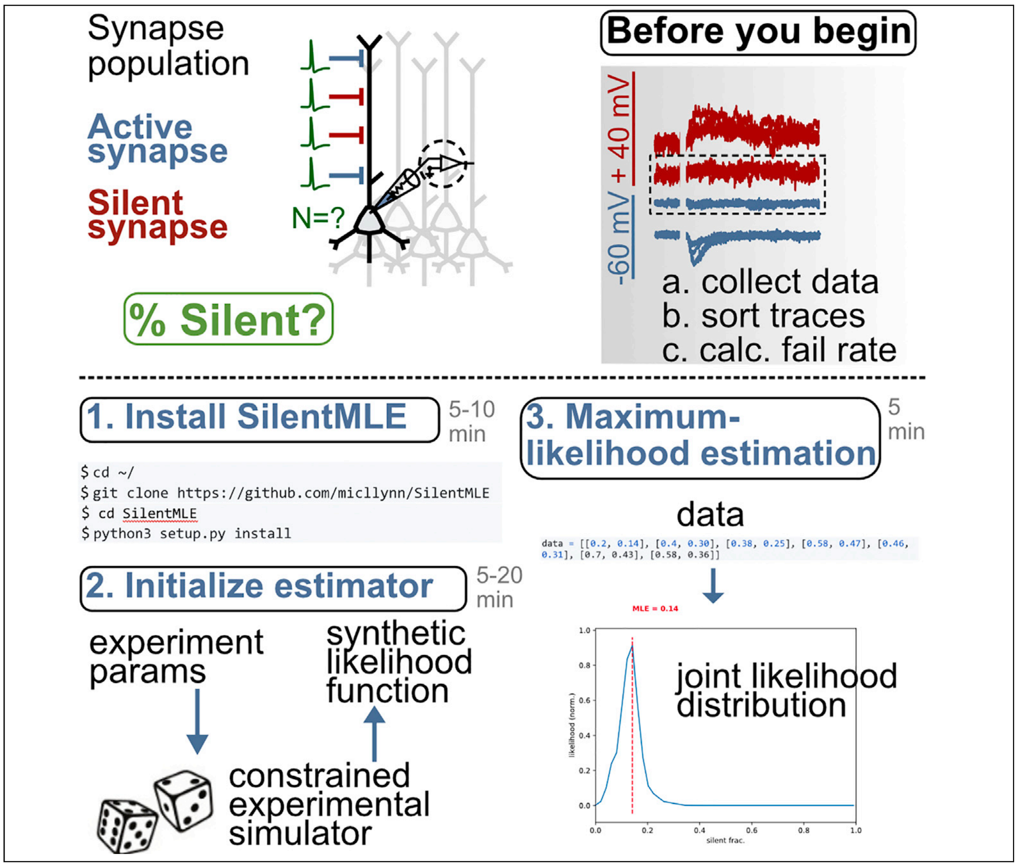
STAR Protocols

Protocol

Accurate Silent Synapse Estimation from Simulator-Corrected Electrophysiological Data Using the SilentMLE Python Package



```
$ cd ~/
$ git clone https://github.com/micllynn/SilentMLE
$ cd SilentMLE
$ python3 setup.py install
```

```
data = [[0.2, 0.14], [0.4, 0.30], [0.38, 0.25], [0.58, 0.47], [0.46,
0.31], [0.7, 0.43], [0.58, 0.36]]
```

The proportion of silent (AMPAR-lacking) synapses is thought to be related to the plasticity potential of neural networks. We created a maximum-likelihood estimator of silent synapse fraction based on simulations of the underlying experimental methodology. Here, we provide a set of guidelines for running a Python package on compatible experimental synaptic data. Compared with traditional failure-rate approaches, this synthetic likelihood estimator improves the validity and accuracy of the estimates of the silent synapse fraction.

Michael Lynn,
Richard Naud,
Jean-Claude Béïque

mlynn101@uottawa.ca

HIGHLIGHTS

Guidelines for acquiring failure-rate data and parsing successes/failures

Simple installation guide for the SilentMLE package

Initializing experimentally constrained estimators of the silent synapse fraction

Expected results and interpretation of estimator output

Lynn et al., STAR Protocols 1, 100176
December 18, 2020 © 2020 The Authors.
https://doi.org/10.1016/j.xpro.2020.100176

Protocol

Accurate Silent Synapse Estimation from Simulator-Corrected Electrophysiological Data Using the SilentMLE Python Package

Michael Lynn,[1,6,7,*] Richard Naud,[1,3,5] and Jean-Claude Béïque[1,2,3,4]

[1]Department of Cellular and Molecular Medicine, University of Ottawa, Ottawa, ON K1H 8M5, Canada

[2]Canadian Partnership for Stroke Recovery, University of Ottawa, Ottawa, ON K1H 8M5, Canada

[3]Centre for Neural Dynamics, University of Ottawa, Ottawa, ON K1H 8M5, Canada

[4]Brain and Mind Research Institute, University of Ottawa, Ottawa, ON K1H 8M5, Canada

[5]Department of Physics, University of Ottawa, STEM Complex, Room 336, 150 Louis Pasteur Pvt., Ottawa, ON K1N 6N5, Canada

[6]Technical Contact

[7]Lead Contact

*Correspondence: mlynn101@uottawa.ca

https://doi.org/10.1016/j.xpro.2020.100176

SUMMARY

The proportion of silent (AMPAR-lacking) synapses is thought to be related to the plasticity potential of neural networks. We created a maximum-likelihood estimator of silent synapse fraction based on simulations of the underlying experimental methodology. Here, we provide a set of guidelines for running a Python package on compatible experimental synaptic data. Compared with traditional failure-rate approaches, this synthetic likelihood estimator improves the validity and accuracy of the estimates of the silent synapse fraction.
For complete details on the use and execution of this protocol, please refer to Lynn et al. (2020).

BEFORE YOU BEGIN

Synthetic likelihoods have been developed to improve the reliability and accuracy of estimated quantities in ecology (Wood, 2010), genetics (Beaumont et al., 2002) and more recently in neuroscience (Greenberg et al., 2019; Lueckmann et al., 2019, Gonçalves et al., 2020). These approaches are computationally heavy, often requiring a simulator and extensive computer codes. Open-access packages for such simulation-based approaches makes these methods more accessible (Tejero-Cantero et al., 2020). Here, we describe the use of a simulator of synaptic electrophysiology experiments along with a calculation of synthetic likelihoods used for the estimation of silent synapse fraction.

The SilentMLE package provides an estimate of silent synapse fraction from electrophysiological data collected using a well-known sampling paradigm known as the failure-rate protocol (see Graziane and Dong (2016) for more details). Stimulation at hyperpolarized potentials leads to synaptic responses which solely reflect the stochastic activation of active synapses (i.e., both NMDAR- and AMPAR-containing), while stimulation at depolarized potentials recruits both active synapses and silent synapses (i.e., NMDAR-only containing synapses). The intuition behind this protocol is that by comparing the failure rates at depolarized and hyperpolarized holding potentials (which may recruit distinct but overlapping synapse populations), one should be able to quantitatively infer the makeup of the synapse population. Here, we describe best practices for performing these experiments and how to seamlessly and optimally use the SilentMLE synthetic likelihood estimation framework to quantify silent synapses.

One should also consider optical approaches to estimate silent synapse fraction (e.g., Lee et al., 2016; Soares et al., 2017), as these approaches are more direct, but require specialized equipment. A comparison between different approaches is presented in Lynn et al. (2020).

Data Collection

SilentMLE operates on processed synaptic failure-rate data collected from electrophysiological recordings of neurons. In principle, compatible synaptic data can be acquired from any functional synaptic population probed by electrophysiological means, yet for a number of practical reasons this approach is typically restricted to either acute or organotypic brain slice recordings that largely maintain *in situ* synaptic organization. Detailed protocols have previously been described (Schwartzkroin, 1975; Schwartzkroin, 1981; Soares et al., 2014). Briefly, brain slices are prepared from an area of interest and neurons are recorded in the whole-cell configuration while a stimulating electrode is placed nearby in the slice. The raw electrophysiological data are typically obtained using specialized hardware (e.g., equipment produced by Molecular Devices, USA), digitized, and stored.

The failure-rate protocol typically consists of 100 consecutive low-frequency synaptic stimulations (ca., 0.1–0.06 Hz), where the effects of the first 50 electrical stimulations (referred to as "sweeps") are recorded while voltage clamping at $V_m = -70$ mV while the subsequent 50 sweeps are recorded at $V_m = +40$ mV. See Graziane and Dong (2016) for more details on the failure-rate protocol. The sweep number should be chosen carefully. Increasing the number of sweeps will improve the accuracy of the estimates returned by SilentMLE (Lynn et al., 2020).

⚠ CRITICAL: Before data collection, it is important to decide on two parameters:

- An acceptable range of failure rates which all recordings should be uniformly subjected to (i.e., the percentage of electrical stimuli which lead to no measurable post-synaptic currents). The failure-rate range has an impact on the number of synapses recorded from, and on the subset bias (Lynn et al., 2020). Typically, a range of failure rates of 20% to 80% is sought. Before each recording, electrical stimulation intensity must be adjusted up or down such that the observed failure rates fall into the acceptable range at $V_m = -70$ mV. A key feature of SilentMLE is that the flexible experimental simulator can take into account different choices of failure-rate ranges, but these choices must be made before data collection.
- A target sample size for sufficient statistical power, rather than relying on *post hoc* justification. A power analysis outlining the expected granularity of silent synapse estimates using this approach with different sample size is provided in the principal paper (Lynn et al., 2020; see Figure 4).

Parsing of Successes and Failures

The analog electrophysiological data must be processed to parse individual events into successes and failures. This classification can be achieved using multiple methods, as long as there is a clear division in the sorted traces between successes and failures. For a principled approach, we recommend constructing a control distribution of mock event amplitudes from the noise. Events can be classified as successes if their amplitude is greater than some threshold (e.g., 3 standard deviations) from the mean of the control distribution. While SilentMLE does not incorporate such an algorithm, it can be easily constructed in one's programming language of choice.

The control distribution should utilize the same amplitude detection protocol employed for detecting real events, and should apply to some equal span of time in the noise. This will produce a distribution of mock events from which the mean and standard deviation can be extracted.

⚠ CRITICAL: The traces should be visually inspected after any automated steps. In particular, the standard deviation threshold may need fine-tuning, and may differ between the hyperpolarized traces and the depolarized traces. The latter may exhibit more noise and thus event parsing should be carefully examined before moving onto the next step. Inaccurate parsing of successes and failures in this step can drastically reduce the accuracy of the SilentMLE method.

Calculation of Failure Rates

Estimated failure rates for the hyperpolarized condition ($\widehat{F}_h$) and depolarized condition ($\widehat{F}_d$) must be calculated for each neuron:

$$\widehat{F}_h = \frac{n_{failures}}{n_{failures} + n_{successes}}$$

$$\widehat{F}_d = \frac{n_{failures}}{n_{failures} + n_{successes}}$$

The SilentMLE package works on the calculated failure rates for each neuron. Alternately, for existing datasets where an estimated silent fraction is already calculated from the failure rates (see Graziane and Dong, 2016; for derivation and equation, see Lynn et al., 2020), SilentMLE can work on these processed estimates instead.

Python Installation

It is necessary to ensure that you have a working, up-to-date installation of Python present on your computer, as well as an integrated development environment (IDE) or editor where code can be written and executed.

There are multiple ways of obtaining a working Python installation. For new users, we recommend an installation of Anaconda, as it provides a current installation of Python, a commonly used IDE (Spyder), as well as a full set of command-line tools.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and Algorithms		
Python	Python Software Foundation	Python >=3.6
NumPy	NumPy (van der Walt et al., 2011)	NumPy >=1.18.1
SciPy	SciPy (Jones et al., 2001)	SciPy >=1.4.1
SilentMLE Python Package	This paper	https://github.com/micllynn/silentmle

MATERIALS AND EQUIPMENT

- Data, processed into failure rates for each neuron (see Before You Begin)
- Python environment and required packages
  - Python >= 3.6
  - Numpy >=1.18.1
  - Scipy >=1.4.1
  - Matplotlib
  - h5py
  - Seaborn
- Hardware
  - No special requirements

STEP-BY-STEP METHOD DETAILS

Installing SilentMLE

Timing: 10 min

This step downloads the SilentMLE package from GitHub and installs it under your local Python installation. We provide two Variants for installation. Variant 1 is recommended for most users and Variant 2 provides guidelines for advanced users.

1. **Variant 1: Simple installation of Python and SilentMLE using Anaconda**
   a. Download and install the Anaconda Python distribution for your operating system using the instructions at https://docs.anaconda.com/anaconda/install/
   b. Open an Anaconda terminal window using the instructions to "Open Anaconda Prompt" found by scrolling down on the page at https://docs.anaconda.com/anaconda/user-guide/getting-started/.
   c. In the terminal window, run the following commands, pressing Enter/Return after each line to execute:
   ```
   conda install -y git
   pip install git+https://github.com/micllynn/SilentMLE
   python
   ```
   d. Verify the installation proceeded correctly
      i. Run the following code from the running Python instance:
      ```
      import silentmle as sil
      ```
      ii. The import should run correctly. If not, see the CRITICAL section below.
2. **Variant 2: Installation of SilentMLE for experienced users**
   a. Experienced users are encouraged to initialize a separate virtual environment before installation:
   ```
   conda create -n silentmle
   conda activate silentmle
   ```
   b. Experienced users can install the package however best fits with their workflow. A setup.py file is provided for this purpose. The following installation command should be executed from the command line with the appropriate installation of Python:
   ```
   python setup.py install
   ```
   c. Experienced users can also use their preferred text editor or IDE in place of Spyder (eg Emacs, vim, Atom, etc.), replacing the relevant sections below.

⚠ CRITICAL: It is important to install SilentMLE using the correct Python installation, or else the package will not be importable. See Troubleshooting if the installation verification does not proceed correctly.

Initializing an Estimator from Constrained Experimental Simulations

Timing: 5–20 min

In this step, we import the package and run the set of constrained experimental simulations which generate the mapping between the ground-truth fraction silent in some population, and the biased measurement returned by electrical stimulation experiments. Two variants are provided: Variant 1 describes how to initialize a simple estimator when synaptic release probabilities are unknown (assumes a null case of a uniform distribution); Variant 2 describes how to initialize a more complex estimator with a known distribution of release probabilities.

3. **Variant 1: Initializing a simple estimator with no strong prior of synaptic release probability distribution.**
   a. To launch the Spyder IDE, open Anaconda Navigator, in the "Home" tab navigate to "Spyder," and click Launch. Instructions for using Spyder can be found at https://docs.spyder-ide.org/current/quickstart.html.
   b. In Spyder, execute the following code to import the package.
   ```
   import silentmle as sil
   ```

c. To prepare the correct parameters, retrieve information from the slice physiology experiments on:
   i. The number of sweeps of electrical stimulation each cell was subjected to in each voltage-clamp condition (hyperpolarized or depolarized potentials) (`num_trials`). For example, in a case of 50 depolarized sweeps and 50 hyperpolarized sweeps, `num_trials=50`.
   ii. The lower and upper bounds of acceptable failure-rate fractions, decided upon before running experiments (`failrate_low` and `failrate_high`). For example, a range of 20% failures to 80% failures would be represented as `failrate_low=0.2`, `failrate_high=0.8`.
   iii. If using previously calculated silent fraction estimates, whether any negative silent fraction estimates were set to zero (`zeroing=True`) or not (`zeroing=False`).

d. SilentMLE takes an object-oriented approach to the estimation process. An estimator object must be initialized first (which runs a set of simulations across the range of possible silent fractions), and once initialized it can be used to perform maximum-likelihood estimation on data. To initialize the estimator, run the following code to initialize an estimator object, replacing the appropriate parameters with those specified in step 3c. (Note that `n_likelihood_points` refers to the number of silent fractions to simulate; lower values will result in more rapid estimator initialization at the expense of estimate precision.)

```
estimator = sil.Estimator(n_likelihood_points=250, num_trials=50,
    failrate_low=0.2, failrate_high=0.8, zeroing=False)
```

4. **Variant 2: Initializing a more complex estimator with an experimentally supported synaptic release probability distribution.**

a. To launch the Spyder IDE, open Anaconda Navigator, navigate to "Spyder," and click Launch. Instructions for using Spyder can be found at https://docs.spyder-ide.org/current/quickstart.html.

b. In Spyder, execute the following code to import the package.

```
import silentmle as sil
```

c. To prepare the correct parameters, retrieve information from the slice physiology experiments on:
   i. The number of sweeps of electrical stimulation each cell was subjected to in each voltage-clamp condition (hyperpolarized or depolarized potentials) (`num_trials`). For example, in a case of 50 depolarized sweeps and 50 hyperpolarized sweeps, `num_trials=50`.
   ii. The lower and upper bounds of acceptable failure-rate fractions, decided upon before running experiments (`failrate_low` and `failrate_high`). For example, a range of 20% failures to 80% failures would be represented as `failrate_low=0.2`, `failrate_high=0.8`.
   iii. If using previously calculated silent fraction estimates, whether any negative silent fraction estimates were set to zero (`zeroing=True`) or not (`zeroing=False`).
   iv. A statistical distribution, with parameters, of synaptic release probabilities.

d. Create a class instance representing the synapse release probability distribution retrieved in step 4civ. Note that any distribution from `scipy.stats` is acceptable, and specified parameters for these distributions are required in the dictionary `args`.

```
import scipy.stats as sp_stats
pr_dist = sil.PrDist(dist=sp_stats.gamma,
    args={'a':1, 'scale':0.3})
```

e. SilentMLE takes an object-oriented approach to the estimation process. An estimator object must be initialized first (which runs a set of simulations across the range of possible silent fractions), and once initialized, the estimator object can then be used to perform maximum-likelihood estimation on data. To initialize the estimator, run the following code to initialize an estimator object, replacing the appropriate parameters with those specified in step 4c. (Note that `n_likelihood_points` refers to the number of silent fractions to simulate; lower values will result in more rapid estimator initialization at the expense of estimate precision.)

```
estimator = sil.Estimator(n_likelihood_points=250,
    num_trials=50, failrate_low=0.2,
    failrate_high=0.8, zeroing=False,
    pr_dist_sil=pr_dist, pr_dist_nonsil=pr_dist)
```

⚠ CRITICAL: It is important to support the assertion of a particular synaptic release probability distribution with data and/or references from the literature. Otherwise, we recommend assuming the null hypothesis of uniformly distributed synaptic release probabilities (Soares et al., 2017).

Note: `help(sil.Estimator)` provides comprehensive documentation of each input and its expected type for this class. Many parameters controlling the experimental simulations can be fine-tuned according to the information here.

Note: All specified simulation and experimental parameters are stored in the class instance as `estimator.params` for easy access. The observation space and hypothesis space employed are stored as `estimator.obs` and `estimator.hyp` respectively. The numerically simulated likelihood function is stored as `estimator.likelihood`.

Performing Maximum-Likelihood Estimation on Collected Data

Timing: 5 min

In this step, we perform maximum-likelihood estimation using the initialized estimator. This can either be done using raw failure rates for each neuron (Variant 1), or using existing datasets consisting of silent synapse estimates from the failure-rate analysis equation (Variant 2). See Before You Begin, section on calculating the failure rates, for more information on the two variants of processing data.

5. **Variant 1: Data in raw failure-rate form**
   a. Create a variable representing the data. Note that we create a list with each cell's failure rates expressed in the format. Replace the values here with your own data. Lists can be of arbitrary length.
   ```
   data = [[0.2, 0.18], [0.4, 0.36], [0.38, 0.48], [0.58, 0.6], [0.46, 0.31],]
       [0.7, 0.43], [0.58, 0.36]]
   ```
   b. Perform maximum-likelihood estimation using the initialized data variable and the existing estimator variable. Ensure that the correct datatype (`dtype`) is set.
   ```
   likelihood = estimator.estimate(data, dtype='failrate')
   ```
   c. Maximum-likelihood estimation should be relatively rapid; wait for the estimation to complete.
   d. Examine the joint likelihood plot that was generated (see Figure 1 for examples). This depicts the likelihood of each hypothetical silent fraction being the correct value, given the data provided. One can conveniently read the maximally likely estimate (MLE) for silent fraction indicated in red.
   e. The normalized joint likelihood function (shown on the y-axis of the returned plot) is stored in likelihood for further processing and analysis. The corresponding hypothesis space of silent synapse fractions (shown on the x-axis of the returned plot) is stored in `estimator.hyp`.
6. **Variant 2: Data in processed silent synapse estimate form**
   a. Create a variable representing the data (estimates of fraction silent synapses). Note that we create a list comprised of each cell's estimated silent synapse fraction. Replace the values here with your own data. Lists can be of arbitrary length.
   ```
   data = [0.42, 0.32, 0.54, 0.43, 1.0, 1.0, -0.06, -0.18, 1, 0.42, 0.16]
   ```

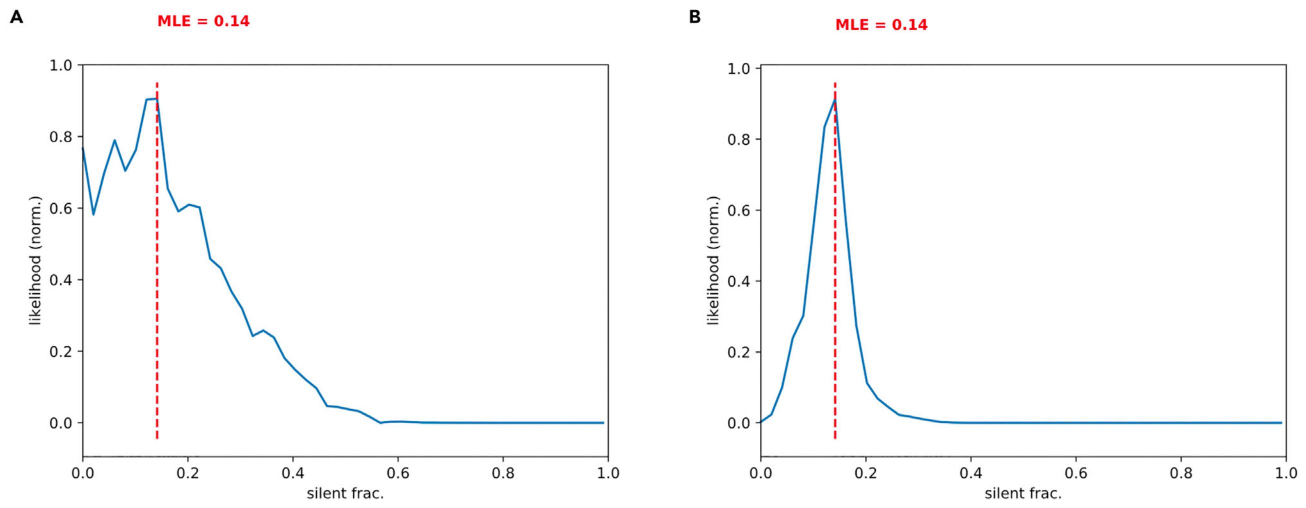


Figure 1. Maximum-Likelihood Estimation on Two Datasets with Synthetic Data
Two example plots depicting the joint likelihood function returned by SilentMLE.
(A) An example likelihood plot for n = 2 artificial data points. Note the broad likelihood distribution.
(B) An example likelihood plot for n = 20 artificial data points. Note the narrower likelihood distribution despite the same maximum-likelihood estimate. The code to generate (B) is available on the project's GitHub site at: https://github.com/micllynn/SilentMLE#example-of-full-estimation-procedure-and-output.

b. Perform maximum-likelihood estimation using the initialized data variable and the existing estimator variable. Ensure that the correct datatype (`dtype`) is set.
`likelihood = estimator.estimate(data, dtype='est')`
c. Maximum-likelihood estimation should be relatively rapid; wait for the estimation to complete.
d. Examine the joint likelihood plot that was generated (see Figure 1 for examples). This depicts the likelihood of each hypothetical silent fraction being the correct value, given the data provided. One can conveniently read the maximally likely estimate (MLE) for silent fraction indicated in red.
e. The normalized joint likelihood function (shown on the y-axis of the returned plot) is stored in likelihood for further processing and analysis. The corresponding hypothesis space of silent synapse fractions (shown on the x-axis of the returned plot) is stored in `estimator.hyp`.

Note: `help(estimator.estimate)` provides comprehensive documentation of the expected type and format of the data, the parameters used for estimation, and the type and format of the returned likelihood vector.

⚠ CRITICAL: If previously obtained silent fraction estimates have been "zeroed" (ie all negative values have been set to 0), the Estimator must have been initialized in steps 3 and 4 with the parameter `zeroing=True`.

EXPECTED OUTCOMES

Joint Likelihood Function

Figure 1 depicts the joint likelihood function associated with two simulated experimental datasets. In Figure 1A, a smaller dataset of n=2 is used as input to the maximum-likelihood estimator, generating a joint likelihood function with large variance and a maximally likely estimate of 0.14. In Figure 1B, a larger dataset of n=20 is used as input to the maximum-likelihood estimator. This generates an identical maximally likely estimate of 0.14, but the joint likelihood function has far smaller variance indicating a greater confidence in the estimate. This illustrates the importance of investigators deciding on a sufficient sample size for their experiments before using SilentMLE, as demonstrated in the power analysis in Lynn et al., 2020, to prevent estimates with low confidence. To aid users in verifying correct package installation, and to give an example of a full step-by-step protocol, the full code to generate Figure 1B is available on the package's GitHub page (see figure legend).

Output of SilentMLE

Upon successful estimation, SilentMLE returns several important variables which can either be analyzed in the same session of Python, or exported for analysis in other software. likelihood represents the normalized joint likelihood function depicted in the y-axis of the plots in Figure 1. The corresponding hypothesis space of silent synapse fractions (shown on the x-axis of the returned plot) is stored in `estimator.hyp`. Together, these can be used to perform further analysis, including confidence intervals and log-likelihood ratio testing between datasets.

To export the variables in a neutral .csv format for use with other software, we recommend the Pandas package:

```
import pandas as pd
mle_output = pd.DataFrame({'likelihood': likelihood, 'silent': estimato-
r.hyp})
mle_output.to_csv('mle_output.csv', index=False)
```

SilentMLE can additionally recreate complete figures from the original paper (Lynn et al., 2020). These take some time to run, as they recompute all relevant analysis steps and simulations. To generate the figures, the following functions can be run:

- `sil.figures.plot_fig1()`
- `sil.figures.plot_fig2()`
- `sil.figures.plot_fig4()`
- `sil.figures.plot_figS1()`
- `sil.figures.plot_figS2()`
- `sil.figures.plot_figS3()`
- `sil.figures.plot_figS4()`

Full documentation for all plotting functions can be found with `help(plotting-function)`. In addition to these functions, core functions are available for running the experimental simulator, performing the failure-rate analysis, and conducting power analyses. Full information on these functions can be found on the GitHub page (https://github.com/micllynn/SilentMLE), or by executing `help(sil.core)`.

LIMITATIONS

The accuracy of SilentMLE depends critically on several factors, including the quality of the electrophysiology data collected as well as its processing (summarized in the Before You Begin section), and the choice of parameters used to initialize the estimator. These assumptions and limitations include the following:

- Special attention should be paid to ensure that the parameters used as input to the computational model match those used during the experimental paradigm. All parameters, and the desired sample size, should be decided before experiments are started.
  - The failure-rate range provided to the model must match those in experiments.
  - The number of sweeps provided to the model must match the number in experiments.
  - Any specified synaptic release probability distribution must have strong support from either experiments or the literature. Otherwise, we recommend the null case of a uniform release probability distribution (Soares et al., 2017).
  - Any other parameters changed upon estimator initiation should be checked carefully and recorded, along with the maximum-likelihood estimate returned, for later reporting.

- A sufficient number of likelihood points are required when initializing the estimator (`n_likelihood_points`) to prevent stochasticity between estimation runs due to the Monte Carlo nature of simulations of the experimental protocol. Ideally, this value should be set to >200.
- The underlying failure-rate analysis protocol (Graziane and Dong, 2016) relies on several key assumptions which, if violated, may limit the applicability of SilentMLE. These assumptions include:
  - Equivalent release probability distributions between silent and nonsilent synapses.
  - Random sampling of silent and nonsilent synapses through electrical stimulation. Our SilentMLE package does take into account the bias in sampling through gradual reduction in electrical stimulation intensity to reach the target failure rate, but it does rely on an initial, unbiased large sample from the population. Formally, any violation of this assumption (e.g., clustered silent or nonsilent synapses, causing large initial sampling biases before adjusting stimulation intensity) may invalidate the results of SilentMLE.
  - NMDA opening is the sole parameter which differs between hyperpolarized and depolarized voltage-clamp conditions. Synapses can be regulated by depolarization-induced release of retrograde messengers, which can either increase or decrease release probability (Carta et al., 2014; Howlett et al., 2004; Chevaleyre et al., 2006). In these circuits, SilentMLE may return unreliable results due to the effects of retrograde messengers.

TROUBLESHOOTING

Problem 1

The SilentMLE package is not importable (steps 3b and 4b).

Potential Solution

It is important to make sure that the SilentMLE package is installed under the correct installation of Python. For example, on macOS there is typically a built-in installation of Python which is separate from the Python installations installed with package managers such as conda. The latter are typically accessed for commonly used IDEs such as Spyder, and thus it is crucial to install the package under them, and not under the built-in Python installation.

To determine which Python installation is associated with a given binary (e.g., python3), the following steps may be necessary. To determine the location of the binary being referenced, one can use the terminal command "$ which python-executable" on UNIX-based systems, or "where python-executable" on Windows systems. This should typically return the directory of a binary within the IDE or package manager folder (e.g., conda), and not within /usr/local/bin. If the wrong folder is associated with a Python binary you have used to install SilentMLE, you may need to reinstall the package using the correct binary. For more information on the correct binary, see your IDE or package manager of choice to determine which installation of Python is utilized.

Problem 2

While initializing the estimator object (steps 3d and 4e), the execution hangs with the following output line : "Generating estimate distributions… 0.5%" (The precise number is not important, only that the simulation hangs at this point)

Potential Solution

If the range of failure rates specified while initializing the estimator (steps 3d and 4e) is very narrow (e.g., 0.4 to 0.6), the experimental simulator may be unable to return the desired number of sets of synapses. This is because too many simulations will not reach a failure rate in the desired range, and thus will be discarded. By default, simulations are "oversampled" by 8× from 0.9<silent_frac<1.0. To control the oversampling factor (default 8x) and threshold (default 0.9), the following kwargs can be passed to the `sil.estimator()` function: `sim_oversample_factor=8, sim_oversample_thresh=0.9`. These kwargs can be altered, but a higher oversampling factor or lower oversampling threshold will require more time to complete the simulations.

Problem 3

While running the estimation on data, you receive the following message: ''ValueError: operands could not be broadcast together…'' (steps 5b and 6b).

Potential Solution

Please ensure that the data initialized in the variable data are formatted correctly, and that they matches the `dtype` parameter of `estimator.estimate()`.

RESOURCE AVAILABILITY

Lead Contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the Lead Contact, Michael Benjamin Lynn (mlynn101@uottawa.ca).

Materials Availability

No Materials were generated in this protocol.

Data and Code Availability

All code associated with this Protocol is available at https://github.com/micllynn/SilentMLE

ACKNOWLEDGMENTS

M.B.L. is thankful to have received graduate scholarships from the Government of Ontario. This work was supported by grants from the Canadian Institutes of Health Research, the Natural Sciences and Engineering Research Council of Canada, the Canada Foundation for Innovation, Brain Canada (Canadian Neurophotonic Platform) and the Krembil Foundation. We thank Emerson Harkin (University of Ottawa) and Dr. Philippe Vincent-Lamarre (University of Ottawa) for valuable help with package troubleshooting.

AUTHOR CONTRIBUTIONS

Conceptualization, M.B.L., R.N., and J.-C.B.; Methodology, M.B.L.; Software, M.B.L.; Formal Analysis, M.B.L.; Investigation, M.B.L.; Writing – Original Draft, M.B.L.; Writing – Review & Editing, M.B.L, R.N., and J.-C.B.; Supervision, J.-C.B.; Funding Acquisition, J.-C.B.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Beaumont, M.A., Zhang, W., and Balding, D.J. (2002). Approximate Bayesian computation in population genetics. Genetics *162*, 2025–2035.

Carta, M., Lanore, F., Rebola, N., Szabo, Z., Da Silva, S.V., Lourenço, J., Verraes, A., Nadler, A., Schultz, C., Blanchet, C., et al. (2014). Membrane lipids tune synaptic transmission by direct modulation of presynaptic potassium channels. Neuron *81*, 787–799.

Chevaleyre, V., Takahashi, K.A., and Castillo, P.E. (2006). Endocannabinoid-mediated synaptic plasticity in the CNS. Ann. Rev. Neurosci. *29*, 37–76.

Graziane, N., and Dong, Y. (2016). Measurement of silent synapses. Electrophysiological Analysis of Synaptic Transmission (Springer), pp. 217–224.

Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. Proceedings of the 36th International Conference on Machine Learning (International Machine Learning Society), pp. 4288–4304.

Gonçalves, P.J., Lueckmann, J.M., Deistler, M., Nonnenmacher, M., Öcal, K., Bassetto, G., Chintaluri, C., Podlaski, W.F., Haddad, S.A., Vogels, T.P., et al. (2020). Training deep neural density estimators to identify mechanistic models of neural dynamics. bioRxiv, 838383.

Howlett, A.C., et al. (2004). Cannabinoid physiology and pharmacology: 30 years of progress. Neuropharmacology *47*, 345–358.

Jones, E., Oliphant, E., Peterson, P., et al. (2001). Scipy: Open Source Scientific Tools for Python. http://www.scipy.org/.

Lee, K.F., Soares, C., Thivierge, J.P., and Béïque, J.C. (2016). Correlated synaptic inputs drive dendritic calcium amplification and cooperative plasticity during clustered synapse development. Neuron *89*, 784–799.

Lueckmann, J.M., Bassetto, G., Karaletsos, T., and Macke, J.H. (2019). Likelihood-free inference with emulator networks. In Symposium on Advances in Approximate Bayesian Inference, C. Zhang, F. Ruiz, T. Bui, A. Bousso Dieng, and D. Liang, eds. (PMLR), pp. 32–53.

Lynn, M.B., Lee, K.F.H., Soares, C., Naud, R., and Béique, J.-C. (2020). A synthetic likelihood solution to the silent synapse estimation problem. Cell Rep. *32*, 107916.

Schwartzkroin, P.A. (1975). Characteristics of CA1 neurons recorded intracellularly in the hippocampal in vitro slice preparation. Brain Res. *85*, 423–436.

Schwartzkroin, P.A. (1981). To slice or not to slice. In Electrophysiology of Isolated Mammalian CNS Preparations, G.A. Kerkut and H.V. Wheal, eds. (Academic Press), pp. 15–49.

Soares, C., Lee, K.F.H., and Béïque, J.-C. (2017). Metaplasticity at CA1 synapses by homeostatic control of presynaptic release dynamics. Cell Rep. *21*, 1293–1303.

Soares, C., Lee, K.F.H., Cook, D., and Béïque, J.-C. (2014). A cost-effective method for preparing, maintaining, and transfecting neurons in organotypic slices. In Patch-Clamp Methods and Protocols. Methods in Molecular Biology (Methods and Protocols), *vol 1183*, M. Martina and S. Taverna, eds. (Humana Press), pp. 205–219.

Tejero-Cantero, A., Boelts, J., Deistler, M., Lueckmann, J.M., Durkan, C., Gonçalves, P.J., Greenberg, D.S., and Macke, J.H. (2020). sbi–a toolkit for simulation-based inference. arXiv, preprint arXiv:2007.09114.

van der Walt, S., Colbert, S.C., and Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. Comput. Sci. Eng. *13*, 22–30.

Wood, S.N. (2010). Statistical inference for noisy nonlinear ecological dynamic systems. Nature *466*, 1102–1104.