Solution assignment 1

Jonas Schöley

September 19, 2017

First we need some data. We can go to http://www.mortality.org to download the data manually or use the `HMDHFDplus` package to download the data from within R. Here I choose to use the package. Before we can load the package via `library(HMDHFDplus)` we need to install it by running `install.packages('HMDHFDplus')`.

We use the function `readHMDweb()` to download time series of death counts and exposures for Sweden. You need to specify your own HMD username and password to the function.

```
library(HMDHFDplus)
# http://www.mortality.org/hmd/SWE/STATS/Deaths_1x1.txt
deaths <- readHMDweb(CNTRY = 'SWE', item = 'Deaths_1x1',
                     username = usr, password = pwd)
str(deaths)
```

```
## 'data.frame':    29304 obs. of  6 variables:
##  $ Year        : int  1751 1751 1751 1751 1751 1751 1751 1751 1751 1751 ...
##  $ Age         : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Female      : num  5988 1286 835 622 471 ...
##  $ Male        : num  6902 1360 882 655 498 ...
##  $ Total       : num  12890 2646 1717 1277 969 ...
##  $ OpenInterval: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
# http://www.mortality.org/hmd/SWE/STATS/Exposures_1x1.txt
exposures <- readHMDweb(CNTRY = 'SWE', item = 'Exposures_1x1',
                        username = usr, password = pwd)
str(exposures)
```

```
## 'data.frame':    29304 obs. of  6 variables:
##  $ Year        : int  1751 1751 1751 1751 1751 1751 1751 1751 1751 1751 ...
##  $ Age         : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Female      : num  28214 26035 25880 23918 19876 ...
##  $ Male        : num  28627 25683 25504 23501 19373 ...
##  $ Total       : num  56841 51718 51385 47419 39250 ...
##  $ OpenInterval: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Both data series have exactly the same format: 6 variables, 29,304 rows, ages 0 to 110+ for years 1751 to 2014. This consistency makes it really easy to transform the data frame to a matrix. But why would we want to do so? Matrices with ages as rows and periods as columns are a convenient structure to store time series of life-table data. Calculating statistics for each year simply requires to loop over the columns of such a matrix.

```
years <- unique(deaths$Year) # labels for the years
ages <- 0:110 # labels for the age groups
w <- length(ages) # number of age groups

# create age-period matrices of death counts and exposures
D <- matrix(deaths$Total, nrow = w, dimnames = list(ages, years))
E <- matrix(exposures$Total, nrow = w, dimnames = list(ages, years))
```

The highest age group in our data is 110+. This makes sense for today's life-tables but historically the maximum age at death in any given year was often considerably lower. In consequence we have exposures and death counts of 0 in the highest ages for some years. Here's an example for the year 1800.

```r
D[90:111, '1800']
```

```
##     89     90     91     92     93     94     95     96     97     98
## 104.58  91.92  68.14  49.39  34.99  24.23  16.39  10.83   6.97   4.40
##     99    100    101    102    103    104    105    106    107    108
##   2.70   1.39   0.65   0.00   0.00   0.00   0.00   0.00   0.00   0.00
##    109    110
##   0.00   0.00
```

```r
E[90:111, '1800']
```

```
##     89     90     91     92     93     94     95     96     97     98
## 301.05 213.68 146.97  98.54  64.46  41.12  25.52  14.79   8.20   4.47
##     99    100    101    102    103    104    105    106    107    108
##   2.03   0.63   0.15   0.00   0.00   0.00   0.00   0.00   0.00   0.00
##    109    110
##   0.00   0.00
```

Zero exposures will cause all kinds of headaches when constructing a life-table. One technique to deal with this issue is to aggregate deaths and exposures over the highest ages, i.e. instead of an age group 110+ we can have 100+ as the last age group.

```r
# sum up deaths and exposures past age 100
D100p <- colSums(D[101:w,])
E100p <- colSums(E[101:w,])

# remove single ages 100 and beyond from exposure and death matrices
D <- D[-(101:w),]
E <- E[-(101:w),]

# add open age group 100+ to exposure and death matrices
D <- rbind(D, D100p)
E <- rbind(E, E100p)
```

What we have now is age-period matrices of death counts and exposures over ages 0 to 100+. We calculate the age and period specific mortality rates (mortality rate matrix M) by dividing deaths by exposures.

```r
M <- D/E
```

Now it's time to write a function calculating life-expectancy and the coefficient of variation for the life-table death distribution. Concerning the life-table construction we follow Preston (2001), p. 49, Box 3.1. As the coefficient of variation for the life-table death distribution depends on life-table statistics, we calculate it inside of our life-table function.

```r
LTFun <- function (x, nmx) {
  # clear names of input vectors
  names(nmx) <- NULL; names(x) <- NULL

  # number of age groups
  w <- length(x)
  # width of age groups
  nx <- c(diff(x), NA)
  # time spent in age group if dying in age group
  nax <- 0.5*nx
  nax[w] <- 1/nmx[w]
  # probability of dying in age group given
  # survival until start of age group
```

```r
  nqx = nx*nmx / (1 + (nx-nax)*nmx)
  nqx[length(nqx)] <- 1
  # probability of surviving age group given
  # survival until start of age group
  npx <- 1 - nqx
  # probability of surviving until start
  # of age group
  lx <- cumprod(c(1, npx[-w]))
  # unconditional probability of dying in
  # age group
  ndx <- c(-diff(lx), lx[w])
  # total exposure time spend in age group
  # by life-table population
  nLx <- c(lx[-1], 0)*nx + ndx*nax
  nLx[w] <- nax[w]*lx[w]
  # total exposure time yet to live by life-table
  # population
  Tx <- rev(cumsum(rev(nLx)))
  # remaining life-expectancy past age x
  ex <- Tx/lx

  # total life-expectancy
  e0 <- ex[1]
  # coefficient of variation of life-table
  # distribution of deaths
  cv <- sqrt(sum(ndx*(x+nax-e0)^2)) / e0

  # function output
  list(
    lt = data.frame(x, nx, nmx, nax, nqx, npx, lx, ndx, nLx, Tx, ex),
    lt_summary = c(e0 = e0, cv = cv)
  )
}
```

We can test this function on the first year in our data.

```r
LTFun(0:100, M[,1])
```

```
## $lt
##       x nx         nmx     nax         nqx       npx           lx
## 1     0  1 0.226774404 0.50000 0.203679730 0.7963203 1.0000000000
## 2     1  1 0.051168600 0.50000 0.049892145 0.9501079 0.7963202705
## 3     2  1 0.033408706 0.50000 0.032859804 0.9671402 0.7565901443
## 4     3  1 0.026935855 0.50000 0.026577906 0.9734221 0.7317287401
## 5     4  1 0.024681101 0.50000 0.024380235 0.9756198 0.7122809225
## 6     5  1 0.019766134 0.50000 0.019572695 0.9804273 0.6949153460
## 7     6  1 0.013787682 0.50000 0.013693282 0.9863067 0.6813139796
## 8     7  1 0.009457109 0.50000 0.009412601 0.9905874 0.6719845550
## 9     8  1 0.006660414 0.50000 0.006638307 0.9933617 0.6656594322
## 10    9  1 0.005420721 0.50000 0.005406068 0.9945939 0.6612405803
## 11   10  1 0.005609643 0.50000 0.005593953 0.9944060 0.6576658685
## 12   11  1 0.006121171 0.50000 0.006102494 0.9938975 0.6539869166
## 13   12  1 0.006511421 0.50000 0.006490290 0.9935097 0.6499959654
## 14   13  1 0.006738204 0.50000 0.006715578 0.9932844 0.6457773030
## 15   14  1 0.006752224 0.50000 0.006729504 0.9932705 0.6414405351
```

```
## 16    15   1 0.006428534 0.50000 0.006407937 0.9935921 0.6371239581
## 17    16   1 0.006047135 0.50000 0.006028906 0.9939711 0.6330413078
## 18    17   1 0.005906809 0.50000 0.005889415 0.9941106 0.6292247612
## 19    18   1 0.006010376 0.50000 0.005992367 0.9940076 0.6255189951
## 20    19   1 0.006298215 0.50000 0.006278444 0.9937216 0.6217706555
## 21    20   1 0.006771070 0.50000 0.006748224 0.9932518 0.6178669034
## 22    21   1 0.007296608 0.50000 0.007270085 0.9927299 0.6136973992
## 23    22   1 0.007736466 0.50000 0.007706655 0.9922933 0.6092357672
## 24    23   1 0.008029954 0.50000 0.007997843 0.9920022 0.6045405974
## 25    24   1 0.008271070 0.50000 0.008237005 0.9917630 0.5997055767
## 26    25   1 0.008446224 0.50000 0.008410705 0.9915893 0.5947657988
## 27    26   1 0.008592461 0.50000 0.008555704 0.9914443 0.5897633992
## 28    27   1 0.008860153 0.50000 0.008821075 0.9911789 0.5847175581
## 29    28   1 0.009238528 0.50000 0.009196049 0.9908040 0.5795597206
## 30    29   1 0.009728055 0.50000 0.009680966 0.9903190 0.5742300610
## 31    30   1 0.010559577 0.50000 0.010504118 0.9894959 0.5686709592
## 32    31   1 0.011550715 0.50000 0.011484388 0.9885156 0.5626975725
## 33    32   1 0.012362153 0.50000 0.012286211 0.9877138 0.5562353351
## 34    33   1 0.012876962 0.50000 0.012794584 0.9872054 0.5494013105
## 35    34   1 0.012971218 0.50000 0.012887634 0.9871124 0.5423719491
## 36    35   1 0.012226992 0.50000 0.012152696 0.9878473 0.5353820579
## 37    36   1 0.011230847 0.50000 0.011168133 0.9888319 0.5288757224
## 38    37   1 0.010680310 0.50000 0.010623579 0.9893764 0.5229691678
## 39    38   1 0.010606182 0.50000 0.010550233 0.9894498 0.5174133638
## 40    39   1 0.011050591 0.50000 0.010989869 0.9890101 0.5119545324
## 41    40   1 0.012473023 0.50000 0.012395717 0.9876043 0.5063282192
## 42    41   1 0.014404399 0.50000 0.014301397 0.9856986 0.5000519181
## 43    42   1 0.016012235 0.50000 0.015885058 0.9841149 0.4929004771
## 44    43   1 0.017259635 0.50000 0.017111962 0.9828880 0.4850707246
## 45    44   1 0.018164015 0.50000 0.018000534 0.9819995 0.4767702126
## 46    45   1 0.017631463 0.50000 0.017477387 0.9825226 0.4681880940
## 47    46   1 0.016192501 0.50000 0.016062456 0.9839375 0.4600053895
## 48    47   1 0.015218651 0.50000 0.015103722 0.9848963 0.4526165733
## 49    48   1 0.014699593 0.50000 0.014592342 0.9854077 0.4457803786
## 50    49   1 0.014690661 0.50000 0.014583540 0.9854165 0.4392753990
## 51    50   1 0.016263498 0.50000 0.016132314 0.9838677 0.4328692088
## 52    51   1 0.019128983 0.50000 0.018947758 0.9810522 0.4258860268
## 53    52   1 0.022064302 0.50000 0.021823541 0.9781765 0.4178164416
## 54    53   1 0.024842156 0.50000 0.024537375 0.9754626 0.4086982073
## 55    54   1 0.027160869 0.50000 0.026796955 0.9732030 0.3986698262
## 56    55   1 0.026946452 0.50000 0.026588223 0.9734118 0.3879866887
## 57    56   1 0.025450854 0.50000 0.025131050 0.9748689 0.3776708123
## 58    57   1 0.025132208 0.50000 0.024820314 0.9751797 0.3681795480
## 59    58   1 0.025734980 0.50000 0.025408042 0.9745920 0.3590412162
## 60    59   1 0.027155129 0.50000 0.026791368 0.9732086 0.3499186819
## 61    60   1 0.030655627 0.50000 0.030192837 0.9698072 0.3405438818
## 62    61   1 0.035037345 0.50000 0.034434105 0.9655659 0.3302618959
## 63    62   1 0.039024625 0.50000 0.038277738 0.9617223 0.3188896229
## 64    63   1 0.043018911 0.50000 0.042113081 0.9578869 0.3066832496
## 65    64   1 0.046620443 0.50000 0.045558465 0.9544415 0.2937678730
## 66    65   1 0.048371720 0.50000 0.047229436 0.9527706 0.2803842596
## 67    66   1 0.049087418 0.50000 0.047911492 0.9520885 0.2671418692
## 68    67   1 0.050508626 0.50000 0.049264485 0.9507355 0.2543427036
## 69    68   1 0.052994078 0.50000 0.051626138 0.9483739 0.2418126413
```

```
## 70   69  1 0.056927854 0.50000 0.055352310 0.9446477 0.2293287885
## 71   70  1 0.064687268 0.50000 0.062660597 0.9373394 0.2166349103
## 72   71  1 0.075756151 0.50000 0.072991378 0.9270086 0.2030604375
## 73   72  1 0.087422487 0.50000 0.083761182 0.9162388 0.1882387764
## 74   73  1 0.099921124 0.50000 0.095166549 0.9048335 0.1724716741
## 75   74  1 0.112695743 0.50000 0.106684310 0.8933157 0.1560581400
## 76   75  1 0.116971061 0.50000 0.110507945 0.8894921 0.1394091851
## 77   76  1 0.113354284 0.50000 0.107274284 0.8927257 0.1240033625
## 78   77  1 0.110908148 0.50000 0.105080980 0.8949190 0.1107009905
## 79   78  1 0.110130738 0.50000 0.104382857 0.8956171 0.0990684220
## 80   79  1 0.114108805 0.50000 0.107949794 0.8920502 0.0887273770
## 81   80  1 0.128260135 0.50000 0.120530506 0.8794695 0.0791492750
## 82   81  1 0.148429341 0.50000 0.138174748 0.8618253 0.0696093729
## 83   82  1 0.168770237 0.50000 0.155636806 0.8443632 0.0599911153
## 84   83  1 0.188088353 0.50000 0.171920254 0.8280797 0.0506542897
## 85   84  1 0.202787763 0.50000 0.184119202 0.8158808 0.0419457914
## 86   85  1 0.208951957 0.50000 0.189186511 0.8108135 0.0342227657
## 87   86  1 0.212423347 0.50000 0.192027758 0.8079722 0.0277482801
## 88   87  1 0.215387989 0.50000 0.194447194 0.8055528 0.0224198401
## 89   88  1 0.219297659 0.50000 0.197627982 0.8023720 0.0180603651
## 90   89  1 0.227782307 0.50000 0.204492429 0.7955076 0.0144911316
## 91   90  1 0.282923355 0.50000 0.247860581 0.7521394 0.0115278049
## 92   91  1 0.302984205 0.50000 0.263123129 0.7368769 0.0086705165
## 93   92  1 0.324537269 0.50000 0.279227417 0.7207726 0.0063891030
## 94   93  1 0.347443352 0.50000 0.296018519 0.7039815 0.0046050903
## 95   94  1 0.370550393 0.50000 0.312628151 0.6873718 0.0032418983
## 96   95  1 0.393939394 0.50000 0.329113924 0.6708861 0.0022283896
## 97   96  1 0.420423892 0.50000 0.347396911 0.6526031 0.0014949956
## 98   97  1 0.452575488 0.50000 0.369061414 0.6309386 0.0009756387
## 99   98  1 0.484510533 0.50000 0.390024938 0.6099751 0.0006155681
## 100  99  1 0.518063837 0.50000 0.411477922 0.5885221 0.0003754812
## 101 100 NA 0.705835649 1.41676 1.000000000 0.0000000 0.0002209790
##              ndx          nLx           Tx        ex
## 1   0.2036797295 0.8981601352 3.810573e+01 38.105729
## 2   0.0397301262 0.7764552074 3.720757e+01 46.724378
## 3   0.0248614042 0.7441594422 3.643111e+01 48.151716
## 4   0.0194478175 0.7220048313 3.568695e+01 48.770743
## 5   0.0173655765 0.7035981343 3.496495e+01 49.088707
## 6   0.0136013664 0.6881146628 3.426135e+01 49.302914
## 7   0.0093294246 0.6766492673 3.357324e+01 49.277188
## 8   0.0063251228 0.6688219936 3.289659e+01 48.954381
## 9   0.0044188519 0.6634500063 3.222777e+01 48.414796
## 10  0.0035747119 0.6594532244 3.156432e+01 47.734995
## 11  0.0036789518 0.6558263926 3.090486e+01 46.991738
## 12  0.0039909513 0.6519914410 3.024904e+01 46.253274
## 13  0.0042186623 0.6478866342 2.959704e+01 45.534197
## 14  0.0043367680 0.6436089190 2.894916e+01 44.828392
## 15  0.0043165769 0.6392822466 2.830555e+01 44.128095
## 16  0.0040826504 0.6350826330 2.766627e+01 43.423680
## 17  0.0038165466 0.6311330345 2.703118e+01 42.700506
## 18  0.0037057660 0.6273718781 2.640005e+01 41.956472
## 19  0.0037483397 0.6236448253 2.577268e+01 41.202073
## 20  0.0039037521 0.6198187794 2.514903e+01 40.447445
## 21  0.0041695042 0.6157821513 2.452922e+01 39.699838
```

```
## 22  0.0044616320 0.6114665832 2.391343e+01 38.966164
## 23  0.0046951698 0.6068881823 2.330197e+01 38.247864
## 24  0.0048350207 0.6021230870 2.269508e+01 37.541033
## 25  0.0049397779 0.5972356877 2.209296e+01 36.839670
## 26  0.0050023995 0.5922645990 2.149572e+01 36.141486
## 27  0.0050458411 0.5872404787 2.090346e+01 35.443799
## 28  0.0051578375 0.5821386394 2.031621e+01 34.745348
## 29  0.0053296596 0.5768948908 1.973408e+01 34.050117
## 30  0.0055591019 0.5714505101 1.915718e+01 33.361509
## 31  0.0059733867 0.5656842659 1.858573e+01 32.682750
## 32  0.0064622374 0.5594664538 1.802005e+01 32.024390
## 33  0.0068340246 0.5528183228 1.746058e+01 31.390635
## 34  0.0070293615 0.5458866298 1.690776e+01 30.774884
## 35  0.0069898912 0.5388770035 1.636188e+01 30.167259
## 36  0.0065063355 0.5321288901 1.582300e+01 29.554592
## 37  0.0059065546 0.5259224451 1.529087e+01 28.912027
## 38  0.0055558040 0.5201912658 1.476495e+01 28.232920
## 39  0.0054588314 0.5146839481 1.424476e+01 27.530707
## 40  0.0056263132 0.5091413758 1.373007e+01 26.818928
## 41  0.0062763011 0.5031900686 1.322093e+01 26.111383
## 42  0.0071514410 0.4964761976 1.271774e+01 25.432840
## 43  0.0078297525 0.4889856008 1.222126e+01 24.794587
## 44  0.0083005120 0.4809204686 1.173228e+01 24.186738
## 45  0.0085821186 0.4724791533 1.125136e+01 23.599121
## 46  0.0081827045 0.4640967417 1.077888e+01 23.022539
## 47  0.0073888162 0.4563109814 1.031478e+01 22.423177
## 48  0.0068361947 0.4491984760 9.858471e+00 21.781065
## 49  0.0065049796 0.4425278888 9.409273e+00 21.107418
## 50  0.0064061902 0.4360723039 8.966745e+00 20.412581
## 51  0.0069831819 0.4293776178 8.530672e+00 19.707275
## 52  0.0080695852 0.4218512342 8.101295e+00 19.022213
## 53  0.0091182342 0.4132573245 7.679444e+00 18.379946
## 54  0.0100283812 0.4036840168 7.266186e+00 17.778855
## 55  0.0106831374 0.3933282575 6.862502e+00 17.213498
## 56  0.0103158764 0.3828287505 6.469174e+00 16.673701
## 57  0.0094912642 0.3729251802 6.086345e+00 16.115477
## 58  0.0091383319 0.3636103821 5.713420e+00 15.518027
## 59  0.0091225343 0.3544799490 5.349810e+00 14.900266
## 60  0.0093748001 0.3452312818 4.995330e+00 14.275687
## 61  0.0102819859 0.3354028888 4.650098e+00 13.654917
## 62  0.0113722730 0.3245757594 4.314696e+00 13.064467
## 63  0.0122063733 0.3127864363 3.990120e+00 12.512542
## 64  0.0129153766 0.3002255613 3.677333e+00 11.990656
## 65  0.0133836134 0.2870760663 3.377108e+00 11.495838
## 66  0.0132423904 0.2737630644 3.090032e+00 11.020703
## 67  0.0127991656 0.2607422864 2.816269e+00 10.542221
## 68  0.0125300623 0.2480776724 2.555526e+00 10.047571
## 69  0.0124838528 0.2355707149 2.307449e+00  9.542300
## 70  0.0126938782 0.2229818494 2.071878e+00  9.034531
## 71  0.0135744728 0.2098476739 1.848896e+00  8.534618
## 72  0.0148216611 0.1956496069 1.639048e+00  8.071727
## 73  0.0157671023 0.1803552252 1.443399e+00  7.667914
## 74  0.0164135341 0.1642649070 1.263044e+00  7.323194
## 75  0.0166489549 0.1477336625 1.098779e+00  7.040829
```

```
## 76  0.0154058226 0.1317062738 9.510450e-01  6.821968
## 77  0.0133023720 0.1173521765 8.193387e-01  6.607391
## 78  0.0116325685 0.1048847063 7.019866e-01  6.341285
## 79  0.0103410450 0.0938978995 5.971019e-01  6.027166
## 80  0.0095781020 0.0839383260 5.032040e-01  5.671349
## 81  0.0095399022 0.0743793239 4.192656e-01  5.297151
## 82  0.0096182575 0.0648002441 3.448863e-01  4.954596
## 83  0.0093368256 0.0553227025 2.800861e-01  4.668793
## 84  0.0087084984 0.0463000405 2.247634e-01  4.437203
## 85  0.0077230257 0.0380842785 1.784633e-01  4.254618
## 86  0.0064744856 0.0309855229 1.403790e-01  4.101920
## 87  0.0053284400 0.0250840601 1.093935e-01  3.942353
## 88  0.0043594750 0.0202401026 8.430947e-02  3.760485
## 89  0.0035692335 0.0162757483 6.406936e-02  3.547512
## 90  0.0029633267 0.0130094682 4.779362e-02  3.298129
## 91  0.0028572884 0.0100991607 3.478415e-02  3.017413
## 92  0.0022814134 0.0075298097 2.468499e-02  2.847003
## 93  0.0017840127 0.0054970967 1.715518e-02  2.685068
## 94  0.0013631920 0.0039234943 1.165808e-02  2.531564
## 95  0.0010135087 0.0027351440 7.734586e-03  2.385820
## 96  0.0007333941 0.0018616926 4.999442e-03  2.243522
## 97  0.0005193568 0.0012353171 3.137750e-03  2.098835
## 98  0.0003600706 0.0007956034 1.902432e-03  1.949935
## 99  0.0002400869 0.0004955247 1.106829e-03  1.798061
## 100 0.0001545022 0.0002982301 6.113044e-04  1.628056
## 101 0.0002209790 0.0003130743 3.130743e-04  1.416760
##
## $lt_summary
##       e0       cv
## 38.10573  0.81675
```

Now we need to apply the `LTFun()` to every year in our data, i.e. to every column of our matrix `M`. We start by preparing an empty list with 264 entries – one entry for each year in our data set. This list is where we will store the output of the life-table function for each year.

```r
# number of years
k <- length(years)

lts <- vector(mode = 'list', length = k)
```

Next, we use a for-loop to iterate over all the different columns in our matrix `M`. For each column we apply the `LTFun()` and store its output in the corresponding entry of `lts`.

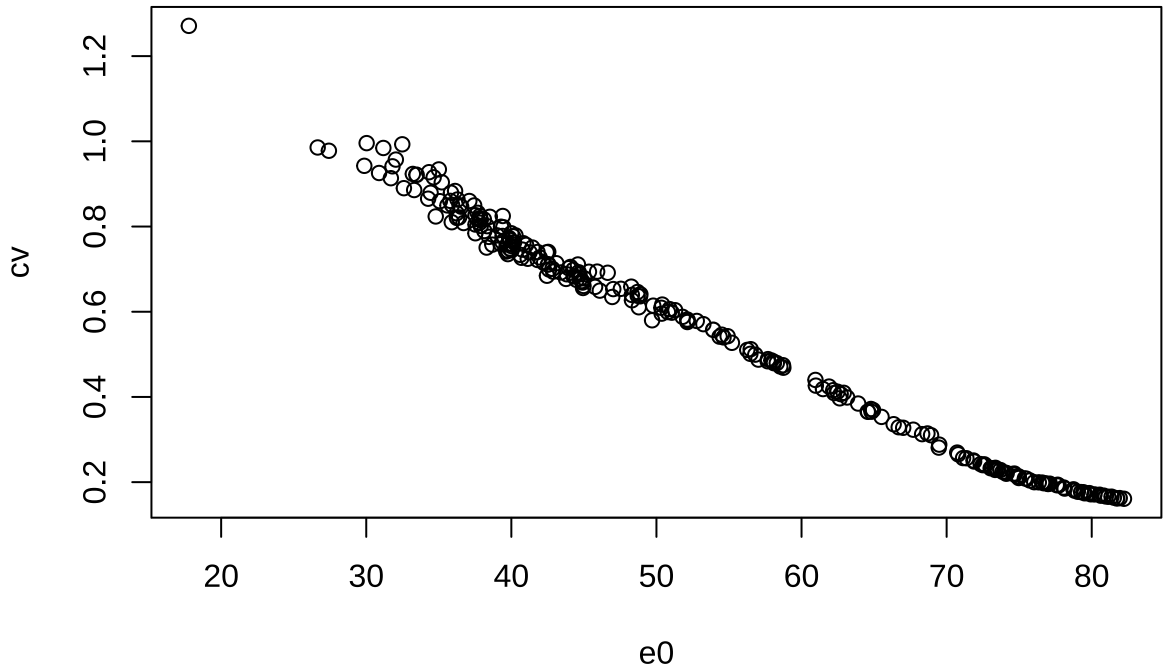
```r
for (i in 1:k) {
  lts[[i]] <- LTFun(x = 0:100, nmx = M[,i])
}
```

What we now have is a list 264 entries, each entry featuring a life-table and some life-table summary statistics. We can use the function `sapply()` to go over each entry of the list and extract the total life-expectancy and the CV. You could achieve the same by writing another for loop.

```r
e0 <- vector(mode = 'numeric', length = k)
for (i in 1:k) {
  e0[i] <- lts[[i]]$lt_summary['e0']
}
cv <- vector(mode = 'numeric', length = k)
```

```
for (i in 1:k) {
  cv[i] <- lts[[i]]$lt_summary['cv']
}
```

```
plot(e0, cv)
```



Literature

Preston, S. H., Heuveline, P., & Guillot, M. (2001). Demography. Oxford, UK: Blackwell.