# 周课 7

陈漂亮 Savvy 2020

Sunday 15 March 2020

# 目录



## 0.1 1. Non-Parametric Model

- Penalized likelihood;
- smoothing;
- fitting wiggly lines through points;
- semi-parametric models;
- splines

$$Y_i \sim \pi(\lambda_i, \theta)$$
$$g(\lambda_i) = X_i\beta + f(W_i)$$

Where,
- $Y_i$ : response
- $\pi(\lambda_i, \theta)$ is the response distribution
- $X_i, W_i arecovariates$
- $f(w)$ : is the smoothing function
- $g(\lambda)$ : is the link function
- $\beta$ : coefficients

### 0.1.1 1. Penalized Likelihood

$$L_P(\beta, f, \alpha; Y) = \log(\pi(Y; \beta, f)) - \alpha \int \left[\frac{\partial^2 f(w)}{\partial w^2}\right]^2 du$$

Where,
- $\alpha$: penalty parameter, $\alpha \uparrow \implies$ smoother $f$
- smoother $f(x) \implies$ smaller $f''(x)$

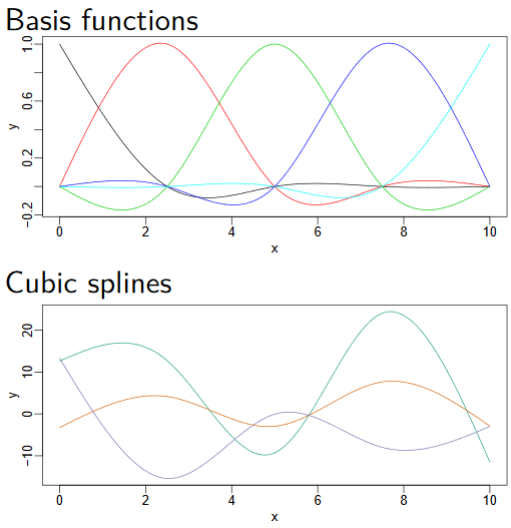
$$\implies \hat{\beta}(\alpha), \hat{f}(\alpha) = \arg\min_{\beta, f} L_P(\beta, f, \alpha; Y)$$

- A good $\hat{f}$ is a compromise between fitting the data and being smooth.

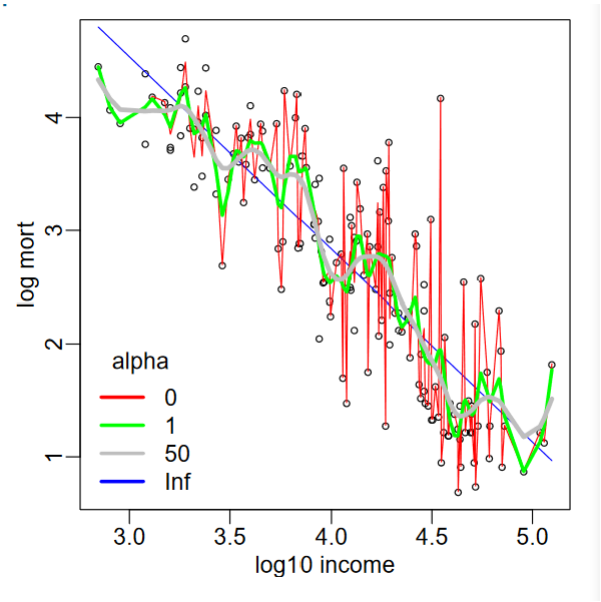
### 0.1.2 1.2 Cubic Spline

- The f that maximizes the penalized likehood must be a cubic spline polynomial…

```
knitr::include_graphics("1.png")
```



```
knitr::include_graphics("2.png")
```




- The basis function of cubic splines: $ax^3 + bx^2 + cx + d$...

**Maiximizing likelihood** over all possible $f$:

- The larger the $\alpha$, the smoother the curve ($f$)...

- When $\alpha \to \infty$, $f$ is a straight line.

**How?**:

- Divide your data (evenly) into K subsets, and fit a cubic spline on each subset. Make sure the f function is continous/1st-order-diff/2nd-order-diff at each knot...

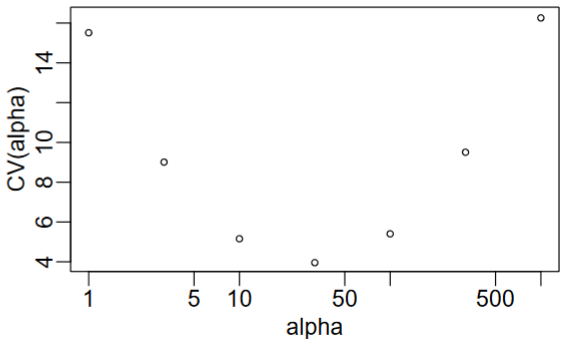
**Choosing** $\alpha$: Cross Validation

```
knitr::include_graphics("3.png")
```

Cross validation

- Find $\hat{\lambda}^{(-i)}$ by excluding observation $i$
- compute $pr(Y_i|\hat{\lambda}^{(-i)})$
- repeat for $i = 1 \ldots N$
- $\text{CV}(\alpha) = -\sum_i \log[pr(Y_i|\hat{\lambda}^{(-i)})]$

$$\hat{\alpha} = \text{argmax}_\alpha \text{CV}(\alpha)$$



## 0.2 2. Generalized Additive Model (GAM)

- Fit a GAM for the Math score data...

```
library('mgcv')
```

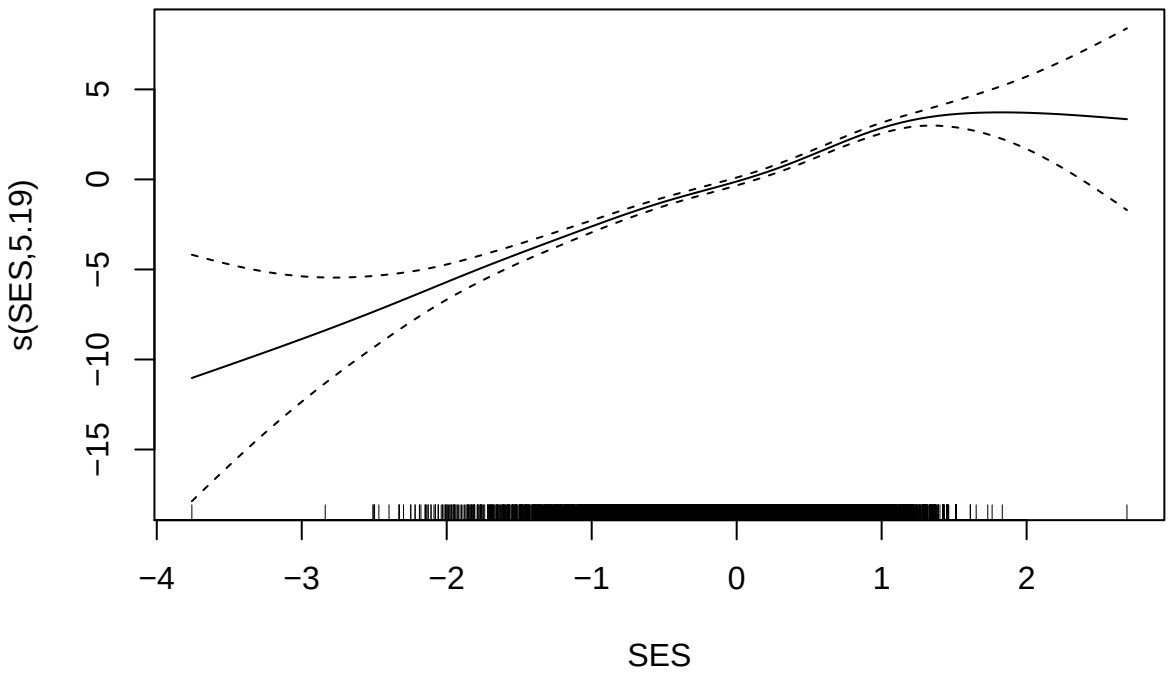
```
## Loading required package: nlme
```

```
## This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.
```

```
mathGam =gam(MathAch~s(SES)+Minority*Sex,
             data=MathAchieve)
knitr::kable(summary(mathGam)$p.table[,1:2],
             digits=1)
```

| | Estimate | Std. Error |
|---|---|---|
| (Intercept) | 14.3 | 0.1 |
| MinorityYes | -2.9 | 0.2 |
| SexFemale | -1.4 | 0.2 |
| MinorityYes:SexFemale | 0.2 | 0.3 |

```
plot(mathGam)
```




```
mathGam$sp # smoothing parameter
```

```
##    s(SES)
## 0.8254378
```

### 0.2.1 2.1 Smoothing Interation

- Now we fit another GAM, with interaction between the covariates that are being smoothed…

```
mathGamInt =gam(MathAch~s(SES,by=Minority)
                +Minority*Sex,
                data=MathAchieve)
knitr::kable(summary(mathGamInt)$p.table[,1:2],
             digits=1)
```

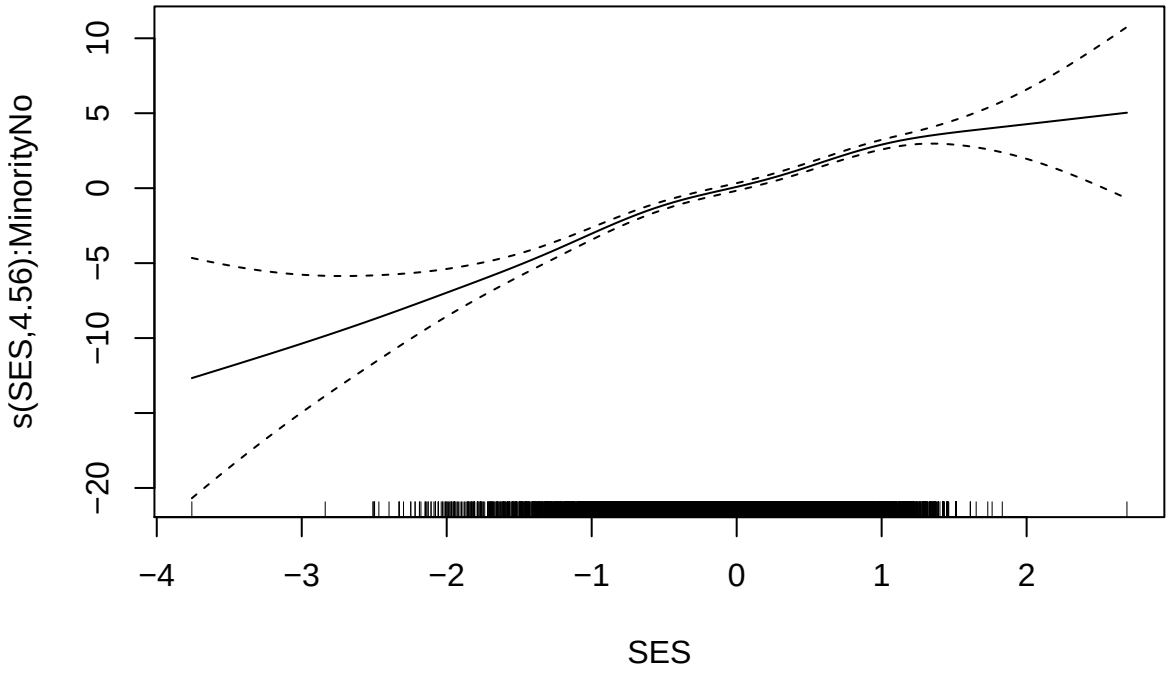
| | Estimate | Std. Error |
|---|---|---|
| (Intercept) | 14.2 | 0.1 |
| MinorityYes | -3.0 | 0.3 |
| SexFemale | -1.4 | 0.2 |

| | Estimate | Std. Error |
|---|---|---|
| MinorityYes:SexFemale | 0.1 | 0.3 |

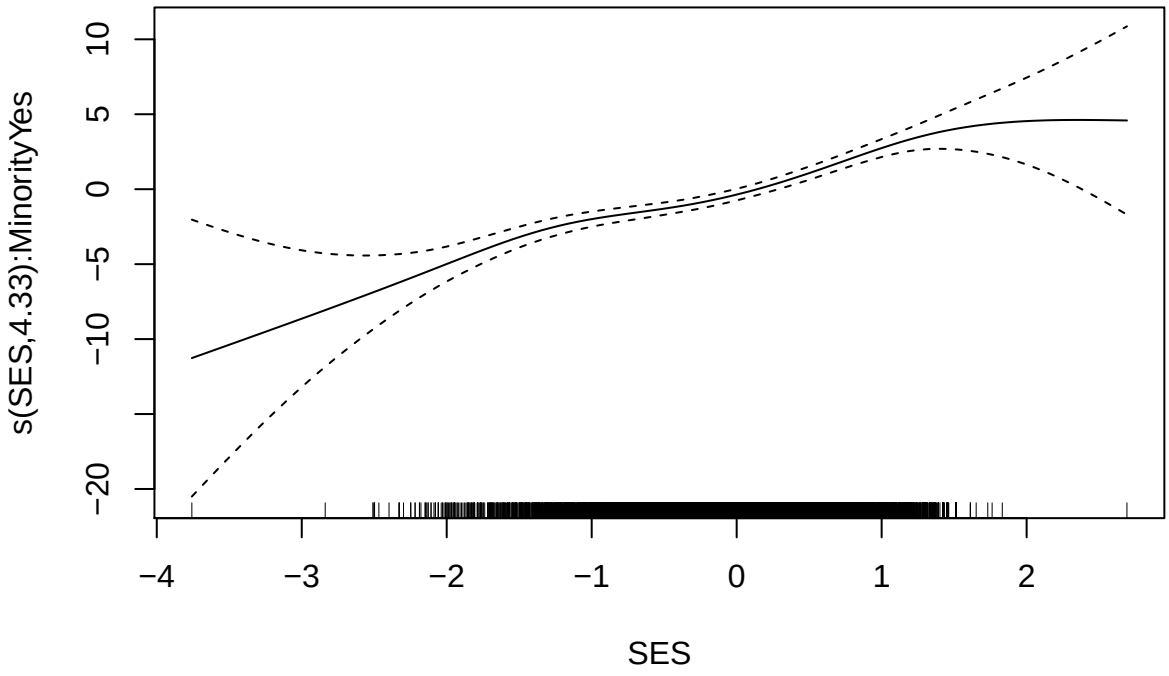
```
mathGamInt$sp
```

```
##  s(SES):MinorityNo s(SES):MinorityYes
##           0.820158           0.614983
```

```
# plot the SES/minority
plot(mathGamInt,select =1)
```




```
plot(mathGamInt,select =2)
```




### 0.2.2 2.2 Common smoothing parameter

```
knitr::include_graphics("4.png")
```

$$Y_{ij} \sim N(\lambda_{ij}, \tau^2)$$
$$\lambda_{ij} = X_{ij}\beta + f_i(W_{ij}; \nu)$$

- $Y_{ij}$ is the observation for individual $j$ in ethnic group $i$
- $X_{ij}$ is a vector of covariates (ethnic group, sex, interaction)
- $f_i(w; \nu)$ is the smoothly-varying function of SES
  - for ethnic group $i$
  - with roughness parameter $\nu$.

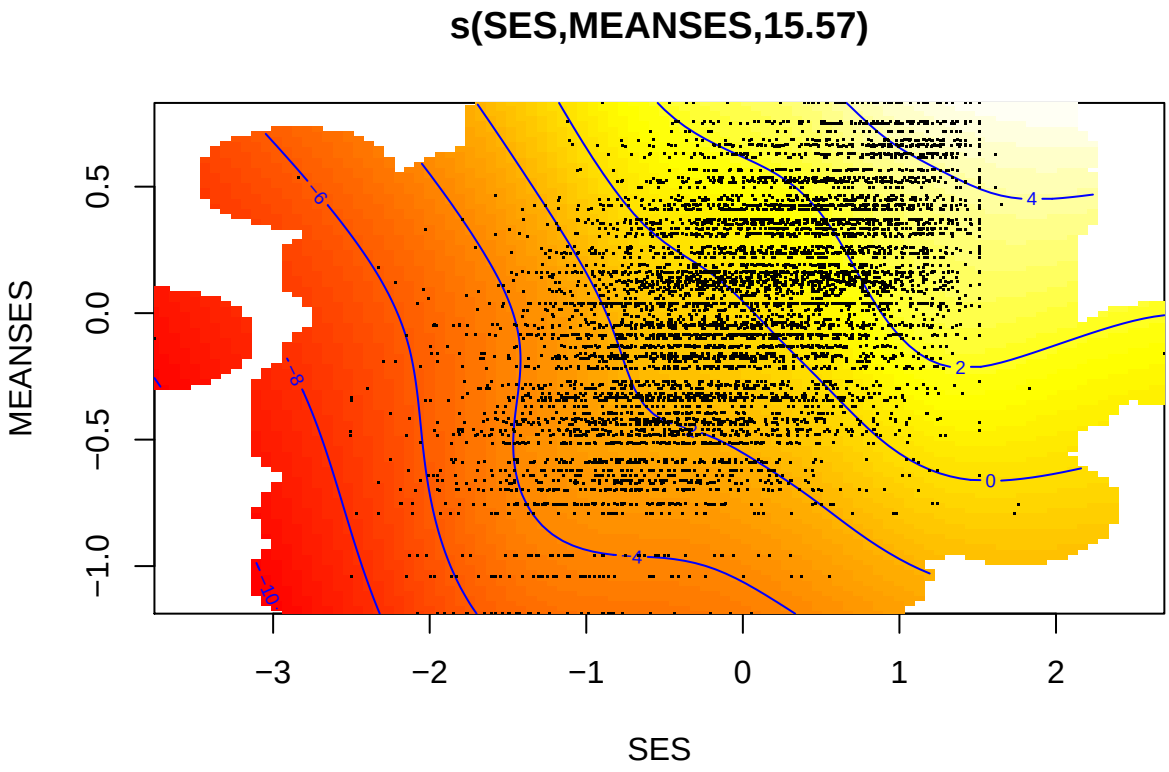
```
mathGamIntC =gam(MathAch~s(SES,by=Minority,id=1) +Minority*Sex,
                 data=MathAchieve)


mathGamIntC$sp
```

```
## s(SES):MinorityNo
```

```
##          0.7492505
```

### 0.2.3 2.3 2-D smoothing

```
mathGam2 =gam(MathAch~s(SES, MEANSES)+Minority*+Sex,
              data=MathAchieve)
plot(mathGam2,scheme =2,n2 =100)
```




- If you are from upper class, your score is still likely higher even if your school is weaker...

### 0.2.4 2.4 Poisson GAM: Ontario deaths

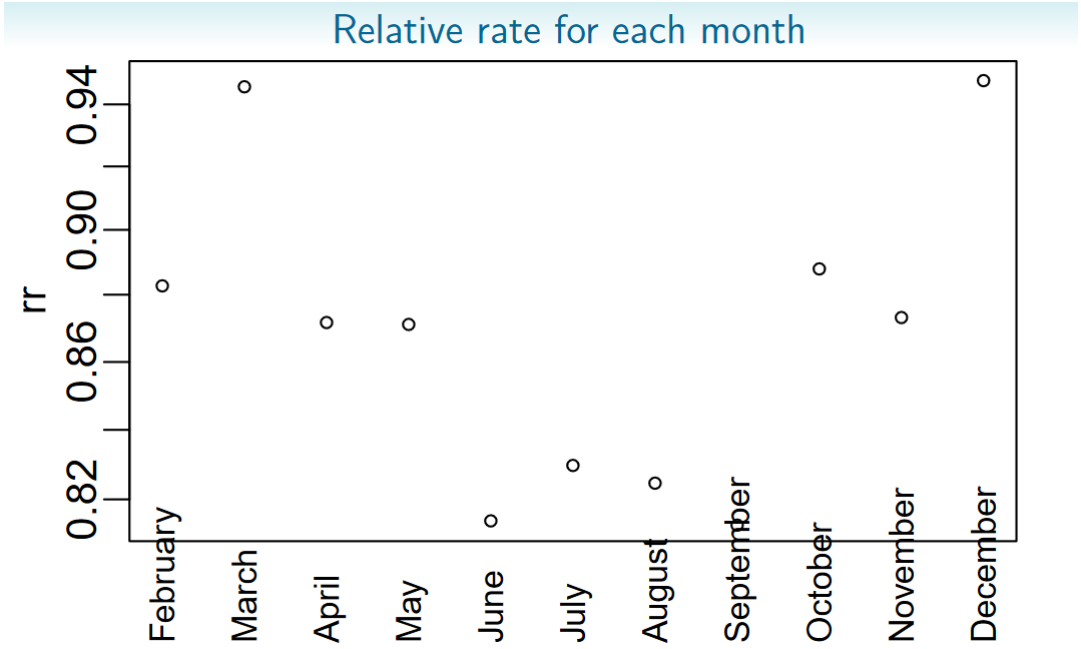
$$Y_i \sim Poisson(\lambda_i)$$
$$\log(\lambda_i) = X_i\beta + f(time)$$

where, • $\lambda_i$ is the relative rate of death in the ith month

```
deathsGam =gam(+Value~month+s(timeNumeric),
              data=oDeaths,family='poisson'+)
```

```
knitr::kable(summary(deathsGam)$p.table[,1:2],
             digits=3,col.names=c('est','se'))
```

| | est | se |
|---|---|---|
| (Intercept) | 9.001 | 0.002 |
| monthFebruary | -0.124 | 0.003 |
| monthMarch | -0.055 | 0.003 |
| monthApril | -0.137 | 0.003 |
| monthMay | -0.138 | 0.003 |
| monthJune | -0.205 | 0.003 |
| monthJuly | -0.186 | 0.003 |
| monthAugust | -0.192 | 0.003 |
| monthSeptember | -0.207 | 0.003 |
| monthOctober | -0.118 | 0.003 |
| monthNovember | -0.135 | 0.003 |
| monthDecember | -0.053 | 0.003 |



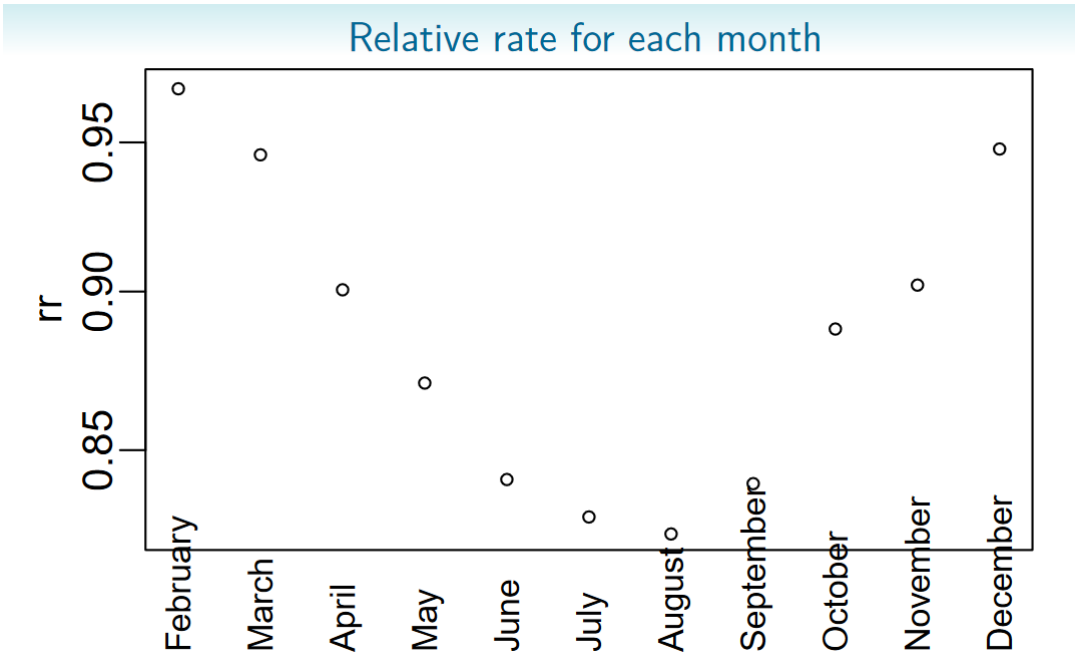

- relative rate: relative to the baseline, i.e. Janurary...
- Note that different month has different number of days, so, offset!!

$$Y_i \sim Poisson(O_i\lambda_i)$$
$$\log(\lambda_i) = X_i\beta + f(time)$$

where,
- $\lambda_i$ is the relative rate of death in the ith month
- $O_i$ is the offset term

```
deathsGam =gam(Value~month+s(timeNumeric)
               +offset(nDays),
               data=oDeaths,
              family='poisson')
knitr::kable(summary(deathsGam)$p.table[,1:2],
             digits=3,col.names=c('est','se'))
```

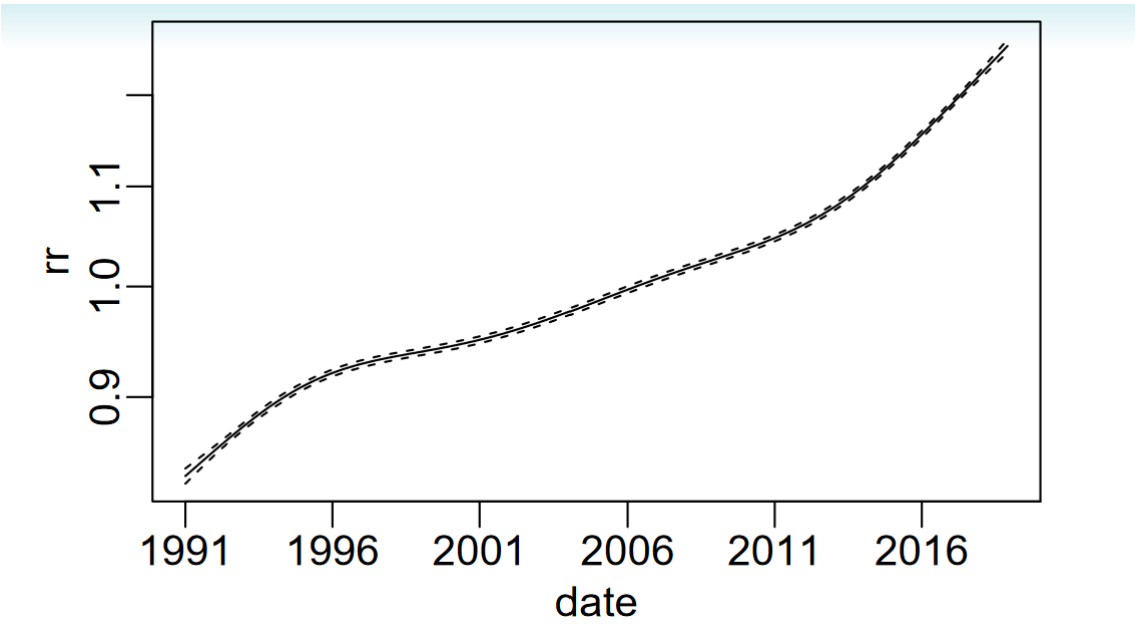
| | est | se |
|---|---|---|
| (Intercept) | 5.567 | 0.002 |
| monthFebruary | -0.031 | 0.003 |
| monthMarch | -0.055 | 0.003 |
| monthApril | -0.104 | 0.003 |
| monthMay | -0.138 | 0.003 |
| monthJune | -0.173 | 0.003 |
| monthJuly | -0.186 | 0.003 |
| monthAugust | -0.192 | 0.003 |
| monthSeptember | -0.174 | 0.003 |
| monthOctober | -0.118 | 0.003 |
| monthNovember | -0.102 | 0.003 |
| monthDecember | -0.053 | 0.003 |




### 0.2.5 2.5 Prediction

#### 0.2.5.1 2.5.1 Trend

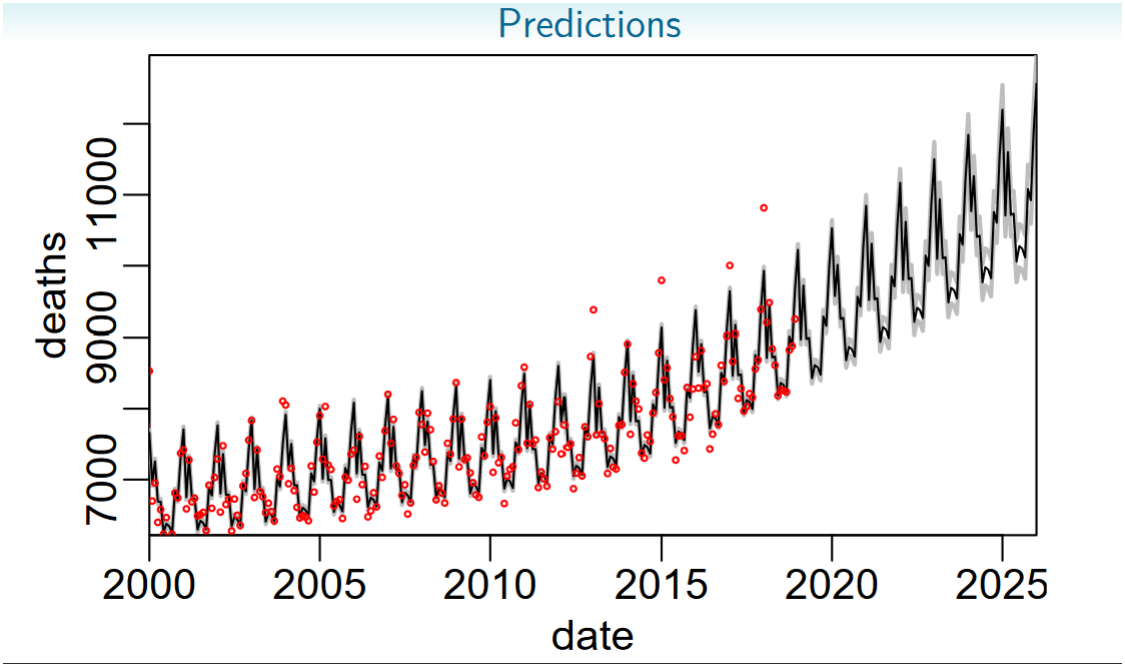
```
dSeq =seq(from =min(oDeaths$date),
          by ="5 years",length.out =10)
deathPred =as.matrix(as.data.frame(predict.gam(deathsGam, oDeaths,type ="terms",terms ="s(timeNume

deathPred =exp(deathPred%*%Pmisc::ciMat())
matplot(oDeaths$timeNumeric, deathPred,log ="y",
        xaxt ="n",xlab ="date",type ="l",
        lty =c(1,+2,2),col ="black",
        ylab ="rr")
axis(1,at =difftime(dSeq,
                    timeOrigin,units ="days"),
     labels =format(dSeq,"%Y"))
```




#### 0.2.5.2 2.5.3 Forcasting

```
Stime =seq(from =as.Date("2000/1/1"),
           to =as.Date("2026/1/1"),
           by ="months")
newX =data.frame(timeNumeric
            =as.numeric(difftime(Stime,
                                 timeOrigin,
                                 units ="days")),
            month =months(Stime),
            nDays =log(Hmisc::monthDays(Stime)))
deathsPred =predict(deathsGam, newX,se.fit =TRUE)

deathsPred =as.data.frame(deathsPred)
deathsPred$lower =deathsPred$fit-2*deathsPred$se.fit
deathsPred$upper =deathsPred$fit+2*deathsPred$se.fit
matplot(Stime,
        exp(deathsPred[,c("lower","upper",+"fit")]),
        type ="l",lty =1,
        col =c("grey",+"grey","black"),
        lwd =c(2,2,1),xlab ="date",
        ylab ="deaths",yaxs ="i",xaxs ="i",
        xaxt ="n")
forAxis =seq(from =as.Date("2000/1/1"),
```

```
              to =as.Date("2026/1/1"),
              by ="5 years")
axis(1,as.numeric(forAxis),format(forAxis,"%Y"))
points(oDeaths$date,
       oDeaths$Value,cex =0.5,col ="red")
```



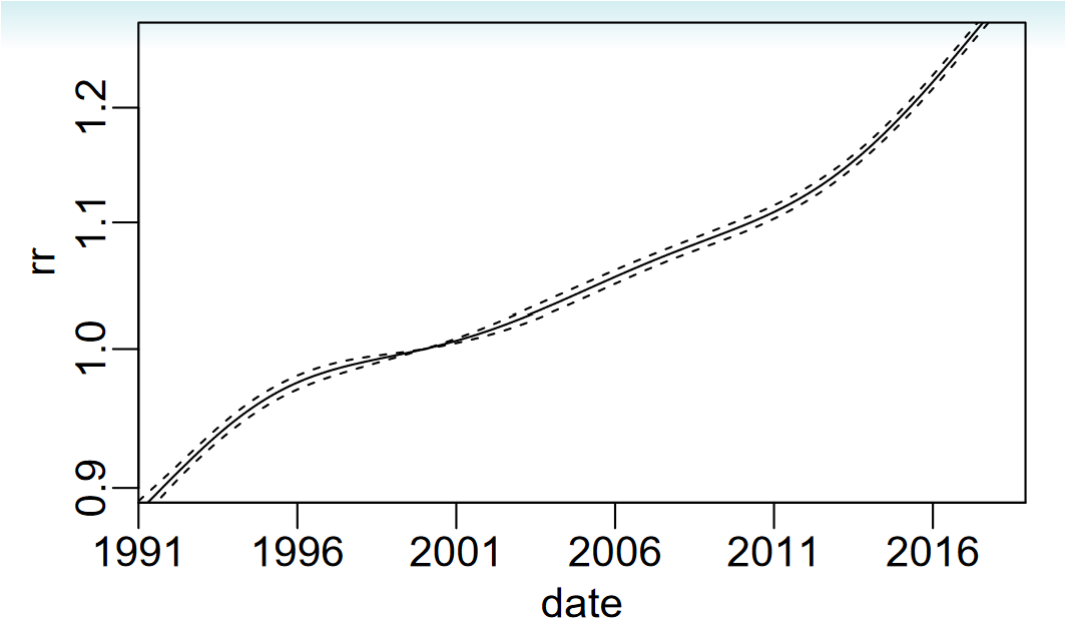

### 0.2.6 2.6 Change the Parameter constraint

- Add a constant to f(x) doesn't change the penalty;
- By default, f(x) sums to 0;
- But we don't know where does f(x) = 0;
- An alternative is to set $f(x_0) = 0$...

```
deathsGamC =gam(Value~month+s(timeNumeric,pc=0)
                +offset(nDays),
                data=oDeaths,+family='poisson')

# pc: param constraint

deathPredC =as.matrix(as.data.frame(predict.gam(deathsGamC,oDeaths,
                                                type ="terms",
```

```
                                                terms ="s(timeNumeric)",
                                                se.fit =TRUE)))

deathPredC =exp(deathPredC%*%Pmisc::ciMat())
matplot(oDeaths$timeNumeric, deathPredC,log ="y",
        xaxt ="n",xlab ="date",type ="l",
        lty =c(1,+2,2),col ="black",ylab ="rr",
        xaxs ="i",+yaxs ="i",ylim =c(0.89,1.28))
axis(1,at =difftime(dSeq, timeOrigin,
                    units ="days"),
     labels =format(dSeq,"%Y"))
```

| | est | se |
|---|---|---|
| (Intercept) | 5.510 | 0.003 |
| monthFebruary | -0.031 | 0.003 |
| monthMarch | -0.055 | 0.003 |
| monthApril | -0.104 | 0.003 |
| monthMay | -0.138 | 0.003 |
| monthJune | -0.173 | 0.003 |
| monthJuly | -0.186 | 0.003 |
| monthAugust | -0.192 | 0.003 |
| monthSeptember | -0.174 | 0.003 |
| monthOctober | -0.118 | 0.003 |
| monthNovember | -0.102 | 0.003 |
| monthDecember | -0.053 | 0.003 |



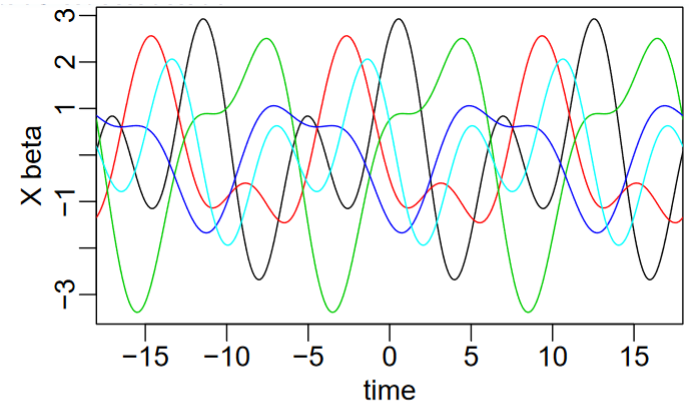
### 0.2.7 2.7 Modelling Seasonality

We can see clear seasonality from the month-effect plot.

- The trick to model seasonality is apply trignomitric function as the basis functions...

- The monthly effect isn't perfectly sinusoidal
- use a 12 month and a 6 month frequency

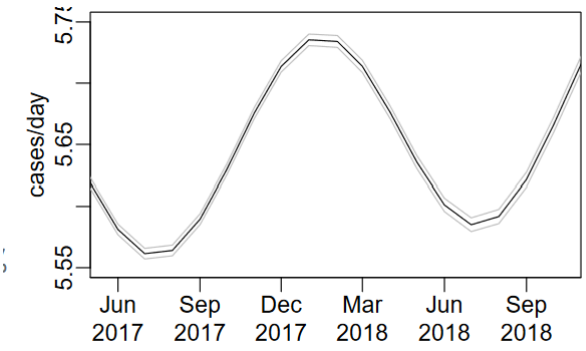
$$
\begin{aligned}
Y_i &\sim \text{Poisson}(O_i\lambda_i) \\
\log(\lambda_i) &= X_i\beta + f(t_i) \\
X_{i0} &= 1 \\
X_{i1} &= \cos(2\pi t_i/12) \\
X_{i2} &= \sin(2\pi t_i/12) \\
X_{i3} &= \cos(2\pi t_i/6) \\
X_{i4} &= \sin(2\pi t_i/6)
\end{aligned}
$$

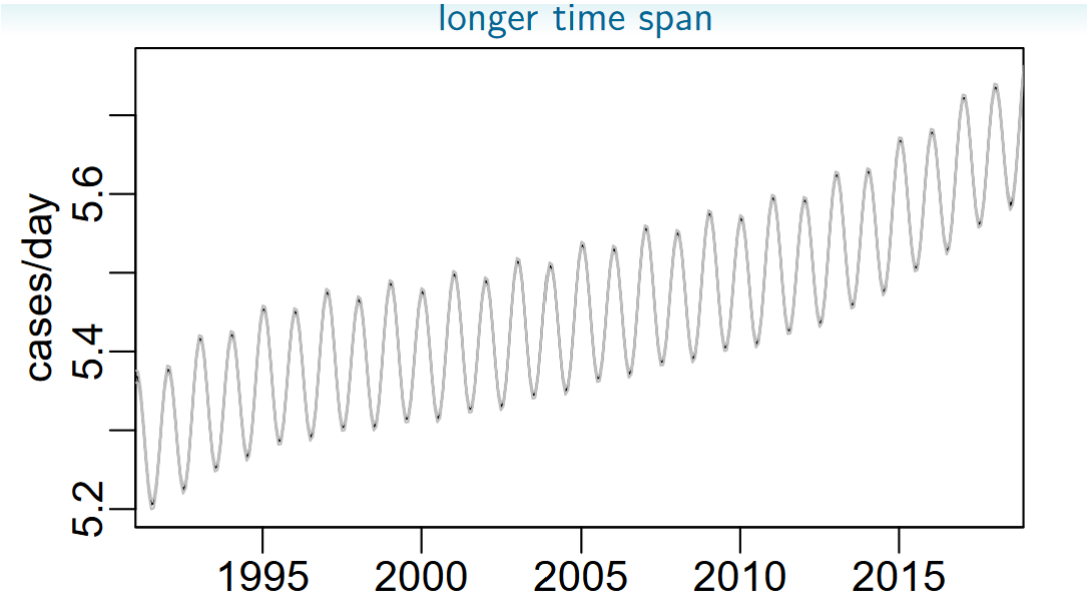



| | | | | | |
|---|---|---|---|---|---|
| cos12 | 1.26 | 0.41 | -0.01 | -1.15 | 0.25 |
| sin12 | -0.33 | -1.54 | 2.40 | -0.29 | -0.89 |
| cos6 | 1.33 | -0.93 | 0.76 | -0.30 | 0.44 |
| sin6 | 1.27 | -0.29 | -0.80 | -0.41 | -1.24 |

```
oDeaths$timeYears =oDeaths$timeNumeric/365.25
oDeaths$cos12 =cos(2*pi*oDeaths$timeYears)
oDeaths$sin12 =sin(2*pi*oDeaths$timeYears)
oDeaths$cos6 =cos(2*pi*oDeaths$timeYears/2)
oDeaths$sin6 =sin(2*pi*oDeaths$timeYears/2)

deathsGamS =gam(Value~cos12+sin12+cos6+sin6
              +s(timeNumeric)+offset(nDays),
              data=oDeaths,family='poisson')
knitr::kable(summary(deathsGamS)$p.table[,1:2],

# Predicting the seasonality
deathGamPred =predict(deathsGamS,cbind(oDeaths[,c("sin12","cos12","sin6",
                                                  "cos6","timeNumeric")],
                                       nDays =log(1)),type ="link",
                      se.fit =TRUE)
deathGamPredMat =do.call(cbind,
                         deathGamPred)%*%+Pmisc::ciMat()
```

| | est | se |
|---|---|---|
| (Intercept) | 5.456 | 0.001 |
| cos12 | 0.085 | 0.001 |
| sin12 | 0.017 | 0.001 |
| cos6 | -0.008 | 0.001 |
| sin6 | -0.003 | 0.001 |




## 0.3 3. Lab this week

The new blog he posted.

```
# ?gam


#+ header, results='asis', echo=FALSE
```

```
Pmisc::markdownHeader(
  title= "STA303, Coronavirus",
  author= 'Patrick Brown, University of Toronto'
)
```

```
#'
```

```
#+ setup, include=FALSE
library('knitr')

knitr::knit_hooks$set(margins = Pmisc::hook_plot_margins)
knitr::knit_hooks$set(plot=Pmisc::hook_plot_htmlsubfig)

knitr::opts_chunk$set(echo=FALSE, prompt=TRUE, comment=NA,
        dev='png', margins=1, fig.cap=' ',
  fig.path=file.path('figure','moreBayes'),
        fig.width=5, fig.height=3, half=NULL,
        tidy=TRUE,tidy.opts=list(indent=2, width.cutoff=45)
)

knitr::opts_hooks$set(half = function(options) {
    options$tidy.opts = list(width.cutoff = 12)#list(options$tidy.opts$width.cutoff/2)
    options
  })

options(width=80)
#'
```

```
Loading required namespace: nCov2019
```

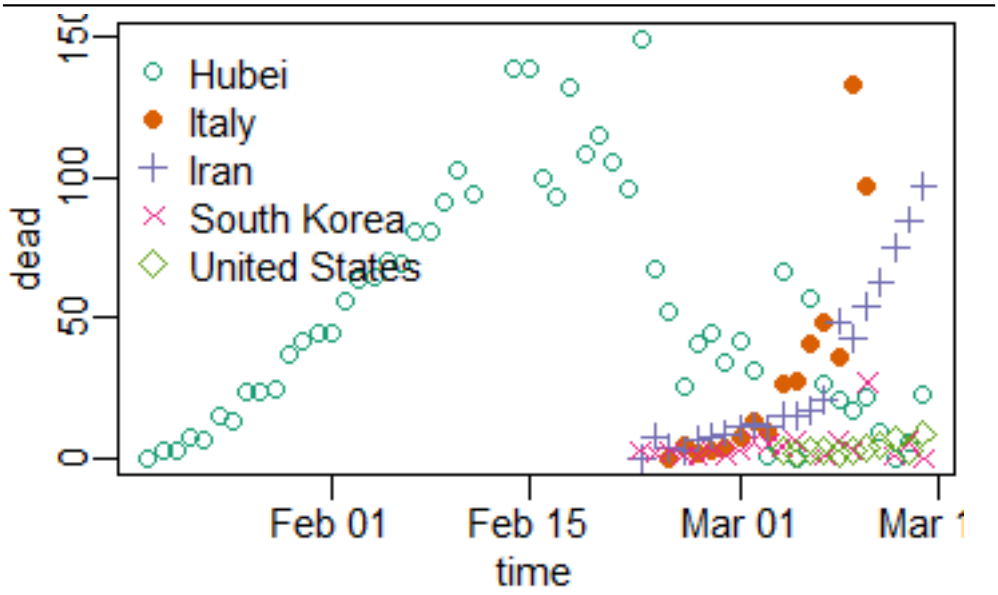
```
Warning in load(system.file("ncovEnv.rda", package = "nCov2019")): strings not
representable in native encoding will be translated to UTF-8
```

```
Warning in load(system.file("ncovEnv.rda", package = "nCov2019")): strings not
representable in native encoding will be translated to UTF-8
```

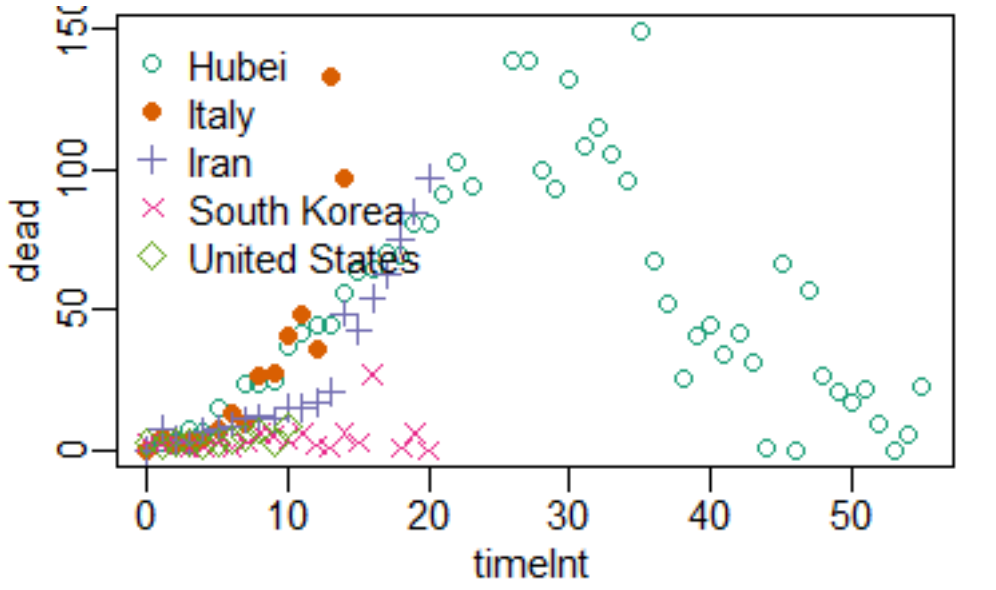
```
Warning in load(system.file("ncovEnv.rda", package = "nCov2019")): strings not
representable in native encoding will be translated to UTF-8
```

```
Warning in readRDS(system.file("country_translate.rds", package = "nCov2019")):
strings not representable in native encoding will be translated to UTF-8
```

表 3: {#tbl:unnamed-chunk-25}

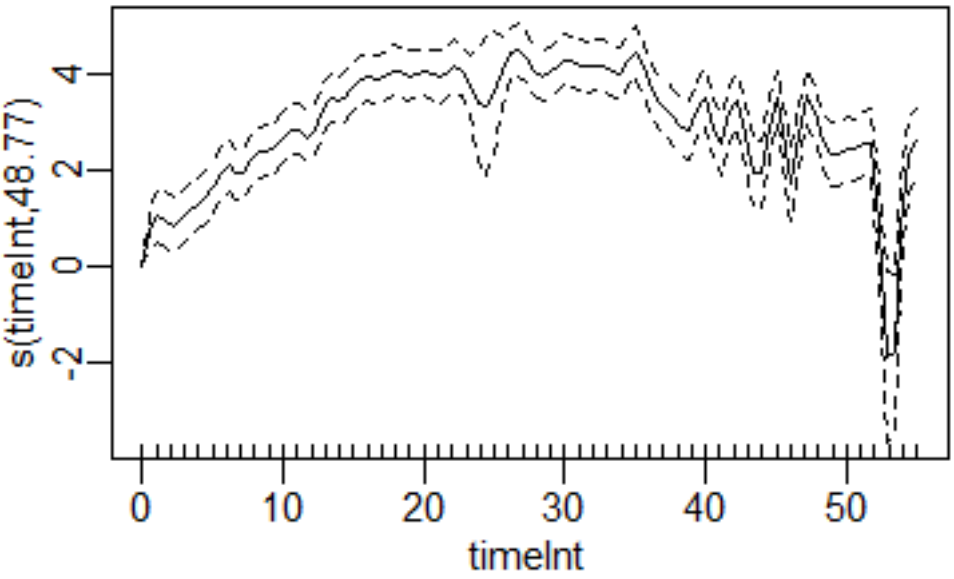



a)




b)

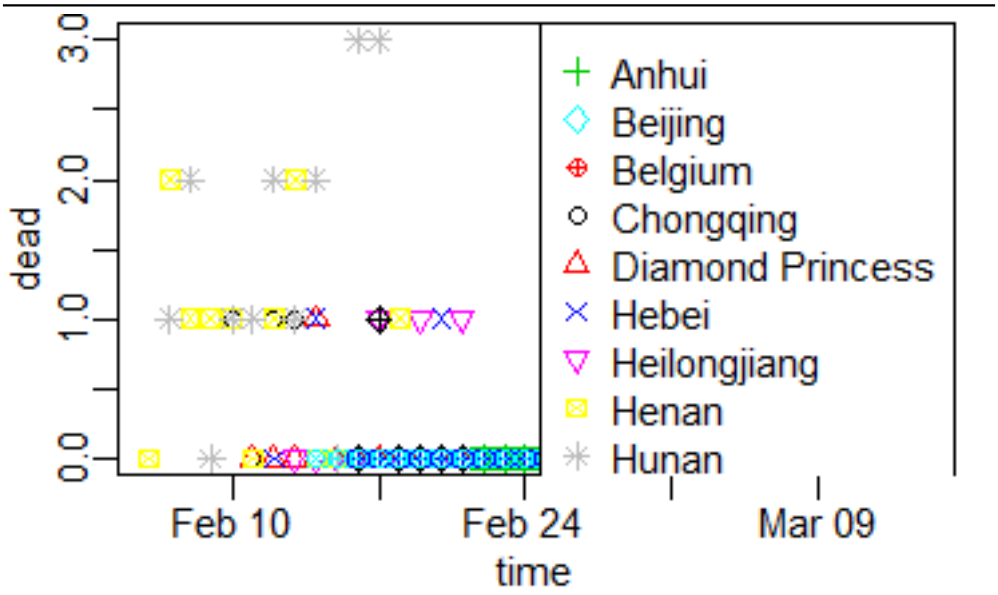
表 4: {#tbl:unnamed-chunk-26}




a)

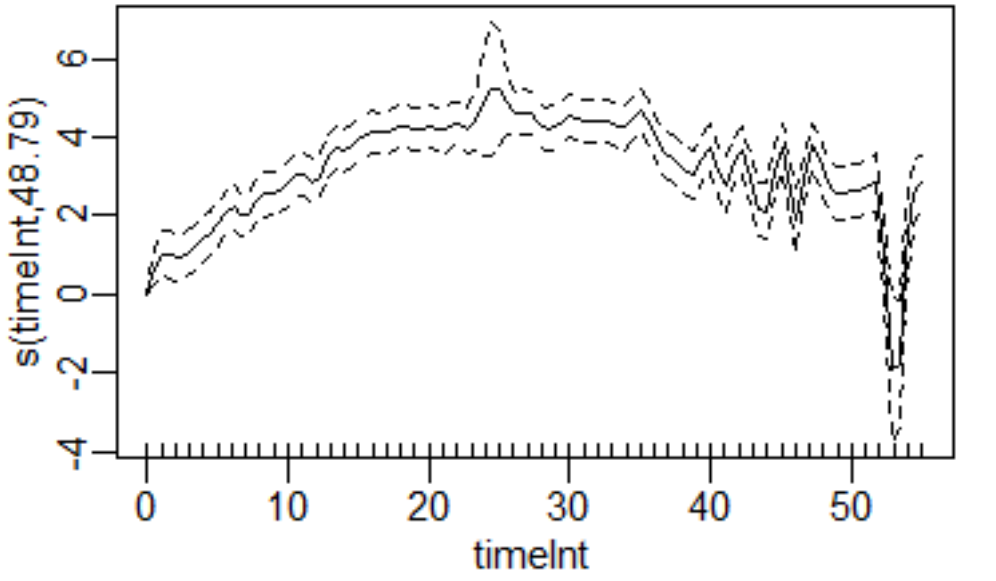
```
Warning in max(which(xx > 0)): no non-missing arguments to max; returning -Inf

Warning in max(which(xx > 0)): no non-missing arguments to max; returning -Inf

Warning in max(which(xx > 0)): no non-missing arguments to max; returning -Inf

Warning in max(which(xx > 0)): no non-missing arguments to max; returning -Inf

Warning in max(which(xx > 0)): no non-missing arguments to max; returning -Inf

Warning in max(which(xx > 0)): no non-missing arguments to max; returning -Inf

Warning in max(which(xx > 0)): no non-missing arguments to max; returning -Inf
```

表 5: {#tbl:unnamed-chunk-27}



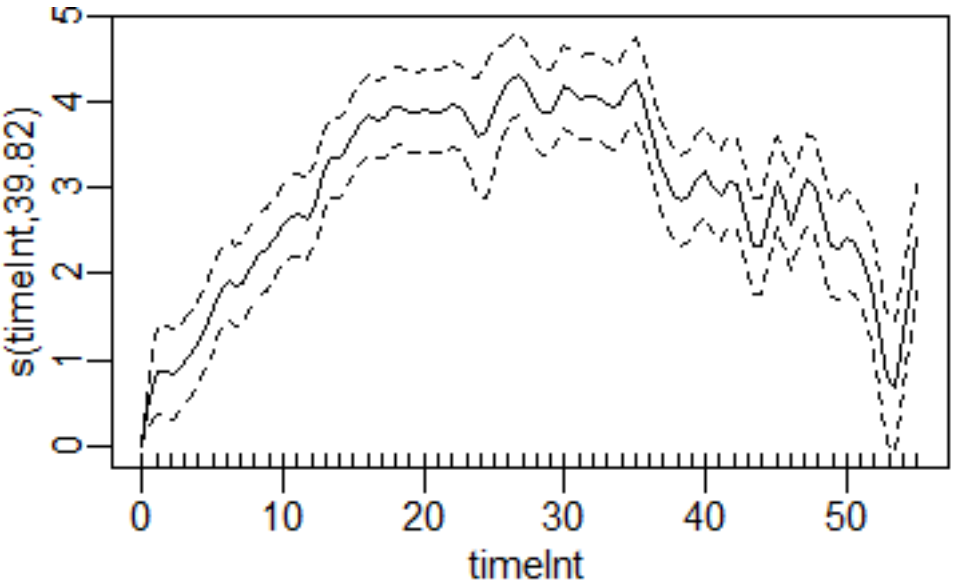

a)

表 6: {#tbl:unnamed-chunk-28}



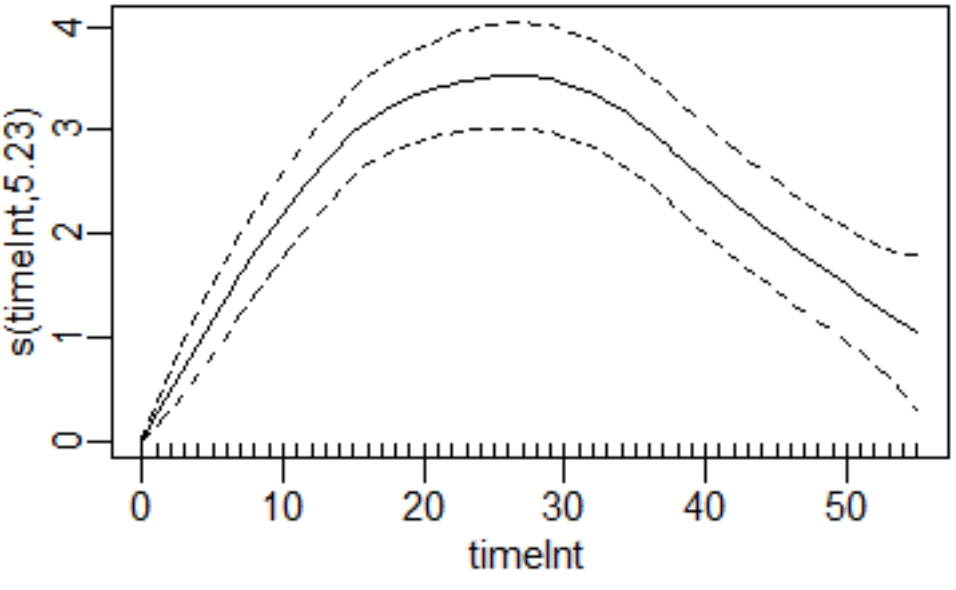

a)

表 7: {#tbl:unnamed-chunk-29}




a)

表 8: {#tbl:unnamed-chunk-31}



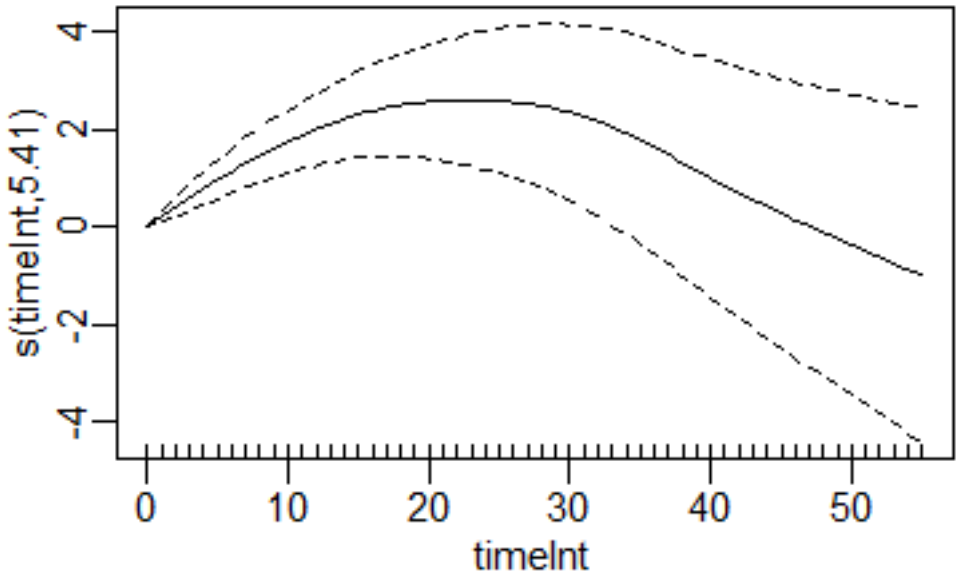

a)

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: poisson  ( log )
```

```
     AIC      BIC   logLik deviance df.resid
  1360.5   1444.7   -657.2   1314.5      265

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.3685 -0.4019 -0.1308  0.1384 10.6837

Random effects:
 Groups                 Name        Variance Std.Dev.
 timeIntInd:countryFac (Intercept) 0.242    0.492
 Xr                     s(timeInt)  5.705    2.388
Number of obs: 288, groups:  timeIntInd:countryFac, 288; Xr, 28

Fixed effects:
                         Estimate Std. Error z value Pr(>|z|)
X(Intercept)              -3.0198     1.0186  -2.965 0.003031 **
XcountryFacFrance          3.5030     1.0274   3.409 0.000651 ***
XcountryFacGermany         3.1483     1.1480   2.743 0.006097 **
XcountryFacGuangdong      -1.7514     1.1387  -1.538 0.124033
XcountryFacHainan         -2.4656     1.1743  -2.100 0.035761 *
XcountryFacHong Kong       0.5421     1.4400   0.376 0.706591
XcountryFacHubei           4.2170     1.0175   4.144 3.41e-05 ***
XcountryFacIran            3.8174     1.0182   3.749 0.000177 ***
XcountryFacIraq            1.4753     1.1027   1.338 0.180925
XcountryFacItaly           4.4858     1.0177   4.408 1.04e-05 ***
XcountryFacJapan           1.0311     1.0504   0.982 0.326298
XcountryFacNetherlands     2.6580     1.1065   2.402 0.016302 *
XcountryFacPhilippines     3.1550     1.2260   2.573 0.010069 *
XcountryFacSan Marino      2.7650     1.2913   2.141 0.032251 *
XcountryFacShandong       -1.5787     1.1384  -1.387 0.165525
XcountryFacSouth Korea     2.0663     1.0256   2.015 0.043945 *
XcountryFacSpain           4.7081     1.0280   4.580 4.65e-06 ***
XcountryFacSwitzerland     3.1937     1.1031   2.895 0.003790 **
XcountryFacUnited Kingdom  3.5799     1.0654   3.360 0.000779 ***
XcountryFacUnited States   3.0706     1.0317   2.976 0.002918 **
Xs(timeInt)Fx1             3.4576     0.8210   4.211 2.54e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Correlation matrix not shown by default, as p = 21 > 12.
Use print(x, correlation=TRUE)  or
    vcov(x)        if you need it
```

表 9: {#tbl:unnamed-chunk-32}

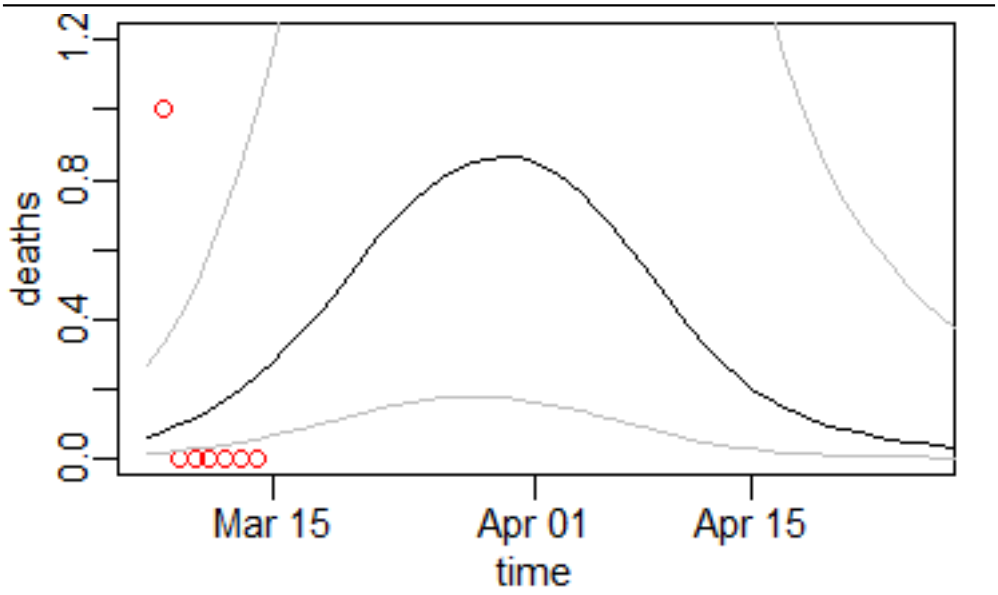



a)

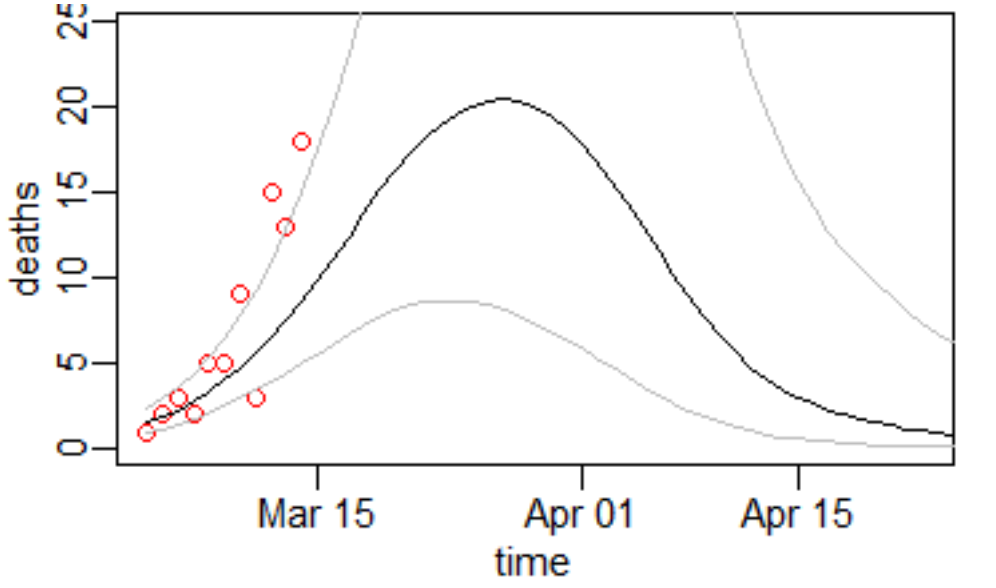
```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: poisson  ( log )

     AIC      BIC   logLik deviance df.resid
  1334.9   1422.8   -643.5   1286.9      264

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.6061 -0.4135 -0.2154  0.1633  4.0676

Random effects:
 Groups                Name        Variance Std.Dev.
 timeIntInd:countryFac (Intercept)  0.1749  0.4182
 Xr                    s(timeInt)   5.5081  2.3469
 countryFac            timeSlope   70.1469  8.3754
Number of obs: 288, groups:  timeIntInd:countryFac, 288; Xr, 28; countryFac, 20
```

```
Fixed effects:
                          Estimate Std. Error z value Pr(>|z|)
X(Intercept)               -2.7531     1.0715  -2.569 0.010189 *
XcountryFacFrance           3.1594     1.1220   2.816 0.004864 **
XcountryFacGermany          2.9580     1.1834   2.499 0.012438 *
XcountryFacGuangdong        0.3765     1.3772   0.273 0.784545
XcountryFacHainan          -0.6097     1.4972  -0.407 0.683857
XcountryFacHong Kong        0.3995     1.4844   0.269 0.787830
XcountryFacHubei            3.9758     1.0859   3.661 0.000251 ***
XcountryFacIran             3.3925     1.0922   3.106 0.001896 **
XcountryFacIraq             1.5570     1.1952   1.303 0.192667
XcountryFacItaly            3.1394     1.0996   2.855 0.004304 **
XcountryFacJapan            1.1799     1.2096   0.975 0.329327
XcountryFacNetherlands      2.4593     1.1615   2.117 0.034233 *
XcountryFacPhilippines      2.9450     1.2515   2.353 0.018615 *
XcountryFacSan Marino       2.5500     1.3154   1.939 0.052554 .
XcountryFacShandong         0.4035     1.3761   0.293 0.769378
XcountryFacSouth Korea      3.1446     1.1049   2.846 0.004428 **
XcountryFacSpain            4.4018     1.1042   3.986 6.71e-05 ***
XcountryFacSwitzerland      2.9858     1.1496   2.597 0.009399 **
XcountryFacUnited Kingdom   3.3829     1.1219   3.015 0.002567 **
XcountryFacUnited States    3.2529     1.1164   2.914 0.003572 **
Xs(timeInt)Fx1              2.7221     0.9077   2.999 0.002708 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation matrix not shown by default, as p = 21 > 12.
Use print(x, correlation=TRUE)  or
    vcov(x)        if you need it

[1] "timeIntInd:countryFac" "Xr"                    "countryFac"

     Guangdong       Shandong    South Korea         Hainan  United States
   -10.7324071    -10.6298566     -7.1541950     -6.5594259     -3.1933197
          Iraq      Australia        Germany     San Marino      Hong Kong
    -2.3876160     -2.2672344      0.3965315      0.5833377      0.5940711
   Philippines          Japan    Switzerland    Netherlands United Kingdom
     0.8131890      1.0539405      2.2961327      2.9570253      2.9775866
         Hubei          Spain           Iran         France          Italy
     3.7853795      5.2770467      5.6250033      5.7560949     14.6912983
```

表 10: {#tbl:unnamed-chunk-34}

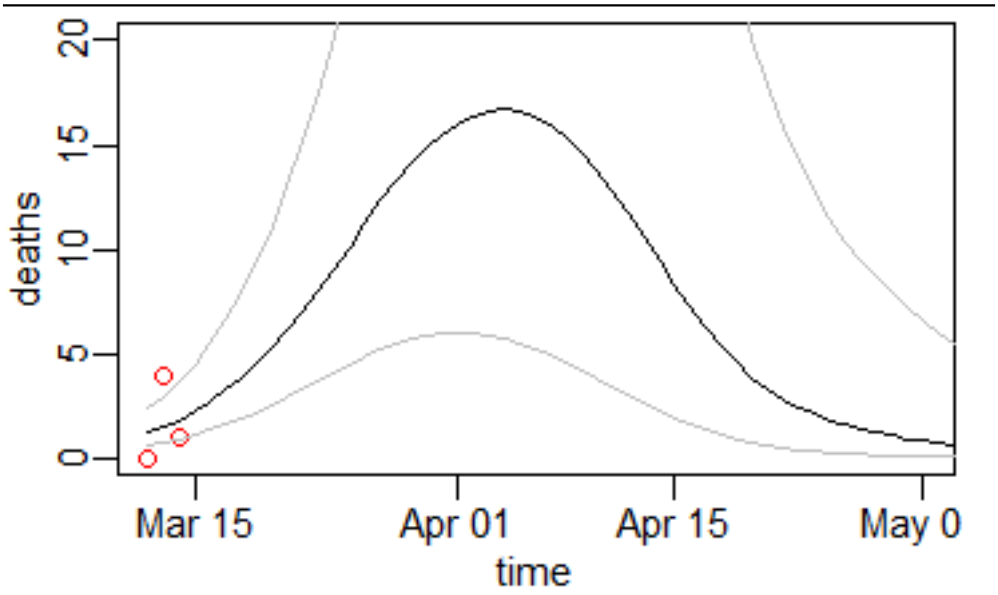



a)

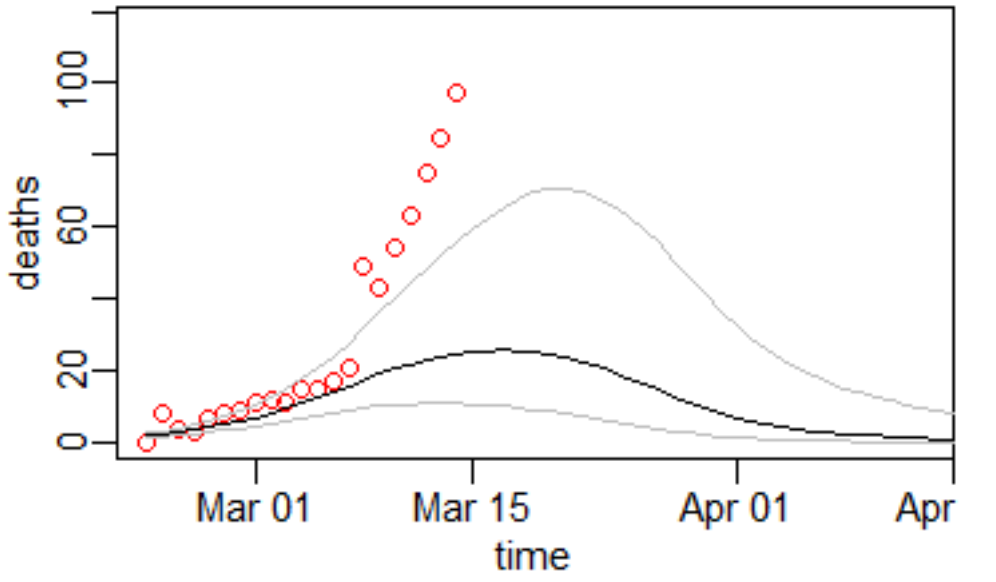



b)

表 10: {#tbl:unnamed-chunk-34}

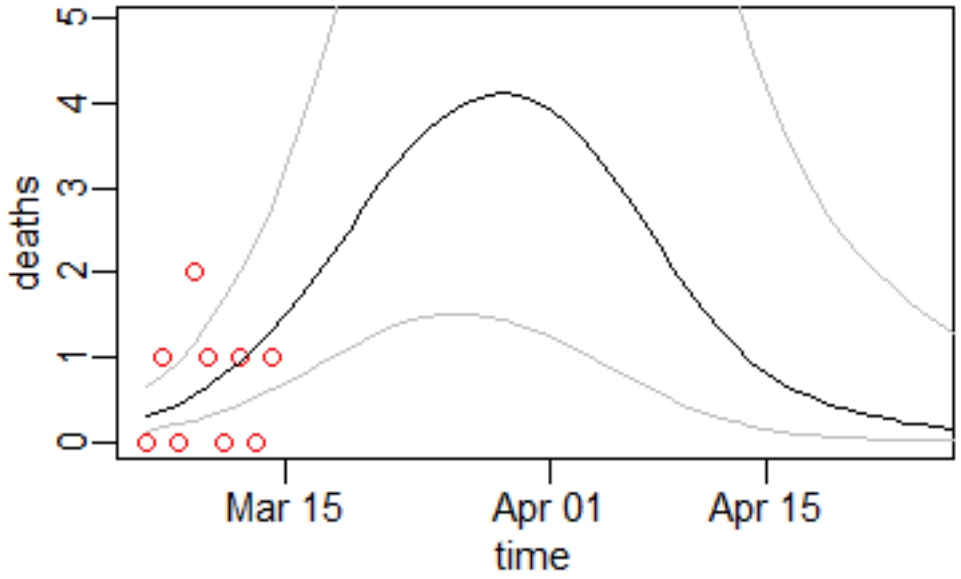



c)

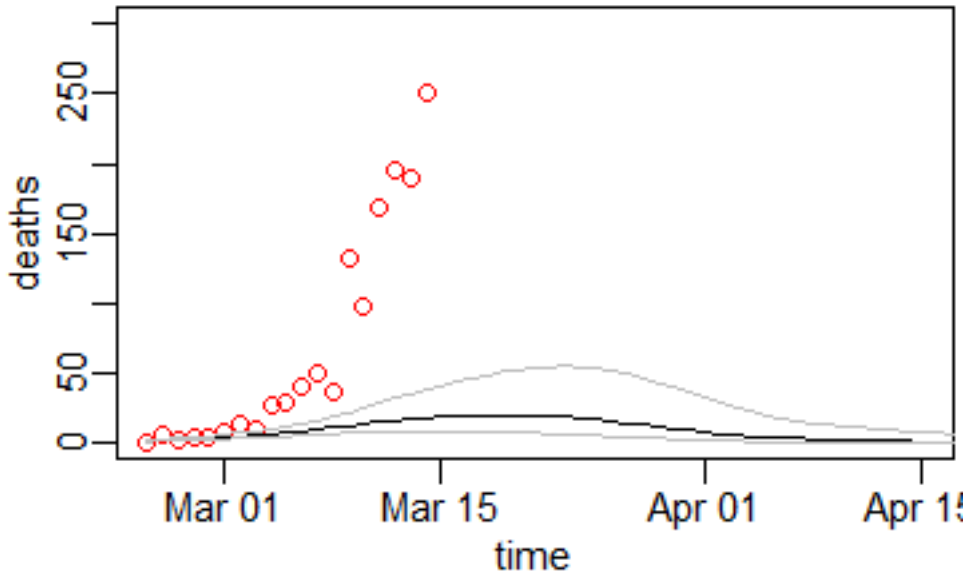



d)

表 10: {#tbl:unnamed-chunk-34}

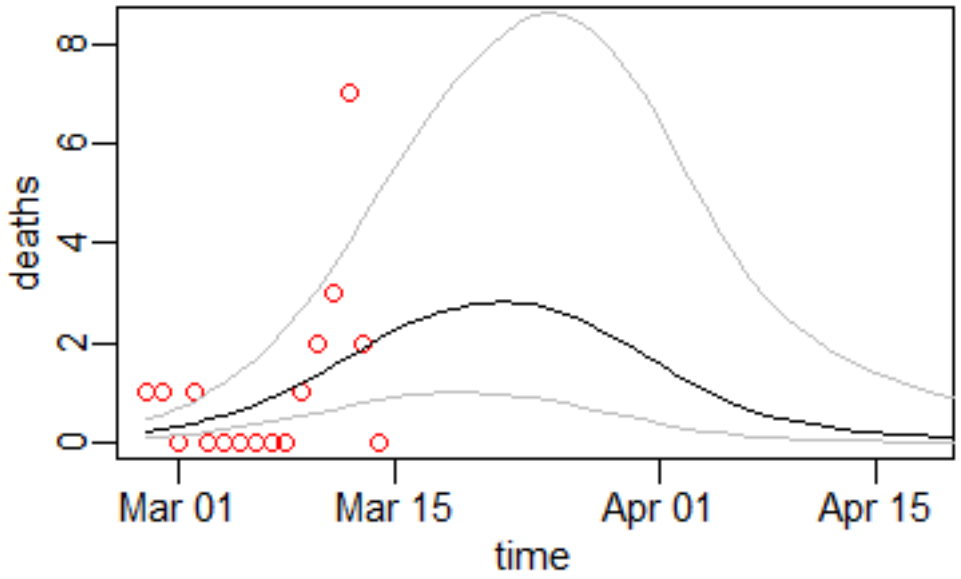



e)

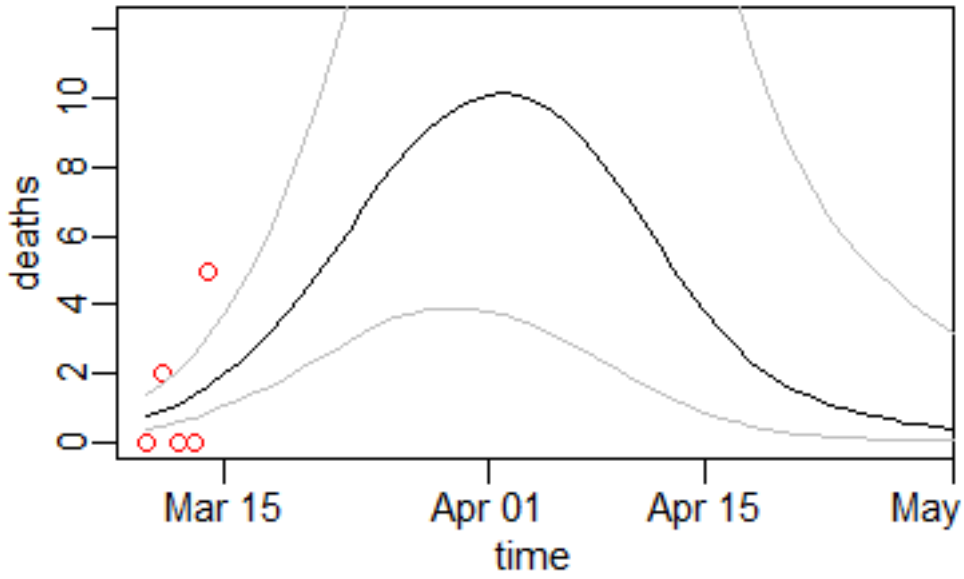



f)

表 10: {#tbl:unnamed-chunk-34}

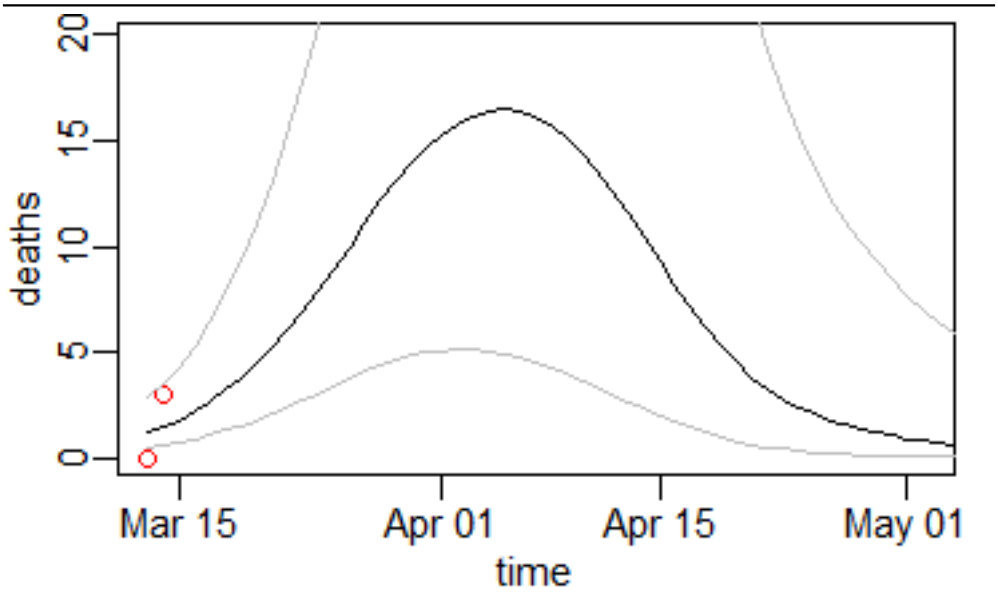



g)

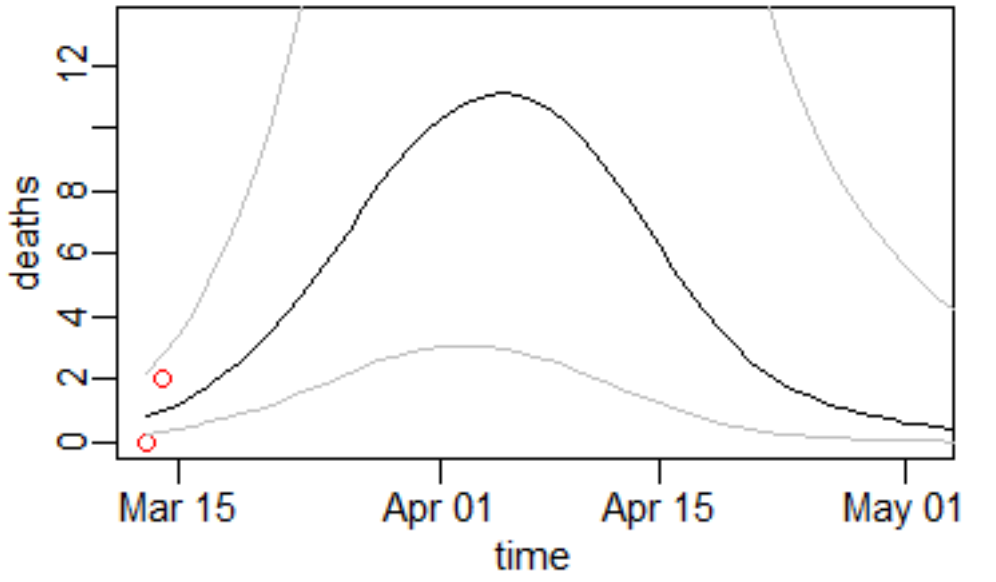



h)

表 10: {#tbl:unnamed-chunk-34}

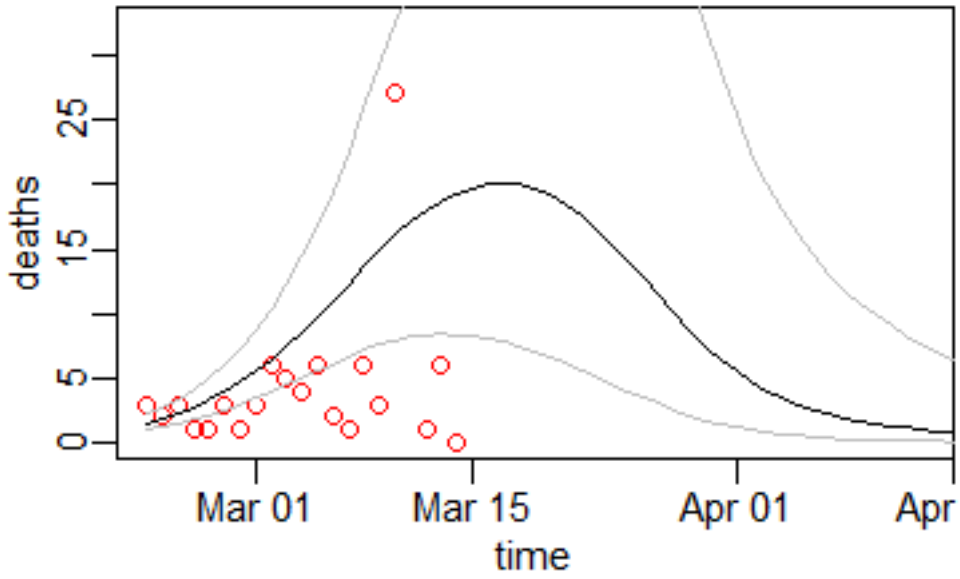



i)

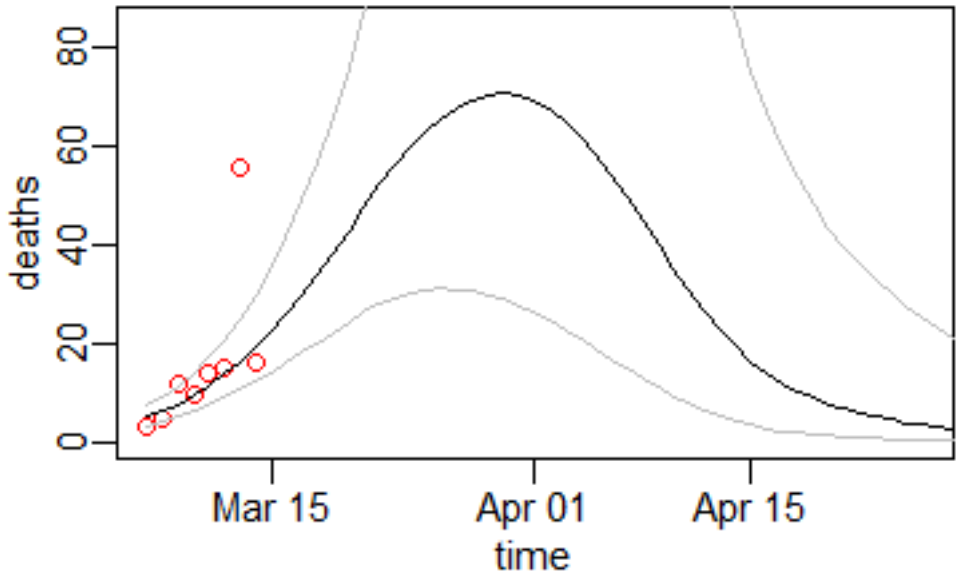



j)

表 10: {#tbl:unnamed-chunk-34}

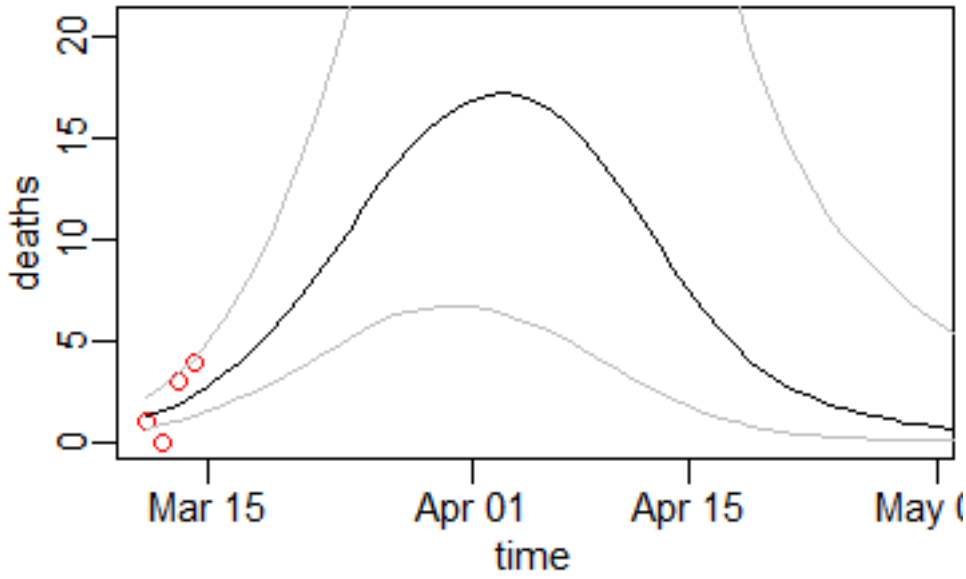



k)

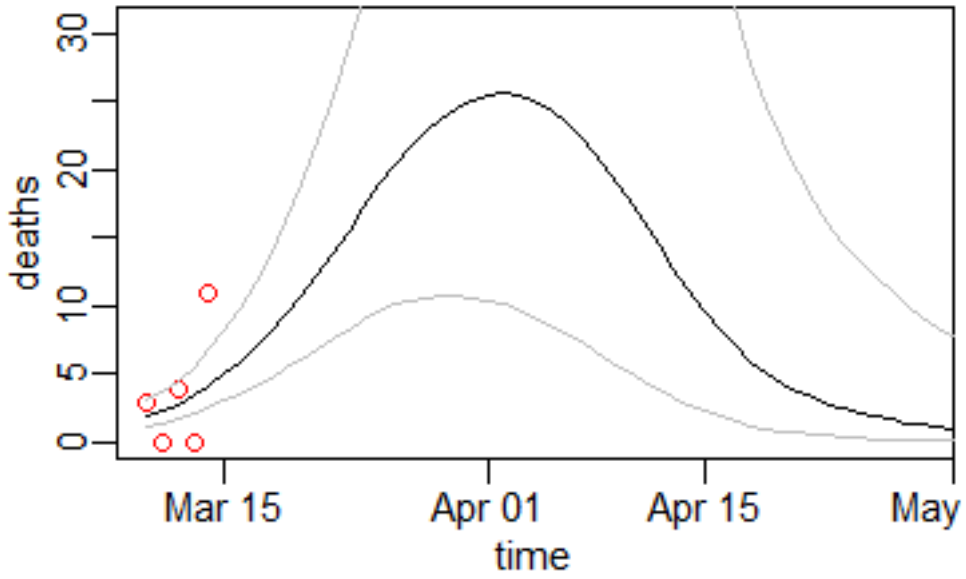



l)

表 10: {#tbl:unnamed-chunk-34}

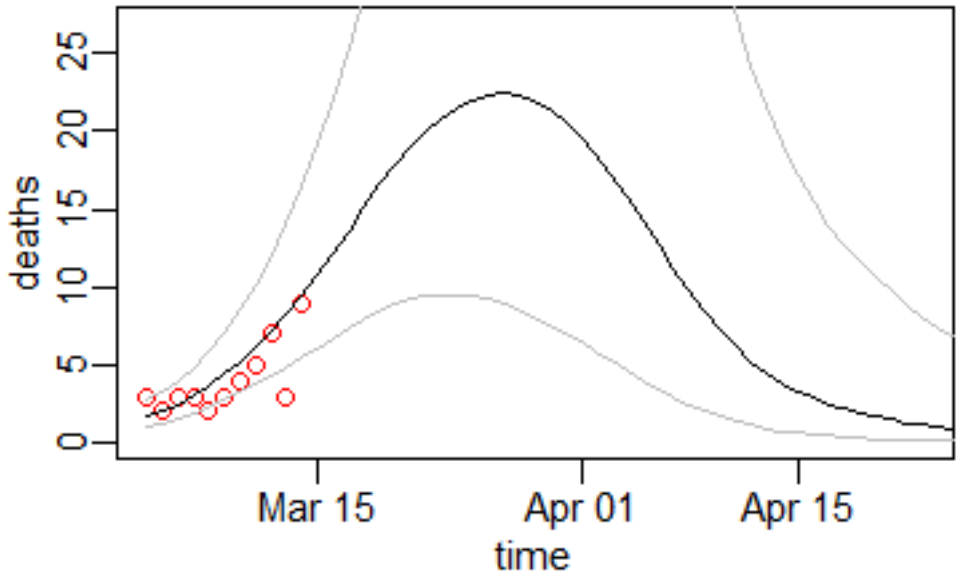



m)

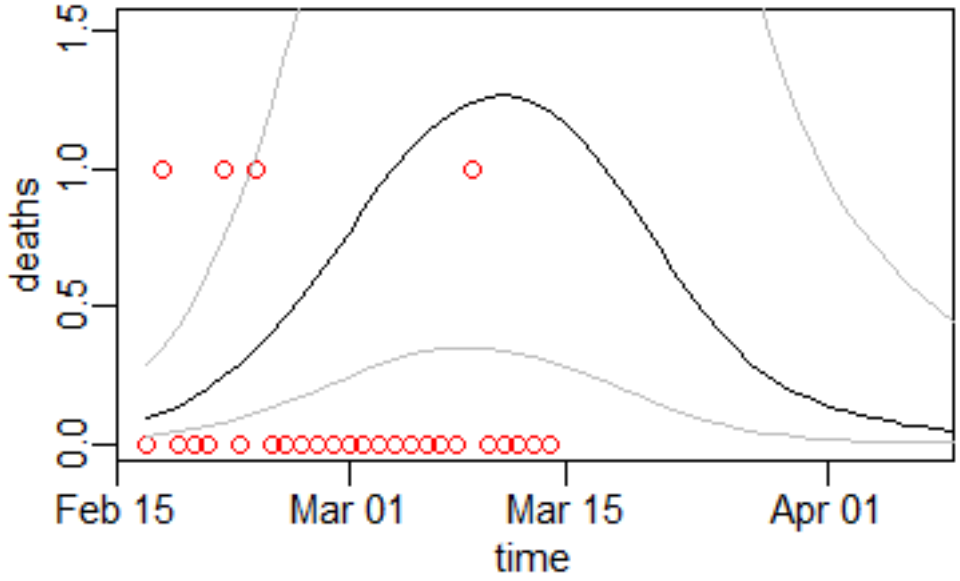



n)

表 10: {#tbl:unnamed-chunk-34}

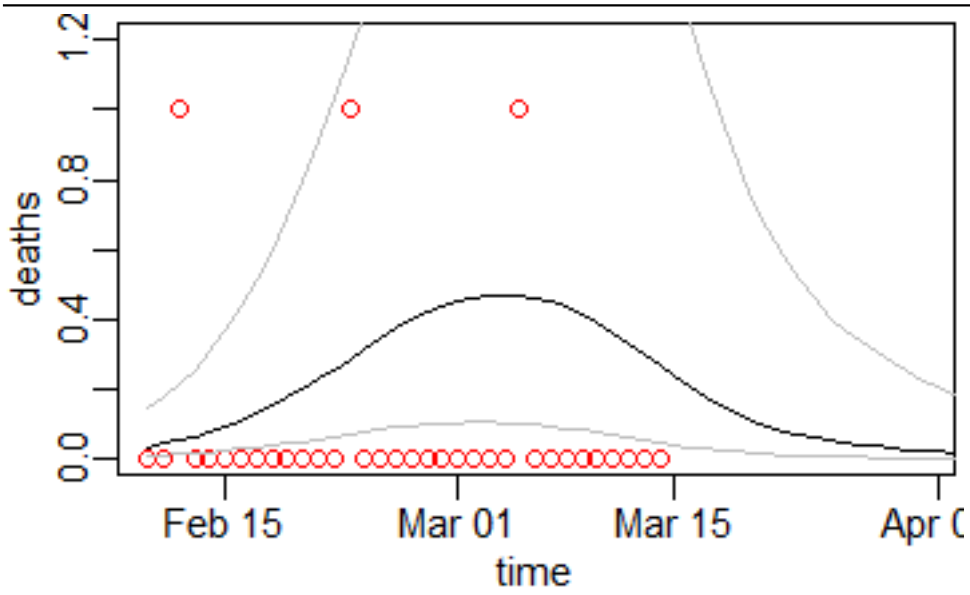



o)

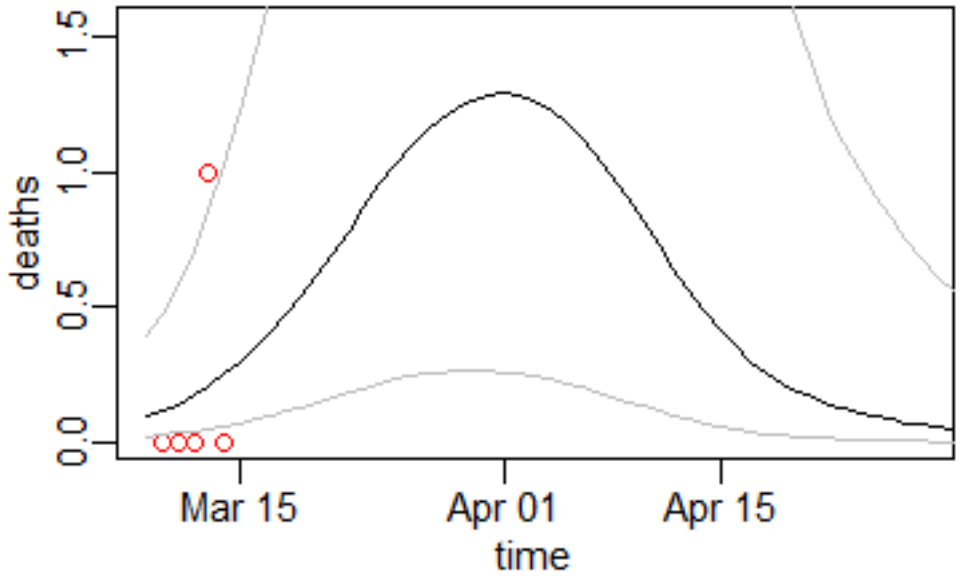



p)

表 10: {#tbl:unnamed-chunk-34}

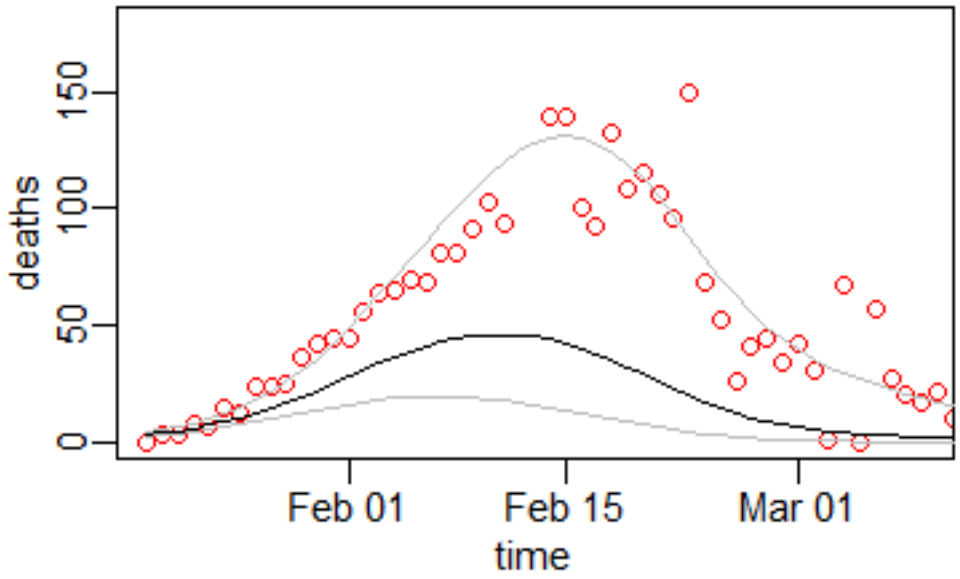



q)

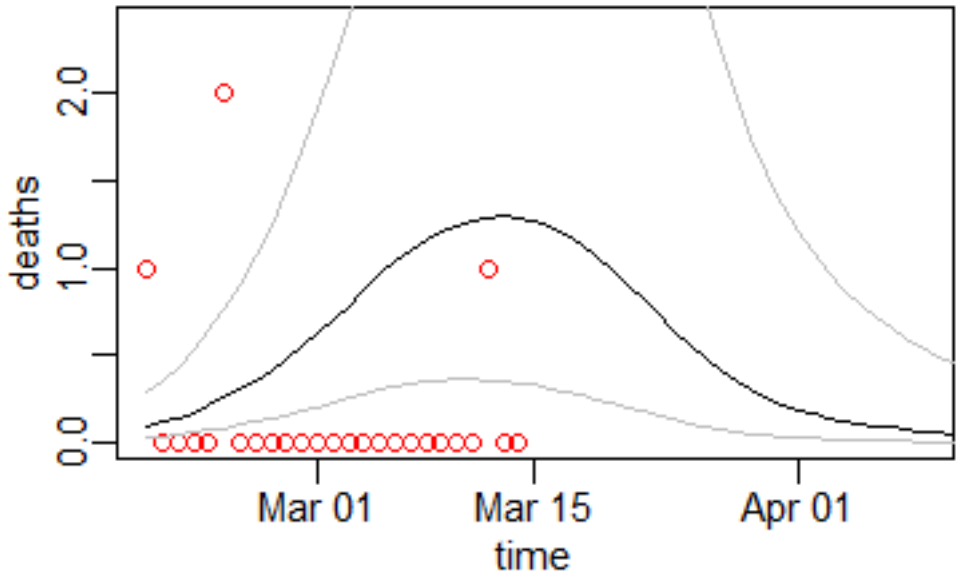



r)

表 10: {#tbl:unnamed-chunk-34}




s)




t)