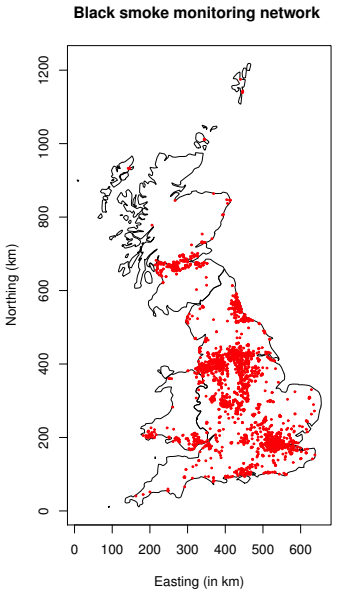
Smooth additive models for large datasets

Simon Wood
School of Mathematics, University of Bristol, U.K.

Example motivation: London smog 1952

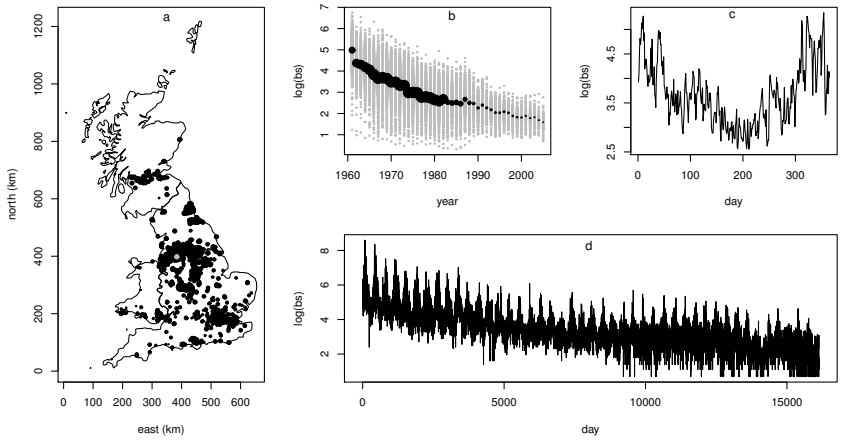
- 5-9 Dec 1952.
- 4-12 thousand premature deaths.
- Black smoke (particulates) and sulphur from domestic coal fires.
- Clean air act 1956.
- Monitoring from 1961.

Black smoke monitoring…



- 4 decades of daily 'black smoke' monitoring at a variable subset of the 2400+ stations shown.
- Started in 1961 to monitor air pollution (then mostly from coal), in wake of 1950s smog deaths.
- Epidemiological studies need estimates of *daily* exposure away from stations.
- $O(10^7)$ measurements and suitable smooth latent Gaussian models have $O(10^4)$ coefficients with 10-30 variance parameters.

Daily BS data



Black smoke modelling

- A reasonable daily black smoke model is

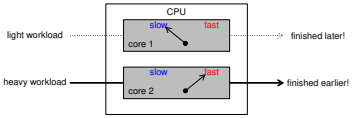
$$\begin{aligned}\log(\mathtt{bs}_i) = {} & f_1(\mathtt{y}_i) + f_2(\mathtt{doy}_i) + f_3(\mathtt{dow}_i) \\ & + f_4(\mathtt{y}_i, \mathtt{doy}_i) + f_5(\mathtt{y}_i, \mathtt{dow}_i) + f_6(\mathtt{doy}_i, \mathtt{dow}_i) \\ & + f_7(\mathtt{n}_i, \mathtt{e}_i) + f_8(\mathtt{n}_i, \mathtt{e}_i, \mathtt{y}_i) + f_9(\mathtt{n}_i, \mathtt{e}_i, \mathtt{doy}_i) + f_{10}(\mathtt{n}_i, \mathtt{e}_i, \mathtt{dow}_i) \\ & + f_{11}(\mathtt{h}_i) + f_{12}(\mathtt{T}_i^0, \mathtt{T}_i^1) + f_{13}(\bar{\mathtt{T}}\mathtt{1}_i, \bar{\mathtt{T}}\mathtt{2}_i) + f_{14}(\mathtt{r}_i) + \alpha_{k(i)} + b_{\mathrm{id}(i)} + e_i\end{aligned}$$

  The model has around 10^4 coefficients, and was well beyond previous model fitting technology.
- Even without worrying about computing time, *storing* a $10^7 \times 10^4$ model matrix requires nearly a terabyte of memory.
- We need ways to reduce the memory footprint and speed up computation. First consider some computational practicalities.

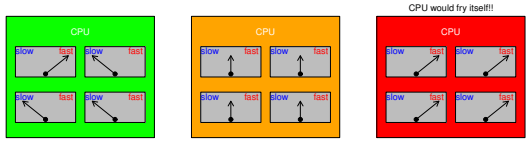
The messy realities of parallel computing

1. Hyper-threading can make parallel slower than serial...

2. Dynamic core clock speed management for power efficiency can make low work thread take most time.



3. Thermal limits: n cores are not n times faster than 1 core.



4. A floating point operation (flop) may take one or two CPU cycles, retrieving a number from memory 10 times that.
   - **Numerical computation is memory bandwidth limited**.

Memory bandwidth, Cache, block algorithms

- Cache is small fast access memory between CPU and main memory.
- Big speed up if most flops involve data already in Cache.
- Consider two 10^6 flop computations
  1. $\mathbf{C}$ is a 1000×1000 matrix, and $\mathbf{y}$ a 1000-vector. Compute $\mathbf{Cy}$. Each of 10^6 elements of $\mathbf{C}$ read once, no re-use.
  2. $\mathbf{A}$ and $\mathbf{B}$ are both 100×100 matrices. Form $\mathbf{AB}$. Repeatedly revisits the 2×10^4 elements of $\mathbf{A}$ and $\mathbf{B}$.

  ...provided $\mathbf{A}$ and $\mathbf{B}$ fit in Cache, 2 is *much* faster.
- Structure algorithms around Cache friendly blocks! e.g.

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}$$

Building a scalable method

- We need low memory footprint, multi-core scalability and numerical stability.
- Here, I'll give a flavour of what is needed to get the first two.
- In particular the regression computations require

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X} + \sum\nolimits_j \lambda_j \mathbf{S}_j)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{z}$$

  while smoothing parameter selection requires computations involving $\log|\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X} + \sum_j \lambda_j \mathbf{S}_j|$.
- Pivoted Cholesky decomposition can be made block oriented[1] so that a parallel version works, and we can build computation around this, if we can obtain $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$ efficiently.

[1] Lucas 2004, LAPACK working paper

Low memory $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$ updating

- Partition $\mathbf{X}$ row-wise into sub-matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots$, and partition $\mathbf{W}$ and $\mathbf{z}$ correspondingly.
- Forming the blocks $\mathbf{X}_j$ one at a time we can use

$$\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X} = \sum_j \mathbf{X}_j^{\mathrm{T}}\mathbf{W}_j\mathbf{X}_j$$

  to accumulate $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ without needing to form $\mathbf{X}$ whole.
- At same time we can accumulate

$$\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{z} = \sum_j \mathbf{X}_j^{\mathrm{T}}\mathbf{W}_j\mathbf{z}_j.$$

- Note that the operations count for $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$ is $O(np^2)$, while the formation of the elements of $\mathbf{X}$ is $O(np)$, so even repeated formation of the $\mathbf{X}_j$ is not a major cost for most bases.

Simple discrete method example

- For a single smooth, its $n \times p_j$ model matrix becomes

$$X_j(i,l) = \bar{X}_j(k_j(i), l)$$

  where $\bar{\mathbf{X}}_j$ is an $m_j \times p_j$ matrix evaluating the smooth at the corresponding gridded values.
- Then, for example

$$X_j^{\mathrm{T}}y = \bar{X}_j^{\mathrm{T}}\bar{y} \text{ where } \bar{y}_l = \sum_{k_j(i)=l} y_i$$

  Cost: $O(n) + O(m_j p_j)$ – for $m_j \ll n$ this a factor of p_j saving.
- In general all required (cross)products are a factor of p_j more efficient, where p_j is the largest (marginal) basis dimension involved in the term.

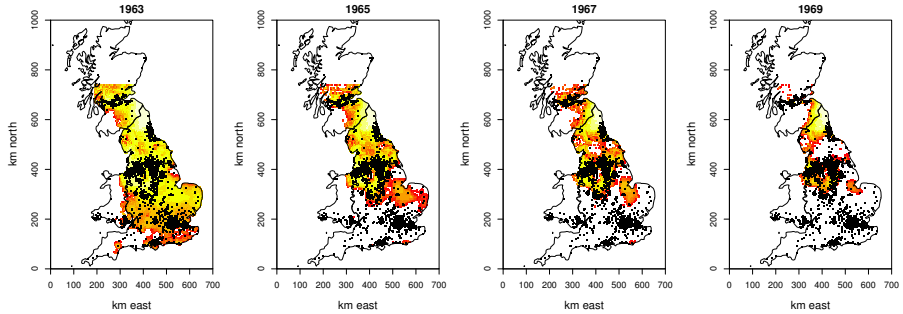
Cheaper $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$: discrete covariate methods

- Formation of $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$ is the leading order cost: $O(np^2)$.
- Lang et al.[2] point out that for a single 1D smooth, $f(x)$, the product $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$ is very efficiently computable if x has only $m \ll n$ discrete values.
- As statisticians we should be prepared to discretise x to $m = O(\sqrt{n})$ bins.
- It is possible to find (novel) efficient computational methods for the multiple discretised covariate case, both for multiple 1D smooths and for 'tensor product' smooths of multiple covariates (which also have to be parallelized).

[2] Lang, Umlauf, Wechselberger, Harttgen & Kneib, 2014, Statistics & Computing.

`bam(...,discrete=TRUE)`

- We also need a somewhat different iteration to the fitting iterations covered so far (omitted here).
- In the end the black smoke model could be estimated in an hour on a 10 core workstation using the methods built in to `mgcv:bam(...,discrete=TRUE)`.[3]
- Map shows average daily probability of exceeding current EU daily limit, for 4 years in the 1960s.



[3] Wood et al. (2017) JASA