6

Analysis of data on origin-destination migration dynamics with R

Abstract. *This chapter presents a collection of R functions that can be used to explore and analyze data on origin-destination migration flows. We aim to introduce the reader to simple routines developed in R that help to understand the complexities of migration data. The presented functions allow us to simulate origin-destination relocation trajectories and derive aggregate measures that are directly related to dynamics of international migration. The measures include number of migrations, which can be viewed as conditional relocations, number of transitions over a one-year period, population size and person-years lived during a one-year period. Some selected results of the functions are plotted to illustrate some useful graphics functionalities that are available in R.*

6.1. Introduction

The aim of this chapter is to illustrate how R software can be used to facilitate the exploration and understanding of data on origin-destination migration flows. R is an open-source programming language and software environment for statistical computing and graphics (R Development Core Team, 2009). The presented routines, therefore, can be immediately implemented by anyone interested. The software and contributed extension packages are available from the Comprehensive R Archive Network (CRAN) repository, which can be found at http://cran.r-project.org/. The main R Web site, http://www.r-project.org, provides all essential information about R. It lists the on-line manuals, related books and other materials.

We present a collection of R functions. One function simulates individual origin-destination relocation histories, where relocation is a change of a country of residence. The other functions produce aggregate measures related to origin-destination population dynamics. The basic measure is the number of migrations. Compared to defining relocation, defining migration includes some additional conditions referring generally to duration of residence. A description of various migration definitions and their ambiguities can be found in Chapters 3 and 4 of this book.

All measures are compiled for the whole virtual population. Such aggregate data are usually available in practice. They should therefore be treated as an accessible basis for estimating the parameters of the underlying relocation processes. If estimates use the simulated numbers, the results should approximate the parameters of the process that was initially used to create relocation sample paths. The knowledge of the underlying processes provides a basis for deriving the requested migration measures.

A function for simulating relocation trajectories is described in Section 6.2. Functions for producing aggregate measures are the subject of Section 6.3. Section 6.4 includes a graphic presentation of some selected results of the functions. The last section concludes the chapter. A complete code for the functions and the example, together with some explanations, is provided in the Appendix.

6.2. Simulation

The aim of the simulation is to generate continuous-time relocation histories of individuals who may repeatedly relocate among a finite number of countries. The relocation paths are generated using the function `simOD()`. The simulation is carried out by determining the next transition times and the next states visited for all individuals in the initial population. The population numbers in each state at time zero are given in an argument vector `Ni`. An ending time of the observation is determined by `tmax`. For a given individual, the simulation terminates when the time of relocation occurrence is greater than `tmax`.

The argument `tdist` specifies interarrival time distribution. The default exponential distribution corresponds to `tdist = "exp"`. Another possible choice is to assume that survival time follows a Weibull distribution and set `tdist = "weibull"`. If an exponential model is assumed, the origin-destination relocation intensities are specified in matrix `M`. This matrix is organized so that rows correspond to origin countries and columns correspond to destination countries. The diagonal of the matrix is ignored. The dimensions of matrix `M` have to correspond to the number of considered countries. In the case of the Weibull distribution, a scale parameter is equal to an inverse of an appropriate transition intensity given in matrix `M`. A common shape parameter `shapeWeibull` has to be additionally specified.

An optional argument `returnMulti` is a scalar specifying relatively higher intensity of return migration to the state of residence at the beginning of observation. For ease of exposition, we refer to this state as the country of birth. The relative impact of `returnMulti` can be of two types, indicated by a `returnType` argument. If `returnType = 1`, the intensity for a particular origin-destination flow is relatively higher for people born in a destination state, which results in the higher total intensity of leaving the current state regardless of direction. If `returnType = 2`, the intensity of relocation to a state of birth is higher than to other possible destinations with no impact on the total intensity of leaving the current state. The intensities of relocating in other directions are relatively decreased.

A further argument, `occur`, may be used to specify occurrence dependence. A function describing occurrence dependence has to be predefined and its name must be given here.

The `simOD()` function also allows one to control for some heterogeneity among population members. The optional argument `X` is a data frame with time-constant covariates and `B` is a vector of corresponding coefficients. The observed covariates given in data frame `X` change the intensity rates proportionally by exp(`BX`). The additional argument `zdist` specifies a distribution of unobserved heterogeneity from which random values are drawn. The possible choices are `zdist = "gamma"` or `zdist = "unif"`. In the case of a uniform distribution, exponentially transformed values are applied. Depending on the distribution additional parameters have to be specified. For `"gamma"` they include `shapeGamma` and `scaleGamma`. For `"unif"` they include `minUnif` and `maxUnif`.

In the simplest default case, all individuals considered are characterized by the same constant origin-destination specific intensities of relocations. Thus, an exponential interarrival time distribution is assumed. The `simOD()` function then only requires a matrix `M` of the origin-destination transition intensities, a vector `Ni` with starting population stocks in each state and an ending time of the observation, `tmax`. For three thousand individuals distributed evenly among three countries, an example call to the `simOD()` function would be of the form

```
> sim <- simOD(M = matrix(c(NA,  0.1,0.05,
                            0.2,  NA, 0.2,
                            0.1, 0.3,  NA), nrow = 3, byrow = T),
               Ni = c(1000,1000,1000), tmax = 20)
```

We assign the result to an object called `sim` in order to save it for further analysis. Similar assignments are carried out for other functions throughout the text and no adjustments are needed before the presented code can be run. The result of the `simOD()` function is a data frame that has the following structure

```
> sim
     ID   time state
  1   1   0.00     1
  2   1  20.25     1
  3   2   0.00     1
  4   2   1.79     3
  5   2   2.92     2
  6   2   4.02     3
  7   2   6.05     3
  ...
```

For each individual distinguished by a unique identifier `ID`, it gives times of relocations (`time`) and countries of residence following relocations (`state`). The value of the `state` variable corresponding to `time` equal to zero refers to a country of residence at the beginning of the observation period. A country of residence at the end of the observation period is indicated by `state` at `time` greater than `tmax`. For non-migrants it is the same as a country of residence at the onset of observation and for migrants it is a destination country of the last observed relocation. If we control for heterogeneity across individuals, the observed values of covariates or random values of unobserved heterogeneity are added to the final output of the function `simOD()` as additional columns in a data frame.

6.3. Migration measures

Aggregate measures are derived for the whole virtual population that is combined from the individual relocation histories generated by the `simOD()` function. We present functions that produce statistics that are directly related to the dynamics of international migration. The annual number of migrations (conditional relocations) is obtained using a `migrations()` function. It also returns individual trajectories of countries of *de jure* residence, which are utilized in other functions. A `transitions()` function produces the number of transitions over a one-year period, that is, the number of migrants that are usually estimated from a comparison of current country of residence and country of residence one year ago. A `population()` function returns population size and person-years lived during a year. All these functions use `data` argument, an output of the function `simOD()`. A second argument, `tdef`, is a vector of duration criteria expressed in years. These criteria refer to length of residence and they are used to distinguish migrations from relocations. Before we move on to a presentation of functions it is necessary to clarify the migration definition that is used in the example.

In order to ensure consistent and complete trajectories of unique countries of residence and changes thereto, a migration is defined as a relocation that is followed by a continuous stay of a specified duration in a destination country, provided a person is a non-

resident thereof. Thus, if an individual leaves his or her country of birth and then relocates a few times to different foreign countries, but does not reside long enough in any of them to become a migrant and thereby a member of the population, he or she remains an official (*de jure*) resident of the country of birth. This prevents the occurrence of a situation in which a person does not belong to any population. Note that a person has one unique place of residence only when all countries use the same minimum-duration threshold. The origin of migration is a previous country of residence and not a previous country of a short stay that does not constitute a residence.

The `migrations()` function produces residence histories of individuals and calculates the aggregate number of migrations as defined above. The result is a list of two components, `ODmx` and `ODdata`. `ODmx` is an array including annual origin-destination migration matrices for all years until `tmax` for all duration thresholds given in `tdef`. For simplicity, all countries use the same duration criterion. `ODdata` is a data frame that includes, inter alia, information on country of residence (`ctr`). If we only need the migration matrices we can use the following command

```
> M0 <- migrations(data = sim, tdef = seq(0,5,.05))$ODmx
> M0
```

In order to decrease the execution time, the number of considered duration criteria can be limited to a few values, for instance, `tdef = c(0,.25,.5,1)`.

The definition of migration presented above leads to a straightforward corresponding measure of transitions. The transition is reflected in a different country of residence at the beginning and the end of the year. No additional duration criteria have to be applied. The `transitions()` function utilizes the `migrations()` function in order to obtain residence histories. This information is used to determine country of residence at the beginning and end of each year. Whereas the migration measure returned by `migrations()` represents an event approach to counting migration, the migration measure returned by `transitions()` represents a status approach. The output of the `transitions()` function has the same structure as the `ODdata` array from the output of the `migrations()` function. An example call to the `transitions()` function would be

```
> T0 <- transitions(data = sim, tdef = seq(0,5,.05))
> T0
```

The population size and person-years lived in the countries are obtained using the `population()` function, which could be called as follows

```
> P0 <- population(data = sim, tdef = seq(0,5,.05))
> P0
```

The output contains a list of three arrays with figures for each country (in columns) and each year (in rows). `POP` includes population numbers at the beginning of each year, `POPavg` gives an average annual population size and `PY` refers to the values of person-

years. Note that different concepts of the population may be applied. Population size may refer to the total number of the residents irrespective of their current place of stay (*de jure* population). We can also count people present in the country irrespective of their countries of residence (*de facto* population). Both population numbers are returned by the `population()` function. They are denoted respectively by `ctr` and `cts` in the third dimension of output arrays `POP` and `POPavg`. In addition, `population()` gives the number of people who are present in their country of residence (`ctrs`). The person-years given in the array `PY` refers to the same population types as given in arrays `POP` and `POPavg`.

6.4. Plotting results

Once the relocation histories have been simulated and the migration measures have been compiled, any data of interest can be easily extracted for further analysis. The comprehensive graphics functionalities that are available in R play an indispensible facilitative role in the data exploration. Two illustrative examples are given below. In the first one, we compare the number of origin-destination specific migrations during a year with the number of corresponding transitions during a year. The figures are compared for the various lengths of duration criterion used to determine a country of residence. This duration threshold is the same for both types of measures of migration considered. In the second example, we look at person-years of residence in a given country and person-years of actual stay in that country of residence.

For the three countries considered in the simulation there are six origin-destination specific migration flows. We do not consider flows within the countries. We aim to look at migration and transition counts for each of them. The Trellis graphics system with multipanel conditioning is particularly useful in such a case. It is provided in R in an add-on package called `lattice`. Before we can call a trellis function that is appropriate for plotting migrations and transitions against duration criterion as used in the definition and that is conditional on direction of flow, the data available in arrays `M0` and `T0` have to be arranged in one data frame. This can be easily done as follows

```
> M1 <- as.data.frame(as.table(M0)); M1$type <- "migrations"
> T1 <- as.data.frame(as.table(T0)); T1$type <- "transitions"
> MT <- rbind(M1, T1)
> head(MT)
   from to year def Freq       type
 1    1  1    1   0    0 migrations
 2    2  1    1   0  200 migrations
 3    3  1    1   0  104 migrations
 4    1  2    1   0  111 migrations
 5    2  2    1   0    0 migrations
 6    3  2    1   0  274 migrations
```

An automatically created variable, `Freq`, is of interest to us. The added variable `type` indicates the type of measure. The `OD` variable denoting the direction of migration, which will be used as a conditional one for plotting, can be created from information included in columns `from` and `to`

```
> MT$OD <- factor(paste("From", MT$from, "to", MT$to, sep = " "))
```

A selection of data to be shown in a graph is a final stage of data preparation. Here we chose the data for the six international origin-destination flows in the tenth `year`, for the various duration criterion `def` that is used in the definition up to 12 months

```
> MT$def <- as.numeric(as.character(MT$def))
> MT     <- MT[MT$from != MT$to & MT$year == 10 & MT$def <= 12,]
```

The `MT` data frame can now be used in the desired trellis function

```
> trellis.device(color = FALSE)
> xyplot(Freq ~ def | OD, groups = type, data = MT, type = "l",
         as.table = T, xlab = "Duration criterion [months]",
         ylab = "Counts", auto.key = list(space = "bottom",
         points = FALSE, lines = TRUE))
```

The resulting plot is shown in Figure 6.1. Note that the command preceding the main call of function `xyplot()` only changes the display colours to black and white. By default the trellis plots are printed in colour.

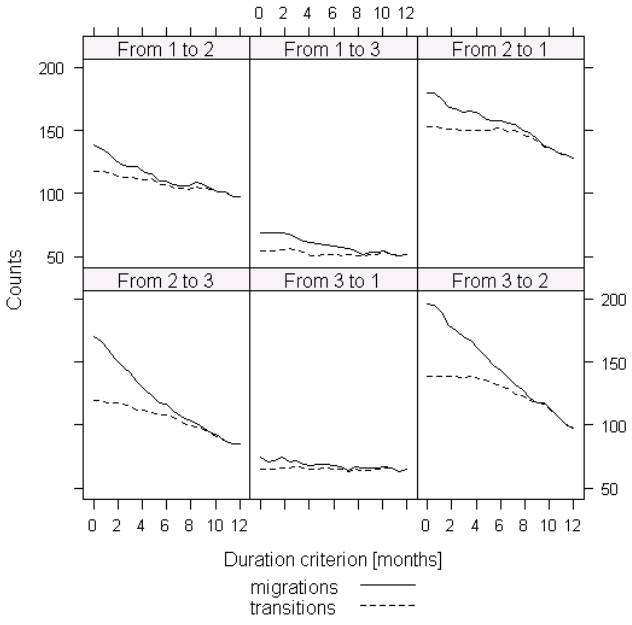

Figure 6.1 Number of origin-destination specific migrations (solid line) and transitions (dashed line) for various durations up to one year.

In the second example dealing with person-years for different population concepts, a subset of data can be obtained directly from the `P0` array. The selection criteria are assigned to the following variables

```
> year  <- 10
> state <- 2
> duration <- as.character(seq(0,5,.5))
```

For the chosen year, country (state) and duration thresholds of the person-years of residence `ctr` can be presented in one bar plot together with the person-years actually spent in this country `ctrs`

```
> barplot(P0$PY[year, state, c("ctr","ctrs"), duration],
          beside = TRUE, space = c(-0.8,.4), las = 1,
          xlab = "Duration criterion [years]", ylab = "Person-years")
```

Figure 6.2 shows the obtained bar plot.

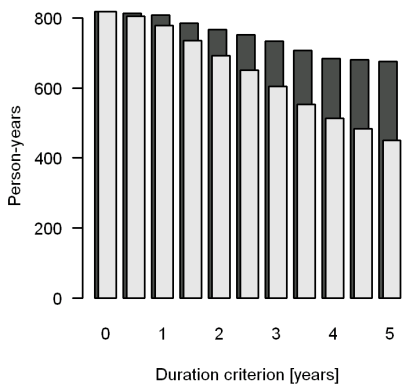

Figure 6.2 Person-years of residence (black bars) and person-years of actual stay in country of residence (grey bars) for various duration criteria in migration definition; country = 2, year = 10

The differences between the two data series indicate person-years spent by the individuals in the countries other than their place of residence.

6.5. Conclusions

This chapter has demonstrated a computer implementation of migration data analysis. The code was developed in the R environment, which has all the statistical, mathematical and graphical capabilities needed for exploring the data. The open-source nature of R offers obvious advantages. First, anyone interested can reproduce the analysis simply by re-executing the scripts. Second, the code can be easily modified to serve one's needs better.

Writing and documenting collections of R functions is, therefore, a very useful and effective way of organizing and communicating about the work.

References

R Development Core Team. 2009. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing. Available at http://www.R-project.org.

Appendix

This appendix includes a complete R code for the functions `simOD()`, `migrations()`, `transitions()` and `population()` described in this chapter and also an example of their application. This example is a collection of all the commands presented in the text. Note that the `migrations()`, `transitions()` and `population()` functions use a `na.locf()` function for replacing each `NA` with the most recent non-`NA` prior to it. It comes from a non-standard package, `zoo`, which has to be installed

```
> install.packages("zoo")
```

and then uploaded

```
> library(zoo).
```

```
#simOD(): simulation of origin-destination relocation trajectories
#---
simOD <- function(M, Ni, tmax, tdist = "exp", shapeWeibull = 0.5,
                  returnMulti = 1, returnType = 1, X = NULL, B = NULL,
                  zdist = NULL, shapeGamma = 2, scaleGamma = 1/2,
                  minUnif = -1, maxUnif = 1, t0 = 0, occur = NULL){

  diag(M) <- NA
  N       <- sum(Ni)
  cc      <- 1:ncol(M)
  ctb     <- rep(cc, Ni)
  corDir  <- 1/(returnMulti + ncol(M) - 2)

#covariates (observed heterogeneity)
  if (is.vector(X)) X <- matrix(X, ncol = 1, nrow = N)
  if (is.null(X) | is.null(B)){
    BX <- rep(0, N)
  } else{
    BX <- X %*% B
  }

#unobserved heterogeneity
  if (is.null(zdist)){
    z <- rep(1, N)
  } else if (zdist == "gamma") {
    z <- rgamma(N, shape = shapeGamma, scale = scaleGamma)
  } else if (zdist == "unif") {
    z <- exp(runif(N, min = minUnif, max = maxUnif))
  }

#initial values
  times   <- vector("list", N); times[]  <- t0
  states  <- vector("list", N); states[] <- ctb
```

```
###individual trajectories
   for (n in 1:N){
   #impact of observed and unobserved heterogeneity on intensity
     M.n <- M*z[n]*exp(BX[n])
   #increased intensity of return migration
     if (returnType == 1){
       M.n[,ctb[n]]  <- M.n[,ctb[n]] * returnMulti
     } else if (returnType == 2){
       M.n[-ctb[n],] <- rep(corDir*rowSums(M.n[-ctb[n],], na.rm = T),
                            ncol(M.n))
       M.n[-ctb[n],ctb[n]] <- returnMulti*M.n[-ctb[n],ctb[n]]
       diag(M.n) <- NA
     }
##timings of relocations and their directions
     i     <- 0
     t.cum <- 0
     while (t.cum <= tmax) {
       i <- i+1
     #possible destinations
       Sorig <- states[[n]][i]
       Sdest <- cc[-Sorig]
     #occurrence dependence
       oc <- ifelse(is.null(occur), 1, eval(parse(text = occur))(i-1))
       Rs <- oc*M.n[Sorig,Sdest]; k <- M[Sorig,Sdest]/Rs
     #waiting times to all possible destinations
       if(tdist == "exp"){
         Ts <- rexp(length(cc)-1, rate = Rs)
       } else if (tdist == "weibull"){
         Ts <- rweibull(length(cc)-1, shape = shapeWeibull,
                        scale = 1/(Rs*k^(1/shapeWeibull-1)))
       }
     #minimum waiting time and a corresponding direction
       t.cum <- times[[n]][i] + min(Ts)
       times [[n]][i+1] <- t.cum
       states[[n]][i+1] <- ifelse(t.cum <= tmax,
                                  Sdest[which(Ts == min(Ts))],
                                  states[[n]][i])
     }
##
###
   }

#output preparation
   ns <- unlist(lapply(states, length))
   x  <- unlist(times)
   y  <- unlist(states)
   msmData <- cbind.data.frame(ID = rep(1:N, ns), time = x, state = y)
   if (!is.null(zdist)) msmData$z    <- rep(z, ns)
   if (!is.null(X)){
     msmCov  <- matrix(ncol = ncol(X), nrow = sum(ns))
     dimnames(msmCov)[[2]] <- as.list(paste("X", 1:ncol(X), sep = ""))
     for (i in 1:ncol(X)) msmCov[,i] <- rep(X[,i], ns)
     msmData <- cbind.data.frame(msmData, msmCov)
   }
   return(msmData)
}
#---
#end of simOD()
```

```
#migrations(): number of migrations for different duration
#---
migrations  <- function(data, tdef){

   cc   <- as.numeric(levels(factor(data$state)))
   ccNo <- length(cc)
   tmax <- floor(min(tapply(data$time, data$ID, max)))
   names(data)[3] <- "cts"
   OD.yy.def           <- array(dim = c(ccNo, ccNo, tmax, length(tdef)))
   dimnames(OD.yy.def) <- list(from = cc, to = cc, year = 1:tmax,
                               def = tdef*12)
   labvar              <- c("ID","time","ctb","cts","ctr","mctr")
   ODdata              <- array(dim = c(nrow(data), length(labvar),
                                length(tdef)))

#country of birth
   data$ctb            <- rep(data$cts[data$time == 0],table(data$ID))
#year of relocation
   data$year           <- cut(data$time, c(0, 1:tmax), labels = FALSE)
#time till next (dnxt) and since previous (dprv) relocation
   data$dnxt           <- c(diff(data$time), NA)
   data$dprv           <- c(NA, diff(data$time))
   data[which(data$time == 0 | data$time>=tmax), c("dnxt","dprv")] <- NA
#previous country of stay (pcts)
   data$pcts           <- c(NA, data$cts[-length(data$cts)])
   data$pcts[which(data$time == 0)] <- data$cts[data$time == 0]
#first relocation
   migs <- !is.na(data$dnxt)
   data$fst[migs]      <- unlist(tapply(1:nrow(data[migs,]),
                                data$ID[migs], seq_along))
   data$dprv[data$pcts == data$ctb & data$fst == 1] <- 99

##migration counts for various duration criteria in definition
   for (t in 1:length(tdef)){
   #relocations followed by a stay satisfying duration condition
     data$def.nxt      <- cut(data$dnxt, c(tdef[t], Inf), labels = FALSE)
     durres            <- which(data$def.nxt == 1)
     fstobs            <- which(data$time == 0)
   #countries of residence (ctr)
     data$ctr <- NA
     data$ctr[durres]  <- data$cts[durres]
     data$ctr[fstobs]  <- data$ctb[fstobs]
     data$ctr          <- na.locf(data$ctr)
   #previous countries of residence (pctr)
     data$pctr         <- c(NA, data$ctr[-nrow(data)])
     data$pctr[fstobs] <- data$ctr[fstobs]
   #migration as a change of country of residence (mctr)
     data$mctr         <- as.numeric(!is.na(data$pctr) &
                                     data$pctr != data$ctr)
     data$mctr[data$mctr == 0] <- NA
   #origin-destination migration matrices
     OD.yy.def[,,,t]   <- with(data, table(from = factor(pctr,cc),
                               to = factor(ctr, cc),
                               factor(year, levels = 1:tmax), mctr))
   #output data frame with crucial variables
     ODdata[,,t]       <- as.matrix(data[,labvar])
   }
##
```

```
  dimnames(ODdata) <- list(1:nrow(ODdata), labvar, def = tdef*12)
  return(list(ODmx = OD.yy.def, ODdata = ODdata))
}
#---
#end of migrations()


#transitions(): number of transitions over one year period
#---
transitions <- function(data, tdef){

#country of residence in continuous time
   Ctr      <- migrations(data = data, tdef = tdef)$ODdata
   cc       <- as.numeric(levels(factor(data$state)))
   ccNo     <- length(cc)
   tmax     <- floor(min(tapply(data$time, data$ID, max)))
   cuts     <- 1:tmax
   names(data)[3] <- "cts"
#country of birth
   data$ctb <- rep(data$cts[data$time == 0], table(data$ID))
   IDs      <- unique(data$ID)
#discrete time points (tPts)
   ID       <- rep(IDs, each = length(cuts))
   time     <- rep(cuts, length(IDs))
   tPts     <- data.frame(ID, time)

   mTrans           <- array(dim = c(ccNo, ccNo, tmax, length(tdef)))
   dimnames(mTrans) <- list(from = cc, to = cc, year = 1:tmax,
                            def = tdef*12)

##transition counts for various duration criteria in definition
   for (t in 1:length(tdef)){
     mig <- cbind(data, Ctr[,c("ctr","mctr"),t])
     mig <- mig[mig$time <= tmax,]
   #country of residence at discrete time points obtained from
   #information on residence at continuous time points
     mD  <- merge(mig, tPts, by = c("ID","time"), all = T)
     mD[,c("cts","ctb","ctr")] <- na.locf(mD[,c("cts","ctb","ctr")],
                                          na.rm = F)
   #country of residence at discrete time points (mD.yy)
     tpts   <- mD$time %% 1 == 0
     mD.yy  <- mD[tpts,]
   #country of residence one year ago
     mD.yy$pctr                  <- c(NA, mD.yy$ctr[-nrow(mD.yy)])
     mD.yy$pctr[mD.yy$time == 0] <- NA
   #transition - change of country of residence at discrete time points
     mD.yy$rtrans                <- as.numeric(mD.yy$pctr != mD.yy$ctr)
     mD.yy$rtrans[mD.yy$rtrans == 0] <- NA
   #origin-destination transition matrices
     mTrans[,,,t] <- with(mD.yy, table(factor(pctr, levels = cc),
                                       factor(ctr, levels = cc),
                                       factor(time, levels = 1:tmax),
                                       rtrans))
   }
##
   return(ODtrans = mTrans)
}
#---
#end of transitions()
```

```
#population(): population size and person-years
#---
population <- function(data, tdef){

#country of residence in continuous time
  Ctr       <- migrations(data = data, tdef = tdef)$ODdata

  cc        <- levels(factor(data$state))
  ccNo      <- length(cc)
  tmax      <- floor(min(tapply(data$time, data$ID, max)))
  cuts      <- 1:tmax
  names(data)[3] <- "cts"
  data$ctb  <- rep(data$cts[data$time == 0], table(data$ID))
  IDs       <- unique(data$ID)

  POP           <- array(dim = c(tmax+1, ccNo, 3, length(tdef)))
  dimnames(POP) <- list(0:tmax, cc, country = c("ctr","cts","ctrs"),
                        def = tdef)
  PY            <- array(dim = c(tmax, ccNo, 3, length(tdef)))
  dimnames(PY)  <- list(1:tmax, cc, country = c("ctr","cts","ctrs"),
                        def = tdef)

#discrete time points
  ID        <- rep(IDs, each = length(cuts))
  time      <- rep(cuts, length(IDs))
  tPts      <- data.frame(ID, time)

  for (t in 1:length(tdef)){
    mig     <- cbind(data, Ctr[,c("ctr","mctr"), t])
    mig     <- mig[mig$time <= tmax,]
  #time between relocations and discrete time points
    mD      <- merge(mig, tPts, by = c("ID","time"), all = T)
    mD$year <- cut(mD$time, c(0,cuts), labels = FALSE, right = F)
    mD$tdif[mD$time != tmax]    <- unlist(tapply(mD$time, mD$ID, diff))
    mD[,c("cts","ctb","ctr")]   <- na.locf(mD[,c("cts","ctb","ctr")],
                                           na.rm = F)
  #presence in country of residence (ctrs)
    mD$ctrs[mD$ctr == mD$cts]   <- mD$ctr[mD$ctr == mD$cts]

  #person-years
    PY[,,"ctr",t]  <- with(mD, tapply(tdif,
                           list(year, factor(ctr, cc)), sum))
    PY[,,"cts",t]  <- with(mD, tapply(tdif,
                           list(year, factor(cts, cc)), sum))
    PY[,,"ctrs",t] <- with(mD, tapply(tdif,
                           list(year, factor(ctrs,cc)), sum))
    PY[is.na(PY)]  <- 0

  #population numbers at discrete time points
    tpts   <- mD$time %% 1 == 0
    mD.yy  <- mD[tpts,]
    POP[,,"ctr",t]  <- with(mD.yy, table(time, state = factor(ctr,cc)))
    POP[,,"cts",t]  <- with(mD.yy, table(time, state = factor(cts,cc)))
    POP[,,"ctrs",t] <- with(mD.yy, table(time, state = factor(ctrs,cc)))
    POP[is.na(POP)] <- 0
  }
```

```
#average population number
   POPavg  <- (POP[-1,,,] + POP[-nrow(POP),,,]) / 2
   return(list(POP = POP, POPavg = POPavg, PY = PY))
}
#---
#end of population()


#The R code for the example given in the text:
#---
#simulation
   sim <- simOD(M = matrix(c(NA,0.1,0.05,0.2,NA,0.2,0.1,0.3,NA),
                nrow = 3, byrow = T), Ni = c(1000,1000,1000), tmax = 20)
#measures
   M0  <- migrations (data = sim, tdef = seq(0,5,.05))$ODmx
   T0  <- transitions(data = sim, tdef = seq(0,5,.05))
   P0  <- population (data = sim, tdef = seq(0,5,.05))

#a trellis plot: migrations and transitions
#rearrangement of data into one data frame
   M1     <- as.data.frame(as.table(M0)); M1$type <- "migrations"
   T1     <- as.data.frame(as.table(T0)); T1$type <- "transitions"
   MT     <- rbind(M1, T1)
   MT$OD  <- factor(paste("From", MT$from, "to", MT$to, sep = " "))
   MT$def <- as.numeric(as.character(MT$def))
   MT     <- MT[MT$from != MT$to & MT$year == 10 & MT$def <= 12,]

   library(lattice)
   trellis.device(color = FALSE)
   xyplot(Freq ~ def | OD, groups = type, data = MT, type = "l",
          as.table = T, xlab = "Duration criterion [months]",
          ylab = "Counts", auto.key = list(space = "bottom",
          points = FALSE, lines = TRUE))

#a bar plot: person-years
   year     <- 10
   state    <- 2
   duration <- as.character(seq(0,5,.5))
   barplot(P0$PY[year, state, c("ctr","ctrs"), duration],
           beside = TRUE, space = c(-0.8,.4), las = 1,
           xlab = "Duration criterion [years]", ylab = "Person-years")
#---
#end of the example
```