A Deep Dive into Generative Scientific Machine Learning

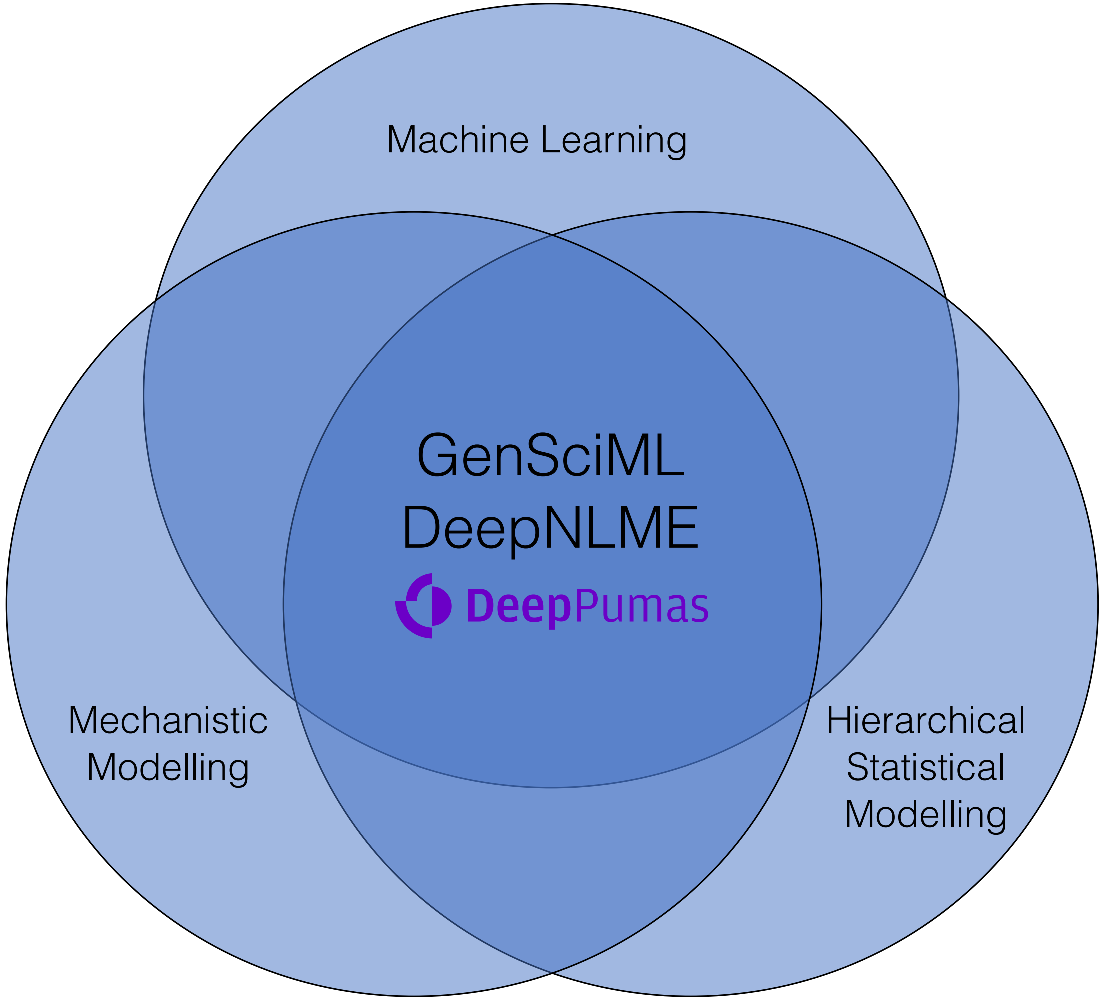
Building models that understand biology,
learn from data, and simulate realistic populations

Lorenzo Contento
PumasAI, DeepPumas team

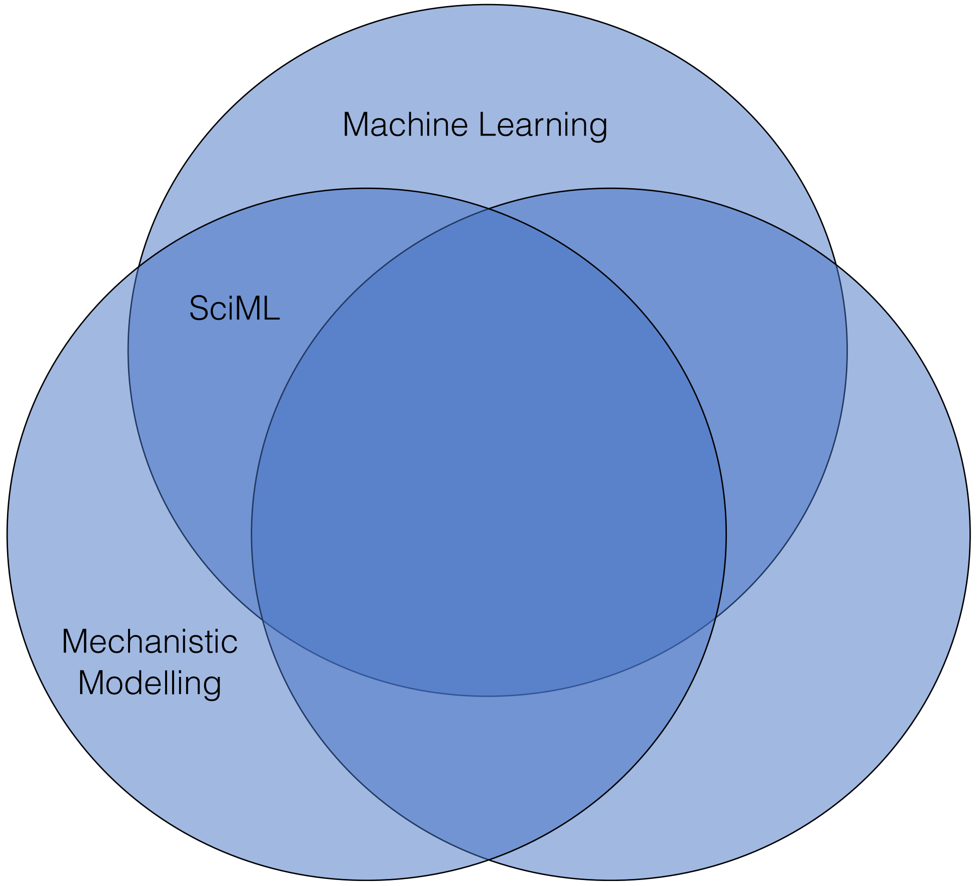
Niklas Korsbo
PumasAI, DeepPumas team

Mohamed Tarek
PumasAI, DeepPumas team

2nd Bonn Conference on Mathematical Life Sciences, 18 March 2026


Machine Learning
GenSciML
DeepNLME
DeepPumas
Mechanistic Modelling
Hierarchical Statistical Modelling

Scientific Machine Learning (SciML)



Scientific Modelling

- Leverages a priori scientific understanding
- Interpretable mechanisms
- Data efficient — generalizes well from small data
- Simple counterfactuals (what-if scenarios)
- **But:** labour-intensive model development
- **But:** misses unintuitive relationships
- **But:** hard to utilize complex data

Machine Learning (ML)

- Data-driven model discovery
- Finds unintuitive relationships
- Handles complex data (images, free-form text)
- **But:** black-box — lacks scientific interpretability
- **But:** requires big data — high risk of overfitting

Scientific Machine Learning (SciML)

- encode known science to constrain the model
- use data-driven flexibility where knowledge is lacking

An example scientific model

InDirect Response with oral drug administration

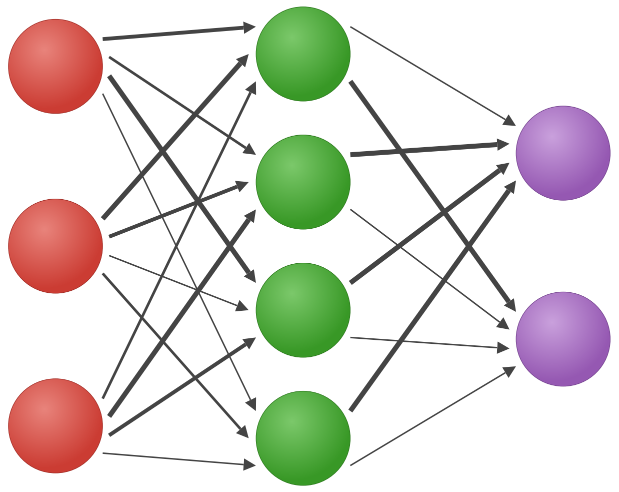
$$\begin{cases} \left.\begin{aligned} \frac{\mathrm{dDepot}}{\mathrm{d}t} &= -K_a \cdot \mathrm{Depot} \\ \frac{\mathrm{dCentral}}{\mathrm{d}t} &= K_a \cdot \mathrm{Depot} - \mathrm{CL} \cdot \frac{\mathrm{Central}}{V_c} \end{aligned}\right\} & \text{Pharmacokinetics (PK) model} \\ \frac{\mathrm{d}R}{\mathrm{d}t} = K_{\mathrm{in}} \cdot \left(1 + h\left(\frac{\mathrm{Central}}{V_c}\right)\right) - K_{\mathrm{out}} \cdot R & \text{Pharmacodynamics (PD) model} \end{cases}$$

where the **true drug-effect** is a Hill function:

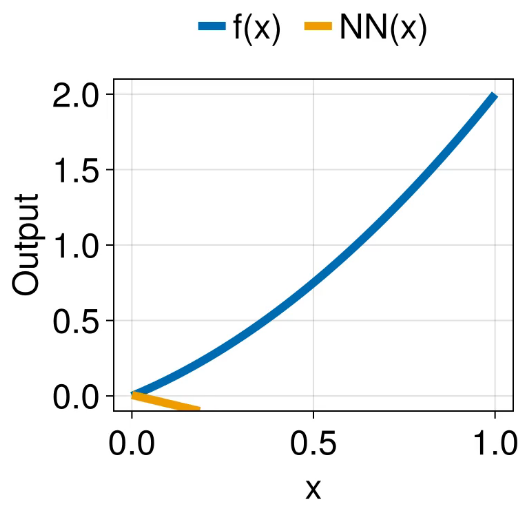
$$h(c) = S_{\max} \cdot \frac{c^n}{\mathrm{SC}_{50}^n + c^n}$$

External forcing: each time the drug is administered Depot undergoes a discontinuous jump.

Neural networks (NNs)



Universal approximators!



- Loosely based on neurons
- Mathematically just a function!
- Usable anywhere you would use a function!

- Approximate *any* function
- Functional form tuned by parameters

Neural Ordinary Differential Equations

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \mathrm{NN}(x, t)$$

A continuous-depth neural network

Similar to recurrent neural networks and ResNets

Universal Differential Equations (UDE)

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = x \cdot y - \mathrm{NN}(x) \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = p - x \cdot y \end{cases}$$

Unknown functional terms are treated as parameters to fit

Parametrized universal function approximators, e.g., neural networks, are used in practice

2018 “Neural Ordinary Differential Equations”, Chen et al.

2020 “Universal Differential Equations for Scientific Machine Learning”, Rackauckas et al.

Increasing levels of a priori knowledge

$$\begin{cases} \dfrac{\mathrm{dDepot}}{\mathrm{d}t} = \mathrm{NN}(\mathrm{Depot}, \mathrm{Central}, R)[1] \\ \dfrac{\mathrm{dCentral}}{\mathrm{d}t} = \mathrm{NN}(\mathrm{Depot}, \mathrm{Central}, R)[2] \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = \mathrm{NN}(\mathrm{Depot}, \mathrm{Central}, R)[3] \end{cases}$$

- Number of states
- Essentially a neural ODE, i.e., pure ML

$$\begin{cases} \dfrac{\mathrm{dDepot}}{\mathrm{d}t} = -\mathrm{NN}_1(\mathrm{Depot}) \\ \dfrac{\mathrm{dCentral}}{\mathrm{d}t} = \mathrm{NN}_1(\mathrm{Depot}) - \mathrm{NN}_2(\mathrm{Central}) \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = \mathrm{NN}_3(\mathrm{Central}, R) \end{cases}$$

- Number of states
- Relationships
- Dependence/independence of terms
- Conservation between Depot and Central

Increasing levels of a priori knowledge

$$\begin{cases} \frac{\mathrm{dDepot}}{\mathrm{d}t} = -K_a \cdot \mathrm{Depot} \\ \frac{\mathrm{dCentral}}{\mathrm{d}t} = K_a \cdot \mathrm{Depot} - \mathrm{CL} \cdot \frac{\mathrm{Central}}{V_c} \\ \frac{\mathrm{d}R}{\mathrm{d}t} = \mathrm{NN}\left(\frac{\mathrm{Central}}{V_c}, R\right) \end{cases}$$

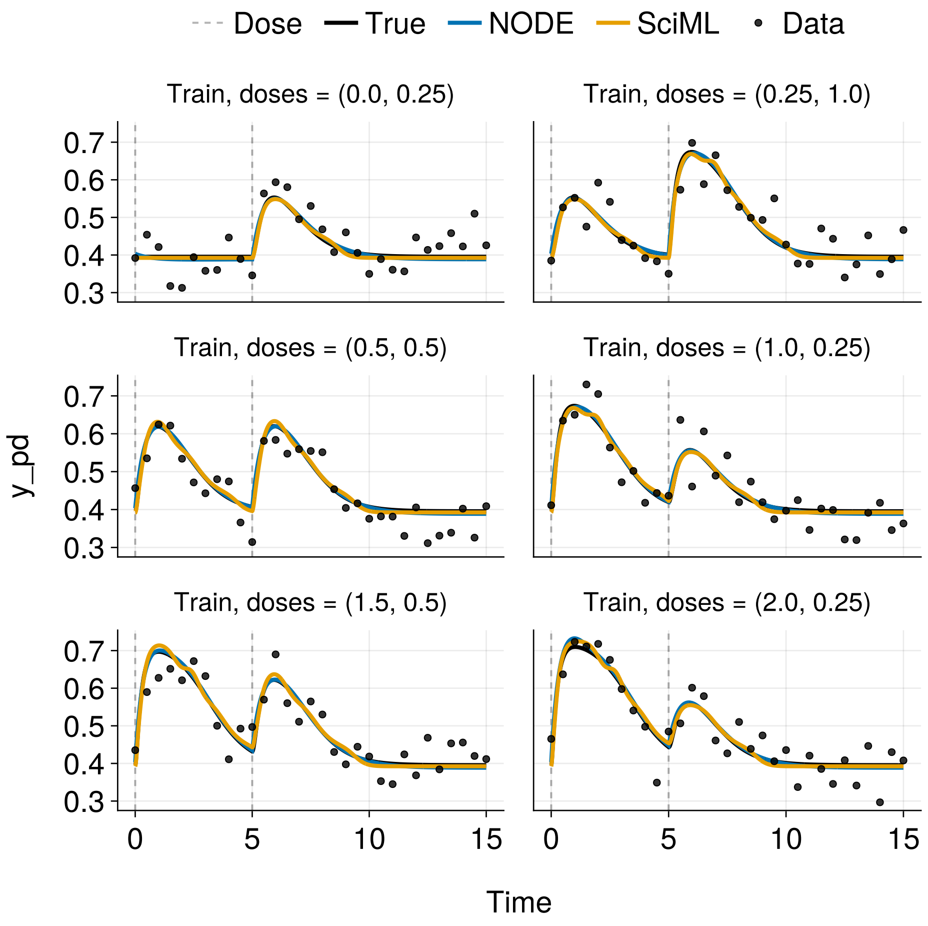
$$\begin{cases} \frac{\mathrm{dDepot}}{\mathrm{d}t} = -K_a \cdot \mathrm{Depot} \\ \frac{\mathrm{dCentral}}{\mathrm{d}t} = K_a \cdot \mathrm{Depot} - \mathrm{CL} \cdot \frac{\mathrm{Central}}{V_c} \\ \frac{\mathrm{d}R}{\mathrm{d}t} = K_{\mathrm{in}} \cdot \left(1 + \mathrm{NN}\left(\frac{\mathrm{Central}}{V_c}\right)\right) - K_{\mathrm{out}} \cdot R \end{cases}$$

- Explicit knowledge of some terms
- Relationships (R independent of Depot!)

- Precise position of the unknown function
- Precise input to the unknown function
- Lots of knowledge!

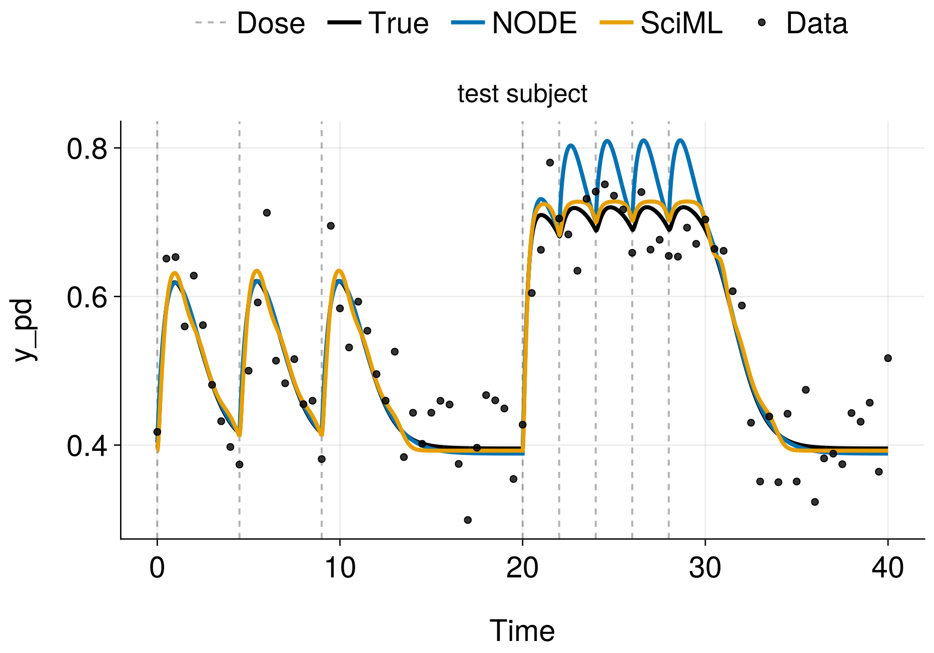
Neural ODE vs UDE

- Train on 36 dosing regimens for one subject
- Both models fit the data (PK + PD) well

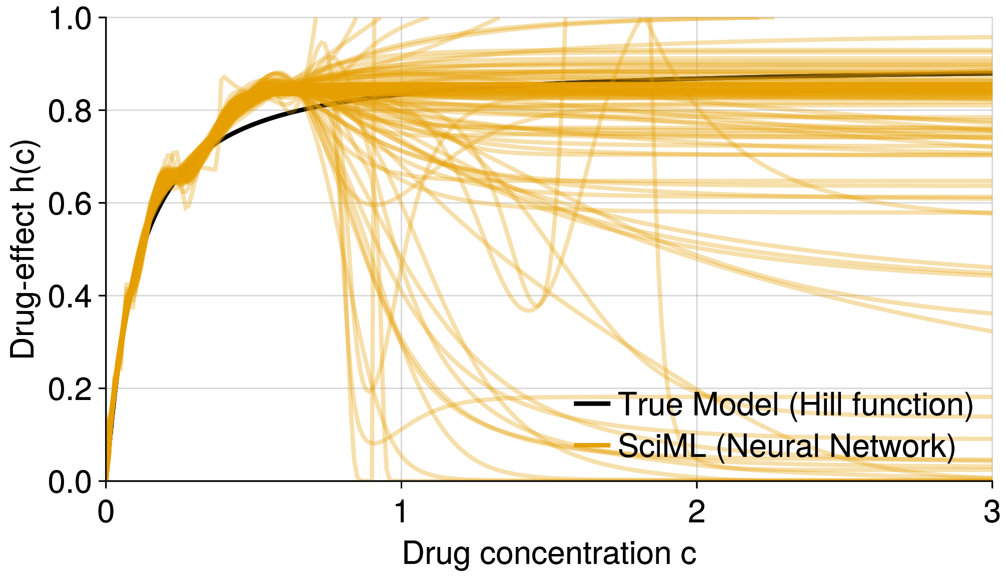


Neural ODE vs UDE

- Train on 36 dosing regimens for one subject
- Both models fit the data (PK + PD) well
- Only SciML generalizes to unseen dosing schedules:
  - seen dosage, but higher frequency
  - note that the first dose in the sequence is correctly predicted by NODE too
- Mechanistic structure constrains the NN, enabling extrapolation.

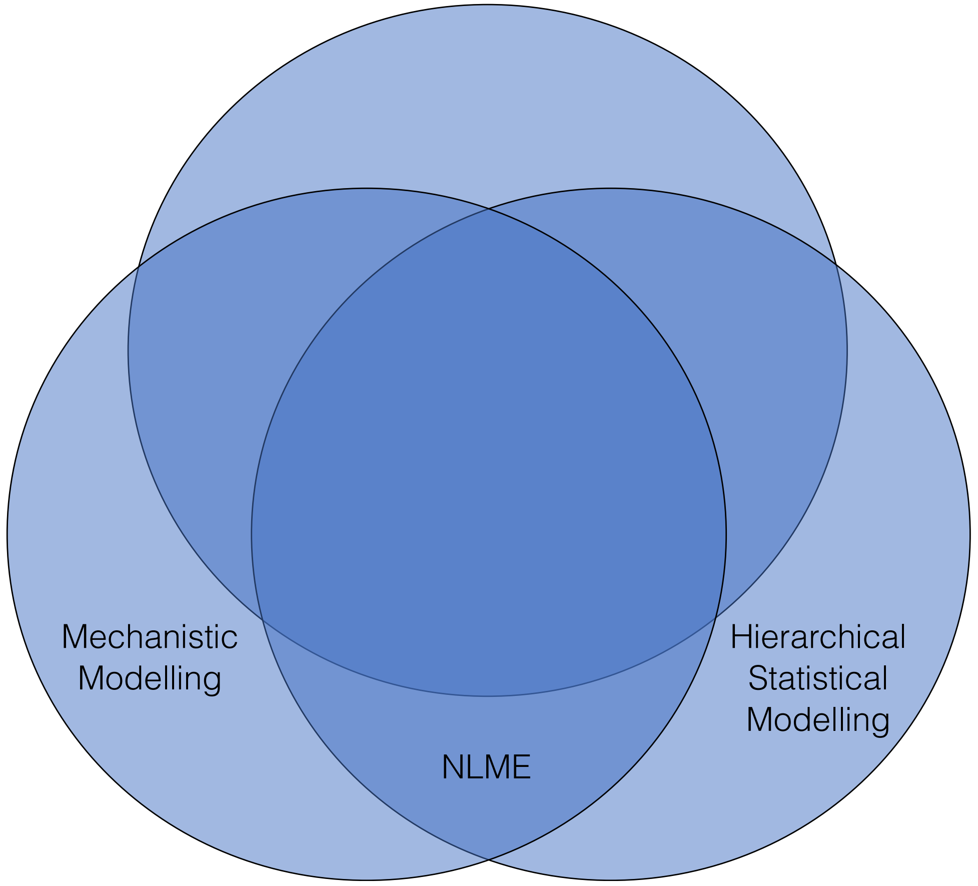


Learned drug-effect function



- Check identifiability with different starting parameters and/or bootstrapping
  - Proper uncertainty quantification (MCMC) is difficult
- Generalizes badly for large concentrations (not seen during training)
  - Next step: enforce saturation effect?

NonLinear Mixed Effects (NLME) modelling

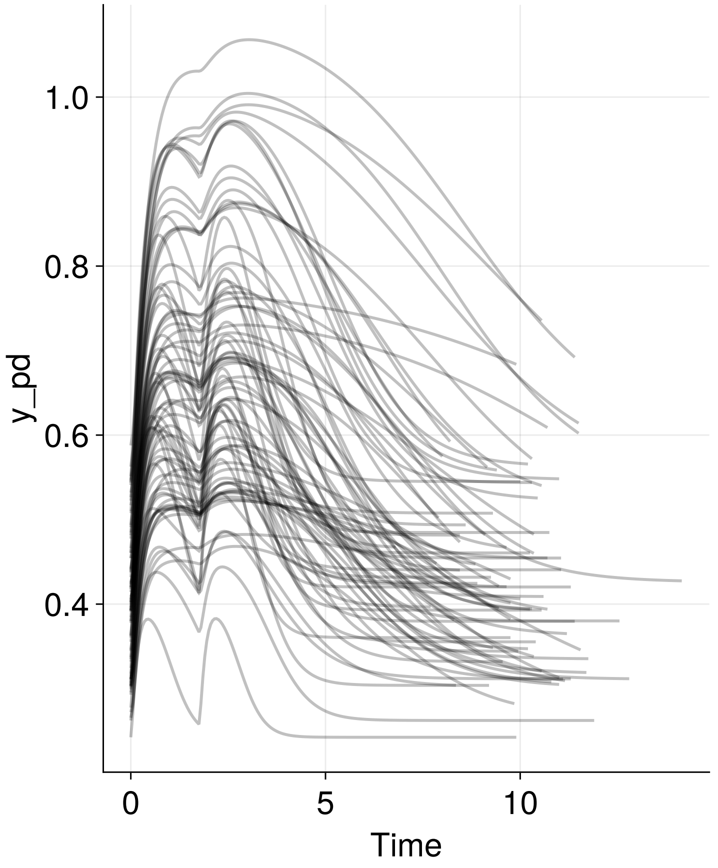


DeepPumas

From a single subject to a population

- Subjects respond differently due to biological variability (**heterogeneous population**)
  - different subjects, same dose $\longrightarrow$ different response

- Fitting one individual model per subject:
  - requires enough data per subject
  - cannot share information across subjects
  - cannot describe the *distribution* of responses



Hierarchical modelling

aka NonLinear Mixed Effects (NLME) modelling

- All subjects share the same model structure, but parameters are partitioned into:
  - **Population parameters** (fixed effects): shared across all subjects
  - **Individual parameters** (random effects): values are unique to each subject
- A **population distribution** for individual parameters links individual models together
  - This distribution can be parametrized using population parameters
  - This enables sharing information across subjects and characterizing variability

A NonLinear Mixed Effects (NLME) model

$$\begin{cases} \dfrac{\mathrm{dDepot}}{\mathrm{d}t} = -K_a \cdot \mathrm{Depot} \\ \dfrac{\mathrm{dCentral}}{\mathrm{d}t} = K_a \cdot \mathrm{Depot} - \mathrm{CL} \cdot \dfrac{Central}{V_c} \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = K_{\mathrm{in}} \cdot \left(1 + h\left(\dfrac{\mathrm{Central}}{V_c}\right)\right) - K_{\mathrm{out}} \cdot R \\ h(c) = S_{\max} \cdot \dfrac{c^n}{\mathrm{SC}_{50}^n + c^n} \end{cases}$$

- CL, V_c, K_{in} and SC_{50} are population parameters
- $K_a, K_{\mathrm{out}}, S_{\max}$ and n are individual parameters
- $(K_a, K_{\mathrm{out}}, S_{\max}, n) \sim \mathrm{MvLogNormal}(\mu, \Omega)$
- $\mu \in \mathbb{R}^4$ (typical values) and $\Omega \in \mathbb{R}^{4\times4}$ (inter-individual variability) are population parameters

Fitting an NLME model

Notation:

y_i	data for subject i
θ	population parameters
η_i	individual parameters for subject i

Goal: estimate θ, since it alone determines the distribution of responses in the population

As usual for a probabilistic model, we maximize the loglikelihood of the data $\{y_i\}_i$:

$$p(y \mid \theta) = \prod_{i=1}^{N} p(y_i \mid \theta) = \prod_{i=1}^{N} \int \underbrace{p(y_i \mid \eta_i, \theta)}_{\text{likelihood of } y_i \text{ given } \eta_i} \cdot \overbrace{p(\eta_i \mid \theta)}^{\text{distribution of } \eta \text{ in the population}} \mathrm{d}\eta_i$$

Since we need to integrate out the individual parameters, this is known as the **marginal loglikelihood**.

Natural regularization effect
By averaging over all plausible η values (weighted by their population distribution), solutions that only fit well for extreme η are down-weighted.

Fitting an NLME model

$$p(y \mid \theta) = \prod_{i=1}^{N} \int p(y_i \mid \eta_i, \theta) \cdot p(\eta_i \mid \theta) \, \mathrm{d}\eta_i$$

Note that the integrand is $\propto p(\eta_i \mid y_i, \theta)$, the posterior distribution of η_i.

The integral is intractable — approximation methods are needed.

Laplace approximation (FOCE)

1. Find the **empirical Bayes estimate (EBE)**: mode of $p(\eta_i \mid y_i, \theta)$
2. Approximate the posterior with a Gaussian (via Hessian at the EBE)
3. Evaluate the integral analytically

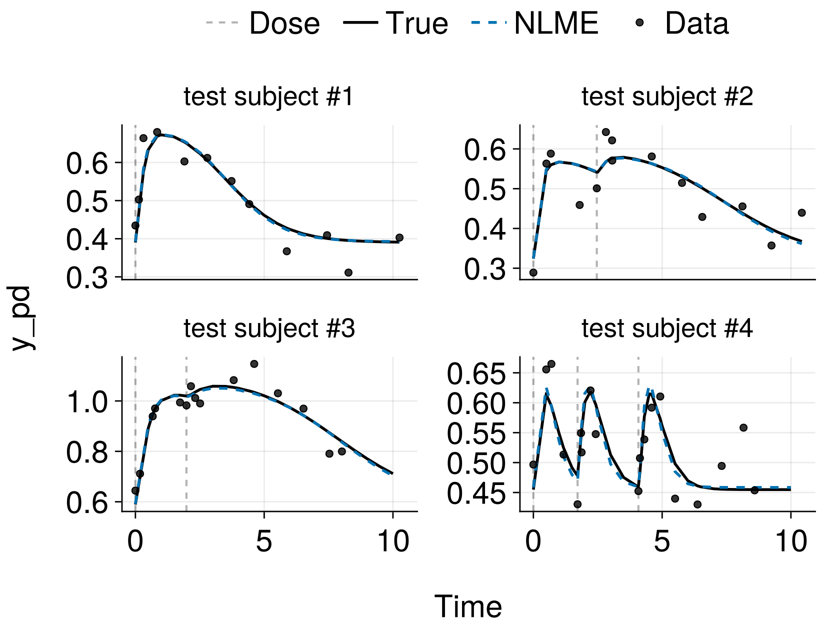
Default approach in pharmacometrics — justified by the **Bernstein–von Mises theorem**: with sufficient data per subject, the posterior converges to a Gaussian.

Evaluation of an NLME model

Train the NLME model on 60 subjects with various dosage regimens

Individual predictions (using the EBEs)

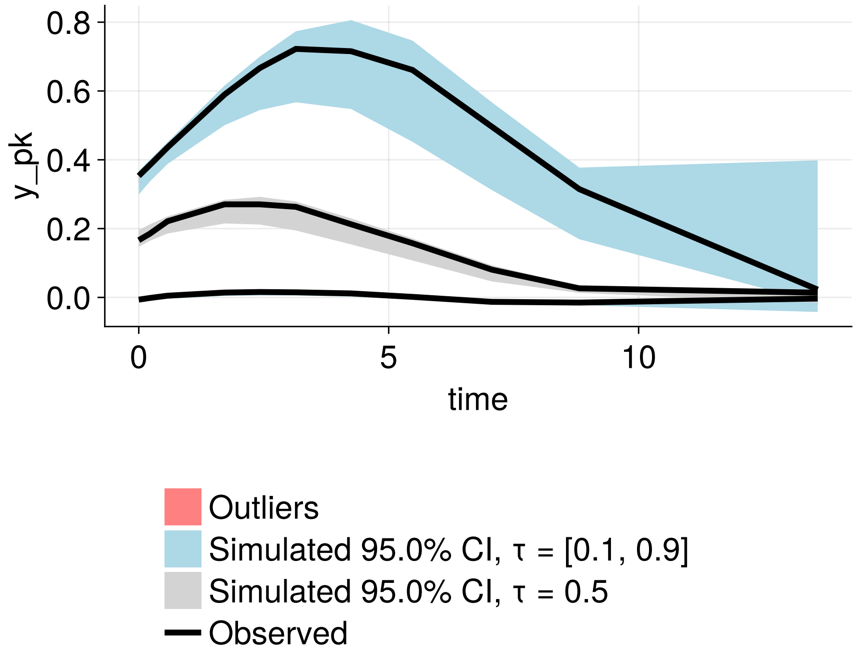
Is the model powerful enough
to describe the data of each subject?



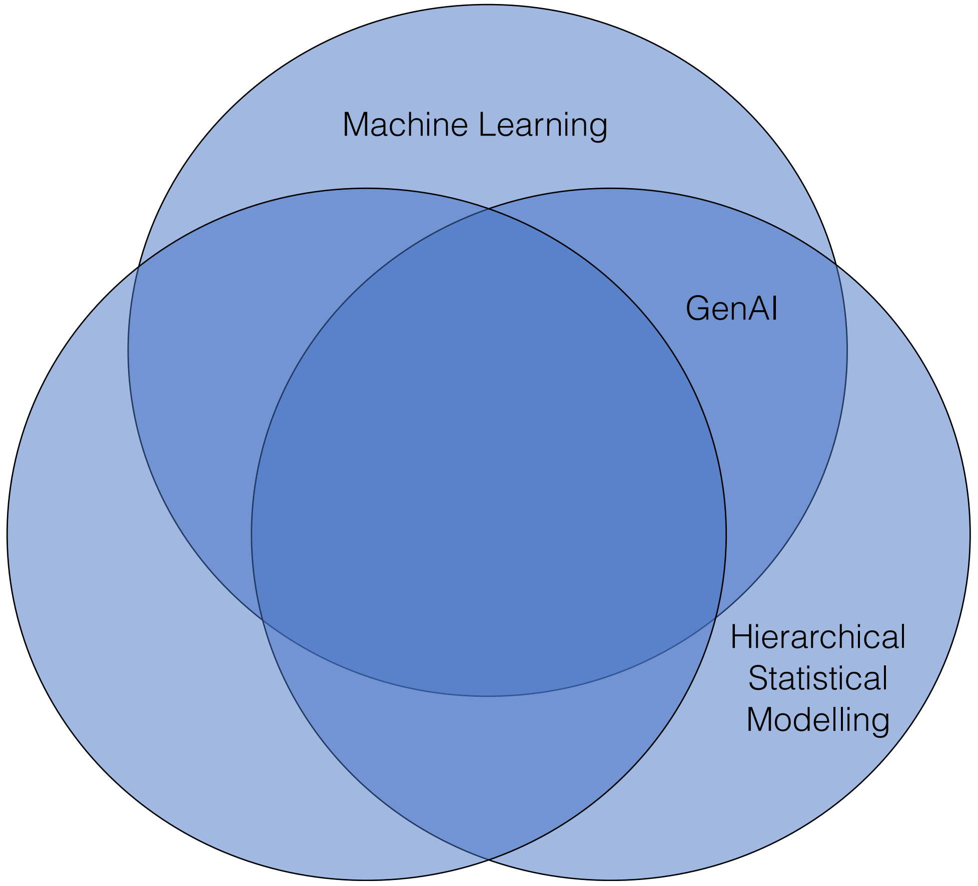
Visual Predictive Check (VPC)

A visual way to check whether distributions align.

Compare quantiles of the train/test population
with CIs obtained from simulated populations



NLME models are generative models



DeepPumas

Generative models

A generative model describes the distribution of a random variable $Y \in \mathbb{R}^n$.

This is usually done by defining a mechanism for sampling from Y:

1. draw a sample z from $Z \in \mathbb{R}^k$, a random variable Z with known distribution
   - Z is known as a *latent variable*
   - Z is usually a standard normal distribution
   - $k \leq n$
2. draw a sample y from $Y \mid Z = z$ by computing $y = f(z) + \varepsilon$, where
   - $f : \mathbb{R}^k \to \mathbb{R}^n$ in a family parametrized by θ
   - $\varepsilon \sim p(\cdot \mid \theta, f(z))$ is a noise term
     - ε may be always zero

In general, the probability density function of Y cannot be computed.

- **Sampling from Y**
  - $Y = f(Z) + \varepsilon$ with $Z \sim \mathcal{N}(0, I)$
- **Encoder**
  - The encoder returns the value of Z that is most likely to generate a given value $Y = y$, i.e., the mode of the posterior $p(z \mid y, \theta)$ (or an approximation to it).
  - If $k < n$ the value returned by the encoder is the lower dimensional *embedding* of y.
  - Similarly the function f is often called the *decoder*.
- **Fitting**
  - Z is a missing variable $\rightarrow$ the EM framework applies!
  - Since the expectation is intractable, the **Evidence Lower Bound (ELBO)** is maximized instead.
  - The encoder is used as the proposal distribution.

Examples

- **Probabilistic PCA**
  - f is a multiplication by a matrix $W \in \mathbb{R}^{n\times k}$
  - ε is a multivariate gaussian with zero mean and diagonal covariance matrix
- **Variational autoencoder (VAE)**
  - f is a neural network
  - ε is usually a multivariate gaussian with zero mean and diagonal covariance matrix
  - the encoder is usually given by another neural network jointly trained with f

NLME models are generative models

Using an NLME model we can sample a response for a subject in the population by:

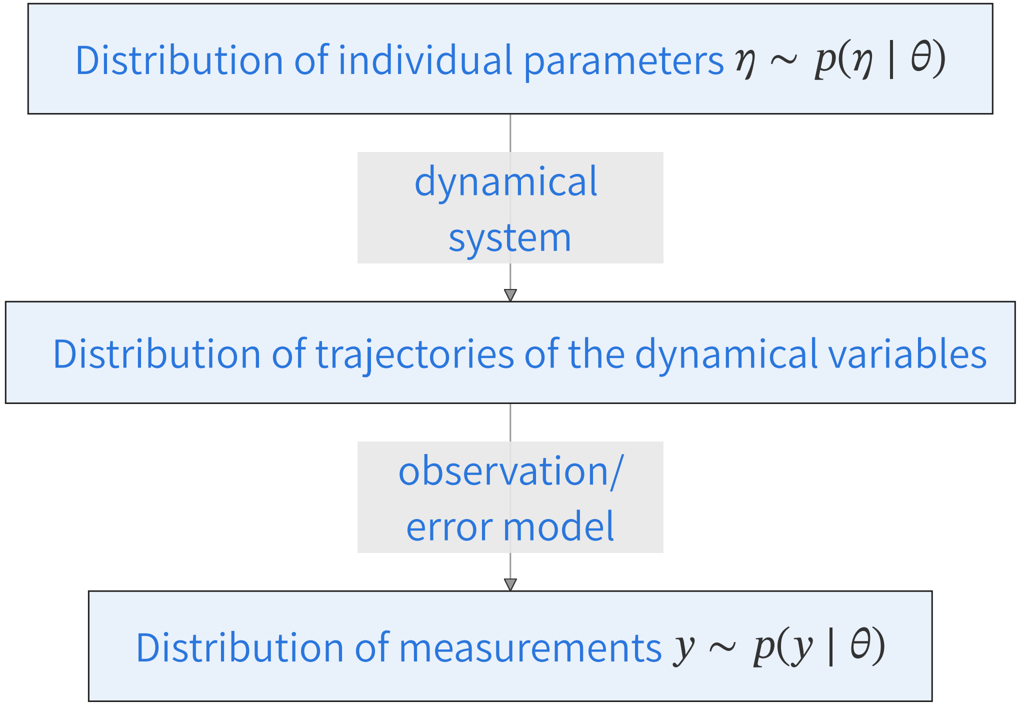
1. Sample new individual parameters η according to $p(\eta \mid \theta)$
   - the fact that the distribution depends on θ is not limiting
2. Solve the ODE to get the trajectories of the dynamical variables
3. Sample observations using the error model

This is exactly a generative model! Just with a different notation:

- The latent variables are η, the individual parameters
- The decoder f is the dynamical model and ε is the error model
- The embeddings are given by the EBEs
  - An NLME model can be thought of as a feature extractor!

2025 “Generative Machine Learning and Nonlinear Mixed Effects Modeling are the Exact Same Thing”, Mohamed Tarek, webinar

The individual model as a map between distributions


Distribution of individual parameters $\eta \sim p(\eta \mid \theta)$
dynamical system
Distribution of trajectories of the dynamical variables
observation/ error model
Distribution of measurements $y \sim p(y \mid \theta)$

DeepPumas

What about fitting?

EM methods are applicable to NLME models too: the individual parameters η are the missing variables.

Stochastic Approximation EM (SAEM) is the most commonly used version in pharmacometrics:

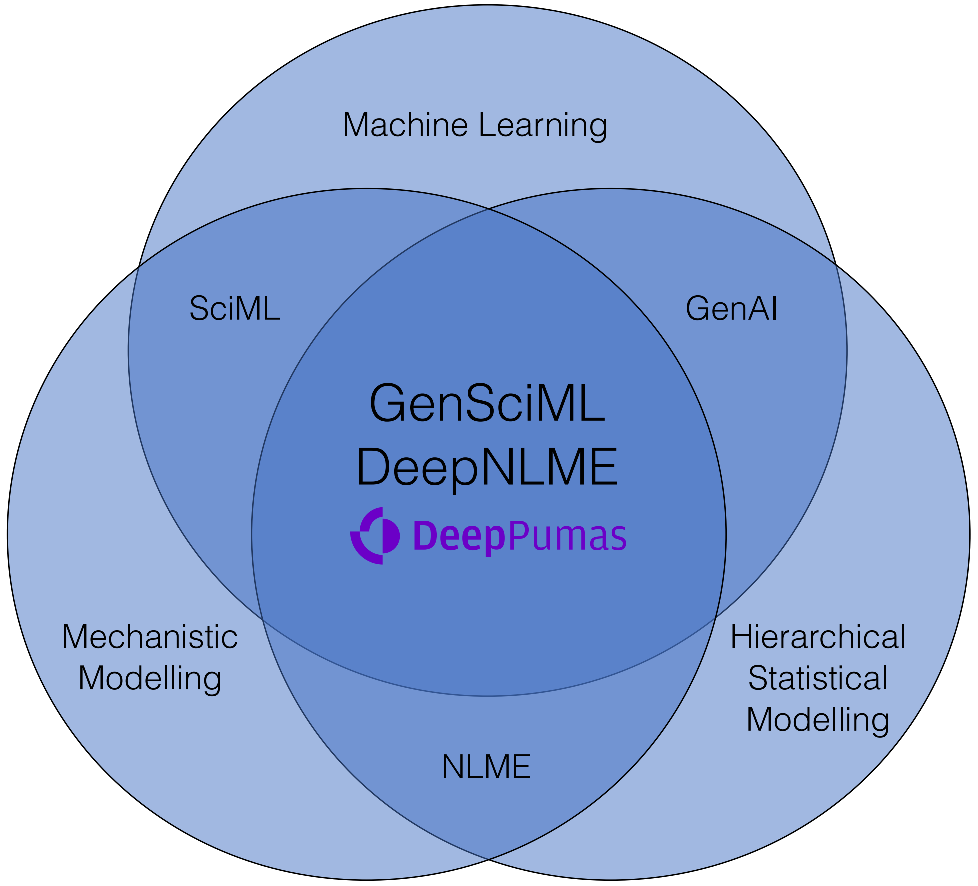
- use MCMC samples from the posterior as proposals
- shines when the data is sparse and the posteriors are not at all Gaussian

What about covariates?

- The distribution of Y may depend on known subject-level covariates x:
  - body weight, age, $\dots$
  - room temperature (time-dependent covariate)
  - the dosing regimen in an NLME model (time-dependent covariate)
  - a text prompt in an image generation model
- A **conditional generative model** takes this into account, by modelling $p(Y \mid x)$
- Its decoder f is parametrized by both θ and the covariate values x

$$y = f(z;\, \theta,\, x) + \varepsilon$$

DeepNLME, an example of GenSciML



From a SciML model to a DeepNLME model

$$\begin{cases} \dfrac{\mathrm{dDepot}}{\mathrm{d}t} = -K_a \cdot \mathrm{Depot} \\ \dfrac{\mathrm{dCentral}}{\mathrm{d}t} = K_a \cdot \mathrm{Depot} - \mathrm{CL} \cdot \dfrac{\mathrm{Central}}{V_c} \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = K_{\mathrm{in}} \cdot \left(1 + \mathrm{NN}\left(\dfrac{\mathrm{Central}}{V_c}\right)\right) - K_{\mathrm{out}} \cdot R \end{cases}$$

The NN is a **population-level** object, shared by all subjects. How do we individualize it?

1. **Make NN weights individual parameters**
   - Problem: NNs have many parameters!
     - Do we really need that many degrees of freedom?
       Certainly not, so how can we choose which parameters to individualize?
     - The Laplace approximation scales poorly with dimensionality

From a SciML model to a DeepNLME model

$$\begin{cases} \dfrac{\mathrm{dDepot}}{\mathrm{d}t} = -K_a \cdot \mathrm{Depot} \\ \dfrac{\mathrm{dCentral}}{\mathrm{d}t} = K_a \cdot \mathrm{Depot} - \mathrm{CL} \cdot \dfrac{\mathrm{Central}}{V_c} \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = K_{\mathrm{in}} \cdot \left(1 + \mathrm{NN}\left(\dfrac{\mathrm{Central}}{V_c}; \eta_{\mathrm{NN}}\right)\right) - K_{\mathrm{out}} \cdot R \end{cases}$$

2. **Pass dedicated random effects as additional NN inputs**

- A **family** of response functions $c \mapsto \mathrm{NN}(c; \eta_{\mathrm{NN}})$ indexed by η_{NN}
- If the population distribution for η_{NN} has independent components, the NN may learn **disentangled, interpretable** roles for each component

"DeepNLME - Conditional Generative Scientific Machine Learning with applications in Pharmacology", Korsbo et al., under review at Quantitative Medicine

DeepPumas tutorial

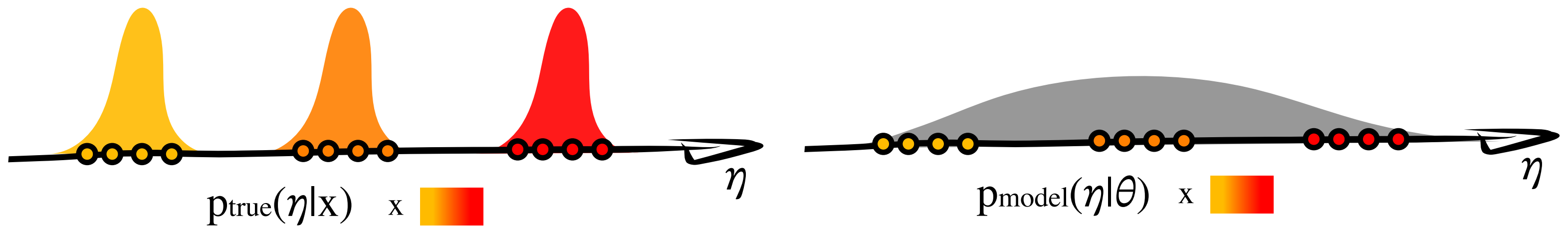
Correcting misspecification in the distribution of the individual parameters

- Traditional NLME: the population distribution of η is a standard parametric family
- **Problem**: the true distribution may have:
  - non-linear correlations: the dynamical model does not account for everything
  - multiple modes: e.g., responders vs non-responders
- **Solution**: learn the distribution from data using a **normalizing flow**.
  - An invertible transformation τ maps a simple base distribution (e.g., $\mathcal{N}(0, I)$) to the target
  - No dimensionality reduction — appropriate since the latent space already has low dimension
  - **The density can be computed**
  - Good interpretability (just plot the PDF!), especially if the dimensionality of η is small

… while conditioning on covariates

The distribution of η can also depend on a subject's **covariates** x:

$$\eta_i \sim p(\eta \mid \theta, x_i)$$



Approaches:

- $\eta_i \sim \mathcal{N}(\mu(x_i),\ \Sigma(x_i))$ where μ and Σ are NNs
- conditional normalizing flow — also captures non-linear correlations and multi-modality

Benefits of data-driven distributions (unconditional and conditional):

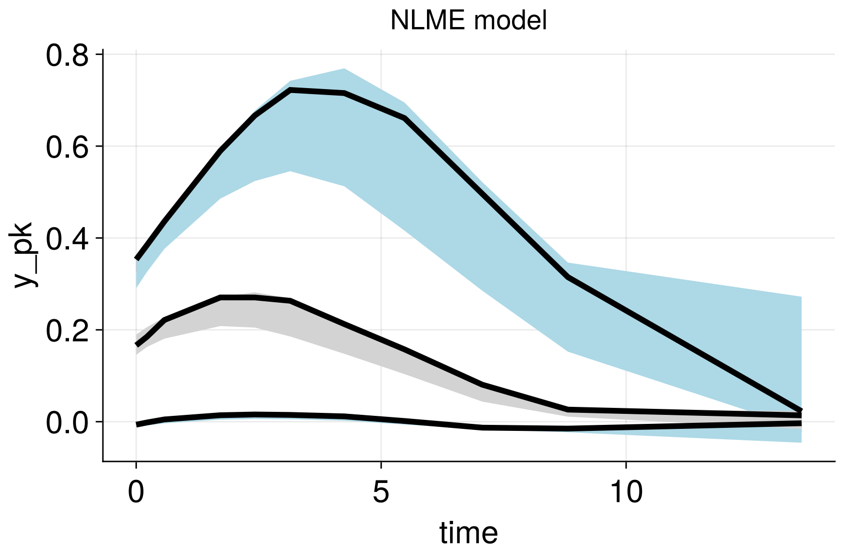
- Reduces unexplained IIV $\longrightarrow$ simulations match the real population more closely
  - Even tighter bands in the VPCs
- Improves downstream tasks built on simulation (e.g., experimental design)
- Less residual variability $\longrightarrow$ more precise predictions with sparse data (early trials, precision medicine)

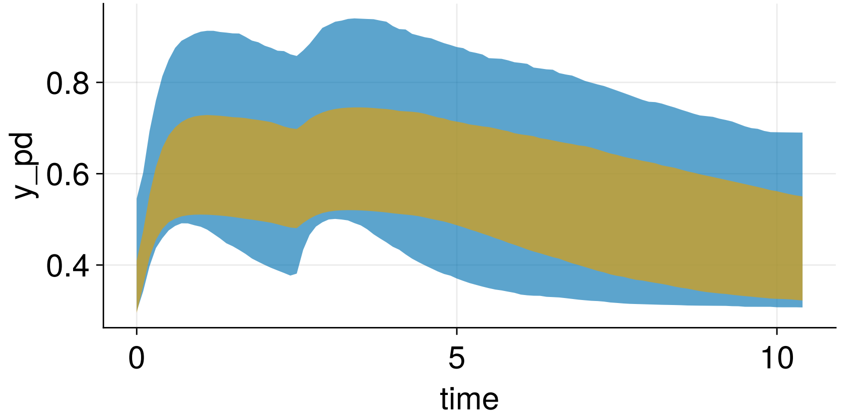
"Improving simulations by learning the true random effects' distribution from a population using generative machine learning", Lorenzo Contento, PAGE 2024 (webinar)

2025 "Finding complex relationships between random effects and covariates using data-driven distributions", Lorenzo Contento, ACoP 2025

"DeepNLME - Conditional Generative Scientific Machine Learning with applications in Pharmacology", Korsbo et al., under review at Quantitative Medicine

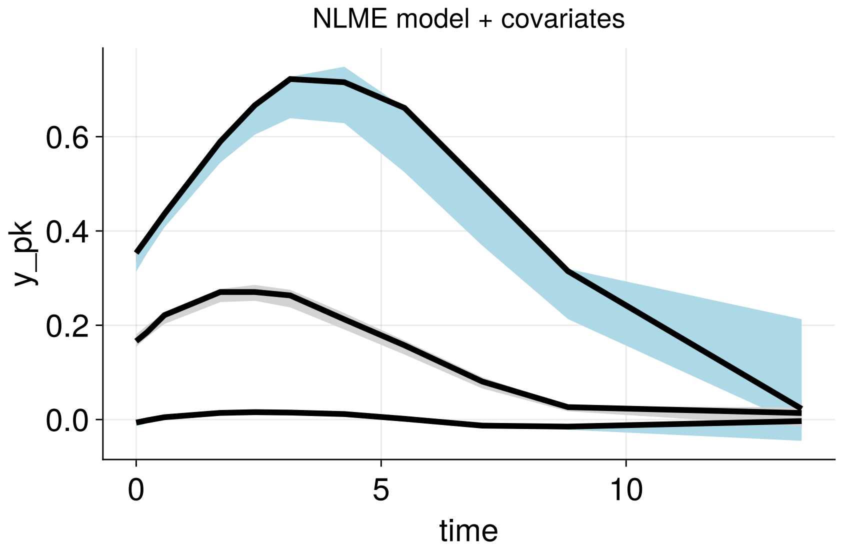
Getting started with DeepPumas tutorial

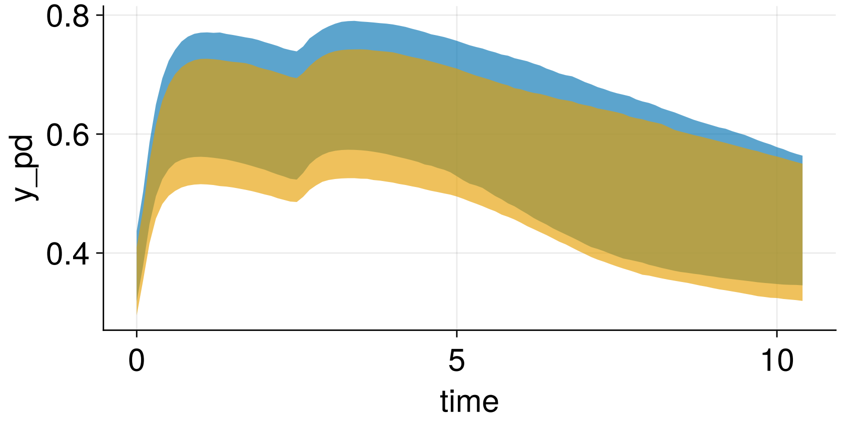




prediction 90% CI, single subject, no data

🟦 NLME model 🟨 True model





prediction 90% CI, single subject, no data

🟦 NLME model + covariates 🟨 True model

DeepPumas

- **SciML** encodes mechanistic knowledge into ML models $\longrightarrow$ data-efficient, generalizable
- **NLME** provides a principled framework for modelling heterogeneous populations
- **NLME models are generative models**:
  individual parameters are embeddings, the dynamical model is the decoder
- **DeepNLME** = generative SciML (genSciML)
  - combines mechanistic structure, neural flexibility, and population-level inference.
- Data-driven distributions (normalizing flows)
  - correct misspecification of the distribution of η in the population
  - conditioning on covariates further reduces IIV

- **Outlook**
  - seeing NLME as generative modelling opens the door to transferring insights from genML
  - NLME models can be linked to embedding models for other data modalities (text, images, omics) through sharing and/or aligning the latent spaces (embeddings $\iff$ individual parameters)
    - simplest approach: use text/image embeddings as covariates inside $p(\eta \mid \theta, x)$

DeepPumas publications are available at pumas.ai/resources/publications

“Free-form text as an NLME covariate and how embedding models enable effective use of predictive factors from a wide range of complex data” Niklas Korsbo, PAGE 2025, slides and video available on the conference website www.page-meeting.org