Supplementary File: Syntax for Rshiny application

```
#### New ACTIVE infections A entry

library(deSolve); library(rootSolve); library(shiny); library(dplyr); library(rsconnect);

# Define UI for miles per gallon app ----
ui <- pageWithSidebar(
  # App title ----
  headerPanel("Pre-semester Testing"),
  # Sidebar panel for inputs ----
  sidebarPanel(
    # slider input
    sliderInput("N.1","On-campus population",7500,min=50,max=50000),
    sliderInput("N.2","Off-campus population",17500,min=50,max=50000),
    sliderInput("max.beds","Isolation bed capacity",500,min=10,max=10000),
    sliderInput("recovery.time","Days in isolation",11,min=7,max=21),
    sliderInput("p.detected.1","Proportion of on-campus infections who isolate",1/3,min=0,max=1),
    sliderInput("p.detected.2","Proportion of off-campus infections who isolate",1/3,min=0,max=1),
    sliderInput("R0","R0",2,min=0.5,max=5),
    sliderInput("p.A","Proportion infectious at baseline",0.02,min=0.001,max=0.10),
    sliderInput("p.R","Proportion immune at baseline",0.10,min=0,max=0.99),
    sliderInput("sensitivity.test","Baseline test sensitivity",0.90,min=0.50,max=0.99),
    sliderInput("exposure.period","Exposure period (days)",3,min=2,max=7),
    sliderInput("infectious.period.A","Infectious period for asymptomatic/undetected cases
(days)",14,min=7,max=21),
    sliderInput("infectious.period.I","Infectious period for symptomatic cases before detection/isolation
(days)",3,min=1,max=14),
    sliderInput("time.max","Semester length (days)",90,min=30,max=120)
  ),
  # Main panel for displaying outputs ----
  mainPanel(
    tabsetPanel(
      tabPanel("Active infections under different pre-semester testing strategies",plotOutput("plot1"))))
)


server=function(input,output)
{
  beta.matrix <- function(n.groups,beta.input,epsilon){
    beta.mat <- matrix(0, nrow=n.groups, ncol=n.groups)
    diag(beta.mat)=beta.input
    beta.mat[upper.tri(beta.mat)]=epsilon*beta.input[1]  ##fill up upper triangular part
    beta.mat[lower.tri(beta.mat)]=epsilon*beta.input[2]  ##fill up lower triangular part
    beta.mat=beta.mat/(1+(n.groups-1)*epsilon)
    return(beta.mat)
  }
```

```
  seaiqr_groupmod = function(t, y, parms){
    par= as.list(c(y,parms))
    with(par,{
      S = y[1:n.groups]
      E = y[(n.groups + 1):(2*n.groups)]
      A = y[(2*n.groups + 1):(3*n.groups)]
      I = y[(3*n.groups + 1):(4*n.groups)]
      Q = y[(4*n.groups + 1):(5*n.groups)]
      R = y[(5*n.groups + 1):(6*n.groups)]
      N = parms[["N"]]
      R0 = parms[["R0"]]
      sigma = parms[["sigma"]]
      phi = parms[["phi"]]
      gamma = parms[["gamma"]]
      rho = parms[["rho"]]
      alpha = parms[["alpha"]]
      epsilon = parms[["epsilon"]]
      #beta = beta.matrix(n.groups,R0*((1-alpha)*phi+alpha*gamma),epsilon)
      beta = beta.matrix(n.groups,R0*((1-alpha)*1/14+alpha*1/3),epsilon)
      dS = -beta%*%((A+I)/N)*S
      dE = beta%*%((A+I)/N)*S - sigma*E
      dA = (1-alpha)*sigma*E - phi*A
      dI = alpha*sigma*E - gamma*I
      dQ = gamma*I - rho*Q
      dR = phi*A + rho*Q
      res = c(dS, dE, dA, dI, dQ, dR)
      list(res)
    }
    )
  }


#input=list(N.1=7500,N.2=17500,time.max=90,R0=2.5,p.A=.02,
#         interaction.rate=.10,exposure.period=3,infectious.period.A=14,infectious.period.I=3,
#         recovery.time=11,p.detected.1=.5,p.detected.2=.25,max.beds=500,
#         sensitivity.test=.90,p.R=.10)

output$plot1 = renderPlot({
  N.group=c(input$N.1,input$N.2);
  N.total=sum(N.group);
  n.groups=length(N.group);
  time.max=input$time.max;
  R0=c(input$R0,input$R0);
  interaction.rate=0.1;
  exposure.period=c(input$exposure.period,input$exposure.period);
  infectious.period.A=c(input$infectious.period.A,input$infectious.period.A);
  infectious.period.I=c(input$infectious.period.I,input$infectious.period.I);
  recovery.time=c(input$recovery.time,input$recovery.time);
```

```
  p.detected=c(input$p.detected.1,input$p.detected.2);
  max.beds=input$max.beds
  sensitivity.test=c(input$sensitivity.test,input$sensitivity.test);
  p.R=c(input$p.R,input$p.R);

  p.I=c(0,0); p.E=c(0,0); p.Q=c(0,0);

  # tests = 0
  p.A = input$p.A*(1-sensitivity.test)^c(0,0)
  p.S = 1 - p.E - p.A - p.I - p.Q - p.R

  # follow-up phase
  parameters = list(n.groups=length(N.group),N=N.group, R0=R0, sigma=1/exposure.period,
phi=1/infectious.period.A,
            gamma=1/infectious.period.I, rho=1/recovery.time, alpha=p.detected, epsilon=interaction.rate)
  start = c(S=p.S*N.group,E=p.E*N.group,A=p.A*N.group,I=p.I*N.group,Q=p.Q*N.group,R=p.R*N.group)
  time=seq(0,time.max,by=1);

  # model
  out=data.frame(ode(y=start,times=time,func=seaiqr_groupmod,parms=parameters))

  # combining output...
  y=round(out[,-1],0)
  S = apply(y[1:n.groups],1,sum)
  E = apply(y[(n.groups + 1):(2*n.groups)],1,sum)
  A = apply(y[(2*n.groups + 1):(3*n.groups)],1,sum)
  I = apply(y[(3*n.groups + 1):(4*n.groups)],1,sum)
  Q = apply(y[(4*n.groups + 1):(5*n.groups)],1,sum)
  R = apply(y[(5*n.groups + 1):(6*n.groups)],1,sum)

  # new infections + active infections
  new.active.infections=numeric(time.max+1)
  for(j in 2:(time.max+1)) new.active.infections[j]=(S[j-1]+E[j-1])-(S[j]+E[j])
  max.new.active.infections.1=ceiling(max(new.active.infections)-1)
  active.infections.1=A+I+Q
  max.active.infections.1=ceiling(max(active.infections.1)-1)
  time.to.peak.1=min(time[which(active.infections.1>max.active.infections.1)])

  # computing maximum beds needed and days to trigger
  max.beds.1=ceiling(max(Q)-1)
  p.max.beds.1=100*round(max.beds.1/N.total,3)
  capacity.beds.1=round((N.total-max.beds.1)/N.total,2)
  days.to.trigger.1 = min(time[which(Q>=max.beds)])

  plot(x=time, y=active.infections.1, col="red",type="n",
     ylab="Active infections", xlab="Days since start of semester", ylim=c(0,max.active.infections.1*1.25),
     main = bquote(R[0] ~ "=" ~ .(R0[1])))
  lines(x=time, y=active.infections.1, col="red",lwd=2,lty=1)
```

```
  # tests = 1
  p.A = input$p.A*(1-sensitivity.test)^c(1,1)
  p.S = 1 - p.E - p.A - p.I - p.Q - p.R

  # follow-up phase
  parameters = list(n.groups=length(N.group),N=N.group, R0=R0, sigma=1/exposure.period,
phi=1/infectious.period.A,
            gamma=1/infectious.period.I, rho=1/recovery.time, alpha=p.detected, epsilon=interaction.rate)
  start = c(S=p.S*N.group,E=p.E*N.group,A=p.A*N.group,I=p.I*N.group,Q=p.Q*N.group,R=p.R*N.group)
  time=seq(0,time.max,by=1);

  # model
  out=data.frame(ode(y=start,times=time,func=seaiqr_groupmod,parms=parameters))

  # combining output...
  y=round(out[,-1],0)
  S = apply(y[1:n.groups],1,sum)
  E = apply(y[(n.groups + 1):(2*n.groups)],1,sum)
  A = apply(y[(2*n.groups + 1):(3*n.groups)],1,sum)
  I = apply(y[(3*n.groups + 1):(4*n.groups)],1,sum)
  Q = apply(y[(4*n.groups + 1):(5*n.groups)],1,sum)
  R = apply(y[(5*n.groups + 1):(6*n.groups)],1,sum)

  # new infections + active infections
  new.active.infections=numeric(time.max+1)
  for(j in 2:(time.max+1)) new.active.infections[j]=(S[j-1]+E[j-1])-(S[j]+E[j])
  max.new.active.infections.2=ceiling(max(new.active.infections)-1)
  active.infections.2=A+I+Q
  max.active.infections.2=ceiling(max(active.infections.2)-1)
  time.to.peak.2=min(time[which(active.infections.2>max.active.infections.2)])

  # computing maximum beds needed and days to trigger
  max.beds.2=ceiling(max(Q)-1)
  p.max.beds.2=100*round(max.beds.2/N.total,3)
  capacity.beds.2=round((N.total-max.beds.2)/N.total,2)
  days.to.trigger.2 = min(time[which(Q>=max.beds)])

  lines(x=time, y=active.infections.2, col="blue",lwd=2,lty=2)

  # tests = 2
  p.A = input$p.A*(1-sensitivity.test)^c(2,2)
  p.S = 1 - p.E - p.A - p.I - p.Q - p.R

  # follow-up phase
  parameters = list(n.groups=length(N.group),N=N.group, R0=R0, sigma=1/exposure.period,
phi=1/infectious.period.A,
            gamma=1/infectious.period.I, rho=1/recovery.time, alpha=p.detected, epsilon=interaction.rate)
```

```
  start = c(S=p.S*N.group,E=p.E*N.group,A=p.A*N.group,I=p.I*N.group,Q=p.Q*N.group,R=p.R*N.group)
  time=seq(0,time.max,by=1);

  # model
  out=data.frame(ode(y=start,times=time,func=seaiqr_groupmod,parms=parameters))

  # combining output...
  y=round(out[,-1],0)
  S = apply(y[1:n.groups],1,sum)
  E = apply(y[(n.groups + 1):(2*n.groups)],1,sum)
  A = apply(y[(2*n.groups + 1):(3*n.groups)],1,sum)
  I = apply(y[(3*n.groups + 1):(4*n.groups)],1,sum)
  Q = apply(y[(4*n.groups + 1):(5*n.groups)],1,sum)
  R = apply(y[(5*n.groups + 1):(6*n.groups)],1,sum)

  # new infections + active infections
  new.active.infections=numeric(time.max+1)
  for(j in 2:(time.max+1)) new.active.infections[j]=(S[j-1]+E[j-1])-(S[j]+E[j])
  max.new.active.infections.3=ceiling(max(new.active.infections)-1)
  active.infections.3=A+I+Q
  max.active.infections.3=ceiling(max(active.infections.3)-1)
  time.to.peak.3=min(time[which(active.infections.3>max.active.infections.3)])

  # computing maximum beds needed and days to trigger
  max.beds.3=ceiling(max(Q)-1)
  p.max.beds.3=100*round(max.beds.3/N.total,3)
  capacity.beds.3=round((N.total-max.beds.3)/N.total,2)
  days.to.trigger.3 = min(time[which(Q>=max.beds)])

  lines(x=time, y=active.infections.3, col="green",lwd=2,lty=3)

  legend("top",lty=c(1,2,3), col=c("red","blue","green"), bty="n",
       c(paste("# NAT = ",0,": Days to trigger = ",days.to.trigger.1,sep=""),
         paste("# NAT = ",1,": Days to trigger = ",days.to.trigger.2,sep=""),
         paste("# NAT = ",2,": Days to trigger = ",days.to.trigger.3,sep="")))
})
}

shinyApp(ui, server)
```