Weekly Friday Meeting Slides for Subteam 1

Christina Catlett & Maya Watanabe
HMC REU SUMMER 2020

Fri May 22

Astrostats Tutorial

- Introduction to Bayesian parameter estimation:
  - Treat both data and parameters probabilistically according to Bayes Theorem $\longrightarrow$ determine the most likely parameters given the data and the model (posterior)

$$P(\theta|D,M) = \frac{P(D|\theta,M)P(\theta|M)}{P(D|M)}$$

- $P(\theta|D,M)$ - posterior
- $P(D|\theta,M)$ - likelihood
- $P(\theta|M)$ - prior
- $P(D|M)$ - normalization constant

Astrostats Tutorial (cont.)

- Task is non-trivial when the posterior distribution is non-standard → Monte Carlo Simulation
  - Monte Carlo Simulation: Simulating events to derive an approximate probability distribution
- *Curse of dimensionality*: difficult to sample efficiently
- Markov Chain Monte Carlo Methods (MCMC): Sample candidate parameters according to the likelihood, compare the probability of seeing these candidate parameters vs. the last sampled parameters: P($\boldsymbol{\theta}$| M,D), accept the candidate according to some acceptance criteria, iterate until a steady state is reached (i.e most probable parameters found).

MCMC Algorithm

- Metropolis MCMC
- choose θ^0 (initial guess for parameter values)
  - calculate posterior
- for each iteration of the Markov Chain:
  - Sample random candidate θ^* from proposal distribution (Gaussian)
    - calculate posterior
  - Accept or reject the proposed θ^* based on the value of a ratio
    - $\varrho = \dfrac{posterior\ of\ \theta^*}{posterior\ of\ current\ \theta}$

      if $\varrho \geq 1$: accept the transition

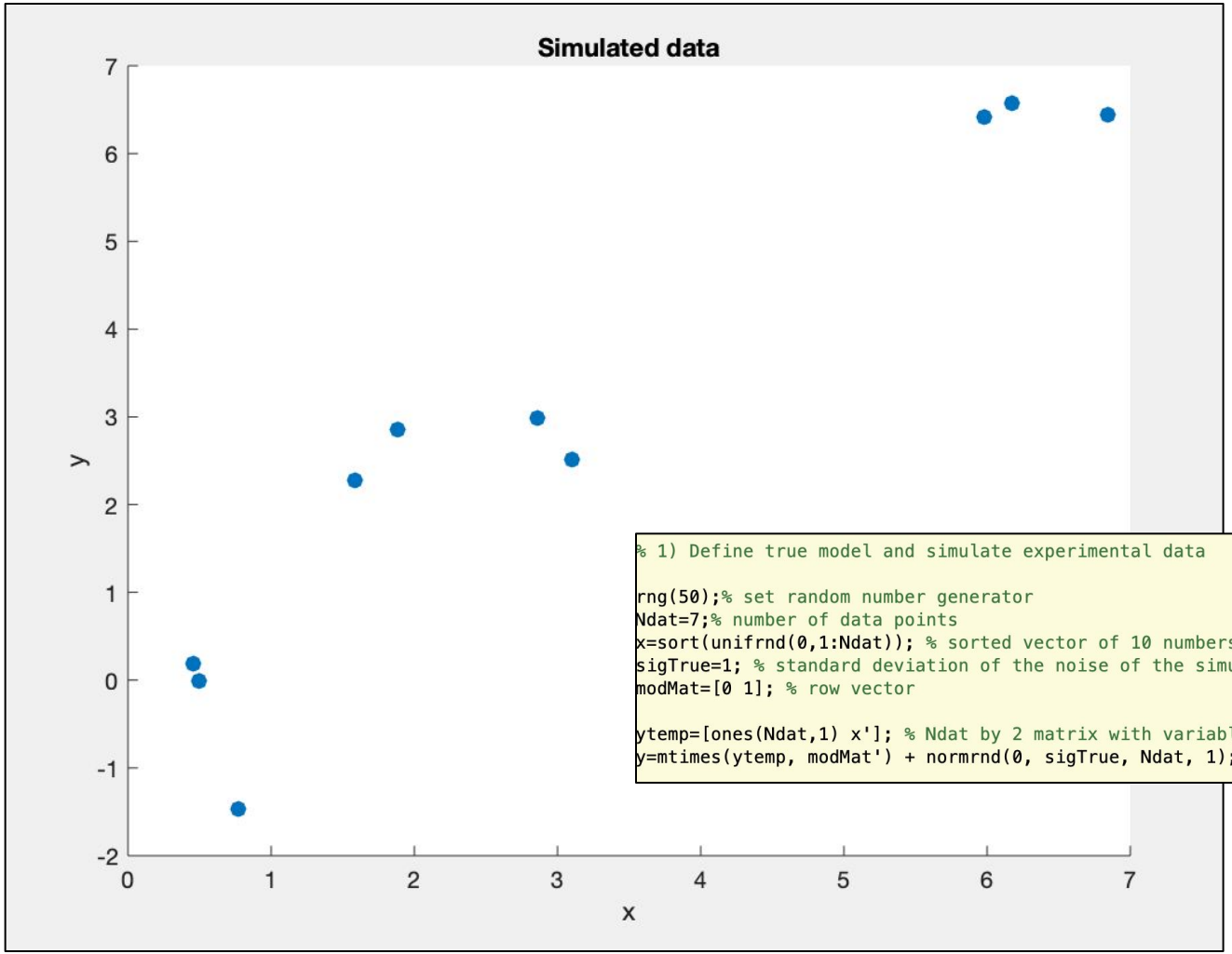
      if $\varrho < 1$: only accept with a probability of ϱ

- stop after sufficient # of samples (10,000 in our case)

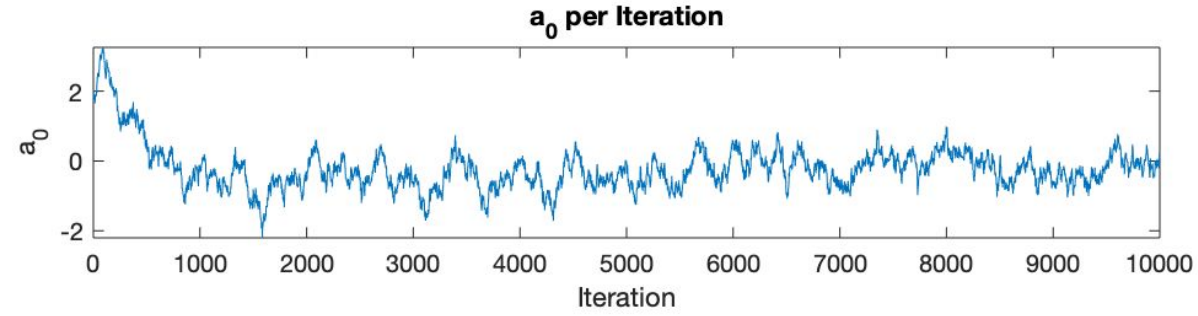
Compute pdf of parameters using MCMC parameter samples

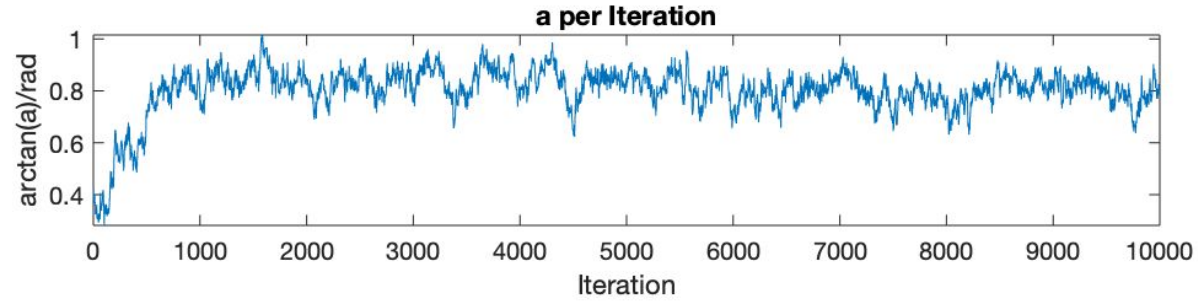
Our model

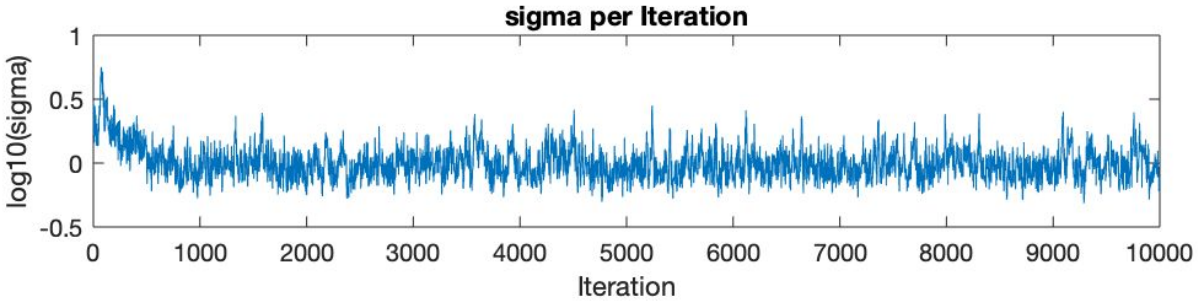
- Linear model with unknown noise
  - parameters $\boldsymbol{\theta}$: a_0 (y-intercept), a_1 (slope), $\boldsymbol{\sigma}$ (st. dev. of Gaussian noise)
- Assume
  - Normal likelihood
  - Uniform prior
  - 10 original data points, 10^4 iterations, 0 burn-in
  - Given covariance matrix, $\boldsymbol{\theta}_{\text{initial}}$= [2, 0.3927, 0.4771]
- Goal
  - fit a linear model to two dimensional data (find the best fitting line)




```
% 1) Define true model and simulate experimental data

rng(50);% set random number generator
Ndat=7;% number of data points
x=sort(unifrnd(0,1:Ndat)); % sorted vector of 10 numbers between 0 and 10
sigTrue=1; % standard deviation of the noise of the simulated data y (113)
modMat=[0 1]; % row vector

ytemp=[ones(Ndat,1) x']; % Ndat by 2 matrix with variable x data from line 108 in the second column
y=mtimes(ytemp, modMat') + normrnd(0, sigTrue, Ndat, 1); % final data (points) with noise added
```





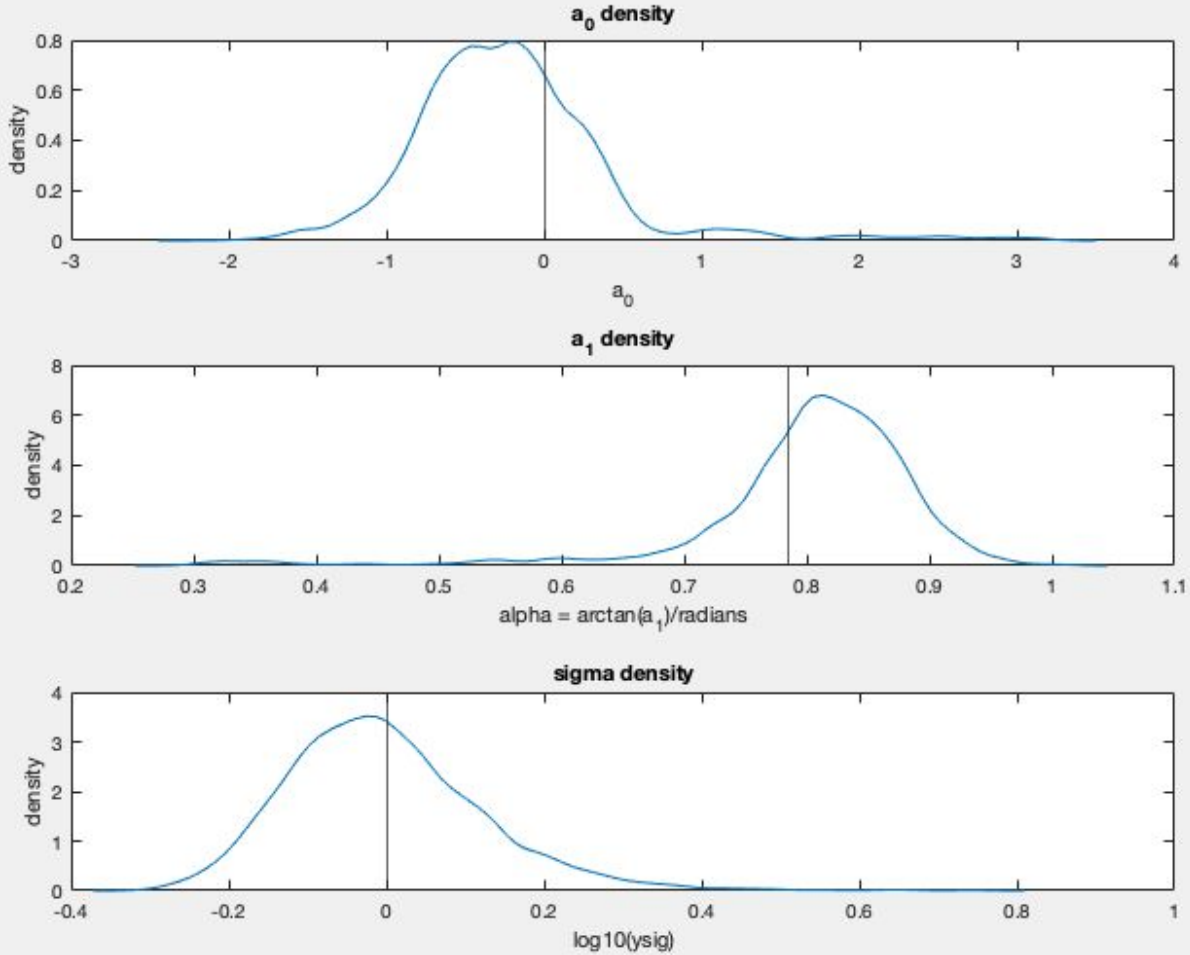


θ_{true}= [0,.7854,0]

θ_{MAP}= [-0.0979,0.7870, -0.0193]

θ_{mean}= [-0.0419,0.7110,0.0279]

- $a_0 \rightarrow$ intercept
- $a \rightarrow$ slope
- $\sigma \rightarrow$ Gaussian noise

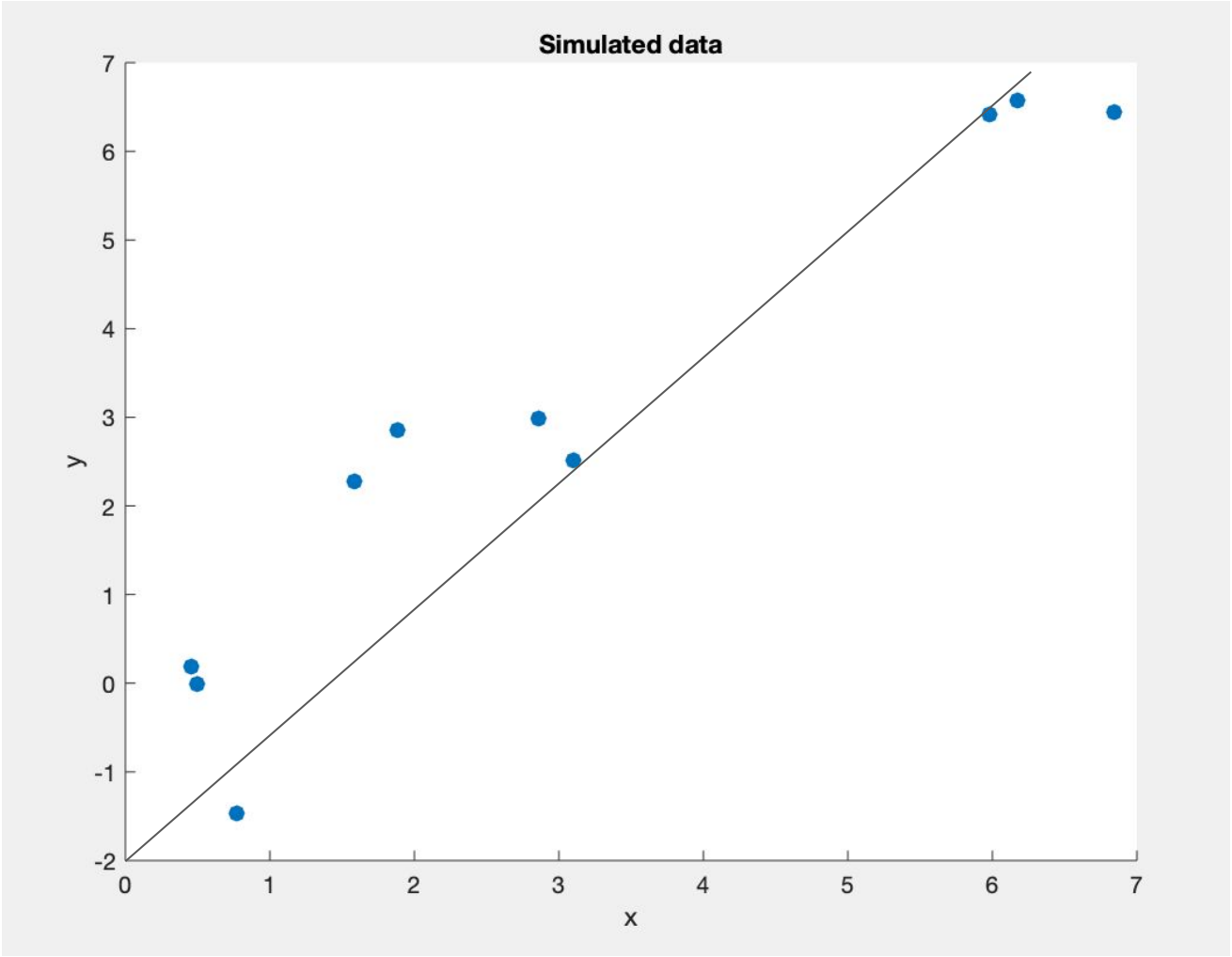


Posterior density distributions (pdf) for each parameter

- vertical lines indicate true parameters ($\boldsymbol{\theta}_{\text{true}}$)

- $a_0 \rightarrow$ intercept
- $a \rightarrow$ slope
- $\boldsymbol{\sigma} \rightarrow$ Gaussian noise

$\boldsymbol{\theta}_{\text{true}} = [0, 0.7854, 0]$


Simulated data
7
6
5
4
3
2
1
0
-1
-2
y
0
1
2
3
4
5
6
7
x

Still to do

- Use MCMC to make predictions (determined posterior pdf)
  - Given new data, what does the model predict?
  - Need to find the entire posterior pdf over the model's prediction
- Apply strategy to other models (eventually systems of diff eq)

Fri May 29

Overview

1. Read ‘MCMC Techniques for Parameter Estimation of ODE Based Models in Systems Biology’
2. Further documented the four discussed MCMC algorithms
3. Began process of using each to fit a simple Lotka Volterra model with sample data

More about the prior distribution

- Recall Bayes' Theorem

$$P(\theta|D,M) = \frac{P(D|\theta,M)P(\theta|M)}{P(D|M)}$$

$P(\theta|D,M)$ - posterior

$P(D|\theta,M)$ - likelihood

$P(\theta|M)$ - prior

$P(D|M)$ - normalization constant

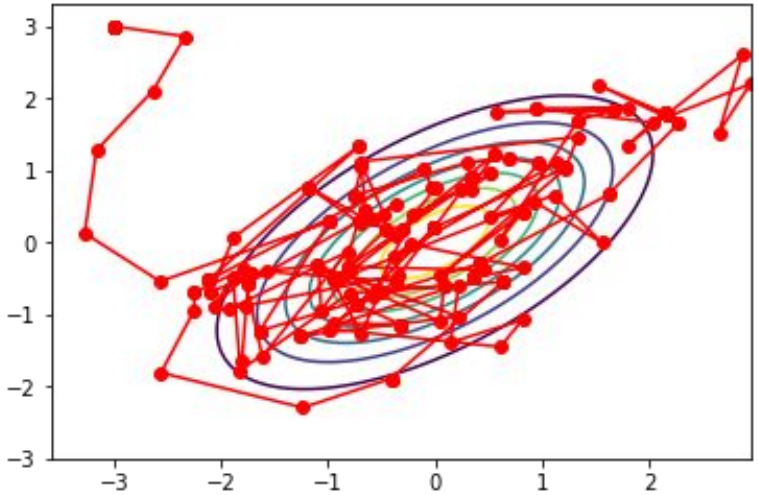
- The *likelihood* is the way in which we want to sample our candidate parameters (based on the assumptions about the noise of the model)
- The *prior* is the initial, general distribution that we think our parameter values are falling under
    - The more we sample, the more we "carve" away parameter values to reveal the true distribution

Valderrama-Bahamòmdez et al., 2019

- Evaluated four different MCMC algorithms for convergence time, accuracy in parameter estimation, sensitivity to informative/non-informative priors, and run time.
  - Metropolis-Hastings (last week)
  - Single Chain Adaptive
  - Parallel Tempering
  - Parallel Adaptive

Single-chain MCMC

1) Metropolis-Hastings



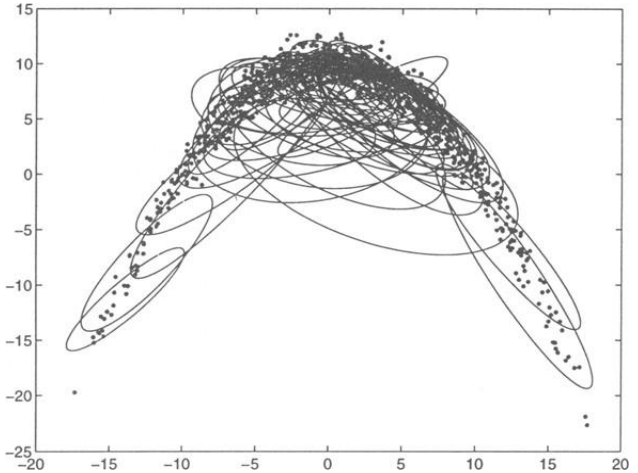
MH

Acceptance ratio:

$$\rho = min(1, \frac{post\,(\theta_{prop})}{post\,(\theta_{cur})})$$

2) Adaptive

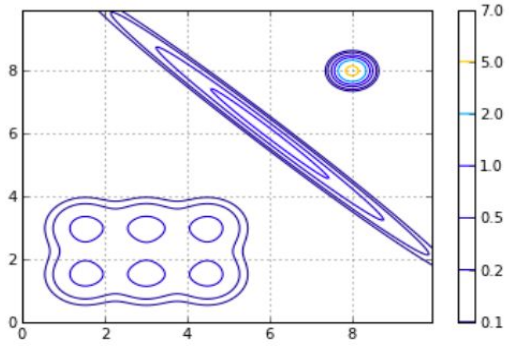
- sample parameters *X* times (MH alg)
- recompute likelihood function using the *X* samples
- draw a new candidate using the new likelihood



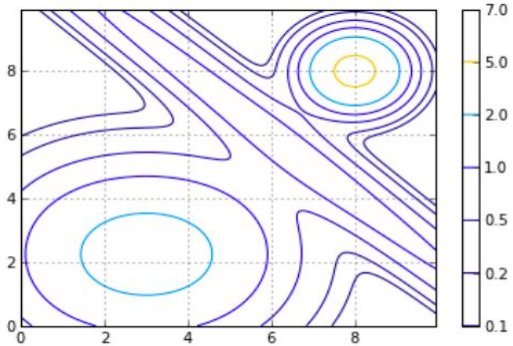
Adaptive

$$\alpha = min(1, \frac{post_{new}(\theta_{prop})}{post_{new}(\theta_{cur})})$$

Parallel MCMC



(a) Example Distribution, t = 1



(b) Example Distribution, t = 10

1) Parallel Tempering

- Multiple chains at different "temperatures" - *T*
- *T* divides the likelihood function of MH

$$posterior = \frac{likelihood}{T}(prior)$$

- Larger *T* = more likely to reject candidate parameter set
  - Sample from a larger space
- Smaller *T* = more likely to accept a candidate parameter set
  - Sample more locally
- Allows Markov chain to explore regions of lower probability
- Swap chains of samples to increase *T*

MH acceptance ratio:

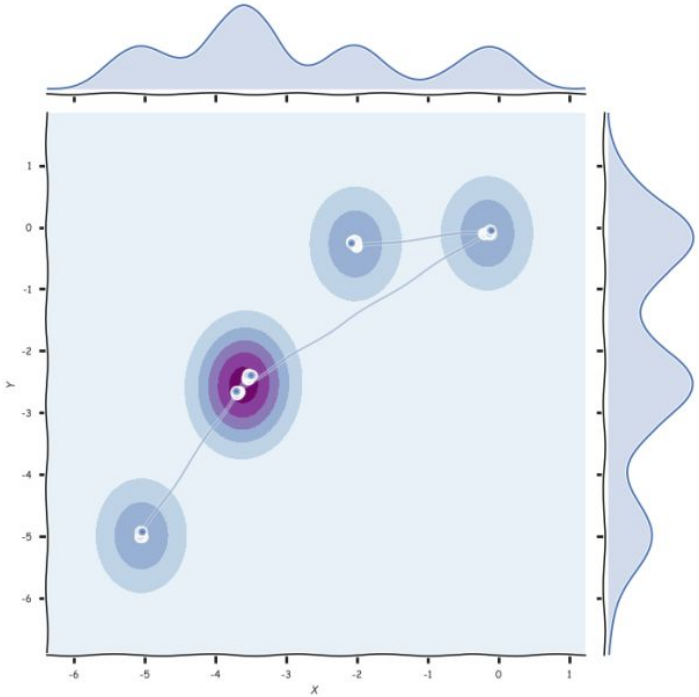
$$\rho = min(1, \frac{post\,(\theta_{prop})}{post\,(\theta_{cur})})$$

Swap ratio:

$$\alpha = min(1, \frac{post_T(\theta_{T+1})post_{T+1}(\theta_T)}{post_T(\theta_T)post_{T+1}(\theta_{T+1})})$$

Parallel MCMC

2) Adaptive Parallel Tempering



- Multiple chains:
  - adaptive MCMC algorithms
    - update likelihood
  - tempering the chains
    - update likelihood + acceptance criteria

Acceptance ratio:

$$\rho = min(1, \frac{post_{new}(\theta_{prop})}{post_{new}(\theta_{cur})})$$

Swap ratio:

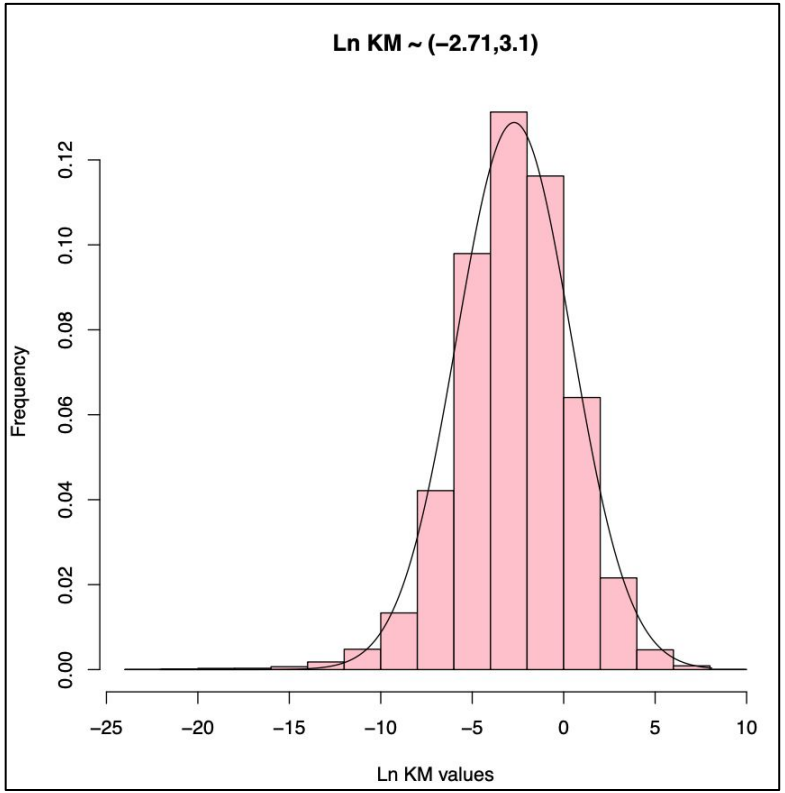
$$\alpha = min(1, \frac{post_{T}(\theta_{T+1}) post_{T+1}(\theta_{T})}{post_{T}(\theta_{T}) post_{T+1}(\theta_{T+1})})$$

Convergence

- Assessing point where chain converges to largely-stable parameter values
  - Informs sufficient number of burn-in iterations
- Convergence diagnostics:
  - Geweke’s Test
    - Difference in mean of first 10% of chain, last 50% of chain
    - Convergence when close to 0
    - Run on equally sized subsections of chain
  - Gelman-Rubin
    - Measure of sameness for within-chain, between-chain variance

Evaluation of Estimated Parameters

- Effective Sample Size (ESS)
- Fraction of true parameters falling within the 95% Bayesian credible interval
- Effect of informative vs. uninformative priors:
  - Specific interest for large noise levels
  - Priors estimated by fitting Gaussian distribution to data from BRENDA database



Conclusions

- Designing a good sampling algorithm involves: sufficiently fast convergence, high ESS, and a high probability to converging to true parameters

Convergence: Adaptive models converge in fewer iterations than MH, tempering. No observed relationship between noise level and convergence failure.

ESS: Parallel models produce larger ESS

Informative priors: No clear effect on convergence, ESS, but increases credibility for single-chain methods

Credibility: Parallel adaptive most robust against noise

Fitting the Lotka-Volterra ODE to MH

Lotka-Volterra

- Simple predator-prey model

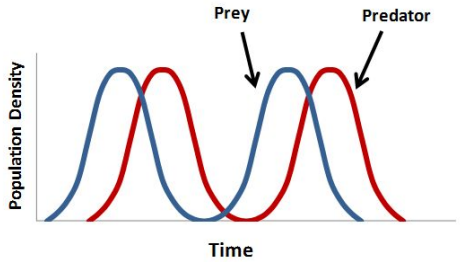
Lotka-Volterra Equations

$$\frac{dP}{dt} = -Pm + bHP$$

$$\frac{dH}{dt} = Hr - aHP$$

$\begin{cases} P = P(t) & \text{Number of Predators} \\ H = H(t) & \text{Number of Prey} \end{cases}$

$\begin{cases} r > 0 & \text{Birth Rate of Prey} \\ m > 0 & \text{Death Rate of Predators} \\ a > 0 & \text{Death Rate of Prey/Predator} \\ b > 0 & \text{Birth Rate of Predators/Prey} \end{cases}$



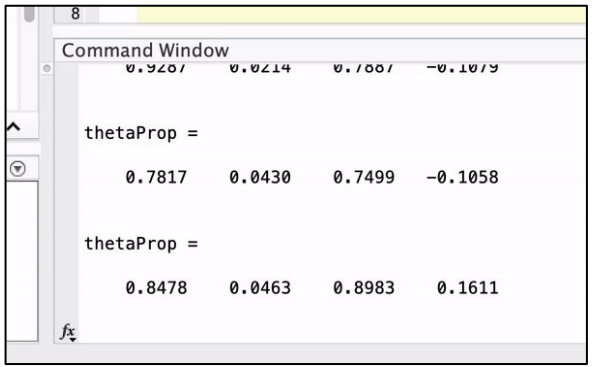
<u>Goal:</u> Run our MH MCMC alg. using the L-V model and given data to produces parameter distributions

<u>Adjustments:</u>

- Parameters (4)
- Use an ODE solver to determine model
- Likelihood (Gaussian) & prior (uniform)
  - Both distributions chosen for simplicity
  - Made adjustments to the functions that did this
- Covariance matrices

Questions

It “works” but…




```
%%% To thin ODE data to match observed dataset length
function ymat_filt = thin(tmat, ymat)
    % Number of datapoints given w ODE solver
    ymat_length = length(ymat(:,1));

    % Number of observed datapoints = final time - initial time + 1
    num_datapoints = tmat(length(tmat)) - tmat(1) + 1;

    % Select every nth datapoint from ODE solver
    % n = round down(length of ODE output / dataset length)
    n = floor(ymat_length/num_datapoints);
    ymat_filt = ymat(n:n:end,:);
    ymat_filt = ymat_filt(2:(length(ymat_filt(:,1))-1),:);
    %%ymat_filt = ymat(1:91,:);
end
```

- Trouble comparing ‘continuous’ results of ODE solver with discrete time data:
  - How do we thin the 469 data points given by ODE solver to the 91 that we need?
- Problems:
  - ODE solver does not solve at whole seconds
  - 91 ≠ 469

Moving Forward

- Run other 3 MCMC algorithms for Lotka-Volterra model
- Read Collis, 2017:
  - ‘Bayesian Calibration, Validation and Uncertainty Quantification for Predictive Modelling of Tumour Growth: A Tutorial’
  - Calibrating and applying VVUQ to MCMC models
    - VVUQ: verification, validation, uncertainty quantification
- Look at delayed rejection model for MCMC (for T1D model!)

Fri June 5

Overview

1. Parameterizing Lotka Volterra System
    a. Metropolis Hastings
        i. Tutorial Code
        ii. Built-in
    b. Delayed Rejection Adaptive Metropolis (DRAM)
        i. *Smith*, 2014: Chapter 8
        ii. ‘mcmcstat’ library

Tutorial Code

1. Set number of chain elements M and design parameters n_s, σ_s
2. Determine $q^0 = \arg\min_q \sum_{i=1}^{n} [v_i - f_i(q)]^2$
3. Set $SS_{q^0} = \sum_{i=1}^{n} [v_i - f_i(q^0)]^2$
4. Compute initial variance estimate: $s_0^2 = \frac{SS_{q^0}}{n-p}$
5. Construct covariance estimate $V = s_0^2[\mathcal{X}^T(q^0)\mathcal{X}(q^0)]^{-1}$ and $R = \text{chol}(V)$
6. For $k = 1, \ldots, M$
   (a) Sample $z_k \sim N(0, I_p)$
   (b) Construct candidate $q^* = q^{k-1} + Rz_k$
   (c) Sample $u_\alpha \sim \mathcal{U}(0, 1)$
   (d) Compute $SS_{q^*} = \sum_{i=1}^{n} [v_i - f_i(q^*)]^2$
   (e) Compute
   $$\alpha(q^*|q^{k-1}) = \min\left(1, e^{-[SS_{q^*} - SS_{q^{k-1}}]/2s_{k-1}^2}\right)$$
   (f) If $u_\alpha < \alpha$,
   Set $q^k = q^*$, $SS_{q^k} = SS_{q^*}$
   else
   Set $q^k = q^{k-1}$, $SS_{q^k} = SS_{q^{k-1}}$
   endif
   (g) Update $s_k^2 \sim$ Inv-gamma(a_{val}, b_{val}), where
   $a_{val} = 0.5(n_s + n)$, $b_{val} = 0.5(n_s\sigma_s^2 + SS_{q^k})$

$$\mathcal{X}_{ik}(q) = \frac{\partial f_i(q)}{\partial q_k}.$$

- Finished from last week
- Followed code from ‘Uncertainty Quantification’
  - Clarification on the proposal distribution:

$$J(q^*|q^{k-1})$$

1. Denotes probabilistic choice of new parameters, q*, based on current, q^(k-1)
2. Defined by the n x p sensitivity matrix of the parameters

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Construct the sensitivity matrix chi, covariance matrix V.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  chi = senseq(q_init, t, pops_init);
  V = inv(chi'*chi)*var;

  % Ensure sensitivity matrix is positive-definite
  try
    R = chol(V);
  catch
    R = chol(nearestSPD(V));
  end
```

Tutorial Code

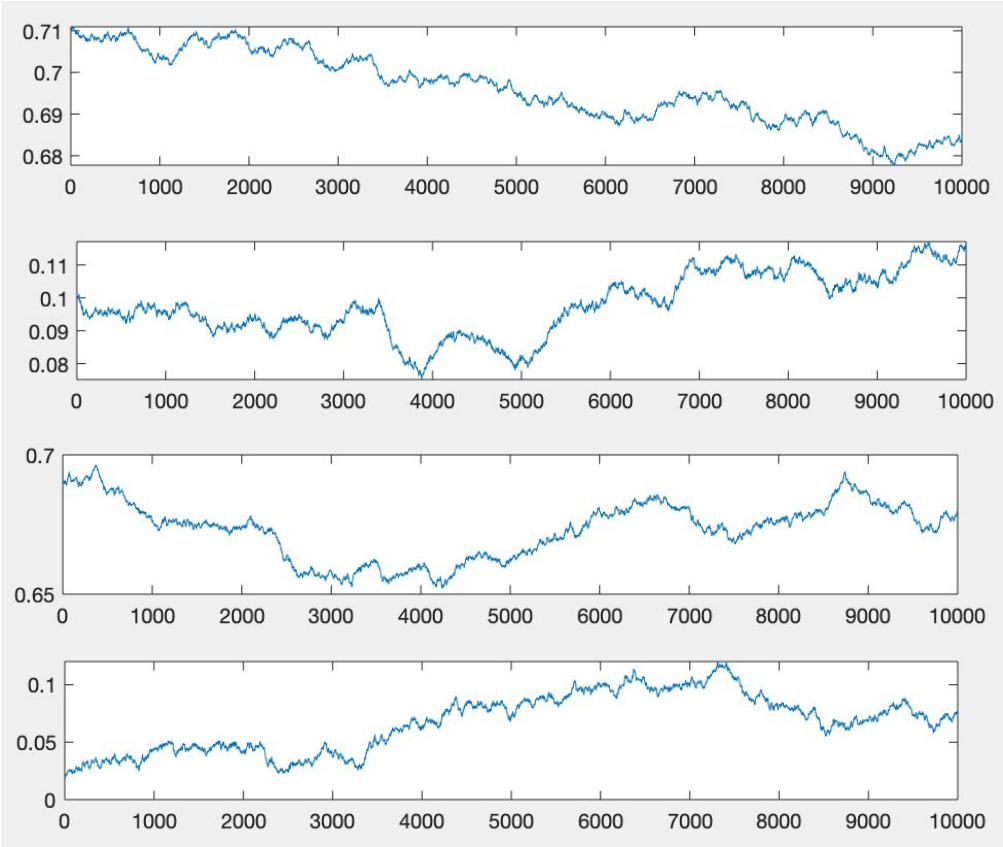
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Solve ODE for initial params
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
model_sltn = model_sol(q_init,t,pops_init)';
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Construct the parameters used when sampling the error variance, adapting covariance matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n0 = .001;
sigma02 = var;
aval = 0.5*(n0 + n);
bval = 0.5*(n0*sigma02 + SS_old);
sigma2 = 1/gamrnd(aval,1/bval);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Construct the sensitivity matrix chi, covariance matrix V.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

chi = senseq(q_init, t, pops_init);
V = inv(chi'*chi)*var;

% Ensure sensitivity matrix is positive-definite
try
  R = chol(V);
catch
  R = chol(nearestSPD(V));
end

Hres = popsdens(:,1) - model_sltn(:,1);
Lres = popsdens(:,2) - model_sltn(:,2);
Hres_T = Hres';
Lres_T = Lres';
% Initial variance estimate
SS_old = Hres_T*Hres + Lres_T*Lres;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Construct a Metropolis chain of length N
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
q_old = q_init;
err = 0;
 for i = 1:N
     % Adapt covariance matrix
     if mod(k0, i) == 1
     end

     % Construct new candidate q
     q_new = mvnrnd(q_old, R);

     % Solve model with candidate q
     new_model_sol = model_sol(q_new, t, pops_init)';

     % Compute new variance, proposal evaluation
     new_Hres = popsdens(:,1) - new_model_sol(:,1);
     new_Lres = popsdens(:,2) - new_model_sol(:,2);
     new_Hres_T = Hres';
     new_Lres_T = Lres';
     SS_new = Hres_T*Hres + Lres_T*Lres;

   proposal = exp(-.5*(SS_new-SS_old)/sigma2);
   alpha = min(1,proposal);

   % Set acceptance criteria
   u_alpha = rand(1);

    % Accept or reject candidate, fill in chain
    if u_alpha < alpha
      Q_MCMC(:,i) = [q_new'; SS_new];
      q_old = q_new;
      SS_old = SS_new;
    else
      Q_MCMC(:,i) = [q_old'; SS_old];
    end
    Sigma2(i) = sigma2;
    bval = 0.5*(n0*sigma02 + SS_old);
    sigma2 = 1/gamrnd(aval,1/bval);
 end
```

Tutorial Code for Lotka-Volterra



a

b

c

d

```
>> mean(Q_MCMC')

ans =

    0.6731    0.0672    0.6961    0.0976
```

MATLAB *mhsample*

Set up for running *mhsample*

- adapted from online tutorial (not ODE example)
- not many papers use *mhsample* to parameterize ODEs

1. Load and correct data
2. Establish:
   a. Number of parameters
   b. Initial parameter guess
   c. Number of samples
   d. Burn-in period
   e. Thinning interval
3. Implement
   a. Proposal distribution (log)
   b. Likelihood distribution
   c. Prior distribution
4. Run *mhsample*

MATLAB *mhsample*

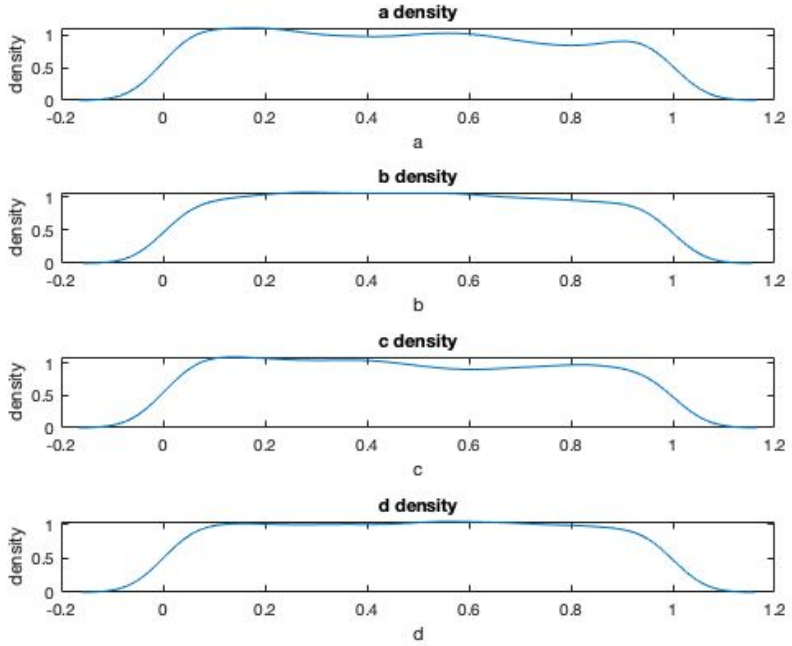
```
% ========================================================================
% Initialize constants
% ========================================================================
% initial parameter estimate
thetaInit=[.7 .1 .7 .1];

% number of parameters in likelihood equation)
n=4;

% sigma for the proposed logPDF, an nxn identity matrix
prop_sig=eye(n);

% number of randomly generated samples
nsamples = 20000;

% Establish burn-in period
K = 1000;

% This parameter controls the size of the new markov chain which omits M-1
% out of M values.  This will curb the effect of autocorrelation.
% M-1 out of M values omitted in the generated sequence
M = 10;
```

Initialize model constants and distributions

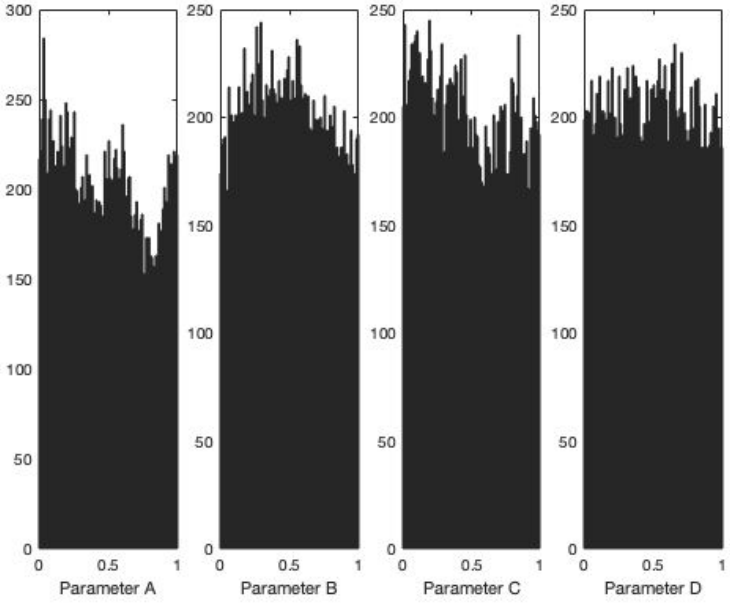
Run MCMC

```
% ========================================================================
% Implement proposal and target distributions
% ========================================================================
% DEFINE THE PROPOSAL DISTRIBTUION (logproppdf) and random number generator
% (proprnd).  The standard deviation for each parameter in the proprnd is
% to be chosen such that the updated guess covers a reasonable range of
% estimates.  These should be small if the initial estimate is a close one.
logproppdf = @(x,y) log10(mvnpdf(x,y,prop_sig)); % lognormal proposal dist.
proprnd = @(y) [normrnd(y(1),0.015),normrnd(y(2),0.04),...
    normrnd(y(3),0.0211),normrnd(y(4),0.04)]; % random sample from proppdf

% DEFINE THE LIKELIHOOD X PRIORS (posteriors)
% The prior distributions for the parameters are assumed to be
% non-informative, therefore they are treated as uniform distributions
% between 1e-5 and 1. These priors multiplied by the product
% of the lognormal PDF of each data point form the LikelihoodxPriors term
% that undergoes Bayesian updating.
% LL_mhsample is the function in which we compute the loglikelihood
% function
logpdf = @(x) log10(unifpdf(x(1),1e-5, 1))+log10(unifpdf(x(2),1e-5, 1))+...
    log10(unifpdf(x(3),1e-5, 1))+log10(unifpdf(x(4),1e-5, 1))+...
    (LL_mhsample(thetaInit, obsdata));

% ========================================================================
% MH-sampling routine
% ========================================================================
[result,accept] = mhsample(thetaInit,nsamples,'logpdf',logpdf,...
    'logproppdf',logproppdf,'proprnd',proprnd,'burnin',K,'thin',M);
```

Results of *mhsample*



Parameter posterior PDFs



Parameter sample distributions

Delayed Rejection Adaptive Metropolis (DRAM)

- DRAM uses techniques of adaptive MCMC

Adaptive Metropolis (AM):

1. Sample parameter sets for period of length k_0 (often 100) using initial covariance matrix V_0
2. At *kth* step update V_0 with information from sampled posteriors, new covariance matrix V
3. Continue sampling

Delayed Rejection (DRAM):

1. At each sample accept or reject proposed parameter set
   a. If accept —> 2.
   b. If reject: propose alternative candidate and pass to acceptance criterion
      i. If accept —> 2.
      ii. If reject —> b.
2. Continue to sample following AM

DRAM for Lotka-Volterra

- Based on code for algae growth example
- Using ‘mcmcstat’ library
  - Quantifies convergence
  - Plots densities
  - Plots predicted functions
- Still working to debug: ensure proper assumptions, consistency

DRAM for Lotka-Volterra

```
%%
% The model sum of squares in file <algaess.html |algaess.m|> is
% given in the model structure.

% The model sum of squares
% Incorporation of the likelihood function (normal)
model.ssfun = @lotkaVolterrassV1;
```

1. Establish likelihood function

```
%% Initialize parameter values
% All parameters are constrained to be positive.
params = {
    {'a', 0.7, 0, 1}
    {'b', 0.1, 0, 1}
    {'c', 0.7, 0, 1}
    {'d', 0.1, 0, 1}
    };
% Dif. initial theta options
% [0.6954 0.3865 0.5766 0.0150]
% [.7 .1 .7 .1]
```

2. Initialize parameters

```
%%
% We assume having at least some prior information on the
% repeatability of the observation and assign rather non informational
% prior for the residual variances of the observed states. The default
% prior distribution is sigma2 ~ invchisq(S20,N0), the inverse chi
% squared distribution (see for example Gelman et al.).

% Attempted uniform prior
model.S20 = [1 1];
model.N0  = [1 1];
```

3. Establish prior distribution

```
%%
% First generate an initial chain.
options.nsimu = 1000;
[results, chain, s2chain]= mcmcrun(model,data,params,options);
```

4. Perform burn-in

```
%%
% Then re-run starting from the results of the previous run,
% this will take couple of minutes.
options.nsimu = 5000;
[results, chain, s2chain] = mcmcrun(model,data,params,options, results);
```

5. Sample

```
%%
% Chain plots should reveal that the chain has converged and we can
% use the results for estimation and predictive inference.
figure(2); clf
mcmcplot(chain,[],results,'pairs');
figure(3); clf
mcmcplot(chain,[],results,'denspanel',2);
```

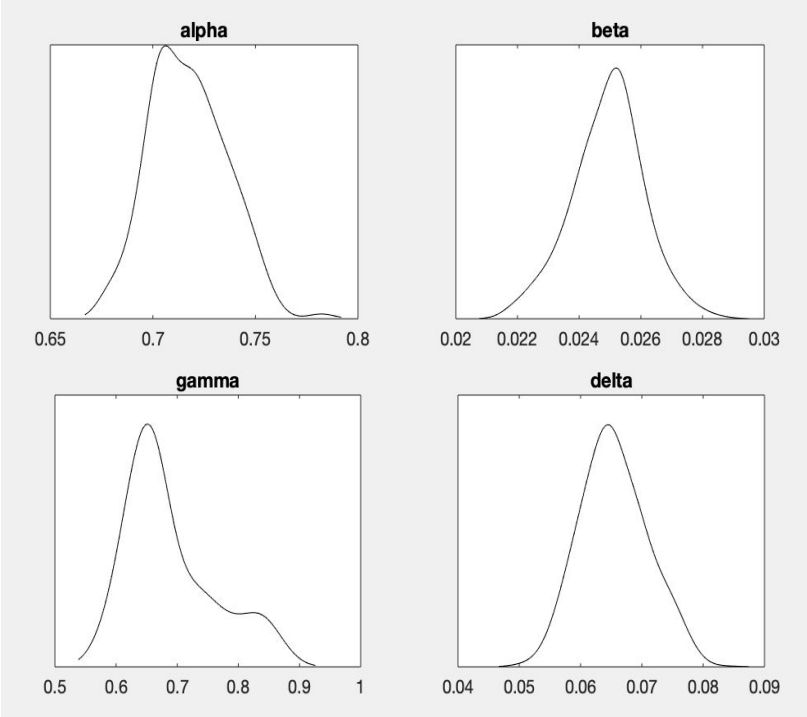
6. Plot & check convergence

```
%%
% Function |chainstats| calculates mean ans std from the chain and
% estimates the Monte Carlo error of the estimates. Number |tau| is
% the integrated autocorrelation time and |geweke| is a simple test
% for a null hypothesis that the chain has converged.
chainstats(chain,results)
```

<u>*Geweke's Test*</u>: convergence when

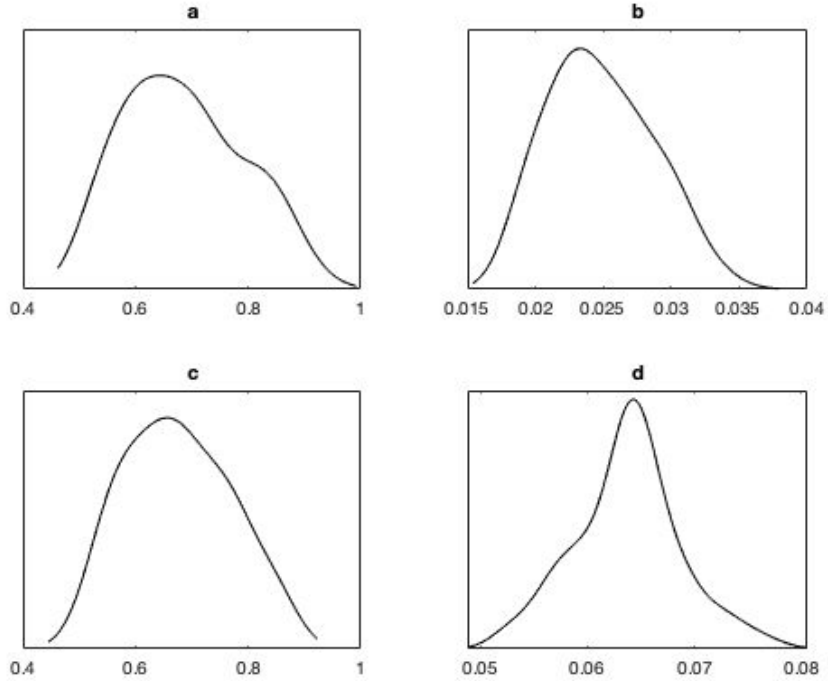
[mean of first 10% of chain]≅ [mean last 50% of chain]

Run #1



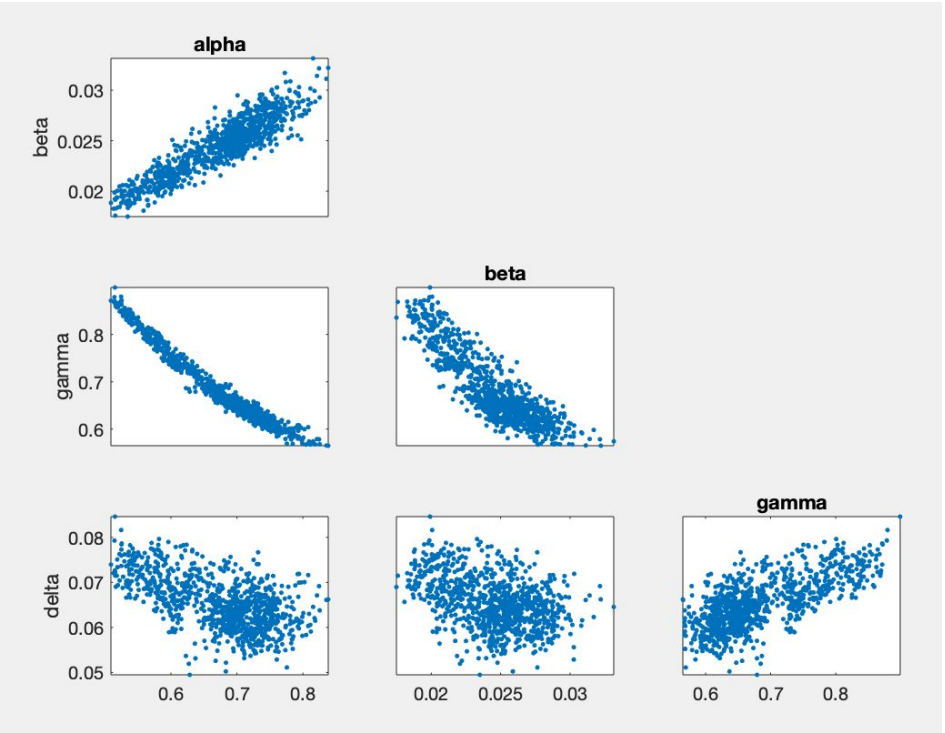
	mean	std	MC_err	tau	geweke
alpha	0.71731	0.018554	0.0037791	819.6	0.96892
beta	0.024935	0.0011731	0.00015603	220.44	0.98647
gamma	0.6882	0.072796	0.016175	575.98	0.85994
delta	0.065511	0.0053789	0.00083638	353.51	0.93472

Run #2

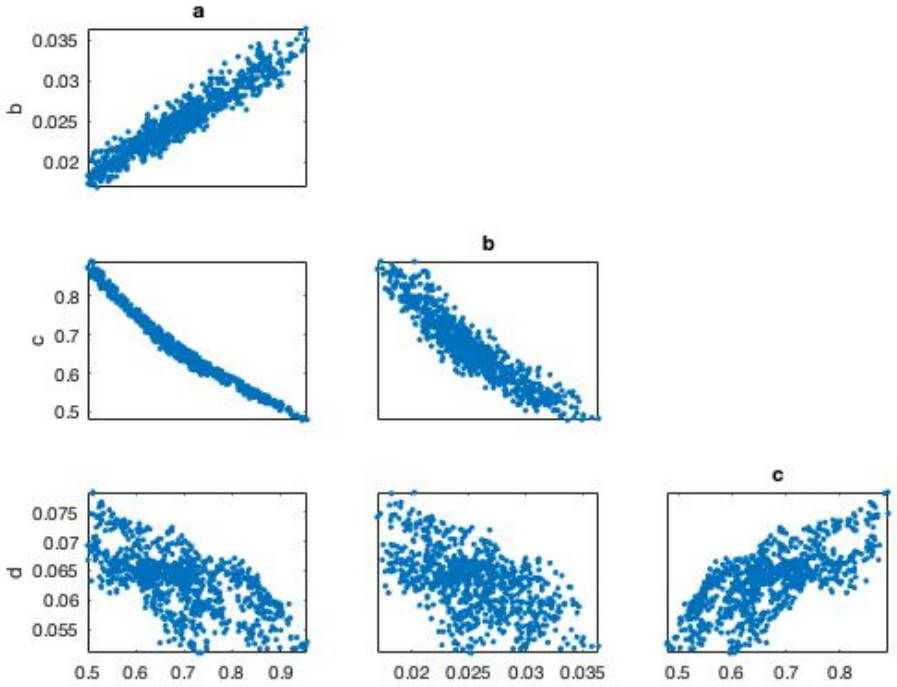


	mean	std	MC_err	tau	geweke
a	0.68574	0.10511	0.022061	512.31	0.59303
b	0.02479	0.0038077	0.00077037	497.52	0.61593
c	0.67581	0.093653	0.01967	544.95	0.63683
d	0.063763	0.0049431	0.00099315	520.72	0.93445

Run #1

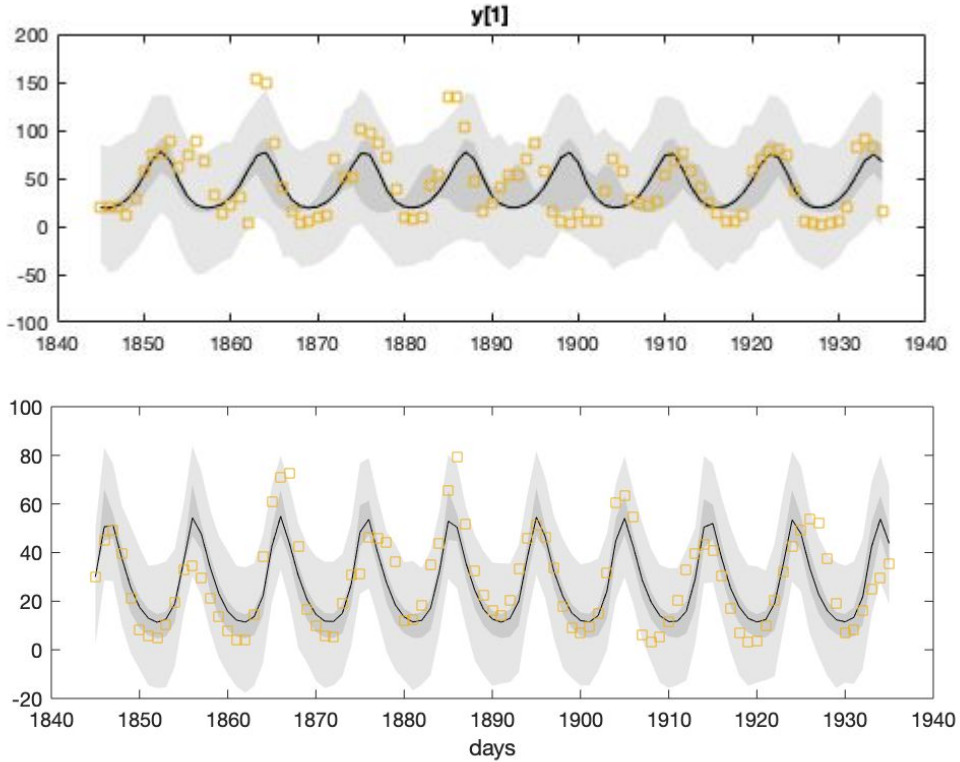

alpha
beta
0.03
0.025
0.02
beta
gamma
0.8
0.7
0.6
gamma
delta
0.08
0.07
0.06
0.05
0.6
0.7
0.8
0.02
0.025
0.03
0.6
0.7
0.8

Run #2


a
b
0.035
0.03
0.025
0.02
b
c
0.8
0.7
0.6
0.5
c
d
0.075
0.07
0.065
0.06
0.055
0.5
0.6
0.7
0.8
0.9
0.02
0.025
0.03
0.035
0.5
0.6
0.7
0.8

Moving Forward

- *Collis, 2017*
  - Create a better understanding of calibration and validation of models
  - Understand where these can be applied in our models
- DRAM
  - Finish debugging DRAM algorithm for Lotka-Volterra model
  - Apply DRAM to T1D model and data


y[1]
200
150
100
50
0
-50
-100
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
100
80
60
40
20
0
-20
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
days

Fri June 12

Work thus far...

Goal: Understand and implement different MCMC methods for parameterization of ODE models

1) Metropolis-Hastings
2) Adaptive Metropolis
3) Delayed Rejection Adaptive Metropolis

Linear model —> Lotka-Volterra —> Algae model —> T1D model

Bayes' Theorem

$$P(\theta|D,M) = \frac{P(D|\theta,M)P(\theta|M)}{P(D|M)}$$

$P(\theta|D,M)$ - posterior

$P(D|\theta,M)$ - likelihood

$P(\theta|M)$ - prior

$P(D|M)$ - normalization constant

Overview: this week

1) Implement DRAM method for T1D model
2) Particle Swarm Optimization

Delayed Rejection Adaptive Metropolis (DRAM)

- Adaptive Metropolis + Delayed Rejection = DRAM

Adaptive Metropolis (AM):

1. Sample parameter sets for period of length k_0 (often 100) using initial covariance matrix V_0
2. At *kth* step update V_0 with information from sampled posteriors, new covariance matrix V
3. Continue sampling

Delayed Rejection (DR):

1. At each sample accept or reject proposed parameter set
   a. If accept —> 2.
   b. If reject: propose alternative candidate and pass to acceptance criterion
      i. If accept —> 2.
      ii. If reject —> b.
2. Continue to sample following AM

DRAM options & assumptions

1) **NOD mice + no wave + simulated data**
2) NOD mice + wave + simulated data
3) **NOD mice + no wave + progressive mice data**
4) **NOD mice + no wave + acute mice data**
5) NOD mice + wave + progressive mice data
6) NOD mice + wave + acute mice data

ODE solver option

Mathews et al. data options

Setting up DRAM

1) Load data

```
%% Loading and plotting observed data

clear all
clear model data params options

readmatrix('acute+progressive_seperated_1.csv');
data=ans(:,[2 3 4 5 8]); % mean glucose data of progressive and acute
% NOTE: the Mathewsetal_data states that mice in the study were all NOD
% mice; see more info in T1Dss.m
```

2) Initialize likelihood

```
%% Calculating model sum of squares

model.ssfun = @(theta, data) T1Dss(theta, data);
```

3) Initialize parameters (53)

```
%% Defining intial parameters
% All parameters are constrained to be positive.
allParam_vals;

% {'param name', initial value, min_val, max_val}
params = {
    % From Table 1
    {'J', J, 0}
    {'k',  k, 0}
    {'b', b, 0}
    {'c', c, 0}
    {'e1', e1, 0}
    {'e2', e2, 0}
    {'scale_factor', scale_factor, 0}

    % Scale factors for clearance rates
    {'f1ns', f1ns, 0}
    {'f1s', f1s, 0}
    {'f2ns', f2ns, 0}
    {'f2s', f2s, 0}
```

4) Initialize prior function

```
%% Defining initial variance
% We assume having at least some prior information on the
% repeatability of the observation and assign rather non informational
% prior for the residual variances of the observed states. The default
% prior distribution is sigma2 ~ invchisq(S20,N0), the inverse chi
% squared distribution (see for example Gelman et al.). The predator and
% prey components have separate variances.

% Attempted uniform prior
model.S20 = [1];
model.N0  = [1];

% This is defaulted to 1 (normal need more points than parameters)
[mse initParam] = T1D_fitInitialParams(params, data);
model.sigma2 = mse;
```

5) Perform burn-in

```
%% Burn-in iterations
% First generate an initial chain.

options.nsimu = 1000;
options.method = 'dram';
[results, chain, s2chain]= mcmcrun(model,data,params,options);
```

6) Sample

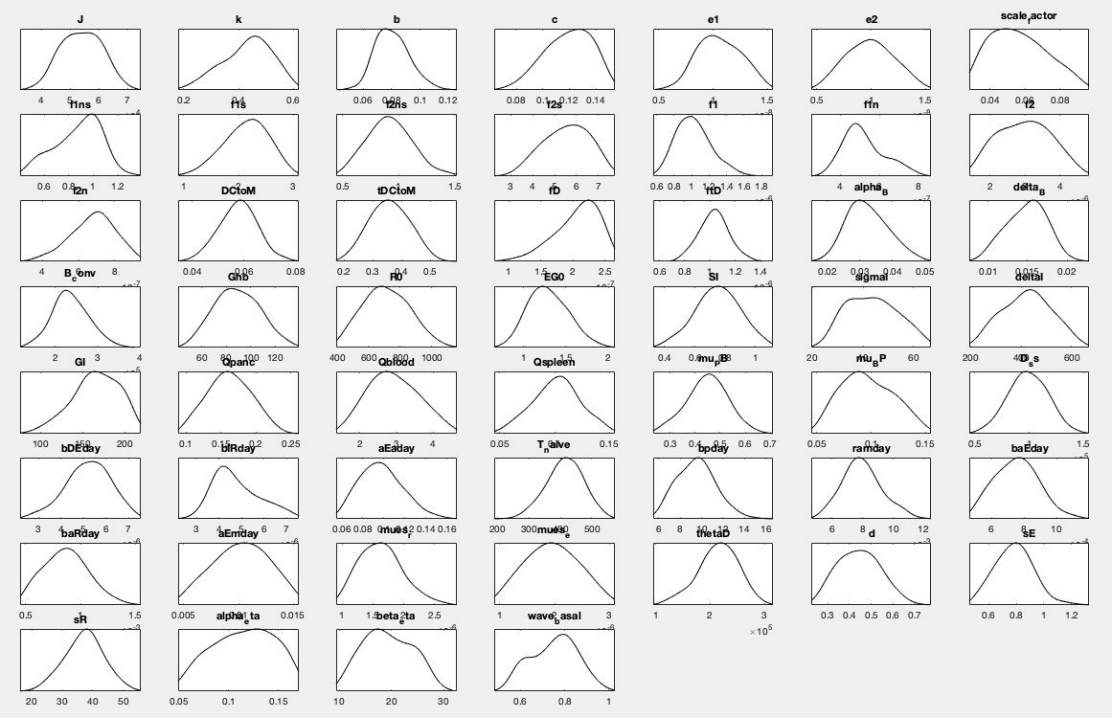
```
%% Running MCMC chain
% Then re-run starting from the results of the previous run,

options.nsimu = 5000;
options.method = 'dram';
[results, chain, s2chain] = mcmcrun(model,data,params,options, results);
```

Geweke's Test: convergence when

[mean of first 10% of chain] ≅ [mean last 50% of chain]

Results

NOD mice + no wave + simulated data



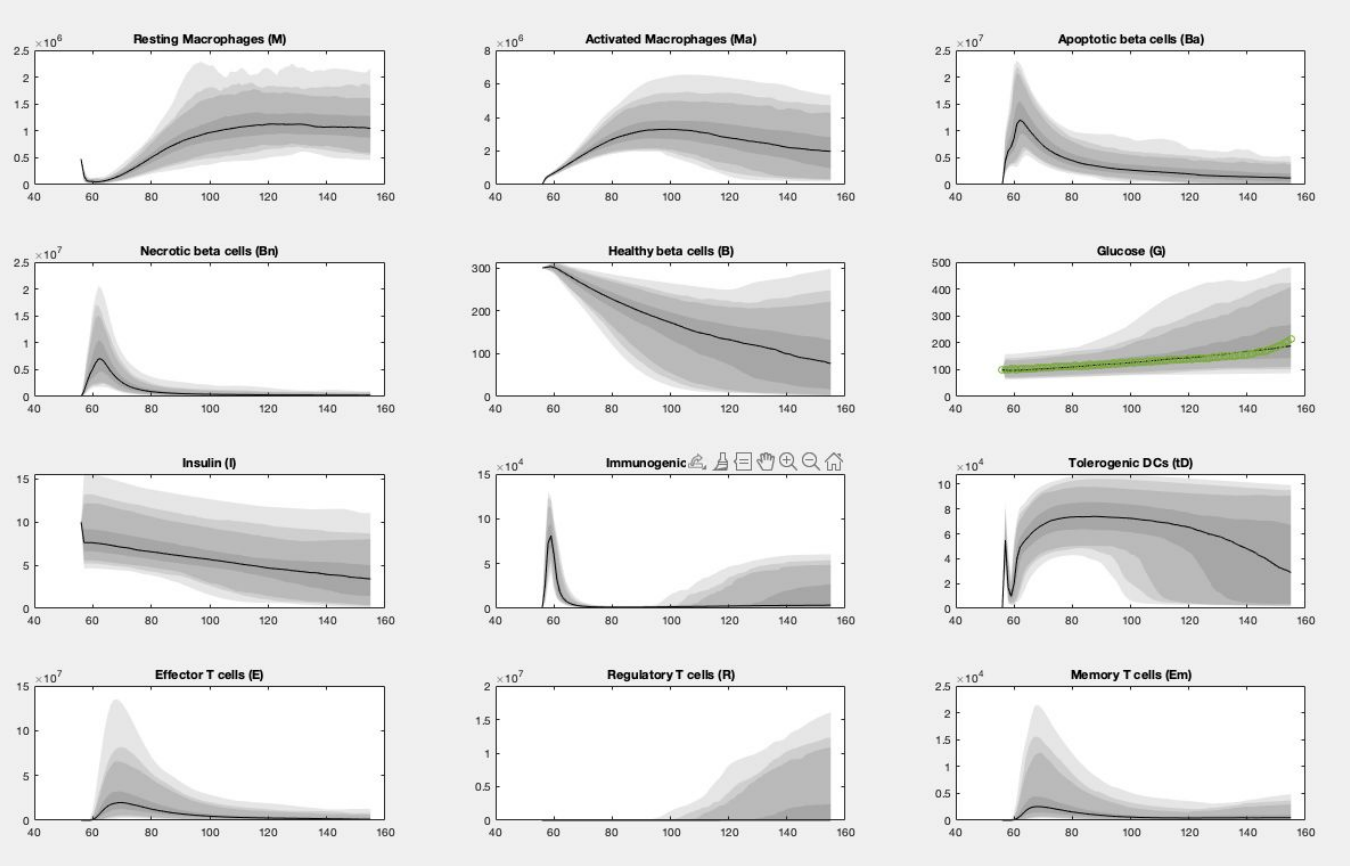
Parameter posterior density plots

	mean	std	MC_err	tau	geweke
J	53974	6875.2	984.93	159.19	0.95768
k	0.43252	0.087196	0.012956	360.98	0.75361
b	0.080428	0.011531	0.0017569	107.77	0.97863
c	0.11919	0.01726	0.0028018	157.93	0.76257
e1	1.0596e-08	1.9405e-09	2.5726e-10	89.372	0.86366
e2	9.92e-09	2.1401e-09	2.6791e-10	96.318	0.77692
scale_factor	0.056254	0.015691	0.0021661	334.56	0.61534
f1ns	0.87644	0.18558	0.032224	442.7	0.69247
f1s	2.1413	0.40046	0.066585	450.25	0.62309
f2ns	0.91731	0.1941	0.030707	165.74	0.92644
f2s	5.5436	1.022	0.13216	92.565	0.73459
f1	1.0251e-06	1.9666e-07	3.3966e-08	387.98	0.66286
f1n	5.206e-07	1.0308e-07	1.7121e-08	520.11	0.69036
f2	2.9834e-06	7.3557e-07	1.0798e-07	90.817	0.72121
f2n	6.7999e-07	1.1938e-07	1.963e-08	534.97	0.78478
DCtoM	0.057777	0.0070723	0.00090037	69.835	0.9955
tDCtoM	0.35712	0.067113	0.012279	247.77	0.57448
fD	2.0636e-07	3.3832e-08	6.196e-09	407.04	0.75611
ftD	1.0315e-06	1.1404e-07	1.6376e-08	99.503	0.99915
alpha_B	0.03234	0.0061066	0.00068464	88.27	0.84909
delta_B	0.014847	0.0024632	0.00040747	173.21	0.86385
B_conv	2.3621e+05	41695	5337.6	98.105	0.8635
Ghb	88.889	17.878	3.158	332.32	0.69416
R0	713.62	142.58	19.786	108.77	0.83778
EG0	1.2912	0.22916	0.03281	109.23	0.80274
SI	0.75147	0.14375	0.023272	150.87	0.89388
sigmaI	43.714	9.7977	1.5876	322.32	0.54848
deltaI	424.15	95.237	15.273	145.49	0.91643
GI	165.63	26.9	3.3559	99.245	0.89439
Qpanc	0.16222	0.030309	0.0042573	110.85	0.98922
Qblood	2.9195	0.67457	0.088611	83.52	0.99969
Qspleen	0.1018	0.020965	0.00336	148.9	0.99398
mu_PB	0.4484	0.079934	0.012118	169.31	0.97602
mu_BP	0.096583	0.023291	0.0034901	194.06	0.63319
D_ss	98944	17603	2749.3	174.8	0.79815
bDEday	5.2096e-06	9.6646e-07	1.6204e-07	147.37	0.93557
bIRday	4.7779e-06	1.0032e-06	1.8274e-07	375.42	0.64459
aEaday	0.097293	0.01987	0.003207	121.83	0.6491
T_naive	414.01	56.898	8.5803	128.18	0.92217
bpday	9.5068	1.7484	0.28257	161.23	0.74832
ramday	0.0079633	0.0012633	0.00018264	144.66	0.93441
baEday	0.00075473	0.00013031	1.7167e-05	92.687	0.85946
baRday	0.0008679	0.00019909	3.0296e-05	142.59	0.61097
aEmday	0.010256	0.0024783	0.00035619	223.2	0.70533
mues_r	1.6314e-06	3.4026e-07	5.6353e-08	360.69	0.879
mues_e	1.9553e-06	4.7516e-07	7.1117e-08	110.95	0.73987
thetaD	2.1479e+05	35982	6470	399.95	0.74653
d	0.44464	0.095279	0.014334	117.23	0.63978
sE	0.78454	0.13593	0.019514	123.04	0.94578
sR	37.24	6.2038	0.96475	154.62	0.99113
alpha_eta	0.11535	0.029932	0.0045729	178.61	0.61593
beta_eta	19.301	4.5358	0.72709	142.28	0.61677
wave_basal	0.75219	0.10035	0.019658	537.96	0.61577

Parameter means and convergence statistics

Results

NOD mice + no wave + simulated data

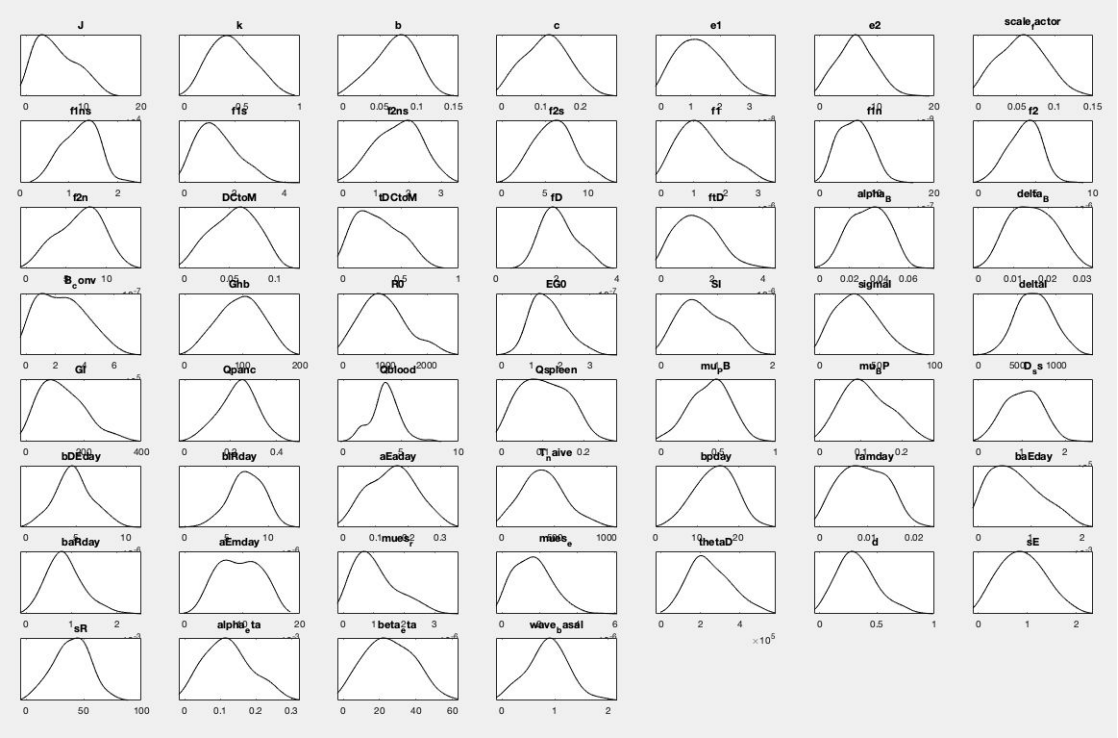


Model prediction based on chain samples (per model variable)

Glucose overlaid with original data in green

Results

NOD mice + no wave + progressive mice data



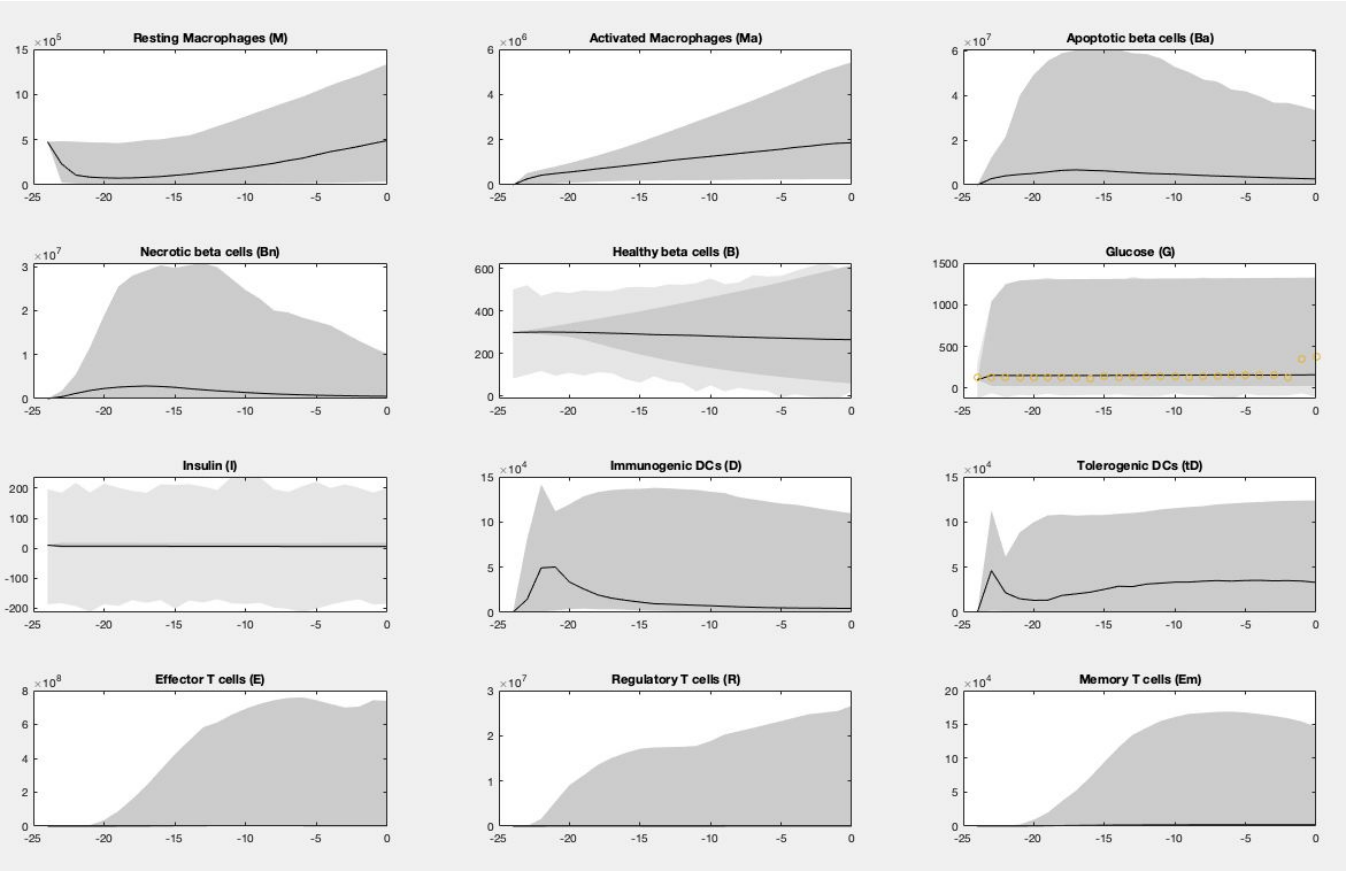
Parameter posterior density plots

	mean	std	MC_err	tau	geweke
J	52731	34336	6125.5	182.39	0.332
k	0.40874	0.18076	0.027968	294.08	0.26972
b	0.070446	0.027799	0.0043551	255.05	0.53246
c	0.11201	0.054691	0.0091856	344.91	0.19287
e1	1.2831e-08	7.3467e-09	1.0519e-09	109.29	0.78886
e2	6.3152e-09	3.1517e-09	4.3893e-10	88.341	0.86902
scale_factor	0.057081	0.029646	0.004222	125.52	0.62521
f1ns	1.225	0.35411	0.055009	180.29	0.61469
f1s	1.2968	0.78689	0.1162	159.83	0.37946
f2ns	1.675	0.6814	0.10847	161.87	0.65421
f2s	6.0211	2.2777	0.34126	340.25	0.41985
f1	1.2355e-06	7.0479e-07	1.2199e-07	282.12	0.62867
f1n	6.0195e-07	2.6763e-07	4.3177e-08	199.48	0.38997
f2	3.9961e-06	1.3704e-06	2.442e-07	234.75	0.86789
f2n	6.7265e-07	2.6956e-07	4.1222e-08	185.56	0.95945
DCtoM	0.054999	0.024288	0.0038234	286	0.27905
tDCtoM	0.29817	0.18552	0.031707	310.26	0.26781
fD	2.059e-07	5.5182e-08	7.8045e-09	112.97	0.98781
ftD	1.3856e-06	8.0911e-07	1.1706e-07	257.04	0.73946
alpha_B	0.032911	0.012532	0.0019635	171.29	0.85406
delta_B	0.015435	0.0061733	0.0010261	491.44	0.38042
B_conv	2.4384e+05	1.5804e+05	23859	164.08	0.18479
Ghb	96.33	36.471	5.345	135.98	0.4842
R0	966.11	535.07	79.688	90.491	0.59679
EG0	1.5735	0.57574	0.08903	202.19	0.9088
SI	0.73092	0.39938	0.063986	231.36	0.50757
sigmaI	32.128	17.854	3.0035	284.6	0.58337
deltaI	726.02	226.63	27.879	66.193	0.98995
GI	125.81	75.294	13.94	270.22	0.35903
Qpanc	0.23263	0.079577	0.013537	158.91	0.96419
Qblood	3.6454	1.2257	0.21439	226.36	0.85346
Qspleen	0.10319	0.056373	0.0097974	295.54	0.35013
mu_PB	0.4244	0.17031	0.027435	123.11	0.99036
mu_BP	0.11163	0.055827	0.0088054	88.694	0.96168
D_ss	1.0284e+05	41427	5694.2	110.6	0.50437
bDEday	4.7748e-06	1.8379e-06	3.01e-07	179.23	0.89108
bIRday	7.3663e-06	1.9897e-06	3.1805e-07	242.85	0.74217
aEaday	0.15206	0.070596	0.01199	197.01	0.66127
T_naive	395.08	201.2	27.805	102.24	0.77193
bpday	14.323	4.7355	0.64099	143.62	0.9825
ramday	0.0091878	0.0047796	0.00073176	110.8	0.34391
baEday	0.00069695	0.00047315	7.5444e-05	169.22	0.051705
baRday	0.00085972	0.00039503	5.9602e-05	144.91	0.76888
aEmday	0.009481	0.0035647	0.00063672	342.76	0.46363
mues_r	1.041e-06	6.9941e-07	1.075e-07	303.6	0.12172
mues_e	1.6479e-06	9.7908e-07	1.1744e-07	76.041	0.83555
thetaD	2.4772e+05	1.0397e+05	17898	306.58	0.34569
d	0.3245	0.16859	0.024125	101.44	0.92405
sE	0.8945	0.4502	0.044893	57.806	0.85598
sR	39.589	15.199	2.5185	112.61	0.93348
alpha_eta	0.12195	0.064021	0.011557	161.85	0.65082
beta_eta	25.321	12.902	1.5814	80.108	0.46734
wave_basal	0.86555	0.37247	0.047824	123.58	0.91442

Parameter means and convergence statistics

Results

NOD mice + no wave + progressive mice data

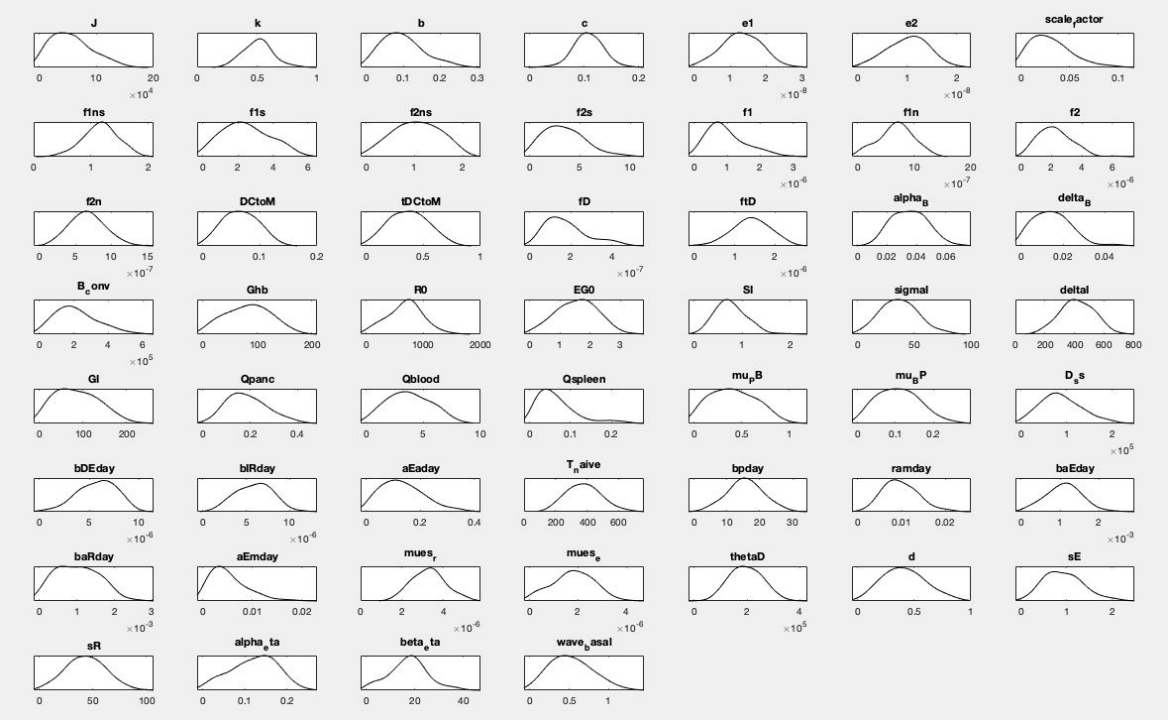


Model prediction based on chain samples (per model variable)

Glucose overlaid with original data in yellow

Results

NOD mice + no wave + acute mice data



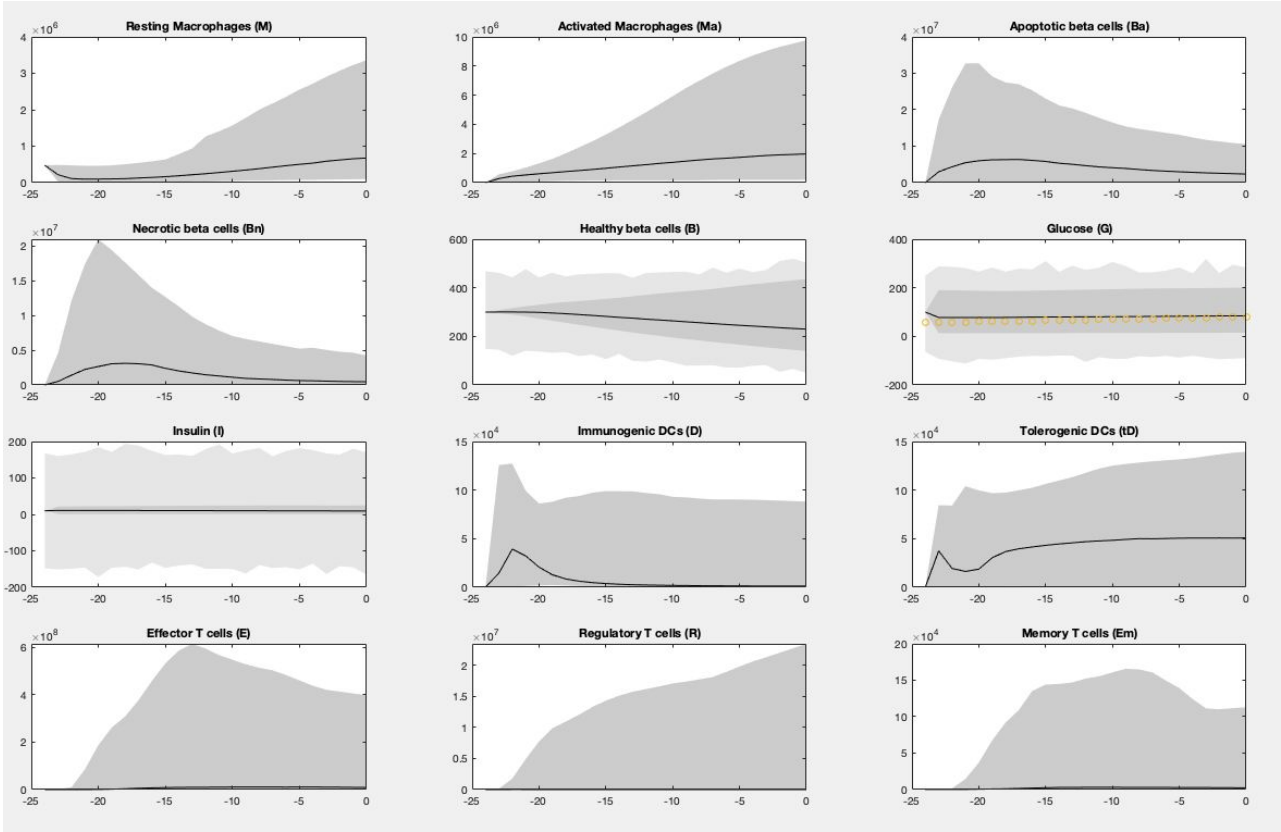
Parameter posterior density plots

	mean	std	MC_err	tau	geweke
J	55675	35226	5335.2	453.81	0.9033
k	0.50644	0.12134	0.01976	328.26	0.9738
b	0.097982	0.055743	0.0098632	445.52	0.56074
c	0.10732	0.026769	0.0040792	153.07	0.79559
e1	1.3147e-08	5.3946e-09	8.0882e-10	84.701	0.78559
e2	1.0219e-08	4.0779e-09	6.3712e-10	87.874	0.77816
scale_factor	0.0306	0.020928	0.0033104	110.31	0.5636
f1ns	1.1655	0.29608	0.040394	122.54	0.9285
f1s	2.4714	1.3268	0.1969	96.85	0.60986
f2ns	1.0437	0.47899	0.081145	417.35	0.41469
f2s	3.4297	2.0549	0.29261	126.9	0.57702
f1	9.7658e-07	5.9588e-07	1.0868e-07	423.75	0.45542
f1n	6.7622e-07	3.0201e-07	5.3421e-08	218.42	0.81481
f2	2.2208e-06	1.0872e-06	1.6758e-07	98.126	0.55397
f2n	6.6722e-07	2.3973e-07	4.071e-08	362.73	0.86388
DCtoM	0.066144	0.030706	0.004236	86.301	0.88089
tDCtoM	0.36984	0.16856	0.027361	270.55	0.50501
fD	1.7672e-07	1.0299e-07	1.9074e-08	451.25	0.4581
ftD	1.411e-06	4.6008e-07	6.7109e-08	184.23	0.91297
alpha_B	0.035299	0.012138	0.0020495	404.47	0.68702
delta_B	0.014783	0.0090727	0.0013476	74.261	0.45092
B_conv	2.1375e+05	1.2098e+05	16560	191.29	0.48429
Ghb	82.806	40.459	5.8416	115.79	0.99104
R0	680.52	307.91	54.699	155.43	0.92835
EG0	1.5582	0.66682	0.096668	71.257	0.94748
SI	0.7646	0.3288	0.057134	455.54	0.98127
sigmaI	36.927	16.316	2.566	139.66	0.61321
deltaI	418.53	112.94	15.483	124.62	0.68345
GI	86.182	49.912	6.1966	55.541	0.7271
Qpanc	0.18721	0.084571	0.012577	210.23	0.69688
Qblood	3.7388	1.8774	0.32221	231.11	0.95824
Qspleen	0.0655	0.050285	0.0088138	255.2	0.55688
mu_PB	0.41144	0.23468	0.03143	231.96	0.92861
mu_BP	0.1073	0.055382	0.0077496	85.513	0.94881
D_ss	87414	43988	6102.3	96.724	0.58781
bDEday	5.7796e-06	1.9154e-06	2.8646e-07	233.47	0.65609
bIRday	5.9014e-06	2.079e-06	2.969e-07	195.27	0.59239
aEaday	0.1336	0.078958	0.011562	103.81	0.65826
T_naive	378.29	108.1	14.598	108.35	0.95252
bpday	15.597	5.3874	0.74261	91.846	0.8428
ramday	0.00977	0.0039451	0.00062858	225.32	0.82623
baEday	0.0010958	0.00045356	5.2118e-05	57.475	0.93471
baRday	0.0010113	0.00056648	9.0987e-05	188.33	0.41718
aEmday	0.0051655	0.003754	0.00052506	166.66	0.39106
mues_r	3.2748e-06	7.93e-07	1.2077e-07	147.74	0.91148
mues_e	1.7968e-06	8.419e-07	1.2964e-07	215.51	0.81438
thetaD	1.9652e+05	61154	8561.5	72.304	0.79099
d	0.41173	0.18592	0.03013	330.65	0.68347
sE	0.92956	0.42477	0.065108	153.26	0.79977
sR	43.427	18.175	2.4513	103.94	0.76246
alpha_eta	0.12116	0.053702	0.0082877	159.97	0.62273
beta_eta	17.379	7.8703	1.1315	235.71	0.9198
wave_basal	0.52359	0.26358	0.042652	189.18	0.44553

Parameter means and convergence statistics

Results

NOD mice + no wave + acute mice data



Model prediction based on chain samples (per model variable)

Glucose overlaid with original data in yellow

Particle Swarm Optimization

- Global optimization technique
- General idea: based on birds, group searches better than an individual; each swarm member listens to the best result, changes its course

<u>Algorithm:</u> Global

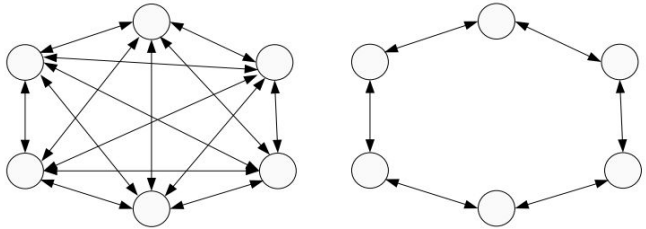
1. Initialize *n* swarm members distributed uniformly across search space; Select coefficients c_1 and c_2 to describe acceleration due to communication.
2. Evaluate objective function for each swarm member, 'remember' best of all, G_{best}, each best, P_{best}.
3. Calculate velocities based on bests:

$$v_{ij}^{t+1} = v_{ij}^{t} + c_1 r_{1j}^{t}\left[P_{best,i}^{t} - x_{ij}^{t}\right] + c_2 r_{2j}^{t}\left[G_{best} - x_{ij}^{t}\right]$$

4. Update swarm locations:

$$x_i^{t+1} = x_i^{t} + v_i^{t+1}$$

5. Repeat for set iterations, or until difference in location at time *t+1* is negligible



(a) Star or gbest. (b) Ring or lbest.

PSO Options

For our use: treat 53 parameters as search space, minimize difference between data/model

- User Choices:
  - NOD vs. wild type
  - Progressive vs. acute
  - Wave on/off
  - Constant vs. variable ICs → 65 parameters
- Assumptions regarding initial parameter range:
  - ±50% for parameters, ±15% for ICs
- PSO Preferences:
  - Gbest model
  - Swarm size of 40
  - Max iterations of 400
  - Automatically selected c_1, c_2

PSO Set-up

```
% Set lower, upper bounds for pars
paramsInit_arr = formatArray(paramsInit_struct); % Convert param struct to array
lb = paramsInit_arr - .80 * paramsInit_arr; % 50% below
ub = paramsInit_arr + .80 * paramsInit_arr; % 50% above

% Adjust bounds for ICs
if wICs == 1
    lb(54:end) = paramsInit_arr(54:end) - .40 * paramsInit_arr(54:end); % 15% below
    ub(54:end) = paramsInit_arr(54:end) + .40 * paramsInit_arr(54:end); % 15% above
end

% If no wave, remove from optimization
if waveon == 0
    paramsInit_arr(53) = 0;
    lb(53) = 0;
    ub(53) = 0;
end
```

1) Set ul/lb for params, account for wave

2) Declare obj func, options

```
% Objective func -> Minimize sum of squares for y6 (glucose)
objfn = @(params) T1Dss(params, y, waveon, type, wICs);

options = optimoptions('particleswarm', 'SwarmSize', 40, 'MaxIterations', 400, ...
    'UseParallel', true, 'Display', 'iter', 'DisplayInterval', 20, 'PlotFcn', @pswplotbestf);
```

3) Run PSO

```
% Run PSO
[out, fval, exitflag, output] = particleswarm(objfn, npars, lb, ub, options);
```

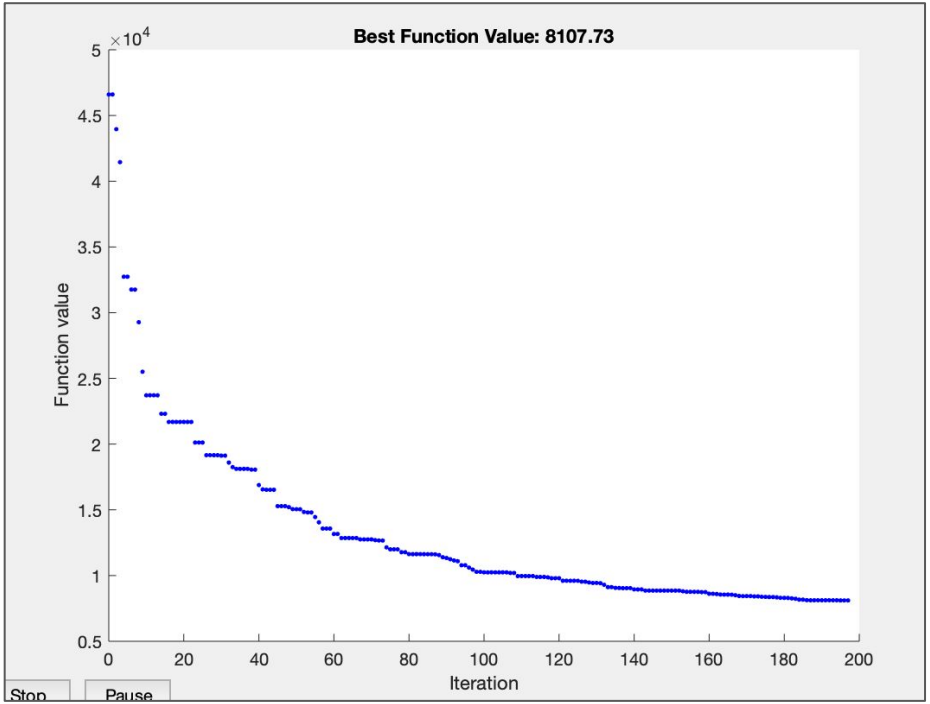
Objective function:

```
function ss = T1Dss(params, obsPts, waveon, type, wICs)
% Solve the ODE, find residual for glucose
[~, modPred] = T1Dfun(params, waveon, type, wICs);
res = (modPred(:,6) - obsPts).*(modPred(:,6) - obsPts);

% Square the error values and sum them up
ss = sum(res);
end
```

- Minimize sum of squares difference b/w model and glucose data

PSO Output

NOD mice + progressive mice data + no wave + variable ICs

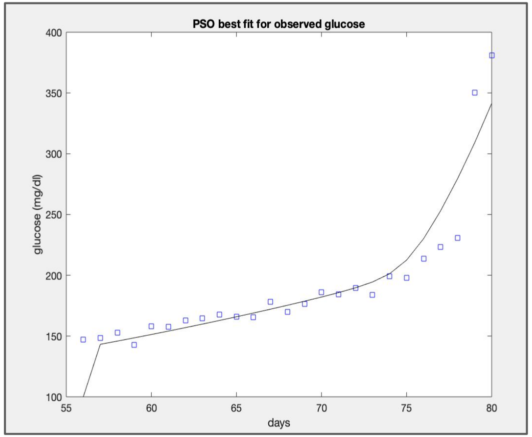


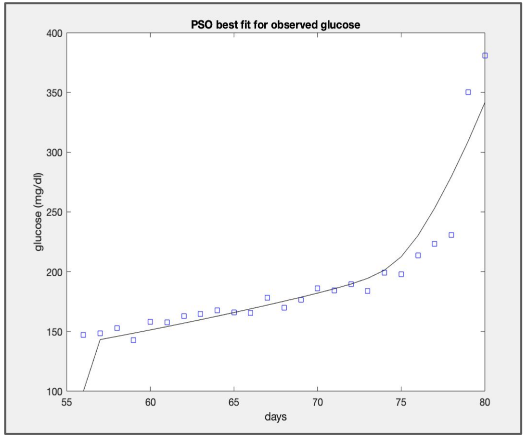
Iteration	f-count	Best f(x)	Mean f(x)	Stall Iterations
0	40	4.66e+04	2.836e+05	0
20	840	2.169e+04	4.176e+05	4
40	1640	1.689e+04	8.208e+04	0
60	2440	1.316e+04	5.26e+04	0
80	3240	1.163e+04	2.697e+04	0
100	4040	1.024e+04	1.63e+05	0
120	4840	9797	2.234e+04	1
140	5640	8944	1.279e+04	0
160	6440	8615	1.178e+04	0
180	7240	8303	8439	0
200	8040	8059	8963	0
220	8840	7750	2.641e+04	1
240	9640	7713	1.019e+04	0
260	10440	7673	8071	2
280	11240	7606	7799	2
300	12040	7583	7609	0
320	12840	7553	1.476e+04	4
340	13640	7545	7588	0
360	14440	7544	7551	1
380	15240	7541	7549	7
400	16040	7540	7548	9

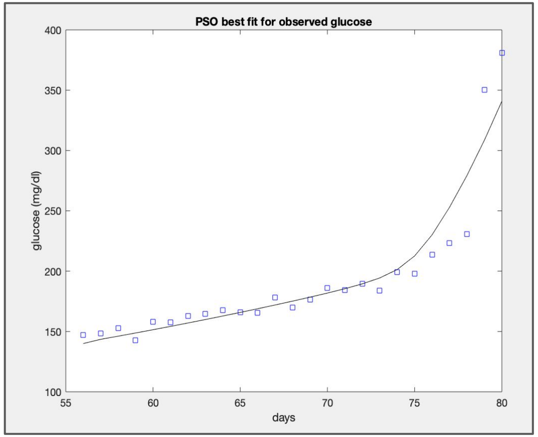
Stall iteration: iteration where value of G_{best} does not change

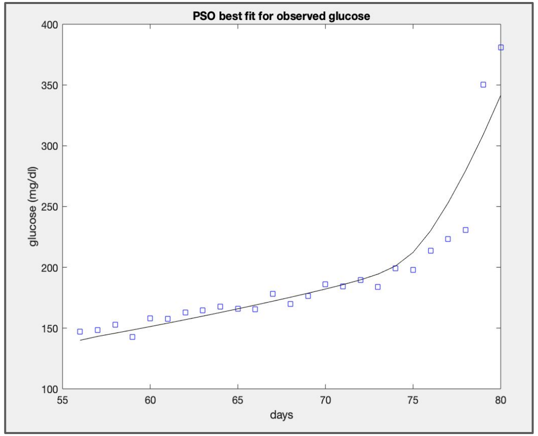
Results

NOD mice + progressive mice data+ **constant ICs**



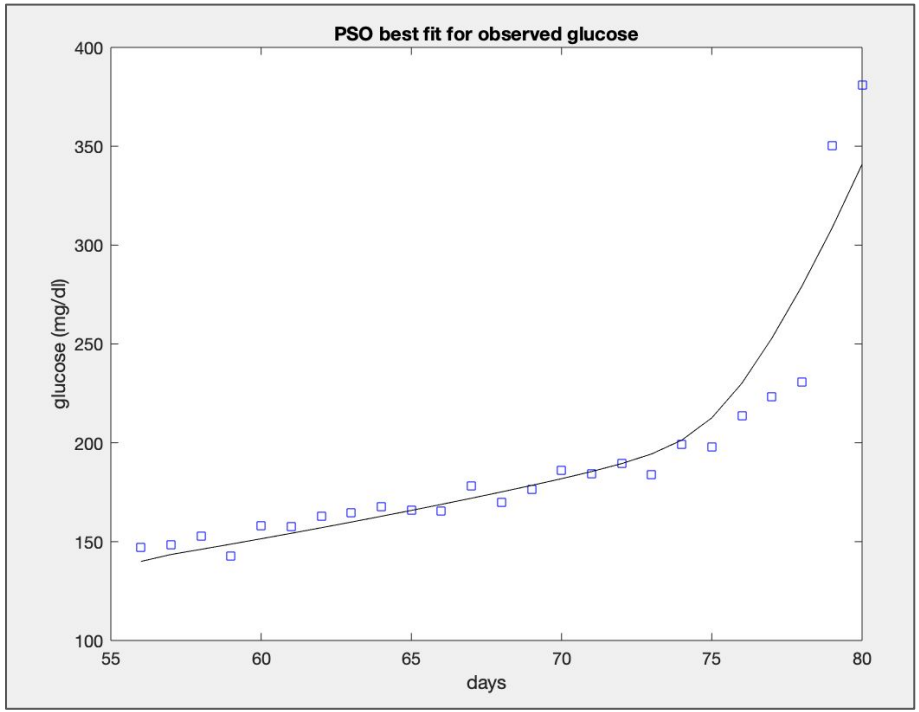




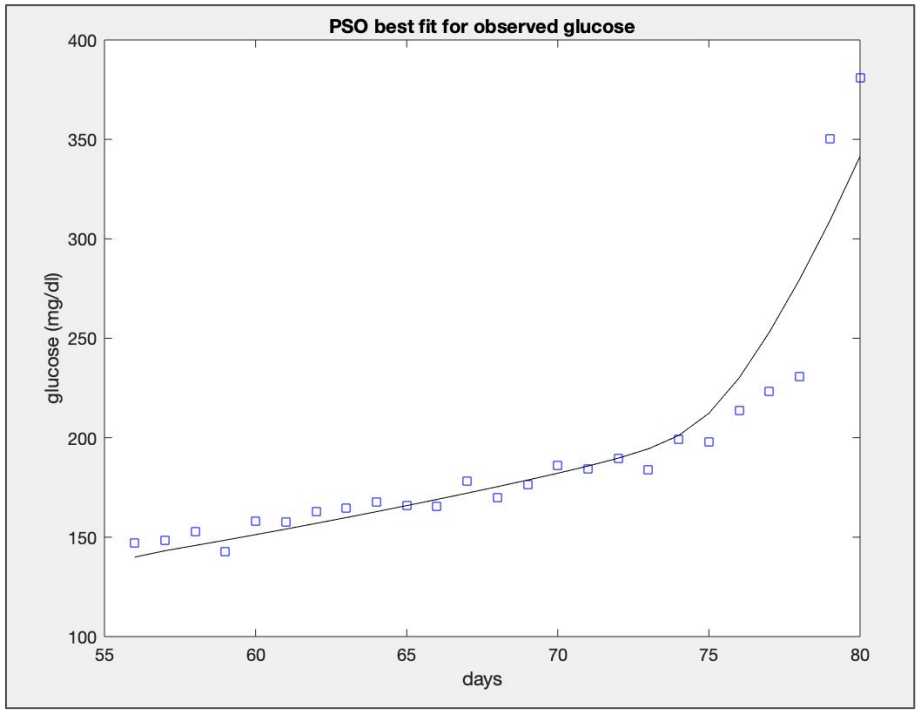


Results

NOD mice + progressive mice data+ **variable ICs**



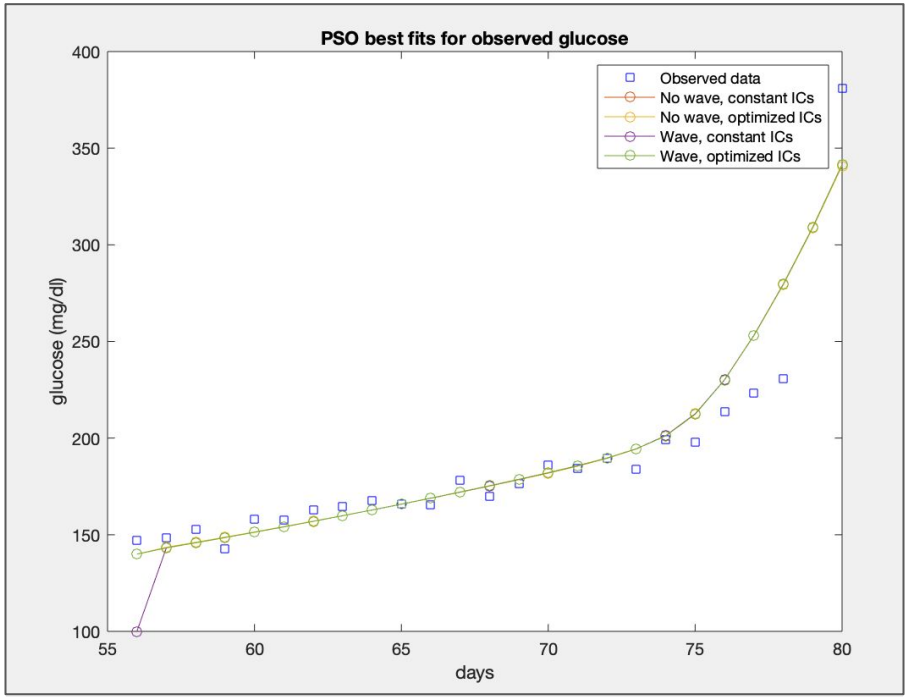
without wave

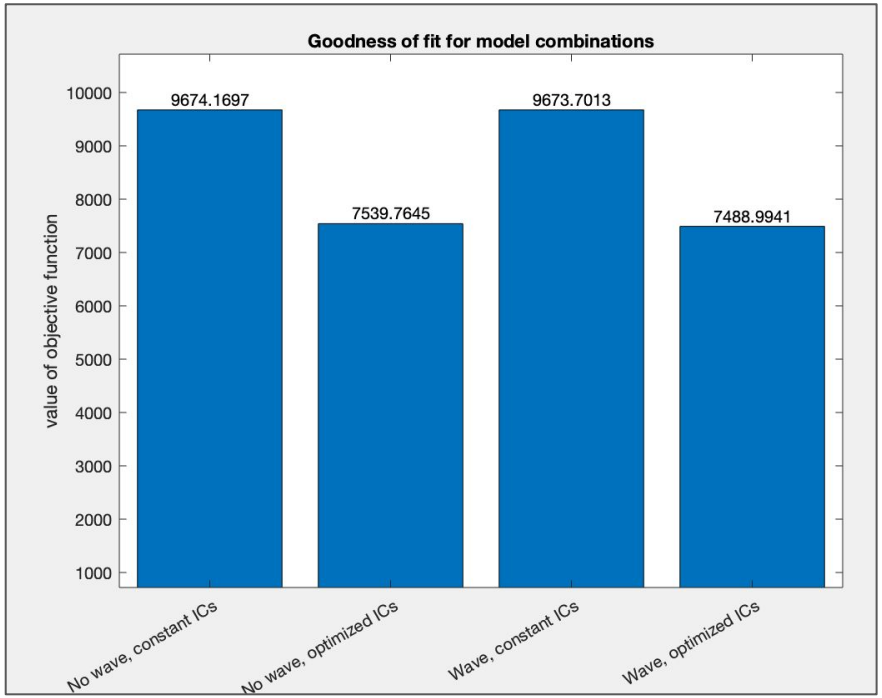


with wave

Comparison of Results

NOD mice + progressive mice data





- Optimizing ICs improves fit, wave doesn't impact fit (as *wave_basal* is being parameterized)

Moving Forward

1) Implement DRAM and PSO on original raw data
    a) Filling in missing data
    b) Unifying the time period
    c) Read Mathews et al. paper
2) Start our write-up

Fri June 19

Overview: this week

1) How should we use the data that we are given?
    a) Mathews vs. Li
    b) Issues with sparsity
    c) Adjusting time scale
    d) Using both data sets together?
2) Changes to MCMC to match data handling

Finishing up last week:

- Incorporated eFAST results from An
    - Changes to bounds
    - Optimize subsets of params:
        - Sensitive
        - Glucose-involved
        - Both

Datasets

Mathews et al.

- Multiple cohorts of NOD mice, total *n*=660
- 489 diabetic:
  - 354 ‘Progressive’
  - 135 ‘Acute’

Diabetes Onset: Blood glucose ≥250 $mgdl^{-1}$ for two consecutive measurements

Progressive: Blood glucose ≥200 $mgdl^{-1}$ prior to diabetes diagnosis

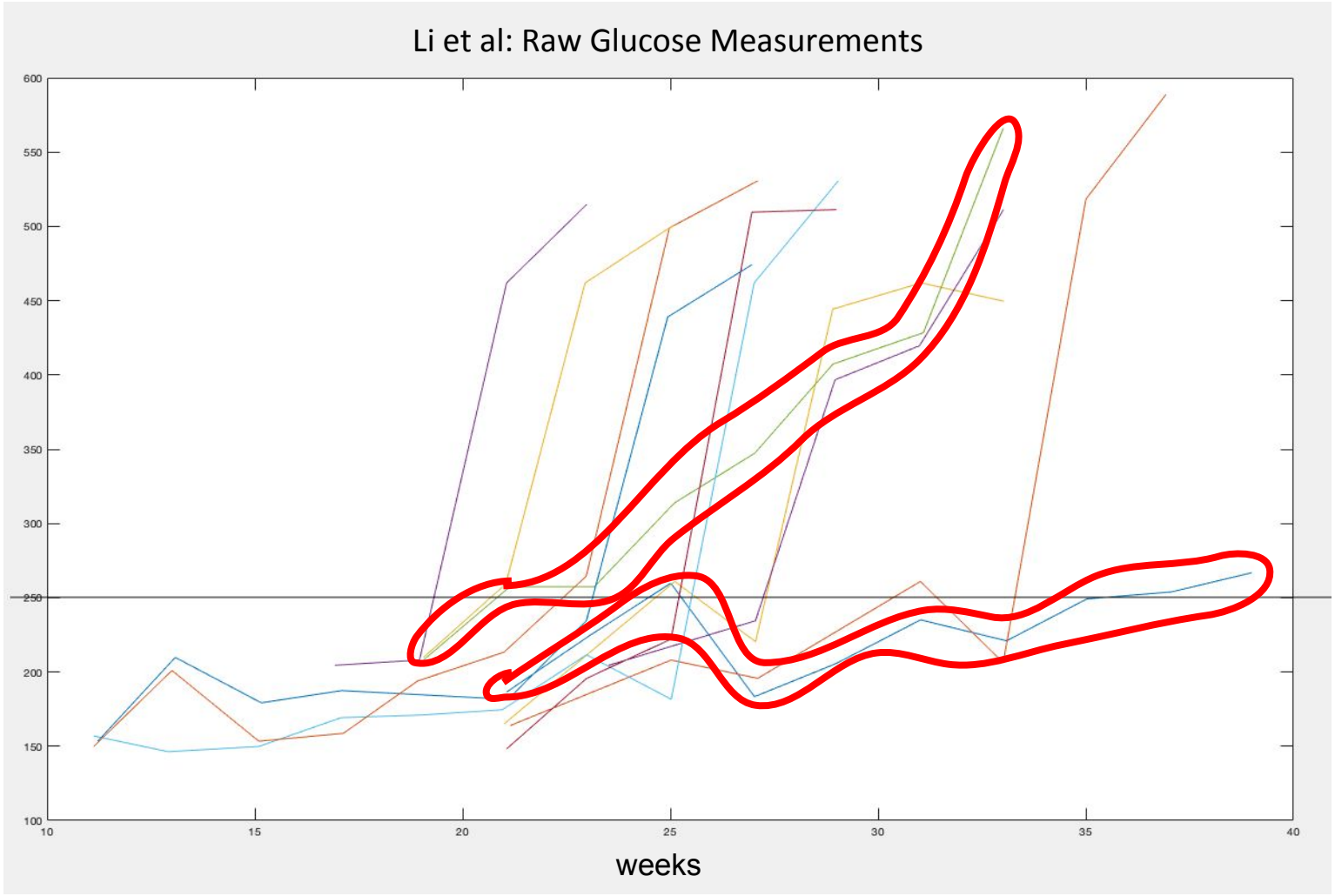
Acute: Blood glucose ‘spiked’ suddenly; did not exceed 200 $mgdl^{-1}$ prior to diagnosis

Li et al.

- Single cohort of NOD mice, total *n* unknown
- 11 diabetic:
  - 9 ‘Progressive’
  - 2 ‘Acute’

Diabetes Onset: Used Mathews definition

Progressive/Acute: *Shape upon visual inspection*


Li et al: Raw Glucose Measurements
600
550
500
450
400
350
300
250
200
150
100
10
15
20
25
30
35
40
weeks

250 $mgdl^{-1}$

Datasets (cont.)

Mathews et al.

- Measurements taken every other day (ideally)
  - When first reading ≥250 $mgdl^{-1}$, measurements moved to every 24 hours
- Measured relatively; days until diagnosis (-25 to 0)

Strengths: large cohort, many data points prior to onset

Weaknesses: large amounts of missing data, difficult to reconstruct absolute timeframe, irregularity in measurements based on values, no data post-onset

Li et al.

- Measurements taken irregularly (generally weekly)
  - Non-discrete
- Measured absolutely from birth

Strengths: absolute time, data for post-onset, no blatant missing values

Weaknesses: small cohort, sparse measurements

Data Alterations: Overview

Goals:

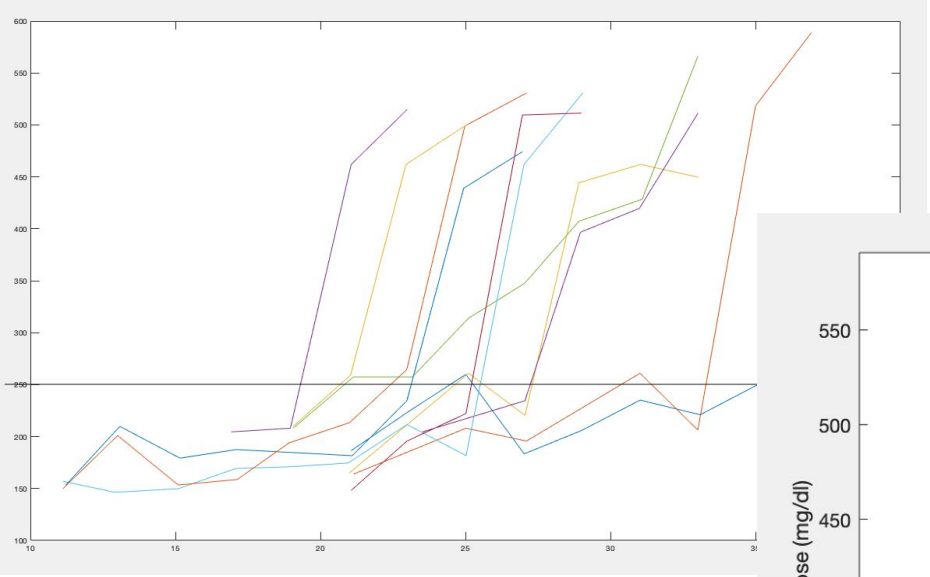
- Work in absolute time
- Take advantage of Mathews's size, Li's post-onset data

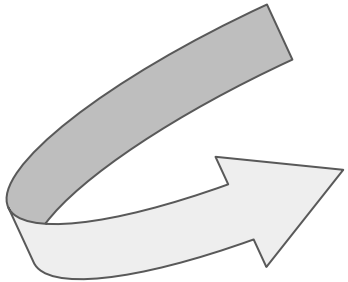
Problems:

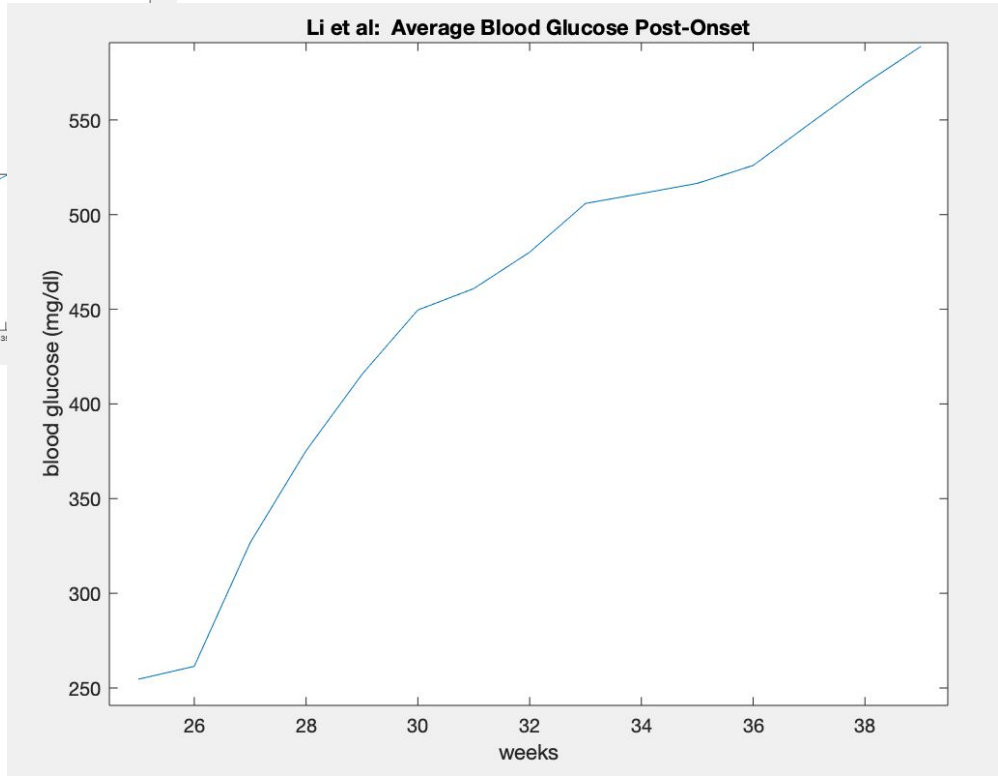
- Averaging unaligned data causes information loss
- If we align data, where? (time)
- Interpolating data at discrete times

Solution:

- Align Li et al. data at point of onset, average aligned data
- 'Place' averaged data at time determined by Mathews data
    - Treat this time as parameter in MCMC


600
550
500
450
400
350
300
250
200
150
100
10
15
20
25
30




Li et al: Average Blood Glucose Post-Onset
blood glucose (mg/dl)
550
500
450
400
350
300
250
26
28
30
32
34
36
38
weeks

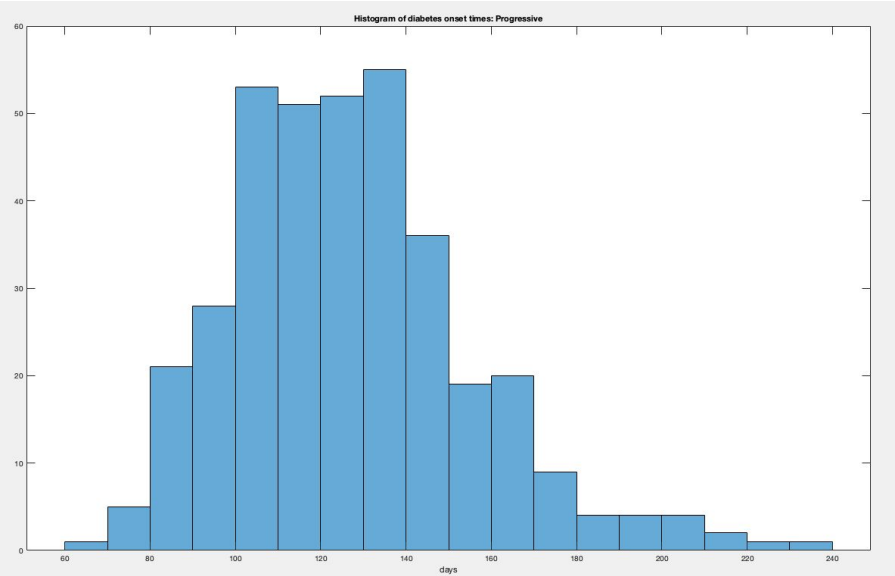
Determining diabetes onset time distributions

Mathews et al & Li et al

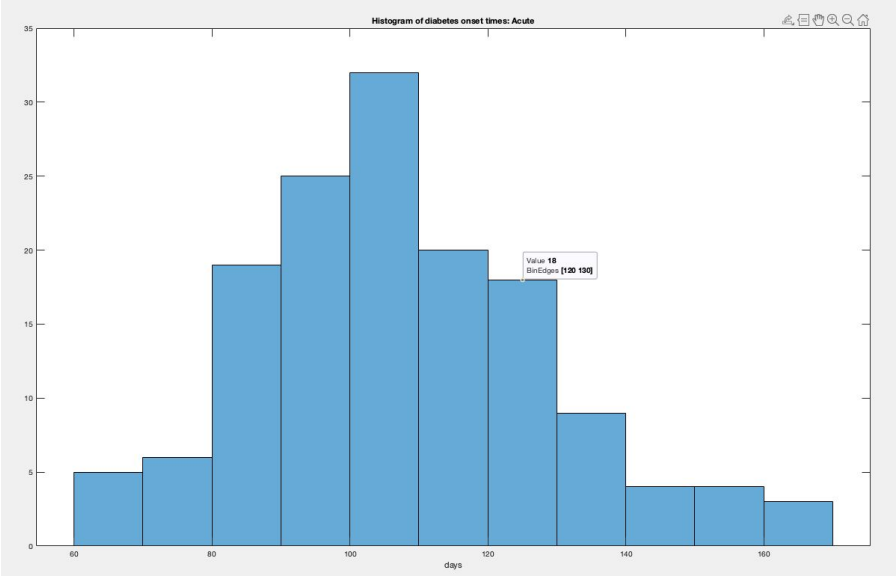
1. Start with shifted raw data (Prof. Shtylla)
    a. acute OR progressive mice
2. For each mouse:
    a. Determine 2 consecutive days ≥ 240 mg/dL glucose
    b. Onset threshold = 2nd day of ≥ 240 mg/dL glucose
    c. Collect thresholds
3. Fit *lognormal* distribution to thresholds
    a. obtain lognormal mean and variance

Histograms: *Mathews et al*

Progressive

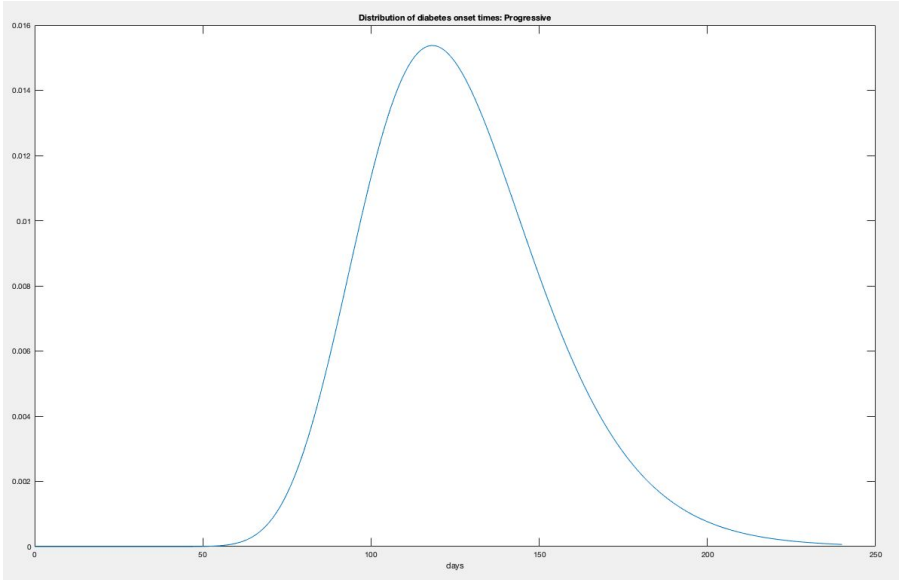


Acute



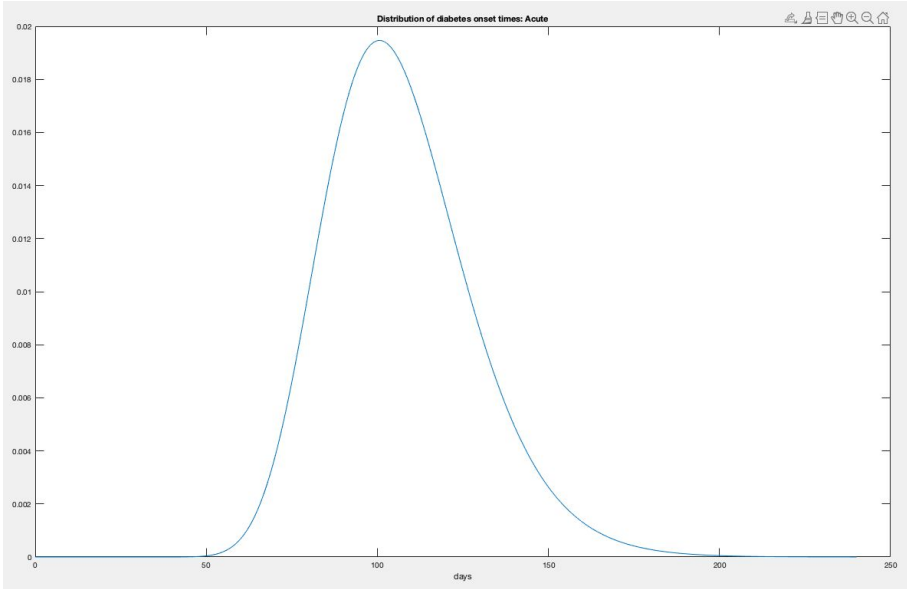
Distributions: *Mathews et al*

Progressive



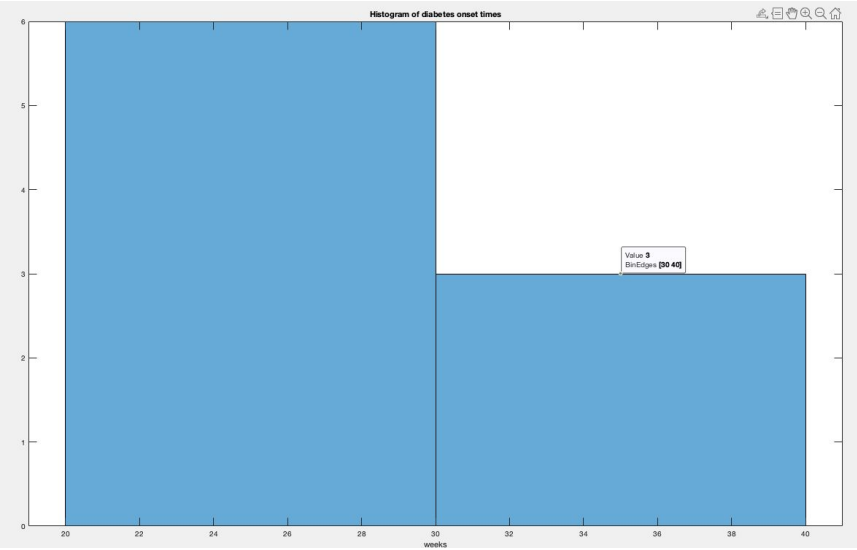

```
Lognormal distribution
     mu = 4.80969    [4.78775, 4.83162]
  sigma =  0.2134    [0.198979, 0.230092]
```

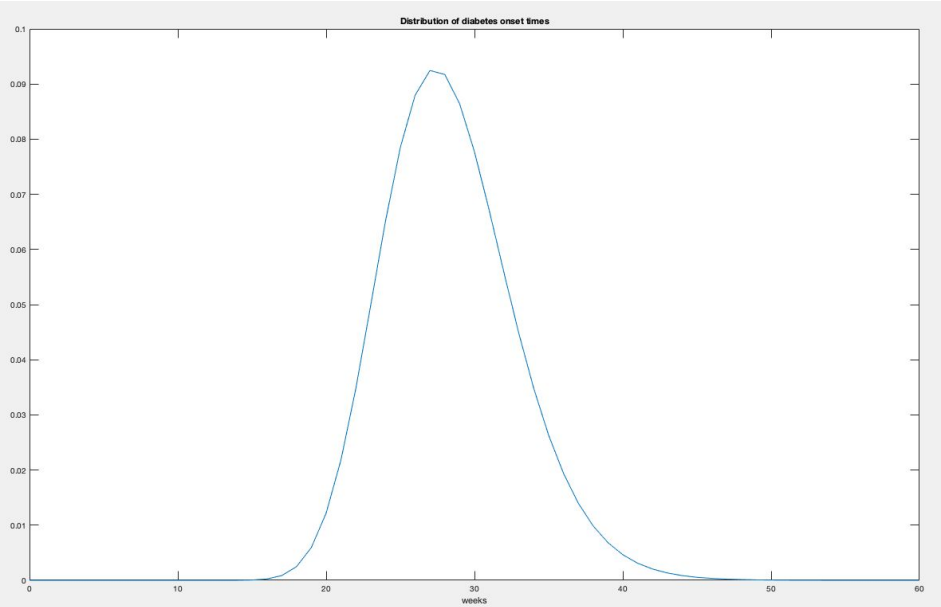
Acute




```
Lognormal distribution
     mu =  4.65979    [4.62775, 4.69183]
  sigma = 0.195195    [0.175019, 0.22067]
```

Li et al





- Progressive only: not enough acute mice for a distribution

```
Lognormal distribution
      mu =  3.33283    [3.21342, 3.45223]
   sigma = 0.155336    [0.104923, 0.297587]
```

Using diabetes onset distribution

Currently

- The given time for glucose measurements is relative

Goal: Use onset distributions to determine a more “fair”/”true” times span for the ODE solver.

1. Obtain lognormal mean and var from distributions
2. Randomly select an onset time
3. Construct time span from that time point
4. Use constructed time in ODE solver

Uses

- Allows us to use Mathews mean data
- Less interpolation
- Possibility of using both Li and Mathews data sets together
  - pre-diabetes = Mathews
  - onset distributions used to determine disease onset
  - post-diabetes = Li

Changes to MCMC

New parameters

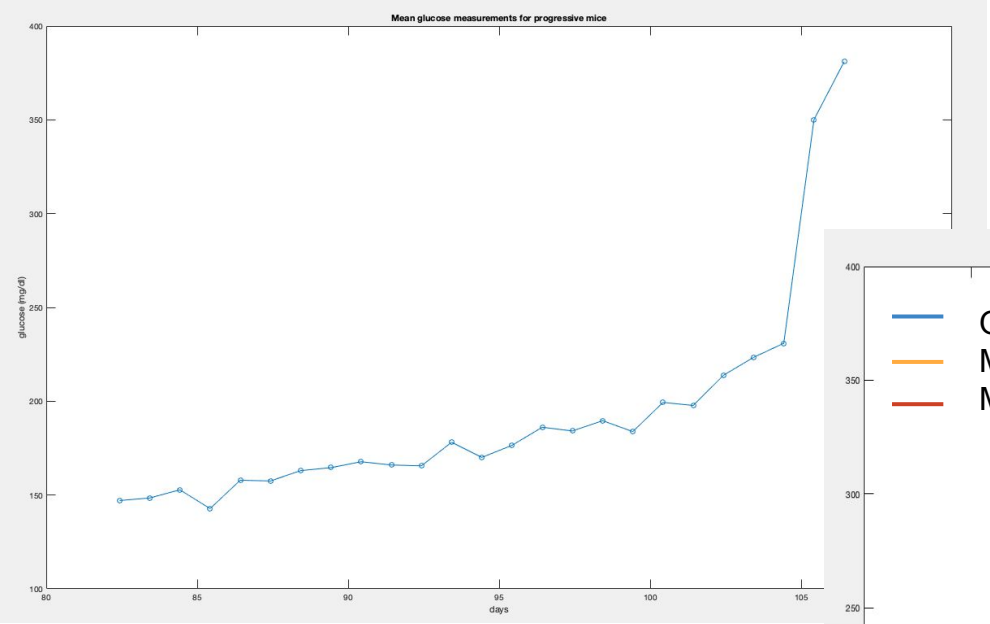
- Glucose + eFAST sensitive
- Mean and variance of time
- Initial conditions

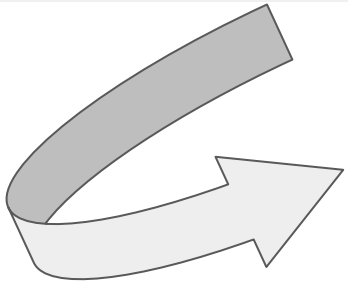
Sum of squares (likelihood)

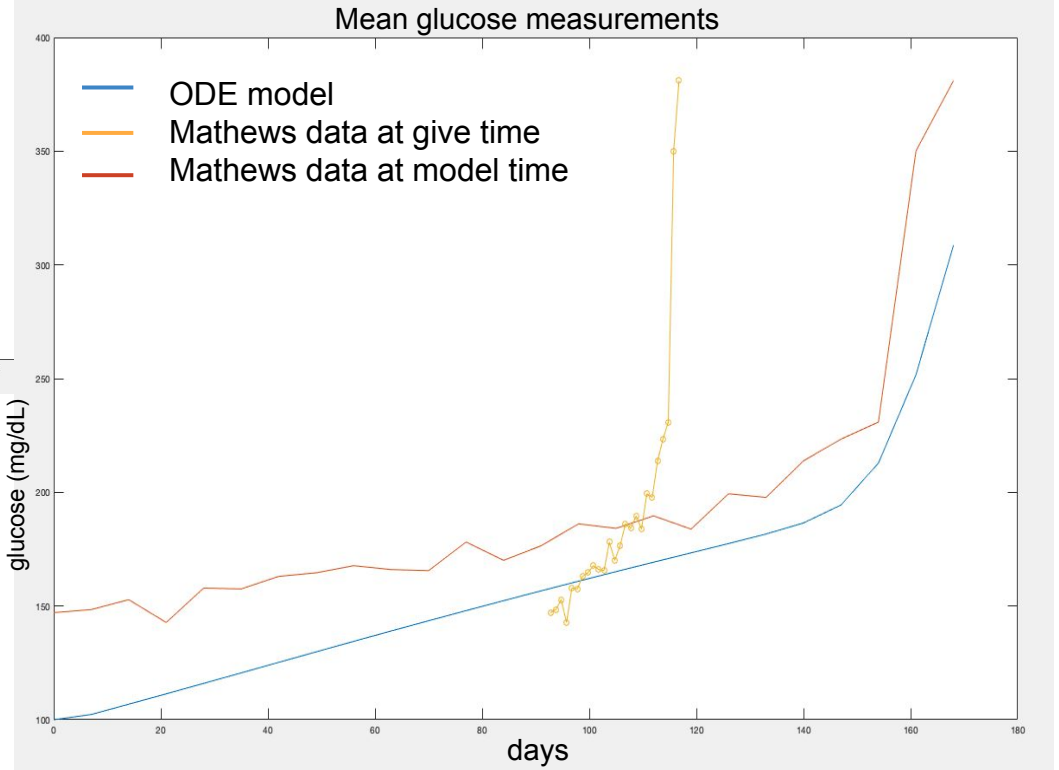
- Solving the ODE for a “large” time range
- Select glucose data from specific time span using onset distributions

```
function ss = T1Dss(params, data)
% Sum-of-squares function

% EXTRACT GLUCOSE DATA
ydata=data(:,2);

% SOLVE ODE MODEL
% Ynowave: no wave wild type (1)
% Ynnowave: NOD no wave (2)
% Ywave: wild type + wave (3)
% Ynwave: NOD + wave (4)
[tmodel ymodel]= T1Dfun_pred(2,data,params);

% DETERMINE ONSET TIME
mu = params(end-1);
sigma = params(end);
onset_time=round(exp(mu+sigma^2/2));% convert to normal from lognormal
onset = ceil(onset_time/7);

% SELECT GLUCOSE DATA
tmodel=tmodel(1:onset+(length(ydata)-onset));
ymodel=ymodel(1:onset+(length(ydata)-onset));

% SUM OF SQUARES
ss = sum((ymodel-ydata).^2);
end
```

Mathews et al mean progressive data time transformation







MCMC cont.

Current issues

- MCMC is rejecting every sample
- Unsure if time parameters are impacting this
- How does varying initial conditions affect the ODE solver?

PSO

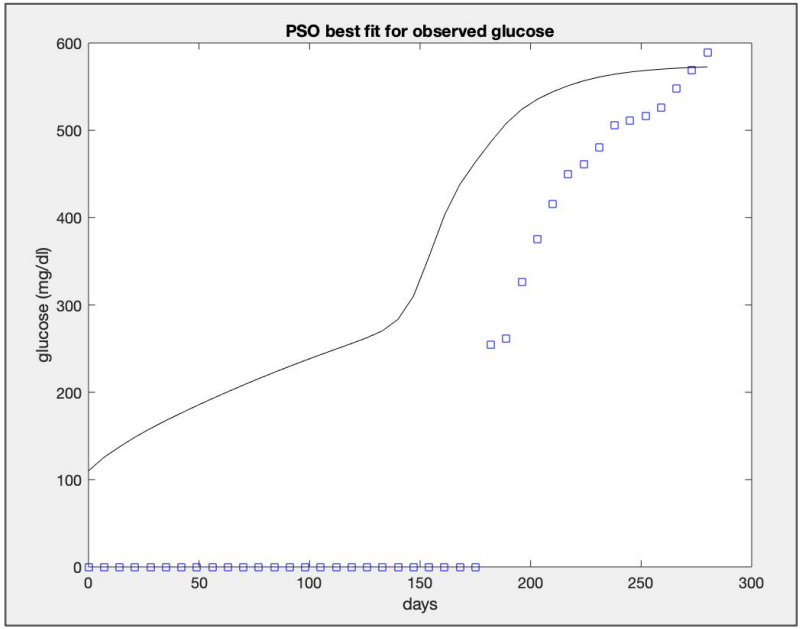
With some issues with MCMC... PSO!
(As a proof of concept)

- Varying onset time does improve fit
- Goodness of fit limited by lack of pre-onset data points
  - Large variability in fits
  - Will try to fix by fusing datasets at onset point (today's project!)

Shifting window of comparison for post-onset data

```
function ss = T1Dss(params, obsPts, waveon, type, wICs)
% Solve the ODE, find residual for glucose
[~, modPred] = T1Dfun(params, waveon, type, wICs);

% Shift data according to estimated onset time
mu = params(end-1);
sig = params(end);
onsettime = ceil(exp(mu+sig^2/2)); % Convert lognormal params to normal

try   % Select proper points from model after onset time
    modPred = modPred(onsettime:(onsettime+length(obsPts)-1), 6);
catch % If not enough points to match obsPts, select as many as possible
    modPred = modPred(onsettime:end, 6);
end

% If onset time is too late for all obsPts...
if length(modPred) < length(obsPts)
    obsPts = obsPts((length(obsPts) - length(modPred) + 1):end);
end

res = (modPred - obsPts).*(modPred - obsPts);

% Square the error values and sum
ss = sum(res);
end
```

PSO



- Differences in mean onset time in Mathews vs. Li data causing unwanted shift:
  ~18 weeks vs. ~29 weeks

```
Optimization ended: relative change in the objective value
over the last OPTIONS.MaxStallIterations iterations is less than OPTIONS.FunctionTolerance.
Warning: Best fit parameters exist at boundaries; ub/lb may need to be adjusted
> In fitT1D (line 174)
```

Improvements

- Debugging!
- Combining datasets for better fits pre-onset:
  - Matthews until onset time, Li afterwards
  - Alterations to objective function (two parts, weighting)
- Reclassification of Li et al mice
- Adjust to non-discrete times
  - No more interpolation

Big Picture: Next Steps

Once we have working MCMC...

- Reduce influence of averaging by combining information gained from Kalman filtering for individual mice with MCMC
  a. Kalman filtering for each mouse shapes parameter distribution
  b. These distributions become priors for MCMC

Fri June 26

Overview: this week

1) Work with altered data from last week
2) Changes to MCMC:
    a) Parameter subsets
    b) Parameterizing acute vs. progressive
3) Changes to PSO:
    a) Scoring fit
    b) Parameter subsets, bounds from UKF
    c) Finding distributions
    d) Validation

MCMC review

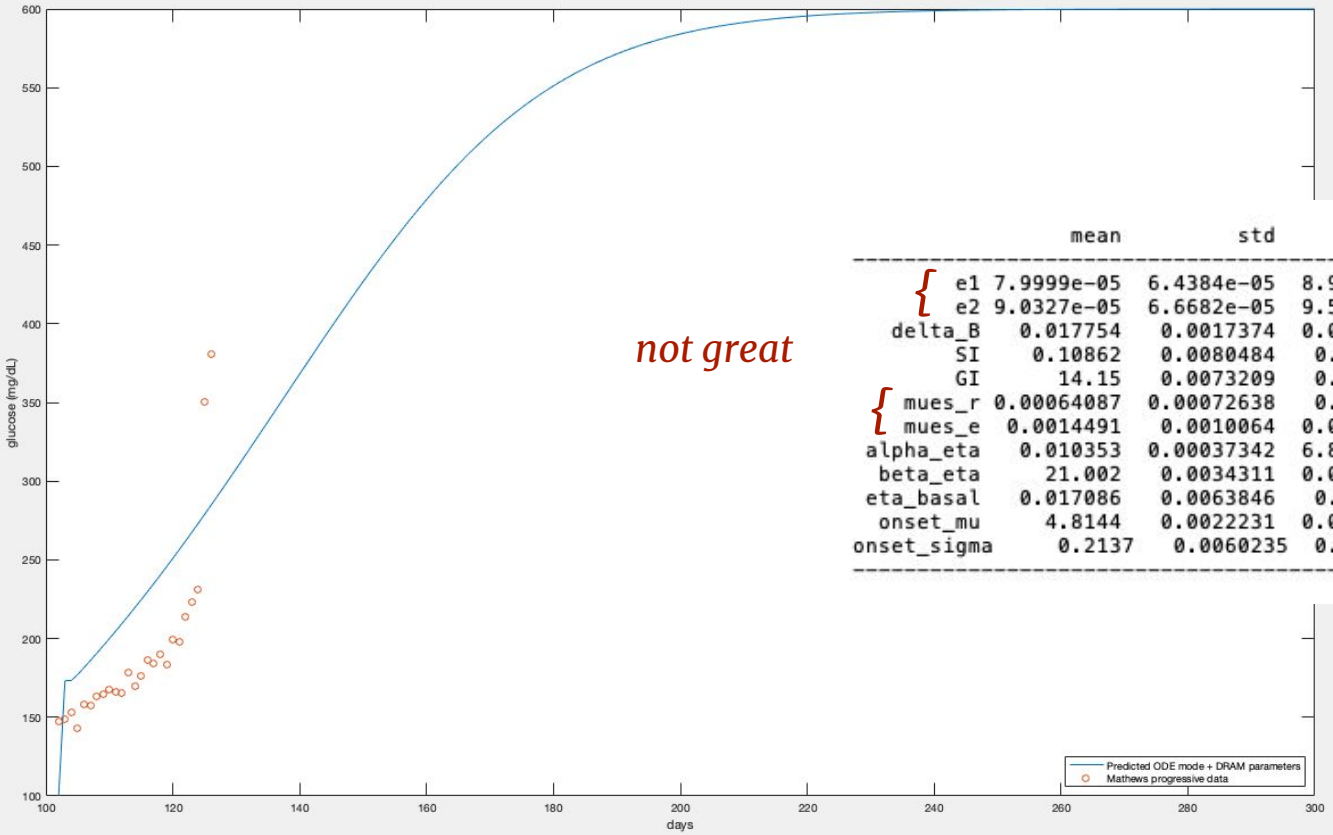
Last week:

- focused on onset time parameters
- debugging: all candidate parameters rejected

This week:

- new parameter subsets + ranges
  - 95% conf. int. : parameters
  - +/- 10%: ICs
- move to parameterizing acute and progressive (Mathews only)
- using Li et al to visually validate predicted models post-DRAM

Some pre-adjustment results



not great

Predicted glucose w/mean param values: *progressive*

	mean	std	MC_err	tau	geweke
e1	7.9999e-05	6.4384e-05	8.9689e-06	330.05	0.017802
e2	9.0327e-05	6.6682e-05	9.5419e-06	370.29	0.017471
delta_B	0.017754	0.0017374	0.00033513	692.21	0.77947
SI	0.10862	0.0080484	0.0016611	783.43	0.80782
GI	14.15	0.0073209	0.0014791	776.3	0.99867
mues_r	0.00064087	0.00072638	0.0001166	418.17	0.025502
mues_e	0.0014491	0.0010064	0.00019087	469.94	0.014461
alpha_eta	0.010353	0.00037342	6.8813e-05	564.24	0.92358
beta_eta	21.002	0.0034311	0.00073701	551.28	0.9997
eta_basal	0.017086	0.0063846	0.0013474	786.94	0.2176
onset_mu	4.8144	0.0022231	0.00039416	513.81	0.9989
onset_sigma	0.2137	0.0060235	0.00086382	156.13	0.99815

Parameter means and convergence

Changes to MCMC

Parameter subsets

1) **7 sensitive (subteam 2) + eta_vary + initial conditions**
2) Glucose, eta_vary, eFAST sensitive, time, initial conditions
3) All eFAST, time, initial conditions

```
params = {
% sensitive parameters (D. Shenker's table)
{'e1', init(5), lb(5), ub(5)}
{'e2', init(6), lb(6), ub(6)}
{'delta_B', init(10), lb(10), ub(10)}
{'SI', init(15), lb(15), ub(15)}
{'GI', init(18), lb(18), ub(18)}
{'mues_r', init(34), lb(34), ub(34)}
{'mues_e', init(35), lb(35), ub(35)}

% eta_vary parameters
{'alpha_eta', init(40), lb(40), ub(40)}
{'beta_eta', init(41), lb(41), ub(41)}
{'eta_basal', 0.0100, 0.0100, 0.0300} % taken from An's eFAST parameters

% Vary non-zero initial conditions (IC)
% IC = [4.77*10^5 0 0 0 300 100 10 D0 tD0 0 0 0];
{'Initial_condition', initIC(1), lbIC(1), ubIC(1)}
{'Beta_cells', initIC(2), lbIC(2), ubIC(2)}
{'Glucose', initIC(3), lbIC(3), ubIC(3)}
{'Insulin', initIC(4), lbIC(4), ubIC(4)}

};
```

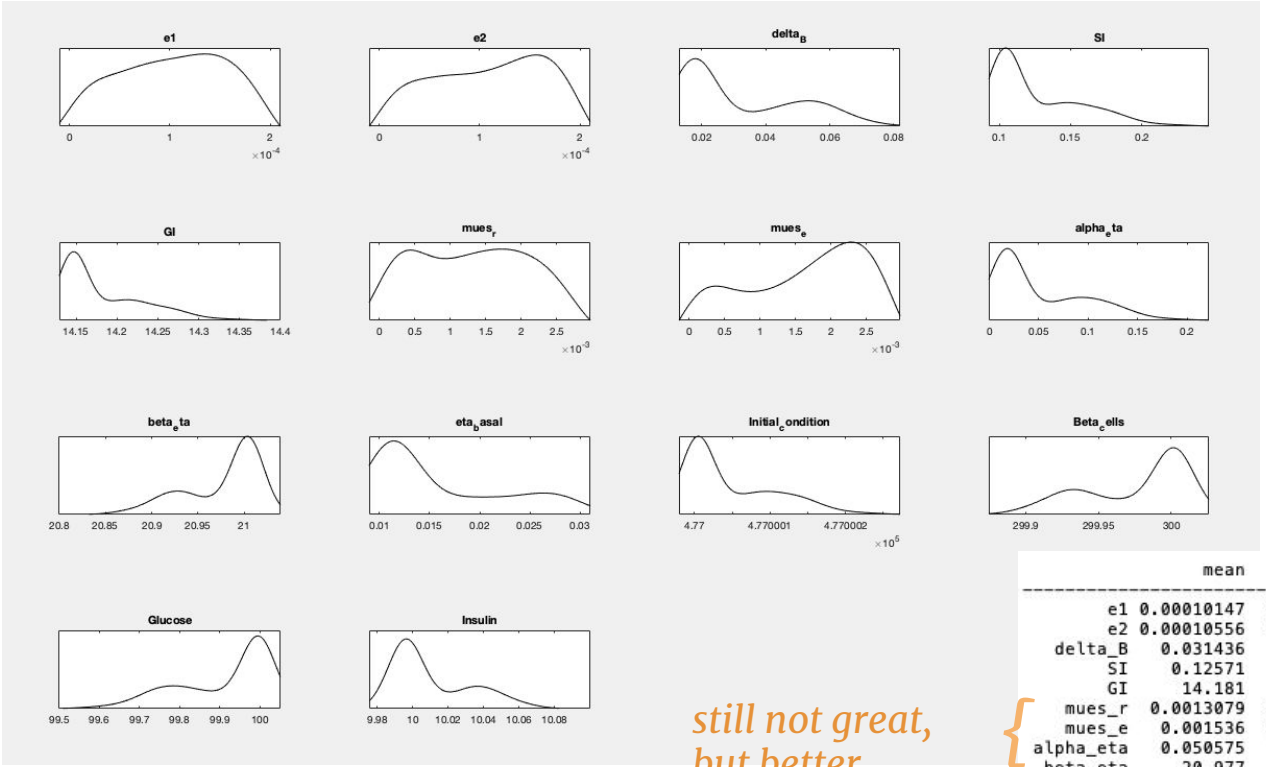
Should we parameterize time in DRAM?

Ultimately, NO

Why?

- *fitdist* already gives a best guess
- parameterizing, won't vary that distribution very much
- did not account for error in onset time distribution (in ss)

Post-adjustment results - *progressive*



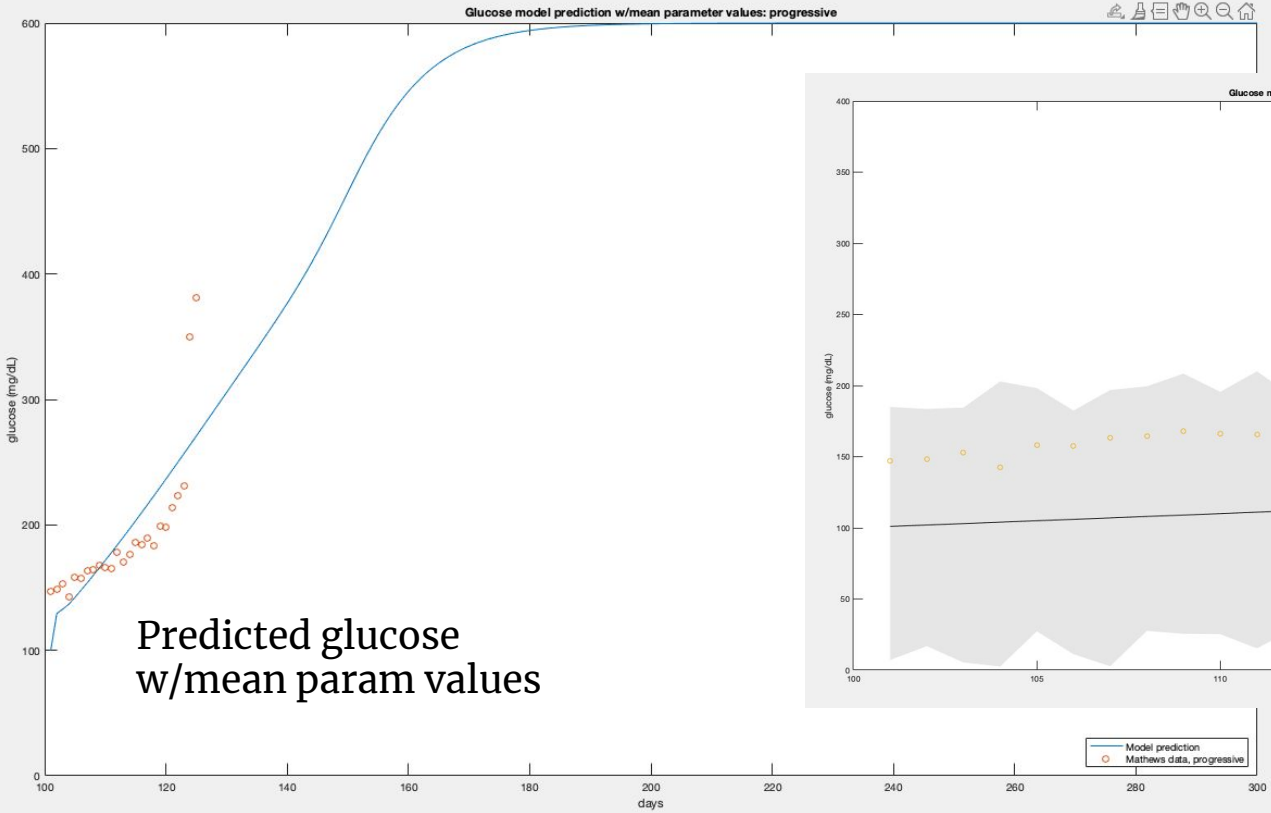
Parameter posterior densities

still not great, but better

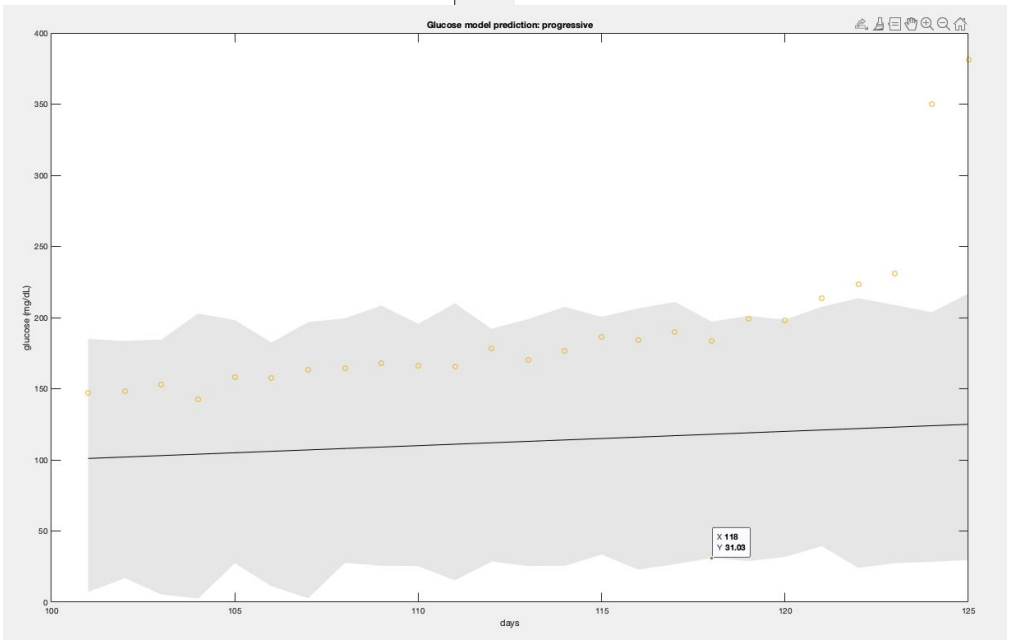
Parameter means and convergence

	mean	std	MC_err	tau	geweke
e1	0.00010147	5.5789e-05	4.2772e-06	30.528	0.95242
e2	0.00010556	5.8919e-05	5.6715e-06	55.449	0.57405
delta_B	0.031436	0.017685	0.003903	782.95	0.15609
SI	0.12571	0.030095	0.0066192	744.84	0.51714
GI	14.181	0.048636	0.010692	744.97	0.99282
mues_r	0.0013079	0.00081087	0.00010417	194.56	0.07199
mues_e	0.001536	0.00088633	0.00014983	518.25	0.023482
alpha_eta	0.050575	0.044861	0.0098474	749.95	0.04665
beta_eta	20.977	0.040183	0.0086973	778.79	0.99656
eta_basal	0.01592	0.0064078	0.0013155	712.72	0.28171
Initial_condition	4.77e+05	0.058306	0.012864	757.79	1
Beta_cells	299.98	0.034556	0.0075763	796.6	0.99977
Glucose	99.913	0.11243	0.024802	772.86	0.99762
Insulin	10.01	0.021539	0.0046626	789.06	0.99638

Post-adjustment results - *progressive*

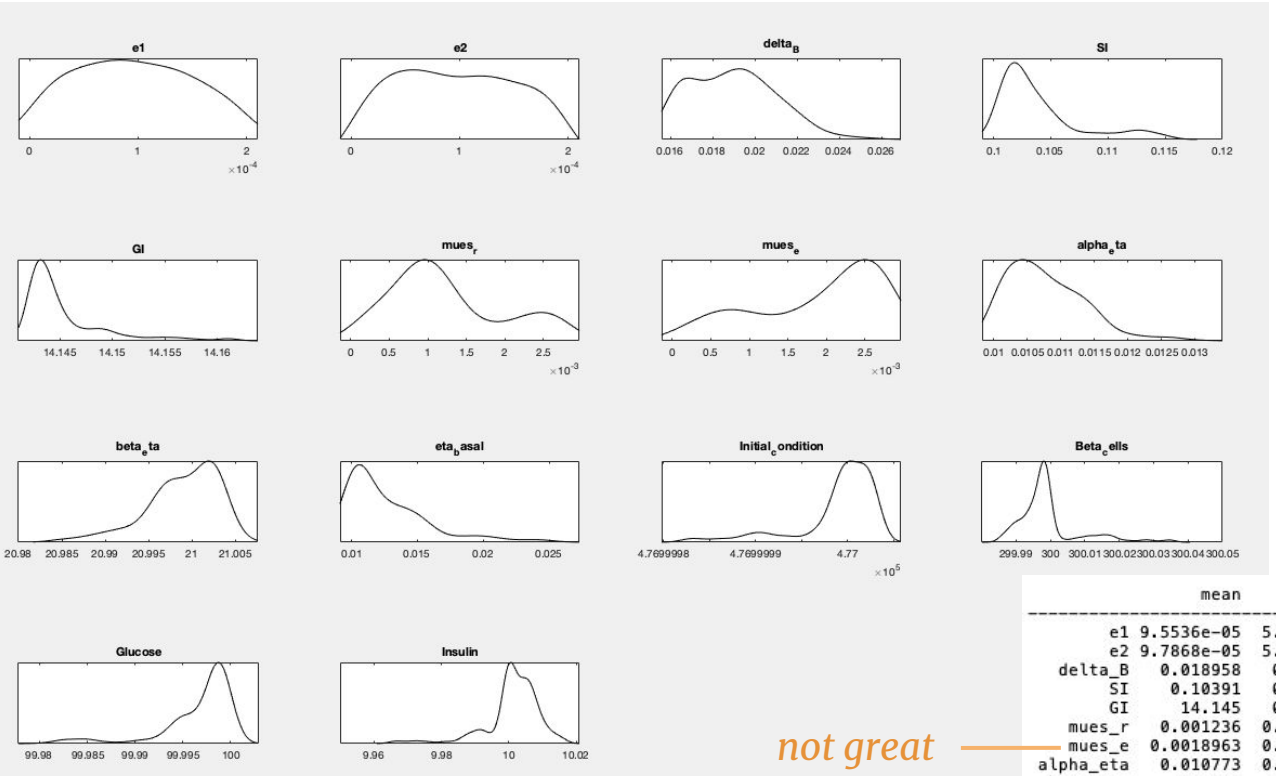


Predicted glucose w/mean param values



mcmcstat model prediction w/confidence interval

Post-adjustment results - *acute*



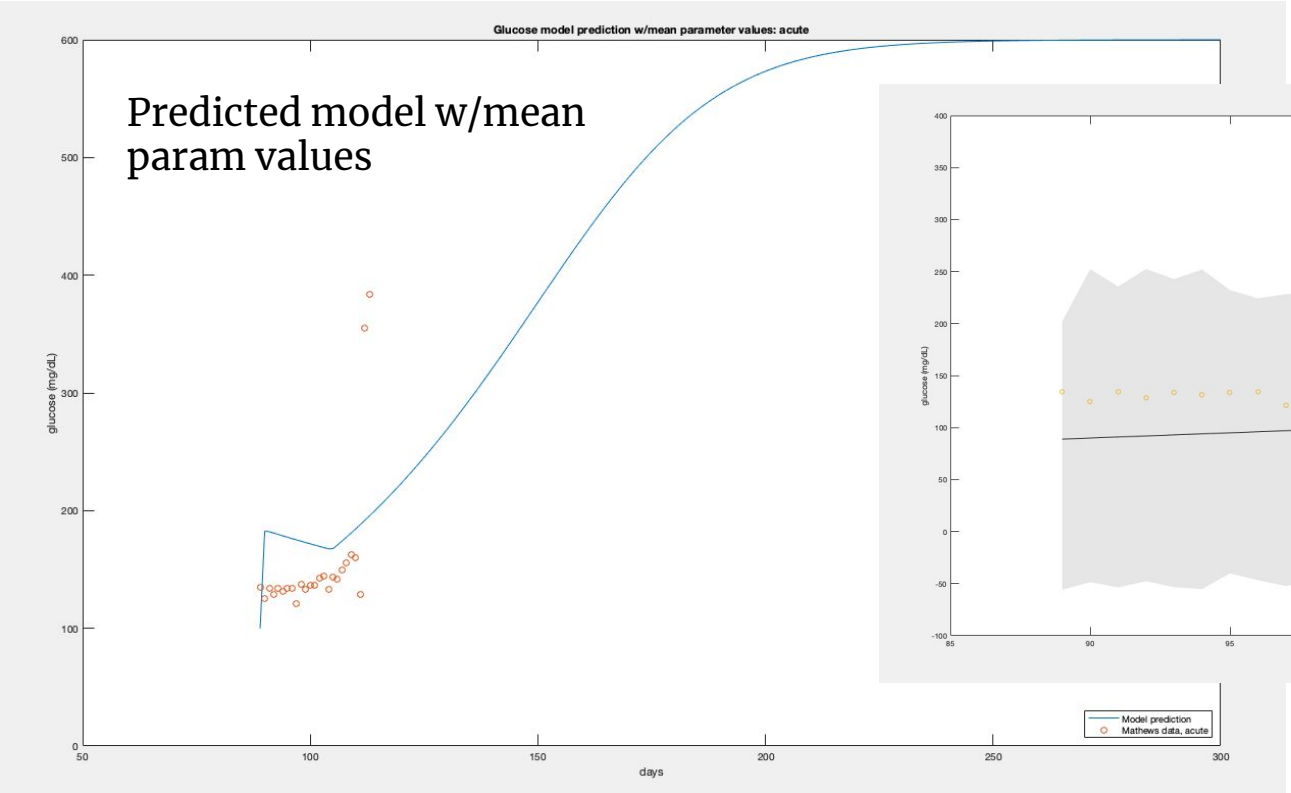
Parameter posterior densities

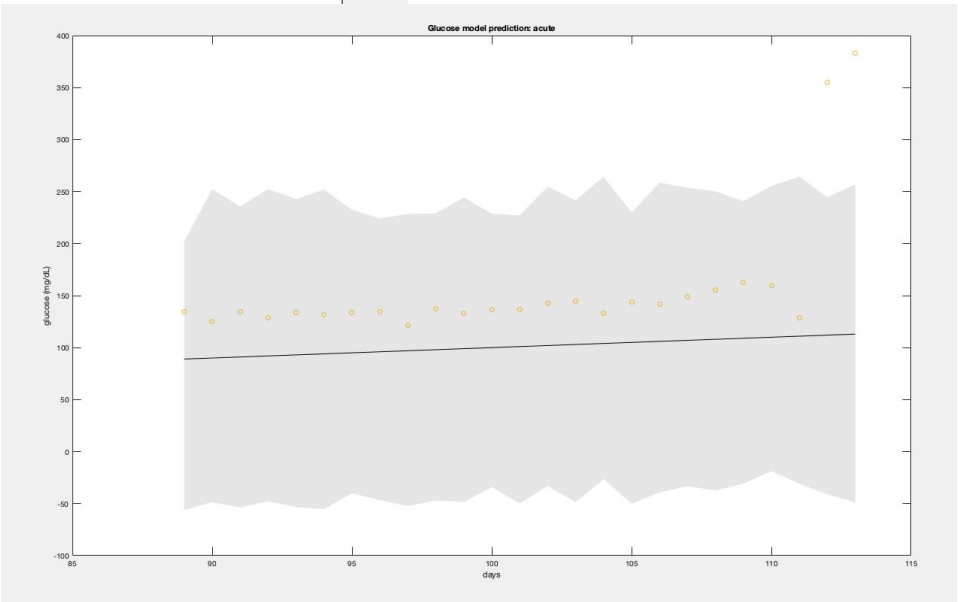
Parameter means and convergence

not great

	mean	std	MC_err	tau	geweke
e1	9.5536e-05	5.3871e-05	5.4333e-06	71.235	0.59923
e2	9.7868e-05	5.7116e-05	7.0039e-06	113.16	0.97791
delta_B	0.018958	0.0019849	0.00035338	394.51	0.73209
SI	0.10391	0.0035974	0.00075046	423.16	0.95594
GI	14.145	0.0039159	0.00080242	342.42	0.99959
mues_r	0.001236	0.00073389	0.00014548	557.73	0.14745
mues_e	0.0018963	0.00081511	0.00016066	644.27	0.070611
alpha_eta	0.010773	0.00056646	0.00010028	547.16	0.92681
beta_eta	20.999	0.0041251	0.00083928	379.57	0.9997
eta_basal	0.012668	0.0031003	0.00064446	516.17	0.55135
Initial_condition	4.77e+05	0.0044439	0.0009605	408.28	1
Beta_cells	300	0.0092251	0.0019484	527.55	0.99999
Glucose	99.997	0.0039484	0.00083399	568.2	0.99995
Insulin	10.001	0.0082674	0.0016756	510.22	0.99993

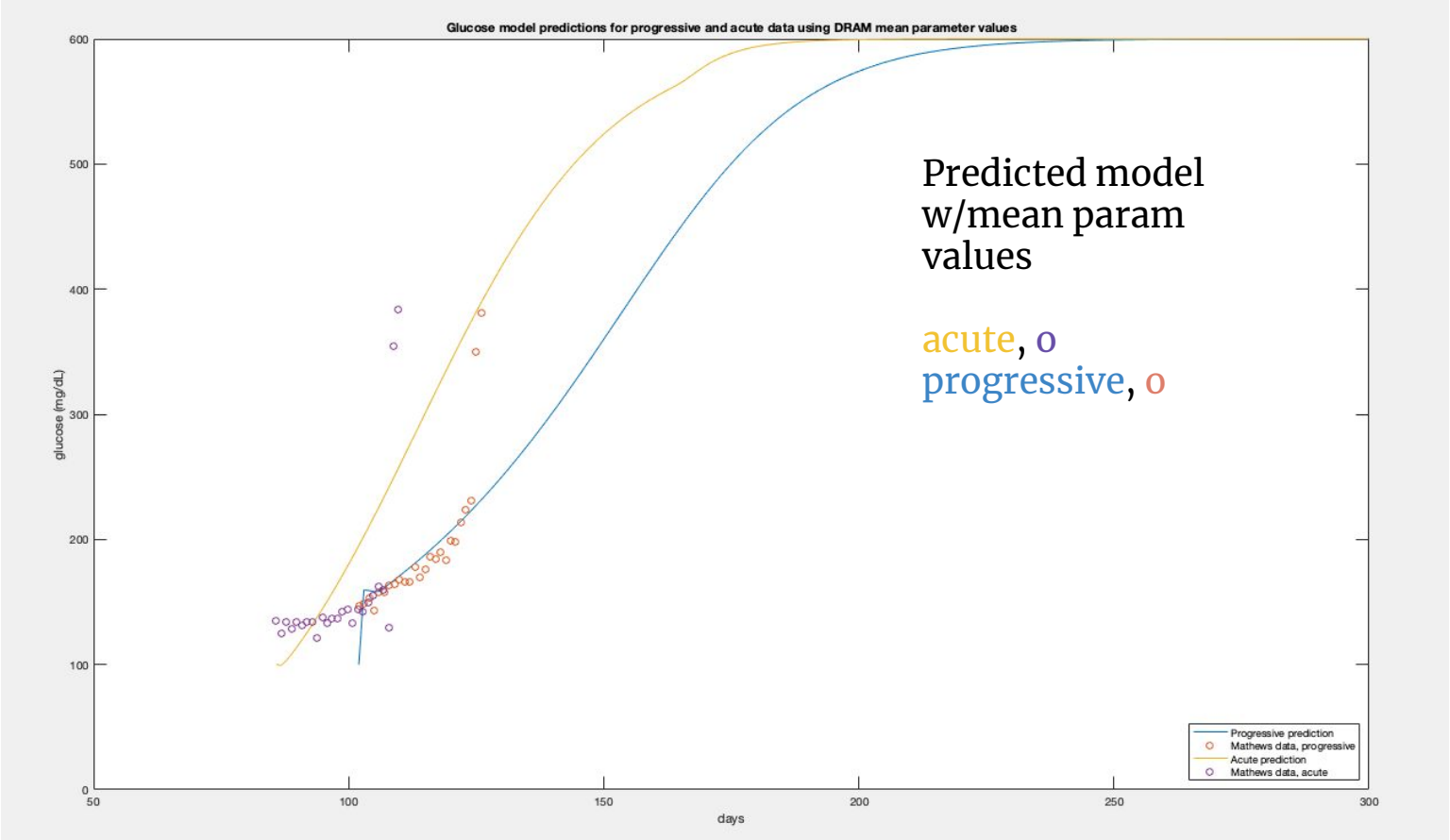
Post-adjustment results - *acute*



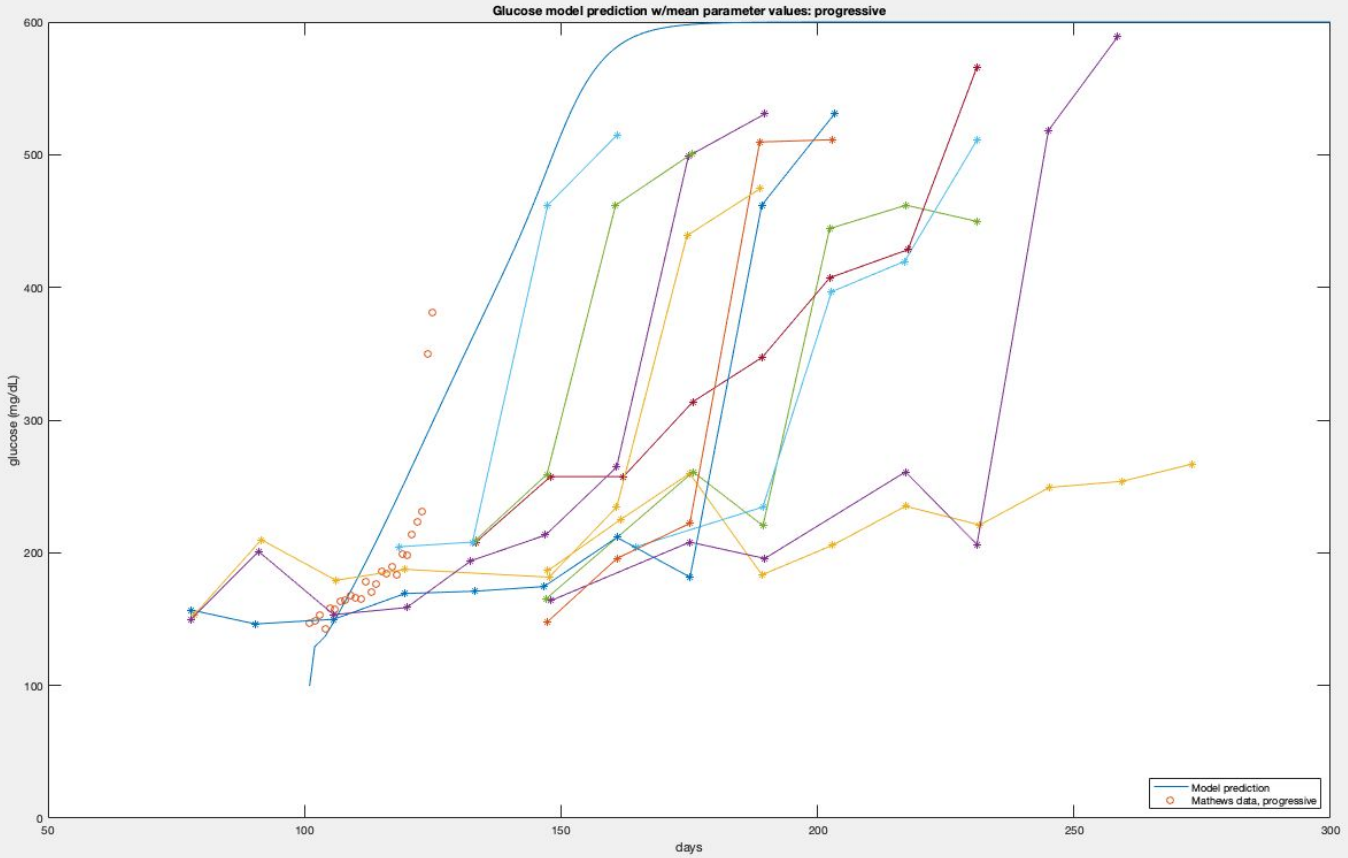


mcmcstat model prediction w/confidence interval

Comparing *acute* & *progressive*



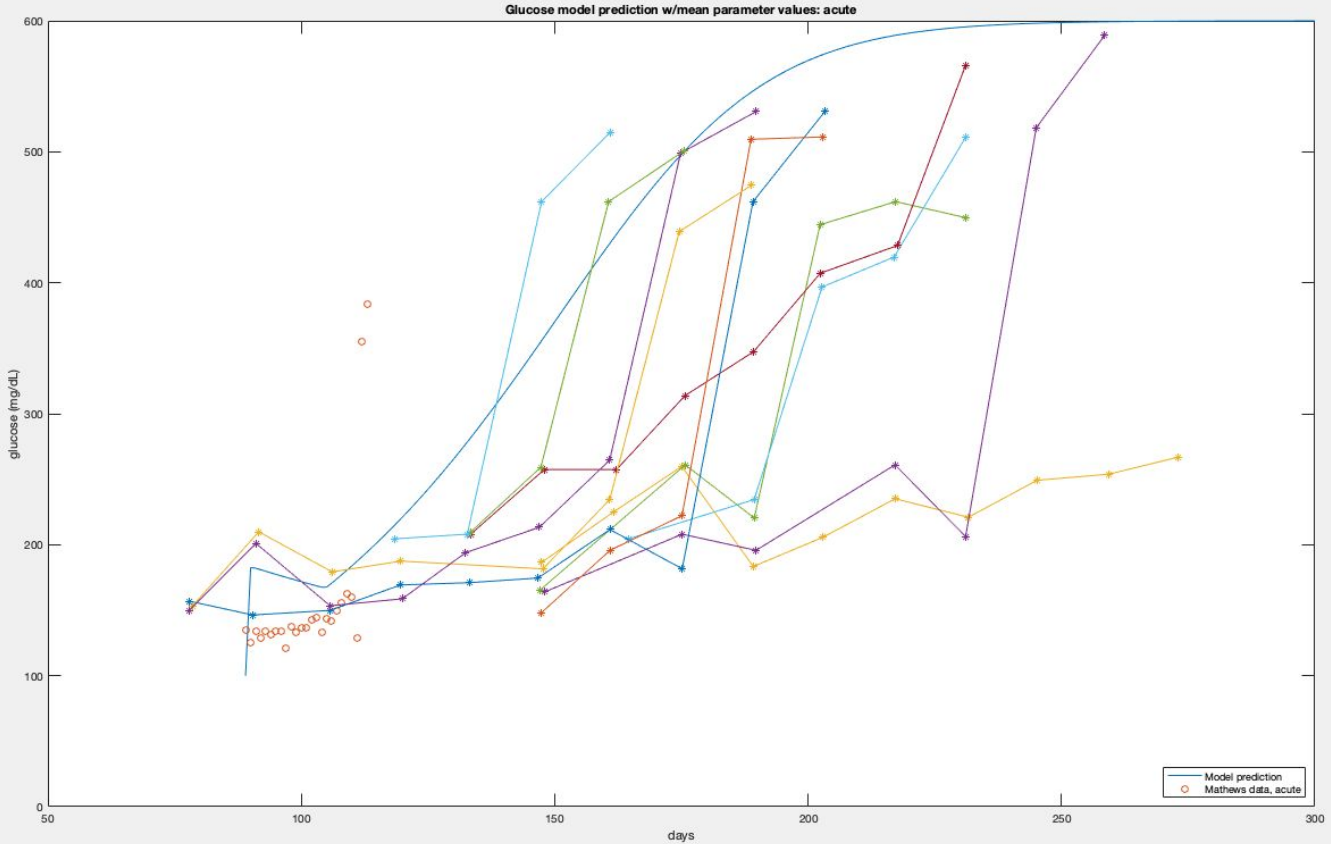
Basic validation MCMC



progressive

Predicted model w/mean param values + Li et al data

Basic validation MCMC

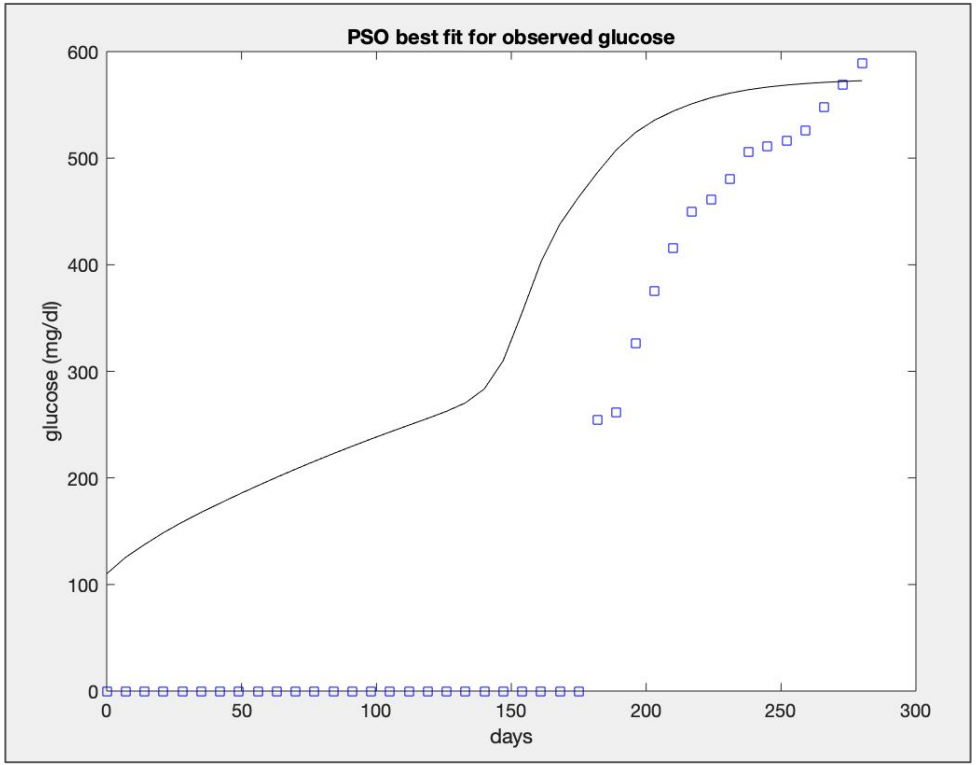


acute

Predicted model w/mean param values + Li et al data

Changes to PSO

- Last week: poor shifting, unknown beginning shape, fit judged visually
- This week: Fixed shifting, more data points at beginning, fit judged numerically
  - Averaged datasets combined at mean of lognormal onset time distribution
  - Score assigned based on onset time, shape



(Last week)

Scoring a fit

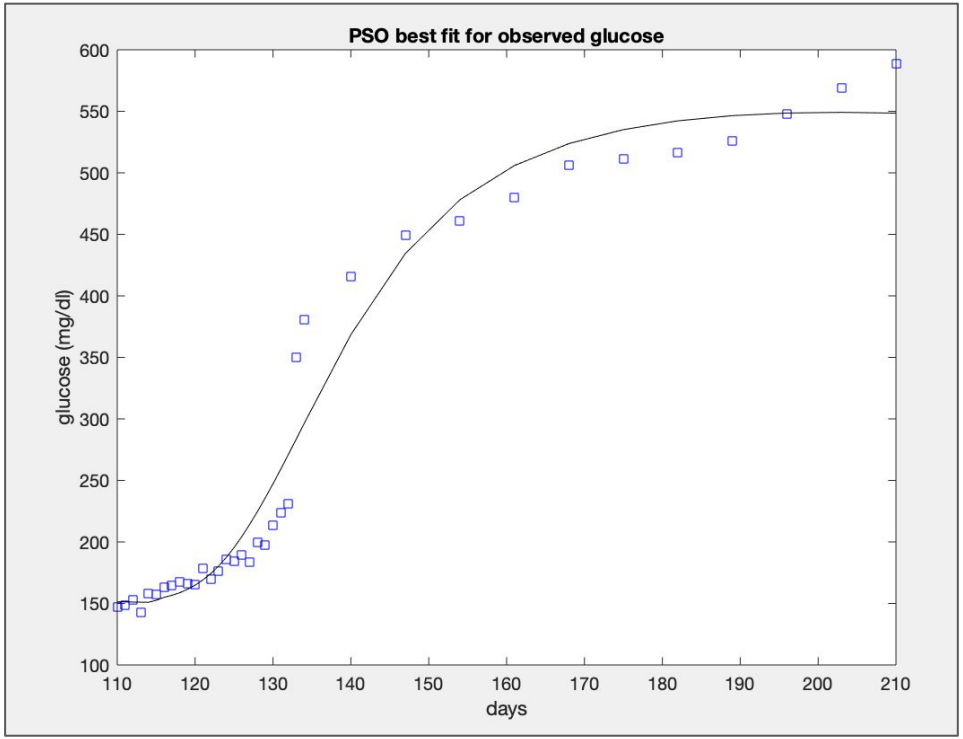
1. Simulate ODE with fitted parameters
2. Determine onset time of model, t_{mod} , and of data, t_{data}
3. Calculate time score:

$$S_{time} = | t_{mod} - t_{data} |$$

4. Segment data, model pre/post onset
5. Determine sum-of-squares difference between corresponding segments:

$$S_{shape} = (y_{mod,\,pre} - y_{data,\,pre})^2 + (y_{mod,\,post} - y_{data,\,post})^2$$

6. Combine time, shape score according to user-provided function



(This week)

Parameter subsets

- Refresher:
  - Subsets: all, glucose-involved, sensitive, glucose+sensitive
  - All parameters allow to ‘wiggle’, but only subset parameters allows to truly vary
- New subsets:
  - Eta parameters
  - ‘Active’ parameters from UKF (>1% change from baseline)
- Altered bounds using variances from Subteam 2
  - Baseline as mean, 95% confidence interval as bounds

Fit Scores for Average **Mathews** Data

Parameter Subset	Time Score	Shape Score
All	2	7.618×10^4
Glucose	27	6.067×10^4
Sensitive	27	6.834×10^4
Glucose+Sensitive	27	7.392×10^4
Active	2	8.840×10^4
Eta	2	9.122×10^4

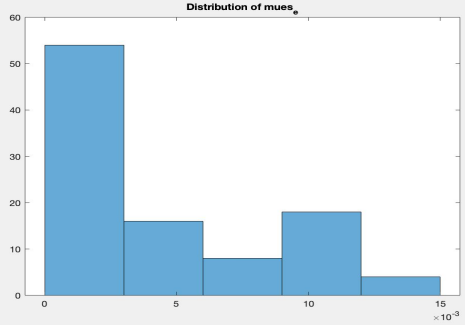
Parameter distributions

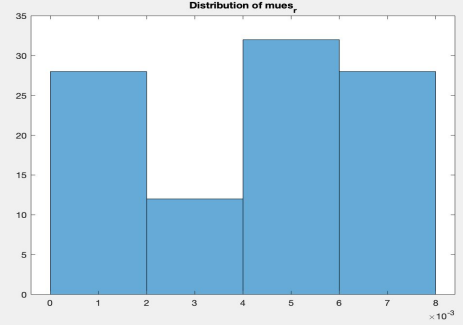
- PSO is a global algorithm… *ideally*
- Unlike MCMC, only produces one best fit

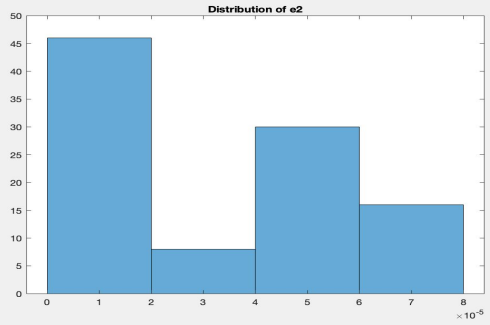
Process:

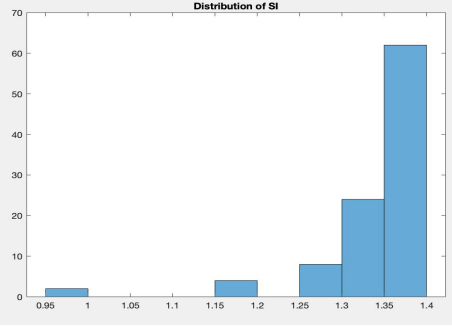
- Run fitting process many times, produce histograms for each parameter
- Fit distributions

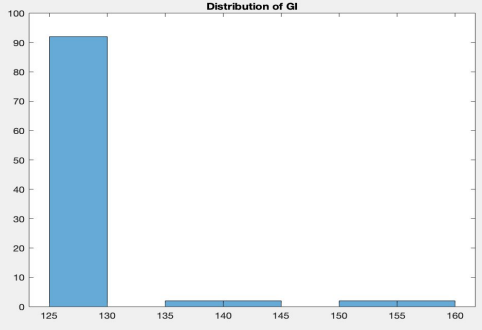
Limitations: needed reduced swarm size, iterations for reasonable run time (~5 hours)

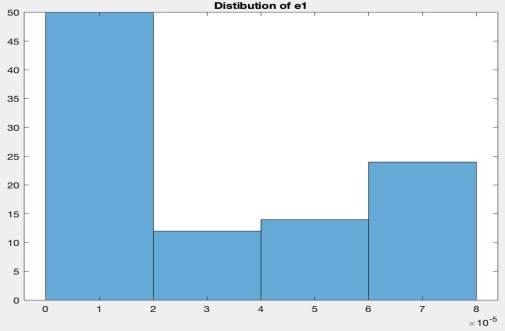

Distribution of mues$_e$
×10^-3

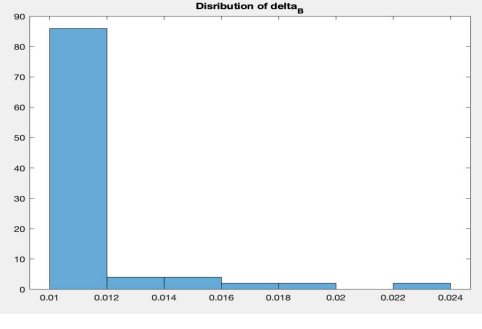

Distribution of mues$_r$
×10^-3

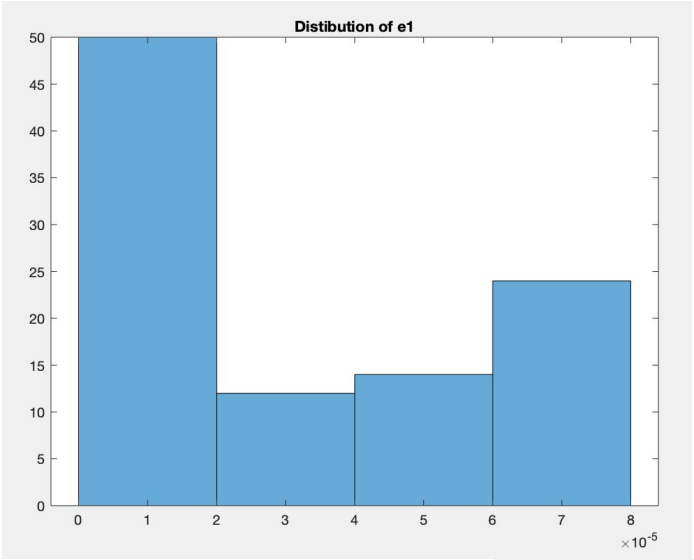

Distribution of e2
×10^-5

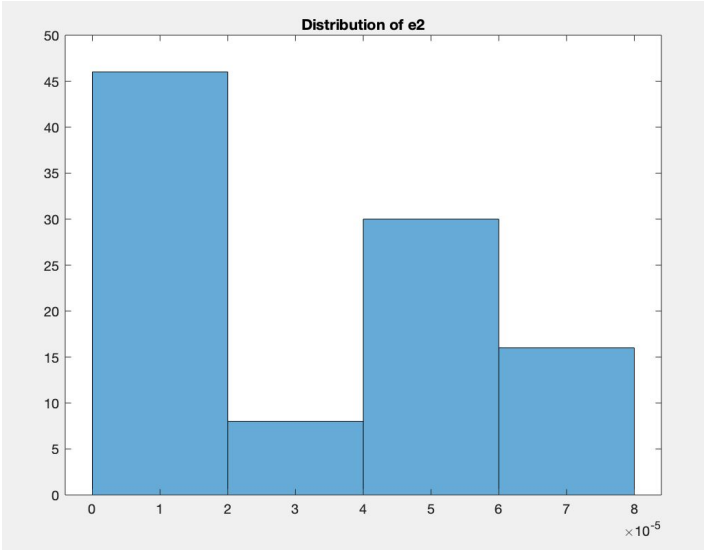

Distribution of SI

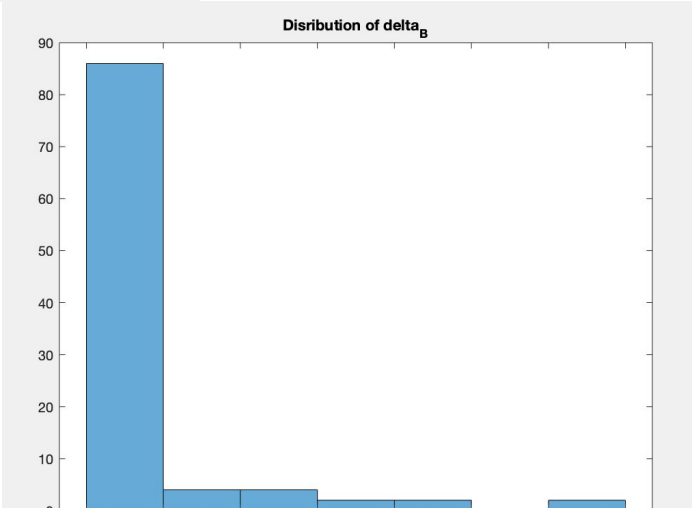

Distribution of GI


Distibution of e1
×10^-5


Disribution of delta$_B$


Distibution of e1
50
45
40
35
30
25
20
15
10
5
0
0 1 2 3 4 5 6 7 8
×10^-5


Distribution of e2
50
45
40
35
30
25
20
15
10
5
0
0 1 2 3 4 5 6 7 8
×10^-5


Disribution of delta_B
90
80
70
60
50
40
30
20
10

Validation

- If fit with Mathews, are the parameters reasonable for Li?
  - Training data: Mathews
  - Validation data: Li
  - Metric: visual inspection, fit scores
- Today's task: Is reasonable behavior produced in other species? For other mouse species/wave occurrences?

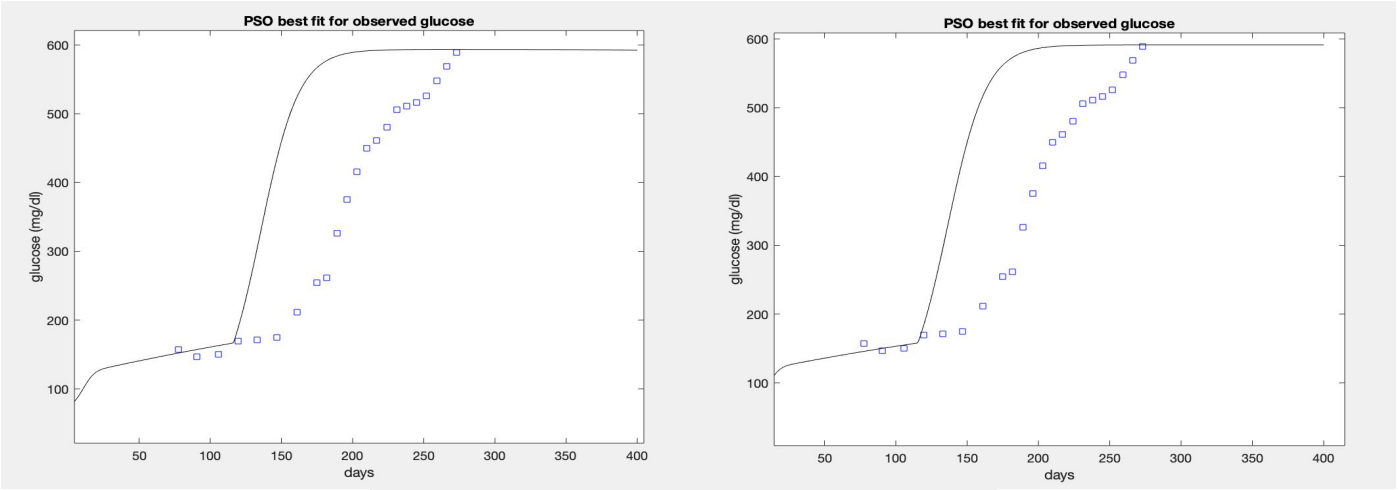
Validation (cont.)

- Using modes, medians, means from histograms made by fitting Mathews
- Initial concern due to onset time differences between datasets

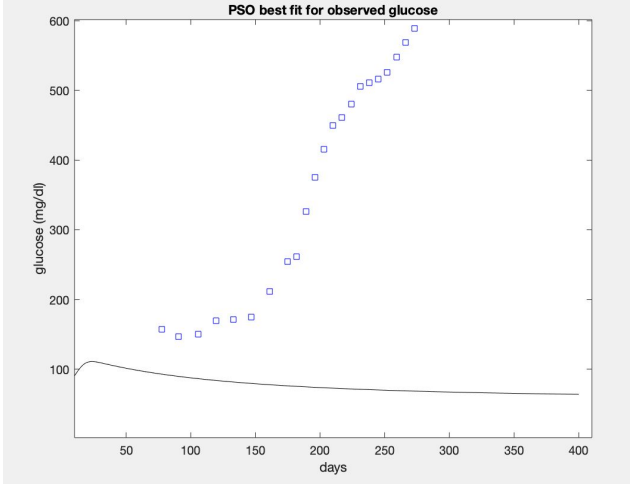
Fit Scores for Average **Li** Data (w/ Mathews Parameters)

Parameter Set	Time Score	Shape Score
Mean	103*	7.359×10^4
Median	104*	7.270×10^4
Mode	Never reaches diabetes	1.147×10^5

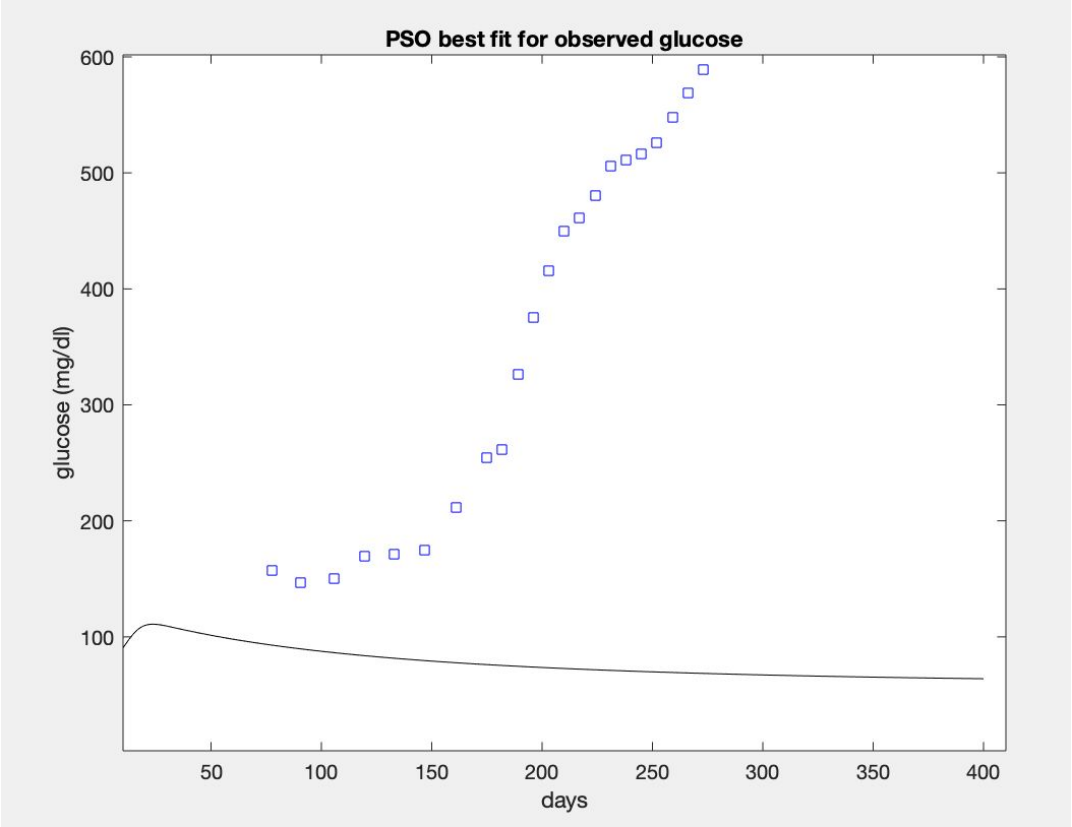
*would expect to be around 49 days off due to shifted data, still leaves 54/55 day offset


PSO best fit for observed glucose
glucose (mg/dl)
days
600
500
400
300
200
100
50
100
150
200
250
300
350
400

PSO best fit for observed glucose
glucose (mg/dl)
days
600
500
400
300
200
100
50
100
150
200
250
300
350
400


PSO best fit for observed glucose
glucose (mg/dl)
days
600
500
400
300
200
100
50
100
150
200
250
300
350
400

mode


PSO best fit for observed glucose
glucose (mg/dl)
600
500
400
300
200
100
50
100
150
200
250
300
350
400
days

Next Steps

MCMC

1. Run again with more samples - 10,000
2. Validation method - quantifiable
3. Looking at other variable predictions post-DRAM

PSO

1. More subset combos
2. Determine how to handle irregular distributions
3. Run loops again; longer this time!

General

1. Compare outputs of PSO, DRAM
2. Refresh Lotka-Volterra DRAM algorithm
    a. possibly add validation step
3. Prepare big REU slides
    a. going back to the basics (Bayes')
4. Organize repo folders & carefully comment code/READMEs
5. Begin writing report :)