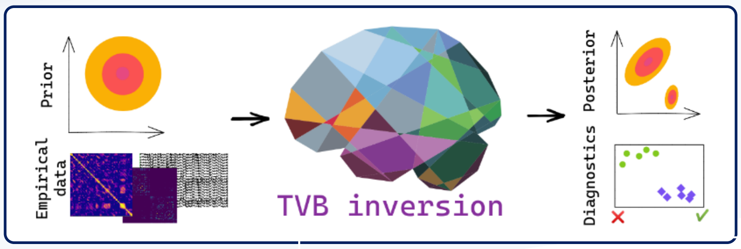
VirtualBrainCloud

Personalized Recommendations for Neurodegenerative Disease



www.VirtualBrainCloud-2020.eu

Public deliverable report

D8.1: Validated and benchmarked data fitting workflow will be provided as a software for brain model personalization

Date	November 2022
Authors	Jan Fousek, Meysam Hashemi, Abolfazl Ziaee Mehr, Viktor Jirsa (Aix-Marseille L'Université) Paula Prodan, Lia Domide(Codemart) Jochen Mersmann (Codebox) Sandra Diaz (Forschungszentrum Jülich) André Gemünd, Horst Schwichtenberg (Fraunhofer SCAI)
	© VirtualBrainCloud consortium
Dissemination level	**public**
Website	www.VirtualBrainCloud-2020.eu

This project has received funding from the European Union´s Horizon 2020 research and innovation programme under grant agreement No 826421

Table of content

1. Introduction

The VirtualBrainCloud (TVB-Cloud) is building a cloud-based brain simulation platform supporting personalized diagnostics and interventions in the context of neurodegenerative diseases. Personalization of the virtual brain models and their use in inference stands at the center of the focus of the workflows implemented in the platform. This deliverable provides a software package implementing such workflow together with suitable diagnostics, benchmarks, and validation.

Whole-brain network models constructed from personalized anatomical data provide a basis for estimation of model parameters using Bayesian inference. The Bayesian framework is a principled method for inference and prediction with a broad range of applications, while the uncertainty in parameter estimation is naturally quantified through probability distribution placed on the parameters (so-called prior distribution, derived from physiological information or previous evidence) updated with the information provided by data (through the so-called likelihood function, i.e. the probability of some observed outcomes given a set of parameters).

Simulation Based Inference (SBI) or likelihood-free inference performs Bayesian inference for complex models where calculation of the likelihood function is either analytically or computationally intractable (Cranmer et al., 2020). Instead, SBI sidesteps this problem by employing an artificial neural network (ANN) to learn an invertible transformation between parameters and data features to approximate the posterior distributions from a set of simulations with parameters randomly drawn from the prior distribution. After the training step, evaluating the posterior distribution for new observations is done quickly by a forward pass through the trained ANN. The advantages of this method include amortization of the invested compute time over repeated inference steps (e.g. different hypotheses or observations), avoiding difficult statistical model parametrization, flexibility, and capacity to deal with multimodal posterior distributions, and it was shown to perform well in the context of epileptogenic zone estimation (Hashemi et al., 2022) and healthy ageing trajectories (Lavanga et al., 2022). By providing fast simulations and informative low-dimensional data features, this methodology can be applied to other whole-brain models, since it requires neither model nor data features to be differentiable (Gonçalves et al. 2020).

In this deliverable we present a brain model personalization workflow based on SBI. This workflow is fully compatible with current data processing pipelines (e.g. Jung, K. et al. 2021), and can be easily adapted for model parameter inference across diverse use-cases. Here we demonstrate the use of the workflow on the example of inference of coupling strength using the data features derived from resting state functional magnetic imaging (fMRI) recordings, and demonstrate how to perform validation on the synthetic data. We provide performance benchmarks for representative systems and scenarios to guide the prospective users in planning their compute infrastructure usage. Finally, we demonstrate suitable diagnostic approaches for assessing the accuracy of the inference and for detecting some of the potential failures arising from modeling and problem formulation choices.

2. Partners involved

AMU: method development, workflow design
CODEMART: software development, benchmarking
CODEBOX: architectural integration
Forschungszentrum Jülich: cloud infrastructure
FRAUNHOFER: cloud infrastructure

3. Description of work performed

The developed data fitting workflow is based on the Simulation Based Inference (SBI) and provides an easy-to-use and flexible interface integrating The Virtual Brain (TVB) simulator and a mature generic toolkit for SBI (Tejero-Cantero et al., 2020). As such it allows the user to formulate with great expressive power both the model and the inference scenario in terms of observed data features and model parameters. Part of the integration entails the possibility to perform in parallel the numerous simulations required for training of the deep neural networks implementing the Normalizing Flows. Normalizing Flows (NFs) are a family of generative models that convert a simple initial distribution into any complex target distribution, where both sampling and density evaluation can be efficient and exact (Rezende & Mohamed, 2015). This sampling step was designed with portability in mind to allow easy deployment on scheduler-based infrastructures such as local clusters and supercomputing centers.

The individual steps of the workflow were benchmarked to assess the computational costs across different model implementations and execution configurations. The decoupling of the inference steps from the simulations allowed developing a compact and extendable interface for defining and parallel execution of large numbers of simulation instances with varying parameters. As such, this can be further used outside the inference context for systematic parameter space exploration, another common task in virtual brain modeling. To improve performance of the individual simulations, TVB was extended with the computational backend functionality allowing to provide the high-performance implementations for selected model configurations while systematically testing for correctness with the reference implementation.

To assess the robustness and the inference capabilities of the SBI workflow, we have performed a validation study on synthetic data. Here, using the simulated data provided with a known ground truth for the parameters allows to demonstrate the limits of the method in the given inference scenario (e.g. resting state) for selected functional data features.

The initial prototypes of the workflow were developed into a portable Python software package implementing the individual steps. This package was deployed and tested at the compute infrastructure provided by TVB-Cloud partners, in particular at the JURECA Supercomputer operated by the Jülich Supercomputing Centre and on the testbed cluster provided by the FRAUNHOFER.

4. Results

In the following sections we detail the overall architecture, individual steps and usage of the workflow, the validation of the used methods, and assess the performance.

4.1. Workflow implementation

The workflow starts with defining the brain network model and parameters which will be subjected to the inference. The actual inference follows in four main steps:

1. define a prior distribution over the parameters of interest.
2. draw samples from the prior, simulate time series with TVB and compute summary statistics (data features)
3. train a ANN to compute the Normalizing flows in a SBI family to estimate a mapping function between input (parameters) and output (data features).
4. compute the summary statistics from empirical functional data and use the trained estimator to compute the posterior distribution of parameters following the Bayes's rule.

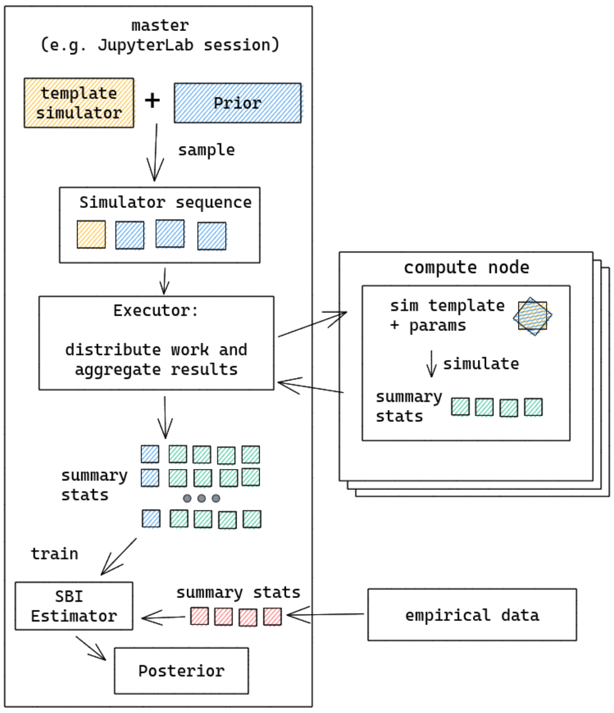


Figure 1: TVB inversion workflow

The individual steps are realized by the corresponding components of the tvb-inversion package and demonstrated in detail in demos in the notebooks folder. The main advantage of the chosen architecture is the decoupling of the simulation step from the rest of the workflow, which allows simple adaptation of the parallel execution to the new deployment environments.

4.2. Validation

Here we first simulated a whole brain model of resting state, in which a Montbrió-Pazó-Roxin (MPR) model is placed at each brain region, which are connected through a structural connectivity (SC) matrix (here, N=68). Then the aim is to invert the model using SBI, in which a deep learning architecture known as normalizing flow is trained on the data features such as functional connectivity (FC) and functional connectivity dynamics (FCD), extracted from a large number of simulations.
In addition to comparison of the posterior distribution to the ground truth, there are two main diagnostic measures which can be applied to assess the inference: posterior shrinkage and the posterior z-scores. In brief, high shrinkage indicates that the posterior was well identified (the posterior distribution is "shrinking" with respect to the initial prior distribution), while the low z-score indicates that the true values (ground truth) are accurately encompassed in the posteriors. Following Betancourt 2018, the posterior shrinkage and the posterior z-scores are defined as:

$$z = |\frac{\bar{\theta} - \theta^*}{\sigma_{post}}|, \qquad (1)$$

$$s = 1 - \frac{\sigma^2_{post}}{\sigma^2_{prior}}, \qquad (2)$$

where $\bar{\theta}$ and θ^* are the estimated-mean and the ground-truth, respectively, whereas σ^2_{prior}, and σ^2_{post} indicate the variance (uncertainty) of the prior and the posterior, respectively.

4.2.1. Basic data features

We demonstrate how the number of simulations and the choice of functional data features impact the results of inference. Here, we take a model built on a structural connectivity built from parcellation of 68 regions and the MPR model governing the activity in each node. We inverted the model with the help of SBI to recover the global coupling scaling parameter G using two data features: the sum over the functional connectivity (FC) and the sum over the functional connectivity dynamics (FCD).

First, we verified that the posterior shrinkage increases with increased number of samples from the prior used to train the estimator (Fig 2). From such a sweep we can determine how many samples are needed in a particular inference scenario defined by a combination of model, inferred parameters, and chosen functional data features. This is very important information as it informs the most computationally demanding step (running simulations with parameters sampled from the prior).

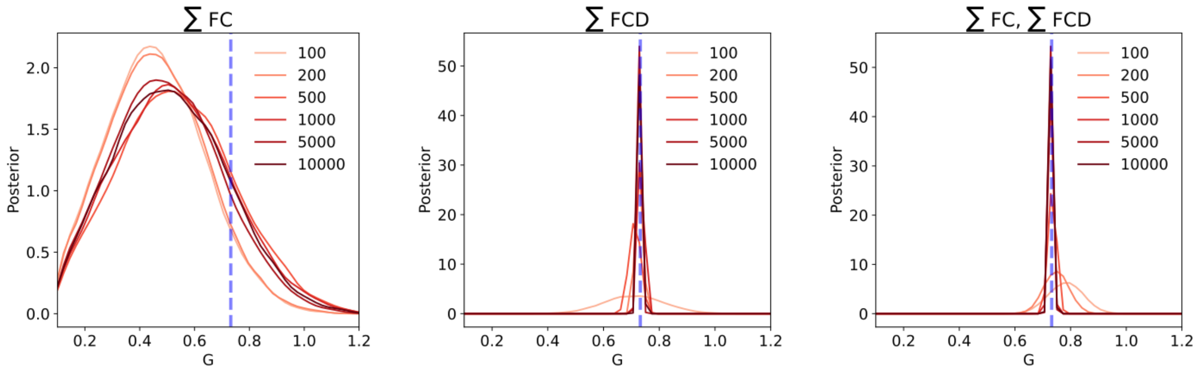


Figure 2: Posterior distribution for the sum of FC and sum of FCD data features across an increasing number of simulations used to train the estimator. Note, that by increasing the simulations the shrinking of the posterior distribution of the FCD sum significantly increased, while the FC sum posterior remains the same suggesting additional data doesn't improve the accuracy of the estimator. Similarly, the posterior distribution is shrinking with number of samples for the combined data feature (panel right), even more so for the low number of samples than the posterior computed using FCD alone.

Second, the combination of both of the data features performs slightly better than FCD alone and significantly better than FC alone in terms of both posterior shrinkage and z-score as can be seen in Fig 3. Based on this, one would choose the combination of these two data features for subsequent analysis. Of course, it might not always be the case that more data-features improve the quality of inference, while in general increasing the number of features does impact the number of samples needed to train the estimator.

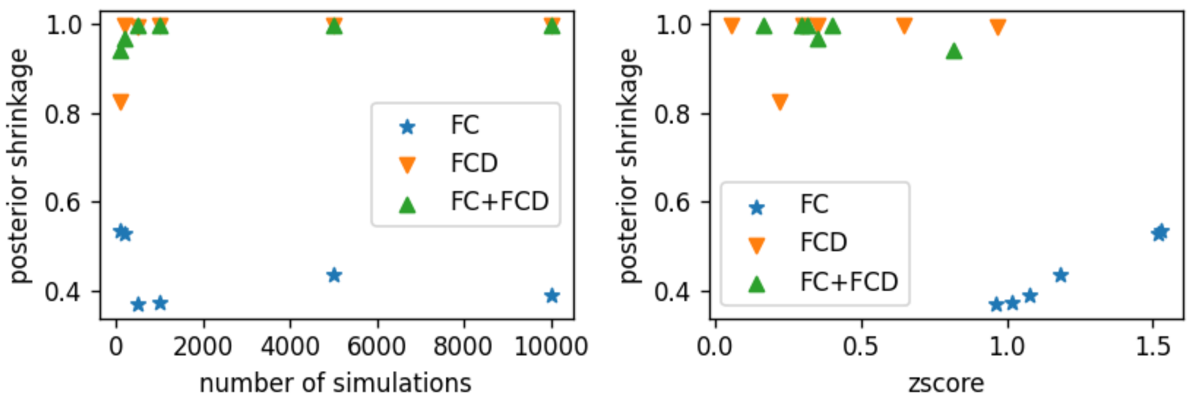


Figure 3: Posterior shrinkage and z-score for the FC and FCD data features and their combination. The left panel shows the change of the posterior shrinkage with the number of simulations (samples from the prior).

The right panel relates together the z-score and the posterior shrinkage, demonstrating the superior precision of the FCD and combined data features over FC alone (higher shrinkage and lower z-score).

4.3. Deployment

4.3.1. Initial prototype

Work has been done iteratively, by putting together a first API version, generic enough to be deployed. The next step has been carried out to actually have a deployment on one of the TVB-Cloud platforms. Therefore, we have been in touch with the FRH team to understand which are the options to deploy at their site. They offered a VM where we could deploy a JupyterLab and run the inference locally, via jupyter notebooks, parallelizing the computation as much as possible on the available resources.

The deployment procedure is automated as a Jenkins job which connects to the FRH VM and via a Docker Compose setup proceeds to build a fresh docker image with the latest code from the tvb-inversion repository and restart the JupyterLab service to use that docker image for the environment.

For each user that connects to this JupyterLab instance, a new docker container is started up with the tvb-inversion package installed and notebooks with usage examples.

The deployment is available at the URL below and any user with a SCAI account can access it:

https://tvb-inversion.scai.fraunhofer.de

4.3.2. EBRAINS Collaboratory

In the second iteration, we focused on finding an additional solution to support a remote parallel execution of selected steps for tvb-inversion. The second step in the API, the priors sampling, has been identified as being the most extensive one, thus the goal for this iteration has been to demonstrate that it could be executed remotely, in a more efficient manner.

In order to demonstrate that such a setup would work for the tvb-inversion use-case, we have prepared a Collab within the EBRAINS platform which already provides HPC access via UNICORE. The architecture consists of a jupyter notebook using the tvb-inversion API to prepare an inference job, connect to HPC via UNICORE, stage-in the necessary data, schedule the job and finally, stage-out the results after the computation is finished.

At the HPC site, the sampling priors step of tvb-inversion will run inside a sarus container. During its run time, the tvb-inversion module installed in EBRAINS Lab will monitor the status of the HPC job, so, when it finishes, the results are brought back to the Lab site and used in the actual inference step which can run locally.

We find such a setup generic enough to work in any other environment that offers access to an HPC backend via UNICORE, therefore, the installation should be compatible with the FRH platform or the VRE after these tools are installed, as it seems to be intended.

The Collab with details on this setup is available at the URL below. Basic documentation on the installation steps and API usage can be found there, together with details on benchmarking and limitations encountered during testing:

https://wiki.ebrains.eu/bin/view/Collabs/tvb-inversion/

Regarding CI, we have a setup in Github Actions to build and push the tvb-inversion docker image used within the HPC to Docker Hub. Also, there is a setup to automatically run the basic unit test suite that is currently available for the package. The release procedure for Pypi is automated as a Github Action as well, ready to be used when the repository will become public.

4.3.3. Clusters with batch schedulers

The last version of the package simplified the deployment process and unified the sampling step with the parameter sweep API. It can be now easily used in any platform providing access to compute resources through scheduling system such as SLURM. This is a common setup in local clusters and in the supercomputing centres, and was deployed and tested in the JUSUF system operated by Julich Supercomputing Centre. In addition, an easily extensible demo is provided demonstrating the use of the parameter sweep API with UNICORE connection to HPC in the EBRAINS Collaboratory. Lastly, the provided examples in the form of Jupyter notebooks can be executed in an interactive computing environment e.g. through the JupyterLab which is today provided in most target platforms.

4.4. Performance evaluation

The Collab setup has been tested on CSCS and some of the results have been benchmarked and documented on the following page, same as the table below:
https://wiki.ebrains.eu/bin/view/Collabs/tvb-inversion/Benchmarking

Table 1: Performance of the reference implementation (backend) of the Montbrio-Pazo-Roxin model in TVB.

Model	Sim length (s)	Regions	Nr simulations	Nr workers	Execution time (hh:mm)
MontbrioPazoRoxin	30	100	30	30	00:17
MontbrioPazoRoxin	30	100	200	20	01:08
MontbrioPazoRoxin	30	100	300	30	01:10
MontbrioPazoRoxin	30	100	400	40	01:18
MontbrioPazoRoxin	30	100	500	50	01:34
MontbrioPazoRoxin	30	100	500	55	01:30
MontbrioPazoRoxin	30	100	600	55	01:45
MontbrioPazoRoxin	30	100	600	60	OOM
MontbrioPazoRoxin	60	100	500	40	03:07
MontbrioPazoRoxin	60	100	500	55	OOM

Tvb-inversion is based on the NbMPRBackend of TVB as it has been proved to be more efficient in terms of simulation computation than the classic TVB backend. Benchmarking between these two backend have been also performed as part of this story, and the results were added to the official TVB documentation available at this URL and in the table below:
https://github.com/the-virtual-brain/tvb-root/blob/master/tvb_documentation/doc_site/benchmarks/tvb_2.6.1_mac.rst

Connectivity:

- 100 Nodes - SC_Schaefer7NW100p_nolog10.txt
- 76 Nodes - tvb_data/connectivity/connectivity_76.zip

Integrator: HeunStochastic
Coupling: Linear
Monitor: TemporalAverage - period=0.1

Results				
Backend	Sim. length	Nodes	Time step	Execution time
	(ms)		(ms)	min:sec
Reference	1000	100	0.005	01:18
NbMPR	1000	100	0.005	00:07
Reference	20000	100	0.005	26:15
NbMPR	20000	100	0.005	01:55
Reference	1000	76	0.005	00:40
NbMPR	1000	76	0.005	00:05

Table 2: Benchmark results comparing the Numba and reference backends for the Montbrio Pazo Roxin model.

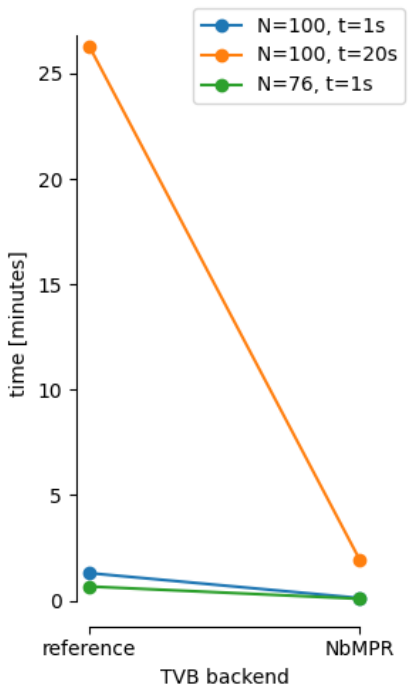


Figure 4: Performance comparison between the backends for MPR model as shown in Table 2.

To relate this to the simulation in the section 4.2.1: when using the Numba backend, each simulation of 5 minutes of BOLD signal for a set of parameters takes about 3 minutes, for 10K simulation this sums up to 500 core-hours.

5. Conclusion, next steps

The workflow and its implementation presented in this deliverable advanced significantly the state-of-the-art of parameter inference for virtual brain model personalization. Based in the Bayesian framework it provides more robust and reliable inference compared to approaches based in optimization, while being more flexible and computationally less demanding than likelihood-based approaches.

The developed workflow was implemented in a portable, easy-to-install software package and deployed on the representative systems in the TVB-Cloud consortium. The source codes together with technical documentation and usage examples are available under open source license in following GitHub repository:

https://github.com/the-virtual-brain/tvb-inversion

The next steps involve elaborating additional examples of the SBI-based inference in other model configurations (e.g. using different popular neural mass models), and different use-cases such as those representative of the neurodegenerative disease progression estimation.

6. References

Tejero-Cantero, A., Boelts, J., Deistler, M., Lueckmann, J.-M., Durkan, C., Gonçalves, P., Greenberg, D., & Macke, J. (2020). sbi: A toolkit for simulation-based inference. Journal of Open Source Software, 5(52), 2505.

Hashemi, M., Vattikonda, A. N., Jha, J., Sip, V., Woodman, M. M., Bartolomei, F., & Jirsa, V. K. (2022). Simulation-based inference for whole-brain network modeling of epilepsy using deep neural density estimators. In bioRxiv. https://doi.org/10.1101/2022.06.02.22275860

Cranmer, K., Brehmer, J., & Louppe, G. (2020). The frontier of simulation-based inference. Proceedings of the National Academy of Sciences of the United States of America, 117(48), 30055–30062.

Lavanga, M., Stumme, J., Yalcinkaya, B. H., Fousek, J., Jockwitz, C., Sheheitli, H., Bittner, N., Hashemi, M., Petkoski, S., Caspers, S., & Jirsa, V. (2022). The virtual aging brain: a model-driven explanation for cognitive decline in older subjects. In bioRxiv (p. 2022.02.17.480902). https://doi.org/10.1101/2022.02.17.480902

Jung, K. et al. (2021) 'Most appropriate RS-processing pipelines for TVB modelling finished', VirtualBrainCloud, ID 82641, H2020, Deliverable 3.2.

Montbrió, E., Pazó, D., & Roxin, A. (2015). Macroscopic Description for Networks of Spiking Neurons. Physical Review X, 5(2), 021028.

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., Greenberg, D. S., & Macke, J. H. (2020). Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, *9*. https://doi.org/10.7554/eLife.56261

Rezende, D., & Mohamed, S. (2015). Variational Inference with Normalizing Flows. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning* (Vol. 37, pp. 1530–1538). PMLR.

Betancourt, M. (2018). Calibrating Model-Based Inferences and Decisions. In *arXiv [stat.ME]*. arXiv. http://arxiv.org/abs/1803.08393