Bayesian Inference on Covid-19 in England

Jonathan Hoover

Practical 5: Bayesian inference on Covid-19 incidence in England

Author: Jonathan Hoover

Description:

This project was an individual R coding project in my statistical programming course at the University of Edinburgh. The emphasis was proper coding, so analysis is minimal. The project prompt is given in the document "practical-5_description.pdf", but the goal is also defined below.

Goal

The goal of this project is to model the the distribution of fatal Covid-19 infections in the UK from March 11, 2020 to May 20, 2020 using daily Covid-19 death records from March 2, 2020 to June 9, 2020 (100 total days). The death counts on each day are referred to as yi where Y is a random variable distributed according to a Poisson with a mean mi, described below (Yi ~ Pois(mi)).

We will take a Bayesian approach to this analysis, assigning prior distributions to some of the variables that influence the distribution of daily fatal covid cases. Using the priors and the likelihood, we will generate a posterior distribution for infectious cases on a given day using Gibbs Sampling implemented via JAGS. The specified model is provided in a file called model.jags, but the parameters and priors are specified below.

The definitions of priors comes from the project description (see "practical-5_description.pdf" for details) and the ISARIC study.

A few notes before specifying the parameters:

1) We are assuming that daily fatal infections can be modelled according to the death data from subsequent days because data on fatal disease duration (time from infection to death) has been shown to follow a log normal distribution. This is what allows us to predict fatal infections from death data (data is from the ISARIC study of 24000+ hospitalized patients).

2) We assume that differences in daily fatal infections from day to day are fairly smooth, which lets us model the daily fatal infection cases as a smooth distribution. The variables we will use in this model are:

- tau: Precision parameter that will partially define the distribution of log daily infections (xi; below). Tau is also a random variable, with a prior distribution tau ~ Gamma(4, 0.04). An initial value for tau must be specified. We chose tau0 = 0.01.
- xi: Modelled log daily fatal infections (modelled as log since infection counts cannot be negative). Modelled as a second order random walk such that x[i+1] - 2[xi] + x[i-1] ~ N(0, tau), where tau is precision. Alternatively, xi can be modelled as xi ~ N(2x[i-1] - x[i-2], tau), which is the formulation we use in the model. Initial values for xi must be specifed. We chose x1 ~ N(0, tau0 = 0.01) and x2 ~ N(x1, tau)

- ni: Modelled daily fatal infections. Can be calculated as exp(xi).
- B: Square transition probability matrix defining the probability of dying (from infection) on day i given an infection on any previous day (lower probabilities for early infectious days). Here, Bij refers to probability of dying on day i given infection on day j. This probability, piD(i - j), is defined as a log normal distribution, log D ~ N(3.235, 0.4147), when i >= j (infected before or on day of death) and as 0 when i < j (Not possible to be infected after death). This is a transition probability matrix since dying is the equivalent of moving from state n to m (see next)
- mi: Modeled daily death counts. Calculated as the ith element of B %% *n, or B[1:i]* %% n[1:i]. It is the sum of Prob(death | infection on day j)*(infections on day j) for all j until i. Also known as expected death count on day i
- yi: Observed deaths on day i. Y is a random variable such that Yi ~ Pois(mi). This only makes sense if The data starts with a reasonable number of zeros. Given no deaths occurred before March 2, it is fair to simply append 20 zeros to the start of y.

Note: While the sampler will provide ni values on the last 20 days given the death data (yi) has data on these days, This data is unreliable as it is modelling fatal infections on death data that has not been observed, so we will discard this data.

The rest of the code sets up the variables required for the Gibbs sampler, runs the gibbs sampler, and plots the modelled/observed data. The final plot shows the posterior means for n (with 95% HPD) and m at each modelled day along with observed data.

```
##############################################################################
##Start of Analysis##

##Import necessary libraries
library(rjags)
```

```
## Warning: package 'rjags' was built under R version 4.1.2
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
library(ggplot2)

## Part 1: Setup Data for JAGS sampler

##Provided Death Data:
##data on daily hospital deaths with Covid in England for the first 100 days
##from March 2nd 2020 from NHS England; no deaths recorded before this day
y <- c(1,2,0,2,2,0,4,4,1,9,14,20,22,27,40,46,66,63,105,103,149,159,
       204,263,326,353,360,437,498,576,645,647,700,778,743,727,813,
       900,792,740,779,718,699,646,686,639,610,571,524,566,486,502,
       451,438,386,380,345,341,325,313,306,268,251,259,252,266,256,
       215,203,196,166,183,162,180,172,167,138,160,144,153,150,122,
       128,116,133,139,122,124,117,92,83,94,109,111,83,86,83,80,73,69)
```

```r
##append 20 zeros to start of data since no deaths were recorded before March 2nd,
##But death data from previous days is required to model number of infections, ni.
y <- c(rep(0, 20), y)
N <- length(y) ##number of observations (including appended zeros)

##Setup B:
##B is a square probability matrix defining Probability of Death on day i given infection
##on day ni modeled by log D ~ N(3.235, 0.41472). See beginning for of code for explanation of B

##For efficiency:
##since vals of i-j shift down for each column (col 1 = 0:n-1, col 2 = -1:n-2 ), we
##just need to calculate dlnorm values for each i-j combination once and then shift these values down
##for each subsequent column. This will create a lower triangular matrix for B, where
##Bij = dlnorm(i-j, 3.235, 0.4147^2) when i >= j and Bij = 0 when i < j.

meanlnorm <-3.235; sdlnorm <- 0.4147 ##log D ~ N(meanlnorm, sdlnorm^2) = N(3.235, 0.4147^2)
ij.diff <- 0:(N-1) ##difference between i and j for first column
ij.dlnorm <- dlnorm(ij.diff, meanlog = meanlnorm, sdlog = sdlnorm) ##calculate pD(i-j) for all possible
B <- matrix(0, nrow = N, ncol = N) ##NxN empty matrix

##create B with method outlined above: values of dlnorm(i-j) shifted down for each column
##Could also run a for loop for a matrix of this size, but for larger matrices a loop is less efficient
B[lower.tri(B, diag = T)] <- unlist(sapply(N:1, function(k) ij.dlnorm[1:k]))
```

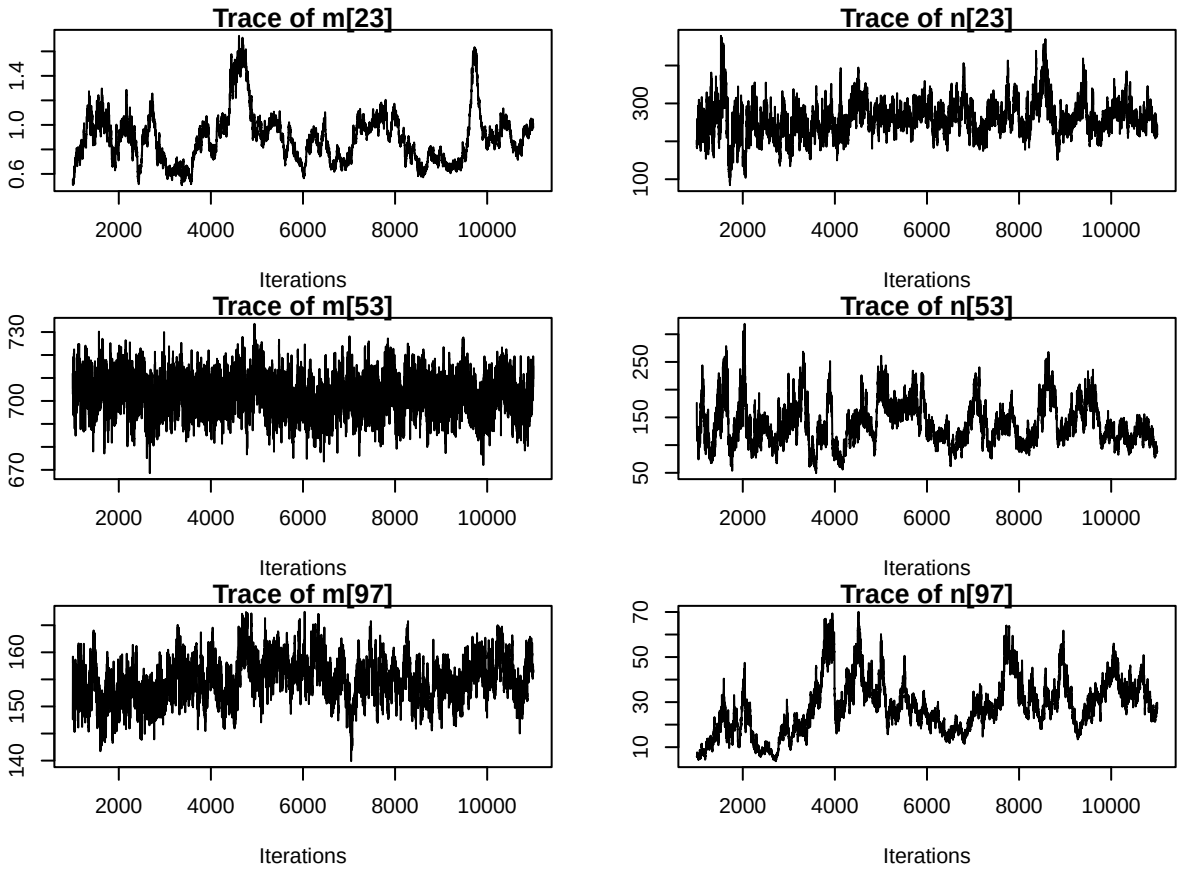
```r
## Part 2: Run JAGS Model
data <- list(y = y, B = B, N = N) ##data to pass to jags model
tau0 <- 0.01 ##initial value of tau to pass to jags model
##Note: Initial values of x are defined in "model.jags" as x1 ~ N(0, tau0) and x2 ~ N(x1, tau)

##Ony running 10000 iterations for sake of efficiency, see comments below for recommendations
##On a real run. Also only monitoring parameters n and m as these are the only ones we plot
mod <- jags.model(file = "model.jags", data = data, inits = list("tau" = tau0)) ##compile jags model wi
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 120
##    Unobserved stochastic nodes: 121
##    Total graph size: 15360
##
## Initializing model
```

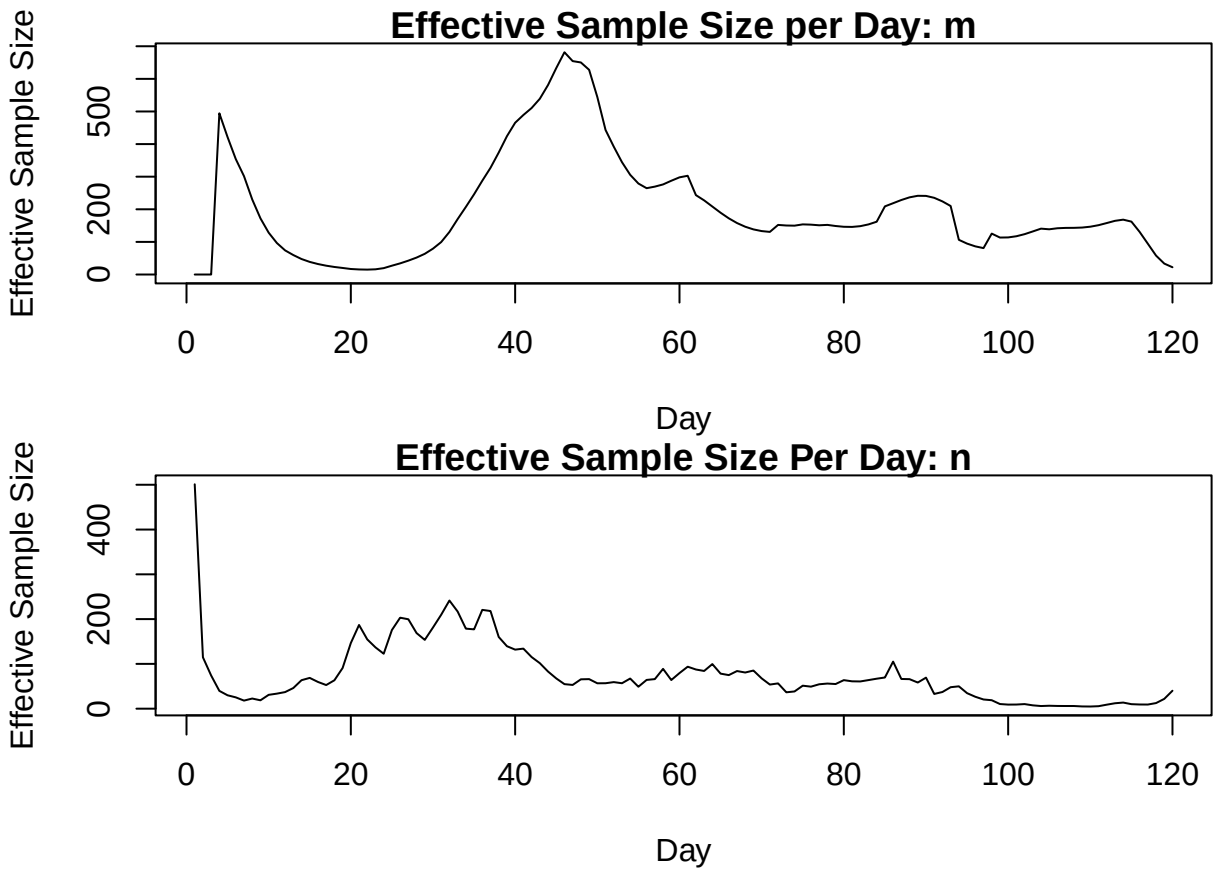
```r
sam.coda <- coda.samples(mod,c("n","m"),n.iter=10000) ##sample 10000 times from the specified model and
```

```r
##Plotting Diagnostics Plots for bayesian sampling model
##Note: Only plotting a few bad samples to identify burn-in period
##      and issues with autocorrelation (that might require model tuning)
##Data shown: n[23] and m[23]; n[53] and m[53]; n[97] and m[97]


##Trace plots
```

```
diagnostic_index <- c(23, 23+120, 53, 53+120, 97, 97 + 120) ##chosen days to plot from above
par(mfrow=c(3,2),mar=c(4,4,1,1))##setting up graphics to plot m and n traceplots side by side
traceplot(sam.coda[[1]][,diagnostic_index]) ##plot selected m and n
```




```
##Effective Sample Size plots (Effective Size per Day)
par(mfrow = c(2,1), mar=c(4,4,1,1)) ##set up graphics
effsize <- effectiveSize(sam.coda) ##calculate effective sample sizes
plot(1:N, (effsize[which(grepl(names(effsize), pattern = "m"))]), type = "l",##subset data for m and dr
     main = "Effective Sample Size per Day: m", xlab = "Day", ylab = "Effective Sample Size")
plot(1:N, (effsize[which(grepl(names(effsize), pattern = "n"))]), type = "l",##subset data for n and dr
     main = "Effective Sample Size Per Day: n", xlab = "Day", ylab = "Effective Sample Size")
```




```
##########################################################################
```

Diagnostic Comments:

The diagnostics at model days 23, 53, and 97 of n and m suggest that mixing and convergence are different for each parameter. M has poor mixing for earlier days (ie day 23) since m is deaths, which is delayed compared to n (infections); however, it is good at later days (53 and 97) N has poor mixing in the middle and at the end, but decent mixing at the beginning. The poor mixing in n is likely due to poor autocorrelation (acf plots show high autocorrelation; these plots are not shown here). This might be caused by the nature of our second order random walk model, which is dependent on infections in previous days. The better mixing in m is likely due to the additional stochasticity from B ~ dlnorm(i-j, 3.235, 0.4147^2)).

M appears to converge quickly on most days. N converges quickly on most days, but around day 80 it starts taking longer to converge. By day 97 (shown here), we see it requires about 9000 iterations to converge. This is likely because we are working on the log scale. Close to the beginning and end of the model, n (number of daily fatal infections) is fairly low. As such, the exploration of the space on the log scale will be very small and the model can take a while to converge.

Lastly, the effective sample size is quite low across all days, which is likely caused by high autocorrelation

Conclusions & Recommendations:

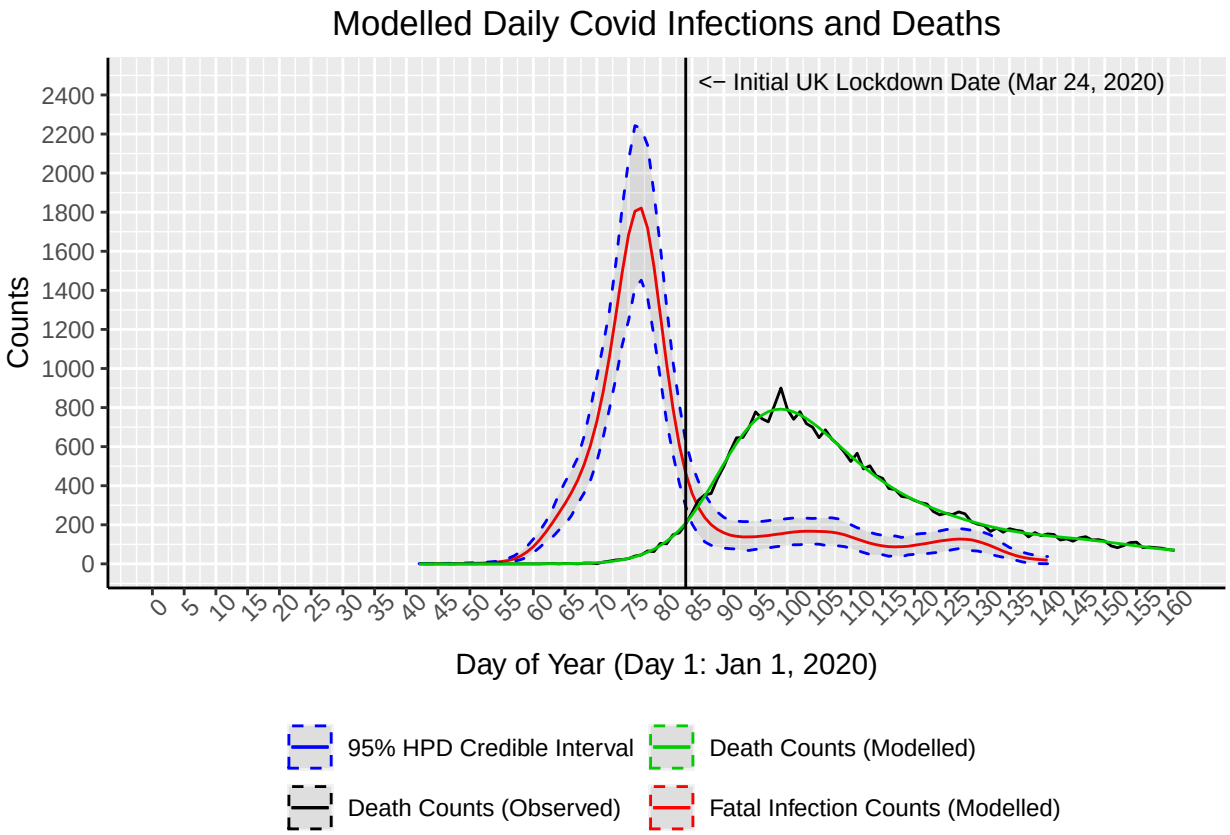
If this were a true study, I would recommend setting a burn-in of 9000-10000 iterations due to the time it takes for later samples (n[97]) to converge. We want to be certain that results are from the data and not due to artifacts from the sampling process. Additionally, I would recommend running the chain for at least

50000 iterations to increase the effective sample size. Lastly, we may wish to tune the tau parameter (or revisit the random walk model) to try to reduce autocorrelation.

Note: I would also recommend plotting density and autocorrelation plots, but due to restrictions in output size (2 A4 pages) they are not including here. The code is provided below:

```
##Density Plot
##densplot(sam.coda[[1]][,diagnostic_index])##plot m and n density plots side by side
##autocorrelation (acf) plots
##acfplot(sam.coda[[1]][,diagnostic_index], aspect = 2)

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

##Part 3: Final Plots

##Summarize data for plotting

data <- as.data.frame(sam.coda[[1]])##convert sampling output to dataframe
ni <- data[, grepl(colnames(data), pattern = "n", fixed = T)] ##subset data for n based on colnames
mi <- data[, grepl(colnames(data), pattern = "m", fixed = T)] ##subset data for m based on colnames
n_mean <- colMeans(ni); m_mean <- colMeans(mi) ##Expected values of n and m on each day

##Generate credible intervals for all data and then
##subset both m and n based on row.name and rename to day index (1:120)
CI <- as.data.frame(HPDinterval(sam.coda[[1]]))##Highest Probability Density Credible Intervals
m_CI <- `row.names<-`(CI[grepl(row.names(CI), pattern = "m", fixed = T),], 1:N) ##m CI subset
n_CI <- `row.names<-`(CI[grepl(row.names(CI), pattern = "n", fixed = T),], 1:N) ##n CI subset

colnames(m_CI) <- c("m_lower", "m_upper") ##rename to give a unique name for m
colnames(n_CI) <- c("n_lower", "n_upper") ##rename to give a unique name for n

##convert day indices (1:120) to day of year starting at Jan 1, 2020 for plotting
death_start <- julian(as.Date("2020-3-2"),origin = as.Date("2019-12-31")) ##start of data collection (d
y_start <- death_start-20 ##y-vector start date given 20 appended zeros (day of year format)
day_of_year <- 0:(N-1) + y_start ##convert each day index to day of year format
lockdown_start <- julian(as.Date("2020-3-24"), origin = as.Date("2019-12-31")) ##Date of initial UK loc

##Create summary dataframe for plotting
summary_data <- data.frame(day_of_year = day_of_year, days = 1:N, y = y,
                           n_mean = n_mean, n_lower = n_CI["n_lower"], n_upper = n_CI["n_upper"],
                           m_mean = m_mean, m_lower = m_CI["m_lower"], m_upper = m_CI["m_upper"])


##Add empty data for the first days of the year before model starts (To give context to the x-axis poin
summary_data <- rbind(data.frame(day_of_year = 1:(y_start-1), days = NA, y = NA,
                                 n_mean = NA, n_lower = NA, n_upper = NA,
                                 m_mean = NA, m_lower = NA, m_upper = NA),
                      summary_data)

##excluding last 20 observations of N since they are unreliable
ind_exclude <- which(summary_data[,"day_of_year"] > (max(day_of_year) - 20)) ##index of samples to excl
summary_data[ind_exclude, c("n_mean", "n_lower", "n_upper")] <- NA ##set those indices as NA values, wh

##y limit
ymax <- max(summary_data$n_upper, na.rm = T)*1.1 ##set y-limit for plot
```

```r
##Plot data
par(mfrow = c(1,1), mar = c(4,4,4,1))##set up graphics for final plot

gg<- ggplot(data = summary_data, aes(x = day_of_year))+ ##summary data with x-axis as days of year
  geom_line(aes(y = round(n_mean, digits = 0), color = "Fatal Infection Counts (Modelled)"))+##plot pos
  geom_ribbon(aes(ymin = n_lower, ymax = n_upper, color = "95% HPD Credible Interval"), linetype=2, alp
  geom_line(aes(y = y, color = "Death Counts (Observed)"))+ ##plot observed death data
  geom_line(aes(y = round(m_mean, digits = 0), color = "Death Counts (Modelled)"))+ ##plot expected mod
  geom_vline(xintercept = lockdown_start)+ ##add vertical line at day of initial lockdown start
  annotate(geom = "text", label = "<- Initial UK Lockdown Date (Mar 24, 2020)",
           x = lockdown_start + 2, y = ymax, size =3, colour = "black", hjust = 0)+ ##add text to expla
  scale_x_continuous(breaks = seq(0, max(day_of_year), by = 5))+ ##Setting up x-axis labels
  scale_y_continuous(breaks = seq(0, ymax, by = 200), limits = c(0, ymax))+ ##Setting up y-axis labels
  scale_colour_manual(values = c("blue", "green3", "black", "red")) + ##specify colors for each plot
  labs(color = "")+ ##get rid of legend label
  ylab("Counts")+##y label
  xlab("Day of Year (Day 1: Jan 1, 2020)")+ ##x label
  ggtitle("Modelled Daily Covid Infections and Deaths")+ ##graph title
  guides(color=guide_legend(nrow=2,byrow=TRUE))+ ##put legend into 2 rows
  theme(legend.position = "bottom",
        axis.text.x = element_text(angle = 45),
        axis.line = element_line(colour = "black"),
        plot.title = element_text(hjust = 0.5)) ##formatting legend, axes, and title

##I am suppressing warnings of removed NA values. Given these NA values
##were intentionally introduced to exclude during data plotting, I am suppressing them
suppressWarnings(print(gg))
```


Modelled Daily Covid Infections and Deaths
<– Initial UK Lockdown Date (Mar 24, 2020)
Counts
Day of Year (Day 1: Jan 1, 2020)
95% HPD Credible Interval
Death Counts (Modelled)
Death Counts (Observed)
Fatal Infection Counts (Modelled)