Table of Contents

# Data access workflow

Prior to Stage 1 submission of the Registered Report, we accessed the cognitive task data for a couple of [preregistered](https://github.com/stefanvermeent/abcd_ddm/preregistrations/2022-09-20_preregistration_DDM.md) data checks. By only accessing the cognitive task data, these steps did not bias or substantive analyses involving measures of adversity. To transparently show when we accessed which data, we created an open science workflow that would automate this process. The main aim of this workflow was to create a transparent log of every major milestone of the project, such as accessing new data, submitting preregistrations, and finalizing analyses.

The main ingredient of this workflow is a set of custom functions that we created for reading in data files (See Figure 1). These are wrappers for the read functions in the *readr* package. Whenever one of these functions (e.g., *read\_csv*) was called, it went through a couple of internal processes. First, the specified data file would be read into R (but not yet accessible to us in the global environment). This could be a single file, or a list of individual data files that would first be combined into a single dataframe. Second, any specified manipulations would be applied to the data. This could be selecting specific variables, filtering specific rows, or randomly shuffling values (e.g., participant IDs). Third, An MD5 hash of the final R object would be generated using the *digest* package. An MD5 hash is a unique, 32-digit string that maps directly onto the content of the R object. The same R object will always generate the same MD5 hash, but as soon as anything changes (e.g., a variable is added, a value is rounded), the MD5 hash changes. Fourth, this MD5 hash would be compared to previously generated hashes.

If the newly generated MD5 hash was not recognized, this triggered an automatic commit to GitHub. At this point, the user gets the choice to abort the process or to continue. Aborting would terminate the process without importing the data. If opting to continue, the user could supply an informative message (e.g., “accessed Flanker data”), which would be added to the Git commit. The Git commit message stored other relevant meta-data as well, such as the object hash and the code used to read and manipulate the data. Committing and pushing to Git was handled using the *gert* package.

Thus, any accessing of raw data was automatically tracked via GitHub. Using this same approach, we also logged other major milestones, such as submitting preregistrations and finalizing analyses.

An automatically generated overview of all milestones can be found in the [Data Access History](https://github.com/stefanvermeent/abcd_ddm/data_access_history.md).

|  |
| --- |
| **Figure S1.** Graphical overview of the data access workflow using R and GitHub. |

# Power analysis

We conducted a power analysis through simulation using the *simulateData* function of the *lavaan* package. On each iteration, we first specified a population model (i.e., the ‘true’ model) with prespecified factor loadings and regression coefficients. Factor loadings in this model were randomly generated between 0.6 and 0.8 following a uniform distribution. Next, we simulated data sets based on the population model. Finally, we fitted a sample model (i.e., without constrained parameters) to the simulated data and extracted the beta coefficients and corresponding *p*-values. We generated population models with beta coefficients of 0.06, 0.08 and 0.1, and simulated data with sample sizes ranging from 1,500 to 8,500 with steps of 1,000. Each combination of coefficients and sample sizes was repeated 500 times, for a total of 12,000 iterations.

The results are shown in Figure S1. The simulations yield power > 90% at around *N* = 2,500 for = 0.1 and *N* = 6,500 for = 0.06. Thus, after taking out 1,500 participants for the training set, the test set will be highly powered.

|  |
| --- |
| **Figure S2.** Results of the power simulations. The dashed line indicates 90% power. |

# Response Distributions of Cognitive Tasks

**Table S1.** Descriptive statistics of mean RTs and accuracy for the cognitive tasks.

|  | RT Mean (SD) | Accuracy Mean (SD) | Accuracy Min | Accuracy Max |
| --- | --- | --- | --- | --- |
| Processing Speed | 2.24 (0.47) | 96.42 (4.3) | 55.17 | 100 |
| Flanker | 0.91 (0.33) | 99.31 (3.25) | 52.63 | 100 |
| Mental Rotation | 2.65 (0.47) | 59.25 (16.81) | 6.25 | 100 |
| Attention Shifting | 1.01 (0.35) | 92.94 (6.76) | 22.22 | 100 |

# Overview of DDM Modeling Procedure

The hierarchical DDM models will be fit using the *runjags* package (**denwood\_2016?**) with JAGS code adapted from Johnson et al. (2017). The JAGS code will be adjusted in a number of ways to meet our purposes. Across all models, the starting point will be fixed to 0.5, and the boundary separation will be will be constrained to be the same across conditions where relevant. Each model will be fit with three Markov Chain Monte Carlo (MCMC) chains. Each chain will contain 2,000 burn-in samples and 10,000 additional samples. Of these samples, every 10th sample will be retained. Posterior samples of all three chains will be combined, resulting in a posterior sample of 3,000 samples. If a model does not converge properly with these settings, we will increase the amount of samples drawn stepwise up to 100,000.

Model convergence will be assessed in several ways. First, we will visually inspect the traces, which should not contain any drifts or large jumps. Second, we will calculate the Gelman-Rubin convergence statistic R^ (Gelman & Rubin (1992)), of which all values should be below 1.1. Third, we will assess whether the model provides a good fit to the participants’ data. To do this, we will simulate RT and accuracy data using the estimated DDM parameters of each participant (using the same number of trials as the participant). We will then fit the DDM to these simulated data using the same model specification. We will compute correlations between the observed and simulated scores for RTs in the 25th, 50th and 75th percentile of the RT distribution as well as for accuracies. We will consider *r* > .80 to indicate a good model fit.

In theory, the hierarchical Bayesian framework allows simultaneously estimating DDM parameters, latent measurement models, and the regression paths between them in a single step (e.g., Schubert et al., 2019; Vandekerckhove, 2014). An advantage of this approach is that information regarding estimation uncertainty (e.g., of the DDM parameters) gets integrated in subsequent steps. However, this approach is very computationally expensive and might even be unfeasible with the current sample size. Therefore, we opted for a two-step estimation approach.

During preprocessing, we discovered that the 5-second response cut-off that was used for the Mental Rotation Task led to severe truncation of the RT distribution. This is problematic because the tail of the distribution holds important information about stages of processing. Truncation of reasonably long RTs can therefore lead to biased DDM parameter estimates. The hierarchical Bayesian framework allows these missing values to be imputed based on the rest of the data, which has been shown to lead to unbiased estimates. The procedure is described in detail in the supplemental materials of Johnson et al. (2017). In short, it involves two steps. First, responses are sampled probabilistically for each missing trial based on the overall accuracy of the participant. For example, if a participant has an overall accuracy of 80%, each missing response has a probability of .80 to be assigned a 1 (i.e., correct response). Second, responses are assigned to three bins. The first bin contains incorrect (imputed) RTs slower than 5 seconds (coded as -5). The second bin contains the observed data, ranging between -5 and 5 seconds. The third bin contains correct (imputed) RTs slower than 5 seconds (coded as 5). JAGS then imputes the response times for missing trials based on these thresholds. We will compare model versions with and without imputation of missing responses.

# DDM simulations: The effect of few trials per participant

The number of trials that is available for each of the cognitive tasks is substantially lower than is typical for DDM analyses. This is especially true for the Flanker Task (8 incongruent trials, 12 congruent trials) and the Attention Shifting Task (7 switch trials, 23 repeat trials). While each participant completed a small number of trials, the hierarchical Bayesian framework can use information from the full sample to estimate and constrain individual estimates. Here, we report simulation studies that aimed to assess whether it would be possible to accurately recover parameter estimates. The analyses are modeled on the Flanker Task, which is the task with the lowest overall number of trials (*N* = 20). For simulations involving two conditions, we assume (as we do in the real data) that the drift rate and non-decision time differ (and are correlated) across conditions, and that the boundary separation is the same across conditions. This latter assumption reflects the fact that conditions are randomly shuffled on a trial-by-trial basis, which prohibits participants from adapting their strategy for different conditions. The starting point is fixed to the mid-point (0.5) for all simulations.

## DDM simulation 1: Single condition with eight trials

First, we simulated task data for 1,500 participants with eight trials per participant. We used the first 2,000 samples as burn-in, and then took an additional 10,000 samples. Every 10th sample was discarded to reduce the size of final model object. We sampled across three chains, which were subsequently combined, for a total of 3,000 samples. The model converged normally (Figure S3). Relative parameter recovery was decent for boundary separation (*r* = .76) and non-decision time (*r* = .73), but not for drift rate (*r* = .54). However, estimates of boundary separation and non-decision time showed substantial bias (See Figure S4).

|  |
| --- |
| **Figure S3.** Convergence of the model in simulation 1. Plots should resemble a ‘fat, hairy caterpillar’. |

|  |
| --- |
| **Figure S4.** Parameter recovery in the case of two conditions. a = Boundary Separation; t = Non-Decision Time; v = Drift Rate |

As discussed above, the models reported in this manuscript will not be constricted to eight trials. Instead, they will be able to use the information of both conditions (e.g., congruent and incongruent for the Flanker Task), as parameters will tend to be correlated across conditions. Therefore, we ran a second simulation adding realistic condition effects.

## DDM simulation 2: Two conditions; Boundary Separation fixed across conditions —-

We again simulated task data for 1,500 participants. Mirroring the real Flanker task, we simulated two conditions, one with 8 trials (incongruent) and one with 12 trials (congruent). On average, drift rates were lower and non-decision times were longer for incongruent trials. Boundary separation was fixed within subjects to be equal across conditions. Non-decision times correlated on average .70 between conditions, and drift rates correlated on average .30 between conditions. These correlations were based on previous studies that we did using the Flanker Task. For more information on the specific settings, see <https://github.com/stefanvermeent/abcd_ddm/scripts/0_simulations/ddm_trial_simulations.R>.

As the model converged without issues in simulation 1, we tried reducing the number of samples (2,000 burn-in with an additional 2,000 samples) to save time. The model converged normally (Figure S5). Correlations between simulated and recovered parameter estimates was high, ranging between *r* = .84 for the drift rate and .95 for the non-decision time (see Figure S6).

|  |
| --- |
| **Figure S5.** Convergence of the model in simulation 2. Plots should resemble a ‘fat, hairy caterpillar’. |

|  |
| --- |
| **Figure S6.** Parameter recovery in the case of two conditions. a = Boundary Separation; t = Non-Decision Time; v = Drift Rate |

Simulation 1 and 2 involved data of 1,500 simulated subjects. However, the sample size of our real data set is roughly 10,000. Thus, in the real data there is substantially more group-level data to inform and constrain the individual parameter estimates. we ran a third simulation to investigate if—and to what extent—the parameter estimates would improve moving from 1,500 to 10,000 participants.

## DDM simulation 3: Two conditions; 10,000 subjects

We simulated task data for 10,000 participants. All other simulation settings were identical to simulation 2.

The model converged normally (Figure S7). Correlations between simulated and recovered parameter estimates were high and very similar to those found in simulation 2, ranging between *r* = .83 for the drift rate and .95 for the non-decision time (see Figure S8). Thus, the benefit of adding more subjects is already saturated around 1,500 participants, with additional participants not improving parameter estimation.

|  |
| --- |
| **Figure S7.** Convergence of the model in simulation 3. Plots should resemble a ‘fat, hairy caterpillar’. |

|  |
| --- |
| **Figure S8.** Parameter recovery in the case of two conditions. a = Boundary Separation; t = Non-Decision Time; v = Drift Rate |

Overall, we conclude that applying hierarchical Bayesian DDM to the ABCD data is feasible.

# DDM Model Fit Assessments

*Will be presented after Stage 1 in-principle acceptance.*

# SEM Fit

*Will be presented after Stage 1 in-principle acceptance.*

# References

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 457–472. <https://doi.org/10.1214/ss/1177011136>

Johnson, D. J., Hopwood, C. J., Cesario, J., & Pleskac, T. J. (2017). Advancing research on cognitive processes in social and personality psychology: A hierarchical Drift Diffusion Model primer. *Social Psychological and Personality Science*, *8*(4), 413–423. <https://doi.org/10.1177/1948550617703174>

Schubert, A.-L., Nunez, M. D., Hagemann, D., & Vandekerckhove, J. (2019). Individual differences in cortical processing speed predict cognitive abilities: A model-based cognitive neuroscience account. *Computational Brain & Behavior*, *2*(2), 64–84.

Vandekerckhove, J. (2014). A cognitive latent variable model for the simultaneous analysis of behavioral and personality data. *Journal of Mathematical Psychology*, *60*, 58–71. <https://doi.org/10.1016/j.jmp.2014.06.004>