# Using GPUs to improve the computational efficiency of Gaussian process models

Colin Rundel

Duke University

February 27, 2014

1 Background

2 Migratory Bird Spatial Assignment Model

3 Speciated $PM_{2.5}$ Modeling

4 Related and Future Work

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample?

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Evaluating the likelihood?

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Evaluating the likelihood? - Invert $\Sigma$

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Evaluating the likelihood? - Invert $\Sigma$ - $\mathcal{O}\left(n^3\right)$!

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Evaluating the likelihood? - Invert $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Fitting a model?

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Evaluating the likelihood? - Invert $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Fitting a model? - Update $\Sigma$

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Evaluating the likelihood? - Invert $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Fitting a model? - Update $\Sigma$ - $\mathcal{O}\left(n^2\right)$

# The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

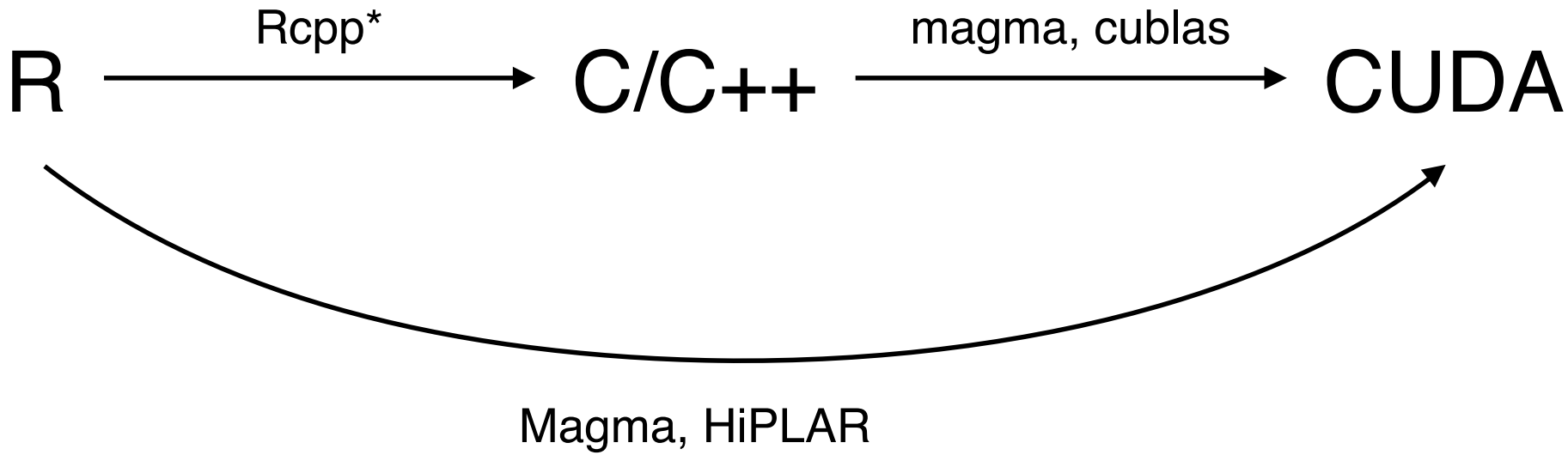
MVN Density:

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Drawing a sample? - Choleksy of $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Evaluating the likelihood? - Invert $\Sigma$ - $\mathcal{O}\left(n^3\right)$!
- Fitting a model? - Update $\Sigma$ - $\mathcal{O}\left(n^2\right)$

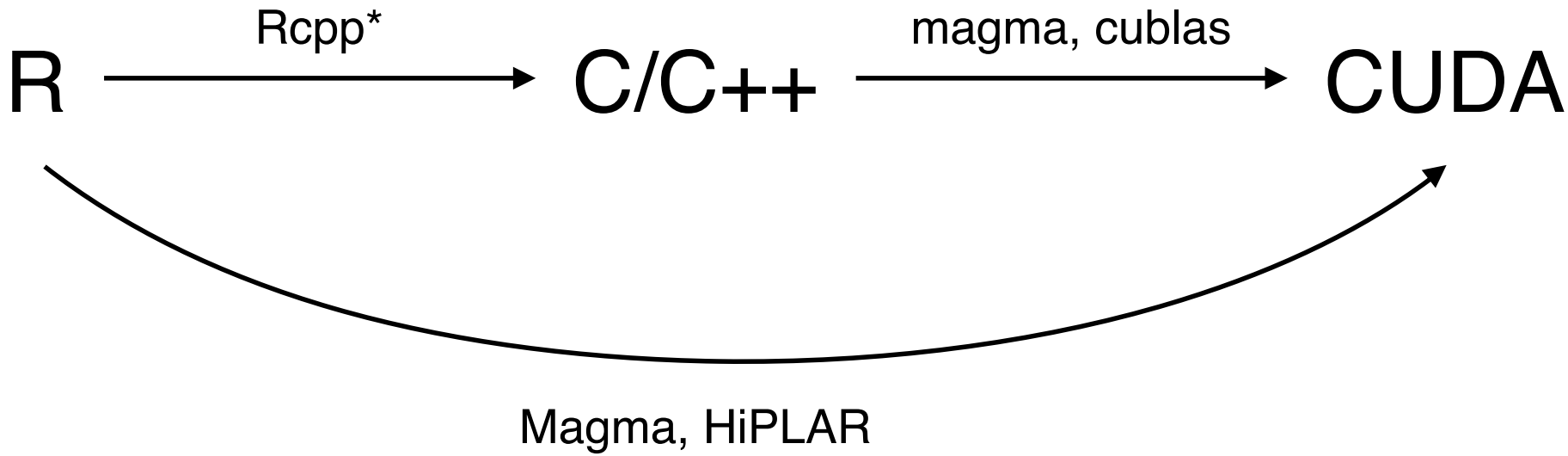
I don't have any clever solutions for these problems, but there are very smart people out there who are working on them.

# Tools and Optimization




Regardless of tools or workflow, measuring / profiling performance is critical.

# Tools and Optimization




Regardless of tools or workflow, measuring / profiling performance is critical.

"*Data or it didn't happen.*"

1 Background

2 Migratory Bird Spatial Assignment Model

3 Speciated $PM_{2.5}$ Modeling

4 Related and Future Work

# Background

Using intrinsic markers (genetic and isotopic signals) for the purpose of inferring migratory connectivity.

- Existing methods are too coarse for most applications
- Large amounts of data are available ( $>$150,000 feather samples from $>$500 species)
- Genetic assignment methods are based on Wasser, et al. (2004)
- Isotopic assignment methods are based on Wunder, et al. (2005)

Rundel, C.W., *et al.* (2013) Novel statistical methods for integrating genetic and stable isotopic data to infer individual-level migratory connectivity. *Molecular Ecology* 22 (16).

# Data - DNA microsatellites and $\delta^2\mathrm{H}$

Hermit Thrush (*Catharus guttatus*)

- 138 individuals
- 14 locations
- 6 loci
- 9-27 alleles / locus


Wilson's Warbler (*Wilsonia pusilla*)

- 163 individuals
- 8 locations
- 9 loci
- 15-31 alleles / locus

# Allele Frequency Model

For the allele $i$, from locus $l$, at location $k$

$$\boldsymbol{y}_{\cdot lk} | \boldsymbol{\Theta} \sim \text{Mult}\left(\textstyle\sum_i y_{ilk},\ \boldsymbol{f}_{\cdot lk}\right)$$

$$f_{ilk} = \frac{\exp(\Theta_{ilk})}{\sum_i \exp(\Theta_{ilk})}$$

$$\boldsymbol{\Theta}_{il} | \boldsymbol{\alpha}, \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_{il},\ \boldsymbol{\Sigma})$$

$$\{\Sigma\}_{i,j} = \alpha_0 \exp\left( -(d_{i,j}/\alpha_1)^{\alpha_2} \right) + \alpha_3 I_{d_{i,j}=0}$$

# Genetic Assignment Model

Assignment model:

$$P(S_G|\boldsymbol{f}, k) = \prod_l P(i_l, j_l|\boldsymbol{f}, k)$$

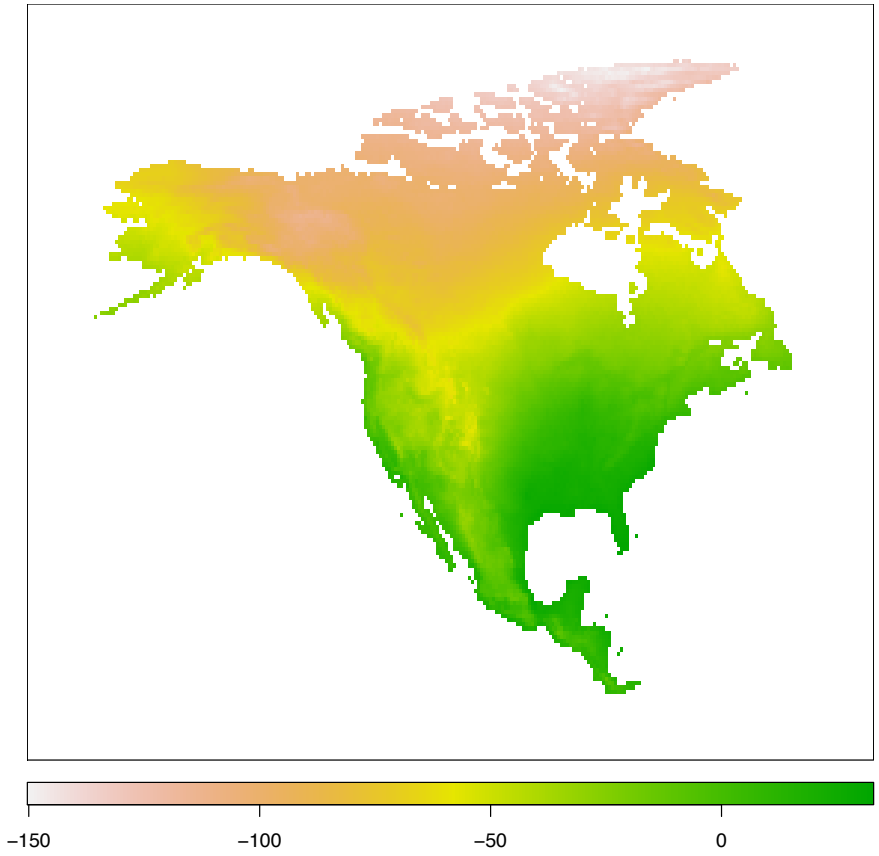
$$P(i_l, j_l|\boldsymbol{f}, k) = \begin{cases} \gamma P(i_l|\boldsymbol{f}, k) + (1-\gamma)P(i_l|\tilde{\boldsymbol{f}}, k)^2 & \text{if } i = j \\ (1-\gamma)P(i_l|\boldsymbol{f}, k)P(j_l|\boldsymbol{f}, k) & \text{if } i \neq j \end{cases}$$

$$P(i_k|\boldsymbol{f}, k) = (1-\delta)f_{lik} + \delta/m_l$$

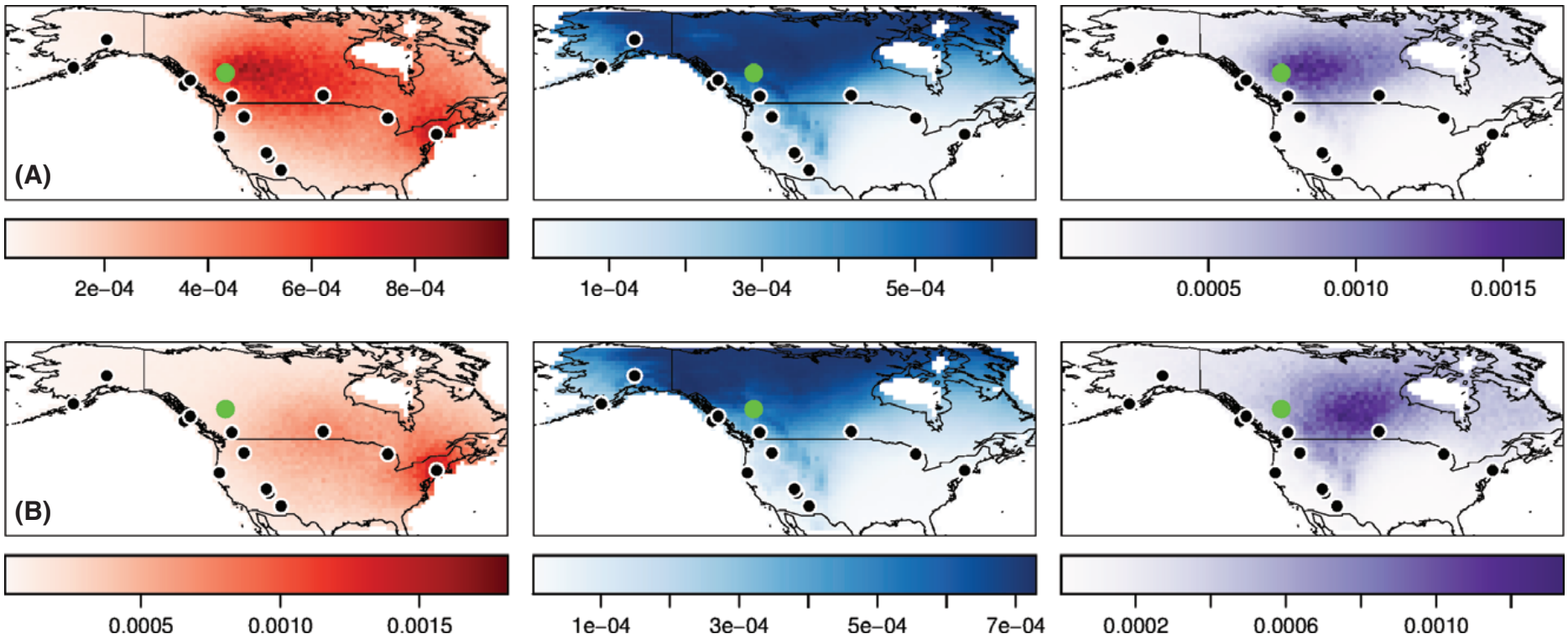
$\delta$ – Genotyping error rate $\quad$ $\gamma$ – Single amplification rate

# Isotope Model

$$S_I | k, \tilde{\boldsymbol{p}}, \omega, \rho, \tau^2 \sim \mathrm{N}(\omega + \rho\, \tilde{p}_k, \tau^2)$$



# Combined Model

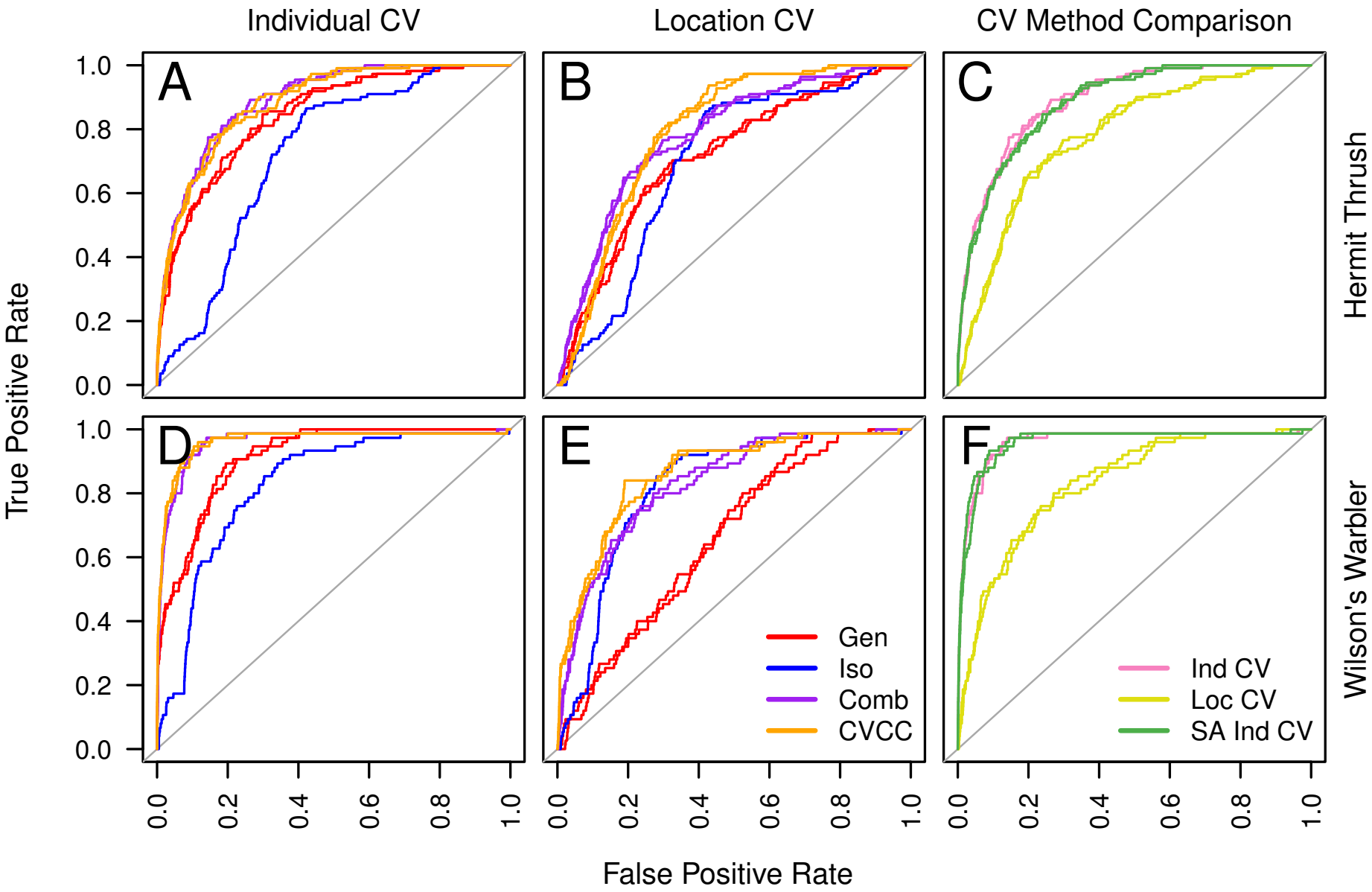


# Implementation

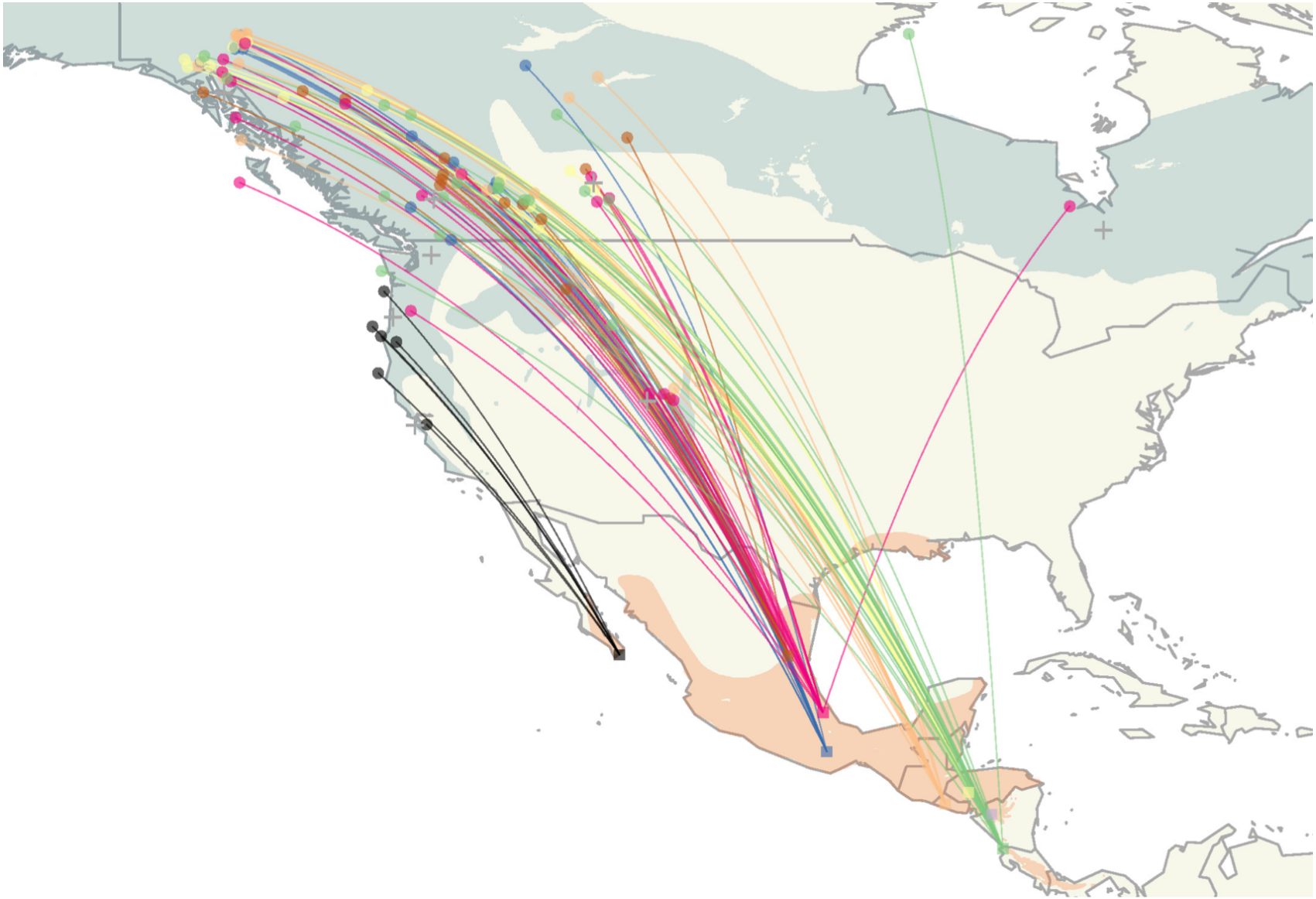
Model fitting is done via MCMC (MH within Gibbs)

- Original implementation in pure C++ with minimal dependencies (Wasser, et al. (2004))
- Rewritten using R / C++ via Rcpp(Armadillo)
  - Code closer to matrix notation (and R)
  - Transparent use of high performance LAPACK implementations
  - R Package - isoscatR - `https://github.com/rundel/isoscatR`
- Model fitting performance is quite good
  - 300,000 iterations in $\sim$ 5.5 minutes
- Bottleneck is sampling posterior predictive
  - 1,000 iterations in $\sim$ 30 minutes

# Model Assessment



# Migratory Connectivity



# Prediction details

Why is the prediction slow?

# Prediction details

Why is the prediction slow?

Predicting allele frequencies for Hermit thrush at 3318 novel locations.

To do so we sample from:

$$\boldsymbol{\Theta}_p|\boldsymbol{\Theta}_m \sim \mathcal{N}(\boldsymbol{\mu}_p + \boldsymbol{\Sigma}_{pm}\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{\Theta}_m - \boldsymbol{\mu}_m),\ \boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_{pm}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\Sigma}_{mp})$$

# Prediction details

Why is the prediction slow?

Predicting allele frequencies for Hermit thrush at 3318 novel locations.

To do so we sample from:

$$\boldsymbol{\Theta}_p|\boldsymbol{\Theta}_m \sim \mathcal{N}(\boldsymbol{\mu}_p + \boldsymbol{\Sigma}_{pm}\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{\Theta}_m - \boldsymbol{\mu}_m),\ \boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_{pm}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\Sigma}_{mp})$$

## Algorithm steps

1. Calculate $\boldsymbol{\Sigma}_{pm}$, $\boldsymbol{\Sigma}_p$, and $\boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_{pm}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\Sigma}_{mp}$
2. Calculate Chol$(\boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_{pm}\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\Sigma}_{mp})$
3. Sample from MVN
4. Calculate allele frequencies

# Posterior predictive sampling timings

| | Step | CPU (secs) | CPU+GPU (secs) | Rel. Performance |
|---|---|---|---|---|
| 1. | Covariances | 1.080 | 0.046 | 23.0 |
| 2. | Cholesky | 0.467 | 0.208 | 2.3 |
| 3. | Samples | 0.049 | 0.052 | 0.9 |
| 4. | Allele Freq | 0.129 | 0.127 | 1.0 |
| | Total | 1.732 | 0.465 | 3.7 |

# Posterior predictive sampling timings

| | Step | CPU (secs) | CPU+GPU (secs) | Rel. Performance |
|---|---|---|---|---|
| 1. | Covariances | 1.080 | 0.046 | 23.0 |
| 2. | Cholesky | 0.467 | 0.208 | 2.3 |
| 3. | Samples | 0.049 | 0.052 | 0.9 |
| 4. | Allele Freq | 0.129 | 0.127 | 1.0 |
| | Total | 1.732 | 0.465 | 3.7 |

Total run time:

- CPU - 28.9 minutes
- CPU+GPU - 7.8 minutes

# Posterior predictive sampling timings

| | Step | CPU (secs) | CPU+GPU (secs) | Rel. Performance |
|---|---|---|---|---|
| 1. | Covariances | 1.080 | 0.046 | 23.0 |
| 2. | Cholesky | 0.467 | 0.208 | 2.3 |
| 3. | Samples | 0.049 | 0.052 | 0.9 |
| 4. | Allele Freq | 0.129 | 0.127 | 1.0 |
| | Total | 1.732 | 0.465 | 3.7 |

Total run time:

- CPU - 28.9 minutes
- CPU+GPU - 7.8 minutes

$\times$ 166 for Hermit Thrush
179 for Wilson's Warbler

# Lessons

- Relatively small changes in one function resulted in 3 - 4x improvement
  - Cross validation results in two days instead of a week
  - 1-2 weeks of implementation, 1 week of tweaking / testing
  - Started with Cholesky, other optimizations followed

# Lessons

- Relatively small changes in one function resulted in 3 - 4x improvement
  - Cross validation results in two days instead of a week
  - 1-2 weeks of implementation, 1 week of tweaking / testing
  - Started with Cholesky, other optimizations followed
- Issues
  - External library dependency makes package development (much) more complicated
  - Additional code verbosity and complexity

# Improving Covariance Calculations

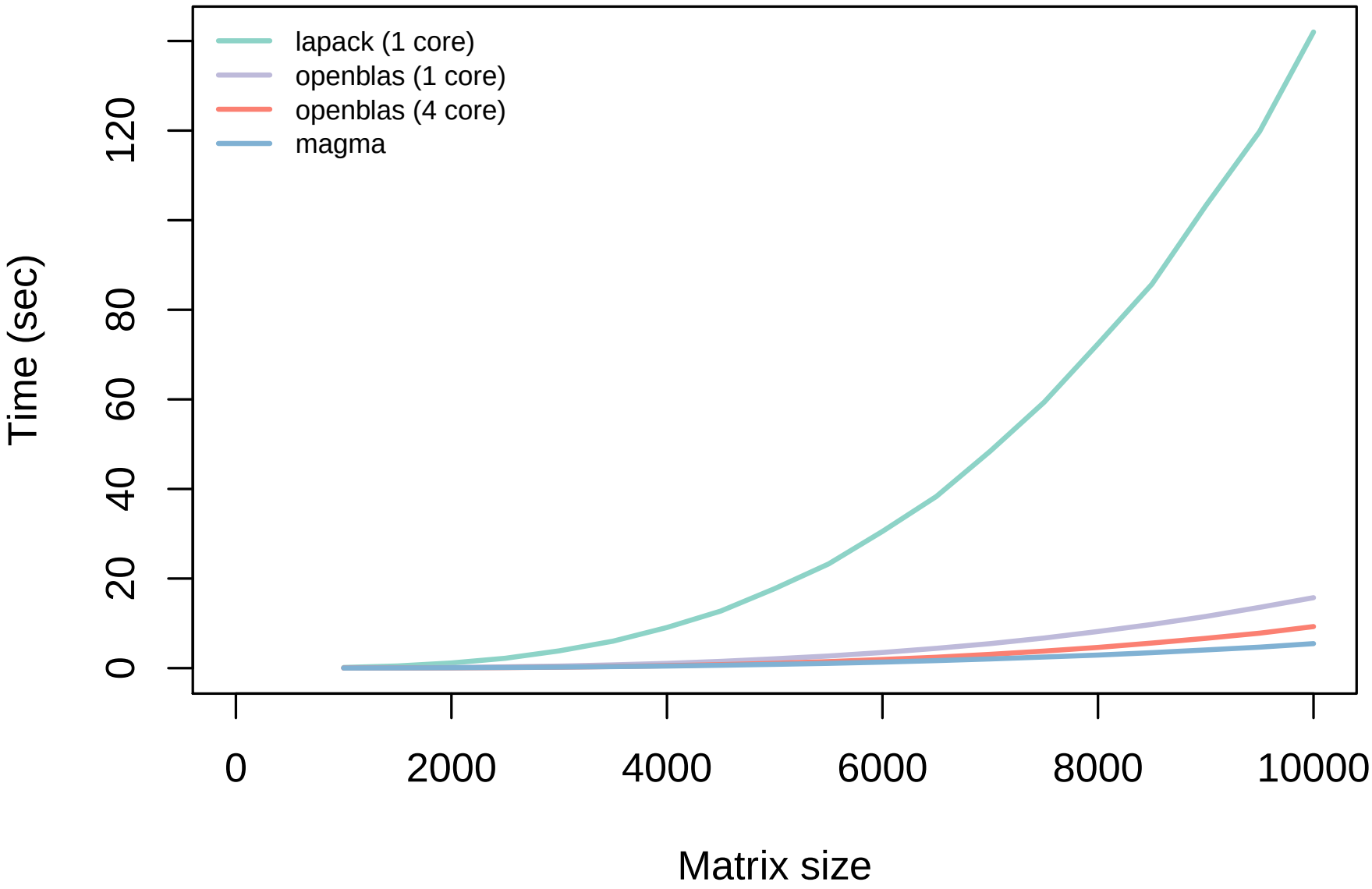
Covariance is assumed to be stationary and isotropic

- Elements of the covariance matrix can be calculated independently
- Small scale "embarrassingly parallel"
- Implementation is straight forward
  (if we don't worry about things like symmetry)

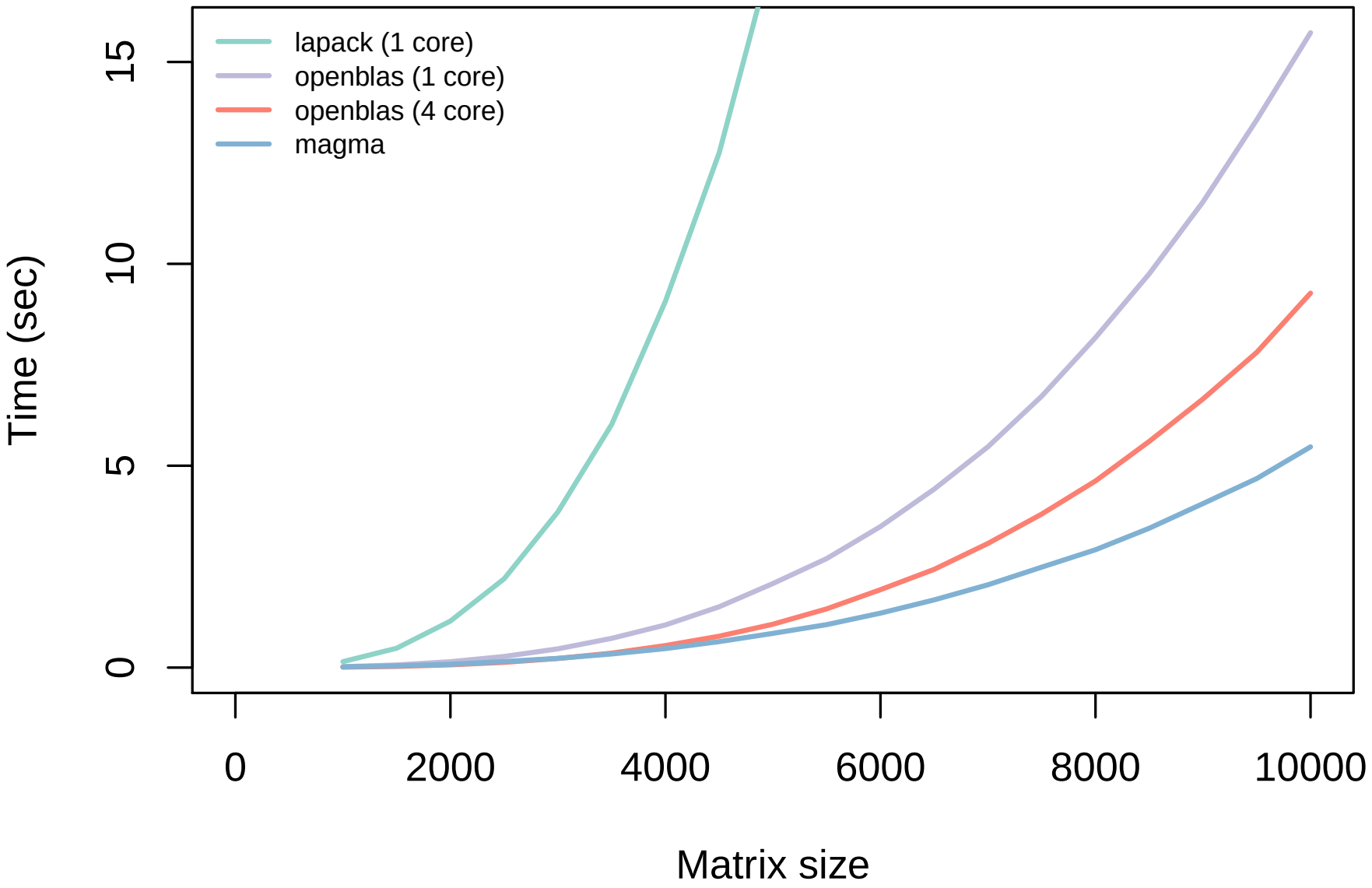
```
__global__ void powered_exponential_cov_kernel(double* dist, double* cov,
                                               const int nm, const double sigma2,
                                               const double phi, const double kappa)
{
    int n_threads = gridDim.x * blockDim.x;
    int pos = blockDim.x * blockIdx.x + threadIdx.x;

    for (int i = pos; i < nm; i += n_threads)
    {
        cov[i] += sigma2 * exp( -pow(dist[i] * phi, kappa) );
    }
}
```

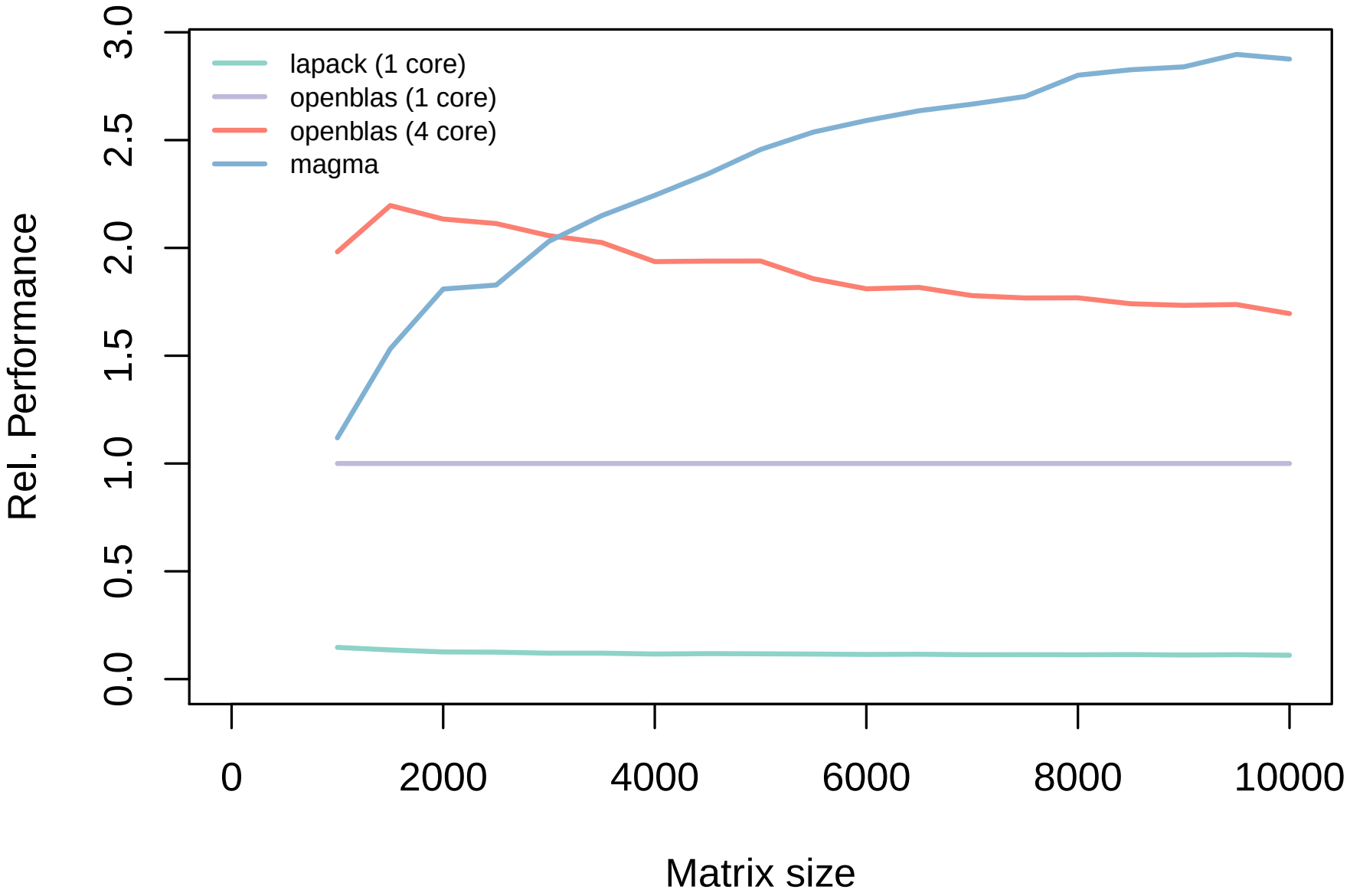
# Improving Cholesky



# Improving Cholesky



# Improving Cholesky



# Building core tools

Common set of (expensive) tasks for GP models

- Covariance calculation
- Cholesky of Cov.
- Inverse of Cov.

Goal is to make performing these tasks on a GPU as painless as possible and allow interoperability with GPU (magma, CUBLAS) and CPU (Armadillo) libraries.

- GPU matrix class
- Modern resource management (RAII)
- Simple translation to and from

R Package - RcppGP - `https://github.com/rundel/RcppGP`

# CPU vs GPU code

```
arma::mat prop_Sigma = arma::exp(-prop_phi * d_CIF);
arma::mat prop_Sigma_U = arma::chol(prop_Sigma);

double prop_Sigma_log_det = 2*arma::accu(arma::log(prop_Sigma_U.diag()));

arma::mat prop_Sigma_U_inv = arma::inv(arma::trimatu(prop_Sigma_U));
arma::mat prop_Sigma_inv = prop_Sigma_U_inv * prop_Sigma_U_inv.t();
```

```
exponential_cov_gpu(d_CIF_gpu.mat, cov_gpu.mat, nr_CIF, nr_CIF, 1.0, prop_phi, 64);
cov_gpu.chol('L',false);

double prop_Sigma_log_det = 2*arma::accu(arma::log(cov_gpu.get_mat().diag()));

cov_gpu.inv_chol('L',true);
arma::mat prop_Sigma_inv = cov_gpu.get_mat();
```

Back

1 Background

2 Migratory Bird Spatial Assignment Model

3 Speciated $PM_{2.5}$ Modeling

4 Related and Future Work

# Background

Fine particulate matter ($PM_{2.5}$ ) is an EPA regulated air pollutant linked to a variety of adverse health effects

- Classified based on particle size ($< 2.5$ $\mu$m diameter)
- Major species: Sulfate, Nitrate, Ammonium, Soil, Carbon.
- Minor species: trace elements (K, Mg, Ca), heavy metals (Cu, Fe), etc.
- Complex spatio-temporal dependence between species

# Data

Speciated $PM_{2.5}$ Sources

- Chemical Speciation Network (CSN) - 221 stations
- Interagency Monitoring of Protected Visual Environments (IMPROVE) - 172 stations

Total $PM_{2.5}$ Sources

- Federal Reference Method (FRM) - 949 stations

Model Output

- Community Multi-scale Air Quality (CMAQ) - 12 km grid

Data Issues

- Monitoring frequency
- Total vs Sum of Species

# Model Details

For each species and network,

$$C_t^i(\boldsymbol{s}) = Z_t^i(\boldsymbol{s}) + \epsilon_{C,t}^i(\boldsymbol{s}) \qquad Z_t^i(\boldsymbol{s}) = \max\left(0,\ \widetilde{Z}_t^i(\boldsymbol{s})\right)$$

$$I_t^i(\boldsymbol{s}) = Z_t^i(\boldsymbol{s}) + \epsilon_{I,t}^i(\boldsymbol{s}) \qquad \widetilde{Z}_t^i(\boldsymbol{s}) = \beta_{0,t}^i + \beta_{0,t}^i(\boldsymbol{s}) + \beta_{1,t}^i\, Q_t^i(B_{\boldsymbol{s}})$$

# Model Details

For each species and network,

$$C_t^i(\boldsymbol{s}) = Z_t^i(\boldsymbol{s}) + \epsilon_{C,t}^i(\boldsymbol{s}) \qquad Z_t^i(\boldsymbol{s}) = \max\left(0,\ \widetilde{Z}_t^i(\boldsymbol{s})\right)$$

$$I_t^i(\boldsymbol{s}) = Z_t^i(\boldsymbol{s}) + \epsilon_{I,t}^i(\boldsymbol{s}) \qquad \widetilde{Z}_t^i(\boldsymbol{s}) = \beta_{0,t}^i + \beta_{0,t}^i(\boldsymbol{s}) + \beta_{1,t}^i\, Q_t^i(B_{\boldsymbol{s}})$$

For total PM,

$$C_t^{tot}(\boldsymbol{s}) = Z_t^{tot}(\boldsymbol{s}) + \epsilon_{C,t}^{tot}(\boldsymbol{s})$$

$$I_t^{tot}(\boldsymbol{s}) = Z_t^{tot}(\boldsymbol{s}) + \epsilon_{I,t}^{tot}(\boldsymbol{s})$$

$$F_t^{tot}(\boldsymbol{s}) = Z_t^{tot}(\boldsymbol{s}) + \epsilon_{F,t}^{tot}(\boldsymbol{s})$$

$$Z_t^{tot}(\boldsymbol{s}) = \sum_{i=1}^{5} Z_t^i(\boldsymbol{s}) + Z_t^o(\boldsymbol{s})$$

# Model Details

For each species and network,

$$C_t^i(\boldsymbol{s}) = Z_t^i(\boldsymbol{s}) + \epsilon_{C,t}^i(\boldsymbol{s}) \qquad Z_t^i(\boldsymbol{s}) = \max\left(0,\ \widetilde{Z}_t^i(\boldsymbol{s})\right)$$

$$I_t^i(\boldsymbol{s}) = Z_t^i(\boldsymbol{s}) + \epsilon_{I,t}^i(\boldsymbol{s}) \qquad \widetilde{Z}_t^i(\boldsymbol{s}) = \beta_{0,t}^i + \beta_{0,t}^i(\boldsymbol{s}) + \beta_{1,t}^i\, Q_t^i(B_{\boldsymbol{s}})$$

For total PM,

$$C_t^{tot}(\boldsymbol{s}) = Z_t^{tot}(\boldsymbol{s}) + \epsilon_{C,t}^{tot}(\boldsymbol{s})$$

$$I_t^{tot}(\boldsymbol{s}) = Z_t^{tot}(\boldsymbol{s}) + \epsilon_{I,t}^{tot}(\boldsymbol{s})$$

$$F_t^{tot}(\boldsymbol{s}) = Z_t^{tot}(\boldsymbol{s}) + \epsilon_{F,t}^{tot}(\boldsymbol{s})$$

$$Z_t^{tot}(\boldsymbol{s}) = \sum_{i=1}^{5} Z_t^i(\boldsymbol{s}) + Z_t^o(\boldsymbol{s})$$

$$Z_t^o(s) = \max\left(0,\ \widetilde{Z}_t^o(s)\right)$$

$$\widetilde{Z}_t^o(\boldsymbol{s}) = \beta_{0,t}^o + \beta_{0,t}^o(\boldsymbol{s}) + \beta_{1,t}^o\, Q_t^o(B_{\boldsymbol{s}})$$

# Model Details Cont.

Spatial dependence enters the model through the $\beta^i_{0,t}(s)$ and $\beta^o_{0,t}(s)$ parameters.

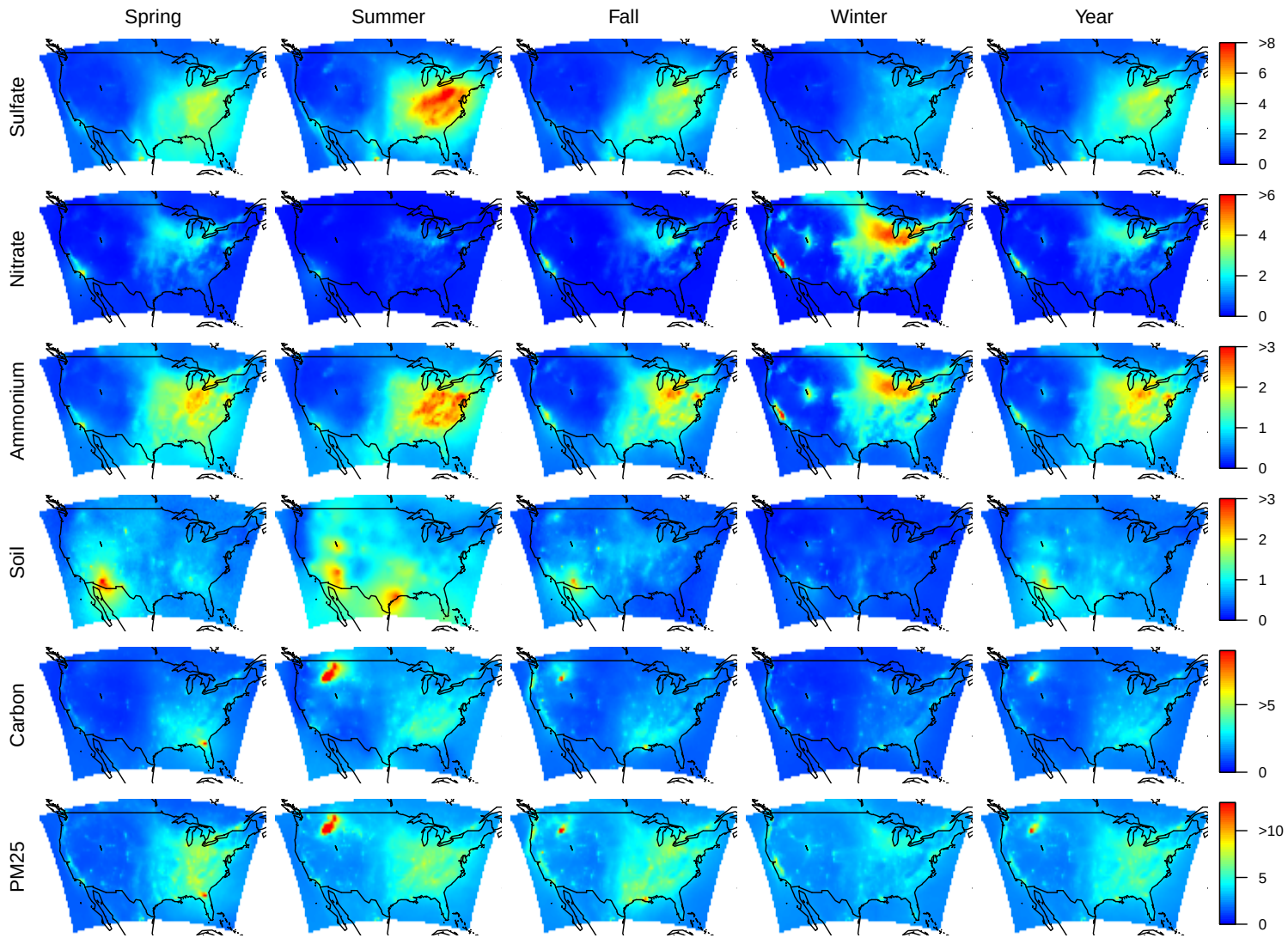
$$\beta^i_{0,t}(\boldsymbol{s}) = \sigma^i_t \; w^i_t(\boldsymbol{s}) \qquad \beta^o_{0,t}(\boldsymbol{s}) = \eta_t \; w^o_t(\boldsymbol{s})$$

where $w^i_t(\boldsymbol{s})$ and $w^o_t(\boldsymbol{s})$ are zero mean, variance 1, Gaussian processes with exponential correlation given by

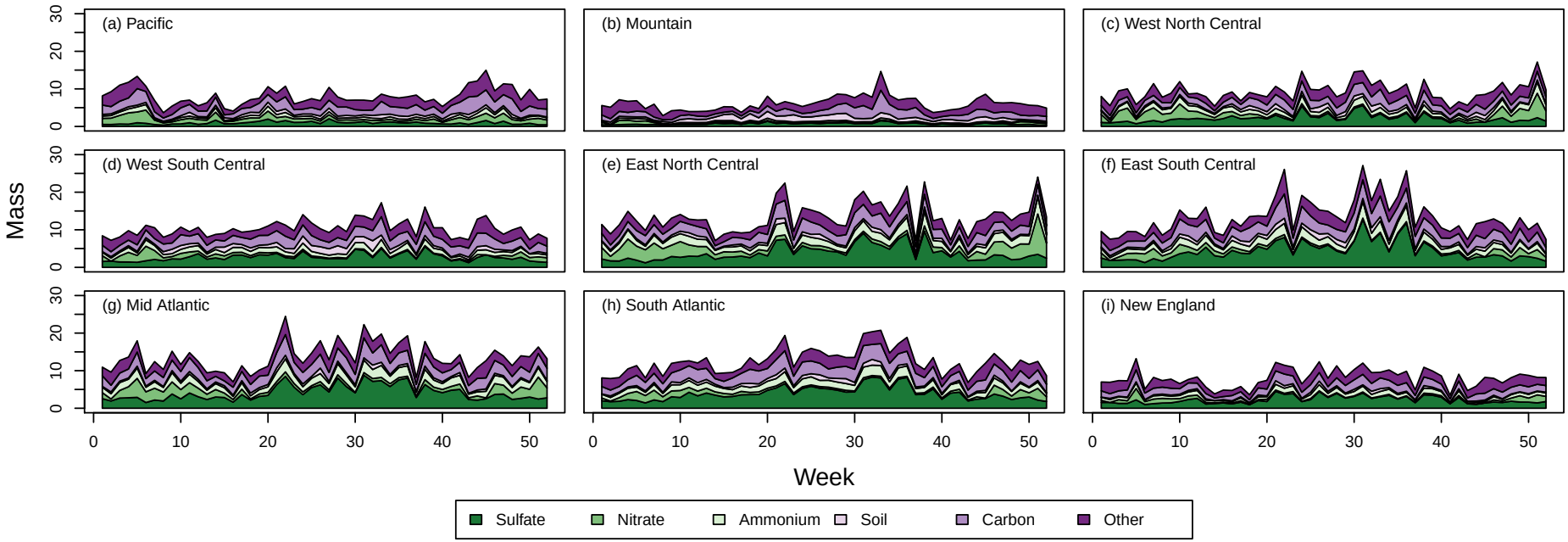
$$\text{corr}(w^i_t(\boldsymbol{s}), w^i_t(\boldsymbol{s}')) = \exp(-\phi^i_t|\boldsymbol{s} - \boldsymbol{s}'|)$$

$$\text{corr}(w^o_t(\boldsymbol{s}), w^o_t(\boldsymbol{s}')) = \exp(-\xi_t|\boldsymbol{s} - \boldsymbol{s}'|)$$

# Model results



# Model results



# Model fitting performance

| Parameter | CPU (secs) | CPU+GPU (secs) | Rel. Performance |
|---|---|---|---|
| $\beta_0,\ \beta_1$ | 0.00029 | 0.00030 | 0.97 |
| $\beta_0(s)$ | 0.09205 | 0.09132 | 1.00 |
| $\sigma^2, \eta^2$ | 0.00383 | 0.00385 | 0.99 |
| $\phi,\ \xi$ | 0.46084 | 0.25174 | 1.83 |
| $\tau_i^2,\ \tau_{tot}^2$ | 0.00003 | 0.00003 | 1.00 |
| Total | 0.55708 | 0.34729 | 1.60 |

# Run times

Total run time for model fitting (50,000 iterations):

- CPU - 7.7 hours
- CPU+GPU - 4.8 hours

$\times$ 52 weeks

# Run times

Total run time for model fitting (50,000 iterations):

- CPU - 7.7 hours
- CPU+GPU - 4.8 hours

$\times$ 52 weeks

Total run time for model prediction at 5950 locations (1,000 iterations):

- CPU - 7.2 hours
- CPU+GPU - 4.3 hours

$\times$ 52 weeks

# Run times

Total run time for model fitting (50,000 iterations):

- CPU - 7.7 hours
- CPU+GPU - 4.8 hours

$\times$ 52 weeks

Total run time for model prediction at 5950 locations (1,000 iterations):

- CPU - 7.2 hours
- CPU+GPU - 4.3 hours

$\times$ 52 weeks

One run takes about 775 hours total on CPU alone, 473 on CPU and GPU.

# Lessons

- Infrastructure makes a huge difference in development time
  - 1 hour to go from CPU implementation to CPU+GPU implementation
  - Code shown previously is 2/3 of the changes necessary Code
- In practice, easier to run CPU only code across more servers (configuration time / effort)
  - Not possible (or at least easy) for models variants that are not independent in time.
- Rcpp Attributes offer huge advantages in development and deployment
  - Simplifies external dependencies (locally)

1 Background

2 Migratory Bird Spatial Assignment Model

3 Speciated $PM_{2.5}$ Modeling

4 Related and Future Work

# GPUs and reduced rank GPs

For example the Gaussian Process,

$$Y(\boldsymbol{s}) = x'(\boldsymbol{s})\,\boldsymbol{\beta} + w(\boldsymbol{s}) + \epsilon(\boldsymbol{s}), \quad \epsilon(\boldsymbol{s}) \sim N(0,\ \tau^2 I_n)$$
$$w(\boldsymbol{s}) \sim N(0,\ \boldsymbol{C}(\boldsymbol{s})), \quad \boldsymbol{C}(\boldsymbol{s}, \boldsymbol{s}') = \sigma^2\,\rho(\boldsymbol{s}, \boldsymbol{s}'|\theta)$$

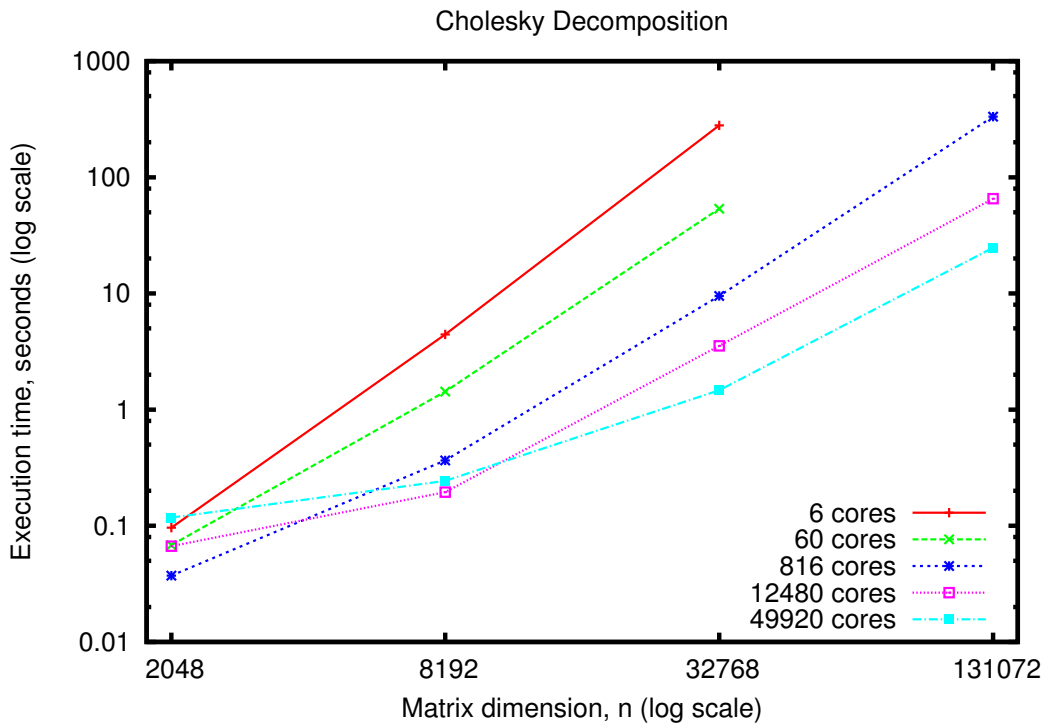
can be approximated using a Gaussian predictive process with knots at locations $\boldsymbol{s}^*$ where

$$Y(\boldsymbol{s}) = x'(\boldsymbol{s})\,\boldsymbol{\beta} + \widetilde{w}(\boldsymbol{s}) + \epsilon(\boldsymbol{s})$$
$$\widetilde{w}(\boldsymbol{s}) = \boldsymbol{C}(\boldsymbol{s}, \boldsymbol{s}^*)\,\boldsymbol{C}(\boldsymbol{s}^*)^{-1}\,w(\boldsymbol{s}^*).$$
$$w(\boldsymbol{s}^*) \sim N(0,\ \boldsymbol{C}(\boldsymbol{s}^*))$$

# Another Approach

bigGP is an R package written by Chris Paciorek, et al.

- Specialized implementation of LA operation for GPs
- Designed to run on large super computer clusters
- Uses both shared and distributed memory
- Able to fit models on the order of $n = 65k$ (32 GB Cov. matrix)



# Future Directions

- Refinement of RcppGP
  - Full Rcpp Attribute support
  - Transparent GPU to CPU failover
- Single vs. Multi-GPU algorithms
- Mixed precision methods
- NVBLAS
- Unified memory

# Acknowledgments

**Migratory Connectivity**

- John Novembre - UChicago
- Thomas Smith - UCLA
- Kristen Ruegg - UCLA, UCSC
- Center for Tropical Research, UCLA IoES

**Speciated $PM_{2.5}$**

- Alan Gelfand - Duke
- Dave Holland - EPA
- Erin Schliep - Duke

# Performance

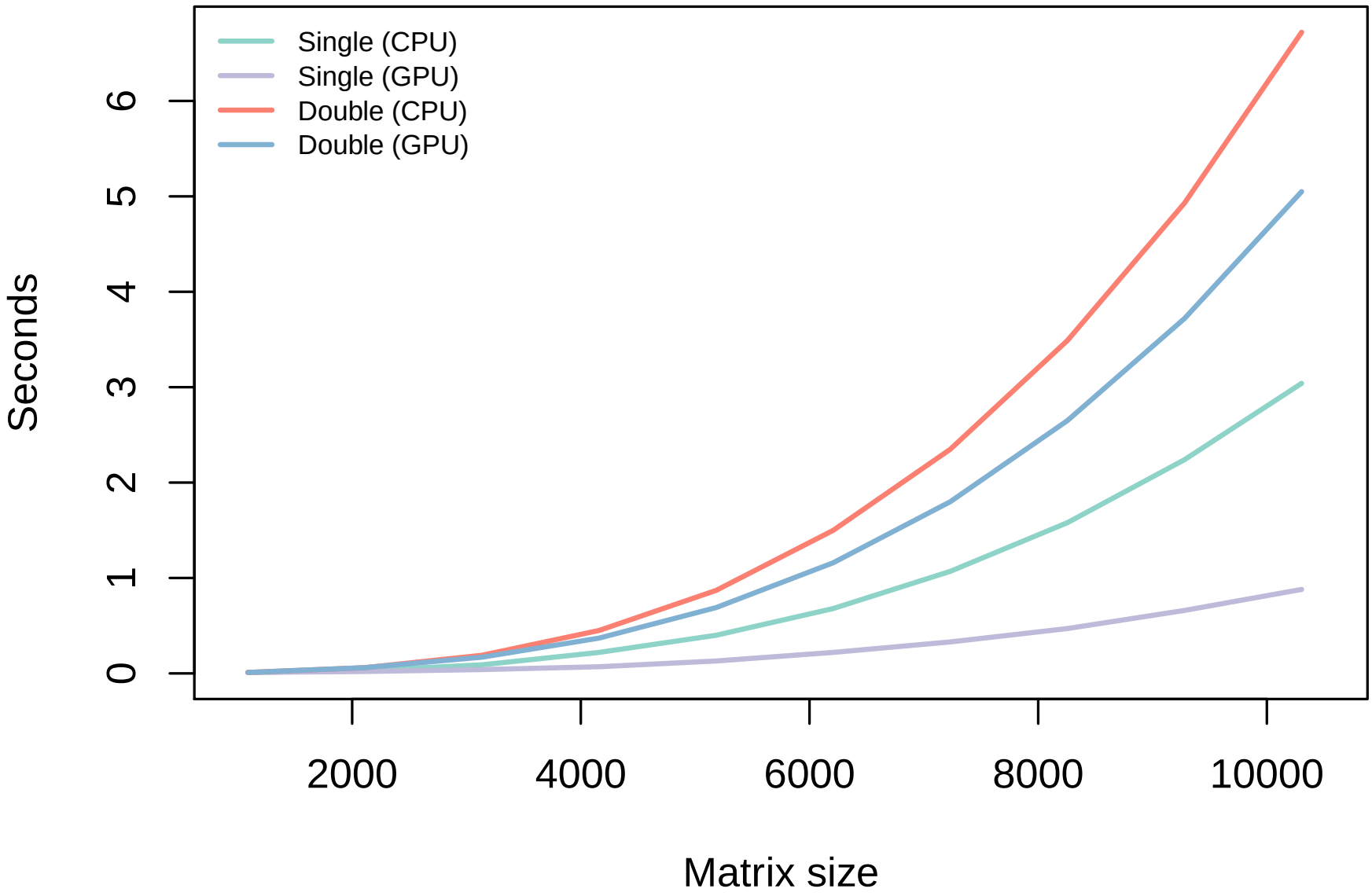
System Specs:

- 4 core Intel i5-2500K @ 3.30 GHz
- 16 GB DDR3 @ 1333 MHz
- GeForce GTX 460

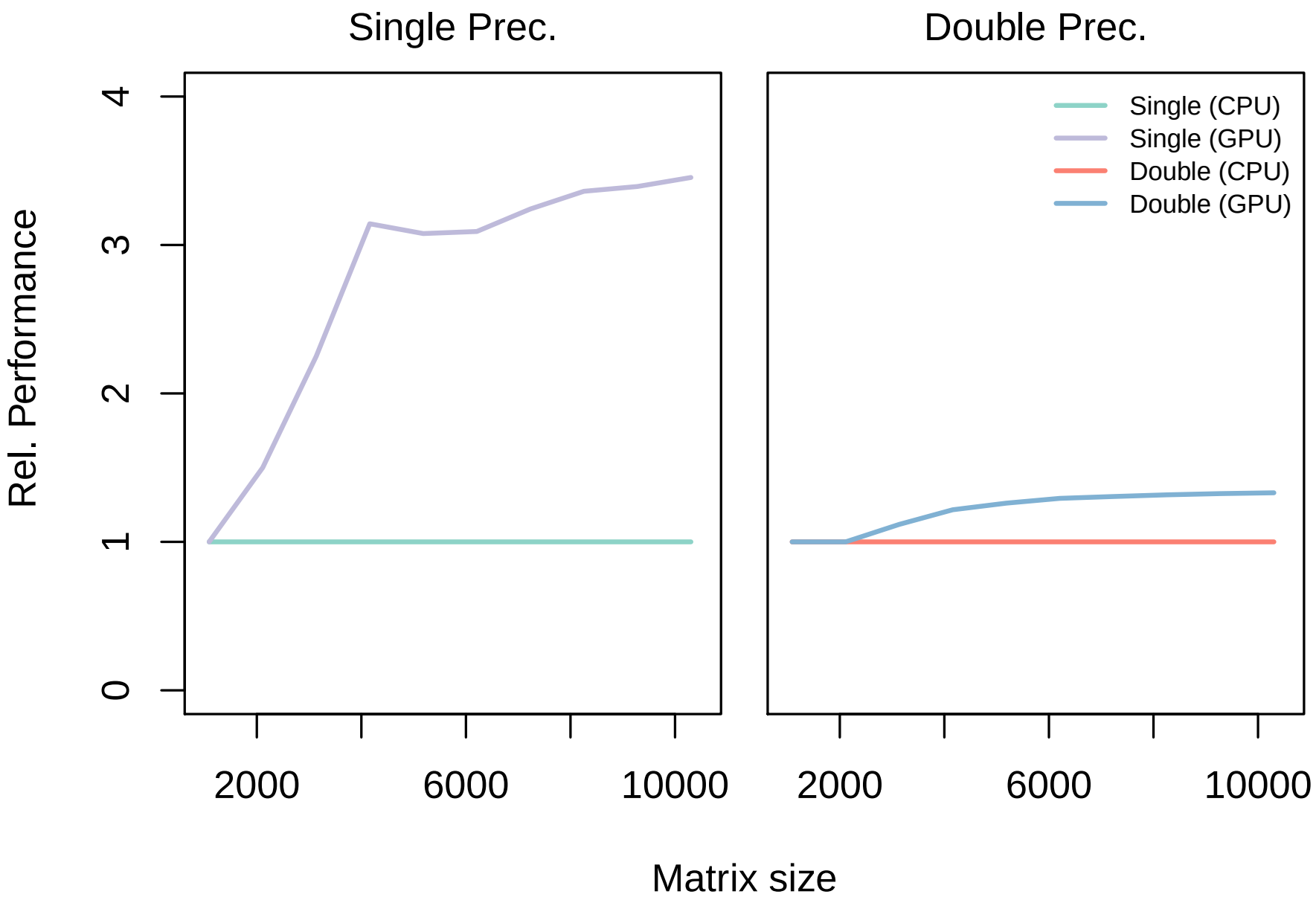
Software Specs:

- Ubuntu 13.10
- OpenBlas 0.2.8
- CUDA 6.0RC
- Magma 1.4.1

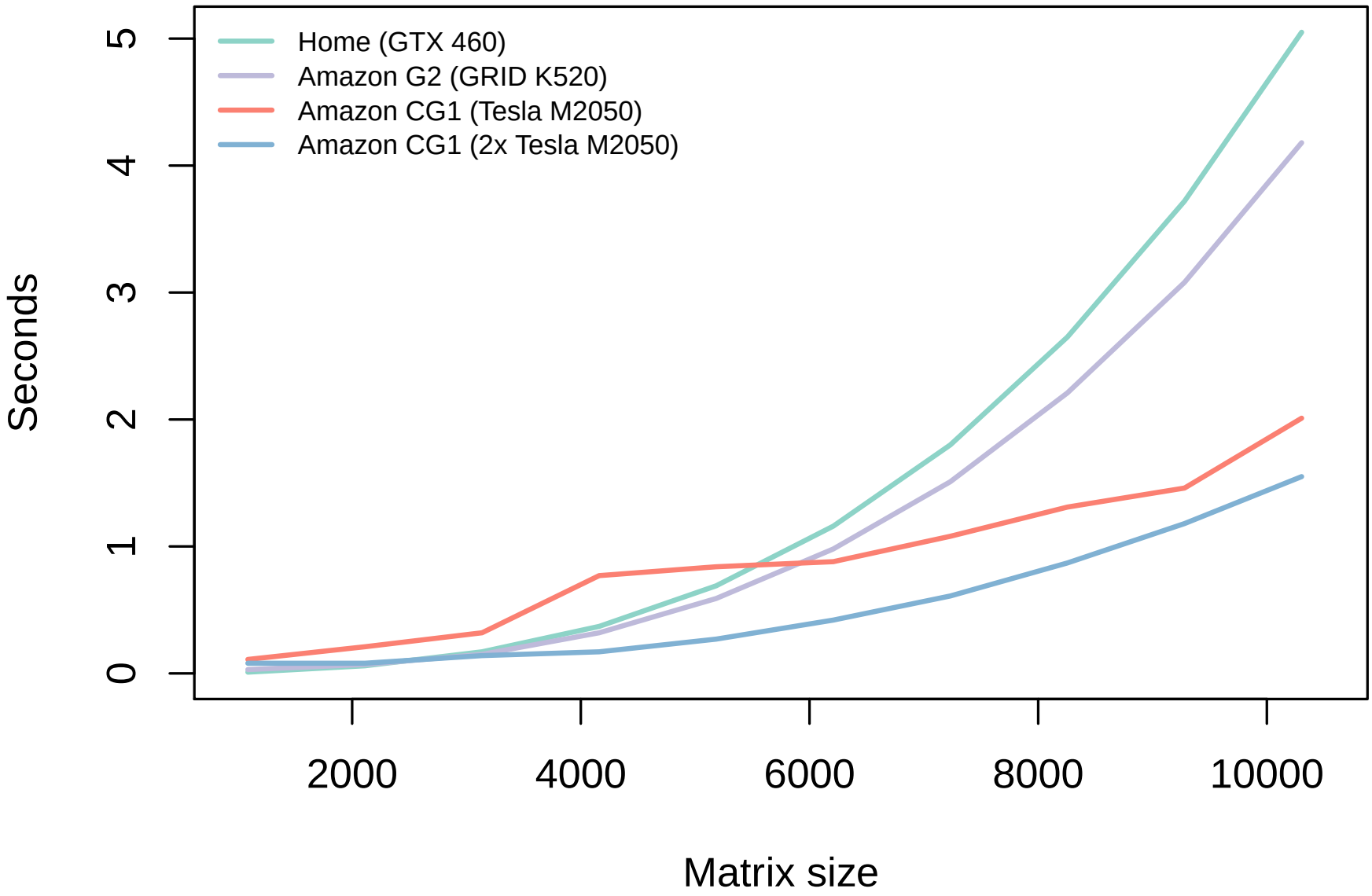
# Double vs single float performance



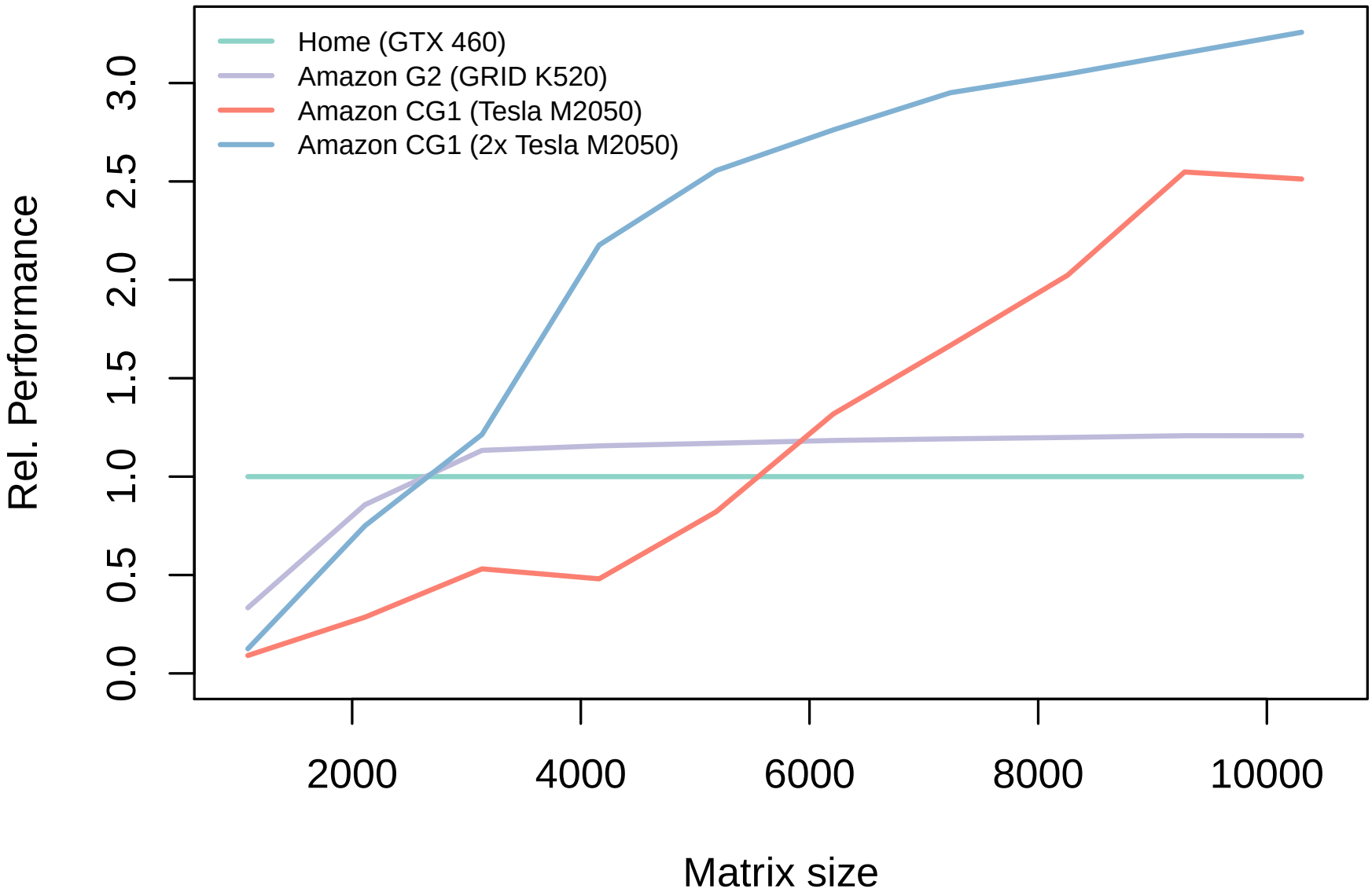
# Double vs single float performance



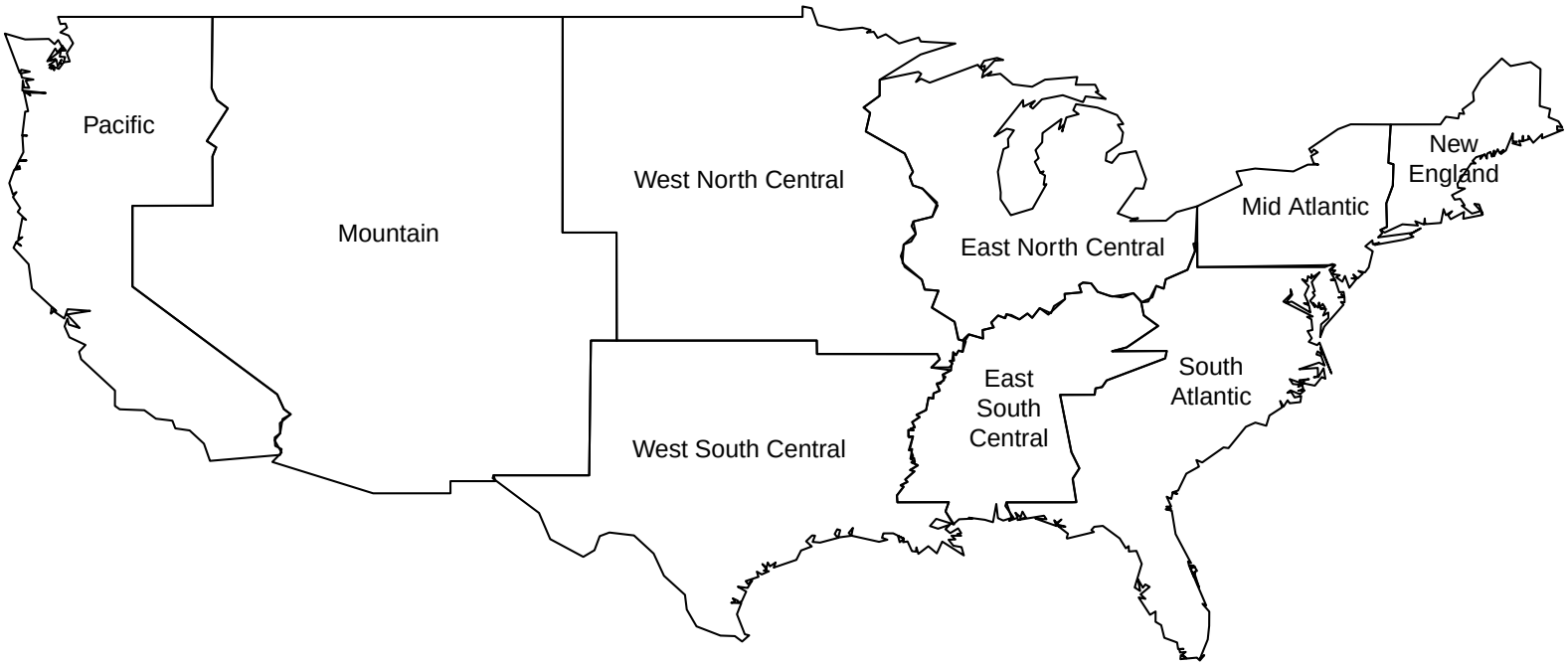
# Amazon EC2 Instances



# Amazon EC2 Instances



# Regions




Back