Computational advances in large n & large p" sparse Bayesian regression for binary & survival outcomes

Aki Nishimura[1]

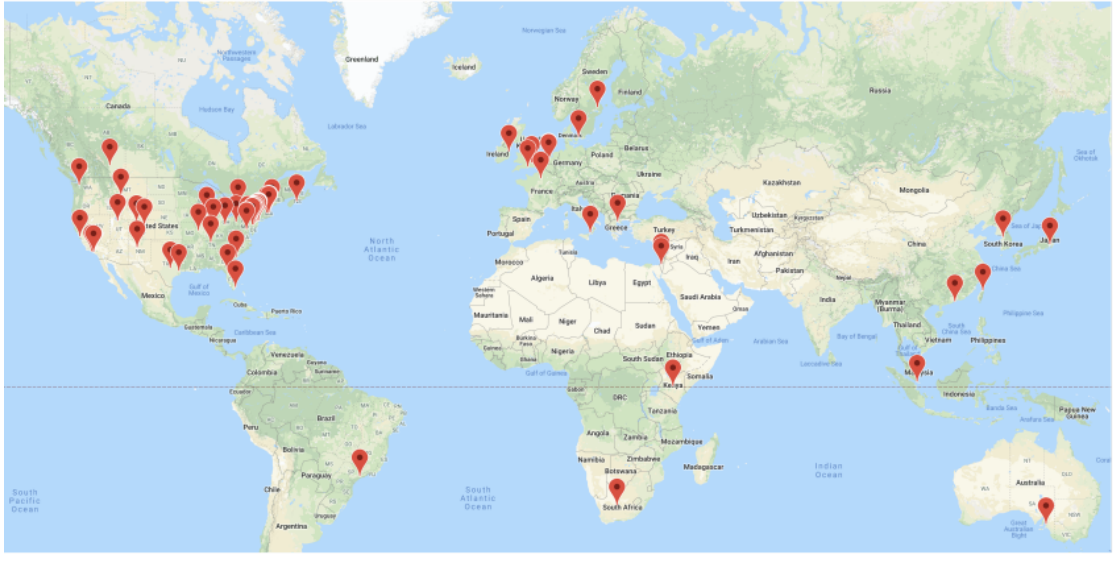
joint work with Marc Suchard

Department of Biomathematics, University of California - Los Angeles [1]

November 27, 2018

Observational Health Data Sciences and Informatics

OHDSI
OBSERVATIONAL HEALTH DATA SCIENCES AND INFORMATICS



COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

UCLA

IQVIA™

janssen
PHARMACEUTICAL COMPANIES
OF Johnson&Johnson

- 84 organizations
- 64 databases
- 1.26 billion patient records in total
- Open source analytic suite: `github.com/ohdsi`
- More info at `ohdsi.org`

"Large n and large p" regression for observational data

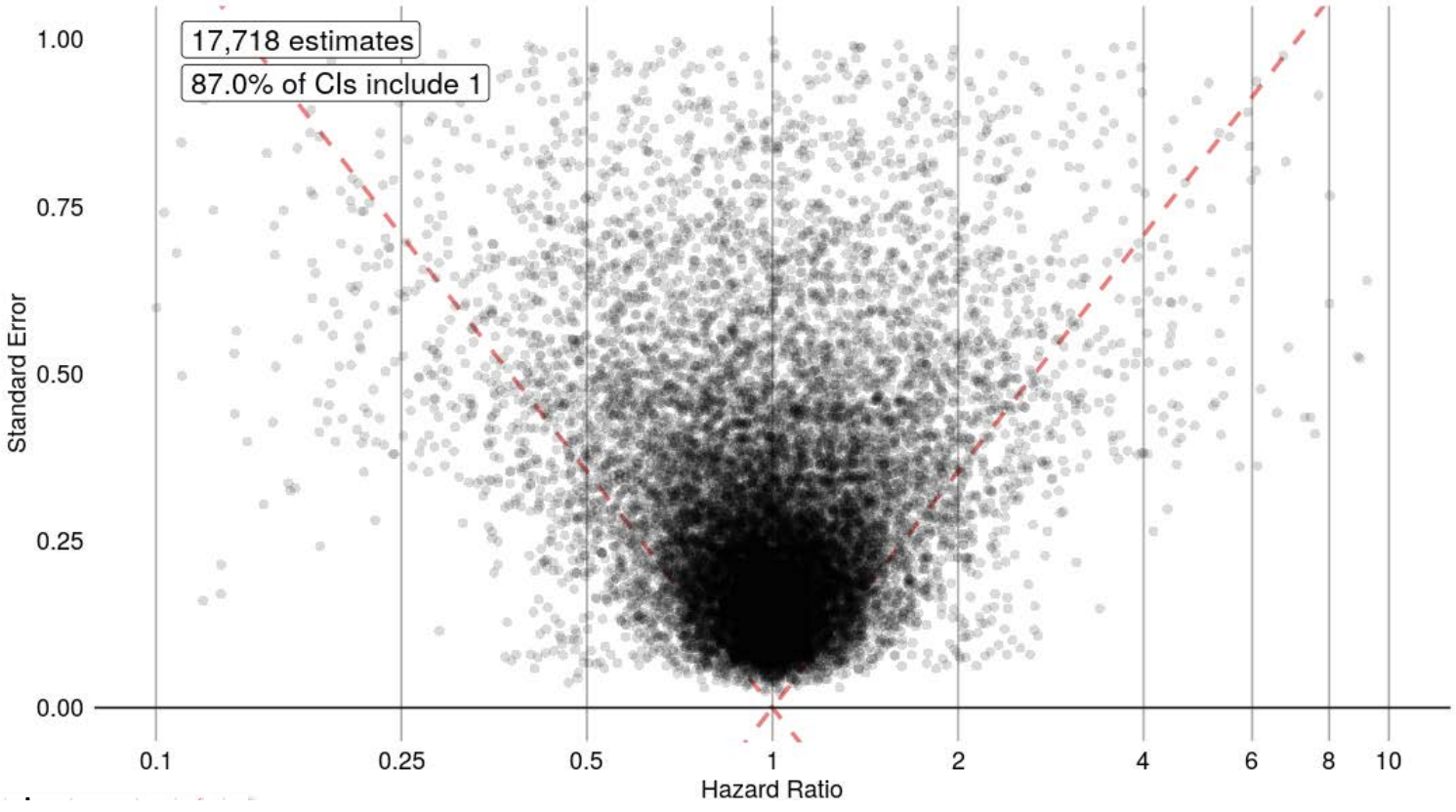
Example study based on observational data from healthcare databases:

- Goal — compare two different drugs (e.g. *dabigatran* and *warfarin*)
  - Efficacy of treatment? (e.g. prevention of blood clot formation)
  - Risk of serious side effects? (e.g. bleeding inside brain)?

"Large n and large p" regression for observational data

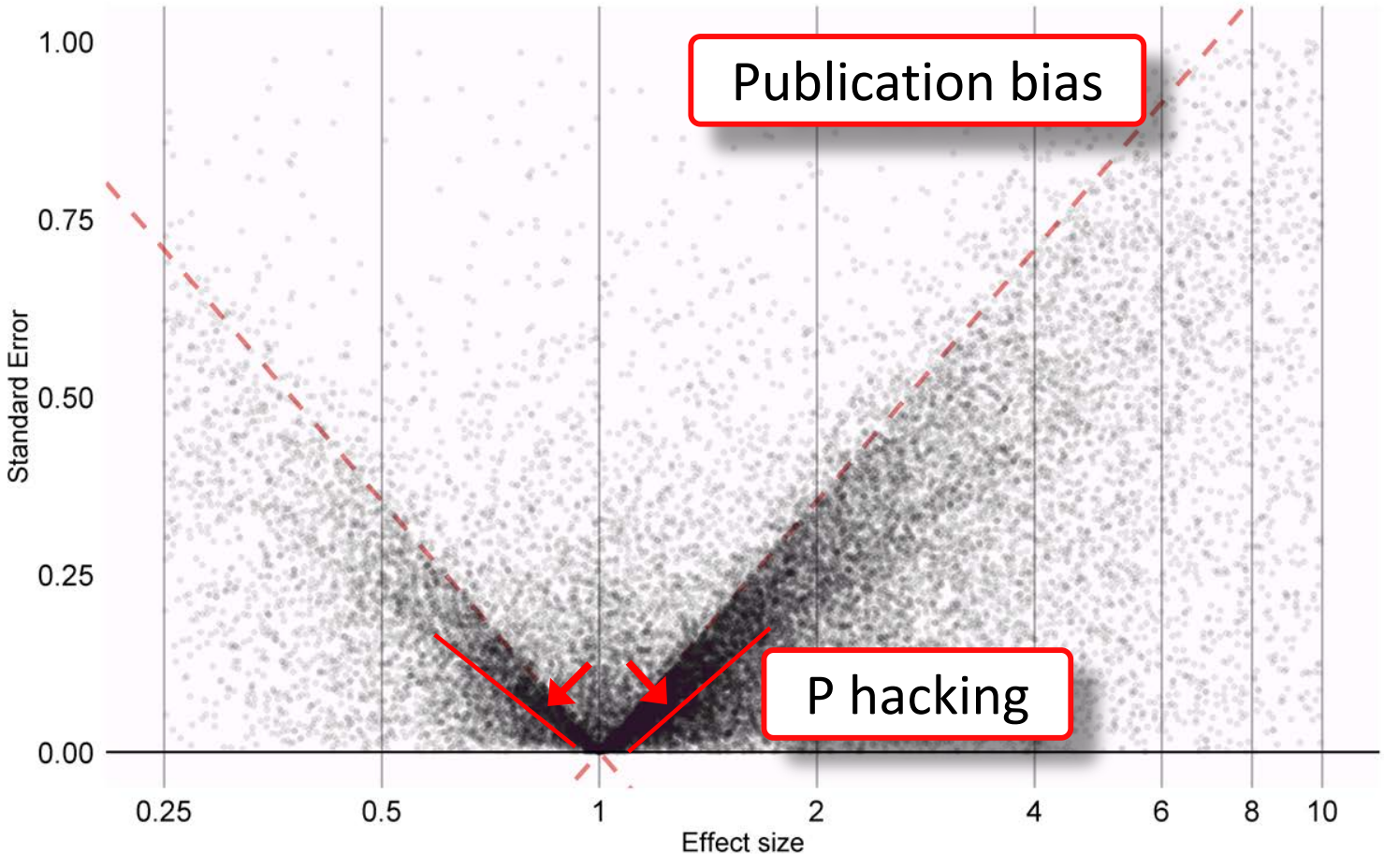
Example study based on observational data from healthcare databases:

- Goal — compare two different drugs (e.g. *dabigatran* and *warfarin*)
  - Efficacy of treatment? (e.g. prevention of blood clot formation)
  - Risk of serious side effects? (e.g. bleeding inside brain)?



"Large n and large p" regression for observational data

OHDSI's approach addresses "replication crisis" (we try, at least):



"Large n and large p" regression for observational data

- Scale of data:
  - Sample size: $n \approx 10^5 \sim 10^6$ (e.g. $n = 72{,}489$)
  - Number of features: $p \approx 10^4 \sim 10^5$ (e.g. $p = 22{,}175$); pre-existing conditions, prior treatments / drugs taken, and etc.

"Large n and large p" regression for observational data

- Scale of data:
  - Sample size: $n \approx 10^5 \sim 10^6$ (e.g. $n = 72{,}489$)
  - Number of features: $p \approx 10^4 \sim 10^5$ (e.g. $p = 22{,}175$); pre-existing conditions, prior treatments / drugs taken, and etc.
- Data characteristics:
  - Design matrix $\boldsymbol{X}$ is sparse, a small fraction of non-zero entries.
  - "Positive" outcome $y_i = 1$ is rare when regressing on serious adverse events (e.g. 192 out of 72,489)

"Large n and large p" regression for observational data

- Two stage estimation of treatment (dabigatran) effect:
  - Propensity score estimation
    — regress treatment indicator on predictors
  - Doubly-robust treatment effect estimation
    — regress outcome on treatment indicator
    + propensity score strata indicators + predictors

Table of Contents

Why Bayes?

- calibrates hyper-parameters via their marginal likelihoods.
  - vs. resampling / cross-validation methods
    — suspicious quality for rare outcomes
    (e.g. 192 cases out of $n = 72{,}489$ & $p = 22{,}175$)

Why Bayes?

- calibrates hyper-parameters via their marginal likelihoods.
  - vs. resampling / cross-validation methods
    — suspicious quality for rare outcomes
    (e.g. 192 cases out of $n = 72{,}489$ & $p = 22{,}175$)
- tends to have better non-asymptotic properties
- provides model selection uncertainty
  - vs. post-selection inference
- can incorporate structures behind given data
  (e.g. hierarchical modeling across different hospitals)

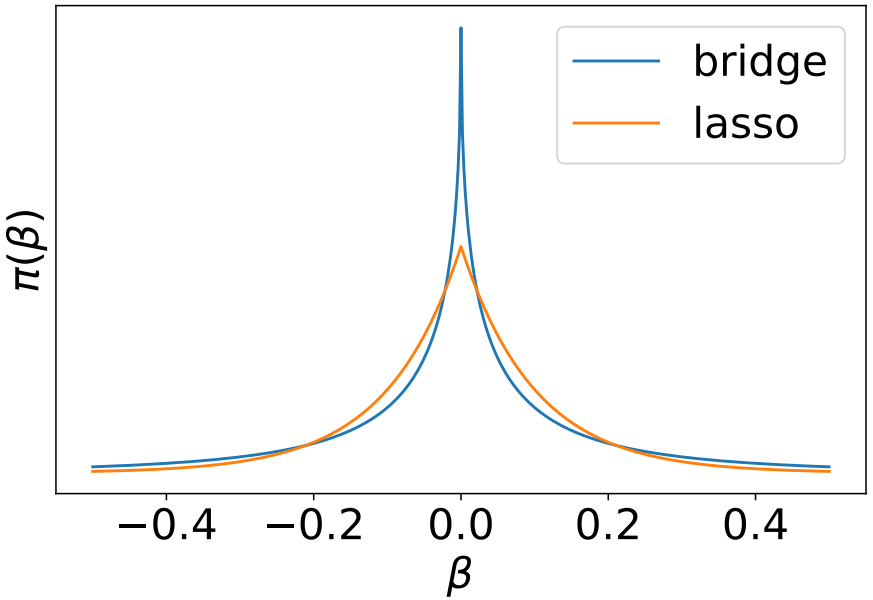
Table of Contents

Global-local shrinkage prior for sparse Bayesian regression

- Global-local shrinkage of regression coefficients $\boldsymbol{\beta}$:
  - Prior of the form $\beta_j \,|\, \tau, \lambda_j \sim \mathcal{N}\left(0, \tau^2 \lambda_j^2\right)$.
  - τ and λ_j are called *global* and *local shrinkage parameters*.
  - $\widehat{\tau} \ll 1$, but can have $\widehat{\tau}\widehat{\lambda}_j = O(1)$ for a small number of j's by virtue of heavy tailed priors on λ_j

Global-local shrinkage prior for sparse Bayesian regression

- Global-local shrinkage of regression coefficients $\boldsymbol{\beta}$:
  - Prior of the form $\beta_j \,|\, \tau, \lambda_j \sim \mathcal{N}\left(0, \tau^2\lambda_j^2\right)$.
  - τ and λ_j are called *global* and *local shrinkage parameters*.
  - $\widehat{\tau} \ll 1$, but can have $\widehat{\tau}\widehat{\lambda}_j = O(1)$ for a small number of j's by virtue of heavy tailed priors on λ_j
- Example — Bayesian bridge (Polson et al., 2014):



$\lambda_j \sim$ alpha-stable distribution with index of stability $\alpha/2$

$$\Updownarrow$$

$$\pi(\beta_j \,|\, \tau) \propto \tau^{-1} \exp\big(-|\beta_j/\tau|^{\alpha}\big)$$

Gibbs sampler for sparse Bayesian logistic regression

- Computation for Bayesian logistic regression is commonly based on the Polya-Gamma data augmentation scheme of Polson et al. (2013).
- Through an auxiliary parameter $\boldsymbol{\omega}$, the conditional likelihood of a binary outcome $\boldsymbol{y}$ becomes

$$y_i' \,|\, \boldsymbol{X}, \boldsymbol{\beta}, \boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{x}_i^\intercal \boldsymbol{\beta}, \omega_i^{-1}) \quad \text{for} \quad y_i' := \omega_i^{-1}\,(y_i - 1/2) \tag{1}$$

Gibbs sampler for sparse Bayesian logistic regression

- Computation for Bayesian logistic regression is commonly based on the Polya-Gamma data augmentation scheme of Polson et al. (2013).
- Through an auxiliary parameter $\boldsymbol{\omega}$, the conditional likelihood of a binary outcome $\boldsymbol{y}$ becomes

$$y_i' \,|\, \boldsymbol{X}, \boldsymbol{\beta}, \boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{x}_i^\intercal \boldsymbol{\beta}, \omega_i^{-1}) \quad \text{for} \quad y_i' := \omega_i^{-1}\,(y_i - 1/2) \tag{1}$$

- Correspondingly, the full conditional distribution of $\boldsymbol{\beta}$ is given by

$$\begin{aligned} \boldsymbol{\beta} \,|\, \boldsymbol{\omega}, \boldsymbol{\lambda}, \tau, \boldsymbol{y}, \boldsymbol{X} &\sim \mathcal{N}(\boldsymbol{\Phi}^{-1} \boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{y}', \boldsymbol{\Phi}^{-1}) \\ &\text{for} \quad \boldsymbol{\Phi} = \boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{X} + \tau^{-2} \boldsymbol{\Lambda}^{-2} \end{aligned} \tag{2}$$

Gibbs sampler for sparse Bayesian logistic regression

- Computation for Bayesian logistic regression is commonly based on the Polya-Gamma data augmentation scheme of Polson et al. (2013).
- Through an auxiliary parameter $\boldsymbol{\omega}$, the conditional likelihood of a binary outcome $\boldsymbol{y}$ becomes

$$y_i' \,|\, \boldsymbol{X}, \boldsymbol{\beta}, \boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}, \omega_i^{-1}) \quad \text{for} \quad y_i' := \omega_i^{-1}\,(y_i - 1/2) \tag{1}$$

- Correspondingly, the full conditional distribution of $\boldsymbol{\beta}$ is given by

$$\begin{aligned} \boldsymbol{\beta} \,|\, \boldsymbol{\omega}, \boldsymbol{\lambda}, \tau, \boldsymbol{y}, \boldsymbol{X} &\sim \mathcal{N}(\boldsymbol{\Phi}^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega}\boldsymbol{y}', \boldsymbol{\Phi}^{-1}) \\ &\text{for} \quad \boldsymbol{\Phi} = \boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega}\boldsymbol{X} + \tau^{-2}\boldsymbol{\Lambda}^{-2} \end{aligned} \tag{2}$$

- Generating samples from (2) is the main computational bottleneck:
  - $O(np^2)$ operations for computing $\boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega}\boldsymbol{X}$
  - $O(p^3)$ operations for the Cholesky factorization $\boldsymbol{\Phi} = \boldsymbol{L}\boldsymbol{L}^\mathsf{T}$
  - $\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{\delta}$ with $\boldsymbol{\delta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p) \Rightarrow \mathrm{Var}(\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{\delta}) = \boldsymbol{L}^{-\mathsf{T}}\boldsymbol{L}^{-1} = \boldsymbol{\Phi}^{-1}$

Current states of sparse Bayesian regression computations

- Significant progress in $n \ll p$ case:
  - Samples from $\boldsymbol{\beta} \,|\, \boldsymbol{\omega}, \boldsymbol{\lambda}, \tau, \boldsymbol{y}, \boldsymbol{X}$ can be generated with only $O(n^2 p)$ operations (Bhattacharya et al., 2016).
  - Further speed-up via approximation + improved mixing of the global shrinkage parameter through partial marginalization (Johndrow et al., 2018).

How bad is the standard computational approach?

Experimental set-up:

- Data: $\boldsymbol{X}$ is of size $72{,}489 \times 22{,}175$ and is 95% sparse
  $\boldsymbol{y}$ indicates the treatment by dabigatran over warfarin
  $y_i = 1$ account for 27.3% of the $72{,}489$ cases
- Computing environment: iMac 2015 with Intel Core i7 CPUs
- Implementation: reasonably optimized Python code
  (vectorized & optimal matrix formats used)

How bad is the standard computational approach?

Experimental set-up:

- Data: $\boldsymbol{X}$ is of size $72{,}489 \times 22{,}175$ and is 95% sparse
  $\boldsymbol{y}$ indicates the treatment by dabigatran over warfarin
  $y_i = 1$ account for 27.3% of the $72{,}489$ cases
- Computing environment: iMac 2015 with Intel Core i7 CPUs
- Implementation: reasonably optimized Python code
  (vectorized & optimal matrix formats used)

Computing time:

- 2 min 30 sec for the matrix-matrix multiplication $\boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{X}$
- 30 sec for the Cholesky factorization of $\boldsymbol{\Phi} = \boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{X} + \tau^{-2} \boldsymbol{\Lambda}^{-2}$

How bad is the standard computational approach?

Experimental set-up:

- Data: $\boldsymbol{X}$ is of size $72{,}489 \times 22{,}175$ and is 95% sparse
  $\boldsymbol{y}$ indicates the treatment by dabigatran over warfarin
  $y_i = 1$ account for 27.3% of the $72{,}489$ cases
- Computing environment: iMac 2015 with Intel Core i7 CPUs
- Implementation: reasonably optimized Python code
  (vectorized & optimal matrix formats used)

Computing time:

- 2 min 30 sec for the matrix-matrix multiplication $\boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{X}$
- 30 sec for the Cholesky factorization of $\boldsymbol{\Phi} = \boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{X} + \tau^{-2} \boldsymbol{\Lambda}^{-2}$
- In total, **15 hours for 300 iterations** of the Gibbs sampler.

How bad is the standard computational approach?

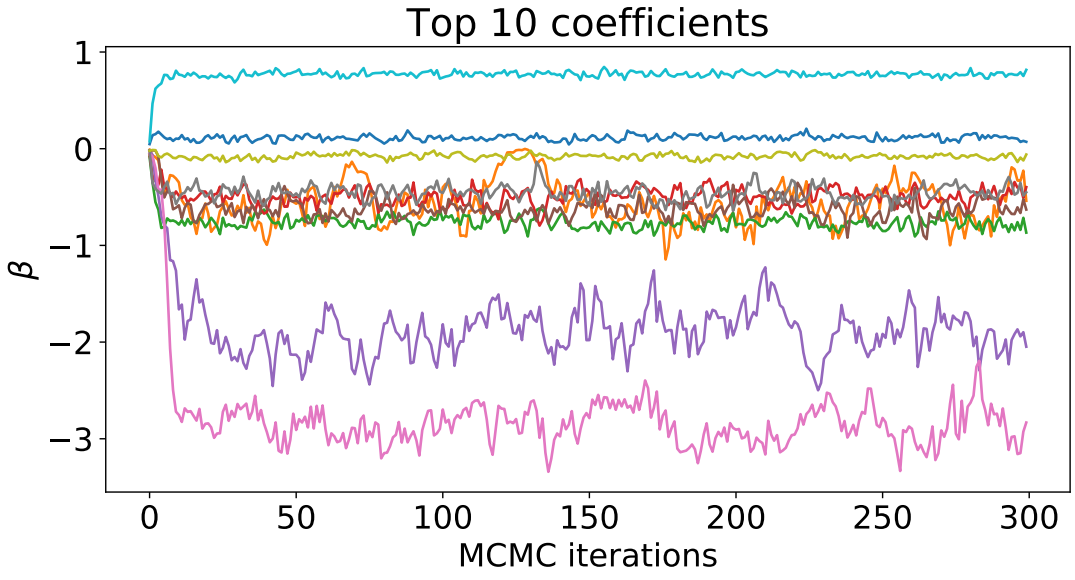
Compare with arguably the most popular sparse regression tool: *glmnet*

```
Loaded glmnet 2.0-13

Warning message:
package 'glmnet' was built under R version 3.4.2
> runtime <- system.time(
+   propensity_score_fit <- cv.glmnet(X, factor(y), family='binomial', standardize=FALSE)
+ )
> print(runtime)
    user   system  elapsed
4938.046   22.916 4947.262
```

How bad is the standard computational approach?

With the new algorithm, Bayesian method is competitive with glmnet.




```
In [2]: n_burnin = 0
   ...: n_post_burnin = 300
   ...: thin = 1
   ...: reg_exponent = .5
   ...: init = {
   ...:     'intercept': log_odds(np.mean(y))
   ...:     'tau': .01,
   ...:     'lambda': np.ones(X.shape[1]),
   ...: }
   ...: bridge = BayesBridge(y, X, link='logit')
   ...: start_time = time.time()
   ...: mcmc_output = bridge.gibbs(
   ...:     n_burnin, n_post_burnin, thin, reg_exponent, init,
   ...:     mvnorm_method='pcg', seed=0
   ...: )
   ...: print(time.time() - start_time)
   ...:
/Users/aki-nishimura/Dropbox/Documents_Academic/Scalable Bayes variable s
bbs/bridge.py:121: UserWarning: The numbers of trials were not specified.
4895.219202041626
```

Table of Contents

Conjugate gradient sampler for multivariate Gaussians

The problem of generating a sample from (3) can be recast as that of solving a linear system via the algorithm below:

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\Phi}^{-1}\boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{y}, \boldsymbol{\Phi}^{-1}) \quad \text{for} \quad \boldsymbol{\Phi} = \boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{X} + \tau^{-2}\boldsymbol{\Lambda}^{-2} \tag{3}$$

Algorithm (Nishimura & Suchard, 2018)

The following procedure generates a sample $\boldsymbol{\beta}$ from the distribution (3):

Conjugate gradient sampler for multivariate Gaussians

The problem of generating a sample from (3) can be recast as that of solving a linear system via the algorithm below:

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\Phi}^{-1}\boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{y}, \boldsymbol{\Phi}^{-1}) \quad \text{for} \quad \boldsymbol{\Phi} = \boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{X} + \tau^{-2}\boldsymbol{\Lambda}^{-2} \tag{3}$$

Algorithm (Nishimura & Suchard, 2018)

The following procedure generates a sample $\boldsymbol{\beta}$ from the distribution (3):

1. Generate $\boldsymbol{b} \sim \mathcal{N}\left(\boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{y}, \boldsymbol{\Phi}\right)$ by sampling independent Gaussian vectors $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ and $\boldsymbol{\delta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$ and then setting

$$\boldsymbol{b} = \boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{y} + \boldsymbol{X}^{\intercal}\boldsymbol{\Omega}^{1/2}\boldsymbol{\eta} + \tau^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{\delta} \tag{4}$$

Conjugate gradient sampler for multivariate Gaussians

The problem of generating a sample from (3) can be recast as that of solving a linear system via the algorithm below:

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\Phi}^{-1}\boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{y}, \boldsymbol{\Phi}^{-1}) \quad \text{for} \quad \boldsymbol{\Phi} = \boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{X} + \tau^{-2}\boldsymbol{\Lambda}^{-2} \tag{3}$$

Algorithm (Nishimura & Suchard, 2018)

The following procedure generates a sample $\boldsymbol{\beta}$ from the distribution (3):

1. Generate $\boldsymbol{b} \sim \mathcal{N}\left(\boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{y}, \boldsymbol{\Phi}\right)$ by sampling independent Gaussian vectors $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$ and $\boldsymbol{\delta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$ and then setting

$$\boldsymbol{b} = \boldsymbol{X}^{\intercal}\boldsymbol{\Omega}\boldsymbol{y} + \boldsymbol{X}^{\intercal}\boldsymbol{\Omega}^{1/2}\boldsymbol{\eta} + \tau^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{\delta} \tag{4}$$

2. Solve the following linear system for $\boldsymbol{\beta}$

$$\boldsymbol{\Phi}\boldsymbol{\beta} = \boldsymbol{b} \tag{5}$$

so that $\mathrm{Var}(\boldsymbol{\beta}) = \mathrm{Var}(\boldsymbol{\Phi}^{-1}\boldsymbol{b}) = \boldsymbol{\Phi}^{-1}\mathrm{Var}(\boldsymbol{b})\boldsymbol{\Phi}^{-1}$.

Conjugate gradient (CG) for solving linear system $\Phi\boldsymbol{\beta} = \boldsymbol{b}$

- CG is an *iterative method* for solving a positive definite system.
- Given an initial guess $\boldsymbol{\beta}_0$, CG generates a sequence $\{\boldsymbol{\beta}_k\}_{k=1,2,\ldots}$ of increasingly accurate approximations to the solution.

Conjugate gradient (CG) for solving linear system $\Phi\beta = b$

- CG is an *iterative method* for solving a positive definite system.
- Given an initial guess β_0, CG generates a sequence $\{\beta_k\}_{k=1,2,\ldots}$ of increasingly accurate approximations to the solution.
- Cost of update $\beta_k \to \beta_{k+1}$ is dominated by the operation $v \to \Phi v$.

Conjugate gradient (CG) for solving linear system $\boldsymbol{\Phi\beta} = \boldsymbol{b}$

- CG is an *iterative method* for solving a positive definite system.
- Given an initial guess $\boldsymbol{\beta}_0$, CG generates a sequence $\{\boldsymbol{\beta}_k\}_{k=1,2,\ldots}$ of increasingly accurate approximations to the solution.
- Cost of update $\boldsymbol{\beta}_k \to \boldsymbol{\beta}_{k+1}$ is dominated by the operation $\boldsymbol{v} \to \boldsymbol{\Phi v}$.
- **Note:** the operation $\boldsymbol{v} \to \boldsymbol{\Phi v}$ requires no explicit formation of $\boldsymbol{\Phi}$
  - Multiplication by $\boldsymbol{\Phi} = \boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega X} + \tau^{-2}\boldsymbol{\Lambda}^{-2}$ can be carried out via $\boldsymbol{v} \to \boldsymbol{Xv}$, $\boldsymbol{w} \to \boldsymbol{X}^\mathsf{T}\boldsymbol{w}$, and some element-wise multiplications.
  - Memory advantage: $\boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega X}$ are often much denser than $\boldsymbol{X}$. (e.g. 74.5 GB of memory to allocate $p \times p$ matrix when $p = 10^5$)

Conjugate gradient (CG) for solving linear system $\boldsymbol{\Phi\beta} = \boldsymbol{b}$

- CG is an *iterative method* for solving a positive definite system.
- Given an initial guess $\boldsymbol{\beta}_0$, CG generates a sequence $\{\boldsymbol{\beta}_k\}_{k=1,2,\ldots}$ of increasingly accurate approximations to the solution.
- Cost of update $\boldsymbol{\beta}_k \to \boldsymbol{\beta}_{k+1}$ is dominated by the operation $\boldsymbol{v} \to \boldsymbol{\Phi v}$.
- **Note:** the operation $\boldsymbol{v} \to \boldsymbol{\Phi v}$ requires no explicit formation of $\boldsymbol{\Phi}$
  - Multiplication by $\boldsymbol{\Phi} = \boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega X} + \tau^{-2}\boldsymbol{\Lambda}^{-2}$ can be carried out via $\boldsymbol{v} \to \boldsymbol{Xv}$, $\boldsymbol{w} \to \boldsymbol{X}^\mathsf{T}\boldsymbol{w}$, and some element-wise multiplications.
  - Memory advantage: $\boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega X}$ are often much denser than $\boldsymbol{X}$. (e.g. 74.5 GB of memory to allocate $p \times p$ matrix when $p = 10^5$)
- CG yields an exact solution in p iterations
  - In the worst case, the number of required arithmetic operations is comparable to that a direct method.
  - But it is possible to achieve $\boldsymbol{\beta}_k \approx \boldsymbol{\beta} = \boldsymbol{\Phi}^{-1}\boldsymbol{b}$ for $k \ll p$.

Illustration of the CG Gaussian sampler

- Dabigatran vs. warfarin comparison data ($n = 72{,}489,\ p = 22{,}175$)
- Compare CG iterates $\boldsymbol{\beta}_k$'s to $\boldsymbol{\beta} = \boldsymbol{\Phi}^{-1}\boldsymbol{b}$ for $\boldsymbol{b} \sim \mathcal{N}\left(\boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{y}, \boldsymbol{\Phi}\right)$.

Illustration of the CG Gaussian sampler

- Dabigatran vs. warfarin comparison data ($n = 72{,}489,\ p = 22{,}175$)
- Compare CG iterates $\boldsymbol{\beta}_k$'s to $\boldsymbol{\beta} = \boldsymbol{\Phi}^{-1}\boldsymbol{b}$ for $\boldsymbol{b} \sim \mathcal{N}\left(\boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{y}, \boldsymbol{\Phi}\right)$.

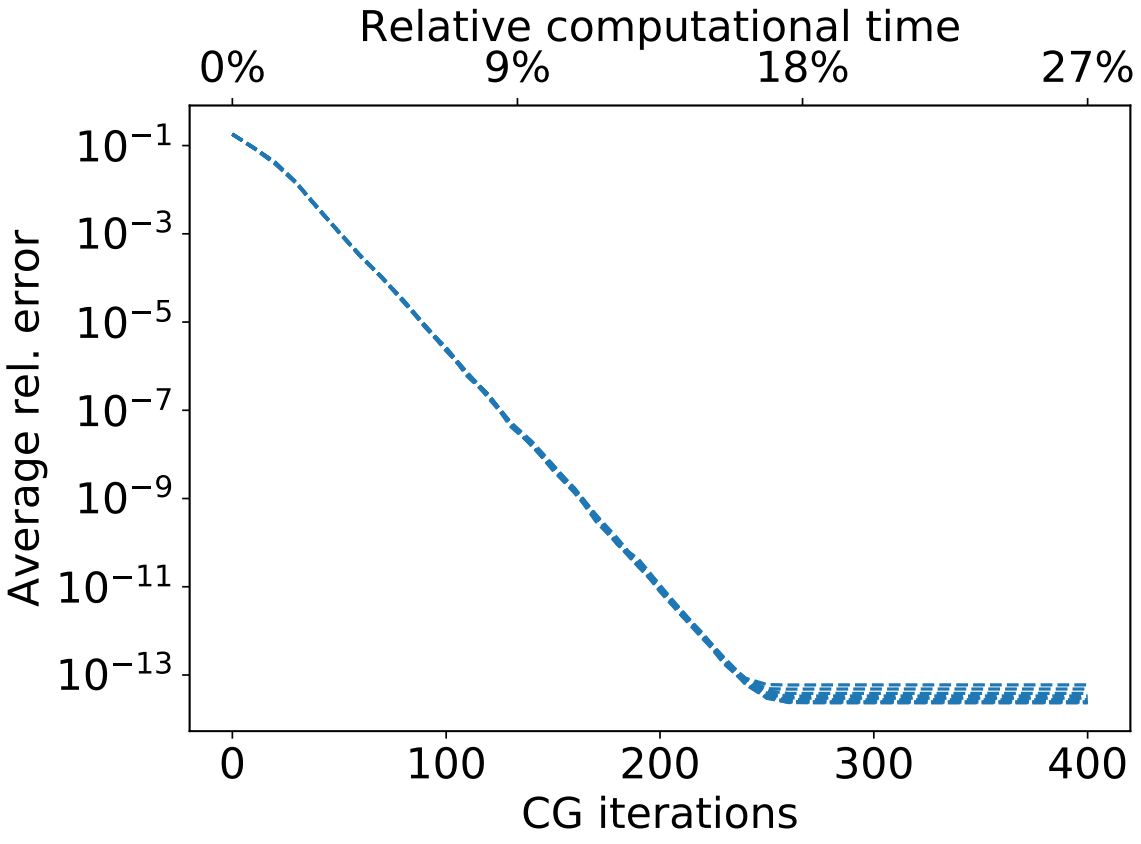


Figure: Error is quantified as the average of $|(\boldsymbol{\beta}_k)_j - \beta_j| / |\beta_j|$'s.

Table of Contents

Preconditioning CG to accelerate its convergence

- Direct application of CG rarely leads to a rapid convergence.

Preconditioning CG to accelerate its convergence

- Direct application of CG rarely leads to a rapid convergence.
- CG is typically applied to a preconditioned system

$$\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{\beta}} = \tilde{\boldsymbol{b}} \quad \text{for } \tilde{\boldsymbol{\Phi}} = \boldsymbol{M}^{-1/2}\boldsymbol{\Phi}\boldsymbol{M}^{-1/2} \text{ and } \tilde{\boldsymbol{b}} = \boldsymbol{M}^{-1/2}\boldsymbol{b} \tag{6}$$

  with a positive definite *preconditioner* matrix $\boldsymbol{M}$.

Preconditioning CG to accelerate its convergence

- Direct application of CG rarely leads to a rapid convergence.
- CG is typically applied to a preconditioned system

$$\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{\beta}} = \tilde{\boldsymbol{b}} \quad \text{for } \tilde{\boldsymbol{\Phi}} = \boldsymbol{M}^{-1/2}\boldsymbol{\Phi}\boldsymbol{M}^{-1/2} \text{ and } \tilde{\boldsymbol{b}} = \boldsymbol{M}^{-1/2}\boldsymbol{b} \tag{6}$$

  with a positive definite *preconditioner* matrix $\boldsymbol{M}$.
- Key considerations when designing a preconditioner:
  - Convergence rate of CG applied to the preconditioned system.
  - One-time cost of computing $\boldsymbol{M}$; better be less than inverting $\boldsymbol{\Phi}$.

Preconditioning CG to accelerate its convergence

- Direct application of CG rarely leads to a rapid convergence.
- CG is typically applied to a preconditioned system

$$\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{\beta}} = \tilde{\boldsymbol{b}} \quad \text{for } \tilde{\boldsymbol{\Phi}} = \boldsymbol{M}^{-1/2}\boldsymbol{\Phi}\boldsymbol{M}^{-1/2} \text{ and } \tilde{\boldsymbol{b}} = \boldsymbol{M}^{-1/2}\boldsymbol{b} \tag{6}$$

  with a positive definite *preconditioner* matrix $\boldsymbol{M}$.
- Key considerations when designing a preconditioner:
  - Convergence rate of CG applied to the preconditioned system.
  - One-time cost of computing $\boldsymbol{M}$; better be less than inverting $\boldsymbol{\Phi}$.

"Finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science."

Saad (2003)

Convergence theory of CG: rule of thumb

- Two most famous results on CG convergence rates — *not* the most useful ones (Nishimura & Suchard, 2018).
- Following rule of thumb is more useful:
  - CG converges quickly if the eigenvalues of Φ are "clustered."
  - if not too many, large eigenvalues cause little delay in convergence.
  - small eigenvalues tend to delay convergence longer.

Illustration of the CG convergence

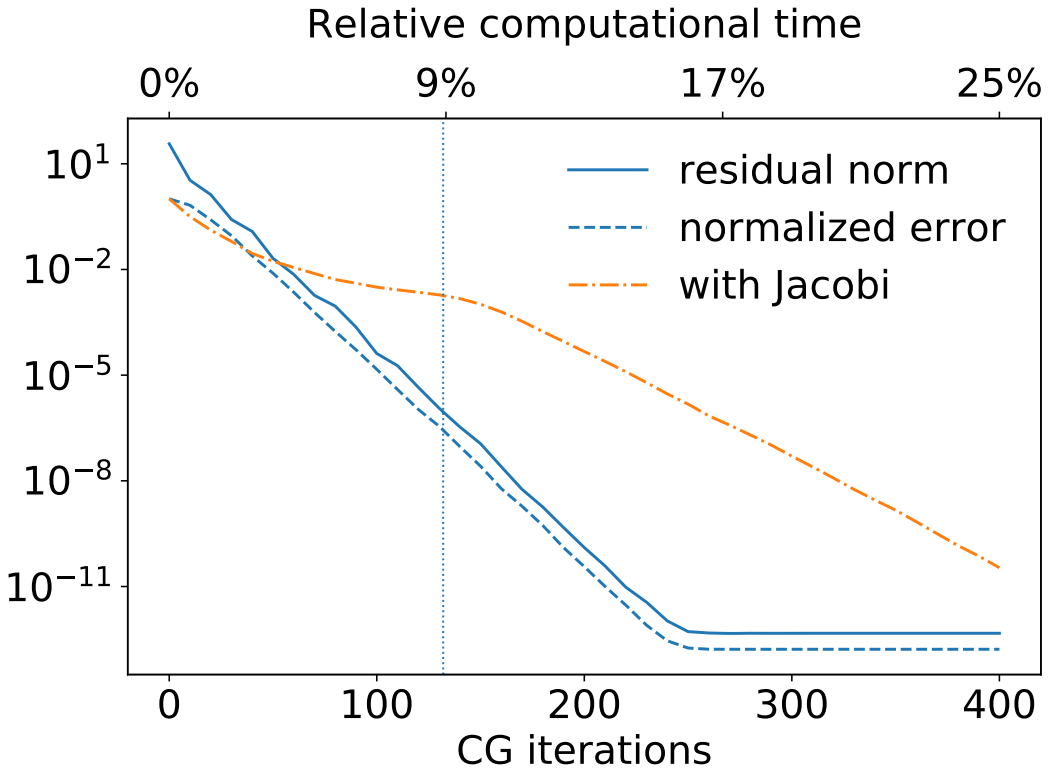
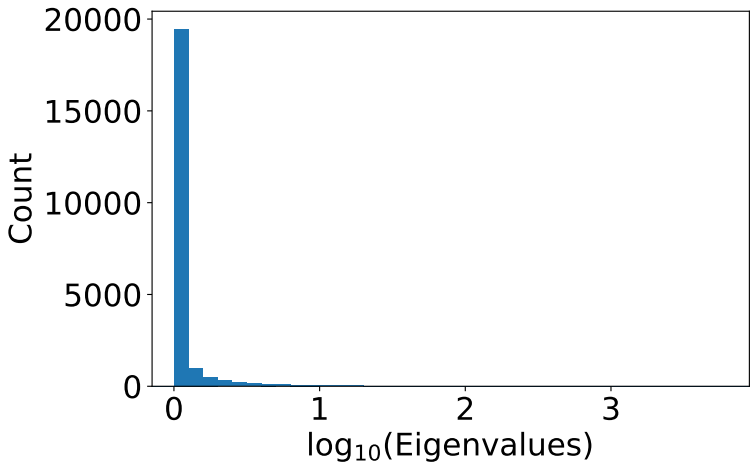


Figure: Comparison of two preconditioning strategies in sparse Bayesian logistic regression context ($n = 72{,}489$ & $p = 22{,}175$).

- Normalized error: $\left\{ \frac{1}{p} \sum_j \hat{\xi}_j^{-2} (\boldsymbol{\beta}_k - \boldsymbol{\beta})_j^2 \right\}^{1/2}$ with $\hat{\xi}_j^2 \approx \mathbb{E}[\beta_j^2 \,|\, \boldsymbol{y}, \boldsymbol{X}]$
- Norm of the preconditioned residual $p^{-1/2} \|\tilde{\boldsymbol{r}}_k\|_2$ as a stopping criteria

Illustration of the CG convergence



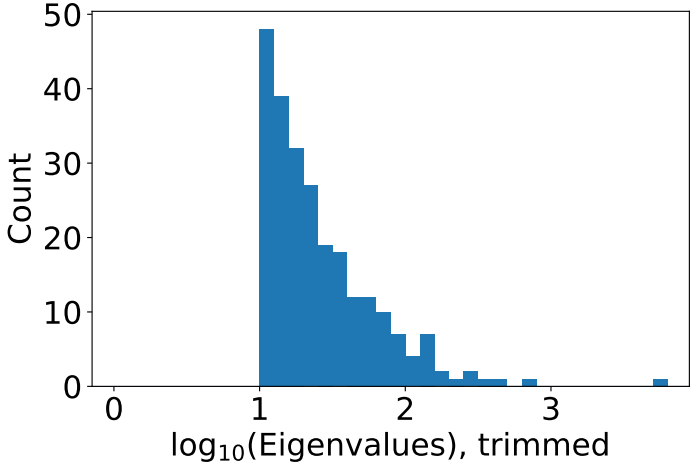
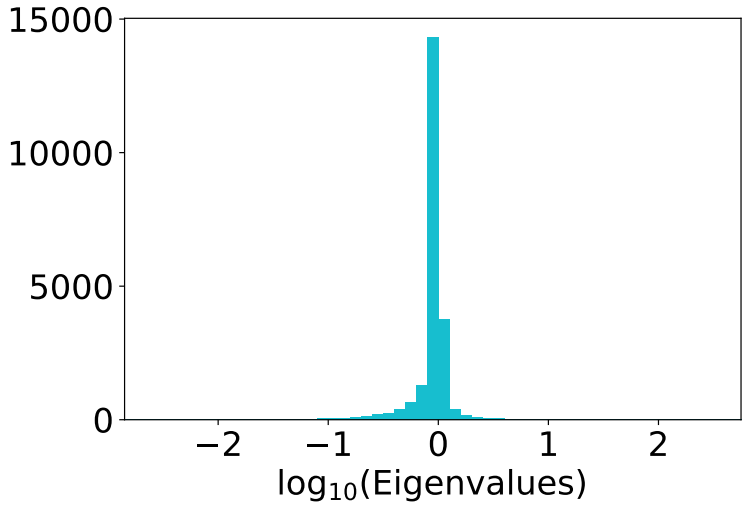


Figure: Eigenvalue distribution under the proposed preconditioner.



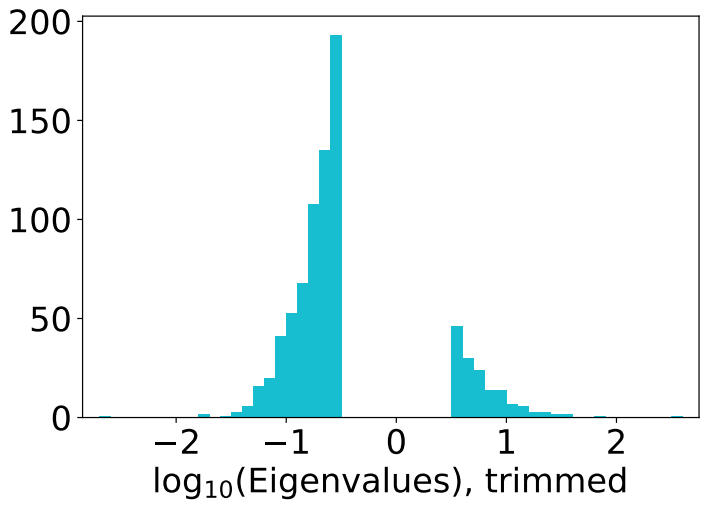


Figure: Eigenvalue distribution of the Jacobi preconditioned matrix.

Preconditioner for sparse Bayesian regression

The magic preconditioner for sparse Bayesian regression problem for $\boldsymbol{\Phi} = \boldsymbol{X}^\intercal \boldsymbol{\Omega} \boldsymbol{X} + \tau^{-2} \boldsymbol{\Lambda}^{-2}$ is . . . (drum roll) . . .

Preconditioner for sparse Bayesian regression

The magic preconditioner for sparse Bayesian regression problem for $\boldsymbol{\Phi} = \boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega}\boldsymbol{X} + \tau^{-2}\boldsymbol{\Lambda}^{-2}$ is . . . (drum roll) . . .

$$\boldsymbol{M} = \tau^{-2}\boldsymbol{\Lambda}^{-2} \tag{7}$$

Preconditioner for sparse Bayesian regression

The magic preconditioner for sparse Bayesian regression problem for $\boldsymbol{\Phi} = \boldsymbol{X}^\mathsf{T}\boldsymbol{\Omega}\boldsymbol{X} + \tau^{-2}\boldsymbol{\Lambda}^{-2}$ is . . . (drum roll) . . .

$$\boldsymbol{M} = \tau^{-2}\boldsymbol{\Lambda}^{-2} \tag{7}$$

. . . isn't it . . . too simple? (= boring = unpublishable?)

Prior preconditioner: made for sparse Bayesian regression

- Fancier (general-purpose) preconditioners exist: successive over-relaxation, incomplete Cholesky, sparse approximate inverse, etc.
- But they stand no chance against the proposed *prior preconditioner*

$$\boldsymbol{M} = \tau^{-2}\boldsymbol{\Lambda}^{-2} = \mathrm{Var}(\boldsymbol{\beta} \,|\, \tau, \boldsymbol{\lambda}, \boldsymbol{\omega})^{-1}$$

- Jacobi's $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{\Phi})$ is reasonable, but is significantly inferior.

Prior preconditioner: made for sparse Bayesian regression

- Fancier (general-purpose) preconditioners exist: successive over-relaxation, incomplete Cholesky, sparse approximate inverse, etc.
- But they stand no chance against the proposed *prior preconditioner*

$$\boldsymbol{M} = \tau^{-2}\boldsymbol{\Lambda}^{-2} = \mathrm{Var}(\boldsymbol{\beta} \,|\, \tau, \boldsymbol{\lambda}, \boldsymbol{\omega})^{-1}$$

- Jacobi's $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{\Phi})$ is reasonable, but is significantly inferior.
- Mathematically, prior-preconditioning is equivalent to sampling from

$$\tau^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{\beta} \,|\, \tau, \boldsymbol{\lambda}, \boldsymbol{\omega}, \boldsymbol{y}, \boldsymbol{X}$$

- Exploits the fundamental feature of sparse Bayesian regression:

  the prior dominates the likelihood — and hence the posterior looks like the prior — except for a small number of directions

Prior preconditioner: made for sparse Bayesian regression

- Fancier (general-purpose) preconditioners exist: successive over-relaxation, incomplete Cholesky, sparse approximate inverse, etc.
- But they stand no chance against the proposed *prior preconditioner*

$$\boldsymbol{M} = \tau^{-2}\boldsymbol{\Lambda}^{-2} = \mathrm{Var}(\boldsymbol{\beta} \,|\, \tau, \boldsymbol{\lambda}, \boldsymbol{\omega})^{-1}$$

- Jacobi's $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{\Phi})$ is reasonable, but is significantly inferior.
- Mathematically, prior-preconditioning is equivalent to sampling from

$$\tau^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{\beta} \,|\, \tau, \boldsymbol{\lambda}, \boldsymbol{\omega}, \boldsymbol{y}, \boldsymbol{X}$$

- Exploits the fundamental feature of sparse Bayesian regression:

  the prior dominates the likelihood — and hence the posterior looks like the prior — except for a small number of directions

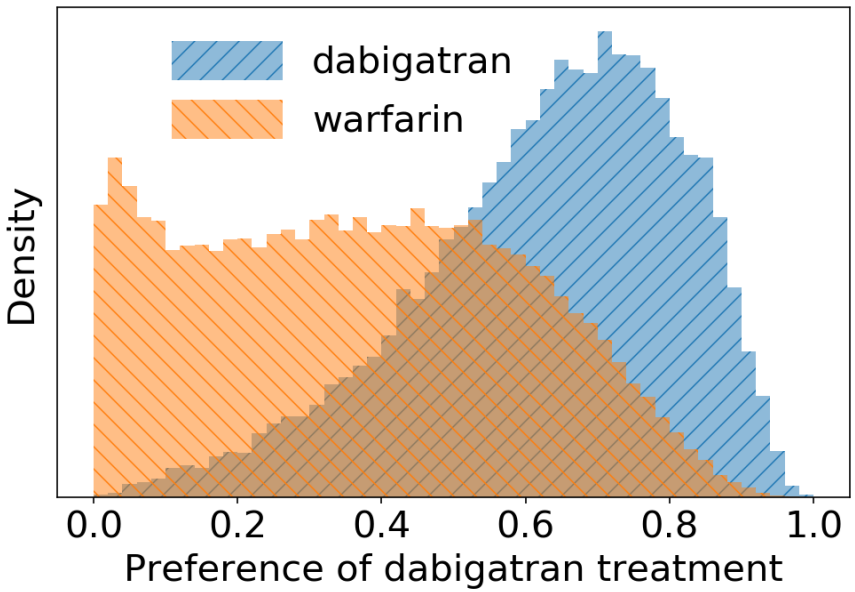
- See Nishimura & Suchard (2018) for mathematically precise theory.

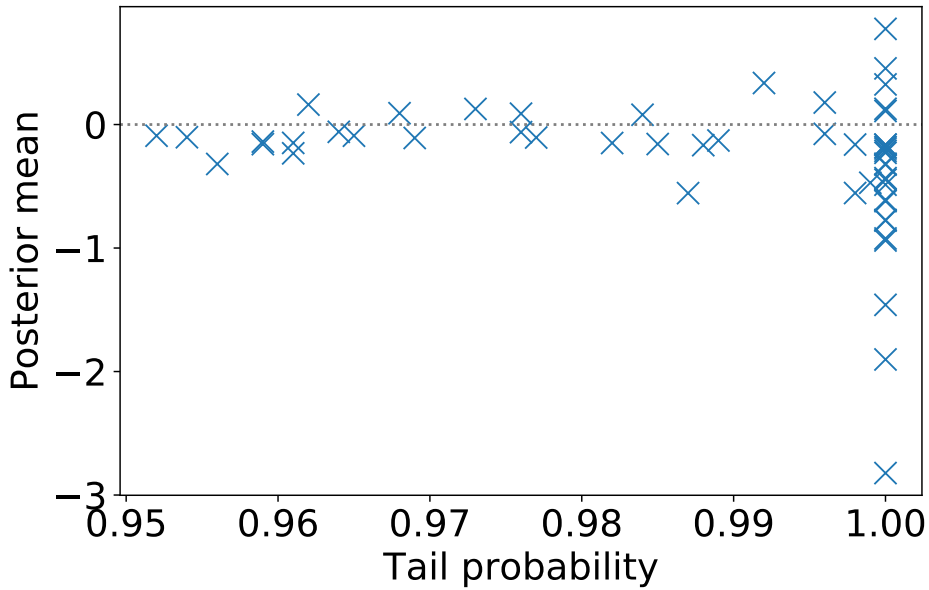
Table of Contents

Speed-up relative to the direct linear algebra Gibbs sampler

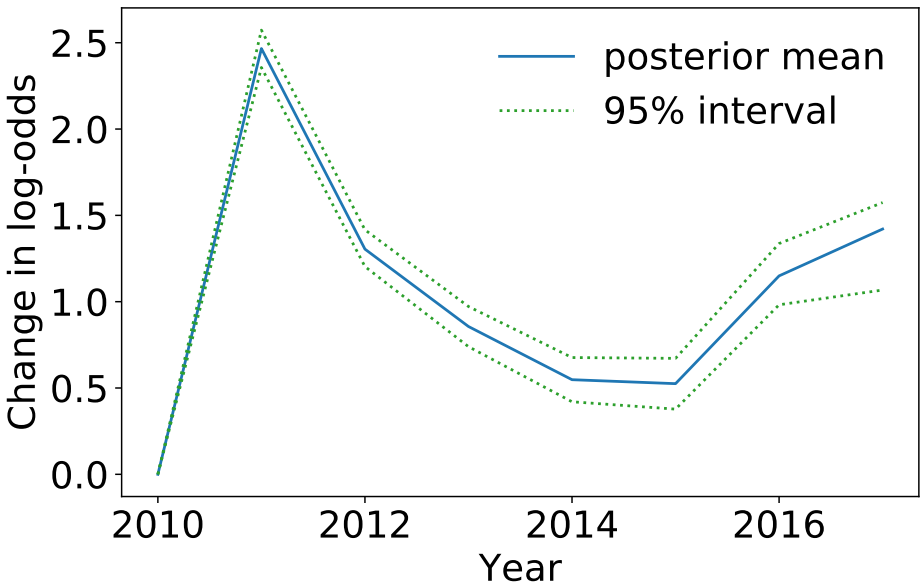
- Propensity score estimation
  - 1,500 iterations in 7.04 (vs 77.4) hours — **11** times speed-up
- Doubly robust treatment effect estimation
  - 2,000 iteration for 4.36 (vs 107) hour — **25** times speed-up

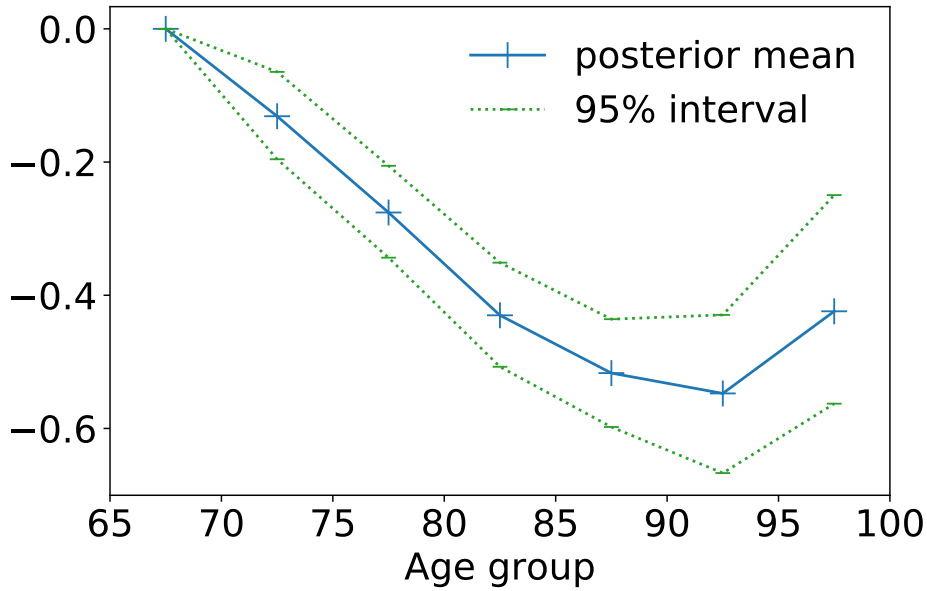
Propensity scores & difference in population



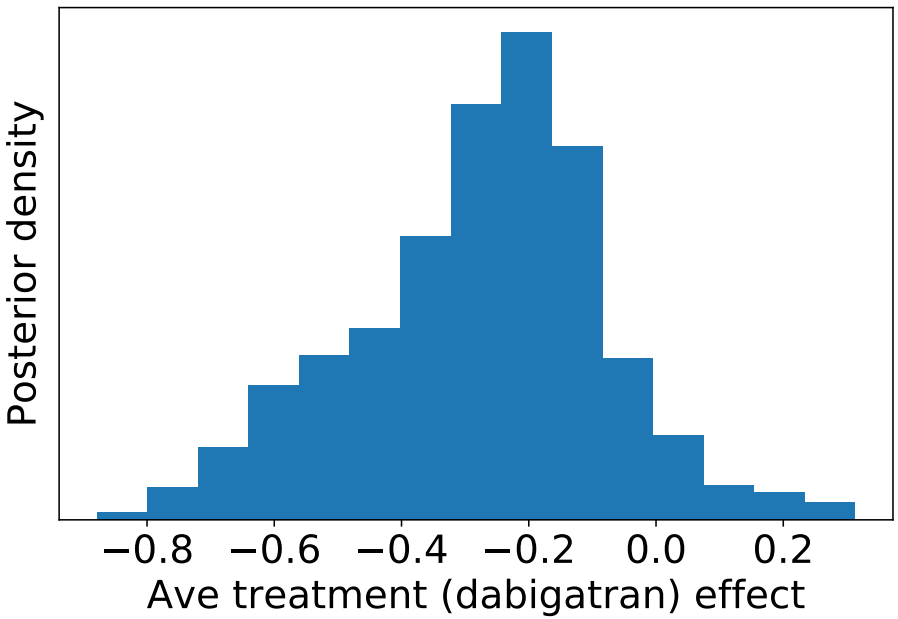


Propensity scores & difference in population





Effect of dabigatran over warfarin



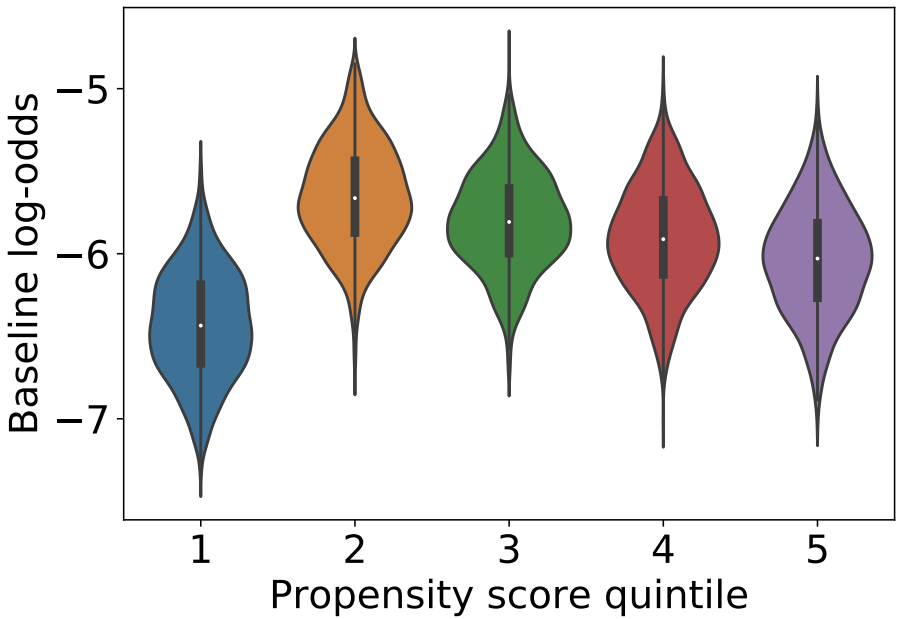


Table of Contents

Sparse Bayesian regression for log-concave likelihood

- Conditional distribution of $\boldsymbol{\beta} \mid \boldsymbol{\lambda}, \tau, \boldsymbol{y}, \boldsymbol{X}$ typically has no closed-form expression but is log-concave with predictable Hessian structures.

Sparse Bayesian regression for log-concave likelihood

- Conditional distribution of $\boldsymbol{\beta} \,|\, \boldsymbol{\lambda}, \tau, \boldsymbol{y}, \boldsymbol{X}$ typically has no closed-form expression but is log-concave with predictable Hessian structures.
- *Hamiltonian Monte Carlo* (HMC) is particularly well-suited to log-concave distributions:
  - computational cost is $O\left(p^{1/4}\left(\frac{M}{m}\right)^{1/2}\right)$ where M and m are the upper and lower bound on eigenvalues of the log-density Hessian (Mangoubi and Smith, 2017).

Sparse Bayesian regression for log-concave likelihood

- Conditional distribution of $\boldsymbol{\beta} \,|\, \boldsymbol{\lambda}, \tau, \boldsymbol{y}, \boldsymbol{X}$ typically has no closed-form expression but is log-concave with predictable Hessian structures.
- *Hamiltonian Monte Carlo* (HMC) is particularly well-suited to log-concave distributions:
  - computational cost is $O\left(p^{1/4}\left(\frac{M}{m}\right)^{1/2}\right)$ where M and m are the upper and lower bound on eigenvalues of the log-density Hessian (Mangoubi and Smith, 2017).
- Running HMC in practice requires knowing M and m:
  - efficient computation by *Lancoz iteration* through a small number of matrix-vector multiplications.

Table of Contents

Summary & Discussion

- Prior-preconditioned CG speeds up the conditional updates of $\boldsymbol{\beta}$, removing the computational bottleneck of sparse Bayesian regression.

Summary & Discussion

- Prior-preconditioned CG speeds up the conditional updates of $\boldsymbol{\beta}$, removing the computational bottleneck of sparse Bayesian regression.
- Iterative methods are promising approaches for accelerating MCMC involving high-dimensional (conditionally) Gaussian distribution because:
  - sparsity in $\boldsymbol{X}$ is natively exploited.
  - approximations to the target covariance make for preconditioners.
  - covariance structures are related from one iteration to another.

Summary & Discussion

- Prior-preconditioned CG speeds up the conditional updates of $\boldsymbol{\beta}$, removing the computational bottleneck of sparse Bayesian regression.
- Iterative methods are promising approaches for accelerating MCMC involving high-dimensional (conditionally) Gaussian distribution because:
  - sparsity in $\boldsymbol{X}$ is natively exploited.
  - approximations to the target covariance make for preconditioners.
  - covariance structures are related from one iteration to another.
- Upcoming: sparse Bayesian regression with log-concave likelihoods.

References

Nishimura, A. and Suchard, M. (2018)
Prior-preconditioned conjugate gradient for accelerated Gibbs sampling in "large n & large p" sparse Bayesian logistic regression models.
arXiv:1810.12437

Python package "bayesbridge" available from PyPI

Source code available at
`https://github.com/aki-nishimura/bayes-bridge`

The End

Thank you!