Torsten: Stan functions for pharmacometric applications

Improvements and new R interface workflow

Yi Zhang and William R. Gillespie Metrum Research Group, Tariffville, CT USA

METRUM RESEARCH GROUP

Objectives

Stan is a widely used, open-source, probabilistic programming language and Bayesian inference engine [1, 2]. It provides a general model specification language and uses HMC simulation for fully Bayesian data analysis. Torsten is a library of Stan functions that simplifies implementation of pharmacometric (PMX) models and extends the range of models that may be implemented [3]. The objective of this presentation is to summarize and demonstrate recent developments of Torsten.

Methods

Implemented in C++, Torsten is licensed under BSD-3 and collected into the Stan Math submodule. Torsten is a superset of Stan with PMX functionality add–ons (supported on all major OS platforms). To use Torsten's parallel capabilities, one must have message-passing interface (MPI) installed, e.g., OpenMPI or MPICH.

Results

Torsten supports the following pharmacokinetics (PK) & pharmacodynamics (PD) models (newly added in *italic*):

One- & two-compartment model with or without first–order absorption
One-compartment PK coupled with effect-compartment model
Two-compartment PK coupled with effect-compartment model
Linear ordinary differential equation (ODE) model (matrix exponential solution)
General ODE model (numerical integration solution)
Coupled model based on analytical PK solution and numerical PD solution
ODE-based population models that permit within-chain parallel computation

Improved events handling

Torsten supports NMTRAN compatible event specification arguments (`TIME`, `AMT`, `RATE`, `II`, `EVID`, `CMT`, `ADDL`, `SS`), with all the `real[]` arguments allowed to be `parameters`. Also, ODE model parameters `theta`, bioavailability fraction `F`, and lag time `tlag` can be event/time dependent. As the most recent improvement `F` and `tlag` are optional:

```
// F = 1.0, tlag = 0, defaut ODE integrator controls
 x = pmx_solve_rk45(time, amt, rate, ii, evid, cmt, addl, ss, theta);
// F = 1.0, tlag = 0, user-defined ODE integrator controls
 x = pmx_solve_rk45(time, amt, rate, ii, evid, cmt, addl, ss, theta, rel_tol, abs_tol, max_num_steps);
// tlag = 0, user-defined F & ODE integrator controls
 x = pmx_solve_rk45(time, amt, rate, ii, evid, cmt, addl, ss, theta, F, rel_tol, abs_tol, max_num_steps);
// user-defined F, tlag & ODE integrator controls
 x = pmx_solve_rk45(time, amt, rate, ii, evid, cmt, addl, ss, theta, F, tlag, rel_tol, abs_tol, max_num_steps);
```

New ODE integration function

The Cash-Karp integrator function has been contributed to upstream Stan (`ode_ckrk`). Benchmarks indicate superior performance when it is applied to problems with near-discontinuities or rapid oscillations (Fig. 1).

Experimental feature of parallel warmup

Based on Torsten's MPI framework we have implemented an experimental cross-chain warmup algorithm that performs dynamic warmup adaptation by chain aggregation [4]: the joint log posterior sampled from parallel chains are collected to calculate $\widehat{R}$ and effective sample sizes (ESS) [5]. The warmup is terminated when $\widehat{R}$ and ESS meet preset values (Fig. 2). This avoids the trial-and-error warmup practice and improves efficiency as the number of both the warmup iterations and sampling iterations can be reduced.

Results

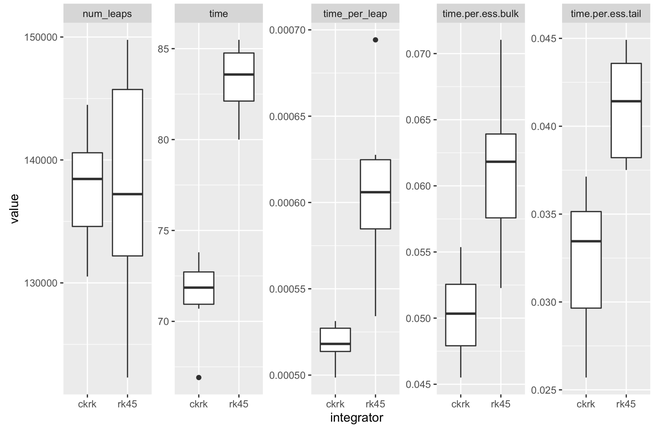

Figure 1: `ode_ckrk` vs `ode_rk45` benchmark: Van der Pol equation (Time in seconds).

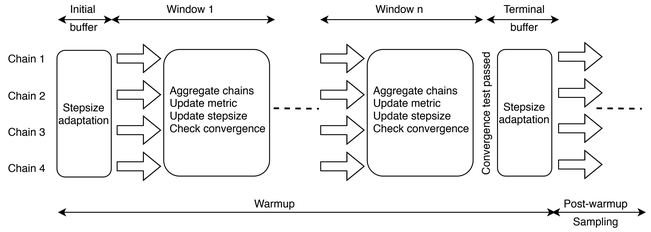

Figure 2: Cross-chain warmup algorithm.

New coupled PK–effect–compartment solvers

The latest PMX additions are PKPD solvers that analytically solve one- & two-compartment PK model coupled with an effect-compartment, so that the following two statements are equivalent:

```
x = pmx_solve_onecpt_effcpt(time, amt, rate, ii, evid, cmt, addl, ss, theta, F, tLag); // theta=[CL,V,ka,ke]
x = pmx_solve_linode(time, amt, rate, ii, evid, cmt, addl, ss, K, F, tLag); // less efficient
```

$$K = \begin{bmatrix} -k_a & 0 & 0 \\ k_a & -CL/V & 0 \\ 0 & k_e & -k_e \end{bmatrix}.$$

R workflow based on `cmdstanr`

We recommend use of the `cmdstanr` package as Torsten's R interface to simplify the installation process across platforms. The following script builds Torsten, fits an effect compartment model example, and generates posterior predictive checking (PPC) plots.

```
library(tidyverse)
library(ggplot2)
library(cmdstanr)
library(bayesplot)
library(posterior)

system('git clone https://github.com/metrumresearchgroup/Torsten.git')
# download Torsten
set_cmdstan_path("Torsten/cmdstan") # point to Torsten's cmdstan
rebuild_cmdstan()                   # build Torsten
mod <- cmdstanr::cmdstan_model("Torsten/example-models/effCpt/effCpt.stan",quiet=FALSE) # compile Torsten model

## model fitting
fit <- mod$sample(data="Torsten/example-models/effCpt/effCpt.data.R",
  ↪  init="Torsten/example-models/effCpt/effCpt.init.R")
pars <- c("CLHat", "QHat", "V1Hat", "V2Hat", "kaHat", "ke0Hat", "EC50Hat") # parameters to examine
subset.pars <- subset_draws(fit$draws(), variable=pars) # cherry-pick parameters' draws
mcmc_dens_overlay(subset.pars)      # density plot

## posterior predictive checking (PPC)
source("https://raw.githubusercontent.com/metrumresearchgroup/Torsten/master/example-models/effCpt/effCpt.data.R
  ↪  ")  # Use observation to check posterior
  ↪  predictions
cobs.pred.summary <- as_draws_df(fit$draws(variables=c("cObsPred"))) %>% summarize_all(function(x)
  ↪  {quantile(x,probs=c(0.05,0.5,0.95))}) %>% select(starts_with("cObsPred"))
pred.data <- rbind(cobs.pred.summary, unlist(mapply(rep, 1:nSubjects, (end - start + 1))), time) %>% t %>%
  ↪  as_tibble() %>% rename(lb=V1, median=V2, ub=V3, subject=V4, time=V5)
obs.data <- tibble(time=time[iObs], y=cObs, subject=unlist(mapply(rep, 1:nSubjects, (end - start + 1)))[iObs])
ppc.cobs <- function(start.id, end.id) {
    ggplot(subset(data, subject >= start.id & subject <= end.id)) +
      ↪  geom_ribbon(aes(x=time,ymin=lb,ymax=ub),fill="#b3cde0",alpha=0.8) +
      ↪  geom_line(aes(x=time,y=median),color="#005b96") + geom_point(data=subset(obs.data, subject >=start.id
      ↪  & subject <=end.id),aes(x=time,y=y),size=1.0)+
        scale_x_continuous(name="time (h)") +
        scale_y_continuous(name="plasma drug concentration (ng/mL)") +
        facet_wrap(.~subject)
}                                   # one can also use ppc functions in bayesplot package
ggsave("ppc_study_1_5mg.pdf", ppc.cobs(1,25)) # PPC for plasma concentration in study 1 of 5mg dosing

## We do PPC for PD in a similar way. See
  ↪  https://github.com/metrumresearchgroup/Torsten/blob/master/example-models/effCpt/run.R for details
```

Conclusions and future work

Torsten proves to be a valuable add–on to Stan for Bayesian PMX modeling. It also facilitates exploring experimental features and algorithms. The light-weight R interface `cmdstanr` enables Torsten to support major OS platforms and simplifies the installation process. In the near future Torsten plans to

1. improve function flexibility, possibly based on the tuple data-type support
2. explore a user module system for Stan
3. develop additional built–in functions such as indirect response models

References

[1] B. Carpenter et al. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1):1–32, January 2017.

[2] B. Carpenter et al. The Stan Math Library: Reverse-Mode Automatic Differentiation in C++. *arXiv:1509.07164 [cs]*, September 2015. arXiv: 1509.07164.

[3] Torsten: library of C++ functions that support applications of Stan in Pharmacometrics. `https://github.com/metrumresearchgroup/Torsten`.

[4] Yi Zhang and William R. Gillespie. Speed up population bayesian inference by combining cross-chain warmup and within-chain parallelization. In *the 11th American Conference on Pharmacometrics*, November 2020.

[5] A. Vehtari et al. Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC. *Bayesian Analysis*, pages 1 – 38, 2021.

yiz@metrumrg.com | billg@metrumrg.com | metrumrg.com/all-publications