Stilt: Easy Emulation of Time Series AR(1) Computer Model Output in Multidimensional Parameter Space

by Roman Olson, Kelsey L. Ruckert, Won Chang, Klaus Keller, Murali Haran, and Soon-Il An

Abstract A common problem in climate science and other fields is one of statistically approximating ("emulating") time series model output in parameter space. There are many packages for spatio-temporal modeling. However, they often lack focus on time series, and exhibit statistical complexity. Here, we present the R package **stilt** designed for simplified AR(1) timeseries Gaussian Process emulation, and provide examples relevant to climate modelling. Notably absent is Markov chain Monte Carlo estimation – a challenging concept to many scientists. We keep the number of user choices to a minimum. Hence, the package can be useful pedagogically, while still applicable to real life emulation problems. We provide functions for emulator cross-validation, empirical coverage, prediction, as well as response surface plotting. While the examples focus on climate model emulation, the emulator is general and can be also used for kriging spatio-temporal data.

Introduction

Emulation of computer model behavior in parameter space is a challenging problem (Drignei et al., 2008; Higdon et al., 2008; Holden et al., 2010; Bhat et al., 2012; Olson et al., 2012; Sexton et al., 2012; Olson et al., 2013; Chang et al., 2014a,b). Often a limited number of runs are available, and we desire to estimate model output at a variety of new parameter settings (Drignei et al., 2008; Holden et al., 2010; Bhat et al., 2012; Olson et al., 2012; Sexton et al., 2012; Olson et al., 2013; Chang et al., 2014b). Emulation is often used in context of another common problem, input parameter estimation. Here, we desire to find probability distribution functions for model input parameters given the observational data (e.g., Olson et al., 2012; Chang et al., 2014b). While model output is often multi-dimensional, here we restrict the discussion to time series (although our approach can apply more generally to vector output).

Gaussian processes (Kennedy and O'Hagan, 2001; Rasmussen and Williams, 2006; Higdon et al., 2008; Rougier, 2008) are a very useful methodology for such emulation. Gaussian processes provide the best linear unbiased prediction under highly general conditions (Stein, 1999). This methodology assumes that model response is a smooth function of parameter settings. Central to the method is formulation of the covariance function, which quantifies the covariance between model output as a function of distance in each model input parameter. This covariance is typically parametrized by parameters controlling, for example, its magnitude, or the correlation ranges in each coordinate dimension. Associated with emulator parameters is a likelihood function given the model output. There are two main approaches: (i) find the "best" parameter setting that maximizes this likelihood or (ii) find the full probability distribution of the parameters. The next step is predicting at new parameter settings. The former approach ignores the uncertainty in emulator parameters during the prediction, while the latter approach accounts for it. The prediction is probabilistic as it includes the predictive mean and the uncertainty around it.

There are many existing R packages to perform Gaussian process emulation for vector, and specifically time series output. They include **gstat** (Gräler et al., 2016), **mlegp** (Dancik and Dorman, 2008; Dancik, 2013), **spBayes** (Finley et al., 2015), **ramps** (Smith et al., 2008), **spTimer** (Bakar and Sahu, 2015a,b) and **RandomFields** (Schlather et al., 2015). These spatio-temporal packages are usually general and do not focus on time series specifically. Moreover, they are usually presented in context of spatial interpolation ("kriging") rather than emulation. They may provide flexible functionality in terms of covariance functions (**RandomFields**), handling replicate (**mlegp**) or missing observations (**spTimer**), or considering both areal as well as point observations (**ramps**). The procedure for fitting emulator parameters differs between packages. Some packages use maximum likelihood or another form of optimization to find the best parameters, whereas others include a Bayesian analysis. This means that prior beliefs in emulator parameters are combined with information provided by the model output, to find the posterior probability distribution of the parameters.

Yet, due to their statistical complexity the broad scientific community may find it challenging to use such software. Particularly, the concept of Markov chain Monte Carlo (MCMC), which is employed in some work, may be unfamiliar to some scientists. In addition, a large variety of modeling or prior choices may leave a user struggling with which model or prior they should employ and why. Many packages appear to be tailored to a statistical audience, and elaborate statistical terminology may be

beyond the grasp for some scientists. Second, the packages are not usually focused on spatio-temporal output ubiquitous in computer modelling. Thus, scientists may invest substantial time in finding the right function among the plethora available. Finally, there is an issue of terminology: many scientists may not realize that "spatio-temporal modeling" may disguise the same technique as "emulation", and hence may be oblivious to the applicability of these packages.

Here, we document the R package **stilt** version 1.3.0 for simplified Gaussian Process AR(1) time series emulation with a focus on climate modeling. The package name has been inspired by the elegant namesake bird "stilt". The package is available on Comprehensive R Archive Network (CRAN) (Olson et al., 2017). The main differences from prior approaches are (i) a simplified framework with less modeling choices and no MCMC that is still applicable to real-life problems, and (ii) focus on emulation of time-series in parameter space. Specifically, there is only one function to fit the emulator to model output, and only one function for prediction. Statistical modeling accounts for random noise in the model output through the nugget term, and also allows for user-chosen linear terms in model parameters and/or time. Mathematically, this is the first public implementation of a separable covariance model of Rougier (2008). In this model, space-time covariance between two locations is a product of a space covariance and a time covariance term. This allows the use of matrix algebra to achieve considerable computational gains. We include useful utility functions for extensive cross-validation, 2D response surface plotting, and empirical 95% prediction interval coverage. In three examples, we apply **stilt** to emulation in spaces ranging from one- to five-dimensional. Note that whilst the software is for emulation of time series output in parameter space, the package is general and can be used for interpolation of observations (or model output) in geographical coordinates (e.g., Cressie and Johannesson, 2008; Jones et al., 2009; Hansen et al., 2010; Bhat et al., 2012; Hirahara et al., 2014).

In the rest of the paper we first outline the statistical approach, then give three examples (in 1D, 2D, and 5D parameter space), and finally present some concluding remarks. We provide some technical details in the Appendix.

Statistical approach

Overview of Approach

Here, we briefly describe our statistical approach. Our statistical model includes optional linear terms in parameters and/or time, a smooth Gaussian process, and a purely random nugget term. The Gaussian process represents smooth non-linear effects of parameters or time on the model output. The nugget term represents purely random effects. For climate model output, this corresponds to internal climate variability.

The first step is fitting the Gaussian process statistical model to the computer model output. Specifically, the statistical model has several parameters. They control the linear slopes in parameters and time, the correlation ranges of the field in each of the input parameters, the temporal autocorrelation, the overall magnitude of the covariance, and the nugget strength. Associated with the emulator parameters is the likelihood which quantifies how likely the emulator parameter values are given the model output. During the emulator fitting we vary the parameters so that this likelihood is maximized to obtain the optimized emulator. We use this optimized emulator for prediction. The prediction follows the standard Gaussian process theory. The rest of the section describes the details of the statistical methodology, and can be skipped on the first reading of the paper.

Statistical model

We base our implementation on standard Gaussian process theory (Cressie, 1993; Stein, 1999; Rasmussen and Williams, 2006). Consider the case of interpolating spatio-temporal model output of a perturbed parameter model ensemble in parameter space. Let $y_{i,j} \in R$ be physical model output at a parameter setting $\boldsymbol{\theta_i}$ and a time t_j. Time values form an n-dimensional regularly spaced vector $\boldsymbol{t} = (t_1, ..., t_n)^T$.

Each parameter setting is an m-dimensional vector: $\boldsymbol{\theta_i} = (\theta_{1,i}, ..., \theta_{m,i})$. For all p model runs, the parameter settings $\boldsymbol{\theta_i}$ form a $p \times m$ parameter matrix Θ. (Note that this notation differs from one commonly used in statistics and other applied areas where parameters, designated by $\boldsymbol{x_i}$, form a parameter matrix $\mathbf{X}$). Furthermore, $\boldsymbol{y_j} = (y_{1,j}, ..., y_{p,j})^T$ is a p-dimensional vector of model outputs for all p parameters for time t_j. Consecutively, the stacked $pn \times 1$ column matrix of all model output for times from 1 through n is $\mathbf{Y} = (\boldsymbol{y_1}^T, ..., \boldsymbol{y_n}^T)^T$. Associated with $\mathbf{Y}$ is the $pn \times (m+1)$ design matrix $\mathbf{D}$.

Its columns represent parameters and time, whereas rows correspond to elements of $\mathbf{Y}$. We calculate the design matrix $\mathbf{D}$ as:

$$\mathbf{D} = \left(\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{n \times 1} \otimes \Theta \qquad \mathbf{t} \otimes \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{p \times 1} \right), \tag{1}$$

where $\otimes$ is the Kronecker product. We model the output as a Gaussian process such that

$$\mathbf{Y} \sim N(\boldsymbol{\mu}_{\boldsymbol{\beta}}, \Sigma(\boldsymbol{\xi}_{\boldsymbol{y}})), \tag{2}$$

where $\boldsymbol{\mu}_{\boldsymbol{\beta}}$ is a mean function that is either constant or linear in any combination of parameters and/or time, and $\boldsymbol{\xi}_{\boldsymbol{y}}$ is a vector of covariance matrix parameters. We assume that $\boldsymbol{\mu}_{\boldsymbol{\beta}} = \mathbf{X}\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is a column vector of regression coefficients, and $\mathbf{X}$ is a matrix of covariates. It always includes the column of ones as its first column to represent the intercept. Depending on the number of regressors, it can additionally have corresponding columns of the design matrix $\mathbf{D}$. As an example, for a mean function that is linear in time, $\boldsymbol{\beta}$ has dimensions of 2×1, and $\mathbf{X}$ is $pn \times 2$:

$$\mathbf{X} = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_n \end{bmatrix}_{n \times 2} \otimes \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{p \times 1.} \tag{3}$$

Under the assumption of separability (see Rougier, 2008), we can represent the covariance matrix Σ as a Kronecker product of a separate covariance matrix in time Σ_t and in parameters Σ_θ:

$$\Sigma = \Sigma_t \otimes \Sigma_\theta. \tag{4}$$

This means that the covariance between any two locations in time and parameter space is a product of the time covariance term, and the parameter covariance term. This is a different approach to speed up the computation to, for example, SVD of model output (Dancik and Dorman, 2008; Dancik, 2013; Higdon et al., 2008; Chang et al., 2014b). We assume that the time covariance matrix Σ_t $(n \times n)$ has an AR(1) structure. AR(1) dependence in time is a feature of many environmental processes (e.g., Hasselmann, 1976; Keller and McInerney, 2008; Olson et al., 2013). To avoid identifiability issues, we do not use any multipliers for this matrix. Specifically, its (j, k) element is:

$$\varsigma_{t,jk} = \frac{\rho^{|t_j - t_k|}}{1 - \rho^2}, \tag{5}$$

where ρ is the lag-1 autocorrelation parameter. We assume that the parameter covariance Σ_θ $(p \times p)$ is squared exponential, as in the package **mlegp** (Dancik and Dorman, 2008; Dancik, 2013). The squared exponential covariance function is frequently used in computer model emulation, as computer model outputs can be often represented using a highly smooth Gaussian process. The (i, j) element of Σ_θ is:

$$\varsigma_{\theta,ij} = \kappa \exp(-\sum_{k=1}^{m} \frac{|\theta_{k,i} - \theta_{k,j}|^2}{\phi_k^2}) + \zeta 1(i = j). \tag{6}$$

Here κ is partial sill, ζ is nugget, and ϕ_k is range parameter for the k^{th} model input parameter. The range parameters form a vector $\boldsymbol{\phi} = \phi_1, ..., \phi_m$. We construct the total covariance matrix $(np \times np)$ as:

$$\Sigma = \begin{bmatrix} \varsigma_{t,11}\Sigma_\theta & \cdots & \varsigma_{t,1n}\Sigma_\theta \\ \vdots & \ddots & \vdots \\ \varsigma_{t,n1}\Sigma_\theta & \cdots & \varsigma_{t,nn}\Sigma_\theta \end{bmatrix}. \tag{7}$$

Hence, the covariance parameters are $\boldsymbol{\xi}_{\boldsymbol{y}} = (\rho, \kappa, \boldsymbol{\phi}, \zeta)^T$. The emulator parameters are $\boldsymbol{\psi} = (\boldsymbol{\beta}^T, \boldsymbol{\xi}_{\boldsymbol{y}}{}^T)^T$. The actual number of emulator parameters will be different depending on the number of model parameters that the ensemble varies and on the number of covariates.

Estimating emulator parameters

We can write the log-likelihood for the model output $\mathbf{Y}$ given the emulator parameters $\boldsymbol{\psi}$ as (see e.g, Rasmussen and Williams, 2006):

$$\ln L(\mathbf{Y}|\boldsymbol{\psi}) = -\frac{1}{2}(\mathbf{Y}-\boldsymbol{\mu}_{\boldsymbol{\beta}})^T\Sigma^{-1}(\mathbf{Y}-\boldsymbol{\mu}_{\boldsymbol{\beta}}) - \frac{1}{2}\ln|\Sigma| - \frac{np}{2}\ln 2\pi. \tag{8}$$

Under the uniform priors for the emulator parameters:

$$L(\boldsymbol{\psi}|\mathbf{Y}) \propto L(\mathbf{Y}|\boldsymbol{\psi}), \tag{9}$$

consequently maximizing the log-likelihood for the model output also maximizes the likelihood for the parameters. Note that this likelihood evaluation involves a computationally expensive inverse of an $np \times np$ matrix. However, we can reduce the dimension of the inverted matrices to $p \times p$ and $n \times n$, and speed up the computation (Magnus and Neudecker, 2007, Appendix). Thus, the computational cost of inversion becomes $O(p^3)$ or $O(n^3)$, as opposed to $O([pn]^3)$. Further computational savings accrue when calculating $|\Sigma|$, considering that Σ is a Kronecker product of two positive definite matrices (Gentle, 2007, Appendix):

We optimize the emulator parameters $\boldsymbol{\psi}$ by maximizing the likelihood function over a reasonable parameter range using a local optimization routine. Specifically, **stilt** uses the **nlminb** function which calls FORTRAN code (Gay, 1990). In the package, there is an option to either fix $\boldsymbol{\beta}$ parameters at their multiple linear regression estimates, or to optimize them along with other emulator parameters.

Prediction

We are interested in predicting model output for all times for a new parameter setting $\boldsymbol{\theta}^*$. We denote this output, an n-dimensional vector, by $\boldsymbol{y}^* = (y_{\boldsymbol{\theta}^*,1}, ..., y_{\boldsymbol{\theta}^*,n})^T$. Associated with the prediction parameter setting is an $n \times 1$ prediction design matrix $\mathbf{D}^*$ and a matrix of covariates $\mathbf{X}^*$ evaluated at predictions points. It is constructed similarly to $\mathbf{X}$. To give an example, for a mean function that is linear in time, $\mathbf{X}^*$ is:

$$\mathbf{X}^* = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_n \end{bmatrix}_{n\times 2.} \tag{10}$$

The prediction is given by the following multivariate normal distribution (Rasmussen and Williams, 2006):

$$\boldsymbol{y}^* \sim N(\boldsymbol{\mu}^*_{\boldsymbol{\beta}}, \Sigma^*). \tag{11}$$

Here,

$$\boldsymbol{\mu}^*_{\boldsymbol{\beta}} = \mathbf{X}^*\boldsymbol{\beta} + (\Sigma_t \otimes \Sigma_{\theta^*\theta})\Sigma^{-1}(\mathbf{Y}-\boldsymbol{\mu}_{\boldsymbol{\beta}}), \tag{12}$$

where $\Sigma_{\theta^*\theta}$ is a $1 \times p$ cross-covariance matrix between the prediction parameter setting and all the ensemble parameters settings. For this matrix, we use the same covariance function as for Σ_θ. To evaluate the mean function, we need the inverse of the Σ matrix ($np \times np$). However, using matrix algebra (Golub and Van Loan, 1996; Rougier, 2008), we can reduce the dimension to $p \times p$ by writing $\boldsymbol{\mu}^*_{\boldsymbol{\beta}}$ in the following way:

$$\boldsymbol{\mu}^*_{\boldsymbol{\beta}} = \mathbf{X}^*\boldsymbol{\beta} + (\mathbf{I}_{n\times n} \otimes \Sigma_{\theta^*\theta}\Sigma_\theta^{-1})(\mathbf{Y}-\boldsymbol{\mu}_{\boldsymbol{\beta}}). \tag{13}$$

The predictive covariance also requires an inversion of a $p \times p$ matrix only:

$$\Sigma^* = (\kappa+\zeta)\Sigma_t - \Sigma_t \otimes \Sigma_{\theta^*\theta}\Sigma_\theta^{-1}\Sigma^T_{\theta^*\theta}. \tag{14}$$

Prediction for many points uses the same inverse parameter covariance. Hence, we can calculate the inverse once and recycle it for prediction at many points. This further enhances computational savings.

Examples using stilt

Preparing model output

Here, we describe how to prepare model output for use with the **stilt** emulator. Two R list objects are required. These objects can be easily prepared by the user before the start of the analysis. The first, `mpars`, contains information on parameter settings in the ensemble. It has the following components: `$par`,`$parnames` and `$parunits`. `$par` is the actual matrix of parameter settings, with rows corresponding to parameter index, and columns – to model run index. `$parnames` is a vector of parameter names corresponding to the rows of `$par`, while `$parunits` is a vector of units. On the other hand, `$moutput` is an object with information on model output. It also has several components. Here, `$out` is the actual model output matrix, with rows referencing time, and columns referencing model run. Other elements contain metadata: `$t` is the corresponding time vector, `$tunuts` are time units, `$outname` is the name of the modeled variable, and `$outunits` are the corresponding units.

1D toy model example

This and the following examples are based on running R version 3.3.3 on a 3 Ghz Intel core i5 16GB 2400 MHz DDR4 Macintosh 10.13.6 computer. The following versions were used for other required packages: fields 9.0, maps 3.3.0, spam 2.1-2 and dotCall64 0.9-5. The results were observed to differ slightly according to the programming environment and the operating system.

We first consider interpolating a toy model dataset consisting of a simple time-series output for one parameter with a total of 21 parameter settings and 11 time settings. The output of this simple model as a function of time t and parameter θ is: $y = \sin(\theta)(1 + 2t + t^2)$. The model is evaluated for $\theta = (0, 1, ..., 20)$, and each run produces a time series for times $t = (0, 1, ..., 10)$. First, we can plot model output for all parameters:

```
R> data("Data.1D.model")
R> data("Data.1D.par")
R> plot(NA, xlim=c(0, 11), ylim=c(-150, 150), xlab="Time", ylab="Model output")
R> for (c in 1:21) {lines(Data.1D.model$t, Data.1D.model$out[,c])}
```

The output of this code is shown in Figure 1.

Now, we fit a Gaussian process emulator to these data. While we do not use any parameter covariances (`par.reg=FALSE`), we do use a linear time covariate (`time.reg=TRUE`). The optimization follows the default behavior of fixing the linear regressors at the multiple linear regression estimates. We select starting values for κ and ζ of 100 for both, because this leads to reasonable optimization results.

```
R> emul1D = emulator(Data.1D.par, Data.1D.model, par.reg=FALSE, time.reg=TRUE,
kappa0=100, zeta0=100)
Initializing the emulator...

Initial regression parameters:
-0.665481 0.570413

Initial emulator likelihood is: -960.2755

Optimizing the emulator...
Obtaining emulator parameter ranges for optimization...

Relative tolerance to be used in optimization: 1e-10

Option 'fix.betas': during optimization beta parameters are going to be
fixed at the following values:
-0.665481 0.570413

------------------------------------
Starting parameter optimization...
------------------------------------
  0:     960.27550: 0.900000  100.000  100.000  10.0000
  1:     947.88498: 0.903188  72.2395  233.781  10.1709
  2:     898.20175: 0.957801 0.00240862  151.661  13.5627
  3:     889.24528: 0.977342 0.00240862  131.201  13.4426
```

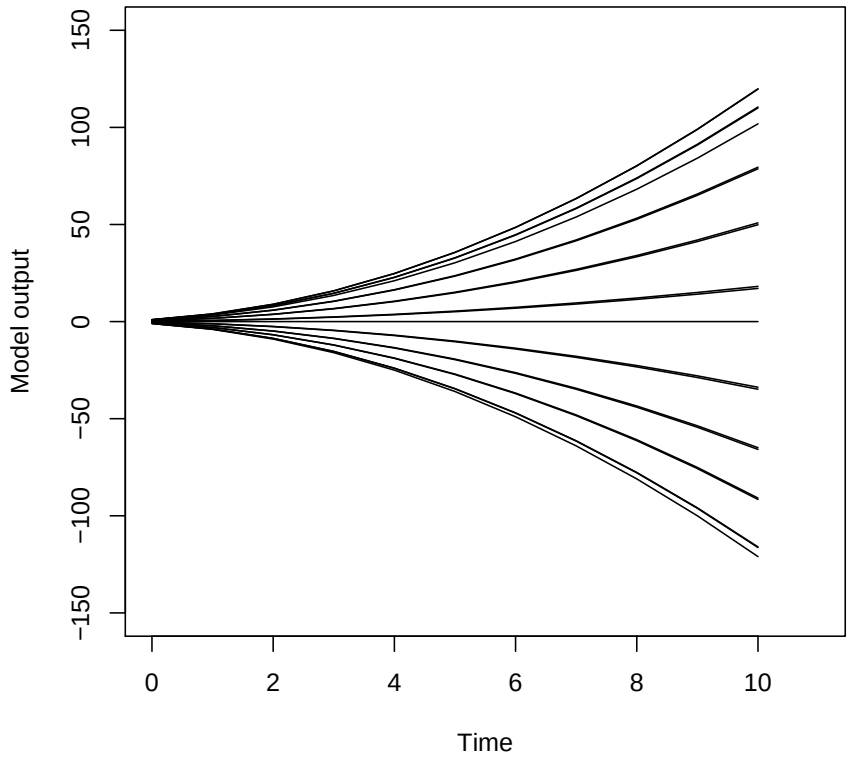

Figure 1: Sample time series output of the toy model for different parameter settings

```
 48:     464.48240: 0.982420  1076.06 0.00240862  3.93464
Optimization SUCCESSFUL! Optimization message below:

relative convergence (4)

Final parameterss
rho kappa zeta phi
0.98242004 1076.05714589 0.00240862 3.93464218

48 iterations were performed
Final likelihood = -464.4824

CAUTION! The optimization might only find a local minimum.
```

The package informs us of key emulator parameter settings, and of optimization results (for brevity only the start and the end of optimization process are shown). The list of parameters to be optimized is, in this case, ρ, κ, ζ and ϕ_1. The final emulator is a highly autocorrelated process with a large partial sill, but a very low nugget. The emulator object `emul1D` is a list with many components, with a secondary custom "emul" class. The components include information on the data used to fit the emulator, optimized emulator parameters, some settings used during optimization, time and parameter covariance matrices, the inverse of the parameter covariance matrix, etc. Now, we validate the emulator using one-at-a-time cross-validation for time index 9 (close to the end of the time series). Specifically, we remove each parameter setting from the ensemble one at a time, and use the emulator to predict at the excluded parameter setting, given the output at the other parameter settings.

```
R> test.all(emul1D, 9)
```

The emulator mean prediction shows a remarkable accuracy at predicting actual model response: the two fall almost perfectly on a 1:1 line of a reliability diagram (Figure 2). The emulator prediction error is a very small fraction of 1% for almost all runs (Figure 2). Note that the emulator does not extrapolate beyond the ensemble parameter range.

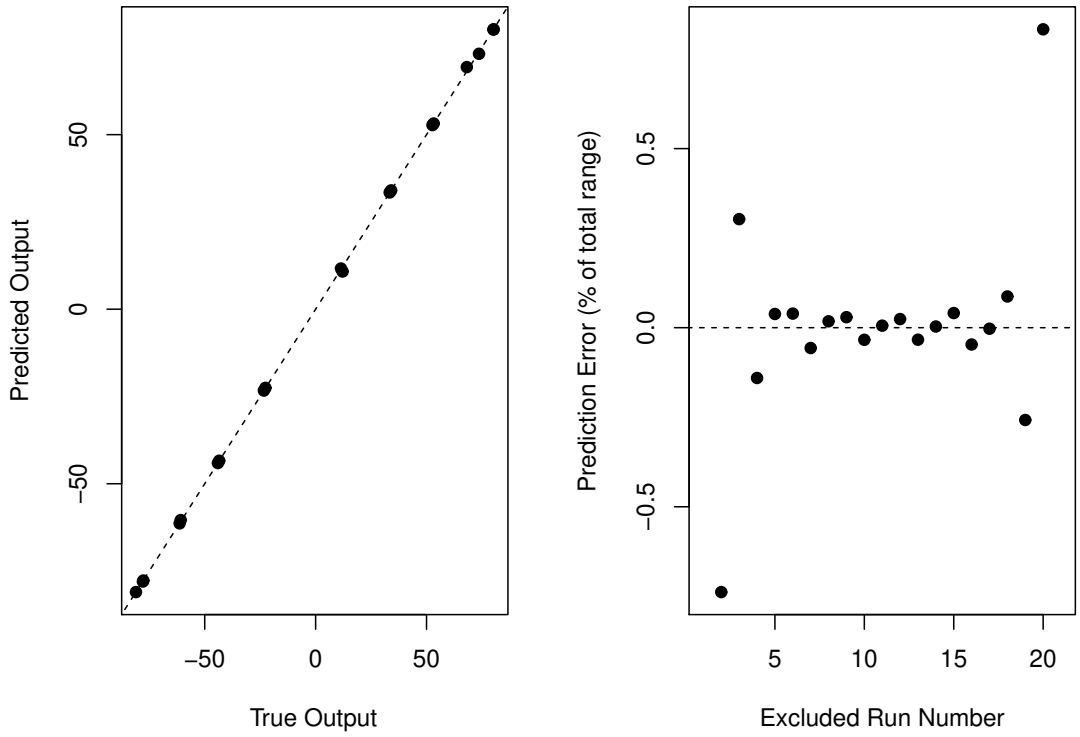

Figure 2: Toy example emulator one-at-a-time cross-validation results: (left) predicted vs. actual model output (black dots) and a 1:1 line; (right) relative prediction error as a function of the excluded model run number.

The emulator predicts at new parameter settings using the function `predict.emul`. This is implemented using the S3 generic "predict" method: it dispatches to the predict.emul function for class "emul". This function returns an object with components `mean` and `covariance` representing the mean and the covariance of the prediction.

A more challenging test case: 2D Korean summer mean maximum temperature variability and change

Next, we consider Korean summer mean maximum temperature output from 29 Coupled Model Intercomparison Project phase 5 (CMIP5) climate models (Taylor et al., 2012) for years 2081-2100 for the RCP8.5 forcing scenario (Moss et al., 2010). All of the models and calculations used here follow Shin et al. (2018) with the exception of the IPSL_CM5A_LR, which is not used. Of interest to us is the relationship between present-day (years 1973-2005) sample red noise properties of annual temperature time-series and future mean maximum summer temperature changes in these models. Hence, each model i is associated with $\boldsymbol{\theta_i} = (\sigma_i, \rho_i)$, where σ_i is sample innovation standard deviation, and ρ_i is sample first-order autocorrelation. Thus, in this case, $m = 2$, $p = 29$, and $n = 20$.

First, we load the relevant datasets and plot the temperature time series for all models.

```
R> data("Data.AR1Korea.model")
R> data("Data.AR1Korea.par")
R> mycolors = rainbow(29)
R> plot.default(NA, xlim=c(2081, 2100), ylim=c(0,10), xlab="Year",
```

```
            ylab="JJA Mean Max Temp Anomaly wrt. 1973-2005 [K]")
R> for (c in 1:29) {lines(Data.AR1Korea.model$t, Data.AR1Korea.model$out[,c],
                    col=mycolors[c])}
```

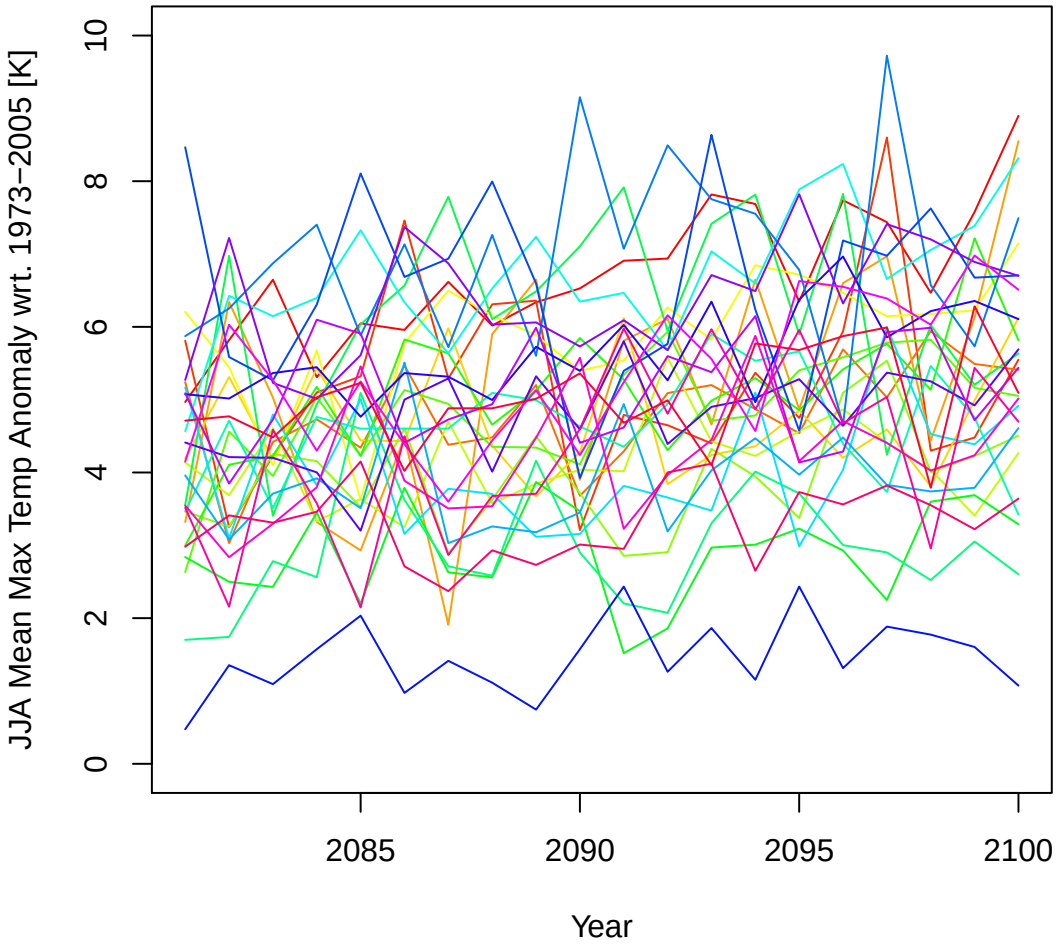

Figure 3: Korean summer mean maximum temperature change in years 2081-2100 with respect to period 1973-2005 for 29 CMIP5 climate models.

We note that the models show a considerable internal variability superimposed on a trend of slow warming (Figure 3). Now, we are going to fit an emulator, while also allowing for optimization of regression slopes in innovation standard deviation and time. We are also going to use a custom relative tolerance, to illustrate the capacity to change this parameter to fit the needs of a user.

```
R> emul = emulator(Data.AR1Korea.par, Data.AR1Korea.model, par.reg=c(TRUE,
FALSE), time.reg=TRUE, kappa0=1, zeta0=1, myrel.tol=1E-9, fix.betas=FALSE)

Initializing the emulator...

Initial regression parameters:
-116.0626827 3.8544881 0.0563853

Initial emulator likelihood is: -937.479

Optimizing the emulator...
```

```
Obtaining emulator parameter ranges for optimization...

Relative tolerance to be used in optimization: 1e-09


-----------------------------------
Starting parameter optimization...
-----------------------------------
 0: 937.479: 0.900000  1.00000  1.00000 -116.063  3.85449 0.0563853 0.322532 0.257
 1: 923.956: 0.899929 0.468880  1.36686 -116.087  3.85444 0.0563734 0.322768 0.257
 2: 901.460: 0.899874 9.24879e-05  1.05990 -116.096  3.85442 0.0563689 0.322900 0.257
 3: 900.363: 0.899860 0.0377996  1.17131 -116.068  3.85445 0.0563831 0.322900 0.257

54: 858.446: 0.616698 9.24879e-05  1.10326 -123.432 3.86546 0.0599039 6.45065 4.396
Optimization SUCCESSFUL! Optimization message below:

relative convergence (4)

Final parameterss
rho kappa zeta beta beta beta phi phi
0.616698 9.24879e-05  1.10326 -123.432 3.86546 0.0599039 6.45065 4.396

54 iterations were performed
Final likelihood = -858.4463

CAUTION! The optimization might only find a local minimum.
```

Note that we format the output slightly to fit the page width. We see that the initial multiple regression provides a reasonable mean and parameter slopes: they are close to the final optimized results (fourth through sixth elements of the optimized parameter vector called "Final parameters" above). There is a considerable linear dependence on both time and innovation standard deviation. The non-linear part of the emulator has a strong random component, and a very weak Gaussian process component. This suggests no systematic dependence of the model output on the present-day autocorrelation. We can predict the temperature response at an arbitrary setting of the parameters using the `predict` function. We do this for $\sigma = 1$ and $\rho = 0$:

```
R> pred = predict(emul, c(1, 0))
R> plot.default(NA, xlim=c(2081, 2100), ylim=c(3.5,7.5), xlab="Year",
            ylab="JJA Mean Max Temp Anomaly wrt. 1973-2005 [K]")
R> lines(emul$t.vec, pred$mean)
R> lines(emul$t.vec, pred$mean + sqrt(diag(pred$covariance)), col="brown")
R> lines(emul$t.vec, pred$mean - sqrt(diag(pred$covariance)), col="brown")
```

The mean vector of the prediction is the `$mean` component of `pred`, and the variance vector is composed of the diagonal entries of `$covariance`. Figure 4 shows the predicted response, with the associated 1-std uncertainty. We note the linearity of the warming in time. This illustrates the ability of the emulator to identify fluctuations of temperature in the models around the linear trend as random, and to not include them into the emulated response.

An even more challenging test case: Five-dimensional ice sheet model output

Next, we consider a 5D emulator for the SICOPOLIS ice sheet model (Greve, 1997; Greve et al., 2011) output of Greenland ice mass loss relative to the year 2003. This is a perturbed parameter 100-member ensemble which varies five model parameters: flow enhancement factor, basal sliding factor, geothermal heat flux, snow positive degree-day (PDD) factor, and ice PDD factor. The future forcing scenario is that of a gradual temperature increase stabilizing at approximately 5 K warmer than present (Applegate et al., 2012). Output is available annually between years 1840 and 2500.

We load relevant data and fit an emulator to the ensemble using all five parameters and time as covariates. For computational expediency, we fix slope parameters at their multiple regression estimates.

```
R> data(Data.Sicopolis.par)
R> data(Data.Sicopolis.model)
R> emul.Sicopolis = emulator(Data.Sicopolis.par, Data.Sicopolis.model,
```

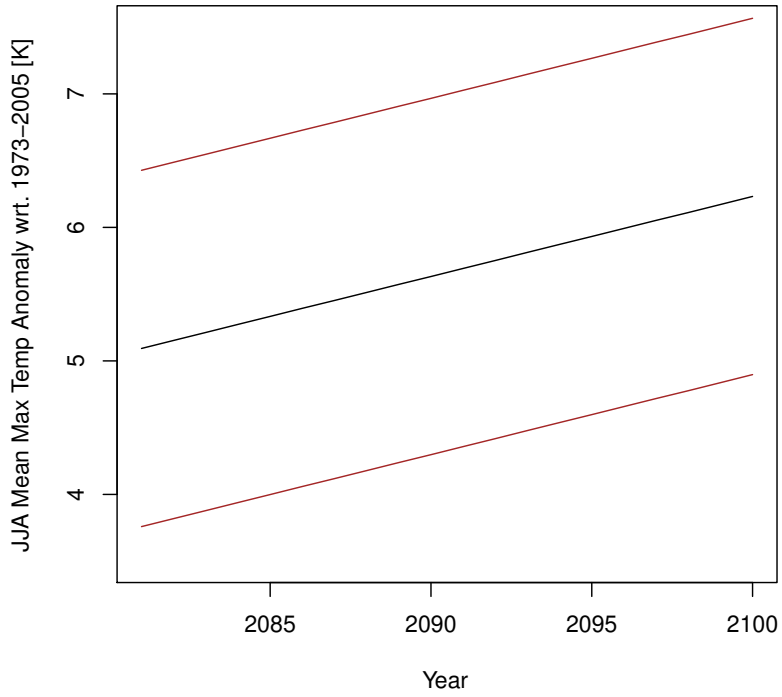

Figure 4: Emulated Korean summer mean maximum temperature change projections with respect to 1973-2005 at $\sigma = 1$ K and $\rho = 0$. Solid and dashed lines: mean response $\pm$ standard errors.

```
        par.reg=c(TRUE, TRUE, TRUE, TRUE, TRUE),
        time.reg=TRUE, kappa0=1000000, zeta0=50000)
Initializing the emulator...

Initial regression parameters:
5154878.375 171.401 8590.267 -182.659 49920.313 1300.488 -2725.378

Initial emulator likelihood is: -10782007

Optimizing the emulator...
Obtaining emulator parameter ranges for optimization...

Relative tolerance to be used in optimization: 1e-10

Option 'fix.betas': during optimization beta parameters are going to be
fixed at the following values:
5154878.375 171.401 8590.267 -182.659 49920.313 1300.488 -2725.378

-----------------------------------
Starting parameter optimization...
-----------------------------------
 0:     10782007.: 0.900 1.00e+06  50000.0  1.97305  9.84502  19.9205  1.98380  7.45
 1:     672836.29:  1.00 3.29e+07 7.05408e+07  2.83691  8.68834  27.1292  2.24728 0.00149
 2:     672613.27: 0.999 3.29e+07 7.05341e+07  2.83691  8.68834  27.1292  2.24728 0.0537
 3:     667210.63: 0.997 3.20e+07 7.08502e+07  2.86488  8.69561  27.3808  2.26410  14.54

79:     485611.17: 0.999989 5.83e+06  41529.0  13.4764  20.1003  199.578  5.72313  10.9
Optimization SUCCESSFUL! Optimization message below:
```

```
relative convergence (4)

Final parameters
rho kappa zeta phi phi phi phi phi
0.999989 5829746.770135 41528.993339 13.476403 20.100261 199.578404 5.723128 10.901509

79 iterations were performed
Final likelihood = -485611.2

CAUTION! The optimization might only find a local minimum.
```

The emulator takes roughly 3 minutes to fit on a 3 Ghz Intel core i5 16GB 2400 MHz DDR4 Macintosh computer. The final emulator is very smooth (as evidenced by the relatively high range parameters compared to the ensemble parameter range), and has an extremely low nugget compared to the partial sill parameter.

Now, we perform cross-validation of the emulator for the entire time series. We withhold three ensemble members, and predict at the withheld parameter settings using the model output at the remaining 97 settings.

```
R> test.csv(emul.Sicopolis, num.test=3, plot.std=TRUE, theseed=13241240)
Predicting for run number:  3
Predicting for run number:  26
Predicting for run number:  93
```

Here, we have specified a random seed, and the code informs us of the model runs that were excluded. Alternatively, we can select the runs to withhold via the `test.runind` argument. We present the results in Figure 5. The emulator has a remarkable skill at recovering the output of the withheld models.

Now, we withhold more runs and perform a more systematic analysis of emulator behavior:

```
R> mytest = test.csv(emul.Sicopolis, num.test=10, plot.std=FALSE, theseed=13241240,
                  make.plot=FALSE)
Predicting for run number:  3
Predicting for run number:  7
Predicting for run number:  26
Predicting for run number:  34
Predicting for run number:  37
Predicting for run number:  43
  ...Prediction error. Likely because prediction parameters are out of bounds
Predicting for run number:  91
Predicting for run number:  93
Predicting for run number:  99
Predicting for run number:  100
NOTE: 1  prediction points were omitted

R> cat("95% CI coverage:", mytest$coverage, "\n")
95% CI coverage: 0.9768028
```

Note that **stilt** does not extrapolate beyond the parameter range of the ensemble. Since one of the parameters is at its maximum among the ensemble for the 43^{rd} run, this run is skipped during the cross-validation. We disable the plotting since our main interest here is empirical coverage of the 95% prediction interval. The coverage (0.9768028) is relatively close to the ideal theoretical value of 0.95. This suggests that the emulator is relatively well calibrated.

We finish with plotting the response surface of the emulator for parameters 4 (snow PDD factor), and 5 (ice PDD factor). We fix flow enhancement factor, basal sliding factor, and geothermal heat flux at values of 3.0, 10.0 $\mathrm{m\,y^{-1}\,Pa^{-1}}$, and 45.0 $\mathrm{m\,W\,m^{-2}}$, respectively.

```
R> rsurface.plot(emul.Sicopolis, parind=c(4,5), parvals=c(3, 10, 45, NA, NA),
              tind=600, n1=10, n2=10)
```

We look at the 600^{th} time index (year 2439). `n1` and `n2` are the number of grid points to use in the x and y directions, respectively. Figure 6 shows the response surface. A monotonic positive relationship exists between the ice mass anomaly as a function of the snow PDD factor across most of the ice PDD factor range. However, the relationship between the ice mass loss and the ice PDD factor appears to be non-monotonic.

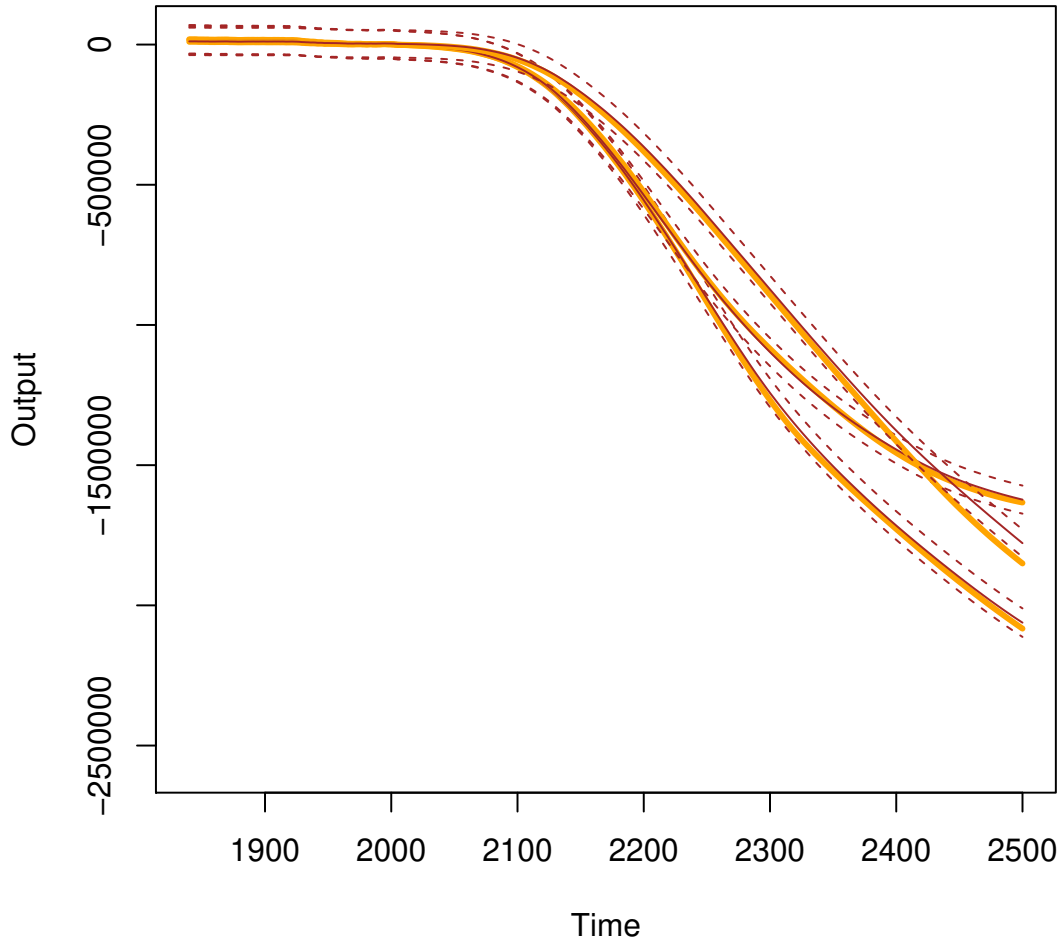

Figure 5: Cross-validation of Greenland Ice Sheet anomaly with respect to year 2003 [Gt] from the SICOPOLIS ensemble. Thick beige lines: actual model output. Brown lines: emulator predictions and confidence intervals associated with standard errors.

Concluding remarks

Here, we present **stilt** - a package for simplified Gaussian process emulation. The package is designed for emulation of time series model output in parameter space, although it can be applied for kriging more generally. The focus is on simplicity, so the package could be easily applied to many challenging problems by users outside of the statistics research community. The streamlined emulator fitting and prediction means the package is also useful pedagogically in demonstrating the Gaussian process emulation. We assume separability in space and time for the Gaussian process. This allows us to reduce computational burden using matrix algebra, and makes it possible to apply the package to moderately large datasets, especially in the time dimension. We showcase package capabilities on three examples, which differ in the amount of parameters in the model ensemble. Specifically, we use a 1D toy dataset, a 2D dataset of Korean summer mean maximum temperatures, and a 5D SICOPOLIS ice sheet model output. Using cross-validation functions, we show the capability of the emulator to predict model response at excluded parameter values in up to five dimensions. We demonstrate the capacity to visualize estimated 2D model response surfaces. The package can be useful when computational resources are limited, and a relatively fast statistical approximator is required for a complex model across a range of parameter space. Some limitations of the package are homoskedasticity, separability, and the fixed covariance structure (e.g., AR(1) and the squared exponential function). We choose the

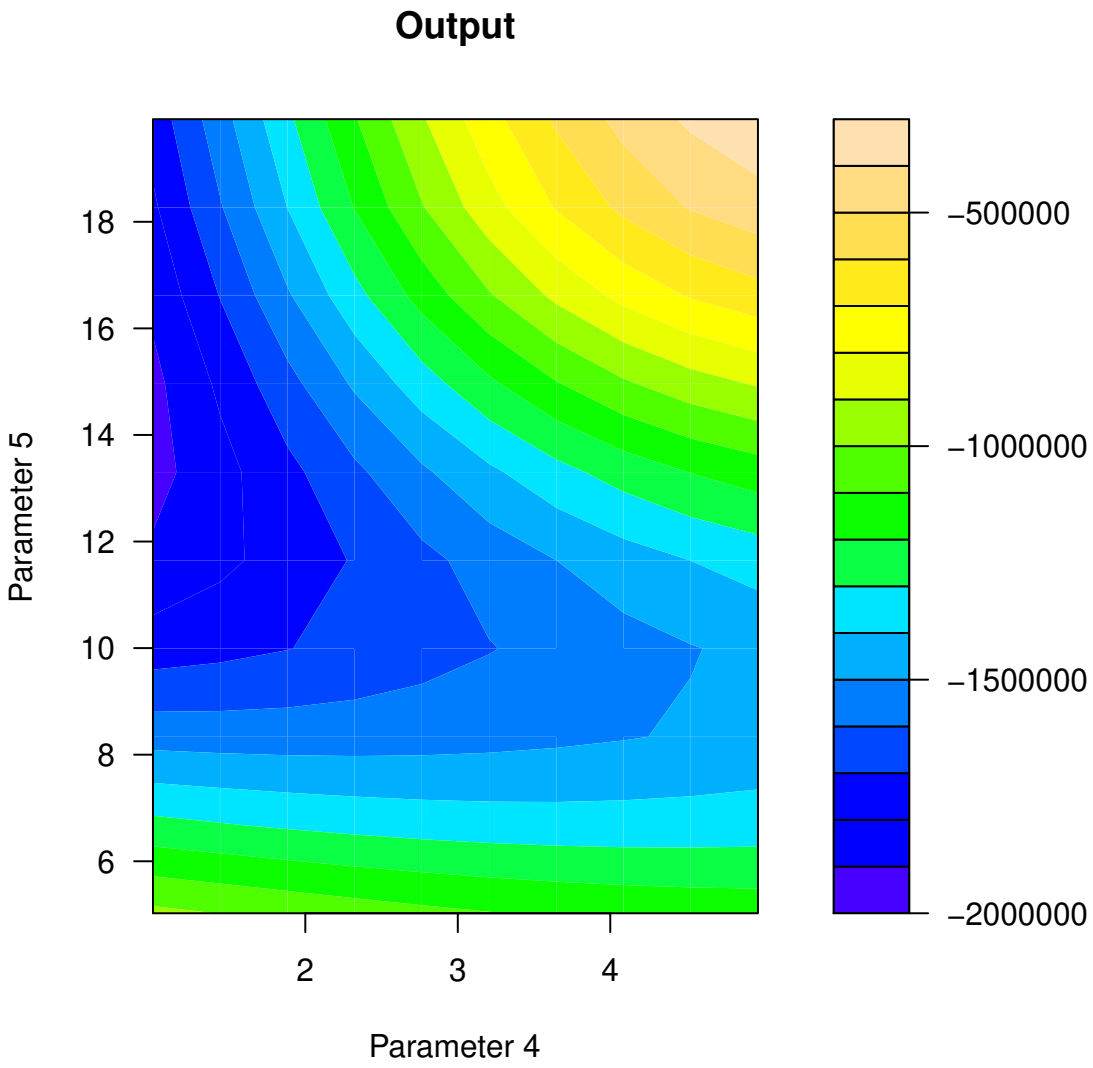

Figure 6: Response surface of SICOPOLIS Greenland Ice Sheet anomaly with respect to year 2003 [Gt] for year 2439 as a function of parameter 4 (snow PDD factor, mm day^{-1} K^{-1}) and parameter 5 (ice PDD factor, mm day^{-1} K^{-1})

AR(1) model because it is applicable to data in diverse scientific fields (Hasselmann, 1976; Keller and McInerney, 2008; **?**; Olson et al., 2013). The disadvantage is that such an emulator may not handle seasonal or periodic effects that may be present in computer model output. Extending the package to account for heteroskedasticity (e.g., **hetGP** package, Binois and Gramacy, 2017) should be considered in future work.

Acknowledgments

For their roles in producing, coordinating, and making available the CMIP5 model output, we acknowledge the climate modeling groups, the World Climate Research Programme's (WCRP) Working Group on Coupled modeling (WGCM), and the Global Organization for Earth System Science Portals (GO-ESSP). We thank Jong-Soo Shin for help with extracting Korean temperature output, and Patrick Applegate for sharing the SICOPOLIS ice sheet model output. We acknowledge financial support from National Research Foundation of Korea (NRF-2009-0093069, NRF-2018R1A5A1024958) , and from the Institute for Basic Science (project code IBS-R028-D1). This work was also co-supported by the National Science Foundation through the Network for Sustainable Climate Risk Management (SCRiM) under NSF cooperative agreement GEO-1240507 and the Penn State Center for Climate Risk

Management. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, or any other foundation or entity.

Appendix

This Appendix describes the technique to reduce computational cost when evaluating the likelihood (Equation 8). In its original form, it involves a computationally expensive inverse of an $np \times np$ matrix. Consider a $p \times n$ matrix $\mathbf{C}$, where $\mathbf{Y} - \boldsymbol{\mu}_{\boldsymbol{\beta}} = \text{vec}(\mathbf{C})$, and the vec operation stacks columns of a matrix into a column vector, from left to right (Rougier, 2008). Using properties of Kronecker products (Golub and Van Loan, 1996) of the vec operator (Magnus and Neudecker, 2007) and other matrix algebra:

$$\begin{aligned}(\mathbf{Y} - \boldsymbol{\mu}_{\boldsymbol{\beta}})^T \Sigma^{-1} (\mathbf{Y} - \boldsymbol{\mu}_{\boldsymbol{\beta}}) &= (\text{vec}(\mathbf{C}))^T [\Sigma_t^{-1} \otimes \Sigma_\theta^{-1}] \text{vec}(\mathbf{C}) \\ &= (\text{vec}(\mathbf{C}))^T [(\Sigma_t^{-1})^T \otimes \Sigma_\theta^{-1}] \text{vec}(\mathbf{C}) \\ &= (\text{vec}(\mathbf{C}))^T \text{vec}(\Sigma_\theta^{-1} \mathbf{C} \Sigma_t^{-1}) \\ &= \text{sum}\left[\mathbf{C} * (\Sigma_\theta^{-1} \mathbf{C} \Sigma_t^{-1})\right],\end{aligned} \tag{15}$$

where $*$ is the Hadamard product. This reduces the dimension of matrices that are inverted to $p \times p$ and $n \times n$, thus substantially reducing computational burden. We further note that both matrices Σ_θ and Σ_t are positive definite (Wicklin, 2013, and others). Thus, the inverses can be found through the Cholesky decomposition of $\Sigma_\theta = \mathbf{R}_\theta^T \mathbf{R}_\theta$, where $\mathbf{R}_\theta$ (called Cholesky factor) is an upper triangular matrix; and similarly for Σ_t.

Additionally, determinant computations can be simplified considerably:

$$|\Sigma| = |\Sigma_t \otimes \Sigma_\theta| = |\Sigma_t|^p |\Sigma_\theta|^n. \tag{16}$$

Thus, we only need to evaluate the determinants of the individual covariance matrices. Moreover, since Σ_θ and Σ_t are positive definite, $|\Sigma_\theta| = \prod_{i=1}^{p} r_{\theta,ii}^2$, where $r_{\theta,ii}$ are diagonal elements of the Cholesky factor $\mathbf{R}_\theta$. Similar calculations can be performed for $|\Sigma_t|$.

Roman Olson
Department of Atmospheric Sciences
Yonsei University
529A Sciences Building, 50 Yonsei-ro
Sinchon-dong, Seodaemun-gu, Seoul, 03722, South Korea
And:
Center for Climate Physics
Institute for Basic Science
Room 1111, Tonghapgigyegwan Building
Busandaehak-ro 63 beon-gil 2
Geumjeong-gu, Busan, 46241
South Korea
And:
Pusan National University, Busan, South Korea, 46241
romanolson@pusan.ac.kr

Kelsey L. Ruckert
Earth and Environmental Systems Institute
320c EES Building
University Park, PA, 16802, United States of America
klr324@psu.edu

Won Chang
Department of Mathematical Sciences
University of Cincinatti
5516 French Hall, 2600 Clifton Ave
Cincinatti, OH, 45220, United States of America
won.chang@uc.edu

Klaus Keller
Department of Geosciences
Penn State University
436 Deike Building
University Park, PA, 16802, United States of America
kzk10@psu.edu

Murali Haran
Department of Statistics
Penn State University
326 Thomas Building
University Park, PA, 16802, United States of America
muh10@psu.edu

Soon-Il An
Department of Atmospheric Sciences
Yonsei University
538 Sciences Building, 50 Yonsei-ro
Sinchon-dong, Seodaemun-gu, Seoul, 03722, South Korea
sian@yonsei.ac.kr

Bibliography

P. J. Applegate, N. Kirchner, E. J. Stone, K. Keller, and R. Greve. An assessment of key model parametric uncertainties in projections of greenland ice sheet behavior. *Cryosphere*, 6(3):589–606, 2012. [p9]

K. S. Bakar and S. K. Sahu. spTimer: Spatio-temporal bayesian modeling using R. *Journal of Statistical Software*, 63(15):1–32, 2015a. URL http://www.jstatsoft.org/v63/i15. [p1]

K. S. Bakar and S. K. Sahu. *spTimer: Spatio-Temporal Bayesian Modeling Using R*, 2015b. R package version 2.0-1. [p1]

K. S. Bhat, M. Haran, R. Olson, and K. Keller. Inferring likelihoods and climate system characteristics from climate models and multiple tracers. *Environmetrics*, 23(4):345–362, 2012. [p1, 2]

M. Binois and R. B. Gramacy. *hetGP: Heteroskedastic Gaussian Process Modeling and Design under Replication*, 2017. URL https://CRAN.R-project.org/package=hetGP. R package version 1.0.1. [p13]

W. Chang, P. J. Applegate, M. Haran, and K. Keller. Probabilistic Calibration of a Greenland Ice Sheet Model Using Spatially Resolved Synthetic Observations: Toward Projections of Ice Mass Loss with Uncertainties. *Geoscientific Model Development*, 7(5):1933–1943, 2014a. ISSN 1991-959X. URL https://doi.org/10.5194/gmd-7-1933-2014. [p1]

W. Chang, M. Haran, R. Olson, and K. Keller. Fast Dimension-Reduced Climate Model Calibration and the Effect of Data Aggregation. *Annals of Applied Statistics*, 8(2):649–673, 2014b. ISSN 1932-6157. URL https://doi.org/10.1214/14-aoas733. [p1, 3]

N. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, New York, U.S.A, 1993. [p2]

N. Cressie and G. Johannesson. Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 70(1):209–226, 2008. [p2]

G. M. Dancik. *Mlegp: Maximum Likelihood Estimates of Gaussian Processes*, 2013. URL https://CRAN.R-project.org/package=mlegp. R package version 3.1.4. [p1, 3]

G. M. Dancik and K. S. Dorman. **mlegp**: Statistical analysis for computer models of biological systems using R. *Bioinformatics*, 24(17):1966–1967, 2008. [p1, 3]

D. Drignei, C. E. Forest, and D. Nychka. Parameter estimation for computationally intensive nonlinear regression with an application to climate modeling. *Annals of Applied Statistics*, 2(4):1217–1230, 2008. [p1]

A. O. Finley, S. Banerjee, and A. E.Gelfand. spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, 63(13):1–28, 2015. URL http://www.jstatsoft.org/v63/i13/. [p1]

D. M. Gay. Usage Summary for Select Optimization Routines. Computing Science Technical Report 153. Technical report, AT&T Bell Laboratories, Murray Hill, New Jersey, U. S. A., 1990. [p4]

J. E. Gentle. *Matrix Algebra. Theory, Computations, and Applications in Statitics*. Springer-Verlag, New York, U.S.A, 2007. [p4]

G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 3 edition, 1996. [p4, 14]

R. Greve. Application of a polythermal three-dimensional ice sheet model to the Greenland ice sheet: Response to steady-state and transient climate scenarios. *Journal of Climate*, 10(5):901–918, 1997. URL https://doi.org/10.1175/1520-0442(1997)010<0901:aoaptd>2.0.co;2. [p9]

R. Greve, F. Saito, and A. Abe-Ouchi. Initial results of the SeaRISE numerical experiments with the models SICOPOLIS and IcIES for the Greenland ice sheet. *Annals of Glaciology*, 52(58):23–30, 2011. [p9]

B. Gräler, E. Pebesma, and G. Heuvelink. Spatio-temporal interpolation using gstat. *The R Journal*, 8:204–218, 2016. URL https://journal.r-project.org/archive/2016-1/na-pebesma-heuvelink.pdf. [p1]

J. Hansen, R. Ruedy, M. Sato, and K. Lo. Global surface temperature change. *Reviews of Geophysics*, 48 (4), 2010. [p2]

K. Hasselmann. Stochastic climate models part I. Theory. *Tellus*, 28(6):473–485, 1976. ISSN 2153-3490. URL https://doi.org/10.1111/j.2153-3490.1976.tb00696.x. [p3, 13]

D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008. [p1, 3]

S. Hirahara, M. Ishii, and Y. Fukuda. Centennial-scale sea surface temperature analysis and its uncertainty. *Journal of Climate*, 27(1):57–75, 2014. [p2]

P. B. Holden, N. R. Edwards, K. I. C. Oliver, T. M. Lenton, and R. D. Wilkinson. A probabilistic calibration of climate sensitivity and terrestrial carbon change in GENIE-1. *Climate Dynamics*, 35(5): 785–806, 2010. [p1]

D. A. Jones, W. Wang, and R. Fawcett. High-quality spatial climate data-sets for Australia. *Australian Meteorological and Oceanographic Journal*, 58(4):233–248, 2009. [p2]

K. Keller and D. McInerney. The dynamics of learning about a climate threshold. *Climate Dynamics*, 30 (2-3):321–332, 2008. [p3, 13]

M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B - Statistical Methodology*, 63(3):425–450, 2001. ISSN 1369-7412. [p1]

J. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, 3rd edition, 2007. [p4, 14]

R. H. Moss, J. A. Edmonds, K. A. Hibbard, M. R. Manning, S. K. Rose, D. P. Van Vuuren, T. R. Carter, S. Emori, M. Kainuma, T. Kram, G. A. Meehl, J. F. B. Mitchell, N. Nakicenovic, K. Riahi, S. J. Smith, R. J. Stouffer, A. M. Thomson, J. P. Weyant, and T. J. Wilbanks. The next generation of scenarios for climate change research and assessment. *Nature*, 463(7282):747–756, 2010. [p7]

R. Olson, R. Sriver, M. Goes, N. M. Urban, H. D. Matthews, M. Haran, and K. Keller. A climate sensitivity estimate using Bayesian fusion of instrumental observations and an Earth system model. *Journal of Geophysical Research Atmospheres*, 117(4), 2012. [p1]

R. Olson, R. Sriver, W. Chang, M. Haran, N. M. Urban, and K. Keller. What is the effect of unresolved internal climate variability on climate sensitivity estimates? *Journal of Geophysical Research Atmospheres*, 118(10):4348–4358, 2013. [p1, 3, 13]

R. Olson, W. Chang, K. Keller, and M. Haran. *Stilt: Separable Gaussian Process Interpolation (Emulation)*, 2017. URL `https://CRAN.R-project.org/package=stilt`. R package version 1.2.0. [p2]

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. URL `http://www.gaussianprocess.org/gpml/chapters/RW.pdf`. [p1, 2, 4]

J. Rougier. Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, 17(4):827–843, 2008. URL `https://doi.org/10.1198/106186008x384032`. [p1, 2, 3, 4, 14]

M. Schlather, A. Malinowski, P. J. Menck, M. Oestin, and K. Strokorb. Analysis, simulation and prediction of multivariate random fields with package randomfields. *Journal of Statistical Software*, 63(8):1–25, 2015. [p1]

D. M. H. Sexton, J. M. Murphy, M. Collins, and M. J. Webb. Multivariate probabilistic projections using imperfect climate models part I: Outline of methodology. *Climate Dynamics*, 38(11-12):2513–2542, 2012. [p1]

J. Shin, R. Olson, and S.-I. An. Projected Heat Wave Characteristics over the Korean Peninsula during the Twenty-First Century. *Asia-Pacific Journal of Atmospheric Sciences*, 54(1):53–61, 2018. [p7]

B. J. Smith, J. Yan, and M. K. Cowles. Unified geostatistical modeling for data fusion and spatial heteroskedasticity with r package ramps. *Journal of Statistical Software*, 25(10):1–21, 2008. [p1]

M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, New York, U.S.A, 1999. [p1, 2]

K. E. Taylor, R. J. Stouffer, and G. A. Meehl. An overview of CMIP5 and the experiment design. *Bulletin of the American Meteorological Society*, 93(4):485–498, 2012. [p7]

R. Wicklin. *Simulating Data with SAS*. The SAS Institute, 2013. [p14]