Simulation Protocol/Design

Sample Size Considerations for Bayesian Multilevel Hidden Markov Models: A Simulation Study and Application to Electroencephalogram (EEG) and Electrooculography (EOG) Data to Detect Sleep States

Written by: Jasper Ginn

Date: 20-02-2020

Last edit: 12-05-2020

Version: 0.3

Version history:

Version	Date	Changelog
v0.1	20-02-2020	First version. Ready for review.
v0.2	17-03-2020	Process comments by Emmeke
v0.3	12-05-2020	Update infrastructure design to match the implementation.

Glossary of terms

The following definitions are copied verbatim from Wikipedia. The link to their source can be visited by clicking on a term.

Term	Description
docker	Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.
image	A Docker image is a read-only template used to build containers. Images are used to store and ship applications
container	A Docker container is a standardized, encapsulated environment that runs applications
Docker hub	Docker Hub is the default registry where Docker looks for images.
docker-compose	Docker-compose is a tool for defining and running multi-container docker applications.
singularity	Singularity is a free, cross-platform and open-source computer program that performs operating-system-level virtualization also known as containerization. One of the main uses of Singularity is to bring containers and reproducibility to scientific computing and the high-performance computing world.
Google Cloud	Google Cloud Platform (GCP), offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube.
Cloud Bucket / Storage	Google Storage stores objects (originally limited to 100 GiB, currently up to 5 TiB) that are organized into buckets (as S3 does) identified within each bucket by a unique, user-assigned key. All requests are authorized using an access control list associated with each bucket and object.
Application Programming Interface (API)	An application programming interface (API) is an interface or communication protocol between different parts of a computer program intended to simplify the implementation and maintenance of software.

Purpose

This document outlines the simulation protocol to be used in my thesis.

Background

Dr. Emmeke Aarts wrote an implementation of Bayesian Multilevel Hidden Markov Models (mHMM) in R called mHMMBayes.

The objective of my thesis is to evaluate the quality of the parameter estimates of this model when varying the following quantities:

1. The number of participants (*n*)
2. The number of time-series observations for each individual (n_t)
3. The between-person variance

The objective of this protocol is to define the procedure by which the simulation study will be conducted and the results will be collected and stored.

The protocol

Simulation procedures

The MC simulations will be conducted using the R statistical language (R core team 2019) and the R library mHMMBayes (Aarts 2016). To ensure reproducibility of simulated data and model estimation, vectors of random seeds will be drawn in advance for both.

It is reasonable to assume that failures may occur (due to e.g. convergence issues). Such events are interesting in and of themselves, and occurrences will be recorded and reported on. Safeguards will be implemented in the case of failures such that the desired number of simulations is conducted.

Given the computational burden required to conduct the simulations, these will be conducted on cloud infrastructure.

Data simulation

The true parameter values used to simulate data to be used in the study will be based on the example of sleep stages. An overview of the parameters to be estimated can be found in the table below.

	Parameter	**Description**	**Total number of parameters**
Component distributions	β_{00mk}	Component distribution fixed effect in component distribution m of outcome variable k.	$k \times m$
	$\sigma^2_{u_0,mk}$	Component distribution random effect in component distribution m of outcome variable.	$k \times m$
TPM	$\bar{\alpha_{ij}}$	TPM fixed effect of the multinomial regression equation row i and column j of the between-subject TPM. Note that $i \in \{1,2,3\}$ and $j \in \{2,3\}$.	$m \times (m-1)$

Simulation scenarios

The total number of scenarios to be investigated in the simulation study is **144**. They consist of a combination of the following parameters:

Metric	**Description**	**Values**
N	Number of subjects	[10,20,40,80]
N_T	Number of observed data points for each subject	[400,800,1600]
ζ	Variance of the between-subject component distributions	[0.25, 0.5,1,2]
Q	Variance of the between-subject transition probabilities	[0.1, 0.2, 0.4]

Storage of simulation results

At each iteration, the parameter estimates for the group-level and subject-level parameters will be stored in a JSON file. Three *Maximum A Posteriori* (MAP) estimates are saved for each parameter (between-subject *and* within-subject): (1) mean of the posterior distribution, (2) the median of the posterior distribution, (3) the standard deviation of the posterior distribution. The lower and upper bounds of the 95% Central Credible Intervals (CCI) will also be stored.

Number of simulations

The number of simulations for each scenario is **250**.

Evaluation of scenarios

To evaluate the model, I will compare the results obtained from the scenarios above to the population parameters. Please see section 3 “methods” in the manuscript for more details.

Computational architecture

In designing the simulation study, I will take into account the following factors:

1. Computational efficiency
2. Scalability
3. Reproducibility
4. Persistent storage of results

I will shortly discuss each of these factors.

Efficiency

Given the computational burden of running the model, it is important that the simulation study be designed in an efficient way. Little can be done as to the design of the algorithm, since it is not within the scope of my thesis. However, some thought should be put into the design of the simulation study with respect to parallelization, computational infrastructure and so on.

The code for this study will be collected in a suite of R libraries. This library contains all the code required to run a single simulation scenario. That is to say, the library contains functions that perform the following operations:

1. Simulate a dataset
2. Run the mHMM
3. Store results

I have opted for a modular and flexible way to run the simulation study. Rather than using a multiprocessing library to execute multiple simulations in parallel, I will run each iteration of the simulation in a separate docker environment.

Scalability

The primary advantage of using docker containers is the reduced overhead in terms of coordinating multiple processes that perform simulations. Rather, each iteration of a simulation

scenario is treated as a self-contained application. Using container orchestration software such as docker-compose then allows for a straightforward way to scale the number of simulations to equal e.g. the number of threads available on a computer.

Reproducibility

Arguably, one of the most important parts of the simulation is the ability to reproduce results. In practice, this means not only that we need to set a seed at the beginning of each R session. It also requires streamlined software environments.

By using docker, this requirement is automatically met. I will compile the docker image used for each simulation and host it on docker hub. Doing so will guarantee not only that all parties that re-run one or more simulation scenarios using the same software environment (as specified by the docker image used), it also guarantees that they are using the exact same version of the software used in the container.

Persistent storage of results

Persistent storage of results means two things:

1. For each simulation scenario, about 10% of all iterations will store all data related to the simulation and modeling stages. This data consists of complete models, input data, posterior distributions etc.
2. The results of each iteration (parameter estimates).

Regardless of the cloud service that is being used, the easiest way to store the first kind of data would be to set up a data bucket on Google Cloud. This is a convenient and persistent way to store the results of the simulations. Furthermore, data stored in these buckets are easy to access and to share with other parties.

Schematic outline of the architecture

Figure 1 below shows the architecture in a flow diagram

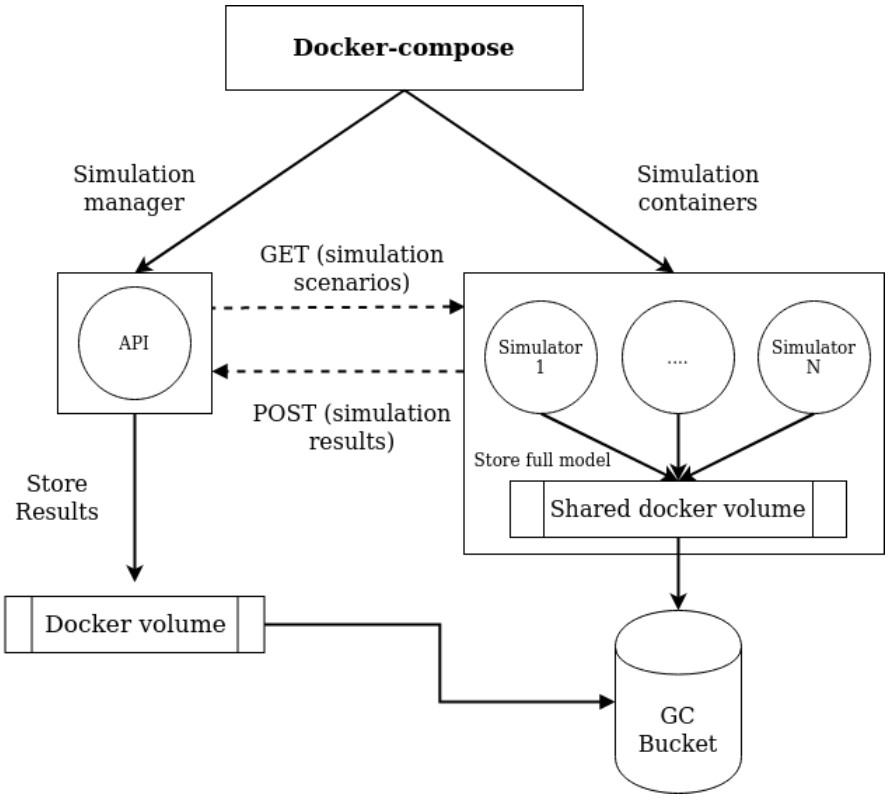

Figure 1: outline of the simulation architecture. Once a simulator is finished, the docker-compose program automatically spins up new simulators. These simulators send a GET request to the simulation manager. This simulation manager acts as a bookkeeper. It returns the scenario parameters to each simulator. Based on the simulation parameters, the simulator then executes an iteration of a simulation scenario. Once this has been completed, the simulator sends a POST request with the simulation results to the API. The API program then stores these results as flat JSON files on a docker volume. All simulators share persistent HDD space by means of a docker volume. This allows simulators to store their data in the same folder.

Outline of the simulation container

A simulation container is a docker container that is tasked with running a single iteration of a simulation scenario.

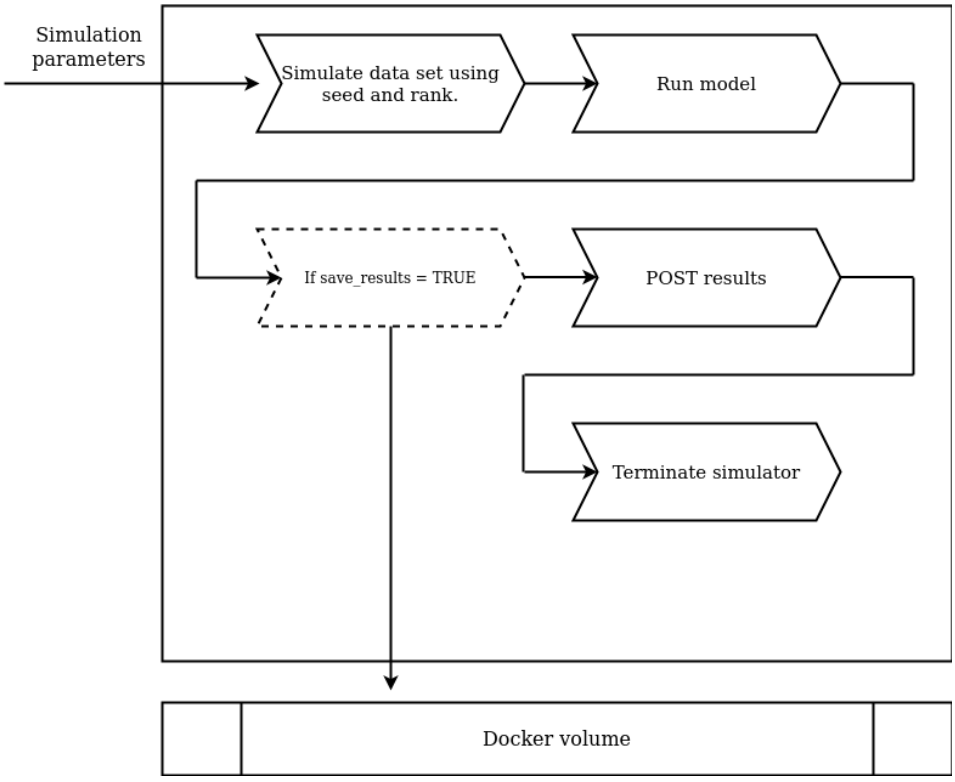

Figure 2: schematic outline of the workings of a single simulation container. After receiving the simulation parameters. The container simulates data, runs the model, optionally saves the data and sends the results to the simulation manager.

It is self-contained in the sense that it simply requires some input about the data it must simulate and some model settings, and will then run the model.

Outline of the simulation manager

The simulation manager is responsible for two things:

1. Assigning a simulation container to a simulation scenario.
2. Collecting the results from each simulation container a storing these results.

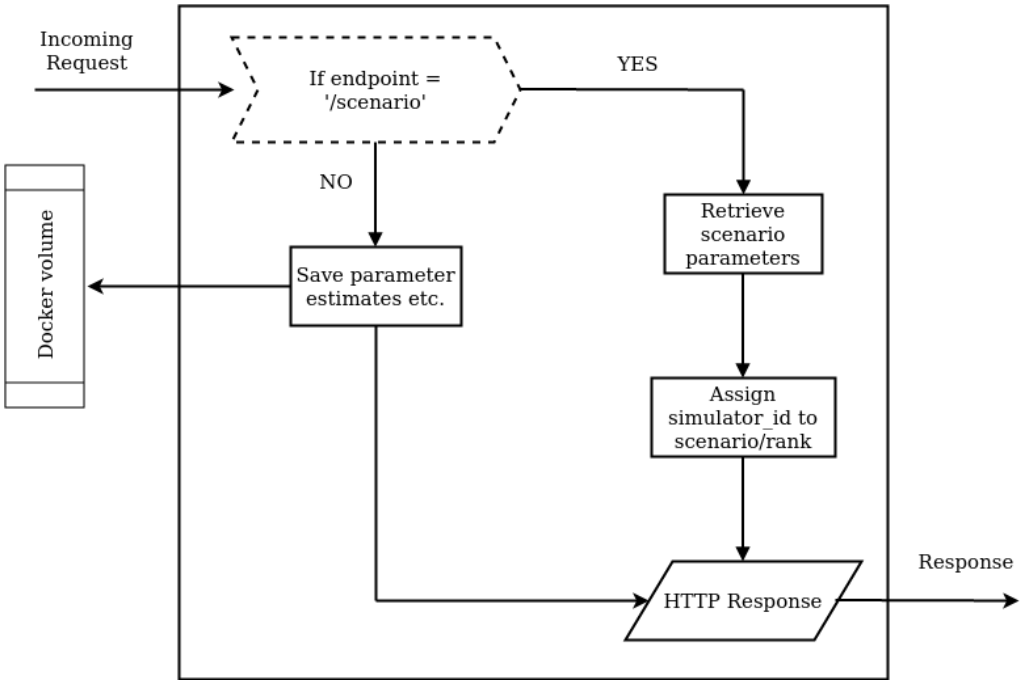

Figure 3: schematic outline of the workings of the simulation manager. This program has two endpoints (see details in figures 4 and 5 below).

To achieve the first task, I will construct a dataset in which each of the iterations is specified according to their simulation parameters (see table below)

Column	**description**
scenario_id	unique ID of the scenario
rank	the iteration number, ranging from 1-250
seed	random seed used for the scenario
n	number of subjects
n_t	length of the observed outcome data for each subject
var_gamma	between-subject variance for transition probabilities
var_depvar	between-subject variances of the dependent variables
start_values_gamma	starting values for the transition probabilities
start_values_depvar	starting values for each dependent variable

The simulation manager takes an incoming request of a simulation container and assigns that container's unique ID to a simulation scenario/rank combination. The parameters of this scenario and then passed to the simulation container by means of an API. See figure 3 below.

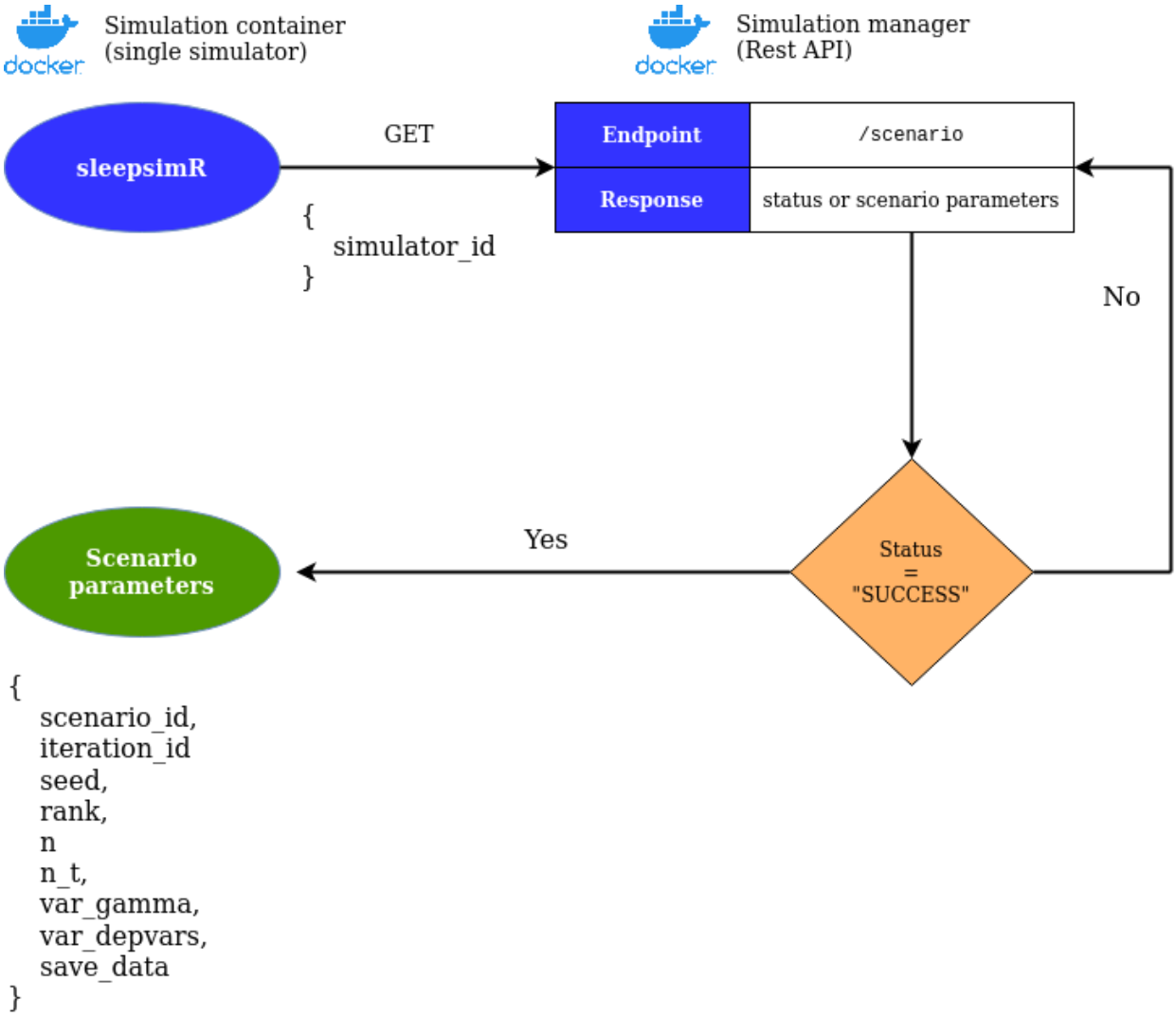

Figure 4: Schematic outline of the GET request sent by each simulator/simulation container to the simulation manager at the start of the simulation progress. A simulator is initialized with a unique ID. The simulator sends this ID to the resource manager. The resource manager registers the ID and assigns a scenario to this pod. This information is then returned to the simulator, after which the iteration of the simulation scenario is executed.

The following data is sent by the simulation container to the simulation manager:

Parameter	**Description**

simulator_id	[Int] Unique ID assigned to the simulation container when it is created.

The following data is returned by the simulation manager:

Parameter	**Description**
scenario_id	[Char] Unique ID of the simulation scenario
data_seed	[Int] Seed value used to generate data used in the scenario
model_seed	[Int] Seed value used when running the model. Makes the outcome exactly replicable.
rank	[Int] Number ranging from 1-250 denoting the iteration number in a simulation scenario.
n	[Int] Number of subjects to simulate in this scenario
n_t	[Int] Length of the observed data for each subject in this scenario
var_gamma	[Float] Variance of the between-subject transition probabilities in this scenario
var_depvars	[Float] Variance of the between-subject dependent variables in this scenario
save_data	[Logical] Whether or not all model data should be saved by this simulation container.

Once the simulation container has finished the iteration it was tasked to work on, it sends back the results to the simulation manager by means of a POST request:

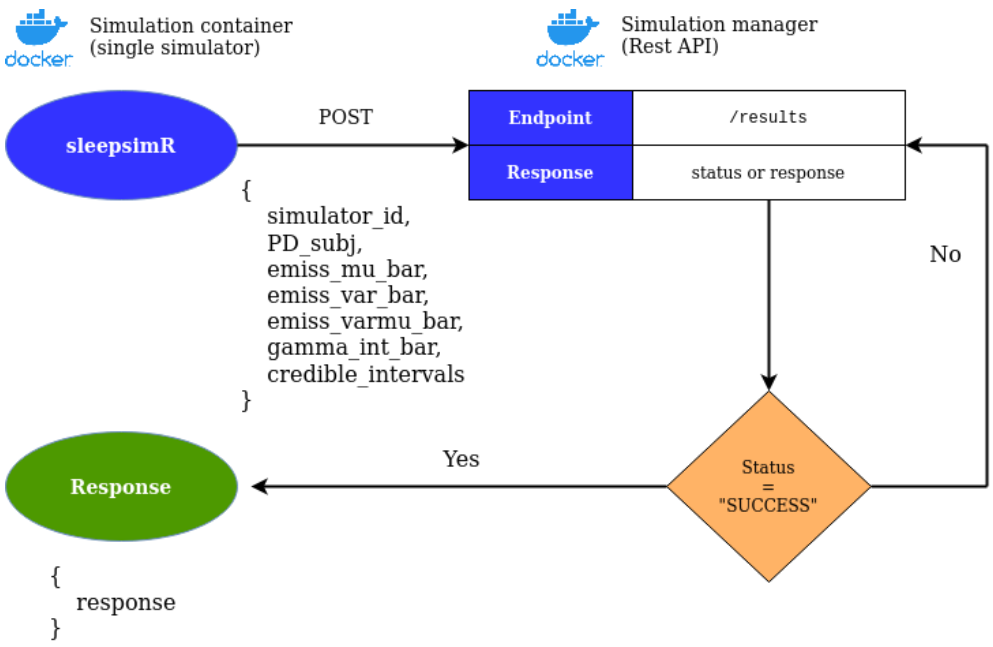

Figure 5: schematic outline of the POST request sent to the simulation manager at the end of each simulation scenario. A simulator sends all required information to the manager. In turn, the manager then saves this information to disk. Once the data transfer is completed, the simulation manager responds with a shutdown message that terminates the pod.

The following data is sent to the simulation manager:

Parameter	**Description**
simulator_id	[Int] The unique ID of the simulation container
PD_subj	[json[json]] subject-specific parameter estimates, for each of N subjects in the simulation iteration.
emiss_mu_bar	[json[json]] component distribution fixed effects
emiss_var_bar	[json[json]] component distribution residual errors
emiss_varmu_bar	[json[json]] component distribution random effects
gamma_int_bar	[json[json]] TPM fixed effects
credible_intervals	[json[json]] 95% central credible intervals for each parameter (except the subject-specific parameters).

The following data is returned by the simulation manager

Parameter	Description
response	[Char] Status code indicating that the pod can be terminated.

Example of a results file

An example of an output file is given in the image below. The subject-specific parameter estimates are truncated such that the estimates of two subjects remain. The name of the results files are equal to the iteration uid. The simulation iteration is uniquely identified by the uid of the simulator (“uid”), the uid of the scenario (“scenario_uid”) and the uid of the simulation iteration (“iteration_uid”).

The entries at the bottom (“label_switch” and “state_orders”) should be disregarded, as these entries are not used in the analysis and the values are not correct.


```
{
	"uid": "c636fd92-8e0b-4233-a3c0-df446d9857ce",
	"scenario_uid": "108677fe17d1458acbd496cfa807e5a6",
	"iteration_uid": "00002cade6007a85e30698df9b95ac5f",
	"PD_subj": [
		{
		"mean": [
		0.114884360612411,
		2.3885215479489,
		-3.36309698077056,
		-1.62463544149812,
		-2.36742354198958,
		-0.0136561533803764,
		-0.281301051602012,
		-1.65555864546615,
		1.74031361613354
		],
		"median": [
		0.123337914000056,
		2.39856292338588,
		-3.35637841175906,
		-1.62850717405998,
		-2.36324112404885,
		-0.00735858533649281,
		-0.288172874501498,
		-1.65303640288278,
		1.7409575286054
		],
		"SE": [
		0.276183357654721,
		0.286233733172006,
		0.321387334601432,
		0.295690287096582,
		0.306701065606111,
```

```
                0.297000409815712,
                0.275501382088701,
                0.289925496688369,
                0.296881627095169
                ]
                },
                ...
                {
                "mean": [
                0.765796752568932,
                0.4192266368362,
                -2.19055648310205,
                0.948137717109231,
                -0.810198737829511,
                -0.552822240289427,
                1.18143174475403,
                -4.08873205084344,
                -1.94756664764956
                ],
                "median": [
                0.765050460623838,
                0.423612209915786,
                -2.19012341202951,
                0.942713693304077,
                -0.813387947315334,
                -0.556049577090628,
                1.17587953056886,
                -4.08323183838668,
                -1.95101404836975
                ],
                "SE": [
                0.267587168190191,
                0.286031448398589,
                0.321162235063665,
                0.296060755679333,
                0.303606441074797,
                0.298779874123433,
                0.288363733456318,
                0.295740588675997,
                0.291462573313425
                ]
                }
        ],
        "emiss_mu_bar": {
                "EEG_mean_beta": {
                "mean": [
                -0.0645442156860314,
                0.0727698021739886,
                0.203409594347231
                ],
                "median": [
                -0.0619822125898962,
                0.0746566434812772,
                0.204471850740298
                ],
                "SE": [
                0.281291159321376,
                0.375390510893672,
                0.791365888994564
                ]
```

```
},
        "EOG_median_theta": {
        "mean": [
        0.808791139315793,
        -0.776522127022836,
        -0.697941665396519
        ],
        "median": [
        0.823688676157774,
        -0.772804057596487,
        -0.704463897221131
        ],
        "SE": [
        0.45031028147072,
        0.4370170606199,
        0.609144344043731
        ]
        },
        "EOG_min_beta": {
        "mean": [
        0.915063509283289,
        -1.161550574287,
        -0.208917572911978
        ],
        "median": [
        0.915850554697655,
        -1.14985587048979,
        -0.220430335737963
        ],
        "SE": [
        0.337631236509684,
        0.528392806329977,
        0.412877158832728
        ]
        }
},
"gamma_prob_bar": {
        "mean": [
        0.949465063988504,
        0.0225973867124772,
        0.0279375800682497,
        0.0279655623208356,
        0.916327199495518,
        0.0557072689528769,
        0.0393444319951472,
        0.0410449911880857,
        0.919610607585998
        ],
        "median": [
        0.953977422287751,
        0.0201816291740066,
        0.0250087192683107,
        0.0253702841610455,
        0.922624658616155,
        0.0508576592827487,
        0.0360186499467115,
        0.0374649343408398,
        0.925662773651893
        ],
        "SE": [
```

```
                0.022890803722637,
                0.0123325451020171,
                0.013665928430457,
                0.0136662144374963,
                0.0358977540355501,
                0.0259799839111418,
                0.0185399282675721,
                0.0197530877452207,
                0.0349137264367339
                ]
        },
        "emiss_var_bar": {
                "EEG_mean_beta": {
                "mean": [
                0.17189027471599,
                0.178651439398243,
                0.205946442290033
                ],
                "median": [
                0.162026987770329,
                0.168852080251506,
                0.192192330435398
                ],
                "SE": [
                0.0438315927126119,
                0.0492202447087295,
                0.0660768422826182
                ]
                },
                "EOG_median_theta": {
                "mean": [
                0.188501460675925,
                0.194046631447225,
                0.189556471615936
                ],
                "median": [
                0.177946420774139,
                0.182415713277344,
                0.176041542776849
                ],
                "SE": [
                0.0514533759350331,
                0.0548251013744541,
                0.057022980675549
                ]
        },
        "EOG_min_beta": {
                "mean": [
                0.181453249252446,
                0.177376019012709,
                0.184735551104932
                ],
                "median": [
                0.172693194356542,
                0.167170510263606,
                0.174907830579831
                ],
                "SE": [
                0.0454083552495604,
                0.050933146848419,
```

```
                0.0515544893132169
                ]
                }
        },
        "emiss_varmu_bar": {
                "EEG_mean_beta": {
                "mean": [
                0.779504418062832,
                1.3886819853938,
                6.61954644348054
                ],
                "median": [
                0.661744166839101,
                1.17573216768081,
                5.73167063342965
                ],
                "SE": [
                0.462737934579468,
                0.829697451640343,
                3.86303741001265
                ]
                },
                "EOG_median_theta": {
                "mean": [
                2.23720750949366,
                2.09541149102069,
                3.9245982950557
                ],
                "median": [
                1.920561876452,
                1.81260860564666,
                3.40433983604838
                ],
                "SE": [
                1.26085320210159,
                1.12182706632326,
                2.11334014837593
                ]
        },
        "EOG_min_beta": {
                "mean": [
                1.15210079398648,
                2.87297866237592,
                1.81868409604851
                ],
                "median": [
                0.988661739748735,
                2.53506096158966,
                1.57208512254878
                ],
                "SE": [
                0.703333363889633,
                1.49712962137032,
                0.969313098268087
                ]
                }
        },
        "credible_intervals": {
                "gamma_prob_bar": [
                0.894100535683423,
```

```
0.979857431379058,
0.00716556305070044,
0.0548487894329905,
0.0103532350016529,
0.0617195399186364,
0.0094469532619571,
0.0618767592452426,
0.827892035594648,
0.967956877808634,
0.0201100737742713,
0.119093137809799,
0.0140916435300899,
0.0864966883626266,
0.0141439416683902,
0.0890776032270243,
0.834591100996883,
0.969071112873072
],
"emiss_mu_bar": {
"EEG_mean_beta": [
-0.638359163913941,
0.506323927124416,
-0.704734812080202,
0.808731579000467,
-1.37394703514634,
1.77542083081945
],
"EOG_median_theta": [
-0.085169208309336,
1.71730625818443,
-1.60266722964248,
0.0872341731207707,
-1.92320353734094,
0.537519036963898
],
"EOG_min_beta": [
0.260370718033382,
1.59011475496857,
-2.22789058783519,
-0.153499005299836,
-1.02063005643517,
0.620089732518197
]
},
"emiss_var_bar": {
"EEG_mean_beta": [
0.116456732774539,
0.280482570518433,
0.116808902762526,
0.303322707136197,
0.125660546465458,
0.360662106796833
],
"EOG_median_theta": [
0.122988960637723,
0.31604416116783,
0.125918896097914,
0.328106752816018,
0.117999161891972,
0.332212746039212
```

```
                ],
                "EOG_min_beta": [
                0.121428139689324,
                0.2952835995407,
                0.1123101502207,
                0.303570210904987,
                0.118778168805329,
                0.312659882661899
                ]
                },
                "emiss_varmu_bar": {
                "EEG_mean_beta": [
                0.28456862505175,
                2.01140516748697,
                0.523149154605457,
                3.52300266175017,
                2.63139285411163,
                15.9227703041885
                ],
                "EOG_median_theta": [
                0.847912884722857,
                5.53274345848819,
                0.841261840665601,
                5.10865534754703,
                1.57521506724842,
                8.91419464385865
                ],
                "EOG_min_beta": [
                0.413401884594569,
                2.83432805618721,
                1.16970040184003,
                6.89077710860635,
                0.747878266550659,
                4.44595146591543
                ]
                }
        },
        "label_switch": [
        0.0,
        0.0,
        0.0,
        39.17,
        99.11,
        99.78,
        40.12,
        46.49,
        8.06
        ],
        "state_order": {
        "EEG_mean_beta": [
        0,
        0,
        0
        ],
        "EOG_median_theta": [
        0,
        0,
        0
        ],
        "EOG_min_beta": [
```

```
            0,
            0,
            0
            ]
            }
}
```