# Reconstructing the Female Population of Burkina Faso, 1960–2005

Mark C. Wheldon

May 18, 2018

## Contents

## List of Figures

# 1 Introduction

**popReconstruct** is a method for reconstructing populations of the recent past. It simultaneously estimates age-specific population counts, fertility rates, mortality rates and net international migration flows from fragmentary data, and incorporates measurement error. Population dynamics over the period of reconstruction are modeled by embedding formal demographic accounting relationships in a Bayesian hierarchical model. Informative priors are required for vital rates, migration rates, population counts at baseline, and their respective measurement error variances. Inference is based on the joint posterior probability distribution which yields fully probabilistic interval estimates. A sample from this distribution is drawn using a Markov chain Monte Carlo algorithm.

The **popReconstruct** package has one main function for doing the reconstruction, `popRecon.sampler()`. See the help file for a complete list of its arguments.

This vignette demonstrates the main features of the **popReconstruct** package by reconstructing the female population of Burkina Faso from 1960–2005, as described in Wheldon et al. (2011). Consult this reference for full details of the method and descriptions of data sources. Suggestions for summarizing and plotting results are also given by way of example.

# 2 Notation

We use the symbols $n$, $s$, $g$ and $f$ to denote population counts, survival (a measure of mortality), net international migration (immigrants minus emigrants) and fertility, respectively. All of these parameters will be indexed by five-year increments of age, denoted by $a$, and time, denoted by $t$. For example, $f_{a,t}$ is the average annual age-specific fertility rate for women in the age range $[a, a+5)$ over the period $[t, t+5)$. Reconstruction will be done over the time interval $[1960, 2005)$. The age scale runs from 0 to 85 for survival and 0 to 80 for all other parameters.

# 3 Doing the Reconstruction

## 3.1 Initial Estimates and Census Counts

Bias-reduced initial estimates of age-specific fertility rates, survival and migration proportions, population counts in the baseline year and census counts in subsequent years are required. Ideally, initial estimates will be based on data which have been pre-processed to reduce systematic biases. For example, population counts based on censuses should be adjusted to reduce bias due to undercount of certain age groups and age heaping. Similarly, fertility rate data based on surveys should be adjusted to reduce bias due to omission and misplacement of births. See Wheldon et al. (2011) for further details.

Initial estimates and census counts for this vignette can be loaded by issuing the command `data(burkina_faso_females)`. This loads the object `burkina.faso.females`, a list with the following components: `fertility.rates`, `survival.proportions`, `migration.proportions`, `baseline.pop.counts` and `census.pop.counts`. Each of these is a matrix with one row per age-group and one column per time period.

The row and column names of the initial estimate matrices are important. They must indicate the start points of the age-groups and time-periods to which the corresponding matrix elements refer. The width of the age-groups and time-periods must be the same and `popRecon.sampler()` uses the row and column names to check this.

**Fertility Rates** These should be average annual age-specific fertility rates, $f_{a,t}$, over five-year age and time intervals. The rates in this matrix should not be pre-multiplied by the width of the age range; total fertility rate is $5 \sum_a f_{a,t}$. Rows corresonding to ages for which fertility is assumed to be zero should contain all zeros. For Burkina Faso females, we have

```
> burkina.faso.females$fertility.rates
         1960       1965       1970       1975      1980       1985
0  0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
5  0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
10 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
15 0.23106147 0.23804582 0.21518491 0.20215256 0.1862599 0.17116025
20 0.36513859 0.37617574 0.36118334 0.35102655 0.3355970 0.32161538
25 0.30798362 0.31729312 0.32172783 0.32253858 0.3190999 0.31455564
30 0.25784739 0.26564141 0.27617698 0.28052655 0.2810202 0.27886583
35 0.17804816 0.18343007 0.20440064 0.21453214 0.2221399 0.22421363
40 0.09118707 0.09394340 0.11090145 0.11878804 0.1253564 0.12492814
45 0.01953255 0.02012296 0.02936577 0.03366143 0.0377618 0.03836757
50 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
55 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
60 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
65 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
70 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
75 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
80 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0.00000000
         1990       1995       2000
0  0.00000000 0.00000000 0.00000000
5  0.00000000 0.00000000 0.00000000
10 0.00000000 0.00000000 0.00000000
15 0.15774539 0.15017654 0.14602395
20 0.30946628 0.29813440 0.28785888
25 0.30698795 0.29369783 0.28117157
30 0.27138711 0.25797783 0.24662582
35 0.21880702 0.20568412 0.19252220
40 0.11853181 0.10822704 0.09985643
45 0.03745104 0.03471384 0.03291629
50 0.00000000 0.00000000 0.00000000
55 0.00000000 0.00000000 0.00000000
60 0.00000000 0.00000000 0.00000000
65 0.00000000 0.00000000 0.00000000
70 0.00000000 0.00000000 0.00000000
75 0.00000000 0.00000000 0.00000000
80 0.00000000 0.00000000 0.00000000
>
```

**Survival Proportions** The survival proportions, $s_{a,t}$, give the proportion of those aged $a-5$ to $a$ at time $t$ who survive to be aged $a$ to $a+5$ at time $t+5$. Also, $s_{80,t}$ is the proportion aged $[75, 80)$ at exact time $t$ who survive to time $t+5$, by which time they are in the age group $[85, \infty)$. We allow for subsequent survival in this age group by letting $s_{85,t}$ be the proportion aged $[85, \infty)$ at time $t$ who survive five more years. The matrix of survival proportions for Burkina Faso females has the same form as the matrix of fertility rates.

**Migration Proportions** Net international migration, $g_{a,t}$, is measured as a proportion of the respective age-, time-specific population size. Therefore, the net number of migrants aged $[a, a+5)$ to the population during the time period $[t, t+5)$ is $g_{a,t}n_{a,t}$. The matrix of survival proportions for Burkina Faso females has the same form as the matrix of fertility rates.

**Population Counts at Baseline** The total number of people in each age group at the baseline year, $n_{a,t_0}$ is entered as a single column matrix. In our case,

```
> burkina.faso.females$baseline.pop.counts
     1960
0  386000
5  292000
10 260000
15 244000
20 207000
25 175000
30 153000
35 135000
40 117000
45  98000
50  78000
55  60000
60  43000
65  29000
70  17000
75   8000
80   2000
>
```

**Census Counts** Bias reduced census counts are also required for at least one of the years between the baseline year and the end year. These must be at regular five-yearly intervals to coincide with the five-yearly intervals of the initial estimates. Censuses were conducted in Burkina Faso in 1975, 1985, 1995 and 2005. The census count matrix follows the same form as the fertility, survival and migration matrices.

## 3.2 MCMC Control Parameters

The reconstruction is done by the function `popRecon.sampler()`. Among other arguments, this function requires the size of the MCMC sample to be specified via `n.iter` and the additional number of burn-in iterations via `burn.in`. The parameters are updated using Metropolis steps with Gaussian random walk proposals. Each age- time-specific parameter has its own proposal variance which can be modified to achieve an acceptable proportion of acceptances. The variances must be supplied via the `prop.vars` argument. This must be a list with components `fert.rate`, `surv.prop`, `mig.prop` and `baseline.pop.count`. Each component is a matrix with the same shape as the corresponding matrix of initial estimates (Section 3.1), except for the `fert.rate` component. The matrix of proposal variances for fertility rates has the rows corresponding to ages of zero fertility removed. Alternatively, the elements can be a vector such that the $i$th element corresponds to the same age and time as the $i$th element of the corresponding component of `burkina.faso.females`, after removing rows of non-zero fertility for the fertility rate matrix. This is the same ordering achieved by applying `as.vector()` to the proposal variance matrices.

Metropolis proposal variances for this example are in the object `burkina.faso.prop.vars`, loaded by `data(burkina-faso-females)`.

### 3.3 Calling `popRecon.sampler()`

This runs the sampler for 50000 iterations with a burn in of 500 iterations, storing every 50th:

```
> ## set the seed random for the random number generator
> set.seed(1)
> ###
> ### The reconstruction:
> ###
> ##
> ## !!! WARNING: This takes over 24 hours !!!
> ##
> ## commented out --->|
> ## BKFem.Recon.MCMC <-
> ##     popRecon.sampler(## Size of the MCMC sample and burn in
> ##                      n.iter = 4E4,
> ##                      burn.in = 500,
> ##                      thin.by = 50,
>
> ##                      ## initial estimates and census counts
> ##                      mean.f = burkina.faso.females$fertility.rates,
> ##                      mean.s = burkina.faso.females$survival.proportions,
> ##                      mean.g = burkina.faso.females$migration.proportions,
> ##                      mean.b = burkina.faso.females$baseline.pop.counts,
> ##                      pop.data = burkina.faso.females$census.pop.counts,
>
> ##                      ## Metropolis proposal variances
> ##                      prop.vars = burkina.faso.prop.vars,
> ##                      verb=TRUE
> ##                      )
> ## save(BKFem.Recon.MCMC, file = "Burkina_Faso_Recon_RESULTS.RData")
> ## |<--- end comment
> load(file = "Burkina_Faso_Recon_RESULTS.RData")
>
```

## 4 Results

In this section, we illustrate how the joint posterior generated by `popRecon.sampler()` might be summarized. `popRecon.sampler()` returns a list containing (among other things) the MCMC chains for each of the age-specific input parameters, namely fertility rates, survival proportions, migration proportions and the population counts at baseline. These are of class `mcmc` from the **coda** package.

### 4.1 Age-Specific Parameters

The marginal posterior distributions for all age-specific input parameters can be summarized by plotting upper and lower quantiles of 95 percent Bayesian confidence intervals (credible intervals) and the posterior median. These are shown in Figures 1–4. These plots are based on the

`fert.rate.mcmc`, `surv.prop.mcmc`, `mig.prop.mcmc` and `baseline.count.mcmc` objects, which are components of the list returned by `popRecon.sampler()`. For example, the posterior quantiles plotted in the first panel of Figure 1 are

```
> apply(BKFem.Recon.MCMC$fert.rate.mcmc, 2, "quantile", c(0.025, 0.5, 0.975))[,1:7]
        1960.15   1960.20   1960.25   1960.30   1960.35    1960.40
2.5%  0.1817824 0.2915397 0.2423299 0.2068800 0.1400807 0.07565362
50%   0.2254250 0.3515437 0.3003057 0.2524275 0.1773157 0.09187010
97.5% 0.2661050 0.4192471 0.3621569 0.3067635 0.2167117 0.11092761
         1960.45
2.5%  0.01612173
50%   0.01951576
97.5% 0.02412479
>
```

R code to produce these plots is in Appendix A.

## 4.2 Age-Summarized Parameters

Posterior estimates for standard age-summarized parameters can also be produced. Here we show total fertility rate (TFR), life expectancy at birth ($e_0$) and total average annual net number of migrants.

### 4.2.1 Total Fertility Rate

Total fertility rate is defined as

$$\mathrm{TFR}_t = 5 \sum_a f_{a,t}.$$

Ninety-five percent Bayesian confidence intervals and posterior medians are shown in Figure 5. These were calculated from the age-specific fertility rate MCMC, `BKFem.Recon.MCMC$fert.rate.mcmc`, chains using the code in Appendix B.1.

### 4.2.2 Life Expectancy at Birth

In a stationary population, the survival proportions can be converted to life expectancies at birth in each five-year period, $e_{0,t}$, by the transformation

$$e_{0,t} = 5 \sum_{a=0}^{80} \prod_{i=0}^{a} s_{i,t} + \left( \prod_{i=0}^{80} s_{i,t} \right) \left( s_{85+,t} / (1 - s_{85+,t}) \right), \tag{1}$$

Ninety-five percent Bayesian confidence intervals and posterior medians are shown in Figure 7. These were calculated from the age-specific survival proportion MCMC chains, `BKFem.Recon.MCMC$-surv.prop.mcmc`, using the function `life.expectancy.stationary()` and the code in Appendix B.2.

### 4.2.3 Total Average Annual Net Migration

A posterior sample of total net number of migrants can be calculated using the posterior samples of the age-specific migration proportions and age-specific population counts at each five-year step within the interval of reconstruction. These can be calculated by deterministically projecting the posterior samples of age-specific fertility rates, survival proportions, migration proportions

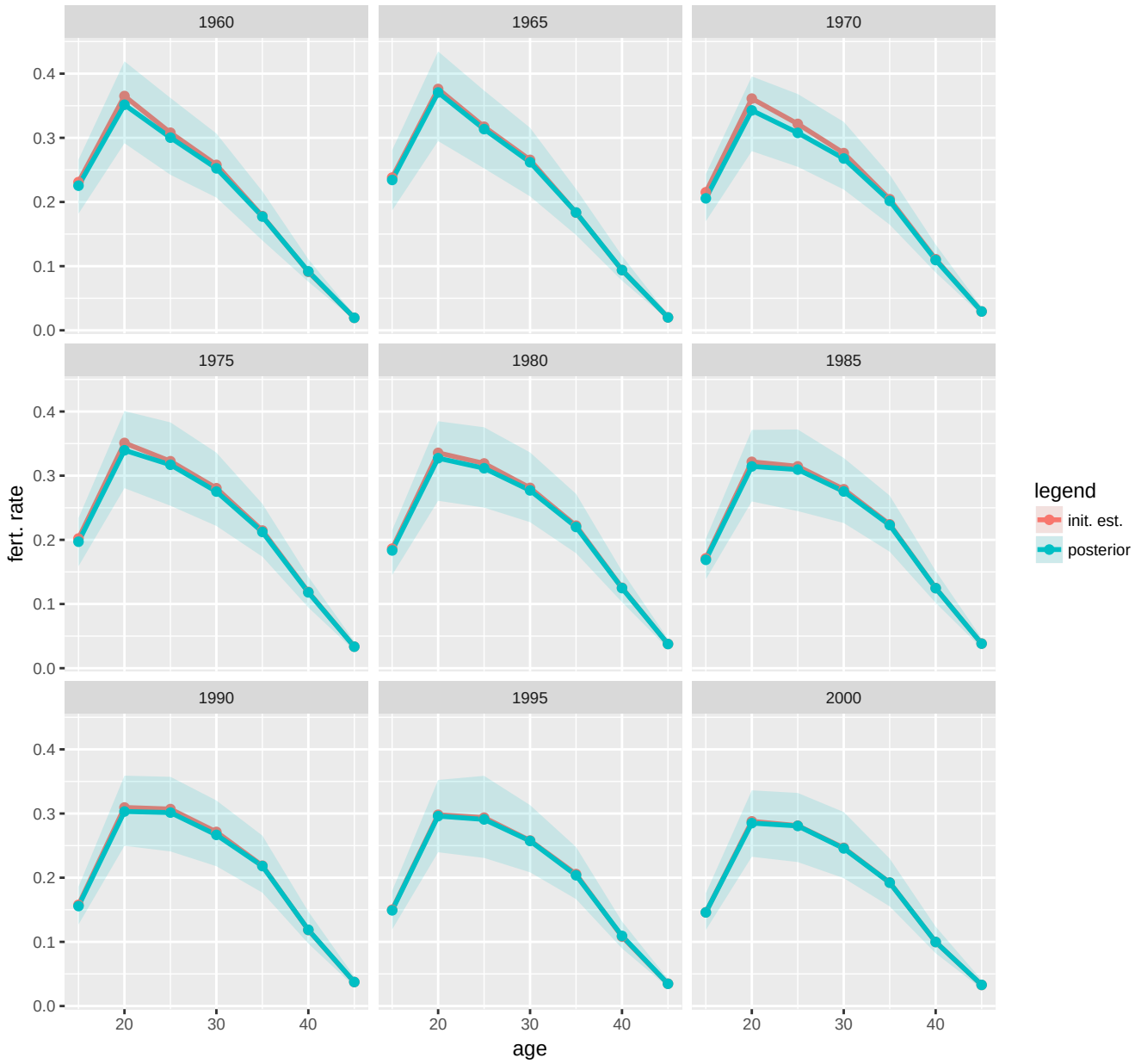


**Figure 1.** Ninety-five percent Bayesian confidence intervals and posterior medians for age-specific fertility rates for the female population of Burkina Faso, 1960–2005. Also shown are the initial estimates.

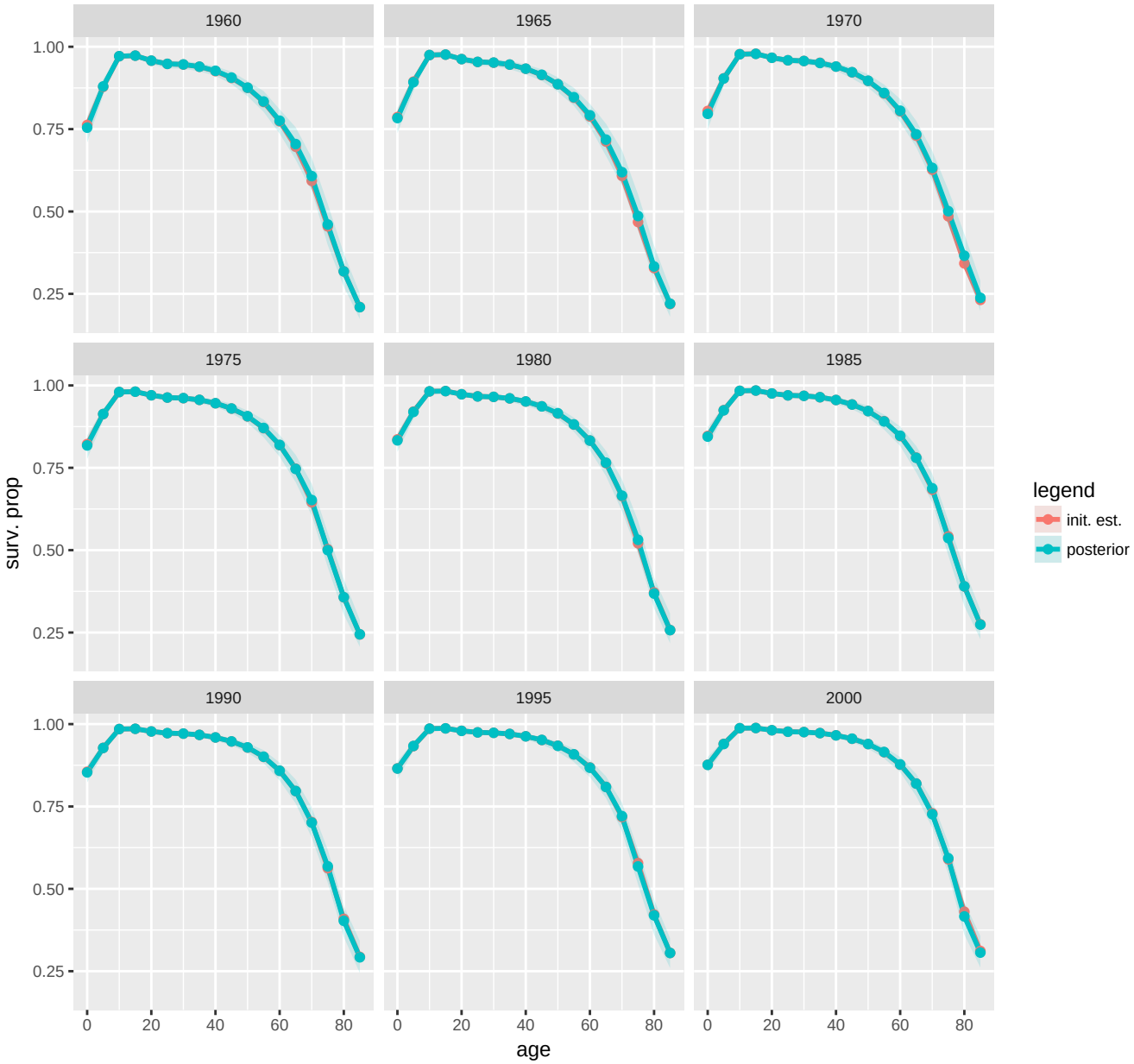


**Figure 2.** Ninety-five percent Bayesian confidence intervals and posterior medians for age-specific survival proportions for the female population of Burkina Faso, 1960–2005. Also shown are the initial estimates.

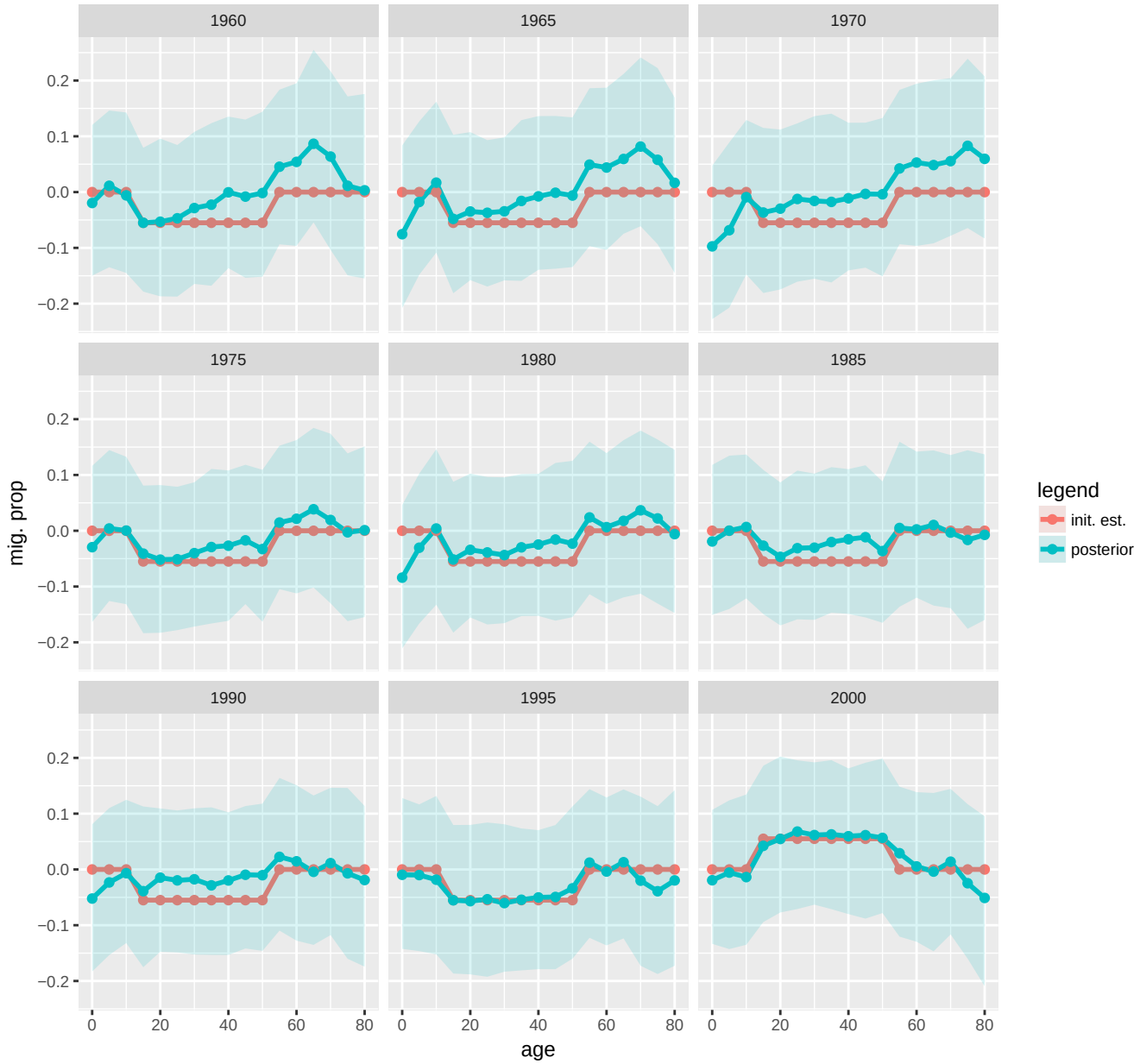


**Figure 3.** Ninety-five percent Bayesian confidence intervals and posterior medians for age-specific migration proportions for the female population of Burkina Faso, 1960–2005. Also shown are the initial estimates.

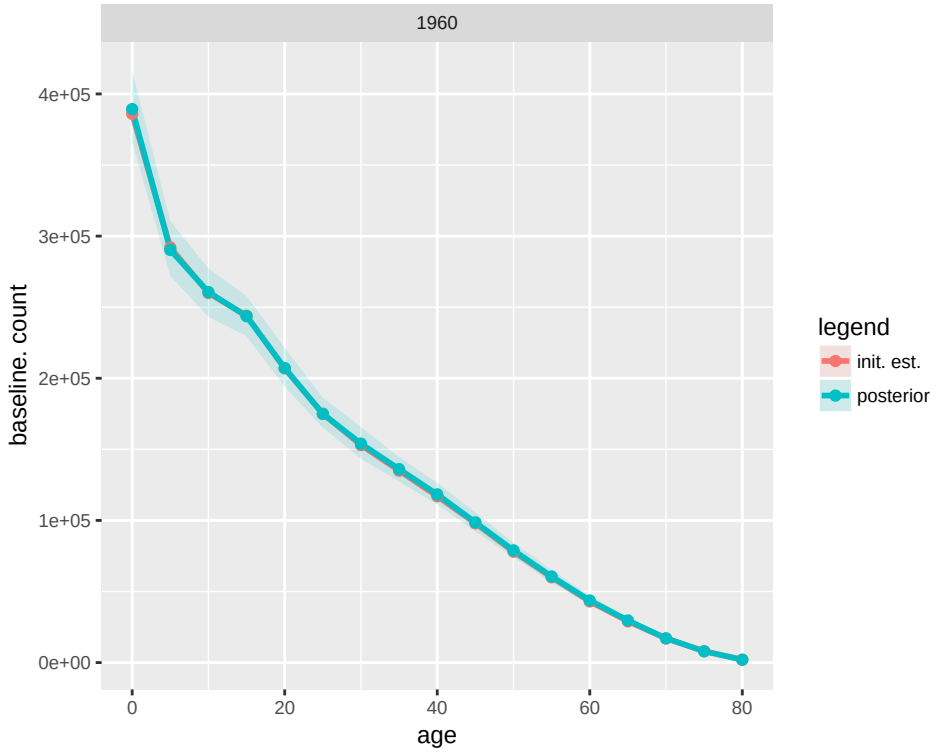


**Figure 4.** Ninety-five percent Bayesian confidence intervals and posterior medians for age-specific baseline counts for the female population of Burkina Faso in 1960. Also shown are the initial estimates.

and baseline counts. The deterministic projection is done by the cohort component method of population projection (CCMPP) (e.g., Preston et al. 2001, Ch. 6), i.e.,

$$\mathbf{n}_{t+5}^{[k]} = L^{[k]} \cdot (\mathbf{n}_t^{[k]} + (1/2) \cdot \widetilde{\mathbf{g}}_t^{[k]}) + (1/2) \cdot \widetilde{\mathbf{g}}_t^{[k]} \tag{2}$$

where $k$ indexes elements of the posterior sample and runs from 1 to the value of `n.iter` passed to `popRecon.sampler()` (see Section 3), $L$ is the Leslie matrix of the projection, $I$ is the identity matrix, and $\widetilde{\mathbf{g}}_t^{[k]} \equiv (\mathbf{g}_t^{[k]})' \mathbf{n}_t^{[k]}$. The average annual net number of migrants for iteration $k$ over the interval $[t, t+5)$, $\widetilde{\mathbf{g}}_t^{[k]}$, is then the solution to (2):

$$\widetilde{\mathbf{g}}_t^{[k]} = 2 \cdot (L^{[k]} + I)^{-1} \cdot (n_{t+5}^{[k]} - L^{[k]} \cdot n_t^{[k]})$$

The CCMPP is implemented in the function `popRecon.ccmp.female`.

Ninety-five percent Bayesian confidence intervals and posterior medians are shown in Figure 7. These were calculated by the code in Appendix B.3 which uses the posterior changes of all the input parameters, namely the components `fert.rate.mcmc`, `surv.prop.mcmc` and `mig.prop.mcmc` in `BKFem.Recon.MCMC`.

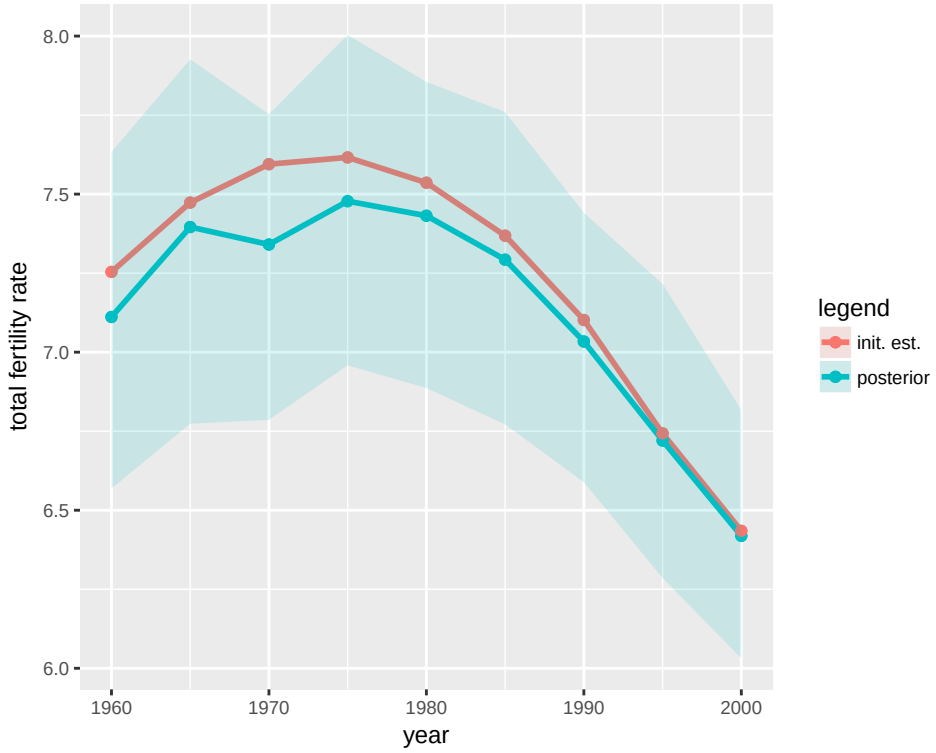


**Figure 5.** Ninety-five percent Bayesian confidence intervals and posterior medians for TFR for the female population of Burkina Faso, 1960–2005. Also shown are the initial estimates.

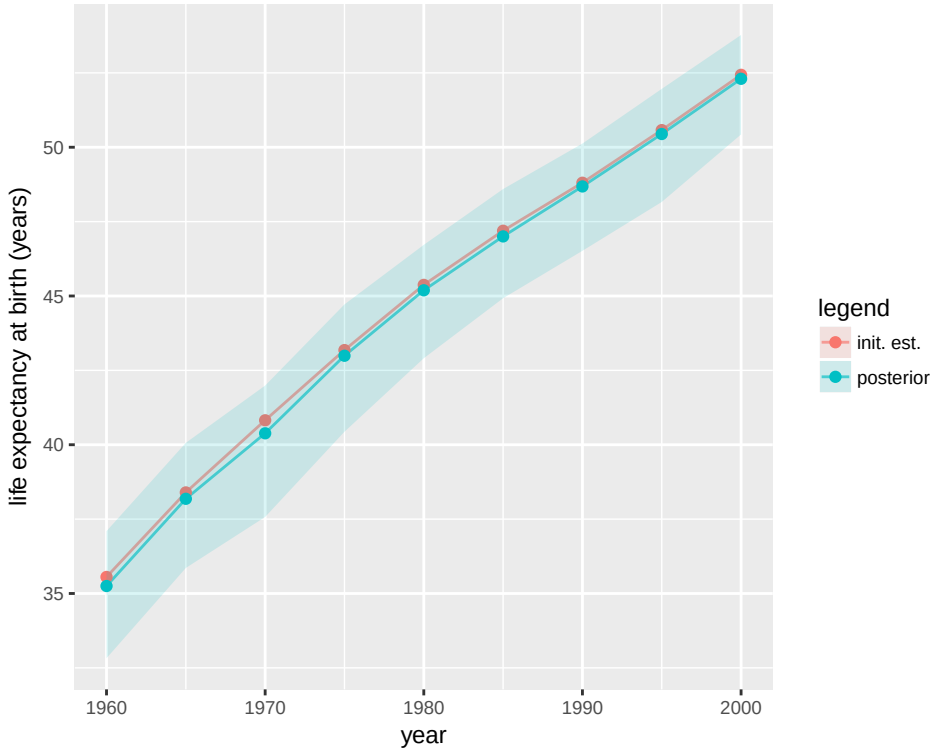


**Figure 6.** Ninety-five percent Bayesian confidence intervals and posterior medians for $e_0$ for the female population of Burkina Faso, 1960–2005. Also shown are the initial estimates.

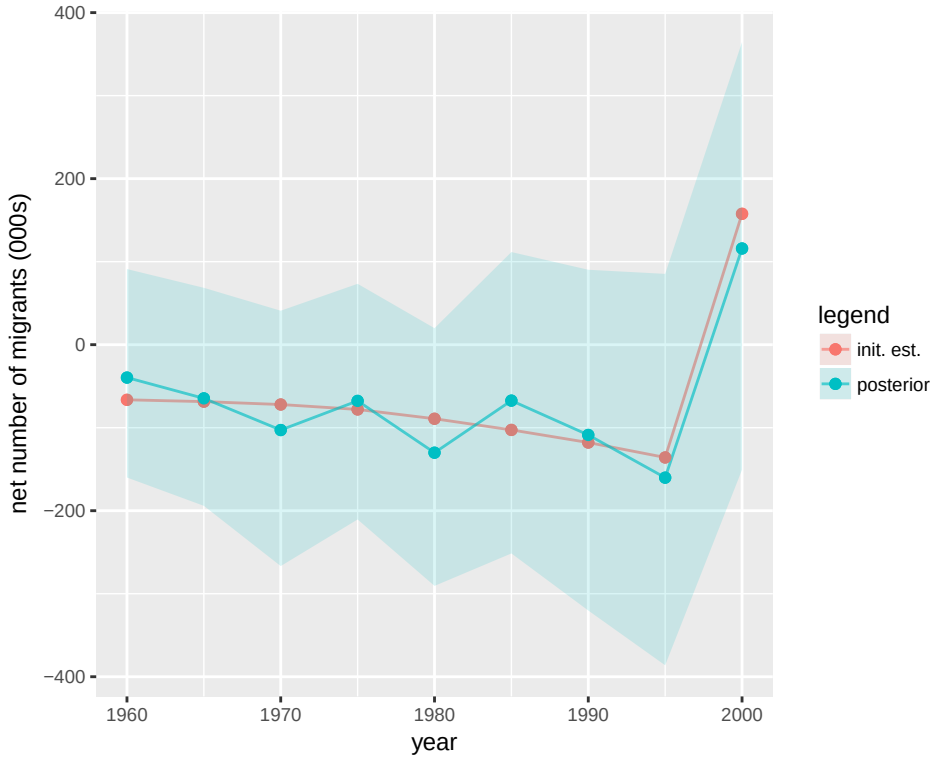


**Figure 7.** Ninety-five percent Bayesian confidence intervals and posterior medians for total average annual net number of migrants for the female population of Burkina Faso, 1960–2005. Also shown are the initial estimates.

# A Code to Produce Plots of Age-Specific Parameters

## A.1 Fertility Rates

```
> ########################################################################
> ###
> ### Calculate posterior quantiles for age-specific fertility rate and
> ### plot
> ###
> ########################################################################
>
> require(ggplot2)
> require(gdata)
> ##
> ## Posterior quantiles
> ##
> vital.chain <- BKFem.Recon.MCMC$fert.rate.mcmc
> q.to.plot = c(0.025, 0.5, 0.975)
> q.vital <- apply(vital.chain, 2, function(z) quantile(z, probs = q.to.plot))
> dimnames(q.vital) <- list(as.character(q.to.plot), colnames(vital.chain))
> ##
> ## Age, year labels
> ##
> colspl <- strsplit(colnames(vital.chain), ".", fixed = TRUE)
> years <- unique(sapply(colspl, FUN = function(z) z[1]))
> fert.ages <- unique(sapply(colspl, FUN = function(z) z[2]))
> fert.ages.numeric <- as.numeric(gsub("[^0-9]", "", fert.ages))
> ##
> ## Reshape data frame
> ##
> qvit.melt <- melt(q.vital)
> qvit.melt.col <- cbind(qvit.melt
                   ,expand.grid(quant = q.to.plot, ages = fert.ages.numeric
                               ,years = years)
                   )
> ##
> ## Initial estimates
> ##
> nzfr <- BKFem.Recon.MCMC$alg.params$non.zero.fert.rows
> vital.init.est <-
     BKFem.Recon.MCMC$fixed.params$mean.fert.rate[nzfr,]
> vital.init.est.melt.col <-
     cbind(value = melt(vital.init.est)$value
           ,expand.grid(ages = fert.ages.numeric
                        ,years = years, quant = 5) # use quant=5 for init.est
         )
> ##
> ## Prepare data sets
> ##
> alpha <- BKFem.Recon.MCMC$fixed.params$alpha.fert.rate
> beta <- BKFem.Recon.MCMC$fixed.params$beta.fert.rate
> qvit.melt.df <- t(q.vital)
> colnames(qvit.melt.df) <-
     paste("fert.rate.", prettyNum(as.numeric(colnames(qvit.melt.df)) * 100)
```

```
          , "pctl", sep = "")
> qvit.melt.df <-
    data.frame(qvit.melt.df
               ,age = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                ,"[[", 2)
               ,year = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                ,"[[", 1)
               ,legend = "posterior"
               )
> vital.init.est <-
    melt(BKFem.Recon.MCMC$fixed.params$mean.fert.rate[nzfr,])
> vital.init.est <-
    rename.vars(vital.init.est, from = c("X1", "X2", "value")
                ,to = c("age", "year", "fert.rate.50pctl"))
> vital.init.est.melt.df <-
    data.frame(vital.init.est, fert.rate.97.5pctl = NA
               ,fert.rate.2.5pctl = NA
               ,legend = "init. est."
               )
> plot.df <- rbind(vital.init.est.melt.df, qvit.melt.df)
> plot.df$age <- as.numeric(plot.df$age)
> plot.df$year <- as.numeric(plot.df$year)
> plot.df$legend <- relevel(factor(plot.df$legend), ref = "init. est.")
> ##
> ## Plot quantiles
> ##
> print(
      ggplot(data = plot.df, aes(x = age, y = fert.rate.50pctl, color = legend)) +
      facet_wrap(~ year) +
      geom_line(size = 1) +
      geom_point() +
      geom_ribbon(aes(ymin = fert.rate.2.5pctl
                      ,ymax = fert.rate.97.5pctl, fill = legend), alpha = 0.15
                  ,color = NA
                  ) +
      ylab("fert. rate")
      )
>
```

## A.2 Survival Proportions

```
> ##########################################################################
> ###
> ### Calculate posterior quantiles for age-specific survival
> ### proportion and plot
> ###
> ##########################################################################
>
> require(ggplot2)
> require(gdata)
> ##
> ## Posterior quantiles
```

```
> ##
> vital.chain <- BKFem.Recon.MCMC$surv.prop.mcmc
> q.to.plot = c(0.025, 0.5, 0.975)
> q.vital <- apply(vital.chain, 2, function(z) quantile(z, probs = q.to.plot))
> dimnames(q.vital) <- list(as.character(q.to.plot), colnames(vital.chain))
> ##
> ## Age, year labels
> ##
> colspl <- strsplit(colnames(vital.chain), ".", fixed = TRUE)
> years <- unique(sapply(colspl, FUN = function(z) z[1]))
> surv.ages <- unique(sapply(colspl, FUN = function(z) z[2]))
> surv.ages.numeric <- as.numeric(gsub("[^0-9]", "", surv.ages))
> ##
> ## Reshape data frame
> ##
> qvit.melt <- melt(q.vital)
> qvit.melt.col <- cbind(qvit.melt
                   ,expand.grid(quant = q.to.plot, ages = surv.ages.numeric
                               ,years = years)
                   )
> ##
> ## Initial estimates
> ##
> vital.init.est <-
    BKFem.Recon.MCMC$fixed.params$mean.surv.prop
> vital.init.est.melt.col <-
    cbind(value = melt(vital.init.est)$value
          ,expand.grid(ages = surv.ages.numeric
                       ,years = years, quant = 5) # use quant=5 for init.est
        )
> ##
> ## Prepare data sets
> ##
> alpha <- BKFem.Recon.MCMC$fixed.params$alpha.surv.prop
> beta <- BKFem.Recon.MCMC$fixed.params$beta.surv.prop
> qvit.melt.df <- t(q.vital)
> colnames(qvit.melt.df) <-
    paste("surv.prop.", prettyNum(as.numeric(colnames(qvit.melt.df)) * 100)
          , "pctl", sep = "")
> qvit.melt.df <-
    data.frame(qvit.melt.df
               ,age = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                ,"[[", 2)
               ,year = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                ,"[[", 1)
               ,legend = "posterior"
               )
> vital.init.est <-
    melt(BKFem.Recon.MCMC$fixed.params$mean.surv.prop)
> vital.init.est <-
    rename.vars(vital.init.est, from = c("X1", "X2", "value")
                ,to = c("age", "year", "surv.prop.50pctl"))
> vital.init.est.melt.df <-
    data.frame(vital.init.est, surv.prop.97.5pctl = NA
```

```
                ,surv.prop.2.5pctl = NA
                ,legend = "init. est."
                )
> plot.df <- rbind(vital.init.est.melt.df, qvit.melt.df)
> plot.df$age <- as.numeric(plot.df$age)
> plot.df$year <- as.numeric(plot.df$year)
> plot.df$legend <- relevel(factor(plot.df$legend), ref = "init. est.")
> ##
> ## Plot quantiles
> ##
> print(
       ggplot(data = plot.df, aes(x = age, y = surv.prop.50pctl, color = legend)) +
       facet_wrap(~ year) +
       geom_line(size = 1) +
       geom_point() +
       geom_ribbon(aes(ymin = surv.prop.2.5pctl
                       ,ymax = surv.prop.97.5pctl, fill = legend), alpha = 0.15
                   ,color = NA) +
       ylab("surv. prop")
       )
>
```

## A.3 Migration Proportions

```
> ##########################################################################
> ###
> ### Calculate posterior quantiles for age-specific migration
> ### proportion and plot
> ###
> ##########################################################################
>
> require(ggplot2)
> require(gdata)
> ##
> ## Posterior quantiles
> ##
> vital.chain <- BKFem.Recon.MCMC$mig.prop.mcmc
> q.to.plot = c(0.025, 0.5, 0.975)
> q.vital <- apply(vital.chain, 2, function(z) quantile(z, probs = q.to.plot))
> dimnames(q.vital) <- list(as.character(q.to.plot), colnames(vital.chain))
> ##
> ## Age, year labels
> ##
> colspl <- strsplit(colnames(vital.chain), ".", fixed = TRUE)
> years <- unique(sapply(colspl, FUN = function(z) z[1]))
> mig.ages <- unique(sapply(colspl, FUN = function(z) z[2]))
> mig.ages.numeric <- as.numeric(gsub("[^0-9]", "", mig.ages))
> ##
> ## Reshape data frame
> ##
> qvit.melt <- melt(q.vital)
> qvit.melt.col <- cbind(qvit.melt
```

```
                  ,expand.grid(quant = q.to.plot, ages = mig.ages.numeric
                               ,years = years)
                  )
> ##
> ## Initial estimates
> ##
> vital.init.est <-
    BKFem.Recon.MCMC$fixed.params$mean.mig.prop
> vital.init.est.melt.col <-
    cbind(value = melt(vital.init.est)$value
          ,expand.grid(ages = mig.ages.numeric
                       ,years = years, quant = 5) # use quant=5 for init.est
        )
> ##
> ## Prepare data sets
> ##
> alpha <- BKFem.Recon.MCMC$fixed.params$alpha.mig.prop
> beta <- BKFem.Recon.MCMC$fixed.params$beta.mig.prop
> qvit.melt.df <- t(q.vital)
> colnames(qvit.melt.df) <-
    paste("mig.prop.", prettyNum(as.numeric(colnames(qvit.melt.df)) * 100)
          , "pctl", sep = "")
> qvit.melt.df <-
    data.frame(qvit.melt.df
               ,age = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                ,"[[", 2)
               ,year = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                ,"[[", 1)
               ,legend = "posterior"
               )
> vital.init.est <-
    melt(BKFem.Recon.MCMC$fixed.params$mean.mig.prop)
> vital.init.est <-
    rename.vars(vital.init.est, from = c("X1", "X2", "value")
                ,to = c("age", "year", "mig.prop.50pctl"))
> vital.init.est.melt.df <-
    data.frame(vital.init.est, mig.prop.97.5pctl = NA
               ,mig.prop.2.5pctl = NA
               ,legend = "init. est."
               )
> plot.df <- rbind(vital.init.est.melt.df, qvit.melt.df)
> plot.df$age <- as.numeric(plot.df$age)
> plot.df$year <- as.numeric(plot.df$year)
> plot.df$legend <- relevel(factor(plot.df$legend), ref = "init. est.")
> ##
> ## Plot quantiles
> ##
> print(
      ggplot(data = plot.df, aes(x = age, y = mig.prop.50pctl, color = legend)) +
      facet_wrap(~ year) +
      geom_line(size = 1) +
      geom_point() +
      geom_ribbon(aes(ymin = mig.prop.2.5pctl
                      ,ymax = mig.prop.97.5pctl, fill = legend), alpha = 0.15
```

```
                  ,color = NA) +
       ylab("mig. prop")
       )
>
```

## A.4 Baseline Counts

```
> ##############################################################################
> ###
> ### Calculate posterior quantiles for age-specific baseline count
> ### and plot
> ###
> ##############################################################################
>
> require(ggplot2)
> require(gdata)
> ##
> ## Posterior quantiles
> ##
> vital.chain <- BKFem.Recon.MCMC$baseline.count.mcmc
> q.to.plot = c(0.025, 0.5, 0.975)
> q.vital <- apply(vital.chain, 2, function(z) quantile(z, probs = q.to.plot))
> dimnames(q.vital) <- list(as.character(q.to.plot), colnames(vital.chain))
> ##
> ## Age, year labels
> ##
> colspl <- strsplit(colnames(vital.chain), ".", fixed = TRUE)
> years <- unique(sapply(colspl, FUN = function(z) z[1]))
> baseline.ages <- unique(sapply(colspl, FUN = function(z) z[2]))
> baseline.ages.numeric <- as.numeric(gsub("[^0-9]", "", baseline.ages))
> ##
> ## Reshape data frame
> ##
> qvit.melt <- melt(q.vital)
> qvit.melt.col <- cbind(qvit.melt
                    ,expand.grid(quant = q.to.plot, ages = baseline.ages.numeric
                                 ,years = years)
                    )
> ##
> ## Initial estimates
> ##
> vital.init.est <-
     BKFem.Recon.MCMC$fixed.params$mean.baseline.count
> vital.init.est.melt.col <-
     cbind(value = melt(vital.init.est)$value
           ,expand.grid(ages = baseline.ages.numeric
                        ,years = years, quant = 5) # use quant=5 for init.est
         )
> ##
> ## Prepare data sets
> ##
> alpha <- BKFem.Recon.MCMC$fixed.params$alpha.population.count
```

```
> beta <- BKFem.Recon.MCMC$fixed.params$beta.population.count
> qvit.melt.df <- t(q.vital)
> colnames(qvit.melt.df) <-
     paste("baseline.count.", prettyNum(as.numeric(colnames(qvit.melt.df)) * 100)
           , "pctl", sep = "")
> qvit.melt.df <-
     data.frame(qvit.melt.df
                ,age = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                 ,"[[", 2)
                ,year = sapply(strsplit(rownames(qvit.melt.df), split = "[^0-9]")
                 ,"[[", 1)
                ,legend = "posterior"
                )
> vital.init.est <-
     melt(BKFem.Recon.MCMC$fixed.params$mean.baseline.count)
> vital.init.est <-
     rename.vars(vital.init.est, from = c("X1", "X2", "value")
                 ,to = c("age", "year", "baseline.count.50pctl"))
> vital.init.est.melt.df <-
     data.frame(vital.init.est, baseline.count.97.5pctl = NA
                ,baseline.count.2.5pctl = NA
                ,legend = "init. est."
                )
> plot.df <- rbind(vital.init.est.melt.df, qvit.melt.df)
> plot.df$age <- as.numeric(plot.df$age)
> plot.df$year <- as.numeric(plot.df$year)
> plot.df$legend <- relevel(factor(plot.df$legend), ref = "init. est.")
> ##
> ## Plot quantiles
> ##
> print(
       ggplot(data = plot.df, aes(x = age, y = baseline.count.50pctl, color = legend)) +
       facet_wrap(~ year) +
       geom_line(size = 1) +
       geom_point() +
       geom_ribbon(aes(ymin = baseline.count.2.5pctl
                       ,ymax = baseline.count.97.5pctl, fill = legend), alpha = 0.15
                   ,color = NA) +
       ylab("baseline. count")
       )
>
```

# B Code to Produce Plots of Age-Summarized Parameters

## B.1 Total Fertility Rate

```
> ##############################################################################
> ###
> ### Calculate posterior quantiles for TFR and plot
> ###
> ##############################################################################
>
```

```
> require(ggplot2)
> q.to.plot = c(0.025, 0.5, 0.975)
> ###
> ### Posterior
> ###
> dn <- list(NULL,
             unique(sapply(strsplit(colnames(BKFem.Recon.MCMC$fert.rate.mcmc)
                                    ,"\\."), FUN = function(z) z[[1]])
                    )
             )
> BKFem.Recon.tfr <-
   matrix(0, nrow = nrow(BKFem.Recon.MCMC$fert.rate.mcmc)
          ,ncol = length(dn[[2]])
          ,dimnames = dn
          )
> fert.rate.mcmc.colYrs <-
   sapply(strsplit(colnames(BKFem.Recon.MCMC$fert.rate.mcmc)
                           ,"\\."), FUN = function(z) z[[1]])
> ##
> ## calculate tfr
> ##
> for(i in 1:ncol(BKFem.Recon.tfr)) {
   colYrs.index <- fert.rate.mcmc.colYrs == colnames(BKFem.Recon.tfr)[i]
   BKFem.Recon.tfr[,i] <-
     apply(BKFem.Recon.MCMC$fert.rate.mcmc[,colYrs.index]
         ,1
         ,FUN = function(z) 5 * sum(z)
         )
 }
> ##
> ## tfr quantiles
> ##
> BKFem.Recon.tfrQuant <- apply(BKFem.Recon.tfr, 2, FUN = function(z)
                          {
                            quantile(z, probs = q.to.plot)
                          })
> BKFem.Recon.tfrQuant.df <-
     as.data.frame(t(BKFem.Recon.tfrQuant))
> colnames(BKFem.Recon.tfrQuant.df) <-
     paste("tfr.", strsplit(colnames(BKFem.Recon.tfrQuant.df), split = "%")
           ,"pctl", sep = "")
> BKFem.Recon.tfrQuant.df$legend = "posterior"
> BKFem.Recon.tfrQuant.df$year = as.numeric(rownames(BKFem.Recon.tfrQuant.df))
> ###
> ### Initial estimates
> ###
> BKFem.Recon.tfr.init.est <-
     data.frame(year = colnames(BKFem.Recon.MCMC$fixed.params$mean.fert.rate)
                ,tfr.50pctl =
                melt(5 * colSums(BKFem.Recon.MCMC$fixed.params$mean.fert.rate))[,1]
                )
> BKFem.Recon.tfr.init.est$legend <- "init. est."
> BKFem.Recon.tfr.init.est$tfr.2.5pctl <-
     BKFem.Recon.tfr.init.est$tfr.97.5pctl <- NA
```

```
> ###
> ### Plot
> ###
> plot.df <-
    rbind(BKFem.Recon.tfrQuant.df, BKFem.Recon.tfr.init.est)
> plot.df$legend <- relevel(factor(plot.df$legend), ref = "init. est.")
> plot.df$year <- as.numeric(plot.df$year)
> print(ggplot(data = plot.df, aes(x = year, y = tfr.50pctl, color = legend)) +
      geom_line(size = 1) +
      geom_point() +
      geom_point() + geom_ribbon(aes(ymin = tfr.2.5pctl
                                     ,ymax = tfr.97.5pctl, fill = legend)
                                 ,alpha = 0.15, color = NA) +
      ylab("total fertility rate")
      )
>
```

## B.2 Life Expectancy at Birth

```
> ############################################################################
> ###
> ### Calculate posterior quantiles for life expectancy at birth and
> ### plot
> ###
> ############################################################################
>
> require(ggplot2)
> q.to.plot = c(0.025, 0.5, 0.975)
> surv.prop.years <-
    sapply(strsplit(colnames(BKFem.Recon.MCMC$surv.prop.mcmc), "\\."), "[[", 1)
> message("Calculating life expectancy at birth ...")
> BKFem.leb.stationary.df <-
    apply(BKFem.Recon.MCMC$surv.prop.mcmc[,], 1, function(z) {
        tapply(z, INDEX = surv.prop.years, FUN = "life.expectancy.stationary")
      })
> message("... done")
> BKFem.leb.stationary.Quantiles <-
    apply(BKFem.leb.stationary.df, 1, "quantile", probs = q.to.plot)
> BKFem.leb.stationary.Quantiles.df <-
    as.data.frame(t(BKFem.leb.stationary.Quantiles))
> colnames(BKFem.leb.stationary.Quantiles.df) <-
    paste("leb."
          , strsplit(colnames(BKFem.leb.stationary.Quantiles.df)
                                              , split = "%")
          ,"pctl", sep = "")
> BKFem.leb.stationary.Quantiles.df$legend <- "posterior"
> BKFem.leb.stationary.Quantiles.df$year <-
    as.numeric(rownames(BKFem.leb.stationary.Quantiles.df))
> ###
> ### Prior by converting posterior quantiles of survival and assuming
> ### stationary population relation holds
> ###
```

```
> lebp.yrs <- as.numeric(colnames(BKFem.Recon.MCMC$fixed.params$mean.surv.prop))
> BKFem.lebPrior.stationary.df <-
     data.frame(year = lebp.yrs
                ,leb.50pctl = apply(BKFem.Recon.MCMC$fixed.params$mean.surv.prop
                 ,2
                 ,FUN = "life.expectancy.stationary"
                 ))
> BKFem.lebPrior.stationary.df$leb.2.5pctl <-
     BKFem.lebPrior.stationary.df$leb.97.5pctl <- NA
> BKFem.lebPrior.stationary.df$legend <- "init. est."
> ###
> ### Plot
> ###
> plot.df <-
     rbind(BKFem.lebPrior.stationary.df, BKFem.leb.stationary.Quantiles.df)
> plot.df$legend <- relevel(factor(plot.df$legend), ref = "init. est.")
> print(ggplot(data = plot.df, aes(x = year, y = leb.50pctl, color = legend)) +
       geom_line(alpha = 0.65) +
       geom_point() +
       geom_ribbon(aes(ymin = leb.2.5pctl
                                    ,ymax = leb.97.5pctl, fill = legend)
                                ,alpha = 0.15, color = NA) +
       ylab("life expectancy at birth (years)")
       )
>
```

## B.3 Total Average Annual Net Migration

```
> ##############################################################################
> ###
> ### Calculate posterior quantiles for average annual total net
> ### number of migrants
> ###
> ##############################################################################
>
> require(ggplot2)
> q.to.plot = c(0.025, 0.5, 0.975)
> ## NB: Can't simply sum migration proportions because they are based
> ##     on different population totals. Need to get net number of migrants
> ##     and convert back into proportions. Use Leslie matrix formula from
> ##     article draft.
> ##
>
> ###
> ### Posterior distribution
> ###
>
> ##
> ## Prepare output matrix
> ##
> BKFem.Recon.netMig <-
   matrix(0, nrow = nrow(BKFem.Recon.MCMC$mig.prop.mcmc)
```

```
          ,ncol = ncol(BKFem.Recon.MCMC$fixed.params$mean.mig.prop)
          ,dimnames = list(NULL,
             unique(sapply(strsplit(colnames(BKFem.Recon.MCMC$mig.prop.mcmc)
                                    ,"\\."), FUN = function(z) z[[1]])
                    )
             )
          )
> ##
> ## The 5-year sub-intervals to be used as an index into the columns of
> ## BKFem.Recon.netMig
> ##
> mig.prop.mcmc.colYrs <-
   sapply(strsplit(colnames(BKFem.Recon.MCMC$mig.prop.mcmc)
                           ,"\\."), FUN = function(z) z[[1]])
> mig.prop.mcmc.colYrsUniq <- unique(mig.prop.mcmc.colYrs)
> ##
> ## Years used in survival proportions
> ##
> surv.prop.mcmc.colYrs <-
     sapply(strsplit(colnames(BKFem.Recon.MCMC$surv.prop.mcmc)
                     ,"\\."), FUN = function(z) z[[1]])
> ##
> ## Concatenate baseline and lx to get a single matrix with population
> ## counts
> ##
>
> pop.mat <- cbind(BKFem.Recon.MCMC$baseline.count.mcmc
                ,BKFem.Recon.MCMC$lx.mcmc)
> ##
> ## Index for population years
> ##
>
> pop.mat.colYrs <- sapply(strsplit(colnames(pop.mat)
                           ,"\\."), FUN = function(z) z[[1]])
> pop.mat.colYrsUniq <- unique(pop.mat.colYrs)
> message("Calculating net number of migrants ...")
> for(k in 1:nrow(BKFem.Recon.MCMC$mig.prop.mcmc)) {
     if(k %% 1000 == 0)
         message(paste("row ", k, " of "
                       ,nrow(BKFem.Recon.MCMC$mig.prop.mcmc), sep = "")
                 )
     ##
     ## cycle through years
     for(i in 1:ncol(BKFem.Recon.netMig)) {
         ##
         ## 5-year sub-intervals for indexing columns
         mig.colYrs.index <-
             colnames(BKFem.Recon.netMig) == mig.prop.mcmc.colYrsUniq[i]
         surv.colYrs.index <-
             surv.prop.mcmc.colYrs == mig.prop.mcmc.colYrsUniq[i]
         fert.colYrs.index <-
             fert.rate.mcmc.colYrs == mig.prop.mcmc.colYrsUniq[i]
         pop.colYrs.index1 <-
             pop.mat.colYrs == mig.prop.mcmc.colYrsUniq[i]
```

```
        pop.colYrs.index2 <-
            pop.mat.colYrs == as.numeric(mig.prop.mcmc.colYrsUniq[i]) + 5
        ##
        ## get vital rates and make leslie matrix
        sk <- BKFem.Recon.MCMC$surv.prop.mcmc[k,surv.colYrs.index]
        fk <- rep(0, nrow(BKFem.Recon.MCMC$fixed.params$mean.fert.rate))
        fk[BKFem.Recon.MCMC$alg.params$non.zero.fert.rows] <-
            BKFem.Recon.MCMC$fert.rate.mcmc[k,fert.colYrs.index]
        popk1 <- pop.mat[k,pop.colYrs.index1]
        popk2 <- pop.mat[k,pop.colYrs.index2]
        Lk <- make.leslie.matrix(pop = popk1, surv = sk, fert = fk, srb = 1.05
                          ,age.int = 5)
        ##
        ## calculate net number of migrants
        netMigk <- net.number.migrants(n1 = popk1, n2 = popk2, L = Lk)
        ##
        ## store
        BKFem.Recon.netMig[k, mig.colYrs.index] <- sum(netMigk)
    }
 }
> message("... done")
> ##
> ## Posterior quantiles
> ##
> BKFem.nmig.post.quant <-
    apply(BKFem.Recon.netMig, 2, FUN = function(z)
                         {
                           quantile(z, probs = q.to.plot)
                         })
> BKFem.nmig.post.quant.df <-
    as.data.frame(t(BKFem.nmig.post.quant))
> colnames(BKFem.nmig.post.quant.df) <-
    paste("total.mig.count.", strsplit(colnames(BKFem.nmig.post.quant.df)
                                       , split = "%")
          ,"pctl", sep = "")
> BKFem.nmig.post.quant.df$legend <- "posterior"
> BKFem.nmig.post.quant.df$year <-
    as.numeric(rownames(BKFem.nmig.post.quant.df))
> ###
> ### Initial estimates
> ###
>
> ##
> ## Prepare output matrix
> ##
> BKFem.nmig.input <- rep(0, ncol(BKFem.Recon.MCMC$fixed.params$mean.mig.prop))
> names(BKFem.nmig.input) <-
    colnames(BKFem.Recon.MCMC$fixed.params$mean.mig.prop)
> ##
> ## Input population counts
> ##
> pop.input.mat <-
    popRecon.ccmp.female(pop=BKFem.Recon.MCMC$fixed.params$mean.baseline.count
                      ,surv=BKFem.Recon.MCMC$fixed.params$mean.surv.prop
```

```
                      ,fert=BKFem.Recon.MCMC$fixed.params$mean.fert.rate
                      ,mig=BKFem.Recon.MCMC$fixed.params$mean.mig.prop
                      )
> ##
> ## Calculate input net migration
> ##
> for(k in 1:(ncol(pop.input.mat)-1)) {
     Lk <- make.leslie.matrix(pop = pop.input.mat[,k]
                       ,surv = BKFem.Recon.MCMC$fixed.params$mean.surv.prop[,k]
                       ,fert = BKFem.Recon.MCMC$fixed.params$mean.fert.rate[,k]
                       ,srb = 1.05
                       ,age.int = 5)
         netMigk <- net.number.migrants(n1 = pop.input.mat[,k]
                                ,n2 = pop.input.mat[,k+1]
                                ,L = Lk)
     BKFem.nmig.input[k] <- sum(netMigk)
 }
> BKFem.nmig.input.df <-
     data.frame(year = as.numeric(names(BKFem.nmig.input))
                ,total.mig.count.50pctl = BKFem.nmig.input
                )
> BKFem.nmig.input.df$total.mig.count.2.5pctl <- NA
> BKFem.nmig.input.df$total.mig.count.97.5pctl <- NA
> BKFem.nmig.input.df$legend <- "init. est."
> ###
> ### Plot
> ###
> plot.df <- rbind(BKFem.nmig.input.df, BKFem.nmig.post.quant.df)
> plot.df$year <- as.numeric(plot.df$year)
> plot.df$legend <- relevel(factor(plot.df$legend), ref = "init. est.")
> print(ggplot(data = plot.df, aes(x = year
              , y = total.mig.count.50pctl/1E3, color = legend)) +
       geom_line(alpha = 0.65) +
       geom_point() +
       geom_point() + geom_ribbon(aes(ymin = total.mig.count.2.5pctl/1E3
                                      ,ymax = total.mig.count.97.5pctl/1E3, fill = legend)
                                  ,alpha = 0.15, color = NA) +
       ylab("net number of migrants (000s)")
       )
>
```

## References


Preston, S. H., Heuveline, P., and Guillot, M. (2001). *Demography: Measuring and Modeling Population Processes*. Blackwell, Malden, Massachusetts.

Wheldon, M. C., Raftery, A. E., Clark, S. J., and Gerland, P. (2011). Estimating demographic parameters with uncertainty from fragmentary data. Working Paper 108, Center for Statistics and the Social Sciences, University of Washington, Seattle, Washington.