Gaussian processes for uncertainty quantification in computer experiments

Richard Wilkinson

University of Nottingham

Gaussian process summer school, Sheffield 2013

Talk plan

(a) UQ and computer experiments
(b) Gaussian process emulators
(c) 3 examples:
    (i) calibration using emulators
    (ii) simulator discrepancy modelling
    (iii) accelerating ABC using GPs

Computer experiments

Baker 1977 (Science):

'Computerese is the new lingua franca of science'

Rohrlich (1991): Computer simulation is

'a key milestone somewhat comparable to the milestone that started the empirical approach (Galileo) and the deterministic mathematical approach to dynamics (Newton and Laplace)'

Computer experiments

Baker 1977 (Science):

> *'Computerese is the new lingua franca of science'*

Rohrlich (1991): Computer simulation is

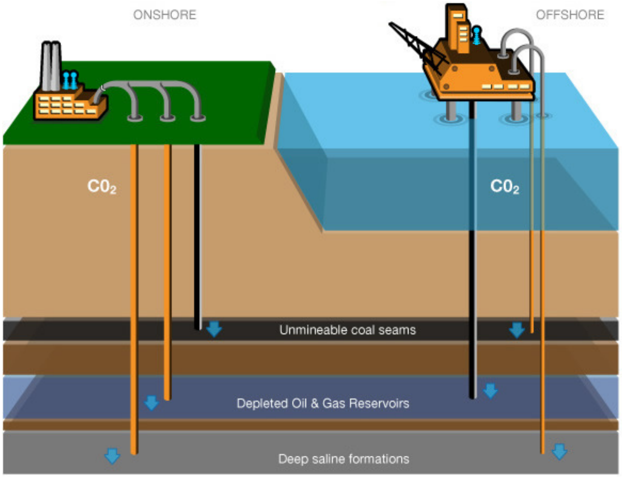
> *'a key milestone somewhat comparable to the milestone that started the empirical approach (Galileo) and the deterministic mathematical approach to dynamics (Newton and Laplace)'*

The gold-standard of empirical research is the designed experiment, which usually involves concepts such as replication, blocking, and randomization.

However, in the past three decades computer experiments (*in silico* experiments) have become commonplace in nearly all fields.

Engineering

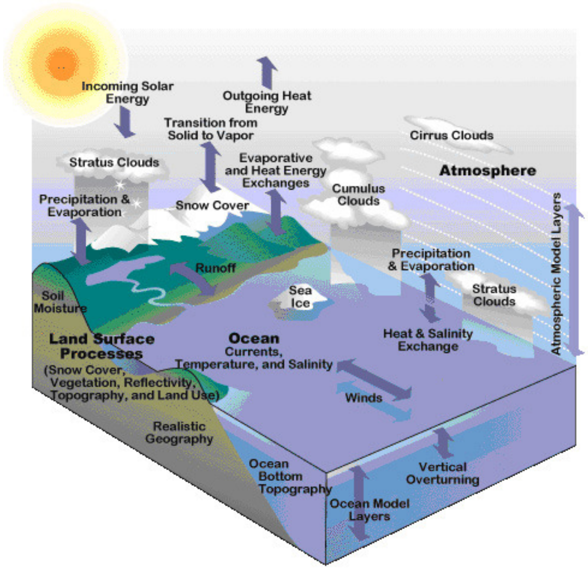
Carbon capture and storage technology - PANACEA project



Knowledge about the geology of the wells is uncertain.

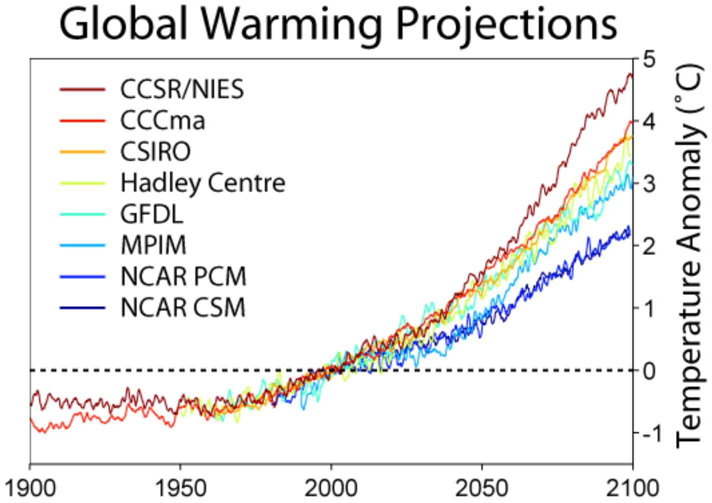
Climate Science

Predicting future climate



Challenges of computer experiments

Climate Predictions



Challenges for statistics

The statistical challenges posed by computer experiments are somewhat different to physical experiments and have only recently begun to be tackled by statisticians.

For example, replication, randomization and blocking are irrelevant because a computer model will give identical answers if run multiple times.

Challenges for statistics

The statistical challenges posed by computer experiments are somewhat different to physical experiments and have only recently begun to be tackled by statisticians.

For example, replication, randomization and blocking are irrelevant because a computer model will give identical answers if run multiple times.

Key questions: How do we make inferences about the world from a simulation of it?

Challenges for statistics

The statistical challenges posed by computer experiments are somewhat different to physical experiments and have only recently begun to be tackled by statisticians.

For example, replication, randomization and blocking are irrelevant because a computer model will give identical answers if run multiple times.

Key questions: How do we make inferences about the world from a simulation of it?

- how do we relate simulators to reality? (model error)
- how do we estimate tunable parameters? (calibration)
- how do we deal with computational constraints? (stat. comp.)
- how do we make uncertainty statements about the world that combine models, data and their corresponding errors? (UQ)

There is an inherent a lack of quantitative information on the uncertainty surrounding a simulation - unlike in physical experiments.

Incorporating and accounting for uncertainty

Perhaps the biggest challenge faced is incorporating uncertainty in computer experiments.

We are used to dealing with uncertainty in physical experiments. But if your computer model is deterministic, there is no natural source of variation and so the experimenter must carefully assess where errors might arise.

Incorporating and accounting for uncertainty

Perhaps the biggest challenge faced is incorporating uncertainty in computer experiments.

We are used to dealing with uncertainty in physical experiments. But if your computer model is deterministic, there is no natural source of variation and so the experimenter must carefully assess where errors might arise.

Types of uncertainty:

- Parametric uncertainty
- Model inadequacy
- Observation errors
- Code uncertainty

Code uncertainty

We think of the simulator as a function

$$\eta : \mathcal{X} \to \mathcal{Y}$$

Typically both the input and output space will be subsets of $\mathbb{R}^n$ for some n.

Monte Carlo (brute force) methods can be used for most tasks if sufficient computational resource is available.

Code uncertainty

We think of the simulator as a function

$$\eta : \mathcal{X} \to \mathcal{Y}$$

Typically both the input and output space will be subsets of $\mathbb{R}^n$ for some n.

Monte Carlo (brute force) methods can be used for most tasks if sufficient computational resource is available.
For example, uncertainty analysis is finding the distribution of $\eta(\theta)$ when $\theta \sim \pi(\cdot)$:

- draw a sample of parameter values from the prior $\theta_1, \ldots, \theta_N \sim \pi(\theta)$,
- Look at $\eta(\theta_1), \ldots, \eta(\theta_N)$ to find the distribution $\pi(\eta(\theta))$.

Code uncertainty

We think of the simulator as a function

$$\eta : \mathcal{X} \rightarrow \mathcal{Y}$$

Typically both the input and output space will be subsets of $\mathbb{R}^n$ for some n.

Monte Carlo (brute force) methods can be used for most tasks if sufficient computational resource is available.
For example, uncertainty analysis is finding the distribution of $\eta(\theta)$ when $\theta \sim \pi(\cdot)$:

- draw a sample of parameter values from the prior $\theta_1, \ldots, \theta_N \sim \pi(\theta)$,
- Look at $\eta(\theta_1), \ldots, \eta(\theta_N)$ to find the distribution $\pi(\eta(\theta))$.

However, for complex simulators, run times might be long.
Consequently, we will only know the simulator output at a finite number of points:

code uncertainty

Code uncertainty

Code uncertainty

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

Code uncertainty

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

- All inference must be done using a finite ensemble of model runs

$$\mathcal{D}_{sim} = \{(\theta_i, \eta(\theta_i))\}_{i=1,\dots,N}$$

Code uncertainty

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

- All inference must be done using a finite ensemble of model runs

$$\mathcal{D}_{sim} = \{(\theta_i, \eta(\theta_i))\}_{i=1,\dots,N}$$

- If θ is not in the ensemble, then we are uncertainty about the value of $\eta(\theta)$.

Code uncertainty

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

- All inference must be done using a finite ensemble of model runs

$$\mathcal{D}_{sim} = \{(\theta_i, \eta(\theta_i))\}_{i=1,\dots,N}$$

- If θ is not in the ensemble, then we are uncertainty about the value of $\eta(\theta)$.

If θ is multidimensional, then even short run times can rule out brute force approaches

- $\theta \in \mathbb{R}^{10}$ then 1000 simulator runs is only enough for one point in each corner of the design space.

Meta-modelling

Idea: If the simulator is expensive, build a cheap model of it and use this in any analysis.

'a model of the model'

We call this meta-model an *emulator* of our simulator.

Meta-modelling

Idea: If the simulator is expensive, build a cheap model of it and use this in any analysis.

'a model of the model'

We call this meta-model an *emulator* of our simulator.

We use the emulator as a cheap approximation to the simulator.

- ideally an emulator should come with an assessment of its accuracy
- rather just predict $\eta(\theta)$ it should predict $\pi(\eta(\theta)|\mathcal{D}_{sim})$ - our uncertainty about the simulator value given the ensemble $\mathcal{D}_{sim}$.

Meta-modelling

Gaussian Process Emulators

Gaussian processes provide a flexible nonparametric distributions for our prior beliefs about the functional form of the simulator:

$$\eta(\cdot) \sim GP(m(\cdot), \sigma^2 c(\cdot, \cdot))$$

where $m(\cdot)$ is the prior mean function, and $c(\cdot, \cdot)$ is the prior covariance function (semi-definite).

Meta-modelling

Gaussian Process Emulators

Gaussian processes provide a flexible nonparametric distributions for our prior beliefs about the functional form of the simulator:

$$\eta(\cdot) \sim GP(m(\cdot), \sigma^2 c(\cdot, \cdot))$$

where $m(\cdot)$ is the prior mean function, and $c(\cdot, \cdot)$ is the prior covariance function (semi-definite).

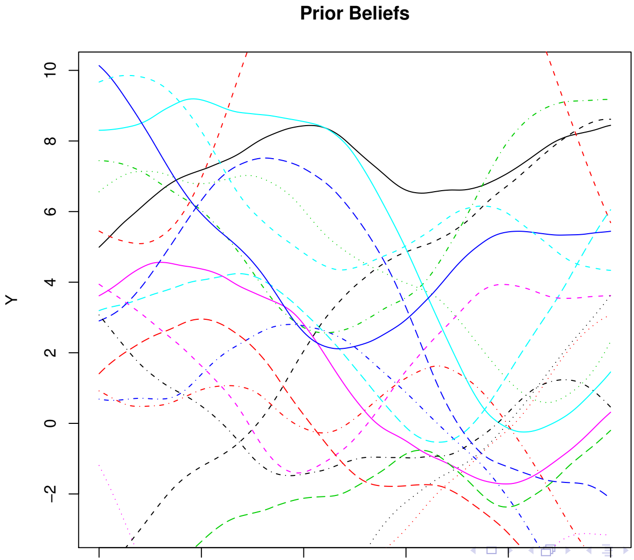
If we observe the ensemble of model runs $\mathcal{D}_{sim}$, then update our prior belief about η in light of the ensemble of model runs:

$$\eta(\cdot)|\mathcal{D}_{sim} \sim GP(m^*(\cdot), \sigma^2 c^*(\cdot, \cdot))$$

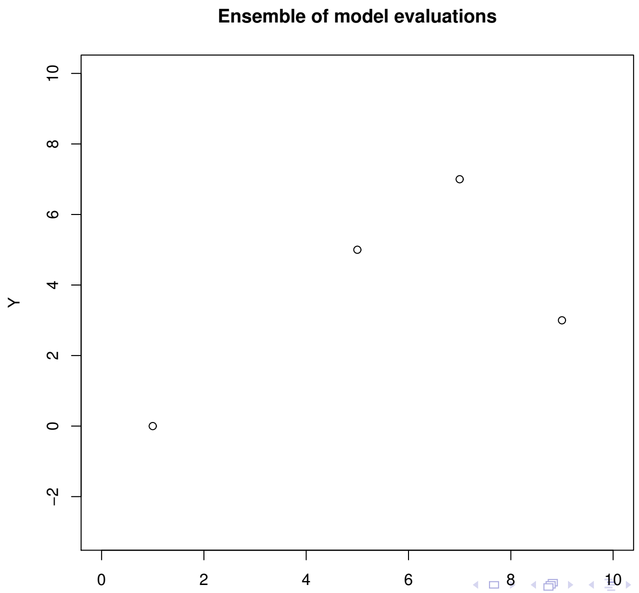
where m^* and c^* are the posterior mean and covariance functions (simple functions of $\mathcal{D}_{sim}$, m and c).

Gaussian Process Illustration

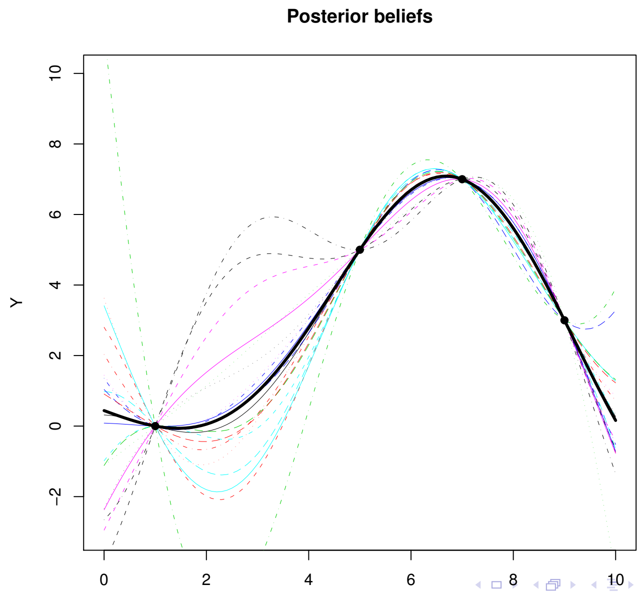
Zero mean



Gaussian Process Illustration



Gaussian Process Illustration



Emulator choices

$$\eta(x) = h(x)\beta + u(x)$$

$$\text{emulator} = \text{mean structure} + \text{residual}$$

Emulator choices

$$\eta(x) = h(x)\beta + u(x)$$
$$\text{emulator} = \text{mean structure} + \text{residual}$$

$u(x)$ can be taken to be a zero-mean Gaussian process

$$u(\cdot) \sim GP(0, c(\cdot,\cdot))$$

Emulator choices

$$\eta(x) = h(x)\beta + u(x)$$

$$\text{emulator} = \text{mean structure} + \text{residual}$$

$u(x)$ can be taken to be a zero-mean Gaussian process

$$u(\cdot) \sim GP(0, c(\cdot, \cdot))$$

Emulator choices:

- mean structure $h(x)$
  - $1, x, x^2, \ldots$, Legendre polynomials?
  - Allows us to build in known trends and exploit power of linear regression

Emulator choices

$$\eta(x) = h(x)\beta + u(x)$$

$$\text{emulator} = \text{mean structure} + \text{residual}$$

$u(x)$ can be taken to be a zero-mean Gaussian process

$$u(\cdot) \sim GP(0, c(\cdot,\cdot))$$

Emulator choices:

- mean structure $h(x)$
  - $1, x, x^2, \ldots$, Legendre polynomials?
  - Allows us to build in known trends and exploit power of linear regression
- covariance function $c(\cdot,\cdot)$ - cf Nicolas' talk
  - Stationary? Smooth?
  - Length-scale?
  - Nb - we don't a nugget term

Multivariate Emulation

Higdon *et al.* 2008

How can we deal with multivariate ouput?

- Build independent or separable multivariate emulators,
- Outer product emulators,
- Linear model of coregionalization?

Multivariate Emulation

Higdon *et al.* 2008

How can we deal with multivariate ouput?

- Build independent or separable multivariate emulators,
- Outer product emulators,
- Linear model of coregionalization?

Instead, if the outputs are highly correlated we can reduce the dimension of the data by projecting the data onto some lower dimensional manifold $\mathcal{Y}^{pc}$.

Multivariate Emulation

Higdon *et al.* 2008

How can we deal with multivariate ouput?

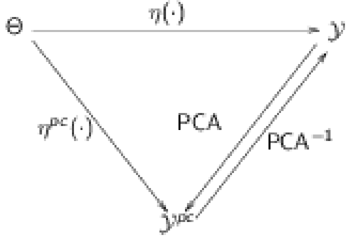
- Build independent or separable multivariate emulators,
- Outer product emulators,
- Linear model of coregionalization?

Instead, if the outputs are highly correlated we can reduce the dimension of the data by projecting the data onto some lower dimensional manifold $\mathcal{Y}^{pc}$.

We can use any dimension reduction technique as long as

- we can reconstruct to the original output space
- we can quantify the reconstruction error.

We can then emulate the function that maps the input space Θ to the reduced dimensional output space $\mathcal{Y}^{pc}$, i.e., $\eta_{pc}(\cdot) : \Theta \rightarrow \mathcal{Y}^{pc}$



It doesn't matter what dimension reduction scheme we use, as long as we can reconstruct from $\mathcal{Y}^{pc}$ and quantify the error in the reconstruction.

Comments

- This approach (PCA emulation) requires that the outputs are highly correlated.
- We are assuming that the output $\mathcal{D}_{sim}$ is really a linear combination of a smaller number of variables,

$$\eta(\theta) = \mathbf{v}_1 \eta_{pc}^1(\theta) + \ldots + \mathbf{v}_{n^*} \eta_{pc}^{n^*}(\theta)$$

  which may be a reasonable assumption in many situations, eg, temporal spatial fields.
- Although PCA is a linear method, the method can be used on highly non-linear models as we are still using non-linear Gaussian processes to map from Θ to $\mathcal{Y}^{pc}$ – the linear assumption applied only to the dimension reduction.
- This method accounts for code uncertainty and automatically accounts for the reconstruction error caused by reducing the dimension of the data.

Example 1: Calibration

Calibration
Inverse problems

For forwards models we specify parameters θ and i.c.s and the model generates output $\mathcal{D}_{sim}$. Usually, we are interested in the inverse-problem, i.e., observe data $\mathcal{D}_{field}$, want to estimate parameter values.

Calibration
Inverse problems

For forwards models we specify parameters θ and i.c.s and the model generates output $\mathcal{D}_{sim}$. Usually, we are interested in the inverse-problem, i.e., observe data $\mathcal{D}_{field}$, want to estimate parameter values.

We have three sources of information that we wish to combine

1. Scientific knowledge captured by the model, η
2. Empirical information contained in the data, $\mathcal{D}_{field}$
3. Expert opinion based upon experience.

Calibration
Inverse problems

For forwards models we specify parameters θ and i.c.s and the model generates output $\mathcal{D}_{sim}$. Usually, we are interested in the inverse-problem, i.e., observe data $\mathcal{D}_{field}$, want to estimate parameter values.

We have three sources of information that we wish to combine

1. Scientific knowledge captured by the model, η
2. Empirical information contained in the data, $\mathcal{D}_{field}$
3. Expert opinion based upon experience.

We want to combine all three sources to produce the 'best' parameter estimates we can.

Carbon feedbacks

- Terrestrial ecosystems currently absorb a considerable fraction of anthropogenic carbon emissions.
- However, the fate of this sink is highly uncertain due to insufficient knowledge about key feedbacks.
  - We are uncertain about the sensitivity of soil respiration to increasing global temperature.
  - GCM predictions don't agree on the sign of the net terrestrial carbon flux.

Carbon feedbacks

- Terrestrial ecosystems currently absorb a considerable fraction of anthropogenic carbon emissions.
- However, the fate of this sink is highly uncertain due to insufficient knowledge about key feedbacks.
  - We are uncertain about the sensitivity of soil respiration to increasing global temperature.
  - GCM predictions don't agree on the sign of the net terrestrial carbon flux.

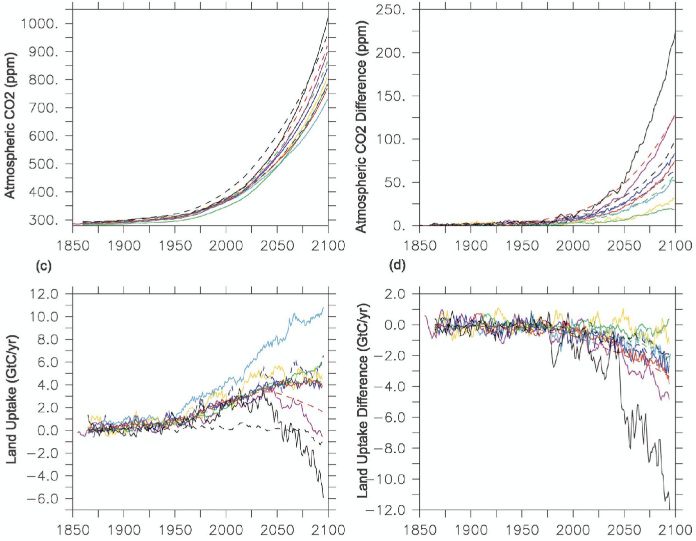
The figure shows inter-model spread in uncalibrated GCM model predictions.

- How much additional spread is there from parametric uncertainty? (as opposed to model structural uncertainty?)
- Would calibration reduce the range of the ensemble predictions? Or would it increase our uncertainty?

We can't answer these questions with full GCMs at present, but we can begin to investigate with simplified EMIC models.

Friedlingstein *et al.* 2006 - 'uncalibrated' GCM carbon cycle predictions

Climate simulators tend to be 'tuned' rather than calibrated, due to their complexity.



UVic Earth System Climate Model

Ricciutio *et al.* 2011, Wilkinson 2010

UVic ESCM is an intermediate complexity model with a general circulation ocean and dynamic/thermodynamic sea-ice components coupled to a simple energy/moisture balance atmosphere. It has a dynamic vegetation and terrestrial carbon cycle model (TRIFFID) as well as an inorganic carbon cycle.

- Inputs: $Q_{10} =$ soil respiration sensitivity to temperature (carbon source) and $K_c = CO_2$ fertilization of photosynthesis (carbon sink).
- Output: time-series of $\mathrm{CO_2}$ values, cumulative carbon flux measurements, spatial-temporal field of soil carbon measurements.

UVic Earth System Climate Model

Ricciutio *et al.* 2011, Wilkinson 2010

UVic ESCM is an intermediate complexity model with a general circulation ocean and dynamic/thermodynamic sea-ice components coupled to a simple energy/moisture balance atmosphere. It has a dynamic vegetation and terrestrial carbon cycle model (TRIFFID) as well as an inorganic carbon cycle.

- Inputs: Q_{10} = soil respiration sensitivity to temperature (carbon source) and $K_c = CO_2$ fertilization of photosynthesis (carbon sink).
- Output: time-series of CO_2 values, cumulative carbon flux measurements, spatial-temporal field of soil carbon measurements.

The observational data are limited, and consist of 60 measurements $\mathcal{D}_{field}$:

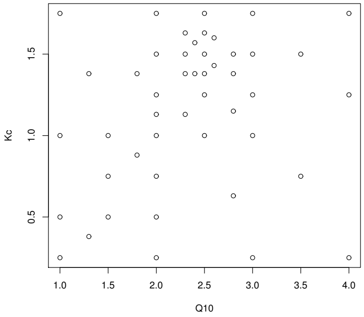
- 40 instrumental CO_2 measurements from 1960-1999 (the Mauna Loa data)
- 17 ice core CO_2 measurements
- 3 cumulative ocean carbon flux measurements

Calibration

The aim is to combine the physics coded into UVic with the empirical observations to learn about the carbon feedbacks.

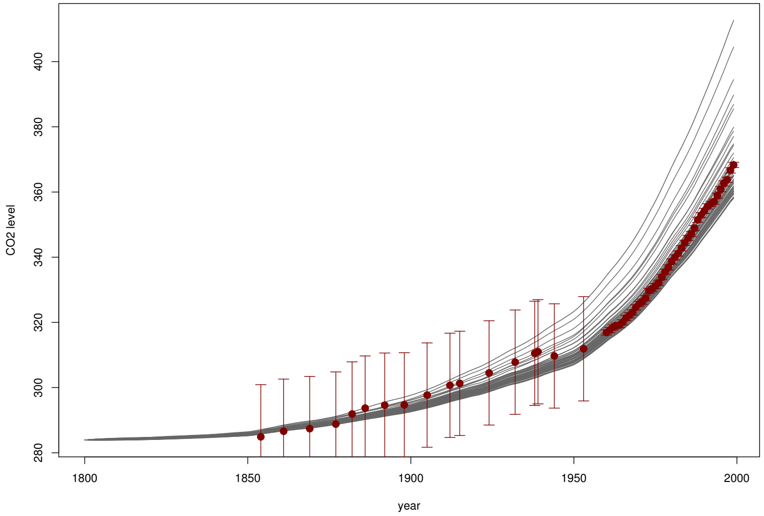
However, UVic takes approximately two weeks to run for a single input configuration. Consequently, all inference must be done from a limited ensemble of model runs.

- 48 member ensemble, grid design D, output $\mathcal{D}_{sim}$ $(48 \times n)$.

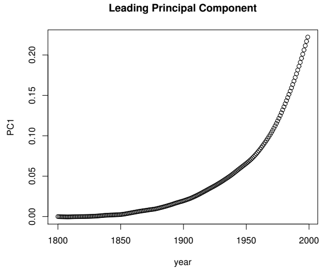


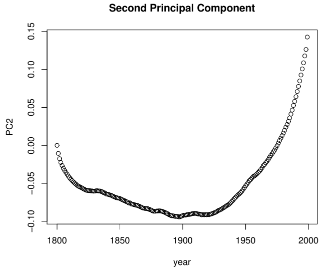
- this is overkill for this model - benefit of sequential designs

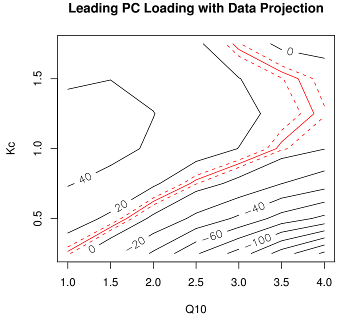
Model runs and data

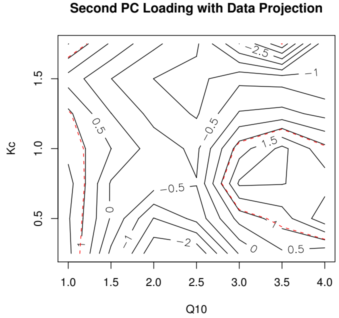


PC Plots


Leading Principal Component
PC1
0.00
0.05
0.10
0.15
0.20
1800
1850
1900
1950
2000
year


Second Principal Component
PC2
−0.10
−0.05
0.00
0.05
0.10
0.15
1800
1850
1900
1950
2000
year


Leading PC Loading with Data Projection
Kc
0.5
1.0
1.5
Q10
1.0
1.5
2.0
2.5
3.0
3.5
4.0
40
20
0
0
−20
−40
−60
−100


Second PC Loading with Data Projection
Kc
0.5
1.0
1.5
Q10
1.0
1.5
2.0
2.5
3.0
3.5
4.0
−2.5
−1
0.5
−0.5
1.5
−0.5
0
1
−1
−2
1
0
0.5

Diagnostics

Diagnostic checks are vital if we are to trust the use of the emulator in place of the simulator.

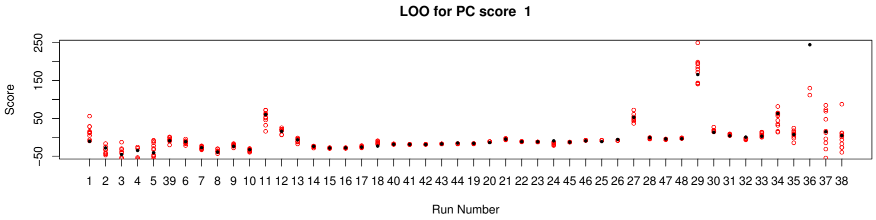
For the PC emulator, we ultimately want to predict the spatial field - so most diagnostic effort should be spent on the reconstructed emulator.

Looking only at the percentage of variance explained by the principal components can be misleading, even if the emulators are perfect, as we can find that PCs that have small eigenvalues (so explain a small amount of variance) can play an important role in prediction.

Leave-one-out (LOA) plots for PC1

Leave-one-out plots are a type of cross-validation to asses whether the final emulator is working well both in terms of the mean prediction, and the uncertainty estimates.

We leave each ensemble member, we leave it out of the training set and build a new emulator. We then predict the left-out ensemble member using the emulator

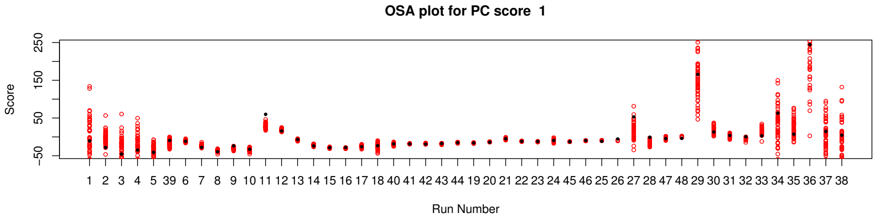


We would like accurate coverage.

One-step-ahead (OSA) plots for PC1

One-step-ahead diagnostics are created by first ordering the ensemble according to one of the input variables, in this case θ_1. We then train an emulator using only the first $n-1$ ensemble members, before predicting the nth ensemble member.

One-step-ahead diagnostics primarily test whether the uncertainty estimates of the emulator are accurate. Because the size of the ensemble grows, we can check more easily whether the length-scale and covariance structure of the emulator are satisfactory.



Calibration Framework

Kennedy and O'Hagan 2001

Assume that reality $\zeta(t)$ is the computer model run at the 'true' value of the parameter $\hat{\theta}$ plus model error:

$$\zeta(t) = \eta(t, \hat{\theta}) + \delta(t)$$

Calibration Framework

Kennedy and O'Hagan 2001

Assume that reality $\zeta(t)$ is the computer model run at the 'true' value of the parameter $\hat{\theta}$ plus model error:

$$\zeta(t) = \eta(t, \hat{\theta}) + \delta(t)$$

We observe reality plus noise:

$$\mathcal{D}_{field}(t) = \zeta(t) + \epsilon(t)$$

so that

$$\mathcal{D}_{field}(t) = \eta(t, \hat{\theta}) + \delta(t) + \epsilon(t).$$

Calibration Framework

Kennedy and O'Hagan 2001

Assume that reality $\zeta(t)$ is the computer model run at the 'true' value of the parameter $\hat{\theta}$ plus model error:

$$\zeta(t) = \eta(t, \hat{\theta}) + \delta(t)$$

We observe reality plus noise:

$$\mathcal{D}_{field}(t) = \zeta(t) + \epsilon(t)$$

so that

$$\mathcal{D}_{field}(t) = \eta(t, \hat{\theta}) + \delta(t) + \epsilon(t).$$

We then aim to find $\pi(\hat{\theta}|\mathcal{D}_{sim}, \mathcal{D}_{field})$.

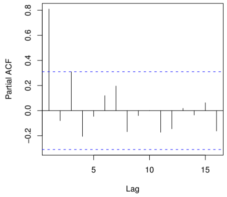
Model Discrepancy

The calibration framework used is:

$$\mathcal{D}_{field}(t) = \eta(\theta, t) + \delta(t) + \epsilon(t)$$

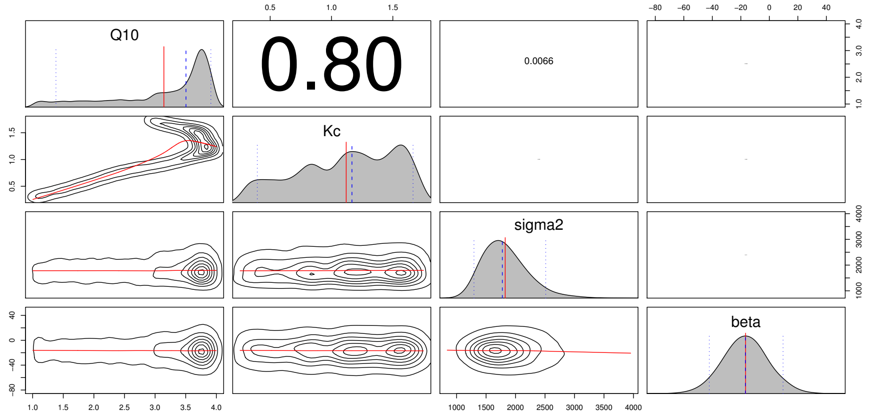
The model predicts the underlying trend, but real climate fluctuates around this. We model

- discrepancy as an AR1 process: $\delta(0) \sim N(0, \sigma_\delta^2)$, and $\delta(t) = \rho\delta(t-1) + N(0, \sigma_\delta^2)$.
- Measurement error as heteroscedastic independent random noise $\epsilon(t) \sim N(0, \lambda(t))$.



Once we have specified all these choices, we can then find the posterior using an MCMC scheme.

Results


Q10
0.80
0.0066
Kc
sigma2
beta
0.5
1.0
1.5
−80
−60
−40
−20
0
20
40
4.0
3.0
2.0
1.0
1000
2000
3000
4000
1500
2500
3500
2.5
3.5

Example 2: simulator discrepancy

All models are wrong, but ...

Kennedy and O'Hagan 2001

Lets acknowledge that most models are imperfect.

All models are wrong, but ...

Kennedy and O'Hagan 2001

Lets acknowledge that most models are imperfect. Consequently,

- predictions will be wrong, or will be made with misleading degree of confidence
- solving the inverse problem $y(x) = f(x, \theta) + e$ may not give sensible estimates of θ.
  - e is measurement error
  - $f(x, \theta)$ is our computer model
  - y is our data

All models are wrong, but ...

Kennedy and O'Hagan 2001

Lets acknowledge that most models are imperfect. Consequently,

- predictions will be wrong, or will be made with misleading degree of confidence
- solving the inverse problem $y(x) = f(x, \theta) + e$ may not give sensible estimates of θ.
  - e is measurement error
  - $f(x, \theta)$ is our computer model
  - y is our data

Can we

- account for the error?
- correct the error?

Dynamic models

Wilkinson *et al.* 2011

Kennedy and O'Hagan (2001) suggested we introduce reality ζ into our statistical inference

- Reality $\zeta(x) = f(x, \hat{\theta}) + \delta(x)$, the best model prediction plus model error $\delta(x)$.
- Data $y(x) = \zeta(x) + e$ where e represents measurement error

In many cases, we may have just a single realisation from which to learn δ

Dynamic models

Wilkinson *et al.* 2011

Kennedy and O'Hagan (2001) suggested we introduce reality ζ into our statistical inference

- Reality $\zeta(x) = f(x, \hat{\theta}) + \delta(x)$, the best model prediction plus model error $\delta(x)$.
- Data $y(x) = \zeta(x) + e$ where e represents measurement error

In many cases, we may have just a single realisation from which to learn δ

For dynamical systems the model sequentially makes predictions before then observing the outcome.

- Embedded in this process is information about how well the model performs for a single time-step.
- We can specify a class of models for the error, and then try to learn about the error from our predictions and the realised data.

Mathematical Framework

Suppose we have

- State vector x_t which evolves through time. Let $x_{0:T}$ denote $(x_0, x_1, \dots, x_T)$.

Mathematical Framework

Suppose we have

- State vector x_t which evolves through time. Let $x_{0:T}$ denote $(x_0, x_1, \ldots, x_T)$.
- Computer model f which encapsulates our beliefs about the dynamics of the state vector

$$x_{t+1} = f(x_t, u_t)$$

which depends on forcings u_t. We treat f as a black-box.

Mathematical Framework

Suppose we have

- State vector x_t which evolves through time. Let $x_{0:T}$ denote $(x_0, x_1, \dots, x_T)$.
- Computer model f which encapsulates our beliefs about the dynamics of the state vector

$$x_{t+1} = f(x_t, u_t)$$

  which depends on forcings u_t. We treat f as a black-box.
- Observations

$$y_t = h(x_t)$$

  where $h(\cdot)$ usually contains some stochastic element

Moving from white to coloured noise

A common approach is to treat the model error as white noise

- State evolution: $x_{t+1} = f(x_t, u_t) + \epsilon_t$ where ϵ_t are iid rvs.

Moving from white to coloured noise

A common approach is to treat the model error as white noise

- State evolution: $x_{t+1} = f(x_t, u_t) + \epsilon_t$ where ϵ_t are iid rvs.

Instead of the white noise model error, we ask whether there is a stronger signal that could be learnt:

- State evolution: $x_{t+1} = f(x_t, u_t) + \delta(x_t, u_t) + \epsilon_t$
- Observations: $y_t = h(x_t)$.

Moving from white to coloured noise

A common approach is to treat the model error as white noise

- State evolution: $x_{t+1} = f(x_t, u_t) + \epsilon_t$ where ϵ_t are iid rvs.

Instead of the white noise model error, we ask whether there is a stronger signal that could be learnt:

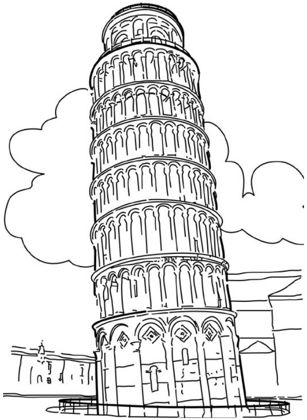
- State evolution: $x_{t+1} = f(x_t, u_t) + \delta(x_t, u_t) + \epsilon_t$
- Observations: $y_t = h(x_t)$.

Our aim is to learn a functional form plus stochastic error description of δ

Why this is difficult?

- $x_{0:T}$ is usually unobserved, but given observations $y_{0:T}$ and a fully specified model we can infer $x_{0:T}$.
  - the filtering/smoothing problem
- When we want to learn the discrepancy $\delta(x)$ we are in the situation where we estimate δ from $x_{0:T}, \ldots$
- but we must estimate $x_{0:T}$ from a description of δ.

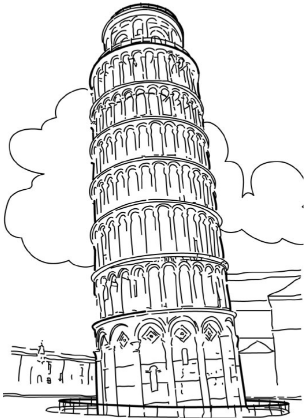
Toy Example: Freefall



Consider an experiment where we drop a weight from a tower and measure its position x_t every Δt seconds.

- Noisy observation: $y_n \sim N(x_n, \sigma^2_{obs})$

Toy Example: Freefall



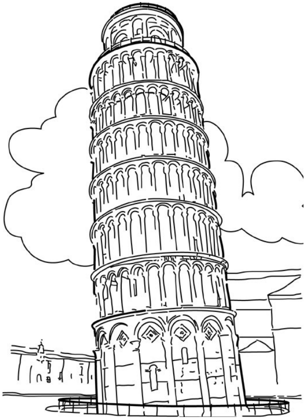
Consider an experiment where we drop a weight from a tower and measure its position x_t every Δt seconds.

- Noisy observation: $y_n \sim N(x_n, \sigma^2_{obs})$

Suppose we are given a computer model based on

$$\frac{\mathrm{dv}}{\mathrm{dt}} = g$$

Toy Example: Freefall



Consider an experiment where we drop a weight from a tower and measure its position x_t every Δt seconds.

- Noisy observation: $y_n \sim N(x_n, \sigma^2_{obs})$

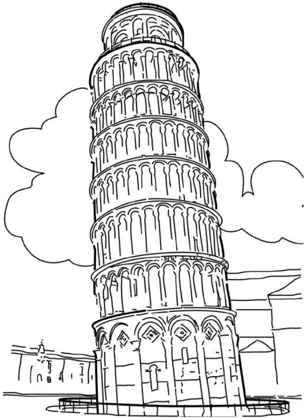
Suppose we are given a computer model based on

$$\frac{\mathrm{dv}}{\mathrm{dt}} = g$$

Which gives predictions at the observations of

- $x_{n+1} = x_n + v_k \Delta t + \frac{1}{2} g (\Delta t)^2$
- $v_{n+1} = v_n + g \Delta t$

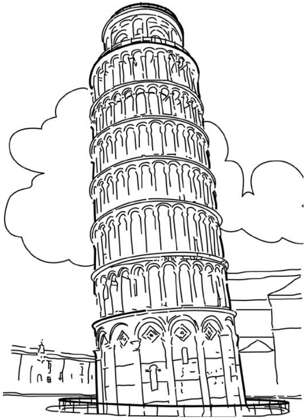
Toy Example: Freefall



Assume that the ‘true’ dynamics include a Stokes’ drag term

$$\frac{\mathrm{dv}}{\mathrm{dt}} = g - kv$$

Toy Example: Freefall



Assume that the 'true' dynamics include a Stokes' drag term

$$\frac{\mathrm{dv}}{\mathrm{dt}} = g - kv$$

Which gives single time step updates

$$x_{n+1} = x_n + \frac{1}{k}(\frac{g}{k} - v_t)(\mathrm{e}^{-\mathrm{k}\Delta\mathrm{t}} - 1) + \frac{\mathrm{g}\Delta\mathrm{t}}{\mathrm{k}}$$
$$v_{n+1} = (v_n - \frac{g}{k})\mathrm{e}^{-\mathrm{k}\Delta\mathrm{t}} + \frac{\mathrm{g}}{\mathrm{k}}$$

Model Error Term

In this toy problem, the true discrepancy function can be calculated.

- It is a two dimensional function

$$\delta = \begin{pmatrix} \delta_x \\ \delta_v \end{pmatrix} = \zeta - f$$

giving the difference between the one time-step ahead dynamics of reality and the prediction from our model.

If we expand $\mathrm{e}^{-\mathrm{k}\Delta\mathrm{t}}$ to second order we find

$$\delta(x, v, t) = \begin{pmatrix} \delta_x \\ \delta_v \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{-gk(\Delta t)^2}{2} \end{pmatrix} - v_t \begin{pmatrix} \frac{k(\Delta t)^2}{2} \\ k\Delta t(1 - \frac{k\Delta t}{2}) \end{pmatrix}$$

Model Error Term

In this toy problem, the true discrepancy function can be calculated.

- It is a two dimensional function

$$\delta = \begin{pmatrix} \delta_x \\ \delta_v \end{pmatrix} = \zeta - f$$

giving the difference between the one time-step ahead dynamics of reality and the prediction from our model.

If we expand $\mathrm{e}^{-\mathrm{k}\Delta \mathrm{t}}$ to second order we find

$$\delta(x, v, t) = \begin{pmatrix} \delta_x \\ \delta_v \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{-gk(\Delta t)^2}{2} \end{pmatrix} - v_t \begin{pmatrix} \frac{k(\Delta t)^2}{2} \\ k\Delta t(1 - \frac{k\Delta t}{2}) \end{pmatrix}$$

This is solely a function of v.

- Note, to learn δ we only have the observations $y_1, \ldots, y_n$ of $x_1, \ldots, x_n$ - we do not observe v.

Expected form of the discrepancy

Forget the previous slide.

Expected form of the discrepancy

Forget the previous slide.

There are three variables in this problem, displacement, velocity and time (x, v, t) so we might think to model δ as a function of these three terms.

Expected form of the discrepancy

Forget the previous slide.

There are three variables in this problem, displacement, velocity and time (x, v, t) so we might think to model δ as a function of these three terms.

However, universality suggests that δ should be independent of x and t.

Expected form of the discrepancy

Forget the previous slide.

There are three variables in this problem, displacement, velocity and time (x, v, t) so we might think to model δ as a function of these three terms.

However, universality suggests that δ should be independent of x and t.

With input from an experienced user of our model, it is feasible we might be able to get other information such as that δ approximately scales with v, or at least that the error is small at low speeds and large at high speeds.

Basic idea

We can use GPs as non-parametric models of the simulator discrepancy

$$\delta(\cdot) \sim GP(m(\cdot), \sigma^2 c(\cdot, \cdot))$$

Basic idea

We can use GPs as non-parametric models of the simulator discrepancy

$$\delta(\cdot) \sim GP(m(\cdot), \sigma^2 c(\cdot, \cdot))$$

We can infer $\delta(x)$ by looping around two steps:

1. Given an estimate for δ, estimate the true trajectory $x_{0:T}$ from $\pi(x_{0:T} \mid y_{0:T}, \delta)$.
2. Given samples from $\pi(x_{0:T} \mid y_{0:T}, \delta)$, estimate a value for δ.

An EM style argument provides the formal justification for why this approach should work.

Basic idea

We can use GPs as non-parametric models of the simulator discrepancy

$$\delta(\cdot) \sim GP(m(\cdot), \sigma^2 c(\cdot, \cdot))$$

We can infer $\delta(x)$ by looping around two steps:

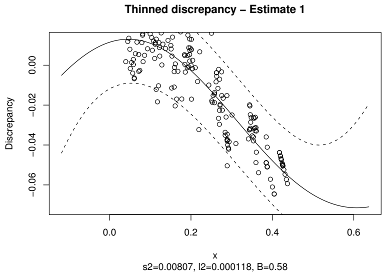
1. Given an estimate for δ, estimate the true trajectory $x_{0:T}$ from $\pi(x_{0:T} \mid y_{0:T}, \delta)$.
2. Given samples from $\pi(x_{0:T} \mid y_{0:T}, \delta)$, estimate a value for δ.

An EM style argument provides the formal justification for why this approach should work.

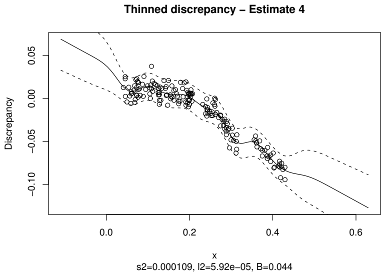
We require samples from the smoothing distribution $\pi(x_{0:T}|y_{0:T}, \theta)$

- We can generate approximate samples using the KF and its extensions, but this can be difficult to achieve good results
- Sequential Monte Carlo methods can be used to generate a more accurate approximation but at great computational cost

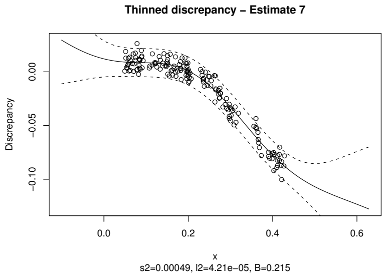
Results



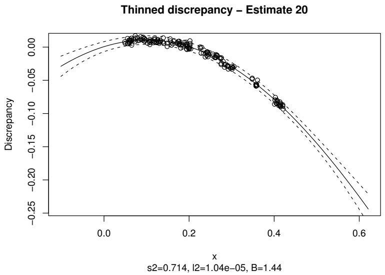
Thinned discrepancy – Estimate 1
Discrepancy
x
s2=0.00807, l2=0.000118, B=0.58



Thinned discrepancy – Estimate 4
Discrepancy
x
s2=0.000109, l2=5.92e−05, B=0.044



Thinned discrepancy – Estimate 7
Discrepancy
x
s2=0.00049, l2=4.21e−05, B=0.215



Thinned discrepancy – Estimate 20
Discrepancy
x
s2=0.714, l2=1.04e−05, B=1.44

Example 3: accelerating ABC

Approximate Bayesian Computation (ABC)

ABC algorithms are a collection of Monte Carlo methods used for calibrating simulators

- they do not require explicit knowledge of the likelihood function $\pi(x|\theta)$
- inference is done using simulation from the model (they are 'likelihood-free').

ABC methods have become popular in the biological sciences and versions of the algorithm exist in most modelling communities.

Approximate Bayesian Computation (ABC)

ABC algorithms are a collection of Monte Carlo methods used for calibrating simulators

- they do not require explicit knowledge of the likelihood function $\pi(x|\theta)$
- inference is done using simulation from the model (they are 'likelihood-free').

ABC methods have become popular in the biological sciences and versions of the algorithm exist in most modelling communities.
ABC methods can be crude but they have an important role to play.

- Scientists are building simulators (intractable ones), and fitting them to data.
  - There is a need for simple methods that can be credibly applied.
  - Likelihood methods for complex simulators are complex.
  - Modelling is something that can be done well by scientists not trained in complex statistical methods.

Uniform ABC algorithms

Uniform ABC

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(\mathcal{D}, X) \leq \epsilon$

Uniform ABC algorithms

Uniform ABC

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(\mathcal{D}, X) \leq \epsilon$

- As $\epsilon \to \infty$, we get observations from the prior, $\pi(\theta)$.
- If $\epsilon = 0$, we generate observations from $\pi(\theta \mid \mathcal{D})$

ϵ reflects the tension between computability and accuracy.

The hope is that $\pi_{ABC}(\theta) \approx \pi(\theta | D, PSH)$ for ϵ small, where PSH='perfect simulator hypothesis'

Uniform ABC algorithms

Uniform ABC

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(\mathcal{D}, X) \leq \epsilon$

- As $\epsilon \to \infty$, we get observations from the prior, $\pi(\theta)$.
- If $\epsilon = 0$, we generate observations from $\pi(\theta \mid \mathcal{D})$

ϵ reflects the tension between computability and accuracy.

The hope is that $\pi_{ABC}(\theta) \approx \pi(\theta|D, PSH)$ for ϵ small, where PSH='perfect simulator hypothesis'
There are uniform ABC-MCMC, ABC-SMC, ABC-EM, ABC-EP, ABC-MLE algorithms, etc.

Generalized ABC (GABC)

Wilkinson 2008, 2013, Fearnhead and Prangle 2012

We can generalise the rejection-ABC algorithm by using arbitrary acceptance kernels:

Generalized rejection ABC (Rej-GABC)

1. $\theta \sim \pi(\theta)$ and $X \sim \pi(x|\theta)$
2. Accept (θ, X) if

$$U \sim U[0,1] \leq \frac{\pi_{ABC}(\theta, x)}{Mg(\theta, x)} = \frac{\pi(D|X)}{\max_x \pi(D|x)}$$

Generalized ABC (GABC)

Wilkinson 2008, 2013, Fearnhead and Prangle 2012

We can generalise the rejection-ABC algorithm by using arbitrary acceptance kernels:

Generalized rejection ABC (Rej-GABC)

1. $\theta \sim \pi(\theta)$ and $X \sim \pi(x|\theta)$
2. Accept (θ, X) if

$$U \sim U[0,1] \leq \frac{\pi_{ABC}(\theta, x)}{Mg(\theta, x)} = \frac{\pi(D|X)}{\max_x \pi(D|x)}$$

In uniform ABC we take

$$\pi(D|X) = \begin{cases} 1 & \text{if } \rho(D,X) \leq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

this reduces the algorithm to

2' Accept θ ifF $\rho(D,X) \leq \epsilon$

ie, we recover the *uniform* ABC algorithm.

Uniform ABC algorithm

This allows us to interpret uniform ABC. Suppose $X, D \in \mathcal{R}$

Proposition

Accepted θ from the uniform ABC algorithm (with $\rho(D, X) = |D - X|$) are samples from the posterior distribution of θ given D where we assume $D = f(\theta) + e$ and that

$$e \sim U[-\epsilon, \epsilon]$$

In general, uniform ABC assumes that

$$D|x \sim U\{d : \rho(d, x) \leq \epsilon\}$$

We can think of this as assuming a uniform error term when we relate the simulator to the observations.

Uniform ABC algorithm

This allows us to interpret uniform ABC. Suppose $X, D \in \mathcal{R}$

Proposition

Accepted θ from the uniform ABC algorithm (with $\rho(D, X) = |D - X|$) are samples from the posterior distribution of θ given D where we assume $D = f(\theta) + e$ and that

$$e \sim U[-\epsilon, \epsilon]$$

In general, uniform ABC assumes that

$$D|x \sim U\{d : \rho(d, x) \leq \epsilon\}$$

We can think of this as assuming a uniform error term when we relate the simulator to the observations.

ABC gives 'exact' inference under a different model!

Problems with Monte Carlo methods

Monte Carlo methods are generally guaranteed to succeed if we run them for long enough.

This guarantee comes at a cost.

- Most methods sample naively - they don't learn from previous simulations.
- They don't exploit known properties of the likelihood function, such as continuity
- They sample randomly, rather than using space filling designs.

This naivety can make a full analysis infeasible without access to a large amount of computational resource.

Problems with Monte Carlo methods

Monte Carlo methods are generally guaranteed to succeed if we run them for long enough.

This guarantee comes at a cost.

- Most methods sample naively - they don't learn from previous simulations.
- They don't exploit known properties of the likelihood function, such as continuity
- They sample randomly, rather than using space filling designs.

This naivety can make a full analysis infeasible without access to a large amount of computational resource.

If we are prepared to lose the guarantee of eventual success, we can exploit the continuity of the likelihood function to learn about its shape, and to dramatically improve the efficiency of our computations.

Likelihood estimation

The GABC framework assumes

$$\begin{aligned}\pi(D|\theta) &= \int \pi(D|X)\pi(X|\theta)\mathrm{dX} \\ &\approx \frac{1}{N}\sum \pi(D|X_i)\end{aligned}$$

where $X_i \sim \pi(X|\theta)$. Or in Wood (2010),

$$\pi(D|\theta) = \phi(D; \mu_\theta, \Sigma_\theta)$$

Likelihood estimation

The GABC framework assumes

$$\begin{aligned}\pi(D|\theta) &= \int \pi(D|X)\pi(X|\theta)\mathrm{dX} \\ &\approx \frac{1}{N}\sum \pi(D|X_i)\end{aligned}$$

where $X_i \sim \pi(X|\theta)$. Or in Wood (2010),

$$\pi(D|\theta) = \phi(D; \mu_\theta, \Sigma_\theta)$$

For many problems, we believe the likelihood is continuous and smooth, so that $\pi(D|\theta)$ is similar to $\pi(D|\theta')$ when $\theta - \theta'$ is small

We can model $\pi(D|\theta)$ and use the model to find the posterior in place of running the simulator.

Example: Ricker Model

The Ricker model is one of the prototypic ecological models.

- used to model the fluctuation of the observed number of animals in some population over time
- It has complex dynamics and likelihood, despite its simple mathematical form.

Ricker Model

- Let N_t denote the number of animals at time t.

$$N_{t+1} = rN_t e^{-N_t + e_r}$$

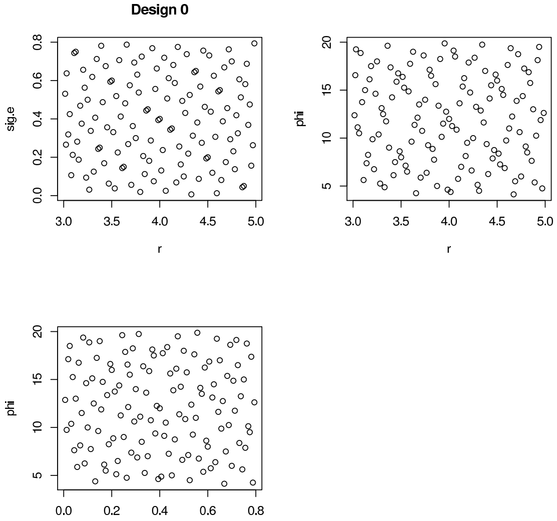
  where e_t are independent $N(0, \sigma_e^2)$ process noise
- Assume we observe counts y_t where

$$y_t \sim Po(\phi N_t)$$

Used in Wood to demonstrate the synthetic likelihood approach.

Design 1 - 128 pts

We use a Sobol sequence on the prior input space to find a design $\{\theta_i\}_{i=1}^d$. We estimate the likelihood at each point in the design, and aim to fit a GP model to estimate the likelihood at θ values not in the design.



Difficulties

i. The likelihood is too difficult to model, so we model the log-likelihood instead.

$$\hat{l}(D|\theta) = \log\left(\frac{1}{N}\sum \pi(D|X_i)\right)$$

ii. Use bootstrapped replicates of the log-likelihood to estimate the variance of the nugget term (we could estimate it as part of the GP fitting, but typically this is very poorly behaved).

Difficulties

i. The likelihood is too difficult to model, so we model the log-likelihood instead.

$$\hat{l}(D|\theta) = \log\left(\frac{1}{N}\sum \pi(D|X_i)\right)$$

ii. Use bootstrapped replicates of the log-likelihood to estimate the variance of the nugget term (we could estimate it as part of the GP fitting, but typically this is very poorly behaved).

iii. $\mathbb{V}\text{ar}(\hat{l}(D|\theta))$ is far from constant as a function of θ - this causes a problem as simple GP covariance functions assume the nugget is constant through space.

   1. Crude fix: pick a small sensible value estimated for a θ value near the mode of the posterior.

History matching waves

The log-likelihood varies across too wide a range of values, e.g., -10 near the mode, but essentially $-\infty$ at the extremes of the prior range. Consequently, any Gaussian process model will struggle to model the log-likelihood across the entire input range.

History matching waves

The log-likelihood varies across too wide a range of values, e.g., -10 near the mode, but essentially $-\infty$ at the extremes of the prior range. Consequently, any Gaussian process model will struggle to model the log-likelihood across the entire input range.

- To fix this we introduce the idea of waves, similar to those used in Michael Goldstein's approach to history-matching.
- In each wave, we build a GP model that can rule out large swathes of space as *implausible*.

History matching waves

The log-likelihood varies across too wide a range of values, e.g., -10 near the mode, but essentially $-\infty$ at the extremes of the prior range. Consequently, any Gaussian process model will struggle to model the log-likelihood across the entire input range.

- To fix this we introduce the idea of waves, similar to those used in Michael Goldstein's approach to history-matching.
- In each wave, we build a GP model that can rule out large swathes of space as *implausible*.

  We decide that θ is implausible if

  $$m(\theta) + 3\sigma < \max_{\theta_i} \log \pi(D|\theta_i) - 10$$

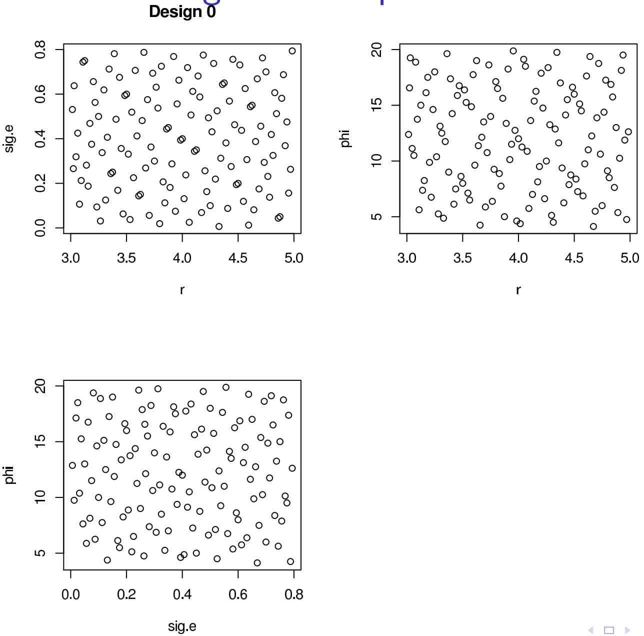
  where $m(\theta)$ is the Gaussian process estimate of $\log \pi(D|\theta)$, and σ is the variance of the GP estimate.

  - We subtract 10, as for the Ricker model, a difference of 10 on the log scale between two likelihoods, means that assigning the θ with the smaller log-likelihood a posterior density of 0 (by saying it is implausible) is a good approximation.

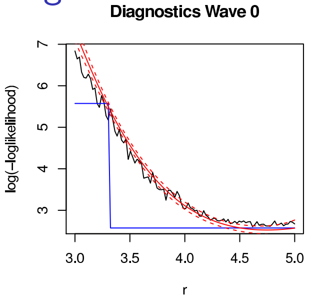
Difficulties II ctd

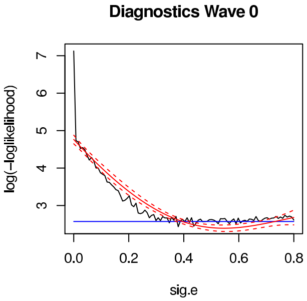
- This still wasn't enough in some problems, so for the first wave we model $\log(-\log \pi(D|\theta))$
- For the next wave, we begin by using the Gaussian processes from the previous waves to decide which parts of the input space are implausible.
- We then extend the design into the not-implaussible range and build a new Gaussian process
- This new GP will lead to a new definition of implausibility
- ...

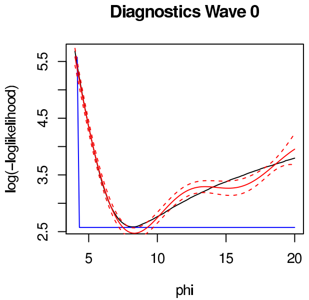
Results - Design 1 - 128 pts



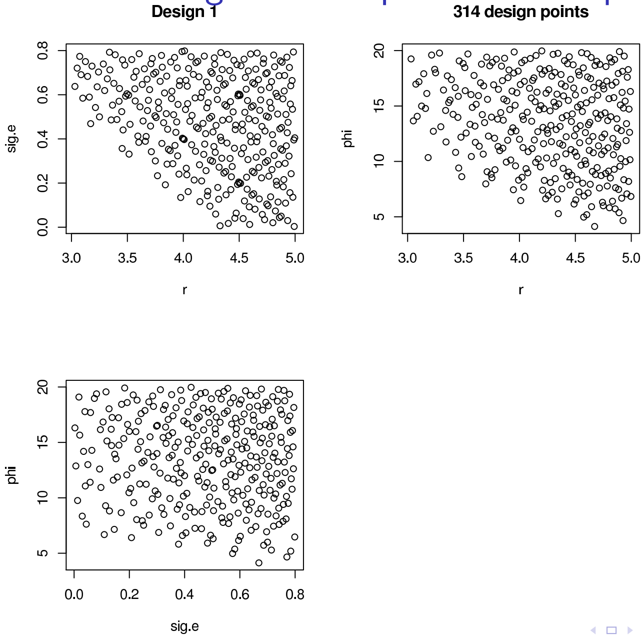
Diagnostics for GP 1 - threshold = 5.6



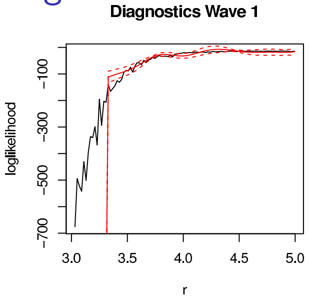


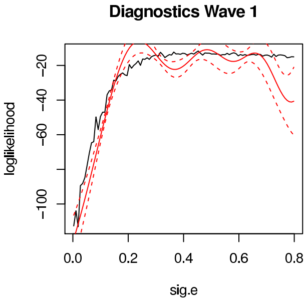


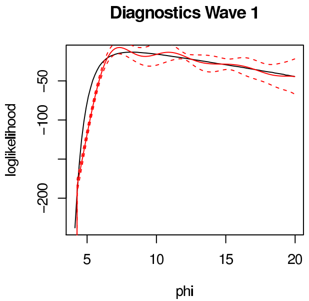
Results - Design 2 - 314 pts - 38% of space implausible



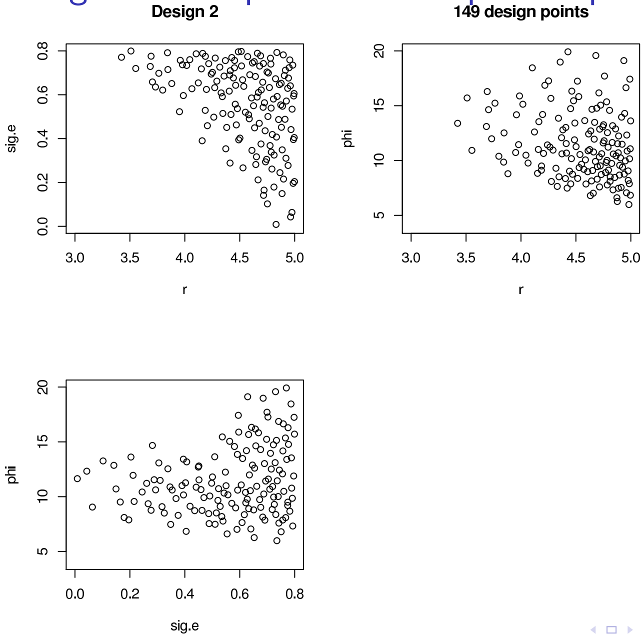
Diagnostics for GP 2 - threshold = -21.8



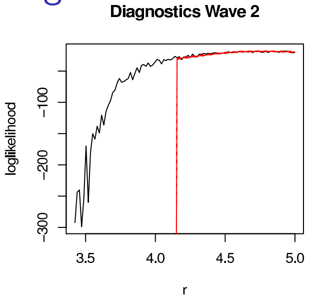


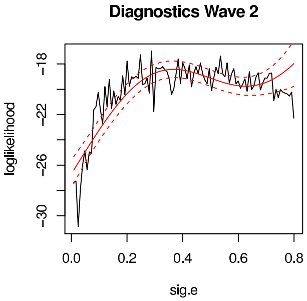


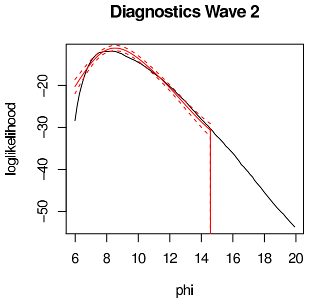
Design 3 - 149 pts - 62% of space implausible



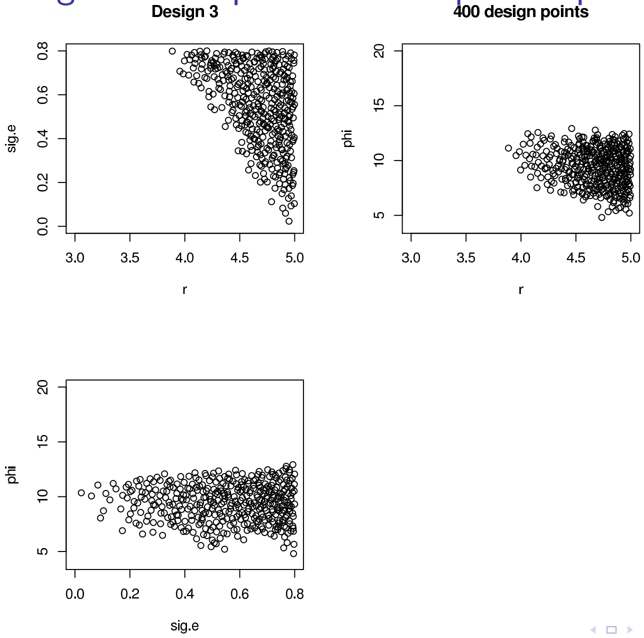
Diagnostics for GP 3 - threshold = -20.7



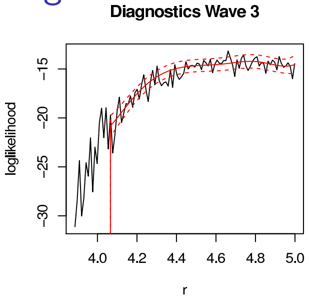


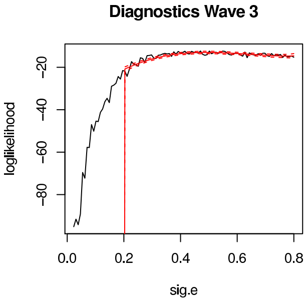


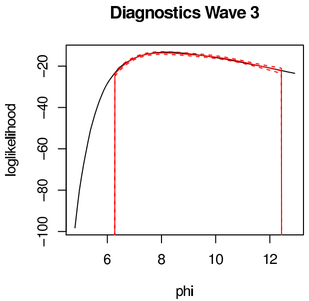
Design 4 - 400 pts - 95% of space implausible



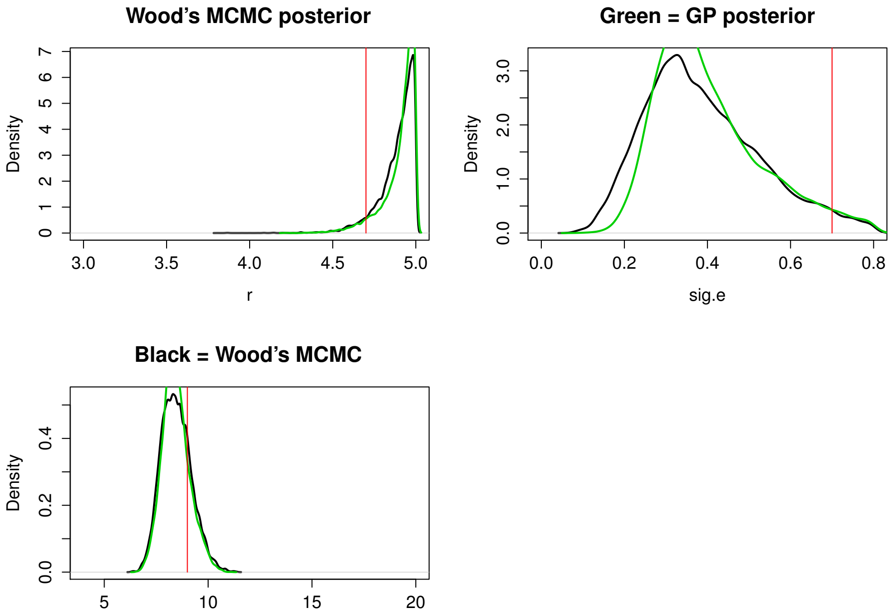
Diagnostics for GP 4 - threshold = -16.4







MCMC Results



Computational details

- The Wood MCMC method used $10^5 \times 500$ simulator runs
- The GP code used $(128 + 314 + 149 + 400) = 991 \times 500$ simulator runs
  - 1/100th of the number used by Wood's method.

By the final iteration, the Gaussian processes had ruled out over 98% of the original input space as implausible,

- the MCMC sampler did not need to waste time exploring those regions.

Conclusions

References

- MUCM toolkit. `www.mucm.aston.ac.uk/MUCM/MUCMToolkit`
- Kennedy M and O'Hagan A 2001 Bayesian calibration of computer models (with discussion). *Journal of the Royal Statistical Society, Series B* **63**, 425–464.
- Higdon D, Gattiker J, Williams B and Rightley M 2008 Computer model calibration using high-dimensional output. *JASA* **103**, 570-583.
- Oakley, J. and O'Hagan, A. (2002). Bayesian inference for the uncertainty distribution of computer model outputs. Biometrika 89, 769-784.
- Oakley, J. and O'Hagan, A. (2004). Probabilistic sensitivity analysis of complex models: a Bayesian approach. Journal of the Royal Statistical Society, Series B 66, 751-769.
- Wilkinson, *Bayesian calibration of expensive multivariate computer experiments*. In 'Large-scale inverse problems and quantification of uncertainty', 2010, Wiley.
- Wilkinson, M. Vrettas, D. Cornford, J. E. Oakley, *Quantifying simulator discrepancy in discrete-time dynamical simulators*. Journal of Agricultural, Biological, and Environmental Statistics: 16(4), 554-570, 2011.
- Wilkinson, *Approximate Bayesian computation (ABC) gives exact results under the assumption of model error*, *Statistical Applications in Genetics and Molecular Biology*, 12(2), 129-142, 2013.
- Wilkinson, *Accelerating ABC methods using Gaussian processes*. In submission.