Supplementary material to

Wulff, D. U. & van den Bos, W. (2016). Modeling choices in delay discounting. *Manuscript submitted to Psychological science*.

0. Introduction

In our reanalysis of the data of Ericson, White, Laibson, and Cohen (2015), we evaluated a total of nine model types, one was a baseline model, three belonged to the class of heuristic models, and five belonged to the class of discounted utility models. In the following we first provide the formulas for each of the models and describe how we conducted the model evaluation. We then present the extended results of the model evaluation and some supporting analyses. Finally, we provide the commented code for our model comparison.

1. Models

We formulated the models evaluated by Ericson et al. (2015) to match the implementation in the code of Ericson et al. (2015). The other models were implemented to match the specifications in the original publications. For notation and definitions, see page 14.

Baseline model

Intercept model (BASE):

$$u(LL) - u(SS) = \beta$$

Heuristic models

Intertemporal choice heuristic (ITCH; Ericson et al., 2015):

$$u(LL) - u(SS) = \beta_0 + \beta_1(x_{LL} - x_{SS}) + \beta_2 \frac{x_{LL} - x_{SS}}{x^*} + \beta_3(t_{LL} - t_{SS}) + \beta_4 \frac{t_{LL} - t_{SS}}{t^*}$$

Difference-Ratio-Interest-Finance-Time model (DRIFT; Read, Frederick, & Scholten, 2013)

$$D = \beta_1 z(x_{LL} - x_{SS})$$

$$R = \beta_2 z\left(\frac{x_{LL} - x_{SS}}{x_{SS}}\right)$$

$$I = \beta_3 z\left(\left(\frac{x_{LL}}{x_{SS}}\right)^{\frac{1}{(t_{LL} - t_{SS})}} - 1\right)$$

$$T = \beta_4 z(t_{LL} - t_{SS})$$

$$u(LL) - u(SS) = \beta_1 + D + R + I + T$$

Trade-off model (TRADE; Scholten & Read, 2010)

$$v(x) = \left(\frac{1}{\gamma}\right)\log(1 + \gamma x)$$

$$w(t) = \left(\frac{1}{\tau}\right)\log(1 + \tau x)$$

$$u(LL) - u(SS) = \big(v(x_{LL}) - v(x_{SS})\big) - \beta\big(w(t_{LL}) - w(t_{SS})\big)$$

Discounting models

Exponential discounting model (EXPO; Samuelson, 1937)

$$u(LL) = x_{LL} e^{-k t_{LL}}$$

$$u(SS) = x_{SS} e^{-k t_{SS}}$$

Hyperbolic discounting model (HYPER; Mazur, 1987)

$$u(LL) = x_{LL} \frac{1}{1 + k t_{LL}}$$

$$u(SS) = x_{SS} \frac{1}{1 + k t_{SS}}$$

Hyperboloid model (HYPER2; Myerson & Green, 1995)

$$u(LL) = x_{LL} \left(\frac{1}{1 + k t_{LL}}\right)^m$$

$$u(SS) = x_{SS} \left(\frac{1}{1 + k t_{SS}}\right)^m$$

Quasi-hyperbolic discounting model (QHYPER; Laibson, 1997)

$$u(LL) = \begin{cases} x_{LL}\delta_1^{t_{LL}} & if\ t_{LL} = 0 \\ x_{LL}\delta_1^{t_{LL}}\ \delta_2 & if\ t_{LL} > 0 \end{cases}$$

$$u(SS) = \begin{cases} x_{SS}\delta_1^{t_{SS}} & if\ t_{SS} = 0 \\ x_{SS}\delta_1^{t_{SS}}\ \delta_2 & if\ t_{SS} > 0 \end{cases}$$

Dual-system model (SYSTEM; van den Bos & McClure, 2013)

$$u(LL) = x_{LL}\left(\omega\delta_1^{t_{LL}} + (1-\omega)\delta_2^{t_{LL}}\right)$$

$$u(SS) = x_{SS}\left(\omega\delta_1^{t_{SS}} + (1-\omega)\delta_2^{t_{SS}}\right)$$

2. Model specifications

We evaluated a total of 25 models: three variants for each of the heuristic and discounting models plus the baseline models. The first variant of each model was equivalent to the specification used by Ericson et al. (2015). The second variant of the heuristic models did not include the bias parameter (e.g., β_1 in the ITCH):

$$u(LL) - u(SS) = \left(\beta_1(x_{LL} - x_{SS}) + \beta_2\frac{x_{LL} - x_{SS}}{x^*} + \beta_3(t_{LL} - t_{SS}) + \beta_4\frac{t_{LL} - t_{SS}}{t^*}\right)$$

Removing the bias parameter allowed us to examine the impact of the bias parameter independent of the sensitivity parameter and the increase in model flexibility that may come with the bias parameter. Note that the bias parameter enables the model to adjust its predictions independently of outcome and delay information. The bias parameter could be interpreted as representing pre-evaluative process, such as the application of a strict aspiration level of receiving something now (see Stewart, Reimers, & Harris, 2014; Wulff, Hills, & Hertwig, 2015). For this interpretation, it actually seems advisable to not use the bias as a mere means to connect the model's predictions to empirical data, but to include it into and present it as part of the psychological model.

Next we added a sensitivity parameter to the heuristic models (as replacement for the stripped bias parameter). This changed the ITCH's formulation to

$$u(LL) - u(SS) = \beta_1 \left(\beta_2 (x_{LL} - x_{SS}) + \beta_3 \frac{x_{LL} - x_{SS}}{x^*} + \beta_4 (t_{LL} - t_{SS}) + \beta_5 \frac{t_{LL} - t_{SS}}{t^*} \right)$$

Note that fitting this altered formulation can involve identifiability issues (for the ITCH and the DRIFT models). For instance, $\boldsymbol{\beta} = (\beta_1 = 10, \beta_2 = \beta_3 = \beta_4 = \beta_5 = 1)$ must result in identical predictions as $\boldsymbol{\beta} = (\beta_1 = 1, \beta_2 = \beta_3 = \beta_4 = \beta_5 = 10)$. This issue can in some cases hamper the identification of the likelihood functions' maximum. More importantly, however, it will only concern the interpretability of the model parameters. In support of this, we found that the restriction $\beta_1 = 1$ did not lead to a consistent improvement in model performance as would be expected in the presence of global identifiability problems. The purpose of removing the bias parameter and replacing it with a sensitivity parameter was to equip the heuristic models with the exact same auxiliary assumptions as were used in the original discounting models.

The second variant of the discounting models relied on a power choice rule instead of the exponential choice rule. Choice rules allow the model to predict choices by mapping the models predicted utilities for the sooner smaller and larger later options onto a probability of choosing the larger later option. The exponential choice rule achieves this by passing the weighted difference between the utility of the larger later and the sooner smaller option through a logistic function (see e.g., Rieskamp, 2008):

$$p(choice = LL) = \frac{1}{1 + e^{-\gamma(u(LL) - u(SS))}}$$

The power choice rule achieves this by normalizing the utility of the larger later option by the sum of the utilities of both options (e.g., Stott, 2006)*:*

$$p(choice = LL) = \frac{u(LL)^\gamma}{u(LL)^\gamma + u(SS)^\gamma}$$

Note that both choice rules satisfy Luce's choice axioms (Luce, 1959/2005; Pleskac, 2015). Further, the power choice rule can be specified only for models in which the utilities of the sooner smaller and the larger later option are expressed by independent model formulas,

which is the case for classic discounting models, but not for the heuristic models. Note that the heuristic models predict the difference in utilities, not the utilities themselves. Finally, note that the power choice rule can not easily accommodate negative values. Specifically, the power choice rule runs into problems whenever one of the utilities is assigned a negative value and the other not, as this can result in both negative values and values above one.

The third variant of the discounting models extended the original specification (first variant) by adding a bias parameter. This changed, for instance, the EXPO's formulation to

$$u(LL) = \beta + x_{LL}e^{-kt_{LL}}$$

$$u(SS) = x_{SS}e^{-kt_{SS}}$$

with β being the bias parameter. The purpose of both alterations in the discounting models was, again, to gauge the impact of different auxiliary assumptions on model performance. Note that the list of possible auxiliary assumptions (and their possible combinations) tested here is not exhaustive.

3. Model evaluation

Consistent with Ericson et al. (2015), we first evaluated the models using cross-validation on the aggregate level. To this end we repeated the following three-step procedure 1,000 times:

(i) Sample a subset of 75% percent of all trials (irrespective of participant and condition)

(ii) Fit the models to the data using maximum-likelihood to yield the best fitting parameters $\boldsymbol{\theta}_{ML}$.

(iii) Evaluate model performance by predicting the remaining the 25% of choices using $\boldsymbol{\theta}_{ML}$.

To yield the best fitting parameters, we fitted each model 10 times using different parameter starting values[1]. To evaluate the models, we computed four different criteria or loss functions: proportion of choices correctly predicted ($\pi_{correct}$),[2] median absolute deviation (MAD), mean squared error (MSE), and log-loss (logLoss). The evaluation criteria were computed based on the vector of n choices $\boldsymbol{x} \in \{0,1\}^n$ and the vector of n predicted probabilities $\boldsymbol{p}_x \in [0,1]^n$:

$$\pi_{correct} = \frac{1}{n}\sum_{i=1}^{n} I(x_i, p_{x,i}) \quad with \quad I(x, p_x) = \begin{cases} 1 & if\ x = round(p_x) \\ 0 & if\ x \neq round(p_x) \end{cases}$$

$$MAD = \frac{1}{n}\sum_{i=1}^{n} \left| x_i - p_{x,i} \right|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \left(x_i - p_{x,i} \right)^2$$

$$logLoss = -\frac{1}{n}\sum_{i=1}^{n} x_i log\left(p_{x,i}\right) + (1 - x_i) x_i log\left(1 - p_{x,i}\right)$$

We report the results for each loss function. Note, however, that there are clear theoretical reasons to favor one loss function over another. First, loss functions have to be matched to the nature of the data (Merkle & Steyvers, 2013). Because the aim of probabilistic models of choice is to predict the true probability of choosing, e.g., the larger later option, one should avoid using MAD. In contrast to MSE and logLoss, MAD does not consistently estimate the expectation of the probability distribution, which is the probability of choosing the larger later option, but the distribution's majority class, i.e., the option chosen most often. MAD does therefore not belong to the class of proper scoring rules for probabilistic prediction (Buja, Stuetzle, & Shen 2005). Additionally, it is important to match the loss function used for

[1] To avoid sampling completely inappropriate parameter values, we chose to sample the sets of starting values from the sets of $\boldsymbol{\theta}_{ML}$ obtained by fitting the models to each individual participant. See the supporting analyses in section 4 of this document.

[2] Note that $\pi_{correct}$ is the inverse of the commonly used zero-one loss.

evaluating the models to the loss function used for fitting them. Specifically, the employed loss functions should estimate the same *functional*, e.g., the distribution's expectation (see Gneiting, 2011). This means that it can be appropriate to fit models using maximum-likelihood and then evaluate their performance using mean squared error. However, if one additionally assumes that the true data-generating model is not among the candidate models, which is generally the case, then one should rely on the exact same loss function for fitting and evaluation (Elliott, Ghanem, & Krüger, 2016). For this reason, logLoss is to be considered the most appropriate loss function among the four. For reasons of comparability, we nonetheless present the results for all four loss functions.

The aggregate-level analysis delivered several interesting results (see Table S1). First, consistent with Ericson et al. (2015), we found that the original formulations of the heuristic models dominated the original specifications of the discounting models, with the ITCH emerging as the best model for each loss function. Second, removing the additive bias parameter proved highly detrimental to the performance of the ITCH and the DRIFT, but not the TRADE, when the MAD loss function was used. However, when we applied the more appropriate LogLoss loss function, the pattern was more complex; the performance of the ITCH deteriorated, the performance of the DRIFT improved and again no change for the TRADE. Third, the discounting models benefited strongly from the exponential choice rule being replaced by the power choice rule, with performance levels close to those of the best performing heuristic models (regardless of loss function considered). Fourth, the performance of the heuristic models benefited from the inclusion of a bias parameter, though not to the same extent as from the power choice rule.

The results of the aggregate-analysis support Ericson et al.'s (2015) conclusion in favor of the ITCH, but they also highlight the crucial role of auxiliary assumptions. When models were compared across implementations, the best heuristic and the best discounting model, i.e., the ITCH-1 and the HYPER-2, showed nearly equivalent performance levels.

Critically, the HYPER-2 even slightly outperformed the ITCH-1 when the logLoss.loss function was used.

Table S1

Results of the Aggregate-Level Model Evaluation

		Evaluation criteria		
Model	$\pi_{correct}$	MAD	MSE	logLoss
BASE	.628	.467	.234	.660
Heuristic models				
ITCH-1	**.700**	**.399**	**.201**	.599
ITCH-2 (no bias)	.675	.432	.216	.623
ITCH-3 (sensitivity)	.676	.432	.216	.622
DRIFT-1	.691	.405	.208	.662
DRIFT-2 (no bias)	.678	.438	.218	.630
DRIFT-3 (sensitivity)	.679	.438	.218	.630
TRADE-1	.628	.459	.229	.651
TRADE-2 (no bias)	.628	.459	.230	.651
TRADE-3 (sensitivity)	.661	.443	.220	.632
Discounting models				
EXPO-1	.575	.493	.246	.686
EXPO-2 (pow.)	.688	.416	.206	.605
EXPO-3 (bias, exp)	.628	.465	.232	.657
HYPER-1	.577	.493	.246	.686
HYPER-2 (pow.)	.693	.413	.206	.601
HYPER-3 (bias, exp)	.628	.465	.232	.657
HYPER2-1 (exp)	.569	.493	.246	.686
HYPER2-2 (pow.)	.692	.410	.205	**.598**
HYPER2-3 (bias, exp)	.628	.465	.232	.657
QHYPER-1	.56.6	.493	.246	.686
QHYPER-2 (pow.)	.693	.411	.205	.599
QHYPER-3 (bias, exp)	.628	.465	.232	.657
SYSTEM-1	.567	.493	.246	.686
SYSTEM-2 (pow.)	.693	.410	.205	**.598**
SYSTEM-3 (bias, exp)	.628	.465	.232	.657

Note. Results represent the mean of 10,000 cross-validation repetitions. The best model performance for each of the four criteria is show in bold and underscore.
Legend:
Sensitivity: Heuristic models in which the bias parameter was replaced by a sensitivity parameter (see text)
Neither: Heuristic models with neither bias nor sensitivity parameters.
Pow.: Discounting models in which the exponential choice rule was replaced by a power choice rule.
Bias, exp.: Discounting models in which the sensitivity parameter was replaced by a bias parameter.

We also evaluated the models on the level of the individual participant. To this end, we repeated 10 times the three-step procedure used in the aggregate-analysis for each of the 939 individual participants. We again fitted each model 10 times and evaluated the model under the best-fitting parameter estimates using the same four evaluation criteria as in the aggregate analysis.

The average results of the individual-level analysis presented in Table S2 yielded results that differed in at least four respects from those found for the aggregate-level analysis. First, performance levels for all models were considerably better in the individual-level than in the aggregate-level analysis. This result suggests substantial heterogeneity in the decision-making processes of individuals (see Marewski & Schooler, 2011, and Rieskamp & Otto, 2006). Second, the adjusted discounting models now dominated the heuristic models in three out of four evaluation criteria, including logLoss. This held even when considering the heuristic model's original formulation, i.e., with the bias and without a sensitivity parameter. Third, in the individual level-analysis, the bias parameter no longer improved model performance in a consistent manner. Specifically, when we considered the logLoss criterion the bias parameter now adversely affected the performance of the heuristic models. Finally and relatedly, the four evaluation criteria led to considerably different conclusions, e.g., when considering the ITCH-1. This divergence was attributable to a combination of model flexibility and the punishment for extreme errors. Specifically, MSE and logLoss punish extreme errors (i.e., predicting the wrong choice with high probability), much more strongly than $\pi_{correct}$ and MAD do. The results therefore imply that, for instance, the ITCH-1 committed much more extreme errors than the HYPER-2 did.

Table S2

Average results of the Individual-Level Model Evaluation

	Evaluation criteria

Model	$\pi_{correct}$	MAD	MSE	logLoss
BASE	.697	.371	.196	.576
Heuristic models				
ITCH-1	.786	**.220**	.198	1.306
ITCH-2 (no bias)	.690	.338	.234	1.018
ITCH-3 (sensitivity)	.688	.340	.235	1.014
DRIFT-1	.776	.231	.205	1.352
DRIFT-2 (no bias)	.712	.315	.224	1.078
DRIFT-3 (sensitivity)	.711	.319	.224	1.066
TRADE-1	.702	.347	.200	.675
TRADE-2 (no bias)	.696	.383	.204	.630
TRADE-3 (sensitivity)	.749	.294	.178	.699
Discounting models				
EXPO-1	.740	.313	.180	.718
EXPO-2 (pow.)	.793	.261	.148	**.501**
EXPO-3 (bias, exp)	.697	.359	.200	.674
HYPER-1	.724	.333	.189	.728
HYPER-2 (pow.)	.778	.286	.159	.537
HYPER-3 (bias, exp)	.698	.367	.194	.578
HYPER2-1 (exp)	.743	.314	.180	.717
HYPER2-2 (pow.)	**.801**	.248	**.146**	.515
HYPER2-3 (bias, exp)	.701	.365	.192	.572
QHYPER-1	.728	.324	.199	.876
QHYPER-2 (pow.)	.793	.250	.155	.601
QHYPER-3 (bias, exp)	.703	.342	.213	.873
SYSTEM-1	.747	.307	.180	.747
SYSTEM-2 (pow.)	.799	.246	**.149**	.550
SYSTEM-3 (bias, exp)	.699	.365	.192	.572

Note. Results represent the mean of 9,390 cross-validation repetitions; 10 for each of 939 participants. The best model performance for each of the four criteria is shown in bold and underscored.
Sensitivity: Heuristic models in which the bias parameter was replaced by a sensitivity parameter (see text).
Neither: Heuristic models with neither bias nor sensitivity parameters.
Pow.: Discounting models in which the exponential choice rule was replaced by a power choice rule.
Bias, exp.: Discounting models in which the sensitivity parameter was replaced by a bias parameter.

Altogether, the individual-level analyses appeared to be more strongly affected than the aggregate-level analysis by the choice of auxiliary assumptions and the choice of evaluation criterion. For this reason, it seems difficult to draw conclusions about any single best-performing model. However, when the rather inappropriate MAD was disregarded, then the class of discounting models did seem to consistently outperform the class of heuristic models. That said, the substantial improvement in the performance of all models in the

individual-level analysis relative to the aggregate-level analysis strongly suggests individual differences in the decision-making processes of individuals. Note that this improvement cannot be attributed to the models being fit to fewer data points in the individual-level analysis; on the contrary, this should have hampered model performance due to an increased overfitting of the data.

To follow up on the notion of heterogeneity in decision-making processes, we also explicitly evaluated individual differences. Specifically, we evaluated which model predicted the behavior of each participant best, second best, …, and fifth best. Figure S1 plots the distribution of ranks based on logLoss for each of the models, in a comparison of the original models (model-1; gray bars) and a comparison of the adjusted models (model-2; blue and red bars). The figure shows that even under logLoss, which appears to favor the discounting models, the heuristic models best explained the data of about 16% and 40% of participants, respectively.

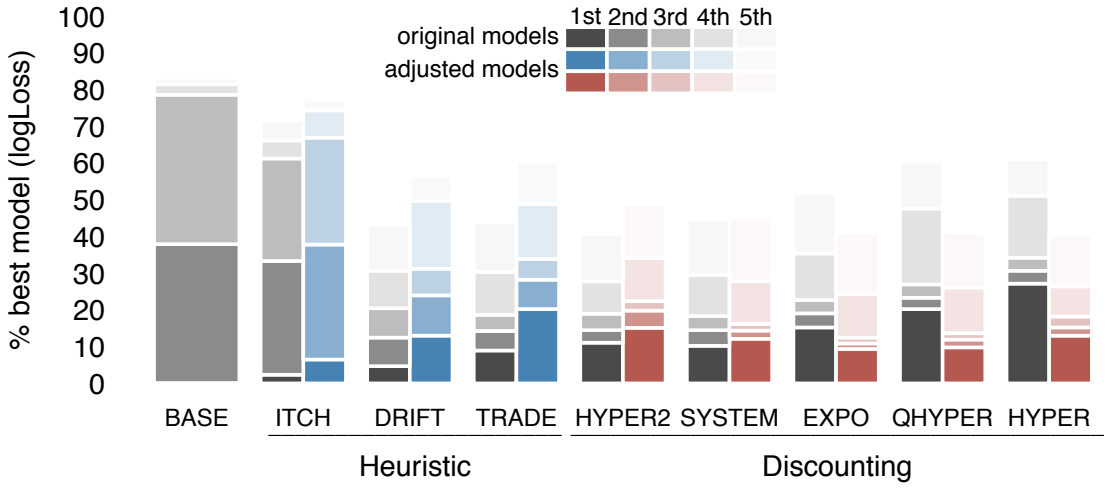

Figure S1. Individual differences in decision-making processes. Plotted are the proportion of times each of the models came out as the best performing, second-best performing, third-best performing, etc., model across all 939 participants. The results are shown for two separate analyses: Grey bars show the comparison of the original models (mode-1), blue and red bars (including the baseline) the comparison of the adjusted models (model-2).

4.Supporting analyses

One possible objection to our model evaluation approach could be that we used only 10 repetitions of the three-step cross-validation procedure for the individual-level analysis, but 1,000 repetitions for the aggregate-level analysis, which could have led to lower precision in the individual-level results than in the aggregate-level results. We have two responses to this objection. First, we chose 10 repetitions in the individual-level analysis to intentionally address the issue that the individual analyses may require more data. Note that 10 repetitions for each of 939 participants yielded nearly 10,000 performance estimates for each of the models, which exceeds the 1,000 estimates in the aggregate-level analysis. Note that Ericson et al. (2015) used only 100 estimates in their analysis of the aggregate data.

Second, we nonetheless analyzed whether an increased number of repetitions (in addition to an increased number of starting values) had any impact on the results and found virtually no effect. Specifically, we reran the original individual-level analysis for three models with 100 cross-validation repetitions and 100 different starting values for three of the original models. The results shown in Table S3 illustrate that an increased number of repetitions and starting values improves model performance in most cases. However, the improvements were very small and did not affect the relative order of the models.

Table S3

Comparison of the Original and Extended Individual-Level Model Analyses

		Evaluation criteria		
Model	$\pi_{correct}$	MAD	MSE	logLoss
Original analysis				
ITCH-1	.786	.220	.198	1.306
EXPO-1	.740	.313	.180	.718
HYPER2-1	.743	.314	.180	.717
Extended analysis				
ITCH-1	.789	.220	.187	1.137
EXPO-1	.740	.311	.180	.723
HYPER2-1	.745	.310	.179	.723

Note. Results represent the mean of 9,390 cross-validation repetitions: 10 for each of 939 participants.

Another potential objection to our approach is that we drew parameter starting values in a pseudo-random fashion (see Footnote 1). Specifically, we determined the best-fitting parameter values for each participant and then used them as a pool of candidate parameter values for both the aggregate-level and the individual-level analyses. To show that this procedure is an advantage rather than a handicap, we compared the original results with results from an individual-level analysis of three models again using 10 repetitions of the cross-validation and 10 different starting values, but this time drawn randomly from the range of permitted values. As shown in Table S4, fitting the models using random start values yielded inferior performances for each of the models, in particular the ITCH and the EXPO. This finding emphasizes that our pseudo-random approach is much more efficient in finding high-performing parameter combinations.

Table S4

Comparison of the Individual-Level Model Analyses implementing Pseudo-Random Versus Random Starting Parameter Values.

		Evaluation criteria		
Model	$\pi_{correct}$	MAD	MSE	logLoss
Original analysis				
ITCH-1	.786	.220	.198	1.306
EXPO-1	.740	.313	.180	.718
HYPER2-1	.743	.314	.180	.717
Extended analysis				
ITCH-1	.671	.329	.328	2.496
EXPO-1	.597	.445	.364	2.373
HYPER2-1	.738	.320	.186	.769

Note. Results represent the mean of 9,390 cross-validation repetitions: 10 for each of 939 participants.

Notation and Definitions

LL - Larger later option

SS - Smaller later option

$u(LL)$ – Subjective utility of larger later option (LL)

$u(SS)$ – Subjective utility of smaller sooner option (LL)

β, γ, τ, k, m – Model parameters

x – Outcome

x^* – Mean outcome

x_{LL}– Outcome of larger later option

x_{SS}– Outcome of smaller sooner option

t – Delay

t^* – Mean delay

t_{LL}– Delay of larger later option

t_{SS}– Delay of smaller sooner option

$v(x_{LL})$– Subjective value of outcome of larger later option

$v(x_{SS})$– Subjective value of outcome of smaller sooner option

$w(x_{LL})$– Subjective weight for delay of larger later option

$w(x_{SS})$– Subjective weight for delay of smaller sooner option

References

Buja, A., Stuetzle, W., and Shen, Y. (2005), "Loss Functions for Binary Class Probability Estimation and Classification: Structure and Applications," manuscript, available at www-stat.wharton.upenn.edu/~buja/.

Elliott, G., Ghanem, D., & Krüger, F. (2016). Forecasting conditional probabilities of binary outcomes under misspecification. *Review of Economics and Statistics*. Advance online publication.

Ericson, K. M. M., White, J. M., Laibson, D., & Cohen, J. D. (2015). Money earlier or later? Simple heuristics explain intertemporal choices better than delay discounting does. *Psychological Science, 26*(6), 826–833. http://dx.doi.org/10.1177/0956797615572232

Gneiting, T. (2011). Making and evaluating point forecasts. *Journal of the American Statistical Association*, *106*(494), 746-762. http://dx.doi.org/10.1198/jasa.2011.r10138

Laibson, D. (1997). Golden eggs and hyperbolic discounting. *The Quarterly Journal of Economics*, 443-477.

Luce, R. D. (2005). *Individual choice behavior: A theoretical analysis*. Mineola, NY: Dover Publications. (Original work published 1959)

Marewski, J. N., & Schooler, L. J. (2011). Cognitive niches: An ecological model of strategy selection. *Psychological Review, 118*(3), 393–437. http://dx.doi.org/10.1037/a0024143

Mazur, J. E. (1987). An adjusting procedure for studying delayed reinforcement. In M. L. Commons, J. E. Mazur, J. A. Nevin, & H. Rachlin (Eds.), *Quantitative analyses of behavior: Vol. 5. The effect of delay and of intervening events on reinforcement value* (pp. 55-73). Hillsdale, NJ: Erlbaum

Merkle, E. C., & Steyvers, M. (2013). Choosing a strictly proper scoring rule. *Decision Analysis, 10*(4), 292–304. http://dx.doi.org/10.1287/deca.2013.0280

Myerson, J., & Green, L. (1995). Discounting of delayed rewards: Models of individual choice. *Journal of the experimental analysis of behavior, 64*(3), 263-276. http://dx.doi.org/10.1901/jeab.1995.64-263

Pleskac, T.J., 2015. Decision and Choice: Luce's Choice Axiom. In: James D. Wright (Ed.), *International Encyclopedia of the Social & Behavioral Sciences* (2nd edition). Oxford: Elsevier.

Read, D., Frederick, S., & Scholten, M. (2013). DRIFT: An analy- sis of outcome framing in intertemporal choice. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *39*, 573–588. http://dx.doi.org/10.2139/ssrn.1933099

Rieskamp, J., & Otto, P. E. (2006). SSL: A theory of how people learn to select strategies. *Journal of Experimental Psychology: General, 135*(2), 207–236. http://dx.doi.org/10.1037/0096-3445.135.2.207.

Rieskamp, J. (2008). The probabilistic nature of preferential choice. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 34*(6), 1446-1465. http://dx.doi.org/10.1037/a0013646

Samuelson, P. A. (1937). A note on measurement of utility. *The Review of Economic Studies*, *4*(2), 155-161. http://dx.doi.org/10.2307/2967612

Scholten, M., & Read, D. (2010). The psychology of intertemporal tradeoffs. *Psychological Review, 117*, 925–944. http://dx.doi.org/10.2139/ssrn.1444094

Stewart, N., Reimers, S., & Harris, A. J. (2014). On the origin of utility, weighting, and discounting functions: How they get their shapes and how to change their shapes. *Management Science, 61*(3), 687–705. http://dx.doi.org/10.1287/mnsc.2013.1853

Stott, H. P. (2006). Cumulative prospect theory's functional menagerie. *Journal of Risk and uncertainty, 32*(2), 101-130. http://dx.doi.org/10.1007/s11166-006-8289-6

van den Bos, W., & McClure, S. M. (2013). Towards a general model of temporal discounting. *Journal of the Experimental Analysis of Behavior*, *99*(1), 58–73. http://dx.doi.org/10.1002/jeab.6

Wulff, D. U., Hills, T. T., & Hertwig, R. (2015). How short-and long-run aspirations impact search and choice in decisions from experience. *Cognition, 144*, 29-37. http://dx.doi.org/10.1016/j.cognition.2015.07.006

```
################################################################################
#
#  © Dirk U. Wulff & Wouter van den Bos
#
#  Analysis code for:
#
#  Wulff, D. U., & van den Bos, W (2016). Modeling Choices in Delay Discounting.
#  Submitted to Psychological Science.
#
#  Contains:
#  0 Preliminaries - Preparatory steps
#  1 Data          - Load and transform data
#  2 Models        - Define models and model related variables
#  3 Helpers       - Define helper functions
#  4 Starts        - Determine start values
#  5 Global        - Run aggregate-level analysis
#  6 Subject       - Run participant-level analysis
#  7 Condition     - Run condition-level analysis
#  8 Test A        - Run test analysis A
#  9 Test B        - Run test analysis B


# ------------------------------------------------------------------------------#
#  0 Preliminaries
#     Clears working directory, loads packages, sets working directory
# ------------------------------------------------------------------------------#

# Clear working directory
rm(list = ls())

# Load snowfall
if(!require(snowfall,quietly = T)) install.packages('snowfall')
require(snowfall)

# Set working directory
DIR = [SPECIFY YOUR ANALYSIS FOLDER HERE]



# ------------------------------------------------------------------------------#
#  1 Data
#     Loads data, transforms variable, and structure data for the upcoming analyses
# ------------------------------------------------------------------------------#

# Load data of Ericson et al (2016)
# Can be downloaded from: https://osf.io/5v9dz/
d  = read.csv(paste0(DIR,'/data/choices.csv'))

# Do variable transformations contained in Ericson et al.'s analysis
d$XStar = (d$X1 + d$X2) / 2
d$TStar = (d$T1 + d$T2) / 2
```

```
d$G = c(scale(d$X2 - d$XStar))
d$R = c(scale((d$X2 - d$X1) / d$XStar))
d$D = c(scale(d$T2 - d$TStar))
d$T = c(scale((d$T2 - d$T1) / d$TStar))
d$DriftD = c(scale(d$X2 - d$X1))
d$DriftR = c(scale((d$X2 - d$X1) / d$X1))
d$DriftI = c(scale((d$X2 / d$X1)^(1 / (d$T2 - d$T1)) - 1))
d$DriftT = c(scale(d$T2 - d$T1))
mx1  = max(d$X1)
mx2  = max(d$X2)
d$X1 = d$X1 / mx1
d$X2 = d$X2 / mx2

# Structure data for upcoming analyses
#   d  : A representation of the original data in which missing values are removed
#   de : list of 1000 repetitions of d (for parallelization)
#   dc : list of 200 repetitions of each of the five condition subsets
#   dl : list of N datasets, one entry per participant
d   = subset(d,!is.na(LaterOptionChosen))
de  = list(); for(i in 1:1000) de[[i]] = d
dc  = list(); for(i in 1:1000) dc[[i]] = subset(d,Condition == i%%5+1)
dl  = split(d,d$Subject)


# ------------------------------------------------------------------------------------#
#  2 Models
#     Defines models, parameter limits, and number of parameters
#     XXXX     specifies the model, limits L to [epsilon, 1-epsilon], and computes LL
#     oXXXX   original model (as specified by Ericson et al.)
#     nXXXX   new model: Heuristics without bias, Discounting with ratio rule
#     n2XXXX new model2: Heuristics with sensitivity, Discounting with bias and expo rule
#     xXXXX.lim        contains parameter limits
# ------------------------------------------------------------------------------------#


BASE.lim = matrix(c(-1e+1,1e+1),nrow=2)
BASE = function(par, d, epsilon, choice = F){
  diff = par
  p.ll = 1/(1+exp(-diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

oITCH.lim = matrix(c(-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf),nrow=2)
oITCH = function(par, d, epsilon, choice = F){
  diff = (par[1]*d$G + par[2]*d$R + par[3]*d$D + par[4]*d$T) + par[5]
  p.ll = 1/(1+exp(-diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
```

```
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

nITCH.lim = matrix(c(-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf,-1e+7,1e+7),nrow=2)
nITCH = function(par, d, epsilon, choice = F){
  diff = (par[1]*d$G + par[2]*d$R + par[3]*d$D + par[4]*d$T)
  p.ll = 1/(1+exp(-par[5]*diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

n2ITCH.lim = matrix(c(-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf),nrow=2)
n2ITCH = function(par, d, epsilon, choice = F){
  diff = (par[1]*d$G + par[2]*d$R + par[3]*d$D + par[4]*d$T)
  p.ll = 1/(1+exp(-diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

oDRIFT.lim = matrix(c(-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf),nrow=2)
oDRIFT = function(par, d, epsilon, choice = F) {
  diff = (par[1]*d$DriftD + par[2]*d$DriftR + par[3]*d$DriftI + par[4]*d$DriftT) + par[5]
  p.ll = 1/(1+exp(-diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
}

nDRIFT.lim = matrix(c(-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf,-1e+7,1e+7),nrow=2)
nDRIFT = function(par, d, epsilon, choice = F) {
  diff = (par[1]*d$DriftD + par[2]*d$DriftR + par[3]*d$DriftI + par[4]*d$DriftT)
  p.ll = 1/(1+exp(-par[5]*diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

n2DRIFT.lim = matrix(c(-Inf,Inf,-Inf,Inf,-Inf,Inf,-Inf,Inf),nrow=2)
```

```
n2DRIFT = function(par, d, epsilon, choice = F) {
  diff = (par[1]*d$DriftD + par[2]*d$DriftR + par[3]*d$DriftI + par[4]*d$DriftT)
  p.ll = 1/(1+exp(-diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

oTRADE.lim = matrix(c(1e-7,1e+7,1e-7,1e+7,1e-7,1e+7,-Inf,Inf),nrow=2)
oTRADE = function(par, d, epsilon, choice = F) {
  cnv = function(x, g) {log(1 + g * x) / g}
  a1 = cnv(d$X2, par[2])
  a2 = cnv(d$X1, par[2])
  a3 = cnv(d$T2, par[3])
  a4 = cnv(d$T1, par[3])
  diff = ((a1 - a2) - par[1] * (a3 - a4)) + par[4]
  p.ll = 1/(1+exp(-diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
}

nTRADE.lim = matrix(c(1e-7,1e+7,1e-7,1e+7,1e-7,1e+7,-1e+7,1e+7),nrow=2)
nTRADE = function(par, d, epsilon, choice = F) {
  cnv = function(x, g) {log(1 + g * x) / g}
  a1 = cnv(d$X2, par[2])
  a2 = cnv(d$X1, par[2])
  a3 = cnv(d$T2, par[3])
  a4 = cnv(d$T1, par[3])
  diff = ((a1 - a2) - par[1] * (a3 - a4))
  p.ll = 1/(1+exp(-par[4]*diff))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
}

n2TRADE.lim = matrix(c(1e-7,1e+7,1e-7,1e+7,1e-7,1e+7),nrow=2)
n2TRADE = function(par, d, epsilon, choice = F) {
  cnv = function(x, g) {log(1 + g * x) / g}
  a1 = cnv(d$X2, par[2])
  a2 = cnv(d$X1, par[2])
  a3 = cnv(d$T2, par[3])
  a4 = cnv(d$T1, par[3])
  diff = ((a1 - a2) - par[1] * (a3 - a4))
  p.ll = 1/(1+exp(-diff))
```

```
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

oEXPO.lim = matrix(c(1e-7,1e+2,1e-7,1e+5),nrow=2)
oEXPO = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * par[1]^d$T2
  u.ss = d$X1 * par[1]^d$T1
  p.ll = 1/(1+exp(-par[2]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
}

nEXPO.lim = matrix(c(1e-3,1e+2,-1e+1,1e+1),nrow=2)
nEXPO = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * par[1]^d$T2
  u.ss = d$X1 * par[1]^d$T1
  p.ll = u.ll**par[2]/(u.ll**par[2] + u.ss**par[2])
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

n2EXPO.lim = matrix(c(1e-7,1e+2,-1e+1,1e+1,-1e+7,1e+7),nrow=2)
n2EXPO = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * par[1]^d$T2 + par[3]
  u.ss = d$X1 * par[1]^d$T1
  p.ll = 1/(1+exp(-par[2]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }


oHYPER.lim = matrix(c(1e-7,1e+2,1e-7,1e+5),nrow=2)
oHYPER = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * (1.0 / (1.0 + par[1] * d$T2))
  u.ss = d$X1 * (1.0 / (1.0 + par[1] * d$T1))
  p.ll = 1/(1+exp(-par[2]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
```

```
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
}

nHYPER.lim = matrix(c(1e-3,1e+2,-1e+1,1e+1),nrow=2)
nHYPER = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * (1.0 / (1.0 + par[1] * d$T2))
  u.ss = d$X1 * (1.0 / (1.0 + par[1] * d$T1))
  p.ll = u.ll**par[2]/(u.ll**par[2] + u.ss**par[2])
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

n2HYPER.lim = matrix(c(1e-7,1e+2,-1e+1,1e+1,-1e+7,1e+7),nrow=2)
n2HYPER = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * (1.0 / (1.0 + par[1] * d$T2)) + par[3]
  u.ss = d$X1 * (1.0 / (1.0 + par[1] * d$T1))
  p.ll = 1/(1+exp(-par[2]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

oHYPER2.lim = matrix(c(1e-7,1e+1,1e-7,1e+1,1e-7,1e+5),nrow=2)
oHYPER2 = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * (1.0 / (1.0 + par[1] * d$T2)**par[2])
  u.ss = d$X1 * (1.0 / (1.0 + par[1] * d$T1)**par[2])
  p.ll = 1/(1+exp(-par[3]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
}

nHYPER2.lim = matrix(c(1e-3,1e+1,1e-7,1e+1,-1e+1,1e+1),nrow=2)
nHYPER2 = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * (1.0 / (1.0 + par[1] * d$T2)**par[2])
  u.ss = d$X1 * (1.0 / (1.0 + par[1] * d$T1)**par[2])
  p.ll = u.ll**par[3]/(u.ll**par[3] + u.ss**par[3])
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
```

```
}

n2HYPER2.lim = matrix(c(1e-7,1e+1,1e-7,1e+1,-1e+1,1e+1,-1e+7,1e+7),nrow=2)
n2HYPER2 = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * (1.0 / (1.0 + par[1] * d$T2)**par[2]) + par[4]
  u.ss = d$X1 * (1.0 / (1.0 + par[1] * d$T1)**par[2])
  p.ll = 1/(1+exp(-par[3]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

oQHYPER.lim = matrix(c(1e-7,1e+7,1e-7,1e+7,1e-7,1e+5),nrow=2)
oQHYPER = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * par[1] ** (d$T2 > 0) * par[2] ** d$T2
  u.ss = d$X1 * par[1] ** (d$T1 > 0) * par[2] ** d$T1
  p.ll = 1/(1+exp(-par[3]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
}

nQHYPER.lim = matrix(c(1e-7,1e+7,1e-3,1e+3,-1e+1,1e+1),nrow=2)
nQHYPER = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * par[1] ** (d$T2 > 0) * par[2] ** d$T2
  u.ss = d$X1 * par[1] ** (d$T1 > 0) * par[2] ** d$T1
  p.ll = u.ll**par[3]/(u.ll**par[3] + u.ss**par[3])
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

n2QHYPER.lim = matrix(c(1e-7,1e+7,1e-7,1e+7,-1e+1,1e+1,-1e+7,1e+7),nrow=2)
n2QHYPER = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * par[1] ** (d$T2 > 0) * par[2] ** d$T2 + par[4]
  u.ss = d$X1 * par[1] ** (d$T1 > 0) * par[2] ** d$T1
  p.ll = 1/(1+exp(-par[3]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

oSYSTEM.lim = matrix(c(1e-7,1,1e-7,1,1e-7,1,1e-7,1e+5),nrow=2)
```

```
oSYSTEM = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * ((par[3] * par[1] ** d$T2) + ((1 - par[3]) * par[2] ** d$T2))
  u.ss = d$X1 * ((par[3] * par[1] ** d$T1) + ((1 - par[3]) * par[2] ** d$T1))
  p.ll = 1/(1+exp(-par[4]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

nSYSTEM.lim = matrix(c(1e-3,1,1e-3,1,1e-7,1,-1e+1,1e+1),nrow=2)
nSYSTEM = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * ((par[3] * par[1] ** d$T2) + ((1 - par[3]) * par[2] ** d$T2))
  u.ss = d$X1 * ((par[3] * par[1] ** d$T1) + ((1 - par[3]) * par[2] ** d$T1))
  p.ll = u.ll**par[4]/(u.ll**par[4] + u.ss**par[4])
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }

n2SYSTEM.lim = matrix(c(1e-7,1,1e-7,1,1e-7,1,-1e+1,1e+1,-1e+7,1e+7),nrow=2)
n2SYSTEM = function(par, d, epsilon, choice = F) {
  u.ll = d$X2 * ((par[3] * par[1] ** d$T2) + ((1 - par[3]) * par[2] ** d$T2)) + par[5]
  u.ss = d$X1 * ((par[3] * par[1] ** d$T1) + ((1 - par[3]) * par[2] ** d$T1))
  p.ll = 1/(1+exp(-par[4]*(u.ll - u.ss)))
  p.ll = epsilon * 0.5 + (1 - epsilon) * p.ll
  if(choice) return(p.ll)
  p.ch = ifelse(as.logical(d$LaterOptionChosen),p.ll,1-p.ll)
  ll   = sum(log(p.ch))
  return(-ll)
  }


# Create vector of model names for later use
ms = c('BASE',
       'oITCH',  'nITCH',   'n2ITCH',
       'oDRIFT', 'nDRIFT',  'n2DRIFT',
       'oTRADE', 'nTRADE',  'n2TRADE',
       'oEXPO',  'nEXPO',   'n2EXPO',
       'oHYPER', 'nHYPER',  'n2HYPER',
       'oHYPER2','nHYPER2', 'n2HYPER2',
       'oQHYPER','nQHYPER', 'n2QHYPER',
       'oSYSTEM','nSYSTEM', 'n2SYSTEM')

# Create vector of the number of parameters for each model for later use
nps = c(1,
        5,5,4,
        5,5,4,
```

```
      4,4,3,
      2,2,3,
      2,2,3,
      3,3,4,
      3,3,4,
      4,4,5)

# Create list of the parameter constrains for each model for later use
lims = list(BASE.lim,
          oITCH.lim,    nITCH.lim,   n2ITCH.lim,
          oDRIFT.lim,   nDRIFT.lim,  n2DRIFT.lim,
          oTRADE.lim,   nTRADE.lim,  n2TRADE.lim,
          oEXPO.lim,    nEXPO.lim,   n2EXPO.lim,
          oHYPER.lim,   nHYPER.lim,  n2HYPER.lim,
          oHYPER2.lim,  nHYPER2.lim, n2HYPER2.lim,
          oQHYPER.lim,  nQHYPER.lim, n2QHYPER.lim,
          oSYSTEM.lim,  nSYSTEM.lim, n2SYSTEM.lim)
names(lims) = ms



# ------------------------------------------------------------------------------------------#
#  3 Helpers
#     Defines helper functions for various aspects of the  analyses
# ------------------------------------------------------------------------------------------#

# draw start values, from individual fits determined in the next section
# I: model string
# O: start values
draw.starts = function(x){start = c(); for(i in 1:ncol(x)) start[i] = sample(x[,i],1); start}

# draw start values at random based on parameter limits
# I: model string
# O: start values
rand.starts = function(m){start = c(); lim = lims[[m]];for(i in 1:ncol(lim)) start[i]=runif(1,
                                                ifelse(lim[1,i] < -1e7,-1e7,lim[1,i]),
                                                ifelse(lim[2,i] >  1e7, 1e7,lim[2,i]));start}
# split data into estimation and prediction set
# I: data d and proportion p
# O: estimation and prediction set
split.data  = function(d,p){
  if(p < 1){
    nchos = nrow(d);
    nfset = floor(nchos * p);
    fset  = sample(1:nchos, nfset)
    return(list(d[fset,],d[(1:nchos)[!(1:nchos)%in%fset],]))
    } else {
    return(list(d,d))
    }
  }
```

```
# Fit function
# Fit models most models using L-BFGS-B and the BASE using Brent
# Repeat the fitting until nfit successful fits have been obtained
# I: to be fitted data dfit, model string m, number of fits nfit
# O: best fit
get.fit = function(dfit, m, nfit){
  fit.v = Inf
  fit.c = 0
  while(fit.c < nfit){
    res.tmp = try(
      if(!m %in% c('BASE')){
      optim(draw.starts(starts[[m]]),
            get(m),
            d = dfit,
            epsilon = .001,
            method = 'L-BFGS-B',
            lower = lims[[m]][1,],upper = lims[[m]][2,])
      } else {
      optim(1,
            get(m),
            d = dfit,
            epsilon = .001,
            method = 'Brent',
            lower = ifelse(m=='BASE',draw.starts(starts[[m]]) - 3,1e-7),
            upper = ifelse(m=='BASE',draw.starts(starts[[m]]) + 3,runif(1,1e-7,1.5e+2)))
    },silent=T)
    if(class(res.tmp) != "try-error"){
      fit.c = fit.c + 1
      if(res.tmp$value < fit.v){
        fit.v = res.tmp$value
        fit.p = res.tmp$par
        }
      } #else print('Problem')
    }
  if(fit.v != -Inf) return(c(fit.v,fit.p)) else return(NULL)
  }


# Result function
# Compute several loss functions: mad, mse, zero-one, logloss
# For checks, also compute extremity abs(p-.5) and whether there NAs
# I: model string
# O: start values
get.results = function(dpre, m, fit) {
  pred = eval(call(m, quote(fit[-1]), quote(dpre), .001, TRUE))
  mad  = mean(abs(pred - dpre$LaterOptionChosen),na.rm=T)
  mse  = mean((pred - dpre$LaterOptionChosen)**2,na.rm=T)
  z_o  = mean(abs(round(pred) - dpre$LaterOptionChosen),na.rm=T)
  logl = sum(log(ifelse(dpre$LaterOptionChosen == 1, pred, 1-pred)))
  extr = mean(abs(pred-.5),na.rm=T)
  nna  = sum(is.na(pred))
```

```
  return(c(mad,mse,z_o,logl,extr,nna))
  }


# ------------------------------------------------------------------------------------------#
#  4 Helpers
#     Determine set of possible start values by fitting each model to each individual
#     Stores start values as text on hard drive
#     Retrieves start values from hard drive
# ------------------------------------------------------------------------------------------#

# Create a new folder to store the start values in
folder = '/start.par/' ; dir.create(paste0(DIR,folder),showWarnings = F)

# Determine starts function
# Fits a model to each individual (paralellized)
# I: string of model m and number of parameters np
# O: Nothing (data is saved in the newly created folder)
determine.starts = function(m,np) {
  sfInit(TRUE, 32)
  sfExport('BASE',
           'oITCH',  'nITCH',   'n2ITCH',
           'oDRIFT', 'nDRIFT',  'n2DRIFT',
           'oTRADE', 'nTRADE',  'n2TRADE',
           'oEXPO',  'nEXPO',   'n2EXPO',
           'oHYPER', 'nHYPER',  'n2HYPER',
           'oHYPER2','nHYPER2', 'n2HYPER2',
           'oQHYPER','nQHYPER', 'n2QHYPER',
           'oSYSTEM','nSYSTEM', 'n2SYSTEM',
           'm','ms','np','lims')
  par = sfClusterApplyLB(dl,function(x) {
    run = 0 ; fit = F
    if('BASE'==m)     starts = 0
    if(m%in%ms[c(2,5,8,13,16,19,22,25)])  starts = c(rep(1,np-1),0)
    if(m%in%ms[c(3,4,6,7,9,10,11,12,14,15,17,18,20,21,23,24)])  starts = rep(1,np)
    while(fit == F){
      run = run + 1
      res.tmp = try(if(run == 1){
        optim(starts,
              get(m),
              d = x,
              epsilon = .001,
              method = ifelse(np > 1, 'L-BFGS-B', 'Brent'),
              lower = lims[[m]][1,],upper = lims[[m]][2,])
        } else {
        optim(rand.starts(m),
              get(m),
              d = x,
              epsilon = .001,
              method = ifelse(np > 1, 'L-BFGS-B', 'Brent'),
              lower = lims[[m]][1,],upper = lims[[m]][2,])
```

```
      })
    if(class(res.tmp) != "try-error") fit = T
    }
 return(res.tmp$par)
 })
 pars = do.call(rbind,par)
 write.table(pars,paste0(DIR,folder,m,'.pars.txt'))
 sfStop()
 }

# Run determine starts function for each model
for(i in 1:length(ms)){
 cat('determine start values for ',ms[i],'\n',sep='')
 determine.starts(ms[i],nps[i])
 }

# Retrieve start values from harddrive
par.nm = paste0(ms,'.pars')
starts = list()
for(i in 1:length(ms)){
 starts[[i]] = read.table(paste0(DIR,folder,par.nm[i],'.txt'))
 }
names(starts) = ms



# ------------------------------------------------------------------------------------#
#  5 Global
#    Run aggregate-level cross-validation analysis
#    Defines run function
#    Executes run function in parallel
# ------------------------------------------------------------------------------------#

# Create folder to store in which the results will be stored
folder = '/cv.global/' ; dir.create(paste0(DIR,folder),showWarnings = F)

# Define run function
# Evaluates a model for a data sets nrep times
# Data is split into estimation and evaluation set according to prop
# Models are evaluated based on the best of nfit fit attempts
# I: data (the entire data set)
# O: Various loss function results (results from results function)
run = function(d){
 nrep   = 1
 nfit   = 10
 prop   = .75
 results = matrix(NA,ncol = 6,nrow = nrep)
 for(i in 1:nrep){
  dsets      = split.data(d, prop)
  fit        = get.fit(dsets[[1]], m, nfit)
  results[i,] = get.results(dsets[[2]], m, fit)
```

```
  }
 return(c(mean(results[,1]),sd(results[,1]),mean(is.null(results[,1])),
        mean(results[,2]),sd(results[,2]),mean(is.null(results[,2])),
        mean(results[,3]),sd(results[,3]),mean(is.null(results[,3])),
        mean(results[,4]),sd(results[,4]),mean(is.null(results[,4])),
        mean(results[,5]),sd(results[,5]),mean(is.null(results[,5])),
        mean(results[,6])))
 }

# Actual analysis
# Executes for all models ms the run function in parallel
# Writes results to harddrive
for(m in ms){
 t = proc.time()[3]
 sfInit(TRUE, 32)
 sfExport('split.data','get.fit','get.results','draw.starts','starts','m','lims',
        'BASE',
        'oITCH',  'nITCH',   'n2ITCH',
        'oDRIFT', 'nDRIFT',  'n2DRIFT',
        'oTRADE', 'nTRADE',  'n2TRADE',
        'oEXPO',  'nEXPO',   'n2EXPO',
        'oHYPER', 'nHYPER',  'n2HYPER',
        'oHYPER2','nHYPER2', 'n2HYPER2',
        'oQHYPER','nQHYPER', 'n2QHYPER',
        'oSYSTEM','nSYSTEM', 'n2SYSTEM')

 res = sfClusterApplyLB(de,run)
 sfStop()
 res = do.call(rbind,res)
 write.table(res,paste0(DIR,folder,'global_',m,'.txt'))
 cat(round(proc.time()[3] - t, 0),'s to complete ',m,'\n',sep='')
 }



# ------------------------------------------------------------------------------------#
#  6 Subject
#     Run subject-level cross-validation analysis
#     Defines run function
#     Executes run function in parallel
# ------------------------------------------------------------------------------------#

# Create folder to store in which the results will be stored
folder = '/cv.subject/' ; dir.create(paste0(DIR,folder),showWarnings = F)

# Define run function
# Evaluates a model for a data sets nrep times
# Data is split into estimation and evaluation set according to prop
# Models are evaluated based on the best of nfit fit attempts
# I: data (of one subject)
# O: Various loss function results (results from results function)
```

```
run = function(d){
  nrep    = 10
  nfit    = 10
  prop    = .75
  results = matrix(NA,ncol = 6,nrow = nrep)
  for(i in 1:nrep){
    dsets       = split.data(d, prop)
    fit         = get.fit(dsets[[1]], m, nfit)
    results[i,] = get.results(dsets[[2]], m, fit)
    }
  return(c(mean(results[,1]),sd(results[,1]),mean(is.null(results[,1])),
           mean(results[,2]),sd(results[,2]),mean(is.null(results[,2])),
           mean(results[,3]),sd(results[,3]),mean(is.null(results[,3])),
           mean(results[,4]),sd(results[,4]),mean(is.null(results[,4])),
           mean(results[,5]),sd(results[,5]),mean(is.null(results[,5])),
           mean(results[,6])))
  }

# Actual analysis
# Executes for all models ms the run function in parallel
# Writes results to harddrive
for(m in ms){
  t = proc.time()[3]
  sfInit(TRUE, 32)
  sfExport('split.data','get.fit','get.results','draw.starts','starts','m','lims',
           'BASE',
           'oITCH',  'nITCH',   'n2ITCH',
           'oDRIFT', 'nDRIFT',  'n2DRIFT',
           'oTRADE', 'nTRADE',  'n2TRADE',
           'oEXPO',  'nEXPO',   'n2EXPO',
           'oHYPER', 'nHYPER',  'n2HYPER',
           'oHYPER2','nHYPER2', 'n2HYPER2',
           'oQHYPER','nQHYPER', 'n2QHYPER',
           'oSYSTEM','nSYSTEM', 'n2SYSTEM')

  res = sfClusterApplyLB(dl,run)
  sfStop()
  res = do.call(rbind,res)
  write.table(res,paste0(DIR,folder,'subject_',m,'.txt'))
  cat(round(proc.time()[3] - t, 0),'s to complete ',m,'\n',sep='')
  }


# ------------------------------------------------------------------------------------#
#  7 Condition
#     Run condition-level cross-validation analysis
#     Defines run function
#     Executes run function in parallel
# ------------------------------------------------------------------------------------#

# Create folder to store in which the results will be stored
```

```
folder = '/cv.condition/' ; dir.create(paste0(DIR,folder),showWarnings = F)

# Define run function
# Evaluates a model for a data sets nrep times
# Data is split into estimation and evaluation set according to prop
# Models are evaluated based on the best of nfit fit attempts
# I: data (of one condition)
# O: Various loss function results (results from results function)
run = function(d){
  nrep    = 1
  nfit    = 10
  prop    = .75
  results = matrix(NA,ncol = 6,nrow = nrep)
  for(i in 1:nrep){
    dsets       = split.data(d, prop)
    fit         = get.fit(dsets[[1]], m, nfit)
    results[i,] = get.results(dsets[[2]], m, fit)
  }
  return(c(mean(results[,1]),sd(results[,1]),mean(is.null(results[,1])),
           mean(results[,2]),sd(results[,2]),mean(is.null(results[,2])),
           mean(results[,3]),sd(results[,3]),mean(is.null(results[,3])),
           mean(results[,4]),sd(results[,4]),mean(is.null(results[,4])),
           mean(results[,5]),sd(results[,5]),mean(is.null(results[,5])),
           mean(results[,6])))
}

# Actual analysis
# Executes for all models ms the run function in parallel
# Writes results to harddrive
for(m in ms){
  t = proc.time()[3]
  sfInit(TRUE, 32)
  sfExport('split.data','get.fit','get.results','draw.starts','starts','m','lims',
           'BASE',
           'oITCH',  'nITCH',   'n2ITCH',
           'oDRIFT', 'nDRIFT',  'n2DRIFT',
           'oTRADE', 'nTRADE',  'n2TRADE',
           'oEXPO',  'nEXPO',   'n2EXPO',
           'oHYPER', 'nHYPER',  'n2HYPER',
           'oHYPER2','nHYPER2', 'n2HYPER2',
           'oQHYPER','nQHYPER', 'n2QHYPER',
           'oSYSTEM','nSYSTEM', 'n2SYSTEM')

  res = sfClusterApplyLB(dc,run)
  sfStop()
  res = do.call(rbind,res)
  write.table(res,paste0(DIR,folder,'global_',m,'.txt'))
  cat(round(proc.time()[3] - t, 0),'s to complete ',m,'\n',sep='')
  }
```

```
# ------------------------------------------------------------------------------------#
#  8 Test A
#     Run test analysis with an extended number of repetitions for three of the models
#     Defines run function
#     Executes run function in parallel
# ------------------------------------------------------------------------------------#

# Create folder to store in which the results will be stored
folder = '/cv.test.ext/' ; dir.create(paste0(DIR,folder),showWarnings = F)

# Define run function
# Evaluates a model for a data sets nrep times - Here extended to 100
# Data is split into estimation and evaluation set according to prop
# Models are evaluated based on the best of nfit fit attempts - Here extended to 100
# I: data (of one subject)
# O: Various loss function results (results from results function)
run = function(d){
  nrep    = 100
  nfit    = 100
  prop    = .75
  results = matrix(NA,ncol = 6,nrow = nrep)
  for(i in 1:nrep){
    dsets       = split.data(d, prop)
    fit         = get.fit(dsets[[1]], m, nfit)
    results[i,] = get.results(dsets[[2]], m, fit)
    }
  return(c(mean(results[,1]),sd(results[,1]),mean(is.null(results[,1])),
           mean(results[,2]),sd(results[,2]),mean(is.null(results[,2])),
           mean(results[,3]),sd(results[,3]),mean(is.null(results[,3])),
           mean(results[,4]),sd(results[,4]),mean(is.null(results[,4])),
           mean(results[,5]),sd(results[,5]),mean(is.null(results[,5])),
           mean(results[,6])))
}

# Actual analysis
# Executes for three models the run function in parallel
# Writes results to harddrive
for(m in c('oITCH','oEXPO','oHYPER2')){
  t = proc.time()[3]
  sfInit(TRUE, 32)
  sfExport('split.data','get.fit','get.results','draw.starts','starts','m','lims',
           'oITCH',
           'oEXPO',
           'oHYPER2'
           )

  res = sfClusterApplyLB(dl,run)
  sfStop()
  res = do.call(rbind,res)
```

```
  write.table(res,paste0(DIR,folder,'subject_',m,'.txt'))
  cat(round(proc.time()[3] - t, 0),'s to complete ',m,'\n',sep='')
  }


# ------------------------------------------------------------------------------------------#
#  9 Test B
#     Run test analysis with random start values for three of the models
#     Defines run function
#     Executes run function in parallel
# ------------------------------------------------------------------------------------------#

# Create folder to store in which the results will be stored
folder = '/cv.test.rand/' ; dir.create(paste0(DIR,folder),showWarnings = F)


# Define new get fit function
# Implements random instead of indvidual level fit start values
# I: estimation data (of one subject), model string m, number of fits nfit
# O: best fit
get.fit = function(dfit, m, nfit){
  fit.v = Inf
  fit.c = 0
  while(fit.c < nfit){
    res.tmp = try(if(!m %in% c('BASE')){
      optim(rand.starts(m),
            get(m),
            d = dfit,
            epsilon = .001,
            method = 'L-BFGS-B',
            lower = lims[[m]][1,],upper = lims[[m]][2,])
      } else {
      optim(1,
            get(m),
            d = dfit,
            epsilon = .001,
            method = 'Brent',
            lower = ifelse(m=='BASE',draw.starts(starts[[m]]) - 3,1e-7),
            upper = ifelse(m=='BASE',draw.starts(starts[[m]]) + 3,runif(1,1e-7,1.5e+2)))
      },silent=T)
    if(class(res.tmp) != "try-error"){
      fit.c = fit.c + 1
      if(res.tmp$value < fit.v){
        fit.v = res.tmp$value
        fit.p = res.tmp$par
        }
      } #else print('Problem')
    }
  if(fit.v != -Inf) return(c(fit.v,fit.p)) else return(NULL)
  }
```

```
# Define run function
# Evaluates a model for a data sets nrep times - Here extended to 100
# Data is split into estimation and evaluation set according to prop
# Models are evaluated based on the best of nfit fit attempts - Here extended to 100
# I: data (the entire data set)
# O: Various loss function results (results from results function)
run = function(d){
  nrep    = 100
  nfit    = 100
  prop    = .75
  results = matrix(NA,ncol = 6,nrow = nrep)
  for(i in 1:nrep){
    dsets       = split.data(d, prop)
    fit         = get.fit(dsets[[1]], m, nfit)
    results[i,] = get.results(dsets[[2]], m, fit)
  }
  return(c(mean(results[,1]),sd(results[,1]),mean(is.null(results[,1])),
           mean(results[,2]),sd(results[,2]),mean(is.null(results[,2])),
           mean(results[,3]),sd(results[,3]),mean(is.null(results[,3])),
           mean(results[,4]),sd(results[,4]),mean(is.null(results[,4])),
           mean(results[,5]),sd(results[,5]),mean(is.null(results[,5])),
           mean(results[,6])))
}


# Actual analysis
# Executes for three models the run function in parallel
# Writes results to harddrive
for(m in c('oITCH','oEXPO','oHYPER2')){
  t = proc.time()[3]
  sfInit(TRUE, 32)
  sfExport('split.data','get.fit','get.results','draw.starts','starts','m','lims',
           'oITCH',
           'oEXPO',
           'oHYPER2'
  )

  res = sfClusterApplyLB(dl,run)
  sfStop()
  res = do.call(rbind,res)
  write.table(res,paste0(DIR,folder,'subject_',m,'.txt'))
  cat(round(proc.time()[3] - t, 0),'s to complete ',m,'\n',sep='')
}
```