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 (Quentin Gronau, 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 rigors 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 through 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 cognitive models and to a lesser degree using already established models (like the DDM or LBA). 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 always be sure that you can simulate data from your 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. We have a paper that reflects this philosophy and provide examples that address more advanced uses of DMC (at the time of writing still in press). If you have time it is worth a read first to get an overview, but the main thing to do is work through the tutorials

The philosophy behind DMC is also reflected in how you add new models, 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. Note also that while we try to keep the comments in the tutorials congruent with the stored data files that sometimes slips, sorry ...

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).

The model directories included in the current version are (see also "TutorialModels.txt" in the dmc/models directory, others are being added constantly, sometimes we slip and include them in a release):

DDM, EXG-SS, LBA, LBA-GoFailure, LBA-GoNoGo, LBA-pda, LBA-T0Dists,
LNR, LNR-GoNoGo, LNR-pda, Wald

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

Amsterdam, 11/July/2019

Tables of DMC lessons and their contents.

Tutorial	Topic	New functions
dmc_1_1	Set up a DMC model object	**load_model** Loads a model class into the R environment. **model.dmc** Creates a dmc model object defining the design of the data. **print.cell.p** Prints parameter values for each cell of a design.
dmc_1_2	Simulating and exploring the LNR/LBA/DDM.	**load_data** Loads data stored in the data folder. **simulate.dmc** Simulates fake data from a model and parameter vector. **data.model.dmc** Converts data into a data-model object that stores additional information about the model necessary for Bayesian sampling. **likelihood.dmc** Returns the likelihood of the data given a parameter vector. **plot.cell.density** Plot the response time density of each response for a given cell of the design. **profile.dmc** Create profile plots (likelihood curves) of data by varying one parameter in the parameter vector.
dmc_1_3	Build an LBA model	Introduces use of the **lba_B.R** model file in models/LBA. **plot.score.dmc** Returns the response accuracy and the mean RT of correct and error responses.
dmc_1_4	Build an LNR model	Introduces use of the **lnr.R** model file in models/LNR.
dmc_1_5	Build a DDM	Introduces use of the **ddm.R** model file in models/DDM.

Tutorial	Topic	New functions
dmc_2_1	Prior distributions - Basic	**prior.p.dmc** Generate a prior object. **plot.prior** Plot the prior distributions. **check.recovery.dmc** Returns a parameter recovery summary, including the difference between the true value and the median sampled value if the true values are given. **summary.dmc** More detailed information about parameter estimates.
dmc_2_2	Prior distributions - Advanced	**log.prior.dmc** Calculates the log of the value of the prior density of a set of parameter values. **log.posterior.dmc** Calculates the summed log posterior likelihood, which is the sum of the log likelihood and the sum of the log prior given specific values of the parameters.
dmc_2_3	Adding new models to DMC	**random.dmc** Internal functions used by *simulate.dmc* to generate random data. **transform.dmc** Internal functions used to transform parameter values to match with the likelihood function. For example, B vs b notation in the LBA.
dmc_2_4	Adding new models to DMC - Advanced	**rlba.norm** Generate LBA data with a normal distributed rate. Based off the rtdists package *rlba_norm*. **p.df.dmc** Internal function that manages parameter values.

Tutorial	Topic	New functions
dmc_3_1	Fit an LNR model	**samples.dmc** Creates an object used to control the sampling process. **run.dmc** Runs sampling iterations. **plot.dmc** Plots DE-MCMC chains. If given a prior object it will compare the prior distributions to the posterior distributions. **rprior.dmc** Sample from a constant prior. **acf.dmc** Plot the auto-correlation of subsequent samples.
dmc_3_2	Assessing the fit of an LNR model	**pick.stuck.dmc** Identify stuck chains. **gelman.diag.dmc** Returns Gelman's diagnostic values, used to determine if the samples have converged. **effectiveSize.dmc** Return the effective sample size of the samples (taking into account the auto-correlations). **post.predict.dmc** Generate predicted data from a posterior. **plot.pp.dmc** Plot density of posterior prediction data and compare to original data. **pairs.dmc** Plot scatters of each possible pair of parameters after sampling. Used to identify correlated parameters. **run.unstuck.dmc** Automatically repeat sampling until there are no stuck chains. **run.converge.dmc** Automatically run sampling until some convergence criteria are met.
dmc_3_3	Fit an LBA model	
dmc_3_4	Fit a DDM model	**ppp.dmc** Posterior predictive p-values.

Tutorial	Topic	New functions
dmc_3_5	Individual subject model selection	**Dstats.dmc** Returns a list of statistics about the posterior deviance. **pd.dmc** Estimates the "true" number of parameters. **plot.deviance.dmc** Plots posterior deviance and shows estimates of the true number of parameters. **posterior.lr.dmc** Performs a posterior likelihood ratio test. **IC.dmc** Calculates DIC in different ways. **wIC.dmc** IC tests with different weights. **trial_log_likes** Returns log likelihoods of samples with thinning. **waic.dmc** Computes Watanabe-Akaike information criterion. **loocompare.dmc** Makes comparisons on two or more WAIC results. **looic.dmc** Computes LOO Information criterion. **p.fun.dmc** Do posterior predictive tests on functions of parameters (e.g., the difference between two parameters).

Tutorial	Topic	New functions
dmc_4_1	Conduct model simulation for multiple subjects	**h.simulate.dmc** Simulate hierarchical data.
dmc_4_2	Prior distributions – Hierarchical models	
dmc_4_3	Posterior distributions – Hierarchical models	**h.log.likelihood.dmc** Calculate the log-likelihood of data given a hierarchical model. **assign.pp** Convenience function used for hierarchical profile plots. **h.profile.dmc** Graph hierarchical profile plots.
dmc_4_4	Fit a fixed effect LNR model	**h.samples.dmc** Set up for hierarchical sampling. **h.run.dmc** Run hierarchical sampling iterations. **h.pick.stuck.dmc** Pick out stuck chains. **h.run.unstuck.dmc** Automatically run hierarchical sampling until there are no stuck chains. **h.run.converge.dmc** Automatically run hierarchical sampling until it reaches convergence criteria. **group_trial_log_likes** Calculates and stores likelihoods at the trial level. **group_subject_log_likes** Calculates and stores likelihoods at the subject level. **h.post.predict.dmc** Generate posterior predictive data from a hierarchical model. **h.check.recovery.dmc** Return information about parameter recovery.
dmc_4_5	Fit a random effect LNR model	**h.gelman.diag.dmc** Calculate Gelman's diagnostic on hierarchical samples.

Tutorial	Topic	New functions
dmc_4_6	Fit fixed- and random-effect LBA models	**get.thin** Estimates how much thinning to do. **make.hstart** Makes hierarchical start points from individual fits **make.theta1** Extracts last sample from individual fits.
dmc_4_7	Fit fixed- and random-effect DDM models	
dmc_5_1	Complex factorial designs	**check.p.vector** Make sure the parameter vector matches the design of the model.
dmc_5_2	Advanced scoring of accuracy	**empty.map** Makes a map object used to define non-standard design. **assign.map** Assigns cells to map object.
dmc_5_3	Hierarchical model selection	**h.IC.dmc** Hierarchical version of *IC.dmc*.
dmc_5_4	Plausible values	**cor.plausible** Calculates plausible correlation values. **postRav** Makes summary of plausible values over subjects. **postRav.Density** Extract density from *postRav* output. **postRav.mean** Extract mean from *postRav* output. **postRav.p** Extract probability < 0 from *postRav* output. **postRav.ci** Extract credible interval from *postRav* output. **compare.r** Compares two sets of plausible value correlations.

Tutorial	Topic	New functions
dmc_5_5	Advanced plotting	**ggplot.RP.dmc** Plots the proportion of each response. **ggplot.RT.dmc** Plots RT quantiles. **get.fitgglist.dmc** Gets a data frame of summary stats for ggplot. **ggplot.RA.dmc** Plots accuracy.
dmc_5_6	Testing parameter effects	**compare.p** Within-subjects posterior inference. **compare.ps** Between-subjects posterior inference.
dmc_5_7	Bayes Factors	**bridge.sampler.dmc** Calculates marginal likelihoods from samples object **bf.dmc** Calcualtes Bayes Factors from marginal likelihoods
dmc_6_1	Fit LBA model with go failure	Introduces use of **lba_Bgf.R** model file in models/LBA-GoFailure.
dmc_6_2	Fit Wald models with go failure	Introduces use of **wald.R** model file in models/Wald.
dmc_6_3	Fit Go-NoGo models	Introduces use of **lnrgng.R** model file in models/LNR-GoNoGo and **lba_Bgng.R** in models/LBA.
dmc_6_4	Fit the ExGaussian stop-signal model with two racers	Introduces use of **exgSS.R** model file in models/EXG-SS. **plot_SS_go.dmc** Plot go RTs. **plot_SS_if.dmc** Plot inhibition function. **plot_SS_srrt.dmc** Plot signal-respond RTs.
dmc_6_5	Fit the ExGaussian stop-signal model with three racers	Introduces use of **exgSSprobit.R** model file in models/EXG-SS.
DMCpaper1	Advanced fitting and assessment of LBA models	**RUN.dmc** Repeatedly run *run.converge* and *run.unstuck* until all the convergence criteria are met. **h.RUN.dmc** Same as above but with multiple subjects.
DMCpaper2	Advanced fitting and assessing of stop-signal models	