# Modelling the Spread of COVID-19 Cases

TCD Final Year Project

Anthony Gibbons

Project Supervisor: Athanasios Georgiadis

March 21, 2021

**Abstract**

We construct various models of the Covid-19 pandemic for multiple countries in early 2021. 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

# List of Tables

# 1 Introduction

The Coronavirus disease (COVID-19) was first characterized by the World Health Organisation as pandemic on 11th March 2020 [9]. The outbreak has affected almost every aspect of human life throughout 2020, and is expected to continue for much of 2021.

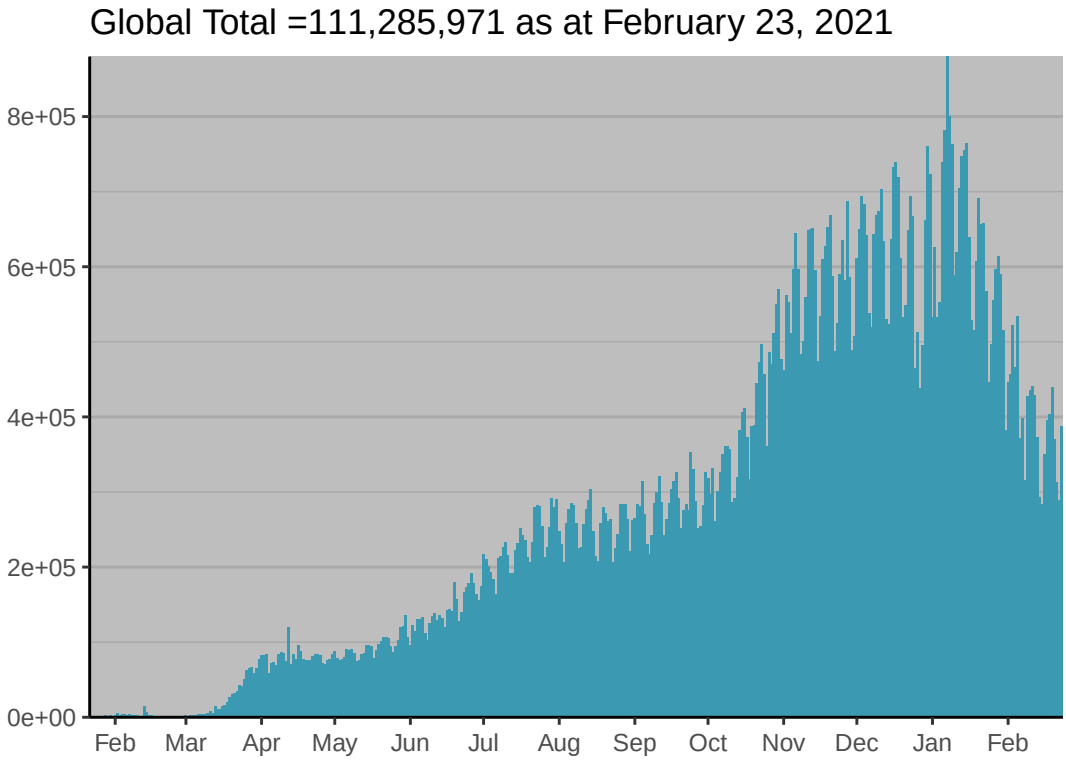



**Figure 1:** Daily Cases Globally

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

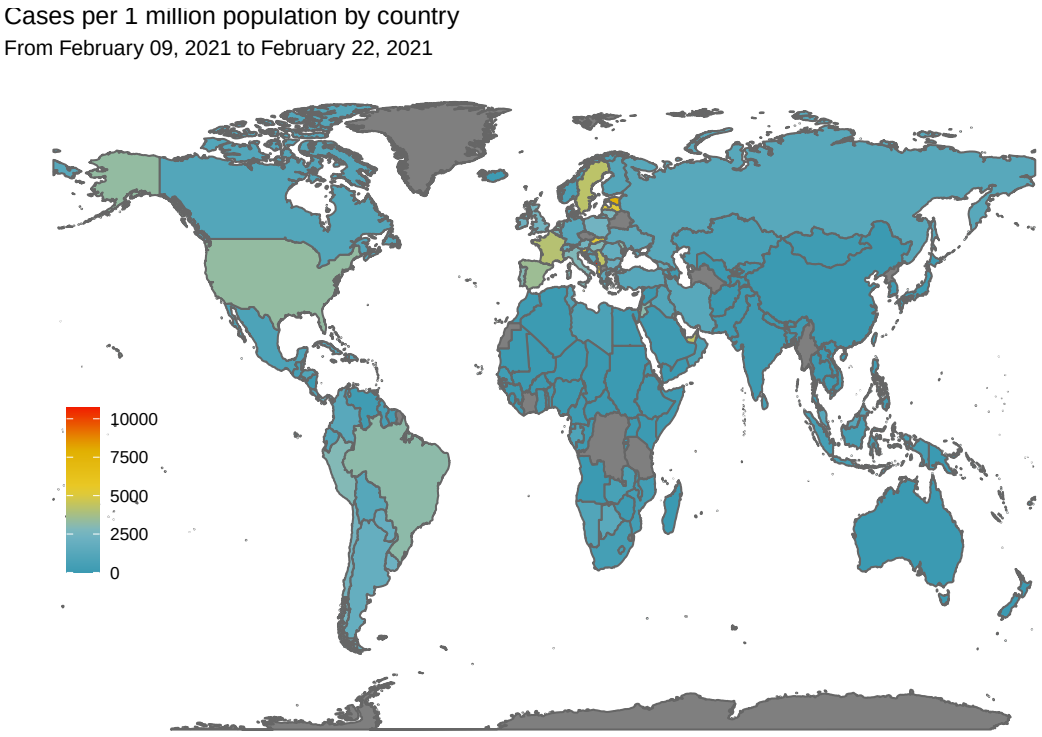



**Figure 2:** Cases per 1 million population, World

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

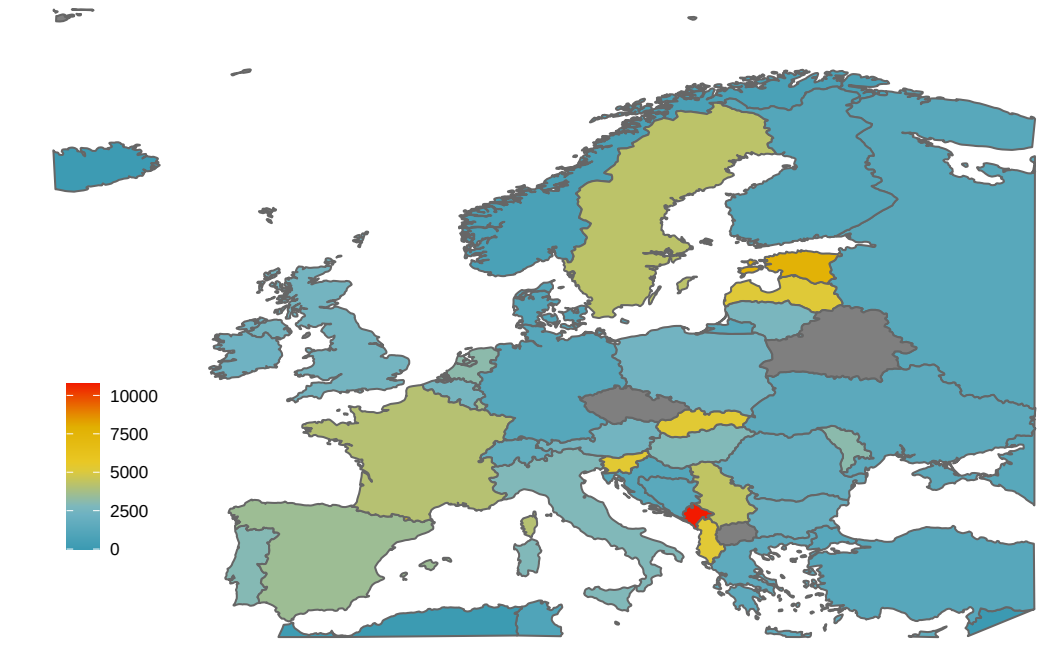



**Figure 3:** Cases per 1 million population, Europe

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.

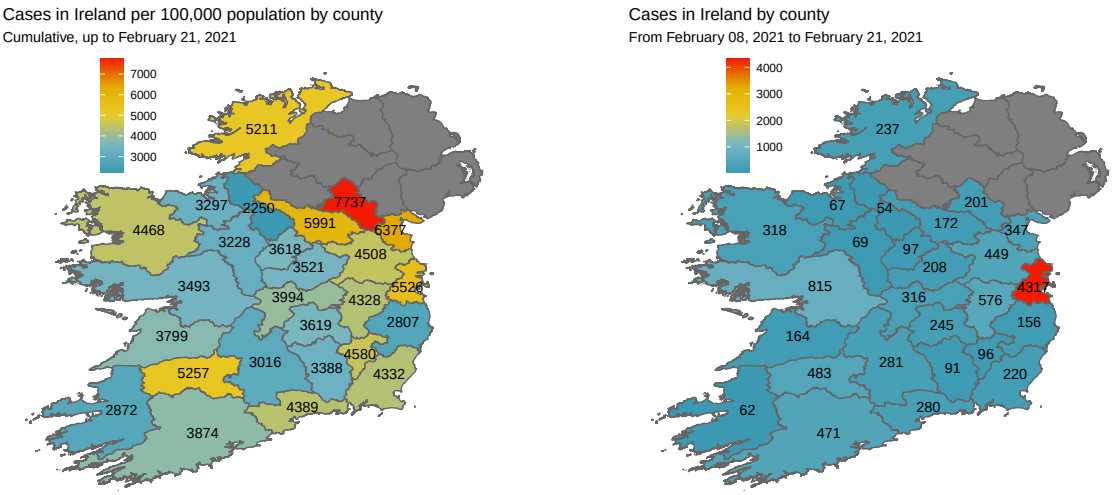



**Figure 4:** Situation by County, Ireland

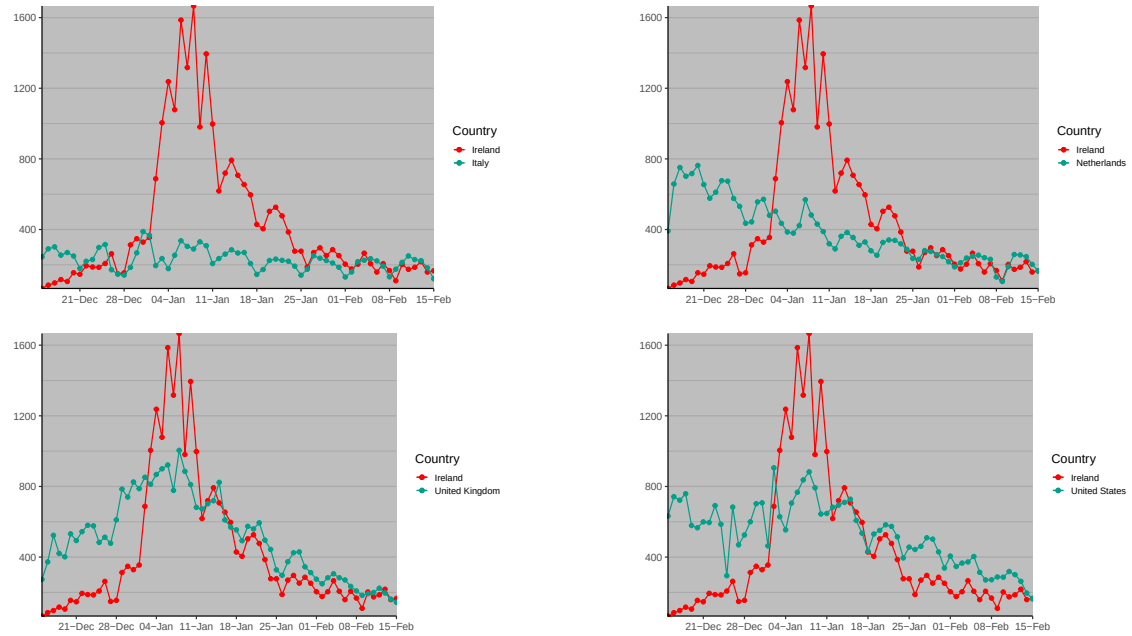



**Figure 5:** Individual Comparison of Ireland with various countries, daily cases per million of the population

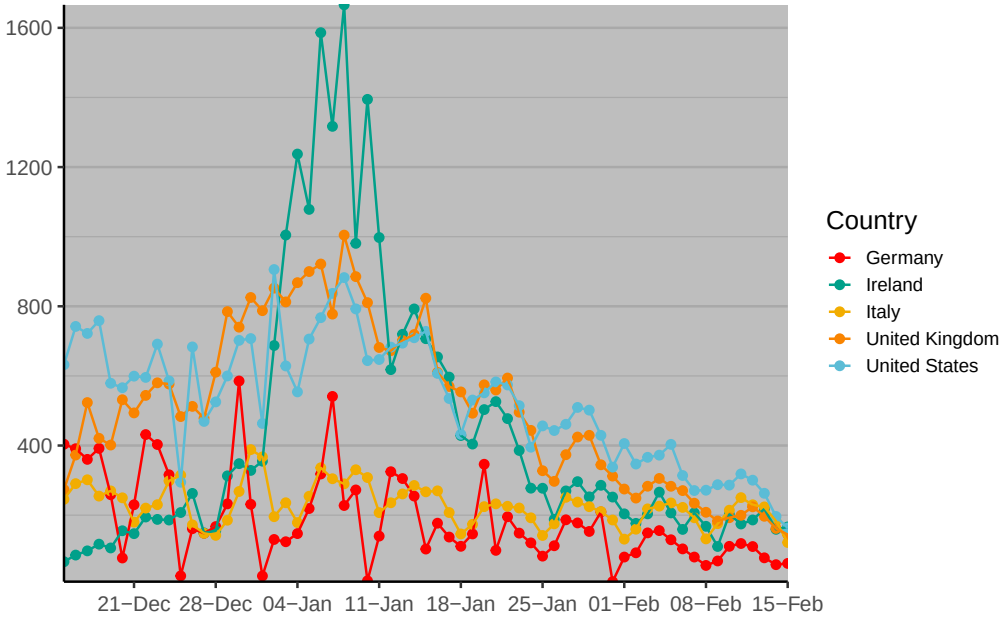



**Figure 6:** Comparison of Ireland with various countries, daily cases per million of the population

## 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 [8], 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.

Artificial intelligence models have also been employed to track disease outbreaks in more local areas. The model developed by [17], 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 [10], 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.

## 1.3 Summary

Below is a summary of results of the applied model for the primary date range

| | model | $\|\|x - x^*\|\|$ | $\|\|y - y^*\|\|$ |
|---|---|---|---|
| 1 | Basic Recursion | 146 | 895 |
| 2 | Moving Average | 131 | 127 |
| 3 | Periodic | 139 | 1030 |
| 4 | HoltWinters | 109 | 334 |
| 5 | ARIMA | 86 | 242 |
| 6 | Neural Network | 92 | 217 |

**Table 1:** Ireland, 2021-01-12 to 2021-02-16

| | model | $\|\|x - x^*\|\|$ | $\|\|y - y^*\|\|$ |
|---|---|---|---|
| 1 | Basic Recursion | 1240 | 5583 |
| 2 | Moving Average | 1086 | 991 |
| 3 | Periodic | 1186 | 5932 |
| 4 | HoltWinters | 818 | 3209 |
| 5 | ARIMA | 973 | 6860 |
| 6 | Neural Network | 1389 | 2458 |

**Table 2:** Italy, 2021-01-02 to 2021-02-16

| | model | $\|\|x - x^*\|\|$ | $\|\|y - y^*\|\|$ |
|---|---|---|---|
| 1 | Basic Recursion | 8716 | 38695 |
| 2 | Moving Average | 7801 | 8319 |
| 3 | Periodic | 5559 | 25111 |
| 4 | HoltWinters | 6473 | 30935 |
| 5 | ARIMA | 8165 | 40136 |
| 6 | Neural Network | 12437 | 16625 |

**Table 3:** United States, 2021-01-06 to 2021-02-16

# 2 Mathematical Models

As per the base and periodic models shown in [10].

## 2.1 Base model

### 2.1.1 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*. The SIR (Susceptible–Infectious–Recovered) model, the most frequently used model in epidemiology, uses a set of 3 such differential equations.

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

### 2.1.2 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 ideally want to minimize both

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

While this is the ideal situation, it is far more important to minimize $||x - x^*||$ as it is generally much more sensitive to variation in the parameters (such as $a$ and $b$).

The proofs and results are in .

[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.

### 2.1.3 How to select the best model

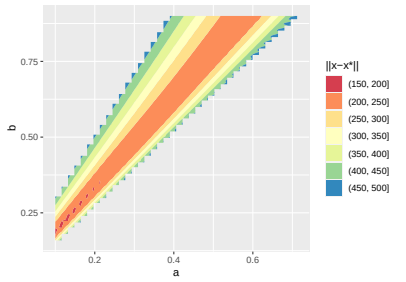



**Figure 7:** Contours, Ireland

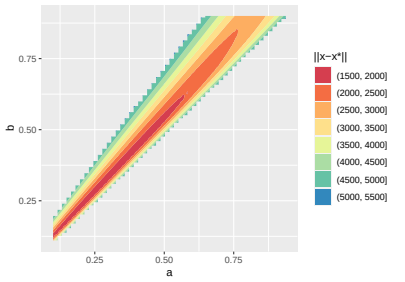



**Figure 8:** Contours, Italy

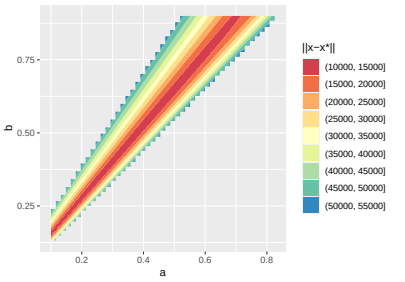



**Figure 9:** Contours, United States

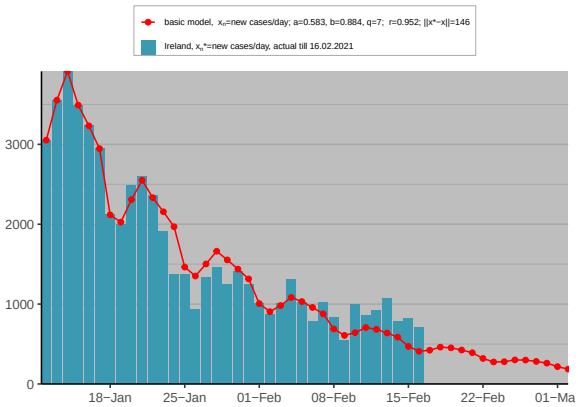
### 2.1.4 Forecasting

### 2.1.5 Implementation in R

```
1  basicmodx <- function(x, pars, len = 0){
2    q <- floor(pars[1])
3    a <- pars[2]
4    b <- pars[3]
5    modx <- x[1:q]
6    for(i in (q+1):(length(x)+len)){
7      modx[i] <- (1-b)*modx[i-1] + a*modx[i-q]
8    }
9    return(modx)
10 }
11 basexn    <- basicmodx(countrydat$xn, optimpars, len = forecastlen)
   ...
   ...
   ...
15 plots[["basexn"]] <- plot_basexn(countrydat, modeldat, cols, labs)
```

**Listing 1:** Algorithm for Base Model

### 2.1.6 Plots



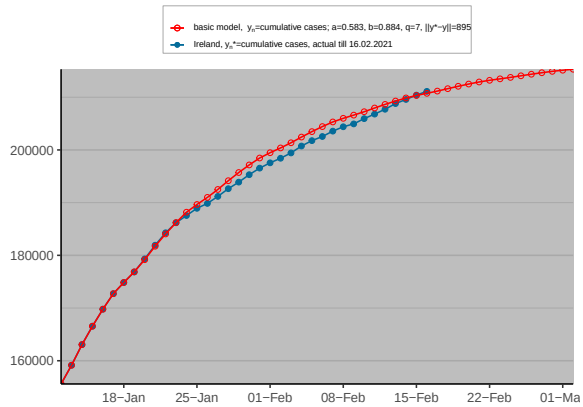




**Figure 10:** Basic model, Ireland

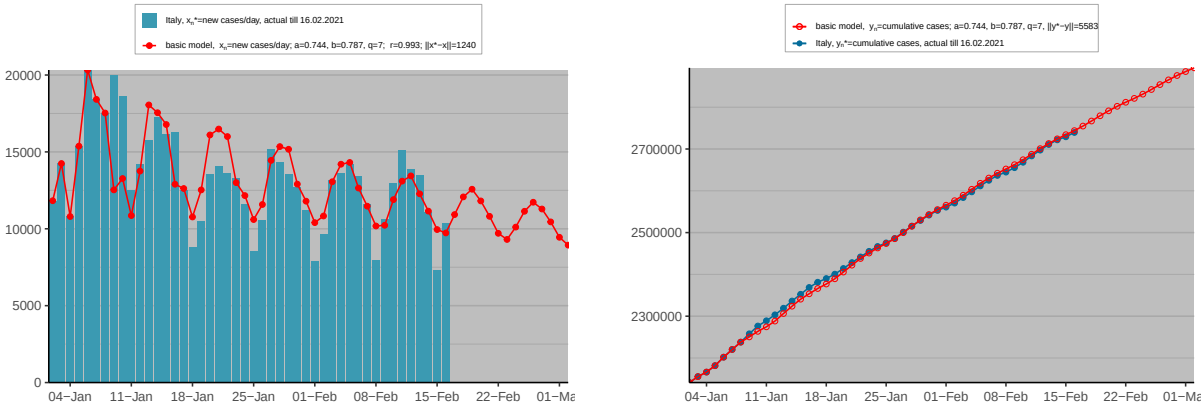



**Figure 11:** Basic model, Italy

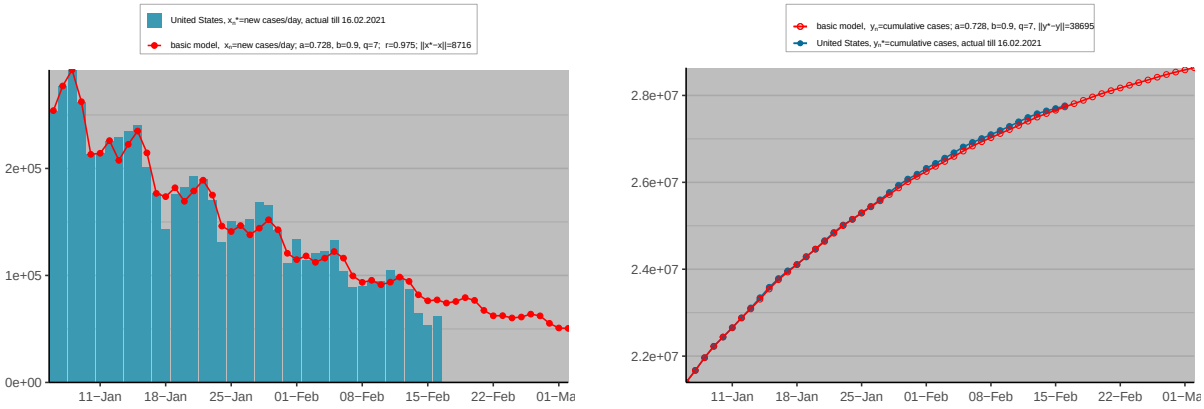



**Figure 12:** Basic model, United States

## 2.2 Limiting curve

The second result in 1 is the equation 20, which defines the limiting behavior of the recurrence relation 17.

```
modeldat$Crn <- optimC*base_r_one^(1:nrow(modeldat))
...
...
...
plots[["Crn"]] <- plot_crn(countrydat, modeldat, cols, labs)
```

**Listing 2:** Algorithm for Limiting Curve

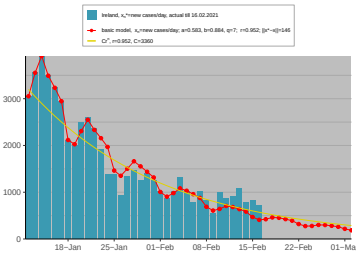


**Figure 13:** Limiting curve $Cr^n$, Ireland

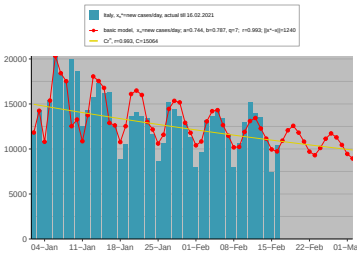


**Figure 14:** Limiting curve $Cr^n$, Italy

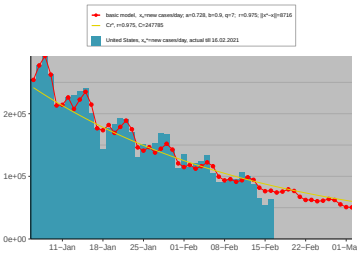


**Figure 15:** Limiting curve $Cr^n$, United States

The limiting curve can of course so exponential growth

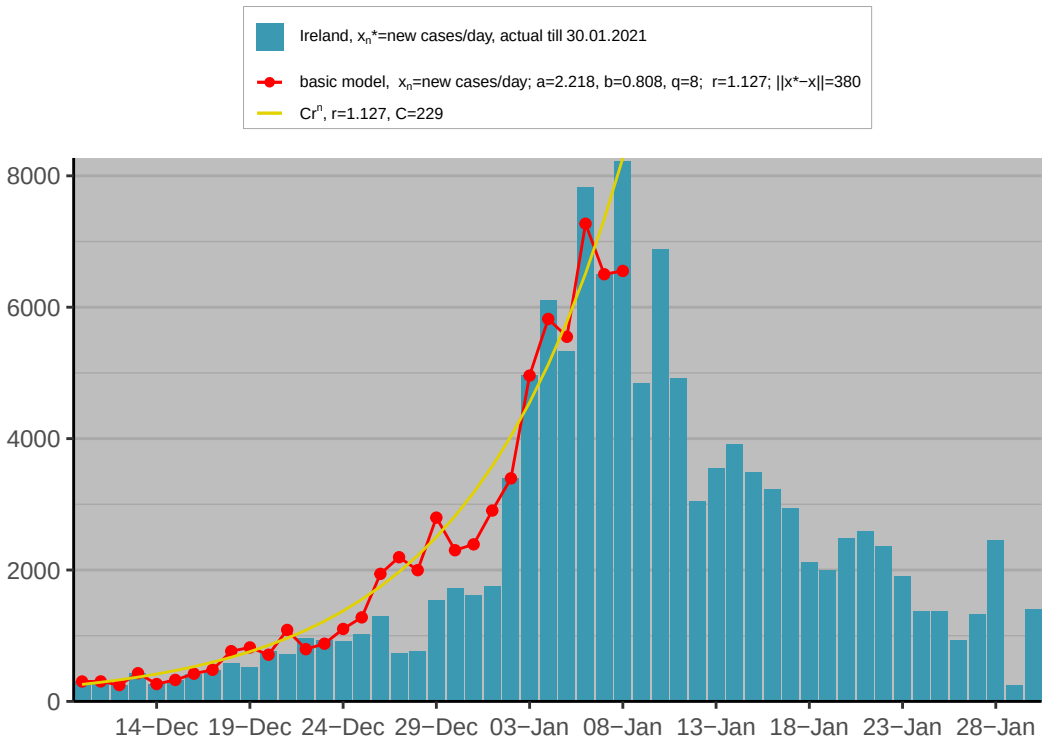



**Figure 16:** Limiting Curve Growing Exponentially, Ireland

## 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.

The 7-day moving average is a good baseline for model performance , as ideally we would want $||x - x^*|| \approx ||x^*(3) - x^*||$ or better.

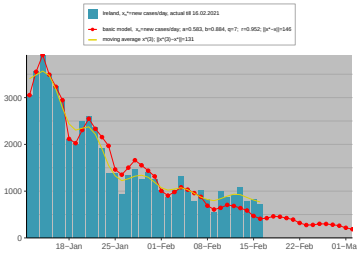


**Figure 17:** Moving average $x_n^*(3)$, Ireland

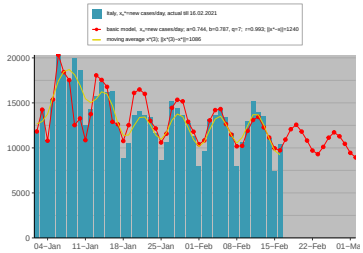


**Figure 18:** Moving average $x_n^*(3)$, Italy

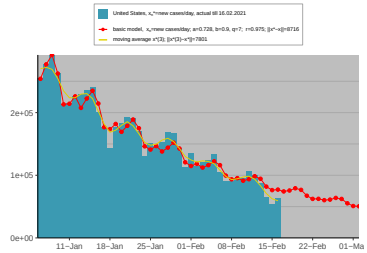


**Figure 19:** Moving average $x_n^*(3)$, United States

## 2.4 Periodic model

### 2.4.1 Definitions and Theory

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

$$a_n := a\left(1 + c_1\left(\sin\left(\frac{2\pi}{p_1}\left(n - n_1\right)\right)\right)\right) \tag{7}$$

$$b_n := b\left(1 + c_2\left(\sin\left(\frac{2\pi}{p_2}\left(n - n_2\right)\right)\right)\right) \tag{8}$$

For new parameters $c_i, p_i, n_i,\ i = 1, 2$ where
$c_i \in [0.04, 0.2]$ small are amplitudes,
$n_i \in 1, 2, \ldots, q$ are lags and
$p_i \in 1, 2, \ldots, q$ are periods.
We then have to optimize 6 with respect to the 9 parameters $a, b, q, c_1, n_1, p_1, c_2, n_2$ and $p_2$.

While Grigorian's paper [10] optimized with (possibly) new values of $a$ and $b$, I will allow the original $a$ and $b$ from the earlier optimization vary,since the average value should be ablut the same. We will sketch the reasons for this below.

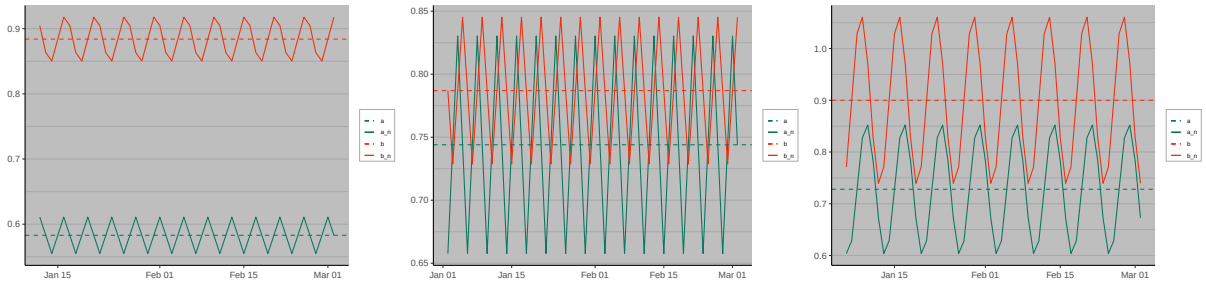



**Figure 20:** Oscillating $a$ and $b$ parameters, Ireland, Italy and United States

### 2.4.2 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)
}
modeldat$periodic  <- modxper(as.numeric(peroptim),countrydat$xn, q = q, forecastlen)
...
...
...
plots[["periodic"]] <- plot_periodic(countrydat, modeldat, cols, labs)
```

**Listing 3:** Algorithm for Periodic Model

### 2.4.3 Plots

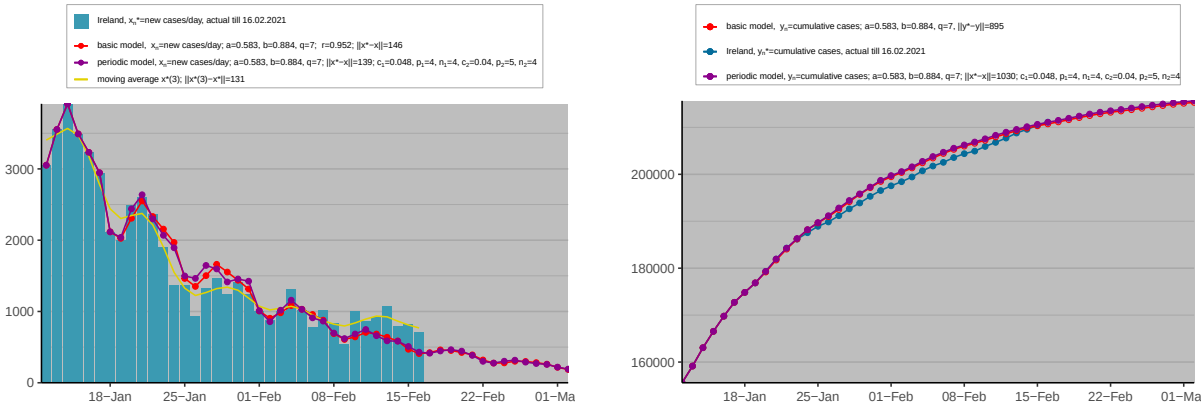



**Figure 21:** Periodic model, Ireland

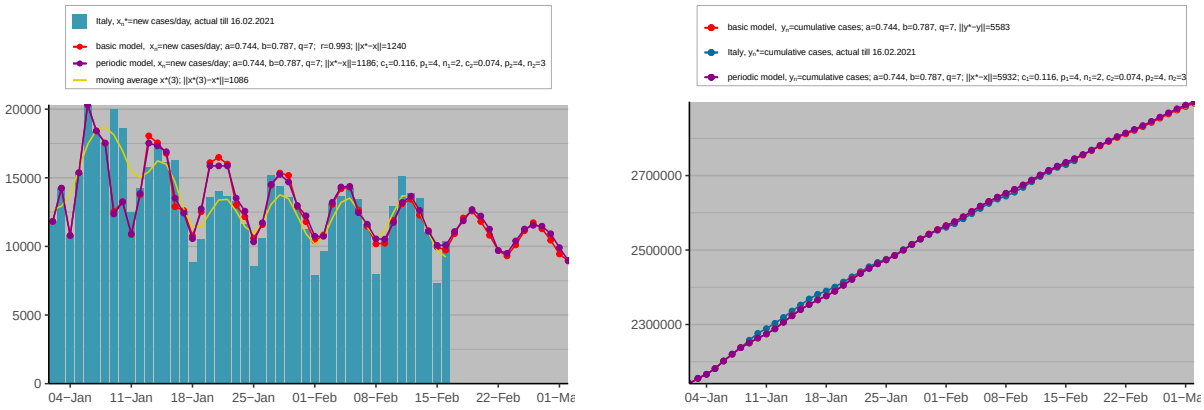



**Figure 22:** Periodic model, Italy

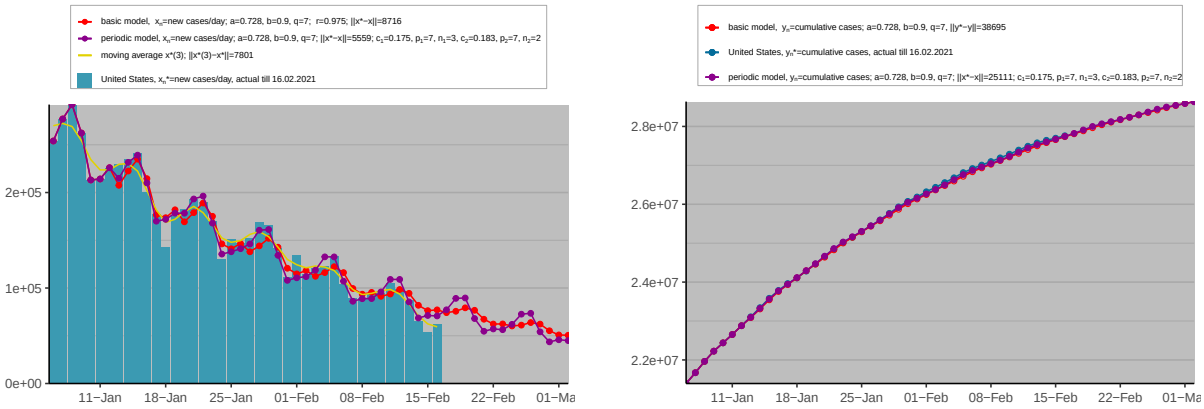



**Figure 23:** Periodic model, United States

## 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

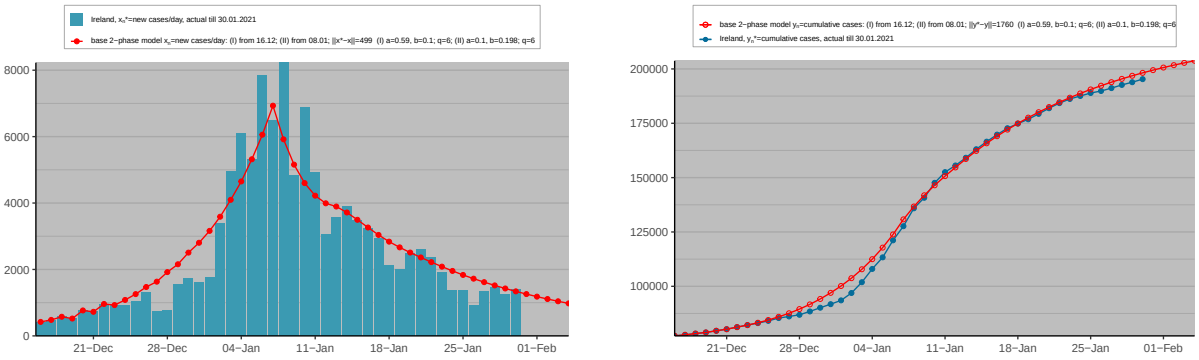



**Figure 24:** Multi-phase model, Ireland

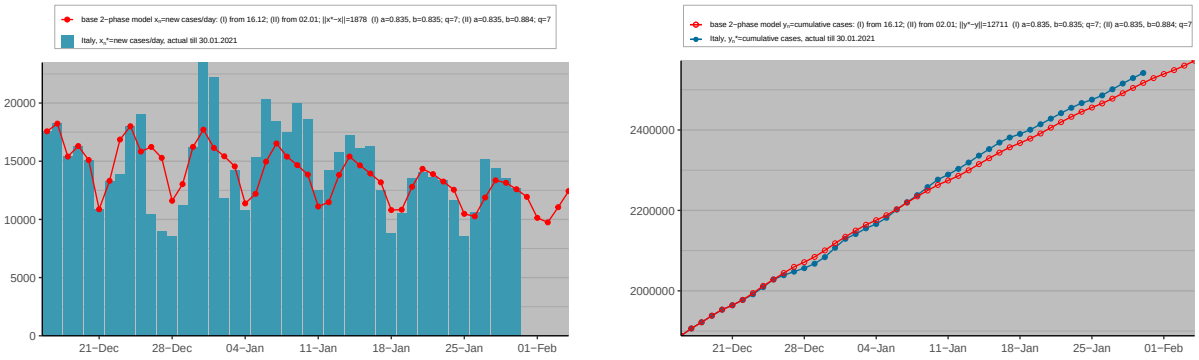



**Figure 25:** Multi-phase model, Italy

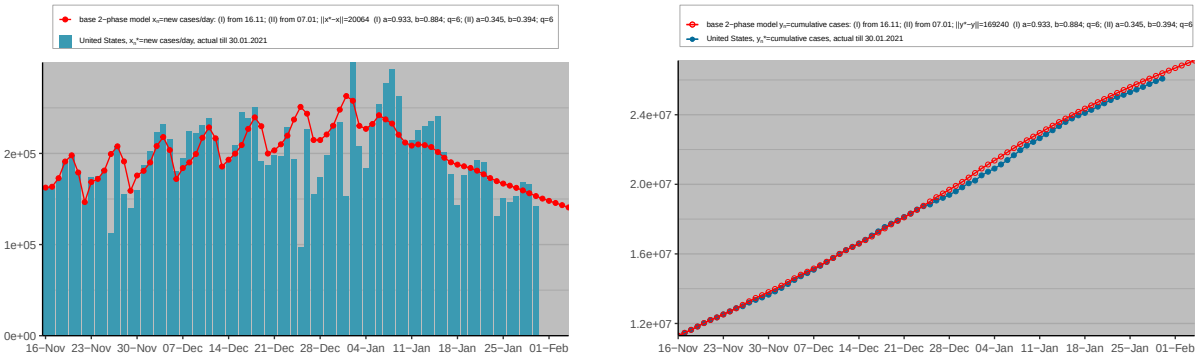



**Figure 26:** Multi-phase model, United States

The periodic model often performs much better

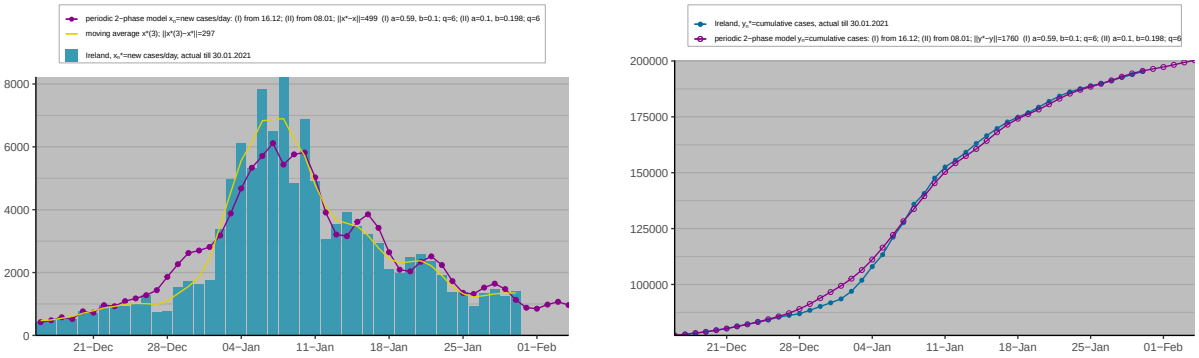



**Figure 27:** Multi-phase periodic model, Ireland

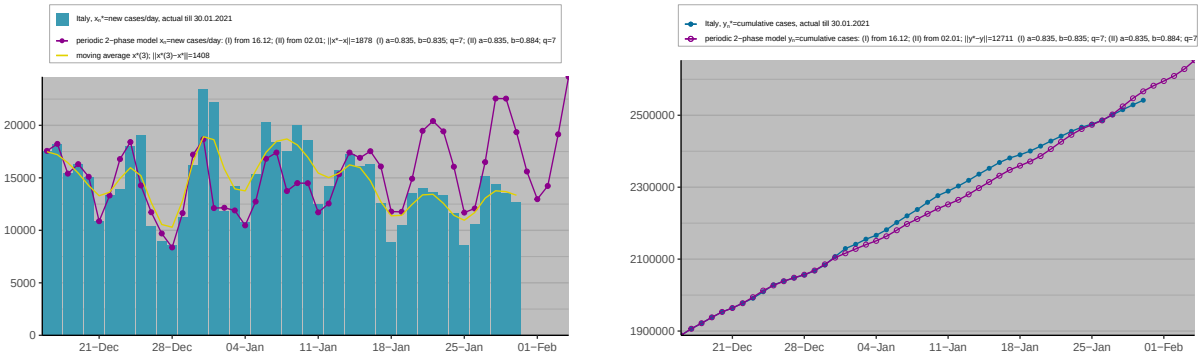



**Figure 28:** Multi-phase periodic model, Italy

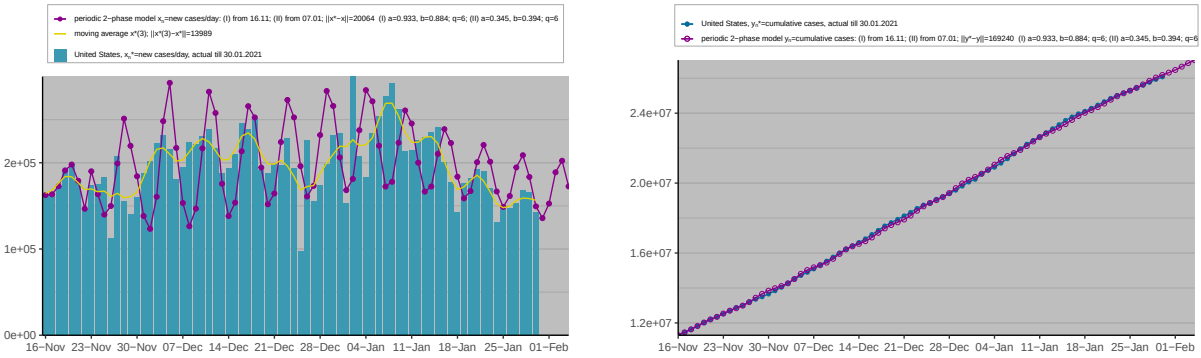



**Figure 29:** Multi-phase periodic model, United States

# 3 Statistical Models

Primary source for this was Hyndman-et-al-2018 [13].

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

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

Using

```
1 forecast::gghistogram(log(dat_ts), add.normal = TRUE, bins = 10)
```

with the full code in 9.

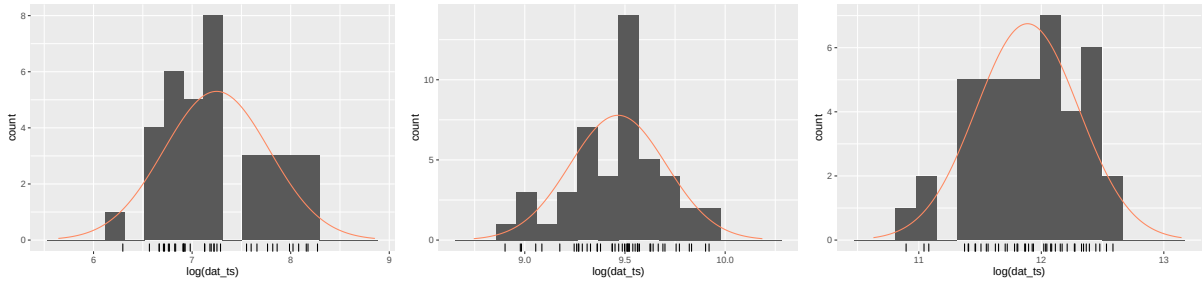



**Figure 30:** Normality checks, Ireland, Italy and United States

## 3.1 Holt-Winters' seasonal method

### 3.1.1 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 31:** Seasonal Holt Winter's Additive Model Algorithm (denoted SHW$_+$)

### 3.1.2 How to select the best model

### 3.1.3 Forecasting

### 3.1.4 Implementation in R

```
1 #lambda=0 ensures values stay positive
2 hwfcst     <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
  ...
  ...
  ...
6 plots[["hw"]]  <- plot_hw(countrydat, modeldat, cols, labs)
```

**Listing 4:** Algorithm for HoltWinters Model

### 3.1.5 Plots

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

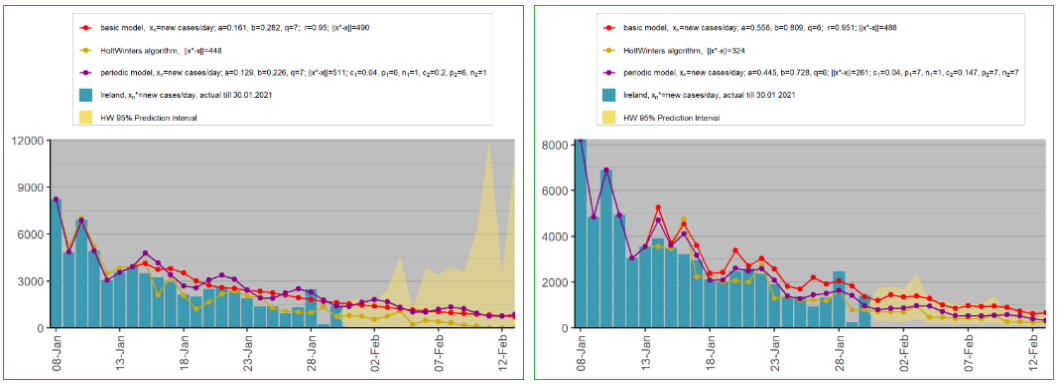



**Figure 32:** Comparison between HoltWinters multiplicative (left, red outline) and additive (right, green outline)algorithms, at some point during the research

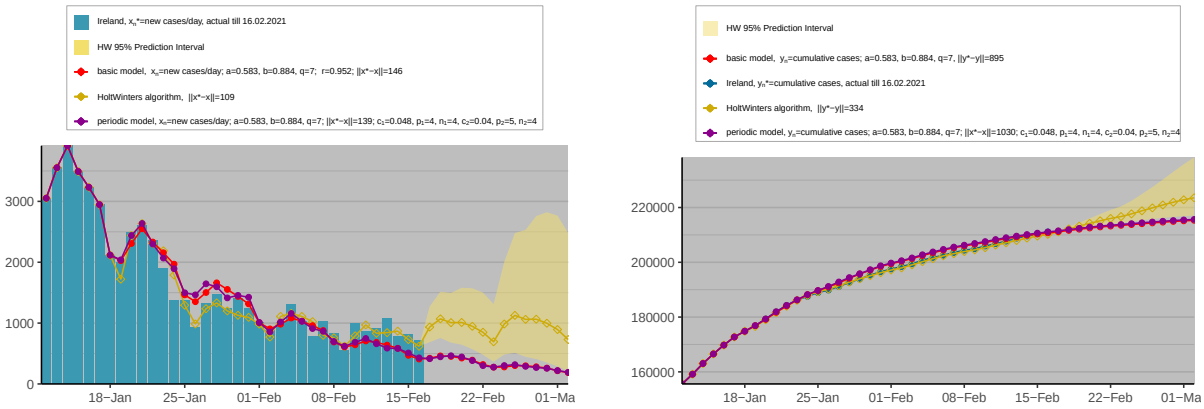



**Figure 33:** HoltWinters model, Ireland

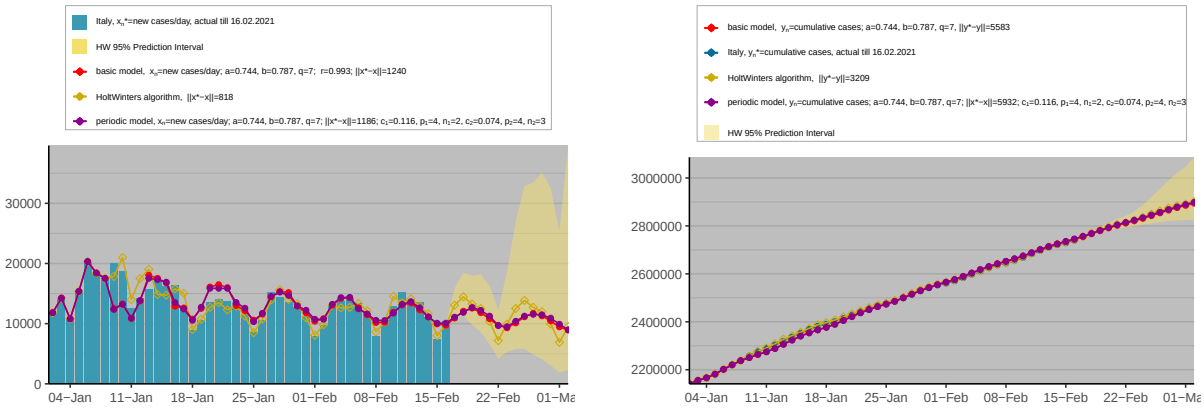



**Figure 34:** HoltWinters model, Italy

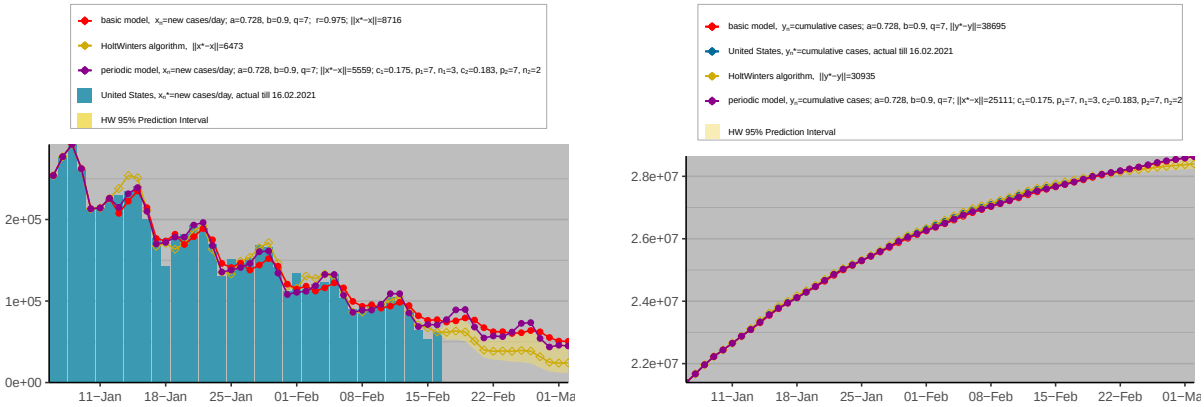



**Figure 35:** HoltWinters model, United States

## 3.2 ARIMA models

### 3.2.1 Definitions and Theory

**Definition 1.** 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 $\varepsilon_n$:

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

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 $B$:

  $$\begin{aligned} x_n &= \phi_0 + \phi_1 B x_n + \phi_2 B^2 x_n + \cdots + \phi_p B^p x_n + \varepsilon_n \\ &= \phi_0 + \left(\phi_1 B + \phi_2 B^2 + \cdots + \phi_p B^p\right) x_n + \varepsilon_n \end{aligned} \tag{9}$$

- 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 $\varepsilon_n$, 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 $B$:

  $$\begin{aligned} x_n &= \psi_0 - \psi_1 B \varepsilon_n - \psi_2 B^2 \varepsilon_n - \cdots - \psi_q B^q \varepsilon_n + \varepsilon_n \\ &= \psi_0 + \left(1 - \psi_1 B - \psi_2 B^2 + \cdots - \psi_q B^q\right) \varepsilon_n \end{aligned} \tag{10}$$

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

  $$\begin{aligned} x_n' &= x_n - x_{n-1} \\ &= x_n - Bx_n \\ &= (1 - B)\, x_n \end{aligned} \tag{11}$$

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 $\varepsilon_n$ are unchanged.

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

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

$$\begin{aligned}(1-B)^d x_n + (1-B)^d \left(-\phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right) x_n &= \phi_0 + \psi_0 + \left(1 - \psi_1 B - \psi_2 B^2 + \cdots - \psi_q B^q\right) \varepsilon_n \\ (1-B)^d \left(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right) x_n &= c + \left(1 - \psi_1 B - \psi_2 B^2 + \cdots - \psi_q B^q\right) \varepsilon_n \end{aligned} \tag{12}$$

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 $B$:

  $$x_n = \beta_0 + \left(\beta_1 B^s + \beta_2 B^{2s} + \cdots + \beta_P B^{Ps}\right) x_n \tag{13}$$

- 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 $\varepsilon_n$, 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 $B$:

  $$\begin{aligned} x_n &= \gamma_0 - \gamma_1 \varepsilon_{n-s} - \gamma_2 \varepsilon_{n-2s} - \cdots - \gamma_Q \varepsilon_{n-Qs} + \varepsilon_n \\ &= \gamma_0 + \left(1 - \gamma_1 B^s - \gamma_2 B^{2s} + \cdots - \gamma_Q B^{Qs}\right) \varepsilon_n \end{aligned} \tag{14}$$

- 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

$$\begin{aligned} &\underbrace{\left(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right)}_{AR(p)} \underbrace{\left(1 - \beta_1 B^s - \beta_2 B^{2s} - \cdots - \beta_P B^{Ps}\right)}_{AR_s(P)} \underbrace{(1-B)^d}_{I(d)} \underbrace{(1-B^s)^D}_{I_s(D)} x_n = \\ &\qquad c + \underbrace{\left(1 - \psi_1 B - \psi_2 B^2 - \cdots - \psi_q B^q\right)}_{MA(q)} \underbrace{\left(1 - \gamma_1 B^s - \gamma_2 B^{2s} - \cdots - \gamma_Q B^{Qs}\right)}_{MA_s(Q)} \varepsilon_n \end{aligned} \tag{15}$$

where the constant $c$ is some function of the constants $\phi_0, \psi_0, \beta_0$ and $\gamma_0$

### 3.2.2 How to select the best model

### 3.2.3 Forecasting

### 3.2.4 Implementation in R

```
#lambda=0 ensures values stay positive
auto.fit <- auto.arima(dat_ts, lambda = 0)
...
...
...
plots[["arima"]]   <- plot_arima(countrydat, modeldat, cols, labs)
```

**Listing 5:** Algorithm for ARIMA Model

### 3.2.5 Plots

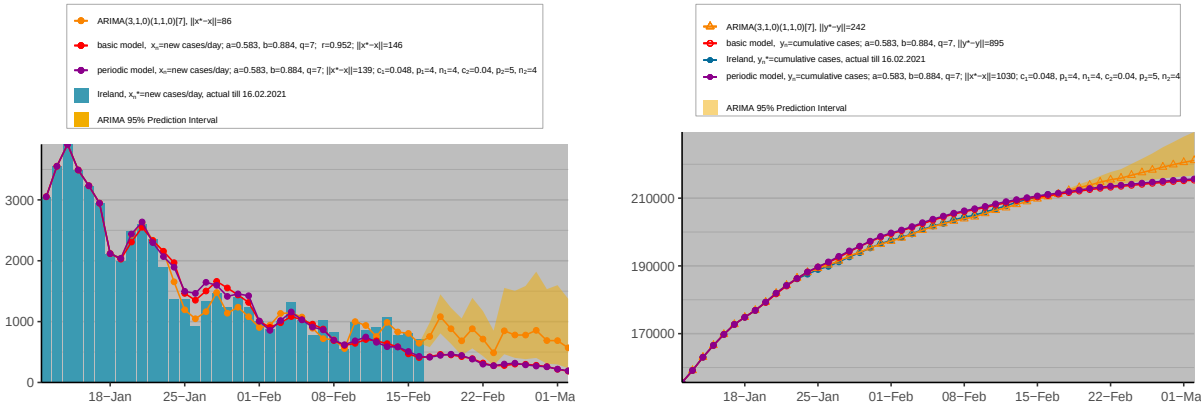



**Figure 36:** ARIMA model, Ireland

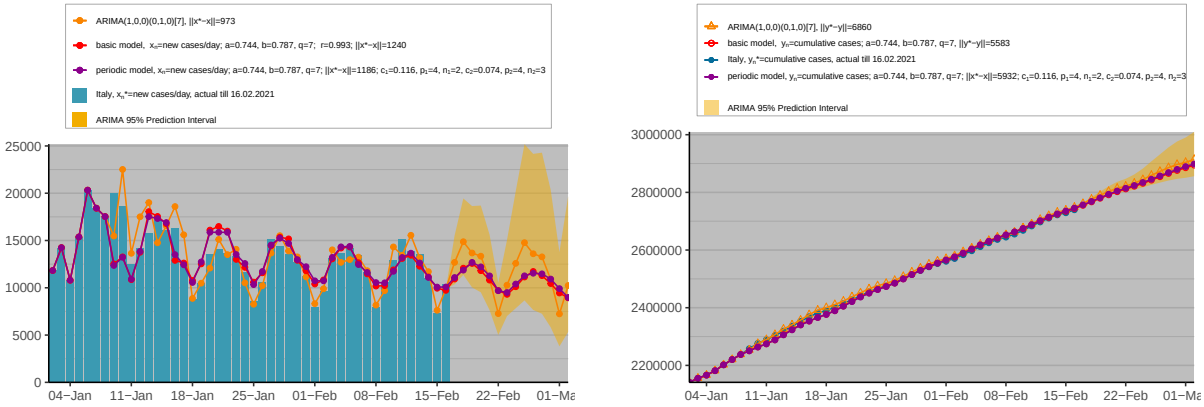



**Figure 37:** ARIMA model, Italy

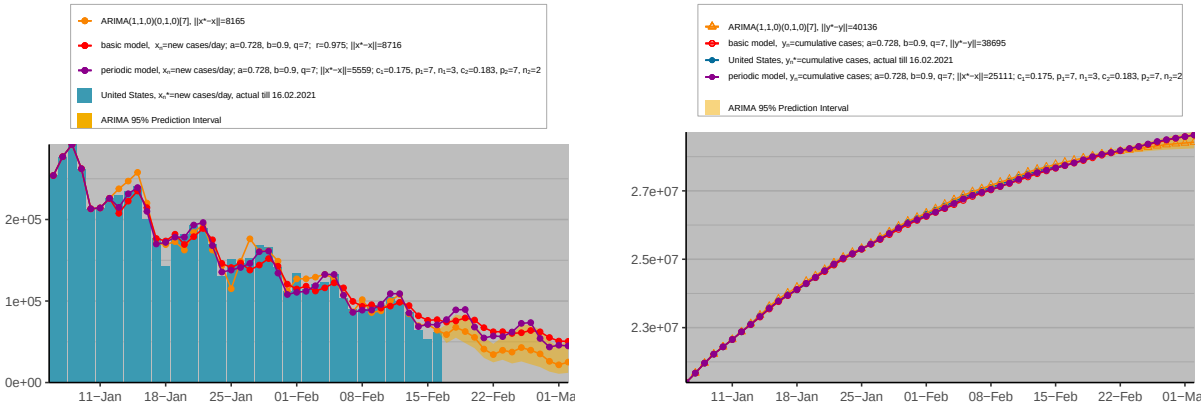



**Figure 38:** ARIMA model, United States

## 3.3 Neural network models

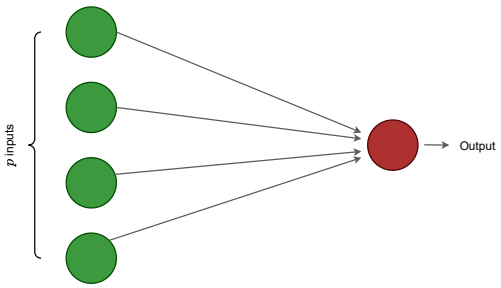



**Figure 39:** A linear regression model, or ARIMA$(p, 0, 0)$ model.

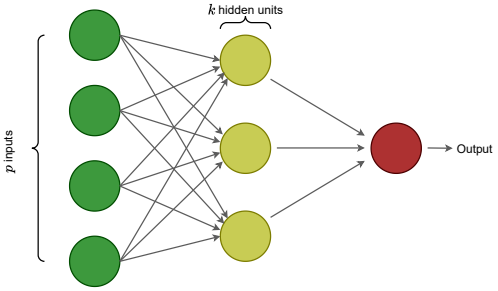



**Figure 40:** A neural network with $p$ inputs and one hidden layer with $k$ hidden neurons.

### 3.3.1 Definitions and Theory

The inputs to the hidden layer are then transformed using a sigmoid function

$$g(z) = \frac{1}{1 + \exp(z)} \tag{16}$$

### 3.3.2 How to select the best model

### 3.3.3 Forecasting

### 3.3.4 Implementation in R

```
#lambda=0 ensures values stay positive
#size is the number of nodes in the hidden layer
nnfit   <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda = 0,
      repeats = 20, maxit = 50)
...
...
...
plots[["nn"]] <- plot_nn(countrydat, modeldat, cols, labs)
```

**Listing 6:** Algorithm for NNAR Model

### 3.3.5 Plots

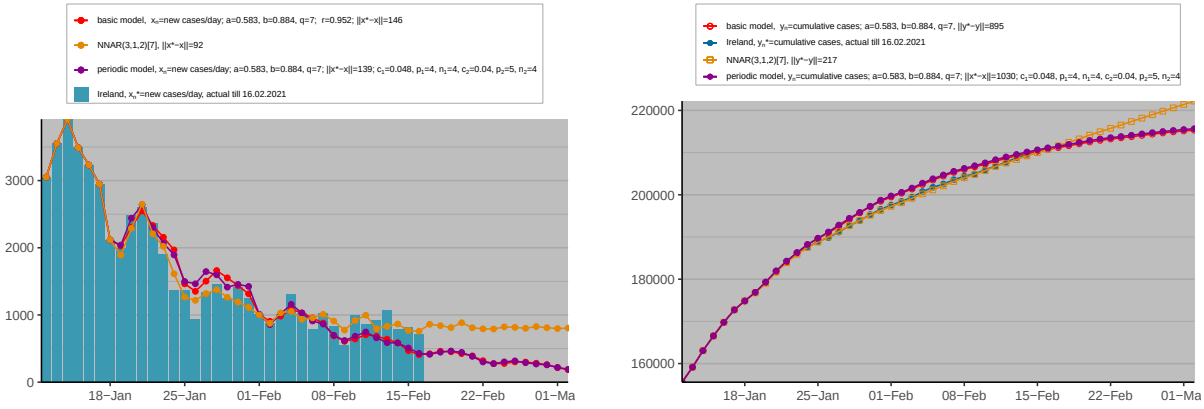



**Figure 41:** Neural Network model, Ireland

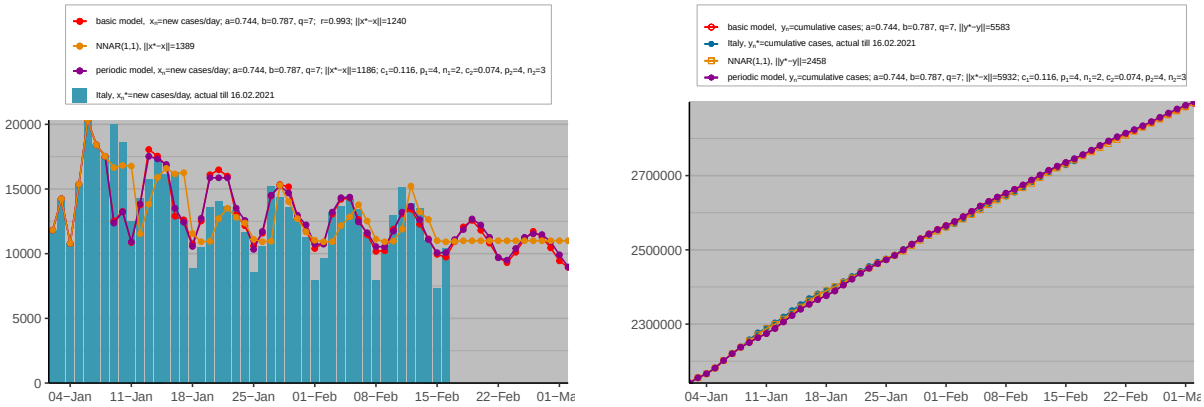



**Figure 42:** Neural Network model, Italy

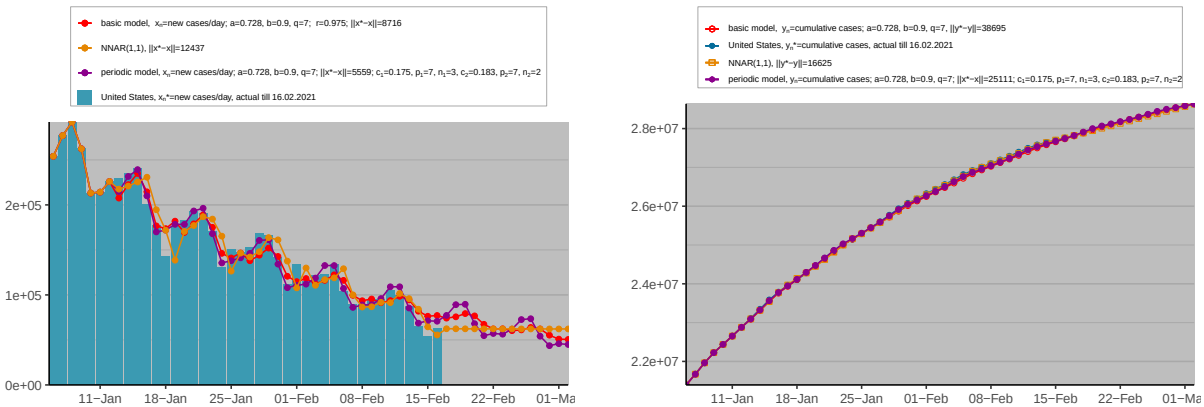



**Figure 43:** Neural Network model, United States

# A Definitions and Theorems for Mathematical Models

## A.1 Recurrence equation

This is our general linear recurrence equation with constant coefficients:

$$x_{n+1} = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \cdots + a_q x_{n-q} \tag{17}$$

The characteristic polynomial of 17

$$f(\lambda) = \lambda^{q+1} - a_0\lambda^q - a_1\lambda^{q-1} - a_2\lambda^{q-2} - \cdots - a_{q-1}\lambda - a_q. \tag{18}$$

**Definition 2.** A root $\lambda$ of $f$ with the maximal absolute value $|\lambda|$ will be referred to as a leading root of the general linear recurrence relation 17.

**Definition 3.** The geormetric mean of $n$ values $x_1, \ldots, x_n$ is

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n} \tag{19}$$

## A.2 Results

**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 18 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 17.

(b) For any positive solution $x_n$ of 17, there exists $C > 0$ such that

$$x_n \sim Cr^n \text{ as } n \to \infty. \tag{20}$$

It follows from 20 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 $\lambda^{q+1}$

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

And so

$$1 = \underbrace{\frac{a_0}{\lambda} + \frac{a_1}{\lambda^2} + \frac{a_2}{\lambda^3} + \cdots + \frac{a_{q-1}}{\lambda^q} + \frac{a_q}{\lambda^{q+1}}}_{g(\lambda)} \tag{21}$$

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 21 (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 $\lambda_k$ are negative or imaginary). Any solution $x_n$ of 17 has the form

$$x_n + Cr^n + \tilde{x}_n \tag{22}$$

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

$$|\tilde{x}_n| = o(r^n) \text{ as } n \to \infty \tag{23}$$

Since $x_n > 0$, it follows from 22 and 23 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 $\rho$ 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

$$X_{n+1} = A_0X_0 + A_1X_{n-1} + \cdots + A_qX_{n-q} \tag{24}$$

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 21, and $g(r) = 1$ that

$$A_0 + A_1 + \cdots + A_q = 1 \tag{25}$$

Set $c := \min(X_1, \ldots, X_{q+1}) > 0$ since $x_n$ have positive initial values. Then we obtain from 24 and 25 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

$$\min\left(1, \left(\frac{a}{b}\right)^{1/m}\right) \leq r \leq \max\left(1, \left(\frac{a}{b}\right)^{1/m}\right) \tag{26}$$

**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 26. The latter is very useful for a basic model as we will see below in an example.

**Proof:**

(a) We have

$$\begin{aligned} f(1) &= 1 - a_0 - a1 - \cdots - a_q \\ &= \underbrace{(1 - a_0)}_{b} - \underbrace{(a_1 + \cdots + a_q)}_{a} \\ &= b - a \end{aligned}$$

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

$$\begin{aligned} 0 < -b r^q + a_m r^{q-m} + \cdots + a_q \\ \implies b r^q &< a_m r^{q-m} + \cdots + a_q \\ &\leq a_m r^{q-m} + \cdots + a_q r^{q-m} \\ &= (a_m + \cdots + a_q) r^{q-m} \\ &= a r^{q-m} \end{aligned}$$

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 $a \leq \max\left(() 1, a\right)$ and $a \geq \min\left(1, a\right)$ to get 26, as required.

□

**Lemma 3.** For the model described by equation 17 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. □

**Result 4.** A good approximation for the leading root $r$ is

$$r \approx= \frac{q\left(\frac{a}{b}\right)^{\frac{1}{2q}} - (q-1)(1-b) + (ab)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \tag{27}$$

**Proof:** Our characteristic polynomial (via 18) for our recurrence equation 3 is

$$f(\lambda) = \lambda^{q+1} - (1-b)\lambda^q - a \tag{28}$$

Let $r$ be its leading root, i.e. $f(r) = 0$
Then by 2, we get

$$\min\left(1, \left(\frac{a}{b}\right)^{1/q}\right) \leq r \leq \max\left(1, \left(\frac{a}{b}\right)^{1/q}\right) \tag{29}$$

since $m = q$ in our polynomial.
Taking the *geometric mean* of the bounds, defined as 19, we get an approximation for $r$

$$r_0 = \left(1 \cdot \left(\frac{a}{b}\right)^{1/q}\right)^{\frac{1}{2}} = \left(\frac{a}{b}\right)^{\frac{1}{2q}} \tag{30}$$

We can then apply Newton's method [16] to find a better approximation for $r$.

**Definition 4.** Let $\{r_n\}_{n\geq 0}$ be a sequence defined by

$$r_{n+1} = r_n - \frac{f(r_n)}{f'(r_n)}, \quad n \geq 0 \tag{31}$$

Then the limit converges to the leading root $r$, i.e.

$$\{r_n\} \to r \quad textasn \to \infty \tag{32}$$

The derivative of our characteristic polynomial 28 is

$$f'(\lambda) = (q+1)\lambda^q - q(1-b)\lambda^{q-1} \tag{33}$$

Then

$$
\begin{aligned}
r_1 &= r_0 - \frac{f(r_0)}{f'(r_0)} \\
&= r_0 - \frac{r_0^{q+1} - (1-b)r_0^q - a}{(q+1)r_0^q - q(1-b)r_0^{q-1}} \\
&= r_0 - \frac{R_0^{q+1} - (1-b)R_0^q - a}{(q+1)R_0^q - q(1-b)R_0^{q-1}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q+1} - (1-b)\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q} - a}{(q+1)\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q} - q(1-b)\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q-1}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\frac{a}{b}\right)^{\frac{q+1}{2q}} - (1-b)\left(\frac{a}{b}\right)^{\frac{1}{2}} - a}{(q+1)\left(\frac{a}{b}\right)^{\frac{1}{2}} - q(1-b)\left(\frac{a}{b}\right)^{\frac{q-1}{2q}}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\frac{a}{b}\right)^{-\frac{1}{2}}}{\left(\frac{a}{b}\right)^{-\frac{1}{2}}} \cdot \frac{\left(\frac{a}{b}\right)^{\frac{q+1}{2q}} - (1-b)\left(\frac{a}{b}\right)^{\frac{1}{2}} - a}{(q+1)\left(\frac{a}{b}\right)^{\frac{1}{2}} - q(1-b)\left(\frac{a}{b}\right)^{\frac{q-1}{2q}}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\frac{a}{b}\right)^{\frac{1}{2q}} - (1-b) - a\left(\frac{a}{b}\right)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\frac{a}{b}\right)^{\frac{1}{2q}} - (1-b) - a\left(\frac{a}{b}\right)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \\
&= \frac{(q+1)\left(\frac{a}{b}\right)^{\frac{1}{2q}} - q(1-b)}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} - \frac{\left(\frac{a}{b}\right)^{\frac{1}{2q}} - (1-b) - a\left(\frac{a}{b}\right)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \\
&= \frac{(q+1)\left(\frac{a}{b}\right)^{\frac{1}{2q}} - q(1-b) - \left(\frac{a}{b}\right)^{\frac{1}{2q}} + (1-b) + a\left(\frac{a}{b}\right)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \\
&= \frac{q\left(\frac{a}{b}\right)^{\frac{1}{2q}} - (q-1)(1-b) + a\left(\frac{b}{a}\right)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \\
&= \frac{q\left(\frac{a}{b}\right)^{\frac{1}{2q}} - (q-1)(1-b) + (ab)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}}
\end{aligned}
$$

□

**Definition 5.** The function $f$ takes on the average value between points $x_1$ and $x_2$, $f_{\text{avg}}$ given by the formula

$$f_{\text{avg}} = \frac{1}{x_2 - x_1}\int_{x_1}^{x_2} f(x)dx \tag{34}$$

**Proposition 5.** The average value of the sequence $a_0, \ldots, a_N$ defined by 7 can be reasonably estimated by our previous parameter $a$.

Similarly, the average value of the sequence $b_0, \ldots, b_N$ defined by 8 can be reasonably estimated by our previous parameter $b$.

**Proof:** Let $a(x)$ be the continuous extension of $a_0, a_1, \ldots, a_N$, i.e.

$$a(x) = a\left(1 + c_1\left(\sin\left(\frac{2\pi}{p_1}(x - n_1)\right)\right)\right), \quad 0 \le x \le N \tag{35}$$

Then, for $x_1 = 0$ and $x_2 = N$

$$
\begin{aligned}
a_{\text{avg}} &= \frac{1}{N-0}\int_0^N a(x)dx \\
&= \frac{1}{N}\int_0^N a\left(1 + c_1\left(\sin\left(\frac{2\pi}{p_1}(x - n_1)\right)\right)\right)dx \\
&= \frac{1}{N}\int_0^N adx + \frac{ac_1}{N}\int_0^N \sin\left(\frac{2\pi}{p_1}(x - n_1)\right)dx
\end{aligned}
$$

Then, using $\sin(A - B) = \sin(A)\cos(B) - \cos(A)\sin(B)$

$$
\begin{aligned}
a_{\text{avg}} &= \frac{1}{N}ax\Big|_0^N + \frac{ac_1}{N}\int_0^N \left(\sin\left(\frac{2\pi x}{p_1}\right)\cos\left(\frac{2\pi n_1}{p_1}\right) - \cos\left(\frac{2\pi x}{p_1}\right)\sin\left(\frac{2\pi n_1}{p_1}\right)\right)dx \\
&= \frac{aN}{N} + \frac{ac_1}{N}\cos\left(\frac{2\pi n_1}{p_1}\right)\int_0^N \sin\left(\frac{2\pi x}{p_1}\right)dx - \frac{ac_1}{N}\sin\left(\frac{2\pi n_1}{p_1}\right)\int_0^N \cos\left(\frac{2\pi x}{p_1}\right)dx \\
&= a + \frac{ac_1}{N}\cos\left(\frac{2\pi n_1}{p_1}\right)\cdot\left(-\frac{p_1}{2\pi}\cos\left(\frac{2\pi x}{p_1}\right)\right)\Big|_0^N - \frac{ac_1}{N}\sin\left(\frac{2\pi n_1}{p_1}\right)\cdot\left(\frac{p_1}{2\pi}\sin\left(\frac{2\pi x}{p_1}\right)\right)\Big|_0^N \\
&= a - \frac{ac_1}{N}\cos\left(\frac{2\pi n_1}{p_1}\right)\frac{p_1}{2\pi}\cos\left(\frac{2\pi N}{p_1}\right) + \frac{ac_1}{N}\cos\left(\frac{2\pi n_1}{p_1}\right)\frac{p_1}{2\pi}\cdot 1 - \frac{ac_1}{N}\sin\left(\frac{2\pi n_1}{p_1}\right)\frac{p_1}{2\pi}\sin\left(\frac{2\pi N}{p_1}\right) + 0 \\
&= a - \frac{ac_1p_1}{2\pi N}\left(\cos\left(\frac{2\pi n_1}{p_1}\right)\cos\left(\frac{2\pi N}{p_1}\right) + \cos\left(\frac{2\pi n_1}{p_1}\right) - \sin\left(\frac{2\pi n_1}{p_1}\right)\sin\left(\frac{2\pi N}{p_1}\right)\right)
\end{aligned}
$$

We use a similar identity $\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$ to get

$$
a_{\text{avg}} = a - \frac{ac_1p_1}{2\pi N}\left(\cos\left(\frac{2\pi\left(N+n_1\right)}{p_1}\right) + \cos\left(\frac{2\pi n_1}{p_1}\right)\right)
$$

Since $\cos(\cdot)$ is bounded between $\pm 1$ and
$c_1p_1 < 0.2 \cdot 8 = 1.6$ and $N > 50$ usually, we see that

$$
\left|\frac{ac_1p_1}{2\pi N}\left(\cos\left(\frac{2\pi\left(N+n_1\right)}{p_1}\right) + \cos\left(\frac{2\pi n_1}{p_1}\right)\right)\right| < \frac{1.6}{300}\cdot(1+1) = \frac{4}{375}
$$

So the average value of $a$ is approximately within $4/375 \approx 1.07\%$ of the value of $a$.

Therefore $a$ is a reasonable estimate for the average $a_{\text{avg}}$ of the periodic parameter $a(x)$.

Similarly, exchanging $c_1, n_1, p_1$ for $c_2, n_2, p_2$ we have a definition for $b(x)$ and hence $b$ is a reasonable estimate for the average $b_{\text{avg}}$.

□

# B Source 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 [10].

## B.1 R packages

`ggplot2` [18] is widely used for easily plotting and visualising the models. `rgdal` [3] allows geospatial `.shp` files to be read into `R`.

`raster` [11] allows this data to be manipulated and plotted.

`dplyr` [19] provides useful data manipulation functions, both for models and geospatial mapping.

Statistical models (HoltWinters, ARIMA and Neural Network Regression) were readily implemented from `forecast` [14].

The majority ofthe colours for plotting are selected using colour oalettes from the `wesanderson` package [15].


**Figure 44:** Wes Anderson Palettes

## B.2 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 [12], while the more detailed county-level shapefile was downloaded from [5].

## B.3 Datasets

Country-based data:

Originally used data from [7], but the ECDC switched from a daily to a weekly update from 14 December 2020. Therefore, I have chosen to use the data from [6], which has remained daily

| 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 45:** OWID World data extract

Ireland cases by county
Downloaded from [4].

| 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 46:** ArcGIS Ireland data extract

## B.4 Source Code

All the code, plots, and even this report, are available at https://github.com/gibbona1/TCD_FinalYearProject. I will include the main code files in text below.

```
modnorm <- function(x,modx) return(floor(sum(abs(x-modx))/length(x)))

xntoyn <- function(xn) return(cumsum(xn))

basicmodx <- function(x, pars, len = 0){
  q <- floor(pars[1])
  a <- pars[2]
  b <- pars[3]
  modx <- x[1:q]
  for(i in (q+1):(length(x)+len)){
    modx[i] <- (1-b)*modx[i-1] + a*modx[i-q]
  }
  return(modx)
}

norm <- function(par, x) return(modnorm(x,basicmodx(x, par)))

normy <- function(par, x, y) return(modnorm(y,xntoyn(basicmodx(x, par))))

normalize <- function(x) return((x-min(x))/(max(x)-min(x)))

basef <- function(lambda,par) {
  return(lambda^(par[1]+1)-(1-par[3])*lambda^(par[1])-par[2])
}

basefprime <- function(lambda,par) {
  return((par[1]+1)*lambda^(par[1])-(1-par[3])*par[1]*lambda^(par[1]-1))
}

normC <- function(par,x,r){
  return(modnorm(x, par*r^(0:(length(x[!is.na(x)])-1))))
}

movingavg <- function(x){
```

```
  mavgx <- (x[1]+x[2])/2
  for(i in 2:(length(x)-1)){
    mavgx[i] <- sum(x[(i-1):(i+1)])/3
  }
  mavgx[length(x)] <- (x[length(x)-1]+x[length(x)])/2
  return(mavgx)
}

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

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)
}
```

**Listing 7:** Model helper functions

```
plot_xn <- function(countrydat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
    geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
    scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+
    scale_y_continuous(expand = c(0,0)) +
    scale_fill_manual(values = cols$xn, name = "", labels = labs$xn) +
    xntheme()
  return(p)
}

plot_yn <- function(countrydat, cols, labs){
  p <- ggplot(countrydat) +
    geom_line(aes(x = date, y = yn, colour = "blue")) +
    geom_point(aes(x = date, y = yn, colour = "blue"))+
    scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+
    scale_y_continuous(expand = c(0,0)) +
    scale_colour_manual(values = cols$yn, name = "", labels = labs$yn) +
    yntheme()
  return(p)
}

plot_basexn <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
    geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
    geom_point(data = modeldat, aes(x = date, y = basexn, colour = cols$basexn)) +
    geom_line(data = modeldat, aes(x = date, y = basexn, colour = cols$basexn)) +
    gg_scale_xy +
    scale_fill_manual(  name = "leg", values = cols$xn,     labels = labs$xn) +
    scale_colour_manual(name = "leg", values = cols$basexn, labels = labs$basexn) +
    xntheme()
  return(p)
}

plot_baseyn <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat) +
    geom_point(aes(x = date, y = yn, colour = "blue")) + geom_line(aes(x = date, y = yn, colour =
          "blue")) +
    geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
    geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
    gg_scale_xy +
    scale_colour_manual(name = "leg",values = c("blue" = cols$yn, "base" = cols$baseyn),
                        labels = c("blue" = labs$yn, "base" = labs$baseyn),
                        guide = guide_legend(override.aes = list(
                          shape = c("blue"=1, "base" = 16)))) +
    yntheme()
  return(p)
}

plot_crn <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
```

```
    geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
    geom_point(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
    geom_line(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
    geom_line(data = modeldat, aes(x = date, y = Crn, colour = "Crn")) +
    gg_scale_xy +
    scale_fill_manual(  name = "leg",values = cols$xn, labels = labs$xn) +
    scale_colour_manual(name = "leg",
                        values = c("basexn" = cols$basexn, "Crn" = cols$Crn),
                        labels = c("basexn" = labs$basexn, "Crn" = labs$Crn),
                        guide = guide_legend(override.aes = list(
                          shape = c("basexn" = 16, "Crn" = NA)))) +
    xntheme()
  return(p)
}

plot_mavgx3 <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
    geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
    geom_point(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
    geom_line(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
    geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
    scale_fill_manual(values = cols$xn, labels = labs$xn) +
    scale_colour_manual(values = c("basexn" = cols$basexn, "x3" = cols$x3),
                        labels = c("basexn" = labs$basexn, "x3" = labs$x3),
                        guide = guide_legend(override.aes = list(
                          shape = c("basexn" = 16, "x3" = NA)))) +
    gg_scale_xy + xntheme()
  return(p)
}

plot_periodic <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat, binwidth = 0) +
    geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
    geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
    geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
    geom_point(data = modeldat,aes(x = date, y = periodic, colour = "periodic")) +
    geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
    gg_scale_xy +
    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
           fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
    scale_fill_manual(values = cols$xn, labels = labs$xn) +
    scale_colour_manual(values = c("base" = cols$basexn, "periodic" = cols$periodic, "x3" = cols$
        x3),
                        labels = c("base" = labs$basexn, "periodic" = labs$periodic, "x3" = labs$
                            x3)) +
    guides(colour = guide_legend(override.aes = list(shape = c("base" = 16, "periodic" = 16, "x3"
         = NA)))) +
    xntheme()
  return(p)
}

plot_periodicy <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat) +
    geom_point(aes(x = date, y = yn, colour = "blue")) +
    geom_line(aes(x = date, y = yn, colour = "blue")) +
    geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
    geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
    geom_point(data = modeldat,aes(x = date, y = periodicy, colour = "periodic")) +
    geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
    gg_scale_xy +
    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE)) +
    scale_colour_manual(values = c("base" = cols$baseyn, "blue" = cols$yn, "periodic" = cols$
        periodic),
                        labels = c("base" = labs$baseyn, "blue" = labs$yn, "periodic" = labs$
                            periodicy),
                        guide = guide_legend(override.aes = list(
                          shape = c("base" = 1, "blue"=16, "periodic"=16)))) +
    yntheme()
  return(p)
}

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"), shape = 5) +
    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 = 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(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)
}

plot_hwy <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat) +
    geom_point(aes(x = date, y = yn, colour = "blue")) +
    geom_line(aes(x = date, y = yn, colour = "blue")) +
    geom_ribbon(data = modeldat, aes(x = date, ymin = hwylo, ymax = hwyhi, fill = "pi"), alpha = 0.5) +
    geom_point(data = modeldat, aes(x = date, y = hwyn, colour = "hw"), shape = 5) +
    geom_line(data = modeldat, aes(x = date, y = hwyn, colour = "hw")) +
    geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
    geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
    geom_point(data = modeldat,aes(x = date, y = periodicy, colour = "periodic")) +
    geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
    gg_scale_xy +
    guides(colour=guide_legend(ncol=1,nrow=4,byrow=TRUE),
           fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
    scale_fill_manual(values = c("pi" = cols$hwpi),
                      labels = c("pi" = labs$hwpi)) +
    scale_colour_manual(values = c("base" = cols$baseyn, "blue" = cols$yn, "hw" = cols$hw, "periodic" = cols$periodic),
                        labels = c("base" = labs$baseyn, "blue" = labs$yn, "hw" = labs$hwy, "periodic" = labs$periodicy),
                        guide = guide_legend(override.aes = list(
                          shape = c("base" = 1, "blue"=16, "hw" = 5, "periodic"=16)))) +
    yntheme()
  return(p)
}

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)
}

```

```
plot_arimay <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat) +
    geom_point(aes(x = date, y = yn, colour = "blue")) +
    geom_line(aes(x = date, y = yn, colour = "blue")) +
    geom_ribbon(data = modeldat, aes(x = date, ymin = arimaylo, ymax = arimayhi, fill = "pi"),
        alpha = 0.5) +
    geom_point(data = modeldat, aes(x = date, y = arimayn, colour = "arima"), shape = 2) +
    geom_line(data = modeldat, aes(x = date, y = arimayn, colour = "arima")) +
    geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
    geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
    geom_point(data = modeldat,aes(x = date, y = periodicy, colour = "periodic")) +
    geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
    gg_scale_xy +
    scale_fill_manual(values = c("pi" = cols$arimapi),
                      labels = c("pi" = labs$arimapi)) +
    scale_colour_manual(values = c("arima" = cols$arima, "base" = cols$baseyn, "blue" = cols$yn,
        "periodic" = cols$periodic),
                        labels = c("arima" = labs$arimay, "base" = labs$baseyn, "blue" = labs$yn,
                             "periodic" = labs$periodicy),
                        guide = guide_legend(override.aes = list(
                          shape = c("arima" = 2, "base" = 1, "blue"=16, "periodic" = 16)))) +
    yntheme()
  return(p)
}

plot_hwarima <- 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 = arimaxn, colour = "arima")) +
    geom_line(data = modeldat, aes(x = date, y = arimaxn, colour = "arima")) +
    geom_point(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
    geom_line(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
    gg_scale_xy +
    guides(colour=guide_legend(ncol=1,nrow=2,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("arima" = cols$arima, "hw" = cols$hw),
                        labels = c("arima" = labs$arima, "hw" = labs$hw)) +
    xntheme()
  return(p)
}

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)
}

plot_nny <- function(countrydat, modeldat, cols, labs){
  p <- ggplot(countrydat) +
    geom_point(aes(x = date, y = yn, colour = "blue")) +
    geom_line(aes(x = date, y = yn, colour = "blue")) +
    geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
    geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
    geom_point(data = modeldat, aes(x = date, y = nnyn, colour = "nn"), shape = 0) +
    geom_line(data = modeldat, aes(x = date, y = nnyn, colour = "nn")) +
    geom_point(data = modeldat,aes(x = date, y = periodicy, colour = "periodic")) +
```

```
253     geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
254     gg_scale_xy +
255     scale_colour_manual(values = c("base" = cols$baseyn, "blue" = cols$yn, "nn" = cols$nn, "
            periodic" = cols$periodic),
256                         labels = c("base" = labs$baseyn, "blue" = labs$yn, "nn" = labs$nny, "
                                periodic" = labs$periodicy),
257                         guide = guide_legend(override.aes = list(
258                           shape = c("base" = 1, "blue"=16, "nn" = 0, "periodic" = 16)))) +
259     yntheme()
260   return(p)
261 }
262
263 plot_multixn <- function(countrydat, modeldat, cols, labs){
264   p <- ggplot(countrydat, binwidth = 0) +
265     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
266     geom_point(data = modeldat, aes(x = date, y = multixn, colour = cols$multixn)) +
267     geom_line(data = modeldat, aes(x = date, y = multixn, colour = cols$multixn)) +
268     gg_scale_xy +
269     scale_fill_manual(  name = "leg",values = cols$xn, labels = labs$xn) +
270     scale_colour_manual(name = "leg",values = cols$multixn, labels = labs$multixn) +
271     xntheme()
272   return(p)
273 }
274
275 plot_multiyn <- function(countrydat, modeldat, cols, labs){
276   p <- ggplot(countrydat) +
277     geom_point(aes(x = date, y = yn, colour = "blue")) +
278     geom_line(aes(x = date, y = yn, colour = "blue")) +
279     geom_point(data = modeldat, aes(x = date, y = multiyn, colour = "base"), shape = 1) +
280     geom_line(data = modeldat, aes(x = date, y = multiyn, colour = "base")) +
281     gg_scale_xy +
282     scale_colour_manual(name = "leg",values = c("blue" = cols$yn, "base" = cols$multiyn),
283                         labels = c("blue" = labs$yn, "base" = labs$multiyn),
284                         guide = guide_legend(override.aes = list(
285                           shape = c("blue"=1, "base" = 16)))) +
286     yntheme()
287   return(p)
288 }
289
290 plot_multiperxn <- function(countrydat, modeldat, cols, labs){
291   p <- ggplot(countrydat, binwidth = 0) +
292     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
293     geom_point(data = modeldat, aes(x = date, y = multipxn, colour = "multi")) +
294     geom_line(data = modeldat, aes(x = date, y = multipxn, colour = "multi")) +
295     geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
296     gg_scale_xy +
297     guides(colour=guide_legend(ncol=1,nrow=2,byrow=TRUE),
298            fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
299     scale_fill_manual(values = cols$xn, labels = labs$xn) +
300     scale_colour_manual(values = c("multi" = cols$multip, "x3" = cols$x3),
301                         labels = c("multi" = labs$multipxn, "x3" = labs$x3)) +
302     guides(colour = guide_legend(override.aes = list(shape = c("multi" = 16, "x3" = NA)))) +
303     xntheme()
304   return(p)
305 }
306
307 plot_multiperyn <- function(countrydat, modeldat, cols, labs){
308   p <- ggplot(countrydat) +
309     geom_point(aes(x = date, y = yn, colour = "blue"), shape = 16) +
310     geom_line(aes(x = date, y = yn, colour = "blue")) +
311     geom_point(data = modeldat, aes(x = date, y = multipyn, colour = "mult"), shape = 1) +
312     geom_line(data = modeldat, aes(x = date, y = multipyn, colour = "mult")) +
313     gg_scale_xy +
314     scale_colour_manual(name = "leg",values = c("blue" = cols$yn, "mult" = cols$multip),
315                         labels = c("blue" = labs$yn, "mult" = labs$multipyn),
316                         guide = guide_legend(override.aes = list(
317                           shape = c("blue"=16, "mult" = 1)))) +
318     yntheme()
319   return(p)
320 }
321
322 plot_worldtotal <- function(dat){
323   p <- ggplot(dat, binwidth = 0) +
324     geom_bar(aes(x = date, y = new_cases),  fill = wes_palettes$Zissou1[1], stat = "identity") +
```

```
    scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+
    scale_y_continuous(expand = c(0,0)) +
    ggtitle(wt_title) + xntheme()
  return(p)
}

xntheme <- function(){
  p <- theme(axis.text.x        = element_text(vjust = 0.5),
             axis.title         = element_blank(),
             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.title       = element_blank(),
             legend.margin      = margin(0,4,0,4,"pt"),
             legend.background  = element_blank(),
             legend.text.align  = 0,
             legend.box.background  = element_rect(linetype="solid", colour ="darkgrey", size = 0
                 .1, fill = "white"),
             legend.spacing     = unit(0, "cm"),
             legend.key.size    = unit(0.8,"line"),
             legend.key         = element_blank(),
             legend.text        = element_text(size = 6),
             legend.direction   = "vertical",
             legend.box         = "vertical",
             legend.box.just    ='left',
             legend.position    = "top")
  return(p)
}

yntheme <- function(){
  p <- theme(axis.text.x        = element_text(vjust = 0.5),
             axis.title         = element_blank(),
             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.title       = element_blank(),
             legend.margin      = margin(4,0,0,4,"pt"),
             legend.background  = element_blank(),
             legend.key         = element_blank(),
             legend.box.background  = element_rect(linetype="solid", colour ="darkgrey", size = 0
                 .1, fill = "white"),
             legend.spacing     = unit(0, "cm"),
             legend.key.size    = unit(0.8,"line"),
             legend.text        = element_text(size = 6),
             legend.direction   = "vertical",
             legend.box         = "vertical",
             legend.box.just    ='left',
             legend.position    = "top")
  return(p)
}

gg_scale_xy <- list(
  scale_x_date(date_breaks = "1 week", date_labels = "%d-%b", expand = c(0,0)),
  scale_y_continuous(expand = c(0,0)))

error_plot <- function(){
  p <- ggplot(data.frame(x = 0,y = 0)) +
    geom_label(x = 0, y = 0, color = "red", size = 5 , fontface = "bold",
               label = "Error: Country data has nonpositive values") +
    xlim(-1,1) + ylim(-1,1) +
    theme(axis.line  = element_blank(),
          axis.text  = element_blank(),
          axis.ticks = element_blank(),
          panel.grid = element_blank())
  return(p)
}
```

**Listing 8:** Plotting functions

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

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],]
  latest_date <- countrydat$date[nrow(countrydat)]

  countrydesc <- paste0(country, ", ", dateBounds[1], " to ", dateBounds[2])

  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 = 150)
  bseq    <- seq(from = 0.1, to = 0.9, length.out = 150)
  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")

```

```
#newnormdat$abnormy <- apply(newnormdat, 1, function(x) normy(x, countrydat$xn, countrydat$yn))

tileoptim <- normdat[which.min(normdat$abnorm),1:3]
optimpars <- c(tileoptim$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)
xnorm       <- norm(optimpars,countrydat$xn)
labs$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||="*.(xnorm)))

ynorm       <- 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||="*.(ynorm)))

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

summarydf <- data.frame(model = "Basic Recursion", xnorm = xnorm, ynorm = ynorm)

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$mavgx3  <- movingavg(mavgx1)
countrydat$mavgx3y <- xntoyn(countrydat$mavgx3)+ prevcases

x3norm  <- modnorm(countrydat$xn,countrydat$mavgx3)
x3normy <- modnorm(countrydat$yn,countrydat$mavgx3y)
labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*.(x3norm)))

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

summarydf <- rbind(summarydf, data.frame(model = "Moving Average", xnorm = x3norm, ynorm = x3
    normy))

#parameters of the form (ci,pi,ni)
##an = a(1 + c1 sin(2pi/p1 (n - n1)))
c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 20)
n_1seq  <- n_2seq <- 1:q
p_1seq  <- p_2seq <- 1:q
normdatp <- expand.grid(a  = a, b  = b,
                        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]*"=cumulative cases;"
                                ~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)

  summarydf <- rbind(summarydf, data.frame(model = "Periodic", xnorm = optpernorm, ynorm =
       optpernormy))

  #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(log(dat_ts), add.normal = TRUE, bins=10)

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

  if(any(countrydat$xn <= 0)){
    plots[["hw"]] <- plots[["hwy"]] <- error_plot()
  } else{
    hwmethod <- "additive"
    #lambda=0 ensures values stay positive
    hwfcst   <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)

    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)
    hwnormy   <- modnorm(countrydat$yn,modeldat$hwyn[1:nrow(countrydat)])

    labs$hw   <- paste0("HoltWinters algorithm,  ||x*-x||=", hwnorm)
    labs$hwy  <- paste0("HoltWinters algorithm,  ||y*-y||=", hwnormy)
    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], "]"))
  }

  summarydf <- rbind(summarydf, data.frame(model = "HoltWinters", xnorm = hwnorm, ynorm = hwnormy
      ))

  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

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

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

  summarydf <- rbind(summarydf, data.frame(model = "ARIMA", xnorm = arimanorm, ynorm = arimanormy
      ))

  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, PI=TRUE, h = forecastlen)

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

  modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
  modeldat$nnlo <- c(nn.fcst$fitted, nn.fcst$lower[,2])
  modeldat$nnhi <- c(nn.fcst$fitted, nn.fcst$upper[,2])

  modeldat$nnyn  <- xntoyn(modeldat$nnxn) + prevcases
  modeldat$nnylo <- xntoyn(modeldat$nnlo) + prevcases
  modeldat$nnyhi <- xntoyn(modeldat$nnhi) + prevcases

  nnnorm  <- modnorm(countrydat$xn,nn.fcst$fitted)
  nnnormy <- modnorm(countrydat$yn,modeldat$nnyn[1:nrow(countrydat)])
```

```
287
288   labs$nn   <- paste0(nnfit$method, ", ||x*-x||=", nnnorm)
289   labs$nny  <- paste0(nnfit$method, ", ||y*-y||=", nnnormy)
290   labs$nnpi <- "NNAR 95% Prediction Interval"
291
292   plots[["nn"]]  <- plot_nn(countrydat, modeldat, cols, labs)
293   plots[["nny"]] <- plot_nny(countrydat, modeldat, cols, labs)
294
295   summarydf <- rbind(summarydf, data.frame(model = "Neural Network", xnorm = nnnorm, ynorm =
          nnnormy))
296   summarydf$xnorm <- as.integer(summarydf$xnorm)
297   summarydf$ynorm <- as.integer(summarydf$ynorm)
298
299   colnames(summarydf) <- c("model", "$||x-x^*||$", "$||y-y^*||$")
300   return(list("plots" = plots, "summary" = summarydf, desc = countrydesc))
301 }
302
303 grigorDates <- c("2020-04-26", "2020-06-09")
304 datebounds <- list(
305   "Italy"         = c("2021-01-02", "2021-02-16"),
306   "United States" = c("2021-01-06", "2021-02-16"),
307   "Ireland"       = c("2021-01-12", "2021-02-16"),
308   "Germany"       = c("2021-01-06", "2021-02-16")
309   #"Netherlands"   = c("2021-01-06", "2021-02-16"),
310   #"Spain"         = c("2021-01-06", "2021-02-16"),
311   #"United Kingdom"= c("2021-01-06", "2021-02-16")
312 )
313
314 owiddat     <- owiddat[!is.na(owiddat$new_cases),]
315 totaldat    <- owiddat[owiddat$location == "World",]
316 latest_date <- totaldat$date[nrow(totaldat)]
317 wt_title <- sprintf('Global Total =%s as at %s',
318                     format(sum(totaldat$new_cases), big.mark=",", scientific=FALSE),
319                     format.Date(latest_date, "%B %d, %Y"))
320
321 plotslist[["WorldTotal"]][["xn"]] <- plot_worldtotal(totaldat)
322
323 for(country in names(datebounds)){
324   plotslist[[country]] <- covidPlots(country, datebounds[[country]], owiddat)
325 }
```

**Listing 9:** Main algorithm

```
1  source("Code/covid-plotutils.R")
2  source("Code/covid-modelutils.R")
3
4  owiddat      <- read.csv("Data/owid-covid-data.csv")
5  owiddat$date <- as.Date(owiddat$date, tryFormats = c("%Y-%m-%d"))
6  multilist <- list()
7
8  multidates <- list(
9    #"Italy"         = list(c("2020-12-16", "2021-01-01"),
10   #                       c("2021-01-02", "2021-01-30")),
11   #"United States" = list(c("2020-11-16", "2021-01-06"),
12   #                       c("2021-01-07", "2021-01-30")),
13   #"Ireland"       = list(c("2020-12-16", "2021-01-07"),
14   #                       c("2021-01-08", "2021-01-30"))
15   "Ireland"       = list(c("2020-02-29", "2020-04-15"),
16                          c("2020-04-16", "2020-06-09"))
17 )
18
19 multiphasePlots <- function(country, dates, data){
20   plots <- list()
21   crows <- grep(country, data$location)
22   countrydat <- data.frame(date = data$date[crows],
23                            xn = data$new_cases[crows], yn = data$total_cases[crows])
24   if(nrow(countrydat[countrydat$date < dates[[1]][1],]) == 0)
25     beforecumcases <- 0
26   else
27     beforecumcases <- sum(countrydat$xn[countrydat$date < dates[[1]][1]])
28
29   countrydat  <- countrydat[countrydat$date >= dates[[1]][1],]
30   countrydat  <- countrydat[countrydat$date <= dates[[length(dates)]][2],]
31   latest_date <- countrydat$date[nrow(countrydat)]
```

```
32
33  forecastlen <- 5
34
35  multimodx <- function(x, multix, pars, oldp = rep(1,10), start=FALSE, len = 0){
36    q <- floor(pars[1])
37    a <- pars[2]
38    b <- pars[3]
39    fitstd <- length(multix)+1
40
41    if(start){
42      multix[fitstd:(fitstd+q-1)] <- x[1:q]
43      for(i in (fitstd+q):(fitstd+length(x)+len-1)){
44        multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
45      }
46    } else {
47      for(i in (fitstd):(fitstd+length(x)+len-1)){
48        multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
49      }
50    }
51    return(multix)
52  }
53
54  multimodxper <- function(par, q=7, x, multix, oldp = rep(1,10), start=FALSE, len = 0){
55    #a,b,c1,c2,p1,p2,n1,n2
56    #first day of this phase
57    fitstd <- length(multix)+1
58    an  <- par[1]*(1+par[3]*sin(2*pi*(1:(fitstd+length(x)+len-1) - par[7])/par[5]))
59    bn  <- par[2]*(1+par[4]*sin(2*pi*(1:(fitstd+length(x)+len-1) - par[8])/par[6]))
60
61    if(start){
62      multix[fitstd:(fitstd+q-1)] <- x[1:q]
63      for(i in (fitstd+q):(fitstd+length(x)+len-1)){
64        multix[i] <- (bn[i]*(1-bn[i-1]))*multix[i-1]/bn[i-1] +
65          (an[i-q]*bn[i])*multix[i-q]/bn[i-q]
66      }
67    } else {
68      for(i in fitstd:(fitstd+length(x)+len-1)){
69        multix[i] <- (bn[i]*(1-bn[i-1]))*multix[i-1]/bn[i-1] +
70          (an[i-q]*bn[i])*multix[i-q]/bn[i-q]
71      }
72    }
73    return(multix)
74  }
75
76  #Specific dates
77  multimodel  <- c()
78  multimodelp <- c()
79  phasepars   <- list()
80  for(i in 1:length(dates)){
81    phase    <- dates[[i]]
82    phasedat <- countrydat[countrydat$date >= phase[1] & countrydat$date <= phase[2],]
83
84    #get basic model for each phase first in order to get
85    # starting a and b to guess for periodic an and bn
86
87    aseq    <- seq(from = 0.1, to = 2.5, length.out = 50)
88    bseq    <- seq(from = 0.1, to = 0.9, length.out = 50)
89    qseq    <- 6:8
90    normdat <- expand.grid(q = qseq, a = aseq, b = bseq)
91
92    if(i == 1)
93      abnorm <- apply(normdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, start
          =TRUE), phasedat$xn))
94    else
95      abnorm <- apply(normdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, oldp
          = phasepars[[i-1]])[length(multimodel) + 1:nrow(phasedat)], phasedat$xn))
96
97    normalize <- function(x){
98      return((x-min(x))/(max(x)-min(x)))
99    }
100   normdat$abnorm  <- abnorm
101
102   tileoptim <- normdat[which.min(normdat$abnorm),1:3]
103   optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
```

```
104
105     q <- optimpars[1]
106     a <- optimpars[2]
107     b <- optimpars[3]
108
109     phasepars[[i]] <- round(optimpars,3)
110
111     if(i == 1 & i != length(dates))
112       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE)
113     if(i == 1 & i == length(dates))
114       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE, len=forecastlen)
115     if(i > 1 & i == length(dates))
116       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]], len=
              forecastlen)
117     if(length(dates) > 2 & i %in% 2:(length(dates)-1))
118       multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]])
119
120     aseqper <- a*seq(from = 0.7, to = 1.3, length.out = 10)
121     bseqper <- b*seq(from = 0.7, to = 1.3, length.out = 10)
122     c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 10)
123     n_1seq  <- n_2seq <- c(1,7)
124     p_1seq  <- p_2seq <- 6:7
125     normdatp <- expand.grid(a  = aseqper, b  = bseqper,
126                             c1 = c_1seq,  c2 = c_2seq,
127                             p1 = p_1seq,  p2 = p_2seq,
128                             n1 = n_1seq,  n2 = n_2seq)
129
130     if(i == 1)
131       pernorm <- apply(normdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1],
              phasedat$xn, multimodelp, start=TRUE), phasedat$xn))
132     else
133       pernorm <- apply(normdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1],
              phasedat$xn, multimodelp)[length(multimodelp)+1:nrow(phasedat)], phasedat$xn))
134
135     normdatp$pernorm <- pernorm
136
137     newnormdatp <- normdatp %>% top_n(pernorm, n = -25)
138
139     if(i == 1)
140       pernormy <- apply(newnormdatp, 1, function(par) modnorm(beforecumcases+xntoyn(multimodxper(
              par, q = optimpars[1], phasedat$xn, multimodelp, start=TRUE)), phasedat$yn))
141     else
142       pernormy <- apply(newnormdatp, 1, function(par) modnorm(beforecumcases+xntoyn(multimodxper(
              par, q = optimpars[1], phasedat$xn, multimodelp))[length(multimodelp)+1:nrow(phasedat)
              ], phasedat$yn))
143
144     newnormdatp$pernormy <- pernormy
145     newnormdatp$combnorm <- normalize(newnormdatp$pernorm) + normalize(newnormdatp$pernormy)
146
147     peroptim <- as.numeric(newnormdatp[which.min(newnormdatp$combnorm),1:8])
148
149     phasepars[[i]] <- c(phasepars[[i]],round(peroptim,3))
150
151     if(i == 1 & i != length(dates))
152       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE)
153     if(i == 1 & i == length(dates))
154       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE, len=
              forecastlen)
155     if(i > 1 & i == length(dates))
156       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, len=forecastlen)
157     if(length(dates) > 2 & i %in% 2:(length(dates)-1))
158       multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp)
159   }
160
161   cols <- list(
162     xn       = wes_palettes$Zissou1[1],
163     yn       = wes_palettes$Darjeeling2[2],
164     multixn  = wes_palettes$Darjeeling1[1],
165     multiyn  = wes_palettes$Darjeeling1[1],
166     multip   = "magenta4",
167     x3       = wes_palettes$FantasticFox1[2]
168   )
169
170   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"))))
  )

  multimodnormval  <- modnorm(multimodel[1:length(countrydat$xn)], countrydat$xn)
  multimodnormyval <- modnorm(beforecumcases+xntoyn(multimodel[1:length(countrydat$xn)]), countrydat$yn)

  b.roman <- function(x){ return(paste0("(", as.roman(x), ")"))}

  multilabd <- c()
  for(i in 1:length(dates)){
    multilabd <- c(multilabd, paste(b.roman(i), "from", format.Date(as.Date(as.character(dates[[i]][1])),"%d.%m")))
  }
  multilabd <- paste0(multilabd, collapse = "; ")

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

  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)))

  modeldat <- data.frame(date     = c(countrydat$date,as.Date(latest_date) + 1:forecastlen),
                         multixn  = multimodel,
                         multiyn  = beforecumcases + xntoyn(multimodel),
                         multipxn = multimodelp,
                         multipyn = beforecumcases + xntoyn(multimodelp))

  plots[["xn"]] <- plot_multixn(countrydat, modeldat, cols, labs)

  plots[["yn"]] <- plot_multiyn(countrydat, modeldat, cols, labs)

  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 10:** 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 11:** 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 12:** 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]][["plots"]])){
    ggsave(filename = paste0(country, "-", p, ".pdf"),
           plot     = plotslist[[country]][["plots"]][[p]],
           path     = "./Plots",
           height   = 10,
           width    = 14,
           units    = "cm"
           )
  }
  if(!is.null(plotslist[[country]][["summary"]])){
    sumtable <- xtable(plotslist[[country]][["summary"]], type = "latex",
                       caption = plotslist[[country]][["desc"]],
                       label = paste0("fig:", country, "summarydf"))
    print(sumtable, sanitize.text.function=function(x){x},
          table.placement="H",
          file = paste0("Report/Chapters/", country, "-summarydf.tex"))
  }
}

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)){
  ggsave(filename = paste0("world-", p, ".pdf"),
         plot     = worldplot[[p]],
         path     = "Plots",
         height   = 14,
         width    = 20,
         units    = "cm"
  )
}
```

**Listing 13:** 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] 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.

[4] 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.

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

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

[7] 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/.

[8] 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.

[9] 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.

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

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

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

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

[14] Rob J Hyndman and Yeasmin Khandakar. *Automatic time series forecasting: the forecast package for R*. 3. 2008, pp. 1–22. URL: https://www.jstatsoft.org/article/view/v027i03.

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

[16] Springer Verlag GmbH, European Mathematical Society. *Encyclopedia of Mathematics, Newton method*. Website. Accessed on 2021-03-19.

[17] 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.

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

[19] 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.