Behav Res (2011) 43:1044–1065
DOI 10.3758/s13428-011-0114-8

WoMMBAT: A user interface for hierarchical Bayesian estimation of working memory capacity

Richard D. Morey · Candice C. Morey

Published online: 24 June 2011

Abstract The change detection paradigm has become an important tool for researchers studying working memory. Change detection is especially useful for studying visual working memory, because recall paradigms are difficult to employ in the visual modality. Pashler (*Perception & Psychophysics*, *44*, 369–378, 1988) and Cowan (*Behavioral and Brain Sciences*, *24*, 87–114, 2001) suggested formulas for estimating working memory capacity from change detection data. Although these formulas have become widely used, Morey (*Journal of Mathematical Psychology*, *55*, 8–24, 2011) showed that the formulas suffer from a number of issues, including inefficient use of information, bias, volatility, uninterpretable parameter estimates, and violation of ANOVA assumptions. Morey presented a hierarchical Bayesian extension of Pashler's and Cowan's basic models that mitigates these issues. Here, we present WoMMBAT (Working Memory Modeling using Bayesian Analysis Techniques) software for fitting Morey's model to data. WoMMBAT has a graphical user interface, is freely available, and is cross-platform, running on Windows, Linux, and Mac operating systems.

Keywords Visual short term memory · Working memory · Visual change detection · Capacity estimates · Bayesian hierarchical models · Bayesian models · Multinomial models

It is presumed that mental representations of visual images are encoded and maintained (Logie, 1995; Paivio, 1990; Repovs & Baddeley, 2006); however, measuring visual memory is a methodological challenge. How can researchers ascertain what participants remember from a visual scene? Unlike with verbal materials, it is not straightforward to elicit free recall of visual materials. Possibilities for approximating recall with visual materials range from prompting participants for verbal descriptions of visual stimuli (Parra, Della Sala, Logie & Abrahams, 2009) to asking for hand-drawn representations of memories (Rubin & Kontis, 1983). These techniques, while adequate for some research questions, are frequently unsuitable. In many instances, researchers need to use stimuli that are visually complex and, perhaps, difficult to verbally label (for instance, variations of a shaded cube or Asian characters; Alvarez & Cavanagh, 2004; Awh, Barton, & Vogel, 2007). These needs limit the usefulness of interpreting participants' verbal descriptions of visual memories. Likewise, for many such stimuli, hand-drawn responses may not be sufficiently clear to distinguish one stimulus from another. These difficulties make recall an implausible technique for identifying limits in visual short-term memory capacity.

Fortunately, recognition memory paradigms do not suffer from the same limitations. Using visual change detection (Luck & Vogel, 1997; Phillips, 1974), researchers can manipulate visual stimuli along any feature dimension. Researchers can choose abstract, rather than easily nameable, stimuli. Researchers can vary the number of stimuli presented, their spatial configuration, and their temporal configuration. This level of control is ideal for hypothesis testing, and thus this technique has produced a wealth of novel research since Luck and Vogel reintroduced this paradigm. However, unlike a study of verbal memory through recall, discovering precisely how much participants remember from some visual display through change recognition requires modeling, and models have been developed for this purpose. Using hit and correct rejection rates, total number of items to be remembered, and sensible assumptions about how participants might use the informa-

R. D. Morey (✉) · C. C. Morey
University of Groningen DMPG,
Grote Kruisstraat 2/1,
9712TS Groningen, The Netherlands
e-mail: r.d.morey@rug.nl

tion available at test to inform guessing, researchers can estimate how many items participants must have remembered in order to achieve some known proportion correct (Cowan, 2001; Cowan, Elliott, Saults, Morey, Mattox, Hismjatullina, & Conway, 2005; Pashler, 1988).

These models are frequently used (e.g., Awh et al., 2007; Fougnie & Marois, 2009; Gold, Fuller, Robinson, McMahon, Braun, & Luck, 2006; Kumar & Jiang, 2005; Morey, Cowan, Morey, & Rouder, 2011; Olsson & Poom, 2005; Treisman & Zhang, 2006; Vogel, McCollough, & Machizawa, 2005; Wilken & Ma, 2004; Xu & Chun 2006) and are often helpful for elucidating relationships that might otherwise be difficult to interpret. For instance, C. C. Morey et al. compared performance on concurrent auditory and visual change detection tasks, with reward for correct responses on each task varying. Because visual and auditory stimulus sets included different numbers of items, comparing proportions correct between tasks would have been problematic. In this case, computing estimates of capacity allowed for precise descriptions of trade-offs observed between the two tasks. Sometimes, however, the simple methods typically used for calculating these estimates result in volatile or uninterpretable output. Users of these models will encounter the occasional negative capacity estimate, which is the result of below-chance responding. When too few trials contribute to the mean of some combination of conditions, some sub–chance means are expected due merely to sampling noise. Simple methods for computing these estimates also produce output that logically conflicts with itself. Consider the data of Woodman and Vogel (2005), which beautifully illustrate a typical pattern observed when set size is manipulated and extremely small set sizes are included. Woodman and Vogel estimated participants' capacity for unmasked sample arrays of one or two items. When these sample arrays included two items, average capacity estimates were greater than one item, but when these sample arrays included only one item, average estimates were less than one item. If capacity estimates are used as dependent variables to express the effects of some other manipulated variable, this is not an alarming logical problem. However, if researchers use these to truly model the capacity of visual short-term memory, internal consistency is necessary, and simple methods for estimating visual short-term memory capacity fail to provide it (R. D. Morey, 2011).

R. D. Morey (2011) introduced a hierarchical Bayesian version of the models that Pashler (1988) and Cowan (2001) advocated for the measurement of working memory capacity and showed that the model was superior in a number of ways to the formulas currently in use for estimating working memory capacity. In this article, we introduce WoMMBAT (working memory modeling using Bayesian analysis techniques), graphical software for estimating capacity using Morey's hierarchical model. Before introducing the WoMMBAT software, however, we first describe Pashler's and Cowan's multinomial tree models, on which the formulas are based, and Morey's hierarchical extension.

Change detection and working memory capacity

Figure 1 shows a typical trial sequence in a visual change detection task. After fixation, a study array is presented for a short time. The study array may be followed by either a mask or a blank screen, which is, in turn, followed by a test display. The test display consists of either a whole array (Fig. 1, left) or a single item (Fig. 1, right); Wheeler and Treisman (2002) called these two variations on the change detection task *whole display* and *single probe*, respectively. On some proportion of trials, a single square differs from the study array. The participant's task is to detect whether a change has occurred and to respond accordingly.

Pashler (1988) suggested a simple psychological model for modeling change detection performance, irrespective of the chosen paradigm. Suppose that participants have a fixed capacity, K, for remembering items in a display. At study, participants encode K out of the N items in the display. Items not encoded are lost. The K encoded items may be used at test to determine whether the display has changed. The all-or-none encoding property of this model is a strong assumption, but the model has been supported empirically (Cowan & Rouder, 2009; Rouder, Morey, Cowan, Zwilling, Morey, & Pratte, 2008; but cf. Bays & Husain, 2009).

Of interest is to estimate the capacity K of a participant, given his or her change detection performance. Pashler (1988) and Cowan (2001) both suggested formulas, derived from different multinomial tree models, that might be used to estimate capacity; they differed in the paradigms that they applied. To see why, consider, first, trials on which a change occurs between study and test; *change* trials can be treated similarly in both whole-display and single-probe paradigms. Because there are N items in the array and the participant encodes K items, the probability that the participant will encode the changed item and, thus, detect the change is

$$D = \min\left(\frac{K}{N}, 1\right).$$

The min function ensures that the probability never exceeds 1. With a probability of 1– D, the participant will not encode the item that changes and does not know whether it changed. In this case, the participant guesses "change" with a probability of G. The tree in Fig. 2, top left, shows a multinomial tree for trials on which the

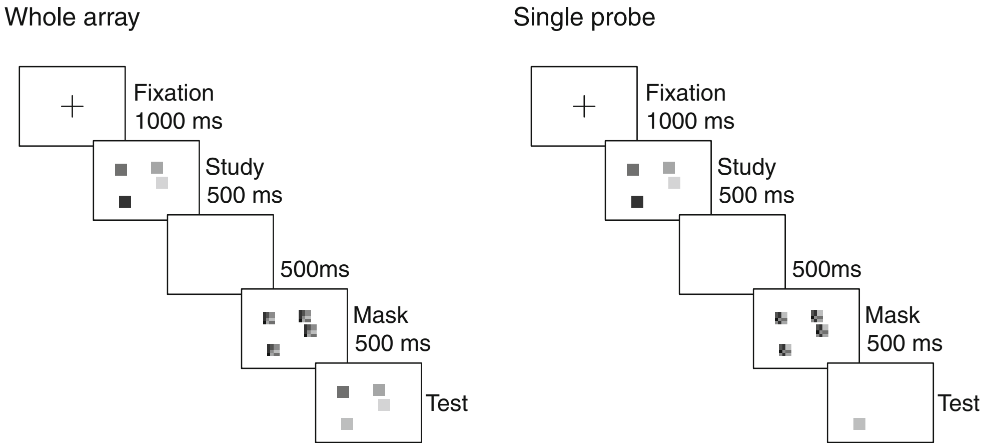


Fig. 1 Visual change detection paradigms. After fixation, an array of colored squares is presented for study and subsequently masked. In the whole-display paradigm (left), the whole array is tested, and one square may have change. In the single-probe paradigm (right), a single item is tested, which may have changed

stimulus changed. From the tree, it is apparent that the true hit rate H is

$$H = D + (1-D)G = \min\left(\frac{K}{N},1\right) + \left(1-\min\left(\frac{K}{N},1\right)\right)G.$$

This true hit rate holds for both the whole-display and single-cue designs. The whole-display and single-cue designs differ, however, in how *same* trials are modeled.

Pashler (1988) considered the equivalent of a whole-display design and noted that, on *same* trials in which the whole stimulus is tested, a participant will always be in a state of uncertainty. No change can be detected, but the participant is unsure about whether no change was detected because there was not a change or because he or she failed to encode the item that changed. Again, the participant guesses "change" with a probability of G. This leads to the true false alarm rate F:

$$F = G'$$
$$G' = \begin{cases} G & K < N \\ 0 & K \geq N. \end{cases}$$

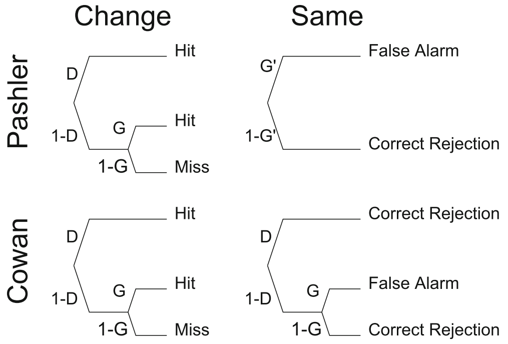


Fig. 2 Pashler's (1988; top row) and Cowan's (2001; bottom row) models of change detection performance, with change detection probability D = min(K/N, 1) and guessing rate G. The left and right columns show the trees for *change* trials and *same* trials, respectively. Pashler's model is appropriate for the whole-display paradigm; Cowan's model is appropriate for the single-probe paradigm

The term G' ensures that if a participant can encode all items in the array, that participant will never respond "change" on a *same* trial. The corresponding multinomial tree is shown in Fig. 2, top right.

Change detection data offer estimates of true hit and false alarm probabilities, which are used to estimate K in whole-display designs:

$$\hat{K}_P = N\left(\frac{\hat{H}-\hat{F}}{1-\hat{F}}\right) \quad K \leq N \tag{1}$$

where N is the array size, $\hat{H}$ and $\hat{F}$ are estimates of hit and false alarm probabilities, and $\hat{K}_P$ is the estimate of capacity (subscripted by P, for *Pashler*). The restriction $K \leq N$ underscores the fact that the formula does not provide meaningful estimates when capacity is greater than the array size; performance in this case is predicted to be perfect.

While the estimate $\hat{K}_P$ is appropriate in whole-display designs, it is not appropriate for estimating capacity in single-probe designs. In single-probe designs, the participant has information not available in whole-display designs: The single probe indicates which, if any, square changed from study to test. This difference manifests itself as a difference between the false alarm rates in the two models. Consider a *same* trial in the single-probe design. If the participant has encoded the probed item, the participant can confidently respond "no change." This occurs with a probability of D = min(K/N,1). If the participant has not encoded the probed item, he or she guesses "change" with a probability of G. The resulting tree for *same* trials in the single-probe design is shown in Fig. 2, bottom right.

The estimates of hit and false alarm rates from single-probe data can be used to estimate capacity results in the following estimate, suggested by Cowan (2001):

$$\hat{K}_C = N(\hat{H}-\hat{F}) \quad K \leq N \tag{2}$$

where $\hat{K}_C$ is the estimate of capacity, subscripted by C, for *Cowan*.

Although for any given hit and false alarm rate it is possible to compute either Pashler's (1988) estimate or Cowan's (2001) estimate of capacity, the estimate will be interpretable only if the formula matches the experimental design used. Pashler's estimate should be used for whole-display designs, and Cowan's estimate should be used only for single-probe designs (Rouder, Morey, Morey, & Cowan, 2011).

The advantage of using the formulas above for estimating capacity from visual change detection data is clear: The estimates give a measure of the amount of available WM capacity, without contamination by response bias. Although the model underlying the formulas requires assumptions, these assumptions are testable and appear to be reasonable in light of current research.

One assumption that has been reconsidered concerns the model predictions when capacity is greater than the array set size. If $K > N$, performance is predicted to be perfect in both whole-display and single-probe designs. If a participant can encode every item in the study array, he or she can always detect a change and correctly reject non–changes. However, this prediction does not appear to hold. Rouder et al. (2008) tested participants in three array set–size conditions: 2, 5, and 8 items. In the 5- and 8-item conditions, participants generally performed consistently with a capacity of 3 to 3.5 items. For participants with capacities of 3 items, an array with only 2 items should pose no difficulty: Both items can be stored, so performance should be perfect. Indeed, participants scored well for the 2-item arrays, often greater than 95% correct. The model, however, predicts perfect performance; the finding of errors in the set size 2 condition appears to violate the assumption that participants have a constant capacity K.

Rouder et al. (2008) suggested a simple extension of the model to account for imperfect performance at low set sizes. Rouder et al. (2008) reasoned that it is too simplistic to assume that participants pay attention to the task on every trial. Participants may "zone-out," or experience an attentional lapse, on a small number of trials. Psychophysical models have long incorporated a lapse rate parameter as a theoretical ceiling on performance. The addition of a lapse parameter to the working memory models outlined is straightforward; on every trial, participants pay attention to the task at hand with a probability of Z.[1] If the participants lapse, they encode no items and guess "change" with a probability of G. If they do not lapse, the appropriate model, as outlined above, applies. The resulting trees, for both the whole-display and single-probe designs, are shown in Fig. 3.

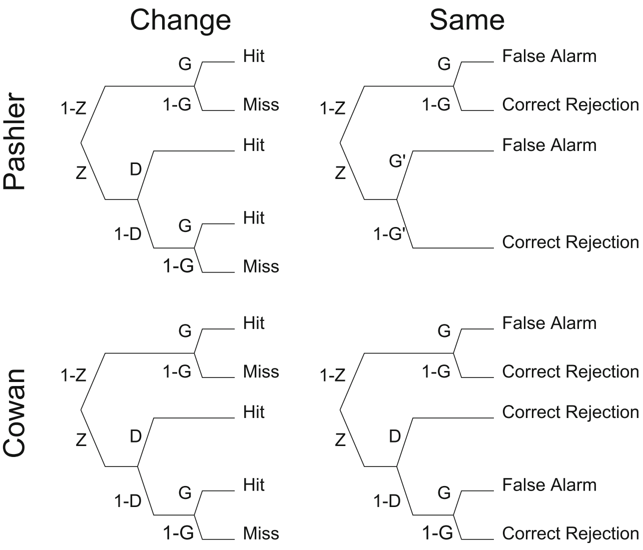


Fig. 3 Extending Pashler's (1988; top row) and Cowan's (2001; bottom row) models of change detection performance to include a (non) lapsing parameter Z. The left and right columns show the trees for change trials and *same* trials, respectively. The extended Pashler model is appropriate for the whole-display paradigm; the extended Cowan model is appropriate for the single-probe paradigm

Although the addition of a lapse rate is a reasonable and necessary addition to the model, there are no longer easy-to-compute formulas for estimating capacity. However, R. D. Morey (2011) showed that the simple formulas in Eqs. 1 and 2 are prone to bias, volatility, inefficient use of information, uninterpretable negative capacity estimates, and violations of ANOVA assumptions. Although the formulas suggested by Pashler (1988) and Cowan (2001) are simple, these problems make an alternative approach desirable. Morey extended Rouder et al. (2008) nonhierarchical model to a hierarchical Bayesian model that mitigates the issues mentioned above, providing efficient, stable estimates of capacity. In the following section, we provide an overview of Morey's model and then present WoMMBAT, which allows analysts to easily fit the hierarchical model to data and make inferences.

Morey's hierarchical model

The mathematical details of the hierarchical model can be found in R. D. Morey (2011). Here, we attempt to give a less technical introduction to the Bayesian hierarchical model that is accessible to non–methodologists. Figure 4 depicts the hierarchical model graphically. The following development of the model follows the figure from top (the observed data) to bottom (the Bayesian priors).

The observed data are at the first level of the hierarchical model. Consider the data from a single condition, for 1 participant. Suppose the participant observes M_c *change* trials and M_s *same* trials. Then the observed number of hits

[1] Although the Z parameter is actually the probability of *not* lapsing, it is referred to as the lapse parameter because it governs lapsing.

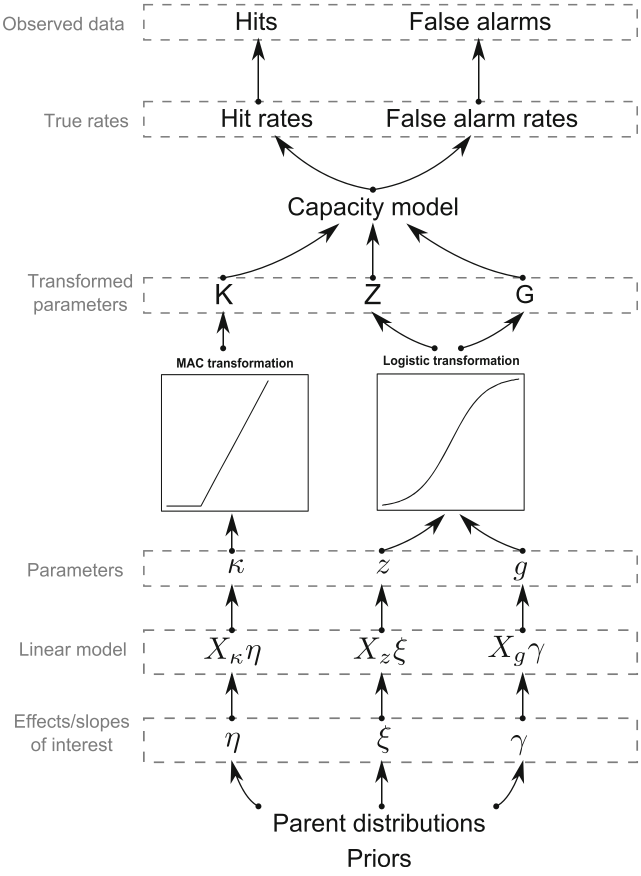


Fig. 4 A graphical depiction of the structure of R. D. Morey's (2011) Bayesian hierarchical model of working memory capacity

and false alarms are assumed to come from binomial distributions:

$$Y_h \sim \text{Binomial}(M_c, p_h) \tag{3}$$

$$Y_f \sim \text{Binomial}\left(M_s, p_f\right) \tag{4}$$

where p_h and p_f are the true hit and false alarm probabilities, respectively. Observed hits and false alarms, Y_h and Y_f, are binomial random variables that arise from the true hit and false alarm rates, as depicted by the arrows joining the first and second rows of Fig. 4.

The true hit and false alarm rates are, in turn, dependent on the underlying capacity, lapse, and guessing parameters for that condition. Given the capacity, lapse, and guessing parameters, the true hit and false alarm rates may be computed using the multinomial trees in Fig. 3. Choosing the correct multinomial tree capacity model for the design that yielded the data (full–display or single–cue) is critical; analyzing the data with the wrong capacity model will yield uninterpretable results.

The capacity (*K*), lapsing (*Z*), and guessing (*G*) parameters are the parameters of interest for researchers. Analysts are most interested in how manipulations affect these three parameters. However, these parameters are not defined in a convenient space for analysis by linear model techniques. The parameters Z and G are both probabilities and are thus constrained to be in the interval (0,1). The capacity parameter *K* is not as constrained; all nonnegative values are possible capacities. However, this constraint still causes difficulties for analysis, such as distortions near floor.

Problems arising from constraints on parameters are well known, and there are a number of solutions in the statistics literature. The most common solution is to treat the constrained parameters as arising from other, unconstrained parameters through a transformation. For instance, in logistic regression, models are placed on probabilities by assuming that the probability parameters arise from transformations of parameters on the log-odds scale (McCullagh & Nelder, 1989). Because log-odds parameters are defined within the range $(-\infty, \infty)$, distortions from ceiling or floor probabilities are mitigated. It is straightforward to place linear models on the unconstrained parameters.

In the case of the hierarchical working memory model, the lapsing (*Z*) and guessing (*G*) parameters are both probabilities, making the logistic transformation a natural choice. This is depicted in Fig. 4 by the logistic function between Z and G, the probabilities, and z and g, the unconstrained parameters. In mathematical notation,

$$Z = \tfrac{1}{1+e^{-z}}$$

$$G = \tfrac{1}{1+e^{-g}}.$$

Linear models may then be placed on z and g.

The situation for *K* is slightly more complicated. Unlike the probabilities *Z* and *G*, *K* is defined for values greater than 1, making the logistic transformation inappropriate. Also, the K parameter has natural units: capacity. A nonlinear transformation, like the logarithmic function, would distort the natural capacity space and make effects difficult to interpret. These two considerations, that *K* is defined for all positive numbers and that we wish to avoid nonlinear transformations, lead to the following transformation between *K*, the constrained parameter, and κ, the unconstrained parameter:

$$K = \max(\kappa, 0).$$

For all positive values, κ and *K* are the same, preserving the natural capacity units. All negative values of κ are mapped to $K = 0$, making this a type of mass-at-chance (MAC) transformation (Morey, Rouder, & Speckman, 2009; Rouder et al. 2007). Negative κ values may, at first glance, seem undesirable. Indeed, when the capacity formulas in Eqs. 1 and 2 are used, negative capacity estimates are interpretable only as the result of sampling error. Using the MAC transformation within the hierarchical model allows for a different interpretation. Imagine that, on the basis of many participants, we know that a certain memory load manipulation decreases WM capacity by 3 units. If we test a participant whose working memory capacity is 2, we expect to see behavior consistent with a capacity of 0. However, the participant is overloaded by 1 unit. If we tested the participant's working memory capacity both under load and not under load, the MAC transformation would allow us to estimate the participant's capacity under load as $\kappa = -1$. Likewise, a participant whose unloaded capacity is estimated at 1 will, under load, yield an estimate of $\kappa = -2$. Although these 2 participants will both behave consistently with having 0 capacity under the load (i.e., $K = 0$), one participant is less overloaded than the other, and the hierarchical model uses negative κ estimates to reflect this.

With the introduction of parameters κ, z, and g whose parameter space is unconstrained, it is possible to place linear models on each of the parameters, depicted in matrix notation in Fig. 4. Up to this point, we have considered the parameters only in a single condition, for a single participant. As in standard linear models such as regression and ANOVA, individual parameters in the hierarchical model arise by linearly combining the effects of various factors. As an example, we continue the previous memory load example. Implied by the previous paragraph is the following main effects model:

$$\kappa_{ij} = \mu^{(\kappa)} + \eta_{1i} + \eta_{2j}$$

where $\mu^{(\kappa)}$ is the grand mean capacity, η_{1i} is the effect on capacity of the ith participant (i.e., how far it is above the mean capacity), and η_{2j} is the load effect of the jth load manipulation condition. In this model, we treat the load manipulation as a factor and estimate the effect of the load at each level of the factor.

We could also estimate the effect of the load manipulation in a regression-type model:

$$\kappa_{ij} = \mu^{(\kappa)} + \eta_{1i} + x_j\eta_2$$

where x_j is the size of the jth load manipulation. Here, we estimate only one parameter for the effect of the memory load, η_2. This parameter represents the slope of the relationship between memory load and capacity. We can use the slope to estimate how much visual working memory capacity is occupied by each unit of load.

Another model we might entertain is

$$\kappa_{ij} = \mu^{(\kappa)} + \eta_{1ij}$$

where η_{1ij} is the effect of the interaction between participant and load; that is, each participant × load combination is allowed to have its own effect. Note that in this interaction model, the two main effects are not included with the interaction, as they would be in an ANOVA model. In many Bayesian hierarchical models, including Morey's working memory model, interaction effects are completely unstructured, aside from being constrained to have a mean of 0 (Gelman, Carlin, Stern, & Rubin, 2004). Thus, the interaction can account for main effects, with the added expense that interaction models are more complex. It is the added complexity that will enable model comparisons between interaction models and main effects models; all other things being equal, less complex models are preferred to more complex models. We discuss model comparisons in the context of our demonstration of the WoMMBAT software below. Regardless of which model we may fit, our interest is in estimating the effect parameters represented by the various η parameters above. The WoMMBAT software allows researchers to easily build models of arbitrary complexity simply by adding factors and continuous covariates in a graphical user interface, whose parameters are then estimated.

After the linear models, there are two additional levels of the hierarchical model. First, each group of parameters—for instance, all levels of a factor—are assumed to come from a normal parent distribution. For categorical factors, the mean of the distribution is constrained to be 0; for slopes, the mean is estimated. Because the model is Bayesian, unknown parameters of these parent distributions have prior distributions placed on them. For more details, see R. D. Morey (2011).

With the structure of the hierarchical model sketched, we now turn to a demonstration of the software designed for building specific models for the analysis of data. In the example below, we analyze the dataset of Rouder et al. (2008) within the WoMMBAT software. The analysis shows the important features of WoMMBAT and gives potential analysts enough information to perform their own analyses.

Analysis of Rouder et al. (2008)

Rouder et al. (2008) presented an experiment in which 23 participants completed a single-probe visual array change detection task, like that shown on Fig. 1 (right). Participants performed 540 trials in nine conditions, created by crossing three set sizes (two, five, and eight) with three levels of probability *change* (30%, 50%, 70%). Rouder et al. (2008) used maximum likelihood methods to show that a nonhierarchical version of Cowan's (2001) model, with the lapse parameter added, fit the data well for the overwhelming majority (20/23) of participants. Rouder et al. (2008) took this as evidence for fixed-capacity, all-or-none models of working memory capacity. R. D. Morey (2011) provided a more detailed analysis of the same data, using the hierarchical model outlined above to show specific color effects on capacity and guessing. This fine-grained analysis would have been difficult or impossible with the methods used by Rouder et al. (2008). The purpose of the remainder of the article is to describe for researchers how to conduct a simple analysis of change detection data, using the Rouder et al. (2008) data as a model data set. In order to demonstrate the use of the WoMMBAT software, we will focus on building models to answer the following question: Did the change-probability manipulation successfully affect participants' guessing biases?

Installing and running the necessary software

The WoMMBAT software necessary to fit R. D. Morey's (2011) hierarchical model is available as a package in R (R Development Core Team, 2009). Install the latest version of R, available for Windows,[2] Macintosh, and Linux operating systems from http://cran.r-project.org. After installing R, visit the main website for the WoMMBAT project at http://wmcapacity.r-forge.r-project.org/ for specific instructions on how to install the WoMMBAT software under your operating system.

After the necessary software is installed, we can start R and begin a WoMMBAT session. The text below is R script, which can be copied and pasted into the R console. The # symbol at the beginning of a line represents a comment, explaining what the code will do when run. First, we load the WMCapacity package in R:

```
# Install the WoMMBAT software,
# from the WMCapacity package
library('WMCapacity')
```

In Windows, this may also be done through the "Packages → Install Package(s)…" menu option (and in MacOS, through the Package Installer). Once the package is loaded, we load the Rouder et al. (2008) data set, which we will use for demonstration.

```
# Load the Rouder et al. (2008) data set;
# it will be placed
# in a variable called "VisualArray"
data(VisualArray)
# Begin the WoMMBAT analysis GUI
wommbatGUI()
```

On running the code above, the WoMMBAT interface should present itself as shown in Fig. 5. The first screen displayed will be brief instructions on the usage of WoMMBAT, including how data should be formatted for import into WoMMBAT.

Data set tab

The first tab after the introduction window is the data set tab, shown in Fig. 6. The data set tab is used for loading data (Fig. 6A), specifying a capacity model for the research design (Fig. 6B), and telling WoMMBAT which columns in the data set are of interest (Fig. 6C–E).

There are four methods of loading data into WoMMBAT, available via the four buttons at the top of the data set tab (Fig. 6A). The leftmost button, "Unlock data," resets all analyses and reloads the currently loaded dataset, allowing the analyst to change the columns of interest. The "Open CSV" button allows loading of a comma separated value (CSV) file. This is useful if data cleaning or analyses are done in another program and the analyst wishes to import the data into WoMMBAT. The "R data frame" button allows loading an R data frame, which is useful if the data are already loaded for cleaning and analysis within R. Finally, the "Open saved analysis" button allows opening a WoMMBAT analysis previously saved via the "Save/Export" tab.

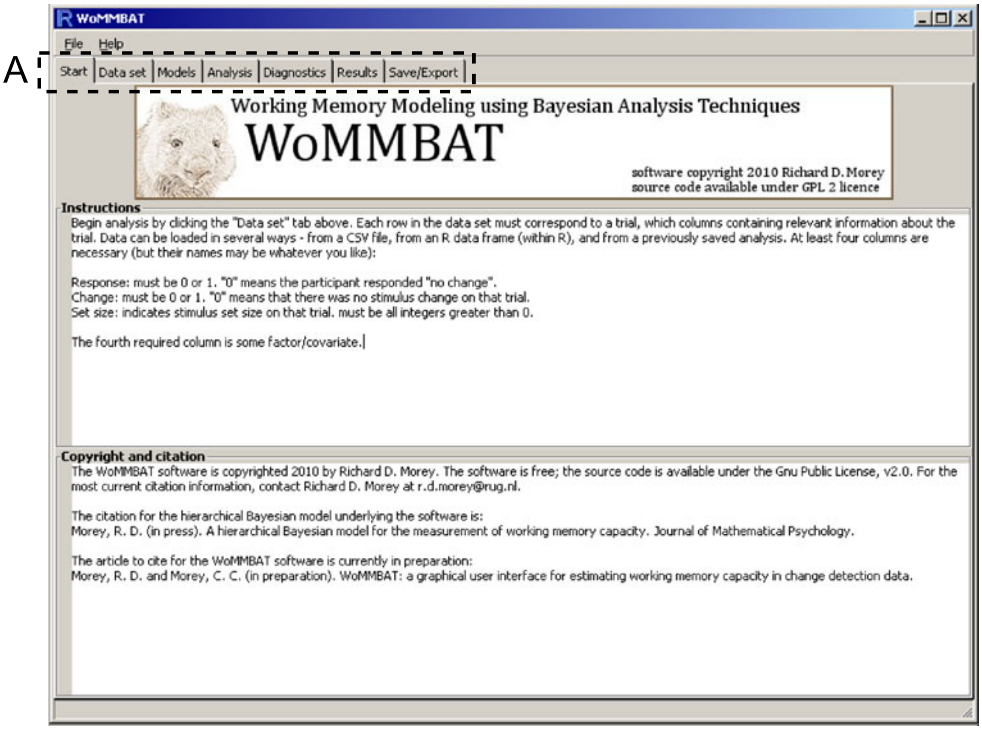
Because we have already loaded the Rouder et al. (2008) data into R via the data() function, the "R data frame" button should be used to load the data into WoMMBAT. Click "R data frame" and then select VisualArray from the resulting window. This should load the Rouder et al. (2008) data, filling the available column list (Fig. 6C) with the column names.

A data set for WoMMBAT analysis should have at least four columns and as many rows as there are total trials. Three necessary columns correspond to elements of the experimental design. Table 1 lists these three necessary columns, which are specified by the corresponding buttons (Fig. 6D). In the case of the Rouder et al. (2008) data set, the response, change, and set size columns are called resp, ischange, and N, respectively. If you make a mistake in specifying the response, change, or set size variables, simply move the correct variable into the spot, and the incorrect variable will reappear in the list of available columns (Fig. 6C).

At least one additional column of interest is required for a WoMMBAT analysis. Any variable that is of theoretical interest can be included in the analysis. The Rouder et al. (2008) data set included 23 participants and three conditions differing in the proportion of change trials. Highlighting sub (for *subject*) and clicking the "Categorical" button will define participant as a factor of interest.

For the change probability manipulation, the data may be analyzed in two ways: We may include prch (probability of a change trial) as a categorical predictor or a continuous predictor. In this data set, the variable prch is coded as the log odds of a change, instead of probability, so it may make sense to include prch as a continuous predictor. The difference between specifying a predictor as continuous or categorical is how they are included in the model; categorical predictors are treated as factors, as in an

[2] The Windows version of the R installation comes with both 32-bit and 64-bit executables. We recommend that Windows users run the 32-bit version of R for compatibility with the WoMMBAT software.



Working Memory Modeling using Bayesian Analysis Techniques

WoMMBAT

software copyright 2010 Richard D. Morey
source code available under GPL 2 licence

Instructions

Begin analysis by clicking the "Data set" tab above. Each row in the data set must correspond to a trial, which columns containing relevant information about the trial. Data can be loaded in several ways - from a CSV file, from an R data frame (within R), and from a previously saved analysis. At least four columns are necessary (but their names may be whatever you like):

Response: must be 0 or 1. "0" means the participant responded "no change".
Change: must be 0 or 1. "0" means that there was no stimulus change on that trial.
Set size: indicates stimulus set size on that trial. must be all integers greater than 0.

The fourth required column is some factor/covariate.

Copyright and citation

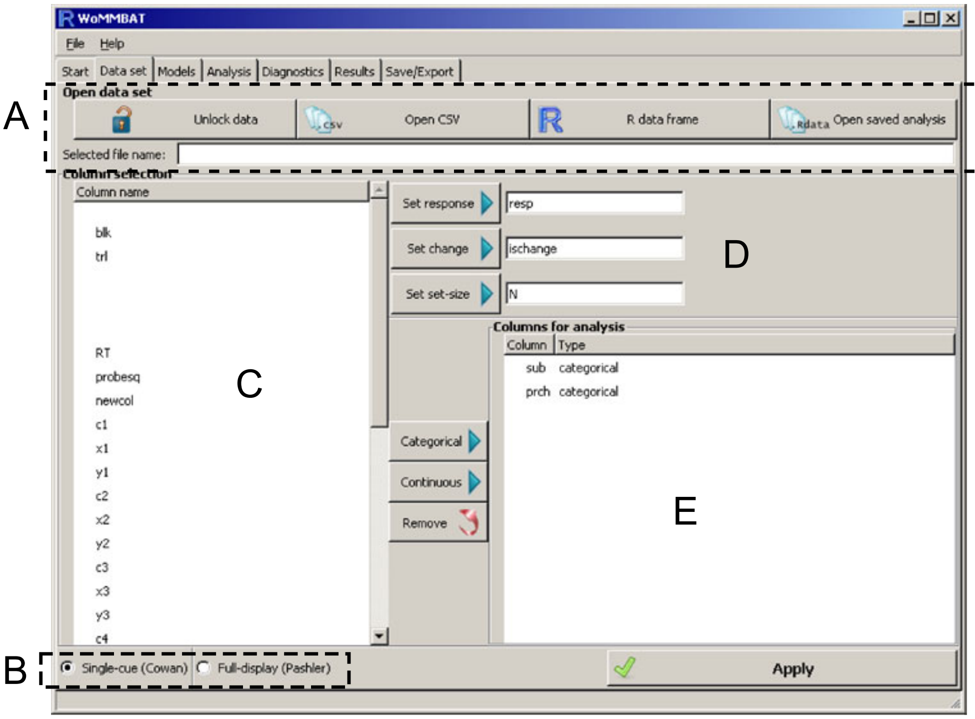
The WoMMBAT software is copyrighted 2010 by Richard D. Morey. The software is free; the source code is available under the Gnu Public License, v2.0. For the most current citation information, contact Richard D. Morey at r.d.morey@rug.nl.

The citation for the hierarchical Bayesian model underlying the software is:
Morey, R. D. (in press). A hierarchical Bayesian model for the measurement of working memory capacity. Journal of Mathematical Psychology.

The article to cite for the WoMMBAT software is currently in preparation:
Morey, R. D. and Morey, C. C. (in preparation). WoMMBAT: a graphical user interface for estimating working memory capacity in change detection data.

Fig. 5 The WoMMBAT starting tab. **a**: The tab interface highlighted by the dashed rectangle may be used to move back and forth between screens

ANOVA. For factors, WoMMBAT will estimate the effect of each factor level. Continuous predictors, on the other hand, allow for regression slopes to be estimated. The WoMMBAT software will automatically select the appropriate model given the predictor type. For this demonstration, include the probability change variable `prch` as a categorical predictor. Figure 6 shows how the data tab window will look after the columns have been specified for our analysis of the Rouder et al. (2008) data set. Clicking the "Apply" button in the lower right will lock the data, preventing further changes to the columns of interest, and will advance the analysis to the models tab.



Set response: resp
Set change: ischange
Set set-size: N

Columns for analysis

Column	Type
sub	categorical
prch	categorical

● Single-cue (Cowan) ○ Full-display (Pashler)

Fig. 6 The WoMMBAT data tab. **a**: Buttons to unlock loaded data (see the text), load a CSV file, load an R data frame, or load a previously saved analysis. **b**: Selection for the appropriate capacity model. **c**: List of available data columns. **d**: Required columns include participant response, whether the stimulus changed, and the set size. **e**: Selection of factors (categorical) and covariates (continuous) to be analyzed

Table 1 It is necessary to specify which data columns contain the response, change, and set size information for each trial

Column	Description	Contents
Response	Whether the participant responded "change"	0 = "no change", 1 = "change"
Change	Whether the trial was a change trial	0 = no change, 1 = change
Set size	Number of to-be-remembered elements	Integer > 0

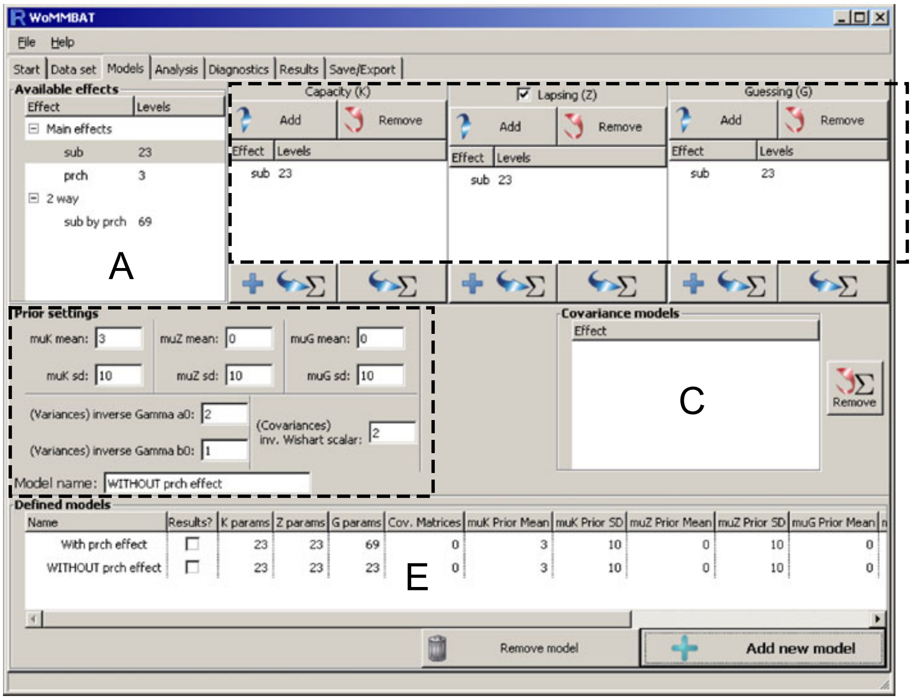
Models tab

After loading the data and specifying the columns of interest, the next step is to build a model using the specified columns. Figure 7 shows the model specification window, which has five main sections: the available effect list (A), the model on each parameter (B), covariance models (C), prior parameters (D), and the list of defined models (E).

The effect list (Fig. 7A) shows every main effect and interaction effect that can be constructed from the specified columns of interest. The goal in the model tab is to build models to help answer our research questions. For the purposes of this demonstration, we build two models that will enable us to answer our question of interest: whether the change probability manipulation had an effect on guessing biases.

As a starting point, we build a model similar to that fit by Rouder et al. (2008). The Rouder et al. (2008) model had five parameters for each participant: one fixed capacity (K), one lapse parameter (Z), and one guessing parameter for each probability change condition. To build this model in the WoMMBAT interface, highlight the main effect sub (Fig. 7A) and click the "Add" button under the K parameter (Fig. 7). Do the same for the Z parameter. This gives every participant his or her own capacity and lapsing parameter. To add a parameter to G for each participant by change probability condition combination, we add the two-way interaction sub by prch. The resulting model, with a main effect of sub on K and Z, and the sub by prch interaction on G, is a Bayesian hierarchical version of the five-parameter model fit by Rouder et al. (2008). We add this model to the list of models to be analyzed by labeling it with a descriptive name (Model name, Fig. 7D) and clicking the "Add new model" button in the lower right.

Our question of interest is whether the change probability manipulation affected guessing bias. In the model defined above, guessing bias G is free to vary. It is possible to frame the question of interest in several ways. First, we could compare the three guessing bias parameters for each participant and see by how much they differ. With 69 total parameters, this would be a tedious process. A second way of framing the question is whether a model that does *not* include an effect of change probability fits the data as well as one that does include an effect of change probability. We defer the specifics of assessing model fit until later; for now, we will discuss building a model that does not include an effect of probability change. We may then compare the first model, which



Available effects

Effect	Levels
⊟ Main effects	
sub	23
prch	3
⊟ 2 way	
sub by prch	69

Capacity (K)

Effect	Levels
sub	23

Lapsing (Z)

Effect	Levels
sub	23

Guessing (G)

Effect	Levels
sub	23

Prior settings

muK mean: 3 muZ mean: 0 muG mean: 0

muK sd: 10 muZ sd: 10 muG sd: 10

(Variances) inverse Gamma a0: 2

(Variances) inverse Gamma b0: 1

(Covariances) inv. Wishart scalar: 2

Model name: WITHOUT prch effect

Covariance models

Effect

Defined models

Name	Results?	K params	Z params	G params	Cov. Matrices	muK Prior Mean	muK Prior SD	muZ Prior Mean	muZ Prior SD	muG Prior Mean
With prch effect	☐	23	23	69	0	3	10	0	10	0
WITHOUT prch effect	☐	23	23	23	0	3	10	0	10	0

Fig. 7 The WoMMBAT model building tab. **a**: All possible main effects and interactions available for analysis. **b**: Models on each parameter. **c**: Covariance matrix models. **d**: Prior parameters and model name. **e**: The list of all defined models

contains a change probability effect, with the second model, which does not. If the model with an effect of change probability fits better than the model without, we conclude that a change probability effect is necessary to explain the data and, thus, that change probability has an effect on guessing biases.

We can build a model without change probability by simply removing the two-way interaction on *G* (click the "Remove" button under *G*, while the interaction is highlighted; Fig. 7B) and adding a main effect of particpant.[3] Under this model, each participant has his or her own guessing bias parameter, but it is not affected by the change probability manipulation. The final step in defining the model is to name it, and click the "Add new model" button.

After defining both models, the model tab should look similar to Fig. 7. Both models should appear in the model list (Fig. 7E). Double-clicking on a model in the model list will copy the model specifications back into the model-building interface, which is useful for modifying already-specified models or for recalling the exact specification of a model.

There are aspects of the model-building interface in Fig. 7 not discussed above: covariance matrices (C) and prior settings (D). The covariance matrix interface can be used to model the covariance between two vectors of parameters that are naturally paired with one another. For instance, the sub main effect on *K* and the sub main effect on *Z* form a natural pair, since they refer to different parameters for the same participant. It might be interesting to ask whether these parameters covary. That is, do participants with low lapsing (high *Z*) also have high capacity (*K*)? To answer this question, we might add both parameters to the covariance modeling window. We would use the buttons labeled with Σ below each parameter model window. The +Σ button adds an effect to a new covariance group, and the Σ button adds an effect to an existing, highlighted covariance group. Covariances and correlations will be modeled between all parameters in one group. Note that a necessary condition for two parameter vectors to be naturally paired is that they have the same number of elements; WoMMBAT will not allow two vectors of parameters with different numbers of parameters to be added to the same covariance group. Having the same number of parameters, however, does not guarantee that the pairing makes sense; parameter vectors may have the same number of parameters by coincidence.

Figure 7 also shows the prior settings interface (E), which allows users to change the prior settings used for analysis of the model. Interested readers may see R. D. Morey (2011) for more details about the priors and may also change the default values to examine their effects on model fits. For the present demonstration and for most other analyses, the default values will serve.

Analysis tab

After defining all the models of interest, the next step is obtaining parameter estimates. As was mentioned previously, in WoMMBAT parameter estimates are obtained using Markov chain Monte Carlo (MCMC) techniques. The analysis tab allows the specification of MCMC settings and starting MCMC sampling for the models defined under the models tab.

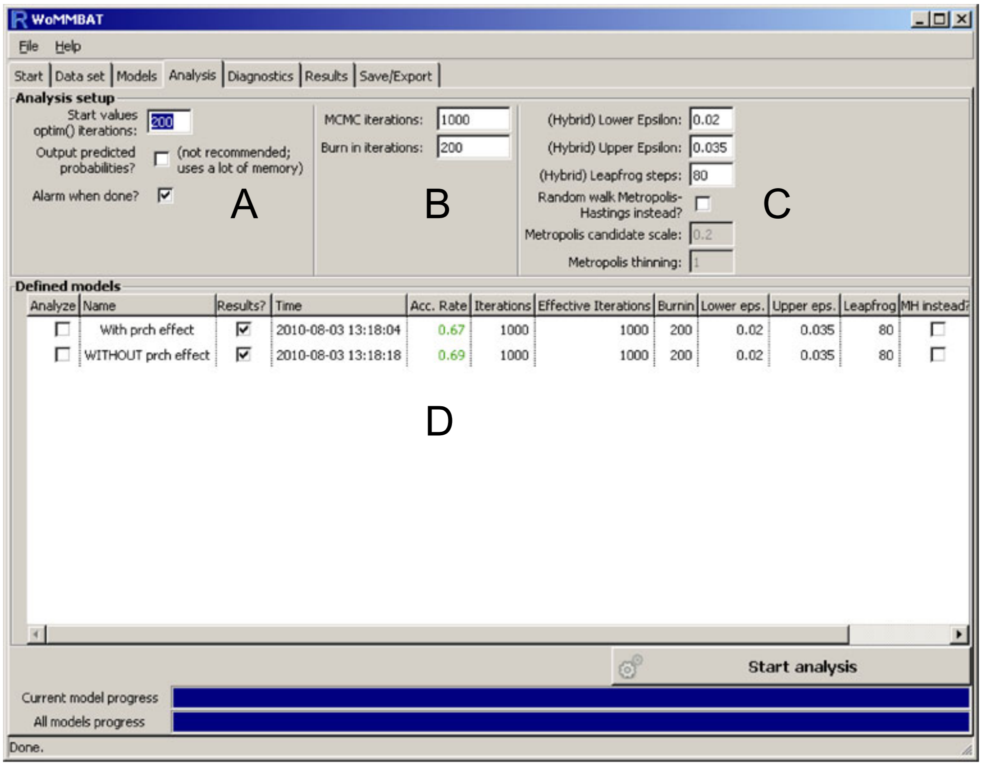
The MCMC techniques used by WoMMBAT draw approximate samples from the joint posterior distribution of all parameters. After each sample is drawn, the sample is then used to draw a new sample, yielding a chain of values. As the chain gets longer, the distribution of the samples becomes a better approximation to the joint posterior.

The quality of the samples is controlled by several factors. For the purposes of this demonstration analysis, we will describe them only briefly; for more depth, there are several introductions to Bayesian analysis with MCMC, some written specifically for psychologists (Ntzoufras, 2009; Rouder & Lu, 2005; Wagenmakers et al. 2010). The first factor controlling the quality of the samples, as has already been mentioned, is the length of the chain. This is controlled by the "Iterations" setting (Fig. 8B). The second is the number of burn-in iterations. *Burn-in* refers to throwing away a number of iterations at the beginning of the chain, to make the chain less sensitive to the initial conditions of the chain. For our demonstration analysis, the default values for number of MCMC and burn-in iterations will suffice. Although the 1,000-iteration default is not enough to obtain precise estimates, they are enough for an initial analysis. Increasing these values leads to more precise (but slower) analyses.

The quality of the samples is also affected by more specific settings of the MCMC sampler (Fig. 8C). The default MCMC method in WoMMBAT is called *hybrid Monte Carlo* (Liu, 2001); for a description of hybrid Monte Carlo, how these settings affect the quality of the MCMC chain, and tips for improving the quality of hybrid Monte Carlo chains, see Appendix 3 to this article or the supplement for R. D. Morey (2011). For the purposes of this demonstration analysis, the default values will be sufficient.

Several less important settings are shown in Fig. 8A. The "Start values optim iterations" settings determine how long the algorithm searches for starting values for MCMC.

[3] There is a third model of intermediate complexity between the two models defined here. One could also add a main effect of `sub` and a main effect of `prch` on *G*. Under this model, the effect of the change probability manipulation is identical for all participants. This model fits much worse than the two models defined here and, so, was omitted for simplicity.



Analysis setup

Start values optim() iterations: 200

Output predicted probabilities? □ (not recommended; uses a lot of memory)

Alarm when done? ☑

MCMC iterations: 1000

Burn in iterations: 200

(Hybrid) Lower Epsilon: 0.02

(Hybrid) Upper Epsilon: 0.035

(Hybrid) Leapfrog steps: 80

Random walk Metropolis-Hastings instead? □

Metropolis candidate scale: 0.2

Metropolis thinning: 1

Defined models

Analyze	Name	Results?	Time	Acc. Rate	Iterations	Effective Iterations	Burnin	Lower eps.	Upper eps.	Leapfrog	MH instead?
□	With prch effect	☑	2010-08-03 13:18:04	0.67	1000	1000	200	0.02	0.035	80	□
□	WITHOUT prch effect	☑	2010-08-03 13:18:18	0.69	1000	1000	200	0.02	0.035	80	□

Fig. 8 The WoMMBAT analysis tab. **a**: Miscellaneous analysis settings. **b**: General MCMC settings. **c**: Specific (hybrid Monte Carlo/random walk metropolis) MCMC settings. **d**: Analysis report for each model

WoMMBAT uses numerical optimization to search for the posterior mode (the most–likely parameter values, given the data), before performing any MCMC. If MCMC is started from highly–likely parameter values, the MCMC algorithm will produce higher quality chains. If the chains appear to be of low quality initially, increasing this value by a factor of 1.5 may help. The “Output predicted probabilities” setting will yield the model’s posterior predicted probability that a response on a given trial will be “change.” These predictions can be compared with the data themselves to compare the fit of the model. How this may be done is beyond the scope of the present article, but interested readers should consult Chap. 6 of Gelman et al. (2004) or Gelman, Goegebeur, Tuerlinckx, & van Mechelen, (2000).

Note that in the list of models to be analyzed, the first column, “Analyze,” is checked for both models. Any model with the “Analyze” column checked will be analyzed using the settings at the top of the analyze tab when the “Start analysis” button is clicked. Click the button to start the analysis, and the progress bars at the bottom will show how much of the analysis has been completed. The analysis may take anywhere from a few seconds to a minute to complete, depending on the speed of the computer on which the analysis is run. Once the analysis is finished, note that the “Analysis” column is unchecked for both models and the “Results” column is checked for both models, indicating that parameter estimates and model fit statistics are available. Before checking the parameter estimates and fit statistics, it is important to diagnose the quality of the MCMC chains obtained from the analysis.

The first step in diagnosing the MCMC chains is to check the sample acceptance rate, which is the fifth column in the model list on the analysis tab. As was mentioned previously, MCMC chains are created by using a sample from the joint posterior to obtain a new sample. In MCMC chains, it is important that the new samples be far enough away from the last sample to ensure that the parameter space is adequately explored, but not too far; samples that are too far are likely to be unreasonable and, thus, rejected. The acceptance rate statistic shows how often a new sample was accepted. For hybrid Monte Carlo, good acceptance rates are between 0.6 and 0.8. Acceptance rates within these bounds are colored green. Acceptance rates inside these bounds will not necessarily produce good chains (further diagnostics will be discussed later), and chains with acceptance rates outside these bounds may yield good quality estimates if the chain is long enough; in general, however, acceptance rates of between 0.6 and 0.8 yield good estimates with fewer iterations. Figure 8 shows acceptance rates of 0.67 and 0.69. Due to the random nature of MCMC, your results will likely differ slightly from these.

In the process of analyzing other data, it is possible that the default values will not yield acceptance rates between 0.6 and 0.8. In this case, the hybrid Monte Carlo settings must be tweaked. Step-by-step instructions for obtaining quality samples are outlined in

Appendix 3 of this article and in the supplement to R. D. Morey (2011).

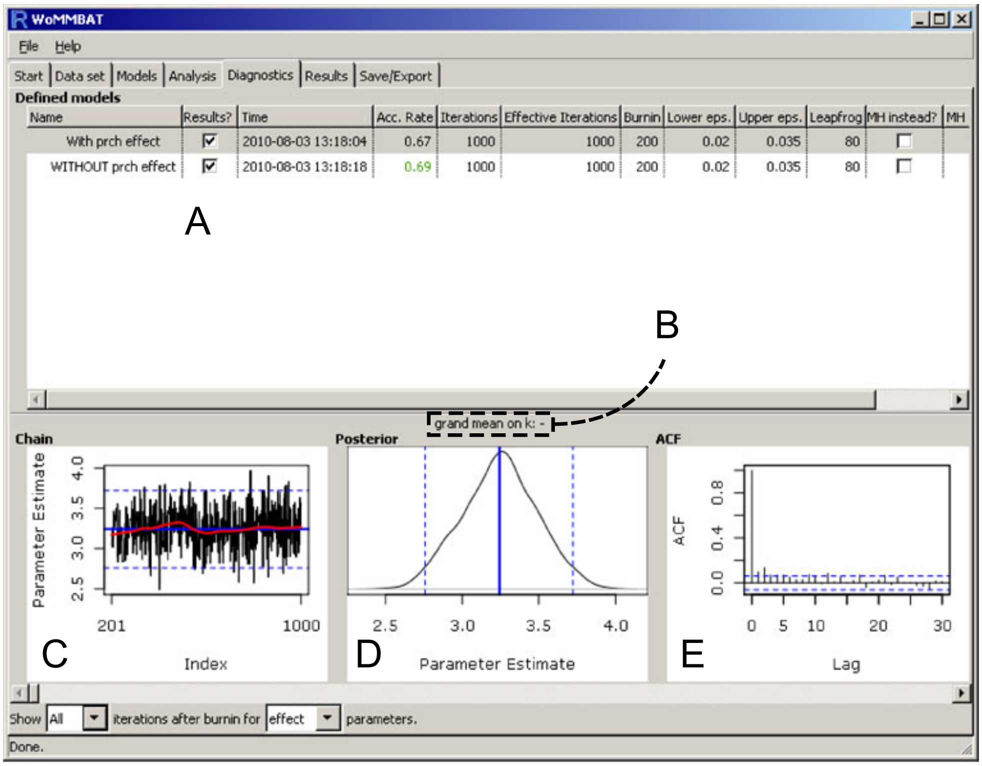
Diagnostics tab

Once MCMC chains have been obtained with good acceptance rates, it is necessary to diagnose the MCMC chains to determine whether they will yield good parameter estimates. The diagnostics tab (Fig. 9) allows the analyst to check the quality of the MCMC chains. The goal of MCMC analysis is to produce a long enough MCMC chain that the chain can be said to have *converged*—that is, that the samples can be treated as if they were true samples from the desired posterior distribution. MCMC methods are guaranteed to produce chains that converge, given an infinitely long chain; in practice, however, analysts must settle for finitely–long chains. Since MCMC chains are random, there is no foolproof method for assessing convergence, but there are a number of methods that, when applied to MCMC chains, can give the analyst confidence in the results of the MCMC analysis.

There are two primary methods for assessing convergence: graphical and quantitative. Graphical methods are easy to use with only a little training and provide most of the information that quantitative methods provide in a fraction of the time and computational effort. However, they are subjective; the analyst must make an informed decision about the convergence without the aid of a number on which to base the decision. In contrast, quantitative methods use numbers that provide an objective (although not complete) assessment of convergence. The disadvantages of quantitative methods are that there are many such statistics to choose from; they are computationally intensive, do not reveal the reason for the lack of convergence, and offer much of the same information as would be contained in a plot. For simplicity, WoMMBAT offers only graphical methods of convergence. Appendix 2 to this article discusses the computation of quantitative measures of convergence outside the WoMMBAT interface, using R.

Convergence for each parameter within each model is assessed individually. Although this may seem tedious, since models sometimes have hundreds of parameters, the WoMMBAT interface offers a fast method for assessing chain convergence, one after another. The first step is selecting the model whose chains will be assessed for convergence in the model list (Fig. 9A). Upon selecting a model, three plots will appear below the model list (Fig. 9C–E), each of which isa separate diagnostic plot for the named parameter (Fig. 9B). The first parameter chosen is always the grand mean capacity, $\mu^{(\kappa)}$. These three plots are the MCMC chain for the parameter plotted as a function of MCMC iteration (C), the kernel density estimate of the marginal posterior distribution for the parameter (D), and the autocorrelation function (ACF) of the MCMC chain (E).

The MCMC chain plot in Fig. 9 (C) is the plot that is perhaps most diagnostic of convergence. The chain plot is a



Defined models

Name	Results?	Time	Acc. Rate	Iterations	Effective Iterations	Burnin	Lower eps.	Upper eps.	Leapfrog	MH instead?	MH
With prch effect	☑	2010-08-03 13:18:04	0.67	1000	1000	200	0.02	0.035	80	□	
WITHOUT prch effect	☑	2010-08-03 13:18:18	0.69	1000	1000	200	0.02	0.035	80	□	

Show All iterations after burnin for effect parameters.

Fig. 9 The WoMMBAT analysis diagnostics tab. **a**: List of defined models. **b**: Name of current parameter whose MCMC chain is plotted. **c**: Plot of the MCMC chain by iteration. **d**: Kernel density plot of the posterior, from the MCMC chain. E: Autocorrelation plot for the MCMC chain

plot of the samples obtained from the MCMC analysis as a function of iteration. The ideal MCMC chain looks like random noise; no trends should be evident from either the chains or the lowess line. However, chains will differ in quality from analysis to analysis. There are four horizontal lines: The solid blue line is the mean of the chain; the red line is a lowess nonparameteric regression line (Cleveland, 1981), which reveals trends in the chain; the two dashed blue lines show the limits of the 95% credible interval.

The left column of Fig. 10 shows 1,000 iterations of four MCMC chains that differ in quality. Chain (A) looks square, because it contains long runs of iterations where a new sample was not accepted. This resulted in a low acceptance rate (18%) and a slowly converging chain. Chain (C) has quite the opposite problem as chain (A); new values were accepted too often, because they were too similar to the old sample. This leads to a slowly wandering chain and bad convergence. Chains (A) and (C) can be improved by tweaking the hybrid Monte Carlo parameters in the analysis tab. Chain (E) looks, at first glance, like a low-quality chain, with moderate autocorrelation. Upon close examination, however, it is obvious that the chain has two states, a high state and a low state, and that the chain oscillates between them (see the effect of participant 19 on *K* for the "With prch effect" model using the diagnostics tab for an example of this type of chain). The reason for oscillation in this case is that the participants' performance is so low that there are two explanations of their bad

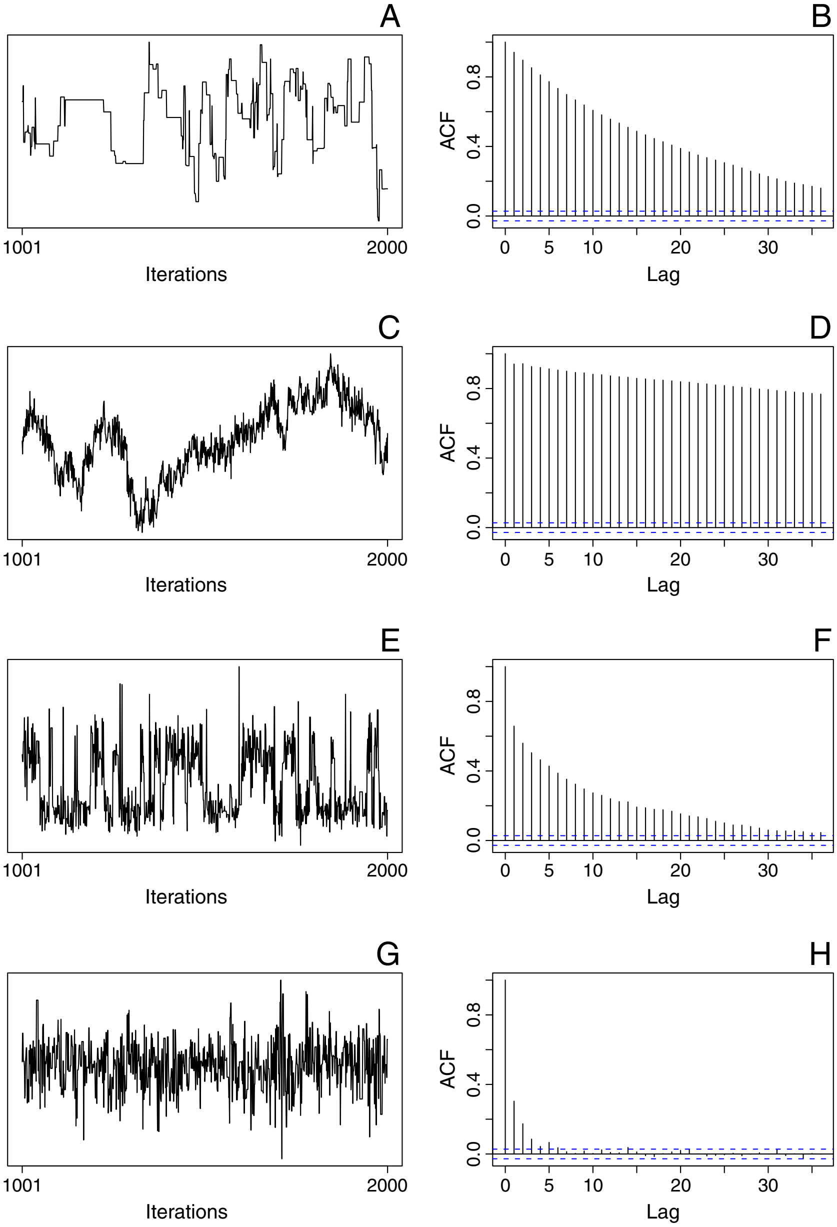


Fig. 10 MCMC chains of varying quality (left), and their respective autocorrelation plots (right). **a, b** MCMC chain with low acceptance rate (18%). The chain looks "square" because new samples were not accepted often enough. **c, d** MCMC chain with extremely high autocorrelation. **e, f** A good MCMC chain, oscillating between two states. **g, h** A good MCMC chain

performance: low attention and low capacity. The Bayesian model makes a compromise, sometimes choosing one solution and sometimes choosing another. This will result in a bimodal posterior distribution for this parameter. Since the bimodality is an effect of the model, there is no way to tweak the hybrid Monte Carlo parameters to improve the chain. However, running a very long MCMC chain will ensure that the chain spends enough time in both states that accurate estimates of the posterior distribution can be obtained. Chain (G) is a good chain; there appear to be no systematic trends. Chains similar to chain (G) are ideal for analysis.

Figure 9 (D) shows the kernel density estimate, a kind of smoothed histogram, of the marginal posterior distribution for the selected parameter. The vertical solid line represents the posterior mean estimated from the MCMC chain; the left and right vertical dashed lines represent the upper and lower bounds of the posterior credible interval. The kernel density plot is perhaps the least useful as a chain diagnostic, but it does allow one to see the marginal posterior density for the given parameter, which is the target of MCMC analysis.

The final graphical diagnostic is the ACF plot, shown in Fig. 9 (E). The ACF plot shows the amount to which one iteration in the chain is correlated with the following iterations. Ideally, iterations would be independent from one another, since independent samples provide more information per sample. However, in MCMC, independence is almost never achieved. In quality chains, dependence from one sample to the next is minimized but never eliminated. Along the *x*-axis in the ACF plot is the lag, which is the distance from any given sample at which the dependence is to be assessed. The lag between a sample and the next is 1, and so forth. The *y*-axis shows the average correlation. The horizontal dashed lines show the 95% limits on the expected correlation, given that independence holds. For highly dependent chains, which are bad-quality chains, the correlation will drop off slowly as a function of lag; for less dependent, higher-quality chains, the ACF will drop off faster. For the ideal independent chain, the correlation will be 0 for all lags. The ACF plot shown in Fig. 9(E) shows a well-behaved chain, where the autocorrelation drops to zero quickly and stays within the 95% bounds thereafter.

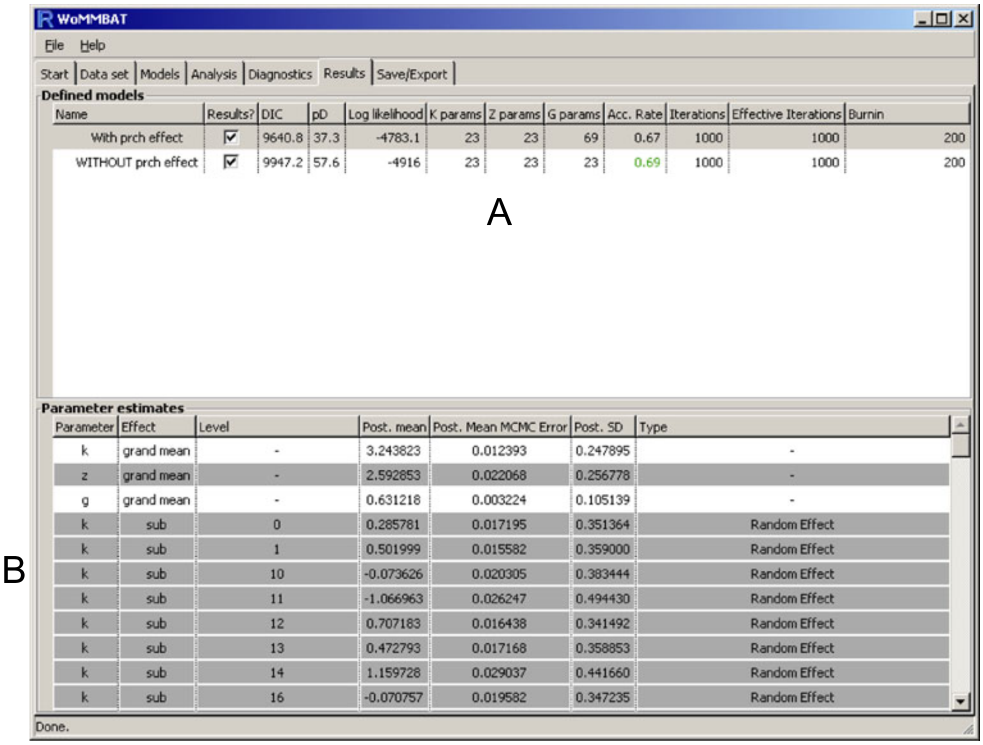
When combined with the total number of iterations, the ACF plot can give the analyst a rough idea of the effective sample size from the posterior. Consider the lag iterations needed for the correlation between two samples to be effectively 0; suppose, for the sake of demonstration, that 10 iterations are needed for the correlation to be essentially 0. If the MCMC chain were thinned by removing all intervening iterations, each sample would essentially be independent. This thinning strategy would keep every10th iteration. If we sampled 1,000 total MCMC iterations, our effective sample size would be 100 samples from the posterior distribution. This estimate is rough and conservative, since the removed iterations provided some information about the posterior distribution, even though they were correlated with one another. However, the rough estimate demonstrates how the information in the ACF plot might be used to determine the reliability of MCMC analyses.

Together, the three plots in the diagnostic tab give a view of how well the MCMC chains are likely to approximate the true posterior distribution for a given parameter. But not all parameters converge at the same rate; it is important to look through all the parameters, to assess the quality of the estimates. The WoMMBAT chain diagnostic interface makes examining the chains for many parameters easy; the scrollbar below the three plots, when dragged from left to right, will change the parameter being examined in the plots. If the interface is slow, due to the number of iterations being plotted being high, the analyst may reduce the number of iterations shown in the chain plot (C) by selecting a smaller number of iterations in the drop-down box under (C). The resulting effect is like a flip-book; many parameters can be diagnosed quickly, and any parameter that catches the eye as being problematic can be examined more closely by stopping on that parameter.

If the analyst is unhappy with the convergence of any parameters, there are two main options available. First, the user can return to the analysis tab and rerun the analysis after tweaking the hybrid Monte Carlo parameters according to the instructions in Appendix 3. If better hybrid Monte Carlo parameters can be found, the convergence of all parameters will improve, sometimes markedly. Alternatively, the analyst may simply choose to rerun the analysis with more iterations. Since convergence is ensured as the number of iterations increases, increasing the number of iterations will be more likely to yield convergent chains. However, for the purposes of exploring a dataset, a low to moderate number (1,000–5,000) of iterations is usually sufficient to get an idea of which models fit and which do not, if the MCMC chains are wellbehaved. For finer-grained analyses sufficient for publication, longer chains (relative to the amount of autocorrelation) are recommended.

Results tab

After the analyst is comfortable with the quality of the MCMC chains, model fit statistics and parameter estimates can be examined. The results tab (Fig. 11) offers an interface for comparing the model fit of all models simultaneously (A), examining parameter estimates (B), and making pairwise comparisons between parameters.



Defined models

Name	Results?	DIC	pD	Log likelihood	K params	Z params	G params	Acc. Rate	Iterations	Effective Iterations	Burnin
With prch effect	☑	9640.8	37.3	-4783.1	23	23	69	0.67	1000	1000	200
WITHOUT prch effect	☑	9947.2	57.6	-4916	23	23	23	0.69	1000	1000	200

Parameter estimates

Parameter	Effect	Level	Post. mean	Post. Mean MCMC Error	Post. SD	Type
k	grand mean	-	3.243823	0.012393	0.247895	-
z	grand mean	-	2.592853	0.022068	0.256778	-
g	grand mean	-	0.631218	0.003224	0.105139	-
k	sub	0	0.285781	0.017195	0.351364	Random Effect
k	sub	1	0.501999	0.015582	0.359000	Random Effect
k	sub	10	-0.073626	0.020305	0.383444	Random Effect
k	sub	11	-1.066963	0.026247	0.494430	Random Effect
k	sub	12	0.707183	0.016438	0.341492	Random Effect
k	sub	13	0.472793	0.017168	0.358853	Random Effect
k	sub	14	1.159728	0.029037	0.441660	Random Effect
k	sub	16	-0.070757	0.019582	0.347235	Random Effect

Fig. 11 The WoMMBAT results tab. **a**: List of defined models, with fit statistics. **b**: Posterior estimates for each parameter, along with error estimates. Highlighting these will copy them to the clipboard; a double left-click will open the multiple comparisons window for the selected parameter

Bayesian hypothesis testing can be viewed in terms of model selection. In order to explore the question of whether the change probability manipulation affected guessing biases, we constructed two models. In the first model, change probability is included in a two-way interaction affecting the *G* parameter. In the second model, change probability does not appear. Answering our question of interest amounts to selecting which of these two models best accounts for the data. In order to assess the models' ability to account for the data, WoMMBAT computes a statistic called the *deviance information criterion* (DIC; Spiegelhalter, Best, Carlin, & van der Linde, 2002). DIC weighs the fit of a model against the complexity of a model in a manner similar to the Akaike information criterion (AIC; Akaike, 1974). The basic idea is that the quality of fit of the model is penalized by the complexity of the model. In AIC, the flexibility of the model is quantified by the number of parameters. In hierarchical models, however, the complexity of a model cannot be directly assessed by the counting parameters, because the fact that parameters come from parent distributions adds additional constraint to hierarchical models that is difficult to quantify. The DIC thus relies on an estimate of the number of parameters (called P_D), which is used as a measure of model complexity or flexibility. Models with lower DICs are preferred over those with higher DICs, after weighing both model fit and model flexibility.

The results tab's model selection list (Fig. 11A) displays every model defined, along with the DIC and P_D computed for each. Clicking on the "DIC" label in the DIC column will sort the models by DIC and will allow the analyst to easily assess the relative fit of all models at once. In the case of the two models we have defined, the model including an effect of change probability on guessing bias (9,641) is preferred by about 300 DIC points to the model without (9,947). Because the DIC

Table 2 Description of the parameter estimate columns in the WoMMBAT results interface

Column	Description
Parameter	Which working memory model parameter the effect is on
Effect	Which factor the parameter is a level of
Level	The level of the factor of which the effect is an estimate
Post. mean	Posterior mean of the effect (the point estimate)
Post. mean MCMC error	An estimate of the uncertainty in the posterior mean, due to MCMC error
Post. *SD*	Posterior standard deviation; represents uncertainty in the parameter estimate
Type	Parameter type; either slope or random effect

statistic is computed from the MCMC chains, it is random, and thus your DIC will differ slightly from these. However, given the size of this difference, your result will not differ substantially. To put this difference in perspective, DIC is computed on a log scale; thus, the difference of 300 points can also be thought of as a ratio of $e^{300} \approx 2 \times 10^{130}$. The amount of support in favor of the model including an effect of change probability on guessing bias is overwhelming.

A second way to assess the question of whether guessing bias is affected by the change probability manipulation is to look at the posterior mean estimates of the guessing parameters. These parameters are shown in the lower half of the results tab (Fig. 11B). Select the model including the two-way interaction of participant with change probability on G in the model list. All parameter estimates will appear in the parameter list below. Each row represents a parameter; the columns contain information about the parameter and its estimate (see Table 2 for a brief description of each column). The parameters of interest, the interaction effects on guessing, may be found by scrolling down in the list of parameters. By adding the grand mean guessing bias parameter (the parameter in row 3) to each of the interaction effects, we obtain the guessing bias estimates for each change probability condition, for each participant, a total of 3 × 23 = 69 parameters.

Figure 12 shows the posterior mean of the guessing parameter for each participant by change probability condition, with each line representing a participant. To create this plot, the grand mean g parameter ($\mu^{(g)}$; third row in the parameter list), was added to each interaction effect, which may be found by scrolling down in the parameter list.[4] As Fig. 12 shows, participants' guessing biases increase as the change probability increases. This increase can be seen in all participants, although the size of the effect differs. The conclusions reached by way of the parameter estimates are highly consistent with the large DIC difference.

Another useful feature of the results tab is the ability to perform Bayesian multiple comparison analyses. For the purpose of demonstration, suppose that we were interested in the difference between the levels of the `sub by prch` interaction on guessing bias for the first participant, participant "0." If we highlight the row in the parameter list corresponding to the interaction effect for level `0.x.0.847297860387203` and then double-click, the multiple-comparisons interface will open (Fig. 13 and Table 3). The multiple comparison window allows the comparison of all other factor levels to the selected factor. In order to perform multiple comparisons, WoMMBAT computes the posterior probability (and odds) that the selected factor level's parameter (A) is greater than each of the other factor levels' parameters (B). Both high and low posterior probabilities yield evidence that the parameters are different. For instance, the posterior probability that the guessing parameter for participant "0" in the change probability 0.7 condition is greater than the one in the change probability 0.3 condition is about 0.98. The results from your own analysis may differ, due to the randomness of MCMC methods, but with longer chains the probability will converge to its true value. The multiple comparison analysis is consistent with Fig. 12; the guessing biases are highest in the condition where the probability of a stimulus change was highest.

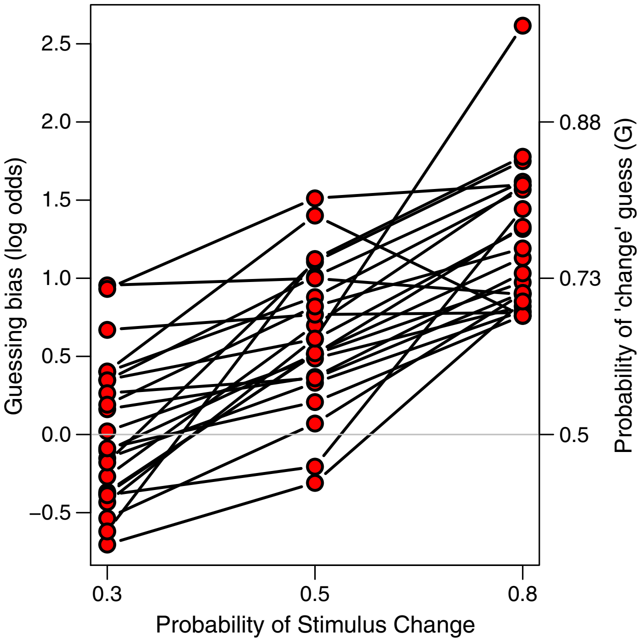
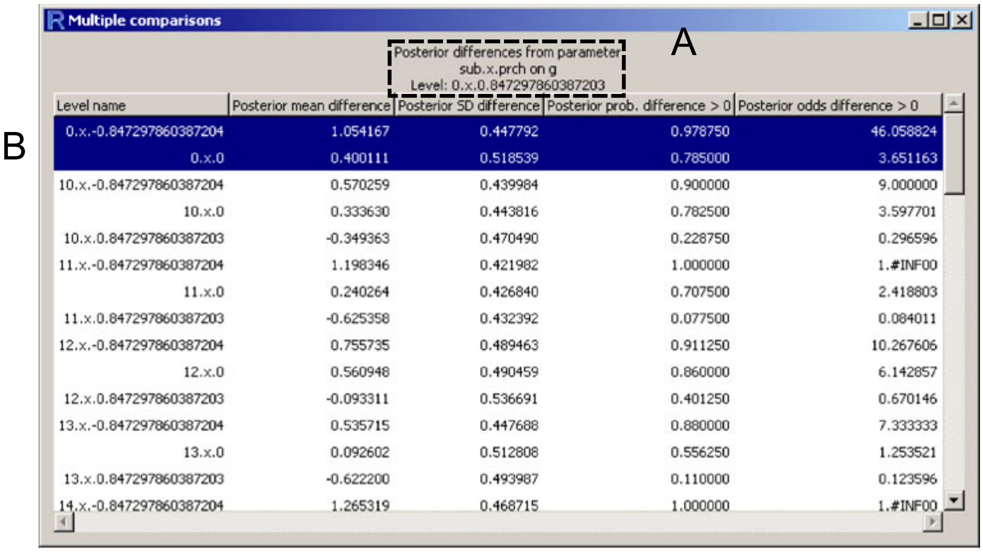


Fig. 12 Posterior mean guessing bias parameters as a function of true change probability. Each line represents a participant

Save/Export tab

Because analysis of models using MCMC takes time, it is useful to save the results of an analysis so that an analyst can return to the analysis later without redefining the models or rerunning the analyses. The save tab allows analysts to save their analyses in two ways: first, as an R data file that can be reloaded via the data set tab, and second, as text files that can be loaded into other statistical software. Because MCMC creates a lot of numbers, the text files can be quite large and take time to save; it is

[4] Highlighting the rows corresponding to the interaction effects will copy them into the operating system clipboard. From the clipboard, they may be pasted into another program, like Microsoft Excel, for easy plotting. For users familiar with R, the estimates may be extracted and plotted within R. See Appendix 1 for details.



Posterior differences from parameter
sub.x.prch on g
Level: 0.x.0.847297860387203

Level name	Posterior mean difference	Posterior SD difference	Posterior prob. difference > 0	Posterior odds difference > 0
0.x.-0.847297860387204	1.054167	0.447792	0.978750	46.058824
0.x.0	0.400111	0.518539	0.785000	3.651163
10.x.-0.847297860387204	0.570259	0.439984	0.900000	9.000000
10.x.0	0.333630	0.443816	0.782500	3.597701
10.x.0.847297860387203	-0.349363	0.470490	0.228750	0.296596
11.x.-0.847297860387204	1.198346	0.421982	1.000000	1.#INF00
11.x.0	0.240264	0.426840	0.707500	2.418803
11.x.0.847297860387203	-0.625358	0.432392	0.077500	0.084011
12.x.-0.847297860387204	0.755735	0.489463	0.911250	10.267606
12.x.0	0.560948	0.490459	0.860000	6.142857
12.x.0.847297860387203	-0.093311	0.536691	0.401250	0.670146
13.x.-0.847297860387204	0.535715	0.447688	0.880000	7.333333
13.x.0	0.092602	0.512808	0.556250	1.253521
13.x.0.847297860387203	-0.622200	0.493987	0.110000	0.123596
14.x.-0.847297860387204	1.265319	0.468715	1.000000	1.#INF00

Fig. 13 The WoMMBAT multiple comparisons window. **a**: The effect, and level of the effect, selected for multiple comparisons. In this case, it is the interaction effect of participant 0 when the log-odds of a stimulus change was 0.85 (probability of 0.7). **b**: Two factor levels are selected: the interaction effect for participant 0 when the log-odds of a change are −0.85 and 0, respectively (probabilities of 0.3 and 0.5). The posterior odds that guessing is higher in the comparison condition (**a**) than in the selected conditions (**b**) are 46:1 and 3.6:1, respectively

recommended that saving to text files be done only when necessary.

Discussion: Building and testing many models

In the demonstration above, we entertained two models, with the purpose of performing a single test. By comparing the DIC statistics for the two models, we were able to clearly reject one of the models, leading to an answer to our question of interest: The change probability manipulation had an effect on guessing biases. In practice, however, most researchers have a more complicated design than the design in the Rouder et al. (2008) experiment, with many more factors. An experiment with five factors would not be unusual. In this case, there is 1 five-way interaction, 5 four-way interactions, 10 three-way interactions, 10 two-way interactions, and five main effects. It is obvious that the number of possible models is quite large, especially considering that we can put different models on *K*, *Z*, and *G*. ANOVA methods offer the convenience of testing all effects simultaneously; unfortunately, this is not possible when using Bayesian hierarchical models. The process of model building and testing in WoMMBAT is analogous to stepwise regression (Hocking, 1976). In stepwise regression, a set of regression predictors is sought that best explains the data. However, in stepwise regression, as in the hierarchical model outlined above, there are often too many predictors to test every combination. We have found the stepwise regression technique of backward selection to be of use in finding models that fit the data well.

In backward selection, we begin with a complicated, flexible model and fit progressively simpler models. Eventually simplification yields progressively worse fits, as measured by DIC. The simplification process works by starting with interactions and breaking those interactions into smaller interactions and main effects. If eliminating a factor increases the DIC, the factor is necessary to explain the data. If eliminating a factor decreases the DIC, it is not.

One possible issue with starting with complex models is the sheer size of the analyses it produces. Consider a 2 × 2 × 4 × 5 design, with 20 participants. The full five-

Table 3 Description of the columns in the WoMMBAT multiple comparisons interface

Column	Description
Level name	Name of the factor level being compared to
Posterior mean difference	Mean of the posterior difference between the two parameters
Posterior *SD* difference	Standard deviation of the posterior difference between the two parameters
Posterior prob. difference >0	The posterior probability that the difference is greater than 0; equivalently, the posterior probability that the selected factor level is greater than this row's factor level
Posterior odds difference >0	The posterior odds that the difference is greater than 0; equivalently, the posterior odds that the selected factor level is greater than this row's factor level

way interaction (with participant) has 1,600 parameters. Assume that we are not concerned with the *Z* parameter, but we place the full five-way interaction on *K* and *G*. This yields a total of 3,200 parameters, which is quite an unwieldy analysis.[5]

There are, however, a number of ways to reduce the number of parameters to be fit.

- Avoid complicated models on *G*. Typically, guessing bias is not of interest to the researcher, and most experiments do not attempt to manipulate it. In our experience, a simple model including only the participant factor on *G* fits best, since factors intended to manipulate capacity do not typically manipulate guessing bias. Putting a simple model on *K* and then performing backward selection on *G* will almost surely lead you to a simple model on *G*, allowing you to concentrate on *K*.
- Only model *Z* if you are both able and interested. Modeling the *Z* parameter requires multiple set sizes—at minimum, one set size that is below capacity and one set size above capacity. If this requirement is not met, turn off the modeling of the *Z* parameter. If your design allows modeling of the *Z* parameter but you have no hypotheses regarding lapsing, use a simple model on *Z* that includes only the main effect of participant.
- Use participant as a main effect on *K*. By using participant as a main effect on *K*, and not in any interactions, you explicitly require the pattern of results to be the same for all participants. This will reduce the number of possible models by eliminating the interactions with the participant factor. If specific hypotheses regarding differences between participants are of interest, it is possible to test interactions with participants; however, for most analyses, participant as a main effect will be a useful simplifying assumption. Because a repeated measures ANOVA disallows the testing of interactions with participants, restricting participants to be a main effect is not any more constraining than traditional analyses.
- Use graphical methods to guide model building. In backward model selection, the analyst starts with the largest interaction and attempts to make the model simpler. Beginning with the largest interaction has an added benefit: The parameter estimates can easily be used to make plots. By plotting the effects of every factor level in the interaction (i.e., all the combinations of the factor levels; see Fig. 12), it will be clearer which factors are needed and which are not. Model building and testing can be focused on models that are suggested by the plots, whereas models highly inconsistent with the plots can be ruled out by shorter MCMC runs, because these models will likely have DICs that are substantially higher than models suggested by the plots.

Model selection in WoMMBAT is not an automatic process, but the guidelines above should enable analysts to restrict the number of models being tested to a reasonable number. Also, because analyses may be easily saved in WoMMBAT, questions about model fit that may arise after the initial analyses can be easily answered by loading the analysis, defining a new model, and testing it. The ability to save and send WoMMBAT analyses to other researchers makes the entire process transparent and open to replication.

Summary and conclusion

In this article, we introduced the WoMMBAT software for estimating working memory capacity from change–detection data. In the demonstration above, we showed how WoMMBAT can be used to fit R. D. Morey's (2011) hierarchical model to data, obtain parameter estimates, and perform statistical inference. Researchers wishing to analyze their own data need only change the details of the analysis; the essential elements of an analysis within WoMMBAT are all contained in this demonstration analysis.

Because R. D. Morey's (2011) hierarchical model has significant advantages over the traditional method of analyzing change detection data, the adoption of WoMMBAT as a standard data analysis tool would be a methodological advance in working memory research. The software is freely available and released under the Gnu Public License v2.0.

Acknowledgements We thank Jeffery Rouder and Stephan Lewandowsky for comments on earlier drafts.

Appendix 1

Extracting model information within R

For researchers familiar with R, or who are interested in learning to use R, it may be useful to be able to access WoMMBAT model analyses from the R console interface. This will be necessary for computing MCMC convergence statistics in the next section of this appendix, but it is also

[5] Incidentally, because each sample of each parameter uses 8 bytes of memory, 10,000 iterations of analysis for the model will be 256 megabytes. This is a sizable proportion of a modern computer's memory.

useful for researchers who want to access the MCMC chains to, for example, create plots.

The `WMCapacity` package includes a function for extracting all model information from WoMMBAT from the R console. This function, `womExtractModel()`, requires the name or number of a model to extract:

```
# Extract the model with the sub by prch
# interaction and save in a variable
# called “winningModel”
winningModel = womExtractModel(’With
prch effect’)
# Or this (same thing)
winningModel = womExtractModel(1)
# Check to see that we got the
# correct model
winningModel$modelName
# Should return:
#[1] “With prch effect”
```

This works because we called the winning model “With prch effect”; replace this name with whatever you called this model. Alternatively, we can use the number of the model in the model tab list.

After assigning the model analysis to the `winningModel` variable (or whatever other variable name we choose), we may access any aspect of the model analysis by accessing a particular element in `winningModel`. Table 4 shows a subset of the more interesting elements of an extracted model, along with a description of what the elements contain.

Appendix 2

Computing convergence statistics within R

It is possible to use the extracted model analysis to compute convergence statistics within R. The `WMCapacity` package uses another package, `coda` (Plummer, Best, Cowles, & Vines, 2006), which contains functions to compute convergence statistics. The `coda` package offers several convergence statistics, including Geweke (1992), Brooks and Gelman (1997), Heidelberger and Welch (1983), and Raftery and Lewis (1995) diagnostics. The specifics of these convergence statistics are beyond the scope of this appendix; for details, see the manual for the `coda` package. Below, we demonstrate how to compute the Geweke diagnostic statistic, using the extracted model information:

```
# Extract model
winningModel = womExtractModel(’With
prch effect’)
# Save chains to another variable, for
convenience
# chains =
winningModel$results$effectChains
```

The code above saves the effect `chains` to the variable chains. WoMMBAT saves chains in a format that is suitable for passing to the MCMC diagnostic functions in `coda`:

```
# Compute the Geweke (1992) statistic for
# each parameter’s MCMC chain
geweke.diag(chains)
```

Table 4 Description of some of the important elements in an extracted WoMMBAT analysis

Element	Description
$modelName	The name of the extracted model
$model	Information used by WoMMBAT to build the analysis
$model$newDat2Cat	Simplified data set used for analysis; categorical part
$model$newDat2Cont	Simplified data set used for analysis; continuous part
$model$namedDat2	Simplified data set used for analysis; factor names
$model$allEffects	Matrix showing all the effects, and how the model was built from these effects
$priors	Prior settings
$settings	MCMC settings used for the analysis
$results	All results, including MCMC chains
$results$effectChains	MCMC chains for all effects, excluding covariance matrix and slope means
$results$covChains	MCMC chains for all covariance matrices
$results$corChains	MCMC chains for all covariance matrices, converted to correlations
$results$meanChains	MCMC chains for all slope means
$results$pointEst	Point estimates (posterior means) for all effects excluding covariances and slope means
$results$DIC	Deviance information criterion

The logic behind Geweke's diagnostic is to assume that the chain has converged. The posterior mean based on an initial segment of the chain is compared with the posterior mean based on the end of the chain. How big the segments are depends on the function settings; the defaults are 10% and 50%, respectively. If the chain has converged, these estimates should be similar. Geweke's statistic is a z score and has a standard normal distribution assuming the chain has converged. Thus, extreme z scores ($|z| > 2.5$ or so) indicate possible convergence problems. It is likely that diagnostic statistics for the demonstration analysis contains some parameters that did not converge; as was mentioned previously, the demonstration analyses' 1,000 iterations is not particularly long, and convergence will probably require a longer MCMC run.

It is useful to compare the statistic yielded by `geweke.diag()` with the diagnostic plots provided by WoMMBAT to see whether the reasons for inadequate convergence can be ascertained. In addition, plotting the Geweke diagnostic statistics against the theoretical quantiles of the standard normal distribution (function `qqnorm()` in R) will aid in assessing the convergence of all parameters simultaneously.

Appendix 3

Adjusting hybrid Monte Carlo parameters for efficient parameter estimation

A critical part of using WoMMBAT is tweaking the hybrid Monte Carlo parameters in the analysis tab to improve the quality of MCMC chains (Fig. 8C). Efficient chains lead to faster, more reliable parameter estimates. Understanding the Hybrid Monte Carlo algorithm helps when tweaking the parameters. A technical explanation of hybrid Monte Carlo may be found in Liu (2001). A less formal explanation is found in the supplement to R. D. Morey (2011). Here, we offer a brief, informal explanation, along with a step-by-step to guide to improving the quality of hybrid Monte Carlo chains.

The goal of all MCMC methods is to sample from a statistical distribution—in the case of WoMMBAT, to sample from the joint posterior distribution of all parameters. The joint posterior distribution of all parameters in R. D. Morey's (2011) model is not a familiar distribution, like a multivariate normal; it is more complicated and difficult to sample from than other, familiar distributions. We must therefore enlist methods like MCMC to approximate samples from the joint posterior distribution.

All MCMC methods, including the hybrid Monte Carlo algorithm used by WoMMBAT, work in a similar way. Assume that we have a sample from the joint posterior distribution; it could be our starting value or an MCMC iteration. MCMC is a set of rules that enable us to obtain a candidate sample, comparing the candidate sample with the current sample and deciding whether to keep the candidate sample or the old sample. If we keep the candidate sample, it becomes our current sample. The algorithm is specified in such a way that if we follow the same steps over and over, our samples will approximate the distribution from which we want to sample.

In order to understand how hybrid Monte Carlo obtains candidate samples from current samples, imagine a curved surface, on which a ball has been placed. The ball will naturally roll into pits in the surface, due to gravity. This surface is an analog to a statistical distribution; high-likelihood areas in the parameter space correspond to low areas on the surface. "Sampling" from the statistical distribution is done by pushing the ball at random, waiting a certain amount of time, and then determining where the ball is after the time is elapsed. If we do this repeatedly, the list of ball locations will be samples from our statistical distribution.

We cannot, however, simulate this exactly on a digital computer, because digital computers cannot divide time and space to arbitrary precision. Instead, we must do a rough simulation. Hybrid Monte Carlo is the rough simulation. In hybrid Monte Carlo, there are two parameters: ε and N_{lf} (epsilon and number of "leapfrog" steps). The ε parameter refers to how finely we approximate time, and N_{lf} represents the number of time simulation steps we perform. WoMMBAT actually has two ε parameters: a lower ε and an upper ε. For each sample, a different ε is sampled uniformly, to improve the quality of the chain (Mackenzie, 1989). A very high quality, but extremely slow, simulation will result from setting a small ε and a large number of leapfrog steps. Our goal is to find smaller values that lead to good quality chains, without taking an inordinate amount of time. If the default WoMMBAT values fail to work for your model, we recommend following these steps, based on our own experimentation:

1. First, start with a simple model. Choose initial values of ε and N_{lf} from a previous, similar data set. If you do not have a previous analysis, use a lower ε of about 0.01, an upper ε of about 0.015, and an N_{lf} value of about 10. Sample from 500 to 1,000 MCMC iterations and compare the chain with the chains in Fig. 10, left column.
2. If the chain looks like Fig. 10c and your acceptance rate is very high, try doubling N_{lf}. If this does not affect the chain or the acceptance rate, increase both ε parameters and repeat these steps.
3. If the chain looks like Fig. 10a, decrease both ε parameters and repeat the previous steps.
4. Once you obtain a chain that looks like Fig. 10g, with an acceptance rate of between 60% and 90%, you have found a good combination. Start running longer chains and building more complicated models, slightly tweaking ε and N_{lf} as needed.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723.

Alvarez, G. A., & Cavanagh, P. (2004). The capacity of visual short-term memory is set both by visual information load and by number of objects. *Psychological Science*, *15*, 106–111.

Awh, E., Barton, B., & Vogel, E. K. (2007). Visual working memory represents a fixed regardless of complexity. *Psychological Science*, *18*, 622–628.

Bays, P.M., & Husain, M. (2009). Response to comment on "Dynamic shifts of limited working memory resources in human vision." *Science*, *323*, 877. Available at http://www.sciencemag.org/cgi/content/abstract/323/5916/877d

Brooks, S., & Gelman, A. (1997). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, *7*, 434–455.

Cleveland, W. S. (1981). Lowess: A program for smoothing scatterplots by robust locally weighted regression. *American Statistician*, *35*, 54.

Cowan, N. (2001). The magic number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, *24*, 87–114.

Cowan, N., Elliott, E. M., Saults, J. S., Morey, CC., Mattox, S., Hismjatullina, A., & Conway, A. R. A. (2005). On the capacity of attention: Its estimation and its role in working memory and cognitive aptitudes. *Cognitive Psychology*, *51*, 42–100.

Cowan, N., & Rouder, J. N. (2009). Comment on "Dynamic shifts of limited working memory resources in human vision." *Science*, *323*, 877. Available at http://www.sciencemag.org/cgi/content/abstract/323/5916/877c

Fougnie, D., & Marois, R. (2009). Dual-task interference in visual working memory: A limitation in storage capacity but not in encoding or retrieval. *Attention, Perception, & Psychophysics*, *71*, 1831–1841.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis* (2nd ed.). London: Chapman and Hall.

Gelman, A., Goegebeur, Y., Tuerlinckx, F., & van Mechelen, I. (2000). Diagnostic checks for discrete data regression models using posterior predictive simulations. *Journal of the Royal Statistical SocietyC*, *49*, 247–268. Available at http://www.jstor.org/stable/2680852

Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In J. M. Bernardo, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.), *Proceedings of the Fourth Valencia International Meeting on Bayesian Statistics*(pp. 169–194). Oxford: Oxford University Press, Clarendon Press.

Gold, J. M., Fuller, R. L., Robinson, B. M., McMahon, R. P., Braun, E. L., & Luck, S. J. (2006). Intact attentional control of working memory encoding in schizophrenia. *Journal of Abnormal Psychology*, *115*, 658–673.

Heidelberger, P., & Welch, P. (1983). Simulation run length control in the presence of an initial transient. *Operations Research*, *31*, 1109–1144. Available at http://www.jstor.org/stable/170841

Hocking, R. R. (1976). The analysis and selection of variables in linear regression. *Biometrics*, *32*, 1–49. Available at http://www.jstor.org/stable/2529336

Kumar, A., & Jiang, Y. (2005). Visual short-term memory for sequential arrays. *Memory & Cognition*, *33*, 488–498.

Liu, J. S. (2001). Monte Carlo strategies in scientific computing. New York: Springer.

Logie, R. H. (1995). *Visuo-spatial working memory*. Hillsdale, NJ: Psychology Press.

Luck, S. J., & Vogel, E. K. (1997). The capacity of visual working memory for features and conjunctions. *Nature*, *390*, 279–281.

Mackenzie, P. B. (1989). An improved hybrid Monte Carlo method. *Physics Letters B*, *226*, 369–371.

McCullagh, P., & Nelder, J. A. (1989). Generalized linear models (2nd ed.). London: Chapman & Hall.

Morey, C. C., Cowan, N., Morey, R. D., & Rouder, J. N. (2011). Flexible attention allocation to visual and auditory working memory tasks: Manipulating reward induces a trade-off. *Attention, Perception, & Psychophysics*, *73*, 458–472.

Morey, R. D. (2011). A hierarchical Bayesian model for the measurement of working memory capacity. *Journal of Mathematical Psychology*, *55*, 8–24.

Morey, R. D., Rouder, J. N., & Speckman, P. L. (2009). A truncated-probit item response model for estimating psychophysical thresholds. *Psychometrika*, *74*, 603–618.

Ntzoufras, I. (2009). *Bayesian modeling using WinBUGS*. Hoboken, NJ: Wiley.

Olsson, H., & Poom, L. (2005). Visual memory needs categories. *Proceedings of the National Academy of Sciences*, *102*, 8776–8780.

Paivio, A. (1990). *Mental representations: A dual coding approach*. New York: Oxford University Press.

Parra, M. A., Della Sala, S., Logie, R. H., & Abrahams, S. (2009). Selective impairment in visual short-term memory binding. *Cognitive Neuropsychology*, *26*, 583–605.

Pashler, H. (1988). Familiarity and visual change detection. *Perception & Psychophysics*, *44*, 369–378.

Phillips, W. A. (1974). On the distinction between sensory storage and short-term visual memory. *Perception & Psychophysics*, *16*, 283–290.

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). CCODA: Convergence diagnosis and output analysis for MCMC. *R News*, *6*, 7–11.

R Development Core Team. (2009). *R: A language and environment for statistical computing* [Computer software manual]. Vienna, Austria: R Foundation for Statistical Computing. Available at http://www.R-project.org

Raftery, A., & Lewis, S. (1995). The number of iterations, convergence diagnostics and generic metropolis algorithms. In W. Gilks, D. Spiegelhalter, & S. Richardson (Eds.), *Practical Markov chain Monte Carlo*(pp. 116–130).London: Chapman and Hall.

Repovs, G., & Baddeley, A. (2006). The multi-component model of working memory: explorations in experimental cognitive psychology. *Neuroscience*, *139*, 5–21.

Rouder, JN. & Lu, J. 2005. An introduction to Bayesian hierarchical models with an application in the theory of signal detection. Psychonomic Bulletin and Review, 12, 573–604.

Rouder, J. N., Morey, R. D., Cowan, N., Zwilling, C. E., Morey, C. C., & Pratte, M. S. (2008). An assessment of fixed-capacity models of visual working memory. *Proceedings of the National Academy of Sciences*, *105*, 5976–5979.

Rouder, J. N., Morey, R. D., Morey, C. C., & Cowan, N. (2011). How to measure working-memory capacity in the change-detection paradigm. *Psychonomic Bulletin & Review.*

Rouder, J. N., Morey, R. D., Speckman, P. L., & Pratte, M. S. (2007). Detecting chance: A solution to the null sensitivity problem in subliminal priming. *Psychonomic Bulletin & Review*, *14*, 597–605.

Rubin, D. C., & Kontis, T. C. (1983). A schema for common cents. *Memory & Cognition*, *11*, 335–341.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2002). Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical SocietyB*, *64*, 583–639.

Treisman, A., & Zhang, W. (2006). Location and binding in visual working memory. *Memory & Cognition*, *34*, 1704–1719.

Vogel, E. K., McCollough, A. W., & Machizawa, M. G. (2005). Neural measures reveal individual differences in controlling access to working memory. *Nature*, *438*, 500–503.

Wagenmakers, E.-J., Lodewyckx, T., Kuriyal, H., & Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage–Dickey method. *Cognitive Psychology*, *60*, 158–189.

Wheeler, M. E., & Treisman, A. M. (2002). Binding in short-term visual memory. *Journal of Experimental Psychology: General*, *131*, 48–64.

Wilken, P., & Ma, W. J. (2004). A detection theory account of change detection. *Journal of Vision*, *4*, 1120–1135.

Woodman, G. F., & Vogel, E. K. (2005). Fractionating working memory: Consolidation and maintenance are independent processes. *Psychological Science*, *16*, 106–113.

Xu, Y., & Chun, M. M. (2006). Dissociable neural mechanisms supporting visual short-term memory for objects. *Nature*, *440*, 91–95.