Journal of Mathematical Psychology 84 (2018) 20–48

Contents lists available at ScienceDirect

Journal of Mathematical Psychology

journal homepage: www.elsevier.com/locate/jmp

A tutorial on joint models of neural and behavioral measures of cognition[☆]

James J. Palestro[a], Giwon Bahg[a], Per B. Sederberg[c], Zhong-Lin Lu[a], Mark Steyvers[b], Brandon M. Turner[a,*]

[a] Department of Psychology, The Ohio State University, United States
[b] Department of Cognitive Science, University of California, Irvine, United States
[c] Department of Psychology, University of Virginia, United States

HIGHLIGHTS

- A tutorial on jointly modeling neural and behavioral measures is presented.
- Simulated data from Directed and Hierarchical models are used.
- A real-world example is used containing data from a perceptual discrimination task.
- User-friendly JAGS code is provided in all examples.

ARTICLE INFO

Article history:
Received 21 June 2017
Received in revised form 7 March 2018
Available online 12 April 2018

Keywords:
Model-based cognitive neuroscience
Joint models
Neural and behavioral measures
Bayesian modeling

ABSTRACT

A growing synergy between the fields of cognitive neuroscience and mathematical psychology has sparked the development of several unique statistical approaches exploiting the benefits of both disciplines (Turner, Forstmann et al., 2017). One approach in particular, called joint modeling, attempts to model the covariation between the parameters of "submodels" intended to capture important patterns in each stream of data. Joint models present an interesting opportunity to transcend conventional levels of analyses (e.g., Marr's hierarchy; Marr, 1982) by providing fully integrative models (Love, 2015). In this manuscript, we provide a tutorial of two flavors of joint models — the Directed and Covariance approaches. Computational procedures have been developed to apply these approaches to a number of cognitive tasks, yet neither have been made accessible to a wider audience. Here, we provide a step-by-step walkthrough on how to develop submodels of each stream of data, as well as how to link the important model parameters to form one cohesive model. For convenience, we provide code that uses the Just Another Gibbs Sampler (Plummer, 2003) software to perform estimation of the model parameters. We close with a demonstration of the approach applied to actual data from a contrast discrimination task where activation parameters of early visual areas are directly mapped to the drift rate parameter in a simplified version of the diffusion decision model (Ratcliff, 1978).

1. Introduction

The evolution of technology for measuring brain signals, such as electroencephalography (EEG) and functional magnetic resonance imaging (fMRI), has provided exciting new opportunities for studying mental processes. Today, scientists interested in studying cognition are faced with many options for relating experimentally-derived variables to the dynamics underlying a cognitive process of interest. While conceptually the presence of these new "modalities" of cognitive measures could have immediately spawned an interesting new integrative discipline, the emergence of such a field has been slow relative to the rapid advancements made in these new technologies. Until a little over a decade ago, much of our understanding of cognition had been advanced by two dominant but virtually non-interacting groups. The largest group, cognitive neuroscientists, relies on statistical models to understand patterns of neural activity brought forth by the new technologies. The models used by cognitive neuroscientists are typically data-mining

[☆] This research was supported by National Science Foundation grant SMA-1533500 and Air Force Research Lab contract FA8650-16-1-6770. GitHub repository hosts all of the code used in this tutorial (https://github.com/MbCN-lab/joint-modeling-tutorial) as well as a repository on the Open Science Framework (https://osf.io/qh7xr/?view_only=aafea8d894e74ee38ec67b7cc3b55780).
[*] Corresponding author.
E-mail address: turner.826@gmail.com (B.M. Turner).

https://doi.org/10.1016/j.jmp.2018.03.003
0022-2496/© 2018 Elsevier Inc. All rights reserved.

techniques, and these models often disregard the computational mechanisms that might detail a cognitive process. The other group, mathematical psychologists, is strongly motivated by *theoretical* accounts of cognitive processes, and instantiates these theories by developing formal mathematical models of cognition. The models often assume a system of computations and equations intended to characterize the processes assumed to take place in the brain. As a formal test of their theory, mathematical psychologists usually rely on their model's ability to fit and predict behavioral data relative to the model's complexity.

Although both groups are concerned with explaining behavior, cognitive neuroscientists and mathematical psychologists tend to approach the challenge from different vantage points. To appreciate the distinction between the fields, we can use Marr's (1982) levels of analysis, where our understanding of the mind can be advanced by considering a computational, algorithmic, and implementational level. At the computational level, our goal is to understand what a system does, and more importantly, why the system does what it does. At the algorithmic level, our goal is to understand exactly how a system does what it does, specifically what types of representations are used to perform the task. At the implementational level, our goal is to understand how the system can be physically realized, or how the representations in the algorithmic level could be created given biological constraints. Mathematical psychologists tend to focus on the computational and algorithmic levels, whereas cognitive neuroscientists tend to focus on the implementation level. Although progress can be made by maintaining a tight focus on one level, many important opportunities are lost (Love, 2015). For example, without an overarching theory explaining how the mind generally solves problems, such as a theory that might be developed at the computational level, it can be difficult to aggregate neuroscientific results from various experimental paradigms that focus on the implementational or algorithmic levels (cf. Coltheart, 2006).

As a remedy, new work has endeavored to integrate the levels of analysis in an effort to relate mechanisms assumed by mathematical models to the neural computations supporting task-specific behavior within the brain. However, integrating the two fields is made difficult by the fact that mechanisms in mathematical models are often necessarily abstract, whereas neurophysiological measures are physical realizations of cognitive processes (Turner, 2015). The importance of solving the integration problem has created several entirely new statistical modeling approaches developed through collaborations between mathematical psychologists and cognitive neuroscientists, collectively forming a new field often referred to as "model-based cognitive neuroscience" (e.g., Boehm, Van Maanen, Forstmann, & Van Rijn, 2014; Daw & Doya, 2006; Daw, Niv, & Dayan, 2005; Forstmann & Wagenmakers, 2014; Forstmann, Wagenmakers, Eichele, Brown, & Serences, 2011; Frank, Seeberger, & O'Reilly, 2004; Love, 2015; Mack, Preston, & Love, 2013; Palmeri, Schall, & Logan, 2015; Turner, Forstmann et al., 2013; Turner, Van Maanen, & Forstmann, 2015; van Maanen et al., 2011).

At this point, there are several approaches for integrating neural and behavioral measures via cognitive models, and these approaches are neither restricted to any particular kind of neural or behavioral measure, nor to any particular cognitive model (see de Hollander, Forstmann, & Brown, 2016; Turner, Forstmann, Love, Palmeri, & Van Maanen, 2017 for reviews). A convenient taxonomy for organizing these approaches can be built from considering a researcher's goals in relating the measures to one another (Turner, Forstmann et al., 2017). One goal might be to use the neural data to constrain a behavioral model. Another goal might be to identify patterns of neural data that are consistent with specific computations carried out in the behavioral model. The final goal, which is the focus of the current article, is to enforce statistically reciprocal relationships between the neural measures and the parameters of a behavioral model by modeling these random variables simultaneously (see Forstmann et al., 2011 for some motivation).

One successful method of performing simultaneous modeling has been the "joint modeling" approach (Cassey, Gaut, Steyvers, & Brown, 2016; Turner, 2015; Turner, Forstmann, et al., 2013; Turner, Rodriguez, Norcia, Steyvers, & McClure, 2016; Turner et al., 2015; Turner, Wang, & Merkel, 2017). Joint models were developed as an alternative to the "two-stage" correlation approaches, where parameters of a fitted cognitive model were simply correlated with a neural measure of interest. While a two-stage correlation approach does give insight into how parameters of a cognitive model are related to brain data, this approach misses an opportunity to enforce a constraint on the model parameters based on the random variation in the neural data. In other words, if one treats the neural data as a covariate, the estimates of the behavioral model parameters can be better informed. This simple covariate approach gives joint models some advantages in articulating brain-behavior relationships. Specifically, joint models are better equipped to (1) handle mismatching (i.e., when the size of the neural data is different from the size of the behavioral data) and missing data, (2) perform inference on the magnitude of brain-behavior relationships (i.e., they are not subject to Type I errors as in the two-stage approach), (3) compare different brain-behavior relationships across models, and (4) make predictions about either neural or behavioral data.

At their highest level, joint models simply require an expression specifying the joint distribution of the measures N obtained by using cognitive neuroscience techniques (e.g., EEG, fMRI) to measures of behavior B (e.g., choice, response time). Given this intentionally vague definition, there are many "classes" of joint models that vary in the way N is structurally related to B. For the purposes of this article, we narrow our focus to three types of joint models: Integrative, Directed, and Covariance. As many of our research efforts have modeled the covariation between N and B via the Covariance approach, we may have given the impression that joint models are inherently structured in a specific way, but this is not the case. Here, we present a more comprehensive account of different types of models that we collectively refer to as "joint models". Three types of joint models are illustrated in Fig. 1 via graphical diagrams, where observed variables (e.g., N and B) are shown as filled square nodes, and parameters are shown as empty circles. Paths between the nodes in the graph indicate dependency among the nodes, where an arrow pointing from one node to another indicates a "parent-to-child" ancestry (Pearl, 1988). In other words, the node being pointed at depends on the node from which the arrow originates. Although the three types of joint models can be illustrated with similar graphical diagrams, the structures introduce different constraints, which have major implications for a joint model's complexity relative to the observed data. We now discuss each of the three classes of joint models in Fig. 1.

1.1. Integrative approach

The first joint modeling approach we will focus on is the Integrative approach, where a single cognitive model is developed to predict neural and behavioral measures simultaneously. The Integrative approach is depicted on the left side of Fig. 1. Here, the neural data N and the behavioral data B are explained together through a single set of parameters θ, indicated by the connections from θ to both N and B. Alternatively, Integrative joint models can use a set of modulators to transform an internal state of a model into a prediction about the precise functional form of the neural measures. For example, different modulators would be necessary to make predictions for a blood oxygenated level dependent (BOLD) response in an fMRI study versus predictions for

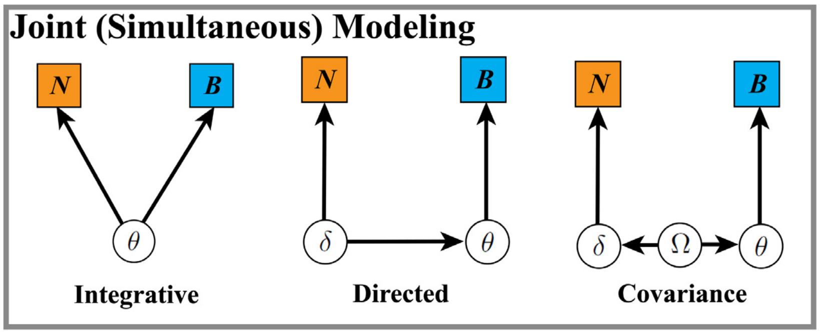

Fig. 1. An illustration of the three joint modeling approaches for linking neural and behavioral data. N represents the neural data, B represents the behavioral data, and θ, δ, and Ω represent model parameters.

an event-related potential (ERP) in an EEG study, simply because the distributions of these neural measures are quite different.

The biggest strength of the Integrative approach is that it requires strong commitments to both what underlying cognitive processes are involved and where these processes arise in the brain. By requiring these types of commitments, a researcher who wishes to create a cognitive model using the Integrative approach must make clear and explicit assumptions about the cognitive processes of interest. However, requiring these commitments is also a weakness of the approach as it can be incredibly difficult to develop a model using the Integrative approach.

In addition to aforementioned theoretical issues, several technical hurdles often arise when modeling random variables with different temporal properties. For example, neural measures (e.g., BOLD activation) are typically measured on a moment-by-moment basis over the length of a trial. Behavioral data (e.g., reaction times), on the other hand, are typically measured at the end of a trial. Thus, to instantiate a cognitive theory within the Integrative framework, one needs a moment-by-moment prediction of the neural data and a trial-by-trial prediction of the behavioral data, usually assumed to arise due to a series of latent processes. Given this unique structure, sophisticated techniques such as Hidden Markov Models (Anderson, 2012; Anderson, Betts, Ferris, & Fincham, 2010), or Bayesian change point analyses (Mohammad-Djafari & Féron, 2006) are often required to properly fit these models to data, which can be difficult to implement and computationally intensive. Thus, the Integrative approach, while strong statistically, lacks approachability.

1.2. Directed approach

The middle panel of Fig. 1 illustrates the second type of joint model we discuss in this article, an approach we refer to as "Directed" (e.g., Cavanagh et al., 2011; Frank et al., 2015; Nunez, Srinivasan, & Vandekerckhove, 2015; Nunez, Vandekerckhove, & Srinivasan, 2017). Whereas the Integrative approach relies on a set of parameters θ to describe how both the neural and behavioral data come about, the Directed approach uses a set of parameters δ to describe the functional properties of the neural data N through some statistical model and also modulate the behavioral model parameters θ through a linking function $\mathcal{M}$, such that

$$\theta = \mathcal{M}(\delta). \quad (1)$$

Usually, this linking function $\mathcal{M}$ will consist of a set of variables that allow for flexibility in the mapping from δ to θ. This is especially beneficial when the behavioral and neural data are on different scales, allowing the Directed approach to escape the technical hurdles that plague the Integrative approach.

The distinction between the Integrative and Directed approaches is a subtle one. The differences lie in the way the model parameters are used to describe the variables N and B. In the Integrative approach, a single set of model parameters jointly explain both manifest variables, whereas in the Directed approach, the parameters describing one set of variables (e.g., δ) are used to modulate another set of variables (e.g., θ). In other words, when the connection between the two streams of data is made at a single parent node, the approach is Integrative. If the connection between the two streams is made from one set of parameters to another (e.g., from δ to θ), the approach is Directed.

While Fig. 1 illustrates how the parameters δ modulate the parameters θ, other models assume the reverse influence, where the behavioral parameters θ inform the neural parameters δ. As a concrete example, van Ravenzwaaij, Provost, and Brown (2017) used a Directed joint model to account for data from a mental rotation task. To accomplish this, they used the Linear Ballistic Accumulator (LBA) model (Brown & Heathcote, 2008) to describe the behavioral data, where the drift rate parameter (i.e., corresponding to θ) – combined with some modulating parameters – was used to describe the mean of an EEG signal (i.e., corresponding to δ in Fig. 1).

1.3. Covariance approach

The final joint modeling approach we will discuss is the Covariance approach (Turner, 2015; Turner, Forstmann et al., 2013; Turner et al., 2015, 2016; Turner, Wang et al., 2017), which is illustrated in the right panel of Fig. 1. The Covariance approach is conceptually similar to the Directed approach as they both describe the joint distribution of the behavioral model parameters θ and the neural model parameters δ through a statistical constraint. However, the two approaches diverge in how they link the parameters θ and δ. In the Directed approach, θ and δ are related through an equality statement, meaning that one set of parameters is just a transformation of another set of model parameters. However, the Covariance approach assumes that θ and δ are related through a probability distribution.

To facilitate the linking between the model parameters, the Covariance approach assumes an overarching distribution governed by parameters Ω, which is used to describe the patterns present in the joint distribution of (θ, δ) across the levels to which they are applied. The connection enforced by the overarching distribution Ω is concrete: one must make a specific assumption about the relationship between θ and δ when considering the underlying cognitive processes involved. In other words, when specifying a Covariance joint model, one must explicitly specify how θ and δ are related through the linking function $\mathcal{M}$ with parameters Ω:

$$(\theta, \delta) \sim \mathcal{M}(\Omega). \quad (2)$$

Here, note that Eq. (2) expresses the joint distribution of θ and δ through a probability distribution, and so neither θ nor δ appear on the right side of the equation, in contrast to Eq. (1). While we will discuss the linking function in more detail later, one example of a linking function $\mathcal{M}$ that could be used to connect neural and behavioral data is the multivariate normal distribution. When assuming the linking function is multivariate normal, Ω consists of the hyper mean vector and the hyper variance–covariance matrix. In an analogous way to the Directed approach above, the Covariance approach would also allow the information contained in the neural data N to automatically inform the behavioral model parameters and vice versa.

One considerable advantage the Covariance approach maintains over the Directed approach is in how it treats the parameters θ and δ. Whereas the Directed approach assumes that θ are either a transformation of the neural parameters δ or some aspect of the neural data N, the Covariance approach assumes that these parameters are instead latent (i.e., not directly observable). This is especially advantageous when dealing with potential problems like outliers or missing observations (Turner et al., 2016). In the Directed approach, if we assume that we are using the neural parameters δ to describe both how the neural data N come about and how θ are specified, then any outliers that are present in N may lead to an unreasonable mapping of δ to θ. However, in a Covariance approach, if outliers are observed in the neural data N, the largest impact will be in the variability terms in the overarching distribution Ω, and the effect on the predictions about the behavioral data B will be lessened across the rest of the behavioral data.

While the Covariance approach has certain advantages over the Integrative and Directed approach, it is not without its disadvantages. One of the most prominent disadvantages is that the use of a probability distribution makes the model complex, and as a result, it often requires computationally intensive methods to sample from the desired posterior distributions of the model parameters. This complexity, while surmountable, causes the Covariance approach to be less approachable than other modeling approaches. Additionally, the complexity also limits the influence the data can have on the joint posterior distributions. As models based on a Covariance approach often feature multiple levels and numerous parameters, it requires a large amount of data for trial-level effects to be noticeable. Thus, there is a strong tradeoff between model complexity and model flexibility that accompanies the use of a Covariance joint model.

1.4. Plan of the tutorial

Having discussed the various joint modeling approaches at a high level, the rest of the tutorial focuses on specific implementations of two approaches. We chose against providing a tutorial on the Integrative approach as its most accessible implementation can be viewed simply as a Directed approach. The more complex forms of Integrative models require enough additional theoretical overhead that they are outside the scope of this tutorial (but see Borst & Anderson, 2017 for a tutorial using ACT-R). First, we present a Directed joint model in Section 2 where parameters describing the neural data directly affect a simple computational model's predictions about behavioral data. Here, we use a simple working example to make the application accessible. In addition, we provide code and a step-by-step walkthrough using JAGS (Plummer, 2003) software to carry out the parameter estimation. The operation of each line in the code is briefly described, and where possible, the code is related to the equations describing the model details. Second, we present a Covariance joint model in Section 3. Here, we build on the same example used in the Directed joint model section (i.e., Section 2) so that the reader can ascertain the differences between these approaches via the implementation. Finally, we provide a more realistic example using experimental data relating measures obtained in an fMRI experiment to parameters of a simplified diffusion decision model (DDM; Ratcliff, 1978). While the details of both the neural and behavioral submodels are more complex than the simple working examples provided in the Directed and Covariance joint model sections, the example is more realistic – it comes from a research study in our own laboratory – with the hope that readers can connect the example to their own research. We close with a discussion about limitations of our approach, as well as some theoretical considerations.

2. A directed joint model

As previously discussed, there are several ways to express the covariation between the neural and behavioral models, all of which fall under the umbrella class labeled "joint models". In this section, we will provide a walkthrough of how to apply the Directed approach to hypothetical data from a recognition memory experiment. In this section of the tutorial, we first describe the generative model that serves as the basis for each modeling approach and generate simulated data from the model. These simulated data are then used to fit the model, so that the accuracy of the parameter estimates can be assessed.

2.1. Generative model

The example we will focus on throughout this tutorial is a classic recognition memory experiment from the area of episodic memory. In this experiment, subjects are given a list of items (e.g., words) and are asked to commit these items to memory. In recognition memory literature, this is operationally defined as the "study phase". Following the study phase, subjects are presented with a second list of items of the same kind (e.g., words), one at a time, and their task is to determine if the presented item had been included on the list in the study phase – an "old" response – or if it is novel — a "new" response. By presenting each subject with a mixture of previously presented (i.e., old) and novel items and examining their responses, we can examine how well each subject encoded the study items into memory.

Although calculating the proportions of "old" and "new" responses for each item type allows us to measure memory performance experimentally, it provides little insight into the mental processes involved in the task such as encoding and retrieval, as these processes are latent. Additionally, the behavioral data we measure from such a task can only take on one of two values, and we observe only one response per item at test. As such, our ability to speak directly to how each item is stored in memory is limited, and we must look to other sources, such as neural data, to help guide our inferences.

In this tutorial, we hope to use hypothetical neural data to enhance a simple cognitive model of trial-by-trial item encoding. In line with episodic memory literature, we start with two basic assumptions: (1) there are some areas of the brain that are related to the formation of episodic memories, and (2) neural activations in these areas are positively related to the probability of memory formation for a studied item. Although this is a hypothetical example, some potential brain areas that have been linked to encoding are typically located in the medial temporal lobe such as the perirhinal cortex (Ranganath et al., 2004) and the hippocampus (Eldridge, Knowlton, Furmanski, Bookheimer, & Engel, 2000; Ranganath et al., 2004). As an illustrative example, greater activation of these areas might represent an increased chance of memory formation of the studied items, and it could be used to understand how "old" and "new" responses are formed at test.

2.1.1. Neural submodel

Suppose we implement our experimental design and obtain neural data in the form of a BOLD responses from each subject on every trial i and at five points in time t. Suppose further that the scanning times consist of the set $T = \{0, 1, 2, 3, 4\}$, which might represent the number of seconds after the presentation of a study item in a sequence. Letting $N_{i,t,k}$ denote the neural data at time t on trial i for the kth region of interest (ROI), $N_{i,t,k}$ might describe the degree of activation of the ROI on a specific trial at a specific time.

While there are many ways to characterize how the brain activity could evolve over time, we chose to employ a simple linear ramping function of the form

$$N_{i,t,k} = T_t \delta_{i,k}, \tag{3}$$

where $\delta_{i,k}$ is the ramping rate parameter on trial i for the kth ROI, which controls the neuronal firing rate across time. The linear ramping function, while simple, is sometimes used to characterize the ramping of activity in neuronal firing (e.g., Purcell et al., 2010; van Ravenzwaaij et al., 2017). Eq. (3) indicates that the rate of ROI activation over time depends on the value of δ. Fig. 2 illustrates how δ interacts with ROI activation for three hypothetical values: as δ increases, the ROI activation grows at a faster rate (i.e., the BOLD response grows faster per unit interval of time). In this illustrative example, δ may represent the latent neural activation of one of the brain areas mentioned above, such as the hippocampus.

We chose the linear ramping function as it is a simple way to describe how the mean predicted BOLD response changes over time. However, it is unlikely that we would actually observe a linear increase in brain activity over time in practice. Instead, we will assume when simulating hypothetical data that the observed BOLD responses $N_{i,t,k}$ are perturbed by some random observation error ϵ, such that

$$N_{i,t,k} = T_t \delta_{i,k} + \epsilon_{i,t,k}. \tag{4}$$

Further, we assume the errors $\epsilon_{i,t,k}$ are independent and identically distributed according to a normal distribution:

$$\epsilon_{i,t,k} \sim \mathcal{N}(0, \sigma),$$

where $\mathcal{N}(0, \sigma)$ denotes a normal distribution with mean zero and standard deviation σ. As these errors are assumed to arise form a normal distribution, the distribution of the neural data N is also normal in form. Hence, we can equivalently write

$$N_{i,t,k} \sim \mathcal{N}(T_t \delta_{i,k}, \sigma). \tag{5}$$

When σ is small, we do not expect significant differences between the estimated and observed BOLD responses. However, for nontrivial values of σ, we can expect the observed BOLD responses $N_{i,t,k}$ to depart from the model's predicted BOLD response of $T_t \delta_{i,k}$ substantially. For example, the left panel in Fig. 2 shows three sets of random realizations of neural activation on trial i at time t as dots along with the predicted BOLD response from the model as lines with coordinating colors. While σ is estimable (e.g., see the experimental application), to keep the model simple, we will assume $\sigma = 0.5$. This value of σ is small relative to the range of BOLD responses and as a result, the dots in Fig. 2 are closely aligned with the predicted BOLD response.

Neural likelihood. We can use Eq. (4) to simulate neural data N from our model with the model parameter δ. However, to determine the likelihood of observing a particular $N_{i,t,k}$ given $\delta_{i,k}$, we can use Eq. (5) to define the probability density function for $N_{i,t,k}$ as

$$p(N_{i,t,k}|\delta_{i,k}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{\left[N_{i,t,k} - T_t\delta_{i,k}\right]^2}{2\sigma^2}\right). \tag{6}$$

Because we are assuming that $\sigma = 0.5$ (i.e., a known quantity), we do not need to include it in our inference procedure, and so we do not include it in the statement $p(N_{i,t,k}|\delta_{i,k})$. From Eq. (6), we can derive the likelihood function $\mathcal{L}(\delta|N)$, which will tell us the likelihood that the single-trial neural parameters δ generated the data N for a given vector of δs and a matrix of neural data N. To define the likelihood function, we take the product of the densities in Eq. (6) evaluated at each data point $N_{i,t,k}$:

$$\begin{aligned} \mathcal{L}(\delta|N) &= \prod_t \prod_i \prod_k p(N_{i,t,k}|\delta_{i,k}) \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{n^* t^* k^*} \\ &\quad \times \prod_t \prod_i \prod_k \exp\left(\frac{\left[N_{i,t,k} - T_t\delta_{i,k}\right]^2}{2\sigma^2}\right), \end{aligned} \tag{7}$$

where n^*, t^*, and k^* denote the number of trials, time points, and ROIs, respectively.

2.1.2. Behavioral submodel

While several theories have been postulated to explain how individuals encode and retrieve items (Dennis & Humphreys, 2001; Osth & Dennis, 2015; Shiffrin & Steyvers, 1997), we will assume a more statistical (and less mechanistic) relationship between items and the observed responses. Here, we assume that the degree of "familiarity" for the ith test item is represented by a parameter θ_i, and that the θ parameters share a monotonic relationship with the probability of responding "old" to a given test item. By virtue of the study phase, we should expect that θ is larger for studied items than for non-studied items, but we impose no such restriction in our model, as our goal is to infer the level of familiarity for each item. Mechanistic models of the same task should provide some theoretical overhead for why familiarity increases with study (e.g., Shiffrin & Steyvers, 1997), but we avoid doing so in our application for the purposes of illustration.

To convert the item familiarities θ_i to a probability of responding "old" (i.e., the probability of remembering that the item was on the previously studied list), we assume a logistic function that maps θ_i onto $p(\text{"old"})$, such that

$$p(\text{"old"} \mid \text{Item } i) = \text{logit}^{-1}(\theta_i).$$

The logit function is convenient for transforming variables with infinite support to variables bounded by [0, 1], which puts the variable on the probability scale. The logit function is

$$\text{logit}(x) = \log\left(\frac{x}{1-x}\right),$$

and the inverse logit function is

$$\text{logit}^{-1}(x) = \log\left(\frac{1}{1+\exp(-x)}\right).$$

The next step is to connect the probability of an "old" response to the observed behavioral variable B_i. To do this, we assume that each B_i is a Bernoulli random deviate drawn with probability $p(\text{"old"} \mid \text{Item } i)$, such that

$$B_i \sim \text{Bernoulli}\left(p(\text{"old"} \mid \text{Item } i)\right). \tag{8}$$

It is important to note that this submodel is kept simplistic in nature for illustrative purposes, and therefore, it is not expected to fit data particularly well, nor does it have any explicit mechanisms built in to describe why the behavioral submodel parameters θ vary from one item to the next.

Behavioral likelihood. As with our neural submodel, we need a statement describing the relationship between the single-trial behavioral parameters θ and the behavioral data to form the likelihood function. Using Eq. (8), which describes how we can generate

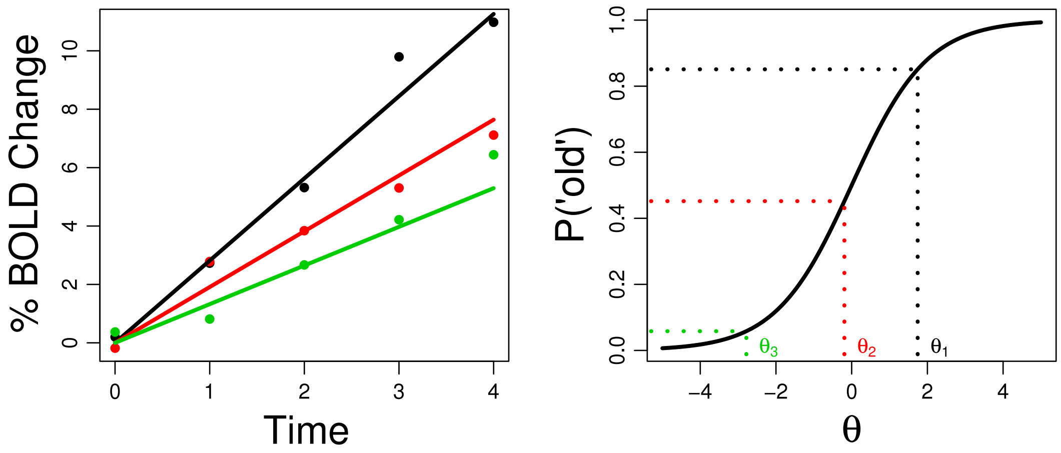

Fig. 2. Three realizations of data simulated from the model. Once θ and δ have been simulated, they can be used to generate predictions for the observed variables B and N, respectively. The left panel shows three ramping functions predicted by the model (lines) along with random draws obtained by simulating the model (dots) with three levels of δ. The right panel shows the corresponding values for θ (x-axis) that are converted into probabilities of memory formation (y-axis) according to a logistic model. As a general rule, larger ramping functions (δ) produce larger probabilities of memory formation (θ) because θ and δ are positively correlated (i.e., $\rho = 0.6$). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

behavioral data using the single-trial behavioral parameters, we can define the conditional probability distribution $p(B_i|\theta_i)$ as

$$p(B_i|\theta_i) = \left(\text{logit}^{-1}(\theta_i)\right)^{B_i}\left(1 - \text{logit}^{-1}(\theta_i)\right)^{1-B_i}.$$

From here, we invert this conditional relationship to form the likelihood function (as in the neural submodel above):

$$\begin{aligned}\mathcal{L}(\theta|B) &= \prod_i p(B_i|\theta_i)\\ &= \prod_i \left(\text{logit}^{-1}(\theta_i)\right)^{B_i}\left(1 - \text{logit}^{-1}(\theta_i)\right)^{1-B_i}. \qquad (9)\end{aligned}$$

2.1.3. Linking equations

As we discussed in the opening sections, there are many ways to express the covariation of neural submodel parameters δ and behavioral submodel parameters θ, and these ways comprise the set of models we consider to be "joint" models. Although most of our applications have expressed the relationship between the submodel parameters via a multivariate normal distribution, other more restrictive expressions naturally follow from the generic linking function specified in Turner, Forstmann et al. (2013).

As an example, maintaining that our behavioral and neural data can still be described via the submodels specified by Eqs. (7) and (9), suppose we wish to fit a joint model like the one presented in Fig. 3. Here, the hyperparameters ϕ and Σ no longer detail the statistical structure between θ and δ, but instead describe the trial-to-trial fluctuations observed only in δ. For example, we might assume

$$\delta_{i,k} \sim \mathcal{N}_p(\phi, \Sigma),$$

where p denotes the number of ROIs and the dimensionality of the multivariate normal distribution. Here, ϕ and Σ describe how the parameters on the ith trial relate to say the jth trial across all ROIs, a model that is more realistic for problems we often face in neuroscience. Note that in Fig. 3, the plate representing different ROIs is not shown to keep the graphical model simplistic.

With an expression for the neural covariates in hand, we can specify how they might be used to constrain the latent parameters θ for the behavioral data B. For example, a simple linear model is

$$\theta_i = \sum_k \delta_{i,k}\beta_k, \qquad (10)$$

where β_k are regression parameters relating each of the ROIs to the behavioral parameter θ_i. Here, θ_i is completely determined by the regression parameters β and the set of neural covariates δ, so the node corresponding to θ in Fig. 3 has a double border to express that it is not freely estimated.

2.1.4. Priors on hyperparameters

The final step in setting up a fully integrative joint model is to specify priors for the hyperparameters ϕ and Σ. For simplicity, we can specify a conjugate prior on $\Omega = (\phi, \Sigma)$, such that

$$p(\Omega) = p(\phi, \Sigma) = p(\phi)p(\Sigma).$$

Conjugacy is a term used to describe the relationship between the prior distribution and the resulting posterior distribution. If a prior can be specified such that the posterior and prior distributions have the same functional form (albeit different shapes), the selected prior is said to be conjugate to the likelihood function (Gelman, Carlin, Stern, & Rubin, 2004). Conjugacy is a desirable goal as it can make the conditional distributions of the model parameters analytically tractable, and as a result, easy to sample from in a Gibbs sampler such as the one we present in the next section. To establish conjugacy for this model (see Turner, 2015 for details), we can specify a multivariate normal prior for $p(\phi)$ and an inverse Wishart prior on $p(\Sigma)$ of the form

$$\begin{aligned}\phi &\sim \mathcal{N}_p(\phi_0, s_0), \text{ and}\\ \Sigma &\sim \mathcal{W}^{-1}(I_0, n_0), \qquad (11)\end{aligned}$$

where $\mathcal{W}^{-1}(a, b)$ denotes the inverse Wishart distribution with dispersion matrix a and degrees of freedom b.

2.2. Fitting a directed joint model to data

2.2.1. Installing JAGS

Before we can begin fitting the model to data, we must first install two software packages. The first is the JAGS software (Plummer, 2003), which can be installed by visiting http://mcmc-jags.sourceforge.net/ and downloading the version of JAGS that corresponds to the operating system installed on your computer. Once JAGS has been downloaded, follow the steps from the JAGS website to make sure that program is properly installed. When this process is complete, open R or the R interface of your choice and enter the following commands into the console:

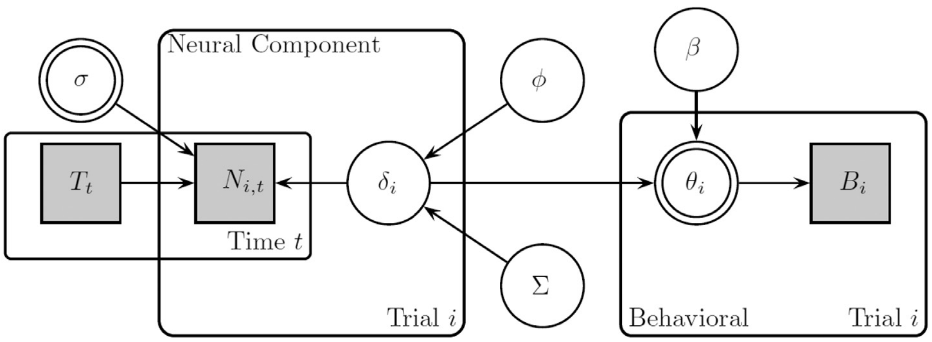

Fig. 3. Graphical diagram for a Directed joint model. Each node represents a variable in the model, where gray nodes correspond to observed variables, white nodes correspond to latent variables, and double-bordered nodes correspond to deterministic nodes (that are not estimated). Paths indicate direct relationships between the variables and plates indicate replications across dimensions (e.g., trials or time). Note that the plate corresponding to multiple neural measures is not shown for visual clarity.

```
install.packages("rjags")
require("rjags")
```

After completing these steps, the JAGS software will be available for use within R. This additional step is not essential for fitting the model with JAGS, but as the tutorial uses R to simulate data from the model (i.e., see Appendix A) and analyze the resulting posteriors, we recommend using R to interface with JAGS. For a more detailed explanation of the `rjags` package in R, please visit https://cran.r-project.org/web/packages/rjags/index.html.

2.2.2. JAGS code

Details and code describing how to generate behavioral and neural data from our recognition memory task can be found in Appendix A. Assuming we have this hypothetical data in hand, the final thing we must do before sampling from the joint posterior distribution is to specify the Directed joint model within the JAGS framework. The goal of constructing and sampling from this model in JAGS is to estimate both the neural and behavioral model parameters, which will provide us with information about the underlying mechanisms involved in completing our recognition memory task. To do so, we will specify priors on the parameters, and use JAGS to compute the posterior distribution from the hypothetical data. Once we have done this, we can sample from the joint posterior distributions to estimate the model parameters and use these estimates in our analyses.

There are two ways this process can be done: (1) you can specify the model directly in R or (2) you can create a separate text file (with a `.txt` extension) using the text editor of your choosing and call the text file in R when specifying the sampler. For this tutorial, we decided to create a separate text file for our model called “`model_directional.txt`”, and we will call this file into R using the code in Section 2.2.3. The JAGS code specifying the model is split into two parts: the first part (lines 6–17) defines the likelihoods for the neural and behavioral data, and the second part (lines 19–32) establishes priors for our model parameters.

```
 1 # JAGS code, file named ‘‘model_directional.txt"
 2 model {
 3     # convert sig to tau for convenience
 4     tau <- pow(sig, -2)
 5
 6     # loop through trials to define likelihood
 7     for (i in 1:n){
 8         for (t in 1:Nt){
 9          for(k in 1:Nroi){
10             # likelihood for neural data
11             N[i,t,k] ~ dnorm(Delta[i,k]*ts[t],tau)
     ;
12             }
13       }
14         theta[i] <- Delta[i,]         # likelihood
        for behavioral data
15         B[i] ~ dbin(1/(1+exp(-theta[i])),1);
16     }
17
18     # loop through trials to define prior on
        delta
19     for(i in 1:n){
20         Delta[i,1:Nroi] ~ dmnorm(phi,Omega);
21     }
22
23     # priors on hyperparameters
24     phi ~ dnorm(phi0,s0);
25     Omega ~ dwish(I0, n0);
26     # convert Omega to Sigma for convenience
27     Sigma <- inverse(Omega);
28     # prior on regression parameters
29     for(k in 1:Nroi){
30         beta[k] ~ dnorm(0,.001)
31     }
32 }
```

For convenience, we begin by converting the standard deviation variable `sig` into the precision variable `tau` in lines 3–4. This is not necessary, but as JAGS parameterizes the normal distribution in terms of the mean and precision (as opposed to the mean and standard deviation as in R), this transformation will become useful when using functions associated with the normal distribution (e.g., the `dnorm` function). Prior to discussing how the likelihoods for the neural and behavior data are calculated, we will first jump to lines 19–22 where the matrix `Delta`, which contains the single-trial neural parameters δ, is specified. Here, we model δ according to our hyperparameters ϕ and Σ, which have multivariate normal and inverse Wishart priors, respectively (see lines 24–28, and Eq. (11)).

With `Delta` calculated, we can use this matrix to calculate both the likelihood of the neural data on line 11 and the single-trial behavioral parameters on line 14. The single-trial behavioral parameters are then used in conjunction with the priors specified for the regression parameters on lines 29–32 to calculate the likelihood of the behavioral data (line 16). These priors are drawn from a normal distribution with mean equal to 0 and precision equal to 0.001, which are set in our list of data above.

2.2.3. R Handler code

If the JAGS software has been properly installed and loaded into R, we should be able to run the JAGS code within R using the `rjags` package. The steps we have performed up to this point have laid the groundwork for using our model to sample from the joint posterior. However, to complete the sampling procedure, we must do four things: (1) establish the model, (2) adapt the sampler, (3) update the chains, and (4) collect the generated samples. The following block of code performs these four steps:

```
 1 # specify the jags model:
 2 # locate the JAGS code, pass variables, setup
       sampler
 3 jags <- jags.model('model_directional.txt',
                      data = dat,
 5                    n.chains = 4,
 6                    n.adapt = 1000)
 7
 8 # continue adapting the sampler to optimize
       sampling efficiency
 9 adapt(jags, 1000, end.adaptation=TRUE);
10
11 # continue sampling to ensure convergence
12 update(jags, 1000)
13
14 # draw final samples, and monitor important
       variables
15 out=jags.samples(jags,
16                  c('phi', 'Sigma', 'beta'),
17                  1000)
```

Lines 2–6 specify the JAGS sampler and store the JAGS object in the variable `jags`. For our purposes, the `jags.model` function takes four arguments, which are broken up into four separate lines. Line 3 calls the text file specifying the Directed joint model that is to be used for the sampling process. Line 4 loads the list of data that we specified earlier into the sampler. Finally, the variable `n.chains` (Line 5) tells JAGS how many chains to sample with and `n.adapt` (Line 6) tells the software how many `adaption` iterations to run in the initialization stage.

Now that the sampler is defined and initialized, we can continue the adaptation stage of the sampling process to further improve sampling efficiency. This is shown in lines 8–9. Here, as we have set the argument `end.adaption` to "TRUE", it will return a TRUE/FALSE statement letting you know whether the adaptation is complete (TRUE) or not (FALSE). Once the sampler is appropriately adapted, we can finally sample from the posterior. To do this, we make use of JAGS `update` function in lines 11–12, which runs the updating process for 1000 iterations for each chain.

The last step is to extract the posterior samples from the `jags` object for use in our analyses. To do so, we use the function `jags.samples` in lines 14–17 and specify our three variables of interest – `phi`, `Sigma`, and `beta` – to store as output in the `out` variable. The function `jags.samples` draws random samples from the posterior distribution of any variable or variables of interest in our model. In the case above, we are drawing 1000 random samples from the posterior distributions of the variables `phi`, `Sigma`, and `beta` and storing them as output in the variable `out`. In other words, what we now have stored in the variable `out` are 1000 random posterior samples for each chain for each parameter. Extracting these variables makes them available for use in data analysis and plotting.

2.3. Recovery analysis

There are several things we can do with our parameter estimates in hand. However, probably the simplest of these is to assess the accuracy of the estimates by comparing them to the true values used to generate the data in R. This is known as a parameter recovery analysis, and it is shown in Fig. 4 with the regression parameters β. Each panel of Fig. 4 shows the corresponding β parameter estimate for each neural covariate. In both panels, the histograms are composed of the random posterior estimates collected by the `jags.samples` function. The red vertical line in each panel is the true value used to generate the data. The priors for β_1 and β_2 are also plotted in each panel, but as they are so diffuse relative to the posterior, they are barely visible. Near perfect recovery of the model's parameters would cause the red line and the peak of the histogram to align. However, what we find is that, while the posterior estimates and the true value do not align perfectly, the true value is encompassed in the posterior estimates. This suggests that the regression parameters have been recovered accurately.

2.4. Summary

In this section, we showed how to implement a Directed joint model using the JAGS software, as well as general recommendations on how to assess the recovery of the model parameters. If the linking function has been selected appropriately, Directed joint models are powerful in that they provide a great deal of constraint on a model in capturing behavioral data. The assumption that neural data necessarily give rise to mechanisms in a cognitive model is a strong one. For example, it is not always the case that such a clear mapping from neural to behavioral data exists, and it is certainly rare to have accurate assumptions when performing initial explorations of brain-behavior relations (Schall, 2004; Teller, 1984). Because there are often properties of the linking function that are not perfectly explained in a Directed joint model, Covariance joint models were proposed to assess the degree of association between the random variables specifying the neural and behavioral submodels. In the next section, we show how to fit such a model to data, while using the same working example shown in this section so that the technical differences between the two approaches can be appreciated.

3. A covariance joint model

This section of this tutorial focuses on the Covariance joint model, which is illustrated in the right panel of Fig. 1. As with the other approaches, there are three main components: the neural submodel, the behavioral submodel, and the linking function. We have discussed the neural and behavioral submodels at length in the previous sections, so the only component that differs from the Directed joint model above is the way in which the parameters of the two submodels are connected. In this section, we first describe the generative model and then discuss the linking function. Finally, we show how to fit the model to simulated data, and assess parameter recovery.

3.1. Generative model

Fig. 5 shows a graphical diagram of a Covariance joint model. Here, we see that Covariance joint models are not that different from Directed joint models, with the exception of the middle area in the figure. Specifically, the relationship between θ and δ are defined by parent nodes or hyperparameters ϕ and Σ, a feature that is in contrast to the Directed joint model in Section 2. Unlike the Directed joint model, the path of influence does not go from neural data to behavioral data, nor does it go from behavioral data to neural data. Instead, the path of dependence starts with the hyperparameters ϕ and Σ, then trickles down to the submodel parameters θ and δ.

In its most general form, θ and δ are connected through some linking function $\mathcal{M}$, dictated by a set of hyperparameters Ω, such that

$$(\theta, \delta) \sim \mathcal{M}(\Omega). \tag{12}$$

In the original presentation, this linking function was purposefully left generic so that one could "plug-in" a number of different linking functions to constrain the estimates of θ and δ (Schall, 2004; Teller, 1984; Turner, Forstmann et al., 2017). However, for the purposes of this tutorial, we must specify this linking function so that we can fit the model to data. Our choice of a linking function will fall in line with previous applications (Turner, Forstmann et al., 2013; Turner et al., 2015, 2016), and we will use a multivariate

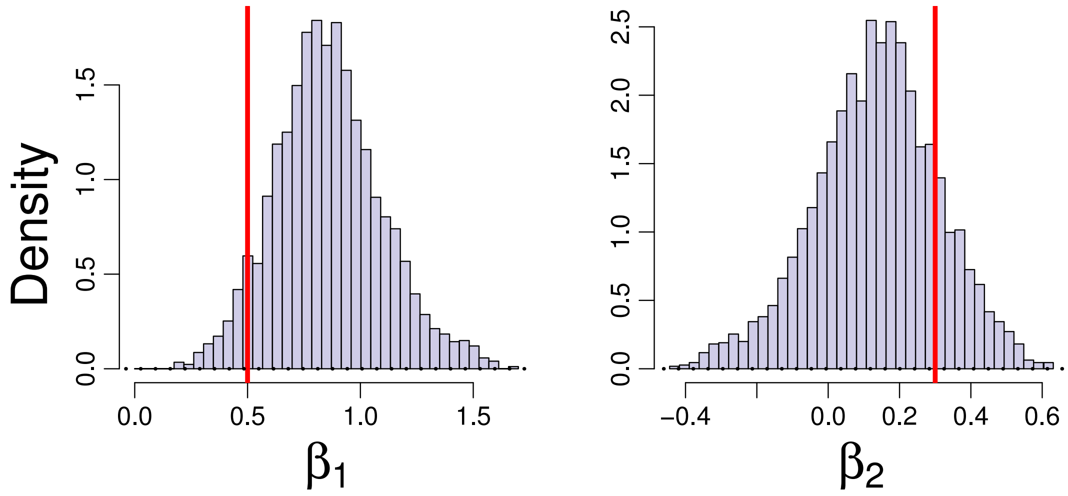

Fig. 4. Estimated regression parameters. The left and right panels show histograms of the estimated posterior distributions for β_1 and β_2, corresponding to the first and second ROI respectively. The true value of the parameter used to generate the data is shown as the vertical red line.

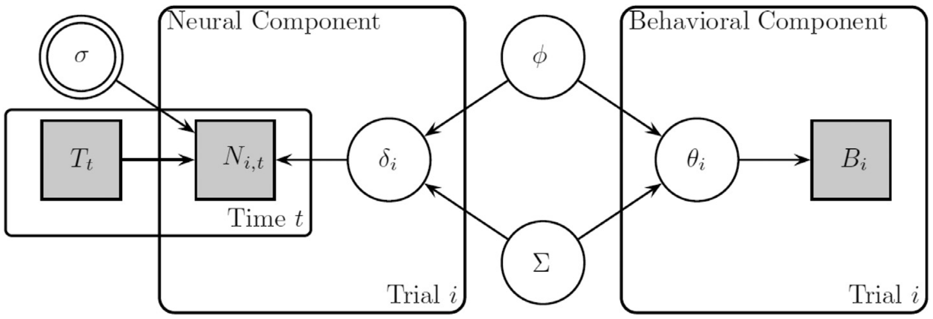

Fig. 5. Graphical diagram for the joint model presented in this section. Each node represents a variable in the model, where gray nodes correspond to observed variables, white nodes correspond to latent variables, and double-bordered nodes correspond to deterministic nodes (that are not estimated). Paths indicate direct relationships between the variables and plates indicate replications across dimensions (e.g., trials or time).

normal distribution with mean vector ϕ and variance–covariance matrix Σ , where $\Omega = \{\phi, \Sigma\}$. Concretely, this linking function takes the form

$$(\theta, \delta) \sim \mathcal{N}_p(\phi, \Sigma), \tag{13}$$

where p is the sum of the length of θ and the length of δ, in other words the dimensionality of the linking function.

We have picked the multivariate normal for several reasons. First, the mean vector ϕ conveniently characterizes the central tendency of each parameter θ and δ. Second, the variance–covariance matrix Σ expresses the degree of relatedness between each pairwise combination of θ and δ, which allows us to assess how well our model relates to brain activity across a set of ROIs. Third, as we will discuss below, with the appropriate prior selection for ϕ and Σ, we can establish a conjugate relationship between the prior and posterior, which facilitates efficient estimation of the model parameters (Turner et al., 2015).

The properties of the hyperparameters will depend on how the lower-level parameters θ and δ are used. For example, θ and δ could represent subject-specific parameters meaning that Ω would describe the distribution of the model parameters between subjects in the group. For example, Turner et al. (2016) recently used this type of linking structure to combine neural measures like EEG and fMRI to the drift rate parameter in the LBA model. By contrast, θ and δ could also represent trial-specific parameters meaning that Ω would be a set of condition- or subject-specific parameters. Turner et al. (2015) used this structure to relate trial-to-trial fluctuations in the BOLD response directly to trial-to-trial parameters of the diffusion decision model. Regardless of the characterization of the model parameters, the hyper mean vector ϕ can be divided into the set of mean parameters for the neural submodel (δ_μ) and the behavioral submodel (θ_μ), such that $\phi = \{\delta_\mu, \theta_\mu\}$. Similarly, the variance–covariance matrix Σ can be partitioned as

$$\Sigma = \left[\begin{array}{c|c} \delta_\sigma^2 & \rho\delta_\sigma\theta_\sigma \\ \hline (\rho\delta_\sigma\theta_\sigma)^T & \theta_\sigma^2 \end{array}\right], \tag{14}$$

where δ_σ is the standard deviation of the neural submodel parameters, θ_σ is the standard deviation of the behavioral submodel parameters, and ρ is the correlation between the submodel parameters. Eq. (14) consists of matrices that characterize various dispersions of the model parameters, where the element $\rho\delta_\sigma\theta_\sigma$ uses the parameter matrix ρ to model the correlation between submodel parameters. Specifying the model in this way allows us to directly infer the degree to which behavioral submodel parameters are related to which neural submodel parameters. To reduce the number of model parameters, we can also constrain elements of this partition to be equal to zero. For example, if we were uninterested in correlations that might exist from one parameter in the behavioral model to another, we could impose a constraint on θ_σ^2 to make the off-diagonal elements equal to zero. Or, if we had a specific brain-to-mechanism hypothesis we wanted to investigate, we could selectively estimate specific elements of ρ (Turner et al., 2016). Such constraints are particularly useful when the intention of one's research is confirmatory rather than exploratory (cf. Turner, Forstmann et al., 2017).

3.1.1. *Linking equations*

The multivariate linking function in Eq. (13) describes how the neural and behavioral parameters can be randomly simulated across trials in an experiment. Using this equation, we can describe the probability distribution $p(\theta_i, \delta_i)$ of a particular $z_i = (\theta_i, \delta_i)$ with the equation

$$p(\theta_i, \delta_i|\phi, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left(-\frac{1}{2}[\phi - z_i]^T \Sigma^{-1} [\phi - z_i]\right), \quad (15)$$

where $|\Sigma|$ is the determinant of Σ. Eq. (15) describes the joint distribution of θ and δ in such a way that they are both informed by their respective streams of data B and N and constrained by the hyperparameters ϕ and Σ. As such, one can surmise that Eq. (15) serves as a prior distribution of θ and δ.

With the model framework in place and our linking function appropriately specified, we can now work toward estimating the parameters of the model. To do so, we must generate samples from the joint posterior distribution of the model parameters conditional on the observed data, written

$$p(\theta, \delta, \phi, \Sigma|N, B) \propto \mathcal{L}(\theta|B)\mathcal{L}(\delta|N)p(\theta, \delta|\phi, \Sigma)p(\phi|\Sigma)p(\Sigma)$$

where each function on the right side is given by the equations listed above.

3.2. *Fitting a covariance joint model to data*

3.2.1. *JAGS code*

As with the Directed joint model, the first thing we must do is to specify our Covariance joint model in JAGS. The code will again be split into two parts: the first part will define the likelihood function, and the second part will define the priors for the parameters of the model. Again, for the purposes of the tutorial, we choose to specify the JAGS code into a separate text file called "`model_covariance.txt`" that will later be called into R.

```
 1 # JAGS code, file named ''model_covariance.txt"
 2 model {
 3     # convert sig to tau for convenience
 4     tau <- pow(sig, -2)
 5 
 6     # loop through trials to define likelihood
 7     for (i in 1:n){
 8         for (t in 1:Nt){
 9             # likelihood for neural data
10             N[i,t] ~ dnorm(DeltaTheta[i,1]*ts[t],
     tau);
11         }
12         # likelihood for behavioral data
13         B[i] ~ dbin(1/(1+exp(-DeltaTheta[i,2])),1)
     ;
14     }
15 
16     # loop through trials to define prior on (
     delta, theta)
17     for(i in 1:n){
18         DeltaTheta[i,1:2] ~ dmnorm(phi,Omega);
19     }
20 
21     # priors on hyperparameters
22     phi ~ dmnorm(phi0,s0);
23     Omega ~ dwish(I0,n0);
24     # convert Omega to Sigma for convenience
25     Sigma <- inverse(Omega);
26 }
```

The model code for the Covariance joint model is similar to the code for the Directed joint model above. The key difference in this code, however, is that the parameter matrix used in the calculation of the neural and behavioral likelihoods contains estimates for both the neural parameters δ and the behavioral parameters θ. This is shown in lines 16–19, where we define the priors on both θ and δ (i.e., the linking function) as opposed to just δ in the Directed joint model. Other than the different linking function, the structure of the code is virtually identical. Lines 6–14 calculate the likelihoods for the behavioral and neural data using Eq. (7) (line 10) and Eq. (9) (line 13). Lines 21–23 specify the priors on the hyperparameters. Finally, lines 3–4 and 24–25 conveniently convert `sig` to `tau` and `Omega` to `Sigma`, respectively.

3.2.2. *R Handler code*

The R code used to sample from the posterior with our Covariance model is similar to that used to sample with the Directed model, so we will not go into great detail. However, it is important to note that when drawing samples from the posterior and storing them as output, we must properly specify which variables are to be stored. Here, `phi` and `sigma` remain the same, but rather than storing output from the `Delta` parameter matrix as we did with the Directed model, we must specify that we want to store output from the new `DeltaTheta` matrix.

```
 1 # specify the jags model:
 2 # locate the JAGS code, pass variables, setup
     sampler
 3 jags <- jags.model('model_covariance.txt',
 4                    data = dat,
 5                    n.chains = 4,
 6                    n.adapt = 1000)
 7 
 8 # continue adapting the sampler to optimize
     sampling efficiency
 9 adapt(jags, 1000, end.adaptation=TRUE);
10 
11 # continue sampling to ensure convergence
12 update(jags, 1000)
13 
14 # draw final samples, and monitor important
     variables
15 out=jags.samples(jags,
16                  c('phi', 'Sigma', 'DeltaTheta'),
17                  1000)
```

3.3. *Recovery analysis*

To assess how accurate the model's estimates are, we can calculate the posterior means (PMs) of the model's parameters and compare these to the values used to generate the data. To do this in R, we simply take the average across both dimensions of our estimated parameter matrix DeltaTheta using the following code:

```
1 # calculate the mean of the posteriors
2 pms=apply(out$DeltaTheta,c(1,2),mean)
3 # delta is the first column, theta is the second
    column
4 delta=pms[,1]
5 theta=pms[,2]
```

Line 2 creates a new variable `pms` that stores the mean of each dimension. Lines 4–5 create the variables `delta` and `theta`, which correspond to the neural node δ and the behavioral node θ in Fig. 5, respectively. We can then use these PM estimates to assess how closely the model's estimates are to the values used to simulate data from the model.

The results of the recovery analysis are illustrated in Fig. 6. Here, the left and right panels plot the estimated model parameters on the y-axis against the true values of the model parameters on the x-axis for θ and δ, respectively. In addition, the correlation coefficient is displayed in the bottom right corner, with higher values of R suggesting a greater correspondence between the true and estimated values. Focusing on the left panel of Fig. 6, we see that model provided accurate estimates for the δ parameters. However, in right panel of Fig. 6, which focuses on the single trial behavioral parameters θ, the recovery of the model parameters

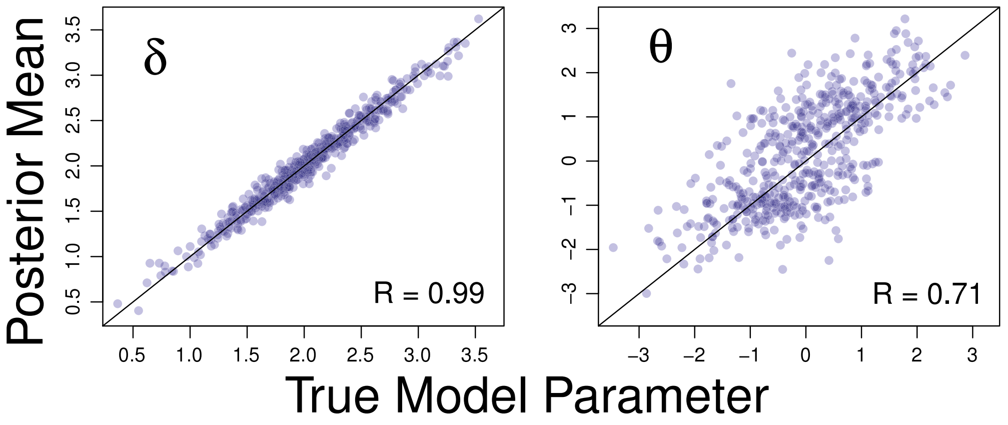

Fig. 6. Recovery of the single-trial model parameters. The posterior mean (PM) (y-axis) is plotted against the true model parameter (x-axis) for the neural model parameters δ (left panel) and the behavioral model parameters θ (right panel). Within each plot, the correlations between parameter estimates and true values of the parameters are reported.

was good albeit poorer. One potential explanation for the poorer recovery is the differences in the number of observations between the neural parameters δ and the behavioral parameters θ. Typically, a model's ability to accurately estimate the posterior of a parameter is positively related to the amount of data available per model parameter, with more data available leading to more accurate posterior estimates. As there was substantially less data available for the θ parameters as compared to the δ parameters – one data point per θ parameter vs five data (i.e., time) points per δ parameter – a poorer fit is expected.

In addition to assessing recovery at the parameter level, we can also assess recovery at the hyper level for the hyperparameters ϕ and Σ. Rather than plotting estimates against the true parameter values, we can instead visualize the recovery of the posterior distributions using violin plots. In Fig. 7, the estimated posterior distribution of each hyperparameter are illustrated using a violin plot with the corresponding true parameter value illustrated as a black "X". To assess accuracy, we can look at two things: (1) the shape of the distribution and (2) its location relative to the true parameter value (i.e., the black X). A more accurate recovery would result in narrower (i.e. less variance) posterior distributions that encompass the X, and poorer recovery would result in wider and more varied distributions and/or the X falling outside of the violin plot.

The left panel of Fig. 7 shows that both hyper mean parameters for ϕ were accurately recovered, with ϕ_1 for the neural data having substantially less variance relative to ϕ_2. The right panel of Fig. 7 shows the estimated posterior distributions of the components of the matrix Σ: the standard deviation of the neural model parameters σ_1 (left), the standard deviation of the behavioral model parameters σ_2 (middle), and the correlation between the single-trial parameters ρ (right). The violin plots suggest that all the components of Σ were accurately recovered, with the neural subcomponent σ_1 showing more accurate recovery relative to the behavioral subcomponent.

3.4. Summary

In this section, we described how to adapt the Directed joint model from the first application to make it suitable for a Covariance joint model. The main difference between these approaches is in the way the parameters of the neural and behavioral submodels are connected. In the Directed approach, one set of model parameters is a deterministic function of another set. In the Covariance approach, both sets of parameters are conditionally independent, although they are mutually constrained via the prior structure in the model's hierarchy. The difference between the two types of architectures has some interesting implications regarding model flexibility and constraint, which is a comparison we will save until the General Discussion. Here, we have shown that despite the complexity of the Covariance approach, the parameters can still be recovered accurately with JAGS. Of course, the analyses in this section were simulation-based, meaning that the true parameter estimates were known all along. In the next section, we transition to a more realistic scenario where the true data generating mechanism is not known; instead, it is inferred directly from experimental data.

4. An application to experimental data

So far, the applications in this tutorial have been simplistic and idealized as a way to introduce the concepts of joint modeling. The worked examples above begin by first simulating data from the model and using JAGS to recover the model parameters. However, in practice, fitting real behavioral and neural data with a joint model can be messy and complicated. Thus, in this section, we show how to construct and fit a joint model to real-world data from an fMRI experiment. Below, we use both Directed and Covariance joint models to examine how neural data can be related to the parameters of a simplified DDM. The structure will be similar to that of the preceding sections: we first describe the experiment and data collection procedure, then we describe the neural and behavioral submodels that comprise the Directed joint model. Finally, we provide JAGS and R handler code to fit the model and evaluate the accuracy of the estimated parameters.

4.1. Experiment

For our experimental application, we conducted a pilot study that consisted of one fMRI session with one healthy subject. The subject was asked to complete a contrast discrimination task, the structure of which is illustrated in Fig. 8. For each trial, the subject was presented with two grating contrast stimuli flickering at 2 Hz, each at different contrast levels, for 8 s (i.e., each stimulus turned on and off every 250 ms) with a mean interstimulus interval of eight seconds. In this task, the higher contrast stimulus had clearer boundaries between the white and black grating columns. After presenting the two grating stimuli, a cue was provided (i.e., a "×" symbol) to elicit a response from the subject about which of the two stimuli had the higher contrast level.

One run of the contrast discrimination task was conducted with 20 trials per run. The grating stimuli could take on one of five contrast levels ranging between 0 and 1 (0.01, 0.03, 0.1, 0.3, 1) in a 5 × 5 factorial design; however, the five stimulus pairs

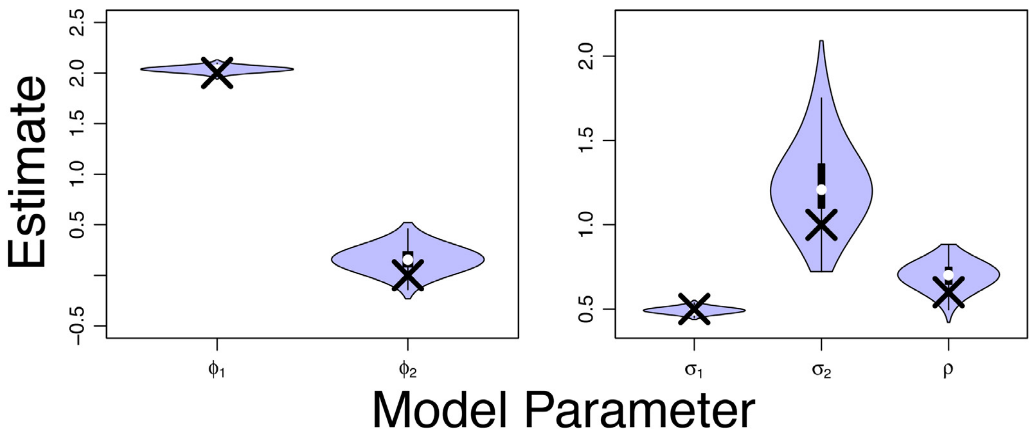

Fig. 7. Recovery of the hyperparameters in the joint model. In each panel, the estimated posterior distributions are illustrated with a violin plot and the true value of the model parameters are shown as the black "X". The estimates corresponding to the parameters ϕ are shown in the left panel, whereas the parameters corresponding to the elements within Σ are shown on the right panel.

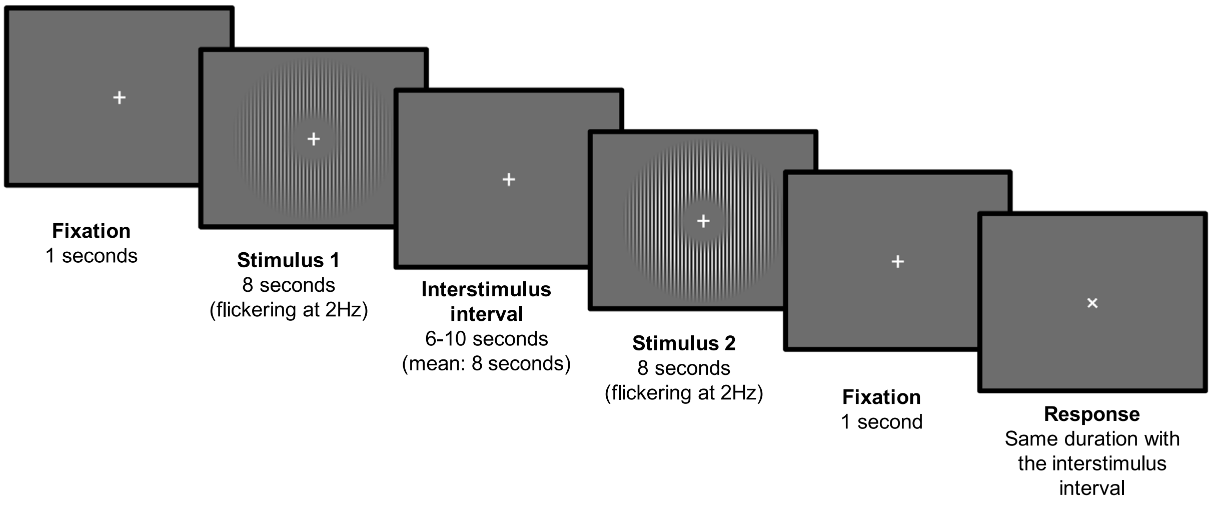

Fig. 8. Structure of the contrast discrimination task. Subjects were asked to determine which of two presented stimuli were of highest contrast.

consisting of the same contrast level were excluded from the task (e.g., 0.3 and 0.3). Each run lasted a total of 704 s, with 12 s fixation periods in the beginning and the end of each run. The fixation period was necessary for the BOLD response to return to baseline, which helps to mitigate the potential overlap in neural activity that may arise from the previous trial or other effects such as the presentation of task instructions. Functional data were recorded every two seconds.

For simplicity, we will assume that preprocessing of the functional images has already been performed. Using the anatomy-based standard ROIs in the Montreal Neurological Institute (MNI) space, we constrained the ROI to early visual area, namely V1 from both hemispheres. The "mask" was used to identify the target voxels that comprised the ROI, and the mean time series data of the voxels within this ROI during the contrast discrimination task were used as the neural measures in our data analysis. More information about how the region of interest (ROI) was defined is available in Appendix C.

4.2. *Mathematical details*

4.2.1. *Neural submodel*

Previous studies have demonstrated that the visual cortex shows greater activation as the contrast level increases (e.g., Boynton, Demb, Glover, & Heeger, 1999). With this in mind, we assumed that the activation level in the visual cortex for each grating stimulus drives the contrast discrimination process and subsequent behavioral responses in our experiment. To this end, the neural submodel should detail the neural activation in the visual cortex for each of the two presented stimuli. The goal then, is to use the activation levels from the neural submodel (described in this section) as a way to derive a decision variable in the behavioral submodel (described in the next section). As the experimental application presented here is more complicated than the hypothetical applications discussed above, we must first describe the details of the neural data to justify the choices we made about how to quantify the neural activation corresponding to each stimulus presentation.

BOLD signal and the hemodynamic response function. In fMRI experiments, we typically measure what is known as the blood-oxygenation-level dependent (BOLD) signal, which is assumed to reflect the neural activation evoked by a stimulus. This assumption is based on the idea that the oxygen level in blood is strongly affected by hemodynamic activities in the blood flow, which typically features a delayed increase to the peak activation level, followed by a temporary undershoot of the baseline level of activity. Based on characteristics of the hemodynamic activities,

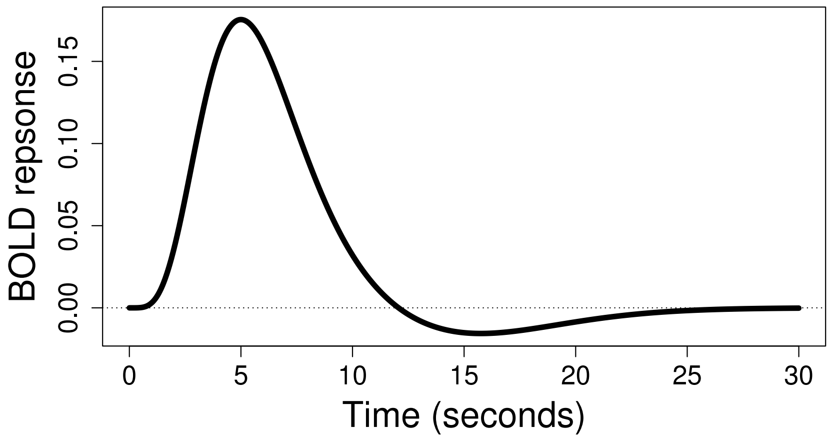

Fig. 9. A canonical form of a hemodynamic response function. The double-gamma model in Eq. (16) was used in our analysis with the following shape parameters: $a_1 = 6$, $a_2 = 16$, $b_1 = 1$, $b_2 = 1$, and $c = 1/6$. For illustrative purposes, we set the activation parameter $\beta = 1$.

several models of hemodynamic responses have been proposed for describing and analyzing fMRI data.[1]

One of the most common and successful models of the hemodynamic response function (HRF) is a canonical form of the double-gamma model implemented in SPM 12 (http://www.fil.ion.ucl.ac.uk/spm/software/spm12/):

$$h(t) = \beta h_0(t) = \beta \left(\frac{t^{a_1-1} {b_1}^{a_1} \exp(-b_1 t)}{\Gamma(a_1)} - c\,\frac{t^{a_2-1} {b_2}^{a_2} \exp(-b_2 t)}{\Gamma(a_2)} \right), \qquad (16)$$

where t represents time, β is the amplitude of the response, and $\Gamma(x) = (x-1)!$ is the gamma function. The shape parameters a, b, and c are conventionally assumed to have fixed values: $a_1 = 6$, $a_2 = 16$, $b_1 = 1$, $b_2 = 1$, and $c = 1/6$. Therefore, β, which scales the peakedness of the function $h_0(t)$, is the only free parameter to be estimated. Fig. 9 shows the form of the canonical HRF defined in Eq. (16). Here, all shape parameters are set to their conventional values, and the amplitude parameter β is set to one. Fig. 9 shows that the double-gamma HRF produces both the steady increase in activation as well as the "post-stimulus dip" that are typically observed in real experiments.

Linear time-invariant property and convolution. Although Fig. 9 and Eq. (16) describe the neural activation that ensues following a single stimulus presentation, in nearly all experiments, we are concerned with modeling the effects of many stimulus presentations over time. As the shape of the HRF in Fig. 9 shows, a problem occurs when stimuli are presented within 20 or 30 s from one another. Namely, the effects of a single stimulus presentation can linger for up to 30 s, and these effects can alter the observed BOLD response of subsequent stimuli from what is predicted by a canonical HRF. Given this, in realistic applications (i.e., unlike the working examples in the first two sections), we must consider the neural measures on every trial to be realizations of a long time series of events starting from the first stimulus presentation and lasting up until the current point in time.

Fortunately, the hemodynamic response itself is known to have a linear time-invariant (LTI) property than can be exploited when modeling the BOLD time series data from our experiment (Boynton, Engel, Glover, & Heeger 1996). The LTI property can be described in two pieces. First, the time-invariance portion of the LTI means that if neural activation is delayed by t seconds, then the hemodynamic response evoked by the neural activity is also delayed by the same amount of time. As experimenters, we typically control the time at which stimuli are presented, which implies that we know at what point in time we should expect to see neural activation. Hence, we can simply assume that each stimulus presentation has a corresponding HRF function, and these HRF functions begin at the time t when a given stimulus was presented.

Second, despite the hemodynamic response function being nonlinear with respect to time (see Fig. 9), the amplitude β of the hemodynamic response is known to be linearly related to the strength of neural activation in a given region. As a consequence, the amplitude parameters can be evaluated in relative terms across subjects, conditions, or even individual stimuli. Furthermore, the amplitude parameters themselves can be treated as blocking variables that correspond to the levels of an independent variable central to our experiment. For example, if one area of the brain responded to the contrast of a stimulus, we would expect greater activation in this area when higher-contrast stimuli are presented. If we were to treat the contrast level as an independent variable in our experiment, we might choose to discretize the contrast space, say on a zero to one scale, while choosing five contrast levels to present to subjects in the experiment. In this scenario, it would be sensible to assume that the estimates of the amplitude parameters could be constrained by knowing to which contrast condition a given stimulus belonged.

As an illustration, Fig. 10 shows how the LTI property can be used to model the BOLD time series data. The left column shows how one would model two stimulus presentations that are only different in the time at which they were presented, whereas the right column shows how one would model two stimulus presentations that differ in both time and neural response. The top row shows what is known as a "design" matrix, where stimulus presentations are represented as spikes at different points in time (x-axis). In this figure, the presentations of the stimuli occur at $t = 0$ and $t = 7$ s. However, the presentations of the stimuli might evoke different neural responses, depending on the properties of the stimuli such as in the hypothetical contrast example discussed above. For example, in the top left panel, two stimuli are presented that evoke the same neural response (i.e., $\beta = 1$), whereas the top right panel shows two stimuli that evoke different neural responses (i.e., $\beta = 1$ and $\beta = 2$). The bottom panel shows the HRFs corresponding to the spikes in the top row. Here, the individual HRFs are clearly separated in a way defined by the design matrix above. Furthermore, the amplitude of the HRFs is determined by the design matrix, illustrated by the heights of the spikes in the top row.

Mathematically, we can specify how the HRFs should be shifted and amplified through a process known as convolution. Using $h(t)$

[1] At this point, we direct the reader to more extensive books detailing fMRI design and analysis, such as Poldrack, Mumford, and Nichols (2011).

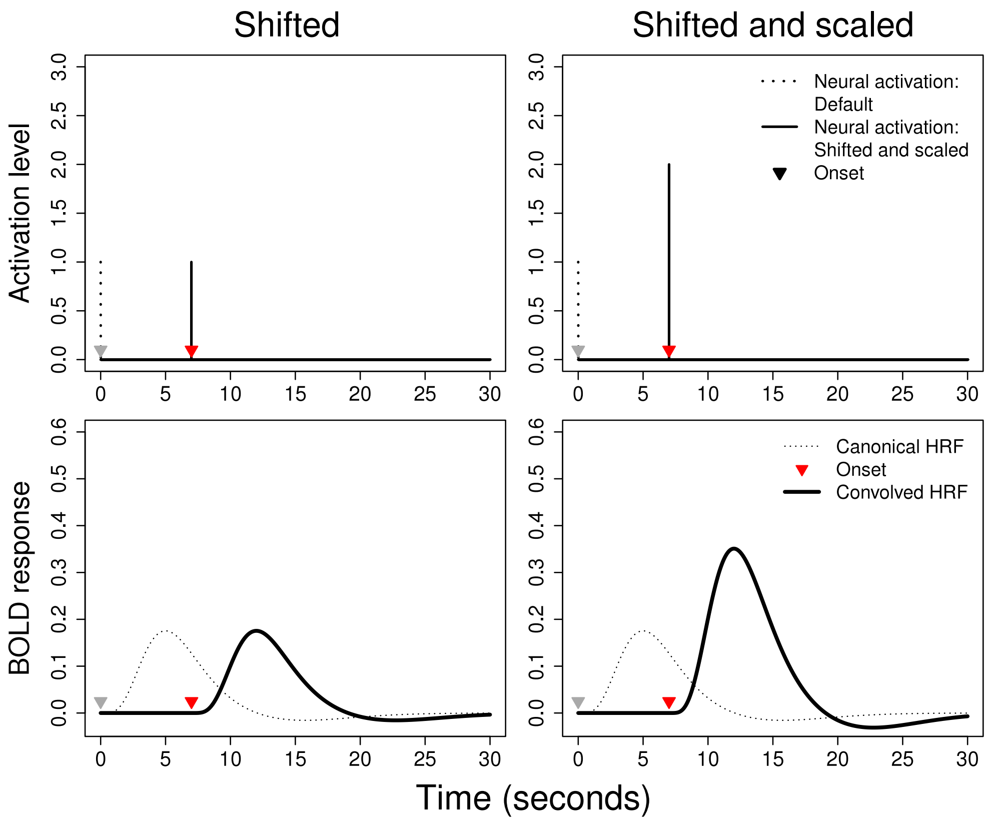

Fig. 10. Convolution of the hemodynamic function (HRF). The plots in the first row show the timing and strength of neural activation. In both columns, stimuli are presented at $t = 0$ and $t = 7$ s. In the left column the activations for both stimuli are $\beta = 1$, whereas the activations in the right column are $\beta = 1$ and $\beta = 2$, respectively. The bottom row shows the canonical form of the double-gamma HRF with the same shape parameters in Fig. 9 (a dotted line) and a convolved HRF according to the activation setting (a bold line) from the top row. In all panels, either a gray or red triangle specifies the timing of the neural activation. Compared to the canonical HRF, it is observed that convolved HRFs show temporal shift by the same amount of the activation time and amplification which is proportional to the strength of the activation in the top row.

to denote the double-gamma HRF from Eq. (16), we can also specify a boxcar function $f(t)$ that details the time at which stimuli are presented. In other words, the function $f(t)$ takes on the value of one at the values of t that a stimulus was presented, but is zero otherwise. Then, to convolve our individual HRFs with $f(t)$, we evaluate the following equation:

$$(f * h)(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau$$
$$= \int_{-\infty}^{\infty} h(\tau)f(t-\tau)d\tau \quad \text{(commutativity)}. \tag{17}$$

While Eq. (17) may seem complicated, it is at least conceptually easy to understand from Fig. 10 in that it centers and scales the double-gamma HRF from Eq. (16) at each point in time that a stimulus was presented.

While Fig. 10 makes clear our goal of formally shifting and scaling separate HRFs for each stimulus presentation, we have not yet addressed how the individual HRFs may affect one another, depending on how far apart they are separated in time. Essentially, when stimulus presentations occur close in time, the effects that one stimulus has on the obtained BOLD response may carry over into the BOLD response observed after the second stimulus has been presented. To decouple the effects underlying the obtained BOLD signal, we must have a way of integrating the individual HRFs into a single convolved HRF. One conventional way to achieve this is setting individual regressors for each trial in the design in the general linear model framework, which is sometimes called beta-series regression (Mumford, Turner, Ashby, & Poldrack, 2012; Rissman, Gazzaley, & D'Esposito, 2004) in the context of multi-voxel analysis.

Suppose in an interval of T units of time (i.e., seconds, milliseconds), we present R stimuli of various levels of the independent variable. We can let the vector $\boldsymbol{\beta}$ contain the degrees of neural activation of each of the R stimuli (i.e., β_i, where $i \in \{1, \ldots, R\}$), plus one baseline activation level parameter β_0, such that

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_R \end{bmatrix}. \tag{18}$$

These βs are the parameters governing the amplitude of the HRFs, and are to be estimated from the data. The top row of Fig. 11 illustrates an example of what the $\boldsymbol{\beta}$ vector might look like for various stimulus presentations at different times. The times themselves are given by the function $f(t)$ described above, but here, the activation levels (i.e., the y-axis) correspond to the values contained in $\boldsymbol{\beta}$, where $\beta_0 = 0$.

Corresponding to each stimulus presentation is an HRF, and the collection of HRFs can be assembled into a matrix $\mathbf{X}$. Like the vector $\boldsymbol{\beta}$, the HRF matrix $\mathbf{X}$ contains a vector corresponding to the baseline activation of the BOLD response, similar to a y-intercept term. As we will see below, a column within $\mathbf{X}$ contains elements equal to one to capture the baseline activation of the BOLD response once it is multiplied by $\boldsymbol{\beta}$. Beyond the baseline activation, the HRF matrix $\mathbf{X}$ contains R HRF time-series vectors for each stimulus presentation, shifted by the onset time as the columns. Given this, a value for each HRF must be specified at each unit of time $t \in \{1, 2, \ldots, T\}$. If for example, a stimulus is presented at $t = 3$ s and the units of

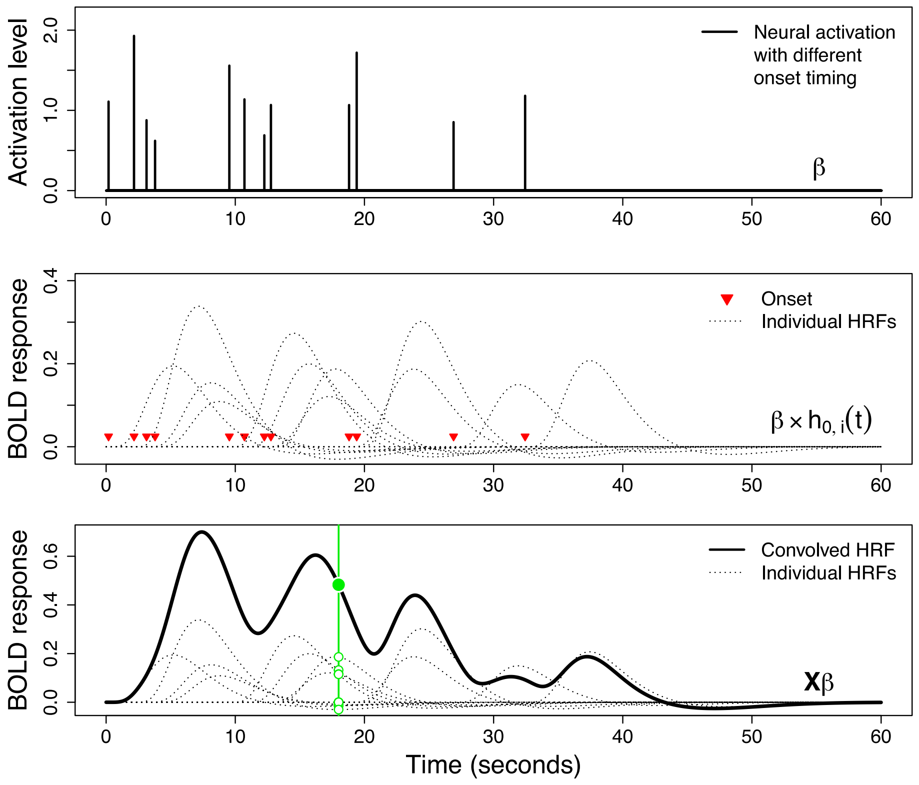

Fig. 11. Convolution of the hemodynamic function (HRF) with multiple times of neural activation. The three rows show how a BOLD response acquired from an fMRI scanner can be considered as a linear combination of individual hemodynamic responses evoked by each stimulus presentation. The first row shows the timing and strength of the neural activation. The middle row shows the individual hemodynamic responses (dotted lines) that correspond to the activation settings specified in the upper plot. The bottom row shows the convolution process across all stimulus presentations (bold line) along with the individual hemodynamic responses (dashed lines). The green line illustrates how the convolved HRF (filled circle) is a linear sum of the individual HRFs (empty circles) at that particular time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

time are in seconds, then the first two rows corresponding to that HRF's column would have zero values because the stimulus has yet to be presented. Given these specifications, we can define the HRF matrix as[2]

$$\mathbf{X} = \begin{bmatrix} 1 & h_{0,1}(1) & h_{0,2}(1) & h_{0,3}(1) & \cdots & h_{0,R}(1) \\ 1 & h_{0,1}(2) & h_{0,2}(2) & h_{0,3}(2) & \cdots & h_{0,R}(2) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & h_{0,1}(T) & h_{0,2}(T) & h_{0,3}(T) & \cdots & h_{0,R}(T) \end{bmatrix}. \quad (19)$$

The middle row of Fig. 11 shows an example of what the HRF matrix could look like once its columns have been scaled by the elements of $\boldsymbol{\beta}$ from the top row. Here, each stimulus presentation is marked by a red triangle, and the figure shows differences in the amplitude of the HRFs in proportion to the values of $\boldsymbol{\beta}$ from the top row. At this point, the middle row shows the individual HRFs together, shifted in time and scaled according to their neural activation according to the convolution operation in Eq. (17), but they have not been combined to form a prediction about the obtained BOLD response over the $T = 60$ units of time.

To produce the final predicted BOLD response, we simply sum up the neural activation from each of the individual HRFs shown in the middle row of Fig. 11. Given our definitions above, the linear summation operation can be simply expressed as $\mathbf{X}\boldsymbol{\beta}$, and the final convolved HRF is shown as the solid black line in the bottom row of Fig. 11. The green vertical line shows how the convolved HRF (filled circle) is constructed by summing up the individual HRFs (empty circle). Alongside the convolved HRF in the bottom panel are the individual HRFs so that one can see how individual stimulus presentations can have unexpected effects on the obtained BOLD response. For example, the convolved HRF has several modalities and undulations, some of which are produced by stronger neural activations (i.e., when β is large) and some of which are produced by more frequent stimulus presentations. In summary, Fig. 11 shows how the obtained BOLD response can be deconstructed when the stimulus presentation times are known, and the HRF amplitude parameters can be estimated.

Neural likelihood. Based on the LTI property of the hemodynamic response, the expected BOLD response (i.e., model prediction) is defined by the convolution of the time-series vector of the canonical HRF and the onset-time vector. Equivalently, it can be calculated by the sum of the individual time-series vectors $h(t)$, which are shifted by their onset time and scaled by the amplitude vector $\boldsymbol{\beta}$. In addition, we assume that the observed BOLD responses are perturbed by some statistical error $\epsilon(t)$, that captures random properties of the time series data that are not predicted by the model. Taken together, we can denote the neural response vector as

$$\begin{aligned} \mathbf{N}(t) &= \beta_0 + \sum_{i=1}^{R} h_i(t) + \epsilon(t) \\ &= \beta_0 + \sum_{i=1}^{R} \beta_i h_{0,i}(t) + \epsilon(t), \end{aligned}$$

where $t = 1, 2, \ldots, T$ represents a given time point, β_0 is the baseline activation level, and R is the number of stimulus presentations.

[2] Note that the design matrix $\mathbf{X}$ does not define columns representing signal drifts for practical purposes. Signal drifts refer to systematic patterns of the signal irrelevant to the task, and it is common to take them into consideration in the design matrix to statistically control exogenous effects. However, we decided to exclude them in this example for approachability.

The only free parameters are the amplitudes of the individual HRFs β_i, which will be estimated using a general linear model analysis.

To assess how well the set of model parameters capture the observed patterns in the data, we must assume a distribution for the error term $\epsilon(t)$. Conventionally, $\epsilon(t)$ is assumed to be distributed according to a normal distribution centered at zero with variance σ^2, such that

$$\epsilon(t) \sim \mathcal{N}(0, \sigma^2).$$

Given our definitions of $\boldsymbol{\beta}$ and $\mathbf{X}$, we can conveniently express the neural data probabilistically, such that

$$\mathbf{N} \sim \mathcal{N}_T(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_T), \tag{20}$$

where $\mathbf{N}$ is the BOLD time-series vector and $\mathbf{I}_T$ is a $T \times T$ identity matrix. Hence, letting N_t denote the BOLD response at time t, the likelihood for the neural activation parameters β and the noise term σ is

$$\mathcal{L}(\boldsymbol{\beta}, \sigma|\mathbf{N}) = \prod_{t=1}^{T} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{\left\{N_t - \left(\beta_0 + \sum_{i=1}^{R} \beta_i h_{0,i}(t)\right)\right\}^2}{2\sigma^2}\right]. \tag{21}$$

4.2.2. Behavioral submodel

While numerous theories have been postulated to explain and understand how humans make decisions in a two-alternative forced choice task, perhaps the most successful attempts involve sequential sampling theory. In their most basic form, models that embody sequential sampling theory assume that upon the presentation of a stimulus, an observer stochastically accumulates evidence until the level of evidence crosses a predetermined "threshold" amount, often referred to as a "boundary". In some model architectures, multiple boundaries exist that correspond to the two alternatives, whereas in other model architectures, multiple accumulators race toward a common boundary, where the accumulators correspond to the alternatives. Either way, once an accumulator reaches a boundary, a decision is made to correspond to the result of the stochastic process, and the outcome (i.e., the amount of time it took to reach the boundary and the boundary that was reached) corresponds to a predicted decision among the choice alternatives.

Within the sequential sampling family, the diffusion decision model (DDM; Ratcliff, 1978; Ratcliff & Rouder, 1998), has been especially successful in accounting for data from simple two-choice decision making tasks. The basic framework of the DDM is similar to the two-boundary models described above. However, what sets the DDM apart from other sequential sampling models is that evidence in the DDM is accumulated continuously over time rather than evidence arriving at discrete time steps (e.g., Merkle & Van Zandt, 2006; Ratcliff, Van Zandt, & McKoon, 1999; Smith & Van Zandt, 2000).

While the DDM has been incredibly successful since its inception in accounting for a variety of choice reaction time data, it has evolved significantly in that time. The original DDM (Ratcliff, 1978) included two sources of variability: within-trial variability in the rate of accumulation (i.e., the drift rate) and between-trial variability in both drift rate and nondecision time. These sources of variability were essential for the model to account for a variety of empirical benchmarks that early sequential sampling models could not produce (Ratcliff & McKoon, 2008; Ratcliff & Tuerlinckx, 2002).The modern DDM (Ratcliff & Rouder, 1998) includes an additional source of variability – between-trial variability in starting point – that allows it account for situations where error responses are faster than correct responses (i.e., fast errors). With the inclusion of this additional source of variability, the DDM has been able to account for a plethora of data from decision-making tasks spanning many domains and disciplines (e.g., Ratcliff, Thapar, & McKoon, 2006; Starns & Ratcliff, 2010; White, Ratcliff, Vasey, & McKoon, 2009).

For the purposes of this application, we chose to implement a simplified version of the DDM, which we refer to as the Wiener diffusion process (see Smith, 2000; Stone, 1960 for a detailed overview). The basic framework of the Wiener diffusion process typically involves four parameters representing response threshold, relative bias, nondecision time, and drift rate. The response threshold parameter α determines the amount of evidence needed to reach a decision. The relative bias parameter ω is often a reparameterization of the starting point z_0 relative to the response threshold:

$$\omega = \frac{z_0}{\alpha} \tag{22}$$

(Navarro & Fuss, 2009; Turner et al., 2015; Vandekerckhove, Tuerlinckx, & Lee, 2008). The relative bias is intended to reflect an observer's initial bias toward one alternative that is not based on the properties of the stimulus per se. The drift rate ξ, represents the mean rate of evidence accumulation. Finally, the nondecision time parameter τ is the summation of encoding time and motor execution time. Although these times are not typically interesting from a cognitive perspective, a nondecision time parameter is often used to allow the model to shift its predictions to be in line with observed data, much like an intercept term in regression models. With this set of parameters, the probability density function describing the distribution of finishing times (i.e., the times such that the accumulator reached a boundary) for the Wiener diffusion process – known as the "first passage of time" – is

$$\begin{aligned} f(t|\alpha, \omega, \xi_i, \tau) = {} & \frac{\pi}{\alpha^2} \exp\left(-\xi_i\alpha\omega - \frac{\xi_i^2(t-\tau)}{2}\right) \\ & \times \sum_{k=1}^{\infty} k \exp\left(-\frac{k^2\pi^2(t-\tau)}{2\alpha^2}\right) \sin(k\pi\omega) \end{aligned} \tag{23}$$

(Feller, 1968; Navarro & Fuss, 2009; Tuerlinckx, 2004). Eq. (23) only describes the times for the accumulator to reach a single boundary. To describe the times associated with the accumulator reaching the other boundary, we simply replace the drift rate ξ_i in Eq. (23) with $-\xi_i$ (because we are assuming no response bias in our model).

Much like the behavioral submodel used in the generative model in the tutorial above, this simplified Wiener diffusion model was chosen based on its simplicity, and it is not expected to fit data particularly well. However, as mentioned, diffusion models have a long standing history of success when fitting behavioral data from simple decision-making tasks, so it is a convenient choice for fitting behavioral data from a task such as ours. In practice, it is likely that a more complex form of diffusion model, such as one that includes several sources of variability (i.e., the "modern" DDM), would provide a more detailed account of these data.

Behavioral likelihood. With the probability density function for the upper and lower boundary given by Eq. (23), we can derive the likelihood function relating the model parameters to the data. In a two-alternative forced choice task, we obtain both a choice and a response time. Denoting the choice and response time on Trial i as c_i and t_i, respectively, the likelihood function is

$$\mathcal{L}(\theta|c, t) = \prod_{i=1}^{N} f(t_i|\alpha, \omega, (-1)^{c_i-1}\xi_i, \tau). \tag{24}$$

4.2.3. Linking equations

The final step in the model specification is to describe how the neural parameters δ inform the behavioral parameters θ. In our experiment, we present the subject with two different grating stimuli per trial, and the subject is asked to determine which of the two stimuli have the highest contrast. The neural submodel provides estimates of the amplitude parameter β of the HRF, which represents the strength of neural activity evoked by a stimulus. Following the LTI property of the HRF, a larger amplitude corresponds to greater neural activity. If we assume that each stimulus evokes a separate and distinct BOLD response, we can also assume that the amplitude of these BOLD responses could be related to the decision variable. For example, if the first stimulus is of high contrast and the second stimulus is of low contrast, we could compare the estimated β parameters for the two stimuli. As a general rule, we might expect that larger β values for the first stimuli relative to the β value of the second stimuli would produce larger probabilities of the subject declaring that the first stimulus was of higher contrast. Hence, comparing the magnitudes of the two β estimates should provide a reasonable proxy to the decision variable used by human observers.

Directed joint model. To map the neural activation parameters to the decision variable in the Wiener diffusion model in a Directed joint modeling framework, we simply assumed that the drift rate parameter on a given trial was given by the difference between the neural activations produced by the two stimuli. Specifically, letting $\beta_{2,i}$ and $\beta_{1,i}$ correspond to the neural activations following the presentations of the second and first grating stimuli, respectively, we set

$$\xi_i = \beta_{2,i} - \beta_{1,i}, \tag{25}$$

where ξ_i represents the drift rate parameter for Trial i. From the logic discussed above, it follows that ξ_i will tend to be larger when the second stimulus is of higher contrast relative to the first, which should produce a larger probability of choosing the second alternative relative to the first. In addition, the Wiener diffusion model makes strong predictions about the speed of each choice, such that larger ξ_i are associated with faster decisions. Both of these dynamics, the choice and response time, should provide strong constraints when mapping the neural activations to the decisions observed in the experiment.

Covariance joint model. To map the neural activation parameters to the decision variable in the Wiener diffusion model in a Covariance framework, we must specify the linking function $\mathcal{M}$ that connects the neural activations $\beta_{2,i}$ and $\beta_{1,i}$ on each trial i to the trial-specific drift rate ξ_i, as predicted by the behavioral submodel. Here, we choose to define the covariance structure in a similar manner as Turner et al. (2015), who used single-trial neural measures (i.e., the BOLD response) to inform the behavioral parameters of the DDM, creating the Neural Diffusion Decision Model (NDDM; Turner et al., 2015). Much like the NDDM, we assume that the single-trial drift rates ξ_i from the Wiener diffusion process and the difference between neural activations from the two contrast stimuli on each trial $\zeta_i = \beta_{2,i} - \beta_{1,i}$ come from a common distribution. Specifically we assumed that the linking function was multivariate normal (see Section 1.3 for a more detailed explanation), such that

$$(\zeta_i, \xi_i) \sim \mathcal{N}_2(\phi, \Sigma). \tag{26}$$

As the neural element of the hypermodel is defined as the difference between $\beta_{2,i}$ and $\beta_{1,i}$, we must also estimate one of the two β parameters. Without loss of generality, we assumed

$$\beta_{1,i} \sim \mathcal{N}(0, \sqrt{1000}^2), \text{ and}$$
$$\beta_{2,i} = \zeta_i + \beta_{1,i}.$$

Hence, ζ_i and $\beta_{1,i}$ are freely estimated, whereas $\beta_{2,i}$ is deterministic. This transformation is only necessary due to syntax constraints within JAGS, and our intention of relating a single neural activation parameter ζ_i to the drift rate parameter ξ_i in the Covariance joint model.

4.3. Fitting the model to data

To fit the Directed and Covariance joint models to data, four steps must be completed. First, the JAGS Wiener module must be installed so that Eq. (23) can be evaluated within JAGS (see Wabersich & Vandekerckhove, 2014 for details). Second, we must import the data from our experiment, so that the model can be fit to it. Third, JAGS code must be specified for the Directed and Covariance models, and finally we must use R to call and handle the sampling algorithms performed in JAGS. We now discuss each of these four steps in turn.

4.3.1. Installing the JAGS Wiener module

To implement the Wiener diffusion model in the joint modeling framework using JAGS, we must first install the JAGS Wiener Module (JWM; Wabersich & Vandekerckhove, 2014). To begin, it is important to verify that JAGS is installed and updated to the most recent version. With JAGS properly installed, the JWM can be installed by downloading the associated files from https://sourceforge.net/projects/jags-wiener/files/ and following the instructions described in Wabersich and Vandekerckhove (2014) for your operating system.

4.3.2. Importing data

Neural and behavioral data from our experiment are provided in the R data set `application_dataset.Rdata`.[3] This file will load six vectors into R: (1) a BOLD response vector preprocessed as percent signal change; (2) an onset-timing vector for the 40 independent stimuli; (3) a stimulus vector that provides the contrast values of the 40 stimuli, (4) a vector containing the response times for each trial; (5) a response vector, where responses are coded as 0 if the participant responded that the first stimulus had a higher contrast level, and a 1 if the participant responded that the second stimulus had a higher contrast; and (6) an accuracy vector coded as 1 for a correct response and 0 for an incorrect response. Note that no missing data exists in this data set. The following block of code will load the data files, apply appropriate transformations of the behavioral data, and construct a list object of the data for transmission to JAGS:

```r
# Load required packages and modules
require("rjags")
load.module("wiener")

# Load the data set
load("application_dataset.Rdata")

# Recode data
rt[temp.resp==0]=rt[temp.resp==0]*-1

# For the hypermodel
R = diag(rep(1, 2))

# Data
TR = 2
lenS = length(onset) # total number of stimuli
    presented in the block

dat = list(N = N, lenN = length(N), TR = TR, t =
    rt,
```

3 All files used in this tutorial can be downloaded from a repository on the Open Science Framework website: https://osf.io/qh7xr/?view_only=aafea8d894e74ee38ec67b7cc3b55780.

```
19            n.trials = length(rt), onset = onset, lenS = lenS,
20            a1 = 6, a2 = 16, b1 = 1, b2 = 1, c = 1/6)
```

Lines 1–3 load the packages and modules needed to sample from the posterior. The rjags package allows for JAGS software to be run in R, and the wiener package allows JAGS to use the functions associated with the JAGS wiener module (JWM). Lines 5–6 load the data set. Lines 8–9 recode the data so that responses to one stimulus in our two-alternative-forced-choice task have positive response times (RTs) and responses to the other stimulus have negative RTs. This is necessary as the first passage of time distribution, as specified in the JWM, is implemented as a univariate distribution. As such, to use the distribution, response times for responses associated with the lower (i.e., negative) boundary need to recoded as negative (Wabersich & Vandekerckhove, 2014). Lines 11–12 declare a matrix for storage to be used when specifying the hyper-prior. Finally, lines 14–20 extract the data relevant to our analyses and stores it in a list to be passed to JAGS. The vector N contains the BOLD responses from the ROI processed as percent signal change. The objects lenN, n.trials, and lenS are scalar values denoting the total number of BOLD measurements, trials, and stimuli, respectively. These will become important in our JAGS model code for calculating the likelihoods of the neural data and behavioral data and estimating the hemodynamic response function. Finally, the data list also contains the repetition time of the fMRI pulse sequence in TR, the onset times of each stimulus in the vector onset, and the shape parameters of the double-gamma function (a1, a2, b1, b2, and c).

4.3.3. JAGS code

In the previous sections, we have specified the model within JAGS in a separate text file; however, this is not essential to the implementation thanks to the additional flexibility afforded by the rjags package. For instance, the following block of code can be pasted into an R script and loaded into the workspace by simply running it:

```
 1 model.double.gamma.wiener = "
 2 model{
 3   # Likelihood
 4   ## The neural submodel
 5   for (i in 1:lenN) {
 6     N[i] ~ dnorm(muN[i], inv.sigma.sq)
 7     Npred[i] ~ dnorm(muN[i], inv.sigma.sq)
 8     muN[i] = beta0 + inprod(beta[], X[i, ])
 9   }
10
11   ### Define a design matrix using a double-gamma HRF
12   for (i in 1:lenS){
13     for (j in 1:lenN){
14       temp[j,i] = (j-1) * TR - onset[i]
15       Xt[j,i] = ifelse(temp[j,i] >= 0, temp[j,i], 0)
16       X[j,i] = (Xt[j,i]^(a1-1) * (b1)^(a1) * exp(-b1*Xt[j,i]) / exp(loggam(a1))) - c * (Xt[j,i]^(a2-1) * (b2)^(a2) * exp(-b2*Xt[j,i]) / exp(loggam(a2)))
17     }
18   }
19
20   ## The behavioral submodel
21   for (i in 1:n.trials){
22     xi[i] = beta[2*i] - beta[2*i-1]
23     t[i] ~ dwiener(alpha, tau, omega, xi[i])
24   }
25
26   # Prior
27   ## The neural submodel
28   inv.sigma.sq ~ dgamma(.001, .001)
29   sigma.sq = 1/inv.sigma.sq # Variance = 1/Precision
30   beta0 ~ dnorm(0, 0.001)
31   for (j in 1:lenS){
32     beta[j] ~ dnorm(0, 0.001)
33   }
34   ## The behavioral submodel
35   alpha ~ dunif(0.0001, 10)
36   tau ~ dunif(0, 0.04)
37   omega = 0.5
38 }
39 "
```

Lines 1–18 define the likelihood function for the neural submodel as described in Eq. (20). On line 7 of this snippet of code, we also declare the object Npred to collect samples from the posterior predictive distribution. These samples will be used in Section 4.4.1 to calculate the 95% credible interval of the posterior predictive distribution, which will allow us to examine how well the joint model will generalize to new and unseen data. Lines 11–18 define a design matrix where the columns are single HRFs with default-level activation amplitudes for each stimulus. Although convolution could be implemented by using matrix multiplication or a dot product in JAGS, a simpler way to define regressors is to manually define the HRFs for each stimulus shifted by their onset time. To do this, we first shift the timeline by the onset time (line 4). On line 15, any cells with negative value are replaced with 0 to avoid potential problems with negative inputs that may distort the HRF.[4] On line 6, we finally construct the HRFs for each individual stimulus in each column by inputting the timeline to the template HRF function. Note that as JAGS does not have a gamma function defined on a linear scale, we must implement the gamma function in the double-gamma HRF by exponentiating a log-transformed gamma function loggam(x). Lines 20–24 calculate the Wiener first passage of time distribution from Eq. (23) using the JWM. On lines 26–37, we specify the prior distributions for the parameters in both the neural and behavioral submodels. In this example, we chose diffuse priors for all parameters except the response threshold parameter α and the nondecision time parameter τ, as α must be positive and τ must be bounded between zero and the fastest response time.

To specify the Covariance joint model in JAGS, we simply replace lines 20–37 in the Directed joint model code above with:

```
 1 # Hypermodel
 2 for (i in 1:n.trials){
 3   beta[2*i] = zeta[i] + beta[(2*i-1)]
 4   t[i] ~ dwiener(alpha, tau, omega, xi[i])
 5   zeta[i] = drift[i,1]
 6   xi[i] = drift[i,2]
 7   drift[i,1:2] ~ dmnorm(hyper.Mu, hyper.inv.Sigma)
 8 }
 9
10 # Prior: Hypermodel
11 for (j in 1:2){
12   hyper.Mu[j] ~ dnorm(0, 0.001)
13 }
14 hyper.inv.Sigma[1:2, 1:2] ~ dwish(R[1:2, 1:2], 2)
15 # Convert hyper.inv.Sigma to hyper.Sigma for convenience
16 hyper.Sigma = inverse(hyper.inv.Sigma)
17
18 # Prior: For other parameters
19 inv.sigma.sq ~ dgamma(.001, .001)
20 sigma.sq = 1/inv.sigma.sq
21 beta0 ~ dnorm(0, 0.001)
22 alpha ~ dunif(0.0001, 10)
23 tau ~ dunif(0, 0.04)
24 omega = 0.5
25
```

[4] When using JAGS to shift the onset time to the stimulus presentation, negative values are produced as they are calculated relative to the presentation time. Because of this complication in JAGS, we replace negative values with zeros in the onset time matrix.

```
26 for (i in 1:n.trials){
27 beta[(2*i-1)] ~ dnorm(0, 0.001)
28 }
```

In this block of code, lines 1–8 specify the hyper-structure of the Covariance model. Here, the difference between the neural activation from the two stimuli ζ on each trial and the draft rate parameter ξ are sampled from a multivariate normal distribution with mean `hyper.mu` and standard deviation `hyper.inv.Sigma` and stored in the matrix `drift` (line 7; see Eq. (26)). On lines 5–6, we store ζ in the variable `zeta`, which is then used to calculate the second neural activation $\beta_{2,i}$ on line 3, and ξ in the variable `xi`, which is used to calculate the Wiener first passage of time distribution on line 4.

The remaining code specifies priors on the parameters in the neural and behavioral submodels and the linking function. Lines 10–14 specify the priors on the hyperparameters, where we specify a normal prior for `hyper.mu` and an invert Wishart prior on `hyper.inv.Sigma`. These priors establish conjugacy between the prior distribution and the posterior distribution, while still remaining uninformative. Lines 15–16 convert the precision matrix `hyper.inv.Sigma` to a covariance matrix `hyper.Sigma` by taking its inverse. Lines 18–24 specify diffuse priors for all the remaining neural and behavioral parameters except for the nondecision time parameter τ and the response threshold α, which are sampled from the same priors as in the Directed joint model above. Finally, lines 26–28 specify the prior for the first neural activation on each trial.

4.3.4. R Handler code

The final step is using R to interface with the JAGS software by using the commands internal to the `rjags` package. In parallel with our examples above, we first construct the JAGS model, generate some initial burn-in samples, and then sample from the desired posterior distribution. The following code samples from the Directed joint model:

```
1  # Initialization
2  model.dgw = jags.model(textConnection(model.
       double.gamma.wiener), data = dat, n.chains =
       3, n.adapt = 2000)
3
4  # Burn-in
5  update(model.dgw, n.iter = 4000, progress.bar = "
       text")
6
7  # Posterior sampling
8  dgw.out = coda.samples(model = model.dgw,
       variable.names = c("beta0", "beta", "sigma.sq
       ", "Npred", "alpha", "tau", "xi"),  n.iter =
       6000)
9
10 dgw.summary = summary(dgw.out)
```

Here, Lines 1–2 start the initialization process, lines 4–5 serve as a burn-in period, and lines 7–8 sample from the posterior using the Directed joint model and store the parameters of interest – namely the neural parameters `beta0` and `beta` and the behavioral parameters `sigma.sq`, `alpha`, `tau`, and `xi` – in the variable `dgw.out`. On line 10, the function `summary` will provide information about statistics such as mean, posterior standard deviation, and quantiles.

To sample from the Covariance joint model, simply replace the model code as described in Section 4.3.3 and add "`zeta`" and "`beta`" to the list of variables on line 9.

4.4. Results

To assess the model's performance, we present the results in two parts. First, we assess the degree to which the BOLD time series was properly recovered by showing predictions from each model against the observed data. Second, we evaluate the fidelity of the mapping hypothesis linking the parameters of the neural submodel to the observed behavioral data.

4.4.1. BOLD recovery

Before we examine the estimated posterior distributions for the model parameters, it is important to first look at how well each model fits the data. To assess how well each joint model captures the important trends in the neural data, we can compare each model's predictions for the BOLD response against the observed BOLD data. Fig. 12 shows the recovered BOLD response from each model estimates (solid line), along with the 95% credible interval of the posterior predictive distribution (dashed line), superimposed onto a plot of the observed BOLD data (dots). The prediction and 95% credible interval from the Directed joint model is illustrated in red, and the prediction and 95% credible interval from the Covariance joint model is illustrated in blue. The posterior predictive distribution allows us to determine how well the model would account for new and hypothetical data that may be observed from the same or a similar task. In other words, the posterior predictive distribution allows us to test for how well the model will generalize to new data, should it be collected. If the model is fitting the data appropriately, we should expect to see the solid line follow the pattern of the dots closely, and the majority of the observed data points should fall within the range of the posterior predictive distribution.

Fig. 12 shows that while each model did not capture the pattern of observed data perfectly, the predicted BOLD response from these models is aligned reasonably well with the majority of the fluctuations in the observed data. Furthermore, the majority of the observed data points fall within the 95% predicted credible set. Comparing across models, it appears that the Covariance joint model captures fluctuations in the time series slightly better at certain time points, but the overall patterns tend to be similar. Together, these evaluations suggest that each joint model provides a reasonable account of the neural data.

4.4.2. Linking hypothesis

Another important evaluation of joint models is in their characterization of the relationship between the two sets of variables. In our model, we have assumed that the differences in the neural activation is related to the parameters of the DDM in two different ways, and so we can compare whether these two model structures reveal any interesting differences. The top panel of Fig. 13 shows the differences in the neural activation parameters ξ_i (i.e., y-axis) against the response times (i.e., x-axis) for the Directed joint model (left panel) and the Covariance joint model (right panel). Here, trials in which the first stimulus was chosen as having the higher contrast value are represented as filled circles, whereas trials in which the second stimulus was chosen are represented as filled squares. The lines running through each point represent the length of the 95% credible interval of the posterior distribution.

Recall that ξ_i in the Directed joint model is the difference between the neural activation in response to the second stimulus minus the neural activation in response to the first stimulus (see Eq. (25)). If we assume that contrast levels and neural activation share a positive relationship, where a greater contrast level produces stronger neural activity, then we should see that larger values of ξ_i are associated with more frequent "second" stimulus responses, a pattern that is clearly observed in the left panel of Fig. 13. Additionally, the ξ_i parameters should be related to the response time. Specifically, larger values of ξ_i should reflect larger strengths of evidence toward one of the alternatives. Because larger strengths of evidence tend to produce faster response times in the Wiener diffusion process, we should see a negative correlation with the absolute value of the drift rate ξ_i and the response time, such that larger ξ_is (i.e., in an absolute sense) are associated with faster response times. The left panel of Fig. 13 affirms that this relationship exists for ξ_i and the response times in the Directed joint model.

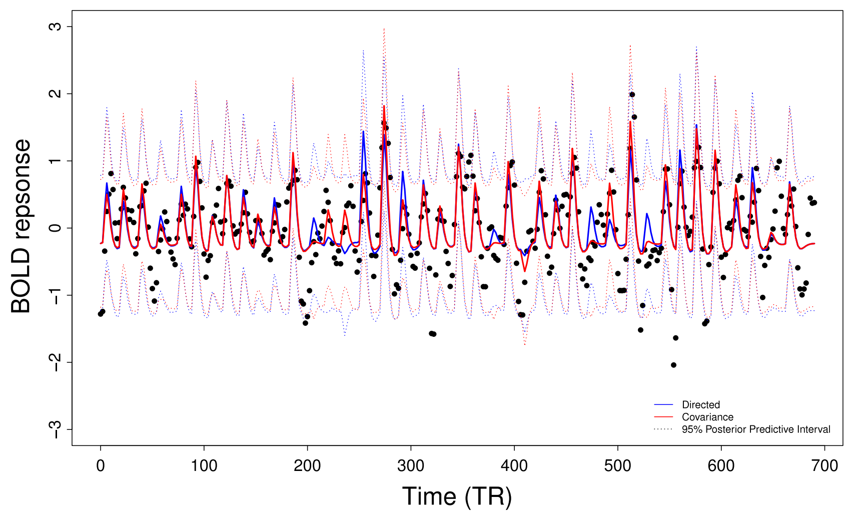

Fig. 12. A recovered BOLD response from the model estimates of the Directed and Covariance joint models. Each model's prediction of the BOLD response (bold line) and its 95% credible interval of the posterior predictive distribution (dotted line) were generated from the estimates of β_0, β_i ($i = 1, \ldots, 40$), and σ. The black dots represent the BOLD data observed in the experiment. The prediction for the BOLD response and 95% credible interval of the posterior predictive distribution from the Directed joint model is illustrated in blue, whereas the prediction for the BOLD response and 95% credible interval of the posterior predictive distribution from the Covariance joint model is illustrated in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

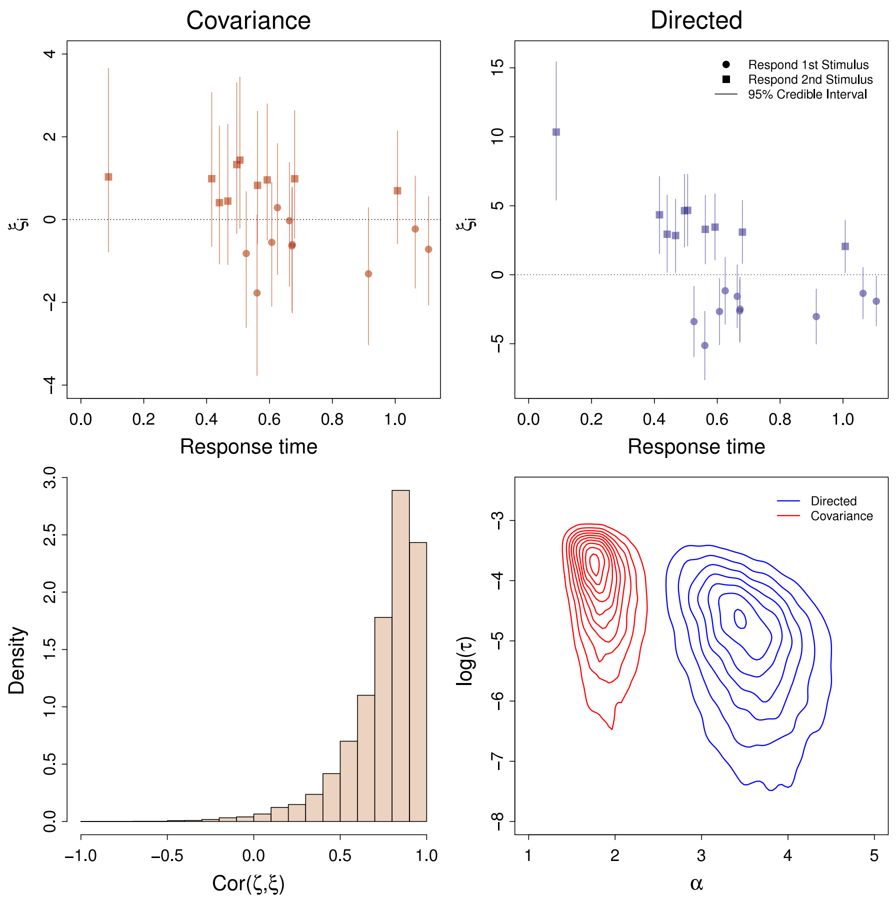

Fig. 13. Comparison of the behavioral submodel parameters between the Directed and Covariance joint models. The top left and top right panels describe associations between neural activation and the behavioral variables in the Directed and Covariance joint models. The predicted differences between the neural activation following the second and first stimulus presentations are shown (y-axis) against the response times (x-axis) for each of the 20 trials. Trials in which the first stimulus was chosen as the higher contrast stimulus are represented as filled circles, whereas trials in which the second stimulus was chosen are represented as filled squares. The bottom left panel illustrates the correlation between the difference in neural activations ζ and the drift rate parameter ξ in the Wiener diffusion model. The bottom right panel presents a contour plot of the joint posterior distribution of the nondecision time parameter τ and the response threshold α. The joint distribution from the Directed joint model is presented as blue contours, whereas the joint distribution from the Covariance model is presented as red contours.

In the Covariance joint model, the trial-specific difference between the neural activations ζ and the drift rate parameter ξ are sampled from a multivariate normal distribution. Under this parameterization, ξ_i is not directly defined by neural activation, but rather shares a common constraint with it. Because we assume that the difference in neural activations is mapped to the drift rate in the DDM, ζ_i and ξ_i should be positively correlated. As such, ξ_i and the choice response times should show a similar relationship to that of the Directed joint model, where ξ_i should be positively related to the probability of selecting the second stimulus, and larger deviations from zero should result in faster response times. The left panel of Fig. 13 supports the expected trends.

The top panel of Fig. 13 shows that the range of drift rates in the Covariance joint model is much smaller than the range of drift rates in the Directed joint model. This difference occurs because the Covariance joint model enforces a probabilistic constraint in the form of the prior, rather than a direct mapping as in the Directed joint model. In this analysis, the two different types of constrain impacted the scaling of the single-trial drift rate parameters, and this effect propagated to other model parameters as well. For example, the bottom right panel of Fig. 13 illustrates the joint posterior distribution of the nondecision time parameter τ and the response threshold α as two-dimensional contour plot. Here, the joint posterior distribution from the Directed joint model is illustrated in blue, and the joint posterior distribution from the Covariance joint model is illustrated in red. Because ξ is specified differently within the two joint models, the estimates of α and τ must adjust to still fit the behavioral data. Specifically, because the range of ξ in the Directed joint model is larger, the response threshold parameter α must increase to keep the overall signal-to-noise ratio similar in the accumulation process.

Finally, the bottom left panel of Fig. 13 displays the correlation between the difference in neural activations ζ and the drift rate parameter ξ, obtained in the Covariance joint model. In detail, posterior samples of the covariance matrix obtained at each sampling step were converted into a correlation coefficient by dividing the covariance term by the product of the standard deviation of ζ and ξ. In R, you can use the function `cov2cor` as a shortcut for this computation.

The bottom left panel of Fig. 13 reveals that ζ and ξ show a strong positive relationship, indicating that when the difference between the trial-specific neural activations increase, the drift rate parameter also increases. This correlation estimate is interesting because it provides some assurance that the direct linear mapping assumption used in the Directed joint model is a reasonable constraint. While the Directed and Covariance joint models were fit for illustrative purposes, because these two parameters show such a strong relationship in the Covariance joint model, in practice it seems reasonable to assume that the additional complexity added by the covariance structure in the Covariance joint model is unnecessary, and so one could forgo the Covariance model in favor of the simpler Directed model.

5. General discussion

In this tutorial, our goal was to provide an overview of the two main types of joint models – Directed and Covariance – and demonstrate that these models could be specified and fit to data using existing Bayesian software packages such as JAGS. To do so, we fit different joint models to both simulated and real data using JAGS, and determined that the models produced reasonable parameter estimates while capturing the essential trends present in the data to which it was fit.

In our experimental application, we fit both Directed and Covariance joint models to fMRI data from a contrast discrimination task. In both the Directed and Covariance models, the neural submodel used the double-gamma function to estimate the amplitude of the BOLD response for each grating stimulus presentation. The two models differed in the linking of the neural parameters δ to the behavioral parameters θ. Whereas the Directed joint model linked the two submodels by taking the difference between the estimated BOLD amplitudes for each pair of stimuli and used this value as the drift rate in a Wiener diffusion model to predict the choice response time data, the Covariance model assumed that the amplitude of each trial-specific neural activation and drift rate of the Wiener diffusion model were connected through an overarching multivariate normal distribution. To determine whether each model could account for the neural data, we compared the model's predicted BOLD response to the observed BOLD data and found that each joint model accounted for the data relatively well, with the covariance structure capturing the fluctuations in the BOLD data slightly better.

5.1. *Why use a joint model?*

In the introduction, we briefly listed a few benefits of modeling behavioral and neural data simultaneously. This list acknowledged that joint models are superior at (1) handling mismatched and missing data, (2) making predictions about either neural or behavioral data (3) characterizing the brain-behavior relationship, and (4) comparing different brain-behavior relationships across models. We will now discuss each of these in turn. We will also provide additional benefits and commentary on when one should consider using a joint model over a traditional unimodal model.

5.1.1. *Mismatched, missing, and predicting data*

Turner, Forstmann et al. (2013) demonstrated the utility of joint modeling, namely a Covariance joint model, when dealing with missing or mismatched neural or behavioral data by describing how a covariance structure can make predictions for missing data using only the relationship among the parameters after fitting the model. In the original paper, the authors demonstrated that, if presented with only one mode of a subject's data (i.e., only neural or behavioral data), fitting a joint model to the single mode can generate a predictive distribution for the missing data based on the relationship between the behavioral and neural submodels. In another application, Turner et al. (2016) showed that this covariance structure could be exploited to combine information across subjects who either provided EEG data, fMRI data, or both. Central to this modeling approach was that these subjects all provided behavioral data, and so a cognitive model was used to enforce a three way covariance structure between EEG, fMRI, and behavioral data.

5.1.2. *Exploring the brain-behavior relationship*

In the introduction, we briefly discussed that joint modeling was motivated by a desire to bridge the gap between Marr's (1982) levels of analysis and bring together the work of two relatively independent groups – cognitive neuroscientists and mathematical psychologists. By providing a framework that combined the work of both groups, joint modeling creates the ability to examine the physical properties of the brain to the higher-level cognitive mechanisms assumed by theoretical accounts of cognition (i.e., cognitive models). By linking the two levels of analysis, joint models can provide more complete and constrained theoretical accounts of cognition by exploiting brain-behavior relationships that are not possible with unimodal models. Perhaps more interesting is that the linking function can be specified in different ways, allowing researchers to use an explorative approach (e.g., the Covariance joint model), or a confirmatory approach (e.g., the Directed and Integrative joint models).

5.1.3. Flexibility

Another attractive feature of the joint modeling framework is that it does not restrict the choice of neural or behavioral submodels. In other words, the joint modeling framework allows any combination of neural and behavioral models, so those wishing to use a joint modeling framework can implement any behavioral and neural model they choose, given they can construct an appropriate and effective linking function. This provides us with unlimited freedom in modeling the joint distribution of data, and provides accessible methods for model comparison. By "plugging in" different cognitive models with a single neural model, one can compare joint relationships within the cognitive theory across the models (Turner, 2015).

5.2. Good modeling practices

The tutorial focused largely on the construction of each of the submodels and how these models could be specified and fit using JAGS while omitting other important aspects of the modeling procedure. In practice, however, there are a variety of "checks" one can do to evaluate a model and its fit to data. These checks include examining the efficacy of the sampling algorithm using chain diagnostics, performing recovery analyses, performing out-of-sample cross validation tests, and assessing model fits using fit statistics. JAGS provides the DIC value with its output, so evaluating relative fits using fit statistics is incredibly easy, and so we will avoid discussing this further. We now discuss a few of these good practices.

5.2.1. Chain diagnostics

One important component of assessing the accuracy of the model is assessing the sampling procedure itself. JAGS implements a standard MCMC sampling algorithm known as Gibbs sampling. While more advanced sampling algorithms exist, Gibbs sampling can sample "chains" of values from the posterior of interest of relatively simple models without issue, which suggests it is adequate for the models of interest in the current tutorial. Still, in practice, it is important to determine if the sampling procedure is drawing samples from the desired posterior distribution. If not, then the ability of the model to explain and account for data cannot be assessed accurately.

In addition to the parameter recovery analysis discussed in Sections 2.3 and 3.3, a simple way to assess the accuracy of the sampling algorithm is check for convergence and autocorrelation among the chains. These can be done informally by plotting the chains and performing a visual examination and/or more formally by calculating statistics such as the Gelman–Rubin diagnostic $\hat{R}$ (Gelman & Rubin, 1992).

Convergence. It is important that each chain moves from its starting point to a stationary distribution. This is know as convergence, and it is important because chains that have converged to a stationary distribution are no longer under the influence of their initial values. A quick and informal check for convergence is to look at the traceplot of each chain (using the `traceplot` function in JAGS). If the chains have converged, one should see what is commonly referred to as a "fuzzy caterpillar", where the chains are mixing properly and virtually indistinguishable from each other. Additionally, the mean of the chains should be relatively stationary and devoid of large movements in either direction (up or down). If the traceplot has these properties, one can assume that the chains have converged. If the chains are not mixing properly and the mean of the chains are moving up or down across iterations, then the sampling procedure should be rerun more iterations and/or the burn-in period should be rerun.

There are also more formal checks of convergence built into JAGS, such as the Gelman–Rubin (Gelman & Rubin, 1992) diagnostic, which determines if there is a significant difference between the within-chain variance and the between-chain variance. If the chains have converged, these variances should be equal. To calculate the Gelman–Rubin diagnostic in JAGS, one can simply use the function `gelman.diag`. This will provide you with a $\hat{R}$ point estimate for each parameter of interest and an upper confidence interval value. To assess convergence, the $\hat{R}$ point estimate should be close to $\hat{R} = 1.00$ (suggesting equal variance), with a general rule of thumb that they be less than $\hat{R} = 1.1$ (Lee & Wagenmakers, 2013). Anything larger than $\hat{R} = 1.1$ suggests the chains have not converged, and the sampler should be run with more iterations.

Autocorrelation. Another issue regarding the sampling procedure surrounds the idea of autocorrelation, where the current sample in a chain is highly dependent on the previous sample. If the chains are highly autocorrelated, the posterior estimates are highly correlated, and a substantial amount of information about the posterior distribution is potentially lost (i.e., the samples do not accurately represent the true posterior distribution). Checking for autocorrelation in JAGS can be done visually using the plotting functions `acfplot` or `autocorr.plot` or numerically using the function `autocorr`. These methods will calculate the autocorrelation function for each MCMC chain at each lag. The lag values will provide you with information about the autocorrelation value if the chains were "thinned" to various degrees, which means that only a certain number of samples are kept from every chain. Thinning the chains can be done using the `n.thin` argument in the sampling function. However, we should mention that the practice of thinning has recently been called into question, with opponents suggesting that thinning may reduce the efficiency of the sampler and result in a loss of information (Link & Eaton, 2012). When autocorrelation is a concern, running the chains for many more iterations may also help mitigate the effects of autocorrelation.

5.2.2. Parameter recovery analyses

In Sections 2.3 and 3.3, we performed a "recovery analysis" where we compared the predictions made by the joint model to the "ground truth", or the value used to generate the data. Here, we determined that because the true value of the parameters of interest was encompassed in the posterior distribution of the model, the parameters were accurately recovered. While this is considered a recovery analysis in its most basic form, in practice, recovery analyses simulate data from the model of interest thousands of times and across many different parameter values to ensure accuracy (Heathcote, Brown, & Wagemakers, 2015). Parameter recovery analyses should be performed regularly to provide assurance that the results of the model fitting procedure are not only valid, but also interpretable and generalizable.

5.3. Software alternatives

For the purposes of the current tutorial, JAGS was chosen based on its approachability, ease of use, and popularity among both novice and veteran cognitive modelers. However, there are a variety of other Bayesian software packages, such as Stan (Carpenter et al., 2016), that could have been used instead. Fortunately, the code provided here could be easily adapted to programs such as WinBUGS, OpenBUGS, or Stan, so the choice of software package is largely contingent on (1) the user's operating system, (2) the complexity of the joint model, or (3) the preference of the user. Regarding operating systems, Windows users have access to all the aforementioned software packages, so they are free to choose among these based on their needs. However, Mac and Linux users are encouraged to forgo WinBugs and use JAGS or Stan, as these software packages do not require the use of an emulator to run. Regarding complexity, the current paper demonstrates that the MCMC samplers built into these existing Bayesian software packages are adequate for sampling from the posterior of the joint models under consideration. However, programs such as Stan include more advanced sampling algorithms, such as the Hamiltonian Monte Carlo sampling algorithm, that can fit more complicated models or models featuring parameters that are highly correlated, such as the DDM. For the purposes of this tutorial, the Weiner first passage of time distribution has also been implemented in Stan (Carpenter et al., 2016), so the Directed and Covariance joint models in the experimental application could be adapted.

Of course, one could also use other sampling methods, such as DE-MCMC (ter Braak, 2006; Turner, Sederberg, Brown, & Steyvers, 2013) by writing their own posterior sampling code. With these methods, one would gain more control with the sampling procedure and not be limited to the algorithms built into the Bayesian software packages. This could be a compelling advantage if the neural submodel requires a finer design matrix that the one used in the code above, or if the behavioral submodel has a complicated mathematical form (Palestro, Sederberg, Osth, Van Zandt, & Turner, 2018; Turner, Dennis, & Van Zandt, 2013; Turner, Schley, Muller, & Tsetsos, 2017; Turner & Van Zandt, 2012). However, please note that these algorithms tend to be complex, especially for novice modelers, so we recommend this option only if one has prior programming knowledge.

5.4. *Joint modeling limitations*

To this point, the tutorial has demonstrated the utility of joint models as a way to comprehensively understand data by bridging levels of analysis. However, no tutorial would be complete without discussing potential limitations of our approach. We now discuss a few such limitations.

5.4.1. *Preprocessing and extracting the neural signal*

The first issue surrounds the preprocessing and extraction of neural signals. When processing the neural data for our experimental example, we decided to perform a region of interest (ROI) analysis and focus solely on the time-series vectors associated with the voxels of pre-specified regions in the brain. One alternative choice would have been to extend these analyses to the rest of the brain and perform a whole-brain analysis, which does not assume any predefined region of interest. In this approach, the time series data within each voxel across the entire brain during the contrast discrimination task could be used as the neural measures in the data analysis. With this time series data, we could assume the neural activation for each stimulus presentation on each trial in each voxel using the neural submodel and use the difference between these as the drift rate parameter in the behavioral submodel (as in Directed joint model), or assume that this difference and the drift rate parameter are sampled from a common distribution (as in the Covariance joint Model). An issue with this type of analysis is that ignores any potential spatial relationship between the voxels one is analyzing (but see Harrison, Penny, Ashburner, Trujillo-Barreto, & Friston, 2007; Penny, Trujillo-Barreto, & Friston, 2005; Woolrich, Jenkinson, Brady, & Smith, 2004). More pragmatically, it would require a significant increase in computation time as many more analyses would be performed.

5.4.2. *Measurements and experiment design*

The second issue is the use of stimulus-level or trial-level neural measures. As previously discussed, the joint modeling framework relies heavily on these measures. However, extracting signals at these levels, depending on the type of measure (e.g., EEG, fMRI hemodynamic responses) and/or the experimental design, can be incredibly difficult and computationally intensive. For example, in the experimental example, we estimated the amplitude of the neural signal at the stimulus-level from the convolved hemodynamic responses. However, due to the natural temporal dragging effect of the hemodynamic response, estimating stimulus-level amplitudes is increasingly difficult with increasing stimulus presentations.

The experimental design can also play a role in how easy or difficult it is to process and analyze the neural data. If an experimental design features a relatively short stimulus presentation time and interstimulus intervals, the estimation of the neural amplitudes can also be complicated. In our running example, the experiment is based on a rapid event-related design with a short stimulus presentation of 250 ms and a mean interstimulus interval of four seconds. As a result, the hemodynamic response overlaps considerably, which causes the estimation process to lose precision and may be the cause of the large posterior standard deviations in ξ_i that we observed in the experimental application.

For experimental designs that have a similar structure, there are several things that one can do to deal with the overlapping stimulus issue and produce more precise measurements. The first thing, as we have demonstrated, is to use a joint modeling framework. While the specific circumstances of the utility of joint modeling have recently been challenged (Hawkins, Mittner, Forstmann, & Heathcote, 2017), the additional constraint introduced by including the behavioral and neural measures into one framework may help mitigate potential problems introduced by experimental design and may lead to more accurate (i.e., more precise) posterior estimates (Turner, Forstmann et al., 2013; Turner et al., 2015; Turner, Wang et al., 2017). Additionally, one can use a sampling method that takes into account potential correlations among the parameters of the model used, such as DE-MCMC (ter Braak, 2006; Turner, Sederberg et al., 2013), which can automatically tune itself to the shape of the posterior and increase precision. Finally, one can simply change the experimental design by increasing the stimulus presentation time and interstimulus interval, making the estimation of the neural activation at the stimulus-level much easier.

5.4.3. *Model specification*

The third issue is that, in the Directed joint model in the experimental example, the choice of neural submodel used to estimate the hemodynamic response can have a large impact on the interpretation of the behavioral parameters. In our analysis of the experimental data above, we found that the posterior standard deviation of the behavioral submodel parameters differed substantially from a model that ignored the neural data completely. However, this effect was driven by the fact that the Directed joint model had larger drift rates than a behavioral-data only model, which in turn resulted in larger estimates for the threshold parameter α relative to the behavioral-data only model. Because Directed joint models rely so heavily on the transformation of neural submodel parameters to set the behavioral submodel parameters, the structure of the model can sometimes lead to a misinterpretation of model parameters as the effect on the behavioral submodel parameters is purely a statistical artifact and not an innate characteristic of the model.

To demonstrate the effect of neural model specification on the behavioral model parameters in a Directed framework, we can compare the impact of different HRF models on the posterior estimates of the behavioral parameters α and τ. Here, we fit another HRF model (Li, Lu, Tjan, Dosher, & Chu, 2008) defined as

$$\begin{aligned} h(t) &= \beta \frac{1}{\max(h_0(t))} h_0(t), \\ h_0(t) &= \left[\left(\frac{t}{d_1}\right)^{a_1} \exp\left(-\frac{t-d_1}{b_1}\right) \right. \\ &\quad \left. - c\left(\frac{t}{d_2}\right)^{a_2} \exp\left(-\frac{t-d_2}{b_2}\right) \right] \end{aligned} \tag{27}$$

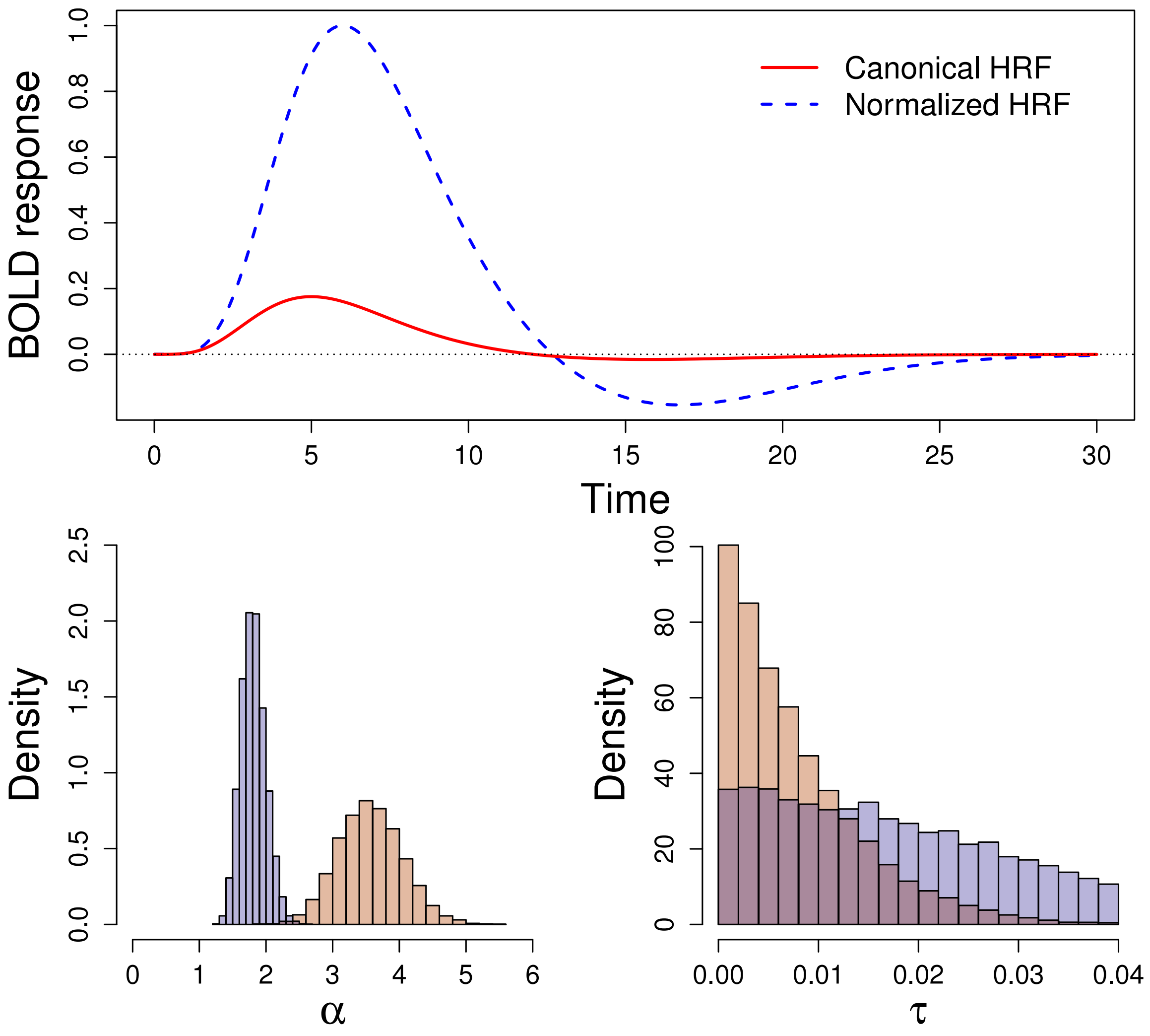

Fig. 14. Two HRF submodels and the associated posterior distributions of α and τ. The upper plot shows two different HRF models with $\beta = 1$. The lower plots are histograms of the posterior distributions of α and τ estimated with the two HRF models. The normalized HRF model results in different posterior distributions of both model parameters relative to the unnormalized HRF model.

where t represents time and β is the amplitude of the response. Shape parameters are assumed to have fixed values: $a_1 = 6$, $a_2 = 16$, $b_1 = 1$, $b_2 = 1$, and $c = 1/6$ with the constraint of $d_1 = a_1b_1$, $d_2 = a_2b_2$. In this new model, the default-level activation amplitude of the new HRF is higher than that of the canonical HRF model we used in the experimental example.

Fig. 14 shows the effect of different HRF models on the interpretation of behavioral model parameters: the threshold α and the nondecision time τ. Fig. 14 shows that the posterior distributions of α and τ show differences in both location and dispersion because the posterior estimates of the drift rate parameters ξ_i are scaled differently across the two neural submodels. Recall that we specified in Eq. (25) that on each trial, ξ_i should be directly related to the difference between the neural activation parameters $\beta_{2,i}$ and $\beta_{1,i}$. However, the mean and standard deviation of the difference variable may have very different properties compared to what is typically observed for drift rates in diffusion models. As a remedy, we could include parameters that standardize the difference, making ξ_i simply proportional to the difference:

$$\xi_i = \frac{\beta_{2,i} - \beta_{1,i}}{\sigma_\beta},$$

where σ_β could be a free parameter in the model. A model with this linking function would clearly have an effect on the estimates of the threshold parameter α, following the same logic illustrated in Fig. 14. From the comparison of the two HRFs in Fig. 14, it is evident that when using a Directed joint model, one must pay close attention to the specification of the neural submodel and the potential effects that it can have on the behavioral submodel (and vice versa) to avoid any misinterpretation of the estimates of the model parameters.

It is also important to note that the ability of the joint model to account for the data is contingent on the neural and behavioral submodels chosen. In both the simulation study and the experimental data example, the models used in the joint modeling framework were chosen based on simplicity (and for illustrative purposes) than for their ability to account for data. For example, the Wiener diffusion model used in the behavioral submodel of the experimental data example is a simple case of the diffusion model that is often used to account for choice response time data from a two-alternative forced-choice task. However, this simplified model lacks certain sources of variability (i.e., within- and between-trial variability in drift, between-trial variability in nondecision time, and between-trial variability in starting point) included in more modern variants of sequential sampling models. These additional trial-to-trial parameters have proven important in allowing off-the-shelf sequential sampling models to account for a much wider range of choice response time data (e.g., Ratcliff & Rouder, 1998). As a result, our simplified joint model may not account for the behavioral and neural data as well as a joint model that includes a more complex specification of trial-to-trial dependencies, such as those made by modern variants (Turner et al., 2015). So, even though the joint modeling framework offers additional constraints that can lead to more precise estimates, these benefits can only be enjoyed if the behavioral and neural models are properly specified and appropriate for explaining the neural activity or behavior of interest.

5.4.4. *The linking function*

Finally, as the neural measures obtained from an experiment are typically high-dimensional, great care must be taken to reduce the complexity of the linking function relating the neural measures to model parameters. In the simple experiment presented here, we had well-defined hypotheses about which brain areas should be related to the decision variables, and so specifying the linking function was straightforward. However, understanding how the brain relates to decision variables in other tasks can be quite complicated, and even subject to individual differences. In these contexts, a good approach is to specify generic linking functions that relate all voxels to the decision variables at hand, where the strength of the brain-behavior relationship can be inferred from the data. Recently, Turner, Wang et al. (2017) have shown that factor analytic linking functions can be an effective way to sieve through high-dimensional data in the search for key neural signals of interest. Turner et al. showed that factor analytic linking functions scale linearly with the complexity of neural data, whereas linking functions such as the multivariate normal distribution in Section 3 scale quadratically with the same complexity. In their analyses, not only were factor analytic linking functions more parsimonious, but they also performed better in cross-validation tests on the predictive performance of missing behavioral data.

5.5. *Conclusions*

Joint models provide an interesting opportunity for researchers who wish to enforce constraints on computational models from neurophysiology. However, until now, joint models have been unapproachable, as there was not a convenient way to apply them to data without extensive training or background in programming and statistics. The tutorial has demonstrated that developing and fitting joint models to data can be quite feasible through the use of general sampling algorithms such as those provided within JAGS.

Thinking in terms of Marr's levels of analyses, the measures obtained via cognitive neuroscience techniques provide exquisite details about the implementational level of analysis, whereas the mathematical model makes specific assumptions about the algorithms involved when completing the task (i.e., details about both the algorithmic and computational levels). By formally specifying a model of the neural measures and connecting the neural submodel's parameters to those assumed in the mathematical model, we can create new models that span all three of Marr's levels of analyses. Ultimately, we hope that our tutorial demonstrates how easily joint models can be implemented, making them more accessible in the emerging field of model-based cognitive neuroscience.

Appendix A. Generating data from the directed model

We will first provide R code that can be used to generate data from the Directed model described in Section 2. For the Directed model, the neural and behavioral data are characterized by the single-trial parameters δ and θ respectively, and ϕ and Σ control the distribution of trial-to-trial fluctuations observed in the neural activation parameter δ. To begin data generation within R, we must first specify the number of trials n and choose values for the elements of the vector phi and the matrix sigma, corresponding to ϕ and Σ in our model. These values will then be used to simulate the single-trial neural activation matrix Delta corresponding to δ in our model. To instantiate this in R, we can run the following code:

```
1  require("rjags")
2  require("mvtnorm")
3
4  # need both logit and logit^{-1} functions
5  logit=function(x)log(x/(1-x))
6  invlogit=function(x){1/(1+exp(-x))}
7
8  # set up model specification
9  n <- 500        # total number of trials
10
11 # establish the hyperparameters for delta
12 sig1 <- .5   # std. dev. of single-trial BOLD
       responses, ROI 1
13 sig2 <- .6   # std. dev. of single-trial BOLD
       responses, ROI 2
14 rho <- .4    # cor b/n brain activations
15
16 # set up hyper variance--covariance matrix Sigma
17 sigma <- matrix(c(sig1^2,          # element [1,1]
18         sig1*sig2*rho,  # element [1,2]
19         sig1*sig2*rho,  # element [2,1]
20         sig2^2),          # element [2,2]
21         2,2,byrow=TRUE)
22
23 # set up hyper mean vector phi
24 phi <- c(1.5,2)
25
26 # simulate single-trial delta matrix
27 Delta <- rmvnorm(n,phi,sigma)
```

In this block of code, lines 1–2 load two packages that are necessary to complete the steps in the tutorial. The first package, rjags, has been discussed previously and should already be installed on your machine. The second package, mtvnorm, may need to be installed using the install.packages() command. This package allows for the use of the multivariate normal distribution in both simulation and evaluations of the probability density function. Lines 5–6 declare two functions that will be necessary to map the parameters of the neural submodel to the parameters of the behavioral submodel. Lines 11–14 specify the individual elements of sigma (lines 17–21), which are then used in conjunction with phi (lines 23–24) to describe how the neural activation changes across trials in Delta (line 27). Here, Delta is a matrix of n random draws from a multivariate normal distribution with mean vector equal to phi and variance–covariance matrix equal to sigma.

With the neural parameters Delta generated, we can use the Delta matrix to (1) randomly generate the neural data N, (2) specify the behavioral parameters theta, and (3) then use theta to generate the behavioral data B. For a set of two regions of interest (i.e., Nroi=2), these three steps can be performed using the following code:

```
1  # generate observed variable nodes
2  ts <- seq(0,4,1)    # set of five scan times
       {0,1,2,3,4}
3  sig <- .5           # the standard deviation of
       BOLD responses
4
5  Nroi <- 2     # total number of ROIs
6
7  # declare some storage objects
8  N=array(NA,c(n,length(ts),Nroi))
9  B=numeric(n)
10 theta=numeric(n)
11
12 # set up regression parameters
13 beta <- c(.5,.3) # one beta parameter for each
       ROI
14
15 # loop over trials
16 for(i in 1:n){
17   for(k in 1:Nroi){
```

```
18     # N is a normal deviate with mean controlled by
          delta
19     N[i,,k]=rnorm(length(ts),Delta[i,k]*ts,sig)
20     }
21     # theta[i] is the single-trial behavioral
         parameter
22     theta[i] <- Delta[i,]  # B is a Bernoulli
         deviate with probability controlled by theta
23     B[i]=rbinom(1,1,invlogit(theta[i]))
24   }
25
26   # combine the generated data into a list to pass
         to JAGS
27   dat = list('n'=n,
28               'B'=B,
29               'N'=N,
30               'ts'=ts,
31               'Nt'=length(ts),
32               'sig'=sig,
33               'I0'=diag(2),
34               'n0'=2,
35               'phi0'=rep(0,2),
36               's0'=diag(2)),
```

This code produces the neural data for a set of ROIs, which in our experiment is limited to two (i.e., `Nroi=2`). Lines 7–10 declare objects for storage. `N` is defined an array with dimensions $500 \times 5 \times 2$ (number of trials by number of time points by number of ROIs). `B` and `theta` are vectors of length 500. Lines 12–13 specify the regression parameters `beta` that will be used to map the neural parameters in `Delta` to the behavioral parameters `theta`. This mapping process actually takes place on lines 15–25. Here, we start by looping over both trials (line 16) and ROIs (line 17) to generate the neural data. More concretely, for every trial and ROI, we need to generate a BOLD response value for each time point in the variable `ts` (line 2), which corresponds to T in Eq. (3). This is shown in lines 18–19 where five random values for the BOLD response are generated from a normal distribution with mean controlled by `Delta` and standard deviation `sig` – corresponding to σ in our model – that we specified in line 3. On lines 21–22, we generate the single-trial behavioral parameter `theta` using matrix multiplication for each of the $n = 500$ trials. Finally, on lines 23–24, we generate the behavioral data node `B` for each trial by drawing a single random value from a binomial distribution with probability given by the inverse logit transformation of the trial-specific value of `theta`.

The final step in the data generation process is to combine all of the variables into a single list to pass to JAGS. This is done in lines 27–37. This step can also be done directly in the JAGS code when specifying the JAGS sampler in Section 2.2.3. However, whether you specify it here or in the JAGS sampler code is a simply matter of preference.

As an aside, in our other applications we have used differential evolution with Markov chain Monte Carlo (DE-MCMC; ter Braak, 2006; Turner, Sederberg et al., 2013) to sample from the joint posterior (Turner, Forstmann et al., 2013; Turner et al., 2015, 2016). DE-MCMC is incredibly useful when the parameters of a model are highly correlated, such as the parameters of the DDM discussed below. When using DE-MCMC on models with correlated parameters, the algorithm can automatically tune itself to approximate the shape of the posterior during the sampling procedure. However, as the purpose of this tutorial is to show how joint modeling can be performed in existing Bayesian software packages such as JAGS, we will use the sampling procedures built into this program instead of advanced algorithms like DE-MCMC. With this is mind, the next section describes how to fit the Directed joint model using the JAGS software package. We first describe the JAGS code for specifying the Directed joint model, and then show how to integrate the JAGS code with the R program for convenience.

Appendix B. Generating data from the Covariance model

To simulate data from the Covariance joint model in Section 3, we can make use of code similar to that used to generate data from the Directed joint model, but with a few differences. Much like the Directed joint model, the neural and behavioral data are characterized by the single-trial parameters δ and θ, respectively. However, whereas ϕ and Σ represent the trial-to-trial fluctuations observed in the neural parameters δ in the Directed joint model, ϕ and Σ in the Covariance joint model describe how the neural and behavioral parameters fluctuate together from trial to trial. When generating data from the model in the code below, line 20 carries out the random sampling of the joint distribution of θ and δ.

To generate data from the model, we must first pick values for `phi` and `sigma` in order to produce the `DeltaTheta` matrix containing the single-trial parameters (δ, θ). Here, we assume that the Covariance joint model characterizes the data for one subject, so the rows of the `DeltaTheta` matrix represent values for (δ, θ) on individual trials. As such, we must also specify the total number of trials, which will determine the total number of rows in the `DeltaTheta` matrix. To generate data from the model, we can run the following block of code:

```
1   # set up model specification
2   n <- 500      # total number of trials
3
4   # establish the hyperparameters
5   sig1 <- .5     # std. dev. of single-trial BOLD
        responses
6   sig2 <- 1     # std. dev. of item memory strength
        (logit scale)
7   rho <- .6     # cor b/n brain activation and
        memory strength
8
9   # set up hyper variance--covariance matrix Sigma
10  sigma <- matrix(c(sig1^2,          # element [1,1]
11                    sig1*sig2*rho,   # element [1,2]
12                    sig1*sig2*rho,   # element [2,1]
13                    sig2^2),          # element
        [2,2]
14                    2,2,byrow=TRUE)
15
16  # set up hyper mean vector phi
17  phi <- c(2,0)
18
19  # simulate single-trial delta and theta matrix
20  DeltaTheta <- rmvnorm(n,phi,sigma)
```

As this code is virtually identical to the code used to generate data from the Directed joint model, we will not go into further details. However, we will point out that the single-trial neural activation matrix δ in the Directed joint model code is replaced by the single-trial neural and behavioral parameter matrix (δ_i, θ_i) on line 20. For a more detailed explanation of this block of code, please refer to Appendix A.

With the single-trial neural and behavioral parameters generated, we can now simulate the neural and behavioral data `N` and `B`. The following block of code deviates quite significantly from the code used to simulate data from the Directed joint model, so we will explain it in greater depth:

```
1   # generate observed variable nodes
2   ts <- seq(0,4,1)    # scan times
3   sig <- .5           # the std. dev. of BOLD
        responses
4
5   # declare some storage objects
6   N <- matrix(NA,n,length(ts))
7   B <- numeric(n)
8
9   # loop over trials
10  for(i in 1:n){
11    # N is a normal deviate with mean controlled by
         delta
```

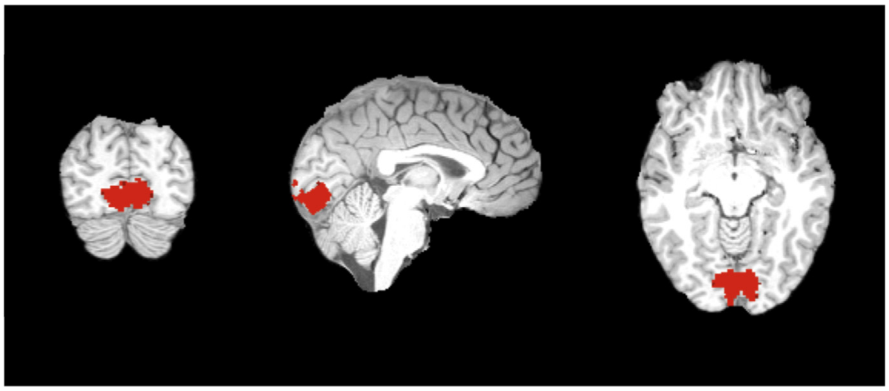

Fig. C.15. Region of interest (from left to right: coronal slice, sagittal slice, and axial slice).

```
12    N[i,]=rnorm(length(ts),DeltaTheta[i,1]*ts,sig)
13    # B is a Bernoulli deviate with probability
        controlled by theta
14    B[i]=rbinom(1,1,invlogit(DeltaTheta[i,2]))
15  }
16
17  # combine the generated data into a list to pass
      to JAGS
18   dat = list('n'=n,
19                'B'=B,
20                'N'=N,
21                'ts'=ts,
22                'Nt'=length(ts),
23                'sig'=sig,
24                'I0'=diag(2),
25                'n0'=2,
26                'phi0'=rep(0,2),
27                's0'=diag(2))
```

On line 2, we first define a set of time points in the object ts, which represents the set of scan times for which the neural activation is measured. Line 3 specifies the standard deviation of the BOLD signal over time, and lines 5–7 declare the objects N and B for storage. On lines 9–15, we generate the observed neural (line 12) and behavioral (line 14) data by looping over the total number of trials specified in the code above. Here, the neural data N is generated by randomly sampling five values (one for each time point in ts) from a normal distribution with mean equal the trial-specific δ value in the first column of the DeltaTheta matrix and standard deviation equal to sig. The behavioral data B is generated by randomly sampling from a binomial distribution with probability equal to the inverse logit of the trial-specific θ value in the second column of the DeltaTheta matrix. Finally, we combine the data in one list on lines 17–27 that will be passed to JAGS when we specify the Covariance joint model in the JAGS code below.

Appendix C. Definition of ROI

To locate the regions activated by the grating annulus used in the main task, we conducted a general linear model analysis using FSL (Smith et al., 2004). First, we defined a regressor as a time-series vector featuring the onset times of the each grating stimulus, regardless of contrast level. Then the contrast between the regressor and baseline was computed by FSL FEAT. Spatial smoothing was employed with the Full Width at Half Maximum (FWHM) of 5 mm. Highpass temporal filtering was applied simultaneously. Based on the z-statistics, any voxels with $z > 3.5$ were selected as potential target regions associated with completing the main task. This was done to help constrain the number of potential regions in the analysis.

The regions associated with the grating stimulus are broadly distributed over visual cortex. However, as it is known that amplitude of hemodynamic responses evoked by different contrast levels could differ across early visual areas (Li et al., 2008), we decided to limit the region of interest to V1. Standard masks of Brodmann Area (BA) 17 of both hemispheres are defined in Jülich Histological Atlas and were used to help locate V1 in a standard MNI space (Amunts, Malikovic, Mohlberg, Schormann, & Zilles, 2000; Eickhoff et al., 2005). After transforming the masks to a subject space, we defined our region of interest as any region with a significant activation level that showed overlap between BA17 and the target regions associated with completing the main task (see Fig. C.15).

References

Amunts, K., Malikovic, A., Mohlberg, H., Schormann, T., & Zilles, K. (2000). Brodmann's areas 17 and 18 brought into stereotaxic space - where and how variable? *Neuroimage*, *11*, 66–84.

Anderson, J. R. (2012). Tracking problem solving by multivariate pattern analysis and hidden Markov model algorithms. *Neuropsychologia*, *50*, 487–498.

Anderson, J. R., Betts, S., Ferris, J. L., & Fincham, J. M. (2010). Neural imaging to track mental states. *Proceedings of the National Academy of Sciences of the United States*, *107*, 7018–7023.

Boehm, U., Van Maanen, L., Forstmann, B., & Van Rijn, H. (2014). Trial-by-trial fluctuations in CNV amplitude reflect anticipatory adjustment of response caution. *Neuroimage*, *96*, 95–105.

Borst, J. P., & Anderson, J. R. (2017). A step-by-step tutorial on using the cognitive architecture ACT-R in combination with fMRI Data. *Journal of Mathematical Psychology*, *76*, 94–103.

Boynton, G. M., Demb, J. B., Glover, G. H., & Heeger, D. J. (1999). Neuronal basis of contrast discrimination. *Vision Research*, *39*, 257–269.

Boynton, G. M., Engel, S. A., Glover, G. H., & Heeger, D. J. (1996). Linear systems analysis of functional magnetic resonance imaging in human V1. *Journal of Neuroscience*, *16*(13), 4207–4221.

Brown, S., & Heathcote, A. (2008). The simplest complete model of choice reaction time: Linear ballistic accumulation. *Cognitive Psychology*, *57*, 153–178.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., et al. (2016). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*, 1–37.

Cassey, P. J., Gaut, G., Steyvers, M., & Brown, S. D. (2016). A generative joint model for spike trains and saccades during perceptual decision making. *Psychonomic Bulletin and Review*, *23*(6), 1757–1778.

Cavanagh, J. F., Wiecki, T. V., Cohen, M. X., Figueroa, C. M., Samanta, J., Sherman, S. J., et al. (2011). Subthalamic nucleus stimulation reverses mediofrontal influence over decision threshold. *Nature Neuroscience*, *14*, 1462–1467.

Coltheart, M. (2006). What has functional neuroimaging told us about the mind (so far)? *Cortex*, *42*, 323–331.

Daw, N. D., & Doya, K. (2006). The computational neurobiology of learning and reward. *Current Opinion in Neurobiology*, *16*, 199–20.

Daw, N. D., Niv, Y., & Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, *8*, 1704–1711.

de Hollander, G., Forstmann, B. U., & Brown, S. D. (2016). Different ways of linking behavioral and neural data via computational cognitive models. *Cognitive Neuroscience and Neuroimaging*, *1*, 101–109.

Dennis, S., & Humphreys, M. S. (2001). A context noise model of episodic word recognition. *Psychological Review*, *108*, 452–478.

Eickhoff, S. B., Stephan, K. E., Mohlberg, H., Grefkes, C., Fink, G. R., Amunts, K., et al. (2005). A new spm toolbox for combining probabilistic cytoarchitectonic maps and functional imaging data. *Neuroimage*, *25*, 1325–1335.

Eldridge, L. L., Knowlton, B. J., Furmanski, C. S., Bookheimer, S. Y., & Engel, S. A. (2000). Sentencing under uncertainty: Anchoring effects in the courtroom. *Nature Neuroscience*, *3*, 1149–1152.

Feller, W. (1968). *An introduction to probability theory and its applications* (Vol. 1). John Wiley: New York.

Forstmann, B. U., Tittgemeyer, M., Wagenmakers, E.-J., Derrfuss, J., Imperati, D., & Brown, S. (2011). The Speed-accuracy tradeoff in the elderly brain: A structural model-based approach. *Journal of Neuroscience*, *31*, 17242–17249.

Forstmann, B. U., & Wagenmakers, E.-J. (2014). *An introduction to model-based cognitive neuroscience*. New York: Springer.

Forstmann, B. U., Wagenmakers, E.-J., Eichele, T., Brown, S., & Serences, J. T. (2011). Reciprocal relations between cognitive neuroscience an formal cognitive models: opposites attract? *Trends in Cognitive Sciences*, *15*, 272–279.

Frank, M., Gagne, C., Nyhus, E., Masters, S., Wiecki, T. V., Cavanagh, J. F., et al. (2015). fMRI and EEG predictors of dynamic decision parameters during human reinforcement learning. *Journal of Neuroscience*, *35*(2), 485–494.

Frank, M. J., Seeberger, L. C., & O'Reilly, R. C. (2004). By carrot or by stick: cognitive reinforcement learning in parkinsonism. *Science*, *306*, 1940–1943.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis*. New York, NY: Chapman and Hall.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, *7*, 457–472.

Harrison, L. M., Penny, W., Ashburner, J., Trujillo-Barreto, N., & Friston, K. J. (2007). Diffusion-based spatial priors for imaging. *Neuroimage*, *38*, 677–695.

Hawkins, G., Mittner, M., Forstmann, B. U., & Heathcote, A. (2017). On the efficiency of neurally-informed cognitive models to identify latent cognitive states. *Journal of Mathematical Psychology*, *76*, 142–155.

Heathcote, A., Brown, S. D., & Wagemakers, E.-J. (2015). An introduction to good practices in cognitive modeling. In B. U. Forstmann, & E.-J. Wagenmakers (Eds.), *An introduction to model-based cognitive neuroscience* (pp. 25–48). New York: Springer.

Lee, M. D., & Wagenmakers, E.-J. (2013). *Bayesian modeling for cognitive science: A practical course*. Cambridge University Press.

Li, X., Lu, Z.-L., Tjan, B. S., Dosher, B. A., & Chu, W. (2008). Blood oxygenation level-dependent contrast response functions identify mechanisms of covert attention in early visual areas. *Proceedings of the National Academy of Sciences of the United States*, *105*, 6202–6207.

Link, W. A., & Eaton, M. J. (2012). On thinning of chains in MCMC. *Methods in Ecology and Evolution*, *3*, 112–115.

Love, B. C. (2015). The algorithmic level is the bridge between computation and brain. *Topics in Cognitive Science*, *7*.

Mack, M. L., Preston, A. R., & Love, B. C. (2013). Decoding the brain's algorithm for categorization from its neural implementation. *Current Biology*, *23*, 2023–2027.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. New York: Freeman.

Merkle, E. C., & Van Zandt, T. (2006). An application of the poisson race model to confidence calibration. *Journal of Experimental Psychology: General*, *135*, 391–408.

Mohammad-Djafari, A., & Féron, O. (2006). A Bayesian approach to change point analysis of discrete time series. *International Journals of Imaging Systems and Technology*, *16*, 215–221.

Mumford, J. A., Turner, B. O., Ashby, F. G., & Poldrack, R. A. (2012). Deconvolving BOLD activation in event-related designs for multivoxel pattern classification analyses. *NeuroIMage*, *59*, 2636–2643.

Navarro, D. J., & Fuss, I. G. (2009). Fast and accurate calculations for first-passage times in Wiener diffusion models. *Journal of Mathematical Psychology*, *53*, 222–230.

Nunez, M. D., Srinivasan, R., & Vandekerckhove, J. (2015). Individual differences in attention influence perceptual decision making. *Frontiers in Psychology*, *8*(18), 1–13.

Nunez, M. D., Vandekerckhove, J., & Srinivasan, R. (2017). How attention influences perceptual decision making: Single-trial EEG correlates of drift-diffusion model parameters. *Journal of Mathematical Psychology*, *76*, 117–130.

Osth, A. F., & Dennis, S. (2015). Sources of interference in item and associative recognition memory. *Psychological Review*, *122*, 260–311.

Palestro, J. J., Sederberg, P. B., Osth, A. F., Van Zandt, T., & Turner, B. M. (2018). *Likelihood-free methods for cognitive science*. New York: Springer.

Palmeri, T., Schall, J., & Logan, G. (2015). Neurocognitive modelling of perceptual decisions. In J. R. Busemeyer, J. Townsend, Z. J. Wang, & A. Eidels (Eds.), *Oxford handbook of computational and mathematical psychology*. Oxford University Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. San Francisco, CA: Morgan Kaufmann.

Penny, W. D., Trujillo-Barreto, N. J., & Friston, K. J. (2005). Bayesian fMRI time series analysis with spatial priors. *Neuroimage*, *24*, 350–362.

Plummer, M. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd international workshop on distributed statistical computing*, 2003.

Poldrack, R. A., Mumford, J. A., & Nichols, T. E. (2011). *Handbook of functional MRI data analysis*. New York: Cambridge University Press.

Purcell, B., Heitz, R., Cohen, J., Schall, J., Logan, G., & Palmeri, T. (2010). Neurally-constrained modeling of perceptual decision making. *Psychological Review*, *117*, 1113–1143.

Ranganath, C., Yonelinas, A. P., Cohen, M. X., Dy, C. J., Tom, S. M., & D'Esposito, M. (2004). Dissociable correlates of recollection and familiarity within medial temporal lobes. *Neuropsychologia*, *42*, 2–13.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, *85*, 59–108.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: theory and data for two-choice decision tasks. *Neural Computation*, *20*, 873–922.

Ratcliff, R., & Rouder, J. N. (1998). Modeling response times for two-choice decisions. *Psychological Science*, *9*, 347–356.

Ratcliff, R., Thapar, A., & McKoon, G. (2006). Aging, practice, and perceptual tasks: a diffusion model analysi. *Psychological and Aging*, *21*, 353–371.

Ratcliff, R., & Tuerlinckx, F. (2002). Estimating parameters of the diffusion model: Approaches to dealing with contaminant reaction time and parameter variability. *Psychonomic Bulletin and Review*, *9*, 438–481.

Ratcliff, R., Van Zandt, T., & McKoon, G. (1999). Comparing connectionist and diffusion models of reaction time. *Psychological Review*, *106*, 261–300.

Rissman, J., Gazzaley, A., & D'Esposito, M. (2004). Measuring functional connectivity during distinct stages of a cognitive task. *Neuroimage*, *23*, 752–763.

Schall, J. D. (2004). On building a bridge between brain and behavior. *Annual Review of Psychology*, *55*, 23–50.

Shiffrin, R. M., & Steyvers, M. (1997). A model for recognition memory: REM –retrieving effectively from memory. *Psychonomic Bulletin and Review*, *4*, 145–166.

Smith, P. L. (2000). Stochastic dynamic models of response time and accuracy: A foundational primer. *Journal of Mathematical Psychology*, *44*, 408–463.

Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E. J., Johansen-Berg, H., et al. (2004). Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage*, *23*, S208–S219.

Smith, P. L., & Van Zandt, T. (2000). Time-dependent Poisson counter models of response latency in simple judgment. *British Journal of Mathematical and Statistical Psychology*, *53*.

Starns, J. J., & Ratcliff, R. (2010). The effects of aging on the speed-accuracy compromise: boundary optimality in the diffusion model. *Psychological Aging*, *25*, 377–390.

Stone, M. (1960). Models for choice reaction time. *Psychometrika*, *25*, 251–260.

Teller, D. Y. (1984). Linking propositions. *Vision Research*, *24*, 1233–1246.

ter Braak, C. J. F. (2006). A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces. *Statistics and Computing*, *16*, 239–249.

Tuerlinckx, F. (2004). The efficient computation of the cumulative distribution and probability density functions in the diffusion model. *Behavior Research Methods, Instruments, & Computers*, *36*, 702–716.

Turner, B. M. (2015). Constraining cognitive abstractions through Bayesian modeling. In B. U. Forstmann, & E. Wagenmakers (Eds.), *An introduction to model-based cognitive neuroscience* (pp. 199–220). New York: Springer.

Turner, B. M., Dennis, S., & Van Zandt, T. (2013). Likelihood-free Bayesian analysis of memory models. *Psychological Review*, *120*, 667–678.

Turner, B. M., Forstmann, B. U., Love, B. C., Palmeri, T. J., & Van Maanen, L. (2017). Approaches to analysis in model-based cognitive neuroscience. *Journal of Mathematical Psychology*, *76*, 65–79.

Turner, B. M., Forstmann, B. U., Wagenmakers, E. J., Brown, S. D., Sederberg, P. B., & Steyvers, M. (2013). A Bayesian framework for simultaneously modeling neural and behavioral data. *Neuroimage*, *72*, 193–206.

Turner, B. M., Rodriguez, C. A., Norcia, T., Steyvers, M., & McClure, S. M. (2016). Why more is better: A method for simultaneously modeling EEG, fMRI, and Behavior. *Neuroimage*, *128*, 96–115.

Turner, B. M., Schley, D. R., Muller, C., & Tsetsos, K. (2017). Competing theories of multialternative, multiattribute preferential choice. *Psychological Review*. http://dx.doi.org/10.1037/rev0000089. Advance online publication.

Turner, B. M., Sederberg, P. B., Brown, S., & Steyvers, M. (2013). A method for efficiently sampling from distributions with correlated dimensions. *Psychological Methods*, *18*, 368–384.

Turner, B. M., Van Maanen, L., & Forstmann, B. U. (2015). Combining cognitive abstractions with neurophysiology: The neural drift diffusion model. *Psychological Review*, *122*, 312–336.

Turner, B. M., & Van Zandt, T. (2012). A tutorial on approximate Bayesian computation. *Journal of Mathematical Psychology*, *56*, 69–85.

Turner, B. M., Wang, T., & Merkel, E. (2017). Factor analysis linking functions for simultaneously modeling neural and behavioral data. *Neuroimage*, *153*, 28–48.

van Maanen, L., Brown, S. D., Eichele, T., Wagenmakers, E.-J., Ho, T., & Serences, J. (2011). Neural correlates of trial-to-trial fluctuations in response caution. *Journal of Neuroscience*, *31*, 17488–17495.

Vandekerckhove, J., Tuerlinckx, F., & Lee, M. D. (2008). A Bayesian approach to diffusion process models of decision-making. In V. M. Sloutsky, B. C. Love, & K. McRae (Eds.), *Proceedings of the 30rd annual conference of the cognitive science society*. Austin, TX: Cognitive Science Society.

van Ravenzwaaij, D., Provost, A., & Brown, S. D. (2017). A confirmatory approach for integrating neural and behavioral data into a single model. *Journal of Mathematical Psychology*, *76*, 131–141.

Wabersich, D., & Vandekerckhove, J. (2014). Extending JAGS: a tutorial on adding custom distributions to JAGS (with a diffusion model example). *Behavior Research Methods*, *46*, 15–28.

White, C., Ratcliff, R., Vasey, M., & McKoon, G. (2009). Dysphoria and memory for emotional material: A diffusion model analysis. *Cognition and Emotion*, *23*, 181–205.

Woolrich, M. W., Jenkinson, M., Brady, J. M., & Smith, S. M. (2004). Fully Bayesian spatio-temporal modeling of fMRI data. *IEEE Transactions on Medical Imaging*, *23*, 213–231.