Probabilistic Machine Learning with Julia

Lecture 4 | Probabilistic Models for Time Series

Amirabbas Asadi

amir.asadi78@sharif.edu

2025

Today's Topics

Today we will have an overview of parametric probabilistic models for time series data:

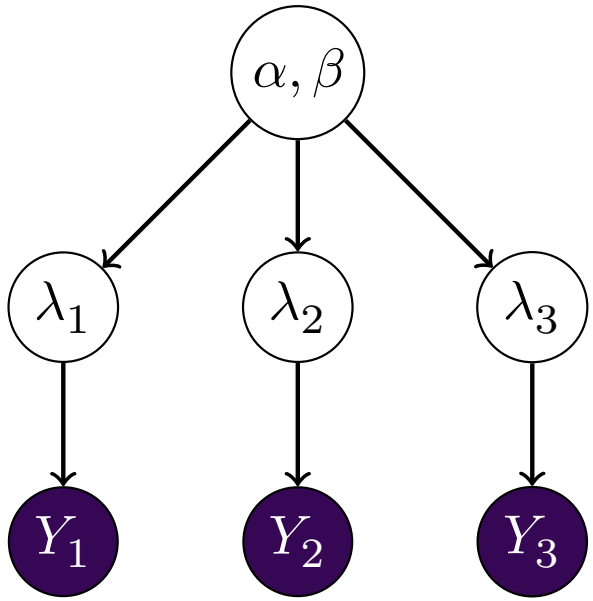
- Bayesian Neural Networks
- Autoregressive models and Dynamic Bayesian Networks
- Time series modeling with Bayesian Differential Equations
- Motivation for nonparametric models

Poisson Regression

In the previous lecture, we discussed a simple form of Poisson Regression:

$$
\begin{aligned}
Y_\tau &\sim \text{Poisson}(\lambda_\tau) \\
\log \lambda_\tau &= \alpha\tau + \beta \\
\alpha, \beta &\sim \text{Normal}(0, 1)
\end{aligned}
$$

```
@model function poisson_linear_model(y)
    α ~ Normal(0.0, 1.0)
    β ~ Normal(0.0, 1.0)

    for τ ∈ 1:length(y)
        log_λ = α*τ + β
        y[τ] ~ Poisson(exp(log_λ))
    end
end
```

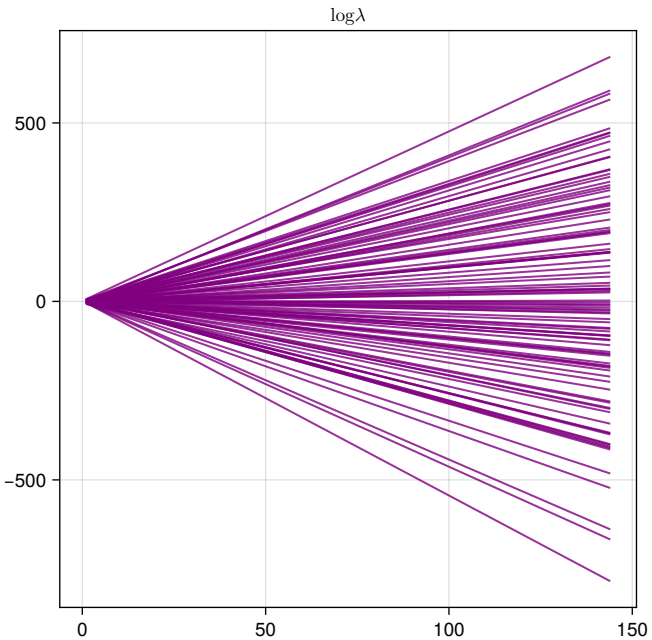


Poisson Regression

$$p(\alpha, \beta) \xrightarrow{\text{Bayesian Inference}} p(\alpha, \beta | y)$$

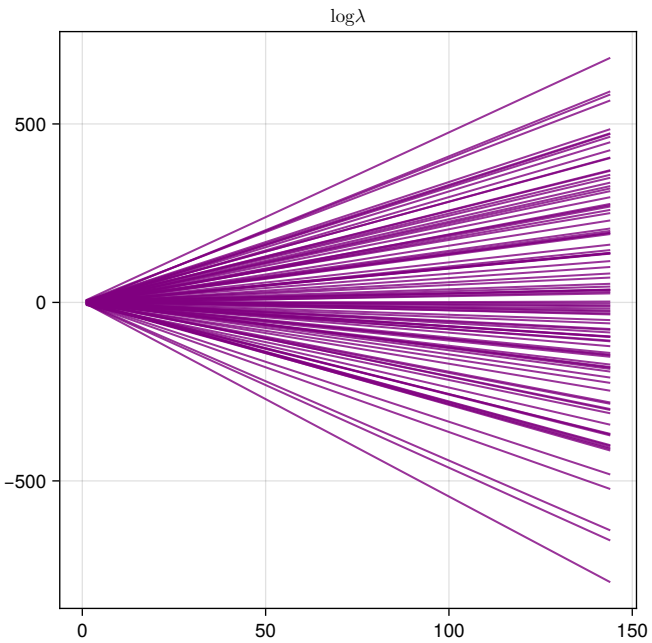
Poisson Regression

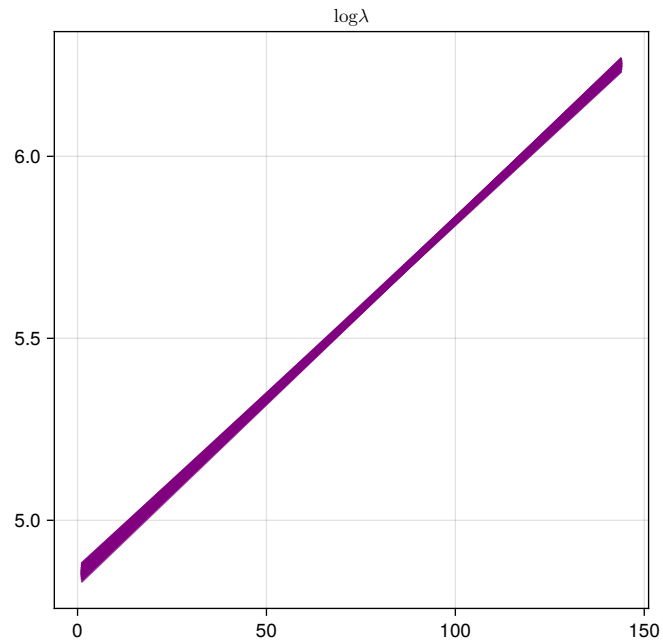
$$p(\alpha, \beta) \xrightarrow{\text{Bayesian Inference}} p(\alpha, \beta | y)$$



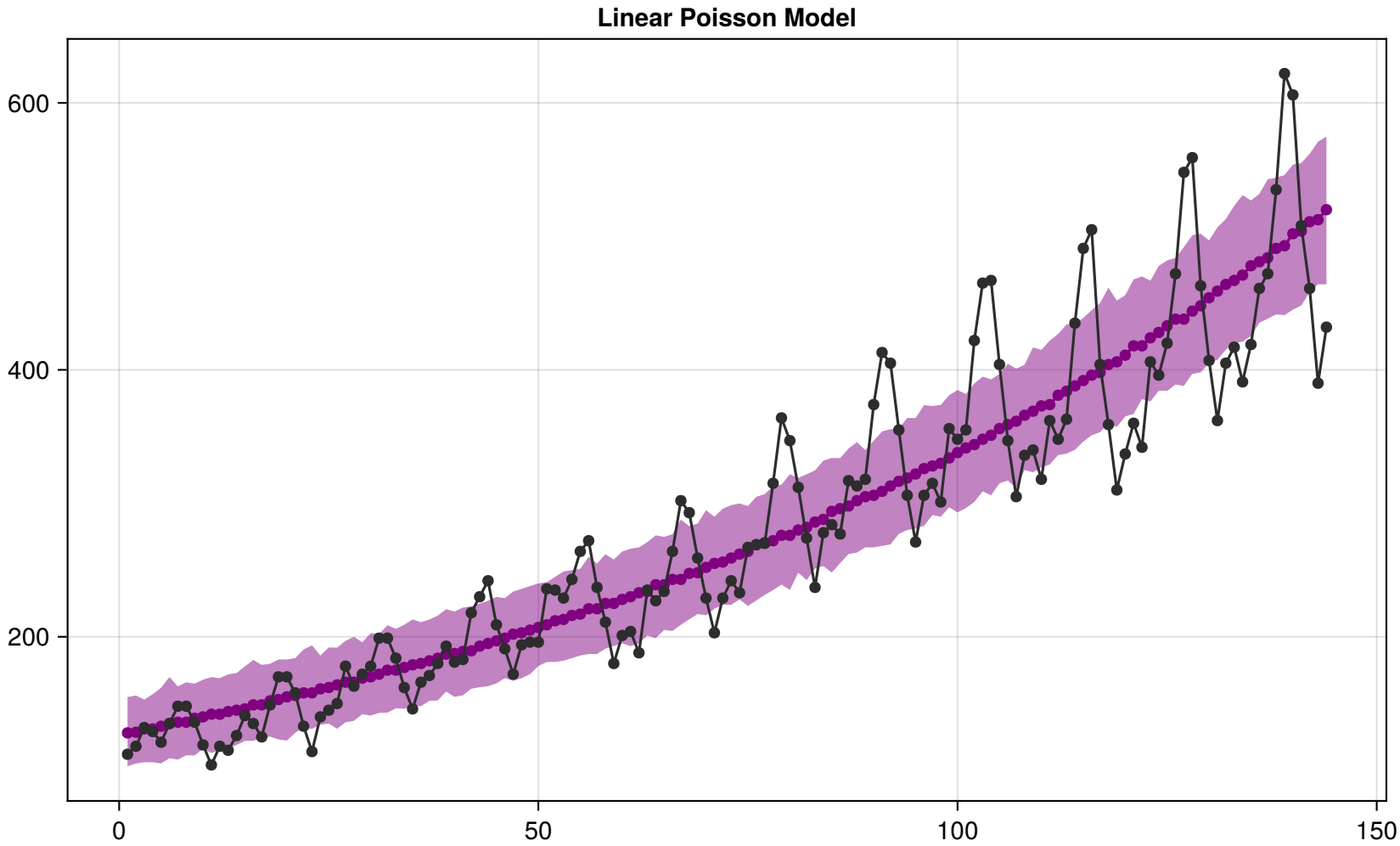
Poisson Regression

$$p(\alpha, \beta) \xrightarrow{\text{Bayesian Inference}} p(\alpha, \beta | y)$$





Poisson Regression



Poisson Regression

It is easy to generalize this model by parametrizing $\log \lambda_\tau$ using a nonlinear function:

$$\log \lambda_\tau = f(.; w)$$

Poisson Regression

It is easy to generalize this model by parametrizing $\log \lambda_\tau$ using a nonlinear function:

$$\log \lambda_\tau = f(.; w)$$

For example in the previous session, we added a nonlinear term for handling seasonality:

$$\log \lambda_\tau = \alpha\tau + \nu \cos(\omega\pi(\tau - \phi)) + \beta$$

Poisson Regression

It is easy to generalize this model by parametrizing $\log \lambda_\tau$ using a nonlinear function:

$$\log \lambda_\tau = f(.; w)$$

For example in the previous session, we added a nonlinear term for handling seasonality:

$$\log \lambda_\tau = \alpha\tau + \nu \cos(\omega\pi(\tau - \phi)) + \beta$$

We can also use more expressive functions like a neural network if necessary.
Let's discuss **Bayesian Neural Network** first for a classification problem and then for our time series example.

Bayesian Neural Network

Let $f(x; \boldsymbol{w})$ be a neural network with three layers and sigmoid activation:

$$\boldsymbol{w} = [\boldsymbol{w}^{L1}, \boldsymbol{w}^{L2}, \boldsymbol{w}^{L3}]$$

$$f(x; \boldsymbol{w}) = \sigma(\sigma(\sigma(x\boldsymbol{w}^{L1})\boldsymbol{w}^{L2})\boldsymbol{w}^{L3})$$

Bayesian Neural Network

Let $f(x; \boldsymbol{w})$ be a neural network with three layers and sigmoid activation:

$$\boldsymbol{w} = [\boldsymbol{w}^{L1}, \boldsymbol{w}^{L2}, \boldsymbol{w}^{L3}]$$

$$f(x; \boldsymbol{w}) = \sigma(\sigma(\sigma(x\boldsymbol{w}^{L1})\boldsymbol{w}^{L2})\boldsymbol{w}^{L3})$$

We define a Multivariavte Normal prior over $\boldsymbol{w}$

$$\boldsymbol{w} \sim \text{Normal}(0, I)$$

Bayesian Neural Network

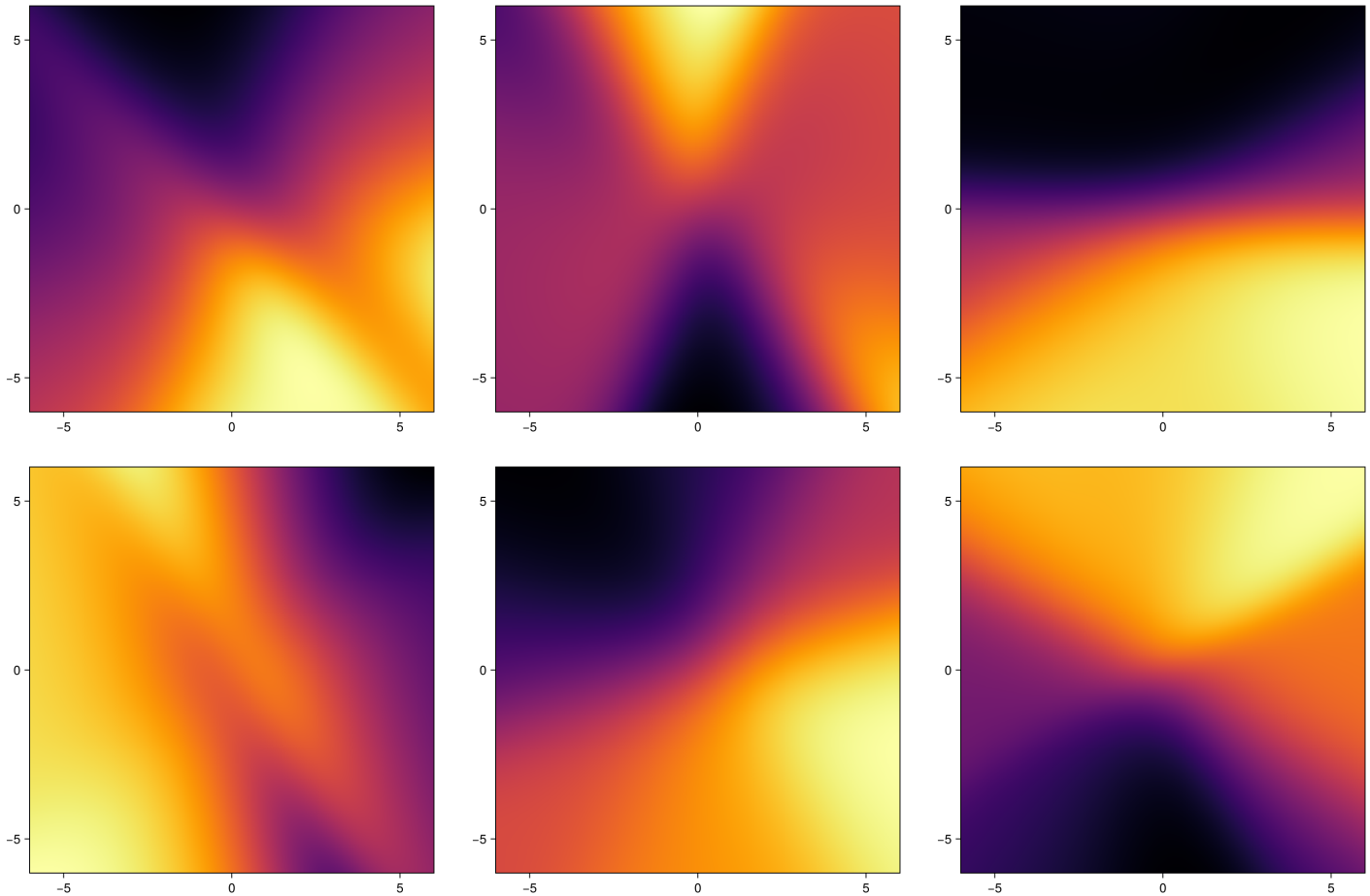
```julia
function neural_network(X, L1, L2, L3, H_dim)
    H = X * reshape(L1, (2, H_dim))
    H = sigmoid.(H)

    H = H * reshape(L2, (H_dim, H_dim))
    H = sigmoid.(H)

    O = H * reshape(L3, (H_dim, 1))
    O = sigmoid.(O)

    return O
end
```

Bayesian Neural Network

```julia
@model function bayesian_neural_network(X, y, σ, H_dim)
    n_L1 = 2 * H_dim
    n_L2 = H_dim * H_dim
    n_L3 = H_dim

    Σ(n) = Diagonal(abs2.(σ .* ones(n)))

    L1 ~ MvNormal(zeros(n_L1), Σ(n_L1))
    L2 ~ MvNormal(zeros(n_L2), Σ(n_L2))
    L3 ~ MvNormal(zeros(n_L3), Σ(n_L3))

    O = neural_network(X, L1, L2, L3, H_dim)

    for i in eachindex(y)
        y[i] ~ Bernoulli(O[:][i])
    end
end
```

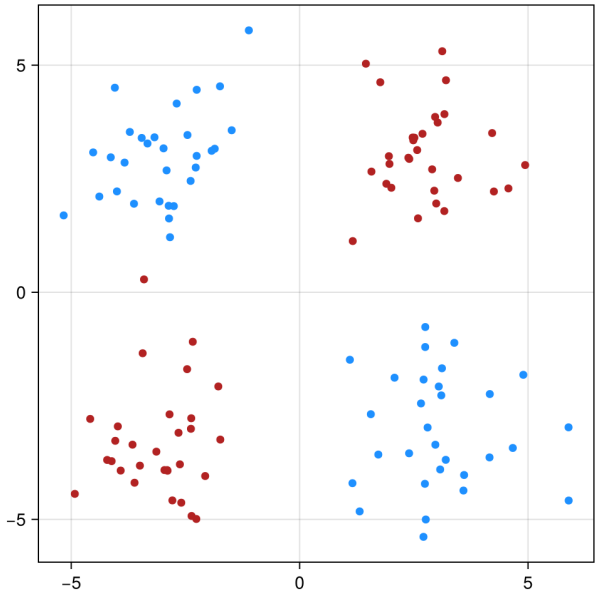
Bayesian Neural Network

Some samples from $p(\boldsymbol{w})$



Bayesian Neural Network

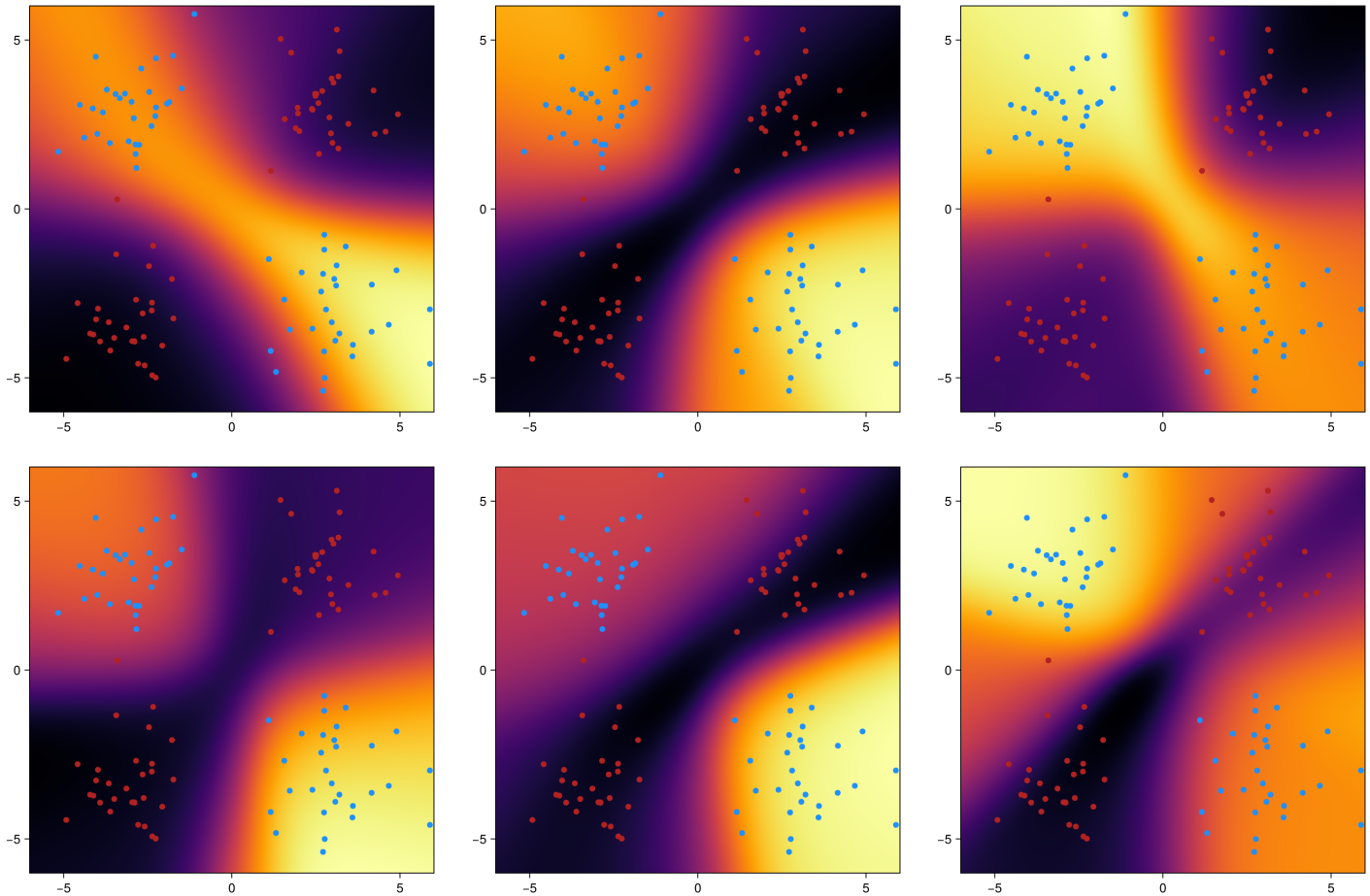
Now we update our belief about network weights using the below dataset



$$y \sim \text{Bernoulli}(f(x; \boldsymbol{w}))$$

Bayesian Neural Network

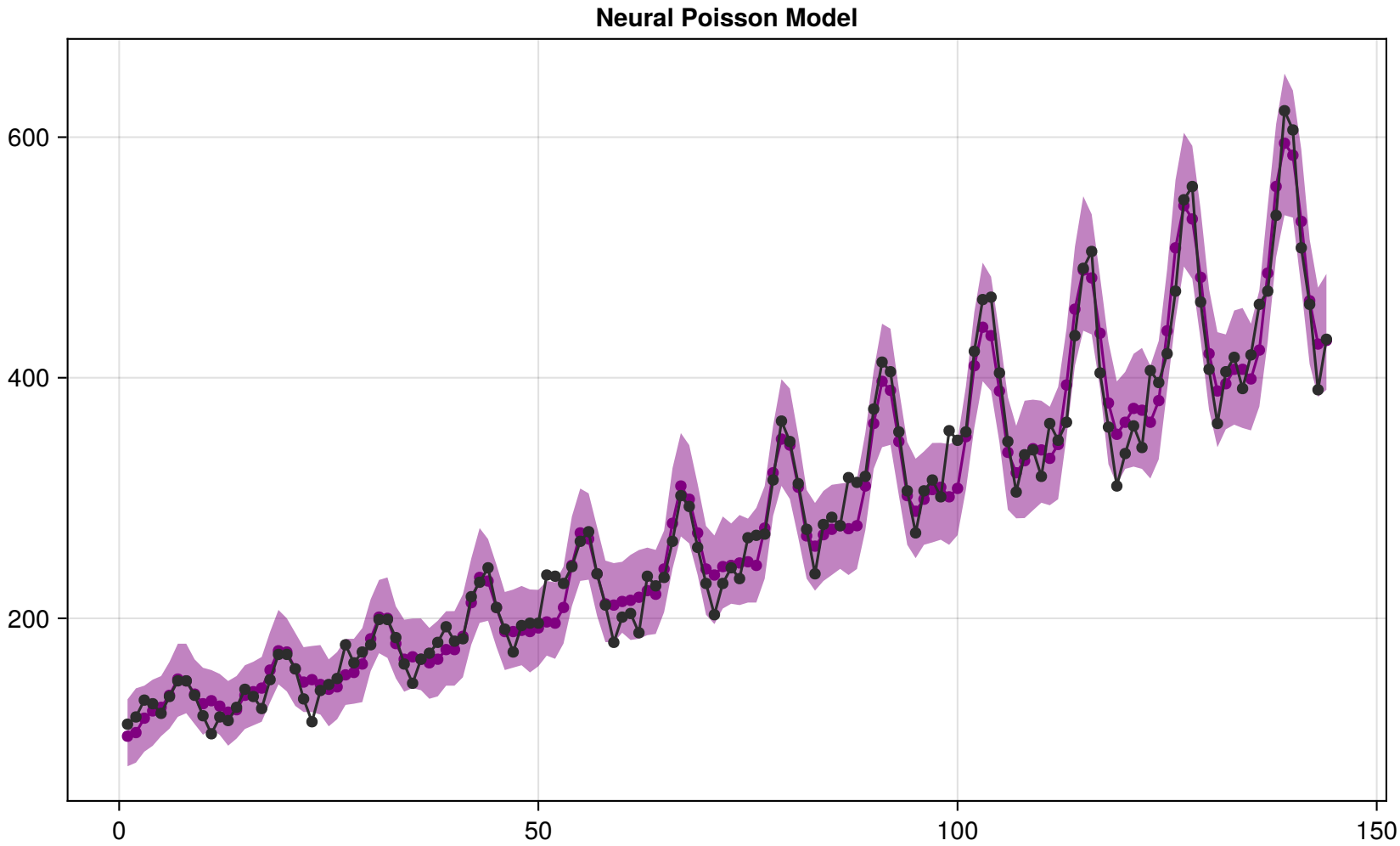
Some samples from $p(\boldsymbol{w}|y)$



Neural Poisson Regression

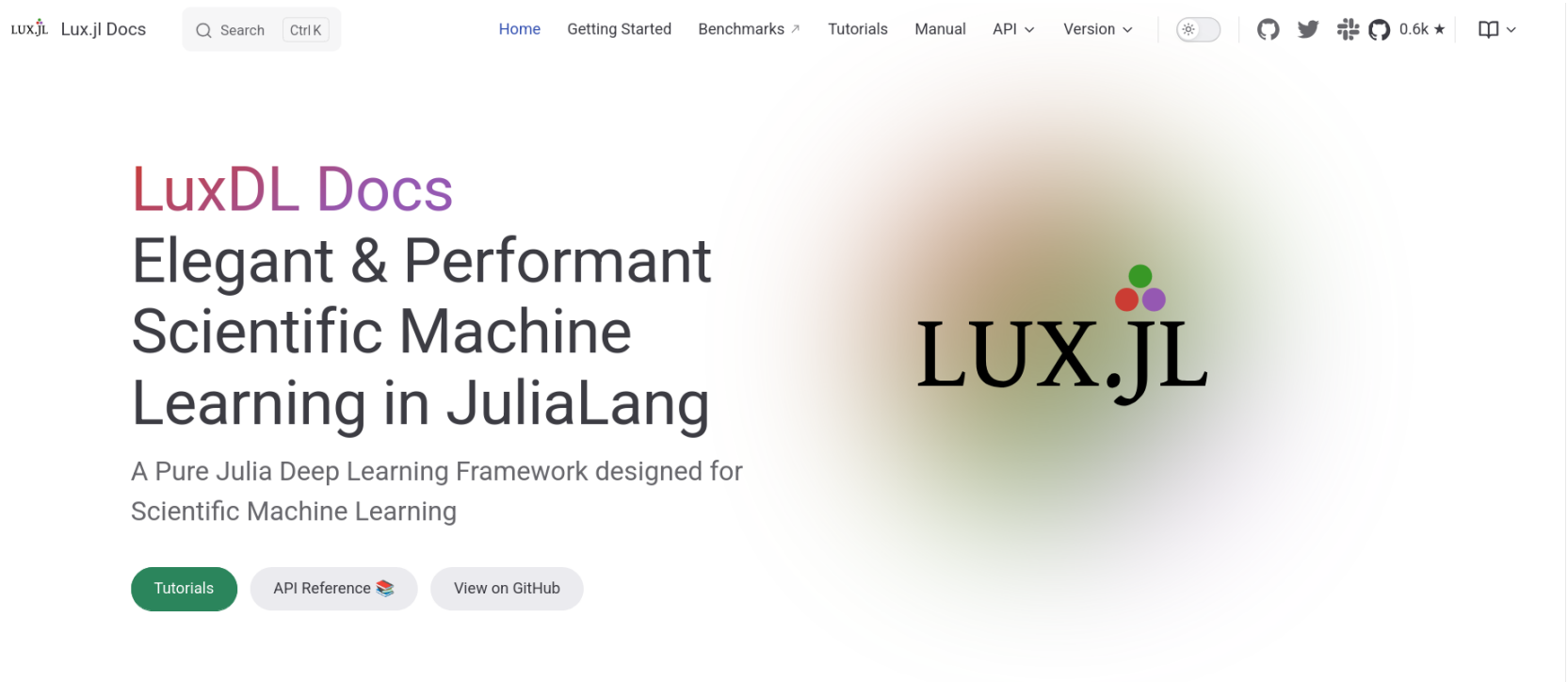
```julia
@model function neural_poisson_model(y)
  w ~ MvNormal(zeros(12), I)
  L1 = reshape(w[1:8], (4,2))
  L2 = reshape(w[9:12], (1,4))
  b = w[12:12]

  ϕ ~ Normal(0.0, 2.0)
  ω ~ Normal(0.0, 2.0)

  for t in 1:length(y)
    x = [t, cos(ω*(t-ϕ))]
    log_λ = ANN(L1, L2, b, x)[1]
    y[t] ~ Poisson(exp(log_λ))
  end
end
```

Neural Poisson Regression



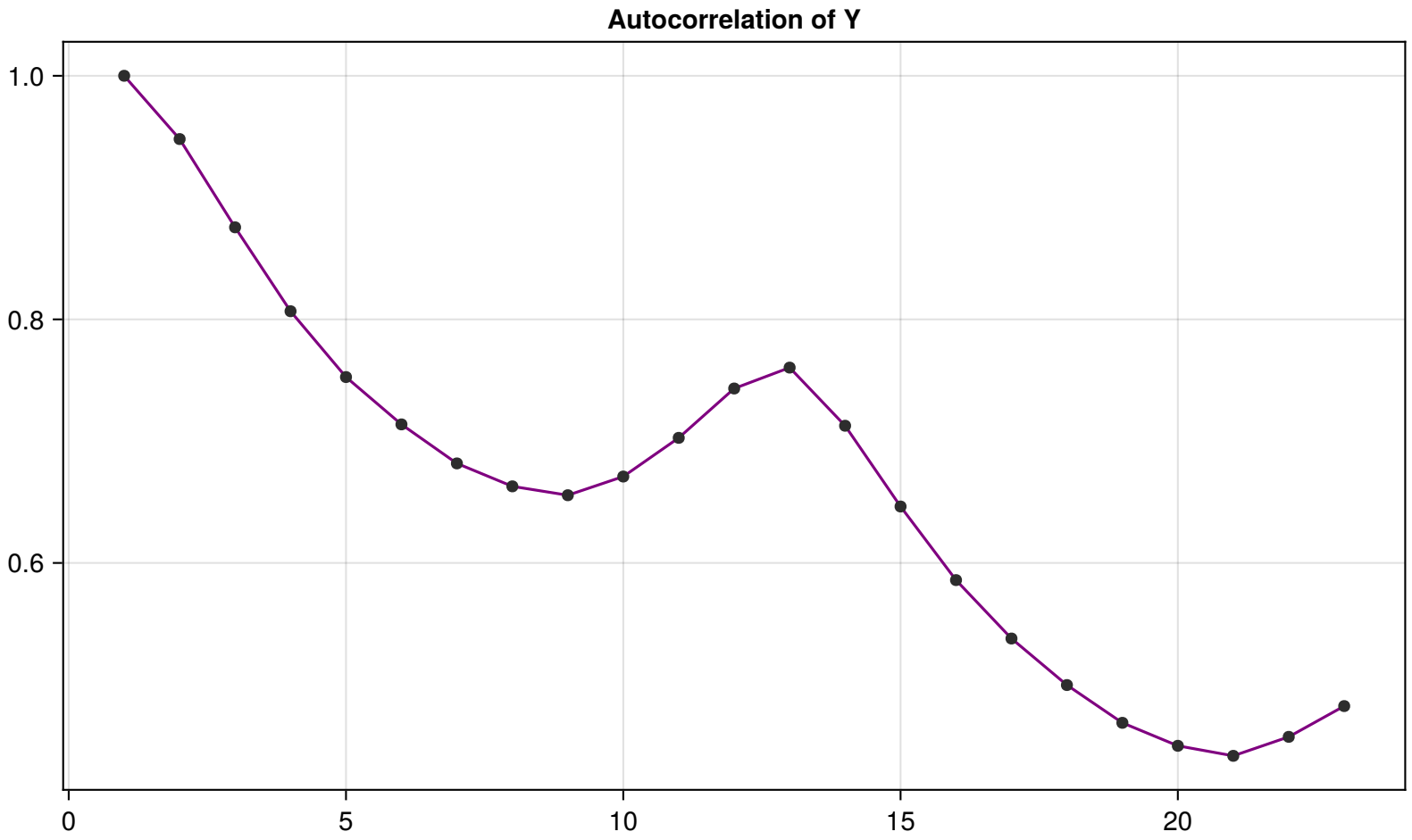
Neural Poisson Regression

Of course, we don't have to manually define neural networks



Autoregressive Models

A popular assumption in modeling time series data is **Autoregression**

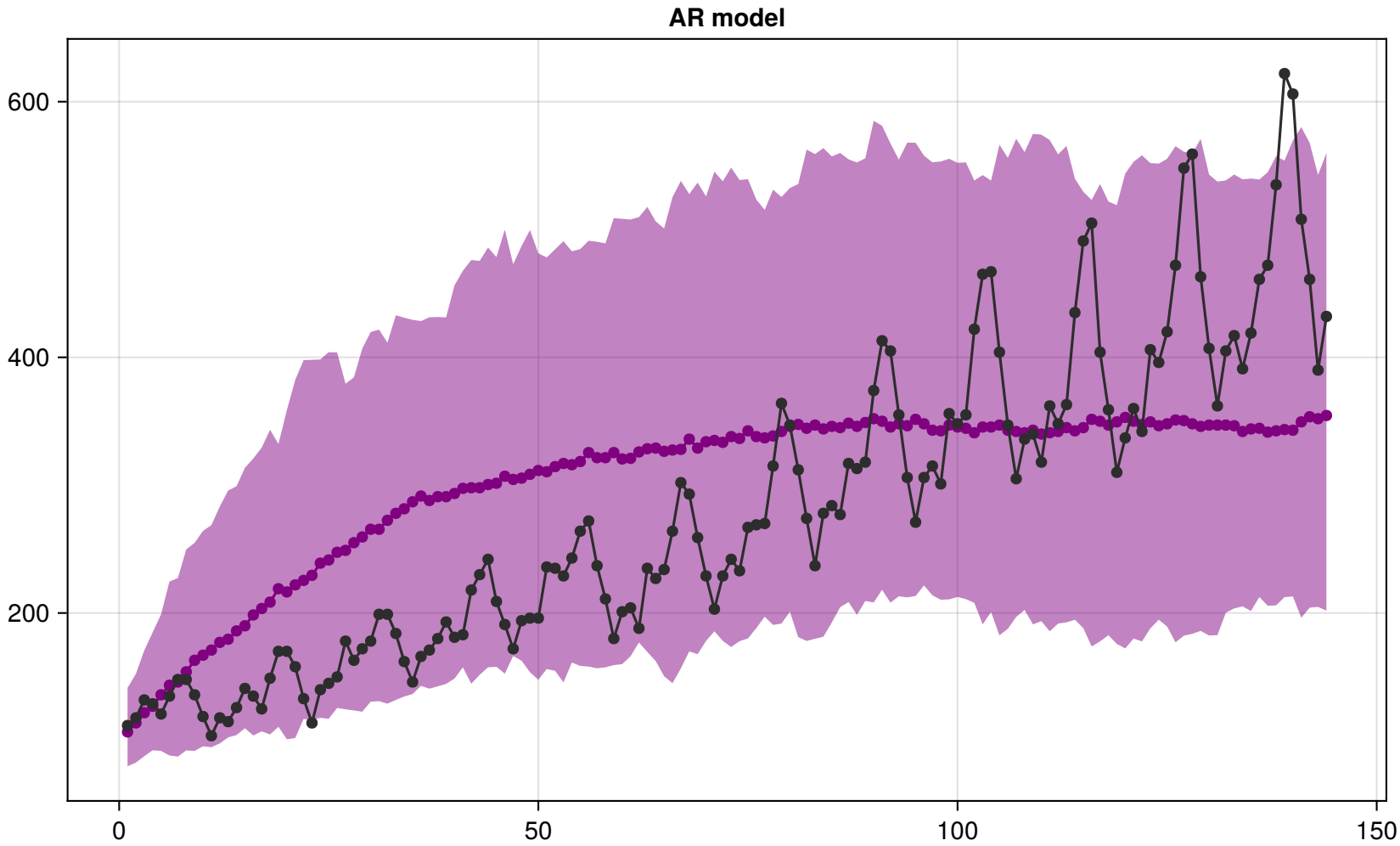


Autoregressive Models

$$
\begin{aligned}
Y_\tau &\sim \text{Poisson}(\lambda_\tau) \\
\log \lambda_\tau &= \alpha \log \lambda_{\tau-1} + \beta \\
\alpha, \beta &\sim \text{Normal}(0, 1) \\
\lambda_0 &\sim \text{Poisson}(100.0)
\end{aligned}
$$

```
@model function AR(y)
  α ~ Normal(0.0, 2.0)
  β ~ Normal(0.0, 2.0)
  y0 ~ filldist(Poisson(100.0), 1)

  for t in 1:length(y)
    log_λ = α * log(lag(y, y0, t-1)) + β
    λ = clamp(exp(log_λ), 100.0, 1000.0)
    y[t] ~ Poisson(λ)
  end
end
```

Autoregressive Models

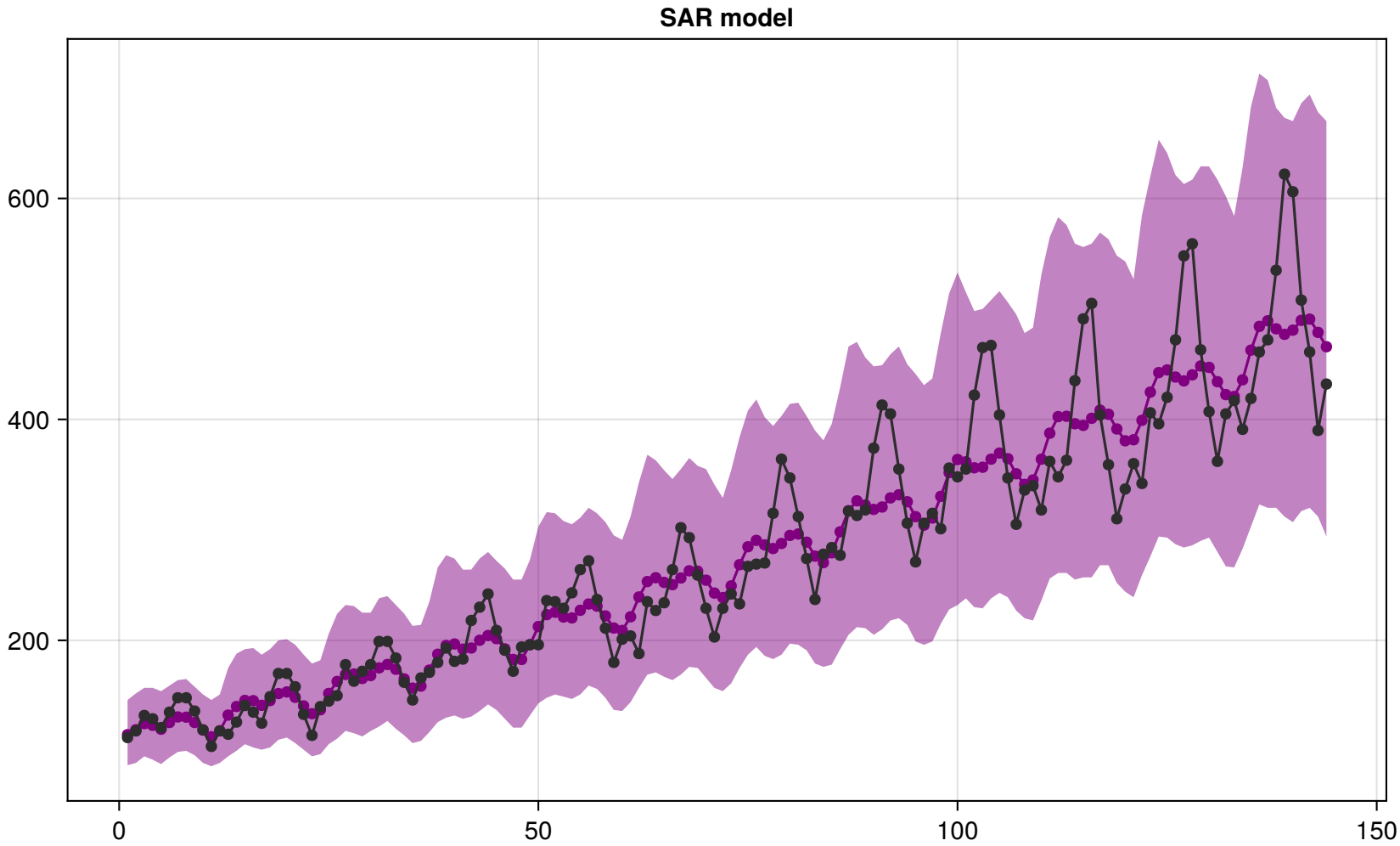


Autoregressive Models

$$Y_\tau \sim \text{Poisson}(\lambda_\tau)$$
$$\log \lambda_\tau = \alpha_1 \log \lambda_{\tau-1} + \alpha_2 \log \lambda_{\tau-2} + \alpha_3 \log \lambda_{\tau-12} + \beta$$

```
@model function SAR(y)
  α1 ~ Normal(0.0, 2.0)
  α2 ~ Normal(0.0, 2.0)
  α3 ~ Normal(0.0, 2.0)
  β ~ Normal(0.0, 2.0)
  y0 ~ filldist(Poisson(100.0), 12)

  for t in 1:length(y)
    log_λ = α1 * log(lag(y, y0, t-1)) +
            α2 * log(lag(y, y0, t-2)) +
            α3 * log(lag(y, y0, t-12)) + β
    λ = clamp(exp(log_λ), 100.0, 1000.0)
    y[t] ~ Poisson(λ)
  end
end
```

Autoregressive Models



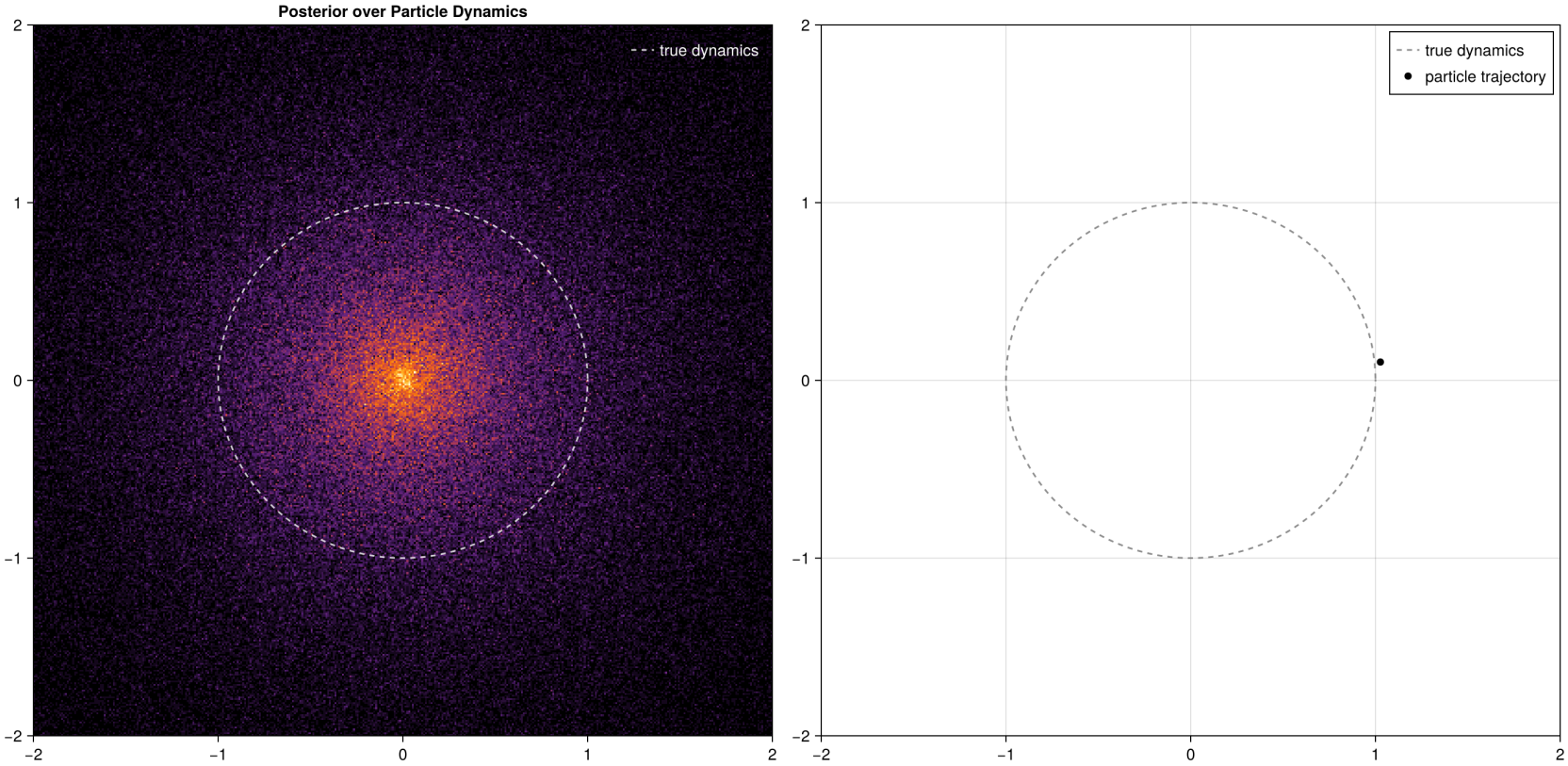
Autoregressive Models

Generalizing to a multivariate scenarios is straightforward:

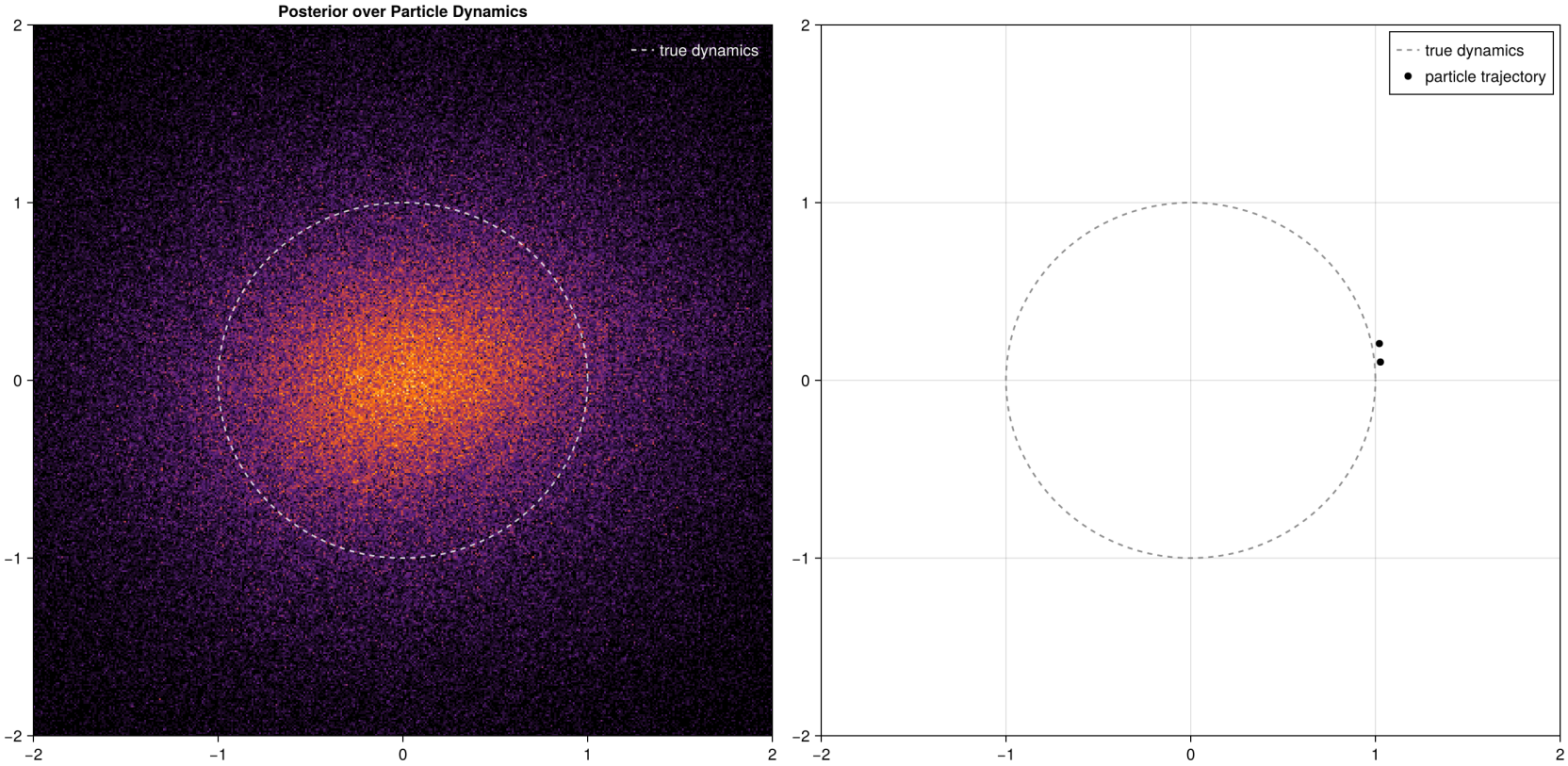
$$X_t \sim \text{MvNormal}(AX_{t-1}, \sigma I)$$
$$A \sim \text{MvNormal}(0, I)$$

```
@model function VAR(x)
    A ~ MvNormal(zeros(4), I)
    σ ~ Exponential(1.0)

    for t in 2:length(x)
        x[t] ~ MvNormal(reshape(A, (2,2)) * x[t-1], σ * I(2))
    end
end
```

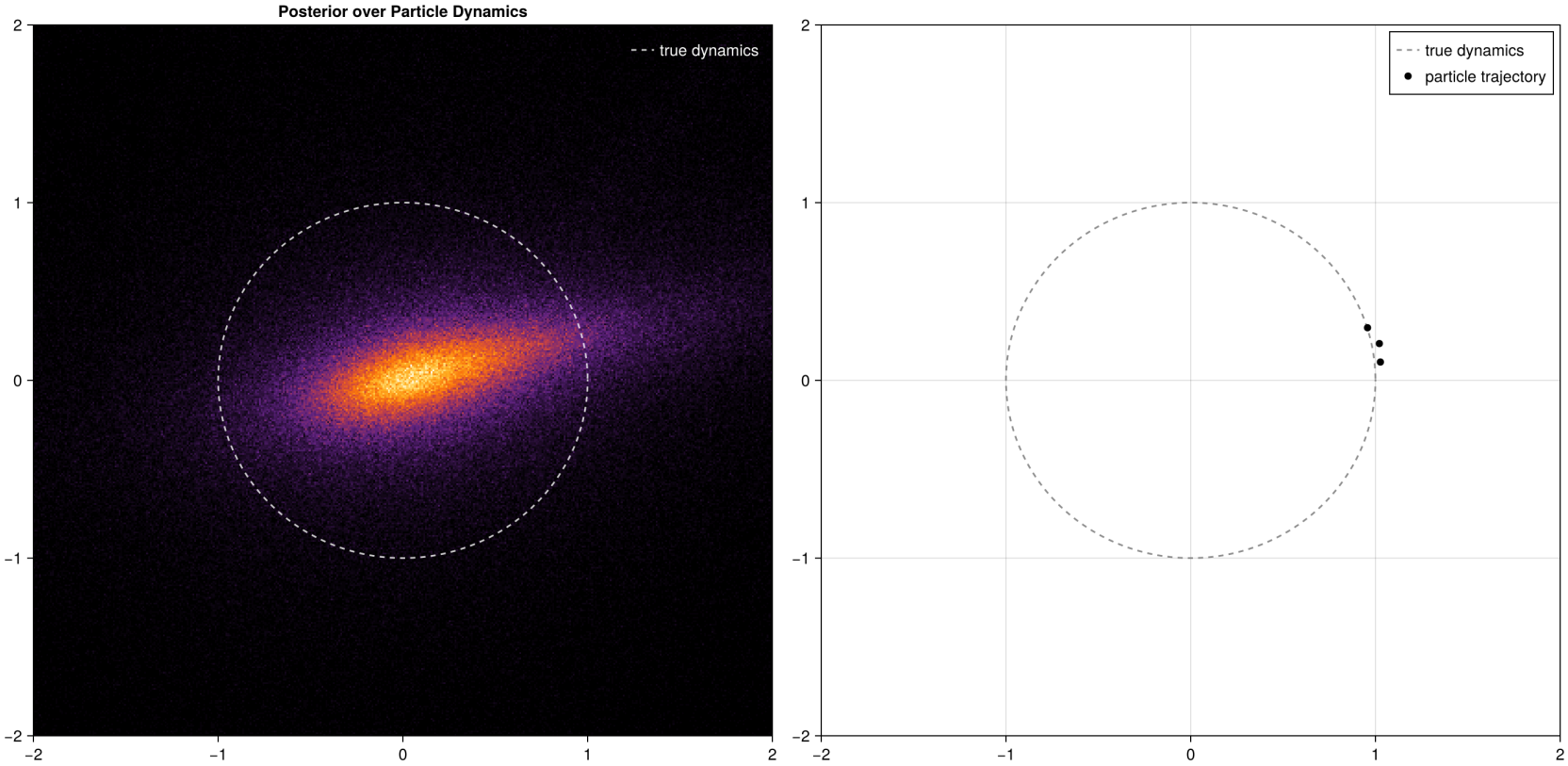
Autoregressive Models



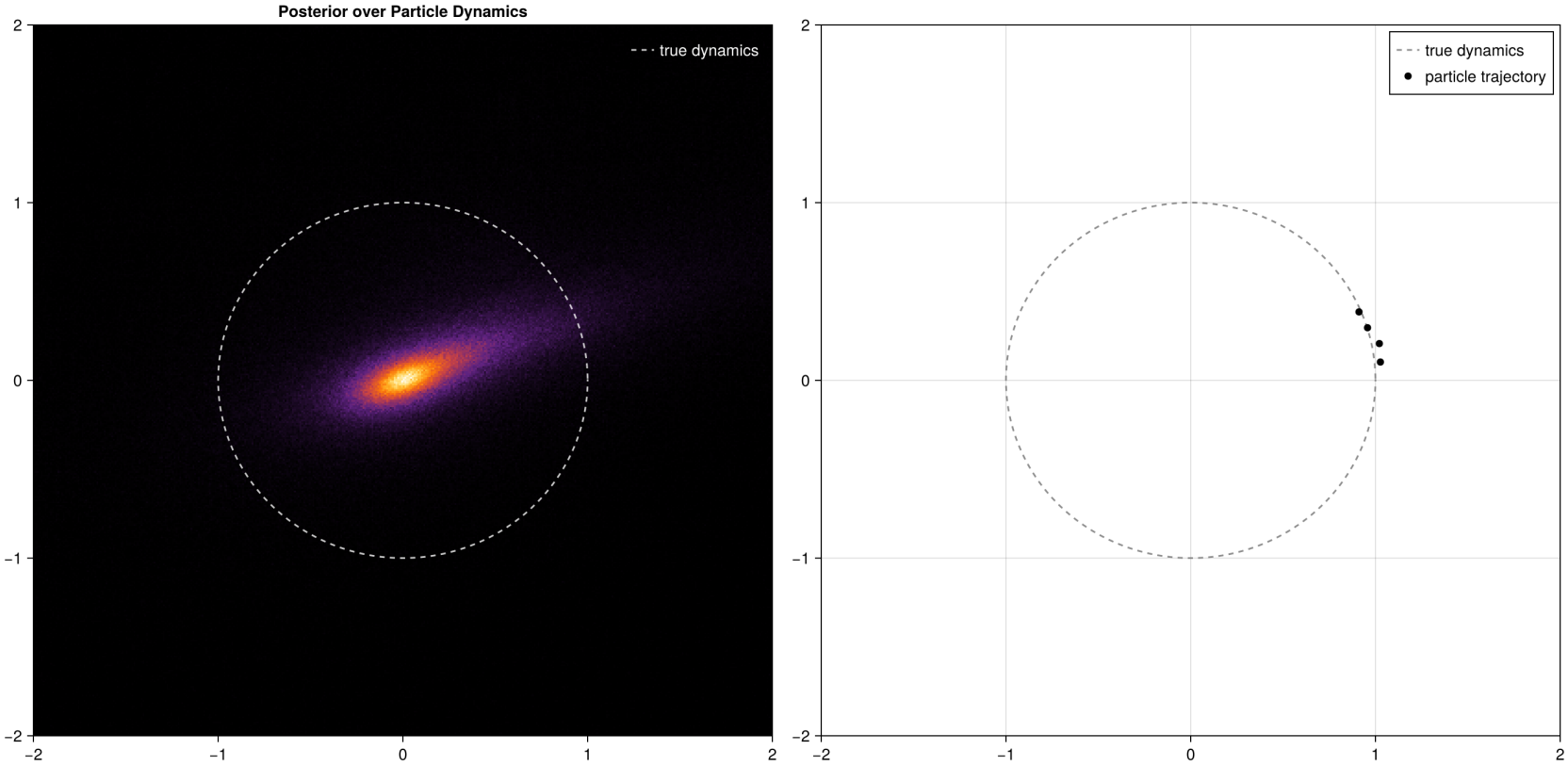
Autoregressive Models



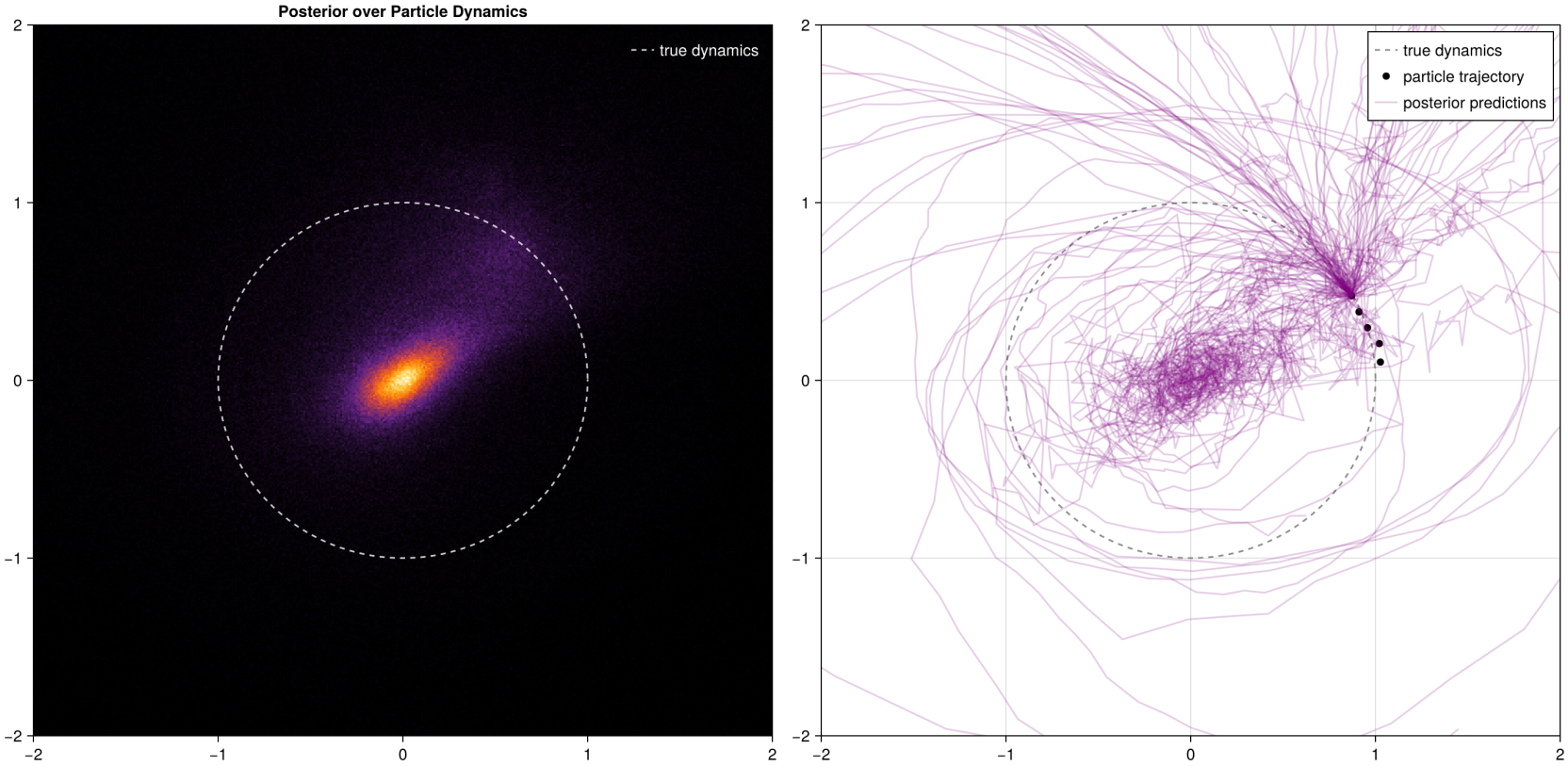
Autoregressive Models



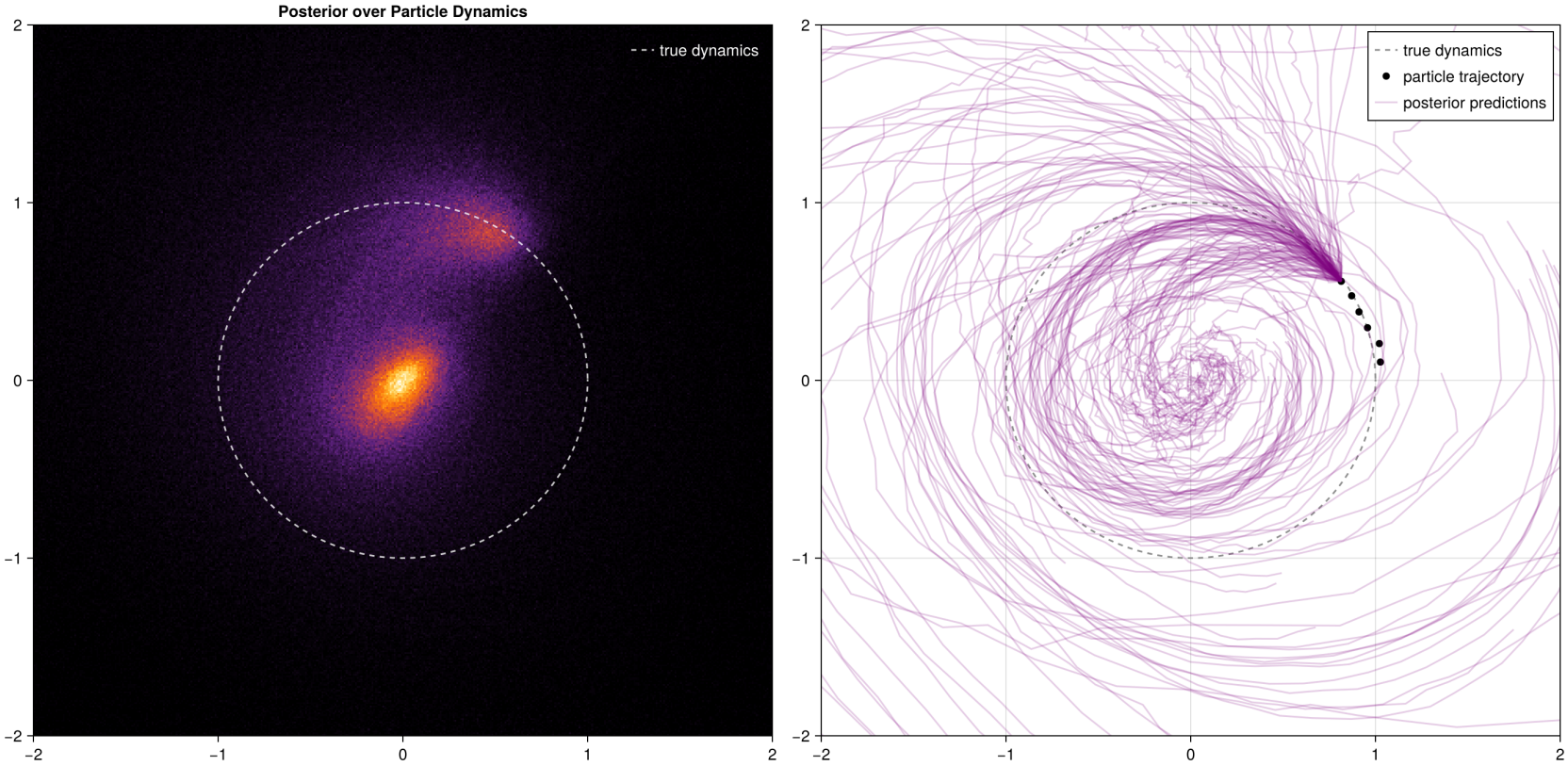
Autoregressive Models



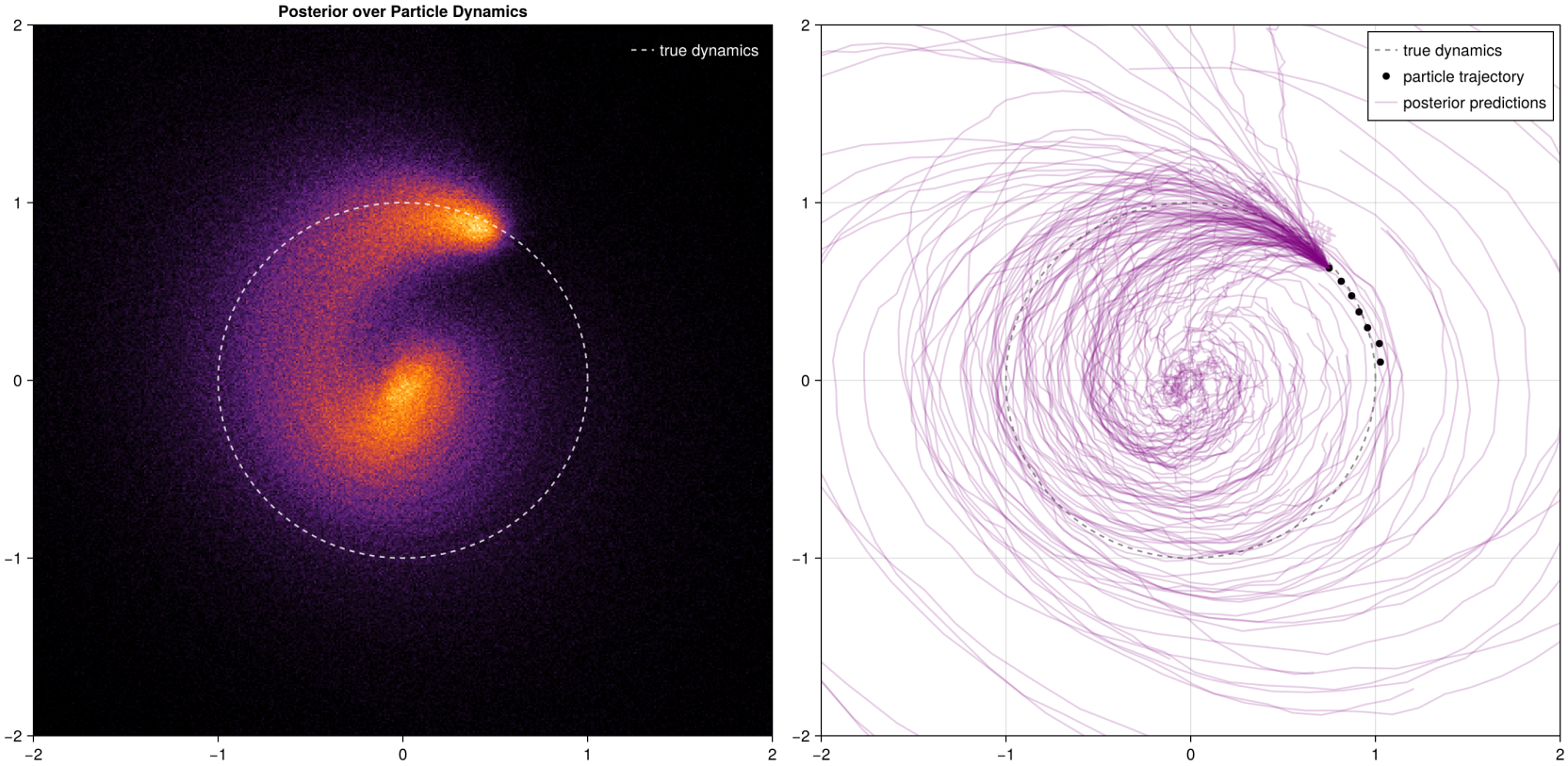
Autoregressive Models



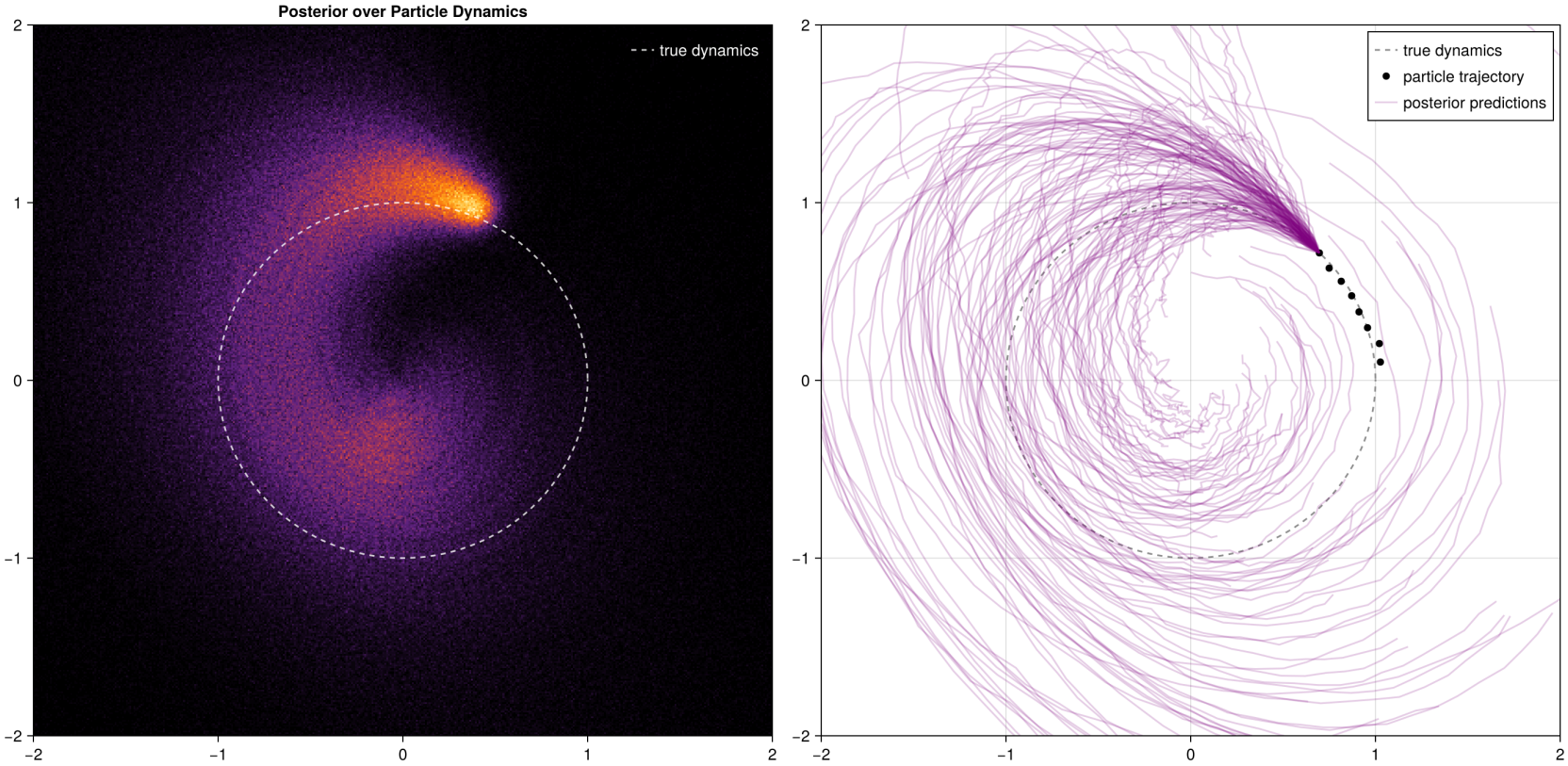
Autoregressive Models



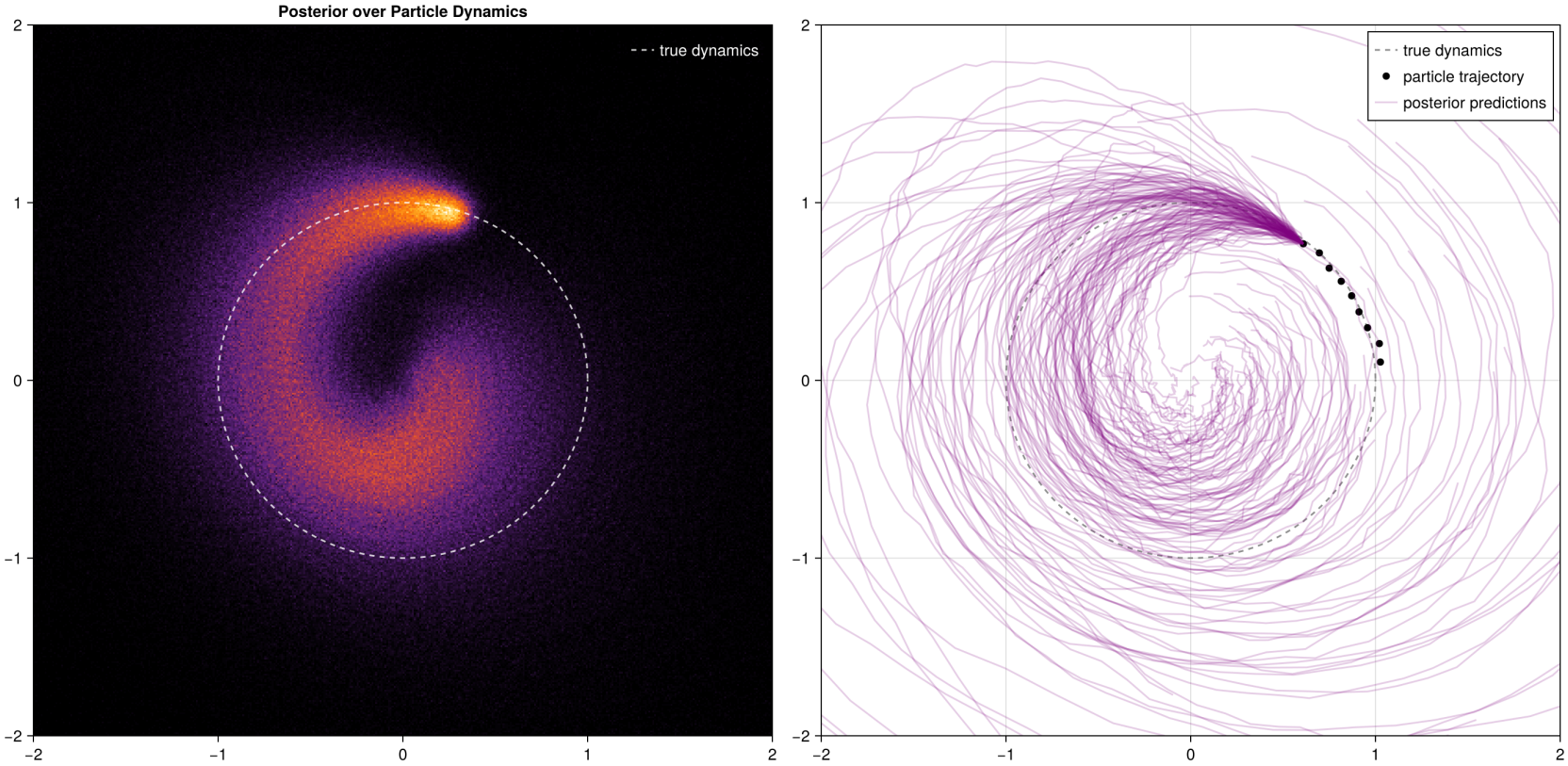
Autoregressive Models



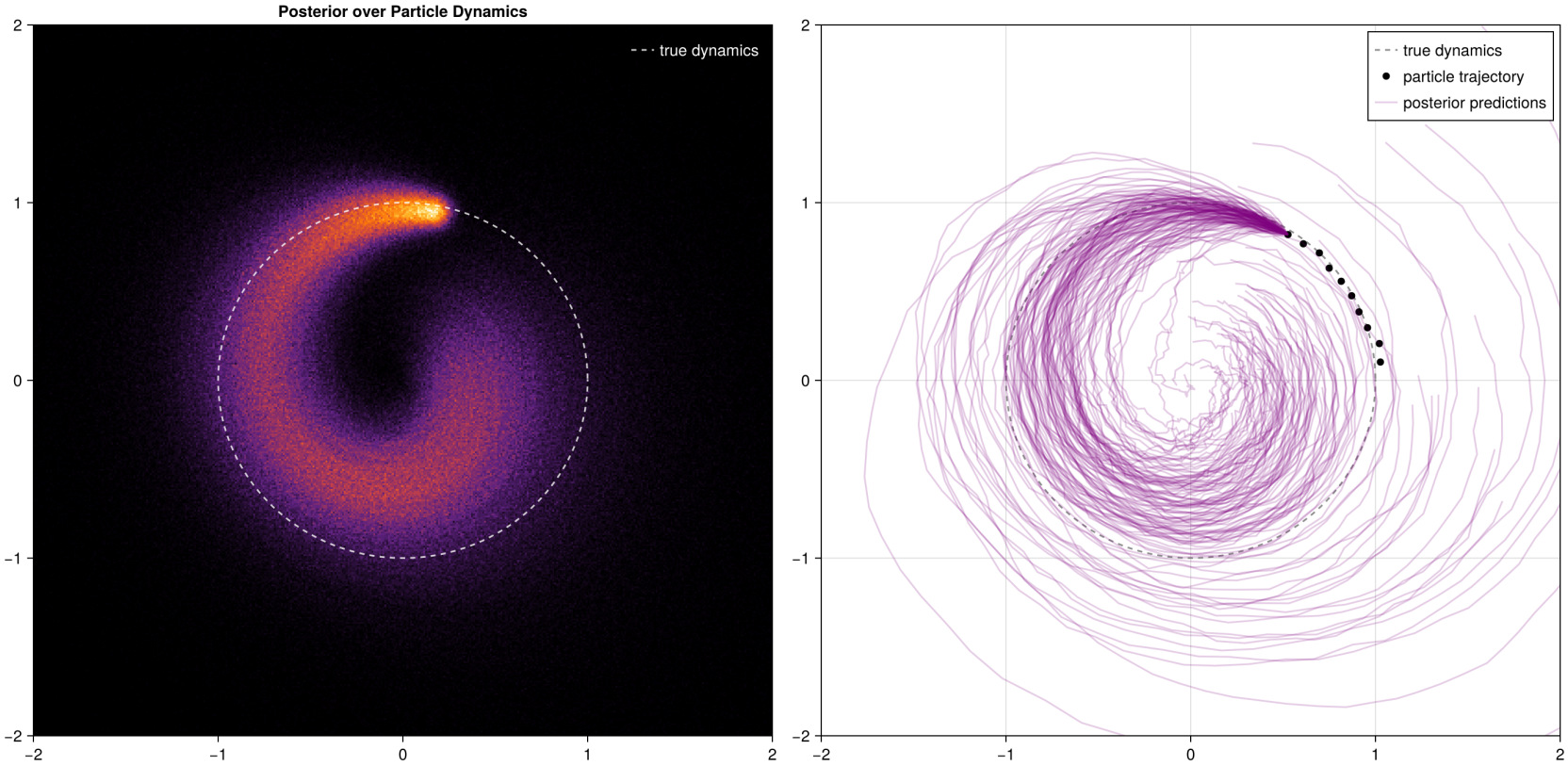
Autoregressive Models



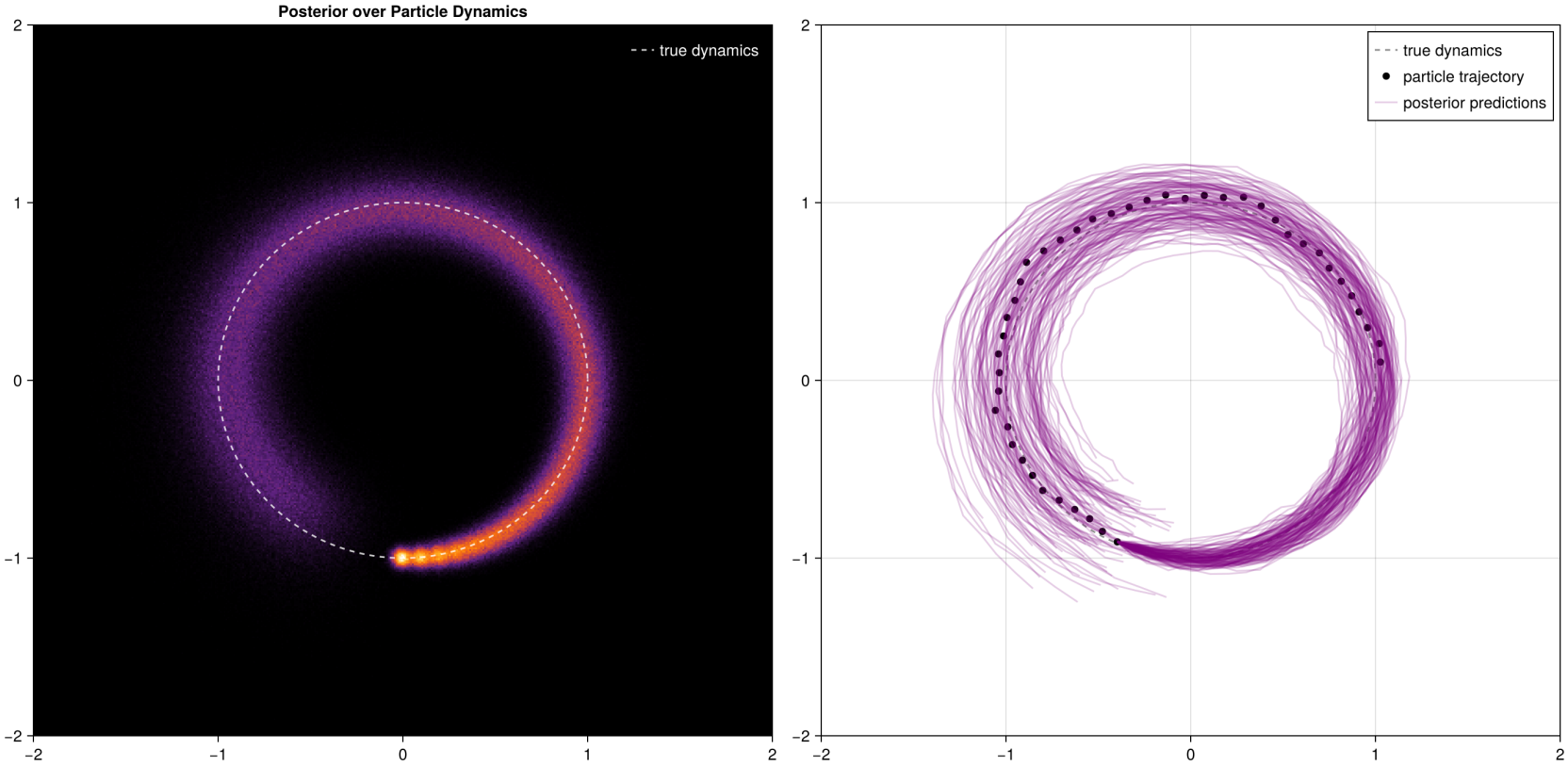
Autoregressive Models



Autoregressive Models



Autoregressive Models



Latent ODE

Autoregressive models and in general Dynamic Bayesian Networks are discrete-time models.

Can we find a continuous-time model which is computationally tractable?

Bayesian Differential Equations

Latent ODE

Remember our linear Poisson model $\log \lambda_\tau = \alpha\tau + \beta$. A similar model can be described by the following ODE with an unknown parameter and initial value:

$$\begin{aligned}
\frac{d \log \lambda}{d\tau} &= \alpha \\
\log \lambda(1) &= \beta \\
\alpha, \beta &\sim \text{Normal}(0, 1)
\end{aligned}$$

We can solve ODEs in a probabilistic program and infer the unknown parameters!

Latent ODE

SciML: Open Source Software for Scientific Machine Learning

- Advanced differential equation solvers
- Differentiable ODE solvers compatible with Turing

Latent ODE

Install DifferentialEquations.jl:

```
Pkg.add("DifferentialEquations.jl")
```

Define the rate dynamics:

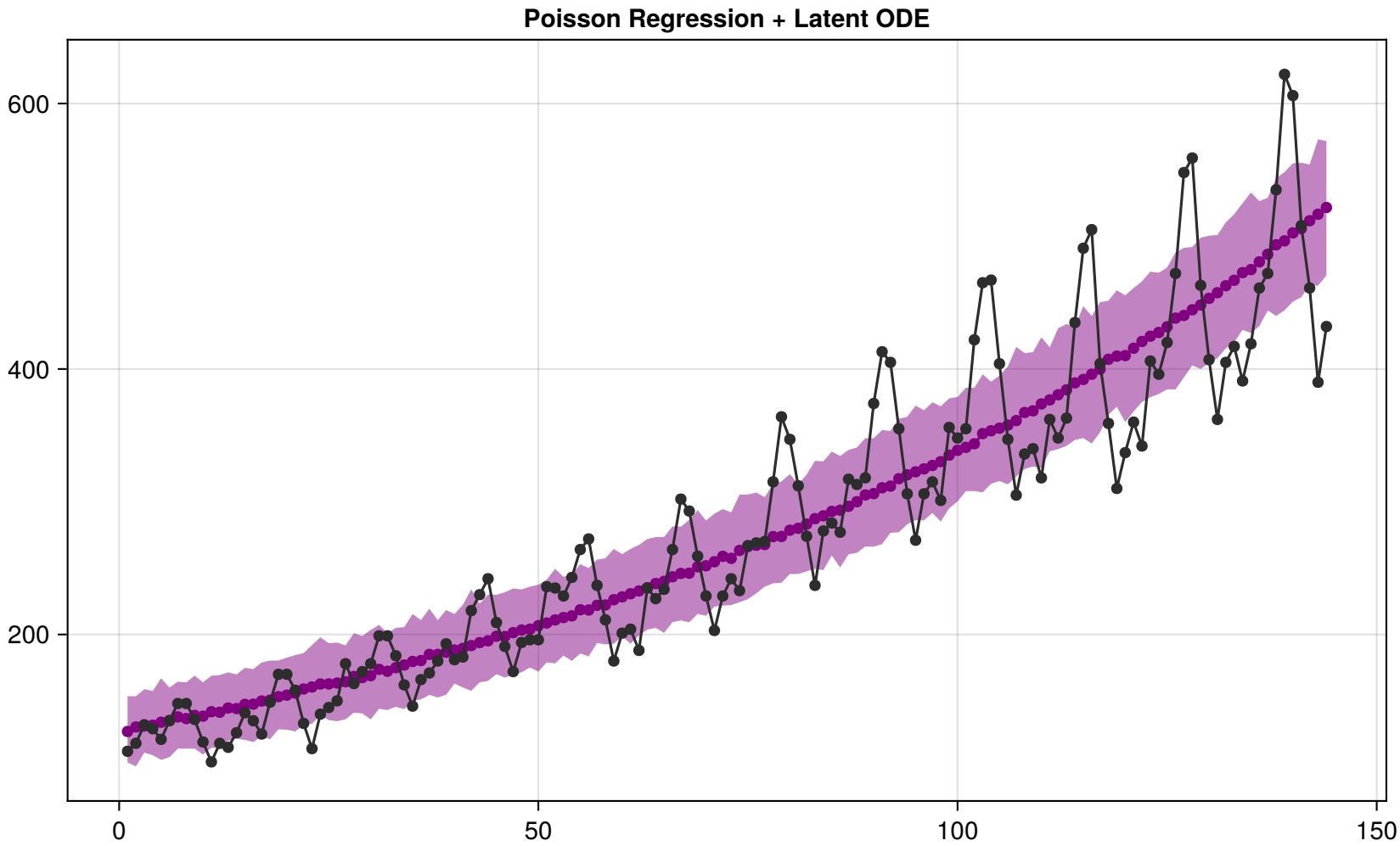
```
function rate_dynamics(dlogλ, logλ, p, t)
    α = p[1]
    dlogλ[1] = α
end
```

Latent ODE

Now we can use the ODE in a probabilistic program:

```
@model function latent_ODE_poisson(y)
    α ~ Normal(0.0, 2.0)
    β ~ Normal(0.0, 2.0)

    # solve the ODE for λ
    logλ0 = [β]
    tspan = (1.0, length(y))
    p = [α]
    prob = ODEProblem(rate_dynamics, logλ0, tspan, p)

    logλ = solve(prob, Tsit5(); saveat=1.0)

    for τ in 1:length(y)
        λ = clamp(exp(logλ[τ][1]), 100.0, 1000.0)
        y[τ] ~ Poisson(λ)
    end
end
```

Latent ODE



Latent ODE

Next we try a system of ODEs with latent parameters

$$\frac{du}{d\tau} = \alpha_1$$
$$\frac{dz_1}{d\tau} = \alpha_2 z_2(\tau) + \alpha_3$$
$$\frac{dz_2}{d\tau} = \alpha_4 z_1(\tau) + \alpha_5$$
$$\lambda(\tau) = e^{u(\tau)} + e^{z_1(\tau)}$$

```
function rate_dynamics(dlogλ, logλ, p, t)
    α1, α2, α3, α4, α5 = p
    dlogλ[1] = α1
    dlogλ[2] = α2 * logλ[3] + α3
    dlogλ[3] = α4 * logλ[2] + α5
end
```

Latent ODE