Spatio-temporal dependence: a blessing and a curse for computation and inference (illustrated by compositional data modeling) (and with an introduction to NIMBLE)

Christopher Paciorek UC Berkeley Statistics

Joint work with:
The PalEON project team (http://paleonproject.org)
The NIMBLE development team (http://r-nimble.org)

NSF-CBMS Workshop on Bayesian Spatial Statistics
August 2017

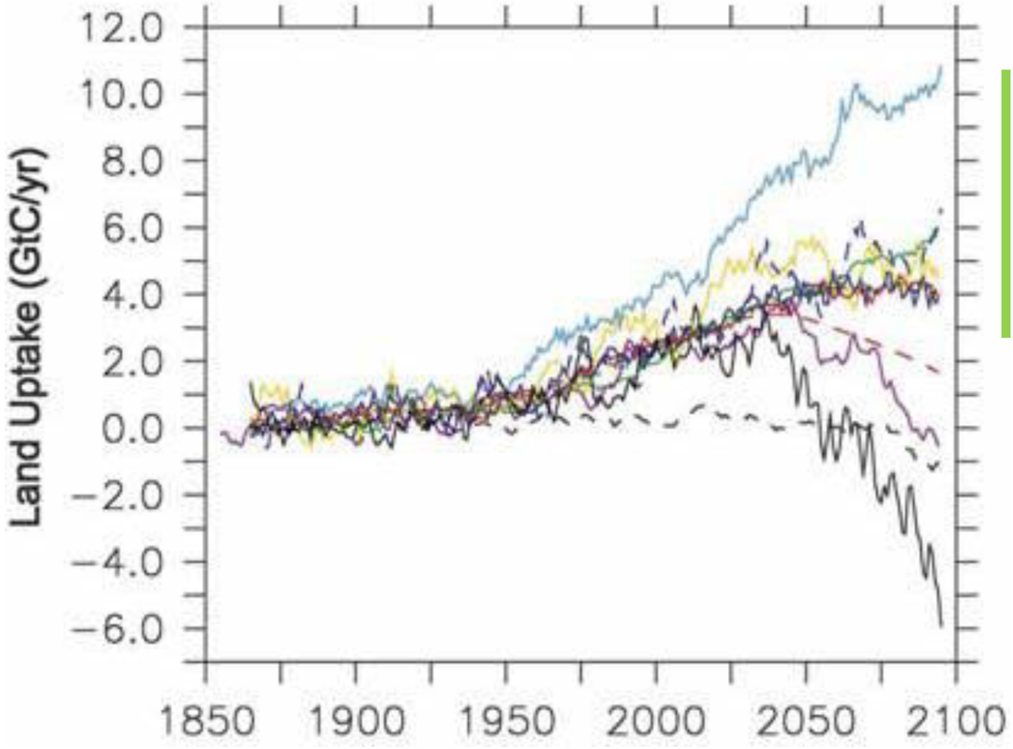
Funded by various NSF grants to the PalEON and NIMBLE projects

PalEON Project

Goal: Improve the predictive capacity of terrestrial ecosystem models

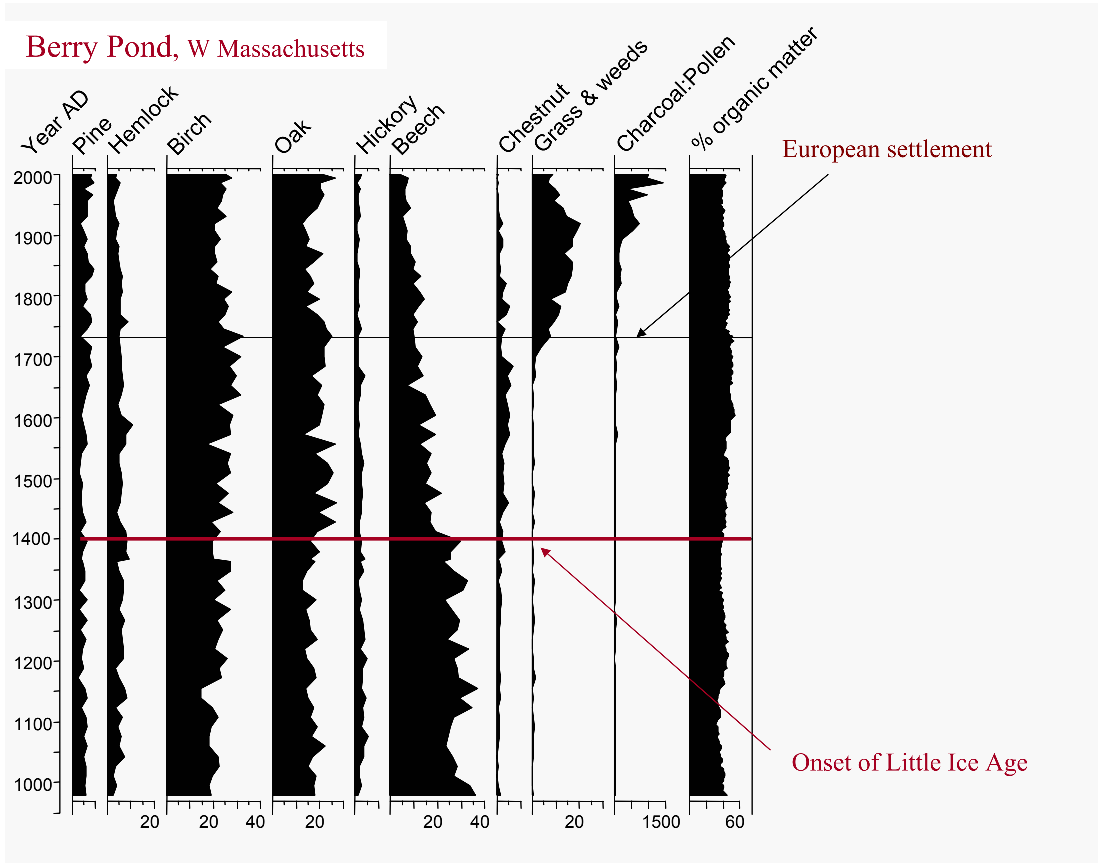
"This large variation among carbon-cycle models … has been called 'uncertainty'. I prefer to call it 'ignorance'."
- Prentice (2013) Grantham Institute

Friedlingstein et al. 2006 J. Climate



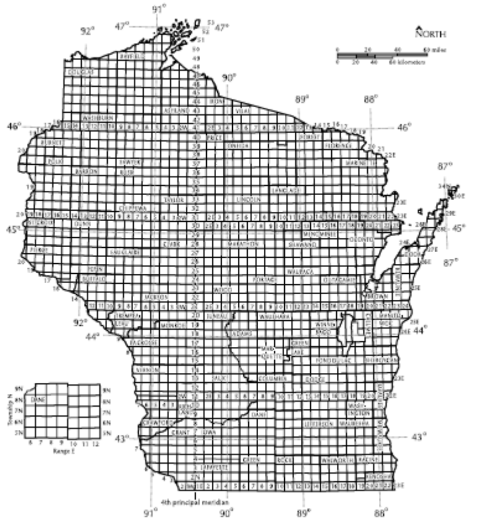
Critical issue: model parameterization and representation of decadal- to centennial-scale processes are poorly constrained by data
Approach: use historical and fossil data to estimate past vegetation and climate and use this information for model initialization, assessment, and improvement

Fossil Pollen Data

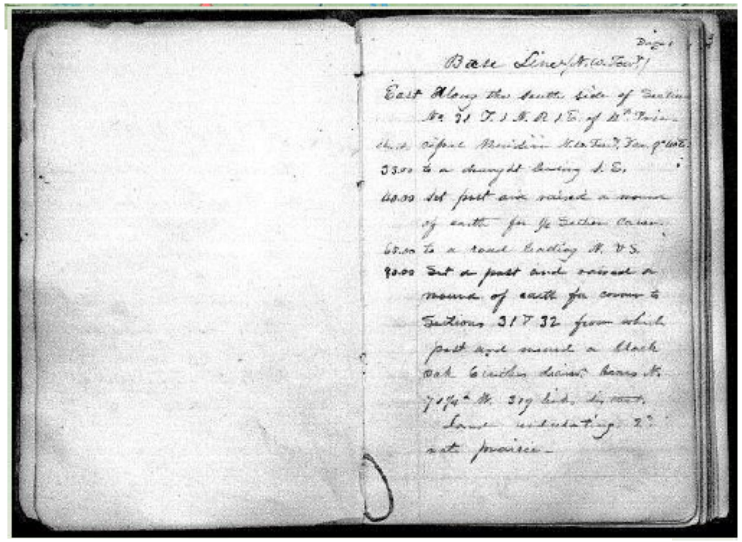


Settlement-era Land Survey Data

Survey grid in Wisconsin



Surveyor notes

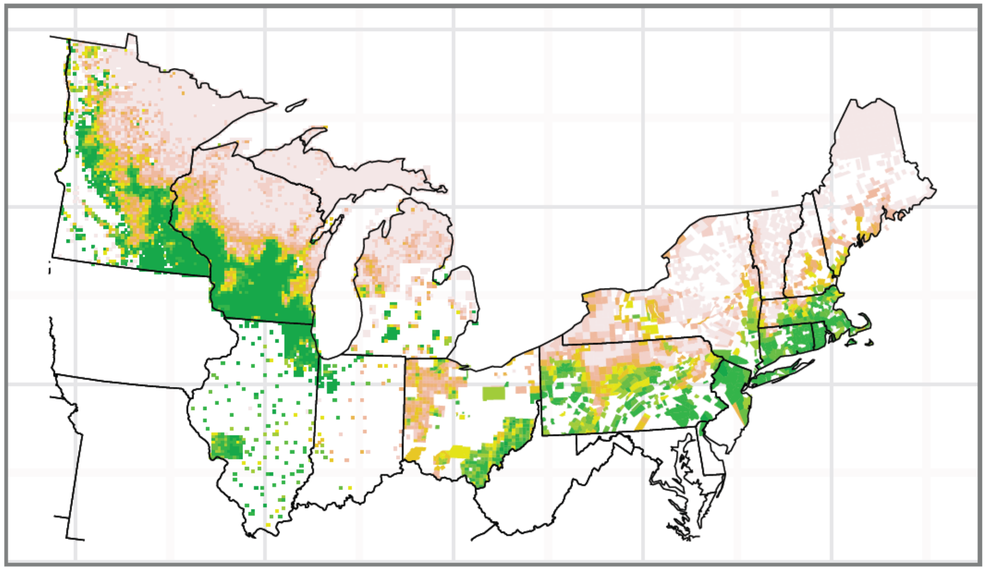


Page 1

Base Line (N.W. Terr'y)

East Along the South side of Section No. 31 T. 1 N. R 1 E. of 4th Princi-
chains pal Meridian N.W. Terr'y Var. 9° East
33.00 to a branch running S.E.
40.00 Set post and raised a mound of earth for 1/4 Section Corner.
65.00 to a road leading N. & S.
80.00 Set a post and raised a mound of earth for corner to Sections 31 & 32 from which post and mound a black oak 6 inches diam. bears N. 71 1/4° W. 37 links dist.
Land undulating 2nd rate prairie

Raw oak tree proportions (on a grid in the western portion and in irregular township areas in the eastern portion)

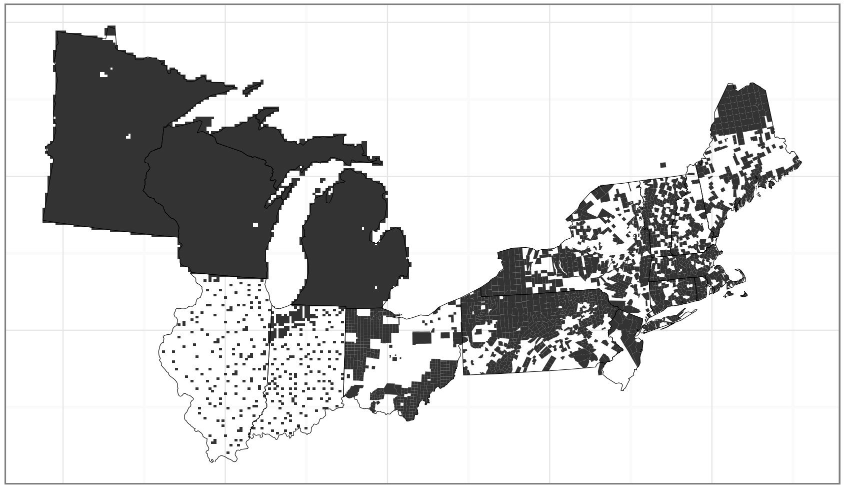


Outline

- Application 1: Spatial smoothing of compositional data
  - Setting: Multivariate data, high-dimensional quantities, non-conjugate models
  - A hierarchical multinomial probit model with CAR spatial process
  - Data augmentation
  - How much smoothness (in space)?
  - Computational implications
- Computational tools
  - Overview of current software
  - Introduction to NIMBLE
- Application 2: Temporal prediction of biomass from compositional data
  - How much smoothness (in time)?
  - A hierarchical stick-breaking compositional model with Generalized Pareto nonstationary temporal smoothing
  - Default MCMC and computational challenges
  - Customized MCMC using NIMBLE
- Concluding thoughts

Application 1: Spatial smoothing of compositional data

- Multivariate: ~20 taxa (species)
  - Sum-to-one constraint on proportions
- 8 km by 8 km grid:
  - ~10,000 grid points
- 1.3 million trees (> 20 cm diameter) in total
  - ~125 trees per grid cell



Application 1: Should we model spatial dependence?

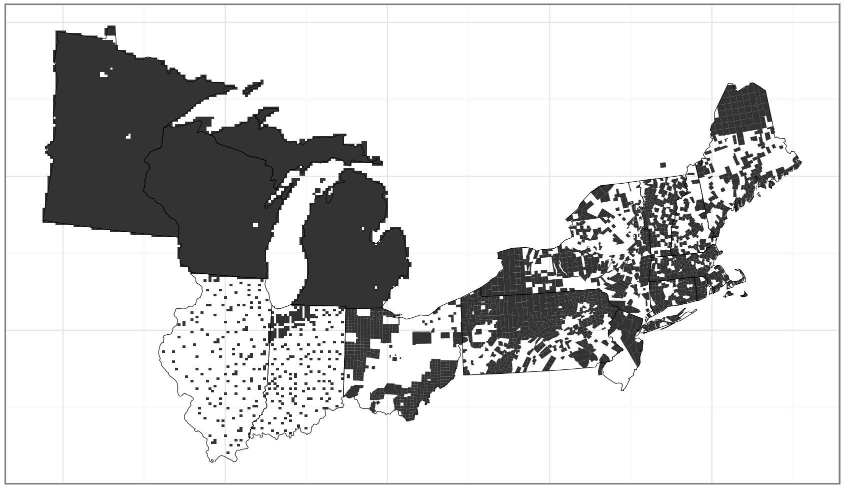
- Yes:
  - We want to estimate composition at all locations.
  - We want to smooth over noise at observed locations.
  - We are interested in joint inference for multiple locations, so we need to account for posterior covariance.
- No:
  - We would need to model the spatial dependence, with the resulting computational implications.

Application 1: Should we model multivariate dependence?

- Yes:
  - Taxa do show correlated abundance (taxa have similarities in their ecological characteristics).
  - If joint inference on multiple tree species is desired, need multivariate correlation structure to properly characterize given our actual knowledge.
- No:
  - Dependence varies by location (nonstationarity)
    - E.g., hemlock/beech positively correlated in general, but beech not present in some locations where hemlock appears (different western range limits)
    - Would require more complex model
  - Locations with data have data for all taxa
    - Imputation is only spatial not multivariate
    - With no measurement error and separable covariance, kriging prediction for a taxon depends only on data from that taxon at other locations
    - Inference not focused on multi-taxon functionals

Application 1: Spatial smoothing of compositional data

- Multivariate: ~20 taxa (species)
    - Sum-to-one constraint on proportions
- 8 km by 8 km grid:
    - ~10,000 grid points
- 1.3 million trees (> 20 cm diameter) in total
    - ~125 trees per grid cell



Model overview:

- Multinomial likelihood (no over-dispersion)
- One spatial process per taxon
    - Sum-to-one constraint based on a multinomial probit specification
    - Otherwise, no multivariate structure
- Spatial process hyperparameters

Application 1: Standard Spatial Multinomial Logit Model

A spatial multinomial logit model:

$$\begin{aligned} y_i &\sim \mathrm{Multi}(n_i, \theta(s_i)) \\ \theta_p(s_i) &= \frac{\exp(g_p(s_i))}{\sum_k \exp(g_k(s_i))} \\ g_p(\cdot) &\sim \mathrm{GP}(\phi_p) \end{aligned}$$

for location i and taxon p.

Computational implications:

- No conjugacy!
- Can't integrate analytically over the latent processes
- How propose good values of each *g* process?

Consider McCulloch and Rossi (1994) multinomial extension of Albert and Chib (1993) data augmentation (DA) trick for probit regression.

Application 1: Spatial Multinomial Probit Model with Data Augmentation

A spatial multinomial probit model:

$$\begin{aligned} y_{ij} &= p \text{ iff } w_{ijp} = \max_k w_{ijk} \\ w_{ijp} &\sim \mathcal{N}(g_p(s_i), 1) \\ g_p(\cdot) &\sim \mathrm{GP}(\phi_p) \end{aligned}$$

for location i, tree j, and taxon p.

Computational implications:

- Data augmentation version allows conjugate updates of each *g* process
- But! Introduce new level in model – higher dimensional and with potential for cross-level dependence to impede MCMC performance

Application 1: How much smoothness?

Application is based on 8 km grid, so CAR style (i.e., Markov random field) models a natural choice.
How smooth spatially?

- First order (simple neighborhood) CAR models: not smooth spatially.

$$\begin{aligned} y_{ij} &= p \text{ iff } w_{ijp} = \max_k w_{ijk} \\ w_{ijp} &\sim \mathcal{N}(g_p(s_i), 1) \\ g_p &\sim \mathcal{N}(0, \sigma_p^2 Q^-) \ \text{(ICAR)} \end{aligned}$$

- Second order (thin-plate spline) CAR models: very smooth spatially.
- Lindgren et al (2011) SPDE approximation to Matern-based Gaussian process: range parameter and limited control over differentiability parameter.

Application 1: Smoothness and computation

- Sparse precision matrices
  - Very computationally efficient for conjugate updates
  - Without conjugacy not clear how to generate good proposals for entire spatial field for a taxon, so computational efficiency of limited relevance
    - Location-specific updates would mix poorly when there is strong spatial dependence
    - Simple CAR models may show reasonable mixing for spatial process values with fixed hyperparameters because of lesser spatial smoothness
- Cross-level dependence from separate updates of latent data values, spatial process values, spatial hyperparameters
  - Updates of spatial process and hyperparameters not directly informed by data

Application 1: MCMC design

$$\begin{aligned} y_{ij} &= p \text{ iff } w_{ijp} = \max_k w_{ijk} \\ w_{ijp} &\sim \mathcal{N}(g_p(s_i), 1) \\ g_p &\sim \mathcal{N}(0, \sigma_p^2 Q^-) \text{ (ICAR)} \end{aligned}$$

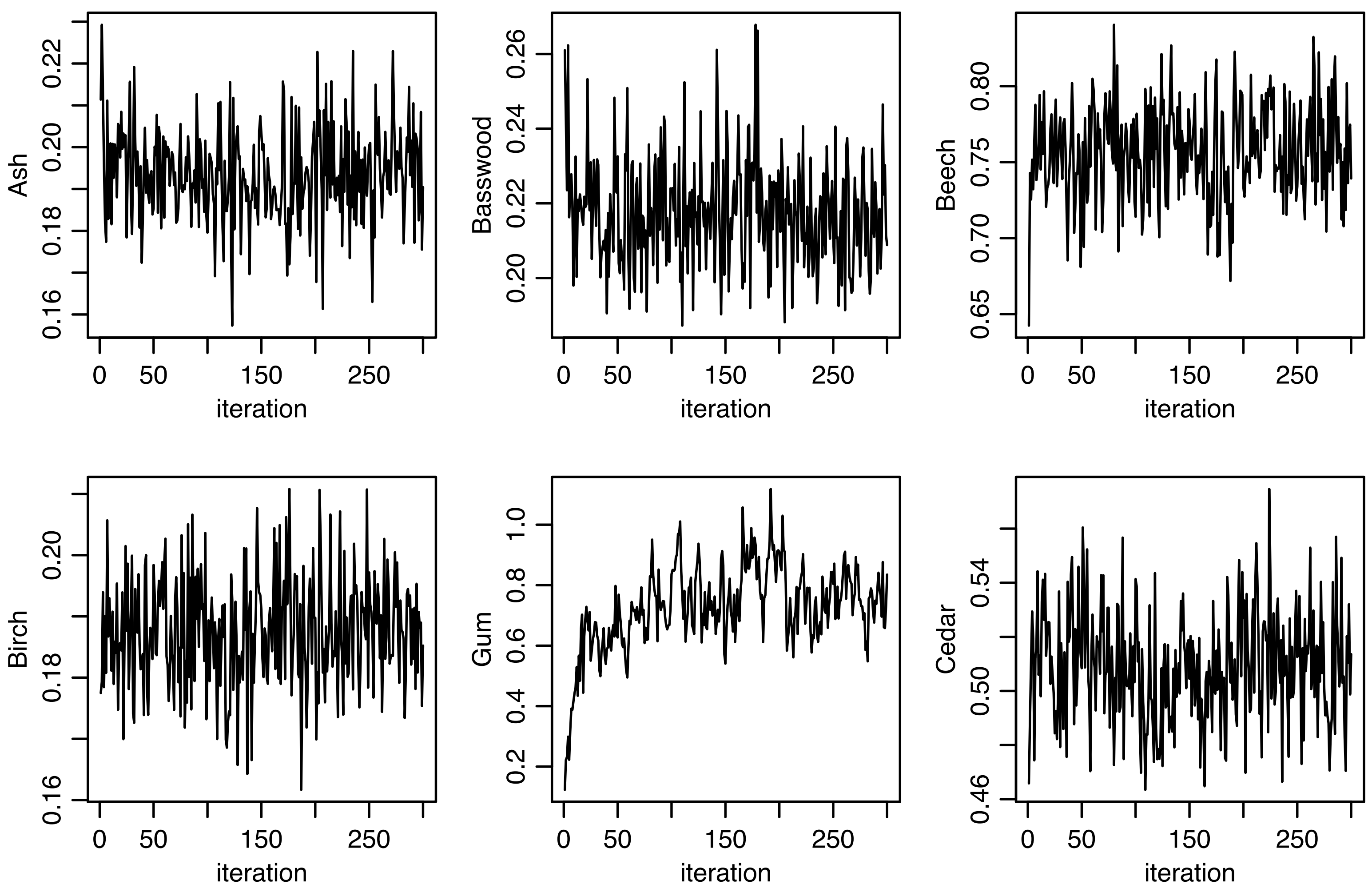
- Cross-level dependence from separate updates of latent data values, spatial process values, spatial hyperparameters
- Adequate performance required joint (cross-level) updates of $\{g_p, \sigma_p\}$:
  - Metropolis proposal for σ_p with conjugate proposal for g_p
  - Equivalent to marginalizing over g_p but avoids correlated truncated normal density for *w*

Application 1: MCMC implementation

$$\begin{aligned} y_{ij} &= p \text{ iff } w_{ijp} = \max_k w_{ijk} \\ w_{ijp} &\sim \mathcal{N}(g_p(s_i), 1) \\ g_p &\sim \mathcal{N}(0, \sigma_p^2 Q^-) \ (\text{ICAR}) \end{aligned}$$

- Overall MCMC written in R
- Truncated normal computations done in C++ via Rcpp (can also use openMP for parallelization)
- Joint $\{g_p, \sigma_p\}$ samples done in R using sparse matrix computations with spam package (which uses Fortran)
- Even with customization, MCMC takes order of two weeks
- Computation pre-dates NIMBLE but NIMBLE designed to allow users to set up customized MCMC sampling for components of models
    - E.g., the joint $\{g_p, \sigma_p\}$ sampling could be coded as a user-defined sampler in NIMBLE (and NIMBLE provides such a sampler for some such situations)

Application 1: MCMC performance



Trace plots for taxon-specific hyperparameters

Application 1: Results

Model selection:

- First order CAR and Lindgren GP approximation have similar performance but GP approximation has anomalies at the spatial boundaries.
- Second order (thin plate spline) CAR too smooth.

Prediction:

http://gandalf.berkeley.edu:3838/paciorek/setVegComp

Bayesian software landscape

Hand-coded algorithms:

- R, Python: fast to develop and easy to share, but slow computation
- C++, Rcpp: slower to develop and harder to share, but fast computation
- Julia: fast to develop and fast computationally but less widely used

Black-box MCMC engines:

- JAGS: single variable samplers with a focus on conjugate samplers
- Stan: Hamiltonian MC, variational Bayes
- PyMCMC3: flexible sampler choice, Hamiltonian MC, variational Bayes

NIMBLE:

- Customizable MCMC and other algorithms plus a system for programming algorithms for hierarchical models in R

Application 1: Software needs

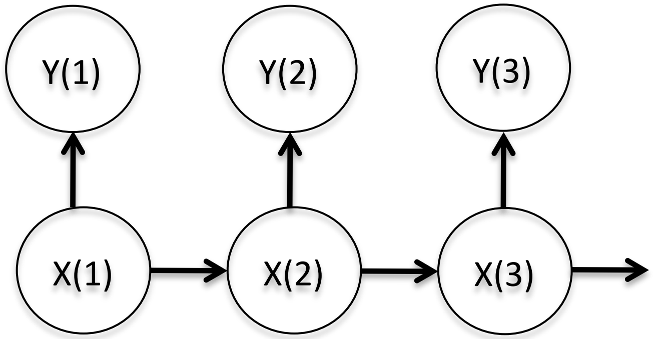
- Exploit sparsity
- Flexibility in choosing samplers for parts of the model
- Joint sampling of spatially-dependent process values
- Customize joint sampling of hyperparameters and spatial process to improve mixing
- Use compiled code for computational bottlenecks

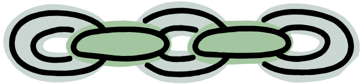
Notes:

- NIMBLE can't do all of this yet (no sparse matrices right now), but designed for such flexibility
- Would be interesting to compare performance of my customized sampling to Stan's HMC

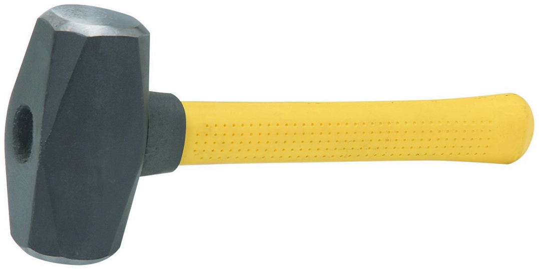
Existing software

Model





Algorithm



e.g., BUGS (WinBUGS, OpenBUGS, JAGS), INLA, Stan,
various R packages

NIMBLE: The Goal

Model

$$\begin{tikzcd}
Y(1) & Y(2) & Y(3) & \\
X(1) \arrow[u] \arrow[r] & X(2) \arrow[u] \arrow[r] & X(3) \arrow[u] \arrow[r] & {}
\end{tikzcd}$$

+

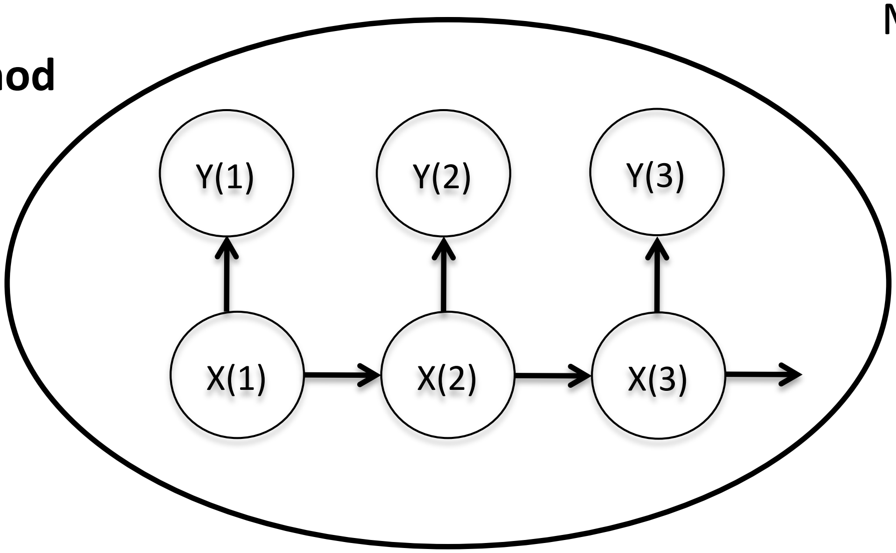
Algorithm language

=

Divorcing Model Specification from Algorithm

MCMC Flavor 1

Your new method



MCMC Flavor 2

Data cloning

Particle Filter

MCEM

Quadrature

Importance Sampler

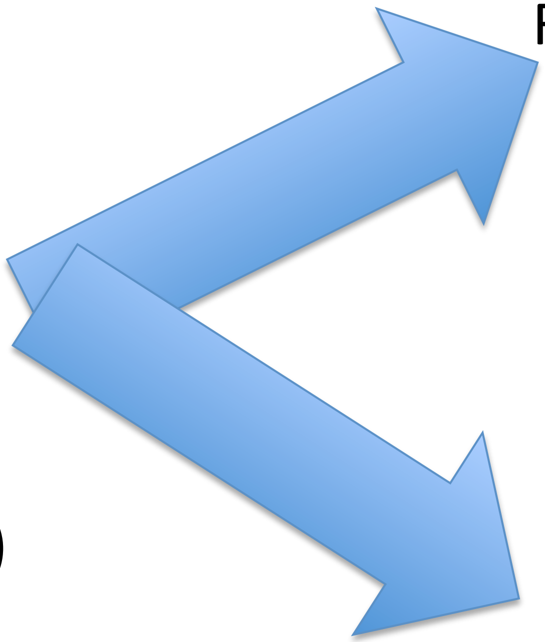
Maximum likelihood

NIMBLE’s goals

- Retaining BUGS compatibility
- Providing a variety of standard algorithms
- **Allowing developers to add new algorithms (including modular combination of algorithms)**
- Allowing users to operate within R
- Providing speed via compilation to C++, with R wrappers

NIMBLE System Summary

statistical model
(BUGS code)
+
algorithm
(nimbleFunction)



R objects + R under the hood

R objects + C++ under the hood

✧ We generate C++ code,
✧ compile and load it,
✧ provide interface object.

NIMBLE

1. Model specification

    BUGS language $\rightarrow$ R/C++ model object

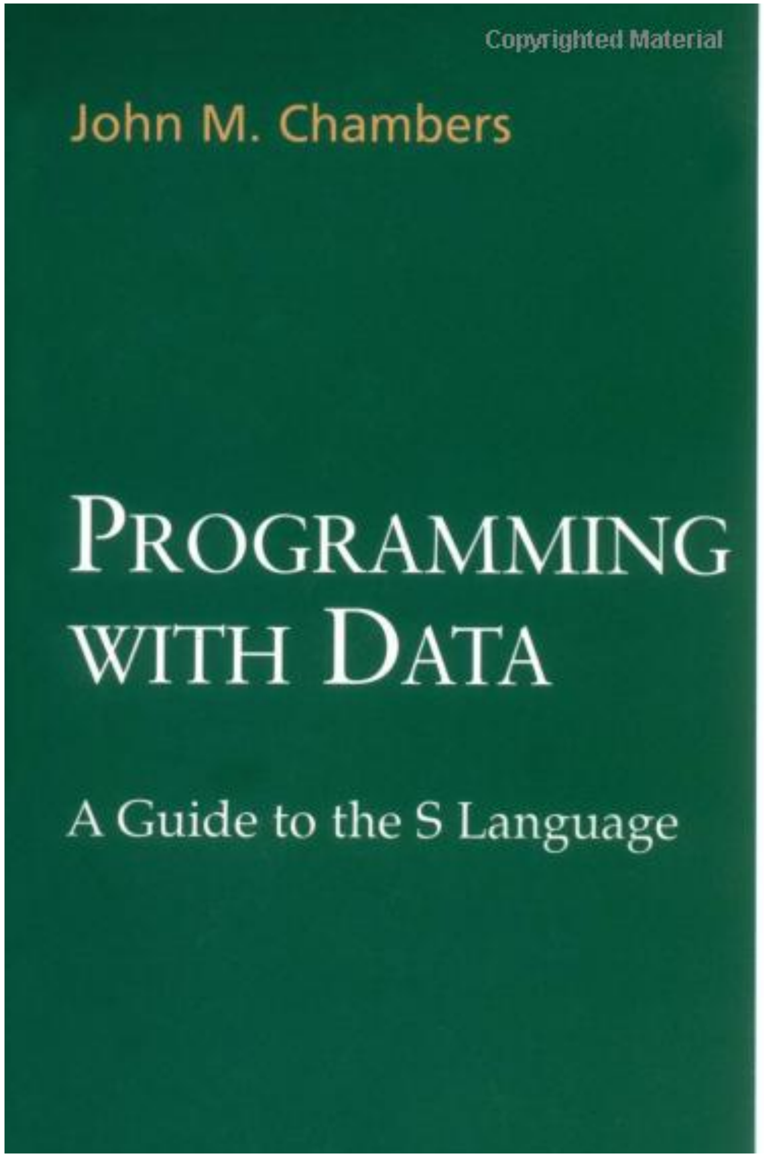
2. Algorithm library

    MCMC, Particle Filter/Sequential MC, etc.

3. Programming algorithms

    NIMBLE programming language within R $\rightarrow$ R/C++ algorithm object

The Success of R

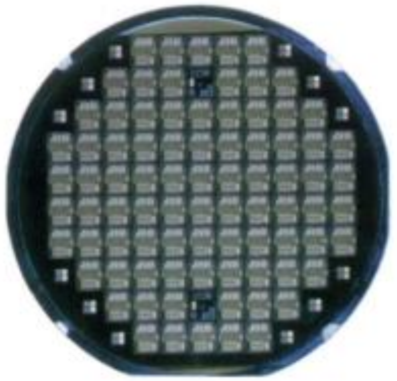


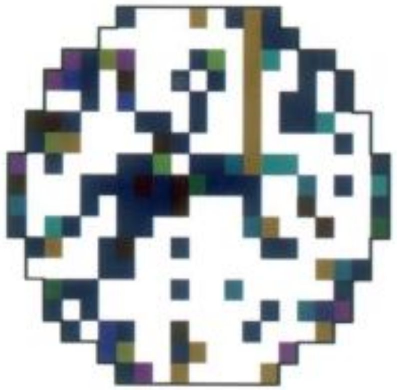
Copyrighted Material

John M. Chambers

PROGRAMMING WITH DATA

A Guide to the S Language





NIMBLE: Programming with Models

You give NIMBLE:

```
littersCode <- nimbleCode( {
for(j in 1:G) {
    for(I in 1:N) {
        r[i, j] ~ dbin(p[i, j], n[i, j]);
        p[i, j] ~ dbeta(a[j], b[j]);
     }
     mu[j] <- a[j]/(a[j] + b[j]);
     theta[j] <- 1.0/(a[j] + b[j]);
     a[j] ~ dgamma(1, 0.001);
     b[j] ~ dgamma(1, 0.001);   } )
```

You get this:

```
> littersModel$a[1] <- 5              # set values in model
> simulate(littersModel, ‘p')         # simulate from prior
> p_deps <- littersModel$getDependencies(‘p’)    # model structure
> calculate(littersModel, p_deps)  # calculate probability density
> getLogProb(pumpModel, ‘r')
```

NIMBLE also extends BUGS: multiple parameterizations, named parameters, and user-defined distributions and functions.

User Experience: Specializing an Algorithm to a Model

```
littersModelCode <- modelCode({
 for(j in 1:G) {
  for(I in 1:N) {
       r[i, j] ~ dbin(p[i, j], n[i, j]);
       p[i, j] ~ dbeta(a[j], b[j]);
   }
   mu[j] <- a[j]/(a[j] + b[j]);
   theta[j] <- 1.0/(a[j] + b[j]);
   a[j] ~ dgamma(1, 0.001);
   b[j] ~ dgamma(1, 0.001);
})
```

```
sampler_slice <- nimbleFunction(
   setup = function((model, mvSaved, control) {
       calcNodes <- model$getDependencies(control$targetNode)
       discrete <- model$getNodeInfo()[[control$targetNode]]$isDiscrete()
[…snip…]
  run = function() {
      u <- getLogProb(model, calcNodes) - rexp(1, 1)
      x0 <- model[[targetNode]]
      L <- x0 - runif(1, 0, 1) * width
 […snip….]
…
```

```
> littersMCMCconf <- configureMCMC(littersModel)
> littersMCMCconf$printSamplers()
[…snip…]
[3] RW sampler;   targetNode: b[1],  adaptive: TRUE,  adaptInterval: 200,  scale: 1
[4] RW sampler;   targetNode: b[2],  adaptive: TRUE,  adaptInterval: 200,  scale: 1
[5] conjugate_beta sampler;   targetNode: p[1, 1],  dependents_dbin: r[1, 1]
[6] conjugate_beta sampler;   targetNode: p[1, 2],  dependents_dbin: r[1, 2]
 […snip…]
> littersMCMCconf$addSampler('a[1]', 'slice', list(adaptInterval = 100))
> littersMCMCconf$addSampler('a[2]', 'slice', list(adaptInterval = 100))
> littersMCMCconf$addMonitors('theta')
> littersMCMC <- buildMCMC(littersMCMCspec)
> littersMCMC_Cpp <- compileNimble(littersMCMC, project = littersModel)
> littersMCMC_Cpp$run(20000)
```

NIMBLE

1. Model specification

   BUGS language → R/C++ model object

2. Algorithm library

   MCMC, Particle Filter/Sequential MC, MCEM, etc.

3. Programming algorithms

   NIMBLE programming language within R → R/C++ algorithm object

NIMBLE’s algorithm library

- MCMC samplers:
  - Conjugate, adaptive Metropolis, adaptive blocked Metropolis, slice, elliptical slice sampler, particle MCMC, specialized samplers for particular distributions (Dirichlet, CAR)
  - Flexible choice of sampler for each parameter
  - User-specified blocks of parameters
- Sequential Monte Carlo (particle filters)
  - Various flavors
- MCEM
- Write your own

NIMBLE

1. Model specification

   BUGS language → R/C++ model object

2. Algorithm library

   MCMC, Particle Filter/Sequential MC, etc.

3. Algorithm specification

   NIMBLE programming language within R → R/C++ algorithm object

NIMBLE: Programming With Models

We want:

- High-level processing (model structure) in R
- Low-level processing in C++

NIMBLE: Programming With Models

```
sampler_myRW <- nimbleFunction(

setup = function(model, mvSaved, targetNode, scale) {
   calcNodes <- model$getDependencies(targetNode)
},
run = function() {
   model_lp_initial <- calculate(model, calcNodes)
   proposal <- rnorm(1, model[[targetNode]], scale)
   model[[targetNode]] <<- proposal
   model_lp_proposed <- calculate(model, calcNodes)
   log_MH_ratio <- model_lp_proposed - model_lp_initial

    if(decide(log_MH_ratio)) jump <- TRUE
      else                   jump <- FALSE
    # .... Various bookkeeping operations ... #    })
```

$\left.\vphantom{\rule{0pt}{10em}}\right\}$ 2 kinds of functions

NIMBLE: Programming With Models

```
sampler_myRW <- nimbleFunction(

setup = function(model, mvSaved, targetNode, scale) {
   calcNodes <- model$getDependencies(targetNode)
},
run = function() {
   model_lp_initial <- calculate(model, calcNodes)
   proposal <- rnorm(1, model[[targetNode]], scale)
   model[[targetNode]] <<- proposal
   model_lp_proposed <- calculate(model, calcNodes)
   log_MH_ratio <- model_lp_proposed - model_lp_initial

    if(decide(log_MH_ratio)) jump <- TRUE
      else                   jump <- FALSE
    # .... Various bookkeeping operations ... #    })
```

query model
structure
ONCE

NIMBLE: Programming With Models

```
sampler_myRW <- nimbleFunction(

setup = function(model, mvSaved, targetNode, scale) {
   calcNodes <- model$getDependencies(targetNode)
},
run = function() {
   model_lp_initial <- calculate(model, calcNodes)
   proposal <- rnorm(1, model[[targetNode]], scale)
   model[[targetNode]] <<- proposal
   model_lp_proposed <- calculate(model, calcNodes)
   log_MH_ratio <- model_lp_proposed - model_lp_initial

   if(decide(log_MH_ratio)) jump <- TRUE
     else                   jump <- FALSE
   # .... Various bookkeeping operations ... #   })
```

$$\left.\begin{array}{l}\\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\}$$ the actual (generic) algorithm

The NIMBLE compiler (run code)

<u>Feature summary:</u>

- R-like matrix algebra (using Eigen library)
- R-like indexing (e.g. X[1:5,])
- Use of model variables and nodes
- Model calculate (logProb) and simulate functions
- Sequential integer iteration
- If-then-else, do-while
- Access to much of Rmath.h (e.g. distributions)
- Automatic R interface / wrapper
- Call out to your own C/C++ or back to R
- Many improvements / extensions planned

NIMBLE: What can I program?

- Your own distribution for use in a model
- Your own function for use in a model
- Your own MCMC sampler for a variable in a model
- A new MCMC sampling algorithm for general use
- A new algorithm for hierarchical models
- An algorithm that composes other existing algorithms (e.g., MCMC-SMC combinations)

NIMBLE: What can I program?

- Your own distribution for use in a model
- Your own function for use in a model
- Your own MCMC sampler for a variable in a model
- A new MCMC sampling algorithm for general use
- A new algorithm for hierarchical models
- An algorithm that composes other existing algorithms (e.g., MCMC-SMC combinations)

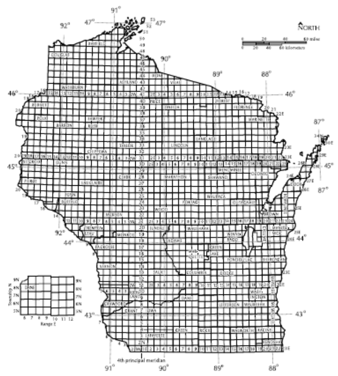
Status of NIMBLE and Next Steps

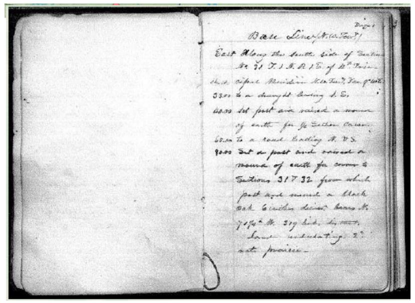
- First release was June 2014 with regular releases since. Lots to do:
  - Improve the user interface and speed up compilation
  - Refinement/extension of the NIMBLE programming language
    - e.g., automatic differentiation, parallelization, sparse matrices
  - Additional algorithms written in NIMBLE DSL
    - e.g., normalizing constant calculations, Laplace approximations, HMC and other samplers
    - Bayesian nonparametrics with Claudia Wehrhahn Cortes and Abel Rodriguez (UCSC)

- Interested?
  - Announcements: nimble-announce Google site
  - User support/discussion: nimble-users Google site
  - Write an algorithm using NIMBLE!
  - Help with development of NIMBLE: email nimble.stats@gmail.com or see github.com/nimble-dev

Application 2: Predicting biomass from compositional data

Calibration: at settlement time we have biomass estimates (based on survey data and a spatial model) and pollen composition (from sediment cores)





Page 1

Base Line (N. W. Terr.)

East Along the south side of Section No. 31 T. 1 N. R. 1 E. of 4th Prin- [illegible] Meridian N. W. Terr. [illegible]

35.00 to a draught leading S. E.

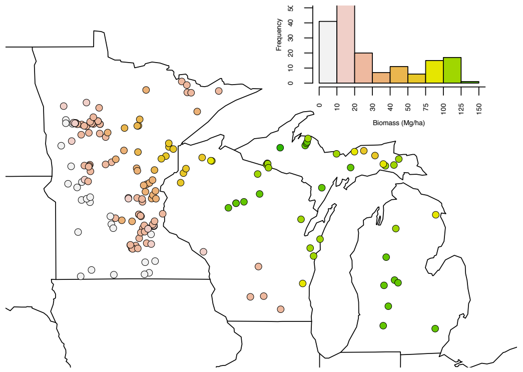
40.00 Set post and raised a mound of earth for 1/4 Section Corner.

60.00 to a road leading N. & S.

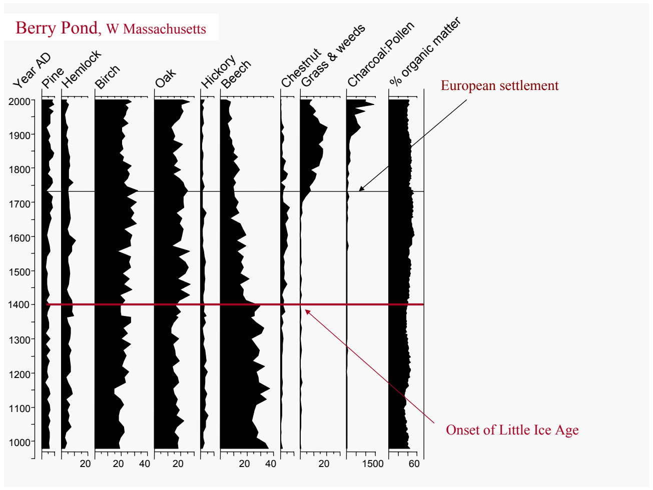
80.00 Set a post and raised a mound of earth for corner to Sections 31 & 32 from which post and mound a black oak 6 inches diamt. bears N. 71 1/4 W. 3 1/7 links distant.

Land undulating 2nd rate prairie.

Biomass estimates at ponds



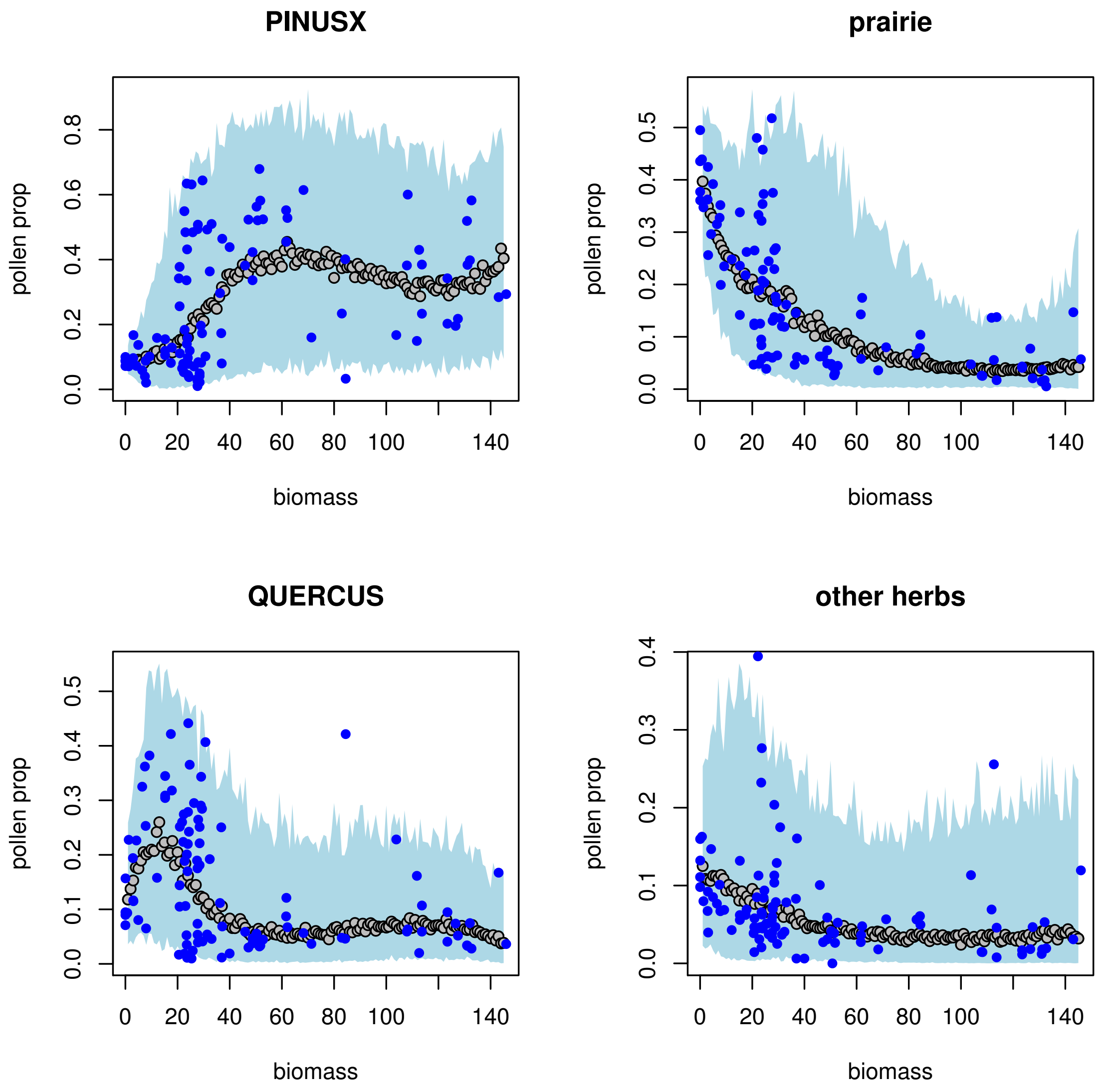
Prediction: based on calibration model and pollen composition over time, predict biomass



Application 2: Calibration model

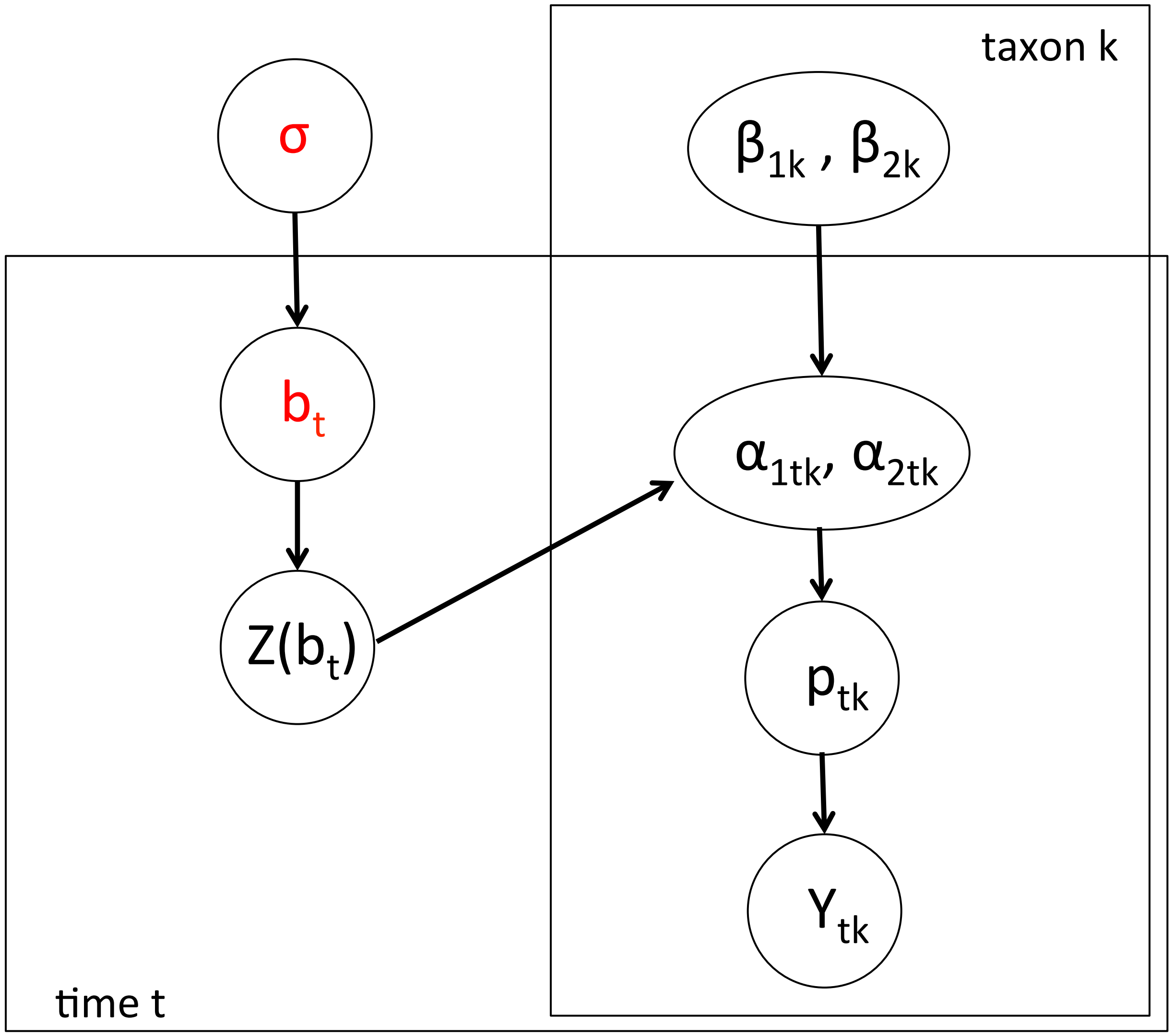
- Pollen proportion for each taxon determined by transformation of a flexible (spline) function of biomass
  - shape1 and shape2 parameters of beta distribution are splines of biomass
  - Primary calibration parameters are spline coefficients
- Multinomial likelihood for pollen counts given modeled proportions
- Fit in NIMBLE (could be fit in various other packages)

Application 2: Calibration model fit



Mean and variability of modeled pollen proportions across ponds vary with biomass

Application 2: Prediction Model



Application 2: Prediction Model

```
for(t in 1:nTimes)
        Y[t, 1] ~ dbetabin(alpha1[t, 1], alpha2[t, 1], n[t])
        for(k in 2:(nTaxa-1)) {
                Y[t, k] ~  dbetabin(alpha1[t, k], alpha2[t, k], n[t]-sum(Y[t, 1:(k-1)]))
```

pollen likelihood

```
for (k in 1:nTaxa)
    for(t in 1:nTimes) {
        alpha1[t, k] <- exp(Zb[t, 1:nKnots] %*% beta1[1:nKnots, k])
        alpha2[t, k] <- exp(Zb[t, 1:nKnots] %*% beta2[1:nKnots, k])
        }
for( t in 1:nTimes)
        Zb[t, 1:nKnots] <- bspline(b[t], knots[1:nKnots])
```

latent predictor

```
for(t in 2:nTimes)
        b[t] ~ dnorm(b[t-1]), sd = sigma)
```

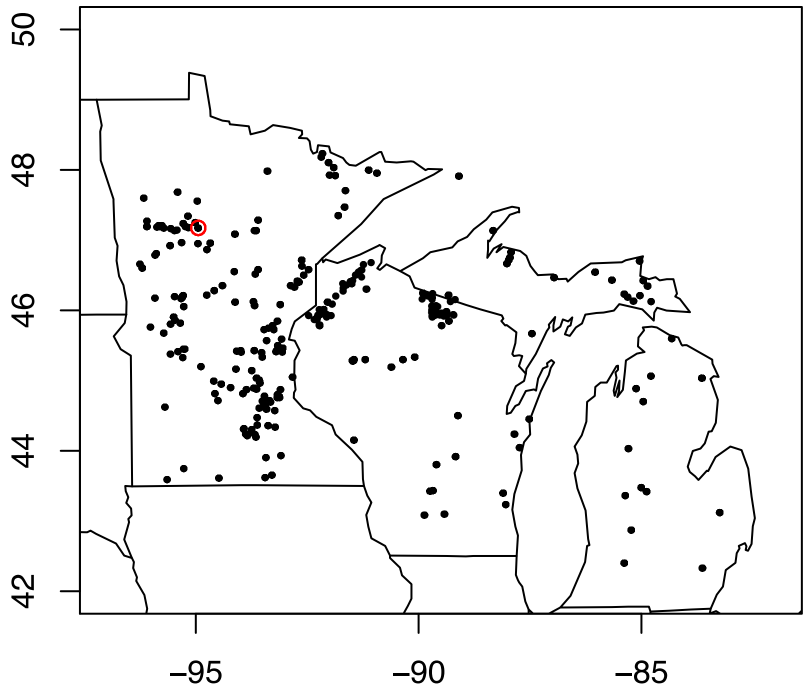
biomass evolution

```
sigma ~ dunif(0, 10) # Gelman (2006)
b[1] ~ dunif(0, 400)
```

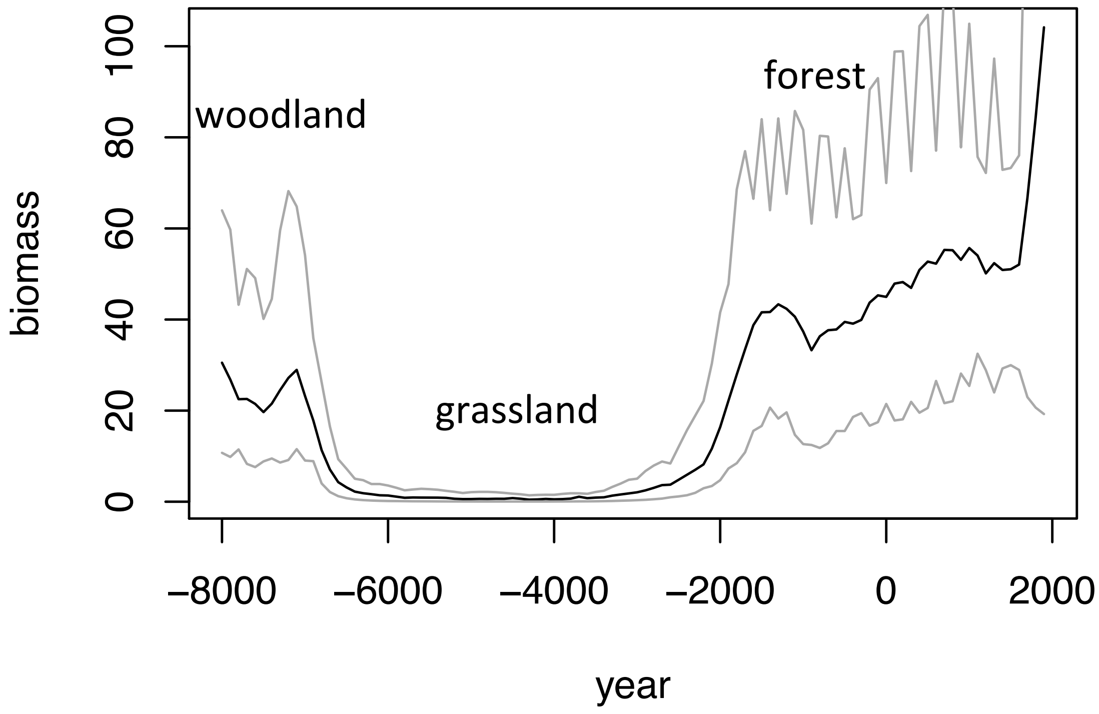
hyperpriors

Application 2: Biomass prediction at one site

Calibration sites and prediction site (red)



Biomass over time



Key ecological question: how does biomass (carbon storage) evolve over time?

Statistical question: how to model temporal process? Smoothness?

- Discrete first-order autoregressive (i.e., CAR) model is not smooth
- Discrete second-order autoregressive (i.e., thin plate spline) is very smooth
- Nonstationarity?

Application 2: Generalized Pareto / Trend filtering

- Discrete autoregressive model is a model (prior) for temporal contrasts (in biomass)
- Nonstationarity could be achieved by setting some contrasts to zero
    - Reversible jump
    - L1 prior (Laplace / double exponential) a la the Lasso
    - Generalized Pareto extends the Laplace prior based on extensive work on properties of shrinkage priors (Carvalho et al (2010), Tansey et al. (2016), Taddy (2013))
        - Looks like Laplace prior but with fatter tails
- Could consider first-order (piecewise constant model), second-order (piecewise linear), third-order (piecewise quadratic) contrasts

Application 2: Generalized Pareto / Trend filtering

- Marginalized model (third order)

$$b_t \sim \mathrm{GenPar}(3b_{t-1} - 3b_{t-2} + b_{t-3}, \psi, \sigma)$$

  - Sparsity-inducing prior and modeling of contrasts produces very complicated and often very strong temporal dependence
  - Hard to make good MCMC proposals

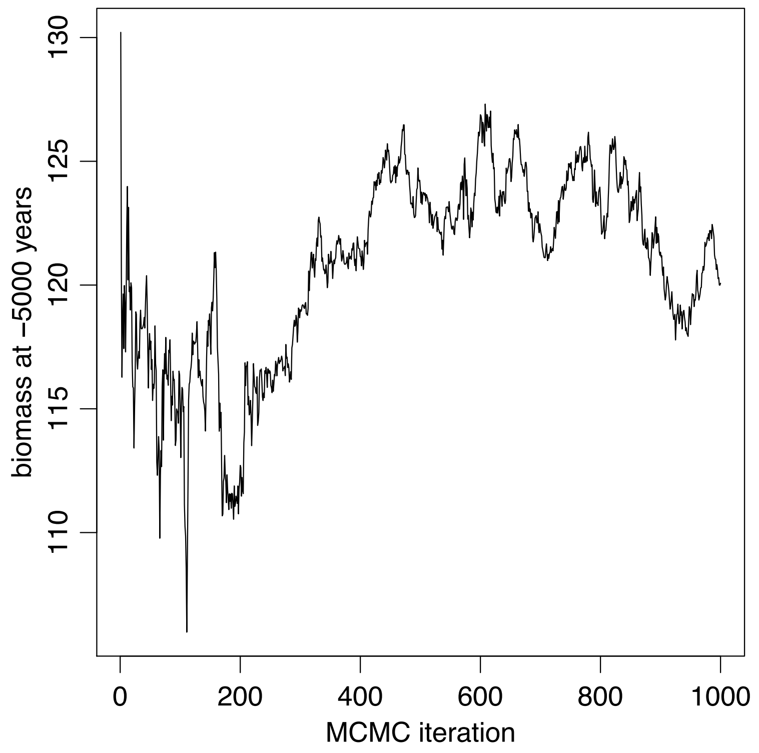
- Model (third order) with data augmentation

$$\begin{aligned} b_t &\sim \mathcal{N}(3b_{t-1} - 3b_{t-2} + b_{t-3}, \omega_t) \\ \omega_t &\sim \mathrm{Exp}(\lambda_t^2/2) \\ \lambda_t &\sim \mathrm{Ga}(\psi, \sigma) \end{aligned}$$

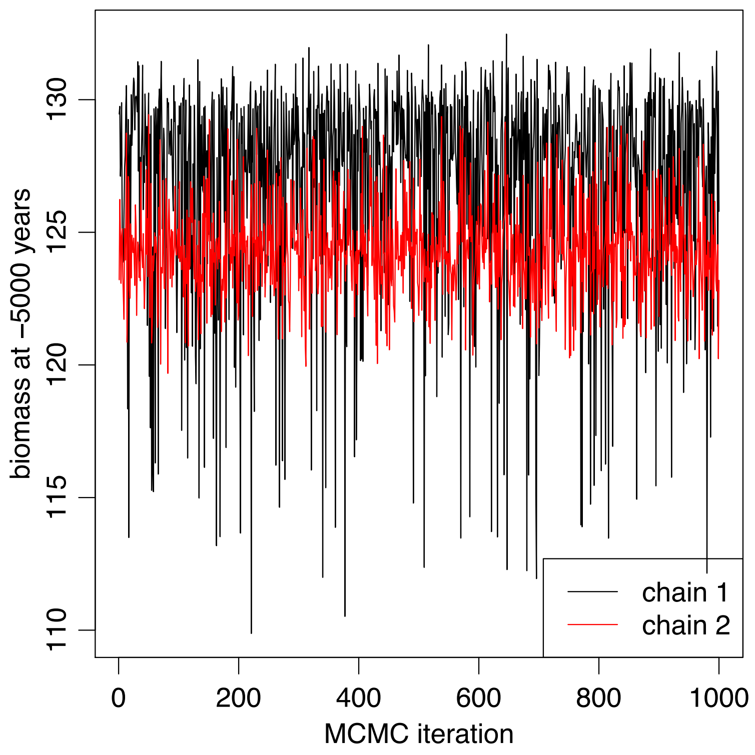
  - Now have normal prior for $b_{1:T}$ but no conjugacy so still hard to find good proposals
  - And we have additional hierarchical levels that can impede MCMC mixing

Application 2: MCMC performance

Mixing with data augmentation using default NIMBLE MCMC



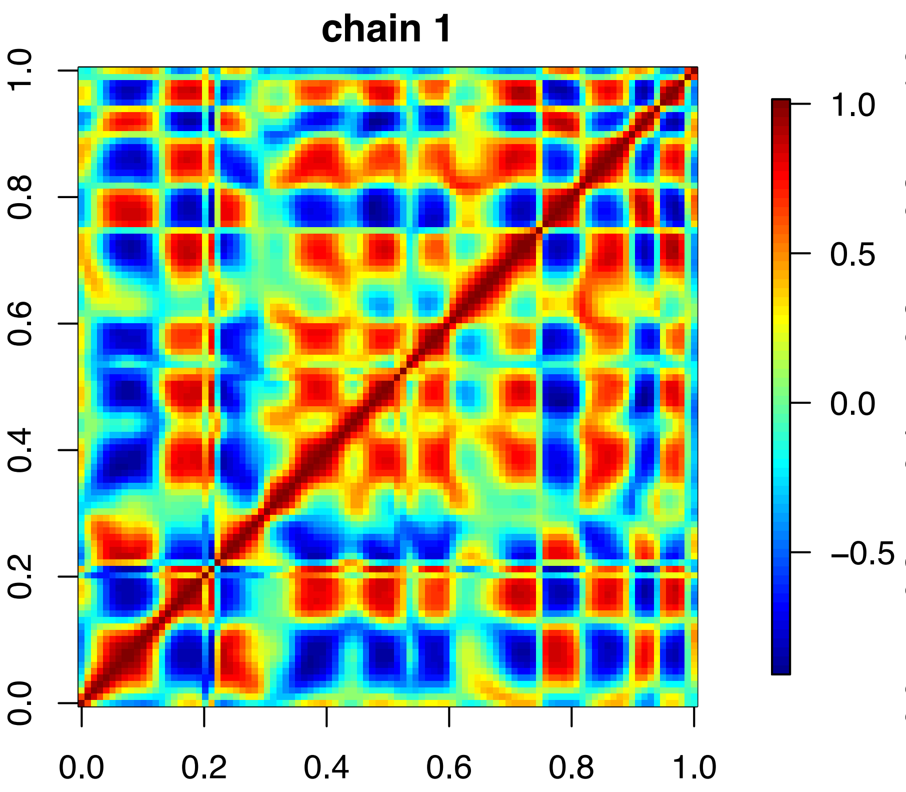
Mixing in marginalized model using HMC in Stan

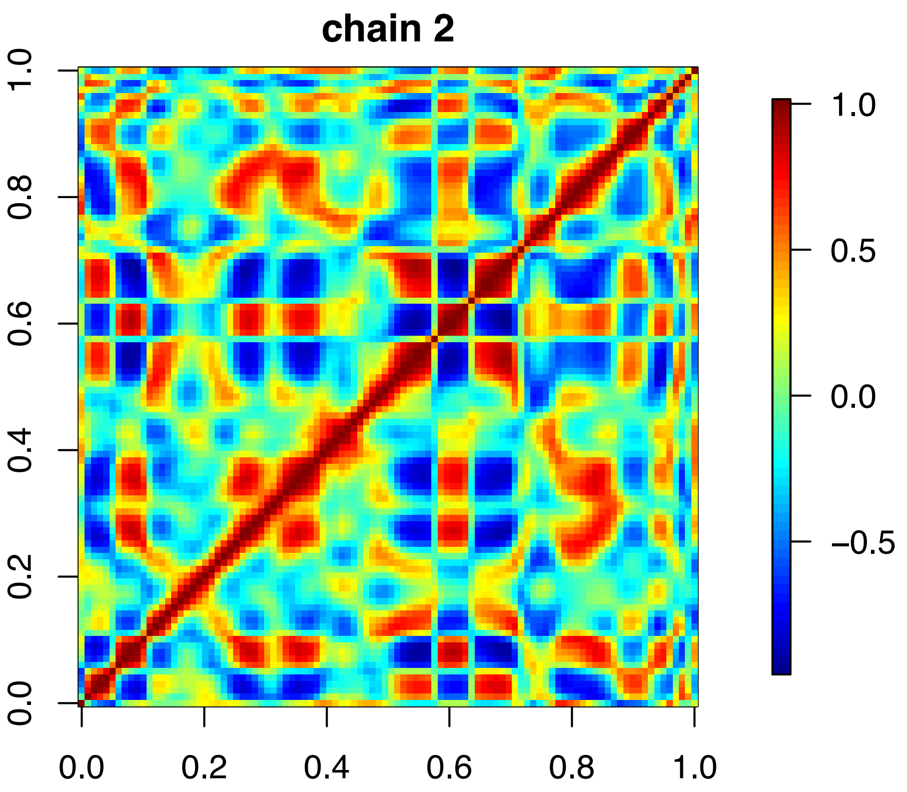


(recall non-differentiable spike at zero from generalized Pareto)

Application 2: MCMC performance (2)

Stan-based posterior correlations of biomass process values





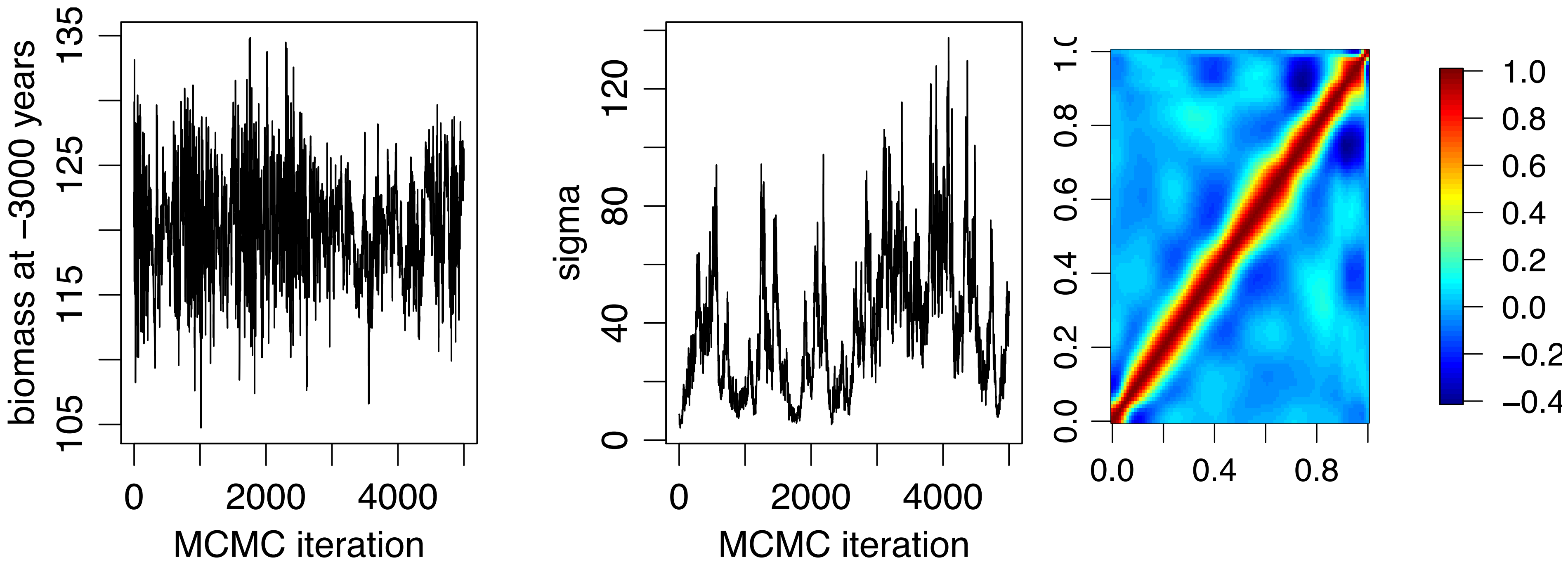
Application 2: Customized block sampling in NIMBLE

1. Use data augmentation with normal approximation to likelihood [y|b] at each point to provide approximately conjugate proposals for biomass process
    - Simple to approximate with mode and curvature of likelihood
2. Joint updates for $\omega_t, \lambda_t, b_{t-l:t+l}$: bivariate random walk for hyperparameters and approximate conjugate update for biomass process values
    - Joint updating of hyperparameters and process addresses cross-level dependency
    - Joint updating of multiple biomass values addresses temporal dependency
    - Local neighborhood updates for biomass reduce computation and avoid high-dimensional approximate conjugacy

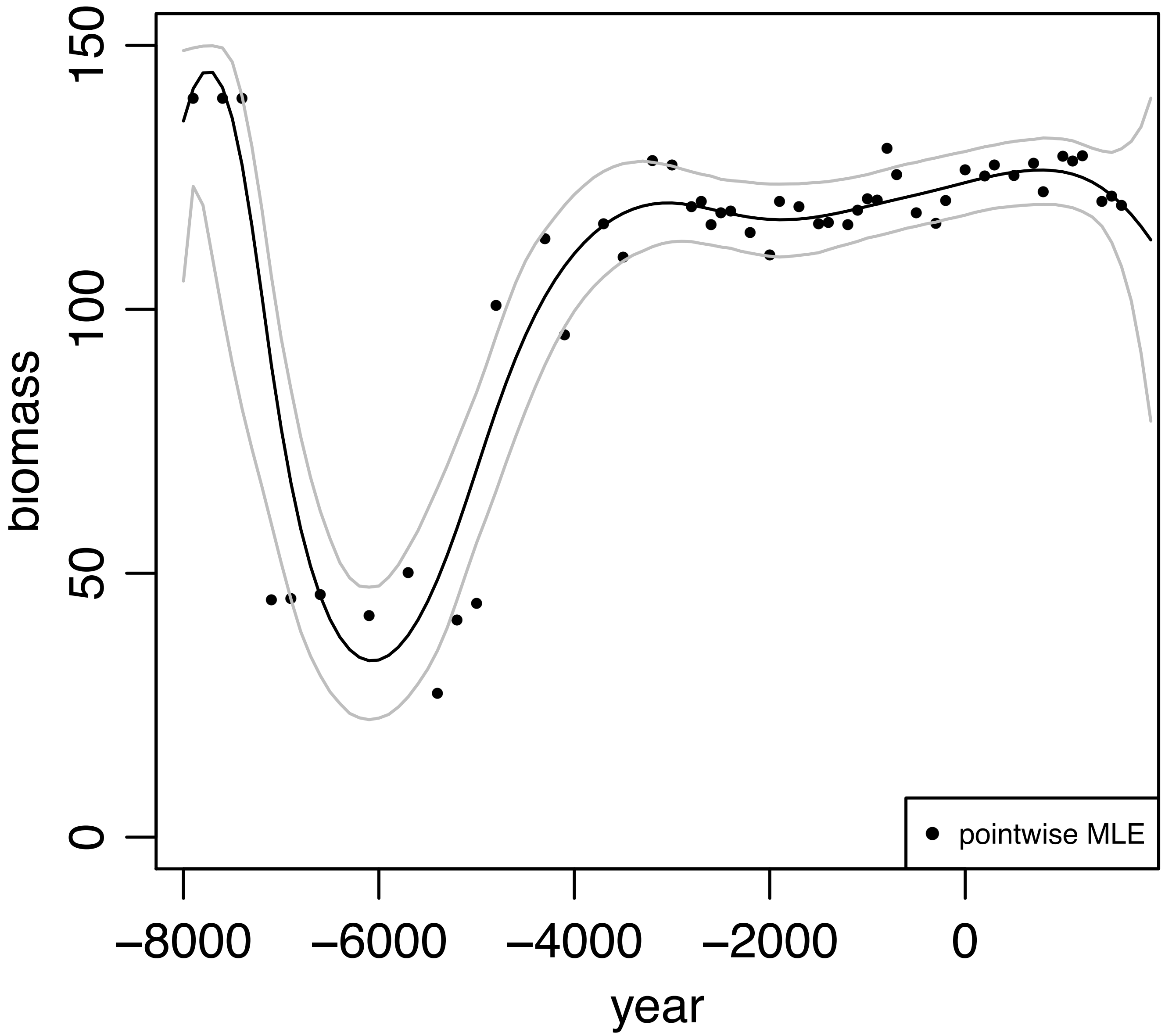
Sampling done in NIMBLE using a user-defined sampler, combined with standard samplers for other model parameters.

Application 2: Customized MCMC performance

Mixing with data augmentation using customized NIMBLE MCMC



Application 2: Initial results



Concluding thoughts

- The spatio(-temporal) dependence we need for smoothing/prediction can greatly affect algorithm performance.
- Blocked sampling can address dependence but good proposals can be hard to find, particularly with:
  - non-conjugate models and
  - dependence across model levels.
- Even with algorithm advances, computational limitations still greatly limit our ability to fit rich model structures.
- NIMBLE provides a platform for
  - customizing algorithms for particular models and
  - developing general-purpose algorithms for hierarchical models.

PalEON Acknowledgements

- Pollen-biomass Collaborators: Ann Raiho, Jason McLachlan (Notre Dame Biology)
- PalEON investigators: Jason McLachlan (Notre Dame, PI), Mike Dietze (Boston U.), Andrew Finley (Michigan State), Amy Hessl (West Virginia), Phil Higuera (Idaho), Mevin Hooten (USGS/Colorado State), Steve Jackson (USGS/Arizona), Dave Moore (Arizona), Neil Pederson (Harvard Forest), Jack Williams (Wisconsin), Jun Zhu (Wisconsin)
- NSF Macrosystems Program

NIMBLE Acknowledgements

NIMBLE development team:

• Perry de Valpine (PI)	UC Berkeley Environmental Science, Policy and Management (ESPM)
• Daniel Turek	Williams College
• Nick Michaud	UC Berkeley Statistics and ESPM
• Fritz Obermeyer	UC Berkeley Statistics and ESPM
• Duncan Temple Lang	UC Davis Statistics

- and various development team alumni

NIMBLE can be installed from CRAN in the usual way for an R package, and a full website with link to the User Manual is at http://r-nimble.org.

References

- Albert, James H., and Siddhartha Chib. "Bayesian analysis of binary and polychotomous response data." Journal of the American statistical Association 88.422 (1993): 669-679.
- Carvalho, Carlos M., Nicholas G. Polson, and James G. Scott. "The horseshoe estimator for sparse signals." Biometrika 97.2 (2010): 465-480.
- de Valpine, Perry, et al. "Programming with models: writing statistical algorithms for general model structures with NIMBLE." Journal of Computational and Graphical Statistics 26.2 (2017): 403-413.
- Lindgren, Finn, Håvard Rue, and Johan Lindström. "An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73.4 (2011): 423-498.
- McCulloch, Robert, and Peter E. Rossi. "An exact likelihood analysis of the multinomial probit model." Journal of Econometrics 64.1 (1994): 207-240.
- Taddy, Matt. "Multinomial inverse regression for text analysis." Journal of the American Statistical Association 108.503 (2013): 755-770.
- Tansey, Wesley, Athey, A., Reinhart, A., and Scott, J. G. "Multiscale spatial density smoothing: an application to large-scale radiological survey and anomaly detection." Journal of the American Statistical Association just-accepted (2017).