Practical Computing for Spatial Data Models

Andrew Finley

Department of Forestry, Michigan State University, Lansing, Michigan

September 19, 2019

Course objectives

In this short amount of time, I hope to provide:

1. simple changes to your current computing environment that yield big computation gains
2. thoughts on selecting computing environments and software to alleviate common bottlenecks
3. an applied glimpse under the hood at some lower-level code (`C/C++` and `FORTRAN`) that can improve your higher-level code (e.g., `R`)
4. introduction to some lower- and higher-level coding tools and tips for parallelization

Topics are generally motivated using geostatistical models applied to settings where we have a lot of data.

Modeling Univariate Spatial Data

September 18, 2019

- Researchers in diverse areas such as ecology, forestry, climatology, and environmental health, are increasingly faced with the task of analyzing data that are:
  - highly multivariate, with many important predictors and response variables,
  - geographically referenced, and often presented as maps, and
  - temporally correlated, as in longitudinal or other time series structures.

$\Rightarrow$ motivates hierarchical modeling and data analysis for complex spatial (and spatiotemporal) data sets.

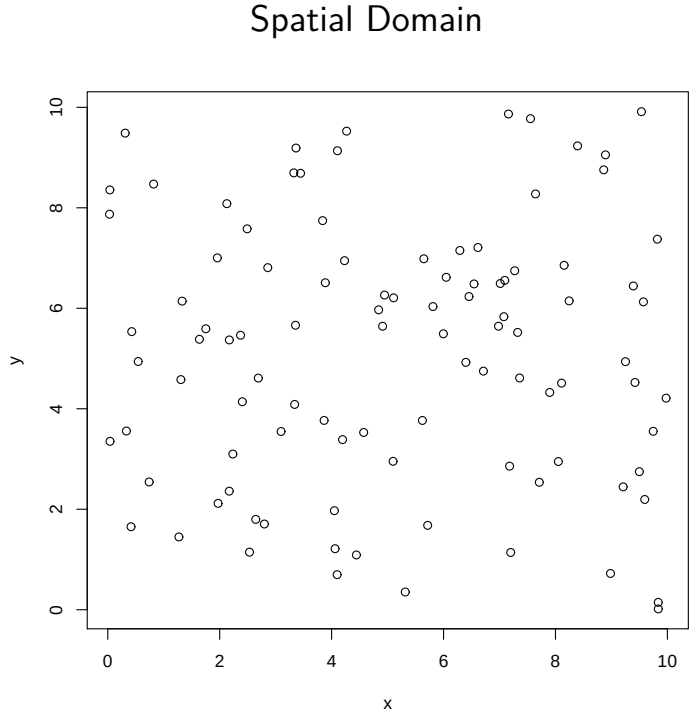
Scaling Problems in Statistics 2019

- point-referenced data, where $y(\mathbf{s})$ is a random vector at a location $\mathbf{s} \in \Re^r$, where $\mathbf{s}$ varies continuously over D, a fixed subset of $\Re^r$ that contains an r-dimensional rectangle of positive volume;

- point-referenced data, where $y(\mathbf{s})$ is a random vector at a location $\mathbf{s} \in \Re^r$, where $\mathbf{s}$ varies continuously over D, a fixed subset of $\Re^r$ that contains an r-dimensional rectangle of positive volume;

- areal data, where D is again a fixed subset (of regular or irregular shape), but now partitioned into a finite number of areal units with well-defined boundaries;

- point-referenced data, where $y(\mathbf{s})$ is a random vector at a location $\mathbf{s} \in \Re^r$, where $\mathbf{s}$ varies continuously over D, a fixed subset of $\Re^r$ that contains an r-dimensional rectangle of positive volume;
- areal data, where D is again a fixed subset (of regular or irregular shape), but now partitioned into a finite number of areal units with well-defined boundaries;
- point pattern data, where now D is itself random; its index set gives the locations of random events that are the spatial point pattern. $y(\mathbf{s})$ itself can simply equal 1 for all $\mathbf{s} \in D$ (indicating occurrence of the event), or possibly give some additional covariate information (producing a marked point pattern process).



Scaling Problems in Statistics 2019

Algorithmic Modeling

- Spatial surface observed at finite set of locations $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n\}$
- Tessellate the spatial domain (usually with data locations as vertices)
- Fit an interpolating polynomial:

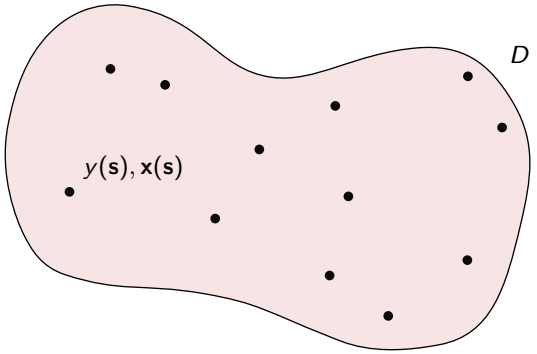
$$f(\mathbf{s}) = \sum_i w_i(\mathcal{S}; \mathbf{s}) f(\mathbf{s}_i)$$

- "Interpolate" by reading off $f(\mathbf{s}_0)$.
- Includes: triangulation, weighted averages, geographically weighted regression (GWR)
- Issues:
  - Sensitivity to tessellations
  - Choices of multivariate interpolators
  - Numerical error analysis

Simple linear model

$$y(\mathbf{s}) = \mu(\mathbf{s}) + \epsilon(\mathbf{s}),$$

- Response: $y(\mathbf{s})$ at location $\mathbf{s}$
- Mean: $\mu = \mathbf{x}(\mathbf{s})^\top \beta$
- Error: $\epsilon(\mathbf{s}) \overset{iid}{\sim} N(0, \tau^2)$

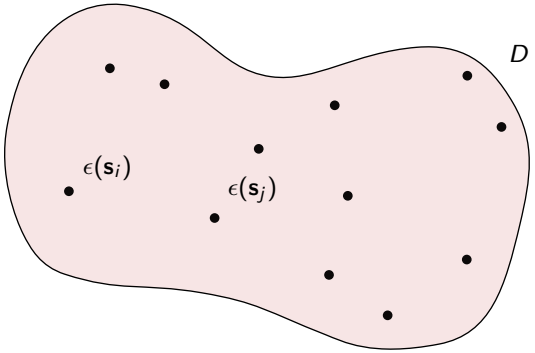


Simple linear model

$$y(\mathbf{s}) = \mu(\mathbf{s}) + \epsilon(\mathbf{s}),$$

Assumptions regarding $\boldsymbol{\epsilon}(\mathbf{s})$:

- $\boldsymbol{\epsilon}(\mathbf{s}) \overset{iid}{\sim} N(0, \tau^2)$

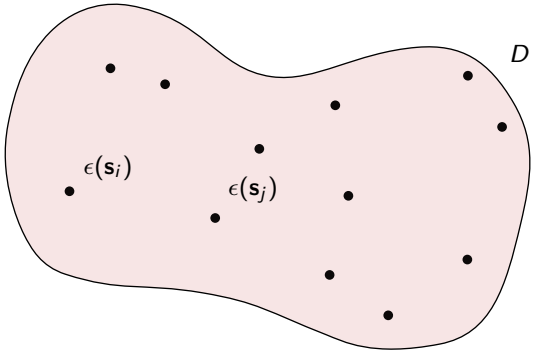


Simple linear model

$$y(\mathbf{s}) = \mu(\mathbf{s}) + \epsilon(\mathbf{s}),$$

Assumptions regarding $\boldsymbol{\epsilon}(\mathbf{s})$:

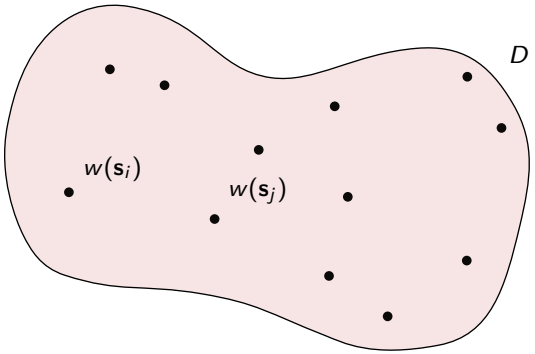
- $\boldsymbol{\epsilon}(\mathbf{s}) \overset{iid}{\sim} N(0, \tau^2)$
- $\epsilon(\mathbf{s}_i)$ and $\epsilon(\mathbf{s}_j)$ are uncorrelated for all $i \neq j$



Spatial Gaussian processes (GP):

- Say $w(\mathbf{s}) \sim GP(0, \sigma^2\rho(\cdot))$ and

$$Cov(w(\mathbf{s}_1), w(\mathbf{s}_2)) = \sigma^2\rho(\phi; \|\mathbf{s}_1 - \mathbf{s}_2\|)$$



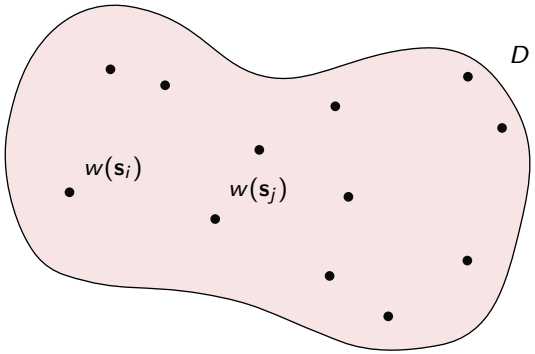
Spatial Gaussian processes (GP):

- Say $w(\mathbf{s}) \sim GP(0, \sigma^2\rho(\cdot))$ and

$$Cov(w(\mathbf{s}_1), w(\mathbf{s}_2)) = \sigma^2\rho(\phi; \|\mathbf{s}_1 - \mathbf{s}_2\|)$$

- Let $\mathbf{w} = [w(\mathbf{s}_i)]_{i=1}^n$, then

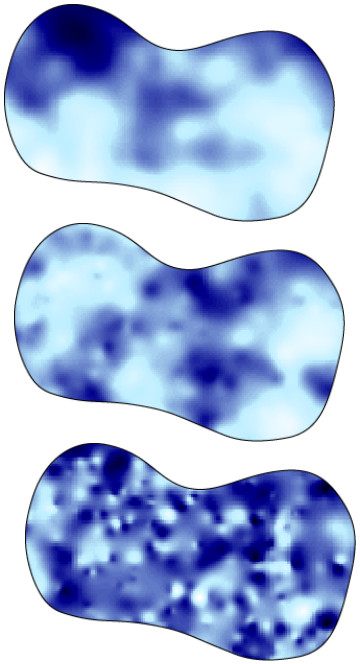
$$\mathbf{w} \sim N(\mathbf{0}, \sigma^2\mathbf{R}(\phi)), \text{ where } \mathbf{R}(\phi) = [\rho(\phi; \|\mathbf{s}_i - \mathbf{s}_j\|)]_{i,j=1}^n$$



Realization of a Gaussian process:

- Changing ϕ and holding $\sigma^2 = 1$:

$$\mathbf{w} \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi)), \text{ where}$$
$$\mathbf{R}(\phi) = [\rho(\phi; \|\mathbf{s}_i - \mathbf{s}_j\|)]_{i,j=1}^n$$



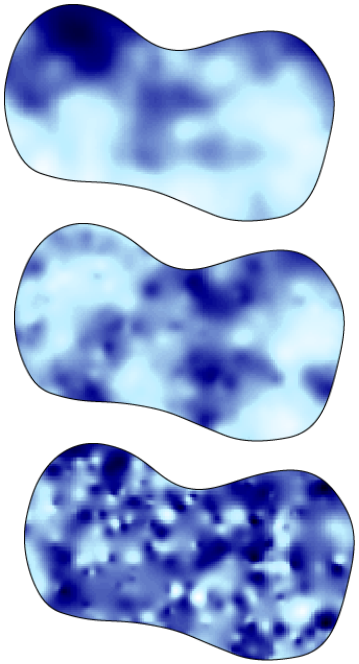
Realization of a Gaussian process:

- Changing ϕ and holding $\sigma^2 = 1$:

$$\mathbf{w} \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi)), \text{ where}$$
$$\mathbf{R}(\phi) = [\rho(\phi; \|\mathbf{s}_i - \mathbf{s}_j\|)]_{i,j=1}^n$$

- Correlation model for $R(\phi)$:
  e.g., exponential decay

$$\rho(\phi; t) = \exp(-\phi t) \text{ if } t > 0.$$



Scaling Problems in Statistics 2019

Realization of a Gaussian process:

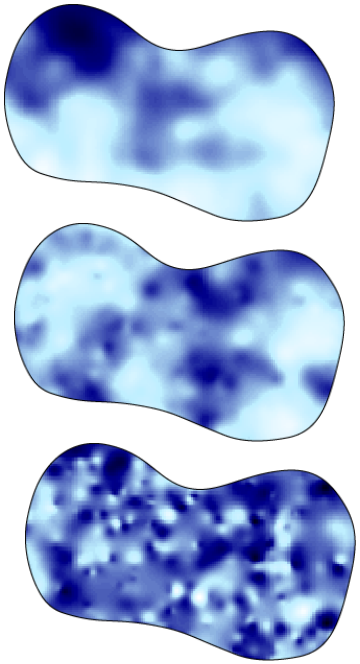
- Changing ϕ and holding $\sigma^2 = 1$:

$$\mathbf{w} \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi)), \text{ where}$$
$$\mathbf{R}(\phi) = [\rho(\phi; \|\mathbf{s}_i - \mathbf{s}_j\|)]_{i,j=1}^n$$

- Correlation model for $R(\phi)$: e.g., exponential decay

$$\rho(\phi; t) = \exp(-\phi t) \text{ if } t > 0.$$

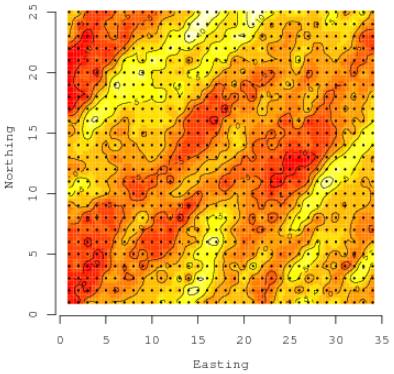
- Other valid models e.g., Gaussian, Spherical, Matérn.
- Effective range, $t_0 = -ln(0.05)/\phi \approx 3/\phi$



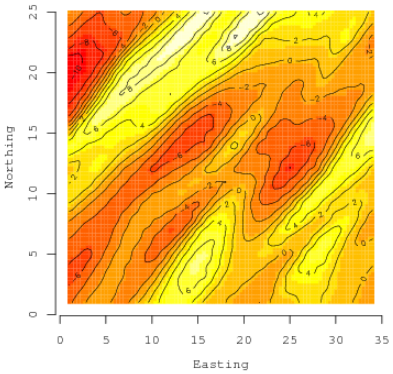
$\mathbf{w} \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi))$ provides complex spatial dependence through simple structured dependence.

E.g., anisotropic Matérn correlation function:
$\rho(\mathbf{s}_i, \mathbf{s}_j; \phi) = \left(1/\Gamma(\nu) 2^{\nu-1}\right) \left(2\sqrt{\nu d_{ij}}\right)^{\nu} \kappa_\nu (2\sqrt{\nu d_{ij}})$, where
$d_{ij} = (\mathbf{s}_i - \mathbf{s}_j)' \Sigma^{-1} (\mathbf{s}_i - \mathbf{s}_j)$, $\Sigma = G(\psi)\Lambda^2 G(\psi)'$. Thus, $\phi = (\nu, \psi, \Lambda)$.

Simulated



Predicted



Scaling Problems in Statistics 2019

Simple linear model + random spatial effects

$$y(\mathbf{s}) = \mu(\mathbf{s}) + w(\mathbf{s}) + \epsilon(\mathbf{s}),$$

- Response: $y(\mathbf{s})$ at some site
- Mean: $\mu = \mathbf{x}(\mathbf{s})^\top \beta$
- Spatial random effects: $w(\mathbf{s}) \sim GP(0, \sigma^2 \rho(\phi; \|\mathbf{s}_1 - \mathbf{s}_2\|))$
- Non-spatial variance: $\epsilon(\mathbf{s}) \overset{iid}{\sim} N(0, \tau^2)$. Interpretation as pure error, measurement error, replication error, microscale error.

Hierarchical modeling

- First stage:

$$\mathbf{y}|\beta, \mathbf{w}, \tau^2 \sim \prod_{i=1}^{n} N(y(\mathbf{s}_i) \,|\, \mathbf{x}(\mathbf{s}_i)^\top \beta + w(\mathbf{s}_i), \tau^2)$$

Hierarchical modeling

- First stage:

$$\mathbf{y}|\boldsymbol{\beta}, \mathbf{w}, \tau^2 \sim \prod_{i=1}^{n} N(y(\mathbf{s}_i) \,|\, \mathbf{x}(\mathbf{s}_i)^\top \boldsymbol{\beta} + w(\mathbf{s}_i), \tau^2)$$

- Second stage:

$$\mathbf{w}|\sigma^2, \phi \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi))$$

Hierarchical modeling

- First stage:

$$\mathbf{y}|\boldsymbol{\beta}, \mathbf{w}, \tau^2 \sim \prod_{i=1}^{n} N(y(\mathbf{s}_i) \,|\, \mathbf{x}(\mathbf{s}_i)^\top \boldsymbol{\beta} + w(\mathbf{s}_i), \tau^2)$$

- Second stage:

$$\mathbf{w}|\sigma^2, \phi \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi))$$

- Third stage: Priors on $\Omega = (\boldsymbol{\beta}, \tau^2, \sigma^2, \phi)$

Hierarchical modeling

- First stage:

$$\mathbf{y}|\beta, \mathbf{w}, \tau^2 \sim \prod_{i=1}^{n} N(y(\mathbf{s}_i) \,|\, \mathbf{x}(\mathbf{s}_i)^\top \beta + w(\mathbf{s}_i), \tau^2)$$

- Second stage:

$$\mathbf{w}|\sigma^2, \phi \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi))$$

- Third stage: Priors on $\Omega = (\beta, \tau^2, \sigma^2, \phi)$
- Collapsed likelihood:

$$\mathbf{y}|\Omega \sim N(\mathbf{X}\beta, \sigma^2 \mathbf{R}(\phi) + \tau^2 \mathbf{I})$$

Hierarchical modeling

- First stage:

$$\mathbf{y}|\beta, \mathbf{w}, \tau^2 \sim \prod_{i=1}^{n} N(y(\mathbf{s}_i) \,|\, \mathbf{x}(\mathbf{s}_i)^\top \beta + w(\mathbf{s}_i), \tau^2)$$

- Second stage:

$$\mathbf{w}|\sigma^2, \phi \sim N(\mathbf{0}, \sigma^2 \mathbf{R}(\phi))$$

- Third stage: Priors on $\Omega = (\beta, \tau^2, \sigma^2, \phi)$
- Collapsed likelihood:

$$\mathbf{y}|\Omega \sim N(\mathbf{X}\beta, \sigma^2 \mathbf{R}(\phi) + \tau^2 \mathbf{I})$$

- Note: Spatial process parametrizes $\mathbf{\Sigma}$:
  $\mathbf{y} = \mathbf{X}\beta + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \Sigma)$, $\mathbf{\Sigma} = \sigma^2 \mathbf{R}(\phi) + \tau^2 I$

Bayesian Computations

- Choice: Fit $[\mathbf{y}|\Omega] \times [\Omega]$ or $[\mathbf{y}|\boldsymbol{\beta}, \mathbf{w}, \tau^2] \times [\mathbf{w}|\sigma^2, \phi] \times [\Omega]$.

Bayesian Computations

- Choice: Fit $[\mathbf{y}|\Omega] \times [\Omega]$ or $[\mathbf{y}|\boldsymbol{\beta}, \mathbf{w}, \tau^2] \times [\mathbf{w}|\sigma^2, \phi] \times [\Omega]$.
- Conditional model:
  - conjugate full conditionals for $\boldsymbol{\beta}$, σ^2, τ^2 and $\mathbf{w}$ – easier to program.

Bayesian Computations

- Choice: Fit $[\mathbf{y}|\Omega] \times [\Omega]$ or $[\mathbf{y}|\boldsymbol{\beta}, \mathbf{w}, \tau^2] \times [\mathbf{w}|\sigma^2, \phi] \times [\Omega]$.
- Conditional model:
  - conjugate full conditionals for $\boldsymbol{\beta}$, σ^2, τ^2 and $\mathbf{w}$ – easier to program.
- Marginalized model:
  - need Metropolis or Slice sampling for σ^2, τ^2 and ϕ. Harder to program.
  - But, reduced parameter space $\Rightarrow$ faster convergence
  - $\sigma^2\mathbf{R}(\phi) + \tau^2\mathbf{I}$ is more stable than $\sigma^2\mathbf{R}(\phi)$.

Bayesian Computations

- Choice: Fit $[\mathbf{y}|\Omega] \times [\Omega]$ or $[\mathbf{y}|\boldsymbol{\beta}, \mathbf{w}, \tau^2] \times [\mathbf{w}|\sigma^2, \phi] \times [\Omega]$.
- Conditional model:
  - conjugate full conditionals for $\boldsymbol{\beta}$, σ^2, τ^2 and $\mathbf{w}$ – easier to program.
- Marginalized model:
  - need Metropolis or Slice sampling for σ^2, τ^2 and ϕ. Harder to program.
  - But, reduced parameter space $\Rightarrow$ faster convergence
  - $\sigma^2\mathbf{R}(\phi) + \tau^2\mathbf{I}$ is more stable than $\sigma^2\mathbf{R}(\phi)$.
- But what about $\mathbf{R}^{-1}(\phi)$?? EXPENSIVE!

Where are the **w**'s?

- Interest often lies in the spatial surface $\mathbf{w}|\mathbf{y}$.

Where are the **w**'s?

- Interest often lies in the spatial surface $\mathbf{w}|\mathbf{y}$.
- They are recovered from

$$[\mathbf{w}|\mathbf{y}, X] = \int [\mathbf{w}|\Omega, \mathbf{y}, X] \times [\Omega|\mathbf{y}, X] d\Omega$$

  using posterior samples:
  - Obtain $\Omega^{(1)}, \ldots, \Omega^{(G)} \sim [\Omega|\mathbf{y}, \mathbf{X}]$
  - For each $\Omega^{(g)}$, draw $\mathbf{w}^{(g)} \sim [\mathbf{w}|\Omega^{(g)}, \mathbf{y}, \mathbf{X}]$
- NOTE: With Gaussian likelihoods $[\mathbf{w}|\Omega, \mathbf{y}, \mathbf{X}]$ is also Gaussian. With other likelihoods this may not be a standard distribution; conditional updating scheme is preferred.

- Often we need to predict $y(\mathbf{s})$ at a *new* set of locations $\{\tilde{\mathbf{s}}_0, \ldots, \tilde{\mathbf{s}}_{\tilde{n}}\}$ with associated predictor matrix $\tilde{\mathbf{X}}$.

- Sample from predictive distribution:

$$\begin{aligned}[\tilde{\mathbf{y}}|\mathbf{y}, \mathbf{X}, \tilde{\mathbf{X}}] &= \int [\tilde{\mathbf{y}}, \Omega|\mathbf{y}, \mathbf{X}, \tilde{\mathbf{X}}] d\Omega \\ &= \int [\tilde{\mathbf{y}}|\mathbf{y}, \Omega, \mathbf{X}, \tilde{\mathbf{X}}] \times [\Omega|\mathbf{y}, \mathbf{X}] d\Omega,\end{aligned}$$

  $[\tilde{\mathbf{y}}|\mathbf{y}, \Omega, \mathbf{X}, \tilde{\mathbf{X}}]$ is multivariate normal. Sampling scheme:
  - Obtain $\Omega^{(1)}, \ldots, \Omega^{(G)} \sim [\Omega|\mathbf{y}, \mathbf{X}]$
  - For each $\Omega^{(g)}$, draw $\tilde{\mathbf{y}}^{(g)} \sim [\tilde{\mathbf{y}}|\mathbf{y}, \Omega^{(g)}, \mathbf{X}, \tilde{\mathbf{X}}]$.

Calling C/C++ from R and Fun with OpenMP

September 19, 2019

Our focus today

What we're not doing

- developing an R package (see, e.g., official documentation Writing R Extensions and Hadly Wickham R packages)
- learning C/C++, FORTRAN, R, or OpenMP
- taking a deep look at any one topic

What we are doing

- mentioning some topics that might get you thinking about how to improve your code
- scratching the surface of some expansive topics in computing
- providing some code and ideas that might point you in the right direction

Why connect R and lower-level code

R is an interpreted language and, as a result, can be slow at

- vectorizing loops whose subsequent iterations depend on previous iterations
- executing recursive functions

Also, we often want to use data structures, algorithms, and libraries written in lower-level code (e.g., BLAS, LAPACK, CHOLMOD, Eigen, GNU Scientific Library, etc.).

We could just write standalone code, but

- R is nice for input/output and other tasks in-between
- we might want to use some of R's C functions, e.g., RNGs and distributions in `Rmath.h`
- we might eventually write an R package
- simplifies in-house code sharing and teaching

R Foreign Language Interfaces

The authoritative document is "Writing R Extensions" found at https://cran.r-project.org.

We'll focus on calling C/C++ for now (but calling FORTRAN and JAVA is similar). There are three approaches for passing stuff between R and C/C++.

1. `.Call()` designed for calling code that understands R objects and environments. Allows multiple arguments to be passed to C/C++ and R objects returned.
2. `External()` like `.Call()` but the C/C++ function is passed a single argument containing a LISTSXP, a pairlist from which the arguments can be extracted.
3. `.C()` (and `.Fortran`) designed to call code that does not know about R. Straightforward, but limited types of arguments and all checking of arguments must be done in R. No return value, but may alter its arguments.

Why not Rcpp?

There are some real advantages to using Rcpp

- Rcpp API (Application Programming Interface) "protects you from many of the historical idiosyncrasies of the R API"–Hadley Wickham
- takes care of memory management
- provides helper methods to working with R objects in C++
- many more advantages, see, e.g., http://www.rcpp.org/

Sounds good, so why use R's API?

- preference to write standalone flexible C/C++ code, then with slight modification can be called from R
- one fewer level of abstraction to deal with (and perhaps some overhead)—feels closer to the metal
- it's what I know well and how most R packages with source code are written

Moving between R and C/C++ types

Mapping between the modes of R atomic vectors and the types of arguments to a C function or FORTRAN subroutine.

R storage mode	C type	FORTRAN type
logical	int *	INTEGER
integer	int *	INTEGER
double	double *	DOUBLE PRECISION
complex	Rcomplex *	DOUBLE COMPLEX
character	char **	CHARACTER*255
raw	unsigned char *	none

Getting started

Calling a C/C++ function from R requires two pieces: a C/C++ function and an R wrapper function that uses `.Call()`.

Compile the C/C++ code and call from R as a shared object `.so` (Linux or MacOS X) or as a `.dll` (Windows).

From within R load the compiled object using `dyn.load()` and unload it using `dyn.unload()`.

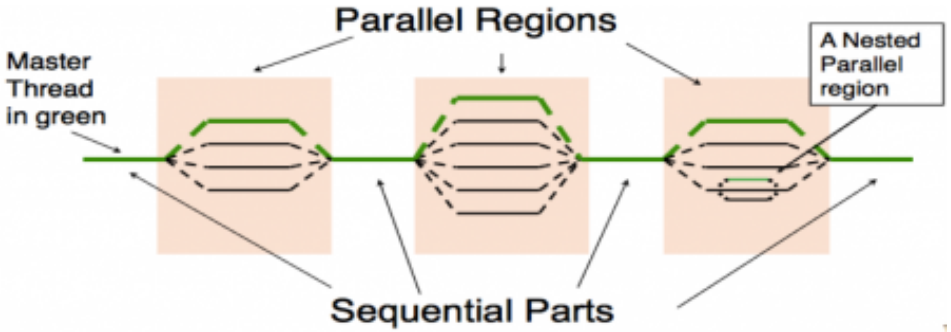
Exercise #2

Let's turn to Exercise 2 `cIDist.R` and `cIDist.cpp` files.

Parallel computing with `OpenMP`

`OpenMP` is an industry standard API of C/C++ and FORTRAN for shared memory parallel programming.

`OpenMP` is based on two concepts: the use of threads (think CPUs) and the fork/join model of parallelism:



All threads have access to the same shared global memory. Each thread has access to its private variables and common variables[1].

[1] figure credit: www.nersc.gov

Parallel computing with `OpenMP`

Some advantages

- high-level directives (`pragma`) used to define parallel regions simplify coding and decisions
- parallelism can be added incrementally
- programs can be run sequentially if needed (e.g., if compilers do not support `OpenMP`)
- compilers (or you) can optimize the number of threads needed by parallel region

"With great power comes great responsibility"—Benjamin Parker (a.k.a Uncle Ben)

You must be sure that what you are doing in parallel regions is **thread-safe**—it's very easy to make mistakes that compile without error.

Exercise #2 revisited

Consider Exercise 2 but now `cIDistOMP.R` and `cIDistOMP.cpp` files.

Code gone wrong (then right)

Let's construct a spatial correlation matrix using the Matern function such that the i, j-th element is equal to

$$R(\theta)_{i,j} = \frac{1}{2^{\nu-1}\Gamma(\nu)}(d_{i,j}\phi)^{\nu}\mathcal{K}_{\nu}(d_{i,j}\phi); \phi > 0, \nu\, 0, \qquad (1)$$

where $\theta = (\phi, \nu)$ with ϕ controlling the decay and ν controlling smoothness, Γ is the Gamma function, and $\mathcal{K}_{\nu}$ is a modified Bessel function of the second kind with order ν.

In `R` speak this is

```
(D*phi)^nu/(2^(nu-1)*gamma(nu))*besselK(x=D*phi, nu=nu)
```

and C using `Rmath.h` functions

```
pow(D[i]*phi, nu)/(pow(2, nu-1)*gammafn(nu)*
                   bessel_k(D[i]*phi, nu, 1.0)
```


Exercise #3

Consider Exercise 3 but now `cRMaternOMPWrong.cpp` and `cRMaternOMPWrong.cpp` files.

Now making it thread-safe

The problem is that `R`'s `bessel_k` C function is not thread-safe. Note the `bk` vector is allocated within `bessel_k.c`.

Instead use undocumented `bessel_k_ex` in `bessel_k.c` as illustrated in `cRMaternOMPSafe.cpp`.

Here we:

- allocate enough working space for each thread outside the parallel region
- use each thread's id (i.e., $0, 1, \ldots,$ nTheads $- 1$) via `omp_get_thread_num()` to index the working space passed to `bessel_k_ex`

Efficient Implementation of Bayesian Hierarchical Linear Models

September 18, 2019

Bayesian hierarchical linear mixed model

$p(\boldsymbol{\theta}) \times N(\beta \,|\, \boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta) \times N(\boldsymbol{\alpha} \,|\, \mathbf{0}, \mathbf{K}(\boldsymbol{\theta})) \times N(\mathbf{y} \,|\, \mathbf{X}\beta + \mathbf{Z}(\boldsymbol{\theta})\boldsymbol{\alpha}, \mathbf{D}(\boldsymbol{\theta}))$

- $\mathbf{y}$ is an $n \times 1$ vector of possibly irregularly located observations,
- $\mathbf{X}$ is a known $n \times p$ matrix of regressors ($p < n$),
- $\mathbf{K}(\boldsymbol{\theta})$ and $\mathbf{D}(\boldsymbol{\theta})$ are families of $r \times r$ and $n \times n$ covariance matrices, respectively,
- $\mathbf{Z}(\boldsymbol{\theta})$ is $n \times r$ with $r \leq n$, all indexed by a set of unknown process parameters $\boldsymbol{\theta}$.
- $\boldsymbol{\alpha}$ is the $r \times 1$ random vector and β is the $p \times 1$ slope vector.

Bayesian hierarchical linear mixed model

$p(\boldsymbol{\theta}) \times N(\beta \,|\, \boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta) \times N(\boldsymbol{\alpha} \,|\, \mathbf{0}, \mathbf{K}(\boldsymbol{\theta})) \times N(\mathbf{y} \,|\, \mathbf{X}\beta + \mathbf{Z}(\boldsymbol{\theta})\boldsymbol{\alpha}, \mathbf{D}(\boldsymbol{\theta}))$

- $\mathbf{y}$ is an $n \times 1$ vector of possibly irregularly located observations,
- $\mathbf{X}$ is a known $n \times p$ matrix of regressors ($p < n$),
- $\mathbf{K}(\boldsymbol{\theta})$ and $\mathbf{D}(\boldsymbol{\theta})$ are families of $r \times r$ and $n \times n$ covariance matrices, respectively,
- $\mathbf{Z}(\boldsymbol{\theta})$ is $n \times r$ with $r \leq n$, all indexed by a set of unknown process parameters $\boldsymbol{\theta}$.
- $\boldsymbol{\alpha}$ is the $r \times 1$ random vector and β is the $p \times 1$ slope vector.

Space-varying intercept model is a special case where $\mathbf{D}(\boldsymbol{\theta}) = \tau^2 I_n$, $\boldsymbol{\alpha} = (w(\mathbf{s}_1), w(\mathbf{s}_2), \ldots, w(\mathbf{s}_n))^\top$, $\mathbf{Z}(\boldsymbol{\theta}) = I_n$, and the $n \times n$ $\mathbf{K}(\boldsymbol{\theta}) = \sigma^2 \mathbf{R}(\phi)$.

For faster convergence, we integrate out $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ from the model and first sample from

$$p(\boldsymbol{\theta} \,|\, \mathbf{y}) \propto p(\boldsymbol{\theta}) \times N(\mathbf{y} \,|\, \mathbf{X}\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_{y \,|\, \theta}),$$

where $\boldsymbol{\Sigma}_{y \,|\, \theta} = \mathbf{X}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top + \mathbf{Z}(\boldsymbol{\theta})\mathbf{K}(\boldsymbol{\theta})\mathbf{Z}(\boldsymbol{\theta})^\top + \mathbf{D}(\boldsymbol{\theta})$.

This involves evaluating

$$\log p(\boldsymbol{\theta} \,|\, \mathbf{y}) = \text{const} + \log p(\boldsymbol{\theta}) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{y \,|\, \theta}| - \frac{1}{2}Q(\boldsymbol{\theta}) \,,$$

where $Q(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)^\top \boldsymbol{\Sigma}_{y \,|\, \theta}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$.

For faster convergence, we integrate out $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ from the model and first sample from

$$p(\boldsymbol{\theta} \,|\, \mathbf{y}) \propto p(\boldsymbol{\theta}) \times N(\mathbf{y} \,|\, \mathbf{X}\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_{y \,|\, \theta}),$$

where $\boldsymbol{\Sigma}_{y \,|\, \theta} = \mathbf{X}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top + \mathbf{Z}(\boldsymbol{\theta})\mathbf{K}(\boldsymbol{\theta})\mathbf{Z}(\boldsymbol{\theta})^\top + \mathbf{D}(\boldsymbol{\theta})$.

This involves evaluating

$$\log p(\boldsymbol{\theta} \,|\, \mathbf{y}) = \text{const} + \log p(\boldsymbol{\theta}) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{y \,|\, \theta}| - \frac{1}{2}Q(\boldsymbol{\theta}) \,,$$

where $Q(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)^\top \boldsymbol{\Sigma}_{y \,|\, \theta}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$.

1. $\mathbf{L} = \texttt{chol}(\boldsymbol{\Sigma}_{y \,|\, \theta})$, lower-triangular Cholesky factor $\mathbf{L}$ of $\boldsymbol{\Sigma}_{y \,|\, \boldsymbol{\theta}}$ ($O(n^3/3)$ flops)

For faster convergence, we integrate out $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ from the model and first sample from

$$p(\boldsymbol{\theta} \,|\, \mathbf{y}) \propto p(\boldsymbol{\theta}) \times N(\mathbf{y} \,|\, \mathbf{X}\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_{y \,|\, \theta}),$$

where $\boldsymbol{\Sigma}_{y \,|\, \theta} = \mathbf{X}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top + \mathbf{Z}(\boldsymbol{\theta})\mathbf{K}(\boldsymbol{\theta})\mathbf{Z}(\boldsymbol{\theta})^\top + \mathbf{D}(\boldsymbol{\theta})$.

This involves evaluating

$$\log p(\boldsymbol{\theta} \,|\, \mathbf{y}) = \text{const} + \log p(\boldsymbol{\theta}) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{y \,|\, \theta}| - \frac{1}{2}Q(\boldsymbol{\theta}) \,,$$

where $Q(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)^\top \boldsymbol{\Sigma}_{y \,|\, \theta}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$.

1. $\mathbf{L} = \texttt{chol}(\boldsymbol{\Sigma}_{y \,|\, \theta})$, lower-triangular Cholesky factor $\mathbf{L}$ of $\boldsymbol{\Sigma}_{y \,|\, \boldsymbol{\theta}}$ ($O(n^3/3)$ flops)
2. $\mathbf{u} = \texttt{trsolve}(\mathbf{L}, \mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$, solves $\mathbf{L}\mathbf{u} = \mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta$ ($O(n^2)$ flops)

For faster convergence, we integrate out $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ from the model and first sample from

$$p(\boldsymbol{\theta} \,|\, \mathbf{y}) \propto p(\boldsymbol{\theta}) \times N(\mathbf{y} \,|\, \mathbf{X}\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_{y \,|\, \theta}),$$

where $\boldsymbol{\Sigma}_{y \,|\, \theta} = \mathbf{X}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top + \mathbf{Z}(\boldsymbol{\theta})\mathbf{K}(\boldsymbol{\theta})\mathbf{Z}(\boldsymbol{\theta})^\top + \mathbf{D}(\boldsymbol{\theta})$.

This involves evaluating

$$\log p(\boldsymbol{\theta} \,|\, \mathbf{y}) = \text{const} + \log p(\boldsymbol{\theta}) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{y \,|\, \theta}| - \frac{1}{2}Q(\boldsymbol{\theta}) \,,$$

where $Q(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)^\top \boldsymbol{\Sigma}_{y \,|\, \theta}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$.

1. $\mathbf{L} = \texttt{chol}(\boldsymbol{\Sigma}_{y \,|\, \theta})$, lower-triangular Cholesky factor $\mathbf{L}$ of $\boldsymbol{\Sigma}_{y \,|\, \boldsymbol{\theta}}$ ($O(n^3/3)$ flops)
2. $\mathbf{u} = \texttt{trsolve}(\mathbf{L}, \mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$, solves $\mathbf{L}\mathbf{u} = \mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta$ ($O(n^2)$ flops)
3. $Q(\boldsymbol{\theta}) = \mathbf{u}^\top\mathbf{u}$ ($2n$ flops)

For faster convergence, we integrate out $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ from the model and first sample from

$$p(\boldsymbol{\theta} \,|\, \mathbf{y}) \propto p(\boldsymbol{\theta}) \times N(\mathbf{y} \,|\, \mathbf{X}\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_{y\,|\,\theta}),$$

where $\boldsymbol{\Sigma}_{y\,|\,\theta} = \mathbf{X}\boldsymbol{\Sigma}_\beta\mathbf{X}^\top + \mathbf{Z}(\boldsymbol{\theta})\mathbf{K}(\boldsymbol{\theta})\mathbf{Z}(\boldsymbol{\theta})^\top + \mathbf{D}(\boldsymbol{\theta})$.

This involves evaluating

$$\log p(\boldsymbol{\theta} \,|\, \mathbf{y}) = \text{const} + \log p(\boldsymbol{\theta}) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{y\,|\,\theta}| - \frac{1}{2}Q(\boldsymbol{\theta}) \,,$$

where $Q(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)^\top \boldsymbol{\Sigma}_{y\,|\,\theta}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$.

1. $\mathbf{L} = \texttt{chol}(\boldsymbol{\Sigma}_{y\,|\,\theta})$, lower-triangular Cholesky factor $\mathbf{L}$ of $\boldsymbol{\Sigma}_{y\,|\,\boldsymbol{\theta}}$ ($O(n^3/3)$ flops)
2. $\mathbf{u} = \texttt{trsolve}(\mathbf{L}, \mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta)$, solves $\mathbf{L}\mathbf{u} = \mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\beta$ ($O(n^2)$ flops)
3. $Q(\boldsymbol{\theta}) = \mathbf{u}^\top\mathbf{u}$ ($2n$ flops)
4. log-determinant is $2\sum_{i=1}^n \log l_{ii}$, where l_{ii} are the diagonal entries in $\mathbf{L}$ (n flops)

Given marginal posterior samples $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta} \,|\, \mathbf{y})$, we can draw posterior samples of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ using *composition sampling*.

We'll consider a portion of this algorithm in a subsequent exercise.

For more details see Finley, A.O., S. Banerjee, A.E. Gelfand. (2015) spBayes for large univariate and multivariate point-referenced spatio-temporal data models. *Journal of Statistical Software*, **63**:1–28.