

Modelling the Spread of COVID-19 Cases

TCD Final Year Project

Anthony Gibbons
Project Supervisor: Athanasios Georgiadis

March 10, 2021

Abstract

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

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

Plagiarism Declaration

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

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

Signed: Anthony Gibbons

Date: 10/03/2021

Contents

1	Introduction	6
1.1	Previous work on Covid-19 identification and modeling	8
1.2	Key aims	8
2	Mathematical Models	9
2.1	Base model	9
2.1.1	Definitions and Theory	9
2.1.2	How to select the best model	9
2.1.3	Forecasting	9
2.1.4	Implementation in R	9
2.1.5	Plots	9
2.2	Limiting curve	11
2.3	Moving average	13
2.4	Periodic model	15
2.4.1	Definitions and Theory	15
2.4.2	How to select the best model	15
2.4.3	Forecasting	15
2.4.4	Implementation in R	15
2.4.5	Plots	17
2.5	Multi-phase model	18
2.5.1	Definitions and Theory	18
2.5.2	How to select the best model	18
2.5.3	Forecasting	18
2.5.4	Implementation in R	18
2.5.5	Plots	18
3	Theorems for Mathematical Model	21
3.1	Model Assumptions	21
3.2	Notation	21
3.2.1	Why minimize both diatances?	21
3.2.2	Recurrence equation	21
3.3	Theorems	22
4	Statistical Models	25
4.1	Holt-Winters' seasonal method	25
4.1.1	Definitions and Theory	25
4.1.2	How to select the best model	25
4.1.3	Forecasting	25
4.1.4	Implementation in R	25
4.1.5	Plots	26
4.2	ARIMA models	28
4.2.1	Definitions and Theory	28
4.2.2	How to select the best model	30
4.2.3	Forecasting	30
4.2.4	Implementation in R	30
4.2.5	Plots	31
4.3	Neural network models	32
4.3.1	Definitions and Theory	33
4.3.2	How to select the best model	33
4.3.3	Forecasting	33
4.3.4	Implementation in R	33
4.3.5	Plots	34
5	R Code and Data Sources	36
5.1	R packages	36
5.2	Plotting and colour	36
5.3	Shapefiles	36
5.4	Datasets	36
A	Appendix	37

List of Figures

1	Daily Cases Globally	6
2	Cases per 1 million population, World	7
3	Cases per 1 million population, Europe	7
4	Situation by County, Ireland	8
5	Contour plots of $\ x - x^*\ $, Ireland, Italy and United States	9
6	Basic model, Ireland	9
7	Basic model, Italy	10
8	Basic model, United States	10
9	Comparison of x_n^* , x_n and the limiting curve Cr^n , Ireland	11
10	Comparison of x_n^* , x_n and the limiting curve Cr^n , Italy	11
11	Comparison of x_n^* , x_n and the limiting curve Cr^n , United States	12
12	Limiting Curve Growing Exponentially, Ireland	12
13	Moving average $x_n^*(3)$, Ireland	13
14	Moving average $x_n^*(3)$, Italy	14
15	Moving average $x_n^*(3)$, United States	14
16	Oscillating a and b parameters, Ireland, Italy and United States	15
17	Periodic model, Ireland	17
18	Periodic model, Italy	17
19	Periodic model, United States	17
20	Multi-phase model, Ireland	18
21	Multi-phase model, Italy	18
22	Multi-phase model, United States	19
23	Multi-phase periodic model, Ireland	19
24	Multi-phase periodic model, Italy	19
25	Multi-phase periodic model, United States	20
26	Normality checks, Ireland, Italy and United States	25
27	Seasonal Holt Winter's Additive Model Algorithm (denoted SHW ₊)	25
28	Comparison between HoltWinters multiplicative (left, red outline) and additive (right, green outline)algorithms, at some point during the research	27
29	HoltWinters model, Ireland	27
30	HoltWinters model, Italy	27
31	HoltWinters model, United States	28
32	ARIMA model, Ireland	31
33	ARIMA model, Italy	31
34	ARIMA model, United States	32
35	A linear regression model, or ARIMA($p, 0, 0$) model.	32
36	A neural network with p inputs and one hidden layer with k hidden neurons.	33
37	Neural Network model, Ireland	34
38	Neural Network model, Italy	35
39	Neural Network model, United States	35
40	Wes Anderson Palettes	36
41	OWID World data extract	37
42	ArcGIS Ireland data extract	37

List of Code

1	Algorithm for Periodic Model	15
2	Plot Periodic Model	16
3	Algorithm for HoltWinters Model	25
4	Plot HoltWinters Model	26
5	Algorithm for ARIMA Model	30
6	Plot ARIMA Model	30
7	Algorithm for Neural Network Model	33
8	Plot Neural Network Model	34
9	Main plotting functions	37
10	Geospatial/Map plots	51
11	Saving plots	53

1 Introduction

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

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

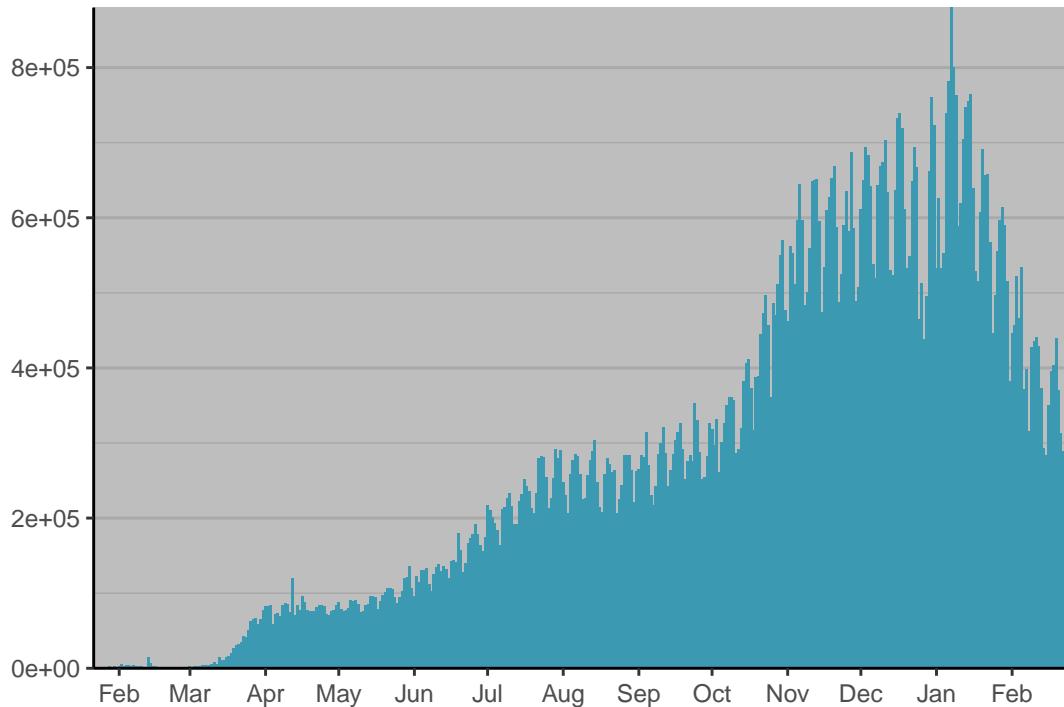


Figure 1: Daily Cases Globally

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

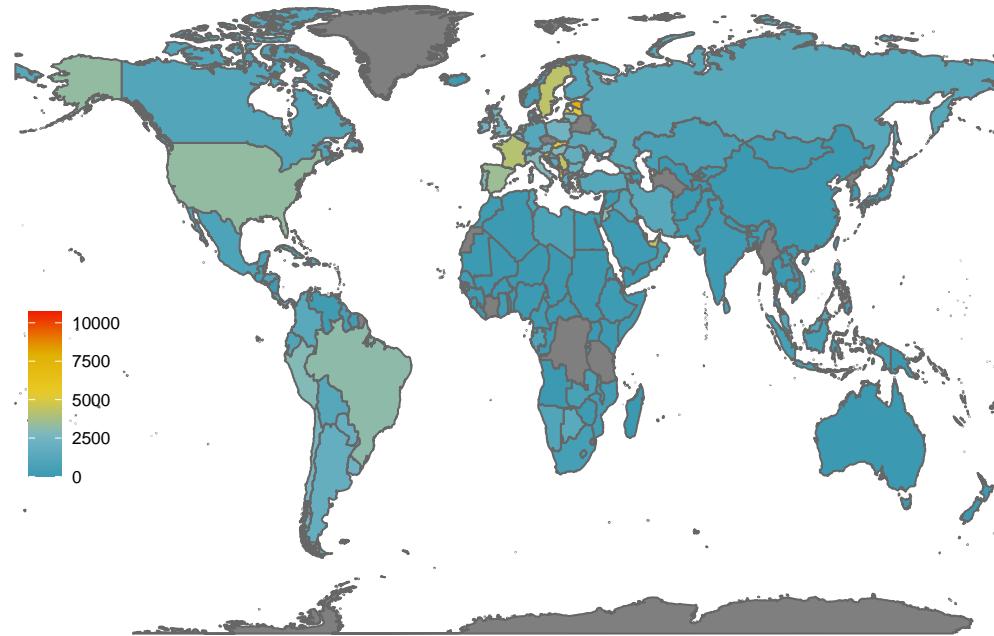


Figure 2: Cases per 1 million population, World

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

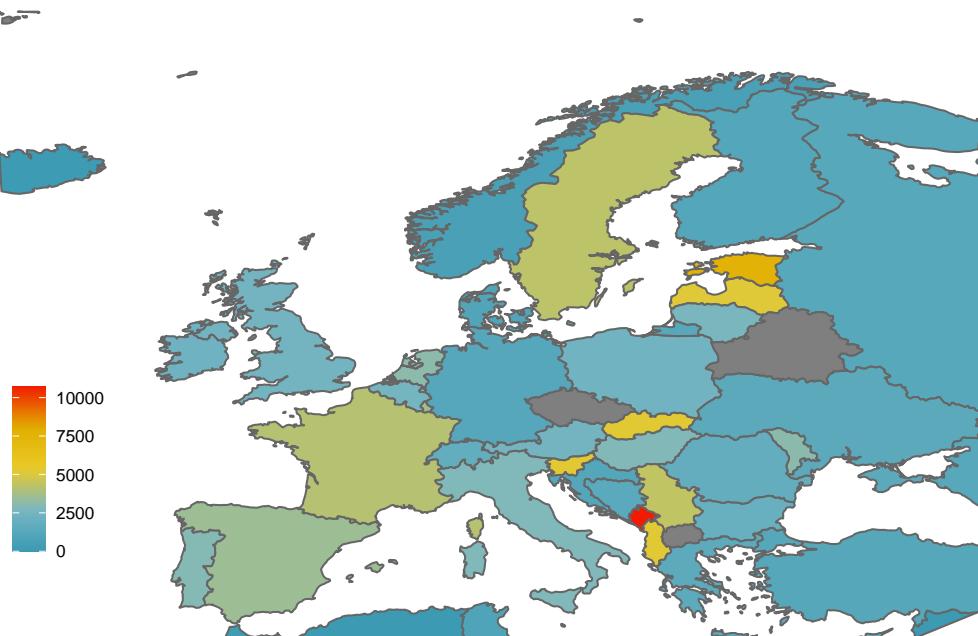


Figure 3: Cases per 1 million population, Europe

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

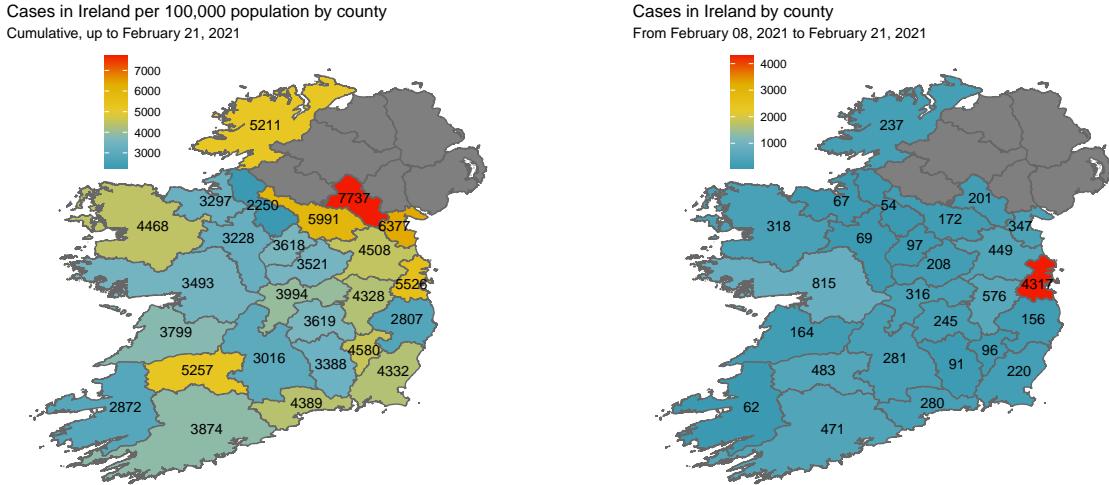


Figure 4: Situation by County, Ireland

1.1 Previous work on Covid-19 identification and modeling

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

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

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

1.2 Key aims

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

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

2 Mathematical Models

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

2.1 Base model

2.1.1 Definitions and Theory

2.1.2 How to select the best model

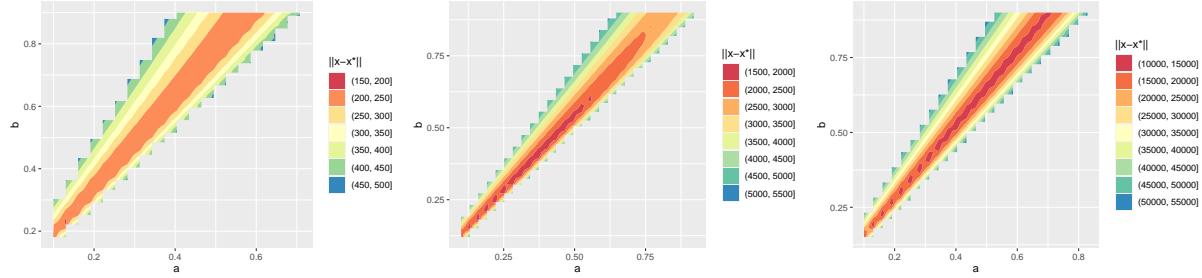


Figure 5: Contour plots of $\|x - x^*\|$, Ireland, Italy and United States

2.1.3 Forecasting

2.1.4 Implementation in R

2.1.5 Plots

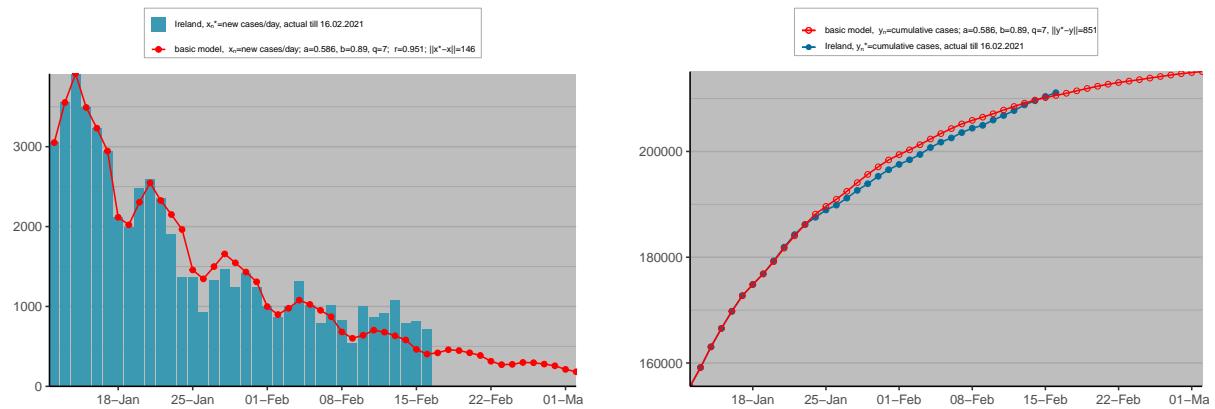


Figure 6: Basic model, Ireland

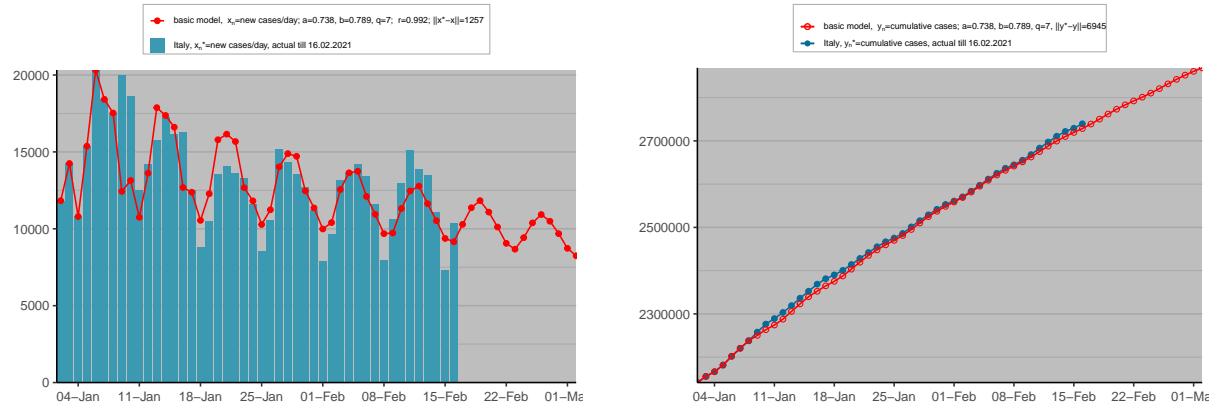


Figure 7: Basic model, Italy

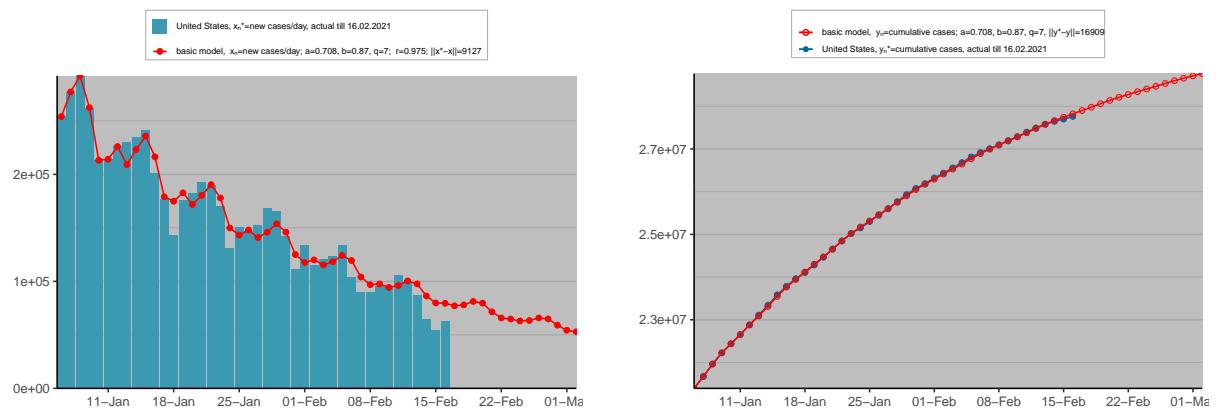


Figure 8: Basic model, United States

2.2 Limiting curve

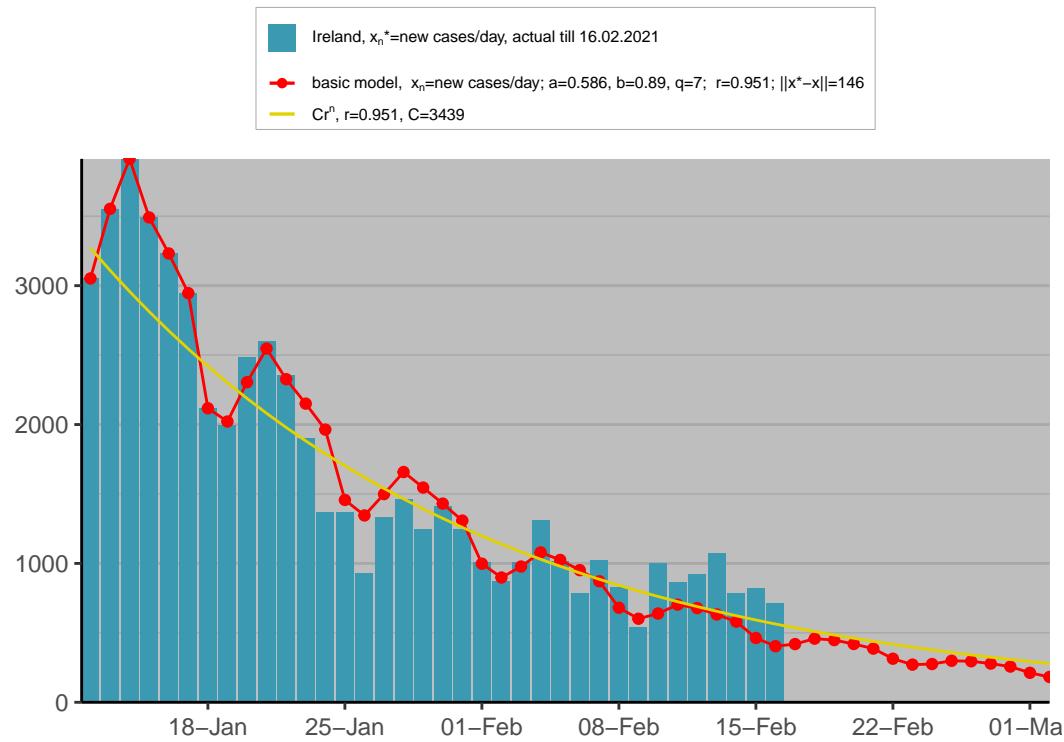


Figure 9: Comparison of x_n^* , x_n and the limiting curve Cr^n , Ireland

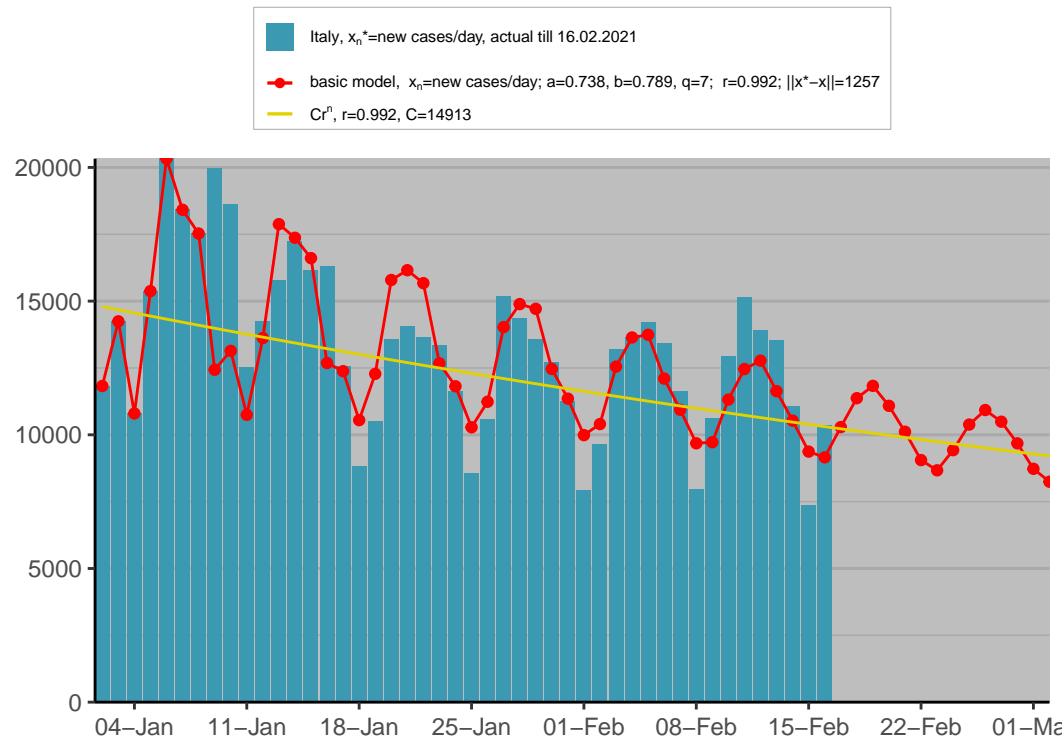


Figure 10: Comparison of x_n^* , x_n and the limiting curve Cr^n , Italy

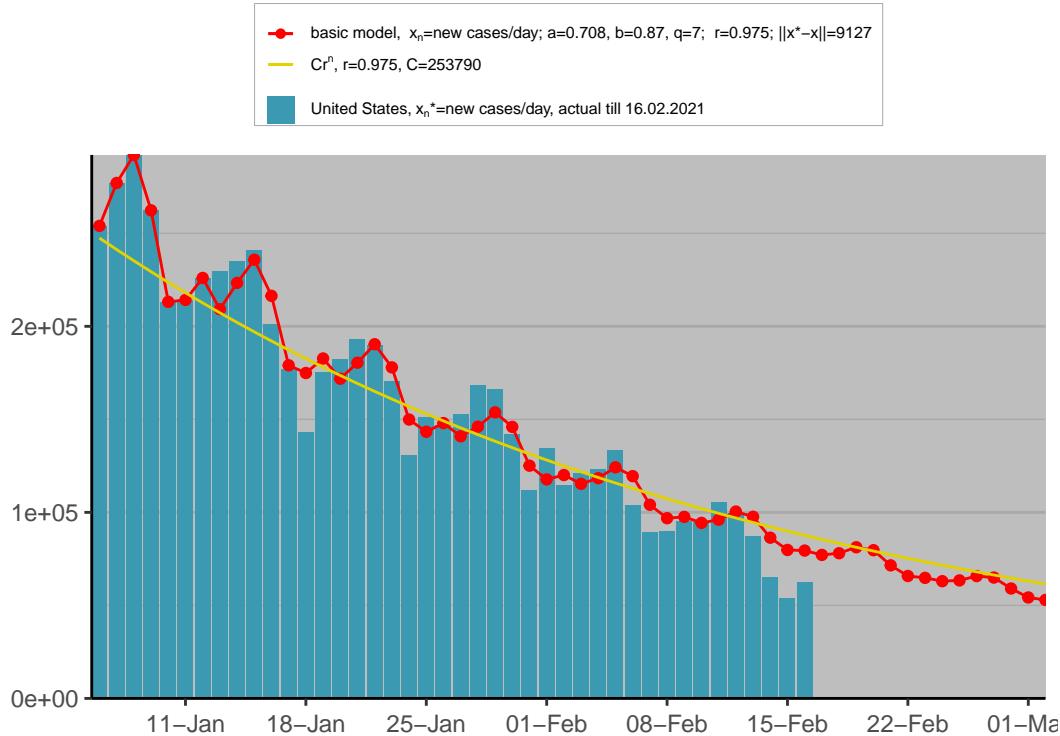


Figure 11: Comparison of x_n^* , x_n and the limiting curve Cr^n , United States

The limiting curve can of course so exponential growth

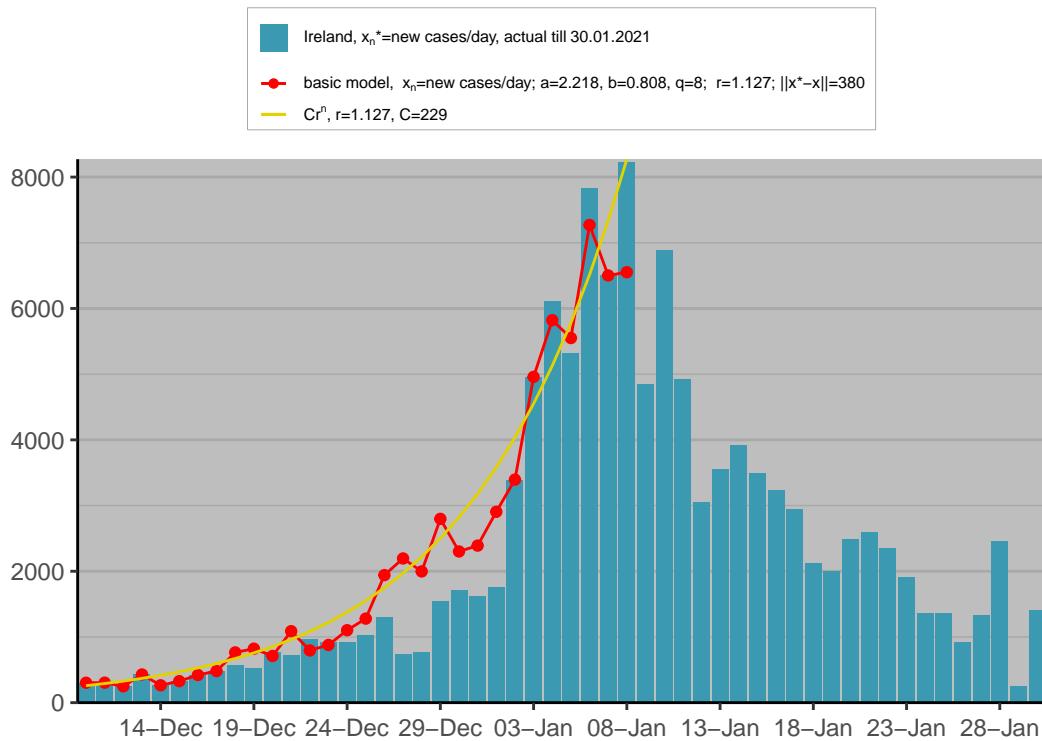


Figure 12: Limiting Curve Growing Exponentially, Ireland

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.

Good baseline for model performance

(want $\|x - x^*\| \approx \|x^*(3) - x^*\|$ or better)

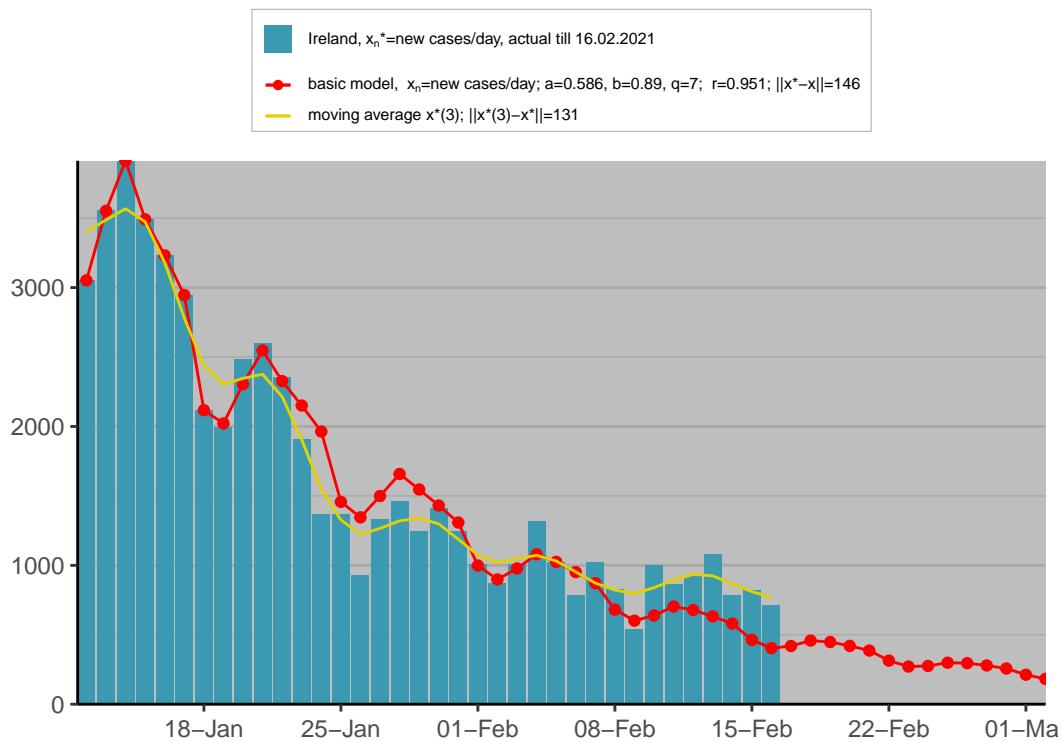


Figure 13: Moving average $x_n^*(3)$, Ireland

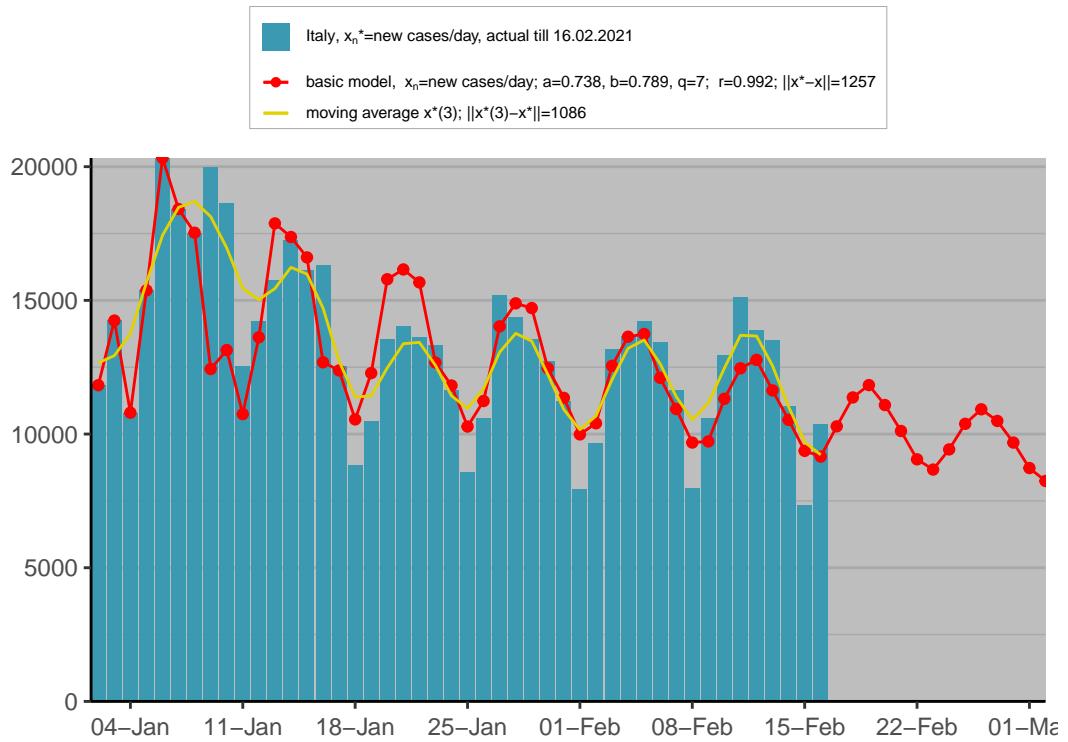


Figure 14: Moving average $x_n^*(3)$, Italy

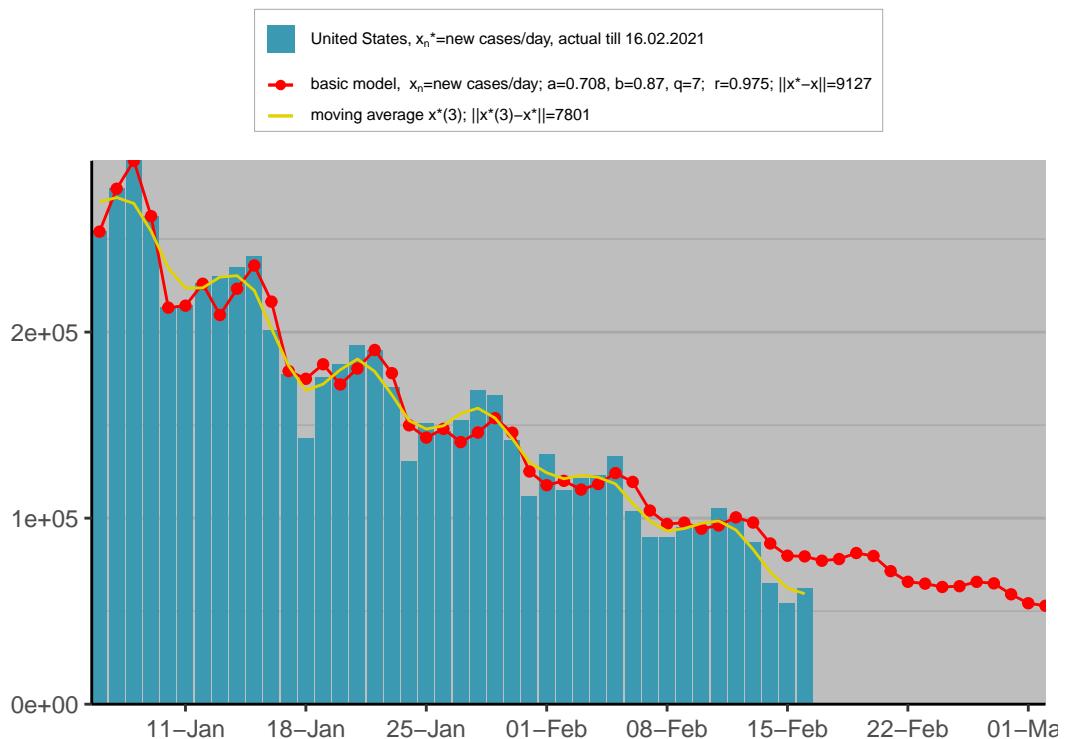


Figure 15: Moving average $x_n^*(3)$, United States

2.4 Periodic model

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

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

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

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

$c_i \in [0.04, 0.2]$ small

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

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

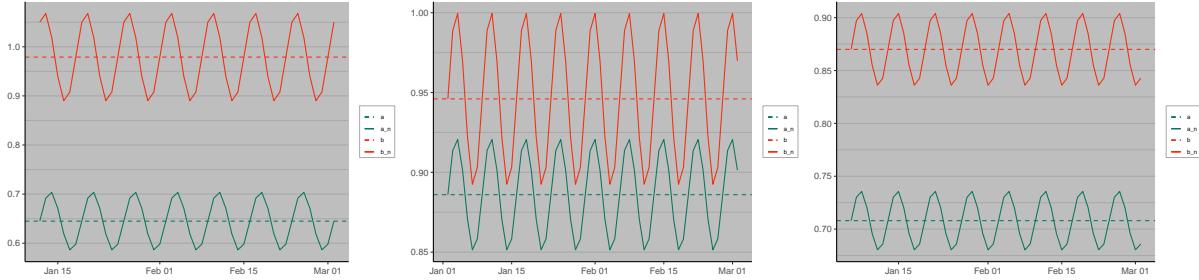


Figure 16: Oscillating a and b parameters, Ireland, Italy and United States

2.4.1 Definitions and Theory

2.4.2 How to select the best model

2.4.3 Forecasting

2.4.4 Implementation in R

```

1 modper <- 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
13 normper <- function(par, q, x) return(modnorm(x, modper(par, q, x)))
14
15 #parameters of the form (ci, pi, ni)
16 ##an = a(1 + c1 sin(2pi/p1 (n - n1)))
17 ##bn = b(1 + c2 sin(2pi/p2 (n - n2)))
18 aseqper <- a.seq(from = 0.8, to = 1.2, length.out = 5)
19 bseqper <- b.seq(from = 0.8, to = 1.2, length.out = 5)
20 c_1seq <- c_2seq <- seq(0.04, 0.2, length.out = 10)
21 n_1seq <- n_2seq <- c(1,7)
22 p_1seq <- p_2seq <- 6:7
23 normdatp <- expand.grid(a = aseqper, b = bseqper,
24                           c1 = c_1seq, c2 = c_2seq,
25                           p1 = p_1seq, p2 = p_2seq,
26                           n1 = n_1seq, n2 = n_2seq)
27
28 pernorm <- apply(normdatp, 1, function(x) normper(x, q = q, countrydat$xn))
29 normdatp$pernorm <- pernorm
30
31 peroptim    <- normdatp[which.min(pernorm),1:8]
32 optpernorm <- normdatp[which.min(pernorm),9]
33
34 modeldat$periodic <- modper(as.numeric(peroptim),countrydat$xn, q = q, forecastlen)
35 modeldat$periodicy <- xntoy(n=10)(modeldat$periodic) + prevcases

```

```

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

```

Listing 1: Algorithm for Periodic Model

and our plotting function for daily cases looks like the following

```

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

```

Listing 2: Plot Periodic Model

2.4.5 Plots

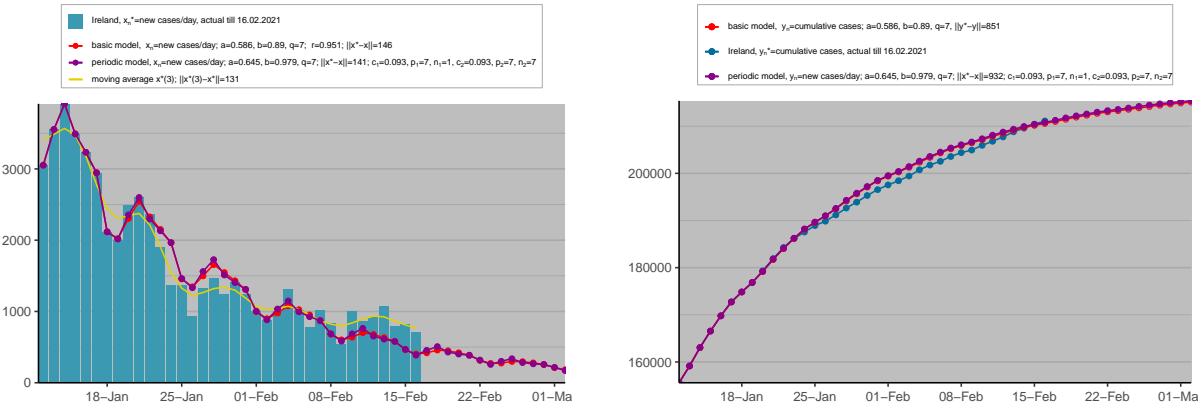


Figure 17: Periodic model, Ireland

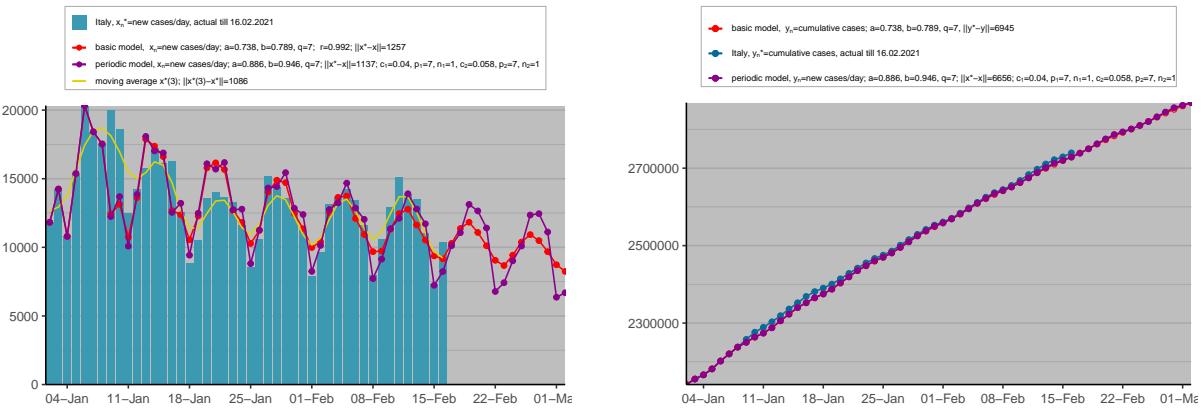
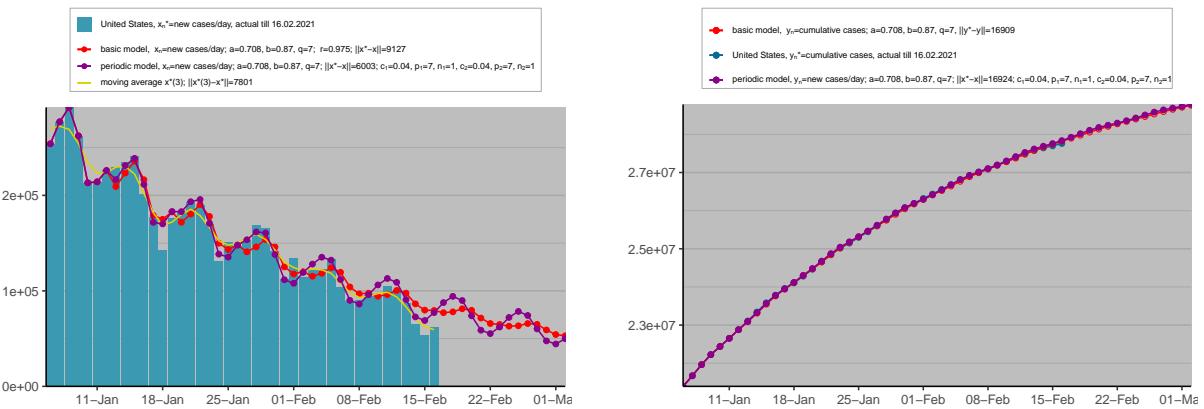


Figure 18: Periodic model, Italy



2.5 Multi-phase model

2.5.1 Definitions and Theory

2.5.2 How to select the best model

2.5.3 Forecasting

2.5.4 Implementation in R

2.5.5 Plots

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

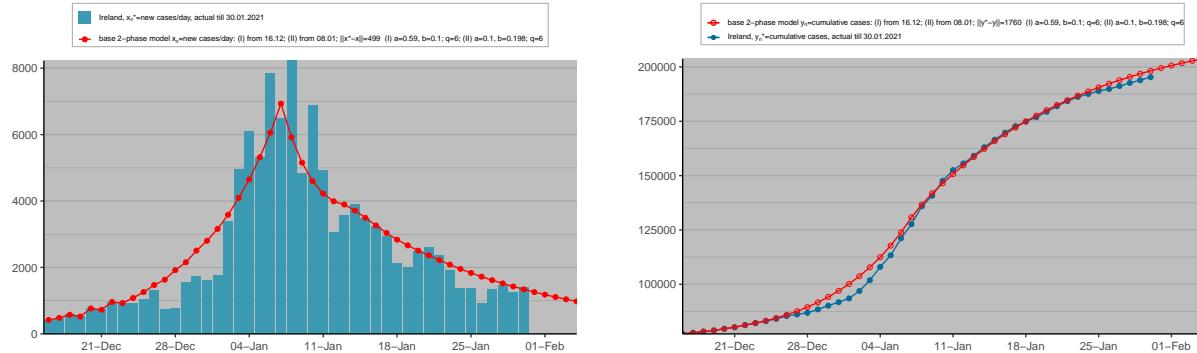


Figure 20: Multi-phase model, Ireland

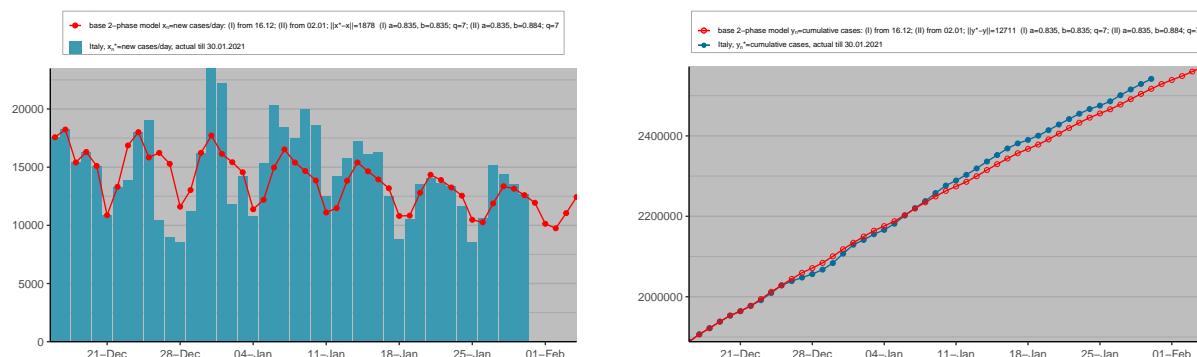


Figure 21: Multi-phase model, Italy

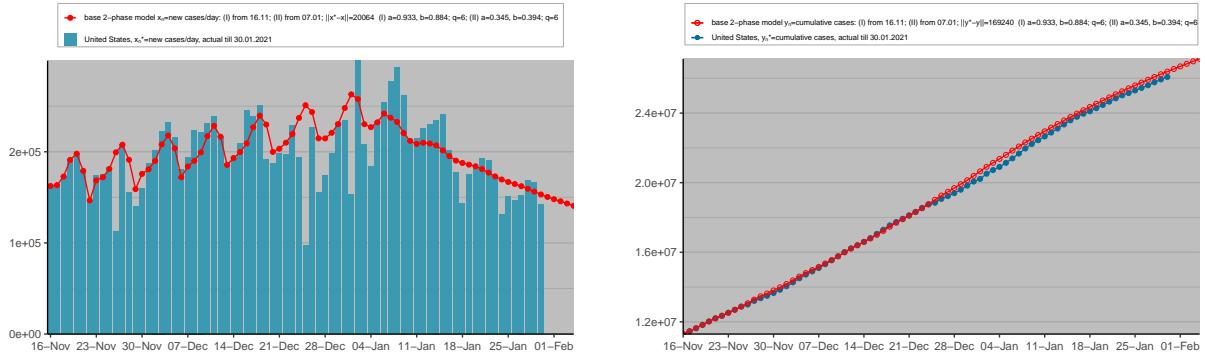


Figure 22: Multi-phase model, United States

The periodic model often performs much better

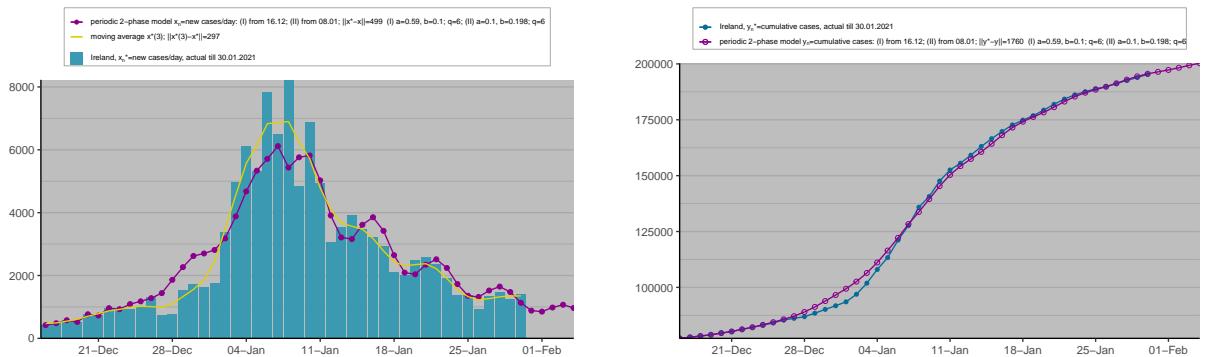


Figure 23: Multi-phase periodic model, Ireland

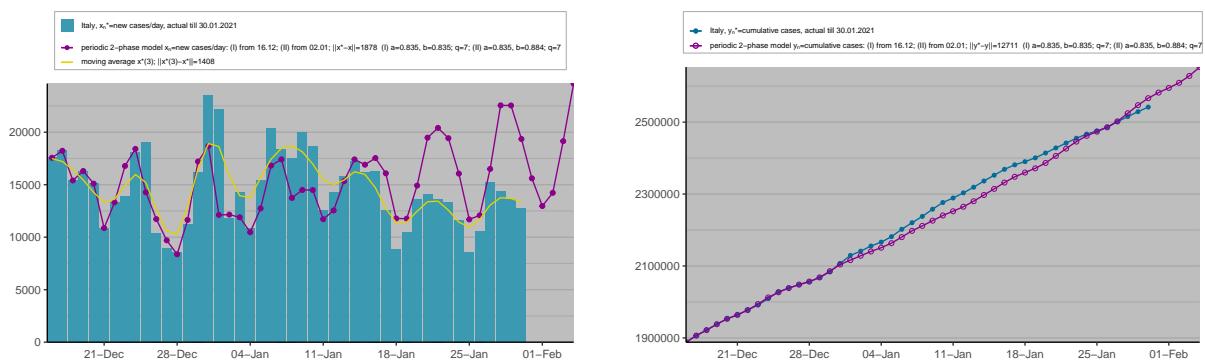


Figure 24: Multi-phase periodic model, Italy

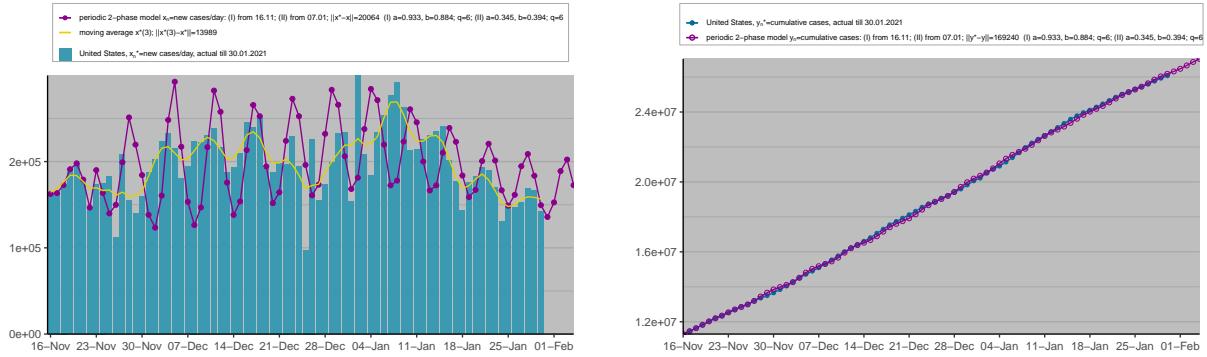


Figure 25: Multi-phase periodic model, United States

3 Theorems for Mathematical Model

3.1 Model Assumptions

- (I) Any infected person becomes ill (symptomatic) and infectious on the q -th day after infection.¹
- (A) During each day, each ill person unconfined infects on average a other persons.
- (B) During each day, a fraction b of ill people loose gets isolated (hospitalized or otherwise) and withdrawn from a further spread of the epidemic.

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

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

3.2 Notation

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

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

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

Recurrence relation for z_n :

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

Using $x_n = bz_n$ 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 want to minimize both

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

3.2.1 Why minimize both diatances?

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

3.2.2 Recurrence equation

This is our general linear recurrence equation with constant coefficients:

$$x_{n+1} = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \dots + a_q x_{n-q} \quad (7)$$

The characteristic polynomial of 7

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

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

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.

3.3 Theorems

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 8 has exactly one positive real root r . Besides, the root r is simple and, for any other root $\lambda \in \mathbb{C}$, we have $|\lambda| < r$. Consequently, r is the leading root of 7.

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

$$x_n \sim Cr^n \text{ as } n \rightarrow \infty. \quad (9)$$

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

Proof:

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

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

dividing across by λ^{q+1}

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

And so

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

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

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

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

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

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

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

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

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

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

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

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

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

$$x_n + Cr^n + \tilde{x}_n \quad (11)$$

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

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

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

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

This satisfies the equation

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

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

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

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

$$A_0 + A_1 + \cdots + A_q = 1 \quad (14)$$

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

$$x_n \geq cr^n$$

as required. □

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

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

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

Proof:

- (a) We have

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

We know f is increasing.

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, \dots, 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

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

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

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

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

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

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

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

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

□

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

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

Proof: Let u be the number of people infected on some day, say 0. On the day $k = 1, \dots, q$ 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

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

since $0 < 1-b < 1$.

other people.

Hence, the initial group of u people infects in total

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

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

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

□

4 Statistical Models

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

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

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

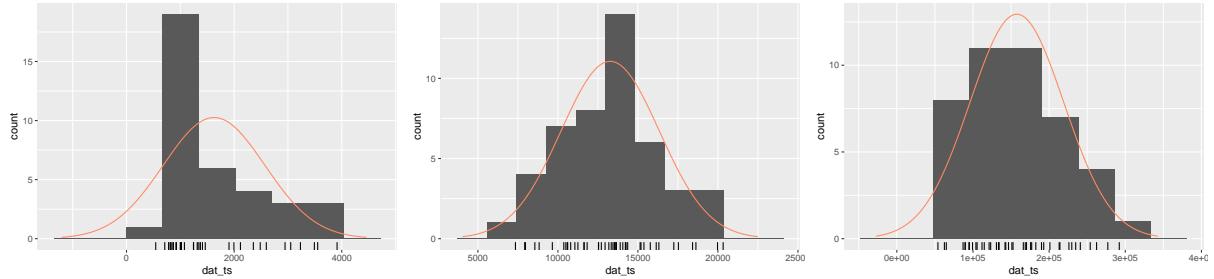


Figure 26: Normality checks, Ireland, Italy and United States

4.1 Holt-Winters' seasonal method

4.1.1 Definitions and Theory

Suppose there are N observations.

Initial step:

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

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 27: Seasonal Holt Winter's Additive Model Algorithm (denoted SHW₊)

4.1.2 How to select the best model

4.1.3 Forecasting

4.1.4 Implementation in R

```

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

```

```

8| #' prevcases The cumulative cases prior to the first day in the range of countrydat
9|
10| #lambda=0 ensures values stay positive
11| hwfcast <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
12| #ensure first q values are equal to the actual data
13| hwfcast$fitted[1:q] <- countrydat$xn[1:q]
14| modeldat$hwxn <- c(hwfcast$fitted, hwfcast$mean)
15| modeldat$hwlo <- c(hwfcast$fitted, hwfcast$lower[,2])
16| modeldat$hwhi <- c(hwfcast$fitted, hwfcast$upper[,2])
17|
18| modeldat$hwyn <- xntoyn(modeldat$hwxn)+prevcases
19| modeldat$hwylo <- xntoyn(modeldat$hwlo)+prevcases
20| modeldat$hwyhi <- xntoyn(modeldat$hwhi)+prevcases
21|
22| hwnorm <- modnorm(countrydat$xn, hwfcast$fitted)
23|
24| labs$hw <- paste0("HoltWinters algorithm, ||x-x|| =", modnorm(countrydat$xn, hwfcast$fitted))
25| labs$hwy <- paste0("HoltWinters algorithm, ||y-y|| =", modnorm(countrydat$yn, modeldat$hwyn[1:nrow(countrydat)]))
26| labs$hwpi <- "HW 95% Prediction Interval"
27|
28| plots[[ "hw" ]] <- plot_hw(countrydat, modeldat, cols, labs)
29| plots[[ "hwy" ]] <- plot_hwy(countrydat, modeldat, cols, labs)

```

Listing 3: Algorithm for HoltWinters Model

and our plotting function for daily cases looks like the following

```

1 plot_hw <- function(countrydat, modeldat, cols, labs){
2   p <- ggplot(countrydat, binwidth = 0) +
3     geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
4     geom_ribbon(data = modeldat, aes(x = date, ymin = hwlo, ymax = hwhi, fill = "hw"),
5                 alpha = 0.5) +
6     geom_point(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
7     geom_line(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
8     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
9     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
10    geom_point(data = modeldat, aes(x = date, y = modxPeriodic, colour = "periodic")) +
11    geom_line(data = modeldat, aes(x = date, y = modxPeriodic, colour = "periodic")) +
12    gg_scale_xy +
13    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
14           fill=guide_legend(ncol=1,nrow=2,byrow=TRUE)) +
15    scale_fill_manual(labels = c("actual" = labs$xn, "hw" = labs$hwpi),
16                      values = c("actual" = cols$xn, "hw" = cols$hwpi))+ +
17    scale_colour_manual(labels = c("base" = labs$basexn, "hw" = labs$hw, "periodic" = labs$periodic),
18                        values = c("base" = cols$basexn, "hw" = cols$hw, "periodic" = cols$periodic))
19  }
20  return(p)
21}

```

Listing 4: Plot HoltWinters Model

4.1.5 Plots

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

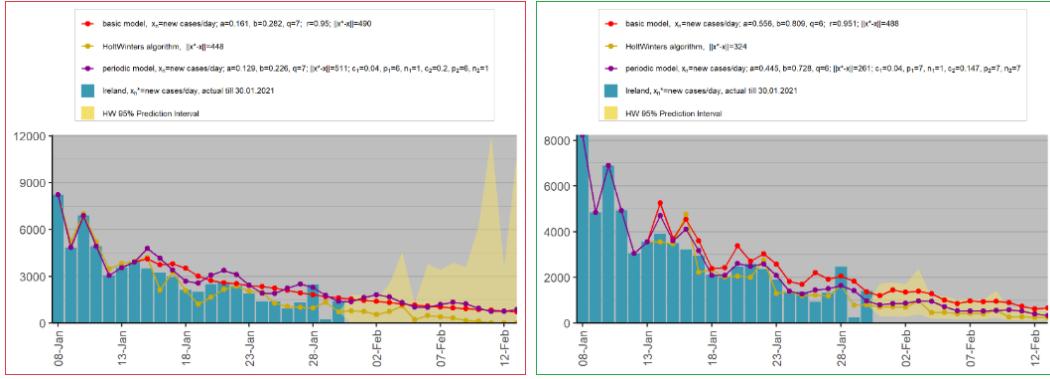


Figure 28: Comparison between HoltWinters multiplicative (left, red outline) and additive (right, green outline) algorithms, at some point during the research

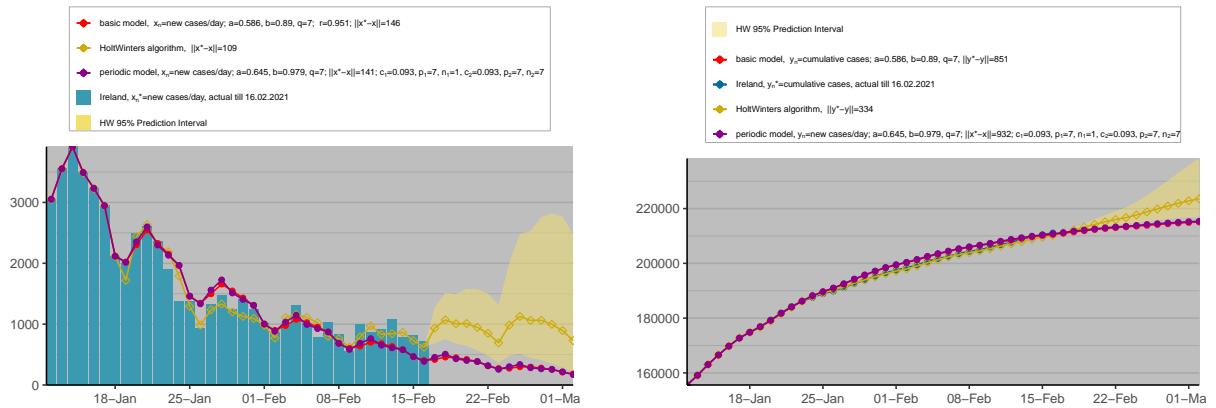


Figure 29: HoltWinters model, Ireland

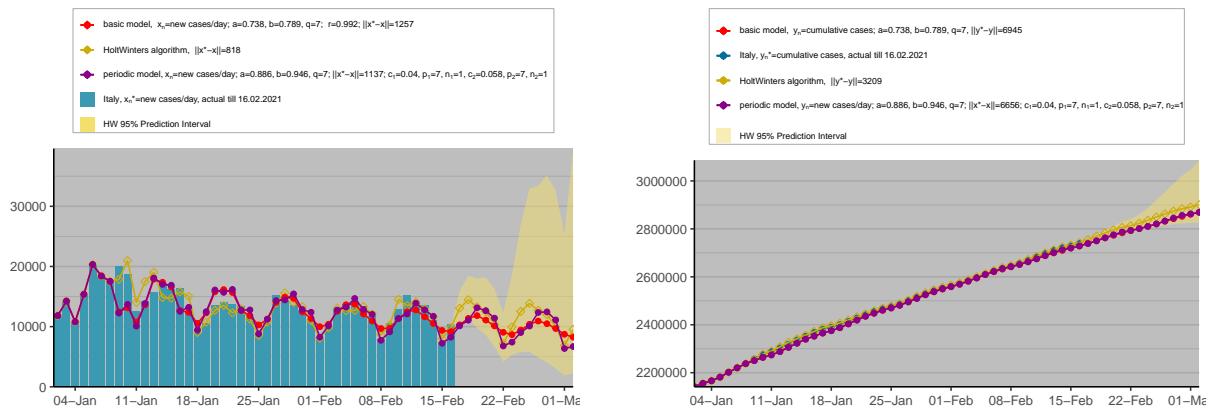


Figure 30: HoltWinters model, Italy

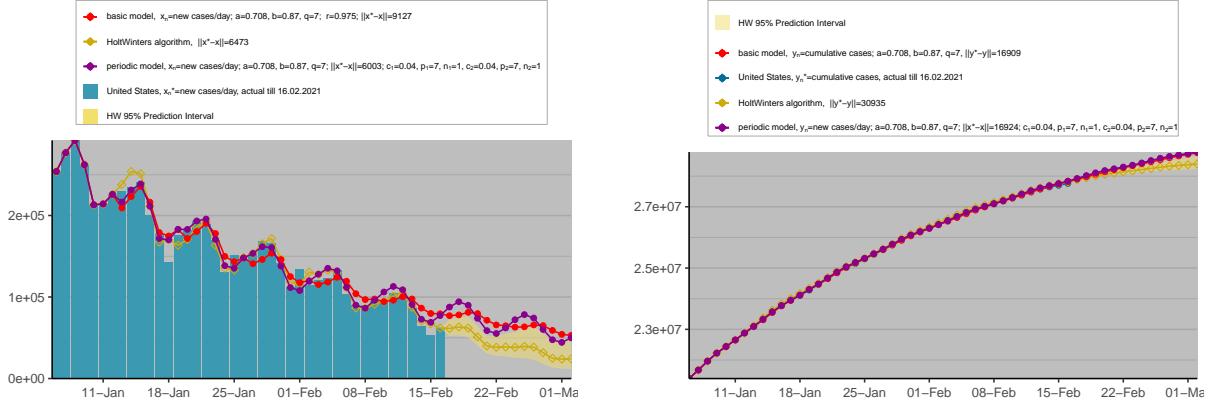


Figure 31: HoltWinters model, United States

4.2 ARIMA models

4.2.1 Definitions and Theory

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

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

- 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 :

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

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

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

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

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

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

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

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

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

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

We also need the seasonal components for an ARIMA(p, d, q)(P, D, Q) _{s}

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

- An $AR(P)_s$ model, or a seasonal autoregressive model of order P of a time series x_1, \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 :

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

- 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 :

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

- 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 the previous ARIMA(p, d, q) to get the definition of an ARIMA(p, d, q)(P, D, Q) _{s} model

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

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

4.2.2 How to select the best model

4.2.3 Forecasting

4.2.4 Implementation in R

```

1 #' dat_ts A time-series of the observations xn
2 #' countrydat A data.frame with the required info to plot case numbers
3 #' modeldat A data.frame with model data to plot (has forecastlen more rows than countrydat)
4 #' cols A list of colours for plotting
5 #' labs A list of labels for plotting
6 #' forecastlen The forecast length
7 #' q The parameter for the basic model, in order to have statistical models comparable with
     mathematical models
8 #' prevcases The cumulative cases prior to the first day in the range of countrydat
9
10 #lambda=0 ensures values stay positive
11 auto.fit <- auto.arima(dat_ts, lambda = 0)
12
13 getArmaModel <- function(arma, pdq = c(1,6,2), PDQ = c(3,7,4), s=5){
14   return(paste0("ARIMA(", paste0(arma[pdq], collapse = ","), ")",
15                 paste0(arma[PDQ], collapse = ","), ")[, arma[s], "])")
16 }
17
18 arima.fcst <- forecast(auto.fit, level = c(80, 95), h = forecastlen)
19 arima.fcst$fitted[1:q] <- countrydat$xn[1:q]
20
21 arimanorm <- modnorm(countrydat$xn, arima.fcst$fitted)
22
23 arimalabs <- getArmaModel(auto.fit$arma)
24 abs$arima <- paste0(arimalabs, ", ||x*-x||=", arimanorm)
25 abs$arimapi <- "ARIMA 95% Prediction Interval"
26
27 modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
28 modeldat$arimalo <- c(auto.fit$fitted, arima.fcst$lower[,2])
29 modeldat$arimahi <- c(auto.fit$fitted, arima.fcst$upper[,2])
30
31 modeldat$arimaxn[1:q] <- countrydat$xn[1:q]
32 modeldat$arimalo[1:q] <- countrydat$xn[1:q]
33 modeldat$arimahi[1:q] <- countrydat$xn[1:q]
34
35 modeldat$arimayn <- xntoyn(modeldat$arimaxn) + prevcases
36 modeldat$arimaylo <- xntoyn(modeldat$arimalo) + prevcases
37 modeldat$arimayhi <- xntoyn(modeldat$arimahi) + prevcases
38
39 labs$arimay <- paste0(arimalabs, ", ||y*-y||=", modnorm(countrydat$yn, modeldat$arimayn[1:
     nrow(countrydat)]))
40
41 plots[["arima"]] <- plot_arima(countrydat, modeldat, cols, labs)
42 plots[["arimay"]] <- plot_arimay(countrydat, modeldat, cols, labs)

```

Listing 5: Algorithm for ARIMA Model

and our plotting function for daily cases looks like the following

```

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

```

```

17 |     labels = c("arima" = labs$arima, "base" = labs$basexn, "periodic" = labs$|
18 |     xntheme()|
19 |     return(p)|
20 |

```

Listing 6: Plot ARIMA Model

4.2.5 Plots

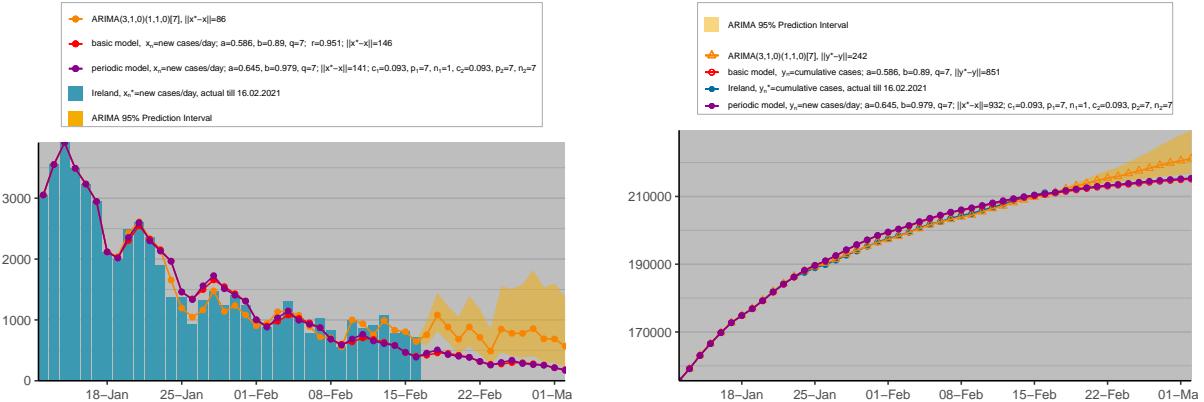


Figure 32: ARIMA model, Ireland

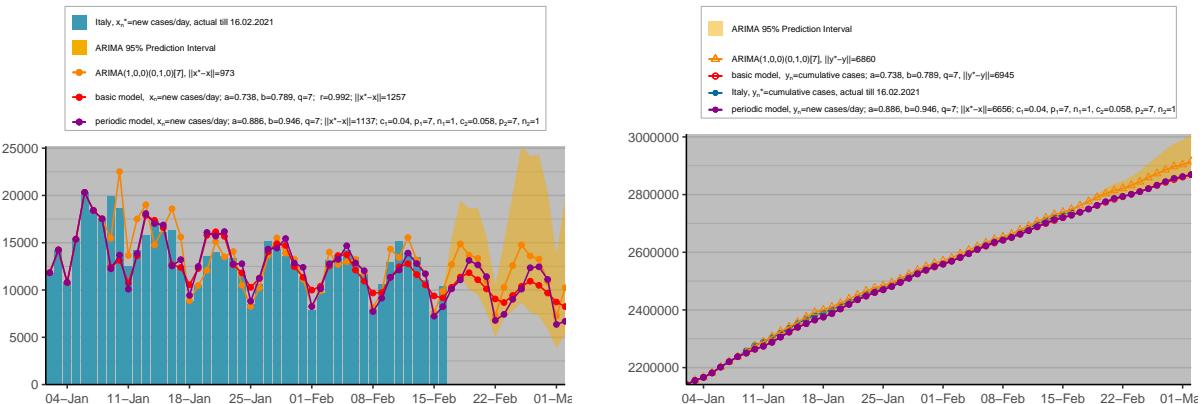


Figure 33: ARIMA model, Italy

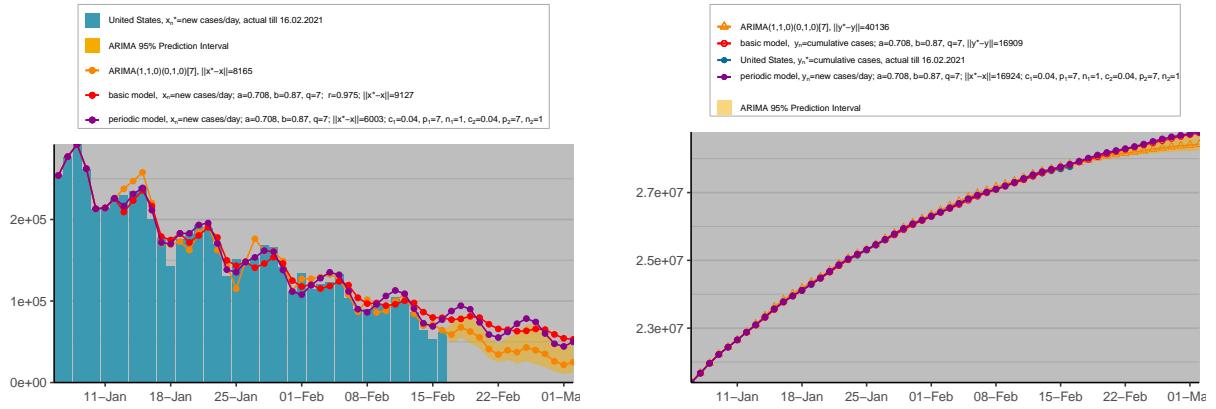


Figure 34: ARIMA model, United States

4.3 Neural network models

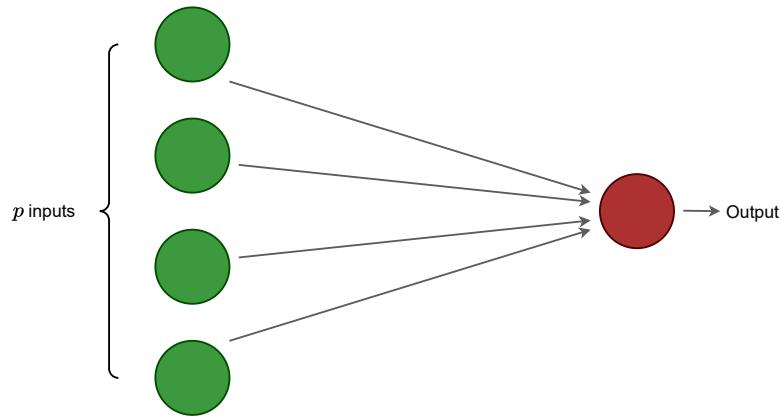


Figure 35: A linear regression model, or ARIMA($p, 0, 0$) model.

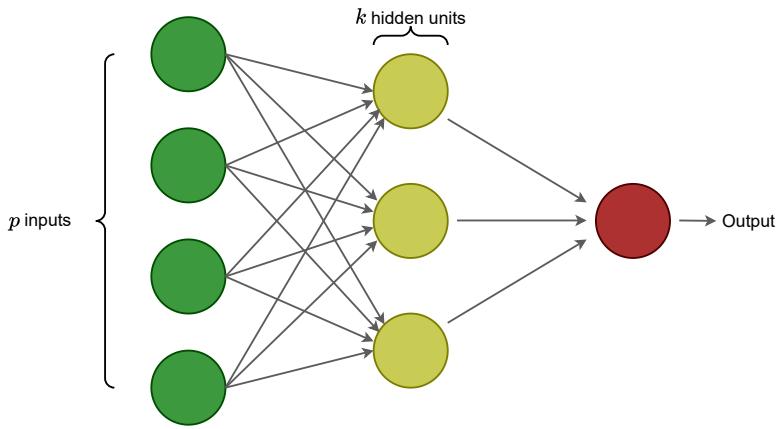


Figure 36: A neural network with p inputs and one hidden layer with k hidden neurons.

4.3.1 Definitions and Theory

4.3.2 How to select the best model

4.3.3 Forecasting

4.3.4 Implementation in R

```

1 #', dat_ts A time-series of the observations x_n
2 #', countrydat A data.frame with the required info to plot case numbers
3 #', modeldat A data.frame with model data to plot (has forecastlen more rows than countrydat)
4 #', cols A list of colours for plotting
5 #', labs A list of labels for plotting
6 #', forecastlen The forecast length
7 #', q The parameter for the basic model, in order to have statistical models comparable with
     mathematical models
8 #' prevcases The cumulative cases prior to the first day in the range of countrydat
9
10 #requires arima auto.fit object from earlier
11 nHidden <- max(1, floor(0.5*(1+auto.fit$arma[1]+auto.fit$arma[3])))
12 #Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.
13 nnfit <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda =
     0, repeats = 20, maxit = 50)
14 nn.fcst <- forecast(nnfit, h = forecastlen)
15
16 nn.fcst$mean[nn.fcst$mean < 0] <- 0
17 nn.fcst$fitted[1:q] <- countrydat$xn[1:q]
18
19 modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
20 modeldat$nnyn <- xtoyn(modeldat$nnxn) + prevcases
21
22 labs$nn <- paste0(nnfit$method, ", ||x-x||=", modnorm(countrydat$xn,nn.fcst$fitted))
23 labs$nnyn <- paste0(nnfit$method, ", ||y-y||=", modnorm(countrydat$yn,modeldat$nnyn[1:nrow(
     countrydat)]))
24
25 plots[["nn"]] <- plot_nn(countrydat, modeldat, cols, labs)
26

```

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

Listing 7: Algorithm for Neural Network Model

and our plotting function for daily cases looks like the following

```
1 plot_nn <- function(countrydat, modeldat, cols, labs){
2   p <- ggplot(countrydat, binwidth = 0) +
3     geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
4     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
5     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
6     geom_point(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
7     geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
8     geom_point(data = modeldat, aes(x = date, y = nnxn, colour = "nn")) +
9     geom_line(data = modeldat, aes(x = date, y = nnxn, colour = "nn")) +
10    gg_scale_xy +
11    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
12           fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
13    scale_fill_manual(values = c("actual" = cols$xn),
14                      labels = c("actual" = labs$xn)) +
15    scale_colour_manual(values = c("base" = cols$basexn, "nn" = cols$nn, "periodic" = cols$periodic),
16                      labels = c("base" = labs$basexn, "nn" = labs$nn, "periodic" = labs$periodic)) +
17    xntHEME()
18  return(p)
19 }
```

Listing 8: Plot Neural Network Model

4.3.5 Plots

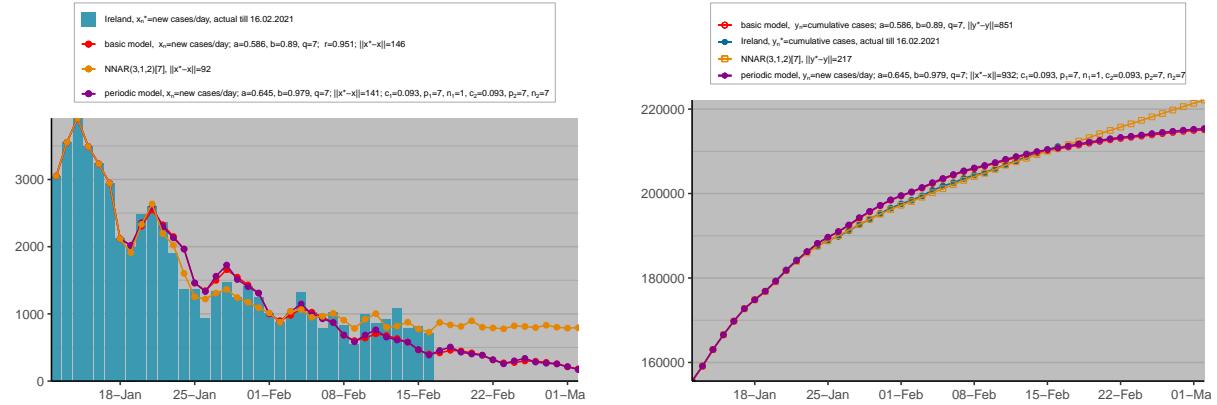


Figure 37: Neural Network model, Ireland

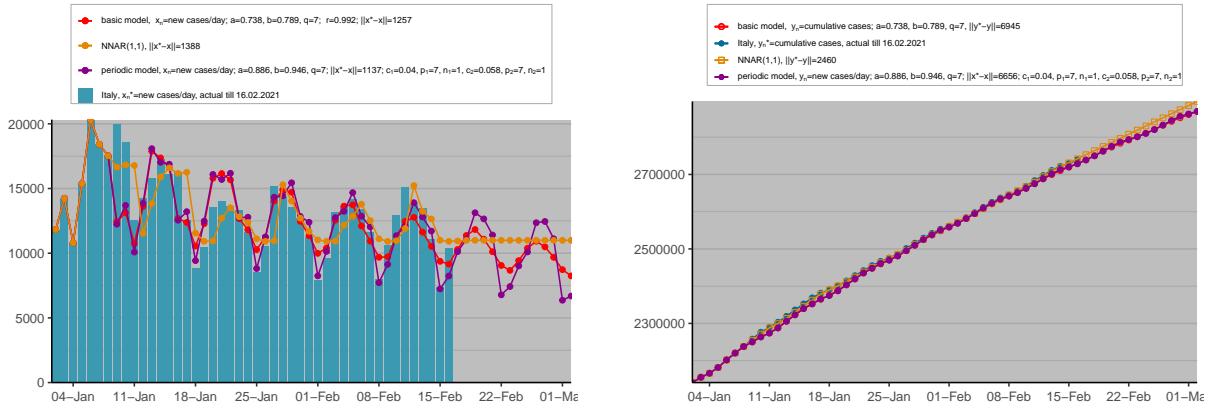


Figure 38: Neural Network model, Italy

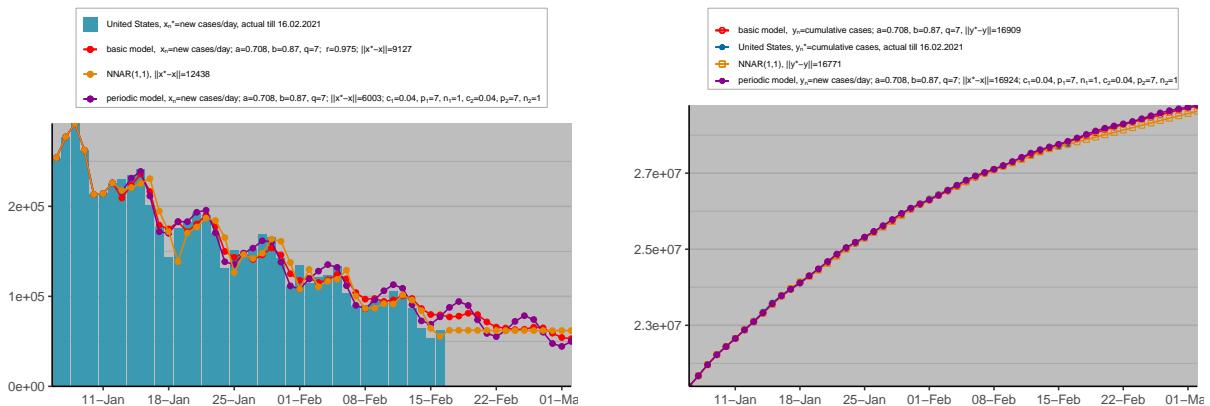


Figure 39: Neural Network model, United States

5 R Code and Data Sources

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

5.1 R packages

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

5.2 Plotting and colour

wesanderson [22]



Figure 40: Wes Anderson Palettes

5.3 Shapefiles

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

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

5.4 Datasets

Country-based data:

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

Ireland cases by county Downloaded from [7].

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

Figure 41: OWID World data extract

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

Figure 42: ArcGIS Ireland data extract

A Appendix

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

```

1 require(ggplot2)
2 require(forecast)
3 require(dplyr)
4 require(wesanderson)
5 require(gridExtra)
6 #webdat <- read.csv("https://opendata.ecdc.europa.eu/covid19/casedistribution/csv")
7 #owiddat <- read.csv("https://covid.ourworldindata.org/data/owid-covid-data.csv")
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 plot_xn <- function(countrydat, cols, labs){
14   p <- ggplot(countrydat, binwidth = 0) +
15     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
16     scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+
17     scale_y_continuous(expand = c(0,0)) +
18     scale_fill_manual(values = cols$xn, name = "", labels = labs$xn) +
19     xntheme()
20   return(p)
21 }
22
23 plot_yn <- function(countrydat, cols, labs){
24   p <- ggplot(countrydat) +
25     geom_line(aes(x = date, y = yn, colour = "blue")) +
26     geom_point(aes(x = date, y = yn, colour = "blue"))+
27     scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+
28     scale_y_continuous(expand = c(0,0)) +
29     scale_colour_manual(values = cols$yn, name = "", labels = labs$yn) +
30     yntheme()
31   return(p)
32 }
33
34 plot_basexn <- function(countrydat, modeldat, cols, labs){
35   p <- ggplot(countrydat, binwidth = 0) +
36     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
37     geom_point(data = modeldat, aes(x = date, y = basexn, colour = cols$basexn)) +

```

```

38|     geom_line(data = modeldat, aes(x = date, y = basexn, colour = cols$basexn)) +
39|     gg_scale_xy +
40|     scale_fill_manual(name = "leg", values = cols$xn, labels = labs$xn) +
41|     scale_colour_manual(name = "leg", values = cols$basexn, labels = labs$basexn) +
42|     xntheme()
43|   return(p)
44| }
45|
46 plot_baseyn <- function(countrydat, modeldat, cols, labs){
47   p <- ggplot(countrydat) +
48     geom_point(aes(x = date, y = yn, colour = "blue")) + geom_line(aes(x = date, y = yn, colour =
49       "blue")) +
50     geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
51     geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
52     gg_scale_xy +
53     scale_colour_manual(name = "leg", values = c("blue" = cols$yn, "base" = cols$baseyn),
54       labels = c("blue" = labs$yn, "base" = labs$baseyn),
55       guide = guide_legend(override.aes = list(
56         shape = c("blue"=1, "base" = 16)))) +
57     yntheme()
58   return(p)
59 }
60|
61 plot_crn <- function(countrydat, modeldat, cols, labs){
62   p <- ggplot(countrydat, binwidth = 0) +
63     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
64     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
65     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
66     geom_line(data = modeldat, aes(x = date, y = Crn, colour = "Crn")) +
67     gg_scale_xy +
68     scale_fill_manual(name = "leg", values = cols$xn, labels = labs$xn) +
69     scale_colour_manual(name = "leg",
70       values = c("basexn" = cols$basexn, "Crn" = cols$Crn),
71       labels = c("basexn" = labs$basexn, "Crn" = labs$Crn),
72       guide = guide_legend(override.aes = list(
73         shape = c("basexn" = 16, "Crn" = NA)))) +
74     xntheme()
75   return(p)
76 }
77|
78 plot_mavgx3 <- function(countrydat, modeldat, cols, labs){
79   p <- ggplot(countrydat, binwidth = 0) +
80     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
81     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
82     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "basexn")) +
83     geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
84     scale_fill_manual(values = cols$xn, labels = labs$xn) +
85     scale_colour_manual(values = c("basexn" = cols$basexn, "x3" = cols$x3),
86       labels = c("basexn" = labs$basexn, "x3" = labs$x3),
87       guide = guide_legend(override.aes = list(
88         shape = c("basexn" = 16, "x3" = NA)))) +
89     gg_scale_xy + xntheme()
90   return(p)
91 }
92|
93 plot_periodic <- function(countrydat, modeldat, cols, labs){
94   p <- ggplot(countrydat, binwidth = 0) +
95     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
96     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
97     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
98     geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
99     geom_point(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
100    geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
101    gg_scale_xy +
102    guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
103           fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
104    scale_fill_manual(values = cols$xn, labels = labs$xn) +
105    scale_colour_manual(values = c("base" = cols$basexn, "periodic" = cols$periodic, "x3" = cols$xn),
106      labels = c("base" = labs$basexn, "periodic" = labs$periodic, "x3" = labs$x3)) +
107    guides(colour = guide_legend(override.aes = list(shape = c("base" = 16, "periodic" = 16, "x3" = NA)))) +
108    xntheme()

```

```

108|     return(p)
109| }
110|
111| plot_periodicy <- function(countrydat, modeldat, cols, labs){
112|   p <- ggplot(countrydat) +
113|     geom_point(aes(x = date, y = yn, colour = "blue")) +
114|     geom_line(aes(x = date, y = yn, colour = "blue")) +
115|     geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
116|     geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
117|     geom_point(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
118|     geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
119|     gg_scale_xy +
120|     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE)) +
121|     scale_colour_manual(values = c("base" = cols$baseyn, "blue" = cols$yn, "periodic" = cols$periodic),
122|                           labels = c("base" = labs$baseyn, "blue" = labs$yn, "periodic" = labs$periodicy),
123|                           guide = guide_legend(override.aes = list(
124|                             shape = c("base" = 1, "blue"=16, "periodic"=16)))) +
125|     yntheme()
126|   return(p)
127| }
128|
129| plot_hw <- function(countrydat, modeldat, cols, labs){
130|   p <- ggplot(countrydat, binwidth = 0) +
131|     geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
132|     geom_ribbon(data = modeldat, aes(x = date, ymin = hwlo, ymax = hwhi, fill = "hw"), alpha = 0.5) +
133|     geom_point(data = modeldat, aes(x = date, y = hwxn, colour = "hw"), shape = 5) +
134|     geom_line(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
135|     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
136|     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
137|     geom_point(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
138|     geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
139|     gg_scale_xy +
140|     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
141|            fill=guide_legend(ncol=1,nrow=2,byrow=TRUE)) +
142|     scale_fill_manual(labels = c("actual" = labs$xn, "hw" = labs$hwpi),
143|                       values = c("actual" = cols$xn, "hw" = cols$hwpi))+ +
144|     scale_colour_manual(labels = c("base" = labs$basexn, "hw" = labs$hw, "periodic" = labs$periodic),
145|                           values = c("base" = cols$basexn, "hw" = cols$hw, "periodic" = cols$periodic))
146|   ) +
147|   xntheme()
148|   return(p)
149| }
150|
151| plot_hwy <- function(countrydat, modeldat, cols, labs){
152|   p <- ggplot(countrydat) +
153|     geom_point(aes(x = date, y = yn, colour = "blue")) +
154|     geom_line(aes(x = date, y = yn, colour = "blue")) +
155|     geom_ribbon(data = modeldat, aes(x = date, ymin = hwylo, ymax = hwyhi, fill = "pi"), alpha = 0.5) +
156|     geom_point(data = modeldat, aes(x = date, y = hwyn, colour = "hw"), shape = 5) +
157|     geom_line(data = modeldat, aes(x = date, y = hwyn, colour = "hw")) +
158|     geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
159|     geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
160|     geom_point(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
161|     geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
162|     gg_scale_xy +
163|     guides(colour=guide_legend(ncol=1,nrow=4,byrow=TRUE),
164|            fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
165|     scale_fill_manual(values = c("pi" = cols$hwpi),
166|                       labels = c("pi" = labs$hwpi)) +
167|     scale_colour_manual(values = c("base" = cols$baseyn, "blue" = cols$yn, "hw" = cols$hw, "periodic" = cols$periodic),
168|                           labels = c("base" = labs$baseyn, "blue" = labs$yn, "hw" = labs$hw, "periodic" = labs$periodicy),
169|                           guide = guide_legend(override.aes = list(
170|                             shape = c("base" = 1, "blue"=16, "hw" = 5, "periodic"=16)))) +
171|     yntheme()
172|   return(p)
173| }
```

```

174|
175| plot_arima <- function(countrydat, modeldat, cols, labs){
176|   p <- ggplot(countrydat, binwidth = 0) +
177|     geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
178|     geom_ribbon(data = modeldat, aes(x = date, ymin = arimalo, ymax = arimahi, fill = "pi"),
179|                 alpha = 0.5) +
180|     geom_point(data = modeldat, aes(x = date, y = arimaxn, colour = "arima")) +
181|     geom_line(data = modeldat, aes(x = date, y = arimaxn, colour = "arima")) +
182|     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
183|     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
184|     geom_point(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
185|     geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
186|     gg_scale_xy +
187|     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
188|            fill=guide_legend(ncol=1,nrow=2,byrow=TRUE)) +
189|     scale_fill_manual(values = c("actual" = cols$xn, "pi" = cols$arimapi),
190|                       labels = c("actual" = labs$xn, "pi" = labs$arimapi)) +
191|     scale_colour_manual(values = c("arima" = cols$arima, "base" = cols$basexn, "periodic" = cols$periodic),
192|                          labels = c("arima" = labs$arima, "base" = labs$basexn, "periodic" = labs$periodic)) +
193|     xntheme()
194|   return(p)
195}
196|
197| plot_arimay <- function(countrydat, modeldat, cols, labs){
198|   p <- ggplot(countrydat) +
199|     geom_point(aes(x = date, y = yn, colour = "blue")) +
200|     geom_line(aes(x = date, y = yn, colour = "blue")) +
201|     geom_ribbon(data = modeldat, aes(x = date, ymin = arimaylo, ymax = arimayhi, fill = "pi"),
202|                 alpha = 0.5) +
203|     geom_point(data = modeldat, aes(x = date, y = arimayn, colour = "arima"), shape = 2) +
204|     geom_line(data = modeldat, aes(x = date, y = arimayn, colour = "arima")) +
205|     geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
206|     geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
207|     geom_point(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
208|     geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
209|     gg_scale_xy +
210|     scale_fill_manual(values = c("pi" = cols$arimapi),
211|                       labels = c("pi" = labs$arimapi)) +
212|     scale_colour_manual(values = c("arima" = cols$arima, "base" = cols$baseyn, "blue" = cols$yn,
213|                                   "periodic" = cols$periodic),
214|                          labels = c("arima" = labs$arimay, "base" = labs$baseyn, "blue" = labs$yn,
215|                                    "periodic" = labs$periodicy),
216|                          guide = guide_legend	override.aes = list(
217|                            shape = c("arima" = 2, "base" = 1, "blue"=16, "periodic" = 16))) +
218|     yntheme()
219|   return(p)
220}
221|
222| plot_hwarima <- function(countrydat, modeldat, cols, labs){
223|   p <- ggplot(countrydat, binwidth = 0) +
224|     geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
225|     geom_point(data = modeldat, aes(x = date, y = arimaxn, colour = "arima")) +
226|     geom_line(data = modeldat, aes(x = date, y = arimaxn, colour = "arima")) +
227|     geom_point(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
228|     geom_line(data = modeldat, aes(x = date, y = hwxn, colour = "hw")) +
229|     gg_scale_xy +
230|     guides(colour=guide_legend(ncol=1,nrow=2,byrow=TRUE),
231|            fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
232|     scale_fill_manual(values = c("actual" = cols$xn),
233|                       labels = c("actual" = labs$xn)) +
234|     scale_colour_manual(values = c("arima" = cols$arima, "hw" = cols$hw),
235|                          labels = c("arima" = labs$arima, "hw" = labs$hw)) +
236|     xntheme()
237|   return(p)
238}
239|
240| plot_nn <- function(countrydat, modeldat, cols, labs){
241|   p <- ggplot(countrydat, binwidth = 0) +
242|     geom_bar(aes(x = date, y = xn, fill = "actual"), stat = "identity") +
243|     geom_point(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
244|     geom_line(data = modeldat, aes(x = date, y = basexn, colour = "base")) +
245|     geom_point(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +

```

```

242|     geom_line(data = modeldat, aes(x = date, y = periodic, colour = "periodic")) +
243|     geom_point(data = modeldat, aes(x = date, y = nnxn, colour = "nn")) +
244|     geom_line(data = modeldat, aes(x = date, y = nnxn, colour = "nn")) +
245|     gg_scale_xy +
246|     guides(colour=guide_legend(ncol=1,nrow=3,byrow=TRUE),
247|             fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
248|     scale_fill_manual(values = c("actual" = cols$xn),
249|                        labels = c("actual" = labs$xn)) +
250|     scale_colour_manual(values = c("base" = cols$basexn, "nn" = cols$nn, "periodic" = cols$periodic),
251|                          labels = c("base" = labs$basexn, "nn" = labs$nn, "periodic" = labs$periodic)) +
252|     xntheme()
253|   return(p)
254| }
255|
256 plot_nny <- function(countrydat, modeldat, cols, labs){
257   p <- ggplot(countrydat) +
258     geom_point(aes(x = date, y = yn, colour = "blue")) +
259     geom_line(aes(x = date, y = yn, colour = "blue")) +
260     geom_point(data = modeldat, aes(x = date, y = baseyn, colour = "base"), shape = 1) +
261     geom_line(data = modeldat, aes(x = date, y = baseyn, colour = "base")) +
262     geom_point(data = modeldat, aes(x = date, y = nnyn, colour = "nn"), shape = 0) +
263     geom_line(data = modeldat, aes(x = date, y = nnyn, colour = "nn")) +
264     geom_point(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
265     geom_line(data = modeldat, aes(x = date, y = periodicy, colour = "periodic")) +
266     gg_scale_xy +
267     scale_colour_manual(values = c("base" = cols$baseyn, "blue" = cols$yn, "nn" = cols$nn, "periodic" = cols$periodic),
268                           labels = c("base" = labs$baseyn, "blue" = labs$yn, "nn" = labs$nn, "periodic" = labs$periodic)),
269     guide = guide_legend(override.aes = list(
270       shape = c("base" = 1, "blue"=16, "nn" = 0, "periodic" = 16))) +
271   yntheme()
272   return(p)
273 }
274|
275 plot_multixn <- function(countrydat, modeldat, cols, labs){
276   p <- ggplot(countrydat, binwidth = 0) +
277     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
278     geom_point(data = modeldat, aes(x = date, y = multixn, colour = cols$multixn)) +
279     geom_line(data = modeldat, aes(x = date, y = multixn, colour = cols$multixn)) +
280     gg_scale_xy +
281     scale_fill_manual( name = "leg",values = cols$xn, labels = labs$xn) +
282     scale_colour_manual(name = "leg",values = cols$multixn, labels = labs$multixn) +
283     xntheme()
284   return(p)
285 }
286|
287 plot_multiyn <- function(countrydat, modeldat, cols, labs){
288   p <- ggplot(countrydat) +
289     geom_point(aes(x = date, y = yn, colour = "blue")) +
290     geom_line(aes(x = date, y = yn, colour = "blue")) +
291     geom_point(data = modeldat, aes(x = date, y = multiyn, colour = "base"), shape = 1) +
292     geom_line(data = modeldat, aes(x = date, y = multiyn, colour = "base")) +
293     gg_scale_xy +
294     scale_colour_manual(name = "leg",values = c("blue" = cols$yn, "base" = cols$multiyn),
295                           labels = c("blue" = labs$yn, "base" = labs$multiyn),
296                           guide = guide_legend(override.aes = list(
297                             shape = c("blue"=1, "base" = 16)))) +
298   yntheme()
299   return(p)
300 }
301|
302 plot_multipernx <- function(countrydat, modeldat, cols, labs){
303   p <- ggplot(countrydat, binwidth = 0) +
304     geom_bar(aes(x = date, y = xn, fill = cols$xn), stat = "identity") +
305     geom_point(data = modeldat, aes(x = date, y = multipxn, colour = "multi")) +
306     geom_line(data = modeldat, aes(x = date, y = multipxn, colour = "multi")) +
307     geom_line(data = countrydat, aes(x = date, y = mavgx3, colour = "x3")) +
308     gg_scale_xy +
309     guides(colour=guide_legend(ncol=1,nrow=2,byrow=TRUE),
310            fill=guide_legend(ncol=1,nrow=1,byrow=TRUE)) +
311     scale_fill_manual(values = cols$xn, labels = labs$xn) +

```

```

312     scale.colour.manual(values = c("multi" = cols$multip , "x3" = cols$x3) ,
313                           labels = c("multi" = labs$multipxn , "x3" = labs$x3)) +
314     guides(colour = guide.legend	override.aes = list(shape = c("multi" = 16, "x3" = NA))) +
315     xntheme()
316   return(p)
317 }
318
319 plot_multiperyn <- function(countrydat, modeldat, cols, labs){
320   p <- ggplot(countrydat) +
321     geom_point(aes(x = date, y = yn, colour = "blue"), shape = 16) +
322     geom_line(aes(x = date, y = yn, colour = "blue")) +
323     geom_point(data = modeldat, aes(x = date, y = multipyn, colour = "mult"), shape = 1) +
324     geom_line(data = modeldat, aes(x = date, y = multipyn, colour = "mult")) +
325     gg_scale_xy +
326     scale.colour.manual(name = "leg", values = c("blue" = cols$yn, "mult" = cols$multip) ,
327                           labels = c("blue" = labs$yn, "mult" = labs$multipyn),
328                           guide = guide.legend	override.aes = list(
329                             shape = c("blue"=16, "mult" = 1)))) +
330     yntheme()
331   return(p)
332 }
333
334 plot_worldtotal <- function(dat){
335   p <- ggplot(dat, binwidth = 0) +
336     geom_bar(aes(x = date, y = new_cases), fill = wes_palettes$Zissou1[1], stat = "identity") +
337     scale_x_date(date_breaks = "1 month", date_labels = "%b", expand = c(0,0))+ 
338     scale_y_continuous(expand = c(0,0)) +
339     ggtitle(wt_title) + xntheme()
340   return(p)
341 }
342
343 modnorm <- function(x,modx) return(floor(sum(abs(x-modx))/length(x)))
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   return(p)
368 }
369
370 yntheme <- function(){
371   p <- theme(axis.text.x      = element_text(vjust = 0.5) ,
372               axis.title       = element_blank() ,
373               axis.line        = element_line() ,
374               panel.background = element_rect(fill = "grey"),
375               panel.grid        = element_line(colour = "darkgrey"),
376               panel.grid.major.x = element_blank() ,
377               panel.grid.minor.x = element_blank() ,
378               legend.title     = element_blank() ,
379               legend.margin    = margin(4,0,0,4,"pt") ,
380               legend.background = element_blank() ,
381               legend.key       = element_blank(),
382               legend.box.background = element_rect(linetype="solid", colour ="darkgrey", size = 0
383                                         .1, fill = "white"),
383               legend.spacing    = unit(0, "cm"),
383               legend.key.size  = unit(0.8,"line"),

```

```

384         legend.text      = element_text(size = 6),
385         legend.direction = "vertical",
386         legend.box       = "vertical",
387         legend.box.just = 'left',
388         legend.position  = "top")
389     return(p)
390   }
391 
392 xntoyn <- function(xn) return(cumsum(xn))
393 
394 gg_scale_xy <- list(
395   scale_x_date(date_breaks = "1 week", date_labels = "%d-%b", expand = c(0,0)),
396   scale_y_continuous(expand = c(0,0)))
397 
398 basicmodx <- function(x, pars, len = 0){
399   q <- floor(pars[1])
400   a <- pars[2]
401   b <- pars[3]
402   modx <- x[1:q]
403   for(i in (q+1):(length(x)+len)){
404     modx[i] <- (1-b)*modx[i-1] + a*modx[i-q]
405   }
406   return(modx)
407 }
408 
409 norm <- function(par, x) return(modnorm(x,basicmodx(x, par)))
410 
411 normy <- function(par, x, y) return(modnorm(y,xntoyn(basicmodx(x, par))))
412 
413 normalize <- function(x) return((x-min(x))/(max(x)-min(x)))
414 
415 basef <- function(lambda,par) {
416   return(lambda^(par[1]+1)-(1-par[3])*lambda^(par[1])-par[2])
417 }
418 
419 basefprime <- function(lambda,par) {
420   return((par[1]+1)*lambda^(par[1])-(1-par[3])*par[1]*lambda^(par[1]-1))
421 }
422 
423 normC <- function(par,x,r){
424   return(modnorm(x, par*r^(0:(length(x)!is.na(x))-1)))
425 }
426 
427 movingavg <- function(x){
428   mavgx <- (x[1]+x[2])/2
429   for(i in 2:(length(x)-1)){
430     mavgx[i] <- sum(x[(i-1):(i+1)])/3
431   }
432   mavgx[length(x)] <- (x[length(x)-1]+x[length(x)])/2
433   return(mavgx)
434 }
435 
436 normper <- function(par, q, x) return(modnorm(x,modxper(par,q,x)))
437 
438 modxper <- function(par, q, x, len = 0){
439   #a,b,c1,c2,p1,p2,n1,n2
440   an <- par[1]*(1+par[3]*sin(2*pi*(1:(length(x)+len) - par[7])/par[5]))
441   bn <- par[2]*(1+par[4]*sin(2*pi*(1:(length(x)+len) - par[8])/par[6]))
442   modx <- x[1:q]
443   for(i in (q+1):(length(x)+len)){
444     modx[i] <- (bn[i]*(1-bn[i-1]))*modx[i-1]/bn[i-1] +
445     (an[i-q]*bn[i])*modx[i-q]/bn[i-q]
446   }
447   return(modx)
448 }
449 
450 covidPlots <- function(country, dateBounds, data){
451   plots <- list()
452   countryrows <- grep(country, data$location)
453   countrydat <- data.frame(date = data$date[countryrows],
454                             xn = data$new_cases[countryrows],
455                             yn = data$total_cases[countryrows])
456   countrydatfull <- countrydat[countrydat$date <= dateBounds[2],]
457 }
```

```

458 | #if (nrow(countrydat[countrydat$date < dateBounds[1],]) == 0)
459 | #  prevcases <- 0
460 | #else
461 | #  prevcases <- sum(countrydat$xn[countrydat$date < dateBounds[1]])
462 |
463 | prevcases <- countrydat$yn[countrydat$date == as.Date(dateBounds[1])-1]
464 | #Specific dates
465 | countrydat <- countrydat[countrydat$date >= dateBounds[1] & countrydat$date <= dateBounds[2],]
466 | #countrydat$xn[countrydat$xn < 0] <- 0
467 | latest_date <- countrydat$date[nrow(countrydat)]
468 |
469 | cols <- list(
470 |   xn      = wes_palettes$Zissou1[1],
471 |   yn      = wes_palettes$Darjeeling2[2],
472 |   basexn  = wes_palettes$Darjeeling1[1],
473 |   baseyn  = wes_palettes$Darjeeling1[1],
474 |   x3      = wes_palettes$FantasticFox1[2],
475 |   Crn     = wes_palettes$FantasticFox1[2],
476 |   arima    = wes_palettes$Darjeeling1[4],
477 |   arimapi  = wes_palettes$Darjeeling1[3],
478 |   hw      = wes_palettes$Moonrise1[2],
479 |   hwpi    = wes_palettes$Moonrise1[1],
480 |   periodic = "magenta4", #wes_palettes$IsleofDogs1[1], #
481 |   nn      = wes_palettes$FantasticFox1[4]
482 | )
483 |
484 | labs <- list(
485 |   xn = list(bquote(..(country)*,"~x[n]*"=new cases/day, actual till"~.(format.Date(latest_date
486 | , "%d.%m.%Y"))),
487 |   yn = list(bquote(..(country)*,"~y[n]*"=cumulative cases, actual till"~.(format.Date(latest_
488 | date, "%d.%m.%Y"))))
489 | )
490 |
491 | plots[["xn"]] <- plot_xn(countrydatfull, cols, labs)
492 |
493 | plots[["yn"]] <- plot_yn(countrydatfull, cols, labs)
494 |
495 | #Basic model: need a,b,q,r using ||x-x*|| and ||y-y*||
496 | ##q - Any infected person becomes ill and infectious on the q-th day after infection.
497 | ##a - During each day, each ill person at large infects on average a other persons.
498 | ##b - During each day, a fraction b of ill people at large gets isolated
499 |
500 | forecastlen <- 14
501 |
502 | aseq   <- seq(from = 0.1, to = 2.5, length.out = 80)
503 | bseq   <- seq(from = 0.1, to = 0.9, length.out = 80)
504 | qseq   <- 6:8
505 | normdat <- expand.grid(q = qseq, a = aseq, b = bseq)
506 | abnorm <- apply(normdat, 1, function(x) norm(x, countrydat$xn))
507 |
508 | normdat$abnorm <- abnorm
509 |
510 | newnormdat <- normdat %>%
511 |   top_n(abnorm, n = -0.07*nrow(.))
512 |
513 | col_grad <- wes_palette("Zissou1", 20, type = "continuous")
514 |
515 | tileoptim <- normdat[which.min(normdat$abnorm), 1:3]
516 | #tileoptim <- newnormdat[which.min(newnormdat$abnorm), 1:3]
517 | optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
518 | plots[["combnorm"]る <- ggplot(newnormdat, aes(x = a, y = b, z = abnorm)) +
519 |   geom_contour_filled() + labs(fill = "||x-x*||") +
520 |   scale_fill_brewer(palette = "Spectral")
521 |
522 | q <- optimpars[1]
523 | a <- optimpars[2]
524 | b <- optimpars[3]
525 |
526 | basexn <- basicmodx(countrydat$xn, optimpars, len = forecastlen)
527 |
528 | modeldat <- data.frame(date = c(countrydat$date, latest_date + 1:forecastlen),
529 |                           basexn = basexn, baseyn = xntoyn(basexn) + prevcases)
530 |
531 | #Newtons method, r_1 = r_0 - f(r_0)/f'(r_0)
```

```

530 base_r_zero <- (a/b)^(1/(2*q))
531 base_r_one <- base_r_zero -basef(base_r_zero, optimpars)/basefprime(base_r_zero, optimpars)
532 base_r_one <- round(base_r_one,3)
533
534 roptimpars <- round(optimpars,3)
535 labs$basexn <- list(bquote("basic model, " ~x[n]*"=new cases/day; a=""*(roptimpars[2])*", b=""*(roptimpars[3])*", q=""*(roptimpars[1])*"; r=""*(base_r_one)*"; ||x*x||=""*(norm(optimpars, countrydat$xn))) )
536
537 ynnorm <- modnorm(countrydat$yn, modeldat$baseyn[1:length(countrydat$yn)])
538 labs$baseyn <- list(bquote("basic model, " ~y[n]*"=cumulative cases; a=""*(roptimpars[2])*", b=""*(roptimpars[3])*", q=""*(roptimpars[1])*", ||y*y||=""*(ynnorm)))
539
540 plots[["basexn"]] <- plot_basexn(countrydat, modeldat, cols, labs)
541
542 plots[["baseyn"]] <- plot_baseyn(countrydat, modeldat, cols, labs)
543
544 optimC <- optim(par = countrydat$xn[1], normC, method = "Brent",
545                   lower = 1, upper = 2*max(countrydat$xn[!is.na(countrydat$xn)]),
546                   x = basexn[1:nrow(countrydat)], r = base_r_one)$par
547
548 modeldat$Crn <- optimC*base_r_one^(1:nrow(modeldat))
549
550 labs$Crn <- list(bquote(Cr^n*, r=""*(base_r_one)*", C=""*(floor(optimC))))
551
552 plots[["Crn"]] <- plot_crn(countrydat, modeldat, cols, labs)
553
554 mavgx1 <- movingavg(countrydat$xn[!is.na(countrydat$xn)])
555
556 countrydat$mavgx3 <- movingavg(mavgx1)
557
558 x3norm <- modnorm(countrydat$xn, countrydat$mavgx3)
559 labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||=""*(x3norm)))
560
561 plots[["mavgx3"]] <- plot_mavgx3(countrydat, modeldat, cols, labs)
562
563 #parameters of the form (ci, pi, ni)
564 ##an = a(1 + c1 sin(2pi/p1 (n - n1)))
565 aseqper <- a*seq(from = 0.8, to = 1.2, length.out = 5)
566 bseqper <- b*seq(from = 0.8, to = 1.2, length.out = 5)
567 c_1seq <- c_2seq <- seq(0.04, 0.2, length.out = 10)
568 n_1seq <- n_2seq <- c(1,7)
569 p_1seq <- p_2seq <- 6:7
570 normdattp <- expand.grid(a = aseqper, b = bseqper,
571                           c1 = c_1seq, c2 = c_2seq,
572                           p1 = p_1seq, p2 = p_2seq,
573                           n1 = n_1seq, n2 = n_2seq)
574
575 pernorm <- apply(normdattp, 1, function(x) normper(x, q = q, countrydat$xn))
576 normdattp$pernorm <- pernorm
577
578 peroptim <- normdattp[which.min(pernorm),1:8]
579 optpernorm <- normdattp[which.min(pernorm),9]
580
581 modeldat$periodic <- modxper(as.numeric(peroptim),countrydat$xn, q = q, forecastlen)
582 modeldat$periodicy <- xntoyn(modeldat$periodic) + prevcases
583
584 optpernormy <- modnorm(countrydat$yn, modeldat$periodicy[1:nrow(countrydat)])
585
586 peroptim <- round(as.numeric(peroptim),3)
587
588 perparamdat <- data.frame(
589   x = modeldat$date,
590   an = peroptim[1]*(1+peroptim[3]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[7])/peroptim[5])),
591   bn = peroptim[2]*(1+peroptim[4]*sin(2*pi*(1:(nrow(modeldat)) - peroptim[8])/peroptim[6]))
592 )
593
594 plots[["perparam"]] <- ggplot(perparamdat) +
595   geom_line(aes(x=x,y=an, col="a_n")) +
596   geom_line(aes(x=x,y=bn, col="b_n")) +
597   geom_hline(aes(yintercept = peroptim[1], col = "a"), linetype="dashed") +
598   geom_hline(aes(yintercept = peroptim[2], col = "b"), linetype="dashed") +
599   xlab("date") + ylab("") +
600   scale_color_manual(values = wes_palettes$Rushmore1[c(3,3,5,5)]) +

```

```

601 guides(colour = guide_legend(override.aes = list(linetype =
602   c("a"="dashed", "a_n"="solid", "b"="dashed", "b_n"="solid")))) +
603   xntheme() + theme(legend.position = "right")
604
605 #a,b,c1,c2,p1,p2,n1,n2
606 labs$periodic <- list(bquote(~x[n]*"=new cases/day;" ~
607   ~a*="*(.perooptim[1])*,"~ b*="*(.perooptim[2])*,"~
608   ~q*="*(.q)*;"~ ||x*-x||="*(.optpernorm)*;"~
609   ~c[1]*="*(.perooptim[3])*,"~ p[1]*="*(.perooptim[5])*,"~
610   ~n[1]*="*(.perooptim[7])*,"~ c[2]*="*(.perooptim[4])*,"~
611   ~p[2]*="*(.perooptim[6])*,"~ n[2]*="*(.perooptim[8])))
612
613 labs$periodicity <- list(bquote(~y[n]*"=new cases/day;" ~
614   ~a*="*(.perooptim[1])*,"~ b*="*(.perooptim[2])*,"~
615   ~q*="*(.q)*;"~ ||x*-x||="*(.optpernorm)*;"~
616   ~c[1]*="*(.perooptim[3])*,"~ p[1]*="*(.perooptim[5])*,"~
617   ~n[1]*="*(.perooptim[7])*,"~ c[2]*="*(.perooptim[4])*,"~
618   ~p[2]*="*(.perooptim[6])*,"~ n[2]*="*(.perooptim[8])))
619
620 plots[["periodic"]] <- plot_periodic(countrydat, modeldat, cols, labs)
621
622 plots[["periodicity"]] <- plot_periodicity(countrydat, modeldat, cols, labs)
623
624 #Statistical methods using timeseries forecasting
625
626 dat_ts <- ts(data = countrydat$xn, frequency = q)
627
628 g1 <- autoplot(dat_ts) + theme(axis.title = element_blank())
629 g2 <- ggAcf(dat_ts) + ggtitle(" ")
630 g3 <- ggPacf(dat_ts) + ggtitle(" ")
631
632 plots[["tsdisplay"]] <- grid.arrange(grobs = list(g1,g2,g3),
633   layout_matrix = rbind(c(1, 1), c(2, 3)))
634
635 plots[["residuals"]] <- gghistogram(dat_ts, add.normal = TRUE)
636
637 plots[["tsdecompose"]] <- autoplot(decompose(dat_ts))
638
639 if(any(countrydat$xn <= 0)){
640   plots[["hw"]]<- ggplot(data.frame(x = 0,y = 0)) +
641     geom_label(x = 0, y = 0, label = "Error: Country data has nonpositive values",
642       color = "red", size = 5 , fontface = "bold" ) +
643     xlim(-1,1) + ylim(-1,1) +
644     theme(axis.line = element_blank(),
645       axis.text = element_blank(),
646       axis.ticks = element_blank(),
647       panel.grid = element_blank())
648 } else{
649   hwmethod <- "additive"
650   #lambda=0 ensures values stay positive
651   hwfcst <- forecast::hw(dat_ts, h = forecastlen, seasonal = hwmethod, lambda = 0)
652
653   hwfcst$fitted[1:q] <- countrydat$xn[1:q]
654   modeldat$hwxn <- c(hwfcst$fitted, hwfcst$mean)
655   modeldat$hwlo <- c(hwfcst$fitted, hwfcst$lower[,2])
656   modeldat$hwhi <- c(hwfcst$fitted, hwfcst$upper[,2])
657
658   modeldat$hwyn <- xtoyn(modeldat$hwxn)+prevcases
659   modeldat$hwylo <- xtoyn(modeldat$hwlo)+prevcases
660   modeldat$hwyhi <- xtoyn(modeldat$hwhi)+prevcases
661
662   hwnorm <- modnorm(countrydat$xn, hwfcst$fitted)
663
664   labs$hw <- paste0("HoltWinters algorithm, ||x*-x||=", modnorm(countrydat$xn, hwfcst$fitted))
665   labs$hwy <- paste0("HoltWinters algorithm, ||y*-y||=", modnorm(countrydat$yn, modeldat$hwyn[1:nrow(countrydat)]))
666   labs$hwpi <- "HW 95% Prediction Interval"
667
668   plots[["hw"]]<- plot_hw(countrydat, modeldat, cols, labs)
669   plots[["hwy"]]<- plot_hwy(countrydat, modeldat, cols, labs)
670 }
671
672 auto.fit <- auto.arima(dat_ts, lambda = 0) #keep values positive

```

```

673| getArmaModel <- function(arma, pdq = c(1,6,2), PDQ = c(3,7,4), s=5){
674|   return(paste0("ARIMA(", paste0(arma[pdq], collapse = ","), ")",
675|                 paste0(arma[PDQ], collapse = ","), ", ", arma[s], "]"))
676| }
677|
678|
679| arima.fcst <- forecast(auto.fit, level = c(80, 95), h = forecastlen)
680| arima.fcst$fitted[1:q] <- countrydat$xn[1:q]
681|
682| arimanorm <- modnorm(countrydat$xn, arima.fcst$fitted)
683|
684| arimalabs <- getArmaModel(auto.fit$arma)
685| labs$arima <- paste0(arimalabs, " ||x-x||=", arimanorm)
686| labs$arimapi <- "ARIMA 95% Prediction Interval"
687|
688| modeldat$arimaxn <- c(auto.fit$fitted, arima.fcst$mean)
689| modeldat$arimalo <- c(auto.fit$fitted, arima.fcst$lower[,2])
690| modeldat$arimahi <- c(auto.fit$fitted, arima.fcst$upper[,2])
691|
692| modeldat$arimaxn[1:q] <- countrydat$xn[1:q]
693| modeldat$arimalo[1:q] <- countrydat$xn[1:q]
694| modeldat$arimahi[1:q] <- countrydat$xn[1:q]
695|
696| modeldat$arimayn <- xntoyn(modeldat$arimaxn) + prevcases
697| modeldat$arimaylo <- xntoyn(modeldat$arimalo) + prevcases
698| modeldat$arimayhi <- xntoyn(modeldat$arimahi) + prevcases
699|
700| labs$arimay <- paste0(arimalabs, " ||y-y||=", modnorm(countrydat$yn, modeldat$arimayn[1:nrow(
701|   countrydat)]))
702| plots[["arima"]] <- plot_arima(countrydat, modeldat, cols, labs)
703|
704| plots[["arimay"]] <- plot_arimay(countrydat, modeldat, cols, labs)
705|
706| plots[["hwarima"]] <- plot_hwarima(countrydat, modeldat, cols, labs)
707|
708| nHidden <- max(1, floor(0.5*(1+auto.fit$arma[1]+auto.fit$arma[3])))
709| #Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.
710| nnfit <- nnetar(dat_ts, p = auto.fit$arma[1], P = auto.fit$arma[3], size = nHidden, lambda =
711|   0, repeats = 20, maxit = 50)
712| nn.fcst <- forecast(nnfit, h = forecastlen)
713|
714| nn.fcst$mean[nn.fcst$mean < 0] <- 0
715| nn.fcst$fitted[1:q] <- countrydat$xn[1:q]
716|
717| modeldat$nnxn <- c(nn.fcst$fitted, nn.fcst$mean)
718| modeldat$nnyn <- xntoyn(modeldat$nnxn) + prevcases
719|
720| labs$nn <- paste0(nnfit$method, " ||x-x||=", modnorm(countrydat$xn, nn.fcst$fitted))
721| labs$nny <- paste0(nnfit$method, " ||y-y||=", modnorm(countrydat$yn, modeldat$nnyn[1:nrow(
722|   countrydat)]))
723|
724| plots[["nn"]] <- plot_nn(countrydat, modeldat, cols, labs)
725|
726| plots[["nny"]] <- plot_nny(countrydat, modeldat, cols, labs)
727|
728| return(plots)
729}
730|
731| grigorDates <- c("2020-04-26", "2020-06-09")
732| datebounds <- list(
733|   "Italy" = c("2021-01-02", "2021-02-16"),
734|   "United States" = c("2021-01-06", "2021-02-16"),
735|   "Ireland" = c("2021-01-12", "2021-02-16"),
736|   "Germany" = c("2021-01-06", "2021-02-16"),
737|   "#Netherlands" = c("2021-01-06", "2021-02-16"),
738|   "#Spain" = c("2021-01-06", "2021-02-16"),
739|   "#UK" = c("2021-01-06", "2021-02-16"))
740|
741| owiddat <- owiddat[!is.na(owiddat$new_cases),]
742| totaldat <- owiddat[owiddat$location == "World",]
743| latest_date <- totaldat$date[nrow(totaldat)]
744| wt_title <- sprintf('Global Total =%s as at %s',

```

```

744         format(sum(totaldat$new_cases), big.mark=",", scientific=FALSE),
745         format.Date(latest_date, "%B %d, %Y"))
746 
747 plotslist[["WorldTotal"]][["xn"]] <- plot_worldtotal(totaldat)
748 
749 for(country in names(datebounds)){
750   plotslist[[country]] <- covidPlots(country, datebounds[[country]], owiddat)
751 }
752 
753 multidates <- list(
754   "Italy"      = list(c("2020-12-16", "2021-01-01"),
755     c("2021-01-02", "2021-01-30")),
756   "United States" = list(c("2020-11-16", "2021-01-06"),
757     c("2021-01-07", "2021-01-30")),
758   "Ireland"    = list(c("2020-12-16", "2021-01-07"),
759     c("2021-01-08", "2021-01-30")))
760 )
761 
762 multiphasePlots <- function(country, dates, data){
763   plots <- list()
764   crows <- grep(country, data$location)
765   countrydat <- data.frame(date = data$date[crows],
766     xn = data$new_cases[crows], yn = data$total_cases[crows])
767   if(nrow(countrydat$countrydat$date < dates[[1]][1],) == 0)
768     beforeecumcases <- 0
769   else
770     beforeecumcases <- sum(countrydat$xn[countrydat$date < dates[[1]][1]])
771 
772   countrydat <- countrydat[countrydat$date >= dates[[1]][1],]
773   countrydat <- countrydat[countrydat$date <= dates[[length(dates)]][2],]
774   latest_date <- countrydat$date[nrow(countrydat)]
775 
776 forecastlen <- 5
777 
778 multimodx <- function(x, multix, pars, oldp = rep(1,10), start=FALSE, len = 0){
779   q <- floor(pars[1])
780   a <- pars[2]
781   b <- pars[3]
782   fitstd <- length(multix)+1
783 
784   if(start){
785     multix[fitstd:(fitstd+q-1)] <- x[1:q]
786     for(i in (fitstd+q):(fitstd+length(x)+len-1)){
787       multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
788     }
789   } else {
790     for(i in (fitstd):(fitstd+length(x)+len-1)){
791       multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
792     }
793     #multix[fitstd] <- b/oldp[[3]]*((1-oldp[[3]])*multix[fitstd-1] + oldp[[2]]*multix[fitstd-q])
794     #for(i in (fitstd+1):(fitstd+q-1)){
795     #  multix[i] <- (1-b)*multix[i-1] + b/oldp[[3]]*oldp[[2]]*multix[i-q]
796     #}
797     #for(i in (fitstd+q):(fitstd+length(x)+len-1)){
798     #  multix[i] <- (1-b)*multix[i-1] + a*multix[i-q]
799     #}
800   }
801   return(multix)
802 }
803 
804 multimodxper <- function(par, q=7, x, multix, oldp = rep(1,10), start=FALSE, len = 0){
805   #a,b,c1,c2,p1,p2,n1,n2
806   #first day of this phase
807   fitstd <- length(multix)+1
808   an <- par[1]*(1+par[3]*sin(2*pi*(1:(fitstd+length(x)+len-1) - par[7])/par[5]))
809   bn <- par[2]*(1+par[4]*sin(2*pi*(1:(fitstd+length(x)+len-1) - par[8])/par[6]))
810 
811   if(start){
812     multix[fitstd:(fitstd+q-1)] <- x[1:q]
813     for(i in (fitstd+q):(fitstd+length(x)+len-1)){
814       multix[i] <- (bn[i]*(1-bn[i-1]))*multix[i-1]/bn[i-1] +
815         (an[i-q]*bn[i])*multix[i-q]/bn[i-q]
816     }

```

```

817 } else {
818   for(i in fitstd:(fitstd+length(x)+len-1)){
819     multix[i] <- (bn[i]*(1-bn[i-1]))*multix[i-1]/bn[i-1] +
820       (an[i-q]*bn[i])*multix[i-q]/bn[i-q]
821   }
822 }
823 return(multix)
824 }
825
826 #Specific dates
827 multimodel <- c()
828 multimodelp <- c()
829 phasepars <- list()
830 for(i in 1:length(dates)){
831   phase <- dates[[i]]
832   phasedat <- countrydat[countrydat$date >= phase[1] & countrydat$date <= phase[2],]
833
834   #get basic model for each phase first in order to get
835   # starting a and b to guess for periodic an and bn
836
837   aseq <- seq(from = 0.1, to = 2.5, length.out = 50)
838   bseq <- seq(from = 0.1, to = 0.9, length.out = 50)
839   qseq <- 6:8
840   normmdat <- expand.grid(q = qseq, a = aseq, b = bseq)
841
842   if(i == 1)
843     abnorm <- apply(normmdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, start=TRUE), phasedat$xn))
844   else
845     abnorm <- apply(normmdat, 1, function(x) modnorm(multimodx(phasedat$xn, multimodel, x, oldp = phasepars[[i-1]])[length(multimodel) + 1:nrow(phasedat)], phasedat$xn))
846
847   normalize <- function(x){
848     return((x-min(x))/(max(x)-min(x)))
849   }
850   normmdat$abnorm <- abnorm
851   #newnormmdat <- normmdat %>% top_n(abnorm,n = -0.1*nrow(.))
852
853   #if(i == 1)
854   # abnormy <- apply(newnormmdat, 1, function(x) modnorm(beforeecumcases + xntoyn(multimodx(phasedat$xn, multimodel, x[1:3], start=TRUE)), phasedat$yn))
855   #else
856   # abnormy <- apply(newnormmdat, 1, function(x) modnorm(beforeecumcases + xntoyn(multimodx(phasedat$xn, multimodel, x[1:3], oldp = phasepars[[i-1]])[length(multimodel)+1:nrow(phasedat)], phasedat$yn)))
857
858   #newnormmdat$abnormy <- normalize(abnormy)
859   #newnormmdat$combnorm <- newnormmdat$abnorm + newnormmdat$abnormy
860   tileoptim <- normmdat[which.min(normmdat$abnorm),1:3]
861   #tileoptim <- newnormmdat[which.min(newnormmdat$combnorm),1:3]
862   optimpars <- c(tileoptim$q, tileoptim$a, tileoptim$b)
863
864   q <- optimpars[1]
865   a <- optimpars[2]
866   b <- optimpars[3]
867
868   phasepars[[i]] <- round(optimpars,3)
869
870   if(i == 1 & i != length(dates))
871     multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE)
872   if(i == 1 & i == length(dates))
873     multimodel <- multimodx(phasedat$xn, multimodel, optimpars, start=TRUE, len=forecastlen)
874   if(i > 1 & i == length(dates))
875     multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]], len=forecastlen)
876   if(length(dates) > 2 & i %in% 2:(length(dates)-1))
877     multimodel <- multimodx(phasedat$xn, multimodel, optimpars, oldp = phasepars[[i-1]])
878
879   aseqper <- a*seq(from = 0.7, to = 1.3, length.out = 10)
880   bseqper <- b*seq(from = 0.7, to = 1.3, length.out = 10)
881   c_1seq <- c_2seq <- seq(0.04, 0.2, length.out = 10)
882   n_1seq <- n_2seq <- c(1,7)
883   p_1seq <- p_2seq <- 6:7
884   normdatp <- expand.grid(a = aseqper, b = bseqper,

```

```

885|           c1 = c_1seq,   c2 = c_2seq,
886|           p1 = p_1seq,   p2 = p_2seq,
887|           n1 = n_1seq,   n2 = n_2seq)
888|
889| if(i == 1)
890|     pernorm <- apply(normmdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1],
891|                                         phasedat$xn, multimodelp, start=TRUE), phasedat$xn))
892| else
893|     pernorm <- apply(normmdatp, 1, function(par) modnorm(multimodxper(par, q = optimpars[1],
894|                                         phasedat$xn, multimodelp)[length(multimodelp)+1:nrow(phasedat)], phasedat$xn))
895|
896| normmdatp$pernorm <- normalize(pernorm)
897|
898| if(i == 1)
899|     pernormy <- apply(newnormmdatp, 1, function(par) modnorm(beforeecumcases+xntoyn(multimodxper(
900|                                         par, q = optimpars[1], phasedat$xn, multimodelp, start=TRUE)), phasedat$yn))
901| else
902|     pernormy <- apply(newnormmdatp, 1, function(par) modnorm(beforeecumcases+xntoyn(multimodxper(
903|                                         par, q = optimpars[1], phasedat$xn, multimodelp))[length(multimodelp)+1:nrow(phasedat)],
904|                                         phasedat$yn)))
905|
906| newnormmdatp$pernormy <- normalize(pernormy)
907| newnormmdatp$combnorm <- newnormmdatp$pernorm + newnormmdatp$pernormy
908|
909| peroptim <- as.numeric(newnormmdatp[which.min(newnormmdatp$combnorm),1:8])
910|
911| phasepars[[i]] <- c(phasepars[[i]], round(peroptim, 3))
912|
913| if(i == 1 & i != length(dates))
914|     multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE)
915| if(i == 1 & i == length(dates))
916|     multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, start=TRUE, len=
917|         forecastlen)
918| if(i > 1 & i == length(dates))
919|     multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp, len=forecastlen)
920| if(length(dates) > 2 & i %in% 2:(length(dates)-1))
921|     multimodelp <- multimodxper(peroptim, q = q, phasedat$xn, multimodelp)
922|
923| }
924|
925| cols <- list(
926|     xn      = wes_palettes$Zissou1[1],
927|     yn      = wes_palettes$Darjeeling2[2],
928|     multixn = wes_palettes$Darjeeling1[1],
929|     multiyn = wes_palettes$Darjeeling1[1],
930|     multip  = "magenta4",
931|     x3      = wes_palettes$FantasticFox1[2]
932| )
933|
934| labs <- list(
935|     xn = list(bquote(.(country)*,"`x[n]*`=new cases/day, actual till`~.(format.Date(latest_date
936| , "%d.%m.%Y")))),
937|     yn = list(bquote(.(country)*,"`y[n]*`=cumulative cases, actual till`~.(format.Date(latest_
938| date, "%d.%m.%Y"))))
939|
940|     )
941|
942| multimodnormval <- modnorm(multimodel[1:length(countrydat$xn)], countrydat$xn)
943| multimodnormyval <- modnorm(beforeecumcases+xntoyn(multimodel[1:length(countrydat$xn)]),
944|                               countrydat$yn)
945|
946| b.roman <- function(x){ return(paste0(", as.roman(x), ") ) }
947|
948| multilabd <- c()
949| for(i in 1:length(dates)){
950|     multilabd <- c(multilabd, paste(b.roman(i), "from", format.Date(as.Date(as.character(dates[[i
951| ]][1])), "%d.%m")))
952| }
953| multilabd <- paste0(multilabd, collapse = "; ")
954|
955| multilabp <- c()
956| for(i in 1:length(dates)){
957|     multilabp <- c(multilabp, paste0(b.roman(i), " a=", phasepars[[i]][[2]], ", b=", phasepars[[i
958| ]][[3]]))}
```

```

948 }
949 multilabp <- paste0(multilabp, paste0(; q=", phasepars[[i]][[1]]), collapse = "; ")
950
951 labs$multixn <- list(bquote("base"~.(length(dates))~-phase model"~x[n]~"=new cases/day: "*
952           .(multilabd)*
953           ";" ||x*-x||="*.(multimodnormval)*" "*
954           .(multilabp)))
955 labs$multiyn <- list(bquote("base"~.(length(dates))~-phase model"~y[n]~"=cumulative cases: "*
956           .(multilabd)*
957           ";" ||y*-y||="*.(multimodnormyval)*" "*
958           .(multilabp)))
959
960 mavgx1 <- movingavg(countrydat$xn[!is.na(countrydat$xn)])
961
962 countrydat$mavgx3 <- movingavg(mavgx1)
963
964 x3norm <- modnorm(countrydat$xn, countrydat$mavgx3)
965 labs$x3 <- list(bquote("moving average x*(3); ||x*(3)-x*||="*.x3norm)))
966
967 modeldat <- data.frame(date      = c(countrydat$date, as.Date(latest_date) + 1:forecastlen),
968                           multixn = multimodel,
969                           multiyn = beforecumcases + xntoyn(multimodel),
970                           multipxn = multimodelp,
971                           multipyn = beforecumcases + xntoyn(multimodelp))
972
973 plots[["xn"]] <- plot_multixn(countrydat, modeldat, cols, labs)
974
975 plots[["yn"]] <- plot_multiyn(countrydat, modeldat, cols, labs)
976
977 multilabp <- c()
978 for(i in 1:length(dates)){
979   multilabp <- c(multilabp, paste0(b.roman(i), " a=", phasepars[[i]][[2]], ", b=", phasepars
980   [[i]][[3]]))
981 }
982 multilabp <- paste0(multilabp, paste0(; q=", phasepars[[i]][[1]]), collapse = "; ")
983
984 labs$multipxn <- list(bquote("periodic"~.(length(dates))~-phase model"~x[n]~"=new cases/day: "*
985           .(multilabd)*
986           ";" ||x*-x||="*.(multimodnormval)*" "*
987           .(multilabp)))
988 labs$multipyn <- list(bquote("periodic"~.(length(dates))~-phase model"~y[n]~"=cumulative cases
989           : "*
990           .(multilabd)*
991           ";" ||y*-y||="*.(multimodnormyval)*" "*
992           .(multilabp)))
993
994 plots[["perxn"]] <- plot_multiperxn(countrydat, modeldat, cols, labs)
995
996 plots[["peryyn"]] <- plot_multiperyn(countrydat, modeldat, cols, labs)
997
998 return(plots)
999 }
1000
1001 multilist <- list()
1002 for(country in names(multidates)){
1003   multilist[[country]] <- multiphasePlots(country, multidates[[country]], owiddat)
1004 }
```

Listing 9: Main plotting functions

```

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 # make shape data ggplot-friendly
27 countyshp@data$id <- rownames(countyshp@data)
28 countyshp.points <- fortify(countyshp, region="id")
29 counties <- inner_join(countyshp.points, countyshp@data, by="id")
30 counties$CountyName <- gsub("County ", "", counties$NAME_EN)
31
32 # 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",

```

```

83         subtitle = paste("Cumulative, up to", format.Date(latest_date, "%B %d, %Y")) + 
84     map_theme() + theme(legend.position = c(0.25,0.87))
85
86 # just latest date
87 latest_date <- as.Date(owiddat$date[nrow(owiddat)-1], tryFormats = c("%Y-%m-%d"))
88 latest_dat <- owiddat[owiddat$date == latest_date,]
89 fortnightRows <- owiddat$date >= latest_date-13 & owiddat$date <= latest_date
90 fortnightCases <- aggregate(owiddat$new_cases_per_million[fortnightRows],
91                             by=list(owiddat$location[fortnightRows]), function(x) sum(x[!is.na(
92 x])))
93 colnames(fortnightCases) <- c("location", "fortnight_cases_per_million")
94 latest_dat <- left_join(latest_dat, fortnightCases, by=c("location" = "location"))
95
96 # make shape data ggplot-friendly
97 worldshp@data$id <- rownames(worldshp@data)
98 worldshp.points <- fortify(worldshp, region="id")
99 countries <- inner_join(worldshp.points, worldshp@data, by="id")
100
101 # join case numbers for latest date to country data, in order to colour nicely
102 world_map <- left_join(countries, latest_dat, by=c("CNTRY_NAME" = "location"))
103 world_map <- world_map[world_map$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"),
137   legend.position = c(0.1,0.3)))

```

Listing 10: Geospatial/Map plots

```

1 setwd("~/GitHub/TCD_FinalYearProject/Code")
2 source("covidplots-group.r")
3 source("countymap.r")
4 setwd("~/GitHub/TCD_FinalYearProject")
5
6 for(country in names(plotslist)){
7   for(p in names(plotslist[[country]])){
8     ggsave(filename = paste0(country, "-", p, ".pdf"),
9            plot = plotslist[[country]][[p]],
10            path = "./Plots",
11            height = 10,
12            width = 14,
13            units = "cm"
14    )
}

```

```

15}
16}
17
18 for(country in names(multilist)){
19   for(p in names(multilist[[country]])){
20     ggsave(filename = paste0(country, "-", p, ".pdf"),
21            plot    = multilist[[country]][[p]],
22            path    = "Plots",
23            height  = 10,
24            width   = 16,
25            units   = "cm"
26      )
27   }
28 }
29
30 for(pair in names(comparelist)){
31   ggsave(filename = paste0("compare-", pair, ".pdf"),
32          plot    = comparelist[[pair]],
33          path    = "Plots",
34          height  = 10,
35          width   = 16,
36          units   = "cm"
37      )
38 }
39
40 for(p in names(countyplotlist)){
41   ggsave(filename = paste0("county-", p, ".pdf"),
42          plot    = countyplotlist[[p]],
43          path    = "Plots",
44          height  = 14,
45          width   = 14,
46          units   = "cm"
47      )
48 }
49
50 for(p in names(worldplot)){
51   ggsave(filename = paste0("world-", p, ".pdf"),
52          plot    = worldplot[[p]],
53          path    = "Plots",
54          height  = 14,
55          width   = 20,
56          units   = "cm"
57      )
58 }

```

Listing 11: 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] Roy M. Anderson. “Discussion: The Kermack-McKendrick epidemic threshold theorem”. In: *Bulletin of Mathematical Biology* 53 (Mar. 1, 1991). ISSN: 522-9602. DOI: <10.1007/BF02464422>. URL: <https://doi.org/10.1007/BF02464422>.
- [4] Derdei Bichara, Abderrahman Iggidr, and Gauthier Sallet. “Global analysis of multi-strains SIS, SIR and MSIR epidemic models”. In: *Journal of Applied Mathematics and Computing* 44 (Feb. 2014), pp. 273–292. DOI: <10.1007/s12190-013-0693-x>.
- [5] Alexander Bird. *A simple introduction to epidemiological modelling—the SIR model*. <https://philosophyandmedicine.org/wp-content/uploads/2020/04/Introduction-to-epidemiological-modelling.pdf>. 2020.
- [6] Roger Bivand, Tim Keitt, and Barry Rowlingson. *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*. R package version 1.5-18. 2020. URL: <https://CRAN.R-project.org/package=rgdal>.
- [7] GeoHive Open Data Catalogue. *Covid-19 Daily Statistics for Ireland by County polygon as reported by the Health Surveillance Protection Centre*. 2020. URL: https://opendata-geohive.hub.arcgis.com/datasets/d9be85b30d7748b5b7c09450b8aede63_0.
- [8] © OpenStreetMap contributors. *Townlands*. 2020. URL: <https://www.townlands.ie/page/download/>.
- [9] Our World in Data. *The complete Our World in Data COVID-19 dataset*. 2020. URL: <https://covid.ourworldindata.org/data/owid-covid-data.csv>.
- [10] European Centre for Disease Prevention and Control. *Historical data on the daily number of new reported COVID-19 cases and deaths worldwide*. 2020. URL: <https://opendata.ecdc.europa.eu/covid19/casedistribution/>.
- [11] Jesús Fernández-Villaverde and Charles I Jones. *Estimating and Simulating a SIRD Model of COVID-19 for Many Countries, States, and Cities*. Working Paper 27128. National Bureau of Economic Research, May 2020. DOI: <10.3386/w27128>. URL: <http://www.nber.org/papers/w27128>.
- [12] Seth Flaxman. “Reply to: The effect of interventions on COVID-19”. In: *Nature* 588.1 (Dec. 1, 2020), pp. 29 –32. ISSN: 1476-4687. DOI: <10.1038/s41586-020-3026-x>. URL: <https://doi.org/10.1038/s41586-020-3026-x>.
- [13] Seth Flaxman and Imperial College COVID-19 Response Team. “Estimating the effects of non-pharmaceutical interventions on COVID-19 in Europe”. In: *Nature* 584.7820 (2020), pp. 257–261. ISSN: 1476-4687. DOI: <10.1038/s41586-020-2405-7>. URL: <https://doi.org/10.1038/s41586-020-2405-7>.
- [14] Dr Tedros Adhanom Ghebreyesus. *WHO Director-General's opening remarks at the media briefing on COVID-19*. World Health Organization. Mar. 11, 2020. URL: <https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020>.
- [15] Anthony Gibbons. *Final Year Project GitHub Repository*. 2021. URL: https://github.com/gibbona1/TCD_FinalYearProject.
- [16] Alexander Grigorian. *Mathematical riddles of COVID-19*. June 2020. URL: <https://www.math.uni-bielefeld.de/~grigor/corv.pdf>.
- [17] Robert J. Hijmans. *raster: Geographic Data Analysis and Modeling*. R package version 3.4-5. 2020. URL: <https://CRAN.R-project.org/package=raster>.
- [18] ArcGIS Hub. *UNIGIS Geospatial Education Resources*. 2020. URL: https://hub.arcgis.com/datasets/a21fdb46d23e4ef896f31475217ccb08_1.

- [19] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice, 2nd edition*. OTexts.com/fpp2. OTexts, 2018.
- [20] Rob J Hyndman and Yeasmin Khandakar. “Automatic time series forecasting: the forecast package for R”. In: *Journal of Statistical Software* 26.3 (2008), pp. 1–22. URL: <https://www.jstatsoft.org/article/view/v027i03>.
- [21] Roni Parshani, Shai Carmi, and Shlomo Havlin. “Epidemic Threshold for the Susceptible-Infectious-Susceptible Model on Random Networks”. In: *Physical Review Letters* 104.25 (June 2010). ISSN: 1079-7114. DOI: [10.1103/physrevlett.104.258701](https://doi.org/10.1103/physrevlett.104.258701). URL: <http://dx.doi.org/10.1103/PhysRevLett.104.258701>.
- [22] Karthik Ram. *Wes Anderson Palettes*. 2013. URL: <https://github.com/karthik/wesanderson>.
- [23] Arni S. R. Srinivasa Rao and Jose A. Vazquez. “Identification of COVID-19 can be quicker through artificial intelligence framework using a mobile phone-based survey when cities and towns are under quarantine”. eng. In: *Infection control and hospital epidemiology* 41.7 (2020). 32122430[pmid], pp. 826–830. ISSN: 1559-6834. DOI: [10.1017/ice.2020.61](https://doi.org/10.1017/ice.2020.61). URL: <https://pubmed.ncbi.nlm.nih.gov/32122430>.
- [24] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: <https://ggplot2.tidyverse.org>.
- [25] Hadley Wickham et al. *dplyr: A Grammar of Data Manipulation*. R package version 1.0.3. 2021. URL: <https://CRAN.R-project.org/package=dplyr>.