# PMXStan: an R package to facilitate Bayesian PKPD modeling with Stan

Yuan Xiong, David A James, Fei He, Wenping Wang

*Novartis Pharmaceuticals Corporation, East Hanover, New Jersey, USA*

## Introduction

Using a Bayesian approach for making statistical inferences has been gaining popularity in recent years. Stan (http://mc-stan.org) is a Bayesian probabilistic programming language that implements an efficient Hamiltonian Monte Carlo method suitable for fitting larger and more complex models, and these capabilities are attracting more and more users, pharmacometricians in particular.

Currently, two hurdles have largely limited a broad application of Stan in pharmacometrics: 1) a steep learning curve for pharmacometricians to write PKPD model-specific C++-like Stan code; 2) no efficient solvers to work seamlessly with Stan's No-U-Turn Sampler (NUTS) for ordinary differential equations (ODEs) that are able to handle stiff ODE systems, often encountered in PKPD modeling.

Here we provide an R package called **PMXStan** to facilitate practical Bayesian PKPD modeling and simulation using Stan.

## PMXStan for Bayesian PKPD modeling

**PMXStan** helps pharmacometricians to focus more on PKPD model building and frees users from intimidating coding not commonly used in the pharmacometrics community. More specifically,

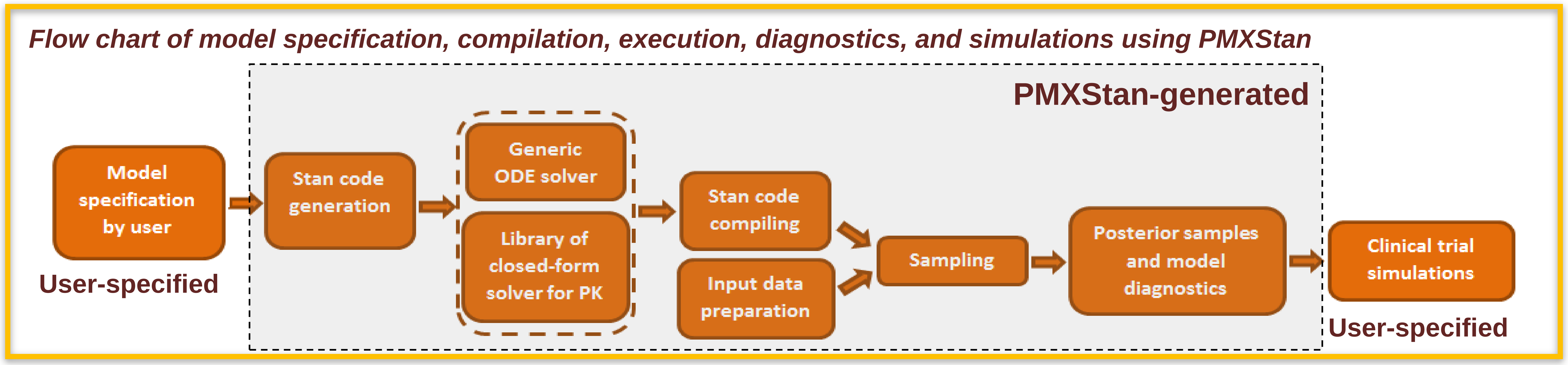
1) PMXStan automatically handles low-level technical details using a set of wrapper functions; and
2) PMXStan provides a NUTS compatible template LSODA solver to deal with stiff ODE systems.

Some advantages of using PMXStan include, but not limited to:

- With a few model specification statements defined by a user, PMXStan **generates model-specific ready-to-run Stan source code**, which is fully accessible and modifiable by the user.
- PMXStan uses data-conversion functions to **translate a conventional NONMEM dataset into a data list readable by Stan**, and provides convergence checks and model diagnostics.
- While **closed-form solutions are provided for PK models** when applicable in PMXStan, for a general PKPD model expressed as a set of ODEs, **the NUTS compatible template LSODA solver can be conveniently called**.

| Specification | Specification Variable | Options |
|---|---|---|
| ***Common for both model types*** | | |
| Model type | m.type | PK \| PKPD |
| File path for the model | m.path | *User input path* |
| Data type | d.type | individual \| **population** |
| Drug administration | m.pk.admin | 1st_order_abs \| IV_bolus \| IV_infusion |
| ***For PK models only*** | | |
| PK model structure | m.pk.struct | 1-cmpt \| **2-cmpt** \| 3-cmpt |
| PK model parameterization | m.pk.param | **CL_V** \| micro_rate |
| PK model solver | m.pk.solver | **closed_form** \| ODE |
| ***For PKPD models only*** | | |
| Index of observed state variable | m.obs.idx | *An integer* |
| Parameters to be estimated | m.theta | *Choose from parameter list* |
| Between-subject random effects | m.eta | *Choose from m.theta* |
| Parameters not to be estimated | m.const | *Input values of constant parameters* |
| Initial values of state variables | m.obs.init | *Extract from data* |

## *Flow chart of model specification, compilation, execution, diagnostics, and simulations using PMXStan*



```
m.type = startModelBuilding("PK")

# Model specification
input.specs = list(
  m.path = "pk_cls_1",
  d.type = "population",
  m.type = m.type,
  m.pk.struct = "2-cmpt",
  m.pk.admin = "IV_infusion",
  m.pk.param = "CL_V",
  m.pk.solver = "closed_form",
  m.datafile = "../datasets/poppk_ivinfus_theo.csv"
)
# Proof checking
model.specs = checkModelSpecs(input.specs)

# Generate Stan source code for the specified model
stanfilename = generateStanCode(model.specs)

# Prepare input data for Stan
dat = prepareInputData(model.specs)

# Model fitting
fit = stan(file.path(model.specs$m.path,stanfilename),
           data=dat, chains=1, iter=400)
# Print model fitting results
capture.output(print(fit,digits = 3, probs = c(0.025, 0.5, 0.975)),
               file = file.path(model.specs$m.path, "summary.txt"))
# Save model specifications and fitting
save(model.specs, stanfilename, dat, fit,
     file = file.path(model.specs$m.path, "model.info.RData"))

# Trace plots for parameters
pdf(file.path(model.specs$m.path,"trace.pdf"))
plotTraces(fit, model.specs)
dev.off()
# Goodness of fit
pdf(file.path(model.specs$m.path,"gof.pdf"))
plotGoF(fit, dat, model.specs)
dev.off()
```

## Model building, fitting, and diagnostics process for a population PK model

**Model specification by user**

A 2-compartment population PK model with IV infusion, parameterized by clearance-volume, and solved by closed form solution

**Automated modeling process**

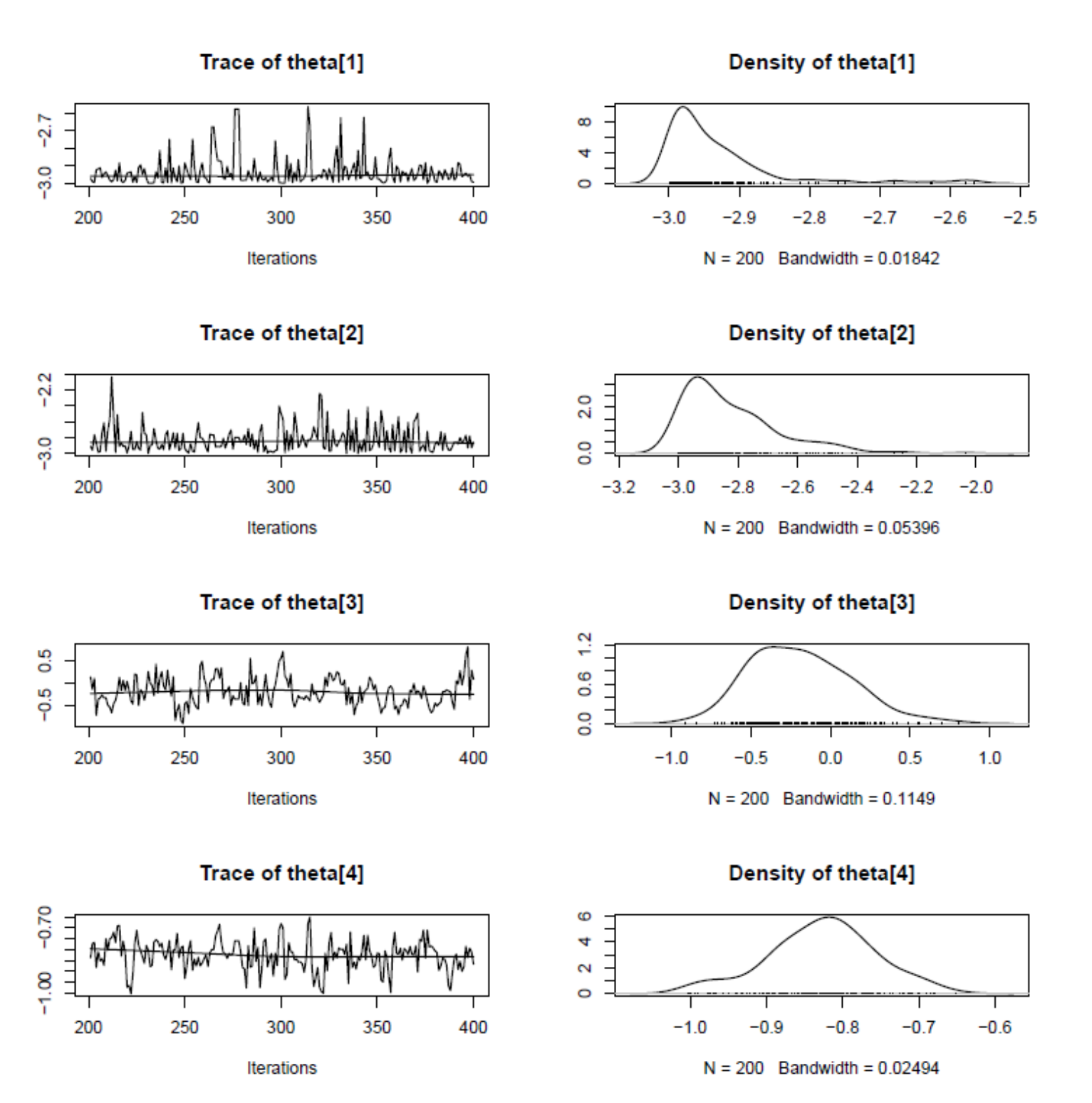
- Stan code generation
- Data preparation
- Invoke Stan for model compiling and sampling

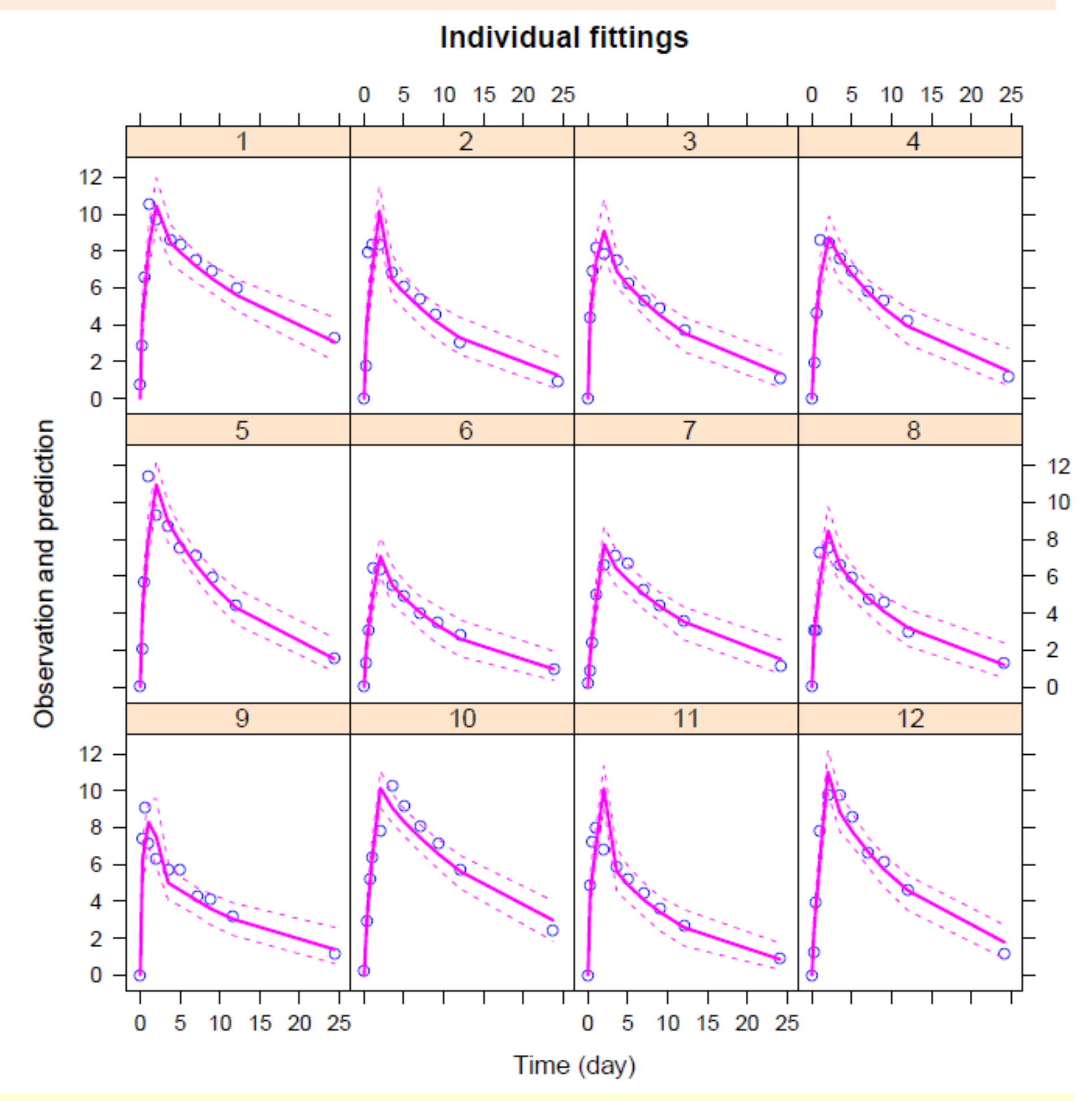
**Post processing**

- Trace plots to check convergence
- Goodness-of-fit plots for model diagnostics

```
data{
    int<lower=0> NSUB;
    int<lower=0> NOBS[NSUB];
    int<lower=0> NDOSE[NSUB];
    vector[sum(NOBS)] conc;
    ...
}
parameters{
    vector<lower=-5.0, upper=5.0>[4] theta;
    vector[4] eta[NSUB];
    ...
}
transformed parameters{
    ...
    {   ...
        for(i in 1:NSUB){
            ...
            g <- linear_cmpt_iv_infusion(...);
            ...
        }
    }
}
model{
    for(k in 1:4){
        for(i in 1:NSUB)
            eta[i,k] ~ normal(0.,1.);
        theta[k] ~ normal(0.,1000.);
        ...
    }
    ...
    conc ~ normal(y_pred, sigma);
}
```

```
> dat
$NSUB
[1] 12
$NOBS
[1] 11 11 11 11 11 11 11 11 11 11 11 11

$obs_time
[1]  0.00  0.25  0.57  1.12  2.02  3.82
[7]  5.10  7.03  9.05 12.12 24.37  0.00
[13] 0.27  0.52  1.00  1.92  3.50  5.02
[19] 7.03  9.00 12.00 24.30  0.00  0.27
...

$conc
[1]  0.74  2.84  6.57 10.50  9.66  8.58
[7]  8.36  7.47  6.89  5.94  3.28  0.00
[13] 1.72  7.91  8.31  8.33  6.85  6.08
[19] 5.40  4.55  3.01  0.90  0.00  4.40
 ...

$NDOSE
[1] 1 1 1 1 1 1 1 1 1 1 1 1

$dose_amt
[1] 4.02 4.40 4.53 4.40 5.86 4.00 4.95 4.53
[9] 3.10 5.50 4.92 5.30

$dose_time
 [1] 0 0 0 0 0 0 0 0 0 0 0 0

$inf_time
 [1] 2 2 2 2 2 2 2 2 2 2 2 2
```





## Generic PKPD models in ODE form

- User provides a set of ODEs
- A customized solver ("ODE extension") is generated for the input ODE system
- System parameters are recognized and output for the convenience of model specification by users

```
> ode <- "
     C2 = centr/V;
     d/dt(depot) =-ka*depot;
     d/dt(centr) = ka*depot - ke*centr;
     d/dt(eff) = (1+Emax*C2/(C2+EC50))*Kin -
Kout*eff;
 "
> instant.stan.extension(ode)
A new ODE extension for Stan has been
created.
System parameters are: V ka ke Emax EC50 Kin
Kout
```

**Main features**

- Written in C++ and highly efficient
- Handling complex dosing events of various routes and schedules
- Capacity to fit multiple endpoints simultaneously

## Acknowledgements

We would like to thank the Stan team and colleagues Amy Racine, Sebastian Weber, Etienne Pigeolet, and Michael Looby for their support and discussions.