A typical work cycle when running simulations

Bram Kuijper

2022-05-31

Contents

Chapter 1

Introduction

This document gives you an overview of some of the key computing practices used in our lab. Enjoy!

Chapter 2

Installing and using software to do your work

To do your research in theoretical or computational biology, you will need to install a bunch of software. I typically try to keep things as free and open access as possible, so that you don't have hassle with license fees. However, this is not always possible (e.g., research projects involving Mathematica or matlab).

Don't install *all* the software listed below - by now, you should have a clear idea what selection of software you need. If not, get in touch asap.

2.1 On a Mac? Install your software using the Homebrew package manager

Homebrew is a small program that allows you to install and update multiple programs on your Mac simultaneously. Indeed, all research software that you will need on your Mac can be installed with homebrew. Using homebrew has the advantage that installation involves a single command, rather than you having to search for the correct version of the software online. Moreover, all installed packages will be updated to their latest versions by using two simple commands: `brew update`, following by `brew upgrade`.

2.1.1 Installing homebrew

To install homebrew, you need to open the Terminal app (see here about where to find the Terminal app on your Mac). Once the terminal is open, copy the single-line install statement from the Homebrew website and paste it into the Terminal app. Then press 'Enter'. You will get a bunch of straightforward questions, after which homebrew installs itself.

2.1.2 Installing your first application using homebrew

After homebrew is installed, you can use it to install other applications. You can search for software available for install through homebrew by using the website formulae.brew.sh.

Let us install the [Textmate] text editor that we might need later. We do so using the Terminal app app, in which we type

```
brew install --cask textmate
```

2.1.3 Updating all applications that have been installed via homebrew

This is where things get handier than installing all software individually:

```
brew update && brew upgrade
```

2.2 Rstudio

Rather than using the absolute bare bones version of R, I'd use Rstudio as it provides a full-blown IDE (integrated development environment) with text editor, variable inspector, file browswer and more. You can either use the Rstudio server that is offered by the University of Exeter, or use a locally installed version of Rstudio. Personally, the server-based version of Rstudio is probably the easiest to use, as it already has tons of packages pre-installed. However, if you have no uni access anymore or don't have continuous internet access, a locally installed version of Rstudio is best.

2.2.1 Using the web-based Rstudio server

You can simply access this by using your web browser. You can access the Rstudio server here: https://rstudio01.cles.ex.ac.uk, using your University of Exeter login.

Once logged in, make sure to run the 4.x.x version, rather than a 3.x.x version. You can change versions on the top right of your Rstudio window within in your web browser. By now

2.2.2 Installing Rstudio on your own computer on windows or mac

If you have a Windows machine, the best way to install Rstudio is to download the Desktop version. If you are on a mac, one could use:

```
brew install --cask rstudio
```

2.3 Installing Python

You will need to install the Spyder IDE and associated packages. To this end, it is best to install Anaconda which provide the whole python bundle and comes with a package manager.

2.4 Installing software to develop C++ programs

When working with the programming language C++, we will need to use the following software:

1. A UNIX Terminal (already installed on a mac, you will need to install on windows - see below)
2. A C++ compiler, which in our case will be `clang` on mac and `g++` on windows (see below).
3. A better-than-normal text editor. When on Windows, install the freely available text editor Notepad++. See the Notepad++ website for installation instructions. When on a mac, install an editor such as textmate (see below).
4. A software build environment like make or cmake.

2.5 Do we really need a UNIX terminal?

Yes, you will need a UNIX terminal to access a broad collection of tools, namely the compiler `g++` o turn your C++ code into an executable programme, the command `git` to get code from repositories and `make` or `cmake` to automate the building of your code.

Sure, it would be possible to do this using Microsoft's Visual Studio. However, we decided to go for a UNIX-based toolchain everywhere, as these allow us to use the same compiler tools as used on the University's Linux computing clusters. This is important, as chances are that later on in your project you may want to run your simulations on one of these clusters to run a lot more replicate simulations or explore a large number of parameter combinations. If one has been working with Visual Studio, it typically is a massive pain to switch back to the different tools/compilers used on these Linux clusters. Hence, that is why we stick to a UNIX toolchain.

2.5.1 Installing a UNIX terminal on Windows

To get a UNIX terminal running on windows, we will install the MSYS2 environment. See here for the installation video. There is also another video that shows you how to subsequently work with the compiler, once installed.

2.5.2 Installing a command-line C++ compiler on windows.

Rather than using any C++ compiler like Visual Studio or Code Blocks, we use the compiler g++, as that compiler is the same as used on our computing clusters. To install it, see the instruction video here.

2.5.3 Using the Mac OS Terminal app

To use the UNIX toolchain on a mac, you will need to use a UNIX terminal. Luckily this is installed as per default on any mac distribution. You can find the Terminal app in the `Applications` > `Utilities` folder on your hard drive. See Apple's support page here for more information on how to open this application.

2.5.4 Installing g++ on a Mac

In order to install g++ on a Mac, you need to follow quite some instructions. Sorry no video.

2.5.5

Chapter 3

A typical work cycle when running simulations

Here we illustrate some of the basic steps we take to run simulations and analyze them. Please type along so that you can see what is happening on your own computer.

3.1 Start the terminal

We assume that you are able to start the a UNIX terminal programme, which should be MSYS2-64 (on windows) or the Terminal programme (on mac). If you don't know where to find your UNIX terminal, read section 2.4 and if that still does not work get in touch.

3.2 Get to your home directory

The home directory on a UNIX terminal is contained in the alias `~` and can be accessed by using the `cd` command in the following way:

```
cd ~
pwd
```

```
## /home/anaduarte
```

where the last line lists the `pwd` command (present working directory) which gives your current location.

Note The home directory in MSYS2 is not the same as your Windows home directory! Rather, MSYS2's home directory `~` maps to something like `C:/MSYS2/home`.

3.3 Check the contents of your home directory

You can check the contents of your home directory using the `ls` (list files) command, where we give it some special 'flags', which are given by a dash `-`, followed by some additional single-character modifiers that change the behaviour of the `ls` command:

```
ls -alnh
```

```
## total 120K
## drwxrwxr-x  5 1000 1000 4.0K May 31 23:02 .
## drwxrwxr-x 31 1000 1000 4.0K May 10 14:39 ..
## drwxrwxr-x  3 1000 1000 4.0K May 31 23:02 _book
## -rw-rw-r--  1 1000 1000  208 May 31 18:31 _bookdown.yml
## -rw-rw-r--  1 1000 1000  564 Jan 29 16:29 Cplusplus.Rmd
## drwxrwxr-x  8 1000 1000 4.0K May 31 23:02 .git
## -rw-rw-r--  1 1000 1000   54 Mar  7 20:42 .gitignore
## -rw-rw-r--  1 1000 1000   51 Mar  7 20:41 .gitignore~
## -rw-rw-r--  1 1000 1000  372 May 31 23:02 index.md
## -rw-rw-r--  1 1000 1000  376 May 31 18:31 index.Rmd
## -rw-rw-r--  1 1000 1000 7.2K May 31 23:02 Installing_Software.md
## -rw-rw-r--  1 1000 1000 7.2K May 31 22:21 Installing_Software.Rmd
## -rw-rw-r--  1 1000 1000  243 May 31 23:02 Lab_handbook.rds
## -rw-rw-r--  1 1000 1000  277 May 31 21:17 lab_handbook.Rproj
## -rw-rw-r--  1 1000 1000  277 Jan 29 11:56 lab_handbook.Rproj~
## -rw-rw-r--  1 1000 1000  491 Jan 29 12:05 _output.yml
## -rw-rw-r--  1 1000 1000  161 Jan 29 12:06 preamble.tex
## -rw-rw-r--  1 1000 1000 6.6K May 31 22:12 refs.bib
## -rw-rw-r--  1 1000 1000 6.6K May 31 22:12 refs.bib.bak
## -rw-rw-r--  1 1000 1000    0 May 31 22:13 refs.bib.sav.tmp
## -rw-rw-r--  1 1000 1000 2.7K Mar  7 22:50 refs_lab_handbook.bib.bak
## -rw-rw-r--  1 1000 1000    0 Mar  7 22:50 refs_lab_handbook.bib.sav.tmp
## -rw-rw-r--  1 1000 1000  128 May 31 23:02 rendera4b72123cd8c.rds
## -rw-rw-r--  1 1000 1000    0 Mar  7 23:46 .Rhistory
## drwxrwxr-x  4 1000 1000 4.0K Jan 29 12:05 .Rproj.user
## -rw-rw-r--  1 1000 1000 4.0K May 31 23:01 SimulationWorkCycle.Rmd
## -rw-rw-r--  1 1000 1000   80 Jan 29 21:07 Unix-toolchain.Rmd
## -rw-rw-r--  1 1000 1000 4.6K May 31 22:14 Writing.Rmd
```

Here we used the flags `-alnh` to make sure (i) we list hidden files (`-a`), (ii) we list all files below each other (the long format: `-l`) rather than dumping all files together on a single line, (iii) we list all files with numeric user and group IDs (`-n`) and finally (iv) we list all dates and numbers in a human-readable format (`-h`). If you want to know more about the documentation of the `ls` command, type:

```
man ls
```

which provides you with a manual page (man page) of the `ls` command. You can close this man page and return to the command line by pressing the `q` button.

3.4 `git clone` the source code if you haven't already done so

All research projects in this lab use code repositories on github, which you can clone to your own computer. Working with code on github has several advantages. One is that it is relatively easy to carry through updates in the code without having to e-mail around umpteen versions of the source code. Moreover, once you have your own github account and I give you write access to my code repository, you can then submit your own updates to my github repository, so that we all have the latest version of the code available.

3.4.1 Finding out whether you previously cloned a code repository

If you vaguely remember using the `git clone` command before, perhaps you should first inspect the output of the `ls -lanh` command to find out whether this is the case. If you find a directory within your home directory that looks remarkably similar to one of the repositories you know enough. If you seem to have `git cloned` before, hold your horses for now and skip the next step.

3.4.2 Clone one of the software repositories

You can clone any particular repository by doing:

```
git clone https://github.com/bramkuijper/YOUR_REPOSITORY
```

where `YOUR_REPOSITORY` is the name of the repository that you should be using from this list.

Once everything is cloned, `cd` into the directory and check out what you find!

3.4.3 Update the software repositroy

If you already have `git cloned` a repository, `cd` to the directory of the repository on your local computer and obtain the most recent version. You do so by typing

```
git pull
```

If you see the following message:

```
Already up-to-date.
```

everything is cool.

3.5 Compiling the software

For a typical simulation, you will find the source code in the `src` directory

Chapter 4

Some Writing Advice

Writing your dissertation or literature review can be a challenge. Here some advice:

4.1 Read about writing: literature that helps improving writing skills

We all read books to improve our writing, even when you are a native speaker and think you have seen it all! (hint: you haven't.) Reading about writing techniques is a great way to improve your skills.

If you want to know more about scientific writing, perhaps the short book by Mack (2018) might well be worth trying. This book is freely available online here. Another good book on how to write scientific papers is Gastel and Day (2022).

For a literature review, it can be helpful to read the paper by Sayer (2018), which is all about how to write review papers.

Next to learning more about the techniques of scientific writing, it might be helpful to improve your general writing skills too. Several good books are available. A first one is Williams (1990) which has later on appeared with different titles. Other books are Pinker (2014) and Zinsser (2006). A classic is Strunk (1959).

Hard copies of most of these books are available at the library or in my office. Of course, you should not be looking around the internet for pdfs of these books!

4.2 Example theory papers to help you write about theory

Papers on theory are a bit different than empirical studies. Hence, here a bunch of examples to see how theory papers are typically written: (Fawcett et al., 2007),(Trimmer et al., 2015),(Kahn et al., 2015).

In a theory paper it is important to explain the parameters that you use and why. In part, such explanations may focus on a comparison to previous theory, as in: "To compare our model to predictions made by the classical hawk-dove game (Maynard Smith and Price, 1973), the cost of losing a fight c and the value v of winning a fight are modeled as unidimensional variables with $c > v$.". Alternatively, you could refer to the empirical literature, as in: "In speckled wood butterflies, it is unlikely that $v > c$, because territories are far too temporary to be of any value."

4.3 Common examples where writing goes wrong

4.3.1 Back up qualitative statements

Avoid sentences like "The random matrix method introduced by May 1977 has been highly important in theoretical ecology. Here we use this technique to understand how mutualisms affect ecosystem dynamics". The first sentence makes a claim about importance, but does not back it up with a statement that indeed testifies of its importance (one may add: "as this approach has been central to the analysis of ecosystem stability in several later studies [citations]"). But even then, you could argue that it still tells very little: why not tell us instead what the random matrix method is about and what it does?

4.3.2 Being overly verbose

Avoid sentences such as "Territory productivity can be measured by a variety of indirect and direct methods (Davies et al 2012). These measures can be used to calibrate simulation models." The mentioning of "indirect and direct methods" adds very little here. Why not give at least an example of an indirect and a direct method? Or perhaps scrap the indirect vs direct and just focus on 'different methods' and then give an example of such a method. Or if you realize the sentence on different methods adds too little, why not omit it altogether?

4.3.3 Make sure each paragraph addresses the broader question

It is easy to get mired into examples or definition questions that are a bit particularly. Unless you make explicit why the paragraph contributes to insight

about the broader question asked in your dissertation, consider what you are writing irrelevant.

4.3.4 Clearly label figures

If you use a figure with different panels, each panel should have a label, such as "A", "B", "C", etc. There are no excuses. If your R package does not do that, edit your figure in a graphics programme, or look at some of the queries about labeling figure panels in `ggplot`, for example here.

4.3.5 Reference all figures

If you don't refer to a figure in the text, you don't need it. Throw it out.

4.3.6 Explain terminology upon first use

Typically, assume that the audience is someone who knows quite something but is not necessarily a scientist. For example, a 2nd-year undergraduate. So any terminology needs to be explained, let alone any abbreviations upon first mentioning.

4.3.7 Various writing errors:

- It is evolutionar*ily* stable strategy, not evolutionary stable strategy
- Temperature-related perturbations instead of temperature related perturbations

Bibliography

Fawcett, T. W., Kuijper, B., Pen, I., and Weissing, F. J. (2007). Should attractive males have more sons? *Behav. Ecol.*, 18(1):71–80.

Gastel, B. and Day, R. A. (2022). *How to write and publish a scientific paper*. Greenwood, Santa Barbara, CA.

Kahn, A. T., Jennions, M. D., and Kokko, H. (2015). Sex allocation, juvenile mortality and the costs imposed by offspring on parents and siblings. *J. Evol. Biol.*, 28(2):428–437.

Mack, C. A. (2018). *How to Write a Good Scientific Paper*. SPIE.

Maynard Smith, J. and Price, G. R. (1973). The logic of animal conflict. *Nature*, 246(5427):15–18.

Pinker, S. (2014). *The sense of style: the thinking person's guide to writing in the 21st century*. Viking, New York, New York.

Sayer, E. J. (2018). The anatomy of an excellent review paper. *Functional Ecology*, 32(10):2278–2281.

Strunk, William, J. (1959). *The Elements of Style (4th edition)*. Allyn & Bacon, Boston, MA.

Trimmer, P. C., Higginson, A. D., Fawcett, T. W., McNamara, J. M., and Houston, A. I. (2015). Adaptive learning can result in a failure to profit from good conditions: implications for understanding depression. *Evolution, Medicine, and Public Health*, 2015(1):123–135.

Williams, J. M. (1990). *Style: toward clarity and grace*. University of Chicago Press, Chicago.

Zinsser, W. (2006). *On Writing Well: The Classic Guide to Writing Nonfiction*. HarperCollins, New York.