Model-based Synthesis

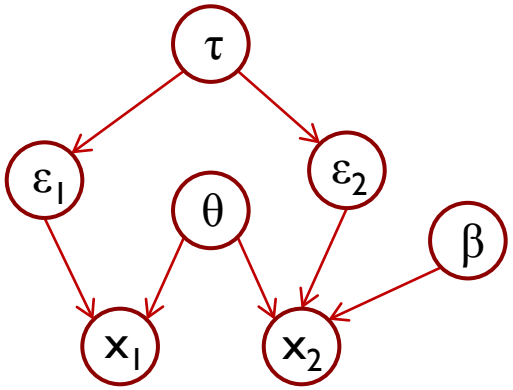
Tony O'Hagan

Stochastic models

Synthesising evidence through a statistical model

Graphical modelling

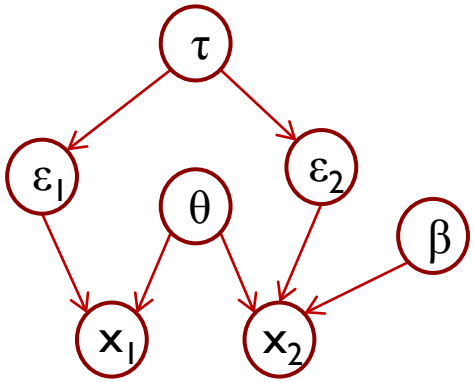
- The kinds of models that we were creating in the meta-analysis section of the previous talk can all be considered in the broad framework of statistical modelling
- Many people find it useful to show the logic of such models graphically



- A model like this is called a DAG (Direct Acyclic Graph)
  - Some powerful computational algorithms operate on DAGs
  - To compute posterior distributions for nodes of interest

DAGs

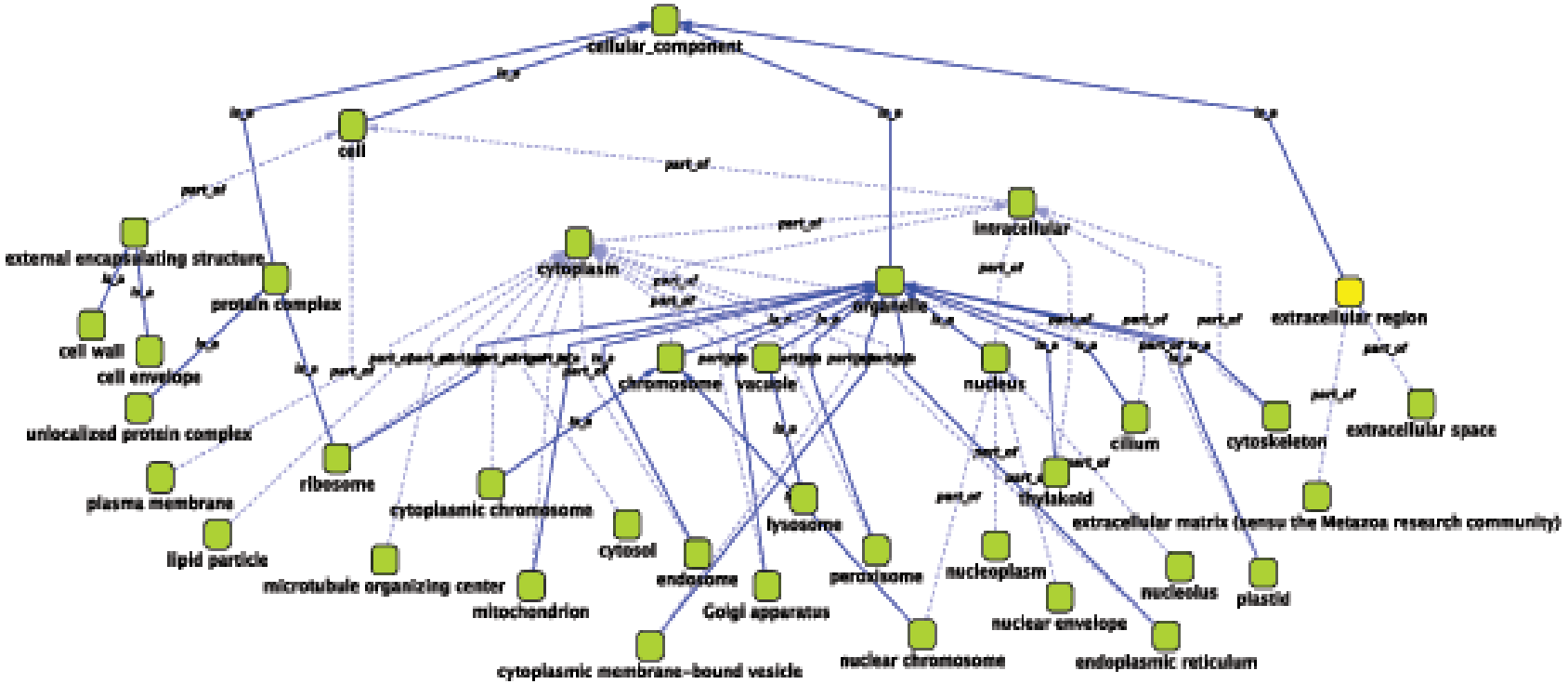
- A DAG is a graph with nodes representing different quantities
  - Parameters, data, random effects
- And with arcs joining selected nodes
  - Showing that there is a link between them
- Rules
  - All arcs are directed, i.e. have an arrow showing direction
    - Quantity at start node (parent) influences the one at the end node (child)
  - A distribution is required for each node
    - Conditional on *all* its parents
    - Although some nodes may be deterministic
  - No cycles are allowed
    - A cycle is a sequence of arcs pointing round in a loop



DAGs (contd.)

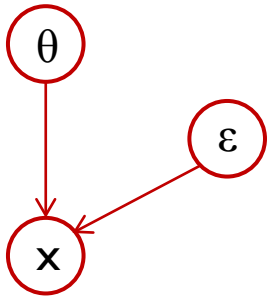
- Absence of an arc is as important as presence
  - It says that these two nodes are conditionally independent given all their parents
- DAGs are sometimes called influence diagrams
- They may also be called expert systems
  - Particularly a large graph relating to a specific complex task
  - And where much of the structure (arcs and conditional distributions) comes from expert knowledge
    - Rather than directly from data

A larger DAG (for protein biosynthesis)



learning in a DAG

- The DAG implicitly specifies a large joint distribution across all the nodes
- If we learn the values of one or more nodes in the DAG, we can update the system to condition on these
  - The result is a posterior joint distribution on the remaining nodes
  - From which we can obtain marginal distributions of nodes of interest
- Bayes' theorem is the simplest of all cases
  - Prior and likelihood specify joint distribution for θ and x
  - Bayes' theorem conditions on observed x
  - To give posterior distribution of θ given x



Computation

- I mentioned that we have nice computational tools to handle this kind of synthesis
  - Markov chain Monte Carlo
    - Powerful tool for computing posterior distributions from general Bayesian models
      - By drawing a large sample from the joint posterior distribution
      - Model must be representable as a DAG
    - Software: WinBUGS
  - Message-passing algorithms
    - Used in large expert system DAGs
    - Very powerful when all nodes are discrete
      - Methods are more patchy and complex for continuous nodes

Mechanistic models

Synthesising evidence through a complex simulator

Health economic modelling

- In order to make decisions on the cost-effectiveness of medical treatments, health economists build models
  - Evidence for effectiveness may come from clinical trials
    - Possibly involving meta-analysis
  - But with even bigger data gaps
    - Trials look at short-term effects, with selected patient groups
    - Health care providers need to look at long-term benefits for the general population
  - Effectiveness needs to be on a common scale of health benefit
    - Quality-adjusted life years
    - Based on a lot of other evidence about how people value health
  - And we need evidence about costs
    - Unit costs and resource usage

Simulators

- Health economic models are instances of a wider phenomenon
- In almost all fields of science, technology, industry and policy making, people use mechanistic models to simulate complex real-world processes
  - For understanding, prediction, control
- Usually implemented in computer codes
  - Often very computationally intensive
  - We'll call them simulators
- There is a growing realisation of the importance of uncertainty in simulator predictions
  - Can we trust them?
  - Without any quantification of output uncertainty, it's easy to dismiss them

Examples

- Climate prediction
- Molecular dynamics
- Nuclear waste disposal
- Oil fields
- Engineering design
- Hydrology

Simulators and synthesis

- Simulators synthesise a lot of different kinds of evidence
  - Scientific understanding about how the outputs are driven by the inputs
    - Often including empirical relationships derived from experimental evidence
  - Knowledge about the inputs
    - Obtained from a variety of sources
    - Data (usually with gaps), expert judgements
- They are used to make predictions about the outputs
  - Based on synthesis of this evidence
  - But we need to be careful about all the sources of uncertainty

Sources of uncertainty

- A simulator takes inputs x and produces outputs
  y = f(x)

- How might y differ from the true real-world value z that the simulator is supposed to predict?
  - Error in inputs x
    - Initial values, forcing inputs, model parameters
  - Error in model structure or solution
    - Wrong, inaccurate or incomplete science
    - Bugs, solution errors
    - Model discrepancy – a kind of gap

Quantifying uncertainty

- The ideal is to provide a probability distribution p(z) for the true real-world value
  - The centre of the distribution is a best estimate
  - Its spread shows how much uncertainty about z is induced by uncertainties on the last slide
- How do we get this?
  - Input uncertainty: characterise p(x), propagate through to p(y)
    - For example, use Monte Carlo sampling
    - Generate random sample of x values from p(x), run the model for each to get a random sample from p(y)
  - Structural uncertainty: characterise p(z-y)

Reducing uncertainty

- To reduce uncertainty, get more information!
- Informal – more/better science
  - Tighten p(x) through improved understanding
  - Tighten p(z-y) through improved modelling or programming
- Formal – using real-world data
  - Calibration – learn about model parameters
  - Data assimilation – learn about the state variables
    - In a dynamic simulator
  - Learn about structural error z-y
  - Validation

So far, so good

- In principle, all this is straightforward
- In practice, there are many technical difficulties
  - Formulating uncertainty on inputs
    - Elicitation of expert judgements
  - Propagating input uncertainty
  - Modelling structural error
  - Anything involving observational data!
    - The last two are intricately linked
  - And *computation*

The problem of big models

- Key tasks require us to run the simulator many times
- Uncertainty propagation
  - Implicitly, we need to run f(x) at all possible x
  - Monte Carlo works by taking a sample of x from p(x)
  - Typically needs thousands of simulator runs
- Calibration
  - Learn about uncertain inputs from observations of the real process
  - Traditionally this is done by searching the x space for good fits to the data
- These tasks are impractical if the simulator takes more than a few seconds to run
  - We need a more efficient technique

Gaussian process representation

- More efficient approach
  - First work in early 1980s
- Consider the simulator as an unknown function
  - f(.) becomes a random process
  - We represent it as a Gaussian process (GP)
    - Conditional on hyperparameters
    - Or its Bayes linear analogue
- Training runs
  - Run simulator for sample of x values
  - Condition GP on observed data
  - Typically requires many fewer runs than MC
    - And x values don't need to be chosen randomly

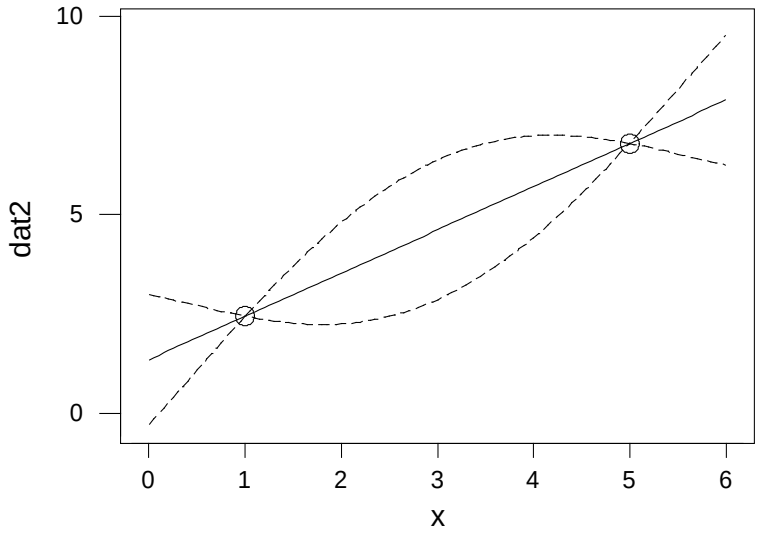
Emulation

- Analysis is completed by prior distributions for, and posterior estimation of, hyperparameters

- The posterior distribution is known as an **emulator** of the simulator
  - Posterior mean estimates what the simulator would produce for any untried x (prediction)
  - With uncertainty about that prediction given by posterior variance
  - Correctly reproduces training data

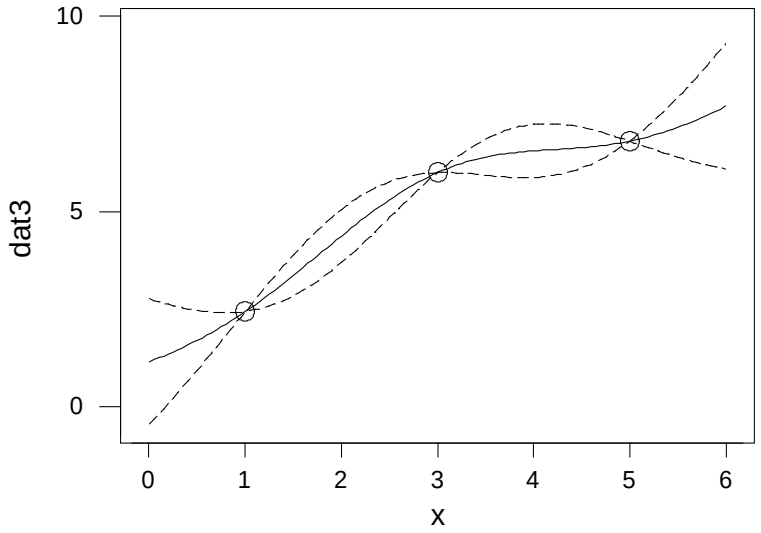
2 code runs

- Consider one input and one output
- Emulator estimate interpolates data
- Emulator uncertainty grows between data points



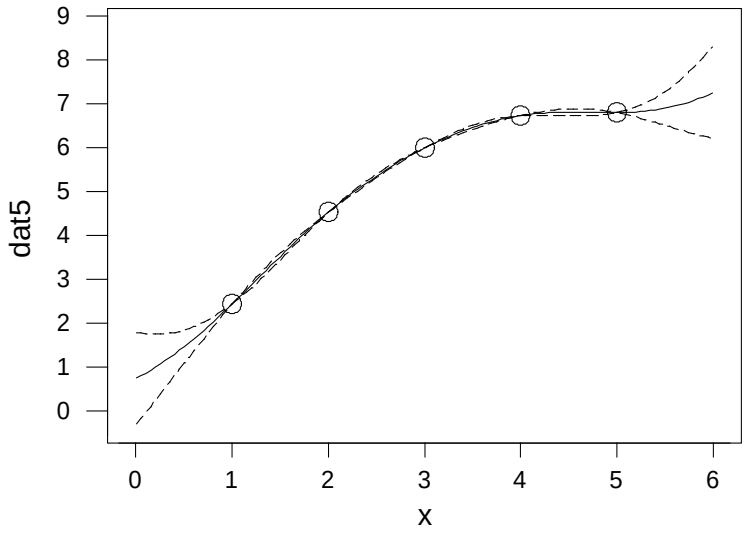
3 code runs

- Adding another point changes estimate and reduces uncertainty



5 code runs

- And so on



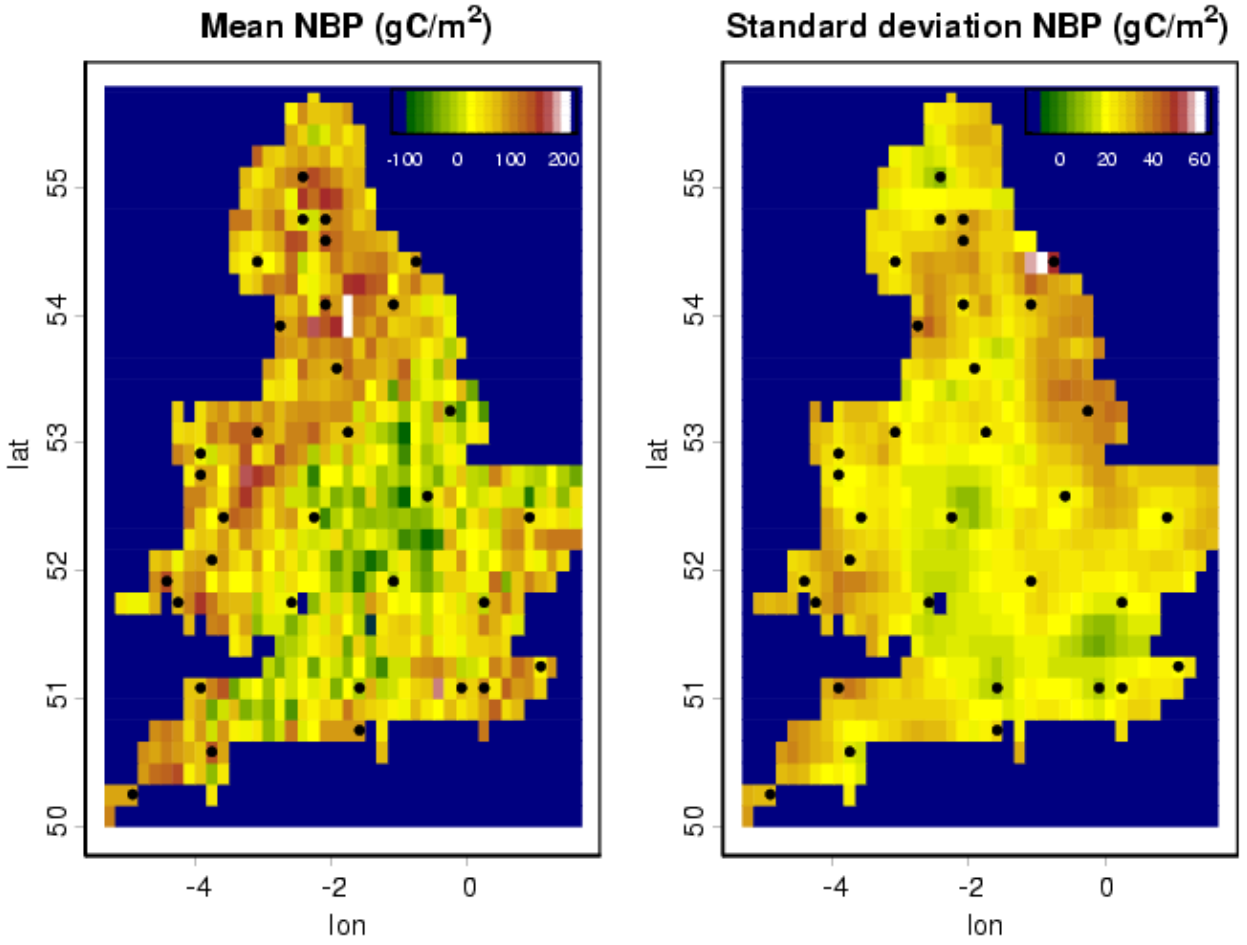
Then what?

- Given enough training data points we can emulate any simulator accurately
  - So that posterior variance is small "everywhere"
  - Typically, this can be done with orders of magnitude fewer simulator runs than traditional methods
- Use the emulator to make inference about other things of interest
  - E.g. uncertainty analysis, calibration
- Conceptually very straightforward in the Bayesian framework
  - But of course can be computationally hard

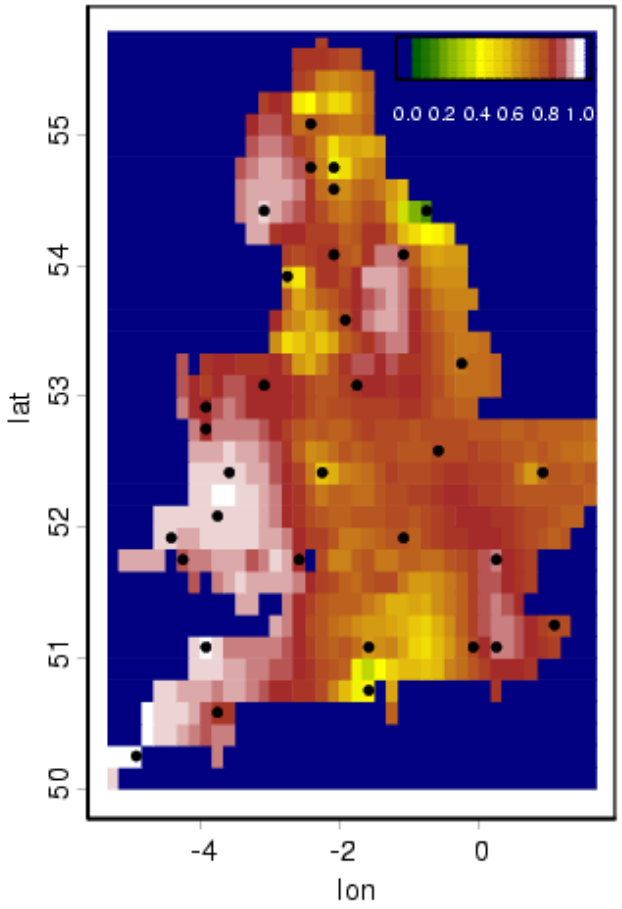
Example: UK carbon flux in 2000

- Vegetation simulator predicts carbon exchange from each of 700 pixels over England & Wales
  - Principal output is Net Biosphere Production
  - Sheffield Dynamic Global Vegetation Model (SDGVM)
- Accounting for uncertainty in inputs
  - Soil properties
  - Properties of different types of vegetation
  - Propagate input uncertainty through the model
- Aggregated to England & Wales total
  - Allowing for correlations
  - Estimate 7.61 Mt C
  - Std deviation 0.61 Mt C

Maps


Mean NBP (gC/m^2)
-100
0
100
200
lat
55
54
53
52
51
50
-4
-2
0
lon
Standard deviation NBP (gC/m^2)
0
20
40
60
lat
55
54
53
52
51
50
-4
-2
0
lon

Sensitivity analysis



- Map shows proportion of overall uncertainty in each pixel that is due to uncertainty in the vegetation parameters
  - As opposed to soil parameters
- Contribution of vegetation uncertainty is largest in grasslands/moorlands

England & Wales aggregate

PFT	Plug-in estimate (Mt C)	Mean (Mt C)	Variance (Mt C^2)
Grass	5.28	4.65	0.323
Crop	0.85	0.50	0.038
Deciduous	2.13	1.69	0.009
Evergreen	0.80	0.78	0.001
Covariances			0.001
Total	9.06	7.61	0.372

Role of emulation

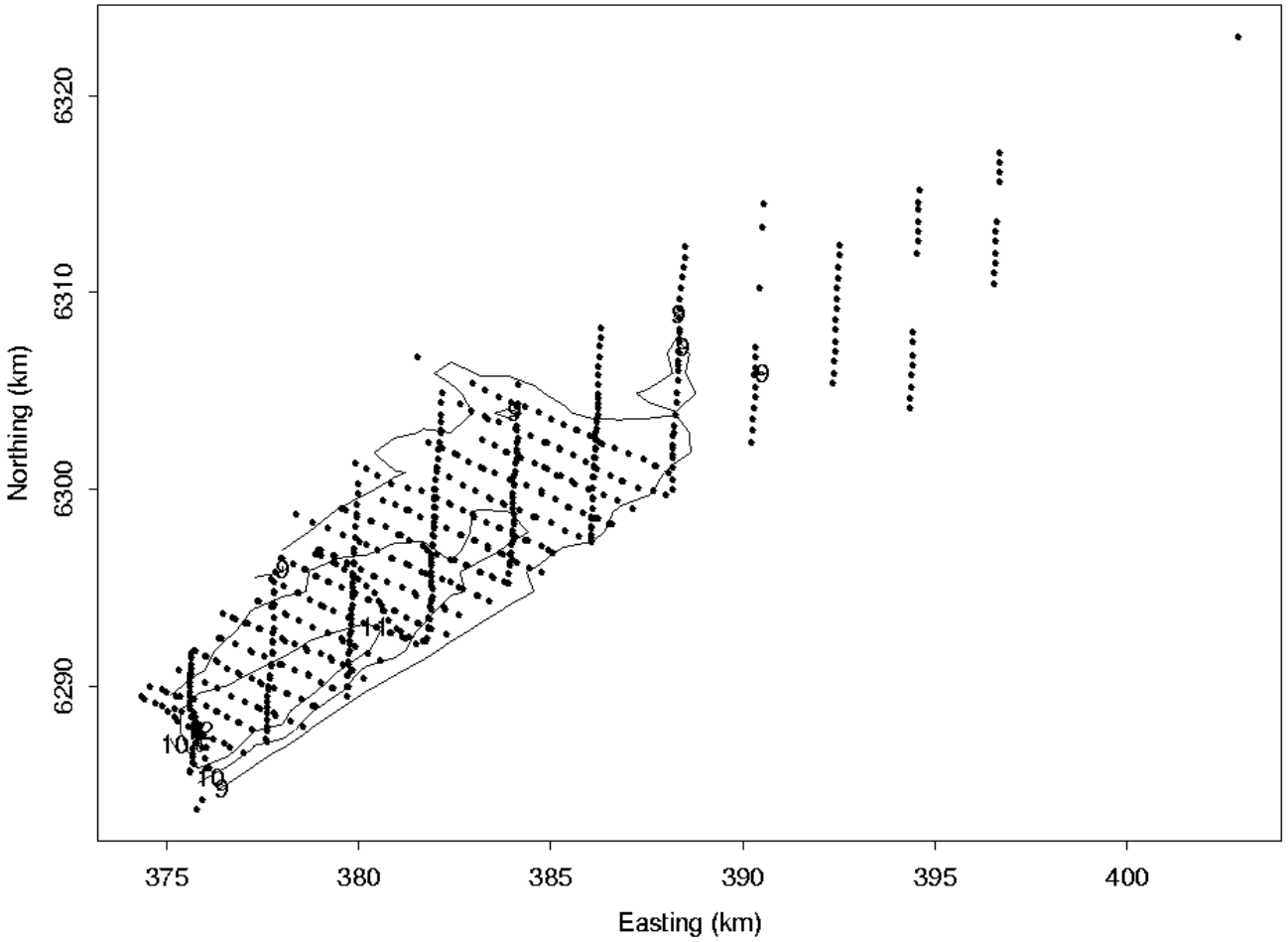
- Gaussian process emulation was crucial to the feasibility of this exercise
  - Almost 3000 simulator runs for a single set of inputs
  - Imagine this repeated hundreds or thousands of times for Monte Carlo
    - And all that repeated to evaluate the sensitivity to each input group
- We emulated each PFT at a sample of 33 sites
  - Typically 200 simulator runs for each
  - Kriging to interpolate between sites
    - Also equivalent to Gaussian process emulation

Kennedy, M. C. et al (2008). Quantifying uncertainty in the biospheric carbon flux for England and Wales. Journal of the Royal Statistical Society A 171, 109-135.

Example: Nuclear accident

- Radiation was released after an accident at the Tomsk-7 chemical plant in 1993
- Data comprise measurements of the deposition of ruthenium 106 at 695 locations obtained by aerial survey after the release
- The simulator is a simple Gaussian plume model for atmospheric dispersion
- Two calibration parameters
  - Total release of ^{106}Ru (source term)
  - Deposition velocity

Data


Northing (km)
6290
6300
6310
6320
375
380
385
390
395
400
Easting (km)

Calibration

- A small sample (N=10 to 25) of the 695 data points was used to calibrate the model
- Then the remaining observations were predicted and RMS prediction error of log-deposition computed

Sample size N	*10*	*15*	*20*	*25*
Best fit calibration	0.82	0.79	0.76	0.66
Bayesian calibration	0.49	0.41	0.37	0.38

- On a log scale, error of 0.7 corresponds to a factor of 2

Role of emulation

- The simulator in this case was fast
  - No need to emulate it
- But emulation is effectively used for the model discrepancy
  - Gaussian process representation
  - Trained on N = 10, 15, 20, 25 data points

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models (with discussion). Journal of the Royal Statistical Society B 63, 425-464.

Summary of session 3

- Synthesis of many different kinds of evidence is usually conducted by building models
- Statistical models – DAGs
  - Nodes linked through conditional probability distributions
  - Deterministic nodes allowed but usually involve simple equations
  - Computation through MCMC or message-passing algorithms
- Mechanistic models
  - Complex equations to model complex real-world processes
    - Differential equations often involved
  - Uncertainty on inputs and on model discrepancy
  - Computation facilitated through emulation