White-tailed deer Age-Period Survival Analysis R code

Alison C. Ketz

1 Analysis Approach

This appendix provides R code for implementing the analysis of the age-period survival models using data from the white-tailed deer 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 e□ects and the kernel convolution model for the period e□ects) 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 di□ering model structure, and consequently, somewhat di□erent 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.

2 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 e□ects specification and Y represents period e□ects 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(".../wtd_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")
```

3 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 o□set index di□erences between the period e□ects and age e□ects. The total number of age e□ects and total number of period e□ects are set as a constants.

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

#vector to calibrate indexes
age2date <- d_fit$left.time - d_fit$ageCapture
```

```
### Number of Age effects and Number of Period effects
nT_age <- max(d_fit$ageRight) - 1
nT_period <- max(d_fit$right.time) - 1
```

4 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 e□ects (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 e□ects used to convolve the kernel density (alphau). The normal kernel density is calculated for each e□ect and knot in temp1. These are normalized by summing the temp1 values and dividing by the sum of the temp1 values for each e□ect. 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 white-tailed deer data, we use a function that caculates a basis that assumes a shape-constrained function to be convex (convex). See Meyer et al (2008) for details.

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

  v1=1:n*0+1
  v2=x
  x.mat=cbind(v1,v2)
```

```
  if(pred.new==TRUE){
    list(sigma=sigma,x.mat=x.mat)}

  else{
    if(pred.new==FALSE){
      coef=solve(t(x.mat)%*%x.mat)%*%t(x.mat)%*%t(sigma)
      list(sigma=sigma, x.mat=x.mat, center.vector=coef)}
  }

}
```

Then we calculate the basis function using the above decconvex 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_cgam <- c(1, round(quantile(d_fit$ageRight - 1,
                      c(seq(quant_age, .99, by = quant_age),
                      .99))))
knots_age_cgam <- unique(knots_age_cgam)
delta_i <- convex(1:nT_age, knots_age_cgam, pred.new = FALSE)
delta <- t(rbind(delta_i$sigma - t(delta_i$x.mat %*% delta_i$center.vector) )  )
delta <- delta / max(delta)
Z_age_cgam <- 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 intervals, 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 <- .05
knots_age_spline <- c(1,
                  round(quantile(d_fit$ageRight - 1,
                                 c(seq(quant_age, .99, by = quant_age), .99))))
knots_age_spline <- unique(knots_age_spline)

##Basis for age hazard
splinebasis <- ns(1:nT_age, knots = knots_age_spline)

##A constraint matrix so time.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_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])
}

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])
}
dev.off()
```

Then, we calculate the distance matrix, or basis functions, for the kernel convolution smoother for the period e□ects for each year seperately. 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.

```
intvl_period <- 1
knots_period <- c(seq(1,nT_period,by=intvl_period),nT_period)
knots_period <- unique(knots_period)
nknots_period <- length(knots_period)

Z_period <- matrix(0, nT_period, nknots_period)
for (i in 1:nrow(Z_period)) {
  for (j in 1:nknots_period) {
    Z_period[i, j] <- abs(i - knots_period[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
```

5 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 e□ects (age_e□ect), and uses the age2date vector to calculate the proper index for the period intervals (period_e□ect). 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 e□ects (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 (beta0) using parameter expansion.

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

Then we specify the prior distribution for the e□ects for the cgam model, and calculate the age e□ects. We centered the age e□ects for the cgam model by subtracting the mean of the age e□ects and then used the sum-to-zero age e□ects. 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 coe□cients 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 e□ects 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 e□ects using the kernel convolution function provided earlier (period_e□ect). 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 (alphau_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
 for (i in 1:(nknots_period)) {
   alpha_period[i] ~ dnorm(0, 1)
   alphau_period[i] <- sda_period * alpha_period[i]
 }
 ratioinf_period <- sdk_period / sda_period #ratio of variability

 period_effect[1:nT_period] <- kernel_conv(
   nT = nT_period,
   Z = Z_period[1:nT_period, 1:nknots_period],
   stauk = stauk_period,
   nconst = nconst,
   tauk = tauk_period,
   nknots = nknots_period,
   alphau = alphau_period[1:nknots_period]
 )
```

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 (censor=1), or whether they have died (censor = 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],
                                   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 e□ects with the period e□ects over specific intervals. A full survival surface could also be calculated.

```
 for (t in 1:nT_age) {
   llambda_age[t] <- beta0 + age_effect[t]
```

```
    UCH0_age[t] <- exp(llambda_age[t])
    S0_age[t] <- exp(-sum(UCH0_age[1:t]))
  }
  for (t in 1:nT_period) {
    llambda_period[t] <- beta0 + period_effect[t]
    UCH0_period[t] <- exp(llambda_period[t])
    S0_period[t] <- exp(-sum(UCH0_period[1:t]))
  }
```

We put this altogether in the model 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
########################
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
for (i in 1:(nknots_period)) {
  alpha_period[i] ~ dnorm(0, 1)
  alphau_period[i] <- sda_period * alpha_period[i]
}
ratioinf_period <- sdk_period / sda_period #ratio of variability

period_effect[1:nT_period] <- kernel_conv(
  nT = nT_period,
  Z = Z_period[1:nT_period, 1:nknots_period],
  stauk = stauk_period,
  nconst = nconst,
  tauk = tauk_period,
  nknots = nknots_period,
  alphau = alphau_period[1:nknots_period]
)

##########################################
### 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],
                                   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]
    UCH0_age[t] <- exp(llambda_age[t])
    S0_age[t] <- exp(-sum(UCH0_age[1:t]))
  }
  for (t in 1:nT_period) {
    llambda_period[t] <- beta0 + period_effect[t]
    UCH0_period[t] <- exp(llambda_period[t])
    S0_period[t] <- exp(-sum(UCH0_period[1:t]))
  }

})#end model statement
```

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

```
#Data
nimData <- list(censor = d_fit$censor,
                Z_period = Z_period,
                Z_age_cgam = Z_age_cgam,
                Z_age_spline = Z_age_spline,
                left_age = d_fit$ageCapture,
                right_age = d_fit$ageRight,
                age2date = age2date
                )

nimConsts <- list(records = n_fit,
                  nT_age = nT_age,
                  nT_period = nT_period,
                  nknots_age_cgam = nknots_age_cgam,
                  nknots_age_spline = nknots_age_spline,
                  nknots_period = nknots_period,
                  nconst = 1 / sqrt(2 * pi)
)

initsFun <- function()list(
                           tau_age_cgam = runif(1, .1, 1),
                           tau_age_spline = runif(1, .1, 1),
                           beta0_temp = rnorm(1, -5, .0001),
                           mix = 1,
                           mix2 = 1,
```

```
                    sda_period = runif(1, 0, 3),
                    ln_sk_period = rnorm(1, 0, 1),
                    alpha_period = rep(0, nknots_period),
                    b_age_spline = rnorm(nknots_age_spline)
```

```
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 = "runtime.Rdata")
save(mcmcout, file = "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 be 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.