Mathematical epidemiology in a data-rich world

Julien Arino

Abstract

I discuss the acquisition and use of "background'' data in mathematical epidemiology models, advocating a pro-active approach to the incorporation of said data. I illustrate various mechanisms for acquiring data, mainly from open data sources. I also discuss incorporating this data into models.

Foreword to this version of the article

This is a version of the paper "Mathematical epidemiology in a data-rich world," which was published in *Infections Disease Modelling* in early 2020 and is accessible using this link. There were some issues with formatting in the published article. In particular, instead of code blocks, image files were generated. Also, some of the code I used at the time no longer ran: until all web available data gets persistent identifiers (such as doi for papers), this type of paper is doomed to run into issues, with data being moved or formats being changed.

In the present version, I have updated the code so that it runs again. I have also made small adaptations to the code: since version 4.0 of `R`, data frames automatically are created with `stringsAsFactors = FALSE`, so this command is not required anymore. The library `wp`, which is used in several places, has been completely revamped and the functions I was using were obsoleted. Besides these changes, the content of the paper is still exactly the same as the published version, save for the fact that some of data has more recent versions and since the code picks up the most up to date version, some figures are a bit more recent (this version was generated on Monday 14 February 2022).

Introduction

Data is now the world's most valuable resource. In 2018, the five most valuable companies in the world were, in decreasing order of valuation, Apple, Alphabet, Microsoft, Amazon and Facebook. Compare this with the situation 10 years earlier, where the top valued companies were, in decreasing order, PetroChina, Exxon, General Electric, China Mobile and ICBC.

This transition towards a data-driven world can also be apprehended when considering the wealth of information that is readily accessible on the Internet. Science is behind the technology that drives this information exchange, so it is not surprising that it would also be involved in generating some of that information. Many areas in the biological sciences are embracing this change. At the forefront, areas such as genomics and proteomics have most of their data openly accessible online. More and more ecological publications require that data be made available to others. Mathematical biology, because it is intrinsically connected to some of these domains, is also benefiting from this change. This abundance of data does not affect all areas of mathematical biology in the same way. Besides omics, population dynamics is a domain

that sees a lot of information put online. However, even within population dynamics, it is important to understand that a lot of data remains difficult to access; for instance, not all epidemic propagation events see their data be made readily available.

Altogether, despite these limitations, it is becoming increasingly evident that not using data when it is available should be a thing of the past. At the very least, a modeller should be *situationally aware*. What are the orders of magnitude of the numbers of individuals in the populations under consideration? What are the time scales involved in the evolution of the quantities being studied? These lecture notes are meant to provide some initial leads on the systematic use of data in the context of mathematical epidemiology.

I have two main goals here. The first is to give a very brief overview of the abundant resources available to develop an understanding of the context in which we are operating. My second goal is to illustrate simple techniques of data incorporation in models. Note that these lecture notes barely scratch the surface of a very rich domain area. Also, other lecture notes in this special issue explain in more detail how to deal with specific problems in the use of data. My aim is less ambitious: I advocate for a more integrated use of data and present techniques that can be used to inform models with data.

This document is organised as follows. In Section 1, I discuss data sources and in particular, the distinction between proprietary and open data. In Section 2, I describe some programmatic mechanisms for accessing and acquiring open data. Even when one has data, incorporating it into models is not necessarily straightforward; in Section 3, I use the very simple example of life expectancy as a cautionary example of the issues involved. The second part of the document presents case studies related to the spatial and temporal spread of epidemics. Some data and general ideas are shown in Section 4, with further discussion on metapopulation models, a way to apprehend this type of problems. Two examples are then given; the case of an SLIAR model for the spread of a disease between 5 countries is considered in Section 5 and the same type of model is used in Section 6 to study the spread of influenza between the regions of France.

Remarks about this document

To illustrate the philosophy of these lecture notes, this entire document is produced using `Rmarkdown` (link), an extension of the `R` programming language. `Rmarkdown` combines the `markdown` language, a simple markup language allowing LaTeX instructions and `R` *code chunks* that are executed when the code is run in `R`. I could also have used `sweave`, another `R` extension allowing, this time, to include `R` code in LaTeX. However, `Rmarkdown` has the advantage that, using almost the exact same file with very few modifications, one can also generate an `html` page.

The source file is accessible as an electronic appendix; it has the extension `Rmd`. All material is also available on my GitHub page (link). I will try to ensure that all links in this document remain current on the GitHub page; if some of the links provided here fail, refer to the document there. Data used to produce figures in this document was pulled off the web and are current as of the date of generation of the `pdf` of these lecture notes (2022-02-14). Some `R` codes are presented in the document, making for a slightly clunky feel. In a normal `Rmarkdown` document, this code would typically be hidden. I have hidden some instructions when they were redundant; they are nonetheless present in the `Rmd` file and their existence is indicated in the text by a comment. Also, in order to improve legibility, some long strings were pre-defined and comments were removed from the displayed code chunks. Finally, rather than applying a function to the result of a function, i.e., `f(g(x))`, I have sometimes used successive calls, i.e., `x <- g(x)` followed by `x <- f(x)`.

To generate the document from the provided `Rmd` file, the following (free) programs are required.

- A recent version (≥ 3.5) of the `R` programming language, which can be downloaded here (link).

- Although not mandatory, using `RStudio` (link) greatly facilitates both `R` programming and, more importantly, the generation of this `pdf` file from the `Rmarkdown` source.
- A functional LaTeX installation is required.
- Several `R` packages (the list of packages used appears in the setup chunk of the `Rmarkdown` code).

Web access is also required. In order to accommodate readers with limited Internet access, the electronic appendix and GitHub repository include a copy of the data current as of the date of compilation. In the first chunk of the `Rmd` file, setting `DOWNLOAD=FALSE` will trigger the use of this downloaded data rather than online one. As a consequence, all web-based queries in the text take the form

```
if (DOWNLOAD) {
  Commands downloading data from the web
}
```

The code could be simplified by removing this check and just running said commands. Two additional remarks about using `Rmarkdown` to produce such a file. First, it is a good idea to name chunks, as this helps when debugging. This is easily done, by adding a name after the call to `R` and before any chunk options, with the chunk header taking the form

```
{r name_of_chunk, chunk_options..}
```

Second, in the provided `Rmarkdown` file, the following chunk options were set globally:

```
warning=FALSE, message=FALSE
```

This has the effect of removing most warning messages. While this is a good idea for a production-ready document such as this one, it should be removed while developing.

1 Data is everywhere

1.1 Proprietary data versus Open data

The Internet contains an enormous quantity of data, at scales that have become virtually impossible to quantify. As one searches for data on this medium, one is confronted to two main types of resources: *proprietary data* and *open data*.

Proprietary data is often generated by companies, governments or research laboratories. It is either impossible to access, or its access is heavily regulated. At the other end of the spectrum, open data is easily accessed. It is often data originating from the same sources as proprietary data, but it is released for common use, typically after a cool-off period. It is important to note that even when data is *open*, it is subject to a variety of licensing frameworks. Before using open data, it is therefore important to establish what type of licensing it is offered under; it is also important to establish citing mechanisms for that data. Another concern with open data is that it can be of varying qualities. This should be ascertained; some brief notions of data quality insurance are discussed later.

1.2 Open data initiatives

An exciting trend for modellers, which started 5 to 10 years ago and is becoming increasingly common, are open data initiatives. Such initiatives see governments (local or higher) create portals where data is centralised and made accessible, usually with very few constraints. The following illustrate these initiatives, from local to global scope:

- https://data.winnipeg.ca/
- https://open.alberta.ca/opendata
- https://open.canada.ca/en/open-data
- https://data.europa.eu/euodp/data/
- http://data.un.org/
- https://data.worldbank.org/
- https://www.who.int/gho/database/en/

2 Acquiring data

Before I review data acquisition methods, let me remark that even when data is readily available online, it is always good practice to keep a copy of the downloaded data once it has been acquired. Indeed, data is sometimes removed or moved, or it can be difficult to access because of poor or nonexistent internet connection or state-imposed filtering.

2.1 Retrieving data from open data portals

There are three main methods for retrieving data from open data portals:

1. Do it *by hand*;
2. Use the site's API if there is one;
3. Use an `R` (or another language) library designed for that.

I now review these methods.

2.1.1 Retrieving data 'by hand'

This proves to be the most annoying way to acquire data from open data portals, by a large measure. To illustrate using this method, suppose I were to browse to the World Health Organisation Global Health Observatory site (link) and follow links there until I find data giving the number of reported cases of measles per country per year, for instance (link). At the time of writing, the WHO GHO data site has recently moved to a new platform, which at present allows exporting the data tables only in `pdf` or `png` form. Those interested in getting the actual data can still (at the time of writing) find this data on another part of the WHO website (link).

2.1.2 Using an API

Application programming interfaces (API) allow *client-side* url-based access to *server-side* functions. In other words and in particular, API can be used to query databases hosted on the Internet. The presence of an API is typically indicated by long URL (web addresses) including symbols such as (`?`, `:`, `*`, `&`). API have become ubiquitous in recent years; it has become quite common to use them to perform quite a few operations server-side.

The WHO GHO data in Section 2.1.1 can be accessed using two different API. Both are well documented; the one I use below (*Athena*) is documented here (link), with, most importantly, some examples (link).

Before going into a bit more detail about the use of API, let me give the example of accessing the WHO GHO measles data mentioned in Section 2.1.1. To do so, I must construct a URL that spells out my query.

In the following chunk, I list the various components required.

```
base_url_who <- "http://apps.who.int/gho/athena/data/GHO/"
indicator <- "WHS3_62"
options1 <- "?filter=COUNTRY:*;REGION:*"
options2 <- "&x-sideaxis=COUNTRY&x-topaxis=GHO;YEAR"
options3 <- "&profile=crosstable"
options4 <- "&format=csv"
```

These components are relatively easy to identify, even without referring to the API documentation. `options1` indicates that the result should include all countries and all regions (the different geographic groupings used by the United Nations and other agencies). `options2` says the rows should be the countries/regions and columns the value of the selected index for each year. `options3` specifies the type of table. Finally, `options4` forces the return value to be a `csv` table. To get the data set in memory directly, it then suffices to reconstruct this address and use the `R` function `read.csv`.

```
full_url <- paste0(base_url_who, indicator, options1, options2,
    options3, options4)
if (DOWNLOAD) {
    measles_data <- read.csv(full_url, header = TRUE, skip = 1)
}
```

The option `skip = 1` is used because there are two lines of information at the top of the `csv` file that are not part of the table itself. The variable `base_url_who` will be used again later. Using the command above, one would end up with a data frame (a common `R` data type), `measles_data`, containing the table under consideration.

Later, in Section 2.1.3, I show some examples of API for which there exists `R` libraries allowing easy access to the data. However, as far as I am aware, at the time there is no such library for the WHO GHO data. However, it is easy to see how the ideas that follow could be made a little more robust and turned into such a library. Therefore, let me briefly explain how one could programmatically browse the content of the WHO GHO database.

Reading the API documentation, top level WHO information comes in XML format, so I have to play with this a bit. Let me gather all the information in one place.

```
if (DOWNLOAD) {
    GHO_xml <- xmlParse(base_url_who)
    GHO_root <- xmlRoot(GHO_xml)[["Metadata"]][["Dimension"]]
    GHO_label <- as.character(xmlSApply(GHO_root, xmlGetAttr,
        "Label"))
    GHO_url <- xmlSApply(GHO_root, xmlGetAttr, "URL")
    GHO_display <- mat.or.vec(nr = xmlSize(GHO_root), nc = 1)
    for (i in 1:xmlSize(GHO_root)) {
        if (length(xmlChildren(GHO_root[[i]])[["Display"]]) >
            0) {
            tmp <- xmlChildren(GHO_root[[i]])[["Display"]]
            GHO_display[i] <- as.character(xmlValue(tmp))
        }
    }
```

Description	Name
Poliomyelitis - number of reported cases	WHS3_49
Polio (Pol3) immunization coverage among 1-year-olds (%)	WHS4_544
Polio immunization coverage among one-year-olds (%)	poliov
Polio immunization coverage among one-year-olds (%)	vpolio

```
}
```

XML documents are structured documents. What the previous chunk of code does is that it browses the XML document tree. I then make a data frame of the result, then tidy up a bit.

```
if (DOWNLOAD) {
    GHO_indicators <- as.data.frame(cbind(GHO_display, GHO_label,
        GHO_url))
    GHO_indicators <- GHO_indicators[which(GHO_indicators$GHO_display !=
        "0"), ]
    colnames(GHO_indicators) <- c("Description", "Name", "URL")
    rownames(GHO_indicators) <- 1:dim(GHO_indicators)[1]
}
```

As a result of this, I have a data frame containing 3287 indicators. This data frame can now be mined for information. Let me for instance find indicators that contain the word `polio`. (Note that the search is performed in a case-insensitive way, so that `polio` and `Polio` are both acceptable results.)

```
idx_polio <- grep("polio", GHO_indicators$Description, ignore.case = TRUE)
```

I obtain the following result.

Using the name of one of the indicators in the table above (e.g., `poliov`) as `indicator` in the first chunk in this section would then allow to load the corresponding dataset.

2.1.3 Using existing `R` libraries

`R` abounds with packages allowing to perform easily the operations I carried out with the WHO GHO indicators in Section 2.1.2. (`Python` also has plethora of packages to access API, but for the present document I use only `R` solutions.) To name a few,

- `WDI`: query World Development Indicators (from World Bank).
- `wbstats`: download World Bank data; I illustrate the use of this library later.
- `openstreetmap`: access Openstreetmap data. This is very useful for mapping.
- `tidycensus`: get USA census data; this requires an API key, i.e., one needs to obtain a key from the relevant authority.
- `cdcfluview`: access the US CDC flu surveillance data.

Let me illustrate such libraries by using `wbstats` to find country life expectancy and population information.

```
if (DOWNLOAD) {
    new_cache_wb <- wb_cache()
    lifeExpectancy_vars_wb <- wb_search(pattern = "expectancy",
```

```
        cache = new_cache_wb)
    pop_vars1_wb <- wb_search(pattern = "population", cache = new_cache_wb)
}
```

My query for life expectancy data returns 39 results. Parsing through `lifeExpectancy_vars_wb`, I find the index `SP.DYN.LE00.IN`, whose description reads *Life expectancy at birth, total (years)*. On the other hand, my search for 'population' returns 4462 results, so a more refined search is carried out.

```
if (DOWNLOAD) {
    pop_vars2_wb <- wb_search(pattern = "total population", cache = new_cache_wb)
}
```

With this more refined query, I find a suitable candidate, `SP.POP.TOTL`, which is described as *Population, total*. I now use the function `wb` to download the corresponding data.

```
if (DOWNLOAD) {
    lifeExpectancy_data_wb <- wb_data(indicator = "SP.DYN.LE00.IN",
        start_date = 2000, end_date = curr_year, return_wide = FALSE)
    pop_data_wb <- wb_data(indicator = "SP.POP.TOTL", start_date = 2000,
        end_date = curr_year, return_wide = FALSE)
}
```

Note that the World Bank data has results for countries as well as for groups of countries. The latter are useful as they allow to work at a broader scale, both geographic (for instance, *Caribbean small states*) and economic (for instance, *Fragile and conflict affected situations*).

Since World Bank data queries are particularly important in this document, let me make a few additional remarks.

- One useful option to the function `wb` is `mrv =`, which returns the indicated number of *most recent values* for the index (or the number of values present in the database if it is lower than the number requested). See an example of usage below.
- However, not all countries have all years present. If querying for more than one country, it is best to instead give a wide range of `startdate` and `enddate`, then find the most recent year for each country, as using `mrv` can have an unintended consequence in this case. A function, `latest_values_general`, is provided in the companion file `useful_functions.R`, which can help with this task. See an example in Section 5.3.2.
- Using the option `POSIXct = TRUE` returns dates that are easier to process, especially for monthly data.

Let me illustrate the use of `mrv` by plotting the population of China. Remark that it is often useful, when exploring data or presenting simulation results, to ensure that axes are easy to read. So instead of the usual `plot` command, I use the function `plot_hr_yaxis` (in the file `useful_functions.R`) that labels units of the y-axis of the plot in a more human readable way.

```
if (DOWNLOAD) {
    pop_data_CHN_wb <- wb_data(country = c("CHN"), indicator = "SP.POP.TOTL",
        mrv = 100)
}
plot_hr_yaxis(pop_data_CHN_wb$date, pop_data_CHN_wb$value, xlab = "Year",
    ylab = "Population", type = "l")
```

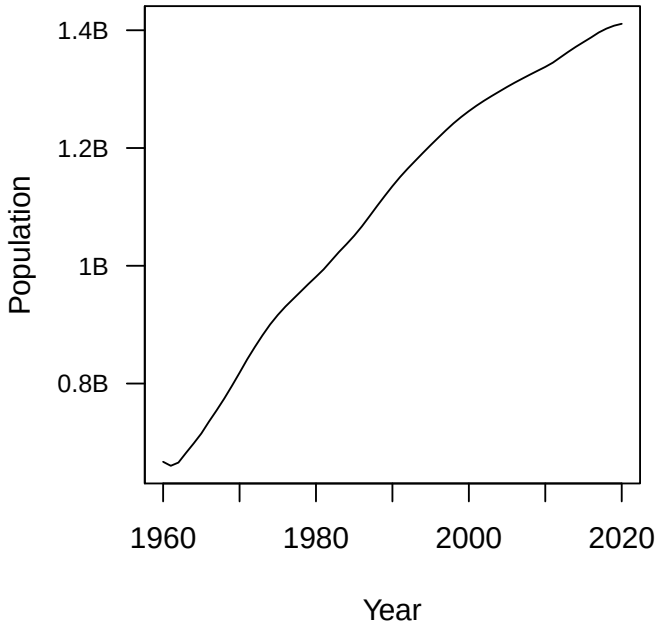


Figure 1: Evolution of the population of China as given by World Bank data.

2.2 Something intermediate – `htmltab`

While being able to obtain data by querying an API is ideal, there are many cases where this is not an available option. It is however easy to grab data from tables found on web pages, using the `R` library `htmltab`. Remark that it is also possible to extract data from tables in `pdf` files, but this is not covered in these notes. To illustrate the use of `htmltab`, let me compute the population density of countries using two tables grabbed from Wikipedia (link1,link2). Note that this is a futile exercise, as Wikipedia also includes tables with population density information, but it serves to illustrate another important `R` command, which allows to merge tables.

Be careful when compiling this `Rmd` file: Wikipedia comments can appear as tables, so you may have to change the option `which = 2` as it is currently set to a different value if the following commands yield an error.

```
if (DOWNLOAD) {
    page_url <- "List_of_countries_and_dependencies_by_population"
    url <- paste0(wiki_url, page_url)
    pop_wiki <- htmltab(url, which = 1)
    page_url <- "List_of_countries_and_dependencies_by_area"
    url <- paste0(wiki_url, page_url)
    surf_wiki <- htmltab(url, which = 1, rm_nodata_cols = FALSE,
        stringsAsFactors = FALSE)
}
```

Now that the data has been acquired, some processing is needed. Tables in Wikipedia show numbers with comma separating groups of three digits; these commas need to be removed. Surface areas are also provided in both kilometres and miles; I remove the latter. Finally, I rename some columns for convenience.

```
colnames(pop_wiki)[c(2, 4)] <- c("state", "population")
colnames(surf_wiki)[2:5] <- c("state", "total", "land", "water")
pop_wiki$population <- as.numeric(gsub(",", "", pop_wiki$population))
for (i in 3:5) {
    surf_wiki[, i] <- gsub(",", "", surf_wiki[, i])
    pos <- regexpr(pattern = "\\(", text = as.character(surf_wiki[,
        i]))
    surf_wiki[, i] <- as.numeric(substr(as.character(surf_wiki[,
        i]), 1, pos - 1))
}
```

Note that `htmltab` allows to apply a function to each column in the table, so some of this processing could have been carried out while the table was being downloaded. I now use an important function, `merge`, to merge data frames with columns containing some common entries. This performs the equivalent of a JOIN command in SQL.

```
pop_surf_wiki <- merge(x = pop_wiki, y = surf_wiki, by.x = "state",
    by.y = "state")
pop_surf_wiki <- pop_surf_wiki[, c(1, 4, 9)]
colnames(pop_surf_wiki) <- c("name", "pop", "surf")
pop_surf_wiki$dens <- pop_surf_wiki$pop/pop_surf_wiki$surf
```

Let me now show how to map spatial results. There are many `R` libraries for mapping. One possible way is to proceed as follows. First, I need to translate place names into ISO 3166 (country) codes, set up bins for values and set up a colour palette. Note that bins are set up here using 20 percentiles. Indeed, using a linear scale results in too little contrast.

```
pop_surf_wiki$iso3c <- countrycode(pop_surf_wiki$name, origin = "country.name",
    destination = "iso3c")
pop_surf_wiki <- pop_surf_wiki[!is.na(pop_surf_wiki$iso3c), ]
for (i in 1:length(pop_surf_wiki$iso3c)) {
    idx <- which(iso3166$a3 == pop_surf_wiki$iso3c[i])
    pop_surf_wiki$name[i] <- iso3166$mapname[idx]
}
pop_surf_wiki <- pop_surf_wiki[which(!is.na(pop_surf_wiki$iso3c)),
    ]
brks <- quantile(x = pop_surf_wiki$dens, probs = seq(0, 1, 0.2))
nb_bins <- length(brks) - 1
pop_surf_wiki$bin <- findInterval(pop_surf_wiki$dens, vec = brks)
my_palette <- colorRampPalette(c("yellow", "red"))(n = nb_bins)
```

Note that I have pruned the list of countries to get rid of ones that pose problem to the mapping routine (typically, small overseas dependencies). Proper use would dictate to go through the list of errors and establish the corresponding region name. Now that places have been binned in terms of their 20 percentile, I plot them.

```
map("worldHires", fill = TRUE, col = "gray95", ylim = c(-55,
    85))
for (i in 1:length(pop_surf_wiki$iso3c)) {
```

```
    try(map("worldHires", region = pop_surf_wiki$name[i], add = TRUE,
        fill = TRUE, col = my_palette[pop_surf_wiki$bin[i]],
        ylim = c(-55, 85)), silent = TRUE)
}
```

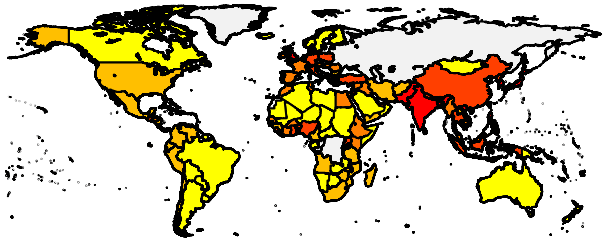


Figure 2: Population density.

2.3 The extreme – Data extraction from figures

One rather anecdotal method is available when data is not open *per se*, which is to capture numerical data from a figure in a publication. Note that this is by far the most time onerous method of data acquisition and should typically be a last resort option.

As soon as work is published, the figures contained therein indeed become part of the common good and the data there can be used, with proper citation of the original work. Many publications nowadays encourage publication of the data, so one should first check if the data used in the paper has been published. In the case where the data is not available online, though, for instance in old papers, one can digitise using programs such as Engauge Digitizer (link) or g3data (link). These programs typically present an interface in which the figure whose data must be digitised is displayed. In a first step, the user enters several reference points on the figure with known positions, for example, the origin and two points known on coordinate axes. Then each point of the data is clicked on and the result is generated as a `csv` file.

To illustrate this method, let me first download the US census data from Wikipedia and plot it. In this case, I save the figure to a file in order to then process it through the digitiser. Note that to make the plot easier to use, I also print a grid, which makes setting reference points easy.

```
if (DOWNLOAD) {
    us_census <- htmltab(paste0(wiki_url, "United_States_Census"),
        which = 2)
}
colnames(us_census)[1:2] <- c("year", "population")
us_census$population <- as.numeric(gsub(",", "", us_census$population))
png(filename = "FIGS/US_census_1790_2010.png", width = 800, height = 600)
plot(us_census$year, (us_census$population/1e+06), xlab = "Year",
    ylab = "Population (Millions)")
grid()
dave <- dev.off()
```

Note that the last instruction is used to completely remove the output of using `dev.off()`, which, interestingly, does not seem to obey the `warning=FALSE, message=FALSE` directive given to `knitr`.

After processing this image with Engauge Digitizer, I obtain a `csv` file (provided in the electronic appendix).

In order to visually check the result, I plot in Figure 3 the content of this file (in red) together with the original data (in black).

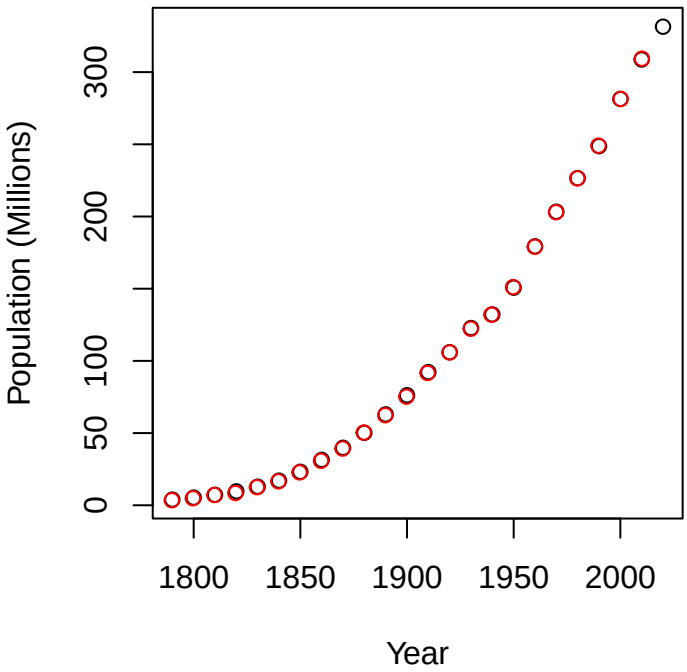


Figure 3: Number of people in the USA as given by the census. Black: original data; red: digitised data.

Clearly, the agreement is very good, since one can barely see any black, meaning the red points completely cover the black ones. The situation was made as ideal as possible, however. In practical cases, the agreement might not be as good. This should be considered a last resort method and good practice dictates that additionally to citing the work from which the data originates, one should additionally mention that the data is digitised, since errors stemming from imprecision in digitisation should not be blamed on the authors of the original work.

3 Incorporating data in models – Initial remarks

Now that the most basic mechanisms for acquiring data from the web have been explained, let me discuss how this data can be incorporated into population dynamics models. Remark, however, that while it is important to be aware of the magnitudes and time scales involved in the processes studied, it is also important to not use data just for the sake of using data. Also, note that I do not discuss in these lecture notes another aspect of data acquisition that is very important, namely the verification and validation of data.

There are a variety of ways for data to be incorporated into dynamical models. The main are as parameters or initial conditions, or as time series with which model solutions can be compared in order to identify other parameters. In essence, as far as model simulations go, parameters and initial conditions can be thought of as being of the same nature, mimicking the theoretic similarity between continuous dependence on initial values and parameters. So, by abuse of language, in these notes, I often call parameters both parameters and initial conditions. Parameters can be classified for instance in terms of their reliability or the way they are derived.

1. *Reliably known* parameters are related to geography, population, vaccination coverage, region centroids,

etc. A typical example would be the population of a country; there can be some uncertainty about the exact value, but the *posted* value can be taken as given.
2. *Known* (or relatively well known) parameters are disease characteristics such as the incubation period or duration of the infectious period. These are typically derived from expert knowledge based on statistical analysis of characteristics.
3. *Imputed* parameters are parameters whose values are not known precisely but can be computed from known parameters by making some assumptions on processes. A typical example is the vaccination rate: it can be derived in a given model from the knowledge of the vaccine coverage.
4. *Identified* parameters are typically obtained by comparing the outcome of the model with a known time series given as data.

3.1 Example – Life expectancy

To illustrate the difficulty of dealing with data, let me consider what is, at first glance, a very easy parameter: *life expectancy*. Suppose I am tracking a cohort of individuals born at a certain time t_0, where the only cause of death is natural death, which occurs at the *per capita* rate d. (Because I am tracking a cohort, there is also no birth into the population.) Without going into details (see for example (Thieme 2003)), the hypothesis underlying the differential equation for the rate of change of the number $N(t)$ of individuals in this population,

$$N' = -dN,$$

is that the duration of life for each individual is exponentially distributed with mean $1/d$. The appropriateness of this hypothesis depends on the aim of the model.

- If one uses the model for long term predictions, then the hypothesis is valid, since over several generations, the important characteristic can be safely assumed to be the mean duration of life. In this case, we can just set $1/d$ to be the life expectancy data grabbed earlier.
- Now suppose that the model is to be used for short term predictions. In this case, the hypothesis of exponential distribution of life durations becomes a problem, as I illustrate in the following example.

Let me consider the population of China. Querying the World Bank data, life expectancy at birth in China in 2019 was 76.912 years. Now recall that for a random variable with exponential distribution with parameter d (or mean $1/d$), the survival probability is given by $\mathcal{S}(s) = \mathbb{P}(t > s) = 1 - \mathbb{P}(t \leq s) = e^{-ds}$. Using the value 76.912 years for the inverse of the death rate, it follows that the proportion of individuals in a cohort born at time $t_0 = 0$ who survive to age $t = s$ is as shown in Figure 4.

If considering a model for the long term behaviour of the population, the important characteristic is the red vertical line, i.e., the mean. However, if one is interested in the short term dynamics, then there are two issues:

1. The initial attrition is too high. In Figure 4, only 80% of the initial cohort remains after 30 years, which is much less than what it is in real life.
2. The tapering off is too slow. Almost 20% of the cohort survives to be 150 years old.

Here, I show two simple methods for addressing this problem. There are many others, more appropriate (but more complex) ones.

- Refine the parameter of the exponential distribution to take into account known survival to given ages.
- Use the fact that the Erlang (Gamma) distribution is the sum of exponential distributions, i.e., add compartments.

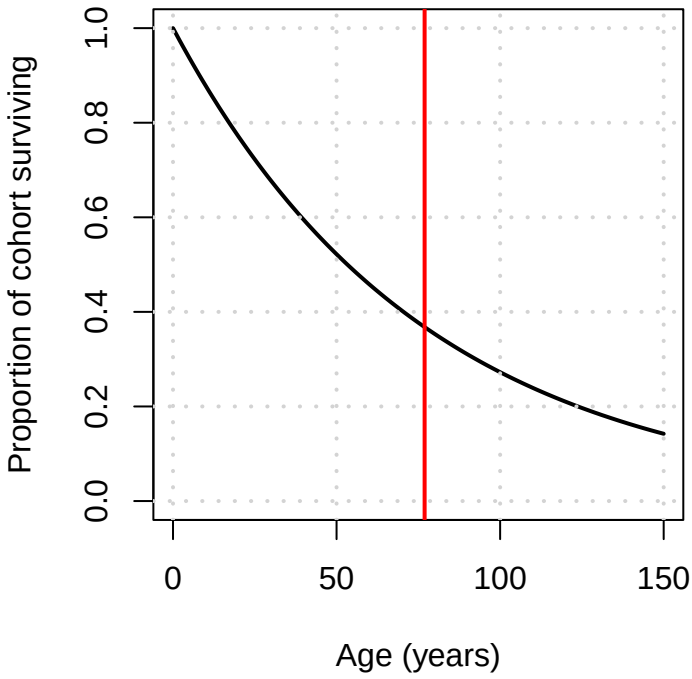


Figure 4: Proportion of a cohort surviving when the mean of the exponential distribution equals the life expectancy at birth in China in the latest available value.

3.1.1 Refining the parameter of the exponential distribution

Looking through World Bank indicators as in Section 2.1.3, I find the indicator `SP.POP.65UP.TO.ZS` that has "Population ages 65 and above (% of total)' ', which I could use to refine the survival function. Call p_{65} that proportion, then

$$\mathcal{S}(65) = p_{65} \iff e^{-65d} = p_{65} \iff d = -(\ln p_{65})/65.$$

Grabbing the data from the World Bank, 11.97% of the population of China was over 65 in 2020. From the formula above, in order to have the (exponentially distributed) lifetime such that $\mathcal{S}(65) = p_{65}$, the mean lifetime should be 30.62 years.

Figure 5 shows the original and the adjusted distributions. Note that while the slow tapering off is resolved with this new value of d, this comes at the price of an even higher early attrition of the population.

3.1.2 Using an Erlang as a sum of Exponentials

Let X_i be independent exponentially distributed random variables with parameter ξ and $Y = \sum_{i=1}^{n} X_i$. Then, the random variable $Y \rightsquigarrow E(n, \xi)$, an Erlang distribution with n the *shape* parameter and ξ the *scale* parameter. (An Erlang distribution is a Gamma distribution with integer scale parameter.)

In terms of compartmental models, this means that if n compartments are traversed successively by individuals, with each compartment having an outflow rate of $1/\xi$ (or a mean sojourn time of ξ), then the time of sojourn from entry into the first compartment to exit from the last is Erlang distributed with mean $E(Y) = n\xi$ and variance $\mathsf{Var}(Y) = n\xi^2$. This is illustrated in Figure 6. For a single compartment as in Figure 6a, the time of sojourn is exponentially distributed following the left-most (yellow) curve in Figure 7a. Adding compartments with the same mean sojourn time per compartment results in increasingly red curves in Figure 7a. Figure 7b shows the corresponding survivals.

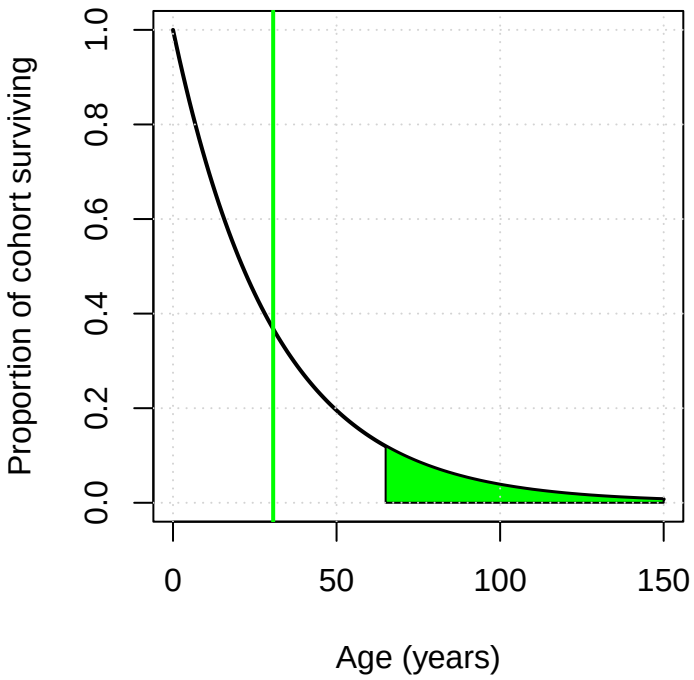
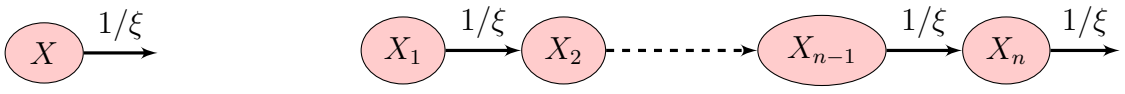


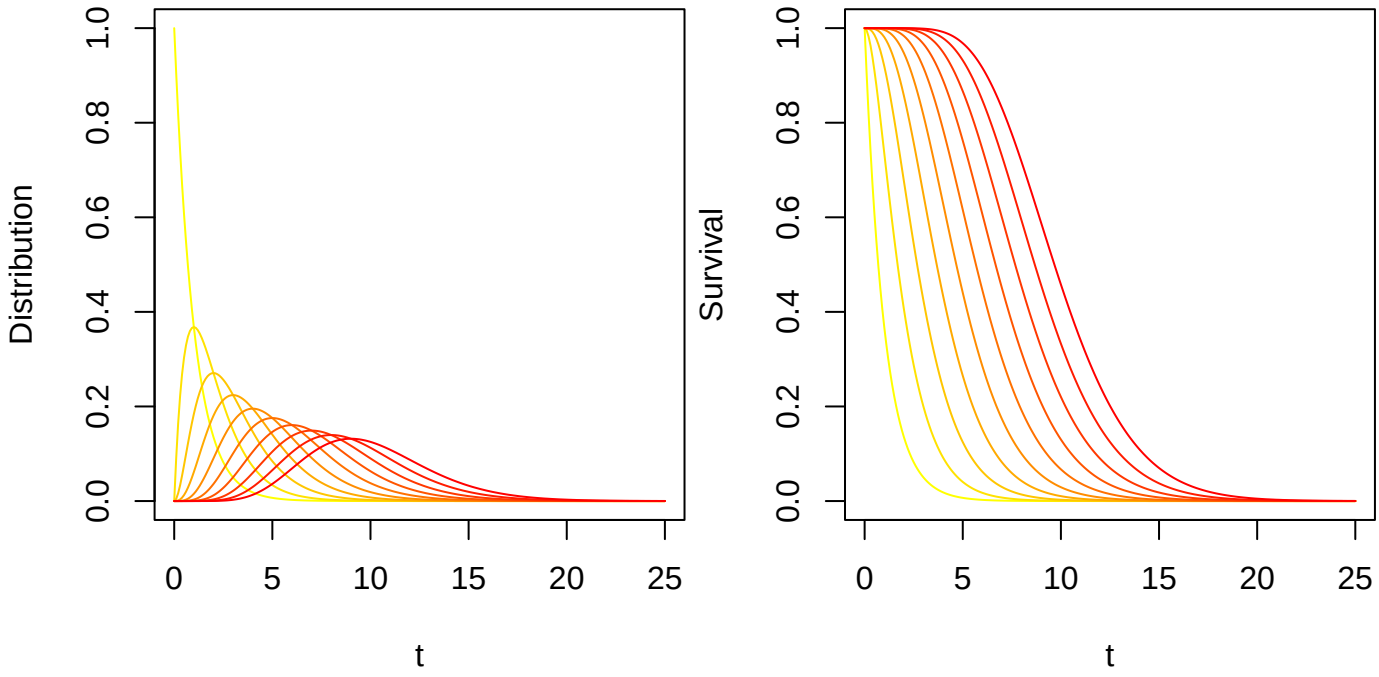
Figure 5: Adjusted (exponential) life expectancy distribution for CHN, taking into account the desired area under curve for survival of 65 years and older.



(a) 1 compartment.

(b) n compartments.

Figure 6: (a) Single compartment case, the time of sojourn is exponentially distributed. (b) Multiple compartments case, the resulting time of sojourn in the chain of compartments is Erlang distributed.



(a) Distribution (b) Survival

Figure 7: Erlang distributions with rate equal to 1 and shape parameters varying from 1 (yellow) to 10 (red). (a) Distribution. (b) Corresponding survival.

As an example of the use of adding compartments to fit known sojourn time distributions, let me consider the incubation period for Ebola Virus Disease. During the 2014 EVD crisis in Western Africa, the WHO Ebola Response Team estimated incubation periods in the paper (WHO Ebola Response Team 2015). Table S2 in the Supplementary Information in that paper gives the best fit for the distribution of incubation periods for EVD as a Gamma distribution with mean 10.3 days and standard deviation 8.2, i.e., $n\varepsilon = 10.3$ and $\varepsilon\sqrt{n} = 8.2$. From this, I obtain that $\varepsilon = 8.2^2/10.3 \simeq 6.53$ and $n = 10.3^2/8.2^2 \simeq 1.57$. However, that is a Gamma distribution.

In order to fit within the context of using multiple compartments to better fit residence times, since the number of compartments is an integer I need to find the closest possible Erlang distribution to this Gamma distribution. To do this, let me compute the square of errors between data points generated from the given Gamma distribution and an Erlang. The following function computes the square of the difference between data points (t_i, d_i) and a Gamma distribution with shape `shape` and scale `theta`, evaluated at the same t_i. (To get an Erlang distribution, `shape` needs to be an integer.)

```
error_Gamma <- function(theta, shape, t, d) {
    test_points <- dgamma(t, shape = shape, scale = theta)
    ls_error <- sum((d - test_points)^2)
    return(ls_error)
}
```

The following function takes as input data points (t_i, d_i) and finds optimal scale and integer shape parameters (so an Erlang distribution) corresponding to these data points. Note that the shape parameter is (arbitrarily) limited to 10, i.e., I allow at most 10 compartments. Note that I use `try` to avoid issues linked to the potential non-success of the call to `optim`.

```
optimize_gamma <- function(t, d) {
    max_shape <- 10
    error_vector <- mat.or.vec(max_shape, 1)
    scale_vector <- mat.or.vec(max_shape, 1)
    for (i in 1:max_shape) {
        result_optim <- try(optim(par = 3, fn = error_Gamma,
            lower = 0, method = "L-BFGS-B", shape = i, t = t,
            d = d), TRUE)
        if (!inherits(result_optim, "try-error")) {
            error_vector[i] <- result_optim$value
            scale_vector[i] <- result_optim$par
        } else {
            error_vector[i] <- NaN
            scale_vector[i] <- NaN
        }
    }
    result_optim <- data.frame(seq(1, max_shape), scale_vector,
        error_vector)
    colnames(result_optim) <- c("shape", "scale", "error")
    result_optim <- result_optim[complete.cases(result_optim),
        ]
    return(result_optim)
}
```

Finally, I call the function above with parameters of the Gamma distribution as given in the paper. If you had your own data points, you could use them instead in the chunk below. (The points in time for your data would be in the vector `time_points`, while the corresponding values would be in `data_points`.)

```
time_points <- seq(0, 60)
data_points <- dgamma(time_points, shape = 1.57, scale = 6.53)
# Run the minimization
optim_fits <- optimize_gamma(time_points, data_points)
# Which is the best Erlang to fit the data
idx_best <- which.min(optim_fits$error)
```

Now plot the result as well as the original curve, giving Figure 8 (code chunk not shown).

4 Spatial spread of epidemics of humans

I now consider two real-life examples dealing with the spatial spread of epidemics, with focus on epidemics of human diseases and more specifically, influenza. I first motivate the need to consider the spatial spread of infections, using some of the techniques presented in the preceding sections of these lecture notes.

4.1 Motivation – Pathogen spread has evolved with human mobility

Pathogens infecting humans spread over space and time together with the humans carrying them. As a working definition, let me define *mobility* as the collection of processes through which individuals change

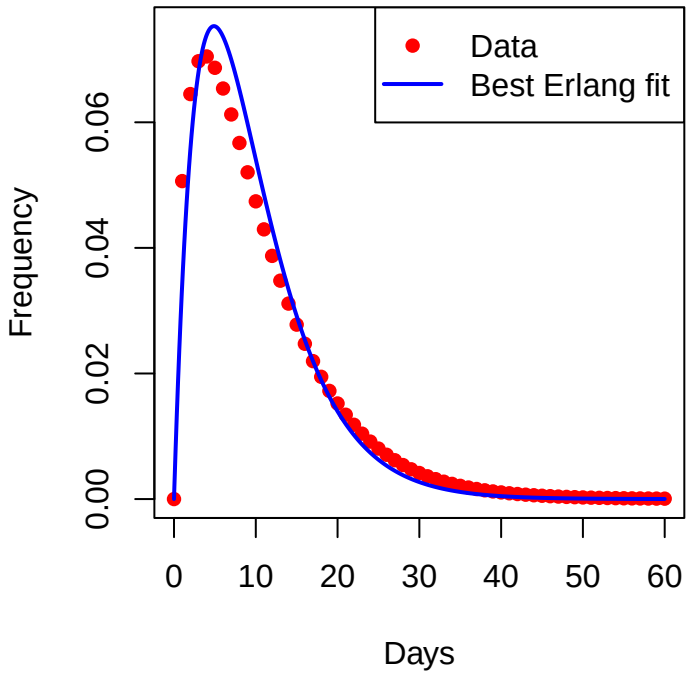


Figure 8: Best Erlang fit of the Gamma given in [?].

their current location. Until the advent of leisure travel in the early 20th century, long range human mobility was mainly along trade routes. Now mobility has evolved and as a consequence, so has the spread of pathogens humans carry.

Using the very broad definition of mobility given earlier, it is clear that the scale of modern mobility is difficult to apprehend. It takes many different forms, is constantly evolving and involves numbers that are colossal.

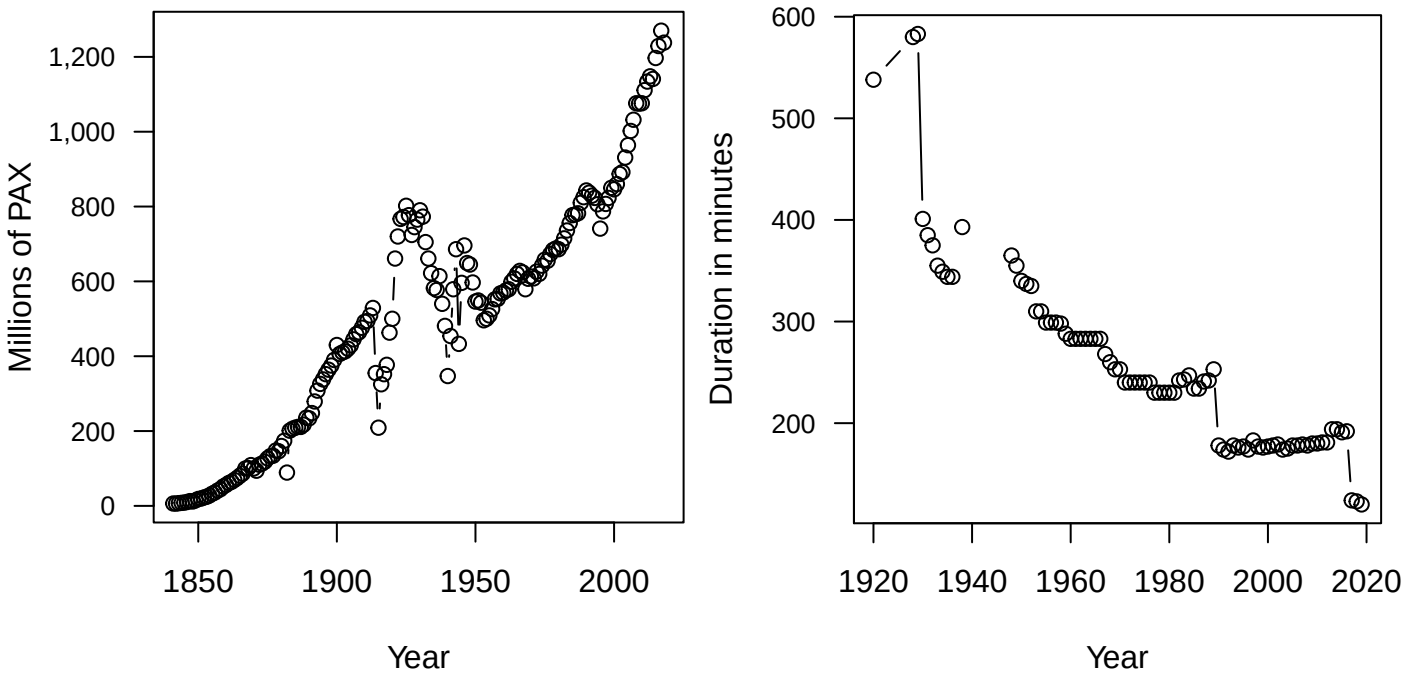
In order to illustrate one of my points in these lecture notes, namely, that a modeller needs to be situation-aware, let me show how one could gather evidence concerning the evolution of travel. This evidence will not be used directly in the models, but I do feel that its knowledge is important in model formulation and simulation.

In Figure 9, I show the number (in millions) of people-trips taken on the French national railway network (SNCF) since 1841 and the evolution of the duration of a trip between Paris and Bordeaux since 1920 on the same network. (Code chunk not shown.)

Thus, clearly, the number of passengers transported by train has increased considerably over the past 150 years, while the amount of time it takes for passengers to cover distances has dropped. Another interesting component is the number of incoming tourists worldwide, obtained from the World Bank. This is shown in Figure 10.

Note that in the chunk used to obtain this figure, I illustrate another method for processing data: the use of the `sqldf` library, which allows to use SQL-type queries on `R` dataframes. Note that `WLD` stands for World.

```
if (DOWNLOAD) {
    tourism_data_arrivals_wb <- wb(indicator = "ST.INT.ARVL",
        startdate = 2000, enddate = 2018)
}
query <- "SELECT date, value
```



(a) Passengers transported

(b) Duration of Paris-Bordeaux

Figure 9: Evolution of the annual number of passengers transported since 1841 and of the duration of a Paris to Bordeaux trip since 1920, by train, in France.

```
FROM tourism_data_arrivals_wb
WHERE iso3c='WLD'
ORDER BY date"
tourism_world <- sqldf(query)
plot_hr_yaxis(tourism_world$date, tourism_world$value, xlab = "Year",
    ylab = "Incoming tourists", type = "b", lwd = 2)
```

Another increasingly popular method for parsing through data not illustrated in these lecture notes are the libraries in the `tidyverse`. They are extremely useful, but as someone who has used SQL a lot in other context, I tend to use `sqldf` more. If you are agnostic in terms of method, I do recommend learning the `tidyverse`.

The massive increase (almost doubling) in inbound tourists since the early 21st century is due to a large extent to the increase of tourism from China and other emerging markets. A lot of tourism travel involves air travel and the rise of numbers in this context is also quite visible, as shown in Figure 11, with data also originating from the World Bank. (Code chunk not shown.)

As a final illustration of the importance of mobility, this time at a more local scale, see this animation of the scheduled positions of buses in the City of Winnipeg (link). To create the animation there, the bus schedules were downloaded and plotted. (The code is available on the linked page.)

4.2 Metapopulation models

Since mobility has become such an important component of the every day life of humans, one needs ways to model this in relation to the spread of infectious diseases. There are many different approaches to do

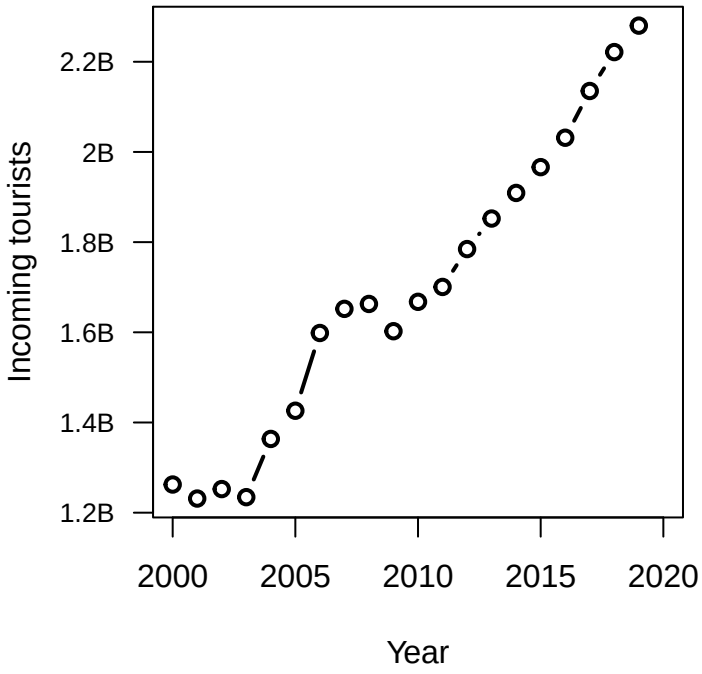
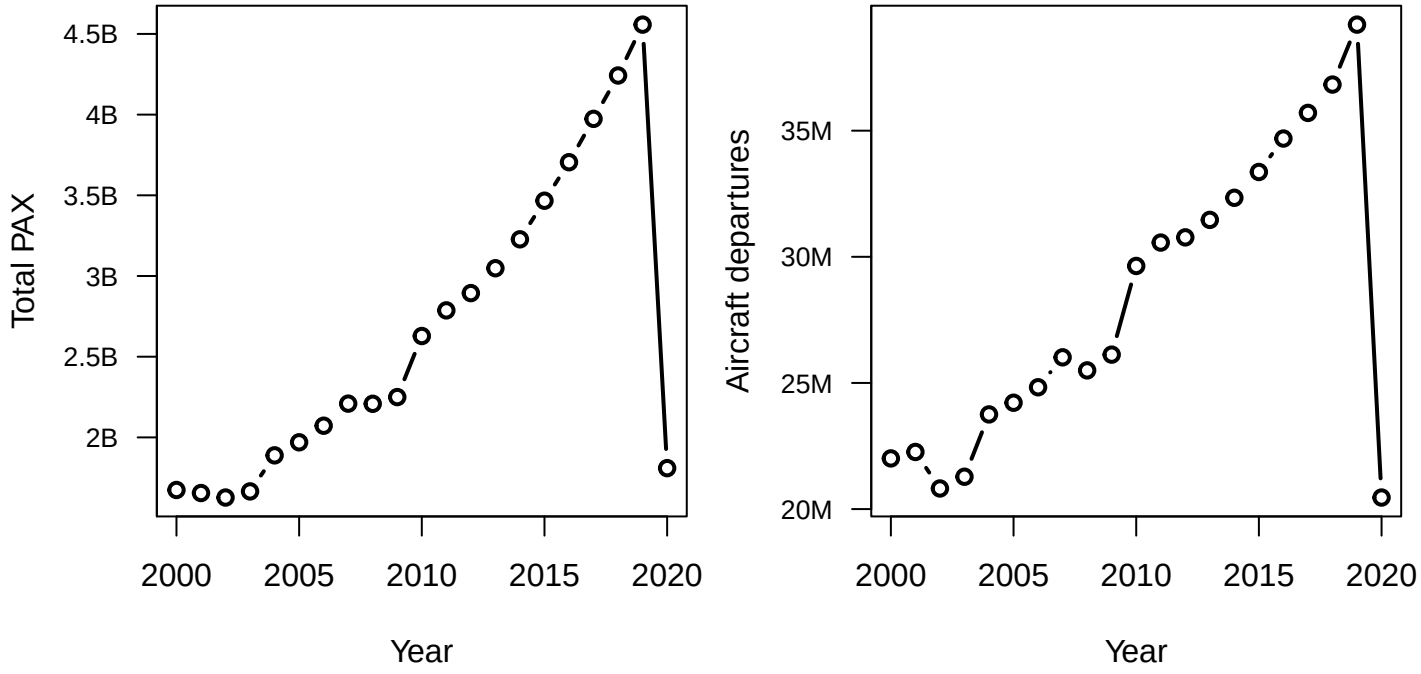


Figure 10: Evolution of the number of incoming tourists worldwide.



(a) Passengers transported

(b) Aircraft movements

Figure 11: Evolution of the annual number of passengers transported by air transport and of the number of aircraft movements.

this. One of them is using so-called *metapopulation* models.

I give here a very short introduction to the topic. Refer to other work on the subject for a more detailed presentation; for instance, see (Arino 2017) and the references therein.

4.2.1 Quick introduction to metapopulation models

Metapopulations split space into $|\mathcal{P}|$ geographical locations called *patches* and are thus appropriate for the description of phenomena involving discrete regions rather than continuous space. Each patch contains *compartments*, relatively homogeneous groups of individuals, e.g., susceptible humans, infected humans, etc. Individuals in a compartment *may* move between patches; m_{cqp} is rate of movement of individuals from compartment $c \in \mathcal{C}$ from patch $p \in \mathcal{P}$ to patch $q \in \mathcal{P}$. Each patch is equipped with a system describing the evolution of the number of individuals in each compartment present. For epidemic models, the general form is as follows. Assume *uninfected* (s) and *infected* (i) compartments in sets $\mathcal{U}$ and $\mathcal{I}$, respectively, with $\mathcal{U} \cup \mathcal{I} = \mathcal{C}$. For all $k \in \mathcal{U}$, $\ell \in \mathcal{I}$ and $p \in \mathcal{P}$,

$$s'_{kp} = f_{kp}(S_p, I_p) + \sum_{q \in \mathcal{P}} m_{kpq} s_{kq} \tag{1a}$$

$$i'_{\ell p} = g_{\ell p}(S_p, I_p) + \sum_{q \in \mathcal{P}} m_{\ell pq} i_{\ell q}, \tag{1b}$$

where $S_p = (s_{1p}, \ldots, s_{|\mathcal{U}|p})$ and $I_p = (i_{1p}, \ldots, i_{|\mathcal{I}|p})$ are the discrete distributions of individuals in the different compartments in patch $p \in \mathcal{P}$. The functions f and g describe the interactions between compartments in a given patch, while the sums describe the movement of compartments between locations and are written compactly by assuming that

$$m_{cpp} = -\sum_{q \in \mathcal{P}} m_{cqp}, \quad \forall c \in \mathcal{U} \cup \mathcal{I}, \tag{2}$$

i.e., by denoting m_{cpp} the rate of movement out of patch $p \in \mathcal{P}$ for individuals from compartment $c \in \mathcal{U} \cup \mathcal{I}$.

5 An SLIAR model for five countries

Let me consider the example of influenza, for which a lot of data is available online. A basic model for influenza is the SLIAR model (Arino et al. 2006).

Before proceeding further, remark that it is important to understand why a model is being used, as this determines the nature of the modelling framework used (ODE, PDE, Markov chains) and the type of model that is formulated. Here, I want to provide a reasonably realistic simulation context in which the spread of influenza between countries or regions in a country is modelled over *one* season. The spatial context calls for a metapopulation framework, while modelling influenza over one season means that an SLIAR model without demography is appropriate.

5.1 Base model in each patch – $SLIAR$ without demography

When formulating a metapopulation model, it is good to start by clearly establishing the model that is used in each patch. Here, the population in each patch is divided into five compartments as a function of the epidemic status of individuals. Susceptible individuals (S) might be infected by the disease; upon infection, individuals go into a phase where they are incubating with the disease (L); after the incubation period finishes, individuals can become either symptomatically (I) or asymptomatically (A) infectious to

others. Finally, after recovery, individuals are immune to reinfection with the strain of influenza they were infected with (R). In terms of the notation of (1), here $\mathcal{U} = \{S, R\}$ and $\mathcal{I} = \{L, I, A\}$. The flow diagram of the model is shown in Figure 12.

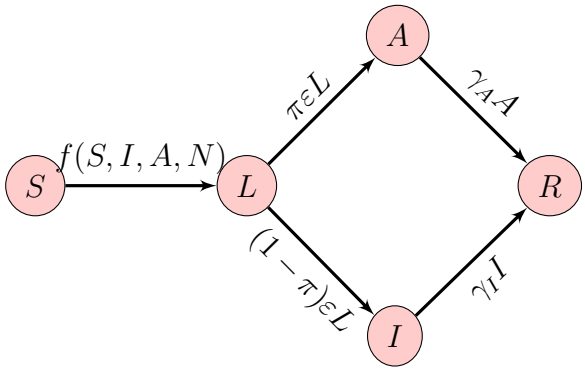


Figure 12: Flow diagram of the SLIAR model used in each patch.

Parameters are as follows: infection occurs at the rate $f(S, I, A, N)$, where $N = S + L + I + A + R$; the incubation period lasts on average $1/\varepsilon$ time units; the proportion of individuals becoming asymptomatically infectious is π; finally, the infectious period lasts on average $1/\gamma_I$ and $1/\gamma_A$ time units for symptomatically and asymptomatically infectious individuals, respectively. I do not consider disease-induced death here. As a consequence, the SLIAR model takes the form

$$S' = -f(S, I, A, N) \tag{3a}$$
$$L' = f(S, I, A, N) - \varepsilon L \tag{3b}$$
$$I' = (1 - \pi)\varepsilon L - \gamma_I I \tag{3c}$$
$$A' = \pi\varepsilon L - \gamma_A A \tag{3d}$$
$$R' = \gamma_I I + \gamma_A A. \tag{3e}$$

Note that at present, the form of the incidence function f has not been specified. What form to use depends on the aim of the model (McCallum, Barlow, and Hone 2001). In the following, $\eta \geq 0$ is a multiplicative factor indicating the change in infectiousness due to being an asymptomatic case. Typically, it is assumed to be in $[0, 1]$.

- Mass action, $f(S, I, A, N) = \beta S(I + \eta A)$, is the easiest mathematically. This form also seems to work well for epidemics.
- Standard incidence, $f(S, I, A, N) = \beta(I + \eta A)S/N$, is better suited for endemic situations or with demography present.
- More elaborate, better for fitting (if sufficient epidemic data available)

  $$f(S, I, A, N) = \beta(I^r + \eta A^s)S^p/N^q,$$

  where r, s, p and q are fitting parameters.

Here, for simplicity, I use mass action incidence. With the model in each patch established, let me now turn to the metapopulation model. This is simple: indices are added to all variables and parameters to indicate what patch is being considered; movement terms are added in order to allow individuals to move

between patches. For simplicity, I assume here that movement rates are independent of disease status, so that for all $p, q \in \mathcal{P}$,

$$m_{pq} := m_{Spq} = m_{Lpq} = m_{Ipq} = m_{Apq} = m_{Rpq}.$$

The rates m_{pp} are defined using (2), dropping the first index because of the assumption that movement rates are independent of disease status.

The resulting $|\mathcal{P}|$-SLIAR model then takes the form, for $p \in \mathcal{P}$,

$$\begin{aligned}
S_p' &= -\beta_p S_p I_p + \textstyle\sum_{q\in\mathcal{P}} m_{pq} S_q && \text{(4a)}\\
L_p' &= \beta_p S_p I_p - \varepsilon_p L_p + \textstyle\sum_{q\in\mathcal{P}} m_{pq} L_q && \text{(4b)}\\
I_p' &= (1-\pi_p)\varepsilon_p L_p - \gamma_{Ip} I_p + \textstyle\sum_{q\in\mathcal{P}} m_{pq} I_q && \text{(4c)}\\
A_p' &= \pi_p \varepsilon_p L_p - \gamma_{Ap} A_p + \textstyle\sum_{q\in\mathcal{P}} m_{pq} A_q && \text{(4d)}\\
R_p' &= \gamma_{Ip} I_p + \gamma_{Ap} A_p + \textstyle\sum_{q\in\mathcal{P}} m_{pq} R_q. && \text{(4e)}
\end{aligned}$$

Initial conditions are taken with $S_p(0) > 0$ for all $p \in \mathcal{P}$ and $\exists q \in \mathcal{P}$ such that $I_q(0) + A_q(0) > 0$ (otherwise the model is trivial); all others initial conditions are nonnegative.

5.2 Mathematical analysis

The focus of this paper is on running numerical simulations integrating data acquired from the Internet. However, it is always a good idea to conduct at least a local stability analysis of the model one is going to simulate, since this allows to get a sense of what the model can be expected to do. It is also useful to set what I have referred to earlier as *imputed parameters* or to get a sense of the range of values one should identify parameters in.

5.2.1 Behaviour when movement is absent

Model (4) is a Kermack-McKendrick-type model, so we can expect from (Arino et al. 2007) that $I_p \to 0$ and $S_p \to S_{p\infty}$ as $t \to \infty$ for all $p \in \mathcal{P}$. Let us confirm this. In patch $p \in \mathcal{P}$, the model in the absence of movement is given by (3) with indices, i.e.,

$$\begin{aligned}
S_p' &= -\beta_p S_p I_p && \text{(5a)}\\
L_p' &= \beta_p S_p I_p - \varepsilon_p L_p && \text{(5b)}\\
I_p' &= (1-\pi_p)\varepsilon_p L_p - \gamma_{Ip} I_p && \text{(5c)}\\
A_p' &= \pi_p \varepsilon_p L_p - \gamma_{Ap} A_p && \text{(5d)}\\
R_p' &= \gamma_{Ip} I_p + \gamma_{Ap} A_p && \text{(5e)}
\end{aligned}$$

A thorough analysis of (5) was conducted in (Arino et al. 2006), let me summarize it here. As often in epidemic models, one first seeks a disease-free equilibrium (DFE). This is obtained by setting $I_p = 0$. Clearly, $I_p = 0 \implies L_p = 0 \implies A_p = 0$, so the DFE has $(S_p, R_p) = (S_{p\infty}, R_{p\infty})$, $(L_p, I_p, A_p) = (0, 0, 0)$. Also, note that $N_p' = (S_p + L_p + I_p + A_p + R_p)' = 0$, which implies that $R_{p\infty} = N_p(0) - S_{p\infty}$. The number $S_p(0) - S_{p\infty}$ is the *final size* of the epidemic; it is typically expressed in the epidemiology literature in terms of the *attack rate* of the epidemic by considering the ratio $S_{p\infty}/S_p(0)$ (expressed as a percentage).

From (Arino et al. 2006), the *basic reproduction number* in isolated patches is given by

$$\mathcal{R}_0^p = S(0)\beta\left(\frac{1-\pi_p}{\gamma_{Ip}} + \frac{\pi_p\eta_p}{\gamma_{Ap}}\right). \tag{6}$$

This is a useful quantity to have, as it can be used to compute parameters so that $\mathcal{R}_0$ is known for patches in isolation.

5.2.2 Behaviour when movement is present

Let me return to the full system (4). In metapopulation models, problems are much easier to deal with when the system is written in vector form (Arino 2017). Here, we have

$$\mathbf{S}' = -\beta \circ \mathbf{S} \circ \mathbf{I} + \mathcal{M}\mathbf{S} \tag{7a}$$
$$\mathbf{L}' = \beta \circ \mathbf{S} \circ \mathbf{I} - \varepsilon\mathbf{L} + \mathcal{M}\mathbf{L} \tag{7b}$$
$$\mathbf{I}' = (\mathbb{I}-\pi)\varepsilon\mathbf{L} - \gamma_I\mathbf{I} + \mathcal{M}\mathbf{I} \tag{7c}$$
$$\mathbf{A}' = \pi\varepsilon\mathbf{L} - \gamma_A\mathbf{A} + \mathcal{M}\mathbf{A} \tag{7d}$$
$$\mathbf{R}' = \gamma_I\mathbf{I} + \gamma_A\mathbf{A} + \mathcal{M}\mathbf{R}, \tag{7e}$$

where $\circ$ denotes the Hadamard product. Note that the vector form is also useful when simulating the system; see Section 5.3.1. In (7), $\mathbf{S}$, $\mathbf{L}$, $\mathbf{I}$, $\mathbf{A}$, $\mathbf{R}$, β, ε, γ_I and γ_A are vectors with $|\mathcal{P}|$ entries, $\pi = \mathsf{diag}(\pi_1, \ldots, \pi_{|\mathcal{P}|})$ is diagonal, $\mathbb{I}$ is the identity matrix and the *movement matrix* is given by

$$\mathcal{M} = \begin{pmatrix} -\sum\limits_{p\in\mathcal{P}\setminus\{1\}} m_{p1} & m_{12} & \cdots & m_{1|\mathcal{P}|} \\ m_{21} & -\sum\limits_{p\in\mathcal{P}\setminus\{2\}} m_{p2} & \cdots & m_{2|\mathcal{P}|} \\ & & & \\ m_{|\mathcal{P}|1} & m_{|\mathcal{P}|2} & & -\sum\limits_{p\in\mathcal{P}\setminus\{|\mathcal{P}|\}} m_{p|\mathcal{P}|} \end{pmatrix}. \tag{8}$$

Note the negative terms on the diagonal; they are the outbound movement rates given by (2). Properties of (8) are important in the analysis of metapopulation models. Refer to (Arino, Bajeux, and Kirkland 2019) for a list of these properties.

Without going into details here, working with a large system of ordinary differential equations such as (7) is not much different from working with the system in a single patch (5). As we did there, we start by looking for the DFE. Set $\mathbf{I} = \mathbf{0}$. Then since $(\mathbb{I}-\pi)\varepsilon$ is invertible (we have assumed that $\pi_p \in (0,1)$ for all $p \in \mathcal{P}$), $\mathbf{L} = \mathbf{0}$. Substituting this into (7d) gives in turn $\mathbf{A} = \mathbf{0}$.

So the DFE satisfies $\mathbf{L} = \mathbf{I} = \mathbf{A} = \mathbf{0}$ and

$$\mathcal{M}\mathbf{S} = \mathcal{M}\mathbf{R} = \mathbf{0}.$$

$\mathcal{M}$ is clearly a singular matrix, since all its columns sum to zero. This implies that $\mathcal{M}\mathbf{S} = \mathcal{M}\mathbf{R} = \mathbf{0}$ have nonzero solutions $\mathbf{S}^\star$ and $\mathbf{R}^\star$. Also note that, summing equations in (7), we find $\mathbf{N}' = \mathcal{M}\mathbf{N}$. This means that the total population in the system, $\mathbb{1}^T\mathbf{N} = \langle\mathbb{1}, \mathbf{N}\rangle$, is constant, since

$$\frac{d}{dt}\langle\mathbb{1}, \mathbf{N}\rangle = \langle\mathbb{1}, \frac{d}{dt}\mathbf{N}\rangle = \langle\mathbb{1}, \mathbf{0}\rangle = \mathbf{0},$$

where $\mathbb{1}^T = (1, \ldots, 1)$. As a consequence, we can proceed as in (Arino and Driessche 2003) and use Cramer's rule to solve the augmented system

$$\begin{pmatrix} \mathbb{1} \\ \mathcal{M} \end{pmatrix} \mathbf{N}^\star = \begin{pmatrix} \mathbf{N}(0) \\ \mathbf{0} \end{pmatrix}. \tag{9}$$

A sufficient condition for $\mathbf{N}^\star \gg \mathbf{0}$, i.e., to be entry-wise positive, is that $\mathcal{M}$ be irreducible, that is, that the graph of patches be strongly connected. We make this assumption and thus know there exists a unique $\mathbf{N}^\star$ that solves the system (9). Note that this solution is a function of $\mathbf{N}(0)$ and as a consequence, so are $\mathbf{S}^\star + \mathbf{R}^\star = \mathbf{N}^\star$. It is probably feasible to use the method in (Arino et al. 2007) to compute these distributions more precisely, but this is beyond the scope of the present work.

5.3 Computational analysis

5.3.1 Define the vector field

Here, I adapt the code in (Arino 2017) to the SLIAR case. As is customary when solving ODEs numerically, I first need to define the vector field. This is done in the following function. Note that I have defined a set of indices for the different epidemic stages, in order to quickly look-up the corresponding entries in the state variables vector `x`. This is not technically required but makes the vector field easier to read.

```
SLIAR_metapop_rhs <- function(t, x, p) {
    with(as.list(x), {
        S <- x[p$idx_S]
        L <- x[p$idx_L]
        I <- x[p$idx_I]
        A <- x[p$idx_A]
        R <- x[p$idx_R]
        Phi <- p$beta * S * (I + p$eta * A)
        dS <- -Phi + p$M %*% S
        dL <- Phi - p$epsilon * L + p$M %*% L
        dI <- (1 - p$pi) * p$epsilon * L - p$gammaI * I + p$M %*%
            I
        dA <- p$pi * p$epsilon * L - p$gammaA * A + p$M %*% A
        dR <- p$gammaI * I + p$gammaA * A + p$M %*% R
        list(c(dS, dL, dI, dA, dR))
    })
}
```

Recall that in `R`, `*` denotes the Hadamard (entry-wise) product of vectors, which I denoted $\circ$ in (7), while `%*%` denotes the usual matrix product. Thus, by writing the system in vector form, numerical integration is not much more complicated that if I were considering a single-population equivalent.

5.3.2 Setting up parameters

This first example is based on the one given in (Arino 2017). I consider five countries: Canada, China, India, Pakistan and the Philippines. The total populations of these countries are known; for instance, I obtain below the most up to date estimates from the World Bank. Also known through other means is the average number of air passengers travelling between each of these countries on a given day (estimates are from 2015).

```
countries <- c("Canada", "China", "India", "Pakistan", "Philippines")
countries_iso3c <- countrycode(countries, origin = "country.name",
    destination = "iso3c")
if (DOWNLOAD) {
    pop_data_5ctr_wb <- wb(country = countries_iso3c, indicator = "SP.POP.TOTL")
}
pop_data_5ctr_wb <- latest_values_general(pop_data_5ctr_wb, "iso3c",
    "date")
T <- matrix(data = c(0, 1268, 900, 489, 200, 1274, 0, 678, 859,
    150, 985, 703, 0, 148, 58, 515, 893, 144, 0, 9, 209, 174,
    90, 2, 0), nrow = 5, ncol = 5, byrow = TRUE)
```

The function `latest_values_general` used above is part of the file `useful_functions.R` and is used to keep only the most recent value in a data frame. I now need to set up the movement matrix $\mathcal{M}$. Here, I proceed in one of two ways that will be presented in these lecture notes, which is based on a method explained in (Arino and Portet 2015). Suppose X and Y are two locations connected by mobility, with the population of X given by N_X. We seek the *per capita* rate m_{YX} of movement from X to Y. If one considers a short enough time interval, then one can assume that the rate of change of the population of X due to travel to Y is governed by $N_X' = -m_{YX}N_X$. Integrating this simple linear differential equation and solving for m_{YX} when $t = 1$ gives

$$m_{YX} = -\ln\left(1 - \frac{T_{YX}}{N_X}\right), \tag{10}$$

where T_{YX} is the number of individuals having moved from X to Y in 1 time unit.

```
p <- list()  # Use a list to store all parameters
p$M <- mat.or.vec(nr = dim(T)[1], nc = dim(T)[2])
for (from in 1:5) {
    for (to in 1:5) {
        p$M[to, from] <- -log(1 - T[from, to]/pop_data_5ctr_wb$value[from])
    }
    p$M[from, from] <- 0
}
p$M <- p$M - diag(colSums(p$M))
```

I now set up the remainder of the parameters. As indicated earlier, it is useful to keep track of indices of the different compartments.

```
p$P <- dim(p$M)[1]
p$idx_S <- 1:p$P
p$idx_L <- (p$P + 1):(2 * p$P)
p$idx_I <- (2 * p$P + 1):(3 * p$P)
p$idx_A <- (3 * p$P + 1):(4 * p$P)
p$idx_R <- (4 * p$P + 1):(5 * p$P)
```

Disease-related parameters are then set. Note that values are here chosen arbitrarily, just for illustration. They are meant to roughly mimic parameters that would be used for influenza.

```
p$eta <- rep(0.3, p$P)
p$epsilon <- rep((1/2), p$P)
```

```
p$pi <- rep(0.7, p$P)
p$gammaI <- rep((1/4), p$P)
p$gammaA <- rep((1/3), p$P)
```

I now set up the initial conditions. As an example, suppose that there are initially two infected individuals in Canada.

```
L0 <- mat.or.vec(p$P, 1)
I0 <- mat.or.vec(p$P, 1)
A0 <- mat.or.vec(p$P, 1)
R0 <- mat.or.vec(p$P, 1)
I0[which(countries == "Canada")] <- 2
S0 <- pop_data_5ctr_wb$value - (L0 + I0 + A0 + R0)
IC <- c(S = S0, L = L0, I = I0, A = A0, R = R0)
```

Although parameters are chosen arbitrarily, to avoid numerical issues it is useful to have some control over parameter values. Here, for instance, I set up the contact parameter β in such a way as to avoid blow up of solutions. Solving $\mathcal{R}_0^p$ for patches in isolation as given by (6) in terms β, I obtain

$$\beta = \frac{\mathcal{R}_0^p}{S_p(0)} \left(\frac{1-\pi_p}{\gamma_{Ip}} + \frac{\pi_p \eta_p}{\gamma_{Ap}} \right)^{-1}.$$

Suppose for instance that $\mathcal{R}_0 = 1.5$ for patches in isolation; this is incorporated into the model by assuming that

```
p$beta <- 1.5/S0 * 1/((1 - p$pi)/p$gammaI + p$pi * p$eta/p$gammaA)
```

The final step is to set up the time span of the simulation, one year here.

```
tspan <- seq(from = 0, to = 365.25)
```

One important remark concerning the vector of times is that it can be tailored to match existing data points. I will return to this with the next example in Section 6. Finally, I call the solver,

```
sol <- ode(y = IC, times = tspan, func = SLIAR_metapop_rhs, parms = p,
    method = "ode23", rtol = 1e-08, atol = 1e-08, maxsteps = 15000)
```

and put the result in a form that is easier to use. Note that in order to overcome numerical issues, I have changed the method used to solve the system from the default `lsoda` to `ode23`, as well as changed tolerances and the maximum number of steps. The issues arising here are linked to the difference in orders of magnitudes of the different quantities involved. For instance, population counts are in millions, whereas values of β_p so that $\mathcal{R}_0^p$ equals 1.5 are of the order of 10^{-8}.

```
times <- sol[, "time"]
S <- sol[, p$idx_S + 1]
L <- sol[, p$idx_L + 1]
I <- sol[, p$idx_I + 1]
A <- sol[, p$idx_A + 1]
R <- sol[, p$idx_R + 1]
N <- S + L + I + A + R
```

In the code above, I have to shift indices of the positions of the different variables by 1, since the first column in the result matrix contains `time`. This is different from the vector field itself, where `x` is the vector of state variables only.

Just for illustration, Figure 13 shows the results. As is often the case with epidemiological data, I show results per 100,000 people rather than actual numbers. (Code chunk not shown.)

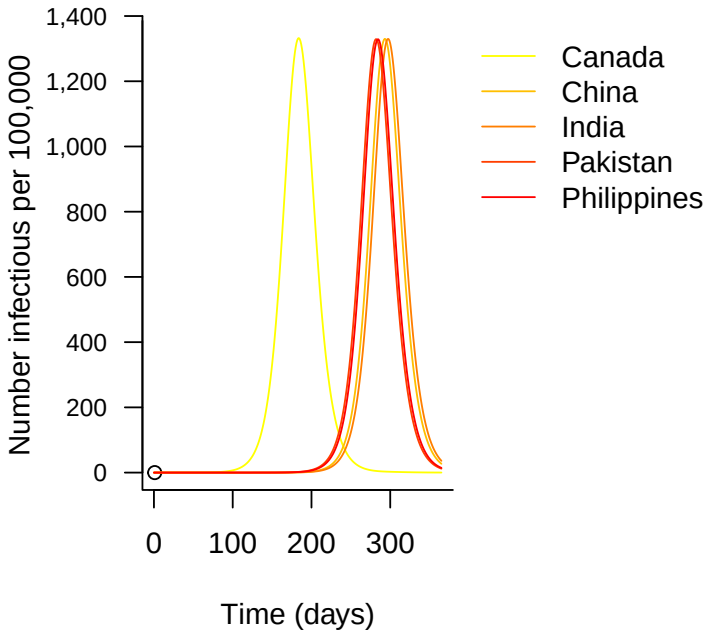


Figure 13: Time evolution of the number of cases of influenza in 5 countries.

6 Spread of influenza between regions in a country

In Section 5, I considered the spread of influenza between five countries. I had air travel mobility numbers and could assume that, to a large extent, the numbers represented a large fraction of the actual means of movement, due to the distances between most of the countries involved. I did not, however, have access to precise epidemiological surveillance of influenza for all these countries and simply simulated the spread.

Here, I consider a somewhat converse problem, where the disease epidemiology is much better known, but mobility patterns need to be inferred. Staying with influenza, I look at its spread between regions of metropolitan France. Influenza data is obtained from the Réseau Sentinelles, which has been conducting practitioner-based surveillance of influenza since 1984 (Valleron et al. 1986). Also known is a rough measure of vaccine uptake in each region; the model will be (slightly) adapted to incorporate this vaccination information.

Influenza data is available online by regions in metropolitan France. Regions in France were redefined in 2016. Until then, there were 22 regions in metropolitan France (Corsica being considered part of metropolitan France). In 2016, some regions were aggregated and the number was reduced to 13. Data can be downloaded in terms of the new regions, or in terms of past regions until 2016 and of new regions since. This illustrates a common problem when dealing with data: adapting the new data to the past regions would require to infer how the data was processed. As a consequence, I use the new definition of regions.

Also available is the regional uptake of influenza vaccine. Of course, the population of regions and their geography is readily available. I now download all of the required data.

```
if (DOWNLOAD) {
    page_url <- "Ranked_list_of_French_regions"
    pop_regions_FR <- htmltab(paste0(wiki_url, page_url), which = 1)
    coverage_influenza_FR <- htmltab(SPF_url, which = 1)
}
centroids_regions_FR <- read.csv("DATA/centroids_regions_FR.csv")
centroids_regions_FR$region <- as.character(centroids_regions_FR$region)
Encoding(centroids_regions_FR$region) <- "UTF-8"
Encoding(pop_regions_FR$Region) <- "UTF-8"
colnames(coverage_influenza_FR)[1] <- "region"
Encoding(coverage_influenza_FR$region) <- "UTF-8"
```

Note that here, I had to impose encoding of the downloaded data because of accents in French text. Interestingly, this issue does not arise under Linux but does in Windows.

6.1 Setting up a gravity matrix

In the 5 countries example of Section 5, I had access to population mobility data that, because of the distances between some of the countries, can be assumed to be a relatively good approximation of the actual travel volume. Another example where data sums up most of the actual mobility is the work in (Arino and Portet 2015), where estimates of the number of travellers, by road this time, between a large city and a few neighbouring smaller ones are obtained and used. In the present case, though, there are two levels of complication.

1. Mobility in a country like France is multi-modal, with the main modes of mobility being air, rail and road travel.
2. Even if numbers were available for all travel modalities, they would be hard to combine.

Given this complexity, in order to approximate the volume of travel between locations using non-proprietary data, I compute a *gravity matrix*. The idea is to use the analogy of gravitation. The gravitational force F_{ij} between two bodies with masses m_i and m_j and centres of mass r distance units apart is given by

$$F_{ij} = G\frac{m_i m_j}{r^2},$$

where G is the gravitational constant. Here, I use this principle with centroids (centres of gravity) of regions as centres of mass and the population of regions as their mass. The result is a symmetric matrix with zeros on the main diagonal and the gravitational constant G as a scalar factor. That constant is later used to set values in the movement matrix so that movement rates are commensurate with the problem under consideration.

As has become the *leitmotiv* throughout this paper, first I create a data frame with all relevant information. As region centroids are given in (`degrees,minutes,seconds`), they must be transformed to decimal degrees.

```
tmp <- conv_unit(centroids_regions_FR$cent_lon, from = "deg_min_sec",
    to = "dec_deg")
centroids_regions_FR$cent_lon <- as.numeric(tmp)
tmp <- conv_unit(centroids_regions_FR$cent_lat, from = "deg_min_sec",
    to = "dec_deg")
centroids_regions_FR$cent_lat <- as.numeric(tmp)
```

```
colnames(pop_regions_FR)[3] <- "pop"
pop_regions_FR$pop <- as.numeric(gsub(",", "", pop_regions_FR$pop))
pop_regions_FR$Region <- stringr::str_trim(pop_regions_FR$Region)
regions_FR <- merge(x = pop_regions_FR, y = centroids_regions_FR,
    by.x = "Region", by.y = "region")
regions_FR <- regions_FR[, c(1, 3, 5, 6, 7)]
colnames(regions_FR)[1] <- "region"
my_palette <- colorRampPalette(c("yellow", "red"))(n = dim(regions_FR)[1])
```

I can now compute the distances between region centroids.

```
dist_matrix <- mat.or.vec(nr = dim(regions_FR)[1], nc = dim(regions_FR)[1])
for (i in 1:dim(regions_FR)[1]) {
    for (j in 1:dim(regions_FR)[1]) {
        dist_matrix[i, j] <- distm(c(regions_FR$cent_lon[i],
            regions_FR$cent_lat[i]), c(regions_FR$cent_lon[j],
            regions_FR$cent_lat[j]), fun = distVincentyEllipsoid)
    }
}
```

Note that I could also have used some internal `R` functions, such as `dist`, to compute the distances. Once this has been done, I compute the gravity between all pairs of regions. Note that if there were many more pairs to deal with, exploiting the symmetry of the gravity matrix would allow to gain some time.

```
grav_matrix <- dist_matrix * 0
for (i in 1:dim(regions_FR)[1]) {
    for (j in 1:dim(regions_FR)[1]) {
        grav_matrix[i, j] <- regions_FR$pop[i] * regions_FR$pop[j]
    }
}
grav_matrix <- grav_matrix/(dist_matrix^2)
grav_matrix[is.infinite(grav_matrix)] <- 0
```

Note that the last instruction is used because in the computation of distances, I did not exclude diagonal entries and thus am dividing by zero along the diagonal. So, finally, I have the gravity matrix, where region names are abbreviated for convenience.

```
colnames(grav_matrix) <- region_names_FR$shortFR
rownames(grav_matrix) <- region_names_FR$shortFR
round(grav_matrix)
```

```
     ARA BFC  Br   Ce Co   GE  HdF  IdF   No  Na  Oc  PL PACA
ARA    0 604  65  213 10  335  181  640   97 402 598 137  894
BFC  604   0  27  129  2  525  148  605   57  94  83  58   99
Br    65  27   0   71  1   47   96  251  190 120  48 428   24
Ce   213 129  71    0  1  138  187 1395  183 197  84 269   48
Co    10   2   1    1  0    3    2    5    1   3   5   1   16
GE   335 525  47  138  3    0  531 1281  112 103  85  84  100
HdF  181 148  96  187  2  531    0 3652  434 112  72 153   59
```

```
IdF  640 605 251 1395  5 1281 3652    0 1225 392 227 564  172
No    97  57 190  183  1  112  434 1225    0 104  50 326   31
Na   402  94 120  197  3  103  112  392  104   0 673 314  127
Oc   598  83  48   84  5   85   72  227   50 673   0  95  292
PL   137  58 428  269  1   84  153  564  326 314  95   0   43
PACA 894  99  24   48 16  100   59  172   31 127 292  43    0
```

Interestingly, the method returns realistic results: the fifth entry in the matrix is Corsica, which has a small population and is quite isolated, being an island. The unadjusted values here are too large and will be scaled down using G prior to being used in the numerical simulations.

6.2 Setting up initial conditions – Vaccination coverage

I could, as I did in Section 5, use as initial conditions the population $N_p(0)$ in each region $p \in \mathcal{P}$. Here, however, I want to incorporate the effect of vaccination. There are more elaborate methods to model vaccination, but a minimalist approach consists in assuming that vaccination simply reduces the susceptible population. Thus, I consider the initial susceptible population $\tilde{N}_p(0) = (1 - v_p)N_p(0)$, where v_p is fraction vaccinated in region $p \in \mathcal{P}$. Note that this means that I am assuming that the vaccine is 100% efficacious; that is far from true, but to keep the problem simple, I make this assumption.

In our recurrent theme, I need to process vaccination data a little in order to be able to use it in simulations. (Chunk not shown.)

Setting initial conditions then proceeds as in Section 5, except that the total population $N_p(0)$ reflects the percentage vaccinated. I start with several cases in, say, Île-de-France (the region comprising Paris). (Chunk not shown.)

For the movement matrix $\mathcal{M}$, I start with the gravity matrix I already computed, which I also store for future use. Here, I normalise by imposing that movement rates in the matrix should not be larger than 0.005, a somewhat arbitrary value I have often used in metapopulation models and gives usually reasonable results in terms of population movement.

```
p$grav_matrix <- grav_matrix
grav_matrix <- p$grav_matrix/max(p$grav_matrix) * 0.005
p$M <- grav_matrix - diag(colSums(grav_matrix))
```

If information about mobility volumes is available, one can "invert" the method used in Section 5 in order to obtain finer estimates for the value of G.

The following is then done (chunks not shown). Parameters are set as before. As I did earlier, I suppose that $\mathcal{R}_0 = 1.5$ for patches in isolation. This having been done, I carry out the numerical integration and plot the solution, giving Figure 14.

6.3 Using epidemiological data

Let me load the Réseau Sentinelles data and prepare it for use. Note that in the case of a service like Réseau Sentinelles, where data typically changes weekly and the equipment does not necessarily support very frequent queries, it is good practice to keep a local cache of the data and refresh only in case of change. So, when loading the data, I use a function called `nice_load` (found in the `useful_functions.R` electronic appendix) that handles this. Some more elaborate mechanisms are available, but I felt it was

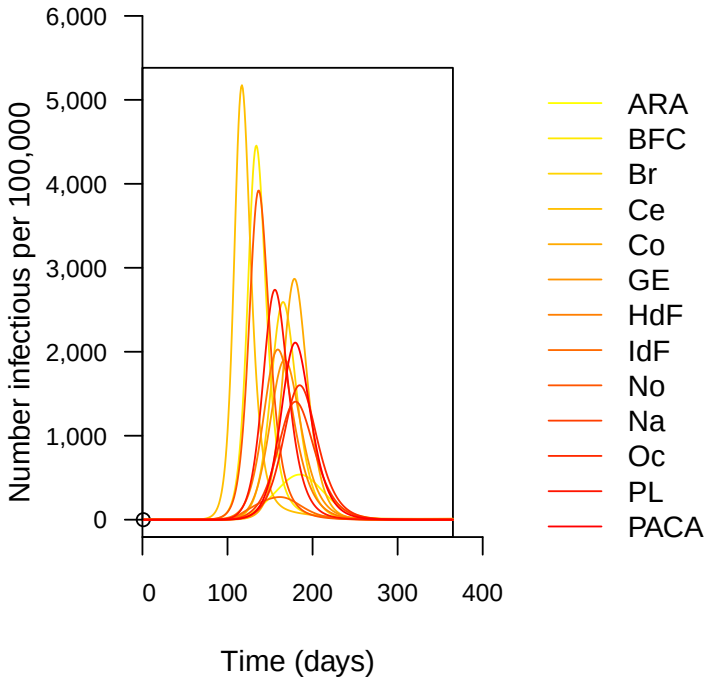


Figure 14: Number of cases of influenza per 100,000 inhabitants (simulation results).

worth illustrating how to do this here. The data has dates in the form YYYY-WW, where WW is the week number. I need to transform this into a regular date.

```
if (DOWNLOAD) {
    influenza_FR <- nice_load(file = "DATA/incidence-RDD-3.csv",
        web = "https://www.sentiweb.fr/datasets/incidence-RDD-3.csv",
        update_days = 30, skip = 1, stringsAsFactors = FALSE)
}
tmp_y <- substring(influenza_FR$week, first = 1, last = 4)
tmp_w <- substring(influenza_FR$week, first = 5, last = 6)
influenza_FR$date <- ISOweek::ISOweek2date(sprintf("%s-W%s-1",
    tmp_y, tmp_w))
```

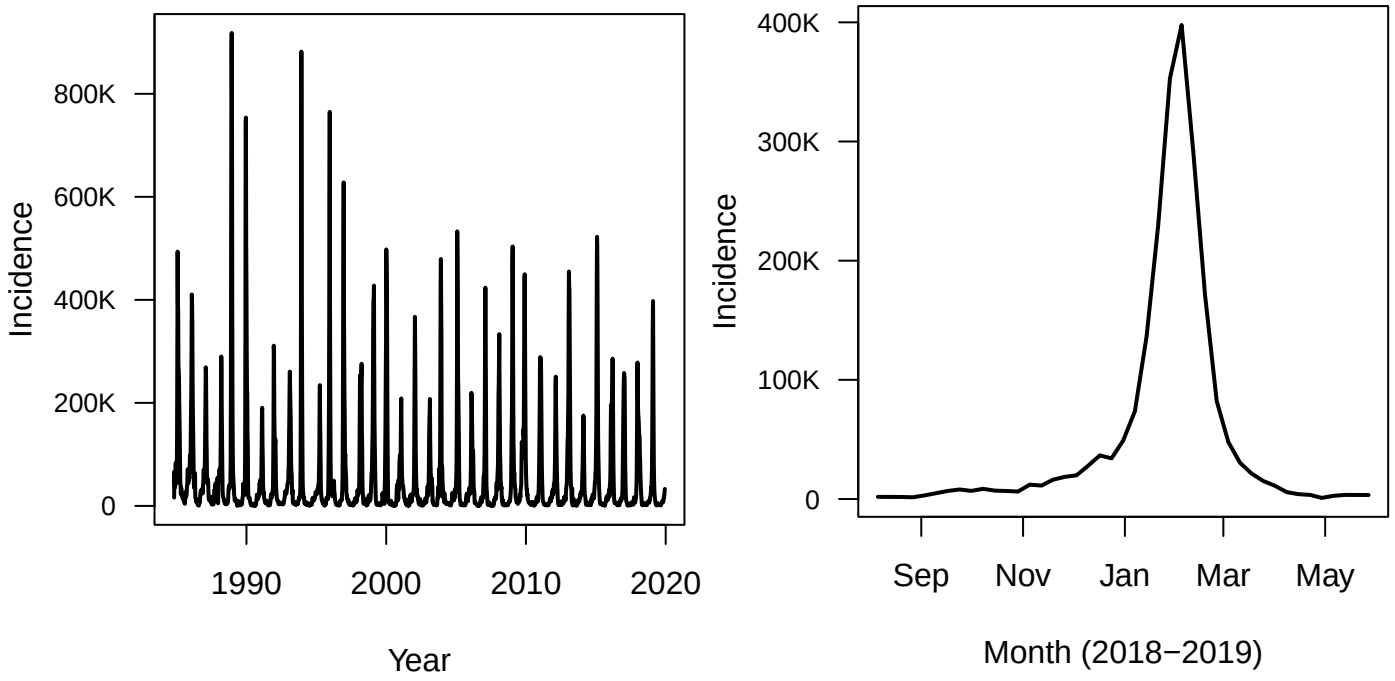
It is always a good idea to explore the data. At the very least, it should be plotted in order to get a sense of the general behaviour. Let me create a few variables that allow to access the data easily. (Chunk not shown.)

Let me take a look at the data. First, in Figure 15a, I show the total weekly number of cases over the entire country for the entire dataset.

Let me now select the latest influenza season at the time of writing. An end date in 2019 is specified because future uses of the code provided here would extend the plot to more than one season.

```
idx <- intersect(which(dates >= "2018-08-01"), which(dates <=
    "2019-06-01"))
```

I then plot, in Figure 15b, the number of cases during the 2018-2019 epidemic season for the entire country and in Figure 16, the breakdown of these numbers region by region (per 100,000 people, in this case, because of the wide variation of population between regions).



(a) All dates.

(b) 2018-2019.

Figure 15: Weekly number of cases of influenza in France as reported by the Réseau Sentinelles. (a) All dates. (b) 2018-2019 epidemic season.

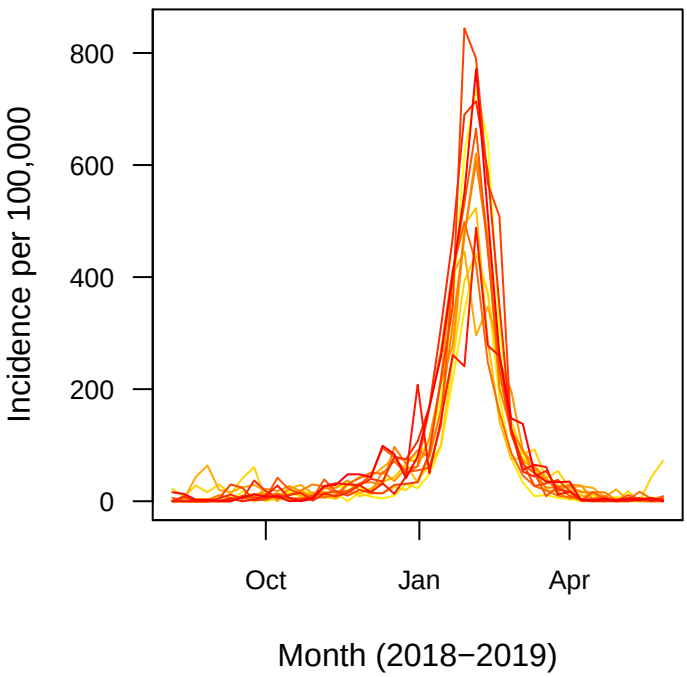


Figure 16: Weekly number of cases of influenza in each of the 13 regions of metropolitan France for the 2018-2019 epidemic season, as reported by the Réseau Sentinelles. Numbers are per 100,000.

6.4 Numerical simulation of the system

For initial conditions, I use actual incidence, not the number of cases per 100,000 that I plotted in Figure 16. For simplicity, I assume that all reported cases are symptomatically infectious. I further assume that, as in System (3), a fraction π_p of cases in patch $p \in \mathcal{P}$ are asymptomatically infectious, so that if there are initially $I_p(0)$ symptomatic cases, there are also $(1 - \pi_p)I_p(0)$ asymptomatic ones.

```
I0 <- mat.or.vec(nr = length(influenza_FR_region), nc = 1)
for (i in 1:length(influenza_FR_region)) {
    I0[i] <- influenza_FR_region[[i]]$inc[idx[1]]
    A0[i] <- (1 - p$pi[i]) * I0[i]
}
S0 <- N0 - (L0 + I0 + A0 + R0)
IC <- c(S = S0, L = L0, I = I0, A = A0, R = R0)
```

First, let me run a naïve simulation. Note that for time span and time points, I use the dates I selected. This feature, that was mentioned earlier, is useful when minimizing errors.

```
tspan <- as.numeric(dates[idx])
```

The remainder of the call (chunk not shown) then proceeds as previous instances; the result is shown in Figure 17.

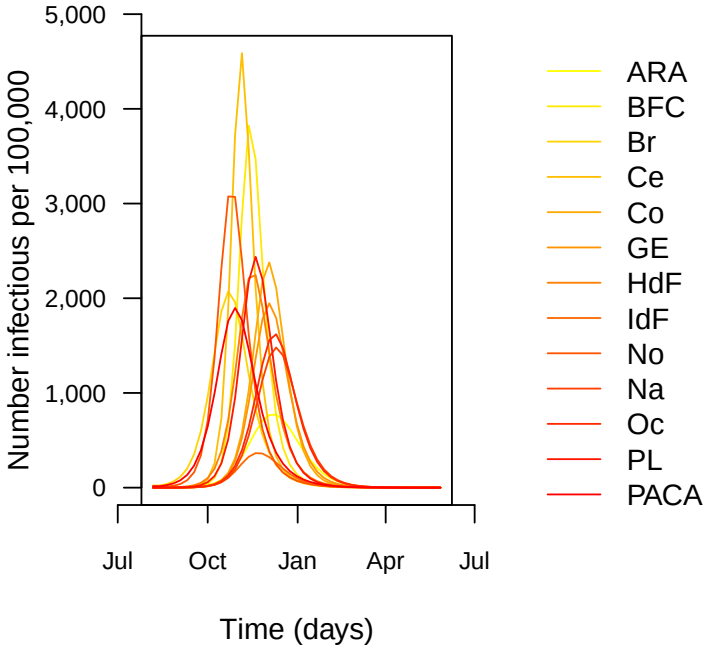


Figure 17: Simulation of 1 year of influenza propagation in France. Numbers are per 100,000.

Let me now carry out a naïve parameter identification routine. As mentioned in the Introduction, the procedure carried out here is very simple and much more information will be gained by following the techniques found in lecture notes from other authors. The results of the procedure are not good by any means and much more attention should be paid to the details in actual work.

The way I proceed is to allow two parameter types to vary: the individual value of β_p in the patches and the gravity constant G. First, I need a function similar to the one used in Section 3.1.2, which, given values of $(\beta_1, \ldots, \beta_{13}, G)$, returns an error, taken here as the sum of the square of the differences between data points

and the simulated solutions. The vector `v` passed as an argument to this function contains the parameters that change, while `param` is the remained of parameters.

```
error_simulation <- function(v, data.df, param, IC, startDate,
    endDate) {
    param$beta <- v[1:13]
    G <- v[14]
    grav_matrix <- grav_matrix/max(grav_matrix) * G
    param$M <- grav_matrix - diag(colSums(grav_matrix))
    idx <- intersect(which(dates >= startDate), which(dates <=
        endDate))
    tspan <- as.numeric(data.df$date[idx])
    sol <- ode(y = IC, times = tspan, func = SLIAR_metapop_rhs,
        parms = param, maxsteps = 10000)
    times <- sol[, "time"]
    S <- sol[, param$idx_S + 1]
    L <- sol[, param$idx_L + 1]
    I <- sol[, param$idx_I + 1]
    A <- sol[, param$idx_A + 1]
    R <- sol[, param$idx_R + 1]
    N <- S + L + I + A + R
    simResult <- (1 - p$pi) * p$epsilon * L/N
    error <- sum((simResult - data.df[idx, 2:14])^2)
    return(error)
}
```

Now that this function is ready, I can use a genetic algorithm to minimise the error. Genetic algorithms work by maximising the so-called fitness function, hence I give as argument the negative of the function I just defined. Note that since the data is incidence data, I need comparable model output. Because I assume that neither latently infected nor asymptomatic cases are detected, it follows that comparison is to the rate of apparition of new symptomatically infectious cases, $(1-\pi)\varepsilon L$.

```
library(parallel)
library(doParallel)
suggestedSol <- c(p$beta, 0.005)
GA <- ga(type = "real-valued", fitness = function(x) -error_simulation(x,
    influenza_FR_region100.df, p, IC, "2018-08-01", "2019-06-01"),
    suggestions = suggestedSol, lower = rep(0, 14), upper = rep(0.1,
        14), popSize = 50, maxiter = 100, monitor = FALSE, seed = 12345,
    parallel = TRUE)
```

Once the method has run, results can be accessed in `GA@solution`. Recall that I have used parameters as $(\beta_1, \ldots, \beta_{13}, G)$, so

```
p$beta <- GA@solution[1:13]
p$G <- GA@solution[14]
grav_matrix <- p$grav_matrix/max(p$grav_matrix) * p$G
p$M <- grav_matrix - diag(colSums(grav_matrix))
```

Once this is done, I solve the ODE numerically and plot the result (chunk not shown). The plot is shown in Figure 18. Obviously, the results are not fantastic. Much more could be done to obtain a better fit; I will refer to other lecture notes in this special issue for details. Two potential ways to address the limitations here are the following.

1. Switch to time units of weeks. Recall that data is the number of new cases detected in one week. The time unit in the simulation is the day, so that $(1-\pi)\varepsilon L$ is expressed per day. To decrease the distance between data and simulation, the genetic algorithm thus selects larger values of β, resulting in an earlier epidemic.
2. I could also play with the date of start of the simulation and make it a parameter to identify as well.

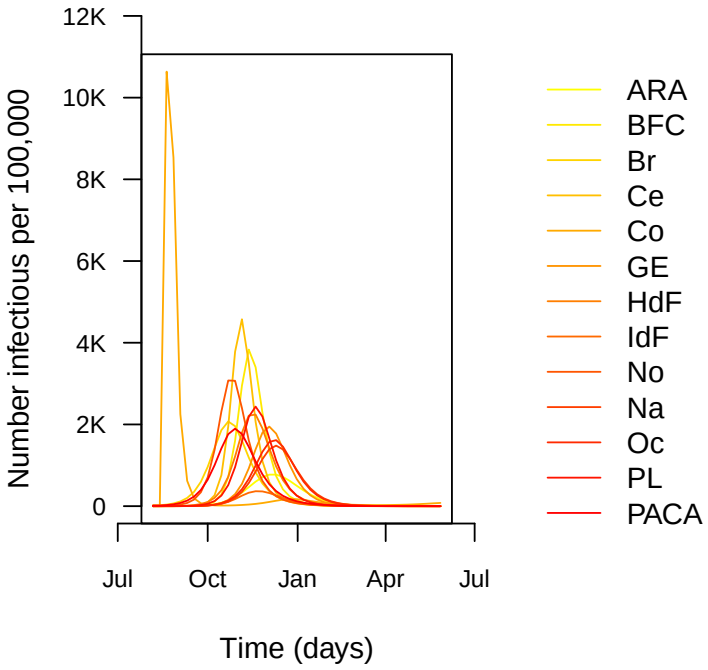


Figure 18: Weekly number of cases of influenza in each of the 13 regions of metropolitan France for the 2018-2019 epidemic season. Parameters obtained by genetic algorithm optimisation. Numbers are per 100,000.

Conclusion

To conclude, let me start by justifying in hindsight the use of the type of methods presented here, by giving a very brief overview of the path that led me toward this type of problems. My systematic use of data in epidemic models began with my work with the Bio.Diaspora Project, now the company Bluedot (link). In this context, we have a very rich database focused on human mobility; see some details in (Arino and Khan 2014). We use these data to understand global public health risks, with emphasis on those aspects of risk linked to mobility and, in particular, air travel. Questions that arise in this context are twofold. First, if an alert is generated by a disease surveillance system, what is the potential for the event being reported on to transform into a significant event with potential for regional or international spread? Second, if the event has potential for transborder spread, what are the most likely next locations it will affect? Clearly, considering these questions requires gathering data on a scale that cannot be realistically achieved without programmatic approaches. Hence, the move towards automated or, at the very least, semi-automated information retrieval. Automated report generation, on the other hand, is prompted by

the need to summarise the same *indicators* about different locations based on the location of an alert.

The methods presented here should be seen as a component in modelling work. Data can help modellers get a better sense of the context they are operating in. Data can also help modellers conduct better numerical investigation of their model. However, data is not a substitute to proper modelling work, nor does it absolve the modeller from conducting some mathematical analysis. In any case, programmatic techniques of data acquisition such as the ones I illustrate in these notes should, in my view, become part of the arsenal of techniques mathematical epidemiologists are familiar with. Ideally, mathematical epidemiologists should also familiarise themselves with the concept of *reproducible research*, which is illustrated by the medium chosen for these notes. While the model formulation and mathematical analysis part of our work is, of course, inherently reproducible, the numerics or data components are not necessarily and adopting reproducible research ideas would be sometimes helpful.

Acknowledgements

My work is partially funded by NSERC. Various funders supported the Bio.Diaspora Project, but the most important support for ideas presented here was a 2013-2015 CIHR/NSERC Collaborative Health Research Projects grant, *Protecting Canadian Health, Security, and Prosperity through the Novel Integration of Global Epidemic Surveillance and Risk Modelling Technologies*, that I held jointly with K. Khan (St Michael's Hospital & Bluedot, Toronto).

References

Arino, J. 2017. "Spatio-Temporal Spread of Infectious Pathogens of Humans." *Infectious Disease Modelling* 2 (2): 218–28.

Arino, J., N. Bajeux, and S. Kirkland. 2019. "Number of Source Patches Required for Population Persistence in a Source–Sink Metapopulation with Explicit Movement." *Bulletin of Mathematical Biology* 81 (6): 1916–42.

Arino, J., F. Brauer, P. van den Driessche, J. Watmough, and J. Wu. 2006. "Simple Models for Containment of a Pandemic." *Journal of the Royal Society Interface* 3 (8): 453–57.

———. 2007. "A Final Size Relation for Epidemic Models." *Mathematical Biosciences and Engineering* 4 (2): 159–75.

Arino, J., and P. van den Driessche. 2003. "A Multi-City Epidemic Model." *Mathematical Population Studies* 10 (3): 175–93.

Arino, J., and K. Khan. 2014. "Analyzing and Modeling Spatial and Temporal Dynamics of Infectious Diseases." In *Analyzing and Modeling Spatial and Temporal Dynamics of Infectious Diseases*, edited by D. Chen, B. Moulin, and J. Wu. Wiley.

Arino, J., and S. Portet. 2015. "Epidemiological Implications of Mobility Between a Large Urban Centre and Smaller Satellite Cities." *Journal of Mathematical Biology* 71 (5): 1243–65.

McCallum, H., N. Barlow, and J. Hone. 2001. "How Should Pathogen Transmission Be Modelled?" *Trends Ecol. Evol.* 16 (6): 295–300.

Thieme, H. R. 2003. *Mathematics in Population Biology*. Princeton Series in Theoretical and Computational Biology. Princeton University Press.

Valleron, A.-J., E. Bouvet, P. Garnerin, J. Ménares, I. Heard, S. Letrait, and J. Lefaucheux. 1986. "A Computer Network for the Surveillance of Communicable Diseases: The French Experiment." *American Journal of Public Health* 76 (11): 1289–92.

WHO Ebola Response Team. 2015. "West African Ebola Epidemic After One Year-Slowing but Not yet

Under Control." *New England Journal of Medicine* 372 (6): 584–87.

Julien Arino, Department of Mathematics & Data Science NEXUS, University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada

E-mail address: Julien.Arino@umanitoba.ca