Population and ODE-based models using Stan and Torsten

Charles Margossian, Yi Zhang

StanCon 2019, Cambridge UK
August 2019
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
Instructors

- Charles Margossian
  - Columbia University, Department of Statistics
- Yi Zhang
  - Metrum Research Group

TA

- Ben Bales
  - Columbia University
Outline

Day 1

▶ Introduction and modeling framework
▶ Pharmacometrics models
▶ Ordinary differential equation (ODE) based models
▶ Numerical ODE integrators

Day 2

▶ Population models
▶ Group/Population ODE integrators and MPI parallelisation
▶ Group/Population solvers and MPI parallelisation
METWORX™, cloud-based modeling & simulation platform by Metrum Research Group.
Logistics

Workshop package
- R scripts and Stan files to do the exercises
- These slides
- Outline of the course

We will be using:
- Torsten v0.87
- RStan v2.19.2
- ggplot, plyr, tidyr, dplyr
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
Preliminary question

- Why Bayesian in a field such as pharmacometrics?
- Example - *Bayesian aggregation of average data: an application in drug development* [Weber et al., 2018]
Modeling framework

Box’s loop
Inference

- find the set of parameters consistent with our model and our data
- approximate this set with draws from the posterior distribution
Sampling algorithm

- Use dynamic HMC to sample $\pi(\theta|y)$
- Requires users to specify $\log \pi(\theta, y) = \log \pi(y|\theta) + \log \pi(\theta)$
The "criticism" step

This step can be broken up in two parts:

1. did we sample from the correct distribution?
2. does our model capture the characteristics of the data we care about?
Diagnosing the inference algorithm

- look at the trace and the density plots
- look at $\hat{R}$ and effective number of samples
- have any warning messages been issued, i.e. divergent transitions?
Example: fitting a linear model

Likelihood:

\[ Y \sim \text{Normal}(x\beta, \sigma^2) \]

Prior:

\[ \beta \sim \text{Normal}(2, 1) \]
\[ \sigma^2 \sim \text{Normal}(1, 1) \]
### $summary$

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>se_mean</th>
<th>sd</th>
<th>2.5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>97.5%</th>
<th>n_eff</th>
<th>Rhat</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta</td>
<td>5.601258</td>
<td>0.01359227</td>
<td>0.5305772</td>
<td>4.479154</td>
<td>5.264460</td>
<td>5.614632</td>
<td>5.966383</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigma</td>
<td>9.502691</td>
<td>0.04383169</td>
<td>1.6813433</td>
<td>6.859379</td>
<td>8.320122</td>
<td>9.282212</td>
<td>10.454978</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lp__</td>
<td>-45.636140</td>
<td>0.02492619</td>
<td>1.0048605</td>
<td>-48.314041</td>
<td>-46.014181</td>
<td>-45.318003</td>
<td>-44.916883</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>se_mean</th>
<th>sd</th>
<th>2.5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>97.5%</th>
<th>n_eff</th>
<th>Rhat</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta</td>
<td>6.570396</td>
<td>1523.749</td>
<td>0.9998578</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigma</td>
<td>13.457200</td>
<td>1471.419</td>
<td>1.0013391</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lp__</td>
<td>-44.651010</td>
<td>1625.173</td>
<td>1.0002468</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
So, how can we improve the model?

Likelihood:

\[ Y \sim \text{Normal}(x\beta, \sigma^2) \]

Prior:

\[ \beta \sim \text{Normal}(2, 1) \]
\[ \sigma^2 \sim \text{Normal}(1, 1) \]
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
What is the effect of a treatment on a patient?

- **pharmacokinetics (PK):** how is the drug absorbed in the body?
- **pharmacodynamics (PD):** once it is absorbed, what are its effects?
Example: Two compartment model

- **GUT**
  - Input: \( ka \)
  - Description: Organs and tissues into which the drug gets absorbed slowly

- **CENTRAL COMPARTMENT**
  - Description: Blood and organs into which the drug gets absorbed rapidly
  - Output: \( CL \)

- **PERIPHERAL COMPARTMENT**
  - Description: Organs and tissues into which the drug gets absorbed slowly
  - Input: \( Q \)
Two compartment model

\[ y'_{gut} = -k_ay_{gut} \]

\[ y'_{cent} = k_ay_{gut} - \left( \frac{CL}{V_{cent}} + \frac{Q}{V_{cent}} \right) y_{cent} + \frac{Q}{V_{peri}} y_{peri} \]

\[ y'_{peri} = \frac{Q}{V_{cent}} y_{cent} - \frac{Q}{V_{peri}} y_{peri} \]
Example 2: Bone mineral density model from [Peterson and Riggs, 2012]
Two compartment model

Denote $\theta = \{ CL, Q, VC, VP, K_a \}$, the ODE coefficients. Then

$$y' = f(y, t, \theta)$$

Given an initial condition $y_0 = y(t_0)$, solving the above ODE gives us the \{natural evolution\} of the system at any given time point.
The event schedule

An event can be:

▶ **a state changer**: an (exterior) intervention that alters the state of the system; for example a bolus dosing or the beginning of an infusion.

▶ **an observation**: a measurement of a quantity of interest at a certain time.
Drug concentration in a patient’s blood
The event schedule

- There is no general theory for the event schedule :(  
- The modeling software NONMEM® proposes a convention for pharmacometrics, which we adopt in Torsten.
Torsten functions

Torsten functions offers additional built-in functions to simulate data from a compartment model.

Each Torsten function requires users to specify:
- a system of ODEs and a method to solve it.
- An event schedule.
Torsten functions

\[
\text{matrix} = \text{pmx\_solve\_onecpt(real[] time, real[] amt, real[] rate, real[] ii, int[] evid, int[] cmt, real[] addl, int[] ss, real[] theta, real[] biovar, real[] tlag);}
\]

\[
\text{matrix} = \text{pmx\_solve\_twocpt(real[] time, real[] amt, real[] rate, real[] ii, int[] evid, int[] cmt, real[] addl, int[] ss, real[] theta, real[] biovar, real[] tlag);}
\]

- Analytically solutions for the one/two cpt models.
- Event schedule
- ODE coefficients, e.g. \( \theta = \{ CL, Q, VC, VP, ka \} \) for two-cpt model.
- bio-availability fraction and lag times.
Example

Clinical trial

- Single patient
- Bolus doses with 1200 mg, administered every 12 hours, for a total of 15 doses.
- Many observations for the first, second, and last doses.
- Additional observation every 12 hours.

Note: the observation are plasma drug concentration measurement.

See data/twoCpt.data.r.
Example

Model

- two compartment model with first-order absorption
- prior information based on clinical trial conducted on a large population
- normal error for the plasma drug concentration measurement.
Example

Prior

\[ \text{CL} \sim \text{lognormal}(\log(10), 0.25); \]
\[ \text{Q} \sim \text{lognormal}(\log(15), 0.5); \]
\[ \text{VC} \sim \text{lognormal}(\log(35), 0.25); \]
\[ \text{VP} \sim \text{lognormal}(\log(105), 0.5); \]
\[ \text{ka} \sim \text{lognormal}(\log(2.5), 1); \]
\[ \text{sigma} \sim \text{cauchy}(0, 1); \]

Likelihood

\[ \log(\text{cObs}) \sim \text{Normal}\left( \log\left( \frac{y_2}{\text{VC}} \right), \sigma^2 \right) \]

Exercise 1

(a) write and fit this model, using \texttt{twoCptModel.r} and \texttt{model/twoCptModel.stan}.
(b) write a generated quantities block and do posterior predictive checks.
Resources

- Torsten repository:
  https://github.com/metrumresearchgroup/Torsten
- Torsten User manual (on GitHub and in the workshop folder).
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
Arsenal of tools

For some examples, see [Margossian and Gillespie, 2017].

- the "optimized - applicable" spectrum is a heuristic; counter-examples can be built.
- coding effort may also be a criterion
Matrix exponential

Consider a system of linear ODEs:

\[ y'(t) = K y(t) \]

where \( K \) is a constant matrix.

Then

\[ y(t) = e^{tK} y_0 \]
Matrix Exponential

\[ e^{tK} = \sum_{n=0}^{\infty} \frac{(tK)^n}{n!} = I + tK + \frac{(tK)^2}{2} + \frac{(tK)^3}{3!} + \ldots \]
Matrix Exponential

For example, the two compartment model generates the following matrix:

$$K = \begin{bmatrix}
-ka & 0 & 0 \\
ka & -(CL + Q)/Vc & Q/Vp \\
0 & Q/Vc & -Q/Vp
\end{bmatrix}$$
Linear ODE solver in Torsten

```plaintext
matrix = pmx_solve_linode(real[] time, real[] amt, real[] rate,
                          real[] ii, int[] evid, int[] cmt,
                          real[] addl, int[] ss,
                          matrix K, real[] biovar, real[] tlag)
```
Numerical integrator

```c
real[ , ] pmx_integrate_ode_rk45(ODE_RHS, real[] y0, real t0,
                           real[] ts, real[] theta, real[] x_r, int[] x_i, real rtol =
                           1.e-6, real atol = 1.e-6, int max_step = 1e6);
```

- **ODE_RHS**: ODE right-hand-side $f$ in $y' = f(y, t, \theta, x_r, x_i)$.
- **y0**: initial condition at time $t_0$.
- **t0**: initial time.
- **ts**: times at which we require a solution.
- **theta**: parameters to be passed to $f$.
- **x_r**: real data to be passed to $f$.
- **x_i**: integer data to be passed to $f$.
- **rtol**, **atol**, and **max_step** are optional control parameters for *relative tolerance*, *absolute tolerance*, and *max number of time steps*, respectively. Their default values have no theoretical justification.
System function

functions {
    real[] system(real time, real[] y,
                   real[] theta, real[] x_r, int[] x_i) {
        real dydt[3];
        real CL = theta[1];
        real Q = theta[2];

        /* .... */

        return dydt;
    }
}
Exercise 2: Write, fit, and diagnose the two compartment model using (a) the pmx_solve_rk45 function and (b) the pmx_solve_linode function. Do the results agree? How does performance vary?
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
Nonlinear ODEs without analytical solution

kinetics of an autocatalytic reaction [Robertson, 1966]

The structure of the reactions is

\[ A \xrightarrow{k_1} B, \quad B + B \xrightarrow{k_2} C + B, \quad B + C \xrightarrow{k_3} C + A, \]

where \( k_1, k_2, k_3 \) are the rate constants and \( A, B \) and \( C \) are the chemical species involved. The corresponding ODEs are

\[ x'_1 = -k_1 x_1 + k_3 x_2 x_3 \]
\[ x'_2 = k_1 x_1 - k_2 x_2^2 - k_3 x_2 x_3 \]
\[ x'_3 = k_2 x_2^2 \]

Given \( k_1 = 0.04, k_2 = 3.0e7, k_3 = 1.0e4 \), we make inference regarding the initial condition for \( x_1(t = 0) \).
Nonlinear ODEs without analytical solution

\[ x_1' = -k_1 x_1 + k_3 x_2 x_3 \]
\[ x_2' = k_1 x_1 - k_2 x_2^2 - k_3 x_2 x_3 \]
\[ x_3' = k_2 x_2^2 \]

Given \( k_1 = 0.04, k_2 = 3.0e7, k_3 = 1.0e4 \), we make inference regarding the initial condition for \( x_1(t = 0) \).

**Exercise 3**
Write Stan function for the above ODE’s RHS.
Stan function for autocatalytic kinetics

\[
\begin{align*}
    x_1' &= -k_1 x_1 + k_3 x_2 x_3 \\
    x_2' &= k_1 x_1 - k_2 x_2^2 - k_3 x_2 x_3 \\
    x_3' &= k_2 x_2^2
\end{align*}
\]

functions{
    real[] reaction(real t, real[] x, real[] theta, real[] r, int[] i) {
        real dxdt[3];
        real k1 = theta[1];
        real k2 = theta[2];
        real k3 = theta[3];
        dxdt[1] = -k1*x[1] + k3*x[2]*x[3];
        dxdt[3] = k2*(x[2])^2;
        return dxdt;
    }
}

What's the initial conditions for \(x_2\) and \(x_3\)?
Numerical integrators

- **Runge-Kutta 4th/5th (rk45)**
  - non-stiff equations
  - Most popular, try this if you don’t know the nature of the ODE, or what you’re doing, or both.

- **Backward differentiation formula (bdf)**
  - stiff equations
  - More expensive to use

- **Adams-Moulton**
  - non-stiff equations
  - higher-order of accuracy (do you really need it?)
  - scales better with number of steps
# Numerical integrators

<table>
<thead>
<tr>
<th>Integrators</th>
<th>Stan</th>
<th>Torsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>rk45</td>
<td>integrate_ode_rk45</td>
<td>pmx_integrate_ode_rk45</td>
</tr>
<tr>
<td>BDF</td>
<td>integrate_ode_bdf</td>
<td>pmx_integrate_ode_bdf</td>
</tr>
<tr>
<td>Adams</td>
<td>integrate_ode_adams</td>
<td>pmx_integrate_ode_adams</td>
</tr>
</tbody>
</table>

```plaintext
real[,] pmx_integrate_ode_rk45(ODE_RHS, real[] y0, real t0,
                               → real[] ts, real[] theta, real[] x_r, int[] x_i, real rtol =
                               → 1.e-6, real atol = 1.e-6, int max_step = 1e6);

- ODE_RHS: ODE right-hand-side \( f \) in \( y' = f(y, t, \theta, x_r, x_i) \).
- y0: initial condition at time t0.
- t0: initial time.
- ts: times at which we require a solution.
- theta: parameters to be passed to \( f \).
- x_r: real data to be passed to \( f \).
- x_i: integer data to be passed to \( f \).```
In each of 8 experiments performed \( x3 \) is observed.

Hierarchical model for \( x0[1] \)

```r
model {
    y0_mu ~ lognormal(log(2.0), 0.5);
    for (i in 1:nsub) {
        y0_1[i] ~ lognormal(y0_mu, 0.5);
    }
    sigma ~ cauchy(0, 0.5);
    obs ~ lognormal(log(x3), sigma);
}
```
Data

Data available for the inference

data {
  int<lower=1> nsub;  /* nb. of subjects */
  int<lower=1> len[nsub]; /* nb. of results-extraction time points for each subject */
  int<lower=1> ntot;  /* total nb. of results-extraction time points */
  real ts[ntot];      /* concatenated array for results-extraction time points */
  real obs[ntot];     /* concatenated array for observed x3 */
}
Exercise 4

Given above data and model, write the rest of Stan code.

▶ Hint: use `chem.stan` as template, also see `chem.data.R` and `chem.init.R`.

▶ Reaction begins with $A$ (on which is also what we’d like to make inference), the other two species are non-existent at the beginning of the reaction.

▶ Which numerical integrator are you using? Why?
Exercise 4

How to build & run?

Edit/Add cmdstan/make/local

TORSTEN_MPI = 1  # flag on torsten's MPI solvers
CXXFLAGS += -isystem /usr/local/include  # path to MPI
→ library's headers

Build in cmdstan

make path_to_workshop/RScript/model/chemical_reactions/chem

Run

./chem sample adapt delta=0.95 random seed=1104508041 data
→ file=chem.data.R init=chem.init.R
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
Data pooled into groups

- sport measurements are grouped by players
- people’s voting intention can be grouped by states, social status, etc.
- medical measurements can be grouped by patients, age groups, treatments, etc.
Data pooled into groups

- medical measurements are grouped by patients
  - Simulated with mrgsolve https://mrgsolve.github.io/
Hierarchical model

With a hierarchical model, we can

- do partial pooling.
- estimate how similar the groups are to one another.
- estimate individual parameters.

\[
\theta = (\theta_1, ..., \theta_L) \sim p(\theta|\theta_{\text{pop}})
\]

\[
y = (y_1, ..., y_N) \sim p(y|\theta, x)
\]
Hierarchical model

\[ \theta = (\theta_1, ..., \theta_L) \sim p(\theta|\theta_{\text{pop}}) \]

\[ y = (y_1, ..., y_N) \sim p(y|\theta, x) \]
Example 3: Hierarchical two compartment model

Likelihood function:

\[ \log \theta \sim \text{Normal}(\log \theta_{\text{pop}}, \Omega) \]

\[ \Omega = \begin{pmatrix}
\omega_1 & 0 & 0 & 0 & 0 \\
0 & \omega_2 & 0 & 0 & 0 \\
0 & 0 & \omega_3 & 0 & 0 \\
0 & 0 & 0 & \omega_4 & 0 \\
0 & 0 & 0 & 0 & \omega_5
\end{pmatrix} \]

\[ \log(c\text{Obs}) \sim \text{Normal} \left( \log \left( \frac{y_2}{VC} \right), \sigma^2 \right) \]
Exercise 5: Write, fit, and diagnose a hierarchical two compartment model for a population of 10 patients. Use data/twoCptPop.data.r and twoCptPop.r.}{

- Start by running 3 chains with 30 iterations.

- Do you get any warning messages?
Divergent transitions

Do you get any warning messages?
There were 29 divergent transitions after warmup.

A divergent transition occurs when we fail to accurately compute a Hamiltonian trajectory.

This is because we approximate trajectories.

Our sampler may not be refined enough to explore the entire typical set.
Divergent transitions

Consider the following hierarchical model:

\[ \alpha_i \sim \text{Normal}(\mu, \sigma) \]
\[ y_i \sim p(y|\alpha_i) \]
Divergent transitions

\[ \alpha_i \sim \text{Normal}(\mu, \sigma) \]

Fitting this model yields the following pairs plot:
This geometric shape is known as Neil’s funnel [Neil, 2003].
Its interactions with HMC is described in [Betancourt and Girolmi, 2015].
It occurs in hierarchical models when we have sparse data and a centered prior.
Reparameterization

Proposition
Reparameterize the model to avoid the funnel shape. We will do so by standardizing $\alpha$.

$$\alpha_{\text{std},i} := \frac{\alpha_i - \mu}{\sigma}$$

Then

$$\alpha_{\text{std}} \sim \text{Normal}(0, 1)$$
Reparameterization

Then

\[ \alpha_i = \mu + \sigma \alpha_{\text{std},i} \]

Hence

\[ y_i \sim \mathcal{P}(\mu + \sigma \alpha_{\text{std},i}) \]

▶ Same data generating process; but how does this affect the geometry of the posterior?
Reparameterization

Our model is a little more complicated than the above example:

▶ a lot of parameters (100 +)! 
▶ multiple population parameters and hierarchical structures.
▶ these parameters follow a log normal distribution (so we need a pairs plot with log $\theta$).
Reparameterization
Reparameterization

**Exercise 6**: Reparametrize the two compartment population model and fit it.

- First, work out the appropriate parametrization. You should start with \( \log \theta_i \sim \text{Normal}(\theta_{\text{pop},i}, \omega) \)
- Write, fit, and check the inference (run 100 chains).
- What kind of predictive checks can we do?
Reparameterization

Need:

- predictions at an individual level
- predictions at a population level

As always, this comes down to properly writing the data generating process in the generated quantities block.
Individual predictions
Population predictions
Further reading

For a very good case study on hierarchical models, see, Bob Carpenter’s *Pooling with Hierarchical Models for Repeated Binary Trials*

https://mc-stan.org/users/documentation/case-studies/pool-binary-trials.html
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
# ODE group integrators

## Single ODE system

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pmx_integrate_ode_rk45</code></td>
<td>ODE group integrators</td>
</tr>
<tr>
<td><code>pmx_integrate_ode_bdf</code></td>
<td>ODE group integrators</td>
</tr>
<tr>
<td><code>pmx_integrate_ode_adams</code></td>
<td>ODE group integrators</td>
</tr>
</tbody>
</table>

## ODE group

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pmx_integrate_ode_group_rk45</code></td>
<td>ODE group integrators</td>
</tr>
<tr>
<td><code>pmx_integrate_ode_group_bdf</code></td>
<td>ODE group integrators</td>
</tr>
<tr>
<td><code>pmx_integrate_ode_group_adams</code></td>
<td>ODE group integrators</td>
</tr>
</tbody>
</table>

### Single ODE system

```c
real[,] pmx_integrate_ode_xxx(
    f,
    real[] y0, real t0,
    real[] ts,
    real[] theta,
    real[] x_r, int[] x_i,
    ...);
```

### ODE group

```c
matrix pmx_integrate_ode_group_xxx(
    f,
    real[,] y0, real t0,
    int[] len, real[,] ts,
    real[,] theta,
    real[,] x_r, int[,] x_i,
    ...);
```
ODE group integrators

Single ODE system

```c
real[,] y0, real t0,
real[] ts,
real[] theta,
real[] x_r, int[] x_i,
...);
```

ODE group

```c
matrix
pmx_integrate_ode_group_xxx(
    f,
    real[,] y0, real t0,
    int[] len, real[] ts,
    real[,] theta,
    real[,] x_r, int[,] x_i,
    ...);
```

- `len` specifies the length of data for each subject within the above ragged arrays, and the size of `len` is the size of the population.
- The group integrators return a single matrix ragged column-wise. The number of rows equals to the size of ODE system.
Exercise 7

autocatalytic reaction model: ODE group version

- Change the loop with the numerical integrator to use group integrator.
- Remember the return of the group integrator is a matrix
  - nb. of rows: nb. of states
  - nb. of cols: nb. of total results-extraction time points.
Exercise 7

Build and run

▶ Edit/Add cmdstan/make/local

TORSTEN_MPI = 1  # flag on torsten's MPI solvers
CXXFLAGS += -isystem /usr/local/include  # path to MPI
  → library's headers

▶ Build in cmdstan

make ../example-models/chemical_reactions/chem_group

▶ Run

mpiexec -n 2 -l ./chem_group sample adapt delta=0.95 random
  → seed=1104508041 data file=chem.data.R init=chem.init.R
Exercise 7

- What does output say?
- How many cores can you use until performance saturates? Why?
- (optional) Can you do it using Stan’s `map_rect`? Is there a difference in style, output, and performance?
Outline

1. Course information

2. Introduction and modeling framework | Charles Margossian

3. Models in pharmacometrics | Charles Margossian

4. ODEs in Stan and Torsten | Charles Margossian

5. Numerical ODE integrators | Yi Zhang

6. Population models | Charles Margossian

7. ODE group integrators | Yi Zhang

8. PMX population solvers | Yi Zhang
PMX population solvers

<table>
<thead>
<tr>
<th>Single ODE system</th>
<th>ODE group</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmx_solve_rk45</td>
<td>pmx_solve_group_rk45</td>
</tr>
<tr>
<td>pmx_solve_bdf</td>
<td>pmx_solve_group_bdf</td>
</tr>
<tr>
<td>pmx_solve_adams</td>
<td>pmx_solve_group_adams</td>
</tr>
</tbody>
</table>

Individual solvers

```c
matrix
pmx_solve_bdf(f, int nCmt,
   real[] time, real[] amt,
   real[] rate, real[] ii,
   int[] evid, int[] cmt,
   real[] addl, int[] ss,
   real[] theta, real[]
   → biovar,
   real[] tlag, real rel_tol,
   real abs_tol, int
   → max_step);
```

Population solvers

```c
matrix
pmx_solve_group_bdf(f, int nCmt,
   int[] len, real[] time,
   real[] amt, real[] rate,
   real[] ii, int[] evid,
   int[] cmt, real[] addl,
   int[] ss, real[ , ] theta,
   real[ , ] biovar, real[ , ] tlag,
   real rel_tol, real abs_tol,
   int max_step);
```
PMX population solvers

matrix

```c
pmx_solve_group_bdf(f, int nCmt, int[] len, real[] time, real[] amt, real[] rate, real[] ii, int[] evid, int[] cmt, real[] add1, int[] ss, real[,] theta, real[,] biovar, real[,] tlag, real rel_tol, real abs_tol, int max_step);
```

Figure: arguments and output of pmx_solve_group_xxx
Time-to-event model

We analyze the time to the first grade 2+ peripheral neuropathy (PN) event in patients treated with an antibody-drug conjugate (ADC) delivering monomethyl auristatin E (MMAE). We will simulate and analyze data using a simplified version of the model reported in [Lu et al., 2017].

- Fauxlatuzumab vedotin 1.2 mg/kg IV boluses q3w × 6 does.
- 19 patients with 6 right-censored (simulated data).

Model scheme

Note

- To keep things simpler, we use the simulated individual CL and V values, and only model PD part of the problem.
- PN hazard is substantially delayed relative to PK exposure.
- Hazard increases over time to an extent not completely described by P
Likelihood

Likelihood for time to first PN \( \geq 2 \) event in the \( i^{th} \) patient:

\[
L(\theta|t_{PN,i}, censor_i, X_i) = \begin{cases} 
    h_i(t_{PN,i}|\theta, X_i) e^{-\int_0^{t_{PN,i}} h_i(u|\theta, X_i) du}, & \text{censor}_i = 0 \\
    e^{-\int_0^{t_{PN,i}} h_i(u|\theta, X_i) du}, & \text{censor}_i = 1
\end{cases}
\]

where

\( t_{PN} \equiv \text{time to first PN} \geq 2 \) or right censoring event

\( \theta \equiv \text{model parameters} \)

\( X \equiv \text{independent variables / covariates} \)

\( \text{censor} \equiv \begin{cases} 
    1, & \text{PN} \geq 2 \text{ event is right censored} \\
    0, & \text{PN} \geq 2 \text{ event is observed}
\end{cases} \)

One can see the expression

\[
e^{-\int_0^{t_{PN,i}} h_i(u|\theta, X_i) du}
\]

as the survival function at time \( t \).
Hazard of PN grade 2+ based on the Weibull distribution, with drug effect proportional to effect site concentration of MMAE:

\[
h_j(t) = \beta E_{\text{drug}j}(t)^{\beta} t^{(\beta-1)}
\]

\[
E_{\text{drug}j}(t) = \alpha c_{ej}(t)
\]

\[
c'_{ej}(t) = k_{e0} (c_j(t) - c_{ej}(t)).
\]

Overall ODE system including integration of the hazard function:

\[
x'_1 = - \frac{CL}{V} x_1 \tag{1}
\]

\[
x'_2 = k_{e0} \left( \frac{x_1}{V} - x_2 \right) \tag{2}
\]

\[
x'_3 = h(t) \tag{3}
\]

where \(x_2(t) = c_e(t)\) and \(x_3(t) = \int_0^t h(u)du\) aka cumulative hazard.
Exercise 8: write the ODE system

```plaintext
functions{
  real[] oneCptPNODE(real t, real[] x, real[] parms, real[]
         x_r, int[] x_i){
    real dxdt[3];
    real CL = parms[1];
    real V = parms[2];
    real ke0 = parms[3];
    real alpha = parms[4];
    real beta = parms[5];
    real Edrug;
    real hazard;
    /* ... */
    return dxdt;
  }
}
```
Exercise 8: the ODE system

```plaintext
real[] oneCptPNODE(real t, real[] x, real[] parms, real[] x_r,
                  int[] x_i){
    real dxdt[3];
    real CL = parms[1];
    real V = parms[2];
    real ke0 = parms[3];
    real alpha = parms[4];
    real beta = parms[5];
    real Edrug;
    real hazard;

    dxdt[1] = -(CL / V) * x[1];
    Edrug = alpha * x[2];
    if(t == 0){
        hazard = 0;
    }else{
        hazard = beta * Edrug^beta * t^(beta - 1);
    }
    dxdt[3] = hazard;
    return dxdt;
}
```
Parameters

parameters

parameters{
    real<lower = 0> ke0;
    real<lower = 0> alpha;
    real<lower = 0> beta;
}

transformed parameters{
    vector<lower = 0>[nPNObs] survObs;
    row_vector<lower = 0>[nPNObs] EdrugObs;
    vector<lower = 0>[nPNObs] hazardObs;
    vector<lower = 0>[nPNCens] survCens;
    matrix<lower = 0>[3, nt] x;
    real<lower = 0> parms[nId, 5];

    for(j in 1:nId) {
        parms[j, ] = {CL[j], V[j], ke0, alpha, beta};
    }
    /* ... */
}
Parameters

\[
\text{transformed parameters}\{
\begin{align*}
\text{vector}<\text{lower} = 0>[\text{nPN0bs}] & \quad \text{survObs}; \\
\text{row_vector}<\text{lower} = 0>[\text{nPN0bs}] & \quad \text{EdrugObs}; \\
\text{vector}<\text{lower} = 0>[\text{nPN0bs}] & \quad \text{hazardObs}; \\
\text{vector}<\text{lower} = 0>[\text{nPN0bs}] & \quad \text{survCens}; \\
\text{matrix}<\text{lower} = 0>[3, nt] & \quad x; \\
\text{real}<\text{lower} = 0> & \quad \text{parms}[\text{nId}, 5];
\end{align*}
\]

\[
\text{for}(j \in 1:\text{nId}) \{
\begin{align*}
\text{parms}[j,] & = \{\text{CL}[j], \text{V}[j], \text{ke0}, \text{alpha}, \text{beta}\}; \\
\end{align*}
\}
\]

Exercise 9

- Use \text{pmx_solve_group_rk45} to solve for \(x\).
- Write likelihood expressions for \text{survObs}, \text{EdrugObs}, \text{hazardObs}, and \text{survCens}. 

Exercise 9

- Stan’s target variable and user-defined likelihood.

```plaintext
model{
  ke0 ~ normal(0, 0.0005);
  alpha ~ normal(0, 0.000003);
  beta ~ normal(0, 1.5);

  target += log(hazardObs .* survObs);  // observed PN event log likelihood
  target += log(survCens);  // censored PN event log likelihood
}
```
Exercise 9

Edit/Add cmdstan/make/local

TORSTEN_MPI = 1  # flag on torsten's MPI solvers
CXXFLAGS += -isystem /usr/local/include  # path to MPI library's
                 headers

Build in cmdstan

make ../example-models/ttnp2/ttnp2_group

Run

mpiexec -n 4 -l ttnp2_group sample num_warmup=500
          num_samples=500 data file=ttnp2.data2.R init=ttnp2.init.R
Exercise 9

- The parallel performance is not optimal, why?
- Can you do it using Stan’s `map_rect`?
Concluding Remarks
Where does Stan fit in the pharmacometrician’s toolkit?

Bayesian modeling can be implemented using an array of softwares:

▶ probabilistic programming languages: TensorFlow probability, PyMC3, Edward
▶ pharmacometrics softwares: NONMEM, Monolix, etc.
Where does Stan fit in the pharmacometrician’s toolkit?

There is synergy between the research we do, and the development of other softwares:

▶ PyMC3, Edward, and NONMEM implement Stan’s No-U Turn Sampler.

▶ Torsten borrows many of NONMEM’s conventions
What do Stan and Torsten bring to the table?

Currently:

- a very flexible and expressive language
- algorithms that are efficient and fast for a full Bayesian inference, and that warn you when they fail.
- Diagnostic tools
- It’s free and open-source

Our goals:

- More expressive features: PDEs and SDEs
- Algorithms for fast approximate Bayesian inference (variational inference, nested Laplace).
- High performance tools: GPU, within solver parallelization.
What we covered

- Writing and fitting compartment models with Torsten
- Defining ODEs and picking a numerical solver
- Building and parameterizing a population model
- Within-chain parallelization for population models
What we didn’t cover

- Computing steady states with an algebraic solver.
- Combining multiple solvers, e.g. analytical and numerical methods
- More elaborate problems that combine all the moving parts we went through
Where can I learn more?

- The Stan book and the Torsten manual
- Bill Gillespie’s workshop: *Advanced use of Stan, Rstan, and Torsten for pharmacometric applications*
- Contributions to the Stan Conference: https://github.com/stan-dev/stancontalks
- Online tutorials: https://mc-stan.org/users/documentation/tutorials.html
- Betancourt’s case studies: https://betanalpha.github.io/writing/
Acknowledgments

Torsten development team:
▶ Bill Gillespie

Stan development team:
▶ Sebastian Weber
▶ Andrew Gelman
▶ Michael Betancourt
▶ Bob Carpenter

Institutions:
▶ Metrum Research Group
▶ Columbia University
Hamiltonian monte carlo for hierarchical models.
*Current trends in Bayesian methodology with applications*, 79.

Time-to-Event Analysis of Polatuzumab Vedotin-Induced Peripheral Neuropathy to Assist in the Comparison of Clinical Dosing Regimens.

Differential equations based models in stan.

Slice sampling.
Predicting nonlinear changes in bone mineral density over time using a multi scale systems pharmacology model.

*CCPT Pharmacometrics, Systems pharmacology.*


*Numerical analysis, an introduction, chapitre The solution of a set of reaction rate equations.*

Academic Press.


Bayesian aggregation of average data: an application in drug development.

*The Annals of applied statistics, 12.*