67100 66331 63088 60247 71921 75898 68160 70572 72104
## 35 62900 66848 66149 62892 59949 71660 75690 67954 70384
## 40 39500 62615 66593 65944 62681 59672 71393 75456 67731
## 45 47900 39079 61949 65917 65305 62063 59034 70673 74725
## 50 51300 46957 38325 60757 64674 64097 60907 57895 69342
## 55 46100 49841 45630 37265 59078 62922 62395 59278 56291
## 60 39600 43968 47550 43543 35586 56419 60131 59666 56672
## 65 29500 36090 40073 43349 39703 32467 51477 54896 54501
## 70 21700 24653 30165 33496 36240 33196 27157 43059 45935
## 75 14400 16054 18235 22314 24778 26811 24561 20099 31869
## 80  7100  8557  9539 10831 13256 14721 15932 14597 11951
## 85  3600  4434  5346  5960  6762  8279  9195  9955  9124
```

```
proj.ryog  # Projection for rest of Yugoslavia, 1961-2001.
```

```
##      1961   1966   1971   1976    1981    1986    1991    1996    2001
## 0  847900 897654 917142 975042 1017002 1035190 1058011 1091063 1130953
## 5  905200 798890 845744 864106  918640  958158  975284  996771 1027894
## 10 808100 902577 796588 843290  861598  915962  955354  972422  993837
## 15 617400 804890 898911 793374  839859  858092  912214  951426  968412
## 20 725500 613569 799676 892942  788144  834270  852382  906105  945021
## 25 774000 719826 608999 793502  885901  781968  827681  845650  898909
## 30 728400 767460 713777 604042  786889  878415  775387  820679  838496
## 35 633300 721349 759971 706831  598257  779264  869843  767837  812666
## 40 392400 625319 712226 750330  697873  590722  769404  858808  758104
## 45 437100 385715 614664 700072  737510  685955  580659  756273  844134
## 50 453800 425956 375872 598978  682192  718662  668428  565841  736956
## 55 389300 435875 409127 361012  575297  655207  690222  641980  543472
## 60 325800 364372 407958 382920  337876  538427  613201  645957  600813
## 65 230600 291703 326237 365254  342832  302492  482040  548966  578276
## 70 180000 191560 242314 270999  303405  284776  251258  400394  455970
## 75 120900 134208 142833 180672  202060  226218  212325  187327  298515
## 80  61200  77450  85975  91504  115743  129444  144917  136015  119997
## 85  39300  61436  77746  86304   91859  116188  129941  145470  136532
```

Replication final subtable table 3.2, pg. 67:

Projected population distribution 2001

```
proj.t01 <- cbind(proj.slov[, 9], proj.ryog[, 9])
proj.t01 <- cbind(rowSums(proj.t01), proj.t01)
```

```
colnames(proj.t01) <- c("total", "slovenia", "r.yogos")
round(proj.t01)
```

```
##      total slovenia r.yogos
## 0  1211954    81001 1130953
## 5  1105866    77972 1027894
## 10 1070518    76680  993837
## 15 1044993    76582  968412
## 20 1022211    77190  945021
## 25  974547    75638  898909
## 30  910599    72104  838496
## 35  883049    70384  812666
## 40  825834    67731  758104
## 45  918859    74725  844134
## 50  806298    69342  736956
## 55  599763    56291  543472
## 60  657485    56672  600813
## 65  632777    54501  578276
## 70  501906    45935  455970
## 75  330384    31869  298515
## 80  131947    11951  119997
## 85  145656     9124  136532
```

Projected percentage population distribution 2001

```
round(prop.table(proj.t01, 2) * 100, 4)
```

```
##     total slovenia r.yogos
## 0  8.7984   7.4607  8.9129
## 5  8.0283   7.1818  8.1007
## 10 7.7717   7.0628  7.8323
## 15 7.5864   7.0537  7.6319
## 20 7.4210   7.1097  7.4476
## 25 7.0749   6.9668  7.0842
## 30 6.6107   6.6413  6.6081
## 35 6.4107   6.4829  6.4045
## 40 5.9953   6.2385  5.9745
## 45 6.6707   6.8827  6.6525
## 50 5.8535   6.3869  5.8079
## 55 4.3541   5.1848  4.2830
## 60 4.7732   5.2199  4.7349
## 65 4.5938   5.0199  4.5573
## 70 3.6437   4.2310  3.5934
## 75 2.3985   2.9354  2.3526
## 80 0.9579   1.1007  0.9457
## 85 1.0574   0.8404  1.0760
```

3.3 The stable equivalent population

We can approximate the stable equivalent to the original population by projecting $n0$ foward a sufficiently large number of steps. This reproduces the percentage distribution of the stable equivalent population, shown in Table 3.3. (p. 70).

```
round(stablepop_pct(n0, G) * 100, 4)
```

```
##         [,1]  [,2]
##  [1,] 7.5419 8.860
##  [2,] 7.2574 8.094
##  [3,] 7.0671 7.826
##  [4,] 6.9926 7.556
##  [5,] 7.0459 7.275
##  [6,] 7.0198 6.995
##  [7,] 6.8666 6.723
##  [8,] 6.6895 6.455
##  [9,] 6.5002 6.181
## [10,] 6.2642 5.892
## [11,] 5.9789 5.569
## [12,] 5.6665 5.187
## [13,] 5.2791 4.708
## [14,] 4.6948 4.088
## [15,] 3.8212 3.293
## [16,] 2.7494 2.381
## [17,] 1.5915 1.479
## [18,] 0.9734 1.440
```

Custom functions

Multistate lifetable functions

```
transfer_matrix
```

```
## function (observed_rates, multiple = TRUE, absorbing_state = "last")
## {
##     n_states <- ncol(observed_rates) - 3 - 1
##     n_ages <- nrow(observed_rates)/n_states
##     ages <- unique(observed_rates[, 1])
##     if (absorbing_state == "last") {
##         absorbing <- n_states + 1
##     }
##     Md.cols <- matrix(observed_rates[, 3 + n_states + 1], ncol = n_states)
```

```
##     Mx.cols <- observed_rates[, 4:(3 + n_states)]
##     Md <- list()
##     for (i in 1:n_ages) {
##         Md[[i]] <- diag(Md.cols[i, ])
##     }
##     Mx <- list()
##     i <- 1
##     for (x in ages) {
##         Mx[[i]] <- t(Mx.cols[observed_rates[, 1] == x, ])
##         i <- i + 1
##     }
##     Mx_it <- lapply(lapply(Mx, colSums), diag)
##     M <- list()
##     for (i in 1:n_ages) {
##         M[[i]] <- Md[[i]] + Mx_it[[i]]
##         M[[i]] <- M[[i]] + (Mx[[i]] * -1)
##     }
##     if (!multiple) {
##         M[[n_ages]] <- Md[[n_ages]]
##     }
##     M
## }
```

```
transfer_prob
```

```
## function (transfer_rates, multiple = TRUE)
## {
##     n_ages <- length(transfer_rates)
##     n_states <- ncol(transfer_rates[[1]])
##     I <- diag(rep(1, n_states))
##     if (!multiple) {
##         P <- lapply(transfer_rates, function(Mx) {
##             Mx_t <- t(Mx)
##             Mx_t.diag <- diag(diag(Mx_t))
##             t(I - 5 * solve(I + 2.5 * Mx_t.diag) %*% Mx_t)
##         })
##     }
##     if (multiple) {
##         P <- lapply(transfer_rates, function(Mx) {
##             solve(I + 2.5 * Mx) %*% (I - 2.5 * Mx)
##         })
##     }
##     P[[n_ages]] <- P[[n_ages]] * 0
##     P
## }
```

```
expected_survivors
```

```
## function (transition_probs, radix = 1e+05)
## {
##     n_state <- ncol(transition_probs[[1]])
##     l0 <- diag(rep(radix, n_state))
##     L <- list(l0)
##     for (x in 1:(length(transition_probs) - 1)) {
##         L[[x + 1]] <- transition_probs[[x]] %*% L[[x]]
##     }
##     L
## }
```

```
years_lived
```

```
## function (expected_survivors, transfer_rates)
## {
##     L.surv <- expected_survivors
##     Mx <- transfer_rates
##     n_ages <- length(L.surv)
##     L.surv <- expected_survivors
##     L.dur <- list()
##     for (x in 1:(n_ages - 1)) {
##         L.dur[[x]] <- 2.5 * (L.surv[[x]] + L.surv[[x + 1]]) %*%
##             solve(L.surv[[1]])
##     }
##     L.dur[[n_ages]] <- solve(Mx[[n_ages]]) %*% L.surv[[n_ages]] %*%
##         solve(L.surv[[1]])
##     L.dur
## }
```

```
survivor_prop
```

```
## function (years_lived)
## {
##     S <- list()
##     for (x in 1:(length(years_lived) - 1)) {
##         S[[x]] <- years_lived[[x + 1]] %*% solve(years_lived[[x]])
##     }
##     S
## }
```

```
birth_prop
```

```
## function (birth_rates, transition_probs, survivor_prop)
## {
##     n_ages <- length(transition_probs)
##     n_states <- ncol(transition_probs[[1]])
##     I <- diag(rep(1, n_states))
##     P <- transition_probs
##     S <- survivor_prop
##     F <- matrix(birth_rates, ncol = n_states)
##     B <- list()
##     for (x in 1:(n_ages - 1)) {
##         Fx <- diag(unlist(F[x, 1:n_states]))
##         Fx5 <- diag(unlist(F[x + 1, 1:n_states]))
##         B[[x]] <- 5/4 * (P[[1]] + I) %*% (Fx + Fx5 %*% S[[x]])
##     }
##     B
## }
```

Projection functions

```
projection_matrix
```

```
## function (fertility, survivorship)
## {
##     B <- fertility
##     S <- survivorship
##     n_ages <- length(B) + 1
##     n_states <- ncol(B[[1]])
##     G <- diag(rep(0, n_ages * n_states))
##     j <- 1
##     i_start <- head(seq(1, (n_ages * n_states), n_states), -1)
##     mwidth <- n_states - 1
##     for (i in i_start) {
##         G[i:(i + mwidth), i:(i + mwidth)] <- S[[j]]
##         j <- j + 1
##     }
##     B[[(n_ages)]] <- diag(rep(0, n_states))
##     G <- rbind(do.call(cbind, B), G)
##     G <- G[1:(n_ages * n_states), ]
##     G
## }
```

```
project
```

```
## function (init, pmat, nsteps, lbls = NULL)
```

```
## {
##     nclasses <- length(init)
##     pops <- matrix(nrow = nsteps + 1, ncol = nclasses)
##     colnames(pops) <- lbls
##     rownames(pops) <- paste("t", seq(0, nsteps), sep = "")
##     pops[1, ] <- init
##     i <- nsteps
##     n <- init
##     while (i > 0) {
##         n <- pmat %*% n
##         pops[nsteps + 2 - i, ] <- n
##         i <- i - 1
##     }
##     pops
## }
```

```
plot_proj
```

```
## function (proj_result)
## {
##     p_num <- melt(proj_result)
##     p_num$type <- rep("num", nrow(p_num))
##     p_prop <- proj_result/rowSums(proj_result)
##     p_prop <- melt(p_prop)
##     p_prop$type <- rep("prop", nrow(p_prop))
##     p <- rbind(p_num, p_prop)
##     names(p) <- c("tlabel", "class", "value", "type")
##     p$time <- as.integer(str_replace(p$tlabel, "t", ""))
##     q <- ggplot(p, aes(x = time, y = value, group = class, colour = class))
##     q <- q + geom_line() + facet_grid(. ~ type)
##     q
## }
```

Helper functions

```
state_table
```

```
## function (data, nstate, rlbl, clbl)
## {
##     tab <- do.call(rbind, lapply(data, function(x) x[, nstate]))
##     if (missing(rlbl)) {
##         rlbl <- 1:length(data)
##     }
##     if (missing(clbl)) {
```

```
##         clbl <- colnames(data[[1]])
##     }
##     rownames(tab) <- rlbl
##     colnames(tab) <- clbl
##     tab
## }
```

```
collapse_interval
```

```
## function (df, interval = 5)
## {
##     max_age <- nrow(df)
##     ages <- seq(0, max_age, interval)
##     l <- list()
##     for (i in ages) {
##         l <- cbind(l, colSums(df[i:(i + (interval - 1)), ]))
##     }
##     df_i <- data.frame(matrix(unlist(l), nrow = length(ages),
##         byrow = T))
##     rownames(df_i) <- paste(ages, ages + (interval - 1), sep = "-")
##     colnames(df_i) <- colnames(df)
##     df_i
## }
```