Appendix G R code for implementation of

#Load essential libraries in R#

library(ggplot2)

library(xlsx)

library(rstan)

#Load data from file#

bugslist<-c("Tetranychus\_urticae","Propylea\_quatuordecimpunctata")

inum<-1 #data index 1 for "Tetranychus” and 2 for "Propylea "#

bugname<-bugslist[inum]

loc\_dir<-"C:\\Users\\..." #local working directory that data files are found

setwd(loc\_dir)

DataBoot<-read.xlsx(paste0(bugname,".xlsx"),sheetIndex=1,header=FALSE,endRow = 10000)

DataBoot<-na.omit(DataBoot) #remove missing values

colnames(DataBoot)<- c("Temp","y",)

y<-c(DataBoot$y) #rensponse variable

temp<-c(DataBoot$Temp) #predictor measures temperature

N <-dim(DataBoot)[1] #sample size N

status<-ifelse((rdata$y==0),0,1) #status 0 is dev. rate is zero else 1

NP<-round(2\*(max(temp)-min(temp)),0) #sample size for predictions

xpred<-seq(min(temp),max(temp),length.out = NP) #NP values for predictor

#create dataset to apply to R-stan#

data <- list(N = N, y = y, x=temp, status= status, t=10.0, NP=NP, xpred=xpred)

#MC paramaters#

nch=4 #number of chains

iter=110000 #number of iterations

bin=100000 #size of burn-in

nth=1 #size of thinning

####BIERI MODEL####

#create initial values function

ttmax=rgamma(1,50,1)

brfun<- function (num\_id=1) {list( a=rbeta(1,1,1)/100,

tmin=ttmax\*rbeta(1,1,1),

b=1.5+0.5\* rbeta(1,1,1),

tmax= ttmax,

shape=runif(1,300,500),

sigmasq=rgamma(1,shape=100,rate=1))}

init<-lapply(1:nch, function(id) brfun(num\_id=id))

#Bieri model and Gaussian distribution#

curfile="stanbieri\_gaussian.stan"

param<-c("a","b","xmin","xmax","sigma","tmin","tdmax","tmax","dev") # use specific parameters of interest

# Bieri model and Inverse-Gamma distribution#

curfile="stanbieri\_igamma.stan"

param<-c("a","b","xmin","xmax","sigma","tmin","tdmax","tmax","dev")

# Bieri model and Zero-inflated Inverse-Gamma distribution#

curfile="stanbieri\_ziig.stan"

param<-c("a","b","xmin","xmax","sigma","tmin","tdmax","tmax","dev")

####BRIERE MODEL####

#create initial values function

ttmax=rgamma(1,50,1)

brfun<- function (num\_id=1) {list(

a=rgamma(1,shape=1e+2,rate=1e+1),

tmin=ttmax\*rbeta(1,1,1),

tmax= ttmax,

shape=100+10\*runif(1,num\_id,10),

sigmasq=rgamma(1,shape=100,rate=1))}

init<-lapply(1:nch, function(id) brfun(num\_id=id))

# Briere model and Gaussian distribution#

curfile="stanbriere\_gaussian.stan"

param<-c("alpha","sigma","tmin","tdmax","tmax","dev")

# Briere model and Inverse-Gamma distribution#

curfile="stanbriere\_igamma.stan"

param<-c("alpha","shape","tmin","tdmax","tmax","dev")

# Briere model and Zero-inflated Inverse-Gamma distribution#

curfile="stanbriere\_ziig.stan"

param<-c("alpha","shape","tmin","tdmax","tmax","dev")

####ANALYTIS MODEL####

#create initial values function

ttmax=rgamma(1,50,1)

brfun<- function (num\_id=1) {list( a=rgamma(1,shape=1e+2,rate=1e+1),

tmin=ttmax\*rbeta(1,1,1),

b=1.5+0.5\* rbeta(1,1,1),

tmax= ttmax,

nn=1+rbeta(1,shape1=1,shape2=1),

mm=0.07\*rbeta(1,shape1=1,shape2=1),

shape=runif(1,300,500),

sigmasq=rgamma(1,shape=100,rate=1))}

init<-lapply(1:nch, function(id) brfun(num\_id=id))

# Analytis model and Gaussian distribution#

curfile="stananalytis\_gaussian.stan"

param<-c("alpha","mm","sigma","nn","tmin","tdmax","tmax","dev")

# Analytis model and Inverse-Gamma distribution#

curfile="stananalytis\_igamma.stan"

param<-c("alpha","mm","shape","nn","tmin","tdmax","tmax","dev")

# Analytis model and Zero-inflated Inverse-Gamma distribution#

curfile="stananalytis\_ziig.stan"

param<-c("alpha","mm","shape","nn","tmin","tdmax","tmax","dev")

####LACTIN MODEL####

#create initial values function

del=runif(1,0.4,0.5)

th=rbeta(1,1,1)

ro=del-(1/100)\* rbeta(1,1,1)

brfun<- function (num\_id=1) {list(tmax= rgamma(1,50,1),

a=((ro/del))\*rbeta(1,1,1),

l=0.1\*rbeta(1,1,1),

del=del,

ro=ro,

shape=runif(1,300,500),

sigmasq=rgamma(1,shape=100,rate=1))}

init<-lapply(1:nch, function(id) brfun(num\_id=id))

# Lactin model and Gaussian distribution#

curfile="stanlactin\_gaussian.stan"

param<-c("l","ro","sigma","delta","a","tdmax","tmax","xmin","xmax","dev")

# Lactin model and Inverse-Gamma distribution#

curfile="stanlactin\_igamma.stan"

param<-c("l","ro","sigma","delta","a","tdmax","tmax","xmin","xmax","dev")

# Lactin model and Zero-inflated Inverse-Gamma distribution#

curfile="stanlactin\_ziig.stan"

param<- c("l","ro","shape","delta","a","tdmax","tmax","xmin","xmax","dev")

###HMC method via STAN###

mod\_stan <- stan(file = curfile,

init = iinit,

control = list(max\_treedepth = 11,adapt\_delta = 0.95),

algorithm = "NUTS",

data = data,

iter = iter,

chains = nch,

warmup=bin,

thin=nth,

verbose = TRUE)

result<- as.data.frame(summary(mod\_stan,pars=param)$summary)

print(result)

### Create posterior predictive plots ###

yypred\_lactin<-as.matrix(mod\_stan, pars="ypred")

#create quantiles#

ymeddata<-data.frame(x = data$xpred, y = apply(yypred\_lactin,2, median))

yquant1data<-data.frame(x = data$xpred, y = apply(yypred\_lactin,2, function(x) quantile(x, c(0.025)) ))

yquant2data<-data.frame(x = data$xpred, y = apply(yypred\_lactin,2, function(x) quantile(x, c(0.975)) ))

yquant3data<-data.frame(x = data$xpred, y = apply(yypred\_lactin,2, function(x) quantile(x, c(0.25)) ))

yquant4data<-data.frame(x = data$xpred, y = apply(yypred\_lactin,2, function(x) quantile(x, c(0.75)) ))

ndata\_lactin<-as.data.frame(data[2:4])

yndata\_lactin<-ymeddata

# convert grouping variables to factor

yndata\_lactin$Legend<-as.factor(yndata\_lactin$x)

yndata\_lactin$y1 <- yquant1data$y

yndata\_lactin$median <- ymeddata$y

yndata\_lactin$y2 <- yquant2data$y

yndata\_lactin$y3 <- yquant3data$y

yndata\_lactin$y4 <- yquant4data$y

Legend<-factor(data$xpred)

colval<-colorRampPalette(c("royalblue","red"))

scale\_color\_brewer(palette="Dark2")

ggplot(data = yndata\_lactin, aes(x = x, y = y)) +

geom\_point(data=ndata\_lactin, size = 3, aes(x = x, y = y, colour = "#000000"),alpha=1) +

geom\_line(aes(x = x, y = median, colour="median"), size=1)+

geom\_ribbon(aes(x = x,ymin=y1,ymax=y2),alpha=0.25, fill="#81DAF5")+

geom\_ribbon(aes(x = x,ymin=y3,ymax=y4),alpha=0.25, fill="#145fa7")+

labs(

x = expression("Temperature"~degree\*C),

y = expression("Developmental rate")

)+

scale\_colour\_manual("Legend",labels = c("Data","Median"),

values = c("black","red"),

guide = guide\_legend(override.aes = list(

linetype = c(rep("blank", 1), "solid"),

shape = c(16, NA,NA),

fill=c("#81DAF5", "#81DAF5"))))+

guides( color = guide\_legend(

order = 1,

override.aes = list(

color = c("black", "red"),

fill = c("white", "white"),

linetype = c("blank", "solid"),

shape = c(16, NA)))) +

theme\_bw() +

theme(text = element\_text(size=20),

legend.key = element\_rect(colour = NA, fill = NA),

legend.box.background = element\_blank())+

geom\_point(aes(x = x, y = y, size = "95% Credible Interval", shape = NA)) +

geom\_point(aes(x = x, y = y, size = "50% Credible Interval", shape = NA)) +

guides(size = guide\_legend(NULL,

order = 2,

override.aes = list(shape = c(15,15),

color = c("#145fa7","#81DAF5"),

size = c(6,6) )))+

theme(legend.position = "none")

#Calculate and record information criteria and record method elapsed time

library(loo)

ttime<-get\_elapsed\_time(mod\_stan)

array.stan <- as.array(mod\_stan)

for( ii in 1:ncol(array.stan) ){

tempo <- as.data.frame(array.stan[,ii,])

tempo$chain <- ii

tempo$iteration <- 1:nrow(tempo)

if ( ii==1 ) df.res <- tempo

else df.res <- rbind(df.res, tempo)

}

rm(tempo, array.stan, ii)

mean.Deviance <- mean(df.res$dev)

var.Deviance <- var(df.res$dev)

df.input<-as.data.frame(mod\_stan)

#Likelihood for Bieri model

mu\_val<-mean(df.input$a)\*temp-mean(df.input$a)\*mean(df.input$tmin)- exp(log(mean(df.input$b))\*(temp-mean(df.input$tmax)))

scalle<-(mean(df.input$shape)-1)\*mu\_val

log\_lik1<-array(dim=N)

log\_lik1<-with(df.input, dinvgamma(y[y>0], shape=mean(df.input$shape), rate=scalle, log = T) )0+log(1-colMeans((df.input[,paste0("th[",seq(1,(N-length(y[y==0]))),"]")])))

log\_lik1[(N-length(y[y==0])+1):N]<-log(colMeans((df.input[,paste0("th[",seq((N-length(y[y==0])+1),N),"]")])))

Deviance.mean<- -2\*sum(log\_lik1)

sscale<-array(dim=N)

llik<-array(dim=N)

mu<-with(df.input,

(mean(a)\*(temp[y>0])-mean(a)\*mean(tmin)-mean(b)^(temp[y>0]-mean(tmax))))

sscale<-ifelse(mu>0,(mean(df.input$shape)-1)\*mu,-1)

for (kk in 1:N){

if(y[kk]>0) llik[kk]<-dinvgamma(y[kk], shape=mean(df.input$shape), rate = sscale[kk], log = T) + log(1-mean(df.input[,paste0("th[",kk,"]")]))

else llik[kk]<-log(mean(df.input[,paste0("th[",kk,"]")]))

}

dev.m<--2\*sum(llik)

dev.m

mean(df.res$dev)

pd\_dic<-2\*sum(llik)+mean(df.input$dev)

#Likelihood for Briere model

test\_val<-ifelse(outer((mean(df.input$tmax)>temp),(mean(df.input$tmin)<temp), "&")[,1],exp(-mean(df.input$a)+log(temp)+log(temp-mean(df.input$tmin))+(1/2)\*log(mean(df.input$tmax)-temp)),0)

scalle<-(mean(df.input$shape)-1)\*test\_val;

log\_lik1<-array(dim=N)

#for Gaussian case

log\_lik1<-with(df.input, dnorm(y, mean=test\_val, sd=mean(df.input$sigma), log = T) )

#for Inverse-Gamma case

log\_lik1<-with(df.input, dinvgamma(y[y>0], shape=mean(df.input$shape), rate=scalle, log = T) )+log(1-colMeans((df.input[,paste0("th[",seq(1,(N-length(y[y==0]))),"]")])))

log\_lik1[(N-length(y[y==0])+1):N]<-log(colMeans((df.input[,paste0("th[",seq((N-length(y[y==0])+1),N),"]")])))

Deviance.mean<- -2\*sum(log\_lik1)

mu<-array(dim=N)

llik<-array(dim=N)

mu<-exp(-mean(df.input$a))\*temp\*(temp-mean(df.input$tmin))\*exp((1/2)\*log(mean(df.input$tmax)-temp))

sscale<-ifelse(mu>0,(mean(df.input$shape)-1)\*mu,-1)

for (kk in 1:N){

if(y[kk]>0) llik[kk]<-dinvgamma(y[kk], shape=mean(df.input$shape), rate = sscale[kk], log = T) + log(1-mean(df.input[,paste0("th[",kk,"]")]))

else llik[kk]<-log(mean(df.input[,paste0("th[",kk,"]")]))}

llik[is.na(llik)]<-0

dev.m<--2\*sum(llik)

pd\_dic<-2\*sum(llik)+mean(df.input$dev)

#Likelihood for Analytis model

test\_val<-ifelse(outer((mean(df.input$tmax)>temp),(mean(df.input$tmin)<temp), "&")[,1],exp(-mean(df.input$a)+mean(df.input$nn)\*log(temp-mean(df.input$tmin))+mean(df.input$mm)\*log(mean(df.input$tmax)-temp)),0)

scalle<-(mean(df.input$shape)-1)\*test\_val;

log\_lik1<-array(dim=N)

#for Gaussian case

log\_lik1<-with(df.input, dnorm(y, mean=test\_val, sd=mean(df.input$sigma), log = T) )

#for Inverse-Gamma case

log\_lik1<-with(df.input, dinvgamma(y[y>0], shape=mean(df.input$shape), rate=scalle, log = T) )+log(1-colMeans((df.input[,paste0("th[",seq(1,(N-length(y[y==0]))),"]")])))

log\_lik1[(N-length(y[y==0])+1):N]<-log(colMeans((df.input[,paste0("th[",seq((N-length(y[y==0])+1),N),"]")])))

Deviance.mean<- -2\*sum(log\_lik1)

#try to calculate DIC with posterior means for parameters

mu<-array(dim=N)

llik<-array(dim=N)

mu<-exp(-mean(df.input$a)+ mean(df.input$nn)\*log(temp-mean(df.input$tmin))+mean(df.input$mm)\*log(mean(df.input$tmax)-temp))

sscale<-ifelse(mu>0,(mean(df.input$shape)-1)\*mu,-1)

for (kk in 1:N){

#sscale[kk] = mean(ssxale$sscale[,kk])

if(y[kk]>0) llik[kk]<-dinvgamma(y[kk], shape=mean(df.input$shape), rate = sscale[kk], log = T) + log(1-mean(df.input[,paste0("th[",kk,"]")]))

else llik[kk]<-log(mean(df.input[,paste0("th[",kk,"]")]))

}

llik[is.na(llik)]<-0

dev.m<--2\*sum(llik)

dev.m

pd\_dic<-2\*sum(llik)+mean(df.input$dev)

dev.m+2\*pd\_dic

#Likelihood for Lactin model

mu\_val<--mean(df.input$l)+exp(mean(df.input$ro)\*(temp))-mean(df.input$a)\*exp(mean(df.input$del)\*(temp))

scale2<-(mean(df.input$shape)-1)\*mu\_val

log\_lik1<-array(dim=N)

log\_lik1<-with(df.input, dinvgamma(y[y>0], shape=mean(df.input$shape), rate=scale2, log = T) )+log(1-colMeans((df.input[,paste0("th[",seq(1,(N-length(y[y==0]))),"]")])))

log\_lik1[(N-length(y[y==0])+1):N]<-log(colMeans((df.input[,paste0("th[",seq((N-length(y[y==0])+1),N),"]")])))

Deviance.mean<- -2\*sum(log\_lik1)

mu<-array(dim=N)

llik<-array(dim=N)

mu<-(-mean(df.input$l)+exp(mean(df.input$ro)\*(temp[y>0]))-mean(df.input$a)\*exp(mean(df.input$del)\*(temp[y>0])))

sscale<-ifelse(mu>0,(mean(df.input$shape)-1)\*mu,-1)

for (kk in 1:N){

if(y[kk]>0) llik[kk]<-dinvgamma(y[kk], shape=mean(df.input$shape), rate = sscale[kk], log = T) + log(1-mean(df.input[,paste0("th[",kk,"]")]))

else llik[kk]<-log(mean(df.input[,paste0("th[",kk,"]")]))

}

llik[is.na(llik)]<-0

dev.m<--2\*sum(llik)

pd\_dic<-2\*sum(llik)+mean(df.input$dev)

# effective number of parameters

pD <- mean.Deviance - Deviance.mean

DIC\_res<-c(mean.Dev=mean.Deviance, Dev.mean=Deviance.mean, pD=mean.Deviance-Deviance.mean, DIC=pD+mean.Deviance, DIC\_est=mean.Deviance+var.Deviance/2, pd\_dic=pd\_dic, DIC\_new=dev.m+2\*pd\_dic)

params<-

length(param[-which(param%in%c("alpha","xmin","xmax","tdmax","lp\_\_","dev","a"))])

AIC\_BIC<-c(AIC=min(df.res$dev)+2\*params, BIC=min(df.res$dev)+params\*log(length(temp)))

lik\_lactin<-extract\_log\_lik( mod\_stan, parameter\_name = "log\_lik",

merge\_chains = TRUE)

waic\_loo<-rbind(as.data.frame(loo(lik\_lactin)$estimates),

as.data.frame(waic(lik\_lactin)$estimates))

result<-as.data.frame(summary(mod\_stan,pars=param)$summary)

des\_res<-data.frame(c(bugname,nch,iter,bin,nth,nch\*(iter-bin)/nth),row.names = c("Insect Name","Number of chains","iterrations","Burn in","Thinning ratio","sample size"))

#save results in excel file with name as in ccfile

ccfile<-paste(fold,"\\",bugname,curfile,"information criteria",".xls",sep="")

write.xlsx(des\_res,ccfile,sheetName="descriptive") #write the number of the itterations

write.xlsx(result,ccfile,append = TRUE, sheetName="results")

write.xlsx(ttime/60,ccfile,append = TRUE, sheetName="elapsed\_time")

write.xlsx(AIC\_BIC,ccfile,append = TRUE, sheetName="AIC\_BIC")

write.xlsx(DIC\_res,ccfile,append = TRUE, sheetName="DIC")

write.xlsx(waic\_loo,ccfile,append = TRUE, sheetName=" Waic\_loo ")