Journal of Multivariate Analysis 174 (2019) 104532

Contents lists available at ScienceDirect

Journal of Multivariate Analysis

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

Sparse network estimation for dynamical spatio-temporal array models

Adam Lund *, Niels Richard Hansen

Københavns Universitet, Institut for Matematiske Fag, Universitetsparken 5, 2100 København Ø, Denmark

ARTICLE INFO

Article history:
Received 22 October 2018
Received in revised form 10 July 2019
Accepted 10 July 2019
Available online 16 July 2019

AMS 2010 subject classifications:
primary 62M10
secondary 62M40

Keywords:
GLAM
Non-differentiable regularization
Stochastic functional differential equation
VSD imaging data

ABSTRACT

Neural field models represent neuronal communication on a population level via synaptic weight functions. Using voltage sensitive dye (VSD) imaging it is possible to obtain measurements of neural fields with a relatively high spatial and temporal resolution. The synaptic weight functions represent functional connectivity in the brain and give rise to a spatio-temporal dependence structure. We present a stochastic functional differential equation for modeling neural fields, which leads to a vector autoregressive model of the data via basis expansions of the synaptic weight functions and time and space discretization. Fitting the model to data is a practical challenge as this represents a large scale regression problem. By using a 1-norm penalty in combination with localized basis functions it is possible to learn a sparse network representation of the functional connectivity of the brain, but still, the explicit construction of a design matrix can be computationally prohibitive. We demonstrate that by using tensor product basis expansions, the computation of the penalized estimator via a proximal gradient algorithm becomes feasible. It is crucial for the computations that the data is organized in an array as is the case for the three dimensional VSD imaging data. This allows for the use of array arithmetic that is both memory and time efficient. The proposed method is implemented and showcased in the `R` package `dynamo` available from CRAN.

1. Introduction

Neural field models are models of aggregated membrane voltage of a large and spatially distributed population of neurons. The neuronal network is determined by spatio-temporal synaptic weight functions in the neural field model, and we will refer to these weight functions as the *propagation network*. This network determines how signals are propagated hence it is of great interest to learn the propagation network from experimental data, which is the inverse problem for neural field models.

The literature on neural fields is vast and we will not attempt a review, but see [2,8] and the references therein. The typical neural field model considered is a deterministic integrodifferential equation. The inverse problem for the deterministic Amari equation was treated in [18] and [28], and stochastic neural field models were, for instance, treated in Chapter 9 in [8] and in [16]. One main contribution of the latter paper, [16], was to treat a stochastic version of the Amari equation in the well developed theoretical framework of functional stochastic evolution equations.

Despite the substantial literature on neural fields, relatively few papers have dealt directly with the estimation of neural field components from experimental data. Pinotsis et al. [27] demonstrated how a neural field model can be used

* Corresponding author.
E-mail address: adam.lund@math.ku.dk (A. Lund).

https://doi.org/10.1016/j.jmva.2019.104532
0047-259X/

as the generative model within the dynamical causal modeling framework, where model parameters can be estimated from electrophysiological data. The modeling of voltage sensitive dye (VSD) imaging data in terms of neural fields was discussed in [7], and Markounikau et al. [25] estimated parameters in a neural field model directly from VSD data.

In this paper, VSD imaging data is considered as well. This *in vivo* imaging technique has a sufficiently high resolution in time and space to detect propagation of changes in membrane potential on a mesoscopic scale, see [29]. A prevalent notion in the neuroscience literature is that the network connecting the neurons, and through which brain signals are propagated, is sparse, and that the propagation exhibits a time delay. If a spiking neuron, for instance, only affects neurons via synaptic connections to a very localized region the network is spatially sparse, while connections to remote regions result in temporal sparsity and long range dependence, see, e.g., [3,4,31,32,34]. Finally the possibility of feedback waves in the brain, e.g., as suggested in [29], could also be explained by spatio-temporal dynamics depending on more than just the instantaneous past. These considerations lead us to suggest a class of stochastic neural field models that allows for time delay, and a proposed estimation methodology that provides sparse nonparametric estimation of synaptic weight functions. Thus we do not make assumptions about spatial homogeneity or isotropy of the functional connectivity, nor do we assume that the signal propagation is instantaneous.

In order to derive a statistical model that, from a computational viewpoint, is feasible for realistically sized data sets, a time and space discretized version of the infinite dimensional dynamical model is obtained by replacing the various integrals with Riemann–Itô type summations and relying on an Euler scheme approximation. This approximation scheme makes it possible to derive a statistical model with an associated likelihood function such that regularized maximum-likelihood estimation becomes computationally tractable. Especially, we show that by expanding each component function in a tensor product basis we can formulate the statistical model as a type of multi-component linear array model, see [23].

The paper is organized as follows. First we give a brief technical introduction to the stochastic dynamical model that forms the basis for the paper. Then we present the aggregated results from the application of our proposed estimation methodology to part of a VSD imaging data set. The remaining part of the paper presents the derivation of the linear array model and the key computational techniques required for the actual estimation of the model using array data. The appendix contains further technical proofs, a meta algorithm and implementation details. Finally to further illustrate our results we also provide a Shiny app available at `shiny.science.ku.dk/AL/NetworkApp/` as well as supplementary material, [22], containing results from fitting the model to individual trials and the aggregated result for the entire data set.

2. A stochastic functional differential equation

The data that we will ultimately consider is structured as follows. With $\tau, T > 0$, $\mathcal{T} := [-\tau, T]$ and $N_x, N_y, M, L \in \mathbb{N}$ we record, to each of $N_t := M + L + 1$ time points

$$-\tau = t_{-L} < \cdots < t_0 < \cdots < t_M = T, \tag{1}$$

a 2-dimensional rectangular $N_x \times N_y$ image of neuronal activity in an area of the brain represented by the Cartesian product $\mathcal{S} := \mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^2$. These images consist of $D := N_x N_y$ pixels each represented by a coordinate (x_i, y_j) lying on a grid $\mathbb{G}^2 \subseteq \mathcal{S}$. To each time point each pixel has a color represented by a value $\boldsymbol{v}(x_i, y_j, t_k) \in \mathbb{R}$. Thus the observations are naturally organized in a 3-dimensional array $\boldsymbol{v} := \{\boldsymbol{v}(x_i, y_j, t_k)\}_{i,j,k}$ where the first two dimensions correspond to the spatial dimensions and the third dimension corresponds to the temporal dimension.

As such it is natural to view $\boldsymbol{v}$ as a discretely observed sample in time and space of an underlying spatio-temporal random field $\boldsymbol{V}$. Following Definition 1.1.1 in [1] any measurable map $\boldsymbol{V} : \Omega \to \mathbb{R}^{\mathcal{R}}$, with $\mathcal{R} \subseteq \mathbb{R}^d$, $d \in \mathbb{N}$, a parameter set, is called a $(d, 1)$-random field or simply a d-dimensional random field. Especially, for the brain image data, $\boldsymbol{V}$ is real valued with a 3-dimensional parameter set $\mathcal{R} := \mathcal{S} \times \mathcal{T}$ where $\mathcal{S}$ refers to space while $\mathcal{T}$ refers to time. We emphasize the conceptual asymmetry between these dimensions by calling $\boldsymbol{V}$ a spatio-temporal random field. For fixed t, as $\boldsymbol{V}(t) := \boldsymbol{V}(\cdot, \cdot, t) : \mathbb{R}^2 \to \mathbb{R}$ this model will inevitably be a stochastic dynamical model on a function space, that is an infinite dimensional stochastic dynamical model.

Following the discussion in the introduction above we propose to model the random neural field $\boldsymbol{V}$ via a stochastic functional differential equation (SFDE) with the propagation network incorporated into the drift as a spatio-temporal linear filter (a convolution) with an impulse-response function (convolution kernel) quantifying the network. The solution to the SFDE in a Hilbert space $\mathcal{H}$ is then the underlying time and space continuous model $\boldsymbol{V}$ for the data $\boldsymbol{v}$.

To introduce the general model more formally let $(\Omega, \mathcal{F}, \Pr)$ be a probability space endowed with an increasing and right continuous family $(\mathcal{F}_t)$ of complete sub-σ-algebras of $\mathcal{F}$. Let $\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) of continuous functions over the compact set $\mathcal{S} \times \mathcal{S}$. Suppose $(\boldsymbol{V}(t))_t$ is a continuous $\mathcal{F}_t$-adapted, $\mathcal{H}$-valued stochastic process and let $\mathcal{C} := \mathcal{C}([-\tau, 0], \mathcal{H})$ denote the Banach space of continuous maps from $[-\tau, 0]$ to $\mathcal{H}$. Then $(\boldsymbol{V}_t)_t$, where

$$\boldsymbol{V}_t := \{\boldsymbol{V}(t+s)\}_{s \in (-\tau, 0)}, \quad t \geq 0, \tag{2}$$

defines a $\mathcal{C}$-valued stochastic process over $\mathbb{R}_+$. We call $\boldsymbol{V}_t$ the τ-memory of the random field $\{\boldsymbol{V}(r)\}_{r \in \mathcal{S} \times \mathcal{T}}$ at time $t \geq 0$.

Let $\mu : \mathcal{C} \times [0, T] \to \mathcal{H}$ be a bounded linear operator and consider the stochastic functional differential equation (SFDE) on $\mathcal{H}$ given by

$$\mathrm{d}\boldsymbol{V}(t) = \mu(\boldsymbol{V}_t, t)\mathrm{d}t + \mathrm{d}\boldsymbol{W}(t). \tag{3}$$

Here $\boldsymbol{W}$ is a spatially homogeneous Wiener process with spectral measure σ (σ is a finite symmetric measure on $\mathbb{R}^2$) as in [26]. That is, $\boldsymbol{W}$ is a centered Gaussian random field such that $\{W(x, y, t)\}_t$ is a $(\mathcal{F}_t)_t$-Wiener process for every $(x, y) \in \mathbb{R}^2$, and for $t, s \geq 0$ and $(x, y), (x', y') \in \mathbb{R}^2$

$$\mathrm{E}\{\boldsymbol{W}(x, y, t)\boldsymbol{W}(x,' y', s)\} = (t \wedge s)c(x - x', y - y'). \tag{4}$$

Here $c : \mathbb{R}^2 \to \mathbb{R}$, the covariance function defined as the Fourier transform of the spectral measure σ, quantifies the spatial correlation in $\boldsymbol{W}$.

Compared to a typical SDE, the infinitesimal dynamic at time t as described by (3) depends on the past via the τ-memory of V in the drift operator μ. Hence processes satisfying (3) will typically be non-Markovian. We note that all the memory in the system is modeled by the drift μ.

The non-Markovian property makes theoretical results regarding existence, uniqueness and stability of solutions to (3) much less accessible. Corresponding to Section 0.2 in [11], in order to obtain theoretical results, it should be possible to lift Eq. (3) and obtain a Markovian SDE on the Banach space $\mathcal{C}$. Consequently an unbounded linear operator (the differential operator) then appears in the drift. It is outside the scope of this paper to pursue a discussion of the theoretical properties of (3). General theoretical results on SDEs on Banach spaces are not abundant, see, e.g., [9], where SDEs and especially SDDEs on Banach spaces are treated. Especially, Corollary 4.17 in [9] gives an existence result for a strong solution to (3). Also in [36] mild existence results are given for an SDDE on a Hilbert space and for the specification introduced next this result can be strengthened to a strong solution result. In general, one neccessary requirement for a solution to exist is, corresponding to the finite dimensional case, that the coefficient operators are Lipschitz continuous, which, e.g., the integral operator presented next satisfies.

2.1. Drift operator

The idea here is that the drift operator μ will capture both external input to the system (the brain in our context) as well as the subsequent propagation of this input over time and space. By decomposing the drift operator we obtain drift components responsible for modeling instantaneous effects and propagation effects respectively. To this end we will specify the drift operator by the decomposition

$$\mu(\boldsymbol{V}_t, t) := \mathrm{S}(t) + \mathrm{F}(\boldsymbol{V}_t) + \mathrm{H}\{\boldsymbol{V}(t)\}. \tag{5}$$

Here $\mathrm{S} : \mathcal{T} \to \mathcal{H}$, $\mathrm{S}(t)(x, y) := s(x, y, t)$, with $s \in L^2(\mathbb{R}^3, \mathbb{R})$ a smooth function of time and space, models a deterministic time dependent external input to the system. $\mathrm{H} : \mathcal{H} \to \mathcal{H}$, $\mathrm{H}\{\boldsymbol{V}(t)\}(x, y) := h(x, y)\boldsymbol{V}(x, y, t)$ where $h \in \mathcal{H}$, a smooth function of space, captures the short range (infinitesimal) memory in the system.

The long range memory responsible for propagating the input to the system over time and space is modeled by the operator $\mathrm{F} : \mathcal{C}([0, \tau], \mathcal{H}) \to \mathcal{H}$ given as the integral operator

$$\mathrm{F}(\boldsymbol{V}_t)(x, y) = \int_{\mathcal{S}} \int_{-\tau}^{0} w(x, y, x', y', r)\boldsymbol{V}(x', y', t + r)\mathrm{d}r\mathrm{d}x'\mathrm{d}y'. \tag{6}$$

Here $w \in L^2(E^5, \mathbb{R})$ is a smooth weight function quantifying the impact of previous states on the current change in the random field $\boldsymbol{V}$. Especially, the value $w(x, y, x', y', r)$ is the weight by which the change in the field at location (x, y) is impacted by the level of the field at location (x', y') with delay r. With this specification, a solution to (3), if it exists, can be written in integral form as

$$\boldsymbol{V}(x, y, t) = \int_0^t \left\{ s(x, y, u) + \int_{\mathcal{S}} \int_{-\tau}^{0} w(x, y, x', y', r)\boldsymbol{V}(x', y', u + r)\mathrm{d}r\mathrm{d}x'\mathrm{d}y' + h(x, y)\boldsymbol{V}(u) \right\} \mathrm{d}u + \int_0^t \mathrm{d}\boldsymbol{W}(x, y, u)\mathrm{d}u. \tag{7}$$

Thus (7) characterizes a solution to a stochastic delay differential equation (SDDE) on $\mathcal{H}$ with delays distributed over time and space (spatio-temporal distributed delays) according to the impulse-response function w. We can think of w as quantifying a spatio-temporal network in the brain that governs how the propagation of the input is to be distributed over time and space, and we will refer to w as the network function.

Standard neural field models usually include a non-linear transformation of $\boldsymbol{V}$, via a so-called gain function, inside the integral operator F. It is possible to include a known gain function transformation in (7) without substantial changes to our proposed methodology for estimating s, w and h. However, to the best of our knowledge the appropriate choice of gain function for empirical data has not been settled. It is, of course, possible to attempt to estimate the gain function from data as well, but to avoid complicating matters we proceed by regarding (7) as corresponding to a linearization of the unknown gain function. We note that the interpretation of w will always be relative to the choice of gain function, and it might not quantitatively represent physiological properties of the brain, if the gain function is misspecified.

Next we present an example where a statistical model based on the spatio-temporal SDDE model proposed above is fitted to real high dimensional brain image data. The derivation of this statistical model relies on a space–time discretization and is discussed in Section 4. We note that key elements in our approach involve expanding the network function (along with the other component functions) using basis functions with compact support in time and space domain. We then apply regularization techniques to obtain a sparse (i.e., space–time localized) estimate of the network.

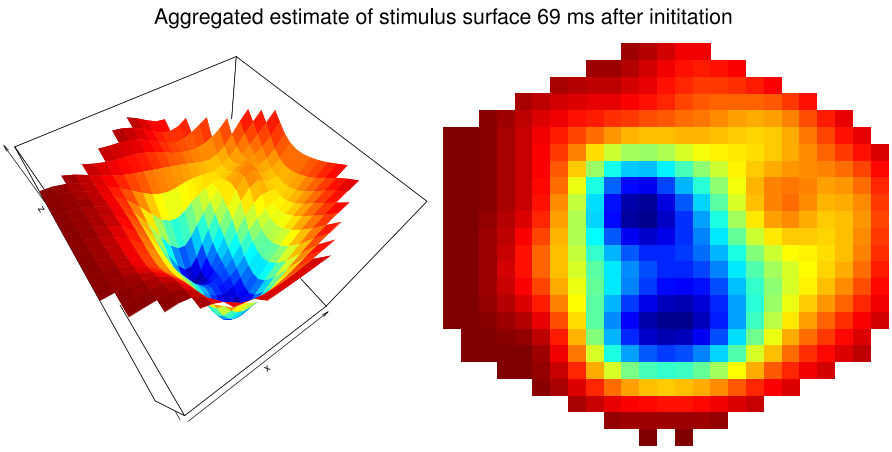


Fig. 1. Mean aggregated estimate 69 ms after stimulus onset. The white dot in the right panel indicates the pixel visualized in Fig. 2.

3. Brain imaging data

The data considered in this section consists of in vivo recordings of the visual cortex in ferret brains provided by Professor Per Ebbe Roland. In the experiment producing the data, a voltage sensitive dye (VSD) technique was used to record a sequence of images of the visual cortex, while presenting a visual stimulus, a stationary white square displayed for 250 ms on a gray screen, to the ferret. Each recording or trial is thus a film showing activity in a live ferret brain before, during and after a visual stimulus is presented. The purpose of the experiment was to study the response in brain activity to the stimulus and its propagation over time and space.

For each of a total of 13 ferrets the experiment was repeated several times producing a large-scale spatio-temporal data set with 10 to 40 trials pr. ferret resulting in 275 trials (films) in total. Each trial contains between 977 and 1720 images with time resolution equal to 0.6136 ms pr. image. For this particular data set each image was recorded using a hexagonal photodiode array with 464 channels and a spatial resolution equal to 0.15 mm pr. channel (total diameter is 4.2 mm), see [29]. The hexagonal array is then mapped to a 25×25 rectangular array to yield one frame or image. This mapping in principle introduces some distortion of the image but we ignore that here and consider 0.15 mm pr. pixel to be the spatial resolution.

Here we present an aggregated fit obtained by first fitting the above model to all trials for animal 308 (12 trials) each consisting of 977 25×25 images. Then by mean aggregating these single trial fits we obtain one fit based on all trials for this animal. Note that each of the 12 single trial fits for animal 308 is visualized in Section 3 in the supplementary material. We have carried out this analysis for all 13 animals – the entire data set – and present the visualizations of the remaining 12 aggregated fits in Section 2 in the supplementary material. The estimation procedure and a snippet of the data is available from CRAN via the `R` package `dynamo`, see [20].

For the analysis we let $L := 50$ thus allowing a 31 ms delay. For each single trial (film) the model is fitted using a lasso regularized linear array model derived in Section 4. The lasso regression is carried out for 10 penalty parameters $\lambda_1 > \cdots > \lambda_{10} > 0$. Here we present the fit for model 6, i.e., $\lambda = \lambda_6$, which we selected using 4-fold cross validation on the 12 trials, see Section 1 in the supplementary material. For additional details about the specific regression setup see Appendix C.1.

3.1. The aggregated stimulus and network estimates

Fig. 1 shows the estimate of the stimulus for all pixels 69 ms after onset. We argue that the "high stimulus" areas visible in Fig. 1 correspond to the expected mapping of the center of field of view (CFOV), see Fig. 1 in [19].

For the pixel indicated with a white dot in the right panel of Fig. 1, we show the raw data for each trial along with the trial specific estimate of the stimulus component and the aggregated stimulus component in Fig. 2. Notice that the estimated stimulus component shows both an on-signal after the stimulus start and an off-signal after the stimulus stop. Also notice the considerable variation over trials in the raw data with some trials displaying a clear signal and others almost no signal.

Visualizing the aggregated estimate of the network is more challenging as this is quantified by the function $w : \mathbb{R}^5 \to \mathbb{R}$. A Shiny app visualizing w is available online (see `shiny.science.ku.dk/AL/NetworkApp/`), and here we present various time and space aggregated measures of propagation effects — some of which are inspired by analogous concepts from graph theory.

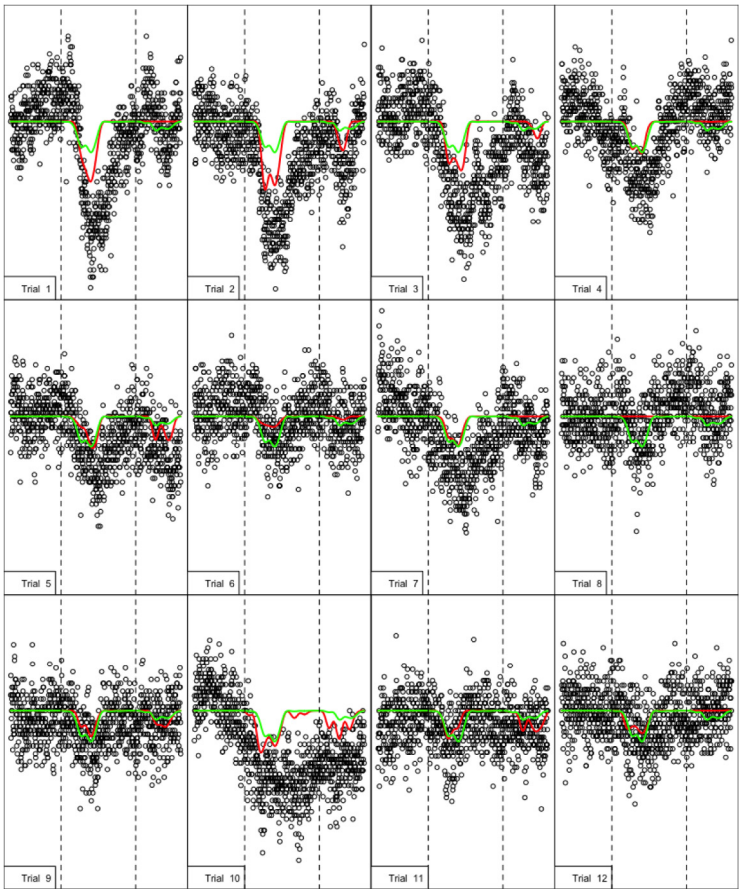


Fig. 2. The data observed at pixel (x, y) (indicated in Fig. 1) (black) and the estimate of $s(x, y, \cdot)$ for each trial (red) and the mean aggregated estimated (green). Dotted vertical lines indicate stimulus start and stop. Notice the considerable variation among the trials . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In Fig. 3 we plot the fitted version of the two bivariate functions $w^-, w^+ : \mathbb{R}^2 \to \mathbb{R}$ given by

$$w^-(x, y) := \frac{1}{\deg^-(x, y)} \int_{\mathcal{S}} \int_{-\tau}^{0} |w(x, y, x', y', t)| \mathrm{d}t \mathrm{d}x' \mathrm{d}y', \tag{8}$$

$$w^+(x', y') := \frac{1}{\deg^+(x', y')} \int_{\mathcal{S}} \int_{-\tau}^{0} |w(x, y, x', y', t)| \mathrm{d}t \mathrm{d}x \mathrm{d}y \tag{9}$$

where

$$\deg^-(x, y) := \int_{\mathcal{S}} \int_{-\tau}^{0} 1_{\{w(x,y,x',y',t)\neq 0\}} \mathrm{d}t \mathrm{d}x' \mathrm{d}y' \quad \text{and} \quad \deg^+(x', y') := \int_{\mathcal{S}} \int_{-\tau}^{0} 1_{\{w(x,y,x',y',t)\neq 0\}} \mathrm{d}t \mathrm{d}x \mathrm{d}y,$$

are the aggregated non-zero effects going in to (x', y') (indegree) respectively out from (x, y) (outdegree). Here $w^+(x', y')$ quantifies the effect of (x', y') on *all* other coordinates relative to the aggregated non-zero effects, that is time and space aggregated propagation effects *from* (x', y'). Similarly $w^-(x, y)$ quantifies time and space aggregated propagation effects *to* (x, y). Mean aggregated estimates of these functions are shown in Fig. 3.

From bottom panel in Fig. 3 we see that an area is identified which across all pixels has a relatively great weight on other pixels across the 12 trials. Thus this area is identified by the model as important in the propagation of neuronal activity across trials. We notice that this high output as quantified by w^+ overlaps with the strongest of the two high stimulus areas (CFOV) in Fig. 2. Thus the estimated weight functions propagate primarily the direct stimulus signal.

From the top panel we see that the pixels receiving propagation effects on the other hand are more scattered around the cortex. However, the high input areas overlap with both of the high stimulus areas (CFOVs) in Fig. 2 suggesting the existence of a propagation network connecting the high stimulus area and the low stimulus area and the immediate surroundings of the high stimulus area.

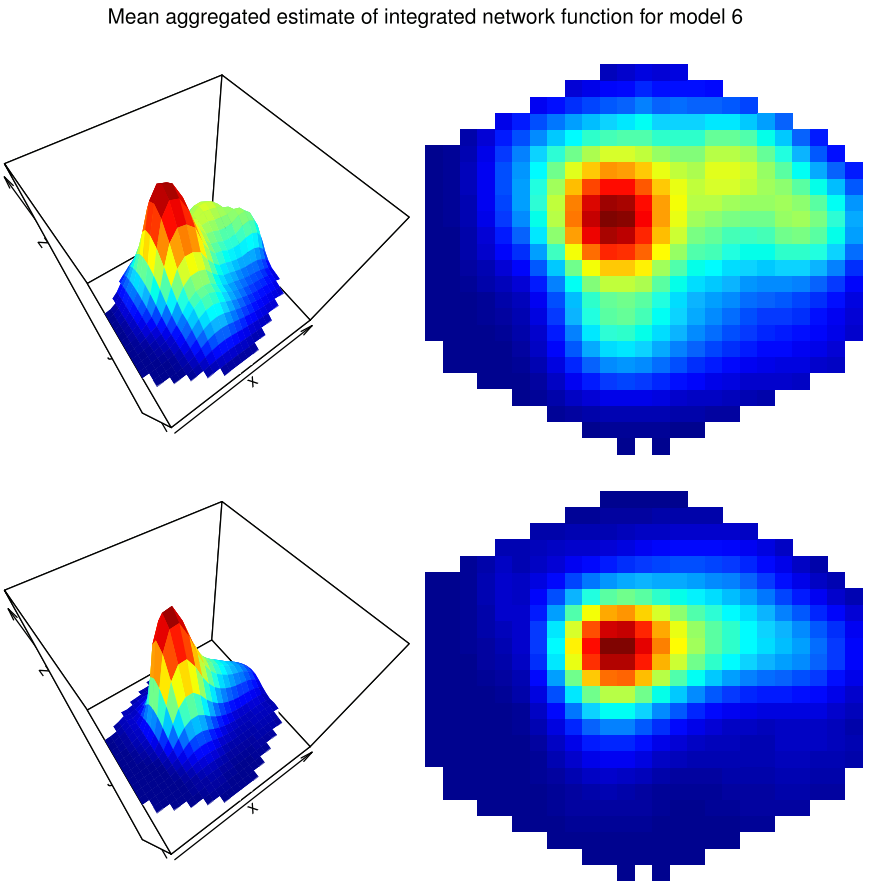


Fig. 3. The aggregated weight functions $\hat{w}^-$ (top) and $\hat{w}^+$ (bottom).

Next for fixed (x', y') consider quantifying the aggregated (in) effects from all points that lie s spatial units away and that arrive with a delay of t time units. Letting p denote the polar coordinate parametrization of the s-sphere $S_1(s)$ in $\mathbb{R}^2$ with center $(0, 0)$,

$$p : [0, 2\pi] \to S_1(s),$$

and compute the desired quantity for fixed s, t, x', y' as a curve integral

$$\int_{S_1(s)} w(p + (x', y'), x', y', t)\mathrm{d}p = \int_0^{2\pi} w\{p_1(r) + x', p_2(r) + y', x', y', t\}|p'(r)|\mathrm{d}r.$$

Integrating this over $\mathcal{S}$ we obtain a bivariate function

$$W(s, t) := \int \left\{ \int_{S_1(s)} w(p + (x', y'), x', y', t)\mathrm{d}p \right\} \mathrm{d}x'\mathrm{d}y'$$

giving the aggregated effects in the entire field as a function of spatial separation s (Euclidian distance) and temporal separation (time delay) t.

From bottom panel in Fig. 3 we see that an area is identified which across all pixels has a relatively great weight on other pixels across the 12 trials. Thus this area is identified by the model as important in the propagation of neuronal activity across trials. We notice that this high output as quantified by w^+ overlaps with the strongest of the two high stimulus areas (CFOV) in Fig. 2. Thus the estimated weight functions propagate primarily the direct stimulus signal.

From the top panel we see that the pixels receiving propagation effects on the other hand are more scattered around the cortex. However, the high input areas overlap with both of the high stimulus areas (CFOVs) in Fig. 2 suggesting the existence of a propagation network connecting the high stimulus area and the low stimulus area and the immediate surroundings of the high stimulus area.

Fig. 4 summarizes the network function as a function of temporal and spatial separation. The largest effects seem to occur with a delay of around 7 ms and have an effect on coordinates approximately 0–0.3 mm away. Especially the

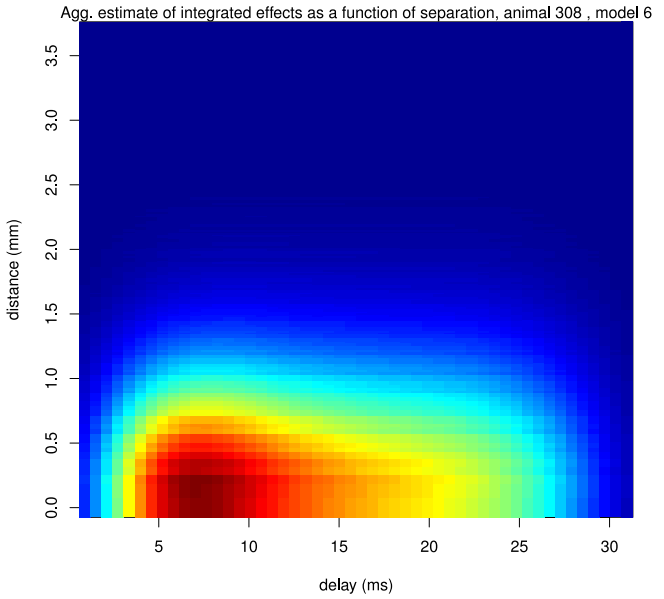


Fig. 4. Plot of $\hat{W}$ quantifying the mean aggregated estimate of propagation effects as a function of temporal and spatial separation.

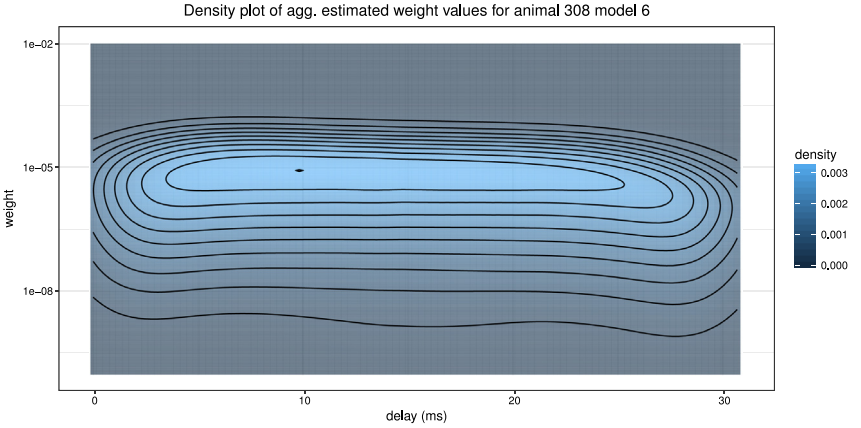


Fig. 5. Truncated density plots of the estimated weight values.

significant propagation effects do not seem to extend beyond 1.2 mm from the source and arrive with no more than 28 ms delay.

Finally, Fig. 5 shows a density plot of the estimated weight values in $\hat{w}$. The density plot is truncated as most weights are estimated to zero. From Fig. 5 we can see that the most frequent delay of the estimated effects is roughly 9 to 10 ms.

4. A linear model

The statistical model underlying the inferential framework used to obtain the results in Section 3 is based on a discretized version of the SFDE (3). The first step in our approach is to discretize space by aggregating the field over small areas. Let $(\mathcal{S}_{m,n})_{m,n} := (\mathcal{X}_m \times \mathcal{Y}_n)_{m=1,n=1}^{N_x,N_y}$ denote a partition of $\mathcal{S}$ with $D := N_x N_y$ elements with size $\mathrm{Leb}(\mathcal{S}_{m,n}) = \Delta_s > 0$ for all m, n. We have from the integral representation of $\boldsymbol{V}$ in (7) that

$$\int_{\mathcal{S}_{m,n}} \boldsymbol{V}(x,y,t)\mathrm{d}x\mathrm{d}y = \int_0^t \Big\{ \int_{\mathcal{S}_{m,n}} s(x,y,u)\mathrm{d}x\mathrm{d}y + \int_{-\tau}^0 \sum_{i,j} \int_{\mathcal{S}_{i,j}} \boldsymbol{V}(x',y',u+r) \int_{\mathcal{S}_{m,n}} w(x,y,x',y',r)\mathrm{d}x\mathrm{d}y\mathrm{d}x'\mathrm{d}y'\mathrm{d}r + \int_{\mathcal{S}_{m,n}} \boldsymbol{V}(x,y,u)h(x,y)\mathrm{d}x\mathrm{d}y \Big\} \mathrm{d}u + \int_{\mathcal{S}_{m,n}} \int_0^t \mathrm{d}\boldsymbol{W}(x,y,u)\mathrm{d}x\mathrm{d}y. \tag{10}$$

Here as noted in [26] the last term for each (x, y) is an Itô-integral with respect to a real valued Wiener process. Furthermore, using the covariance function for the random field from (4), the covariance of two such terms is

$$\begin{aligned}\mathrm{E}\left\{\int_{\mathcal{S}_{m,n}}\int_0^t \mathrm{d}\boldsymbol{W}(x,y,u)\mathrm{d}x\mathrm{d}y \int_{\mathcal{S}_{i,j}}\int_0^t \mathrm{d}\boldsymbol{W}(x,y,u)\mathrm{d}x\mathrm{d}y'\right\} &= \int_{\mathcal{S}_{m,n}}\int_{\mathcal{S}_{i,j}} \mathrm{E}\{\boldsymbol{W}(x,y,t)\boldsymbol{W}(x',y',t)\}\mathrm{d}x'\mathrm{d}y'\mathrm{d}x\mathrm{d}y \\ &= \int_{\mathcal{S}_{m,n}}\int_{\mathcal{S}_{i,j}} tc(x-x',y-y')\mathrm{d}x'\mathrm{d}y'\mathrm{d}x\mathrm{d}y.\end{aligned}$$

We then apply a Riemann type approximation of the space integrals over the partition sets $\mathcal{S}_{m,n}$ on the left and the right of (10). This leads us to consider the D-dimensional real valued stochastic process denoted $\tilde{\boldsymbol{V}}$, with the (m, n)th entry process given by

$$\tilde{\boldsymbol{V}}_{m,n}(t) = \int_0^t \left\{\tilde{\mathbf{S}}_{m,n}(u) + \int_{-\tau}^0 \sum_{i,j} \tilde{\boldsymbol{V}}_{i,j}(u+r)\int_{\mathcal{S}_{m,n}} w(x,y,x_i,y_j,r)\mathrm{d}x\mathrm{d}y\mathrm{d}r + \tilde{\mathbf{H}}_{m,n}\{\tilde{\boldsymbol{V}}(u)\}\right\}\mathrm{d}u + \int_0^t \mathrm{d}\tilde{\boldsymbol{W}}_{m,n}(u)\mathrm{d}u, \quad (11)$$

as a space discretized model for the random field $\boldsymbol{V}$, where $\tilde{\mathbf{S}}_{m,n}(u) := \Delta_s s(x_m, y_n, u)$ and $\tilde{\mathbf{H}}_{m,n}\{\tilde{\boldsymbol{V}}(u)\} := \Delta_s h(x_m, y_n)\tilde{\boldsymbol{V}}_{m,n}(u)$.

Notice that in (11) $\tilde{\boldsymbol{V}}(t)$ is an $N_x \times N_y$ matrix. However we might as well think of it as an $N_xN_y \times 1$ vector $\tilde{V}(t)$. When necessary we will distinguish between the matrix (array) form and the vector form using the following notation.

The mapping $[i_1, \ldots, i_d] : \bigtimes_{j=1}^d \{1, \ldots, N_{i_j}\} \to \{1, \ldots, \prod_{j=1}^d N_{i_j}\}$ is a bijection, and for an array $\boldsymbol{A} := (\boldsymbol{A}_{i_1,\ldots,i_d})_{i_1,\ldots,i_d}$ we denote the corresponding vector as $A_{[i_1,\ldots,i_d]} = \boldsymbol{A}_{i_1,\ldots,i_d}$. For $d = 2$ we may take $[i_1, i_2] := i_1 + (i_2 - 1)N_{i_2}$.

Using this notation let $\tilde{W} = \{(\tilde{W}_{[1,1]}(t), \ldots, \tilde{W}_{[N_x,N_y]}(t))\}_t$ denote a D-dimensional Brownian motion on $(\Omega, \mathcal{F}, \Pr)$ adapted to $(\mathcal{F}_t)_t$ with covariance matrix $\tilde{C}$ having rows given by

$$\tilde{C}_{[m,n]} := (c(x_m - x_1, y_n - y_1), \ldots, c(x_m - x_{N_x}, y_n - y_{N_y}))\Delta_s^2,$$

for each $[m, n] \in \{1, \ldots, D\}$, then $\tilde{\boldsymbol{W}}$ with $\tilde{\boldsymbol{W}}_{i,j} = \tilde{W}_{[i,j]}$ is the array version appearing in (11).

Aggregating $\boldsymbol{V}$ over the spatial partition $(\mathcal{S}_{m,n})_{m,n}$ leads to a multi-dimensional SDDE as an approximate model for our data. Such models are mathematically easier to handle compared to the random field model (7), see [24]. Especially, we can obtain a discrete time approximation to the solution (11) using an Euler scheme following [5]. The proof is in Appendix A.

Proposition 4.1. *Let $(t_k)_{k=-L}^M$ with $\Delta_t := t_{k+1} - t_k > 0$ denote a partition of $\mathcal{T}$. With $(\boldsymbol{w}_\ell)_{\ell=-L}^{-1}$ a sequence of weight matrices depending on the network function w, the solution $(\tilde{\tilde{V}}_k)_k$ to the forward Euler scheme*

$$\tilde{\tilde{V}}_{k+1} - \tilde{\tilde{V}}_k = \left\{\tilde{\mathrm{S}}(t_k) + \sum_{\ell=-L}^{-1} \boldsymbol{w}_\ell \tilde{\tilde{V}}_{k+\ell} + \tilde{\mathrm{H}}(\tilde{\tilde{V}}_k)\right\}\Delta_t + \tilde{C}\sqrt{\Delta_t}\epsilon_k, \quad (12)$$

where $(\epsilon_k)_k$ are independent and $\mathcal{N}(0, I_D)$ distributed, converges weakly to the (vectorized) solution $\tilde{V}$ to (11).

Note that for each k, $\tilde{\mathrm{S}}(t_k)$ and $\tilde{\mathrm{H}}(\tilde{\tilde{V}}_k)$ are just the vectorized versions of $\tilde{\mathbf{S}}(t_k)$ and $\tilde{\mathbf{H}}(\tilde{\tilde{\boldsymbol{V}}}_k)$ respectively with $\tilde{\tilde{\boldsymbol{V}}}$ the array version of $\tilde{\tilde{V}}$.

Next as the component functions; the stimulus functions s, the network function w and the short range memory h, are assumed to be square integrable we can use basis expansions to represent them, i.e., for $p_s, p_w, p_h \in \mathbb{N}$,

$$s(x,y,t) = \sum_{q=1}^{p_s} \alpha_q \phi_q(x,y,t), \quad (13)$$

$$w(x,y,x',y',t) = \sum_{q=1}^{p_w} \beta_q \phi_q(x,y,x',y',t), \quad (14)$$

$$h(x,y) = \sum_{q=1}^{p_h} \gamma_q \phi_q(x,y). \quad (15)$$

where $(\alpha_q)_q$, $(\beta_q)_q$ and $(\gamma_q)_q$ are three sets of basis coefficients and $(\phi_q)_q$ is generic notation for a set of basis functions. We let $p := p_s + p_w + p_h$ denote the total number of basis functions used. Using (13)–(15) we can formulate the dynamical model (12) as a vector autoregressive model. The proof is given in Appendix A.

Proposition 4.2. *There exist a $D \times 1$ vector y_k and a $D \times p$ matrix X_k such that we can write the dynamical model (12) as an autoregression*

$$y_k = X_k\theta + e_k, \quad k = 0, \ldots, M-1 \quad (16)$$

where $(e_k)_k$ *are i.i.d.* $\mathcal{N}_D(0, \Sigma)$ *with* $\Sigma := \tilde{C}^\top \tilde{C} \Delta_t$. *Defining*

$$y := \begin{pmatrix} y_1 \\ \vdots \\ y_M \end{pmatrix}, \quad X := \begin{pmatrix} X_1 \\ \vdots \\ X_M \end{pmatrix}, \tag{17}$$

the associated negative log-likelihood can be written as

$$\ell_X(\theta, \Sigma) = \frac{M}{2} \ln |\Sigma| + \left\| \left(I_M \otimes \Sigma^{-1/2}\right)(y - X\theta) \right\|_2^2, \tag{18}$$

with I_M *the* $M \times M$ *identity matrix.*

We note that due to the structure of the linear model (16) the MLEs of θ and Σ are not available in closed form. Especially the normal equations characterizing the MLEs are coupled in this model leading to a generalized least squares type estimation problem. Furthermore, the $DM \times p$ design matrix X will for realistically sized data become very large making it infeasible to fit the model (16) directly by minimizing (18) . Next we will discuss how to compute regularized estimates of the parameters θ and Σ by exploiting the array structure of the problem to reformulate the autoregressive model (16) as a (partial) linear array model. This in turn makes the computations involved in the fitting procedure feasible.

4.1. Penalized linear array model

In order to obtain time and space localized estimates of the component functions, we will minimize a regularized version of (18). Letting $\Omega := \Sigma^{-1}$ denote the precision matrix, this is achieved by solving the unconstrained problem

$$\min_{\theta \in \mathbb{R}^p, \Omega \in M^{D \times D}} \ell_X(\theta, \Omega^{-1}) + \lambda J_1(\theta) + \nu J_2(\Omega), \tag{19}$$

where J_1 and J_2 are convex penalty functions and $\lambda \geq 0$ and $\nu \geq 0$ the penalty parameters controlling the amount of regularization. For non-differentiable penalty functions J_i, $i = 1, 2$, solving (19) results in sparse estimates of θ and Ω. In the following we will use the lasso or ℓ_1 penalty, i.e., $J_1 = J_2 = \|\cdot\|_1$, see [33].

Following [30] we solve (19) using their approximate MRCE algorithm. In our setup the steps are as follows:

1. For fixed $\hat{\Omega}$ and each penalty parameter $\lambda_1 \geq \cdots \geq \lambda_K$, $K \in \mathbb{N}$, solve
$$\min_{\theta \in \mathbb{R}^p} \ell_X(\theta, \hat{\Omega}^{-1}) + \lambda_i J_1(\theta). \tag{20}$$
2. For $i \in \{1, \ldots, K\}$ let $\hat{\theta}_{\lambda_i}$ denote the estimate from step 1, let $\hat{\Sigma}_{R,i} := (y - X\hat{\theta}_{\lambda_i})^\top (y - X\hat{\theta}_{\lambda_i})$ and use graphical lasso, see [17], to solve
$$\hat{\Omega} := \arg\min_{\Omega} \operatorname{tr}(\hat{\Sigma}_{R,i}\Omega) - \ln |\Omega| + \nu J_2(\Omega).$$
3. Repeat step 1 and 2 with weighted data $\tilde{y} := \hat{\Omega}^{1/2} y$ and $\tilde{X} := \hat{\Omega}^{1/2} X$.

Now, solving the problem (20) requires a numerical procedure. However from a computational viewpoint evaluating ℓ_X given in (18) becomes infeasible as the design matrix X in practice is enormous. For instance for each of the trials considered in Section 3 and the setup described in Appendix C.1 the design X takes up around 200 GB of memory. In addition, even if we could allocate the memory needed to store X the time needed to compute the entries in X would be considerable and any algebraic operation involving X potentially computationally infeasible.

The solution to this computational problem is to choose basis functions in the representations (13)–(15) that will lead to a decomposable design matrix X. In particular, assume $p_x, p_y, p_t, p_\ell \in \mathbb{N}$ such that $p_s = p_x p_y p_t$, $p_w = p_x p_y p_x p_y p_\ell$ and $p_h = p_x p_y$. Then using tensor product basis functions we can write (13)–(15) as

$$s(x, y, t) = \sum_{j_1}^{p_x} \sum_{j_2}^{p_y} \sum_{j_3}^{p_t} \boldsymbol{\alpha}_{j_1, j_2, j_3} \phi_{j_1}^x(x) \phi_{j_2}^y(y) \phi_{j_3}^t(t), \tag{21}$$

$$w(x, y, x', y', t) = \sum_{j_1}^{p_x} \sum_{j_2}^{p_y} \sum_{j_3}^{p_x} \sum_{j_4}^{p_y} \sum_{j_5}^{p_\ell} \boldsymbol{\beta}_{j_1, j_2, j_3, j_4, j_5} \phi_{j_1}^x(x) \phi_{j_2}^y(y) \phi_{j_3}^x(x') \phi_{j_4}^y(y') \phi_{j_5}^l(t), \tag{22}$$

$$h(x, y) = \sum_{j_1}^{p_x} \sum_{j_2}^{p_y} \boldsymbol{\gamma}_{j_1, j_2} \phi_{j_1}^x(x) \phi_{j_2}^y(y). \tag{23}$$

Note that $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are the array versions of α, β and γ respectively.

Using (21)–(23), it turns out that X can essentially be componentwise tensor factorized, see the proof of Proposition 4.3 in Appendix A, implying that we can perform algebraic operations involving X without having to construct it. This is achieved by using the so called rotated H-transform from [10].

Definition 4.1. Let $\boldsymbol{A}$ be $p_1 \times \cdots \times p_d$ and X_i $n_i \times p_i$ for $i \in \{1, \ldots, d\}$. The rotated H-transform is defined as the map ρ such that

$$X_d \otimes \cdots \otimes X_1 \mathrm{vec}(\boldsymbol{A}) = \mathrm{vec}[\rho\{X_d, \rho(\ldots, \rho[X_1, \boldsymbol{A}])\}], \tag{24}$$

where vec is the vectorization operator.

The above definition is not very enlightening in terms of how ρ actually computes the matrix–vector product. These details can be found in [10]. Definition 4.1 shows, however, that for a tensor structured matrix we can compute matrix–vector products via ρ using only its tensor components. This makes ρ very memory efficient. Furthermore, as discussed in [6,10,13], ρ is also computationally efficient as the left hand side of (24) takes more multiplications to compute than the right hand side. In this light the following proposition is relevant. The proof is in Appendix A.

Proposition 4.3. *There exist arrays ϕ^x, ϕ^y, ϕ^t, Φ^{xyt}, Φ^x, Φ^y and $\boldsymbol{C}$ such that*

$$X\theta = \mathrm{vec}\Big[\rho[\phi^t, \rho\{\phi^y, \rho(\phi^x, \boldsymbol{\alpha})\}] + \rho[\Phi^{xyt}, \rho\{\Phi^y, \rho(\Phi^x, \boldsymbol{\beta})\}] + \tilde{\tilde{\boldsymbol{V}}}_{-1} \odot \boldsymbol{C}\Big]. \tag{25}$$

Here $\tilde{\tilde{\boldsymbol{V}}}_{-1} := (\tilde{\tilde{\boldsymbol{V}}}_k)_{k=-1}^{M-1}$, ϕ^{x} and Φ^{x} are $N_{\mathrm{x}} \times p_{\mathrm{x}}$ for $\mathrm{x} \in \{t, x, y\}$, Φ^{xyt} is $M \times p_x p_y p_\ell$, $\boldsymbol{C}$ is $N_x \times N_y \times M$ and $\odot$ denotes the Hadamard product.

Proposition 4.3 shows that we can in fact write the linear model (16) as a three component partial 3-dimensional linear array model, see [23] for multi-component array models. Note we say partial since the last component does not have a full tensor decomposition. This in turn has important computational consequences for the model fitting procedure as shown in [23], where a gradient descent proximal gradient (gd-pg) algorithm is proposed for this kind of model setup. The gd-pg algorithm uses a minimal amount of memory and can exploit efficient array arithmetic (the ρ operator in (25)) while solving the non-differentiable penalized estimation problem (20). Especially it is fairly straightforward to construct an estimation procedure based on the gd-pg algorithm that will solve the penalized problem (20) for all three model components at once, that is minimize the penalized log-likelihood over all parameters α, β, γ simultaneously.

4.2. Block relaxation model fitting

We propose a variation of the gd-pg algorithm, which uses a block relaxation scheme, to solve (20) for one parameter block (e.g., $\boldsymbol{\alpha}$) at a time while fixing the rest ($\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$), see [14]. We note that due to the additive mean structure this approach corresponds to a back-fitting algorithm where the partial residuals, resulting from fixing all but one parameter block, are fitted to data for one model component at the time.

The block relaxation approach is directly motivated by the application to neural field models as a way to achieve a particular structure on the estimated stimulus component. Without any structural constraints, the stimulus component would effectively just be a smoothing of the observed signal, and we would not achieve the decomposition of the drift into a direct stimulus component and a propagation component. We argue that the temporal evolution in the direct reaction to the stimulus is homogeneous across space, though the size of the direct stimulus component vary. Thus the stimulus component consists of a spatially modulated temporal signal.

This model constraint corresponds to assuming that s can be factorized into a product of a bivariate function of space and a univariate function of time. Representing each of these factors in a tensor basis then leads to the representation

$$s(x, y, t) = \sum_{k=1}^{p_t} \sum_{j=1}^{p_y} \sum_{i=1}^{p_x} \phi_k^t(t) \phi_j^y(y) \phi_i^x(x) \zeta_k \boldsymbol{\eta}_{i,j}.$$

Using a block relaxation scheme we can place the reduced rank restriction

$$\boldsymbol{\alpha}_{i,j,k} = \zeta_k \boldsymbol{\eta}_{i,j}$$

on the coefficients in the 3-dimensional coefficient array $\boldsymbol{\alpha}$ when solving the subproblem pertaining to the stimulus component. We note that the resulting reduced rank subproblem may then be solved with a modified version of the gd-pg algorithm where (again) a block relaxation algorithm is incorporated to obtain the desired factorization of the parameter array.

Applying this block relaxation approach we may use the existing software in the `glamlasso` R package (see [21]) to obtain the estimates for the parameter blocks $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ respectively. In Appendix C we give additional details on how to construct the algorithm, which is implemented in the R package `dynamo` available on CRAN, see [20].

5. Discussion

The VSD imaging data analyzed in this paper exemplifies noisy spatio-temporal array data with a potentially complicated spatio-temporal dependence structure. Our proposed methodology entails modeling this type of data as a non-stationary stochastic process (random field) in order to obtain a viable statistical model. We note that one immediate

challenge inherent in this methodology is to find a sensible decomposition of the drift into a stimulus component and a propagation component such that the statistical model can yield meaningful estimates of these components. Two related issues turned out to be of particular importance: first, the resulting fitted dynamical model should be stable; and second, both the fitted stimulus component and the fitted propagation component should be non-zero. The reduced rank procedure we used for fitting the stimulus component specifically addressed the latter issue while the sparsity inducing penalty addressed the former.

Next we showed how to exploit the particular array-tensor structure inherent in this specific data-model combination, to obtain a computationally tractable estimation procedure. The computational challenge was primarily addressed via the discretization schemes and basis expansions which resulted in a linear array model, that can be fitted using the algorithm proposed in [23]. This allowed us to obtain a design matrix free procedure with a minimal memory footprint that utilizes the highly efficient array arithmetic, see [6,13] and [10]. Consequently we were able to fit the model to VSD imaging data for a single trial (film) with 625 pixels in the spatial dimension and recorded over thousands of time points while modeling very long time delays, on a standard laptop computer in a matter of minutes. Given the results in [23] we expect our procedure to scale well with the size of the data.

In conclusion we highlight that both in terms of interpretability and computability the methodology developed in this paper compared to existing methods is particularly attractive for larger-scale applications. For instance we note that fitting data on this scale is computationally prohibitive using conventional time series techniques. An unrestricted vector autoregression (VAR) has a parameter dimension that grows quadratically in the size of the spatial dimension, see [15], and would be difficult to interpret and suffer from large variances on the parameter estimates. See, e.g., [35] for a VAR model applied to fMRI brain image or the approach in [12] to high dimensional VAR analysis, both considering data with size on a much smaller scale than the VSD imaging data. Specifically, fitting a traditional VAR type model to the data and setup considered here would result in a model with $LD^2 = 19,531,250$ parameters. In comparison our proposed estimation framework only uses 46,848 parameters of which only around 1,800 are non-zero in the aggregated estimate (model no. 6, animal 308). Within our methodological framework, this dimension reduction is obtained by modeling the array data as a random field, which makes it possible to represent the drift components in terms of smooth functions that in turn are easier to interpret compared to millions of single parameter estimates. In addition, by using a combination of local basis functions and sparsity inducing penalties we achieve even more regularization without restricting the model to a narrow parametric class.

Acknowledgment

The research was supported by VILLUM FONDEN via research grant 13358.

Appendix A. Proofs

Proof of Proposition 4.1. With w the network function let $\tilde{w}$ denote a $D \times D$ matrix-valued signed measure on $[-\tau, 0]$ with density $\tilde{F} : \mathbb{R} \to \mathbb{R}^{D\times D}$,

$$\tilde{F}(t) := \begin{pmatrix} \int_{\mathcal{S}_{1,1}} w(x, y, x_i, y_j, t)\mathrm{d}x\mathrm{d}y & \dots & \int_{\mathcal{S}_{N_x,N_y}} w(x, y, x_i, y_j, t)\mathrm{d}x\mathrm{d}y \\ \vdots & \ddots & \vdots \\ \int_{\mathcal{S}_{1,1}} w(x, y, x_{N_x}, y_{N_y}, t)\mathrm{d}x\mathrm{d}y & \dots & \int_{\mathcal{S}_{N_x,N_y}} w(x, y, x_{N_x}, y_{N_y}, t)\mathrm{d}x\mathrm{d}y \end{pmatrix},$$

with respect to the Lebesgue measure on $\mathbb{R}$. Following [5] we consider a stochastic delay differential equation for a D-dimensional (vector) process $\tilde{V}$ given by

$$\mathrm{d}\tilde{V}(t) = \left[\tilde{\mathrm{S}}(t) + \int_{-\tau}^{0} \tilde{w}(\mathrm{d}r)\tilde{V}(t + r) + \tilde{\mathrm{H}}\{\tilde{V}(t)\}\right]\mathrm{d}t + \tilde{C}\mathrm{d}\tilde{W}(t) \tag{A.1}$$

$$\tilde{V}(0) \in \mathbb{R}^D, \quad \tilde{V}(u) = \tilde{V}_0(u) \quad u \in (-\tau, 0), \tag{A.2}$$

where $\tilde{V}(0)$ and $\tilde{V}_0 \in C([-\tau, 0], \mathbb{R}^D)$ are initial conditions. A solution to (A.1) and (A.2) is then given by the integral equation (11) giving the coordinate-wise evolution in the space discretized model for the random field $\boldsymbol{V}$.

The sequence of $D \times D$ weight matrices $(\boldsymbol{w}_\ell)_\ell$ is defined by

$$\boldsymbol{w}_\ell := \int_{-\tau}^{0} \mathbf{1}_{[t_\ell, t_{\ell+1})}(s)\mathrm{d}\tilde{w}(s) = \int_{-\tau}^{0} \mathbf{1}_{[t_\ell, t_{\ell+1})}(s)\tilde{F}(s)\mathrm{d}s,$$

where $\mathbf{1}_{[t_\ell, t_{\ell+1})}$ is equal to I_D on $[t_\ell, t_{\ell+1})$ and zero otherwise. Especially the entry in the $[m, n]$th row and $[i, j]$th column of $\boldsymbol{w}_\ell$ is given as

$$\boldsymbol{w}_{[m,n],[i,j],\ell} = \int_{t_k}^{t_{k+1}} \int_{\mathcal{S}_{m,n}} w(x, y, x_i, y_j, s)\mathrm{d}x\mathrm{d}y\mathrm{d}s, \tag{A.3}$$

and we note that $\boldsymbol{w}$ is a $D\times D\times L$ array. Letting $(\epsilon_k)_k$ denote a sequence of i.i.d. $\mathcal{N}(0, I_D)$ variables the Euler scheme from [5] is now given by the D-dimensional discrete time (vector) process $\tilde{\tilde{V}} = (\tilde{\tilde{V}}_k)_k$ solving the stochastic difference equation (12) for $k \in \{0, \ldots, M-1\}$ with initial conditions $\tilde{\tilde{V}}_\ell = \tilde{V}_0(t_\ell)$ for $\ell \in \{-L, \ldots, -1\}$ and $\tilde{\tilde{V}}_0 = \tilde{V}(0)$. Thus by Theorem 1.2 in [5] and using that the deterministic function s is continuous, for $\Delta_t \downarrow 0$, $\tilde{\tilde{V}}$ defined in (12) converges weakly to the (vector) process solving SDDE (A.1) and (A.2), i.e., the vector process $\tilde{V}$ with evolution identical to that of $\tilde{\boldsymbol{V}}$ given by (11). □

Note $\tilde{\tilde{V}}$ is simply the vectorized version of $\tilde{\tilde{\boldsymbol{V}}}$, that is $\tilde{\tilde{V}}_{[i,j],k} = \tilde{\tilde{\boldsymbol{V}}}_{i,j,k}$.

Proof of Proposition 4.2. First using the expansion in (14) we can write the entries in each weight matrix $\boldsymbol{w}_\ell$ from (A.3) as

$$\boldsymbol{w}_{[m,n],[i,j],\ell} = \sum_q \beta_q \int_{t_\ell}^{t_{\ell+1}} \int_{\mathcal{S}_{m,n}} \phi_q(x, y, x_i, y_j, s)\mathrm{d}x\mathrm{d}y\mathrm{d}s.$$

Then we can write the $[m, n]$th coordinate of the vector process from (12) as

$$\begin{aligned}\tilde{\tilde{V}}_{[m,n],k+1} &= \left\{\tilde{s}(x_m, y_n, t_k) + \sum_{\ell=-L}^{-1}\sum_{i,j} \boldsymbol{w}_{[m,n],[i,j],\ell}\tilde{\tilde{V}}_{[i,j],k+\ell} + (1+\tilde{h}(x_m, y_n))\tilde{\tilde{V}}_{[m,n],k}\right\}\Delta_t + \tilde{C}_{[m,n]}\sqrt{\Delta_t}\epsilon_k \\ &= \Delta_t \sum_q \alpha_q\phi_q(x_m, y_n, t_k) + \beta_q \mathbf{F}_{[m,n],k,q} + \gamma_q\phi_q(x_m, y_n)\tilde{\tilde{V}}_{[m,n],k} + \tilde{C}_{[m,n]}\sqrt{\Delta_t}\epsilon_k \end{aligned} \tag{A.4}$$

where

$$\mathbf{F}_{[m,n],k,q} := \sum_{l=-L}^{-1}\sum_{i,j} \tilde{\tilde{V}}_{[i,j],k+l} \int_{t_l}^{t_{l+1}} \int_{\mathcal{S}_{m,n}} \phi_q(x, y, x_i, y_j, r)\mathrm{d}x\mathrm{d}y\mathrm{d}r \tag{A.5}$$

for $q \in \{1, \ldots, p_w\}$. Then letting $\theta := (\alpha, \beta, \gamma)^\top$, $y_k := \tilde{\tilde{V}}_{k+1}$, $e_k := \tilde{C}_{[m,n]}\sqrt{\Delta_t}\epsilon_k$, and $X_k := (S_k \mid F_k \mid H_k)$ with

$$\begin{aligned} S_k &:= \begin{pmatrix} \phi_1(x_1, y_1, t_k) & \cdots & \phi_{p_s}(x_1, y_1, t_k) \\ \vdots & \vdots & \vdots \\ \phi_1(x_{N_x}, y_{N_y}, t_k) & \cdots & \phi_{p_s}(x_{N_x}, y_{N_y}, t_k) \end{pmatrix}, \quad F_k := \begin{pmatrix} \mathbf{F}_{[1,1],k} \\ \vdots \\ \mathbf{F}_{[N_x,N_y],k} \end{pmatrix}, \\ H_k &:= \begin{pmatrix} \phi_1(x_1, y_1)\tilde{\tilde{V}}_{[1,1],k} & \cdots & \phi_{p_h}(x_1, y_1)\tilde{\tilde{V}}_{[1,1],k} \\ \vdots & \vdots & \vdots \\ \phi_1(x_{N_x}, y_{N_y})\tilde{\tilde{V}}_{[N_x,N_y],k} & \cdots & \phi_{p_h}(x_{N_x}, y_{N_y})\tilde{\tilde{V}}_{[N_x,N_y],k} \end{pmatrix}. \end{aligned} \tag{A.6}$$

the model equation (16) follows from (A.4).

To obtain the likelihood (18) note that the transition density for the model is

$$f(y_k \mid X_k) = (\sqrt{2\pi})^{-D}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(y_k - X_k\theta)^\top\Sigma^{-1}(y_k - X_k\theta)\right\}.$$

As $y_k \mid X_k$ is independent of $y_0, \ldots, y_{k-1}, X_0, \ldots, X_{k-1}$ we get by successive conditioning that we can write the joint conditional density as

$$f(y_0, \ldots, y_{M-1} \mid X_0, \ldots, X_{M-1}) = (\sqrt{2\pi})^{-MD}|\Sigma|^{-M/2}\exp\left\{-\frac{1}{2}\sum_{k=0}^{M-1}(y_k - X_k\theta)^\top\Sigma^{-1}(y_k - X_k\theta)\right\}.$$

Taking $-\ln$ yields the negative log-likelihood

$$\ell_X(\theta, \Sigma) := \frac{M}{2}\ln|\Sigma| + \frac{1}{2}\sum_{k=0}^{M-1}(y_k - X_k\theta)^\top\Sigma^{-1}(y_k - X_k\theta).$$

With X and y as in (17) it follows that

$$\left(I_M \otimes \Sigma^{-1/2}\right)(y - X\theta) = \begin{pmatrix} \Sigma^{-1/2}(y_0 - X_0\theta) \\ \vdots \\ \Sigma^{-1/2}(y_{M-1} - X_{M-1}\theta) \end{pmatrix} = \begin{pmatrix} \sum_j^D \Sigma_{1,j}^{-1/2}(y_0 - X_0\theta)_j \\ \vdots \\ \sum_j^D \Sigma_{D,j}^{-1/2}(y_0 - X_0\theta)_j \\ \vdots \\ \sum_j^D \Sigma_{D,j}^{-1/2}(y_{M-1} - X_{M-1}\theta)_j \end{pmatrix}.$$

Hence,

$$\left\| \left(I_M \otimes \Sigma^{-1/2}\right)(y - X\theta) \right\|_2^2 = \sum_{k=0}^{M-1} \sum_i^D \left\{ \sum_j^D \Sigma_{i,j}^{-1/2}(y_k - X_k\theta)_j \right\}^2 = \sum_{k=0}^{M-1} \| \Sigma^{-1/2}(y_k - X_k\theta) \|_2^2 = \sum_{k=0}^{M-1} (y_k - X_k\theta)^\top \Sigma^{-1}(y_k - X_k\theta)$$

yielding the expression for the negative log-likelihood given in (18). □

Proof of Proposition 4.3. Noting that

$$(S \mid F \mid H) := \begin{pmatrix} S_1 & F_1 & H_1 \\ \vdots & \vdots & \vdots \\ S_M & F_M & H_M \end{pmatrix} = X$$

the claim follows if we show that S and F are appropriate 3-tensor matrices and that $H\gamma = \mathrm{vec}(\tilde{\tilde{\boldsymbol{V}}}_{-1} \odot \boldsymbol{C})$ for an appropriately defined array $\boldsymbol{C}$.

Letting $\phi^x := \{\phi_q(x_i)\}_{i,q}$ denote a $N_x \times p_x$ matrix with p_x basis functions evaluated at N_x points in the x domain it follows directly from the definition of the tensor product that we can write $S = \phi^x \otimes \phi^y \otimes \phi^t$.

Next let $\Phi^x := \{\int_{\mathcal{X}_i} \phi_q(x)\}_{i,q}$ denote the integrated version of ϕ^x. Then inserting the tensor basis functions in to (A.5) we can write

$$\begin{aligned} \mathbf{F}_{[m,n],k,[q_1,q_2,q_3,q_4,q_5]} &= \sum_{\ell=-L}^{-1} \sum_{i,j} \tilde{\tilde{V}}_{[i,j],k+\ell} \int_{t_l}^{t_{\ell+1}} \int_{\mathcal{S}_{m,n}} \phi_{q_1}(x)\phi_{q_2}(y)\phi_{q_3}(x_i)\phi_{q_4}(y_j)\phi_{q_5}(s)\mathrm{d}x\mathrm{d}y\mathrm{d}s \\ &= \int_{\mathcal{X}_m} \phi_{q_1}(x)\mathrm{d}x \int_{\mathcal{Y}_n} \phi_{q_2}(y)\mathrm{d}y \sum_{\ell=-L}^{-1} \sum_{i,j} \tilde{\tilde{V}}_{[i,j],k+\ell} \phi_{q_3}(x_i)\phi_{q_4}(y_j) \int_{t_\ell}^{t_{\ell+1}} \phi_{q_5}(s)\mathrm{d}s. \end{aligned}$$

Letting $\Phi^{xy\ell}$ denote a $M \times p_x p_y p_\ell$ matrix with entries

$$\Phi^{xy\ell}_{k,[q_3,q_4,q_5]} := \sum_{\ell=-L}^{-1} \sum_{i,j} \tilde{\tilde{V}}_{[i,j],k+\ell} \phi_{q_3}(x_i)\phi_{q_4}(y_j) \int_{t_\ell}^{t_{\ell+1}} \phi_{q_5}(s)\mathrm{d}s \tag{A.7}$$

we can write $F = \Phi^{xyt} \otimes \Phi^y \otimes \Phi^x$.

Finally let $\boldsymbol{C}$ be a $N_x \times N_y \times M$ array such that $\boldsymbol{C}_k = \phi^y \boldsymbol{\gamma} (\phi^x)^\top$. With H_k given in (A.6) we can write

$$H_k\gamma = \begin{pmatrix} \tilde{\tilde{V}}_{[1,1],k} \sum_{q_1,q_2} \phi^x_{q_1}(x_1)\phi^y_{q_2}(y_1)\gamma_{[q_1,q_2]} \\ \vdots \\ \tilde{\tilde{V}}_{[N_x,N_y],k} \sum_{q_1,q_2} \phi^x_{q_1}(x_{N_x})\phi^y_{q_2}(y_{N_y})\gamma_{[q_1,q_2]} \end{pmatrix} = \tilde{\tilde{V}}_k \odot \phi^x \otimes \phi^y \gamma = \mathrm{vec}(\tilde{\tilde{\boldsymbol{V}}}_k \odot \boldsymbol{C}_k)$$

showing that $H\gamma = \mathrm{vec}(\tilde{\tilde{\boldsymbol{V}}}_{-1} \odot \boldsymbol{C})$. □

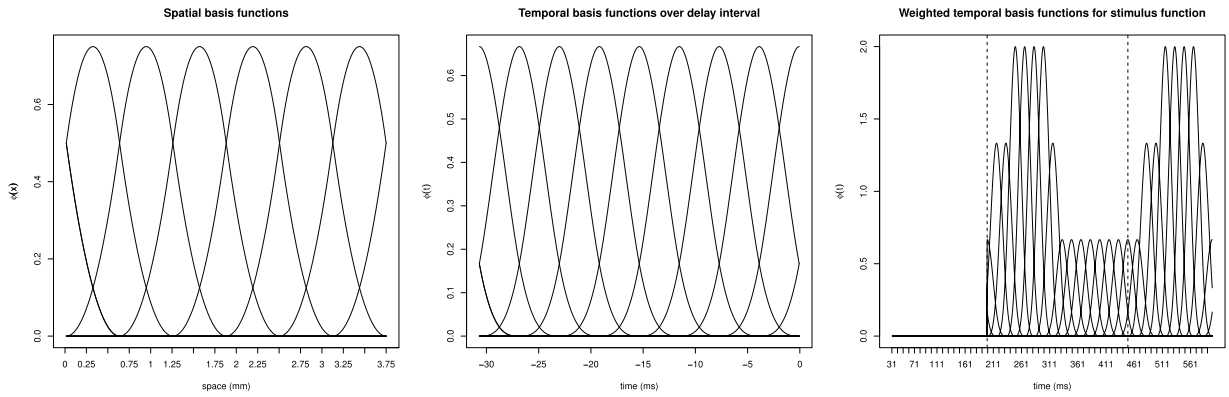


Fig. C.6. Spatial basis splines (left) and temporal basis splines (middle and right).

Appendix B. Tensor product basis

Consider a d-variate function $f \in L^2(\mathbb{R}^d)$ that we want to represent using some basis expansion. Instead of directly specifying a basis for $L^2(\mathbb{R}^d)$ we will specify d marginal sets of univariate functions

$$(\phi_{1,k_1})_{k_1=1}^{\infty}, \ldots, (\phi_{d,k_d})_{k_d=1}^{\infty} \tag{B.1}$$

with each marginal set a basis for $L^2(\mathbb{R})$. Then for any $(k_1, \ldots, k_d) \in \mathbb{N}^d$ we may define a d-variate function $\pi_{k_1,\ldots,k_d} : \mathbb{R}^d \to \mathbb{R}$ via the product of the marginal functions, i.e.,

$$(x_1, \ldots, x_d) \mapsto \pi_{k_1,\ldots,k_d}(x_1, \ldots, x_d) := \prod_{i=1}^{d} \phi_{i,k_i}(x_i). \tag{B.2}$$

Since each marginal set of functions in (B.1) constitutes a basis of $L^2(\mathbb{R})$ it follows using Fubini's theorem, that the induced set of d-variate functions $(\pi_{k_1,\ldots,k_d})_{k_1,\ldots,k_d}$ constitutes a basis of $L^2(\mathbb{R}^d)$. Especially again by Fubini's theorem we note that if the functions in (B.1) all are orthonormal marginal bases then they generate an orthonormal basis of $L^2(\mathbb{R}^d)$. Finally, if the marginal functions have compact support then the induced d-variate set of functions will also have compact support.

Appendix C. Implementation details

Here we elaborate a little on various details pertaining to the implementation of the inferential procedure for the specific data considered in Section 3 and on some more general details relating to the computations.

C.1. Regression setup

For animals in the data set we use the first 600 ms of the recording corresponding to 977 images. Thus for each trial of VSD data we have a total of 250112 observations arranged in a $25 \times 25 \times 977$ grid. We model a delay of $L := 50$ images corresponding to $\tau \approx 31$ ms which gives us $M := 926$ modeled time points.

As explained above we can use tensor basis functions to represent the component functions. For the analysis of the brain image data we will use B-splines as basis functions in each dimension as these have compact support. Specifically in the spatial dimensions we will use quadratic B-splines while in the temporal dimension we will use cubic B-splines, see Fig. C.6. Note that for the temporal factor of the stimulus function we choose basis function covering the stimulus interval and post stimulus interval.

The number of basis functions in each spatial dimension is $p_x = p_y := 8$ and in the temporal dimension we have $p_\ell := 11$ basis functions to capture the delay and $p_t := 27$ temporal basis functions to model the stimulus, see Fig. C.6. With this setup we have a total of $p := p_s + p_w + p_h = p_x p_y p_t + p_x p_y p_x p_y p_\ell + p_x p_y = 46848$ model parameters that need to be estimated.

For the lasso regression we give all parameters the same weight 1, except for the basis function representing the temporal component of the stimulus. In particular, basis functions located right after stimulus start and stop are penalized less than the rest of the basis function. Fig. C.6 (right) shows the basis function weighted with the inverse of the parameter weights. With this weighing scheme the stimulus component will be focused on picking up the direct stimulus effect while it will be less likely to pick up propagation effects.

We fit the model for a sequence of penalty parameters $\lambda_{max} := \lambda_1 > \cdots > \lambda_K =: \lambda_{min}$ with $K := 10$. Here given data, λ_{max} is the smallest value yielding a zero solution to (20). The results presented in the text are from model number 6. See the supplementary material, Section 1, for more on model selection.

C.2. Algorithmic details

We will here sketch how to solve the subproblem (20) using a generalized block relaxation algorithm and the R software package `glamlasso`, see [21]. The entire method, which relies on Algorithm 1, is available from CRAN via the R package `dynamo`, see [20].

Algorithm 1 Block relaxation algorithm

Require: $\{\boldsymbol{\alpha}^{(0)}, \boldsymbol{\beta}^{(0)}, \boldsymbol{\gamma}^{(0)}\}$, $\boldsymbol{\Phi}^{xy\ell}$, $\boldsymbol{\Phi}^{t}$, $\boldsymbol{\Phi}^{x}$, $\boldsymbol{\Phi}^{y}$, $\hat{\Sigma}$
  while $a < N \in \mathbb{N}$ **do**
    $a = a + 1$
    for $b = 1$ to 3 **do**
      if $b = 1$ **then**
        compute $\hat{Y}_{-\boldsymbol{\alpha}}$ using $\hat{\boldsymbol{\beta}}^{(a)}$, $\hat{\boldsymbol{\gamma}}^{(a)}$ and solve the reduced rank problem

$$\hat{\boldsymbol{\alpha}}^{(a+1)} = \arg\min_{\boldsymbol{\alpha}} O_\lambda(\boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}^{(a)}, \hat{\boldsymbol{\gamma}}^{(a)}) \quad \text{s.t.} \quad \boldsymbol{\alpha}_{i,j,k} = \zeta_k \eta_{i,j} \tag{C.1}$$

        using data $\hat{Y}_{-\boldsymbol{\alpha}}$, $\boldsymbol{\Phi}^{x}$, $\boldsymbol{\Phi}^{y}$, $\boldsymbol{\Phi}^{t}$, $\hat{\Sigma}$.
      else if $b = 2$ **then**
        compute $\hat{Y}_{-\boldsymbol{\beta}}$ using $\hat{\boldsymbol{\alpha}}^{(a+1)}$, $\hat{\boldsymbol{\gamma}}^{(a)}$, and solve

$$\hat{\boldsymbol{\beta}}^{(a+1)} = \arg\min_{\boldsymbol{\beta}} O_\lambda(\hat{\boldsymbol{\alpha}}^{(a+1)}, \boldsymbol{\beta}, \hat{\boldsymbol{\gamma}}^{(a)}) \tag{C.2}$$

        using data $\hat{Y}_{-\boldsymbol{\beta}}$, $\boldsymbol{\Phi}^{x}$, $\boldsymbol{\Phi}^{y}$, $\boldsymbol{\Phi}^{xy\ell}$, $\hat{\Sigma}$.
      else if $b = 3$ **then**
        compute $\hat{Y}_{-\boldsymbol{\gamma}}$ using $\hat{\boldsymbol{\alpha}}^{(a+1)}$, $\hat{\boldsymbol{\beta}}^{(a+1)}$ and solve

$$\hat{\boldsymbol{\gamma}}^{(a+1)} = \arg\min_{\boldsymbol{\gamma}} O_\lambda(\hat{\boldsymbol{\alpha}}^{(a+1)}, \hat{\boldsymbol{\beta}}^{(a+1)}, \boldsymbol{\gamma}) \tag{C.3}$$

        using data $\hat{Y}_{-\boldsymbol{\gamma}}$, $\boldsymbol{\Phi}^{x}$, $\boldsymbol{\Phi}^{y}$, $\hat{\Sigma}$.
      end if
    end for
    if convergence criterion is satisfied **then**
      break
    end if
  end while

Algorithm 1 works by fixing different blocks of the parameter in an alternating fashion and then optimize over the non fixed blocks. Let $O_\lambda := \ell_X + \lambda J_1$, $\lambda > 0$ denote the objective function in (20) considered as a function of the parameter arrays (or blocks) $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\}$. Let $\hat{Y}_{-b}$ denote the partial residuals array obtained by setting parameter block $b \in \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\}$, to zero when computing the linear predictor in (25), for given estimates of the remaining parameter blocks, and then using this to compute the model residuals.

Each subproblem (C.1)–(C.2) in Algorithm 1 can be solved with a function from the `glamlasso` package. Especially (C.1) is solved with `glamlassoRR`, (C.2) is solved with `glamlasso` and (C.3) is solved with `glamlassoS`. These functions are based on the coordinate descent proximal gradient algorithm, see [23]. We note that the function `glamlassoRR`, performing the reduced rank regression in this tensor setup, also utilizes a block relaxation scheme to obtain the desired factorization of the parameter array $\boldsymbol{\alpha}$. Furthermore we note that when iterating the steps 1–3 in the approximate MCRE algorithm above we have to reweigh data with the estimated covariance matrix. This in turn leads to weighted estimation problems in each subproblem (C.1)–(C.2).

C.3. Computing the convolution tensor

We note that for the filter component, the tensor component $\boldsymbol{\Phi}^{xyt}$, as shown in the proof of Proposition 4.3, corresponds to a convolution of the random field. This component has to be computed upfront which in principle could be very time consuming. However considering (A.7) this computation can be carried out using array arithmetic. Especially we can write the convolution tensor as

$$\boldsymbol{\Phi}^{xyt} = \begin{pmatrix} \operatorname{vec}(\boldsymbol{\Phi}_1^{xy\ell}) \\ \vdots \\ \operatorname{vec}(\boldsymbol{\Phi}_M^{xy\ell}) \end{pmatrix},$$

where Φ_k^{xyt} for each k is a $p_x \times p_y \times p_\ell$ array which according to (A.7) can be computed using the array arithmetic as

$$\Phi_k^{xy\ell} := \rho[(\phi^\ell)^\top, \rho\{(\phi^y)^\top, \rho((\phi^x)^\top, (\tilde{\tilde{\boldsymbol{V}}}_\ell)_{\ell=k-L}^{k-1})\}].$$

Appendix D. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jmva.2019.104532.

References

[1] R.J. Adler, J.E. Taylor, Random Fields and Geometry, Springer Science & Business Media, 2009.
[2] P.C. Bressloff, Spatiotemporal dynamics of continuum neural fields, J. Phys. A 45 (3) (2012).
[3] P.C. Bressloff, M.A. Webber, Front propagation in stochastic neural fields, SIAM J. Appl. Dyn. Syst. 11 (2) (2012) 708–740.
[4] N. Brunel, Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons, J. Comput. Neurosci. 8 (3) (2000) 183–208.
[5] E. Buckwar, T. Shardlow, Weak approximation of stochastic differential delay equations, IMA J. Numer. Anal. 25 (1) (2005) 57–86.
[6] P.E. Buis, W.R. Dyksen, Efficient vector and parallel manipulation of tensor products, ACM Trans. Math. Softw. (TOMS) 22 (1) (1996) 18–23.
[7] S. Chemla, F. Chavane, Voltage-sensitive dye imaging: technique review and models, J. Physiol.-Paris 104 (1) (2010) 40–50.
[8] S. Coombes, P. beim Graben, R. Potthast, J. Wright (Eds.), Neural Fields, Springer, Heidelberg, 2014.
[9] S.G. Cox, Stochastic Differential Equations in Banach Spaces: Decoupling, Delay Equations, and Approximations in Space and Time (Ph.D. Thesis), 2012.
[10] I.D. Currie, M. Durban, P.H. Eilers, Generalized linear array models with applications to multidimensional smoothing, J. R. Stat. Soc. Ser. B Stat. Methodol. 68 (2) (2006) 259–280.
[11] G. Da Prato, J. Zabczyk, Stochastic Equations in Infinite Dimensions, Cambridge university press, 2014.
[12] R.A. Davis, P. Zang, T. Zheng, Sparse vector autoregressive modeling, J. Comput. Graph. Statist. 25 (4) (2016) 1077–1096.
[13] C. De Boor, Efficient computer manipulation of tensor products, ACM Trans. Math. Softw. (TOMS) 5 (2) (1979) 173–182.
[14] J. De Leeuw, Block-relaxation algorithms in statistics, in: Information Systems and Data Analysis, Springer, 1994, pp. 308–324.
[15] J. Fan, J. Lv, L. Qi, Sparse high dimensional models in economics, Annu. Rev. Econ. 3 (2011) 291.
[16] O. Faugeras, J. Inglis, Stochastic neural field equations: a rigorous footing, J. Math. Biol. 71 (2) (2015) 259–300.
[17] J. Friedman, T. Hastie, R. Tibshirani, Sparse inverse covariance estimation with the graphical lasso, Biostatistics 9 (3) (2008) 432–441.
[18] P. beim Graben, R. Potthast, Inverse problems in dynamic cognitive modeling, Chaos 19 (1) (2009).
[19] M.A. Harvey, S. Valentiniene, P.E. Roland, Cortical membrane potential dynamics and laminar firing during object motion, Front. Syst. Neurosci. 3 (2009) 7.
[20] A. Lund, Dynamo: Fit a Stochastic Dynamical Array Model to Array Data, 2018, URL https://CRAN.R-project.org/package=dynamo, R package version 1.0.
[21] A. Lund, Glamlasso: Penalization in Large Scale Generalized Linear Array Models, 2018, URL https://CRAN.R-project.org/package=glamlasso, R package version 3.0.
[22] A. Lund, N.R. Hansen, Supplement to "Sparse Network Estimation for Dynamical Spatio-temporal Array Models", 2018.
[23] A. Lund, M. Vincent, N.R. Hansen, Penalized estimation in large-scale generalized linear array models, J. Comput. Graph. Statist. 26 (3) (2017) 709–724.
[24] X. Mao, Stochastic Differential Equations and Applications, Elsevier, 2007.
[25] V. Markounikau, C. Igel, A. Grinvald, D. Jancke, A dynamic neural field model of mesoscopic cortical activity captured with voltage-sensitive dye imaging, PLoS Comput. Biol. 6 (9) (2010).
[26] S. Peszat, J. Zabczyk, Stochastic evolution equations with a spatially homogeneous wiener process, Stochastic Process. Appl. 72 (2) (1997) 187–204.
[27] D. Pinotsis, R. Moran, K. Friston, Dynamic causal modeling with neural fields, NeuroImage 59 (2) (2012) 1261–1274.
[28] R. Potthast, P. beim Graben, Inverse problems in neural field theory, SIAM J. Appl. Dyn. Syst. 8 (4) (2009) 1405–1433.
[29] P.E. Roland, A. Hanazawa, C. Undeman, D. Eriksson, T. Tompa, H. Nakamura, S. Valentiniene, B. Ahmed, Cortical feedback depolarization waves: A mechanism of top-down influence on early visual areas, Proc. Natl. Acad. Sci. 103 (33) (2006) 12586–12591.
[30] A.J. Rothman, E. Levina, J. Zhu, Sparse multivariate regression with covariance estimation, J. Comput. Graph. Statist. 19 (4) (2010) 947–962.
[31] A. Roxin, E. Montbri, How effective delays shape oscillatory dynamics in neuronal networks, Physica D 240 (3) (2011) 323–345.
[32] O. Sporns, D.R. Chialvo, M. Kaiser, C.C. Hilgetag, Organization, development and function of complex brain networks, Trends in Cogn. Sci. 8 (9) (2004) 418–425.
[33] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1) (1996) 267–288.
[34] J. Touboul, Propagation of chaos in neural fields, Ann. Appl. Probab. 24 (3) (2014) 1298–1328.
[35] P.A. Valdés-Sosa, J.M. Sánchez-Bornot, A. Lage-Castellanos, M. Vega-Hernández, J. Bosch-Bayard, L. Melie-García, E. Canales-Rodríguez, Estimating brain functional connectivity with sparse multivariate autoregression, Philos. Trans. R. Soc. B 360 (1457) (2005) 969–981.
[36] D. Xu, X. Wang, Z. Yang, Existence-uniqueness problems for infinite dimensional stochastic differential equations with delays, J. Appl. Anal. Comput. 2 (4) (2012) 449–463.

Supplement to: Sparse Network Estimation for Dynamical Spatio-temporal Array Models

Adam Lund[a,*], Niels Richard Hansen[a]

[a]*Københavns Universitet,*
Institut for Matematiske Fag,
Universitetsparken 5,
2100 København Ø, Danmark

Abstract

For model selection purpose we provide a cross validation study and a simulation study. We also provide results for the remaining 12 animals as well as single trial results for animal 308.

1. Choosing a penalty parameter

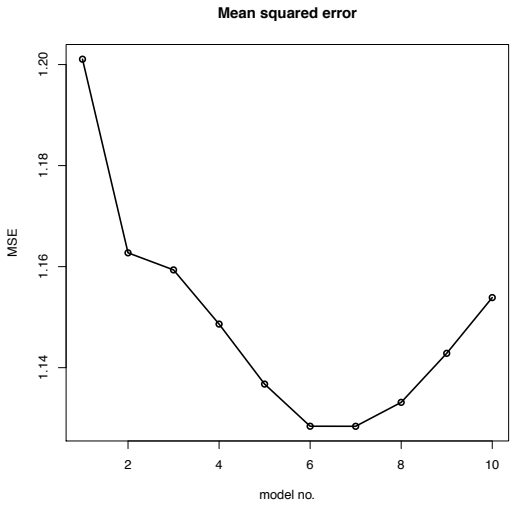


Fig. 1: MSE resulting from a 4-fold cross validation for animal 308.

For all animals we fit the model to data for 10 values of the penalty parameter, $\lambda_1 > \lambda_2 > \ldots > \lambda_{10}$. For animal 308 we presented, in Section 3, the trial aggregated fit for model no. 6 (λ_6). Here we shall briefly discuss how to evaluate the fit associated with a particular λ in order to select a model (λ).

For animal 308, after fitting each of the 12 trials to the model for 10 values of the penalty parameter λ we perform for each λ a type of a 4-fold cross validation. We do this by dividing the trials into 4 groups (3 trials in each) and then for each group aggregate the trial fits over the complement (9 trials). We then compute the resulting MSE obtained

*Corresponding author. Email address: adam.lund@math.ku.dk

when using this aggregated fit on the held out trials. In Figure 1 the average MSE over these groups is shown for each model. Model 6 yields the minimum MSE.

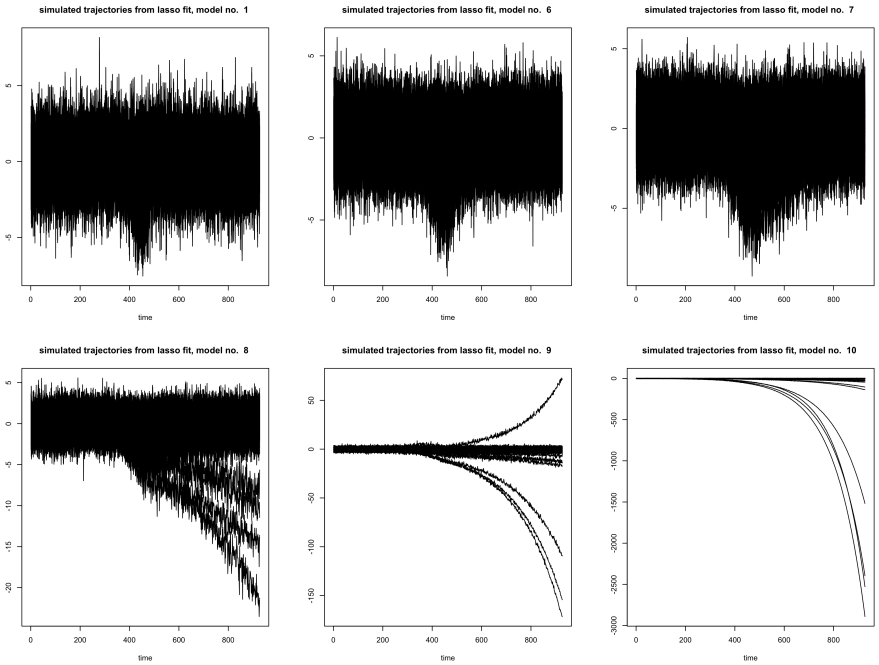


Fig. 2: For $i = 2, 6, 7, 8, 9, 10$ we plot all trajectories simulated with parameters given by the fit from trial 1, animal 308, when using penalty λ_i.

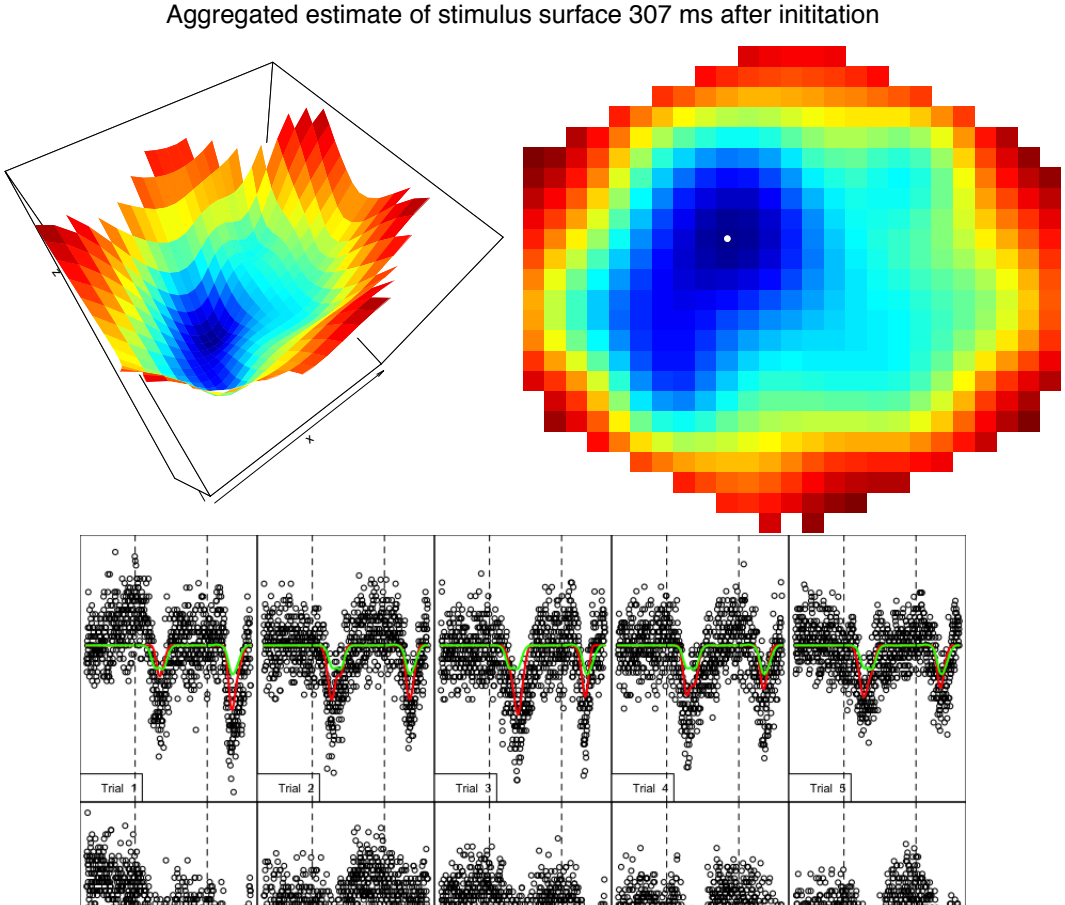
Another way to evaluate the fit for a given model no. is to perform a simulation study. For given λ we can use the parameter estimates for a single trial to simulate from the Euler scheme equation. This in turn gives us a simulated data set that should display characteristics similar to that of the specific trial. Figure 2 below shows all the simulated trajectories (i.e. the data for each pixel plotted over time for all pixels) based on the fit from trial 1, animal 308, for $\lambda_i, i \in \{2, 6, 7, 8, 9, 10\}$. We see that for models 8, 9 and 10 the simulations is increasingly unstable leading to explosive behavior of the simulated sample paths. The penalty dampens the aggregation in the filter component which causes the simulations from the model to stabilize.

2. Aggregated fit for each individual animal

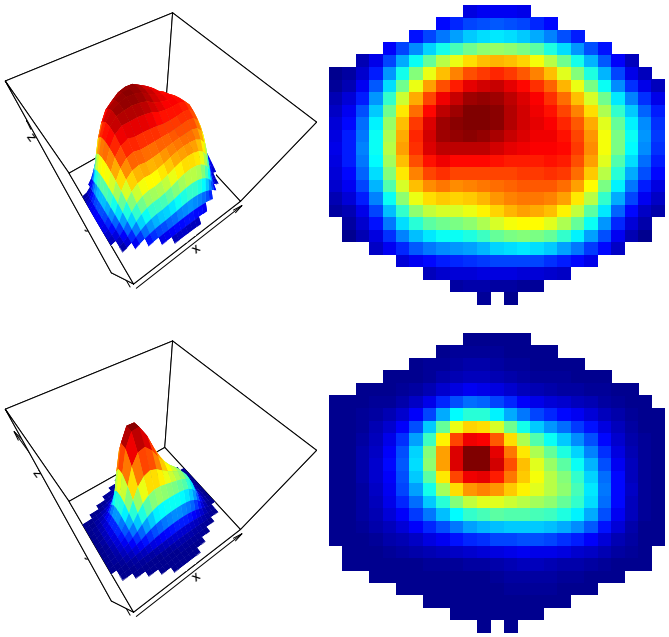
Next for each of the 12 remaining animals we present visualizations of the mean aggregated fit, corresponding to that presented for animal 308 in Section 3 (model no. 6). We note that the mean aggregation is carried out by averaging the trial parameter estimates $(\hat{\theta}_t)_{t=1}^T$ over the T trials, for each animal,

$$\hat{\theta}_{agg} = \frac{1}{T} \sum_{t=1}^{T} \hat{\theta}_t.$$

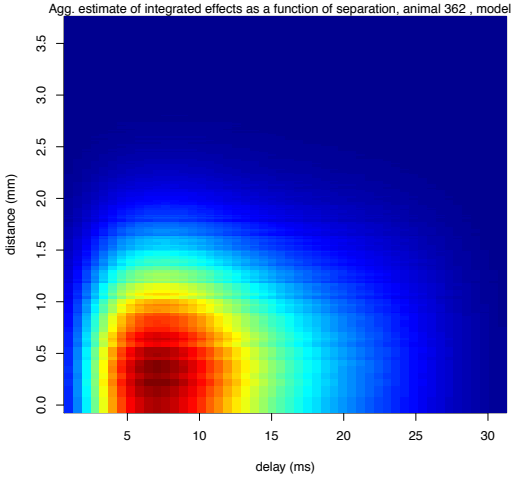
2.1. Animal 362



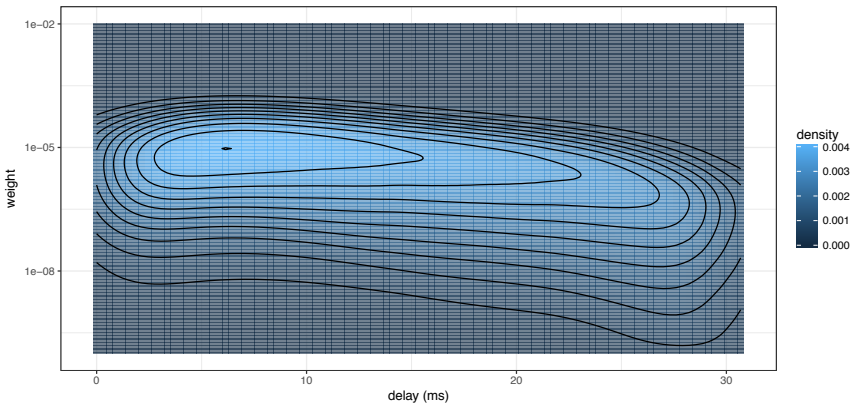
Mean aggregated estimate of integrated network function for model 6



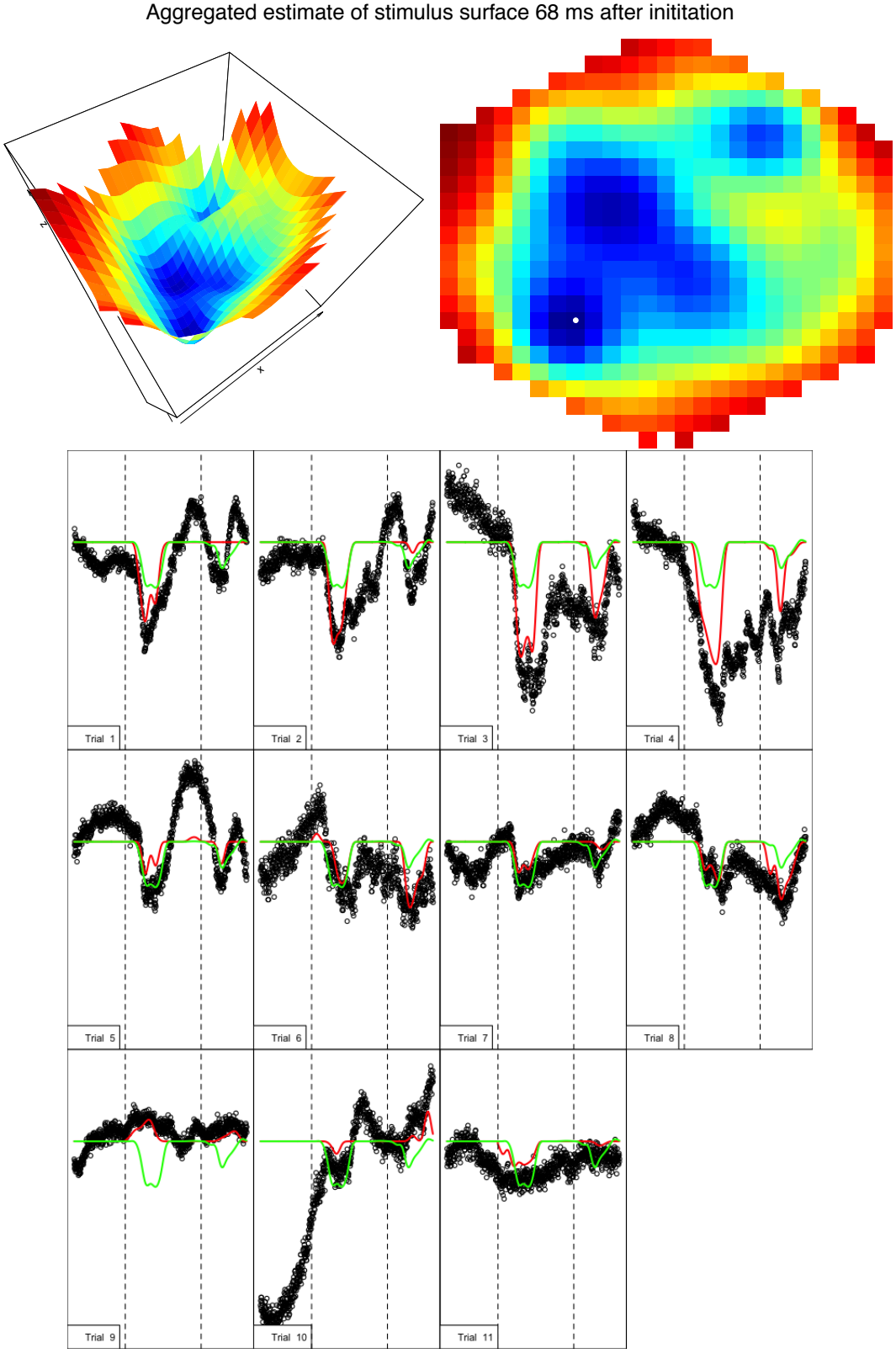
Agg. estimate of integrated effects as a function of separation, animal 362 , model 6



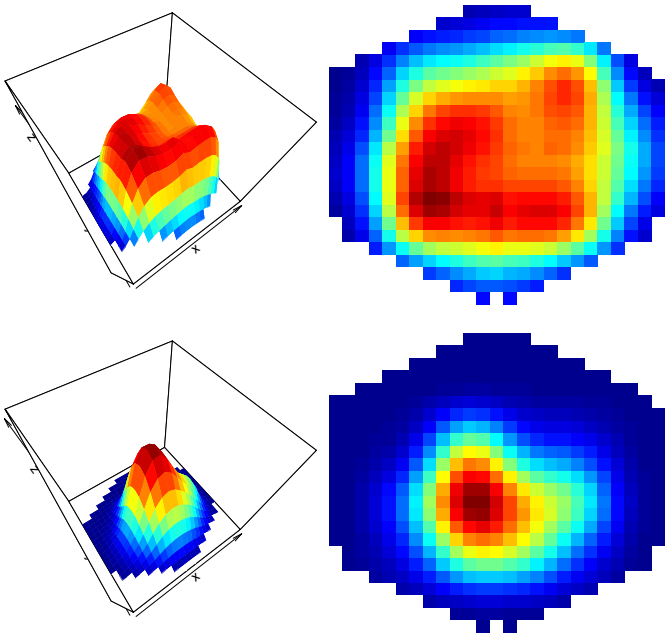
Density plot of agg. estimated weight values for animal 362 model 6



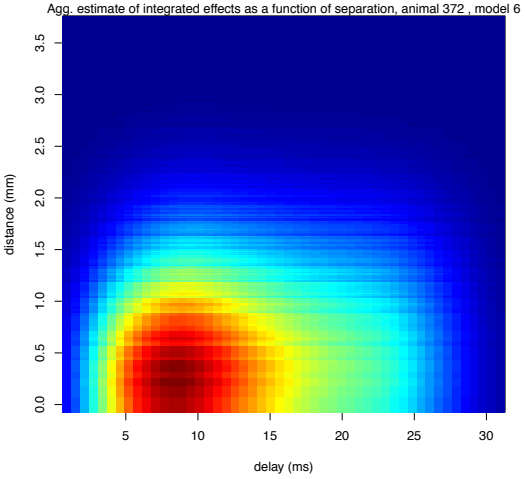
2.2. Animal 372



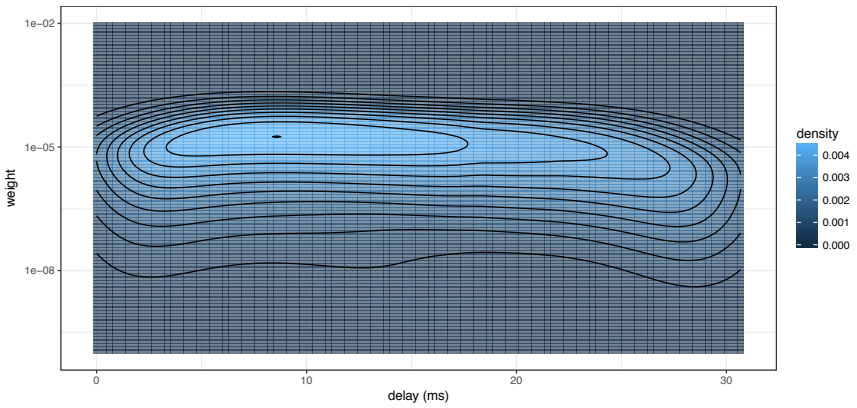
Mean aggregated estimate of integrated network function for model 6



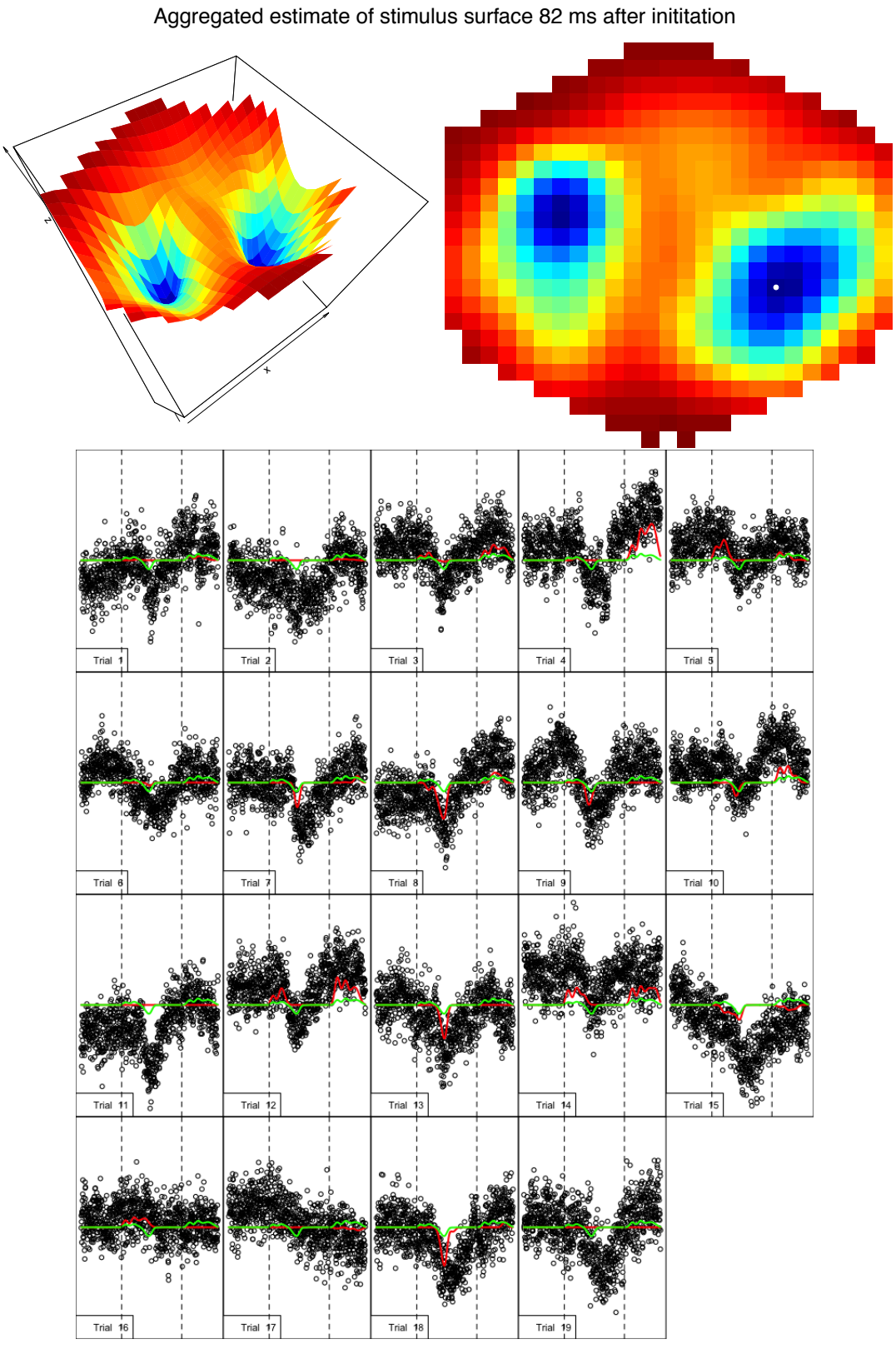
Agg. estimate of integrated effects as a function of separation, animal 372 , model 6



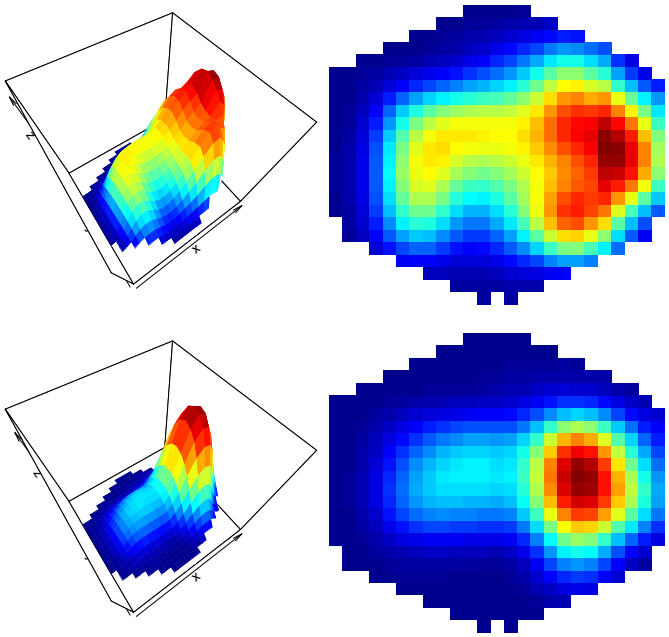
Density plot of agg. estimated weight values for animal 372 model 6



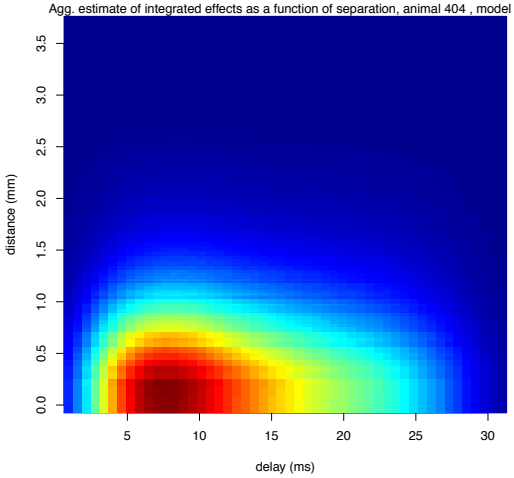
2.3. Animal 404



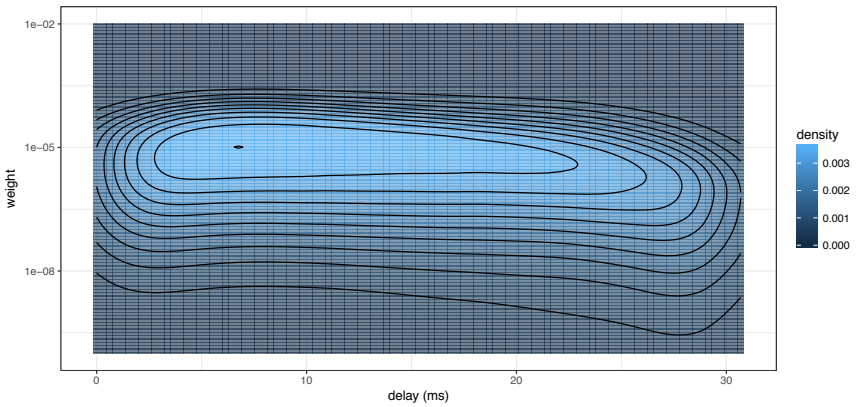
Mean aggregated estimate of integrated network function for model 6



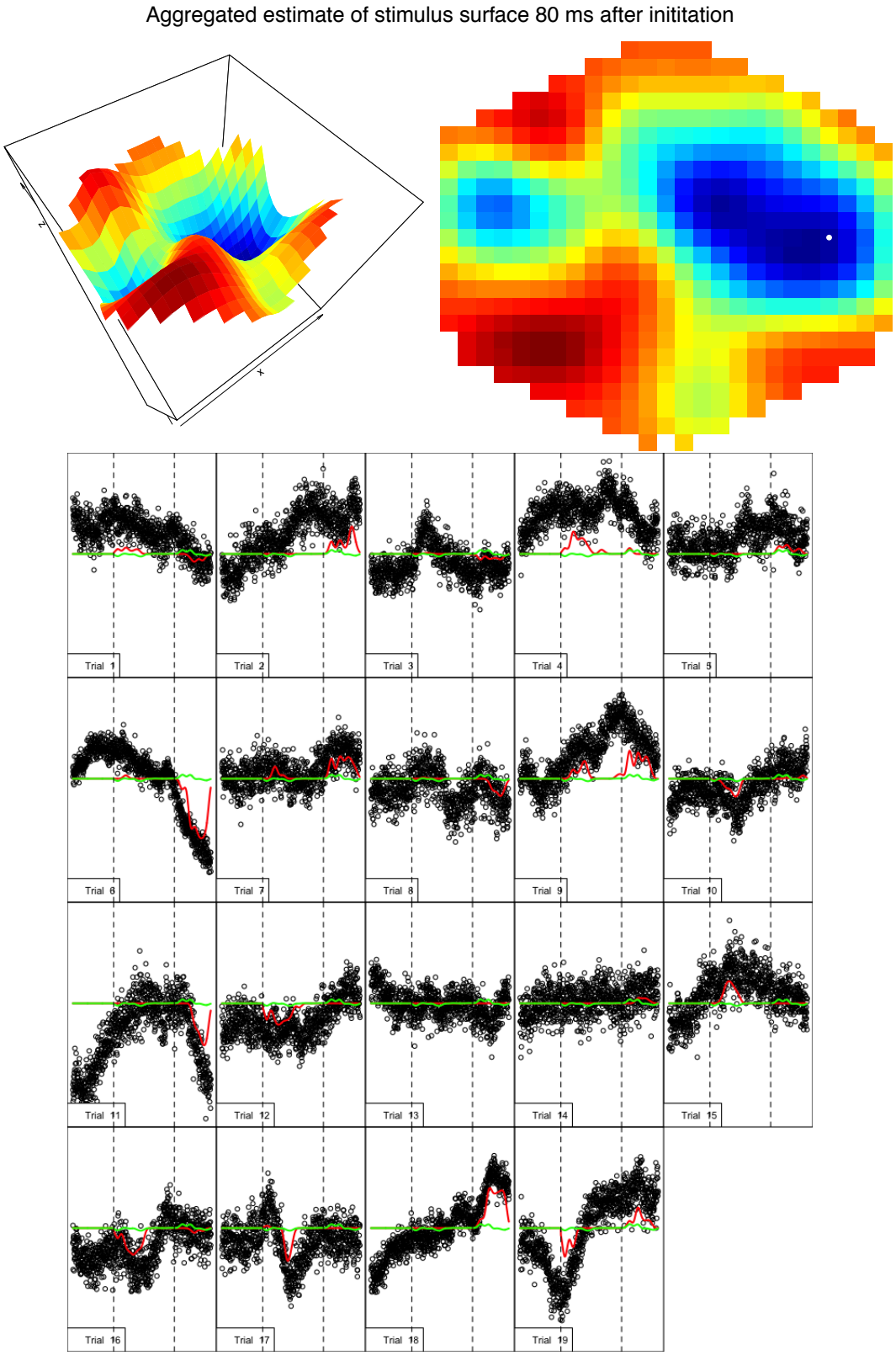
Agg. estimate of integrated effects as a function of separation, animal 404 , model 6



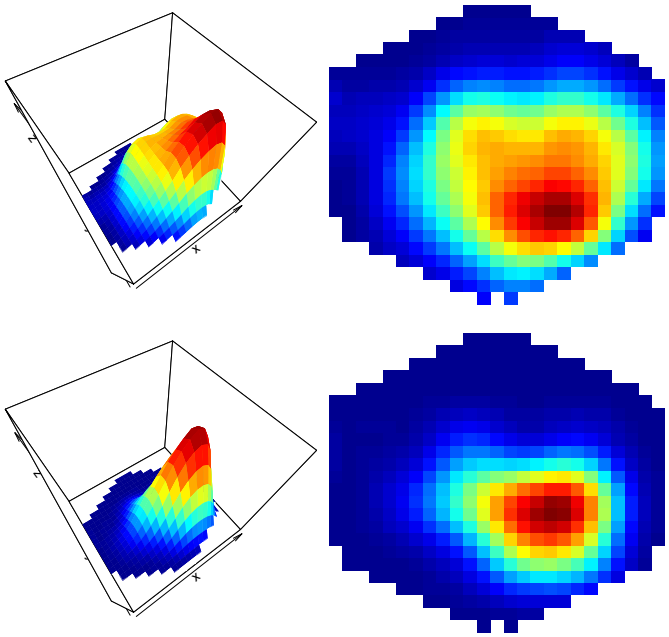
Density plot of agg. estimated weight values for animal 404 model 6



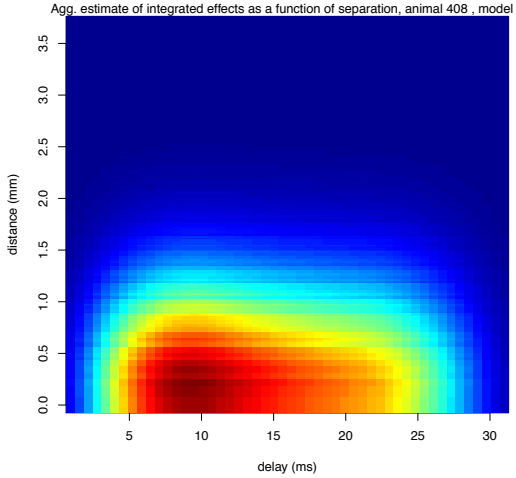
2.4. Animal 408



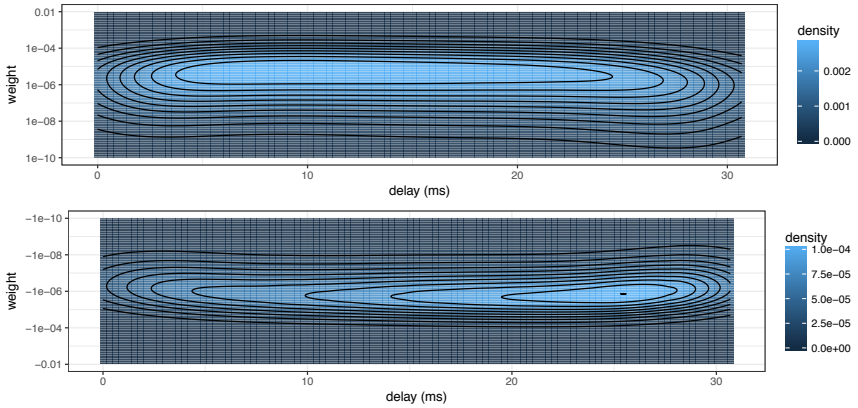
Mean aggregated estimate of integrated network function for model 6



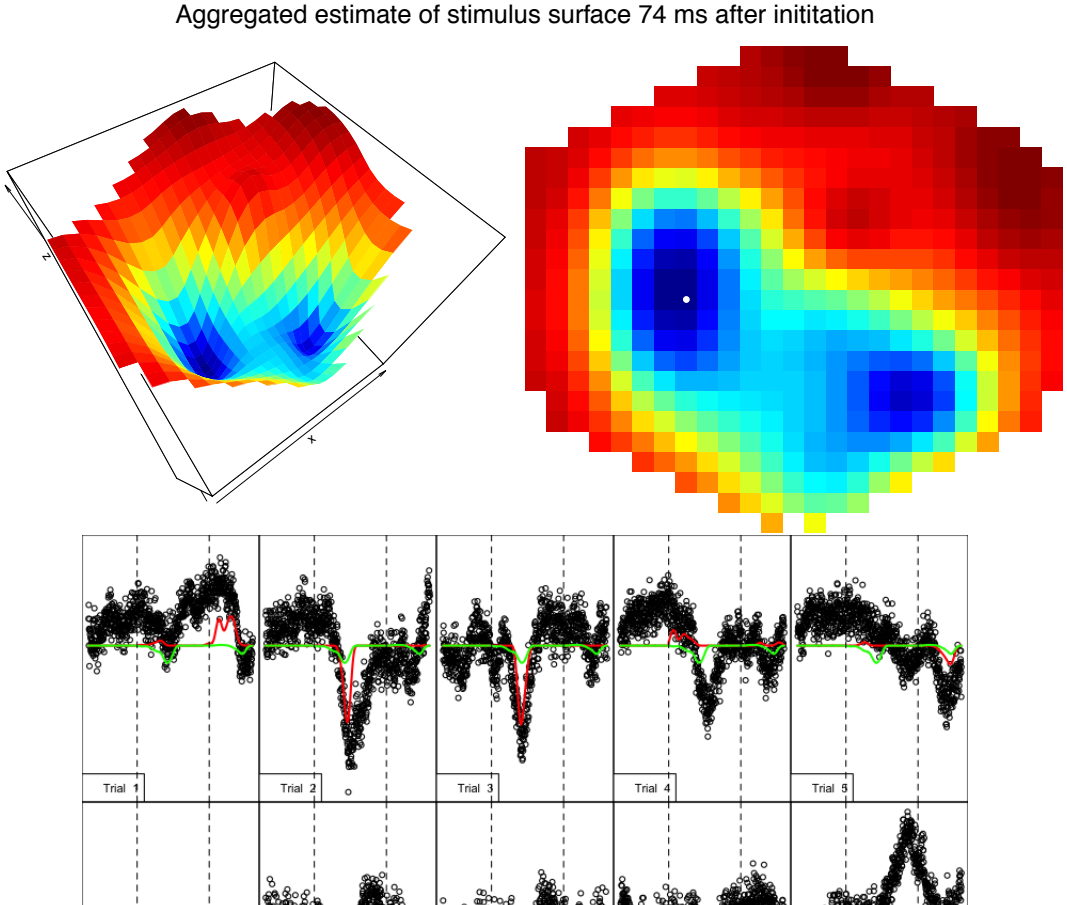
Agg. estimate of integrated effects as a function of separation, animal 408 , model 6



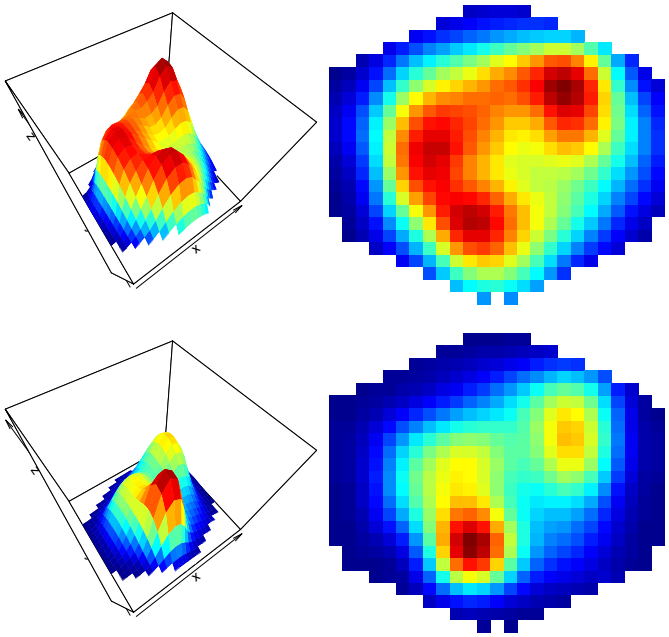
Density plot of aggregated estimates of weight values for animal 408 model 6



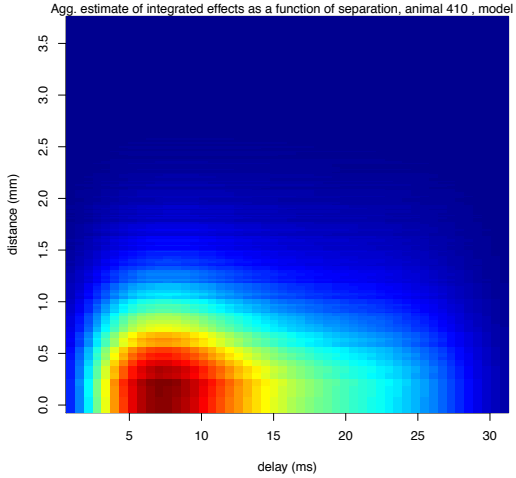
2.5. Animal 410



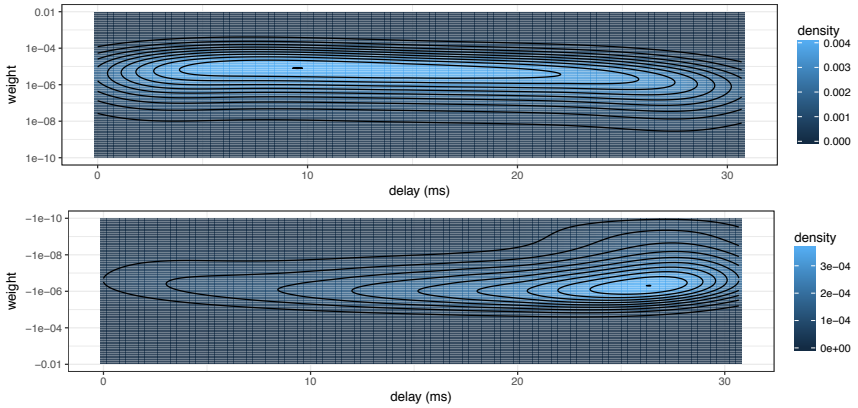
Mean aggregated estimate of integrated network function for model 6



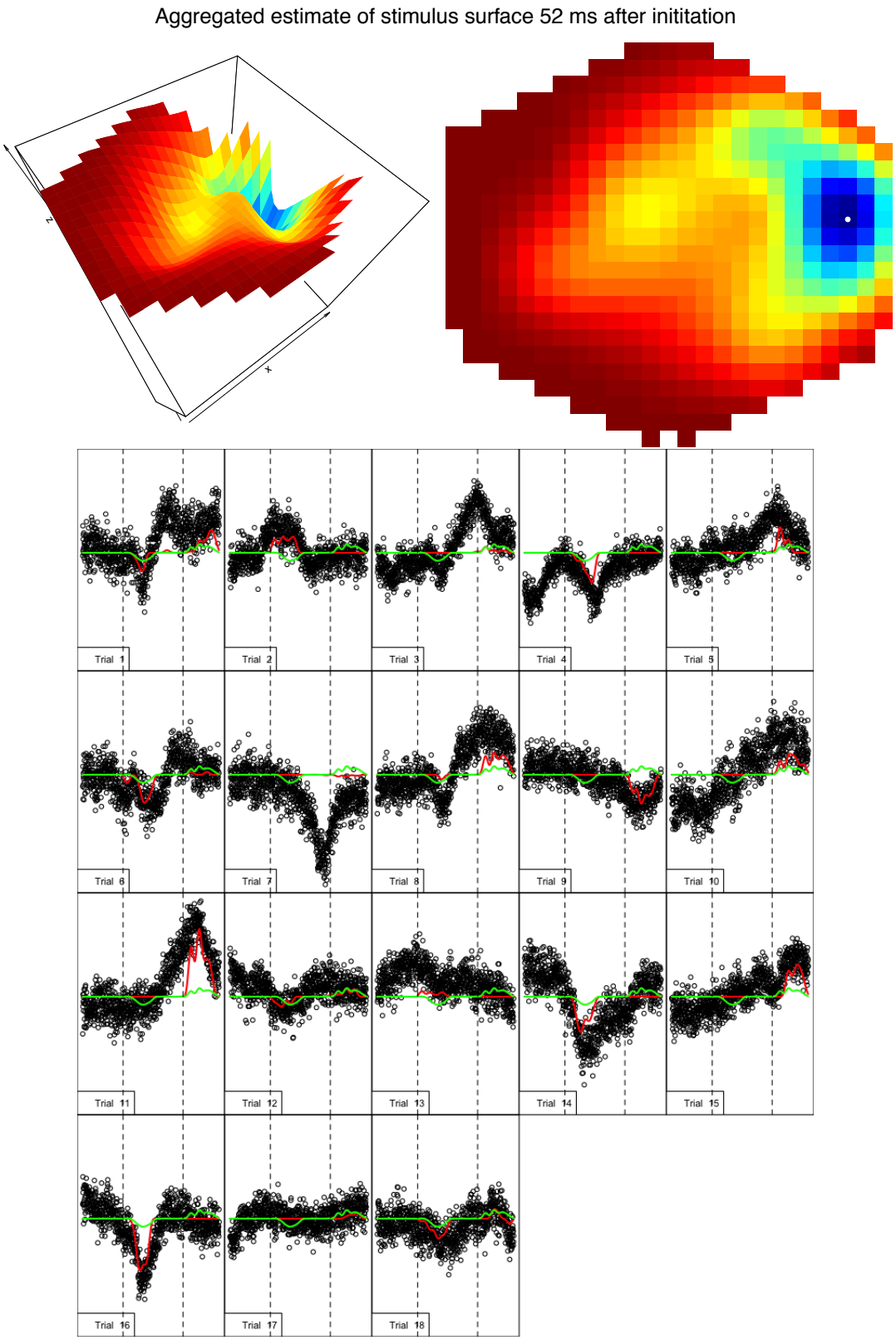
Agg. estimate of integrated effects as a function of separation, animal 410 , model 6



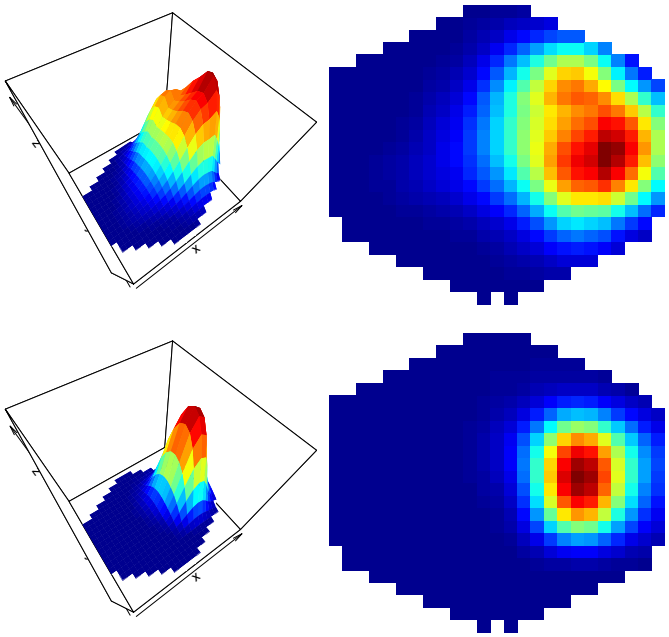
Density plot of aggregated estimates of weight values for animal 410 model 6



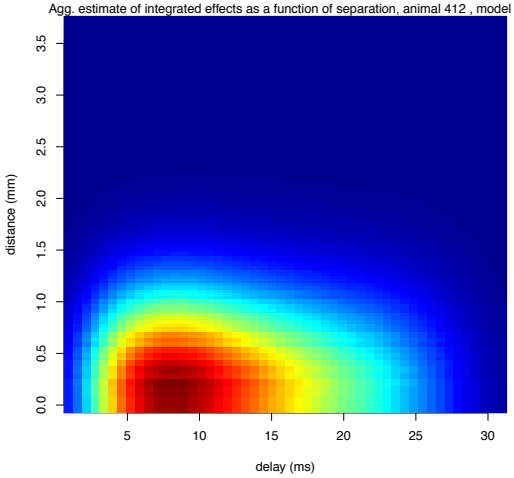
2.6. Animal 412



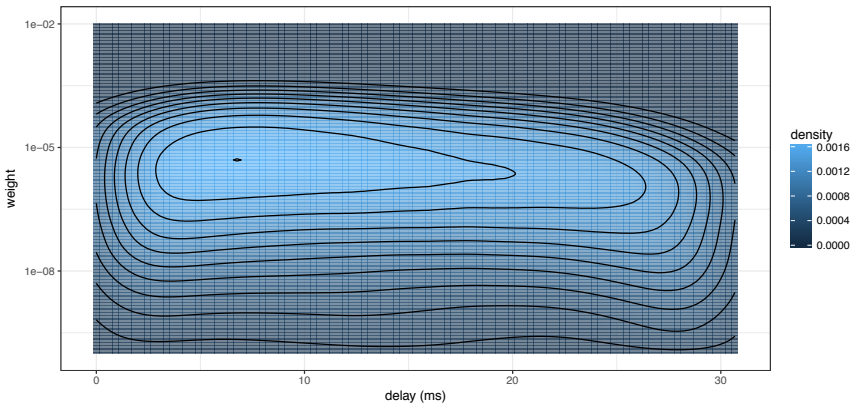
Mean aggregated estimate of integrated network function for model 6



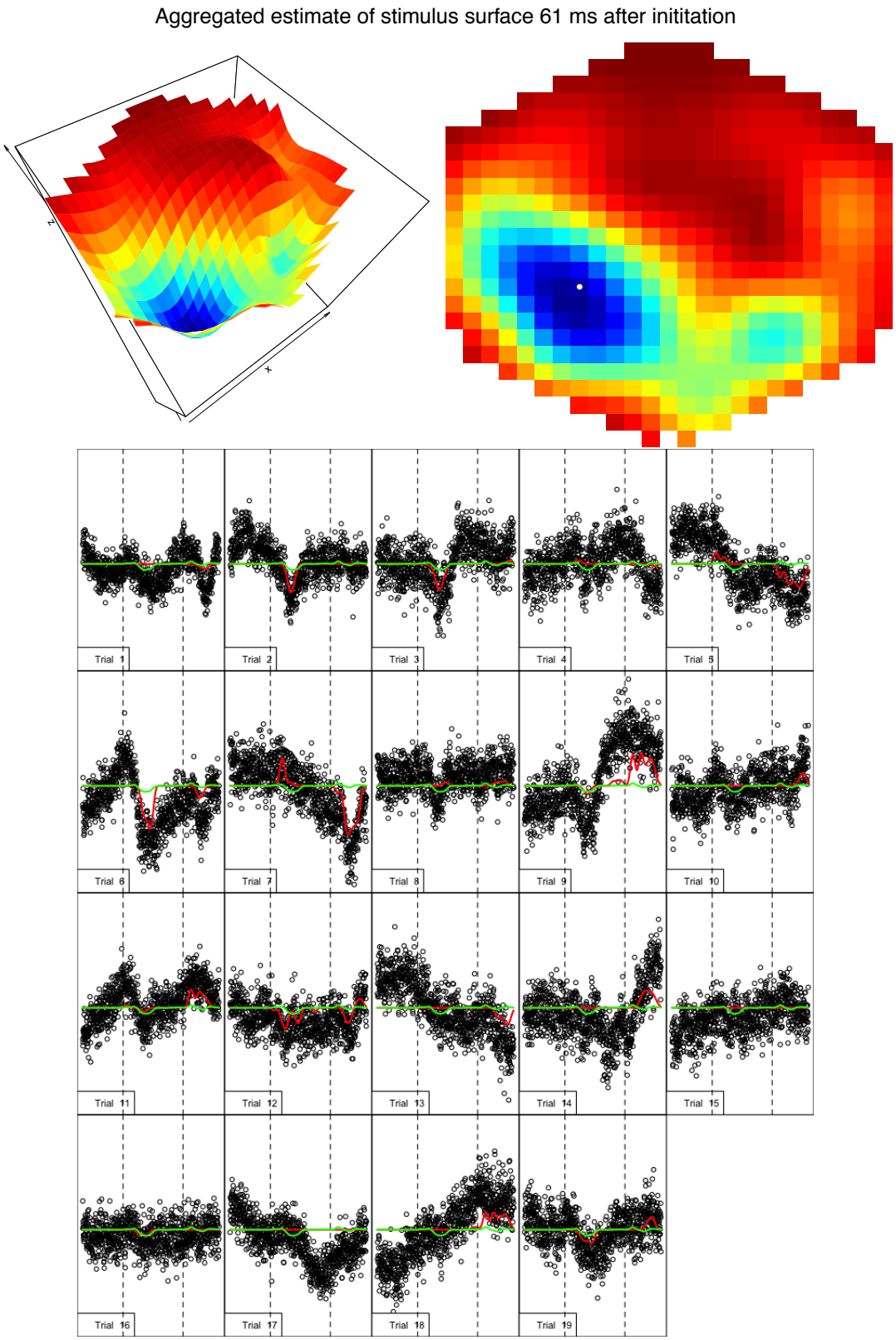
Agg. estimate of integrated effects as a function of separation, animal 412 , model 6



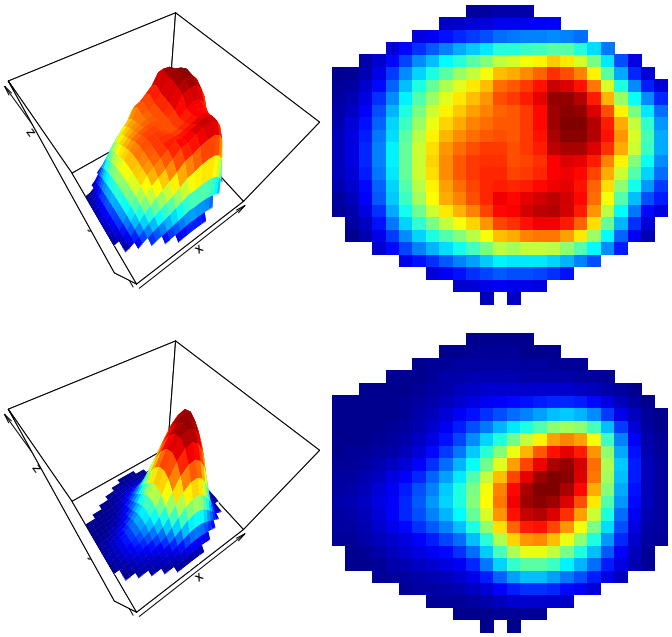
Density plot of agg. estimated weight values for animal 412 model 6



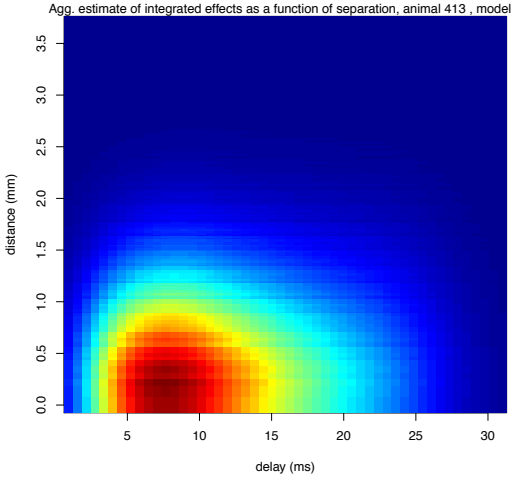
2.7. Animal 413



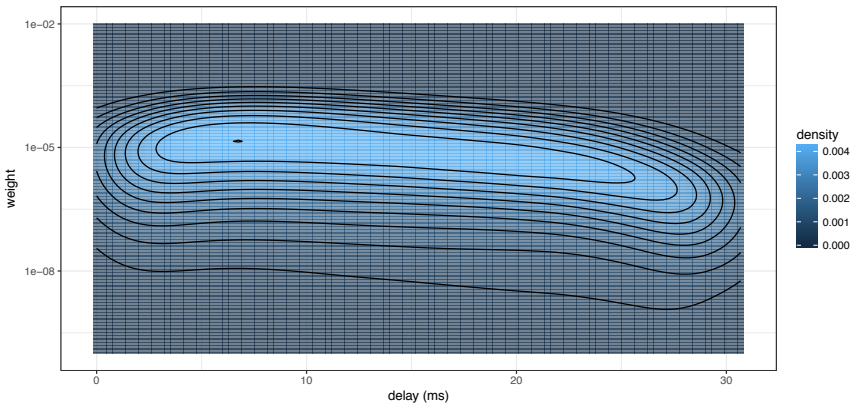
Mean aggregated estimate of integrated network function for model 6



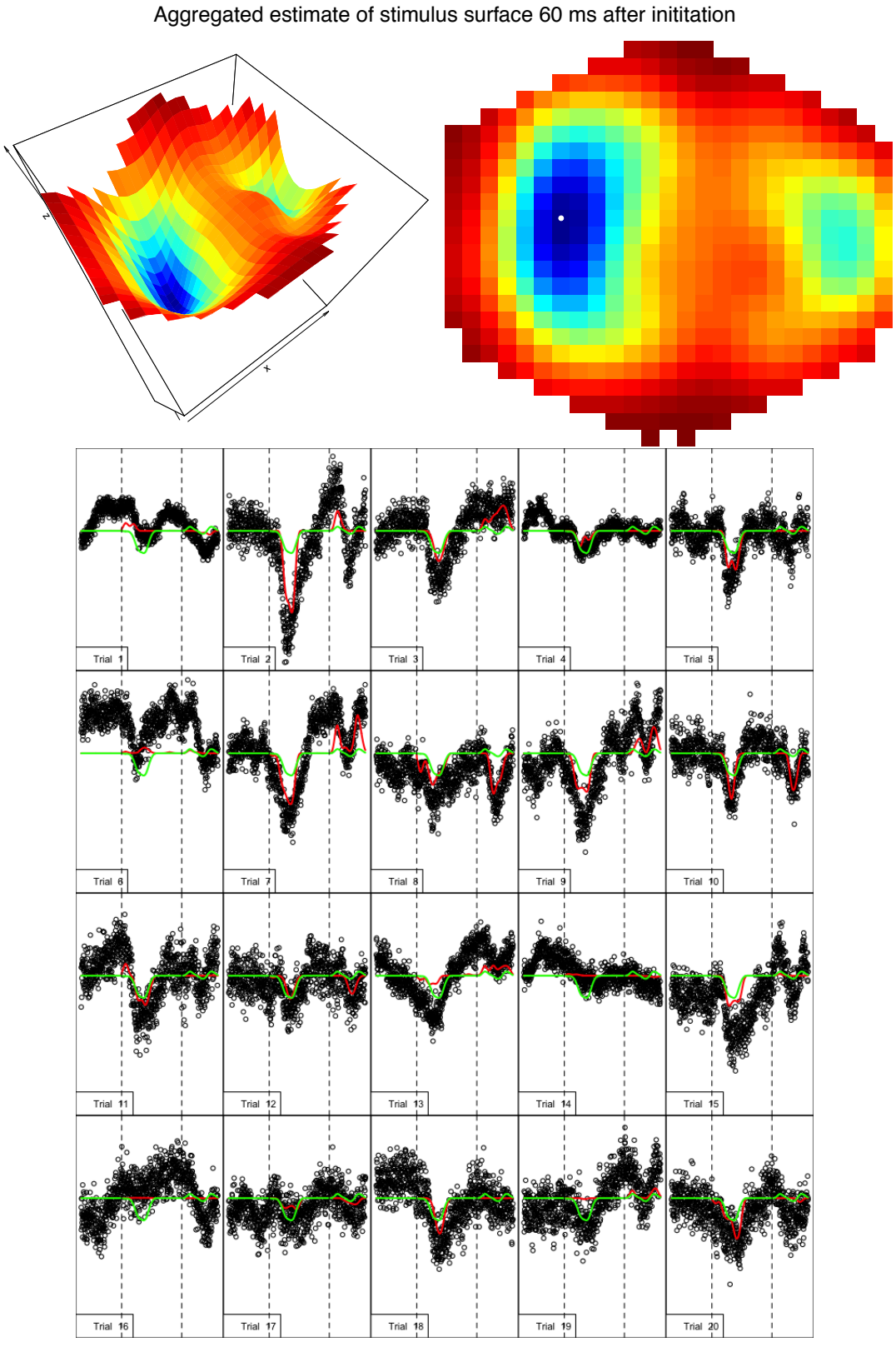
Agg. estimate of integrated effects as a function of separation, animal 413 , model 6



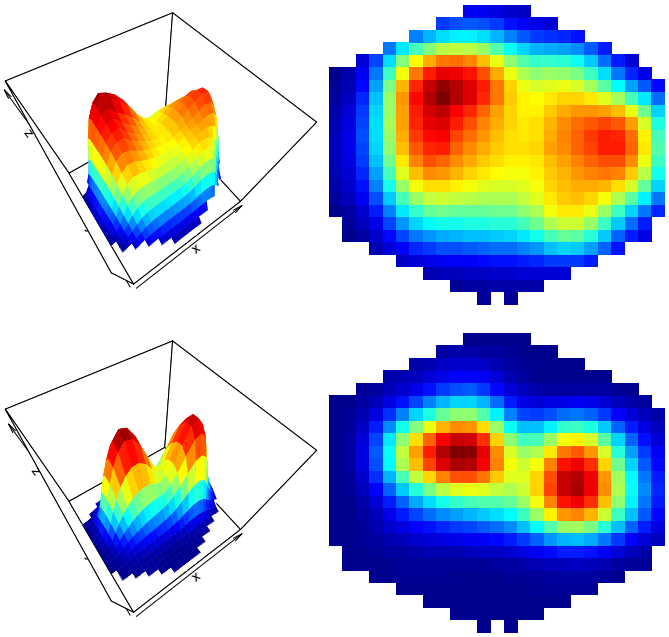
Density plot of agg. estimated weight values for animal 413 model 6



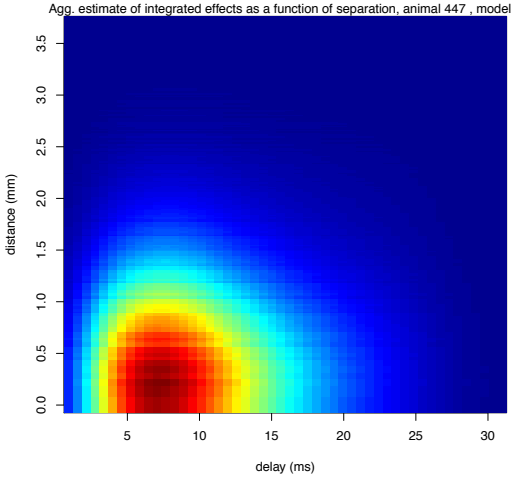
2.8. Animal 447



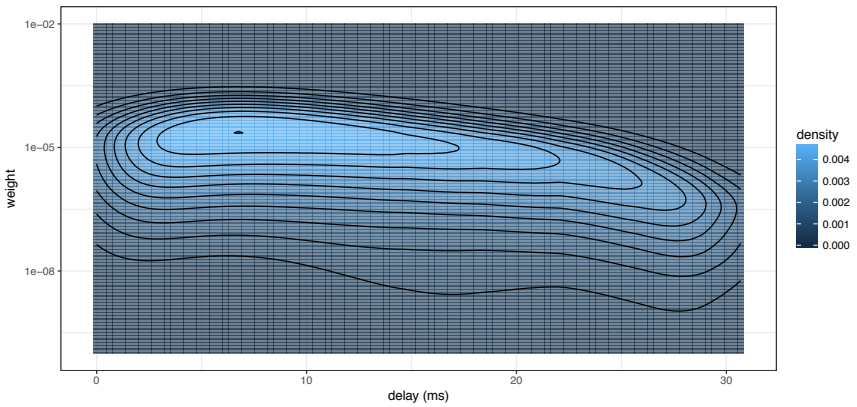
Mean aggregated estimate of integrated network function for model 6



Agg. estimate of integrated effects as a function of separation, animal 447 , model 6

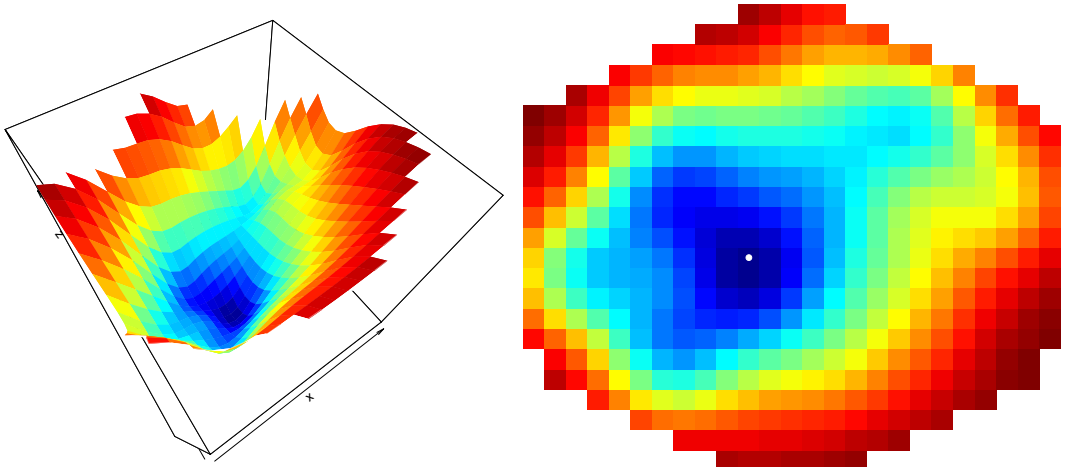


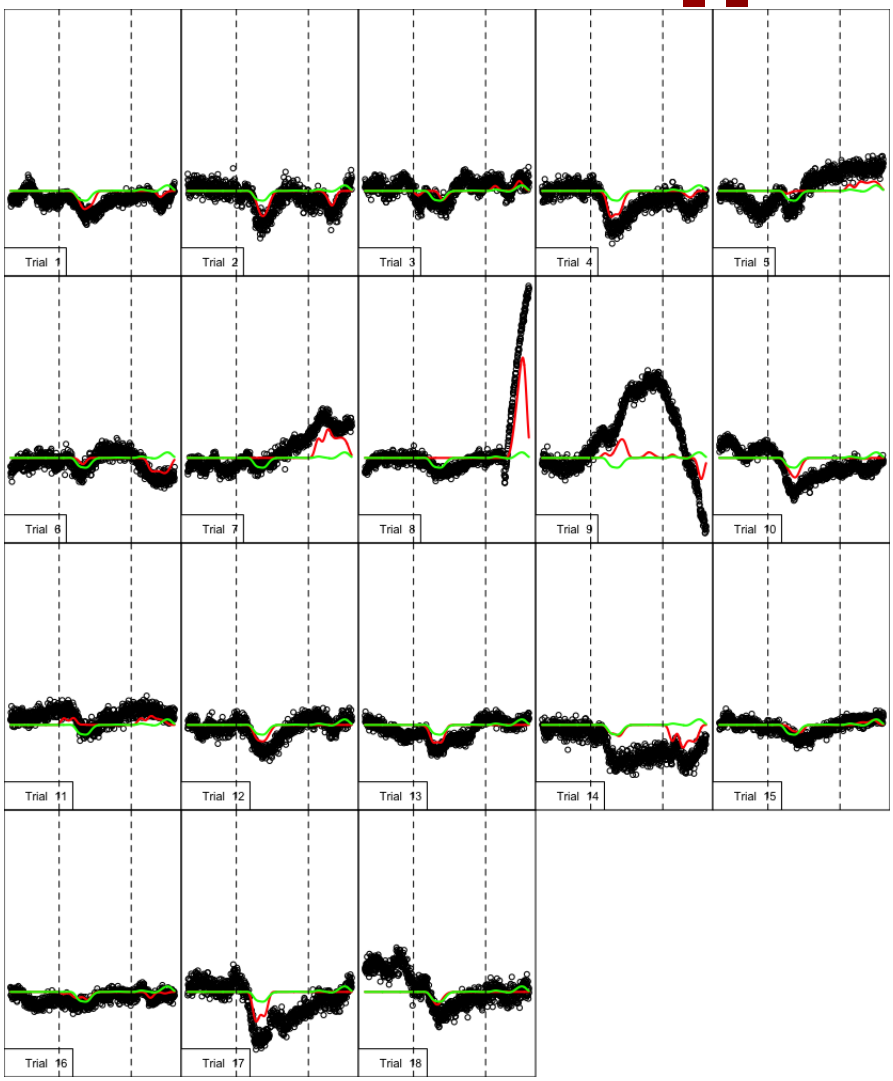
Density plot of agg. estimated weight values for animal 447 model 6



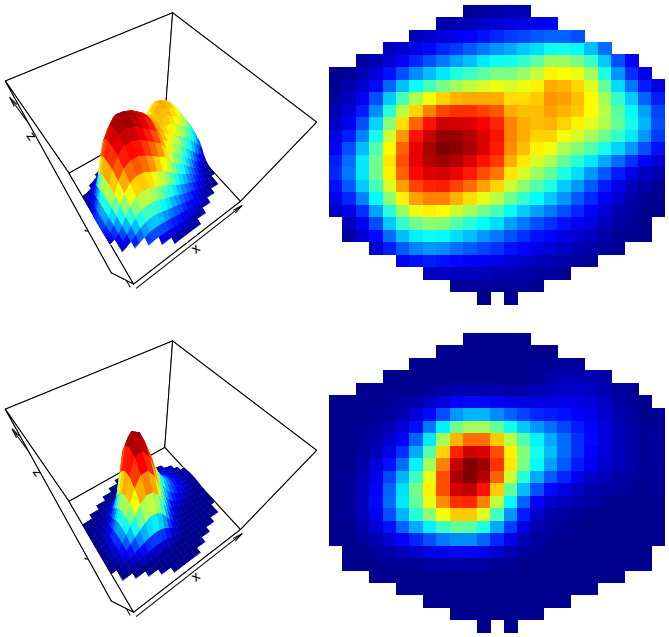
2.9. Animal 568

Aggregated estimate of stimulus surface 60 ms after inititation

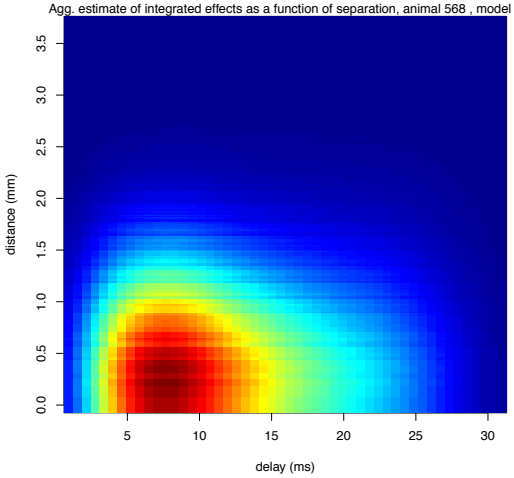




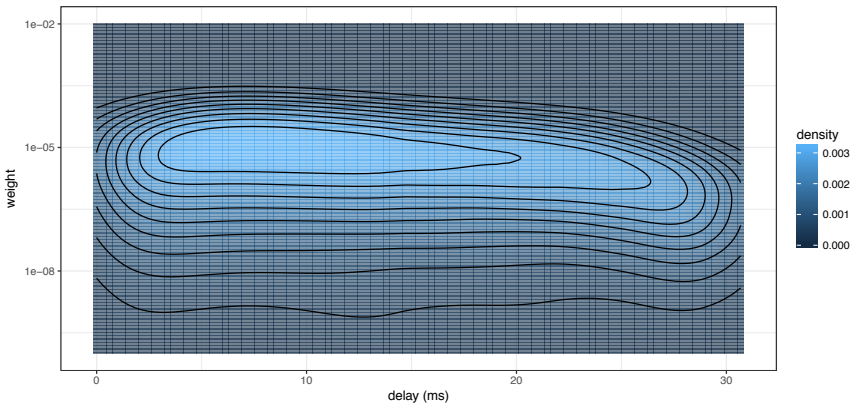
Mean aggregated estimate of integrated network function for model 6



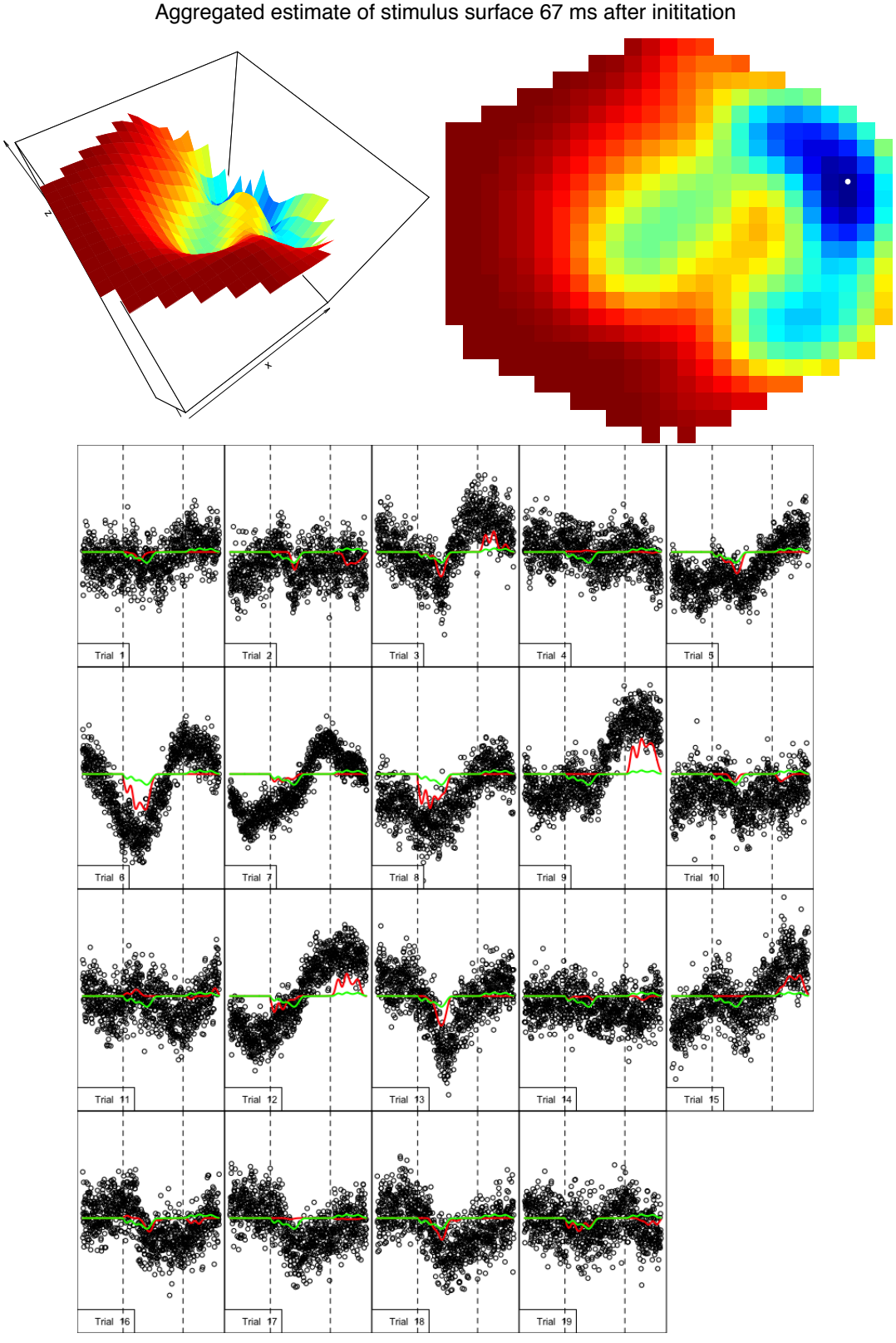
Agg. estimate of integrated effects as a function of separation, animal 568 , model 6



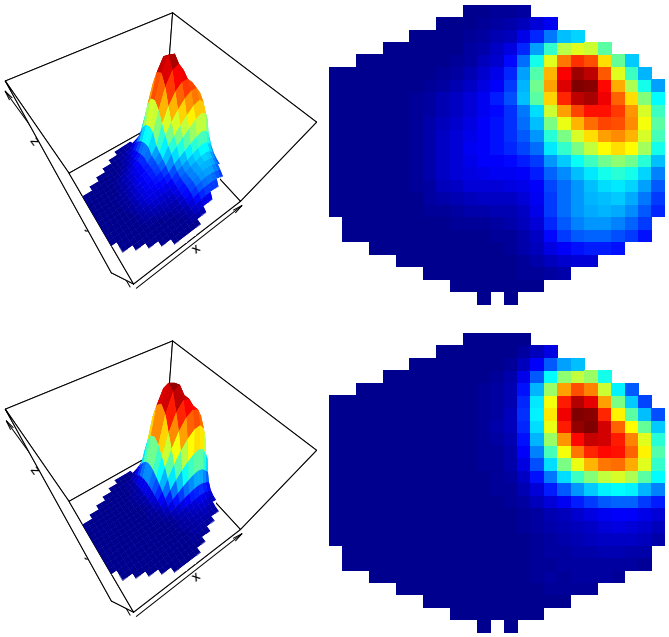
Density plot of agg. estimated weight values for animal 568 model 6



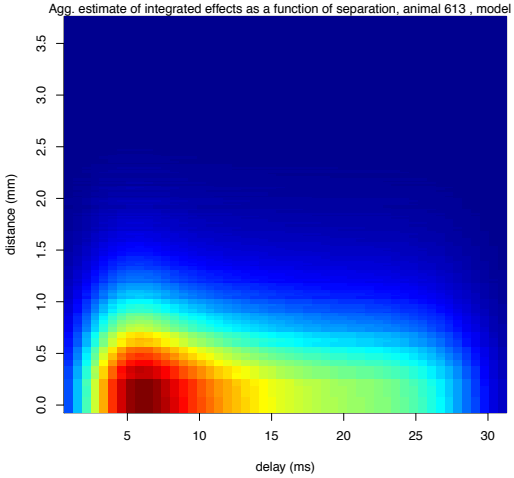
2.10. Animal 613



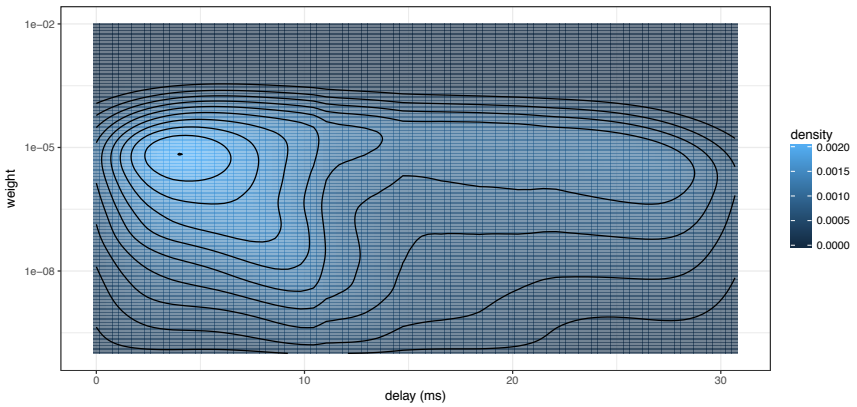
Mean aggregated estimate of integrated network function for model 6



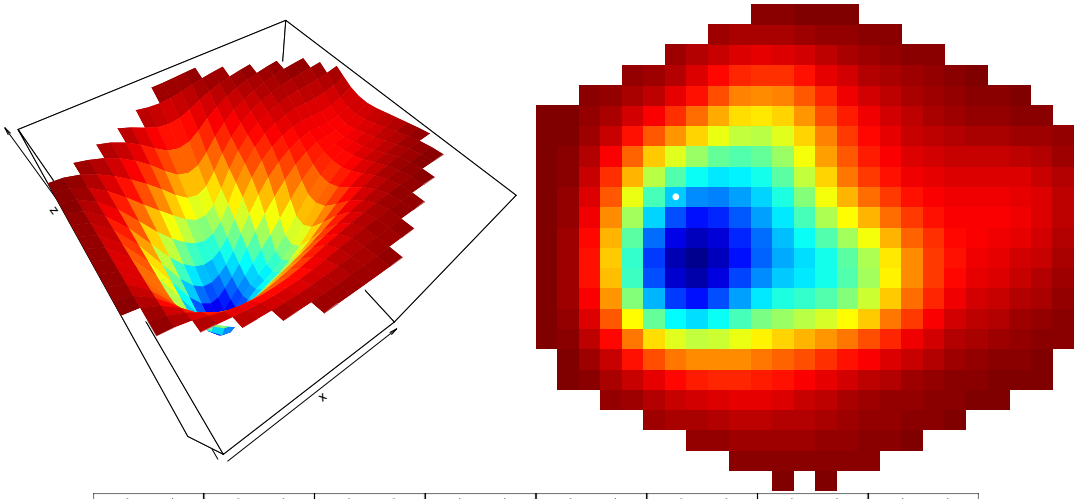
Agg. estimate of integrated effects as a function of separation, animal 613 , model 6

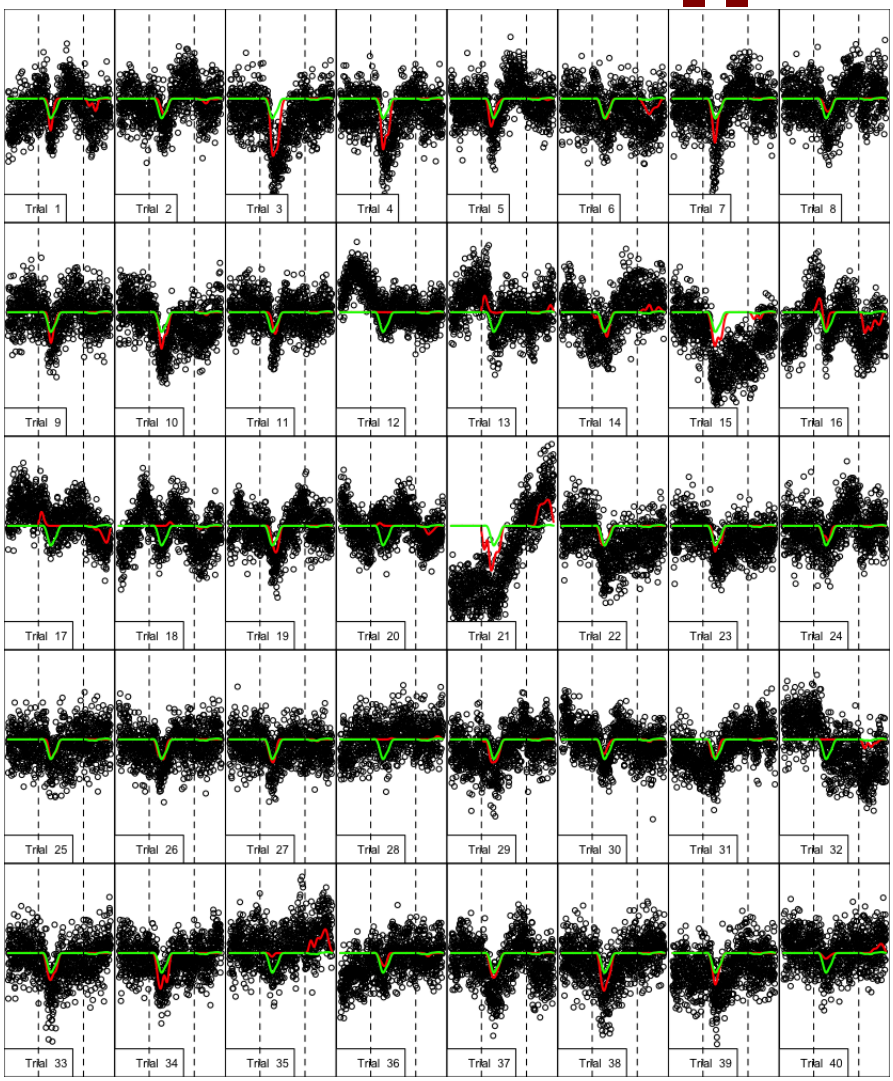


Density plot of agg. estimated weight values for animal 613 model 6

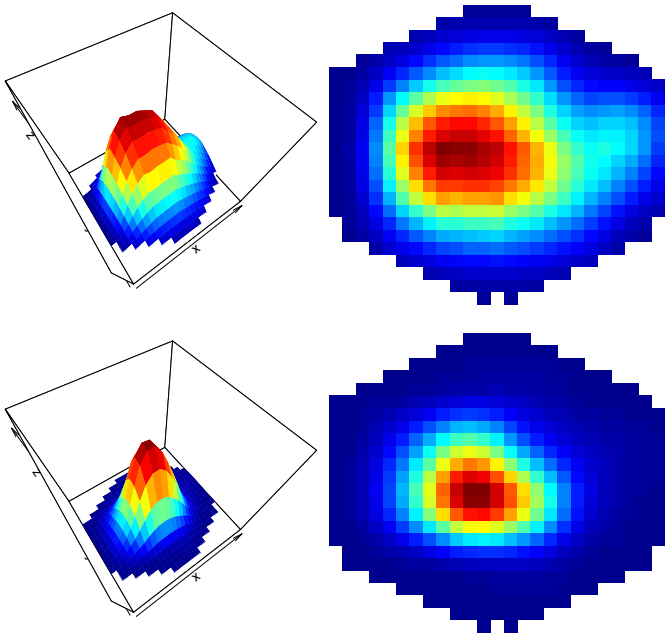


2.11. Animal 679

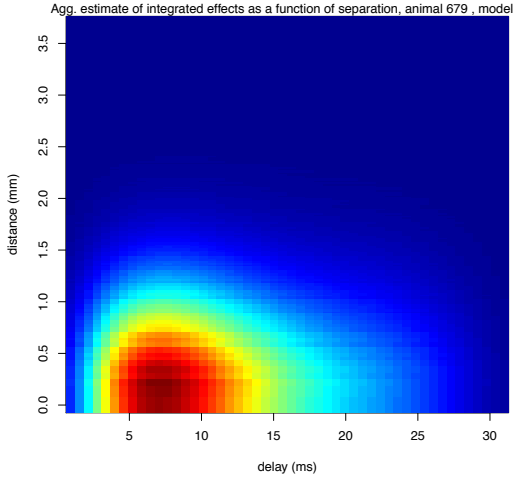




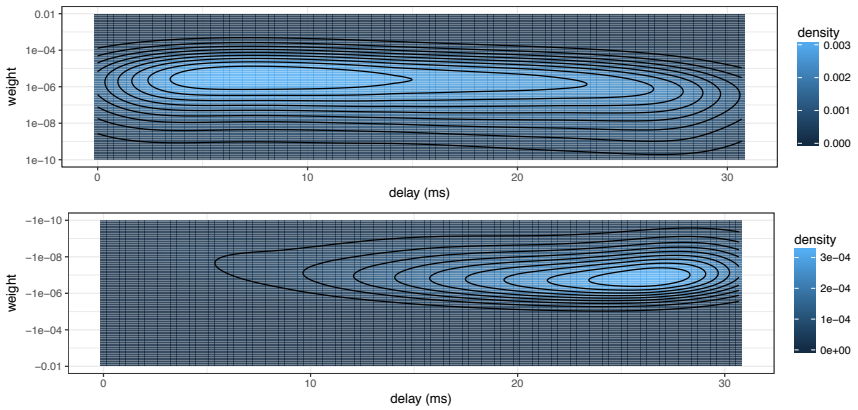
Mean aggregated estimate of integrated network function for model 6



Agg. estimate of integrated effects as a function of separation, animal 679 , model 6

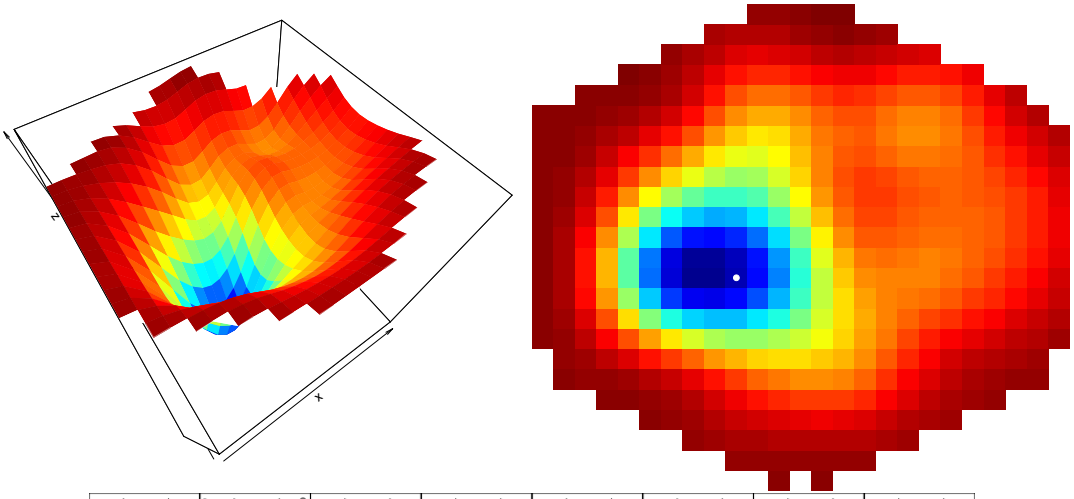


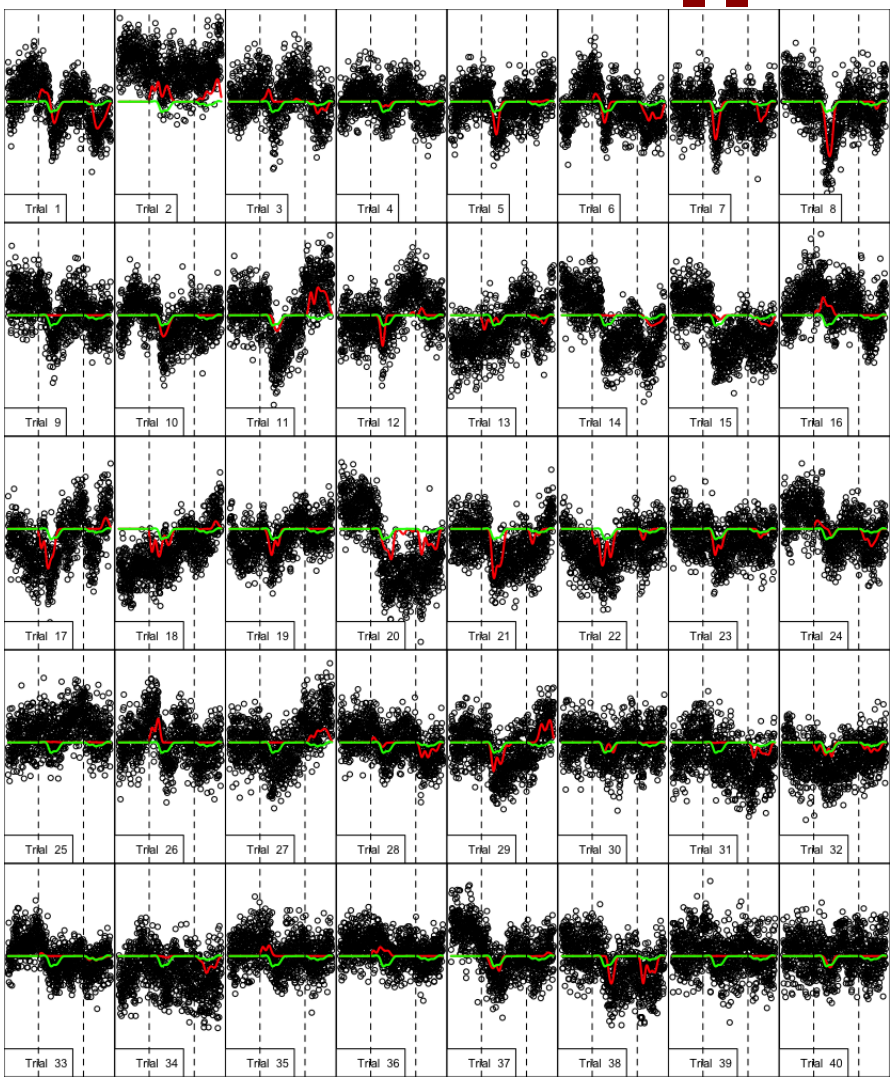
Density plot of aggregated estimates of weight values for animal 679 model 6



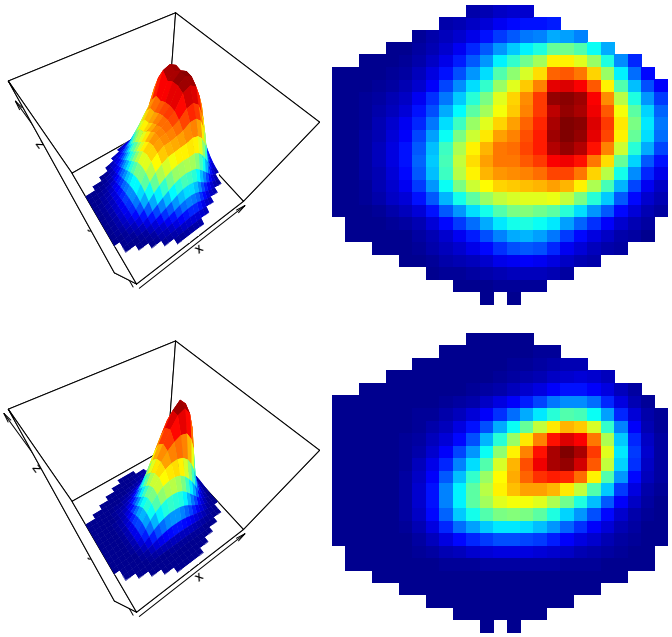
2.12. Animal 685

Aggregated estimate of stimulus surface 40 ms after inititation

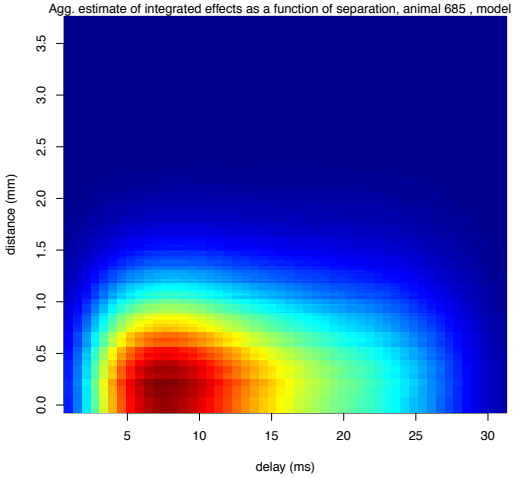




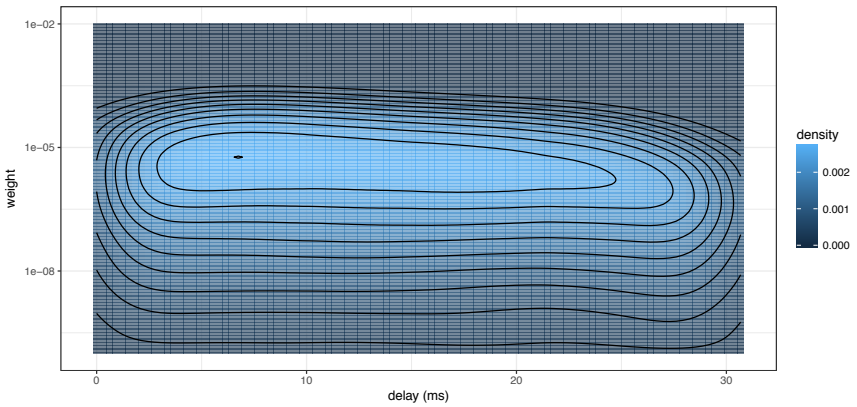
Mean aggregated estimate of integrated network function for model 6



Agg. estimate of integrated effects as a function of separation, animal 685 , model 6



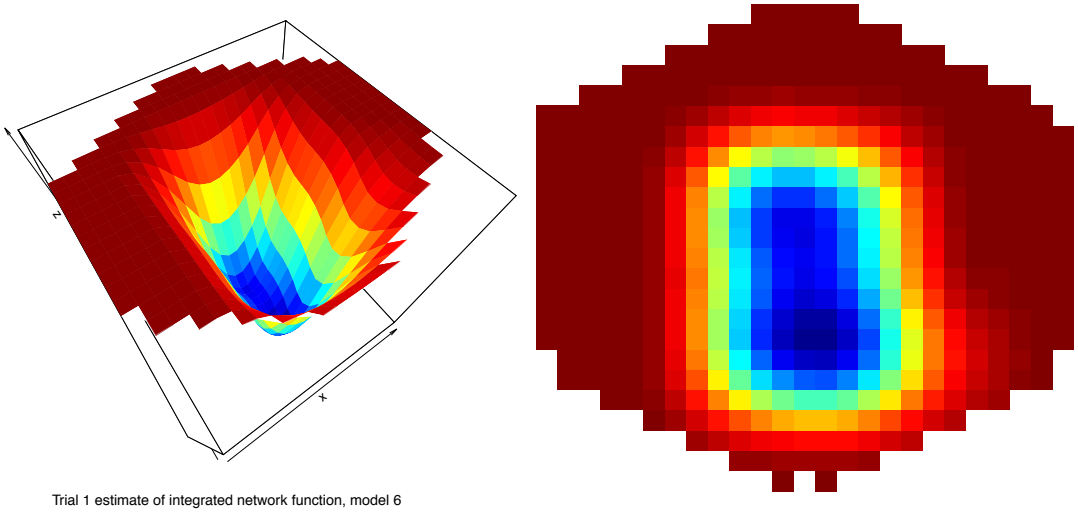
Density plot of agg. estimated weight values for animal 685 model 6



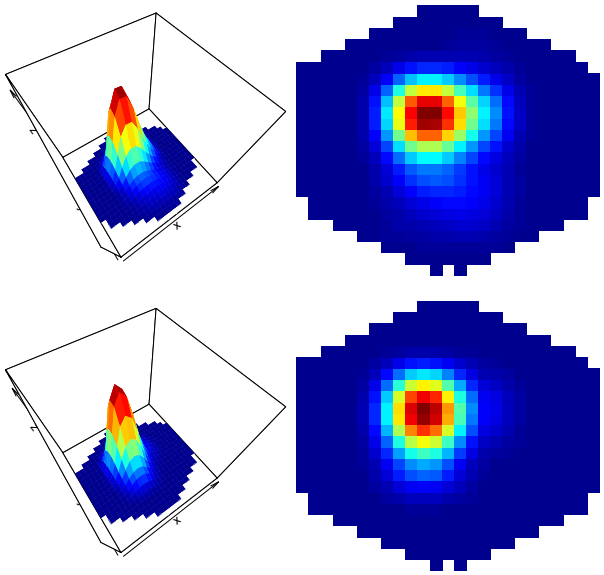
3. Individual fit for each trial for animal 308

Here we present the visualizations of the individual fits to each of the 12 trials for animal 308 that were used to obtain the mean aggregated estimates presented in Section 3, for model no. 6.

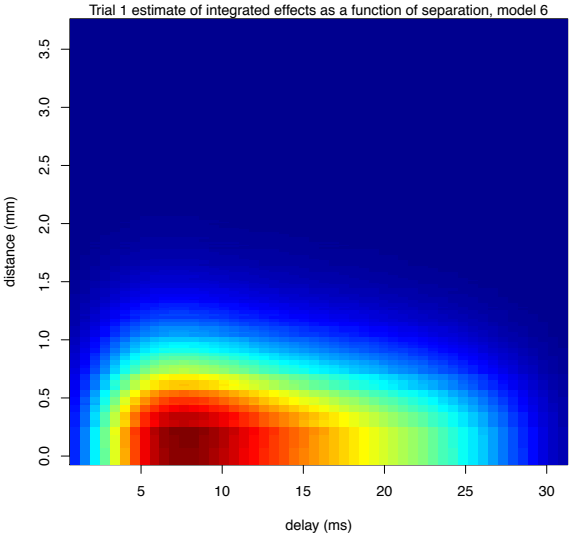
Estimated stimulus surface 70 ms after inititation, trial 1 model 6



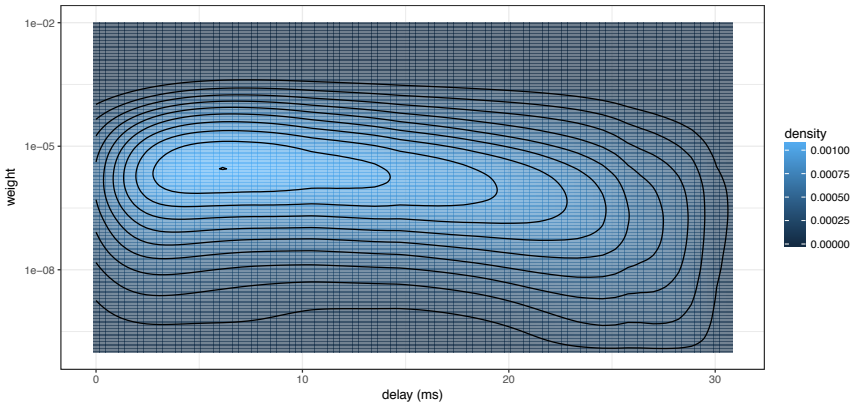
Trial 1 estimate of integrated network function, model 6



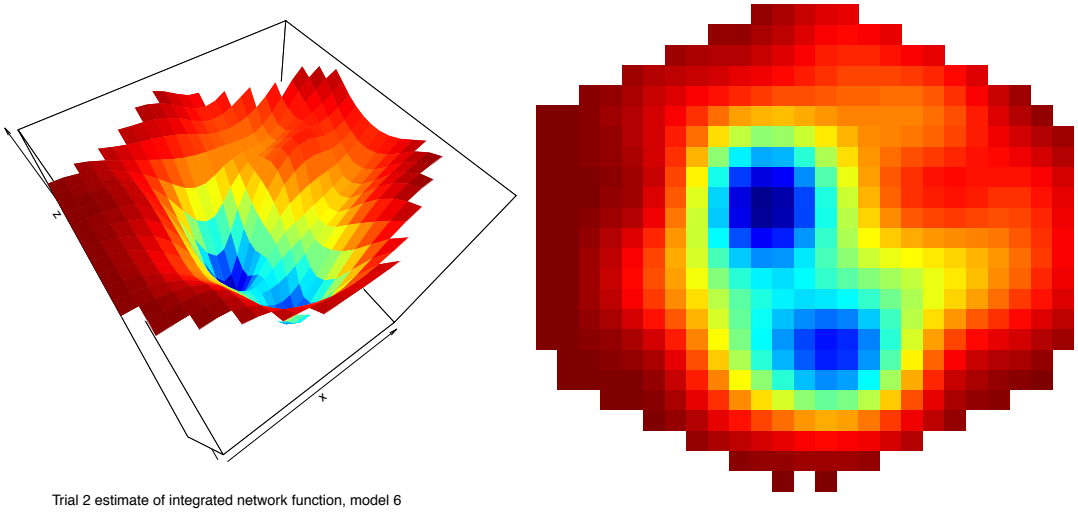
Trial 1 estimate of integrated effects as a function of separation, model 6



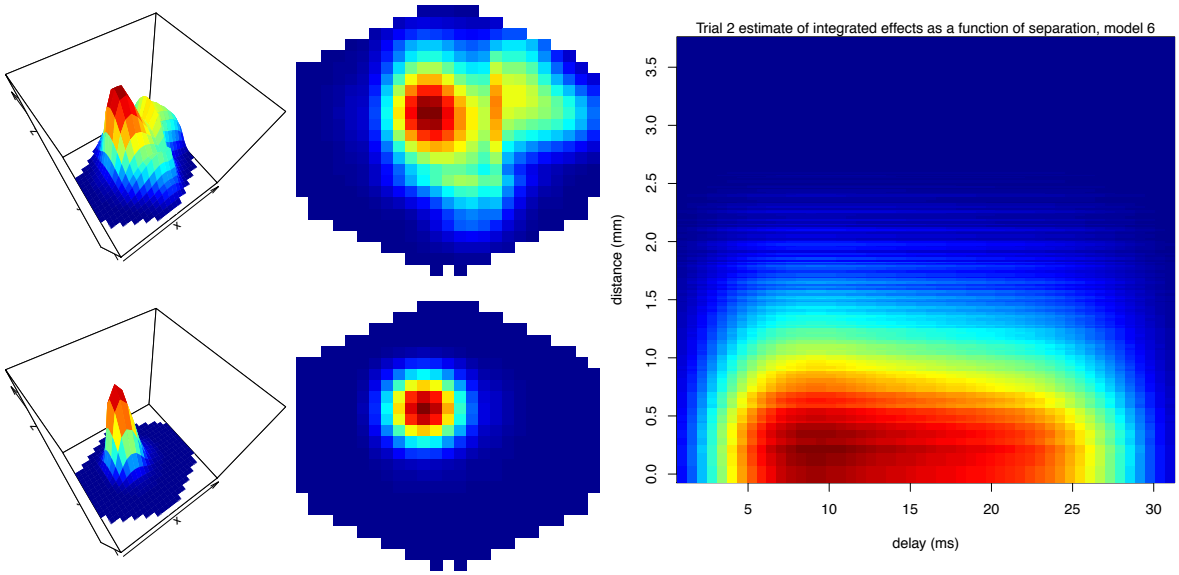
Density plot of estimated weight values for trial 1 model 6



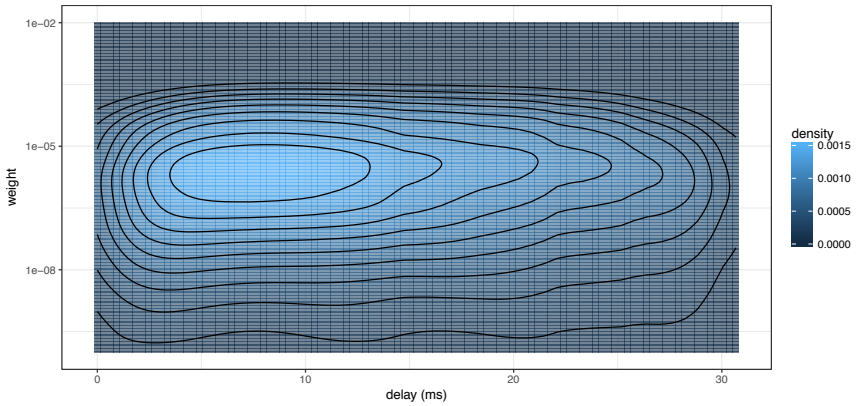
Estimated stimulus surface 70 ms after inititation, trial 2 model 6



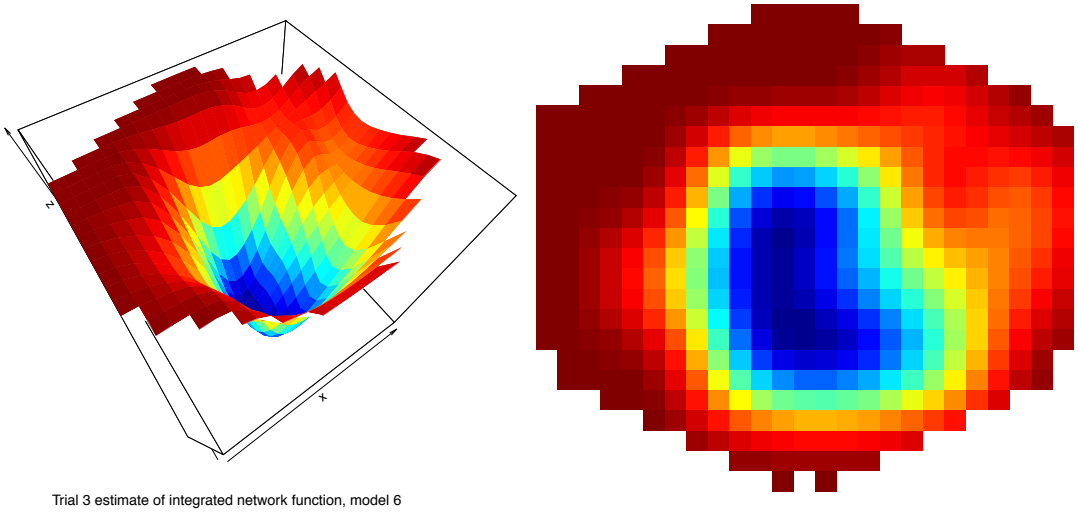
Trial 2 estimate of integrated network function, model 6



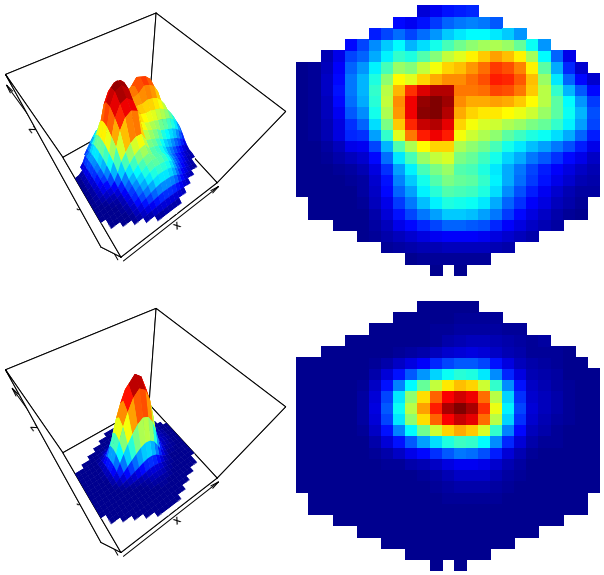
Density plot of estimated weight values for trial 2 model 6



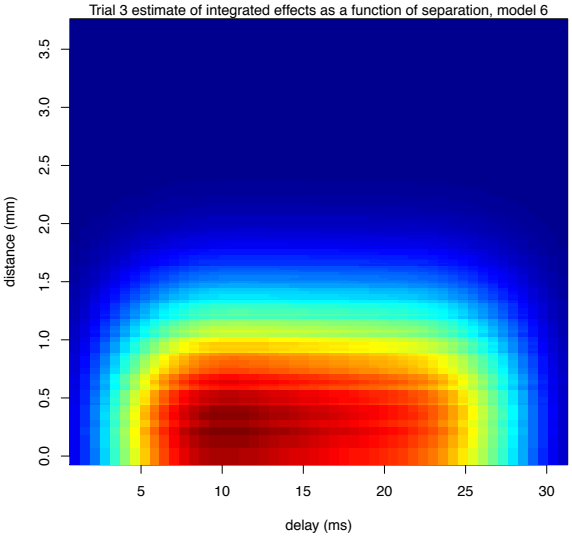
Estimated stimulus surface 70 ms after inititation, trial 3 model 6



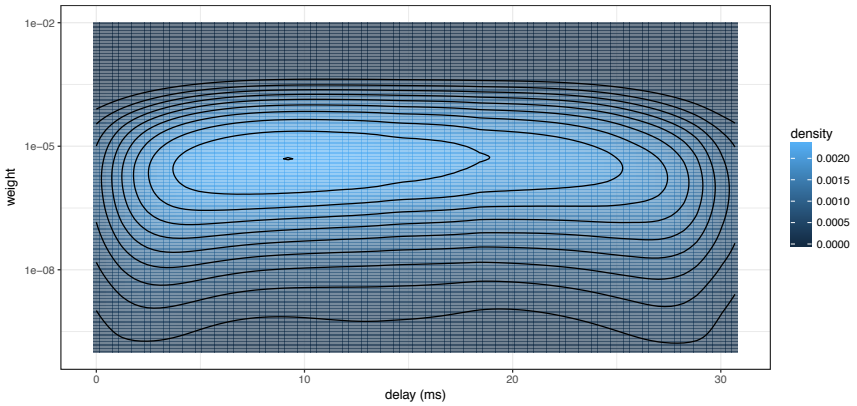
Trial 3 estimate of integrated network function, model 6



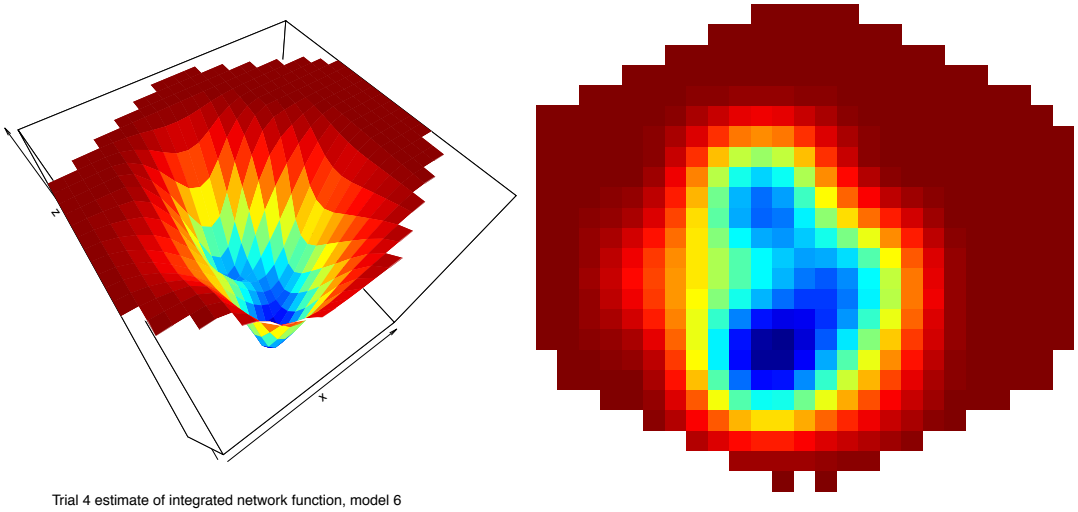
Trial 3 estimate of integrated effects as a function of separation, model 6



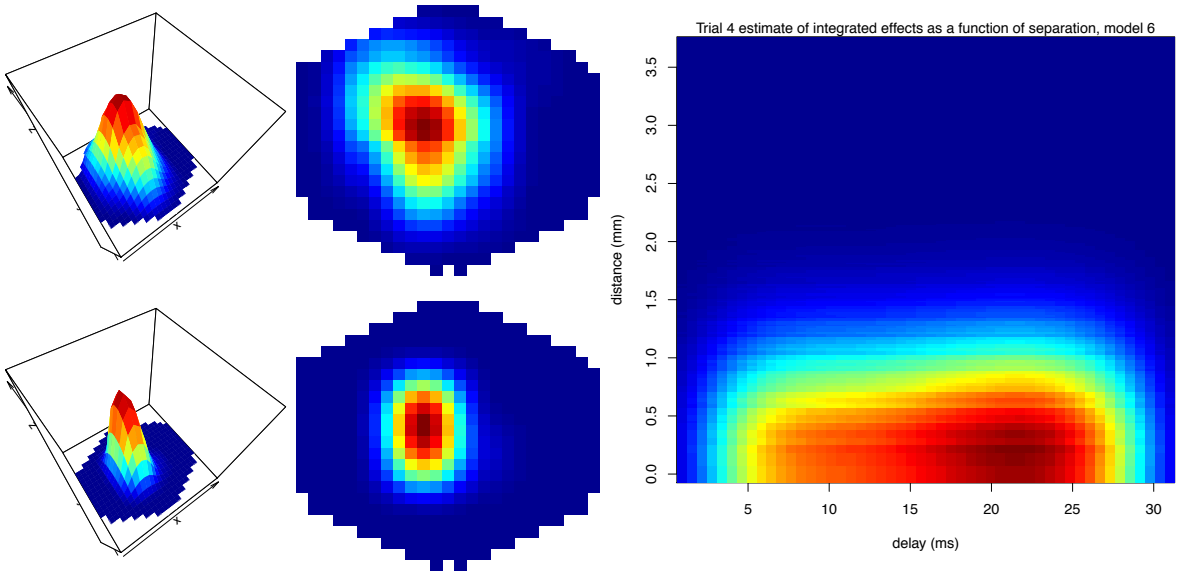
Density plot of estimated weight values for trial 3 model 6



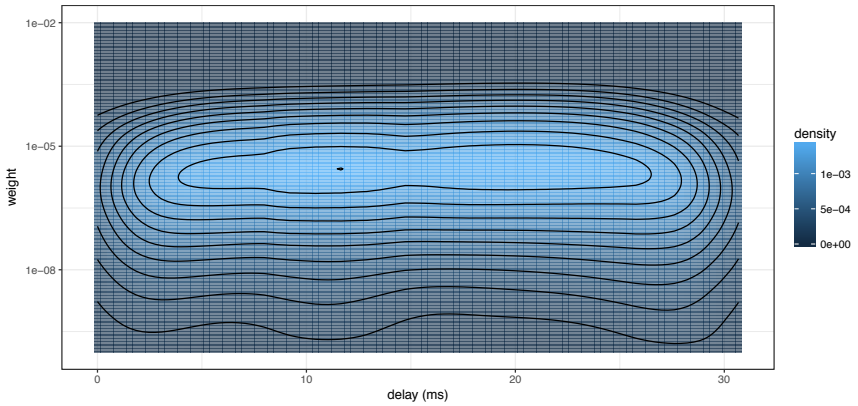
Estimated stimulus surface 70 ms after inititation, trial 4 model 6



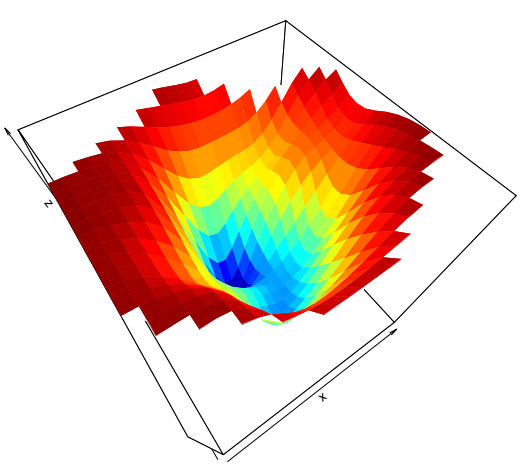
Trial 4 estimate of integrated network function, model 6

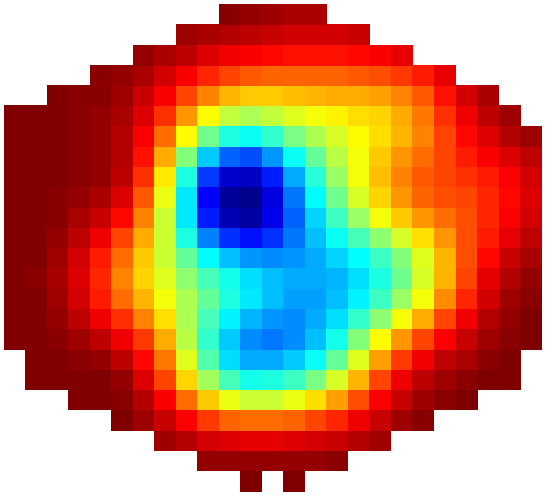


Density plot of estimated weight values for trial 4 model 6

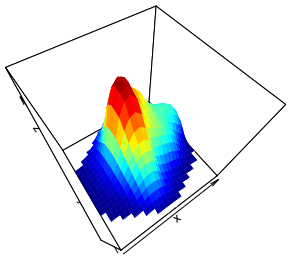


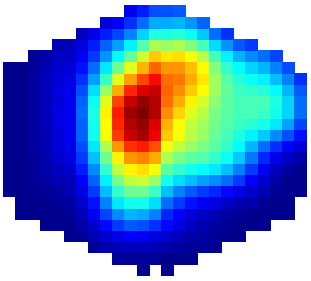
Estimated stimulus surface 70 ms after inititation, trial 5 model 6

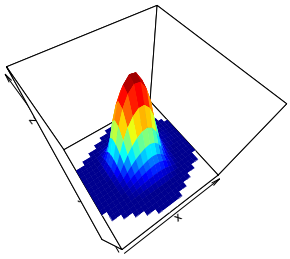


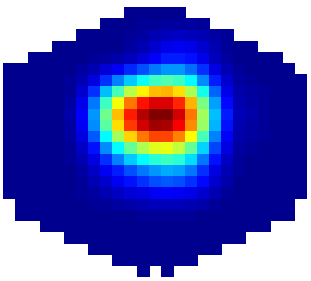


Trial 5 estimate of integrated network function, model 6

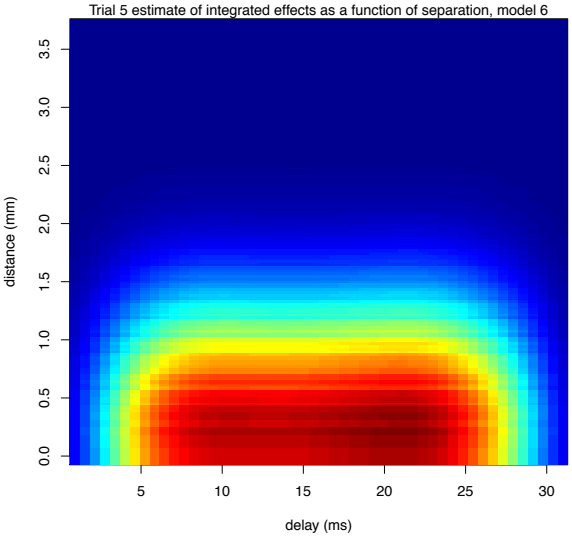




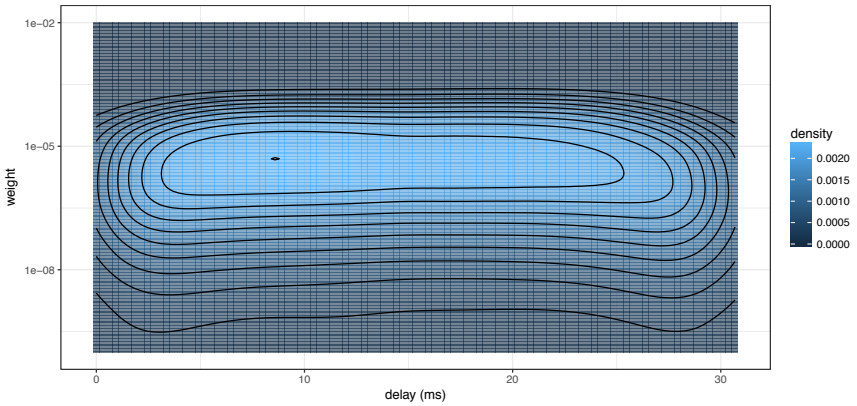




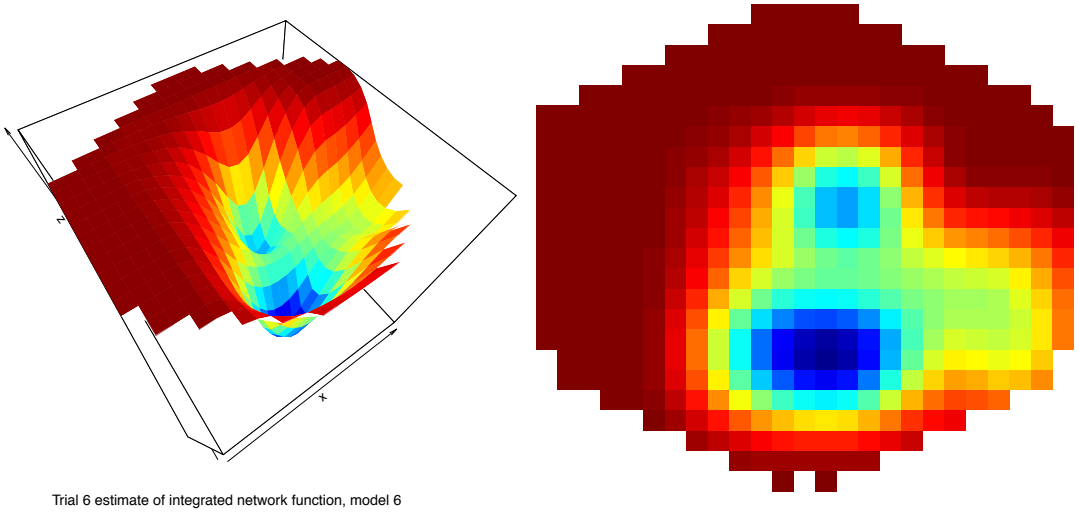
Trial 5 estimate of integrated effects as a function of separation, model 6



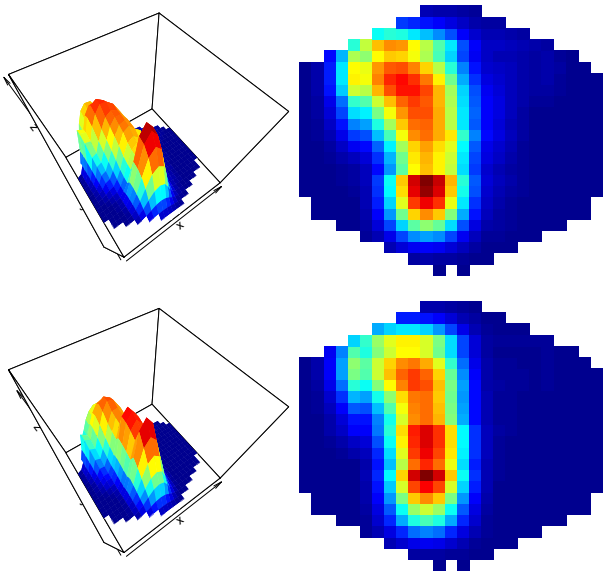
Density plot of estimated weight values for trial 5 model 6



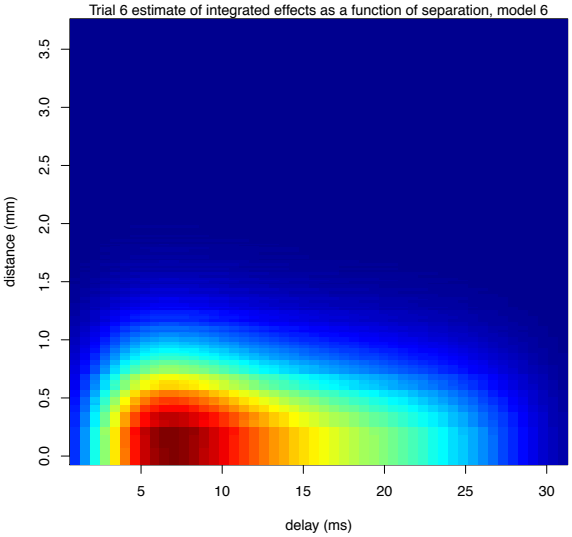
Estimated stimulus surface 70 ms after inititation, trial 6 model 6



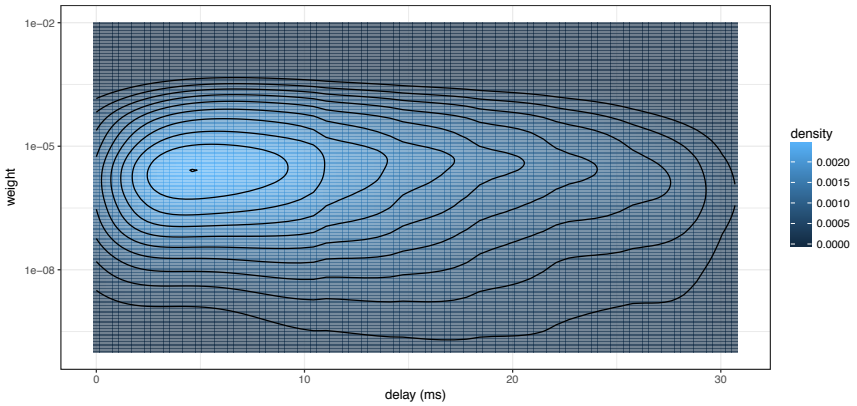
Trial 6 estimate of integrated network function, model 6



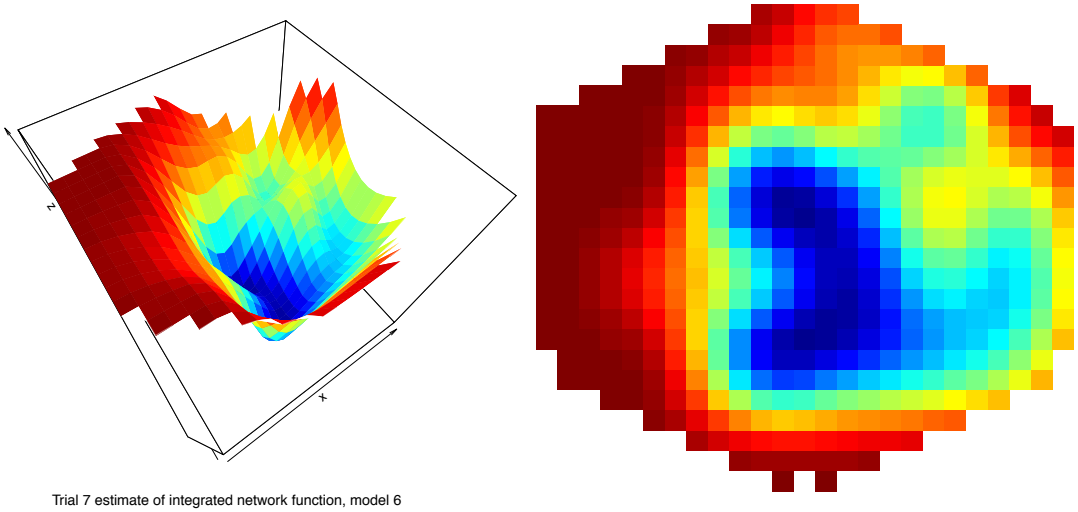
Trial 6 estimate of integrated effects as a function of separation, model 6



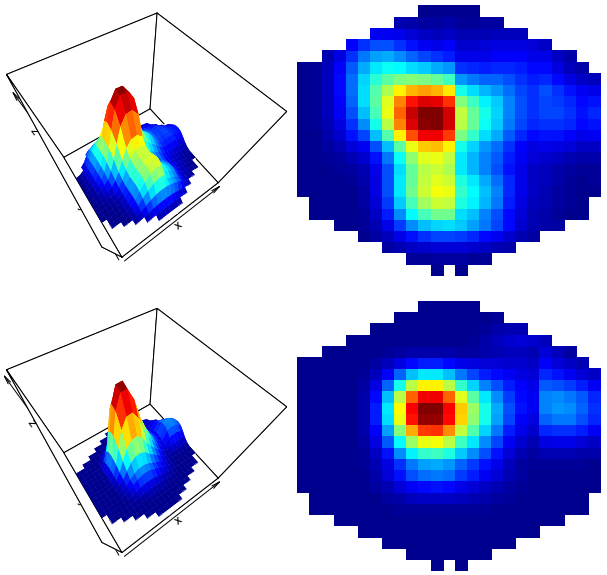
Density plot of estimated weight values for trial 6 model 6



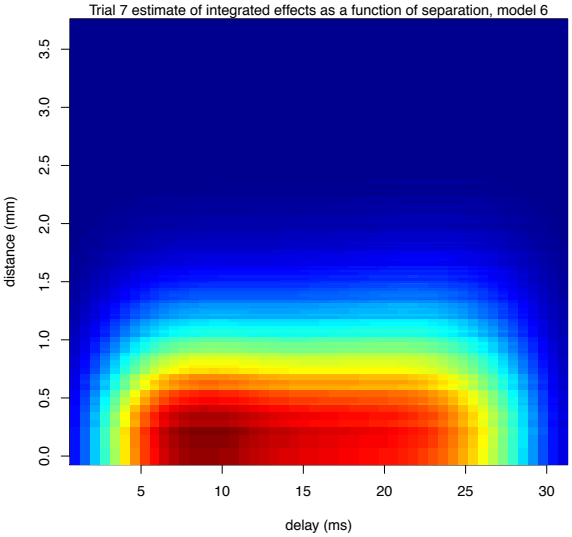
Estimated stimulus surface 70 ms after inititation, trial 7 model 6



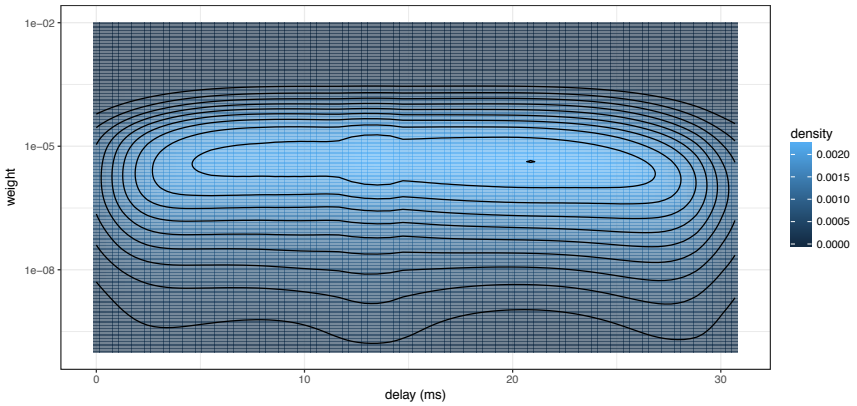
Trial 7 estimate of integrated network function, model 6



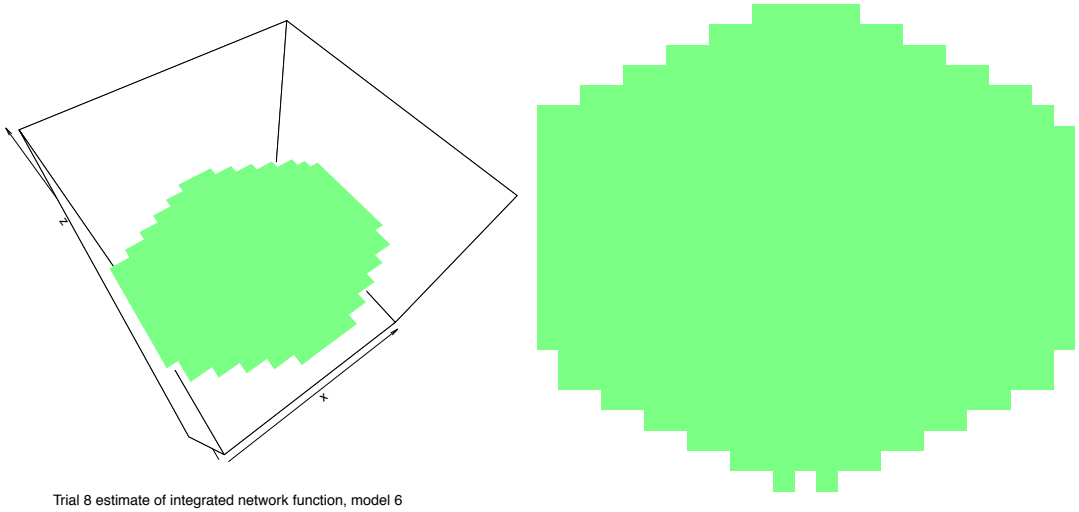
Trial 7 estimate of integrated effects as a function of separation, model 6



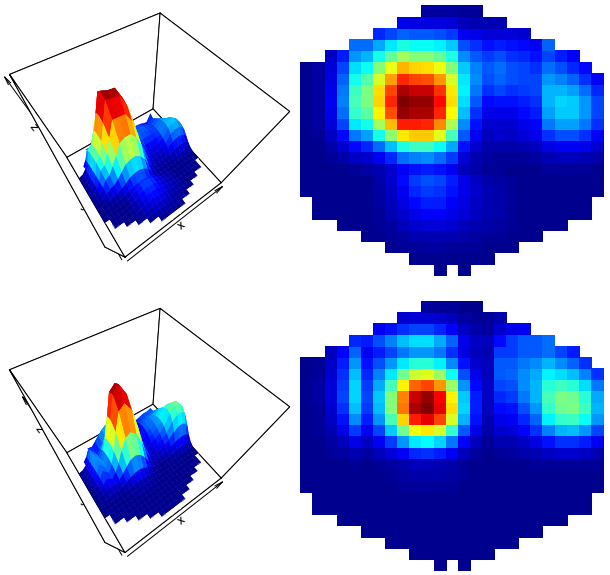
Density plot of estimated weight values for trial 7 model 6



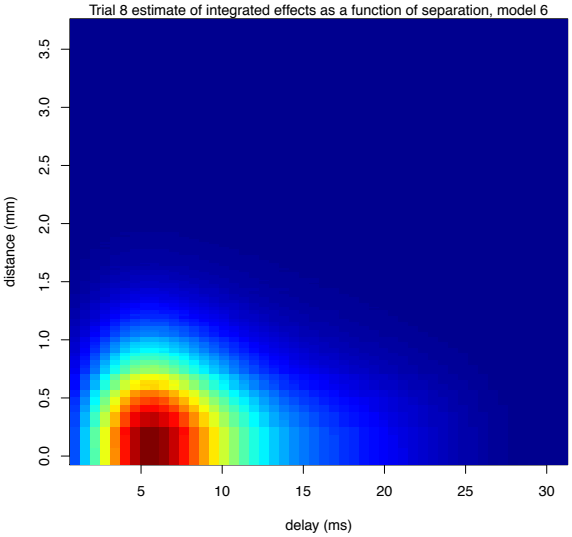
Estimated stimulus surface 70 ms after inititation, trial 8 model 6



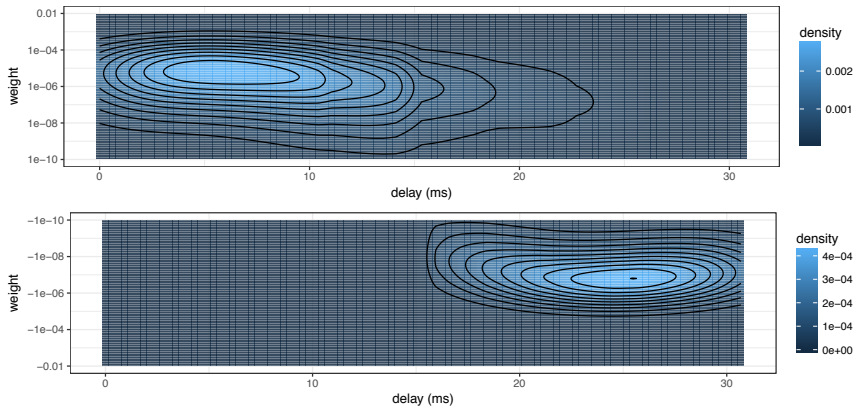
Trial 8 estimate of integrated network function, model 6



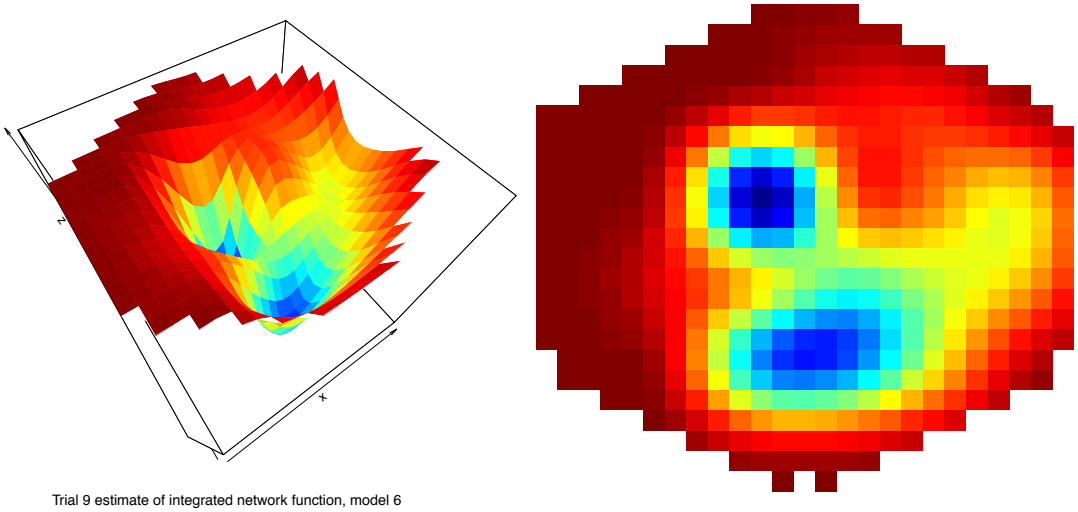
Trial 8 estimate of integrated effects as a function of separation, model 6



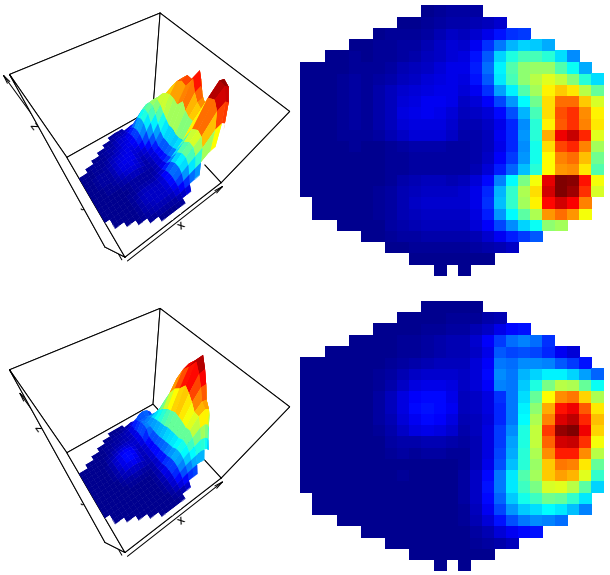
Density plot of estimated weight values for trial 8 model 6



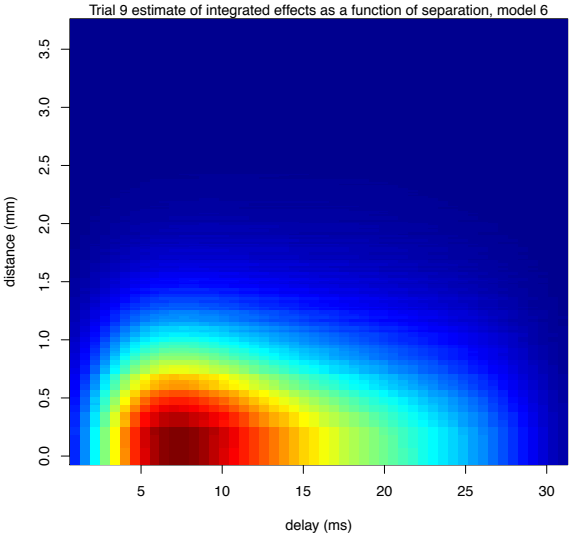
Estimated stimulus surface 70 ms after inititation, trial 9 model 6



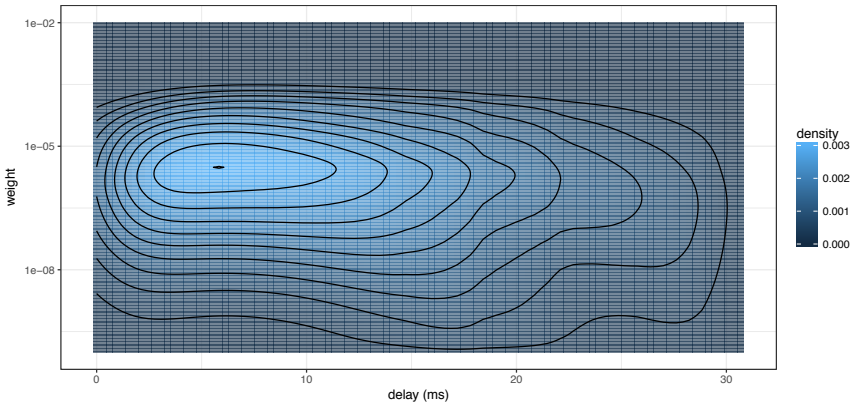
Trial 9 estimate of integrated network function, model 6



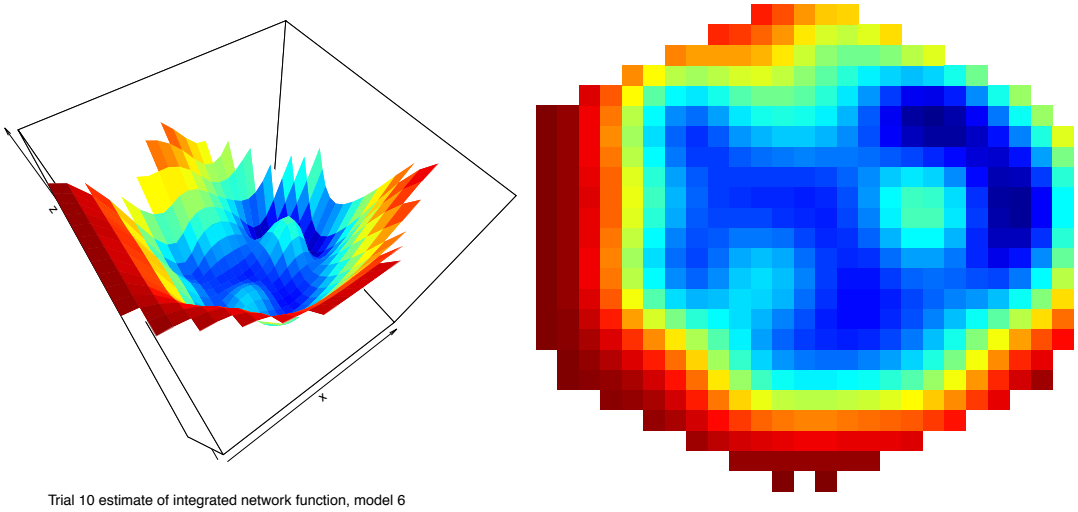
Trial 9 estimate of integrated effects as a function of separation, model 6



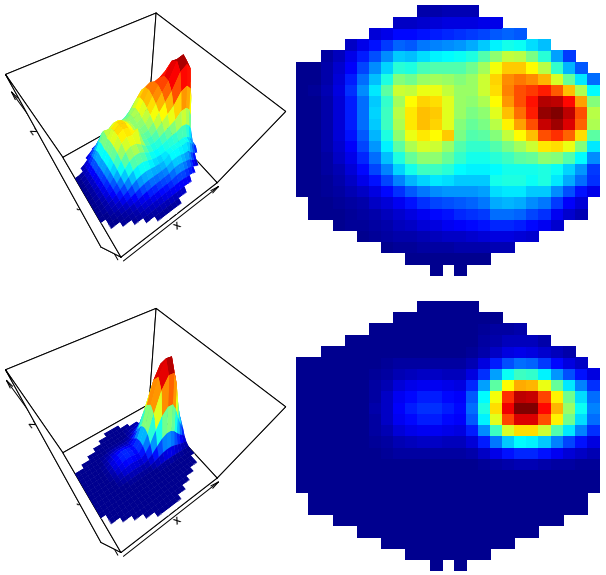
Density plot of estimated weight values for trial 9 model 6



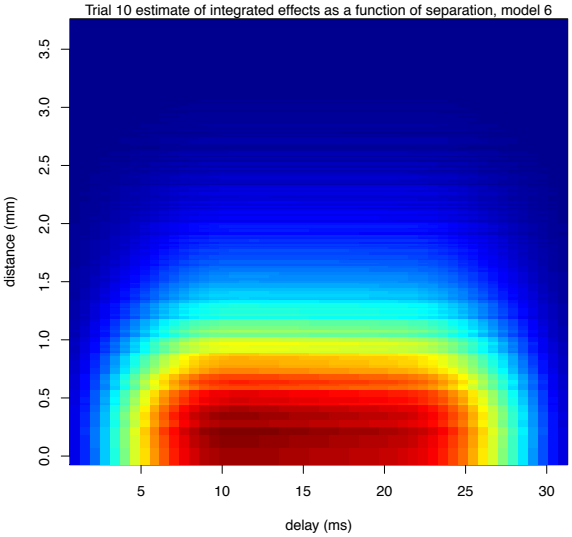
Estimated stimulus surface 70 ms after inititation, trial 10 model 6



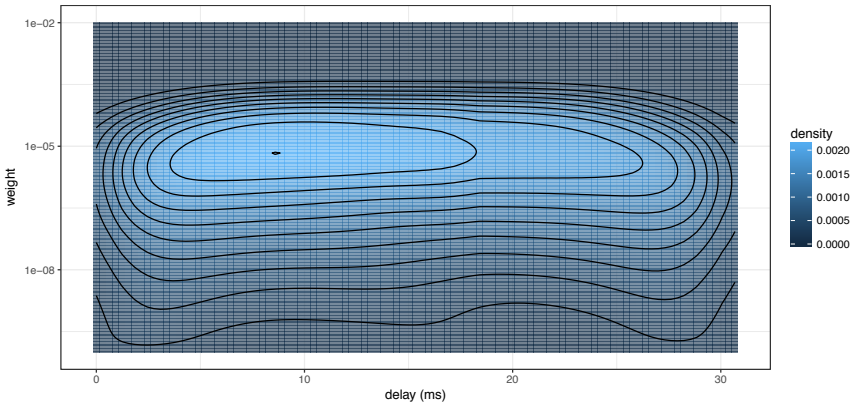
Trial 10 estimate of integrated network function, model 6



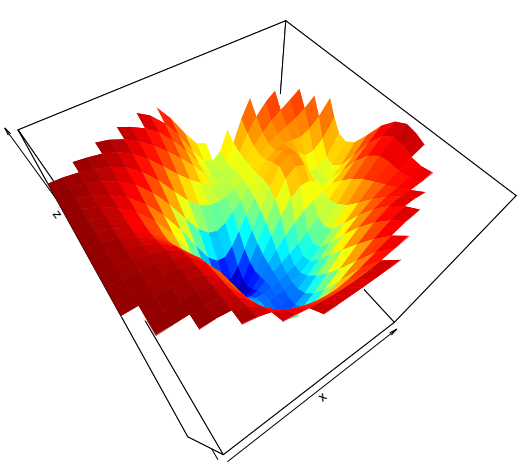
Trial 10 estimate of integrated effects as a function of separation, model 6

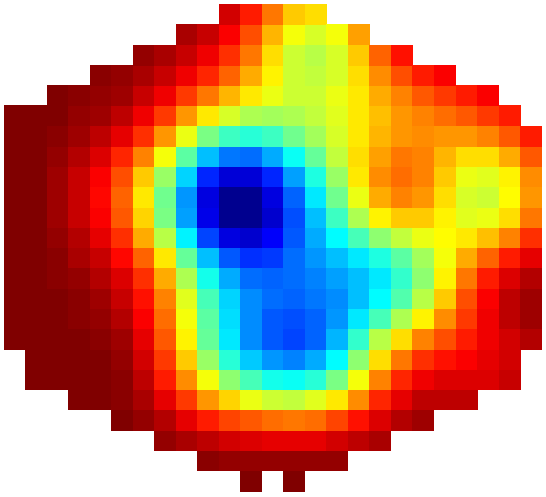


Density plot of estimated weight values for trial 10 model 6

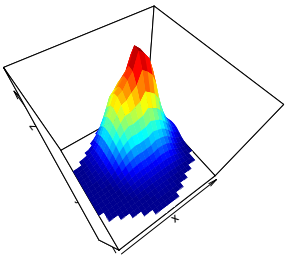


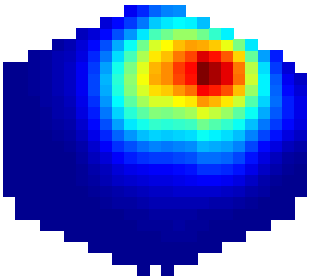
Estimated stimulus surface 70 ms after inititation, trial 11 model 6

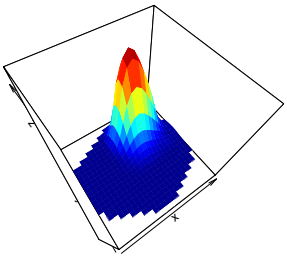


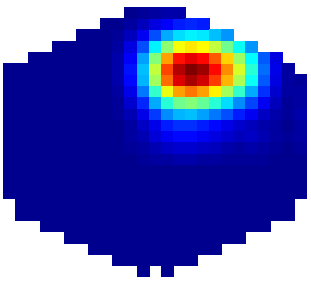


Trial 11 estimate of integrated network function, model 6

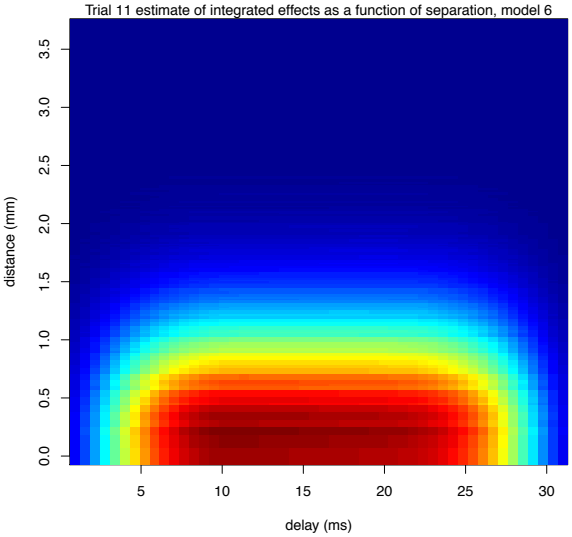




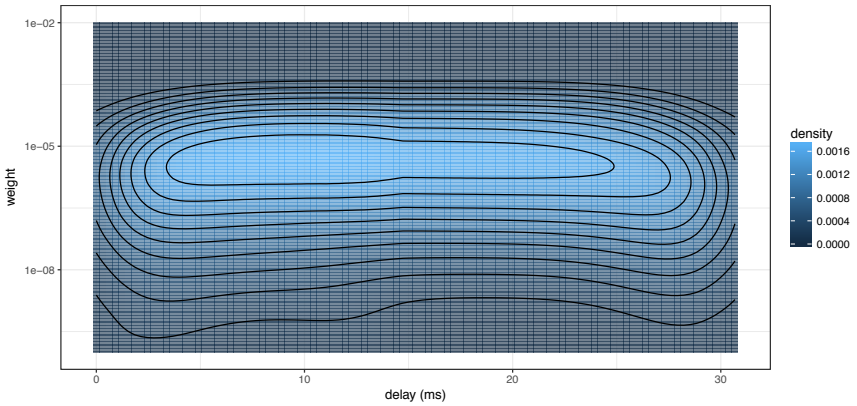




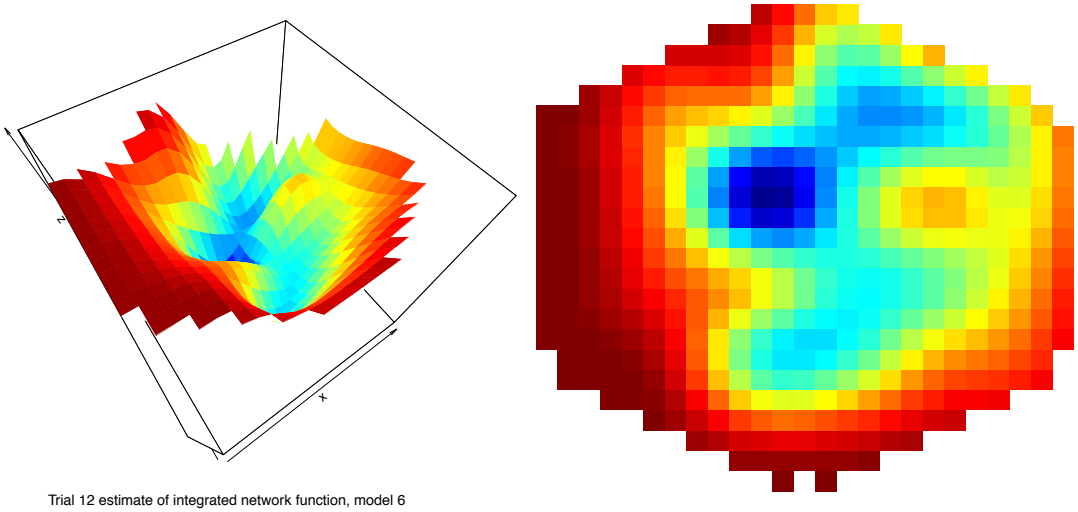
Trial 11 estimate of integrated effects as a function of separation, model 6



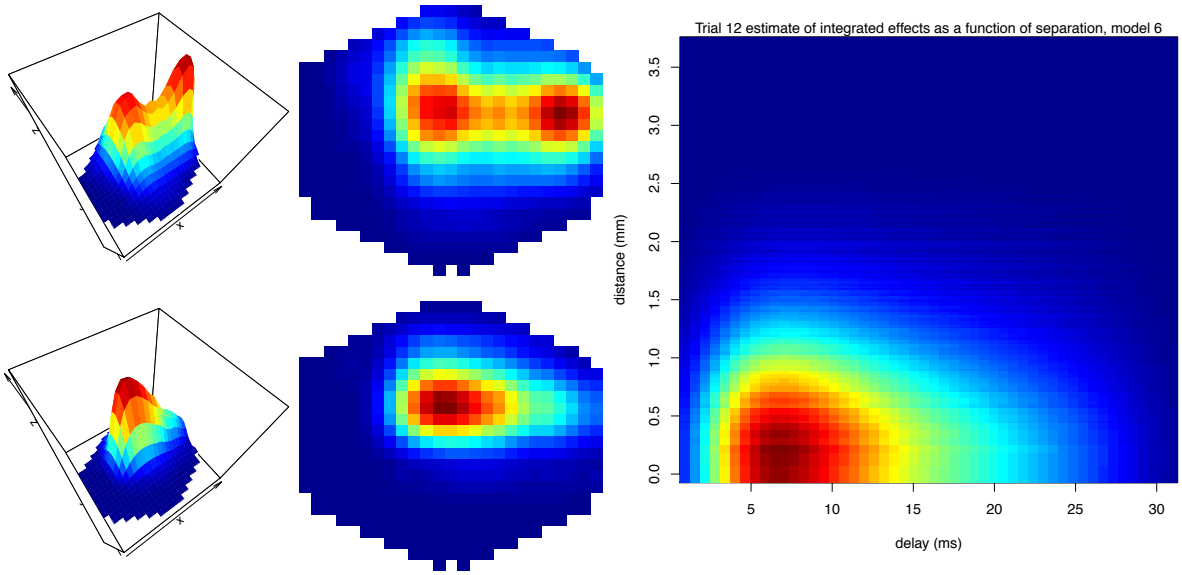
Density plot of estimated weight values for trial 11 model 6



Estimated stimulus surface 70 ms after inititation, trial 12 model 6



Trial 12 estimate of integrated network function, model 6



Density plot of estimated weight values for trial 12 model 6