

# Appendix S1

Alison C. Ketz

11/10/2021

## 1 Analysis Approach

This appendix provides R code for implementing the analysis of the age-period survival models using data from the Columbian sharp-tailed grouse case study. The model fitting procedure for a single model (CSL\_K, an additive constrained generalized additive model with natural splines specified with the Bayesian LASSO for age effects and the kernel convolution model for the period effects) is described in detail. The other models have similarly conducted analysis structure so the step-by-step break down of running each model will be analogous despite the somewhat differing model structure, and consequently, somewhat different R code. We have provided code for each of the 10 models that were fit using NIMBLE, where each model has five R scripts including: 1) A script to execute and run all the additional Rscripts as well as for providing necessary libraries. 2) A script to load the data and compute constants needed to run the analysis. 3) A script that computes the necessary basis functions, and calculates any other constants needed for model fitting. 4) A script for running the models in nimble with additional nimble functions used for intermediary calculations. 5) An R script for doing model checking, calculations of posterior distribution summary statistics, and plotting of results. These five R scripts for each model are provided for all 10 models.

### 1.1 Mother script to run all scripts

The first script in each analysis is used to run all of the other scripts. It is named S1\_01\_grouse\_CSL\_K.R, where each model signifier will change according to the initials described in the main body of the manuscript. Thus, each mother scripts will be specified using S1\_01\_grouse\_X\_Y.R, where X represents the age effects specification and Y represents period effects specification. First, clear the memory of R and set the working directory. Then, load the libraries needed in the analysis. Sourcing each of the subsequent R scripts will run the complete analysis.

```
#clear memory
rm(list = ls())

#setwd
setwd("../grouse_CSL_K")

#load libraries
library(nimble)
library(Matrix)
```

```

library(coda)
library(lattice)
library(splines)
library(Hmisc)
library(lubridate)
library(ggplot2)
library(gridExtra)
library(xtable)
library(RColorBrewer)

#####
### Load data
#####

source("S1_02_load_format_data.R")

#####
### Setting constants
#####

source("S1_03_prelim_CSL_K.R")

#####
### Run model
#####

source("S1_04_run_model_CSL_K.R")

#####
### Post processing: summaries, plots
#####

source("S1_05_results_plots_sum_CSL_K.R")

```

## 1.2 Loading and Formatting Data

The R script S1\_02\_load\_format\_data.R is used to load the data and calculate constants. First, we load the data, and set the number of records in the data frame. The data are already formatted for model fitting. Then we calculate the vector age2date, which is used to offset index differences between the period effects and age effects. The number of age effects are set as a constant. The number of period effects for each year are calculated from the data, and the total possible number of period effects are also calculated.

```

d_fit <- read.csv("../grouse_data.csv")
n_fit <- dim(d_fit)[1]

```

*#vector to calibrate indexes*

```

age2date <- d_fit$left.time - d_fit$left.age

### Number of Age effects
nT_age <- max(d_fit$right.age) - 1

#Number of Period effects for each
#year seperately and combined
nT_yr1 <- length(as.numeric(names(table(d_fit[d_fit[,5]==1,1]))[1])):as.numeric(names(table(d_fit[d_fit[,5]==1,1]))[1]))
nT_yr2 <- length(as.numeric(names(table(d_fit[d_fit[,6]==1,1]))[1])):as.numeric(names(table(d_fit[d_fit[,6]==1,1]))[1]))
nT_yr3 <- length(as.numeric(names(table(d_fit[d_fit[,7]==1,1]))[1])):as.numeric(names(table(d_fit[d_fit[,7]==1,1]))[1]))
nT_period_total <- nT_yr1+nT_yr2+nT_yr3
nT_period_max <- max(nT_yr1, nT_yr2, nT_yr3)
nT_period <- max(d_fit$right.time)

```

### 1.3 Calculating Basis Functions

The R script S1\_03\_prelim\_CSL\_K.R is used to calculate the basis functions used in the analysis. First, we specify and compile a Nimble function to calculate the kernel convolution. This function takes seven arguments. The maximum number of effects ( $nT$ ), the basis function, or distance matrix ( $Z$ ), the square root of the precision parameter of the kernel convolution ( $stauk$ ), a constant ( $nconst = 1/\sqrt{2 * \pi}$ ), the precision of the kernel ( $tauk$ ), the number of knots which also equals the number of columns in the distance matrix ( $nknots$ ), and the white noise random effects used to convolve the kernel density ( $alphau$ ). The normal kernel density is calculated for each effect and knot in `temp1`. These are normalized by summing the `temp1` values and dividing by the sum of the `temp1` values for each effect. Then the normalized kernel density is convolved with the white noise process in `temp`. These are mean centered to ultimately calculate the kernel convolution process ( $KA$ ), which is then returned.

```

kernel_conv <- nimbleFunction(
  run = function(nT = double(0),
                 Z = double(2),
                 stauk = double(0),
                 nconst = double(0),
                 tauk = double(0),
                 nknots = double(0),
                 alphau = double(1)
  ){
    temp <- nimMatrix(value = 0, nrow = nT, ncol = nknots)
    temp1 <- nimMatrix(value = 0, nrow = nT, ncol = nknots)
    temp2 <- nimNumeric(nknots)
    KA <- nimNumeric(nT)

    for (i in 1:nT) {
      for (j in 1:nknots) {
        temp1[i, j] <- stauk * nconst * exp(-0.5 * Z[i, j]^2 * tauk)
      }
    }
  }

```

```

for (j in 1:nknots) {
  temp2[j] <- sum(temp1[1:nT, j])
}

for (i in 1:nT) {
  for (j in 1:nknots) {
    temp[i, j] <- (temp1[i, j] / temp2[j]) * alphau[j]
  }
  KA[i] <- sum(temp[i, 1:nknots])
}
muKA <- mean(KA[1:nT])
KA[1:nT] <- KA[1:nT] - muKA

returnType(double(1))
return(KA[1:nT])
})

```

```
Ckernel_conv <- compileNimble(kernel_conv)
```

Next, we specify the function used for the constrained generalized additive model basis calculation. For the grouse data, we use a function that calculates a basis that assumes a shape-constrained function to be decending an convex (deconvex). See Meyer et al (2008) for details.

```

deconvex=function(x,t){
  n=length(x)
  k=length(t)-2
  m=k+3
  sigma=matrix(1:(m*n)*0,nrow=m,ncol=n)
  for(j in 1:k){
    i1=x<=t[j]
    sigma[j,i1] = x[i1]-t[1]
    i2=x>t[j]&x<=t[j+1]
    sigma[j,i2] = t[j]-t[1]+((t[j+1]-t[j])^3-(t[j+1]-x[i2])^3)/3/(t[j+1]-t[j])/(t[j+2]-t[j]) +
    i3=x>t[j+1]&x<=t[j+2]
    sigma[j,i3] = t[j]-t[1] + (t[j+1]-t[j])^2/3/(t[j+2]-t[j]) + (t[j+2]-t[j+1])*(t[j+1]-t[j])/
    i4=x>=t[j+2]
    sigma[j,i4] = t[j]-t[1] + (t[j+1]-t[j])^2/3/(t[j+2]-t[j]) + (t[j+2]-t[j+1])*(t[j+1]-t[j])/
  }
  i1=x<=t[2]
  sigma[k+1,i1]=-(t[2]-x[i1])^3/3/(t[2]-t[1])^2
  i2=x>t[2]
  sigma[k+1,i2]=0
  i1=x<=t[k+1]
  sigma[k+2,i1]=x[i1]-t[1]
  i2=x>t[k+1]&x<=t[k+2]
  sigma[k+2,i2]=t[k+1]-t[1]+((t[k+2]-t[k+1])^2*(x[i2]-t[k+1])-(x[i2]-t[k+1])^3/3)/(t[k+2]-t[k+1])
  i3=x>t[k+2]

```

```

sigma[k+2,i3]=t[k+1]-t[1]+((t[k+2]-t[k+1])^2*(t[k+2]-t[k+1])-(t[k+2]-t[k+1])^3/3)/(t[k+2]-t[k+1])
sigma[k+3,]=x

center.vector=apply(sigma,1,mean)

list(sigma=-sigma, center.vector=-center.vector)
}

```

Then we calculate the basis function using the above deconvex function specified above. We specify the CGAM basis for the age effects. We set knots based on quantiles of the failure or censoring intervals, with a goal of 6 knots total. We calculate the basis (delta\_i), then mean center it (delta). Then we rescale this basis by dividing by the maximum value of the basis (delta) to aid convergence. We set a variable with the number of knots (nknots\_age\_cgam).

```

quant_age <- .2
knots_age <- c(1, round(quantile(d_fit$right.age - 1,
                                c(seq(quant_age,
                                      .99,
                                      by = quant_age),
                                      .99))))))
knots_age_cgam <- unique(knots_age)
delta_i <- deconvex(1:nT_age, knots_age_cgam)
delta <- t(delta_i$sigma - delta_i$center.vector)
Z_age_cgam <- delta / max(delta)
nknots_age_cgam <- dim(Z_age_cgam)[2]

```

Then we calculate the basis function for the age effects using natural splines, using the spline package function (ns). Again, we use quantile knots based on the mortality and censoring interval frequencies, and we use an equal quantile interval that allows for the number of knots to be close to 20. We calculate a constraint matrix (constr\_sumzero) so that the age effects will sum to zero. We use QR factorization (qrc) to find the null space of the constraint matrix (set to Z). Then we calculate the basis function provided this constraint matrix (Z\_age\_spline). We calculate the number of knots (nknots\_age\_spline).

```

quant_age <- .04
knots_age <- c(1,
              round(quantile(d_fit$right.age - 1,
                              c(seq(quant_age,
                                    .99,
                                    by = quant_age),
                                    .99))))))
knots_age_spline <- unique(knots_age)

splinebasis <- ns(1:nT_age, knots = knots_age_spline)

##A constraint matrix so age.effects = 0
constr_sumzero <- matrix(1, 1, nrow(splinebasis)) %*% splinebasis

##Get a basis for null space of constraint

```

```

qrc <- qr(t(constr_sumzero))
Z <- qr.Q(qrc,
          complete = TRUE)[, (nrow(constr_sumzero) + 1):ncol(constr_sumzero)]
Z_age_spline <- splinebasis %*% Z
nknots_age_spline <- dim(Z_age_spline)[2]

```

We plot the basis functions.

```

pdf("figures/basis_function_age.pdf")
plot(1:nT_age,
     Z_age_spline[, 1],
     ylim = c(-1, 1),
     type = "l",
     main = "Basis Function Age Effect Spline")
for (i in 2:nknots_age_spline) {
  lines(1:nT_age, Z_age_spline[, i])
}
plot(1:nT_age,
     Z_age_cgam[, 1],
     ylim = c(-1, 1),
     type = "l",
     main = "Basis Function Age Effect CGAM")
for (i in 2:nknots_age_cgam) {
  lines(1:nT_age, Z_age_cgam[, i])
}
dev.off()

```

Then, we calculate the distance matrix, or basis functions, for the kernel convolution smoother for the period effects for each year separately. We used the same interval for each year with a knot at each period (day). We calculate the absolute distance between each period for each knot.

```

#####
###
### Year 1 = 2015
###
#####

intvl_period_yr1 <- intvl_period # interval for knots
knots_period_yr1 <- unique(c(seq(1,
                                nT_yr1,
                                intvl_period_yr1),
                                nT_yr1)) #knots
nknots_period_yr1 <- length(knots_period_yr1)

Z_period_yr1 <- matrix(0,
                      nT_yr1,
                      nknots_period_yr1)
for (i in 1:nrow(Z_period_yr1)) {
  for (j in 1:nknots_period_yr1) {

```

```

    Z_period_yr1[i, j] <- abs(i - knots_period_yr1[j])
  }
}

#####
###
### Year 2 = 2016
###
#####

intvl_period_yr2 <- intvl_period # interval for knots

knots_period_yr2 <- unique(c(seq(1,
                                nT_yr2,
                                intvl_period_yr2),
                                nT_yr2)) #knots
nknots_period_yr2 <- length(knots_period_yr2)
Z_period_yr2 <- matrix(0,
                      nT_yr2,
                      nknots_period_yr2)
for (i in 1:nrow(Z_period_yr2)) {
  for (j in 1:nknots_period_yr2) {
    Z_period_yr2[i, j] <- abs(i - knots_period_yr2[j]) #absolute value
  }
}

#####
###
### Year 3 = 2017
###
#####

intvl_period_yr3 <- intvl_period # interval for knots
knots_period_yr3 <- unique(c(seq(1,
                                nT_yr3,
                                intvl_period_yr3),
                                nT_yr3)) #knots
nknots_period_yr3 <- length(knots_period_yr3)

Z_period_yr3 <- matrix(0, nT_yr3, nknots_period_yr3)
for (i in 1:nrow(Z_period_yr3)) {
  for (j in 1:nknots_period_yr3) {
    Z_period_yr3[i, j] <- abs(i - knots_period_yr3[j])
  }
}

```

We set the number of MCMC iterations (reps), the burn-in period (bin), the number of MCMC chains (n\_chains), and the thinning interval (n\_thin).

```

reps <- 50000
bin <- reps * .5
n_chains <- 3
n_thin <- 1

```

## 1.4 Run the model using the NIMBLE package

The following R script `S1_04_run_model_CSL_K.R` is used to run the model using the NIMBLE package. First, we provide a function that can calculate the probability of a mortality for each individual across the intervals (age and period) that the individual is collared and alive in the study. The function is based on indexing over the age effects (`age_effect`), and uses the `age2date` vector to calculate the proper index for the period intervals (`period_effect`). The intercept (`beta0`) is added to the log hazard. The summation from left to right provides the approximation of the integration of the log hazard for obtaining the cumulative probability of survival during each age interval and each period interval. The number of rows in the data (`records`), describes where individuals that die during the study have 2 rows and individuals that are right censored have a single row. Additional arguments include the age of entry (`left`), and age of exiting the study (`right`), and the total number of age effects (`nT_age`).

```

state_transition <- nimbleFunction(
  run = function(records = double(0),
                 left = double(1),
                 right = double(1),
                 beta0 = double(0)
                 age_effect = double(1),
                 period_effect = double(1),
                 age2date = double(1),
                 nT_age = double(0)
  ){

    SLR <- nimNumeric(records)
    UCH <- nimMatrix(value = 0, nrow = records, ncol = nT_age)
    for (j in 1:records) {
      for (k in left[j]:(right[j] - 1)) {
        UCH[j, k] <- exp(beta0 +
                        age_effect[k] +
                        period_effect[k - age2date[j]])
      }
      SLR[j] <- exp(-sum(UCH[j, left[j]:(right[j] - 1)]))
    }
    returnType(double(1))
    return(SLR[1:records])
  })

cstate_transition <- compileNimble(state_transition)

```

Then we specify the model statement using the NIMBLE syntax, which is based on declarative BUGS



language. We specify the prior distribution for the intercept ( $\beta_0$ ) using parameter expansion.

```
beta0_temp ~ dnorm(0, .01)
mix ~ dunif(-1, 1)
beta0 <- beta0_temp * mix
```

Then we specify the prior distribution for the effects for the cgam model, and calculate the age effects. We centered the age effects for the cgam model by subtracting the mean of the age effects and then used the sum-to-zero age effects. We could not use QR factorization to obtain the null space of the basis function when calculating the basis function, because this mapping of the basis functions prevented the non-negative constraint for the coefficients to ensure the concave shape.

```
for (k in 1:nknots_age_cgam) {
  ln_b_age_cgam[k] ~ dnorm(0, tau_age_cgam)
  b_age_cgam[k] <- exp(ln_b_age_cgam[k])
}
tau_age_cgam ~ dgamma(1, 1)

for (t in 1:nT_age) {
  age_effect_temp[t] <- inprod(b_age_cgam[1:nknots_age_cgam],
                               Z_age_cgam[t, 1:nknots_age_cgam])
  age_effect_cgam[t] <- age_effect_temp[t] - mu_age_cgam
}
mu_age_cgam <- mean(age_effect_temp[1:nT_age])
```

The prior for the spline model was specified using the double exponential, or Laplace prior rather than a normal prior distribution. The QR factorization that was implemented on the basis function ensures these effects are summing to zero. We combined the cgam and spline models by adding them.

```
for (k in 1:nknots_age_spline) {
  b_age_spline[k] ~ ddexp(0, tau_age_spline)
}
tau_age_spline ~ dgamma(.1, .1)
for (t in 1:nT_age) {
  age_effect_spline[t] <- inprod(b_age_spline[1:nknots_age_spline],
                                Z_age_spline[t, 1:nknots_age_spline])
}
#####
### Age effect combine
#####
age_effect[1:nT_age] <- age_effect_cgam[1:nT_age] +
  age_effect_spline[1:nT_age]
```

We calculate the period effects using the kernel convolution function provided earlier (`period_effect`). We used a shared specification for the kernel smoother for the three years. We specify the prior for the kernel smoother on the log scale ( $\ln\_sk\_period$ ), along with a parameter expansion mixing parameter to aid in mixing ( $mix2$ ). We generate the Gaussian white noise process ( $\alpha_{hau\_period}$ ) by generating the standard normal ( $\alpha_{period}$ ) and then multiply this by the standard deviation ( $sda\_period$ ) that has a truncated normal prior.

```

mix2 ~ dunif(-1, 1)
ln_sk_period ~ dnorm(0, sd = 1)
sdk_period <- exp(mix2 * ln_sk_period)
tauk_period <- 1 / sdk_period^2
stauk_period <- sqrt(tauk_period)
sda_period ~ T(dnorm(0, sd = 1), 0, Inf)
taua_period <- 1 / sda_period^2
ratioinf_period <- sdk_period / sda_period

###
### Period effects for year 1
###

for (i in 1:(nknots_period_yr1)) {
  alpha_period_yr1[i] ~ dnorm(0, 1)
  alphau_period_yr1[i] <- sda_period * alpha_period_yr1[i]
}
period_effect_yr1[1:nT_yr1] <- kernel_conv(
  nT = nT_yr1,
  Z = Z_period_yr1[1:nT_yr1, 1:nknots_period_yr1],
  stauk = stauk_period,
  nconst = nconst,
  tauk = tauk_period,
  nknots = nknots_period_yr1,
  alphau = alphau_period_yr1[1:nknots_period_yr1]
)

###
### Period effects for year 2
###

for (i in 1:(nknots_period_yr2)) {
  alpha_period_yr2[i] ~ dnorm(0, 1)
  alphau_period_yr2[i] <- sda_period * alpha_period_yr2[i]
}

period_effect_yr2[1:nT_yr2] <- kernel_conv(
  nT = nT_yr2,
  Z = Z_period_yr2[1:nT_yr2, 1:nknots_period_yr2],
  stauk = stauk_period,
  nconst = nconst,
  tauk = tauk_period,
  nknots = nknots_period_yr2,
  alphau = alphau_period_yr2[1:nknots_period_yr2]
)

###

```

```

### Period effects for year 3
###

for (i in 1:(nknots_period_yr3)) {
  alpha_period_yr3[i] ~ dnorm(0, 1)
  alphau_period_yr3[i] <- sda_period * alpha_period_yr3[i]
}

period_effect_yr3[1:nT_yr3] <- kernel_conv(
  nT = nT_yr3,
  Z = Z_period_yr3[1:nT_yr3, 1:nknots_period_yr3],
  stauk = stauk_period,
  nconst = nconst,
  tauk = tauk_period,
  nknots = nknots_period_yr3,
  alphau = alphau_period_yr3[1:nknots_period_yr3]
)

###
### Combine period effects into a single vector
###

period_effect[1:nT_yr1] <- period_effect_yr1[1:nT_yr1]
period_effect[(nT_yr1 + 1):(nT_yr1 + nT_yr2)] <- period_effect_yr2[1:nT_yr2]
period_effect[(nT_yr1 + nT_yr2 + 1):(nT_period_total)] <- period_effect_yr3[1:nT_yr3]

```

Then we compute the cumulative probability of mortality within each period using the `state_transition` function. The likelihood is just a Bernoulli distribution for each individual given whether they are alive and in the study (`sensor=1`), or whether they have died (`sensor = 0`).

```

SLR[1:records] <- state_transition(records = records,
  left = left_age[1:records],
  right = right_age[1:records],
  nT_age = nT_age,
  age_effect = age_effect[1:nT_age],
  period_effect = period_effect[1:nT_period_total],
  age2date = age2date[1:records],
  beta0 = beta0)

#####
### Likelihood
#####
for (j in 1:records) {
  censor[j] ~ dbern(SLR[j])
}

```

We calculate derived parameters consisting of the cumulative probability of survival over the age intervals and period intervals separately. We also calculated combined age+period survival by

adding the age effects with the period effects over specific intervals. A full survival surface could also be calculated.

```
#####
### Derived parameters
#####

for (t in 1:nT_age) {
  llambda_age[t] <- beta0 + age_effect[t]
  UCHO_age[t] <- exp(llambda_age[t])
  SO_age[t] <- exp(-sum(UCHO_age[1:t]))
}

for (t in 1:nT_yr1) {
  llambda_period[t, 1] <- beta0 + period_effect_yr1[t]
  UCHO_period[t, 1] <- exp(llambda_period[t, 1])
  SO_period[t, 1] <- exp(-sum(UCHO_period[1:t, 1]))
}
for (t in 1:(nT_yr2)) {
  llambda_period[t, 2] <- beta0 + period_effect_yr2[t]
  UCHO_period[t, 2] <- exp(llambda_period[t, 2])
  SO_period[t, 2] <- exp(-sum(UCHO_period[1:t, 2]))
}
for (t in 1:(nT_yr3)) {
  llambda_period[t, 3] <- beta0 + period_effect_yr3[t]
  UCHO_period[t, 3] <- exp(llambda_period[t, 3])
  SO_period[t, 3] <- exp(-sum(UCHO_period[1:t, 3]))
}
#combining age+period
for (t in 1:nT_age) {#nT_age<nT_yr1
  llambda_ageperiod[t, 1] <- beta0 + period_effect_yr1[t] + age_effect[t]
  UCHO_ageperiod[t, 1] <- exp(llambda_ageperiod[t, 1])
  SO_ageperiod[t, 1] <- exp(-sum(UCHO_ageperiod[1:t, 1]))
}
for (t in 1:(nT_yr2)) {
  llambda_ageperiod[t, 2] <- beta0 + period_effect_yr2[t] + age_effect[t]
  UCHO_ageperiod[t, 2] <- exp(llambda_ageperiod[t, 2])
  SO_ageperiod[t, 2] <- exp(-sum(UCHO_ageperiod[1:t, 2]))
}
for (t in 1:(nT_yr3)) {
  llambda_ageperiod[t, 3] <- beta0 + period_effect_yr3[t] + age_effect[t]
  UCHO_ageperiod[t, 3] <- exp(llambda_ageperiod[t, 3])
  SO_ageperiod[t, 3] <- exp(-sum(UCHO_ageperiod[1:t, 3]))
}
}
```

We put this altogether in the modelcode statement.

```
modelcode <- nimbleCode({
```

*#Priors for Age and Period effects*

```
beta0_temp ~ dnorm(0, .01)
mix ~ dunif(-1, 1)
beta0 <- beta0_temp * mix
```

```
#####
### Age effect cgam
#####
```

```
for (k in 1:nknots_age_cgam) {
  ln_b_age_cgam[k] ~ dnorm(0, tau_age_cgam)
  b_age_cgam[k] <- exp(ln_b_age_cgam[k])
}
tau_age_cgam ~ dgamma(1, 1)
```

```
for (t in 1:nT_age) {
  age_effect_temp[t] <- inprod(b_age_cgam[1:nknots_age_cgam],
                                Z_age_cgam[t, 1:nknots_age_cgam])
  age_effect_cgam[t] <- age_effect_temp[t] - mu_age_cgam
}
mu_age_cgam <- mean(age_effect_temp[1:nT_age])
```

```
#####
### Age effect spline
#####
```

```
for (k in 1:nknots_age_spline) {
  b_age_spline[k] ~ ddexp(0, tau_age_spline)
}
tau_age_spline ~ dgamma(.1, .1)
for (t in 1:nT_age) {
  age_effect_spline[t] <- inprod(b_age_spline[1:nknots_age_spline],
                                Z_age_spline[t, 1:nknots_age_spline])
}
```

```
#####
### Age effect combine
#####
age_effect[1:nT_age] <- age_effect_cgam[1:nT_age] +
  age_effect_spline[1:nT_age]
```

```
#####
### Period effects
### for entire study (3 years)
### with separate period effects for each year,
### but shared hyperparams
#####
```

```

mix2 ~ dunif(-1, 1)
ln_sk_period ~ dnorm(0, sd = 1)
sdk_period <- exp(mix2 * ln_sk_period)
tauk_period <- 1 / sdk_period^2
stauk_period <- sqrt(tauk_period)
sda_period ~ T(dnorm(0, sd = 1), 0, Inf)
taua_period <- 1 / sda_period^2
ratioinf_period <- sdk_period / sda_period

###
### Period effects for year 1
###

for (i in 1:(nknots_period_yr1)) {
  alpha_period_yr1[i] ~ dnorm(0, 1)
  alphau_period_yr1[i] <- sda_period * alpha_period_yr1[i]
}
period_effect_yr1[1:nT_yr1] <- kernel_conv(
  nT = nT_yr1,
  Z = Z_period_yr1[1:nT_yr1, 1:nknots_period_yr1],
  stauk = stauk_period,
  nconst = nconst,
  tauk = tauk_period,
  nknots = nknots_period_yr1,
  alphau = alphau_period_yr1[1:nknots_period_yr1]
)

###
### Period effects for year 2
###

for (i in 1:(nknots_period_yr2)) {
  alpha_period_yr2[i] ~ dnorm(0, 1)
  alphau_period_yr2[i] <- sda_period * alpha_period_yr2[i]
}

period_effect_yr2[1:nT_yr2] <- kernel_conv(
  nT = nT_yr2,
  Z = Z_period_yr2[1:nT_yr2, 1:nknots_period_yr2],
  stauk = stauk_period,
  nconst = nconst,
  tauk = tauk_period,
  nknots = nknots_period_yr2,
  alphau = alphau_period_yr2[1:nknots_period_yr2]
)

###

```

```

### Period effects for year 3
###

for (i in 1:(nknots_period_yr3)) {
  alpha_period_yr3[i] ~ dnorm(0, 1)
  alphau_period_yr3[i] <- sda_period * alpha_period_yr3[i]
}

period_effect_yr3[1:nT_yr3] <- kernel_conv(
  nT = nT_yr3,
  Z = Z_period_yr3[1:nT_yr3, 1:nknots_period_yr3],
  stauk = stauk_period,
  nconst = nconst,
  tauk = tauk_period,
  nknots = nknots_period_yr3,
  alphau = alphau_period_yr3[1:nknots_period_yr3]
)

###
### Combine period effects into a single vector
###

period_effect[1:nT_yr1] <- period_effect_yr1[1:nT_yr1]
period_effect[(nT_yr1 + 1):(nT_yr1 + nT_yr2)] <- period_effect_yr2[1:nT_yr2]
period_effect[(nT_yr1 + nT_yr2 + 1):(nT_period_total)] <- period_effect_yr3[1:nT_yr3]

### Computing state transisiton probability
SLR[1:records] <- state_transition(records = records,
  left = left_age[1:records],
  right = right_age[1:records],
  nT_age = nT_age,
  age_effect = age_effect[1:nT_age],
  period_effect = period_effect[1:nT_period_total],
  age2date = age2date[1:records],
  beta0 = beta0)

#####
### Likelihood
#####
for (j in 1:records) {
  censor[j] ~ dbern(SLR[j])
}

#####
### Derived parameters
#####

for (t in 1:nT_age) {

```

```

    llambda_age[t] <- beta0 + age_effect[t]
    UCHO_age[t] <- exp(llambda_age[t])
    SO_age[t] <- exp(-sum(UCHO_age[1:t]))
  }

  for (t in 1:nT_yr1) {
    llambda_period[t, 1] <- beta0 + period_effect_yr1[t]
    UCHO_period[t, 1] <- exp(llambda_period[t, 1])
    SO_period[t, 1] <- exp(-sum(UCHO_period[1:t, 1]))
  }
  for (t in 1:(nT_yr2)) {
    llambda_period[t, 2] <- beta0 + period_effect_yr2[t]
    UCHO_period[t, 2] <- exp(llambda_period[t, 2])
    SO_period[t, 2] <- exp(-sum(UCHO_period[1:t, 2]))
  }
  for (t in 1:(nT_yr3)) {
    llambda_period[t, 3] <- beta0 + period_effect_yr3[t]
    UCHO_period[t, 3] <- exp(llambda_period[t, 3])
    SO_period[t, 3] <- exp(-sum(UCHO_period[1:t, 3]))
  }
  #combining age+period
  for (t in 1:nT_age) {#nT_age<nT_yr1
    llambda_ageperiod[t, 1] <- beta0 + period_effect_yr1[t] + age_effect[t]
    UCHO_ageperiod[t, 1] <- exp(llambda_ageperiod[t, 1])
    SO_ageperiod[t, 1] <- exp(-sum(UCHO_ageperiod[1:t, 1]))
  }
  for (t in 1:(nT_yr2)) {
    llambda_ageperiod[t, 2] <- beta0 + period_effect_yr2[t] + age_effect[t]
    UCHO_ageperiod[t, 2] <- exp(llambda_ageperiod[t, 2])
    SO_ageperiod[t, 2] <- exp(-sum(UCHO_ageperiod[1:t, 2]))
  }
  for (t in 1:(nT_yr3)) {
    llambda_ageperiod[t, 3] <- beta0 + period_effect_yr3[t] + age_effect[t]
    UCHO_ageperiod[t, 3] <- exp(llambda_ageperiod[t, 3])
    SO_ageperiod[t, 3] <- exp(-sum(UCHO_ageperiod[1:t, 3]))
  }
} #end model statement

```

We specify data, constants and initial values for all of the parameters in the model.

```

#Data
nimData <- list(censor = d_fit$censor,
               Z_period_yr1 = Z_period_yr1,
               Z_period_yr2 = Z_period_yr2,
               Z_period_yr3 = Z_period_yr3,
               Z_age_cgam = Z_age_cgam,
               Z_age_spline = Z_age_spline,

```



```

left_age = d_fit$left.age,
right_age = d_fit$right.age,
age2date = age2date,
llambda_period = matrix(NA, nr = nT_period_max, nc = 3),
UCHO_period = matrix(NA, nr = nT_period_max, nc = 3),
SO_period = matrix(NA, nr = nT_period_max, nc = 3),
llambda_ageperiod = matrix(NA, nr = nT_period_max, nc = 3),
UCHO_ageperiod = matrix(NA, nr = nT_period_max, nc = 3),
SO_ageperiod = matrix(NA, nr = nT_period_max, nc = 3),
period_effect = rep(NA, nT_period_total)
)

nimConsts <- list(records = n_fit,
  nT_age = nT_age,
  nT_period_total = nT_period_total,
  nT_yr1 = nT_yr1,
  nT_yr2 = nT_yr2,
  nT_yr3 = nT_yr3,
  nknots_age_cgam = nknots_age_cgam,
  nknots_age_spline = nknots_age_spline,
  nknots_period_yr1 = nknots_period_yr1,
  nknots_period_yr2 = nknots_period_yr2,
  nknots_period_yr3 = nknots_period_yr3,
  nconst = 1 / sqrt(2 * pi)
)

initsFun <- function()list(beta0_temp = rnorm(1, -5, .0001),
  mix = 1,
  mix2 = 1,
  sda_period = runif(1, 0, 5),
  ln_sk_period = rnorm(1, 0, 1),
  alpha_period_yr1 = rep(0, nknots_period_yr1),
  alpha_period_yr2 = rep(0, nknots_period_yr2),
  alpha_period_yr3 = rep(0, nknots_period_yr3),
  tau_age_cgam = runif(1, .1, 1),
  tau_age_spline = runif(1, .1, 1),
  ln_b_age_cgam = runif(nknots_age_cgam, -10, -5),
  b_age_spline = rnorm(nknots_age_spline) * 10^-4,
  llambda_period = matrix(0,
    nr = nT_period_max,
    nc = 3),
  UCHO_period = matrix(0,
    nr = nT_period_max,
    nc = 3),
  SO_period = matrix(0,
    nr = nT_period_max,
    nc = 3),

```

```

        llambda_ageperiod = matrix(0,
                                   nr = nT_period_max,
                                   nc = 3),
        UCHO_ageperiod = matrix(0,
                                 nr = nT_period_max,
                                 nc = 3),
        S0_ageperiod = matrix(0,
                               nr = nT_period_max,
                               nc = 3),
        period_effect = rep(0, nT_period_total)
    )
nimInits <- initsFun()

```

Then we build the model, MCMC object, and run the MCMC approximation. We save the MCMC iterations.

```

Rmodel <- nimbleModel(code = modelcode,
                     constants = nimConsts,
                     data = nimData,
                     inits = initsFun()
                     )

```

*#identify params to monitor*

```

parameters <- c(
    "beta0",
    "beta0_temp",
    "mix",
    "mix2",
    "sdk_period",
    "sda_period",
    "ratioinf_period",
    "S0_age",
    "llambda_age",
    "age_effect",
    "age_effect_cgam",
    "age_effect_spline",
    "period_effect_yr1",
    "period_effect_yr2",
    "period_effect_yr3",
    "S0_period",
    "llambda_period",
    "tau_age_cgam",
    "tau_age_spline",
    "b_age_cgam",
    "b_age_spline",
    "S0_ageperiod",
    "llambda_ageperiod"
)

```

```

starttime <- Sys.time()
confMCMC <- configureMCMC(Rmodel,
                          monitors = parameters,
                          thin = n_thin,
                          useConjugacy = FALSE,
                          enableWAIC = TRUE)
nimMCMC <- buildMCMC(confMCMC)
Cnim <- compileNimble(Rmodel)
CnimMCMC <- compileNimble(nimMCMC,
                          project = Rmodel)
mcmcout <- runMCMC(CnimMCMC,
                  niter = reps,
                  nburnin = bin,
                  nchains = n_chains,
                  inits = initsFun,
                  samplesAsCodaMCMC = TRUE,
                  summary = TRUE,
                  WAIC = TRUE
                  )

runtime <- difftime(Sys.time(),
                   starttime,
                   units = "min")

save(runtime, file = "results/runtime.Rdata")
save(mcmcout, file = "results/mcmcout.Rdata")

```

All of the code in this section is aggregated and combined in the file S1\_04\_run\_model\_CSL\_K.R. Code for plotting this results of the simulation from a single seed, as well as for obtaining convergence diagnostics are provided in the file S1\_05\_results\_plots\_sum\_CSL\_K.

It is not possible to extensively describe all of the models. However, they were all fit using similar code. In the model described above, several of the basis functions were used, which are then used in isolation for the different age or period effects for the other model specifications.