Modelling the Spread of COVID-19 Cases

TCD Final Year Project

Anthony Gibbons

Project Supervisor: Athanasios Georgiadis

April 1, 2021

Abstract

We construct various models of the Covid-19 pandemic for multiple countries in early 2021. We first construct simple model of the epidemic by using a recurrence equation. We also add a periodic complexity to these simpler models. These recurrence relation models are based on the paper *Mathematical riddles of COVID-19* by Grigorian [1], who studied the outbreak in the first half of 2020. We then use more statistical methods, modelling using time series forecasting methods such as Holt-Winters, ARIMA and Neural Network methods. All of this is with the aim of predicting the course of the pandemic within each chosen country. We implement both new and existing algorithms in `R` and visualise the results.

Keywords— ARIMA, Autoregressive model, COVID-19, Coronavirus, Data analysis, Forecasting, Holt-Winters, Mathematical model, Neural Network, NNAR, Pandemic, Parameter estimation, Prediction, SARS-CoV-2, Statistical model, Time series data.

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: 01/04/2021

Anthony Gibbons

Contents

List of Figures

List of Code

List of Tables

1 Introduction

The Coronavirus disease (COVID-19) was first characterized by the World Health Organisation as pandemic on 11th March 2020 [2]. The outbreak has affected almost every aspect of human life throughout 2020, and is expected to continue for much of 2021.

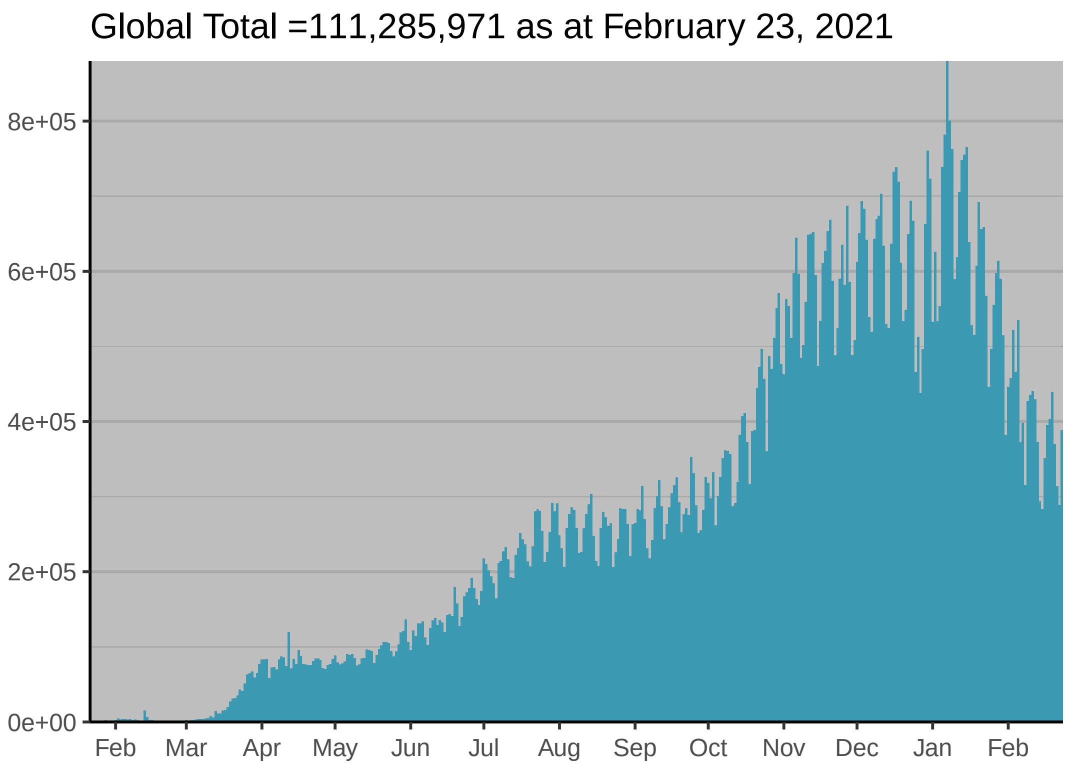

Figure 1: Daily Cases Globally

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

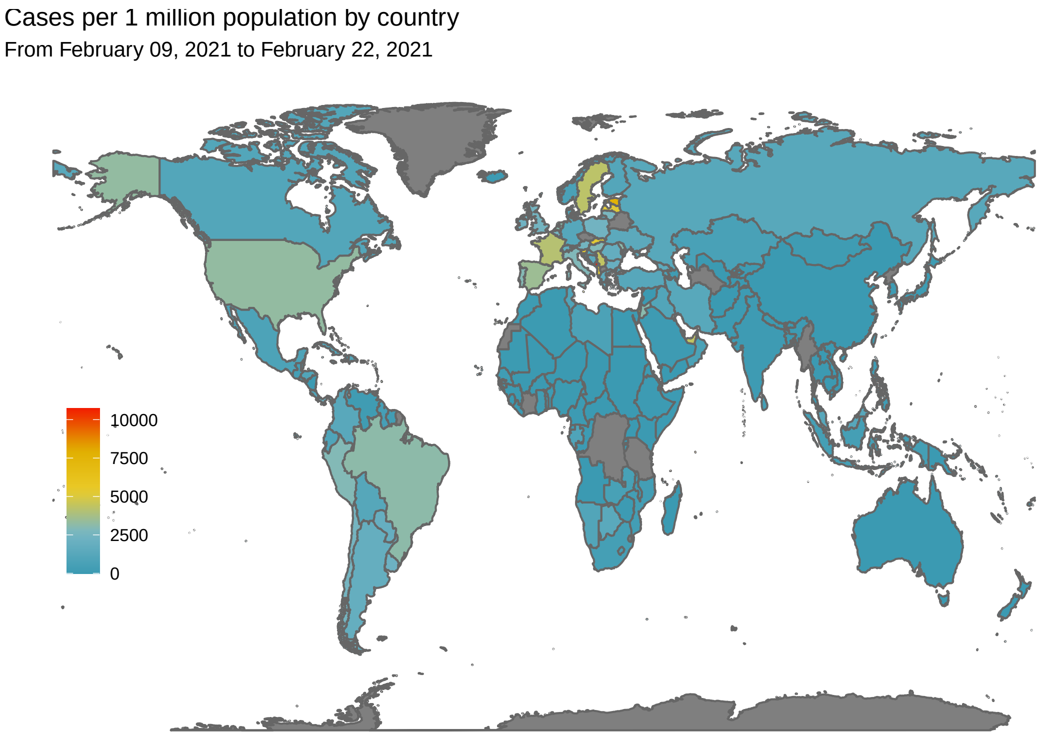

Figure 2: Cases per 1 million population, World

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

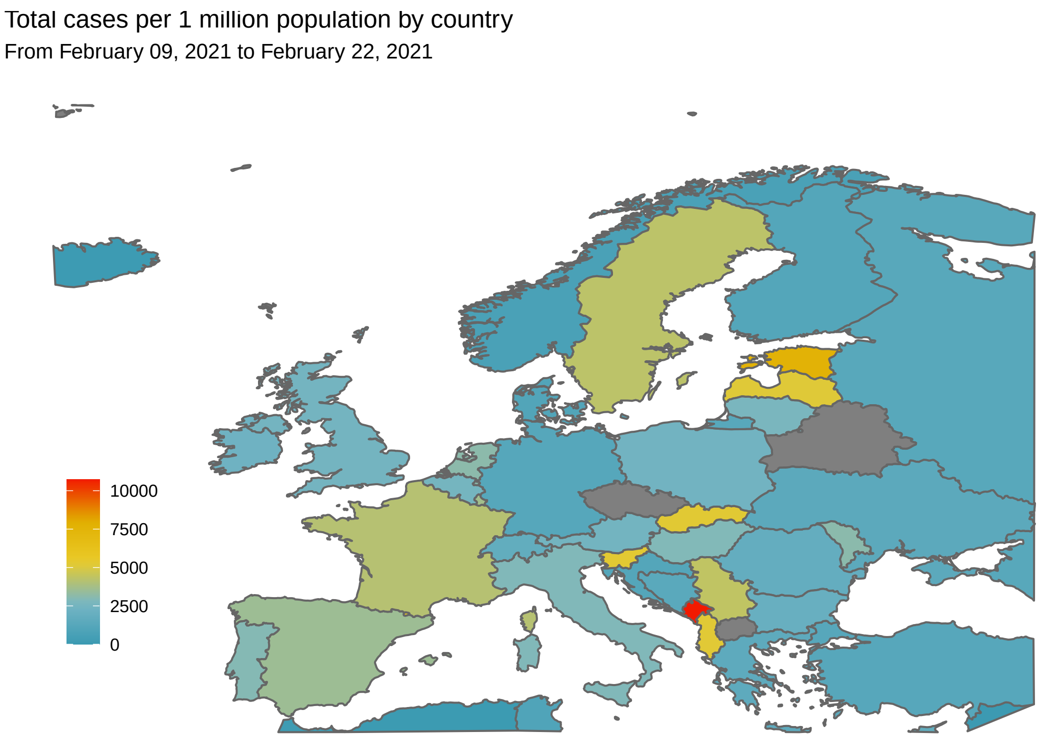

Figure 3: Cases per 1 million population, Europe

More locally, we see that Ireland also has a clear variation in concentration of cases to date, with Donegal and much of Leinster experiencing sometimes twice as many cases per 100,000 population as the rest of the country.

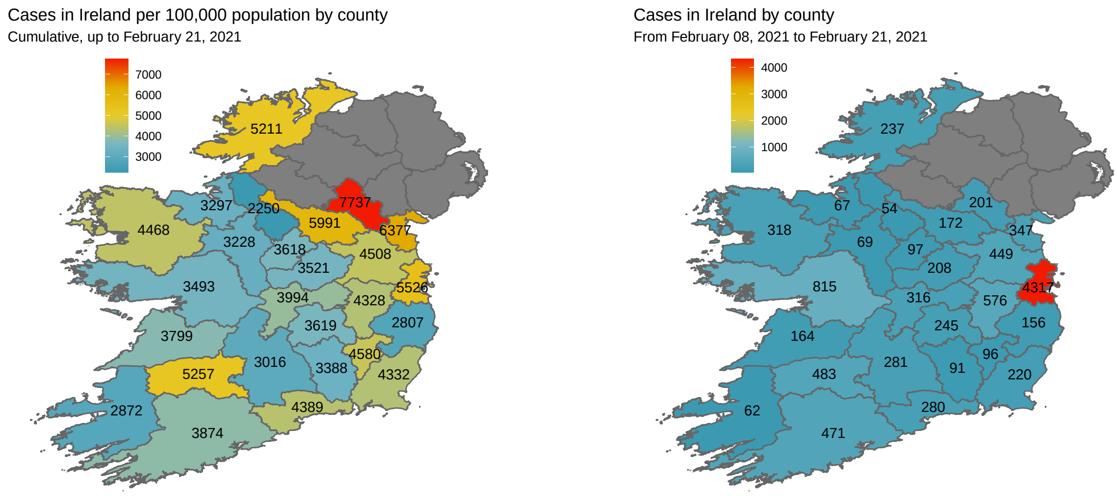

Figure 4: Situation by County, Ireland

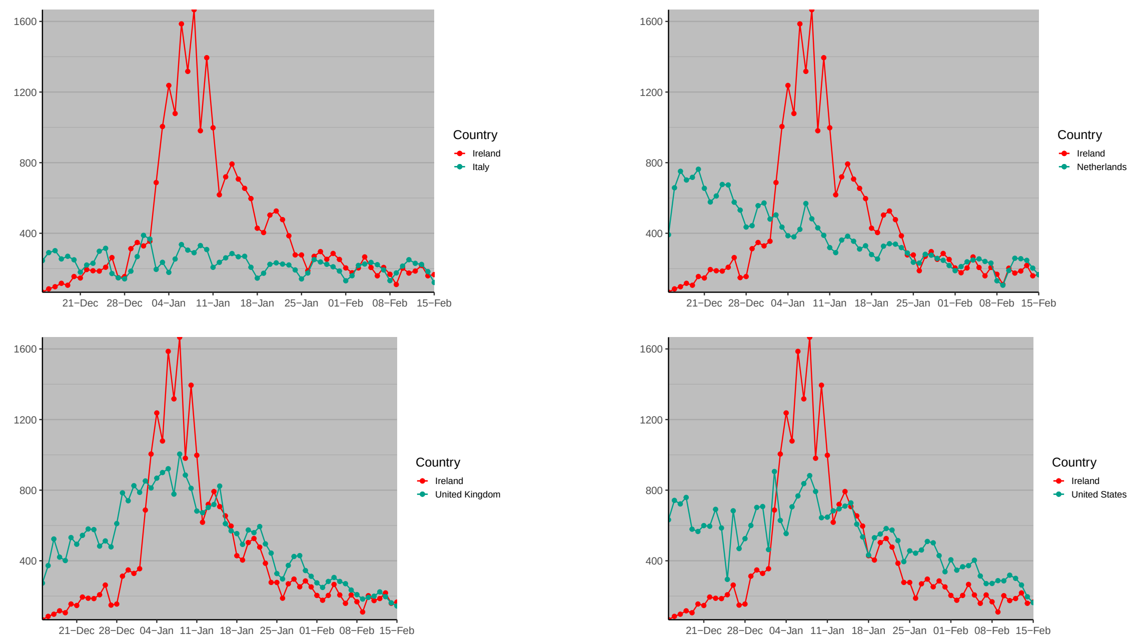

Figure 5: Individual Comparison of Ireland with various countries, daily cases per million of the population

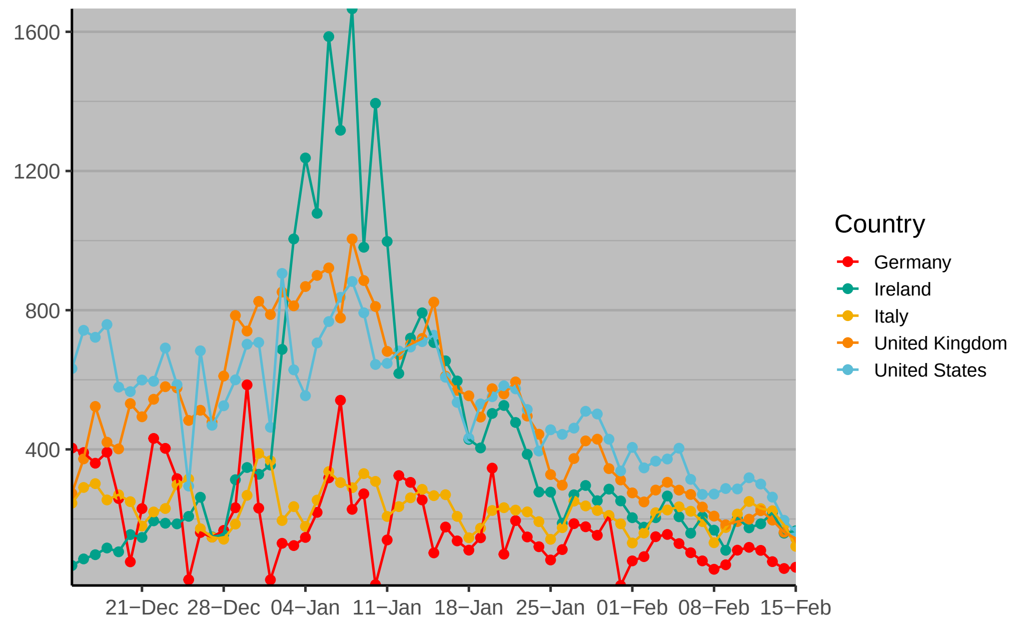

Figure 6: Comparison of Ireland with various countries, daily cases per million of the population

1.1 Previous work on Covid-19 identification and modeling

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

The work involving *external* factors such as government travel restrictions or full lockdowns, carried out by [3], 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 [4], which relies on phone–based surveys, certainly has the long-run potential to keep the public informed and hopefully reduce the the severity of outbreaks in areas where the app is widely used. One drawback of the initial model was its estimation of the peak of case numbers (which is notoriously difficult to predict) being the highest value in the case numbers so far. This does not take into account the shape of many time series during the early stages of the virus outbreak. For example, a strictly increasing time series would have its maximum at the latest time.

1.2 Key aims

This project is based on the work in [1], 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 [5].

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

1.3 Limitations

There are some limitations to the models used in this paper, some of which we can quickly see after the visualisation of the models versus actual cases reported:

- The models seem to fit well when there is an obvious trend in one direction. They do not perform well at or past a peak in case numbers.

- Countries may see their testing capabilities strained or varied, and so some reported numbers may be attributed to a later date than usual.
- Vaccinations began in late 2020 to early 2021 and will likely affect the spread of the virus.
- The modeling is done on a univariate time series, whereas external factors such as population densities and other demographics. Spatial data would then be needed to model the disease spread, but this is beyond the scope of the paper.

2 Mathematical Models

As per the base and periodic models shown in [1].

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.

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

(B) During each day, a proportion $b \in (0,1)$ of ill people loose gets isolated (hospitalized or otherwise) and withdrawn from a further spread of the epidemic.

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

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 models (and even some of the statistical models) make use of *recurrence equations*, which have some correspondence to differential equations [6].

2.1.2 Notation

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

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

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

Recurrence relation for z_n:

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.

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

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

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

It is necessary to let the model equal the actual data for the first $q+1$ days, as the recurrence equation needs initial data:

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

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

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

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

We choose N to be the length of the data (number of actual observations x_n^*) in general.

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

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

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

The proofs and results are in A.

2.1.3 How to select the best model

The closest-fiting model will be one that minimizes (5), which can bee seen using contour plots below.

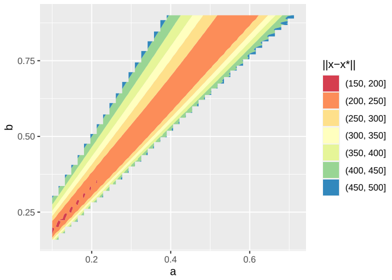

Figure 7: Contours, Ireland

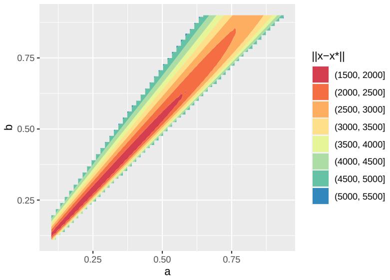

Figure 8: Contours, Italy

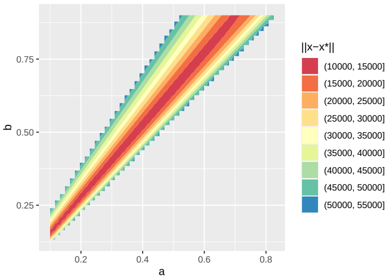

Figure 9: Contours, United States

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 $x_{N+1}, \ldots, x_{N+m}$ defined recursively using 3.2.1.

2.1.5 Implementation in R

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

Listing 1: Algorithm for Base Model

2.1.6 Plots

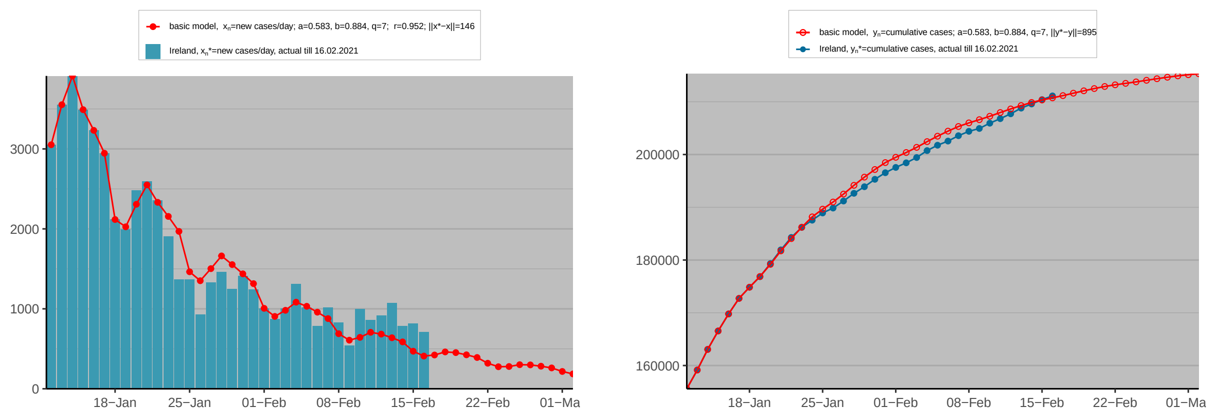

Figure 10: Basic model, Ireland

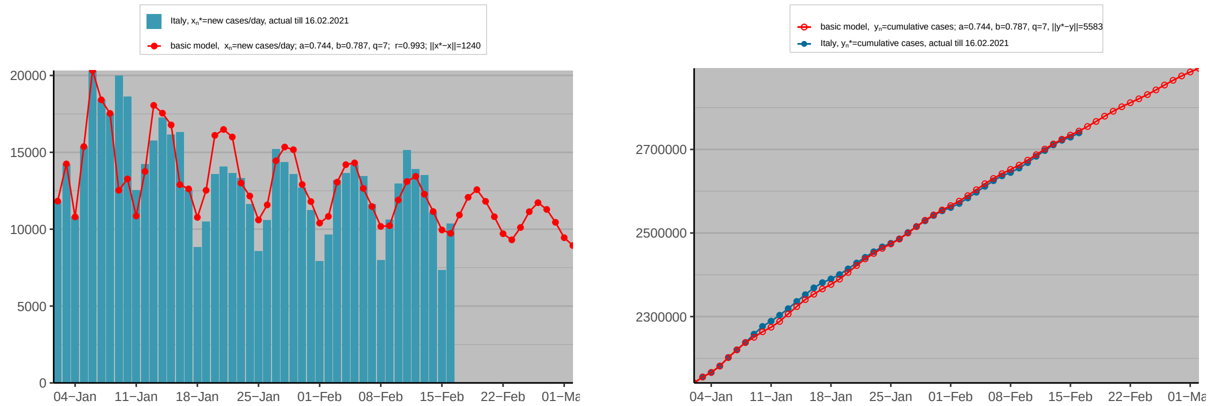

Figure 11: Basic model, Italy

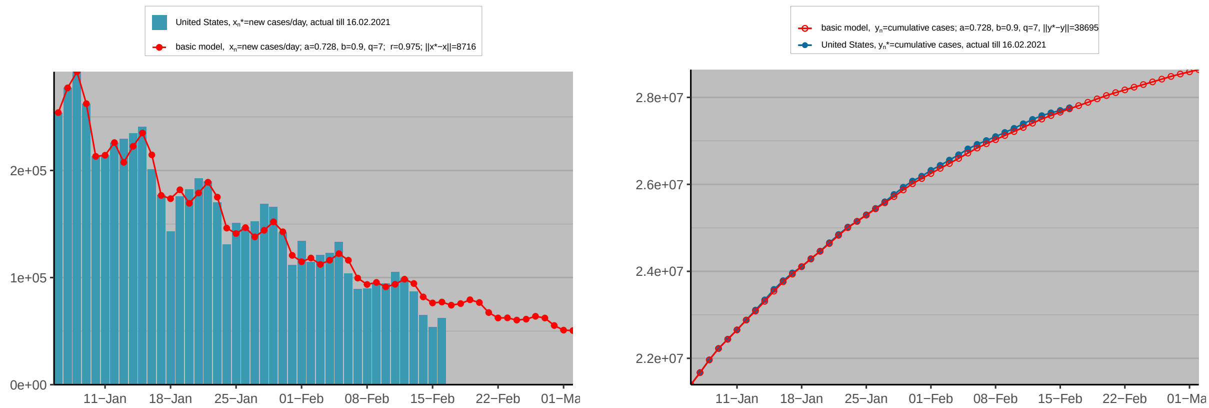

Figure 12: Basic model, United States

2.2 Limiting curve Cr^n

The second result in 1 is the equation (20), which defines the limiting behavior of the recurrence relation (3.2.1).

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

Listing 2: Algorithm for Limiting Curve

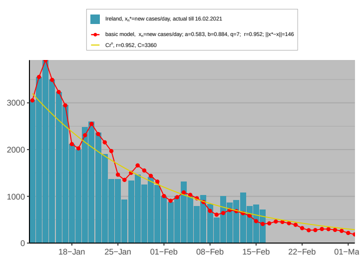

Figure 13: Limiting curve Cr^n, Ireland

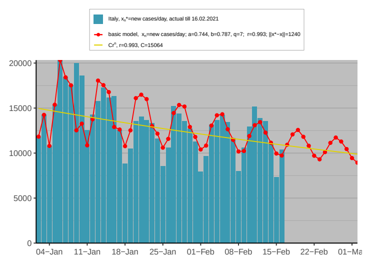

Figure 14: Limiting curve Cr^n, Italy

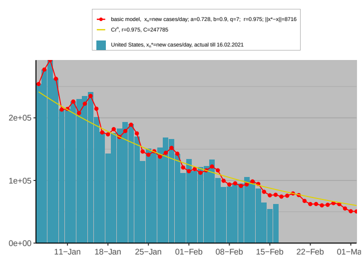

Figure 15: Limiting curve Cr^n, United States

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

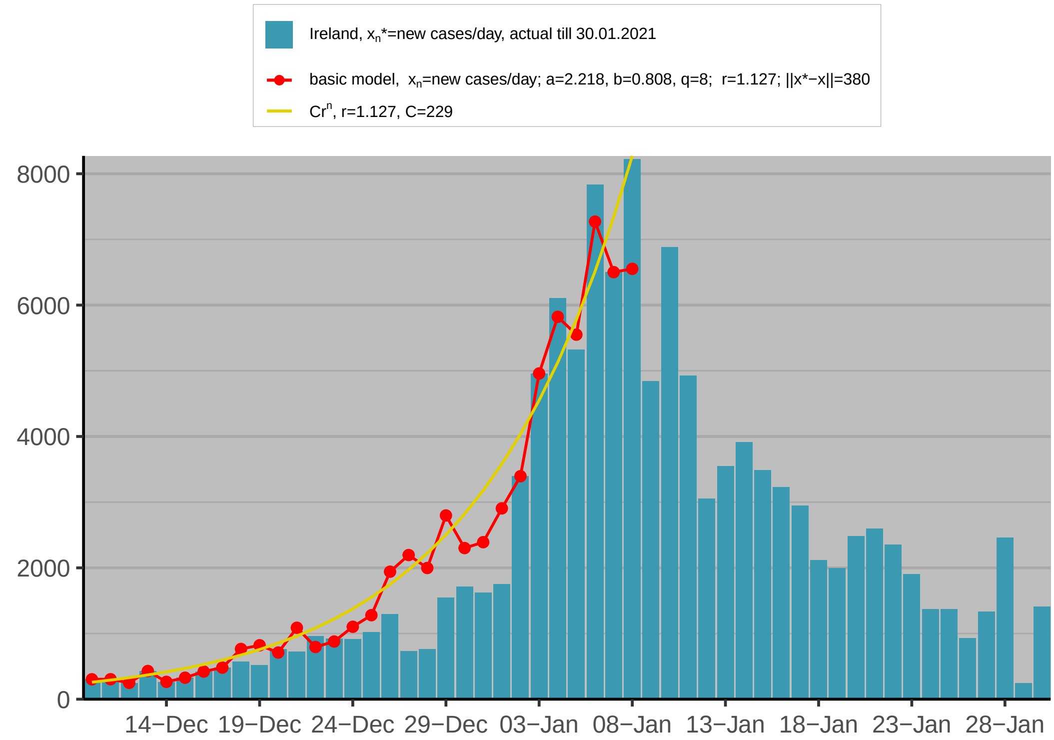

Figure 16: Limiting Curve Growing Exponentially, Ireland

2.3 Moving average

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

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

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

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

And then

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

is the 7-day moving average of cases.

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

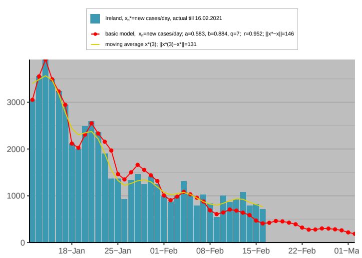

Figure 17: Moving average $x_n^*(3)$, Ireland

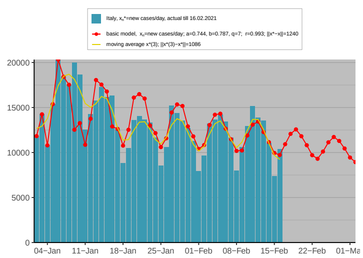

Figure 18: Moving average $x_n^*(3)$, Italy

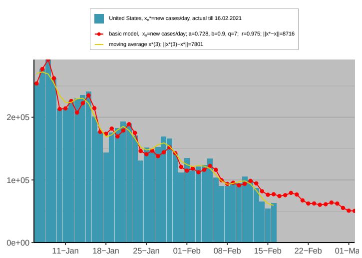

Figure 19: Moving average $x_n^*(3)$, United States

2.4 Periodic model

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

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

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

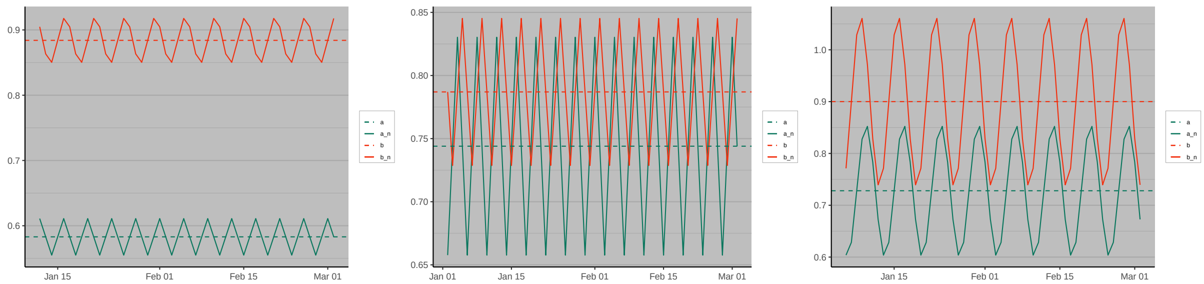

Figure 20: Oscillating a and b parameters, Ireland, Italy and United States

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

While Grigorian's paper [1] 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

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

Listing 3: Algorithm for Periodic Model

2.4.2 Plots

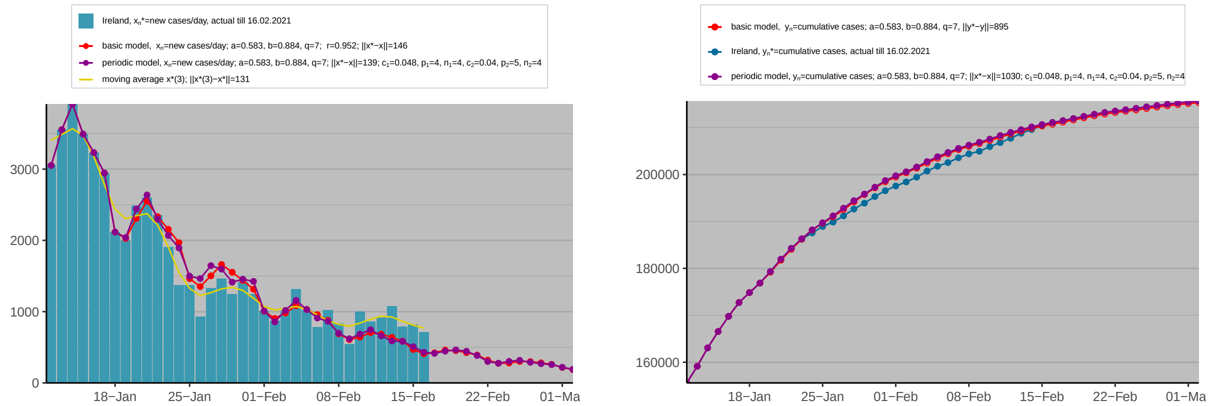

Figure 21: Periodic model, Ireland

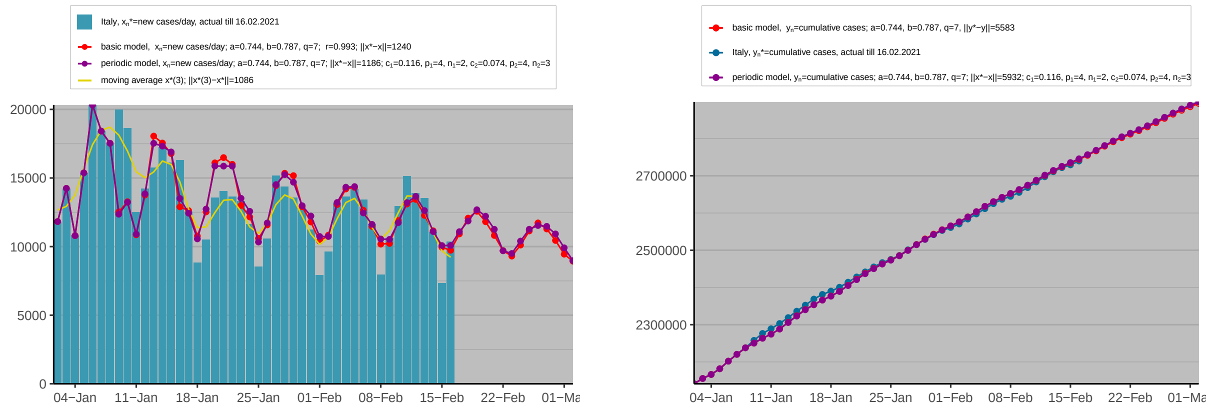

Figure 22: Periodic model, Italy

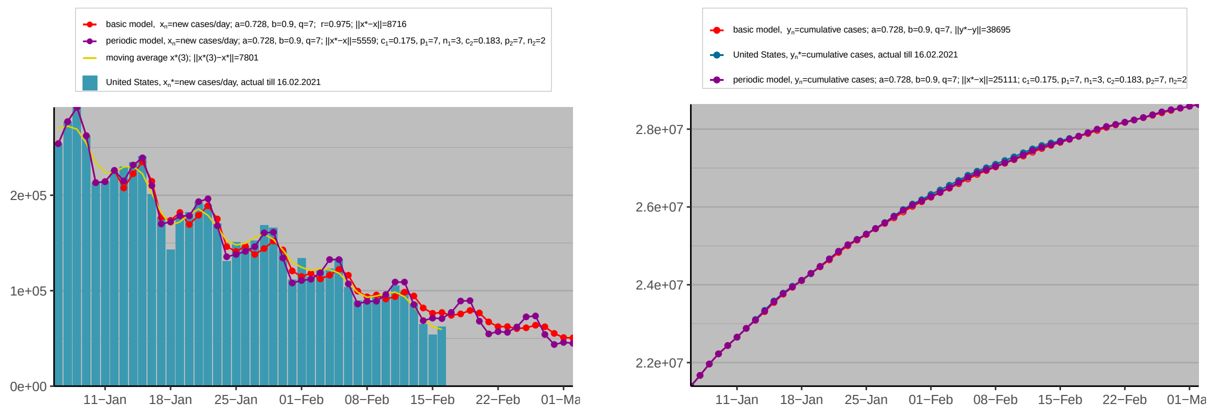

Figure 23: Periodic model, United States

2.5 Multi-phase model

I have generalized the two-phase model formulated in [1] 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 seperately for each phase.

Therefore we have phases

(1) parameters a_1, b_1, q_1 and mdel x_n for $0 \leq n \leq N_1$

(2) parameters a_2, b_2, q_2 and mdel x_n for $N_1 + 1 \leq n \leq N_2$

$\vdots$

(M) parameters a_M, b_M, q_M and mdel 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 seperately 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 (5) 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 unrealstic, and is purely for demonstration

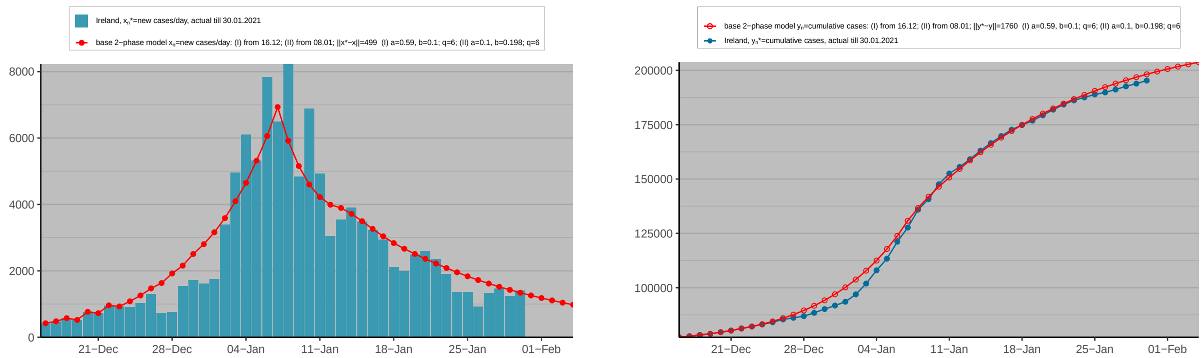

Figure 24: Multi-phase model, Ireland

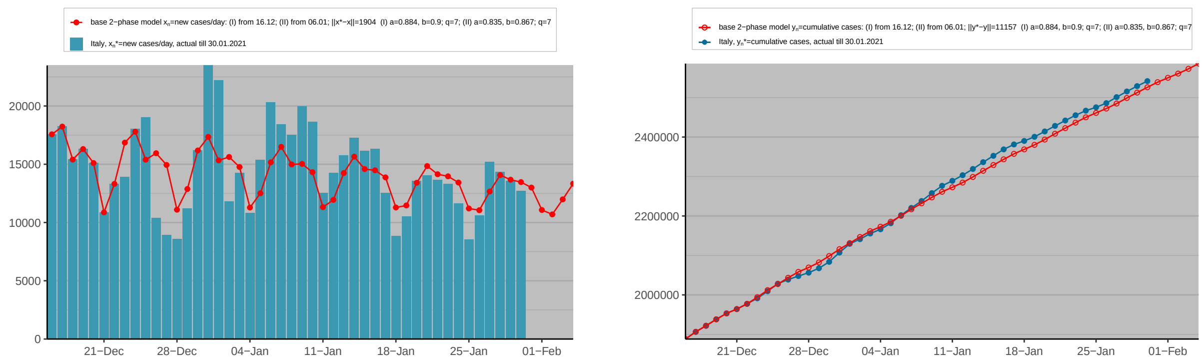

Figure 25: Multi-phase model, Italy

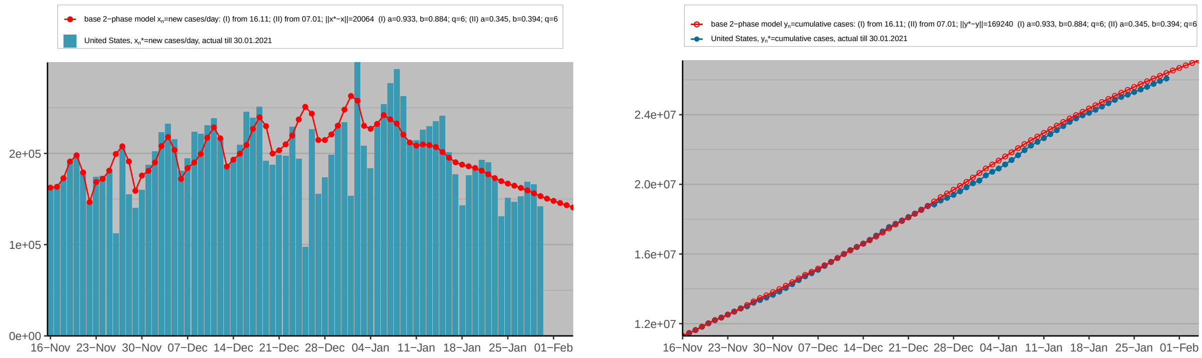

Figure 26: Multi-phase model, United States

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

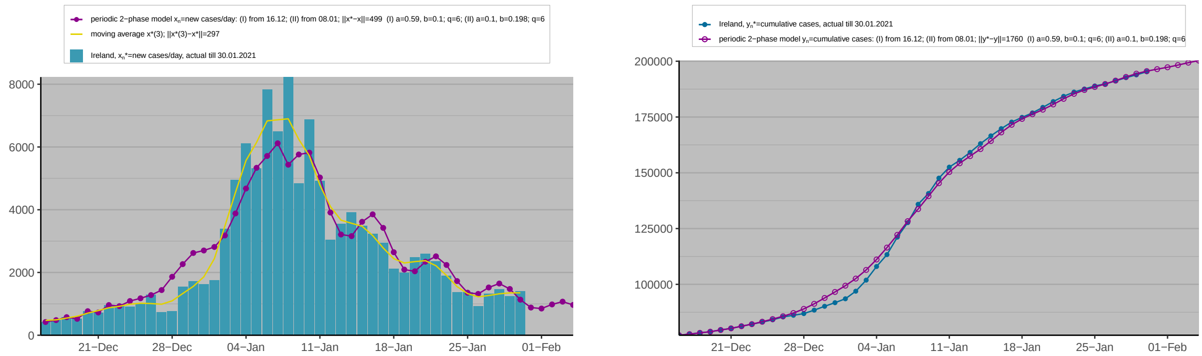

Figure 27: Multi-phase periodic model, Ireland

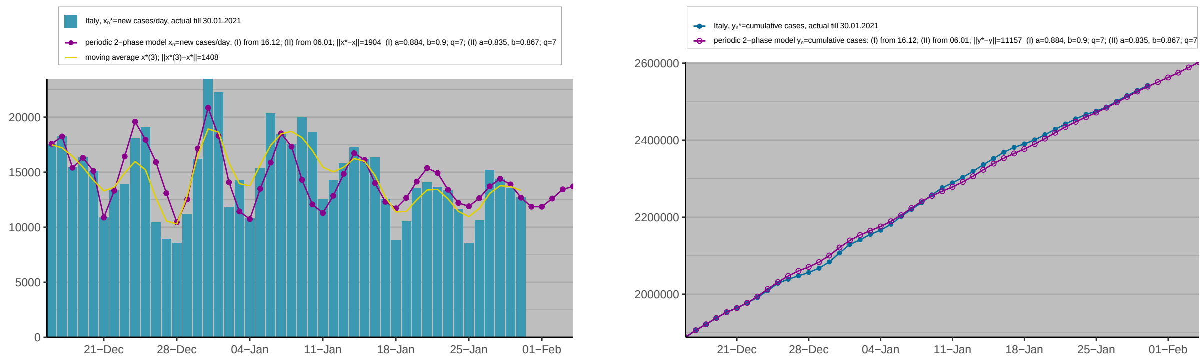

Figure 28: Multi-phase periodic model, Italy

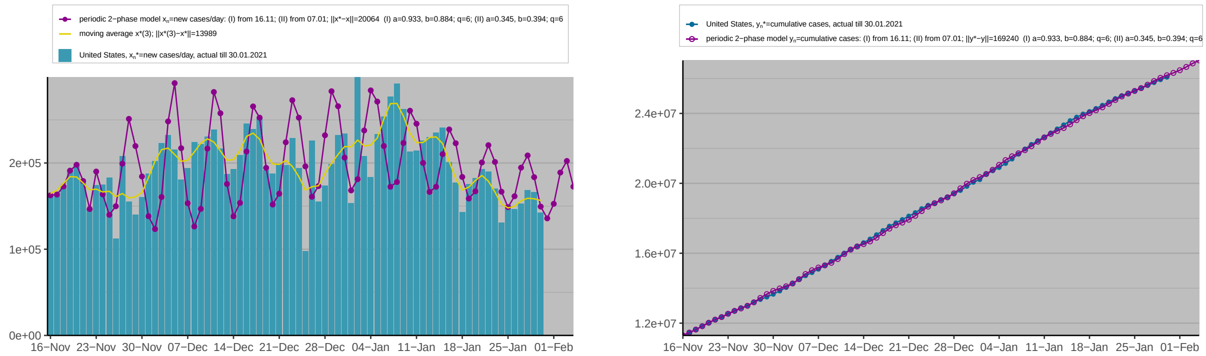

Figure 29: Multi-phase periodic model, United States

3 Statistical Models

Primary source for this was Hyndman and Athanasopoulos [7].

Some of our statistical models require *homoscedasticity*, i.e., that the model errors are identically distributed with have 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

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

with the full code in 8.

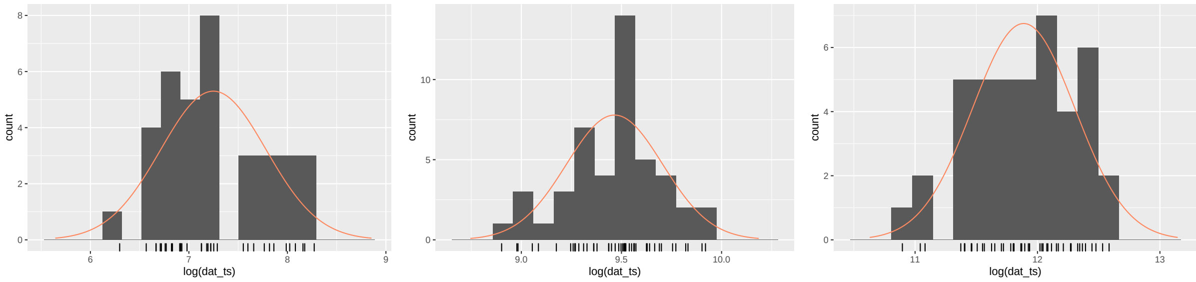

Figure 30: Normality checks, Ireland, Italy and United States

3.1 Holt-Winters' seasonal method

The Holt-Winters seasonal exponential smoothing algorithm models three aspects of the time series: the average value at different levels, the trend over time, and any seasonality.

3.1.1 Definitions and Theory

Suppose there are N observations.

Initial step:

$$L_s = \frac{1}{s}\sum_{i=1}^{s} x_i$$

$$b_s = \frac{1}{s}\left[\frac{x_{s+1}-x_1}{s} + \frac{x_{s+2}-x_2}{s} + \cdots + \frac{x_{2s}-x_s}{s}\right]$$

$$S_n = x_n - L_s, \; n = 1, \ldots, s$$

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

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

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

For subsequent observations,

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

Figure 31: Seasonal Holt-Winters' Additive Model Algorithm (denoted SHW_+)

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

3.1.2 Implementation in R

```
#lambda=0 ensures values stay positive
hwfcst     <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
...
...
...
```

```
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.

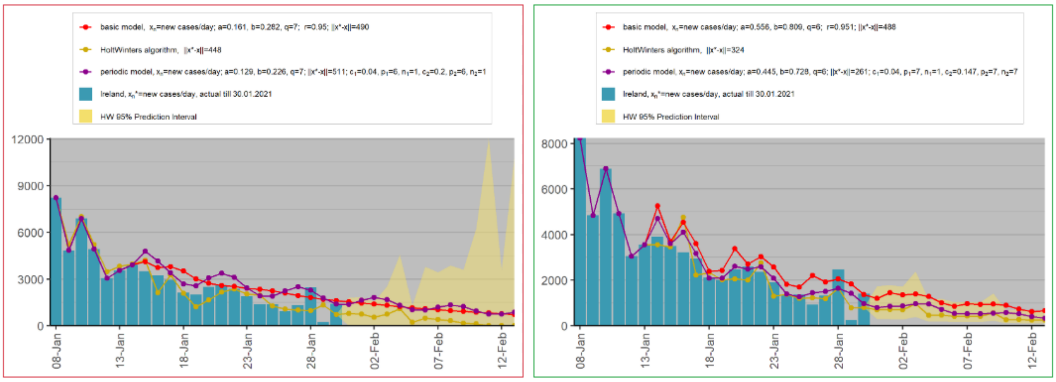

Figure 32: Comparison of HoltWinters multiplicative (left) and additive (right) algorithms

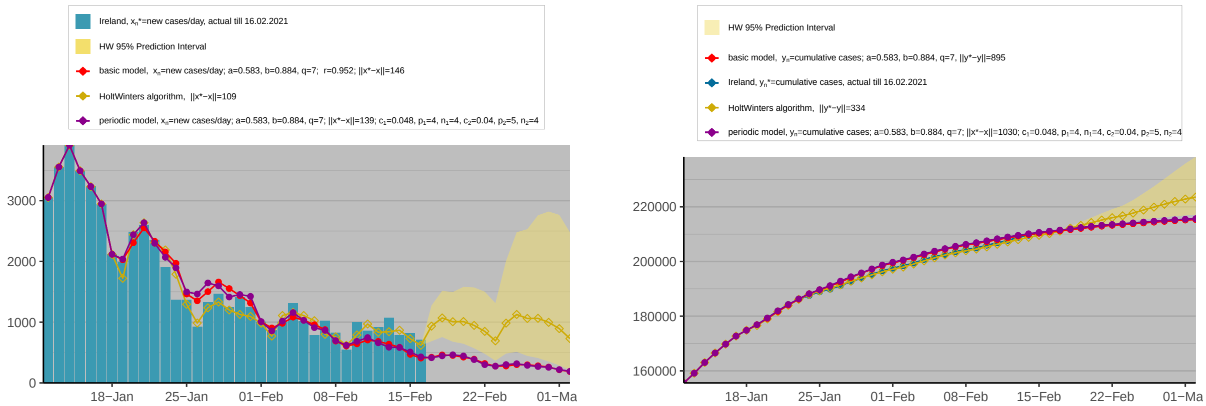

Figure 33: HoltWinters model, Ireland

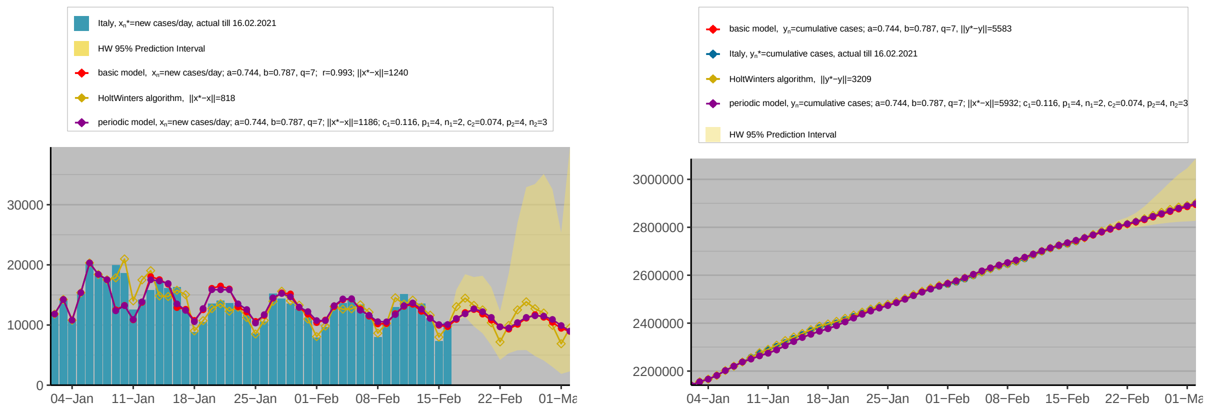

Figure 34: HoltWinters model, Italy

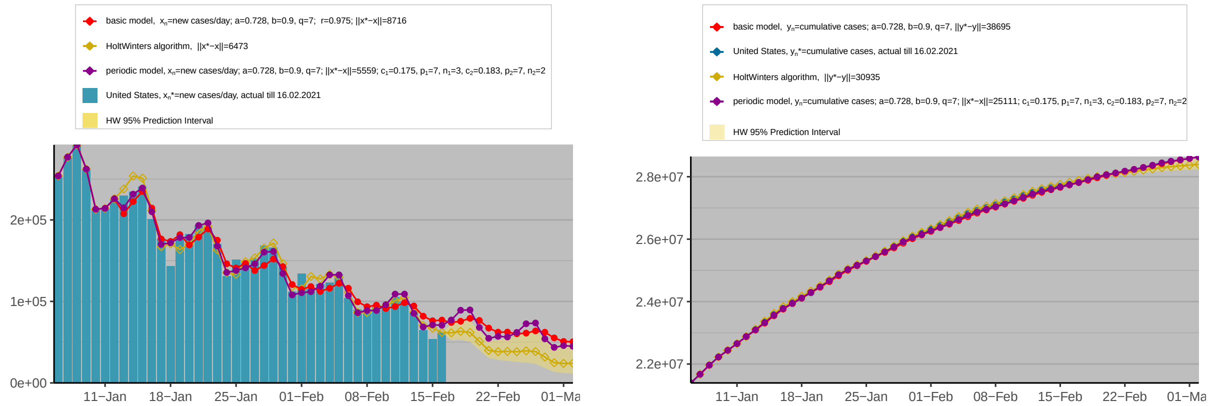

Figure 35: HoltWinters model, United States

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 genrerally:

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

And similarly for the independent errors ε_n:

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

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

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

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

  Where there are constraints on the $\phi_0, \phi_1, \ldots, \phi_p \in \mathbb{R}$
  We can simplify using the backshift operator B:

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

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

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

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

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

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

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

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

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

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

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

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

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

where $c = \phi_0 + \psi_0$ (it is zero if $d \geq 1$).

We also need the seasonal components for an ARIMA$(p, d, k)(P, D, K)_s$

Suppose a time series x_n has period s (seasonal pattern every s values)

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

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

  We can simplify using the backshift operator B:

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

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

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

  Again, by convention we use minus signs in the coefficients $\gamma_1, \ldots, \gamma_K$

  We can simplify using the backshift operator B:

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

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

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

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

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

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

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

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

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

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

The $P+K+p+k+1$ coefficients $c, \phi_1, \ldots, \phi_p, \psi_1, \ldots, \psi_k, \beta_1, \ldots, \beta_P, \gamma_1, \ldots, \gamma_K$ are computed using Maximum Likelihood Estimation.

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

Remark. Most of the literature, for example [7], use q for non-seasonal and Q for seasonal Moving Average models. I am using k and K respectively to avoid confusion with q in the mathematical models .

3.2.2 Implementation in R

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

Listing 5: Algorithm for ARIMA Model

3.2.3 Plots

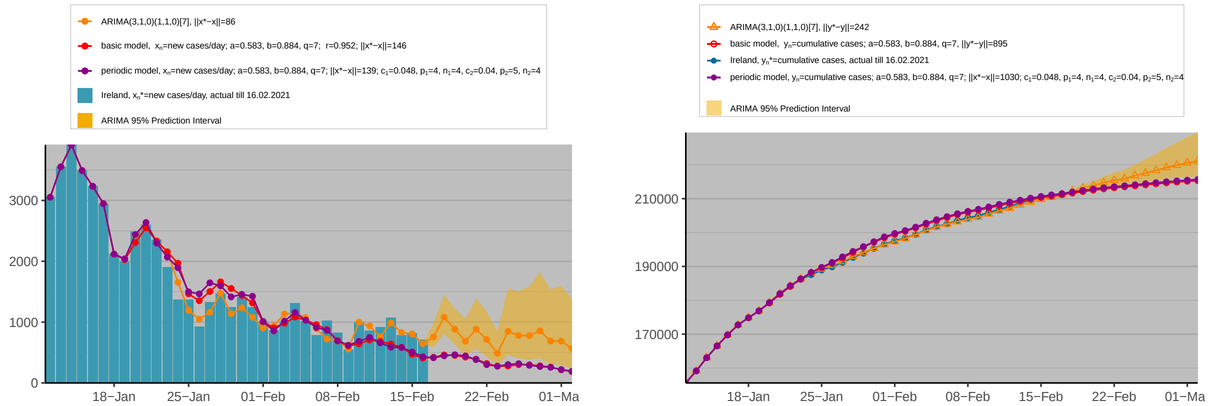

Figure 36: ARIMA model, Ireland

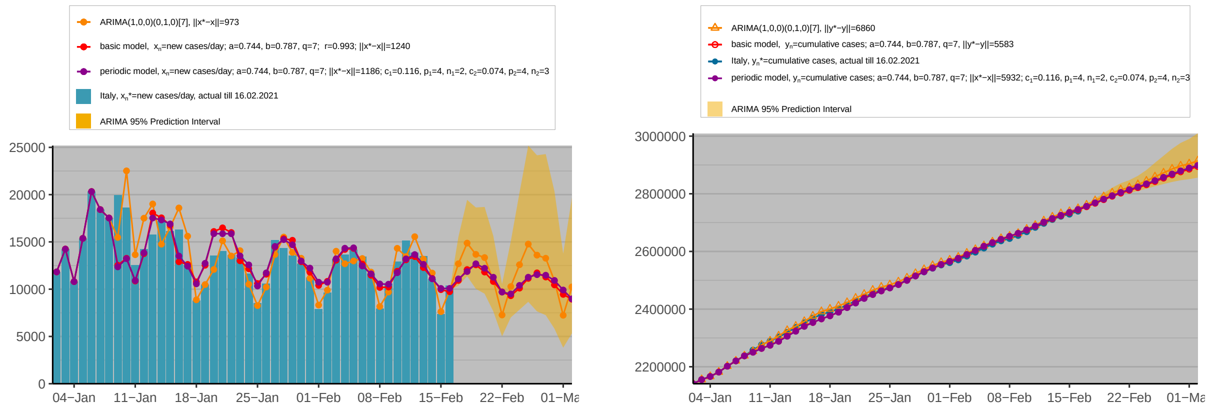

Figure 37: ARIMA model, Italy

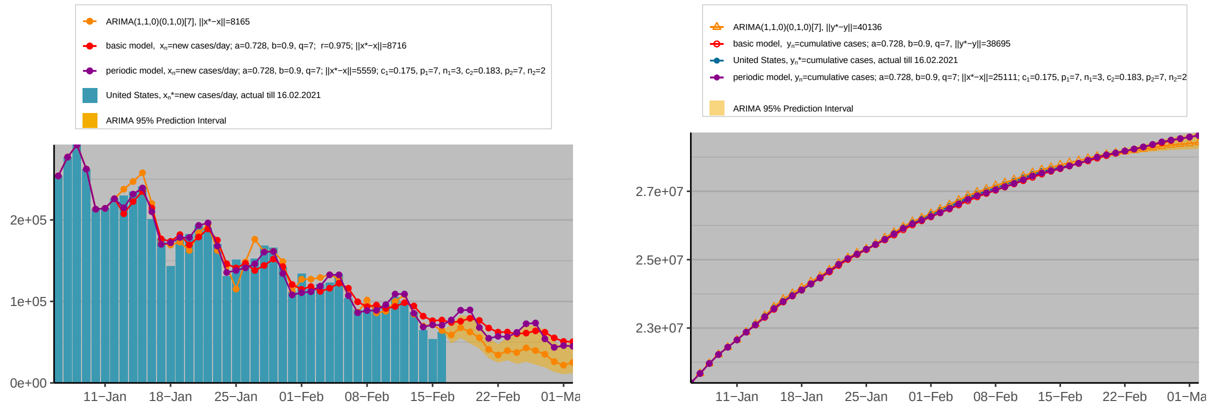

Figure 38: ARIMA model, United States

3.3 Neural network models

An ARIMA$(p, 0, 0)$ has inputs $x_{n-1}, \dots, x_{n-p}$ and parameters $\phi_1, \dots, \phi_p$ 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.

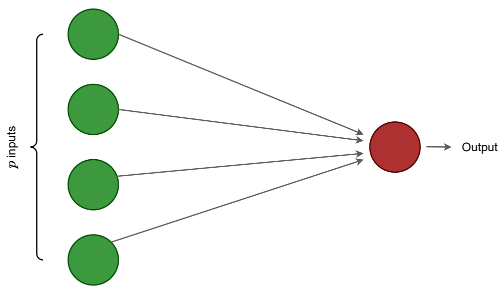

Figure 39: A linear regression model, or ARIMA$(p, 0, 0)$ model.

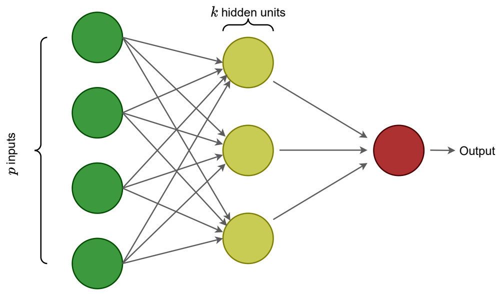

Figure 40: A neural network with p inputs and one hidden layer with k hidden neurons.

With a Neiral Network Auroregressive 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

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

The diagram 40 is a *multilayer feed-forward network*, as each lconsecutive layer (left to right) of nodes takes inputs from the previous layer, adding complexity.

The `forecast::nnetar()` function only allows for one layer to reduce the likelihood of overfiting of a univariate time series.

3.3.1 Implementation in R

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

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

Listing 6: Algorithm for NNAR Model

3.3.2 Plots

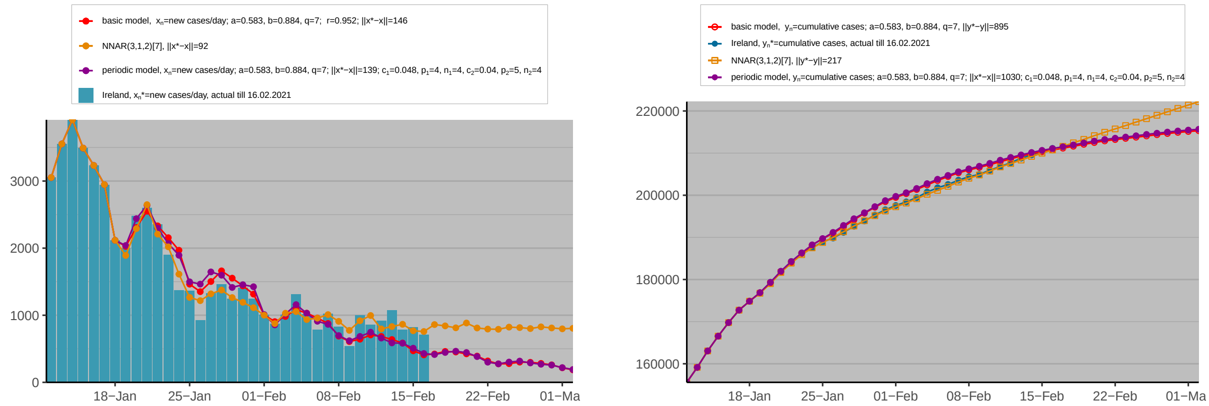

Figure 41: Neural Network Autoregression model, Ireland

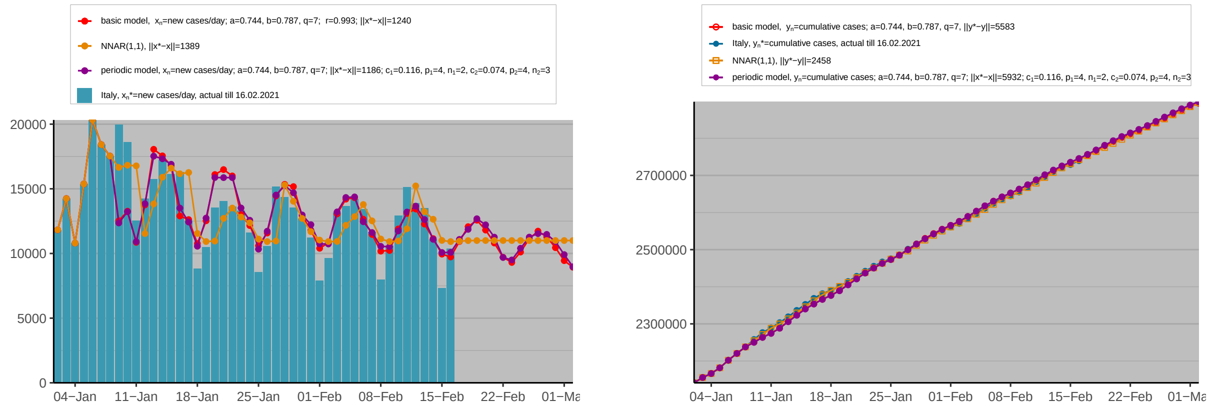

Figure 42: Neural Network Autoregression model, Italy

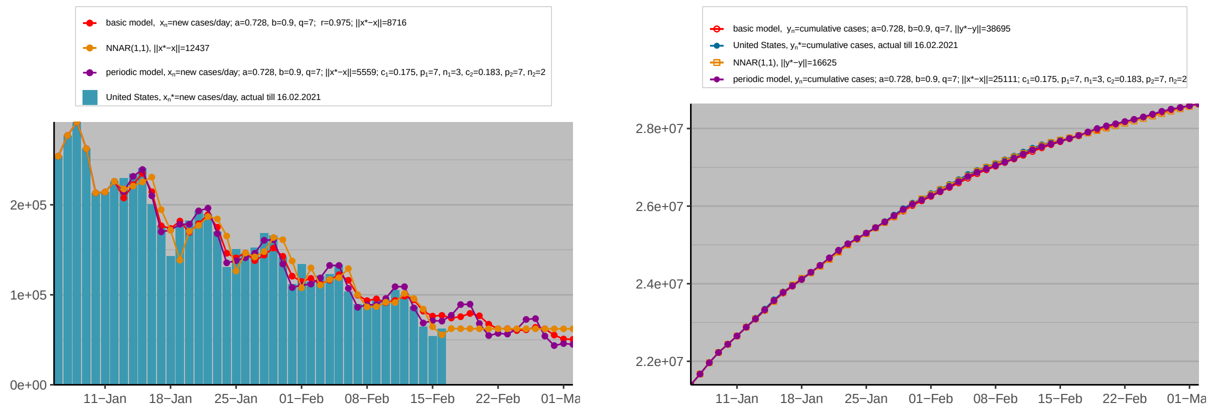

Figure 43: Neural Network Autoregression model, United States

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*.

4.1 Summary of model parameters and results

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 1: 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 2: 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	0.13	7.00	3.00	0.15	7.00	2.00

Table 3: United States: mathematical models parameters

Below is a summary of results of the applied model for the primary date range for the training set (at least 4 weeks). The test set is then the following fourteen days of cases.

	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	91	221	163	556

Table 4: 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	1388	2460	4623	24717

Table 5: 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	5613	25841	8785	22280
4	HoltWinters	5763	22833	21703	93870
5	ARIMA	8223	37447	24889	107978
6	Neural Network	12586	16902	8945	47923

Table 6: United States, 2021-01-06 to 2021-02-16

The moving average is only here for comparison and usually outperforms all other models.

4.2 Base and Periodic models

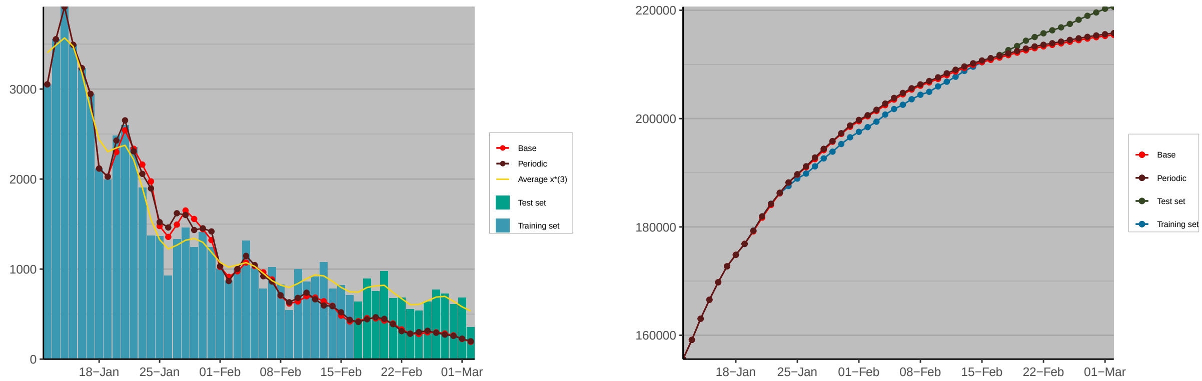

Figure 44: Periodic model with train/test split, Ireland

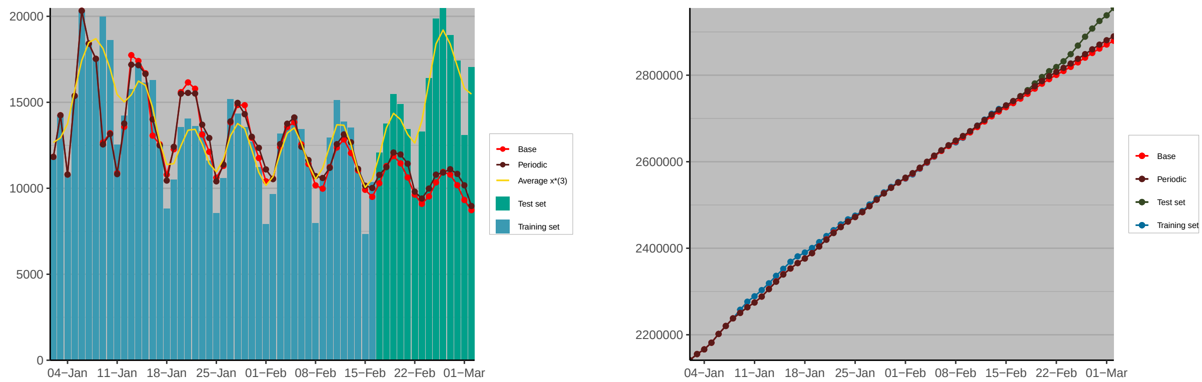

Figure 45: Periodic model with train/test split, Italy

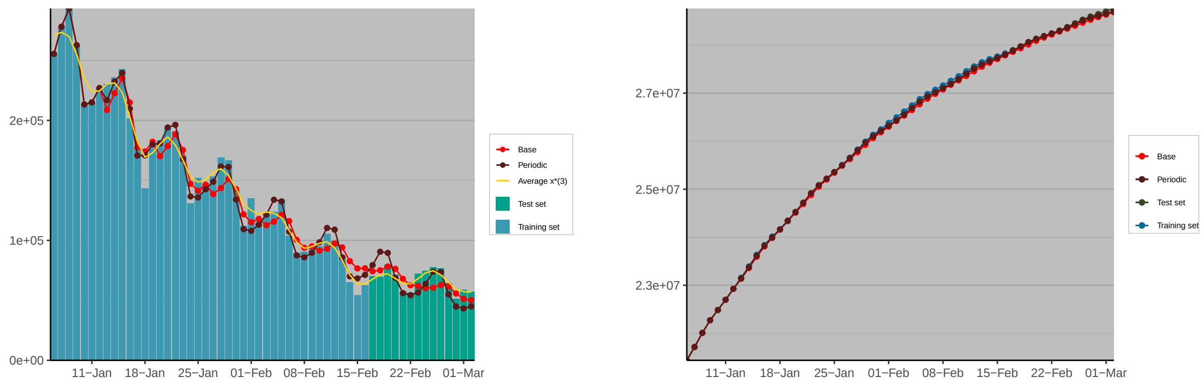

Figure 46: Periodic model with train/test split, United States

4.3 HoltWinters model

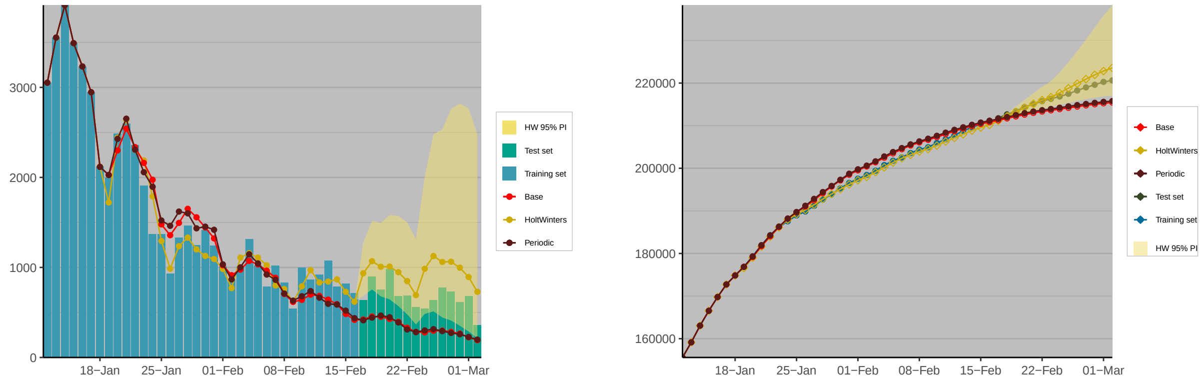

Figure 47: HoltWinters model with train/test split, Ireland

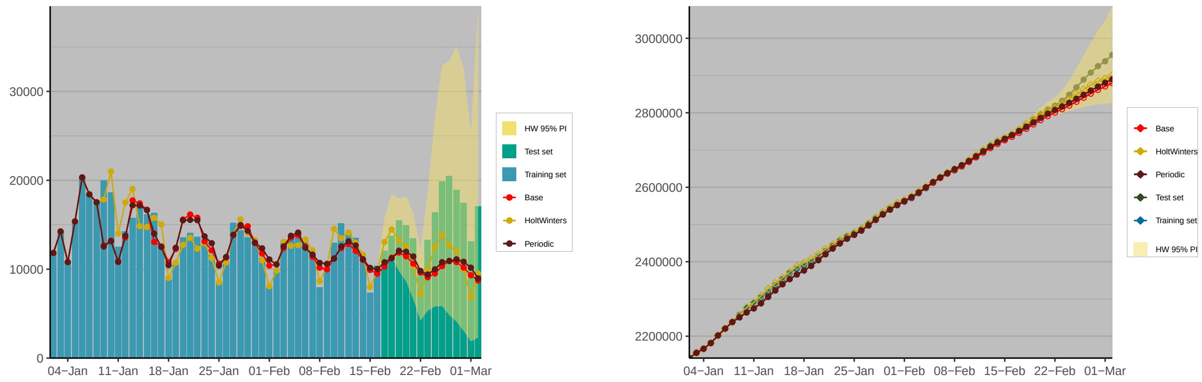

Figure 48: HoltWinters model with train/test split, Italy

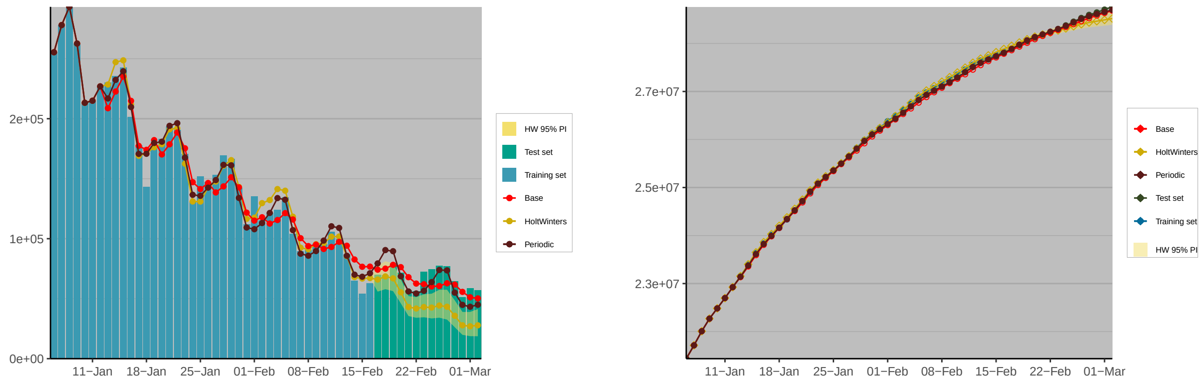

Figure 49: HoltWinters model with train/test split, United States

4.4 ARIMA model

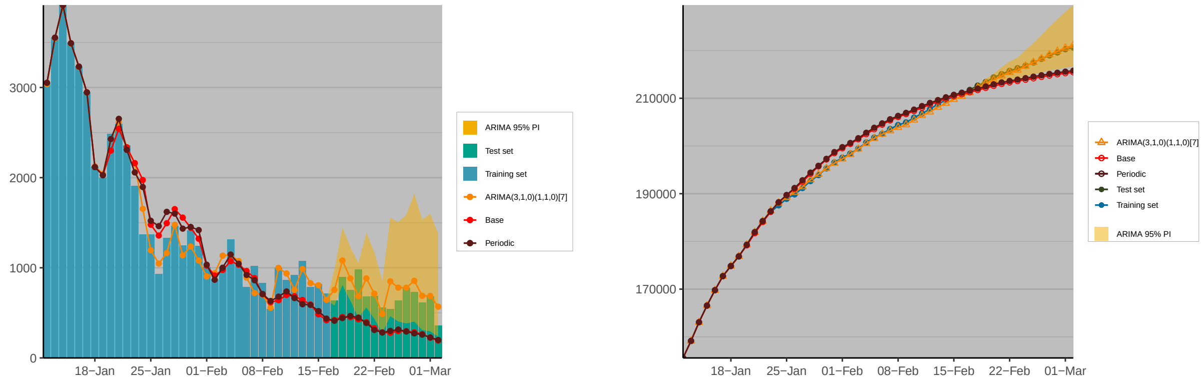

Figure 50: ARIMA model with train/test split, Ireland

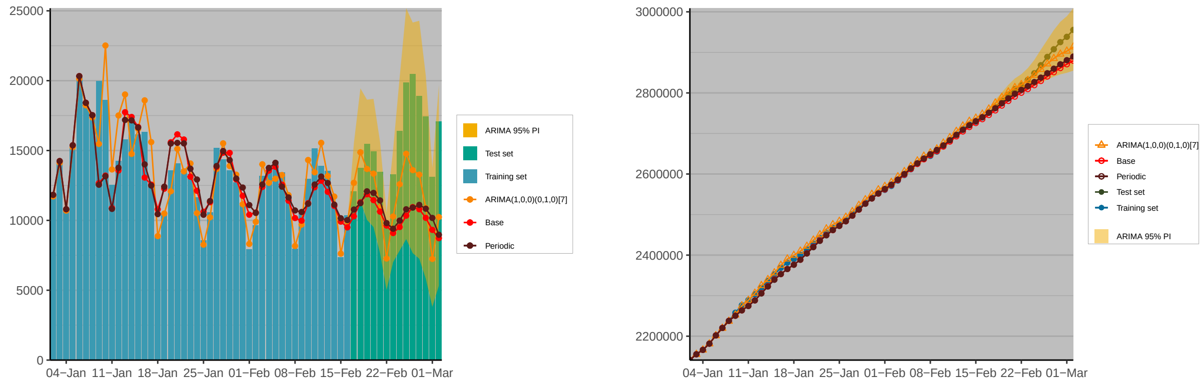

Figure 51: ARIMA model with train/test split, Italy

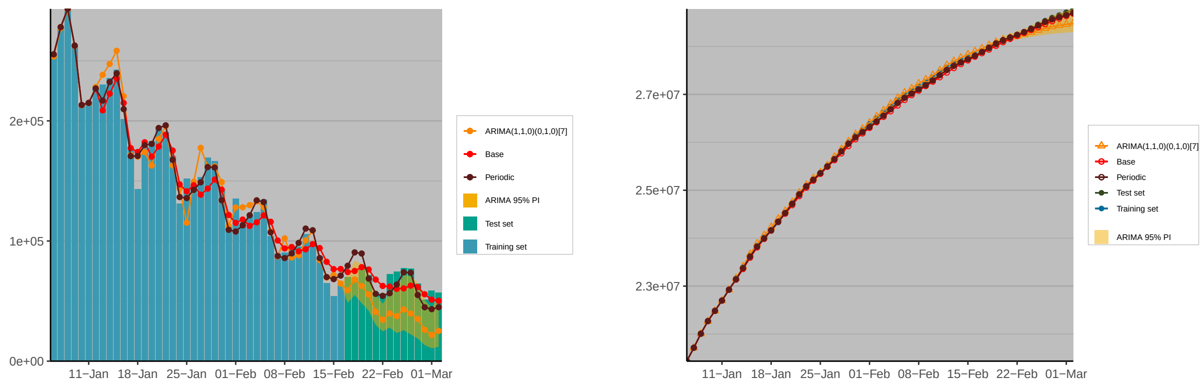

Figure 52: ARIMA model with train/test split, United States

4.5 NNAR model

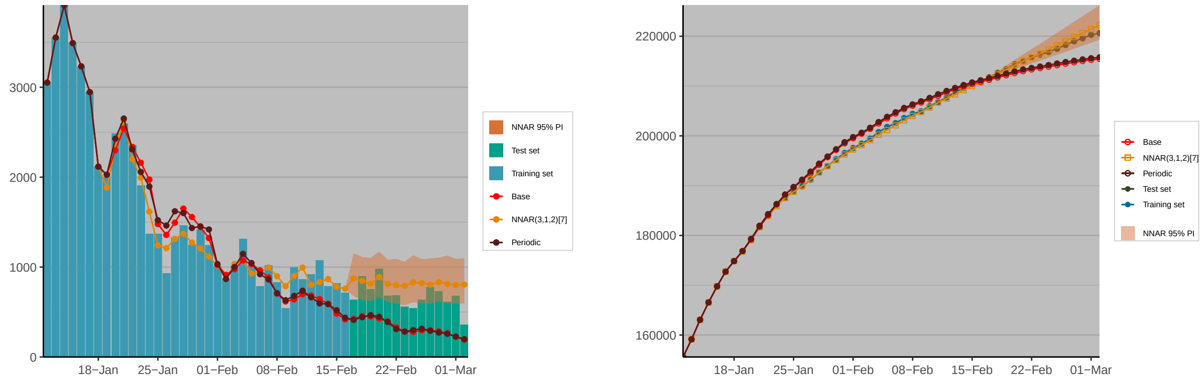

Figure 53: Neural Network Autoregression model with train/test split, Ireland

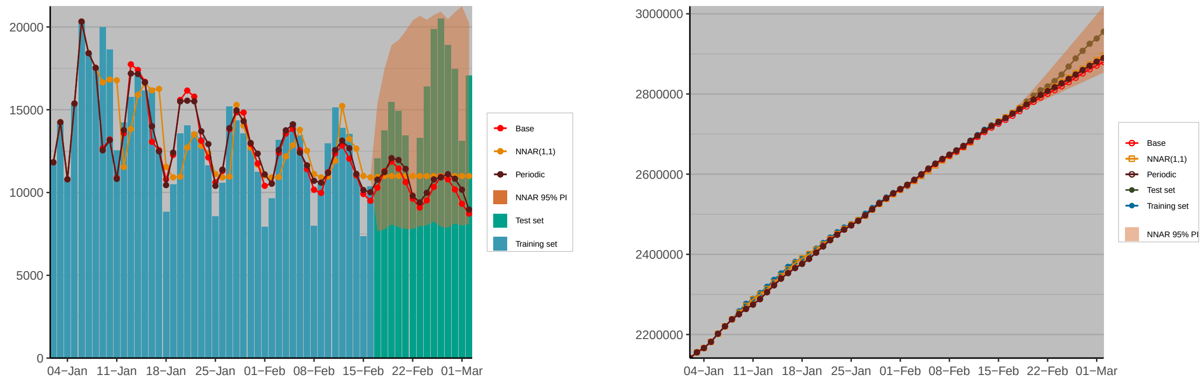

Figure 54: Neural Network Autoregression model with train/test split, Italy

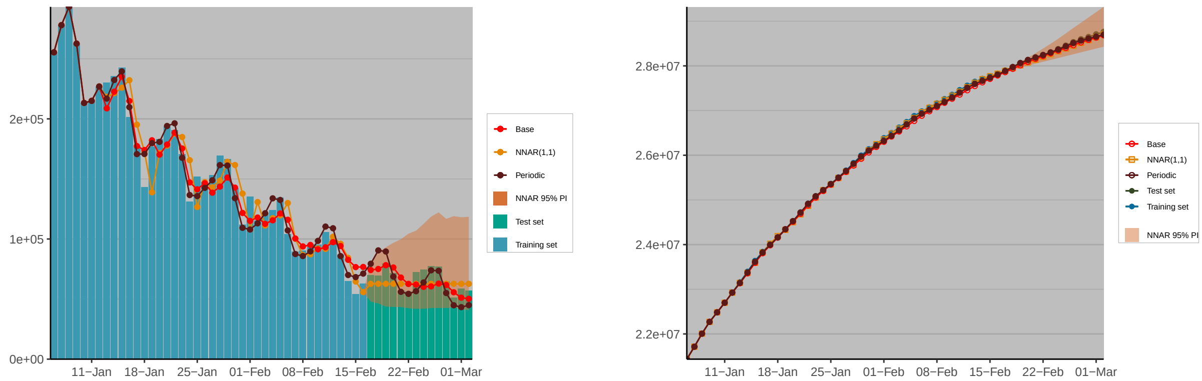

Figure 55: Neural Network Autoregression model with train/test split, United States

A Definitions and Theorems for Mathematical Models

A.1 Recurrence equation

This is our general linear recurrence equation with constant coefficients:

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

The characteristic polynomial of (17)

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

Definition 2. A root 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 geormetric mean of n values $x_1, \ldots, x_n$ is

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

A.2 Results

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

(a) (Cauchy, 1829) The polynomial $f(\lambda)$ from (18) has exactly one positive real root r. Besides, the root r is simple and, for any other root $\lambda \in \mathbb{C}$, we have $|\lambda| < r$. Consequently, r is the leading root of (17).

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

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

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

Proof:

(a) The equation $f(\lambda) = 0$ is equivalent to

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

dividing across by λ^{q+1}

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

And so

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

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

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

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

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

Now, let $\lambda \in \mathbb{C}\backslash\{0\}$ be another root of f. We obtain from (21) (using the triangle inequality) that

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

Since g is strictly decreasing,
$g(r) \leq g(|\lambda|)$ which implies $|\lambda| \leq r$.
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 $g(r) = 1$

Again, since the function g is strictly decreasing, $g(r') > g(r) \implies r' < r$

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

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

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

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

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

Since $x_n > 0$, it follows from (22) and (23) that $C \geq 0$.

We need to show that $C > 0$.

Consider a new sequence

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

This satisfies the equation

$$X_{n+1} = A_0X_n + A_1X_{n-1} + \cdots + A_qX_{n-q} \tag{24}$$

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

Since

$$\begin{aligned}
A_0X_0 + A_1X_{n-1} + \cdots + A_qX_{n-q} &= \frac{a_0}{r^1}\frac{x_n}{r^n} + \frac{a_1}{r^2}\frac{x_{n-1}}{r^{n-1}} + \cdots + \frac{a_q}{r^{q+1}}\frac{x_{n-q}}{r^{n-q}} \\
&= \frac{a_0x_n}{r^{n+1}} + \frac{a_1x_{n-1}}{r^{n+1}} + \cdots + \frac{a_qx_{n-q}}{r^{n+1}} \\
&= \frac{1}{r^{n+1}}\left(a_0x_n + a_1x_{n-1} + \cdots + a_qx_{n-q}\right) \\
&= \frac{1}{r^{n+1}}\left(x_{n+1}\right) \quad \text{from (17)} \\
&= \frac{x_{n+1}}{r^{n+1}}\left(x_{n+1}\right) \\
&= X_{n+1}
\end{aligned}$$

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

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

Set $c := \min(X_1, \ldots, X_{q+1}) > 0$ since x_n have positive initial values. Then we obtain from (24) and (25) by induction that $X_n \geq c$ for all $n \in \mathbb{N}$, which implies
$x_n \geq cr^n$
But equating with (22) we get

$$Cr^n + \tilde{x}_n \geq cr^n$$

$$C + \frac{\tilde{x}_n}{r^n} \geq c$$

Taking the limit as $n \to \infty$

$$C \geq c > 0 \tag{26}$$

Therefore $C > 0$, as required.

□

Theorem 2. Let $a_k \geq 0$ for all $k = 0, \ldots, q$. Denote $a = a_1 + \cdots + a_q, b = 1 - a_0$ and assume that $a > 0, b > 0$.
(a) We have the equivalences: $r < 1 \iff a < b$ and $r > 1 \iff a > b$.
(b) Let $m \geq 1$ be such that $a_1 = \cdots = a_{m-1} = 0$ and $a_m > 0$. Then

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

Proof:
(a) We have
$$\begin{aligned} f(1) &= 1 - a_0 - a1 - \cdots - a_q \\ &= \underbrace{(1 - a_0)}_{b} - \underbrace{(a_1 + \cdots + a_q)}_{a} \\ &= b - a \end{aligned}$$
We know f is increasing.
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
$r^{q+1} - a_0 r^q - a_1 r^{q-1} - a_2 r^{q-2} - \cdots - a_{q-1} r - a_q = 0$
But any $a_1, \ldots, a_{m-1}$ are all zero
$\implies r^{q+1} - a_0 r^q - a_m r^{q-m} - a_{m+1} r^{q-m-1} - \cdots - a_{q-1} r - a_q = 0$
$\implies r^{q+1} - (1-b) r^q - a_m r^{q-m} - \cdots - a_q = 0$
$\implies r^{q+1} - r^q + br^q - a_1 r^{q-m} - \cdots - a_q = 0$
$\implies r^{q+1} - r^q = -br^q + a_m r^{q-m} + \cdots + a_q$
If $r > 1$ then $r^{q+1} > r^q$ and so $r^{q+1} - r^q > 0$
and so
$0 < -br^q + a_m r^{q-m} + \cdots + a_q$
$$\begin{aligned} \implies br^q &< a_m r^{q-m} + \cdots + a_q \\ &\leq a_m r^{q-m} + \cdots + a_q r^{q-m} \\ &= (a_m + \cdots + a_q) r^{q-m} \\ &= ar^{q-m} \end{aligned}$$
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 $x \leq \max(1, x)$ and $x \geq \min(1, x)$ (for any $x \in \mathbb{R}$) to get (27), as required.

□

Remark. The inequality (27) will be useful to help approximate the leading root r in 4.

Lemma 3. For the model described by recurrence 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 $k = 1, \ldots, q$ the number $c_k u$ of them become ill and can infect other people. On the day $k+1$ they infect $ac_k u$ people while $bc_k u$ 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

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

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 reprodiction 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 [8] to find a better approximation for r.

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

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

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

$$\{r_n\} \to r \quad \text{as } n \to \infty \tag{29}$$

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

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

Proof: Our characteristic polynomial (via (18)) for our recurrence equation (3.2.1) is

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

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

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

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

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

The derivative of our characteristic polynomial (31) is

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

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}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q+1} - (1-b)\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q} - a}{(q+1)\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q} - q(1-b)\left(\left(\frac{a}{b}\right)^{\frac{1}{2q}}\right)^{q-1}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\frac{a}{b}\right)^{\frac{q+1}{2q}} - (1-b)\left(\frac{a}{b}\right)^{\frac{1}{2}} - a}{(q+1)\left(\frac{a}{b}\right)^{\frac{1}{2}} - q(1-b)\left(\frac{a}{b}\right)^{\frac{q-1}{2q}}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\frac{a}{b}\right)^{-\frac{1}{2}}}{\left(\frac{a}{b}\right)^{-\frac{1}{2}}} \cdot \frac{\left(\frac{a}{b}\right)^{\frac{q+1}{2q}} - (1-b)\left(\frac{a}{b}\right)^{\frac{1}{2}} - a}{(q+1)\left(\frac{a}{b}\right)^{\frac{1}{2}} - q(1-b)\left(\frac{a}{b}\right)^{\frac{q-1}{2q}}} \\
&= \left(\frac{a}{b}\right)^{\frac{1}{2q}} - \frac{\left(\frac{a}{b}\right)^{\frac{1}{2q}} - (1-b) - a\left(\frac{a}{b}\right)^{-\frac{1}{2}}}{(q+1) - q(1-b)\left(\frac{a}{b}\right)^{-\frac{1}{2q}}} \qquad \square \\
&= \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 $\lim_{n\to\infty} r_n = r$.

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

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

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

$$f(r_1) = f(r_0) - f'(c)\frac{f(r_0)}{f'(r_0)} \tag{37}$$
$$= f(r_0)\left(1 - \frac{f'(c)}{f'(r_0)}\right) \tag{38}$$

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

$$r' < r < r_1 \tag{39}$$

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

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

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

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

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

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

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

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

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

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

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

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

$$\left|\frac{c_1p_1}{2\pi N}\left(\cos\left(\frac{2\pi\,(N+n_1)}{p_1}\right) + \cos\left(\frac{2\pi n_1}{p_1}\right)\right)\right| < \frac{1.6}{300}\cdot(1+1) = \frac{4}{375}$$

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}.

$\square$

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 [1].

B.1 R packages

ggplot2 [9] is widely used for easily plotting and visualising the models.
rgdal [10] allows geospatial .shp files to be read into R.
raster [11] allows this data to be manipulated and plotted.
dplyr [12] provides useful data manipulation functions, both for models and geospatial mapping.
Statistical models (HoltWinters, ARIMA and Neural Network Regression) were readily implemented from forecast [13].
The majority ofthe colours for plotting are selected using colour palettes from the wesanderson package [14].

B.2 Shapefiles and Datasets

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 [15], while the more detailed county-level shapefile for Ireland was downloaded from [16].

Country-level data:
I have chosen to use the data from [17], which has detailed daily data for nearly 200 countries and territories.
Ireland county-level data:
Downloaded from [18] and used to visualise the 14-day incidence rate per 1 million of the population, for each county.

B.3 Source Code

All the code, plots, and even this report, are available at my GitHub.

I will include the main code files in text below.

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

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

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

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

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

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

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

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

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

movingavg <- function(x){
  mavgx <- (x[1]+x[2])/2
  for(i in 2:(length(x)-1)){
```

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

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

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

Listing 7: Model helper functions

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

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

plotslist <- list()

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

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

  forecastlen <- 14

  prevcases <- countrydat$yn[countrydat$date == as.Date(dateBounds[1])-1]
  #Specific dates
  countrydattest  <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= (dateBounds
      [2] + forecastlen),]
  countrydat <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= dateBounds[2],]
  latest_date <- dateBounds[2]
  testBounds <- dateBounds[2] + c(1:forecastlen)

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

  cols <- list(
    xn       = wes_palettes$Zissou1[1],
    xntest   = wes_palettes$Darjeeling1[2],
    yn       = wes_palettes$Darjeeling2[2],
    yntest   = wes_palettes$BottleRocket2[4],
    basexn   = wes_palettes$Darjeeling1[1],
    baseyn   = wes_palettes$Darjeeling1[1],
    x3       = wes_palettes$BottleRocket2[1],
    Crn      = wes_palettes$FantasticFox1[2],
    arima    = wes_palettes$Darjeeling1[4],
    arimapi  = wes_palettes$Darjeeling1[3],
    hw       = wes_palettes$Moonrise1[2],
    hwpi     = wes_palettes$Moonrise1[1],
    periodic = wes_palettes$GrandBudapest1[3], #"magenta",
    nn       = wes_palettes$FantasticFox1[4],
    nnpi     = wes_palettes$GrandBudapest1[4]
```

```r
)

labs <- list(
  train    = "Training set",
  test     = "Test set",
  base     = "Base",
  x3       = "Average x*(3)",
  periodic = "Periodic",
  hw       = "HoltWinters",
  hwpi     = "HW 95% PI",
  arimapi  = "ARIMA 95% PI",
  nnpi     = "NNAR 95% PI"
)

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

aseq    <- seq(from = 0.1, to = 1.5, length.out = 40)
bseq    <- seq(from = 0.1, to = 0.9, length.out = 40)
qseq    <- 6:8
normdat <- expand.grid(q = qseq, a = aseq, b = bseq)
abnorm  <- apply(normdat, 1, function(x) norm(x, countrydat$xn))

normdat$abnorm  <- abnorm
normdat$abnormn <- normalize(abnorm)

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

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

newnormdat$combnorm <- 1*newnormdat$abnormn + 0.00*newnormdat$abnormyn

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

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

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

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

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

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

modeldat$tgroup <-  ifelse(modeldat$date <= dateBounds[2], "train", "test")

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

trainrange  <- 1:nrow(countrydat)
testrange   <- nrow(countrydat) + 1:forecastlen

roptimpars  <- round(optimpars,3)
xnorm       <- norm(optimpars,countrydat$xn)
xnormtest   <- modnorm(basexn[testrange], countrydattest$xn[testrange])

ynorm       <- modnorm(countrydat$yn, modeldat$baseyn[1:length(countrydat$yn)])
ynormtest   <- modnorm(modeldat$baseyn[testrange], countrydattest$yn[testrange])

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

```r

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

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

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

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

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

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

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

x3norm   <- modnorm(modeldat$xn[trainrange], modeldat$mavgx3[trainrange])
x3normt  <- modnorm(modeldat$xn[testrange], modeldat$mavgx3[testrange])
x3normy  <- modnorm(modeldat$yn[trainrange], modeldat$mavgx3y[trainrange])
x3normyt <- modnorm(modeldat$yn[testrange], modeldat$mavgx3y[testrange])

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

summarydf <- rbind(summarydf, data.frame(model = "Moving Average", xnorm = x3norm, ynorm = x3normy,
                                         xnormt = x3normt, ynormt = x3normyt))

#parameters of the form (ci,pi,ni)
##an = a(1 + c1 sin(2pi/p1 (n - n1)))
c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 17)
n_1seq  <- n_2seq <- 1:q
p_1seq  <- p_2seq <- 1:q
normdatp <- expand.grid(a  = a,       b  = b,
                        c1 = c_1seq,  c2 = c_2seq,
                        p1 = p_1seq,  p2 = p_2seq,
                        n1 = n_1seq,  n2 = n_2seq)

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

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

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

pernormy  <- modnorm(countrydat$yn, modeldat$periodicy[trainrange])
pernormt  <- modnorm(modeldat$xn[testrange], modeldat$periodic[testrange])
pernormyt <- modnorm(modeldat$yn[testrange], modeldat$periodicy[testrange])

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

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

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

#a,b,c1,c2,p1,p2,n1,n2

```

```r
mathperamdat <- data.frame(a = peroptim[1], b  = peroptim[2], q  = q, r = base_r_one,
                           c1 = peroptim[3], p1 = peroptim[5], n1 = peroptim[7],
                           c2 = peroptim[4], p2 = peroptim[6], n2 = peroptim[8])

colnames(mathperamdat) <- c("$a$", "$b$", "$q$", "$r$",
                             "$c_1$", "$p_1$", "$n_1$", "$c_2$", "$p_2$", "$n_2$")

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

summarydf <- rbind(summarydf, data.frame(model = "Periodic", xnorm = pernorm, ynorm = pernormy,
                                         xnormt = pernormt, ynormt = pernormyt))

#Statistical methods using timeseries forecasting

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

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

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

plots[["residuals"]] <- gghistogram(log(dat_ts), add.normal = TRUE, bins=10)

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

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

  hwfcst$fitted[1:q] <- countrydat$xn[1:q]
  modeldat$hwxn <- c(hwfcst$fitted, hwfcst$mean)
  modeldat$hwlo <- c(hwfcst$fitted, hwfcst$lower[,2])
  modeldat$hwhi <- c(hwfcst$fitted, hwfcst$upper[,2])

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

  hwnorm   <- modnorm(countrydat$xn, hwfcst$fitted)
  hwnormt  <- modnorm(modeldat$xn[testrange], hwfcst$mean)
  hwnormy  <- modnorm(countrydat$yn, modeldat$hwyn[trainrange])
  hwnormyt <- modnorm(modeldat$yn[testrange], modeldat$hwyn[testrange])

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

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

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

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

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

arimanorm  <- modnorm(countrydat$xn,arima.fcst$fitted)
arimanormt <- modnorm(modeldat$xn[testrange], arima.fcst$mean)

arimalabs   <- getArmaModel(auto.fit$arma)
labs$arima  <-  arimalabs

modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
```

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

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

  arimanormy   <- modnorm(countrydat$yn,modeldat$arimayn[trainrange])
  arimanormyt  <- modnorm(modeldat$yn[testrange], modeldat$arimayn[testrange])

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

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

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

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

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

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

  nnnorm      <- modnorm(countrydat$xn, nn.fcst$fitted)
  nnnormtest  <- modnorm(modeldat$xn[testrange], nn.fcst$mean)
  nnnormy     <- modnorm(countrydat$yn,modeldat$nnyn[trainrange])
  nnnormytest <- modnorm(modeldat$yn[testrange], modeldat$nnyn[testrange])

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

  summarydf <- rbind(summarydf, data.frame(model = "Neural Network", xnorm = nnnorm, ynorm = nnnormy,
                                           xnormt = nnnormtest, ynormt = nnnormytest))
  summarydf$xnorm  <- as.integer(summarydf$xnorm)
  summarydf$ynorm  <- as.integer(summarydf$ynorm)
  summarydf$xnormt <- as.integer(summarydf$xnormt)
  summarydf$ynormt <- as.integer(summarydf$ynormt)

  colnames(summarydf) <- c("model", "$||x-x^*||_{train}$", "$||y-y^*||_{train}$",
                           "$||x-x^*||_{test}$", "$||y-y^*||_{test}$")
  return(list("plots" = plots, "summary" = summarydf, "desc" = countrydesc,
              "dat" = modeldat, "mathmodelpars" = mathperamdat))
}

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

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

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

for(country in names(datebounds)){
```

```
  plotslist[[country]] <- covidPlots(country, datebounds[[country]], owiddat)
}
```

Listing 8: Main algorithm

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

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

multidates <- list(
  "Italy"         = list(c("2020-12-16", "2021-01-05"),
                         c("2021-01-06", "2021-01-30")),
  "United States" = list(c("2020-11-16", "2021-01-06"),
                         c("2021-01-07", "2021-01-30")),
  "Ireland"       = list(c("2020-12-16", "2021-01-07"),
                         c("2021-01-08", "2021-01-30"))
)

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

  countrydat  <- countrydat[countrydat$date >= dates[[1]][1],]
  countrydat  <- countrydat[countrydat$date <= dates[[length(dates)]][2],]
  latest_date <- countrydat$date[nrow(countrydat)]

  forecastlen <- 5

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

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

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

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

```r
  }
  return(multix)
}

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

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

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

  if(i == 1)
    abnorm <- apply(normdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, start=TRUE), phasedat$xn))
  else
    abnorm <- apply(normdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, oldp = phasepars[[i-1]])[length(multimodel) + 1:nrow(phasedat)], phasedat$xn))

  normalize <- function(x){
    return((x-min(x))/(max(x)-min(x)))
  }
  normdat$abnorm  <- abnorm

  tileoptim <- normdat[which.min(normdat$abnorm),1:3]
  optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)

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

  phasepars[[i]] <- round(optimpars,3)

  if(i == 1 & i != length(dates))
    multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE)
  if(i == 1 & i == length(dates))
    multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE, len=forecastlen)
  if(i > 1 & i == length(dates))
    multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]], len=forecastlen)
  if(length(dates) > 2 & i %in% 2:(length(dates)-1))
    multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]])

  aseqper <- a*seq(from = 0.7, to = 1.3, length.out = 10)
  bseqper <- b*seq(from = 0.7, to = 1.3, length.out = 10)
  c_1seq  <- c_2seq <- seq(0.04, 0.2, length.out = 10)
  n_1seq  <- n_2seq <- c(1,7)
  p_1seq  <- p_2seq <- 6:7
  normdatp <- expand.grid(a  = aseqper, b  = bseqper,
                          c1 = c_1seq,  c2 = c_2seq,
                          p1 = p_1seq,  p2 = p_2seq,
                          n1 = n_1seq,  n2 = n_2seq)

  if(i == 1)
    pernorm <- apply(normdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1], phasedat$xn, multimodelp, start=TRUE), phasedat$xn))
  else
    pernorm <- apply(normdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1], phasedat$xn, multimodelp)[length(multimodelp)+1:nrow(phasedat)], phasedat$xn))

  normdatp$pernorm <- pernorm

  newnormdatp <- normdatp %>% top_n(pernorm, n = -25)

  if(i == 1)
    pernormy <- apply(newnormdatp, 1, function(par) modnorm(beforecumcases+xntoyn(multimodxper(
```

```
        par, q = optimpars[1], phasedat$xn, multimodelp, start=TRUE)), phasedat$yn))
    else
      pernormy <- apply(newnormdatp, 1, function(par) modnorm(beforecumcases+xntoyn(multimodxper(
        par, q = optimpars[1], phasedat$xn, multimodelp))[length(multimodelp)+1:nrow(phasedat)
        ], phasedat$yn))

    newnormdatp$pernormy <- pernormy
    newnormdatp$combnorm <- normalize(newnormdatp$pernorm) + normalize(newnormdatp$pernormy)

    peroptim <- as.numeric(newnormdatp[which.min(newnormdatp$combnorm),1:8])

    phasepars[[i]] <- c(phasepars[[i]],round(peroptim,3))

    if(i == 1 & i != length(dates))
      multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE)
    if(i == 1 & i == length(dates))
      multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE, len=
        forecastlen)
    if(i > 1 & i == length(dates))
      multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, len=forecastlen)
    if(length(dates) > 2 & i %in% 2:(length(dates)-1))
      multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp)
  }

  cols <- list(
    xn        = wes_palettes$Zissou1[1],
    yn        = wes_palettes$Darjeeling2[2],
    multixn   = wes_palettes$Darjeeling1[1],
    multiyn   = wes_palettes$Darjeeling1[1],
    multip    = "magenta4",
    x3        = wes_palettes$FantasticFox1[2]
  )

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

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

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

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

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

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

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

  countrydat$mavgx3 <- movingavg(mavgx1)

  x3norm  <- modnorm(countrydat$xn,countrydat$mavgx3)
```

```
  labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*.(x3norm)))

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

  plots[["xn"]] <- plot_multixn(countrydat, modeldat, cols, labs) + theme(legend.position = "top")

  plots[["yn"]] <- plot_multiyn(countrydat, modeldat, cols, labs) + theme(legend.position = "top")

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

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

  plots[["perxn"]] <- plot_multiperxn(countrydat, modeldat, cols, labs) + theme(legend.position = "top")

  plots[["peryn"]] <- plot_multiperyn(countrydat, modeldat, cols, labs) + theme(legend.position = "top")

  return(plots)
}

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

Listing 9: Multi-phase models

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

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

[8] Springer Verlag GmbH, European Mathematical Society. *Encyclopedia of Mathematics, Newton method*. Website. Accessed on 2021-03-19. URL: https://encyclopediaofmath.org/index.php?title=Newton_method.

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

[10] 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.

[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] 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.

[13] 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.

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

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

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

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

[18] 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.