

# Nonlinear Mixed Effects Modelling with Pumas.jl

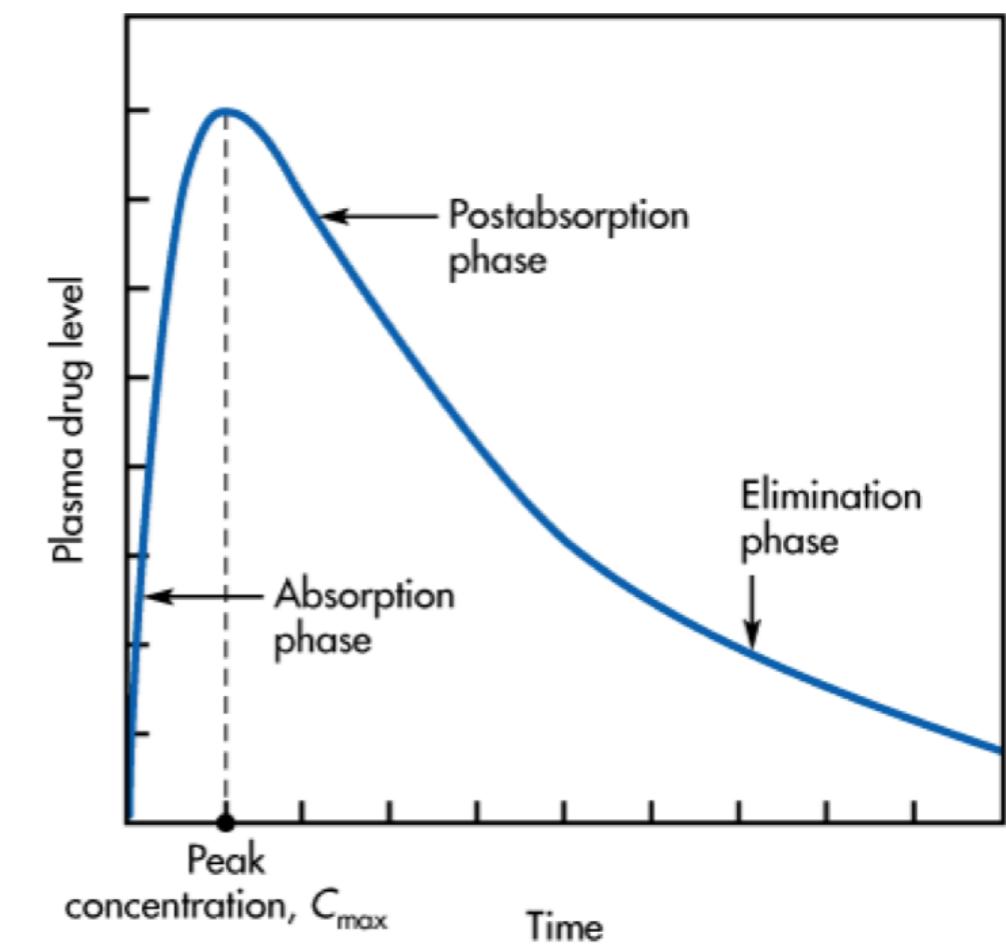
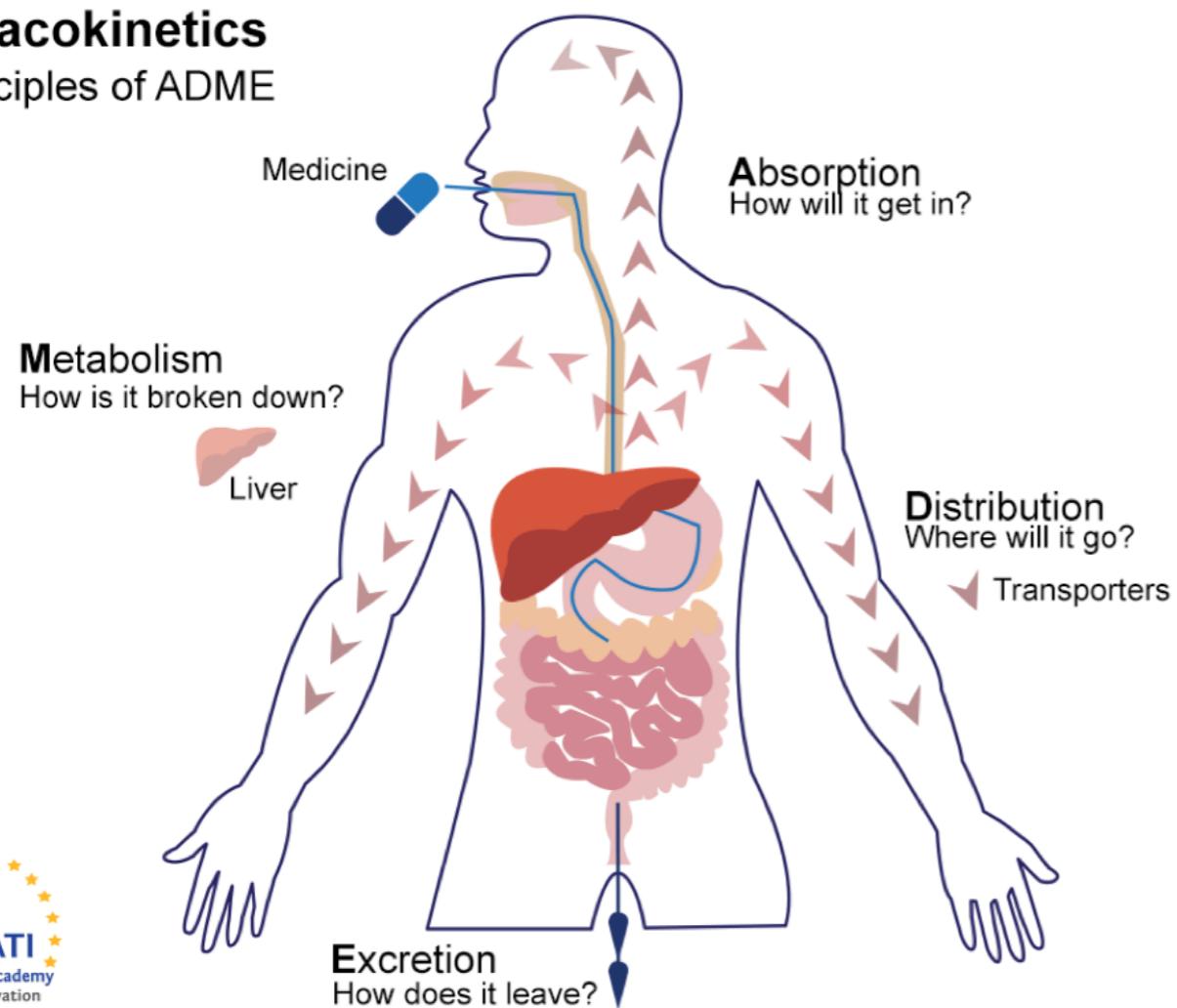
Vaibhav Kumar Dixit



@Vaibhavdixit02

# Pharmacokinetics

The principles of ADME



Source: Shargel L, Wu-Pong S, Yu ABC: *Applied Biopharmaceutics & Pharmacokinetics*  
6th Edition: [www.accesspharmacy.com](http://www.accesspharmacy.com)

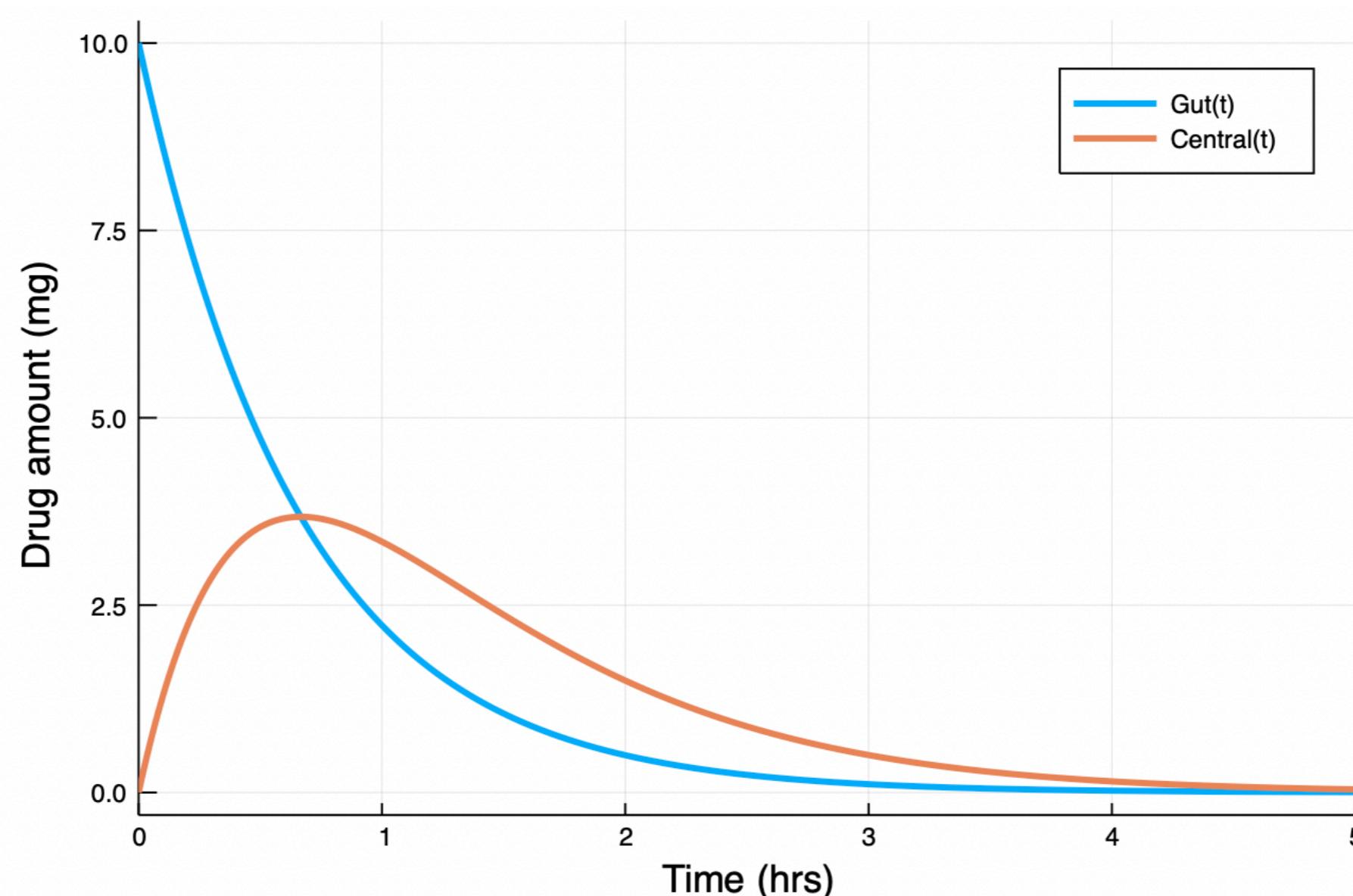
Copyright © The McGraw-Hill Companies, Inc. All rights reserved.

<https://blog.biogeniq.ca/hubfs/figure1-article-alexe2-EN.png>

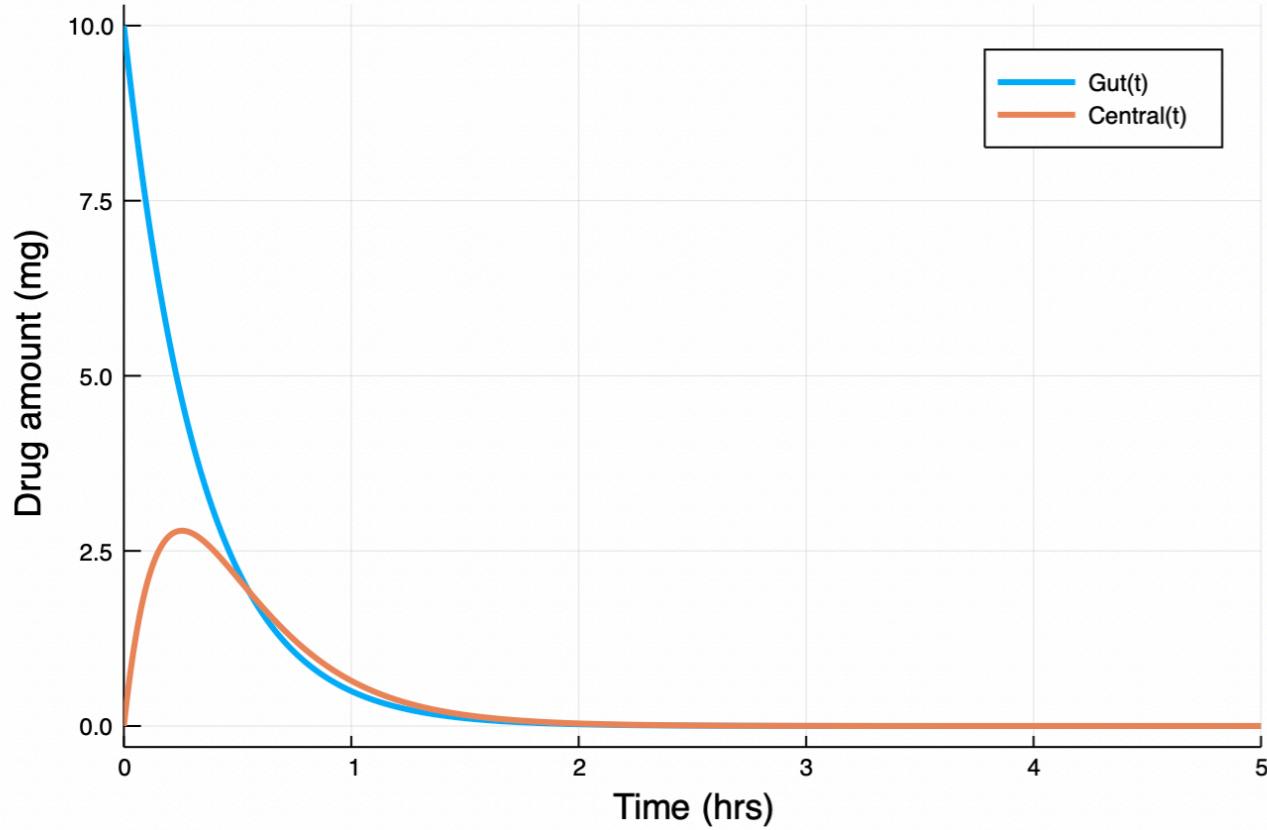
# Modelling the drug amount in the body

$$\frac{dGut}{dt} = -Ka \times Gut$$

$$\frac{dCentral}{dt} = Ka \times Gut - \frac{CL}{V} \times Central$$

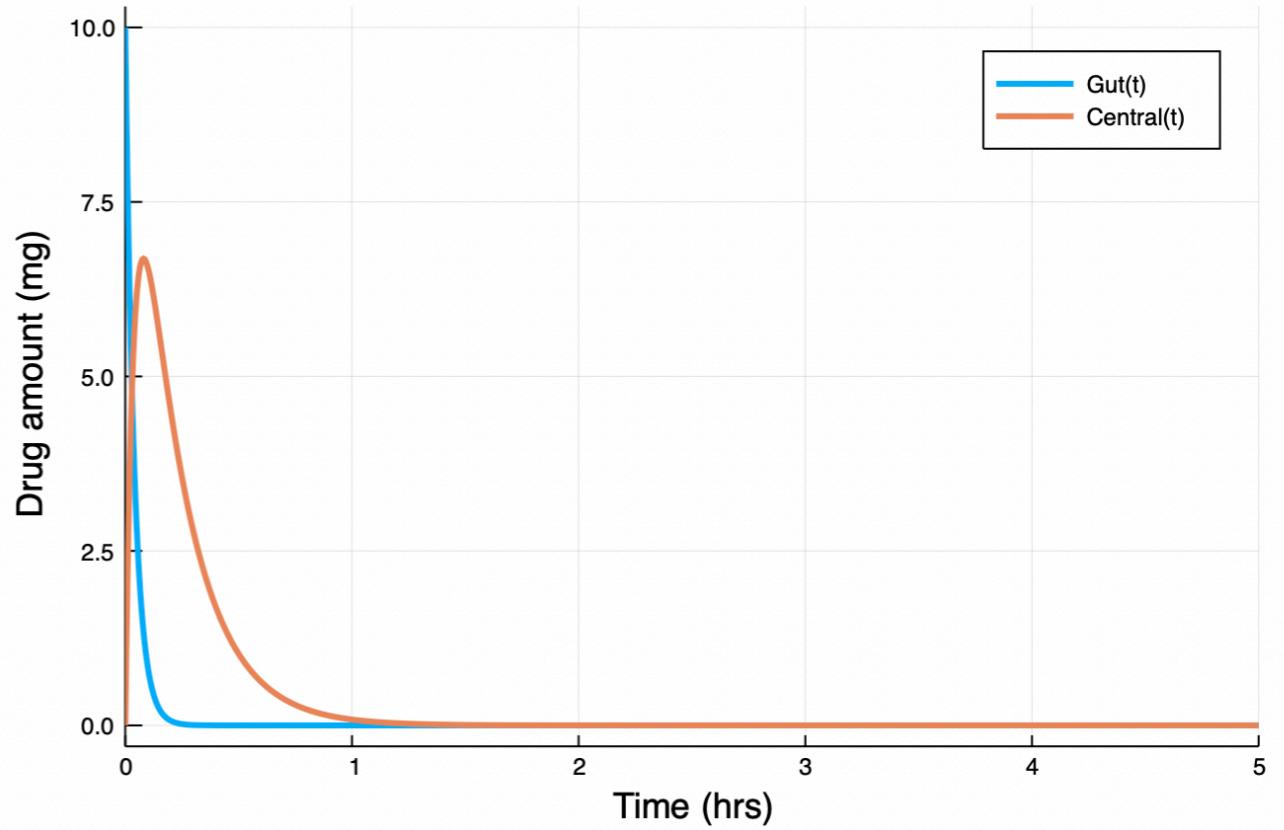


# But the parameters?



$$K_a = 3$$

$$CL/V = 5$$



$$K_a = 25$$

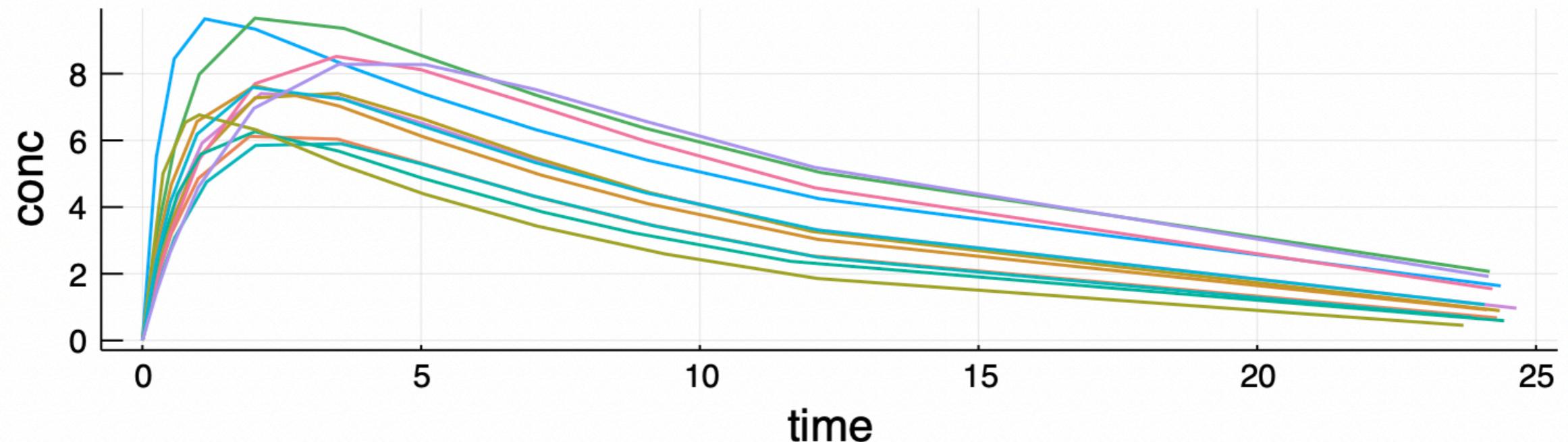
$$CL/V = 5$$

# Moving from an individual to a population

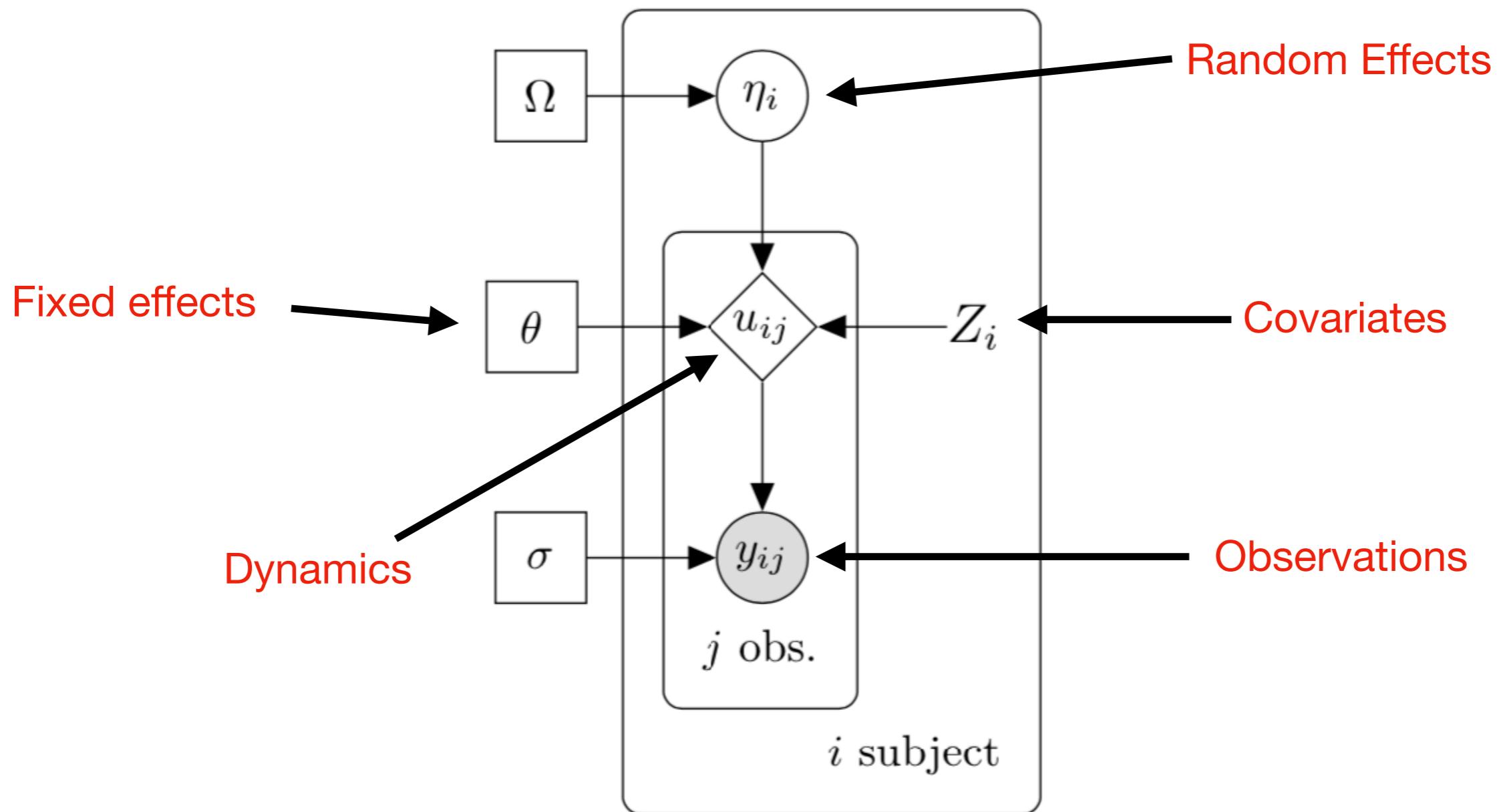
So the same dose, same model, but different concentration profiles for different individuals because of the variation in the parameters that define a subject.

How do we capture this variability?

Population Simulation



# Non Linear Mixed Effects Models



# Defining NLME Models in Pumas

```
1 nlme_model = @model begin
2   @param begin
3     θ ∈ VectorDomain(3,init=[3.24467E+01, 8.72879E-02, 1.49072E+00])
4     Ω ∈ PSDDomain(init=Matrix{Float64}([ 1.93973E-02  1.20854E-02  5.69131E-02
5                                         1.20854E-02  2.02375E-02 -6.47803E-03
6                                         5.69131E-02 -6.47803E-03  4.34671E-01 ]))
7     Σ ∈ PDiagDomain(init=[1.70385E-02, 8.28498E-02])
8   end
9
10  @random begin
11    η ~ MvNormal(Ω)
12  end
13
14  @pre begin
15    V = θ[1] * exp(η[1])
16    Ke = θ[2] * exp(η[2])
17    Ka = θ[3] * exp(η[3])
18    CL = Ke * V
19  end
20
21  @vars begin
22    conc = Central / V
23  end
24
25  @dynamics begin
26    Gut' = -Ka*Gut
27    Central' = Ka*Gut - Ke*Central
28  end
29
30  @derived begin
31    dv ~ @. Normal(conc,sqrt(conc^2 *Σ.diag[1] + Σ.diag[end])+eps())
32  end
33 end
```

## @param: Model parameters (Population averages)

```
1 @param begin
2   θ1 ∈ RealDomain(lower=0.1, init=2.77, upper=5.0)
3   θ2 ∈ RealDomain(lower=0.008, init=0.0781, upper=0.5)
4   θ3 ∈ RealDomain(lower=0.004, init=0.0363, upper=0.9)
5   Ω ∈ PSDDomain(2, init=diagm([5.55, 0.515]))
6   σ2 ∈ RealDomain(lower=0.0, init=0.388)
7 end
```

This specifies the parameters of the model, and their domains:

- `RealDomain`: real values (possibly subject to upper/lower bounds)
- `VectorDomain`: vector of real values
- `PSDDomain`: symmetric positive-definite matrices (e.g. for covariance matrices)
- `PDiagDomain`: positive-valued diagonal matrices

## @random: Random effects (Individual differences)

```
1 @random begin
2   η ~ MvNormal(Ω)
3 end
```

Specifies the random effects, dependent on parameters. The random effects are defined by a distribution from `Distributions.jl`.

## @pre: pre-processing

```
1 @pre begin
2   Ka = θ1*exp(η[1])
3   K  = θ2
4   CL = θ3*wt*exp(η[2])
5   V  = CL/K
6   SC = V/wt
7 end
```

This determines how the model parameters, random effects and covariates are combined before the differential equations solver.

## @init: Initial values (optional)

```
1 @init begin
2   Depot  = 0.0
3   Central = 0.0
4 end
```

Specifies the starting values of the differential equations at the time of first dose.

## @covariates: Covariates

```
1 @covariates sex wt
```

Specifies the covariate terms from the dataset.

## @dynamics: ODE specification

```
1 @dynamics begin
2     dDepot    = -Ka*Depot
3     dCentral =  Ka*Depot - K*Central
4 end
```

Specifies the system of differential equations. Differential variables are declared by having a line defining their derivative.

Also provide special cases for known closed-form solutions, e.g.

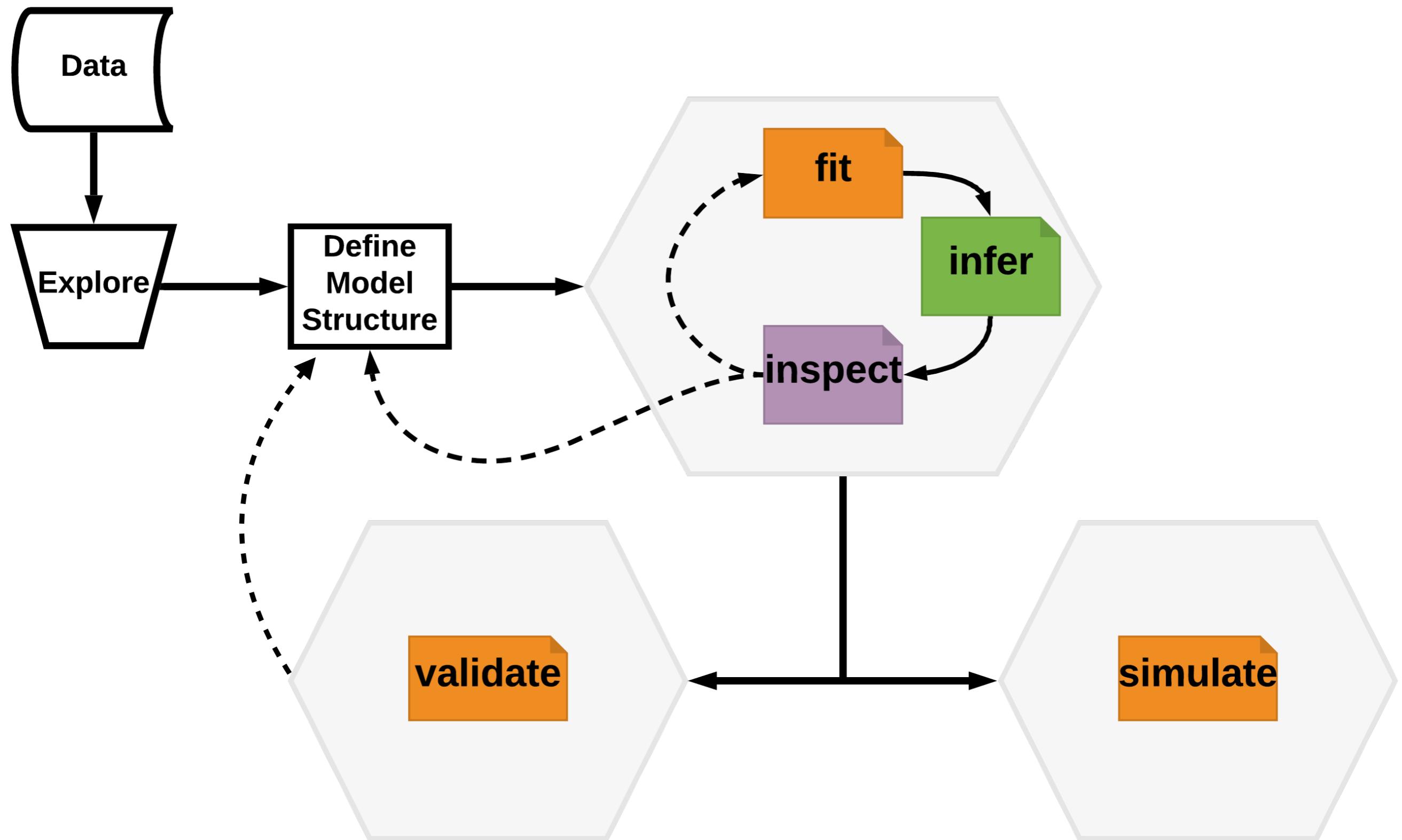
```
1 @dynamics OneCompartmentModel
```

## @derived: post-processing

```
1 @derived begin
2     dv ~ @. Normal(conc,sqrt(conc^2 *Σ.diag[1] + Σ.diag[end])+eps( ))
3 end
```

The derived block specifies variables that can be used for post-processing of results. It can be an continuous distribution shown above or a discrete one like

```
1 @derived begin
2     dv ~ @. Poisson(baseline*(1-dose/(dose + d50)))
3 end
```



# Define the model

```
model = @model begin
    @param begin
        tvcl ∈ RealDomain(lower=0, init = 4.0)
        tvv ∈ RealDomain(lower=0, init = 70)
        pmoncl ∈ RealDomain(lower = -0.99, init= -0.7)
        Ω ∈ PDiagDomain(init=[0.09,0.09])
        σ_prop ∈ RealDomain(lower=0,init=0.04)
    end

    @random begin
        η ~ MvNormal(Ω)
    end

    @pre begin
        CL = tvcl * (1 + pmoncl*isPM) * (wt/70)^0.75 * exp(η[1])
        V = tvv * (wt/70) * exp(η[2])
    end
    @covariates wt isPM

    @dynamics ImmediateAbsorptionModel
    #@dynamics begin
    #    Central' = - (CL/V)*Central
    #end

    @derived begin
        cp = @. 1000*(Central / V)
        dv ~ @. Normal(cp, sqrt(cp^2*σ_prop))
    end
end
```

Let's suppose that there was clinical trial for a new exciting drug JuliaCon, it relieves you of headaches due to using other languages.

We observed the concentrations but we did not see the response we wanted. We think that the dosage we were giving, say 100mg was not sufficient, we need to ramp it up.

The question arises, what if I increase the dose to say, 200mg, does that fix the problem? What about 150mg would that work?

To answer this question we would have to run the entire clinical trial again with these different doses.

**That is slow and costs a lot!**

So how do we answer this question?



fit

## Read the data

```
julia> data = read_pumas(simdf,cvs=[:isPM, :wt])  
Population  
Subjects: 10  
Covariates: isPM, wt  
Observables: dv
```

Use this data to estimate the parameters of our model.

```
julia> res = fit(model,data,param,Pumas.FOCEI())  
FittedPumasModel  
  
Successful minimization: true  
  
Likelihood approximation: Pumas.FOCEI  
Objective function value: 8084.54  
Total number of observation records: 1210  
Number of active observation records: 1210  
Number of subjects: 10  
  
-----  
Estimate  
-----  
tvcl 4.8809  
tvv 89.739  
pmoncl -0.73558  
 $\Omega_{1,1}$  0.10822  
 $\Omega_{2,2}$  0.051508  
 $\sigma_{\text{prop}}$  0.042149  
-----
```



```
julia> infer(res)
Calculating: variance-covariance matrix
. Done.
FittedPumasModelInference

Successful minimization:           true

Likelihood approximation:          Pumas.FOCEI
Objective function value:          8084.54
Total number of observation records: 1210
Number of active observation records: 1210
Number of subjects:                10

-----
             Estimate        RSE      95.0% C.I.
-----
tvcl     4.8809    12.932  [ 3.6438 ; 6.1181 ]
tvv      89.739     7.3011 [ 76.898 ; 102.58 ]
pmoncl -0.73558   -7.9587 [ -0.85032 ; -0.62084 ]
Ω₁,₁     0.10822    26.546  [ 0.051913 ; 0.16452 ]
Ω₂,₂     0.051508   41.275  [ 0.0098391 ; 0.093176 ]
σ_prop   0.042149   3.3957  [ 0.039344 ; 0.044954 ]
```



Predictions: The difference between the observations and the model expectation

- Population - EPRED, PRED, CPRED, CPREDI
- Individual - IPRED, CIPRED, CIPREDI, EIPRED

Residuals:

- Population - NPDE, WRES, CWRES, CWRESI
- Individual - IWRES, ICWRES, ICWRESI, EIRES

$\eta$ shrinkage,  $\epsilon$ shrinkage, AIC, BIC

```
julia> resout = DataFrame(inspect(res))
julia> first(resout, 6)
```

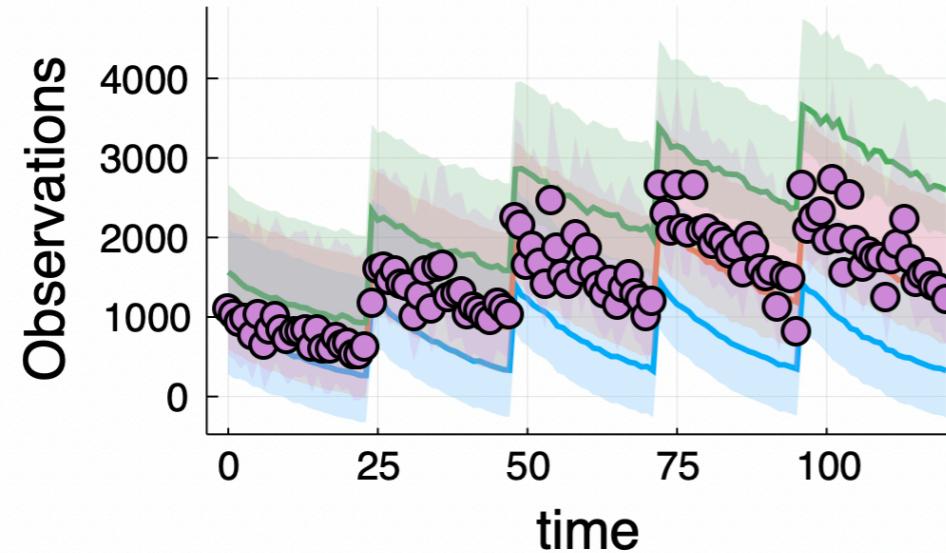
6×13 DataFrame

Row	id	time	isPM	wt	pred	ipred	pred_approx	wres	iwres	wres_approx	ebe_1	ebe_2	ebes_approx
	String	Float64	Int64	Int64	Float64	Pumas.FOCEI	Float64	Float64	Float64	Pumas.FOCEI	Float64	Float64	Pumas.FOCEI
1	1	0.0	0	68	1326.95	1290.5	FOCEI()	0.0141867	0.164838	FOCEI()	-0.273173	0.0282462	FOCEI()
2	1	1.0	0	68	1255.42	1236.44	FOCEI()	0.247655	0.414528	FOCEI()	-0.273173	0.0282462	FOCEI()
3	1	2.0	0	68	1187.56	1184.65	FOCEI()	-1.44113	-1.53356	FOCEI()	-0.273173	0.0282462	FOCEI()
4	1	3.0	0	68	1123.17	1135.03	FOCEI()	-0.66784	-1.10145	FOCEI()	-0.273173	0.0282462	FOCEI()
5	1	4.0	0	68	1062.1	1087.49	FOCEI()	-0.67988	-1.29264	FOCEI()	-0.273173	0.0282462	FOCEI()
6	1	5.0	0	68	1004.17	1041.94	FOCEI()	1.14917	0.521982	FOCEI()	-0.273173	0.0282462	FOCEI()

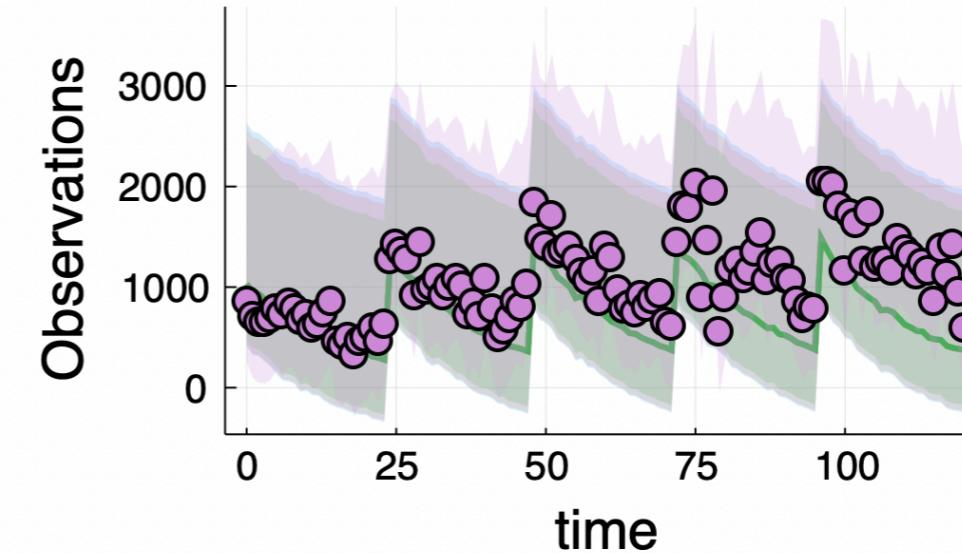
# validate

```
julia> vpc(res,200;stratify_on=[:wt]) |> plot
```

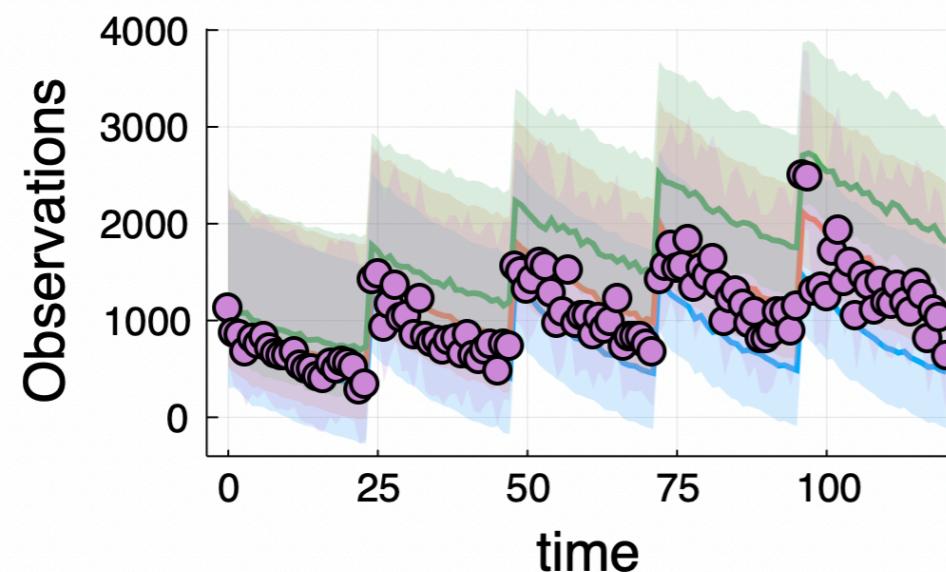
Stratified on: wt 73.0



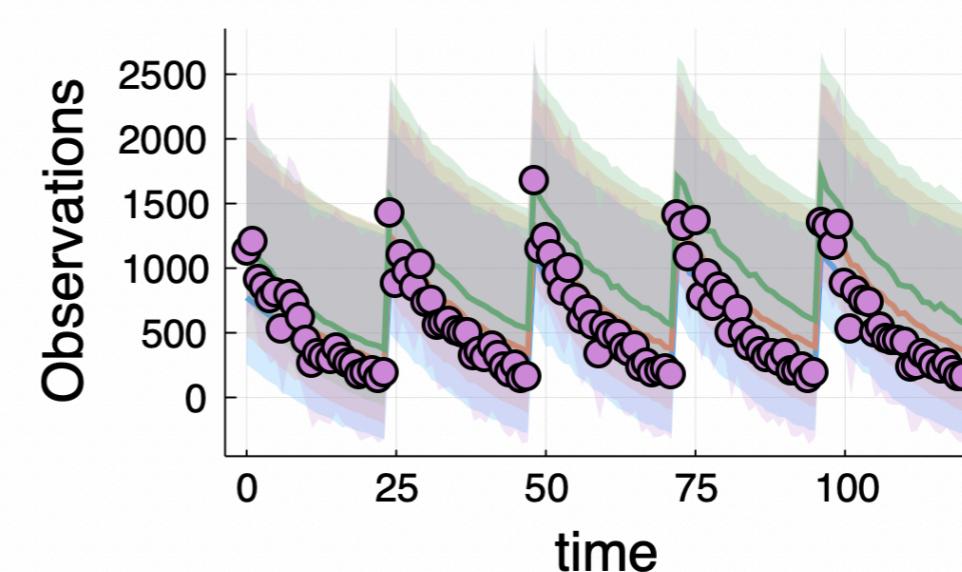
Stratified on: wt 76.5



Stratified on: wt 78.5



Stratified on: wt 80.0





## We run the simulation with a simobs call

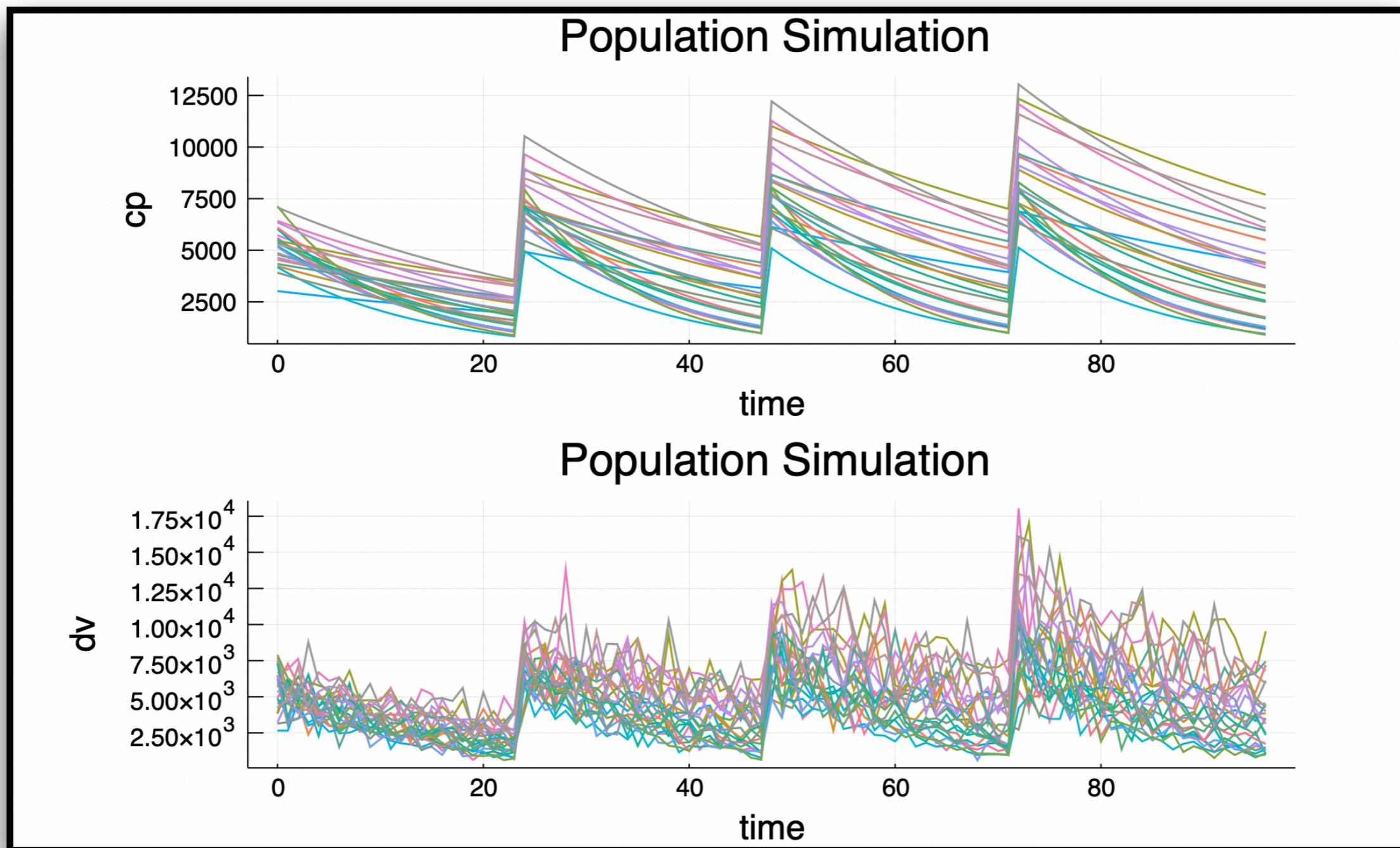
```
julia> sim1 = simobs(m_diffeq, ev1, p; abstol=1e-14, reltol=1e-14)
```

## The subject wise simulation output

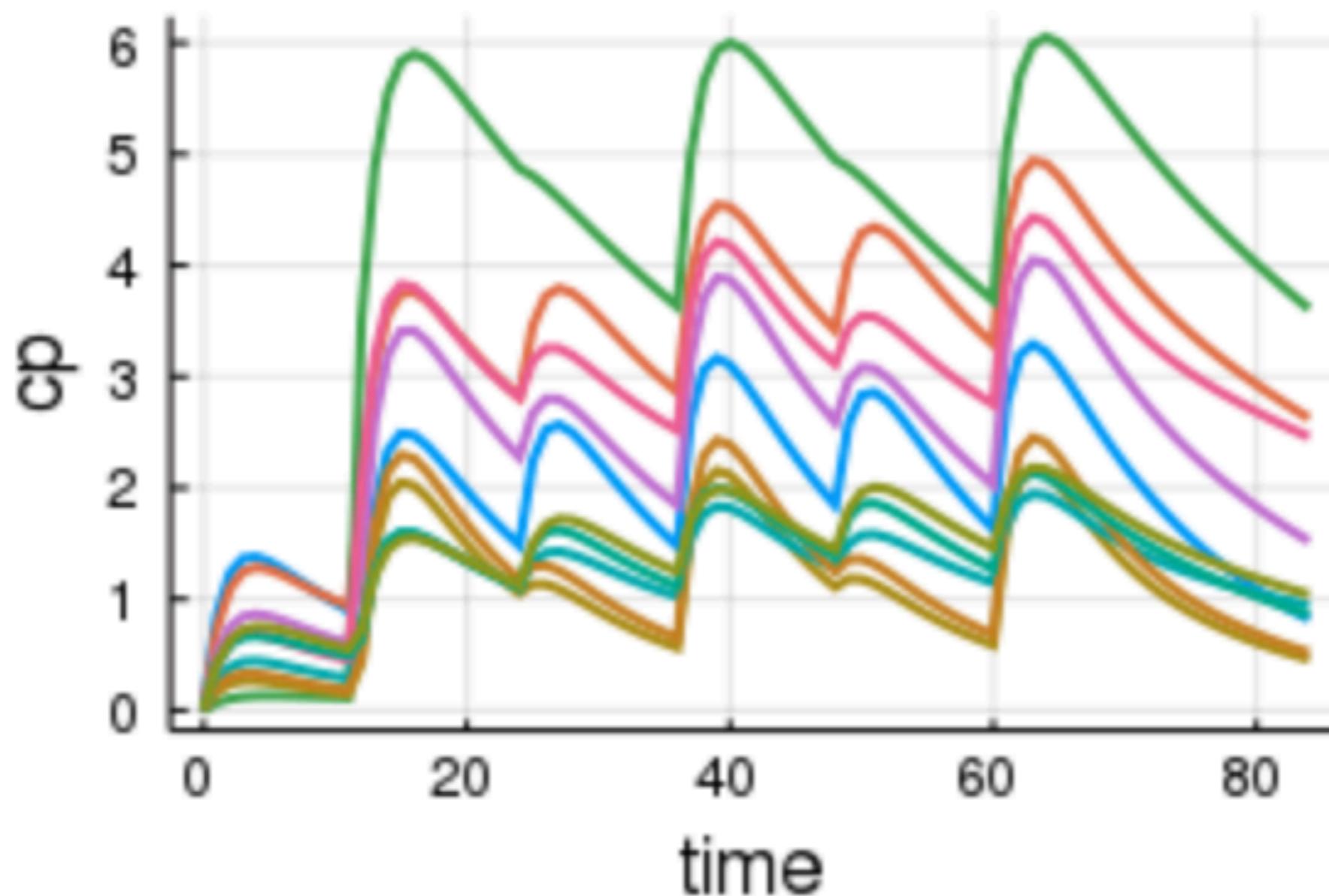
```
Subject
ID: 1
Events: 4
, 0.0:1.0:96.0, (cp = [4114.33, 4057.56, 4001.57, 3946.35, 3891.9, 3838.2, 3785.24, 3733.0,
3681.5, 3630.7 ... 8676.2, 8556.49, 8438.42, 8321.98, 8207.15, 8093.91, 7982.22, 7872.08, 7763.46,
7656.33], dv = [3254.99, 1998.61, 3497.23, 4780.26, 3041.26, 2604.17, 3277.17, 3295.85, 4776.44,
3386.44 ... 8335.86, 8939.89, 10491.5, 7646.75, 7337.47, 8872.07, 5317.99, 6404.97, 7250.41,
9919.32]))
```

<https://docs.pumas.ai/dev/basics/simulation/#Simulation-of-Pumas-Models-1>

```
julia> plot(sim1)
```



To answer the question, just change the dosage regimen and plug it into the simobs function to simulate different scenarios.



# Acknowledgments

- Vijay Ivaturi
- Chris Rackauckas
- Andreas Noack
- Patrick Mogensen
- Joga Gobburu
- JC Bangalore Team
- Simon Byrne

