

Modelling the Spread of COVID-19 Cases

TCD Final Year Project

Anthony Gibbons
Project Supervisor: Athanasios Georgiadis

March 25, 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. We apply the implement both new and existing algorithms in R.

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

1	Introduction	7
1.1	Previous work on Covid-19 identification and modeling	10
1.2	Key aims	10
2	Mathematical Models	11
2.1	Base model	11
2.1.1	Model Assumptions	11
2.1.2	Notation	11
2.1.3	How to select the best model	12
2.1.4	Forecasting	12
2.1.5	Implementation in R	12
2.1.6	Plots	12
2.2	Limiting curve	13
2.3	Moving average	14
2.4	Periodic model	14
2.4.1	Implementation in R	15
2.4.2	Plots	16
2.5	Multi-phase model	16
2.5.1	Plots	17
3	Statistical Models	20
3.1	Holt-Winters' seasonal method	20
3.1.1	Definitions and Theory	20
3.1.2	Implementation in R	20
3.1.3	Plots	21
3.2	ARIMA models	22
3.2.1	Definitions and Theory	22
3.2.2	Implementation in R	24
3.2.3	Plots	24
3.3	Neural network models	25
3.3.1	Implementation in R	25
3.3.2	Plots	26
4	Model performance with Train/Test sets	27
4.1	Summary	27
4.2	Base Model	28
4.3	Periodic model	28
4.4	HoltWinters model	28
4.5	ARIMA model	29
4.6	NNAR model	29
A	Definitions and Theorems for Mathematical Models	30
A.1	Recurrence equation	30
A.2	Results	30
B	Source Code and Data Sources	37
B.1	R packages	37
B.2	Shapefiles	37
B.3	Datasets	37
B.4	Source Code	38

List of Figures

1	Daily Cases Globally	7
2	Cases per 1 million population, World	8
3	Cases per 1 million population, Europe	8
4	Situation by County, Ireland	9
5	Individual Comparison of Ireland with various countries, daily cases per million of the population	9
6	Comparison of Ireland with various countries, daily cases per million of the population	10
7	Contours, Ireland	12
8	Contours, Italy	12
9	Contours, United States	12
10	Basic model, Ireland	12
11	Basic model, Italy	13
12	Basic model, United States	13
13	Limiting curve Cr^n , Ireland	13
14	Limiting curve Cr^n , Italy	13
15	Limiting curve Cr^n , United States	13
16	Limiting Curve Growing Exponentially, Ireland	14
17	Moving average $x_n^*(3)$, Ireland	14
18	Moving average $x_n^*(3)$, Italy	14
19	Moving average $x_n^*(3)$, United States	14
20	Oscillating a and b parameters, Ireland, Italy and United States	15
21	Periodic model, Ireland	16
22	Periodic model, Italy	16
23	Periodic model, United States	16
24	Multi-phase model, Ireland	17
25	Multi-phase model, Italy	17
26	Multi-phase model, United States	18
27	Multi-phase periodic model, Ireland	18
28	Multi-phase periodic model, Italy	18
29	Multi-phase periodic model, United States	19
30	Normality checks, Ireland, Italy and United States	20
31	Seasonal Holt Winter's Additive Model Algorithm (denoted SHW ₊)	20
32	Comparison of HoltWinters multiplicative (left) and additive (right) algorithms	21
33	HoltWinters model, Ireland	21
34	HoltWinters model, Italy	21
35	HoltWinters model, United States	22
36	ARIMA model, Ireland	24
37	ARIMA model, Italy	24
38	ARIMA model, United States	25
39	A linear regression model, or ARIMA($p, 0, 0$) model.	25
40	A neural network with p inputs and one hidden layer with k hidden neurons.	25
41	Neural Network Autoregression model, Ireland	26
42	Neural Network Autoregression model, Italy	26
43	Neural Network Autoregression model, United States	26
44	Basic model with train/test split, Ireland	28
45	Periodic model with train/test split, Ireland	28
46	HoltWinters model with train/test split, Ireland	28
47	ARIMA model with train/test split, Ireland	29
48	Neural Network Autoregression model with train/test split, Ireland	29
49	Wes Anderson Palettes	37
50	OWID World data extract	38
51	ArcGIS Ireland data extract	38

List of Code

1	Algorithm for Base Model	12
2	Algorithm for Limiting Curve	13
3	Algorithm for Periodic Model	15
4	Algorithm for HoltWinters Model	20
5	Algorithm for ARIMA Model	24
6	Algorithm for NNAR Model	25
7	Model helper functions	38
8	Plotting functions	39
9	Main algorithm	45
10	multi-phase models	50
11	Geospatial/Map plots	53
12	Country Comparison plots	55
13	Saving plots	56

List of Tables

1	Ireland, 2021-01-12 to 2021-02-16	27
2	Italy, 2021-01-02 to 2021-02-16	27
3	United States, 2021-01-06 to 2021-02-16	27
4	Ireland: mathematical models parameters	27
5	Italy: mathematical models parameters	27
6	United States: mathematical models parameters	27

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.

Global Total =111,285,971 as at February 23, 2021

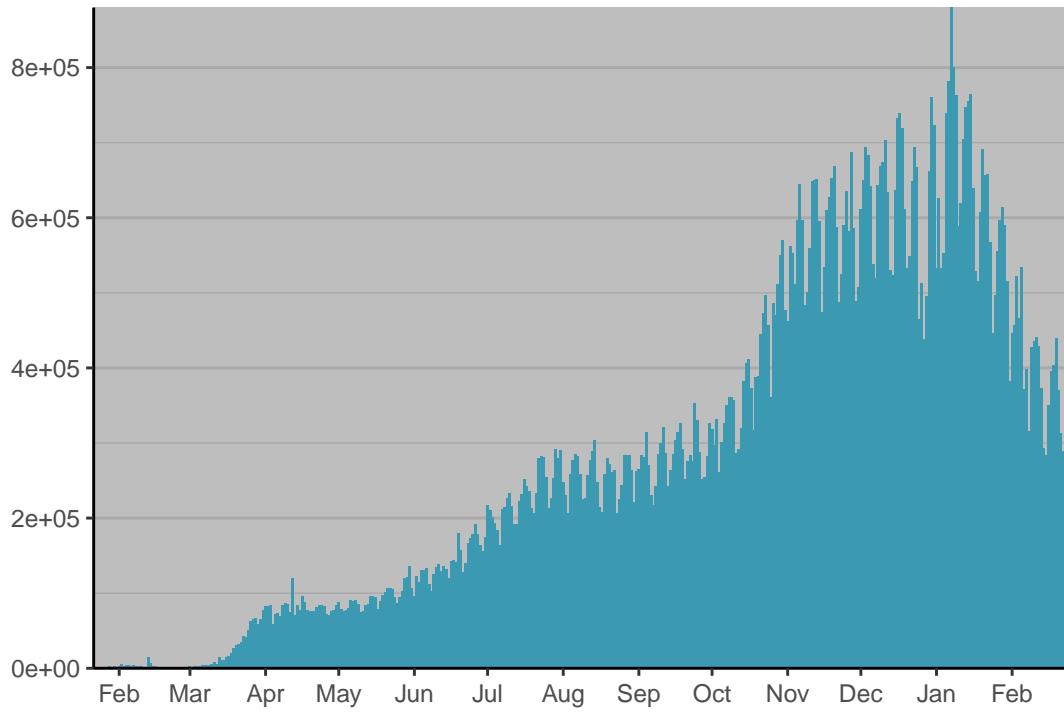


Figure 1: Daily Cases Globally

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

Cases per 1 million population by country
From February 09, 2021 to February 22, 2021

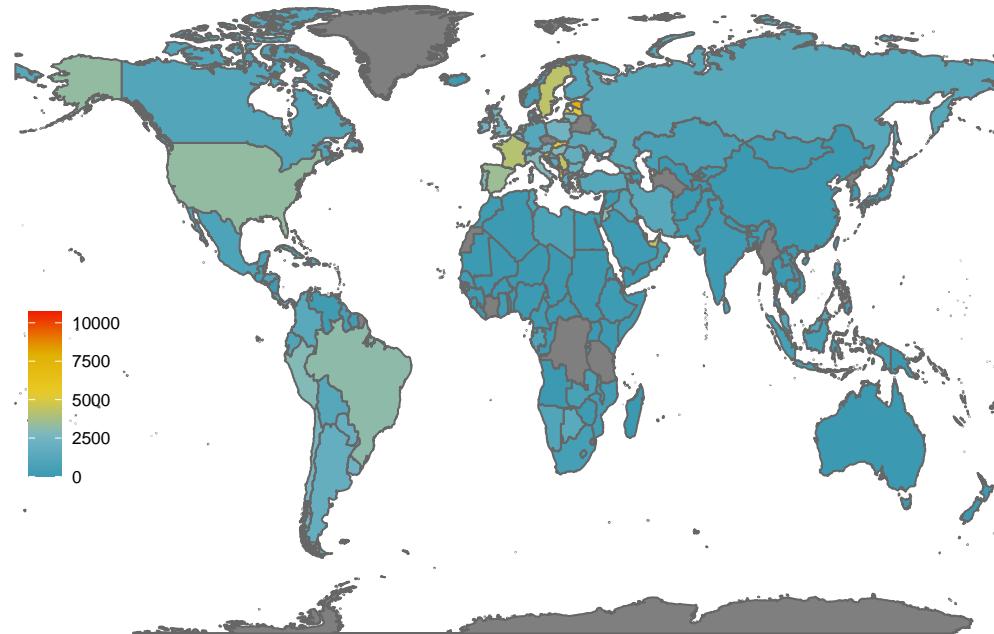


Figure 2: Cases per 1 million population, World

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

Total cases per 1 million population by country
From February 09, 2021 to February 22, 2021

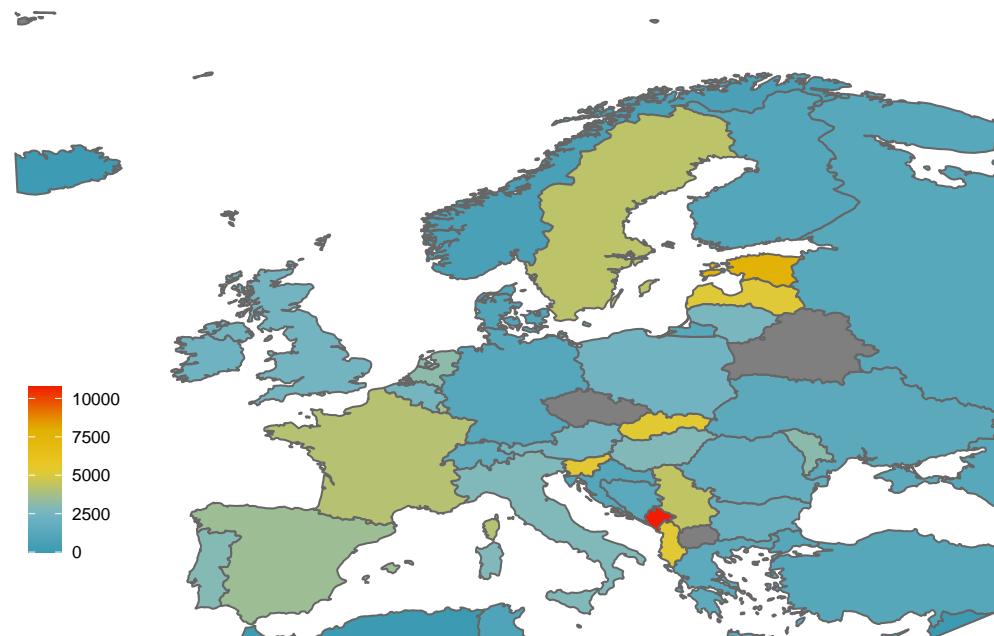


Figure 3: Cases per 1 million population, Europe

More locally, we see that Ireland also has a clear variation in concentration in 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.

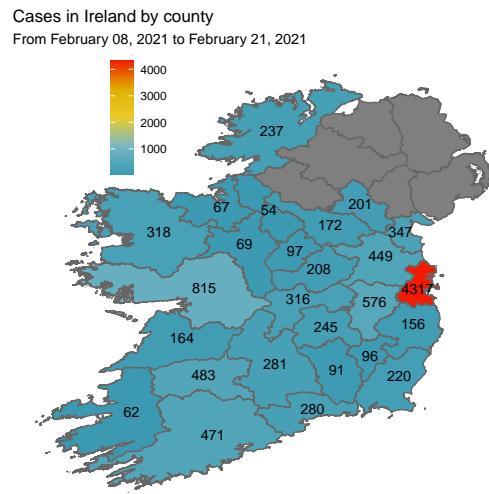
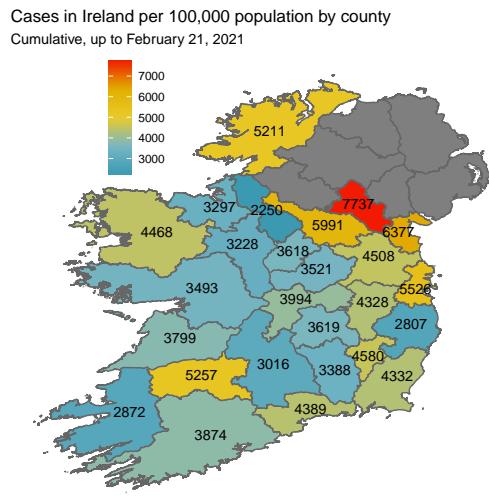


Figure 4: Situation by County, Ireland

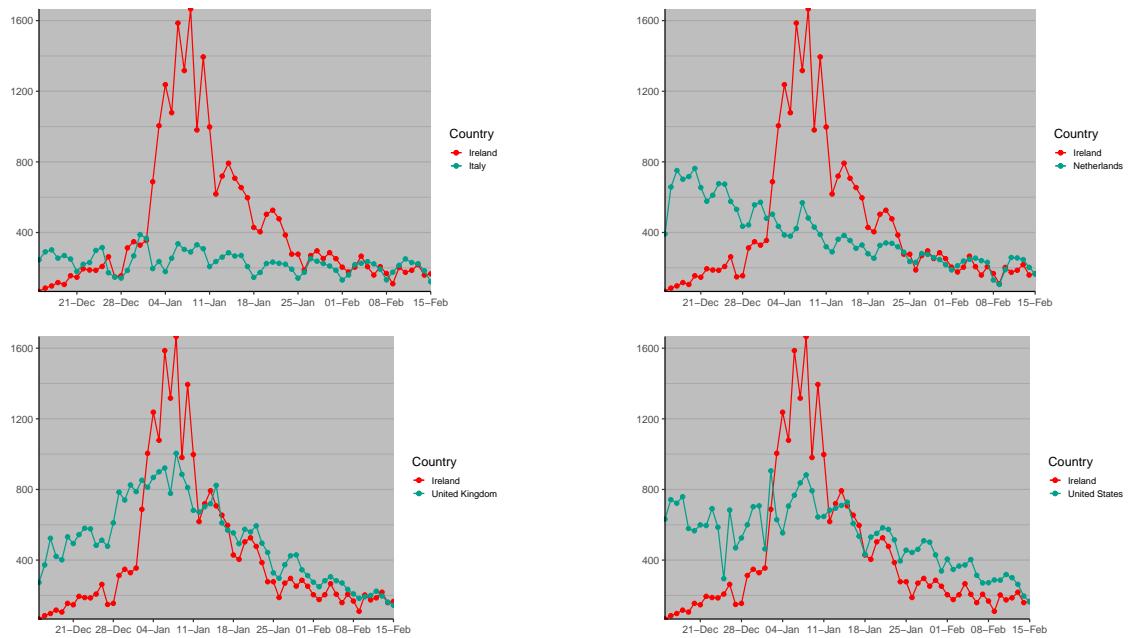


Figure 5: Individual Comparison of Ireland with various countries, daily cases per million of the population

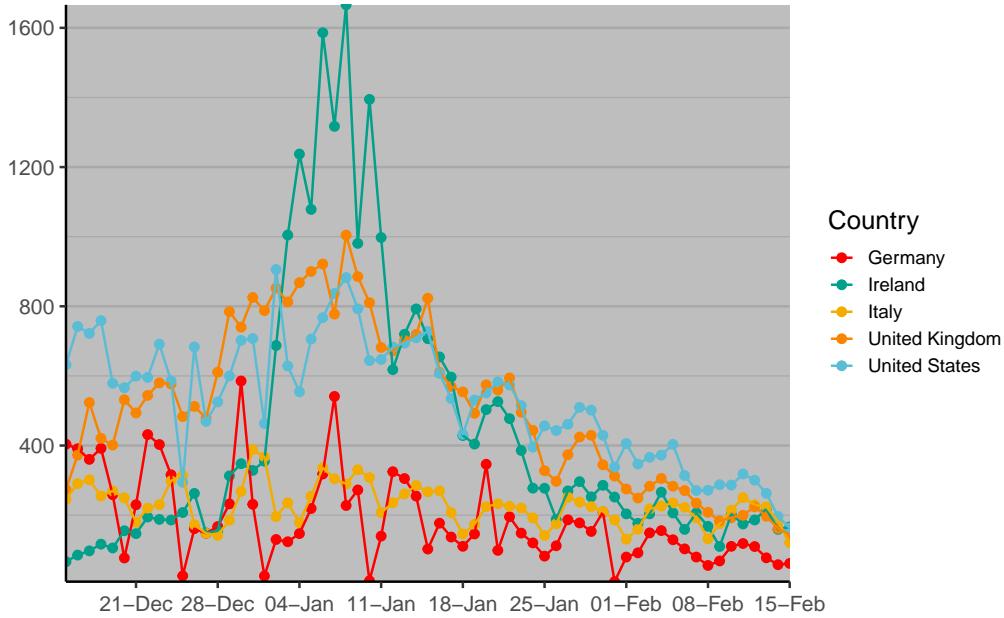


Figure 6: Comparison of Ireland with various countries, daily cases per million of the population

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 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.

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:

$$r \approx R_0^{\frac{1}{2q}}. \quad (1)$$

Recurrence relation for z_n :

The number of ill people at large on day $n + 1$ is the number at large on day n , minus the number who were detected and isolated on day n , plus the number of people who were infected q days ago, i.e.

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

Using $x_n = bz_n$ and $u_{n-q} = ax_{n-q}$ we obtain the following equation for x_n :

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

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

$$x_n = x_n^* \text{ for } n = 0, 1, \dots, q, \quad (4)$$

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

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

Similarly we define $\|y - y^*\|$

In order to determine values a, b, q , we ideally want to minimize both

$$\|x - x^*\| \text{ and } \|y - y^*\| \quad (6)$$

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 .

¹The 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

We primarily seek to minimize 5, which can be seen using contour plots below.

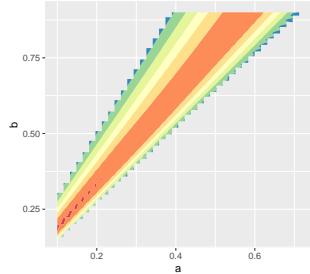


Figure 7: Contours, Ireland

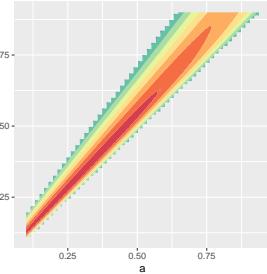


Figure 8: Contours, Italy

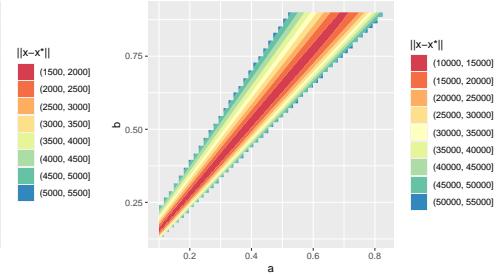


Figure 9: Contours, United States

The `optim()` function in R appeared to have a danger of converging to local optima, so with some trade-off in computation time over accuracy, I have chosen to iterate the algorithm `basicmodx()` over all combinations of a, b, q .

2.1.4 Forecasting

The model can easily be extended m days ahead, with x_{N+1}, \dots, x_{N+m} defined in 3.

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

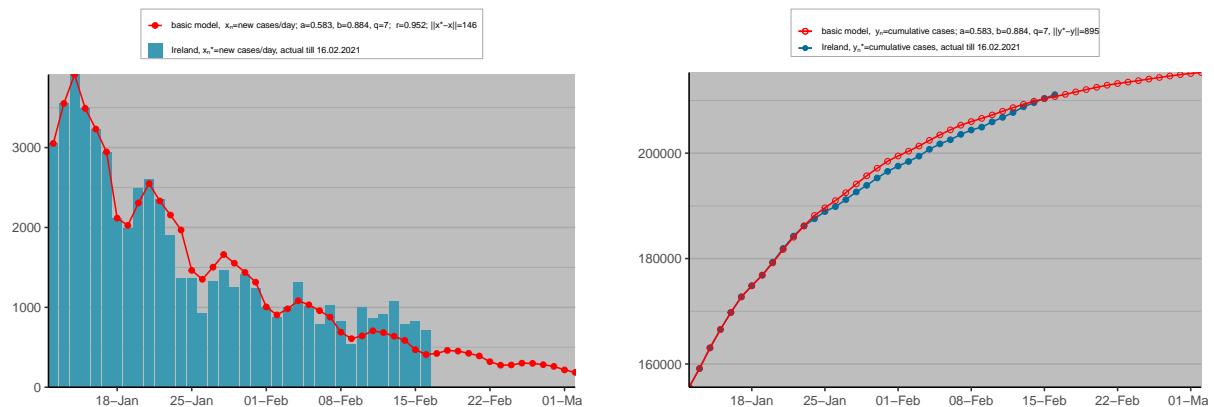


Figure 10: Basic model, Ireland

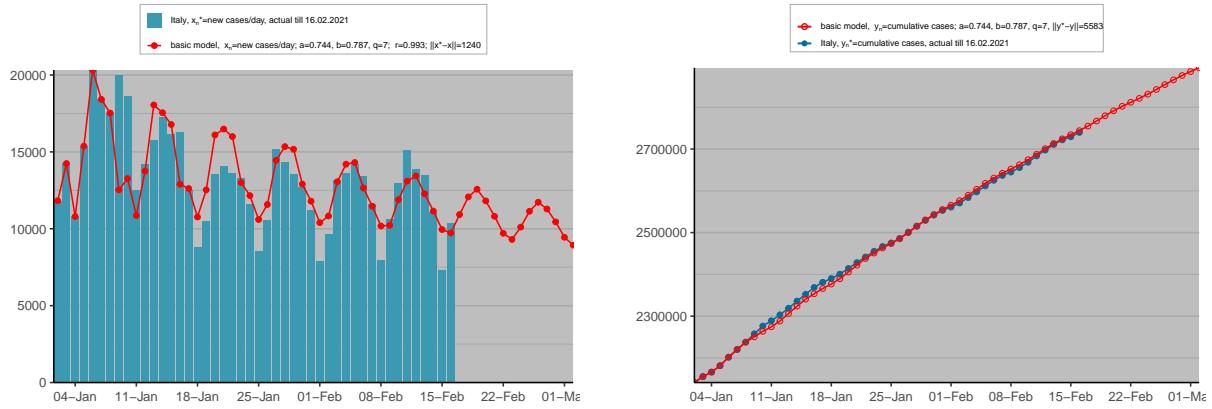


Figure 11: Basic model, Italy

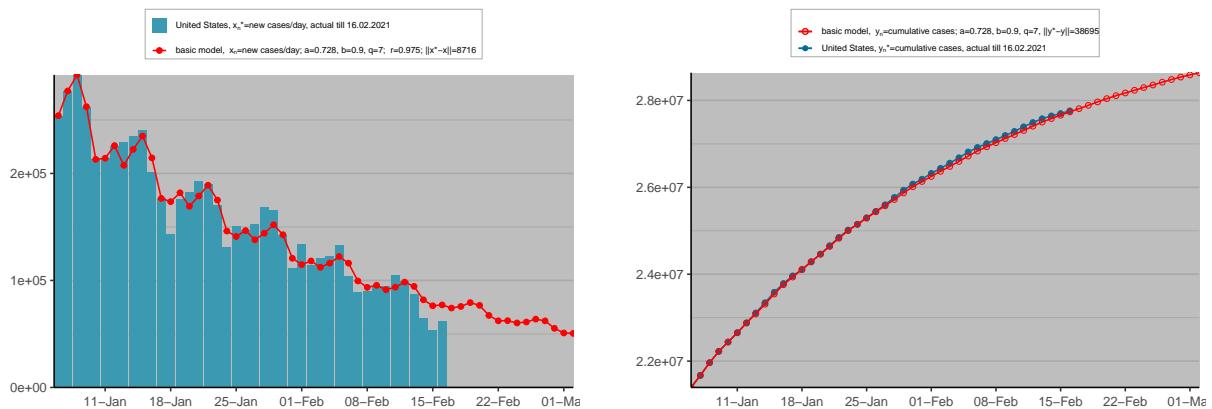


Figure 12: Basic model, United States

2.2 Limiting curve

The second result in 1 is the equation 20, which defines the limiting behavior of the recurrence relation 3.

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

Listing 2: Algorithm for Limiting Curve

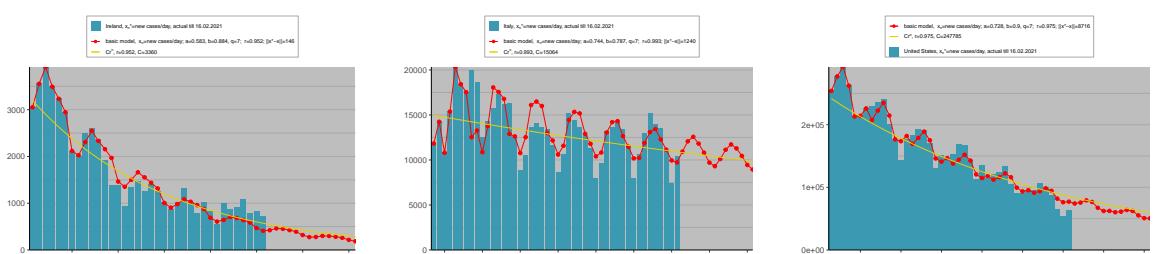


Figure 13: Limiting curve Cr^n , **Figure 14:** Limiting curve Cr^n , **Figure 15:** Limiting curve Cr^n ,
Ireland Italy United States

The limiting curve can also show exponential growth and quickly get out of control

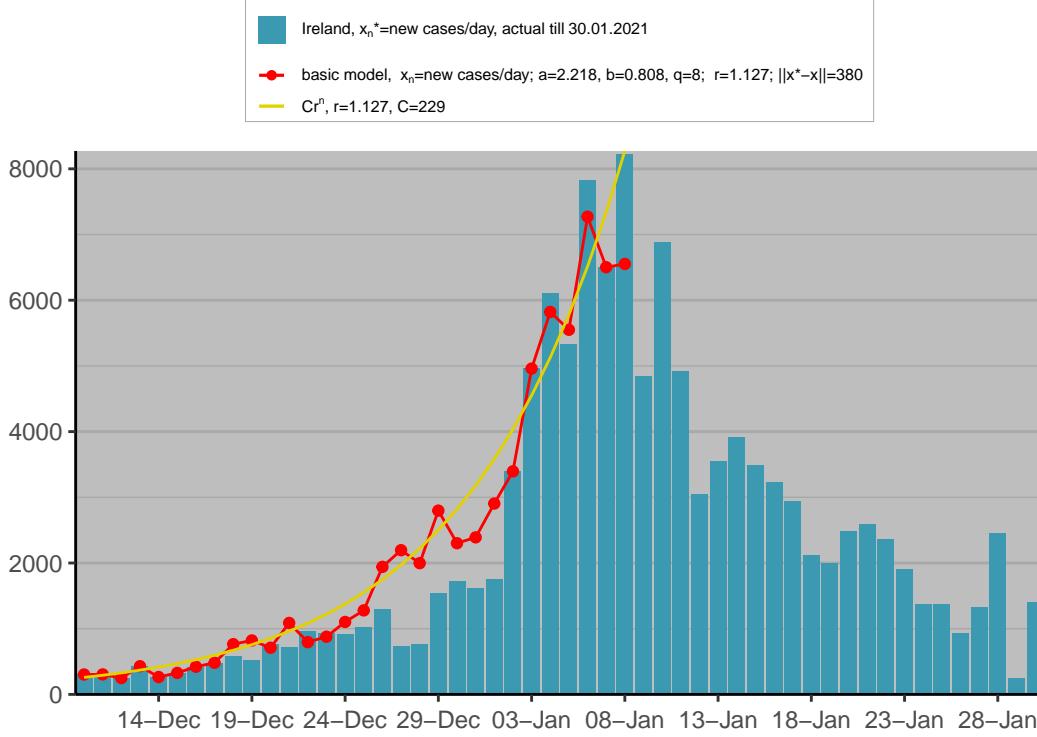


Figure 16: Limiting Curve Growing Exponentially, Ireland

2.3 Moving average

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

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

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

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

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.

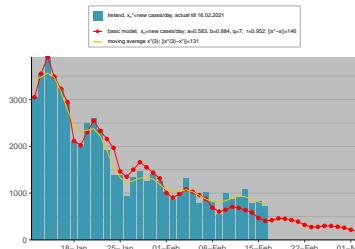


Figure 17: Moving average $x_n^*(3)$, **Ireland**

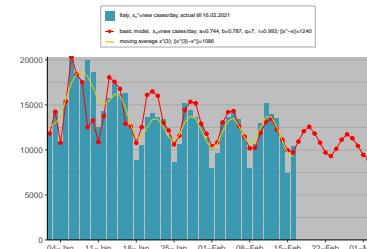


Figure 18: Moving average $x_n^*(3)$, **Italy**

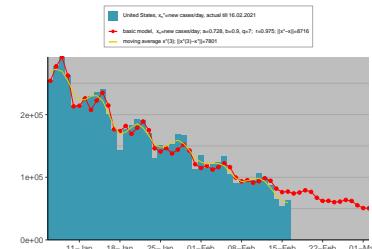


Figure 19: Moving average $x_n^*(3)$, **United States**

2.4 Periodic model

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

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

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



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

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

$c_i \in [0.04, 0.2]$ small are amplitudes,

$n_i \in 1, 2, \dots, q$ are lags and

$p_i \in 1, 2, \dots, q$ 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 to calculate the a_n and b_n , since the average behavior should be close to the same.

I included a sketch of the proof in the appendix 6.

This means that we can keep the original parameters a, b, q (we assume the latency period is constant), and we are instead optimizing over the 6 parameters $c_1, n_1, p_1, c_2, n_2, p_2$

This enables the algorithm to speed up by at least an order of magnitude.

2.4.1 Implementation in R

```

1 modxper <- function(par, q, x, len = 0){
2   #a,b,c1,c2,p1,p2,n1,n2
3   an <- par[1]*(1+par[3]*sin(2*pi*(1:(length(x)+len) - par[7])/par[5]))
4   bn <- par[2]*(1+par[4]*sin(2*pi*(1:(length(x)+len) - par[8])/par[6]))
5   modx <- x[1:q]
6   for(i in (q+1):(length(x)+len)){
7     modx[i] <- (bn[i]*(1-bn[i-1]))*modx[i-1]/bn[i-1] +
8       (an[i-q]*bn[i])*modx[i-q]/bn[i-q]
9   }
10  return(modx)
11 }
12 modeldat$periodic <- modxper(as.numeric(perooptim),countrydat$xn, q = q, forecastlen)
...
...
...
16 plots[["periodic"]] <- plot_periodic(countrydat, modeldat, cols, labs)

```

Listing 3: Algorithm for Periodic Model

2.4.2 Plots

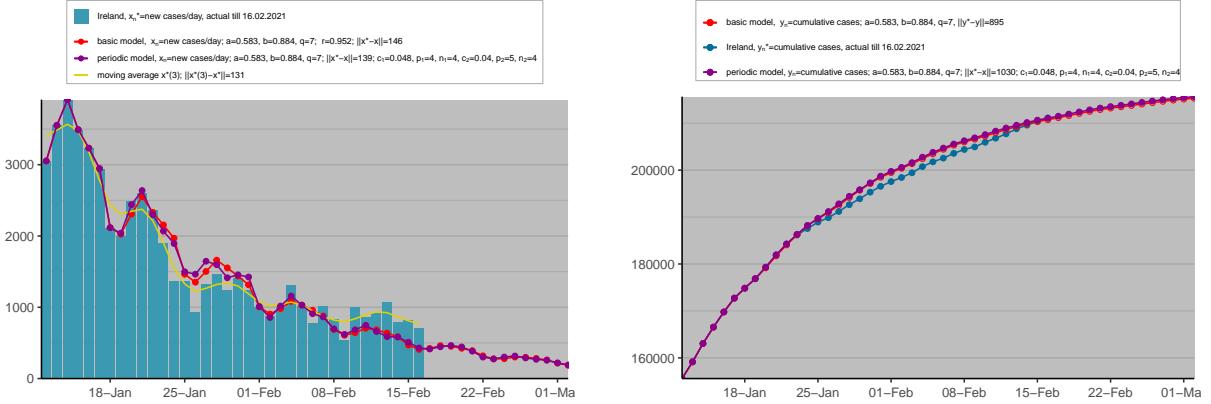


Figure 21: Periodic model, Ireland

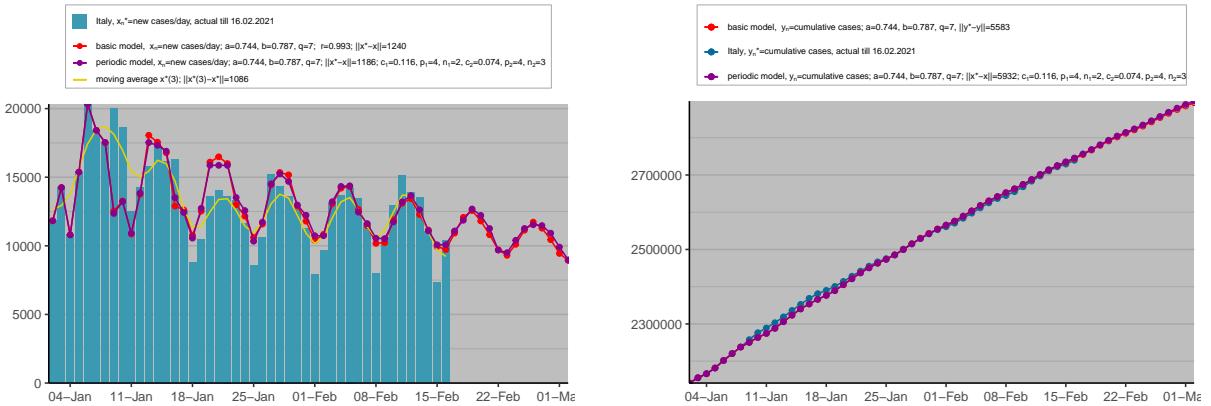


Figure 22: Periodic model, Italy

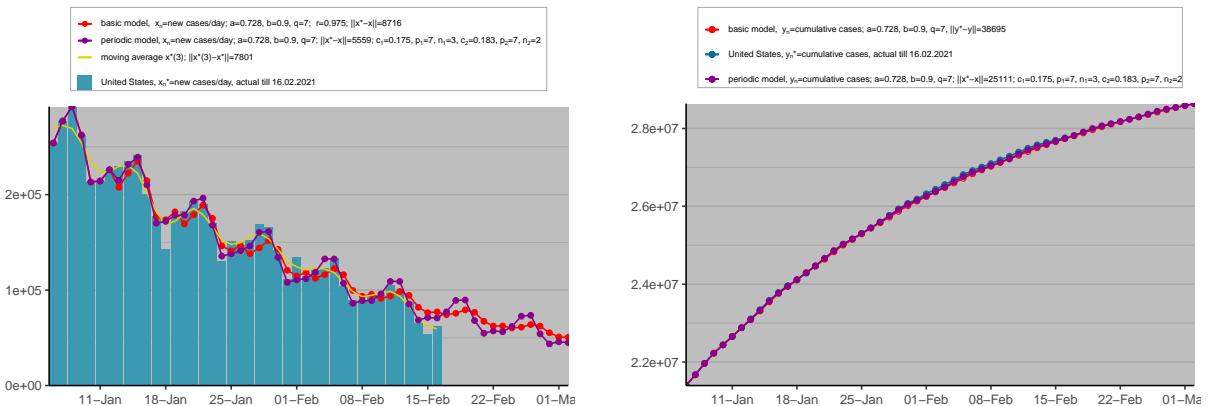


Figure 23: Periodic model, United States

2.5 Multi-phase model

While Grigorian's paper only formulated a two-phase model, I have generalized it to any number of phases, both in the equations below and the code.

Suppose there are M phases of the pandemic, and we wish to compute the parameters a, b, q separately for each phase.

Therefore we have phases

- (1) parameters a_1, b_1, q_1 and model x_n for $0 \leq n \leq N_1$
- (2) parameters a_2, b_2, q_2 and model x_n for $N_1 + 1 \leq n \leq N_2$
- \vdots
- (M) parameters a_M, b_M, q_M and model x_n for $N_{M-1} + 1 \leq n \leq N_M = N$

Grigorian's paper uses a form of smoothing of parameter b , which is using $\frac{b_{\text{new}}}{b_{\text{old}}}$ for a few days around the change point to deal with the transition between phase.

This perturbation is already dealt with sufficiently in the periodic version so I have excluded this in the event that $\frac{b_{\text{new}}}{b_{\text{old}}}$ is much larger than 1.

The model parameters are computed separately for each phase, with the values x_n feeding into the next phase to continue the recursive definition (e.g. phase 2 needs some values from phase 1 of the model).

The model parameters are chosen to minimize $\sum |y_i - \hat{y}_i|^p$ for each phase, and the overall model performance displayed is computed for the model as a whole against the data as a whole.

2.5.1 Plots

The multi-phase model without periodicity can be sharp and unrealistic, and is purely for demonstration

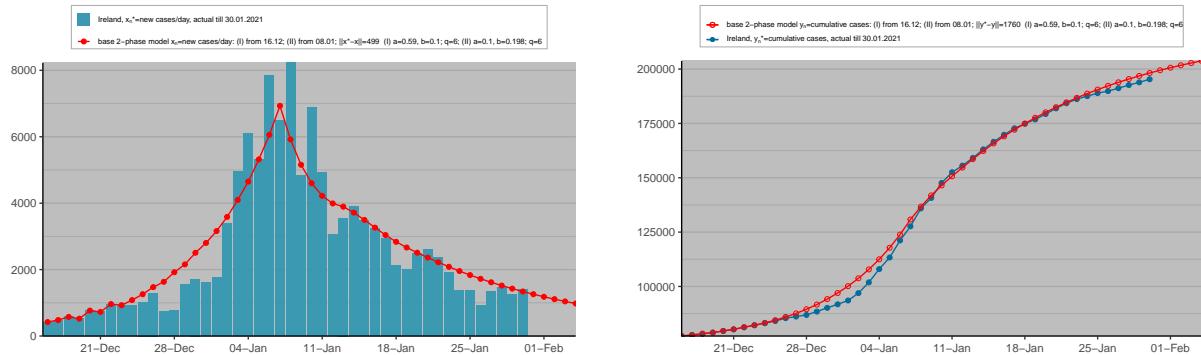


Figure 24: Multi-phase model, Ireland

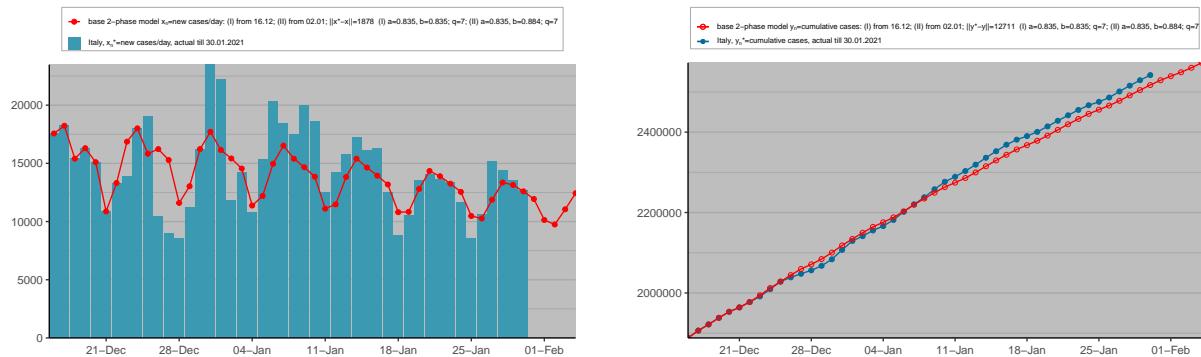


Figure 25: Multi-phase model, Italy

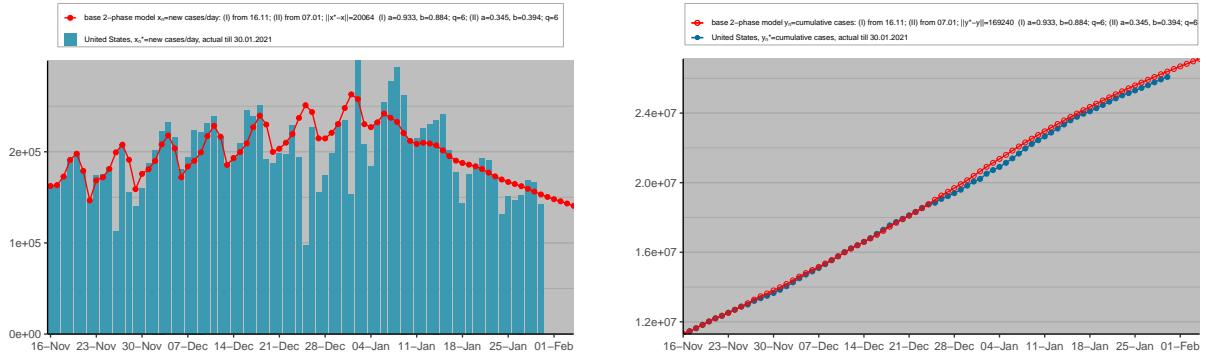


Figure 26: Multi-phase model, United States

The periodic model is a similar extension as in 2.4 and often performs better.

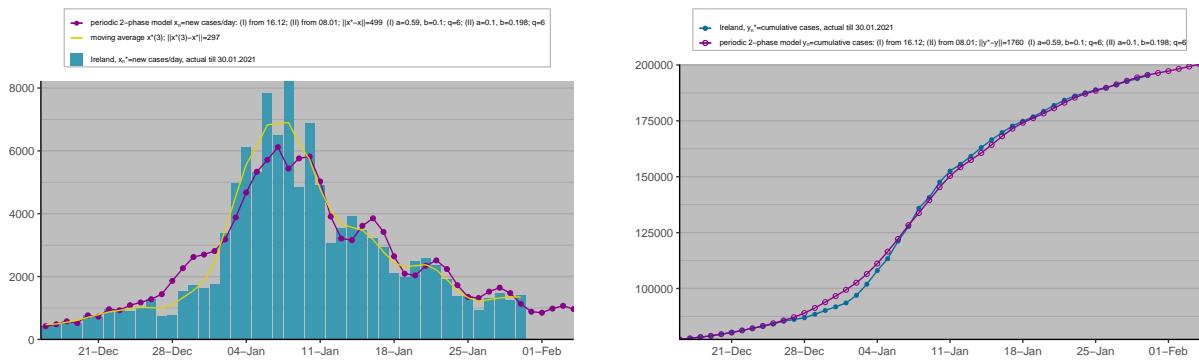


Figure 27: Multi-phase periodic model, Ireland

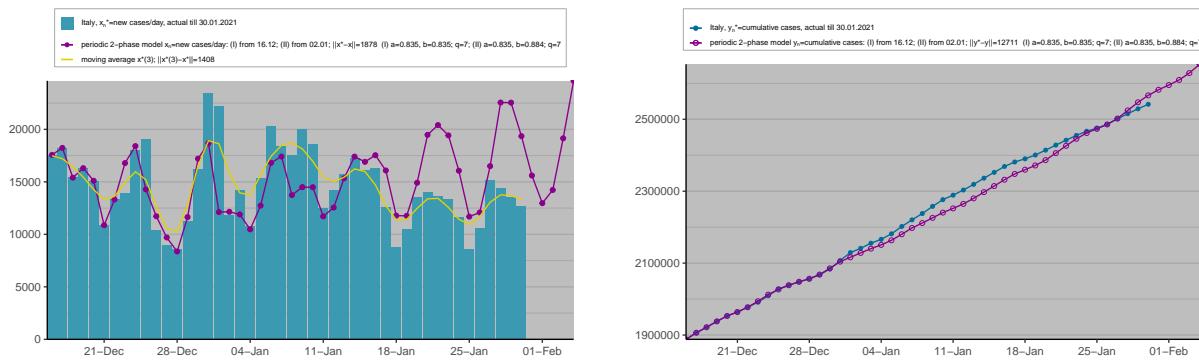


Figure 28: Multi-phase periodic model, Italy

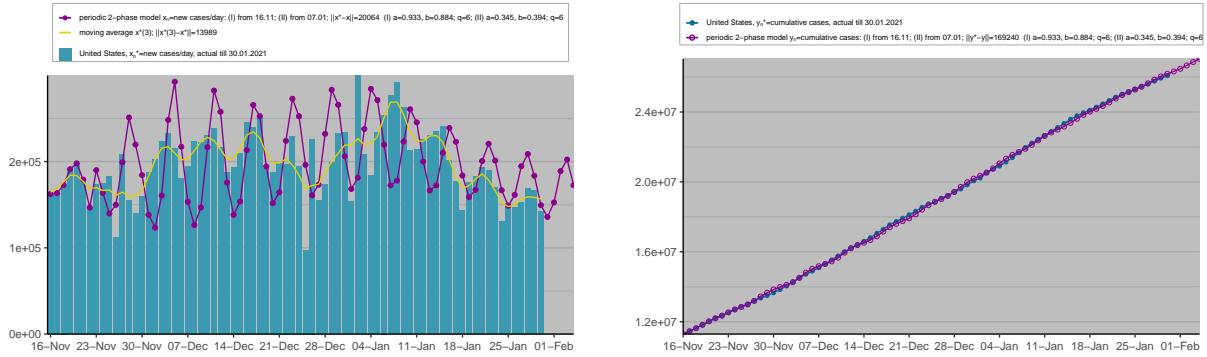


Figure 29: Multi-phase periodic model, United States

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 σ^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.

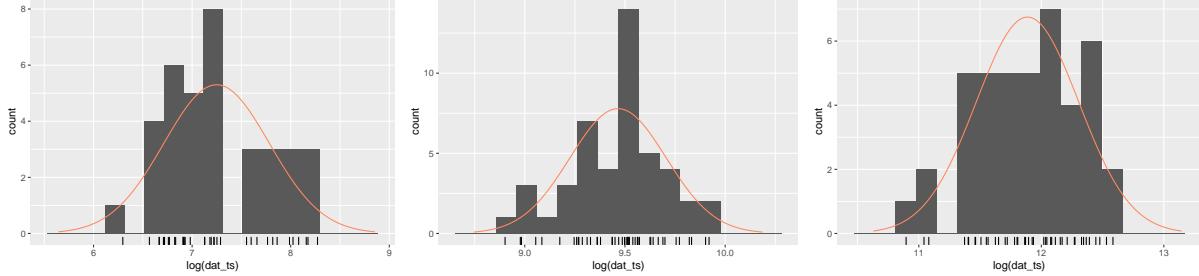


Figure 30: Normality checks, Ireland, Italy and United States

3.1 Holt-Winters' seasonal method

3.1.1 Definitions and Theory

Suppose there are N observations.

Initial step:

$$\begin{aligned} L_s &= \frac{1}{s} \sum_{i=1}^s x_i \\ b_s &= \frac{1}{s} \left[\frac{x_{s+1}-x_1}{s} + \frac{x_{s+2}-x_2}{s} + \dots + \frac{x_{2s}-x_s}{s} \right] \\ S_n &= x_n - L_s, \quad n = 1, \dots, s \end{aligned}$$

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

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

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

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₊)

The parameters α, β, γ are selected using Maximum Likelihood Estimation.

3.1.2 Implementation in R

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

Listing 4: Algorithm for HoltWinters Model

3.1.3 Plots

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

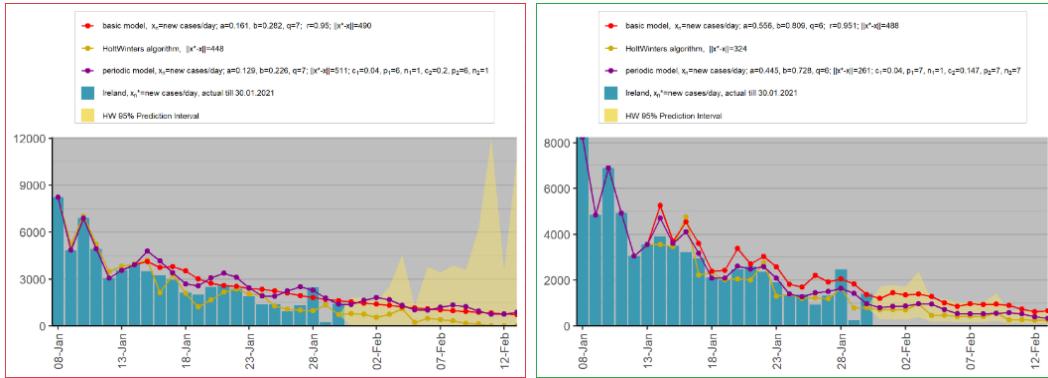


Figure 32: Comparison of HoltWinters multiplicative (left) and additive (right) algorithms

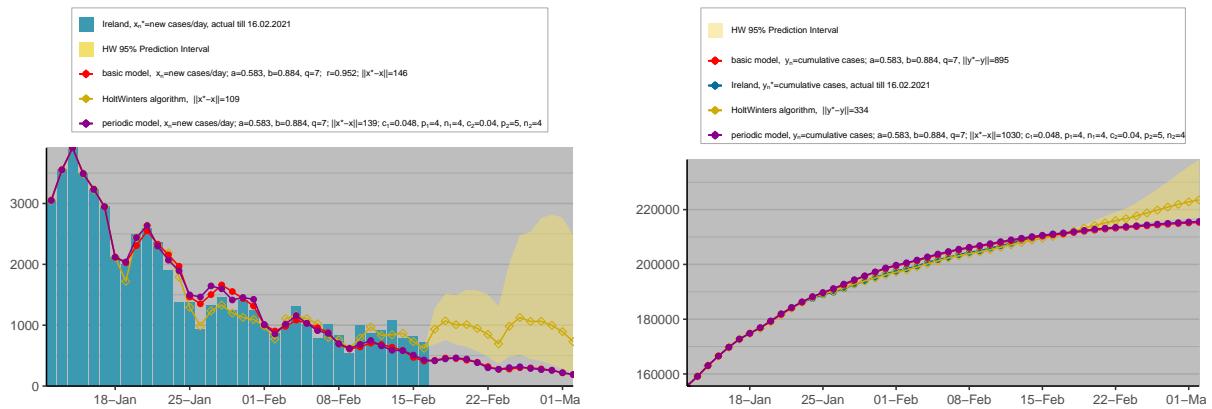


Figure 33: HoltWinters model, Ireland

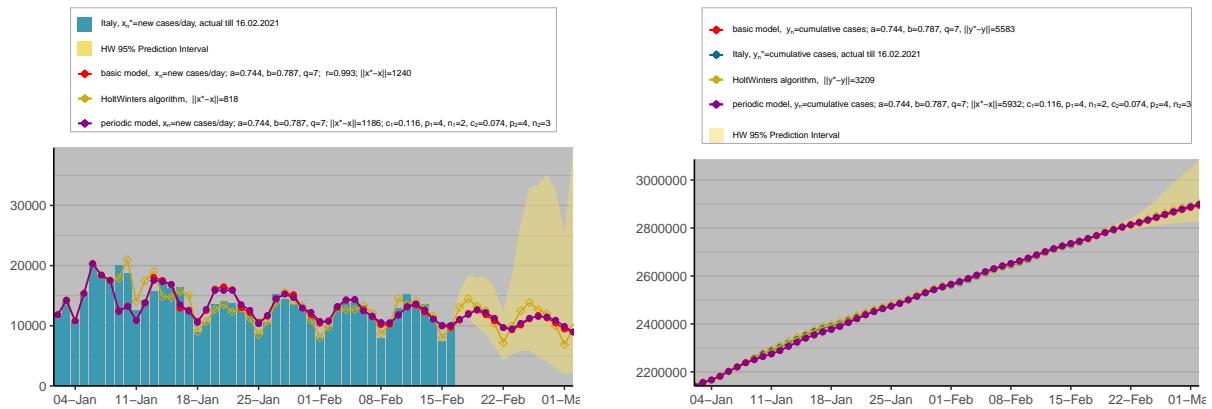


Figure 34: HoltWinters model, Italy

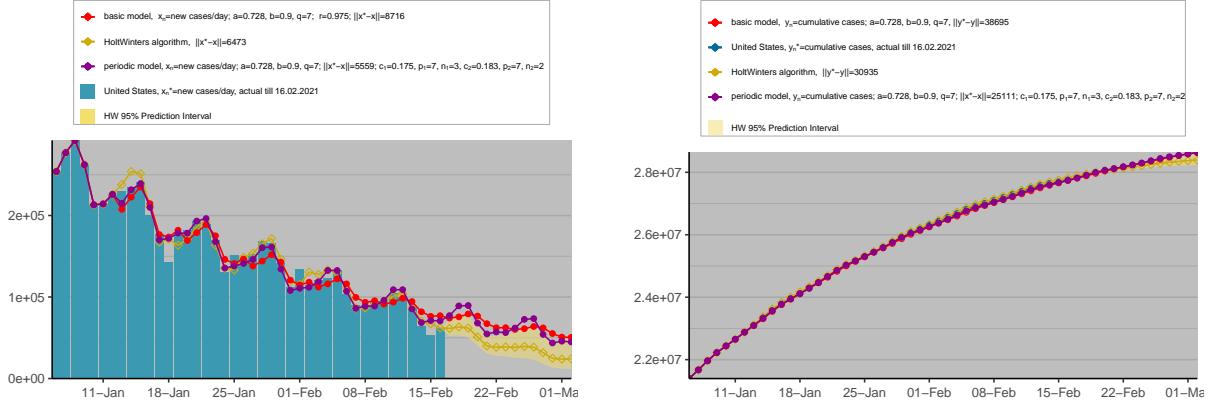


Figure 35: HoltWinters model, United States

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 generally:

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

And similarly for the independent errors ε_n :

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

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

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

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

We can simplify using the backshift operator B :

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

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

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

By convention we use minus signs in the coefficients ψ_1, \dots, ψ_q . We can simplify using the backshift operator B :

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

- The first order differencing of the time series, $I(1)$, is evaluated 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 ε_n are unchanged. Therefore, an ARIMA(p, d, q) model can be evaluated by combining the $AR(p)$, $I(d)$ and $MA(q)$

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

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

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, \dots, x_N states that each x_n is a *linear function* of $x_{n-Ps}, x_{n-(P-1)s}, \dots, x_{n-s}$ 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 :

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

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

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

Again, by convention we use minus signs in the coefficients $\gamma_1, \dots, \gamma_Q$

We can simplify using the backshift operator B :

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

- The first order seasonal differencing of the time series, $I_s(1)$, is evaluated 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 seasonal period s) with the previous ARIMA(p, d, q) to get the definition of an ARIMA(p, d, q)(P, D, Q) _{s} model

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

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

The orders p, d, q and P, D, Q are computed by analysing the correlation functions (ACF and PACF).

The $P + Q + p + q + 1$ coefficients $c, \phi_1, \dots, \phi_p, \psi_1, \dots, \psi_q, \beta_1, \dots, \beta_P, \gamma_1, \dots, \gamma_Q$ are computed using Maximum Likelihood Estimation.

Again, 15 can similarly be used to compute forecasted values x_{N+1}, \dots, x_{N+m} .

3.2.2 Implementation in R

```

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

```

Listing 5: Algorithm for ARIMA Model

3.2.3 Plots

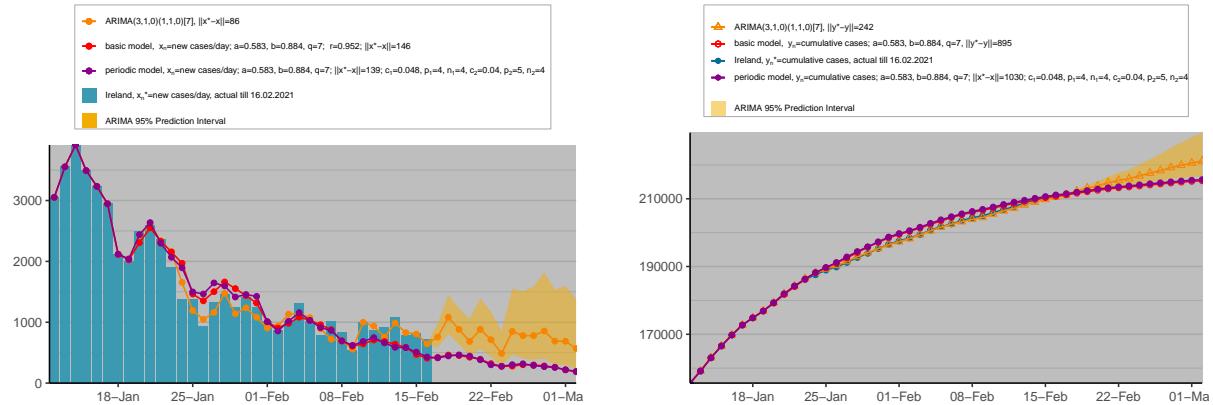


Figure 36: ARIMA model, Ireland

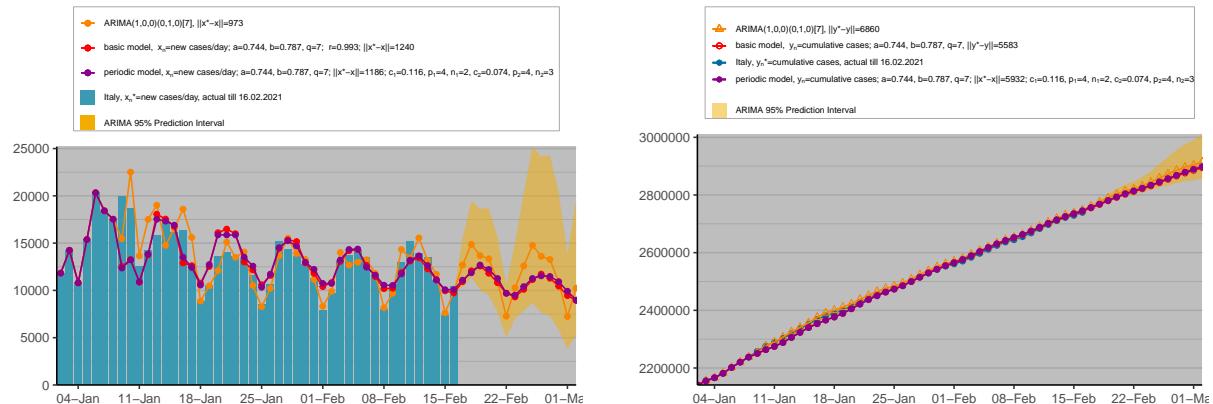


Figure 37: ARIMA model, Italy

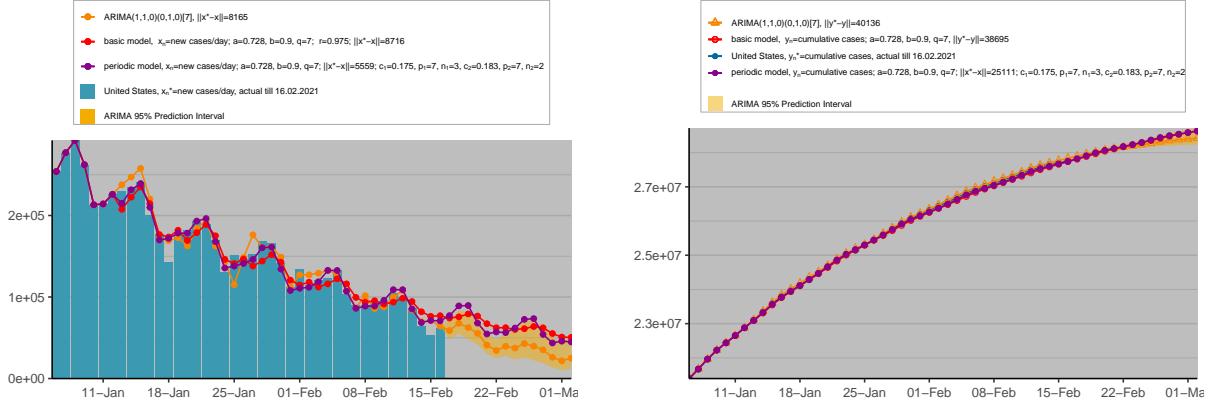


Figure 38: ARIMA model, United States

3.3 Neural network models

An ARIMA($p, 0, 0$) has inputs x_{n-1}, \dots, x_{n-p} and parameters ϕ_1, \dots, ϕ_p in order to compute x_n via a linear combination.

Similarly, an ARIMA($p, 0, 0$)($P, 0, 0$)_s has inputs $x_{n-1}, \dots, x_{n-p}, x_{n-s}, \dots, x_{n-Ps}$ and parameters $\phi_1, \dots, \phi_p, \beta_1, \dots, \beta_P$ in order to compute x_n via another linear combination.

With a Neural Network Auto-regressive model, a *hidden layer* of k inputs is introduced.

The model then becomes non-linear and allows for interactions between inputs (previous values in the time series).

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

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

The diagram ?? is a *multilayer feed-forward network*, as each consecutive layer (left to right) of nodes takes inputs from the previous layer, adding complexity.

The `forecast::nnar()` function only allows for one layer to avoid overfitting of a univariate time series.

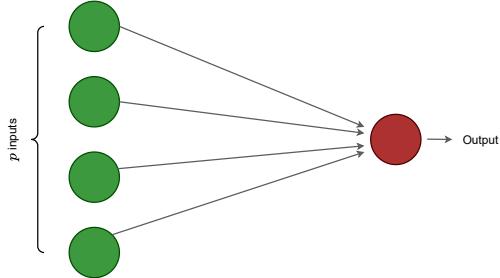


Figure 39: A linear regression model, or ARIMA($p, 0, 0$) model.

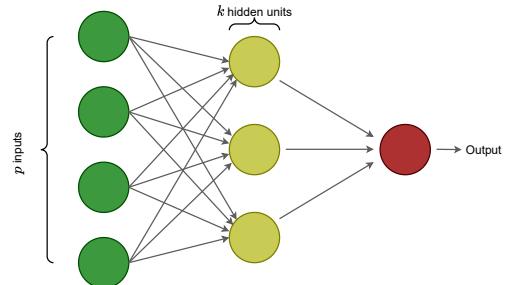


Figure 40: A neural network with p inputs and one hidden layer with k hidden neurons.

3.3.1 Implementation in R

```

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

```

Listing 6: Algorithm for NNAR Model

3.3.2 Plots

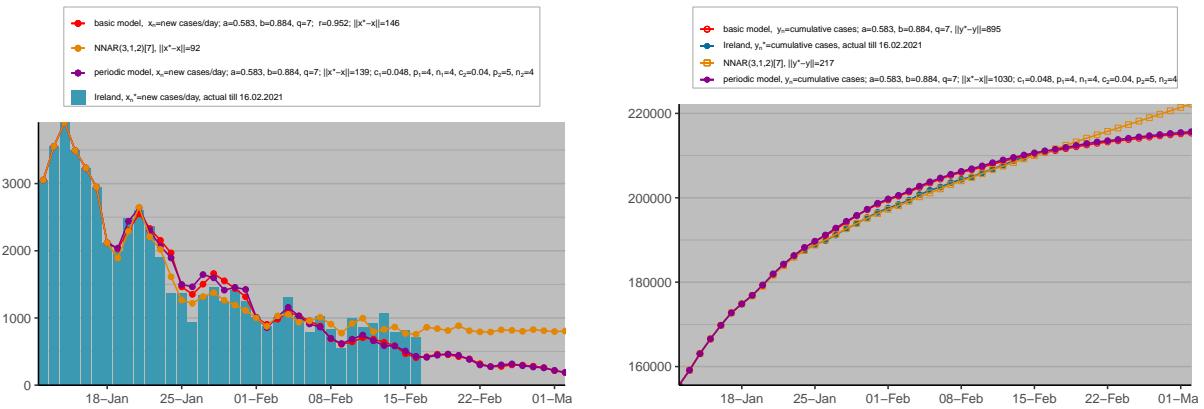


Figure 41: Neural Network Autoregression model, Ireland

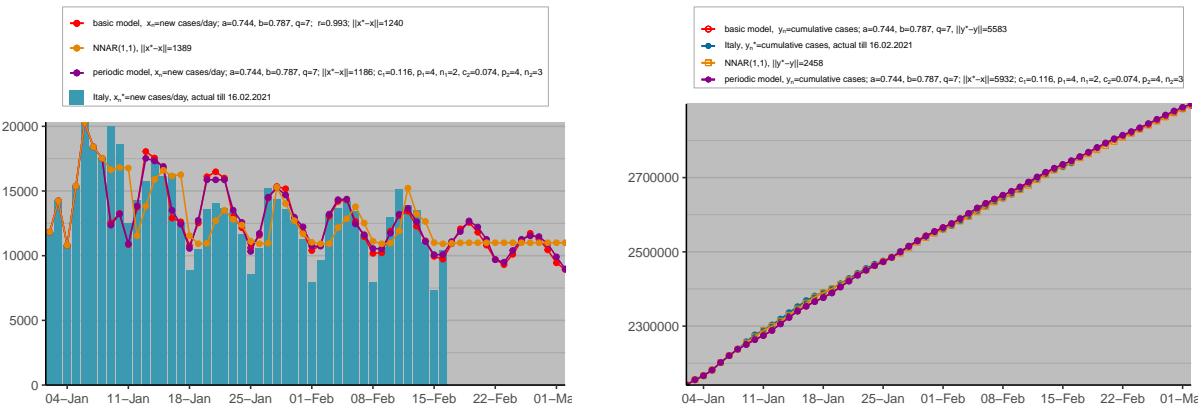


Figure 42: Neural Network Autoregression model, Italy

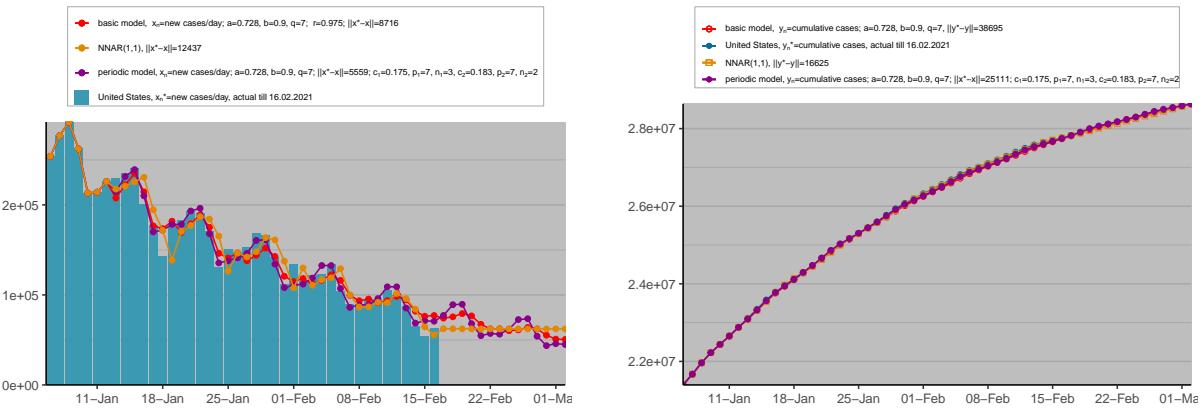


Figure 43: Neural Network Autoregression model, United States

4 Model performance with Train/Test sets

To be more rigorous about judging model performance, and to avoid overfitting, we can fit our models using a *training set* and evaluate it on new data in the *test set*.

We will just display the plots for Ireland here for the sake of brevity.

4.1 Summary

Below is a summary of results of the applied model for the primary date range for the training set. The test set is then the following fourteen days.

model	$\ x - x^*\ _{train}$	$\ y - y^*\ _{train}$	$\ x - x^*\ _{test}$	$\ y - y^*\ _{test}$
1 Basic Recursion	148	916	350	2902
2 Moving Average	130	126	75	109
3 Periodic	140	1049	350	2576
4 HoltWinters	109	334	274	1124
5 ARIMA	86	255	135	306
6 Neural Network	93	210	158	530

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

model	$\ x - x^*\ _{train}$	$\ y - y^*\ _{train}$	$\ x - x^*\ _{test}$	$\ y - y^*\ _{test}$
1 Basic Recursion	1247	5093	5126	34386
2 Moving Average	1072	1030	1034	1175
3 Periodic	1183	5119	4767	26857
4 HoltWinters	818	3209	4357	18774
5 ARIMA	973	6172	3834	14472
6 Neural Network	1384	2468	4602	24578

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

model	$\ x - x^*\ _{train}$	$\ y - y^*\ _{train}$	$\ x - x^*\ _{test}$	$\ y - y^*\ _{test}$
1 Basic Recursion	9049	45009	7882	42965
2 Moving Average	7869	8525	3926	5445
3 Periodic	5660	29129	8945	30279
4 HoltWinters	5763	22833	21703	93870
5 ARIMA	8223	37447	24889	107978
6 Neural Network	12587	17043	8971	49167

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

Below is a summary of the parameters for the mathematical models, also located in the plots in the body of the report

	a	b	q	r	c_1	p_1	n_1	c_2	p_2	n_2
1	0.57	0.86	7.00	0.95	0.04	4.00	4.00	0.04	5.00	4.00

Table 4: Ireland: mathematical models parameters

	a	b	q	r	c_1	p_1	n_1	c_2	p_2	n_2
1	0.71	0.76	7.00	0.99	0.15	4.00	2.00	0.10	4.00	3.00

Table 5: Italy: mathematical models parameters

	a	b	q	r	c_1	p_1	n_1	c_2	p_2	n_2
1	0.71	0.88	7.00	0.97	0.17	7.00	3.00	0.18	7.00	2.00

Table 6: United States: mathematical models parameters

4.2 Base Model

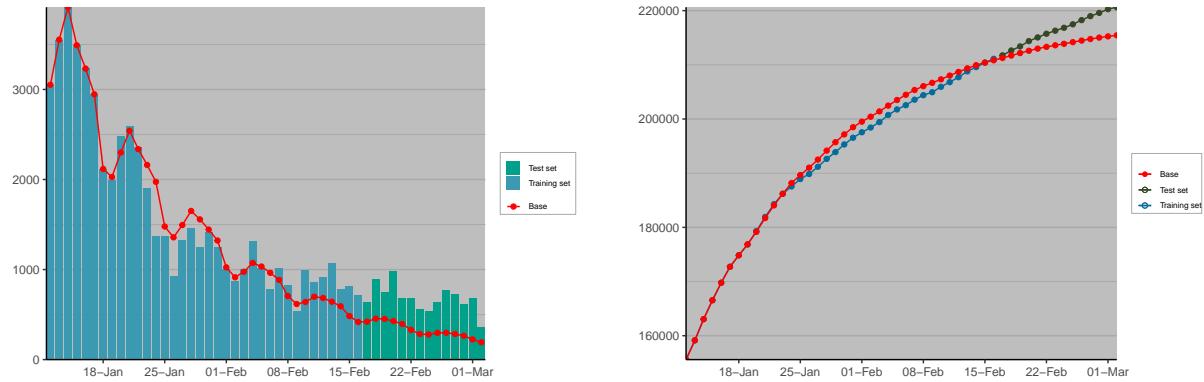


Figure 44: Basic model with train/test split, Ireland

4.3 Periodic model

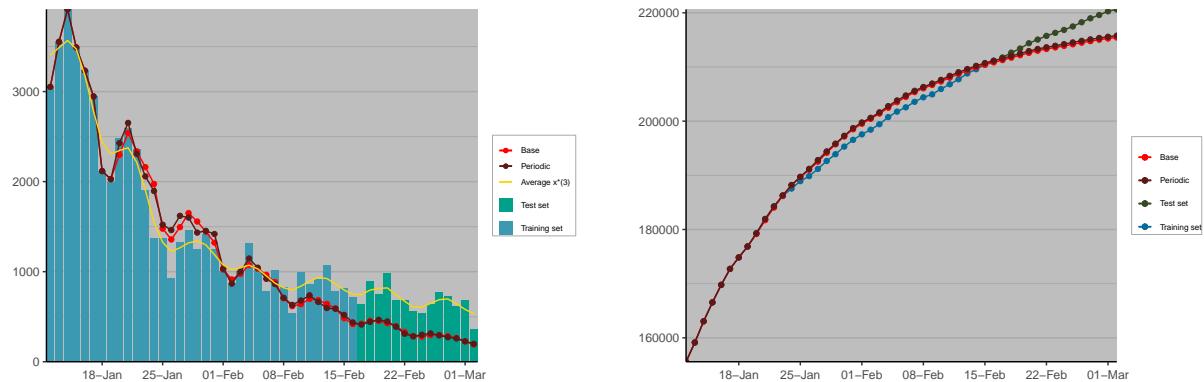


Figure 45: Periodic model with train/test split, Ireland

4.4 HoltWinters model

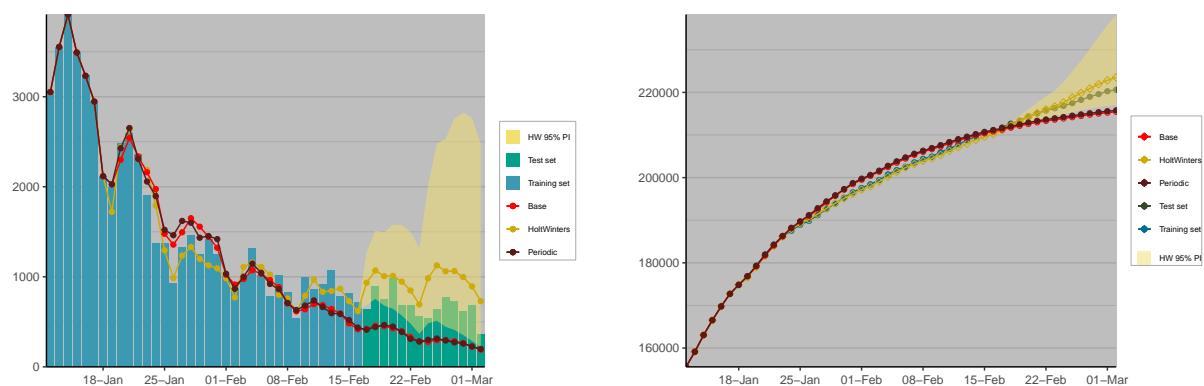


Figure 46: HoltWinters model with train/test split, Ireland

4.5 ARIMA model

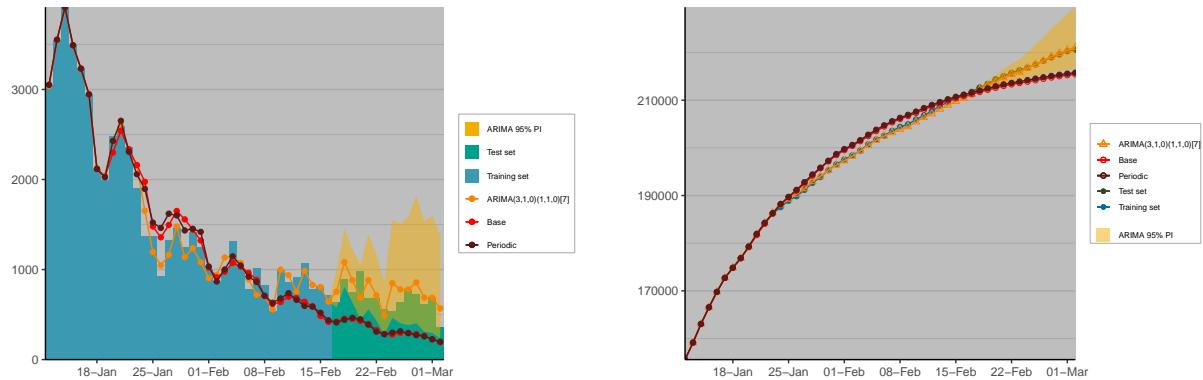


Figure 47: ARIMA model with train/test split, Ireland

4.6 NNAR model

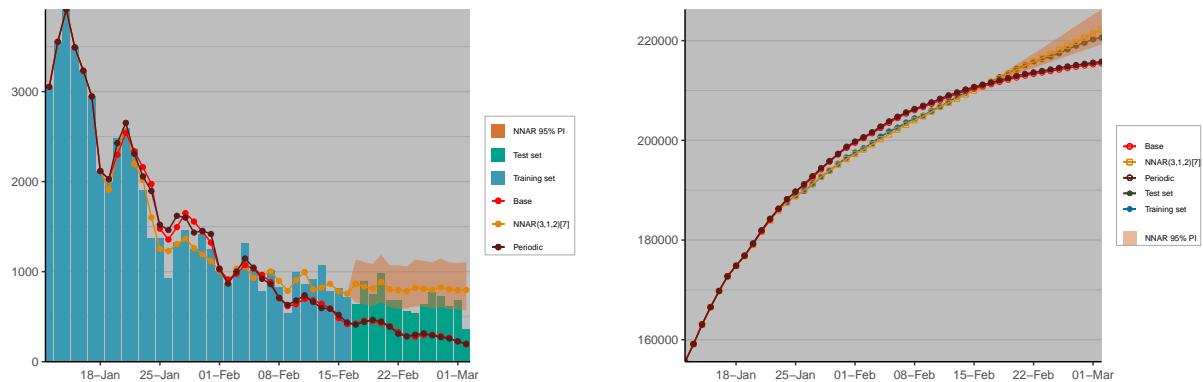


Figure 48: Neural Network Autoregression model with train/test split, Ireland

A Definitions and Theorems for Mathematical Models

A.1 Recurrence equation

This is our general linear recurrence equation with constant coefficients:

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

The characteristic polynomial of 17

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

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

Definition 3. The geometric mean of n values x_1, \dots, x_n is

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

A.2 Results

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

- (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

$$x_n \sim C r^n \text{ as } n \rightarrow \infty. \quad (20)$$

It follows from 20 that if $r < 1$ then the epidemic fades away, whereas if $r > 1$ then it will spread indefinitely.

Proof:

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

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

dividing across by λ^{q+1}

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

And so

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

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

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

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 $\lambda \in \mathbb{C} \setminus \{0\}$ 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.

Therefore r is simple, and is the leading root of 17.

- (b) Let $\lambda_1, \lambda_2, \dots$ be all other distinct roots of f apart from r (so that λ_k are negative or imaginary). Any solution x_n of 17 has the form

$$x_n = Cr^n + \tilde{x}_n \quad (22)$$

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

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

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 Consider a new sequence

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

This satisfies the equation

$$X_{n+1} = A_0 X_n + A_1 X_{n-1} + \cdots + A_q X_{n-q} \quad (24)$$

with $A_k = \frac{a_k}{r^{k+1}}$.

Since

$$\begin{aligned} A_0 X_0 + A_1 X_{n-1} + \cdots + A_q X_{n-q} &= \frac{a_0}{r^1} \frac{x_n}{r^n} + \frac{a_1}{r^2} \frac{x_{n-1}}{r^{n-1}} + \cdots + \frac{a_q}{r^{q+1}} \frac{x_{n-q}}{r^{n-q}} \\ &= \frac{a_0 x_n}{r^{n+1}} + \frac{a_1 x_{n-1}}{r^{n+1}} + \cdots + \frac{a_q x_{n-q}}{r^{n+1}} \\ &= \frac{1}{r^{n+1}} (a_0 x_n + a_1 x_{n-1} + \cdots + a_q x_{n-q}) \\ &= \frac{1}{r^{n+1}} (x_{n+1}) \quad \text{from 17} \\ &= \frac{x_{n+1}}{r^{n+1}} (x_{n+1}) \\ &= X_{n+1} \end{aligned}$$

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

$$A_0 + A_1 + \cdots + A_q = 1 \quad (25)$$

Set $c := \min(X_1, \dots, X_{q+1}) > 0$ 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

$$x_n \geq cr^n$$

But equating with 22 we get

$$\begin{aligned} Cr^n + \tilde{x}_n &\geq cr^n > 0 \\ C + \frac{\tilde{x}_n}{r^n} &\geq c > 0 \end{aligned}$$

Taking the limit as $n \rightarrow \infty$

$$C \geq c > 0 \quad (26)$$

Therefore $C > 0$, as required. \square

Theorem 2. Let $a_k \geq 0$ for all $k = 0, \dots, q$. 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

$$\min\left(1, \left(\frac{a}{b}\right)^{1/m}\right) \leq r \leq \max\left(1, \left(\frac{a}{b}\right)^{1/m}\right) \quad (27)$$

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

Proof:

(a) We have

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

We know f is increasing.

If $r = 1$ then the inequality obviously holds.

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

$$r^{q+1} - a_0r^q - a_1r^{q-1} - a_2r^{q-2} - \cdots - a_{q-1}r - a_q = 0$$

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

$$\implies r^{q+1} - a_0r^q - a_mr^{q-m} - a_{m+1}r^{q-m-1} - \cdots - a_{q-1}r - a_q = 0$$

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

$$\implies r^{q+1} - r^q + br^q - a_1r^{q-m} - \cdots - a_q = 0$$

$$\implies r^{q+1} - r^q = -br^q + a_mr^{q-m} + \cdots + a_q$$

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

and so

$$0 < -br^q + a_mr^{q-m} + \cdots + a_q$$

$$\implies br^q < a_mr^{q-m} + \cdots + a_q$$

$$\leq a_mr^{q-m} + \cdots + a_qr^{q-m}$$

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

$$= ar^{q-m}$$

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

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

We can combine both cases with $x \leq \max(1, r)$ and $x \geq \min(1, r)$ (for any $x \in \mathbb{R}$) to get 27, 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, set this to be day 0. On the day $k = 1, \dots, q$ 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

$$\begin{aligned} ac_k u + a(1 - b)c_k u + a(1 - b)^2 c_k u + \dots &= ac_k u \sum_{n=0}^{\infty} (1 - b)^n \\ &= \frac{ac_k u}{1 - (1 - b)} \quad \text{since } 0 < 1 - b < 1 \\ &= \frac{a}{b} c_k u \end{aligned}$$

other people.

Hence, the initial group of u people infects in total

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

other people.

And we defined R_0 as being the unit reproduction number per infected person ($u = 1$).

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

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

$$r_{n+1} = r_n - \frac{f(r_n)}{f'(r_n)}, \quad n \geq 0 \quad (28)$$

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

$$\{r_n\} \rightarrow r \quad \text{as } n \rightarrow \infty \quad (29)$$

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

$$r \approx \frac{q \left(\frac{a}{b}\right)^{\frac{1}{2q}} - (q-1)(1-b) + (ab)^{-\frac{1}{2}}}{(q+1) - q(1-b) \left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \quad (30)$$

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

$$f(\lambda) = \lambda^{q+1} - (1-b)\lambda^q - a \quad (31)$$

Let r be its leading root, i.e. $f(r) = 0$

Then by 2, we get

$$\min \left(1, \left(\frac{a}{b}\right)^{1/q} \right) \leq r \leq \max \left(1, \left(\frac{a}{b}\right)^{1/q} \right) \quad (32)$$

since $m = q$ in our polynomial.

Taking the geometric mean of the bounds, defined as 19, we get an approximation for r

$$r_0 = \left(1 \cdot \left(\frac{a}{b}\right)^{1/q} \right)^{\frac{1}{2}} = \left(\frac{a}{b}\right)^{\frac{1}{2q}} \quad (33)$$

The derivative of our characteristic polynomial 31 is

$$f'(\lambda) = (q+1)\lambda^q - q(1-b)\lambda^{q-1} \quad (34)$$

Then we apply Newton's method 28 once to get

$$\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}} \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}{2}}} \\
&= \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}$$

□

While one iteration of Newton's method is often enough with our initial choice $r_0 = (R_0)^{\frac{1}{2q}}$, we will show that it converges to r in the limit

Lemma 5. Let r' be the largest non-negative root of $f'(\lambda)$, the derivative of 18.

If we choose an initial $r_0 > r'$ then $\lim_{n \rightarrow \infty} r_n = r$.

Proof: Since $f(\lambda)$ defined above has leading root r and has leading coefficient 1, we know that $f(\lambda) > 0$ for $\lambda > r$.

Also, since r is the only positive root, by part (a) of 1, then $f(\lambda) < 0$ for $0 < \lambda < r$.

We have

$$r_1 = r_0 - \frac{f(r_0)}{f'(r_0)} \tag{35}$$

We require r_0 and r_1 to be in the open interval (r', ∞) in order for r_n to converge to r .

There are two cases to consider:

- If our chosen r_0 is less than r , then $f(r_0) < 0$ and $f'(r_0) > 0$.

$$r_1 = r_0 - \underbrace{\frac{f(r_0)}{f'(r_0)}}_{<0} > r_0 > r'$$

Then $r_1 > r'$ and the sequence will converge.

- If our chosen r_0 is greater than r , then $f(r_0) > 0$ and $f'(r_0) > 0$.

By a similar argument to above,

$$r_1 < r_0$$

so we need to do more work to show $r_1 > r'$ in this case

Since f is continuous in $[r_1, r_0]$ and differentiable in (r_1, r_0) , we can apply the *Mean Value Theorem* to imply that there exists some point $c \in (r_1, r_0)$ such that

$$f'(c) = \frac{f(r_0) - f(r_1)}{r_0 - r_1} \quad (36)$$

We can rearrange this equation

$$f(r_1) = f(r_0) + f'(c)(r_1 - r_0)$$

But from 35 we have

$$r_1 - r_0 = -\frac{f(r_0)}{f'(r_0)}$$

And so

$$f(r_1) = f(r_0) - f'(c) \frac{f(r_0)}{f'(r_0)} \quad (37)$$

$$= f(r_0) \left(1 - \frac{f'(c)}{f'(r_0)} \right) \quad (38)$$

But since the function f is convex in the open interval (r', ∞) , we know that $0 < f'(c) < f'(r_0)$. Therefore

$$0 < f(r_1) < f(r_0)$$

which gives

$$r < r_1 < r_0$$

and finally

$$r' < r < r_1 \quad (39)$$

And so $r_1 > r$ in both cases, then Newton's method ensures $\lim_{n \rightarrow \infty} r_n = r$, as required. \square

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

$$f_{\text{avg}} = \frac{1}{x_2 - x_1} \int_{x_1}^{x_2} f(x) dx \quad (40)$$

Proposition 6. The average value of the sequence a_0, \dots, a_N defined by 7 can be reasonably estimated by our previous parameter a .

Similarly, the average value of the sequence b_0, \dots, 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, \dots, a_N , i.e.

$$a(x) = a \left(1 + c_1 \left(\sin \left(\frac{2\pi}{p_1} (x - n_1) \right) \right) \right), \quad 0 \leq x \leq N \quad (41)$$

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 $\sin(A - B) = \sin(A)\cos(B) - \cos(A)\sin(B)$

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

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

$$a_{\text{avg}} = a - \frac{ac_1 p_1}{2\pi N} \left(\cos\left(\frac{2\pi(N + n_1)}{p_1}\right) + \cos\left(\frac{2\pi n_1}{p_1}\right) \right)$$

Since $\cos(\cdot)$ is bounded between ± 1 and

$c_1 p_1 < 0.2 \cdot 8 = 1.6$ and $N > 50$ usually, we see that

$$\left| \frac{c_1 p_1}{2\pi N} \left(\cos\left(\frac{2\pi(N + n_1)}{p_1}\right) + \cos\left(\frac{2\pi n_1}{p_1}\right) \right) \right| < \frac{1.6}{300} \cdot (1 + 1) = \frac{4}{375}$$

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

Therefore a is a reasonable estimate for the average a_{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_{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 of the colours for plotting are selected using colour palettes from the `wesanderson` package [15].



Figure 49: 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 50: 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	-7.939	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 51: 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.

```

1 modnorm <- function(x,modx) return(floor(sum(abs(x-modx))/length(x)))
2
3 xntoyn <- function(xn) return(cumsum(xn))
4
5 basicmodx <- function(x, pars, len = 0){
6   q <- floor(pars[1])
7   a <- pars[2]
8   b <- pars[3]
9   modx <- x[1:q]
10  for(i in (q+1):(length(x)+len)){
11    modx[i] <- (1-b)*modx[i-1] + a*modx[i-q]
12  }
13  return(modx)
14}
15
16 norm <- function(par, x) return(modnorm(x,basicmodx(x, par)))
17
18 normy <- function(par, x, y) return(modnorm(y,xntoyn(basicmodx(x, par))))
19
20 normalize <- function(x) return((x-min(x))/(max(x)-min(x)))
21
22 basef <- function(lambda,par) {
23   return(lambda^(par[1]+1)-(1-par[3])*lambda^(par[1])-par[2])
24 }
25
26 basefprime <- function(lambda,par) {
27   return((par[1]+1)*lambda^(par[1])-(1-par[3])*par[1]*lambda^(par[1]-1))
28 }
29
30 normC <- function(par,x,r){
31   return(modnorm(x, par*r^(0:(length(x)!is.na(x))-1)))
32 }
33
34 movingavg <- function(x){

```

```

35|   mavgx <- (x[1]+x[2])/2
36|   for(i in 2:(length(x)-1)){
37|     mavgx[i] <- sum(x[(i-1):(i+1)])/3
38|   }
39|   mavgx[length(x)] <- (x[length(x)-1]+x[length(x)])/2
40|   return(mavgx)
41}
42
43 normper <- function(par, q, x) return(modnorm(x, modxper(par, q, x)))
44
45 modxper <- function(par, q, x, len = 0){
46   #a,b,c1,c2,p1,p2,n1,n2
47   an <- par[1]*(1+par[3]*sin(2*pi*(1:(length(x)+len) - par[7])/par[5]))
48   bn <- par[2]*(1+par[4]*sin(2*pi*(1:(length(x)+len) - par[8])/par[6]))
49   modx <- x[1:q]
50   for(i in (q+1):(length(x)+len)){
51     modx[i] <- (bn[i]*(1-bn[i-1]))*modx[i-1]/bn[i-1] +
52       (an[i-q]*bn[i])*modx[i-q]/bn[i-q]
53   }
54   return(modx)
55}

```

Listing 7: Model helper functions

```

1 plot_xn <- function(countrydat, cols, labs){
2   p <- ggplot(countrydat, binwidth = 0) +
3     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
4     scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+
5     scale_y_continuous(expand = c(0,0)) +
6     scale_fill_manual(values = cols$xn, name = "", labels = labs$train) +
7     xntheme()
8   return(p)
9 }
10
11 plot_yn <- function(countrydat, cols, labs){
12   p <- ggplot(countrydat) +
13     geom_line(aes(x = date, y = yn, colour = "blue")) +
14     geom_point(aes(x = date, y = yn, colour = "blue"))+
15     scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+
16     scale_y_continuous(expand = c(0,0)) +
17     scale_colour_manual(values = cols$yn, name = "", labels = labs$train) +
18     yntheme()
19   return(p)
20 }
21
22 plot_basexn <- function(modeldat, cols, labs){
23   p <- ggplot(modeldat, binwidth = 0) +
24     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
25     geom_point(aes(x = date, y = basexn, colour = cols$basexn)) +
26     geom_line(aes(x = date, y = basexn, colour = cols$basexn)) +
27     gg_scale_xy +
28     scale_fill_manual(name = "leg", values = c("test" = cols$xntest, "train" = cols$xn),
29                       labels = c("test" = labs$test, "train" = labs$train)) +
30     scale_colour_manual(name = "leg", values = cols$basexn, labels = labs$base) +
31     xntheme()
32   return(p)
33 }
34
35 plot_baseyn <- function(modeldat, cols, labs){
36   p <- ggplot(modeldat) +
37     geom_point(aes(x = date, y = yn, colour = tgroup)) +
38     geom_line(aes(x = date, y = yn, colour = tgroup)) +
39     geom_point(aes(x = date, y = baseyn, colour = "base")) +
40     geom_line(aes(x = date, y = baseyn, colour = "base")) +
41     gg_scale_xy +
42     scale_colour_manual(name = "leg",
43                         values = c("base" = cols$baseyn, "test" = cols$yntest, "train" = cols$yn),
44                         labels = c("base" = labs$base, "test" = labs$test, "train" = labs$train),
45                         guide = guide_legend	override.aes = list(
46                           shape = c("base" = 16, "test" = 1, "train" = 1)))) +
47     yntheme()
48   return(p)
49 }

```

```

50| plot_crn <- function(modeldat, cols, labs){
51|   p <- ggplot(modeldat, binwidth = 0) +
52|     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
53|     geom_point(aes(x = date, y = basexn, colour = "basexn")) +
54|     geom_line(aes(x = date, y = basexn, colour = "basexn")) +
55|     geom_line(aes(x = date, y = Crn, colour = "Crn")) +
56|     gg_scale_xy +
57|     scale_fill_manual(name = "leg", values = c("test" = cols$xntest, "train" = cols$xn),
58|                        labels = c("test" = labs$test, "train" = labs$train)) +
59|     scale_colour_manual(name = "leg",
60|                           values = c("basexn" = cols$basexn, "Crn" = cols$Crn),
61|                           labels = c("basexn" = labs$base, "Crn" = labs$Crn),
62|                           guide = guide_legend(override.aes = list(
63|                                         shape = c("basexn" = 16, "Crn" = NA)))) +
64|     xntheme()
65|   return(p)
66| }
67|
68|
69| plot_mavgx3 <- function(modeldat, cols, labs){
70|   p <- ggplot(modeldat, binwidth = 0) +
71|     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
72|     geom_point(aes(x = date, y = basexn, colour = "basexn")) +
73|     geom_line(aes(x = date, y = basexn, colour = "basexn")) +
74|     geom_line(aes(x = date, y = mavgx3, colour = "x3")) +
75|     scale_fill_manual(name = "leg", values = c("test" = cols$xntest, "train" = cols$xn),
76|                        labels = c("test" = labs$test, "train" = labs$train)) +
77|     scale_colour_manual(values = c("basexn" = cols$basexn, "x3" = cols$x3),
78|                           labels = c("basexn" = labs$base, "x3" = labs$x3),
79|                           guide = guide_legend(override.aes = list(
80|                                         shape = c("basexn" = 16, "x3" = NA)))) +
81|     gg_scale_xy + xntheme()
82|   return(p)
83| }
84|
85| plot_periodic <- function(modeldat, cols, labs){
86|   p <- ggplot(modeldat, binwidth = 0) +
87|     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
88|     geom_point(aes(x = date, y = basexn, colour = "base")) +
89|     geom_line(aes(x = date, y = basexn, colour = "base")) +
90|     geom_point(aes(x = date, y = periodic, colour = "periodic")) +
91|     geom_line(aes(x = date, y = periodic, colour = "periodic")) +
92|     geom_line(aes(x = date, y = mavgx3, colour = "x3")) +
93|     gg_scale_xy +
94|     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
95|            fill=guide_legend(ncol=1,nrow=2,byrow=TRUE)) +
96|     scale_fill_manual(name = "leg",
97|                       values = c("test" = cols$xntest, "train" = cols$xn),
98|                       labels = c("test" = labs$test, "train" = labs$train)) +
99|     scale_colour_manual(values = c("base" = cols$basexn, "periodic" = cols$periodic, "x3" = cols$x3),
100|                           labels = c("base" = labs$base, "periodic" = labs$periodic, "x3" = labs$x3
101|                                         )) +
102|     guides(colour = guide_legend(override.aes = list(shape = c("base" = 16, "periodic" = 16, "x3"
103|                                         = NA)))) +
104|     xntheme()
105|   return(p)
106|
107| plot_periodicity <- function(modeldat, cols, labs){
108|   p <- ggplot(modeldat) +
109|     geom_point(aes(x = date, y = yn, colour = tgroup)) +
110|     geom_line(aes(x = date, y = yn, colour = tgroup)) +
111|     geom_point(aes(x = date, y = baseyn, colour = "base")) +
112|     geom_line(aes(x = date, y = baseyn, colour = "base")) +
113|     geom_point(aes(x = date, y = periodicy, colour = "periodic")) +
114|     geom_line(aes(x = date, y = periodicy, colour = "periodic")) +
115|     gg_scale_xy +
116|     guides(colour=guide_legend(ncol=1,nrow=4,byrow=TRUE)) +
117|     scale_colour_manual(values = c("base" = cols$baseyn, "periodic" = cols$periodic, "test" =
118|                                         cols$yntest, "train" = cols$yn),
119|                           labels = c("base" = labs$base, "periodic" = labs$periodic, "test" = labs$test,
120|                                         "train" = labs$train),
121|                           guide = guide_legend(override.aes = list(

```

```

119          shape = c("base" = 1, "periodic" = 1, "test" = 16, "train" = 16)))) +
120      yntheme()
121      return(p)
122  }
123
124 plot_hw <- function(modeldat, cols, labs){
125   p <- ggplot(modeldat, binwidth = 0) +
126     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
127     geom_ribbon(aes(x = date, ymin = hwlo, ymax = hwhi, fill = "hw"), alpha = 0.5) +
128     geom_point(aes(x = date, y = hwxn, colour = "hw")) +
129     geom_line(aes(x = date, y = hwxn, colour = "hw")) +
130     geom_point(aes(x = date, y = basexn, colour = "base")) +
131     geom_line(aes(x = date, y = basexn, colour = "base")) +
132     geom_point(aes(x = date, y = periodic, colour = "periodic")) +
133     geom_line(aes(x = date, y = periodic, colour = "periodic")) +
134     gg_scale_xy +
135     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
136            fill=guide_legend(ncol=1,nrow=3,byrow=TRUE)) +
137     scale_fill_manual(labels = c("hw" = labs$hwpi, "test" = labs$test, "train" = labs$train),
138                       values = c("hw" = cols$hwpi, "test" = cols$xntest, "train" = cols$xn)) +
139     scale_colour_manual(labels = c("base" = labs$base, "hw" = labs$hw, "periodic" = labs$periodic
140                           ),
141                           values = c("base" = cols$basexn, "hw" = cols$hw, "periodic" = cols$periodic),
142                           guide = guide_legend(override.aes = list(
143                                         shape = c("base" = 16, "hw" = 18, "periodic" = 16)))) +
144   xntheme()
145   return(p)
146 }
147
148 plot_hwy <- function(modeldat, cols, labs){
149   p <- ggplot(modeldat) +
150     geom_point(aes(x = date, y = yn, colour = tgroup)) +
151     geom_line(aes(x = date, y = yn, colour = tgroup)) +
152     geom_ribbon(aes(x = date, ymin = hwylo, ymax = hwyhi, fill = "pi"), alpha = 0.5) +
153     geom_point(aes(x = date, y = hwyn, colour = "hw"), shape = 5) +
154     geom_line(aes(x = date, y = hwyn, colour = "hw")) +
155     geom_point(aes(x = date, y = baseyn, colour = "base"), shape = 1) +
156     geom_line(aes(x = date, y = baseyn, colour = "base")) +
157     geom_point(aes(x = date, y = periodicy, colour = "periodic")) +
158     geom_line(aes(x = date, y = periodicy, colour = "periodic")) +
159     gg_scale_xy +
160     guides(colour=guide_legend(ncol=1,nrow=5,byrow=TRUE),
161            fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
162     scale_fill_manual(values = c("pi" = cols$hwpi),
163                       labels = c("pi" = labs$hwpi)) +
164     scale_colour_manual(values = c("base" = cols$baseyn, "hw" = cols$hw, "periodic" = cols$periodic,
165                               "test" = cols$yntest, "train" = cols$yn),
166                         labels = c("base" = labs$base, "hw" = labs$hw, "periodic" = labs$periodic,
167                               "test" = labs$test, "train" = labs$train),
168                         guide = guide_legend(override.aes = list(
169                                       shape = c("base" = 1, "hw" = 5, "periodic" = 1, "test" = 16, "train" = 16)))) +
170   yntheme()
171   return(p)
172 }
173
174 plot_arima <- function(modeldat, cols, labs){
175   p <- ggplot(modeldat, binwidth = 0) +
176     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
177     geom_ribbon(aes(x = date, ymin = arimalo, ymax = arimahi, fill = "pi"), alpha = 0.5) +
178     geom_point(aes(x = date, y = arimaxn, colour = "arima")) +
179     geom_line(aes(x = date, y = arimaxn, colour = "arima")) +
180     geom_point(aes(x = date, y = basexn, colour = "base")) +
181     geom_line(aes(x = date, y = basexn, colour = "base")) +
182     geom_point(aes(x = date, y = periodic, colour = "periodic")) +
183     geom_line(aes(x = date, y = periodic, colour = "periodic")) +
184     gg_scale_xy +
185     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
186            fill=guide_legend(ncol=1,nrow=3,byrow=TRUE)) +
187     scale_fill_manual(values = c("pi" = cols$arimapi, "test" = cols$xntest, "train" = cols$xn),
188                       labels = c("pi" = labs$arimapi, "test" = labs$test, "train" = labs$train)) +
189     scale_colour_manual(values = c("arima" = cols$arima, "base" = cols$basexn, "periodic" = cols$periodic))

```

```

    periodic),
187     labels = c("arima" = labs$arima, "base" = labs$base, "periodic" = labs$periodic)) +
188   xntheme()
189   return(p)
190 }
191
192 plot_arimay <- function(modeldat, cols, labs){
193   p <- ggplot(modeldat) +
194     geom_point(aes(x = date, y = yn, colour = tgroup)) +
195     geom_line(aes(x = date, y = yn, colour = tgroup)) +
196     geom_ribbon(aes(x = date, ymin = arimaylo, ymax = arimayhi, fill = "pi"), alpha = 0.5) +
197     geom_point(aes(x = date, y = arimayn, colour = "arima"), shape = 2) +
198     geom_line(aes(x = date, y = arimayn, colour = "arima")) +
199     geom_point(aes(x = date, y = baseyn, colour = "base"), shape = 1) +
200     geom_line(aes(x = date, y = baseyn, colour = "base")) +
201     geom_point(aes(x = date, y = periodicy, colour = "periodic")) +
202     geom_line(aes(x = date, y = periodicy, colour = "periodic")) +
203     gg_scale_xy +
204     scale_fill_manual(values = c("pi" = cols$arimapi),
205                       labels = c("pi" = labs$arimapi)) +
206     scale_colour_manual(values = c("arima" = cols$arima, "base" = cols$baseyn, "periodic" = cols$periodic,
207                          "test" = cols$xntest, "train" = cols$yn),
208                         labels = c("arima" = labs$arima, "base" = labs$base, "periodic" = labs$periodic,
209                         "test" = labs$test, "train" = labs$train),
210                         guide = guide_legend(override.aes = list(
211                           shape = c("arima" = 2, "base" = 1, "periodic" = 1, "test"=16, "train"=16),
212                           fill = c("arima" = "#F0E68C", "base" = "#D9E1F2", "periodic" = "#B2DF8A",
213                           "test" = "#F0E68C", "train" = "#D9E1F2")))) +
214   yntheme()
215   return(p)
216 }
217
218 plot_hwarima <- function(modeldat, cols, labs){
219   p <- ggplot(modeldat, binwidth = 0) +
220     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
221     geom_point(aes(x = date, y = arimaxn, colour = "arima")) +
222     geom_line(aes(x = date, y = arimaxn, colour = "arima")) +
223     geom_point(aes(x = date, y = hwxn, colour = "hw")) +
224     geom_line(aes(x = date, y = hwxn, colour = "hw")) +
225     gg_scale_xy +
226     guides(colour=guide_legend(ncol=1,nrow=2,byrow=TRUE),
227            fill=guide_legend(ncol=1,nrow=2,byrow=TRUE)) +
228     scale_fill_manual(values = c("test"=cols$xntest, "train"=cols$xn),
229                       labels = c("test"=labs$test, "train"=labs$train)) +
230     scale_colour_manual(values = c("arima" = cols$arima, "hw" = cols$hw),
231                       labels = c("arima" = labs$arima, "hw" = labs$hw)) +
232     xntheme()
233   return(p)
234 }
235
236 plot_nn <- function(modeldat, cols, labs){
237   p <- ggplot(modeldat, binwidth = 0) +
238     geom_bar(aes(x = date, y = xn, fill = tgroup), stat = "identity") +
239     geom_ribbon(aes(x = date, ymin = nnlo, ymax = nnhi, fill = "pi"), alpha = 0.5) +
240     geom_point(aes(x = date, y = nnxn, colour = "nn")) +
241     geom_line(aes(x = date, y = nnxn, colour = "nn")) +
242     geom_point(aes(x = date, y = basexn, colour = "base")) +
243     geom_line(aes(x = date, y = basexn, colour = "base")) +
244     geom_point(aes(x = date, y = periodic, colour = "periodic")) +
245     geom_line(aes(x = date, y = periodic, colour = "periodic")) +
246     gg_scale_xy +
247     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
248            fill=guide_legend(ncol=1,nrow=3,byrow=TRUE)) +
249     scale_fill_manual(values = c("pi" = cols$npipi, "test"=cols$xntest, "train"=cols$xn),
250                       labels = c("pi" = labs$npipi, "test"=labs$test, "train"=labs$train)) +
251     scale_colour_manual(values = c("base" = cols$basexn, "nn" = cols$nn, "periodic" = cols$periodic),
252                       labels = c("base" = labs$base, "nn" = labs$nn, "periodic" = labs$periodic)),
253     guide = guide_legend(override.aes = list(
254       shape = c("base" = 16, "nn" = 0, "periodic" = 16)))) +
255   xntheme()
256   return(p)
257 }
```

```

254|
255| plot_nny <- function(modeldat, cols, labs){
256|   p <- ggplot(modeldat) +
257|     geom_point(aes(x = date, y = yn, colour = tgroup)) +
258|     geom_line(aes(x = date, y = yn, colour = tgroup)) +
259|     geom_ribbon(aes(x = date, ymin = nnylo, ymax = nnhi), fill = "pi"), alpha = 0.5) +
260|     geom_point(aes(x = date, y = nnyn, colour = "nn"), shape = 0) +
261|     geom_line(aes(x = date, y = nnyn, colour = "nn")) +
262|     geom_point(aes(x = date, y = baseyn, colour = "base"), shape = 1) +
263|     geom_line(aes(x = date, y = baseyn, colour = "base")) +
264|     geom_point(aes(x = date, y = periodicy, colour = "periodic")) +
265|     geom_line(aes(x = date, y = periodicy, colour = "periodic")) +
266|     gg_scale_xy +
267|     scale_fill_manual(values = c("pi" = cols$nnpi),
268|                       labels = c("pi" = labs$nnpi)) +
269|     scale_colour_manual(values = c("base" = cols$baseyn, "nn" = cols$nn, "periodic" = cols$periodic,
270|                                   "test" = cols$yntest, "train" = cols$yn),
271|                          labels = c("base" = labs$base, "nn" = labs$nn, "periodic" = labs$periodic,
272|                                   "test" = labs$test, "train" = labs$train),
273|                          guide = guide_legend(override.aes = list(
274|                            shape = c("base" = 1, "nn" = 0, "periodic" = 1, "test" = 16, "train" =
275|                                   16)))) +
276|     yntheme()
277|   return(p)
278| }
279|
280| plot_multixn <- function(countrydat, modeldat, cols, labs){
281|   p <- ggplot(countrydat, binwidth = 0) +
282|     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
283|     geom_point(data = modeldat, aes(x = date, y = multixn, colour = cols$multixn)) +
284|     geom_line(data = modeldat, aes(x = date, y = multixn, colour = cols$multixn)) +
285|     gg_scale_xy +
286|     scale_fill_manual( name = "leg", values = cols$xn, labels = labs$xn) +
287|     scale_colour_manual(name = "leg", values = cols$multixn, labels = labs$multixn) +
288|     xntheme()
289|   return(p)
290| }
291|
292| plot_multiyn <- function(countrydat, modeldat, cols, labs){
293|   p <- ggplot(countrydat) +
294|     geom_point(aes(x = date, y = yn, colour = "blue")) +
295|     geom_line(aes(x = date, y = yn, colour = "blue")) +
296|     geom_point(data = modeldat, aes(x = date, y = multiyn, colour = "base"), shape = 1) +
297|     geom_line(data = modeldat, aes(x = date, y = multiyn, colour = "base")) +
298|     gg_scale_xy +
299|     scale_colour_manual(name = "leg", values = c("blue" = cols$yn, "base" = cols$multiyn),
300|                           labels = c("blue" = labs$yn, "base" = labs$multiyn),
301|                           guide = guide_legend(override.aes = list(
302|                             shape = c("blue"=1, "base" = 16)))) +
303|     yntheme()
304|   return(p)
305| }
306|
307| plot_multipernx <- function(countrydat, modeldat, cols, labs){
308|   p <- ggplot(countrydat, binwidth = 0) +
309|     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
310|     geom_point(data = modeldat, aes(x = date, y = multipxn, colour = "multi")) +
311|     geom_line(data = modeldat, aes(x = date, y = multipxn, colour = "multi")) +
312|     geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
313|     gg_scale_xy +
314|     guides(colour=guide_legend(ncol=1,nrow=2,byrow=TRUE),
315|            fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
316|     scale_fill_manual(values = cols$xn, labels = labs$xn) +
317|     scale_colour_manual(values = c("multi" = cols$multip, "x3" = cols$x3),
318|                           labels = c("multi" = labs$multipxn, "x3" = labs$x3)) +
319|     guides(colour = guide_legend(override.aes = list(shape = c("multi" = 16, "x3" = NA)))) +
320|     xntheme()
321|   return(p)
322| }
323|
324| plot_multiperyn <- function(countrydat, modeldat, cols, labs){
325|   p <- ggplot(countrydat) +
326|     geom_point(aes(x = date, y = yn, colour = "blue"), shape = 16) +
327|     geom_line(aes(x = date, y = yn, colour = "blue")) +

```

```

325|     geom_point(data = modeldat, aes(x = date, y = multipyn, colour = "mult"), shape = 1) +
326|     geom_line(data = modeldat, aes(x = date, y = multipyn, colour = "mult")) +
327|     gg_scale_xy +
328|     scale_colour_manual(name = "leg", values = c("blue" = cols$yn, "mult" = cols$multip), 
329|                           labels = c("blue" = labs$yn, "mult" = labs$multipyn),
330|                           guide = guide_legend(override.aes = list(
331|                             shape = c("blue"=16, "mult" = 1)))) +
332|     yntheme()
333|   return(p)
334}
335
336 plot_worldtotal <- function(dat){
337   p <- ggplot(dat, binwidth = 0) +
338     geom_bar(aes(x = date, y = new_cases), fill = wes_palettes$Zissou1[1], stat = "identity") +
339     scale_x_date(date_breaks = "1 month", date_labels = "%d-%b", expand = c(0,0))+
340     scale_y_continuous(expand = c(0,0)) +
341     ggtitle(wt_title) + xntheme()
342   return(p)
343 }
344
345 xntheme <- function(){
346   p <- theme(axis.text.x      = element_text(vjust = 0.5),
347             axis.title       = element_blank(),
348             axis.line        = element_line(),
349             panel.background = element_rect(fill = "grey"),
350             panel.grid        = element_line(colour = "darkgrey"),
351             panel.grid.major.x = element_blank(),
352             panel.grid.minor.x = element_blank(),
353             legend.title     = element_blank(),
354             legend.margin    = margin(0,4,0,4,"pt"),
355             legend.background = element_blank(),
356             legend.text.align = 0,
357             legend.box.background = element_rect(linetype="solid", colour ="darkgrey", size = 0
358 .1, fill = "white"),
359             legend.spacing    = unit(0, "cm"),
360             legend.key.size  = unit(0.8,"line"),
361             legend.key       = element_blank(),
362             legend.text      = element_text(size = 6),
363             legend.direction = "vertical",
364             legend.box       = "vertical",
365             legend.box.just  = 'left'
366             #legend.position  = "top"
367           )
368   return(p)
369 }
370
371 yntheme <- function(){
372   p <- theme(axis.text.x      = element_text(vjust = 0.5),
373             axis.title       = element_blank(),
374             axis.line        = element_line(),
375             panel.background = element_rect(fill = "grey"),
376             panel.grid        = element_line(colour = "darkgrey"),
377             panel.grid.major.x = element_blank(),
378             panel.grid.minor.x = element_blank(),
379             legend.title     = element_blank(),
380             legend.margin    = margin(4,0,0,4,"pt"),
381             legend.background = element_blank(),
382             legend.key       = element_blank(),
383             legend.box.background = element_rect(linetype="solid", colour ="darkgrey", size = 0
384 .1, fill = "white"),
385             legend.spacing    = unit(0, "cm"),
386             legend.key.size  = unit(0.8,"line"),
387             legend.text      = element_text(size = 6),
388             legend.direction = "vertical",
389             legend.box       = "vertical",
390             legend.box.just  = 'left'
391             #legend.position  = "top"
392           )
393   return(p)
394 }
395 gg_scale_xy <- list(
396   scale_x_date(date_breaks = "1 week", date_labels = "%d-%b", expand = c(0,0)),
397   scale_y_continuous(expand = c(0,0)))

```

```

397
398 error_plot <- function(){
399   p <- ggplot(data.frame(x = 0,y = 0)) +
400     geom_label(x = 0, y = 0, color = "red", size = 5 , fontface = "bold",
401                 label = "Error: Country data has nonpositive values") +
402     xlim(-1,1) + ylim(-1,1) +
403     theme(axis.line = element_blank(),
404           axis.text = element_blank(),
405           axis.ticks = element_blank(),
406           panel.grid = element_blank())
407   return(p)
408 }
```

Listing 8: Plotting functions

```

1 require(ggplot2)
2 require(forecast)
3 require(dplyr)
4 require(wesanderson)
5 require(gridExtra)
6 require(xtable)
7
8 owiddat <- read.csv("Data/owid-covid-data.csv")
9 owiddat$date <- as.Date(owiddat$date , tryFormats = c("%Y-%m-%d"))
10
11 plotslist <- list()
12
13 source("Code/covid-plotutils.R")
14 source("Code/covid-modelutils.R")
15
16 covidPlots <- function(country , dateBounds , data){
17   plots <- list()
18   dateBounds <- as.Date(dateBounds)
19   countryrows <- grep(country , data$location)
20   countrydat <- data.frame(date = data$date[countryrows],
21                             xn = data$new_cases[countryrows],
22                             yn = data$total_cases[countryrows])
23
24   forecastlen <- 14
25
26   prevcases <- countrydat$yn[countrydat$date == as.Date(dateBounds[1])-1]
#Specific dates
28   countrydattest <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= (dateBounds
      [2] + forecastlen),]
29   countrydat <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= dateBounds[2],]
30   latest_date <- dateBounds[2]
31   testBounds <- dateBounds[2] + c(1:forecastlen)
32
33   countrydesc <- paste0(country , " , " , dateBounds[1] , " to " , dateBounds[2])
34
35   cols <- list(
36     xn      = wes_palettes$Zissou1[1],
37     xntest  = wes_palettes$Darjeeling1[2],
38     yn      = wes_palettes$Darjeeling2[2],
39     yntest  = wes_palettes$BottleRocket2[4],
40     basexn  = wes_palettes$Darjeeling1[1],
41     baseyn  = wes_palettes$Darjeeling1[1],
42     x3      = wes_palettes$BottleRocket2[1],
43     Crn     = wes_palettes$FantasticFox1[2],
44     arima    = wes_palettes$Darjeeling1[4],
45     arimapi  = wes_palettes$Darjeeling1[3],
46     hw      = wes_palettes$Moonrise1[2],
47     hwpi    = wes_palettes$Moonrise1[1],
48     periodic = wes_palettes$GrandBudapest1[3] , "#magenta",
49     nn      = wes_palettes$FantasticFox1[4],
50     nnpi    = wes_palettes$GrandBudapest1[4]
51   )
52
53   labs <- list(
54     train   = "Training set",
55     test    = "Test set",
56     base    = "Base",
57     x3     = "Average x*(3)",
58     periodic = "Periodic",
59   )
60 }
```

```

59  hw      = "HoltWinters",
60  hwpi    = "HW 95% PI",
61  arimapi = "ARIMA 95% PI",
62  nnpi    = "NNAR 95% PI"
63 )
64
65 #Basic model: need a,b,q,r using ||x-x*|| and ||y-y*||
66 ##q - Any infected person becomes ill and infectious on the q-th day after infection.
67 ##a - During each day, each ill person at large infects on average a other persons.
68 ##b - During each day, a fraction b of ill people at large gets isolated
69
70 aseq   <- seq(from = 0.1, to = 1.5, length.out = 40)
71 bseq   <- seq(from = 0.1, to = 0.9, length.out = 40)
72 qseq   <- 6:8
73 normdat <- expand.grid(q = qseq, a = aseq, b = bseq)
74 abnorm <- apply(normdat, 1, function(x) norm(x, countrydat$xn))
75
76 normdat$abnorm <- abnorm
77 normdat$abnormn <- normalize(abnorm)
78
79 newnormdat <- normdat %>% top_n(abnorm, n = -0.07*nrow(.))
80
81 abnormy <- apply(newnormdat, 1, function(x) normy(x, countrydat$xn, countrydat$yn))
82 newnormdat$abnormyn <- normalize(abnormy)
83
84 newnormdat$combnorm <- 1*newnormdat$abnormn + 0.00*newnormdat$abnormyn
85
86 col_grad <- wes_palette("Zissou1", 20, type = "continuous")
87
88 #newnormdat$abnormy <- apply(newnormdat, 1, function(x) normy(x, countrydat$xn, countrydat$yn))
89
90 tileoptim <- newnormdat[which.min(newnormdat$abnorm), 1:3]
91 optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
92 plots[["combnorm"]]  
 <- ggplot(newnormdat, aes(x = a, y = b, z = abnorm)) +
93   geom_contour_filled() + labs(fill = "||x-x*||") +
94   scale_fill_brewer(palette = "Spectral")
95
96 q <- optimpars[1]
97 a <- optimpars[2]
98 b <- optimpars[3]
99
100 basexn <- basicmodx(countrydat$xn, optimpars, len = forecastlen)
101
102 modeldat <- data.frame(date = c(countrydat$date, latest_date + 1:forecastlen),
103                           basexn = basexn, baseyn = xntoyn(basexn) + prevcases,
104                           xn = countrydattest$xn, yn = countrydattest$yn)
105
106 modeldat$tgroup <- ifelse(modeldat$date <= dateBounds[2], "train", "test")
107
108 #Newtons method, r_1 = r_0 - f(r_0)/f'(r_0)
109 base_r_zero <- (a/b)^(1/(2*q))
110 base_r_one <- base_r_zero -basef(base_r_zero, optimpars)/basefprime(base_r_zero, optimpars)
111 base_r_one <- round(base_r_one,3)
112
113 trainrange <- 1:nrow(countrydat)
114 testrange <- nrow(countrydat) + 1:forecastlen
115
116 roptimpars <- round(optimpars,3)
117 xnorm <- norm(optimpars, countrydat$xn)
118 xnormtest <- modnorm(basexn[testrange], countrydattest$xn[testrange])
119
120 ynorm <- modnorm(countrydat$yn, modeldat$baseyn[1:length(countrydat$yn)])
121 ynormtest <- modnorm(modeldat$baseyn[testrange], countrydattest$yn[testrange])
122
123 plots[["basexn"]]  
 <- plot_basexn(modeldat, cols, labs)
124 plots[["baseyn"]]  
 <- plot_baseyn(modeldat, cols, labs)
125
126 summarydf <- data.frame(model = "Basic Recursion", xnorm = xnorm, ynorm = ynorm,
127                           xnormt = xnormtest, ynormt = ynormtest)
128
129 optimC <- optim(par = countrydat$xn[1], normC, method = "Brent",
130                   lower = 1, upper = 2*max(countrydat$xn[!is.na(countrydat$xn)]),
131                   x = basexn[trainrange], r = base_r_one$par
132

```

```

133 modeldat$Crn <- optimC*base_r_one^(1:nrow(modeldat))
134 labs$Crn <- list(bquote(Cr~n^*, r=~*(base_r_one)^*, C=~*(floor(optimC))))
135
136 plots[["Crn"]] <- plot_crn(modeldat, cols, labs)
137
138 mavgx1 <- movingavg(modeldat$xn[!is.na(modeldat$xn)])
139
140 modeldat$mavgx3 <- movingavg(mavgx1)
141 modeldat$mavgx3y <- xntoyn(modeldat$mavgx3) + prevcases
142
143 x3norm <- modnorm(modeldat$xn[trainrange], modeldat$mavgx3[trainrange])
144 x3normt <- modnorm(modeldat$xn[testrange], modeldat$mavgx3[testrange])
145 x3normy <- modnorm(modeldat$yn[trainrange], modeldat$mavgx3y[trainrange])
146 x3normyt <- modnorm(modeldat$yn[testrange], modeldat$mavgx3y[testrange])
147
148 plots[["mavgx3"]] <- plot_mavgx3(modeldat, cols, labs)
149
150 summarydf <- rbind(summarydf, data.frame(model = "Moving Average", xnorm = x3norm, ynorm = x3
151 normy,
152 xnormt = x3normt, ynormt = x3normyt))
153
154 #parameters of the form (ci, pi, ni)
155 ##an = a(1 + c1 sin(2pi/p1 (n - n1)))
156 c_1seq <- c_2seq <- seq(0.04, 0.2, length.out = 17)
157 n_1seq <- n_2seq <- 1:q
158 p_1seq <- p_2seq <- 1:q
159 normmdatp <- expand.grid(a = a,
160                           c1 = c_1seq, c2 = c_2seq,
161                           p1 = p_1seq, p2 = p_2seq,
162                           n1 = n_1seq, n2 = n_2seq)
163
164 pernorm <- apply(normmdatp, 1, function(x) normper(x, q = q, countrydat$xn))
165 normmdatp$pernorm <- pernorm
166
167 peroptim <- normmdatp[which.min(pernorm), 1:8]
168 pernorm <- normmdatp[which.min(pernorm), 9]
169
170 modeldat$periodic <- modxper(as.numeric(peropty), countrydat$xn, q = q, forecastlen)
171 modeldat$periodicy <- xntoyn(modeldat$periodic) + prevcases
172
173 pernormy <- modnorm(countrydat$yn, modeldat$periodicy[trainrange])
174 pernormt <- modnorm(modeldat$xn[testrange], modeldat$periodic[testrange])
175 pernormyt <- modnorm(modeldat$yn[testrange], modeldat$periodicy[testrange])
176
177 peropty <- round(as.numeric(peropty), 3)
178
179 perparamdat <- data.frame(
180   x = modeldat$date,
181   an = peropty[1] * (1 + peropty[3] * sin(2 * pi * (1:(nrow(modeldat)) - peropty[7]) / peropty[5])),
182   bn = peropty[2] * (1 + peropty[4] * sin(2 * pi * (1:(nrow(modeldat)) - peropty[8]) / peropty[6]))
183 )
184
185 plots[["perparam"]] <- ggplot(perparamdat) +
186   geom_line(aes(x = x, y = an, col = "a_n")) +
187   geom_line(aes(x = x, y = bn, col = "b_n")) +
188   geom_hline(aes(yintercept = peropty[1], col = "a"), linetype = "dashed") +
189   geom_hline(aes(yintercept = peropty[2], col = "b"), linetype = "dashed") +
190   xlab("date") + ylab("") +
191   scale_color_manual(values = wes_palettes$Rushmore1[c(3, 3, 5, 5)]) +
192   guides(colour = guide_legend(override.aes = list(linetype =
193     c("a" = "dashed", "a_n" = "solid", "b" = "dashed", "b_n" = "solid")))) +
194   xntheme() + theme(legend.position = "right")
195
196 #a, b, c1, c2, p1, p2, n1, n2
197
198 mathparamdat <- data.frame(a = peropty[1], b = peropty[2], q = q, r = base_r_one,
199                             c1 = peropty[3], p1 = peropty[5], n1 = peropty[7],
200                             c2 = peropty[4], p2 = peropty[6], n2 = peropty[8])
201
202 colnames(mathparamdat) <- c("a", "b", "q", "r",
203                             "c1", "p1", "n1", "c2", "p2", "n2")
204
205 plots[["periodic"]] <- plot_periodic(modeldat, cols, labs)

```

```

206 plots[["periodicity"]] <- plot_periodicity(modeldat, cols, labs)
207
208 summarydf <- rbind(summarydf, data.frame(model = "Periodic", xnorm = pernorm, ynorm = pernormy,
209                         xnormt = pernormt, ynormt = pernormyt))
210
211 #Statistical methods using timeseries forecasting
212
213 dat_ts <- ts(data = countrydat$xn, frequency = q)
214
215 g1 <- autoplot(dat_ts) + theme(axis.title = element_blank())
216 g2 <- ggAcf(dat_ts) + ggtitle("")
217 g3 <- ggPacf(dat_ts) + ggtitle("")
218
219 plots[["tsdisplay"]] <- grid.arrange(grobs = list(g1,g2,g3), layout_matrix = rbind(c(1, 1), c(2
220 , 3)))
221
222 plots[["residuals"]] <- gghistogram(log(dat_ts), add.normal = TRUE, bins=10)
223
224 plots[["tsdecompose"]] <- autoplot(decompose(dat_ts))
225
226 if(any(countrydat$xn <= 0)){
227   plots[["hw"]]  
plots[["hwy"]]  
error_plot()
228 } else{
229   hwmethod <- "additive"
230   #lambda=0 ensures values stay positive
231   hwfcst <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
232
233   hwfcst$fitted[1:q] <- countrydat$xn[1:q]
234   modeldat$hwxn <- c(hwfcst$fitted, hwfcst$mean)
235   modeldat$hwlo <- c(hwfcst$fitted, hwfcst$lower[,2])
236   modeldat$hwhi <- c(hwfcst$fitted, hwfcst$upper[,2])
237
238   modeldat$hwyn <- xntoyn(modeldat$hwxn) + prevcases
239   modeldat$hwylo <- xntoyn(modeldat$hwlo) + prevcases
240   modeldat$hwyhi <- xntoyn(modeldat$hwhi) + prevcases
241
242   hwnorm <- modnorm(countrydat$xn, hwfcst$fitted)
243   hwnormt <- modnorm(modeldat$xn[testrange], hwfcst$mean)
244   hwnormy <- modnorm(countrydat$yn, modeldat$hwyn[trainrange])
245   hwnormyt <- modnorm(modeldat$yn[testrange], modeldat$hwyn[testrange])
246
247   plots[["hw"]]  
plot_hw(modeldat, cols, labs)
248   plots[["hwy"]]  
plot_hwy(modeldat, cols, labs)
249 }
250
251 auto.fit <- auto.arima(dat_ts, lambda = 0) #keep values positive
252
253 getArmaModel <- function(arma, pdq = c(1,6,2), PDQ = c(3,7,4), s=5){
254   return(paste0("ARIMA(", paste0(arma[pdq], collapse = ","), ")",
255                 paste0(arma[PDQ], collapse = ","), ")[", arma[s], "]"))
256 }
257
258 summarydf <- rbind(summarydf, data.frame(model = "HoltWinters", xnorm = hwnorm,
259                         ynorm = hwnormy,
260                         xnormt = hwnormt, ynormt = hwnormyt))
261
262 arima.fcst <- forecast(auto.fit, level = c(80, 95), h = forecastlen)
263 arima.fcst$fitted[1:q] <- countrydat$xn[1:q]
264
265 arimanorm <- modnorm(countrydat$xn, arima.fcst$fitted)
266 arimanormt <- modnorm(modeldat$xn[testrange], arima.fcst$mean)
267
268 arimalabs <- getArmaModel(auto.fit$arma)
269 labs$arima <- arimalabs
270
271 modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
272 modeldat$arimalo <- c(auto.fit$fitted, arima.fcst$lower[,2])
273 modeldat$arimahi <- c(auto.fit$fitted, arima.fcst$upper[,2])
274
275 modeldat$arimayn <- xntoyn(modeldat$arimaxn) + prevcases
276 modeldat$arimaylo <- xntoyn(modeldat$arimalo) + prevcases
277 modeldat$arimayhi <- xntoyn(modeldat$arimahi) + prevcases
278
279 arimanormy <- modnorm(countrydat$yn, modeldat$arimayn[trainrange])

```

```

278 arimanormyt <- modnorm(modeldat$yn[testrange], modeldat$arimayn[testrange])
279 plots[["arima"]] <- plot_arima(modeldat, cols, labs)
280 plots[["arimay"]] <- plot_arimay(modeldat, cols, labs)
281 plots[["hwarima"]] <- plot_hwarima(modeldat, cols, labs)
282
283 summarydf <- rbind(summarydf, data.frame(model = "ARIMA", xn = arimanorm, yn = arimanormy
284 ,
285 xn = arimanormt, yn = arimanormyt))
286
287 nHidden <- max(1, floor(0.5*(1+auto.fit$arma[1]+auto.fit$arma[3])))
288 #Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.
289 nnfit <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda =
290 0, repeats = 20, maxit = 50)
291 nn.fcst <- forecast(nnfit, PI=TRUE, h = forecastlen)
292 labs$nn <- nnfit$method
293
294 nn.fcst$fitted[1:q] <- countrydat$xn[1:q]
295
296 modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
297 modeldat$nnlo <- c(nn.fcst$fitted, nn.fcst$lower[,2])
298 modeldat$nnhi <- c(nn.fcst$fitted, nn.fcst$upper[,2])
299
300 modeldat$nnyn <- xtoyn(modeldat$nnxn) + prevcases
301 modeldat$nnyo <- xtoyn(modeldat$nnlo) + prevcases
302 modeldat$nnyh <- xtoyn(modeldat$nnhi) + prevcases
303
304 nnnorm <- modnorm(countrydat$xn, nn.fcst$fitted)
305 nnnormtest <- modnorm(modeldat$xn[testrange], nn.fcst$mean)
306 nnnormy <- modnorm(countrydat$yn, modeldat$nnyn[trainrange])
307 nnnormytest <- modnorm(modeldat$yn[testrange], modeldat$nnyn[testrange])
308
309 plots[["nn"]] <- plot_nn(modeldat, cols, labs)
310 plots[["nny"]] <- plot_nny(modeldat, cols, labs)
311
312 summarydf <- rbind(summarydf, data.frame(model = "Neural Network", xn = nnnorm, yn = nnnormy,
313 xn = nnnormtest, yn = nnnormytest))
314 summarydf$xn <- as.integer(summarydf$xn)
315 summarydf$yn <- as.integer(summarydf$yn)
316 summarydf$xn <- as.integer(summarydf$xn)
317 summarydf$yn <- as.integer(summarydf$yn)
318 colnames(summarydf) <- c("model", "$||x-x^*||_{train}$", "$||y-y^*||_{train}$",
319 "$||x-x^*||_{test}$", "$||y-y^*||_{test}$")
320 return(list("plots" = plots, "summary" = summarydf, "desc" = countrydesc,
321 "dat" = modeldat, "mathmodelpars" = mathparamdat))
322 }
323
324 grigorDates <- c("2020-04-26", "2020-06-09")
325 datebounds <- list(
326 "Italy" = c("2021-01-02", "2021-02-16"),
327 "United States" = c("2021-01-06", "2021-02-16"),
328 "Ireland" = c("2021-01-12", "2021-02-16")
329 #Germany = c("2021-01-06", "2021-02-16")
330 #Netherlands = c("2021-01-06", "2021-02-16"),
331 #Spain = c("2021-01-06", "2021-02-16"),
332 #United Kingdom = c("2021-01-06", "2021-02-16")
333 )
334
335 owiddat <- owiddat[!is.na(owiddat$new_cases),]
336 totaldat <- owiddat[owiddat$location == "World",]
337 latest_date <- totaldat$date[nrow(totaldat)]
338 wt_title <- sprintf('Global Total =%as as at %s',
339 format(sum(totaldat$new_cases), big.mark=",", scientific=FALSE),
340 format(latest_date, "%B %d, %Y"))
341
342 plotslist[["WorldTotal"]][["xn"]] <- plot_worldtotal(totaldat)
343
344 for(country in names(datebounds)){
345 plotslist[[country]] <- covidPlots(country, datebounds[[country]], owiddat)
346 }

```

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
77multimodel <- c()
78multimodelp <- c()
79phasepars <- list()
80for(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  normmdat <- expand.grid(q = qseq, a = aseq, b = bseq)
91
92  if(i == 1)
93    abnorm <- apply(normmdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, start=TRUE), phasedat$xn))
94  else
95    abnorm <- apply(normmdat, 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 normmdat$abnorm <- abnorm
101
102 tileoptim <- normmdat[which.min(normmdat$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$xn, multimodel, optimpars, oldp = phasepars[[i-1]])
119
120 aseqper <- a*seq(from = 0.7, to = 1.3, length.out = 10)
121 bseqper <- b*seq(from = 0.7, to = 1.3, length.out = 10)
122 c_1seq <- c_2seq <- seq(0.04, 0.2, length.out = 10)
123 n_1seq <- n_2seq <- c(1,7)
124 p_1seq <- p_2seq <- 6:7
125 normmdatp <- expand.grid(a = aseqper, b = bseqper,
126                           c1 = c_1seq, c2 = c_2seq,
127                           p1 = p_1seq, p2 = p_2seq,
128                           n1 = n_1seq, n2 = n_2seq)
129
130 if(i == 1)
131   pernorm <- apply(normmdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1], phasedat$xn, multimodelp, start=TRUE), phasedat$xn))
132 else
133   pernorm <- apply(normmdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1], phasedat$xn, multimodelp)[length(multimodelp)+1:nrow(phasedat)], phasedat$xn))
134
135 normmdatp$pernorm <- pernorm
136
137 newnormmdatp <- normmdatp %>% top_n(pernorm, n = -25)
138
139 if(i == 1)
140   pernormy <- apply(newnormmdatp, 1, function(par) modnorm(beforeecumcases+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(
143   par, q = optimpars[1], phasedat$xn, multimodelp))[length(multimodelp)+1:nrow(phasedat)
144   ], phasedat$yn))
145 newnormdatp$pernormy <- pernormy
146 newnormdatp$combnorm <- normalize(newnormdatp$pernorm) + normalize(newnormdatp$pernormy)
147 peroptim <- as.numeric(newnormdatp[which.min(newnormdatp$combnorm), 1:8])
148 phasepars[[i]] <- c(phasepars[[i]], round(peroptim, 3))
149
150 if(i == 1 & i != length(dates))
151   multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE)
152 if(i == 1 & i == length(dates))
153   multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE, len=
154   forecastlen)
155 if(i > 1 & i == length(dates))
156   multimodelp <- multimodxper(peroptim, q = q, phasedat$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(
171   xn = list(bquote(.(country)*","~x[n]*"=new cases/day, actual till"~.(format.Date(latest_date
172   , "%d.%m.%Y")))),
173   yn = list(bquote(.(country)*","~y[n]*"=cumulative cases, actual till"~.(format.Date(latest_
174   date, "%d.%m.%Y"))))
175 )
176
177 multimodnormval <- modnorm(multimodel[1:length(countrydat$xn)], countrydat$xn)
178 multimodnormyval <- modnorm(beforecumcases+xntoyn(multimodel[1:length(countrydat$xn)]),
179   countrydat$yn)
180
181 b.roman <- function(x){ return(paste0(", as.roman(x), ") ) }
182
183 multilabd <- c()
184 for(i in 1:length(dates)){
185   multilabd <- c(multilabd, paste(b.roman(i), "from", format.Date(as.Date(as.character(dates[[i
186   ]][1])), "%d.%m")))
187 }
188 multilabd <- paste0(multilabd, collapse = "; ")
189
190 multilabp <- c()
191 for(i in 1:length(dates)){
192   multilabp <- c(multilabp, paste0(b.roman(i), " a=", phasepars[[i]][[2]], ", b=", phasepars[[i
193   ]][[3]]))
194 }
195 multilabp <- paste0(multilabp, paste0("; q=", phasepars[[i]][[1]]), collapse = "; ")
196
197 labs$multixn <- list(bquote("base"~(length(dates))*"-phase model"~x[n]*"=new cases/day: "*
198   .(multilabd)*
199   ";" ||x*-x||="*(.multimodnormval)*" "*
200   .(multilabp)))
201 labs$multiyn <- list(bquote("base"~(length(dates))*"-phase model"~y[n]*"=cumulative cases: "*
202   .(multilabd)*
203   ";" ||y*-y||="*(.multimodnormyval)*" "*
204   .(multilabp)))
205
206 mavgx1 <- movingavg(countrydat$xn[!is.na(countrydat$xn)])
207 countrydat$mavgx3 <- movingavg(mavgx1)
208
209 x3norm <- modnorm(countrydat$xn, countrydat$mavgx3)
210 labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*(x3norm)))

```

```

208 modeldat <- data.frame(date = c(countrydat$date, as.Date(latest_date) + 1:forecastlen),
209                         multixn = multimodel,
210                         multiyn = beforecumcases + xntoy(multimodel),
211                         multipxn = multimodelp,
212                         multipyn = beforecumcases + xntoy(multimodelp))
213
214 plots[["xn"]] <- plot_multixn(countrydat, modeldat, cols, labs)
215 plots[["yn"]] <- plot_multiyn(countrydat, modeldat, cols, labs)
216
217 multilabp <- c()
218 for(i in 1:length(dates)){
219   multilabp <- c(multilabp, paste0(b.roman(i), " a=", phasepars[[i]][[2]], ", b=", phasepars
220     [[i]][[3]]))
221 }
222 multilabp <- paste0(multilabp, paste0("; q=", phasepars[[i]][[1]])), collapse = "; ")
223
224 labs$multipxn <- list(bquote(~(length(dates))~"-phase model"~x[n]~"=new cases/day: "
225   *
226   .(multilabd)*
227   "; ||x-x||="~.(multimodnormval)*"~"
228   .(multilabp)))
229 labs$multipyn <- list(bquote(~(length(dates))~"-phase model"~y[n]~"=cumulative cases
230   :
231   *
232   .(multilabd)*
233   "; ||y-y||="~.(multimodnormyval)*"~"
234   .(multilabp)))
235
236 plots[["perxn"]] <- plot_multipernx(countrydat, modeldat, cols, labs)
237 plots[["peryyn"]] <- plot_multiperyn(countrydat, modeldat, cols, labs)
238
239 return(plots)
240 }
241 for(country in names(multidates)){
242   multilist[[country]] <- multiphasePlots(country, multidates[[country]], owiddat)
243 }
```

Listing 10: multi-phase models

```

1 # load required libraries
2 library(ggplot2)
3 library(rgdal)
4 library(raster)
5 library(wesanderson)
6 library(dplyr)
7
8 countyplotlist <- list()
9
10 # read the shape files
11 setwd("~/GitHub/TCD_FinalYearProject/Data/")
12 countyshp <- readOGR("counties/counties.shp")
13 worldshp <- readOGR("world/world.shp")
14
15 # read the county case data
16 countycases <- read.csv("Data/Covid19CountyStatisticsHPSCIreland.csv")
17 owiddat <- read.csv("Data/owid-covid-data.csv")
18 owiddat$date <- as.Date(owiddat$date)
19 countycases$TimeStamp <- as.Date(countycases$TimeStamp)
20
21 # just latest date
22 latest_date <- countycases$TimeStamp[nrow(countycases)]
23 latest_dat <- countycases[countycases$TimeStamp == latest_date,]
24 fortnightbefore_dat <- countycases[countycases$TimeStamp == latest_date-13,]
25 latest_dat$ConfirmedCovidCases <- latest_dat$ConfirmedCovidCases - fortnightbefore_dat$ConfirmedCovidCases
26
27 # make shape data ggplot-friendly
28 countyshp@data$id <- rownames(countyshp@data)
29 countyshp.points <- fortify(countyshp, region="id")
30 counties <- inner_join(countyshp.points, countyshp@data, by="id")
31 counties$CountyName <- gsub("County ", "", counties$NAME_EN)
32
33 # join case numbers for latest date to county data, in order to colour nicely
```

```

33| countycase_map <- left_join(counties, latest_dat, by=c("CountyName" = "CountyName"))
34|
35| # color gradient
36| col_grad <- wes_palette("Zissou1", 20, type = "continuous")
37|
38| #theme for map plotting
39| map_theme <- function(){
40|   p<- theme(axis.title      = element_blank(),
41|             axis.text       = element_blank(),
42|             axis.ticks      = element_blank(),
43|             panel.background = element_blank(),
44|             legend.title    = element_blank(),
45|             legend.background = element_blank())
46|   return(p)
47| }
48|
49| # county plots
50| countyplotlist[["rep"]] <- ggplot(countycase_map) +
51|   aes(long, lat, group=group, fill=PopulationProportionCovidCases) +
52|   geom_polygon(colour="grey40") + labs(fill = "Cases per 100k") +
53|   scale_fill_gradientn(colours = col_grad) +
54|   geom_text(data = latest_dat, aes(x = Long, y = Lat, label = floor(
55|     PopulationProportionCovidCases)), inherit.aes = FALSE) +
56|   ggtitle("Cases in Ireland per 100,000 population by county",
57|           subtitle = paste("Cumulative, up to", format.Date(latest_date, "%B %d, %Y"))) +
58|   map_theme() + theme(legend.position = c(0.25,0.87))
59|
60| countyplotlist[["fourteendaycases"]] <- ggplot(countycase_map) +
61|   aes(long, lat, group=group, fill=ConfirmedCovidCases) +
62|   geom_polygon(colour="grey40") + labs(fill = "Cases") +
63|   scale_fill_gradientn(colours = col_grad) +
64|   geom_text(data = latest_dat, aes(x = Long, y = Lat, label = floor(ConfirmedCovidCases)),
65|             inherit.aes = FALSE) +
66|   ggtitle("Cases in Ireland by county",
67|           subtitle = paste("From", format.Date(latest_date-13, "%B %d, %Y"),
68|                             "to", format.Date(latest_date, "%B %d, %Y"))) +
69|   map_theme() + theme(legend.position = c(0.25,0.87))
70|
71| countyplotlist[["names"]] <- ggplot(countycase_map) +
72|   aes(long, lat, group=group, fill=PopulationProportionCovidCases) +
73|   geom_polygon(colour="grey40") + labs(fill = "Cases per 100k") +
74|   scale_fill_gradientn(colours = col_grad) +
75|   geom_text(data = latest_dat, aes(x = Long, y = Lat, label = CountyName), size=3,inherit.aes =
76|             FALSE) +
77|   ggtitle("Cases in Ireland per 100,000 population by county",
78|           subtitle = paste("Cumulative, up to", format.Date(latest_date, "%B %d, %Y"))) +
79|   map_theme() + theme(legend.position = c(0.25,0.87))
80|
81| countyplotlist[["blank"]] <- ggplot(countycase_map) +
82|   aes(long, lat, group=group, fill=PopulationProportionCovidCases) +
83|   geom_polygon(colour="grey40") + labs(fill = "Cases per 100k") +
84|   scale_fill_gradientn(colours = col_grad) +
85|   ggtitle("Cases in Ireland per 100,000 population by county",
86|           subtitle = paste("Cumulative, up to", format.Date(latest_date, "%B %d, %Y"))) +
87|   map_theme() + theme(legend.position = c(0.25,0.87))
88|
89| # just latest date
90| latest_date <- as.Date(owiddat$date[nrow(owiddat)-1], tryFormats = c("%Y-%m-%d"))
91| latest_dat <- owiddat[owiddat$date == latest_date,]
92| fortnightRows <- owiddat$date >= latest_date-13 & owiddat$date <= latest_date
93| fortnightCases <- aggregate(owiddat$new_cases_per_million[fortnightRows],
94|                               by=list(owiddat$location[fortnightRows]), function(x) sum(x[!is.na(
95|                               x)]))
96| colnames(fortnightCases) <- c("location", "fortnight_cases_per_million")
97| latest_dat <- left_join(latest_dat, fortnightCases, by=c("location" = "location"))
98|
99| # make shape data ggplot-friendly
100| worldshp@data$id <- rownames(worldshp@data)
101| worldshp.points <- fortify(worldshp, region="id")
102| countries <- inner_join(worldshp.points, worldshp@data, by="id")
103|
104| # join case numbers for latest date to country data, in order to colour nicely
105| world_map <- left_join(countries, latest_dat, by=c("CNTRY_NAME" = "location"))

```

```

103 world_map <- world_map[world_map$lat >= -75,]
104
105 worldplot <- list()
106
107 worldplot[["blank"]] <- ggplot(world_map) +
108   aes(long, lat, group=group, fill=fortnight_cases_per_million) +
109   geom_polygon(colour="grey40") + labs(fill = "Cases per million") +
110   scale_fill_gradientn(colours = col_grad) +
111   ggtitle("Cases per 1 million population by country",
112         subtitle = paste("From", format.Date(latest_date-13, "%B %d, %Y"), "to", format.Date(
113           latest_date, "%B %d, %Y")) +
114   map_theme() + theme(plot.margin = margin(0, 0, 0, 0, "cm"),
115     legend.position = c(0.1,0.4))
116
117 worldplot[["cumulative"]] <- ggplot(world_map) +
118   aes(long, lat, group=group, fill=total_cases_per_million) +
119   geom_polygon(colour="grey40") + labs(fill = "Cases per million") +
120   scale_fill_gradientn(colours = col_grad) +
121   ggtitle("Total cases per 1 million population by country",
122         subtitle = paste("Up to", format.Date(latest_date, "%B %d, %Y"))) +
123   map_theme() + theme(plot.margin = margin(0, 0, 0, 0, "cm"),
124     legend.position = c(0.1,0.4))
125
126 europe_map <- world_map[world_map$long >= -20 & world_map$long <= 40,]
127 europe_map <- europe_map[europe_map$lat >= 35 & europe_map$lat <= 75,]
128
129 worldplot[["europe"]] <- ggplot(europe_map) +
130   aes(long, lat, group=group, fill=fortnight_cases_per_million) +
131   geom_polygon(colour="grey40") + labs(fill = "Cases per million") +
132   scale_fill_gradientn(colours = col_grad) +
133   ggtitle("Total cases per 1 million population by country",
134         subtitle = paste("From", format.Date(latest_date-13, "%B %d, %Y"), "to", format.Date(
135           latest_date, "%B %d, %Y")) +
136   map_theme() + theme(plot.margin = margin(0, 0, 0, 0, "cm"),
     legend.position = c(0.1,0.3))

```

Listing 11: Geospatial/Map plots

```

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
7 comparelist <- list()
8 comparepairs <- list("IreUK" = c("Ireland", "United Kingdom"),
9                      "IreUS" = c("Ireland", "United States"),
10                     "IreIta" = c("Ireland", "Italy"),
11                     "IreNl" = c("Ireland", "Netherlands"),
12                     "IreIce" = c("Ireland", "Iceland"))
13
14 compDates <- c("2020-12-15", "2021-02-15")
15
16 compare_theme <- function(){
17   p <- theme(axis.text.x      = element_text(vjust=0.5),
18             axis.line       = element_line(),
19             panel.background = element_rect(fill = "grey"),
20             panel.grid       = element_line(colour = "darkgrey"),
21             panel.grid.major.x = element_blank(),
22             panel.grid.minor.x = element_blank(),
23             legend.key      = element_blank(),
24             legend.key.size = unit(0.8,"line"),
25             legend.text     = element_text(size = 8))
26
27   return(p)
28 }
29
30 compareplots <- function(dat, countries, dates){
31   getcountrydat <- function(dat, country, dates){
32     cind <- grep(country, dat$location)
33     countrydat <- data.frame(date = dat$date[cind], casemp = dat$new_cases_per_million[cind])
34     #Specific dates
35     countrydat <- countrydat[countrydat$date >= dates[1] & countrydat$date <= dates[2],]
36     return(countrydat)
37   }

```

```

36}
37
38 countryA    <- countries[1]
39 countryB    <- countries[2]
40 countryAdat <- getcountrydat(owiddat, countryA, compDates)
41 countryBdat <- getcountrydat(owiddat, countryB, compDates)
42
43 compcols <- wes_palettes$Darjeeling1[1:2]
44 p <- ggplot(countryAdat) +
45   geom_point(data = countryAdat, aes(x = date, y = casepm, colour = countryA)) +
46   geom_line(data = countryAdat, aes(x = date, y = casepm, colour = countryA)) +
47   geom_point(data = countryBdat, aes(x = date, y = casepm, colour = countryB)) +
48   geom_line(data = countryBdat, aes(x = date, y = casepm, colour = countryB)) +
49   gg_scale_xy + ylab(" ") + xlab(" ") + labs(colour = "Country")+
50   scale_colour_manual(values = compcols) + compare_theme()
51
52 return(p)
53
54 for(pair in names(comparepairs)){
55   comparelist[[pair]] <- compareplots(owiddat, comparepairs[[pair]], compDates)
56 }
57
58 compdaterange <- owiddat$date >= compDates[1] & owiddat$date <= compDates[2]
59 countriescmp <- c("Ireland", "United Kingdom", "United States", "Italy", "Germany")
60 comparelist[["all"]] <- ggplot(owiddat[owiddat$location %in% countriescmp & compdaterange,]) +
61   geom_point(aes(x = date, y = new_cases_per_million, colour = location)) +
62   geom_line(aes(x = date, y = new_cases_per_million, colour = location)) +
63   gg_scale_xy + ylab(" ") + xlab(" ") + labs(colour = "Country")+
64   scale_colour_manual(values = wes_palette("Darjeeling1", length(countriescmp))) +
65   compare_theme()

```

Listing 12: Country Comparison plots

```

1 setwd("~/GitHub/TCD_FinalYearProject")
2 source("Code/covid-main.r")
3 source("Code/covid-multimodel.r")
4 source("Code/covid-mapplots.r")
5 source("Code/covid-compare.r")
6
7 for(country in names(plotslist)){
8   for(p in names(plotslist[[country]][["plots"]])){
9     ggsave(filename = paste0(country, "-", p, "-traintest.pdf"),
10       plot      = plotslist[[country]][["plots"]][[p]],
11       path      = "./Plots",
12       height    = 10,
13       width     = 15,
14       units     = "cm"
15     )
16   }
17   if(!is.null(plotslist[[country]][["summary"]])){
18     sumtable <- xtable(plotslist[[country]][["summary"]], type = "latex",
19                         caption = plotslist[[country]][["desc"]],
20                         label = paste0("fig:", country, "summarydf"))
21     print(sumtable, sanitize.text.function=function(x){x},
22           table.placement="H",
23           file = paste0("Report/Chapters/", country, "-summarydf.tex"))
24   }
25   if(!is.null(plotslist[[country]][["mathmodelpars"]])){
26     sumtable <- xtable(plotslist[[country]][["mathmodelpars"]], type = "latex",
27                         caption = paste0(country, ": mathematical models parameters"),
28                         label = paste0("fig:", country, "mathmodelpars"))
29     print(sumtable, sanitize.text.function=function(x){x},
30           table.placement="H",
31           file = paste0("Report/Chapters/", country, "-mathmodelpars.tex"))
32   }
33 }
34
35 for(country in names(multilist)){
36   for(p in names(multilist[[country]])){
37     ggsave(filename = paste0(country, "-", p, "mult.pdf"),
38       plot      = multilist[[country]][[p]],
39       path      = "Plots",
40       height    = 10,
41       width     = 16,

```

```

42         units    = "cm"
43     }
44 }
45 }
46
47 for(pair in names(comparelist)){
48   ggsave(filename = paste0("compare-", pair, ".pdf"),
49           plot    = comparelist[[pair]],
50           path    = "Plots",
51           height  = 10,
52           width   = 16,
53           units   = "cm"
54   )
55 }
56
57 for(p in names(countyplotlist)){
58   ggsave(filename = paste0("county-", p, ".pdf"),
59           plot    = countyplotlist[[p]],
60           path    = "Plots",
61           height  = 14,
62           width   = 14,
63           units   = "cm"
64   )
65 }
66
67 for(p in names(worldplot)){
68   ggsave(filename = paste0("world-", p, ".pdf"),
69           plot    = worldplot[[p]],
70           path    = "Plots",
71           height  = 14,
72           width   = 20,
73           units   = "cm"
74   )
75 }

```

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](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>.