Population and ODE-based models using Stan and Torsten

Charles Margossian, Yi Zhang

StanCon 2019, Cambridge UK
August 2019

Outline

Outline

Instructors

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

TA

- Steve Bronder
  - Capital One

Outline

Day 1

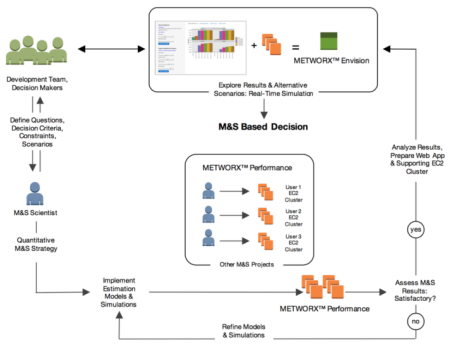
- Introduction and modeling framework
- Pharmacometrics models
- Ordinary differential equation(ODE) based models
- Numerical ODE integrators

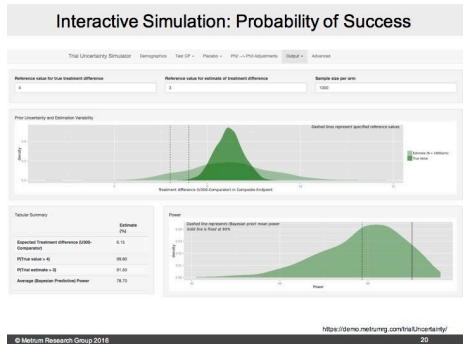
Day 2

- Population models
- Group/Population ODE integrators and MPI parallelisation
- Group/Population solvers and MPI parallelisation

Logistics

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
- pAdditional documentation

We will be using:

- Torsten v0.87
- RStan v2.19.2
- ggplot, plyr, tidyr, dplyr

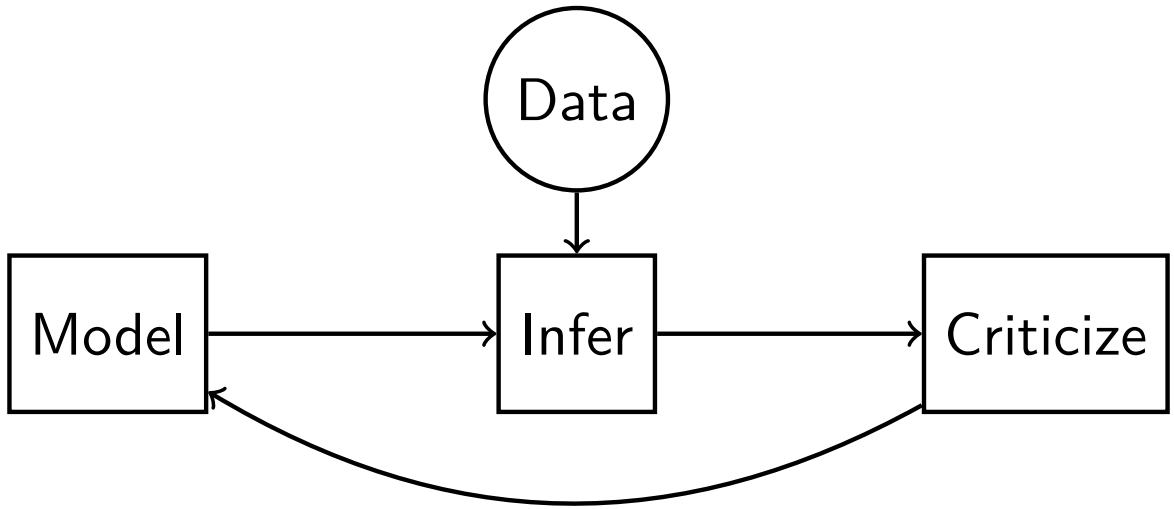
Outline

Preliminary question

- Why Bayesian in a field such as pharmacometrics?
- Example - *Bayesian aggregation of average data: an application in drug development* [**?**]

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 the NUTS to sample $\pi(\theta|y)$
- Requires users the 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:

$$\beta \sim \text{Normal}(2, 1)$$
$$\sigma^2 \sim \text{Normal}(1, 1)$$

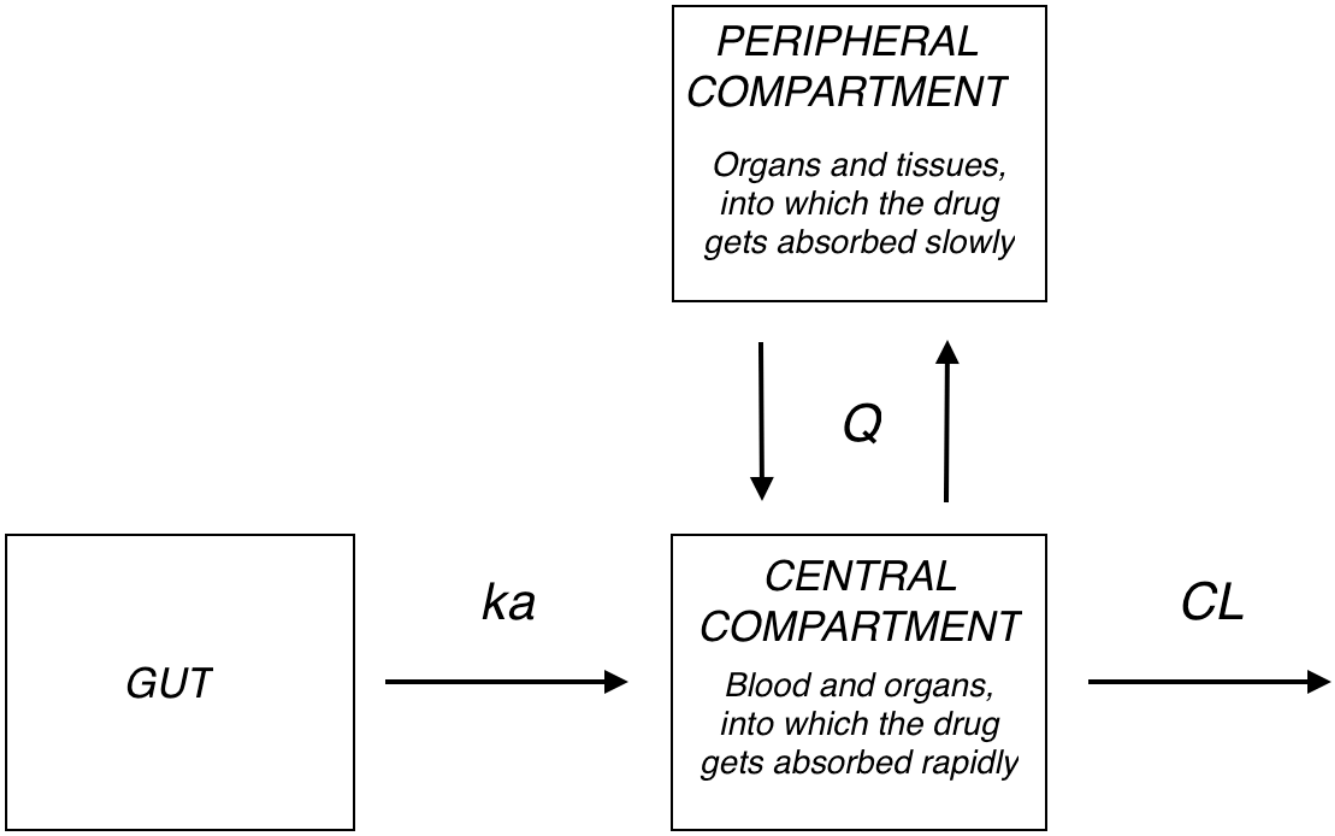
Reference

Outline

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?

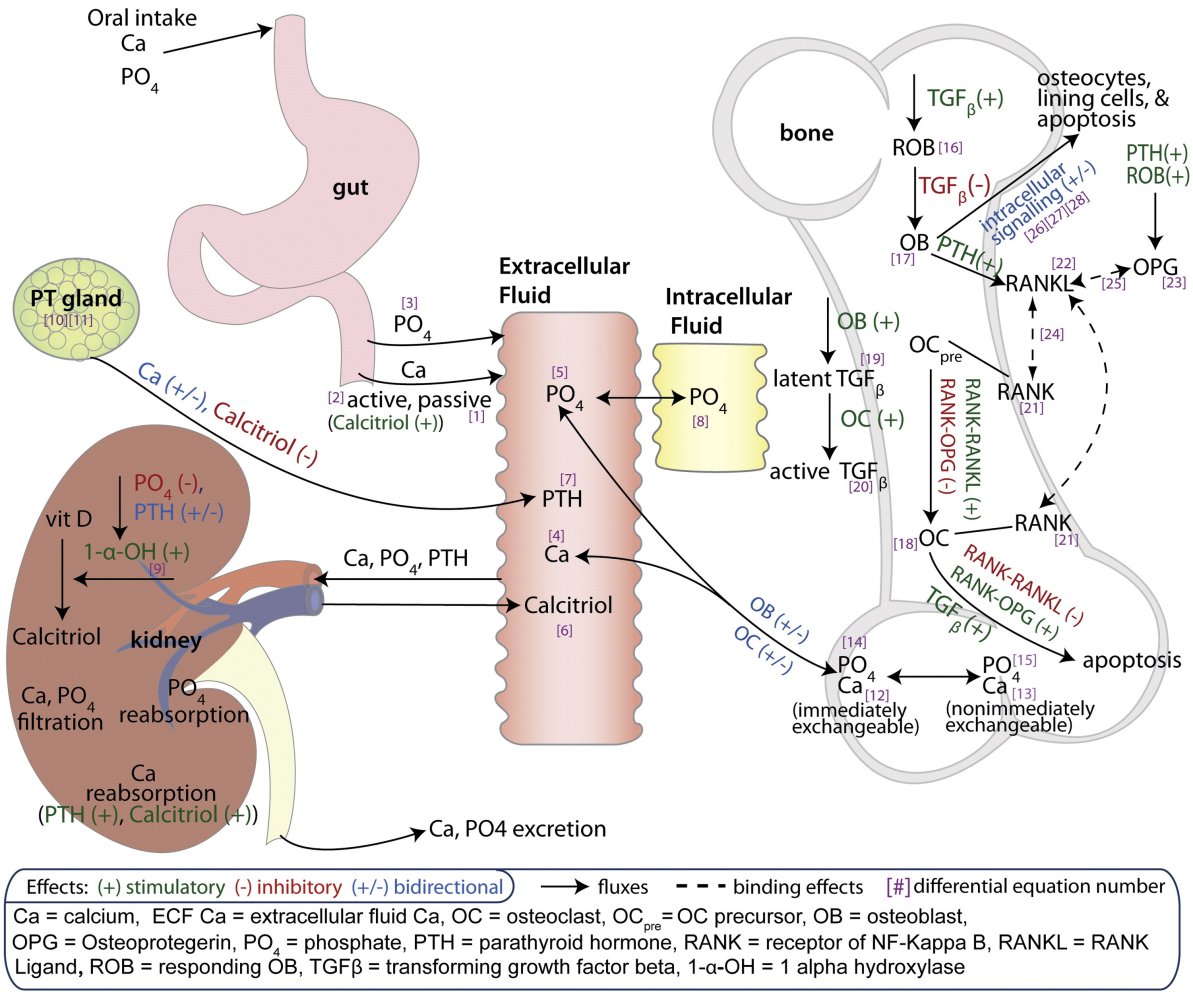
Example: Two compartment model



Two compartment model

$$
\begin{aligned}
y'_{\text{gut}} &= -k_a y_{\text{gut}} \\
y'_{\text{cent}} &= k_a y_{\text{gut}} - \left(\frac{CL}{V_{\text{cent}}} + \frac{Q}{V_{\text{cent}}} \right) y_{\text{cent}} + \frac{Q}{V_{\text{peri}}} y_{\text{peri}} \\
y'_{\text{peri}} &= \frac{Q}{V_{\text{cent}}} y_{\text{cent}} - \frac{Q}{V_{\text{peri}}} y_{\text{peri}}
\end{aligned}
$$

Example 2: Bone mineral density model from [?]



Two compartment model

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

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

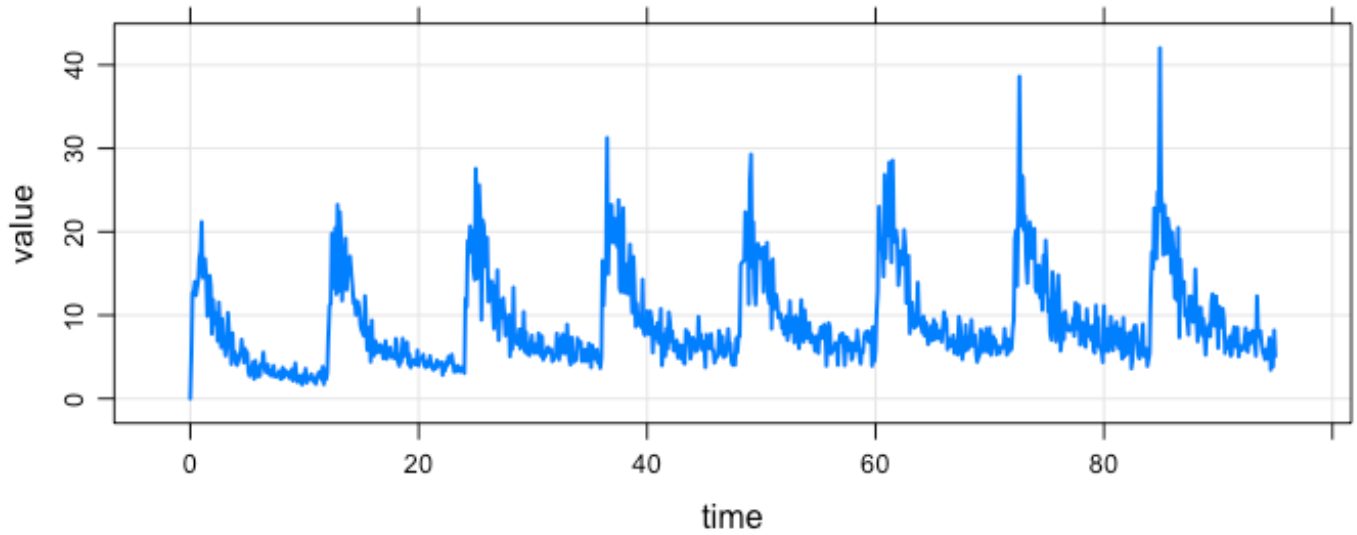
Given an initial condition $y_0 = y(t_0)$, solving the above ODE gives us the $\{$*natural evolution*$\}$ of the system at any given time point.

The event schedule

An event can be:

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

```
matrix = 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);

matrix = 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-availibility 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

```
CL ~ lognormal(log(10), 0.25);
Q ~ lognormal(log(15), 0.5);
VC ~ lognormal(log(35), 0.25);
VP ~ lognormal(log(105), 0.5);
ka ~ lognormal(log(2.5), 1);
sigma ~ cauchy(0, 1);
```

Likelihood

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

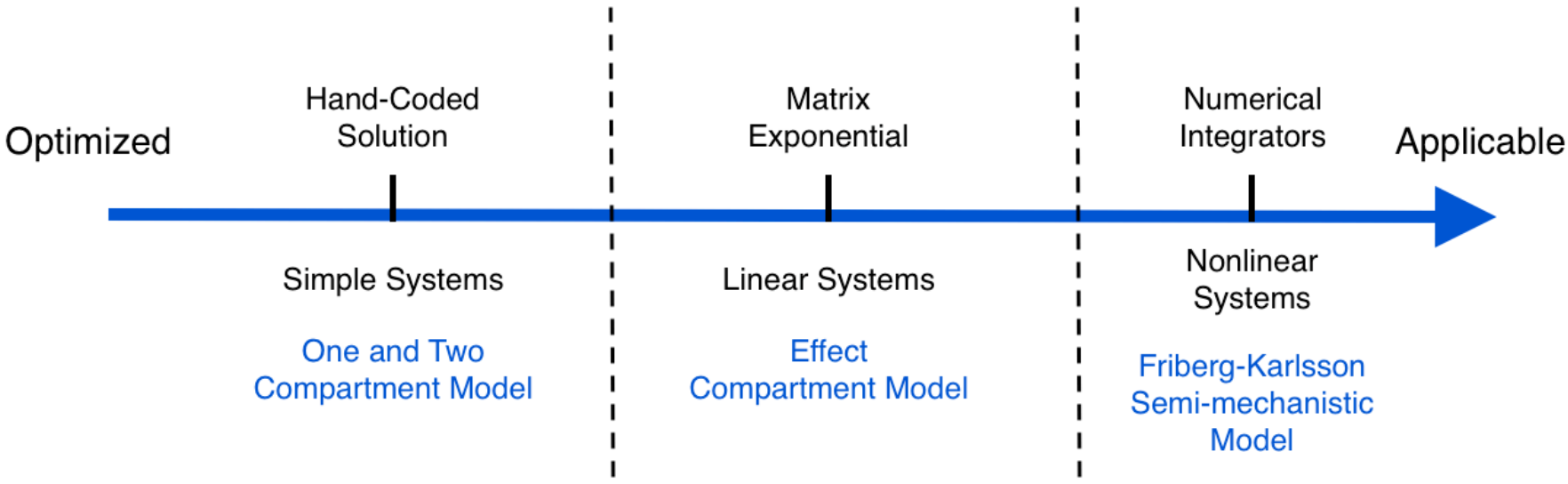
Exercise 1: write and fit this model, using `twoCptModel.r` and `model/twoCptModel.stan`. Exercise 2: Write a generated quantities block and do posterior predictive checks.

Resources

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

Outline

Arsenal of tools



For some examples, see [**?**].

- 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:

$$y'(t) = Ky(t)$$

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:

$$K = \begin{bmatrix} -ka & 0 & 0 \\ ka & -(CL+Q)/Vc & Q/Vp \\ 0 & Q/V_c & -Q/V_p \end{bmatrix}$$

Linear ODE solver in Torsten

```
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

```
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.
- `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;
  }
}
```

Torsten function

```
matrix pmx_solve_rk45(ODE_system, 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);
```

Exercise 3: Write, fit, and diagnose the two compartment model using the `pmx_solve_rk45` function.

Reference

Outline

Nonlinear ODEs without analytical solution

kinetics of an autocatalytic reaction [?]

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

$$\begin{aligned} x_1' &= -k_1 x_1 + k_3 x_2 x_3 \\ x_2' &= k_1 x_1 - k_2 x_2^2 - k_3 x_2 x_3 \\ x_3' &= k_2 x_2^2 \end{aligned}$$

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

Write Stan function for the above ODE's RHS.

Stan function for autocatalytic kinetics

$$x_1' = -k_1 x_1 + k_3 x_2 x_3$$
$$x_2' = k_1 x_1 - k_2 x_2^2 - k_3 x_2 x_3$$
$$x_3' = k_2 x_2^2$$

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

Integrators	Stan	Torsten
rk45	integrate_ode_rk45	pmx_integrate_ode_rk45
BDF	integrate_ode_bdf	pmx_integrate_ode_bdf
Adams	integrate_ode_adams	pmx_integrate_ode_adams

```
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.

Exercise

- In each of 8 experiments performed $x3$ is observed.
- Hierarchical model for $x0[1]$

```
model {
  x0_mu ~ lognormal(log(2.0), 0.5);     /* hyperparam */
  for (i in 1:nsub) {
    x0_1[i] ~ lognormal(x0_mu, 0.5);    /* loop each subject,
     ↪  sample initial condition */
  }
  sigma ~ cauchy(0, 1);                 /* observation
   ↪  uncertainty */
  obs ~ lognormal(log(x3), sigma);
}
```

Exercise

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 */
}
```

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

- Hint: see `chem.data.R` and `chem.init.R` .
- Which numerical integrator are you using? Why?

Exercise

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 ../example-models/examples/chemical_reactions/chem
```

Run

```
./chem sample adapt delta=0.95 random seed=1104508041 data
 ↪   file=chem.data.R init=chem.init.R
```

Reference

Outline

ODE group integrators

Single ODE system	ODE group
`pmx_integrate_ode_rk45`	`pmx_integrate_ode_group_rk45`
`pmx_integrate_ode_bdf`	`pmx_integrate_ode_group_bdf`
`pmx_integrate_ode_adams`	`pmx_integrate_ode_group_adams`

Single ODE system

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

ODE group

```
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

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

ODE group

```
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

autocatalytic reaction model: ODE group version

- Change the loop with the numerical integrator to use group integrator.
- 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

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

Outline

PMX population solvers

Single ODE system	ODE group
`pmx_solve_rk45`	`pmx_solve_group_rk45`
`pmx_solve_bdf`	`pmx_solve_group_bdf`
`pmx_solve_adams`	`pmx_solve_group_adams`

Individual solvers

```
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

```
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
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);
```

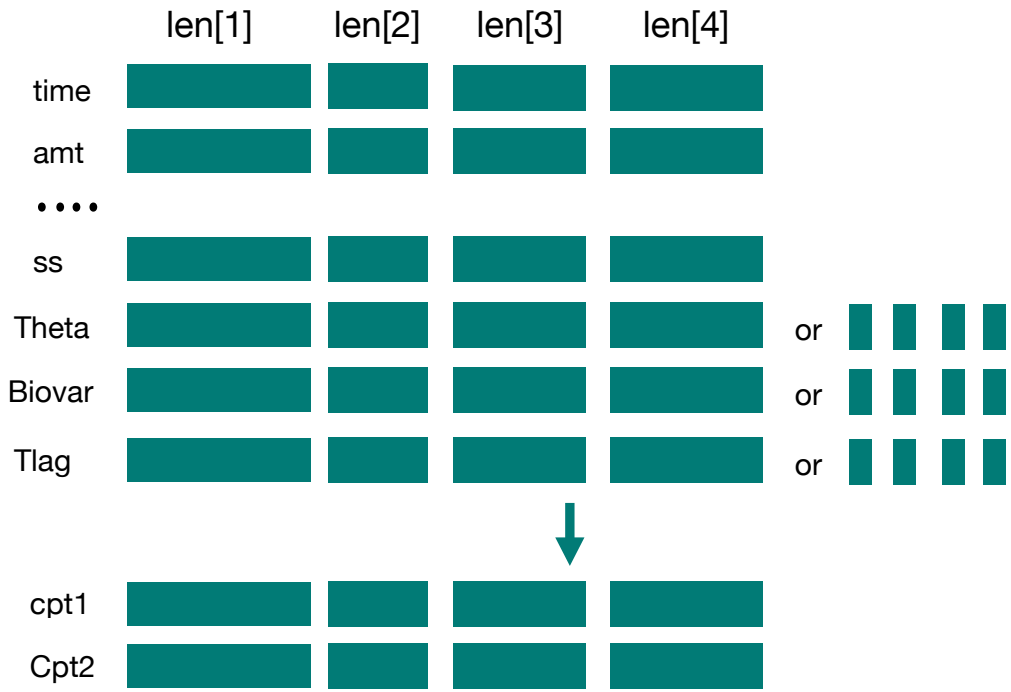


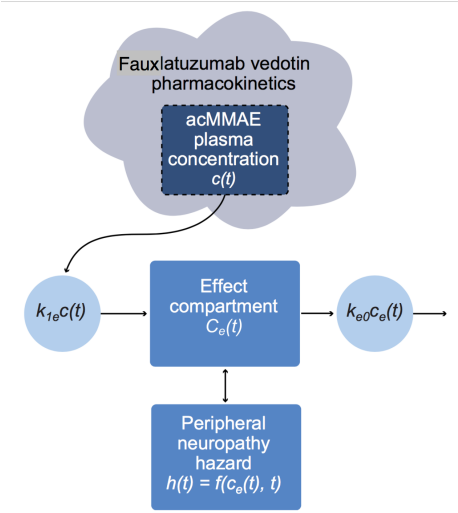
Figure: arguments and output of `pmx_solve_group_xxx`

Exercise

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 [**?**].

- Fauxlatuzumab vedotin 1.2 mg/kg IV boluses q3w $\times$ 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 PK.

Exercise

Likelihood for time to first PN ≥ 2 event in the i^{th} patient:

$$L\left(\theta | t_{\text{PN},i}, \text{censor}_i, X_i\right)$$

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

where

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

$$\theta \equiv \text{model parameters}$$

$$X \equiv \text{independent variables / covariates}$$

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

One can see the expression

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

as the survival function at time t.

Exercise

Hazard of PN grade 2+ based on the Weibull distribution, with drug effect proportional to effect site concentration of MMAE:

$$\begin{aligned} h_j(t) &= \beta E_{\text{drug}j}(t)^{\beta} t^{(\beta-1)} \\ E_{\text{drug}j}(t) &= \alpha c_{ej}(t) \\ c'_{ej}(t) &= k_{e0}\left(c_j(t) - c_{ej}(t)\right). \end{aligned}$$

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

"just walk in a minute ago, literally" mode

Apply `pmx_solve_group_rk45` function

Intermediate mode

Code `pmx_solve_group_rk45` function and its args. Use input data file `ttp2n.data2.R` as hint.

hard mode

Code ODE, `pmx_solve_group_rk45` function and its args, and the likelihood for harzard and censor event. Use input data file `ttp2n.data2.R` and `model` block as hint.

"why bother" mode

Exercise

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/ttpn2/ttpn2_group
```

Run

```
mpiexec -n 4 -l ttpn2_group sample num_warmup=500
 ↪  num_samples=500 data file=ttpn2.data2.R init=ttpn2.init.R
```

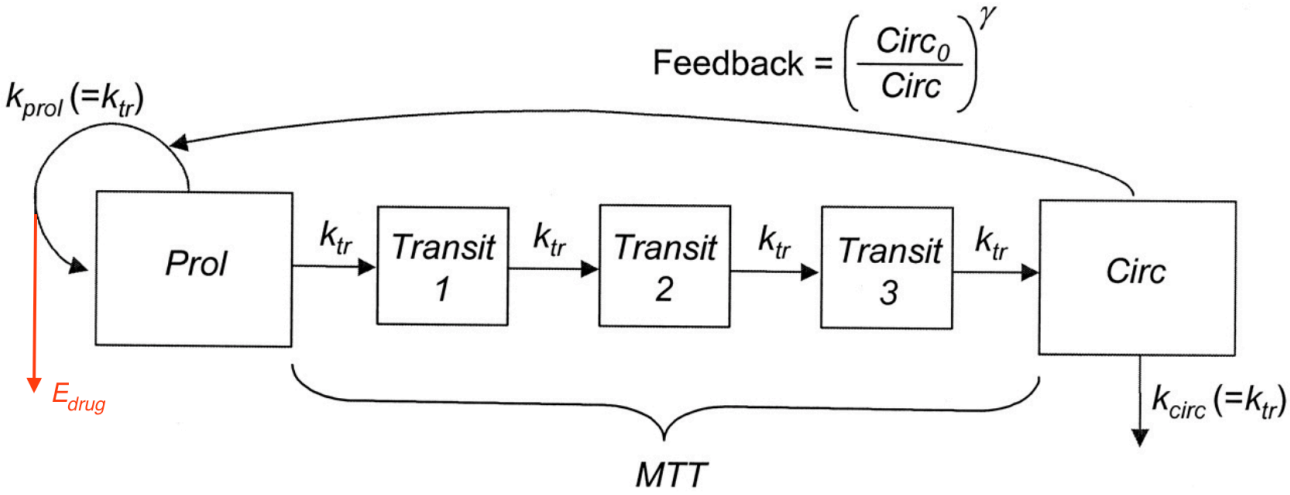
Exercise

- The parallel performance is not optimal, why?
- Can you do it using Stan's `map_rect`?

Reference

Outline

Friberg-Karlsson semi-mechanistic model [?]



ODE system of F-K model

$$
\begin{aligned}
y'_{\text{prol}} &= k_{\text{tr}} y_{\text{prol}} (1 - E_{\text{drug}}) \left(\frac{Circ_0}{y_{\text{circ}}} \right)^{\gamma} - k_{\text{tr}} y_{\text{prol}} \\
y'_{\text{tr1}} &= k_{\text{tr}} y_{\text{prol}} - k_{\text{tr}} y_{\text{tr1}} \\
y'_{\text{tr2}} &= k_{\text{tr}} y_{\text{tr1}} - k_{\text{tr}} y_{\text{tr2}} \\
y'_{\text{tr3}} &= k_{\text{tr}} y_{\text{tr2}} - k_{\text{tr}} y_{\text{tr3}} \\
y'_{\text{circ}} &= k_{\text{tr}} y_{\text{tr3}} - k_{\text{tr}} y_{\text{circ}}
\end{aligned}
$$

where $E_{\text{drug}} = \alpha \frac{y_{\text{cent}}}{V_{\text{cent}}}$, $ktr = 4/MTT$, and $\alpha \approx 3e-4$.

- y_{cent} is obtained from a two compartment model.
- Our PK/PD model therefore has a total of 8 equations.
- This problem can be solved using `pmx_solve_*`.

Coupled PK-PD system

Alternatively, we may elect to solve the PK ODEs analytically and the PD ODEs numerically.

- This can yield some speedup, in particular for problems that require ODE solutions and sensitivities (e.g [**?**]).

```
real[] pmx_solve_twocpt_rk45(reduced_ODE_system, int nOde,
 ↪   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,
 ↪   real max_step)
```

Coupled PK-PD system

```
real[] pmx_solve_twocpt_rk45(reduced_ODE_system, int nOde,
 ↪   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,
 ↪   real max_step)
```

- we now pass a "reduced system".
- we specify the number of ODEs to be solved numerically, not the number of compartments.
- `theta` now contains the parameters for the two cpt model, followed by the parameters that get passed to the numerical solver:

```
theta = {CL, Q, VC, VP, ka, /* ... */ };
```

Reduced system

```
real[] reduced_system(real time, real[] y, real[] yPK, real[]
 ↪   theta, real[] x_r, int[] x_i) {
  real[3] dydt;
  /* .... */
  return dydt;
}
```

Exercise 4 (optional): Write, fit, and diagnose a Friberg-Karlsson model with a two compartment with first order absorption PK. Use `FKModel.r` and `data/FKModel.data.r`.

Exercise

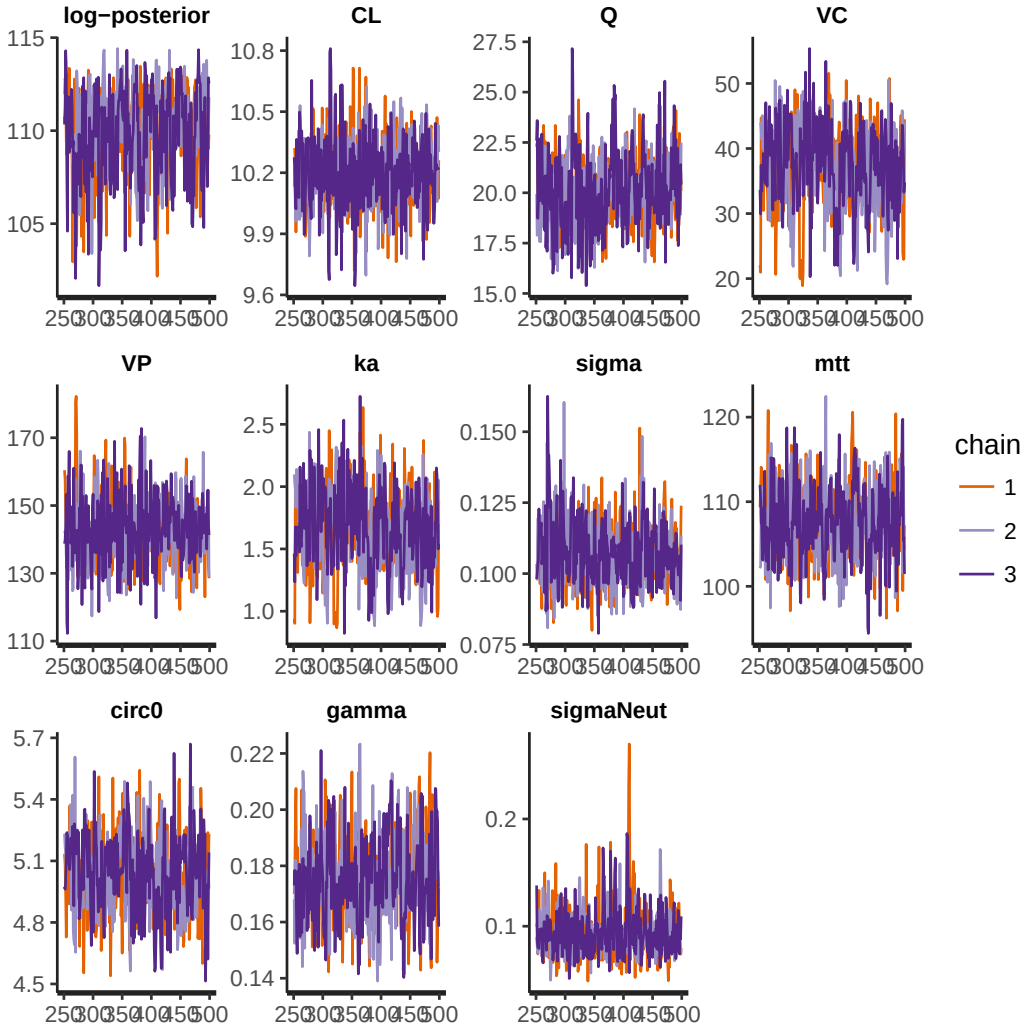
Write, fit, and diagnose a Friberg-Karlsson model with a two compartment with first order absorption PK. Use `FKModel.r` and `data/FKModel.data.r`.

- You may either use `pmx_solve_*` or `pmx_solve_twocpt_*`.
- Use $\alpha = 3e-4$ and estimate all other 8 ODE coefficients, i.e. $\theta = \{CL, Q, VC, VP, ka, MTT, circ0, \gamma\}$.
- The initial state for the neutrophil count is $Circ_0$. Either edit the event schedule to reflect this at time 0, or write the solution to your ODEs as a deviation from the baseline.

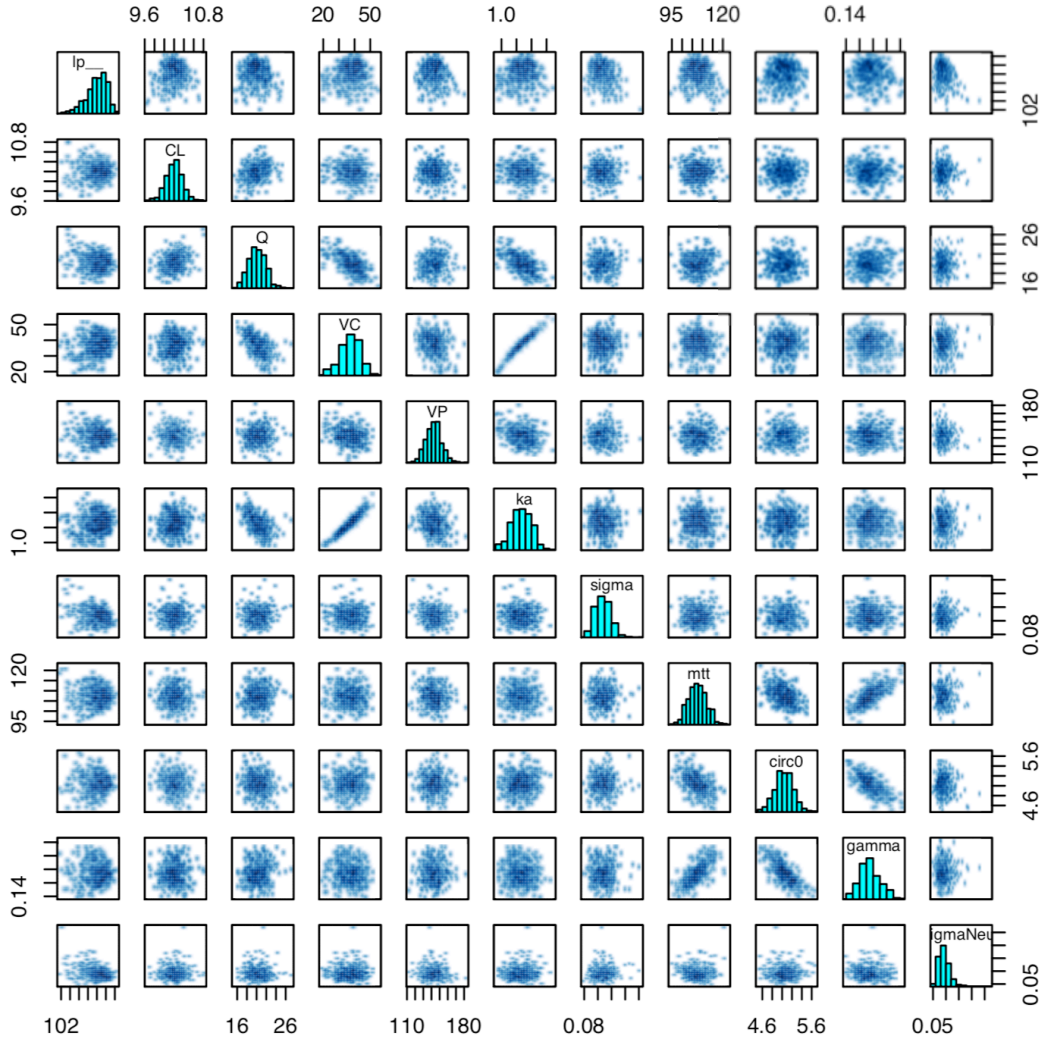
Exercise

- This exercise entails a few subtleties; in the interest of time we won't go through it in class.
- Here are however results I get from 3 chains with 500 iterations you can use as a benchmark.

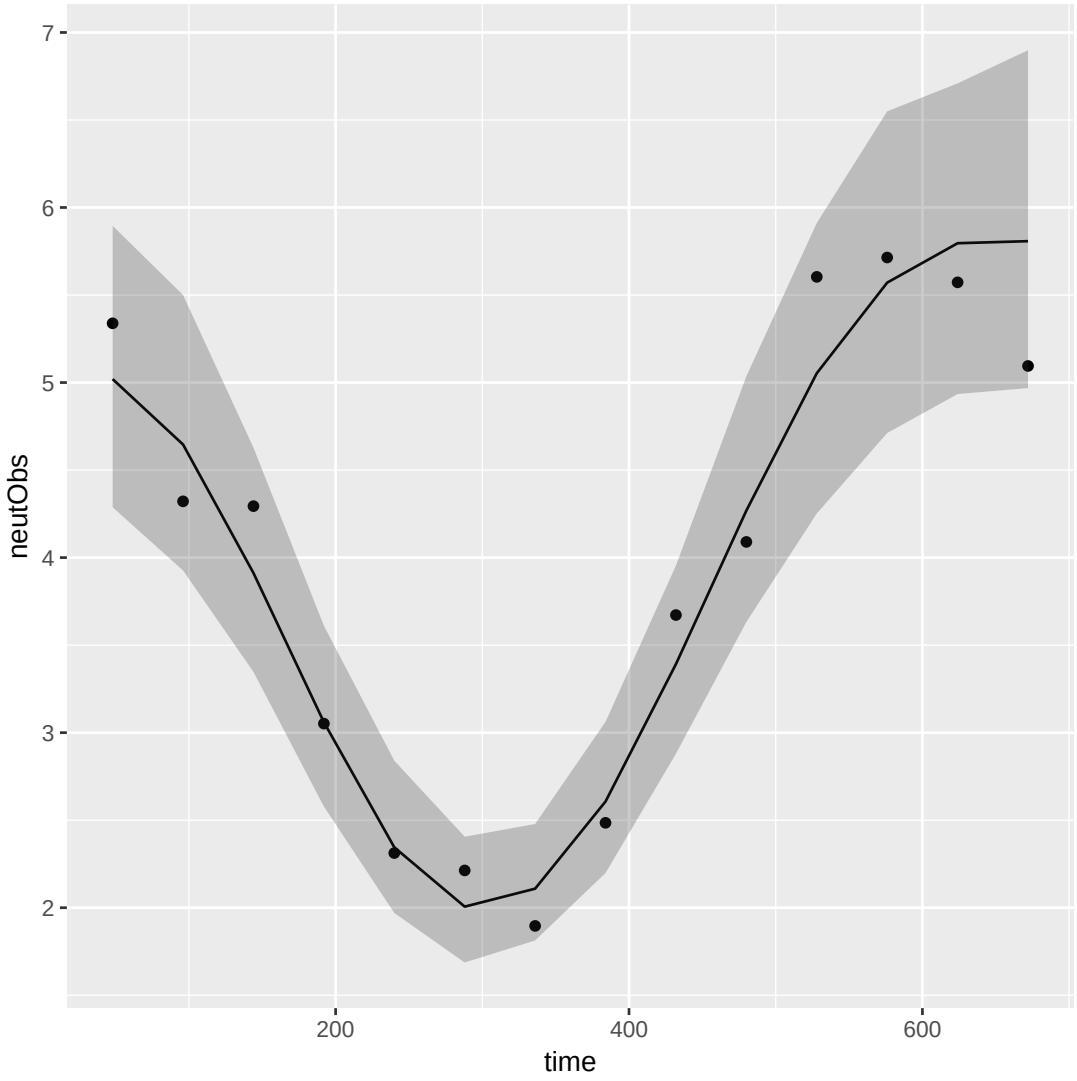
Exercise


log−posterior
CL
Q
VC
VP
ka
sigma
mtt
circ0
gamma
sigmaNeut
chain
1
2
3

Exercise


lp__
CL
Q
VC
VP
ka
sigma
mtt
circ0
gamma
gmaNeu
9.6
10.8
20
50
1.0
95
120
0.14
102
16
26
110
180
0.08
4.6
5.6
0.05

Exercise



References

Reference