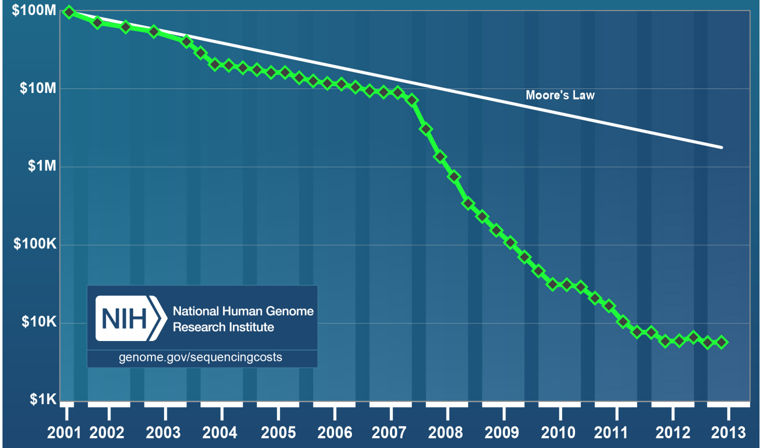
# "All the genomes in the world"
## Scalable Bayesian Computation using emulation

Daniel Lawson

Heilbronn Institute, University of Bristol
work with Niall Adams, Imperial College London

2014

## Cost per Genome



## Background

- We will ‘soon’ be able to sequence all the genomes in the world for less than the cost of the logistics of obtaining or processing them
- Current project to sequence all 50K Faroe Islanders
- What would we do with ‘all the genomes in the world’?
- Can we run appropriate models on them?
- More modest goal: scalable framework for statistical methodology
- This includes MCMC, which I will use here
- But can be replaced by optimisation for really scary volumes of data

## Motivation

For Large datasets

- "Statistics doesn't work" – estimates get worse as we get more data! (for linear compute)
  - e.g. MCMC
- Simple analytics can extract many useful features
  - e.g. K-medians clustering, etc
- Informative in practice - and still hard to get working!
- *Exploit averaging* over lots of data
- But many interesting quantities are subtle...
- or local, so we only have a small amount of data about them
- Always a place for models closer matching reality

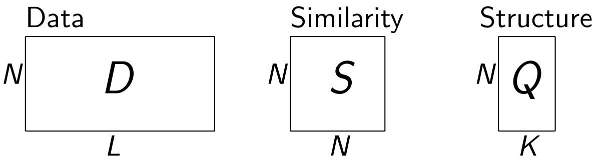
## Model of interest: FineSTRUCTURE

Clustering $Q$ with associated uncertainty from SNP data $D$

- The $N$ Individuals are highly structured (Populations)
- The $L$ SNPs are complexly correlated
- $D|S$ describes individuals with a Hidden Markov Model with each other individual as the hidden state
- $S_i$ is the vector of expected times each individual is transitioned to in the HMM
- MCMC inference for $Q$ using genetics model $S|Q$
- $S|Q$ is approximately multi-variate normal with structured covariance
- Problem: $S$ is $O(LN^2)$ to evaluate

## Similarity

$$p(D|S)p(S|Q)p(Q)$$



- Compare $N$ items about which we have a large amount of data $D$ (trivial extension: to $M = O(N)$ other items).
- Similarity $S(i,j)$ is computationally costly to evaluate
- $S$ structured by a model $Q$
- i.e. Similarity model $p(D|S)$ separates the data $D$ from the structure model $p(S|Q)$
  - If rows of $Q$ sum to 1 this is a mixture model
  - if only 1 element is non-zero it is a partitioning

## Random or convenience filtering

- See 'big data'[1] as better sampling of data
- Why not throw away elements from $D$?
  - Convenience sampling - what can we measure? $=$'data'
  - Systematic sampling - retain every n-th data point
  - Simple random sampling - retain fraction $p$
  - Stratified sampling
  - etc
- For example:
  - Use $L' \ll L$
  - Use $N' \ll N$
- Can fix $N'$ and $L'$ to fix computational cost

1: *Big data: any data that can't be processed in memory on a single high spec computer*

## Emulated Likelihood Models (ELMs)

Fundamental idea: Replace $p(D|S,\theta)$ in

$$p(D|S,\theta)p(\theta)p(S|Q,\phi)p(Q,\phi)$$

using $S^*$ computed and $S^\dagger$ emulated similarities, s.t. $S = S^* \cup S^\dagger$:

$$\hat{p}(D|S,\theta) = p(D|S^* \cup S^\dagger,\theta) = \int p(D|S^* \cup S^\dagger,\theta,\psi)p(S^\dagger|S^*,\psi)d\psi$$

- $S_{ij}$ is costly to compute, and needed for $S|Q$
- But are highly structured (e.g. clusters)
- So can emulate $S_{ij}$ rather than computing
- Choose $S^*$ to be approximately sufficient for $p(D|S,\theta)$
- Carefully downweight emulated data
- Weights are only modification to $S|Q$

## The oracle

Which $NT$ elements $S^*$ of $S$ should we evaluate?
Natural solution using oracle knowledge of the true $S$.

- Use a loss function $\mathcal{L}(S^*)$
- Seek $\operatorname{argmin}_{S^*(T)} \mathcal{L}(S^*(T))$
- Choices for $\mathcal{L}$ might be:
  - KL divergence between the true and the emulated posterior
  - A loss based on model usage (e.g. is the clustering correct? Control the false positive rate, etc)
- Choose $T$ based on acceptable loss

## Building blocks of a real Emulated Likelihood Model

The oracle is too costly. We instead must:

- Iteratively choose the next $S_{ij}$ to add to $S^*$
- From a limited set produced by a restriction operator $\mathcal{R}(S^*)$
- Construct an estimator $\hat{\mathcal{L}}$ for $\mathcal{L}$
- Decide on the next point to evaluate using $\operatorname{argmin}_{S_{ij}} \mathbb{E}\left(\hat{\mathcal{L}}(S^* \cup S_{ij}) | S^*\right)$
    - We can consider different histories to evaluate performance
- Stopping rule: convergence of $\hat{\mathcal{L}}$

$\hat{\mathcal{L}}$ can be implicit from $\mathcal{R}$, if it returns points ordered by $\mathbb{E}(\hat{\mathcal{L}})$

## The emulator

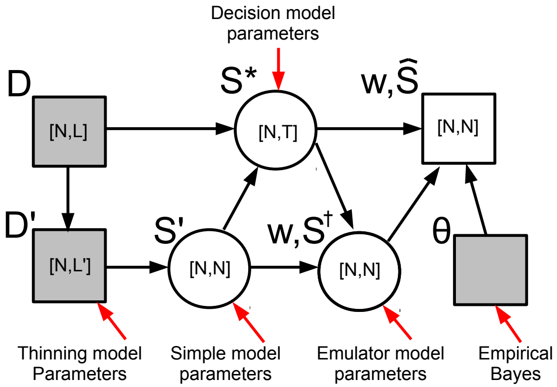
'Machine learning' emulator with the usual caveats:

- Rarely optimal, rarely unbiased
- Prediction error estimated using online cross validation
- Respects computational constraints:
  - $L \gg N$: Consider $O(N^2 + LN)$ algorithms
  - $N \gg L$: Consider $O(LN)$ algorithms
  - Massive data: Consider $O(LN^\alpha + N)$ algorithms with $\alpha < 1$.
- Yet ... Low rank similarity matrices can be nearly losslessly reconstructed*

*Candes & Plan 'Matrix completion with noise', Proc. IEEE, 2010

## Fast finestructure - outline

Emulator:

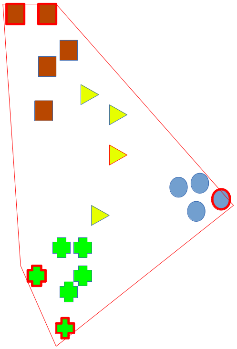


*S* becomes the data for inferring *Q*:

$$\left[p(D|\hat{S},\hat{\theta})\right] p(S|Q,\phi)p(Q,\phi)$$

## Fast finestructure - decision and emulation

- Decision $\mathcal{R}$: Choose item to evaluate $i_t^*$ using the *point most distant to all evaluated points*
- We are evaluating $S^*$ in entire rows
- Emulation: Predict $S^\dagger_{\cdot i}$: mixture model $S_i^* = \mathrm{MM}(S_{i*}^*)$, and regression on $\hat{S}_{ij} = \mathrm{LM}(S_{ij}^*, S_{ij}')$
- Implicit loss function $\mathcal{L}$:
  - Minimise the *maximum* prediction error
  - Finds outliers and clusters
  - $S^*$ form convex hull of $S$



## Fast finestructure - in practice

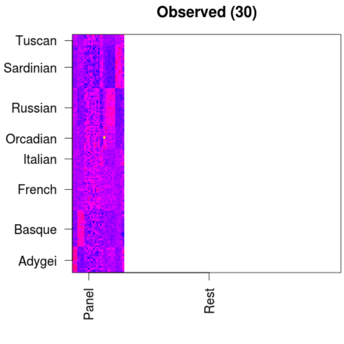
- Computation of $S$ costs $O(N^2L' + NT^2L) \ll O(N^2L)$
- Current datasets: $L = 10,000,000$, $N = 5000$, $L' = 10,000$ $T = 100$, predicted saving ratio is 100

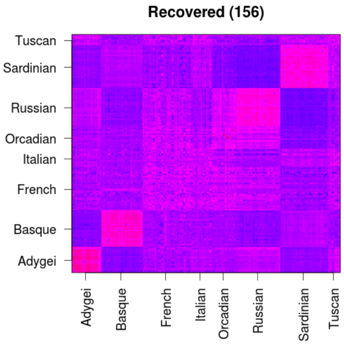
We can save up to factor 10000 by reducing $T$, the cost of the linearised model on the reduced dataset. $L$ will not grow beyond this, but $N$ will - can introduce an emulation step for $S'$

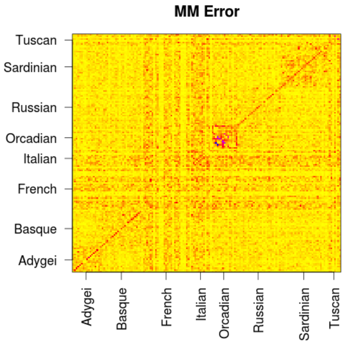
## Fast finestructure - Parallel MCMC algorithm

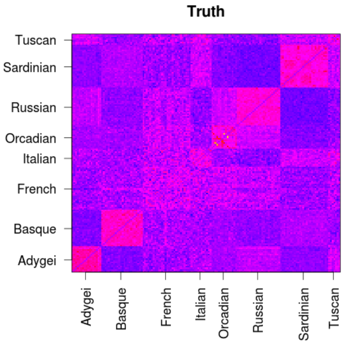
A parallel tempering algorithm for when MCMC parallelises poorly

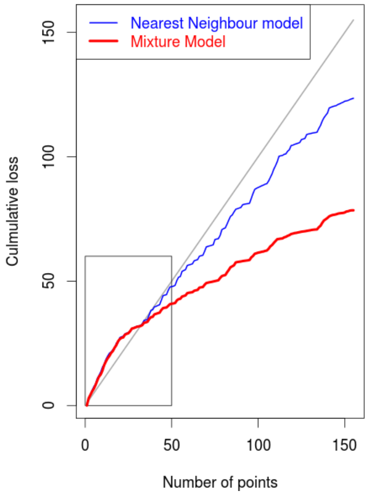
- Evaluate the unlinked model $S'$
- Master node: perform MCMC clustering to find $\hat{Q}_t$ using $\hat{S}_t$, when there are $t$ rows $S^*_t$ computed
- Worker nodes compute $S^*_{\cdot i}$ in the order chosen by the master
- Stopping rule: posterior distribution of $\hat{Q}$ converges
  - No new information added when increasing $t$
  - (Or if the MCMC is slower than the evaluation of $S$, sometime afterwards)


Observed (30)
Tuscan
Sardinian
Russian
Orcadian
Italian
French
Basque
Adygei
Panel
Rest

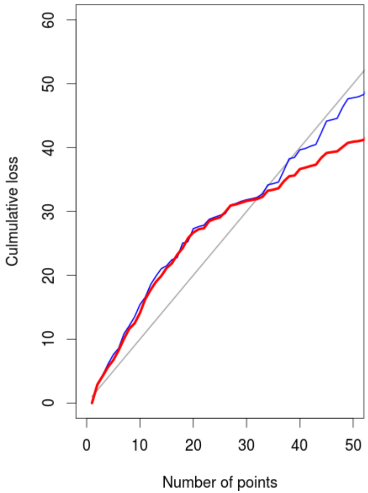

Recovered (156)
Tuscan
Sardinian
Russian
Orcadian
Italian
French
Basque
Adygei
Adygei
Basque
French
Italian
Orcadian
Russian
Sardinian
Tuscan


MM Error
Tuscan
Sardinian
Russian
Orcadian
Italian
French
Basque
Adygei
Adygei
Basque
French
Italian
Orcadian
Russian
Sardinian
Tuscan


Truth
Tuscan
Sardinian
Russian
Orcadian
Italian
French
Basque
Adygei
Adygei
Basque
French
Italian
Orcadian
Russian
Sardinian
Tuscan



Nearest Neighbour model
Mixture Model
Culmulative loss
Number of points
0
50
100
150



Culmulative loss
Number of points
0
10
20
30
40
50
60

## Emulated Likelihood Models for general Bayesian problems

General emulation for big (but not so big) problems

$$\hat{p}(D|S,\theta) = \int p(D|S^* \cup S^\dagger, \theta) p(S^\dagger|S^*, \theta, \psi) d\psi$$

- i.e. Can use $\theta$ to emulate $S^\dagger(\theta)$ - e.g. regression in $(S, \theta)$ space
  - Gaussian Process for $S_{ij}(\theta)$ is a natural choice
- If $S^\dagger$ is an unbiased estimator of $S^*$ this is a pseudo-marginal approach (and hence targeting the correct posterior)

## Discussion

- Goal: Approximate answers to the right questions using exact answers to the wrong questions
- Machine learning is increasingly important for large datasets
- Statistical modelling still has a place
- Proposed the Emulated Likelihood Model:
    - Full statistical modelling
    - Machine learning algorithms used for the calculation
    - Statistical estimation of parameters is retained but approximated