R code description for cross-strain transmission analysis

Author: Daniel Walsh
Program: SIRanalysis_kernconv_sds_20.R
Date: 11/18/21

Disclaimer:

This software is preliminary or provisional and is subject to revision. It isbeing provided to meet the need for timely best science. The software has notreceived final approval by the U.S. Geological Survey (USGS). No warranty,expressed or implied, is made by the USGS or the U.S. Government as to thefunctionality of the software and related material nor shall the fact of releaseconstitute any such warranty. The software is provided on the condition thatneither the USGS nor the U.S. Government shall be held liable for any damagesresulting from the authorized or unauthorized use of the software.

Data Preparation

I begin by loading the necessary libraries that will be used in the subsequent analyses.

```
library(R2WinBUGS)
library(R2jags)
library(ggmcmc)
library(matrixStats)
library(coda)
library(Rcpp)
library(RcppArmadillo)
library(parallel)
library(beepr)
library(knitr)
```

Next I determine the number of cores for parallel processing, which will be used to summarize the MCMC results, and set the working directory. I also source the .cpp file for conducting the summary of the results.

```
cores<-detectCores()
sourceCpp("derivedvalues20.cpp")
sourceCpp("probest.cpp")
sourceCpp("probest_multinomial.cpp")
```

I import the raw data files for use in the analyses. I combine individual's data that were reinfected and died into either the infected to disease-dead or infected to dead data sets. Thus, due to small sample sizes, we treat individuals that become reinfected as the same as individuals that have become infected for the first time.

```
h2d<-read.csv("htod.final.csv")  #susceptible to dead data
colnames(h2d)[1] <- "sheep"
h2i<-read.csv("htoi.final.csv")  #susceptible to infected data
i2dd<-read.csv("itodd.final.csv") #infected to disease death data
i2d<-read.csv("itod.final.csv")  #infected to non-disease death data
i2r<-read.csv("itor.final.csv")  #infected to recovered data
```

```
r2i<-read.csv("rtori.final.csv")  #recovered to infected data
r2d<-read.csv("rtod.final.csv")  #recovered to non-disease death data
ritod<-read.csv("ritod.final.csv") #reinfected to non-disease death data
ritodd<-read.csv("ritodd.final.csv") #reinfected to disease death data
hcens<-read.csv("hcens.final.csv") #censored healthy (no clinical signs of disease)
icens<-read.csv("icens.final.csv") #censored infected
rcens<-read.csv("rcens.final.csv") #censored recovered
covar<-read.csv("covariates.csv") #indivdual sheep covariates

colnames(ritod) <- colnames(i2d)  #put reinfected with infecteds
i2d <- rbind(i2d,ritod)

colnames(ritodd) <- colnames(i2dd)
i2dd <- rbind(i2dd,ritodd)
```

Next, I import each individual's covariates, and standardize them by disease state. Missing values are then set equal to the mean covariate level (i.e., 0). I then create and run a function that merges each data file with associated covariate values from a look-up table.

```
##Standardize covariates - for each state's data
covar.elisa <- read.csv("covariates.rawelisa.final.csv")

#Initial Elisa values
covar.elisa$std.initial <- (covar.elisa$initialELISA -
                              mean(covar.elisa$initialELISA,na.rm=T))/
  sd(covar.elisa$initialELISA,na.rm=T)

covar.elisa$std.initial[is.na(covar.elisa$std.initial)] <- 0 #set missing = mean

#Elisa values during infection
covar.elisa$infectELISA <- as.numeric(as.character(covar.elisa$infectELISA))

covar.elisa$std.infectELISA <- (covar.elisa$infectELISA -
                                  mean(covar.elisa$infectELISA,na.rm=T))/
  sd(covar.elisa$infectELISA,na.rm=T)
covar.elisa$std.infectELISA [is.na(covar.elisa$std.infectELISA )] <- 0 #set missing = mean

#Elisa values during recovery
covar.elisa$std.recovELISA <- (covar.elisa$recovELISA -
                                 mean(covar.elisa$recovELISA,na.rm=T))/
  sd(covar.elisa$recovELISA,na.rm=T)

covar.elisa$std.recovELISA [is.na(covar.elisa$std.recovELISA )] <- 0 #set missing = mean

colnames(covar.elisa)[colnames(covar.elisa) %in% c("std.initial","std.infectELISA")] <-
  c("elisaint","elisaa")
levels(covar.elisa$sheep)[2] <- levels(covar$sheep)[2]

###Function to add covariate values to each dataset
covmerge<-function(x,y){
  out<-merge(x,y, by="sheep")
  return(out)
}
```

```
h2d<-covmerge(h2d,covar.elisa) #susceptible to dead
h2i<-covmerge(h2i, covar.elisa) #susceptible to infected
i2dd<-covmerge(i2dd,covar.elisa) #infected to disease-dead
i2d<-covmerge(i2d,covar.elisa)   #infected to dead
i2r<-covmerge(i2r,covar.elisa)   #infected to recovered
r2i<-covmerge(r2i,covar.elisa)   #recovered to reinfected
r2d<-covmerge(r2d,covar.elisa)   #recovered to dead
hcens<-covmerge(hcens,covar.elisa) #susceptible - censored
icens<-covmerge(icens,covar.elisa) #infected - censored
rcens<-covmerge(rcens,covar.elisa) #recovered - censored
```

I then organize the raw data files into a useful format for analyses in JAGS.

```
###Function organize data for creation of data file
extrdata<-function(x,namelist){
  nameout<-vector(length=11,mode="list")
  names(nameout)<-namelist
  nameout[[1]]<-nrow(x)
  nameout[[2]]<-length(unique(x$sheep))
  nameout[[3]]<- x[,grep("left",colnames(x))]
  nameout[[4]]<- x[,grep("right",colnames(x))]
  nameout[[5]]<- x[,grep("censor",colnames(x))]
  nameout[[6]]<- x[,grep("elisaint",colnames(x))]
  nameout[[7]]<- x[,grep("elisaa" ,colnames(x))]
  nameout[[8]]<- x[,grep("dist",colnames(x))]
  nameout[[9]]<- x[,grep("strain",colnames(x))]
  nameout[[10]]<- x[,grep("sex",colnames(x))]-1
  nameout[[11]]<- x[,grep("igs400",colnames(x))]

  return(nameout)
}


#Susceptible to dead #1
h2da<-extrdata(h2d,c("records1","n1","left1","right1","censored1",
                     "elisaint","elisaa","dist","intstr","sex","igs400"))

#Sucsceptible to infected #2
h2ia<-extrdata(h2i,c("records2","n2","left2","right2","censored2",
                     "elisaint","elisaa","dist","intstr","sex","igs400"))

#Infected to infected disease dead #3
i2dda<-extrdata(i2dd,c("records3","n3","left3","right3","censored3",
                       "elisaint","elisaa","dist","intstr","sex","igs400"))

#Infected to infected recovered #4
i2ra<-extrdata(i2r,c("records4","n4","left4","right4","censored4",
                     "elisaint","elisaa","dist","intstr","sex","igs400"))

#Infected to infected non-disease dead #5
i2da<-extrdata(i2d,c("records5","n5","left5","right5","censored5",
                     "elisaint","elisaa","dist","intstr","sex","igs400"))

#Recovered to infected #6
```

```
r2ia<-extrdata(r2i,c("records6","n6","left6","right6","censored6",
                     "elisaint","elisaa","dist","intstr","sex","igs400"))

#Recovered to non-disease dead #7
r2da<-extrdata(r2d,c("records7","n7","left7","right7","censored7",
                     "elisaint","elisaa","dist","intstr","sex","igs400"))
```

I set various maximum time values, and organize the data for censored individuals.

```
T1<-800  #study length
T2<-666  #max time after infection for analysis
T3<-533  #max time after recovery for analysis

lefth<-hcens$lefth  #Censored individuals for susceptible/healthy state
lefti<-icens$lefti  #Censored individuals for infected state
leftr<-rcens$leftr  #Censored individuals for recovered state
righth<-hcens$righth
righti<-icens$righti
rightr<-rcens$rightr
nh1<-length(unique(hcens$sheep))
ns1<-length(unique(icens$sheep))
nr1<-length(unique(rcens$sheep))
```

I use a kernel convolution model to regularize across days of the study for calculating the infection and death due to disease transition hazards. This modeling approach provides a flexible means of accounting for temporal autocorrelation, and permits the estimation of the level of smoothing supported by the data. Details of the approach can be found here. In short, I begin by creating a "distance matrix", which calculates how far each day of the study is from a user-specified set of knots. In our case, I created a knot for each day of the study. So, I create a 800 X 800 distance matrix. For our model I will be using a truncated normal kernel (i.e., $N\left(0,\sigma^2\right)I[d \leq 120]$), which is truncated at 120 days (e.g., days $>$ 120 from day A do not influence the infection time effect for day A).

```
##Distance matrix for time-kernel convolution
intvl<-1 # day interval for knots
knots<-seq(1,800,intvl) #daily knots
distmat<-matrix(0,T1,length(knots))

for(i in 1:nrow(distmat)){
  for(j in 1:length(knots)){
    distmat[i,j]<-abs(i-knots[j]) #absolute value
  }
}
```

To reduce computing time, I determine the indices of the knots that will contribute to the time effect for each day. This is done separately for the infection and death due to disease time effects.

```
col.index<-matrix(0,nrow(distmat),2)  #create indices of knots to include in kernel effect
##for each time point -- truncated at 120 days

col.index<-apply(distmat,1,function(x){  #for infection
  x[x>120]<--1
  out<-c(min(which(x>=0)),
         max(which(x>=0)))
  return(out)
```

```
})

col.index<-t(col.index)

col.index2<-matrix(0,nrow(distmat),2) #for disease death

col.index2<-apply(distmat[1:T2,1:length(seq(1,T2,intvl))],1,function(x){
  x[x>120]<--1
  out<-c(min(which(x>=0)),
         max(which(x>=0)))
  return(out)
})

col.index2<-t(col.index2)
```

I continue the data preparation by creating a zero variable for use with the "zeros" trick in JAGS for specifying the likelihood. I also arrange the data for use with the various distribution functions in JAGS.

```
z<-0  #Zeros for trick in BUGS/JAGS
###create infected matrix for dcat
ninfected2<-cbind(rep(1,i2dda$n3+i2ra$n4+i2da$n5),c(rep(1,i2dda$n3),rep(2,i2ra$n4),
                                                    rep(3,i2da$n5)))

###Create covariate matrix for ninfected
covarcomb<-cbind(c(i2dda$elisaint[i2dda$left3==1],i2ra$elisaint[i2ra$left4==1],
                   i2da$elisaint[i2da$left5==1]),
                 c(i2dda$elisaa[i2dda$left3==1],i2ra$elisaa[i2ra$left4==1],
                   i2da$elisaa[i2da$left5==1]),
                 c(i2dda$dist[i2dda$left3==1],i2ra$dist[i2ra$left4==1],
                   i2da$dist[i2da$left5==1]),
                 c(i2dda$intstr[i2dda$left3==1],i2ra$intstr[i2ra$left4==1],
                   i2da$intstr[i2da$left5==1]),
                 c(i2dda$sex[i2dda$left3==1],i2ra$sex[i2ra$left4==1],
                   i2da$sex[i2da$left5==1]),
                 c(i2dda$igs400[i2dda$left3==1],i2ra$igs400[i2ra$left4==1],
                   i2da$igs400[i2da$left5==1]))

#covariates for i2dda, i2ra, and i2da
colnames(covarcomb)<-c("elisaint","elisaa", "dist","intstr","sex","igs400")

#change column names
colnames(rcens)[colnames(rcens)=="initial.strain"] <- "strain"

colnames(icens)[colnames(icens)=="initial.strain"] <- "strain"

colnames(hcens)[colnames(hcens)=="initial.strain"] <- "strain"
```

Finally, I create a list object that contains all the data that will be used by JAGS during estimation, which is the required format for the R2jags package.

```
#R list for creating datafile for use with BUGS and used directly in JAGS
datain<-list(records1=h2da$records1,n1 = h2da$n1, left1 = h2da$left1, right1 =h2da$right1,
             censored1 = h2da$censored1, elisaint1 = h2da$elisaint, elisaa1 = h2da$elisaa,
             dist1 = h2da$dist, intstr1 = h2da$intstr, sex1 = h2da$sex,
```

```
elisaint1a = h2da$elisaint[h2da$left1==1],
elisaa1a = h2da$elisaa[h2da$left1==1],dist1a = h2da$dist[h2da$left1==1],
intstr1a = h2da$intstr[h2da$left1==1], sex1a = h2da$sex[h2da$left1==1],

records2=h2ia$records2,n2 = h2ia$n2, left2 = h2ia$left2, right2 =h2ia$right2,
censored2 = h2ia$censored2, elisaint2 = h2ia$elisaint, elisaa2 = h2ia$elisaa,
dist2 = h2ia$dist, intstr2 = h2ia$intstr, sex2 = h2ia$sex,
elisaint2a = h2ia$elisaint[h2ia$left2==1],
elisaa2a = h2ia$elisaa[h2ia$left1==1], dist2a = h2ia$dist[h2ia$left2==1],
intstr2a = h2ia$intstr[h2ia$left2==1], sex2a = h2ia$sex[h2ia$left2==1],


records3=i2dda$records3, left3 = i2dda$left3, right3 = i2dda$right3,
censored3 = i2dda$censored3, elisaint3 = i2dda$elisaint,elisaa3=i2dda$elisaa,
dist3 = i2dda$dist, intstr3 = i2dda$intstr, sex3 = i2dda$sex,
igs4003 = i2dda$igs400,

records4=i2ra$records4,left4 = i2ra$left4, right4 = i2ra$right4,
censored4 = i2ra$censored4, elisaint4 = i2ra$elisaint, elisaa4 = i2ra$elisaa,
dist4 = i2ra$dist, intstr4 = i2ra$intstr, sex4 = i2ra$sex,

records5=i2da$records5,left5 = i2da$left5, right5 = i2da$right5,
censored5 = i2da$censored5, elisaint5 = i2da$elisaint, elisaa5 = i2da$elisaa,
intstr5 = i2da$intstr, sex5 = i2da$sex,

records6=r2ia$records6,n6 = r2ia$n6, left6 = r2ia$left6,
right6 = r2ia$right6, censored6 = r2ia$censored6,
elisaint6 = r2ia$elisaint, elisaa6 = r2ia$elisaa,
dist6 = r2ia$dist, intstr6 = r2ia$intstr, sex6 = r2ia$sex,
elisaint6a = r2ia$elisaint[r2ia$left6==1],
elisaa6a=r2ia$elisaa[r2ia$left1==1],
dist6a = r2ia$dist[r2ia$left6==1], intstr6a = r2ia$intstr[r2ia$left6==1],
sex6a = r2ia$sex[r2ia$left6==1],igs4006 = r2ia$igs400,

records7=r2da$records7,n7 = r2da$n7, left7 = r2da$left7,right7 = r2da$right7,
censored7 = r2da$censored7, elisaint7 = r2da$elisaint, elisaa7 = r2da$elisaa,
dist7 = r2da$dist, intstr7 = r2da$intstr, sex7 = r2da$sex,
elisaint7a = r2da$elisaint[r2da$left7==1],
elisaa7a=r2da$elisaa[r2da$left1==1],
dist7a = r2da$dist[r2da$left7==1], intstr7a = r2da$intstr[r2da$left7==1],
sex7a = r2da$sex[r2da$left7==1], igs4007 = r2da$igs400,

elisab = hcens$elisaint,elisaba=hcens$elisaa, distb = hcens$dist,
intstrb = hcens$strain, sexb = hcens$sex,

elisac = icens$elisaint, elisaca=icens$elisaa, distc = icens$dist,
intstrc = icens$strain, sexc = icens$sex, igs400c = icens$igs400,

elisar = rcens$elisaint, elisara=rcens$elisaa, distr = rcens$dist,
intstrr = rcens$strain, sexr = rcens$sex, igs400r = rcens$igs400,

covarcomb = covarcomb,T1=T1, T2=T2, lefth=lefth, righth=righth,lefti=lefti,
righti=righti, leftr=leftr, rightr=rightr, z=z, nh1=nh1, ns1=ns1, nr1=nr1,
```

```
              distmat=distmat,

              NI =i2dda$n3+i2ra$n4+i2da$n5, ninfected=ninfected2, nconst=1/sqrt(2*pi),
              nknots=length(knots), nknots2=length(seq(1,666,intvl)),
              knots=knots,colindex=col.index, colindex2=col.index2)
```

JAGS model

I now describe the JAGS model that describes the underlying probability models that were used in estimation. This model is a multi-state model where the state transitions are modeled using the mixture model for competing risks described by Larson and Dinse (1985). This mixture model involves estimating the probability an individual transitions from one state to another, and estimating the hazard rate of that transition, which describes the time to make each transition. The states are "susceptible", "infected", "recovered", "dead", and "death due to disease". The latter two are absorbing states. There are several possible outcomes for an individual within our multi-state model, and I assign each of these a number. This number will be associated with the various parameters to indicate what hazard or probability of an outcome they correspond. Note, the JAGS model code is contained within an R function as required for the R2jags package. I will describe each portion of the code individually.

```
model <- function(){

  ####Susceptible, infected, recovered model

  ## 1 - Susceptible to dead transition
  ## 2 - Susceptible to infected transition (i.e., clinical disease)
  ## 3 - Infected to disease related dead transition
  ## 4 - Infected to recovered transition (i.e., no more/background clinical signs)
  ## 5 - Infected to non-disease related death transition
  ## 6 - Recovered to infected transition
  ## 7 - Recovered to dead transition
```

The model begins with specifying the priors associated with the kernel convolution model of the time effects for the hazard of infection. As mentioned earlier, we use a truncated normal kernel, $N\left(0,\sigma_k^2\right) I[d \leq 120]$, for the smoothing kernel at each knot. I specify a Gamma (1, 1) prior for $\frac{1}{\sigma_k^2}$ (tauk). For each knot, I specified the latent infection time process, $\overrightarrow{\alpha}$, arising from a $N\left(0,\sigma_\alpha^2\right)$ prior, and used a Gamma (1, 1) prior on the precision, $\frac{1}{\sigma_\alpha^2}$ (taua). Note for efficient computing, I specified the prior on $\overrightarrow{\alpha}$ as $N\left(0,1\right) \times \sigma_\alpha$. I also calculate the ratio of the standard deviations (ratioinf), $\frac{\sigma_k}{\sigma_\alpha}$, which provides an assessment of the amount of smoothing to random variation in the kernel convolution model. A large ratio would indicate that the time effects will be smooth, whereas a low ratio will result in minor smoothing and the random, latent process dominating the time effect.

```
  ##Kernel convolution- time to infection
  # lnsdk ~ dunif(0,4.1)
  # sdk <- exp(lnsdk)
  tauk ~ dgamma(1,1)#<-1/(sdk*sdk)
  sdk <- pow(1/tauk,0.5)
  stauk<-sqrt(tauk)

  taua ~ dgamma(1,1) #precision of time random effect
  sda <- 1/sqrt(taua)
```

```
for(i in 1:nknots){
  alpha[i] ~ dnorm(0,1)
  alphau[i] <- sda*alpha[i]
}

ratioinf<-sdk/sda #ratio of variability
```

The last step is to calculate the random time effect for the infection hazard for each day of the study. This is done using the following equation:

$$KA_i = \sum_{k=1}^{K} \frac{1}{\sqrt{2\pi}\sigma_k} exp\left(\frac{dist_{i,k}^2}{2\sigma_k^2}\right) I[d \leq 120] \times \alpha_k,$$

where KA_i is the time effect for the i^{th} day. It is a sum over the K knots of the truncated Normal density (described above) values for the distance each day is from each knot multiplied by the latent infection time process at each knot. Additionally, the columns are restricted to sum to 1.

```
for(i in 1:T1){
  for(j in 1:nknots){
    temp1[i,j] <-(stauk*nconst*exp(-0.5*distmat[i,j]^2*tauk))
  }
}

# for(j in 1:nknots){
#   temp2[j] <-  sum(temp1[1:T1,j])
# }


for(i in 1:T1){
  for(j in colindex[i,1]:colindex[i,2]){
    temp[i,j] <- temp1[i,j]*alphau[j]#(temp1[i,j]/temp2[j])*alphau[j]

  }
  KA[i] <- sum(temp[i,colindex[i,1]:colindex[i,2]])
}
```

I then construct a similar kernel convolution model of the time effects for the hazard of dying from disease.

```
##Kernel convolution- time to infection to disease-death
# lnsdkdd ~ dunif(0,4.1)
# sdkdd <- exp(lnsdkdd)
taukdd ~ dgamma(1,1)#<- 1/(sdkdd*sdkdd)
sdkdd <- pow(1/taukdd,0.5)

tauadd ~ dgamma(1,1) #precision of time random effect
sdadd <-  1/sqrt(tauadd)

staukdd<-sqrt(taukdd)
ratiodd<-sdkdd/sdadd #ratio of variability

for(i in 1:nknots2){
  alphadd[i] ~ dnorm(0,1)
  alphaddu[i]<-sdadd*alphadd[i]
}
```

```
for(i in 1:T2){
  for(j in 1:nknots2){
    tempdd1[i,j] <-(staukdd*nconst*exp(-0.5*distmat[i,j]^2*taukdd))
  }
}

# for(j in 1:nknots2){
#   tempdd2[j] <-  sum(tempdd1[1:T2,j])
# }


for(i in 1:T2){
  for(j in colindex2[i,1]:colindex2[i,2]){
    tempdd[i,j] <- tempdd1[i,j]*alphaddu[j]# (tempdd1[i,j]/tempdd2[j])*alphaddu[j]

  }
  KAdd[i] <- sum(tempdd[i,colindex2[i,1]:colindex2[i,2]])
}
```

Next, I develop the model for the hazard of transitioning from the susceptible state to the dead state. This hazard is modeled as a constant, and we specify a Uniform(-100,100) prior on the intercept term (gamma1) specified on the log hazard rate. We then calculate the contribution to the log-likelihood for each individual for this hazard using a daily, piece-wise constant model. Throughout the model, UCH variables represent the unit cumulative hazard, and are the integration of the hazard rate over each day.

```
##1
  #Priors
  gamma1 ~ dunif(-100,100) #baseline-log hazard

  #Event-time Log-Likelihood
  for(j in 1:records1){
    for(k in left1[j]:(right1[j]-1)){
      UCH1[j,k] <- exp(gamma1)  # constant hazard model for non-disease related deaths
    }
    surv1[j] <- (-sum(UCH1[j,left1[j]:(right1[j]-1)]))*censored1[j] +
      (1-censored1[j])*log(1-exp(-sum(UCH1[j,left1[j]:(right1[j]-1)])) )
  }
```

I then specify the priors for the parameters for the probability that an individual transitions to the infected or the dead state (i.e., 1-Pr[transitioning to infected]). I use diffuse Normal (0, 100) priors for the intercept and the categorical effects of the individual's *Mycoplasma ovipneumoniae* (Movi) strain type as they entered the study. These parameters are specified on the logit scale, and I create the individual probability of transitioning to the infected state. Lastly, I determine the total contribution to the log-likelihood for each individual that died while in the susceptible state. This involves summing the log-likelihood contribution for the hazard described above with the natural log of 1-Pr[transitioning to infected] for each individual that transitioned from susceptible state to the dead state.

```
  #Priors
  for(i in 1:2){
    beta01[i] ~ dnorm(0,0.01)
  }

  beta01str[1] <- 0 #set to zero for base-line strain
  for(i in 2:4){
```

```
  beta01str[i] ~ dnorm(0,0.01) # strain parm- categorical
}


#Risk-type Log-Likelihood
for(i in 1:n1){
  logit(p1[i])<-beta01[1] + beta01[2]*elisaint1[i] + beta01str[intstr1[i]]


  risk1[i] <- log(1 - p1[i]) #probability not transitioning to infected
}


logl1<-sum(surv1[]) + sum(risk1[])  #log likelihood for all individuals in ##1
```

I follow similar methods for the 6 remaining outcomes. The next outcome (#2) is the transition from the susceptible state to the infected state. The hazard model for this model includes the intercept term (gamma2), an effect of the initial Movi elisa result as the individual entered the study (betah1[1]), the distance from the pen containing individuals from Nevada (betah1[2]), and the time effect (KA) derived from the kernel convolution model described above. We specify diffuse Normal (0, 100) priors for the elisa and distance effects, and a Uniform(-100, 100) prior for the intercept.

```
##2
#Priors
gamma2 ~ dunif(-100,100) #baseline-log hazard

for(i in 1:2){
  betah1[i] ~ dnorm(0,0.01)
}


#Event-time Log-Likelihood
for(j in 1:records2){
  for(k in left2[j]:(right2[j]-1)){
    UCH2[j,k] <- exp(gamma2 + betah1[1] *elisaint2[j]+betah1[2]*dist2[j] + KA[k])
  }
    surv2[j] <- (-sum(UCH2[j,left2[j]:(right2[j]-1)]))*censored2[j] +
    (1-censored2[j])*log(1-exp(-sum(UCH2[j,left2[j]:(right2[j]-1)])) )
}
```

For the probability of transitioning to infected from the susceptible state, I included an effect for the individual's initial Movi elisa result (beta01) and initial Movi strain (beta01str). Priors for these parameters were described above. I then calculated the log-likelihood contribution of each individual that became infected with Movi.

```
#Risk-type Log-Likelihood
for(i in 1:n2){
  logit(p2[i])<-beta01[1]+beta01[2]*elisaint2[i]+beta01str[intstr2[i]]
  risk2[i]<-log(p2[i])  #probability transitioning to infected
}

logl2<-sum(risk2[])+sum(surv2[])  #log likelihood for all individuals in ##2
```

The last outcome for individuals in the susceptible state is to remain in this state for the entire study (i.e., right censored). These individuals use the same models described above and I develop the log-likelihood contribution for each individual in the same manner as previously described.

```
    ##Stay Susceptible - Censored Individuals
    for(i in 1:nh1){
      for(k in lefth[i]:(T1-1)){
        UCHa[i,k] <- exp(gamma1)    # constant hazard model for non-disease related deaths
      }
      for(k in lefth[i]:(T1-1)){
        UCHb[i,k] <- exp(gamma2 + betah1[1]*elisab[i] + betah1[2]*distb[i] + KA[k])
        # constant hazard model for infection
      }

      # transitioning to infected
      logit(p1a[i]) <-beta01[1] + beta01[2]*elisab[i] + beta01str[intstrb[i]]

      #log likelihood for censored
      loglh[i] <- log(p1a[i] *exp(-sum(UCHb[i,lefth[i]:(T1-1)]))+(1-p1a[i])*
                        exp(-sum(UCHa[i,lefth[i]:(T1-1)])))
    }
```

I finally sum each of the log-likelihood contributions for individuals that transitioned or remained in the susceptible state, which represents the likelihood contribution from the susceptible state.

```
    logl1all<-sum(loglh[])+logl1+logl2  #Susceptible contribution to log-likelihood
```

The next state is the infected state. There are three possible transitions from this state: recovered, death, and death due to disease. I specify the same diffuse priors as used before for each of the parameters for the various hazard rates of transitioning from the infected state.

```
    ##Priors
    gamma3~ dunif(-100,100) #baseline-log hazard
    gamma4 ~ dunif(-100,100) #baseline-log hazard

    for(i in 1:2){
      betah3[i] ~ dnorm(0,0.01)
      betah4[i] ~ dnorm(0,0.01)
    }


    betah4str[1] <-0
    betah4str[2] <-0
    betah4str[3] ~ dnorm(0,0.01) #only 2 and 3 strains recovered
    betah4str[4] <-0
```

The first transition hazard is the transition to death due to disease. I model the log hazard to include an intercept term (gamma3), an effect of the individual's initial Movi elisa result (betah3[1]), the individual's Movi average elisa values over all Movi serological test results (betah3[2]), an indicator of whether the individual was ever infected with the 400 strain of Movi (betaigs3), and the time effect (KAdd) from the kernel convolution model described previously. I also calculate the log-likelihood contribution for this hazard rate.

```
##3

#Event-time Log-Likelihood
for(j in 1:records3){
  for(k in left3[j]:(right3[j]-1)){
    UCH3[j,k] <- exp(gamma3 + betah3[1]*elisaint3[j] + betah3[2]*elisaa3[j] +
                       KAdd[k]) # hazard model for disease-death
  }
  surv3[j] <- (-sum(UCH3[j,left3[j]:(right3[j]-1)]))*censored3[j] +
    (1-censored3[j])*log(1-exp(-sum(UCH3[j,left3[j]:(right3[j]-1)])) )
}

logl3<-sum(surv3[])  #log likelihood for all individuals in ##3
```

The next transition is from the infected state to the recovered state. I model the log hazard to include an intercept term (gamma4), an effect of the individual's initial Movi elisa result (betah4[1]), the individual's Movi average elisa values over all Movi serological test results (betah4[2]), and the individual's initial Movi strain (betah4str). I also calculate the log-likelihood contribution for this hazard rate.

```
##4
#Event-time Log-Likelihood
for(j in 1:records4){
  for(k in left4[j]:(right4[j]-1)){
    UCH4[j,k] <- exp(gamma4 + betah4[1]*elisaint4[j] + betah4[2]*elisaa4[j] +
                       betah4str[intstr4[j]]) # hazard model for recovery
  }
  surv4[j] <- (-sum(UCH4[j,left4[j]:(right4[j]-1)]))*censored4[j] +
    (1-censored4[j])*log(1-exp(-sum(UCH4[j,left4[j]:(right4[j]-1)])) )
}


logl4<-sum(surv4[])  #log likelihood for all individuals in ##4
```

The last transition hazard is for the transition to the dead state. This is modeled using the same parameter (gamma1) for the transition to the dead state from the susceptible state because this risk is independent of disease state.

```
##5 - constrained to be equal to ##1 - healthy deaths

#Event-time Log-Likelihood
for(j in 1:records5){
  for(k in left5[j]:(right5[j]-1)){
    UCH5[j,k] <- exp(gamma1)# constant hazard model for death - same as healthy death
  }
  surv5[j] <- (-sum(UCH5[j,left5[j]:(right5[j]-1)]))*censored5[j] +
    (1-censored5[j])*log(1-exp(-sum(UCH5[j,left5[j]:(right5[j]-1)])) )
}


logl5<-sum(surv5[])  #log likelihood for all individuals in ##5
```

The next step is to estimate the probabilities of transitioning using the categorical distribution, which are constrained to sum to 1. I first specify the prior values for the parameters for the various probabilities.

```
#Risk-type Likelihood
##Priors
for(i in 1:4){
  beta0dd[i] ~ dnorm(0,0.01)
  beta0rec[i] ~ dnorm(0,0.01)
}


## @knitr ddhaz

for(i in 2:4){
  beta0ddstr[i] ~ dnorm(0,0.01)
  beta0recstr[i] ~ dnorm(0,0.01)
}

beta0ddstr[1]<-0  #set to zero - 400 set as base-line strain - probability
beta0recstr[1]<-0
```

I model the probability of dying of disease as a function of an intercept (beta0dd[1]), the individual's initial Movi elisa result (beta0dd[2]), the individual's average Movi elisa values (beta0dd[3]), and the individual's initial Movi strain (beta0ddstr). I model the probability of recovering from infection as a function of an intercept (beta0rec[1]), individual's initial Movi elisa result (beta0rec[2]), the individual's average Movi elisa values (beta0rec[3]), and the individual's initial Movi strain (beta0recstr). The final probability, the probability of dying from cause unrelated to disease is modeled as 1-sum of the previous two probabilities to enforce the sum to 1 constraint. All parameters are on the logit scale.

```
for(i in 1:NI){
  #probability of disease death
  P[i,1] <- exp(beta0dd[1] + beta0dd[2]*covarcomb[i,1] + beta0dd[3]*covarcomb[i,2] +
                beta0dd[4]*covarcomb[i,6] + beta0ddstr[covarcomb[i,4]])/
    (1 + exp(beta0dd[1] + beta0dd[2]*covarcomb[i,1] + beta0dd[3]*covarcomb[i,2] +
               beta0dd[4]*covarcomb[i,6] + beta0ddstr[covarcomb[i,4]]) +
          exp(beta0rec[1] + beta0rec[2]*covarcomb[i,1] + beta0rec[3]*covarcomb[i,2]+
               beta0rec[4]*covarcomb[i,6] + beta0recstr[covarcomb[i,4]]))

  #probability of recovery
  P[i,2] <- exp(beta0rec[1] + beta0rec[2]*covarcomb[i,1] +
                beta0rec[3]*covarcomb[i,2] + beta0rec[4]*covarcomb[i,6] +
                beta0recstr[covarcomb[i,4]])/
    (1+exp(beta0dd[1] + beta0dd[2]*covarcomb[i,1] + beta0dd[3]*covarcomb[i,2] +
          beta0dd[4]*covarcomb[i,6] + beta0ddstr[covarcomb[i,4]]) +
       exp(beta0rec[1] + beta0rec[2]*covarcomb[i,1] + beta0rec[3]*covarcomb[i,2] +
          beta0rec[4]*covarcomb[i,6] + beta0recstr[covarcomb[i,4]]))

  P[i,3]<-1-sum(P[i,1:2])   #probability of non-disease death

  ninfected[i,2] ~ dcat(P[i,])  #likelihood contribution
}
```

Finally, I model the individuals that remain in the infected state until the end of the study, and calculate their log-likelihood contribution.

```
    ##Stay Infected - Censored Individuals
    for(i in 1:ns1){
      for(k in lefti[i]:(righti[i]-1)){
        UCHc[i,k] <- exp(gamma3 + betah3[1]*elisac[i] + betah3[2]*elisaca[i]+
                           KAdd[k])
            # hazard model for disease related deaths
      }
      for(k in lefti[i]:(righti[i]-1)){
        UCHd[i,k] <- exp(gamma4 + betah4[1]*elisac[i] + betah4[2]*elisaca[i] +
                           betah4str[intstrc[i]])  # hazard model for recovery
      }
      for(k in lefti[i]:(righti[i]-1)){
        UCHe[i,k] <- exp(gamma1)   # constant hazard model for non-disease related deaths
      }

      # probability of disease death
      pc[i,1] <-exp(beta0dd[1] + beta0dd[2]*elisac[i] + beta0dd[3]*elisaca[i] +
                  beta0dd[4]*igs400c[i]+ beta0ddstr[intstrc[i]])/
        (1 + exp(beta0dd[1] + beta0dd[2]*elisac[i] + beta0dd[3]*elisaca[i] +
                  beta0dd[4]*igs400c[i] + beta0ddstr[intstrc[i]]) +
              exp(beta0rec[1]+beta0rec[2]*elisac[i] + beta0rec[3]*elisaca[i] +
                    beta0rec[4]*igs400c[i] + beta0recstr[intstrc[i]]))

      #probability of infected to recovered censored
      pc[i,2] <- exp(beta0rec[1] + beta0rec[2]*elisac[i] + beta0rec[3]*elisaca[i] +
                     beta0rec[4]*igs400c[i] + beta0recstr[intstrc[i]])/
        (1+exp(beta0dd[1] + beta0dd[2]*elisac[i] + beta0dd[3]*elisaca[i] +
                 beta0dd[4]*igs400c[i] + beta0ddstr[intstrc[i]]) +
           exp(beta0rec[1] + beta0rec[2]*elisac[i] + beta0rec[3]*elisaca[i] +
                 beta0rec[4]*igs400c[i] + beta0recstr[intstrc[i]]))


      pc[i,3]<-1-sum(pc[i,1:2])

      #log likelihood for censored
      logls[i] <- log(pc[i,1] *exp(-sum(UCHd[i,lefti[i]:(righti[i]-1)]))+pc[i,2]*
                        exp(-sum(UCHc[i,lefti[i]:(righti[i]-1)]))
                      +pc[i,3]*exp(-sum(UCHe[i,lefti[i]:(righti[i]-1)])))
    }
```

The total contribution of infected individuals to the log-likelihood is calculated.

```
    #Infected contribution to log-likelihood
    logl2all<-sum(logls[]) + logl3 + logl4 + logl5
```

The final state that is not an absorbing state is the recovered state. The development follows those previously described. I first set the priors for the transition hazards.

```
   ##6
    #Priors
    gamma6 ~ dunif(-100,100) #baseline-log hazard
    betah6 ~ dnorm(0,0.01)
    beta6igs[1] <- 0
    beta6igs[2] ~ dnorm(0,0.01)
```

I model the hazard of transitioning back to the infected state (i.e., reinfection) as a function of an intercept (gamma6), the individual's average Movi elisa values (betah6), and whether the individual was ever infected with the 400 strain of Movi (betaigs6) on the logit scale.

```
    #Event-time Log-Likelihood
    for(j in 1:records6){
      for(k in left6[j]:(right6[j]-1)){
        UCH6[j,k] <- exp(gamma6) # hazard model for recovered to infected
      }
      surv6[j] <- (-sum(UCH6[j,left6[j]:(right6[j]-1)]))*censored6[j] +
        (1-censored6[j])*log(1-exp(-sum(UCH6[j,left6[j]:(right6[j]-1)])) )
    }
```

Next I specify the probability of reinfection as a function of an intercept (beta06[1]), individual's initial Movi elisa result (beta06[2])the individual's average Movi elisa values (beta06[2]), and the individual's initial Movi strain (beta06str). I also calculate the log-likelihood contribution of all individual that reinfect.

```
   for(i in 1:n6){
      logit(p6[i])<-beta06[1] + beta06str[intstr6[i]]
      risk6[i] <- log(p6[i]) #probability recovered transitioning to infected
    }


    logl6<-sum(surv6[])+sum(risk6[])  #log likelihood for all individuals in ##6
```

The final outcome is dying in the recovered state. This transition hazard was modeled using the same parameter that was used in the previous states (gamma1).

```
    ##7
    #Event-time Log-Likelihood
    for(j in 1:records7){
      for(k in left7[j]:(right7[j]-1)){
        UCH7[j,k] <- exp(gamma1) # hazard model for recovered to infected
      }
      surv7[j] <- (-sum(UCH7[j,left7[j]:(right7[j]-1)]))*censored7[j] +
        (1-censored7[j])*log(1-exp(-sum(UCH7[j,left7[j]:(right7[j]-1)])) )
    }
```

The probability of dying in the recovered state is 1-Pr(reinfection). I calculate the log-likelihood contribution for individuals that die in the recovered state.

```
    for(i in 1:n7){
      logit(p7[i])<-beta06[1] + beta06str[intstr7[i]]
      risk7[i] <- log(1-p7[i]) #probability recovered to non-disease death
    }

    logl7<-sum(risk7[]) + sum(surv7[])  #log likelihood for all individuals in ##7
```

I then account for individuals that were in the recovered state at the end of the study.

```
    ##Stay Recovered - Censored Individuals
    for(i in 1:nr1){
      for(k in leftr[i]:(rightr[i]-1)){
        UCHf[i,k] <- exp(gamma1)   # constant hazard model for non-disease related deaths
```

```
      }
      for(k in leftr[i]:(rightr[i]-1)){
        UCHg[i,k] <- exp(gamma6) # constant hazard model for infection
      }

      #probability transitioning to infected
      logit(p7a[i]) <-beta06[1] + beta06str[intstrr[i]]

      #log likelihood for censored
      loglh7[i] <- log(p7a[i] *exp(-sum(UCHg[i,leftr[i]:(rightr[i]-1)])) + (1-p7a[i])*
                         exp(-sum(UCHf[i,leftr[i]:(rightr[i]-1))))
    }
```

The total log-likelihood contribution for all individuals that were in the recovered state.

```
    logl3all<-sum(loglh7[]) + logl6 + logl7  #Recovered contribution to log-likelihood
```

Finally, I conclude the JAGS model statement with the "zeros" trick, which allows us to use the Poisson likelihood to incorporate all the log-likelihood components that we calculated above. This trick is necessary because our model doesn't align with any standard probability distribution in JAGS. We add a large constant to the Poisson parameter (phi) to ensure that its values is greater than zero.

```
    #"Zeros" trick (Lunn et al. 2013, p.204)
    const<-10000 #ensures phi[i] >0
    phi <--(logl1all + logl2all +logl3all ) + const
      z ~ dpois(phi)
}
```

I now set up the estimation procedure using the R2jags package to leverage JAGS within program R. First, we specify a function that creates the initial values for each of the parameters for each of three MCMC chains.

```
###Create initial values - use random number generator so don't have to
####specify starting values for each chain

init1 <- function(){list("gamma1"=runif(1,-3,0), "gamma2"=runif(1,-3,0),
              "gamma3"=runif(1,-3,0), "gamma4"=runif(1,-3,0), "gamma6"=runif(1,-3,0),
              "beta01"=runif(2,-2,0.5), "betah1"=runif(2,-2,0.5),"betah3"=runif(2,-2,0.5),
              "betah4"=runif(2,-2,0.5), "beta0dd"=c(-1,0,0,0),
              "beta0rec"=c(-1,0,0,0), "beta01str" = c(NA,runif(3,-4,0.5)),
              "betah4str" = c(NA,NA,runif(1,-4,0),NA), "beta0ddstr"=c(NA,runif(3,-4,0.5)),
              "beta0recstr"=c(NA,runif(3,-4,0.5)),"betaigs3"=c(NA,runif(1,-4,0.5)),
              "beta06str"= c(NA,NA,runif(1,-4,0.5),NA),
              "beta06"=runif(1,-2,0.5),"alpha"=rep(0,nknots),"alphadd"=rep(0,nknots2))}
```

I set the parameters which I would like to monitor during the estimation routine.

```
#identify parms to monitor
parms <- c("beta0rec","beta0dd","beta0ddstr","beta0recstr","gamma1","gamma2","gamma3",
         "gamma4","gamma6","beta01","betah1","betah3","betah4","beta01str","betah4str",
         "betaigs3","beta06str","sdk", "sda","sdkdd", "sdadd","KA",
         "KAdd","beta06","ratioinf","ratiodd")

  ## @knitr fit
```

```
  jagsfit <- jags(datain, init1, parameters.to.save=parms,
                  jags.module = c("dic"), n.chains=1, n.iter=10,
                  model.file=model, n.burnin=1, n.thin=1,
                  DIC=FALSE, working.directory=NULL)  #fit the model in JAGS

  ## @knitr temp3
  for(i in 1:100){
    beep(2)
  }
```

The final step is to run the MCMC chains. I use the parallel computing algorithms within the R2jags package to increase computational efficiency.

Summarize results

To summarize the results of the MCMC routine, I convert the MCMC results to a coda object in R.

```
####Summarize Posterior Distributions
jags.mcmc <- as.mcmc(jagsfit)  #convert to mcmc object for plotting below
```

To properly summarize the time effects for infection and dying due to disease transition hazards, I create a function that properly orders them in the output by day. The original ordering by R treats the name of the time variables as character values, which causes the ordering to be incorrect (e.g., 1, 10, 11, etc.). I then use the function to reorder the time variables for both hazards. I also create a design matrix for each individual's initial Movi strain for use in the summary functions.

```
##Obtain colnames for proper sorting of random time effects
orderfunc <- function(name){
  temp<-strsplit(colnames(jags.mcmc[[1]])[grep(name,colnames(jags.mcmc[[1]]))],"\\[")
  temp2<-vector(length=length(temp),mode="numeric")
  for(i in 1:length(temp)){
    temp2[i]<-as.numeric(strsplit(temp[[i]][2],"\\]")[[1]]  ) #colnames as numeric
  }
  return(order(temp2))
}


orderkadd <- orderfunc("KAdd")
orderka <- orderfunc("KA\\[")
test <- as.matrix(jags.mcmc)
test2 <- test[,grep("KA\\[",colnames(test))]
test2 <- test2[,orderka]


test3 <- test[,grep("KAdd",colnames(test))]
test3 <- test3[,orderkadd]


strain.design <- matrix(0,nrow(covar),4)
for(i in 1:nrow(covar)){
  strain.design[i,covar[i,"strain"]]<-1
}
```

I then source a C++ file that I created that provides the values for the "survival" curves (i.e., probability of remaining in each state through time), the non-constant transition hazard for infection and dying due to disease, and the temporally varying states of a population of 150 bighorns that experience the disease event that we experienced during our study. This latter analysis is to provide managers a frame of reference for how a population in the wild may respond to the introduction of a novel Movi strain.

```
sourceCpp("derivedvalues20.cpp")
sourceCpp("probest.cpp")
sourceCpp("probest_multinomial.cpp")
results <- deriveval(test[,"gamma1"],test[,"gamma2"],test[,"gamma3"],test[,"gamma4"],
          test[,"gamma6"], test[,grep("beta01",colnames(test))],
          test[,grep("beta06",colnames(test))],test[,grep("beta0dd",colnames(test))],
          test[,grep("beta0rec",colnames(test))],test[,grep("betah1",colnames(test))],
          test[,c(grep("betah3",colnames(test)),grep("betaigs3",colnames(test)))],
          test[,grep("betah4",colnames(test))],
          test2,
          test3,150,seq(0,720,by=30),
          cbind(1,covar.elisa[,"elisaint"], strain.design, covar.elisa[,"elisaa"],
                covar.elisa[,"dist"], 0, covar.elisa[,"igs400"]),
          cores)




colnames(results$Nmu) <- seq(0,720,by=30)
rownames(results$Nmu) <- c("Susceptible","Dead","Infected","Recovered","Disease-dead")
rownames(results$quantile025) <- c("Susceptible","Dead",
                                   "Infected","Recovered","Disease-dead")
rownames(results$quantile975) <- c("Susceptible","Dead",
                                   "Infected","Recovered","Disease-dead")
```

Next I provide the summary of the parameter estimates I monitored during the MCMC procedure. Note, I remove the time effects for the non-constant hazards because they are best summarized in plots. I also rename the parameters to make the names more meaningful, and then I write the summary results to a .csv file.

```
#remove random time effects for summary
jags.mcmc2 <- jags.mcmc[,-grep("KA\\[",colnames(jags.mcmc[[1]]))]
jags.mcmc2 <- jags.mcmc2[,-grep("KAdd",colnames(jags.mcmc2[[1]]))]

for(i in 1:length(jags.mcmc2)){
  colnames(jags.mcmc2[[i]])[c(1:48)]<-c(
            "probinfec-int","probinfec-elisai","probinfec-400","probinfec-404",
            "probinfec-393", "probinfec-398",

            "probr2i-int","probr2i-400", "probr2i-404", "probr2i-393",
            "probr2i-398",

            "probdd-int", "probdd-elisai", "probdd-elisaa", "probdd-igs400",
            "probdd-400", "probdd-404", "probdd-393", "probdd-398",

            "probrec-int","probrec-elisai", "probrec-elisaa", "probrec-igs400",
            "probrec-400", "probrec-404", "probrec-393", "probrec-398",
```

```
            "timeinfec-elisai", "timeinfec-dist",

            "timei2dd-elisai", "timei2ddelisaa",

            "timerecov-elisai", "timerecov-elisaa",
            "timerecov-400", "timerecov-404", "timerecov-393", "timerecov-398",

            "time2d-int", "time2i-int", "timei2dd-int", "timei2r-int",
            "timer2i-int", "ratiokernelsd-dd","ratiokernelsd-inf","precision-inf",
            "precision-dd", "precision-time-inf","precision-time-dd")

}

sumresults <- summary(jags.mcmc2,quantiles = c(0.025, 0.25, 0.5, 0.75, 0.975))
write.csv(cbind(sumresults[[1]],sumresults[[2]]),"results_final19.csv")
```

I conclude the code by creating plots of the survival and hazard curves.

```
##Create plots of hazard/survival functions

#hazard of infected to disease death
data.p1 <- as.data.frame(results$hazdd)
colnames(data.p1) <- c("median","LCB","UCB")
hazdd.plot <- ggplot(data=data.p1, aes(x=1:nrow(data.p1),y=median,group=1)) +
  geom_line() +
  geom_ribbon(aes(ymin=LCB,ymax=UCB),alpha=0.3) +
  xlab("days") +
  ylab("death due to disease hazard")
hazdd.plot

#hazard of susceptible to infected
data.p2 <- as.data.frame(results$hazinf)
colnames(data.p2) <- c("median","LCB","UCB")
hazinf.plot <- ggplot(data=data.p2,aes(x=1:nrow(data.p2),y=median,group=1)) +
  geom_line() +
  geom_ribbon(aes(ymin=LCB,ymax=UCB),alpha=0.3) +
  xlab("days") +
  ylab("infection hazard")
hazinf.plot


#survival curve of susceptible
data.p3 <- as.data.frame(results$Ssucept)
colnames(data.p3) <- c("median","LCB","UCB")
Ss.plot <- ggplot(data=data.p3,aes(x=1:nrow(data.p3),y=median)) +
  geom_line() +
  geom_ribbon(aes(ymin=LCB,ymax=UCB),alpha=0.3) +
  xlab("days") +
  ylab("Transition curve - susceptible")
Ss.plot


#survival curve of infected
```

```
data.p4 <- as.data.frame(results$Sinfect)
colnames(data.p4) <- c("median","LCB","UCB")
Si.plot <- ggplot(data=data.p4, aes(x=1:nrow(data.p4),y=median)) +
  geom_line() +
  geom_ribbon(aes(ymin=LCB,ymax=UCB),alpha=0.3) +
  xlab("days") +
  ylab("Transition curve - infected")
Si.plot

#survival curve of recovered
data.p5 <- as.data.frame(results$Srec)
colnames(data.p5) <- c("median","LCB","UCB")
Sr.plot <- ggplot(data=data.p5,aes(x=1:nrow(data.p5),y=median)) +
  geom_line() +
  geom_ribbon(aes(ymin=LCB,ymax=UCB),alpha=0.3) +
  xlab("days") +
  ylab("Transition curve - recovered")
Sr.plot

#plot of elisa effect for prob h2i

beta.in <- test[,grep("beta01\\[",colnames(test))]
covar.range <- round(range(c(datain$elisaint1,datain$elisaint2)),2)
X.in <- cbind(1,seq(covar.range[1],covar.range[2], by = 0.05))

prob.effect1 <- probest(X.in, beta.in, cores)

data.p6 <- data.frame(x = X.in[,2], mu = t(prob.effect1$mu),
                      lcl = prob.effect1$quantile025,
                      ucl = prob.effect1$quantile975)

peffect1.plot <- ggplot(data=data.p6,aes(x=x,y=mu)) +
  geom_line() +
  geom_ribbon(aes(ymin=lcl,ymax=ucl),alpha=0.3) +
  xlab("Initial standardized ELISA value") +
  ylab("Probability of infection")
peffect1.plot


#plot of average elisa effect for prob i2dd
beta.in1 <- test[,grep("beta0dd\\[",colnames(test))]
beta.in2 <- test[,grep("beta0rec\\[",colnames(test))]

beta.in1 <- beta.in1[,1:3] #remove strain covariate
beta.in2 <- beta.in2[,1:3] #remove strain covariate

covar.range1 <- round(range(c(datain$covarcomb[,2],datain$covarcomb[,2])),2)

X.in1 <- cbind(1, mean(datain$covarcomb[,1]),
               seq(covar.range1[1],covar.range1[2], by = 0.05))

#Infected to disease dead
prob.effect2 <- probestmulti(X.in1, beta.in1, beta.in2, cores)
```

```
data.p7 <- data.frame(x = X.in1[,3], mu = t(prob.effect2$mu),
                      lcl = prob.effect2$quantile025,
                      ucl = prob.effect2$quantile975)

peffect2.plot <- ggplot(data=data.p7,aes(x=x,y=mu)) +
  geom_line() +
  geom_ribbon(aes(ymin=lcl,ymax=ucl),alpha=0.3) +
  xlab("Average standardized ELISA value") +
  ylab("Probability of dying of disease")
peffect2.plot


#Infected to recovered
prob.effect3 <- probestmulti(X.in1, beta.in2, beta.in1, cores)


data.p8 <- data.frame(x = X.in1[,3], mu = t(prob.effect3$mu),
                      lcl = prob.effect3$quantile025,
                      ucl = prob.effect3$quantile975)

peffect3.plot <- ggplot(data=data.p8,aes(x=x,y=mu)) +
  geom_line() +
  geom_ribbon(aes(ymin=lcl,ymax=ucl),alpha=0.3) +
  xlab("Average standardized ELISA value") +
  ylab("Probability of recovering from disease")
peffect3.plot
```