Smooth additive models for large datasets

Simon Wood, University of Bristol, United Kingdom

in collaboration with

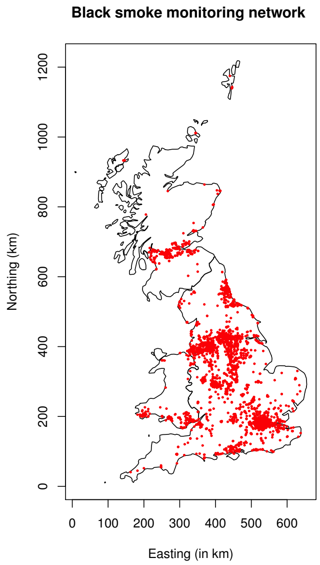
Zheyuan Li, **Gavin Shaddick, Nicole Augustin**
University of Bath, Untied Kingdom

Funded by EPSRC and the University of Bath

London smog 1952

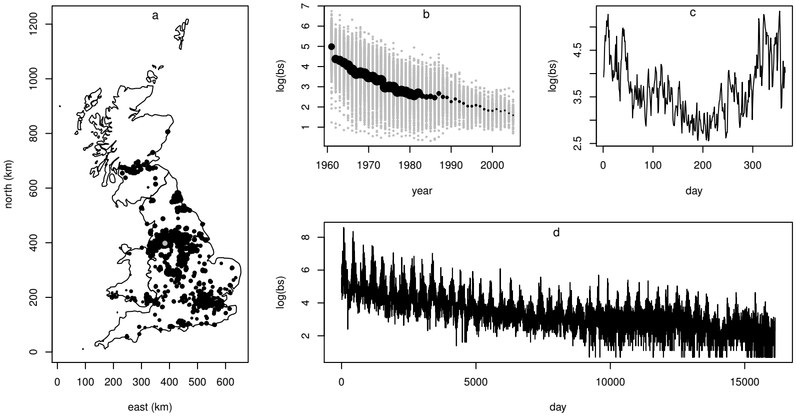
- 5-9 Dec 1952.
- Black smoke (particulates) and sulphur from domestic coal fires.
- 4-12 thousand premature deaths.
- Clean air act 1956.

Black smoke monitoring from 1961…



- 4 decades of daily 'black smoke' monitoring at a variable subset of the 2400+ stations shown.
- 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.
- Previously computationally infeasible (in mgcv or INLA)

Daily BS data



The model class

- Concentrate on models of the form $y_i \sim \text{EF}(\mu_i, \phi)$

$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\theta} + \sum_j f_j(x_{ji}) + \mathbf{Z}_i\mathbf{b}$$

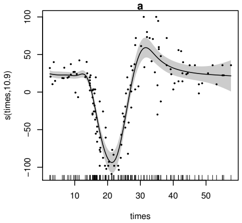
  $\mathbf{A}, \mathbf{Z}$ are model matrices, f_j are smooth functions, $\boldsymbol{\theta}$ is parameters, $\mathbf{b}$ contains independent Gaussian random effects. g is a known link function. x_j may be vector.

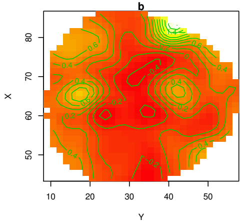
- Represent smooth functions, f, using spline basis expansions with coefficients $\boldsymbol{\beta}$

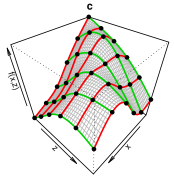
$$f(x) = \sum_{k=1}^{K} \beta_k b_k(x)$$

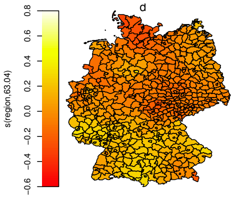
- …and define a quadratic smoothing penalty, e.g. $\int f''^2 dx = \boldsymbol{\beta}^\mathrm{T}\mathbf{S}\boldsymbol{\beta}$ (suggesting $K = O(n^{1/9-1/5})$).

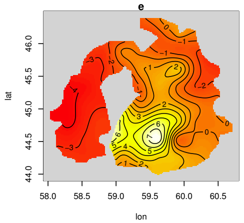
Wide range of basis penalty smoothers available

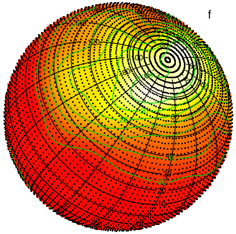












Estimation of model coefficients

- Given bases for the smooths, the model can be written as

  $$y_i \sim \text{EF}(\mu_i, \phi), \quad g(\mu_i) = \mathbf{X}_i\boldsymbol{\beta} = \eta_i$$

  where $\mathbf{X}$ ($n \times p$) contains $\mathbf{A}$, $\mathbf{Z}$ and evaluated basis functions, and $\boldsymbol{\beta}$ is a combined coefficient vector.
- The combined penalty can be written $\sum_j \lambda_j \boldsymbol{\beta}^\text{T}\mathbf{S}_j\boldsymbol{\beta}$, where the λ_j are *smoothing parameters*.
- Then $\hat{\boldsymbol{\beta}} = \text{argmax}_\beta l(\boldsymbol{\beta}) - \sum_j \lambda_j \boldsymbol{\beta}^\text{T}\mathbf{S}_j\boldsymbol{\beta}/2$.
- …this *penalized log likelihood* can be given Bayesian motivation using the prior[1]

  $$\boldsymbol{\beta} \sim N\{\mathbf{0}, (\sum_j \lambda_j \mathbf{S}_j)^-\}.$$

[1] which also covers simple Gaussian random effects.

Full fitting algorithm (PQL)

- Iterate…

[2]Breslow & Clayton, 1993, JASA

Full fitting algorithm (PQL)

- Iterate…
  1. Let $z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i) + \hat{\eta}_i$, $W_{ii}^{-1} = V(\hat{\mu}_i)g'(\hat{\mu}_i)^2$ and $\eta_i = g(\mu_i)$. V is a known and determined by EF.

[2]Breslow & Clayton, 1993, JASA

Full fitting algorithm (PQL)

- Iterate…
  1. Let $z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i) + \hat{\eta}_i$, $W_{ii}^{-1} = V(\hat{\mu}_i)g'(\hat{\mu}_i)^2$ and $\eta_i = g(\mu_i)$. V is a known and determined by EF.
  2. Use the prior on $\boldsymbol{\beta}$ and estimate

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-) \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi),$$

  including $\boldsymbol{\lambda}$, as a linear mixed model (this is PQL[2]).
  $\mathbf{S}_\lambda = \sum \lambda_j \mathbf{S}_j$. $\boldsymbol{\lambda}$ estimation uses REML.

[2] Breslow & Clayton, 1993, JASA

Full fitting algorithm (PQL)

- Iterate…
  1. Let $z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i) + \hat{\eta}_i$, $W_{ii}^{-1} = V(\hat{\mu}_i)g'(\hat{\mu}_i)^2$ and $\eta_i = g(\mu_i)$. V is a known and determined by EF.
  2. Use the prior on $\boldsymbol{\beta}$ and estimate

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-) \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi),$$

     including $\boldsymbol{\lambda}$, as a linear mixed model (this is PQL[2]). $\mathbf{S}_\lambda = \sum \lambda_j \mathbf{S}_j$. $\boldsymbol{\lambda}$ estimation uses REML.
- But $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi)$ is not true!

[2] Breslow & Clayton, 1993, JASA

Full fitting algorithm (PQL)

- Iterate…
  1. Let $z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i) + \hat{\eta}_i$, $W_{ii}^{-1} = V(\hat{\mu}_i)g'(\hat{\mu}_i)^2$ and $\eta_i = g(\mu_i)$. V is a known and determined by EF.
  2. Use the prior on $\boldsymbol{\beta}$ and estimate

     $$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-) \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi),$$

     including $\boldsymbol{\lambda}$, as a linear mixed model (this is PQL[2]). $\mathbf{S}_\lambda = \sum \lambda_j \mathbf{S}_j$. $\boldsymbol{\lambda}$ estimation uses REML.
- But $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi)$ is not true!
- Let $\mathbf{QR} = \sqrt{\mathbf{W}}\mathbf{X}$ and $\mathbf{f} = \mathbf{Q}^\mathrm{T}\sqrt{\mathbf{W}}\mathbf{z}$. Assume $\phi = 1$.

[2] Breslow & Clayton, 1993, JASA

Full fitting algorithm (PQL)

- Iterate…
  1. Let $z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i) + \hat{\eta}_i$, $W_{ii}^{-1} = V(\hat{\mu}_i)g'(\hat{\mu}_i)^2$ and $\eta_i = g(\mu_i)$. V is a known and determined by EF.
  2. Use the prior on $\boldsymbol{\beta}$ and estimate
     $$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-) \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi),$$
     including $\boldsymbol{\lambda}$, as a linear mixed model (this is PQL[2]). $\mathbf{S}_\lambda = \sum \lambda_j \mathbf{S}_j$. $\boldsymbol{\lambda}$ estimation uses REML.
- But $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi)$ is not true!
- Let $\mathbf{QR} = \sqrt{\mathbf{W}}\mathbf{X}$ and $\mathbf{f} = \mathbf{Q}^{\mathrm{T}}\sqrt{\mathbf{W}}\mathbf{z}$. Assume $\phi = 1$.
- $n/p \to \infty \Rightarrow \mathbf{f} \sim N(\mathbf{R}\boldsymbol{\beta}, \mathbf{I})$ and appropriate -2 $\times$ REML is

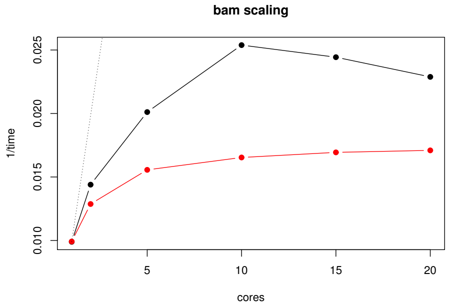
$$\mathcal{V} = \|\mathbf{f} - \mathbf{R}\hat{\boldsymbol{\beta}}_\lambda\|^2 + \hat{\boldsymbol{\beta}}_\lambda^{\mathrm{T}}\mathbf{S}_\lambda\hat{\boldsymbol{\beta}}_\lambda + \log|\mathbf{R}^{\mathrm{T}}\mathbf{R} + \mathbf{S}_\lambda| - \log|\mathbf{S}_\lambda|_+$$

…same as REML in 2. (Relaxing $\phi = 1$ is easy).

[2]Breslow & Clayton, 1993, JASA

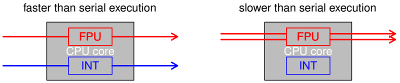
Naive parallelization of GAM fit method

- **Aim**: parallelize preceding method.
- **Hope**: 24 cores turns one day of computing into 1 hour.
- **Reality**: what happened when we parallelized all steps of flop cost $\geq O(p^3)$ in preceding (`mgcv:bam`) method...

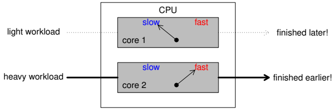


The messy realities of parallel computing

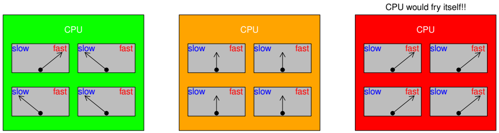
1. Hyper-threading can make parallel slower than serial…



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. Most data-intensive computations are limited by the speed of data retrieval from memory (RAM) **not** by CPU speed. More cores does not help this.

Latency, bandwidth, Cache and block algorithms

- Reading main memory takes $10-20\times$ as long as a flop.
- If data used only once, memory retrieval can't keep up with single core, let alone several.
- Cache is small fast 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.

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

Consequence: Cholesky only methods

- Our main problem is optimizing the REML objective

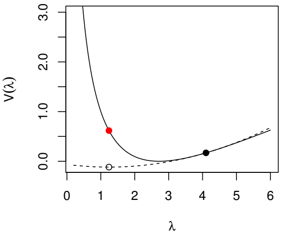
$$\mathcal{V} = \|\mathbf{f} - \mathbf{R}\hat{\boldsymbol{\beta}}_\lambda\|^2 + \hat{\boldsymbol{\beta}}_\lambda^{\mathrm{T}}\mathbf{S}_\lambda\hat{\boldsymbol{\beta}}_\lambda + \log|\mathbf{R}^{\mathrm{T}}\mathbf{R} + \mathbf{S}_\lambda| - \log|\mathbf{S}_\lambda|_+$$

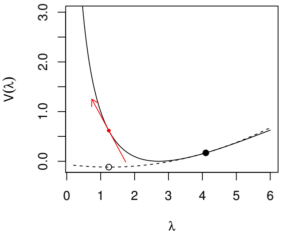
  where $\mathbf{QR} = \sqrt{\mathbf{W}}\mathbf{X}$.
- Actually $\mathbf{R}$ is also the Cholesky factor of $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$, and there exist good parallelizable block Cholesky[3] algorithms.
- But stable computation of $\log|\mathbf{R}^{\mathrm{T}}\mathbf{R} + \mathbf{S}_\lambda|$ requires eigen and QR methods that are not so block oriented and do not parallelize well — we must avoid log('numerical zero').
- Simple idea: optimize $\mathcal{V}$ without evaluating the log determinants: no need for QR and eigen decompositions.

[3]Lucas 2004, LAPACK working paper

Gradient only Newton optimization

- Newton step, Δ, uses only 1st and 2nd derivatives of $\mathcal{V}$.
- Function values only required for step control
  - 'halve Δ if $\mathcal{V}(\lambda + \Delta) > \mathcal{V}(\lambda)$.'
- Instead 'halve Δ if $\Delta^{\mathrm{T}} \nabla \mathcal{V}(\lambda + \Delta) > 0$'.





- This eliminates the 'log(0) problem'. e.g., if $\rho = \log \lambda$,

$$\frac{\partial \log |\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X} + \sum \lambda_j \mathbf{S}_j|}{\partial \rho_j} = \mathrm{tr}\left\{ (\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X} + \sum \lambda_j \mathbf{S}_j)^{-1} \mathbf{S}_j \lambda_j \right\}$$

Simple idea 2

1. Don’t fit the working model at each PQL iteration, just take one Newton step improving its fit.
2. …because you will discard the previous working model at the next PQL step anyway.
3. Step control slightly involved, but provably convergent for some cases (unlike original PQL!).

Simple idea 3: discrete covariate methods

- The methods so far scale like the pivoted Cholesky (rather than QR) and turn months of computing into days-weeks.
- Formation of $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$ is the leading order cost: $O(np^2)$.
- Lang et al.[4] 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.

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

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.

Discrete cross-product example

- Let $\mathbf{X}_A$ and $\mathbf{X}_B$ be model matrices for two different smooths.
- To form $\mathbf{X}_A^{\mathrm{T}}\mathrm{diag}(\mathbf{w})\mathbf{X}_B$
  1. Set $m_A \times m_B$ matrix $\bar{\mathbf{W}}$ to 0.
  2. For $l = 1 \ldots n$ do $\bar{W}[k_A(l), k_B(l)] \mathrel{+}= w(l)$
  3. Form $\bar{\mathbf{X}}_A^{\mathrm{T}}\bar{\mathbf{W}}\bar{\mathbf{X}}_B$.

  ...such terms make up $\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}$, which carries the dominant cost of fitting. Cost reduced by factor up to $O(p_A p_B)$.
- What if $m_A \times m_B \gg n$ (i.e. $\bar{\mathbf{W}}$ too big)? Two options:
  1. $\bar{\mathbf{W}}$ has n non zero elements: sparse representation (via hash table).
  2. Accumulate $\bar{\mathbf{W}}\bar{\mathbf{X}}_B$ or $\bar{\mathbf{X}}_A^{\mathrm{T}}\bar{\mathbf{W}}$ directly.

Discrete method complications

- Model matrices for tensor product smooth interactions are made up of row Kronecker products of *marginal* model matrices from marginal univariate smooths.
- To maintain accuracy we discretize each margin: *not jointly*.
- Algorithms must deal with row Kronecker structure on the fly.
- Identifiability constraints then have to be applied on the fly too.
- …also terms may involve linear functionals of smooths…
- A library of such algorithms needed + Cache friendly implementation.

Black smoke modelling

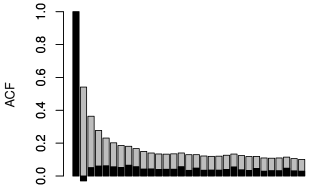
- Method based on parallel Cholesky, determinant free iteration and discretization of covariates implemented in `mgcv` function `bam(...,discrete=TRUE)`.
- The current 'best' daily black smoke model is

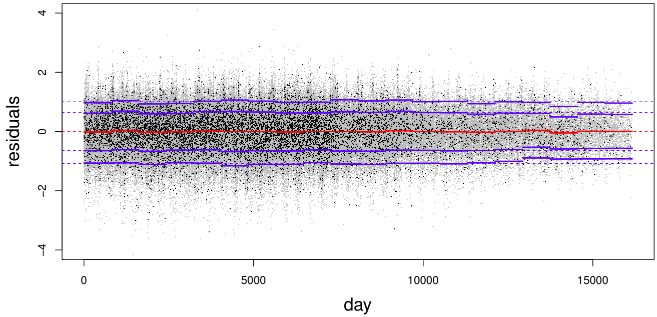
$$\begin{aligned}\log(\texttt{bs}_i) = f_1(\texttt{y}_i) + f_2(\texttt{doy}_i) + f_3(\texttt{dow}_i) \\ + f_4(\texttt{y}_i, \texttt{doy}_i) + f_5(\texttt{y}_i, \texttt{dow}_i) + f_6(\texttt{doy}_i, \texttt{dow}_i) \\ + f_7(\texttt{n}_i, \texttt{e}_i) + f_8(\texttt{n}_i, \texttt{e}_i, \texttt{y}_i) +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

  The model has around 10^4 coefficients and $r^2 = 0.79$.
- With the new parallel discretised methods fit time is around 5 minutes. We estimate that previous methods would have required > 1 month. Memory footprint is about 15Gb.

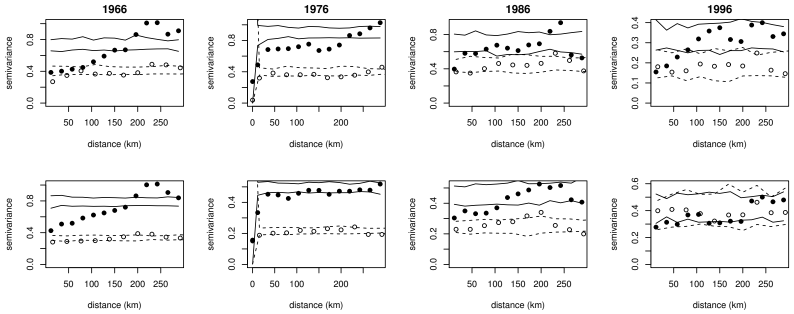
Daily black smoke model - 10 day timestep

BS residuals in time



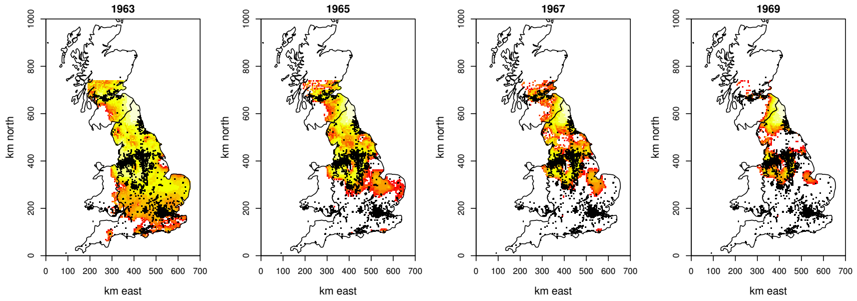


BS variograms in space



- Black solid is raw data, open is model residuals.
- Lines are permutation based reference intervals.
- Top row, day 40. Bottom row, day 200.

Derived maps — Posterior exceedance probabilities



- Map shows average daily probability of exceeding current EU daily limit, for 4 years in the 1960s.
- Notice how the switch from coal to gas for domestic heating cleans up the London area quite quickly.
- The northern industrial city areas take longer.

Conclusions

- Possible to obtain three orders of magnitude computational speed-up in large additive model fitting by combining
  1. A new iterative algorithm for REML based additive model estimation. Requires only Cholesky decomposition, and avoids evaluation of unstable log determinants.
  2. OpenMP parallelization of modern pivoted block Cholesky.
  3. New methods to efficiently compute all the model matrix products required in fitting, based on discretization of covariates (including tensor product smooths).
- The methods facilitate the first *daily* models of UK black smoke densities based on the multi-decade daily data from the UK black smoke monitoring network.
- See Wood, Li, Shaddick & Augustin (2017) JASA, and Li & Wood (2019) Statistics and Computing.

Aside: penalty choice matters - a bad model movie