1 The emulator

We treat the output of the simulator y as an uncertain function $f()$ of the simulator inputs x, so that $y = f(x)$. We wish to produce a predictive distribution for y at any model input, conditional on the points already run, or the design (Y, X). Throughout the study, we use a kriging function, similar to a Gaussian process regression emulator, as coded in the package DiceKriging (Roustant et al., 2012) in the statistical programming environment R (R Core Team, 2016), for prediction of climate simulator output at untried inputs. The kriging model or Gaussian Process regression is specified hierarchically with a separate mean and covariance function. For prediction purposes, *a priori* assume that the trend is a simple linear function of the inputs, and adjust with a Gaussian process.

$$f(x) = h(x)^T \beta + Z(x)$$

Where $h(x)^T \beta$ is the mean function, and the residual process Z is a zero mean stationary Gaussian process. The covariance kernel c of Z

$$Cov(Z, Z') = \sigma^2 c(x, x')$$

can be specified in a number of different ways: we use the default diceKriging option of a Matern $v = 5/2$ function so that

$$c(x, x') = (1 + \frac{\sqrt{5}|x - x'|}{\theta} + \frac{5|x - x'|^2}{3\theta^2}) exp(-\frac{\sqrt{5}|x - x'|}{\theta})$$

where θ describes the *characteristic length scales* - a measure of how quickly information about the function is lost moving away from a design point, in any dimension. This and other hyperparameters are estimated via maximum likelihood estimation from the design (Y, X), meaning that the approach is not fully Bayesian (such an approach would find posterior distributions for the hyperparameters rather than point estimates). We use Universal Kriging, with no 'nugget' term, meaning that the uncertainty on model outputs shrinks to zero at the design points.

Full details of the Universal kriging process used can be found in Roustant et al. (2012), section 2.1, details of the kernel can be found in section 2.3, and examples of the trend and hyperparameter estimation in section 3 of the same publication.

2 Further emulator verification

Bias correction using the augmented emulator relies on good predictions uing the emulator in the regions of temperature/precipitation space corresponding to observations of the real world. A concern is that there are few ensemble members near the observed values for the Amazon and Central African forests, and that the emulator is forced to extrapolate to estimate forest fraction at these locations. Gaussian process emulators can sometimes perform poorly in extrapolation. Further, it is a concern that the lack of ensemble members near the observations means it is difficult to est the accuracy of the emulator at these important locations.

In an ideal world, we would generate ensemble members at or near the observations in question, as a way to validate the emulator and ensure our predictions are correct. This is impractical for two reasons 1) we don't have access to the model and setup in order to generate new runs. While it sounds like a weakness of the design, this is a feature of the paper, in that this is a common situation when analysts are working with models from other groups, with older versions of the model, or with very computationally expensive models where more runs cannot be afforded. 2) There is no way to directly control the temperature and precipitation in the model in order to generate a particular design. These inputs to the emulator are in fact outputs of the model, controlled largely by an inaccessible set of parameter perturbations. Given that we cannot validate the emulator at the observations, we suggest that we can at least show that the emulator performs well, even when required to extrapolate into the broader region of temperature and precipitation where the observations in question lie.

In order to test ewe hold out 6 ensemble members in the region of and nearest to the observations of temperature and precipitation of the Amazon and Central Africa. We hold out ensemble members with a precipitation below 0.2 and temperature below 0.4 in the normalised ensemble. These ensemble members occur in the bottom-left of the temperature-precipitation phase

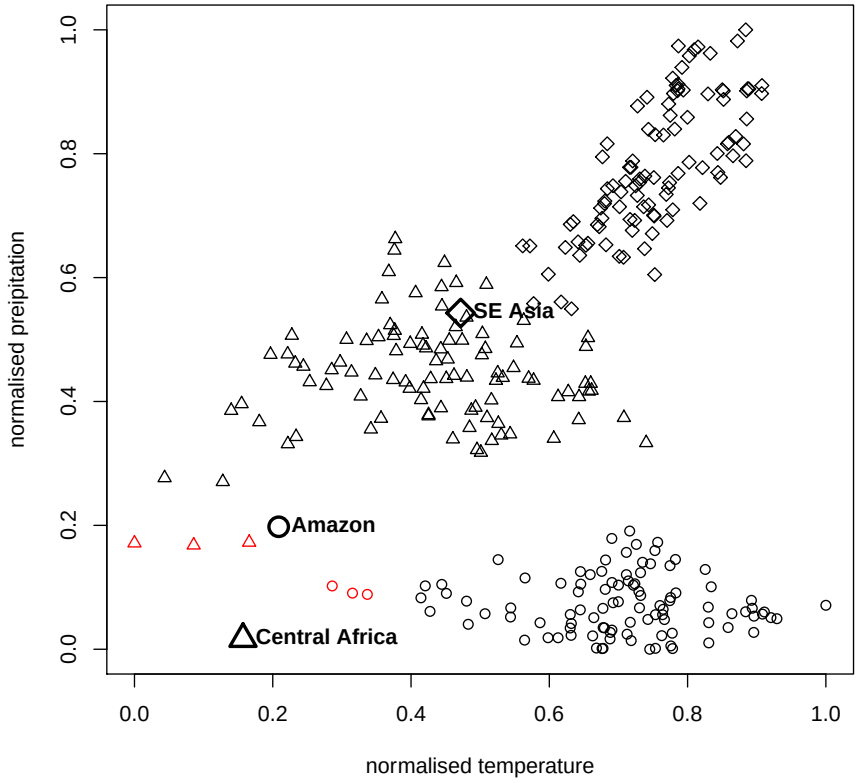

Figure 1. Location of held-out ensemble members in temperature and precipitation space.

space, closest to the Central African and Amazon observations (figure 1). They consist of three members each from the Central African and Amazon forests. These held-out members include one member at the very edge of the temperature space, that is it must be a marginal extrapolation. In our experience, marginal extrapolation is less accurate than extrapolating within the marginal limits of a multidimensional space.

Figure 2 shows the prediction of the 6 held-out ensemble members (red dots) in the context of the leave-one-out validation (black dots). In the held-out case, we fit the emulator based on the 294 remaining ensemble members, and predict all 6 held out members at the same time. As the training set is slightly smaller than each leave-one-out training set (299 members), and the emulator is expected to extrapolate further, we might expect a significant degradation in the performance of the emulator in prediction. As we see in fig. 2, there is little evidence of such a degradation. Both prediction error and estimated uncertainty are well within the bounds of that found during the leave-one-out validation exercise.

Figure 3 shows the prediction error for the 6 members, in the context of the histogram of errors from the leave-one-out exercise. None of the errors are near the limits of the distribution, even though they might be expected to be larger, with a smaller training set and deeper extrapolation.

When making a direct comparison of prediction of the 6 held-out members (fig. 4), we see that there is some small degradation in the performance of the emulator - predictions tend to be slightly further from the held-out ensemble member, and uncertainty bounds wider. However, it should be noted that the error of the held-out samples is 1) only slightly larger than

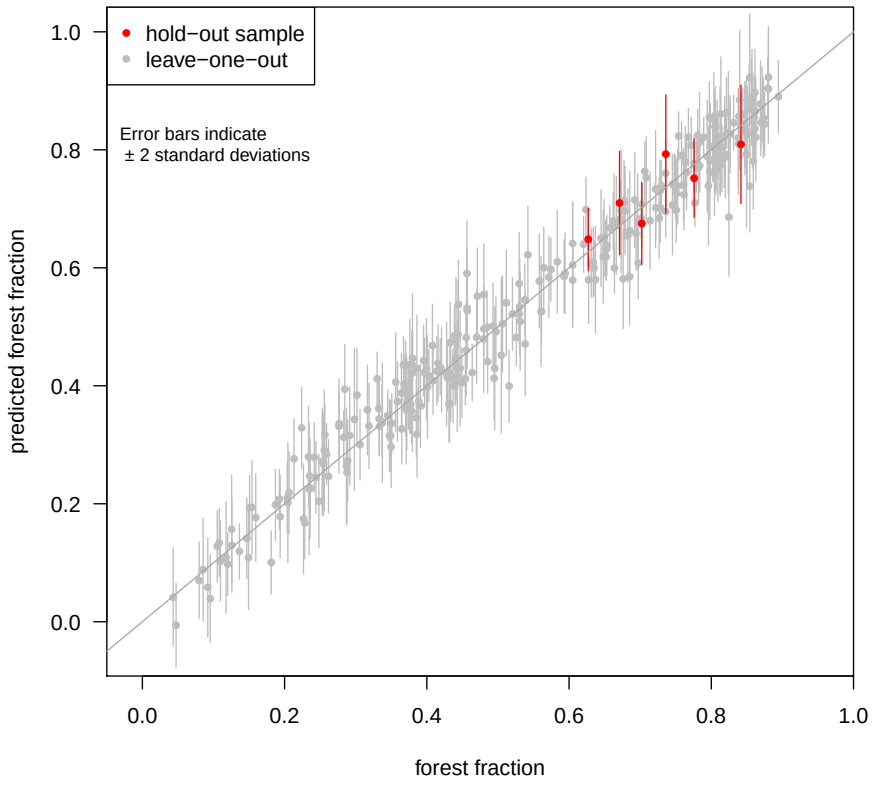

Figure 2. Predictions of forest fraction ensemble members in a leave-one-out validation exercise (grey dots) and for the 6 held-out ensemble members (red dots).

in the leave-one-out case, 2) small when compared to the range of the ensemble, and 3) prediction uncertainty intervals are certainly appropriate and do not increase dramatically. There seems to be no question that even when asked to predict ensemble members that are near the edge of parameter space, and are a significant extrapolation, the emulator performs well. Obviously, this shouldn't be taken as meaning that there is no risk of the emulator performing poorly when extrapolating to the regions of the Amazon and Central African temperature and precipitation. However, we hope we have shown that there is little evidence to suggest that the emulator will perform poorly there.

2.1 The importance of the linear prior form for emulator predictions

We test the importance of the prior form of the emulator may be important in extrapolation to the regions of the observations. In figure 5, we look at the error of prediction for an emulator trained using a constant, or "flat" prior form (our standard emulator is built using a linear model prior). We find that the performance of the emulator is very similar in both situations, suggesting that the prior form is not critical in determining the performance of the emulator in extrapolating at least as far as the observations that we have.

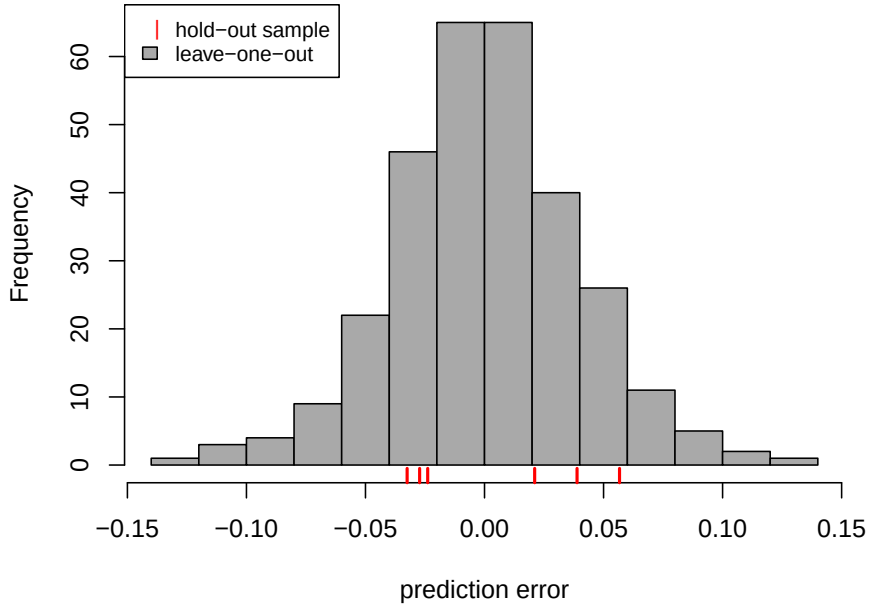

Figure 3. Emulator prediction error in the leave-one out validation exercise (grey histogram), with 6 held-out ensemble members.

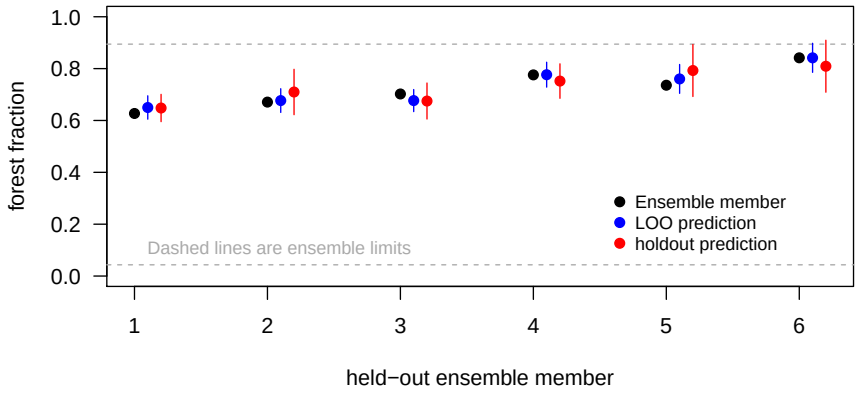

Figure 4. Direct comparison of prediction of the held-out ensemble members in both the leave-one-out (LOO, blue points) and held-out (red points) validation exercises.

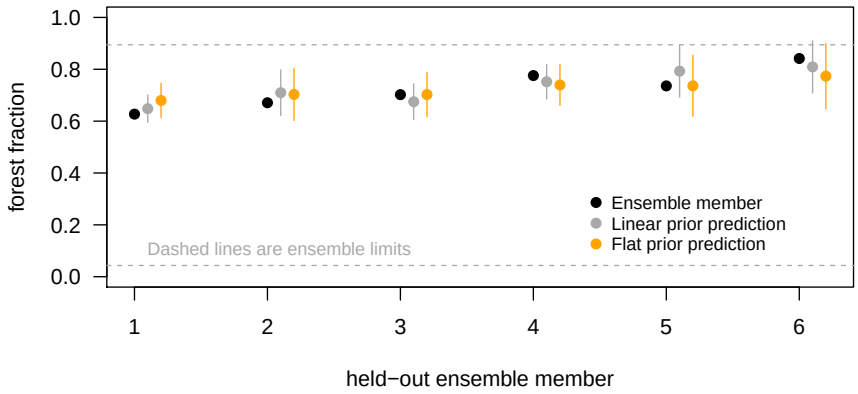

Figure 5. Comparison of emulators for prediction of held-out ensemble members. Black points are the held-out ensemble members, with grey points representing the standard (linear model prior) emulator, and vertical lines $\pm$ 2 standard deviations. Orange points represent prediction with a "constant" or "flat" prior, from which the Gaussian process models deviates.

3 Monte Carlo filtering

We investigate the impact of sample size on the Monte Carlo Filtering (MCF) estimates of parameter uncertainty and their uncertainty. We calculate a sampling uncertainty by calculating the MCF sensitivity metrics 1000 times, each time using a sample size of between 100 and 3000 emulated points from the input space. In this way, we estimate both the mean and the uncertainty (standard deviation) of that mean, when using a different number of ensemble members to calculate the MCF sensitivity indices, including that for 300 members, our ensemble size. We plot these in fig. 6.

We note that the sensitivity indices are estimated higher when a small number of ensemble members are used, as well as with a higher uncertainty. The change in both the estimated statistic and its uncertainty have begun to become small by the time 3000 ensemble members are used, suggesting that we should use at least this many emulated ensemble members to obtain an approximately unbiased sensitivity analysis. We use 5000 emulated members for our analysis in the main paper.

We examine the relationship between the MCF sensitivity measures and the FAST99 sensitivity measures, to see if the latter might overestimate the sensitivity of forest fraction to temperature and precipitation, due to sampling a corner of input space with no tropical forest. We plot only the FAST99 first-order sensitivity, as we do not expect MCF sensitivity to be able to measure interactions between inputs accurately. We find a fairly strong relationship between the two sensitivity measures, although we would expect some differences, as they are measuring different things, and MCF is not sampling from locations in temperature and precipitation space where there are no ensemble members. The FAST99 algorithm produces very similar sensitivity indices (perhaps fortuitously, as they measure on a different scale) for temperature and precipitation as the MCF algorithm for the Amazon forest, but the Southeast Asian and Central African forests appear less sensitive to these inputs when estimated using the MCF algorithm.

4 The FAMOUS climate model

4.1 Model Parameters

We show a list of FAMOUS model land surface input parameters in table 1.

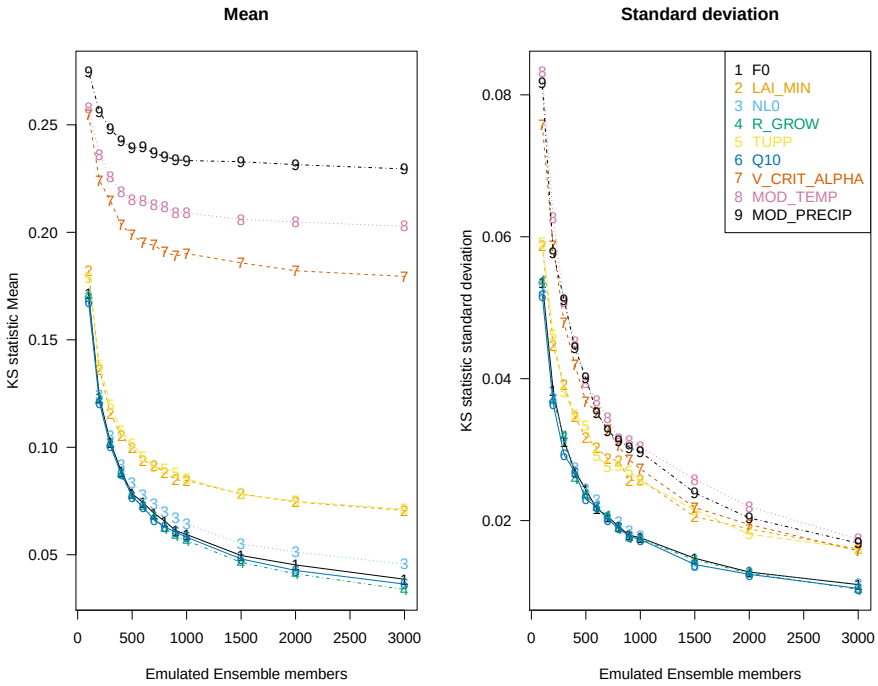

Figure 6. Mean (left) and standard deviation (right) of the KS statistic the Monte Carlo Filtering index of sensitivity, estimated using different sizes of emulated ensembles.

Table 1. Land surface input parameters for FAMOUS

Parameter	Default	Units	Description
F0	0.875		Ratio of CO_2 concentrations inside and outside leaves at zero humidity deficit.
LAI_MIN	3		PFT must achieve this value of LAI before starting to contend with other PFTs for growing area.
NL0	0.03	kgN/kgC	Top leaf nitrogen concentration. The amount of nitrogen per amount of carbon.
R_GROW	0.250		Growth respiration fraction.
TUPP	36	°C	Control on variation of photosynthesis with temperature.
Q10	2		Control on soil respiration with temperature.
V_CRIT_ALPHA	0.5		Control of photosynthesis with soil moisture.

4.2 Forest regions

Forest fraction data is taken by calculating the mean broadleaf forest fraction in the areas shown in figure 8. Mean temperature and precipitation from the model are calculated for the corresponding regions and time period. The regions are: Amazon 15°S - 15°N, 270°E - 315°E; Central Africa; 15°S - 10°N, 7.5°E - 30°E; SE Asia 12°S - 10°N, 90°E - 150°E.

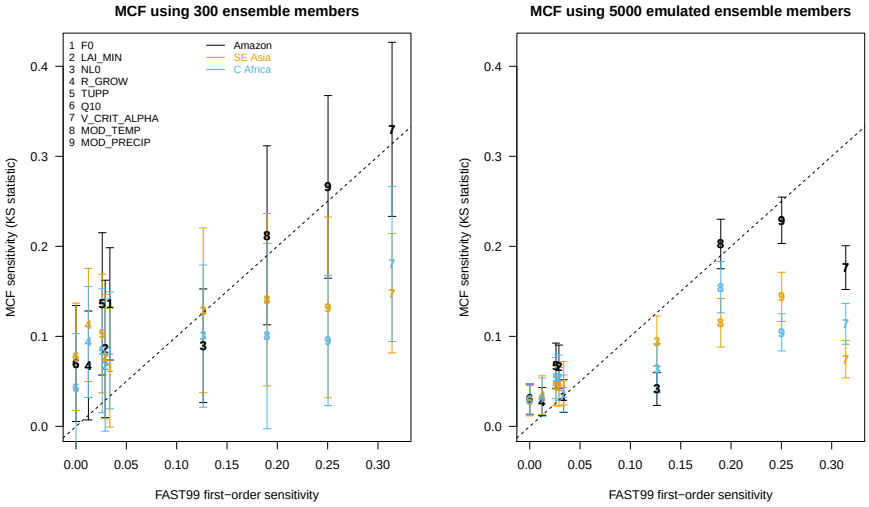

Figure 7. Relationship between the first-order sensitivity of input parameters calculated by the FAST99 algorithm, and that calculated by the Monte Carlo Filtering (MCF) algorithm. The sensitivity indices calculated only using the ensemble members are plotted on the left, with uncertainty estimated by using an emulated 300 member ensemble. On the right, we plot the sensitivity indices and associated uncertainty when calculated using 5000 emulated ensemble members.

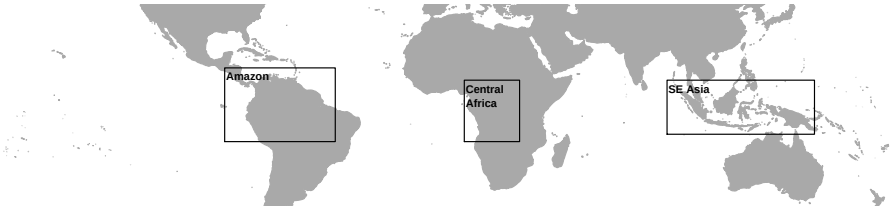

Figure 8. A map of the forest regions used in the study.

References

R Core Team: R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/, 2016.

Roustant, O., Ginsbourger, D., and Deville, Y.: DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization, Journal of Statistical Software, 51, 1–55, http://dx.doi.org/10.18637/jss.v051.i01, 2012.