COVID-19 Death Counts Prediction in Italy and Hubei

Timothy Lee, credits to Prof Patrick Brown and Liza Bolton (University of Toronto) for starter code

Saturday 11 April 2020

1 Context

This data analysis report aims to predict the number of COVID-19 related deaths for Hubei and Italy using GAMM.

```r
library(devtools)
library(mgcv)
library(gamm4)
library(tidyverse)
```

1.1 Preliminary Analysis

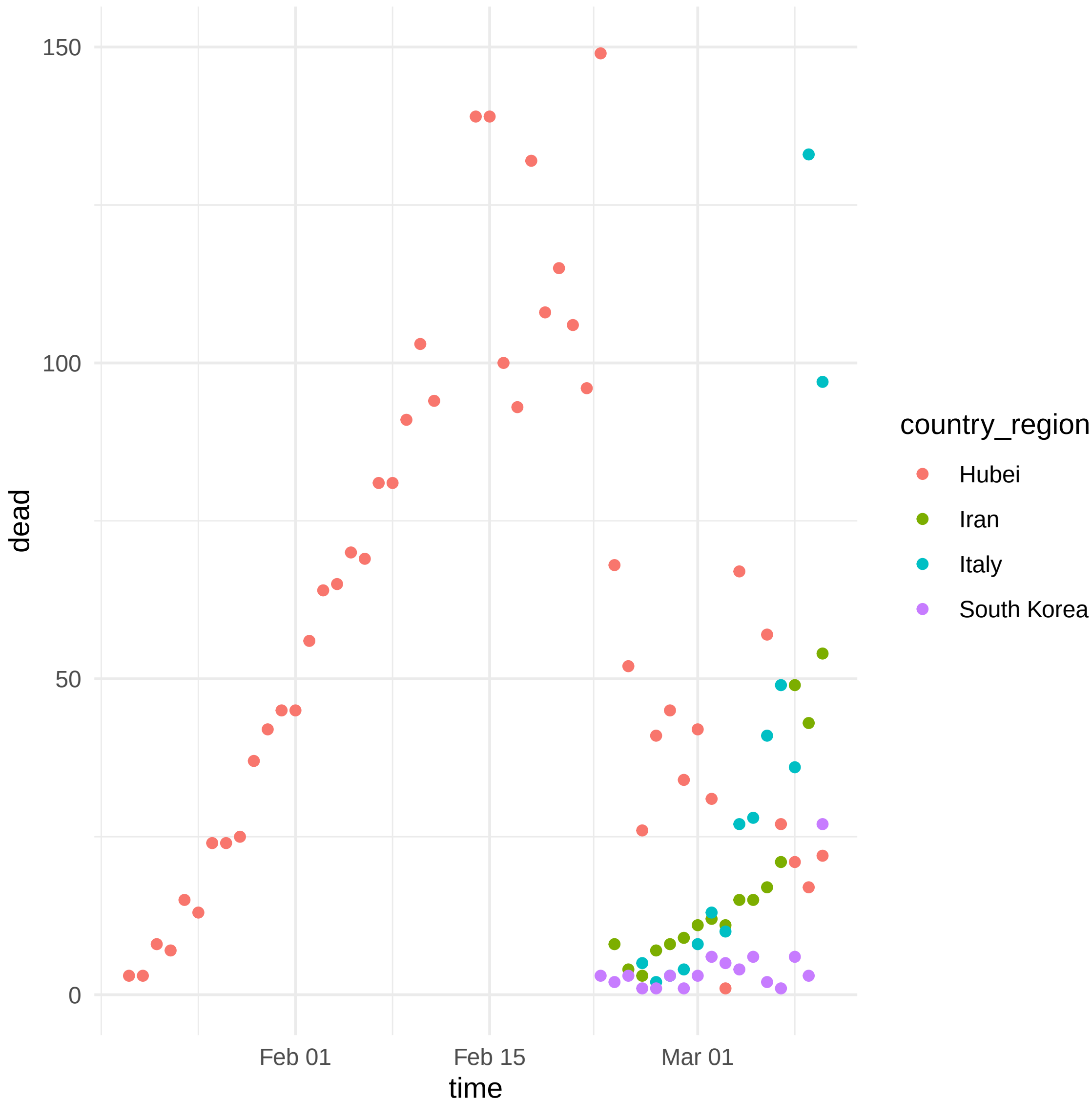
2 nCOVID-19 data

Plot deaths from nCOVID-19

```r
# Load nCOVID-19 data
covid_data <- read_csv("covid_data.csv")
```

3 Plot over time

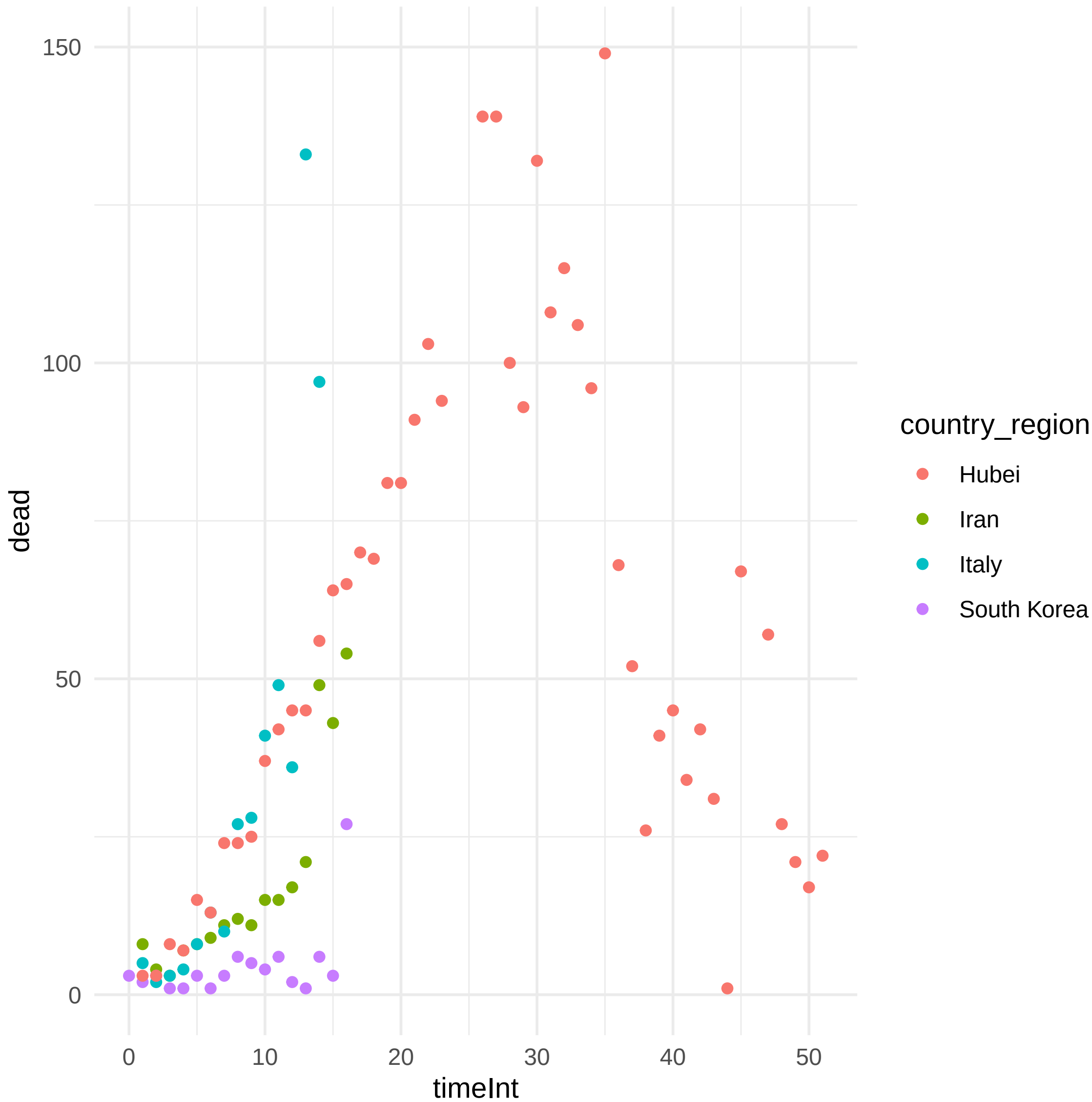
```r
covid_data %>%
  filter(country_region %in% c('Hubei','Italy','Iran','South Korea','USA')) %>%
  na.omit() %>%
  ggplot(aes(time, dead, color=country_region)) +
  geom_point() +
  theme_minimal()
```




```
# height <- 7
```

4 Plot from initial death in region

```
covid_data %>%
  filter(country_region %in% c('Hubei','Italy','Iran','South Korea','USA')) %>%
  na.omit() %>%
  ggplot(aes(timeInt, dead, color=country_region)) +
  geom_point() +
  theme_minimal()
```



Fit a GAM with `dead` as the response a smooth on `timeInt` and `country_region` as covariate. In the smooth, use `pc=0`, which indicates a *point constraint*. The smooth will pass through 0 at this point. `timeInt` indicates time since the first death, so the line should predict no deaths before any deaths occured.

```
resGam= mgcv::gam(
  dead ~ s(timeInt, pc=0) + country_region,
  data=covid_data,
  family=poisson(link='log'))
```

Summary Tables

```
summary(resGam)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## dead ~ s(timeInt, pc = 0) + country_region
##
```

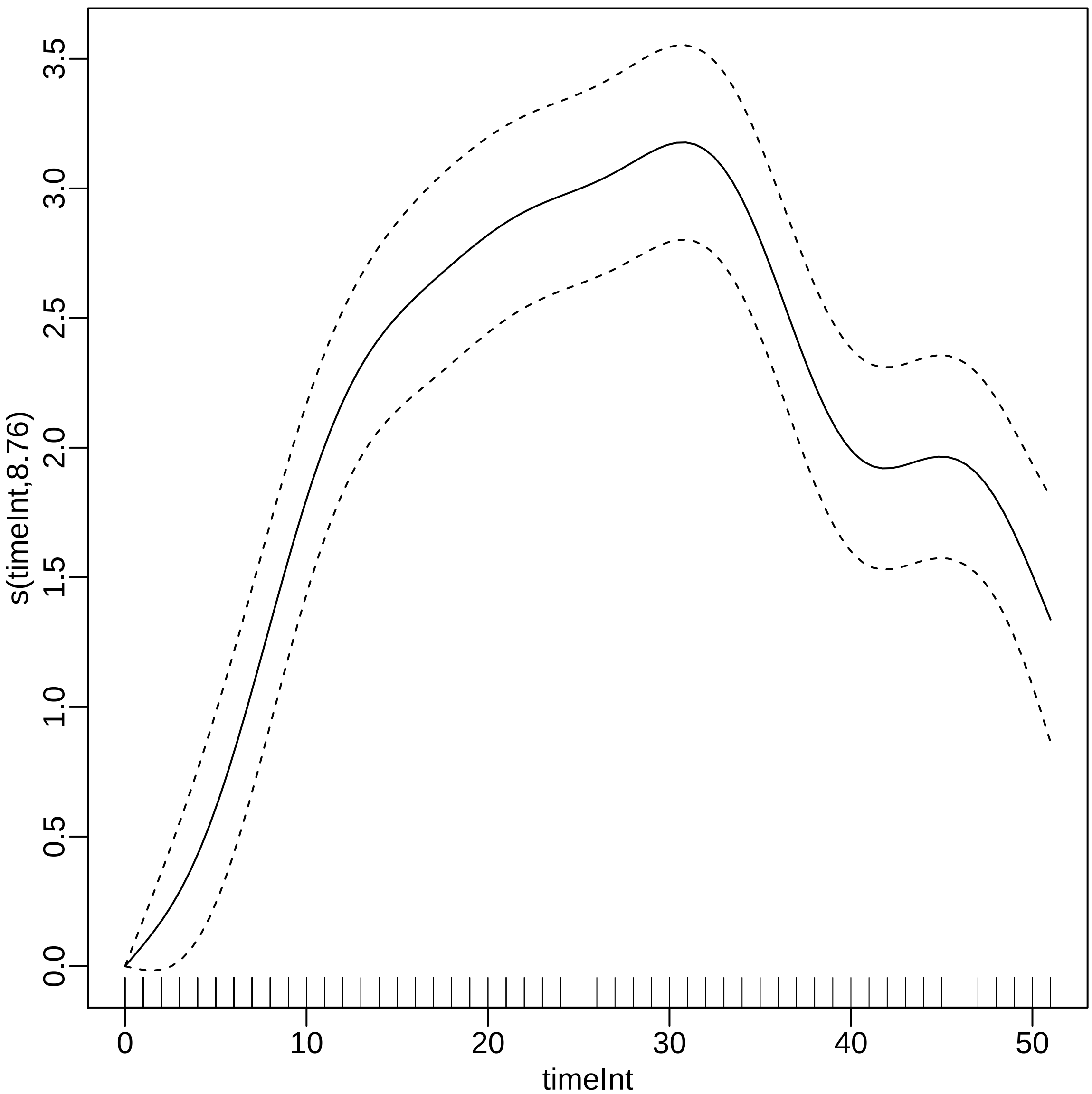
```
## Parametric coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -0.160352   0.583136  -0.275 0.783329
## country_regionAustralia       0.078106   1.155196   0.068 0.946094
## country_regionBeijing        -1.940556   0.739512  -2.624 0.008688 **
## country_regionChongqing      -0.535153   0.819679  -0.653 0.513833
## country_regionFrance          1.127419   0.610845   1.846 0.064940 .
## country_regionGuangdong      -1.608135   0.771882  -2.083 0.037215 *
## country_regionHainan         -2.168937   0.824279  -2.631 0.008506 **
## country_regionHebei          -0.763389   0.823787  -0.927 0.354092
## country_regionHeilongjiang   -1.118993   0.666038  -1.680 0.092943 .
## country_regionHenan          -1.208796   0.631050  -1.916 0.055425 .
## country_regionHubei           1.815819   0.589066   3.083 0.002052 **
## country_regionHunan           0.078106   1.155196   0.068 0.946094
## country_regionIran            1.321243   0.590201   2.239 0.025180 *
## country_regionIraq            0.171690   0.764797   0.224 0.822375
## country_regionItaly           2.117238   0.588802   3.596 0.000323 ***
## country_regionJapan          -1.361864   0.654921  -2.079 0.037578 *
## country_regionShandong        0.215099   0.817422   0.263 0.792440
## country_regionSouth Korea    -0.005497   0.597876  -0.009 0.992664
## country_regionSpain           2.033865   0.605583   3.359 0.000784 ***
## country_regionUnited Kingdom  1.258965   0.820598   1.534 0.124979
## country_regionUnited States   0.827315   0.621365   1.331 0.183042
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##               edf Ref.df Chi.sq p-value
## s(timeInt) 8.758  8.982   1309  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.894   Deviance explained = 93.5%
## UBRE = 2.0019  Scale est. = 1         n = 170
```

```r
coef(resGam)
```

```
##                  (Intercept)      country_regionAustralia
##                 -0.160352456                  0.078105515
##        country_regionBeijing      country_regionChongqing
##                 -1.940556292                 -0.535153159
##         country_regionFrance      country_regionGuangdong
##                  1.127419488                 -1.608135374
##         country_regionHainan          country_regionHebei
##                 -2.168937066                 -0.763389041
##   country_regionHeilongjiang          country_regionHenan
##                 -1.118993096                 -1.208796089
##          country_regionHubei          country_regionHunan
##                  1.815818734                  0.078105515
##           country_regionIran           country_regionIraq
##                  1.321243223                  0.171690309
##          country_regionItaly          country_regionJapan
##                  2.117237701                 -1.361864231
##       country_regionShandong    country_regionSouth Korea
##                  0.215099168                 -0.005496802
```

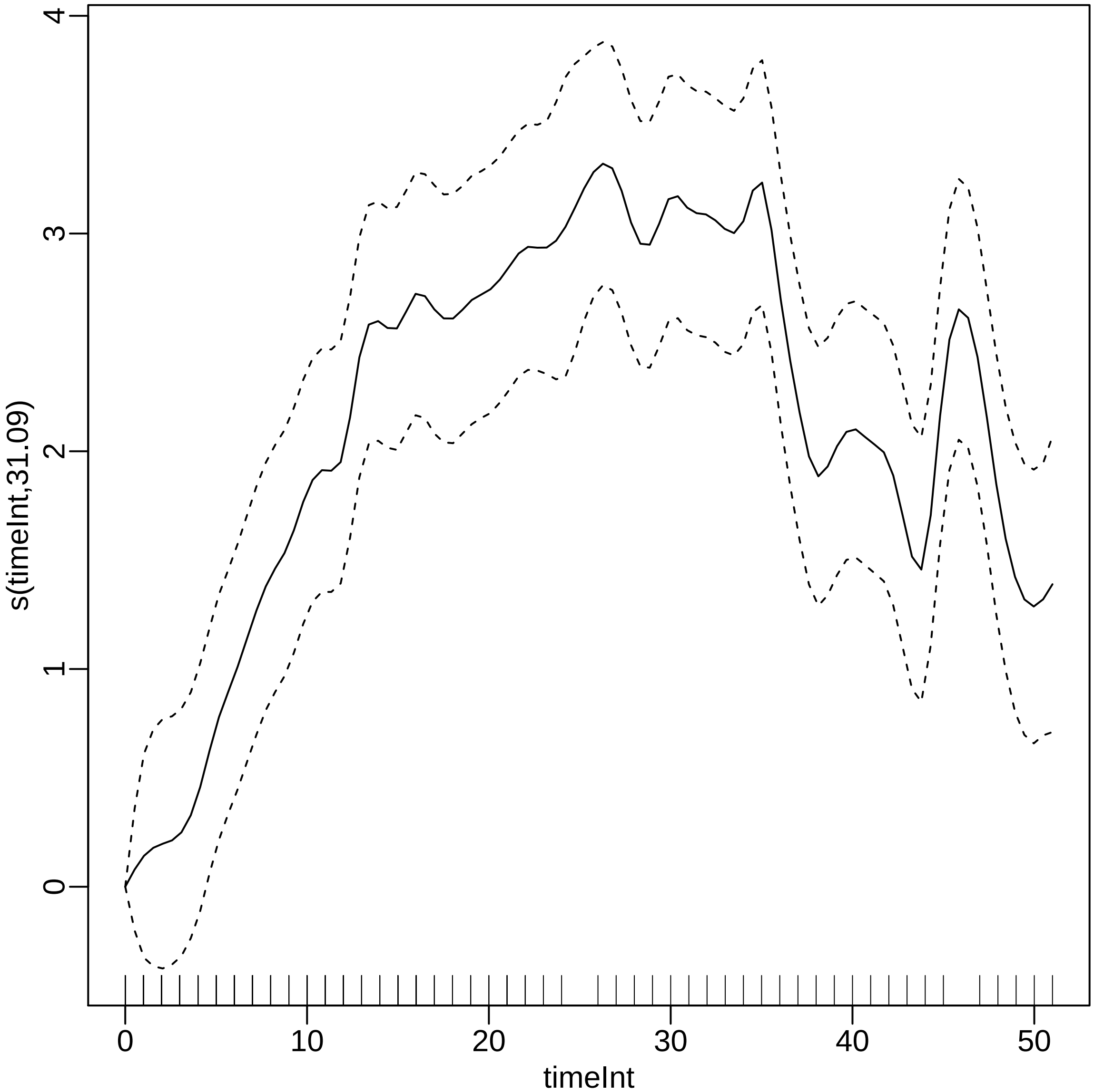
```
##         country_regionSpain country_regionUnited Kingdom
##                 2.033864959                  1.258964745
##  country_regionUnited States                s(timeInt).1
##                 0.827314895                  0.436070190
##                s(timeInt).2                 s(timeInt).3
##                 0.162668721                  0.695995274
##                s(timeInt).4                 s(timeInt).5
##                -0.257405570                  0.133254518
##                s(timeInt).6                 s(timeInt).7
##                 1.140898783                 -0.022139449
##                s(timeInt).8                 s(timeInt).9
##                 4.992873514                 -1.020359041
```

```r
plot(resGam)
```

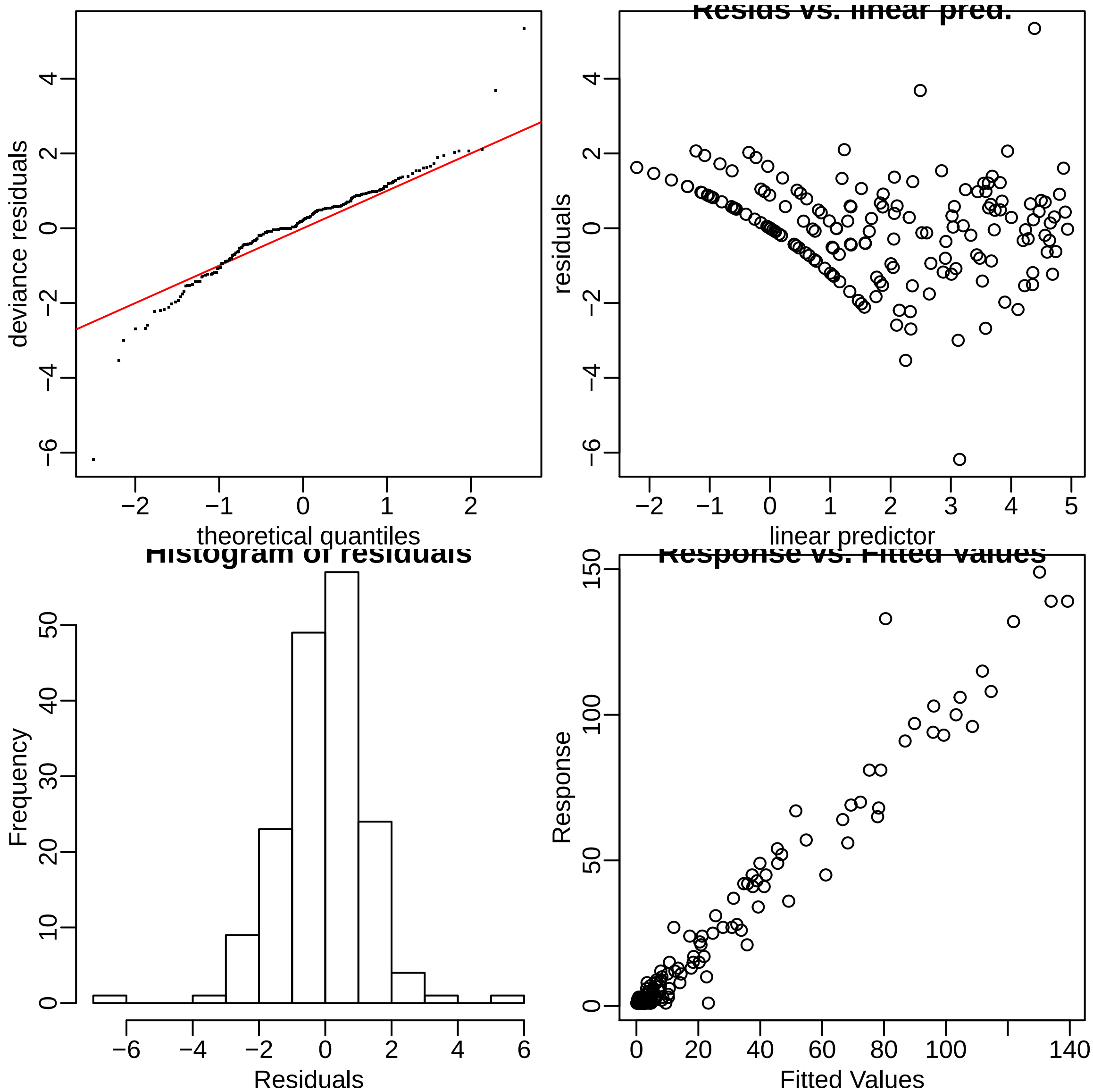


Fit and plot two more GAMs with the same model but with `k = 50` and `k = 20`. Run `gam.check()` for both. Since a higher `k` could lead to overfitting.

```r
resGam3= mgcv::gam(
  dead ~ s(timeInt, k=50, pc=0) + country_region, data=covid_data,
  family=poisson(link='log'), method='ML')
plot(resGam3)
```

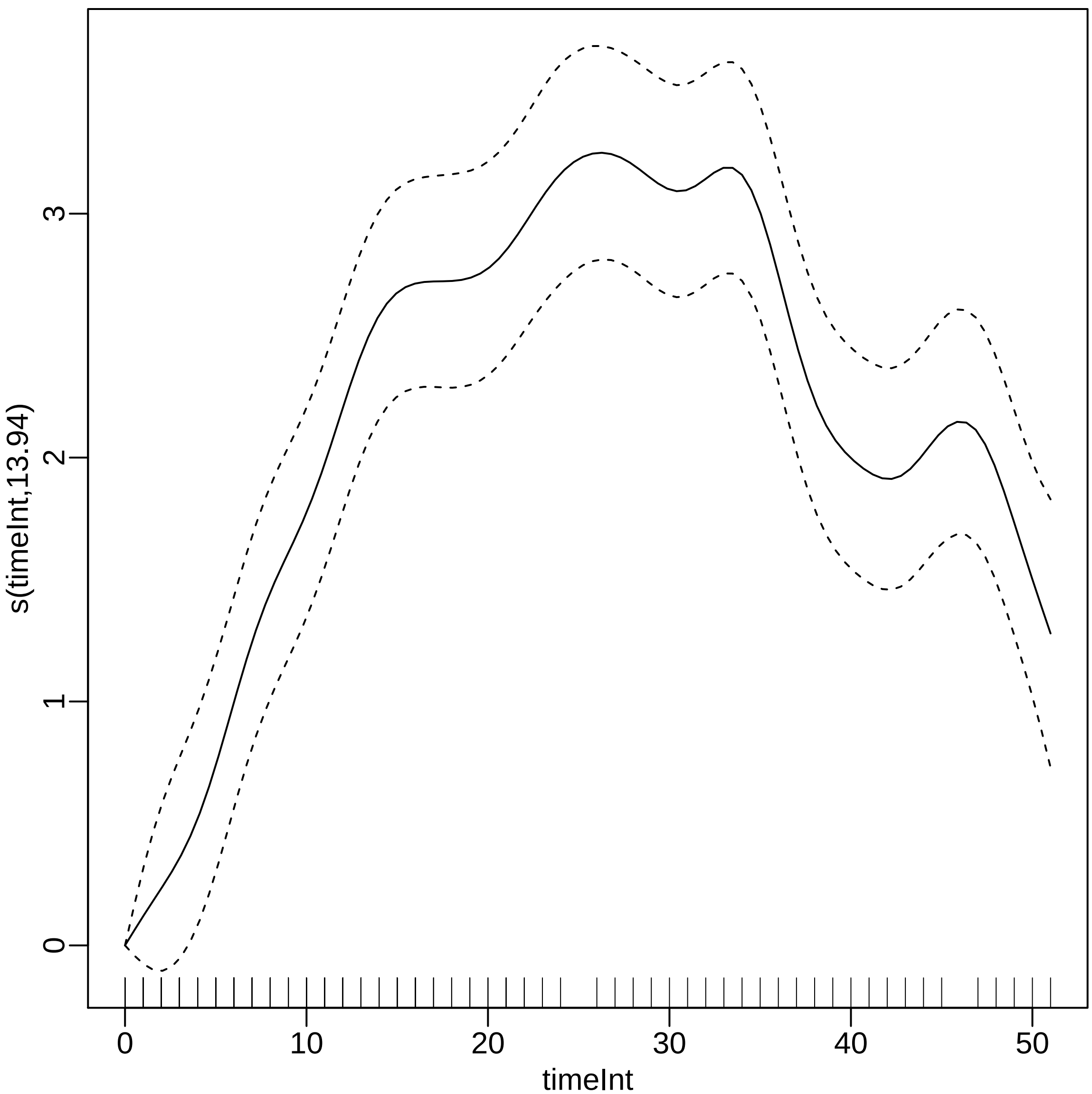



```r
gam.check(resGam3)
```

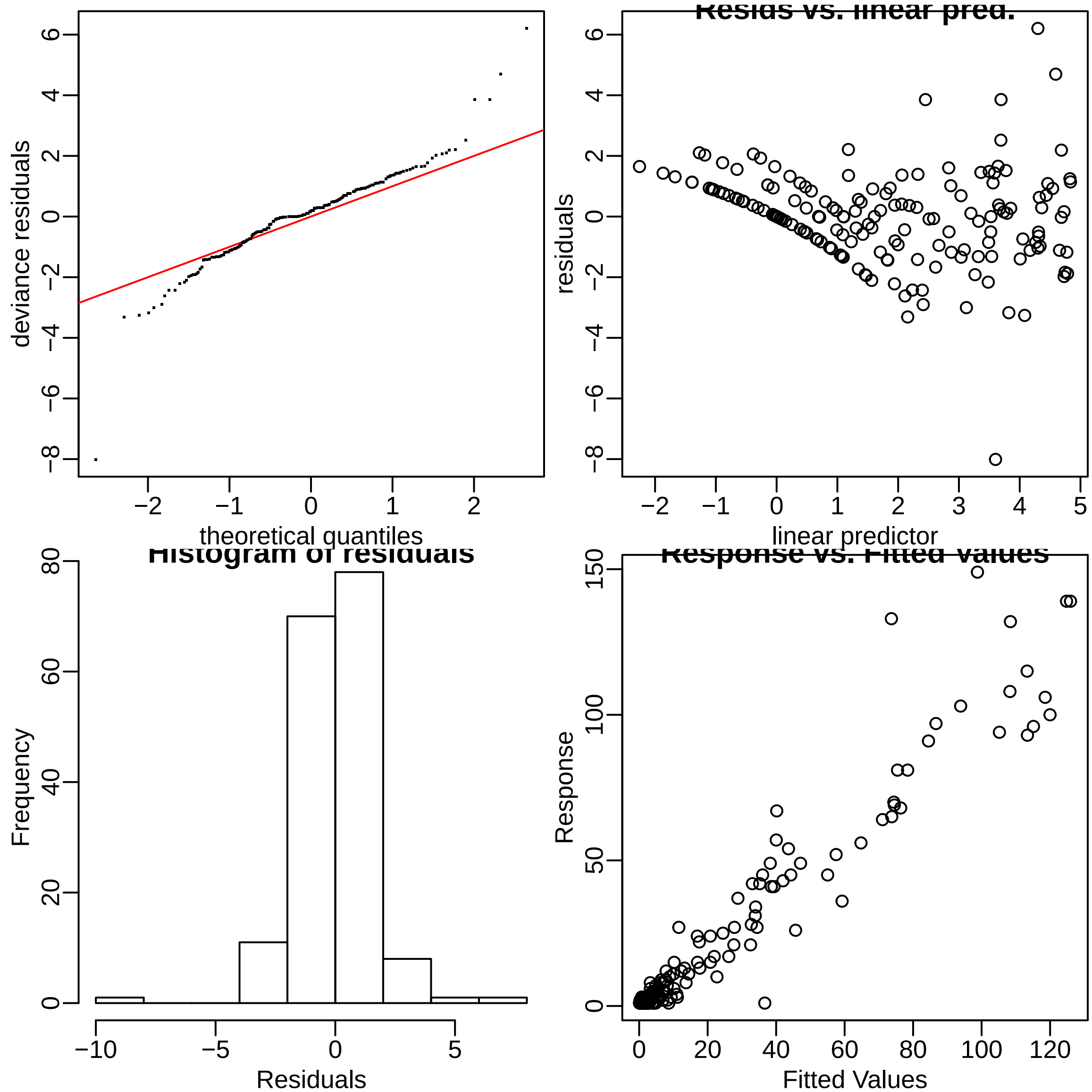



```
##
## Method: ML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-1.704072e-05,-1.704072e-05]
## (score 540.3471 & scale 1).
## Hessian positive definite, eigenvalue range [4.080029,4.080029].
## Model rank =  70 / 70
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##               k'  edf k-index p-value
## s(timeInt) 49.0 31.1    1.25       1
```

```
resGam4 = mgcv::gam(
  dead ~ s(timeInt, k=20, pc=0) + country_region, data=covid_data,
  family=poisson(link='log'), method='ML')
plot(resGam4)
```



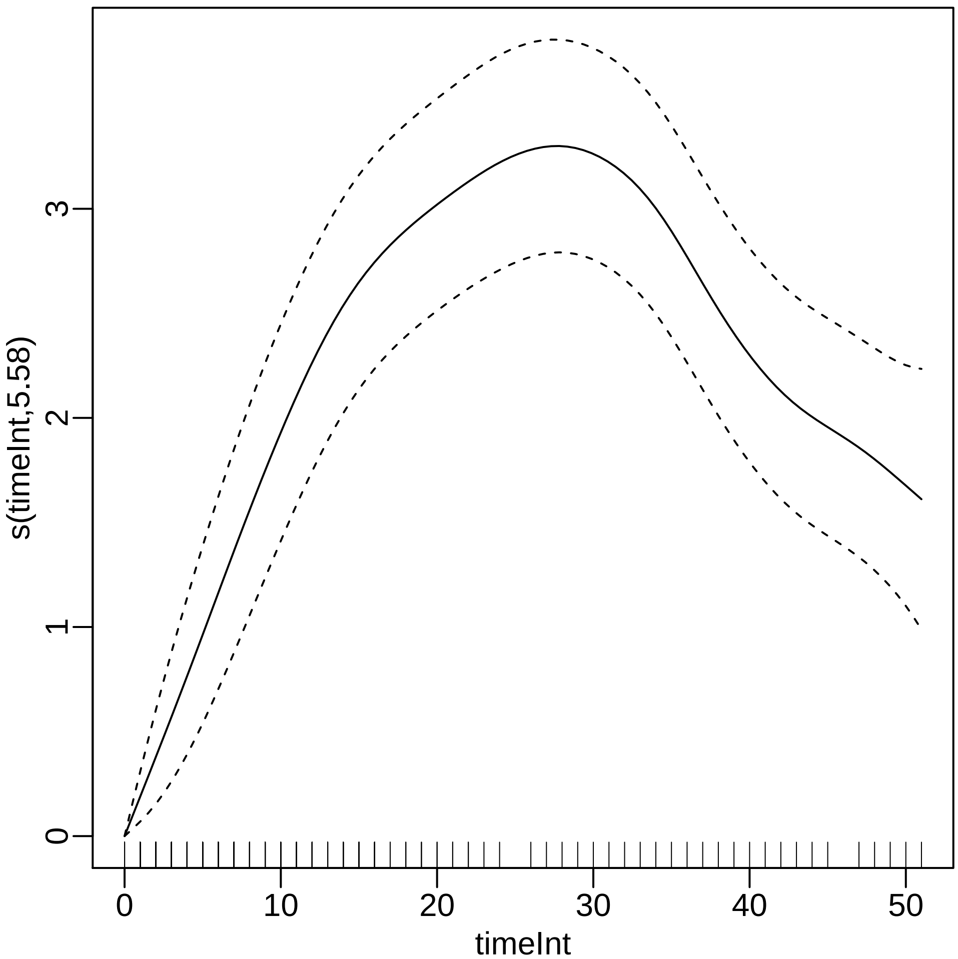

```
gam.check(resGam4)
```




```
##
## Method: ML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [3.691928e-06,3.691928e-06]
## (score 554.3095 & scale 1).
## Hessian positive definite, eigenvalue range [3.724135,3.724135].
## Model rank =  40 / 40
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##               k'  edf k-index p-value
## s(timeInt) 19.0 13.9    1.15    0.96
```

Create a new variable in dataset called `timeIntInd`, which is is just a copy of `timeInt`. Use `gamm4()` to fit the same model as before but additionally with `country_region` nested within `timeIntInd` (since data within countries is likely to highly correlated, so we need to fit a random effect for country).

```r
covid_data$timeIntInd = covid_data$timeInt
resGammInd = gamm4::gamm4(
  dead ~ country_region +
      s(timeInt, k=20, pc=0),
    random = ~ (1|timeIntInd),
    data=covid_data, family=poisson(link='log'))
#extract mer and gam
plot(resGammInd$gam)
```




```r
summary(resGammInd$mer)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
##
##      AIC      BIC   logLik deviance df.resid
##   1082.2   1157.4   -517.1   1034.2      146
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2542 -0.5002  0.0522  0.8694  5.2817
##
## Random effects:
```

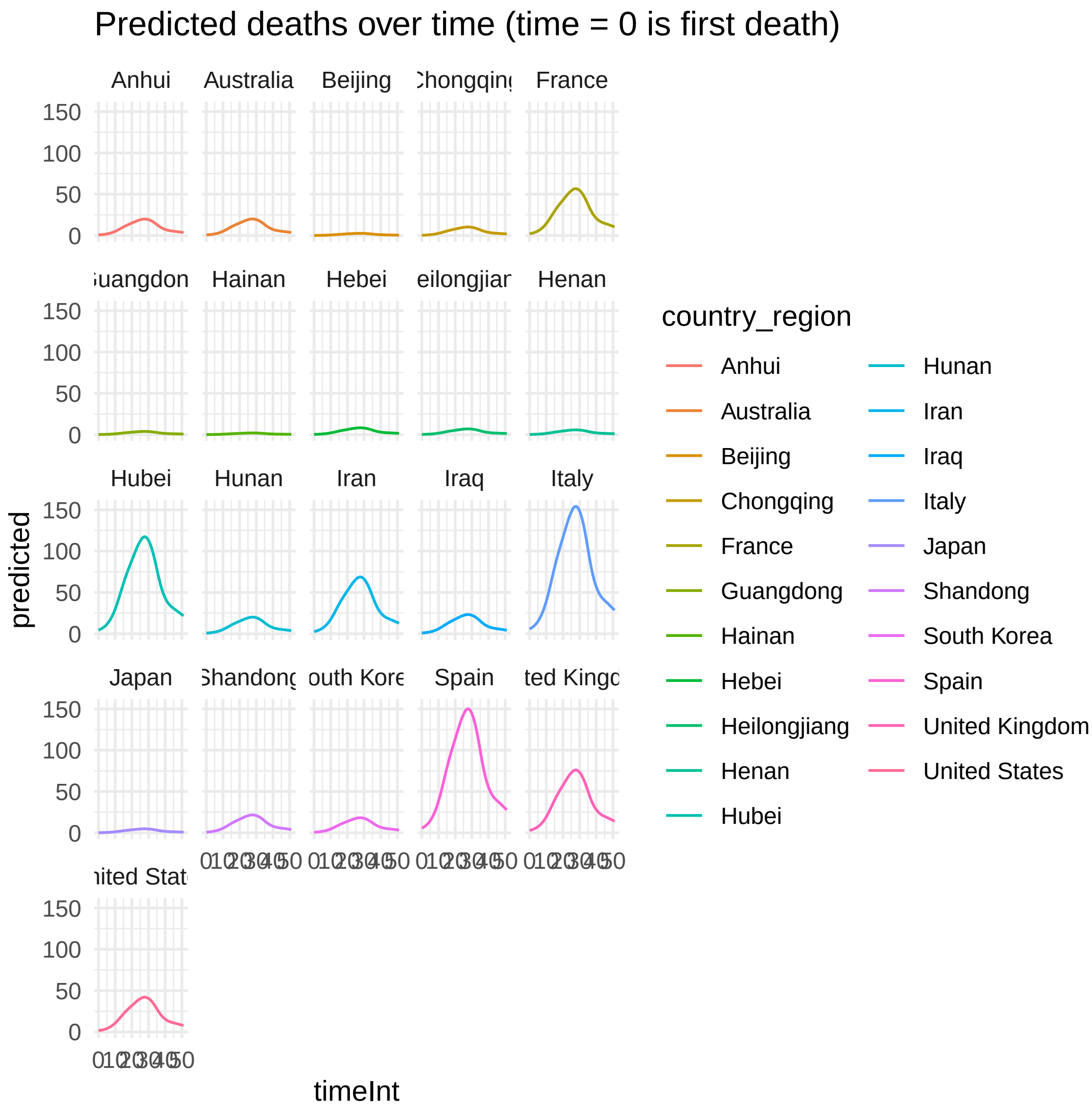
```
##  Groups     Name        Variance Std.Dev.
##  timeIntInd (Intercept) 0.08203  0.2864
##  Xr         s(timeInt)  5.19008  2.2782
## Number of obs: 170, groups:  timeIntInd, 50; Xr, 18
##
## Fixed effects:
##                                 Estimate Std. Error z value Pr(>|z|)
## X(Intercept)                   -0.306437   0.605247  -0.506 0.612645
## Xcountry_regionAustralia        0.006211   1.163606   0.005 0.995741
## Xcountry_regionBeijing         -2.011586   0.741460  -2.713 0.006668 **
## Xcountry_regionChongqing       -0.656670   0.823484  -0.797 0.425202
## Xcountry_regionFrance           1.045388   0.612974   1.705 0.088113 .
## Xcountry_regionGuangdong       -1.641550   0.775496  -2.117 0.034279 *
## Xcountry_regionHainan          -2.299258   0.843853  -2.725 0.006436 **
## Xcountry_regionHebei           -0.882377   0.825905  -1.068 0.285351
## Xcountry_regionHeilongjiang    -1.054878   0.668917  -1.577 0.114797
## Xcountry_regionHenan           -1.241664   0.633177  -1.961 0.049878 *
## Xcountry_regionHubei            1.772182   0.591076   2.998 0.002716 **
## Xcountry_regionHunan            0.006235   1.163595   0.005 0.995725
## Xcountry_regionIran             1.236426   0.592365   2.087 0.036863 *
## Xcountry_regionIraq             0.151171   0.768721   0.197 0.844099
## Xcountry_regionItaly            2.044937   0.590876   3.461 0.000538 ***
## Xcountry_regionJapan           -1.417752   0.657104  -2.158 0.030961 *
## Xcountry_regionShandong         0.083857   0.822855   0.102 0.918828
## Xcountry_regionSouth Korea     -0.088674   0.599927  -0.148 0.882494
## Xcountry_regionSpain            2.018096   0.605039   3.335 0.000852 ***
## Xcountry_regionUnited Kingdom  1.338388   0.832973   1.607 0.108107
## Xcountry_regionUnited States   0.745249   0.623374   1.196 0.231888
## Xs(timeInt)Fx1                  2.801095   0.765145   3.661 0.000251 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(resGammInd$gam)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## dead ~ country_region + s(timeInt, k = 20, pc = 0)
##
## Parametric coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -0.306437   0.608598  -0.504 0.614603
## country_regionAustralia         0.006211   1.169960   0.005 0.995765
## country_regionBeijing          -2.011586   0.744887  -2.701 0.006923 **
## country_regionChongqing        -0.656670   0.827589  -0.793 0.427502
## country_regionFrance            1.045388   0.616680   1.695 0.090040 .
## country_regionGuangdong        -1.641550   0.779153  -2.107 0.035132 *
## country_regionHainan           -2.299258   0.850210  -2.704 0.006844 **
## country_regionHebei            -0.882377   0.829983  -1.063 0.287725
## country_regionHeilongjiang     -1.054878   0.672517  -1.569 0.116752
## country_regionHenan            -1.241664   0.636740  -1.950 0.051172 .
## country_regionHubei             1.772182   0.594618   2.980 0.002879 **
## country_regionHunan             0.006235   1.169950   0.005 0.995748
```

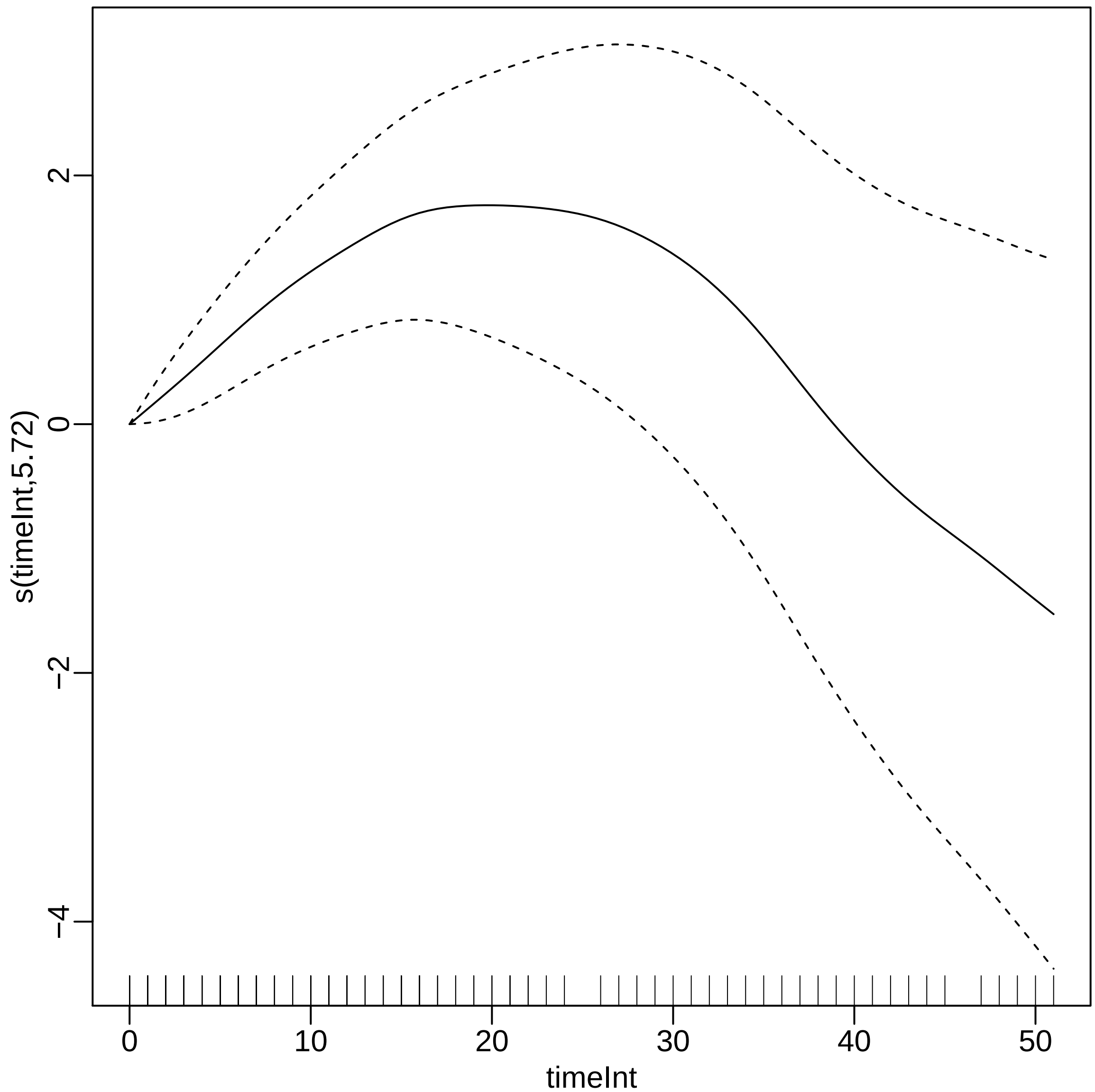
```
## country_regionIran            1.236426   0.595893   2.075 0.037995 *
## country_regionIraq            0.151171   0.773140   0.196 0.844979
## country_regionItaly           2.044937   0.594410   3.440 0.000581 ***
## country_regionJapan          -1.417752   0.660624  -2.146 0.031867 *
## country_regionShandong        0.083857   0.827682   0.101 0.919300
## country_regionSouth Korea    -0.088674   0.603450  -0.147 0.883175
## country_regionSpain           2.018096   0.608770   3.315 0.000916 ***
## country_regionUnited Kingdom  1.338388   0.839193   1.595 0.110746
## country_regionUnited States   0.745249   0.627122   1.188 0.234690
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq p-value
## s(timeInt) 5.58   5.58  289.7  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.884
## glmer.ML = 250.06  Scale est. = 1         n = 170
```

```r
covid_data_2 <- expand_grid(covid_data$timeInt, covid_data$country_region) %>%
  as_tibble() %>%
  rename(timeInt = 1, country_region = 2) %>%
  distinct()

covid_data_2$predicted <- predict(resGammInd$gam, newdata=covid_data_2, type="response")

#covid_data_3 <- bind_cols(covid_data_2, predicted) %>%
  #mutate(lower = fit - 2*se.fit, upper = fit + 2*se.fit)

covid_data_2 %>%
  ggplot(aes(timeInt, predicted, colour=country_region)) +
  geom_line() +
  theme_minimal() +
  facet_wrap(~country_region) +
  ggtitle("Predicted deaths over time (time = 0 is first death)")
```



Fit this model with a random slope for time.

```
covid_data$timeSlope = covid_data$timeInt/100

resGammSlope = gamm4::gamm4(
  dead ~ country_region + s(timeInt, k=30, pc=0),
    random = ~(0+timeSlope|country_region) +
    (1|timeIntInd:country_region),
  data=covid_data, family=poisson(link='log'))
#save(resGammSlope, file='resGamSlope.RData')
plot(resGammSlope$gam)
```




```
summary(resGammSlope$mer)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
##
##      AIC      BIC   logLik deviance df.resid
##    991.2   1069.6   -470.6    941.2      145
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2172 -0.3074 -0.0140  0.2211  2.0847
##
## Random effects:
##  Groups                     Name        Variance Std.Dev.
##  timeIntInd:country_region (Intercept)  0.08516 0.2918
##  Xr                         s(timeInt)   3.57400 1.8905
##  country_region             timeSlope   55.12954 7.4249
```

```
## Number of obs: 170, groups:
## timeIntInd:country_region, 170; Xr, 28; country_region, 21
##
## Fixed effects:
##                              Estimate Std. Error z value Pr(>|z|)
## X(Intercept)                 -0.24983    0.62028  -0.403  0.68711
## Xcountry_regionAustralia      0.09284    1.20936   0.077  0.93881
## Xcountry_regionBeijing       -0.62977    1.20219  -0.524  0.60038
## Xcountry_regionChongqing     -0.25546    0.92020  -0.278  0.78131
## Xcountry_regionFrance         0.95323    0.69540   1.371  0.17045
## Xcountry_regionGuangdong     -0.31291    0.95815  -0.327  0.74399
## Xcountry_regionHainan        -0.56876    1.18325  -0.481  0.63075
## Xcountry_regionHebei         -0.55696    0.98021  -0.568  0.56990
## Xcountry_regionHeilongjiang   0.14652    0.77875   0.188  0.85076
## Xcountry_regionHenan          0.43941    0.72414   0.607  0.54398
## Xcountry_regionHubei          1.80307    0.65400   2.757  0.00583 **
## Xcountry_regionHunan          0.09264    1.20939   0.077  0.93894
## Xcountry_regionIran           1.34643    0.66560   2.023  0.04309 *
## Xcountry_regionIraq           0.16340    0.83442   0.196  0.84475
## Xcountry_regionItaly          0.98691    0.68092   1.449  0.14724
## Xcountry_regionJapan          0.17604    0.82252   0.214  0.83053
## Xcountry_regionShandong       0.24186    0.89834   0.269  0.78775
## Xcountry_regionSouth Korea    0.46812    0.69387   0.675  0.49989
## Xcountry_regionSpain          1.97645    0.66863   2.956  0.00312 **
## Xcountry_regionUnited Kingdom 1.31476    0.89621   1.467  0.14237
## Xcountry_regionUnited States  0.93930    0.69625   1.349  0.17731
## Xs(timeInt)Fx1                1.57797    0.81611   1.934  0.05317 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
names(lme4::ranef(resGammSlope$mer))
```

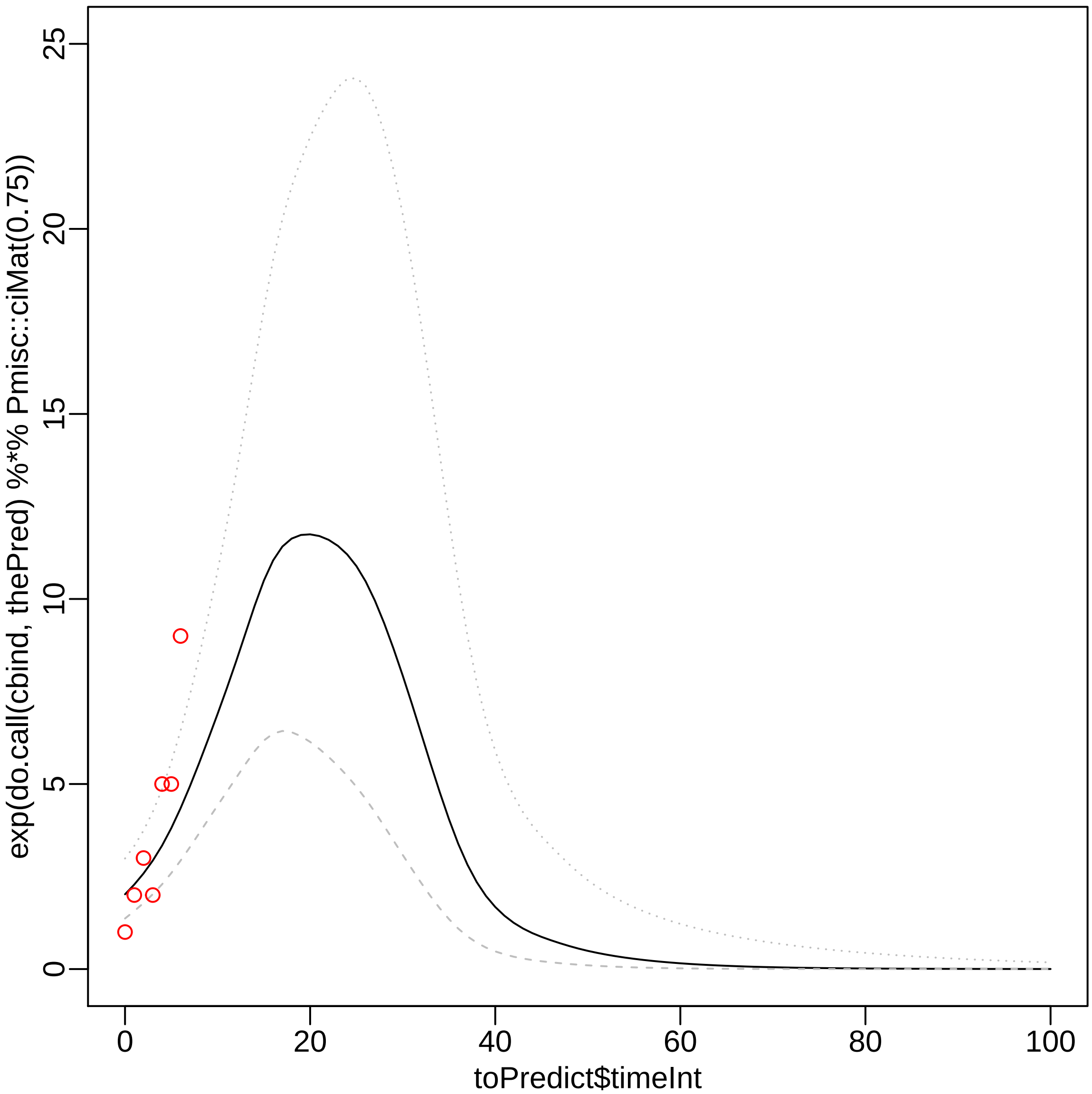
```
## [1] "timeIntInd:country_region" "Xr"
## [3] "country_region"
```

```r
theRanef = lme4::ranef(resGammSlope$mer, condVar = TRUE)$country_region
(theRanefVec = sort(drop(t(theRanef))))
```

```
##          Japan          Henan   Heilongjiang      Guangdong         Hainan
##    -7.45700329    -7.39204545    -6.59644587    -4.01325392    -3.18141365
##        Beijing  United States      Chongqing          Anhui          Hebei
##    -2.65662719    -1.74352611    -1.45020071    -0.17986002    -0.15065353
##           Iraq United Kingdom      Australia          Hunan       Shandong
##    -0.02902226     0.00000000     0.01707593     0.01717482     0.25244992
##    South Korea          Spain         France           Iran          Hubei
##     1.40467381     3.16634613     5.55008263     5.63517128     6.01197138
##          Italy
##    16.14480838
```

```r
Dcountry = 'France'
toPredict = expand.grid(
  timeInt = 0:100,
  country_region = Dcountry)
toPredict$timeSlope = toPredict$timeIntInd =
  toPredict$timeInt
thePred = predict(resGammSlope$gam,
```

```
                    newdata=toPredict, se.fit=TRUE)

matplot(toPredict$timeInt,
        exp(do.call(cbind, thePred) %*% Pmisc::ciMat(0.75)),
        type='l',
        col=c('black','grey','grey'),
        ylim = c(0, 25))
points(covid_data[covid_data$country_region == Dcountry,c('timeInt','dead')],
       col='red')
```

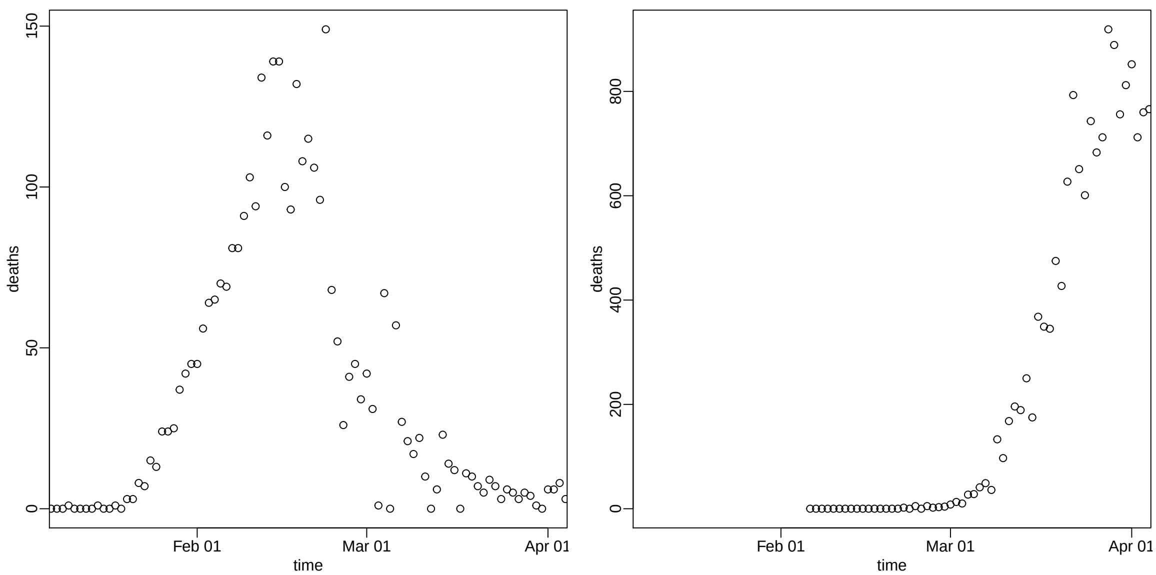


5 In-depth analysis for Italy and Hubei

```
if(!requireNamespace("nCov2019")) {
    devtools::install_github("GuangchuangYu/nCov2019")
}
```

```
x1 <- nCov2019::load_nCov2019(lang = 'en')
hubei = x1$province[which(x1$province$province == 'Hubei'), ]
hubei$deaths = c(0, diff(hubei$cum_dead))
italy = x1$global[which(x1$global$country == 'Italy'), ]
italy$deaths = c(0, diff(italy$cum_dead))
x = list(Hubei= hubei, Italy=italy)
```

```
for(D in names(x)) {
    plot(x[[D]][,c('time','deaths')], xlim = as.Date(c('2020/1/10', '2020/4/1')))
}
```



(a) Hubei (b) Italy

Figure 1: Covid 19 deaths

```
x$Hubei$weekday = format(x$Hubei$time, '%a')
x$Italy$weekday = format(x$Italy$time, '%a')
x$Italy$timeInt = as.numeric(x$Italy$time)
x$Hubei$timeInt = as.numeric(x$Hubei$time)
x$Italy$timeIid = x$Italy$timeInt
x$Hubei$timeIid = x$Hubei$time
```

```
gamItaly = gamm4::gamm4(deaths ~ weekday + s(timeInt, k=40), random = ~(1|timeIid),
    data=x$Italy, family=poisson(link='log'))
gamHubei = gamm4::gamm4(deaths ~ weekday + s(timeInt, k=100), random = ~(1|timeIid),
    data=x$Hubei, family=poisson(link='log'), REML=False)
```

6 Model for gamItaly

$$Y_t \sim Poisson(\lambda_t)$$
$$log(\lambda_t) = X_t\beta + f(t;\upsilon_1) + Z_t$$
$$Z_t \sim N(0, \sigma_2^2)$$

, for time t in Italy.

We use Poisson regression, where our response (number of deaths in time t) is linked to a linear combination of *weekday* covariates, X_t and an overdispersion term with a log link.

X_t are our *weekday* covariates (Monday, Tuesday, ..., Sunday with Friday as our intercept), $f(t)$ is a smoothly-varying function of *timeInt* for time t with 40 knots and υ_1 is its roughness parameter.

Z_t is the overdispersion or the independent random effect (random intercept) for each time t (*timeIid*).

7 Model for gamHubei

$$
\begin{aligned}
Y_t &\sim Poisson(\lambda_t) \\
log(\lambda_t) &= X_t\beta + f(t;\upsilon_2) + Z_t \\
Z_t &\sim N(0, \sigma_1^2)
\end{aligned}
$$

, for time t in Hubei.

We use Poisson regression, where our response (number of deaths in time t) is linked to a linear combination of covariates of *weekday* covariates, X_t and an overdispersion term with a log link.

X_t are our *weekday* covariates (Monday, Tuesday, ..., Sunday with Friday as our intercept), $f(t)$ is a smoothly-varying function of *timeInt* for time t with 100 knots and υ_1 is its roughness parameter.

Z_t is the overdispersion or the independent random effect (random intercept) for each time t (*timeIid*).

```
lme4::VarCorr(gamItaly$mer)
```

```
##  Groups  Name        Std.Dev.
##  timeIid (Intercept) 0.15538
##  Xr      s(timeInt)  2.57786
```

```
lme4::VarCorr(gamHubei$mer)
```

```
##  Groups  Name        Std.Dev.
##  timeIid (Intercept) 0.40273
##  Xr      s(timeInt)  6.57654
```

```
knitr::kable(cbind(summary(gamItaly$mer)$coef[,1:2], summary(gamHubei$mer)$coef[,1:2]), digits=3)
```

	Estimate	Std. Error	Estimate	Std. Error
X(Intercept)	2.749	0.228	-0.990	0.911
XweekdayMon	0.105	0.102	-0.243	0.198
XweekdaySat	0.114	0.103	-0.106	0.196
XweekdaySun	-0.044	0.104	-0.059	0.195
XweekdayThu	-0.008	0.098	-0.439	0.201
XweekdayTue	-0.037	0.102	-0.565	0.206
XweekdayWed	0.113	0.099	-0.056	0.194
Xs(timeInt)Fx1	2.915	1.197	5.161	4.225

```
toPredict = data.frame(time = seq(as.Date('2020/1/1'), as.Date('2020/4/10'), by='1 day'))
toPredict$timeInt = as.numeric(toPredict$time)
toPredict$weekday = 'Fri'
Stime = pretty(toPredict$time)
matplot(toPredict$time,
    exp(do.call(cbind, mgcv::predict.gam(gamItaly$gam, toPredict, se.fit=TRUE)) %*% Pmisc::ciMat()),
    col='black', lty=c(1,2,2), type='l', xaxt='n', xlab='', ylab='count', ylim = c(0.5, 5000),
```

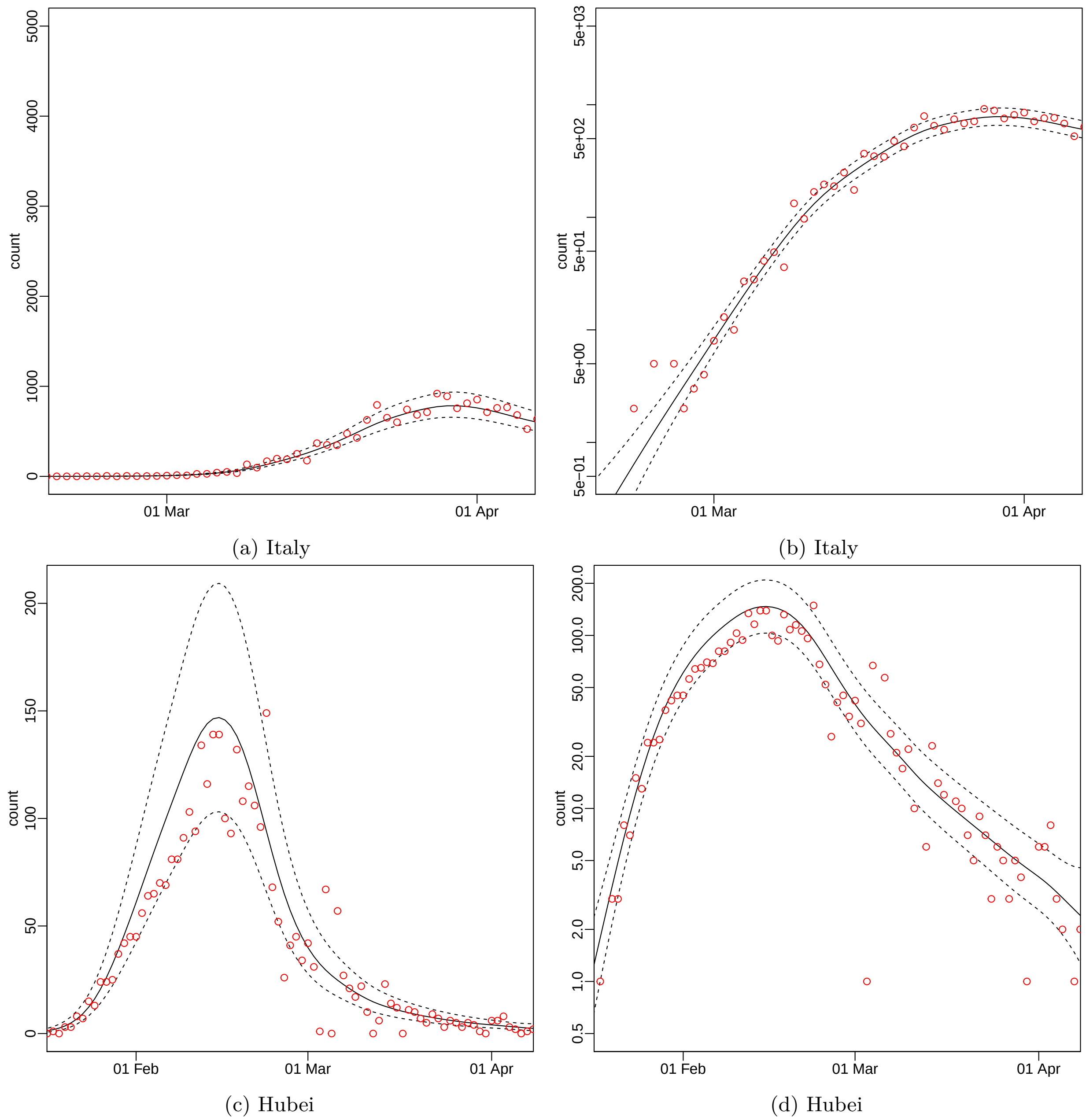
```r
    xlim = as.Date(c('2020/2/20', '2020/4/5')))
axis(1, as.numeric(Stime), format(Stime, '%d %b'))
points(x$Italy[,c('time','deaths')],  col='red')
matplot(toPredict$time,
    exp(do.call(cbind, mgcv::predict.gam(gamItaly$gam, toPredict, se.fit=TRUE)) %*% Pmisc::ciMat()),
    col='black', lty=c(1,2,2), type='l', xaxt='n', xlab='', ylab='count', ylim = c(0.5, 5000),
    xlim = as.Date(c('2020/2/20', '2020/4/5')), log='y')
axis(1, as.numeric(Stime), format(Stime, '%d %b'))
points(x$Italy[,c('time','deaths')],  col='red')

matplot(toPredict$time,
    exp(do.call(cbind, mgcv::predict.gam(gamHubei$gam, toPredict, se.fit=TRUE)) %*% Pmisc::ciMat()),
    col='black', lty=c(1,2,2), type='l', xaxt='n', xlab='', ylab='count',
        xlim = as.Date(c('2020/1/20', '2020/4/5')))

axis(1, as.numeric(Stime), format(Stime, '%d %b'))
points(x$Hubei[,c('time','deaths')],  col='red')
matplot(toPredict$time,
    exp(do.call(cbind, mgcv::predict.gam(gamHubei$gam, toPredict, se.fit=TRUE)) %*% Pmisc::ciMat()),
    col='black', lty=c(1,2,2), type='l', xaxt='n', xlab='', ylab='count',
        xlim = as.Date(c('2020/1/20', '2020/4/5')), log='y', ylim =c(0.5, 200))

axis(1, as.numeric(Stime), format(Stime, '%d %b'))
points(x$Hubei[,c('time','deaths')],  col='red')
```

8 Brief Analysis

Firstly, we can conclude (from figure 5) that we are reasonably confident that the number of deaths from COVID-19 in Italy is in an increasing trend from early March to 23 March (last date of our collected data). We are also rather confident that this increasing trend is going to maintain, perhaps even more sharply so, going into the month of April. But there is some degree of uncertainty as to how fast the increase of deaths will be, but we are very confident that number of deaths will be increasing. For Hubei, however, we have observed with reasonable certainty that there is also a sharp increasing trend of deaths from early Feburary to late Feburary, with a small peak (of number of deaths) at around mid-Feburary, and that the number of deaths have started to decrease consistently until March 23. However, we are less certain that the decreasing trend for Hubei will continue going into April (i.e., there is still a lot room for sudden increases/decreases of death) and this is mainly due to the lack of data collected. Also, we can conclude with reasonabe certainity that days in the week doesn't seem to have a strong effect (if any at all) on the number of deaths for both Italy and Hubei.

9 Likelihood ratio tests (with boundary corrections) for various models

```r
Hubei2 = gamm4::gamm4(deaths ~ 1 + s(timeInt, k=100), random = ~(1|timeIid),
    data=x$Hubei, family=poisson(link='log'), REML=FALSE)
Hubei3 = mgcv::gam(deaths ~ weekday + s(timeInt, k=100),
    data=x$Hubei, family=poisson(link='log'), method='ML')
Hubei4 = lme4::glmer(deaths ~ weekday + timeInt + (1|timeIid),
    data=x$Hubei, family=poisson(link='log'))
```



(a) Italy (b) Italy

(c) Hubei (d) Hubei

Figure 2: Predicted cases

10 LRT for significance of fixed effect of weekday

```
lmtest::lrtest(Hubei2$mer, gamHubei$mer)
```

```
## Likelihood ratio test
##
## Model 1: y ~ X - 1 + (1 | Xr) + (1 | timeIid)
## Model 2: y ~ X - 1 + (1 | Xr) + (1 | timeIid)
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   4 -337.20
## 2  10 -330.69  6 13.012    0.04285 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

11 LRT for significance of random effect of timeIid, uses boundary correction

```
nadiv::LRTest(logLik(gamHubei$mer),logLik(Hubei3), boundaryCorrect=TRUE)
```

```
## $lambda
## 'log Lik.' 24.25991 (df=10)
##
## $Pval
## 'log Lik.' 4.208634e-07 (df=10)
##
## $corrected.Pval
## [1] TRUE
```

12 LRT for significance of smoothing function, uses boundary correction

```
nadiv::LRTest(logLik(gamHubei$mer),logLik(Hubei4), boundaryCorrect=TRUE)
```

```
## $lambda
## 'log Lik.' 240.6041 (df=10)
##
## $Pval
## 'log Lik.' 1.451966e-54 (df=10)
##
## $corrected.Pval
## [1] TRUE
```