The Journal of Open Source Software

DSWL package: a Python implementation of the Debiased Spatial Whittle Likelihood

Arthur P. Guillaumin [1], **Thomas Goodwin** [5], **Olivia Walbert** [4], **Adam M. Sykulski** [2], **Sofia C. Olhede** [3], **and Frederik J. Simons** [4]

1 Queen Mary University of London, United Kingdom **2** Imperial College London, United Kingdom **3** Ecole Polytechnique Fédérale de Lausanne, Switzerland **4** Princeton University, United States of America **5** School of Economics, University of New South Wales, Australia

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: George K. Thiruvathukal

Reviewers:

- @MarineChap
- @weiji14

Submitted: 13 March 2025
Published: unpublished

Summary

The Debiased Spatial Whittle Likelihood (DSWL) package is an open-source Python package that implements the eponymous paper (Guillaumin et al., 2022). The methodology allows users to efficiently infer the parameters of stationary / homogeneous spatial and spatio-temporal covariance models for univariate or multivariate processes from gridded data with potential missing observations, e.g. due to natural boundaries. It leverages the Fast Fourier Transform, and therefore can benefit from further computational gains through GPU implementations offered by PyTorch (Paszke et al., 2019) or Cupy (Okuta et al., 2017), both made available within the package as alternative backends to Numpy (Harris et al., 2020). As such, DSWL on GPU allows to fit covariance models to data observed on grids with tens of millions of locations.

Statement of need

Describing patterns of spatial and spatio-temporal covariance is of interest to practitioners in a wide range of applied sciences such as geosciences, meteorology or climate science. Stationary covariance modelling allows for a first-order approximation of the covariance structure, and leads to many practical applications such as kriging (Stein, 1999) and forecasting via the conditional Gaussian multivariate distribution. The inference of parameters for a physics-based covariance model can also be of interest in its own right.

A major hurdle in spatio-temporal modelling is the computational cost of the Gaussian likelihood function. This is particularly relevant for modern spatio-temporal datasets, from physics simulations to real-world data. The computational burden of parameter inference also arises from complex spatio-temporal covariance models with a large number of parameters which typically require a high number of likelihood evaluations during the optimization process or when running an MCMC sampler.

A common means to circumvent this computational burden is to use approximations to the Gaussian likelihood. Among these methods, the Whittle likelihood is a standard spectral domain method for gridded data. Along its computational benefits, the Whittle likelihood provides robustness to departures from Gaussianity and allows to restrict the second-order model to a specific range of spatio-temporal frequencies. However, for spatial and spatio-temporal data where the dimension d is greater than 2, the standard Whittle likelihood suffers from a large bias and typically does not allow for missing observations.

`DSWL` is a Python implementation of the Debiased Spatial Whittle likelihood (Guillaumin et al., 2022), a method that addresses the bias of the Whittle likelihood (Sykulski et al., 2019). Although it requires gridded data as it relies on the Fast Fourier Transform, the implemented method additionally allows for missing observations, making it amenable to practical applications where a full hyperrectangle of data measurements might not be available. Missing observations might occur due to natural boundaries or due to measurement constraints. As an example, in Figure 1 we show a simulated sample from an exponential covariance model observed on a domain with the shape of metropolitan France (sans Corsica), along with the distribution of estimates obtained from 1000 independent samples generated from the same model and the predicted distribution of estimates, which can be used to build confidence intervals.

The package allows the user to treat multivariate data, including those cases where the missingness patterns might differ between the variates. For instance, in Figure 2 we show a realization of a bivariate random field with distinct patterns of missing observations between the two variates, from which we can still infer the parameters of the model, such as the correlation between the two fields. The code base also includes tapering, the use of which can further alleviate boundary effects (Dahlhaus & Künsch, 1987). Finally, the user can switch between several computing backends, Numpy, Cupy and PyTorch. This allows to further benefit from computational gains via GPU implementations of the Fast Fourier Transform. In practice, we observe computational GPU-versus-CPU speed-ups of order $\times 10$ up to order $\times 100$ as reported in Figure 3 (CPU: Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz, GPU: NVIDIA A100-PCIE 40GB, numpy 1.26.4, cupy 13.4.0).

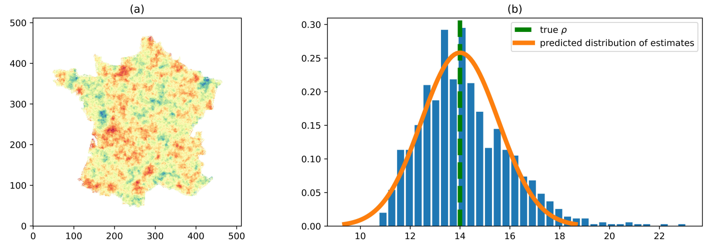

Figure 1: A simulated sample from an exponential covariance kernel observed on a domain with the shape of metropolitan France (sans Corsica) (a), along with the distribution of estimates obtained from 1000 independent realizations from the same model with range parameter $\rho = 14$ spatial units (b)

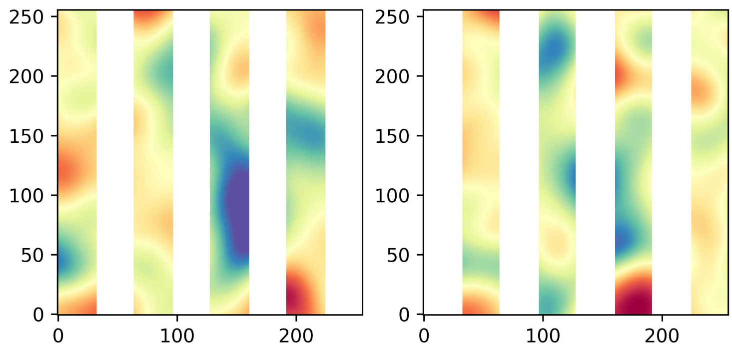

Figure 2: An example of a bivariate random field with distinct patterns of missing observations

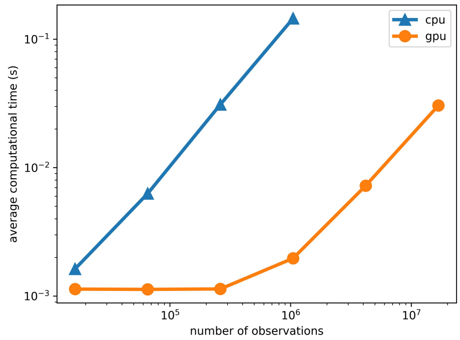

Figure 3: Computational time of the Debiased Spatial Whittle Likelihood averaged over 1000 samples on square grids of increasing sizes, compared between CPU and GPU (Cupy backend)

Other approximation techniques are available for the inference of spatio-temporal covariance models. Among those, we can mention Vecchia-type likelihood approximations (Katzfuss & Guinness, 2021) implemented e.g. by Jurek (2023), Katzfuss (2023) and Joseph Guinness (2023), and covariance tapering (Kaufman et al., 2008), although for the latter we are not aware of open-source implementations.

Software structure

The software is organized around several modules that can be grouped into the following categories:

- grids and sampling:
  - `grids.py`: this module is used to define the rectangular grids where the data sit via the class RectangularGrid. A mask of zeros (missing) and ones (not missing) can be set to specify potential missing observations, for instance to account for natural boundaries
  - `simulation.py`: this module allows to efficiently sample a realization from a model on a grid via circulant embedding (Dietrich & Newsam, 1997).
- models:
  - `models.py`: this module allows to define a covariance model. Standard covariance models are pre-defined, such as the exponential covariance model, the squared exponential (Gaussian) covariance model and the Matérn covariance model. These standard covariance models can also be combined (e.g. via summation) to form more complex covariance models.
- estimation:
  - `periodogram.py`: this module allows to compute the periodogram of the data, and to obtain the expected periodogram for a given model and grid combination.
  - `multivariate_periodogram.py`: this module allows to compute the periodogram for multivariate data.
  - `likelihood.py`: this module allows to define the Debiased Whittle Likelihood and the corresponding estimator. The optimizer can be selected among those offered by the optimize package of the Scipy library (Virtanen et al., 2020).

A documentation including example notebooks is available, and issues can be raised on Github. Example notebooks can also be run directly in the browser via mybinder.org.

The Journal of Open Source Software

Acknowledgements

This research utilised Queen Mary University of London's Apocrita HPC facility, supported by QMUL Research-IT. doi:10.5281/zenodo.438045. In particular, this research made use of the OnDemand portal (Hudak et al., 2018).

References

Dahlhaus, R., & Künsch, H. (1987). Edge effects and efficient parameter estimation for stationary random fields. *Biometrika*, *74*(4), 877–882. https://doi.org/10.1093/biomet/74.4.877

Dietrich, C. R., & Newsam, G. N. (1997). Fast and Exact Simulation of Stationary Gaussian Processes through Circulant Embedding of the Covariance Matrix. *SIAM Journal on Scientific Computing*, *18*(4), 1088–1107. https://doi.org/10.1137/S1064827592240555

Guillaumin, A. P., Sykulski, A. M., Olhede, S. C., & Simons, F. J. (2022). The Debiased Spatial Whittle Likelihood. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *84*(4), 1526–1557. https://doi.org/10.1111/rssb.12539

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hudak, D., Johnson, D., Chalker, A., Nicklas, J., Franz, E., Dockendorf, T., & McMichael, B. L. (2018). Open OnDemand: A web-based client portal for HPC centers. *Journal of Open Source Software*, *3*(25), 622. https://doi.org/10.21105/joss.00622

Joseph Guinness, Y. F., Matthias Katzfuss. (2023). *GpGp: Fast gaussian process computation using vecchia's approximation*. CRAN. https://cran.r-project.org/web/packages/GpGp/index.html

Jurek, M. (2023). pyMRA: Multi-resolution approximation for gaussian processes. In *GitHub repository*. GitHub. https://github.com/marcinjurek/pyMRA

Katzfuss, M. (2023). GPvecchia: Fast gaussian-process inference using vecchia approximations. In *GitHub repository*. GitHub. https://github.com/katzfuss-group/GPvecchia

Katzfuss, M., & Guinness, J. (2021). A General Framework for Vecchia Approximations of Gaussian Processes. *Statistical Science*, *36*(1). https://doi.org/10.1214/19-STS755

Kaufman, C. G., Schervish, M. J., & Nychka, D. W. (2008). Covariance Tapering for Likelihood-Based Estimation in Large Spatial Data Sets. *Journal of the American Statistical Association*, *103*(484), 1545–1555. https://doi.org/10.1198/016214508000000959

Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible library for NVIDIA GPU calculations. *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS)*. http://learningsys.org/nips17/assets/papers/paper_16.pdf

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 8024–8035. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

Stein, M. L. (1999). *Interpolation of spatial data: Some theory for kriging*. Springer New York, NY. https://doi.org/10.1007/978-1-4612-1494-6

Sykulski, A. M., Olhede, S. C., Guillaumin, A. P., Lilly, J. M., & Early, J. J. (2019). The debiased Whittle likelihood. *Biometrika*, *106*(2), 251–266. https://doi.org/10.1093/biomet/asy071

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2