INTRODUCTION

DMC (Dynamic Models of Choice) is a collection of R functions and associated tutorials written by Andrew Heathcote with contributions from my colleagues (growing out of DE-MCMC code originally written by Brandon Turner and Scott Brown, with stop-signal material contributed by Dora Matzke), postdoctoral fellows (Yishin Lin and Luke Strickland) and students (Angus Reynolds and Matthew Gretton). Its aim is to allow researchers to fit conventional dynamic models of choice (often called "evidence accumulation models") using Bayesian methods, and, more importantly, to develop their own new DMCs.

DMCs are hard to work with because they are computationally expensive and have strong correlations among their parameters (they are of the charmingly named class of "sloppy" models common in biology). The aim of DMC is to ease some of those difficulties by providing functions that are convenient and as effective as possible given the challenges, as well as to distill our experience in working with these models into a set of tutorials that provide advice as well as showing how to use the functions.

The DMC tutorials were developed as a means of getting supported hands-on experience in workshops with introductory lectures and recommended background reading. Working through the DMC tutorials by yourself is NOT a good place to start if you are new to Bayesian methods and cognitive models. If that describes you then we recommend you first work through Michael Lee and EJ Wagnemakers wonderful "Bayesian Cognitive Modelling" (https://bayesmodels.com/; that was how I learned, by translating the Matlab in an early draft to R). It deals with much more tractable models where things usually work as expected, preparing you for the rigours of DMCs, which are often much less co-operative! More broadly we also recommend "Computational Modelling of Cognition and Behaviour" by Simon Farrell and Steve Lewandowsky. That said, there are lots of notes and references in the tutorials, so if you are more experienced and wiling to experiment then you should be fine to work though them by yourself.

Implementing DMC in R fits with the paramount importance of openness, reproducibility, and being able to quickly modify existing models and (relatively!) easily develop new models. However, that means that speed can be an issue; where that becomes an impediment the open source nature of the code means re-implementing in a faster language is always possible. This also has the advantage of the existing DMC as a benchmark. However, we have found that DMC is often sufficient as long as you have access to larger multicore systems, which usually run Linux. DMC was primarily developed for such systems, and that means it usually also works well on Mac, but Windows can be problematic because, at least at the time of writing, the only multicore solution is the Snowfall package, which is outdated and inefficient (especially in terms of memory management).

THE PHILOSOPHY BEHIND DMC

The best way to think of DMC is to be grateful that you didn't have to implement this stuff from the ground up yourself! It is offered freely but with no guarantees; there is no financial support for the project and no financial benefit to the developers, who are (in the main) cognitive modellers not software engineers, whose main focus is developing new models. That means our code is frequently being updated and likely far from perfect, but remember that you are at liberty to improve it, and probably have more time to do so for particular focused projects. We strongly recommend you check everything, and that is easy to do because DMC is completely open.

We have documented DMC through the tutorials specifically because we want you to have to learn the material they contain before you use the system. Our experience is that providing canned solutions in this area just leads to bad science in the form of just accepting estimates or model selection results without checking whether they are valid. To counter this, the idea of "parameter recovery studies", is built into the bones of DMC. You should alway be sure that you can simulate data from you experimental design with your model and then recover the parameters that generated that model before you trust the results of any fits to real data. For some purposes you might have a lower standard, "data recovery" (being able to fit the

simulated data well but perhaps with some difficulty in getting the parameters back), but even in this case the point is you need to simulate and fit in order to know where you stand.

This philosophy is reflected in how you add new models to DMC, which requires specifying a random function (that simulates data), not just a likelihood function (that allows you to fit data), and in the tutorials, which are all about simulating data and then fitting it. Indeed it is not until the end of the first part of lesson 4 (`dmc_4_1_Simulating.R`) that we explicitly tell you how to use DMC with real data! We strongly recommend that you finish to the end of lesson 4 before you think about fitting your own data. At that time you might also want to work through the advanced lessons that are flagged throughout the first four (mainly those in the lesson 5 series). The first four lessons address three standard models; for those interested in going beyond that, a few new models are developed in the lesson 6 series. If you want to add your own models to DMC then read lessons 2.3 and 2.4 first.

USING DMC

Before using DMC go into the packages directory, open `install_packages.R` and make sure all of the standard CRAN packages are available, as well as installing the tweaked version of the coda package. Many lessons have associated data files that contain the results that are summarised in the lesson. When you run commands you will get different results, so for comparison (or just a first run through because it can often take quite a while to run the commands) you can load these data files: `load_data` commands to do so are in the lessons but commented out. Note that if the RData objects are not in the tutorial/data subdirectory `load_data` will fetch them from the internet. If you want to have all data files local from the start use the `get_all_data.R` script in the tutorial subdirectory.

We welcome bug reports (via email to myself) and like to help users, but our capacity to do so is limited. We are constantly updating DMC, mainly with the aim of improving its ability to implement new models, and that can sometimes cause instabilities for old models. Old versions of DMC are archived on the OSF site (https://osf.io/pbwx8/; available under Release History in the Files section), so take note of the version you use with each project. OSF also provides a facility to archive a self contained version of your work (i.e., the particular version of the base DMC code you used, your modifications, and any scripts, parameters recovery studies, fits and associated data etc.), so that others can verify it or modify it to use in their own projects. This can be achieved via forking the DMC project to your own OSF account (via the 'fork' button at the top right of the project page).

Andrew Heathcote (andrew.heathcote@utas.edu.au)

Hobart, Tasmania, 1/July/2017