Data structures

Jonas Schöley

September 13, 2017

Contents

Todays concepts

- data structures
- vector
- data frame
- matrix
- array
- list
- indexing
- by position
- by name
- vectorization

Todays operators

- `[]`
- `[[]]`
- `$`
- `:`

Todays functions

- `c()`
- `seq()`
- `length()`
- `cumprod()`
- `diff()`
- `names()`

Different data analysis problems call for different *data structures*. Like most programming languages R is very flexible in that regard and features numerous ways to represent data. A *data frame* is a table comparable to the tables you work with in Excel, STATA or SPSS. A vector comes in handy when you want to store n values and index them $1 \dots n$, i.e. it *always* comes in handy. On *matrices* you can do matrix algebra. *Arrays* are just matrices generalized to more than 2 dimensions. *Lists* are the most flexible data structure in R. You can use them to represent hierarchical data or to store many different things (plots, matrices, data frames) in a single object.

Will the contraceptices fail?

This excercise shows you how to work with vectors. In statistics and data analysis we rarely work with single numbers. Instead we work on collections of numbers (e.g. population size by age, average clutch size by bird etc.). Treating these collections of numbers as vectors is a convenient abstraction.

We create a vector of numbers using the function.

```
# Number of people using contraception
# at beginning of interval
Nx <- c(100, 80, 70, 60, 56)
# Number of people becoming pregnant
# during the interval
Dx <- c(5, 4, 5, 2)
```

There are other ways apart from `c()` to create a vector. Below we use the `seq()` function to create a sequence of ages 0 to 12 in intervals of 3.

```
# Age at beginning of interval
x <- seq(from = 0, to = 12, by = 3)
```

Assigning a single value to an object creates a vector of length 1, i.e. a *scalar*.

```
# Width of age interval
nx <- 3
```

We can divide each element of a vector by the corresponding element of a different vector of same length just by dividing the vectors (this is also true for addition, substraction and multiplication). Our `Nx` vector has one element more than our `Dx` vector. In order to make `Nx` the same length as `Dx` we remove the last element of `Nx`.

```
# Probability of getting pregnant
# during the interval [x, x+n)
qx <- Dx / Nx[-length(Nx)]
qx
```

```
## [1] 0.05000000 0.05000000 0.07142857 0.03333333
```

We can also do arithmetic with a vector and a scalar. Here we substract each element of the qx vector from the scalar 1.

```
# Probability of not getting pregnant
# during the interval [x, x+n)
px <- 1-qx
px
```

```
## [1] 0.9500000 0.9500000 0.9285714 0.9666667
```

The `cumprod()` function returns the cumulative product of its input vector. Its output is of the same length as its input.

```
# Probability not getting pregnant up until start of interval
lx <- cumprod(c(1, px))
lx
```

```
## [1] 1.0000000 0.9500000 0.9025000 0.8380357 0.8101012
```

The cumulative distribution function gives the probability of getting pregnant until x. It is the additive inverse of the survival function.

```
# Probability of getting pregnant until start of interval
Fx <- 1-lx
```

The last element of the `Fx` vector is the probability of getting pregnant during the first year of contraceptive use. We first count the number of elements in the `Fx` vector (`length(Fx)`) and use this number to index the last element of `Fx`.

```
# Probability of getting pregnant during
# first year of contraceptive use
Fx[length(Fx)]
```

```
## [1] 0.1898988
```

We estimate the probability of getting pregnant during the first year of contraceptive use as being 18.9 %. Demographers however would not be very happy about our methodology because it is based on conditional probabilities ($q(x), p(x)$) as opposed to *occurence-exposure* rates (i.e. mortality rates), the latter being thougt of as a better estimate for the risk of experiencing an event during some time interval. So let's do this excercise again, the demographers way, and compare results.

We have written the probability of getting pregnant in interval [x, x+n) as ${}_nD_x/N_x$, with N_x being the number of people who are not pregnant at the start of the interval. If we write ${}_nD_x/{}_nE_x$ and let ${}_nE_x$ be the person-years of exposure to risk of getting pregnant during interval $[x, x+n)$ we get the *pregancy rate*.

```
# Number of censorings during interval
Cx <- diff(-Nx)-Dx

# Person-months of exposure to risk during interval assuming constant
# risk of pregnancy and censoring during interval
Ex <- (diff(Nx)*nx) / log(Nx[-1]/Nx[-length(Nx)])

# Pregancy rate during interval
Mx <- Dx/Ex

# Probability of getting pregnant during interval
qx2 <- 1-exp(-nx*Mx)
# Probability of not getting pregnant during the interval
px2 <- 1-qx2

# Probability not getting pregnant up until start of interval
lx2 <- cumprod(c(1, px2))

# Probability of getting pregnant until start of interval
Fx2 <- 1-lx2
Fx2[5]
```

```
## [1] 0.1980991
```

```
# Putting it all in a table
data.frame(
  age = x[1:4],
  width = nx,
  Nx = Nx[1:4],
  Dx = Dx,
  qx, qx2, delta = qx-qx2
)
```

```
##   age width  Nx Dx         qx        qx2         delta
## 1   0     3 100  5 0.05000000 0.05425839 -0.0042583910
## 2   3     3  80  4 0.05000000 0.05201117 -0.0020111677
## 3   6     3  70  5 0.07142857 0.07417990 -0.0027513288
## 4   9     3  60  2 0.03333333 0.03390822 -0.0005748836
```

Calculating life-expectancy

```
swe <- read.table('swe_dxnx.txt', skip = 3, header = TRUE)
str(swe)
```

```
## 'data.frame':    648 obs. of  4 variables:
##  $ period   : Factor w/ 27 levels "1751-1759","1760-1769",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ age_group: Factor w/ 24 levels "0","100-104",..: 1 6 16 3 7 8 9 10 11 12 ...
##  $ deaths   : num  123647 72225 22043 10526 9210 ...
##  $ exposure : num  501259 1789971 1705802 1610573 1436730 ...
```

```
swe1751 <- swe[swe$period == '1751-1759',]
swe1751$x <- c(0, 1, 5, seq(10, 110, 5))
swe1751$nx <- c(diff(swe1751$x), Inf)
swe1751$nmx <- swe1751$deaths / swe1751$exposure
swe1751$npx <- exp(-swe1751$nx*swe1751$nmx)
swe1751$lx <- c(1, cumprod(swe1751$npx)[-nrow(swe1751)])
swe1751$ndx <- c(-diff(swe1751$lx), swe1751$lx[nrow(swe1751)])
swe1751$nLx <- -swe1751$ndx*swe1751$nx / log(swe1751$npx)
swe1751$nLx[is.nan(swe1751$nLx)] <- 0
swe1751$Tx <- rev(cumsum(rev(swe1751$nLx)))
swe1751$ex <- swe1751$Tx / swe1751$lx
```

The `within()` function allows you to perform operations "within" a data frame. Doing so you don't need to specify the data frame anymore if you want to select or add a column.

```
within(swe1751, {
  x <- c(0, 1, 5, seq(10, 110, 5))
  nx <- c(diff(x), Inf)
  nmx <- deaths / exposure
  npx <- exp(-nx*nmx)
  lx <- c(1, cumprod(npx)[-nrow(swe1751)])
  ndx <- c(-diff(lx), lx[nrow(swe1751)])
  nLx <- -ndx*nx / log(npx)
  nLx[is.nan(nLx)] <- 0
  Tx <- rev(cumsum(rev(nLx)))
  ex <- Tx / lx
})
```

```
##       period age_group    deaths   exposure   x  nx         nmx
## 1  1751-1759         0 123647.00  501258.74   0   1 0.246673006
## 2  1751-1759       1-4  72224.98 1789970.89   1   4 0.040349807
## 3  1751-1759       5-9  22042.99 1705802.06   5   5 0.012922361
## 4  1751-1759     10-14  10525.98 1610572.98  10   5 0.006535550
## 5  1751-1759     15-19   9209.98 1436729.96  15   5 0.006410377
## 6  1751-1759     20-24  11210.02 1402540.88  20   5 0.007992651
## 7  1751-1759     25-29  12460.98 1322120.43  25   5 0.009424996
## 8  1751-1759     30-34  14024.96 1216605.52  30   5 0.011527944
## 9  1751-1759     35-39  11651.98 1025218.10  35   5 0.011365367
```

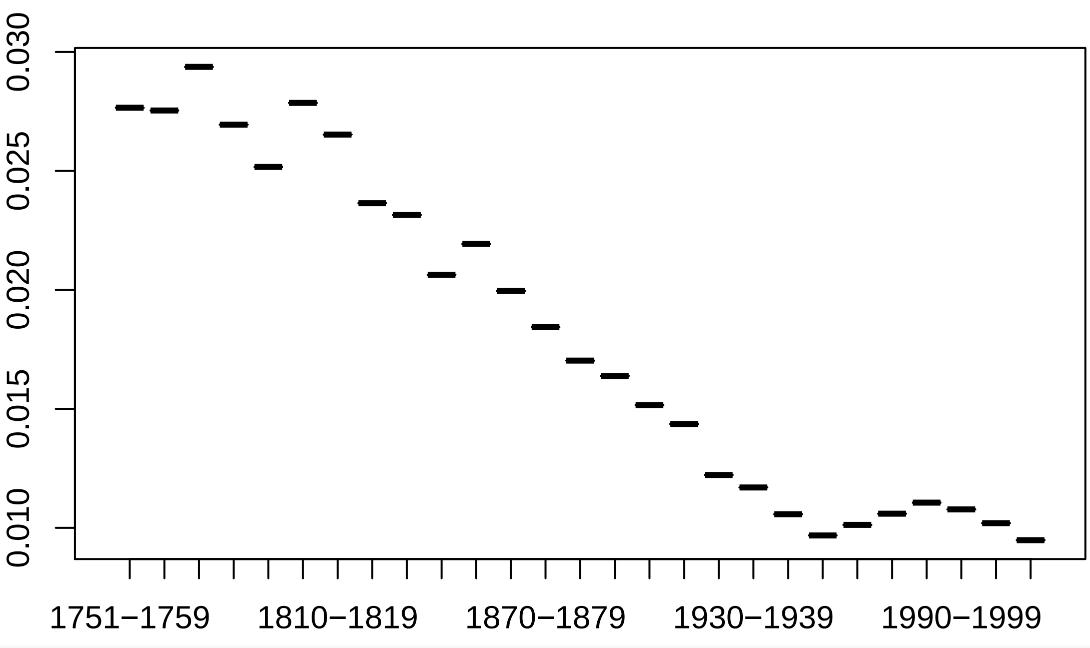
```
## 10 1751-1759     40-44  14577.11  913347.70  40   5 0.015960088
## 11 1751-1759     45-49  12765.00  763709.80  45   5 0.016714464
## 12 1751-1759     50-54  15478.98  713426.43  50   5 0.021696673
## 13 1751-1759     55-59  15535.92  570216.23  55   5 0.027245664
## 14 1751-1759     60-64  20790.98  543123.13  60   5 0.038280417
## 15 1751-1759     65-69  21439.05  428351.04  65   5 0.050050188
## 16 1751-1759     70-74  25111.99  300705.00  70   5 0.083510384
## 17 1751-1759     75-79  18772.99  155492.82  75   5 0.120732198
## 18 1751-1759     80-84  14352.04   94794.82  80   5 0.151401100
## 19 1751-1759     85-89   7371.95   36766.38  85   5 0.200507910
## 20 1751-1759     90-94   3213.03   11330.16  90   5 0.283582050
## 21 1751-1759     95-99    965.86    2491.16  95   5 0.387714960
## 22 1751-1759   100-104    201.37     350.61 100   5 0.574341861
## 23 1751-1759   105-109     13.70       9.69 105   5 1.413828689
## 24 1751-1759      110+      0.00       0.00 110 Inf         NaN
##             npx           lx          ndx          nLx           Tx
## 1  0.7813961639 1.000000e+00 2.186038e-01 8.862090e-01 3.607786e+01
## 2  0.8509522792 7.813962e-01 1.164653e-01 2.886391e+00 3.519165e+01
## 3  0.9374312989 6.649308e-01 4.160386e-02 3.219525e+00 3.230526e+01
## 4  0.9678503999 6.233270e-01 2.003971e-02 3.066263e+00 2.908574e+01
## 5  0.9684563347 6.032873e-01 1.902989e-02 2.968608e+00 2.601948e+01
## 6  0.9608247432 5.842574e-01 2.288843e-02 2.863685e+00 2.305087e+01
## 7  0.9539681623 5.613689e-01 2.584084e-02 2.741735e+00 2.018718e+01
## 8  0.9439899881 5.355281e-01 2.999494e-02 2.601933e+00 1.744545e+01
## 9  0.9447576540 5.055332e-01 2.792684e-02 2.457188e+00 1.484352e+01
## 10 0.9233005802 4.776063e-01 3.663213e-02 2.295233e+00 1.238633e+01
## 11 0.9198245623 4.409742e-01 3.535530e-02 2.115252e+00 1.009109e+01
## 12 0.8971938193 4.056189e-01 4.170013e-02 1.921960e+00 7.975843e+00
## 13 0.8726433664 3.639188e-01 4.634747e-02 1.701095e+00 6.053883e+00
## 14 0.8258004786 3.175713e-01 5.532077e-02 1.445145e+00 4.352788e+00
## 15 0.7786053761 2.622505e-01 5.806086e-02 1.160053e+00 2.907643e+00
## 16 0.6586572937 2.041897e-01 6.969866e-02 8.346107e-01 1.747590e+00
## 17 0.5468061143 1.344910e-01 6.095051e-02 5.048405e-01 9.129792e-01
## 18 0.4690689536 7.354051e-02 3.904494e-02 2.578907e-01 4.081386e-01
## 19 0.3669463788 3.449557e-02 2.183755e-02 1.089111e-01 1.502479e-01
## 20 0.2422196669 1.265802e-02 9.592002e-03 3.382443e-02 4.133674e-02
## 21 0.1439089026 3.066023e-03 2.624795e-03 6.769908e-03 7.512306e-03
## 22 0.0566020935 4.412279e-04 4.162535e-04 7.247487e-04 7.423981e-04
## 23 0.0008509617 2.497443e-05 2.495317e-05 1.764936e-05 1.764936e-05
## 24          NaN 2.125228e-08 2.125228e-08 0.000000e+00 0.000000e+00
##            ex
## 1  36.0778633
## 2  45.0368915
## 3  48.5843957
## 4  46.6620880
## 5  43.1294958
## 6  39.4532772
## 7  35.9606347
## 8  32.5761586
## 9  29.3621003
## 10 25.9341790
## 11 22.8836394
## 12 19.6633904
## 13 16.6352595
```

```
## 14 13.7064903
## 15 11.0872703
## 16  8.5586590
## 17  6.7884024
## 18  5.5498474
## 19  4.3555703
## 20  3.2656548
## 21  2.4501796
## 22  1.6825726
## 23  0.7066974
## 24  0.0000000
```

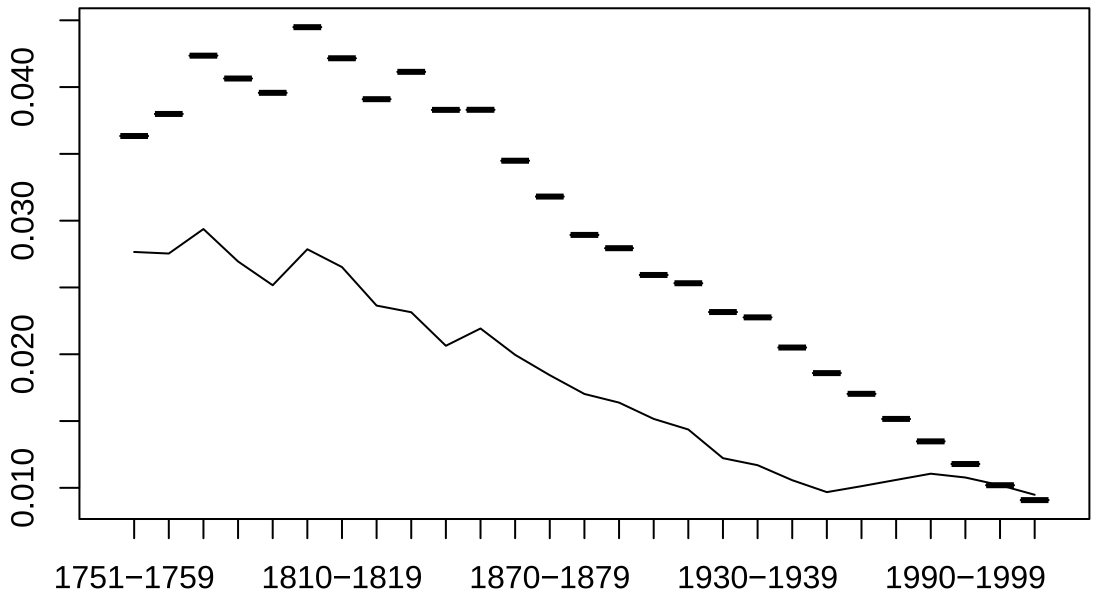
Age-standardization of death rates

```
periods <- unique(swe$period)
age_groups <- unique(swe$age_group)
D <- matrix(swe$deaths, nrow = length(age_groups), dimnames = list(age_groups, periods))
E <- matrix(swe$exposure, nrow = length(age_groups), dimnames = list(age_groups, periods))

M <- D/E
M[is.nan(M)] <- 0
CMRt <- colSums(D) / colSums(E)
plot(x = periods, y = CMRt)
```




```
pE <- prop.table(E, 2)
sM <- t(M)%*%pE
plot(x = periods, y = sM[,'2000-2009'])
lines(x = periods, y = diag(sM))
```



Lists

```r
library(demography)
```

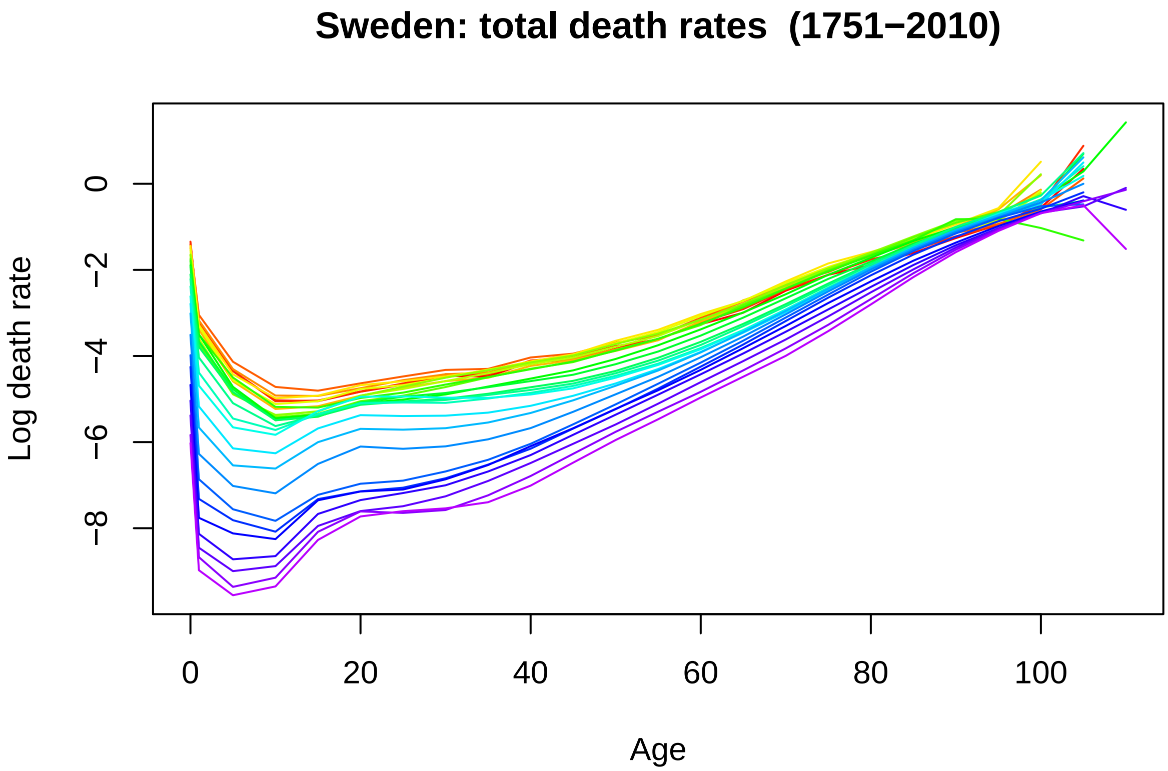
```
## Loading required package: forecast
```

```
## This is demography 1.20
```

```r
dd <-demogdata(M, pop = E,
               ages = c(0, 1, seq(5, 110, 5)),
               years = c(1751, seq(1760, 2010, 10)),
               type = 'mortality', label = 'Sweden', name = 'Total')
str(dd)
```

```
## List of 7
##  $ year  : num [1:27] 1751 1760 1770 1780 1790 ...
##  $ age   : num [1:24] 0 1 5 10 15 20 25 30 35 40 ...
##  $ rate  :List of 1
##   ..$ Total: num [1:24, 1:27] 0.24667 0.04035 0.01292 0.00654 0.00641 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:24] "0" "1" "5" "10" ...
##   .. .. ..$ : chr [1:27] "1751" "1760" "1770" "1780" ...
##  $ pop   :List of 1
##   ..$ Total: num [1:24, 1:27] 501259 1789971 1705802 1610573 1436730 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:24] "0" "1" "5" "10" ...
##   .. .. ..$ : chr [1:27] "1751" "1760" "1770" "1780" ...
##  $ type  : chr "mortality"
##  $ label : chr "Sweden"
##  $ lambda: num 0
##  - attr(*, "class")= chr "demogdata"
```

```r
plot(dd)
```


Sweden: total death rates (1751−2010)
Log death rate
0
−2
−4
−6
−8
0
20
40
60
80
100
Age