Supporting Information for: Estimates of the ongoing need for social distancing and control measures post-"lockdown" from trajectories of COVID-19 cases and mortality.

Mike Lonergan and James Chalmers

Division of Molecular and Clinical Medicine, School of Medicine, University of Dundee, Dundee,

UK

Corresponding author: Mike Lonergan, Division of Molecular and Clinical Medicine, University of Dundee,

Ninewells Hospital and Medical School, Dundee, DD1 9SY, m.lonergan@dundee.ac.uk

Contents:

1) Details of Supplementary Table S1: Parameter estimates from models of exponential trajectories within COVID-19 outbreaks for 87 countries, though most have very limited or imprecise data

2) Supplementary plots of trajectories, similar to those in Figure 1 of the paper, for all the countries listed in Supplementary table 1

3) The R code used for the calculations.

1) **Structure of Supplementary Table S1**: Parameter estimates from models of exponential trajectories within COVID-19 outbreaks for 87 countries, though most have very limited or imprecise data.

The data are in a separate .csv file.

Column headings start with c for cases or d for deaths; then b for before lockdown or a for after locking down. Those containing the bounds of 95% CI end in l or u.

Cores of column names:

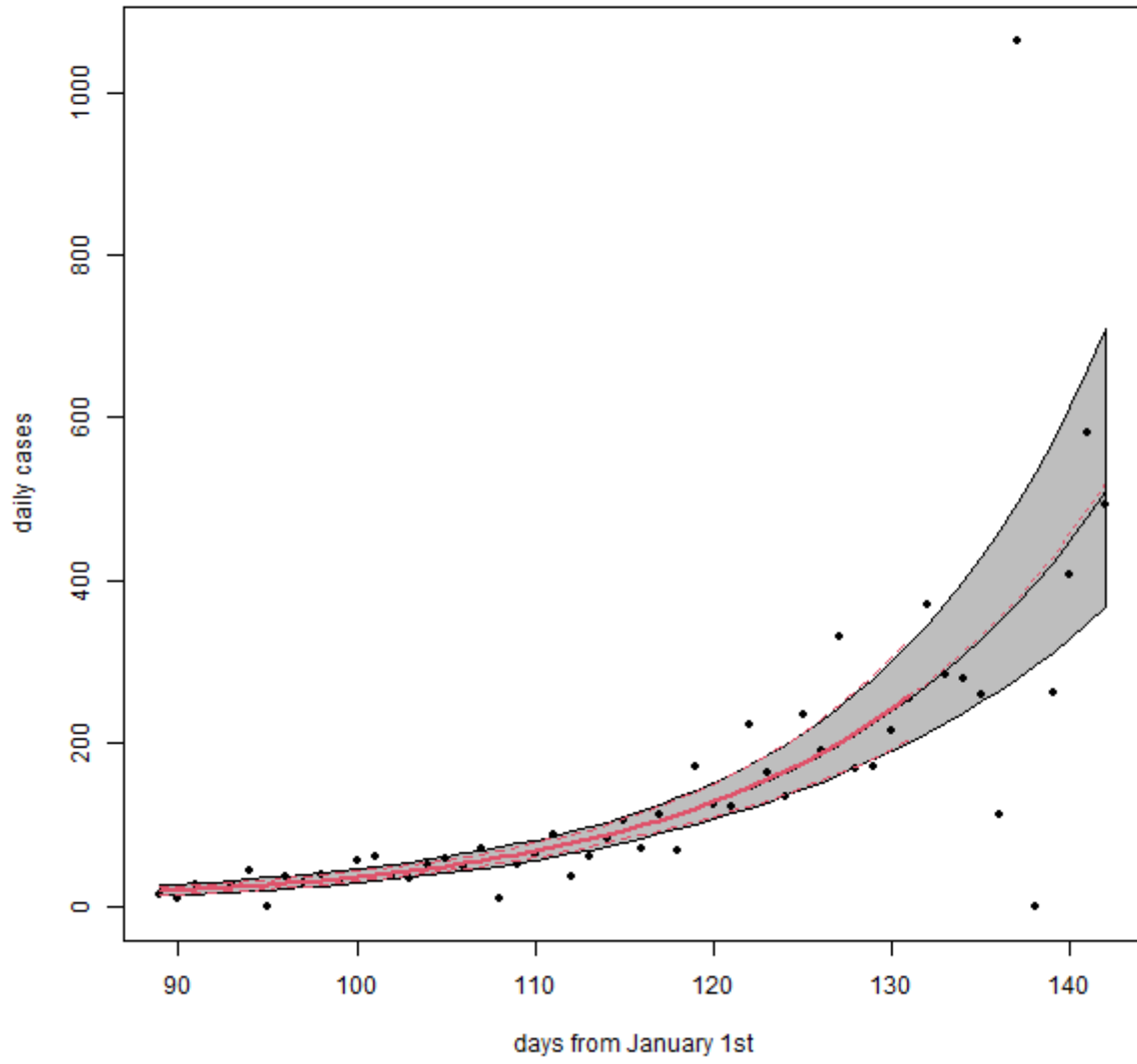|	
casemodel	"?" indicates doubt about the case models
deathmodel	"?" indicates doubt about the mortality models
cases	total number of recorded cases
deaths	total number of deaths recorded
**firstday	day of year of start of exponential period
**lastday	day of year of end of exponential period
**slope	slope of exponential model
**se	standard error of slope of exponential model
**dble	doubling time estimate
*ahalve	halving time estimate (included for convenience)
**r0	estimate of R0
sloperat	ratio of slopes of the two exponential periods
*propunlock	proportion of time not on lockdown
*prf	proportion of reclaimable of former behavior (based on R0s)

2) Plots for individual countries

The dots are numbers of new cases or deaths reported to the ECDC for each day up to May 22nd 2020. Each grey pipe shows the 95% confidence interval around a smooth trajectory (black line) estimated by a generalised additive model. In red are increasing and decreasing exponential patterns (mean & 95% CI) fitted to subsets of the data. Details of the models are in the main text.

Only countries for which at least one exponential period was identified are shown.

For some countries the approach produces convincing results, for others, especially where there is little data, the results are much less convincing. Adjusting the constraints and model selection approach will, subjectively, improve some results but worsen others. It was felt more important to use a consistent method than optimise the results for each individual country, and the poor fits are included to give a fair impression of the approach and its limitations.

While visually very strong, the grey pipe around each GAM result should not be mistaken for the truth: it is also the output of a model with a set of assumptions.
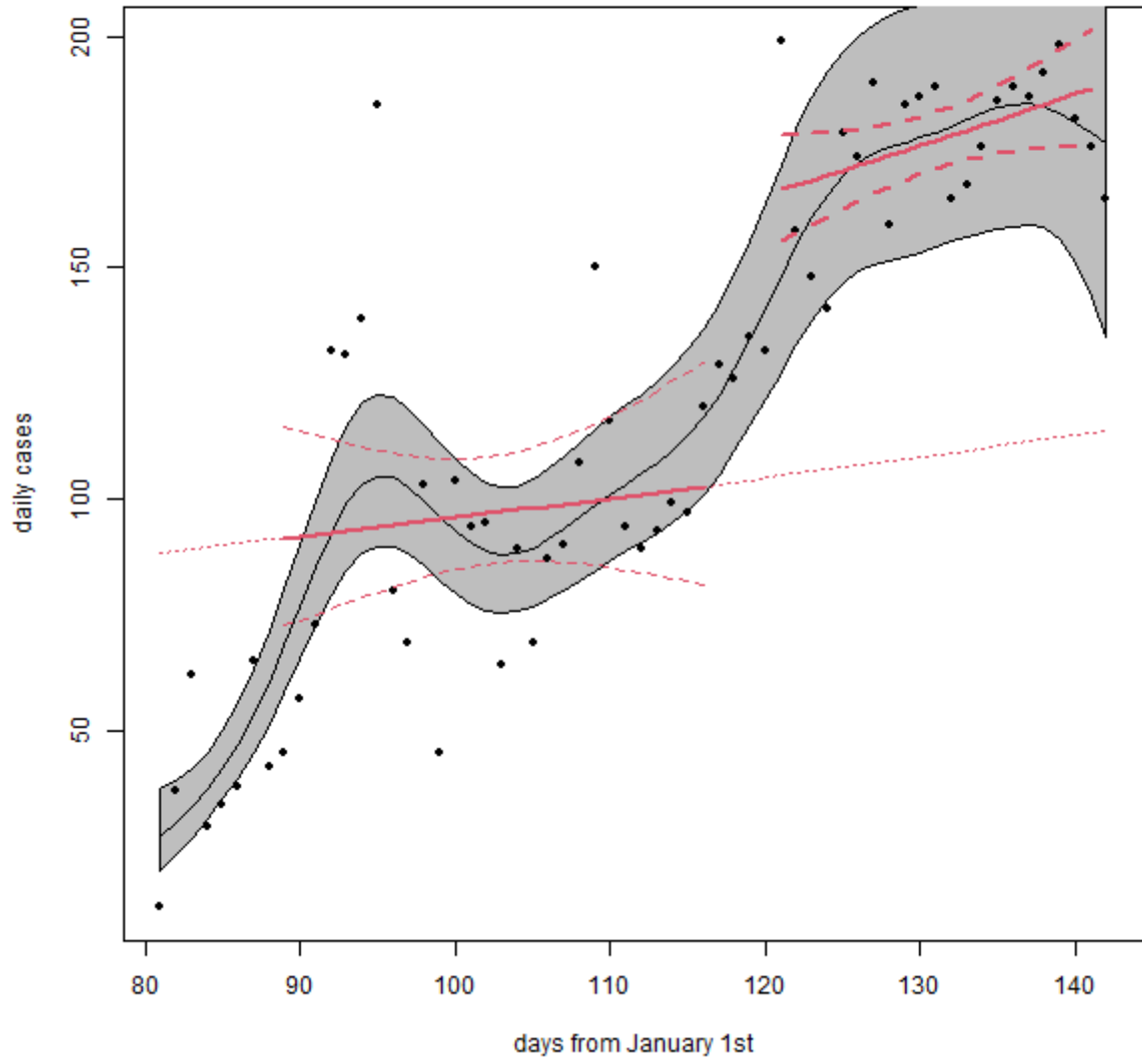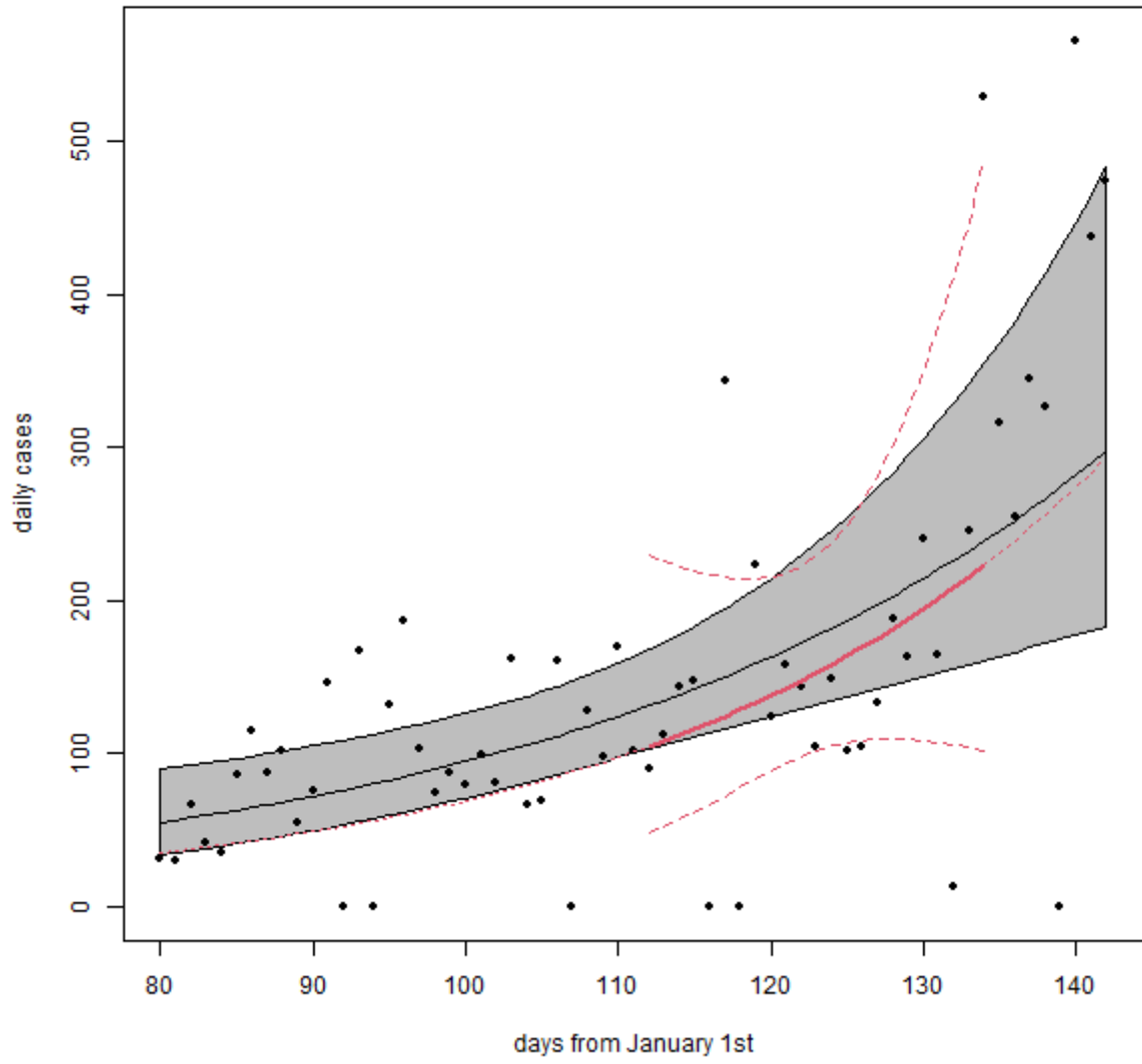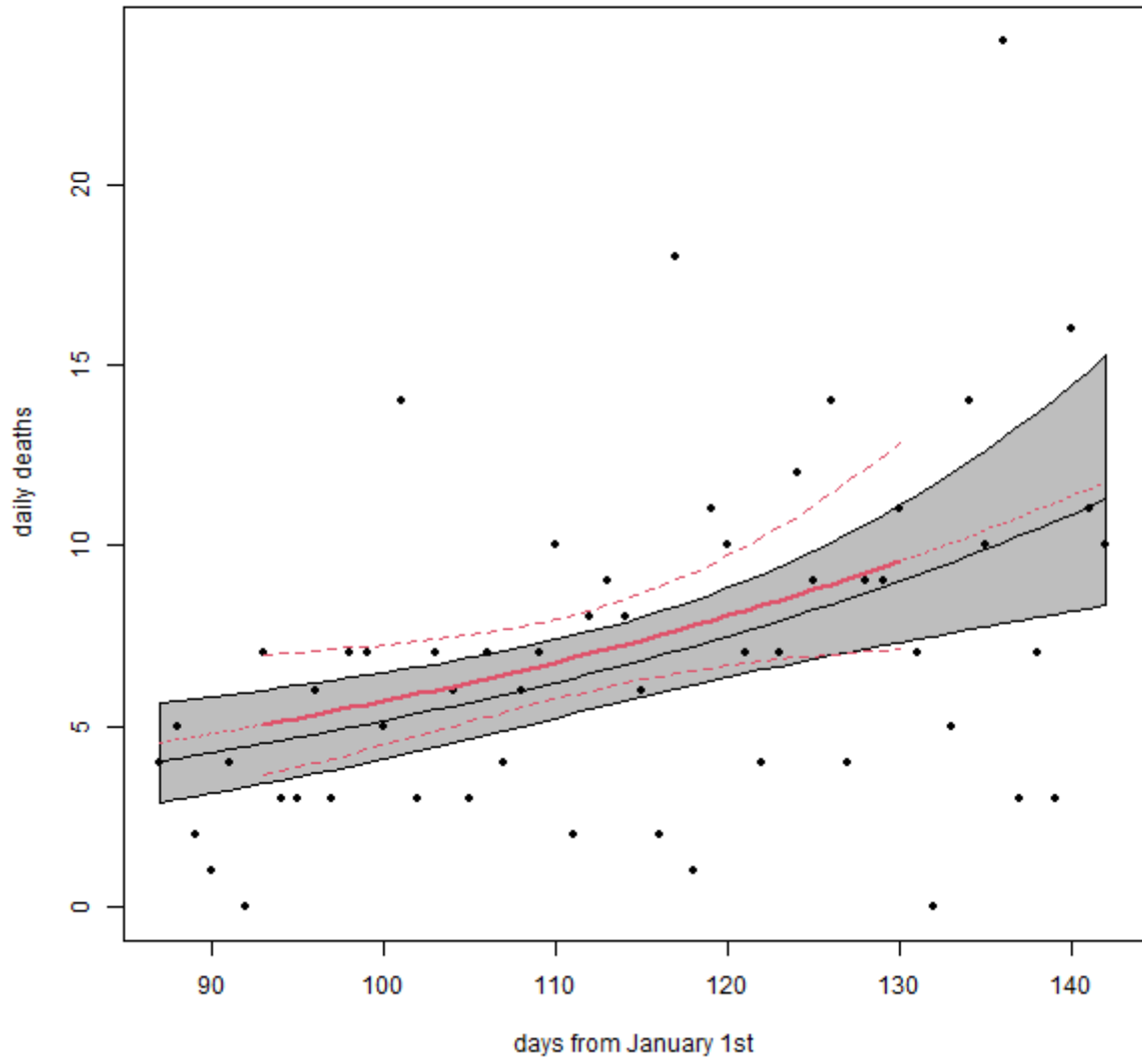
Afghanistan

daily cases

days from January 1st

Afghanistan

days from January 1st
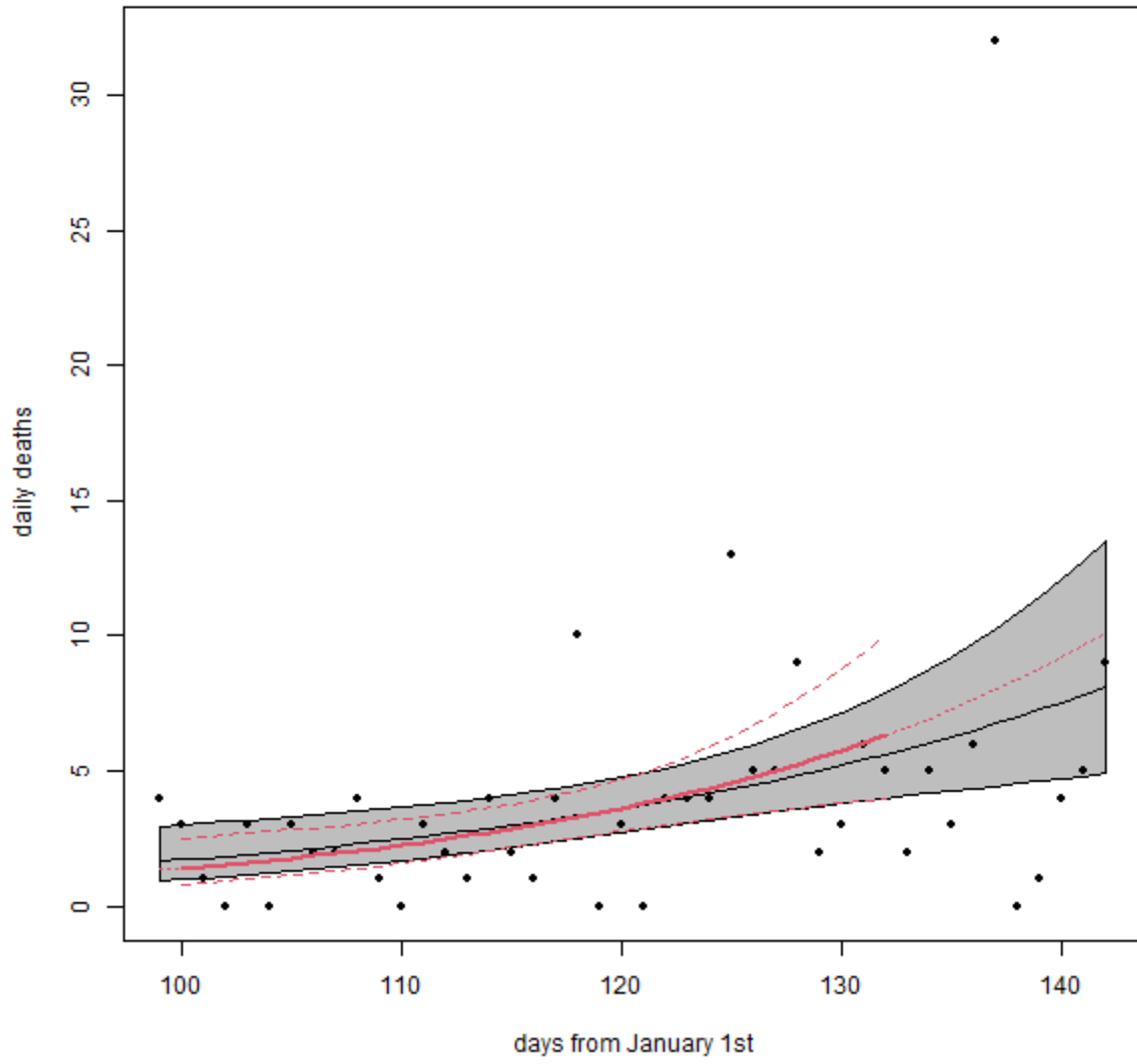
Algeria

Argentina

Argentina

Armenia

days from January 1st

Austria

Austria

daily deaths

days from January 1st

Azerbaijan

daily cases

days from January 1st
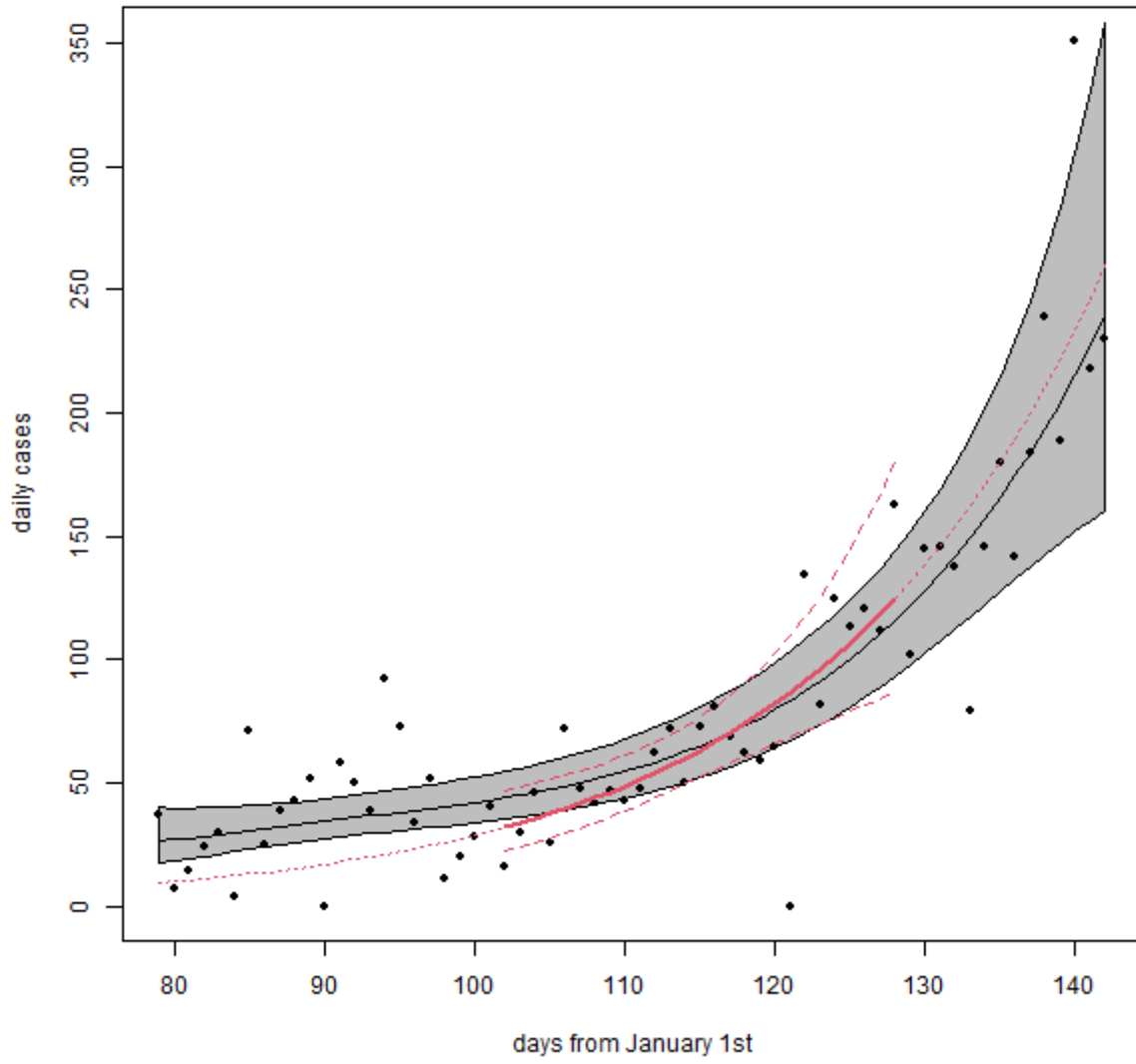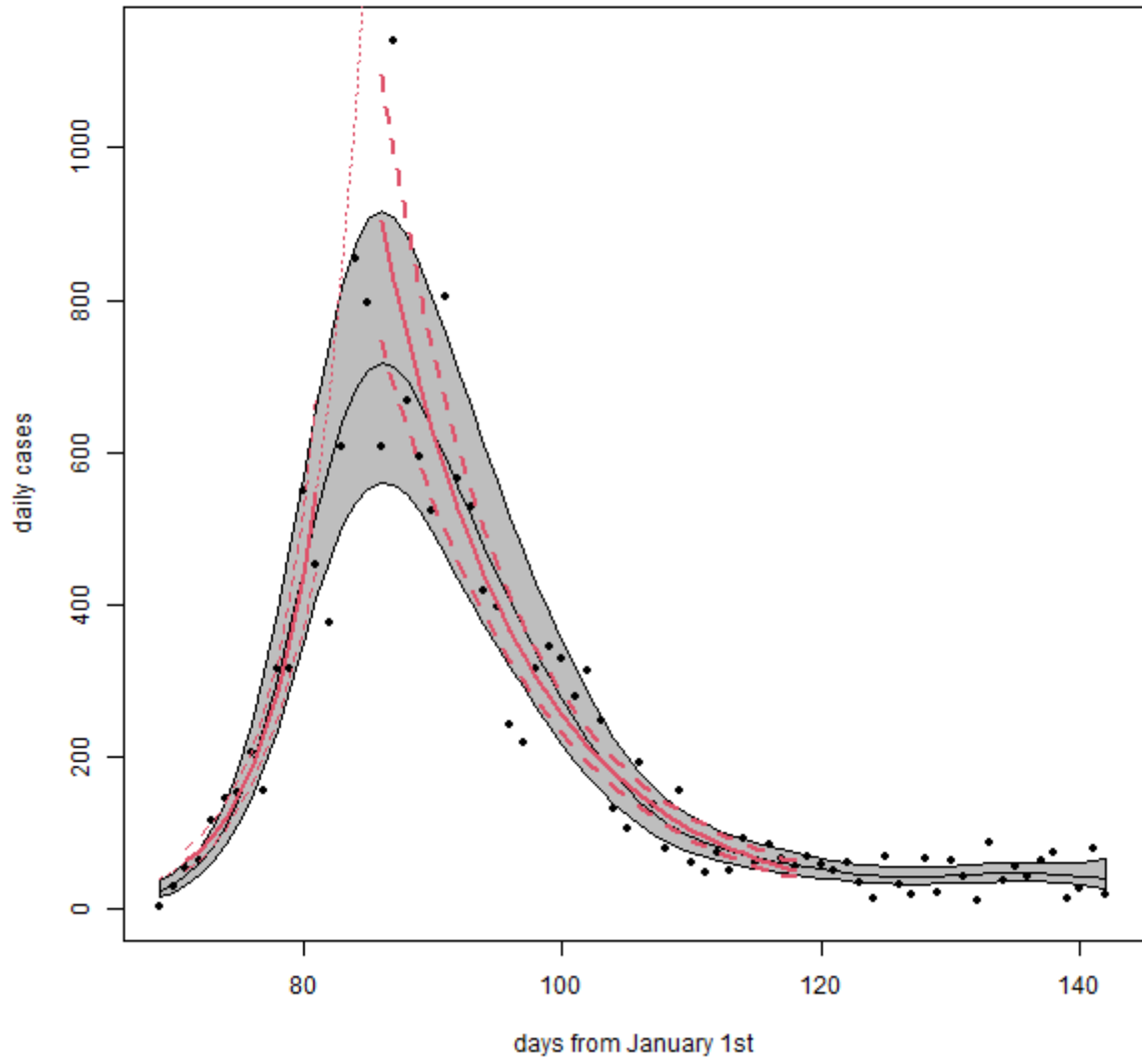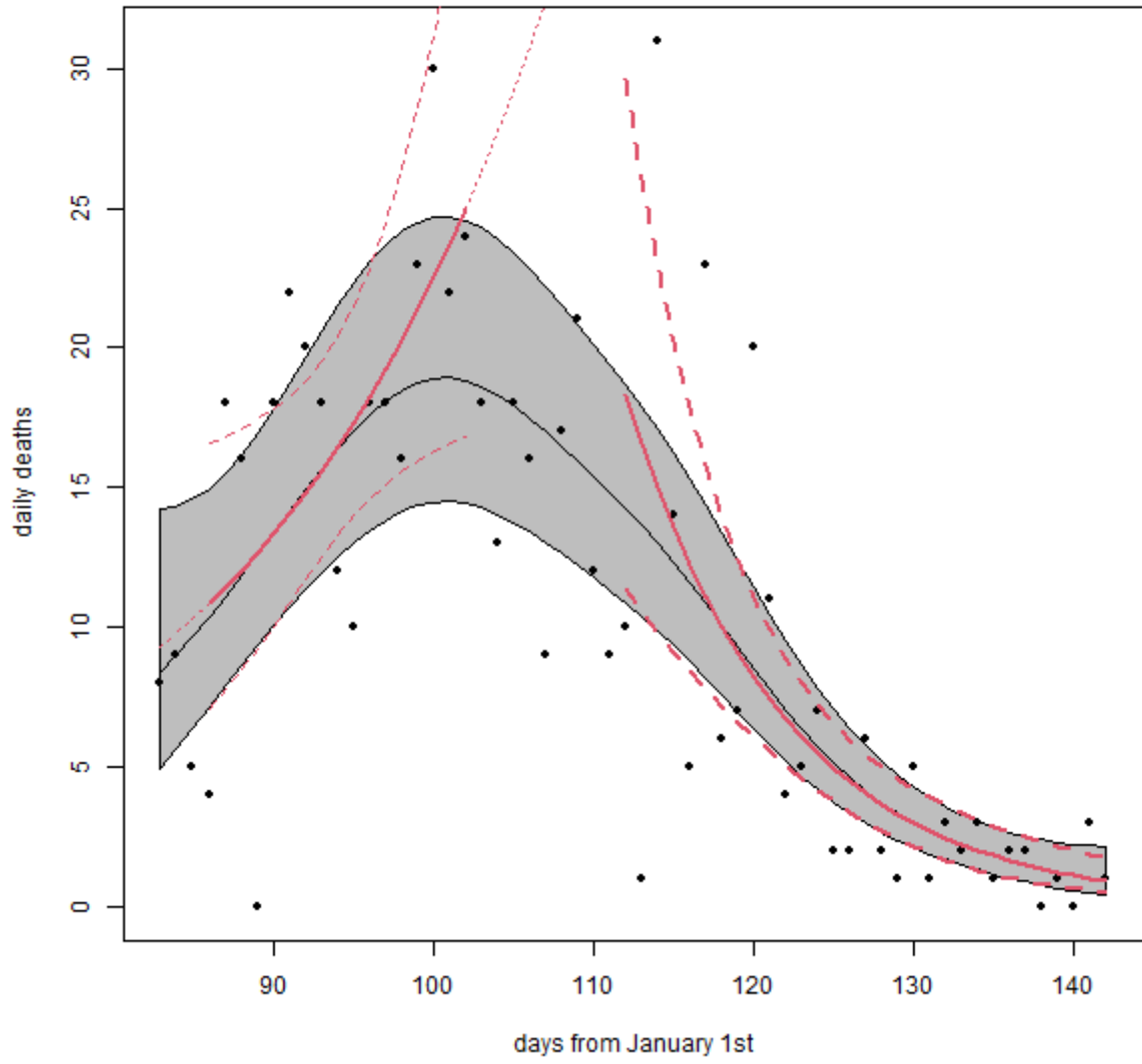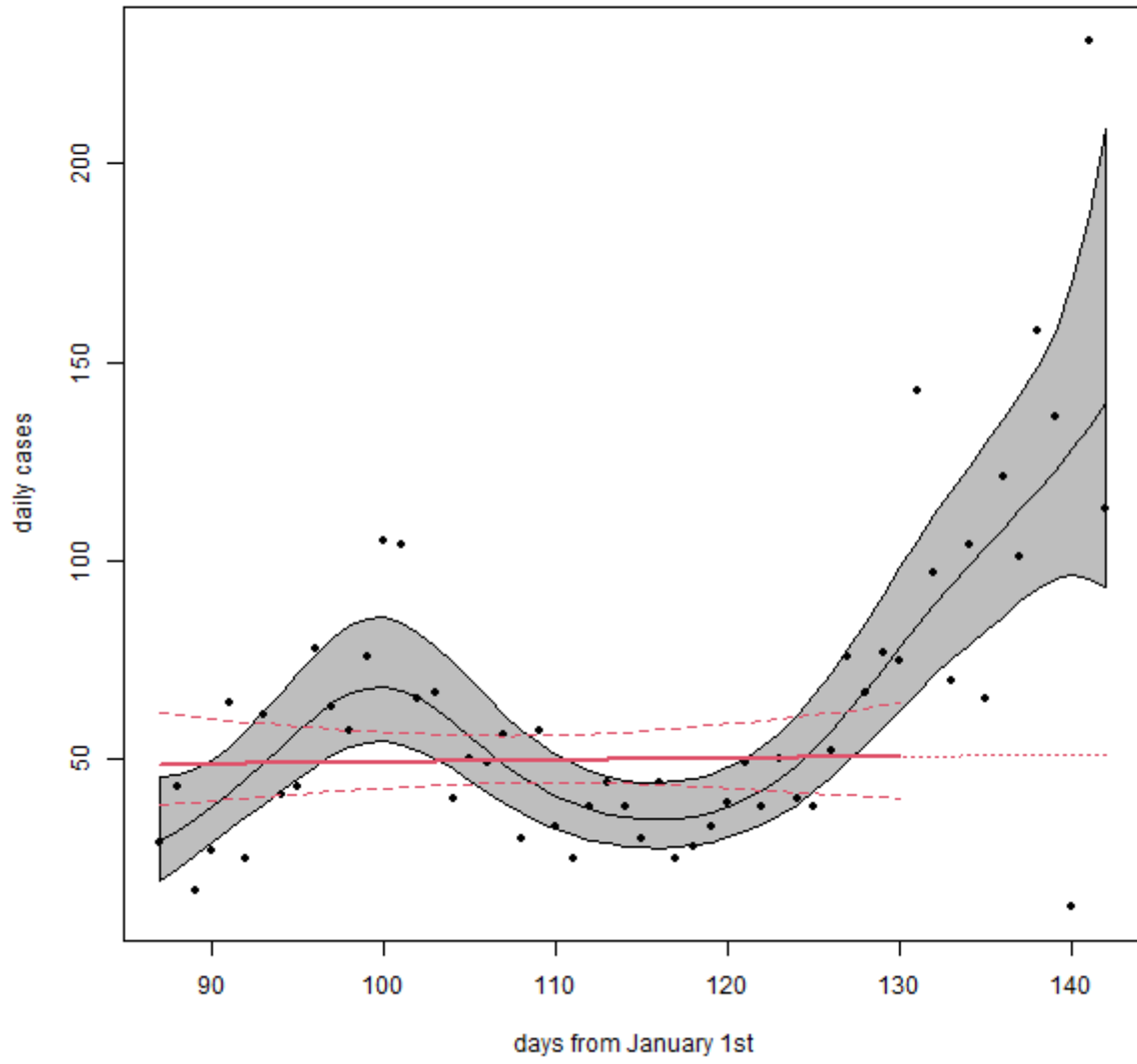
Bahrain
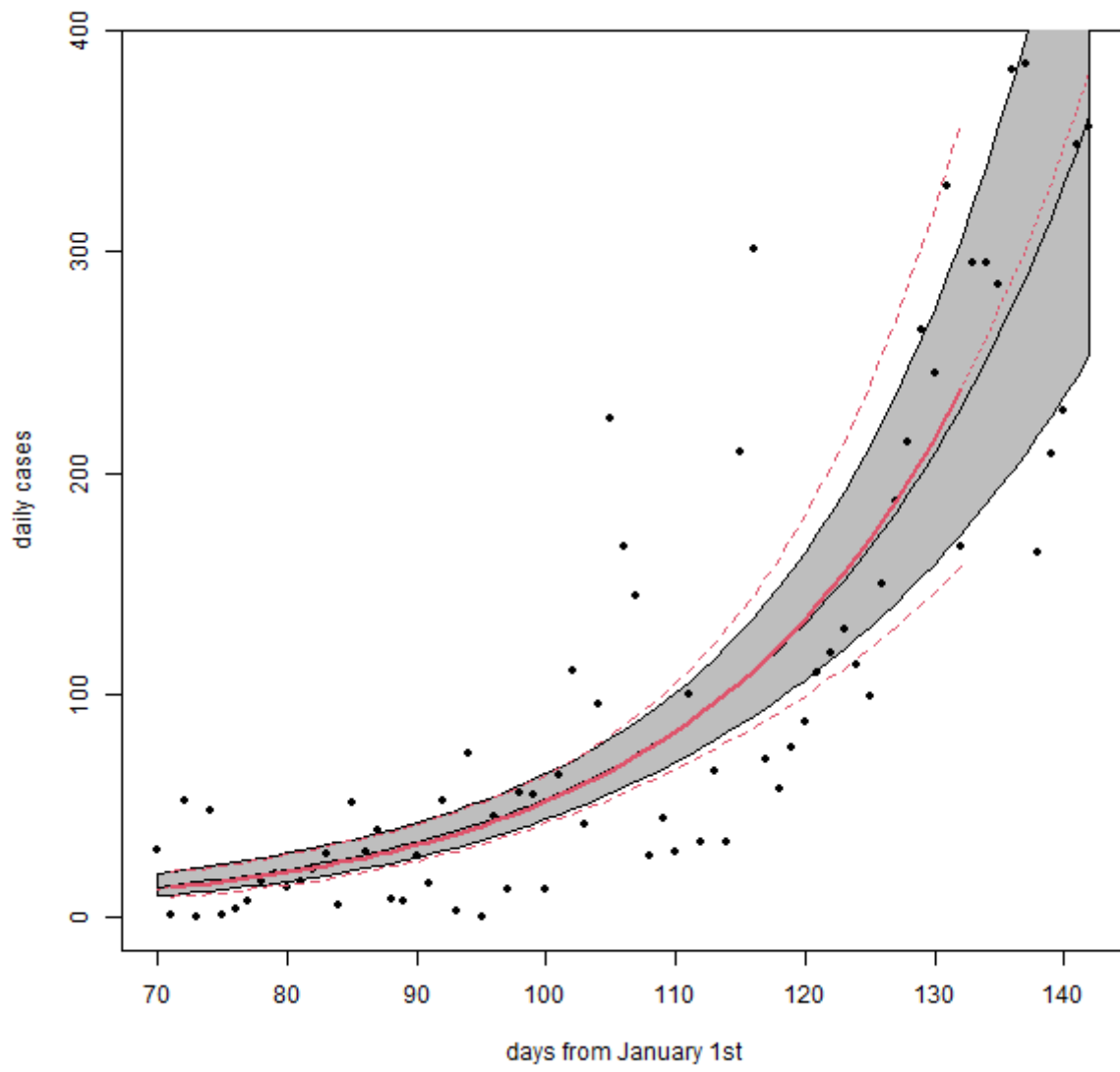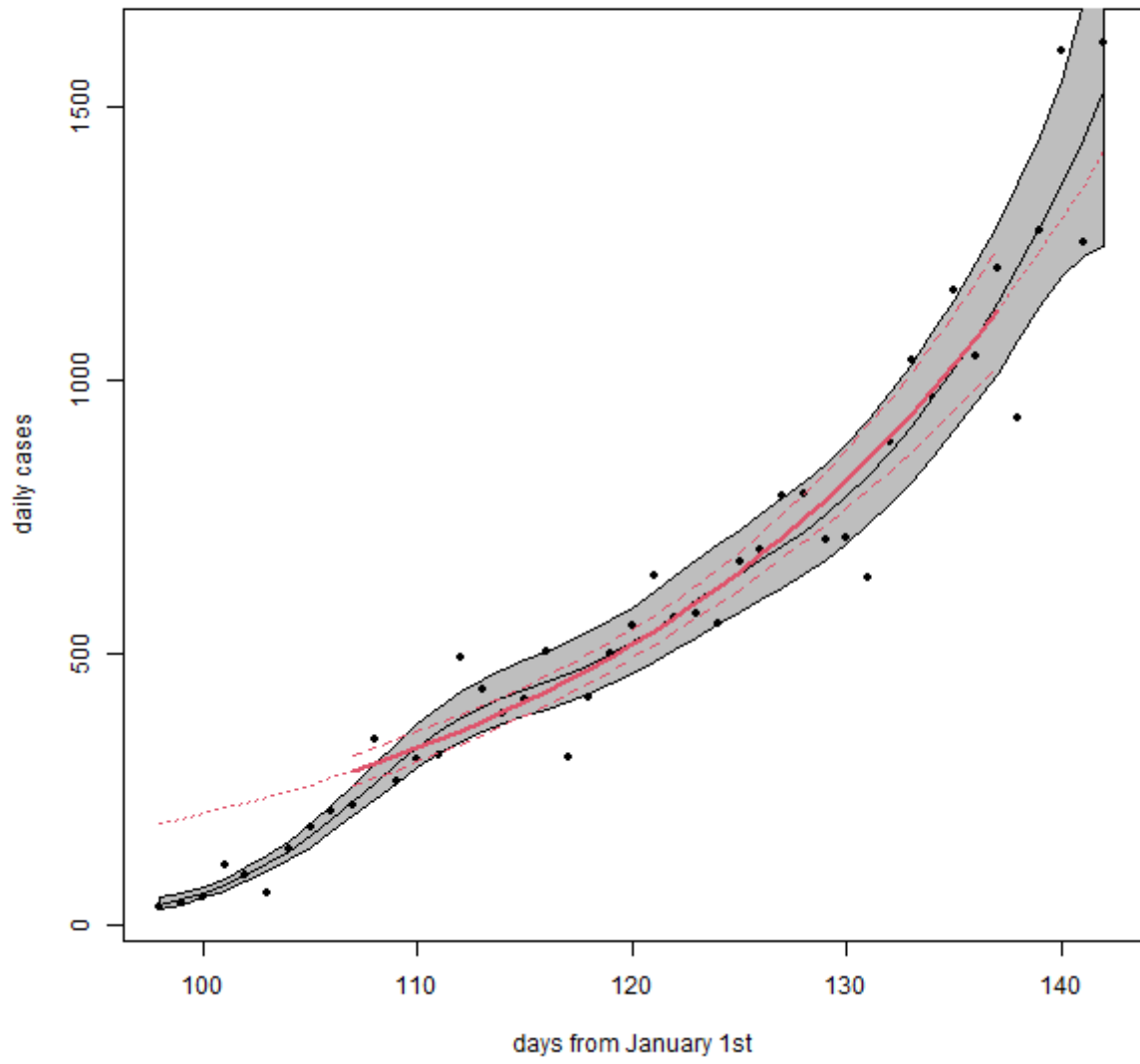
Bangladesh

Bangladesh
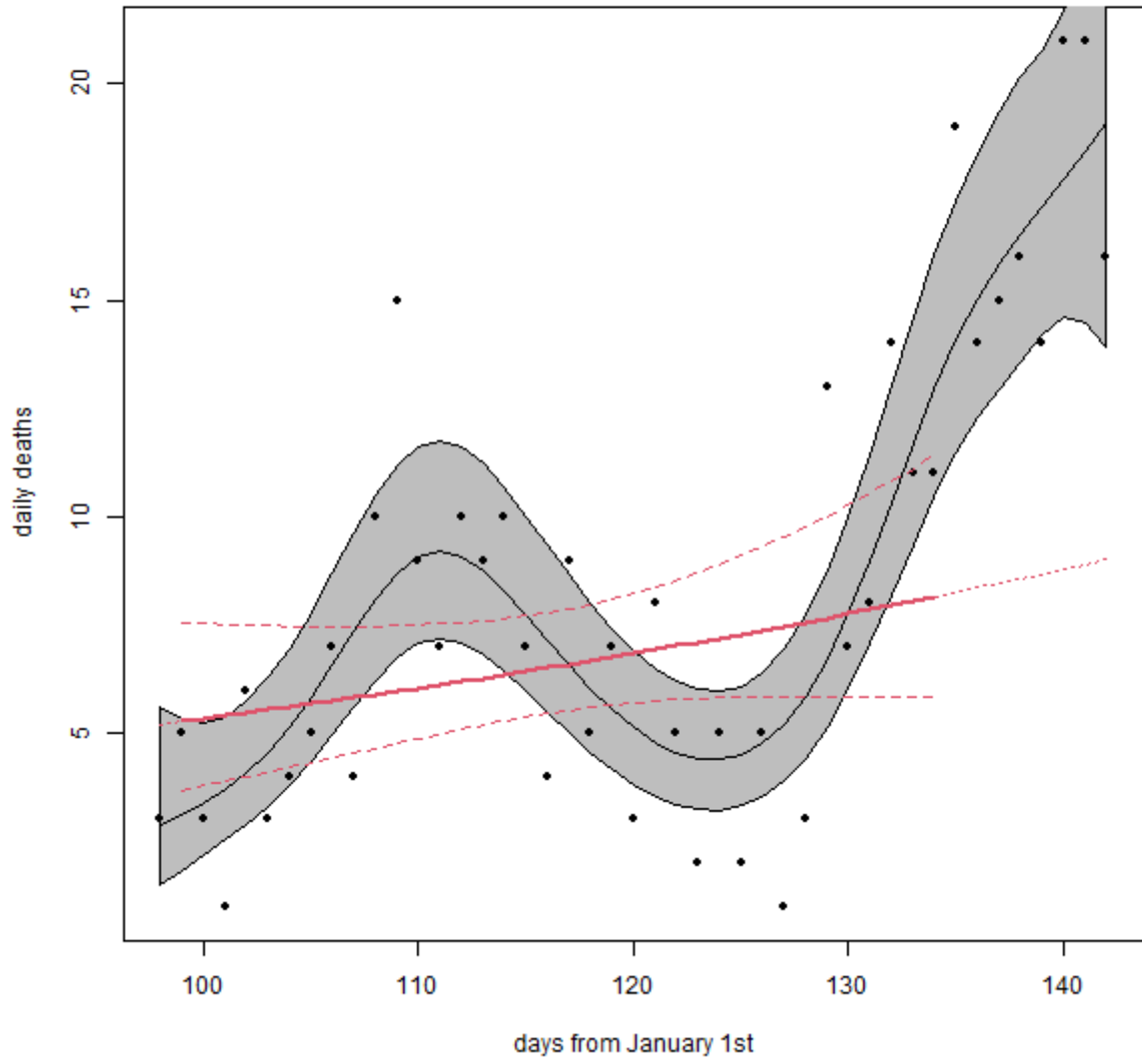
Belarus
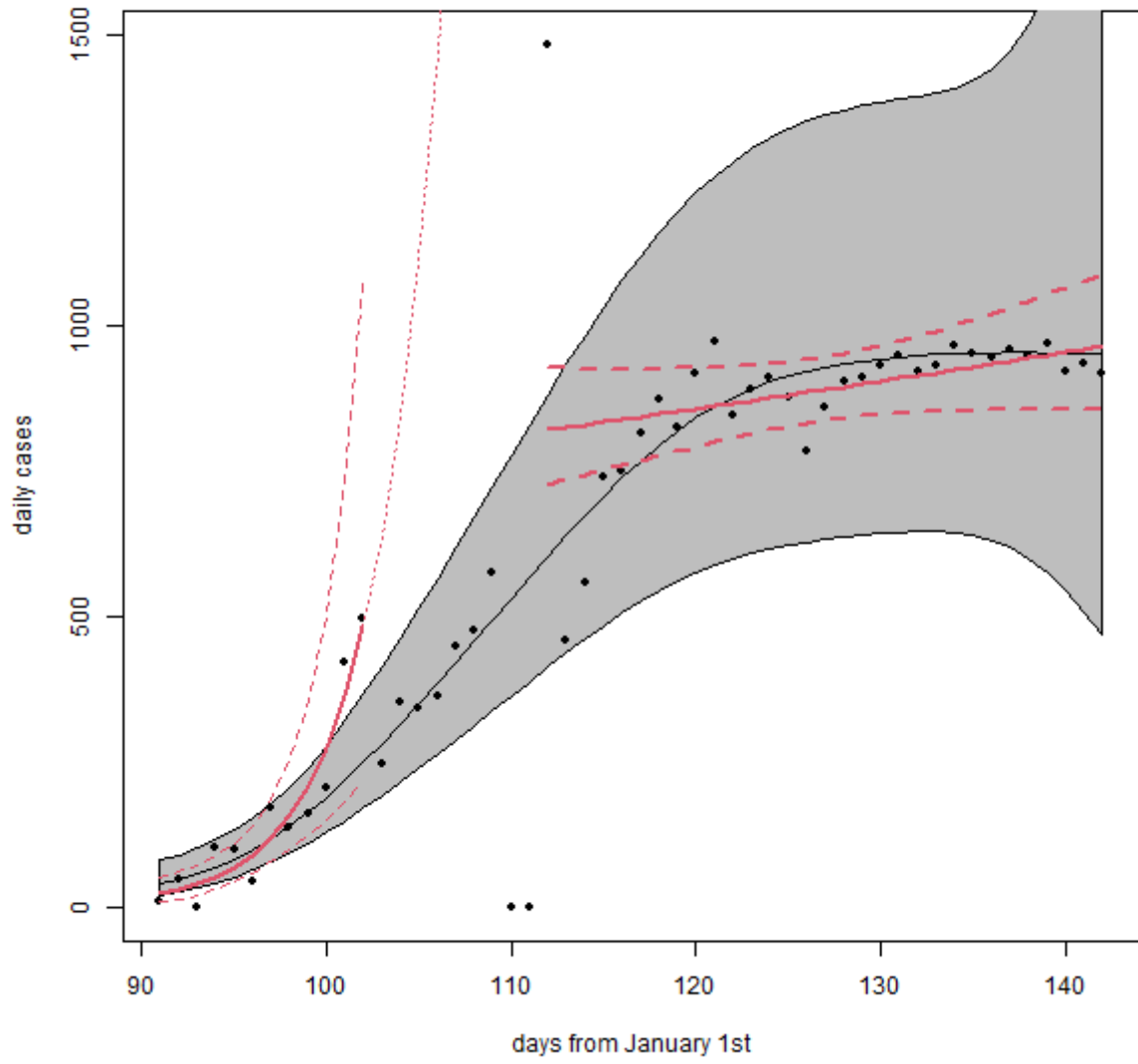
Belarus

Belgium

Belgium

days from January 1st

Bolivia

Bolivia

Bosnia

daily cases

days from January 1st

Bosnia

daily deaths

days from January 1st
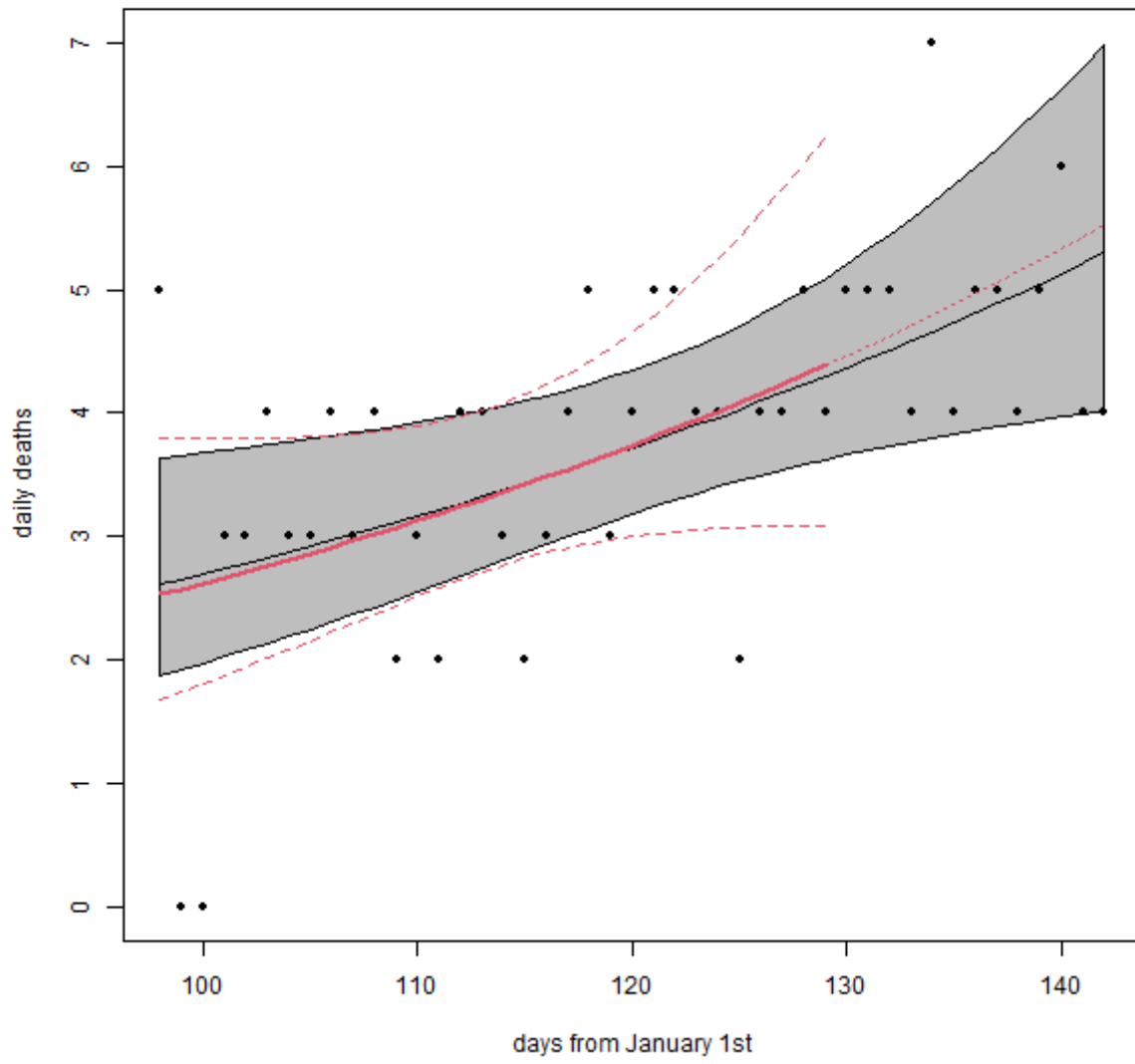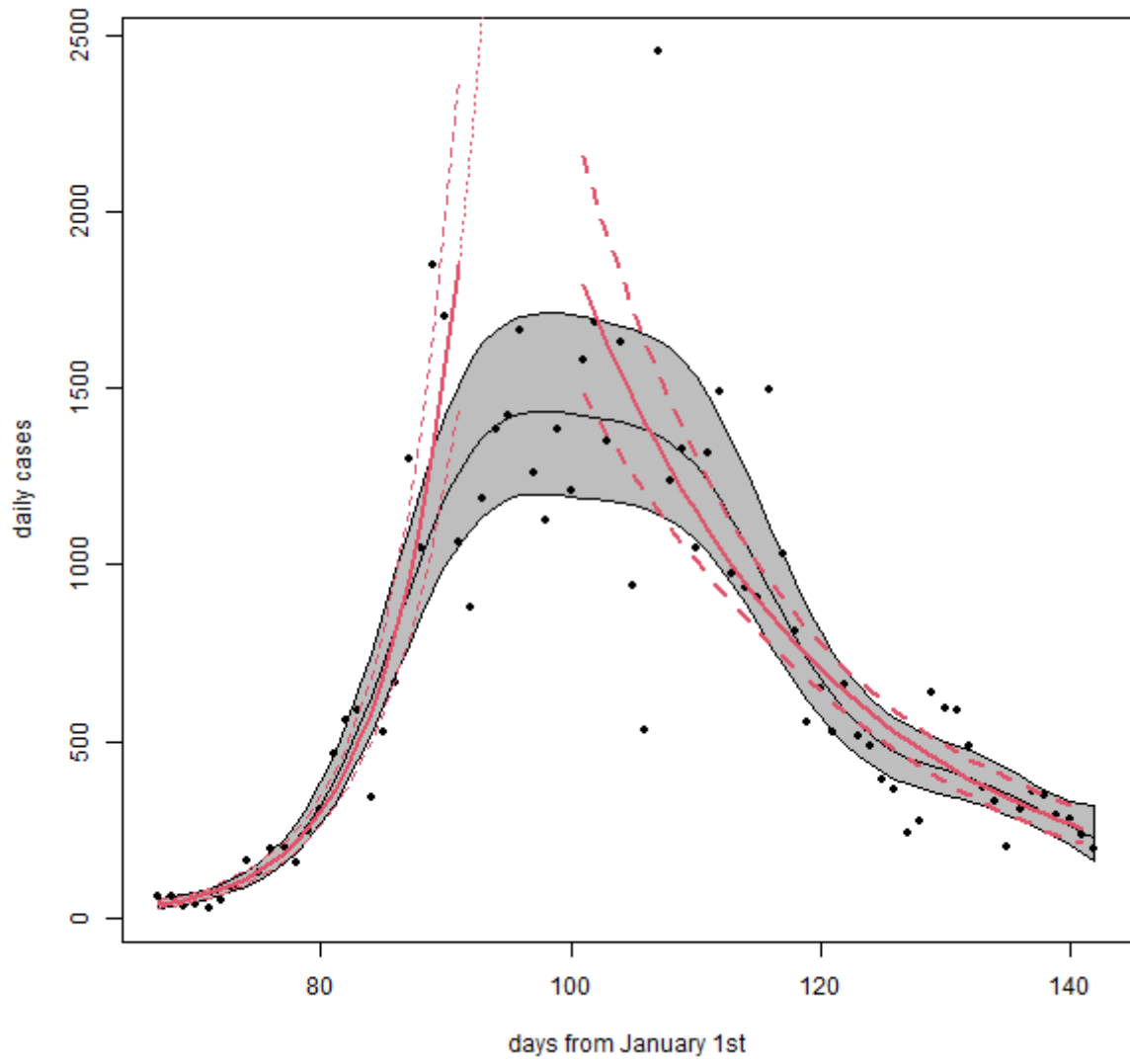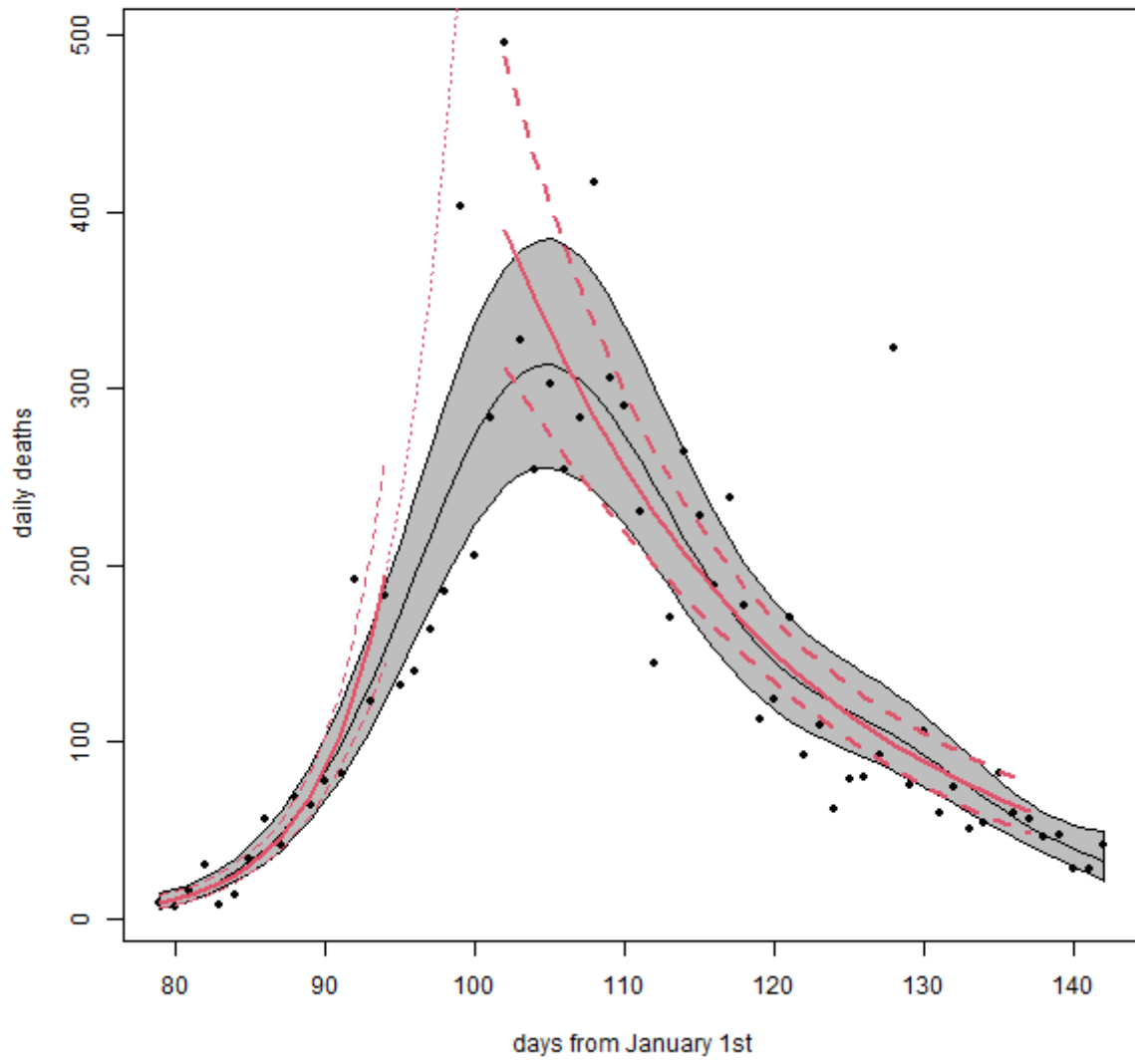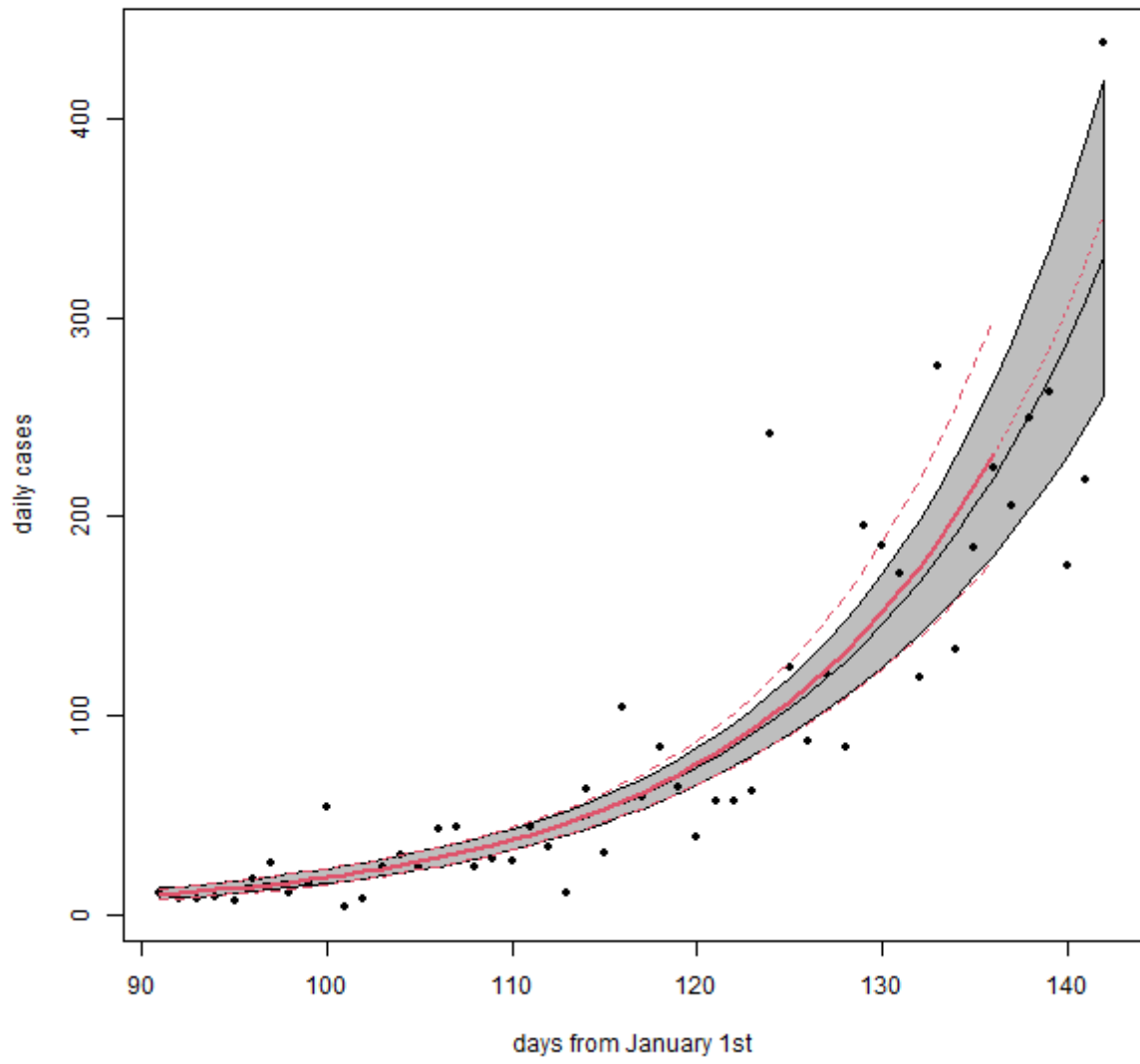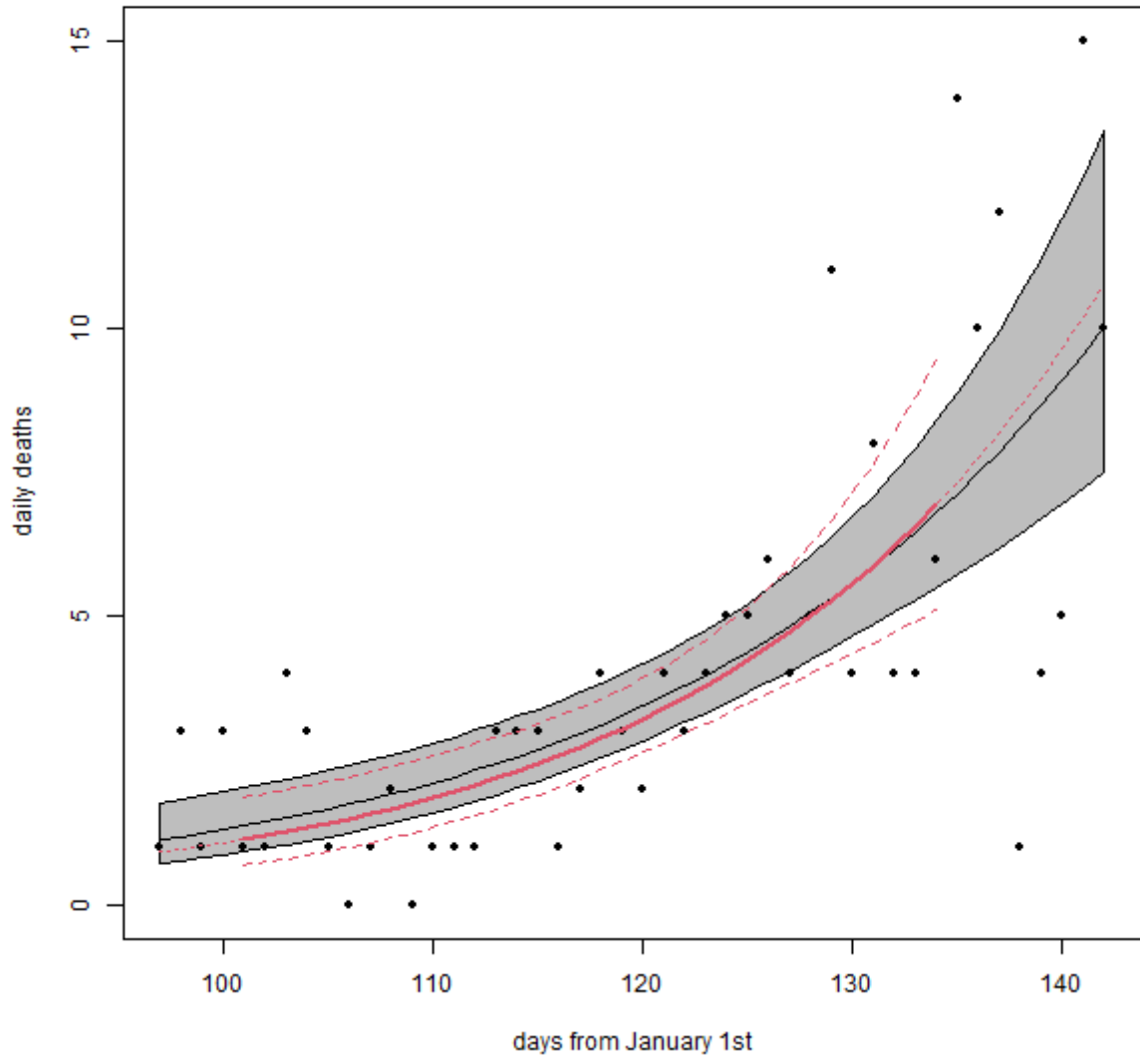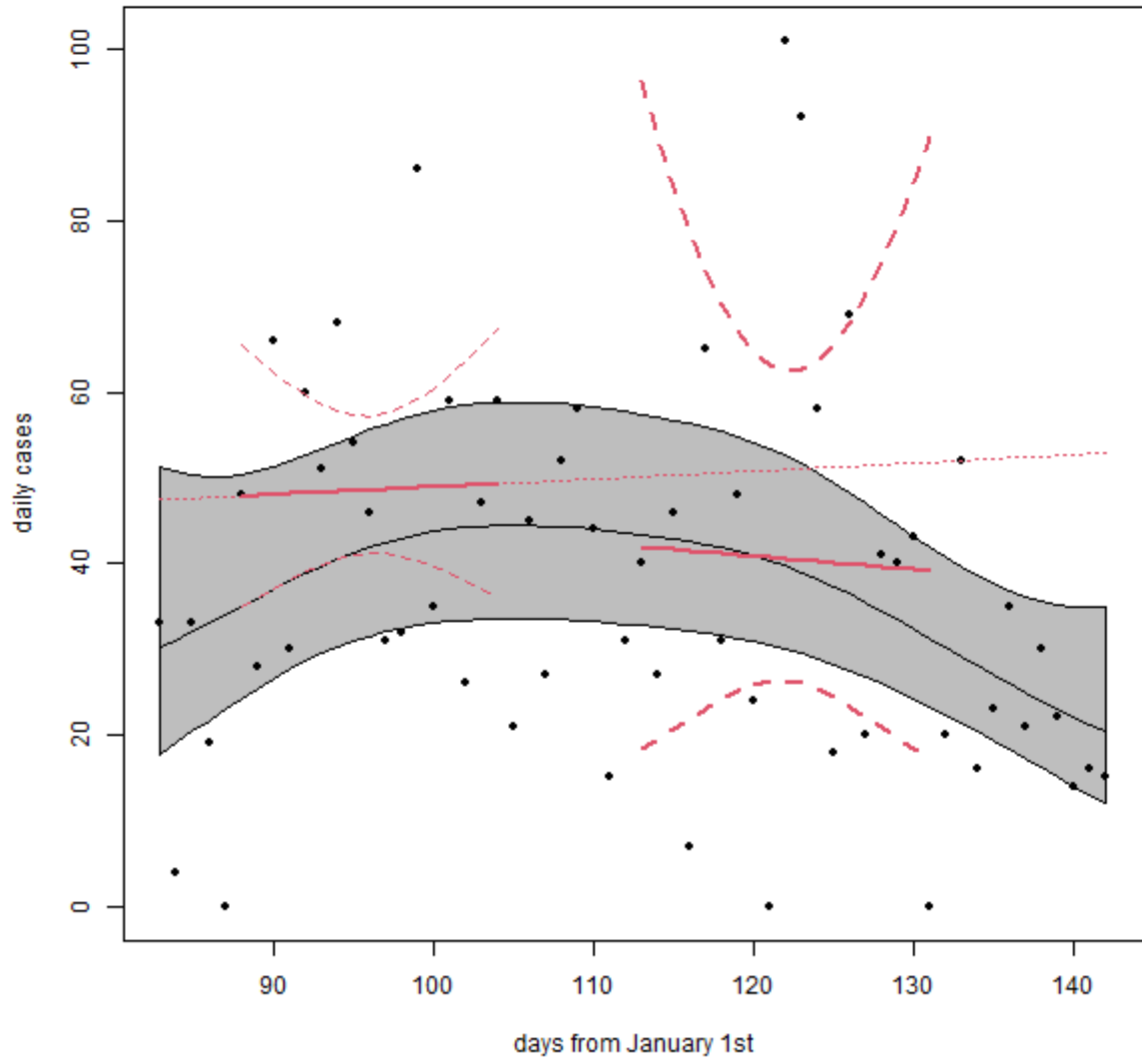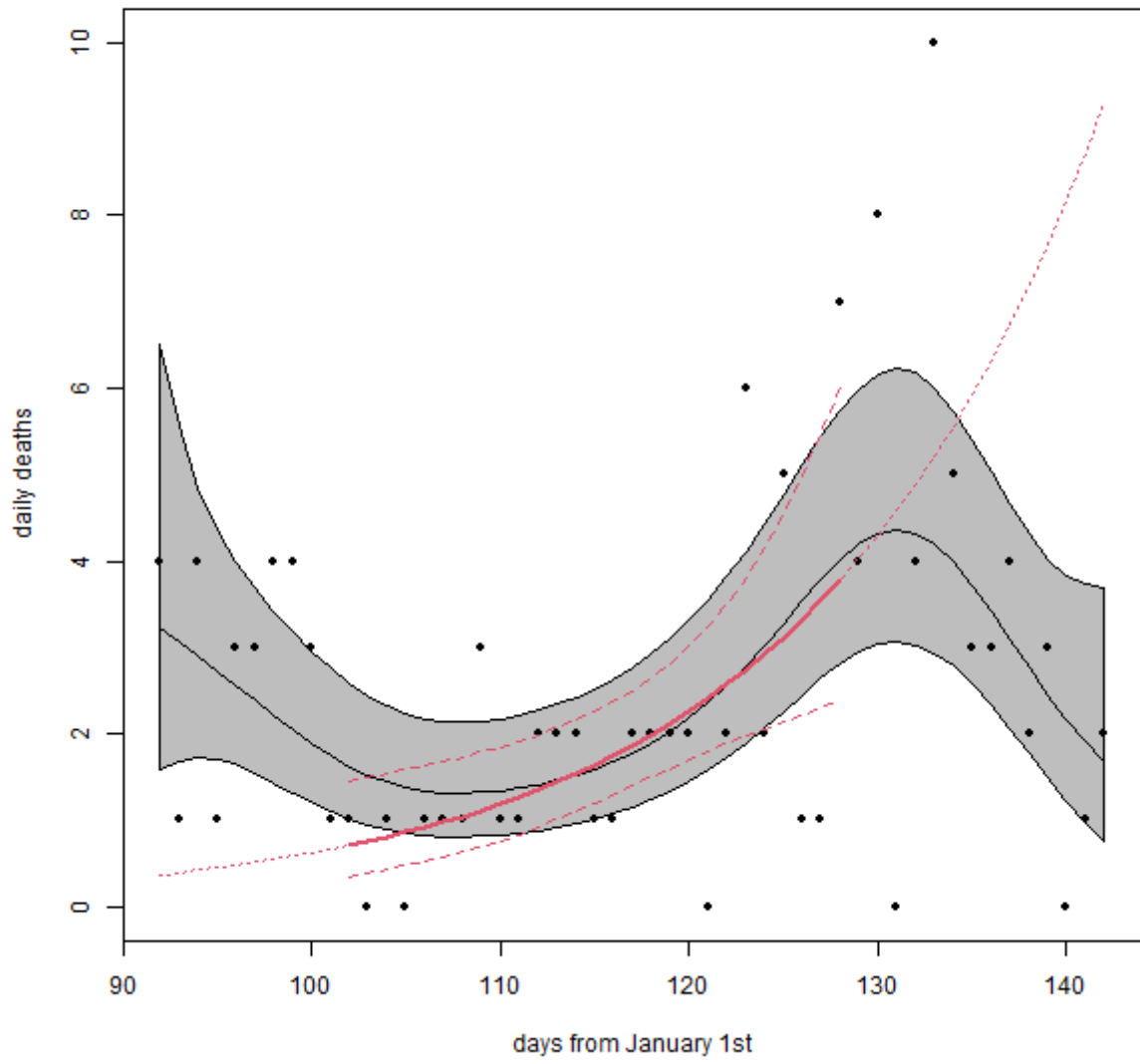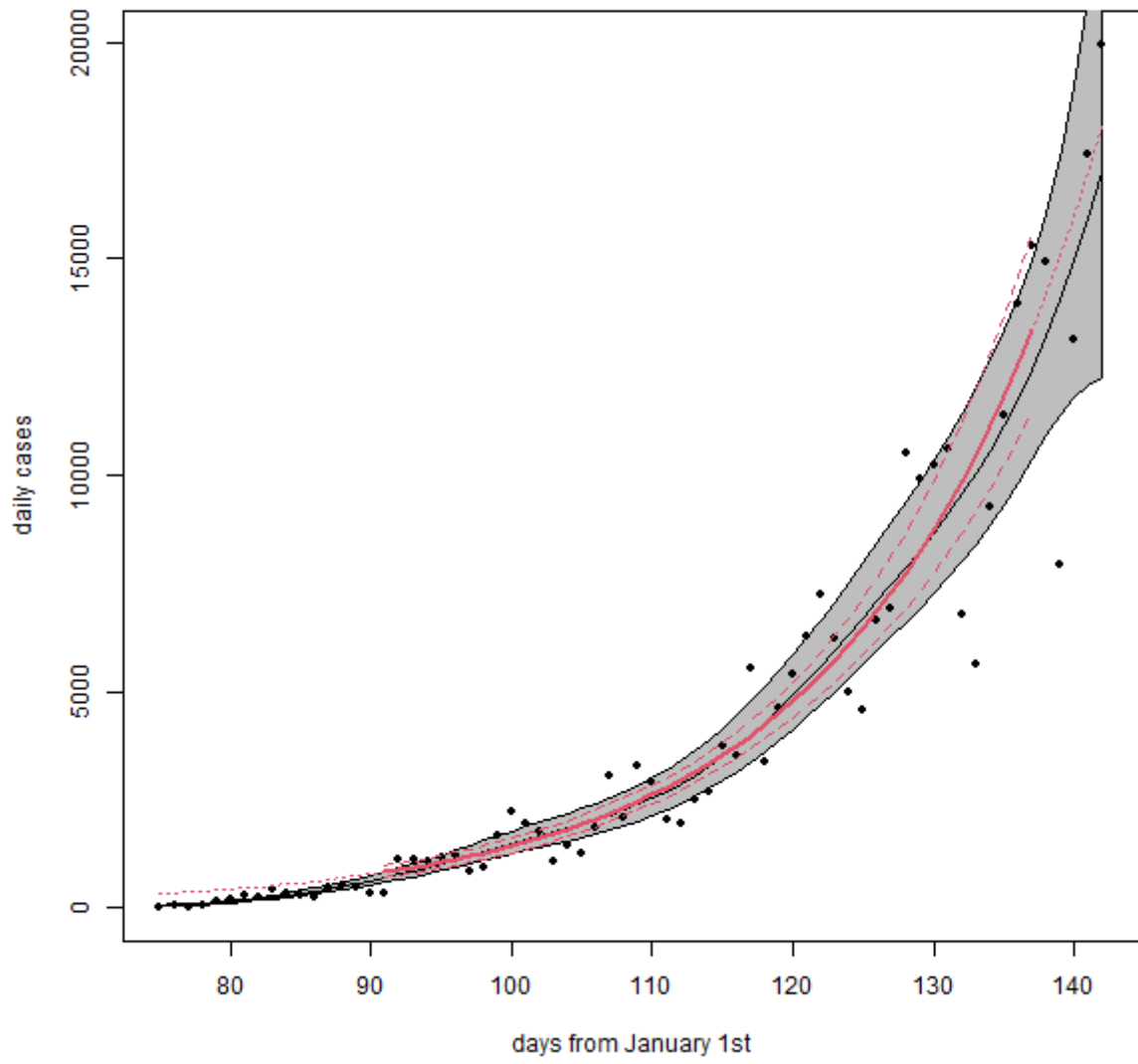
Brazil

Brazil
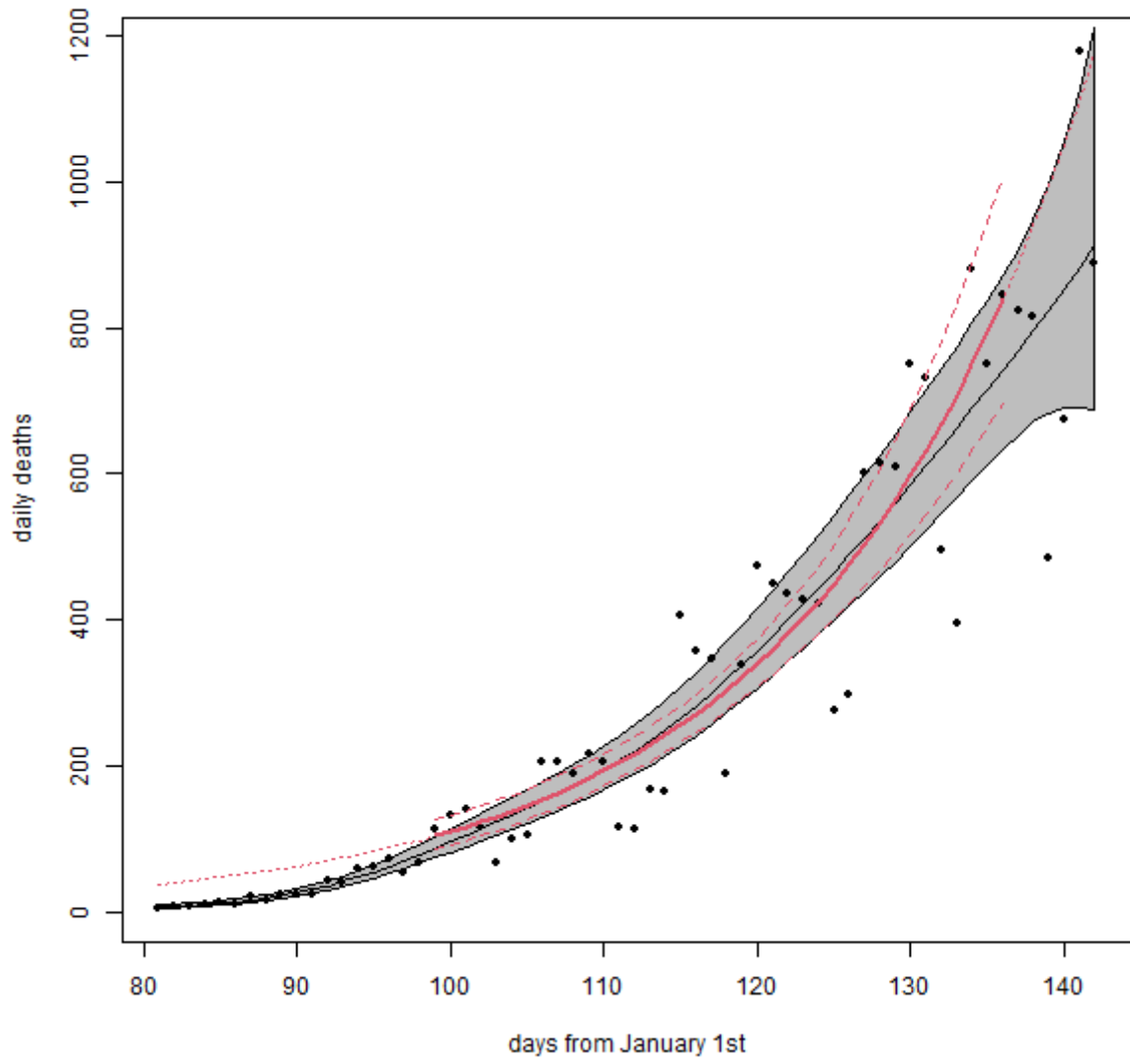
Bulgaria

Bulgaria

Cameroon

daily cases

days from January 1st

Cameroon

daily deaths

days from January 1st

Canada

Canada

daily deaths

days from January 1st

Chile

daily cases

days from January 1st

Chile

daily deaths

days from January 1st

China

days from January 1st
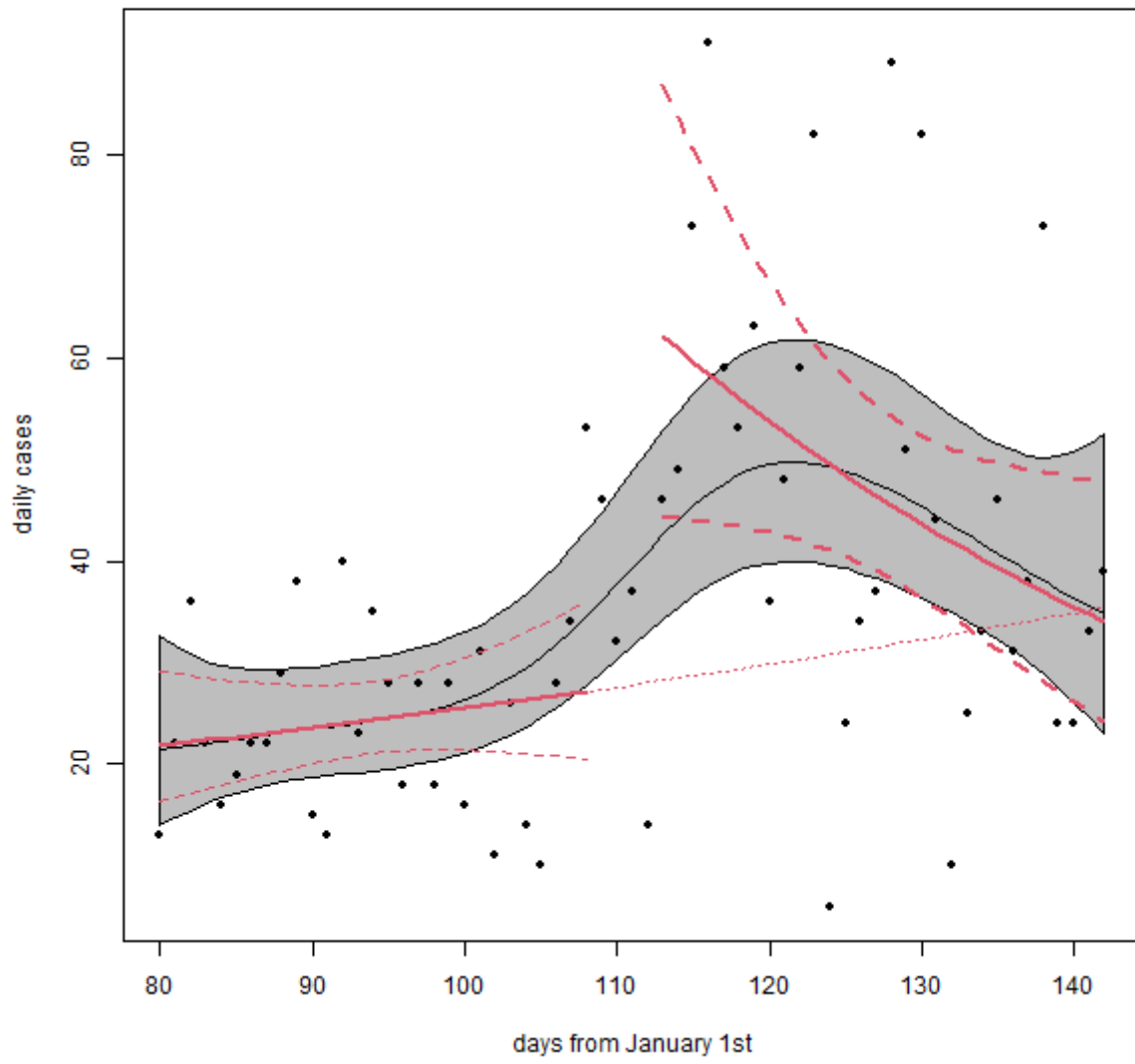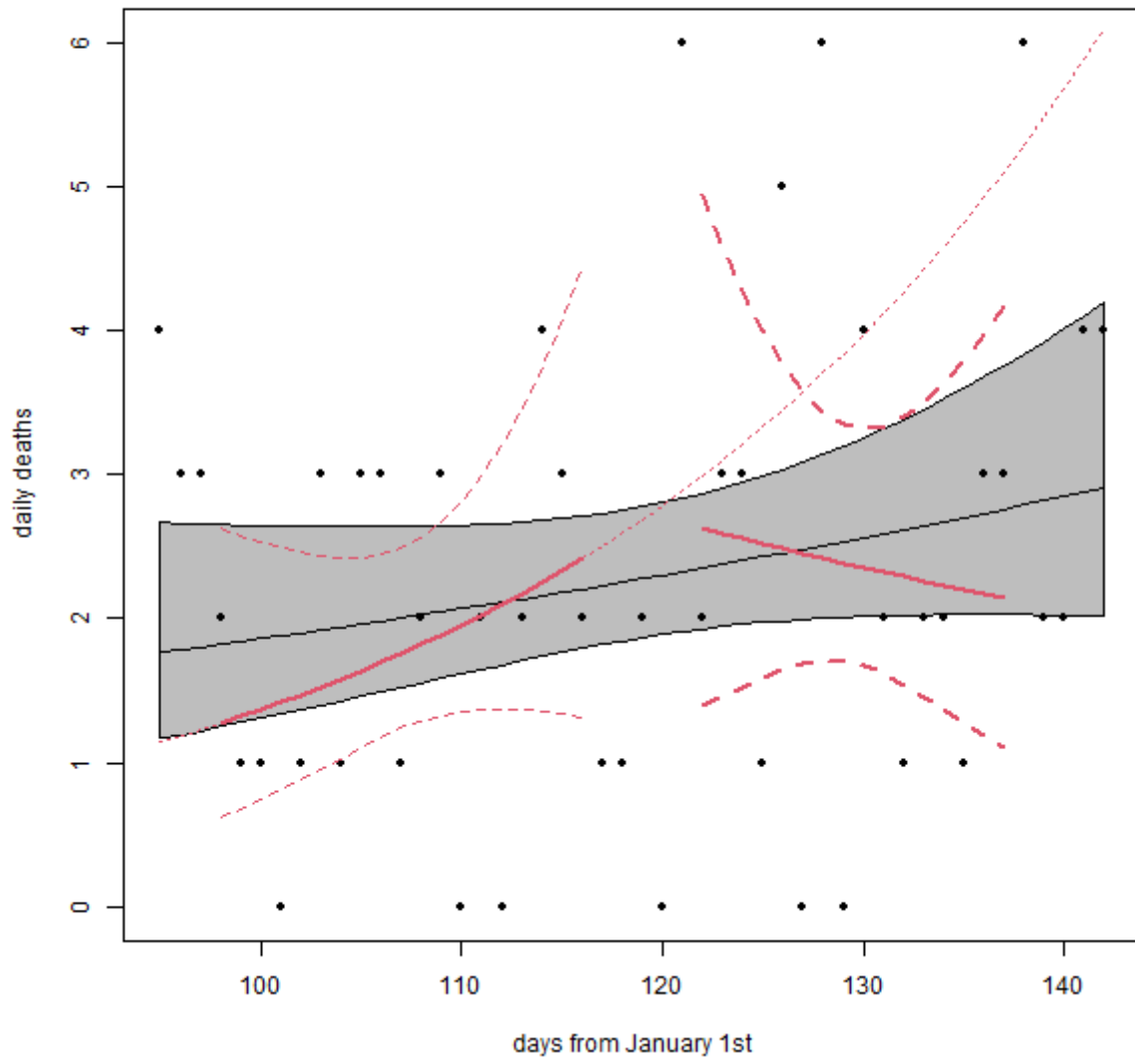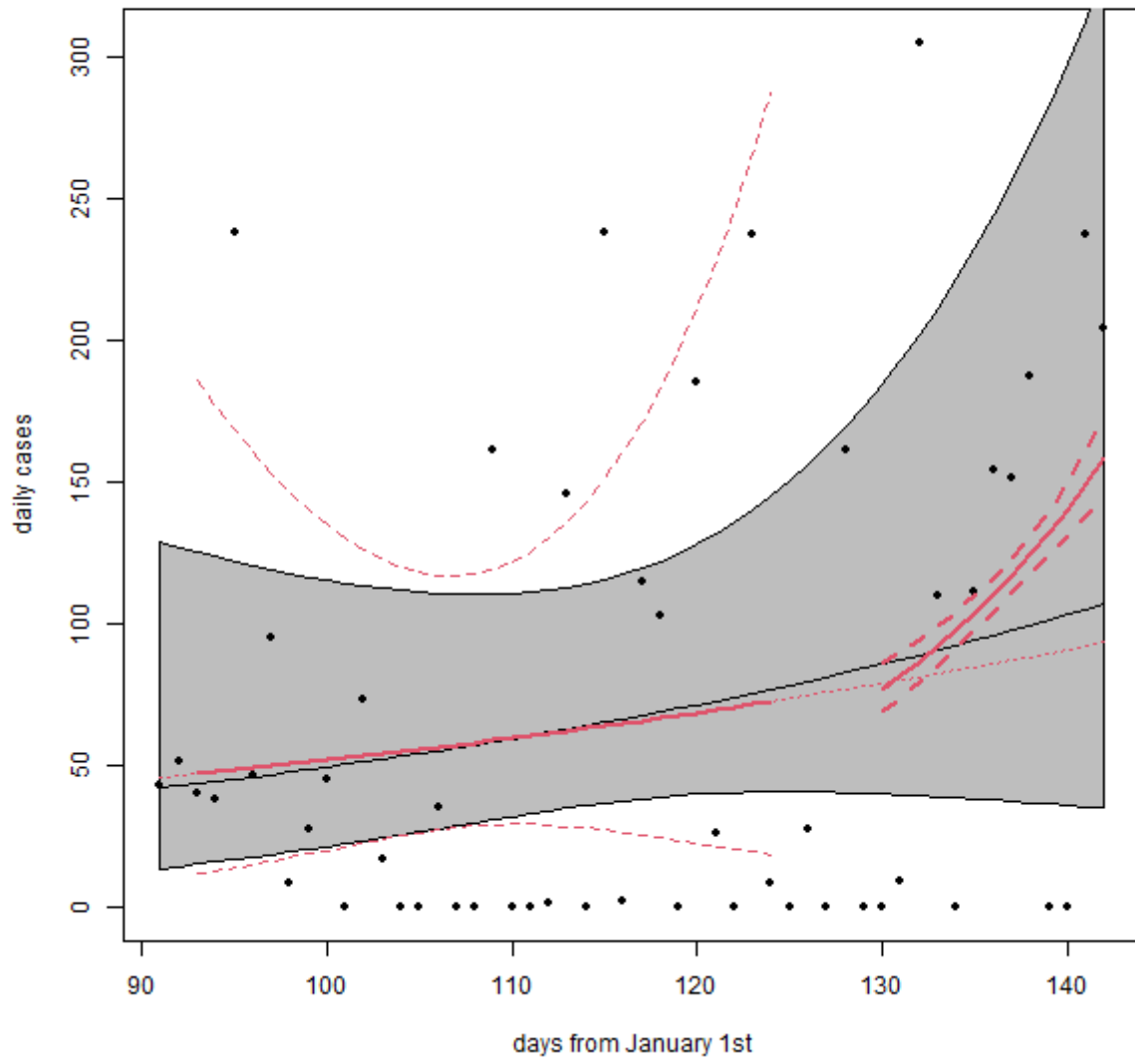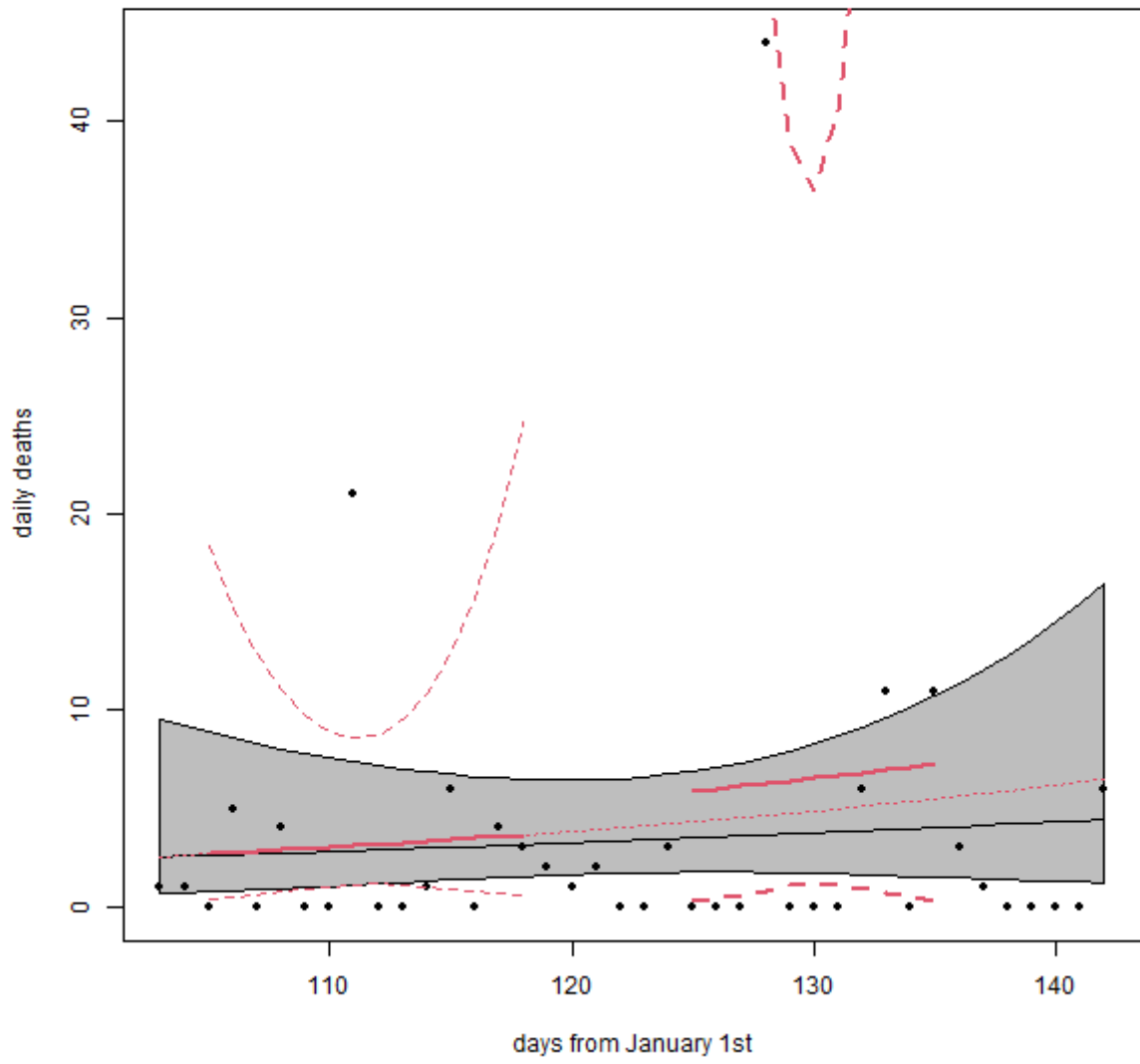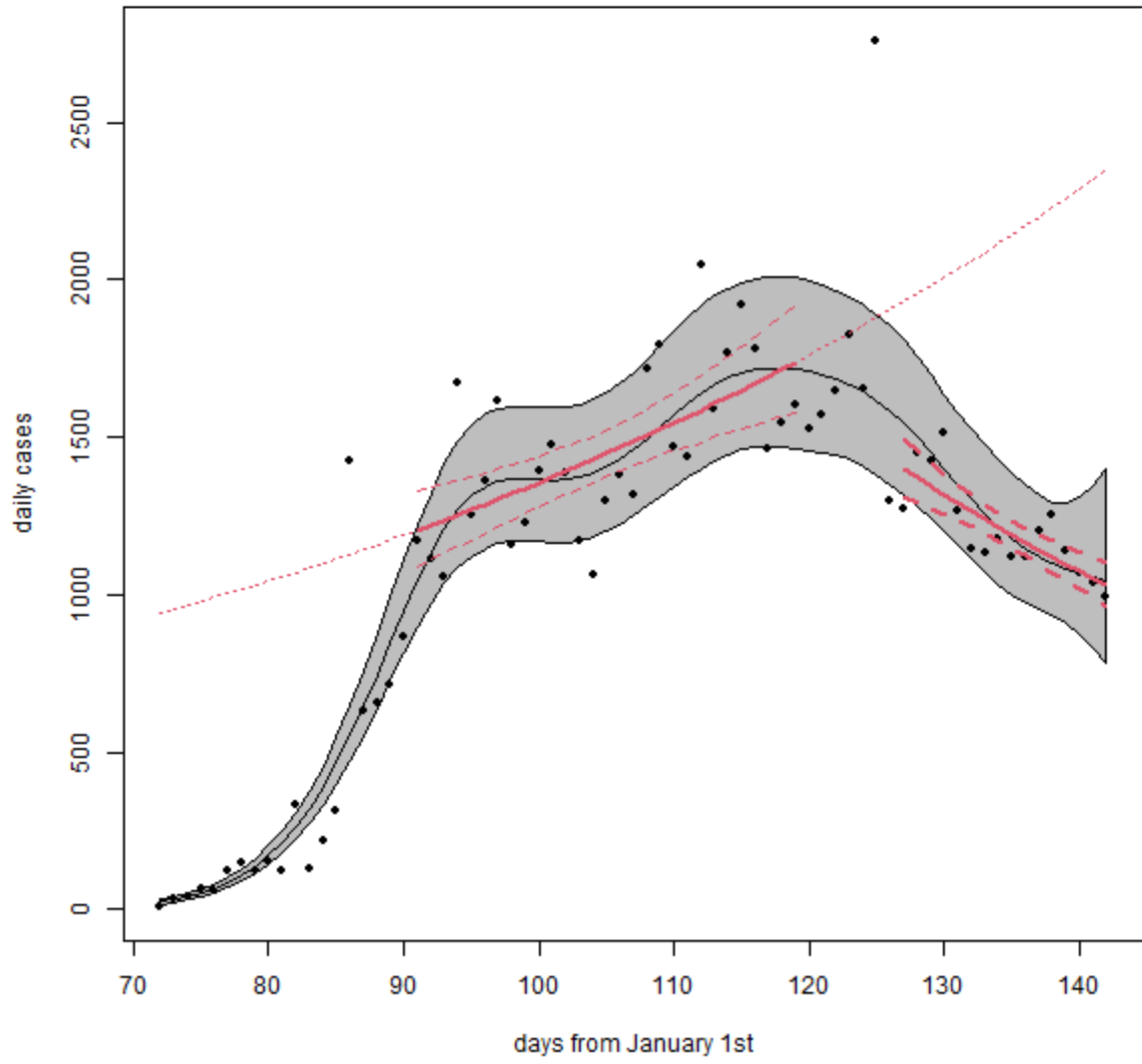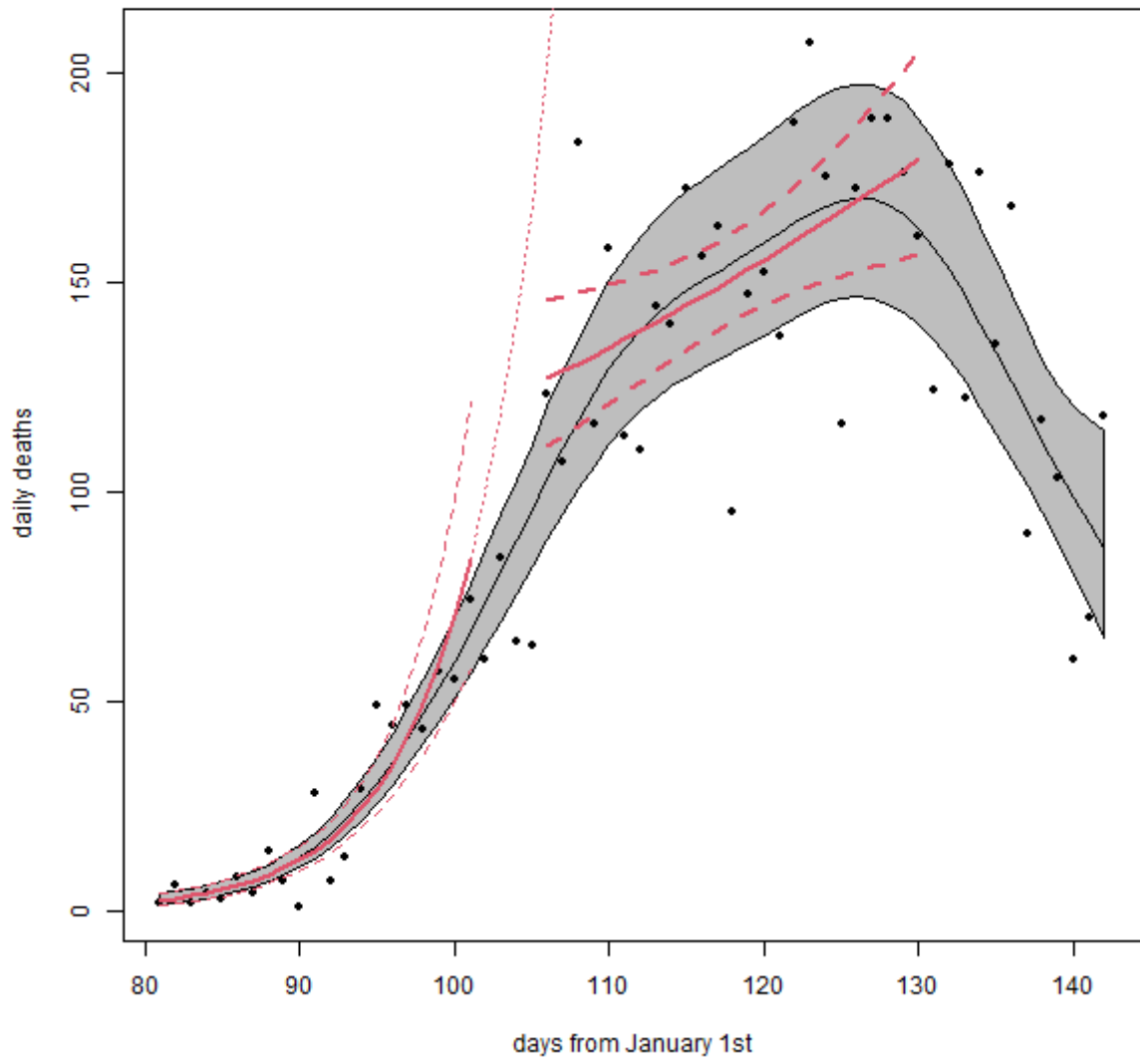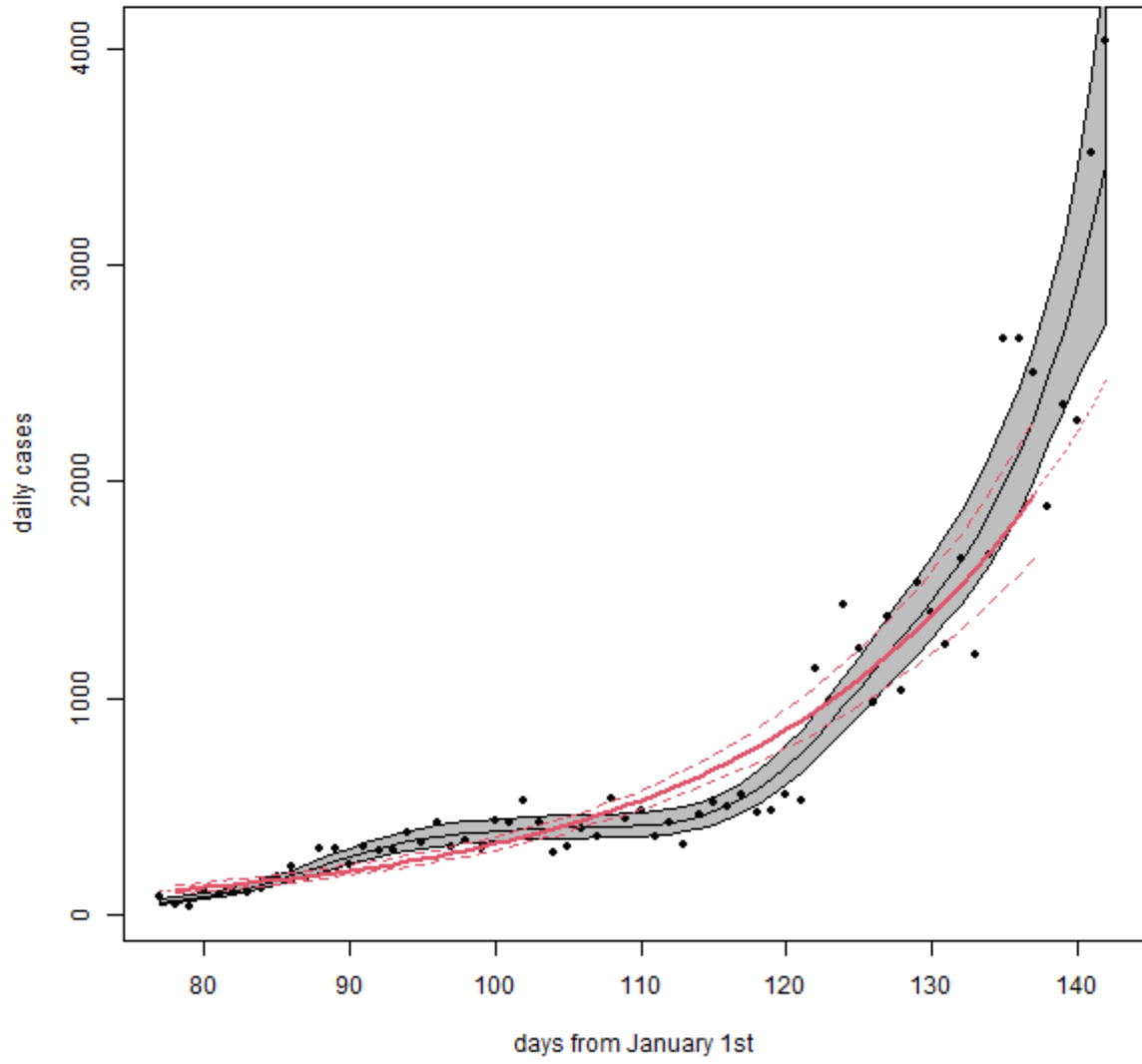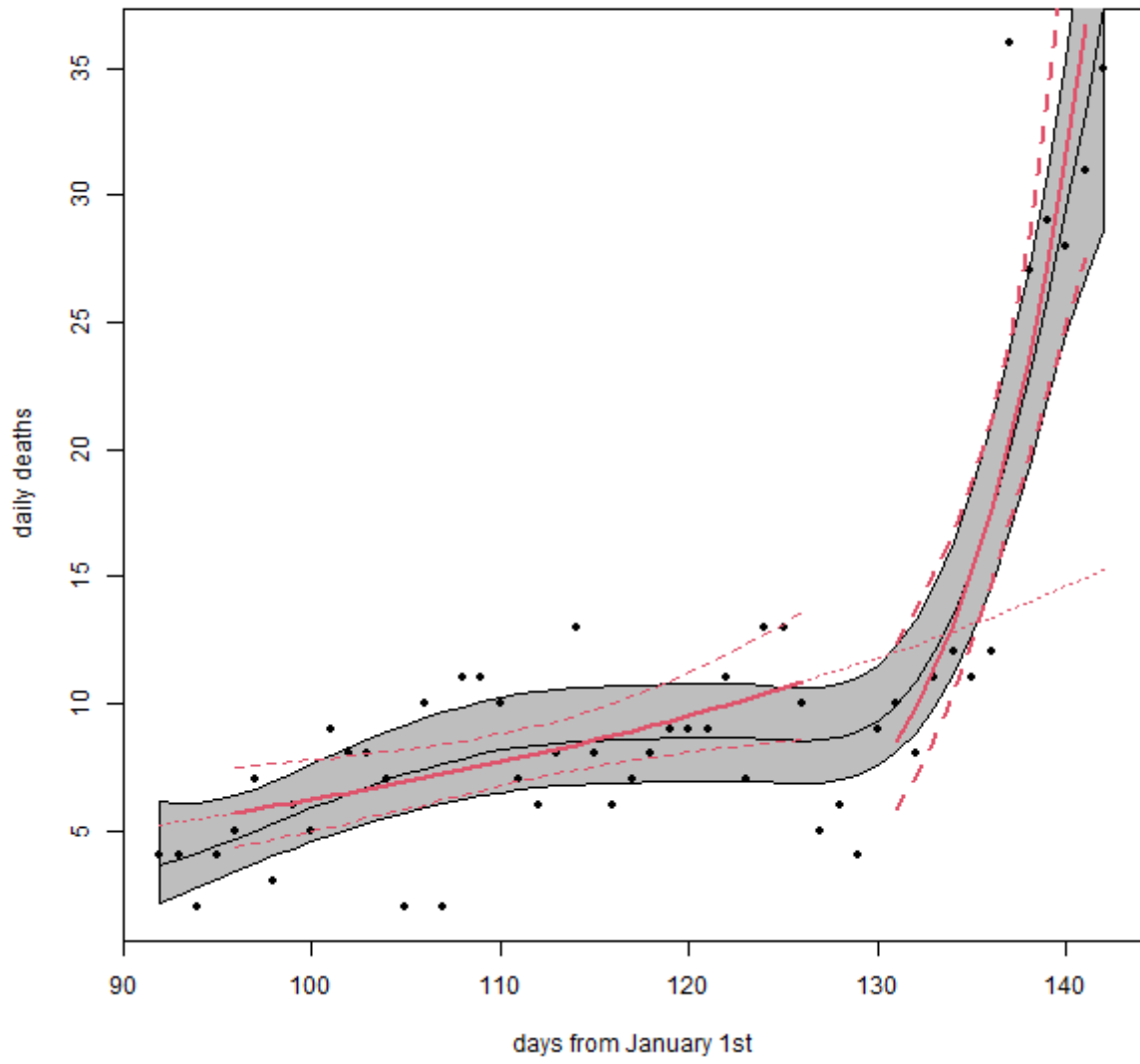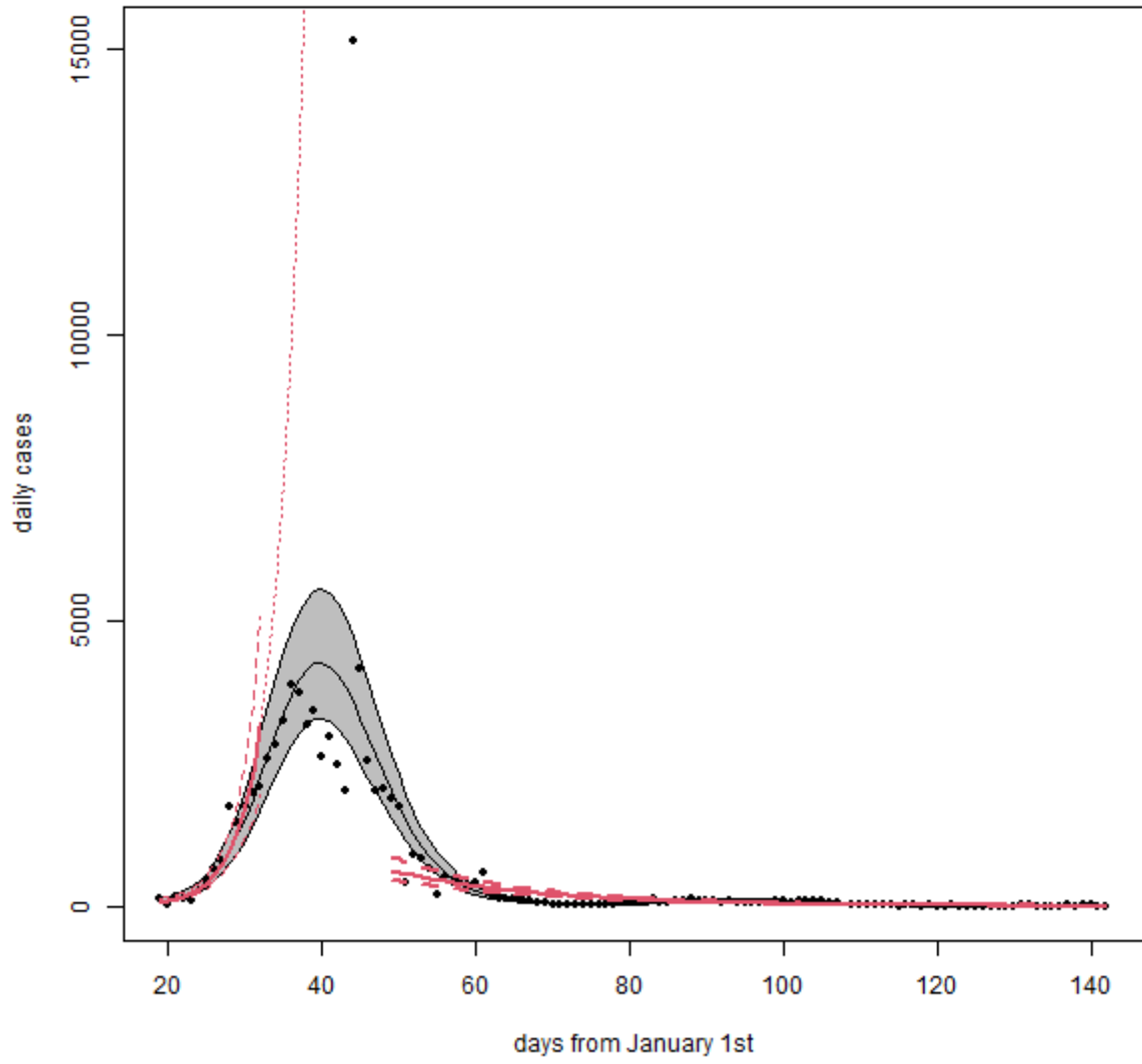
China
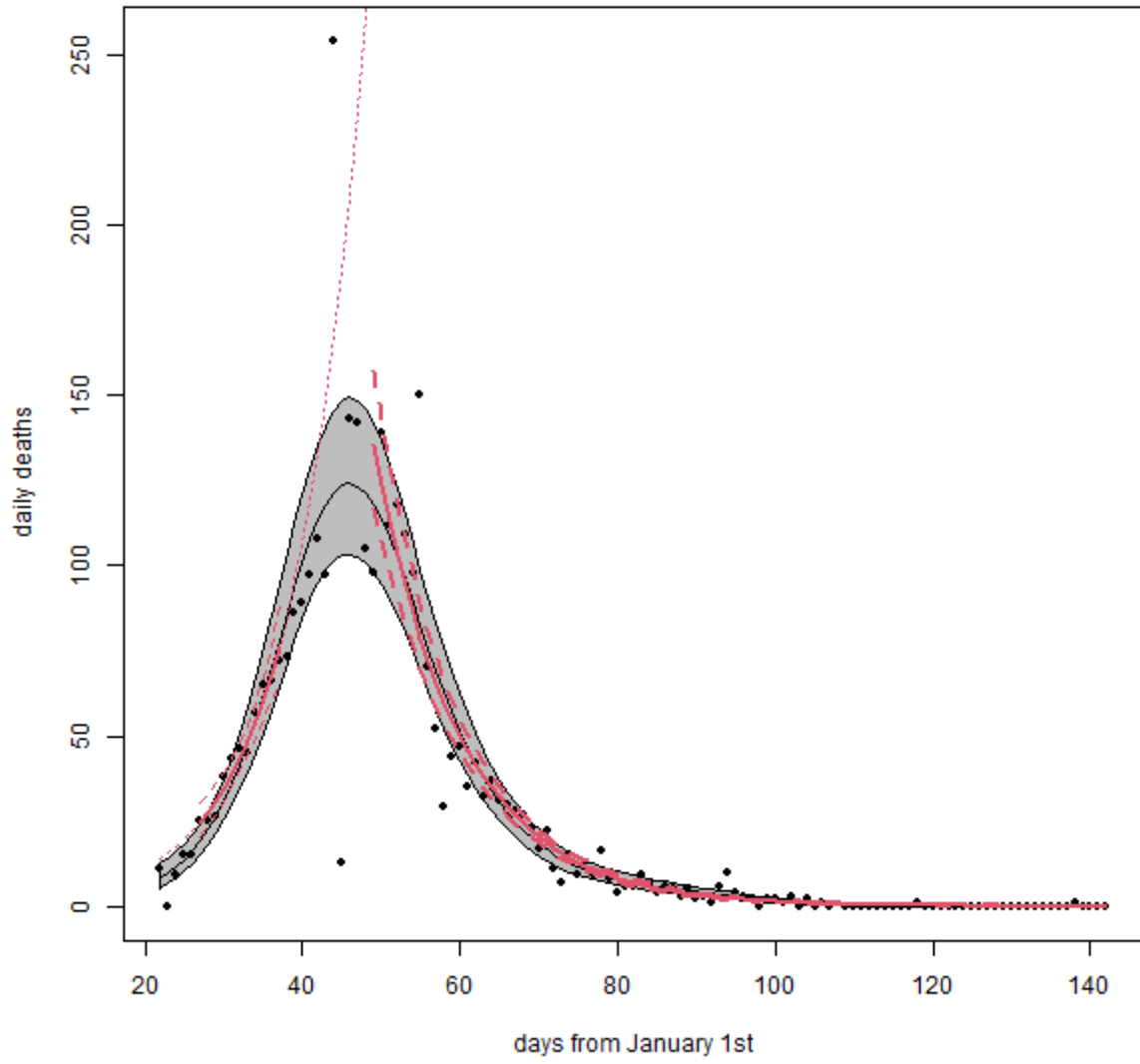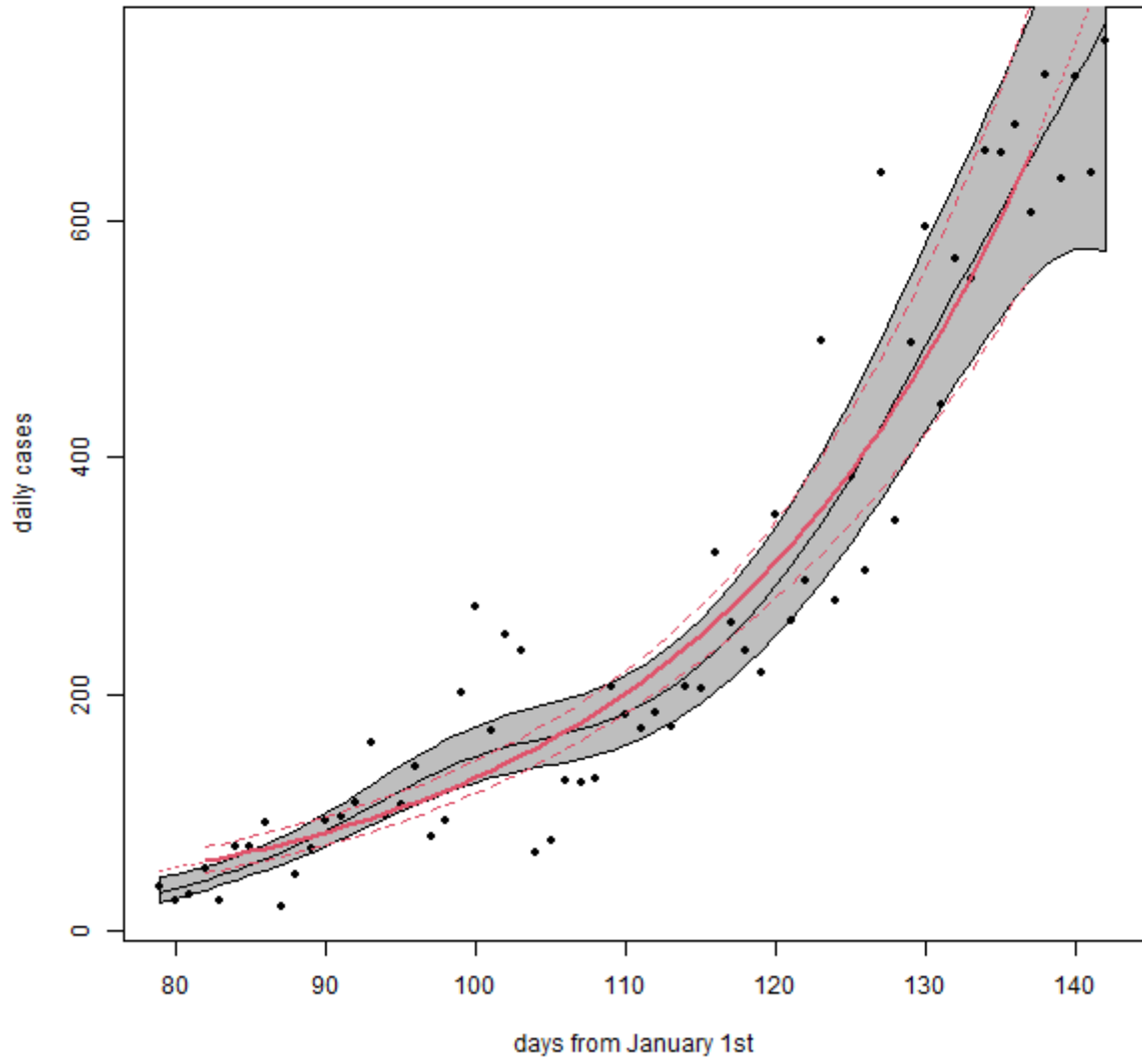
Colombia

Colombia

Cote_dIvoire

Cuba

daily cases

days from January 1st

Czechia

daily deaths

days from January 1st
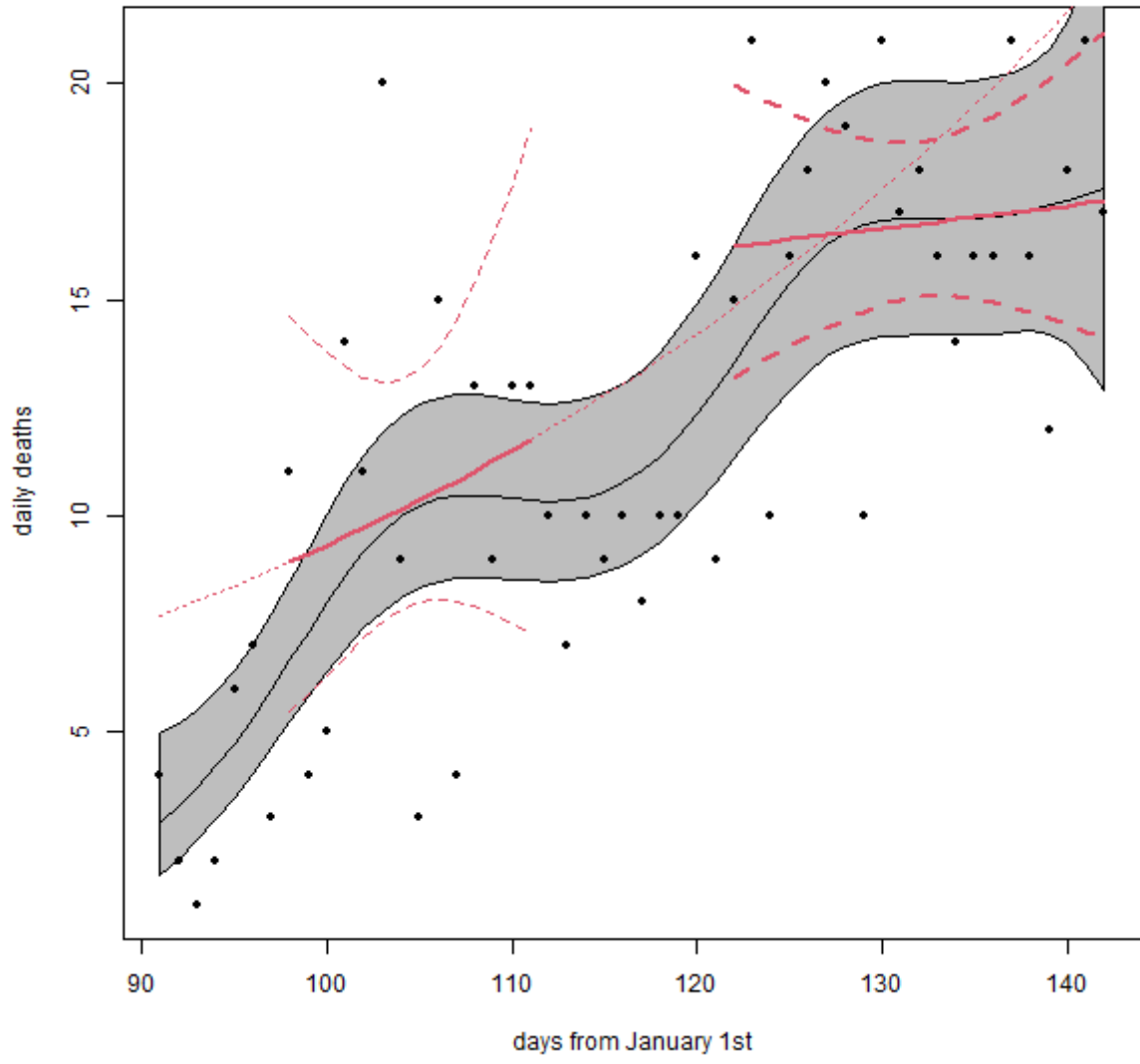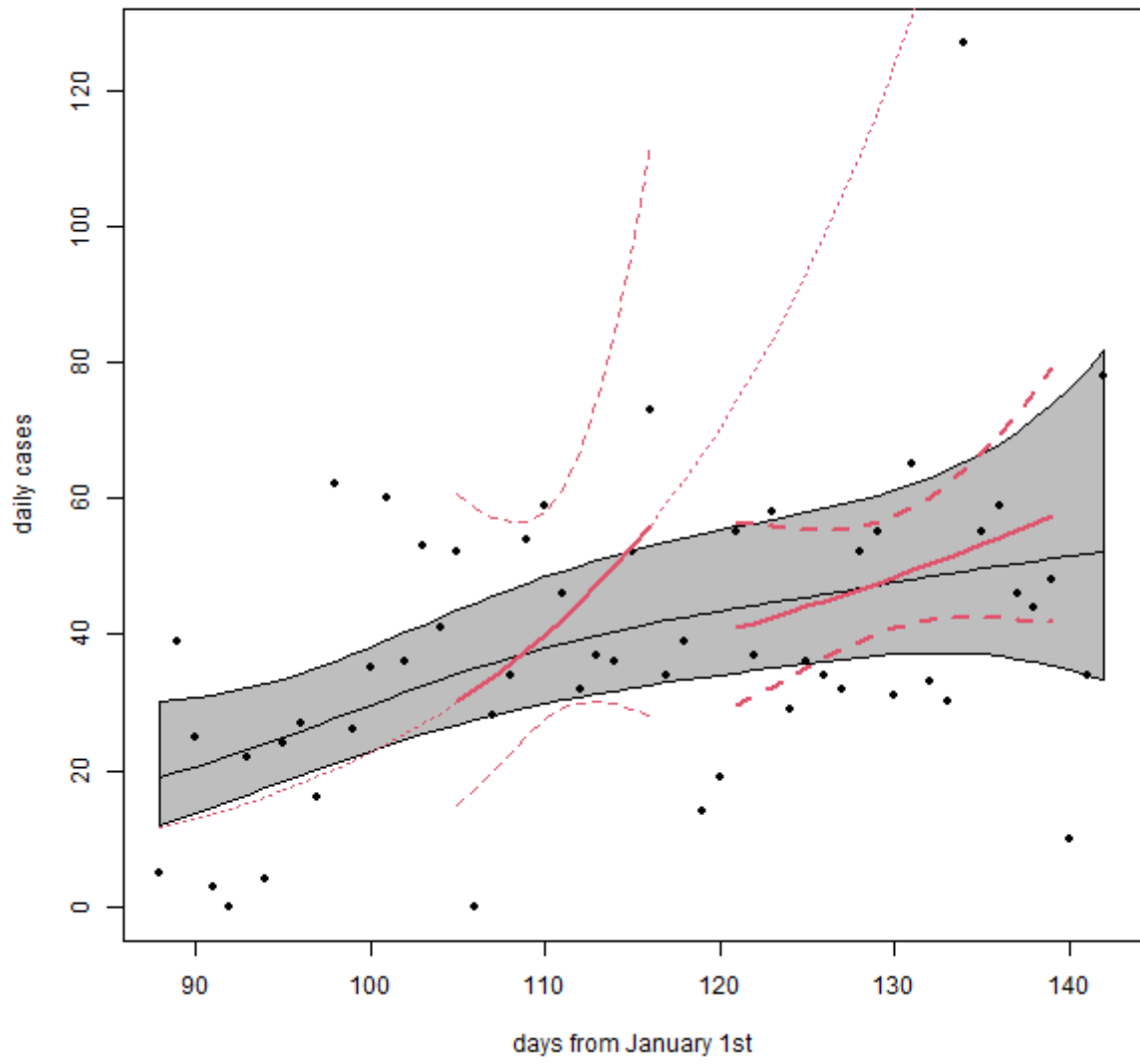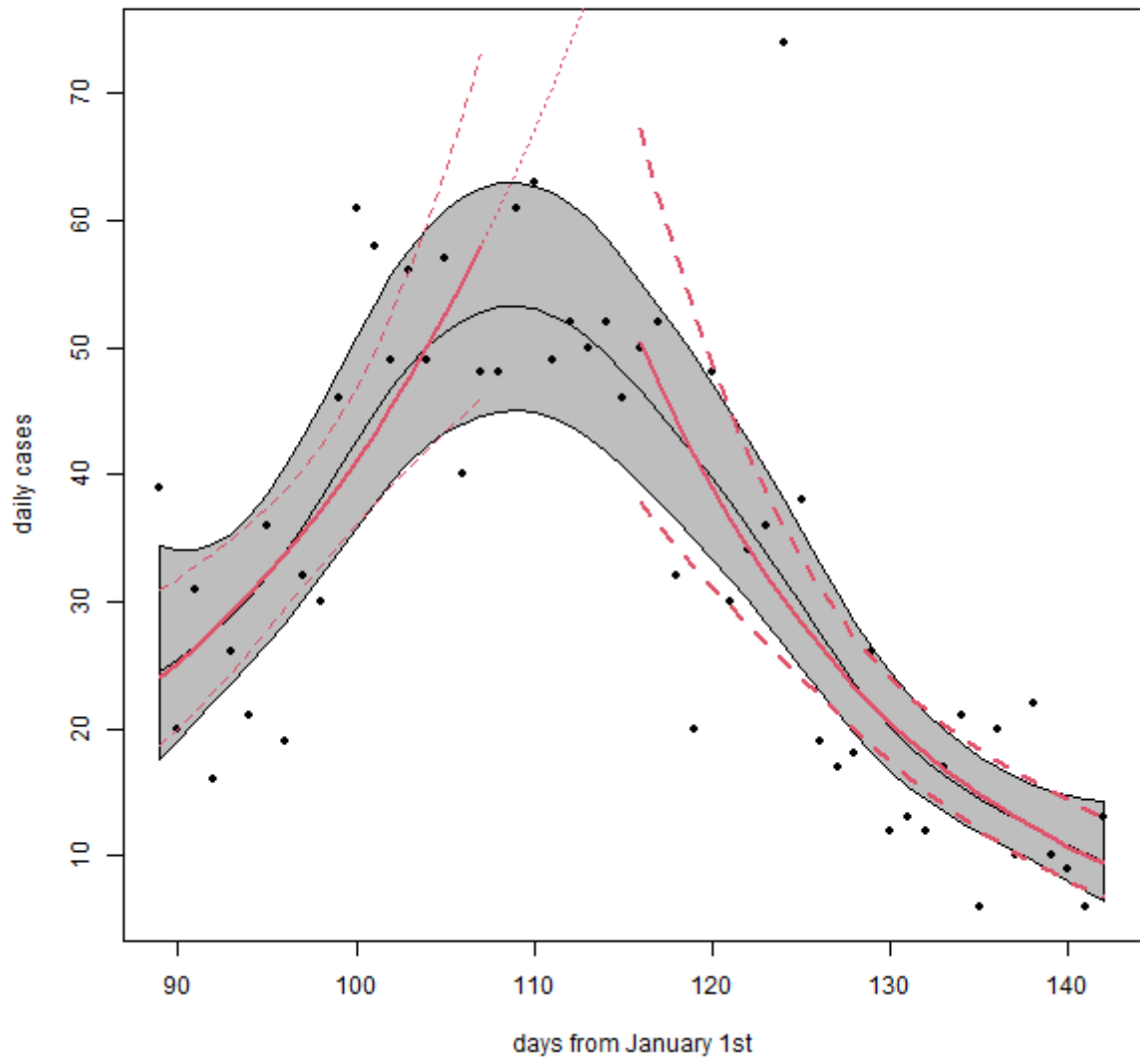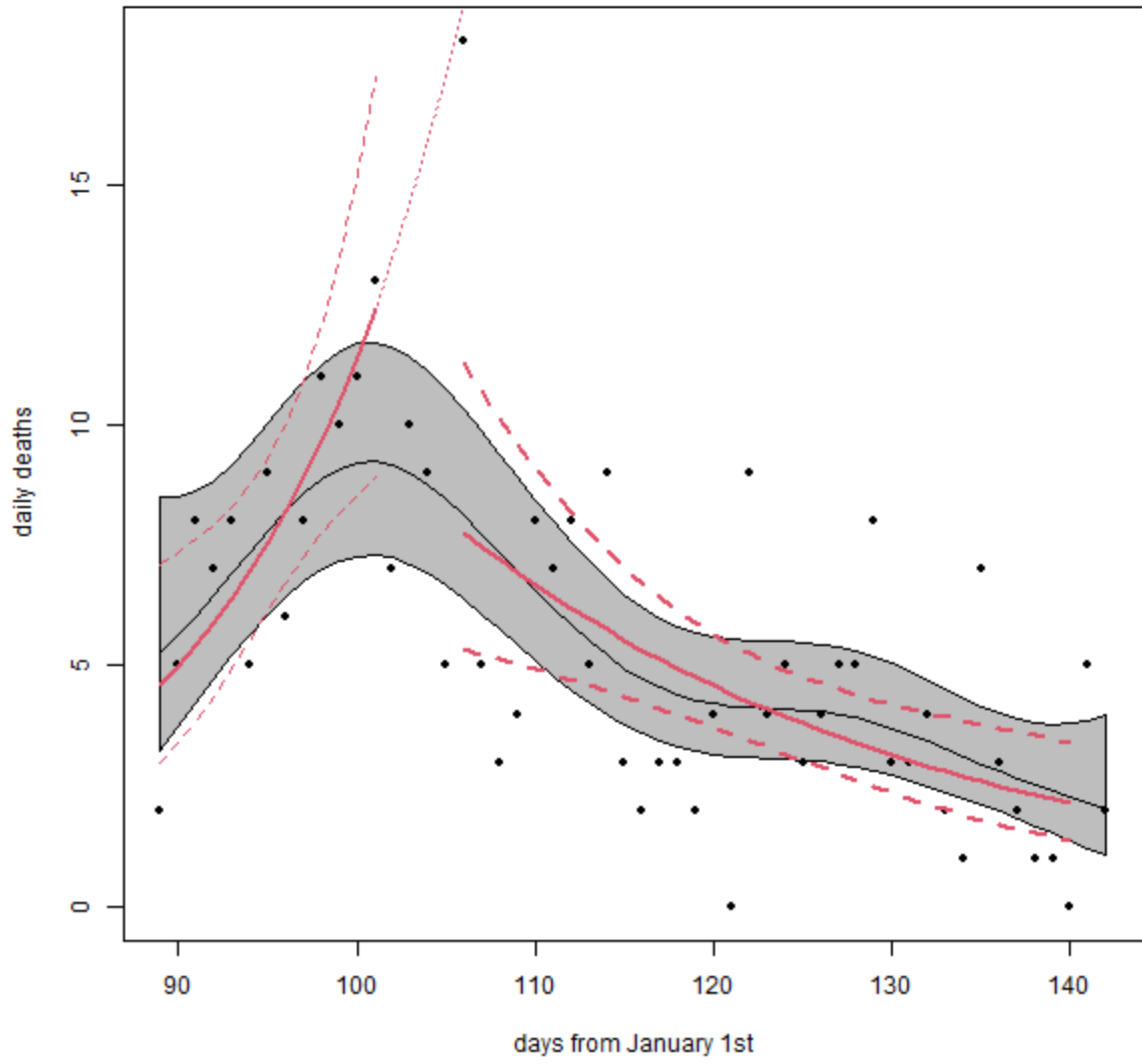
Democratic_Republic_of_the_Congo
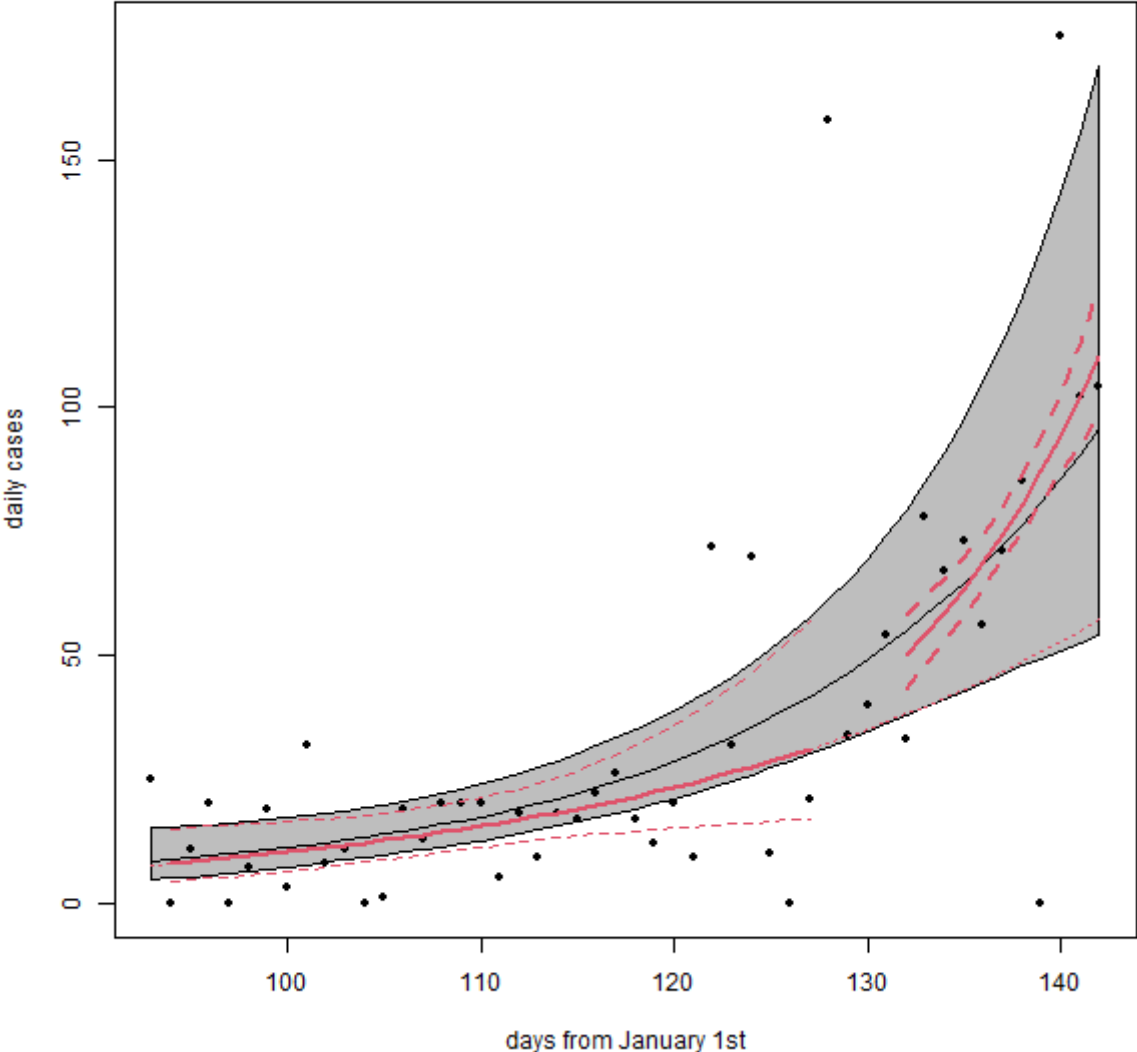
Denmark
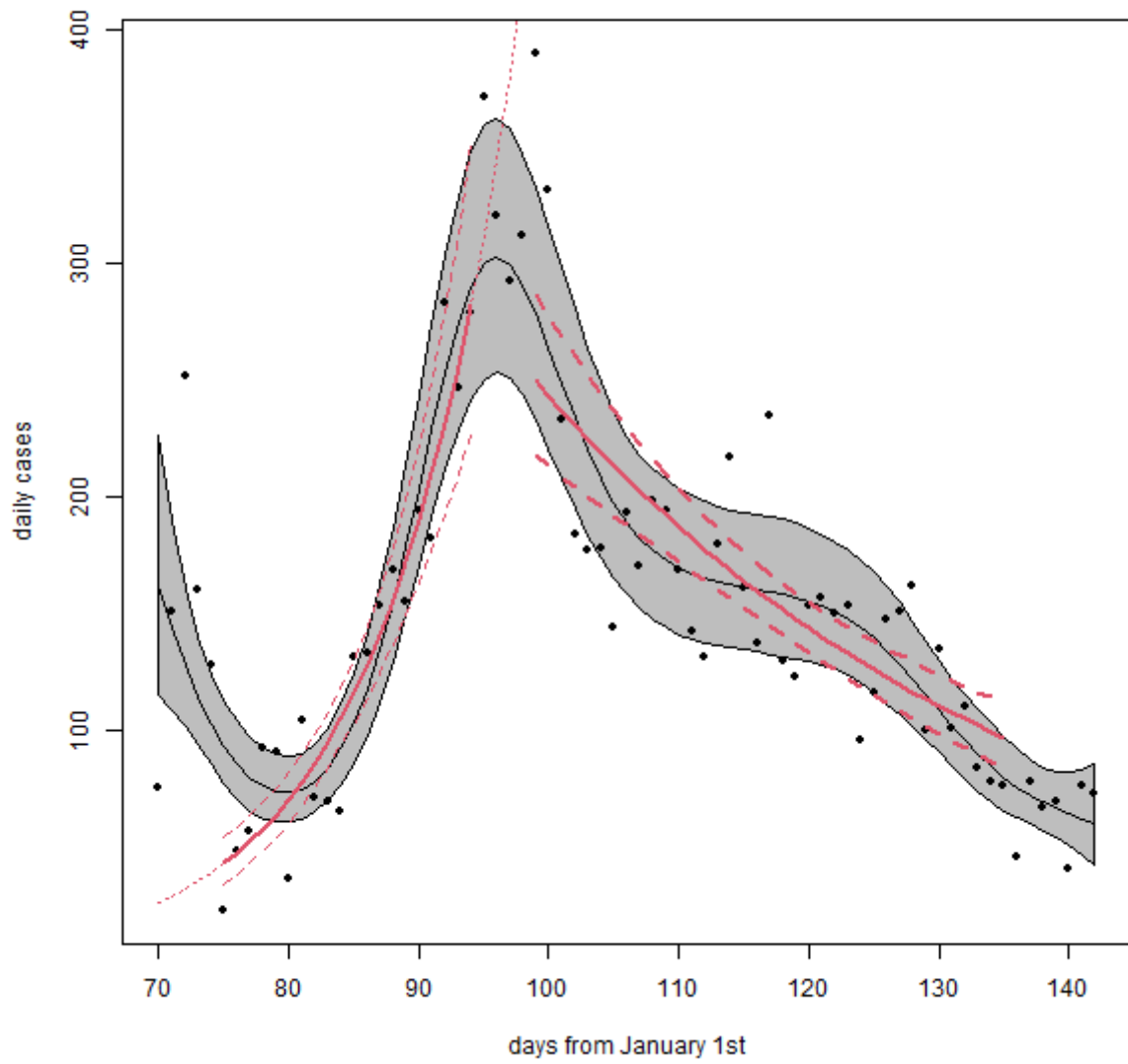
Djibouti
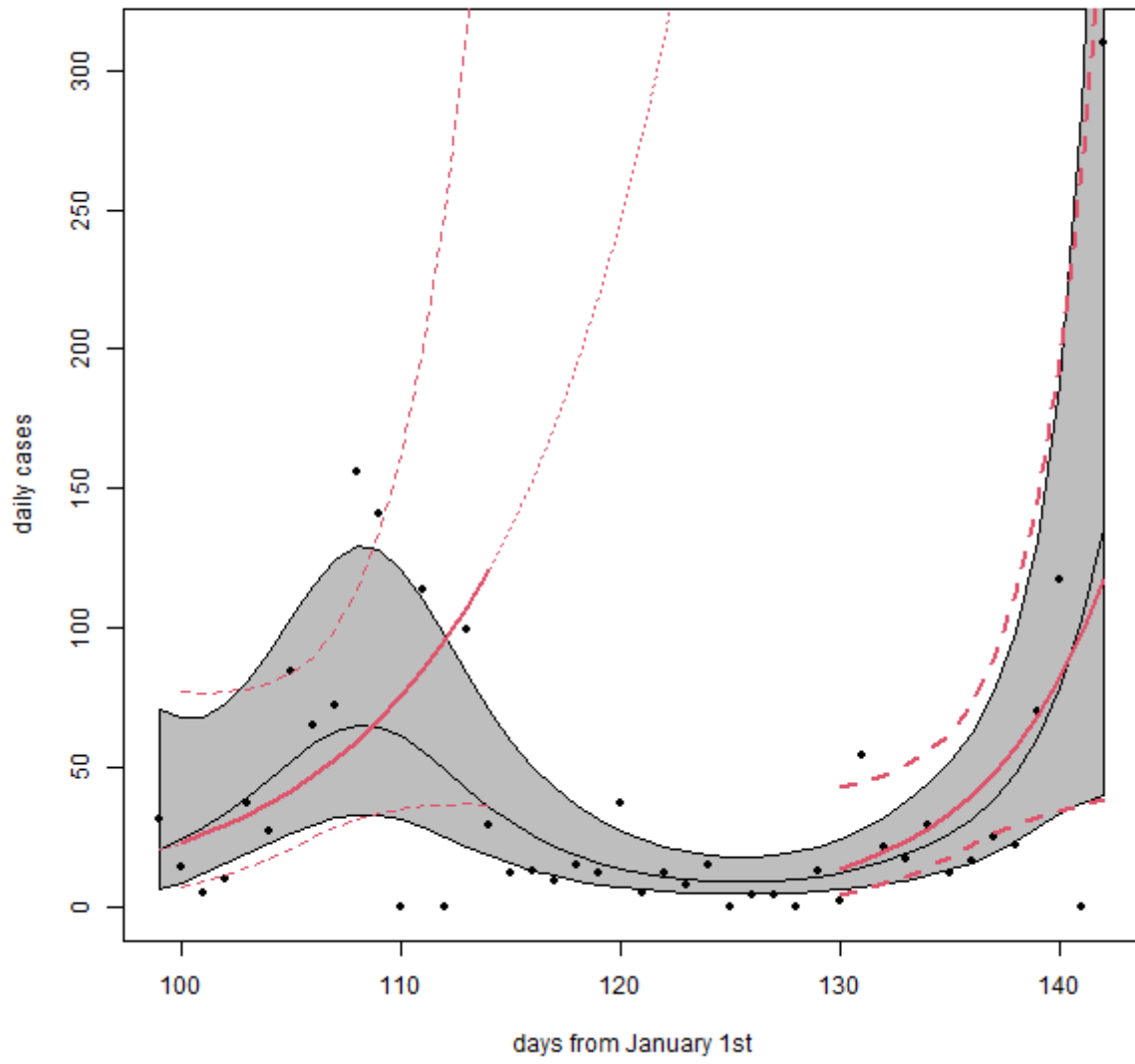
Dominican_Rep

DR_Congo

Ecuador

daily cases

days from January 1st
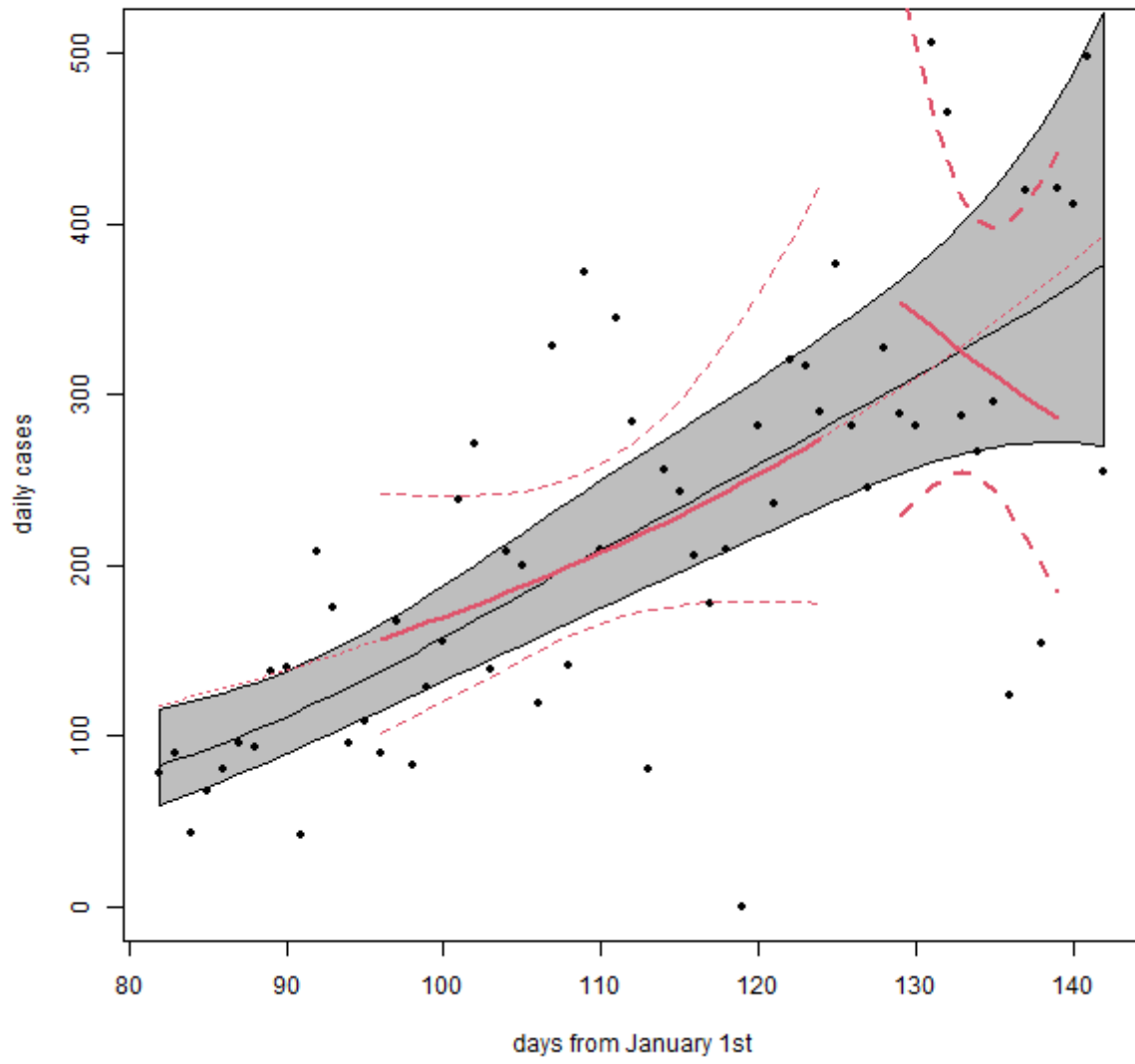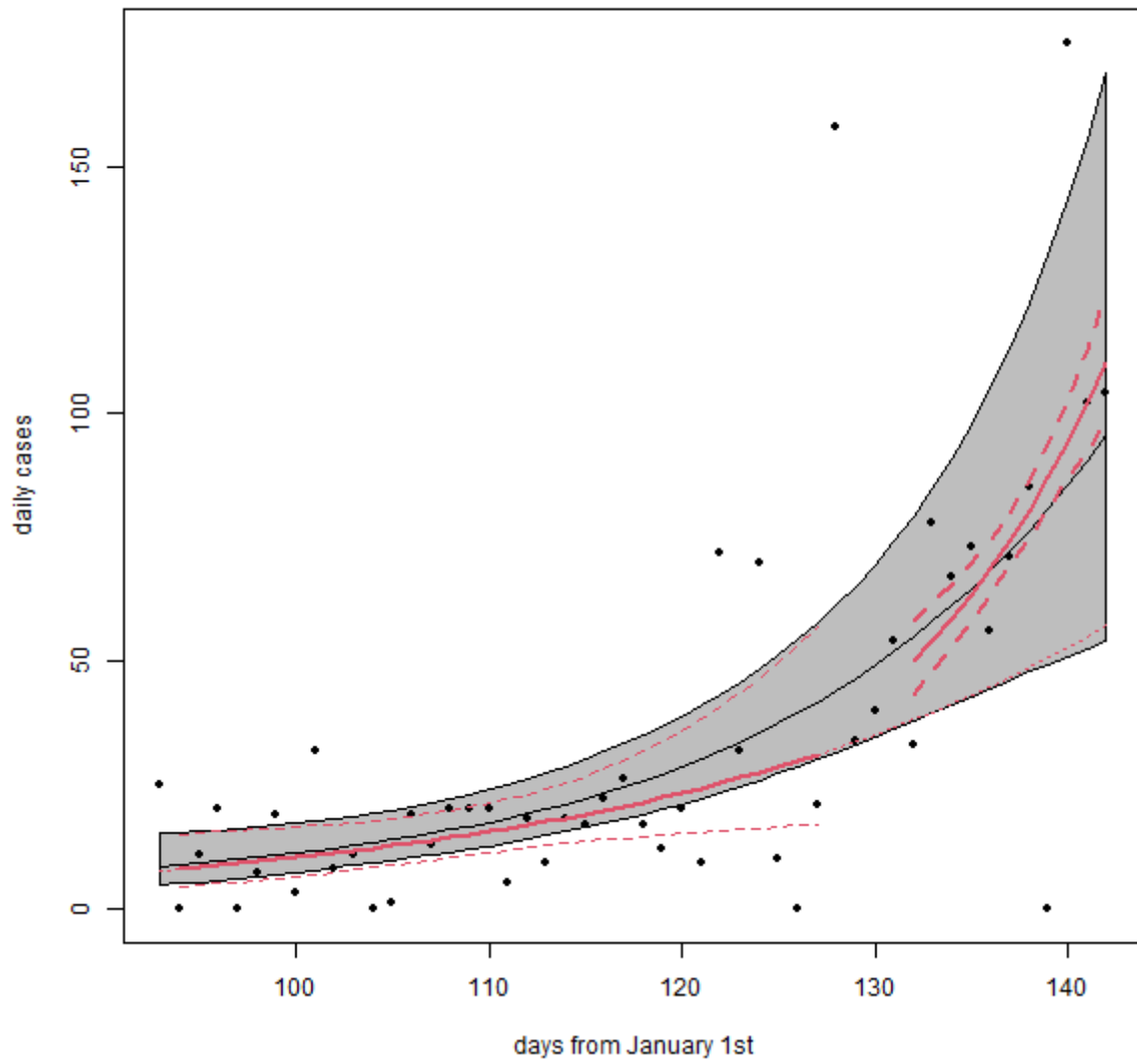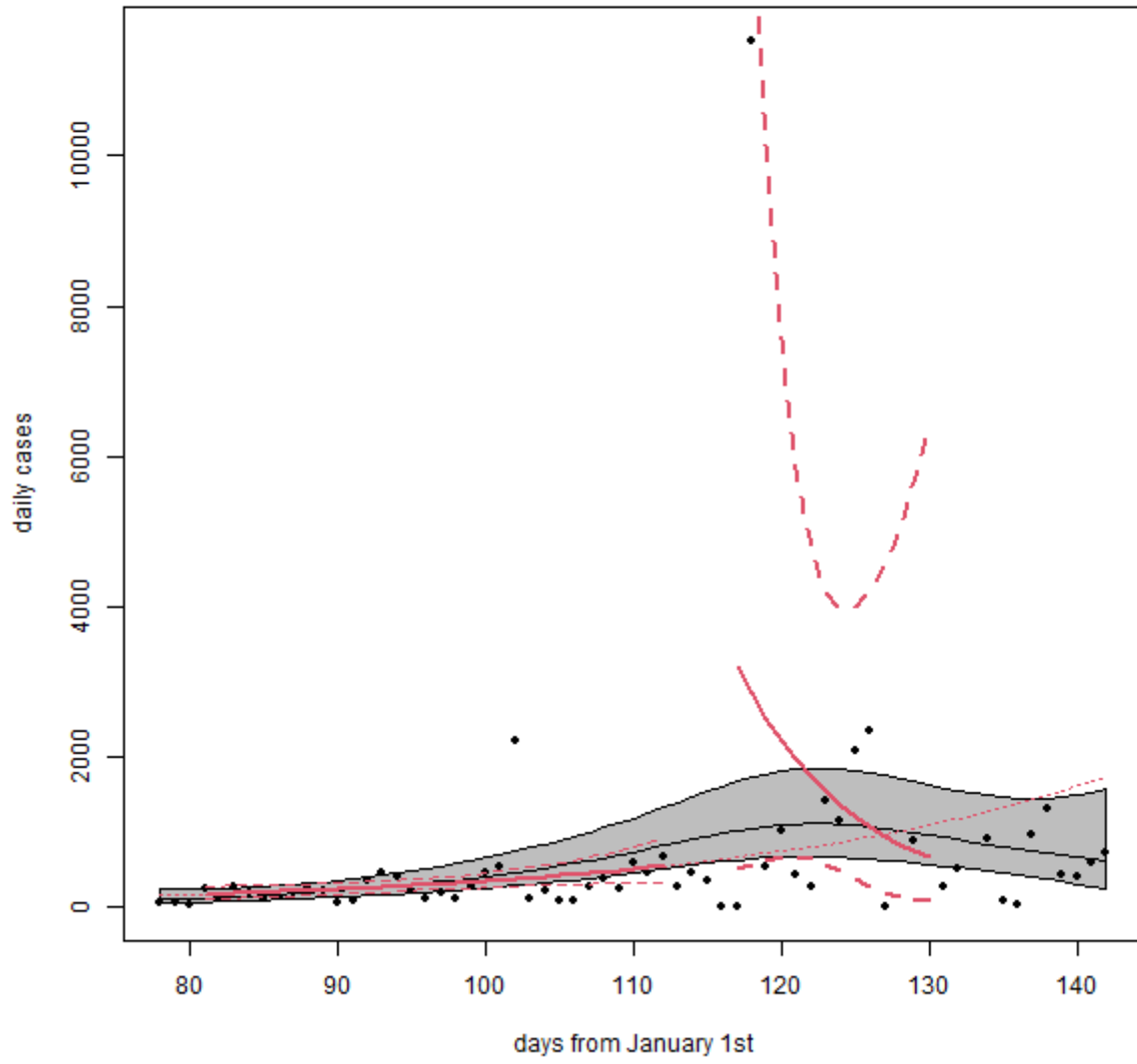
Ecuador

Egypt

Egypt

El_Salvador

daily cases

days from January 1st

Finland

daily cases

days from January 1st

Finland

daily deaths

days from January 1st

France

daily cases

days from January 1st

France

daily deaths

days from January 1st

Gabon

daily cases

days from January 1st
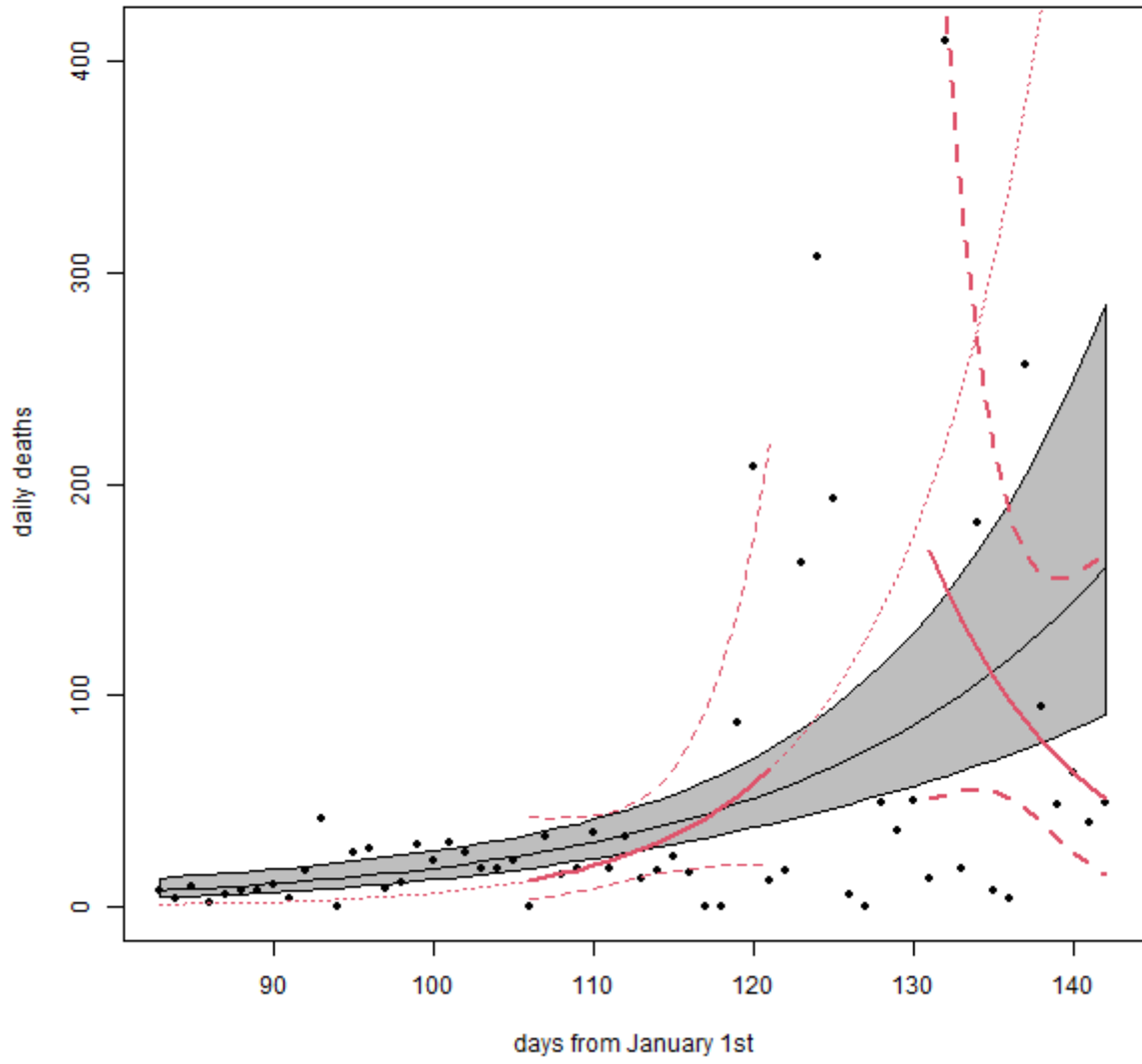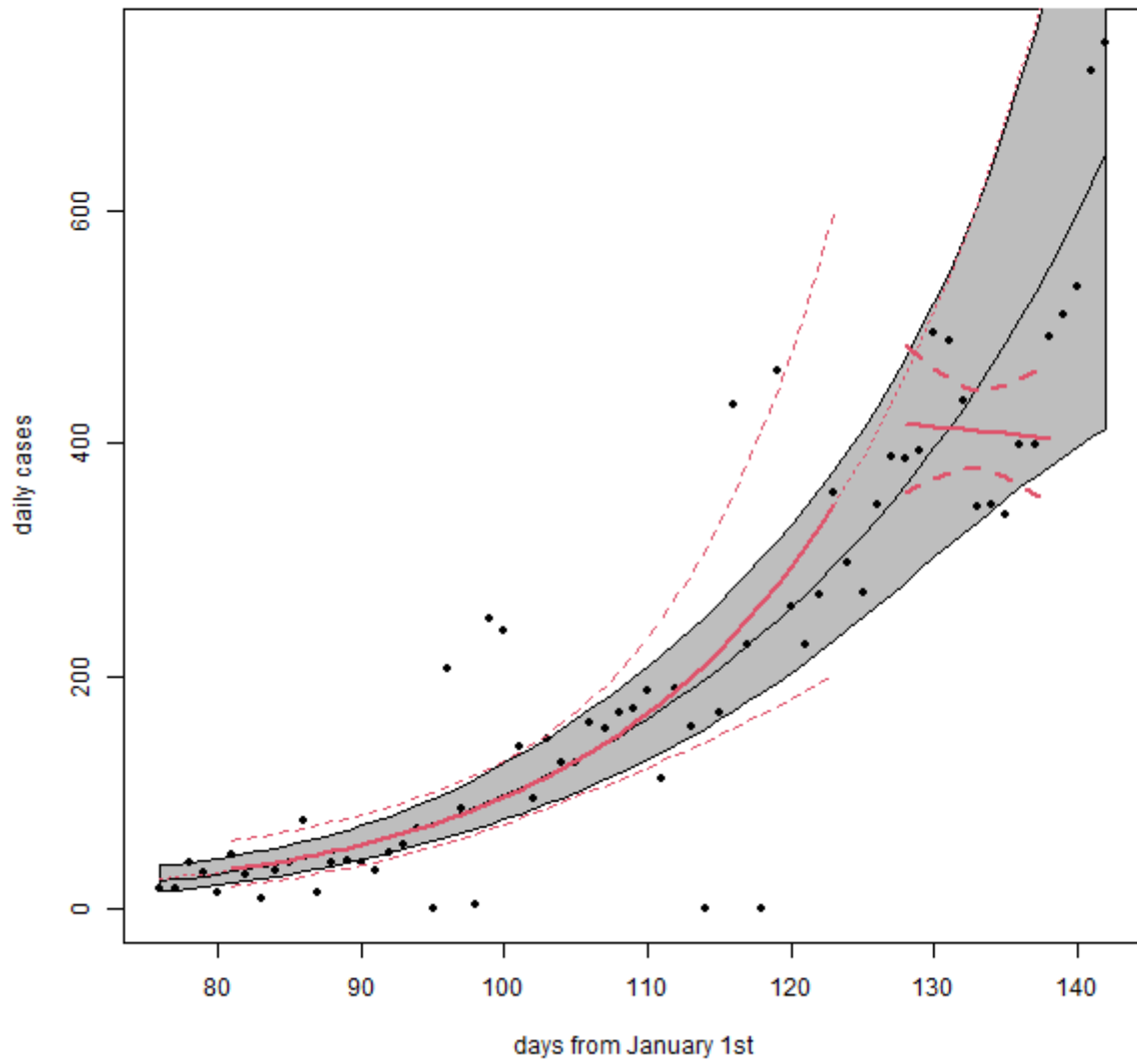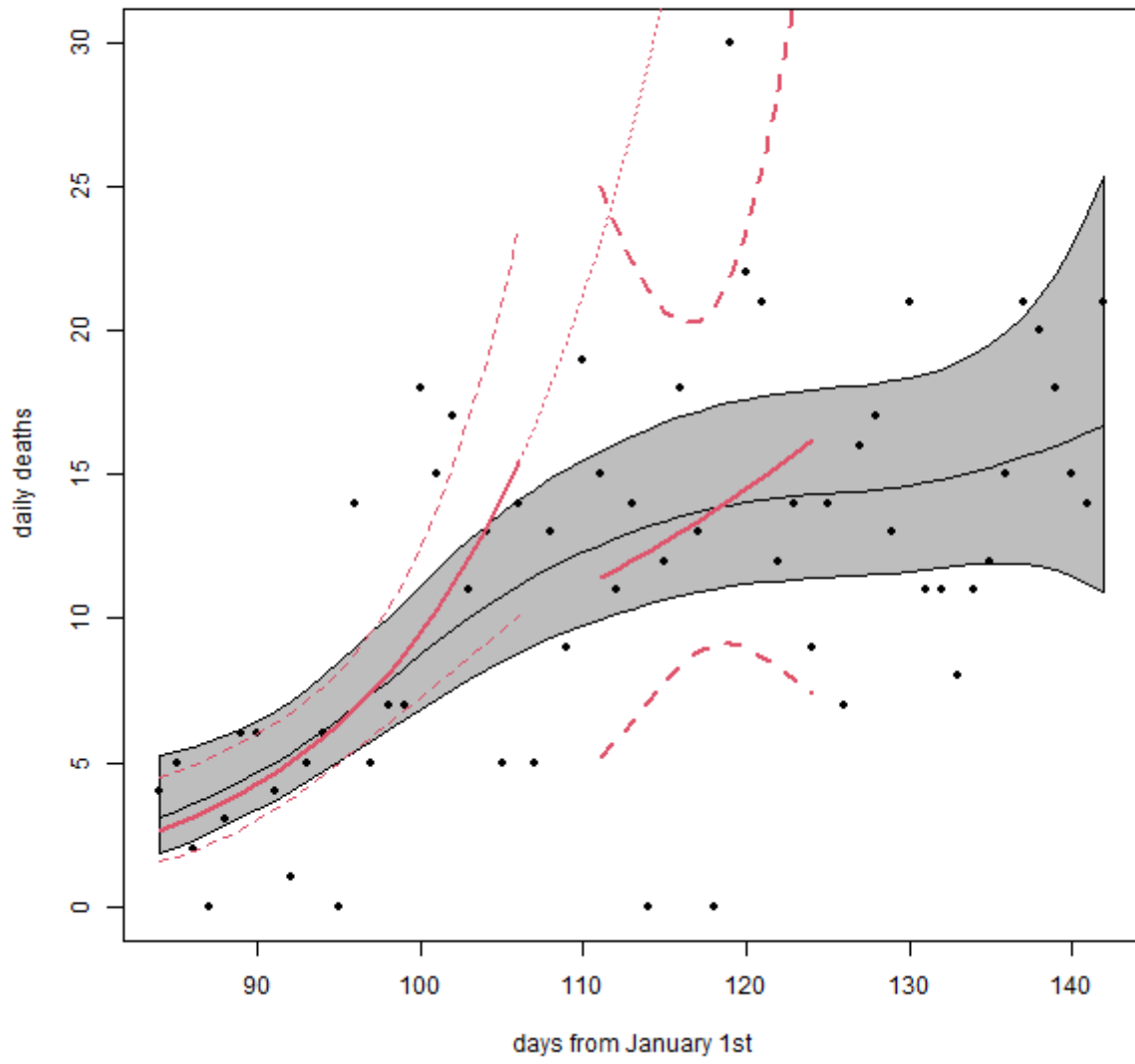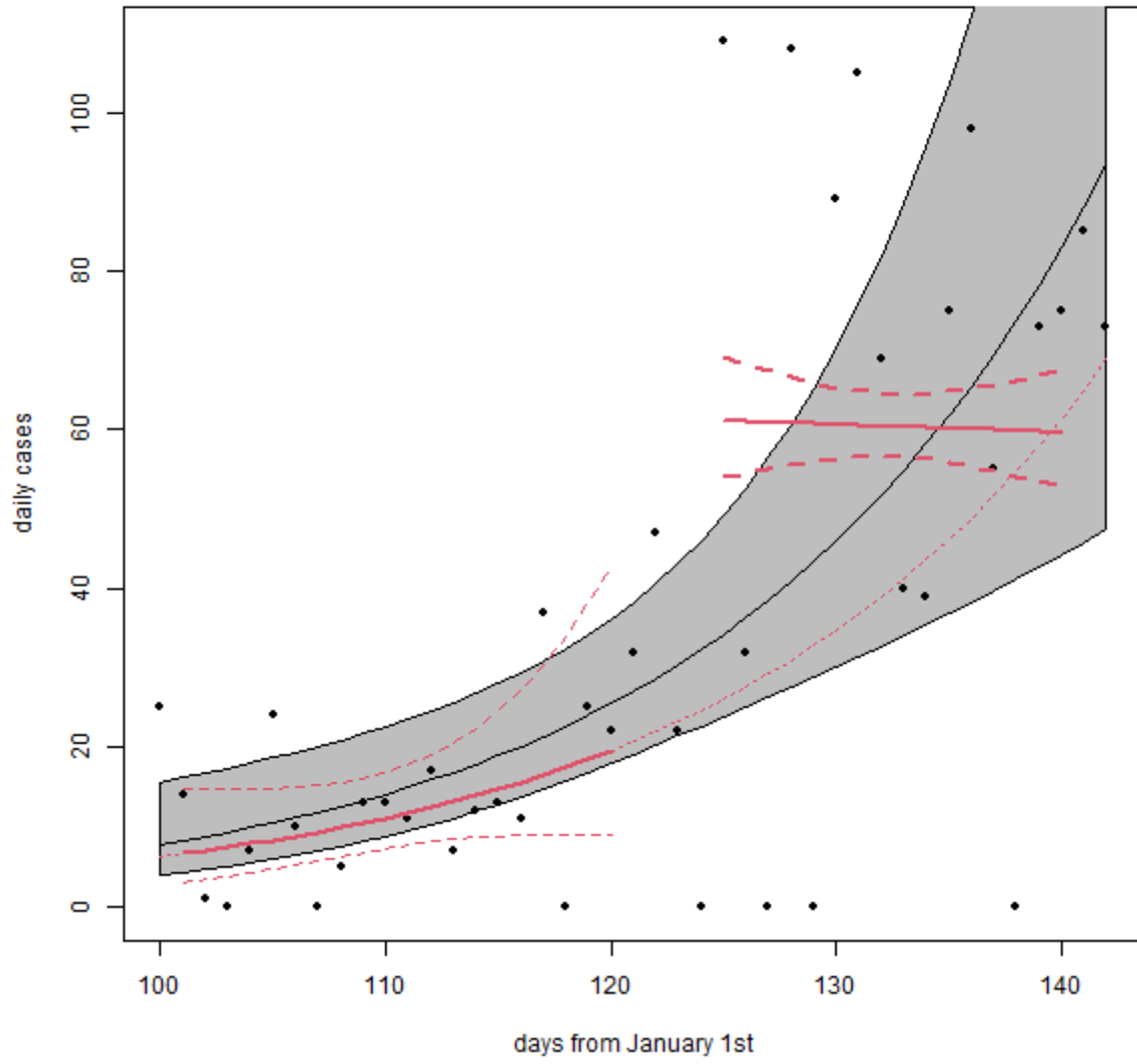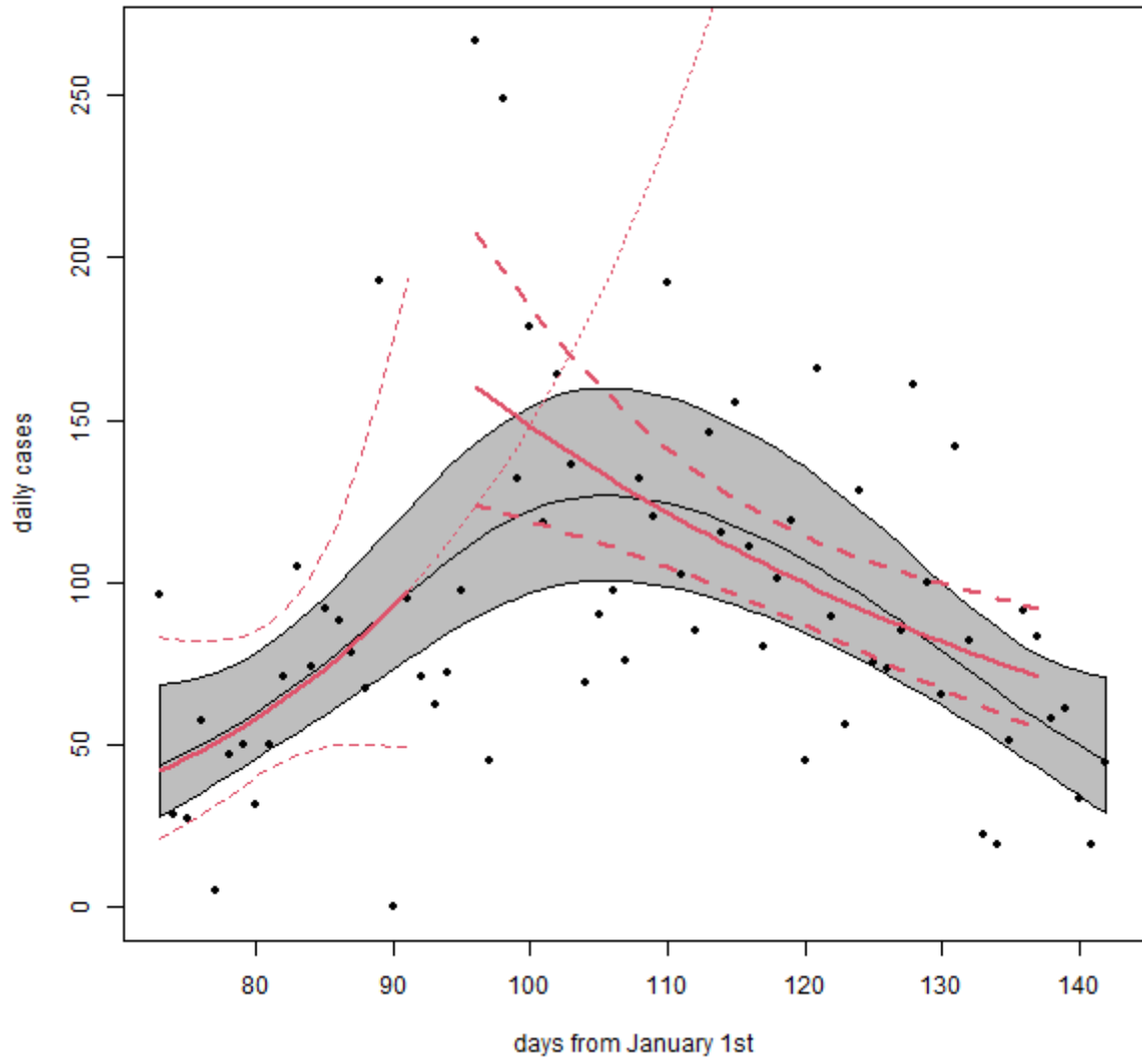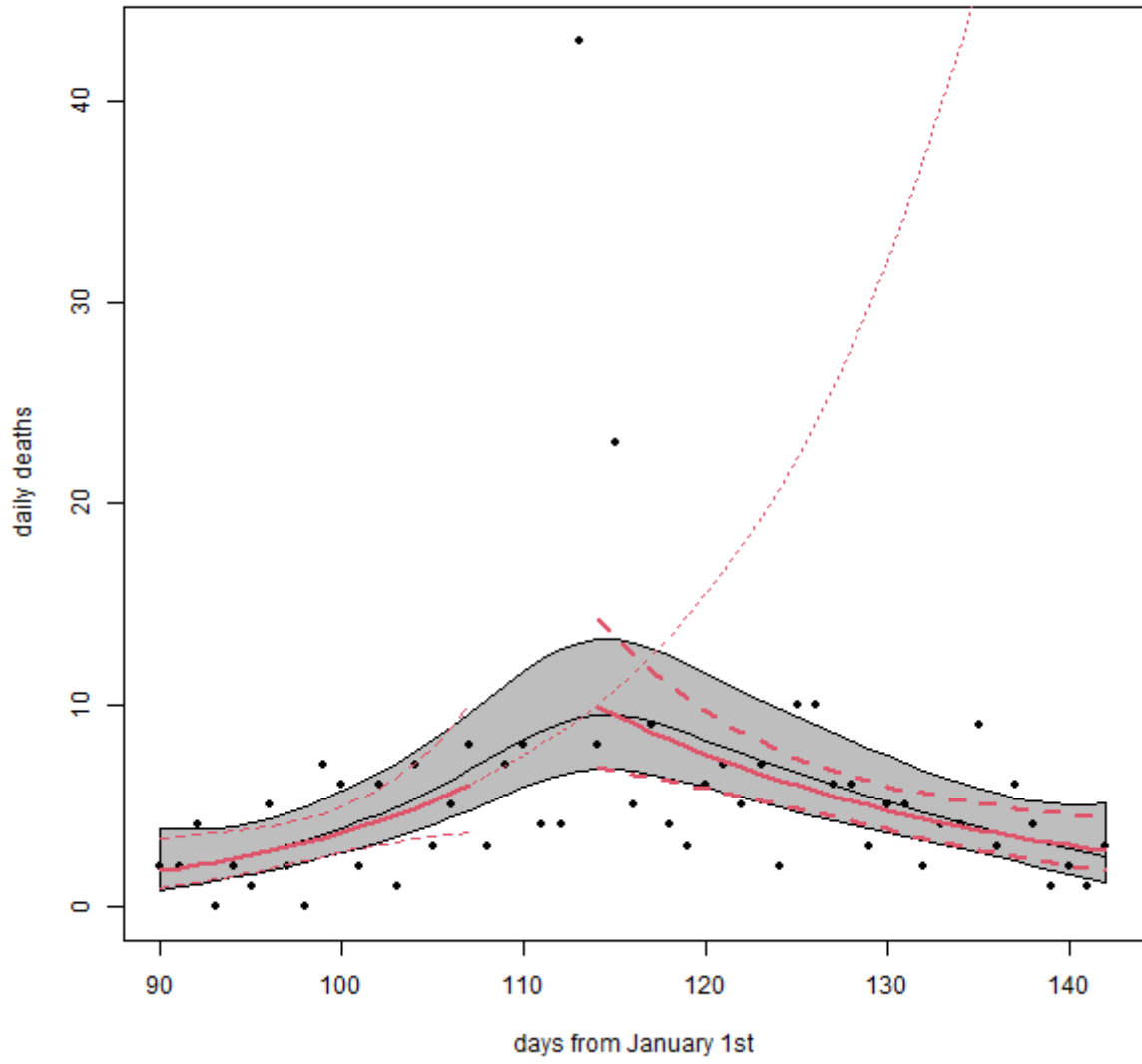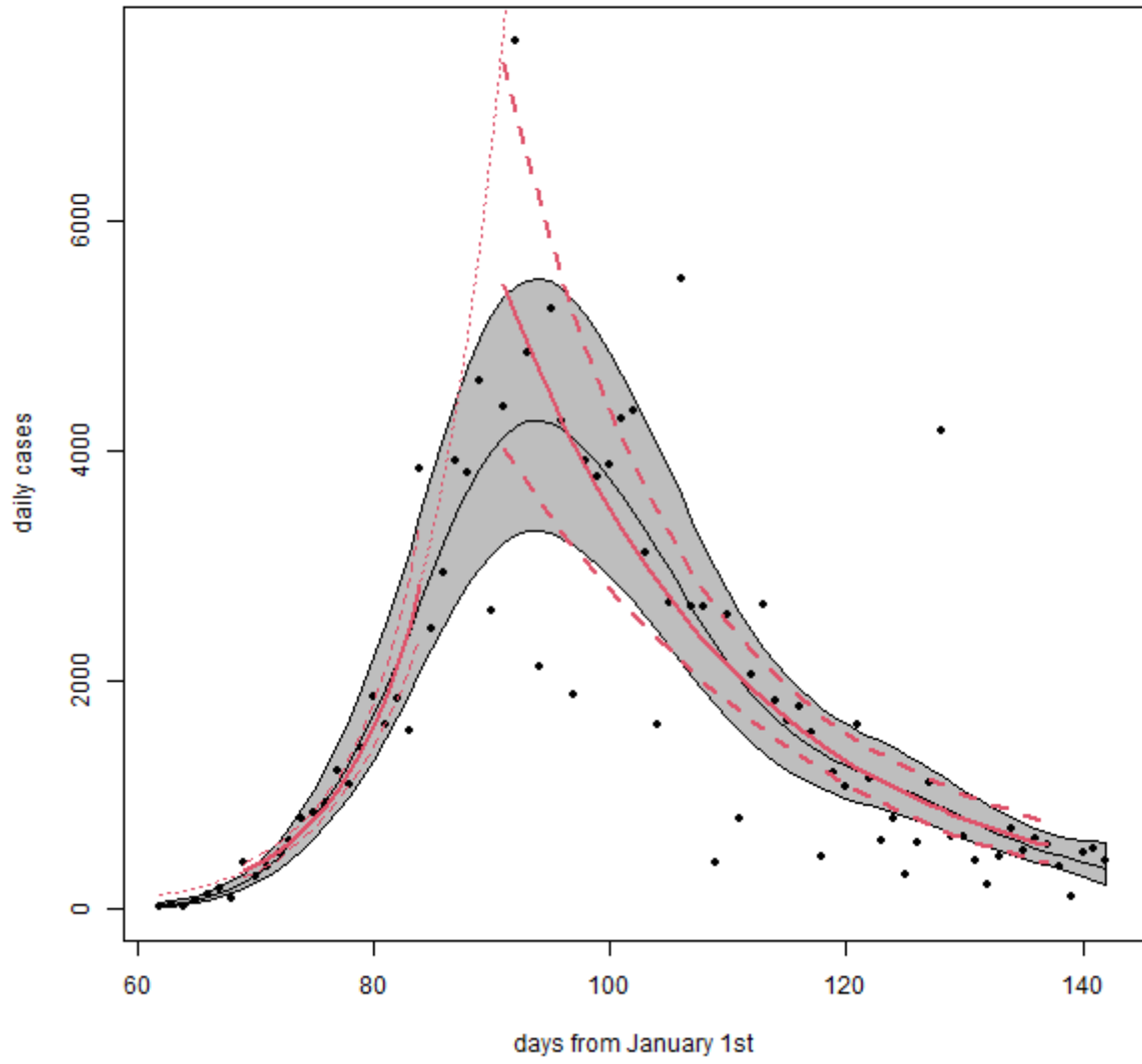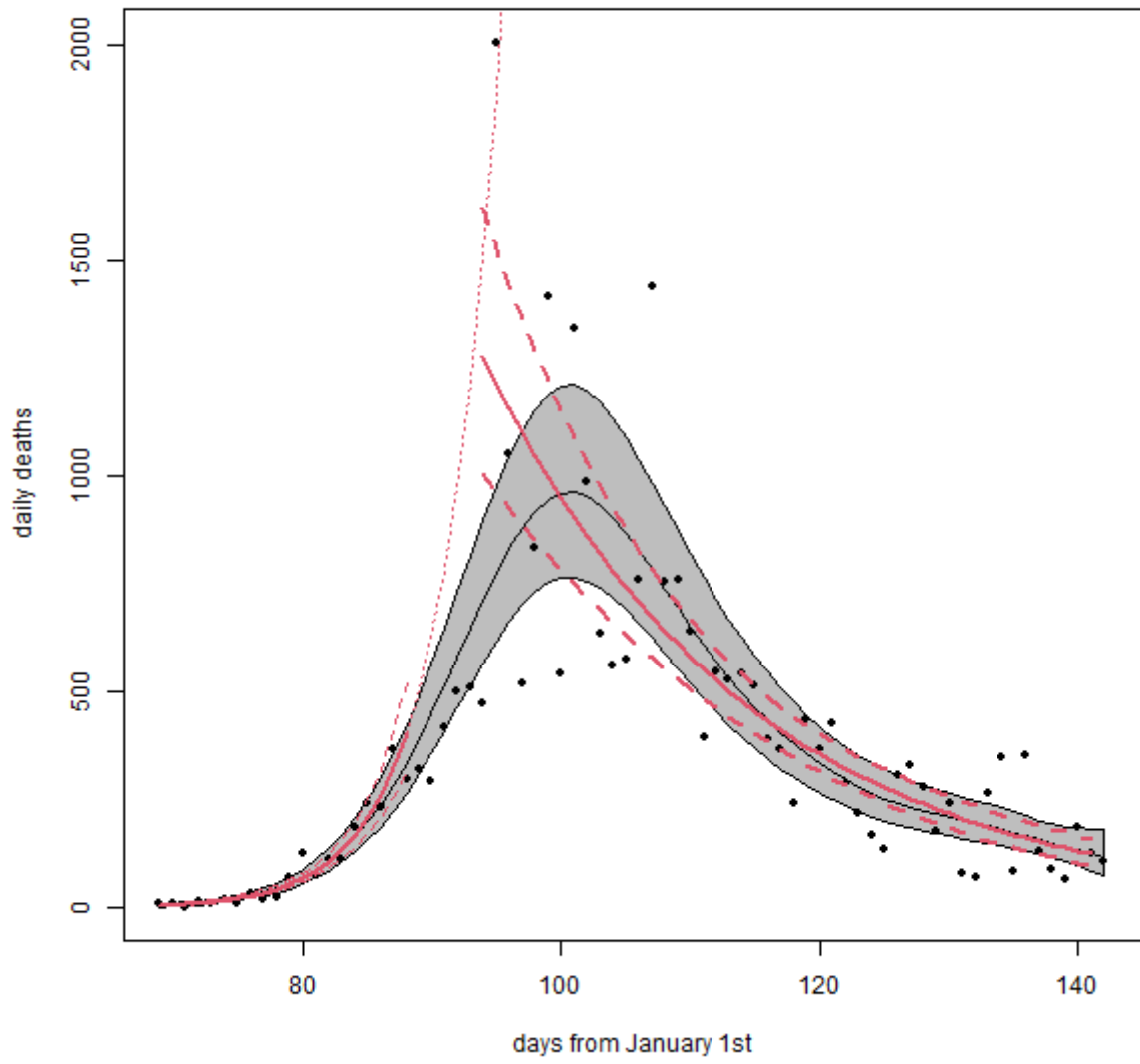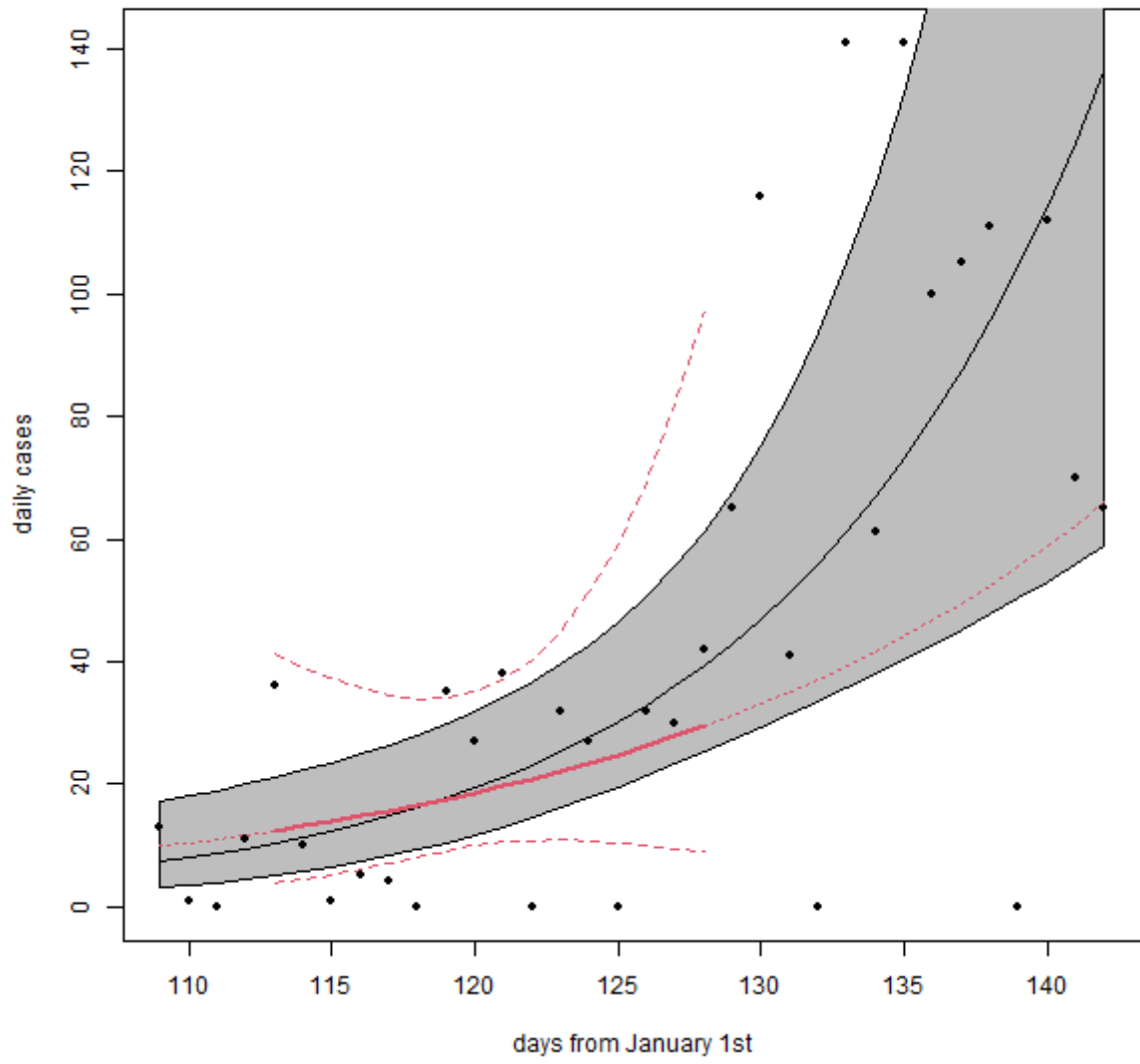
Germany

Germany

Ghana

Greece

daily cases

days from January 1st

Guatemala

daily cases

days from January 1st

Guinea

daily cases

days from January 1st

Honduras

Honduras

daily deaths

days from January 1st
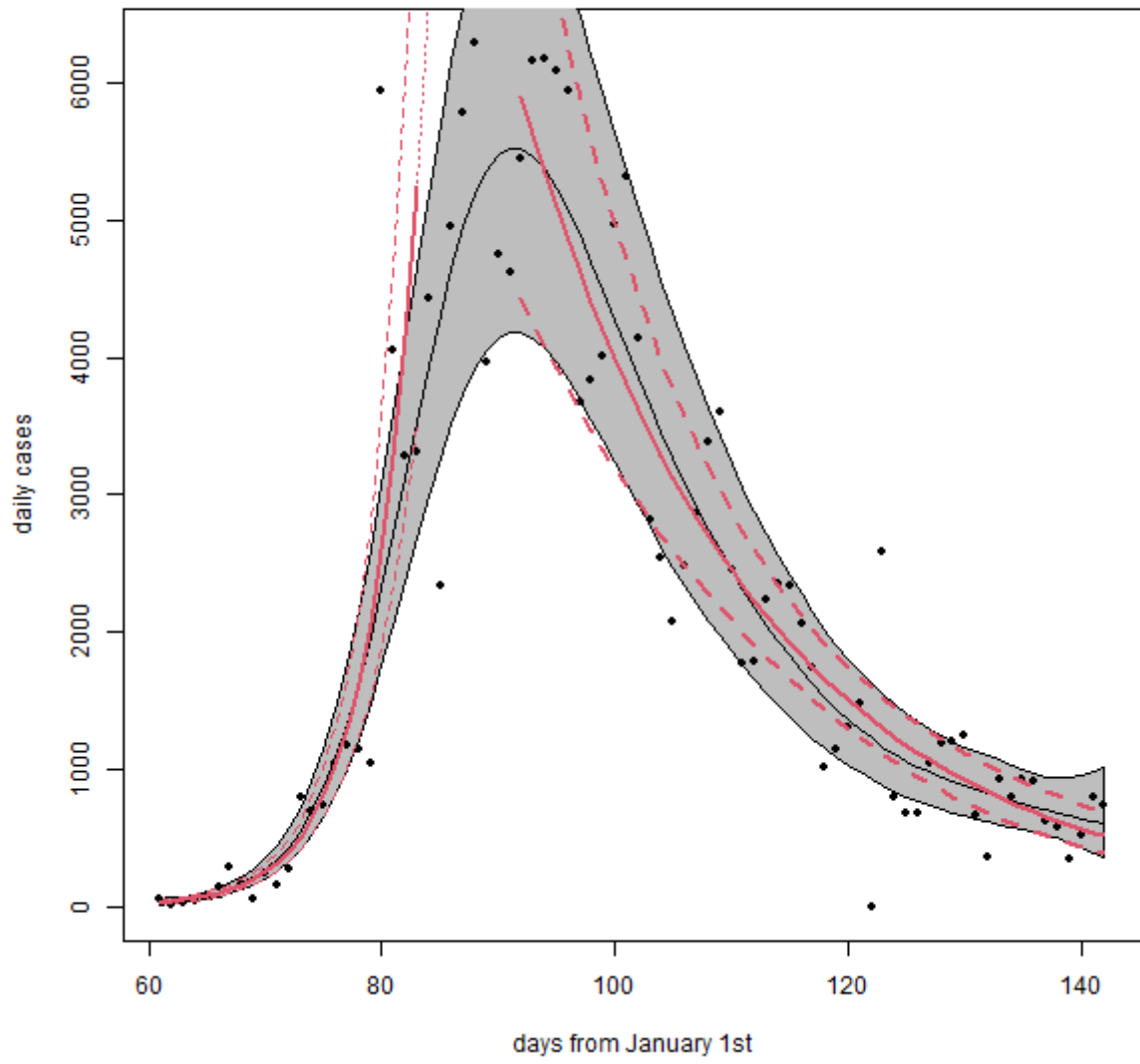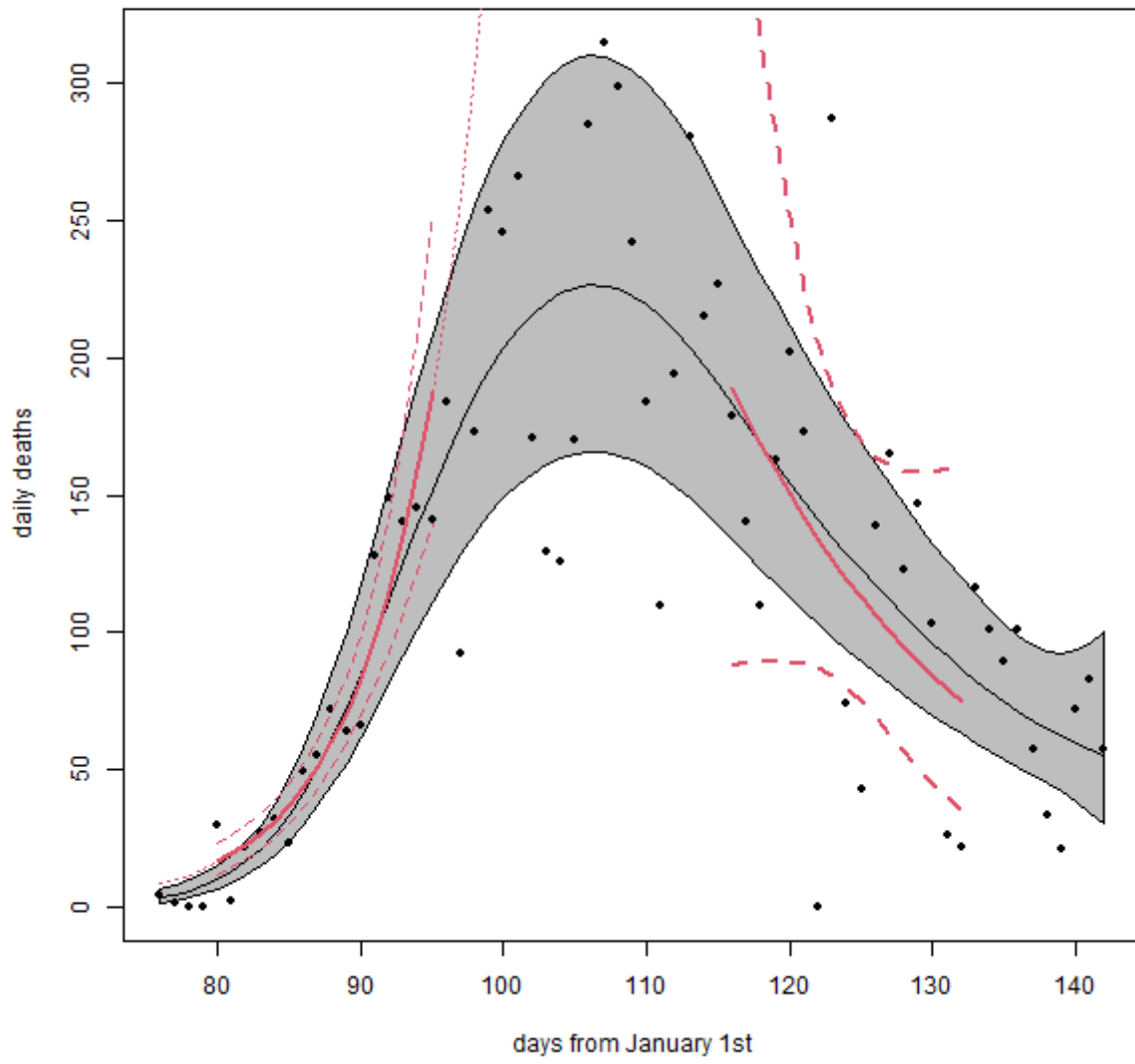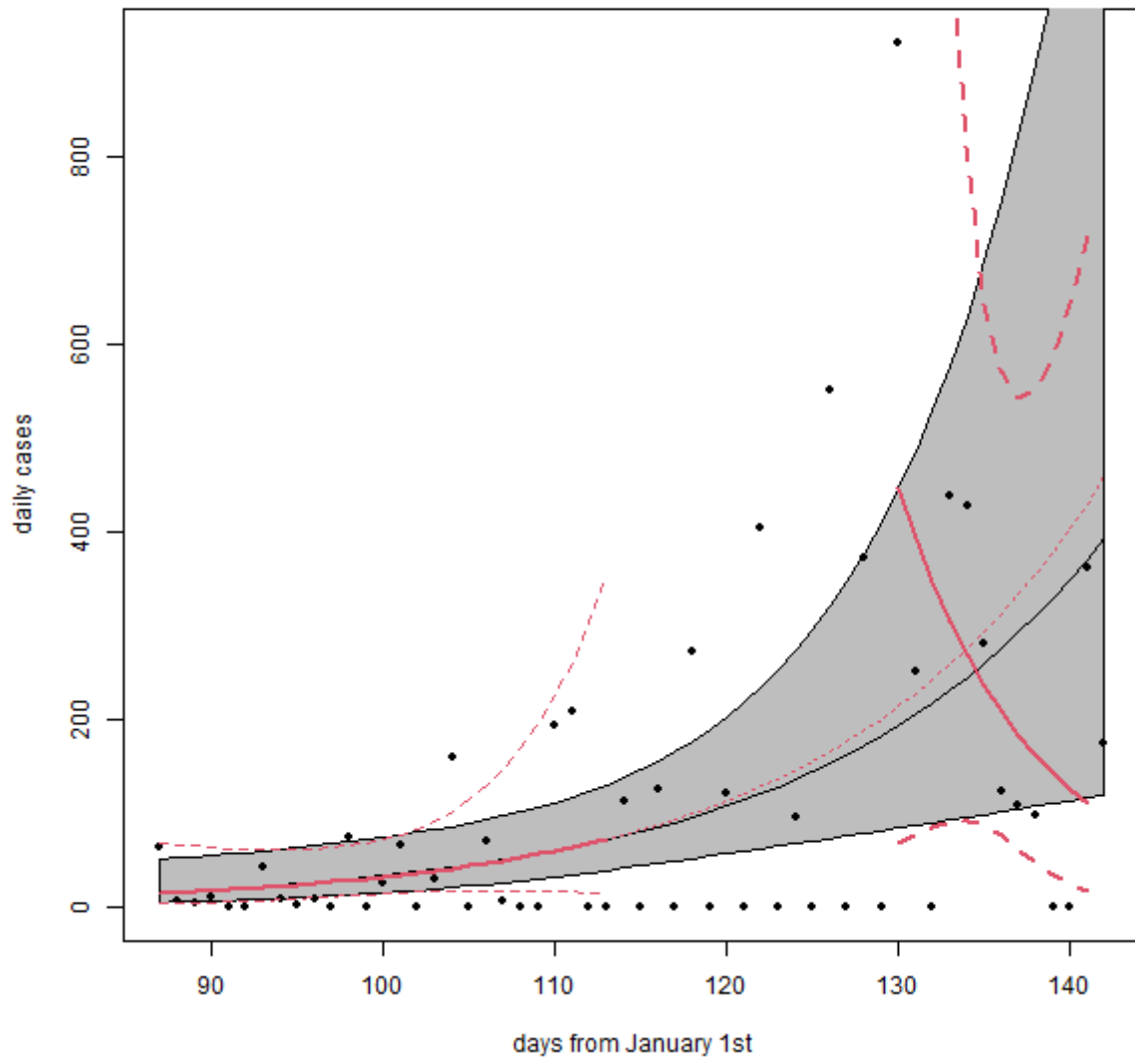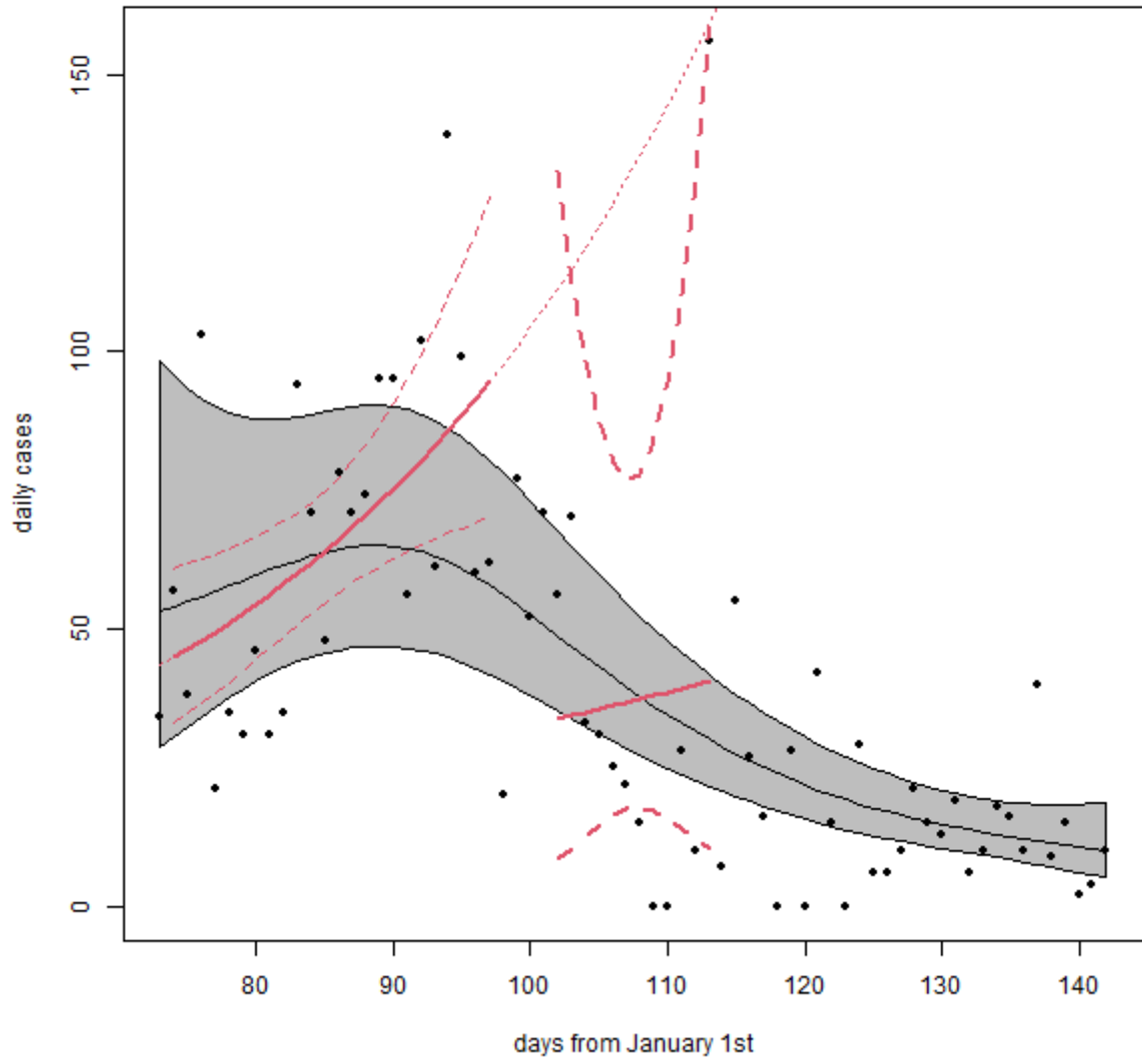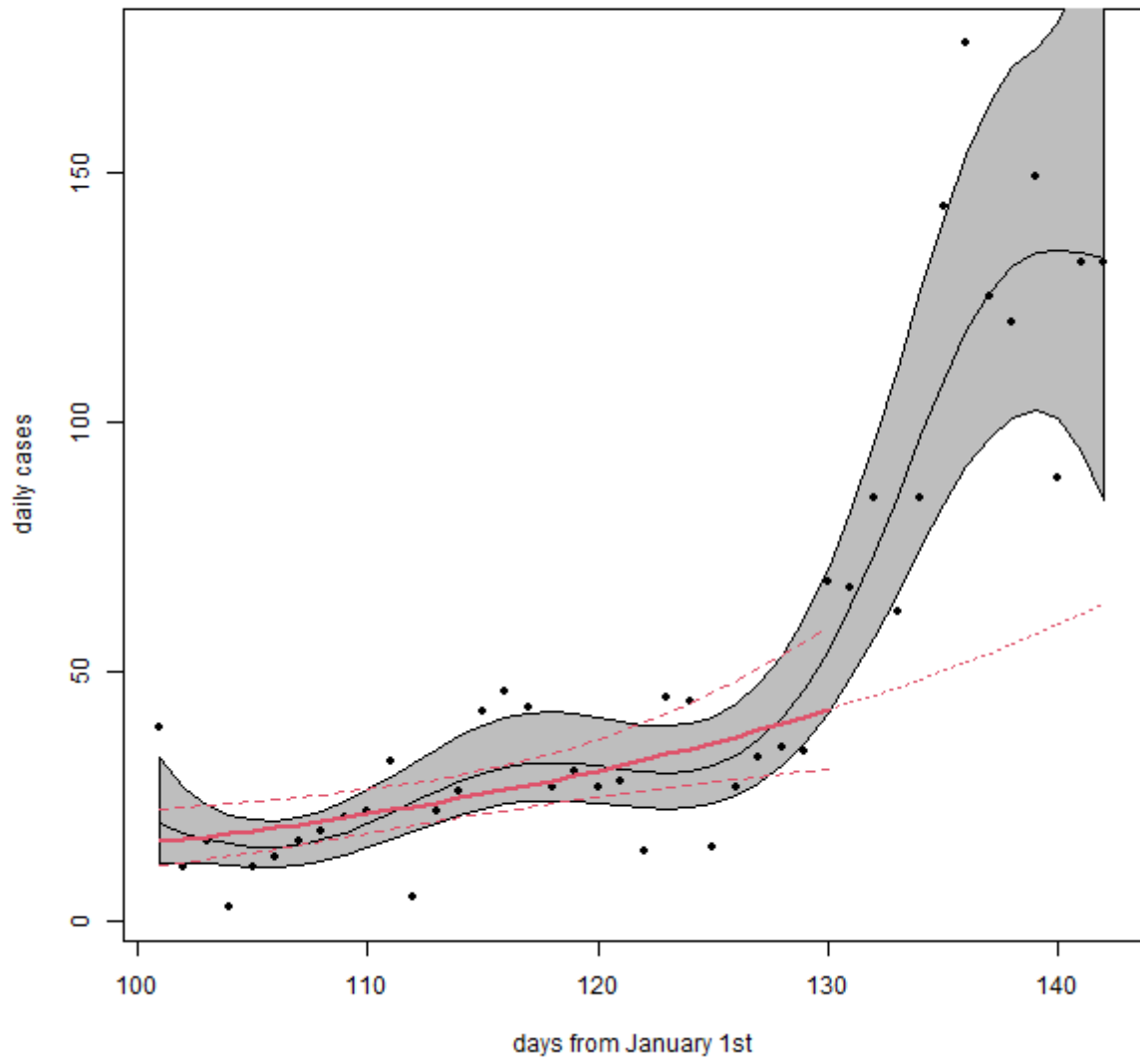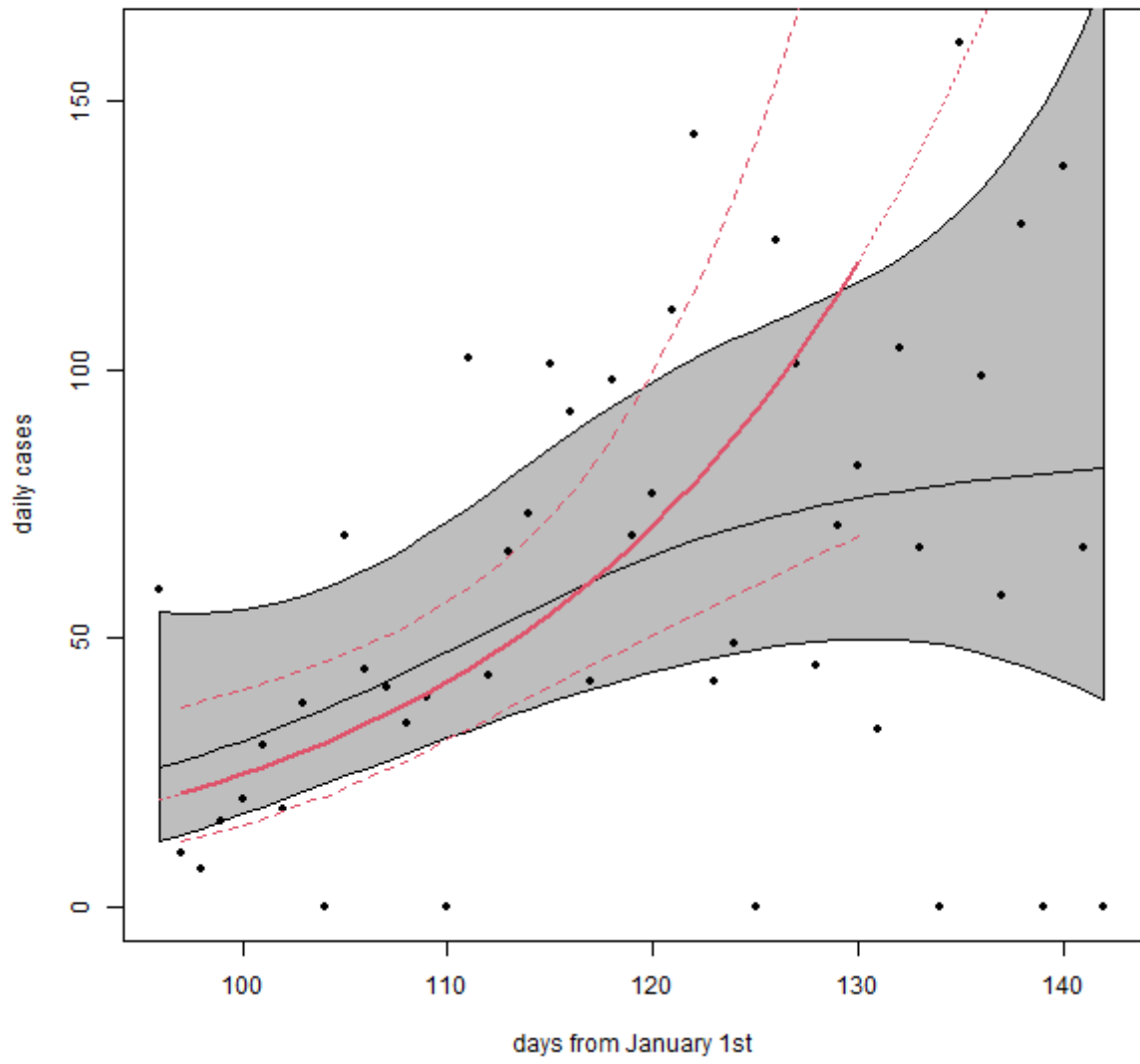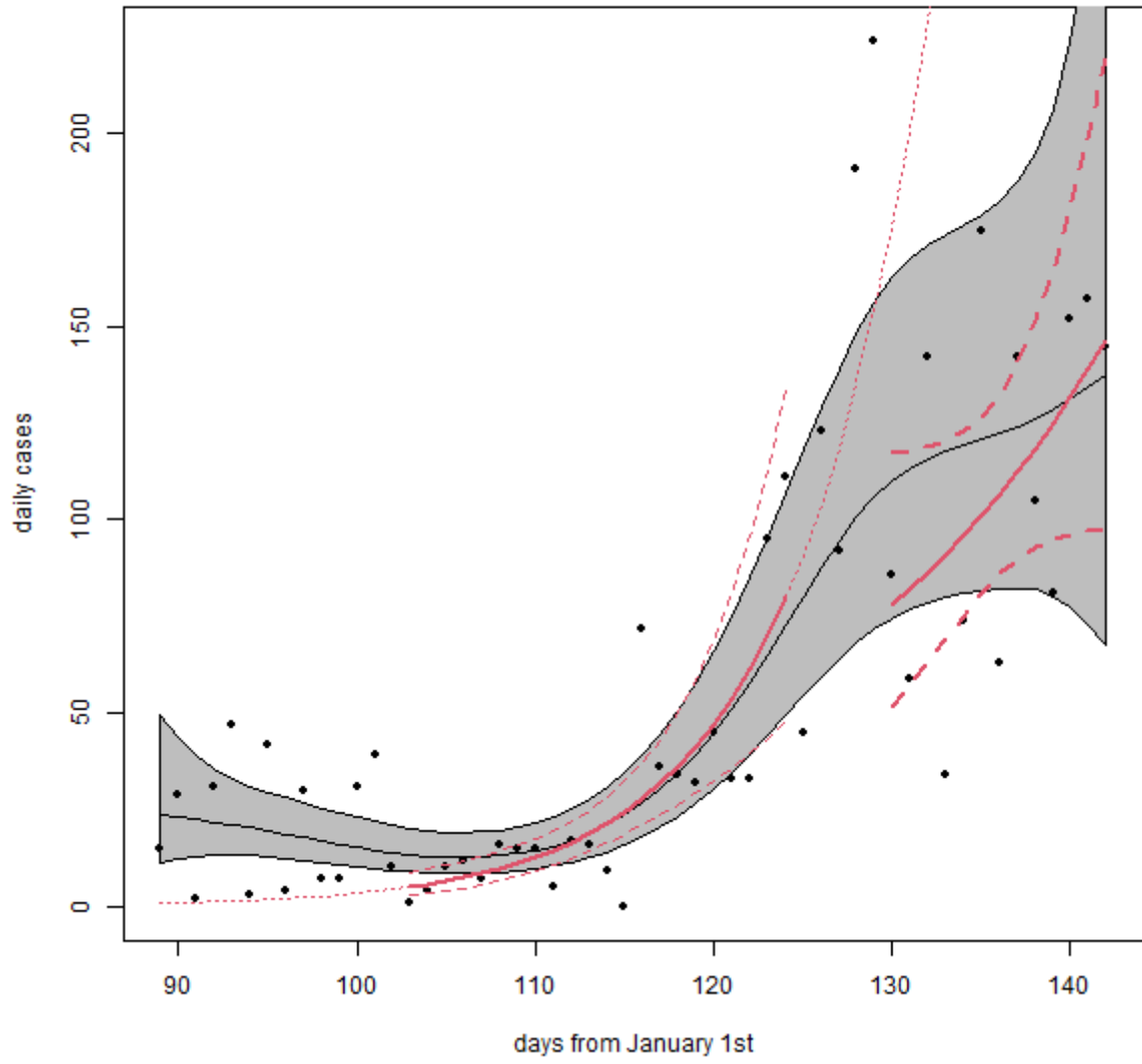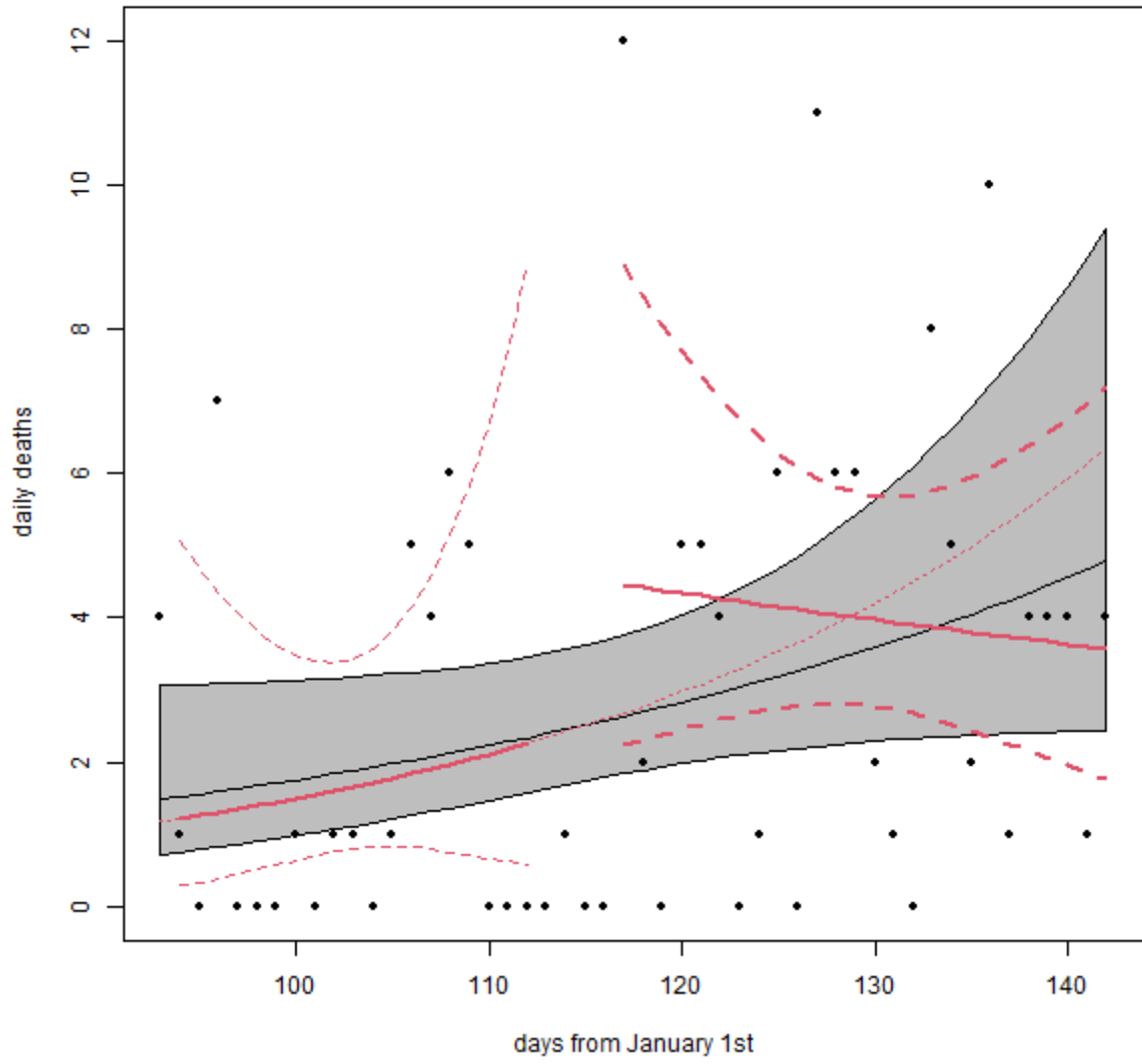
Hungary

Hungary

Iceland

days from January 1st

India

daily cases

days from January 1st

India

daily deaths

days from January 1st
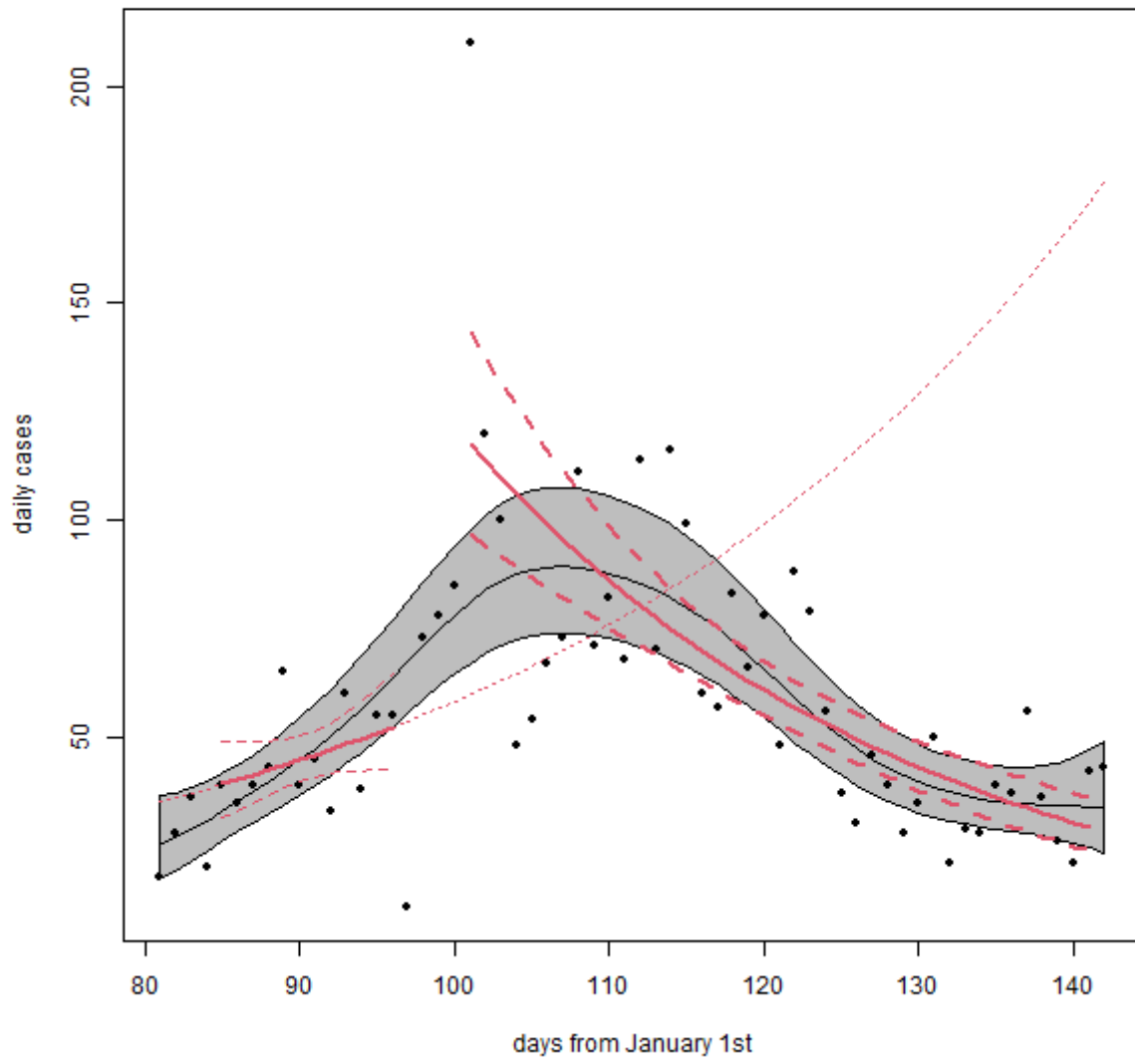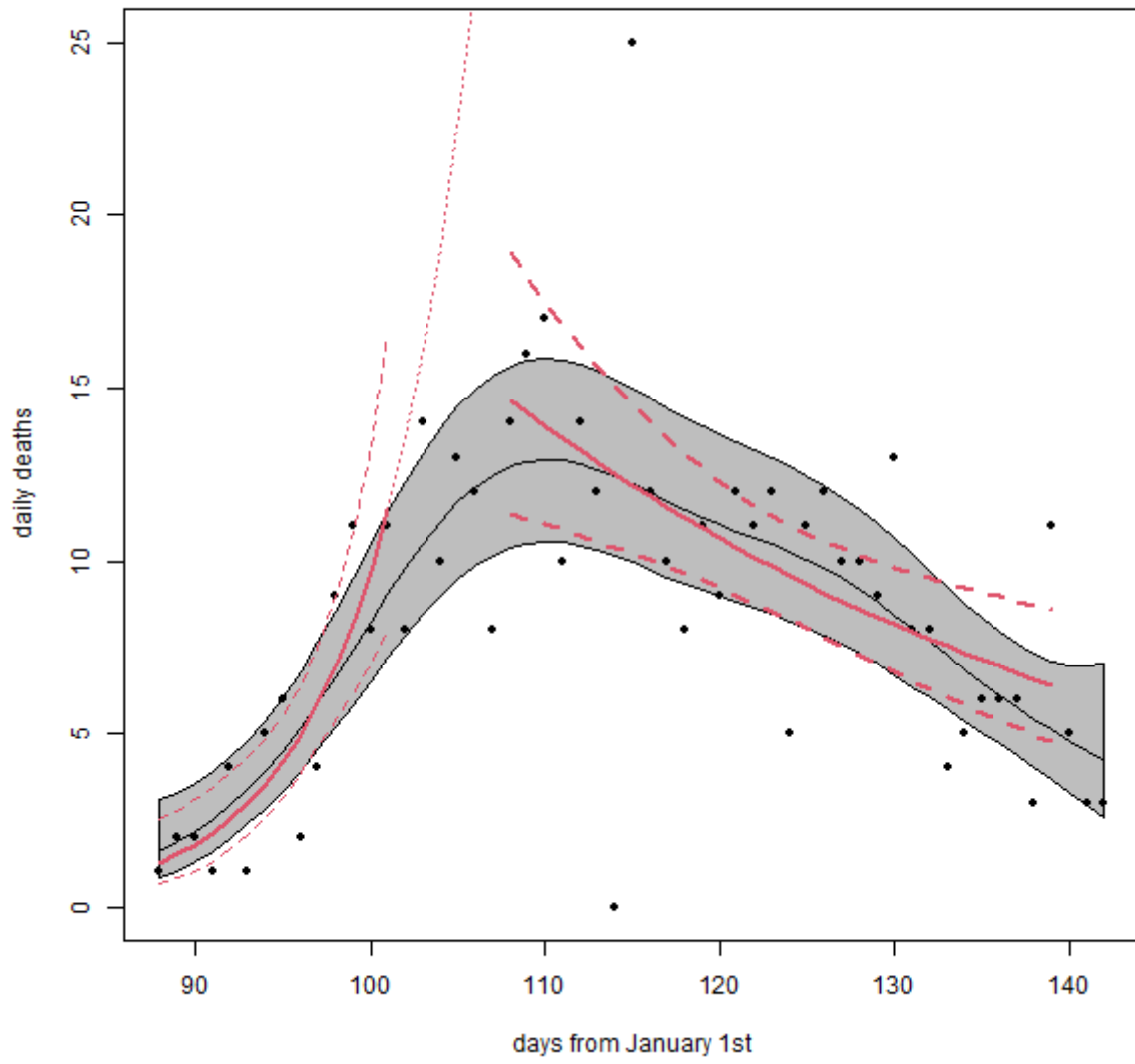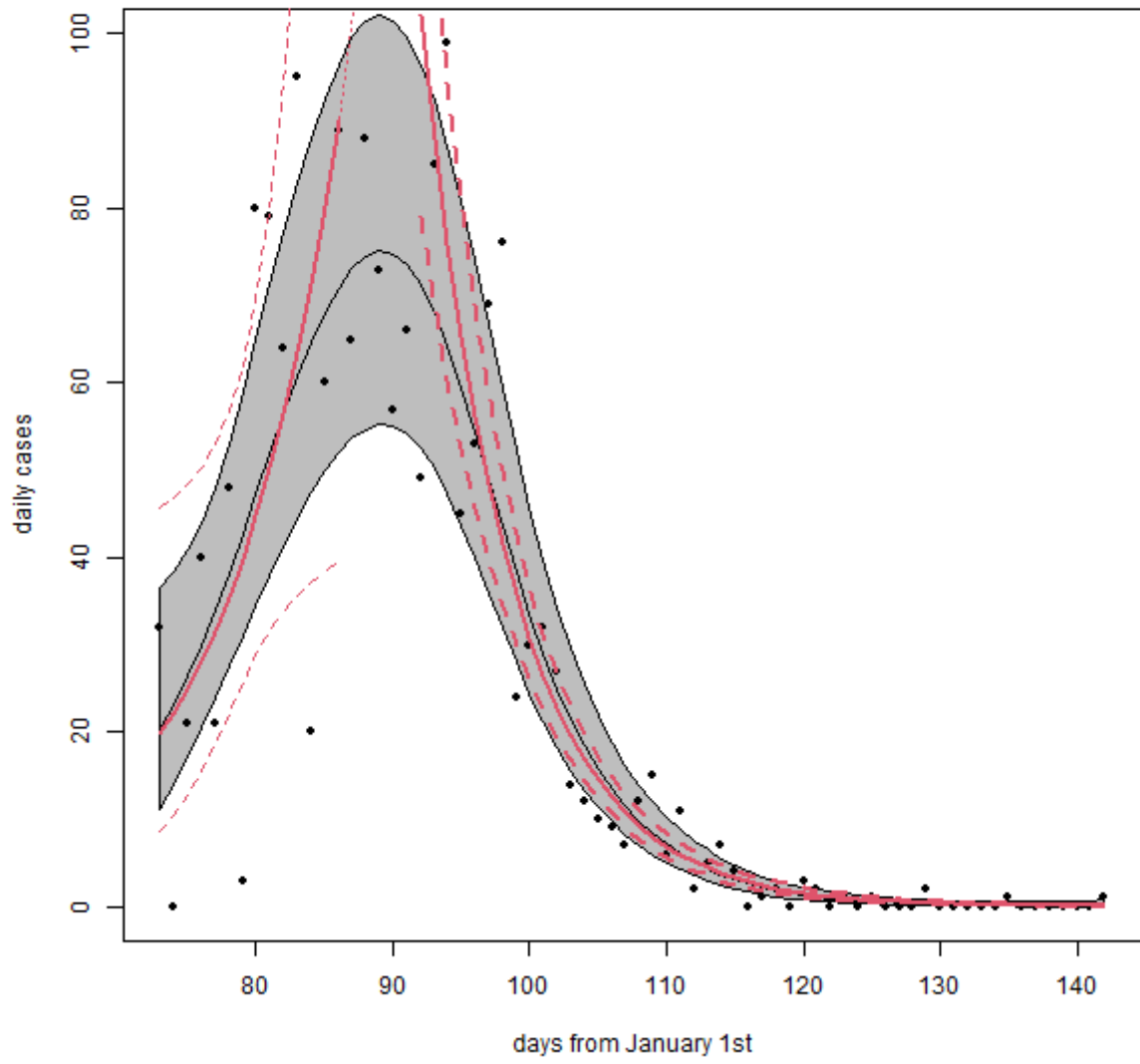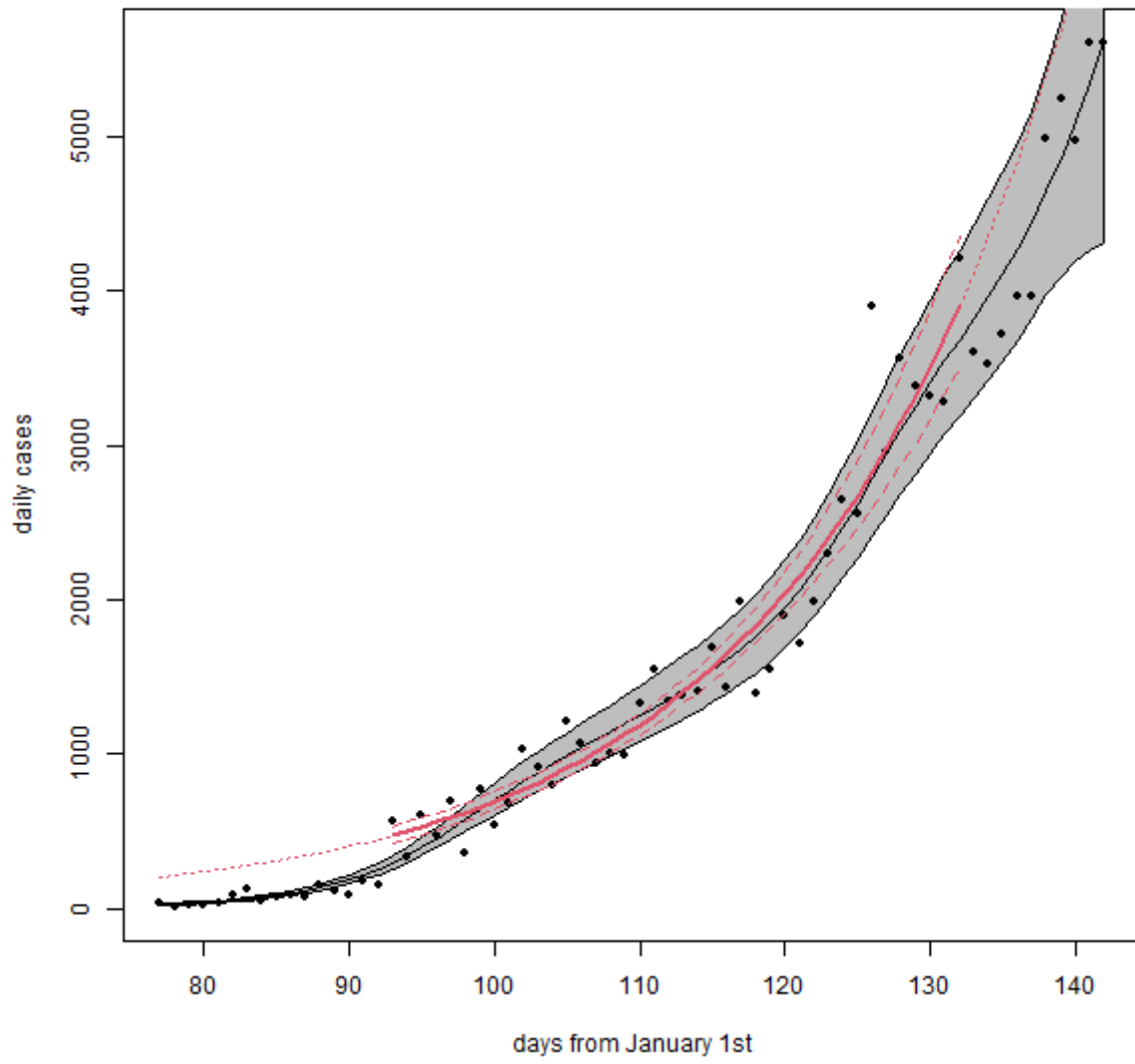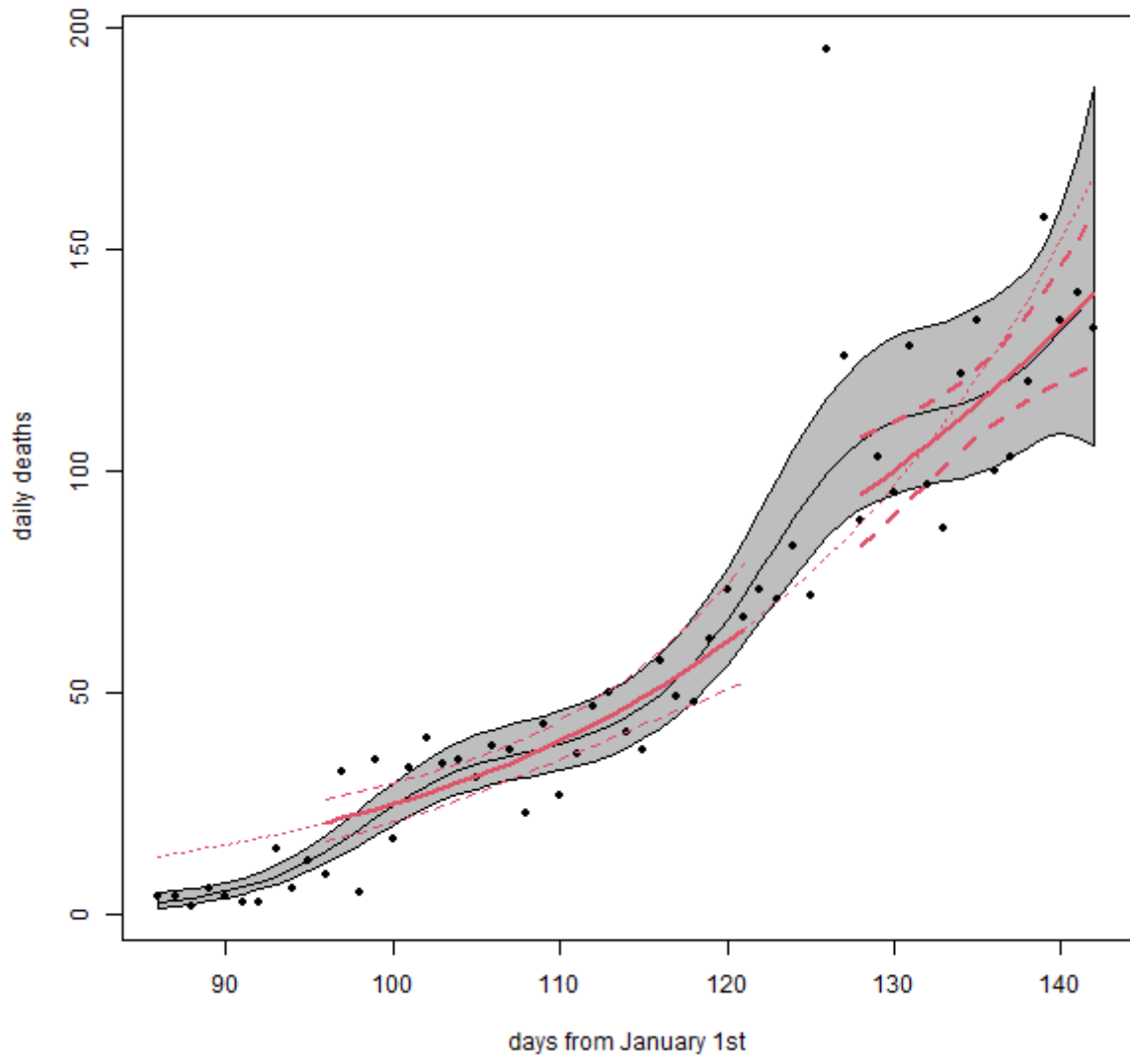
Indonesia

Indonesia

Iran
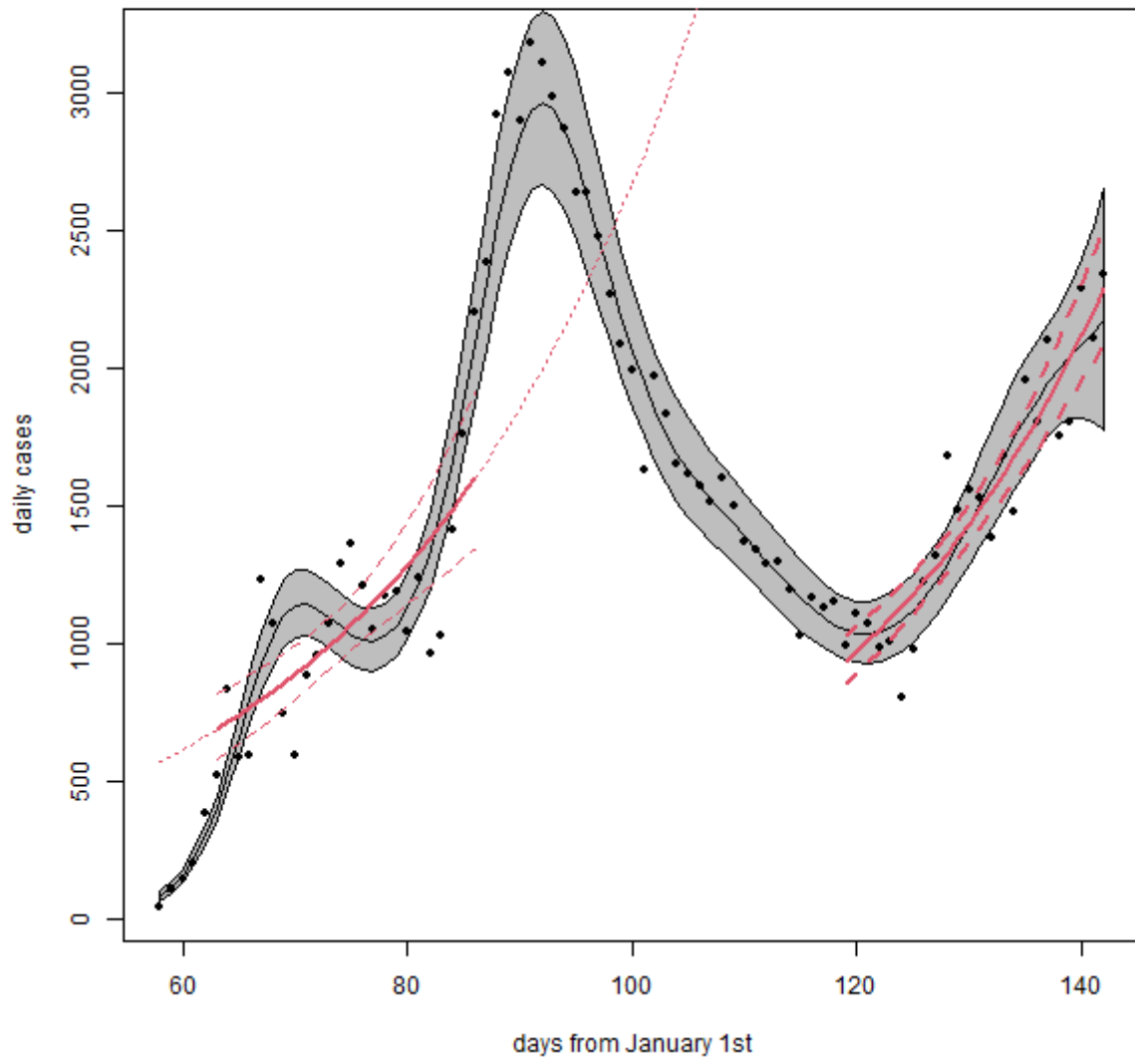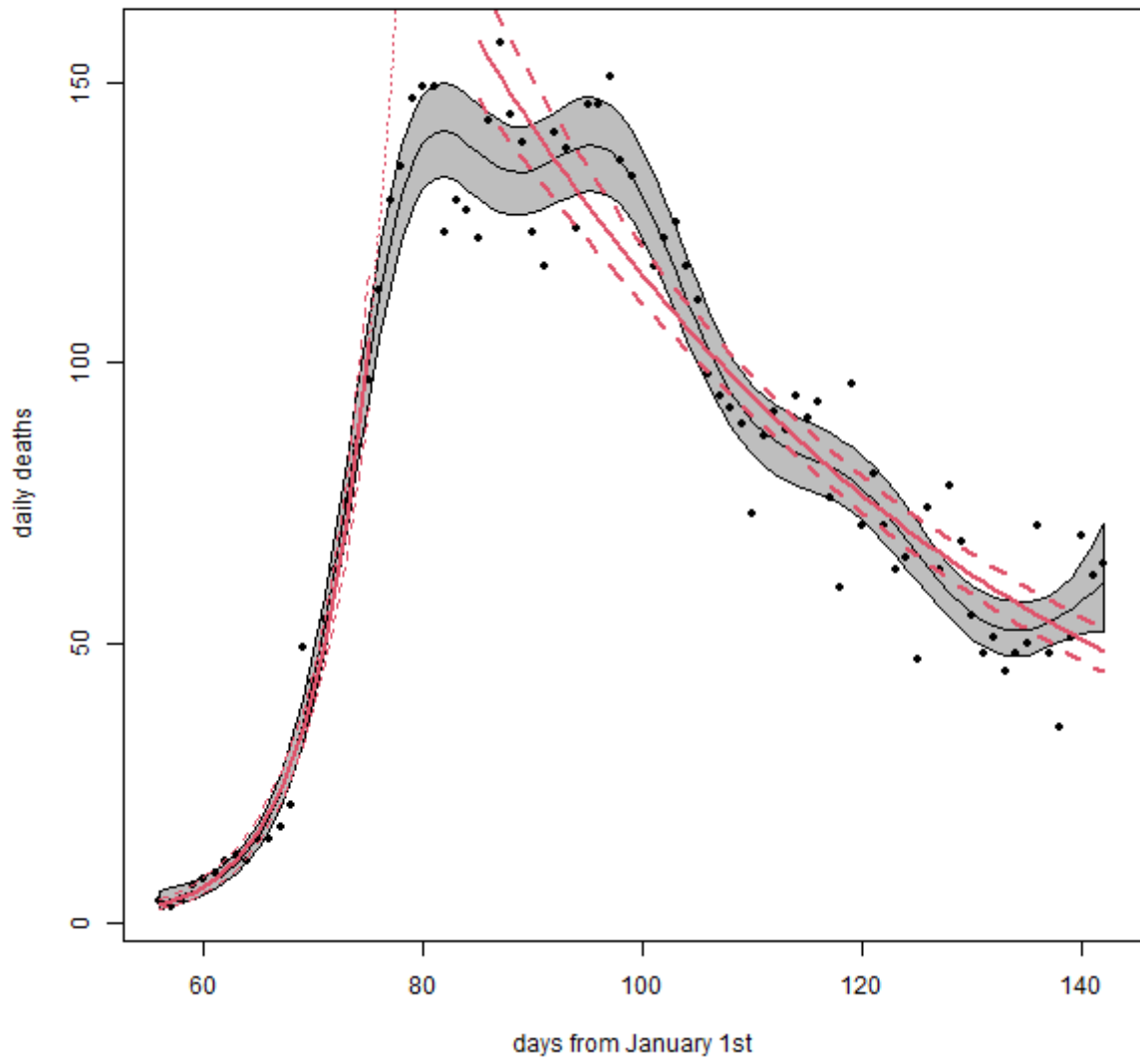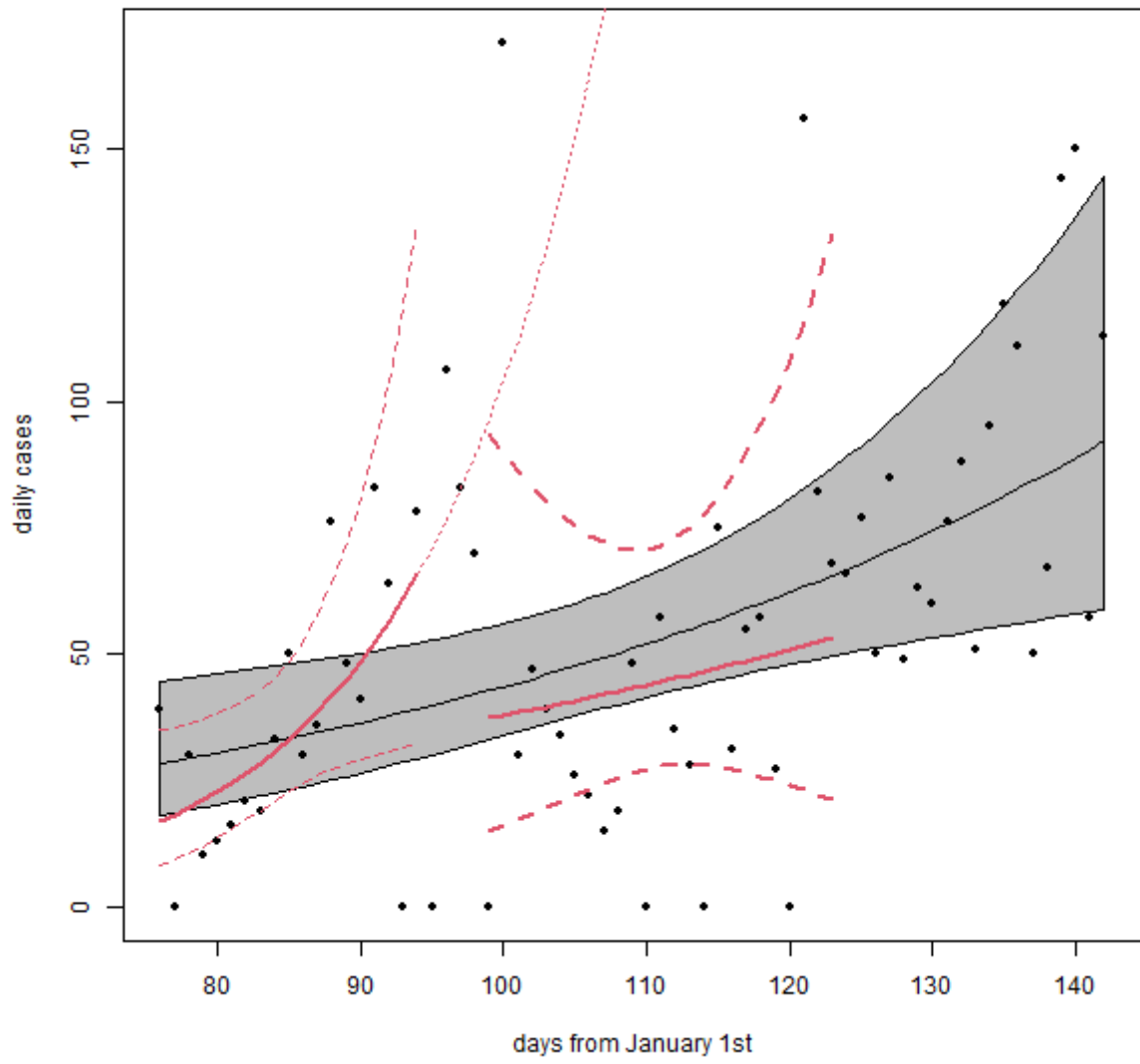
daily cases

days from January 1st

Iran

Iraq

daily cases

days from January 1st

Ireland

daily cases

days from January 1st

Ireland

daily deaths

days from January 1st

Israel

Italy

Italy

daily deaths

days from January 1st

Japan

daily cases

days from January 1st

Japan

daily deaths

days from January 1st

Kazakhstan

daily cases

days from January 1st

Kenya

daily cases

days from January 1st

Kuwait

daily cases

days from January 1st

Kuwait

daily deaths
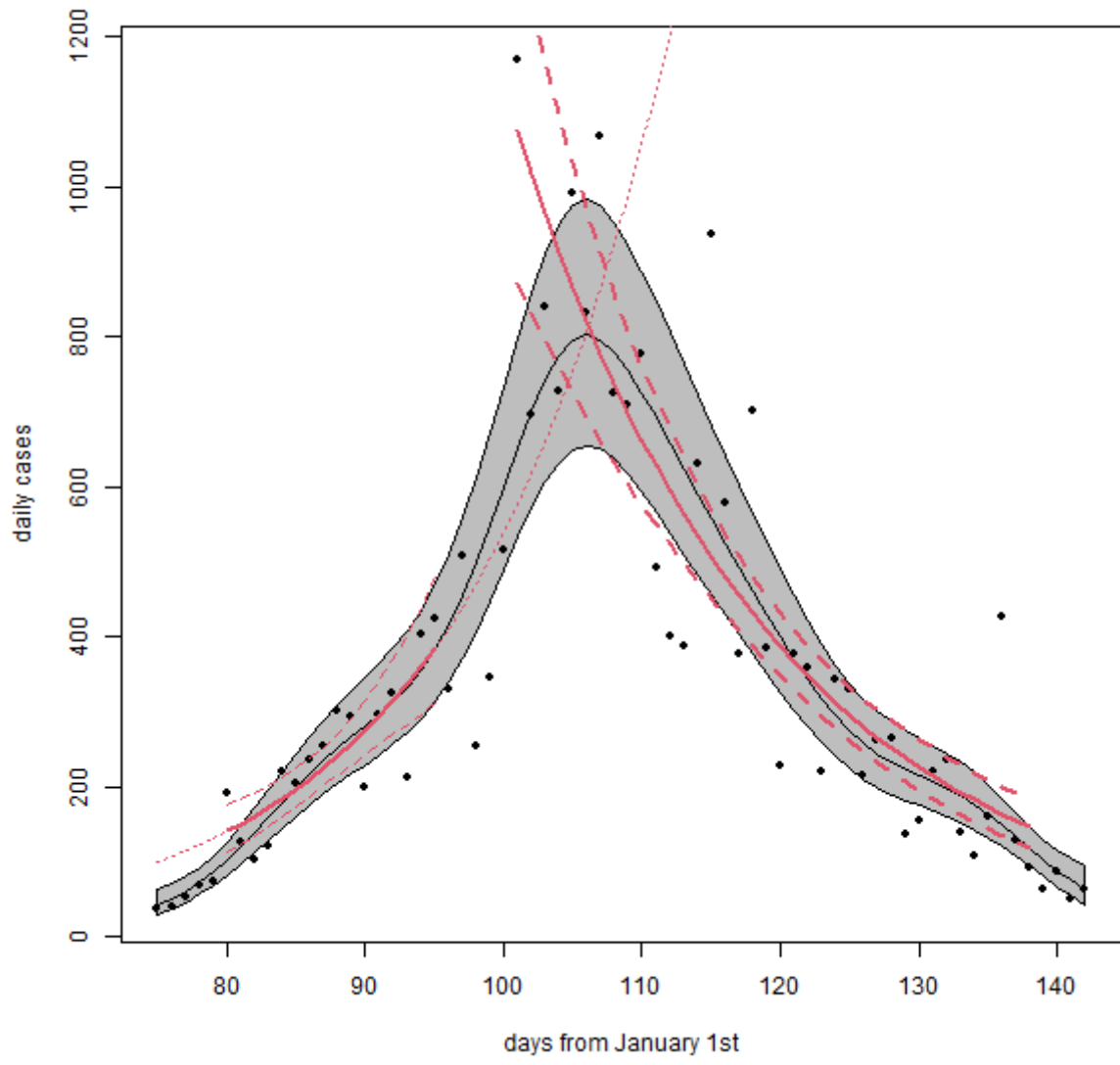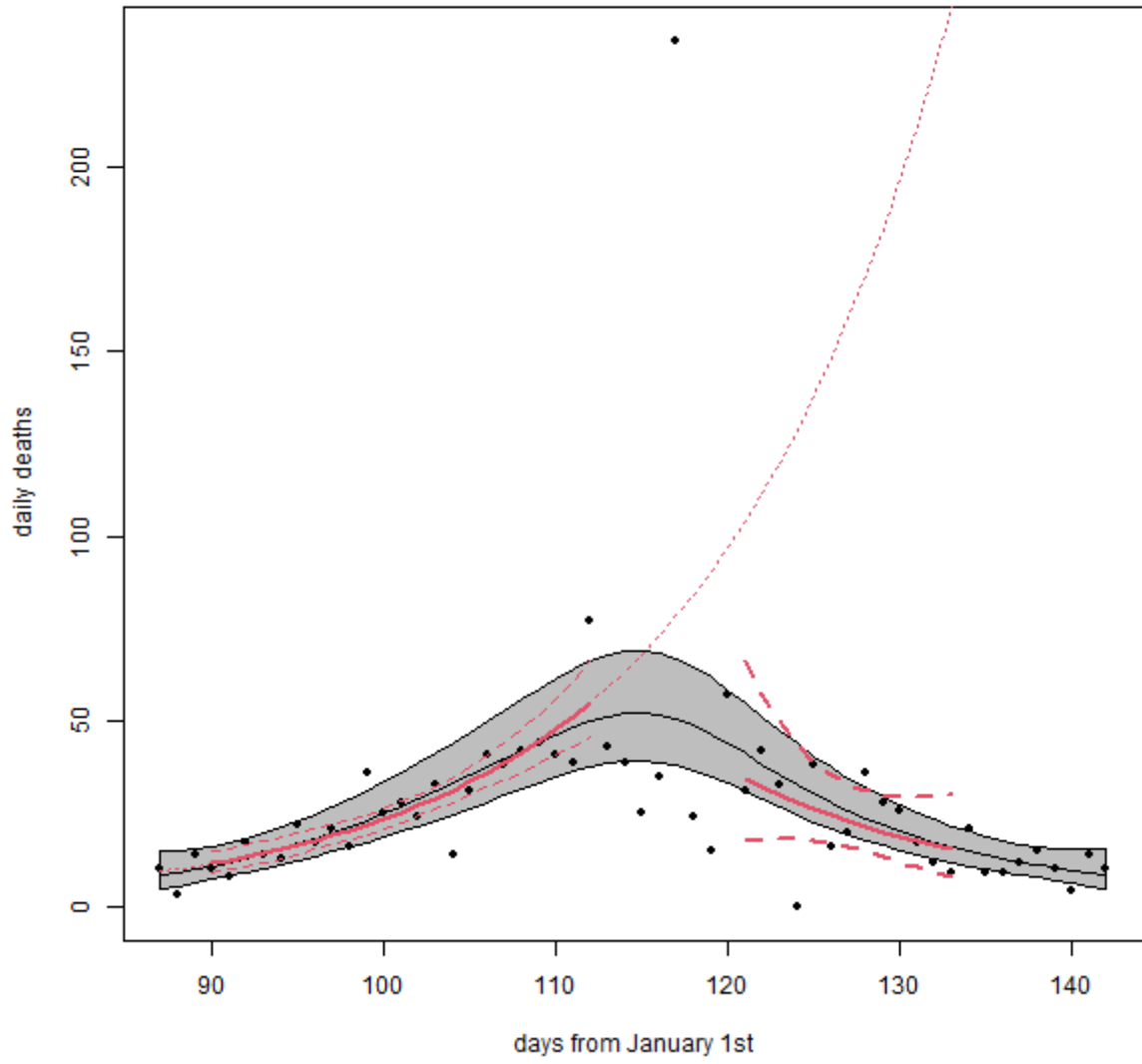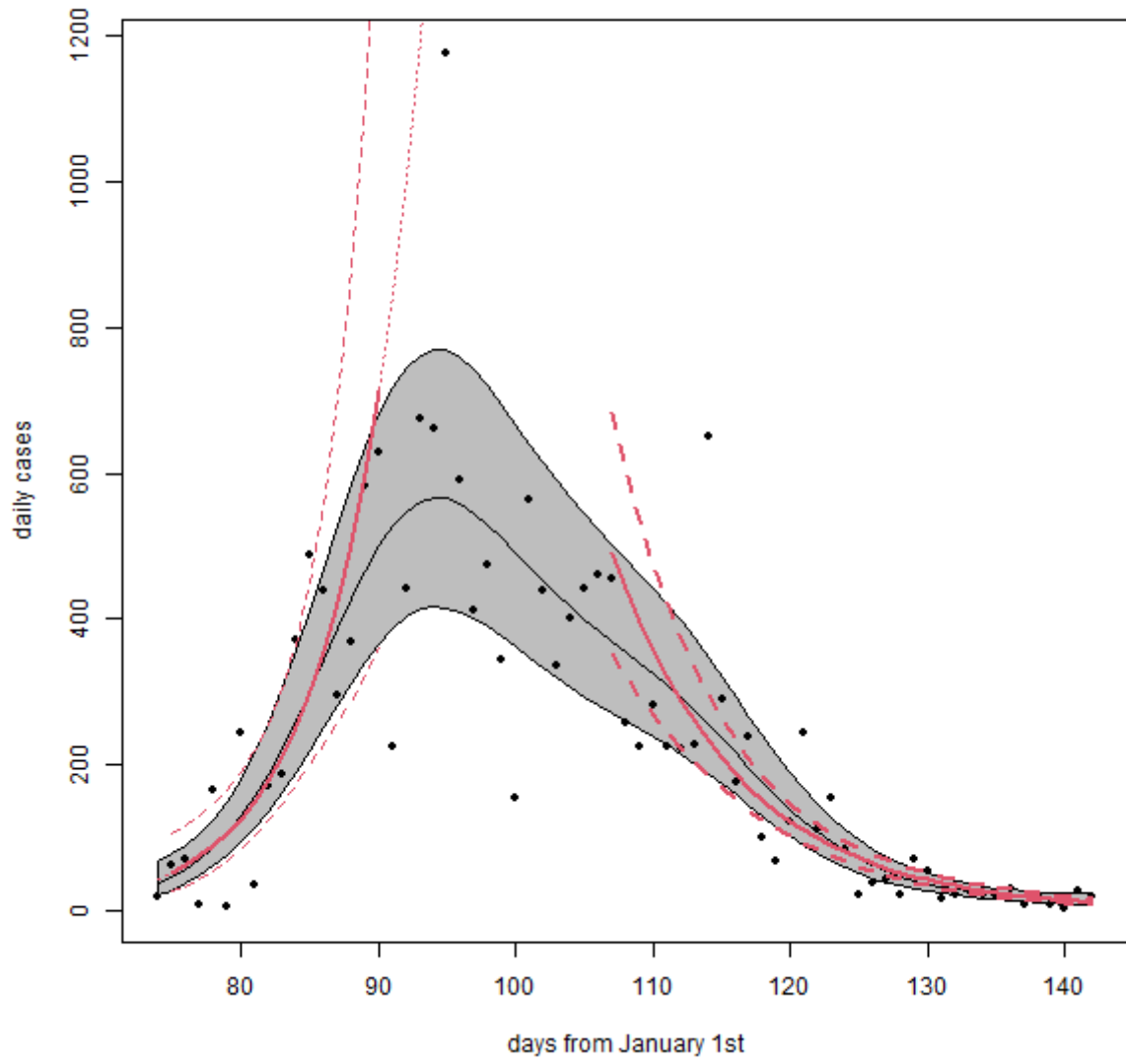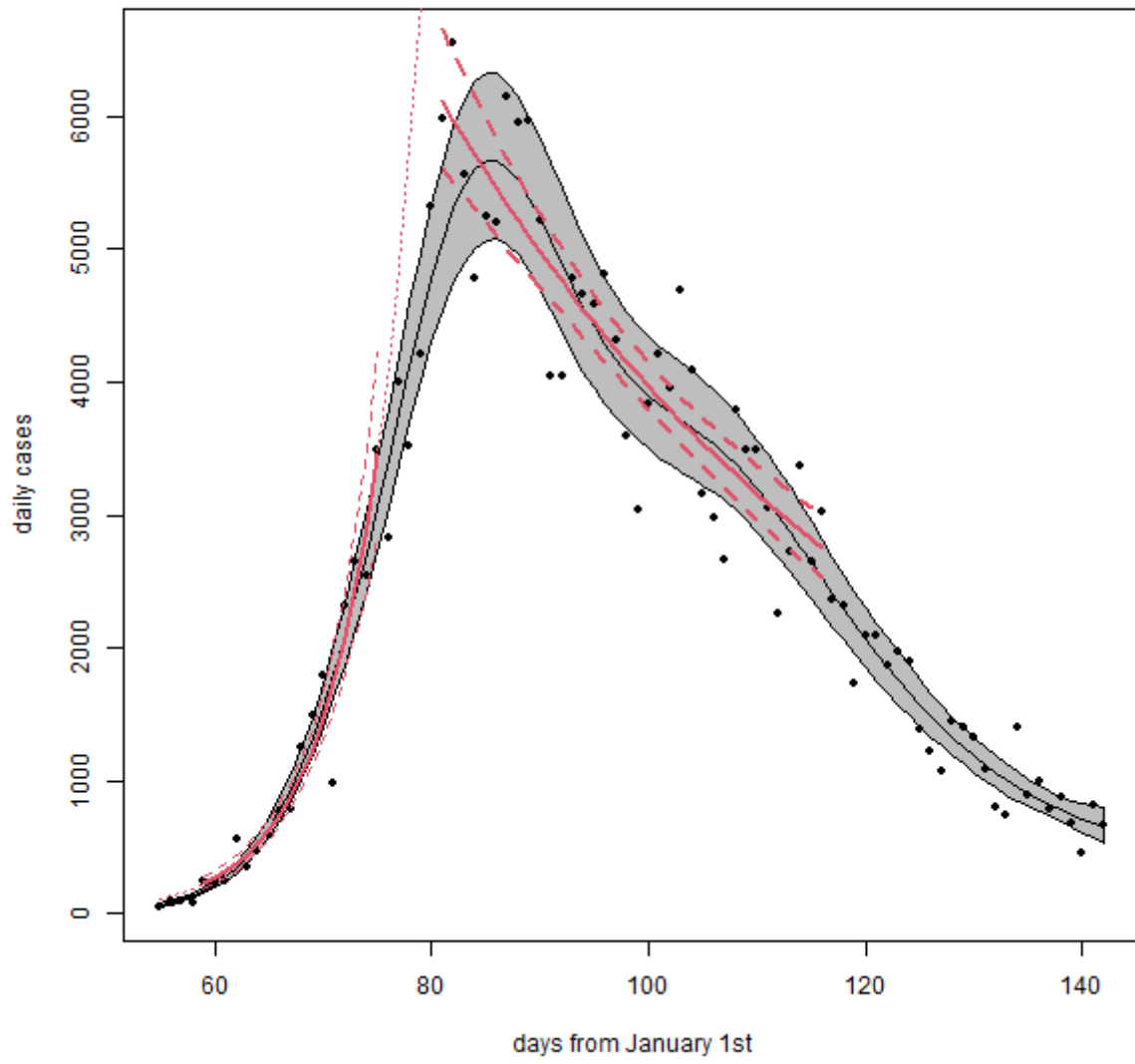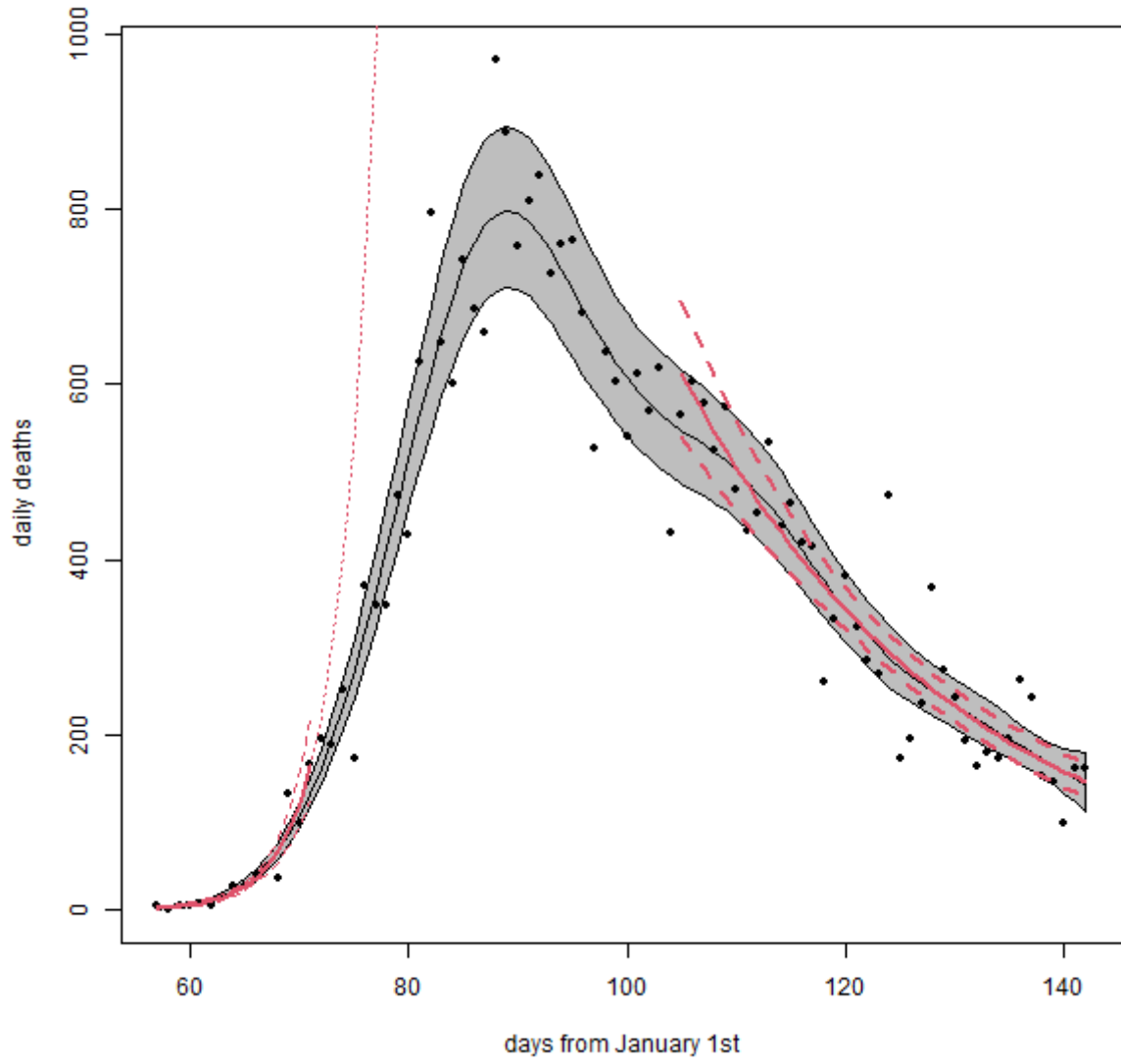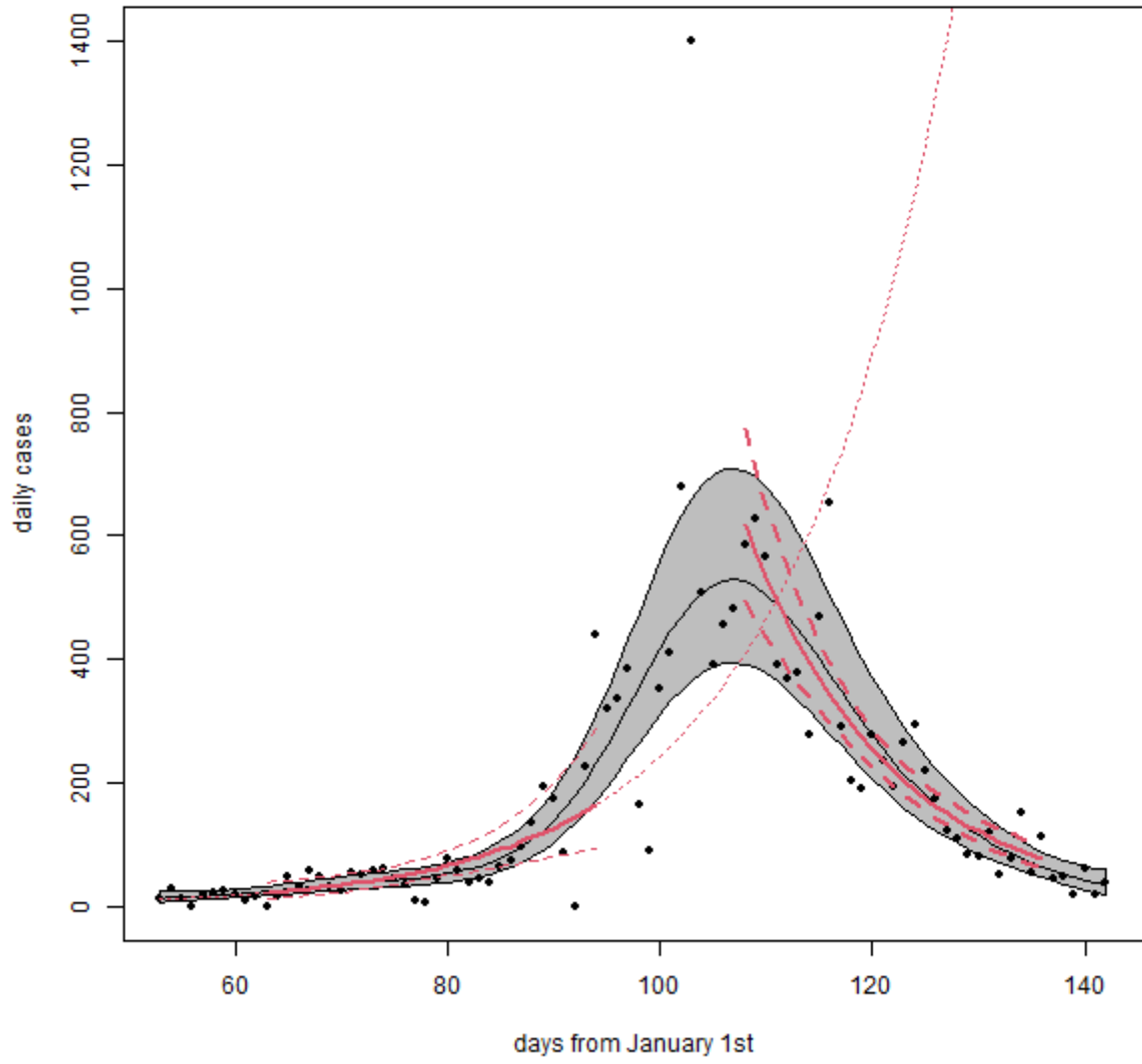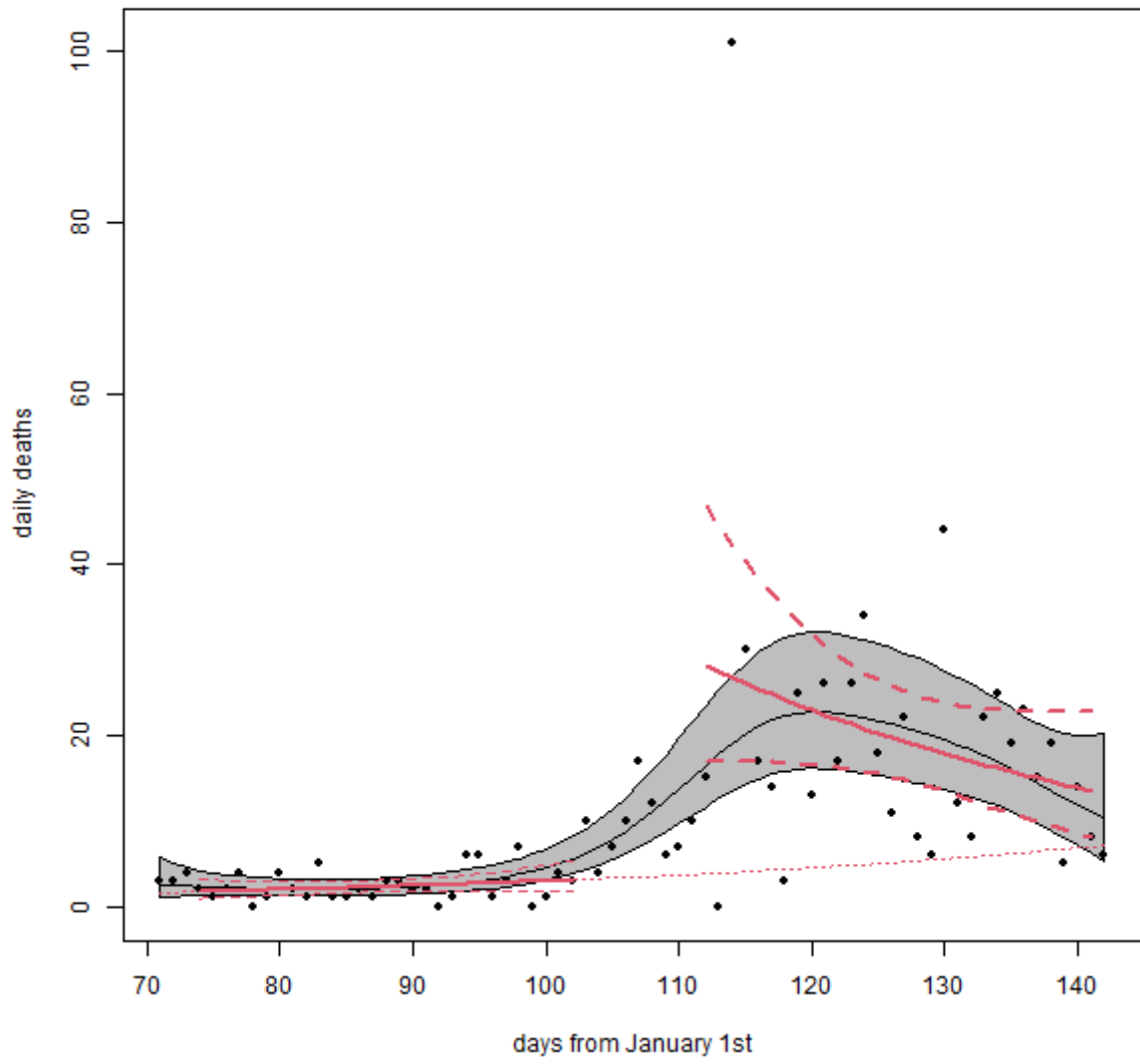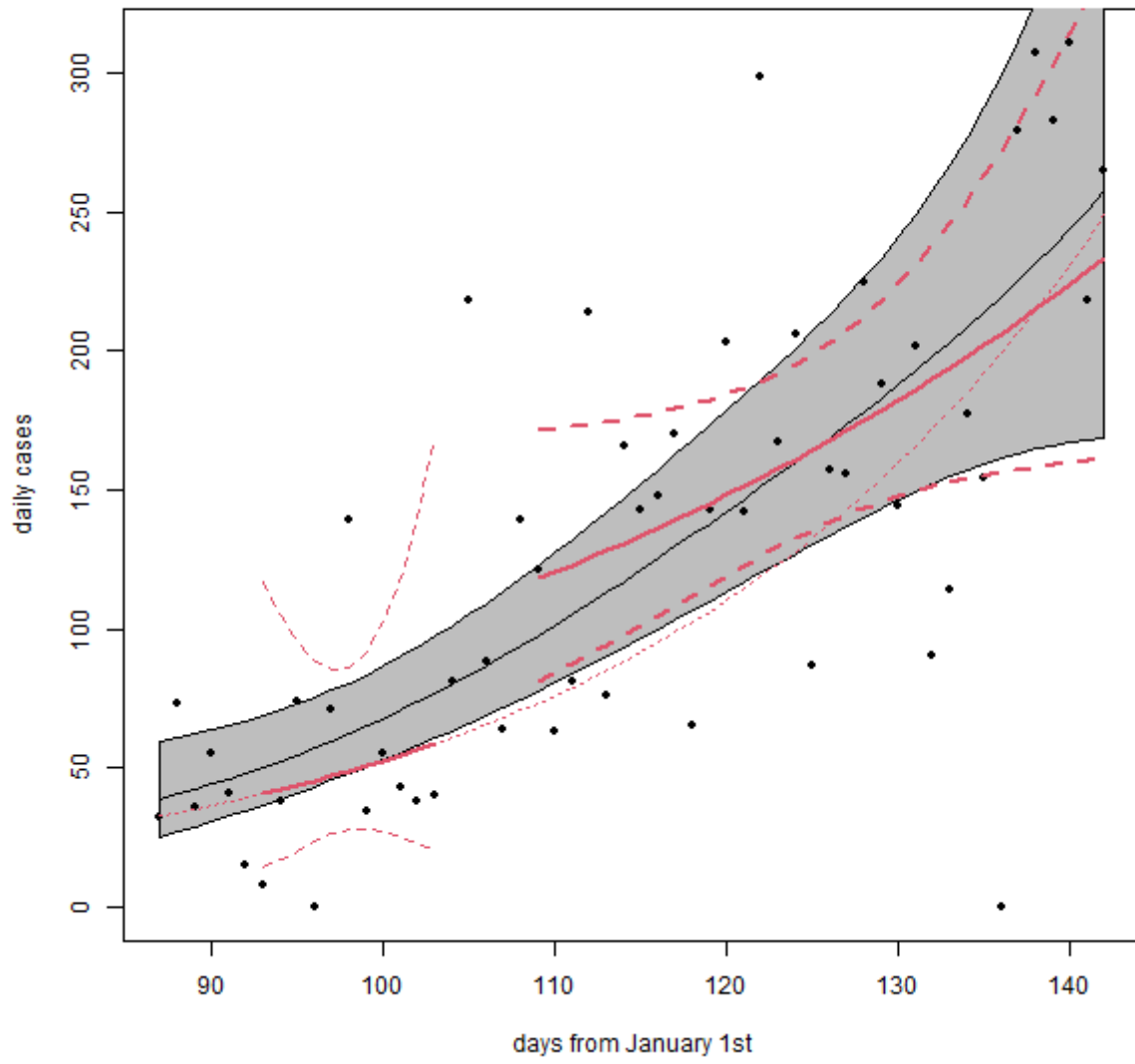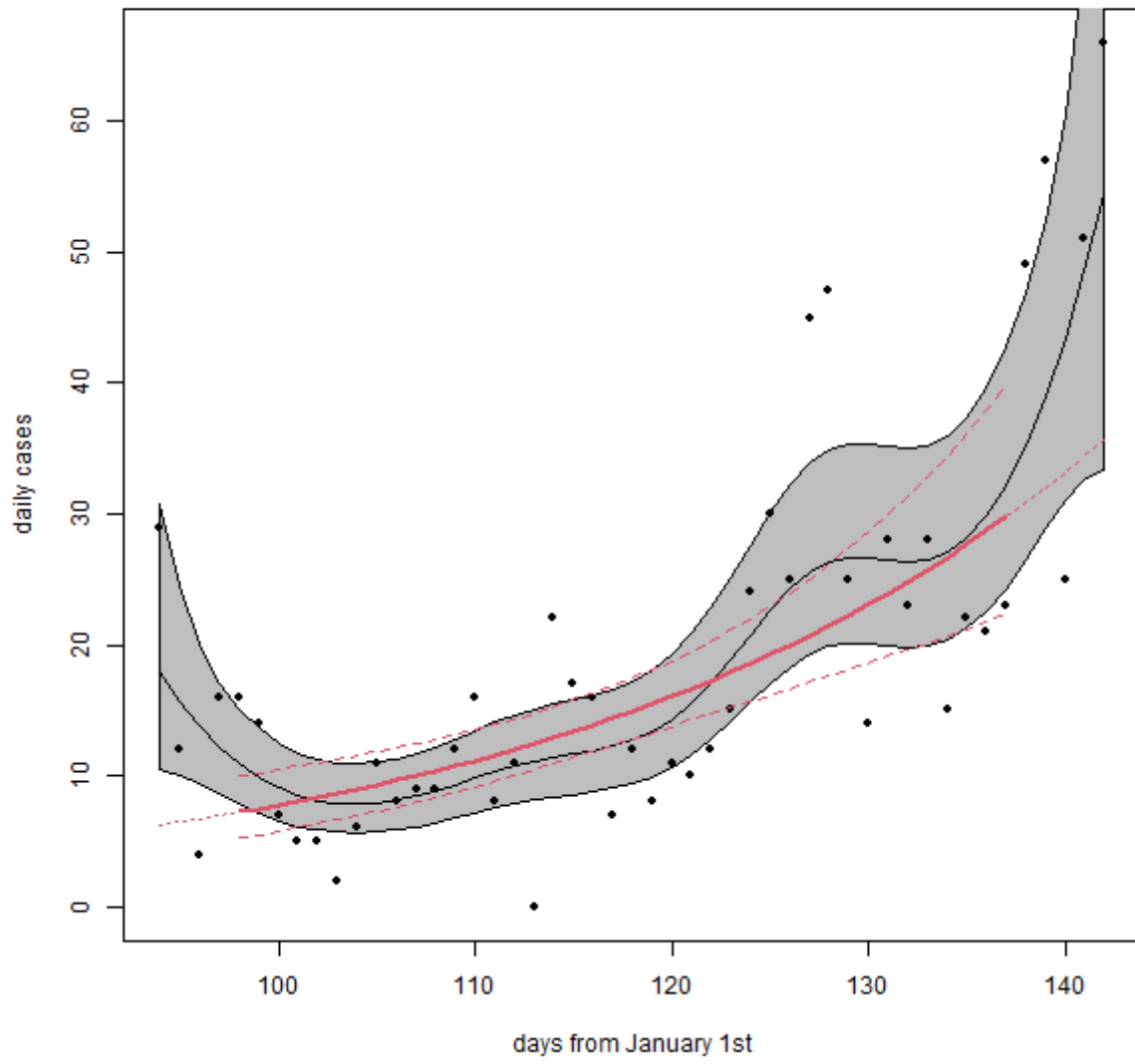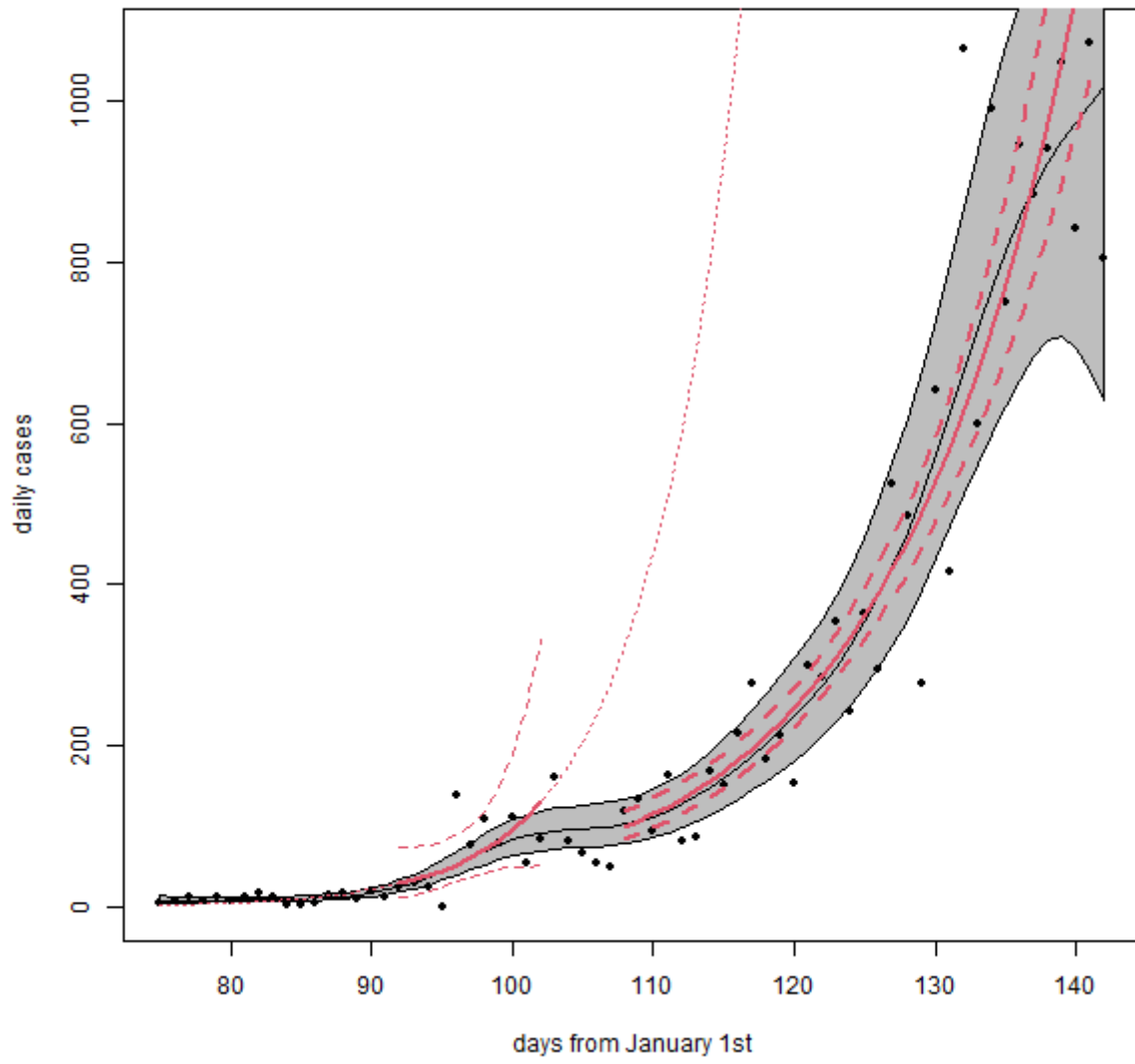
days from January 1st
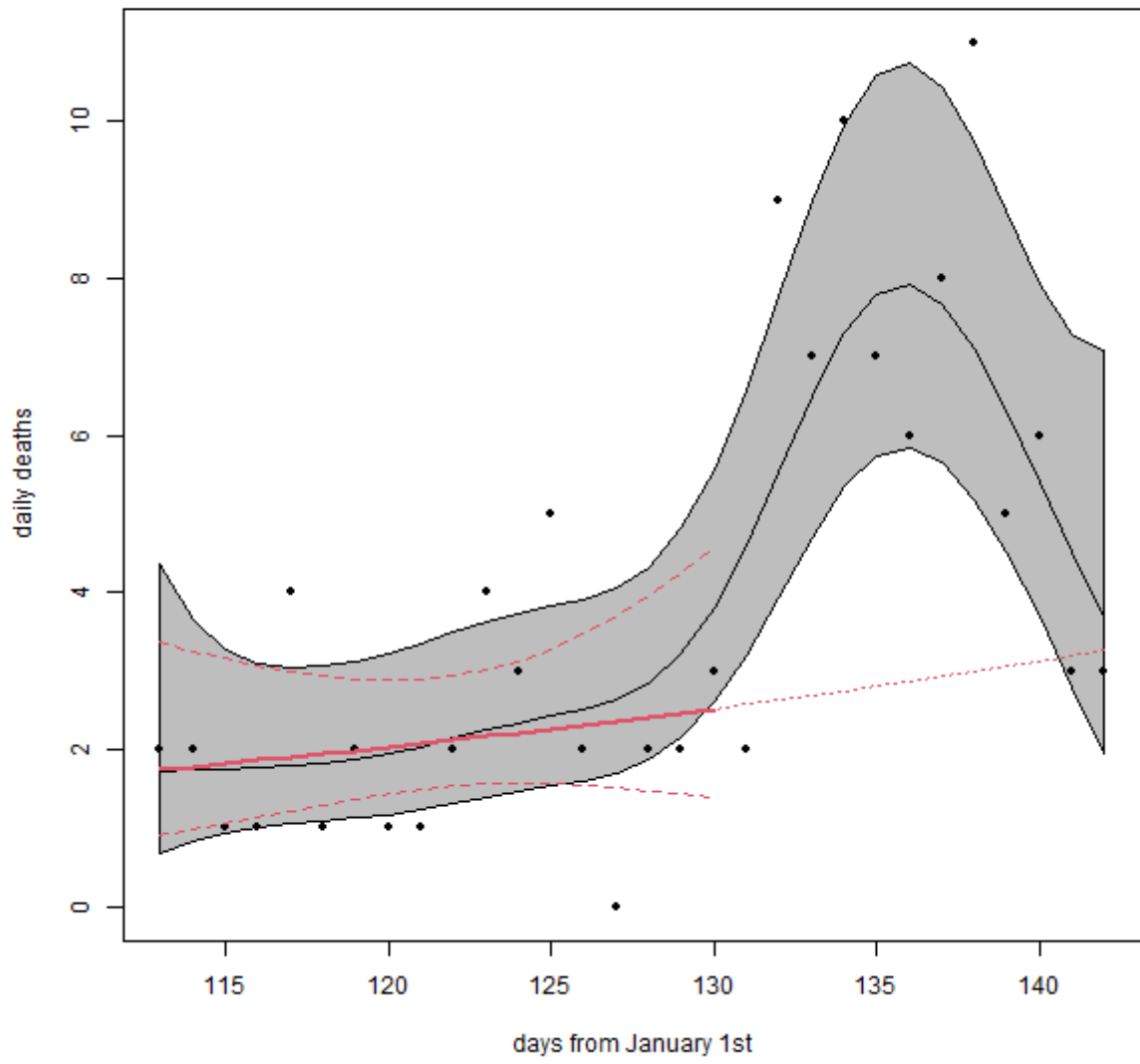
Kyrgyzstan
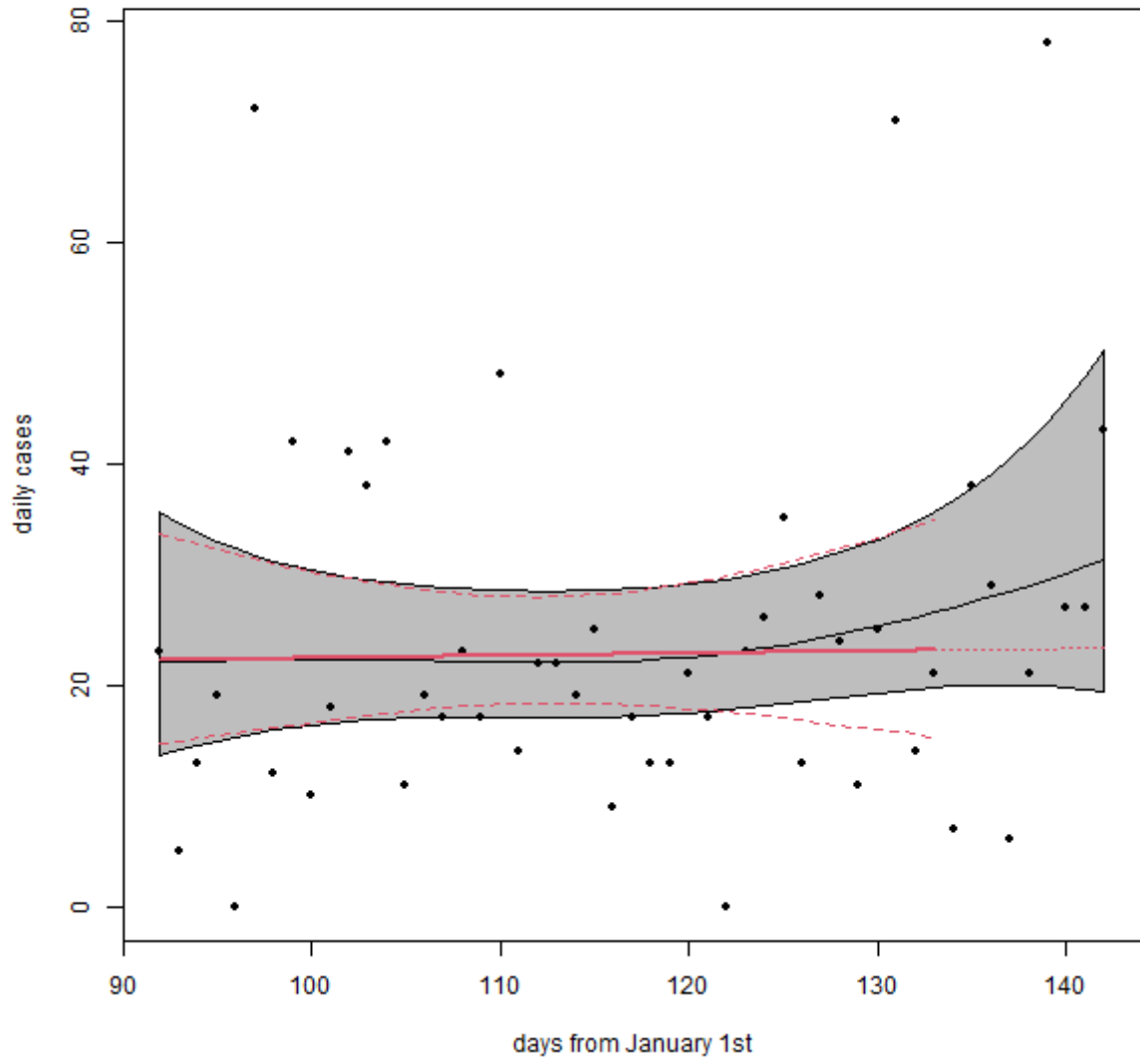
Malaysia

Mexico

Mexico

Moldova

daily cases

days from January 1st
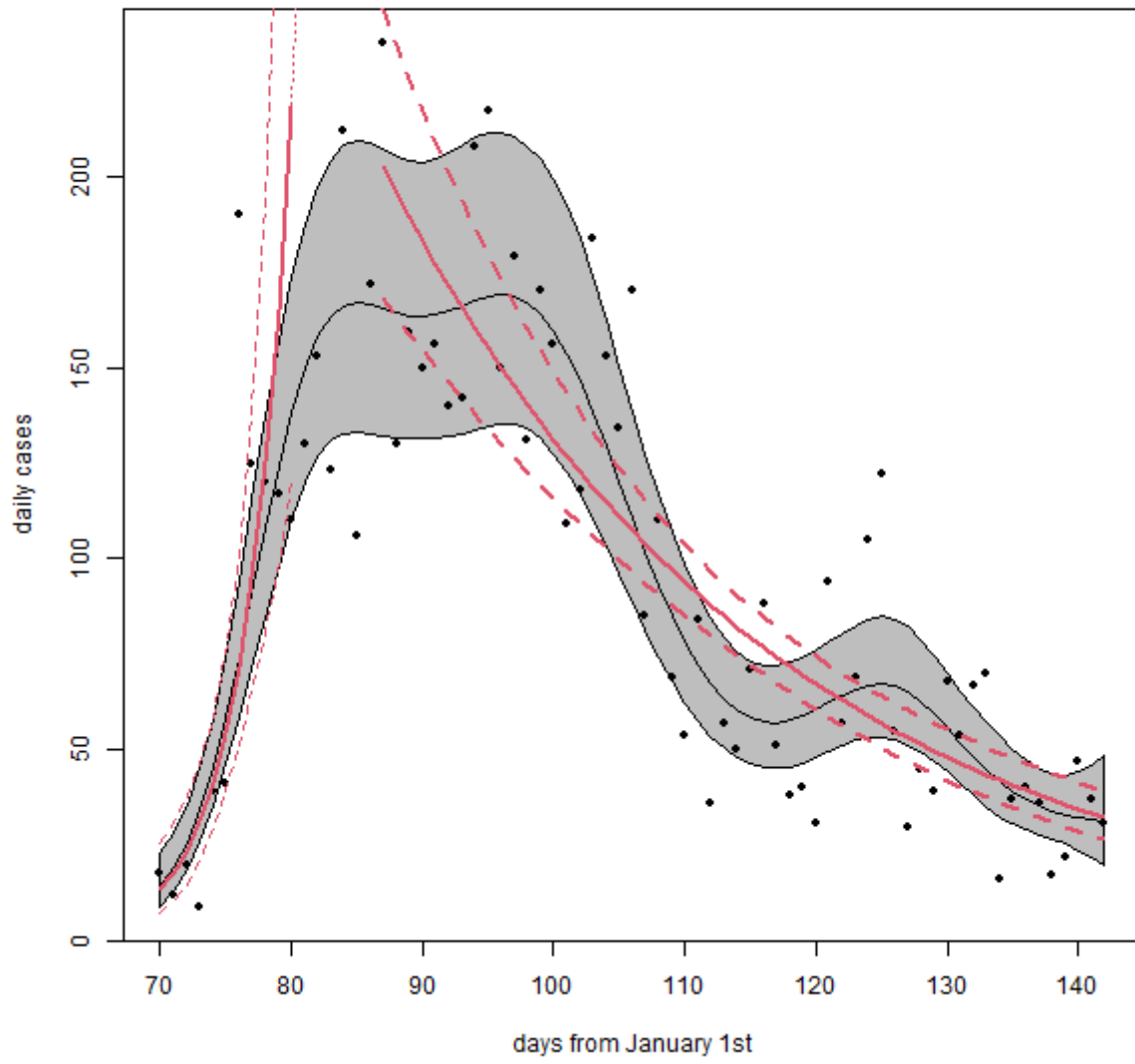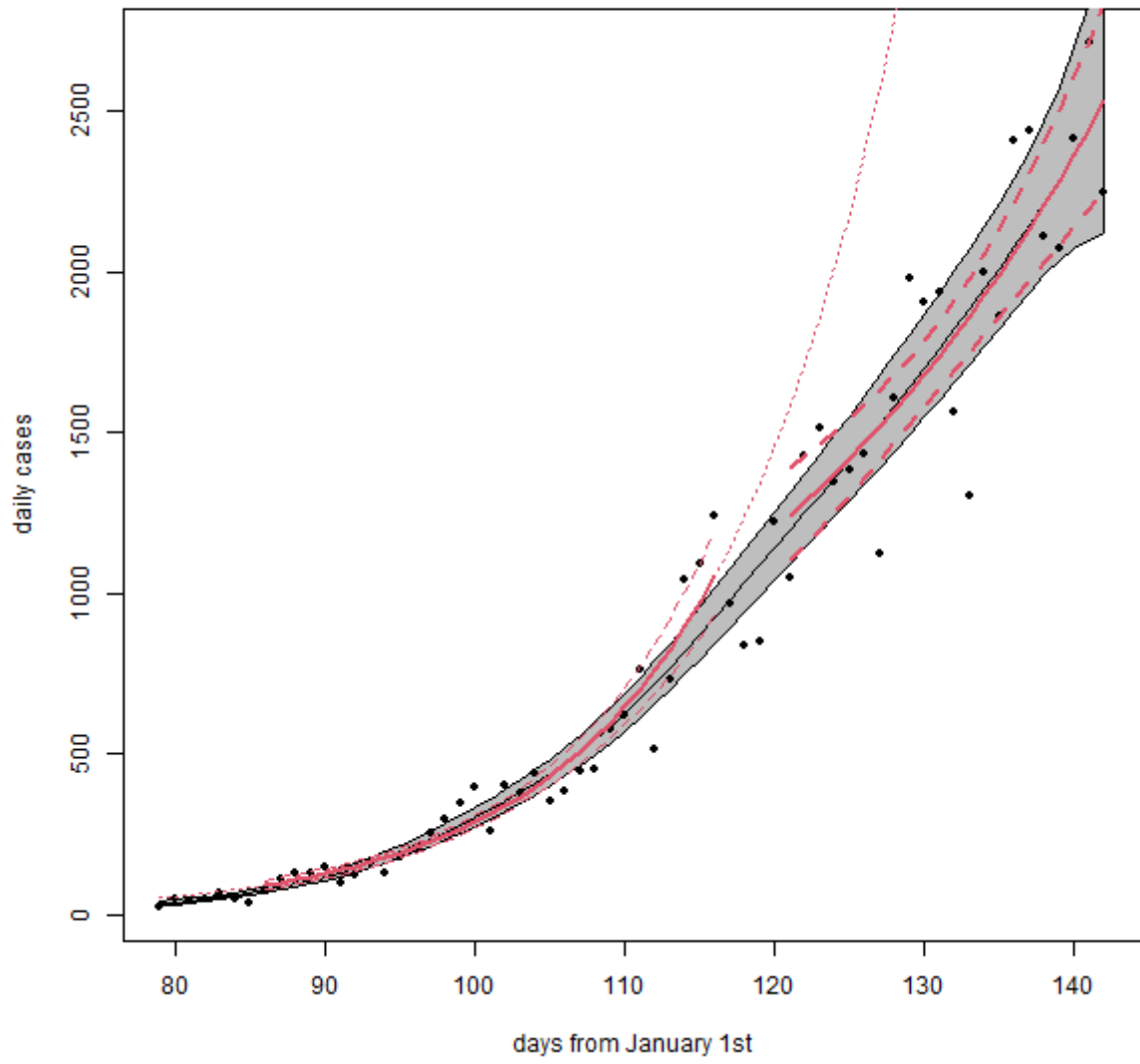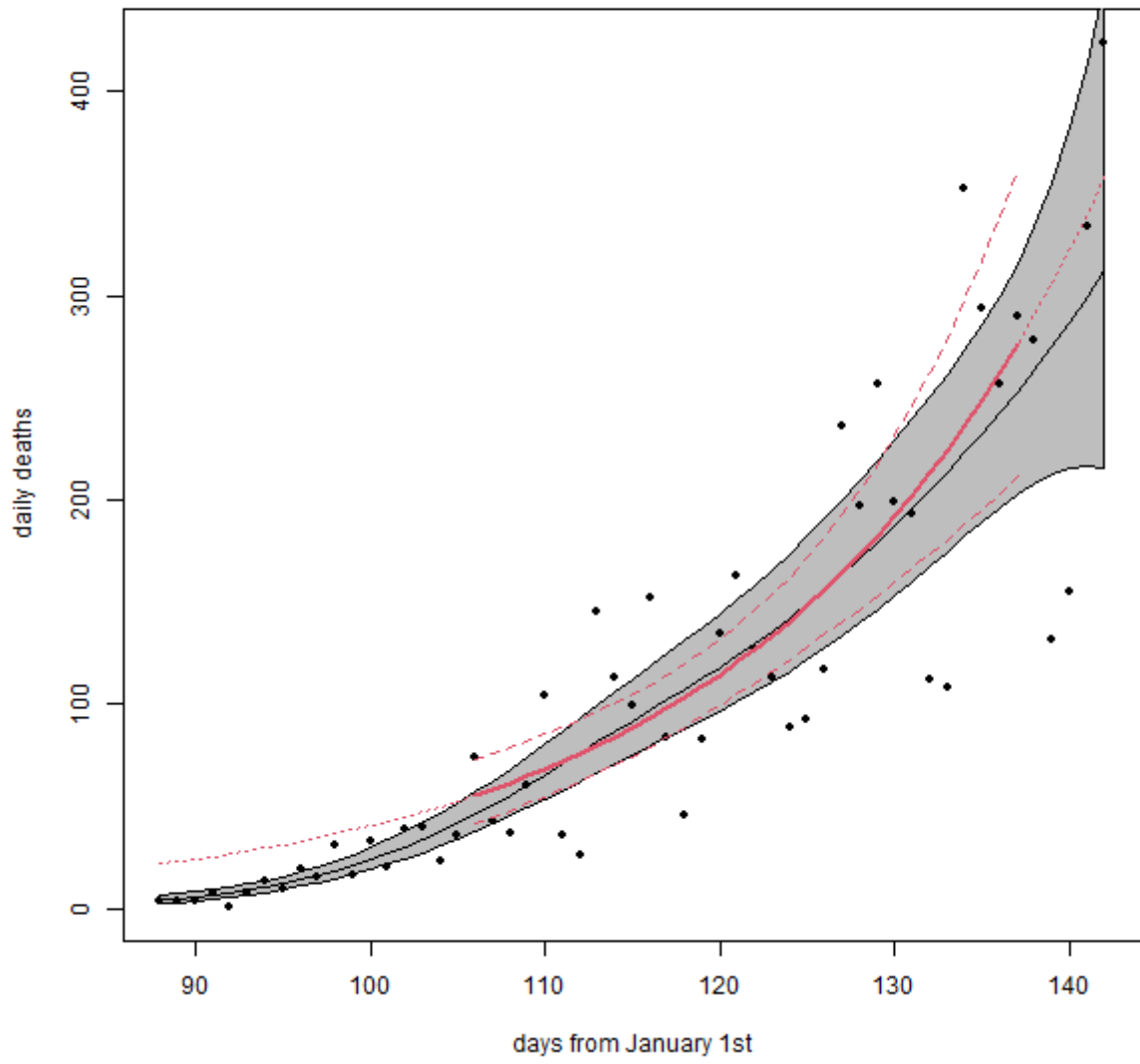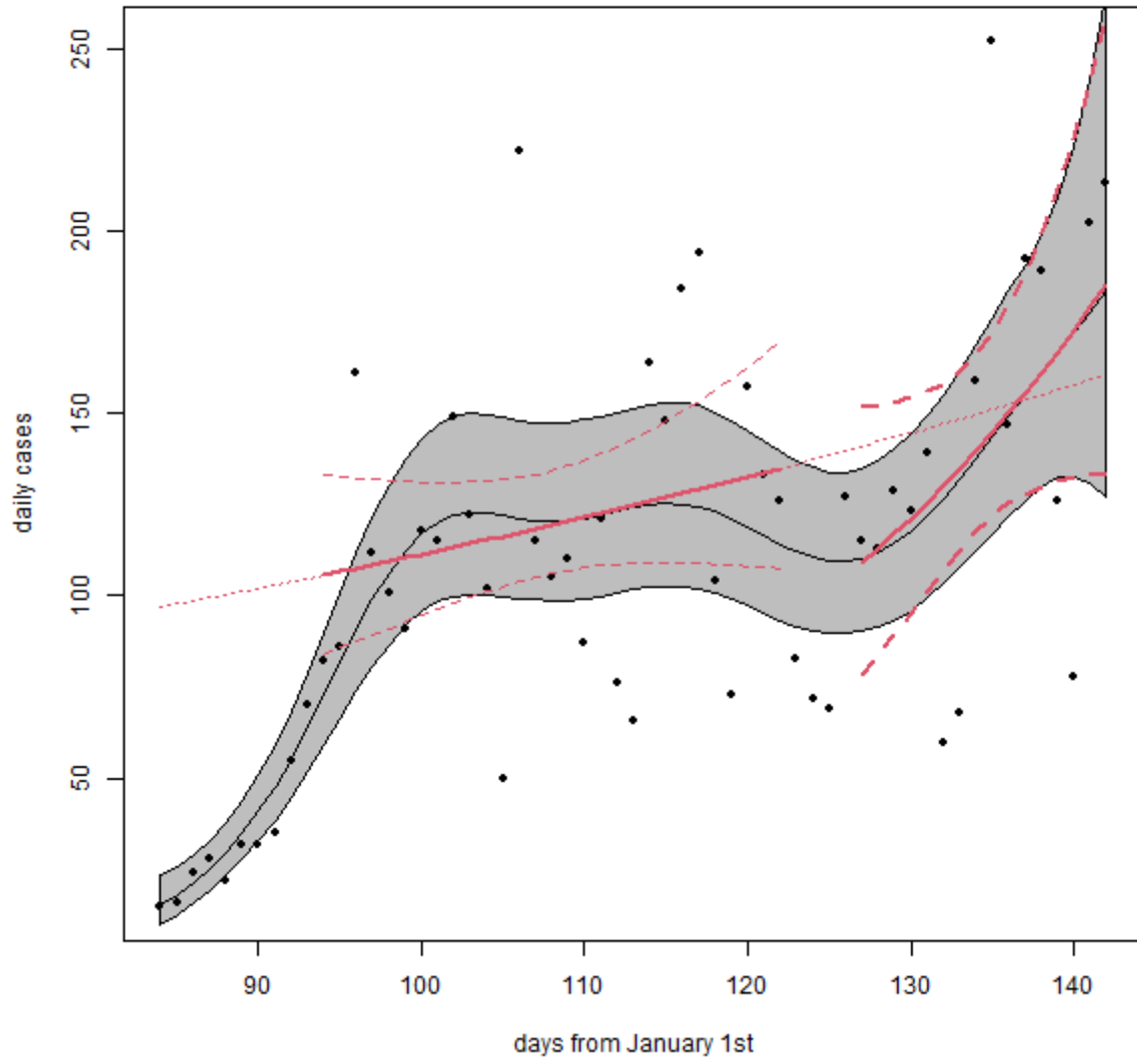
Moldova

Morocco

N_Macedonia
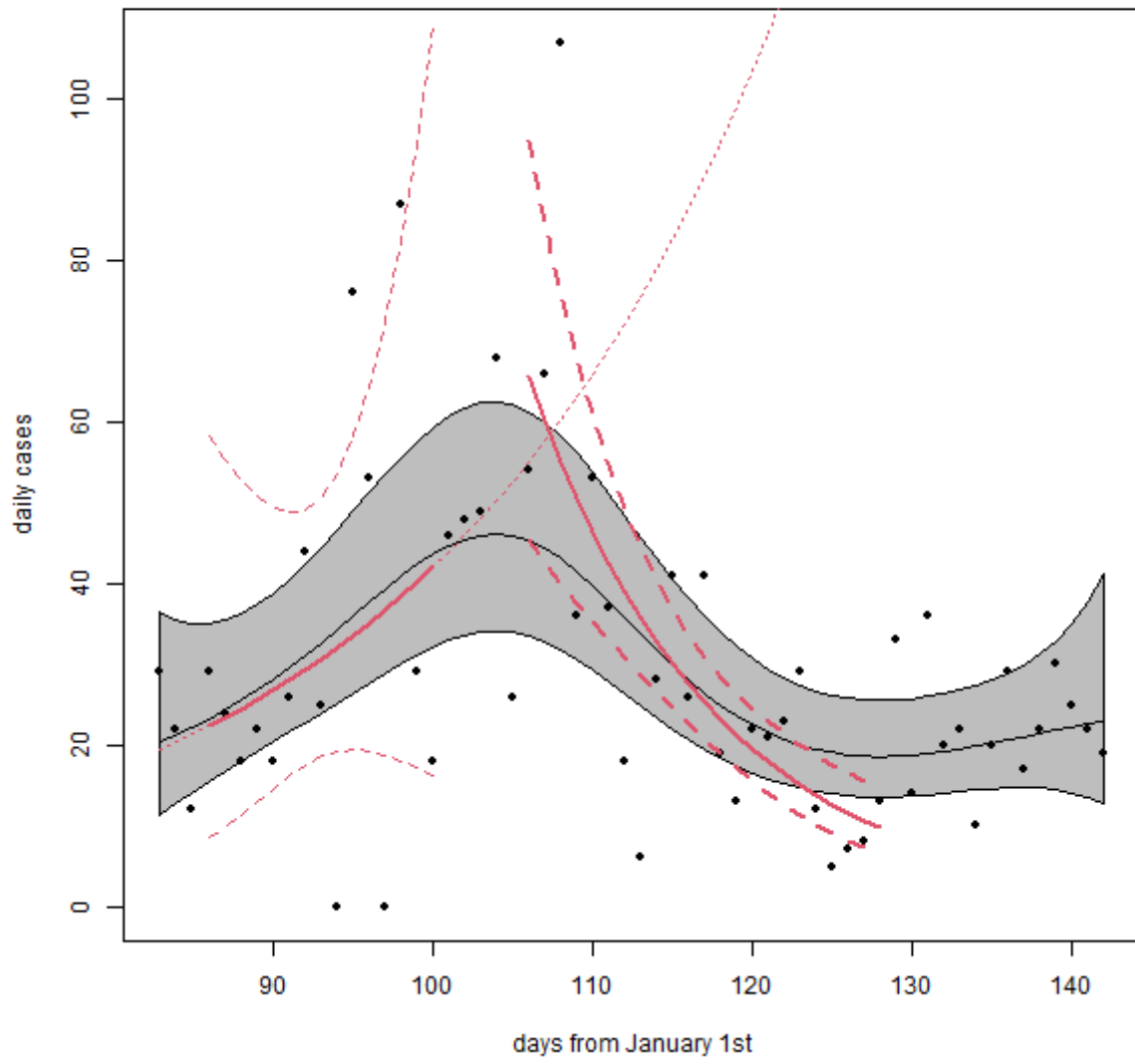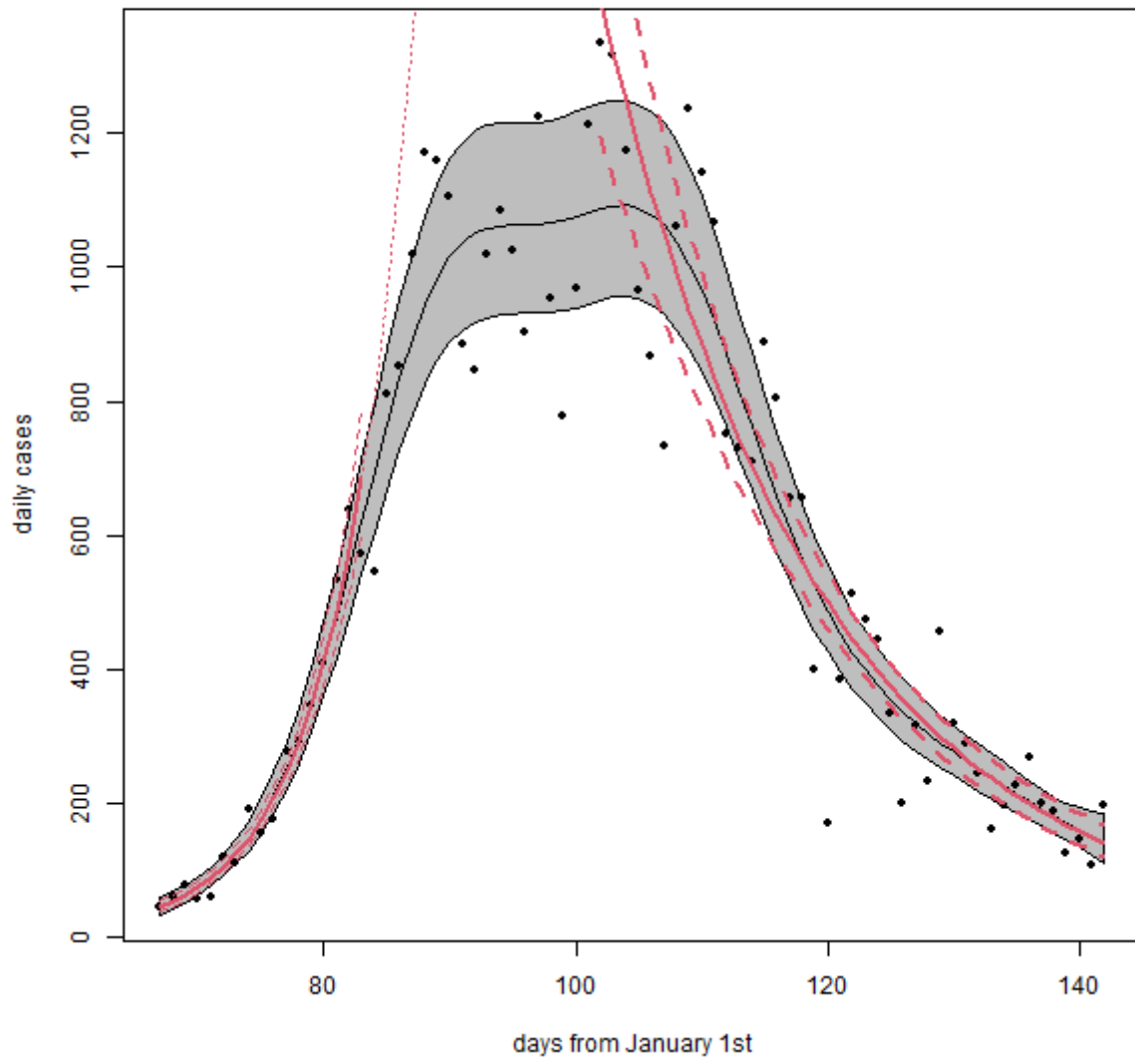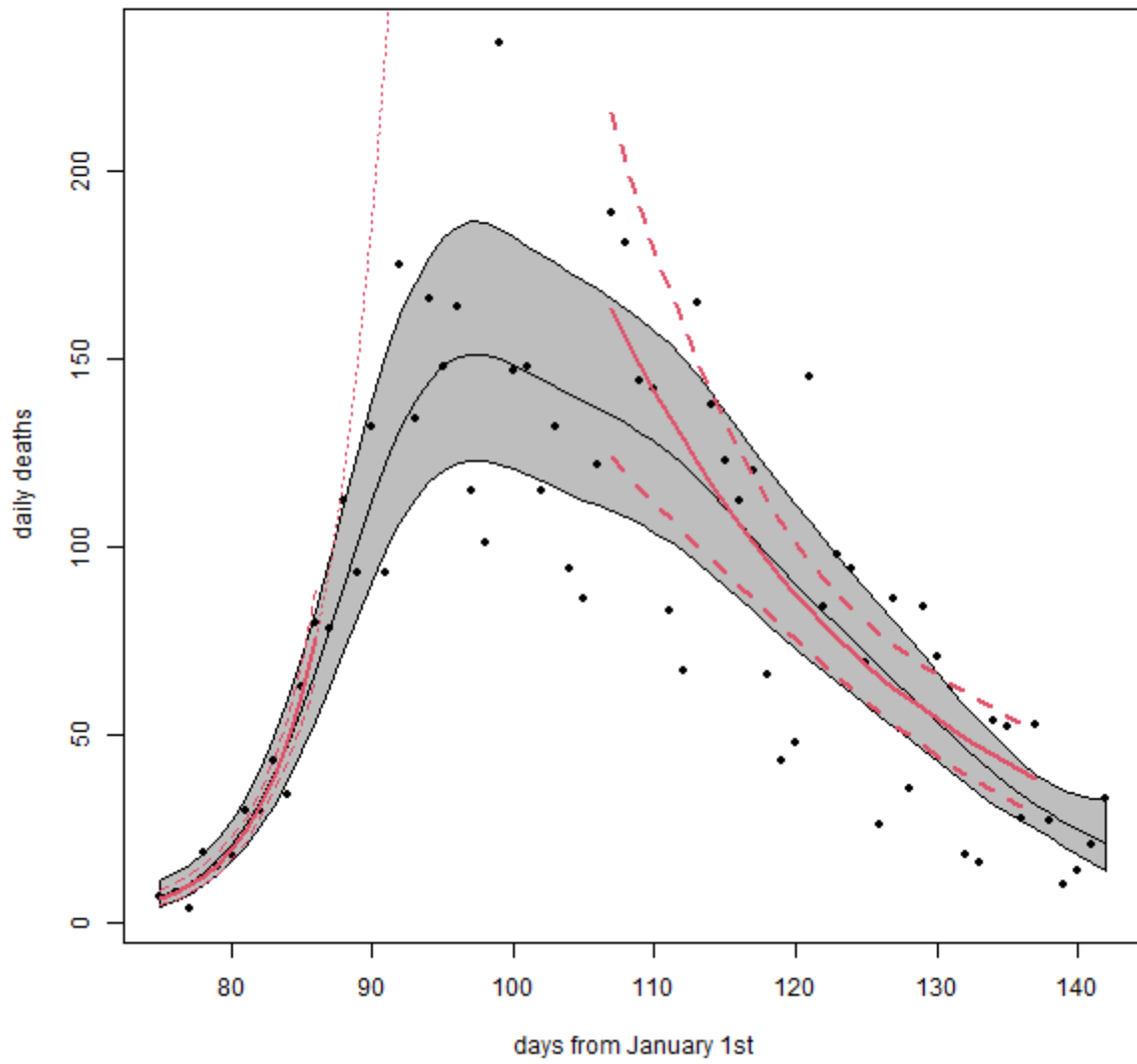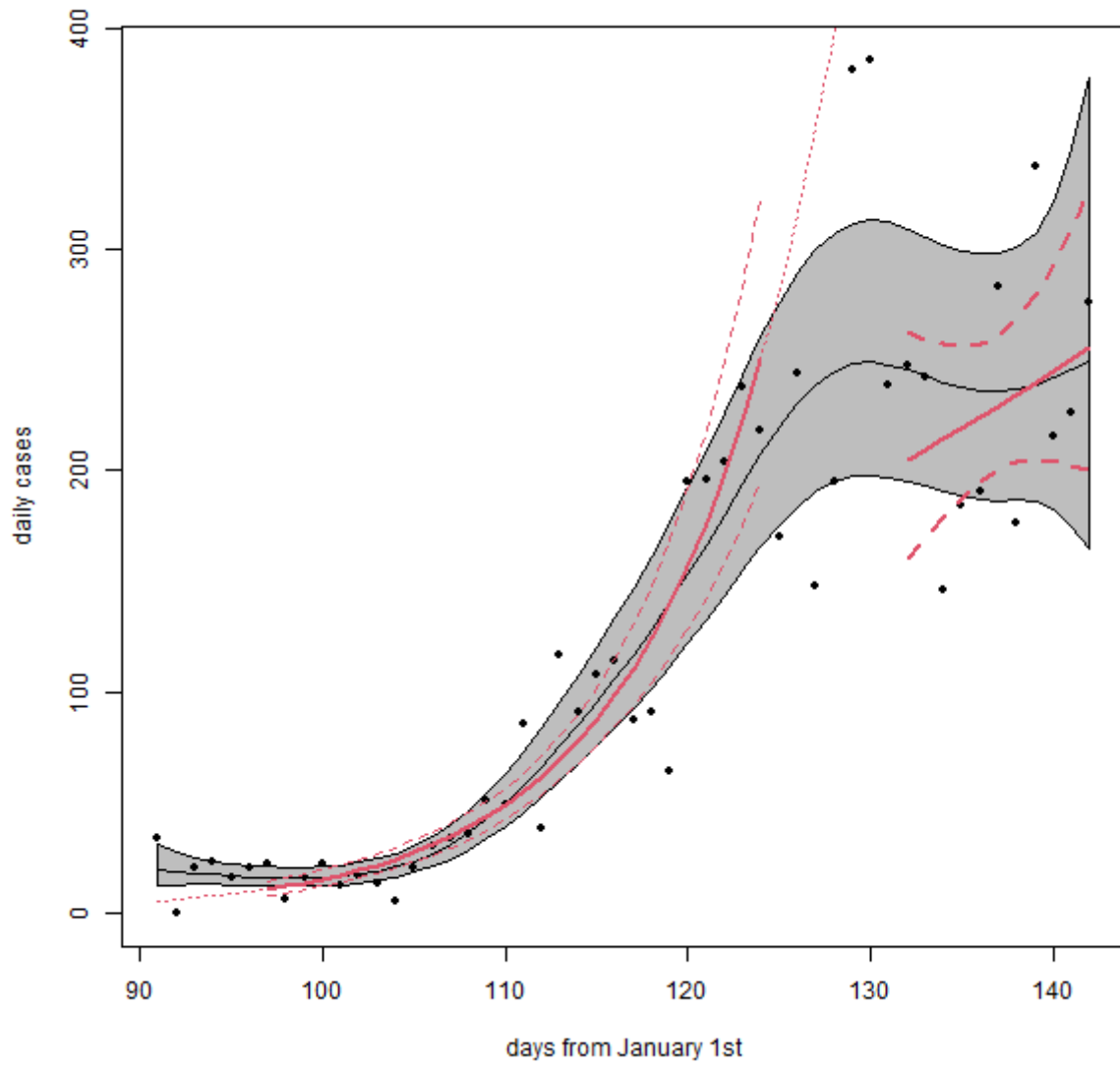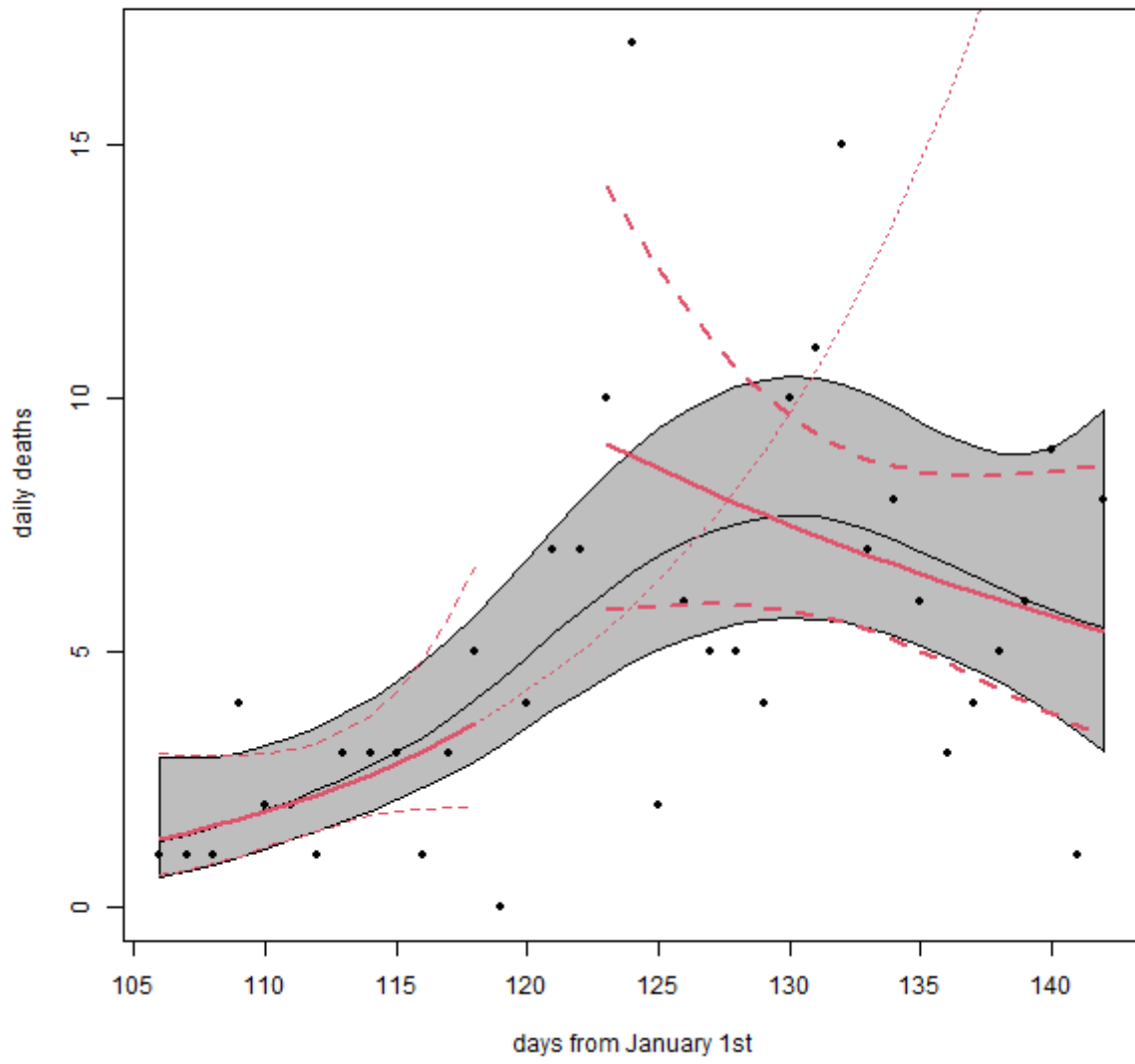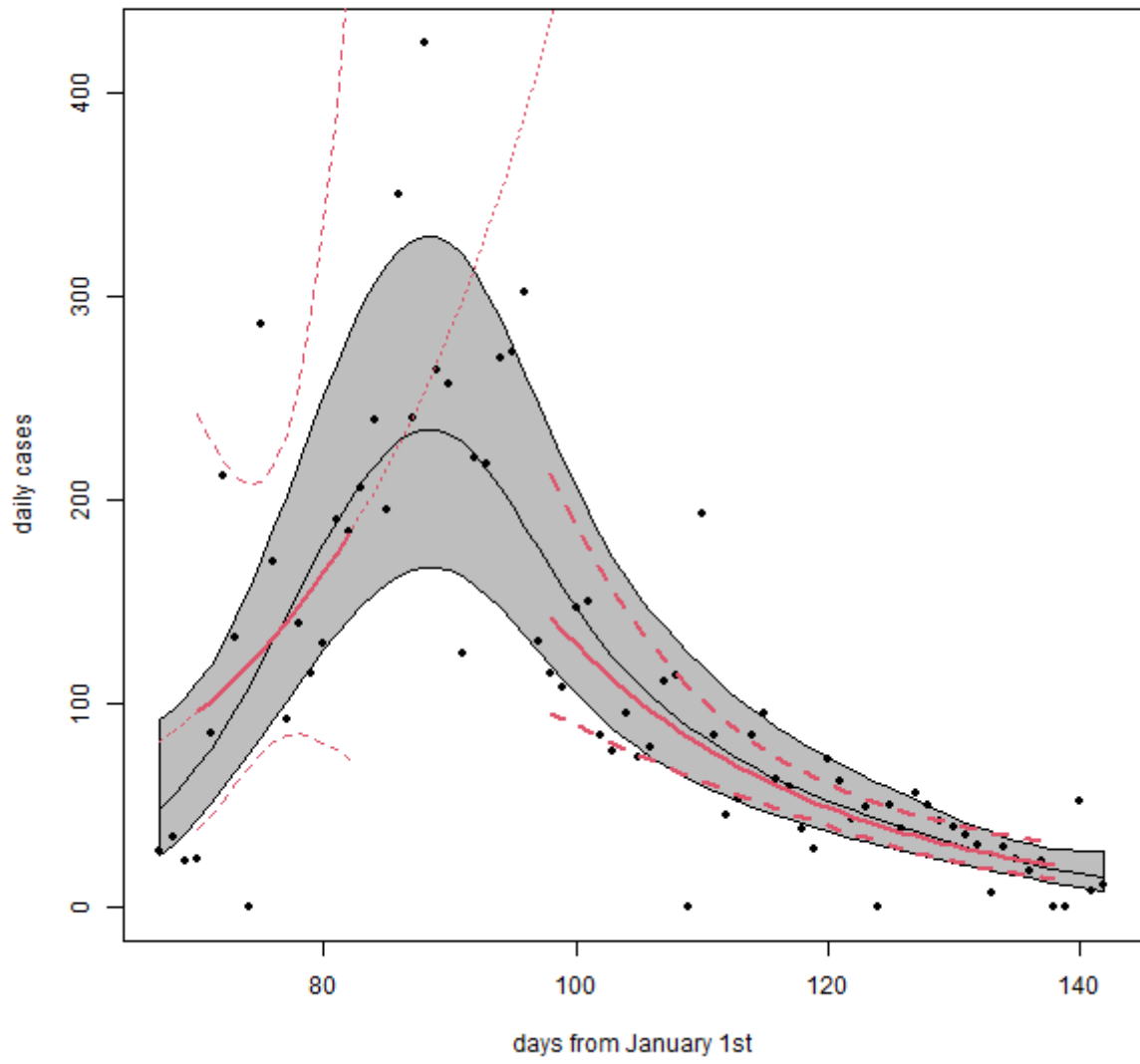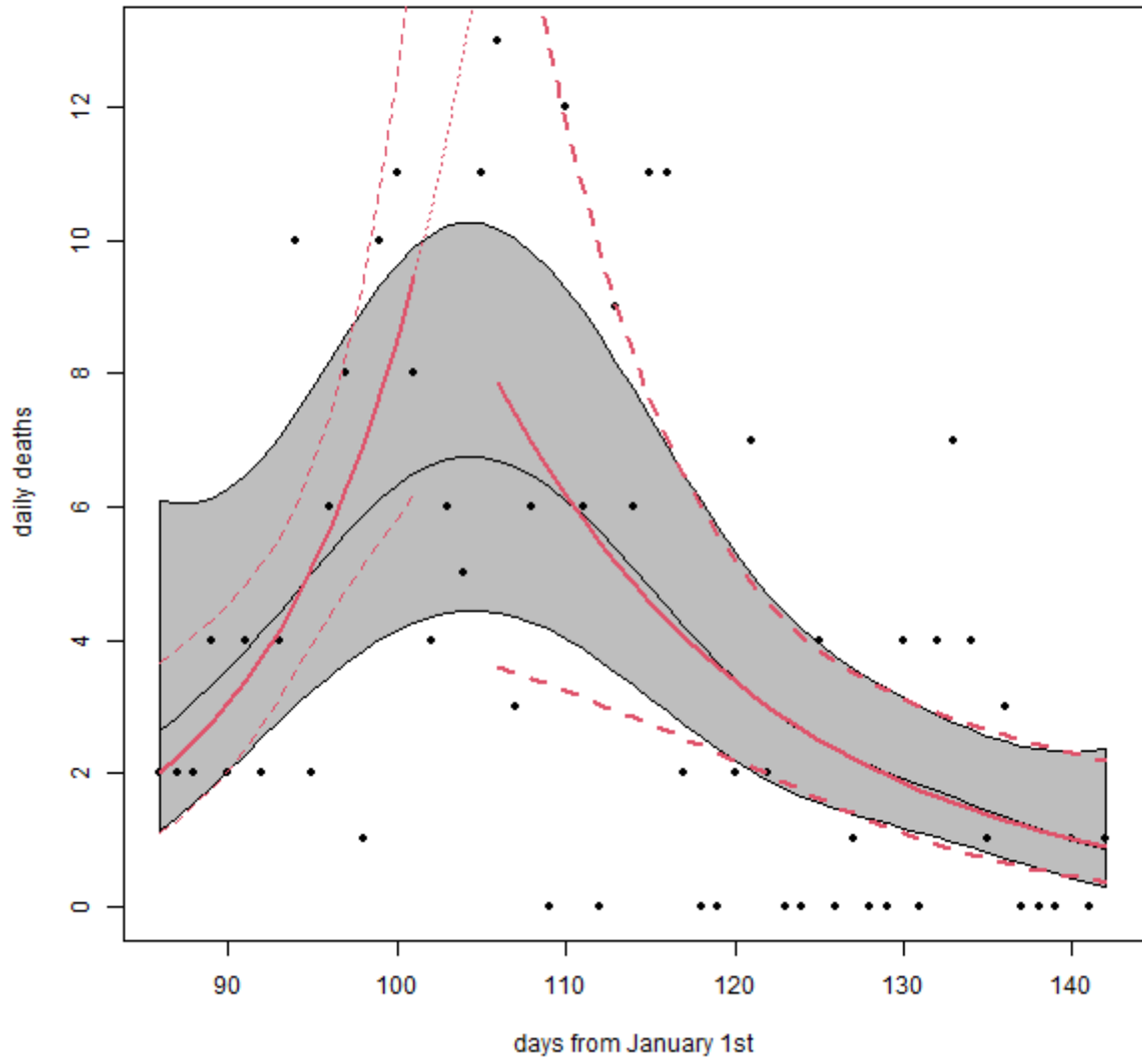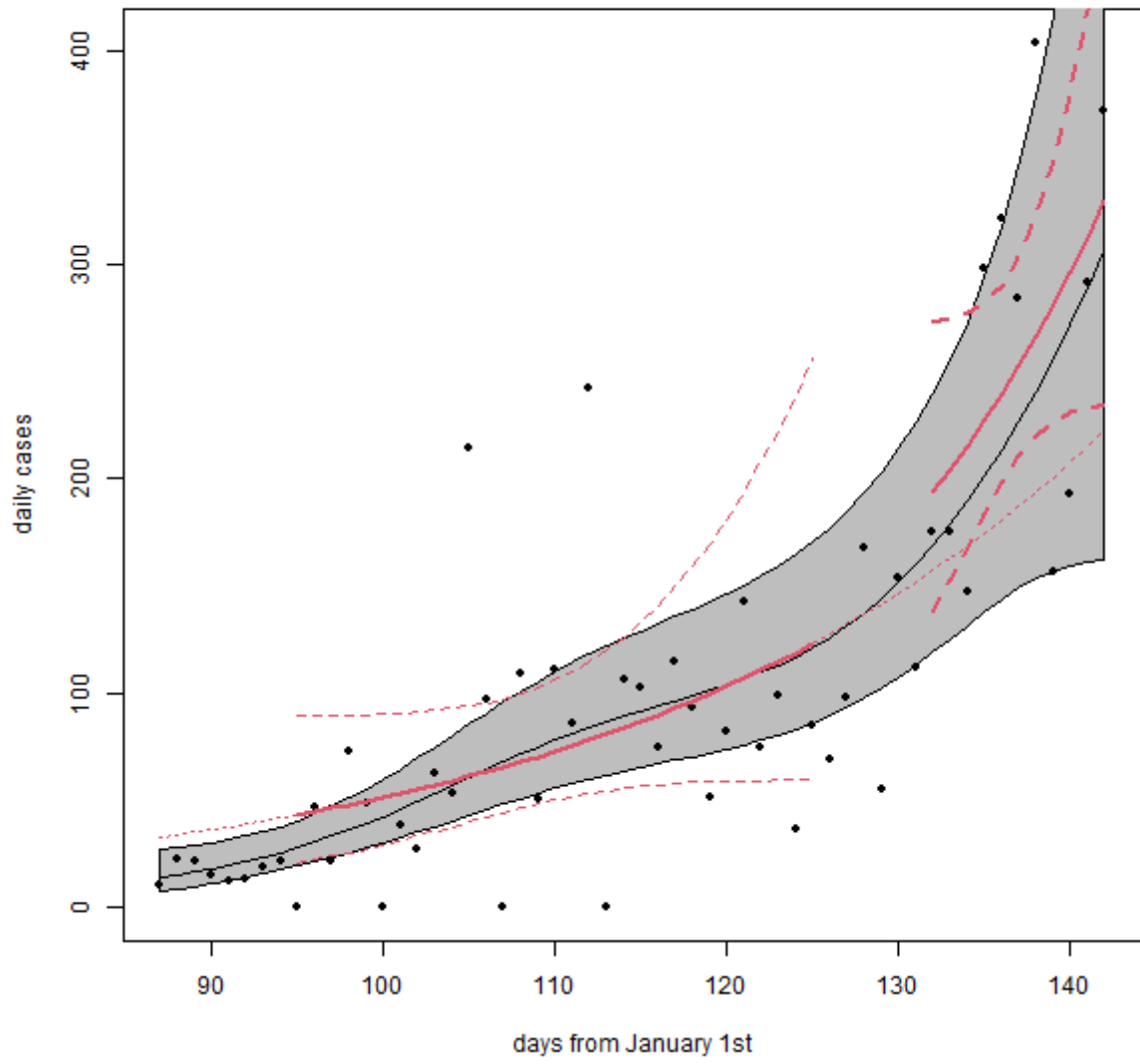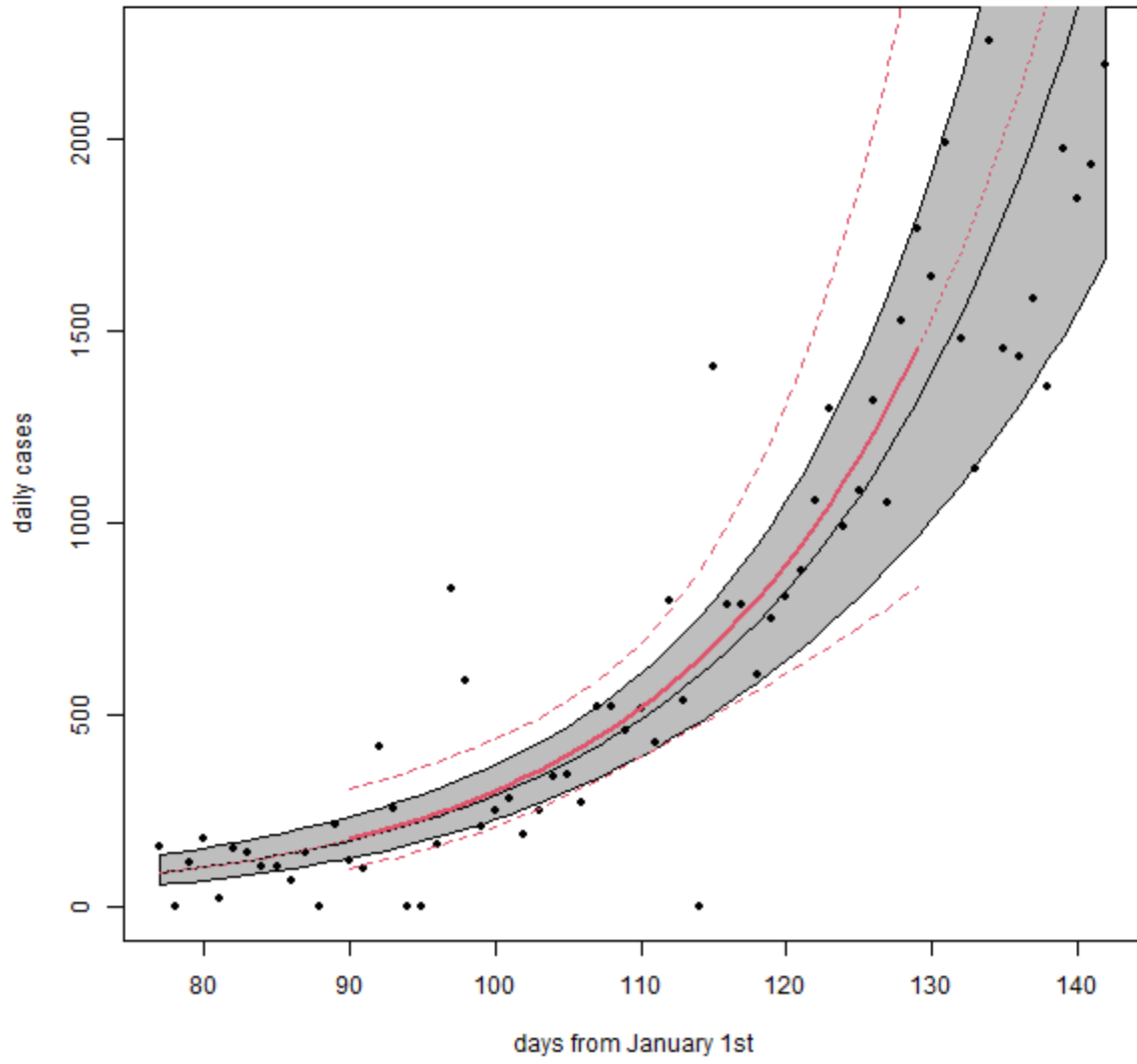
daily cases

days from January 1st

Netherlands

daily cases

days from January 1st

Netherlands

daily deaths

days from January 1st

Nigeria

Nigeria

days from January 1st

Norway

daily cases

days from January 1st

Norway

daily deaths

days from January 1st

Oman

days from January 1st

Pakistan

daily cases

days from January 1st

Pakistan

daily deaths

days from January 1st

Panama

days from January 1st

Panama

daily deaths

days from January 1st

Peru

days from January 1st

Peru

daily deaths

days from January 1st

Philippines

daily cases

days from January 1st

Philippines

days from January 1st

Poland

Poland

daily deaths

days from January 1st
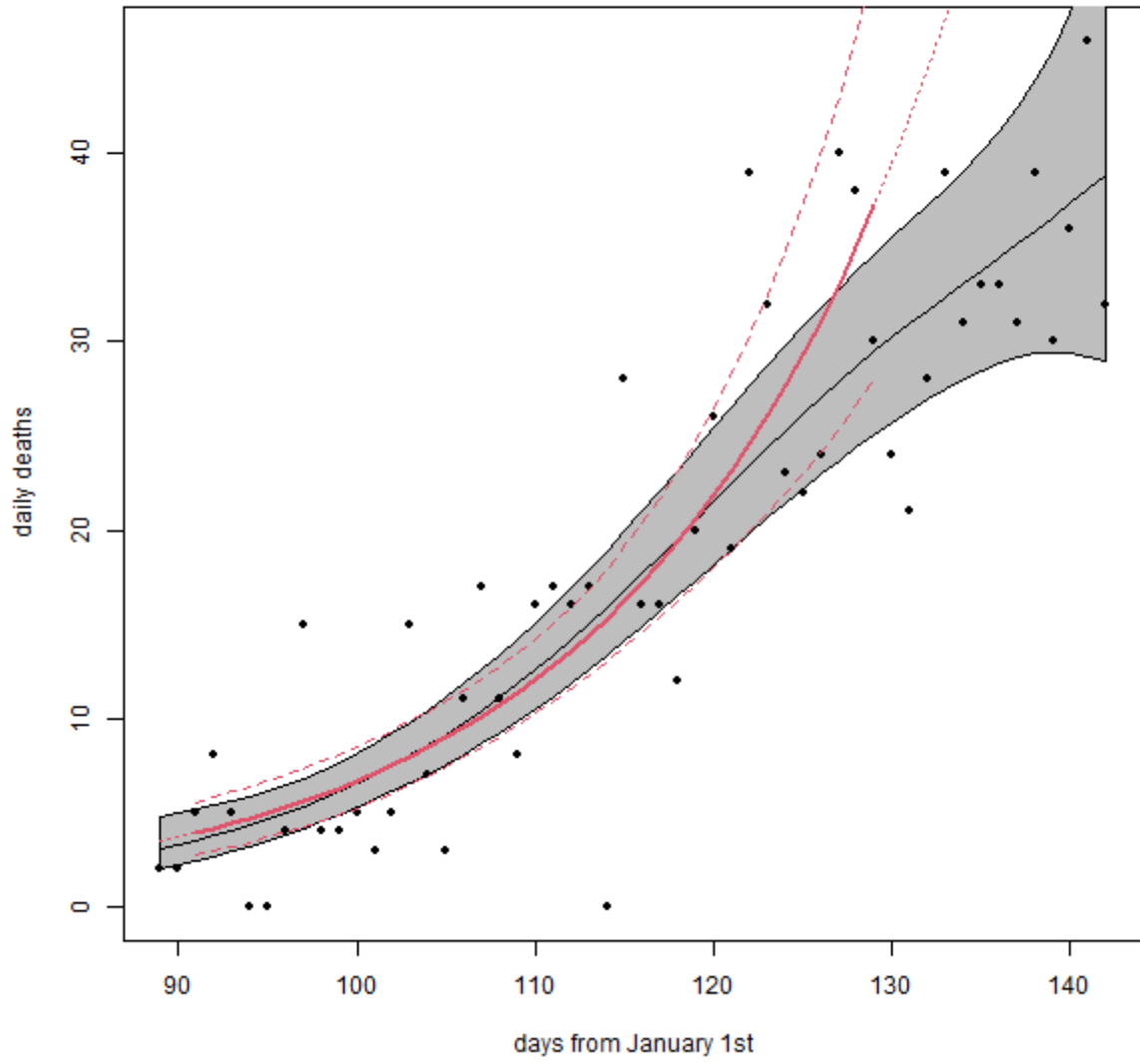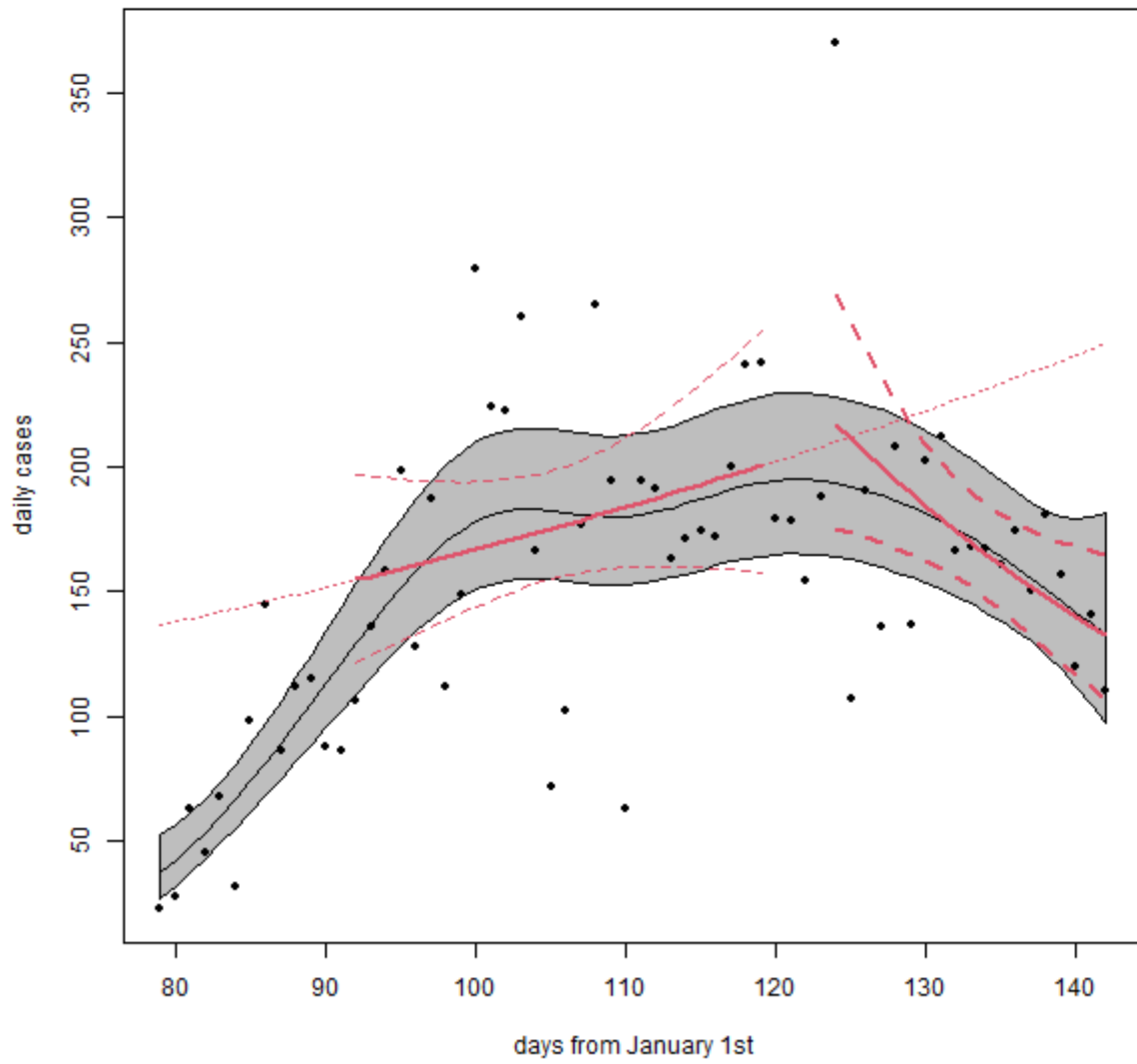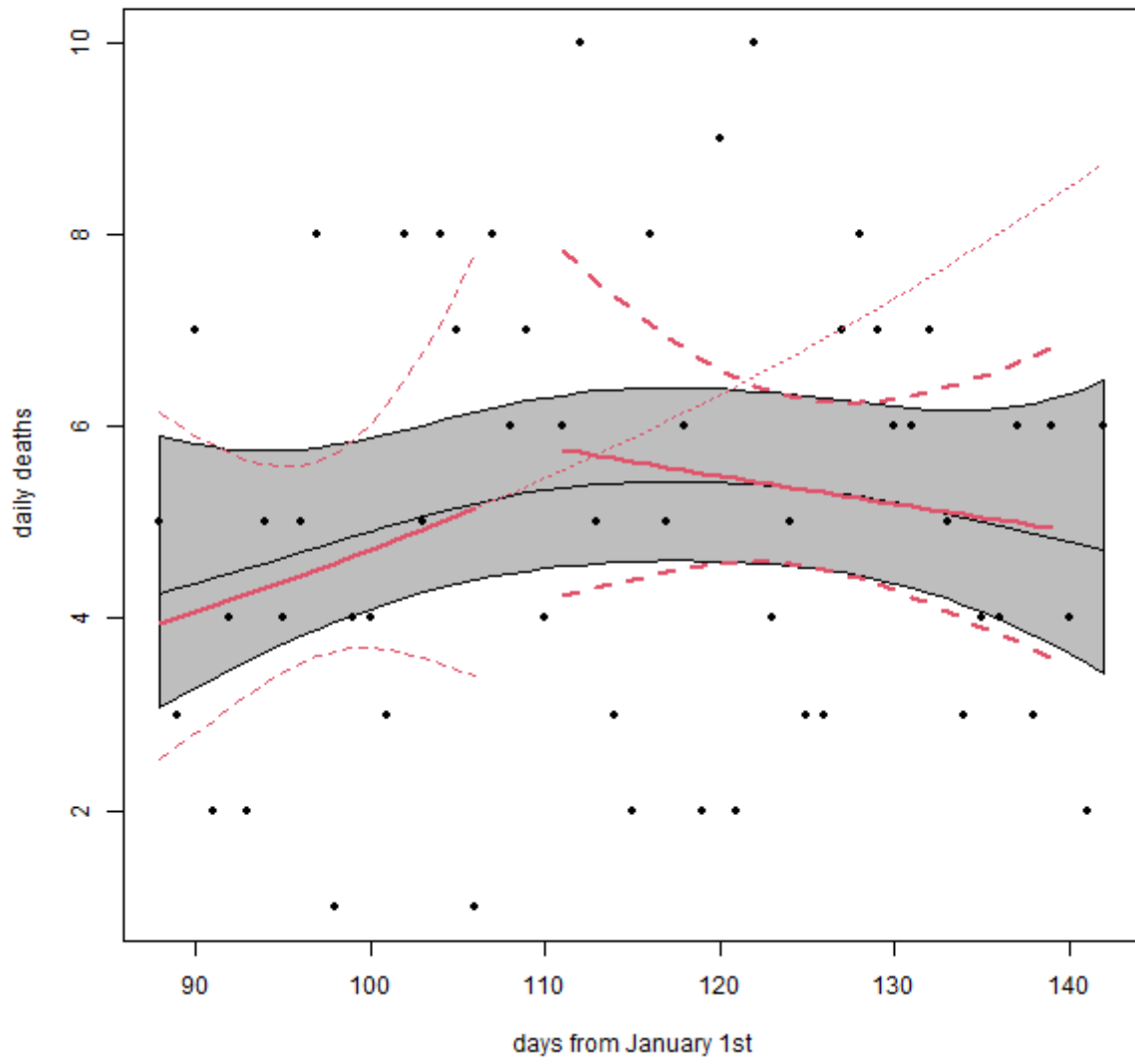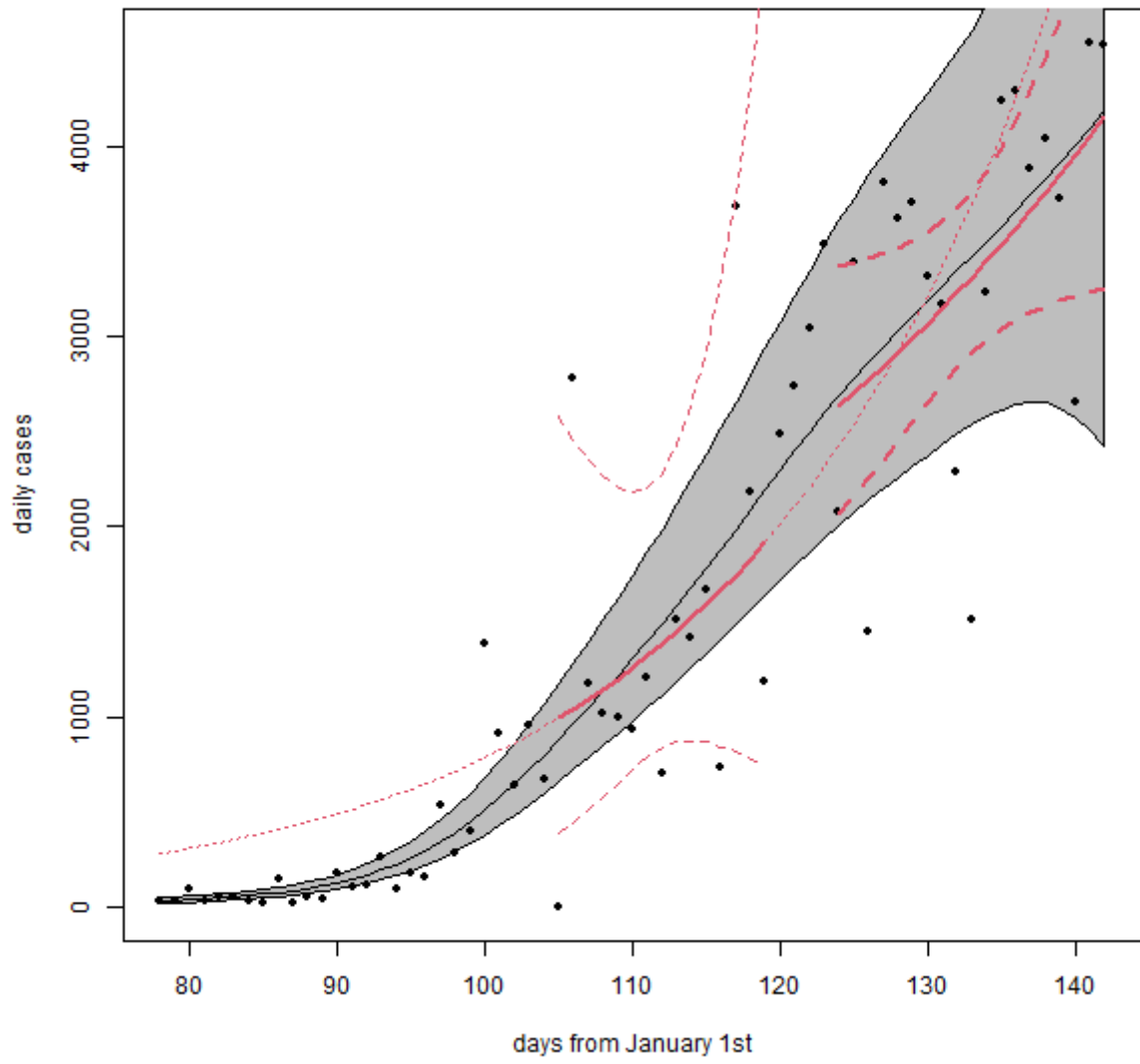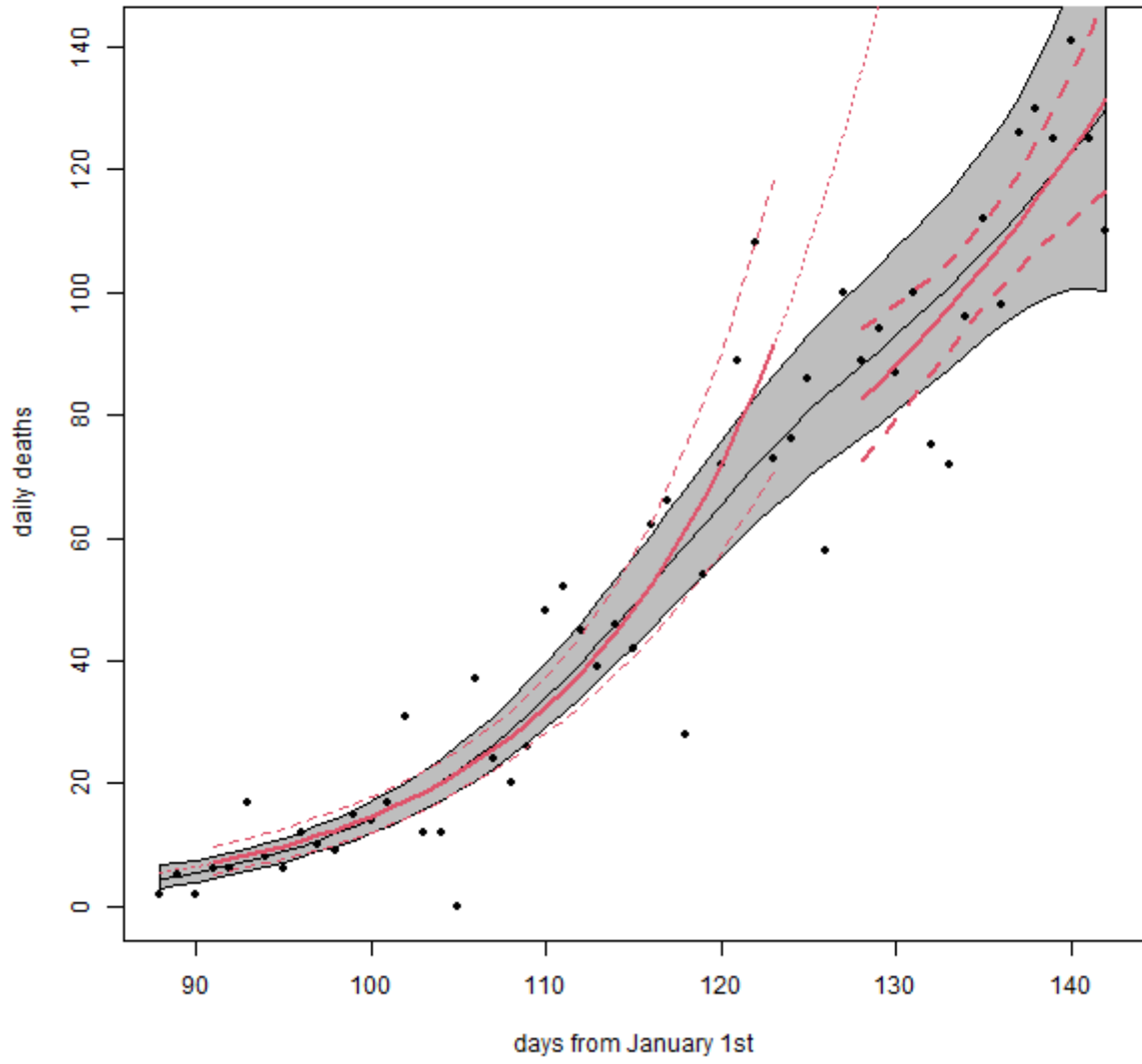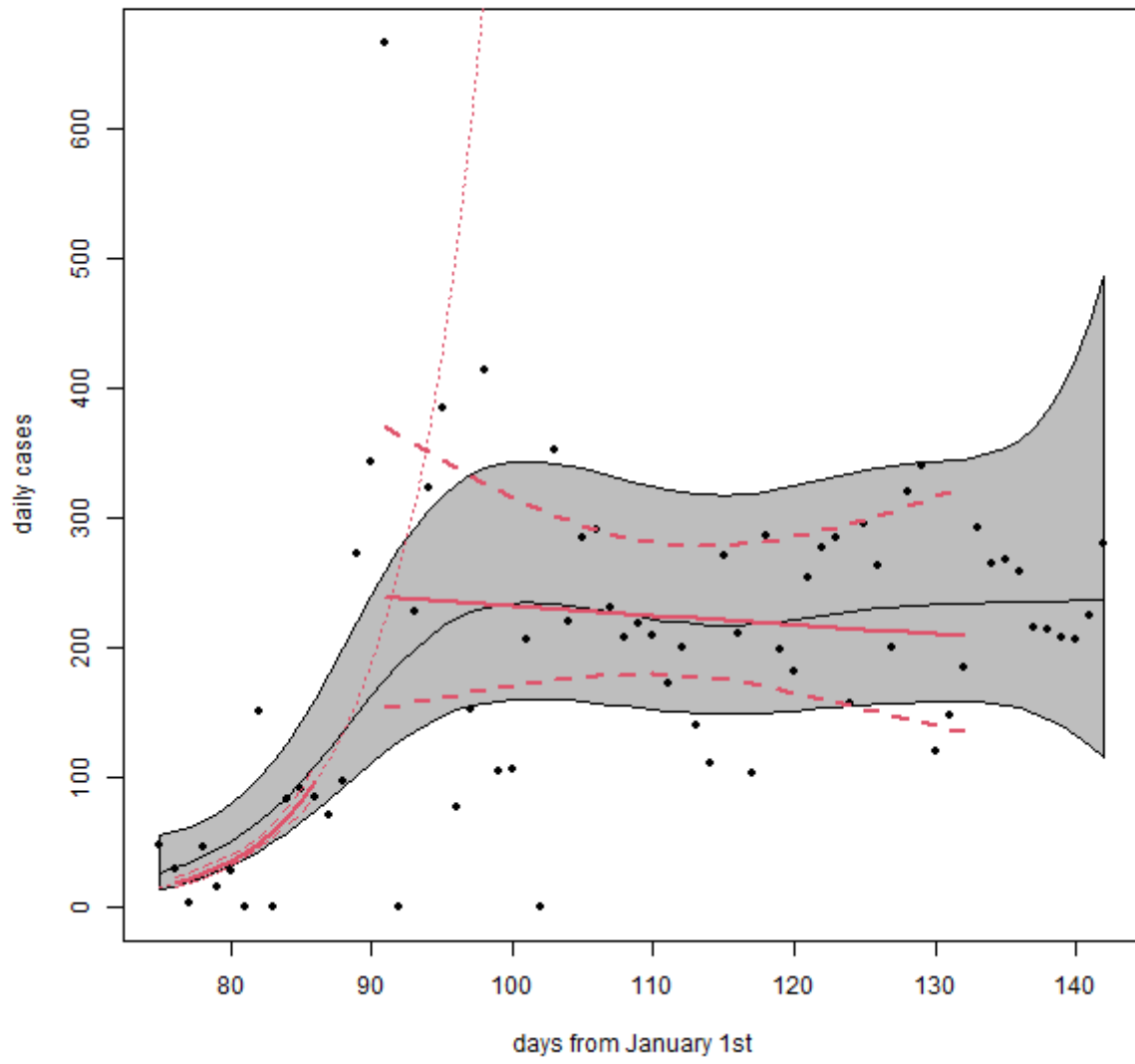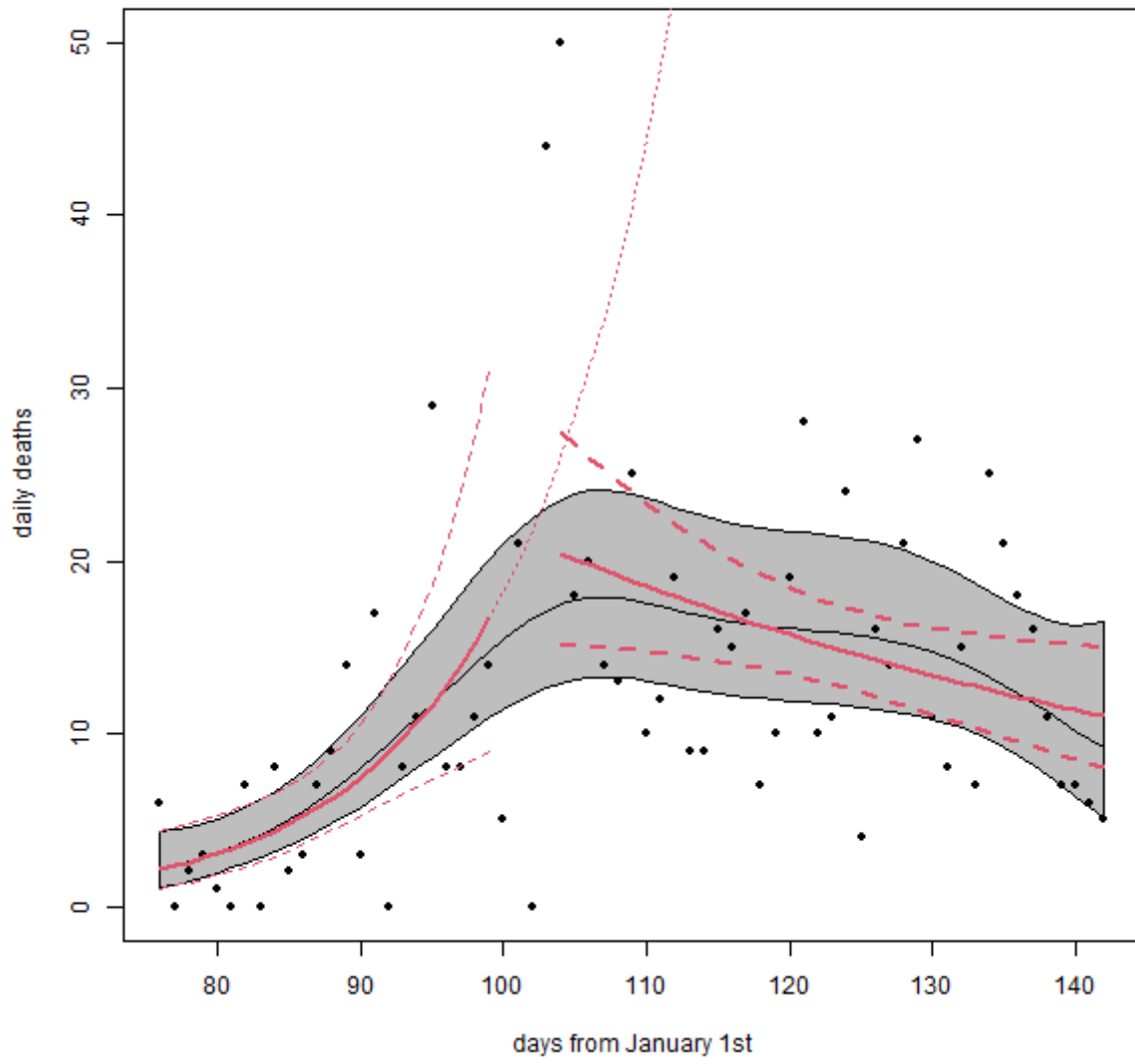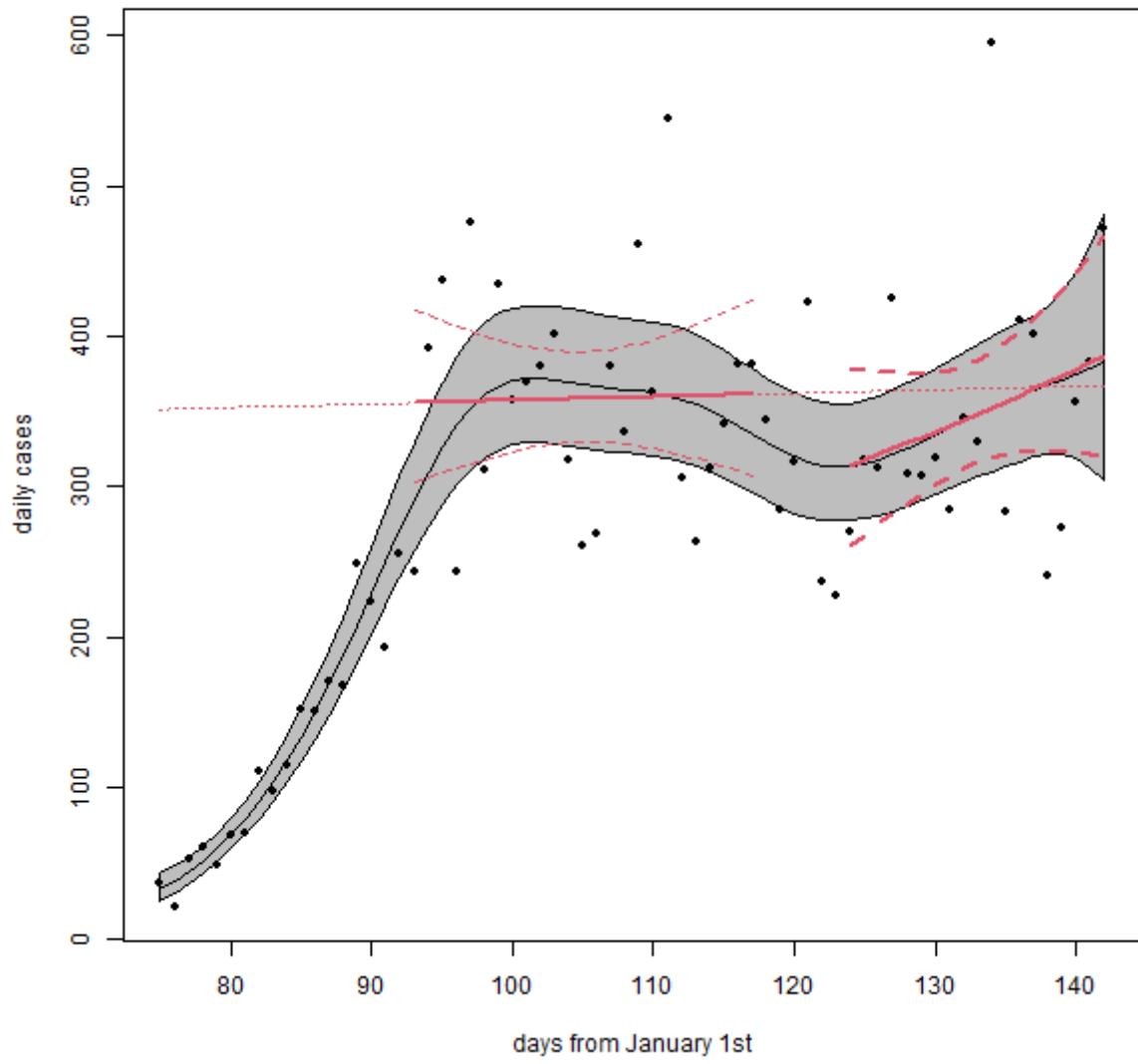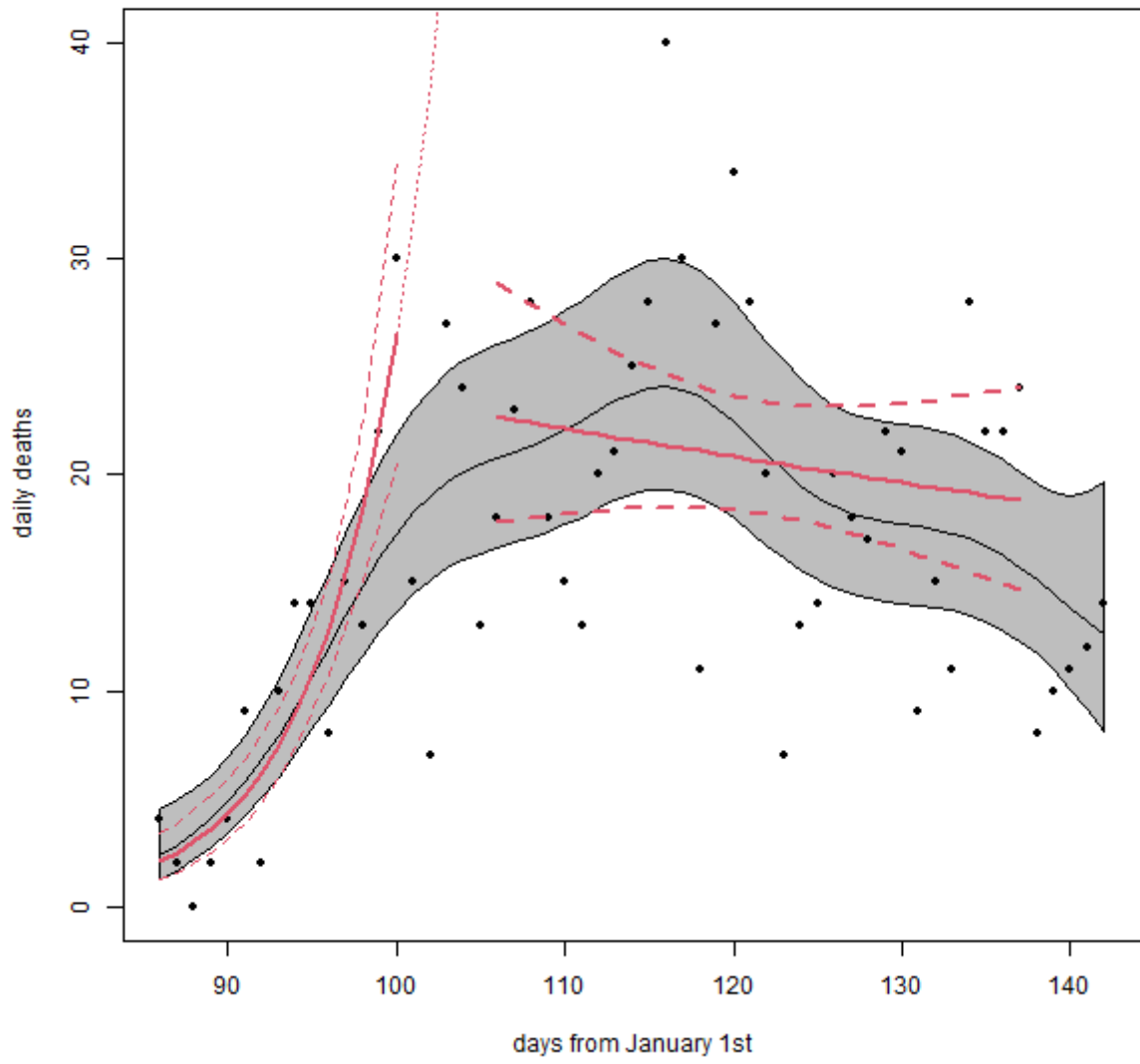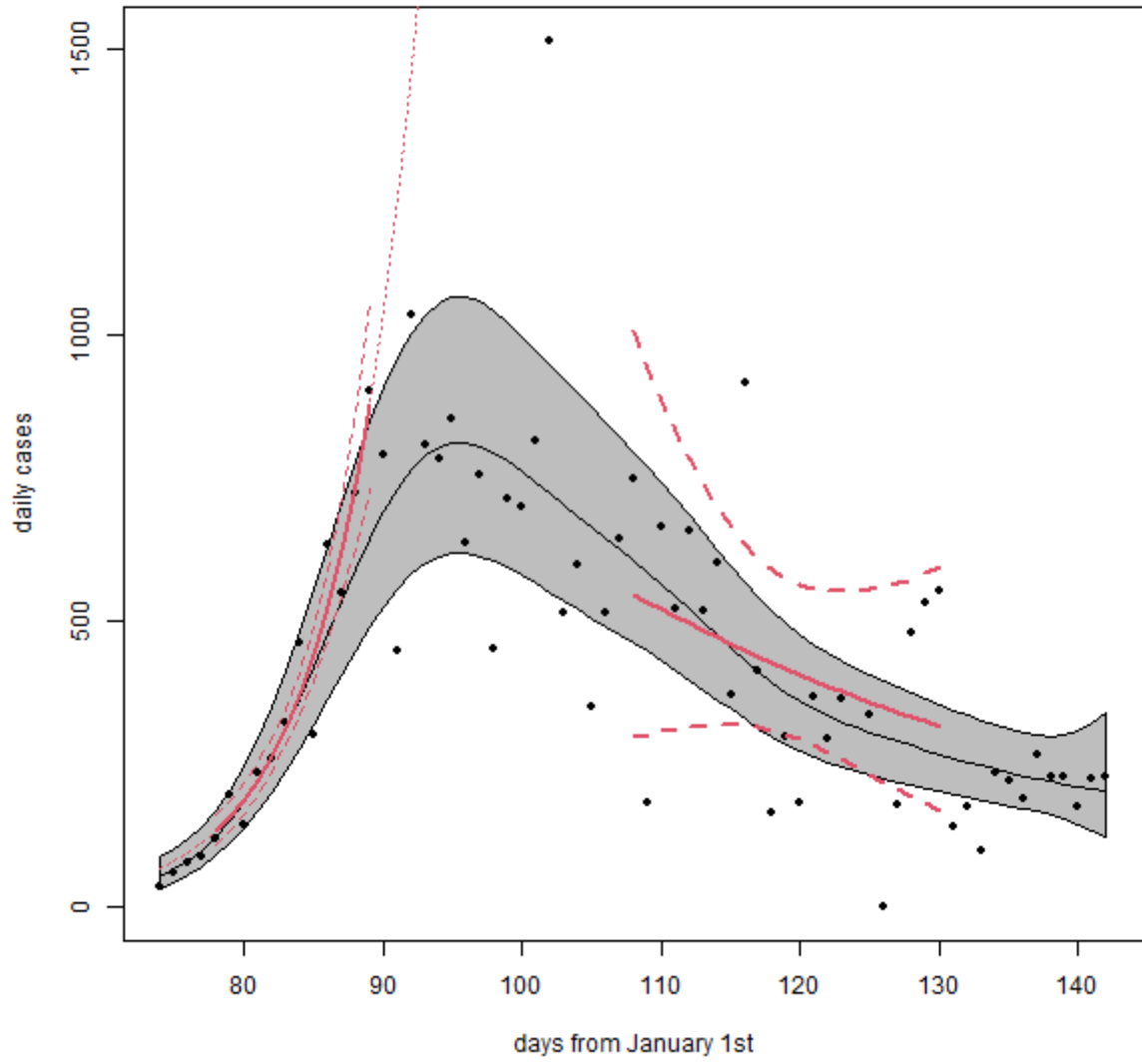
Portugal

Portugal
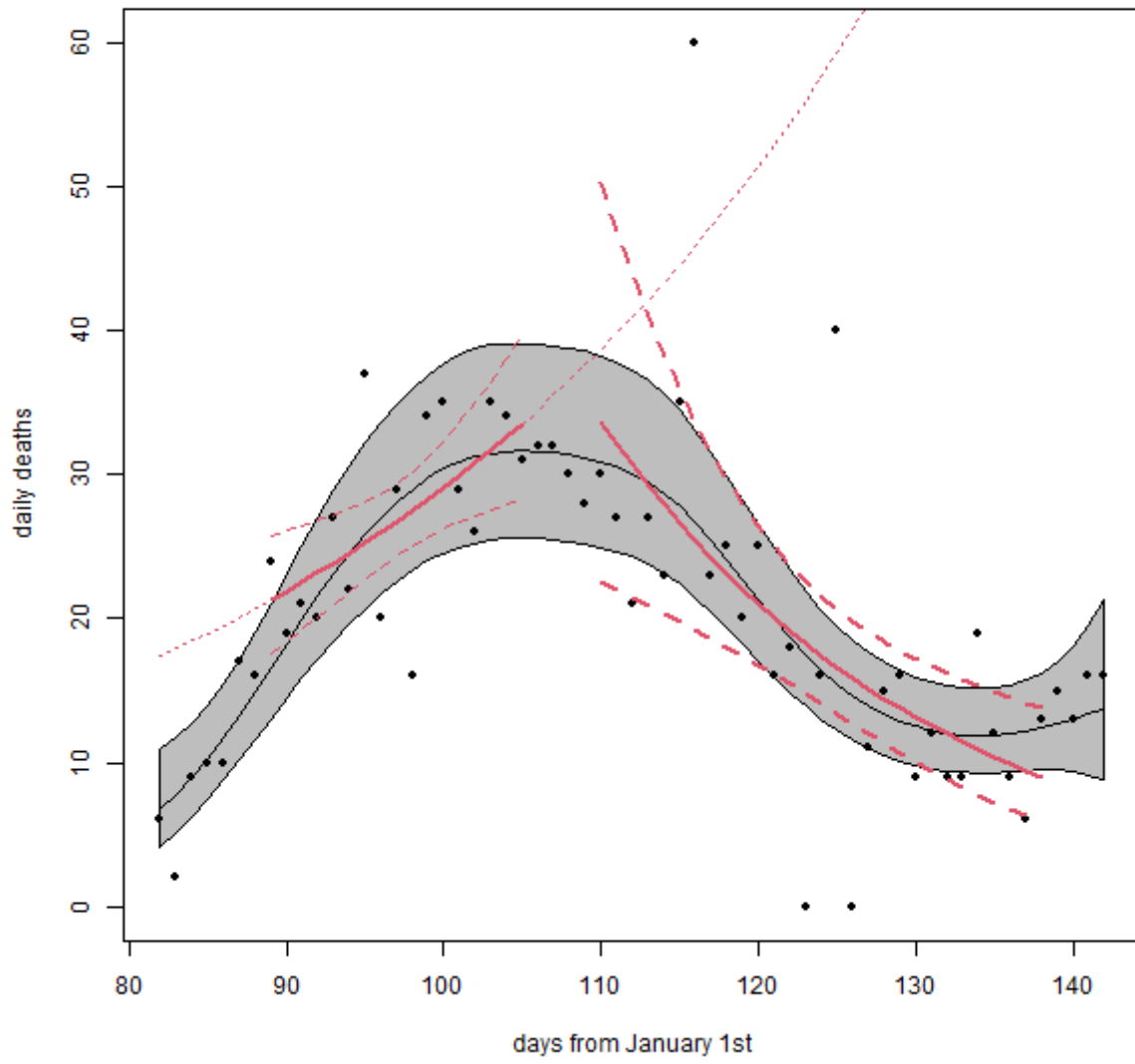
Puerto_Rico

Qatar

Romania

daily cases

days from January 1st

Romania

daily deaths

days from January 1st

Russia

Russia

days from January 1st

Saudi_Arabia

daily cases

days from January 1st
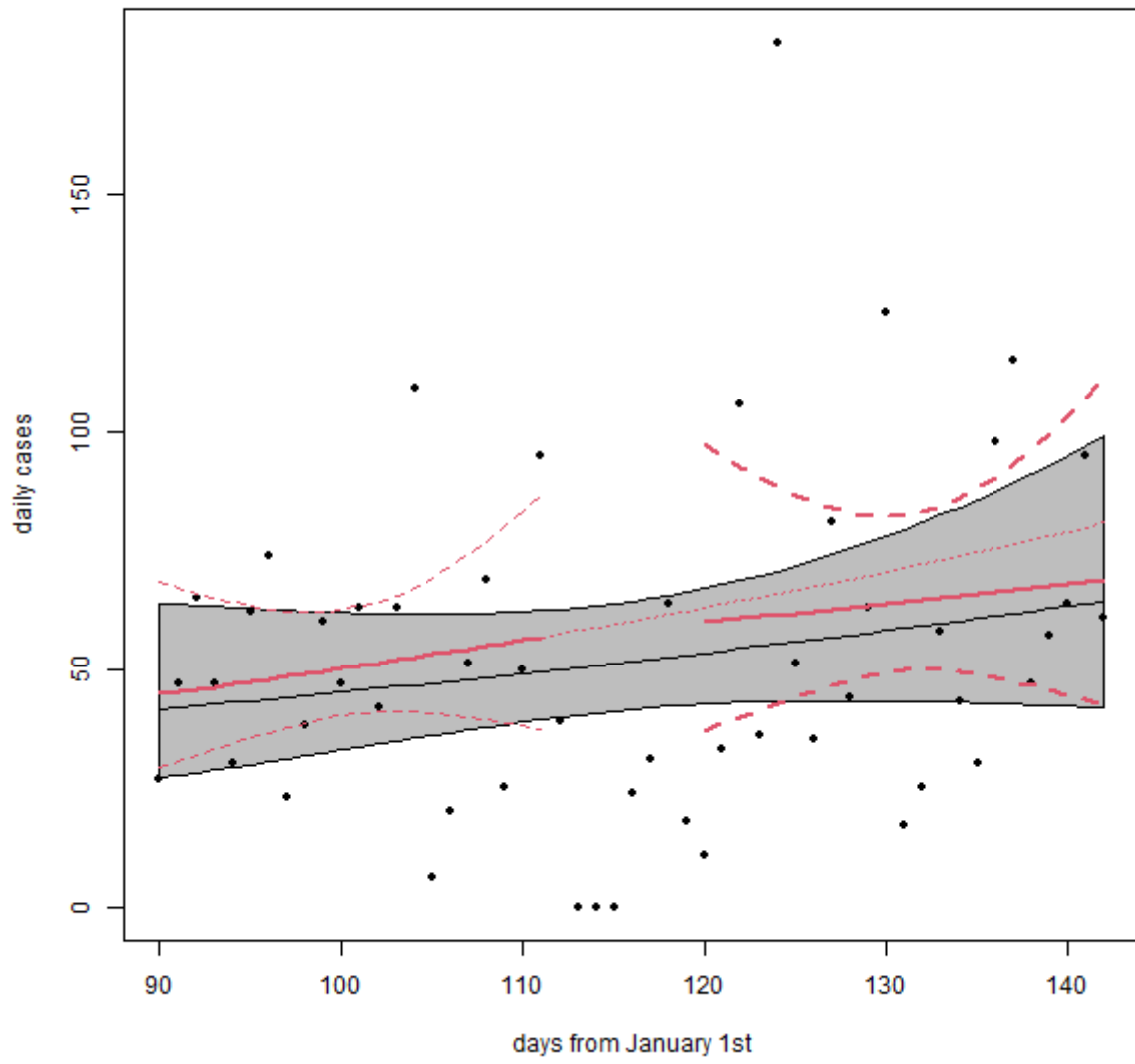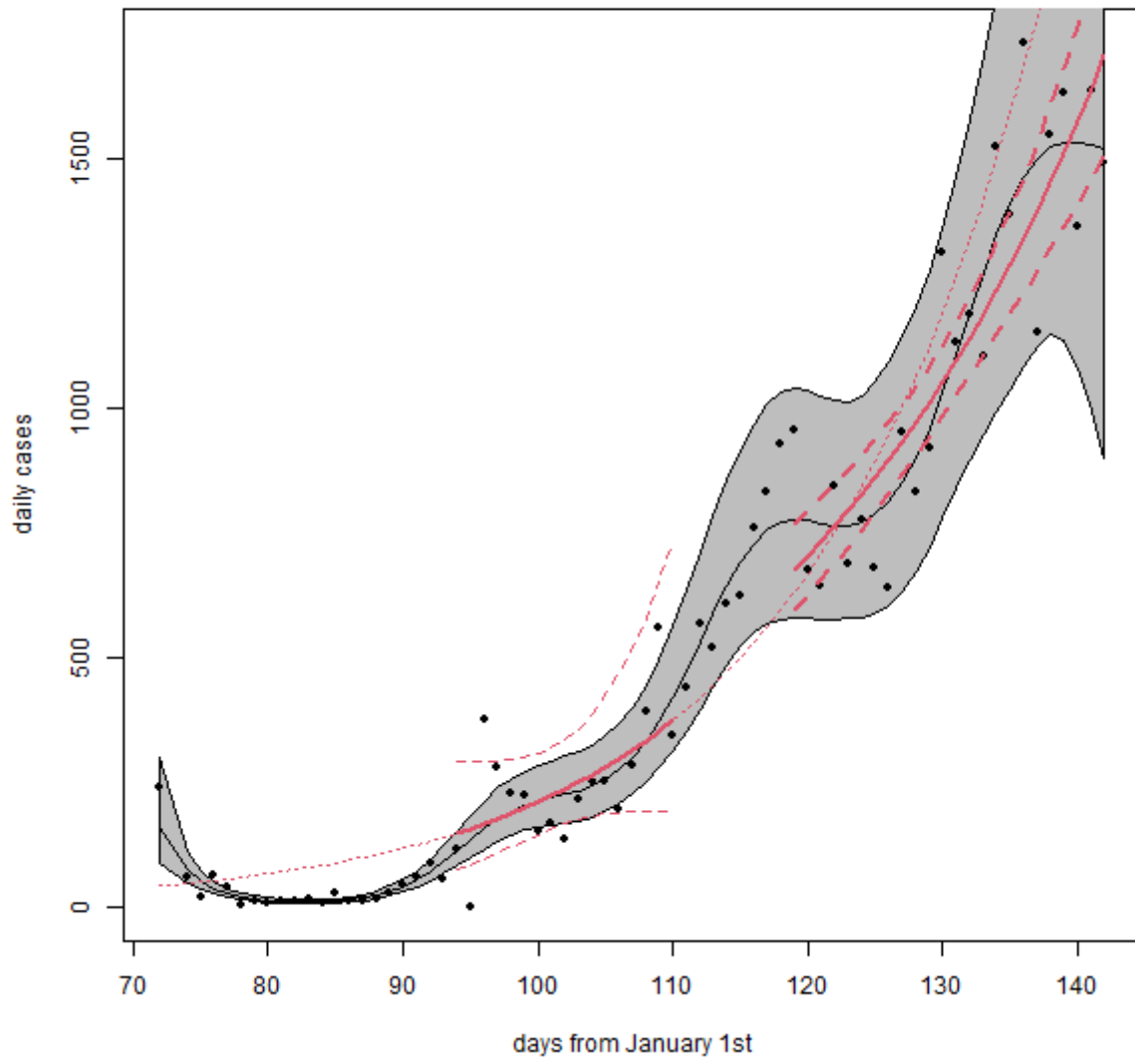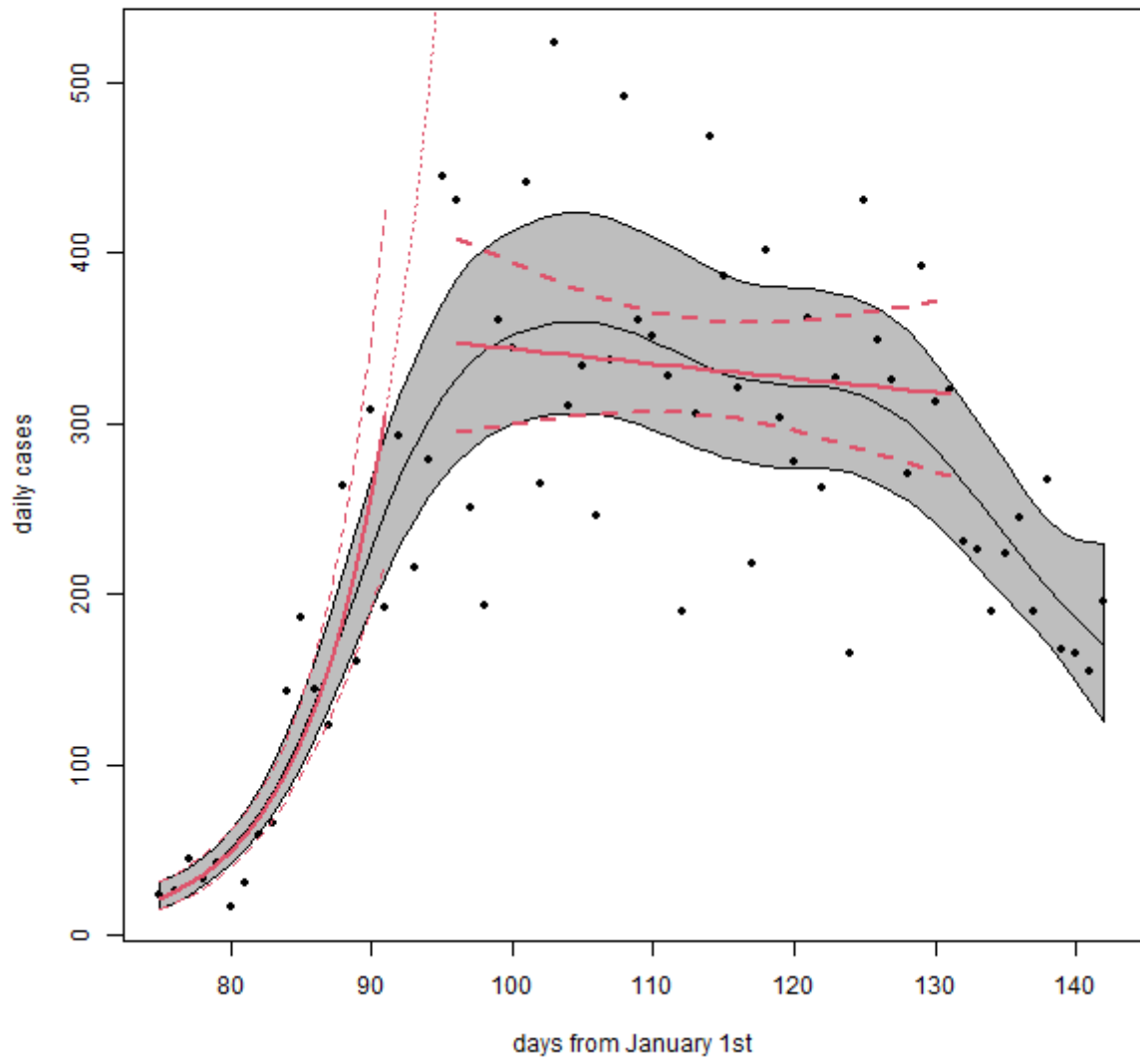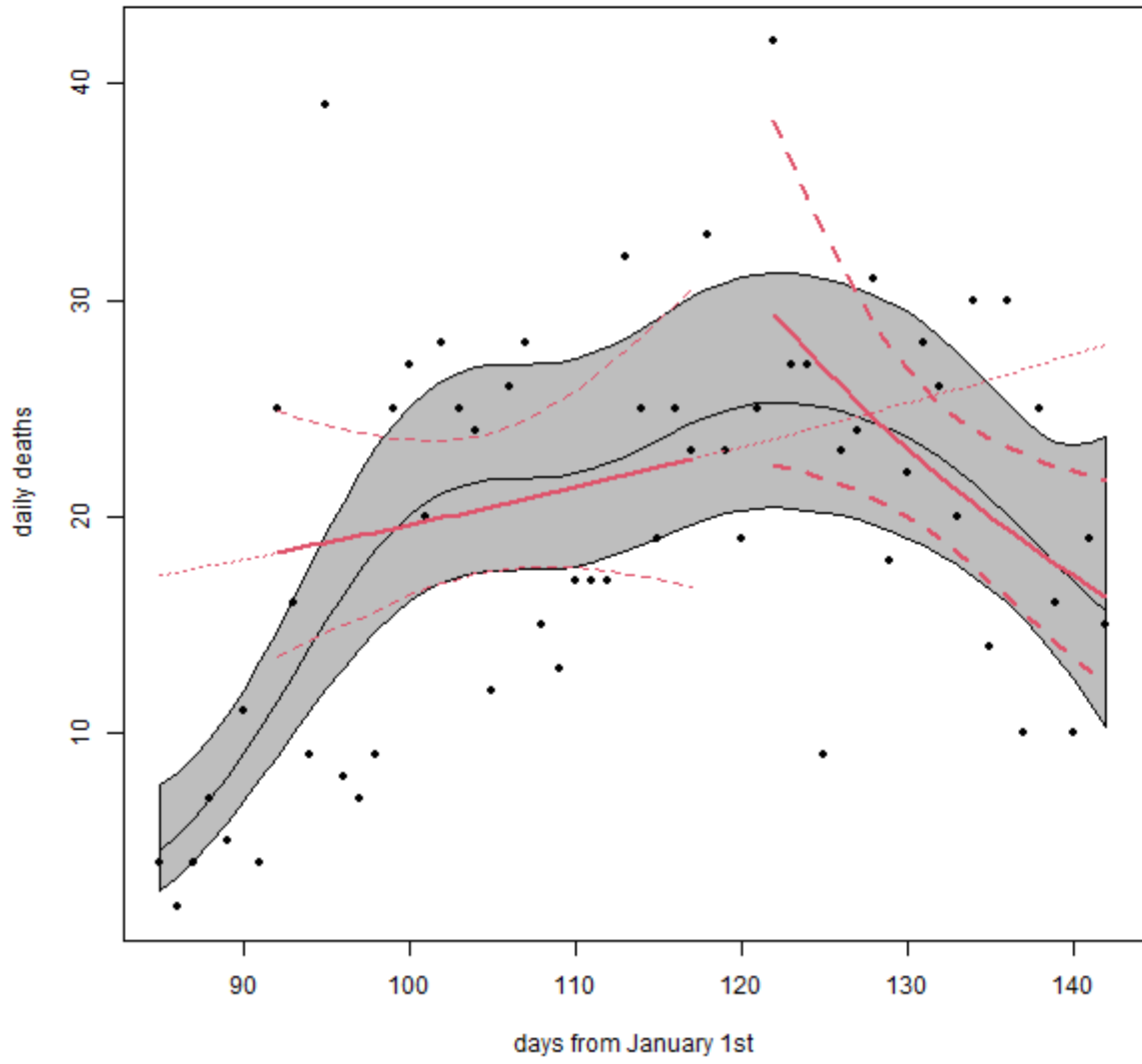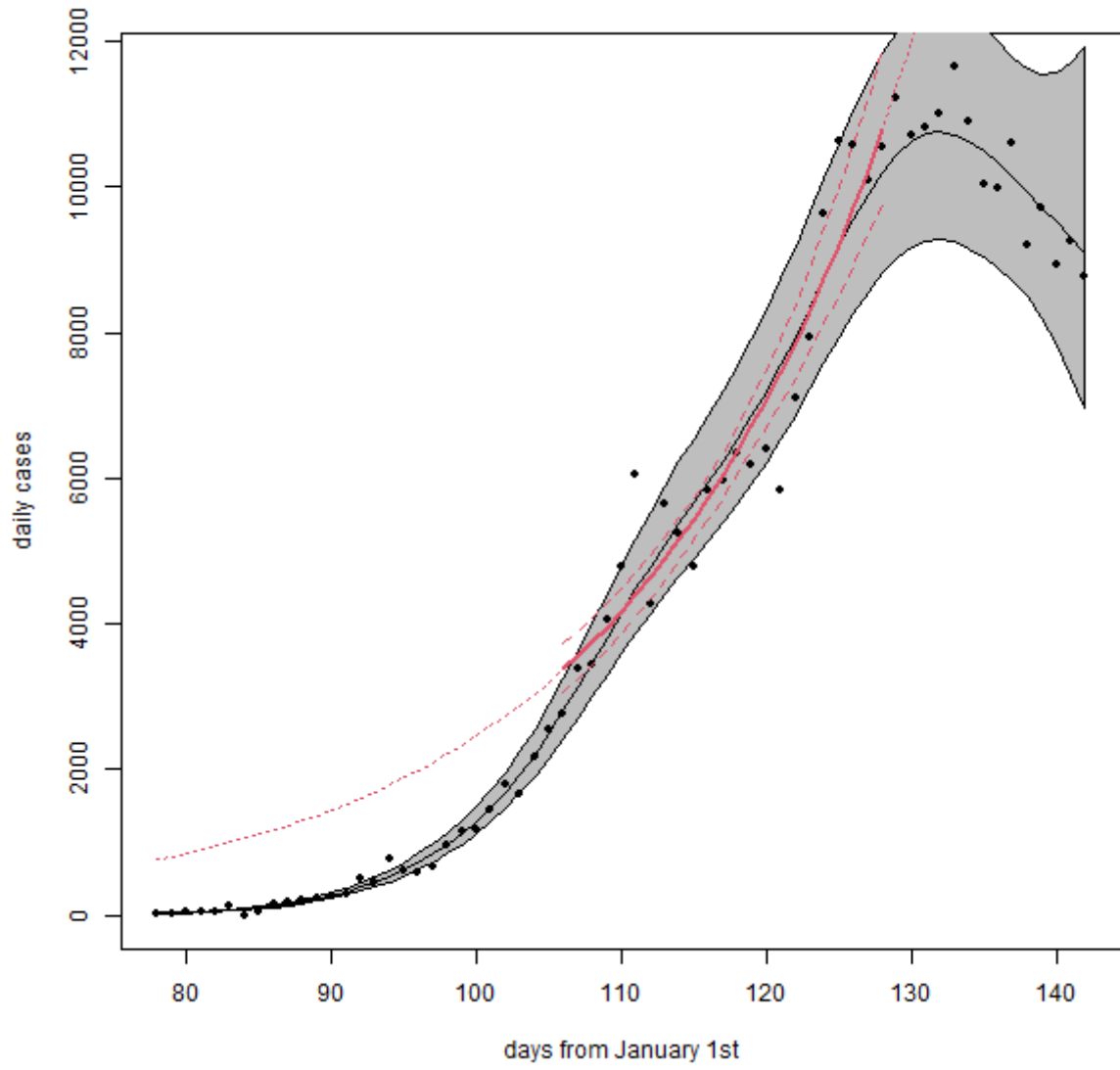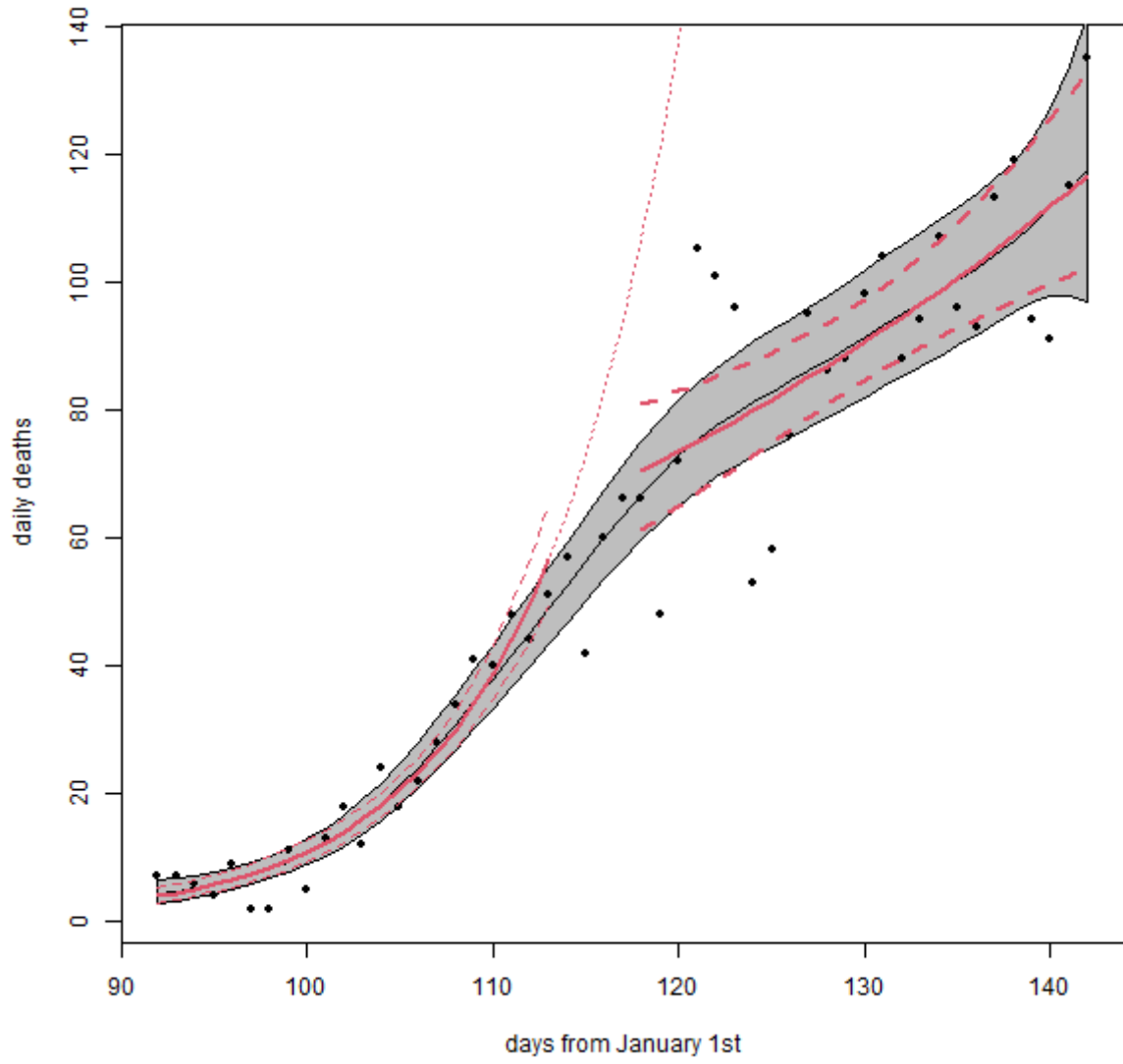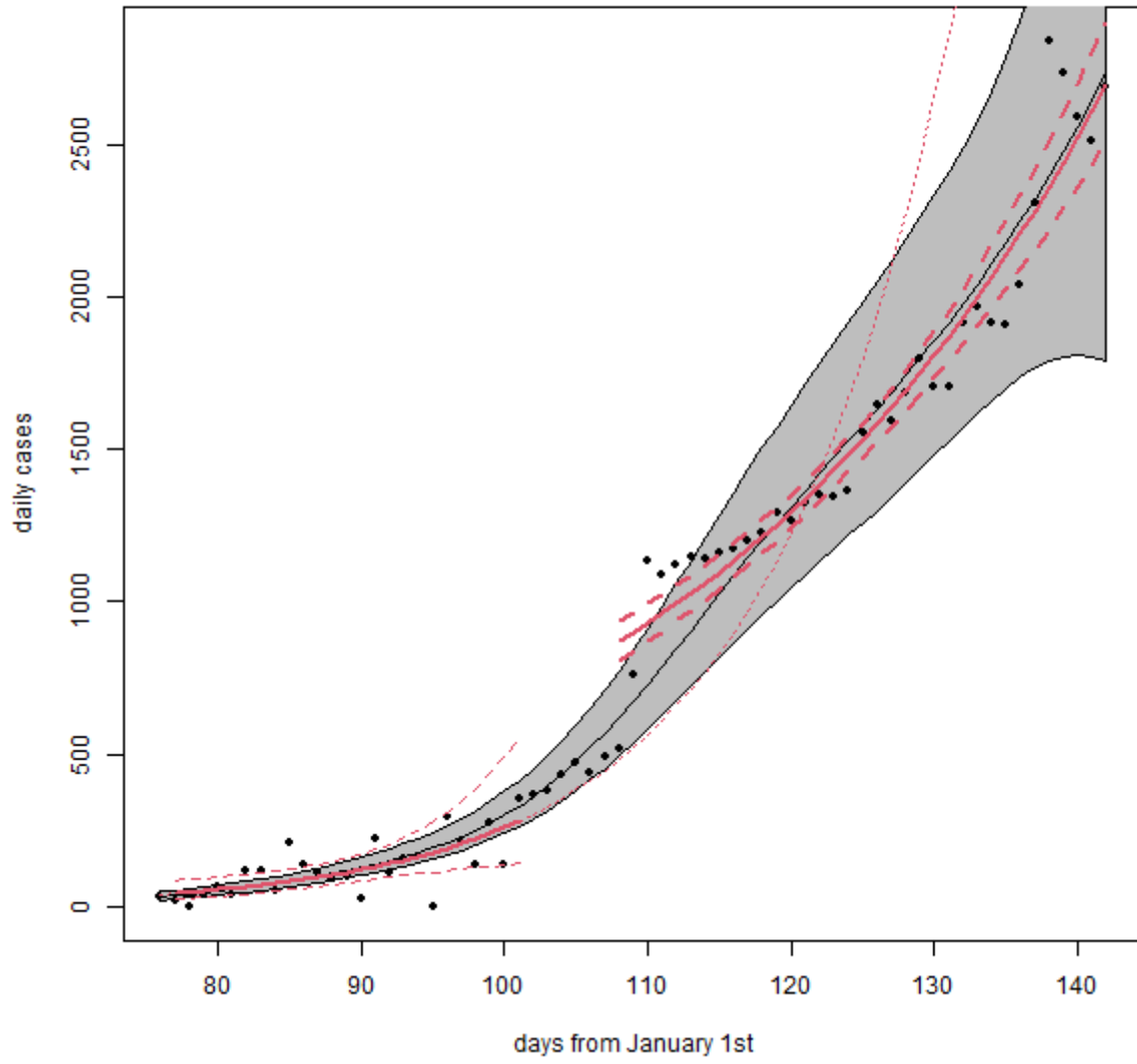
Saudi_Arabia
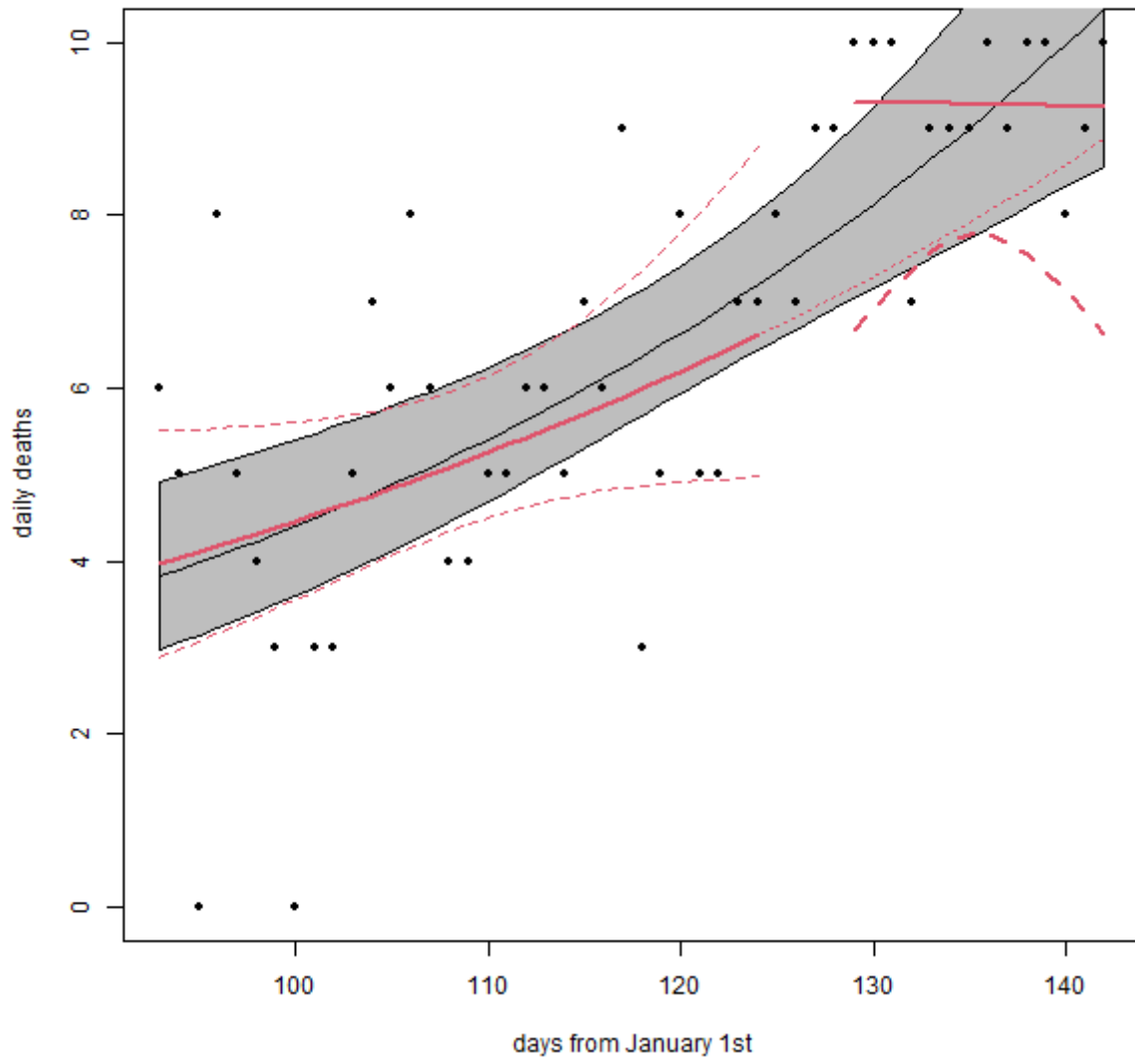
Senegal
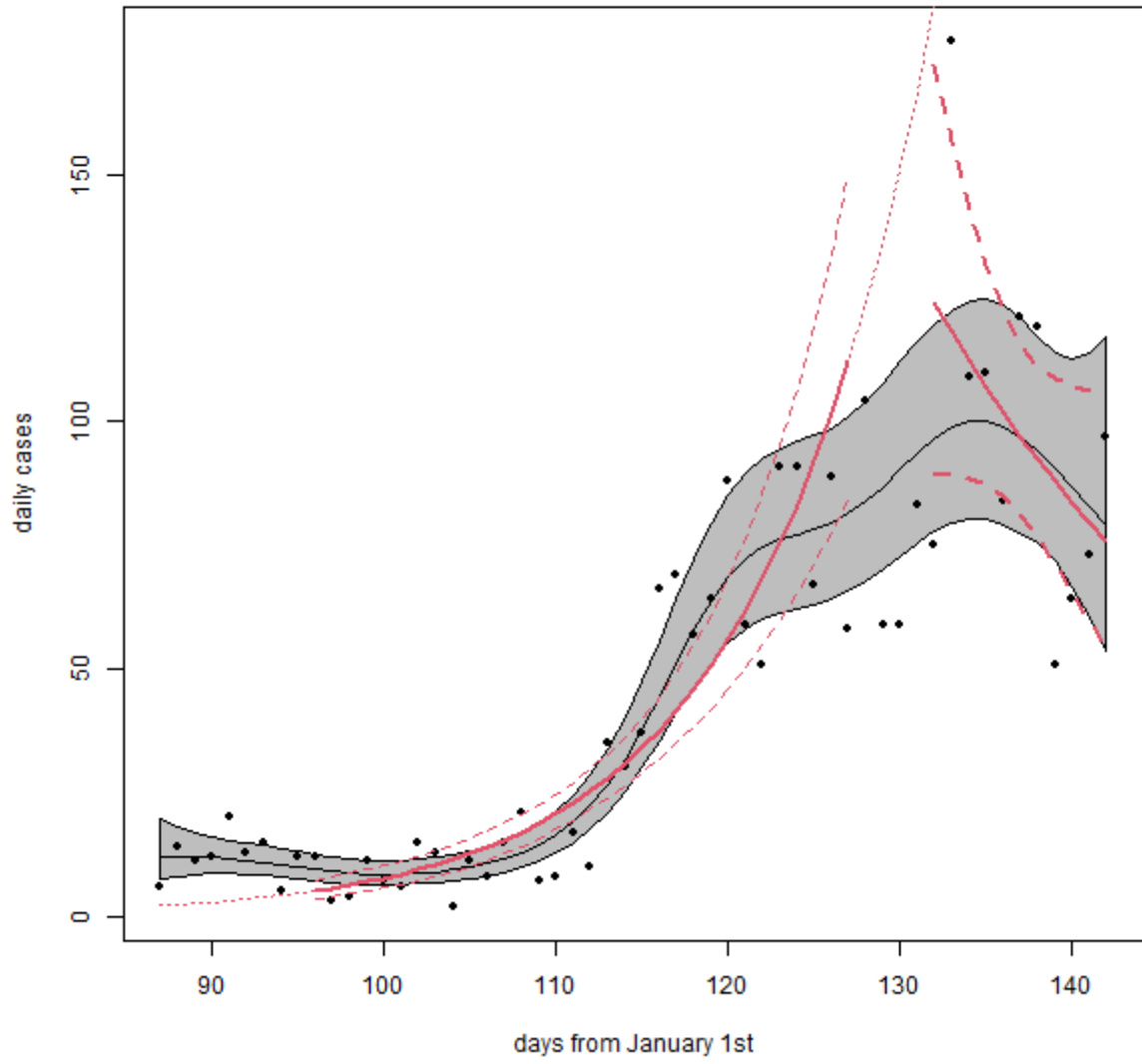
Serbia

Singapore

Slovakia

daily cases

days from January 1st
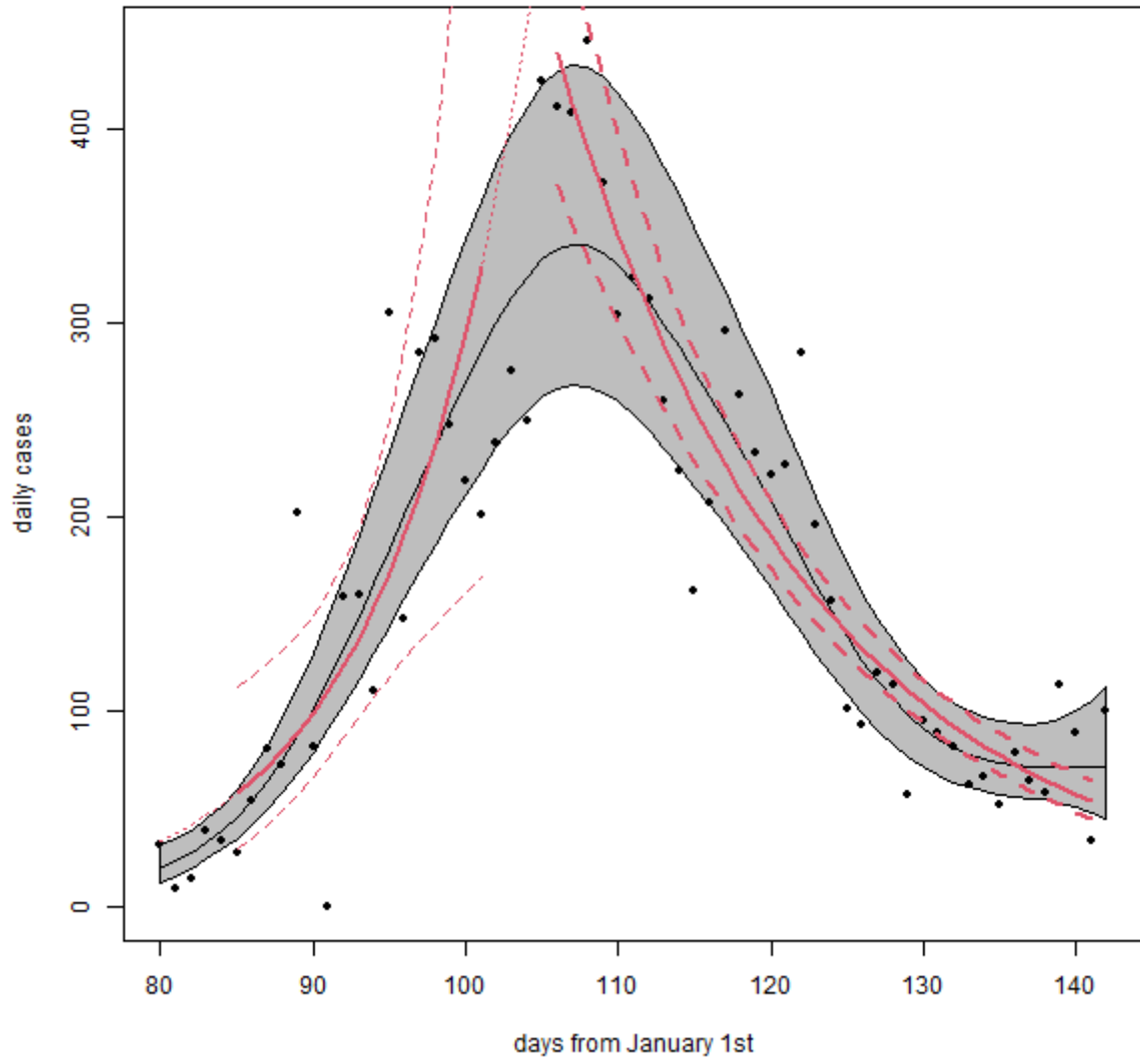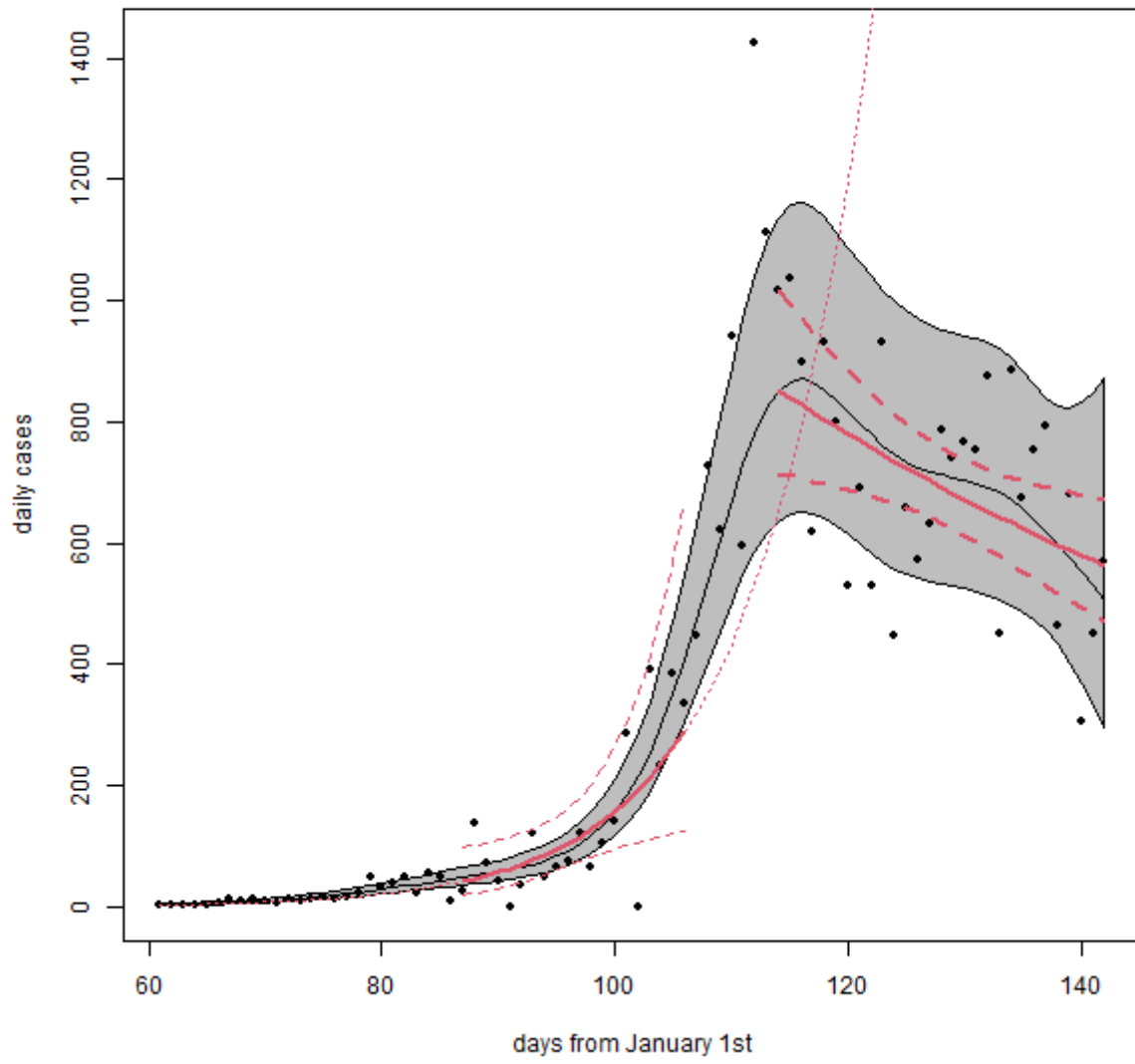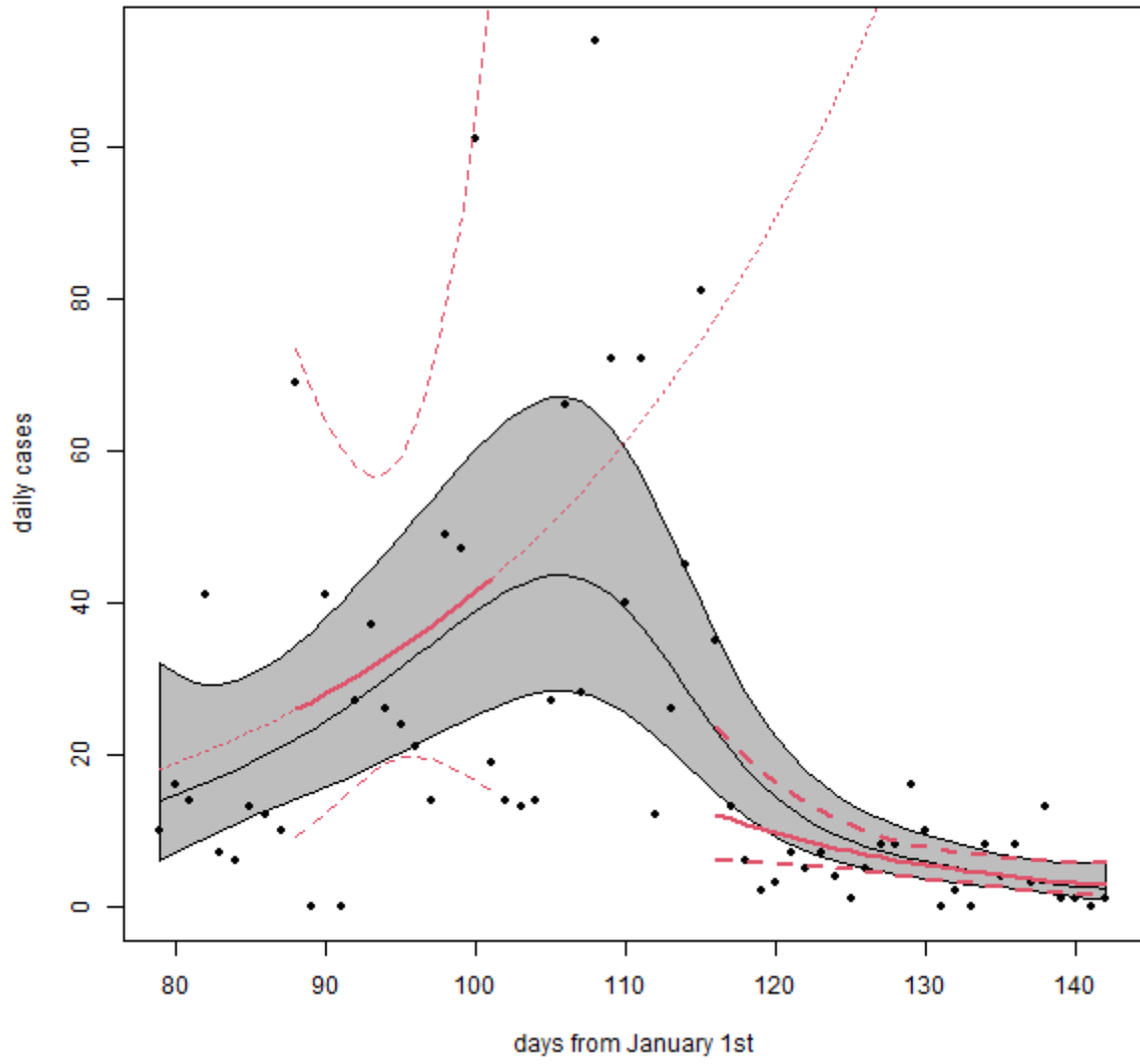
Somalia
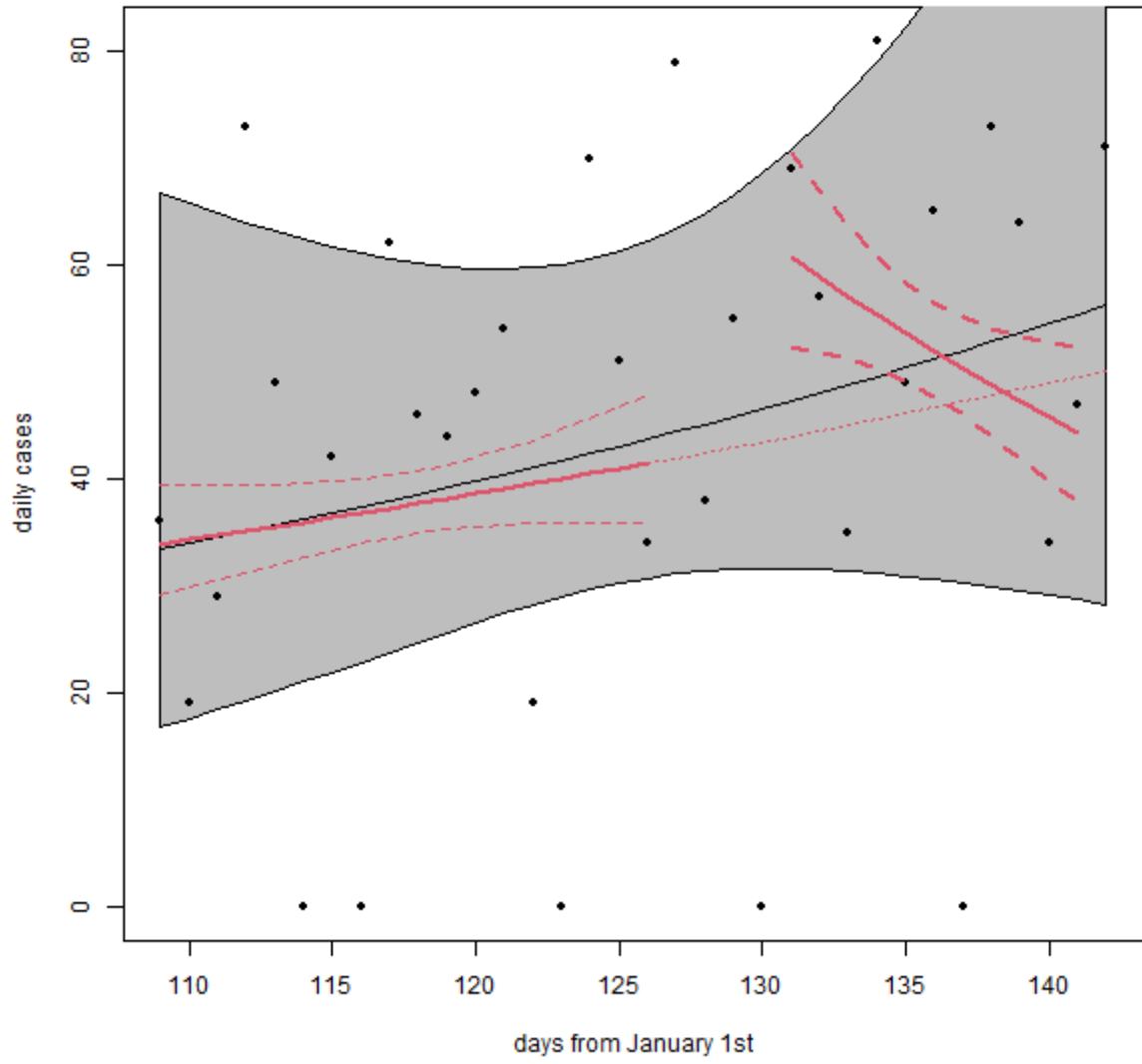
South_Africa

South_Africa

South_Korea

daily deaths

days from January 1st

Spain

Spain

daily deaths

days from January 1st

Sri_Lanka

Sudan

Sweden

daily cases

days from January 1st

Sweden

daily deaths

days from January 1st
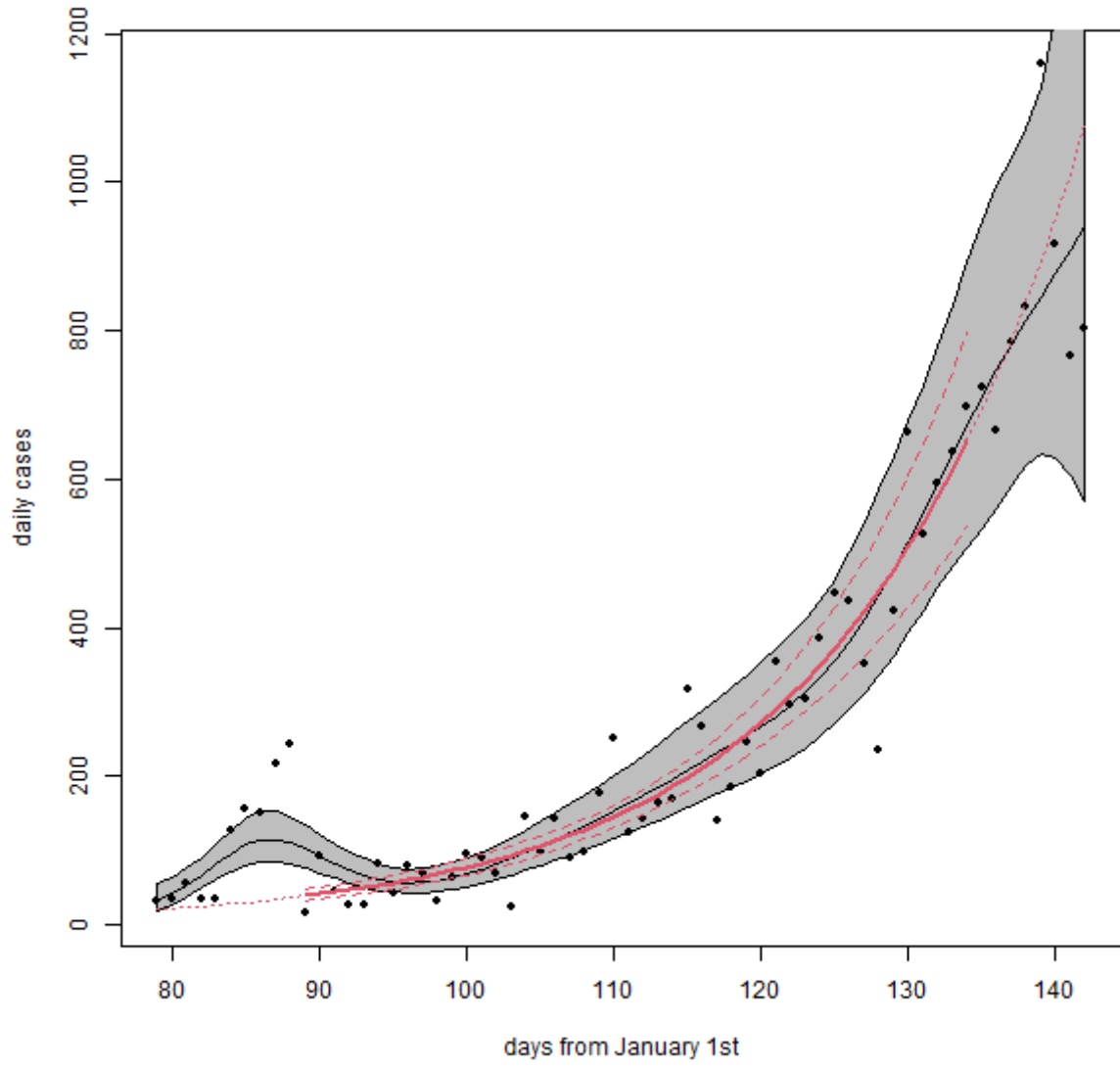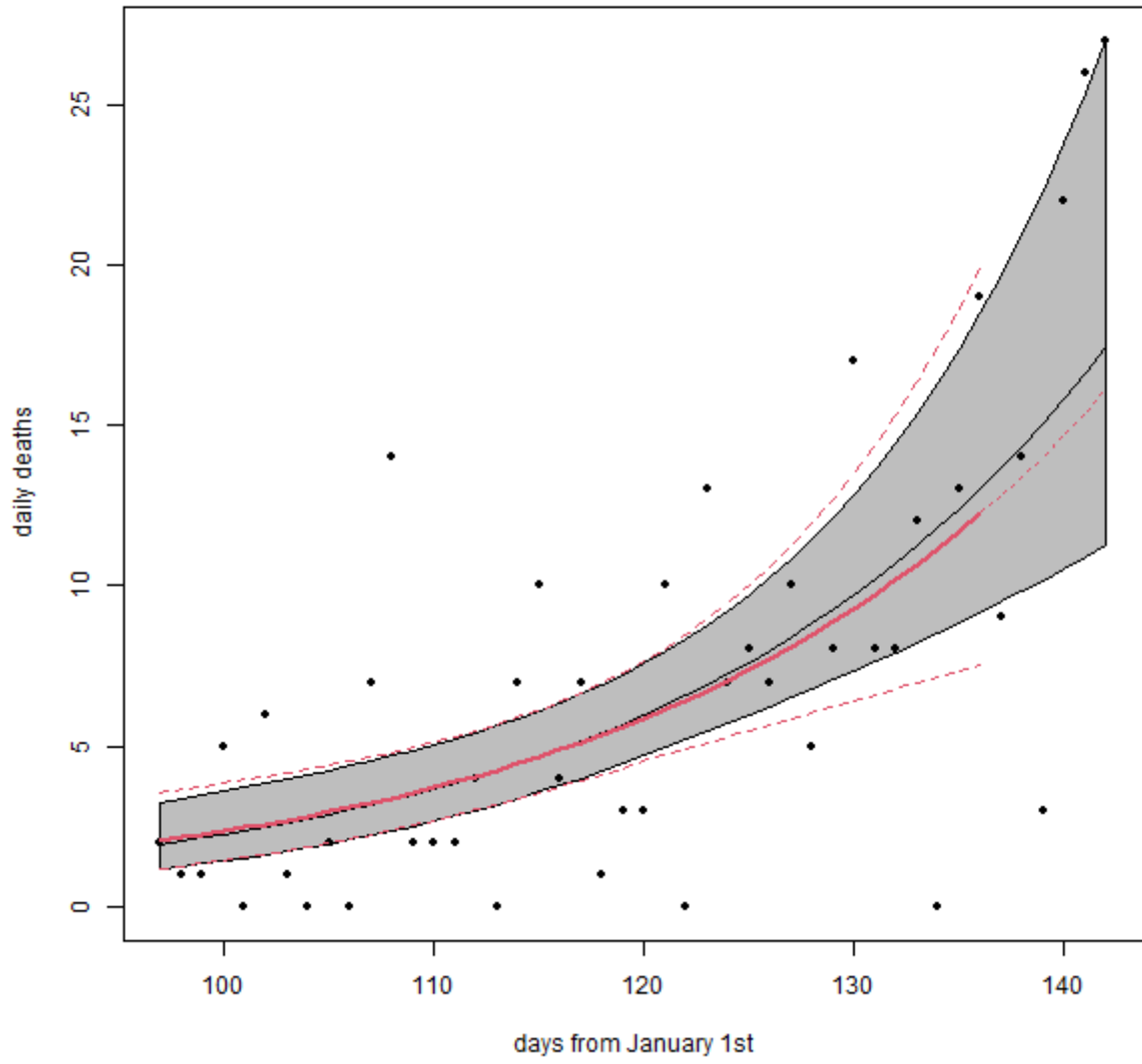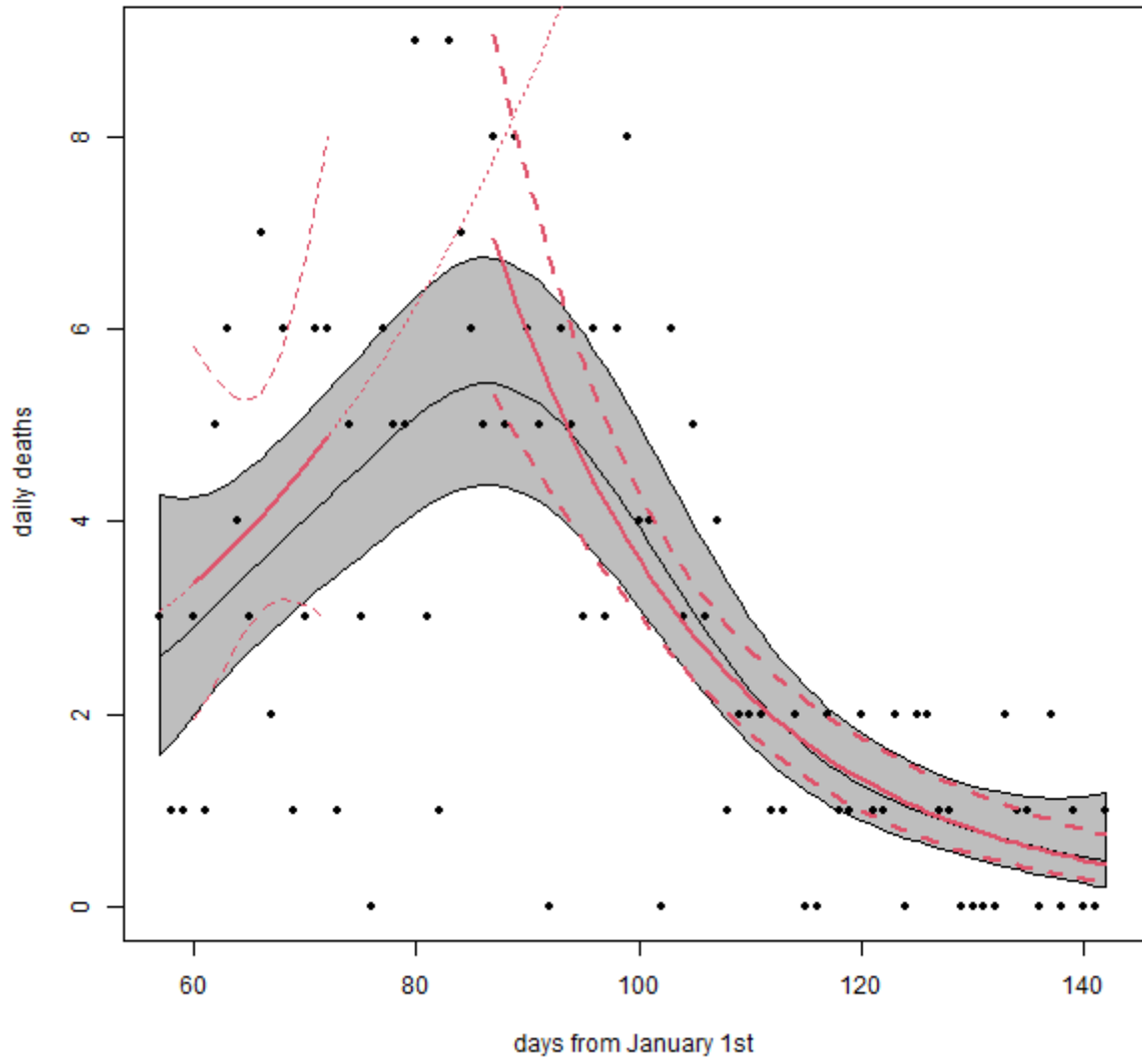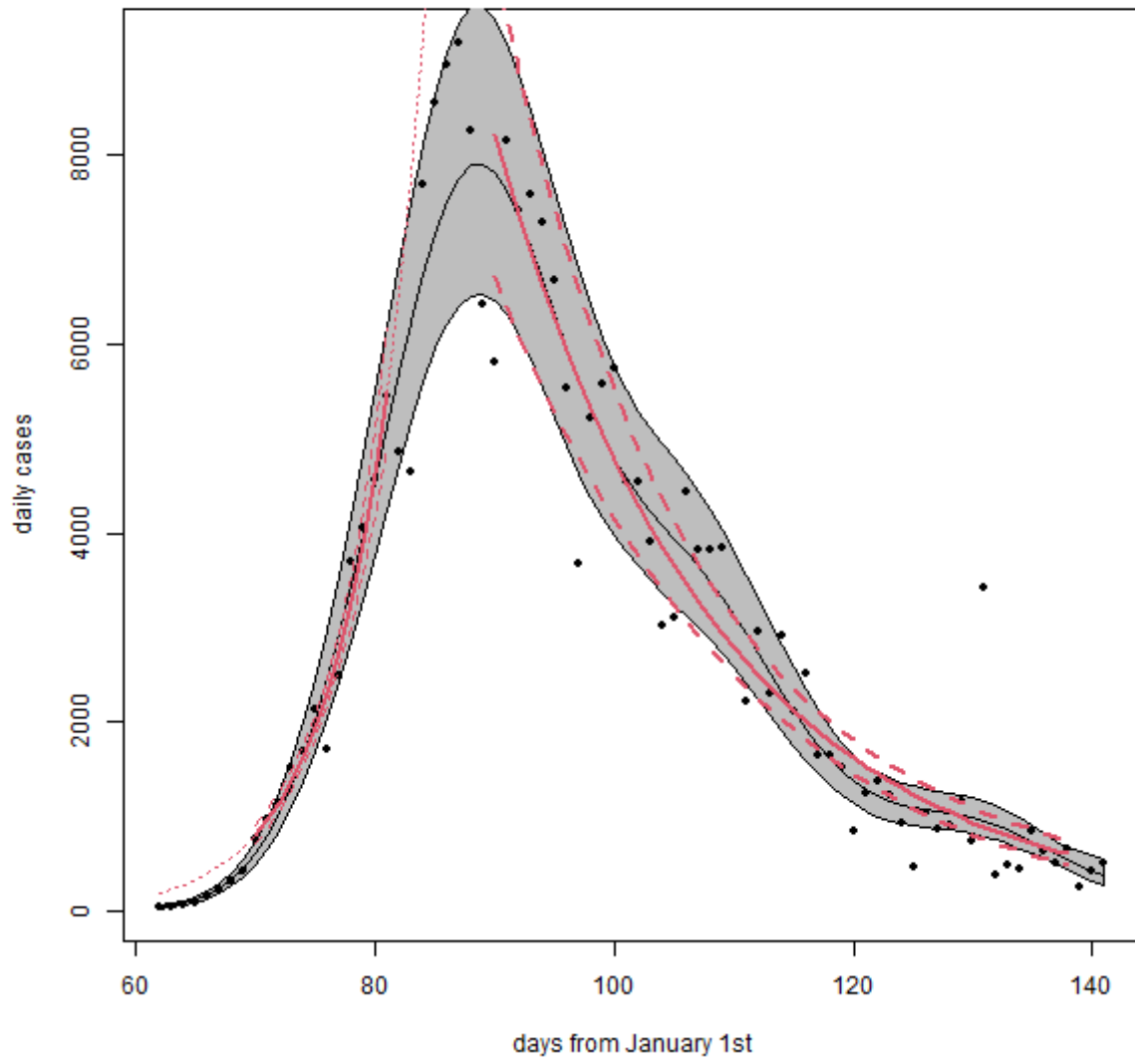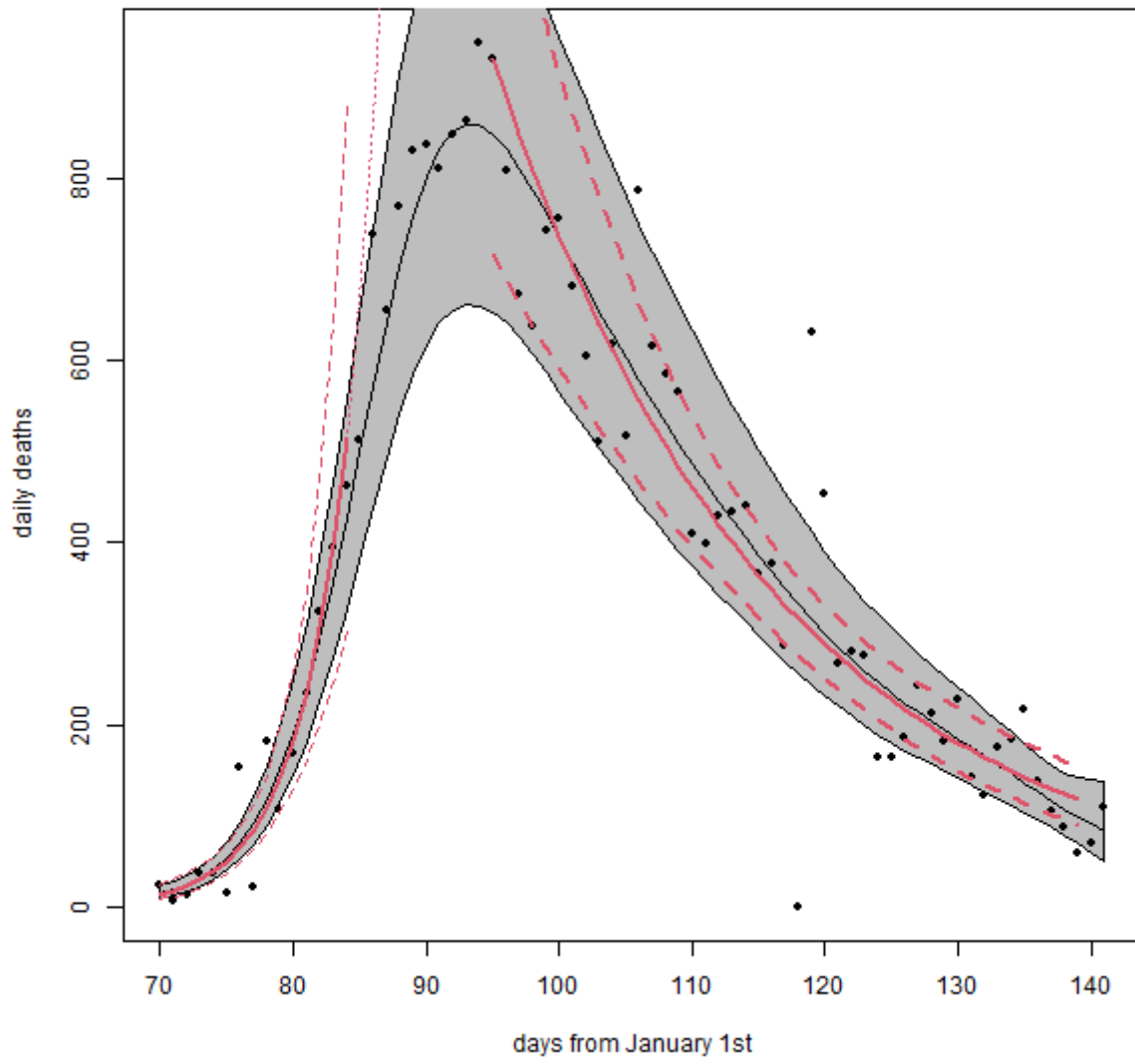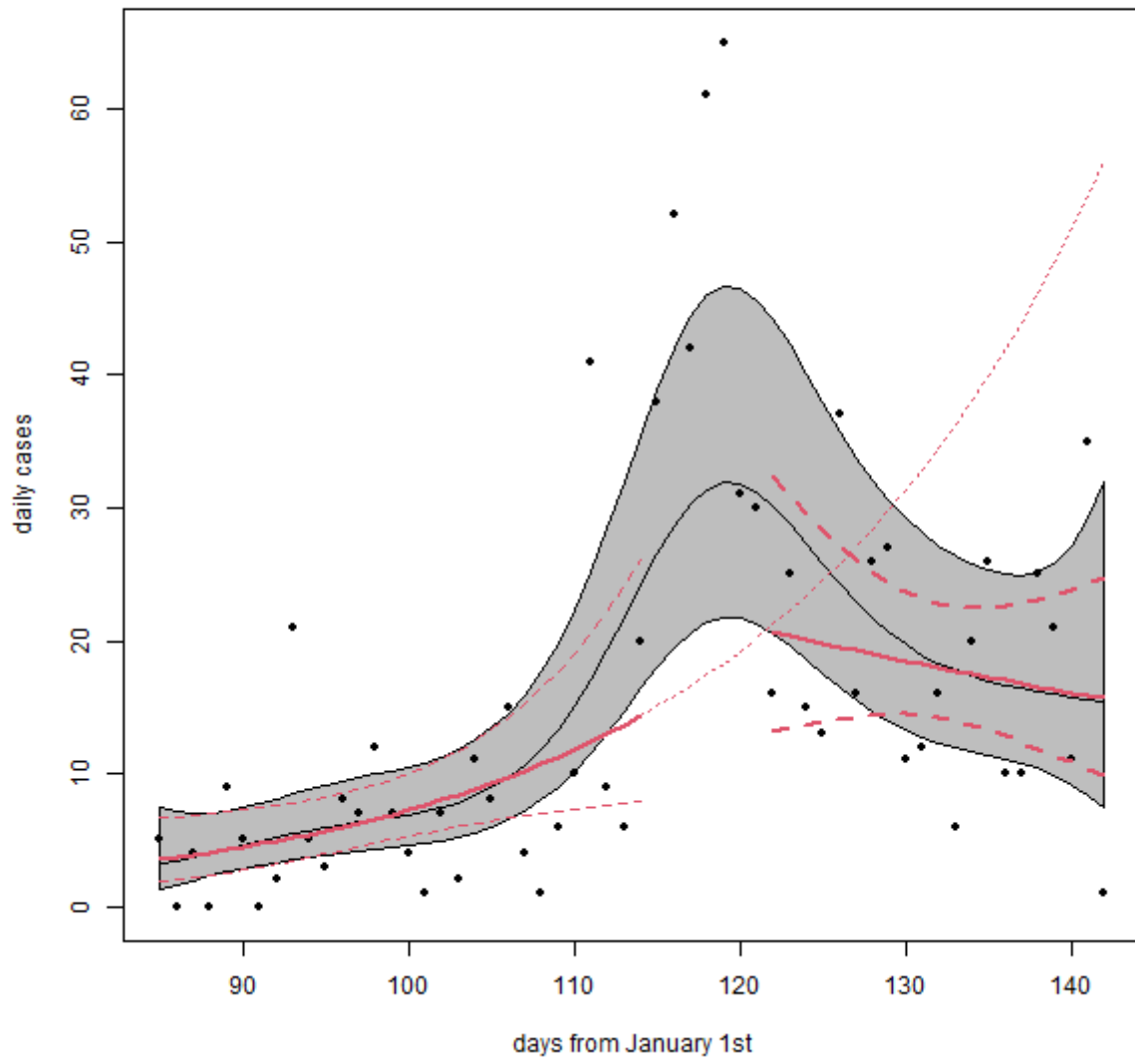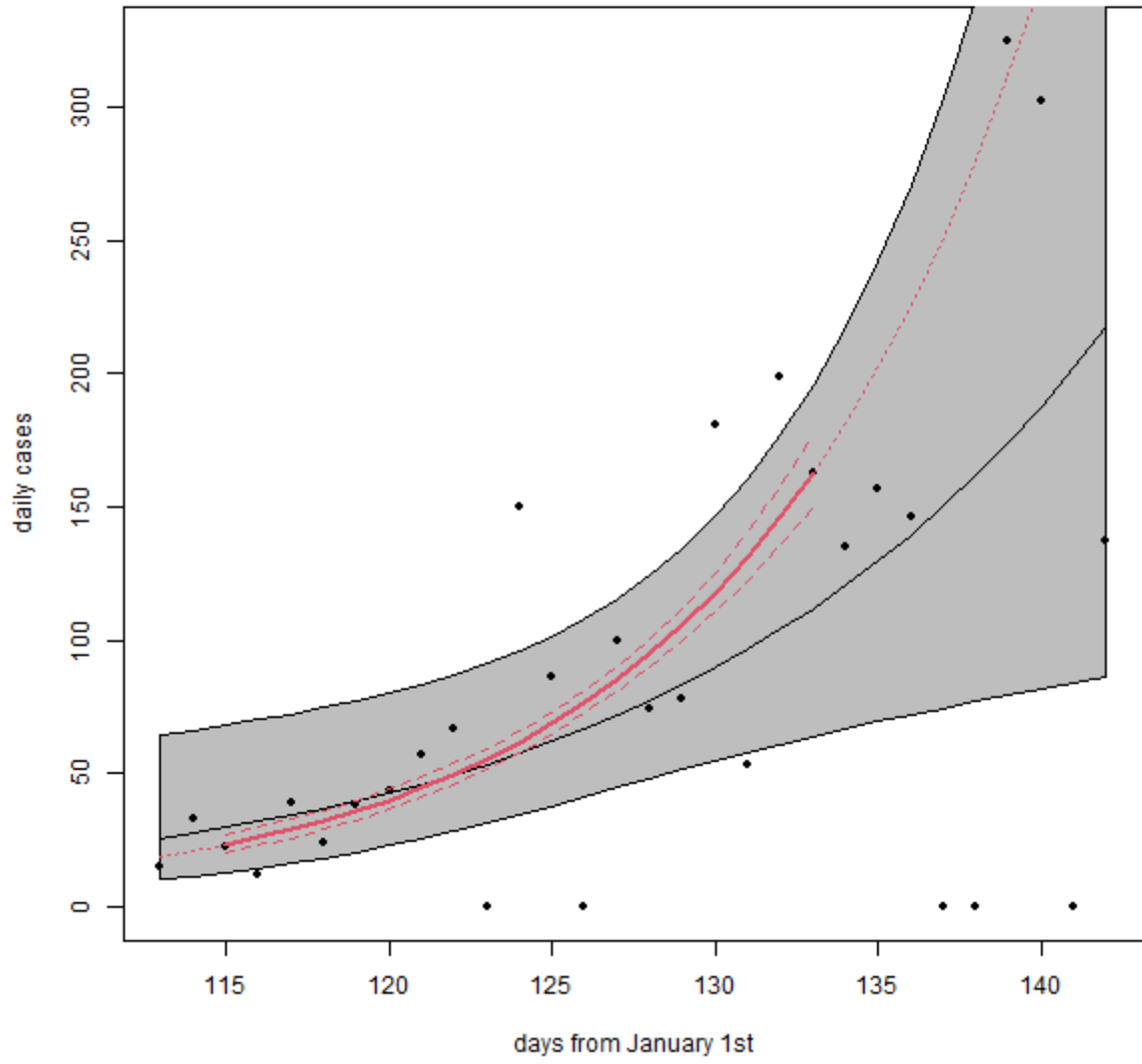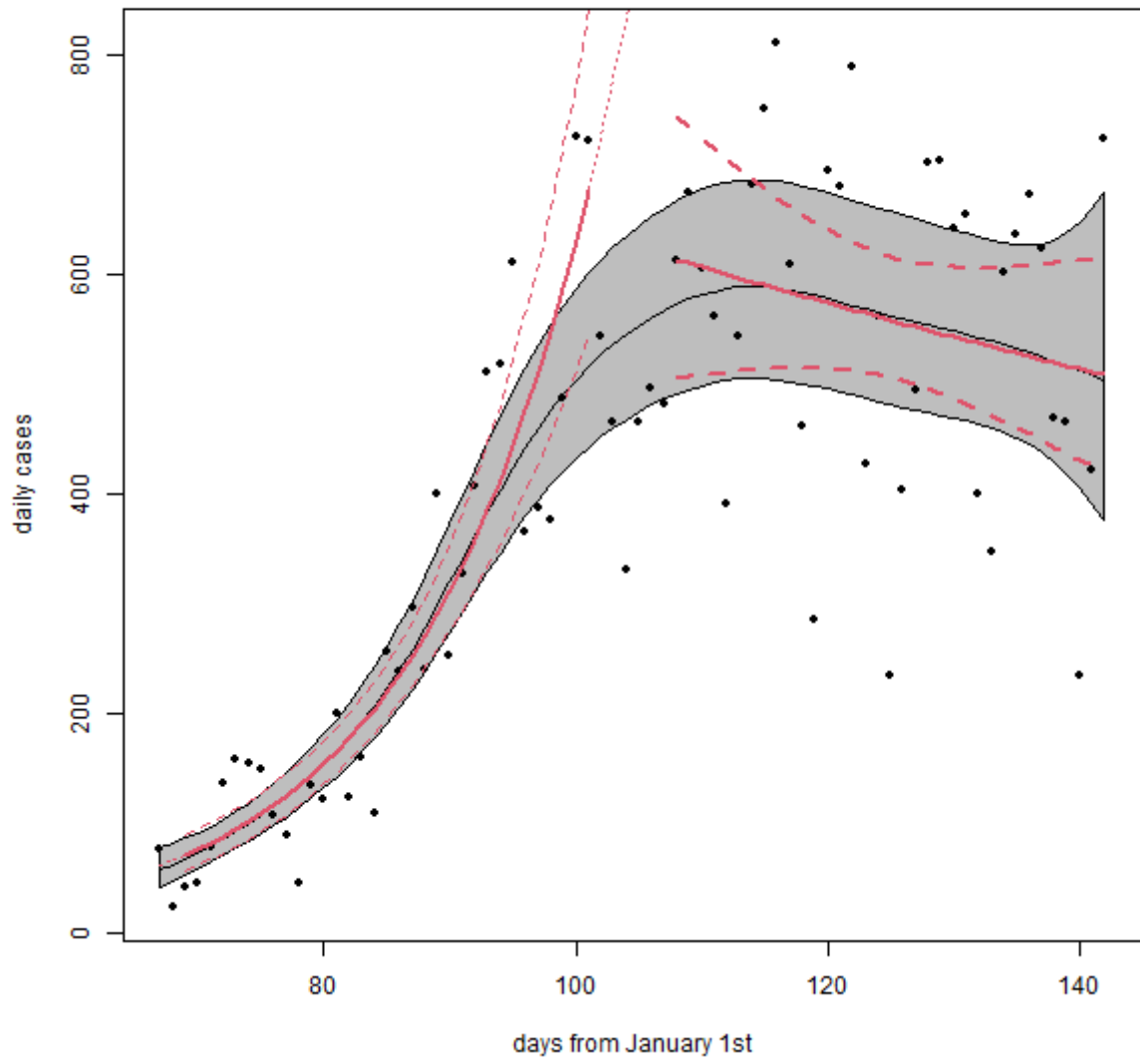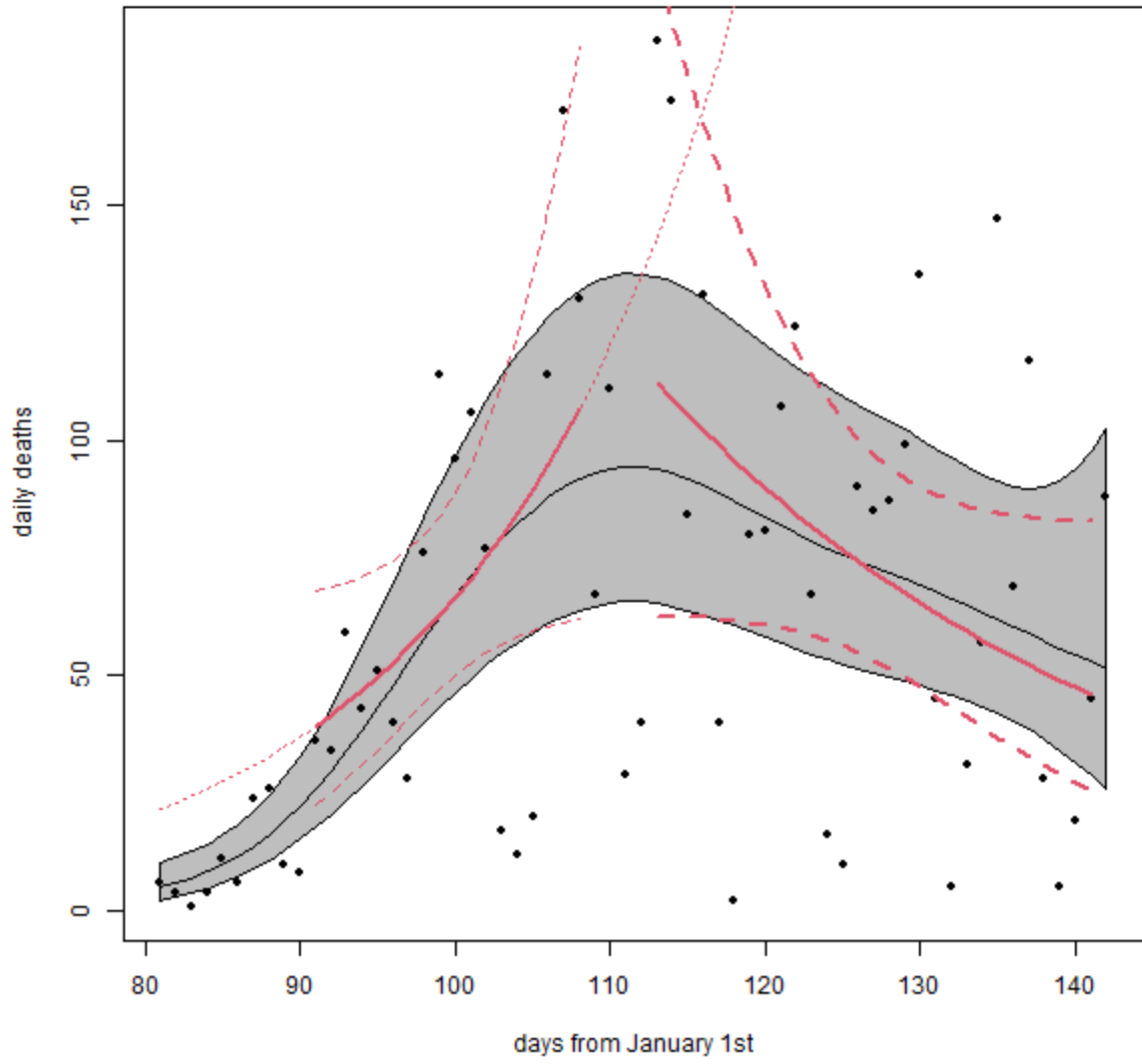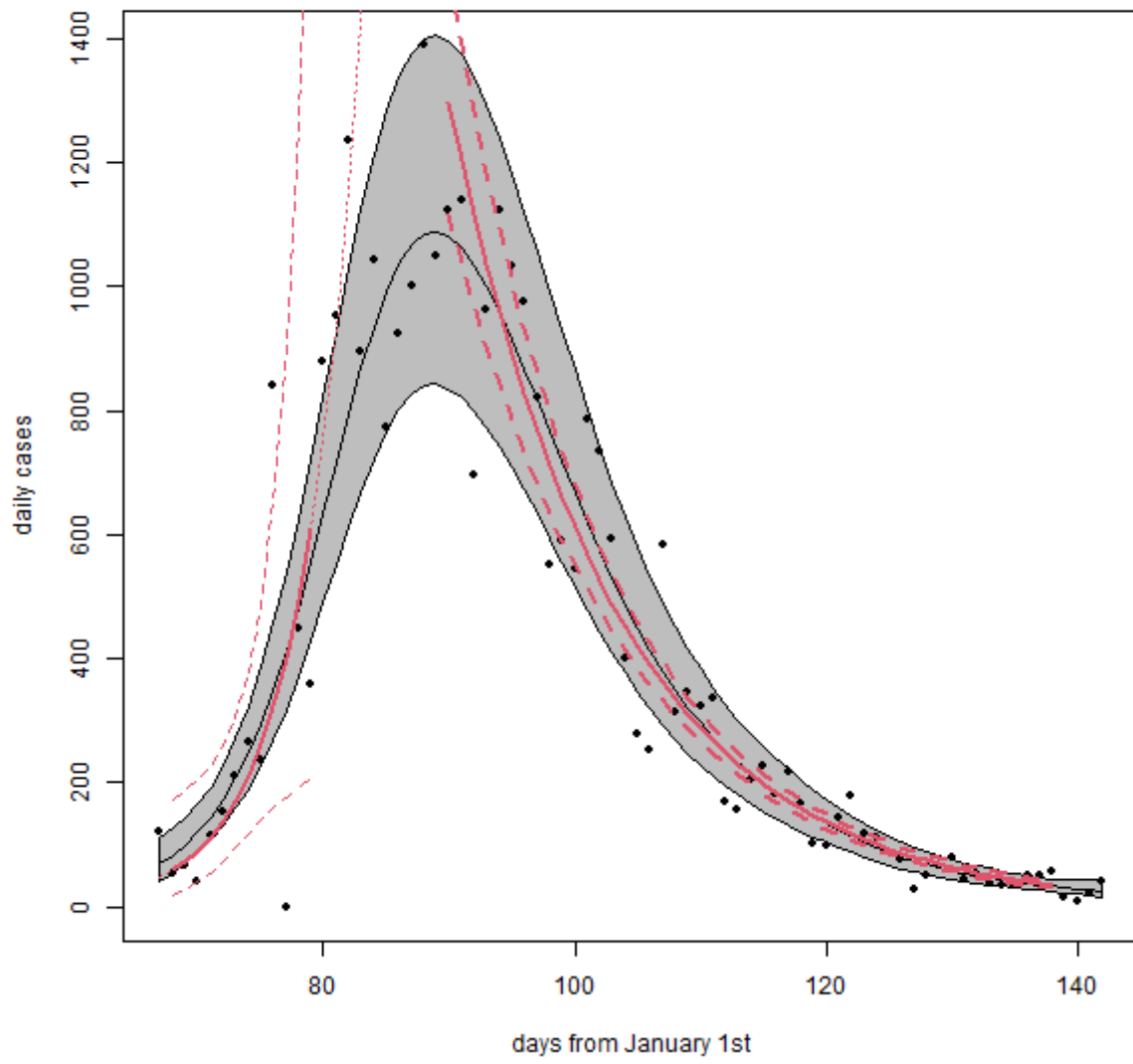
Switzerland

Switzerland

Turkey

Turkey

daily deaths

days from January 1st

UAE

daily cases

days from January 1st

UAE

UK

daily cases

days from January 1st

UK

days from January 1st

Ukraine

daily cases

days from January 1st

Ukraine

daily deaths

days from January 1st
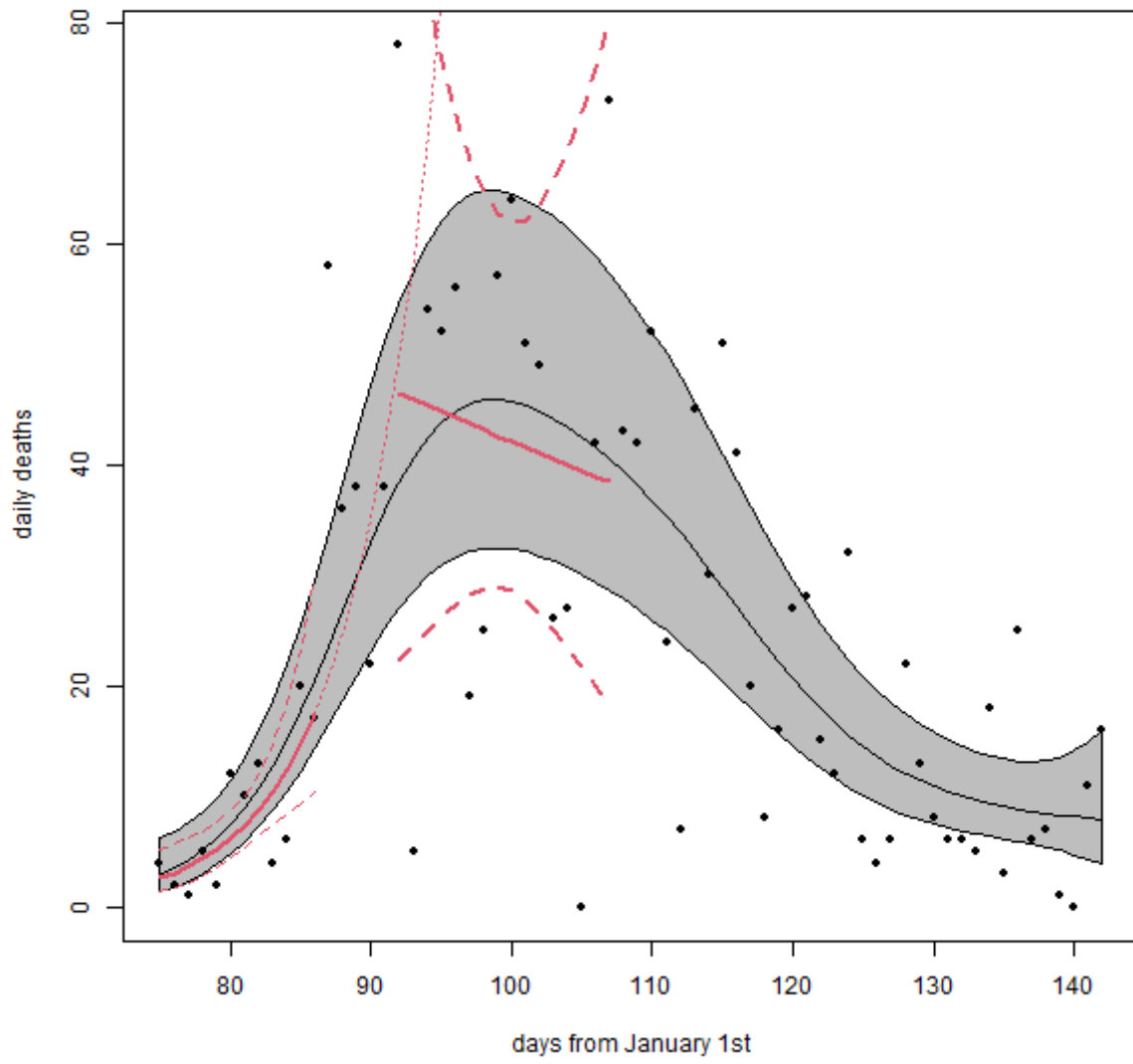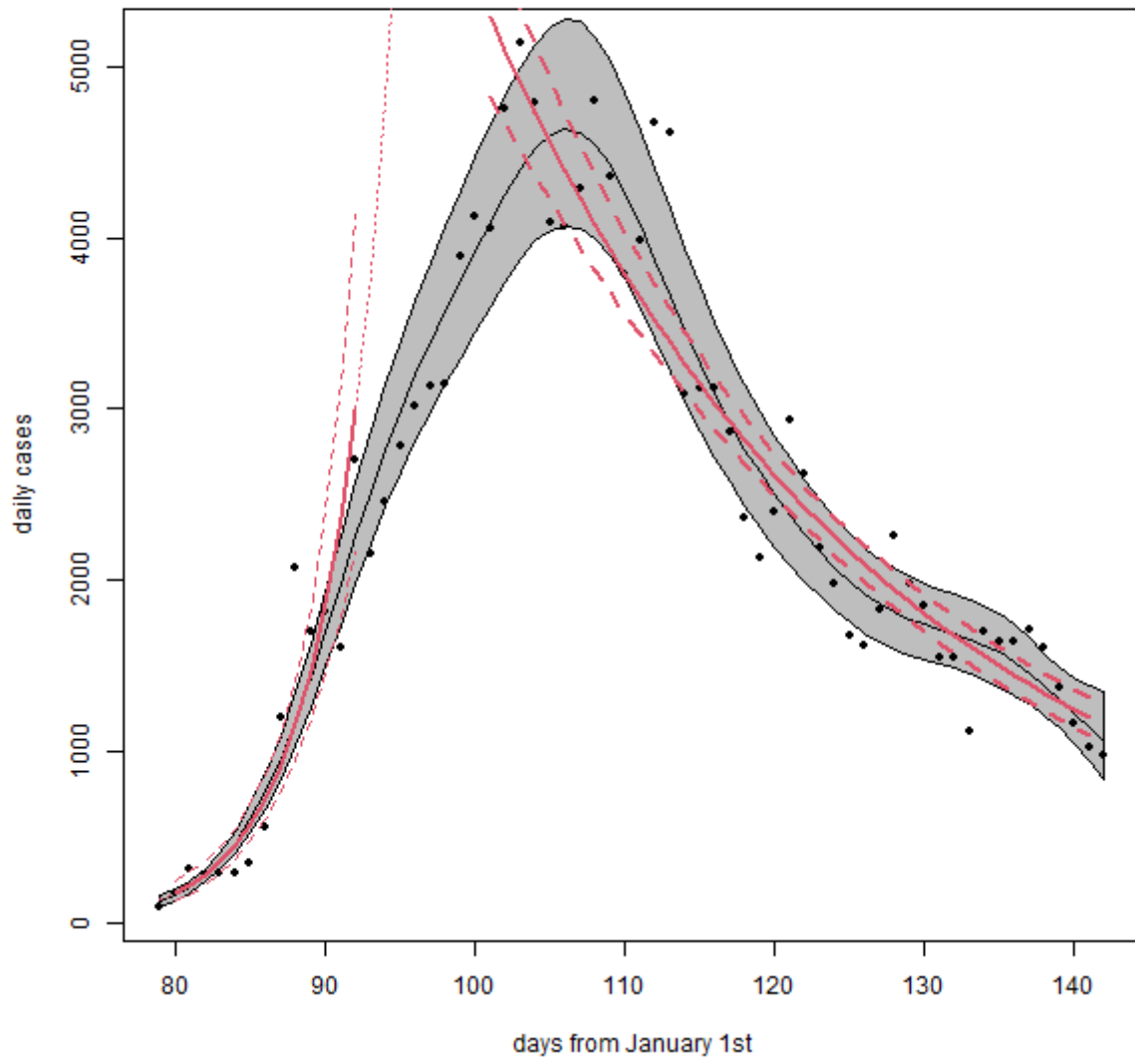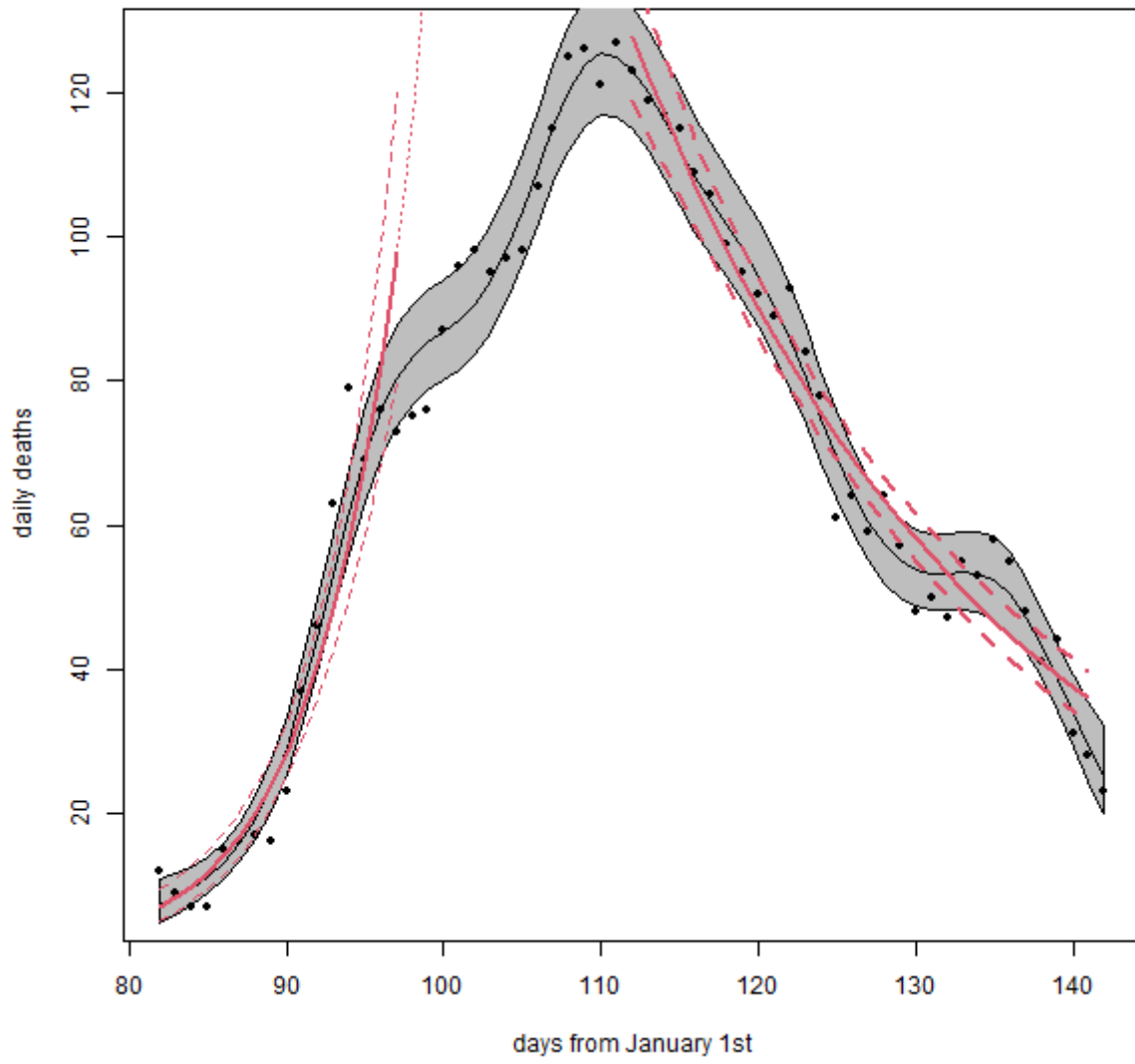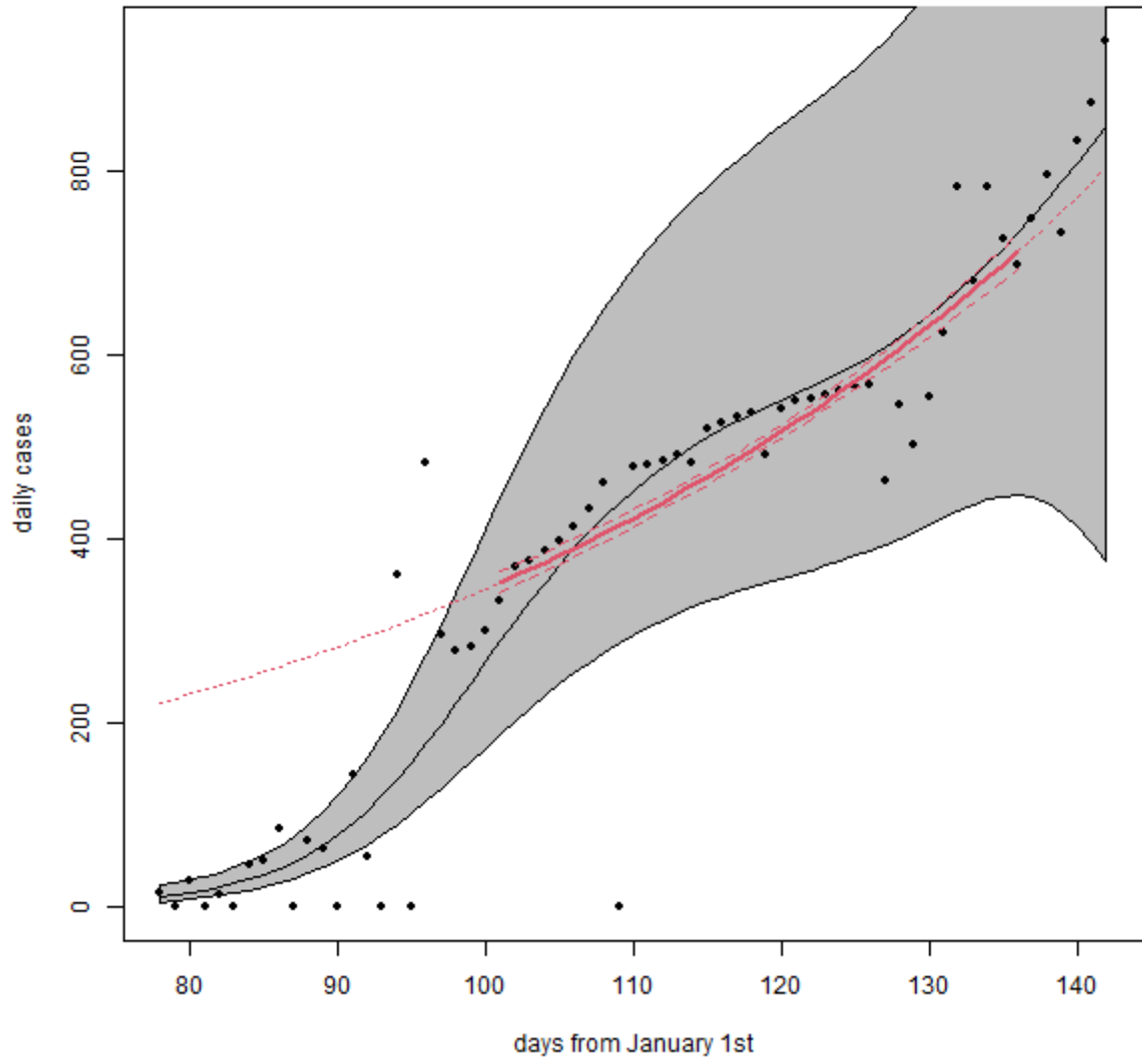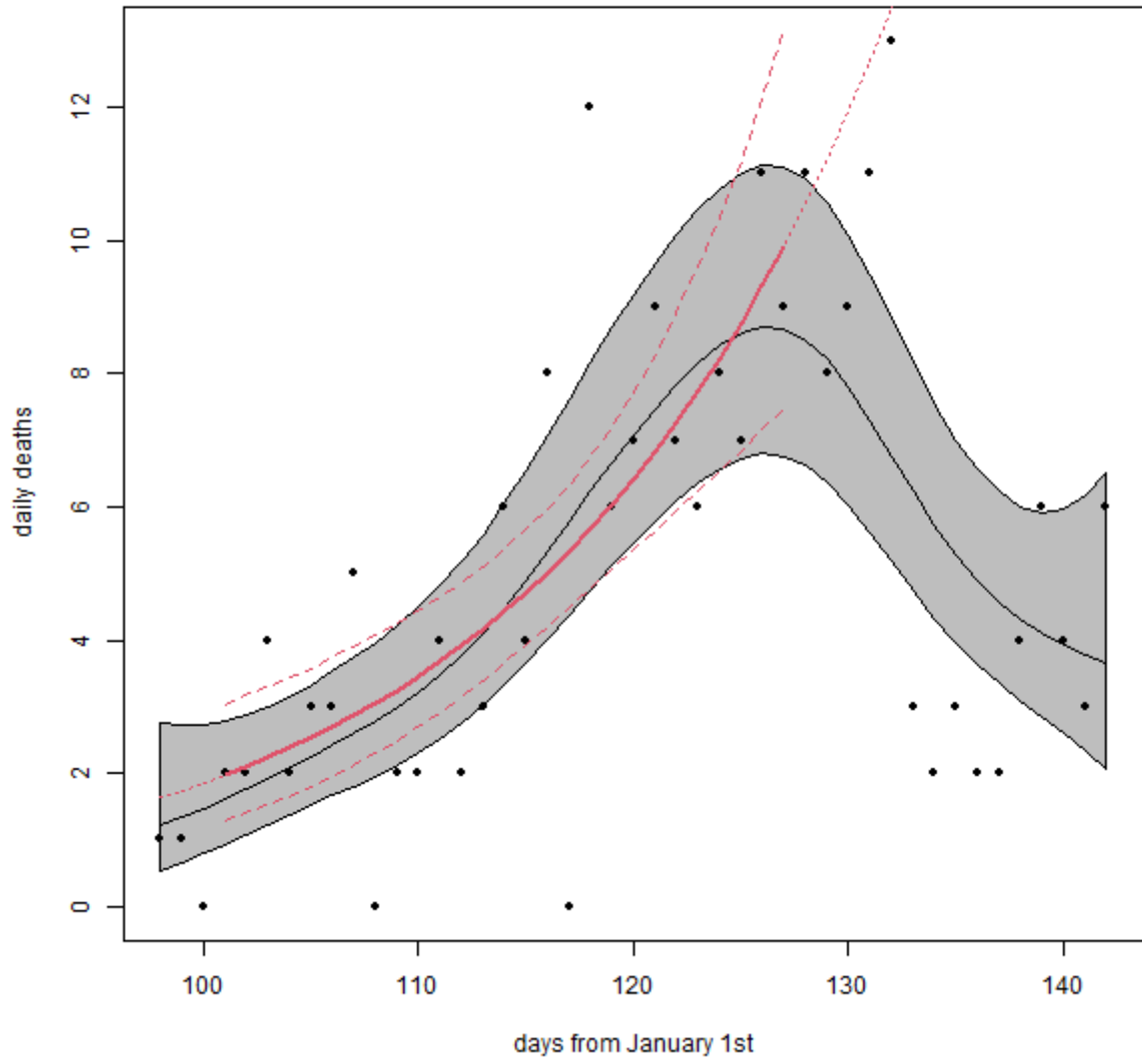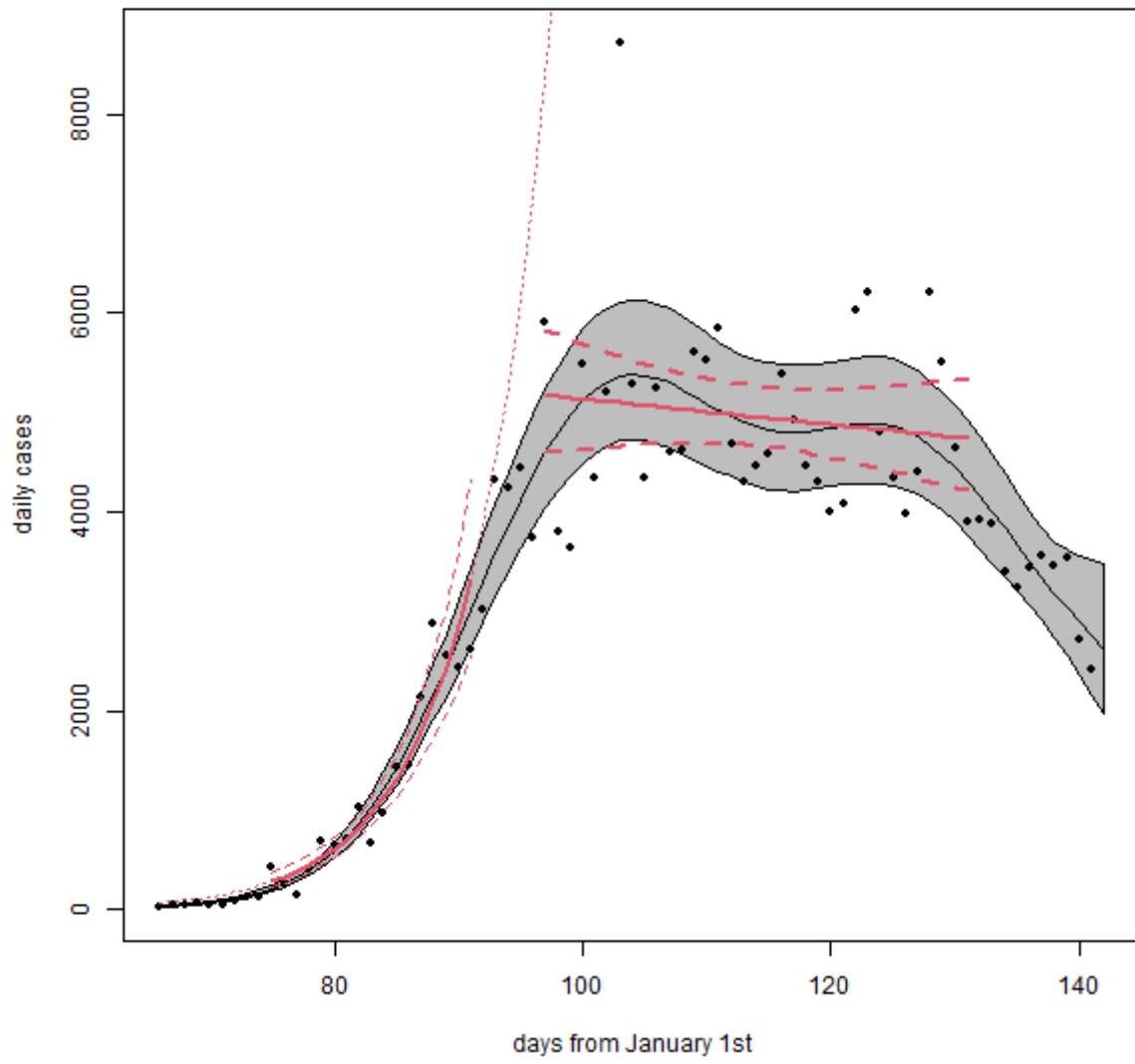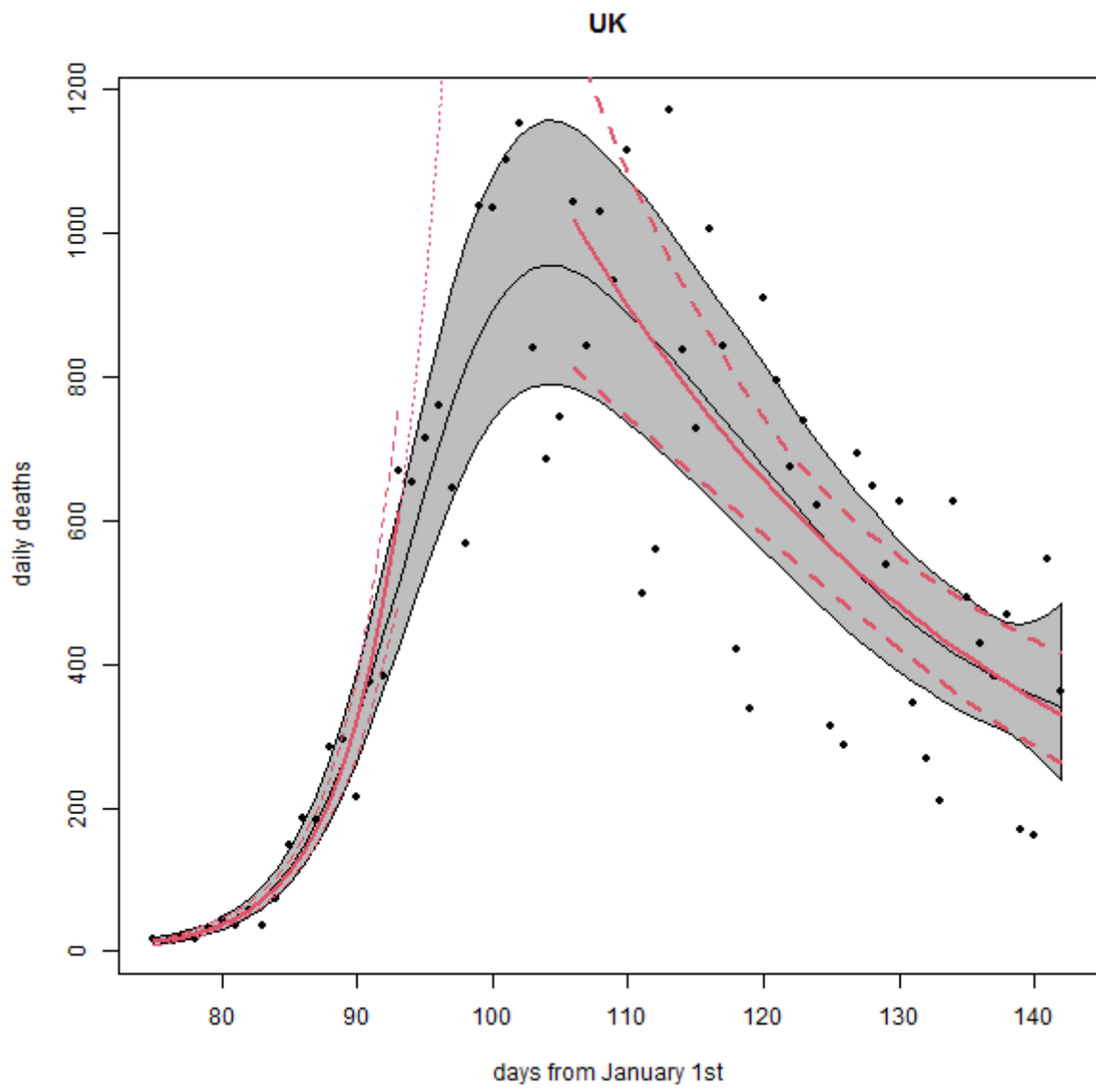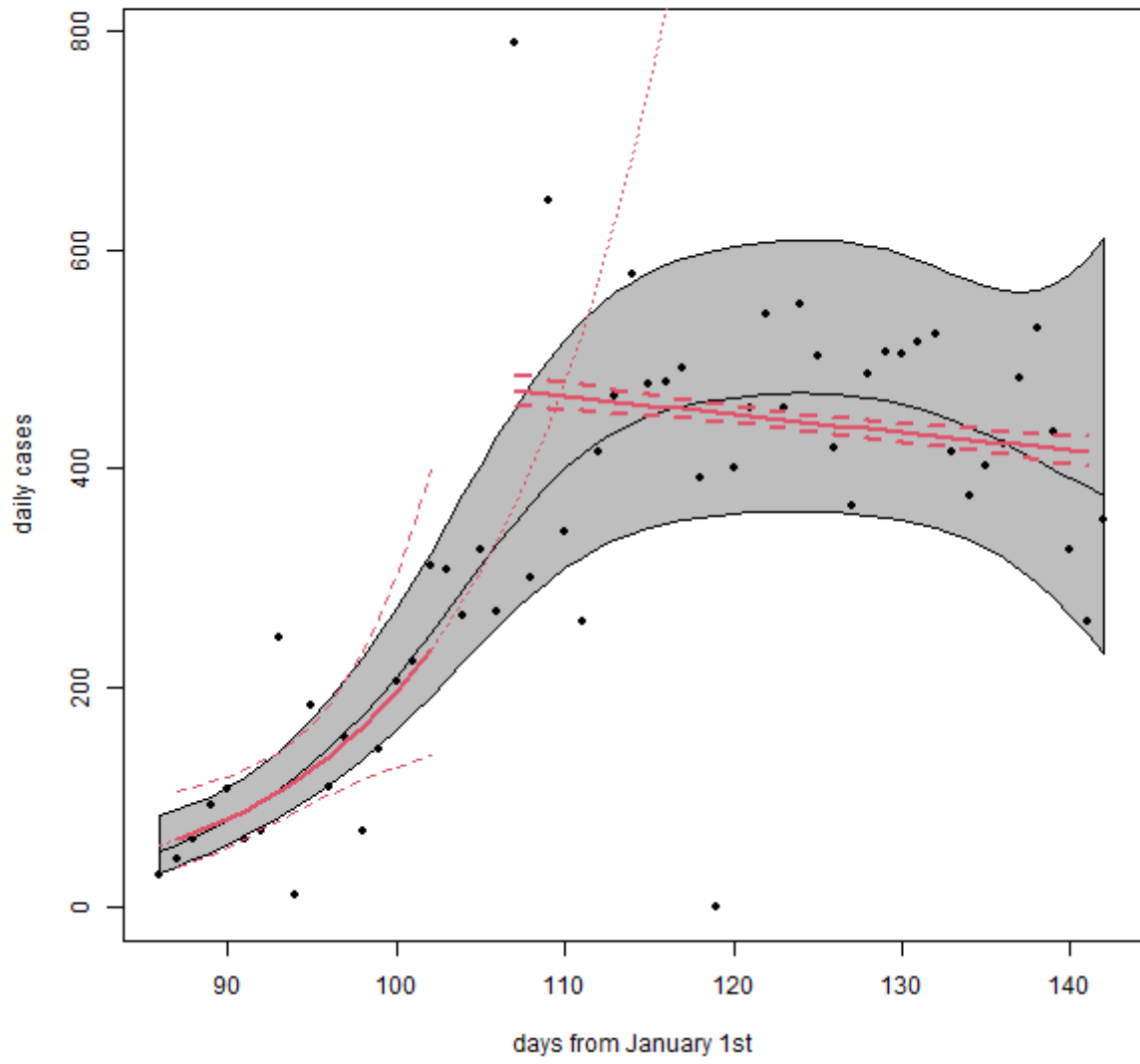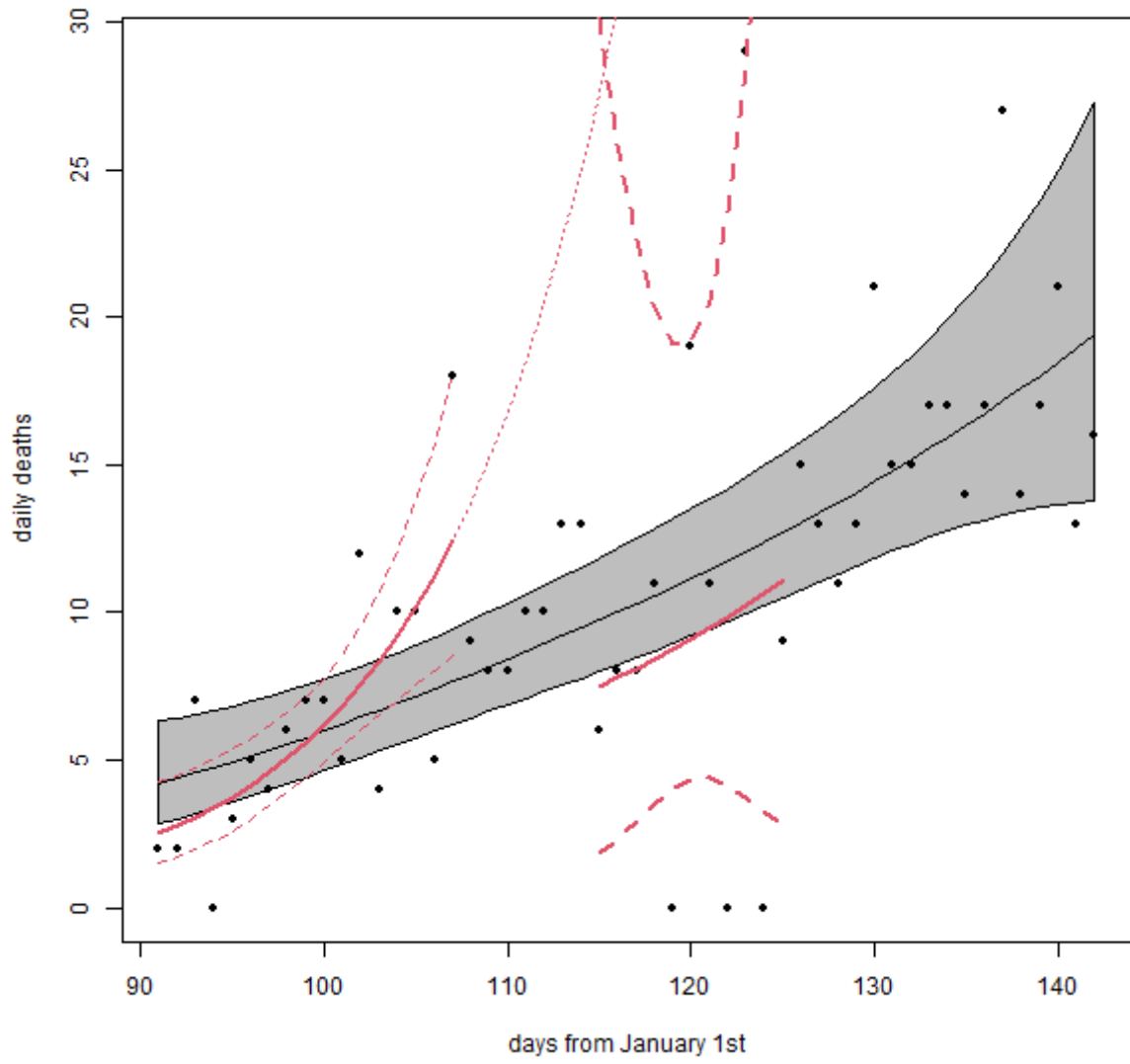
USA
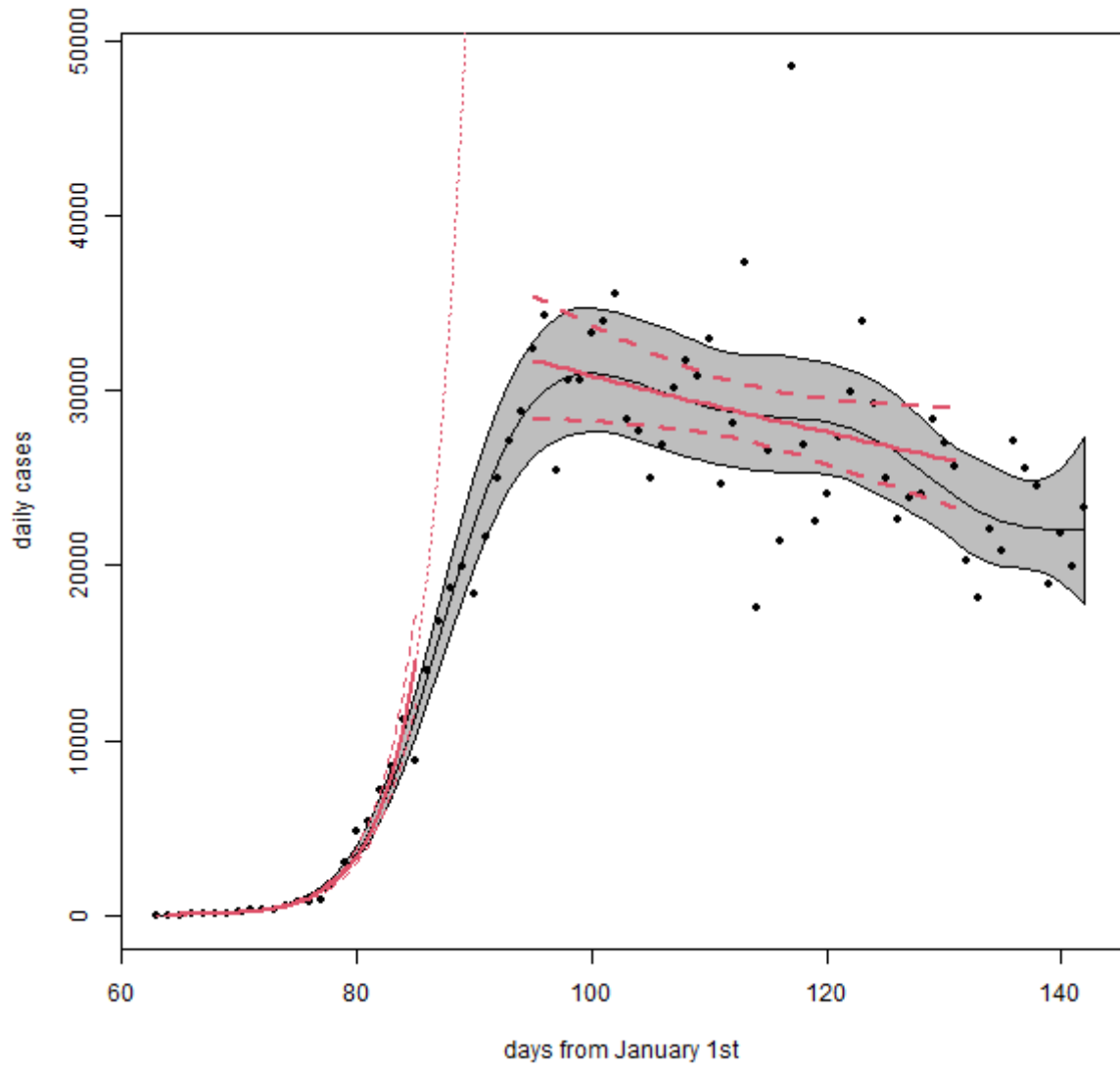
USA
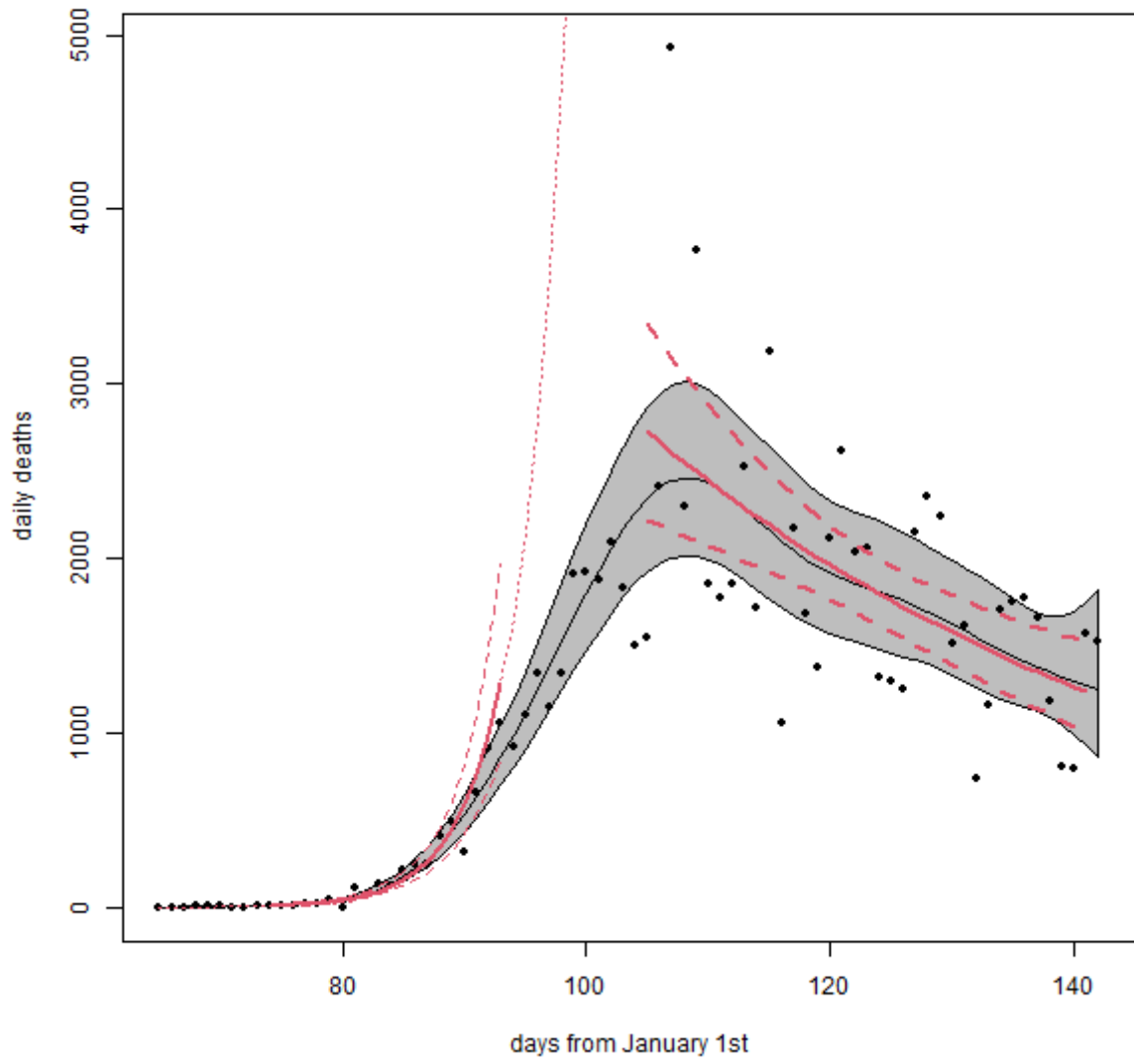
3) R code

```
# data from:
# https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-
distribution-covid-19-cases-worldwide

dat11<-read.csv("C:/mike/covid/COVID-19-geographic-disbtribution-worldwide.csv",as.is=TRUE)

# reverse so in order of increasing date
dat11<-dat11[nrow(dat11):1,]

# shorten country
names(dat11)<-ifelse(names(dat11)=="countriesAndTerritories","country",names(dat11))

# shorten some names
dat11$country[dat11$country=="United_States_of_America"]<-"USA"
dat11$country[dat11$country=="United_Kingdom"]<-"UK"
dat11$country[dat11$country=="Dominican_Republic"]<-"Dominican_R"
dat11$country[dat11$country=="United_Arab_Emirates"]<-"UAE"
dat11$country[dat11$country=="Bosnia_and_Herzegovina"]<-"Bosnia"
dat11$country[dat11$country=="North_Macedonia"]<-"N_Macedonia"
dat11$country[dat11$country=="Democratic_Republic_of_the_Congo"]<-"DR_Congo"

dat11$d2020<-dayssince1900(strptime2(dat11$dateRep))-120*365-28

# there is negative number for the cases on 10 days where data were adjusted
dat11[is.na(dat11$d2020) | dat11$cases<0 | dat11$deaths<0,]
# just set those to na
#dat11<-dat11[!is.na(dat11$d2020) & dat11$cases>=0 & !is.na(dat11$deaths) & dat11$deaths>=0,]
dat11$cases[dat11$cases<0]<-NA_real_


# and in Iran the data for the 4th April has been put on the 5th; share those out (it is an
odd number)
dat11[dat11$country=="Iran" & dat11$dateRep=="04/04/2020",c("cases","deaths")]<-
        floor(dat11[dat11$country=="Iran" & dat11$dateRep=="05/04/2020",c("cases","deaths")]/2)
dat11[dat11$country=="Iran" & dat11$dateRep=="05/04/2020",c("cases","deaths")]<-
        ceiling(dat11[dat11$country=="Iran" &
dat11$dateRep=="05/04/2020",c("cases","deaths")]/2)

# in China a huge pile got added on the 17/04/2020; discard that day
dat11<- dat11[dat11$country!="China" | dat11$dateRep!="17/04/2020",]


# add running totals for each country
dat11$cumdeaths<-0
dat11$cumcases<-0

# need utility so cumsum copes with NAs
cumsum2<-function(x) cumsum(ifelse(is.na(x),0,x))

for(ct in unique(dat11$country))
{
   dat11$cumcases[dat11$country==ct]<-cumsum2(dat11$cases[dat11$country==ct])
   dat11$cumdeaths[dat11$country==ct]<-cumsum2(dat11$deaths[dat11$country==ct])
}


# make table of slopes

slopetab<-data.frame(country=sort(unique(dat11$country)),cases=NA_real_,deaths=NA_real_)

for(i in 1:nrow(slopetab))
{  datc<-dat11[dat11$country==slopetab$country[i],]
   slopetab$cases[i]<-sum(datc$cases,na.rm=TRUE)
   slopetab$deaths[i]<-sum(datc$deaths,na.rm=TRUE)
}

slopetab<-slopetab[order(-slopetab$deaths),]

# quick squint at countries

slopetabc<-slopetab[order(-slopetab$cases),][1:16,]
par(mfrow=c(4,4))
```

```
for(i in 1:nrow(slopetabc))
{  if(!is.na(slopetabc$cases[i]) )
    {
        datc<-dat11[dat11$country==slopetabc$country[i],]

        plot(datc$d2020, datc$cases,pch=20,type="b",
                    xlab="days from Jan 1st",
                    ylab="cases",main=paste(slopetabc$country[i],sum(datc$cases)))
    }
}
slopetabc


par(mfrow=c(4,4))
for(i in 1:16)
{  if(!is.na(slopetab$deaths[i]) )
    {
        datc<-dat11[dat11$country==slopetab$country[i],]

        plot(datc$d2020,
                datc$deaths,pch=20,type="b",
                    xlab="days from Jan 1st",
                    ylab="deaths",main=paste(slopetab$country[i],sum(datc$deaths)))
    }
}
slopetab[1:30,]


#####################################

# estimate exponential rates
# do for each possible start and end point that gives at least 5 datapoints
# don't include days before a total of 10 deaths/ 100 cases had been recorded
# only do for countries with 100 deaths, or 1000 cases, spread over at least 6 days
# neg binomial glms - use +-2se in 95%CI for simplicity

# store results in array: first two dimensions start and end times only fill above diagonal
# 3rd dimension: slope then slope se, then dAIC for quadratic, then dBIC for quadratic, then
theta
# mark date of peak with -1 in the diagonal of the theta layer so can use it to
# prevent overlapping of the first and second intervals

slopearray<-array(NA_real_,c(length(unique(dat11$d2020)),length(unique(dat11$d2020)),5))
slopelistd<-list()

library(MASS)
library(MuMIn)
library(mgcv)

# deaths
for(i in 1:sum(slopetab$deaths>100))
{
    slopelistd[[i]]<-slopearray
    names(slopelistd)[i]<-as.character(slopetab$country[slopetab$deaths>100][i])

    datc<-dat11[dat11$country==names(slopelistd)[i] & dat11$cumdeaths>10,]

  # date of peak
    d2020ofmax<-datc$d2020[which.max(datc$deaths)]
    slopelistd[[i]][d2020ofmax+1,d2020ofmax+1,5]<--1

    if(nrow(datc)>6)
    {
        for(j in min(datc$d2020):(max(datc$d2020)-4))
        {  for(k in (j+4):max(datc$d2020))
            {
                g1<-try(glm.nb(deaths~d2020,data=datc[datc$d2020>=j & datc$d2020<=k,]),TRUE)
                g2<-try(glm.nb(deaths~d2020+I(d2020^2),data=datc[datc$d2020>=j &
datc$d2020<=k,]),TRUE)
                if("glm" %in% class(g1) && dim(summary(g1)$coef)[1]==2 )
                { slopelistd[[i]][j+1,k+1,1]<-g1$coef[2]
                    slopelistd[[i]][j+1,k+1,2]<-summary(g1)$coef[2,2]
                    if("glm" %in% class(g2))
                    { slopelistd[[i]][j+1,k+1,3]<-AIC(g1)-AIC(g2)
                        slopelistd[[i]][j+1,k+1,4]<-BIC(g1)-BIC(g2)
```

```
                }
                slopelistd[[i]][j+1,k+1,5]<-g1$theta
            } }
      } }
 print(paste(i,date()))
}

# cases
slopelistc<-list()
for(i in 1:sum(slopetab$cases>1000))
{
  slopelistc[[i]]<-slopearray
   names(slopelistc)[i]<-as.character(slopetab$country[slopetab$cases>1000][i])

   datc<-dat11[dat11$country==names(slopelistc)[i] & dat11$cumcases>100,]

  # date of peak
   d2020ofmax<-datc$d2020[which.max(datc$cases)]
   slopelistc[[i]][d2020ofmax+1,d2020ofmax+1,5]<--1

   if(nrow(datc)>6)
   {
      for(j in min(datc$d2020):(max(datc$d2020)-4))
       { for(k in (j+4):max(datc$d2020))
          {
             g1<-try(glm.nb(cases~d2020,data=datc[datc$d2020>=j & datc$d2020<=k,]),TRUE)
             g2<-try(glm.nb(cases~d2020+I(d2020^2),data=datc[datc$d2020>=j &
datc$d2020<=k,]),TRUE)
             if("glm" %in% class(g1) && dim(summary(g1)$coef)[1]==2 )
             { slopelistc[[i]][j+1,k+1,1]<-g1$coef[2]
               slopelistc[[i]][j+1,k+1,2]<-summary(g1)$coef[2,2]
               if("glm" %in% class(g2))
               { slopelistc[[i]][j+1,k+1,3]<-AIC(g1)-AIC(g2)
                 slopelistc[[i]][j+1,k+1,4]<-BIC(g1)-BIC(g2)
               }
               slopelistc[[i]][j+1,k+1,5]<-g1$theta
            } }
      } }
 print(paste(i,date()))
}




# function to find which one has positive slope
# and which subsequent one has lower slope
# where BIC for linear "most better" than BIC quadratic

minse<-function(sa,daic=0,gap=5,minl=10,bp=5)
{
  # output a matrix of start and end positions with biggest BIC difference
  # first row positive slope, second row negative
   res<-matrix(NA_real_,2,2)

  # find date of peak
    dpeak<-which(sa[,,5]==-1,TRUE)[1]

  # filter if less than minl days long
    for(i in 1:(dim(sa)[1]))
       sa[i,i:min(i+minl-1,dim(sa)[1]),]<-NA_real_

  # filter for bic daic better for linear
   sa[,,1]<-ifelse(sa[,,4]<daic,sa[,,1],NA_real_)
   sa[,,4]<-ifelse(sa[,,4]<daic,sa[,,4],NA_real_)

  # positives
   sap<-sa
   sap[,,4]<-ifelse(sa[,,1]>0,sa[,,4],NA_real_)
  # filter out any that end after bp days before date of peak
   sap[,-(1:(dpeak-bp)),]<-NA_real_

  #best bic difference
   if(mean(is.na(sap[,,4]))<1)
      res[1,]<-which(sap[,,4]==min(sap[,,4],na.rm=TRUE),TRUE)[1,]
```

```r
   # was negative, now just require smaller slope for second group - TAKEN THIS OUT
 #  sa[,,4]<-ifelse(sa[,,1]<sa[res[1,1],res[1,2],1],sa[,,4],NA_real_)
   # filter out any that start less than gap after the end of the first exponential growth
   if(!is.na(res[1,2]))
       sa[1:min((res[1,2]-1+gap),dim(sa)[1]),,]<-NA_real_


   if(mean(is.na(sa[,,4]))<1)
   {   second<-which(sa[,,4]==min(sa[,,4],na.rm=TRUE),TRUE)
       res[2,]<-second[nrow(second),]
   }

   return(res)
}

minse(slopelistd[[2]],0)

##########################

# fill table of slopes

slopetab$dbfirstday<-NA_real_
slopetab$dblastday<-NA_real_
slopetab$dbslope<-NA_real_
slopetab$dbse<-NA_real_
slopetab$dafirstday<-NA_real_
slopetab$dalastday<-NA_real_
slopetab$daslope<-NA_real_
slopetab$dase<-NA_real_
slopetab$cbfirstday<-NA_real_
slopetab$cblastday<-NA_real_
slopetab$cbslope<-NA_real_
slopetab$cbse<-NA_real_
slopetab$cafirstday<-NA_real_
slopetab$calastday<-NA_real_
slopetab$caslope<-NA_real_
slopetab$case<-NA_real_

for(i in 1:nrow(slopetab))
{
   if(slopetab[i,1] %in% names(slopelistd))
   {   eval(parse(text=paste("theseslopes<-slopelistd$",slopetab[i,1],sep="")))
       thesebounds<-minse(theseslopes,0)
       slopetab$dbfirstday[i]<-thesebounds[1,1]-1  # need to subtract 1 to get back to d2020
       slopetab$dblastday[i]<-thesebounds[1,2]-1
       slopetab$dafirstday[i]<-thesebounds[2,1]-1
       slopetab$dalastday[i]<-thesebounds[2,2]-1
       slopetab$dbslope[i]<-theseslopes[thesebounds[1,1],thesebounds[1,2],1]
       slopetab$dbse[i]<-theseslopes[thesebounds[1,1],thesebounds[1,2],2]
       slopetab$daslope[i]<-theseslopes[thesebounds[2,1],thesebounds[2,2],1]
       slopetab$dase[i]<-theseslopes[thesebounds[2,1],thesebounds[2,2],2]
   }
   if(slopetab[i,1] %in% names(slopelistc))
   {   eval(parse(text=paste("theseslopes<-slopelistc$",slopetab[i,1],sep="")))
       thesebounds<-minse(theseslopes,0)
       slopetab$cbfirstday[i]<-thesebounds[1,1]-1
       slopetab$cblastday[i]<-thesebounds[1,2]-1
       slopetab$cafirstday[i]<-thesebounds[2,1]-1
       slopetab$calastday[i]<-thesebounds[2,2]-1
       slopetab$cbslope[i]<-theseslopes[thesebounds[1,1],thesebounds[1,2],1]
       slopetab$cbse[i]<-theseslopes[thesebounds[1,1],thesebounds[1,2],2]
       slopetab$caslope[i]<-theseslopes[thesebounds[2,1],thesebounds[2,2],1]
       slopetab$case[i]<-theseslopes[thesebounds[2,1],thesebounds[2,2],2]
   }
}


# discard those with no initial exponential models
slopetab<-slopetab[!is.na(slopetab$dbfirstday) | !is.na(slopetab$cbfirstday),]


# doubling times : also need version that just does halving

slopetab$dbdble<-log(2)/slopetab$dbslope
slopetab$dbdblel<-log(2)/(slopetab$dbslope+2*slopetab$dbse)
```

```r
slopetab$dbdbleu<-log(2)/(slopetab$dbslope-2*slopetab$dbse)

slopetab$dadble<-log(2)/abs(slopetab$daslope)
slopetab$dadblel<-log(2)/(abs(slopetab$daslope)+2*slopetab$dase)
slopetab$dadbleu<-log(2)/(abs(slopetab$daslope)-2*slopetab$dase)

# using 1000000 for Inf to ease plotting
slopetab$dahalve<-ifelse(slopetab$daslope<0,slopetab$dadble,NA_real_)
slopetab$dahalvel<-slopetab$dadblel
slopetab$dahalveu<-ifelse((slopetab$daslope+2*slopetab$dase)<0,slopetab$dadbleu,1000000)

slopetab$cbdble<-log(2)/slopetab$cbslope
slopetab$cbdblel<-log(2)/(slopetab$cbslope+2*slopetab$cbse)
slopetab$cbdbleu<-log(2)/(slopetab$cbslope-2*slopetab$cbse)

slopetab$cadble<-log(2)/abs(slopetab$caslope)
slopetab$cadblel<-log(2)/(abs(slopetab$caslope)+2*slopetab$case)
slopetab$cadbleu<-log(2)/(abs(slopetab$caslope)-2*slopetab$case)

slopetab$cahalve<-ifelse(slopetab$caslope<0,slopetab$cadble,NA_real_)
slopetab$cahalvel<-slopetab$cadblel
slopetab$cahalveu<-ifelse((slopetab$caslope+2*slopetab$case)<0,slopetab$cadbleu,1000000)



# r0

# 4 day -old way
slopetab$dbr4<-exp(4*slopetab$dbslope)
slopetab$dbr4l<-exp(4*(slopetab$dbslope-2*slopetab$dbse))
slopetab$dbr4u<-exp(4*(slopetab$dbslope+2*slopetab$dbse))
slopetab$dar4<-exp(4*slopetab$daslope)
slopetab$dar4l<-exp(4*(slopetab$daslope-2*slopetab$dase))
slopetab$dar4u<-exp(4*(slopetab$daslope+2*slopetab$dase))

slopetab$cbr4<-exp(4*slopetab$cbslope)
slopetab$cbr4l<-exp(4*(slopetab$cbslope-2*slopetab$cbse))
slopetab$cbr4u<-exp(4*(slopetab$cbslope+2*slopetab$cbse))
slopetab$car4<-exp(4*slopetab$caslope)
slopetab$car4l<-exp(4*(slopetab$caslope-2*slopetab$case))
slopetab$car4u<-exp(4*(slopetab$caslope+2*slopetab$case))

# new

library(R0)

# Nishiura et al 2020: lognormal mean 4.7 sd 2.9
mGT <- generation.time("lognormal", c(4.7,2.9))


# because these are only a sample of the people, R0 doesn't like working directly off them
# so go via the slope estimates

slopetab$dbr0<-NA_real_
slopetab$dbr0l<-NA_real_
slopetab$dbr0u<-NA_real_
slopetab$dar0<-NA_real_
slopetab$dar0l<-NA_real_
slopetab$dar0u<-NA_real_

slopetab$cbr0<-NA_real_
slopetab$cbr0l<-NA_real_
slopetab$cbr0u<-NA_real_
slopetab$car0<-NA_real_
slopetab$car0l<-NA_real_
slopetab$car0u<-NA_real_

for(i in 1:nrow(slopetab))
{
   if(!is.na(slopetab$dblastday[i]))
   { slopetab$dbr0[i]<- try(R0:::R.from.r(slopetab$dbslope[i],GT=mGT),TRUE)
     slopetab$dbr0l[i]<- try(R0:::R.from.r(slopetab$dbslope[i]-
2*slopetab$dbse[i],GT=mGT),TRUE)
     slopetab$dbr0u[i]<-
try(R0:::R.from.r(slopetab$dbslope[i]+2*slopetab$dbse[i],GT=mGT),TRUE)
```

```
    }

    if(!is.na(slopetab$dalastday[i]))
    {  slopetab$dar0[i]<- try(R0:::R.from.r(slopetab$daslope[i],GT=mGT),TRUE)
       slopetab$dar0l[i]<- try(R0:::R.from.r(slopetab$daslope[i]-
2*slopetab$dase[i],GT=mGT),TRUE)
       slopetab$dar0u[i]<-
try(R0:::R.from.r(slopetab$daslope[i]+2*slopetab$dase[i],GT=mGT),TRUE)
    }

    if(!is.na(slopetab$cblastday[i]))
    {  slopetab$cbr0[i]<- try(R0:::R.from.r(slopetab$cbslope[i],GT=mGT),TRUE)
       slopetab$cbr0l[i]<- try(R0:::R.from.r(slopetab$cbslope[i]-
2*slopetab$cbse[i],GT=mGT),TRUE)
       slopetab$cbr0u[i]<-
try(R0:::R.from.r(slopetab$cbslope[i]+2*slopetab$cbse[i],GT=mGT),TRUE)
    }

    if(!is.na(slopetab$calastday[i]))
    {  slopetab$car0[i]<- try(R0:::R.from.r(slopetab$caslope[i],GT=mGT),TRUE)
       slopetab$car0l[i]<- try(R0:::R.from.r(slopetab$caslope[i]-
2*slopetab$case[i],GT=mGT),TRUE)
       slopetab$car0u[i]<-
try(R0:::R.from.r(slopetab$caslope[i]+2*slopetab$case[i],GT=mGT),TRUE)
    }


}


# ratio of slopes; before over after

slopetab$dsloperat<--slopetab$dbslope/slopetab$daslope
slopetab$dsloperatl<-NA_real_
slopetab$dsloperatu<-NA_real_

slopetab$csloperat<--slopetab$cbslope/slopetab$caslope
slopetab$csloperatl<-NA_real_
slopetab$csloperatu<-NA_real_

for(i in 1:nrow(slopetab))
{  rats<-rnorm(10000,slopetab$dbslope[i],slopetab$dbse[i])/
           rnorm(10000,-slopetab$daslope[i],slopetab$dase[i])
   if(sum(is.na(rats))==0)
   {  slopetab$dsloperatl[i]<-quantile(rats,0.025)
      slopetab$dsloperatu[i]<-quantile(rats,0.975)
   }

   rats<-rnorm(10000,slopetab$cbslope[i],slopetab$cbse[i])/
           rnorm(10000,-slopetab$caslope[i],slopetab$case[i])
   if(sum(is.na(rats))==0)
   {  slopetab$csloperatl[i]<-quantile(rats,0.025)
      slopetab$csloperatu[i]<-quantile(rats,0.975)
   }
}

# and proportion of time off lockdown: -after/(before+-after)

slopetab$dpropunlock<-NA_real_
slopetab$dpropunlockl<-NA_real_
slopetab$dpropunlocku<-NA_real_

slopetab$cpropunlock<-NA_real_
slopetab$cpropunlockl<-NA_real_
slopetab$cpropunlocku<-NA_real_

for(i in 1:nrow(slopetab))
{  before<-rnorm(10000,slopetab$dbslope[i],slopetab$dbse[i])
   after<-rnorm(10000,slopetab$daslope[i],slopetab$dase[i])
   rats<-ifelse(after>0,0,ifelse(before<=0,1,-after/(before-after)))
   if(sum(is.na(rats))==0)
   {  slopetab$dpropunlockl[i]<-quantile(rats,0.025)
      slopetab$dpropunlock[i]<-quantile(rats,0.5)
      slopetab$dpropunlocku[i]<-quantile(rats,0.975)
   }
```

```
     before<-rnorm(10000,slopetab$cbslope[i],slopetab$cbse[i])
     after<-rnorm(10000,slopetab$caslope[i],slopetab$case[i])
     rats<-ifelse(after>0,0,ifelse(before<=0,1,-after/(before-after)))
     if(sum(is.na(rats))==0)
     { slopetab$cpropunlockl[i]<-quantile(rats,0.025)
        slopetab$cpropunlock[i]<-quantile(rats,0.5)
        slopetab$cpropunlocku[i]<-quantile(rats,0.975)
     }
}


# proportion reclaimable of forgone

slopetab$dprf<-NA_real_
slopetab$dprfl<-NA_real_
slopetab$dprfu<-NA_real_

slopetab$cprf<-NA_real_
slopetab$cprfl<-NA_real_
slopetab$cprfu<-NA_real_

for(i in 1:nrow(slopetab))
{ bit<-pmax(0,rnorm(10000,slopetab$dbr0[i],(slopetab$dbr0u[i]-slopetab$dbr0l[i])/4))
   ait<-pmax(0,rnorm(10000,slopetab$dar0[i],(slopetab$dar0u[i]-slopetab$dar0l[i])/4))
   rats<-ifelse(ait>=pmin(1,bit),0, (1-ait)/(bit-ait))
   if(sum(is.na(rats))==0)
   { slopetab$dprfl[i]<-quantile(rats,0.025)
      slopetab$dprf[i]<-quantile(rats,0.5)
      slopetab$dprfu[i]<-quantile(rats,0.975)
   }

   bit<-pmax(0,rnorm(10000,slopetab$cbr0[i],(slopetab$cbr0u[i]-slopetab$cbr0l[i])/4))
   ait<-pmax(0,rnorm(10000,slopetab$car0[i],(slopetab$car0u[i]-slopetab$car0l[i])/4))
   rats<-ifelse(ait>=pmin(1,bit),0, (1-ait)/(bit-ait))
   if(sum(is.na(rats))==0)
   { slopetab$cprfl[i]<-quantile(rats,0.025)
      slopetab$cprf[i]<-quantile(rats,0.5)
      slopetab$cprfu[i]<-quantile(rats,0.975)
   }
}


# function for country plot

trajplot<-function(ctry="Italy",response="deaths",dat=dat11,slopetable=slopetab)
{
   datc<-dat11[dat11$country==ctry,]
   if(response=="deaths")
   { datc$resp<-datc$deaths
     firstnlast<-as.numeric(slopetable[slopetable$country==ctry,
                       c("dbfirstday", "dblastday", "dafirstday", "dalastday")])
     datc<-datc[cumsum2(datc$resp)>10,]
   } else
   { datc$resp<-datc$cases
     firstnlast<-as.numeric(slopetable[slopetable$country==ctry,
                       c("cbfirstday", "cblastday", "cafirstday", "calastday")])
     datc<-datc[cumsum2(datc$resp)>100,]
   }

   require(mgcv)
   gam1<-gam(resp~s(d2020),data=datc,family=nb(),na.action=na.exclude)
   pgam1<-predict(gam1,newdata=datc,se=TRUE)

   plot(datc$d2020,datc$resp,pch=20,
                 xlab="days from January 1st",ylab=paste("daily",response),main=ctry)
   polygon(c(datc$d2020, rev(datc$d2020)),
           c(exp(pgam1$fit+2*pgam1$se.fit),rev(exp(pgam1$fit-2*pgam1$se.fit))), col="grey")

   lines(datc$d2020,exp(pgam1$fit))
   points(datc$d2020, datc$resp,pch=20)

   if(!is.na(firstnlast[2]))
```

```
    {  gb<-glm.nb(resp~d2020,data=datc[datc$d2020>=firstnlast[1] &
datc$d2020<=firstnlast[2],],na.action=na.exclude)
      pgb<-predict(gb,newdata=datc[datc$d2020>=firstnlast[1] &
datc$d2020<=firstnlast[2],],se=TRUE)

      lines(min(datc$d2020):max(datc$d2020),

predict(gb,data.frame(d2020=min(datc$d2020):max(datc$d2020)),type="response"),col=2,lty="dott
ed")
      lines(datc$d2020[datc$d2020>=firstnlast[1] & datc$d2020<=firstnlast[2]],
            exp(pgb$fit),col=2,lwd=2)
      lines(datc$d2020[datc$d2020>=firstnlast[1] & datc$d2020<=firstnlast[2]],
            exp(pgb$fit+2*pgb$se.fit),col=2,lty="dashed")
      lines(datc$d2020[datc$d2020>=firstnlast[1] & datc$d2020<=firstnlast[2]],
            exp(pgb$fit-2*pgb$se.fit),col=2,lty="dashed")
    }

    if(!is.na(firstnlast[4]))
    {  ga<-glm.nb(resp~d2020,data=datc[datc$d2020>=firstnlast[3] &
datc$d2020<=firstnlast[4],],na.action=na.exclude)
      pga<-predict(ga,newdata=datc[datc$d2020>=firstnlast[3] &
datc$d2020<=firstnlast[4],],se=TRUE)

      lines(datc$d2020[datc$d2020>=firstnlast[3] & datc$d2020<=firstnlast[4]],
            exp(pga$fit),col=2,lwd=2)
      lines(datc$d2020[datc$d2020>=firstnlast[3] & datc$d2020<=firstnlast[4]],
            exp(pga$fit+2*pga$se.fit),col=2,lty="dashed",lwd=2)
      lines(datc$d2020[datc$d2020>=firstnlast[3] & datc$d2020<=firstnlast[4]],
            exp(pga$fit-2*pga$se.fit),col=2,lty="dashed",lwd=2)
    }
}

##################


# list of problematic situations
badcases<-c("Algeria", "Bosnia", "Canada","China", "Greece", "Iran", "Moldova", "Panama",
"Poland")
baddeaths<-c("Canada", "Romania")

###############

# plots


# fig 1 example trajectories

tiff("C:/mike/covid/trajectories submitted erj/ms figs2/fig1.tif",
    width = 800, height = 800, units = "px")#, compression = "lzw")
par(mfrow=c(5,2))
par(mar=c(2, 4, 2, 1) + 0.1)
trajplot("USA","cases")
trajplot("USA")
trajplot("Italy","cases")
trajplot()
trajplot("Spain","cases")
trajplot("Spain")
trajplot("France","cases")
trajplot("France")
trajplot("UK","cases")
trajplot("UK")
dev.off()

# fig 2: doubling time before and after, and R before and after
# make negatives in ahalvel and bdbleu large for plotting


tiff("C:/mike/covid/trajectories submitted erj/ms figs2/fig2.tif",
        width = 800, height = 1200, units = "px")#, compression = "lzw")
slopetabt<-slopetab[order(slopetab[,1]),][nrow(slopetab):1,]

par(mfrow=c(1,4))
par(mar=c(5, 6, 4, 2) + 0.1)
```

```
plot(slopetabt$dbdble ,1:nrow(slopetabt),xlim=c(0,12),ylim=c(0,nrow(slopetabt)+1),yaxs="i",ya
xt="n",ylab="",
             xlab="A) doubling time (days)",xaxs="i",
             pch=ifelse(slopetabt$country  %in% baddeaths,21,19))

axis(2,1:nrow(slopetabt), slopetabt$country ,las=2)

segments(slopetabt$dbdblel ,1:nrow(slopetabt),ifelse(slopetabt$dbdbleu >=0,slopetabt$dbdbleu
,100000),1:nrow(slopetabt),
             lwd=ifelse(slopetabt$country  %in% baddeaths,1,2))

points(slopetabt$cbdble ,1:nrow(slopetabt)+0.2,col=2,
             pch=ifelse(slopetabt$country  %in% badcases,21,19))

segments(slopetabt$cbdblel ,1:nrow(slopetabt)+0.2,

ifelse(slopetabt$cbdbleu >=0,slopetabt$cbdbleu ,100000),1:nrow(slopetabt)+0.2,col=2,
             lwd=ifelse(slopetabt$country  %in% badcases,1,2))

abline(v=7,lty="dotted")


plot(slopetabt$dahalve ,1:nrow(slopetabt),ylim=c(0,nrow(slopetabt)+1),yaxs="i",yaxt="n",ylab=
"",
           xlab="B) halving time (days)",xlim=c(0,60),xaxs="i",
             pch=ifelse(slopetabt$country  %in% baddeaths,21,19))

axis(2,1:nrow(slopetabt), slopetabt$country ,las=2)

segments(slopetabt$dahalvel ,
                  1:nrow(slopetabt),slopetabt$dahalveu ,1:nrow(slopetabt),
             lwd=ifelse(slopetabt$country  %in% baddeaths,1,2))

points(slopetabt$cahalve ,1:nrow(slopetabt)+0.2,col=2,
             pch=ifelse(slopetabt$country  %in% badcases,21,19))

segments(slopetabt$cahalvel ,1:nrow(slopetabt)+0.2,
                 slopetabt$cahalveu ,1:nrow(slopetabt)+0.2,col=2,
             lwd=ifelse(slopetabt$country  %in% badcases,1,2))

abline(v=7,lty="dotted")


plot(exp(4*(slopetabt$dbslope )),1:nrow(slopetabt),ylim=c(0,nrow(slopetabt)+1),yaxs="i",yaxt=
"n",ylab="",
           xlab="C) R0 before lockdown",xlim=c(0,4.2),xaxs="i",
             pch=ifelse(slopetabt$country  %in% baddeaths,21,19))

axis(2,1:nrow(slopetabt), slopetabt$country ,las=2)

segments(exp(4*(slopetabt$dbslope -2*slopetabt$dbse )),1:nrow(slopetabt),
           exp(4*(slopetabt$dbslope +2*slopetabt$dbse )),1:nrow(slopetabt),
             lwd=ifelse(slopetabt$country  %in% baddeaths,1,2))

points(exp(4*(slopetabt$cbslope )),1:nrow(slopetabt)+0.2,col=2,
             pch=ifelse(slopetabt$country  %in% badcases,21,19))

segments(exp(4*(slopetabt$cbslope -2*slopetabt$cbse )),1:nrow(slopetabt)+0.2,
               exp(4*(slopetabt$cbslope +2*slopetabt$cbse )),1:nrow(slopetabt)+0.2,col=2,
             lwd=ifelse(slopetabt$country  %in% badcases,1,2))
#abline(v=1,lty="dotted")


plot(exp(4*(slopetabt$daslope )),1:nrow(slopetabt),
ylim=c(0,nrow(slopetabt)+1),yaxs="i",yaxt="n",ylab="",
           xlab="D) R0 under lockdown",xlim=c(0,2),xaxs="i",
             pch=ifelse(slopetabt$country  %in% baddeaths,21,19))

axis(2,1:nrow(slopetabt), slopetabt$country ,las=2)

segments(exp(4*(slopetabt$daslope -2*slopetabt$dase )),1:nrow(slopetabt),
           exp(4*(slopetabt$daslope +2*slopetabt$dase )),1:nrow(slopetabt),
             lwd=ifelse(slopetabt$country  %in% baddeaths,1,2))

points(exp(4*(slopetabt$caslope )),1:nrow(slopetabt)+0.2,col=2,
```

```
            pch=ifelse(slopetabt$country  %in% badcases,21,19))

segments(exp(4*(slopetabt$caslope -2*slopetabt$case )),1:nrow(slopetabt)+0.2,
         exp(4*(slopetabt$caslope +2*slopetabt$case )),1:nrow(slopetabt)+0.2,col=2,
         lwd=ifelse(slopetabt$country  %in% badcases,1,2))

#abline(v=1,lty="dotted")
abline(v=0.75,lty="dotted")


dev.off()


# fig 3: proportion of time off lockdown

slopetabt<-slopetab[order(slopetab[,1]),][nrow(slopetab):1,]

tiff("C:/mike/covid/trajectories submitted erj/ms figs2/fig3.tif", width = 800, height = 1200,
units = "px")#, compression = "lzw")

# only show lines that don't go 0 to 1
slopetabt$dpropunlock[slopetabt$dpropunlocku-slopetabt$dpropunlockl>=1]<-NA_real_
slopetabt$dpropunlockl[is.na(slopetabt$dpropunlock)]<-NA_real_
slopetabt$dpropunlocku[is.na(slopetabt$dpropunlock)]<-NA_real_
slopetabt$cpropunlock[slopetabt$cpropunlocku-slopetabt$cpropunlockl>=1]<-NA_real_
slopetabt$cpropunlockl[is.na(slopetabt$cpropunlock)]<-NA_real_
slopetabt$cpropunlocku[is.na(slopetabt$cpropunlock)]<-NA_real_
slopetabt<-slopetabt[!is.na(slopetabt$dpropunlock) | !is.na(slopetabt$cpropunlock),]

par(mar=c(5, 7, 4, 2) + 0.1)
par(mfrow=c(1,1))

plot(slopetabt$dpropunlock ,1:nrow(slopetabt),ylim=c(0,nrow(slopetabt)+1),yaxs="i",yaxt="n",y
lab="",xaxs="i",
              xlab="Time ratio: the proportion of time off lockdown without epidemic
resurgance",xlim=c(0,1),
            pch=ifelse(slopetabt$country  %in% baddeaths,21,19))

axis(2,1:nrow(slopetabt), slopetabt$country ,las=2)

segments(slopetabt$dpropunlockl ,1:nrow(slopetabt),slopetabt$dpropunlocku ,1:nrow(slopetabt),
         lwd=ifelse(slopetabt$country  %in% baddeaths,1,2))

points(slopetabt$cpropunlock ,1:nrow(slopetabt)+0.2,col=2,
         pch=ifelse(slopetabt$country  %in% badcases,21,19))

segments(slopetabt$cpropunlockl ,1:nrow(slopetabt)+0.2,
         slopetabt$cpropunlocku ,1:nrow(slopetabt)+0.2,col=2,
         lwd=ifelse(slopetabt$country  %in% badcases,1,2))

abline(v=7*12/365,lty="dotted")

dev.off()


# fig 4: proportion of forgone that can be reclaimed

slopetabt<-slopetab[order(slopetab[,1]),][nrow(slopetab):1,]

tiff("C:/mike/covid/trajectories submitted erj/ms figs2/fig4.tif", width = 800, height = 1200,
units = "px")#, compression = "lzw")

slopetabt<-slopetabt[(!is.na(slopetabt$dprf) & slopetabt$dprfu<1) |
                        (!is.na(slopetabt$cprf) & slopetabt$cprfu<1),]

# only show lines that don't go 0 to 1
slopetabt$dprf[slopetabt$dprfu-slopetabt$dprfl>=1]<-NA_real_
slopetabt$dprfl[is.na(slopetabt$dprf)]<-NA_real_
slopetabt$dprfu[is.na(slopetabt$dprf)]<-NA_real_
slopetabt$cprf[slopetabt$cprfu-slopetabt$cprfl>=1]<-NA_real_
slopetabt$cprfl[is.na(slopetabt$cprf)]<-NA_real_
slopetabt$cprfu[is.na(slopetabt$cprf)]<-NA_real_
slopetabt<-slopetabt[!is.na(slopetabt$dprf) | !is.na(slopetabt$cprf),]
```

```
par(mar=c(5, 7, 4, 2) + 0.1)
par(mfrow=c(1,1))

plot(slopetabt$dprf ,1:nrow(slopetabt),ylim=c(0,nrow(slopetabt)+1),yaxs="i",yaxt="n",ylab="",
xaxs="i",
          xlab="Reclaimable fraction: the proportion of contacts, forgone under lockdown,
reclaimable without epidemic resurgance",xlim=c(0,1),
          pch=ifelse(slopetabt$country  %in% baddeaths,21,19))

axis(2,1:nrow(slopetabt), slopetabt$country ,las=2)

segments(slopetabt$dprfl ,1:nrow(slopetabt),slopetabt$dprfu ,1:nrow(slopetabt),
          lwd=ifelse(slopetabt$country  %in% baddeaths,1,2))

points(slopetabt$cprf ,1:nrow(slopetabt)+0.2,col=2,
          pch=ifelse(slopetabt$country  %in% badcases,21,19))

segments(slopetabt$cprfl ,1:nrow(slopetabt)+0.2,
             slopetabt$cprfu ,1:nrow(slopetabt)+0.2,col=2,
          lwd=ifelse(slopetabt$country  %in% badcases,1,2))

abline(v=0.2,lty="dotted")

par(mar=c(5, 4, 4, 2) + 0.1)

dev.off()




# supplementary plots: all country trajectories

for(i in 1:nrow(slopetab))
{
   if(!is.na(slopetab$cbfirstday[i]))
   { tiff(paste("C:/mike/covid/trajectories submitted erj/ms figs2/",slopetab[i,1]," cases
fig.tif",sep=""),
          width = 600, height = 600, units = "px") #, compression = "lzw")
      par(mar=c(5, 4, 4, 2) + 0.1)
      try(trajplot(slopetab[i,1],"cases"),TRUE)
      dev.off()
   }

   if(!is.na(slopetab$dbfirstday[i]))
   {   tiff(paste("C:/mike/covid/trajectories submitted erj/ms figs2/",slopetab[i,1]," deaths
fig.tif",sep=""),
               width = 600, height = 600, units = "px") #, compression = "lzw")
      par(mar=c(5, 4, 4, 2) + 0.1)
      try(trajplot(slopetab[i,1],"deaths"),TRUE)
      dev.off()
   }
}


# supplementary table 1: summary data
#  cases to date; deaths to date;
# first and last d2020, slopes, ses, doubling time + CI, r0 + CI, for each exponential period;
# - sort by name

slopetab2<-slopetab[order(slopetab[,1]),]

# and restore the NAs that were set to 1000000
slopetab2[slopetab2==1000000]<-NA_real_

# round and export
slopetab2<-cbind(country=slopetab2[,1],casemodel=ifelse(slopetab2[,1] %in% badcases,"?"," "),
                      deathmodel=ifelse(slopetab2[,1] %in% baddeaths,"?"," "),
round(slopetab2[,-1],3))
write.csv(slopetab2,"C:/mike/covid/trajectories submitted erj/ms figs2/slopetab2.csv")

# for table 1

# show information for the 6 largest epidemics
```

```
cbind(slopetab[order(-slopetab$deaths),][1:5,]$country,
        round(slopetab[order(-slopetab$deaths),][1:5,c("cases", "deaths", "dbdble", "dbdblel",
"dbdbleu",
                                                        "dahalve", "dahalvel",
"dahalveu",
                                    "cbdble","cbdblel", "cbdbleu", "cahalve", "cahalvel",
"cahalveu")],3))


cbind(slopetab[order(-slopetab$deaths),][1:5,]$country,
        round(slopetab[order(-slopetab$deaths),][1:5,c("cases", "deaths",
                                    "dbr0", "dbr0l", "dbr0u", "dar0", "dar0l", "dar0u",
                                    "cbr0" ,"cbr0l", "cbr0u", "car0", "car0l", "car0u")],2))

## dataset description
dim(slopetab)
sum(dat11$cases)
sum(dat11$deaths)
length(unique(dat11$country))

sum(!is.na(slopetab$dbslope))
sum(!is.na(slopetab$daslope))
sum(!is.na(slopetab$cbslope))
sum(!is.na(slopetab$caslope))

which(slopetab$daslope>slopetab$dbslope)
which(slopetab$caslope>slopetab$cbslope)

hist(slopetab$car0,breaks=30)

slopetab$country[which(slopetab$cprfl>0 & slopetab$dprfl>0)]

slopetab$country[which(slopetab$car0u<0.9)]
slopetab$country[which(slopetab$dar0u<0.9)]
```