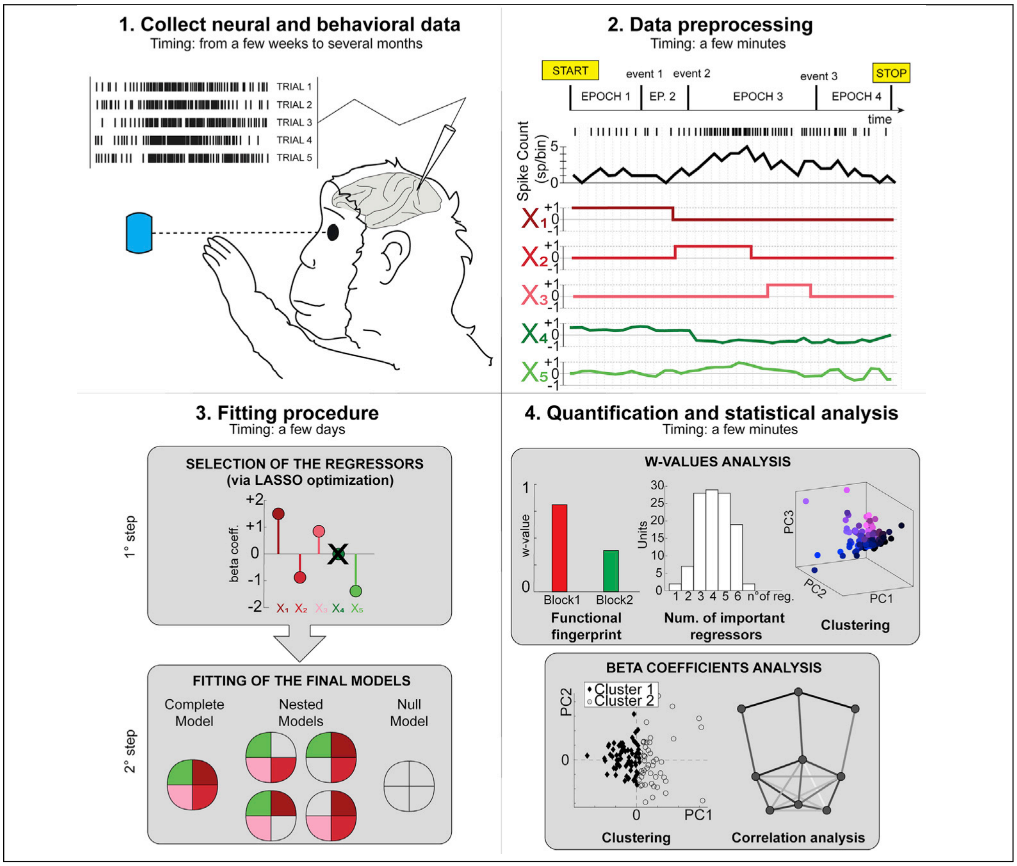
STAR Protocols

CellPress
OPEN ACCESS

Protocol

A Poisson generalized linear model application to disentangle the effects of various parameters on neurophysiological discharges



The protocol provides an extensive guide to apply the generalized linear model framework to neurophysiological recordings. This flexible technique can be adapted to test and quantify the contributions of many different parameters (e.g., kinematics, target position, choice, reward) on neural activity. To weight the influence of each parameter, we developed an intuitive metric ("w-value") that can be used to build a "functional fingerprint" characteristic for each neuron. We also provide suggestions to extract complementary useful information from the method.

Francesco Edoardo Vaccari, Stefano Diomedi, Matteo Filippini, Claudio Galletti, Patrizia Fattori

filippini.matteo7@unibo.it (M.F.)
patrizia.fattori@unibo.it (P.F.)

Highlights

GLM applied to neuronal discharges reveals parameters that modulate their activity

Applying regularizers helps to discard minority parameters and reduces noise

Each neuron can be characterized by the weight assigned to each parameter tested

Results are well suited for population analyses (i.e., clustering, correlation, etc)

Vaccari et al., STAR Protocols *2*, 100413
June 18, 2021 © 2021 The Authors.
https://doi.org/10.1016/j.xpro.2021.100413

Protocol

A Poisson generalized linear model application to disentangle the effects of various parameters on neurophysiological discharges

Francesco Edoardo Vaccari,[1,2,3] Stefano Diomedi,[1,2] Matteo Filippini,[1,4,*] Claudio Galletti,[1] and Patrizia Fattori[1,*]

[1] Department of Biomedical and Neuromotor Sciences, University of Bologna, Bologna 40126, Italy

[2] These authors contributed equally

[3] Technical contact

[4] Lead contact

*Correspondence: filippini.matteo7@unibo.it (M.F.), patrizia.fattori@unibo.it (P.F.)
https://doi.org/10.1016/j.xpro.2021.100413

SUMMARY

The protocol provides an extensive guide to apply the generalized linear model framework to neurophysiological recordings. This flexible technique can be adapted to test and quantify the contributions of many different parameters (e.g., kinematics, target position, choice, reward) on neural activity. To weight the influence of each parameter, we developed an intuitive metric (''w-value'') that can be used to build a ''functional fingerprint'' characteristic for each neuron. We also provide suggestions to extract complementary useful information from the method.

For complete details on the use and execution of this protocol, please refer to Diomedi et al. (2020).

BEFORE YOU BEGIN

Collect neural data

Timing: weeks – months (highly depending on the task and purpose of the study)

The protocol assumes the availability: i) of a dataset describing the spiking activity of a population of neurons recorded during the execution of a given task, ii) descriptors of observable behavior or parameters that change during the execution of the task (Figure 1, left and center). If the dataset is not available, before starting the proposed methodology, the next steps need to be completed:

1. Training of the experimental animals.
2. Implantation of the electrodes array – recording chamber in the area of interest.
3. Neural recordings during task execution.
4. Identification of single units for each recording channel (spike sorting).

Note that the analyses in this protocol do not require necessary simultaneous recordings since this approach seeks to explain the activity of each single cell with the recorded parameters. If simultaneous recordings from more units are available, the model can be extended to account also for cross-correlations between cells of the same population. These additional neural data should be included in the regressors matrix (see below), after being eventually passed through the same filters as the cell past spiking activity (see paragraph 'data preprocessing').

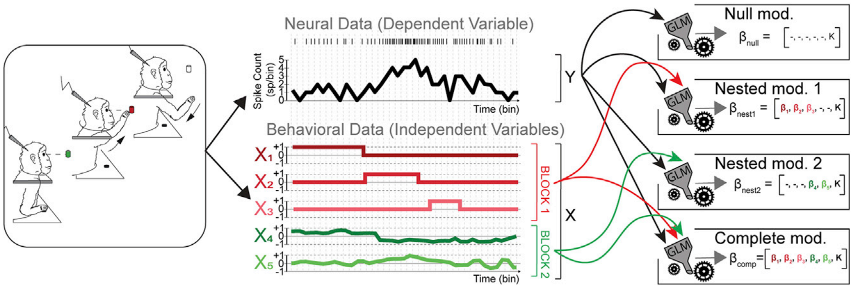

Neural Data (Dependent Variable)
Spike Count (sp/bin)
5
4
3
2
1
0
Time (bin)
Y
Behavioral Data (Independent Variables)
X_1
X_2
X_3
X_4
X_5
+1
0
-1
Time (bin)
BLOCK 1
BLOCK 2
X
GLM
Null mod.
$\beta_{null} = [-,-,-,-,-,K]$
Nested mod. 1
$\beta_{nest1} = [\beta_1, \beta_2, \beta_3, -, -, K]$
Nested mod. 2
$\beta_{nest2} = [-,-,-, \beta_4, \beta_5, K]$
Complete mod.
$\beta_{comp} = [\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, K]$

Figure 1. Workflow schematic representation

Left: experimental session during which neural data are recorded simultaneously to behavioral data. Center: data are pre-processed (binning, normalization) to build the vector of spike-count Y and the regressor matrix X; regressors can be in form of dummy (e.g., X_{1-3} in the figure) or continuous (e.g., X_{4-5}) variables and, regardless of their type, variables with similar meaning can be grouped together (e.g., in the scheme, X_{1-3} and X_{4-5} are grouped in block1 and block2, respectively). Right: several models are fitted with the data (the null model based only on the neural data, the nested models with certain block of regressors removed and the complete model) and beta coefficients are estimated. Note that in the scheme, the LASSO feature selection is omitted for the sake of simplicity.

In Diomedi et al., 2020, two macaque monkeys performed a delayed fix-to-reach task toward different spatial positions. Temporal markers describing behavior, gaze position and spiking activity from neurons of the posterior parietal cortex (area V6A) were collected during task execution. The protocol can be applied to neural data collected from other animal species with no need for modifications.

KEY RESOURCES TABLE

RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
MATLAB® v. R2019a	MathWorks Inc.	https://www.mathworks.com/products/matlab.html
Statistics and Machine Learning Toolbox™	MathWorks Inc.	https://www.mathworks.com/products/statistics.html

MATERIALS AND EQUIPMENT

System requirements: We recommend high CPU clock rate and a large available RAM (at least 8 Gb). In Diomedi et al. (2020), we used a machine with Intel Core i5 1.60 GHz and 8 Gb RAM.

> *Note:* The timing indications provided in this protocol refer only to the computations and they are quite variable, depending on the amount of data (number of units and length of the recordings) and on the complexity of the model that need to be fitted. For example, in Diomedi et al. (2020) the model included 10^4 datapoints and almost 150 regressors. With the system requirements already cited (see above), it took 30–60 min / neuron to complete the fitting procedure. The other steps of this protocol (data preprocessing and final statistical analysis) took a few minutes each. Conversely, coding time highly depends on the ability of the user, so we did not mention it.

STEP-BY-STEP METHOD DETAILS

Data preprocessing

> **Timing:** A few minutes (with the computer used in Diomedi et al., 2020, see 'materials and equipment')

Generalized linear models (GLMs) are a flexible generalization of ordinary linear regression used for dependent variables that have a distribution other than Gaussian. Indeed, Poisson distribution is the most used for modelling the number of spikes that a neuron generate in a brief time interval (bin) (Triplett and Goodhill, 2019; Pillow et al., 2008; Truccolo et al., 2005; Paninski, 2004b; Dayan and Abbott, 2001). Generally speaking, fitting a Poisson GLM requires a dependent variable of which variations can be explained by the variations of a set of independent variables (hereafter called 'regressors'). Thus, in our application, before fitting the models, data must be pre-processed to create the vector of spike counts Y (the dependent variable, vector of size [total N° of bins × 1]) and the regressors matrix X (matrix of size [total N° of bins × N° of regressors]). X and Y must have the same number of rows to associate the spike count in each bin to the corresponding values of the regressors.

Depending on the aims of the study and the task, the independent variables can be both dummy (i.e., with 0 or 1 values only) or continuous (for more details, see 'Note' of this paragraph; Figure 1, central part). A huge variety of different features can be included in the GLM, based on the inputs processed by the brain area of interest. Indeed, this method has been applied to the visual domain (e.g., Pillow et al., 2008), to the motor domain (e.g., Goodman et al., 2019; Takahashi et al., 2017; Paniski et al., 2004a, 2004b) using as regressors kinematics, joint apertures and forces and also to the decision making (e.g., Park et al., 2014). Each application involves different independent variables and consequently a different construction of the X matrix. We suggest to carefully review the literature for each domain since it is not possible to provide here all the details. We will spend just a few words on filtering the variables. Whereas it is fundamental to pass the visual stimuli (i.e., images) through filters (the most used are Gabor-like that simulate the receptive fields of the neurons in the early visual cortices) and fit the neural activity with the result of this convolution (Liu et al., 2016), this step is not necessary for models that involve kinematics or motor parameters in general. GLMs with both filtered and non-filtered (Goodman et al., 2019; Takahashi et al., 2017) features have been used. When filters are applied, the most common are raised cosine functions (Pillow et al., 2008; Truccolo et al., 2010), but also simple sine and cosine functions (Truccolo et al., 2005).

It is recommendable to include in the model also the cell previous spike activity that can account for internal computations not directly linked with external, measured variables. The spike history is usually included at different time lags, both filtered with raised cosine functions (Pillow et al., 2008) or not (Diomedi et al., 2020).

1. Choose the time window of interest (or the entire trial). In the case that neurons have not been recorded simultaneously, align each cell neural activity on the timing of an event of reference.

   Note: the analysis can focus on a fixed time window around the event of alignment or in variable period on a trial basis between two behavioral events. In the latter case, since the time duration could be variable between repetitions, each trial will result in a different number of bins. For the sake of simplicity, in Diomedi et al. (2020), we chose a fixed time interval (from 3000 ms before movement onset to 1720 ms after it, to get an integer number of bins, see below).

2. Choose an appropriate bin width depending on the dynamics of the processes of interest. The choice of the best bin width depends on the focus of the study. To capture fine temporal dynamics such as cell refractory period and correlation between neurons, bin width in the order of a few ms should be used (as small as 1–2 ms, Pillow et al., 2008). For 'slower' processes (or with an uncertain temporal variability), such as kinematics encoding, wider bin widths have been used (40 ms, Diomedi et al., 2020; 20 ms, Goodman et al., 2019; 50 ms, Hatsopoulos et al., 2007).

3. For each trial:
   a. Bin the spike trains within the chosen time frame with the chosen bin width to obtain the spike count Y vector (recommended: *histcounts* MATLAB function).

b. Average each independent variable (e.g., the eye tracks, kinematics data …) within the same bin edges that the Y vector (recommended: *histcounts* MATLAB function).

Note: the spike count of the cell (or even of other units, if recorded in parallel, Truccolo et al., 2010) with different time lags can be included among the independent variables, thus adding information about spike history.

c. If needed, build the vectors for the dummy independent variables: for each bin of the Y vector, assign 1 when a particular condition is met, otherwise 0. See the note below for an example. In Diomedi et al. (2020), we built a number of dummy variables that contained information about the different task phases (behavioral events): planning, movement, holding phases, etc., toward specific targets.

4. Concatenate tip-to-tail the spike counts of all trials to obtain a unique Y vector (size: [total N° of bins × 1]) that contains the cell activity. Concatenate also the independent variables vectors and pool them together in a unique X matrix [total N° of bins × N° of regressors].

Note: To avoid the inclusion in the model of certain continuous variables, it is possible to discretize them in fixed intervals and transform them into a set of dummy variables. This procedure is useful especially in such contexts in which the proper encoding of some variables is unknown. For example, let's consider a variable A to be included in the model because it is thought to modulate neural activity. It can be a regressor as it is (A), but also, for example, squared (A^2) or filtered with a set of Gaussian kernels (varying mean and sigma) producing profoundly different effects on GLM fitting. When understanding the precise type of encoding of the variable is not the focus of the work, the discretization in a bunch of dummy variables can be a solution to avoid annoying, complex data elaboration. This approach has some drawbacks: first, as already mentioned, it does not provide information about the type of encoding; second, it can critically enlarge the dimensionality of the model. For example, if the variable A (range [0 10]) is discretized in dummy variables using fixed intervals with unitary width (i.e., the first will take 1 when A is in the range [0 1), the second will take 1 when A is between [1 2) and so on), we will end up with a set of 10 dummy variables that represent A. To give an experimental example, in Diomedi et al. (2020), we discretized the gaze position version, elevation and vergence in a 3D-grid associating to each little spatial volume a dummy variable that took the value 1 in every bin in which the animal fixates in it (otherwise, 0). In conclusion, discretization of continuous variables can help in such situations in which the type of encoding is unknown and out of the scope of the work, but it should be applied after a careful evaluation of pros and contra.

⚠ CRITICAL:

- Once the X matrix has been built, it is highly recommendable to standardize (calculating the z-score, i.e., first subtracting the mean and then dividing for the standard deviation; Bring, 1994) the continuous variables to get beta coefficients directly comparable. This step is fundamental when one wants to further study the beta coefficients estimated during the fitting, especially when the regressors have different scales and/or are expressed with different units of measurement. Note that dummy variables will be 0 or 1 by definition and unitless, so they can be introduced in the model without further standardization. It would not make much sense to get a beta coefficient that refers to 'standard deviation' increments (or decrements) of a dummy variable.
- Be careful in adding new independent variables since if the sample size (i.e., the number of bins) is too small respect to the number of regressors, the fitting can lead to a poor estimation and unreliable beta coefficients. Although there is not a rule that always applies, you can use the 'one in ten rule', a rule of thumb which states that the maximal number of regressors in a model is equal to the

number of observations (bins) / 10 (Steyeberg and Harrel, 2004; Peduzzi et al., 1996; Harrel et al., 1996).

Fitting procedure

Timing: A few days (with the computer used in Diomedi et al., 2020, **see** '**materials and equipment**'; **highly depending on the number of cells and complexity of the model, see the note in** '**materials and equipment**' **section**)

The direct comparisons between beta coefficients in non-linear models are usually not straight-forward and often discouraged since many calculations on them are not statistically correct (for example, when they are input of an exp function, as in this case; see the equation in the paragraph below). For this reason, we suggest grouping the regressors with similar 'meaning' (e.g., the variables that indicate the 3D position of gaze in terms of version, elevation and vergence angles or x, y and z; the variables that encode kinematics; the variables that contain information about the decision…) in 'blocks'. After the fitting of the complete model, each block of regressors will be removed in turn to evaluate its influence in terms of goodness-of-fit (see below; Figure 1, right).

In this protocol, the fitting procedure consists of two stages: 1) selection of most influential regressors via LASSO optimization; 2) GLM fitting of the complete and nested models considering only the regressors previously selected. During this latter stage, the models are not LASSO regularized to get a value of the goodness of fit that is not penalized by LASSO additional term. Moreover, the protocol requires that blocks of regressors are removed 'manually' starting from the complete model to build the nested models, whereas if LASSO was used in this step, there would be the possibility that other regressors would be removed 'automatically' (by LASSO) besides those removed 'manually' in an uncontrollable way.

Note: The LASSO optimization introduces a penalization term during the fitting procedure that shrinks the beta estimate values and sets the less influent to 0. The weight of this new term is represented by the hyper-parameter λ and it can be adjusted to avoid overfitting and/or to handle a lower number of selected variables (see Figure 2). This optimization is commonly used when a model includes many independent variables and their correlation with the dependent variable is not known *a priori*.

The fitting procedure includes the following steps:

5. Fit a cross-validated (10-fold) LASSO GLM with Poisson link function to explain the spike count in Y with the regressors in X (in MATLAB: *betas = lassoglm (X, Y, 'poisson', 'CV', 10)*). Calling the MATLAB *lassoglm* function with these inputs, it will automatically vary the shrinking parameter λ to individuate the value that returns the minimal cross-validated deviance of the model (see Figure 2).
6. Remove from the X matrix the independent variables (columns) that correspond to the zero beta coefficients assigned during LASSO fitting (take the betas of the model with the λ that minimizes the deviance).
7. Fit the complete GLM with Poisson canonical link function (no LASSO regularized; *fitglm* MATLAB function) to explain the spike count in Y with all the remaining regressors in X after the LASSO selection. For each block of variables, fit a nested model (*fitglm* MATLAB function) with all the remaining regressors in X (non-zero LASSO betas) except those belonging to that block. To avoid overfitting, an additional cross-validation (k-fold or leave-one-out) can be performed during this step, training on a part of the dataset and testing on the other part (see the Note below).

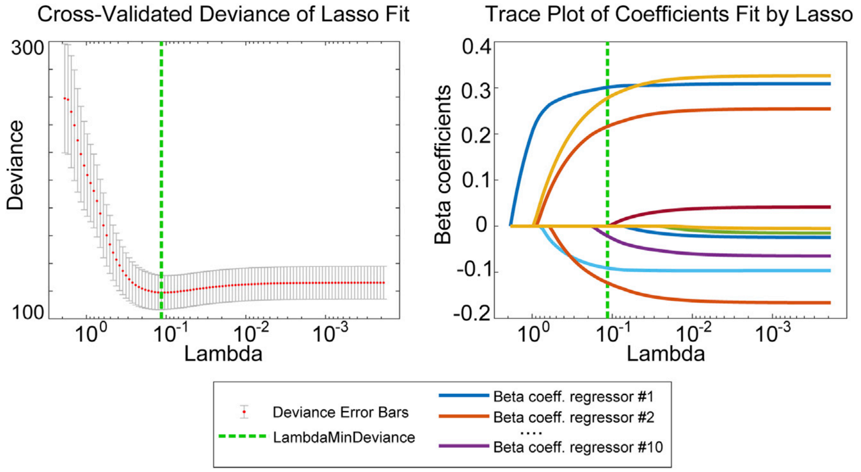


Figure 2. Example of feature selection using LASSO optimization method on randomly generated data

Left: Cross-validated deviance of LASSO fit models as a function of the shrinking parameter lambda (λ). The green dashed line corresponds to the lambda that produce the minimum deviance. Lower lambdas (on the right of the green line) produce a model with too many parameters that suffer of overfitting. Higher lambdas (on the left of the green line) tend to shrink too much the model removing important regressors. Right: Beta coefficients fitted by LASSO as a function of λ. Each coloured line represents the value of the coefficient corresponding to a regressor in the model. The green dashed line corresponds to the lambda that produce the minimum deviance, thus the optimal beta coefficients. Lower lambdas (on the right of the green line) produce a model that retains more features (fewer beta coefficients have a 0 estimate). With higher lambdas (on the left of the green line) important features are removed from the model, greatly worsening the fit.

8. Fit also the 'null' model that includes no regressors at all. The goodness-of-fit of this last model will be used as reference value (*fitglm* with (real) Y and a vector of 1s of the same length, as an X placeholder). If the cross-validation was performed during step 7, it must be used here as well to fit the 'null' model on the same parts of the dataset.

As use case, in Diomedi et al. (2020), we grouped the regressors in 'extrinsic' blocks (namely EYE POSITION, EYE SPEED/DIR, POSTSACC, DELAY, PREP, PREMOV, MOV, HOLD, PREMOV2, MOV2) that carried information about task phases (and so, likely the ongoing corresponding neural processes); and the 'intrinsic' SPIKE HISTORY block that carried information about the previous cell spiking activity. We thus obtained i) 1 complete model that included all the regressors blocks, ii) 10 nested models removing a different extrinsic block for each run, iii) 1 'only extrinsic' model using all the extrinsic blocks but not the SPIKE HISTORY, iv) 1 'only intrinsic' model removing all the extrinsic blocks (i.e., X matrix consisted only of the variables in SPIKE HISTORY) and v) the 'null' model.

Prior to further evaluation (see next section) of the fitting, it is possible to assess the appropriateness of the model at glance by plotting an estimate of the firing rate (*predict* MATLAB function) vs the real spike rate. Alternatively, the firing rate at time t predicted by the model can be calculated as the exponential of the linear combination of the regressors:

$$\mu_t = \exp(\beta_0 + \beta_1 X_{1,t} + \ldots + \beta_K X_{K,t}) \qquad \text{(Equation 1)}$$

where K is the number of regressors, $\{\beta_k\}_{k=1,\ldots K}$ are the beta coefficients, $X_{k,t}$ is the value of kth variable at time t and β_0 is the intercept of the model. Figure 3 (left) shows an example of recorded (black line) vs estimated (red line) firing rate of a parietal neuron during a reaching task. The predicted activity closely matches the observed, peaking just after movement onset (time: 0 s) and resulting slightly inhibited respect to baseline during the hold phase.

Note: Apply this procedure separately for each cell data.

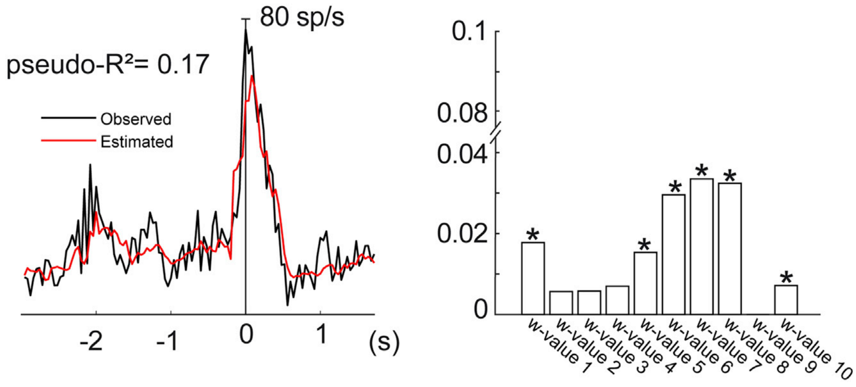


Figure 3. Neural data recorded from a parietal neuron during a fix-to-reach task in darkness

Left: Observed (black line) and estimated (red line) firing rates (non-overlapping 40 ms bins). In the plot, data are cross-validated (for the estimation data never seen by the model are used) and averaged across 10 trials. Right: Bars show the weights (w-values) of the 10 blocks of regressors (the 'functional fingerprint' of the neuron) on the neural activity. Asterisks indicate most important w-values for each cell. Adapted from Diomedi et al., 2020.

Note: For non-LASSO regularized models, the cross-validation strategy is not necessary. When dealing with highly variable data such as neural recordings, the approach can be useful to get more robust estimates. In this protocol, we suggest averaging cross-validated models that revealed to be the simplest, but valid solution to finally handle unique values (Zhang and Zou, 2020; Jung and Hu, 2015).

Statistical analysis

⏱ **Timing: A few minutes (with the computer used in Diomedi et al., 2020, see above)**

After the fitting procedure during which we estimated the beta coefficients for all the models detailed above, it is possible to extract information in different ways from the models depending on the aims of the study. Our statistical analyses focused on the goodness-of-fit (GOF) of the complete vs the nested models to get insights about the underlying neural modulations. We will also give some advices on how to treat the beta coefficients of the complete model to get more complementary information.

9. Goodness-of-fit: get the w-values. In the context of non-linear models, the goodness-of-fit is measured as likelihood (or its logarithm). The log-likelihood ℓ, easier to compute, is used by the algorithms to estimate the parameters (betas) through a procedure called Maximum Likelihood Estimation (MLE). In our case, the likelihood is the probability, given a model, to observe a given spike train thus it ranges from 0 to 1, while its logarithm (the log-likelihood) ranges from $-\infty$ to 0.
   a. Assess the GOF of all the models. It can be done in two ways:
      i. the ℓ value is directly provided by the fitting function (in MATLAB; from the GeneralizedLinearModel object in MATLAB obtained with *fitglm* function.
      ii. by calculation with the following formula, remembering that the firing rate depends on the beta coefficients β (see Equation 1):

$$\ell(y,\beta) = \sum_{t=1}^{T} y_t \log(\mu_t) + \sum_{t=1}^{T} y_t \log(\Delta) - \sum_{t=1}^{T} \log(y_t!) - \Delta \sum_{t=1}^{T} \mu_t \qquad \text{(Equation 2)}$$

where y is the spike count, μ is the firing rate predicted by the model (see above), Δ is the bin width, t is the bin number and T the total number of bins.

b. For higher interpretability, calculate McFadden's pseudo-R^2 (Cameron and Windmeijer, 1997):

$$R^2_{\text{pseudo}} = 1 - \frac{\ell_{\text{complete}}}{\ell_{\text{null}}} \qquad \text{(Equation 3)}$$

Starting from the log-likelihood of the complete (ℓ_{complete}) and null (ℓ_{null}) models.ℓ_{null} represented the fitting of the simplest possible model and, by definition, it is independent from every regressor (the model includes only a constant term). McFadden's pseudo-R^2 can be interpreted as the more common R^2 in ordinary linear regression, ranging from 0 (extremely poor fit) to 1 (perfect fit), but it tends to be remarkably lower (values of 0.2 to 0.4 are considered excellent fit).

c. Select only the units with a R^2_{pseudo}>*threshold* to discard noisier cells for which the model failed to capture neural modulations. In Diomedi et al. (2020), we set the *threshold* at 0.05, as in previous works (Goodman et al., 2019; Paninski et al., 2004a).

d. For each nested model, compute a relative pseudo-R^2 as:

$$R^2_{\text{relativepseudo}} = \frac{\ell_{\text{nested}} - \ell_{\text{null}}}{\ell_{complete} - \ell_{\text{null}}} \qquad \text{(Equation 4)}$$

where ℓ_{nested} is the log-likelihood of the nested model.

This value compares the log-likelihood (i.e., the goodness-of-fit) of each nested model with the complete model (and the null model).

e. Convert the relative pseudo-R^2 in a weight for each nested model:

$$w - value = 1 - R^2_{relativepseudo} \qquad \text{(Equation 5)}$$

This score is directly associated with the importance of the group (block) of variables removed to build the nested model with respect to the complete model. Whether a block of regressors contained important information for the model, its removal causes a great worsening of the fit, the relative pseudo-R^2 will decrease (towards 0) and the w-value will increase (towards 1). Vice versa, whether a regressors' block had little influence on the complete model, its removal will cause a little worsening of the fit, an increase (towards 1) in the relative pseudo-R^2 resulting finally in a low w-value (towards 0). The Figure 3 (right) shows an example 'functional fingerprint' of a parietal neuron composed by 10 different w-values. The w-values marked with the asterisks are important to describe the neural modulations (computed as described in the next section, point B.). For more details about the 10 w-values meaning, please see Diomedi et al., 2020.

⚠ CRITICAL: If the complete, nested and null models have been cross-validated during the fitting (steps 7 and 8 in 'fitting procedure' section), average the log-likelihoods across the different testing partitions of the data to compute all the scores in this section.

10. Goodness-of-fit and analysis of the w-values, a few suggestions. The set of w-values describes a 'functional fingerprint' characteristic for each cell and summarizes the neural modulations elicited by the entire blocks of regressors. The 'functional fingerprints' can be further analyzed to investigate the relative weights of the groups of variables on the population, the presence of specialized subpopulations of cells, the dynamics of the encoding…

    Here we provide a few suggestions following the analyses in Diomedi et al. (2020), but the w-values can be hypothetically treated with many other approaches.

    a. It is possible to compare directly the distributions of the w-values across the neural population to investigate the relevance of each block of variables on the spiking activity. We suggest the use of the median values and non-parametric tests such as Wilcoxon's to evaluate the significance of the observed differences (see Figure 4A for an application).

    b. It is also possible to assess which regressor blocks significantly influence each cell activity.

       i. For each cell, sum all its w-values.

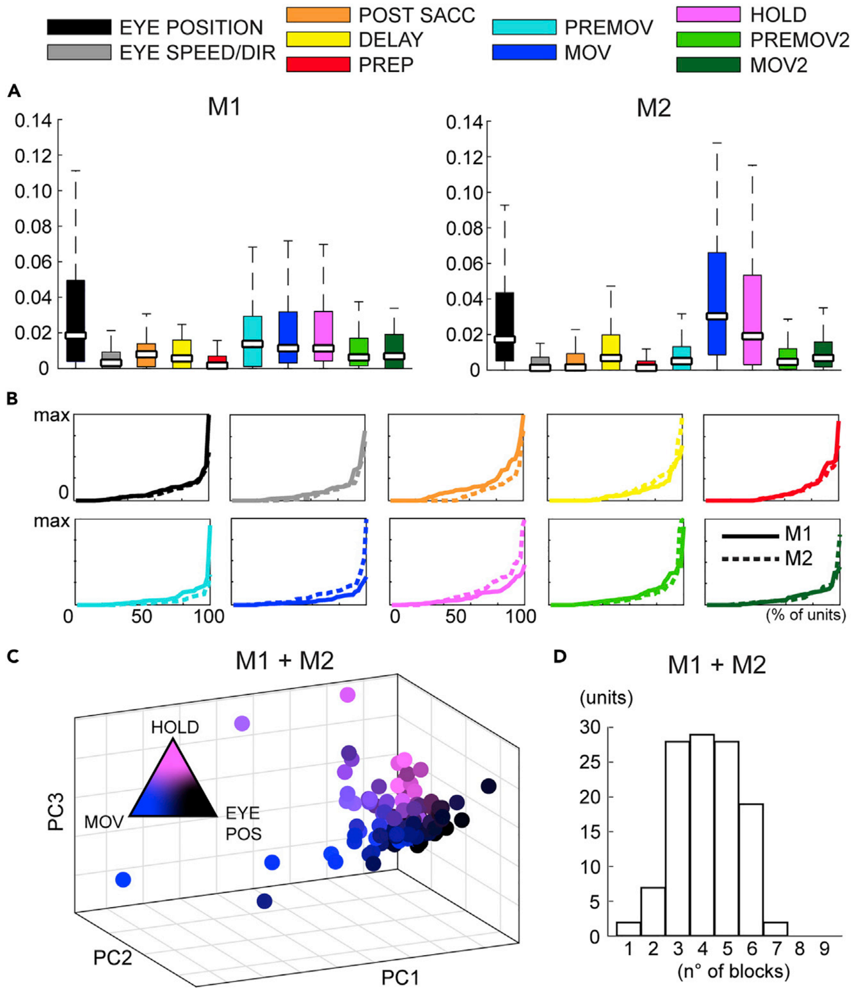


Figure 4. Figure and data adapted from Diomedi et al. (2020) showing the w-values across the population

(A) Box plot of w-values for each block of regressors across the population (2 animals separated).

(B) The w-values are plotted in ascending order for each block of regressors.

(C) The neural population (a dot for each cell) is projected onto the 3 first principal components (PCs) of the principal component analysis (PCA) performed on the 10 w-values.

(D) Histogram showing the minimum number of regressors' blocks (w-values) necessary for each cell to reach at least the 85% of its extrinsic w-values total sum (i.e., the blocks of regressors with a significant effect on cell modulations).

ii. Iteratively, add together the w-values in descending order up to reach the 85% of the total sum. The blocks needed to achieve this value are considered significant in modulating cell activity (see Figure 4D).

c. The 'functional fingerprints' describe single cell activity patterns. However, it might be interesting to move towards a population analysis starting from the compact view provided by the 'functional fingerprints' and to seek functionally specialized sub-populations. This can be done in a few steps:

i. Visualize the N-dimensional data points (N = number of w-values) in a 2D or 3D space performing a Principal Component (PC) Analysis on the matrix [N° of units × N] and plotting the projections of the neurons on the first PCs (2 or 3).

ii. It is possible to apply standard clustering algorithms (e.g., K-means or hierarchical clustering) on the functional fingerprints and identify clusters of units with shared activity patterns.

Figure 4C shows the functional structure of our neural population (Diomedi et al., 2020) that was characterized by the lack of units clustered according to their functional fingerprints.

11. Beta coefficients analysis: a few suggestions. Beta coefficients carry information about the effect of each single regressor included in the model on neural activity. However, since the non-linearity of the Poisson GLMs (an additive change in the predictors has a multiplicative effect on the response, see Equation 1) and the normalization needed to compare the betas, usually it is not recommended to interpret directly these regression coefficients. Anyway, we here report a couple of indirect analyses that can be performed on beta coefficients of the complete GLM to extract additional information (see Diomedi et al., 2020).
    a. Correlation analysis to investigate the encoding of variables in the population:
        i. For each variable of interest, build a beta vector that represents the population response to that variable extracting from the complete model of each unit the beta value corresponding to the variable. Mathematically, the beta vector for the kth variable will be $\{\beta_{c,k}\}_{c=1,\ ...M}$ where M is the number of the units in the population.
        ii. Compute the correlation coefficient r between beta vectors (Zhang et al., 2017). We recommend using Spearman's rank correlation that is best suited to deal with the non-linearity of the modulations rather than standard Pearson coefficient.

Note: high correlation coefficients mean high similarity in the population response to the two tested variables.

    b. Clustering: similarly to what suggested for the functional fingerprints, it is possible to run standard clustering algorithms on the beta coefficients in order to eventually identify sub-populations that process information in different ways. For example, in Diomedi et al. (2020), we found two separate clusters within our neural population that were differently influenced by their own previous spiking activity.

EXPECTED OUTCOMES

Whereas the functional expected outcomes of this protocol totally depend on the purpose of its application and the data considered, here we can make a few considerations from a statistical point of view.

LASSO regularization was found to be well suited to handle models with many variables and to select only the most important to approximate the neural response. For example, in Diomedi et al. (2020), this optimization retained on average 90 out of about 150 independent variables across the population. Similarly, in Goodman et al. (2019), a number between 55 and 95 of LASSO selected features were reported to approximate the activity of neurons (recorded in M1 and somatosensory areas during reaching and grasping tasks).

During the evaluation of the model goodness of fit, one can expect a removal of 10%–40% of the neurons initially included in the population. (10%–30%, Goodman et al., 2019; 22%–44%, Diomedi et al., 2020) if a threshold of 0.05 is applied to the pseudo-R^2 values (see step 9 in 'statistical analysis' section). Moreover, an expected pseudo-R^2 distribution across the population can be low as 0.127 - 0.185 (mean; Goodman et al., 2019) or 0.102 [0.073, 0.147] (median [25th, 75th percentile]; Diomedi et al., 2020), which are substantially lower than the normal R^2.

LIMITATIONS

Note that the GLM application we here propose is an extremely versatile statistical tool to investigate neural dynamics with a single-cell approach. This technique *per se* is almost free of major drawbacks, but a crucial point is the choice of the features to include or not in the model and their preprocessing.

Imagine a model that aims to explain the firing rate based on the x-position of a spatial target. The values of x could be set positive increasing towards right, negative decreasing towards left. If x is directly used to fit the neural activity, we are implicitly forcing the model to choose between a right-selective (with a positive beta: the activity will grow as the target moves further to the right that means higher values of x), a left-selective (with a negative beta), or no-spatial selective unit (beta around 0). For how the model is built in this case, a preference for central or peripheral space (irrespectively to left or right) cannot be taken into account.

One solution may be to include in the model x squared (or alternatively the absolute value of x) so that the model also considers a preference for the central space (with a positive beta, or inhibition for the central space when beta is negative, see Figure 5).

Thus, both linearities and non-linearities can be considered with an appropriate choice or an appropriate processing of the features as well as different temporal lags with which the regressors are introduced in the model. Note that these steps require an extensive preprocessing of the data. Other methods, such as neural networks can automatically handle high-nonlinear relationships between variables, but with a higher quantity of data needed for the training and a lower interpretability of the resulting models. Tailored approaches can couple the two methods, passing for example some independent variables through the input layers of an autoenconder and using the hidden layer as the input of the GLM.

TROUBLESHOOTING

Problem 1

We suggest removing from the analysis the cells that scored a low R^2_{pseudo} (see step 9c in "statistical analysis" section). Due to this selection many neurons may be discarded massively reducing the population.

Potential solution

The model (i.e., the chosen features) is not appropriate to explain neural activity. Try to add the same variables with different temporal lags (or even with different temporal scale), enrich the model with more regressors if other data are available, manipulate (during a pre-processing phase with square-root, filtering with sine and cosine functions…) the features already included in the model in order to allow more types of neural modulations or convert the regressors in a number of dummy-variables.

Problem 2

Collinearity between regressors could invalidate the model (step 3 in 'data preprocessing' section).

Potential solution

In order to get a reliable estimation of the beta coefficients and avoid a confusing effect on the analysis, the regressors should be as independent from each other as possible. Indeed, even if LASSO regularization is robust to collinearity in the features, it is always a good choice to avoid correlations with a coefficient $|r| > 0.7$ between regressors (Dormann et al., 2013). Whether this would be the case, we suggest removing the less important among the more redundant variables or alternatively aggregate them through some dimensionality reduction technique such as PCA.

Problem 3

Model fitting can require quite a long time. This can happen especially when using computers with limited resources or when trying to fit very complex models both in terms of number of data points and number of variables (steps 5–7 in 'fitting procedure' section).

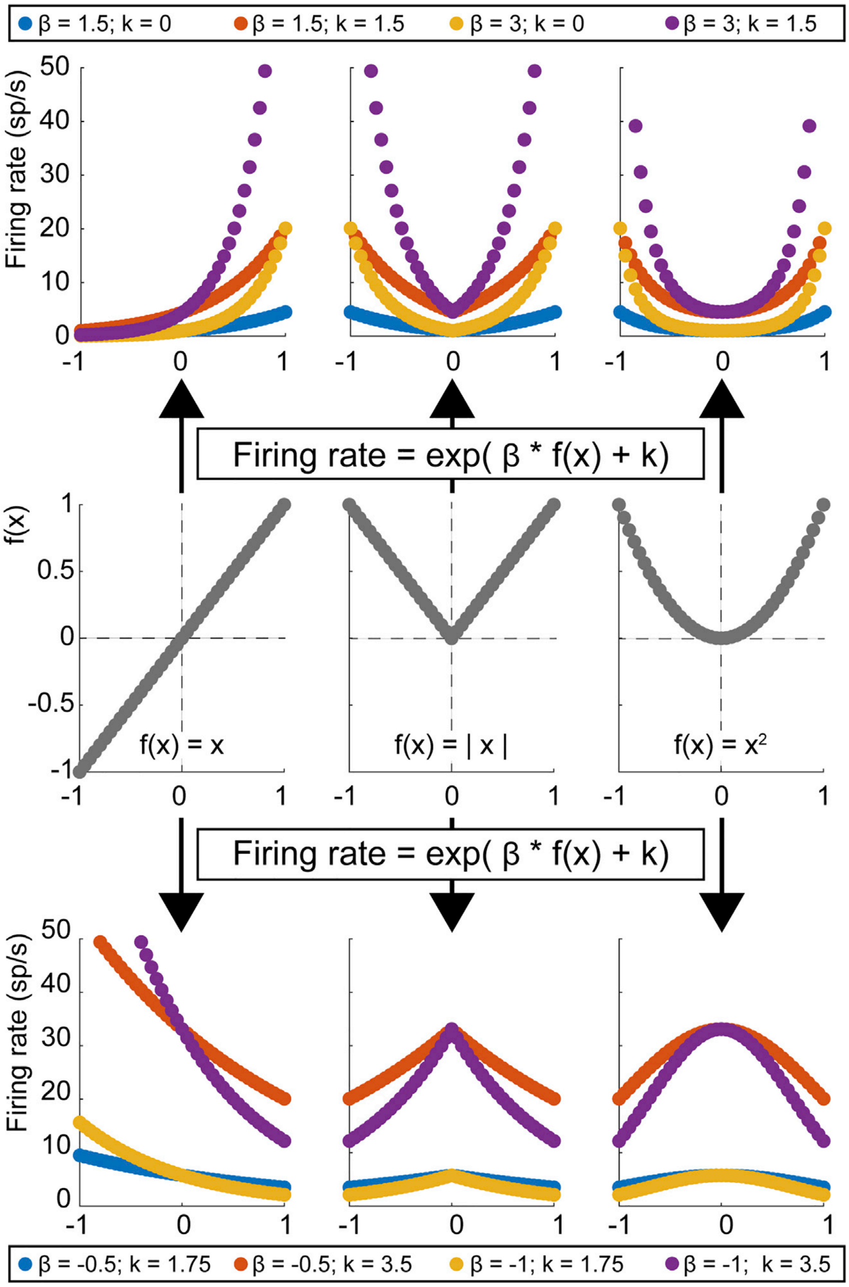


Figure 5. Simulation of the spatial tuning allowed by different pre-processing of a regressor

A variable x (e.g., the horizontal position of a target) in the range [_1 +1] can be directly passed through an exponential function (left panels) to produce firing rates with a preference for right (positive beta coefficients, left-superior panel) or left (negative betas, left-inferior panel) positions. The same x can be pre-processed (absolute value or squared, central and right panels respectively) to produce a preference for lateral (positive beta coefficients, central and right-superior panels) or central (negative betas, central and right-inferior panels) positions. Four tuning curves are obtained for each panel by simply varying the parameters of the exponential function.

Potential solution

To exploit parallel computation and distribute the computational load on all the available cores, a good automatic solution is to set 'UseParallel' option in *lassoglm* MATLAB function (which is highly time consuming) to true to compute in parallel and exploit more cores. When possible, try to parallelize the entire code as much as possible. In MATLAB, for example, it can be done using 'parfor' loops instead of the simple 'for' loops to have many nested GLMs or the GLMs of many units estimated at the same time. Since the script design must be adapted, check the software documentation to get more details.

It is possible also possible to reduce the overall computational load in several ways. For example, one can try to adapt the bin width reducing the total number of bins and reaching a good compromise between time resolution and computational load. To reduce the number of variables, a preselection of important regressors can be performed on a small subset of units to then scale up the process to the entire population retaining only the variables with no 0 beta coefficients (during LASSO step) in a significant percentage of the subset of cells. If discretization of continuous variables has been performed, it is possible to reduce the size of the resulting set of variables through a coarser discretization.

Problem 4

The matrices that contain the data (especially the X matrices with the regressors) can be memory consuming and if more GLMs are handled at the same time, the RAM could be saturated (steps 5–7 in 'fitting procedure' section).

Potential solution

This problem can be easily solved with best coding practices. Write the script to re-use X matrices used to estimate the complete GLM without building a new one for each nested model. Save the results of the computations frequently, delete them from temporary memory and eventually reload them when needed to save RAM. In MATLAB, if a big matrix contains many 0s, we recommend the use of the 'sparse' function that will convert the matrix into its sparse form squeezing out the zero elements and saving memory. The 'full' function, applied to the sparse matrix, will return the original matrix.

RESOURCE AVAILABILITY

Lead contact

Further information and request for resources should be directed to and will be fulfilled by the lead contact, Matteo Filippini (matteo.filippini7@unibo.it).

Materials availability

This study did not generate new unique reagents.

Data and code availability

The dataset and code supporting the current study are available from the lead contact upon reasonable request.

ACKNOWLEDGMENTS

We thank professor Giuliano Galimberti (Department of Statistical Sciences, University of Bologna, Italy) and "Statistics Clinic" service of the University of Bologna. This work was supported by the Ministero dell'Università e della Ricerca (Italy, PRIN2017-2017KZNZLN) and by Fondazione Cassa di Risparmio in Bologna Bando Ricerca 2018/0373.

AUTHOR CONTRIBUTIONS

F.E.V. and S.D. conceived the statistical model and wrote the original draft. M.F. and C.G. supervised the project. P.F. provided funding and facilities. All authors contributed to editing the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Bring, J. (1994). How to standardize regression coefficients. Am. Stat. *48*, 209–213.

Cameron, A.C., and Windmeijer, F.A.G. (1997). An R-squared measure of goodness of fit for some common nonlinear regression models. J. Econom. *77*, 329–342.

Dayan, P., and Abbott, L.F. (2001). Theoretical Neuroscience - Computational and Mathematical Modeling of Neural Systems (The MIT Press).

Diomedi, S., Vaccari, F.E., Filippini, M., Fattori, P., and Galletti, C. (2020). Mixed selectivity in macaque medial parietal cortex during eye-hand reaching. iScience *23*, 10.

Dormann, C.F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., Marquéz, J.R.G., Gruber, B., Lafourcade, B., Leitão, P.J., et al. (2013). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. Ecography *36*, 27–46.

Goodman, J.M., Tabot, G.A., Lee, A.S., Suresh, A.K., Rajan, A.T., Hatsopoulos, N.G., and Bensmania, S. (2019). Postural Representations of the hand in the primate sensorimotor cortex article postural representations of the hand in the primate sensorimotor cortex. Neuron *104*, 1000–1009.e7.

Harrell, F.E., Jr., Lee, K.L., and Mark, D.B. (1996). Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. Stat. Med. *15*, 361–387.

Hatsopoulos, N.G., Xu, Q., and Amit, Y. (2007). Encoding of movement fragments in the motor cortex. J. Neurophysiol. *27*, 5105–5114.

Jung, Y., and Hu, J. (2015). A K-fold averaging cross-validation procedure. J. Nonparametr. Stat. *27*, 167–179.

Liu, L., She, L., Chen, M., Liu, T., Lu, H.D., Dan, Y., and Poo, M.M. (2016). Spatial structure of neuronal receptive field in awake monkey secondary visual cortex (V2). Proc. Natl. Acad. Sci. U S A *113*, 1913–1918.

Paninski, L., Fellows, M.R., Hatsopoulos, N.G., and Donoghue, J.P. (2004a). Spatiotemporal tuning of motor cortical neurons for hand position and velocity. J. Neurophysiol. *91*, 515–532.

Paninski, L. (2004b). Maximum likelihood estimation of cascade point-process neural encoding models. Netw. Comput. Neural Syst. *15*, 243–262.

Park, I.M., Meister, M.L., Huk, A.C., and Pillow, J.W. (2014). Encoding and decoding in parietal cortex during sensorimotor decision-making. Nat. Neurosci. *17*, 1395–1403.

Peduzzi, P., Concato, J., Kemper, E., Holford, T.R., and Feinstein, A.R. (1996). A simulation study of the number of events per variable in logistic regression analysis. J. Clin. Epidemiol. *49*, 1373–1379.

Pillow, J.W., Shlens, J., Paninski, L., Sher, A., Litke, A.M., Chichilnisky, E.J., and Simoncelli, E.P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. Nature *454*, 995–999.

Steyeber, E.W., and Harrel, F.E. (2004). Chapter 8: Statistical models for prognostication: problems with regression models. In "Symptom Research: Methods and Opportunities"; interactive textbook, M. Mitchell and L. Joanne, eds. (National Institutes of Health). https://web.archive.org/web/20041018142600/http://painconsortium.nih.gov/symptomresearch/index.htm.

Takahashi, K., Best, M.D., Huh, N., Brown, K.A., Tobaa, A.A., and Hatsopoulos, N.G. (2017). Encoding of both reaching and grasping kinematics in dorsal and ventral premotor cortices. J. Neurosci. *37*, 1733–1746.

Triplett, M.A., and Goodhill, G.J. (2019). Probabilistic encoding models for multivariate neural data. Front. Neural Circ. *13*.

Truccolo, W., Hochberg, L.R., and Donoghue, J.P. (2010). Collective dynamics in human and monkey sensorimotor cortex: Predicting single neuron spikes. Nat. Neurosci. *13*, 105–111.

Truccolo, W., Eden, U.T., Fellows, M.R., Donoghue, J.P., and Brown, E.N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. J. Neurophysiol. *93*, 1074–1089.

Zhang, C.Y., Aflalo, T., Revechkis, B., Rosario, E.R., Ouellette, D., Pouratian, N., and Andersen, R.A. (2017). Partially mixed selectivity in human posterior parietal association cortex. Neuron *95*, 697–708.e4.

Zhang, H., and Zou, G. (2020). Cross-validation model averaging for generalized functional linear model. Econometrics *8*, 7.