Sensitivity of Bayesian estimates to noise

Intro

The Bayesian parameter estimates might be sensitive to noise in the observation and forcing data, however, our package can be used to understand these dependencies. To this end, this package contains two functions `noise_sens_single_it` and `noise_sensitivity`, explained below:
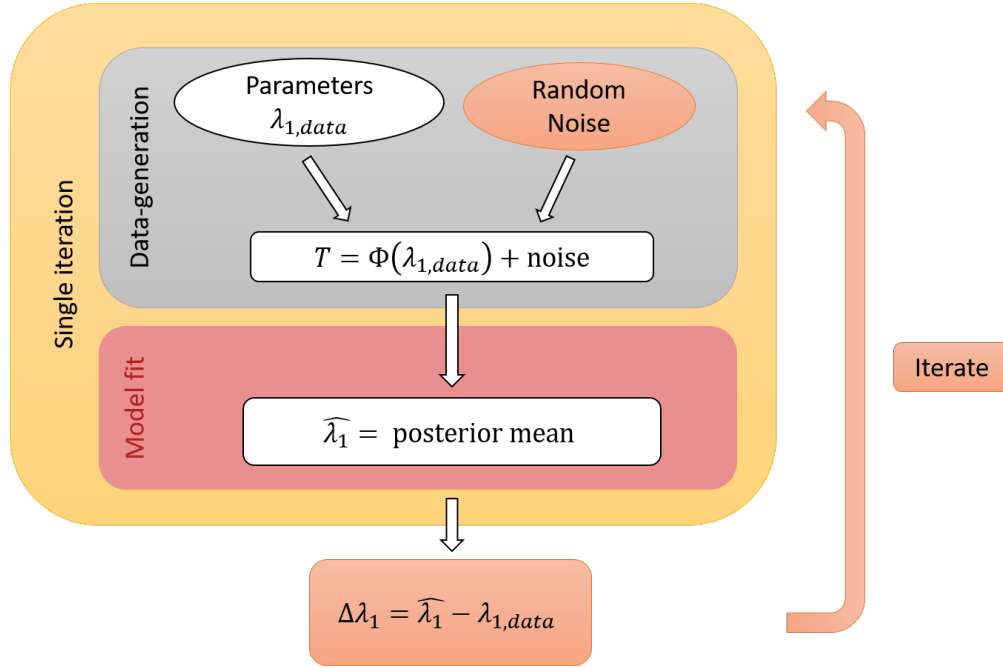

Figure 1: Workflow for noise sensitivity tests

Here, we study the noise senstivity of the estimate of λ_1. As shown in the above figure, synthetic timeseries are first created using the EBM solution with parameter $\lambda_{1,\text{data}}$ and additional random noise (grey box). Second, we fit the EBM to this synthetic data (red box) and consider the posterior mean of our parameter estimate $\hat{\lambda}_1$. The latter might differ from the true $\lambda_{1,\text{data}}$ due to the added noise. These two steps are implemented by `noise_sens_single_it` (see yellow box) and represent one iteration.

Yet, the result will depend on the single realization of the noise. To overcome this potential limitation, it is necessary to repeat the fit with sampled noise. `noise_sensitivity` performs this iteration and stores the estimated deviation $\Delta\lambda_1 = \hat{\lambda}_1 - \lambda_{1,\text{data}}$ after each step (orange boxes).

Example

We use a simple step-function to demonstrate the two functions. To generate the synthetic data, we define a forcing time series and a set of initial and fixed parameters that serve as input to `noise_sens_single_it`. In addition, we specify the standard deviation (SD) of the added noise (white noise and AR(1) process). The function returns the result of the fit to the synthetic data.

```r
forc = c(0, numeric(99) + 10)
vars = list(lambda = 0.1, weights = 1, Cap = 10.1, T0 = 0, F0 = 0)
res_list <- noise_sens_single_it(forc, vars,
        sd_white = 0.5, sd_ar1 = 0.1,
        config_file = system.file('extdata/ebm_fit_config.yml',
                                    package = 'ClimBayes'),
        config = "noise_sens")
print(res_list)
#> Fitted 1-box EBM to data
#> from years 0 to 99
#> Estimated parameters:
#> Parameter lambda1 with mean: 0.099
#> Parameter T0 with mean: 0.704
#> RMSE: 0.466
plot(res_list)
#> $gg_fit
```
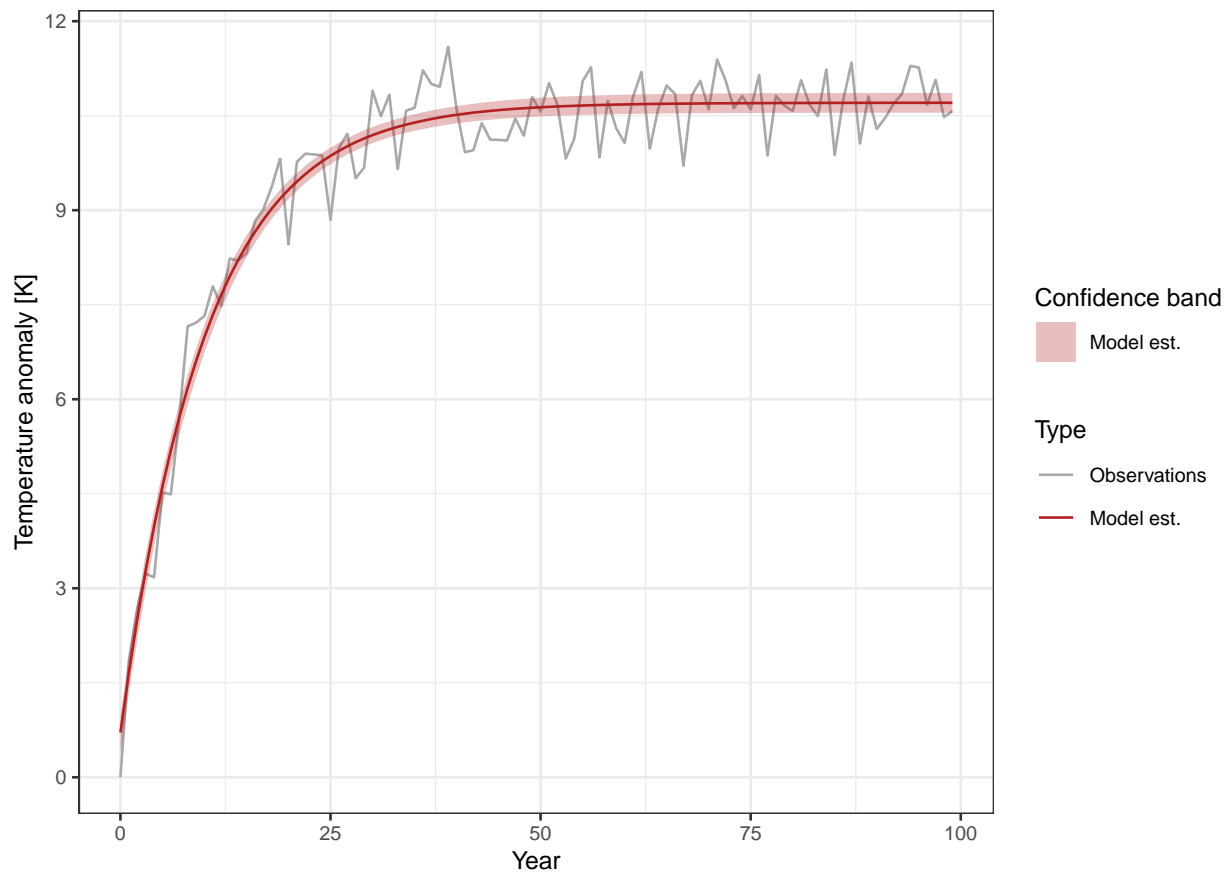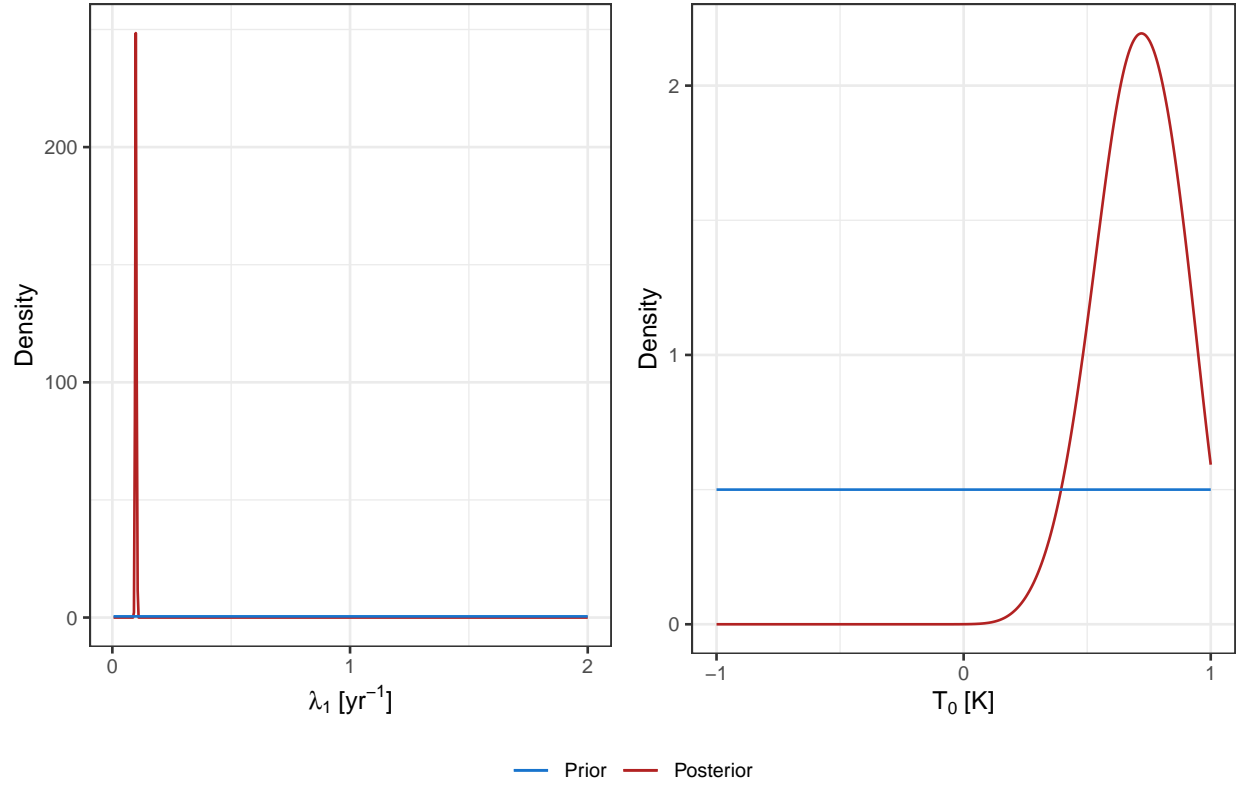


```r
#>
#> $gg_marginals
```

Probability densities

Next, we call `noise_sensitivity` to repeat this fit. This function can not only take a single value for the SDs of the noise, but also a list. In this way, we can test the effect of different noise amplitudes. We specify 10 iterations (`reps = 10`). The output is a data frame with rows corresponding to the choice of the SD and the iteration steps. Here, we choose two different values for `sd_white` and 20 repetitions. Hence the output will have 40 rows. Please note that depending on the computing device, repeating this experiment might take a few minutes. Therefore, we call the pre-saved output `"data/results_noise_sensitivity.rda"` here:

```
results <- noise_sensitivity(sd_white_list = c(0.5, 1),
                             sd_ar1_list = 0,
                             reps = 20,
                             forc = forc,
                             vars = vars,
                             config = "noise_sens")
```

```
print(results)
#> # A tibble: 40 x 4
#>    sd_white sd_ar1   rep delta_lambda1
#>       <dbl>  <dbl> <int>         <dbl>
#> 1       0.5      0     1       0.00130
#> 2       0.5      0     2      -0.00290
#> 3       0.5      0     3       0.00102
#> 4       0.5      0     4       0.000382
#> 5       0.5      0     5      -0.000523
#> 6       0.5      0     6       0.00153
#> 7       0.5      0     7       0.00103
#> 8       0.5      0     8       0.00176
```

```
#> 9      0.5      0     9     0.000632
#> 10     0.5      0     10    -0.000764
#> # ... with 30 more rows
```

Note that, we estimated both λ_1 and T_0 within the Bayesian framework, but are more interested in the effect on λ_1. Thus, `noise_sensitivity` only stores the error $\Delta\lambda_1$, not the result of the estimated T_0, which could be readily changed.

To illustrate the results, consider the following histogram:

```
ggplot2::ggplot(results, ggplot2::aes(x = delta_lambda1,
                                      fill = as.factor(sd_white),
                                      col = as.factor(sd_white))) +
ggplot2::geom_histogram(bins = 10, position = "identity", alpha = 0.5) +
ggplot2::theme_bw() +
ggplot2::labs(fill = "SD of white noise") +
ggplot2::guides(col = "none")
```
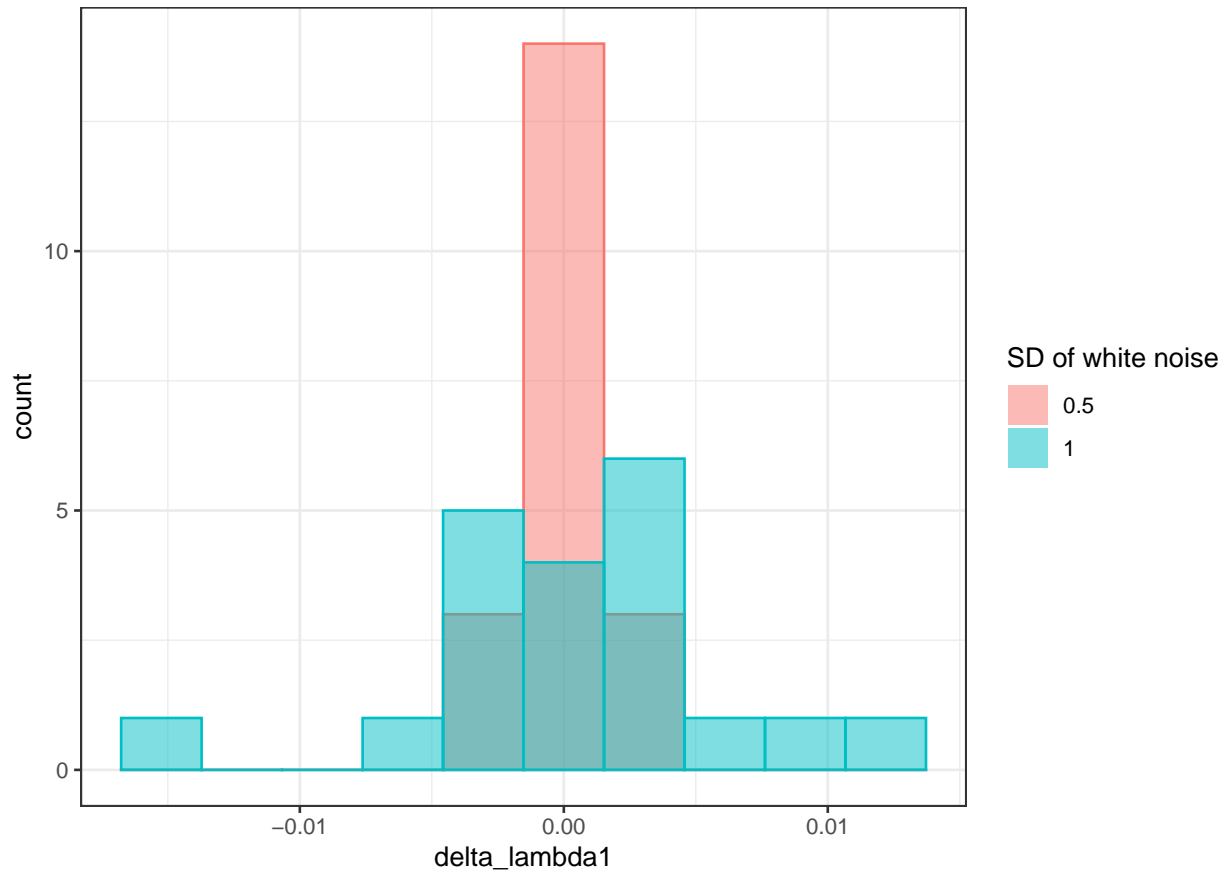

We see that both positive and negative deviations $\Delta\lambda_1$ occur, i.e., the feedback parameter can be under- and overestimated depending on the noise. On average, the error is near zero. For higher noise levels (`sd_white = 0.5`), the deviation increases. This is as expected as increased noise likely influences the Bayesian estimation more strongly.

Results

The above functions can be applied to any forcing and number of boxes. As another example, we use historical forcing from 1850-2000 and the two-box model. We choose the data-generating parameters as parameter estimates from fitting the model to HadCRUT5 observations. Further, we change the values $\Delta\lambda_i$ to relative deviations (as percentage of the true value) and consider the equilibrium climate sensitivity (ECS), too. The following plot shows the results of this experiment:
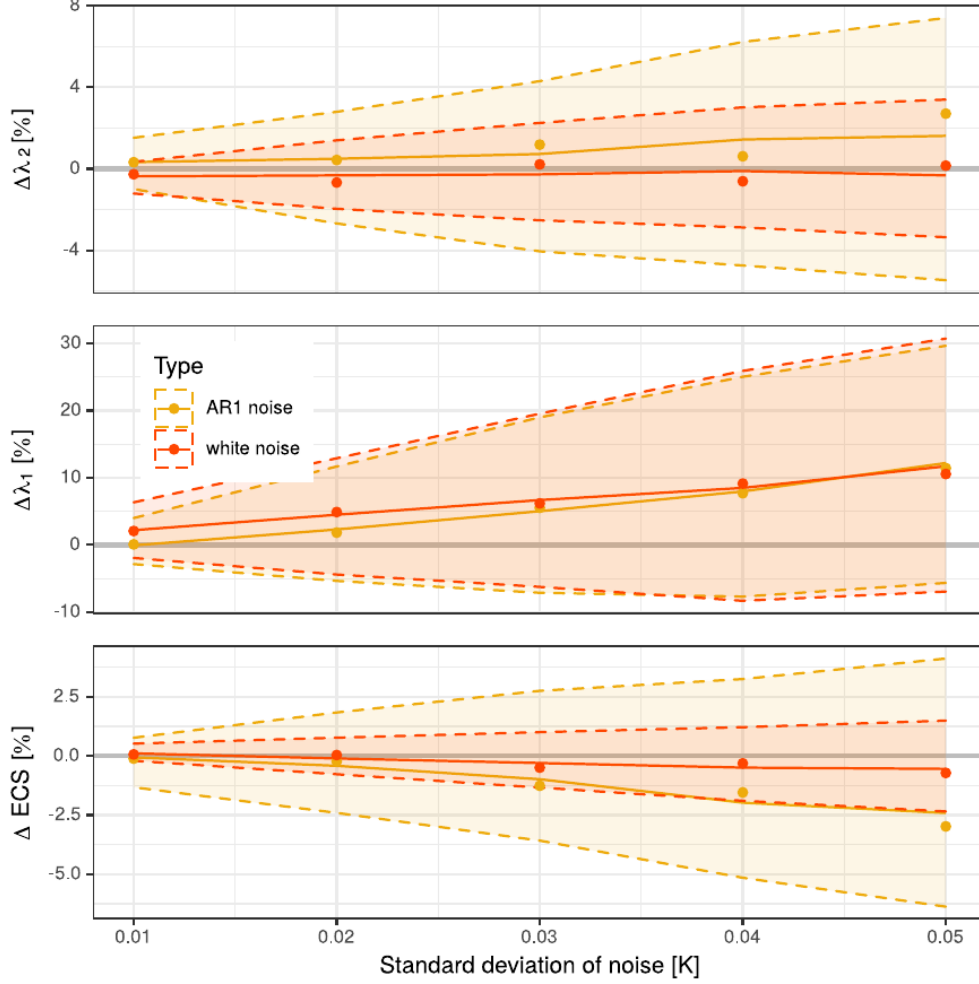

Figure 2: Noise sensitivity for historic forcing and 2-box

The Figure shows the spread of relative errors after 150 iterations of varying standard deviation. The dots and solid line present the median relative errors. Dashed lines show the first and third quartiles. Lines are computed from the `loess`-function in R. Colors indicate the type of the noise added: white or correlated AR(1) noise.

Key findings from this plot are:

- Higher levels of noise (i.e. higher SD on the x-axis) leads to a higher probability of larger deviations (wider spread of the curves).

- Correlated noise (yellow, AR(1) noise) has a larger influence on λ_2 (the slow feedback) than white noise (orange, white noise).

- For the fast feedback parameter λ_1, the type of noise has no detectable influence.

- The ECS behaves similar to λ_2 (which is as expected since it is dominated by slow feedback).