Supplementary material to *Hierarchical diffusion models for two-choice response times*: I. Software implementation of the hierarchical diffusion model for two-choice response times

Joachim Vandekerckhove, Francis Tuerlinckx
Department of Psychology, University of Leuven

Michael Lee
Department of Cognitive Sciences, University of California, Irvine, CA

Abstract

In this document, we present `wiener.odc` and `wienereta.odc`, two pieces of Component Pascal code that can be incorporated into the popular Bayesian computation program WinBUGS (Lunn, Thomas, Best, & Spiegelhalter, 2000). With `wiener.odc` and `wienereta.odc` installed, WinBUGS's full range of general-purpose Markov chain Monte Carlo (MCMC) methods can be applied to the Wiener diffusion's two-choice reaction time distribution.

This document is a supplement to Vandekerckhove, Tuerlinckx, and Lee (in press). Its purpose is to instruct the reader on how to apply a hierarchical diffusion model (HDM) using freely available software. The main software package to use is WinBUGS, a popular Bayesian computation program (Lunn et al., 2000). WinBUGS allows a user to apply Bayesian models in a very straightforward manner, requiring only that a list of model assumptions is explicitly written down. From these mode assumptions, WinBUGS will select an appropriate sampling algorithm and produce samples from the joint posterior distribution of all parameters.

In what follows, we will first provide instructions on the installation and use of `wiener.odc` and `wienereta.odc`. We also provide some example code for a basic analysis and some more advanced examples. We will finish with some warnings regarding (computational) limitations to the code.

This research was funded by grants GOA/00/02–ZKA4511, GOA/2005/04–ZKB3312, IUAP P5/24, and K.2.215.07.N.01. This paper is part of the doctoral project of JV. This research was conducted utilizing high performance computational resources provided by the University of Leuven, `http://ludit.kuleuven.be/hpc`. We are further indebted to Microsoft Corporation and Dell Inc. for generously providing us with additional computing resources. Correspondence concerning this article may be addressed to: Joachim Vandekerckhove, University of Leuven, Department of Psychology, Tiensestraat 102 B3713, B–3000 Leuven, Belgium; ph: +3216326118; fax: +3216325993; e: joachim.vandekerckhove@psy.kuleuven.be

Installing the files

The `wiener.odc` and `wienereta.odc` files are released under a non-exclusive, non-transferrable license, which is included in the source code files. The files can be obtained from `http://ppw.kuleuven.be/okp/software/wienerodc/`.

Required materials

In order to use these files, you need to download and install three pieces of software, all of which are freely available on the internet. Install them *in the order given*. If you already have BlackBox installed, read the WinBUGS development page for instructions (`http://www.winbugs-development.org.uk/`).

1. *WinBUGS.* This is the basic program you will be using. It can be downloaded from `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml`. You need to register to get a key, but registration is free. Download and install the most recent version (at the time of writing, version 1.4.3).

2. *WinBUGS Development Interface (WBDev).* To be downloaded via `http://www.winbugs-development.org.uk/`. Unzip the executable to your WinBUGS directory. Then open, with WinBUGS, the `wbdev_01_09_04.txt` file that has appeared there and follow the instructions at the top of the file.

3. *BlackBox Component Builder.* This is an integrated development environment for programs written in Component Pascal (as WinBUGS is). It can be freely downloaded from `http://www.oberon.ch/blackbox.html`. This page also has a tutorial on Component Pascal, which may be useful in case you would like to write your own distributions or adapt the `wiener.odc` file. The most recent version of this program is 1.5 at the time of writing. Note that BlackBox Component Builder only runs on Windows platforms.

Download and install these three programs. Install WinBUGS in `/Program Files/WinBUGS` and BlackBox in `/Program Files/BlackBox Component Builder 1.5`. WBDev will have created its own directory `/WinBUGS/WBDev`.

Preparing BlackBox and compiling the ODC files

In your browser, open the WinBUGS directory and select all files (**Ctrl+A**) and copy them (**Ctrl+C**). Then open the BlackBox directory and paste those files there (**Ctrl+V**). Select "Yes to all" if asked about replacing files. Once this is done, you will be able to open BlackBox and run WinBUGS from inside it.

Now copy the files `wiener.odc` and `wienereta.odc` to the `/BlackBox Component Builder 1.5/WBDev/Mod` directory and then use BlackBox to open it. Press **Ctrl+K** to compile the distribution.

Now open the file `/BlackBox Component Builder 1.5/WBDev/Rsrc/Distributions.odc` and add the following lines of text to the end of the file (right above the `END` statement):

```
        s ~ "dwiener"(s, s, s, s)I(s, s) "WBDevWiener.Install"

    s ~ "dwiener.eta"(s, s, s, s, s)I(s, s) "WBDevWienerEta.Install".
```

Restart BlackBox to begin using the new distributions.

Using the distribution

Difference between `wiener.odc` *and* `wienereta.odc`

It is a common practice to assume trial-to-trial variability of the diffusion model parameters. While this is in principle easy in a Bayesian context, numerical problems arise in the `wiener.odc` function that make that variability in drift rate becomes difficult to estimate. For this reason, we also provide `wienereta.odc`. This file directly implements a diffusion model with trial-to-trial variability in drift rate (using the logarithm of Eq. 30 in Tuerlinckx, 2004, for the correct PDF for this case) and is numerically more robust for this case. In all other respects, the files are the same.

The mathematical forms of the distribution functions are

$$\text{Wiener}(t, x=0|\alpha,\tau,\zeta_{\text{init}},\delta) = \frac{\pi s^2}{\alpha^2}\exp\left(-\frac{\zeta_{\text{init}}\delta}{s^2}\right)\sum_{j=1}^{\infty} j\sin\left(\frac{\pi j\zeta_{\text{init}}}{\alpha}\right)\exp\left[-\frac{1}{2}\left(\frac{\delta^2}{s^2}+\frac{\pi^2 j^2 s^2}{\alpha^2}\right)(t-\tau)\right] \quad (1)$$

for `wiener.odc`, and

$$\text{WienerEta}(t, x=0|\alpha,\tau,\zeta_{\text{init}},\nu,\eta) = \frac{\pi s^2}{\alpha^2}\exp\left[-\frac{1}{2}\frac{\nu^2(t-\tau)}{(t-\tau)\eta^2+s^2}\right]\frac{1}{\sqrt{\frac{(t-\tau)\eta^2}{s^2}+1}}\exp\left\{-\frac{1}{2}\frac{\left(2\nu\zeta_{\text{init}}-\frac{\zeta_{\text{init}}^2\eta^2}{s^2}\right)}{[(t-\tau)\eta^2]+s^2}\right\} \times\sum_{j=1}^{\infty} j\sin\left(\frac{\pi j\zeta_{\text{init}}}{\alpha}\right)\exp\left[-\frac{1}{2}\left(\frac{\pi^2 j^2 s^2}{\alpha^2}\right)(t-\tau)\right] \quad (2)$$

for `wienereta.odc`. In both cases, the $x = 1$ case is obtained by substituting ζ_{init} with $\alpha - \zeta_{\text{init}}$ and ξ or ν with $-\xi$ and $-\nu$, and s is hard-coded to 0.1.

Formatting two-choice reaction time data

As user–contributed distributions in WinBUGS are necessarily unidimensional, we need to apply a trick to get it to accept the bivariate diffusion PDF. Consider that one dimension of the PDF is binary (response, denoted x_i), and the other is defined only on the positive half–line (reaction time, denoted t_i). It follows that a distribution on the full real line is defined by:

$$y_i = \begin{cases} t_i & \text{if } x_i = 1 \\ -t_i & \text{if } x_i = 0 \end{cases}$$

When using `wiener.odc` or `wienereta.odc`, you will need to code your two–choice reaction time to match the format of y_i. The functions will internally convert the negative response times to positive–valued error responses and treat the distribution as bivariate for the calculation of the likelihood value.

Forcing specific samplers in WinBUGS

WinBUGS is a so-called expert system that decides on the basis of a fixed set of rules which sampling algorithm is appropriate to use (see the section "Introduction > MCMC Methods" in the WinBUGS manual). However, sometimes WinBUGS' initial choices cause numerical under- or overflow when the Wiener distribution is used. In these situations, the choice of sampler can be adapted by opening the file `/BlackBox Component Builder 1.5/Updater/Rsrc/Methods.odc` (see "Changing MCMC Defaults" in the WinBUGS manual). In this list, the entries for "1 real non linear" and "9 log concave" are most relevant for hierarchical diffusion models. Changing the default settings to "UpdaterMetnormal" forces the application of a Metropolis-Hastings algorithm, improving numerical stability in some situations (at the cost of some computation speed).

If numerical issues occur, it is useful to find out which sampler is causing the problems. This can be discovered by accessing the "Info > Node Info" menu and selecting 'Methods' after entering the name of the node.

Examples of usage

Basic usage in WinBUGS

In order to implement `wiener.odc`, simply use a line like

`s ~ dwiener(alpha, zinit, tau, delta)`

in your WinBUGS code where `zinit` is the starting point in absolute value (i.e., $\alpha\beta$). Below we give a simple example for a data set with nc conditions (labeled $1, \ldots, nc$) and a total of N data points. The responses (properly formatted, in seconds, with positive and negative numbers) are stored in the variable y and condition indicators are in *cond*. This model assumes boundary separation alpha, bias beta, and nondecision time tau to be constant across conditions, but allows drift rate delta to differ (i.e., a fixed effect of condition on drift rate).

```
Fit a simple Wiener diffusion model {
    # Define priors on parameters
    beta  ~ dunif(0.1,0.9)
    tau   ~ dunif(0.05,1.00)
    alpha ~ dunif(0.03,0.25)
    # Insert a 'plate' to define multiple deltas
    for (r in 1:nc)
    {
        delta[r] ~ dunif(-0.75,0.75)
    }
    # Compute zinit from alpha and beta
    zinit <- alpha*beta
    # Connect the data to the Wiener process
```

```
    for (i in 1:N)
    {
        y[i] ~ dwiener(alpha,tau,zinit,delta[cond[i]])
    }
}
```

Advanced usage 1: Mixed model on nondecision time

In many cases, it is desirable to allow a parameter to vary from trial to trial according to a certain distribution (i.e., to use a "mixed model"). In the classical statistical framework, this leads to complicated integrals in the likelihood function. For example, to allow the nondecision time parameter to vary over trials, the likelihood function for conditions $p = 1, \ldots, P$ and items $i = 1, \ldots, I$ becomes complicated as in Equation 3, where the Wiener PDF is given in Equation 1 and $\mathrm{TN}(\mu, \sigma, L, U)$ indicates the truncated normal distribution with mean μ, standard deviation σ and lower and upper bounds L and U, respectively.

$$L(t_{(ip)}, x_{(ip)}|\alpha_{(p)}, \zeta_{(p)}, \delta_{(p)}, \mu_\tau, \sigma_\tau^2, L, U) = \prod_{p=1}^{P}\prod_{i=1}^{I}\int_{-\infty}^{+\infty} \mathrm{Wiener}(t_{(ip)}, x_{(ip)}|\alpha_{(p)}, \tau, \zeta_{(p)}, \delta_{(p)}) \times \mathrm{TN}\left(\tau|\mu_\tau, \sigma_\tau^2, L, U\right) \mathrm{d}\tau \quad (3)$$

In a classical statistical context, this integration significantly increases the computational cost of the likelihood function, which needs to be evaluated many times in order to numerically find the parameters corresponding to its maximum (see, e.g., Vandekerckhove & Tuerlinckx, 2007). In a Bayesian context, however, this integration can be performed by the MCMC algorithm, and does not pose further computational issues. The following code performs just such an analysis in WinBUGS, where

$$\boldsymbol{Y}_{(ij)} \sim W\left(\alpha_{(i)}, \beta, \tau_{(ij)}, \delta\right)$$

and

$$\tau_{(ij)} \sim TN\left(\theta, \chi^2, 0, 1\right).$$

```
Fit a Wiener diffusion model with mixing over nondecision time {
    # Define priors on parameters
    delta ~ dunif(-0.9,0.9) # assume only one drift rate now
    beta ~  dunif(0.01,0.99)
    # but suppose different boundary separations
    for (i in 1:nc)
```

```
    {
        alpha[i] ~ dunif(0.03,0.40)
        zinit[i] <- alpha[i]*beta
    }
    # Use a truncated normal distribution for tau, with mean theta and
    # standard deviation chi
    theta ~ dunif(0.05,0.80)
    chi   ~ dgamma(0.001,0.001)
    # Note that, for the parametrization of the normal distribution,
    # WinBUGS uses 1/variance (precision) instead of the standard
    # deviation
    precision <- pow(chi,-2)
    # Connect the data to the Wiener process but add the tau
    # distribution as well
    for (i in 1:N)
    {
        # Use I(X,Y) to truncate below X and above Y
        tau[i] ~ dnorm(theta,precision)I(0,1)
        y[i]   ~ dwiener(alpha[cond[i]],tau[i],zinit[cond[i]],delta)
    }
}
```

Advanced usage 2: Mixed model on drift rate

In order to apply a mixed model on drift rate, the above method may lead to numerical instability in the computation of the PDF. For this reason, we have also provided `wienereta.odc`, which is optimized for this case. In this implementation, the drift rate δ for each individual trial is assumed to be a draw from a normal distribution with mean ν and standard deviation η. Accordingly, when using the distribution, a fifth input parameter is required. Thus, in order to use `wienereta.odc`, type

`s` $\sim$ `dwiener.eta(alpha, zeta, tau, nu, eta)`

The parameter η (`eta`) should be restricted (in its prior) to be positive. The following is some example code for using `wienereta.odc`.

```
Fit a Wiener diffusion model {
    # Define priors on parameters
    beta  ~ dunif(0.01,0.99)
    tau   ~ dunif(0.05,0.80)
    alpha ~ dunif(0.03,0.50)
    nu    ~ dunif(-0.6,0.6)
```

```
    eta   ~ dunif(0,0.4)
    # Compute zinit from alpha and beta
    zinit <- alpha*beta
    # Connect the data to the Wiener process
    for (i in 1:N)
    {
        y[i] ~ dwiener.eta(a,ter,z,v,eta)
    }
}
```

Advanced usage 3: Posterior predictive values

The `wiener.odc` and `wienereta.odc` files are equipped with efficient simulators for two–choice reaction time data under a Wiener diffusion model (Tuerlinckx, Maris, Ratcliff, & De Boeck, 2001). This sampler has two uses. Firstly, it can be employed in a simulation study (e.g., for debugging, power analysis, etc.). Secondly, and more importantly, it can be used to generate posterior predictive values (Gelman, Carlin, Stern, & Rubin, 2004). Applying this in WinBUGS is straightforward, and a simple example is given below. WinBUGS can then be made to output these posterior predictive samples to a so-called coda file, which can be read by an external program (e.g., MATLAB or R), which can then compute summary statistics on the posterior predictives and compare the distributions of these samples to the values found in the data.

```
Fit a Wiener diffusion model {
    # Define priors on parameters
    delta ~ dunif(-0.9,0.9)
    beta ~ dunif(0.01,0.99)
    alpha ~ dunif(0.03,0.50)
    zinit <- alpha*beta
    tau ~ dunif(0.05,0.80)
    # Connect the data to the Wiener process
    for (i in 1:N)
    {
        y[i] ~ dwiener(alpha,tau,zinit,delta)
    }
    # Use the 'cut' function to prevent WinBUGS from including
    # the PPF in the posterior
    alpha.ppf <- cut(alpha)
    tau.ppf <- cut(tau)
    zinit.ppf <- cut(zinit)
    delta.ppf <- cut(delta)
    # Generate the PPF with the Wiener sampler
```

```
    for (i in 1:N)
    {
        ppf[i] ~ dwiener(alpha.ppf,tau.ppf,zinit.ppf,delta.ppf)
    }
}
```

Known issues

The software has some computational limits. Firstly, since evaluating the Wiener PDF requires much computation, any analysis will be relatively slow (depending on the size of the data set, the complexity of the hierarchical model, and of course the hardware, a real-life analysis may take hours or even days). The same is true for the sample generator for the Wiener PDF if posterior predictives are requested. Computation time can be reduced by distributing the computation over as many processors as possible (i.e., run one chain per processor; sometimes it is possible to split data sets that have no parameters in common). In the applications, we sometimes observed high autocorrelation within sample chains. While this in itself is not a severe problem, it is inefficient as it takes longer chains to walk through the entire region of the distribution.

Another issue occurs when extreme values are provided for certain parameters (in particular, those pertaining to the drift rate). This may cause numerical over- or underflow (i.e., real nonzero values that are so extreme that the software cannot distinguish them from zero or infinity), which makes WinBUGS crash (usually with the trap message "undefined real result"). Extreme values can be disallowed by truncating the relevant priors to a reasonable range.

A final issue occurs when WinBUGS' built-in slice sampler gets stuck (resulting in the error message "Could not bracket slice for node"). This can be dealt with by forcing WinBUGS to use a Metropolis-Hastings sampler instead of the slice sampler.

References

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis (2nd ed.)*. Boca Raton, FL: Chapman & Hall/CRC.

Lunn, D., Thomas, A., Best, N., & Spiegelhalter, D. (2000). WinBUGS — a Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing*, *10*, 325–337.

Tuerlinckx, F. (2004). The efficient computation of the distribution function of the diffusion process. *Behavior Research Methods, Instruments, & Computers*, *36*, 702-716.

Tuerlinckx, F., Maris, E., Ratcliff, R., & De Boeck, P. (2001). A comparison of four methods for simulating the diffusion process. *Behavior Research Methods, Instruments, & Computers*, *33*, 443–456.

Vandekerckhove, J., & Tuerlinckx, F. (2007). Fitting the Ratcliff diffusion model to experimental data. *Psychonomic Bulletin and Review*, *14*, 1011–1026.

Vandekerckhove, J., Tuerlinckx, F., & Lee, M. D. (in press). Hierarchical diffusion models for two-choice response times. *Psychological Methods*.