Mathematics 747 / 5GT3 Topics in Mathematical Biology

http://davidearn.github.io/tmb2020

Fall 2020 ASSIGNMENT 2

This project is **due at 11:59pm** on **Friday 18 December 2020**.

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

Imagine that the Public Health Agency of Canada (PHAC) has approached you and asked you to forecast—using the **McMasterPandemic** package—the trajectory of the COVID-19 epidemic over the next year under the following scenarios:

1. the status quo is maintained, and no vaccine becomes available for Canadians until the end of 2021 (*i.e.,* after the end of your forecast);

2. strict lockdown is imposed throughout the country on 18 December 2021, and then relaxed six weeks later;

3. sufficient vaccine to immunize 10 million Canadians will be available on 15 January 2021;

4. no vaccine will be available to Canadians until 15 June 2021.

PHAC really wants you to do this for each province and territory, but that's not realistic given how soon they want your forecast, and because data for some regions is not adequate to make meaningful forecasts. In addition, the four scenarios above are not completely clear, and you will need to clarify with PHAC what they are expecting, and make sure you agree on things you can deliver by the deadline.

Your report to PHAC should include:

- a cover letter, in which you thank them for the opportunity to help, and remind them what you agreed to do for them (and why this is less than what they really wanted); [you can be creative and make up the results of imagined interactions with PHAC; there is no "right answer" – I wasn't there when you were zooming with your PHAC contacts]

- an executive summary (ideally about half a page, and certainly no more than one full page), explaining your results in terms that should be understandable by anyone at PHAC (including those with no mathematical or statistical background);

- a brief report of your results written in LaTeX (perhaps two or three pages of text, plus any figures and tables that help you describe your results clearly);

- a `knitr` supplement that reproduces your results, and is sufficiently well documented that any PHAC employee with an undergraduate degree in mathematics or statistics should be able to follow (you can also assume the reader of the supplement is familiar with R, but not that they are an R wizard).

Your report should be written clearly, concisely, and honestly. Be careful and cautious when describing your inferences, and be sure to emphasize uncertainties and the limitations of your model. Don't claim more than you can.

Your project should be submitted as a tarball or zipped folder that can be run to reproduce your final pdf files. Please read this entire document, including all the technical comments below before starting to work on the project.

1 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 LaTeX document: `\documentclass[12pt]{article}`.

2. The LaTeX *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 LaTeX source file you will get $\mathcal{R}_0$ in your `pdf` output.

   - Please use the logo R to refer to the `R` language. Do this by defining this as an `\Rlogo` macro in your LaTeX 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 R 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 LaTeX document. For example, the 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 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 (R). 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 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 R 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 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. LaTeX needs you to use single opening and closing quotes. A double quote (`"`) is always interpretted 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 R 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 R 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 R'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 R'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 R or LaTeX errors.

18. To make your R 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 LaTeX `%`. The comment character in R is `#`.

21. Any graphics must be created in R. Once you get the hang of it, the easiest way to combine R with 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 LaTeX. (Note that you will be required to use `knitr` for the final project.)

22. A note on importing R graphics into LaTeX: When you run an 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 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 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 LaTeX's `\scalebox{}` command) and frequently you will want graphs to appear as figures with captions (which is done using the 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 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, LaTeX renumbers 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 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 R code is `"$\\sum_n\\sin{n\\theta}$"` (note the double backslashes!). To use `tikz` in ordinary 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 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 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` interpretter 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: December 2, 2020 @ 23:06