Accurately Explaining Exploratory Decisions in a Non-Stationary Bandit Problem: A Recovery Study of the Kalman Filter Model

Alexander O. Savi
University of Amsterdam

Abstract

Daw, O'Doherty, Dayan, Seymour, and Dolan (2006) claim that a model consisting of the Kalman filter and softmax rule can be used to explain human decisions in a non-stationary four-armed bandit task. This paper aims to evaluate whether the model's parameters can be recovered accurately, while keeping the original conditions as much as possible intact. It is shown that three parameters could not be recovered, which indicates serious identification problems. Our conclusion is that the model must be used with caution and suggestions are included to improve recovery.

Keywords: Decision Making, Softmax Rule, Exploitation-Exploration Trade-Off, Identification Problem

Imagine you are a stockbroker. Everyday you face the same decisions over and over again: buy stocks, sell stocks, or wait? The decisions you make are highly uncertain; you are unable to get conclusive information, and outcomes will typically change over time.

How humans decide under uncertainty has been studied for decades, and traditionally focuses on optimal decision-making. A popular paradigm for studying decision-making under uncertainty is reinforcement learning (see Sutton & Barto, 1998, for a comprehensive introduction). Reinforcement learning uses a computational approach to discover how participants (called agents) maximize their cumulative reward (called value).

Alexander Savi, Department of Psychological Methods, University of Amsterdam.

Many thanks go out to Eric-Jan Wagenmakers (University of Amsterdam), Helen Steingroever (University of Amsterdam), and Ruud Wetzels (University of Amsterdam), for their excellent supervision of this internship project and their infinite patience.

Correspondence concerning this report should be addressed to Alexander Savi, University of Amsterdam, Department of Psychological Methods, Weesperplein 4, 1018 XA Amsterdam, The Netherlands. E-mail: o.a.savi@gmail.com

In order to maximize value, both exploration and exploitation are important. Agents need to sufficiently try all options (i.e., explore the different options available to the agent in order to learn which decisions yield the largest value), but also need to sufficiently choose the option with the largest value (i.e., exploit the decisions that maximize the expected value). The exploration-exploitation trade-off is a typical and well-studied dilemma in reinforcement learning (e.g., Audibert, Munos, & Szepesvári, 2009; Avner, Mannor, & Shamir, 2012).

A common means of studying the exploration-exploitation trade-off is the multi-armed bandit problem (e.g., Acuna & Schrater, 2008; Steyvers, Lee, & Wagenmakers, 2009), originally proposed by the mathematician Herbert Robbins (Robbins, 1952). In the multi-armed bandit problem, agents have to choose repeatedly between i slot machines, each with a different and unknown payoff distribution. The agents need to maximize their cumulative reward, by carefully balancing between exploring all slot machines and exploiting the slot machine with the largest expected payoff.

The classical multi-armed bandit problem can be extended to a problem with non-stationary payoff distributions (e.g., Granmo & Berg, 2010; Koulouriotis & Xanthopoulos, 2008). In a non-stationary bandit problem the payoffs not only fluctuate noisily around their respective means, but these means also change randomly and independently from trial to trial (Daw et al., 2006). Daw et al. (2006) proposed a model (hereinafter called the *Kalman filter* model) that aims to disentangle psychological processes that drive decisions of agents on a non-stationary four-armed bandit problem. The *Kalman filter* model employs two different strategies, the Kalman filter and the softmax rule[1].

The Kalman filter is a method for tracking the non-stationary payoff means and variances (Bishop & Welch, 2001). The filter tracks the means and variances by simply updating a prior mean and variance using posterior information. In the *Kalman filter* model, six parameters, which originally lack a conceptual meaning, are involved in tracking the mean and variance of the diffusion process (a comprehensive explanation of the filter and its parameters is given in the following section).

Whereas the Kalman filter is involved in keeping track of the payoff means and variances, the softmax rule governs the agent's choice rules (Daw et al., 2006). The softmax rule employs a single parameter (β), which determines to what degree an agent will explore the different options. The softmax rule guides the suboptimal decision by the expected value of that suboptimal decision (a comprehensive explanation of this rule and its parameter is given in the following section).

Unfortunately, previous efforts of Eric-Jan Wagenmakers and Ruud Wetzels to accurately recover the parameters of the *Kalman filter* model failed (informal simulation studies). The model turned out to be unidentified (i.e., different parameter estimates returned equivalent model fits). The goal of this study is to discover which parameters of the *Kalman filter* model can be recovered accurately.

Explanation of the Non-Stationary Bandit Task and the *Kalman Filter* Model

In this section, both the non-stationary bandit task and *Kalman filter* model are thoroughly discussed. The theory and equations in this section rely heavily on the online

[1] Beside the softmax rule, they also evaluated ϵ-greedy and the softmax rule with an exploration bonus. However in the current study only the unchanged softmax rule is evaluated, as is justified in the following section.

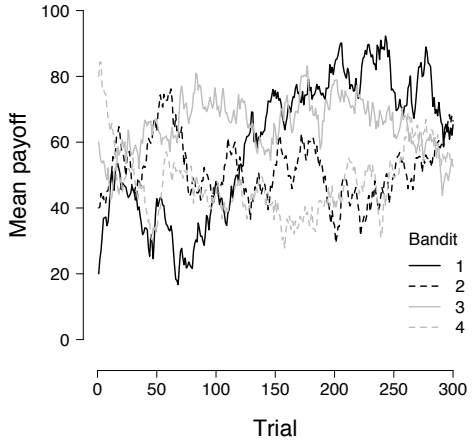


Figure 1. Example of mean payoffs for four bandits on 300 trials (using parameter values from Table 1).

supplementary methods of Daw et al. (2006). The main addition of this section to the understanding of the task and model is a more thorough explanation and discussion of both. In the remainder of this section Daw et al. (2006) will be referred to as the authors.

Non-Stationary Bandit Task

Decaying Gaussian Random Walk. In the non-stationary bandit task, agents choose from i bandits. Each bandit is associated with a payoff that changes over time t. The change in payoffs is independent for each bandit i. An example of the payoff distributions of four bandits is presented in Figure 1. It shows the mean payoff $\mu_{i,t}$ for each bandit on each trial (as opposed to the received payoff, which is drawn from a Gaussian distribution $N(\mu_{i,t}, \sigma_o)$). The depicted task consists of 300 trials. The parameters were set to the values as specified in Table 1.

The received payoffs for each bandit and time-point are drawn from a Gaussian distribution $N(\mu_{i,t}, \sigma_o)$ and rounded to the nearest integer. In order to introduce non-stationarity, a decaying Gaussian random walk is used (Equation 1).

$$\mu_{i,t+1} = \lambda\mu_{i,t} + (1 - \lambda)\theta + \nu \tag{1}$$

Explanation. Decay is provided by a decay parameter λ and decay center θ. The decay parameter λ ranges from 0 to 1 and specifies the rate to which the payoffs decay towards the decay center θ. The Gaussian random walk is established by adding a zero-mean diffusion noise ν, distributed $N(\mu_d, \sigma_d)$, to this decaying $\mu_{i,t}$. If λ is set to 1, the mean payoff on the next trial, $\mu_{i,t+1}$, equals the mean payoff on the current trial, $\mu_{i,t}$, adjusted by some zero-mean noise ν. Decreasing the decay parameter λ allows for more decay towards the decay center θ.

Discussion. As specified in Table 1, the authors set the parameters to $\sigma_o = 4$, $\lambda = 0.9836$, $\theta = 50$, $\mu_d = 0$, and $\sigma_d = 2.8$. However, it is unclear how the mean payoffs on the first trial, $\mu_{i,1}$, are specified. We therefore assume that these are handpicked by the

researchers. Moreover, the payoffs are restricted to lie between 1 and 100 points. Since the authors did not describe how they ensured that the payoffs lie in this range, again we assume that tasks are handpicked by the researchers to fall within this range. The independent Gaussian random walks for each bandit are necessary to ensure non-stationarity, however the authors do not explain why they chose for the payoff means to decay in the first place, and decay to 50 in the second place. In the current study we assume that this decay is a simple measure to ensure that the random walk remains within bounds. Nevertheless, it should be noted that this measure is not strictly necessary to keep the payoffs between 1 and 100 because generating a task with the authors' parameter settings (i.e., a Gaussian random walk with mean zero and only 300 trials) is sufficient to keep the payoffs within bounds most of the time.

Parameters. In sum, the decaying Gaussian random walk comprises of six parameters.

1. The initial payoff means of the first trial $\mu_{i,1}$.
2. The standard deviation of each payoff σ_o.
3. The decay parameter of the diffusion process λ.
4. The decay center of the diffusion process θ.
5. The mean of the diffusion noise μ_d.
6. The standard deviation of the diffusion noise σ_d.

Kalman Filter Model

Kalman Filter. The Kalman filter is the first part of the *Kalman filter* model. This filter tracks the means and variances (uncertainties) of the diffusion process described in the previous section.

The tracking process is initiated by the prior distribution in Equation 2, and followed by two distinct and repeating updating procedures. Note that subscript t for the prior (i.e., $\hat{\mu}^{\text{pre}}_{c_t,t}$) refers to the estimate just before the choice was made on trial t, while the same subscript for the posterior (i.e., $\hat{\mu}^{\text{post}}_{c_t,t}$) refers to the estimate right after the choice was made on trial t. Also note that $\hat{\mu}^{\text{pre}}_{c_t,t}$ and $\hat{\mu}^{\text{post}}_{c_t,t}$ will be referred to as mean trackers, and $\hat{\sigma}^{2\text{post}}_{c_t,t}$ and $\hat{\sigma}^{2\text{pre}}_{c_t,t}$ will be referred to as uncertainty trackers, while the other parameters are equivalent to their counterpart-parameters of the task.

$$N(\hat{\mu}^{\text{pre}}_{i,1}, \hat{\sigma}^{2\text{pre}}_{i,1}) \tag{2}$$

First Update. The first updating procedure only concerns the mean and uncertainty trackers for the chosen bandits, while the second updating procedure concerns the mean and uncertainty trackers for both the chosen and the unchosen bandits. The first updating procedure is discussed first. The mean and uncertainty trackers ($\hat{\mu}^{\text{pre}}_{c_t,t}$ and $\hat{\sigma}^{2\text{pre}}_{c_t,t}$) of the chosen bandit are updated by Equation 3 and 4, respectively.

$$\hat{\mu}^{\text{post}}_{c_t,t} = \hat{\mu}^{\text{pre}}_{c_t,t} + \kappa_t \delta_t \tag{3}$$

$$\hat{\sigma}^{2\text{post}}_{c_t,t} = (1 - \kappa_t)\hat{\sigma}^{2\text{pre}}_{c_t,t} \tag{4}$$

$$\text{with prediction error} \qquad \delta_t = r_t - \hat{\mu}^{\text{pre}}_{c_t,t} \tag{5}$$

$$\text{and learning rate} \qquad \kappa_t = \frac{\hat{\sigma}^{2\text{pre}}_{c_t,t}}{\hat{\sigma}^{2\text{pre}}_{c_t,t} + \hat{\sigma}^2_o} \tag{6}$$

Explanation. The mean tracker $\hat{\mu}^{\text{pre}}_{c_t,t}$ of the chosen bandit (Equation 3) is updated by adding the prediction error δ_t multiplied by the learning rate κ_t. The prediction error δ_t is the difference between the received payoff r_t for the chosen bandit, and the mean tracker $\hat{\mu}^{\text{pre}}_{c_t,t}$ for that choice (Equation 5). Naturally, adding the prediction error δ_t to the mean tracker $\hat{\mu}^{\text{pre}}_{c_t,t}$, gives a more accurate expectation. However before adding the prediction error δ_t, it is multiplied by the learning rate κ_t, see Equation 6. The learning rate (or 'gain') κ_t determines to what extent the prediction error δ_t updates the mean tracker $\hat{\mu}^{\text{pre}}_{c_t,t}$. The update is penalized by a large (squared) expected payoff standard deviation $\hat{\sigma}_o$. If this standard deviation is 0, the prediction error δ_t is multiplied by 1 and thus fully added to the mean tracker $\hat{\mu}^{\text{pre}}_{c_t,t}$. However, the larger this standard deviation is, the more uncertainty about the true payoff mean, and thus the more subtle the update.

The uncertainty tracker $\hat{\sigma}^{2\text{pre}}_{c_t,t}$ of the chosen bandit (Equation 4) is updated by multiplying it by 1 minus the learning rate κ_t. Notice that the uncertainty tracker $\hat{\sigma}^{2\text{post}}_{c_t,t}$ can naturally only diminish; it can only become smaller after observing the payoff for that bandit. The learning rate κ_t determines to what extent the uncertainty tracker $\hat{\sigma}^{2\text{post}}_{c_t,t}$ diminishes. The higher the expected payoff standard deviation $\hat{\sigma}_o$, the smaller κ_t, and thus the less the uncertainty tracker $\hat{\sigma}^{2\text{post}}_{c_t,t}$ diminishes. In other words, a large expected payoff standard deviation $\hat{\sigma}_o$ penalizes the diminishing rate of the uncertainty tracker $\hat{\sigma}^{2\text{post}}_{c_t,t}$ of the chosen bandit.

Second Update. The second update procedure concerns both chosen and unchosen bandits, and is discussed next. The mean and uncertainty trackers ($\hat{\mu}^{\text{post}}_{i,t}$ and $\hat{\sigma}^{2\text{post}}_{i,t}$) of all bandits are updated by Equation 7 and 8, respectively.

$$\hat{\mu}^{\text{pre}}_{i,t+1} = \hat{\lambda}\hat{\mu}^{\text{post}}_{i,t} + (1-\hat{\lambda})\hat{\theta} \tag{7}$$

$$\hat{\sigma}^{2\text{pre}}_{i,t+1} = \hat{\lambda}^2\hat{\sigma}^{2\text{post}}_{i,t} + \hat{\sigma}^2_d \tag{8}$$

Explanation. The mean trackers $\hat{\mu}^{\text{post}}_{i,t}$ of all bandits are updated using the same decay process as described in the task (see Equation 1). Thus, similar to the task, a $\hat{\lambda}$ of 1 results in no decay, while decreasing that expected decay parameter $\hat{\lambda}$ allows for more decay towards the expected decay center $\hat{\theta}$. The uncertainty trackers $\hat{\sigma}^{2\text{post}}_{i,t}$ of all bandits are updated by multiplying them by the squared decay parameter $\hat{\lambda}^2$, and adding and squaring the expected diffusion noise $\hat{\sigma}_d$.

Discussion. The authors indicate that, in general, tracking the uncertainties $\hat{\sigma}^{2\text{post}}_{i,t}$ results in decreased uncertainties for sampled options and increased uncertainties for unsampled options. It seems natural that adding the (squared) standard deviation of the diffusion noise $\hat{\sigma}_d$ accounts for the added diffusion noise ν in the task. By multiplying the

uncertainty trackers $\hat{\sigma}_{i,t}^{2\text{post}}$ with $\hat{\lambda}^2$, the authors reduce the uncertainty if the decay is increased. This seems natural as well, since an increased decay pulls the mean towards the decay center θ, which increases certainty about the mean.

Confusion may arise about the corresponding meanings of the uncertainty tracker $\hat{\sigma}_{c_t,t}^{2\text{pre}}$ and the (squared) estimated payoff standard deviation $\hat{\sigma}_o$. In essence, both represent the agent's uncertainty about the payoff. $\hat{\sigma}_o$ can best be seen as the fixed estimate of the standard deviation of the payoff used to create the task, while $\hat{\sigma}_{c_t,t}^{2\text{pre}}$ can best be seen as a continuously updated estimate of the uncertainty about the payoff observed during the task. Nonetheless, theoretically both are difficult to disentangle.

Softmax Rule. The authors examined three choice rules; ϵ-greedy, the softmax rule, and the softmax rule with an exploration bonus. In the current study only the unchanged softmax rule (Equation 9) is evaluated, as it turned out to perform best (Daw et al., 2006).

$$P_{i,t} = \frac{e^{\beta \hat{\mu}_{i,t}^{\text{pre}}}}{\sum_{j=1}^{n} e^{\beta \hat{\mu}_{j,t}^{\text{pre}}}} \tag{9}$$

Explanation. The softmax rule is a simple rule determining the probability of choosing a certain bandit weighted according to the mean trackers $\hat{\mu}_{i,t}^{\text{pre}}$ of all bandits, and the tendency to explore β. A β of 0 always results in equal probabilities of choosing each bandit on trial t, and thus pure indifference of choosing a certain bandit. This might be called pure exploration, but must not be confused with necessarily choosing a bandit different from the one with the highest mean tracker $\hat{\mu}_{i,t}^{\text{pre}}$, as it has an equal chance to be chosen. As β increases to infinity it moves towards pure exploitation, thus choosing the bandit with the highest mean tracker $\hat{\mu}_{i,t}^{\text{pre}}$.

Discussion. The choice for the softmax rule in this study is purely pragmatic. The ϵ-greedy rule assigns equal probabilities to all bandits when it explores the different bandits. The softmax rule is thus preferred to the ϵ-greedy rule if some of the bandits are associated with really small payoffs (Sutton & Barto, 1998). As this is not necessarily the case in the non-stationary bandit task, this does not always count as a valid reason. Nonetheless, the softmax rule weights the probabilities according to the estimated values which seems to be more natural.

Parameters. In sum, the *Kalman filter* model comprises seven parameters. A meaningful explanation is sought for each of the parameters.

1. The initial mean trackers of the first trial $\hat{\mu}_{i,1}^{\text{pre}}$. This parameter represents the agent's expectation of what the payoff will be for each bandit on the first trial given that the agent has no prior information on the distribution of the payoffs, and assumed that the agent is told beforehand what the range of payoffs is. Thus, given that the agent knows that the payoffs lie between 1 and 100, it is a more or less wild guess within that area.

2. The initial uncertainty trackers of the first trial $\hat{\sigma}_{i,1}^{2\,\text{pre}}$. This parameter represents the agent's uncertainty about what the payoff will be for each bandit on the first trial.

3. The estimated standard deviation of the payoff $\hat{\sigma_o}$. This parameter represents the agent's expectation about the variance in payoffs. Note the difference with $\hat{\sigma}_{i,1}^{2\,\text{pre}}$. Daw et al. (2006) fix this parameter to prevent model degeneracy. Although the authors do not explain what they mean with model degeneracy, we assume they refer to a problem with the identification of the parameter.

4. The estimated decay parameter $\hat{\lambda}$. This parameter represents the degree to which agents remember their tracked estimates of the payoff means. The lower this parameter, the faster agents forget, thus the more their tracked estimates of the payoff means deviate to the decay center.

5. The estimated decay center $\hat{\theta}$. This parameter represents the estimated payoff value agents default to if they forget their tracked estimate of the payoff means.

6. The estimated standard deviation of the diffusion noise $\hat{\sigma_d}$. This parameter represents the agent's estimate of the degree to which the estimated mean payoffs diffuse.

7. The exploration parameter β. This parameter represents the degree to which agents explore suboptimal choices, and, vice versa, exploit optimal choices.

Implementation of the Task and Model in R

After having discussed both the non-stationary bandit task and *Kalman filter* model, we will now address their implementation. The task and model were implemented using the free and open-source statistical software R 3.0.0 (`www.r-project.org`) in conjunction with RStudio (`www.rstudio.com`), an IDE for R. The R-scripts can be found in Appendix A and B and the author's website (`www.alexandersavi.nl`), and are made available under GPL$\geq$ 3.

The task was implemented as described above. Payoffs were rounded to the nearest integer; halves were rounded downwards. The payoff range was restricted between 1 and 100 by simulating a new task until it satisfied the range condition. $\mu_{i,1}$ were selected by the researcher.

In contrast to Daw et al. (2006) who used two sessions of 150 trials, in the current study a single session of 300 trials was used. Moreover, the authors used three different payoff schemes, while in the current study a single payoff scheme is used for each of the 14 synthetic agents. Table 1 shows the generative values of the task parameters used in this study. With the exception of $\mu_{i,1}$, these values correspond to the values used by Daw et al. (2006).

Table 1
Generative Values of the Task Parameters

	σ_o	λ	θ	μ_d	σ_d	$\mu_{i,1}$
Value	4	.9836	50	0	2.8	$\{20, 40, 60, 80\}$

Note. The generated tasks consist of 4 bandits and 300 trials.

Both the Kalman filter and softmax rule were also implemented as described above. Table 2 shows the generative values of the model parameters used in this study. These values correspond to the obtained estimated values using the softmax rule by Daw et al. (2006). Each synthetic agent was assigned an individual β parameter; for each agent this parameter was set to the mean obtained estimated β value in Daw et al. (2006). We used as many synthetic agents as participated in Daw et al. (2006) (i.e., 14).

Table 2
Generative Values of the Kalman Filter and Softmax Rule Parameters

	β	$\hat{\lambda}$	$\hat{\theta}$	$\hat{\sigma_d}$	$\hat{\sigma_o}$	$\hat{\mu}_{i,1}^{\text{pre}}$	$\hat{\sigma}_{i,1}^{2\,\text{pre}}$
Value	.122	.924	50.5	51.3	4	85.7	4.61

Note. The simulated data consists of 14 synthetic agents.

Maximum Likelihood Estimation

In this section, the methods to estimate the parameters of the *Kalman filter* model are discussed. Subsequently, the accuracy to which both the individual parameters of the model and the full model can be recovered will be presented.

To estimate the parameters of the *Kalman filter* model, Daw et al. (2006) searched for those parameter values that minimize the discrepancy between the agents' data and predicted data. This discrepancy is measured by maximizing the likelihood (L). L is obtained by collecting the model's estimated probabilities (the results of the softmax rule) for the chosen bandits as observed in the data. Multiplying these probabilities gives the product below (Equation 10). The parameter combination with the highest product has the lowest discrepancy between the predicted choices and actual choices, and thus the best model fit. Namely, this parameter combination assigns the overall highest probabilities to the bandits that were chosen in the true data, and thus gives rise to data that comes closest to the true data.

$$L = \prod_s \prod_t P_{c_{s,t},t} \tag{10}$$

As is customary, in the end the negative log-likelihood ($-LL$) is minimized. Since the logarithm of a product is the sum of the logarithms of the multiplicators, minimization of $-LL$ is obtained by taking the sum of the logarithms of the discussed probabilities, and multiplying the result by -1 (Equation 11).

$$-LL = -1 \sum_s \sum_t \log P_{c_{s,t},t} \tag{11}$$

The authors use "a combination of nonlinear optimization algorithms (Matlab optimization toolbox)"[2] to find the smallest discrepancies (Daw et al., 2006). Also, they use a not further explicated method to search for different starting locations, and assume all parameters except for β to be equal across agents (as, according to the authors, is standard in similar analyses), while β was estimated for each agent separately. Finally they conducted an alternative analysis fitting all parameters within agents. This was done to see whether their initial results were biased. Table 3 shows the generative values and parameter estimates from Daw et al. (2006).

The authors argue that most of the estimated parameters are similar to the generative values, that is, the parameters are estimated accurately, except for the inflated $\hat{\sigma_d}$. They assume that the failure to estimate $\hat{\sigma_d}$ may be due to agents overestimating the speed of diffusion in the payoffs. They further argue that large values of $\hat{\sigma_d}$ induce high learning

[2] `www.cs.bris.ac.uk/home/rafal/rltoolbox/index.html`

Table 3
Generative Values and Predicted/Fixed Parameter Values in the Study by Daw, O'Doherty, Dayan, Seymour, and Dolan (2006)

Parameter	Generative value	Original estimate
β		$.122 \pm .0547$
$\hat{\lambda}$	.9836	.924
$\hat{\theta}$	50	50.5
$\hat{\sigma_d}$	2.8	51.3
$\hat{\sigma_o}$	4	4 (fixed)
$\hat{\mu}_{i,1}^{\text{pre}}$		85.7
$\hat{\sigma}_{i,1}^{2\,\text{pre}}$		4.61

rates κ_t, and that this thus indicates that agents are maybe too sensitive for the most recent retrieved payoffs. It should be noted that this inflation is mostly noticeable in the softmax results.

Implementation of Maximum Likelihood Estimation in R

Now the implementation of the discrepancy measure and the choice of an optimizer in the current study is discussed. The discrepancy measure was implemented as described above. The R-script can be found in Appendix C and the author's website (`www.alexandersavi.nl`), and is made available under GPL$\geq$ 3.

To find a suitable optimizer in R, we compared the accuracy in recovering a single parameter and the required time of three popular optimizers. The best performing optimizer was chosen for the remainder of this study. The three optimizers are `psoptim()` (standard PSO 2007, package `pso`), `optim()` (method L-BFGS-B, package `stats`), and `nlminb()` (package `stats`), and are compared with respect to the recovery of $\hat{\theta}$. $\hat{\theta}$ was used since informal recovery studies showed that this parameter can be accurately recovered. One important advantage of `psoptim()` to the other optimizers is that it uses a swarm of starting locations (called particles) that communicate their location and corresponding likelihood, and use that information to improve optimization (Kennedy, 2010). `Psoptim()` defaults to a swarm size of 12 particles in the single parameter case. The other optimizers were ran 12 times with starting locations evenly distributed across the parameter space (bounded between 0 and 100), and the parameter estimate with the smallest discrepancy value was chosen. In order to increase the speed of `psoptim()`, the maximum number of iterations was set to 30, after which discrepancies did not seem to decrease any further. Other settings were left to their defaults. The performance of the three optimizers is summarized in Table 4. From the table, it is evident that the recovered parameter value and the corresponding discrepancy are identical for the three optimizers. Even though `psoptim()` requires more computational time to estimate the unknown parameter $\hat{\theta}$, we decided to use `psoptim()` in the remainder of this article because we expect it to perform more accurately than its competitors if confronted with estimating several parameters (informal results).

Table 5 and 6 summarize all important parameters and differences between the study by Daw et al. (2006) and the current study. The parameter estimates by Daw et al. (2006)

Table 4
Recovered Parameter Values, Discrepancy Values, and Speed of Evaluated Optimizers

Function (package)	$\hat{\theta}$	Discrepancy	Speed (in sec.)
True parameter value	50.5		
psoptim() (pso)	52.086	312.372	47.72
optim() (stats), L-BFGS-B	52.086[a]	312.372[a]	33.69
nlminb() (stats)	52.086[a]	312.372[a]	28.86

[a]Recovered parameter values and discrepancy values did not differ for different starting locations.

(see Table 3) were used to generate the synthetic data in the current study.

Table 5
Summary of Generative Parameters for the Non-Stationary Bandit Task of Both Daw, O'Doherty, Dayan, Seymour, and Dolan (2006) and Current Study

		Original	Current
agents	# agents	14	14
	type	flesh and blood	synthetic
task	# tasks	3	1
	# bandits	4	4
	# trials	2×150	1×300
	μ_d	0	0
	λ	.9836	.9836
	θ	50	50
	σ_d	2.8	2.8
	σ_o	4	4
	$\mu_{i,1}$	unknown	$\{20, 40, 60, 80\}$

Recovery of the Individual Parameters of the *Kalman Filter* Model

This section presents the accuracy of the *Kalman filter* model in recovering its individual parameters separately. That is, we estimate a single parameter and fix all remaining parameters to their true values. The recovery of each parameter except for β will have one free parameter, whereas the recovery of β will have 14 free parameters (one for each agent). For each parameter, 100 tasks and data sets were simulated. `Psoptim()` was set to the settings as described in the previous section. With respect to the recovery of β, `psoptim()` defaults to 17 particles in the 14 parameters case, and the maximum number of iterations was increased to 250, after which discrepancies did not seem to decrease any further. Table 7 shows the true parameter values and the corresponding parameter bounds.

Figure 2 - 5 show for each parameter separately the results of the maximum likelihood estimation. The left panels show kernel densities of the maximum likelihood estimates of 100 synthetic data sets. These figures show the relative frequency of recovering the value on the x-axis. The dotted lines indicate the true parameter values. Ideally, the figures are unimodal

Table 6
Summary of Recovered Parameters of the Kalman filter *Model of Daw, O'Doherty, Dayan, Seymour, and Dolan (2006) and Generative Parameters for Synthetic Agents of the Current Study*

		Original recoveries	Generative for current study
model	$\hat{\lambda}$	.924	.924
	$\hat{\theta}$	50.5	50.5
	$\hat{\sigma_d}$	51.3	51.3
	$\hat{\sigma_o}$	4 (fixed)	4
	$\hat{\mu}_{i,1}^{\text{pre}}$	85.7	85.7
	$\hat{\sigma}_{i,1}^{2\,\text{pre}}$	4.61	4.61
	β	$.122 \pm .0547$	$.122 \pm .0000$
recovery	software	Matlab	R 3.0.0
	optimizer	Matlab toolbox[a]	psoptim()
	starting locations	search[a]	psoptim() default
	# free parameters	19[b]	depends on recoveries[c]

[a] `www.cs.bris.ac.uk/home/rafal/rltoolbox/index.html`.
[b] All parameters except for β were assumed to be equal across agents, while β was recovered for each agent separately. $\hat{\sigma_o}$ was fixed to its generative value.
[c] The number of free parameters is indicated in the running text where applicable.

Table 7
True Parameter Value and Corresponding Lower and Upper Bounds (Psoptim() Bounds Between Brackets)

	β	$\hat{\lambda}$	$\hat{\theta}$	$\hat{\sigma_d}$	$\hat{\sigma_o}$	$\hat{\mu}_{i,1}^{\text{pre}}$	$\hat{\sigma}_{i,1}^{2\,\text{pre}}$
True value	.122	.924	50.5	51.3	4	85.7	4.61
Lower bound	0	0	1	0	0	1	0
Upper bound	∞(7[a])	1	100	∞(100[b])	∞(100[c])	100	∞(100[b])

[a] An upper bound above 7 resulted in numerical/computational problems. Nonetheless, the most distinguishing range of β is roughly between 0 and 5, depending on the mean trackers $\hat{\mu}_{i,t}^{\text{post}}$.
[b] Estimates of $\hat{\sigma_d}$ and $\hat{\sigma}_{i,1}^{2\,\text{pre}}$ typically did not exceed 100, or lied at the upper bound.
[c] Estimates of $\hat{\sigma_o}$ did not exceed 100.

and very concentrated around the true parameter value (i.e., unbiased and low variance). However, as these figures do not show the discrepancy measures for each of the likelihood estimates, we show these discrepancies in the right panel. Ideally, the smallest discrepancies should be near the true parameter value (dashed line). Additionally, the bottom panels of Figure 5 show likelihood functions of $\hat{\sigma_d}$ of which the likelihood estimates do not converge. These figures show the likelihood estimates for a range of parameter values. Ideally, this function is convex, with the minimum at the true parameter value (dashed line).

Recovery of $\hat{\theta}$. Figure 2(a) shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of $\hat{\theta}$ typically lie in between 48 and 52.

Recovery of $\hat{\lambda}$. Figure 2(c) shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of $\hat{\lambda}$ typically lie in between .915 and .935.

Recovery of $\hat{\sigma_o}$. Figure 3(a) shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of $\hat{\sigma_o}$ typically lie in between 2.5 and 7.5, with a few estimates at more extreme values up to 25.

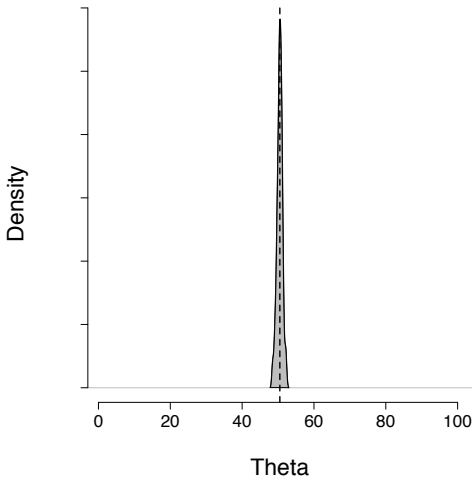
Recovery of β. Figure 3(c) shows the results for the arithmetic mean of the 14 synthetic agents. It shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of β typically lie in between .0 and .3, with a single estimate at the upper bound of 7.

Recovery of $\hat{\mu}_{i,1}^{\mathbf{pre}}$. Figure 4(a) shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. However, there is relatively high uncertainty, indicated by the relatively big spread of the distribution. Estimates of $\hat{\mu}_{i,1}^{\text{pre}}$ typically lie in between 75 and 100.

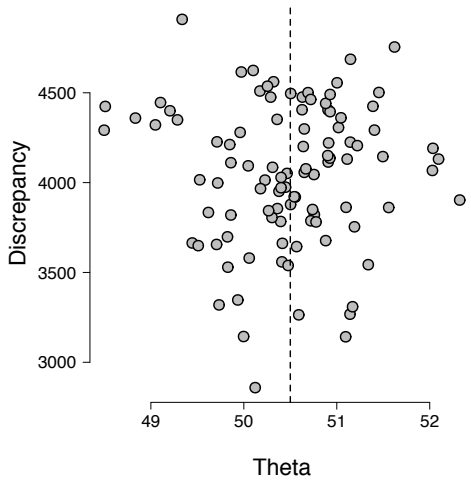
Recovery of $\hat{\sigma}_{i,1}^{2\,\mathbf{pre}}$. Figure 4(c) shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of $\hat{\sigma}_{i,1}^{2\,\text{pre}}$ typically lie in between 0 and 10, and it is interesting to note the small concentration of estimates at exactly 0.

Recovery of $\hat{\sigma_d}$. Figure 5(a) shows two modes which are on both sides of the true parameter value, the likelihood estimates thus seem extremely biased. Naturally, there also is extremely high uncertainty around the true parameter value, indicated by the large spread of the distribution (i.e., the recovered parameter values cover the entire parameter range). Figure 5(c) shows why there is mode at 100: some estimates are simply limited by the upper bound. Estimates of $\hat{\sigma_d}$ thus typically lie in between 0 and 40, or at the upper bound.

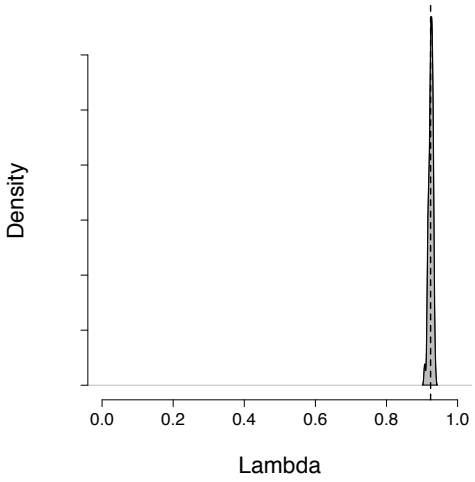
Figure 5(c) and 5(d) show two typical likelihood functions of $\hat{\sigma_d}$. Although at first sight no significant differences can be observed, a more closely observation shows a small minimum around 4 in Figure 5(d), while no such minimum is found in Figure 5(c). This slight difference explains the two modes in Figure 5(a).



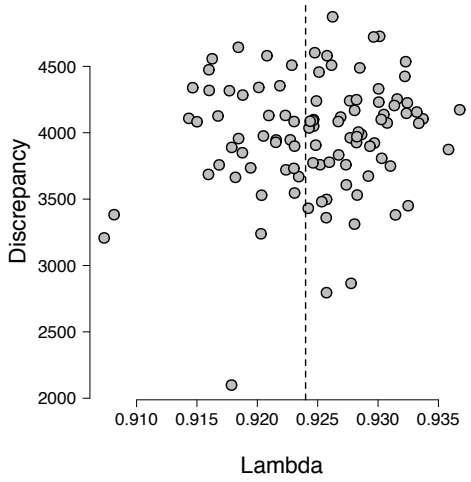
(a) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.



(b) Discrepancy measures associated with the likelihood estimates in Figure 2(a).

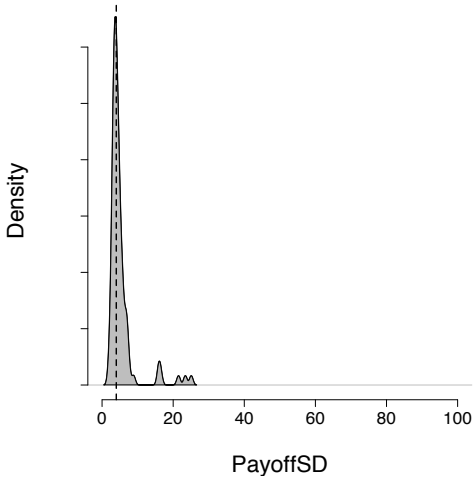


(c) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.

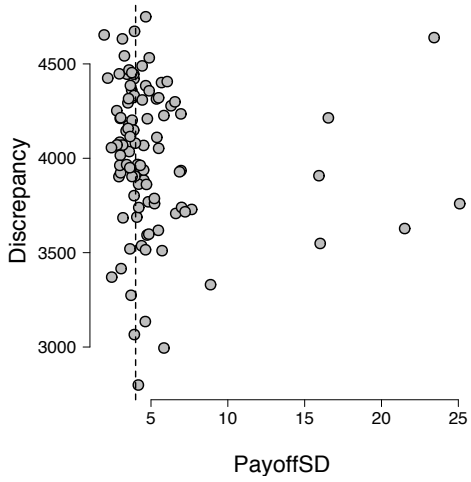


(d) Discrepancy measures associated with the likelihood estimates in Figure 2(c).

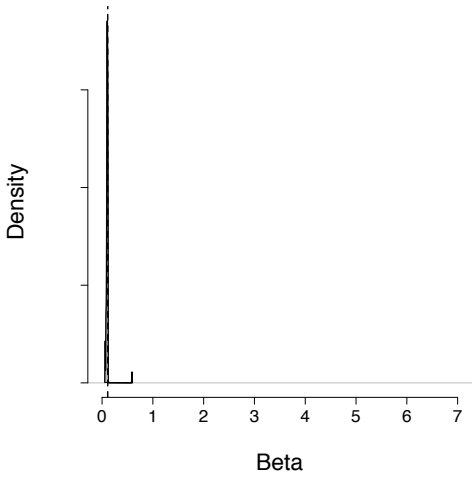
Figure 2. Kernel densities of the maximum likelihood estimates of 100 synthetic data sets (left panel). Discrepancy measures associated with the corresponding likelihood estimates (right panel).



(a) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.

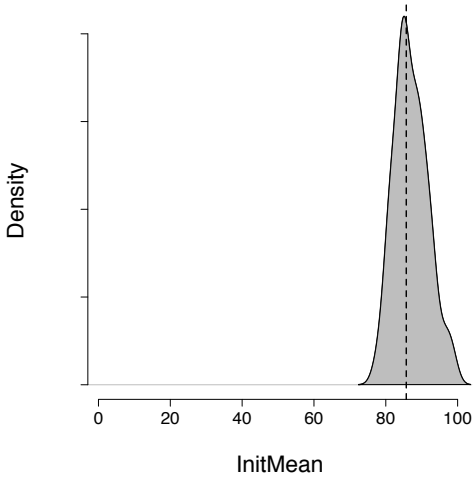


(b) Discrepancy measures associated with the likelihood estimates in Figure 3(a).

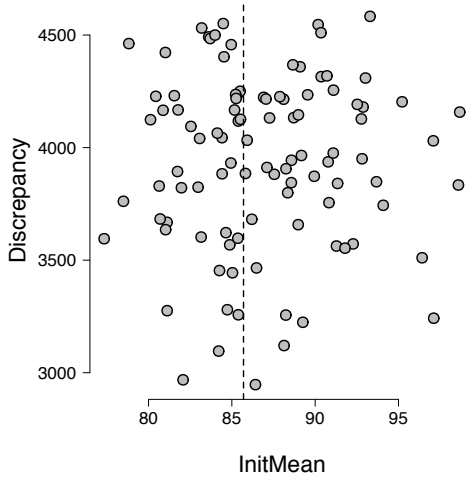


(c) Kernel density of the arithmetic mean of the 14 recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.

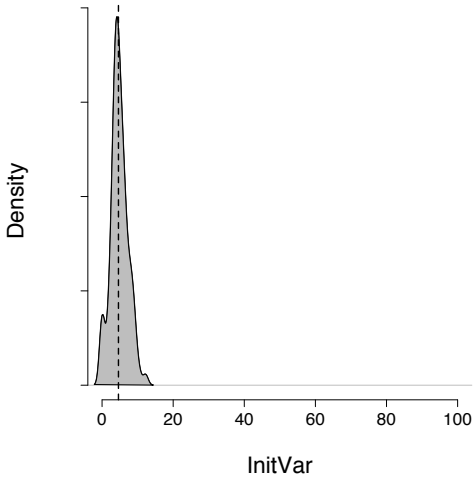
Figure 3. Kernel densities of the maximum likelihood estimates of 100 synthetic data sets (left panel). Discrepancy measures associated with the corresponding likelihood estimates (right panel).



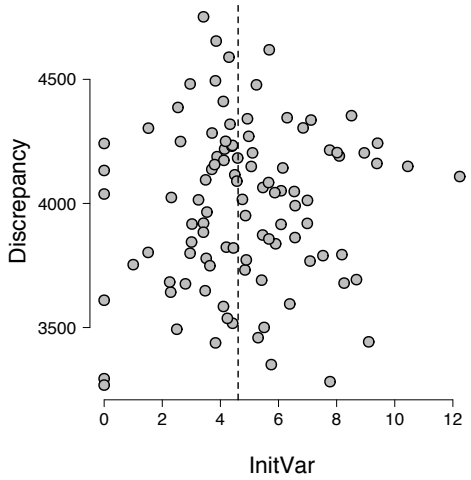
(a) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.



(b) Discrepancy measures associated with the likelihood estimates in Figure 4(a).

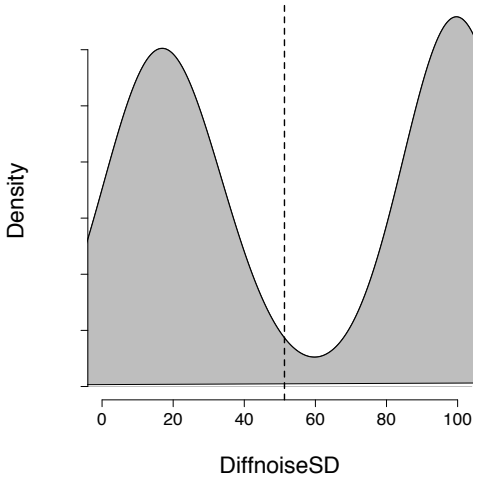


(c) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.

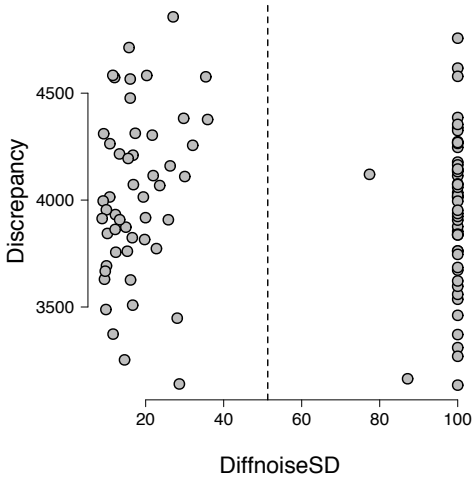


(d) Discrepancy measures associated with the likelihood estimates in Figure 4(c).

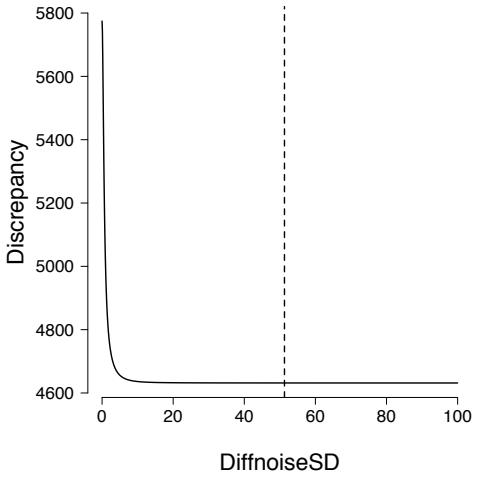
Figure 4. Kernel densities of the maximum likelihood estimates of 100 synthetic data sets (left panel). Discrepancy measures associated with the corresponding likelihood estimates (right panel).



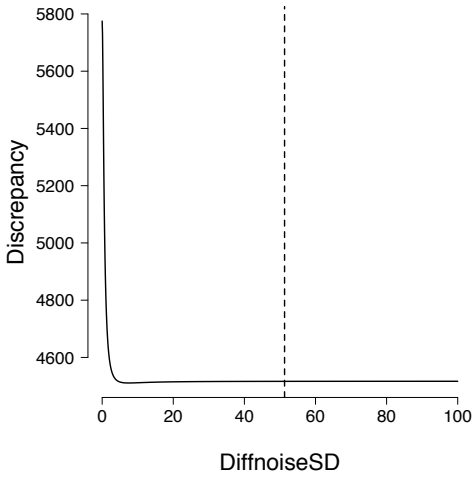
(a) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.



(b) Discrepancy measures associated with the likelihood estimates in Figure 5(a).



(c) Typical likelihood function of $\hat{\sigma_d}$.



(d) Typical likelihood function of $\hat{\sigma_d}$.

Figure 5. Kernel densities of the maximum likelihood estimates of 100 synthetic data sets (upper left panel). Discrepancy measures associated with the corresponding likelihood estimates (upper right panel). Typical likelihood functions of $\hat{\sigma_d}$ of which the likelihood estimates do not converge (bottom panels).

Recovery of the Individual Parameters of the *Kalman Filter* Model for Single Synthetic Agents

Additionally, parameter recoveries are obtained for a single agent's data (not shown). The most notable differences are shortly discussed. The recoveries of $\hat{\theta}$, $\hat{\lambda}$, and $\hat{\sigma_o}$ are equivalent, however naturally show greater uncertainty, which is most noticeable for $\hat{\sigma_o}$. The recoveries of $\hat{\mu}_{i,1}^{\text{pre}}$ and $\hat{\sigma}_{i,1}^{2\,\text{pre}}$ not only show greater uncertainty, but also have a tendency towards the bounds. The results for $\hat{\sigma_d}$ are equally flawed as for the data from 14 agents. Finally the recoveries of β naturally are equivalent, since in the data from 14 agents they were recovered for each agent as well. However, the results for β show less uncertainty.

Recovery of the Full *Kalman Filter* Model

This section presents the accuracy of the *Kalman filter* model in recovering all of its parameters at once (rather than each parameter separately). Note however that the estimated standard deviation of the diffusion noise $\hat{\sigma_d}$ is fixed to its true value, since it could not be recovered in the previous section. Also note that Daw et al. (2006) do recover $\hat{\sigma_d}$, but fix $\hat{\sigma_o}$ to its generative value, which is in its turn recovered accurately in this study. The full model consists of 19 free parameters; 5 for the parameters except β, and 14 for the β of each agent. 100 tasks and data sets were simulated. Table 7 shows the true parameter values and the corresponding parameter bounds.

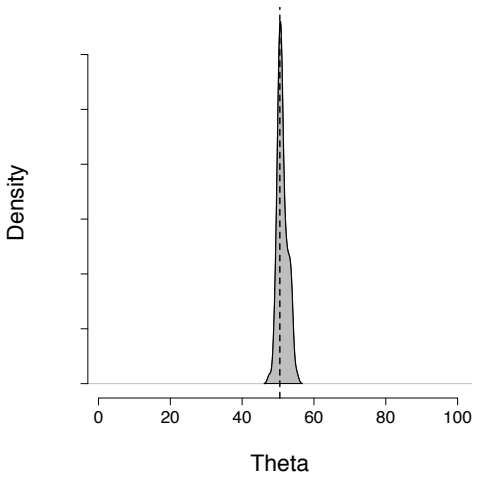
`Psoptim()` defaults to a swarm size of 18 particles in the 19 parameters case. Despite of `psoptim()`'s low speed, its maximum number of iterations was set to 1500 to allow for proper recovery (after 1200 iterations discrepancies did not seem to decrease any further). Other settings were left to their defaults. Figure 6 - 7 show for each parameter separately the results of the maximum likelihood estimation.

Recovery of $\hat{\theta}$. Figure 6(a) shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of $\hat{\theta}$ typically lie in between 49 and 54.

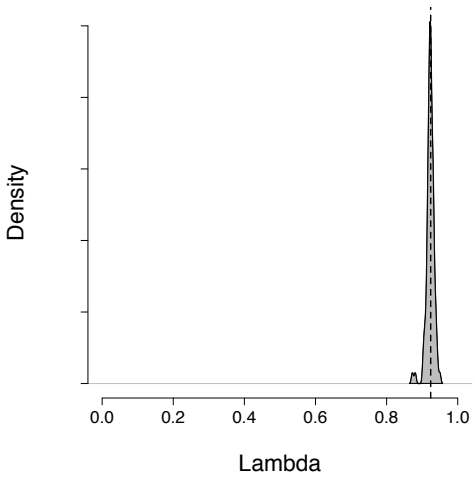
Recovery of $\hat{\lambda}$. Figure 6(b) shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of $\hat{\lambda}$ typically lie in between .90 and .94.

Recovery of $\hat{\sigma_o}$. Figure 6(c) shows three modes of which one is at the true parameter value. One bigger mode is near the true parameter value, whereas one smaller mode is at the upper bound. The likelihood estimates thus seem really biased. Naturally, there also is high uncertainty around the true parameter value, indicated by the large spread of the distribution. Estimates of $\hat{\sigma_o}$ typically lie in between 0 and 25, with a few estimates at the upper bound of 100.

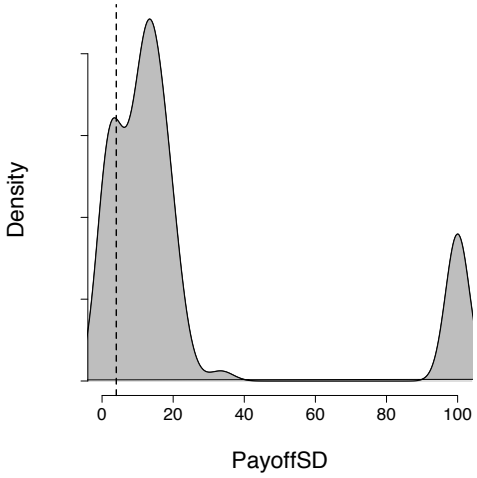
Recovery of β. Figure 6(d) shows the results for the arithmetic mean of the 14 synthetic agents. It shows that the mode of the distribution is at the true parameter value, the likelihood estimates thus seem relatively unbiased. Moreover, there is relatively low uncertainty, indicated by the small spread of the distribution. Estimates of β typically lie in between .10 and .15.



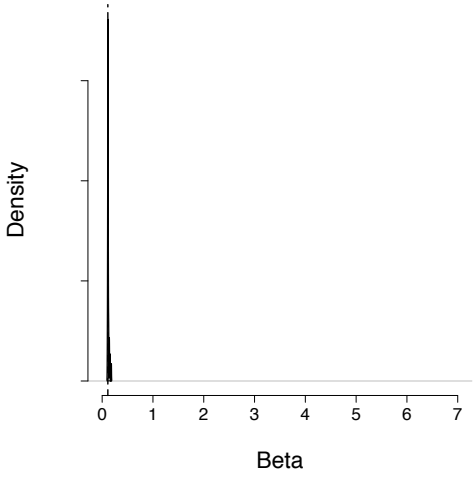
(a) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.



(b) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.

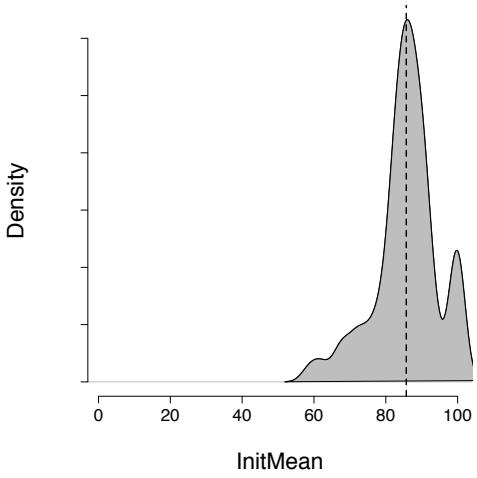


(c) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.

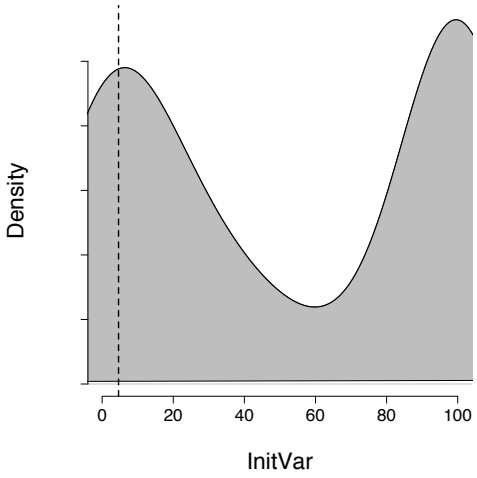


(d) Kernel density of the arithmetic mean of the 14 recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.

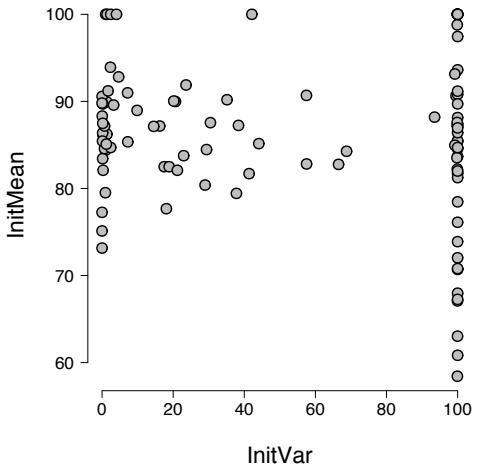
Figure 6. Kernel densities of the maximum likelihood estimates of 100 synthetic data sets.



(a) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.



(b) Kernel density of recovered parameter values (psoptim() likelihood estimates) from 100 tasks and synthetic data sets.



(c) Likelihood estimates of $\hat{\mu}_{i,1}^{\text{pre}}$ relative to likelihood estimates of $\hat{\sigma}_{i,1}^{2\,\text{pre}}$

Figure 7. Kernel densities of the maximum likelihood estimates of 100 synthetic data sets (top panels). Likelihood estimates of $\hat{\mu}_{i,1}^{\text{pre}}$ relative to likelihood estimates of $\hat{\sigma}_{i,1}^{2\,\text{pre}}$ (bottom panel).

Recovery of $\hat{\mu}_{i,1}^{\mathbf{pre}}$. Figure 7(a) shows two modes of which one is at the true parameter value. The other smaller mode is at the upper bound, which is restricted by the model. There is relatively high uncertainty, indicated by the large spread of the distribution. Estimates of $\hat{\mu}_{i,1}^{\text{pre}}$ typically lie in between 60 and 100.

Recovery of $\hat{\sigma}_{i,1}^{2\,\mathbf{pre}}$. Figure 7(b) shows two modes of which one is at the true parameter value. One bigger mode is at the upper bound. The likelihood estimates thus seem really biased. Naturally, there also is high uncertainty around the true parameter value, indicated by the large spread of the distribution. Estimates of $\hat{\sigma}_{i,1}^{2\,\text{pre}}$ spread across the complete parameter range, and concentrate just below the true parameter value and at the upper bound.

Interestingly, the mean with which the tracking process was initialized is recovered more accurate than the corresponding variance. Figure 7(c) shows whether the biased recoveries of both parameters correspond. It is clearly shown that this is not the case, since the biased recoveries of $\hat{\sigma}_{i,1}^{2\,\text{pre}}$ (e.g., $\hat{\sigma}_{i,1}^{2\,\text{pre}} = 100$) occur across the whole range of recoveries of $\hat{\mu}_{i,1}^{\text{pre}}$. Likewise, many of the biased recoveries of $\hat{\mu}_{i,1}^{\text{pre}}$ (e.g., $\hat{\mu}_{i,1}^{\text{pre}} = 100$) occur at accurately recovered values of $\hat{\sigma}_{i,1}^{2\,\text{pre}}$.

Discussion

The goal of this study was to investigate which parameters of the *Kalman filter* model can be recovered accurately. First we showed that, if only a single parameter is estimated and all remaining parameters are fixed to their true values, all parameters except the standard deviation of the diffusion noise ($\hat{\sigma_d}$) can be accurately recovered. Then, if the standard deviation of the diffusion noise is neglected, we showed that if the full model is tried to be estimated at once, two more parameters fail to be accurately recovered. The recoveries of the standard deviation of the payoffs ($\hat{\sigma_o}$) and the variances with which the tracking process is initialized ($\hat{\sigma}_{i,1}^{2\,\text{pre}}$) are biased and show considerable variability. Nonetheless, the decay center ($\hat{\theta}$), decay parameter ($\hat{\lambda}$), the means with which the tracking process is initialized ($\hat{\mu}_{i,1}^{\text{pre}}$), and the exploration parameter (β) could all be accurately recovered. The exploration parameter could even be accurately recovered for each single agent.

Given the above findings, some of the results by Daw et al. (2006) should be interpreted with care. For instance, the authors assume that their failure to estimate the standard deviation of the diffusion noise may be due to agents overestimating the speed of diffusion in the payoffs. They further argue that these large values induce high learning rates κ_t, and that this thus indicates that agents might be too sensitive for the most recent retrieved payoffs. The current study shows that it is rather the large variability of the estimates that explain the inflation of the standard deviation of the diffusion noise. Because of this large variability, no other reliable explanation for the inflation can be conceived.

Two more parameters from the *Kalman filter* model were biased. Of these, Daw et al. (2006) fixed the standard deviation of the payoffs to its generative value. In the current study this parameter was estimated, since we could recover it accurately if all other parameters were fixed to their generative values. Unfortunately, the recovery failed when the full model was tried to be estimated. Also, the variances with which the tracking process was initialized could not be accurately recovered in this case. One might argue that this second parameter has only little meaning, both content-wise and with respect to its relation

to other parameters of the model. The parameter is solely used for the initialization of the trackers. After initialization, it is updated by the decay parameter, the variance of the diffusion noise, and the learning rate, which is a function of the variance of the payoff (see Equation 4 and 8). A biased initial variance thus has the potential to get corrected. As one might have observed however, it must also be noted that the two other unidentified parameters are involved in the same process (i.e., the updating of the variance trackers).

The decay center and decay parameter, as well as the exploration parameters for each bandit and the means with which the tracking process was initialized can be recovered accurately. For these parameters, as well as the biased parameters, it must be stressed that the parameters recovered in the current study were estimated under the true model. Of course, this does not hold for the parameters in the study by Daw et al. (2006). It is not inconceivable that the authors' participants used one or more different models to make their choices in the bandit task. If this would be the case, the parameter estimates by Daw et al. (2006) would be inaccurate regardless of the results of the current study. In such a case the model might serve a predictive rather than a descriptive function.

Assuming that the *Kalman filter* model indeed underlay the agents' behavior, another implicit but important assumption is that all parameters but β are equal across participants. By taking into account the variability among agents the accuracy of the recoveries might suffer even more. Nonetheless, it is shown that at least the exploration parameter can be recovered accurately for each agent separately.

If one wants to use the model regardless of its identification problems[3], it is important to note that the number of estimated parameters significantly impedes the accuracy to which those parameters can be estimated. To overcome this problem, improvements on the model side and on the recovery side are suggested. One suggestion is to simplify the *Kalman filter* model. For instance highly correlating parameters might be combined, such that the number of parameters can be reduced. Another suggestion is to fix parameters that cannot be accurately recovered. For instance the variances with which the tracking process is initialized, the standard deviation of the diffusion noise, or the standard deviation of the payoffs might be fixed. Of course, the justification for fixing a parameter might depend on the significance of its meaning, whether it can be assumed to be constant across participants, and whether a sensible value can be conceived. Finally, on the recovery side, a posterior distribution instead of point estimate could be used. For instance in the case of the standard deviation of the diffusion noise, a posterior distribution would clearly show that increasing the value beyond say ten, would not increase the likelihood (see Figure 5). Also, it is advised to check the robustness of the current findings by varying the parameters of both the task and model, or by getting data that are more informative.

In conclusion, it is important to state that the parameters of the *Kalman filter* model as presented by Daw et al. (2006) cannot be recovered accurately. We meticulously replicated their study using synthetic data, thus guaranteeing that the true model underlay the data. Nonetheless, the estimates of multiple parameters were biased, leaving those parameters unidentified. This ultimately means that different parameter values return equivalent model fits, and the informal findings by Eric-Jan Wagenmakers and Ruud Wetzels are thus replicated.

[3]See Wickens (1967) for an in-depth discussion of identification problems and their treatment.

References

Acuna, D. & Schrater, P. (2008). Bayesian modeling of human sequential decision-making on the multi-armed bandit problem. In *Proceedings of the 30th annual conference of the cognitive science society* (Vol. 100, pp. 200–300). Washington, DC: Cognitive Science Society.

Audibert, J.-Y., Munos, R., & Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, *410*(19), 1876–1902.

Avner, O., Mannor, S., & Shamir, O. (2012). Decoupling exploration and exploitation in multi-armed bandits. *arXiv preprint arXiv:1205.2874*.

Bishop, G. & Welch, G. (2001). An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, *8*, 27599–3175.

Daw, N., O'Doherty, J., Dayan, P., Seymour, B., & Dolan, R. (2006). Cortical substrates for exploratory decisions in humans. *Nature*, *441*(7095), 876–879.

Granmo, O. & Berg, S. (2010). Solving non-stationary bandit problems by random sampling from sibling kalman filters. *Trends in Applied Intelligent Systems*, 199–208.

Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of machine learning* (pp. 760–766). Springer.

Koulouriotis, D. & Xanthopoulos, A. (2008). Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation*, *196*(2), 913–922.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, *58*(5), 527–535.

Steyvers, M., Lee, M., & Wagenmakers, E. (2009). A bayesian analysis of human decision-making on bandit problems. *Journal of Mathematical Psychology*, *53*(3), 168–179.

Sutton, R. & Barto, A. (1998). *Reinforcement learning: an introduction*. Cambridge Univ Press.

Wickens, T. D. (1967). *Parameter estimation in markov chain learning models* (Doctoral dissertation, Brown University).

Appendix A
R-Script: Non-Stationary Bandit Task

```
nsbpPayoff <- function(nbandit, nsession, ntrial, payoffInitMean, payoffSD=4,
                       lambda=.9836, theta=50, diffnoiseMean=0, diffnoiseSD=2.8) {

  # Creates a non-stationary payoff scheme using a random walk, for x bandits,
  # k sessions, and n trials. Each session has same parameters, starting values, etc.
  #
  # Args:
  #   nbandit: Number of bandits.
  #   nsession: Number of sessions.
  #   ntrial: Number of trials.
  #   payoffInitMean: Vector of initial payoff means.
  #   payoffSD: Payoff standard deviation. Default=4.
  #   lambda: Decay parameter. Default=.9836.
  #   theta: Decay center. Default=50.
  #   diffnoiseMean: Diffusion noise mean. Default=0.
  #   diffnoiseSD: Diffusion noise standard deviation. Default=2.8.
  #
```

```
  # Returns:
  #   Payoff scheme, including true parameter values underlying the scheme.

  #repeat until all payoffs are between 1 and 100
  repeat{
    #data structure
    payroll <- list(
      payoff          = list(),
      payoffMean      = list(),
      lambda          = lambda,
      theta                 = theta,
      payoffInitMean       = payoffInitMean,
      payoffSD             = payoffSD,
      diffnoiseMean         = diffnoiseMean,
      diffnoiseSD           = diffnoiseSD
    )

    for (i in 1:nsession) {
      payoffExact <- matrix(, ntrial, nbandit)
      payoffMean <- matrix(, ntrial, nbandit)

      #payoff means at t=1
      payoffMean[1, ] <- payoffInitMean

      #payoffs at t=1
      for (j in 1:nbandit) {
        payoffExact[1, j] <- round(rnorm(1, payoffMean[1, j], payoffSD), 0)
      }

      #diffusion noise
      nu <- matrix(rnorm(ntrial*nbandit, diffnoiseMean, diffnoiseSD), ntrial,
                   nbandit)

      #gaussian random walk
      for (j in 1:nbandit) {
        for (t in 1:(ntrial-1)) {
          payoffMean[t+1, j] <- (lambda * payoffMean[t, j]) + ((1-lambda)
                                                               * theta) + nu[t, j]
          payoffExact[t+1, j]        <- round(rnorm(1, payoffMean[t+1, j], payoffSD),
                                              0)
        }
      }
      payroll[["payoff"]][[i]] <- payoffExact
      payroll[["payoffMean"]][[i]] <- payoffMean
    }

    ### repeat until all payoffs are between 1 and 100 ###
    select <- 1:nsession #vector of all sessionnumbers
    k <- T
    for (i in select) {
      if (k == T) {
        test <- payroll$payoff[[i]]
        k <- F
      } else {
        test <- rbind(test, payroll$payoff[[i]])
      }
    }
    if(all(test >= 1) && all(test <= 100)) break()
  }
  return(payroll)
}
```

Appendix B
R-Script: Kalman Filter Model

```
nsbpData <- function(payoff, n, beta, lambda, theta, payoffSD, diffnoiseSD,
                     initMean, initVar, fit=F, trueData) {

  # Generates synthetic data based on a payoff scheme (data generated according
  # to the kalman filter / softmax routine, and given the specified parameters).
  #
  # Args:
  #   payoff: Payoff scheme.
  #   n: Number of participants.
  #   beta: Softmax exploration parameter. Vector of n values.
  #   lambda: Kalman filter parameter: decay parameter.
  #   theta: Kalman filter parameter: decay center.
  #   payoffSD: Kalman filter parameter: sd of payoff.
  #   diffnoiseSD: Kalman filter parameter: sd of diffusion noise.
  #   initMean: Kalman filter parameter: initial prior mean (at t=1).
  #   initVar: Kalman filter parameter: initial prior variance (at t=1).
  #   fit: Set to TRUE if function is used for fitting the model to real data.
  #     Set to FALSE if function is used to generate synthetic data.
  #   trueData: Dataset if fit=TRUE.
  #
  # Returns:
  #   Synthetic data of choices on a bandit task with specified payoff scheme and
  #     parameters.

  data <- list(subject = list())

  for(i in 1:n) {
    choices <- c()
    probabilities <- matrix(,nTrial,nBandit)
    kalmanMean <- c()
    kalmanVar <- c()

    kalmanMean[1:nBandit] <- initMean
    kalmanVar[1:nBandit] <- initVar

    for(t in 1:nTrial) {

      ##### BEGIN SOFTMAX RULE #####################################################
      #calculate probabilities
      softmax <- exp(beta[i]*kalmanMean) / sum(exp(beta[i]*kalmanMean))
      ##### END SOFTMAX RULE #######################################################

      #choose bandit given probabilities
      if(fit == F) {
        #use choice from current (estimated) data
        chosen <- which(rmultinom(n=1, size=1, prob=softmax) == 1)
      }
      if(fit == T) {
        #use choice from original (true) data
        chosen <- trueData$subject[[i]]$choices[t]
      }
      unchosen <- (1:nBandit)[-chosen]

      ##### BEGIN KALMAN FILTER ####################################################
      #calculate posterior of chosen bandit
      delta <- payoff[t, chosen] - kalmanMean[chosen]
      kappa <- kalmanVar[chosen] / (kalmanVar[chosen] + payoffSD^2)

      #update to posterior
      kalmanMean[chosen] <- kalmanMean[chosen] + kappa*delta
      kalmanVar[chosen] <- (1-kappa) * kalmanVar[chosen]
```

```
62
63      #update to prior
64      kalmanMean <- lambda*kalmanMean + (1-lambda)*theta
65      kalmanVar <- lambda^2*kalmanVar + diffnoiseSD^2
66      ##### END KALMAN FILTER ##################################################
67
68      #write choices and softmax probabilities
69      choices[t] <- chosen
70      probabilities[t, ] <- softmax
71    }
72
73    data$subject[[i]] <- list()
74    data$subject[[i]]$choices <- choices
75    data$subject[[i]]$probabilities <- probabilities
76  }
77
78  return(data)
79 }
```

Appendix C
R-Script: Discrepancy Measure

```
1  nsbpLnL <- function(parms, payoff, data) {
2
3    # Negative loglikelihood function for Kalman filter model.
4    #
5    # Args:
6    #   parms: Parameter values.
7    #   payoff: Payoff scheme.
8    #   data: True data.
9    #
10   # Returns:
11   #   Negative loglikelihood estimate given set of parameter values.
12
13   nSubject <- length(data$subject)
14
15   #fix all to data generative values
16   startBeta         <- estBeta
17   startLambda       <- estLambda
18   startTheta        <- estTheta
19   startPayoffSD     <- estPayoffSD
20   startDiffnoiseSD  <- estDiffnoiseSD
21   startInitMean     <- estInitMean
22   startInitVar      <- estInitVar
23
24   #free the ones that must be fitted
25 #  startTheta        <- parms[1]
26 #  startLambda       <- parms[1]
27 #  startPayoffSD     <- parms[1]
28 #  startDiffnoiseSD  <- parms[1]
29 #  startInitMean     <- parms[1]
30 #  startInitVar      <- parms[1]
31 #  startBeta         <- parms[1:nSubject]
32
33   trueData <- data
34
35   estData <- nsbpData(
36     payoff = payoff,
37     n = nSubject,
38     beta = startBeta,
39     lambda = startLambda,
40     theta = startTheta,
```

```
    payoffSD = startPayoffSD,
    diffnoiseSD = startDiffnoiseSD,
    initMean = startInitMean,
    initVar = startInitVar,
    fit = T,
    trueData = trueData)

  #bind all subjects (probabilities) in a single matrix
  est <- c()
  for (i in 1:nSubject) {
    est <- rbind(est, estData$subject[[i]]$probabilities)
  }

  #bind all subjects (choices) in a single matrix
  true <- c()
  for (i in 1:nSubject) {
    true <- c(true, trueData$subject[[i]]$choices)
  }

  #compute negative loglikelihood
  a <- c()
  for (t in 1:length(true)) {
    a[t]  <- est[t, true[t]]
  }

  print(-sum(log(a)))
  -sum(log(a))
}
```