A simple, fast sampler for simulating spatial data and other Markovian data structures + `intRo`

Andee Kaplan
Iowa State University
ajkaplan@iastate.edu

November 29, 2016

A simple, fast sampler for simulating spatial data and other Markovian data structures

Goal

- Markov random field models are possible for spatial or network data
- Rather than specifying a joint distribution directly, a model is specified through a set of full conditional distributions for each spatial location
- Assume the spatial data are on a regular lattice (wrapped on a torus)

Goal: A new, provably fast approach for simulating spatial/network data.

Spatial Markov random field (MRF) models

Notation

- Variables $\{Y(\boldsymbol{s}_i) : i = 1, \ldots, n\}$ at locations $\{\boldsymbol{s}_i : i = 1, \ldots, n\}$
- Neighborhoods: N_i specified according to some configuration
- Neighboring Values: $\boldsymbol{y}(N_i) = \{y(\boldsymbol{s}_j) : \boldsymbol{s}_j \in N_i\}$
- Full Conditionals: $\{f_i(y(\boldsymbol{s}_i)|\boldsymbol{y}(N_i), \boldsymbol{\theta}) : i = 1, \ldots, n\}$
  - $f_i(y(\boldsymbol{s}_i)|\boldsymbol{y}(N_i), \boldsymbol{\theta})$ is conditional pmf/pdf of $Y(\boldsymbol{s}_i)$ given values for its neighbors $\boldsymbol{y}(N_i)$
  - Often assume a common conditional cdf $F_i = F$ form ($f_i = f$) for all i

Exponential family examples

1. Conditional Gaussian (3 parameters):

$$f_i(y(\boldsymbol{s}_i)|\boldsymbol{y}(N_i), \alpha, \eta, \tau) = \frac{1}{\sqrt{2\pi}\tau} \exp\left(-\frac{[y(\boldsymbol{s}_i) - \mu(\boldsymbol{s}_i)]^2}{2\tau^2}\right)$$

$Y(\boldsymbol{s}_i)$ given neighbors $\boldsymbol{y}(N_i)$ is normal with variance τ^2 and mean

$$\mu(\boldsymbol{s}_i) = \alpha + \eta \sum_{\boldsymbol{s}_j \in N_i} [y(\boldsymbol{s}_j) - \alpha]$$

2. Conditional Binary (2 parameters):
$Y(\boldsymbol{s}_i)$ given neighbors $\boldsymbol{y}(N_i)$ is Bernoulli $p(\boldsymbol{s}_i, \kappa, \eta)$ where

$$\mathrm{logit}[p(\boldsymbol{s}_i, \kappa, \eta)] = \mathrm{logit}(\kappa) + \eta \sum_{\boldsymbol{s}_j \in N_i} [y(\boldsymbol{s}_j) - \kappa]$$

In both examples, η represents a dependence parameter.

Concliques

Cliques – Hammersley and Clifford [1971]

Singletons and sets of locations such that each location in the set is a neighbor of all other locations in the set

Example: Four nearest neighbors gives cliques of sizes 1 and 2

The Converse of Cliques – Concliques

Sets of locations such that no location in the set is a neighbor of any other location in the set

4 Nearest Neighbors

$$\begin{array}{ccc} \cdot & * & \cdot \\ * & \boldsymbol{s} & * \\ \cdot & * & \cdot \end{array}$$

Concliques 4 Nearest Neighbors

$$\begin{array}{cccc} 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \end{array}$$

8 Nearest Neighbors

$$\begin{array}{ccc} * & * & * \\ * & \boldsymbol{s} & * \\ * & * & * \end{array}$$

Concliques 8 Nearest Neighbors

$$\begin{array}{cccc} 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \\ 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \end{array}$$

Generalized spatial residuals

Definition

- $F(y|\boldsymbol{y}(N_i), \boldsymbol{\theta})$ is the conditional cdf of $Y(\boldsymbol{s}_i)$ under the model
- Substitute random variables, $Y(\boldsymbol{s}_i)$ and neighbors $\{Y(\boldsymbol{s}_j) : \boldsymbol{s}_j \in N_i\}$, into (continuous) conditional cdf to define residuals:

$$R(\boldsymbol{s}_i) = F(Y(\boldsymbol{s}_i)|\{Y(\boldsymbol{s}_j) : \boldsymbol{s}_j \in N_i\}, \boldsymbol{\theta}).$$

Key Property

Let $\{\mathcal{C}_j : j = 1, \dots, q\}$ be a collection of concliques that partition the integer grid. Under the conditional model, **spatial residuals** *within* **a conclique are iid Uniform**$(0, 1)$**-distributed**:

$$\{R(\boldsymbol{s}_i) : \boldsymbol{s}_i \in \mathcal{C}_j\} \overset{iid}{\sim} \text{Uniform}(0, 1) \qquad \text{for } j = 1, \dots, q$$

(Kaiser et al. [2012])

Conclique-based Gibbs sampler

Using the conditional independence of random variables at locations within a conclique along with the probability integral transform we propose a conclique-based Gibbs sampling algorithm for sampling from a MRF.

1. Split locations into Q disjoint concliques, $\mathcal{D} = \cup_{i=1}^{Q} \mathcal{C}_i$.
2. Initialize the values of $\{Y^{(0)}(\boldsymbol{s}) : \boldsymbol{s} \in \{\mathcal{C}_2, \dots, \mathcal{C}_Q\}\}$.
3. Sample from the conditional distribution of $Y(\boldsymbol{s})$ given $\{Y(\boldsymbol{t}) : \boldsymbol{t} \in \mathcal{N}(\boldsymbol{s})\}$ for $\boldsymbol{s} \in \mathcal{C}_1$,
    1. Sample $\{U(s) : s \in \mathcal{C}_1\} \stackrel{iid}{\sim} Unif(0, 1)$
    2. For each $\boldsymbol{s} \in \mathcal{C}_1$, $Y^{(i)}(\boldsymbol{s}) = F^{-1}(U(\boldsymbol{s}) | Y^{(i-1)}(\boldsymbol{t}), \boldsymbol{t} \in \mathcal{N}(\boldsymbol{s}))$
4. Sample from the conditional distribution of $Y(\boldsymbol{s})$ given $\{Y(\boldsymbol{t}) : \boldsymbol{t} \in \mathcal{N}(\boldsymbol{s})\}$ for $\boldsymbol{s} \in \mathcal{C}_j; j = 2, \dots, Q$,
    1. Sample $\{U(s) : s \in \mathcal{C}_2\} \stackrel{iid}{\sim} Unif(0, 1)$
    2. For each $\boldsymbol{s} \in \mathcal{C}_j$, $Y^{(i)}(\boldsymbol{s}) = F^{-1}(U(\boldsymbol{s}) | \{Y^{(i)}(\boldsymbol{t}), \boldsymbol{t} \in \mathcal{N}(\boldsymbol{s}) \cap \mathcal{C}_k \text{ where } k < j\}, \{Y^{(i-1)}(\boldsymbol{t}), \boldsymbol{t} \in \mathcal{N}(\boldsymbol{s}) \cap \mathcal{C}_k \text{ where } k > j\})$

It's (provably) fast!

1. In many (commonly used) four-nearest neighbor models (including Gaussian and binary), the conclique-based Gibbs sampler is provably **geometrically ergodic**.
2. Because we are using batch updating vs. sequential updating of each location, this approach is also **computationally fast**.
3. This approach is implemented in a flexible `R` package (called `conclique`, to appear on CRAN) that implements a conclique-based Gibbs sampler while allowing the user to specify an arbitrary model.

Preliminary simulations

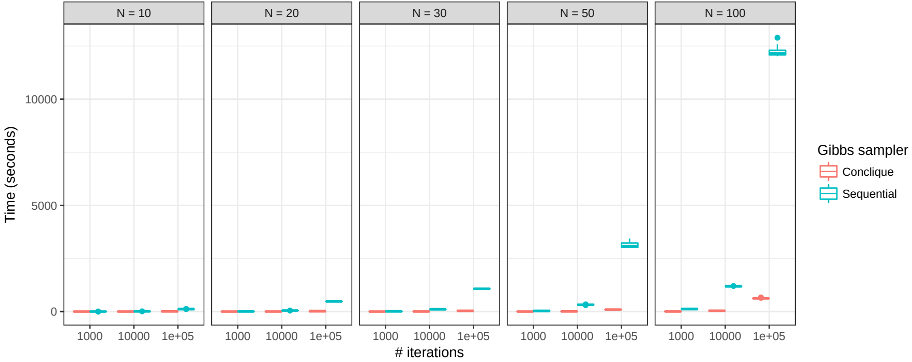

Figure 1: Comparisons of timing for simulation of 4NN Gaussian Markov Random Field data on a lattice of size $N \times N$ for various size grids, $N = 10, 20, 30, 50, 100$, using sequential and conclique-based Gibbs samplers.

Ideas and connections

- Goodness-of-fit test for network data
  - The model-based method of resampling re-frames network into a collection of neighborhoods by using covariate information
  - Creates concliques on a graph structure.
  - Use a conditionally specified network distribution to sample network data in a blockwise conclique-based Gibbs sampler.

`intRo`: statistical analysis software for teaching

Do we really need another statistical software package?

Short answer: **yes**

- `R` is great, but requires students to have some knowledge/interest in programming
- `JMP`, `Deducer`, `Rcmdr` are powerful, but too big
    - Licenses and installation
- New tools recently released to spark an interest in `R`
    - `Swirl` and DataCamp teach `R` programming
    - Project MOSAIC facilitates learning, but assumes knowledge of `R`
- Want students to focus on data analysis rather than fight with software

What is `intRo`?

- A simple **web-based** application for performing basic data analysis and statistical routines and accompanying utility package
- Built using `R` and `Shiny`
- Extensible modular structure
- Designed for a first statistics class student
- Assists in the learning of statistics rather than acting as a stand-alone deliverer of statistics education

Easy

- Focused on aspects of the user interface (UI) and output that make it easy to pick up without training
- Minimal necessary functionality for an introductory statistics course
- Organized around specific tasks a student may perform in the process of a data analysis

Exciting

- Fun, easy to use (available on the web)
- Interactive plots using `ggvis`

Ulterior motive: get students excited about programming

- By navigating about the user interface of `intRo`, students are creating a fully-executable `R` script that they can download and run locally
- Viewing their script change real-time within the application

Extensible

- User interaction with `intRo` is split into bitesize chunks that we call *modules*
- Each module is a self contained set of `R` code that is dynamically added to the application at run time
- `intRo` can be easily extended by the addition of modules within the frame-work underlying the application
- Allows instructors/collaborators to tailor `intRo` to the needs of a particular course

Take a look http://intro-stats.com

intRo package

Installation:

```
devtools::install_github("rstudio/shinyapps")
devtools::install_github("gammarama/intRo")`
```

Functions:

- `download_intRo` - Downloads a current revision of `intRo` to your machine
- `run_intRo` - Runs an `intRo` session locally with the specified options
- `deploy_intRo` - Deploys an instance of `intRo` to ShinyApps.io with the specified options

Future work - Module creation

Modularity is a key feature of `intRo`, but module creation is currently:

- Undocumented
- Entirely manual
- Unnecessarily lengthy

Idea: Include funcitonality in current R package to automate creation of `intRo` modules

References I

John M Hammersley and Peter Clifford. Markov fields on finite graphs and lattices. 1971.

Mark S Kaiser, Soumendra N Lahiri, and Daniel J Nordman. Goodness of fit tests for a class of markov random field models. *The Annals of Statistics*, pages 104–130, 2012.