Documentation for HMM-Bayes

Laboratory for Computational Biology and Biophysics, MIT

Table of Contents

Section 1. User Agreement

This software is covered by the MIT License.

Section 2. Introduction

HMM-Bayes is designed to analyze single particle trajectories to infer switching between diffusive and directed motion states. HMM-Bayes infers both the total number of motion states present in a trajectory as well as whether each state includes directed motion. This model selection process is performed using Bayesian inference, which penalizes model complexity to avoid overfitting. In other words, the more parameters in a model, the more data are needed to support that model.

HMM-Bayes annotates the inferred motion states in time and space along each trajectory and also reports the maximum likelihood parameters of motion for each state, including velocities and diffusion coefficients (Figures 1 and 2). HMM-Bayes can handle motion in any number of dimensions.

Subsection 2.1. Runtime considerations

We recommend that analyses of large numbers of pooled or long trajectories be performed using distributed computing on a cluster. Runtime of HMM-Bayes can be improved by using the supplied C code for the forward algorithm implementation. This C code is precompiled for Windows and OS X, but sources are included for reference and compilation on other platforms. See **Section 5** below for details on compilation. Runtime also increases with the number of tested HMM states (Kmax in **Section 4.2**).

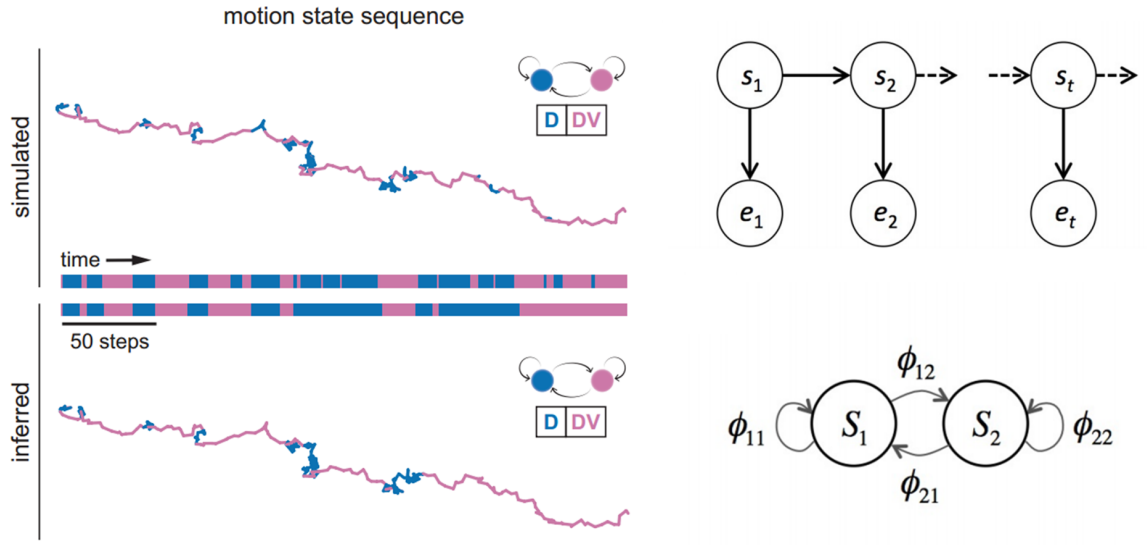

Figure 1. (left) A simulated trajectory with two states: one pure diffusion (D) and one diffusion with flow (DV) that flows to the right. HMM-Bayes infers which steps along the trajectory correspond to each motion state and where the switches between these states occur in time and space. **(right)** A graphical model representation of a two-state HMM.

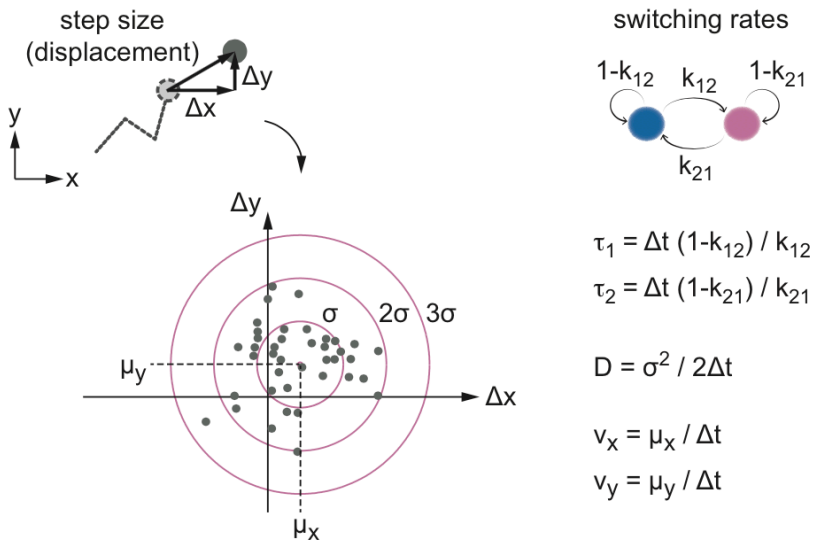

Figure 2. (left) HMM-Bayes operates on the observed displacements, or steps, along a trajectory. Each displacement is defined as the vector between two successive positions on a trajectory. These displacements are normally distributed for a diffusion process. A distribution of displacements that includes directed motion is not centered at the origin, as shown in the scatter plot. **(right)** HMM-Bayes infers a number of model parameters, including rates of transition between motion states, average state lifetimes, and motion parameters including diffusion coefficients and velocities. Examples of how to extract and plot these parameters are given in **Section 3**. Discussion and technical details of specific input and output variables can be found in **Section 4**.

Subsection 2.2. Usage considerations

The following are considerations that should be taken into account when using the HMM-Bayes software.

The detection of multiple states along a trajectory depends on the trajectory length. The longer the trajectory, the more likely that there will be enough evidence to detect multiple states. Simulations of how long trajectories need to be to resolve the presence of multiple states, depending on the motion parameters, can be found in the manuscript below.

An alternative is to pool multiple trajectories together to increase the number of observed displacements. Note that trajectory pooling with the standard implementation of HMM-Bayes with Gaussian displacement distributions assumes that all trajectories are flowing in the same directions as one another; i.e., any inferred directed motion state is the same across all trajectories. Flow will not be correctly inferred when pooling tracks with similar speeds but different directions of motion; e.g. a particle transported along a large curved path. To address such applications, one solution is to use the chi-squared HMM

implementation of HMM-Bayes, which is included with this package. The chi-squared HMM-Bayes procedure requires more or longer trajectories to adequately resolve motion models, but it readily handles pooling trajectories without being limited by the directionality of the flow. The chi-square HMM currently requires a long runtime, with a C implementation expected in the future.

Subsection 2.3. Details of the HMM-Bayes algorithm

More information can be found in the HMM-Bayes manuscript:

Monnier N, Barry Z, et al. 2015. Inferring transient particle transport dynamics in live cells. Nature Methods. doi:10.1038/nmeth.3483

Section 3. Examples

Note: Please be certain that you have the “HMMBayes_package/” and “Demo code/util/” folders added to your MATLAB path before running the examples. Navigate to the “Demo code/” folder.

Subsection 3.1. Example of plotting HMM-Bayes results

First, we examine the results of an HMM-Bayes analysis. Open **hmm_demo_result.m**. This file loads HMM results from analyzing the trajectory in Figure 1 of the manuscript and plots them using the **hmm_results.m** routine. See the .mat file for this trajectory as well as corresponding raw output from the HMM, including state annotations, diffusion coefficients, and velocities in matrix form which can be explored using the MATLAB workspace feature. For descriptions of each of the output variables, see the Output Variables section below. For examples on how to extract the parameters, see **hmm_results.m**. The **hmm_results.m** routine creates a number of plots as shown in the following figures.

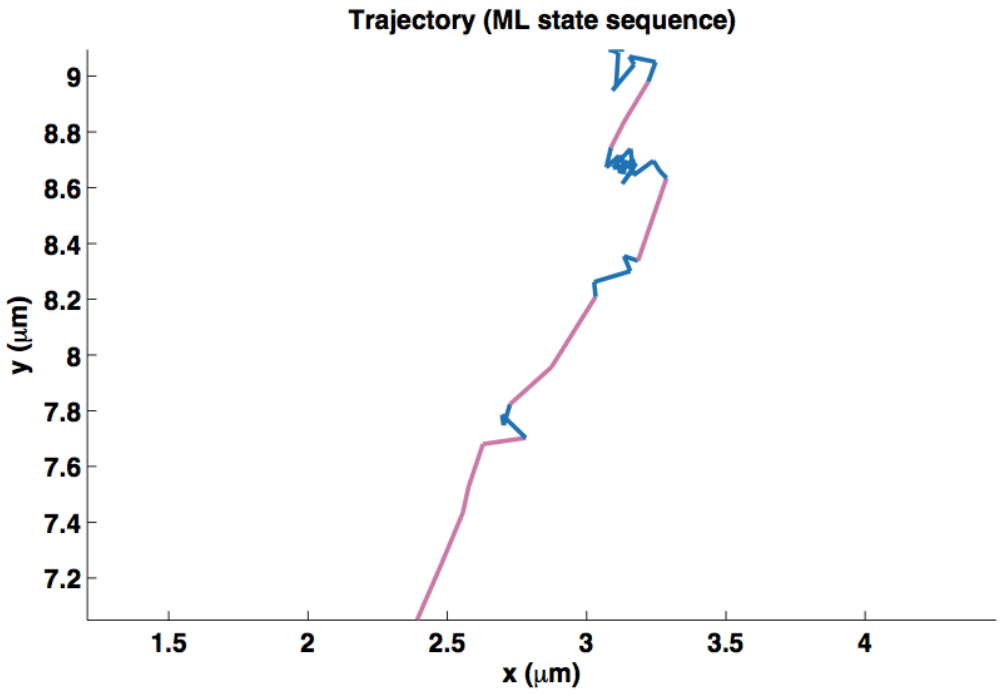

Figure 3. The input trajectory generated by single-particle tracking. HMM-Bayes infers switching between diffusion (blue) and directed motion (pink) and assigns one of these states of motion to each displacement along the trajectory.

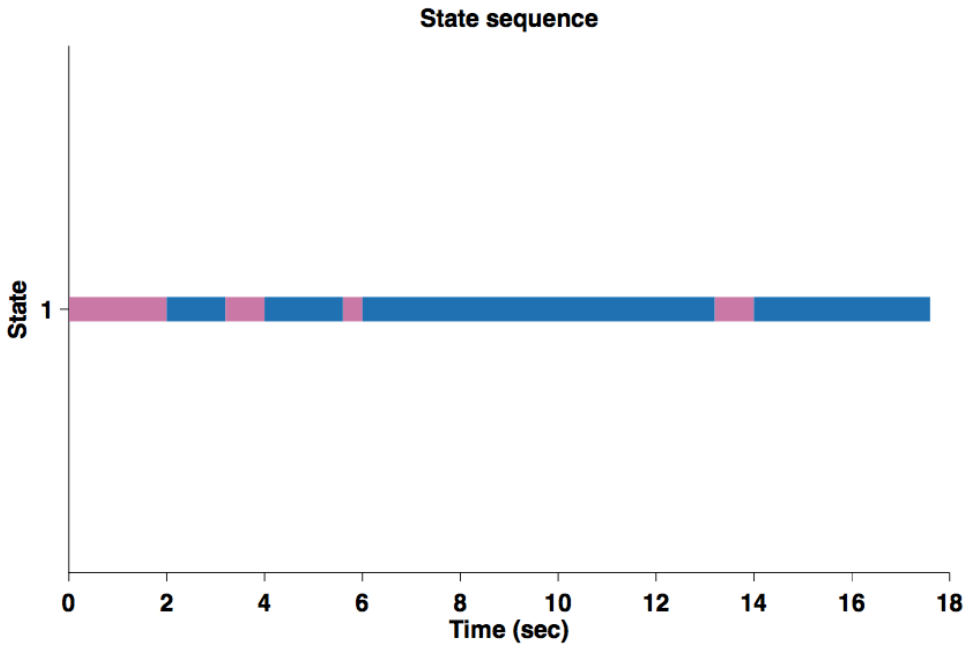

Figure 4. The temporal state sequence, or the time series of state annotations, inferred by HMM-Bayes. The colors match those used in Figure 3. Here the order of the states over time is shown, rather than the positions of the states in space.

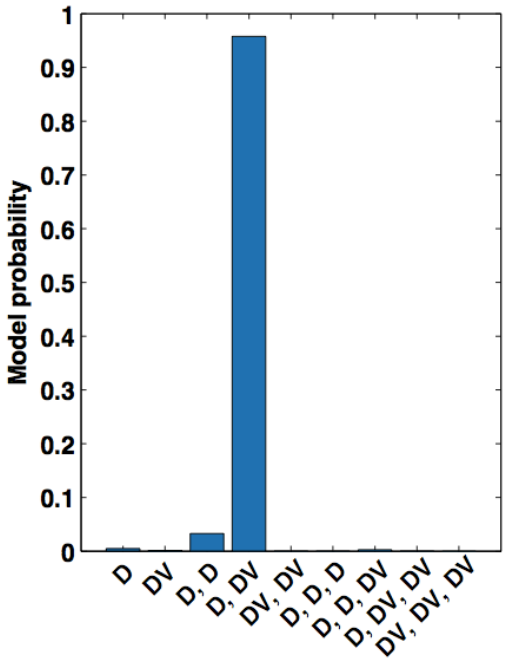

Figure 5. Model probabilities for all the different motion models tested in the HMM-Bayes analysis. Here, the “D, DV” two-state model has the highest probability, indicating that there is one diffusion state and one transport state.

Average lifetimes:

τ_1: 3.40000 (sec)

τ_2: 1.00000 (sec)

ML (estimated) parameters:

D_1: 0.00321 ($\mu m^2/s$)

V_1: [0.00000;0.00000] ($\mu m/s$)

D_2: 0.00510 ($\mu m^2/s$)

V_2: [0.20361;0.36121] ($\mu m/s$)

Figure 6. The inferred model parameters for each motion state detected by HMM-Bayes. The average lifetimes are the averages of the time periods spent in each state before switching. The maximum likelihood estimated parameters are the parameters of motion, including diffusion coefficients and velocities. The subscripts indicate the state to which the parameters correspond. In this example, state 1 is the diffusion state and state 2 is the directed motion state (matching the D, DV annotation in the model probability bar chart in Figure 5 above). The colors correspond to the state colors in Figures 3 and 4.

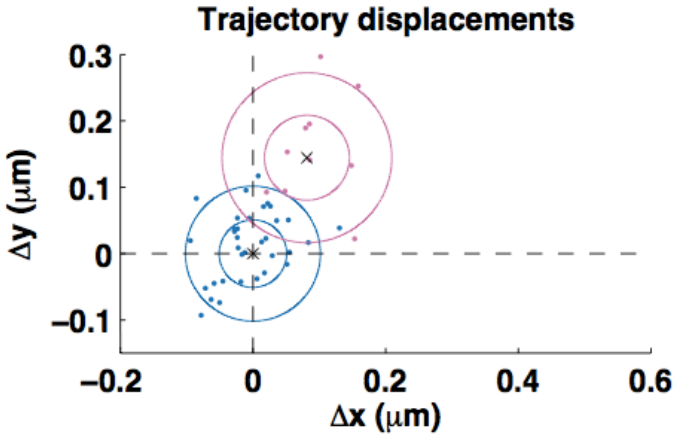

Figure 7. Scatter plot of the displacements observed in the trajectory, colored according to the state to which they were assigned by HMM-Bayes. The circles indicate one and two standard deviations for the inferred displacement distributions, with each σ related to the corresponding diffusion coefficient. The inferred distribution centers μ_x and μ_y are related to the velocities. See Figure 2 and **Section 4.3** below for converting between these parameters and the diffusion coefficients and velocities.

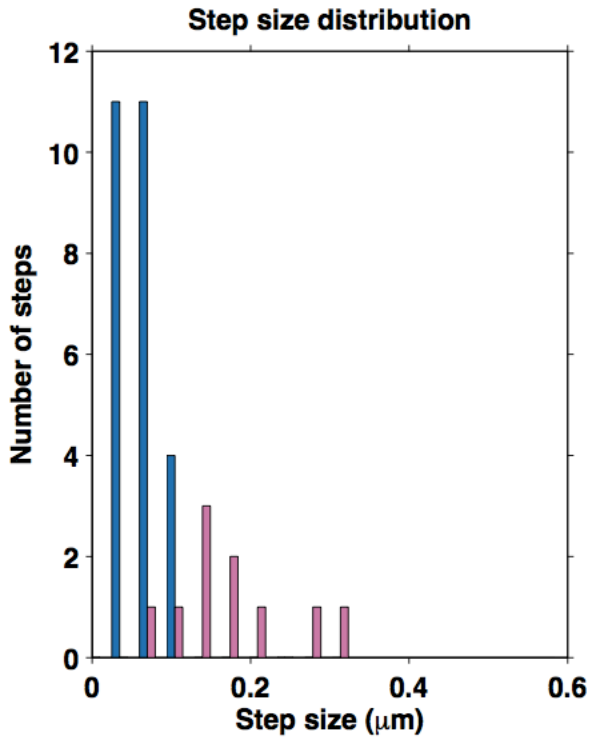

Figure 8. The step size distribution is a histogram of the displacement magnitudes colored according to the state to which they were assigned by HMM-Bayes. The magnitude of each displacement is calculated as $\sqrt{\Delta x^2 + \Delta y^2}$.

Subsection 3.2. Running HMM-Bayes on a trajectory

Next, we run HMM-Bayes on the trajectory from Figure 1 of the manuscript. See **hmm_skeleton.m** for the implementation. You can alternatively supply your own trajectory by changing the following lines,

```
data = importdata('data/example_track.mat');
track = data.track_fig1;
cfg = data.cfg_fig1;
```

to the input .mat file and the variable names appropriate for your trajectory. This file performs the following steps:

1. Loads a trajectory from the supplied file.
2. Sets up the parameters necessary to run HMM-Bayes. Note that these are the default options. The parallel option can be turned to 'off' in the case where it makes more sense to parallelize HMM-Bayes analyses on separate trajectories. For example, if you are running on a cluster, you may want to turn this off.

3. Creates displacements from the trajectory.
4. Runs the HMM-Bayes algorithm and creates a structure containing all of the output variables.
5. Saves the HMM-Bayes results to output files. The samples from the Markov Chain Monte Carlo (MCMC) procedure (discussed below) are saved to a separate file due to their size.
6. Plots the results of the analysis using the **hmm_results_plot.m** function as shown in **Section 2**. Note that you can plot the results again without rerunning HMM-Bayes; e.g. by modifying **hmm_demo_result.m** to point to the new results file "data/analysis_output.mat".

Note that because HMM-Bayes uses a stochastic MCMC sampling algorithm, the exact values of the maximum likelihood fit parameters are expected to vary slightly between runs.

Subsection 3.3. Plotting internals of the MCMC sampling

The sampling of the likelihood function by Metropolis MCMC for each tested model can be visualized in parameter space. The raw parameter values for each iteration of the MCMC sampling can be found in the *full_fitting* variable output by HMM-Bayes. The example **hmm_demo_samples.m** displays the distributions of sampled model parameters for one of the tested models, in this case the single-state model with flow (DV), for the trajectory from Figure 1 of the manuscript. The wider the distribution with respect to a particular parameter, the greater the uncertainty in the value of that parameter. The error in the estimated model parameters returned by HMM-Bayes (**Section 4.3**) is calculated using the standard deviation of these distributions. The average user will generally not need to examine the details of the MCMC sampling, but this example is provided for reference.

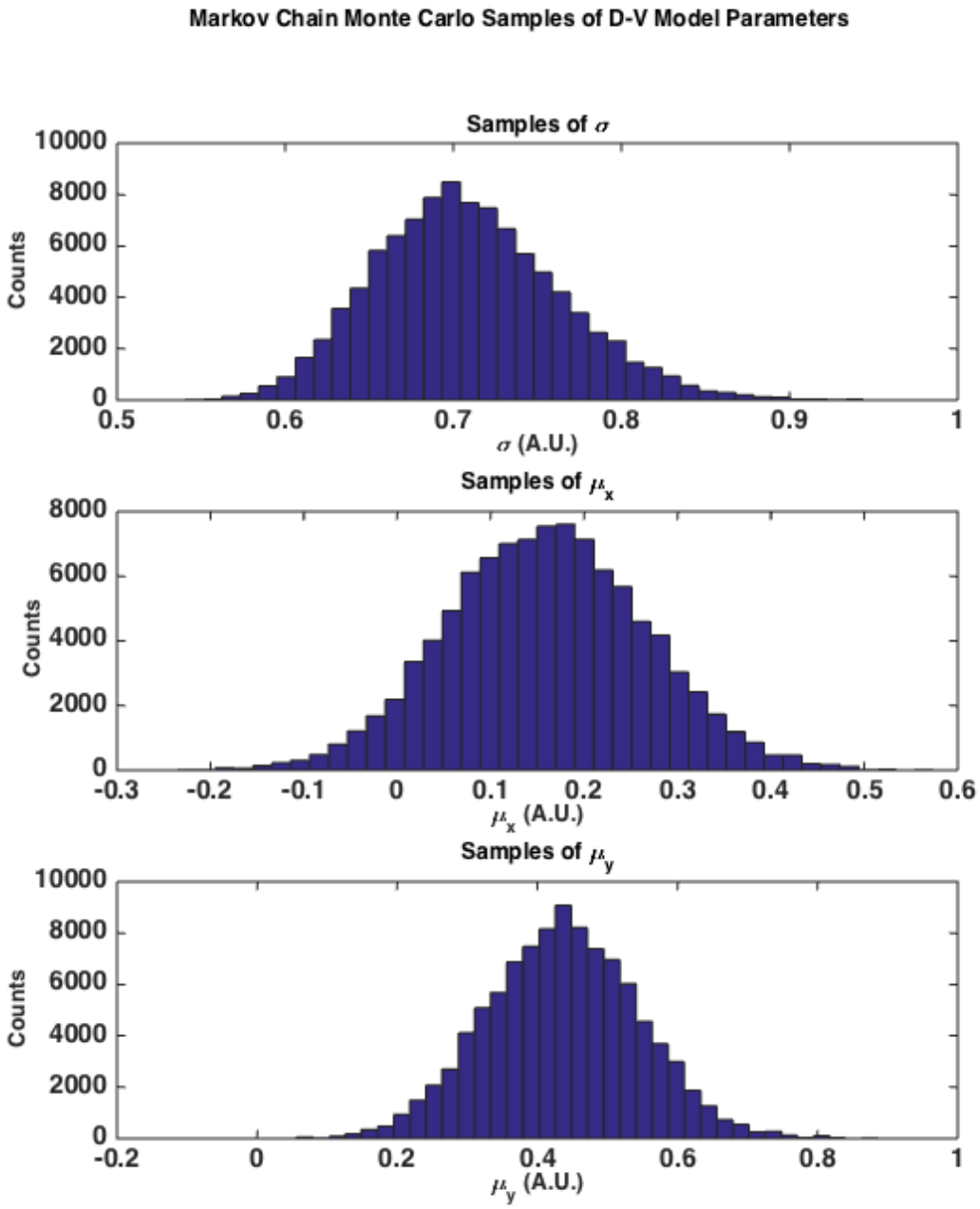

Figure 9. Distributions of the MCMC samples of σ, μ_x, and μ_y, the motion parameters of the single-state DV model, from running HMM-Bayes on the trajectory above.

Section 4. Details of HMM-Bayes input and output variables

Subsection 4.1. Entrypoint

This is the main function called to execute HMM-Bayes.

[PrM, ML_states, ML_params, full_results, full_fitting, logl] =
hmm_process_dataset(steps, Kmax, mcmc_params)

The following are explanations of the input parameters and returned output variables.

Subsection 4.2. Input parameters

steps

steps is a *d* x *N* floating point matrix of displacements or observations operated on by HMM-Bayes, where *d* is the number of dimensions and *N* is the number of displacements. For pooling trajectories, where multiple trajectories are analyzed simultaneously, *steps* becomes a *1xW* cell array of *d* x *N* displacement matrices, where *W* is the number of trajectories.

Kmax

Kmax is a positive integer that specifies the maximum number of hidden states that will be analyzed by HMM-Bayes. Competing models of up to *Kmax* motion states will be considered. Generally a max of 2-3 states is used, depending on the biological questions of interest. Note that runtime increases with increasing *Kmax*.

mcmc_params

mcmc_params is a structure containing HMM-Bayes options. The average user will generally not need to explore any options other than the *parallel* option below.

mcmc_params.parallel

parallel = 'on'

Parallelize the operation of HMM-Bayes across multiple CPU cores. Each competing motion model is treated as a separate thread and distributed to a core. Note that for processing a large number of trajectories, it may be better to run your own parallelization across trajectories instead of across models (often via *parfor* loop).

parallel = ‘off’

Disable parallel processing of motion models. Note this must be set to *‘off’* if you are using parallel processing across multiple trajectories instead.

Subsection 4.3. Output variables

Many of the output variables contain information about each tested model, where the order of the models corresponds to the pattern shown in Table 1.

Model	# of states			# of independent parameters				# of	TOTAL
	total (K)	D (V=0)	DV	π	ϕ	D	V	dimensions	PARAMETERS
D	1	1	0	0	0	1	0	1	1
							0	2	1
							0	3	1
DV	1	0	1	0	0	1	1	1	2
							2	2	3
							3	3	4
D-D	2	2	0	1	2	2	0	1	5
							0	2	5
							0	3	5
D-DV	2	1	1	1	2	2	1	1	6
							2	2	7
							3	3	8
DV-DV	2	0	2	1	2	2	2	1	7
							4	2	9
							6	3	11
D-D-D	3	3	0	2	6	3	0	1	11
							0	2	11
							0	3	11
D-D-DV	3	2	1	2	6	3	1	1	12
							2	2	13
							3	3	14
D-DV-DV	3	1	2	2	6	3	2	1	13
							4	2	15
							6	3	17
DV-DV-DV	3	0	3	2	6	3	3	1	14
							6	2	17
							9	3	20

Table 1. All possible tested motion models for a *Kmax* of three states. The vertical order of models in the table (lower to higher numbers of states, and fewer to more states with directed motion) matches the order of the indices of any output variable that contains results across different models.

PrM

Normalized model probabilities, where the preferred motion model as determined by HMM-Bayes is the index with the greatest probability.

ML_states

The maximum likelihood state sequence as determined by the Viterbi algorithm using the inferred maximum likelihood parameters for the most probable model. *ML_states* is a vector with each entry a number from 1 to *K*, where *K* is the total number of states in the selected model. This vector indicates the motion states inferred for the particle for each displacement along the trajectory.

ML_params

Maximum likelihood parameters for the most probable motion model.

ML_params.p_start

Vector of starting probabilities for the motion states of the model.

ML_params.p_trans

Matrix of transition probabilities of size *[K,K]* for a total number of hidden states *K.* The transition order is from *i -> j*, where *i* is the row and *j* is the column of this matrix.

ML_params.mu_emit

Matrix of emission means for each motion state, corresponding to the velocities of each state. Rows are the individual dimensions or velocity components, e.g. *x, y, z*, and columns are the states. To convert between mean and velocity:

$$\mu = v\Delta t$$

where v is the velocity and Δt is the time step between successive trajectory positions.

ML_params.sigma_emit

Vector of emission standard deviations for each motion state, corresponding to the diffusion coefficients. To convert between standard deviation and diffusion coefficient without localization error:

$$\sigma = \sqrt{2D\Delta t}$$

where D is the diffusion coefficient and Δt is the time step between successive trajectory positions. To convert with an estimate of the localization error:

$$\sigma = \sqrt{2D\Delta t + 2\sigma_\epsilon^2}$$

where σ_ϵ is the localization error or standard deviation of the positional noise.

full_results

Fitting results for each tested motion model, not just the most likely model.

full_results.PrM

Normalized model probability of this particular tested model. Identical to the corresponding index of the *PrM* vector discussed above.

full_results.K

Total number of states present in this particular tested model.

full_results.Vstates

Binary vector of length *full_results.K* that indicates whether each state in this particular test model has a nonzero velocity associated with it; i.e., whether it is a "DV" state rather than a "D" state.

full_results.ML_states

Maximum likelihood state sequence vector for this particular tested model, similar to *ML_states* described above.

full_results.ML_params

Maximum likelihood parameters for this particular tested model, similar to *ML_params* described above.

full_results.ML_params_error

Error estimates for the parameters of this particular tested model, equal to the standard deviation of the samples of each model parameter as calculated from the MCMC sampling of the likelihood function.

full_fitting

A structure containing MCMC samples for each tested motion model. As noted above, these samples are used to calculate the errors in the estimated parameters. See **hmm_demo_samples.m** for an example of how to extract and use these samples.

logl

Vector of unnormalized model log likelihoods.

Section 5. Compiling C code for hmm_forward.c

A C version of the forward algorithm for Windows and OS X is provided. If you wish to compile the mex file yourself, the C source is provided. It can be compiled by migrating to the HMM package folder and running:

mex hmm_forward_entry.c hmm_forward.c -output hmm_forward

For a list of supported compilers for each platform:

http://www.mathworks.com/support/compilers/R2014b/index.html;jsessionid=4a66ec0c9fcc1c10d6797c88872a