Modelling the Spread of COVID-19 Cases

TCD Final Year Project

Anthony Gibbons

Project Supervisor: Athanasios Georgiadis

March 15, 2021

Abstract

We construct various models of the Covid-19 pandemic over various periods of 2020. We first construct simple model of a the epidemic by using a recurrence equation. We also add a periodic complexity to these simpler models. We then use more statistical methods, modelling using time series forecasting methods such as HoltWinters, ARIMA and Neural Network methods. All of this is with the aim of predicting the course of the epidemic.

Keywords— ARIMA, Autoregressive model, COVID-19; Coronavirus, Forecasting, Mathematical model, Neural Network, Pandemic, Parameter estimation, SARS-CoV-2, Statistical Model.

Plagiarism Declaration

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

Signed: Anthony Gibbons
Date: 10/03/2021

Contents

List of Figures

List of Code

1 Introduction

The Coronavirus disease (COVID-19) was first characterized by the World Health Organisation as pandemic on 11th March 2020 [14]. The outbreak has affected almost every aspect of human life throughout 2020, and is expected to continue for much of 2021.

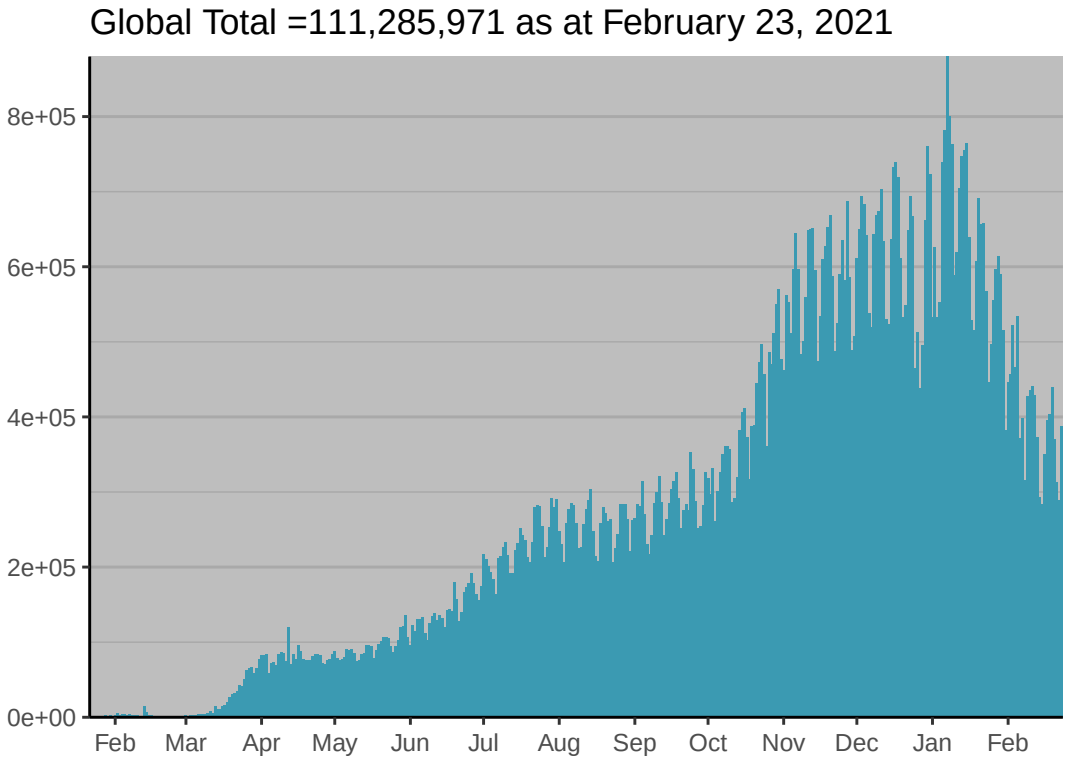

Figure 1: Daily Cases Globally

We can map the cumulative number of cases per 100,000 population for each country to see the varying severity of disease spread.

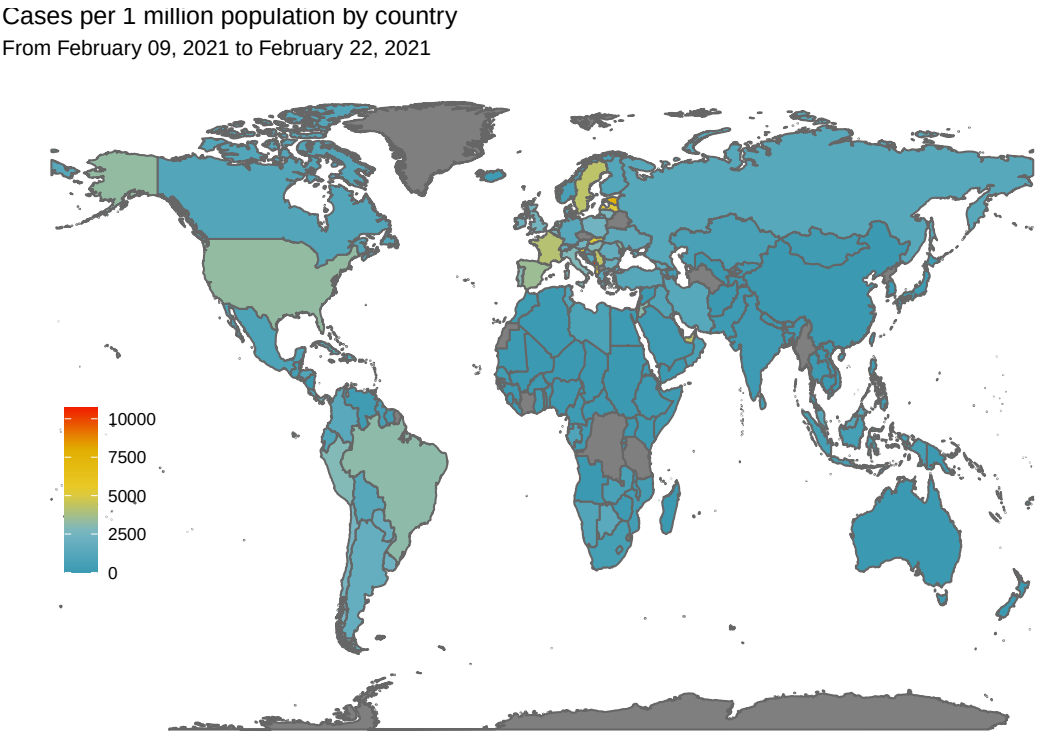

Figure 2: Cases per 1 million population, World

Europe is experiencing an especially high number of cases, proportionally, as well as the US.

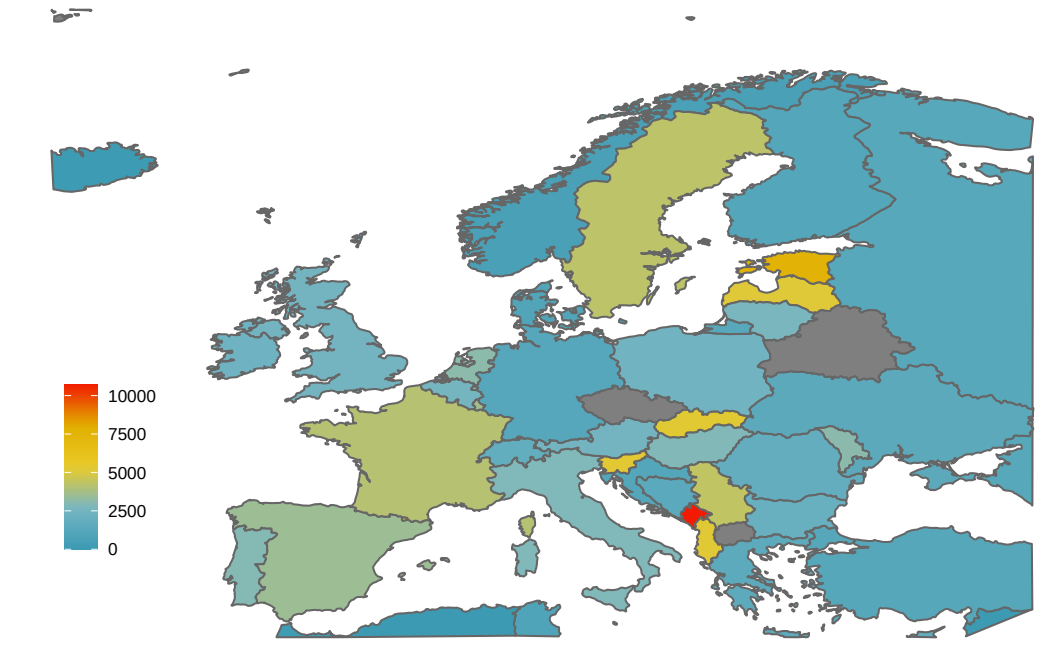

Figure 3: Cases per 1 million population, Europe

More locally, we see that Ireland also has a clear variation in concentration of cases to date, with Donegal and

much of Leinster experiencing sometimes twice as many cases per 100,000 population as the rest of the country.

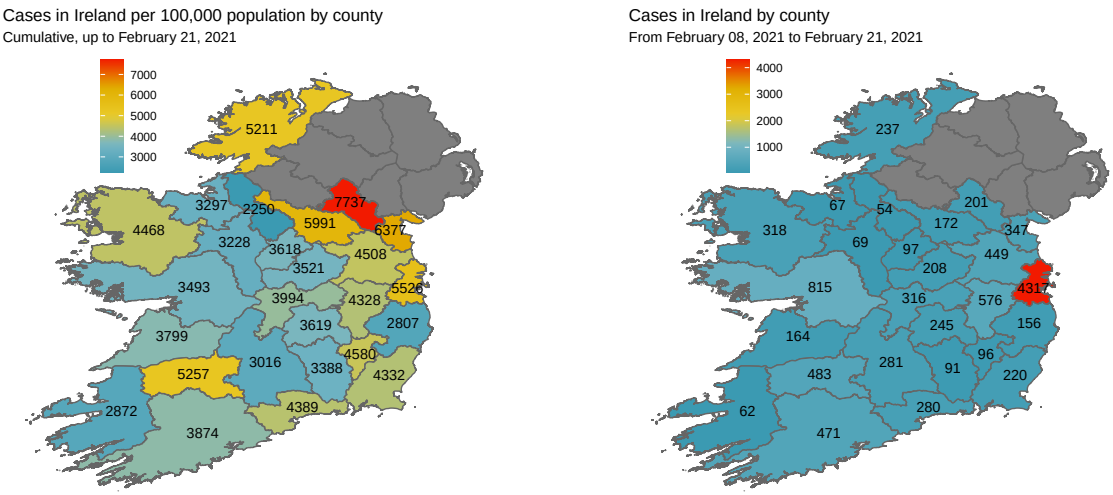

Figure 4: Situation by County, Ireland

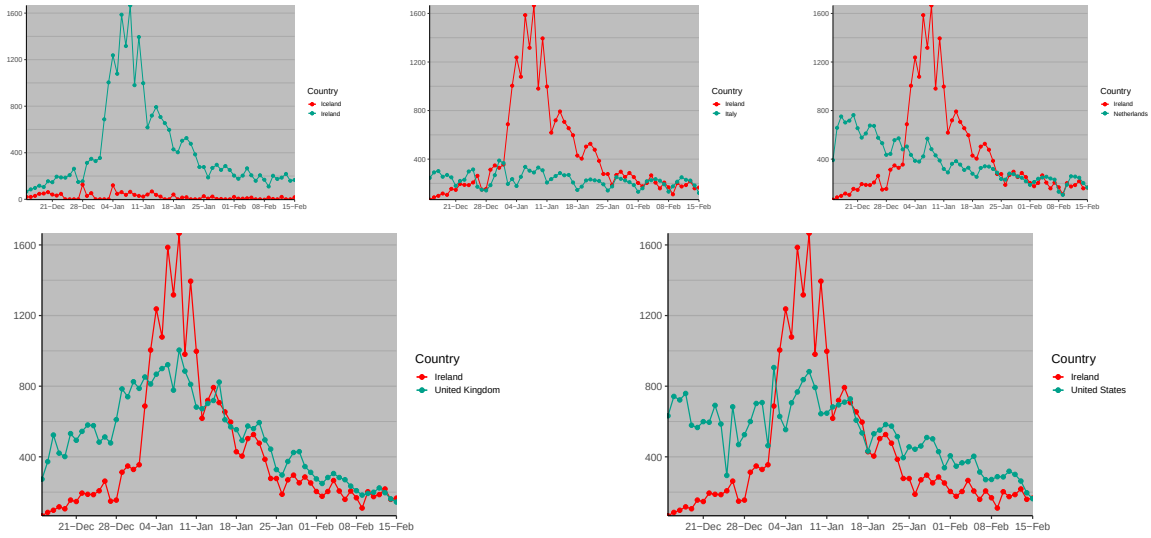

Figure 5: Individual Comparison of Ireland with various countries, daily cases per million of the population

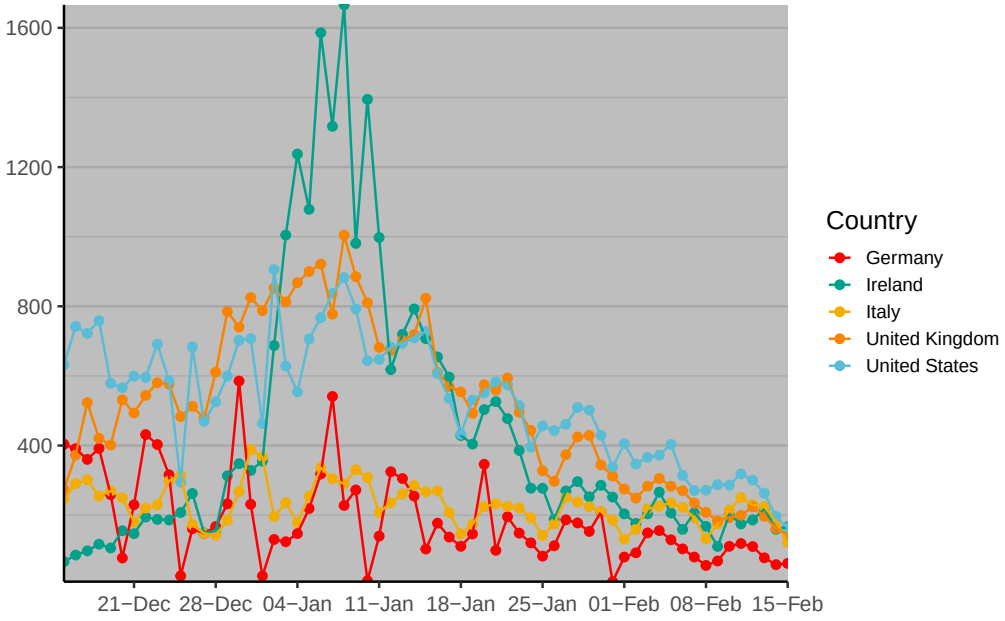

Figure 6: Comparison of Ireland with various countries, daily cases per million of the population

1.1 Previous work on Covid-19 identification and modeling

Research in the area of modeling the spread of the pandemic has been extensive and as such it would be impossible to acknowledge all the previous and ongoing work here. I would like to note a few studies (first published towards the beginning of the pandemic) that differ to my approach.

The work involving *external* factors such as government travel restrictions or full lockdowns, carried out by [13], was widely read. However, it was also criticised for their model (which tried to assign a quantitative effect of interventions on disease spread for multiple countries) lacking practical statistical distinguishability, and prompted revisions [12].

Artificial intelligence models have also been employed to track disease outbreaks in more local areas. The model developed by [23], which relies on phone–based surveys, certainly has the long-run potential to keep the public informed and hopefully reduce the the severity of outbreaks in areas where the app is widely used. One drawback of the initial model was its estimation of the peak of case numbers (which is notoriously difficult to predict) being the highest value in the case numbers so far.This does not take into account the shape of many time series during the early stages of the virus outbreak. For example, a strictly increasing time series would have its maximum at the latest time.

1.2 Key aims

This project is based on the work in [16], where I attempt to reconstruct the recurrence relation to model the pandemic. This is a largely mathematical model (based on practical assumptions), but of course does not fit well in the long run. It is efficient at explaining singular phases of the pandemic (with a consistent trend), and calculating the infamous R_0 number, defined below, from [2].

Definition. The number R_0 is called the *basic reproduction number* and is unquestionably the most important quantity to consider when analyzing any epidemic model for an infectious disease. Each infective individual can be expected to infect R_0 individuals.

2 Mathematical Models

As per the base and periodic models shown in [16].

2.1 Base model

2.1.1 Definitions and Theory

2.1.2 How to select the best model

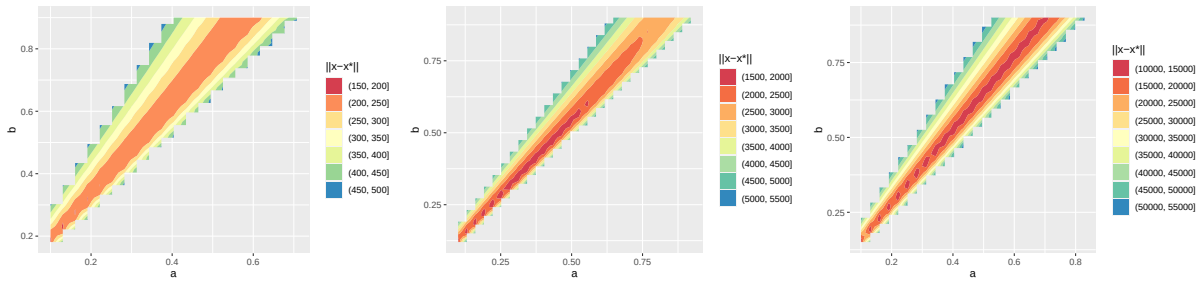

Figure 7: Contour plots of $||x - x^*||$, Ireland, Italy and United States

2.1.3 Forecasting

2.1.4 Implementation in R

2.1.5 Plots

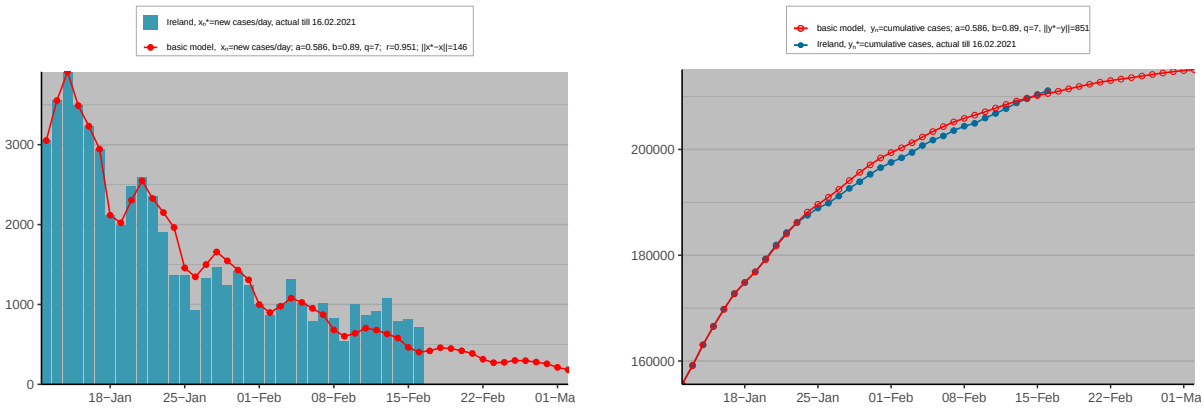

Figure 8: Basic model, Ireland

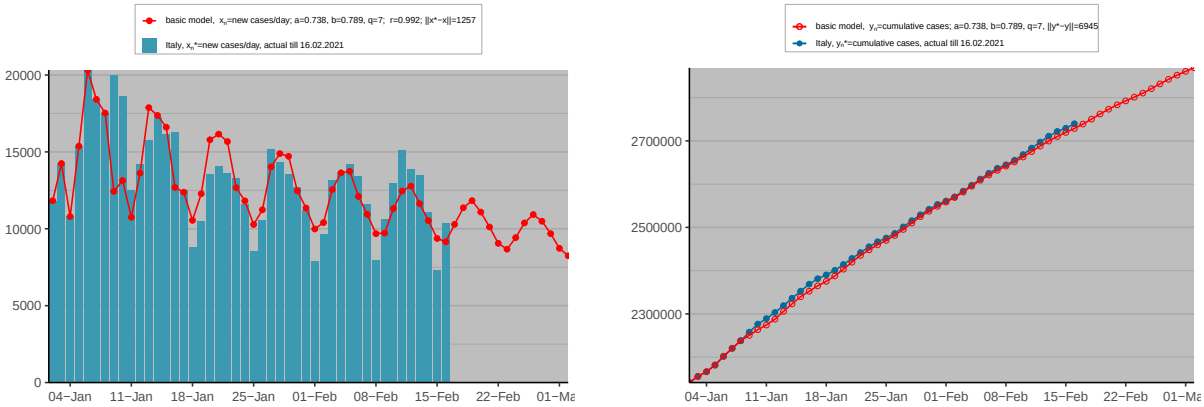

Figure 9: Basic model, Italy

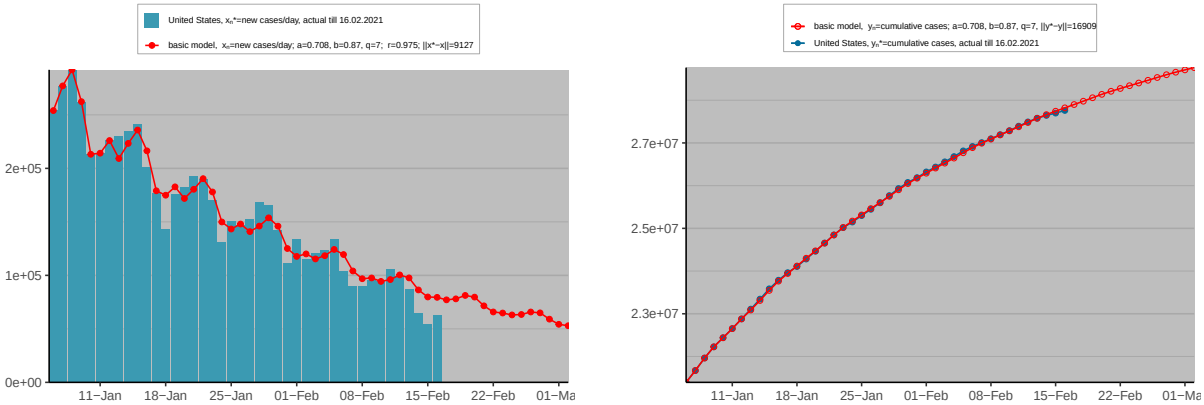

Figure 10: Basic model, United States

2.2 Limiting curve

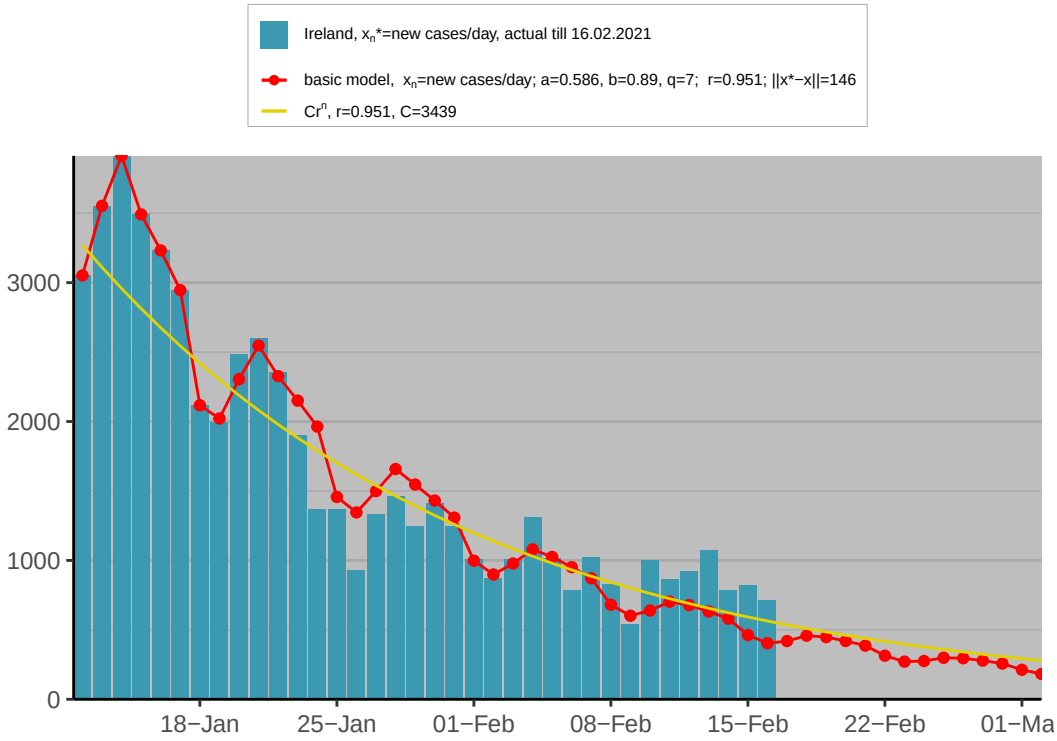

Figure 11: Comparison of $x_n^*,\ x_n$ and the limiting curve Cr^n, Ireland

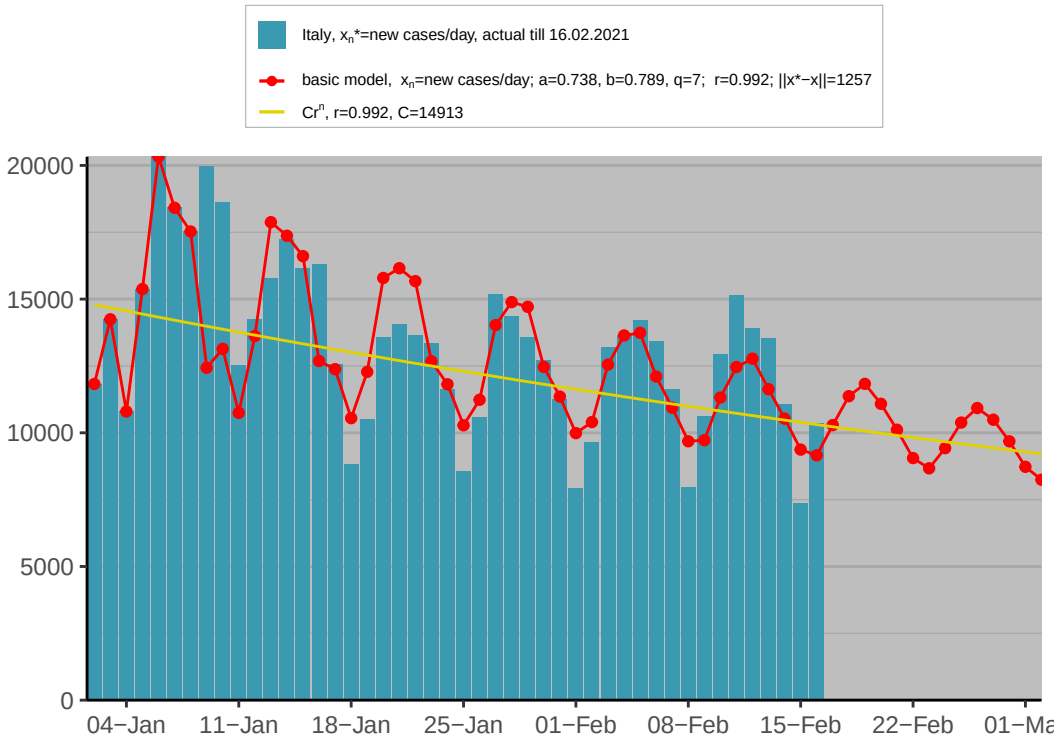

Figure 12: Comparison of $x_n^*,\ x_n$ and the limiting curve Cr^n, Italy

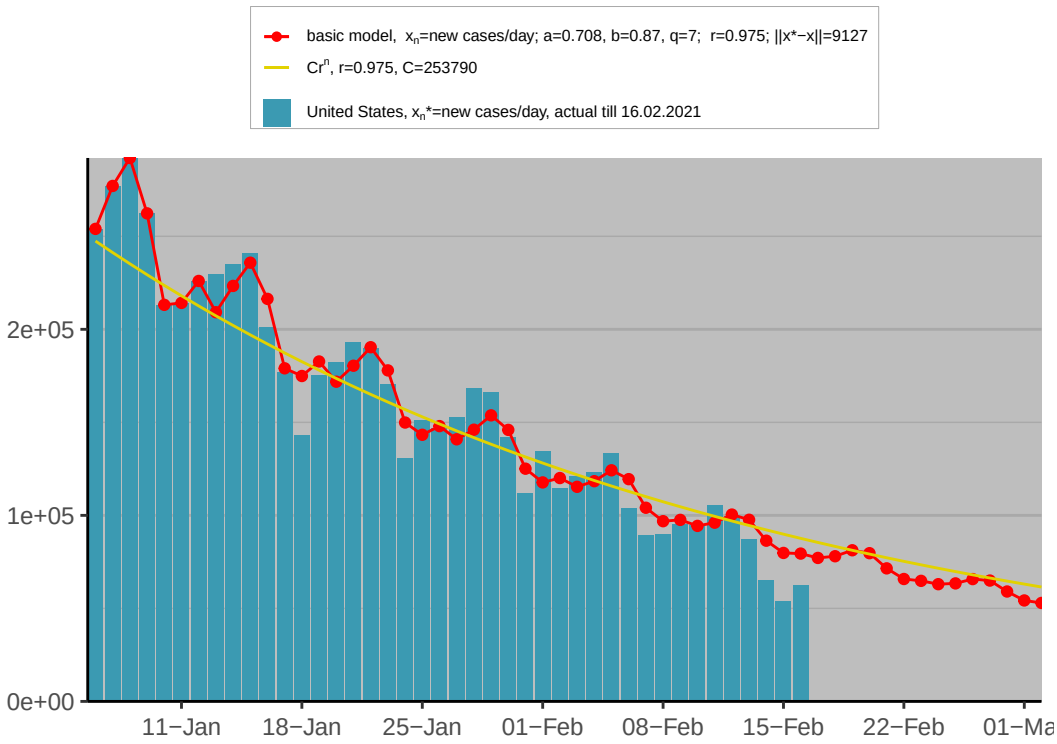

Figure 13: Comparison of $x_n^*,\ x_n$ and the limiting curve Cr^n, United States

The limiting curve can of course so exponential growth

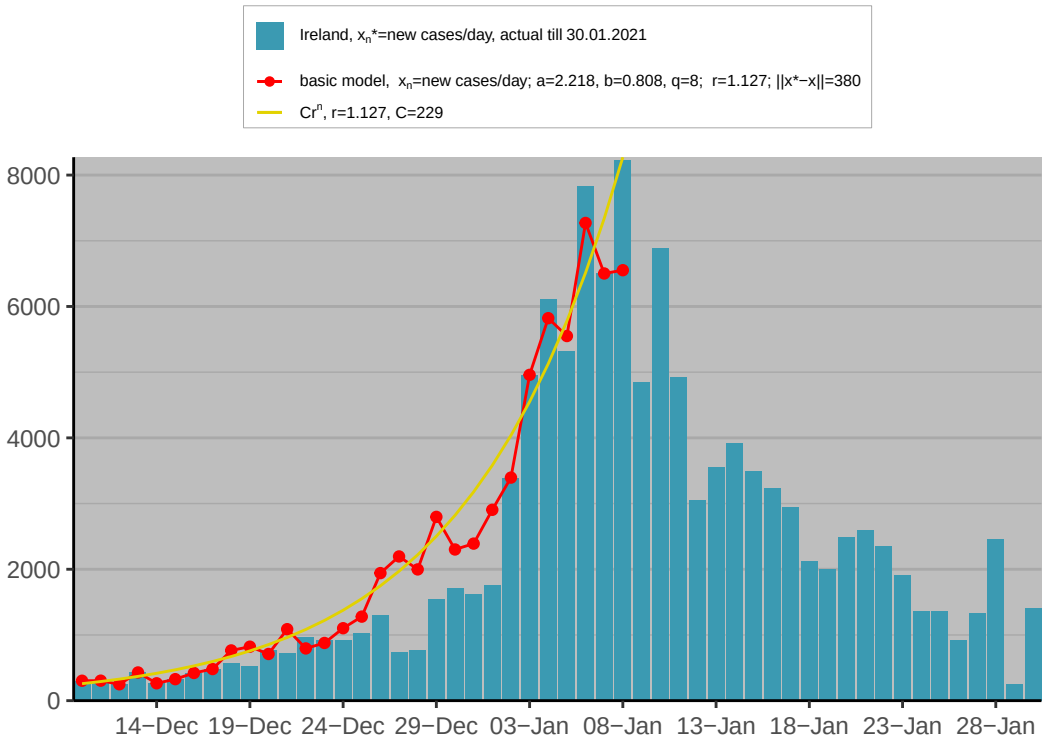

Figure 14: Limiting Curve Growing Exponentially, Ireland

2.3 Moving average

Define the $2k+1$-day moving average of actual data x_n^* by $x^*(k)$

$$x_n^*(1) = \frac{x_{n-1}^* + x_n^* + x_{n+1}^*}{3}, \quad 1 \le n < N$$

$$x_0^*(1) = \frac{x_0^* + x_1^*}{2}$$

$$x_N^*(1) = \frac{x_{N-1}^* + x_N^*}{2}$$

And then

$$x_n^*(3) := x_n^*(x_n^*(1))$$

is the 7-day moving average of cases.

Good baseline for model performance
(want $||x - x^*|| \approx ||x^*(3) - x^*||$ or better)

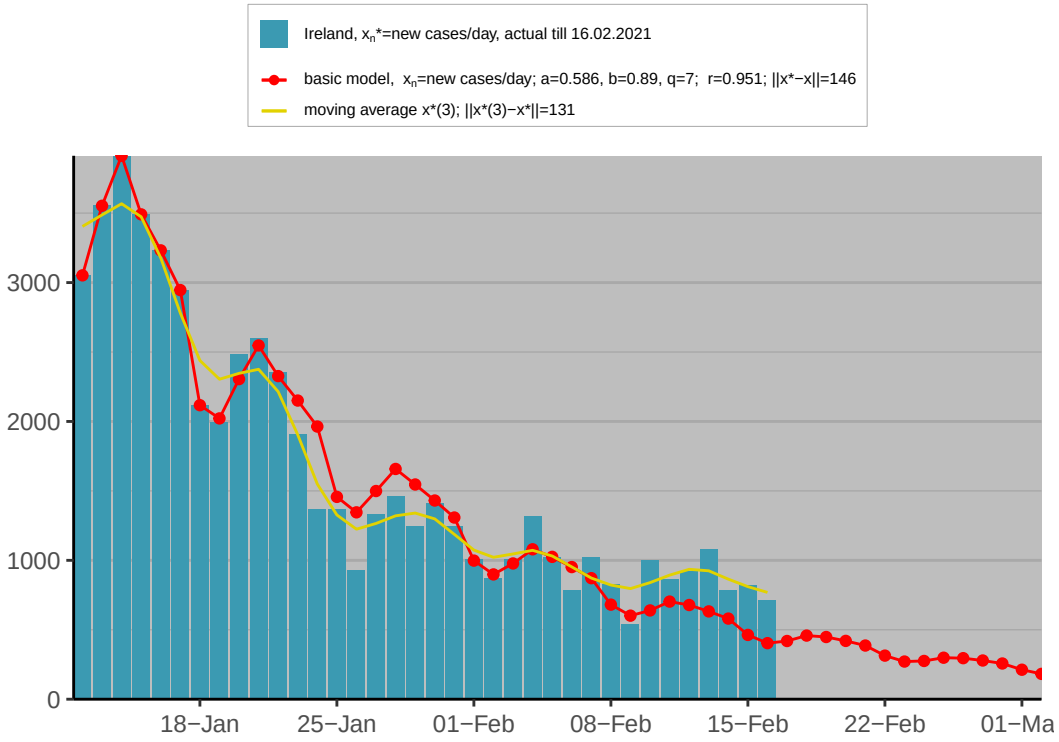

Figure 15: Moving average $x_n^*(3)$, Ireland

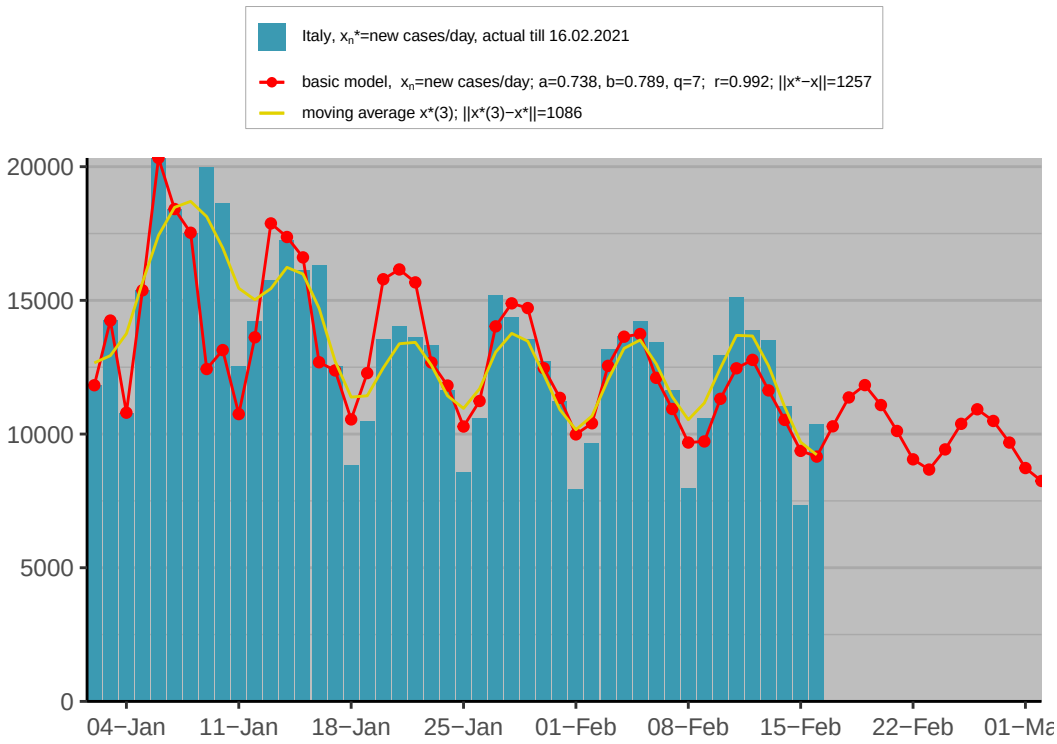

Figure 16: Moving average $x_n^*(3)$, Italy

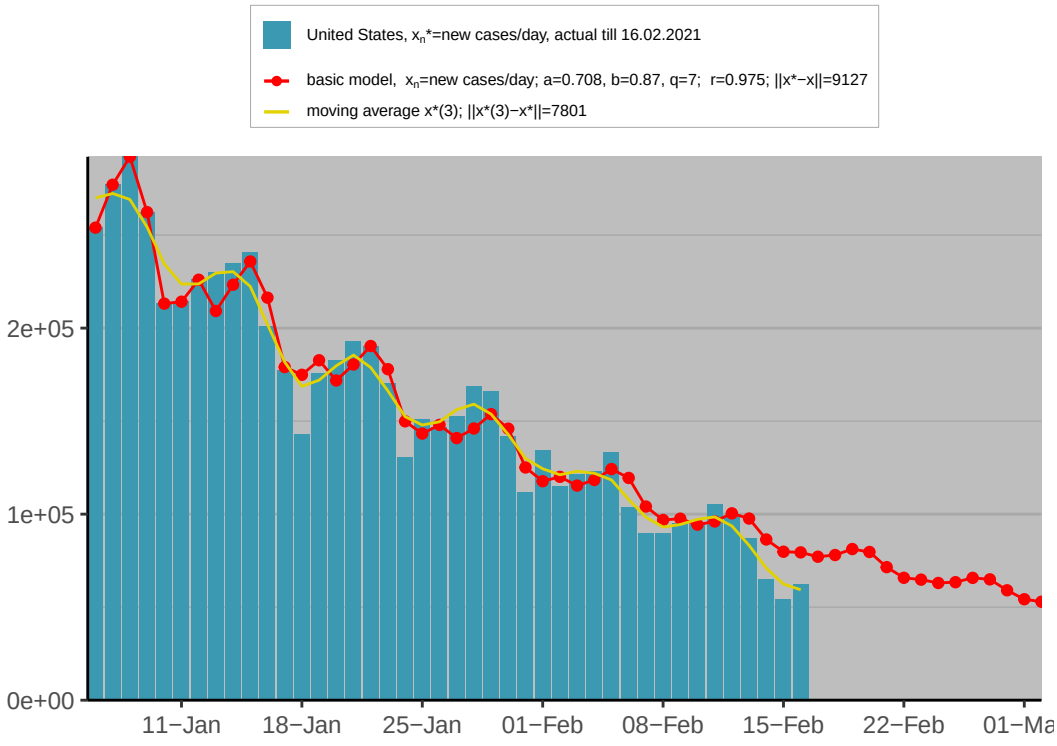

Figure 17: Moving average $x_n^*(3)$, United States

2.4 Periodic model

Instead of constant parameters a, b, we vary them slightly over time:

$a_n := a\left(1 + c_1\left(\sin\left(\frac{2\pi}{p_1}\left(n - n_1\right)\right)\right)\right)$

$b_n := b\left(1 + c_2\left(\sin\left(\frac{2\pi}{p_2}\left(n - n_2\right)\right)\right)\right)$

For new parameters $c_i, p_i, n_i,\ i = 1, 2$ where

$c_i \in [0.04, 0.2]$ small

$n_i \in 1, 2, \ldots, q$

$p_i \in 1, 2, \ldots, q$

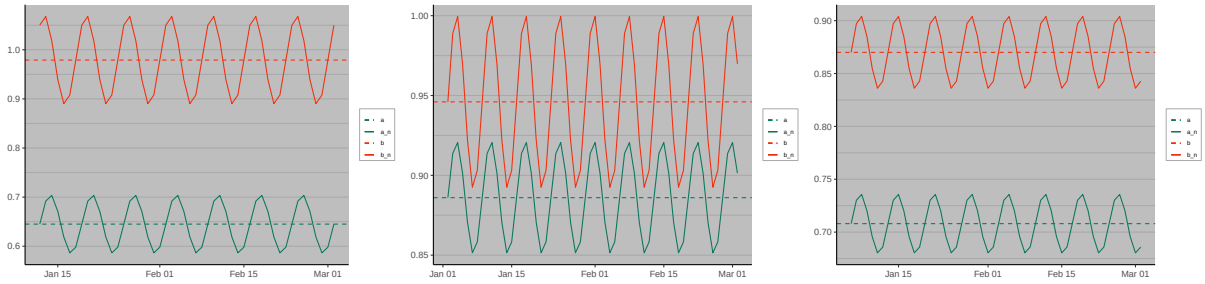

Figure 18: Oscillating a and b parameters, Ireland, Italy and United States

2.4.1 Definitions and Theory

2.4.2 How to select the best model

2.4.3 Forecasting

2.4.4 Implementation in R

```
modxper <- function(par, q, x, len = 0){
  #a,b,c1,c2,p1,p2,n1,n2
  an   <- par[1]*(1+par[3]*sin(2*pi*(1:(length(x)+len) - par[7])/par[5]))
  bn   <- par[2]*(1+par[4]*sin(2*pi*(1:(length(x)+len) - par[8])/par[6]))
  modx <- x[1:q]
  for(i in (q+1):(length(x)+len)){
    modx[i] <- (bn[i]*(1-bn[i-1]))*modx[i-1]/bn[i-1] +
      (an[i-q]*bn[i])*modx[i-q]/bn[i-q]
  }
  return(modx)
}

normper <- function(par, q, x) return(modnorm(x,modxper(par,q,x)))

#parameters of the form (ci,pi,ni)
  ##an = a(1 + c1 sin(2pi/p1 (n - n1)))
  ##bn = b(1 + c2 sin(2pi/p2 (n - n2)))
  aseqper <- a*seq(from = 0.8, to = 1.2, length.out = 5)
  bseqper <- b*seq(from = 0.8, to = 1.2, length.out = 5)
  c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 10)
  n_1seq  <- n_2seq <- c(1,7)
  p_1seq  <- p_2seq <- 6:7
  normdatp <- expand.grid(a  = aseqper, b  = bseqper,
                          c1 = c_1seq,  c2 = c_2seq,
                          p1 = p_1seq,  p2 = p_2seq,
                          n1 = n_1seq,  n2 = n_2seq)

  pernorm <- apply(normdatp, 1, function(x) normper(x, q = q, countrydat$xn))
  normdatp$pernorm <- pernorm

  peroptim     <- normdatp[which.min(pernorm),1:8]
  optpernorm <- normdatp[which.min(pernorm),9]

  modeldat$periodic  <- modxper(as.numeric(peroptim),countrydat$xn, q = q, forecastlen)
  modeldat$periodicy <- xntoyn(modeldat$periodic) + prevcases
```

```
36
37    optpernormy <- modnorm(countrydat$yn, modeldat$periodicy[1:nrow(countrydat)])
38
39    peroptim <- round(as.numeric(peroptim),3)
40
41    #a,b,c1,c2,p1,p2,n1,n2
42    labs$periodic <- list(bquote("periodic model,"~x[n]*"=new cases/day;"
43                                 ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
44                                 ~q*"="*.(q)*";""~"||x*-x||="*.(optpernorm)*";"
45                                 ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
46                                 ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
47                                 ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))
48
49    labs$periodicy <- list(bquote("periodic model,"~y[n]*"=new cases/day;"
50                                  ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
51                                  ~q*"="*.(q)*";""~"||x*-x||="*.(optpernormy)*";"
52                                  ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
53                                  ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
54                                  ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))
55
56    plots[["periodic"]] <- plot_periodic(countrydat, modeldat, cols, labs)
57
58    plots[["periodicy"]] <- plot_periodicy(countrydat, modeldat, cols, labs)
```

Listing 1: Algorithm for Periodic Model

and our plotting function for daily cases looks like the following

```
1  plot_periodic <- function(countrydat, modeldat, cols, labs){
2    p <- ggplot(countrydat, binwidth = 0) +
3      geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
4      geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
5      geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
6      geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
7      geom_point(data = modeldat,aes(x = date, y = periodic, colour = "periodic")) +
8      geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
9      gg_scale_xy +
10     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
11            fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
12     scale_fill_manual(values = cols$xn, labels = labs$xn) +
13     scale_colour_manual(values = c("base" = cols$basexn, "periodic" = cols$periodic, "x3" = cols$
           x3),
14                         labels = c("base" = labs$basexn, "periodic" = labs$periodic, "x3" = labs$
                               x3)) +
15     guides(colour = guide_legend(override.aes = list(shape = c("base" = 16, "periodic" = 16, "x3"
           = NA)))) +
16     xntheme()
17   return(p)
18 }
```

Listing 2: Plot Periodic Model

2.4.5 Plots

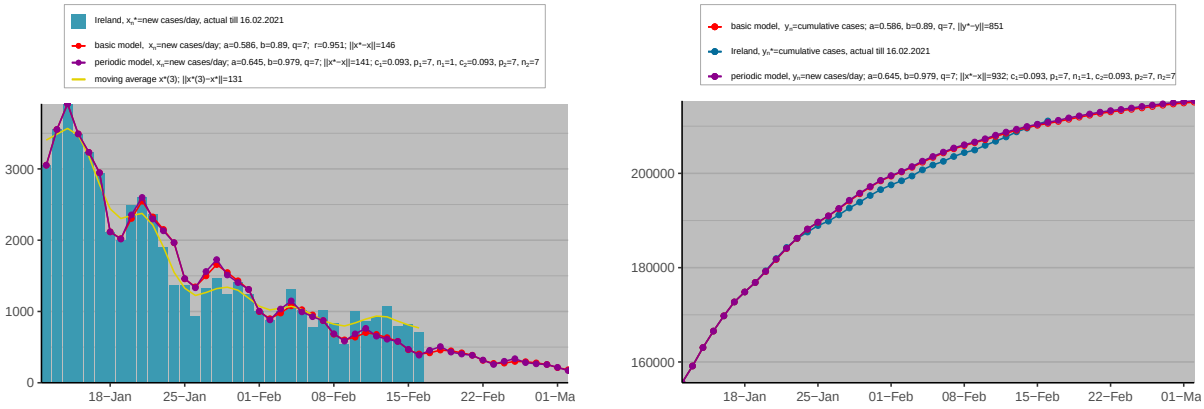

Figure 19: Periodic model, Ireland

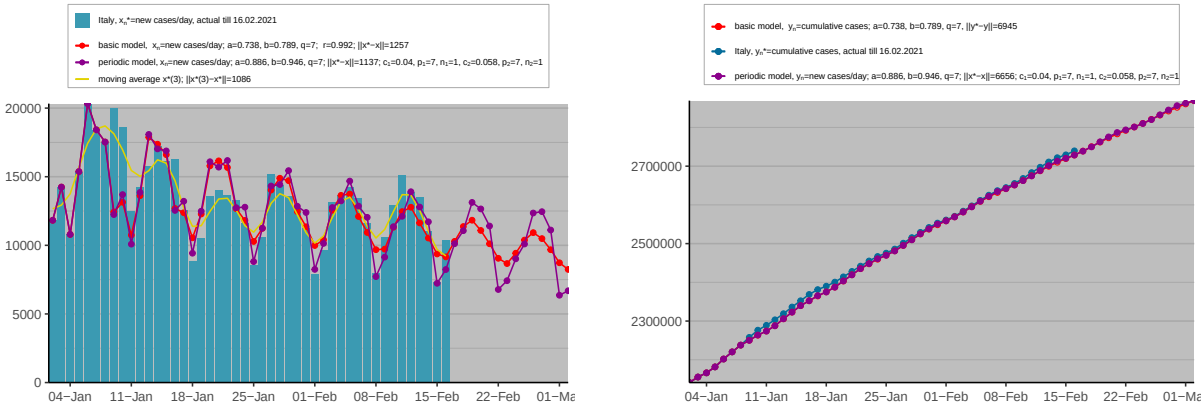

Figure 20: Periodic model, Italy

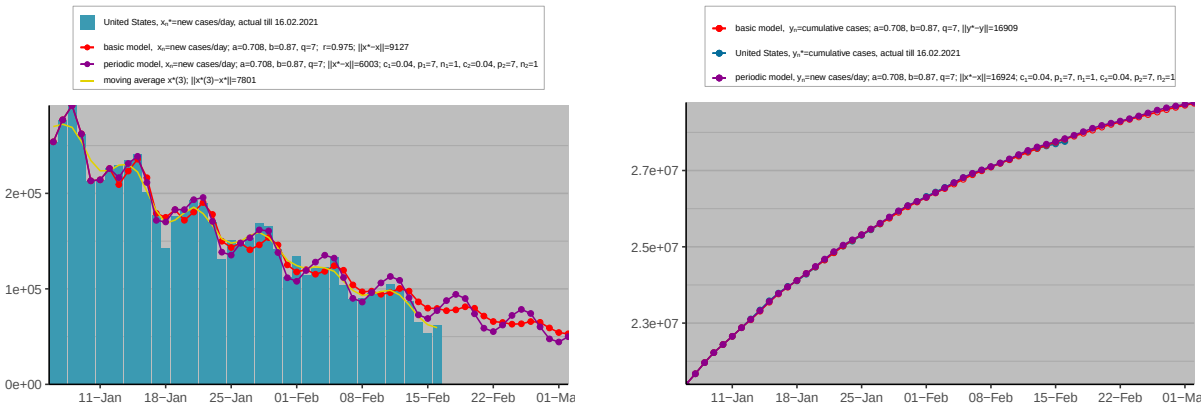

Figure 21: Periodic model, United States

2.5 Multi-phase model

2.5.1 Definitions and Theory

2.5.2 How to select the best model

2.5.3 Forecasting

2.5.4 Implementation in R

2.5.5 Plots

The multi-phase model without periodicity can be sharp and unrealstic

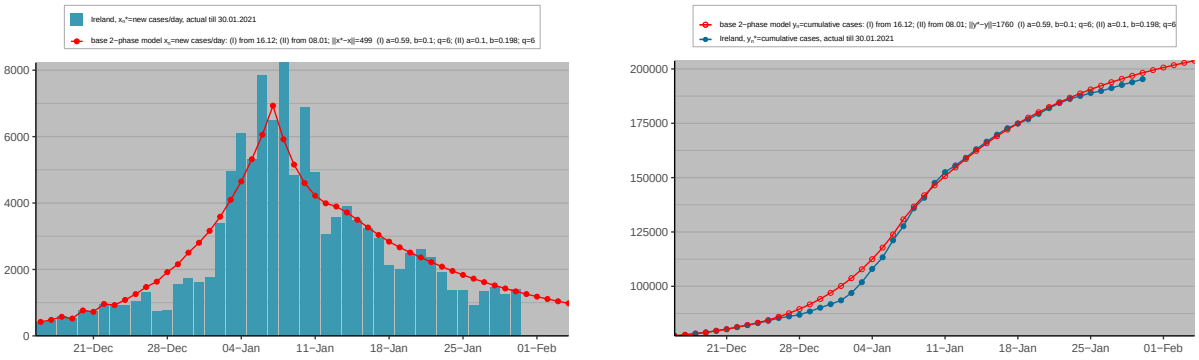

Figure 22: Multi-phase model, Ireland

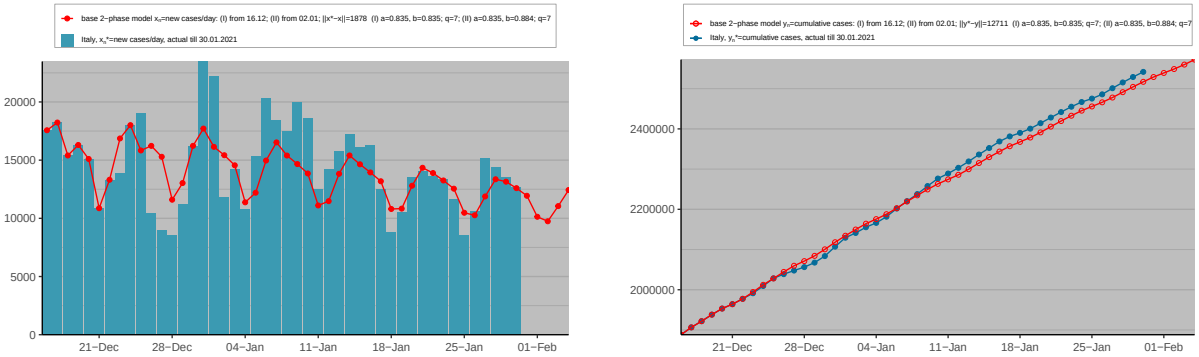

Figure 23: Multi-phase model, Italy

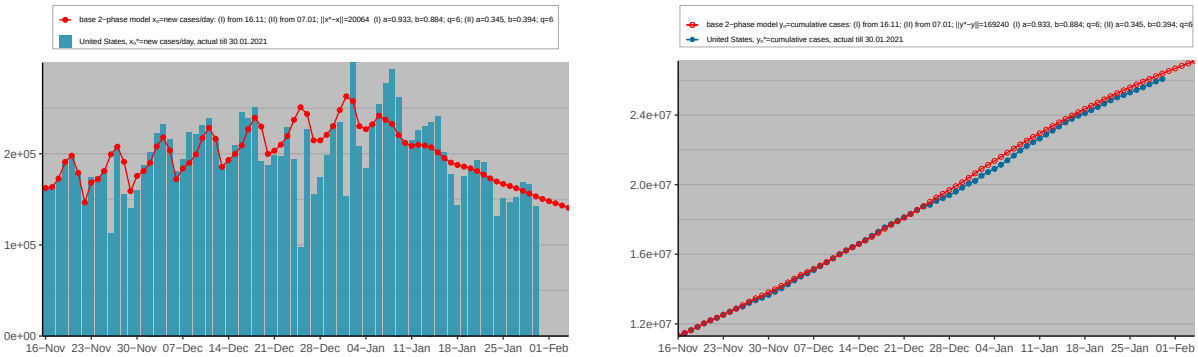

Figure 24: Multi-phase model, United States

The periodic model often performs much better

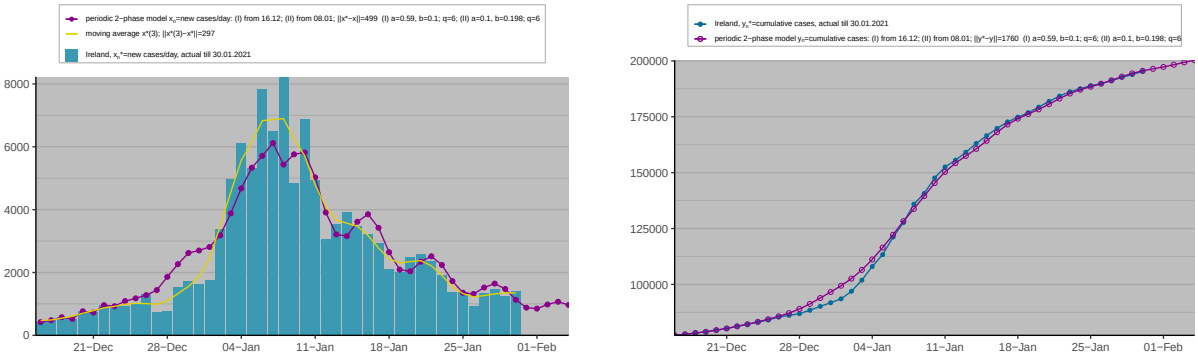

Figure 25: Multi-phase periodic model, Ireland

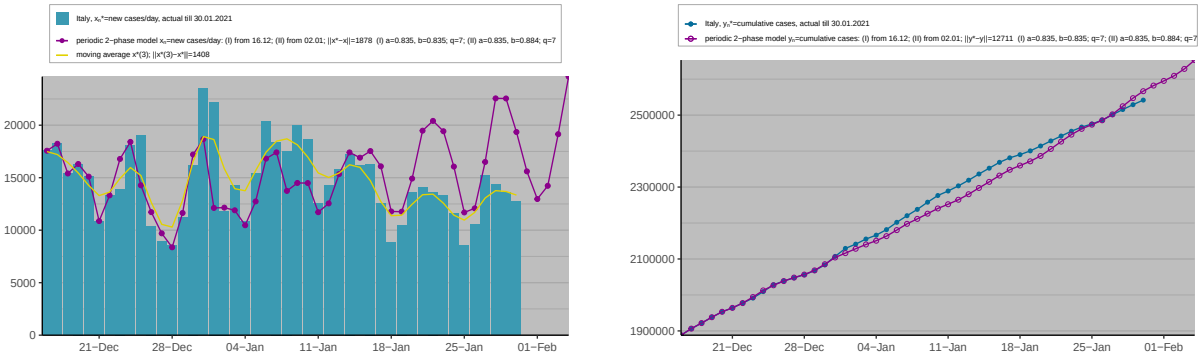

Figure 26: Multi-phase periodic model, Italy

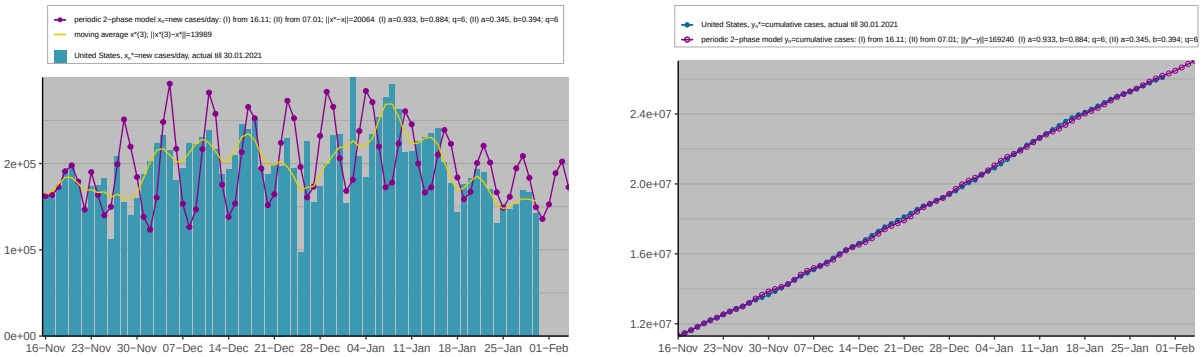

Figure 27: Multi-phase periodic model, United States

3 Theorems for Mathematical Model

3.1 Model Assumptions

(I) Any infected person becomes ill (symptomatic) and infectious on the q-th day after infection.[1]

(A) During each day, each ill person unconfined infects on average a other persons.

(B) During each day, a fraction b of ill people loose gets isolated (hospitalized or otherwise) and withdrawn from a further spread of the epidemic.

Many models use a set of differential equations for to describe the movement of people between *groups* or *compartments*[21, 4, 11]. The SIR (Susceptible–Infectious–Recovered) model, the most frequently used model in epidemiology, uses a set of 3 such differential equations [3, 5].

Our main mathematical model (and even some of the statistical models) make use recurrence equations, which have some correspondence to differential equations [1].

3.2 Notation

- x_n - the number of infected people that are detected and isolated during the day n;
- y_n – the cumulative number of detected cases from the beginning of epidemic by the beginning of the day n;
- z_n – the number of ill people at large by the beginning of the day n (that is, those who were infected at least q days ago and stay unisolated);
- u_n – the number of people newly infected during the day n.

We will obtain the following relation between the leading root r and the basic reproductive rate R_0 that is a main characteristic of an epidemic in epidemiology:

$$r \approx R_0^{\frac{1}{2q}}. \tag{1}$$

Recurrence relation for z_n:

$$z_{n+1} = z_n - x_n + u_{n-q}. \tag{2}$$

Using $x_n = bz_n$ we obtain the following equation for x_n:

$$x_{n+1} = (1-b)x_n + ax_{n-q}. \tag{3}$$

We let the model equal the actual data for the first $q+1$ days

$$x_n = x_n^* \text{ for } n = 0, 1, \ldots, q, \tag{4}$$

To fit our model we optimize against the normalized 1-norm:

$$||x - x^*|| := \frac{1}{N+1}\sum_{n=0}^{N} |x_n - x_n^*|, \tag{5}$$

Similarly we define $||y - y^*||$
In order to determine values a, b, q, we want to minimize both

$$||x - x^*|| \text{ and } ||y - y^*|| \tag{6}$$

3.2.1 Why minimize both diatances?

Do 3 pairs of plots: - xn/yn for just x-norm - xn/yn for just y-norm - xn/yn for both x-norm and y-norm

3.2.2 Recurrence equation

This is our general linear recurrence equation with constant coefficients:

$$x_{n+1} = a_0x_n + a_1x_{n-1} + a_2x_{n-2} + \cdots + a_qx_{n-q} \tag{7}$$

The characteristic polynomial of 7

$$f(\lambda) = \lambda^{q+1} - a_0\lambda^q - a_1\lambda^{q-1} - a_2\lambda^{q-2} - \cdots - a_{q-1}\lambda - a_q. \tag{8}$$

Definition 1. A root λ of f with the maximal absolute value $|\lambda|$ will be referred to as a leading root of the general linear recurrence relation 7.

[1] The number of days before an infected person becomes infectious is called the latent period, and before he/she becomes symptomatically ill – the incubation period. Here we assume for simplicity that these two periods are equal.

3.3 Theorems

Theorem 1. Let $a_k \geq 0$ for all $k \in \{0, \ldots, q\}$ and $a_{k_0} > 0$ for some $k_0 \in \{0, \ldots, q\}$.

(a) (Cauchy, 1829) The polynomial $f(\lambda)$ from 8 has exactly one positive real root r. Besides, the root r is simple and, for any other root $\lambda \in \mathbb{C}$, we have $|\lambda| < r$. Consequently, r is the leading root of 7.

(b) For any positive solution x_n of 7, there exists $C > 0$ such that

$$x_n \sim Cr^n \text{ as } n \to \infty. \tag{9}$$

It follows from 9 that if $r < 1$ then the epidemic fades away, whereas if $r > 1$ then it spreads unlimited.

Proof:

(a) Although this statement is not new, we give here the proof as it is quite simple and a part of the argument will be used below. The equation $f(\lambda) = 0$ is equivalent to

$$0 = \lambda^{q+1} - a_0\lambda^q - a_1\lambda^{q-1} - a_2\lambda^{q-2} - \cdots - a_{q-1}\lambda - a_q$$

dividing across by λ^{q+1}

$$= 1 - \frac{a_0}{\lambda} - \frac{a_1}{\lambda^2} - \frac{a_2}{\lambda^3} - \cdots - \frac{a_{q-1}}{\lambda^q} - \frac{a_q}{\lambda^{q+1}}$$

And so

$$1 = \underbrace{\frac{a_0}{\lambda} + \frac{a_1}{\lambda^2} + \frac{a_2}{\lambda^3} + \cdots + \frac{a_{q-1}}{\lambda^q} + \frac{a_q}{\lambda^{q+1}}}_{g(\lambda)} \tag{10}$$

Since $a_{k_0} > 0$ for some k_0, and the remaining a_k are non-negative, $g(\lambda)$ is strictly monotone decreasing in $\lambda > 0$ (if $c\lambda$ is increasing,then $\frac{c}{\lambda}$ is decreasing), and we have the limits

- $\lim_{\lambda \to 0^+} g(\lambda) = +\infty$
- $\lim_{\lambda \to +\infty} g(\lambda) = 0^+$

Hence, there is exactly one positive value $\lambda = r$ that satisfies this $g(r) = 1$, that is,

$$1 = \frac{a_0}{r} + \frac{a_1}{r^2} + \frac{a_2}{r^3} + \cdots + \frac{a_{q-1}}{r^q} + \frac{a_q}{r^{q+1}}.$$

Now, let $\lambda \in \mathbb{C}\backslash\{0\}$ be another root of f. We obtain from 10 (using the triangle inequality) that

$$1 \leq \frac{a_0}{|\lambda|} + \frac{a_1}{|\lambda|^2} + \frac{a_2}{|\lambda|^3} + \cdots + \frac{a_{q-1}}{|\lambda|^q} + \frac{a_q}{|\lambda|^{q+1}}$$

And so $g(r) \leq g(|\lambda|)$ which implies $|\lambda| \leq r$ by the definition of decreasing functions.

We next need to show that the root r is simple. Denote by r' the largest non-negative root of the derivative $f'(\lambda)$ that exists for the following reason. If $a_k > 0$ for some $k < q$ then the polynomial $\frac{1}{q+1}f'(\lambda)$ satisfies the hypotheses of the present theorem and, by the above argument, $f'(\lambda)$ has exactly one positive root, that is r'. If $a_k = 0$ for all $k < q$ then $f'(\lambda) = (q+1)\lambda^q$ has the only root 0, and, hence, $r' = 0$.

Let us verify that $r' < r$, which will also imply that r is simple. If $r' = 0$ then it is clear. If $r' > 0$ then it follows from $f'(r') = 0$ that

$$\begin{aligned}
f'(\lambda) &= (q+1)\lambda^q - qa_0\lambda^{q-1} - (q-1)a_1\lambda^{q-2} - (q-2)a_2\lambda^{q-3} - \cdots - a_{q-1} - 0 \\
\frac{1}{q+1}f'(\lambda) &= \lambda^q - \frac{q}{q+1}a_0\lambda^{q-1} - \frac{q-1}{q+1}a_1\lambda^{q-2} - \frac{q-2}{q+1}a_2\lambda^{q-3} - \cdots - \frac{1}{q+1}a_{q-1} \\
\frac{1}{q+1}f'(r') &= (r')^q - \frac{q}{q+1}a_0(r')^{q-1} - \frac{q-1}{q+1}a_1(r')^{q-2} - \frac{q-2}{q+1}a_2(r')^{q-3} - \cdots - \frac{1}{q+1}a_{q-1} \\
0 &= (r')^q - \frac{q}{q+1}a_0(r')^{q-1} - \frac{q-1}{q+1}a_1(r')^{q-2} - \frac{q-2}{q+1}a_2(r')^{q-3} - \cdots - \frac{1}{q+1}a_{q-1} \\
(r')^q &= \frac{q}{q+1}a_0(r')^{q-1} + \frac{q-1}{q+1}a_1(r')^{q-2} + \frac{q-2}{q+1}a_2(r')^{q-3} + \cdots + \frac{1}{q+1}a_{q-1}
\end{aligned}$$

dividing both sides by $(r')^q > 0$

$$\begin{aligned}
1 &= \frac{qa_0}{(q+1)r'} + \frac{(q-1)a_1}{(q+1)(r')^2} + \cdots + \frac{a_{q-1}}{(q+1)(r')^q} \\
&= \left(\frac{q+1-1}{q+1}\right)\frac{a_0}{r'} + \left(\frac{q+1-2}{q+1}\right)\frac{a_1}{(r')^2} + \cdots + \left(\frac{q+1-q}{q+1}\right)\frac{a_{q-1}}{(r')^q} \\
&= \left(1 - \frac{1}{q+1}\right)\frac{a_0}{r'} + \left(1 - \frac{2}{q+1}\right)\frac{a_1}{(r')^2} + \cdots + \left(1 - \frac{q}{q+1}\right)\frac{a_{q-1}}{(r')^q} \\
&< \frac{a_0}{r'} + \frac{a_1}{(r')^2} + \cdots + \frac{a_{q-1}}{(r')^q}
\end{aligned}$$

So $g(r') > 1$, but $g(r) = 1$
$\implies g(r') > g(r) \implies r' < r$ by the definition of decreasing functions.

(b) Let $\lambda_1, \lambda_2, \ldots$ be all other distinct roots of f apart from r (so that λ_k are negative or imaginary). Any solution x_n of 7 has the form

$$x_n + Cr^n + \tilde{x}_n \tag{11}$$

where $\tilde{x}_n$ is a linear combination of the functions $n^j \lambda_k^n$. Since by (a) we have $|\lambda_k| < r$, it follows that

$$|\tilde{x}_n| = o(r^n) \text{ as } n \to \infty \tag{12}$$

Since $x_n > 0$, it follows from 11 and 12 that $C \geq 0$. Let us verify that $C > 0$, which will finish the proof. It is tempting to say that if $C = 0$ then $x_n = \tilde{x}_n$ is a linear combination of terms of the form $n^j \rho^n \sin(\phi n)$ and $n^j \rho^n \cos(\phi n)$ and, therefore, cannot stay positive. However, it is not easy to make this argument rigorous because different roots of f may have the same absolute value ρ and an uncontrollable cancellation of the terms can occur. We employ here a different, simpler approach that takes advantage of nonnegative coefficients a_k. To that end, consider a new sequence

$$X_n = \frac{x_n}{r^n}.$$

This satisfies the equation

$$X_{n+1} = A_0 X_0 + A_1 X_{n-1} + \cdots + A_q X_{n-q} \tag{13}$$

with $A_k = \frac{a_k}{r^{k+1}}$. Since r is a root of f, we have

$$\begin{aligned} A_0 + A_1 + \cdots + A_q &= \frac{a_0}{r^1} + \frac{a_1}{r^2} + \cdots + \frac{a_q}{r^{q+1}} \\ &= g(r) \end{aligned}$$

This implies, by 10, and $g(r) = 1$ that

$$A_0 + A_1 + \cdots + A_q = 1 \tag{14}$$

Set $c := \min(X_1, \ldots, X_{q+1}) > 0$ since x_n have positive initial values. Then we obtain from 13 and 14 by induction that $X_n \geq c$ for all $n \in \mathbb{N}$, which implies
$x_n \geq cr^n$
as required.

□

Theorem 2. Let $a_k \geq 0$ for all $k = 0, \ldots, q$. Denote $a = a_1 + \cdots + a_q, b = 1 - a_0$ and assume that $a > 0, b > 0$.

(a) We have the equivalences: $r < 1 \iff a < b$ and $r > 1 \iff a > b$.

(b) Let $m \geq 1$ be such that $a_1 = \cdots = a_{m-1} = 0$ and $a_m > 0$. Then

$$\min\left(1, \left(\frac{a}{b}\right)^{1/m}\right) \leq r \leq \max\left(1, \left(\frac{a}{b}\right)^{1/m}\right) \tag{15}$$

Remark 1. Although there are in the literature plenty of estimates of the leading roots of polynomial (see, for example, [2]), none of them seems to imply 15. The latter is very useful for a basic model as we will see below in an example.

Proof:

(a) We have
$f(1) = 1 - a_0 - a1 - \cdots - a_q$
$= \underbrace{(1 - a_0)}_{b} - \underbrace{(a_1 + \cdots + a_q)}_{a}$
$= b - a$
We know f is increasing.
So if $r < 1$, we have $f(1) > 0$ and then $b - a > 0 \implies a < b$.
And if $r > 1$, we have $f(1) < 0$ and then $b - a < 0 \implies a > b$

(b) $f(r) = 0$ is equivalent to

$r^{q+1} - a_0 r^q - a_1 r^{q-1} - a_2 r^{q-2} - \cdots - a_{q-1} r - a_q = 0$

But any $a_1, \ldots, a_{m-1}$ are all zero

$\implies r^{q+1} - a_0 r^q - a_m r^{q-m} - a_{m+1} r^{q-m-1} - \cdots - a_{q-1} r - a_q = 0$

$\implies r^{q+1} - (1-b) r^q - a_m r^{q-m} - \cdots - a_q = 0$

$\implies r^{q+1} - r^q + b r^q - a_1 r^{q-m} - \cdots - a_q = 0$

$\implies r^{q+1} - r^q = -b r^q + a_m r^{q-m} + \cdots + a_q$

If $r > 1$ then $r^{q+1} > r^q$ and so $r^{q+1} - r^q > 0$

and so

$0 < -b r^q + a_m r^{q-m} + \cdots + a_q$

$\implies b r^q < a_m r^{q-m} + \cdots + a_q$

$\leq a_m r^{q-m} + \cdots + a_q r^{q-m}$

$= (a_m + \cdots + a_q) r^{q-m}$

$= a r^{q-m}$

So $b r^q < a r^{q-m} \iff r^m = \frac{a}{b} \iff r < \left(\frac{a}{b}\right)^{1/m}$

And if $r < 1$ we get $r < \left(\frac{a}{b}\right)^{1/m}$.

We can combine both cases with $a \leq \max(() 1, a)$ and $a \geq \min(1, a)$ to get 15, as required.

□

Lemma 3. For the model described by equation 7 we have

$$R_0 = \frac{a}{b}$$

Proof: Let u be the number of people infected on some day, say 0. On the day $k = 1, \ldots, q$ the number $c_k u$ of them become ill and can infect other people. On the day $k+1$ they infect $a c_k u$ people while $b c_k u$ of them get isolated. On the day $k+1$, the remaining $(1-b) c_k u$ people infect further $a(1-b) c_k u$ people. Continuing this way, we obtain that this group of $c_k u$ people infects in total

$$a c_k u + a(1-b) c_k u + a(1-b)^2 c_k u + \cdots = a c_k u \sum_{n=0}^{\infty} (1-b)^n = \frac{a c_k u}{1-(1-b)} = \frac{a}{b} c_k u$$

since $0 < 1 - b < 1$.

other people.

Hence, the initial group of u people infects in total

$$\sum_{k=0}^{q} \frac{a}{b} c_k u = \frac{a}{b} u \sum_{k=0}^{q} c_k = \frac{a}{b} u$$

So we know R_0 is the unit reprodiction number per infected person ($u = 1$).

And so we get the result $R_0 = \frac{a}{b}$ as required. □

4 Statistical Models

Primary source for this was Hyndman-et-al-2018 [19].

Some of our statistical models require *homoscedasticity*, i.e., that the model errors are identically distributed with the same variance σ^2.

We can check this by plotting histograms and checking that they are centred around zero and approximately fit the overlaying normal curve.

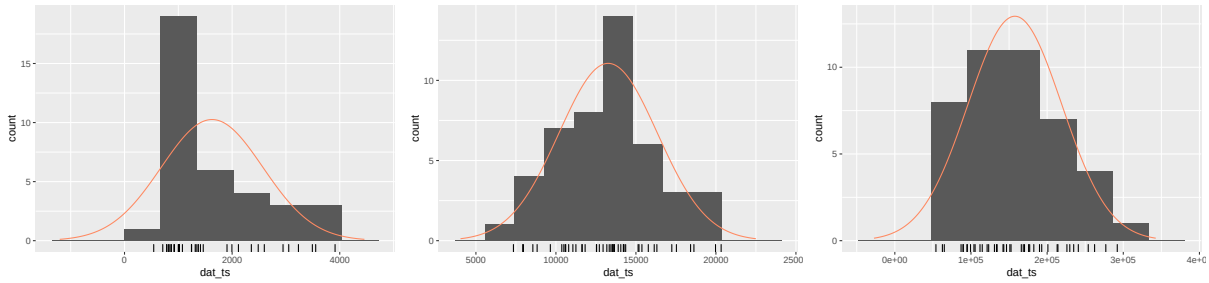

Figure 28: Normality checks, Ireland, Italy and United States

4.1 Holt-Winters' seasonal method

4.1.1 Definitions and Theory

Suppose there are N observations.
Initial step:

$$L_s = \frac{1}{s}\sum_{i=1}^{s} x_i$$
$$b_s = \frac{1}{s}\left[\frac{x_{s+1}-x_1}{s} + \frac{x_{s+2}-x_2}{s} + \cdots + \frac{x_{2s}-x_s}{s}\right]$$
$$S_n = x_n - L_s,\ n = 1, \ldots, s$$

and choose parameters $0 \leq \alpha \leq 1,\ 0 \leq \beta \leq 1$ and $0 \leq \gamma \leq 1$

Then compute for $s < n \leq N$:

Level	L_n	$= \alpha(x_n - S_{n-s}) + (1-\alpha)(L_{n-1} + b_{n-1})$
Trend	b_n	$= \beta(L_n - L_{n-1}) + (1-\beta)b_{n-1}$
Seasonal	S_n	$= \gamma(x_n - L_n) + (1-\gamma)S_{n-s}$
Forecast	F_{n+1}	$= L_n + b_n + S_{n+1-s}$

For subsequent observations,

$$F_{N+k} = L_N + k \cdot b_N + S_{N+k-s}$$

Figure 29: Seasonal Holt Winter's Additive Model Algorithm (denoted SHW_+)

4.1.2 How to select the best model

4.1.3 Forecasting

4.1.4 Implementation in R

```
#' dat_ts A time-series of the ovservations x_n
#' countrydat A data.frame with the required info to plot case numbers
#' modeldat A data.frame with model data to plot (has forecastlen more rows than countrydat)
#' cols A list of colours for plotting
#' labs A list of labels for plotting
#' forecastlen The forecast length
#' q The parameter for the basic model, in order tohave statistical models comparable with
    mathematical models
```

```
#' prevcases The cumulative cases prior to the first day in the range of countrydat

    #lambda=0 ensures values stay positive
    hwfcst     <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
    #ensure first q values are equal to the actual data
    hwfcst$fitted[1:q] <- countrydat$xn[1:q]
    modeldat$hwxn <- c(hwfcst$fitted, hwfcst$mean)
    modeldat$hwlo <- c(hwfcst$fitted, hwfcst$lower[,2])
    modeldat$hwhi <- c(hwfcst$fitted, hwfcst$upper[,2])

    modeldat$hwyn  <- xntoyn(modeldat$hwxn)+prevcases
    modeldat$hwylo <- xntoyn(modeldat$hwlo)+prevcases
    modeldat$hwyhi <- xntoyn(modeldat$hwhi)+prevcases

    hwnorm <- modnorm(countrydat$xn,hwfcst$fitted)

    labs$hw   <- paste0("HoltWinters algorithm,  ||x*-x||=", modnorm(countrydat$xn,hwfcst$fitted)
        )
    labs$hwy  <- paste0("HoltWinters algorithm,  ||y*-y||=", modnorm(countrydat$yn,modeldat$hwyn[
        1:nrow(countrydat)]))
    labs$hwpi <- "HW 95% Prediction Interval"

    plots[["hw"]]  <- plot_hw(countrydat, modeldat, cols, labs)
    plots[["hwy"]] <- plot_hwy(countrydat, modeldat, cols, labs)
```

Listing 3: Algorithm for HoltWinters Model

and our plotting function for daily cases looks like the following

```
plot_hw <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
    geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
    geom_ribbon(data = modeldat, aes(x = date, ymin = hwlo, ymax = hwhi, fill = "hw"), alpha = 0.
        5) +
    geom_point(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
    geom_line(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
    geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_point(data = modeldat,aes(x = date, y = modxPeriodic, colour = "periodic")) +
    geom_line(data = modeldat, aes(x = date, y = modxPeriodic, colour = "periodic")) +
    gg_scale_xy +
    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
           fill=guide_legend(ncol=1,nrow=2,byrow=TRUE)) +
    scale_fill_manual(labels = c("actual" = labs$xn, "hw" = labs$hwpi),
                      values = c("actual" = cols$xn, "hw" = cols$hwpi))+
    scale_colour_manual(labels = c("base" = labs$basexn, "hw" = labs$hw, "periodic" = labs$
        periodic),
                        values = c("base" = cols$basexn, "hw" = cols$hw, "periodic" = cols$
                            periodic)
                        ) +
    xntheme()
  return(p)
}
```

Listing 4: Plot HoltWinters Model

4.1.5 Plots

We see that the additive seasonal method is a better choice for both model fit and confidence interval size.

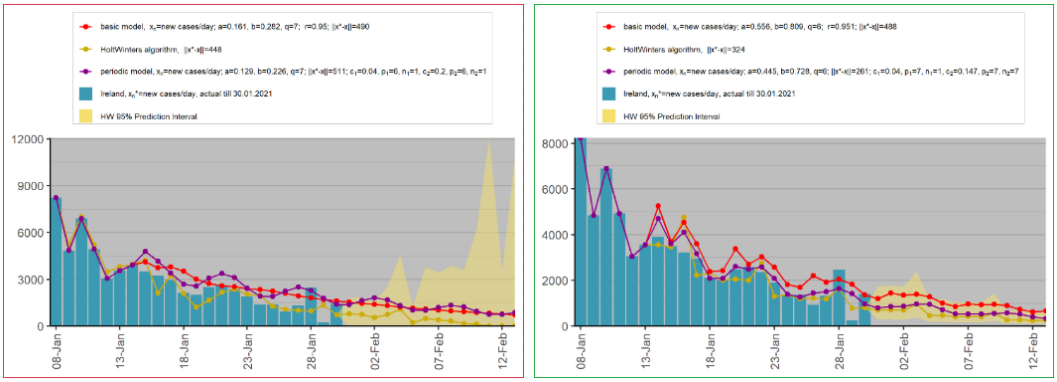

Figure 30: Comparison between HoltWinters multiplicative (left, red outline) and additive (right, green outline)algorithms, at some point during the research

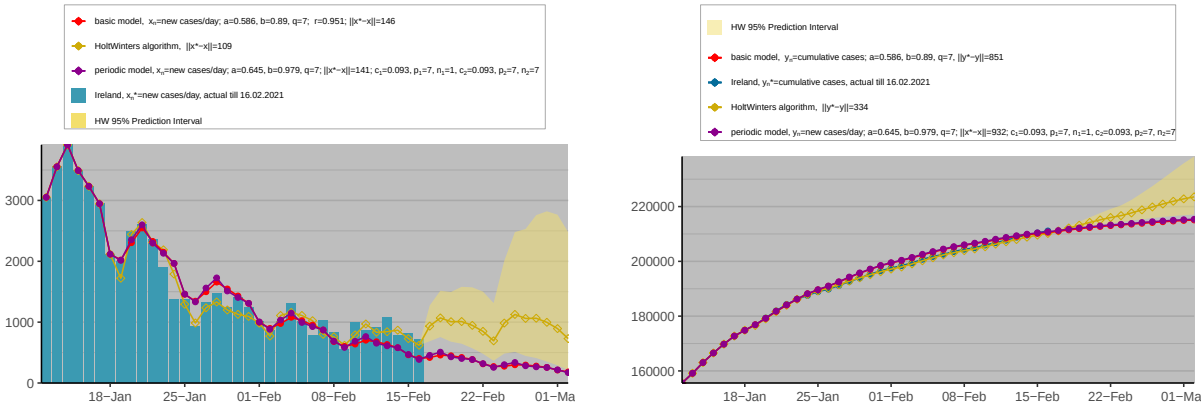

Figure 31: HoltWinters model, Ireland

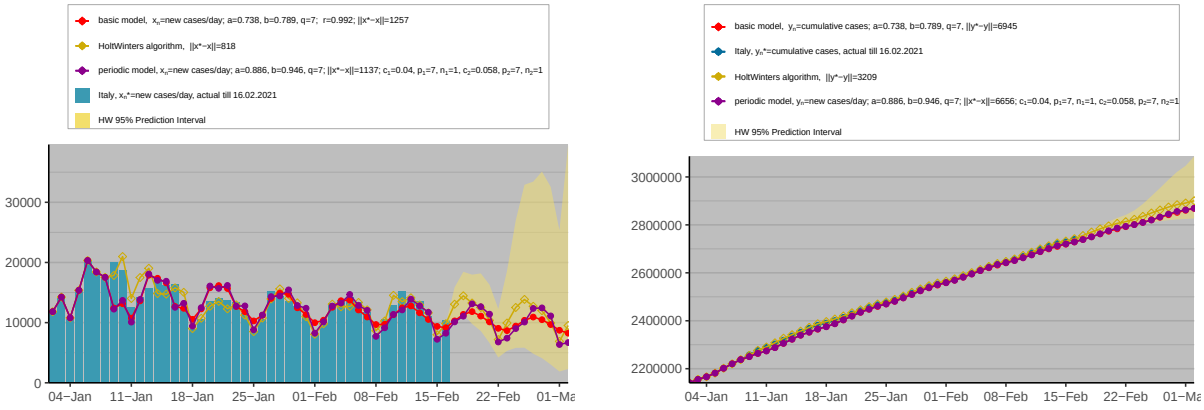

Figure 32: HoltWinters model, Italy

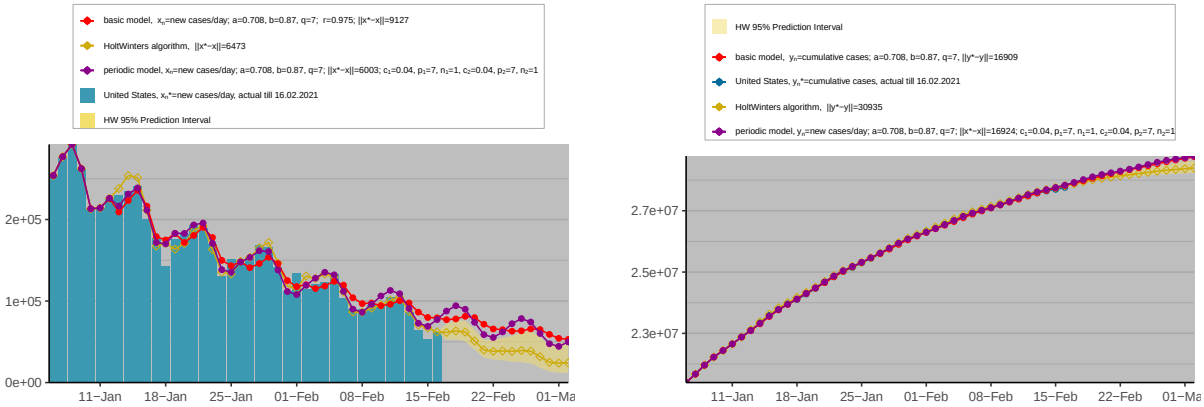

Figure 33: HoltWinters model, United States

4.2 ARIMA models

4.2.1 Definitions and Theory

Definition 2. The *backshift operator* B is a function on a time series $(x_n)_{n\geq 1}$ such that $Bx_n = x_{n-1}$ and more genrerally:

$$B^k x_n = x_{n-k}, \quad n > k$$

And similarly for the independent errors ε_n:

$$B^k \varepsilon_n = \varepsilon_{n-k}, \quad n > k$$

We must first define the each component of a non-seasonal ARIMA model (suitable for time series with a trend).

- An $AR(p)$ model, or an autoregressive model of order p of a time series $x_1, \ldots, x_N$ states that each x_n is a *linear function* of $x_{n-p}, x_{n-p+1}, \ldots, x_{n-1}$ and an error term, i.e.

  $$x_n = \phi_0 + \phi_1 x_{n-1} + \phi_2 x_{n-2} + \cdots + \phi_p x_{n-p} + \varepsilon_n, \quad n > p, \quad \varepsilon_n \sim N(0, \sigma^2)$$

  We can simplify using the backshift operator B:

  $$\begin{aligned} x_n &= \phi_0 + \phi_1 B x_n + \phi_2 B^2 x_n + \cdots + \phi_p B^p x_n + \varepsilon_n \\ &= \phi_0 + \left(\phi_1 B + \phi_2 B^2 + \cdots + \phi_p B^p\right) x_n + \varepsilon_n \end{aligned} \tag{16}$$

- An $MA(q)$ model, or a moving average model of order q of a time series $x_1, \ldots, x_N$ states that each x_n is a *linear function* of the q previous errors $\varepsilon_{n-q}, \varepsilon_{n-q+1}, \ldots, \varepsilon_{n-1}$, plus the current error ε_n, i.e.

  $$x_n = \psi_0 - \psi_1 \varepsilon_{n-1} - \psi_2 \varepsilon_{n-2} - \cdots - \psi_q \varepsilon_{n-p} + \varepsilon_n, \quad n > p$$

  By convention we use minus signs in the coefficients $\psi_1, \ldots, \psi_q$ We can simplify using the backshift operator B:

  $$\begin{aligned} x_n &= \psi_0 - \psi_1 B \varepsilon_n - \psi_2 B^2 \varepsilon_n - \cdots - \psi_q B^q \varepsilon_n + \varepsilon_n \\ &= \psi_0 + \left(1 - \psi_1 B - \psi_2 B^2 + \cdots - \psi_q B^q\right) \varepsilon_n \end{aligned} \tag{17}$$

- The first order differencing of the time series, $I(1)$, is evalueated as

  $$\begin{aligned} x_n' &= x_n - x_{n-1} \\ &= x_n - B x_n \\ &= (1 - B)\, x_n \end{aligned} \tag{18}$$

More generally, the differencing of order d, denoted $I(d)$ is

$$(1-B)^d x_n$$

This only affects the x_n (although constants are differenced to zero) and the errors ε_n are unchanged.

Therefore, an ARIMA(p,d,q) model can be evaluated by combining the $AR(p)$, $I(d)$ and $MA(q)$

$$(1-B)^d x_n = \phi_0 + (1-B)^d \left(\phi_1 B + \phi_2 B^2 + \cdots + \phi_p B^p\right) x_n + \psi_0 + \left(\psi_1 B + \psi_2 B^2 + \cdots + \psi_q B^q\right) \varepsilon_n$$

$$\begin{aligned}(1-B)^d x_n + (1-B)^d \left(-\phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right) x_n &= \phi_0 + \psi_0 + \left(1 - \psi_1 B - \psi_2 B^2 + \cdots - \psi_q B^q\right) \varepsilon_n \\ (1-B)^d \left(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right) x_n &= c + \left(1 - \psi_1 B - \psi_2 B^2 + \cdots - \psi_q B^q\right) \varepsilon_n \end{aligned} \tag{19}$$

where $c = \phi_0 + \psi_0$ (it is zero if $d \geq 1$).
We also need the seasonal components for an ARIMA$(p,d,q)(P,D,Q)_s$
Suppose a time series x_n has period s (seasonal pattern every s values)

- An $AR(P)_s$ model, or a seasonal autoregressive model of order P of a time series $x_1, \ldots, x_N$ states that each x_n is a *linear function* of $x_{n-Ps}, x_{n-(P-1)s}, \ldots, x_{n-s}$ and an error term, i.e.

  $$x_n = \beta_0 + \beta_1 x_{n-s} + \beta_2 x_{n-2s} + \cdots + \beta_P x_{n-Ps} + \varepsilon_n$$

  We can simplify using the backshift operator B:

  $$x_n = \beta_0 + \left(\beta_1 B^s + \beta_2 B^{2s} + \cdots + \beta_P B^{Ps}\right) x_n \tag{20}$$

- An $MA(Q)_s$ model, or a seasonal moving average model of order Q of a time series $x_1, \ldots, x_N$ states that each x_n is a *linear function* of the Q errors $\varepsilon_{n-Ws}, \varepsilon_{n-(Q-1)s}, \ldots, \varepsilon_{n-s}$, plus the current error ε_n, i.e.

  $$x_n = \gamma_0 - \gamma_1 \varepsilon_{n-s} - \gamma_2 \varepsilon_{n-2s} - \cdots - \gamma_Q \varepsilon_{n-Qs} + \varepsilon_n$$

  Again, by convention we use minus signs in the coefficients $\gamma_1, \ldots, \gamma_Q$
  We can simplify using the backshift operator B:

  $$\begin{aligned} x_n &= \gamma_0 - \gamma_1 \varepsilon_{n-s} - \gamma_2 \varepsilon_{n-2s} - \cdots - \gamma_Q \varepsilon_{n-Qs} + \varepsilon_n \\ &= \gamma_0 + \left(1 - \gamma_1 B^s - \gamma_2 B^{2s} + \cdots - \gamma_Q B^{Qs}\right) \varepsilon_n \end{aligned} \tag{21}$$

- The first order seasonal differencing of the time series, $I_s(1)$, is evalueated as

  $$x_n - x_{n-s} = (1 - B^s)\, x_n$$

  More generally, the seasonal differencing of order D, denoted $I_s(D)$ is

  $$(1 - B^s)^D x_n$$

  The purpose of this is to make the time series stationary in mean

Then we can similarly compose our seasonal components with the previous ARIMA(pd,q) to get the definition of an ARIMA$(p,d,q)(P,D,Q)_s$ model

$$\begin{aligned} &\underbrace{\left(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right)}_{AR(p)} \underbrace{\left(1 - \beta_1 B^s - \beta_2 B^{2s} - \cdots - \beta_P B^{Ps}\right)}_{AR_s(P)} \underbrace{(1-B)^d}_{I(d)} \underbrace{(1-B^s)^D}_{I_s(D)} x_n = \\ &\qquad c + \underbrace{\left(1 - \psi_1 B - \psi_2 B^2 - \cdots - \psi_q B^q\right)}_{MA(q)} \underbrace{\left(1 - \gamma_1 B^s - \gamma_2 B^{2s} - \cdots - \gamma_Q B^{Qs}\right)}_{MA_s(Q)} \varepsilon_n \end{aligned} \tag{22}$$

where the constant c is some function of the constants ϕ_0, ψ_0, β_0 and γ_0

4.2.2 How to select the best model

4.2.3 Forecasting

4.2.4 Implementation in R

```
#' dat_ts A time-series of the ovservations x_n
#' countrydat A data.frame with the required info to plot case numbers
#' modeldat A data.frame with model data to plot (has forecastlen more rows than countrydat)
#' cols A list of colours for plotting
#' labs A list of labels for plotting
#' forecastlen The forecast length
#' q The parameter for the basic model, in order tohave statistical models comparable with
    mathematical models
#' prevcases The cumulative cases prior to the first day in the range of countrydat

    #lambda=0 ensures values stay positive
    auto.fit <- auto.arima(dat_ts, lambda = 0)

    getArmaModel <- function(arma, pdq = c(1,6,2), PDQ = c(3,7,4), s=5){
      return(paste0("ARIMA(", paste0(arma[pdq],collapse = ","), ")(",
                  paste0(arma[PDQ], collapse = ","), ")[", arma[s], "]"))
    }

    arima.fcst <- forecast(auto.fit, level = c(80, 95), h = forecastlen)
    arima.fcst$fitted[1:q] <- countrydat$xn[1:q]

    arimanorm <- modnorm(countrydat$xn,arima.fcst$fitted)

    arimalabs    <- getArmaModel(auto.fit$arma)
    abs$arima   <- paste0(arimalabs, ", ||x*-x||=", arimanorm)
    abs$arimapi <- "ARIMA 95% Prediction Interval"

    modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
    modeldat$arimalo <- c(auto.fit$fitted,arima.fcst$lower[,2])
    modeldat$arimahi <- c(auto.fit$fitted,arima.fcst$upper[,2])

    modeldat$arimaxn[1:q] <- countrydat$xn[1:q]
    modeldat$arimalo[1:q] <- countrydat$xn[1:q]
    modeldat$arimahi[1:q] <- countrydat$xn[1:q]

    modeldat$arimayn  <- xntoyn(modeldat$arimaxn) + prevcases
    modeldat$arimaylo <- xntoyn(modeldat$arimalo) + prevcases
    modeldat$arimayhi <- xntoyn(modeldat$arimahi) + prevcases

    labs$arimay  <- paste0(arimalabs, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$arimayn[1:
        nrow(countrydat)]))

    plots[["arima"]]   <- plot_arima(countrydat, modeldat, cols, labs)
    plots[["arimay"]] <- plot_arimay(countrydat, modeldat, cols, labs)
```

Listing 5: Algorithm for ARIMA Model

and our plotting function for daily cases looks like the following

```
plot_arima <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
    geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
    geom_ribbon(data = modeldat, aes(x = date, ymin = arimalo, ymax = arimahi, fill = "pi"),
        alpha = 0.5) +
    geom_point(data = modeldat, aes(x = date, y = arimaxn, colour = "arima")) +
    geom_line(data = modeldat, aes(x = date, y = arimaxn, colour = "arima")) +
    geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_point(data = modeldat,aes(x = date, y = periodic, colour = "periodic")) +
    geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
    gg_scale_xy +
    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
           fill=guide_legend(ncol=1,nrow=2,byrow=TRUE)) +
    scale_fill_manual(values = c("actual" = cols$xn, "pi" = cols$arimapi),
                      labels = c("actual" = labs$xn, "pi" = labs$arimapi)) +
    scale_colour_manual(values = c("arima" = cols$arima, "base" = cols$basexn, "periodic" = cols$
        periodic),
```

```
                    labels = c("arima" = labs$arima, "base" = labs$basexn, "periodic" = labs$
                        periodic)) +
    xntheme()
  return(p)
}
```

Listing 6: Plot ARIMA Model

4.2.5 Plots

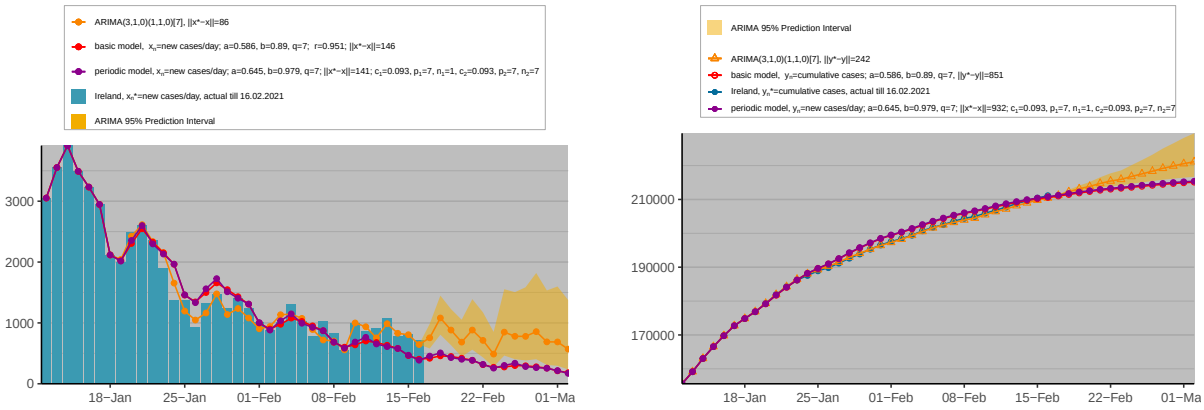

Figure 34: ARIMA model, Ireland

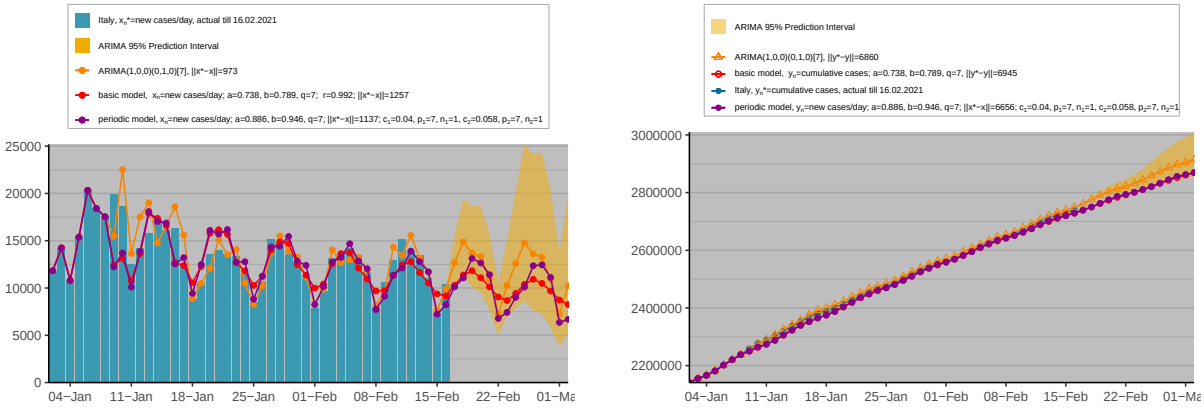

Figure 35: ARIMA model, Italy

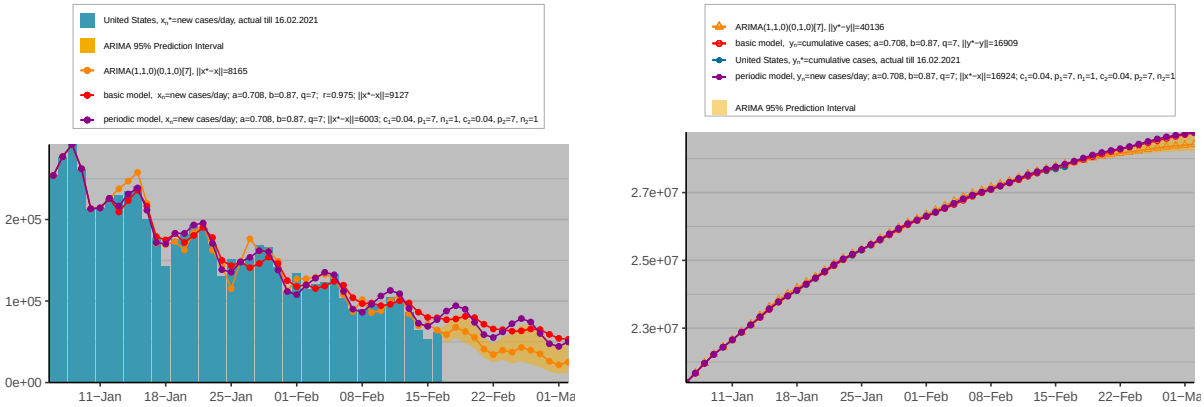

Figure 36: ARIMA model, United States

4.3 Neural network models

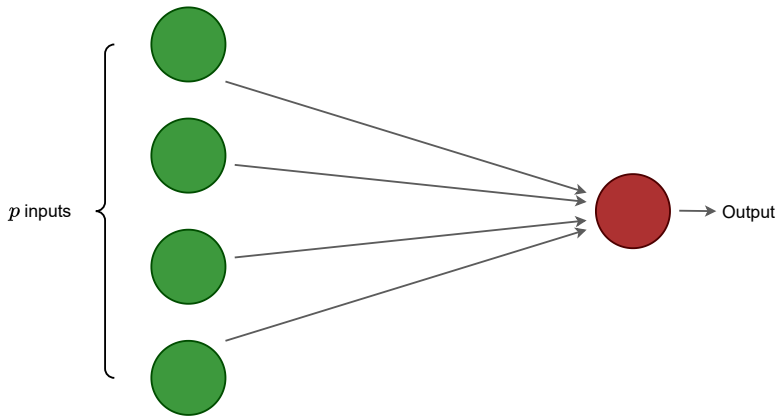

Figure 37: A linear regression model, or ARIMA$(p, 0, 0)$ model.

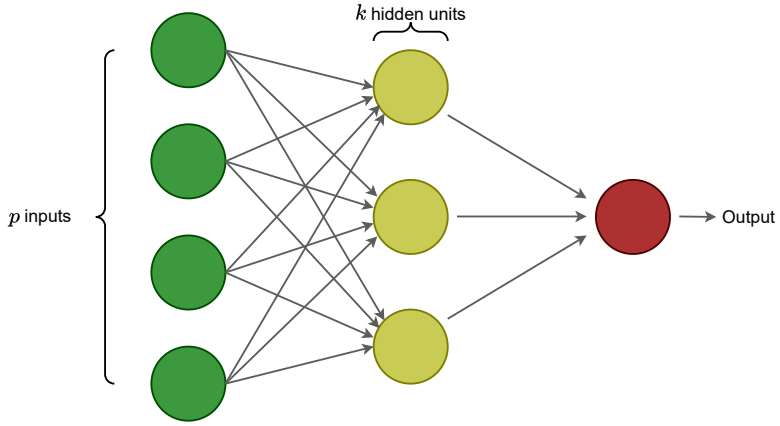

Figure 38: A neural network with p inputs and one hidden layer with k hidden neurons.

4.3.1 Definitions and Theory

4.3.2 How to select the best model

4.3.3 Forecasting

4.3.4 Implementation in R

```
#' dat_ts A time-series of the ovservations x_n
#' countrydat A data.frame with the required info to plot case numbers
#' modeldat A data.frame with model data to plot (has forecastlen more rows than countrydat)
#' cols A list of colours for plotting
#' labs A list of labels for plotting
#' forecastlen The forecast length
#' q The parameter for the basic model, in order tohave statistical models comparable with
    mathematical models
#' prevcases The cumulative cases prior to the first day in the range of countrydat

#requires arima auto.fit object from earlier
   nHidden <- max(1,floor(0.5*(1+auto.fit$arma[1]+auto.fit$arma[3])))
  #Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.
  nnfit   <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda =
      0, repeats = 20, maxit = 50)
  nn.fcst <- forecast(nnfit, h = forecastlen)

  nn.fcst$mean[nn.fcst$mean < 0] <- 0
  nn.fcst$fitted[1:q] <- countrydat$xn[1:q]

  modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
  modeldat$nnyn <- xntoyn(modeldat$nnxn) + prevcases

  labs$nn  <- paste0(nnfit$method, ", ||x*-x||=", modnorm(countrydat$xn,nn.fcst$fitted))
  labs$nny <- paste0(nnfit$method, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$nnyn[1:nrow(
      countrydat)]))

  plots[["nn"]] <- plot_nn(countrydat, modeldat, cols, labs)

```

```
27    plots[["nny"]] <- plot_nny(countrydat, modeldat, cols, labs)
```

Listing 7: Algorithm for Neural Network Model

and our plotting function for daily cases looks like the following

```
plot_nn <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
    geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
    geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_point(data = modeldat,aes(x = date, y = periodic, colour = "periodic")) +
    geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
    geom_point(data = modeldat, aes(x = date, y = nnxn, colour = "nn")) +
    geom_line(data = modeldat, aes(x = date, y = nnxn, colour = "nn")) +
    gg_scale_xy +
    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
           fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
    scale_fill_manual(values = c("actual" = cols$xn),
                      labels = c("actual" = labs$xn)) +
    scale_colour_manual(values = c("base" = cols$basexn, "nn" = cols$nn, "periodic" = cols$
        periodic),
                        labels = c("base" = labs$basexn, "nn" = labs$nn, "periodic" = labs$
                            periodic)) +
    xntheme()
  return(p)
}
```

Listing 8: Plot Neural Network Model

4.3.5 Plots

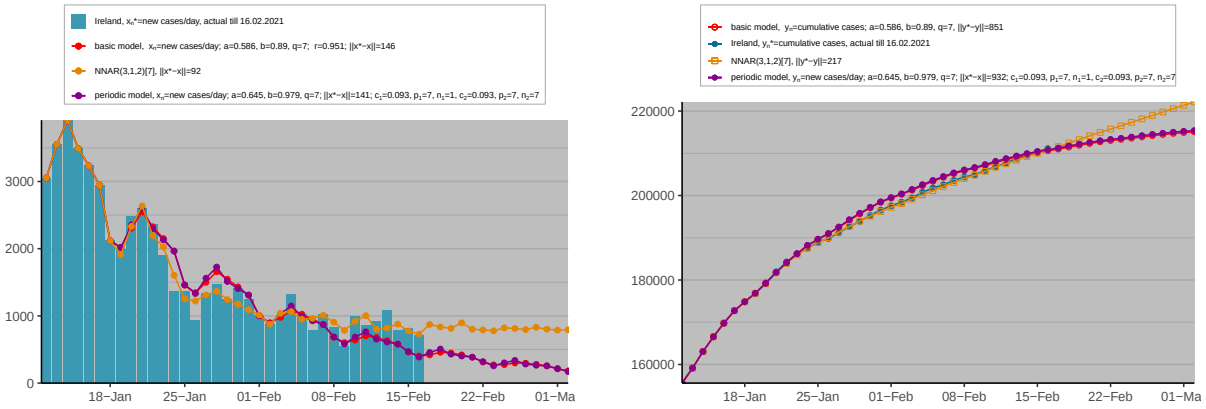

Figure 39: Neural Network model, Ireland

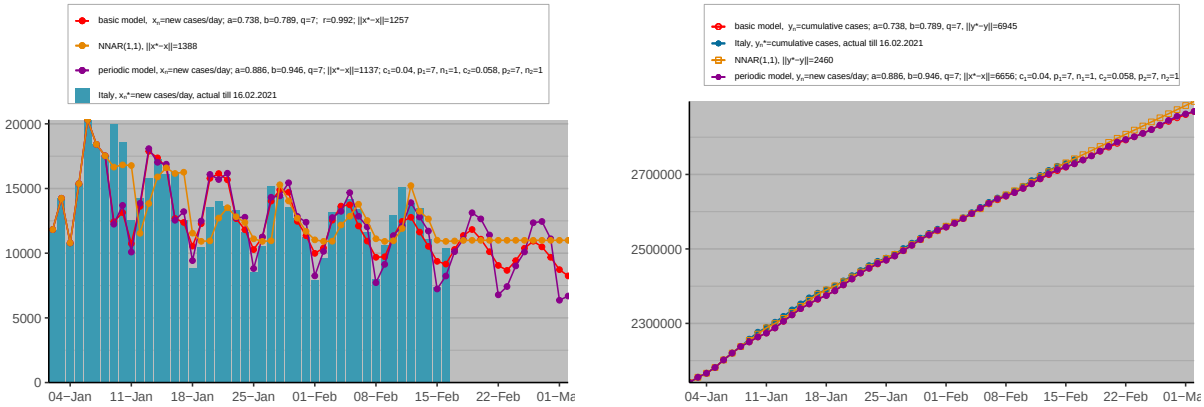

Figure 40: Neural Network model, Italy

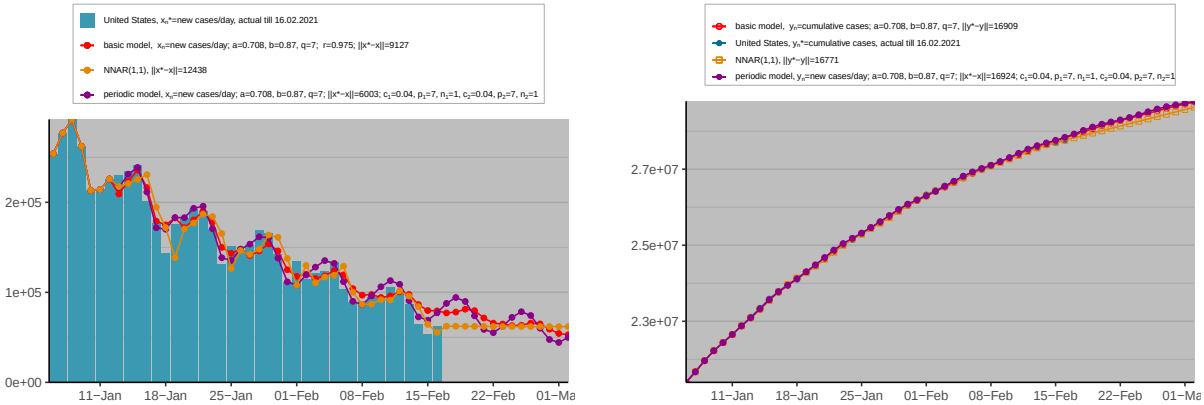

Figure 41: Neural Network model, United States

5 R Code and Data Sources

Much of the code was written from scratch for this project, or is a close to direct translation of the formulas described in papers such as Grigorian's [16].

5.1 R packages

`ggplot2` [24] is widely used for easily plotting and visualising the models. `rgdal` [6] allows geospatial `.shp` files to be read into `R`. `raster` [17] allows this data to be manipulated and plotted. `dplyr` [25] provides useful data manipulation functions, both for models and geospatial mapping. Statistical models (HoltWinters, ARIMA and Neural Network Regression) were readily implemented from `forecast` [20].

5.2 Plotting and colour

wesanderson [22]

Figure 42: Wes Anderson Palettes

5.3 Shapefiles

This data includes the geospatial vector data which can be used to *draw* country (and county) coastlines and borders.

World country shape data was obtained from [18], while the more detailed county-level shapefile was downloaded from [8].

5.4 Datasets

Country-based data:

Originally used data from [10], but the ECDC switched from a daily to a weekly update from 14 December 2020. Therefore, I have chosen to use the data from [9], which has remained daily

Ireland cases by county Downloaded from [7].

A	B	C	D	E	F	G	H	I	J	K
iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million
IRL	Europe	Ireland	2021-01-16	169780	3232	4150.429	2595	59	37	34383.762
IRL	Europe	Ireland	2021-01-17	172726	2946	3587.571	2608	13	37.714	34980.384
IRL	Europe	Ireland	2021-01-18	174843	2117	3186.286	2616	8	37.714	35409.118
IRL	Europe	Ireland	2021-01-19	176839	1996	3035.429	2708	92	44.429	35813.347
IRL	Europe	Ireland	2021-01-20	179324	2485	2882.857	2768	60	44	36316.608
IRL	Europe	Ireland	2021-01-21	181922	2598	2695	2818	50	47.143	36842.753
IRL	Europe	Ireland	2021-01-22	184279	2357	2533	2870	52	47.714	37320.092
IRL	Europe	Ireland	2021-01-23	186184	1905	2343.429	2947	77	50.286	37705.891
IRL	Europe	Ireland	2021-01-24	187554	1370	2118.286	2970	23	51.714	37983.343
IRL	Europe	Ireland	2021-01-25	188923	1369	2011.429	2977	7	51.571	38260.592
IRL	Europe	Ireland	2021-01-26	189851	928	1858.857	3066	89	51.143	38448.53

Figure 43: OWID World data extract

OBJECTID	ORIGID	CountyName	PopulationCensus16	TimeStamp	IGEasting	IGNorthing	Lat	Long	UGI	ConfirmedCovidCases
8456	6	Dublin	1347359	2021/01/19 00:0	313762	235813	53.3605	-6.292	http://data.geohi	61224
8457	7	Galway	258058	2021/01/19 00:0	151045	235818	53.3705	-8.7362	http://data.geohi	6938
8458	8	Kerry	147707	2021/01/19 00:0	92975	102996	52.1689	-9.565	http://data.geohi	3827
8459	9	Kildare	222504	2021/01/19 00:0	281262	221513	53.238	-6.7837	http://data.geohi	7951
8460	10	Kilkenny	99232	2021/01/19 00:0	253094	148060	52.5816	-7.2175	http://data.geohi	3012
8461	11	Laois	84697	2021/01/19 00:0	244211	193996	52.9952	-7.3423	http://data.geohi	2417
8462	12	Leitrim	32044	2021/01/19 00:0	200446	319670	54.1261	-7.9939	http://data.geohi	586
8463	13	Limerick	194899	2021/01/19 00:0	149743	141780	52.5255	-8.7412	http://data.geohi	8785
8464	14	Longford	40873	2021/01/19 00:0	220162	275901	53.7325	-7.6952	http://data.geohi	1208
8465	15	Louth	128884	2021/01/19 00:0	299463	297349	53.9161	-6.487	http://data.geohi	6863
8466	16	Mayo	130507	2021/01/19 00:0	117679	297355	53.9191	-9.2537	http://data.geohi	4768

Figure 44: ArcGIS Ireland data extract

A Appendix

All the code, plots, and even this report, are available at [15]. I will include the 3 main code files in text below.

```
require(ggplot2)
require(forecast)
require(dplyr)
require(wesanderson)
require(gridExtra)

owiddat <- read.csv("Data/owid-covid-data.csv")
owiddat$date <- as.Date(owiddat$date, tryFormats = c("%Y-%m-%d"))

plotslist <- list()

source("Code/covid-plotutils.R")
source("Code/covid-modelutils.R")

covidPlots <- function(country, dateBounds, data){
  plots <- list()
  countryrows <- grep(country, data$location)
  countrydat <- data.frame(date = data$date[countryrows],
                           xn   = data$new_cases[countryrows],
                           yn   = data$total_cases[countryrows])
  countrydatfull <- countrydat[countrydat$date <= dateBounds[2],]

  prevcases <- countrydat$yn[countrydat$date == as.Date(dateBounds[1])-1]
  #Specific dates
  countrydat <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= dateBounds[2],]
  #countrydat$xn[countrydat$xn < 0] <- 0
  latest_date <- countrydat$date[nrow(countrydat)]

  cols <- list(
    xn       = wes_palettes$Zissou1[1],
    yn       = wes_palettes$Darjeeling2[2],
    basexn   = wes_palettes$Darjeeling1[1],
    baseyn   = wes_palettes$Darjeeling1[1],
    x3       = wes_palettes$FantasticFox1[2],
    Crn      = wes_palettes$FantasticFox1[2],
    arima    = wes_palettes$Darjeeling1[4],
    arimapi  = wes_palettes$Darjeeling1[3],
```

```
    hw       = wes_palettes$Moonrise1[2],
    hwpi     = wes_palettes$Moonrise1[1],
    periodic = "magenta4", #wes_palettes$IsleofDogs1[1], #
    nn       = wes_palettes$FantasticFox1[4]
  )

  labs <- list(
    xn = list(bquote(.(country)*","~x[n]*"*=new cases/day, actual till"~.(format.Date(latest_date
         ,"%d.%m.%Y")))),
    yn = list(bquote(.(country)*","~y[n]*"*=cumulative cases, actual till"~.(format.Date(latest_
         date,"%d.%m.%Y"))))
  )

  plots[["xn"]] <- plot_xn(countrydatfull, cols, labs)

  plots[["yn"]] <- plot_yn(countrydatfull, cols, labs)

  #Basic model: need a,b,q,r using ||x-x*|| and ||y-y*||
  ##q - Any infected person becomes ill and infectious on the q-th day after infection.
  ##a - During each day, each ill person at large infects on average a other persons.
  ##b - During each day, a fraction b of ill people at large gets isolated

  forecastlen <- 14

  aseq    <- seq(from = 0.1, to = 2.5, length.out = 80)
  bseq    <- seq(from = 0.1, to = 0.9, length.out = 80)
  qseq    <- 6:8
  normdat <- expand.grid(q = qseq, a = aseq, b = bseq)
  abnorm  <- apply(normdat, 1, function(x) norm(x, countrydat$xn))

  normdat$abnorm  <- abnorm

  newnormdat <- normdat %>%
    top_n(abnorm, n = -0.07*nrow(.))

  col_grad <- wes_palette("Zissou1", 20, type = "continuous")

  tileoptim <- normdat[which.min(normdat$abnorm),1:3]
  #tileoptim <- newnormdat[which.min(newnormdat$abnormy),1:3]
  optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
  plots[["combnorm"]] <- ggplot(newnormdat, aes(x = a, y = b, z = abnorm)) +
    geom_contour_filled() + labs(fill = "||x-x*||") +
    scale_fill_brewer(palette = "Spectral")

  q <- optimpars[1]
  a <- optimpars[2]
  b <- optimpars[3]

  basexn   <- basicmodx(countrydat$xn, optimpars, len = forecastlen)

  modeldat <- data.frame(date = c(countrydat$date,latest_date + 1:forecastlen),
                         basexn = basexn, baseyn = xntoyn(basexn) + prevcases)

  #Newtons method, r_1 = r_0 - f(r_0)/f'(r_0)
  base_r_zero <- (a/b)^(1/(2*q))
  base_r_one  <- base_r_zero -basef(base_r_zero,optimpars)/basefprime(base_r_zero,optimpars)
  base_r_one  <- round(base_r_one,3)

  roptimpars  <- round(optimpars,3)
  labs$basexn <- list(bquote("basic model, "~x[n]*"=new cases/day; a="*.(roptimpars[2])*", b="*.(
      roptimpars[3])*", q="*.(roptimpars[1])*";  r="*.(base_r_one)*"; ||x*-x||="*.(norm(optimpars
      ,countrydat$xn))))

  ynnorm      <- modnorm(countrydat$yn, modeldat$baseyn[1:length(countrydat$yn)])
  labs$baseyn <- list(bquote("basic model, "~y[n]*"=cumulative cases; a="*.(roptimpars[2])*", b="
      *.(roptimpars[3])*", q="*.(roptimpars[1])*", ||y*-y||="*.(ynnorm)))

  plots[["basexn"]] <- plot_basexn(countrydat, modeldat, cols, labs)

  plots[["baseyn"]] <- plot_baseyn(countrydat, modeldat, cols, labs)

  optimC <- optim(par = countrydat$xn[1], normC, method = "Brent",
                  lower = 1, upper = 2*max(countrydat$xn[!is.na(countrydat$xn)]),
                  x = basexn[1:nrow(countrydat)], r = base_r_one)$par
```

```
107
108  modeldat$Crn <- optimC*base_r_one^(1:nrow(modeldat))
109
110  labs$Crn <- list(bquote(Cr^n*", r="*.(base_r_one)*", C="*.(floor(optimC))))
111
112  plots[["Crn"]] <- plot_crn(countrydat, modeldat, cols, labs)
113
114  mavgx1 <- movingavg(countrydat$xn[!is.na(countrydat$xn)])
115
116  countrydat$mavgx3 <- movingavg(mavgx1)
117
118  x3norm  <- modnorm(countrydat$xn,countrydat$mavgx3)
119  labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*.(x3norm)))
120
121  plots[["mavgx3"]] <- plot_mavgx3(countrydat, modeldat, cols, labs)
122
123  #parameters of the form (ci,pi,ni)
124  ##an = a(1 + c1 sin(2pi/p1 (n - n1)))
125  aseqper <- a*seq(from = 0.8, to = 1.2, length.out = 5)
126  bseqper <- b*seq(from = 0.8, to = 1.2, length.out = 5)
127  c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 10)
128  n_1seq  <- n_2seq <- c(1,7)
129  p_1seq  <- p_2seq <- 6:7
130  normdatp <- expand.grid(a  = aseqper, b  = bseqper,
131                          c1 = c_1seq,  c2 = c_2seq,
132                          p1 = p_1seq,  p2 = p_2seq,
133                          n1 = n_1seq,  n2 = n_2seq)
134
135  pernorm <- apply(normdatp, 1, function(x) normper(x, q = q, countrydat$xn))
136  normdatp$pernorm <- pernorm
137
138  peroptim   <- normdatp[which.min(pernorm),1:8]
139  optpernorm <- normdatp[which.min(pernorm),9]
140
141  modeldat$periodic  <- modxper(as.numeric(peroptim),countrydat$xn, q = q, forecastlen)
142  modeldat$periodicy <- xntoyn(modeldat$periodic) + prevcases
143
144  optpernormy <- modnorm(countrydat$yn, modeldat$periodicy[1:nrow(countrydat)])
145
146  peroptim <- round(as.numeric(peroptim),3)
147
148  perparamdat <- data.frame(
149    x  = modeldat$date,
150    an = peroptim[1]*(1+peroptim[3]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[7])/peroptim[5])),
151    bn = peroptim[2]*(1+peroptim[4]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[8])/peroptim[6]))
152  )
153
154  plots[["perparam"]] <- ggplot(perparamdat) +
155    geom_line(aes(x=x,y=an,col="a_n")) +
156    geom_line(aes(x=x,y=bn,col="b_n")) +
157    geom_hline(aes(yintercept = peroptim[1], col = "a"), linetype="dashed") +
158    geom_hline(aes(yintercept = peroptim[2], col = "b"), linetype="dashed") +
159    xlab("date") + ylab("") +
160    scale_color_manual(values = wes_palettes$Rushmore1[c(3,3,5,5)]) +
161    guides(colour = guide_legend(override.aes = list(linetype =
162         c("a"="dashed", "a_n"="solid", "b"="dashed", "b_n"="solid")))) +
163    xntheme() + theme(legend.position = "right")
164
165  #a,b,c1,c2,p1,p2,n1,n2
166  labs$periodic <- list(bquote("periodic model,"~x[n]*"=new cases/day;"
167                               ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
168                               ~q*"="*.(q)*";"~"||x*-x||="*.(optpernorm)*";"
169                               ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
170                               ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
171                               ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))
172
173  labs$periodicy <- list(bquote("periodic model,"~y[n]*"=new cases/day;"
174                                ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
175                                ~q*"="*.(q)*";"~"||x*-x||="*.(optpernormy)*";"
176                                ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
177                                ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
178                                ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))
179
180  plots[["periodic"]] <- plot_periodic(countrydat, modeldat, cols, labs)
```

```
181
182   plots[["periodicy"]] <- plot_periodicy(countrydat, modeldat, cols, labs)
183
184   #Statistical methods using timeseries forecasting
185
186   dat_ts <- ts(data = countrydat$xn, frequency = q)
187
188   g1 <- autoplot(dat_ts) + theme(axis.title = element_blank())
189   g2 <- ggAcf(dat_ts)    + ggtitle("")
190   g3 <- ggPacf(dat_ts)   + ggtitle("")
191
192   plots[["tsdisplay"]] <- grid.arrange(grobs = list(g1,g2,g3),
193     layout_matrix = rbind(c(1, 1), c(2, 3)))
194
195   plots[["residuals"]] <- gghistogram(dat_ts, add.normal = TRUE)
196
197   plots[["tsdecompose"]] <- autoplot(decompose(dat_ts))
198
199   if(any(countrydat$xn <= 0)){
200     plots[["hw"]] <- ggplot(data.frame(x = 0,y = 0)) +
201       geom_label(x = 0, y = 0, label = "Error: Country data has nonpositive values",
202                  color = "red", size = 5 , fontface = "bold" ) +
203       xlim(-1,1) + ylim(-1,1) +
204       theme(axis.line  = element_blank(),
205             axis.text  = element_blank(),
206             axis.ticks = element_blank(),
207             panel.grid = element_blank())
208   } else{
209     hwmethod   <- "additive"
210     #lambda=0 ensures values stay positive
211     hwfcst     <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
212
213     hwfcst$fitted[1:q] <- countrydat$xn[1:q]
214     modeldat$hwxn <- c(hwfcst$fitted, hwfcst$mean)
215     modeldat$hwlo <- c(hwfcst$fitted, hwfcst$lower[,2])
216     modeldat$hwhi <- c(hwfcst$fitted, hwfcst$upper[,2])
217
218     modeldat$hwyn  <- xntoyn(modeldat$hwxn)+prevcases
219     modeldat$hwylo <- xntoyn(modeldat$hwlo)+prevcases
220     modeldat$hwyhi <- xntoyn(modeldat$hwhi)+prevcases
221
222     hwnorm <- modnorm(countrydat$xn,hwfcst$fitted)
223
224     labs$hw   <- paste0("HoltWinters algorithm,  ||x*-x||=", modnorm(countrydat$xn,hwfcst$fitted)
          )
225     labs$hwy  <- paste0("HoltWinters algorithm,  ||y*-y||=", modnorm(countrydat$yn,modeldat$hwyn[
          1:nrow(countrydat)]))
226     labs$hwpi <- "HW 95% Prediction Interval"
227
228     plots[["hw"]]  <- plot_hw(countrydat, modeldat, cols, labs)
229     plots[["hwy"]] <- plot_hwy(countrydat, modeldat, cols, labs)
230   }
231
232   auto.fit <- auto.arima(dat_ts, lambda = 0) #keep values positive
233
234   getArmaModel <- function(arma, pdq = c(1,6,2), PDQ = c(3,7,4), s=5){
235     return(paste0("ARIMA(", paste0(arma[pdq], collapse = ","), ")(",
236                   paste0(arma[PDQ], collapse = ","), ")[", arma[s], "]"))
237   }
238
239   arima.fcst <- forecast(auto.fit, level = c(80, 95), h = forecastlen)
240   arima.fcst$fitted[1:q] <- countrydat$xn[1:q]
241
242   arimanorm <- modnorm(countrydat$xn,arima.fcst$fitted)
243
244   arimalabs    <- getArmaModel(auto.fit$arma)
245   labs$arima   <- paste0(arimalabs, ", ||x*-x||=", arimanorm)
246   labs$arimapi <- "ARIMA 95% Prediction Interval"
247
248   modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
249   modeldat$arimalo <- c(auto.fit$fitted,arima.fcst$lower[,2])
250   modeldat$arimahi <- c(auto.fit$fitted,arima.fcst$upper[,2])
251
252   modeldat$arimaxn[1:q] <- countrydat$xn[1:q]
```

```
  modeldat$arimalo[1:q] <- countrydat$xn[1:q]
  modeldat$arimahi[1:q] <- countrydat$xn[1:q]

  modeldat$arimayn  <- xntoyn(modeldat$arimaxn) + prevcases
  modeldat$arimaylo <- xntoyn(modeldat$arimalo) + prevcases
  modeldat$arimayhi <- xntoyn(modeldat$arimahi) + prevcases

  labs$arimay  <- paste0(arimalabs, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$arimayn[1:nrow(
      countrydat)]))

  plots[["arima"]] <- plot_arima(countrydat, modeldat, cols, labs)

  plots[["arimay"]] <- plot_arimay(countrydat, modeldat, cols, labs)

  plots[["hwarima"]] <- plot_hwarima(countrydat, modeldat, cols, labs)

  nHidden <- max(1,floor(0.5*(1+auto.fit$arma[1]+auto.fit$arma[3])))
  #Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.
  nnfit   <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda =
      0, repeats = 20, maxit = 50)
  nn.fcst <- forecast(nnfit, h = forecastlen)

  nn.fcst$mean[nn.fcst$mean < 0] <- 0
  nn.fcst$fitted[1:q] <- countrydat$xn[1:q]

  modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
  modeldat$nnyn <- xntoyn(modeldat$nnxn) + prevcases

  labs$nn  <- paste0(nnfit$method, ", ||x*-x||=", modnorm(countrydat$xn,nn.fcst$fitted))
  labs$nny <- paste0(nnfit$method, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$nnyn[1:nrow(
      countrydat)]))

  plots[["nn"]] <- plot_nn(countrydat, modeldat, cols, labs)

  plots[["nny"]] <- plot_nny(countrydat, modeldat, cols, labs)

  return(plots)
}

grigorDates <- c("2020-04-26", "2020-06-09")
datebounds <- list(
  "Italy"          = c("2021-01-02", "2021-02-16"),
  "United States" = c("2021-01-06", "2021-02-16"),
  "Ireland"        = c("2021-01-12", "2021-02-16"),
  "Germany"        = c("2021-01-06", "2021-02-16")
  #"Netherlands"    = c("2021-01-06", "2021-02-16"),
  #"Spain"          = c("2021-01-06", "2021-02-16"),
  #"UK"             = c("2021-01-06", "2021-02-16")
)

owiddat     <- owiddat[!is.na(owiddat$new_cases),]
totaldat    <- owiddat[owiddat$location == "World",]
latest_date <- totaldat$date[nrow(totaldat)]
wt_title <- sprintf('Global Total =%s as at %s',
                    format(sum(totaldat$new_cases), big.mark=",", scientific=FALSE),
                    format.Date(latest_date, "%B %d, %Y"))

plotslist[["WorldTotal"]][["xn"]] <- plot_worldtotal(totaldat)

for(country in names(datebounds)){
  plotslist[[country]] <- covidPlots(country, datebounds[[country]], owiddat)
}
```

Listing 9: Model helper functions

```
require(ggplot2)
require(forecast)
require(dplyr)
require(wesanderson)
require(gridExtra)

owiddat <- read.csv("Data/owid-covid-data.csv")
owiddat$date <- as.Date(owiddat$date, tryFormats = c("%Y-%m-%d"))

```

```
plotslist <- list()

source("Code/covid-plotutils.R")
source("Code/covid-modelutils.R")

covidPlots <- function(country, dateBounds, data){
  plots <- list()
  countryrows <- grep(country, data$location)
  countrydat <- data.frame(date = data$date[countryrows],
                           xn   = data$new_cases[countryrows],
                           yn   = data$total_cases[countryrows])
  countrydatfull <- countrydat[countrydat$date <= dateBounds[2],]

  prevcases <- countrydat$yn[countrydat$date == as.Date(dateBounds[1])-1]
  #Specific dates
  countrydat <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= dateBounds[2],]
  #countrydat$xn[countrydat$xn < 0] <- 0
  latest_date <- countrydat$date[nrow(countrydat)]

  cols <- list(
    xn       = wes_palettes$Zissou1[1],
    yn       = wes_palettes$Darjeeling2[2],
    basexn   = wes_palettes$Darjeeling1[1],
    baseyn   = wes_palettes$Darjeeling1[1],
    x3       = wes_palettes$FantasticFox1[2],
    Crn      = wes_palettes$FantasticFox1[2],
    arima    = wes_palettes$Darjeeling1[4],
    arimapi  = wes_palettes$Darjeeling1[3],
    hw       = wes_palettes$Moonrise1[2],
    hwpi     = wes_palettes$Moonrise1[1],
    periodic = "magenta4", #wes_palettes$IsleofDogs1[1], #
    nn       = wes_palettes$FantasticFox1[4]
  )

  labs <- list(
    xn = list(bquote(.(country)*","~x[n]*"*=new cases/day, actual till"~.(format.Date(latest_date,"%d.%m.%Y")))),
    yn = list(bquote(.(country)*","~y[n]*"*=cumulative cases, actual till"~.(format.Date(latest_date,"%d.%m.%Y"))))
  )

  plots[["xn"]] <- plot_xn(countrydatfull, cols, labs)

  plots[["yn"]] <- plot_yn(countrydatfull, cols, labs)

  #Basic model: need a,b,q,r using ||x-x*|| and ||y-y*||
  ##q - Any infected person becomes ill and infectious on the q-th day after infection.
  ##a - During each day, each ill person at large infects on average a other persons.
  ##b - During each day, a fraction b of ill people at large gets isolated

  forecastlen <- 14

  aseq    <- seq(from = 0.1, to = 2.5, length.out = 80)
  bseq    <- seq(from = 0.1, to = 0.9, length.out = 80)
  qseq    <- 6:8
  normdat <- expand.grid(q = qseq, a = aseq, b = bseq)
  abnorm  <- apply(normdat, 1, function(x) norm(x, countrydat$xn))

  normdat$abnorm  <- abnorm

  newnormdat <- normdat %>%
    top_n(abnorm, n = -0.07*nrow(.))

  col_grad <- wes_palette("Zissou1", 20, type = "continuous")

  tileoptim <- normdat[which.min(normdat$abnorm),1:3]
  #tileoptim <- newnormdat[which.min(newnormdat$abnormy),1:3]
  optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
  plots[["combnorm"]] <- ggplot(newnormdat, aes(x = a, y = b, z = abnorm)) +
    geom_contour_filled() + labs(fill = "||x-x*||") +
    scale_fill_brewer(palette = "Spectral")

  q <- optimpars[1]
  a <- optimpars[2]
```

```
b <- optimpars[3]

basexn   <- basicmodx(countrydat$xn, optimpars, len = forecastlen)

modeldat <- data.frame(date = c(countrydat$date,latest_date + 1:forecastlen),
                       basexn = basexn, baseyn = xntoyn(basexn) + prevcases)

#Newtons method, r_1 = r_0 - f(r_0)/f'(r_0)
base_r_zero <- (a/b)^(1/(2*q))
base_r_one  <- base_r_zero -basef(base_r_zero,optimpars)/basefprime(base_r_zero,optimpars)
base_r_one  <- round(base_r_one,3)

roptimpars  <- round(optimpars,3)
labs$basexn <- list(bquote("basic model, "~x[n]*"=new cases/day; a="*.(roptimpars[2])*", b="*.(
    roptimpars[3])*", q="*.(roptimpars[1])*";  r="*.(base_r_one)*"; ||x*-x||="*.(norm(optimpars
    ,countrydat$xn))))

ynnorm      <- modnorm(countrydat$yn, modeldat$baseyn[1:length(countrydat$yn)])
labs$baseyn <- list(bquote("basic model, "~y[n]*"=cumulative cases; a="*.(roptimpars[2])*", b="
    *.(roptimpars[3])*", q="*.(roptimpars[1])*", ||y*-y||="*.(ynnorm)))

plots[["basexn"]] <- plot_basexn(countrydat, modeldat, cols, labs)

plots[["baseyn"]] <- plot_baseyn(countrydat, modeldat, cols, labs)

optimC <- optim(par = countrydat$xn[1], normC, method = "Brent",
                lower = 1, upper = 2*max(countrydat$xn[!is.na(countrydat$xn)]),
                x = basexn[1:nrow(countrydat)], r = base_r_one)$par

modeldat$Crn <- optimC*base_r_one^(1:nrow(modeldat))

labs$Crn <- list(bquote(Cr^n*", r="*.(base_r_one)*", C="*.(floor(optimC))))

plots[["Crn"]] <- plot_crn(countrydat, modeldat, cols, labs)

mavgx1 <- movingavg(countrydat$xn[!is.na(countrydat$xn)])

countrydat$mavgx3 <- movingavg(mavgx1)

x3norm  <- modnorm(countrydat$xn,countrydat$mavgx3)
labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*.(x3norm)))

plots[["mavgx3"]] <- plot_mavgx3(countrydat, modeldat, cols, labs)

#parameters of the form (ci,pi,ni)
##an = a(1 + c1 sin(2pi/p1 (n - n1)))
aseqper <- a*seq(from = 0.8, to = 1.2, length.out = 5)
bseqper <- b*seq(from = 0.8, to = 1.2, length.out = 5)
c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 10)
n_1seq  <- n_2seq <- c(1,7)
p_1seq  <- p_2seq <- 6:7
normdatp <- expand.grid(a  = aseqper, b  = bseqper,
                        c1 = c_1seq,  c2 = c_2seq,
                        p1 = p_1seq,  p2 = p_2seq,
                        n1 = n_1seq,  n2 = n_2seq)

pernorm <- apply(normdatp, 1, function(x) normper(x, q = q, countrydat$xn))
normdatp$pernorm <- pernorm

peroptim   <- normdatp[which.min(pernorm),1:8]
optpernorm <- normdatp[which.min(pernorm),9]

modeldat$periodic  <- modxper(as.numeric(peroptim),countrydat$xn, q = q, forecastlen)
modeldat$periodicy <- xntoyn(modeldat$periodic) + prevcases

optpernormy <- modnorm(countrydat$yn, modeldat$periodicy[1:nrow(countrydat)])

peroptim <- round(as.numeric(peroptim),3)

perparamdat <- data.frame(
  x  = modeldat$date,
  an = peroptim[1]*(1+peroptim[3]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[7])/peroptim[5])),
  bn = peroptim[2]*(1+peroptim[4]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[8])/peroptim[6]))
)
```

```
153
154   plots[["perparam"]] <- ggplot(perparamdat) +
155     geom_line(aes(x=x,y=an,col="a_n")) +
156     geom_line(aes(x=x,y=bn,col="b_n")) +
157     geom_hline(aes(yintercept = peroptim[1], col = "a"), linetype="dashed") +
158     geom_hline(aes(yintercept = peroptim[2], col = "b"), linetype="dashed") +
159     xlab("date") + ylab("") +
160     scale_color_manual(values = wes_palettes$Rushmore1[c(3,3,5,5)]) +
161     guides(colour = guide_legend(override.aes = list(linetype =
162          c("a"="dashed", "a_n"="solid", "b"="dashed", "b_n"="solid")))) +
163     xntheme() + theme(legend.position = "right")
164
165   #a,b,c1,c2,p1,p2,n1,n2
166   labs$periodic <- list(bquote("periodic model,"~x[n]*"=new cases/day;"
167                                ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
168                                ~q*"="*.(q)*";""~"||x*-x||="*.(optpernorm)*";"
169                                ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
170                                ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
171                                ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))
172
173   labs$periodicy <- list(bquote("periodic model,"~y[n]*"=new cases/day;"
174                                ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
175                                ~q*"="*.(q)*";""~"||x*-x||="*.(optpernormy)*";"
176                                ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
177                                ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
178                                ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))
179
180   plots[["periodic"]] <- plot_periodic(countrydat, modeldat, cols, labs)
181
182   plots[["periodicy"]] <- plot_periodicy(countrydat, modeldat, cols, labs)
183
184   #Statistical methods using timeseries forecasting
185
186   dat_ts <- ts(data = countrydat$xn, frequency = q)
187
188   g1 <- autoplot(dat_ts) + theme(axis.title = element_blank())
189   g2 <- ggAcf(dat_ts)    + ggtitle("")
190   g3 <- ggPacf(dat_ts)   + ggtitle("")
191
192   plots[["tsdisplay"]] <- grid.arrange(grobs = list(g1,g2,g3),
193     layout_matrix = rbind(c(1, 1), c(2, 3)))
194
195   plots[["residuals"]] <- gghistogram(dat_ts, add.normal = TRUE)
196
197   plots[["tsdecompose"]] <- autoplot(decompose(dat_ts))
198
199   if(any(countrydat$xn <= 0)){
200     plots[["hw"]] <- ggplot(data.frame(x = 0,y = 0)) +
201       geom_label(x = 0, y = 0, label = "Error: Country data has nonpositive values",
202                  color = "red", size = 5 , fontface = "bold" ) +
203       xlim(-1,1) + ylim(-1,1) +
204       theme(axis.line  = element_blank(),
205             axis.text  = element_blank(),
206             axis.ticks = element_blank(),
207             panel.grid = element_blank())
208   } else{
209     hwmethod   <- "additive"
210     #lambda=0 ensures values stay positive
211     hwfcst     <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
212
213     hwfcst$fitted[1:q] <- countrydat$xn[1:q]
214     modeldat$hwxn <- c(hwfcst$fitted, hwfcst$mean)
215     modeldat$hwlo <- c(hwfcst$fitted, hwfcst$lower[,2])
216     modeldat$hwhi <- c(hwfcst$fitted, hwfcst$upper[,2])
217
218     modeldat$hwyn  <- xntoyn(modeldat$hwxn)+prevcases
219     modeldat$hwylo <- xntoyn(modeldat$hwlo)+prevcases
220     modeldat$hwyhi <- xntoyn(modeldat$hwhi)+prevcases
221
222     hwnorm <- modnorm(countrydat$xn,hwfcst$fitted)
223
224     labs$hw   <- paste0("HoltWinters algorithm,  ||x*-x||=", modnorm(countrydat$xn,hwfcst$fitted)
          )
225     labs$hwy  <- paste0("HoltWinters algorithm,  ||y*-y||=", modnorm(countrydat$yn,modeldat$hwyn[
```

```
        1:nrow(countrydat)]))
    labs$hwpi <- "HW 95% Prediction Interval"

    plots[["hw"]]  <- plot_hw(countrydat, modeldat, cols, labs)
    plots[["hwy"]] <- plot_hwy(countrydat, modeldat, cols, labs)
  }

  auto.fit <- auto.arima(dat_ts, lambda = 0) #keep values positive

  getArmaModel <- function(arma, pdq = c(1,6,2), PDQ = c(3,7,4), s=5){
    return(paste0("ARIMA(", paste0(arma[pdq], collapse = ","), ")(",
                  paste0(arma[PDQ], collapse = ","), ")[", arma[s], "]"))
  }

  arima.fcst <- forecast(auto.fit, level = c(80, 95), h = forecastlen)
  arima.fcst$fitted[1:q] <- countrydat$xn[1:q]

  arimanorm <- modnorm(countrydat$xn,arima.fcst$fitted)

  arimalabs    <- getArmaModel(auto.fit$arma)
  labs$arima   <- paste0(arimalabs, ", ||x*-x||=", arimanorm)
  labs$arimapi <- "ARIMA 95% Prediction Interval"

  modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
  modeldat$arimalo <- c(auto.fit$fitted,arima.fcst$lower[,2])
  modeldat$arimahi <- c(auto.fit$fitted,arima.fcst$upper[,2])

  modeldat$arimaxn[1:q] <- countrydat$xn[1:q]
  modeldat$arimalo[1:q] <- countrydat$xn[1:q]
  modeldat$arimahi[1:q] <- countrydat$xn[1:q]

  modeldat$arimayn  <- xntoyn(modeldat$arimaxn) + prevcases
  modeldat$arimaylo <- xntoyn(modeldat$arimalo) + prevcases
  modeldat$arimayhi <- xntoyn(modeldat$arimahi) + prevcases

  labs$arimay  <- paste0(arimalabs, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$arimayn[1:nrow(
      countrydat)]))

  plots[["arima"]] <- plot_arima(countrydat, modeldat, cols, labs)

  plots[["arimay"]] <- plot_arimay(countrydat, modeldat, cols, labs)

  plots[["hwarima"]] <- plot_hwarima(countrydat, modeldat, cols, labs)

  nHidden <- max(1,floor(0.5*(1+auto.fit$arma[1]+auto.fit$arma[3])))
  #Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.
  nnfit   <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda =
      0, repeats = 20, maxit = 50)
  nn.fcst <- forecast(nnfit, h = forecastlen)

  nn.fcst$mean[nn.fcst$mean < 0] <- 0
  nn.fcst$fitted[1:q] <- countrydat$xn[1:q]

  modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
  modeldat$nnyn <- xntoyn(modeldat$nnxn) + prevcases

  labs$nn  <- paste0(nnfit$method, ", ||x*-x||=", modnorm(countrydat$xn,nn.fcst$fitted))
  labs$nny <- paste0(nnfit$method, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$nnyn[1:nrow(
      countrydat)]))

  plots[["nn"]] <- plot_nn(countrydat, modeldat, cols, labs)

  plots[["nny"]] <- plot_nny(countrydat, modeldat, cols, labs)

  return(plots)
}

grigorDates <- c("2020-04-26", "2020-06-09")
datebounds <- list(
  "Italy"         = c("2021-01-02", "2021-02-16"),
  "United States" = c("2021-01-06", "2021-02-16"),
  "Ireland"       = c("2021-01-12", "2021-02-16"),
  "Germany"       = c("2021-01-06", "2021-02-16")
  #"Netherlands"   = c("2021-01-06", "2021-02-16"),
```

```
  #"Spain"          = c("2021-01-06", "2021-02-16"),
  #"UK"             = c("2021-01-06", "2021-02-16")
)

owiddat     <- owiddat[!is.na(owiddat$new_cases),]
totaldat    <- owiddat[owiddat$location == "World",]
latest_date <- totaldat$date[nrow(totaldat)]
wt_title <- sprintf('Global Total =%s as at %s',
                    format(sum(totaldat$new_cases), big.mark=",", scientific=FALSE),
                    format.Date(latest_date, "%B %d, %Y"))

plotslist[["WorldTotal"]][["xn"]] <- plot_worldtotal(totaldat)

for(country in names(datebounds)){
  plotslist[[country]] <- covidPlots(country, datebounds[[country]], owiddat)
}
```

Listing 10: Plotting functions

```
require(ggplot2)
require(forecast)
require(dplyr)
require(wesanderson)
require(gridExtra)

owiddat <- read.csv("Data/owid-covid-data.csv")
owiddat$date <- as.Date(owiddat$date, tryFormats = c("%Y-%m-%d"))

plotslist <- list()

source("Code/covid-plotutils.R")
source("Code/covid-modelutils.R")

covidPlots <- function(country, dateBounds, data){
  plots <- list()
  countryrows <- grep(country, data$location)
  countrydat <- data.frame(date = data$date[countryrows],
                           xn   = data$new_cases[countryrows],
                           yn   = data$total_cases[countryrows])
  countrydatfull <- countrydat[countrydat$date <= dateBounds[2],]

  prevcases <- countrydat$yn[countrydat$date == as.Date(dateBounds[1])-1]
  #Specific dates
  countrydat <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= dateBounds[2],]
  #countrydat$xn[countrydat$xn < 0] <- 0
  latest_date <- countrydat$date[nrow(countrydat)]

  cols <- list(
    xn       = wes_palettes$Zissou1[1],
    yn       = wes_palettes$Darjeeling2[2],
    basexn   = wes_palettes$Darjeeling1[1],
    baseyn   = wes_palettes$Darjeeling1[1],
    x3       = wes_palettes$FantasticFox1[2],
    Crn      = wes_palettes$FantasticFox1[2],
    arima    = wes_palettes$Darjeeling1[4],
    arimapi  = wes_palettes$Darjeeling1[3],
    hw       = wes_palettes$Moonrise1[2],
    hwpi     = wes_palettes$Moonrise1[1],
    periodic = "magenta4", #wes_palettes$IsleofDogs1[1], #
    nn       = wes_palettes$FantasticFox1[4]
  )

  labs <- list(
    xn = list(bquote(.(country)*","~x[n]*"*=new cases/day, actual till"~.(format.Date(latest_date
         ,"%d.%m.%Y")))),
    yn = list(bquote(.(country)*","~y[n]*"*=cumulative cases, actual till"~.(format.Date(latest_
        date,"%d.%m.%Y"))))
  )

  plots[["xn"]] <- plot_xn(countrydatfull, cols, labs)

  plots[["yn"]] <- plot_yn(countrydatfull, cols, labs)

  #Basic model: need a,b,q,r using ||x-x*|| and ||y-y*||
```

```
##q - Any infected person becomes ill and infectious on the q-th day after infection.
##a - During each day, each ill person at large infects on average a other persons.
##b - During each day, a fraction b of ill people at large gets isolated

forecastlen <- 14

aseq    <- seq(from = 0.1, to = 2.5, length.out = 80)
bseq    <- seq(from = 0.1, to = 0.9, length.out = 80)
qseq    <- 6:8
normdat <- expand.grid(q = qseq, a = aseq, b = bseq)
abnorm  <- apply(normdat, 1, function(x) norm(x, countrydat$xn))

normdat$abnorm  <- abnorm

newnormdat <- normdat %>%
  top_n(abnorm, n = -0.07*nrow(.))

col_grad <- wes_palette("Zissou1", 20, type = "continuous")

tileoptim <- normdat[which.min(normdat$abnorm),1:3]
#tileoptim <- newnormdat[which.min(newnormdat$abnormy),1:3]
optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
plots[["combnorm"]] <- ggplot(newnormdat, aes(x = a, y = b, z = abnorm)) +
  geom_contour_filled() + labs(fill = "||x-x*||") +
  scale_fill_brewer(palette = "Spectral")

q <- optimpars[1]
a <- optimpars[2]
b <- optimpars[3]

basexn   <- basicmodx(countrydat$xn, optimpars, len = forecastlen)

modeldat <- data.frame(date = c(countrydat$date,latest_date + 1:forecastlen),
                       basexn = basexn, baseyn = xntoyn(basexn) + prevcases)

#Newtons method, r_1 = r_0 - f(r_0)/f'(r_0)
base_r_zero <- (a/b)^(1/(2*q))
base_r_one  <- base_r_zero -basef(base_r_zero,optimpars)/basefprime(base_r_zero,optimpars)
base_r_one  <- round(base_r_one,3)

roptimpars  <- round(optimpars,3)
labs$basexn <- list(bquote("basic model, "~x[n]*"=new cases/day; a="*.(roptimpars[2])*", b="*.(
    roptimpars[3])*", q="*.(roptimpars[1])*";  r="*.(base_r_one)*"; ||x*-x||="*.(norm(optimpars
    ,countrydat$xn))))

ynnorm      <- modnorm(countrydat$yn, modeldat$baseyn[1:length(countrydat$yn)])
labs$baseyn <- list(bquote("basic model, "~y[n]*"=cumulative cases; a="*.(roptimpars[2])*", b="
    *.(roptimpars[3])*", q="*.(roptimpars[1])*", ||y*-y||="*.(ynnorm)))

plots[["basexn"]] <- plot_basexn(countrydat, modeldat, cols, labs)

plots[["baseyn"]] <- plot_baseyn(countrydat, modeldat, cols, labs)

optimC <- optim(par = countrydat$xn[1], normC, method = "Brent",
                lower = 1, upper = 2*max(countrydat$xn[!is.na(countrydat$xn)]),
                x = basexn[1:nrow(countrydat)], r = base_r_one)$par

modeldat$Crn <- optimC*base_r_one^(1:nrow(modeldat))

labs$Crn <- list(bquote(Cr^n*", r="*.(base_r_one)*", C="*.(floor(optimC))))

plots[["Crn"]] <- plot_crn(countrydat, modeldat, cols, labs)

mavgx1 <- movingavg(countrydat$xn[!is.na(countrydat$xn)])

countrydat$mavgx3 <- movingavg(mavgx1)

x3norm  <- modnorm(countrydat$xn,countrydat$mavgx3)
labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*.(x3norm)))

plots[["mavgx3"]] <- plot_mavgx3(countrydat, modeldat, cols, labs)

#parameters of the form (ci,pi,ni)
##an = a(1 + c1 sin(2pi/p1 (n - n1)))
```

```
aseqper <- a*seq(from = 0.8, to = 1.2, length.out = 5)
bseqper <- b*seq(from = 0.8, to = 1.2, length.out = 5)
c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 10)
n_1seq  <- n_2seq <- c(1,7)
p_1seq  <- p_2seq <- 6:7
normdatp <- expand.grid(a  = aseqper, b  = bseqper,
                        c1 = c_1seq,  c2 = c_2seq,
                        p1 = p_1seq,  p2 = p_2seq,
                        n1 = n_1seq,  n2 = n_2seq)

pernorm <- apply(normdatp, 1, function(x) normper(x, q = q, countrydat$xn))
normdatp$pernorm <- pernorm

peroptim   <- normdatp[which.min(pernorm),1:8]
optpernorm <- normdatp[which.min(pernorm),9]

modeldat$periodic  <- modxper(as.numeric(peroptim),countrydat$xn, q = q, forecastlen)
modeldat$periodicy <- xntoyn(modeldat$periodic) + prevcases

optpernormy <- modnorm(countrydat$yn, modeldat$periodicy[1:nrow(countrydat)])

peroptim <- round(as.numeric(peroptim),3)

perparamdat <- data.frame(
  x  = modeldat$date,
  an = peroptim[1]*(1+peroptim[3]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[7])/peroptim[5])),
  bn = peroptim[2]*(1+peroptim[4]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[8])/peroptim[6]))
)

plots[["perparam"]] <- ggplot(perparamdat) +
  geom_line(aes(x=x,y=an,col="a_n")) +
  geom_line(aes(x=x,y=bn,col="b_n")) +
  geom_hline(aes(yintercept = peroptim[1], col = "a"), linetype="dashed") +
  geom_hline(aes(yintercept = peroptim[2], col = "b"), linetype="dashed") +
  xlab("date") + ylab("") +
  scale_color_manual(values = wes_palettes$Rushmore1[c(3,3,5,5)]) +
  guides(colour = guide_legend(override.aes = list(linetype =
       c("a"="dashed", "a_n"="solid", "b"="dashed", "b_n"="solid")))) +
  xntheme() + theme(legend.position = "right")

#a,b,c1,c2,p1,p2,n1,n2
labs$periodic <- list(bquote("periodic model,"~x[n]*"=new cases/day;"
                             ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
                             ~q*"="*.(q)*";"~"||x*-x||="*.(optpernorm)*";"
                             ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
                             ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
                             ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))

labs$periodicy <- list(bquote("periodic model,"~y[n]*"=new cases/day;"
                              ~a*"="*.(peroptim[1])*","~ b*"="*.(peroptim[2])*","
                              ~q*"="*.(q)*";"~"||x*-x||="*.(optpernormy)*";"
                              ~c[1]*"="*.(peroptim[3])*","~p[1]*"="*.(peroptim[5])*","
                              ~n[1]*"="*.(peroptim[7])*","~c[2]*"="*.(peroptim[4])*","
                              ~p[2]*"="*.(peroptim[6])*","~n[2]*"="*.(peroptim[8])))

plots[["periodic"]] <- plot_periodic(countrydat, modeldat, cols, labs)

plots[["periodicy"]] <- plot_periodicy(countrydat, modeldat, cols, labs)

#Statistical methods using timeseries forecasting

dat_ts <- ts(data = countrydat$xn, frequency = q)

g1 <- autoplot(dat_ts) + theme(axis.title = element_blank())
g2 <- ggAcf(dat_ts)    + ggtitle("")
g3 <- ggPacf(dat_ts)   + ggtitle("")

plots[["tsdisplay"]] <- grid.arrange(grobs = list(g1,g2,g3),
  layout_matrix = rbind(c(1, 1), c(2, 3)))

plots[["residuals"]] <- gghistogram(dat_ts, add.normal = TRUE)

plots[["tsdecompose"]] <- autoplot(decompose(dat_ts))

```

```
199   if(any(countrydat$xn <= 0)){
200     plots[["hw"]] <- ggplot(data.frame(x = 0,y = 0)) +
201       geom_label(x = 0, y = 0, label = "Error: Country data has nonpositive values",
202                  color = "red", size = 5 , fontface = "bold" ) +
203       xlim(-1,1) + ylim(-1,1) +
204       theme(axis.line  = element_blank(),
205             axis.text  = element_blank(),
206             axis.ticks = element_blank(),
207             panel.grid = element_blank())
208   } else{
209     hwmethod   <- "additive"
210     #lambda=0 ensures values stay positive
211     hwfcst     <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
212
213     hwfcst$fitted[1:q] <- countrydat$xn[1:q]
214     modeldat$hwxn <- c(hwfcst$fitted, hwfcst$mean)
215     modeldat$hwlo <- c(hwfcst$fitted, hwfcst$lower[,2])
216     modeldat$hwhi <- c(hwfcst$fitted, hwfcst$upper[,2])
217
218     modeldat$hwyn  <- xntoyn(modeldat$hwxn)+prevcases
219     modeldat$hwylo <- xntoyn(modeldat$hwlo)+prevcases
220     modeldat$hwyhi <- xntoyn(modeldat$hwhi)+prevcases
221
222     hwnorm <- modnorm(countrydat$xn,hwfcst$fitted)
223
224     labs$hw   <- paste0("HoltWinters algorithm,  ||x*-x||=", modnorm(countrydat$xn,hwfcst$fitted)
          )
225     labs$hwy  <- paste0("HoltWinters algorithm,  ||y*-y||=", modnorm(countrydat$yn,modeldat$hwyn[
          1:nrow(countrydat)]))
226     labs$hwpi <- "HW 95% Prediction Interval"
227
228     plots[["hw"]]  <- plot_hw(countrydat, modeldat, cols, labs)
229     plots[["hwy"]] <- plot_hwy(countrydat, modeldat, cols, labs)
230   }
231
232   auto.fit <- auto.arima(dat_ts, lambda = 0) #keep values positive
233
234   getArmaModel <- function(arma, pdq = c(1,6,2), PDQ = c(3,7,4), s=5){
235     return(paste0("ARIMA(", paste0(arma[pdq], collapse = ","), ")(",
236                   paste0(arma[PDQ], collapse = ","), ")[", arma[s], "]"))
237   }
238
239   arima.fcst <- forecast(auto.fit, level = c(80, 95), h = forecastlen)
240   arima.fcst$fitted[1:q] <- countrydat$xn[1:q]
241
242   arimanorm <- modnorm(countrydat$xn,arima.fcst$fitted)
243
244   arimalabs    <- getArmaModel(auto.fit$arma)
245   labs$arima   <- paste0(arimalabs, ", ||x*-x||=", arimanorm)
246   labs$arimapi <- "ARIMA 95% Prediction Interval"
247
248   modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
249   modeldat$arimalo <- c(auto.fit$fitted ,arima.fcst$lower[,2])
250   modeldat$arimahi <- c(auto.fit$fitted ,arima.fcst$upper[,2])
251
252   modeldat$arimaxn[1:q] <- countrydat$xn[1:q]
253   modeldat$arimalo[1:q] <- countrydat$xn[1:q]
254   modeldat$arimahi[1:q] <- countrydat$xn[1:q]
255
256   modeldat$arimayn  <- xntoyn(modeldat$arimaxn) + prevcases
257   modeldat$arimaylo <- xntoyn(modeldat$arimalo) + prevcases
258   modeldat$arimayhi <- xntoyn(modeldat$arimahi) + prevcases
259
260   labs$arimay  <- paste0(arimalabs, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$arimayn[1:nrow(
          countrydat)]))
261
262   plots[["arima"]] <- plot_arima(countrydat, modeldat, cols, labs)
263
264   plots[["arimay"]] <- plot_arimay(countrydat, modeldat, cols, labs)
265
266   plots[["hwarima"]] <- plot_hwarima(countrydat, modeldat, cols, labs)
267
268   nHidden <- max(1,floor(0.5*(1+auto.fit$arma[1]+auto.fit$arma[3])))
269   #Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.
```

```
  nnfit   <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda =
      0, repeats = 20, maxit = 50)
  nn.fcst <- forecast(nnfit, h = forecastlen)

  nn.fcst$mean[nn.fcst$mean < 0] <- 0
  nn.fcst$fitted[1:q] <- countrydat$xn[1:q]

  modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
  modeldat$nnyn <- xntoyn(modeldat$nnxn) + prevcases

  labs$nn  <- paste0(nnfit$method, ", ||x*-x||=", modnorm(countrydat$xn,nn.fcst$fitted))
  labs$nny <- paste0(nnfit$method, ", ||y*-y||=", modnorm(countrydat$yn,modeldat$nnyn[1:nrow(
      countrydat)]))

  plots[["nn"]] <- plot_nn(countrydat, modeldat, cols, labs)

  plots[["nny"]] <- plot_nny(countrydat, modeldat, cols, labs)

  return(plots)
}

grigorDates <- c("2020-04-26", "2020-06-09")
datebounds <- list(
  "Italy"         = c("2021-01-02", "2021-02-16"),
  "United States" = c("2021-01-06", "2021-02-16"),
  "Ireland"       = c("2021-01-12", "2021-02-16"),
  "Germany"       = c("2021-01-06", "2021-02-16")
  #"Netherlands"   = c("2021-01-06", "2021-02-16"),
  #"Spain"         = c("2021-01-06", "2021-02-16"),
  #"UK"            = c("2021-01-06", "2021-02-16")
)

owiddat     <- owiddat[!is.na(owiddat$new_cases),]
totaldat    <- owiddat[owiddat$location == "World",]
latest_date <- totaldat$date[nrow(totaldat)]
wt_title <- sprintf('Global Total =%s as at %s',
                    format(sum(totaldat$new_cases), big.mark=",", scientific=FALSE),
                    format.Date(latest_date, "%B %d, %Y"))

plotslist[["WorldTotal"]][["xn"]] <- plot_worldtotal(totaldat)

for(country in names(datebounds)){
  plotslist[[country]] <- covidPlots(country, datebounds[[country]], owiddat)
}
```

Listing 11: Main algorithm

```
source("Code/covid-plotutils.R")
source("Code/covid-modelutils.R")

owiddat      <- read.csv("Data/owid-covid-data.csv")
owiddat$date <- as.Date(owiddat$date, tryFormats = c("%Y-%m-%d"))
multilist <- list()

multidates <- list(
  "Italy"         = list(c("2020-12-16", "2021-01-01"),
                         c("2021-01-02", "2021-01-30")),
  "United States" = list(c("2020-11-16", "2021-01-06"),
                         c("2021-01-07", "2021-01-30")),
  "Ireland"       = list(c("2020-12-16", "2021-01-07"),
                         c("2021-01-08", "2021-01-30"))
)

multiphasePlots <- function(country, dates, data){
  plots <- list()
  crows <- grep(country, data$location)
  countrydat <- data.frame(date = data$date[crows],
                           xn = data$new_cases[crows], yn = data$total_cases[crows])
  if(nrow(countrydat[countrydat$date < dates[[1]][1],]) == 0)
    beforecumcases <- 0
  else
    beforecumcases <- sum(countrydat$xn[countrydat$date < dates[[1]][1]])

  countrydat  <- countrydat[countrydat$date >= dates[[1]][1],]
```

```
  countrydat  <- countrydat[countrydat$date <= dates[[length(dates)]][2],]
  latest_date <- countrydat$date[nrow(countrydat)]

  forecastlen <- 5

  multimodx <- function(x, multix, pars, oldp = rep(1,10), start=FALSE, len = 0){
    q <- floor(pars[1])
    a <- pars[2]
    b <- pars[3]
    fitstd <- length(multix)+1

    if(start){
      multix[fitstd:(fitstd+q-1)] <- x[1:q]
      for(i in (fitstd+q):(fitstd+length(x)+len-1)){
        multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
      }
    } else {
      for(i in (fitstd):(fitstd+length(x)+len-1)){
        multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
      }
      #multix[fitstd] <- b/oldp[[3]]*((1-oldp[[3]])*multix[fitstd-1] + oldp[[2]]*multix[fitstd-q
          ])
      #for(i in (fitstd+1):(fitstd+q-1)){
      #  multix[i] <- (1-b)*multix[i-1] + b/oldp[[3]]*oldp[[2]]*multix[i-q]
      #}
      #for(i in (fitstd+q):(fitstd+length(x)+len-1)){
      #  multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
      #}
    }
    return(multix)
  }

  multimodxper <- function(par, q=7, x, multix, oldp = rep(1,10), start=FALSE, len = 0){
    #a,b,c1,c2,p1,p2,n1,n2
    #first day of this phase
    fitstd <- length(multix)+1
    an  <- par[1]*(1+par[3]*sin(2*pi*(1:(fitstd+length(x)+len-1) - par[7])/par[5]))
    bn  <- par[2]*(1+par[4]*sin(2*pi*(1:(fitstd+length(x)+len-1) - par[8])/par[6]))

    if(start){
      multix[fitstd:(fitstd+q-1)] <- x[1:q]
      for(i in (fitstd+q):(fitstd+length(x)+len-1)){
        multix[i] <- (bn[i]*(1-bn[i-1]))*multix[i-1]/bn[i-1] +
          (an[i-q]*bn[i])*multix[i-q]/bn[i-q]
      }
    } else {
      for(i in fitstd:(fitstd+length(x)+len-1)){
        multix[i] <- (bn[i]*(1-bn[i-1]))*multix[i-1]/bn[i-1] +
          (an[i-q]*bn[i])*multix[i-q]/bn[i-q]
      }
    }
    return(multix)
  }

  #Specific dates
  multimodel  <- c()
  multimodelp <- c()
  phasepars   <- list()
  for(i in 1:length(dates)){
    phase    <- dates[[i]]
    phasedat <- countrydat[countrydat$date >= phase[1] & countrydat$date <= phase[2],]

    #get basic model for each phase first in order to get
    # starting a and b to guess for periodic an and bn

    aseq    <- seq(from = 0.1, to = 2.5, length.out = 50)
    bseq    <- seq(from = 0.1, to = 0.9, length.out = 50)
    qseq    <- 6:8
    normdat <- expand.grid(q = qseq, a = aseq, b = bseq)

    if(i == 1)
      abnorm <- apply(normdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, start
          =TRUE), phasedat$xn))
    else
```

```
100       abnorm <- apply(normdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, oldp
              = phasepars[[i-1]])[length(multimodel) + 1:nrow(phasedat)], phasedat$xn))
101
102     normalize <- function(x){
103       return((x-min(x))/(max(x)-min(x)))
104     }
105     normdat$abnorm  <- abnorm
106     #newnormdat      <- normdat %>% top_n(abnorm,n = -0.1*nrow(.))
107
108     #if(i == 1)
109     #  abnormy <- apply(newnormdat, 1, function(x) modnorm(beforecumcases + xntoyn(multimodx(
              phasedat$xn, multimodel, x[1:3], start=TRUE)), phasedat$yn))
110     #else
111     #  abnormy <- apply(newnormdat, 1, function(x) modnorm(beforecumcases + xntoyn(multimodx(
              phasedat$xn, multimodel, x[1:3], oldp = phasepars[[i-1]]))[length(multimodel)+1:nrow(
              phasedat)], phasedat$yn))
112
113     #newnormdat$abnormy  <- normalize(abnormy)
114     #newnormdat$combnorm <- newnormdat$abnorm + newnormdat$abnormy
115     tileoptim <- normdat[which.min(normdat$abnorm),1:3]
116     #tileoptim <- newnormdat[which.min(newnormdat$combnorm),1:3]
117     optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
118
119     q <- optimpars[1]
120     a <- optimpars[2]
121     b <- optimpars[3]
122
123     phasepars[[i]] <- round(optimpars,3)
124
125     if(i == 1 & i != length(dates))
126       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE)
127     if(i == 1 & i == length(dates))
128       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE, len=forecastlen)
129     if(i > 1 & i == length(dates))
130       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]], len=
              forecastlen)
131     if(length(dates) > 2 & i %in% 2:(length(dates)-1))
132       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]])
133
134     aseqper <- a*seq(from = 0.7, to = 1.3, length.out = 10)
135     bseqper <- b*seq(from = 0.7, to = 1.3, length.out = 10)
136     c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 10)
137     n_1seq  <- n_2seq <- c(1,7)
138     p_1seq  <- p_2seq <- 6:7
139     normdatp <- expand.grid(a  = aseqper, b  = bseqper,
140                             c1 = c_1seq,  c2 = c_2seq,
141                             p1 = p_1seq,  p2 = p_2seq,
142                             n1 = n_1seq,  n2 = n_2seq)
143
144     if(i == 1)
145       pernorm <- apply(normdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1],
              phasedat$xn, multimodelp, start=TRUE), phasedat$xn))
146     else
147       pernorm <- apply(normdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1],
              phasedat$xn, multimodelp)[length(multimodelp)+1:nrow(phasedat)], phasedat$xn))
148
149     normdatp$pernorm <- normalize(pernorm)
150
151     newnormdatp <- normdatp %>% top_n(pernorm,n = -0.05*nrow(.))
152
153     if(i == 1)
154       pernormy <- apply(newnormdatp, 1, function(par) modnorm(beforecumcases+xntoyn(multimodxper(
              par, q = optimpars[1], phasedat$xn, multimodelp, start=TRUE)), phasedat$yn))
155     else
156       pernormy <- apply(newnormdatp, 1, function(par) modnorm(beforecumcases+xntoyn(multimodxper(
              par, q = optimpars[1], phasedat$xn, multimodelp))[length(multimodelp)+1:nrow(phasedat)
              ], phasedat$yn))
157
158     newnormdatp$pernormy <- normalize(pernormy)
159     newnormdatp$combnorm <- newnormdatp$pernorm + newnormdatp$pernormy
160
161     peroptim <- as.numeric(newnormdatp[which.min(newnormdatp$combnorm),1:8])
162
163     phasepars[[i]] <- c(phasepars[[i]],round(peroptim,3))
```

```
164
165     if(i == 1 & i != length(dates))
166       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE)
167     if(i == 1 & i == length(dates))
168       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE, len=
              forecastlen)
169     if(i > 1 & i == length(dates))
170       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, len=forecastlen)
171     if(length(dates) > 2 & i %in% 2:(length(dates)-1))
172       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp)
173   }
174
175   cols <- list(
176     xn       = wes_palettes$Zissou1[1],
177     yn       = wes_palettes$Darjeeling2[2],
178     multixn  = wes_palettes$Darjeeling1[1],
179     multiyn  = wes_palettes$Darjeeling1[1],
180     multip   = "magenta4",
181     x3       = wes_palettes$FantasticFox1[2]
182   )
183
184   labs <- list(
185     xn = list(bquote(.(country)*","~x[n]*"*=new cases/day, actual till"~.(format.Date(latest_date
            ,"%d.%m.%Y")))),
186     yn = list(bquote(.(country)*","~y[n]*"*=cumulative cases, actual till"~.(format.Date(latest_
            date,"%d.%m.%Y"))))
187   )
188
189   multimodnormval  <- modnorm(multimodel[1:length(countrydat$xn)], countrydat$xn)
190   multimodnormyval <- modnorm(beforecumcases+xntoyn(multimodel[1:length(countrydat$xn)]),
          countrydat$yn)
191
192   b.roman <- function(x){ return(paste0("(", as.roman(x), ")"))}
193
194   multilabd <- c()
195   for(i in 1:length(dates)){
196     multilabd <- c(multilabd, paste(b.roman(i), "from", format.Date(as.Date(as.character(dates[[i
            ]][1])),"%d.%m")))
197   }
198   multilabd <- paste0(multilabd, collapse = "; ")
199
200   multilabp <- c()
201   for(i in 1:length(dates)){
202     multilabp <- c(multilabp, paste0(b.roman(i), " a=", phasepars[[i]][[2]], ", b=", phasepars[[i
            ]][[3]]))
203   }
204   multilabp <- paste0(multilabp, paste0("; q=", phasepars[[i]][[1]]), collapse = "; ")
205
206   labs$multixn <- list(bquote("base"~.(length(dates))*"-phase model"~x[n]*"=new cases/day: "*
207                               .(multilabd)*
208                               "; ||x*-x||="*.(multimodnormval)*"  "*
209                               .(multilabp)))
210   labs$multiyn <- list(bquote("base"~.(length(dates))*"-phase model"~y[n]*"=cumulative cases: "*
211                               .(multilabd)*
212                               "; ||y*-y||="*.(multimodnormyval)*"  "*
213                               .(multilabp)))
214
215   mavgx1 <- movingavg(countrydat$xn[!is.na(countrydat$xn)])
216
217   countrydat$mavgx3 <- movingavg(mavgx1)
218
219   x3norm  <- modnorm(countrydat$xn,countrydat$mavgx3)
220   labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*.(x3norm)))
221
222   modeldat <- data.frame(date     = c(countrydat$date,as.Date(latest_date) + 1:forecastlen),
223                          multixn  = multimodel,
224                          multiyn  = beforecumcases + xntoyn(multimodel),
225                          multipxn = multimodelp,
226                          multipyn = beforecumcases + xntoyn(multimodelp))
227
228   plots[["xn"]] <- plot_multixn(countrydat, modeldat, cols, labs)
229
230   plots[["yn"]] <- plot_multiyn(countrydat, modeldat, cols, labs)
231
```

```
  multilabp <- c()
  for(i in 1:length(dates)){
    multilabp <- c(multilabp,  paste0(b.roman(i),  " a=", phasepars[[i]][[2]], ", b=", phasepars
        [[i]][[3]]))
  }
  multilabp <- paste0(multilabp, paste0("; q=", phasepars[[i]][[1]]), collapse = "; ")

  labs$multipxn <- list(bquote("periodic"~.(length(dates))*"-phase model"~x[n]*"=new cases/day: "
      *
                               .(multilabd)*
                               "; ||x*-x||="*.(multimodnormval)*"  "*
                               .(multilabp)))
  labs$multipyn <- list(bquote("periodic"~.(length(dates))*"-phase model"~y[n]*"=cumulative cases
      : "*
                               .(multilabd)*
                               "; ||y*-y||="*.(multimodnormyval)*"  "*
                               .(multilabp)))

  plots[["perxn"]] <- plot_multiperxn(countrydat, modeldat, cols, labs)

  plots[["peryn"]] <- plot_multiperyn(countrydat, modeldat, cols, labs)

  return(plots)
}

for(country in names(multidates)){
  multilist[[country]] <- multiphasePlots(country, multidates[[country]], owiddat)
}
```

Listing 12: multi-phase models

```
# load required libraries
library(ggplot2)
library(rgdal)
library(raster)
library(wesanderson)
library(dplyr)

countyplotlist <- list()

# read the shape files
setwd("~/GitHub/TCD_FinalYearProject/Data/")
countyshp <- readOGR("counties/counties.shp")
worldshp  <- readOGR("world/world.shp")

# read the county case data
countycases <- read.csv("Data/Covid19CountyStatisticsHPSCIreland.csv")
owiddat     <- read.csv("Data/owid-covid-data.csv")
owiddat$date          <- as.Date(owiddat$date)
countycases$TimeStamp <- as.Date(countycases$TimeStamp)

# just latest date
latest_date <- countycases$TimeStamp[nrow(countycases)]
latest_dat  <- countycases[countycases$TimeStamp == latest_date,]
fortnightbefore_dat <- countycases[countycases$TimeStamp == latest_date-13,]
latest_dat$ConfirmedCovidCases <- latest_dat$ConfirmedCovidCases - fortnightbefore_dat$
    ConfirmedCovidCases
# make shape data ggplot-friendly
countyshp@data$id <- rownames(countyshp@data)
countyshp.points  <- fortify(countyshp, region="id")
counties <- inner_join(countyshp.points, countyshp@data, by="id")
counties$CountyName <- gsub("County ", "", counties$NAME_EN)

# join case numbers for latest date to county data, in order to colour nicely
countycase_map <- left_join(counties, latest_dat, by=c("CountyName" = "CountyName"))

# color gradient
col_grad <- wes_palette("Zissou1", 20, type = "continuous")

#theme for map plotting
map_theme <- function(){
  p<- theme(axis.title       = element_blank(),
            axis.text        = element_blank(),
            axis.ticks       = element_blank(),
```

```
            panel.background = element_blank(),
            legend.title      = element_blank(),
            legend.background = element_blank())
return(p)
}

# county plots
countyplotlist[["rep"]] <- ggplot(countycase_map) +
  aes(long, lat, group=group, fill=PopulationProportionCovidCases) +
  geom_polygon(colour="grey40") + labs(fill = "Cases per 100k") +
  scale_fill_gradientn(colours = col_grad) +
  geom_text(data = latest_dat, aes(x = Long, y = Lat, label = floor(
      PopulationProportionCovidCases)), inherit.aes = FALSE) +
  ggtitle("Cases in Ireland per 100,000 population by county",
          subtitle = paste("Cumulative, up to", format.Date(latest_date, "%B %d, %Y"))) +
  map_theme() + theme(legend.position   = c(0.25,0.87))

countyplotlist[["fourteendaycases"]] <- ggplot(countycase_map) +
  aes(long, lat, group=group, fill=ConfirmedCovidCases) +
  geom_polygon(colour="grey40") + labs(fill = "Cases") +
  scale_fill_gradientn(colours = col_grad) +
  geom_text(data = latest_dat, aes(x = Long, y = Lat, label = floor(ConfirmedCovidCases)),
      inherit.aes = FALSE) +
  ggtitle("Cases in Ireland by county",
          subtitle = paste("From", format.Date(latest_date-13, "%B %d, %Y"),
                           "to", format.Date(latest_date, "%B %d, %Y"))) +
  map_theme() + theme(legend.position   = c(0.25,0.87))

countyplotlist[["names"]] <- ggplot(countycase_map) +
  aes(long, lat, group=group, fill=PopulationProportionCovidCases) +
  geom_polygon(colour="grey40") + labs(fill = "Cases per 100k") +
  scale_fill_gradientn(colours = col_grad) +
  geom_text(data = latest_dat, aes(x = Long, y = Lat, label = CountyName), size=3,inherit.aes =
      FALSE) +
  ggtitle("Cases in Ireland per 100,000 population by county",
          subtitle = paste("Cumulative, up to", format.Date(latest_date, "%B %d, %Y"))) +
  map_theme() + theme(legend.position   = c(0.25,0.87))

countyplotlist[["blank"]] <- ggplot(countycase_map) +
  aes(long, lat, group=group, fill=PopulationProportionCovidCases) +
  geom_polygon(colour="grey40") + labs(fill = "Cases per 100k") +
  scale_fill_gradientn(colours = col_grad) +
  ggtitle("Cases in Ireland per 100,000 population by county",
          subtitle = paste("Cumulative, up to", format.Date(latest_date, "%B %d, %Y"))) +
  map_theme() + theme(legend.position   = c(0.25,0.87))

# just latest date
latest_date <- as.Date(owiddat$date[nrow(owiddat)-1], tryFormats = c("%Y-%m-%d"))
latest_dat  <- owiddat[owiddat$date == latest_date,]
fortnightRows <- owiddat$date >= latest_date-13 & owiddat$date <= latest_date
fortnightCases <- aggregate(owiddat$new_cases_per_million[fortnightRows],
                            by=list(owiddat$location[fortnightRows]), function(x) sum(x[!is.na(
                                x)]))
colnames(fortnightCases) <- c("location", "fortnight_cases_per_million")
latest_dat <- left_join(latest_dat, fortnightCases, by=c("location" = "location"))


# make shape data ggplot-friendly
worldshp@data$id <- rownames(worldshp@data)
worldshp.points  <- fortify(worldshp, region="id")
countries <- inner_join(worldshp.points, worldshp@data, by="id")

# join case numbers for latest date to country data, in order to colour nicely
world_map <- left_join(countries, latest_dat, by=c("CNTRY_NAME" = "location"))
world_map <- world_map[world_map$lat >= -75,]

worldplot <- list()

worldplot[["blank"]] <- ggplot(world_map) +
  aes(long, lat, group=group, fill=fortnight_cases_per_million) +
  geom_polygon(colour="grey40") + labs(fill = "Cases per million") +
  scale_fill_gradientn(colours = col_grad) +
  ggtitle("Cases per 1 million population by country",
          subtitle =  paste("From", format.Date(latest_date-13, "%B %d, %Y"), "to", format.Date(
```

```
                latest_date, "%B %d, %Y"))) +
  map_theme() + theme(plot.margin     = margin(0, 0, 0, 0, "cm"),
                      legend.position = c(0.1,0.4))

worldplot[["cumulative"]] <- ggplot(world_map) +
  aes(long, lat, group=group, fill=total_cases_per_million) +
  geom_polygon(colour="grey40") + labs(fill = "Cases per million") +
  scale_fill_gradientn(colours = col_grad) +
  ggtitle("Total cases per 1 million population by country",
          subtitle =  paste("Up to", format.Date(latest_date, "%B %d, %Y"))) +
  map_theme() + theme(plot.margin     = margin(0, 0, 0, 0, "cm"),
                      legend.position = c(0.1,0.4))

europe_map <- world_map[world_map$long >= -20 & world_map$long <= 40,]
europe_map <- europe_map[europe_map$lat >= 35 & europe_map$lat <= 75,]


worldplot[["europe"]] <- ggplot(europe_map) +
  aes(long, lat, group=group, fill=fortnight_cases_per_million) +
  geom_polygon(colour="grey40") + labs(fill = "Cases per million") +
  scale_fill_gradientn(colours = col_grad) +
  ggtitle("Total cases per 1 million population by country",
          subtitle =  paste("From", format.Date(latest_date-13, "%B %d, %Y"), "to", format.Date(
              latest_date, "%B %d, %Y"))) +
  map_theme() + theme(plot.margin     = margin(0, 0, 0, 0, "cm"),
                      legend.position = c(0.1,0.3))
```

Listing 13: Geospatial/Map plots

```
source("Code/covid-plotutils.R")
source("Code/covid-modelutils.R")

owiddat      <- read.csv("Data/owid-covid-data.csv")
owiddat$date <- as.Date(owiddat$date, tryFormats = c("%Y-%m-%d"))

comparelist  <- list()
comparepairs <- list("IreUK"  = c("Ireland", "United Kingdom"),
                     "IreUS"  = c("Ireland", "United States"),
                     "IreIta" = c("Ireland", "Italy"),
                     "IreNl"  = c("Ireland", "Netherlands"),
                     "IreIce" = c("Ireland", "Iceland"))

compDates <- c("2020-12-15", "2021-02-15")

compare_theme <- function(){
  p <- theme(axis.text.x        = element_text(vjust=0.5),
             axis.line          = element_line(),
             panel.background   = element_rect(fill  = "grey"),
             panel.grid         = element_line(colour = "darkgrey"),
             panel.grid.major.x = element_blank(),
             panel.grid.minor.x = element_blank(),
             legend.key         = element_blank(),
             legend.key.size    = unit(0.8,"line"),
             legend.text        = element_text(size = 8))
  return(p)
}

compareplots <- function(dat, countries, dates){
  getcountrydat <- function(dat, country, dates){
    cind <- grep(country, dat$location)
    countrydat <- data.frame(date = dat$date[cind], casepm = dat$new_cases_per_million[cind])
    #Specific dates
    countrydat <- countrydat[countrydat$date >= dates[1] & countrydat$date <= dates[2],]
    return(countrydat)
  }

  countryA    <- countries[1]
  countryB    <- countries[2]
  countryAdat <- getcountrydat(owiddat, countryA, compDates)
  countryBdat <- getcountrydat(owiddat, countryB, compDates)

  compcols <- wes_palettes$Darjeeling1[1:2]
  p <- ggplot(countryAdat) +
    geom_point(data = countryAdat, aes(x = date, y = casepm, colour = countryA)) +
```

```
    geom_line(data = countryAdat, aes(x = date, y = casepm, colour = countryA)) +
    geom_point(data = countryBdat, aes(x = date, y = casepm, colour = countryB)) +
    geom_line(data = countryBdat, aes(x = date, y = casepm, colour = countryB)) +
    gg_scale_xy +  ylab(" ") + xlab(" ") + labs(colour = "Country")+
    scale_colour_manual(values = compcols) + compare_theme()
  return(p)
}

for(pair in names(comparepairs)){
  comparelist[[pair]] <- compareplots(owiddat, comparepairs[[pair]], compDates)
}

compdaterange <- owiddat$date >= compDates[1] & owiddat$date <= compDates[2]
countriescmp <- c("Ireland", "United Kingdom", "United States", "Italy", "Germany")
comparelist[["all"]] <- ggplot(owiddat[owiddat$location %in% countriescmp & compdaterange,]) +
  geom_point(aes(x = date, y = new_cases_per_million, colour = location)) +
  geom_line(aes(x = date, y = new_cases_per_million, colour = location)) +
  gg_scale_xy +  ylab(" ") + xlab(" ") + labs(colour = "Country")+
  scale_colour_manual(values = wes_palette("Darjeeling1", length(countriescmp))) +
  compare_theme()
```

Listing 14: Country Comparison plots

```
setwd("~/GitHub/TCD_FinalYearProject")
source("Code/covid-main.r")
source("Code/covid-multimodel.r")
source("Code/covid-mapplots.r")
source("Code/covid-compare.r")

for(country in names(plotslist)){
  for(p in names(plotslist[[country]])){
    ggsave(filename = paste0(country, "-", p, ".pdf"),
           plot     = plotslist[[country]][[p]],
           path     = "./Plots",
           height   = 10,
           width    = 14,
           units    = "cm"
           )
  }
}

for(country in names(multilist)){
  for(p in names(multilist[[country]])){
    ggsave(filename = paste0(country, "-", p, "mult.pdf"),
           plot     = multilist[[country]][[p]],
           path     = "Plots",
           height   = 10,
           width    = 16,
           units    = "cm"
    )
  }
}

for(pair in names(comparelist)){
  ggsave(filename = paste0("compare-", pair, ".pdf"),
         plot     = comparelist[[pair]],
         path     = "Plots",
         height   = 10,
         width    = 16,
         units    = "cm"
  )
}

for(p in names(countyplotlist)){
  ggsave(filename = paste0("county-", p, ".pdf"),
         plot     = countyplotlist[[p]],
         path     = "Plots",
         height   = 14,
         width    = 14,
         units    = "cm"
  )
}

for(p in names(worldplot)){
```

```
52    ggsave(filename = paste0("world-", p, ".pdf"),
53           plot     = worldplot[[p]],
54           path     = "Plots",
55           height   = 14,
56           width    = 20,
57           units    = "cm"
58    )
59  }
```

Listing 15: Saving plots

References

[1] Ravi Agarwal et al. "Dynamic equations on time scales: a survey". In: *Journal of Computational and Applied Mathematics* 141.1 (2002). Dynamic Equations on Time Scales, pp. 1 –26. ISSN: 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(01)00432-0. URL: http://www.sciencedirect.com/science/article/pii/S0377042701004320.

[2] L.J.S. Allen et al. *Mathematical Epidemiology*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 2008. ISBN: 9783540789109. URL: https://books.google.ie/books?id=gcP5l1a22rQC.

[3] Roy M. Anderson. "Discussion: The Kermack-McKendrick epidemic threshold theorem". In: *Bulletin of Mathematical Biology* 53 (Mar. 1, 1991). ISSN: 522-9602. DOI: 10.1007/BF02464422. URL: https://doi.org/10.1007/BF02464422.

[4] Derdei Bichara, Abderrahman Iggidr, and Gauthier Sallet. "Global analysis of multi-strains SIS, SIR and MSIR epidemic models". In: *Journal of Applied Mathematics and Computing* 44 (Feb. 2014), pp. 273–292. DOI: 10.1007/s12190-013-0693-x.

[5] Alexander Bird. *A simple introduction to epidemiological modelling—the SIR model*. https://philosophyandmedicine.org/wp-content/uploads/2020/04/Introduction-to-epidemiological-modelling.pdf. 2020.

[6] Roger Bivand, Tim Keitt, and Barry Rowlingson. *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*. R package version 1.5-18. 2020. URL: https://CRAN.R-project.org/package=rgdal.

[7] GeoHive Open Data Catalogue. *Covid-19 Daily Statistics for Ireland by County polygon as reported by the Health Surveillance Protection Centre*. 2020. URL: https://opendata-geohive.hub.arcgis.com/datasets/d9be85b30d7748b5b7c09450b8aede63_0.

[8] © OpenStreetMap contributors. *Townlands*. 2020. URL: https://www.townlands.ie/page/download/.

[9] Our World in Data. *The complete Our World in Data COVID-19 dataset*. 2020. URL: https://covid.ourworldindata.org/data/owid-covid-data.csv.

[10] European Centre for Disease Prevention and Control. *Historical data on the daily number of new reported COVID-19 cases and deaths worldwide*. 2020. URL: https://opendata.ecdc.europa.eu/covid19/casedistribution/.

[11] Jesús Fernández-Villaverde and Charles I Jones. *Estimating and Simulating a SIRD Model of COVID-19 for Many Countries, States, and Cities*. Working Paper 27128. National Bureau of Economic Research, May 2020. DOI: 10.3386/w27128. URL: http://www.nber.org/papers/w27128.

[12] Seth Flaxman. "Reply to: The effect of interventions on COVID-19". In: *Nature* 588.1 (Dec. 1, 2020), pp. 29 –32. ISSN: 1476-4687. DOI: 10.1038/s41586-020-3026-x. URL: https://doi.org/10.1038/s41586-020-3026-x.

[13] Seth Flaxman and Imperial College COVID-19 Response Team. "Estimating the effects of non-pharmaceutical interventions on COVID-19 in Europe". In: *Nature* 584.7820 (2020), pp. 257–261. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2405-7. URL: https://doi.org/10.1038/s41586-020-2405-7.

[14] Dr Tedros Adhanom Ghebreyesus. *WHO Director-General's opening remarks at the media briefing on COVID-19*. World Health Organization. Mar. 11, 2020. URL: https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020.

[15] Anthony Gibbons. *Final Year Project GitHub Repository*. 2021. URL: https://github.com/gibbona1/TCD_FinalYearProject.

[16] Alexander Grigorian. *Mathematical riddles of COVID-19*. June 2020. URL: https://www.math.uni-bielefeld.de/~grigor/corv.pdf.

[17] Robert J. Hijmans. *raster: Geographic Data Analysis and Modeling*. R package version 3.4-5. 2020. URL: https://CRAN.R-project.org/package=raster.

[18] ArcGIS Hub. *UNIGIS Geospatial Education Resources*. 2020. URL: https://hub.arcgis.com/datasets/a21fdb46d23e4ef896f31475217cbb08_1.

[19] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice, 2nd edition*. OTexts.com/fpp2. OTexts, 2018.

[20] Rob J Hyndman and Yeasmin Khandakar. "Automatic time series forecasting: the forecast package for R". In: *Journal of Statistical Software* 26.3 (2008), pp. 1–22. URL: https://www.jstatsoft.org/article/view/v027i03.

[21] Roni Parshani, Shai Carmi, and Shlomo Havlin. "Epidemic Threshold for the Susceptible-Infectious-Susceptible Model on Random Networks". In: *Physical Review Letters* 104.25 (June 2010). ISSN: 1079-7114. DOI: 10.1103/physrevlett.104.258701. URL: http://dx.doi.org/10.1103/PhysRevLett.104.258701.

[22] Karthik Ram. *Wes Anderson Palettes*. 2013. URL: https://github.com/karthik/wesanderson.

[23] Arni S. R. Srinivasa Rao and Jose A. Vazquez. "Identification of COVID-19 can be quicker through artificial intelligence framework using a mobile phone-based survey when cities and towns are under quarantine". eng. In: *Infection control and hospital epidemiology* 41.7 (2020). 32122430[pmid], pp. 826–830. ISSN: 1559-6834. DOI: 10.1017/ice.2020.61. URL: https://pubmed.ncbi.nlm.nih.gov/32122430.

[24] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

[25] Hadley Wickham et al. *dplyr: A Grammar of Data Manipulation*. R package version 1.0.3. 2021. URL: https://CRAN.R-project.org/package=dplyr.