Individual-level discrete time competing risks survival modeling

Austin Schumacher

2021-06-14

Simulation

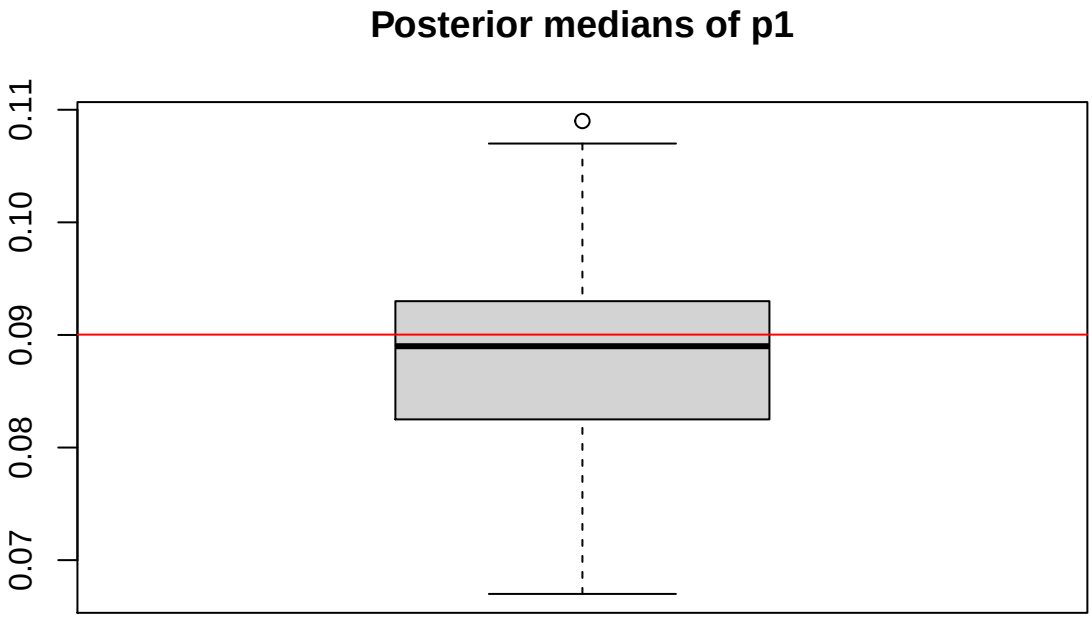
Multinomial

3 categories

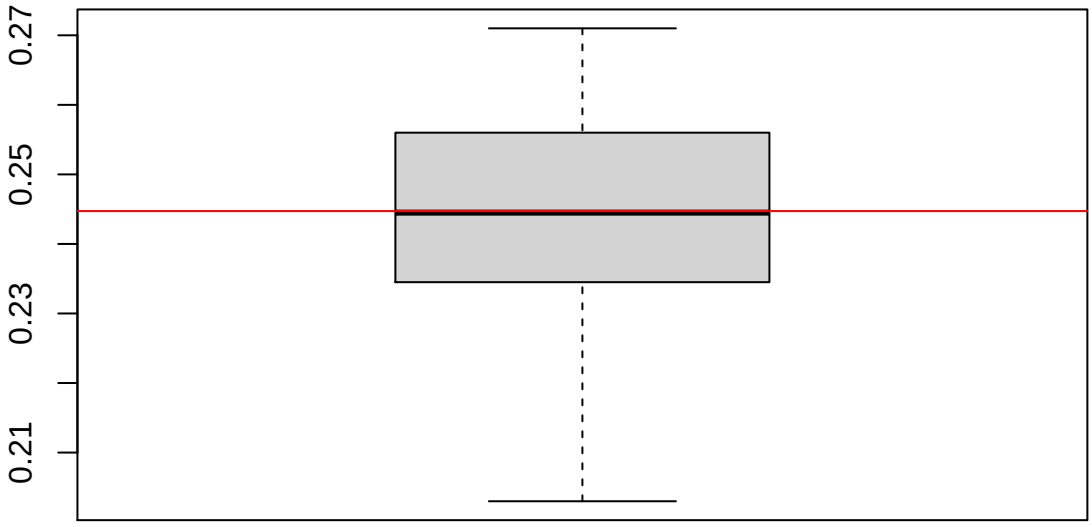
We will first simulate n draws from a multinomial distribution with 3 categories

```
library(devtools)
# install_github("https://github.com/richardli/SUMMER/tree/dev") # Johnny's dev version with pop sim fu
library(svyVGAM)
library(SUMMER)
library(fields)
set.seed(98125)

# sim setup and storage of results
nsim <- 100
p.res <- matrix(NA, nrow = nsim, ncol = 2)
coverage.res <- matrix(NA, nrow = nsim, ncol = 2)
cov.res <-array(NA, dim = c(2, 2, nsim))

# parameters
n <- 1000
alpha <- c(-2, -1)
denom_p <- sum(exp(alpha)) + 1
p_cause <- c(exp(alpha)/denom_p)
p <- c(p_cause, 1 - sum(p_cause))

for (s in 1:nsim) {
    # cat(paste0("Starting sim ", s, "....\n"))
    # simulate data
    y <- rmultinom(n, 1, p)
    dat <- data.frame(obs = 1:n,
                      cause1 = y[1,],
                      cause_other = y[2,],
                      alive = y[3,])

    # fit model
    my.svydesign <- survey::svydesign(ids = ~ 1,
                                      data = dat)

    mult.mod <- svy_vglm(cbind(cause1, cause_other, alive) ~ 1,
```

```
                        family = multinomial,
                        design = my.svydesign)
    denom <- 1 + sum(exp(mult.mod$coef))
    p.res[s, ] <- exp(mult.mod$coef)/denom
    cis <- confint(mult.mod)
    coverage.res[s, 1] <- alpha[1] > cis[1, 1] & alpha[1] < cis[1, 2]
    coverage.res[s, 2] <- alpha[2] > cis[2, 1] & alpha[2] < cis[2, 2]
    cov.res[,,s] <- mult.mod$var
}

boxplot(p.res[, 1], main = "Posterior medians of p1", ylim = range(c(p.res[, 1], p[1])))
abline(h = p[1], col = "red")
```




```
boxplot(p.res[, 2], main = "Posterior medians of p2", ylim = range(c(p.res[, 2], p[2])))
abline(h = p[2], col = "red")
```




```
apply(coverage.res, 2, mean)
```

```
## [1] 0.96 0.94
```

```
apply(cov.res, c(1, 2), mean)
```

```
##              [,1]        [,2]
## [1,] 0.012952300 0.001500962
## [2,] 0.001500962 0.005609496
```

Individual-level SRS

3 categories (1 cause, all other causes, and alive), no age modeling

We will simulate n individuals over 60 months, where in each month for all alive individuals, we apply the multinomial probabilities of dying from cause 1, dying from all other causes, or remaining alive.

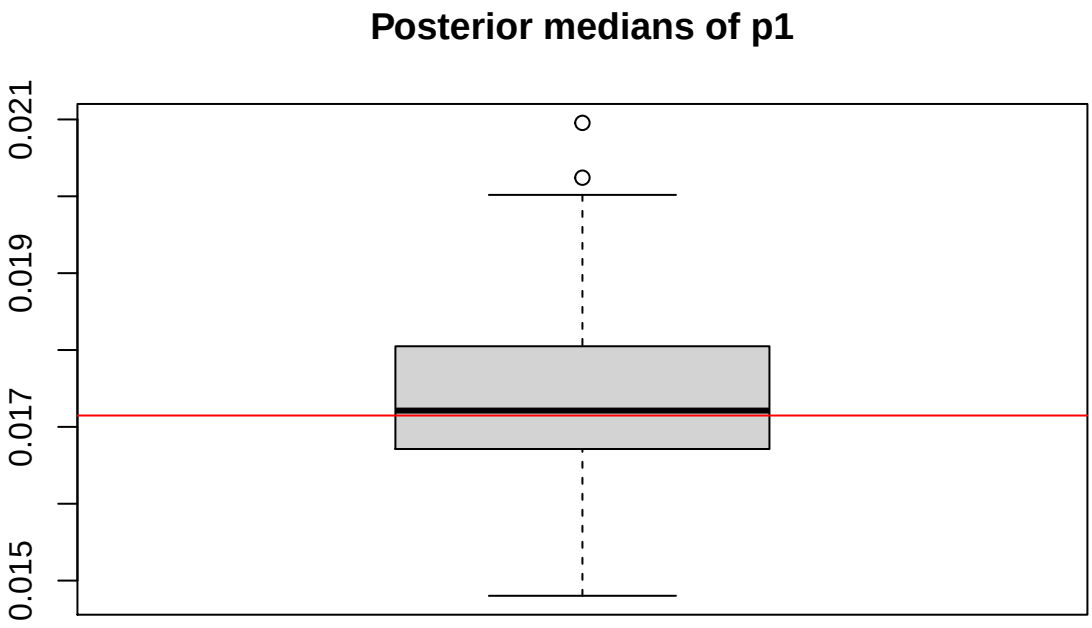
Thus, we have the discrete-time survival data generating mechanism as follows:

TODO write this up

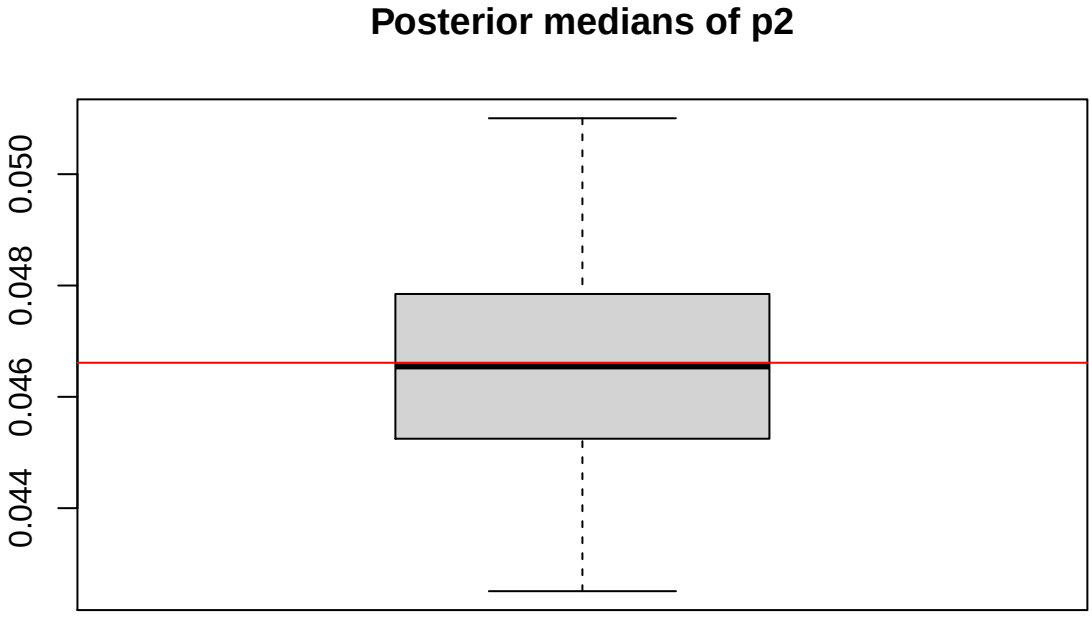
```
# sim setup and storage of results
nsim <- 100
p.res <- matrix(NA, nrow = nsim, ncol = 2)
coverage.res <- matrix(NA, nrow = nsim, ncol = 2)
cov.res <-array(NA, dim = c(2, 2, nsim))

# parameters
n <- 1000
alpha <- c(-4, -3)
denom_p <- sum(exp(alpha)) + 1
p_cause <- c(exp(alpha)/denom_p)
p <- c(p_cause, 1 - sum(p_cause))
p_cum <- cumsum(p)

for (s in 1:nsim) {
```

```
    # simulate data
    y <- vector(mode = "list", length = n)
    for(i in 1:n) {
        y[[i]] <- matrix(NA, ncol = 5, nrow = 60)
        y[[i]][, 4] <- i # person id
        y[[i]][, 5] <- 1:60 # month indicator
        # cat(paste("\nStarting person", i, "\n"))
        t <- 1
        death <- 0
        while(death == 0) {
            # if (t %% 10 == 0) cat(paste("\t month", t, "\n"))
            rand <- runif(1, 0, 1)
            if (rand < p_cum[1]) {
                y[[i]][t, 1:3] <- c(1, 0, 0)
                death <- 1
            } else if (rand >= p_cum[1] & rand < p_cum[2]) {
                y[[i]][t, 1:3] <- c(0, 1, 0)
                death <- 1
            } else {
                y[[i]][t, 1:3] <- c(0, 0, 1)
            }
            t <- t + 1
            if (t == 60) death <- 1
        }
    }

    ymat <- do.call(rbind, y)
    ymat <- ymat[complete.cases(ymat), ]

    # create data frame
    dat <- data.frame(id = ymat[, 4],
                      time = ymat[, 5],
                      cause1 = ymat[, 1],
                      cause_other = ymat[, 2],
                      alive = ymat[, 3])

    # fit model
    my.svydesign <- survey::svydesign(ids = ~ id,
                                      data = dat)

    mult.mod <- svy_vglm(cbind(cause1, cause_other, alive) ~ 1,
                         family = multinomial,
                         design = my.svydesign)
    denom <- 1 + sum(exp(mult.mod$coef))
    p.res[s, ] <- exp(mult.mod$coef)/denom
    cis <- confint(mult.mod)
    coverage.res[s, 1] <- alpha[1] > cis[1, 1] & alpha[1] < cis[1, 2]
    coverage.res[s, 2] <- alpha[2] > cis[2, 1] & alpha[2] < cis[2, 2]
    cov.res[,,s] <- mult.mod$var
}

# results
boxplot(p.res[, 1], main = "Posterior medians of p1", ylim = range(c(p.res[, 1], p[1])))
```

```
abline(h = p[1], col = "red")
```




```
boxplot(p.res[, 2], main = "Posterior medians of p2", ylim = range(c(p.res[, 2], p[2])))
abline(h = p[2], col = "red")
```




```
apply(coverage.res, 2, mean)
```

```
## [1] 0.94 0.95
```

```
apply(cov.res, c(1, 2), mean)
```

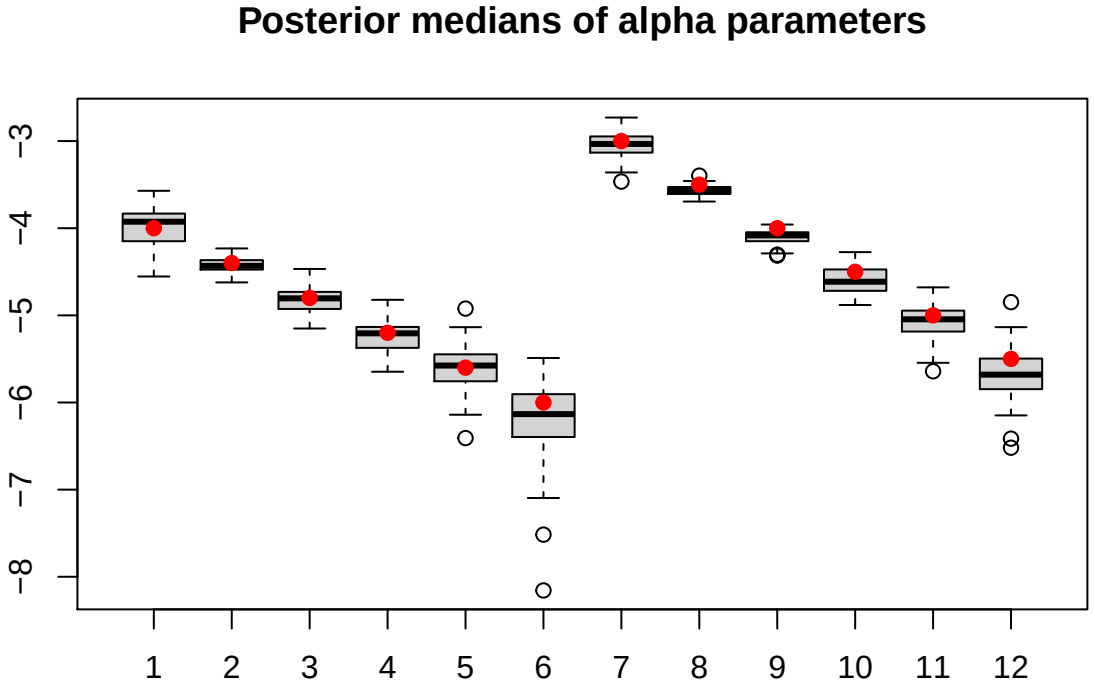
```
##               [,1]         [,2]
## [1,] 3.847064e-03 6.719748e-05
## [2,] 6.719748e-05 1.468054e-03
```

3 categories (1 cause, all other causes, and alive), age-group-specific intercepts

We will again simulate n individuals over 60 months, where in each month for all alive individuals, we apply the multinomial probabilities of dying from cause 1, dying from all other causes, or remaining alive. This time, those probabilities are allowed to vary in 6 different age bands.

```
# sim setup and storage of results
nsim <- 50
alpha.res <- matrix(NA, nrow = nsim, ncol = 12)
coverage.res <- matrix(NA, nrow = nsim, ncol = 12)
cov.res <-array(NA, dim = c(12, 12, nsim))

# parameters
n <- 1000
alpha <- matrix(c(seq(-4, -6, length.out = 6), seq(-3, -5.5, length.out = 6)),
                nrow = 6, ncol = 2)
alpha_vec <- as.vector(alpha)
denom_p <- apply(alpha, 2, function(x){sum(exp(x)) + 1})
p_cause <- cbind(exp(alpha[,1])/denom_p[1], exp(alpha[,2])/denom_p[2])
p <- cbind(p_cause, 1 - apply(p_cause, 1, sum))
p_cum <- apply(p, 1, cumsum)
p_cum_monthly <- p_cum[,c(1, rep(2,11), rep(c(3,4,5,6), each = 12))]

# simulate data
for (s in 1:nsim) {
    # simulate data
    y <- vector(mode = "list", length = n)
    for(i in 1:n) {
        y[[i]] <- matrix(NA, ncol = 6, nrow = 60)
        y[[i]][, 4] <- i # person id
        y[[i]][, 5] <- 1:60 # month indicator
        y[[i]][, 6] <- c(1, rep(2,11), rep(c(3,4,5,6), each = 12)) # age group indicator
        # cat(paste("\nStarting person", i, "\n"))
        t <- 1
        death <- 0
        while(death == 0) {
            # if (t %% 10 == 0) cat(paste("\t month", t, "\n"))
            rand <- runif(1, 0, 1)
            if (rand < p_cum_monthly[1, t]) {
                y[[i]][t, 1:3] <- c(1, 0, 0)
                death <- 1
            } else if (rand >= p_cum_monthly[1, t] & rand < p_cum_monthly[2, t]) {
                y[[i]][t, 1:3] <- c(0, 1, 0)
                death <- 1
            } else {
                y[[i]][t, 1:3] <- c(0, 0, 1)
            }
            t <- t + 1
            if (t == 60) death <- 1
        }
    }

    ymat <- do.call(rbind, y)
    ymat <- ymat[complete.cases(ymat), ]
```

```
    # create data frame
    dat <- data.frame(id = ymat[, 4],
                      time = ymat[, 5],
                      cause1 = ymat[, 1],
                      cause_other = ymat[, 2],
                      alive = ymat[, 3],
                      agegp = ymat[, 6])

    # fit model
    my.svydesign <- survey::svydesign(ids = ~ id,
                                      data = dat)

    mult.mod <- svy_vglm(cbind(cause1, cause_other, alive) ~ -1 + factor(agegp),
                         family = multinomial,
                         design = my.svydesign)
    alpha.res[s,] <- coef(mult.mod)[c(1,3,5,7,9,11,2,4,6,8,10,12)]
    cis <- confint(mult.mod)[c(1,3,5,7,9,11,2,4,6,8,10,12),]
    coverage.res[s, ] <- alpha_vec > cis[,1] & alpha_vec < cis[,2]
    cov.res[,,s] <- mult.mod$var
}

# results
boxplot(alpha.res, use.cols= TRUE, main = "Posterior medians of alpha parameters")
points(1:12, alpha_vec, col = "red", pch = 19)
```




```
apply(coverage.res, 2, mean)
```

```
##  [1] 0.98 0.98 0.96 0.98 0.92 0.96 0.92 0.80 0.82 0.94 0.90 0.94
```

```
apply(cov.res, c(1, 2), mean)
```

```
##                [,1]          [,2]          [,3]          [,4]          [,5]
```

```
##  [1,]  6.128270e-02  1.068222e-03 -1.648867e-17  5.731235e-18  6.284696e-17
##  [2,]  1.068222e-03  2.362706e-02 -2.471902e-17  3.876574e-18  5.860315e-17
##  [3,] -1.648867e-17 -2.471902e-17  1.039203e-02  1.283834e-04  7.989133e-17
##  [4,]  5.731235e-18  3.876574e-18  1.283834e-04  4.464969e-03  7.363577e-17
##  [5,]  6.284696e-17  5.860315e-17  7.989133e-17  7.363577e-17  2.032375e-02
##  [6,]  7.114686e-18  9.076285e-19 -1.651324e-18  4.118937e-18  1.525898e-04
##  [7,]  7.098981e-16  7.047957e-16  8.837653e-16  8.863612e-16  1.228559e-15
##  [8,]  6.480078e-18  9.062351e-18  1.859802e-17  3.366117e-18  1.691167e-17
##  [9,]  3.210295e-13  3.210239e-13  4.023742e-13  4.023769e-13  5.570113e-13
## [10,] -1.412447e-17 -1.887693e-17 -2.136668e-17 -5.763881e-18 -2.637587e-17
## [11,]  3.751778e-11  3.751777e-11  4.684754e-11  4.684755e-11  6.630291e-11
## [12,]  2.498332e-12  2.498326e-12  3.122357e-12  3.122360e-12  4.376740e-12
##               [,6]          [,7]          [,8]         [,9]         [,10]
##  [1,]  7.114686e-18  7.098981e-16  6.480078e-18 3.210295e-13 -1.412447e-17
##  [2,]  9.076285e-19  7.047957e-16  9.062351e-18 3.210239e-13 -1.887693e-17
##  [3,] -1.651324e-18  8.837653e-16  1.859802e-17 4.023742e-13 -2.136668e-17
##  [4,]  4.118937e-18  8.863612e-16  3.366117e-18 4.023769e-13 -5.763881e-18
##  [5,]  1.525898e-04  1.228559e-15  1.691167e-17 5.570113e-13 -2.637587e-17
##  [6,]  9.903672e-03  1.228419e-15  1.658390e-17 5.570113e-13 -2.638235e-17
##  [7,]  1.228419e-15  3.907511e-02  2.260241e-04 7.140834e-13 -3.303927e-17
##  [8,]  1.658390e-17  2.260241e-04  2.062189e-02 7.140836e-13 -3.325768e-17
##  [9,]  5.570113e-13  7.140834e-13  7.140836e-13 6.709416e-02  2.627960e-04
## [10,] -2.638235e-17 -3.303927e-17 -3.325768e-17 2.627960e-04  3.915827e-02
## [11,]  6.630291e-11  8.375580e-11  8.375580e-11 9.826319e-11  9.826320e-11
## [12,]  4.376740e-12  5.626661e-12  5.626660e-12 6.684539e-12  6.684540e-12
##              [,11]        [,12]
##  [1,] 3.751778e-11 2.498332e-12
##  [2,] 3.751777e-11 2.498326e-12
##  [3,] 4.684754e-11 3.122357e-12
##  [4,] 4.684755e-11 3.122360e-12
##  [5,] 6.630291e-11 4.376740e-12
##  [6,] 6.630291e-11 4.376740e-12
##  [7,] 8.375580e-11 5.626661e-12
##  [8,] 8.375580e-11 5.626660e-12
##  [9,] 9.826319e-11 6.684539e-12
## [10,] 9.826320e-11 6.684540e-12
## [11,] 1.660309e-01 2.425947e-04
## [12,] 2.425947e-04 8.657618e-02
```

Simulated data

First, we will simulate a population of births in enumeration areas (point locations) using an inhomogeneous Poisson process with rate proportional to population density in Kenya (using data from IPUMS).

TODO extract this same info for BGD.

EAs and populations are dispersed conditional on the "known number" of EAs, households, and target population at the admin 1 level using multilevel multinomial sampling, first sampling the EAs, then distributing households among the EAs, then the target population among the households. Any areal level below the 'areas' we call 'subareas'. For instance, the 'areas' might be Admin1 if that is the smallest level at which the number of EAs, households, and people is known, and the 'subareas' might be Admin2. The multilevel multinomial sampling may be stratified by urban/rural within the areas if the number of EAs, households, and people is also approximately known at that level.

```
# # simulate population similar to kenya
# data(kenyaMaps)
# data(kenyaPopulationData)
# simPop = simPopSPDE(nsim=1, easpa=easpaKenyaNeonatal,
#                     popMat=popMatKenya, targetPopMat=popMatKenyaNeonatal,
#                     poppsub=poppsubKenya, spdeMesh=kenyaMesh,
#                     margVar=1, sigmaEpsilon=1,
#                     gamma=1, effRange=1, beta0=1,
#                     seed=123, inla.seed=12, nHHSampled=25,
#                     stratifyByUrban=TRUE, subareaLevel=TRUE,
#                     doFineScaleRisk=TRUE,
#                     min1PerSubarea=TRUE)
# eadat <- simPop$pixelPop$eaDatList[[1]]
# eadatlong <- as.data.frame(lapply(eadat, rep, eadat$N))
# # quilt.plot(eadat$lon, eadat$lat, eadat$N)
#
# # expand a row for each birth
# eadatlong <- eadatlong[, c("lon","lat","area", "subarea","urban","east","north")]
#
# # Sample (SRS) births
# prob_samp <- 0.001
# ps <- runif(nrow(eadatlong), 0, 1)
# sampdat <- eadatlong[ps < prob_samp,]
```

```
# # sim setup and storage of results
# nsim <- 10
# alpha.res <- matrix(NA, nrow = nsim, ncol = 12)
# coverage.res <- matrix(NA, nrow = nsim, ncol = 12)
# cov.res <-array(NA, dim = c(12, 12, nsim))
#
# # parameters
# n <- nrow(sampdat)
# alpha <- matrix(c(seq(-4, -6, length.out = 6), seq(-3, -5.5, length.out = 6)),
#                 nrow = 6, ncol = 2)
# alpha_vec <- as.vector(alpha)
# nreg <- length(unique(sampdat$area))
# # gamma <- rnorm(nreg, 0, 0.5)
# gamma <- rep(0, nreg)
# names(gamma) <- unique(sampdat$area)
#
# # simulate data
# for (s in 1:nsim) {
#     # simulate data
#     y <- vector(mode = "list", length = n)
#     for(i in 1:n) {
#         y[[i]] <- matrix(NA, ncol = 6, nrow = 60)
#         y[[i]][, 4] <- i # person id
#         y[[i]][, 5] <- 1:60 # month indicator
#         y[[i]][, 6] <- c(1, rep(2,11), rep(c(3,4,5,6), each = 12)) # age group indicator
#         # cat(paste("\nStarting person", i, "\n"))
#         t <- 1
#         death <- 0
#
#         # get prob of death
```

```
#         alpha_temp <- alpha + gamma[as.character(sampdat$area[i])]
#         denom_p <- apply(alpha_temp, 2, function(x){sum(exp(x)) + 1})
#         p_cause <- cbind(exp(alpha_temp[,1])/denom_p[1], exp(alpha_temp[,2])/denom_p[2])
#         p <- cbind(p_cause, 1 - apply(p_cause, 1, sum))
#         p_cum <- apply(p, 1, cumsum)
#         p_cum_monthly <- p_cum[,c(1, rep(2,11), rep(c(3,4,5,6), each = 12))]
#
#         while(death == 0) {
#             # if (t %% 10 == 0) cat(paste("\t month", t, "\n"))
#             rand <- runif(1, 0, 1)
#             if (rand < p_cum_monthly[1, t]) {
#                 y[[i]][t, 1:3] <- c(1, 0, 0)
#                 death <- 1
#             } else if (rand >= p_cum_monthly[1, t] & rand < p_cum_monthly[2, t]) {
#                 y[[i]][t, 1:3] <- c(0, 1, 0)
#                 death <- 1
#             } else {
#                 y[[i]][t, 1:3] <- c(0, 0, 1)
#             }
#             t <- t + 1
#             if (t == 60) death <- 1
#         }
#     }
#
#     ymat <- do.call(rbind, y)
#     ymat <- ymat[complete.cases(ymat), ]
#
#     # create data frame
#     dat <- data.frame(id = ymat[, 4],
#                       time = ymat[, 5],
#                       cause1 = ymat[, 1],
#                       cause_other = ymat[, 2],
#                       alive = ymat[, 3],
#                       agegp = ymat[, 6])
#
#     # fit model
#     my.svydesign <- survey::svydesign(ids = ~ id,
#                                       data = dat)
#
#     mult.mod <- svy_vglm(cbind(cause1, cause_other, alive) ~ -1 + factor(agegp),
#                          family = multinomial,
#                          design = my.svydesign)
#     alpha.res[s,] <- coef(mult.mod)[c(1,3,5,7,9,11,2,4,6,8,10,12)]
#     cis <- confint(mult.mod)[c(1,3,5,7,9,11,2,4,6,8,10,12),]
#     coverage.res[s, ] <- alpha_vec > cis[,1] & alpha_vec < cis[,2]
#     cov.res[,,s] <- mult.mod$var
# }
#
# # results
# boxplot(alpha.res, use.cols= TRUE, main = "Posterior medians of alpha parameters")
# points(1:12, alpha_vec, col = "red", pch = 19)
#
# apply(coverage.res, 2, mean)
```

```
#
# apply(cov.res, c(1, 2), mean)
```