Behavior Research Methods manuscript No.
(will be inserted by the editor)

DMCfun: An R package for fitting Diffusion Model of Conflict (DMC) to reaction time and error rate data

Ian G. Mackenzie · Carolin Dudschig

Received: date / Accepted: date

Abstract Decision processes within choice reaction-time (CRT) tasks are often modelled using variations of a Diffusion Decision Model (DDM, for a review, see Ratcliff & McKoon, 2008). Ulrich et al. (2015) introduced a Diffusion Model for Conflict tasks (DMC). The DMC model combines common features from within standard diffusion models with the addition of superimposed controlled and automatic activation. The DMC model is used to explain distributional reaction time (and error rate) patterns in common behavioural conflict-like tasks (e.g., Flanker task, Simon task). This paper introduces the R-package DMCfun, which implements the DMC model and provides functionality to fit the model to observed data.

Keywords reaction time · conflict task · Diffusion Model Conflict (DMC) · R package

1 Introduction

Conflict tasks (e.g., Flanker task, Simon task, Stroop task, see Eriksen & Eriksen, 1974; Lu & Proctor, 1995; MacLeod, 1991, respectively) are common experimental paradigms used to investigate decision processes under response competition (i.e., the activation of competing response alternatives). Such conflict tasks involve stimuli with multiple features that are either relevant or irrelevant to the required task. For example, a typical Flanker task consists of an array (e.g., HHHHH, HHSHH) with the central item being the "target"

Ian G. Mackenzie
Schleichstrasse 4, Rm 4.427
Tübingen
Germany
Tel.: +49 (0)7071 29-75589
E-mail: ian.mackenzie@uni-tuebingen.de

(task-relevant) and the surrounding items being the "flankers" (task-irrelevant). Responses are typically mapped to the left and right hand (e.g., H = left, S = right). When the target and the flankers match (mismatch), both the relevant and irrelevant dimensions indicate the same (different) response. In a typical Simon task, participants respond to a feature of a stimulus (e.g., colour, shape; relevant dimension) that is presented at different spatial locations (e.g., left vs. right side of the screen; irrelevant dimension). Again, responses are mapped to the left and right hands (e.g., blue = left, red = right). When a stimulus presented in a left (right) location requires a left-hand response, the relevant and irrelevant stimulus dimensions match (mismatch). In such tasks, responses are typically faster and less error-prone when the relevant and irrelevant response dimensions indicate the same response (compatible) than when they indicate different responses (incompatible). The longer reaction time and larger error rate in incompatible trials indicates that task-irrelevant information interferes with the processing of the task relevant information.

This compatibility effect pattern is consistent across all conflict-like tasks. However, when the size of the effect (incompatible - compatible) is compared across different percentiles of the RT distribution, different patterns can emerge. Such differences across the RT distribution are investigated via the use of delta functions (or delta plots) (De Jong et al., 1994; Speckman et al., 2008). Such plots depict the compatibility effect (y-axis) at n bins (x-axis) (see Figure 1 bottom-right panel for an example of a delta plot). Such distributional analyses have indicated that the compatibility effect in some conflict tasks (e.g., Stroop task) increases with slower responses (i.e., positive-going delta slope) whereas in other conflict tasks (e.g., Simon task) smaller compatibility effects are observed with slower responses (i.e., negative-going delta slope) (see Pratte et al., 2010, for Stroop and Simon tasks).

The observation of both positive and negative-going delta slopes across different conflict tasks has strong theoretical implications for processing models of RT. For example, traditional diffusion models of RT cannot account for negative-going delta functions (Pratte et al., 2010). These models propose that task-relevant information accumulates over time until one of two decision boundaries is reached (correct vs. incorrect response boundary). Wagenmakers and Brown (2007) demonstrate that the positive relationship between the mean and standard deviation of RT excludes the possibility of observing negative-going delta functions. As summarized by Ulrich et al. (2015), traditional RT models have difficulty explaining negative-going delta functions. The DMC model is an extension of the standard diffusion model that can explain observed delta functions (both positive and negative going) across a range of conflict tasks. Since its publication, simulation and fitting procedures of the DMC model have been applied in various research contexts in order to provide important insights into the mechanisms at play in conflict tasks(e.g., Mittelstädt & Miller, 2020; Servant et al., 2016).

Conditional accuracy functions (CAFs) also offer an additional tool with which to investigate compatibility effects across the RT distribution. Here, accuracy rates are calculated across N equally sized RT bins. For compatible trials, accuracy is typically high across all bins, whereas for incompatible trials, the lowest level of accuracy is observed at fastest RT bins. At the slowest RT bins, there is little difference in accuracy between compatible and incompatible conditions.

The following section introduces the DMC model Ulrich et al. (2015). It must be noted that other extensions of standard drift diffusion models with specific reference to conflict-like tasks have been proposed. For example, the dual-stage two-phase model (DSTP, Hübner et al., 2010) or the shrinking spotlight model (SSP, White et al., 2011). A direct comparison between such models is beyond the scope of the current paper with such comparisons provided elsewhere (see Evans & Servant, 2020; White et al., 2018, albeit with specific reference to the flanker task only).

1.1 DMC Model Details

The DMC model of Ulrich et al. (2015) builds upon the idea that conflict tasks consist of two processes that proceed in parallel (automatic and controlled processing). Here, controlled processing concerns the task-relative information, whereas automatic processing concerns the task-irrelevant information. Whilst the architecture of such combined controlled/automatic processing within individual conflict tasks is under-specified, Ulrich et al. (2015) suggest two possible architectures (see Figure 1 in Ulrich et al., 2015) under which automatic and controlled processing interact. Under both architectures, DMC assumes that a single accumulation process (combined automatic and controlled) determines the executed response. Critically, automatic processes will facilitate (impede) controlled processes in compatible (incompatible) trials. In addition, it is assumed that the activation output from automatic processes increases to a maximum before decreasing back to zero (pulse function).

The formal model specification for DMC is provided in Ulrich et al. (2015) and thus, will only be briefly described here. The time-course of automatic activation is modelled using a Gamma density function with a shape parameter (aaShape, with $\alpha > 1$) and a scale parameter (tau, τ) multiplied by a constant (amp, +ve for compatible, -ve for incompatible trials), with the drift rate (mu, μ) being the first derivative of the density function. This gives DMC 7 parameters that can vary: amp, tau, aaShape, mu, bnds, resMean, and resSD (see Table 1). Like standard diffusion models (see Ratcliff, 2013), DMC also allows increased trial-to-trial variability by allowing the starting point to vary. This starting point is sampled from a general beta distribution (spShape $\alpha > 0$) centered around zero within the boundary range. DMC assumes that total RT is a combination of decision (D) and residual non-decision (R) processes and assumes that compatibility effects influence D processes only. The predictions of DMC are independent of the distributional properties of R. However, to provide more realistic RTs, the non-decisional component (R) was sampled from a normal distribution with a given mean (resMean) and standard deviation (resSD). Within the DMC model, activation accumulates over *t (t > 0)*, with t being the time since stimulus onset. $\mathbf{X}_c(t)$ represents represents the accumulation of noisy information from controlled processes, whereas $\mathbf{X}_a(t)$ is the accumulation of noisy information from automatic processes. For controlled activation, drift rate (μ_c) is assumed to be time-independent, whereas for automatic activation, drift rate (μ_a) depends on *t*. The critical assumption is that total accumulation of information is the sum of the controlled and automatic processes ($\mathbf{X}(t) = \mathbf{X}_c(t) + \mathbf{X}_a(t)$) since it is assumed that automatic activation spills over to the controlled process. Responses are generated whenever total activation exceeds some boundary (*b*).

Table 1 DMC Model Parameters and Default Values within the function dmcSim.

Parameter	Description	Default
amp	Amplitude of automatic activation	20
tau	Time of peak automatic activation with peak being tau · (aaShape - 1)	30
aaShape	Shape parameter of the gamma distribution for tau	2
mu	Drift rate of the controlled activation	0.5
bnds	Decision boundary where response executed	±75
resMean	Mean of the normal distribution of the residual (R) stage	300
resSD	Standard deviation of the normal distribution of the residual (R) stage	30
spShape	Shape parameter of the beta distribution for starting point variability	3
sigm	Scalling parameter of the drift diffusion process	4

Figure 1 (top-left panel) illustrates the model dynamics for automatic and controlled activation averaged over 100,000 compatible and incompatible trials. The solid black line represents controlled activation. The green and red dotted lines represent automatic activation in compatible and incompatible trials, respectively. Critically, the automatic activation induced by the task-irrelevant dimension is brief and time-dependent. The solid green and red lines represent the summed automatic and controlled activation. For compatible trials, automatic and controlled activation both accumulate in the same direction as the correct response, whereas for incompatible trials, automatic and controlled activation produce opposite activation levels. This results in delayed boundary level activation for incompatible compared to compatible trials.

Ulrich et al. (2015) collected data from both a flanker task and a simon task and fitted this data to DMC. Here, the delta (RT) and CAF (accuracy) values from the observed and predicted data were calculated. The model was fitted by minimising the root-mean-square error (RMSE) weighted difference between the predicted and observed delta and CAF values using the SIMPLEX minimization routine of Nelder and Mead (1965).

2 R Package: DMCfun

The R-package DMCfun provides functionality allowing both the simulation of the DMC model and fitting the model to observed data. The simulation code is written in C++ utilising the R-package Rcpp (Eddelbuettel et al., 2011). The use of C++ gives a significant performance boost over base R whilst the Rcpp interface allows convenient use from within the standard R environment and also direct access to plotting facilities within R. The fitting procedure uses the R-package optimr Nash (2016) with method = “Nelder-Mead”. As in Ulrich et al. (2015) the model fit was evaluated by calculating the RMSE weighted difference between the predicted and observed delta/CAF values. The following sections will provide a tutorial-type overview of using the package, both for running the basic simulation and also fitting the observed data from Ulrich et al. (2015).

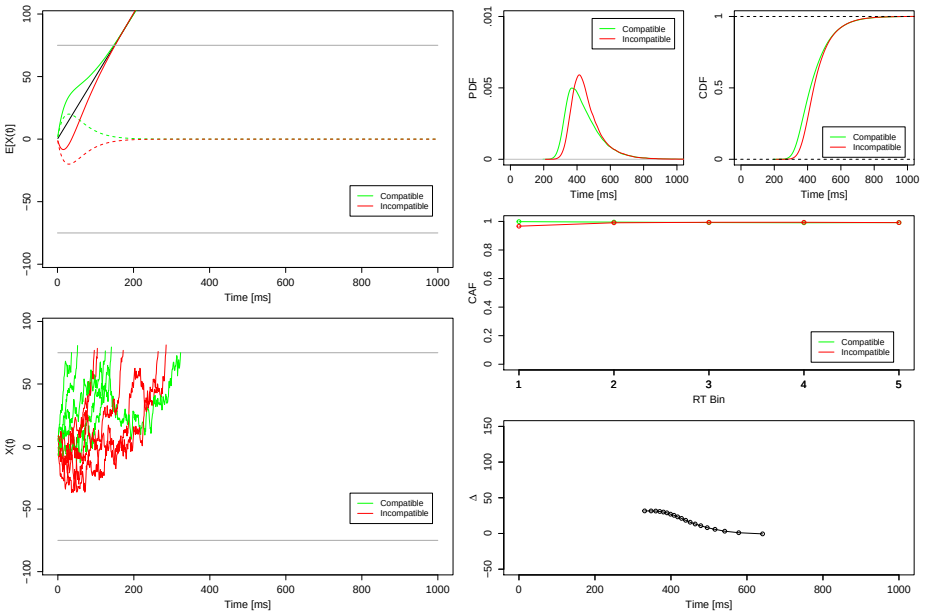

Fig. 1 DMC simulation. This simulation involved 100,000 trials for each compatibility condition with the following default simulation parameters: amp = 20, tau = 30, mu = 0.5, bnds = 75, resMean = 300, resSD = 30, aaShape = 2. The upper-left panel shows the mean activation functions with the black line representing controlled activation. The dotted green and red lines represent the automatic activation for compatible and incompatible trials, respectively. The solid green and red lines represent the sum of controlled and compatible and incompatible automatic activation, respectively. The bottom left panels shows 5 individual compatible and incompatible trials. A response is generated when activation exceeds the boundary level (horizontal straight lines). The right upper two panels show the probability distribution function (PDF) and the cumulative distribution function (CDF) for both compatible and incompatible trials. The middle right panel show the conditional accuracy function (CAF) for compatible and incompatible trials. The bottom right panel show the delta function for the compatibility effect (incompatible - compatible).

2.1 Gettting Started

The package is currently hosted on GitHub (https://github.com/igmmgi/DMCfun). It is anticipated that the package will also be submitted to the R CRAN repository. The package can be installed directly from GitHub using the devtools package (via install_github, see Code Example 1).

R Code Example 1: Instalation

```
> # install.packages("devtools")  # if required
> devtools::install_github("igmmgi/DMCfun")
> library(DMCfun)
> help(package = "DMCfun") # or ?DMCfun
```

2.2 DMC Simulation

The DMC simulation is run using the function *dmcSim* (see Code Example 2 and Table 1 for DMC parameters). This function allows the user to specify parameter values using named arguments (?dmcSim). A single simulation run with 100,000 (default) trials in each compatibility condition takes approximately $<$ 150 ms (standard desktop PC). Figures from the simulation are produced by calling *plot* on the output of dmcSim. The function dmcSims allows the user to specify a list of input parameters in order to run multiple simulations. For example, Ulrich et al. (2015) observed that the parameter tau (τ) largely determined the slope to the delta function. In Code Example 3, the DMC simulation is run using tau starting parameters ranging from 20 to 170 in steps of 10 (all other parameters held constant, see Figure 2).

R Code Example 2: Calling dmcSim and plotting output

```
> dmc <- dmcSim(fullData = TRUE)            # Fig 3 Ulrich et al. (2015)
> dmc <- dmcSim(fullData = TRUE, tau = 150) # Fig 4 Ulrich et al. (2015)
> ?plot.dmcsim                              # Plot options for dmcSim
> plot(dmc)
> plot(dmc, figType = "caf")
> plot(dmc, figType = "delta")
>
> dmc <- dmcSim()
> plot(dmc)                                 # Fig 3 Ulrich et al. (2015)
> dmc <- dmcSim(tau = 150, bnds = 100)
```

R Code Example 3: Calling dmcSims and plotting output

```
> dmc <- dmcSims(list(tau = seq(20,170,10)))
> plot(dmc[[1]])  # full plot first combination only
> plot(dmc)       # plot delta functions for each combination
```

2.3 DMC Fit: Real data

The package includes the raw data used in the original simuluations from Ulrich et al. (2015). This data was collected from a participants performing both a standard visual simon task and a flanker task. It can be accessed in raw form or a summarized form from within R (see Code Example 2). The raw data contains the four essential columns required; specifically, a column coding participant number (VP), a column coding the compatibility condition (Comp), a column with the reaction time data (RT) and a column indicating if the response was correct or incorrect (Error, with 1 indicating an error). The summarized data contains the particpant and grand-average mean RTs, erorr-rates, CAF values and delta values. This data can be fitted to DMC via the function *dmcFitAgg* (see Code Example 5). This will fit DMC to the data averaged across all participants, whereas the function *dmcFitVPs* will fit DMC to individual participants. A summary of the fit is provided. It is possible to

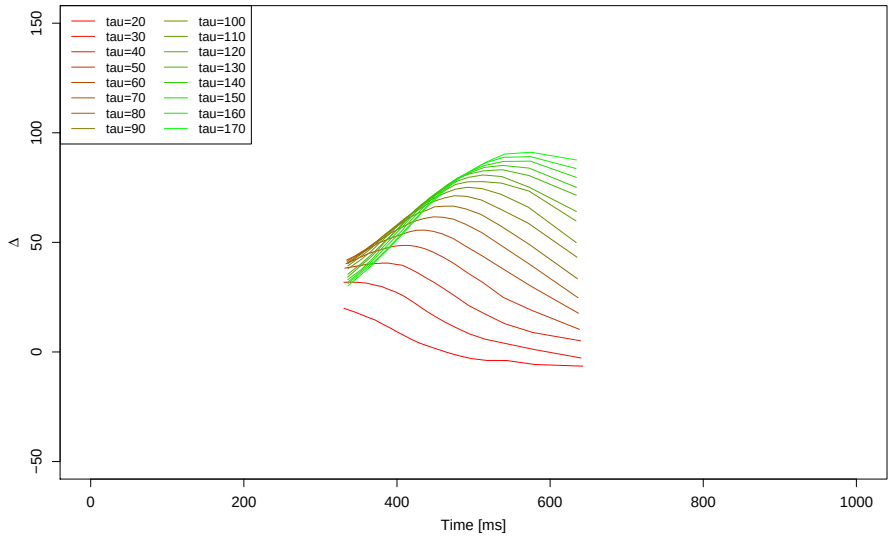

Fig. 2 DMC Simulation delta functions with varying tau parameter only.

Table 2 DMC Fitting Parameters and Default Values within the functions *dicFitAgg/dmcFitVPs*. Note that the defualt settings within *dicFitAgg/dmcFitVPs* fix the sigm value to its starting value.

Parameter	Starting Value	Min Value	Max Value
amp	20	10	40
tau	200	5	300
mu	0.5	0.1	1
bnds	75	20	150
resMean	300	200	800
resSD	30	5	100
aaShape	2	1	3
spShape	3	2	4
sigm	4	1	10

fit the following parameters: amp, tau, mu, bnds, resMean, resSD, aaShape, spShape and sigm (NB. By default, the value sigm is fixed). The functions *dmcFitAgg/dmcFitVPs* use the R-package optimr (Nash, 2016) with method = "Nelder-Mead", and allows the specification of a number of options including which parameters to fit and their starting values. Table 2 provides a summary of the initial starting values for the to-be-estimated parameters and also the boundaries of the search space for each parameter.

The cost function to minimize was identical to that used within Ulrich et al. (2015); specifically, the root-mean-square-error (RMSE). In detail, a cost value was calculated for both the percentile reaction time data ($RMSE_{RT}$) and the binned error data ($RMSE_{CAF}$) with the total cost being a weighted sum of the two

$$RMSE = w_{RT} \cdot RMSE_{RT} + w_{CAF} \cdot RMSE_{CAF}$$

with w_{RT} being determined by the number of bins within the RT and CAF data across both compatibility conditions such that

$$w_{\text{RT}} = \frac{2 \cdot N_{\text{RT}}}{2 \cdot N_{\text{RT}} + N_{\text{CAF}}} \quad and \quad w_{\text{CAF}} = (1 - w_{\text{RT}}) \cdot 1500$$

where N is the number of bins. The actual calculation of the RMSE_{CAF} and RMSE_{RT} is as follows:

$$RMSE_{\text{RT/CAF}} = \sqrt{\frac{1}{N \cdot 2} \sum_{b=1}^{N} \sum_{c=1}^{2} [X_{th;b,c} - X_{ob;b,c}]^2}$$

where X is the is the data from either the percentile RTs (RMSE_{RT}) or CAF (RMSE_{CAF}) bins. Here, *b* runs over all of the bins, *c* over the two compatibility conditions, and *th* and *ob* are the theoretical and observed data, respectively.

Ulrich et al. (2015) observed that the model fit was highly influenced by the tau parameter. By default setting of *dmcFitAgg/dmcFitVPs* is to first searche a grid space (default steps N = 10) within this single parameter to find the "best" starting value of tau before the simplex minimisation routine is implemented. An initial search of a parameter space has been shown to improve parameter recovery (Hübner & Pelzer, 2020). Such a initial search grid and subsequent fitting procedure takes $<$ 1 minute. It is also possible to search the initial grid space across multiple parameters with the search space being determined by the input arguments minVals, maxVals, and fitInitialGridN. It should be noted that increasing the size of the initial grid space will increase computational time prior to the simplex minimisation routine dramatically. For example, searching 6 parameters within a gridspace of 6, results in 46,656 initial parameter sets. Although these are independent and are run in parallel, such an initial grid space takes approximately 20 minutes (tested on 12 core desktop PC). As a result, the default setting is to perform the initial search in the tau space only. This procedure produces consistently good fits with the data of Ulrich et al. (2015) (RMSE $<$ 15), although this does not exclude the possibility that other datasets would benefit from an extended search grid (or alternative starting values) regarding the initial values for the simplex routine.

The function *dmcObservedData* creates the required dataformat for *dmcFitAgg/dmcFitVPs* from raw behavioural data (see Code Example 6). Whilst the focus of the Ulrich et al. (2015) paper was on the flanker and simon tasks, DMC can be used to fit data from any "conflict-like" task , where there is both a task-relevant and a task-irrelevant dimension. The function dmcObservedData accepts as the first argument, an R dataframe (or *.txt file(s)) (see Table 3 and calculates the required variables. The R dataframe or *.txt data files require a minimum of rour data columns: one for participant number, one for compatibility, one for reaction time, and one for error rate. The specific column names and column codings can be specified using function arguments.

R Code Example 4: The package contains the data from Ulrich et al. (2015). A figure of the summarized data can be produced by calling *plot* with the observed data as an input argument.

```
flankerDataRaw      # raw flanker data
flankerData         # summarised flanker data
plot(flankerData)
simonDataRaw        # raw simon data
simonData           # summarised simon data
plot(simonData)
```

R Code Example 5: The example data from Ulrich et al. (2015) can be fitted using the function *dmcFitAgg*. This fits DMC to the aggregated data. A call to *plot* with the output of fit as the first input argument and the observed data as the second input argument will plot the predicted values from the model fit against the observed values from the data (see Figures 3 and 4). A call to *summary* will display the parameter estimates.

```
> fit <- dmcFitAgg(flankerData) # flanker data from Ulrich et al. (2015)
> plot(fit, flankerData)
> summary(fit)

# output values from flanker data
    amp   tau  mu  bnds resMean resSD aaShape spShape sigm RMSE
1 19.42 84.22 0.6 56.47  325.14 28.28    2.24     2.8    4 7.69

> fit <- dmcFitAgg(simonData)   # simon data from Ulrich et al. (2015)
> plot(fit, simonData)
> summary(fit)

# output values from simon data
   amp   tau   mu  bnds resMean resSD aaShape spShape sigm  RMSE
1 17.5 41.33 0.61 61.31  312.51 30.67    1.93    3.32    4 11.57
```

R Code Example 6: The function *dmcObservedData* creates the required data format to use within *dmcFitAgg*.

```
> ?dmcObservedData
> datOb <- dmcObservedData(dat,
                           columns = c("SNo", "comp", "rt", "err"),
                           compCoding = c(1, 2),
                           errorCoding = c(0, 1))
> plot(datOb)
> dmcFit <- dmcFitAgg(datOb)
> plot(dmcFit, datOb)
```

Table 3 Example Raw Data Format. To calculate the required observed data from raw reaction times and error data, four columns are essential: 1) A column coding participant number 2) A colum coding compatibility condition 3) The trial RT (in ms) and 4) A colum coding if the trial was correct/incorrect. It should be noted that the column names and the coding used within the compatibility and error columns can be set within the function dmcObservedData. The function *dmcObservedData* accepts both a filename(s) or an existing R dataframe.

VP	Comp	RT	Error
1	comp	509	0
1	comp	451	0
1	comp	398	0
1	incomp	539	1
...	...	...	...
2	incomp	712	0
2	comp	589	1
2	incomp	756	0
2	incomp	639	1
...	...	...	...

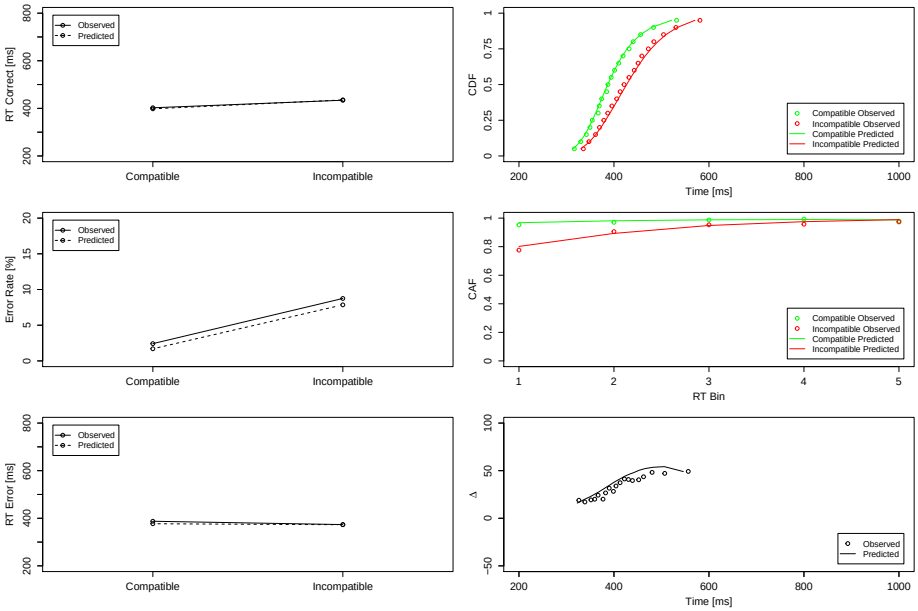

Fig. 3 DMC Fit from the Flanker data presented in Ulrich et al. (2015). Refer to Code Example 5.

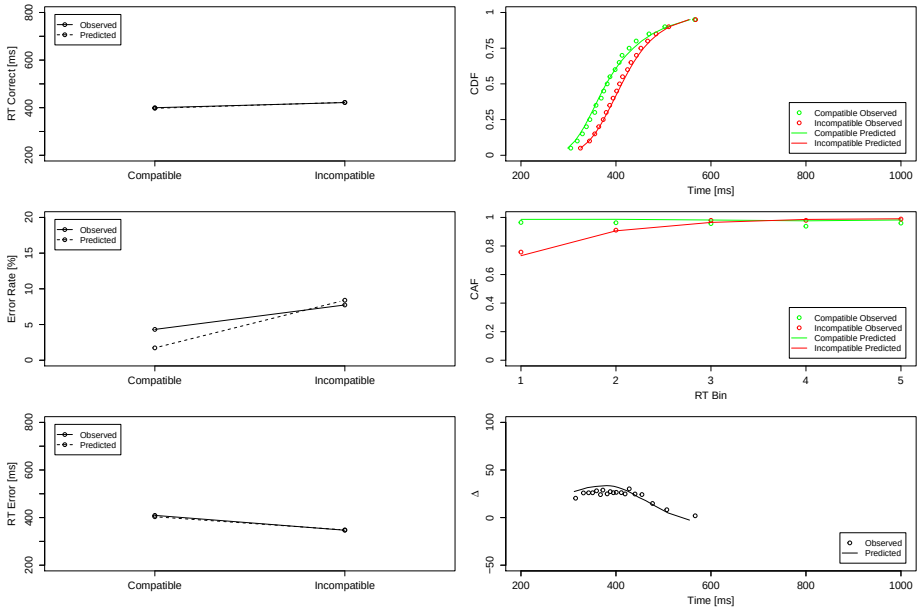

Fig. 4 DMC Fit from the Simon data presented in Ulrich et al. (2015). Refer to Code Example 5.

3 Conclusions

To summarize, the R-package DMCfun provides users with an easy-to-use interface for both simulating the DMC model with various input parameters and fitting this model to observed behavioural data from within conflict-like tasks.

Acknowledgements We would like to thank Rolf Ulrich for providing the raw data from Ulrich et al. (2015) and Rolf Ulrich, Hartmut Leuthold, and Victor Mittelstädt for providing valuable comments on an earlier draft of this manuscript.

Conflict of interest

The authors declare that they have no conflict of interest.

The source code and data used are available at https://github.com/igmmgi/DMCfun

References

De Jong, R., Liang, C.-C., & Lauber, E. (1994). Conditional and unconditional automaticity: A dual-process model of effects of spatial stimulus-response correspondence. *Journal of Experimental Psychology: Human Perception and Performance*, *20*, 731–750.

Eddelbuettel, D., François, R., Allaire, J., Ushey, K., Kou, Q., Russel, N., Chambers, J., & Bates, D. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*, 1–18.

Eriksen, B. A., & Eriksen, C. W. (1974). Effects of noise letters upon the identification of a target letter in a nonsearch task. *Perception & Psychophysics*, *16*, 143–149.

Evans, N. J., & Servant, M. (2020). A comparison of conflict diffusion models in the flanker task through pseudolikelihood bayes factors. *Psychological Review*, *127*, 114–135.

Hübner, R., & Pelzer, T. (2020). Improving parameter recovery for conflict drift-diffusion models. *Behavior Research Methods*, 1–19.

Hübner, R., Steinhauser, M., & Lehle, C. (2010). A dual-stage two-phase model of selective attention. *Psychological Review*, *117*, 759–784.

Lu, C.-H., & Proctor, R. W. (1995). The influence of irrelevant location information on performance: A review of the simon and spatial stroop effects. *Psychonomic Bulletin & Review*, *2*, 174–207.

MacLeod, C. M. (1991). Half a century of research on the stroop effect: An integrative review. *Psychological Bulletin*, *109*, 163–203.

Mittelstädt, V., & Miller, J. (2020). Beyond mean reaction times: Combining distributional analyses with processing stage manipulations in the simon task. *Cognitive Psychology*, *119*, 101275.

Nash, J. C. (2016). *optimr: A replacement and extension of the 'optim' function*. https://CRAN.R-project.org/package=optimr

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*, 308–313.

Pratte, M. S., Rouder, J. N., Morey, R. D., & Feng, C. (2010). Exploring the differences in distributional properties between stroop and simon effects using delta plots. *Attention, Perception, & Psychophysics*, *72*, 2013–2025.

Ratcliff, R. (2013). Parameter variability and distributional assumptions in the diffusion model. *Psychological Review*, *120*, 281–292.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, *20*, 873–922.

Servant, M., White, C., Montagnini, A., & Burle, B. (2016). Linking theoretical decision-making mechanisms in the simon task with electrophysiological data: A model-based neuroscience study in humans [PMID: 27315275]. *Journal of Cognitive Neuroscience*, *28*https://doi.org/10.1162/jocn˙a˙00989, 1501–1521. https://doi.org/10.1162/jocn_a_00989

Speckman, P. L., Rouder, J. N., Morey, R. D., & Pratte, M. S. (2008). Delta plots and coherent distribution ordering. *The American Statistician*, *62*, 262–266.

Ulrich, R., Schröter, H., Leuthold, H., & Birngruber, T. (2015). Automatic and controlled stimulus processing in conflict tasks: Superimposed diffusion processes and delta functions. *Cognitive Psychology*, *78*, 148–174.

Wagenmakers, E.-J., & Brown, S. (2007). On the linear relation between the mean and the standard deviation of a response time distribution. *Psychological Review*, *114*, 830–841.

White, C. N., Ratcliff, R., & Starns, J. J. (2011). Diffusion models of the flanker task: Discrete versus gradual attentional selection. *Cognitive Psychology*, *63*, 210–238.

White, C. N., Servant, M., & Logan, G. D. (2018). Testing the validity of conflict drift-diffusion models for use in estimating cognitive processes: A parameter-recovery study. *Psychonomic Bulletin & Review*, *25*, 286–301.