



This assignment is **due before midnight** on **Friday 26 November 2020**.


---

THIS IS A DRAFT VERSION OF THE ASSIGNMENT. THE FINAL VERSION OF THE ASSIGNMENT WILL BE CIRCULATED AS SOON AS IT IS READY. – DE

---

The purpose of this assignment is for you to learn to use the **McMasterPandemic**  package, run simulations with the package, and fit parameters to a *simulated* epidemic (for which the underlying parameter values are known). Your submission should ideally be written using **knitr** (the alternative is  $\text{\LaTeX}$  and separate  scripts), and must be submitted as a tarball or zipped folder that can be run to reproduce your final pdf. Read this entire document, including all the [technical comments at the end](#) before starting to work on the assignment.

## 1 Check out the McMasterPandemic shiny

Zach Levine has written a [web interface to the McMasterPandemic package](#) (is it written using the [R shiny](#) package). Zach’s web interface provides a way to explore the simulation functionality of the package without installing it on your computer or knowing anything about .

Your first task is to be a tester of the web interface. Explore it, see what you can learn from it, and try to discover its limits (*i.e.*, what makes it “break”). Write a brief summary of what you think the point of it is, what you felt was missing as you explored, what was confusing or unclear to you, and any other comments that you think would be helpful for improving it. Please e-mail any questions to [both me and Zach using this link](#).

*Note:* Since Zach is taking the course, his task will instead be to implement some of the suggested enhancements.

## 2 Install McMasterPandemic package

**McMasterPandemic** can be installed from a [GitHub repository](#) using function `install_github()` from the **remotes** package. You are likely to find that it takes 15 minutes or so to install, because **McMasterPandemic** depends on many other packages that you probably do not have installed already. If you are installing an update to **McMasterPandemic** then it won’t take very long at all because most dependencies should not need to be reinstalled.

```
## install remotes package if necessary:
if (!require(remotes)) {
  install.packages("remotes")
}
## install development version of bbmle:
if (!require("bbmle") || packageVersion("bbmle") < "1.0.23.5") {
  remotes::install_github("bbolker/bbmle")
}
## install the target package and all its dependencies:
remotes::install_github("bbolker/McMasterPandemic",
  dependencies = TRUE,
  build_vignettes = TRUE
)
library(McMasterPandemic)
```

If you have any difficulty installing the package, please [e-mail both me and Mikael Jagan using this link](#).

### 3 Familiarize yourself with McMasterPandemic

There is a “getting started” vignette, that should be available to you if the installation went smoothly. You should be able to access it via:

```
library(McMasterPandemic)
vignette("getting_started", package = "McMasterPandemic")
```

Please check that this works, *i.e.*, that you are able to view the vignette with this command. If it fails, please e-mail me and Mikael using the above link.

- Read the vignette and try out all the examples in the vignette.
- Use `run_sim()` to replicate things you tried when exploring the shiny.
- Check the documentation (`?run_sim`) and explore the effects of various changes in the arguments to `run_sim()`.

Did this go smoothly? Were there parts of the vignette that were unclear? Were there things you could do with the shiny that you did not see how to do using the R package directly? Are there things you understand how to do with the package but not with the shiny?

### 4 Simulate a COVID-19 epidemic using the package

Use `run_sim()` to find parameter settings that make the simulation look something like the first wave of the COVID-19 pandemic in Ontario (don’t worry about trying to get an exact match). Include both observation noise and process noise in your simulation.

## 5 Fit parameters to your simulation

Imagine that your simulation of the first wave in Ontario is actually the real data. Use the fitting machinery in **McMasterPandemic** to estimate the model parameters and compare your estimated parameters with the known underlying parameters that you used to generate the simulation.

*This is an important test before using the software to estimate parameters from real data (which you will do in the final project).*

Before attempting this part, you will need to wait for me to add to the “Calibration” section of the vignette, and improve some of the documentation of package functions associated with calibration.

## 6 Feedback on McMasterPandemic

Please provide any comments or criticisms you have about the **McMasterPandemic** package, and the `getting_started` vignette, so we can improve them.

## 7 Technical Comments (read carefully!)

*The comments below apply to all work in this course.*


1. Change the default font size from 10 point to 12 point. The default font size is set in the first line of your L<sup>A</sup>T<sub>E</sub>X document: `\documentclass[12pt]{article}`.
2. The L<sup>A</sup>T<sub>E</sub>X *preamble* used for assignments in this course can be downloaded from the [Assignments](#) page on the course web site. You should `\input{4mbapreamble.tex}` in the preamble of your assignment (*i.e.*, between `\documentclass[12pt]{article}` and `\begin{document}`). This file addresses the following issues and many others:
  - You will need to keep referring to  $\mathcal{R}_0$  in your solutions. To make your life easier, define a new `\R` command like so:

```
\newcommand{\R}{\mathcal R}
```

Then if you type `$_R_0$` in your L<sup>A</sup>T<sub>E</sub>X source file you will get  $\mathcal{R}_0$  in your pdf output.

- Please use the logo  to refer to the R language. Do this by defining this as an `\Rlogo` macro in your L<sup>A</sup>T<sub>E</sub>X preamble like so:



```
\usepackage{xspace}
\newcommand{\Rlogo}{\protect\includegraphics[height=2ex,keepaspectratio]
  {images/Rlogo.pdf}\xspace}
```

Note that you will also need the image file `Rlogo.pdf`, which can be downloaded from the [Assignments](#) page on the course web site. Place `Rlogo.pdf` in an `images` subfolder of the folder where your  script lives.

- Define a comment macro as follows:

```
\usepackage{color}
\newcommand{\de}[1]{\color{red}\bfseries DE:} #1}}
```

This macro allows me to add comments easily in your  $\text{\LaTeX}$  document. For example, the  $\text{\LaTeX}$  code `\de{What a great idea!}` yields **DE: What a great idea!**

3. Good notation is important for making your documents easily comprehensible. Given the  $\text{\LaTeX}$  definitions of `\R` and `\Rlogo` above, it is easy to distinguish the removed class ( $R$ ), the reproduction number ( $\mathcal{R}$ ) and the programming language (). Always do this! Not doing so is sloppy and confusing to readers of your work. Also pay close attention to any other potentially confusing notational issues.
4. Run your source file(s) through a spell checker. Don't submit work that has typos or spelling errors. There is information about spell-checkers that work with  $\text{\LaTeX}$  on the [Software](#) page of the course web site.
5. File and folder names: Please avoid spaces and non-alphanumeric characters in file names and folder names (*e.g.*, do not use `\`, `&`, `#`, `!`, `^`, `%`, `$`, `*` or brackets, though the underscore `_` is fine). For example, instead of naming an  script

`My Math 747 Assignment #1 Question 2(b).R`

choose the file name to be something like

`MyName_Math747_A1_2b.R`







A sensible filename for a  $\text{\LaTeX}$  document that contains your submission for Assignment 1 would be

`MyName_Math747_A1.tex`

and a sensible name for the folder that contains all files for the assignment would be

`MyName_Math747_A1`

6. Do not include any absolute file paths in your code. For example, you should *not* refer to `C:\MyName\Documents\MyFavoriteCourse\datafile.csv` in your code. Keep files you need to read in a subfolder of the folder where you are executing your scripts. Make sure anyone can produce the final `pdf` file without altering any of the files in any way.
7.  $\text{\LaTeX}$  needs you to use single opening and closing quotes. A double quote (`"`) is always interpreted as a closing double quote (`"`). Thus, if you say `"quoted words"` you get `"quoted words"`, whereas ``quoted words'` yields `"quoted words"`.

8. Always use math mode to typeset math. For example: `f(x)` yields  $f(x)$  whereas `$f(x)$` yields  $f(x)$ .
9. Use `typewriter type` when referring to filenames. For example, `{\tt filename.R}` yields `filename.R`. (An alternative is `\texttt{filename.R}`.)
10. When including images in a  $\LaTeX$  document, it is best to save the images from  as `pdf`. If you save as `png` or `jpg` then  $\LaTeX$  will still be happy to display them, but the quality of the image is reduced unnecessarily.
11. The  $\LaTeX$  command for the “much less than” symbol is `\ll`. Don’t use `<<` for this. For example: `$a<<b$` yields  $a << b$  whereas `$a\ll b$` yields  $a \ll b$ . In general, if you typeset some math and it doesn’t look like what you would expect to see in a professionally typeset math book then you can be certain you’re not using the intended  $\LaTeX$  syntax.
12. Always use  $\LaTeX$ ’s built-in function names, *e.g.*,  `$\log(t)$`  correctly yields  $\log(t)$  whereas `$\log(t)$` yields  $\log(t)$ .
13. Avoid explicit spacing commands in  $\LaTeX$  if possible. For example, if you want to have space between each paragraph of your document, then don’t include an extra line break at the end of each paragraph (which could be done via `\`). A better approach is to set the value of the paragraph skip in the preamble (via `\parskip=10pt`, for example). Then you can change the spacing easily in the entire document, and if you later want to use a different format then there won’t be explicit spacing commands lurking around to wreck your output. Even setting the `\parskip` explicitly is considered undesirable by  $\LaTeX$  aficionados, because it will override directives in a  $\LaTeX$  style file; in any case, if you can keep formatting changes to the preamble, it will make your life simpler.
14. Every  script should begin with an opening comment explaining what the script does. What is the purpose of the script? What output will be produced when it is run?
15. Take advantage of ’s vector syntax wherever convenient. For example, if setting line styles for a sequence of lines in a plot or legend, rather than `lty=c(1, 2, 3, 4, 5, 6)` say `lty=1:6`.
16. Wherever appropriate, use ’s assignment operator (`<-`) rather than equals (`=`).
17. Your  $\LaTeX$  code must compile without any errors. Producing a pdf file is not adequate. Others must be able to reproduce the pdf without getting any  or  $\LaTeX$  errors.
18. To make your  code readable, it is very important that you indent appropriately. If you are using **Emacs** then `tab` will indent the current line of code according to standard convention.
19. Make sure figures appear where you want them. The `figure` environment has options that allow you to control placement in the document.

20. Explain your logic in computer code using embedded comments. The comment character in  $\text{\LaTeX}$  is `%`. The comment character in  $\text{\R}$  is `#`.
21. Any graphics must be created in  $\text{\R}$ . Once you get the hang of it, the easiest way to combine  $\text{\R}$  with  $\text{\LaTeX}$  is to use the `knitr` package. You are encouraged to use `knitr`, but if you prefer you can create graphics separately and input them as `pdf` files into  $\text{\LaTeX}$ . (Note that you will be required to use `knitr` for the final project.)
22. A note on importing  $\text{\R}$  graphics into  $\text{\LaTeX}$ : When you run an  $\text{\R}$  script in `RStudio`, any plots are shown by default in the bottom right pane of the `RStudio` window. When you are developing the code to make a plot, that is usually what you want. But in order to get the plot into a  $\text{\LaTeX}$  document you must save the plot as a `pdf` file instead. In order to save a plot into the file `mylovelyplot.pdf`, do the following

```
pdf("mylovelyplot.pdf")
#### INSERT PLOTTING CODE HERE ####
dev.off()
```

The `pdf()` command changes the graphics output device to the named `pdf` file. This command has various optional arguments, such as `width` and `height`, which you may well want to use (rather than accepting the default width and height). The closing command `dev.off()` shuts off the current graphics output device, which means that the `pdf` file be complete. If you forget `dev.off()` then your `pdf` viewer will complain that the file you are trying to view is corrupt. Once you have created the required `pdf` file, to include it in your  $\text{\LaTeX}$  document you can use the following command at the point where you want the plot:

```
\includegraphics{mylovelyplot.pdf}
```

Often, you need to control the size of the plot in your document (which is done with  $\text{\LaTeX}$ 's `\scalebox{}` command) and frequently you will want graphs to appear as figures with captions (which is done using the  $\text{\LaTeX}$  `figure` environment). I also recommend putting all included graphics files into an `images` subfolder rather than cluttering the main folder where you are working. Here's how to implement all these things:

```
\begin{figure}
  \begin{center}
    \scalebox{0.5}{
      \includegraphics{images/mylovelyplot.pdf}
    }
  \end{center}
  \caption{This lovely plot is really inspirational for me.}
  \label{F:mylovelyplot}
\end{figure}
```

Note that I've added a few more details above: I made sure the plot will be centred using the `center` environment and I included a caption using the `\caption{}` command. I also created a label for the figure, the purpose of which is to allow us to refer to this figure by number without knowing what the number is. For example, in your  $\text{\LaTeX}$  document you might say

```
my lovely plot is shown in Figure~\ref{F:mylovelyplot}
```

which will appear in the typeset version as “my lovely plot is shown in Figure 2” (assuming the figure in question is currently the second figure; if you reorder the figures in your document,  $\text{\LaTeX}$  rennumbers everything for you).

*Note:* For those of you who have chosen to use the `knitr` package, you do not want to use the `pdf()` command. `knitr` takes care of the graphics file generation for you.

Finally, if you want to get really fancy (*i.e.*, publication-quality graphics), then rather than the `pdf` device you can use the `tikz` device, which understands  $\text{\LaTeX}$  code in character strings. You will first need to install the `tikzDevice` package. Something that seems to catch everyone who uses `tikz` is that backslashes must be escaped in strings. Thus, for example, if you want  $\sum_n \sin n\theta$  to appear on your plot, the character string you need in your  $\text{\R}$  code is `"$\sum_n \sin\{n\theta\}$"` (note the double backslashes!). To use `tikz` in ordinary  $\text{\R}$  code you would typically use this structure:

```
library("tikzDevice")
tikz("mygraph.tex",standAlone=TRUE)
#### GRAPHICS CODE HERE ####
dev.off()
```

and then you would need to run `mygraph.tex` through  $\text{\LaTeX}$  to produce the desired `pdf` file. If you are using `knitr` then you need the `tikz` chunk option (`dev="tikz"`).

23. If you have a data frame and want to display it nicely in a  $\text{\LaTeX}$  or `knitr` document, you can use `knitr::kable` to get a perfectly reasonable result, or `Hmisc::latex` to get a gorgeous result.
24. In order to use `knitr`, you must select `knitr` (rather than `Sweave`) as the `Sweave` interpreter in `RStudio`. To check this setting in `RStudio`, go to

Preferences → Sweave → Weave Rnw files using

and choose `knitr`.

— END OF ASSIGNMENT —

Compile time for this document: November 18, 2020 @ 21:58