R Code for Collision Fatality Model

The code for the collision fatality model is written as several separate components that will eventually be put together into an R package. For now it is necessary to save several separate R files:

TemplateCMData.R –project data for the collision model (includes exposure survey data/results); this file may be renamed (e.g. Example ProjectCMData.R) and should be run first.
CollisionModel.R – FWS basic model for predicting collision fatalities; this file may be renamed.
DistFcns.R— functions for distributions that are required; this file should not be renamed.
FatalFcns.R— fatality estimation functions; this file should not be renamed.
rvsmry.R—summarizes the output from the rv package; this file should not be renamed.

The code for each of these is presented below (with some example project data included in CMDataTemplate.R) and should be saved as individual files titled as above. Code lines are presented in red with project specific input indicated in blue (e.g. **cProject<-"ExampleProject"**); other lines are in black should be excluded or commented out using '#'. Some familiarity with R will helpful before attempting to run this analysis. R resources and the R software package are available at http://www.R-project.org/. Additionally, running the collision model will require installing the rv package for R.

TemplateCMData.R:
(the text between the solid lines below may be copied directly into a text file and saved as an R file (e.g. ExmplProjectCMData.R)

```
### <Project Name> Exposure Survey and Project Data ###

# Name the project (this is used to label the final output)
 cProject<-"Patuxent Wind Company"

# Set the number of turbines being considered for the project
 nTurbine<-18

# Set the rotor diameter and calculate the radius (in kilometers)
 RotorDKm <-50/1000 # Rotor diameter
# (the rotor diameter in this case is 50 meters, which is being converted into 50/1000, or 0.05,
# kilometers)

# The rotor radius can be calculated from the rotor diameter
 RotorRKm<- RotorDKm/2
# or can be set directly in lieu of setting the rotor diameter
# RotorRKm<-0.05

# the rotor radius buffer (to account for turbulence if appropriate); the default is 0 km.
 RotorBuffKm<-0 # Turbulence buffer
```

```
# the rotor radius buffer is added to the rotor radius to determine the final hazardous area radius for
# each turbine
  HazRadKm<-RotorRKm+RotorBuffKm

# If the turbine sizes are not known, the hazardous area radius may be defined directly (comment out
# the rotor size code above and uncomment the command defining HazRadKm shown here
#  HazRadKm<-100/1000

# Use the radius of the hazardous area around a turbine to calculate the total hazardous area (in
# square kilometers) for the project by multiplying by the number of turbines.
  HzKM2<-nTurbine*pi*HazRadKm^2

# count duration in hours (default is 2 hr)
  CntHr<-2  # count duration
# (20 minute counts would be converted into 20/60, or 0.3333333, hours)

# total days of the year that each strata represents (if only calculating 1 annual estimate StrDays
# would be 365 in non-leap years):
  Days=c(365)
# average number of daylight hours per day (default is 12)
  LtHrPerDay=c(12)

# Here we create an ExpSvy data frame. Using a data frame simplifies the simultaneous consideration
# of varying inputs for multiple strata but can also accommodate inputs for a single strata (as with the
# example here)
  ExpSvy<-data.frame(
# name the strata
  row.names=c("Overall"),
# indicate the total eagle minutes observed for each strata
  EMin=c(60),
# indicate the number of counts that were conducted (each count is presumed to be of the duration
# indicated for CntHr as defined above)
  nCnt=c(168),
# the size of the area observed during counts (in square kilometers); (in this example the count area was
# a circular plot with an 800-m radius)
  CntKM2=c(pi*(800/1000)^2),
# the total daylight hours for each strata
  DayLtHr=c(Days*LtHrPerDay)
 )
# the ExpSvy data frame is now complete
```

```
# Indicate whether strata should be added during each simulation to get a grand total (AddTot<-TRUE)
# or, in the case where only one strata is considered (as with this example), not
 AddTot<-FALSE
```

CollisionModel.R:
(the text between the solid lines below may be copied directly into a text file and saved as CollisionModel.R)

```
### USFWS Collision Fatality Estimate - Basic Model ###

# define the path where additional required files are stored (R will need to find these files in order to call
# functions from them)
RPath<-"/Projects/Eagles/R"
# or, alternatively, use
# RPath<-"R"
 sapply(c("FatalFcns","DistFcns","rvsmry"),function(iFcn)
  invisible(source(paste(RPath,"/",iFcn,".R",sep="")))
 )

## Analysis Inputs ##

# we can designate whether we would prefer R show graph of plots or create a .jpg file
  PlotFile<-NULL
# or
# PlotFile<-"FatalPlot.jpg"

# set the upper confidence limit(s) to be examined
 UCI<-c(0.5,0.8,0.9,0.95,0.975)
# we can consider many to compare, or,
## UCI<-0.8

# load the rv package:
 require(rv)
# Set the number of simulations:
 nSim<-100000
 setnsims(nSim)
 getnsims()

### Survey Inputs ###
```

```
# Create a vector called cSvy with the row names from ExpSvy that will be used later
 nSvy<-nrow(ExpSvy)
 cSvy<-(rownames(ExpSvy))

# Calculate the fatalities and store as a temporary object
 tmp<-with(ExpSvy,mapply(simFatal,EMin=EMin,nCnt=nCnt,CntHr=CntHr,
  CntKM2=CntKM2,DayLtHr=DayLtHr,HzKM2=HzKM2,
  SIMPLIFY=FALSE
 ))

# R code to get the survey specific simulations into an rv vector.
 Fatalities<-rvnorm(nSvy)
 Exp<-data.frame(Mean=rep(NA,nSvy),SD=NA,row.names=cSvy)
 for(i in 1:nSvy){
# i<-1
  Fatalities[i]<-tmp[[i]]
  Exp[i,]<-attr(tmp[[i]],"Exp")
 }
 rm(tmp)
 names(Fatalities)<-cSvy

# Summarize the surveys, including a total if indicated in the data file
 nSvy<-length(Fatalities)
 if(is.null(nSvy))nSvy<-1
 FatalStats<-RVSmry(cSvy,Fatalities,probs=UCI)
 if(AddTot){
  FatalStats<-rbind(
   FatalStats,
   RVSmry("Total",sum(Fatalities),probs=UCI)
  )
 }

# Look at the results
 cat(cProject,"\n")
 print(nTurbine)
 print(ExpSvy)
 print(Exp,digits=3)
 print(FatalStats,digits=2)

# Plots
 nPlot<-nSvy+as.integer(AddTot)
 nCol<-floor(sqrt(nPlot))
```

```
 nRow<-ceiling(nPlot/nCol)
 xlim<-range(rvrange(Fatalities))

 if(!is.null(PlotFile))jpeg(PlotFile)
 par(mfrow=c(nRow,nCol))
 for(iPlot in 1:nSvy){
# iPlot<-1
  plotFatal(Fatalities[iPlot],probs=UCI,
#  xlim=xlim,add=FALSE,  # uncomment this line to put the graphs for all of the strata
# on the same scale
  main=cSvy[iPlot])
 }
 if(AddTot)plotFatal(sum(Fatalities),main="Total")
 if(!is.null(PlotFile))dev.off()
```

DistFcns.R:

```
# gamma with mean and variance
 rGamma<-function(n=10000,mn=1,sd=1){
  x<-if(length(sd)==1){
   if(sd==0.0){
    rep(mn,length=n)
   } else {
    a<-(mn/sd)^2
    s<-sd^2/mn
    rgamma(n,a,scale=s)
   }
  } else {
   mxlen<-max(n,length(mn),length(sd))
   mxlen<-min(n,mxlen)
   mn<-rep(mn,length=mxlen)
   sd<-rep(sd,length=mxlen)
   x<-mapply(function(mn,sd){
    if(sd==0){
     mn
    } else {
     a<-(mn/sd)^2
     s<-sd^2/mn
     rgamma(1,a,scale=s)
```

```
   }
  },mn,sd)
 }
 return(x)
}

pGamma<-function(q,mn=1,sd=1){
 pval<-ifelse(sd==0,ifelse(q==mn,1,0),{
  a<-(mn/sd)^2
  s<-sd^2/mn
  pgamma(q,a,scale=s)
 })
 return(pval)
}

dGamma<-function(q,mn=1,sd=1){
 d<-if(length(sd)==1){
  if(sd==0.0){
   ifelse(q==mn,1,0)
  } else {
   a<-(mn/sd)^2
   s<-sd^2/mn
   dgamma(q,a,scale=s)
  }
 } else {
  mxlen<-max(q,length(mn),length(sd))
  q<-rep(q,length=mxlen)
  mn<-rep(mn,length=mxlen)
  sd<-rep(sd,length=mxlen)
  x<-mapply(function(q,mn,sd){
   if(sd==0){
    mn
   } else {
    a<-(mn/sd)^2
    s<-sd^2/mn
    dgamma(q,a,scale=s)
   }
  },mn,sd)
 }
 return(d)
}
```

```
qGamma<-function(p,mn=1,sd=1){
 a<-(mn/sd)^2
 s<-sd^2/mn
 q<-qgamma(p,shape=a,scale=s)
 return(q)
}

rNBinom<-function(n,mu=1,od=0.0){
 if(od==0.0){
  rpois(n,mu)
 } else {
  rnbinom(n,mu,size=mu/od)
 }
}


rBinom<-function(nSim=1,p,Conc,SD=sqrt(p*(1-p)*Conc)){
 ifelse(SD==0,p,{
  SD<-ifelse(SD^2>p*(1-p),{
    warning("SD greater than the maximum binomial SD.")
    SD<-sqrt(p*(1-p))
   },SD
  )
  Fac<-p*(1-p)/SD^2-1
  a<-p*Fac
  b<-(1-p)*Fac
  rbeta(1,a,b)
 })
}

rBinom<-function(n=1,p,conc,sd=sqrt(p*(1-p)*conc)){
 sd<-ifelse(sd^2>p*(1-p),{
   warning("sd greater than the maximum binomial sd.")
   sd<-sqrt(p*(1-p))
  },sd
 )
 x<-if(length(sd)==1){
  if(sd==0.0){
   rep(p,length=n)
  } else {
   fac<-p*(1-p)/sd^2-1
   a<-p*fac
```

```
    b<-(1-p)*fac
    rbeta(n,a,b)
   }
  } else {
   mxlen<-max(n,length(p),length(sd))
   mxlen<-min(n,mxlen)
   p<-rep(p,length=mxlen)
   sd<-rep(sd,length=mxlen)
   mapply(function(p,sd){
    if(sd==0){
     p
    } else {
     fac<-p*(1-p)/sd^2-1
     a<-p*fac
     b<-(1-p)*fac
     rbeta(1,a,b)
    }
   },p,sd)
  }
  return(x)
 }
```

FatalFcns.R

```
simFatal<-function(EMin,nCnt,
  CntHr=2,CntKM2=0.8^2*pi,HzKM2,DayLtHr=365.25*12,
  aPriExp=0.132,bPriExp=0.246,aPriCPr=1.191613,bPriCPr=176.6611){

  require(rv)

# Update the exposure prior
  aPostExp<-aPriExp+EMin
  bPostExp<-bPriExp+nCnt*CntHr*CntKM2

  Exp<-rvgamma(n=1,aPostExp,bPostExp)
  CPr<-if(bPriCPr==0){
   aPriCPr
  } else {
   rvbeta(n=1,aPriCPr,bPriCPr)
  }
```

```
 Fatalities<-DayLtHr*HzKM2*Exp*CPr
 attr(Fatalities,"Exp")<-c(Mean=rvmean(Exp),SD=rvsd(Exp))
 return(Fatalities)
}


plotFatal<-function(Fatalities,probs=0.8,xlim=NULL,col="red",add=FALSE,...){
 Names<-if(is.null(names(Fatalities))) 1:length(Fatalities) else
  names(Fatalities)
 Smry<-RVSmry(Names,Fatalities,probs=probs)
 ColIdx<-grepl("CI",colnames(Smry))
 CIs<-Smry[,ColIdx]

 if(!add){
  if(is.null(xlim)) xlim<-c(0,1.1*rvquantile(Fatalities,probs=0.99))
  rvhist(Fatalities,xlab="Collisions",ylab="Density",
   xlim=xlim,freq=FALSE,...)
 }
 lines(density(as.numeric(Fatalities[[1]],bw="sj")),col=if(add) col else "blue")
 abline(v=Smry$Mean,col=if(add) col else "black")
 abline(v=CIs,col=col)
 invisible(NULL)
}
```

---

rvsmry.R

```
RVSmry<-function(Names,Series,probs=c(0.5,0.05,0.95)){
 Smry<-data.frame(
  Mean=as.vector(rvmean(Series)),SD=as.vector(rvsd(Series)),
#   rvquantile(Series,probs=probs),
  matrix(rvquantile(Series,probs=probs),ncol=length(probs)),
  row.names=rownames(Names)
 )
 colnames(Smry)[2+1:length(probs)]<-paste("CI",format(100*probs),sep="")
 return(data.frame(Names,Smry))
}

rvsmry<-function(X,probs=c(Median=0.5,LCI90=0.025,UCI90=0.975)){
 Smry<-data.frame(Mean=rvmean(X),SD=rvsd(X),
  UCI=as.matrix(rvquantile(X,probs=probs))
 )
```

```
 colnames(Smry)[2+1:length(probs)]<-paste("CI",format(100*probs),sep="")
 return(Smry)
}
```

Once the analysis is complete, you may need to scroll up to see your results (highlighted below):

```
Patuxent Wind Company

First we review the model inputs that were used:
>  print(nTurbine)
[1] 18

# 18 turbines

>  print(ExpSvy)
        EMin nCnt   CntKM2 DayLtHr
Overall   60  168 2.010619    4380

# 60 eagle minutes, 168 counts, 2.01 km² count area, 4380 total daylight hours

>  print(Exp,digits=3)
         Mean     SD
Overall 0.089 0.0114

# 0.89 eagle minutes per kilometer per hour (with a standard deviation of
0.0114) is the mean exposure for the project

>  print(FatalStats,digits=2)
    Names  Mean    SD CI50.0 CI80.0 CI90.0 CI95.0 CI97.5
1 Overall 0.092 0.086  0.068   0.15    0.2   0.26   0.32

# Based on the model inputs, we would predict 0.15 eagle fatalities from
collisions with turbines per year using the 80% upper credible limit. This
means that with 80% certainty we would predict approximately 1 eagle fatality
every 6-7 years for this example project based on the model.
```