Bayesian Learning for Uncertainty Quantification and Decision Making

Mattias Villani 👱‍♂️

Department of Statistics
Stockholm University

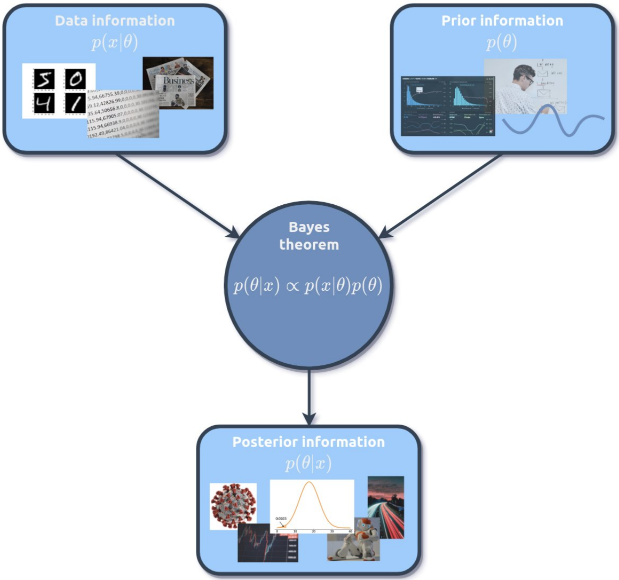
mattiasvillani.com @matvil @matvil mattiasvillani

Overview

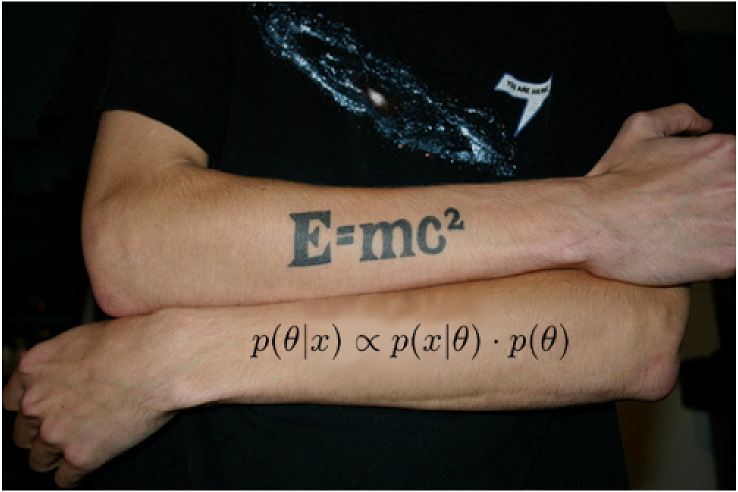
- **The Bayesics**
- **Prediction**
- **Decision making**
- **Regularization and Bayes**
- **Posterior simulation and approximation**
- **Probabilistic programming languages for Bayes**

- **Slides**: http://mattiasvillani.com/news.

The Bayesics



Great theorems make great tattoos



Am I really getting my 20Mbit/sec?

- I have a 50Mbit/sec internet connection.
- ISP promises at least 20Mbit/sec on average.
- **Data**: $\mathrm{x} = (15.77, 20.5, 8.26, 14.37, 21.09)$ Mbit/sec.
- **Measurement errors**: $\sigma = 5$ (± 10Mbit with 95% probability)
- **Data model**

$$X_1, ..., X_n | \theta \overset{\text{iid}}{\sim} \mathrm{N}(\theta, \sigma^2)$$

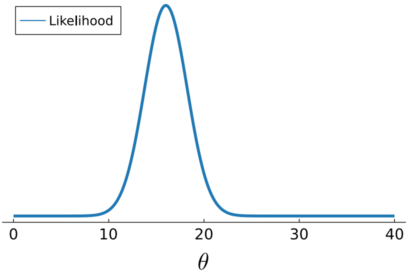
Likelihood function

- **Likelihood function**

$$X_1, ..., X_n | \theta \overset{\text{iid}}{\sim} \mathrm{N}(\theta, \sigma^2)$$

viewed as a function of θ, for **observed data** $x_1, ..., x_n$.

- The likelihood is **proportional** to a $\mathrm{N}(\bar{x}, \sigma^2/n)$ density.

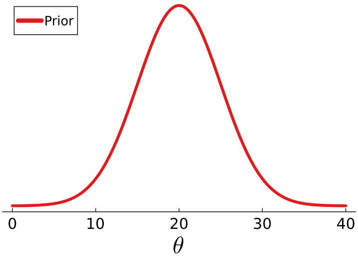


- The likelihood is **not** a probability density for θ. 😢
- But wait, how **could** it be? θ is not random! 🤔

Subjective probability and Bayes! 🎤

- Bayesian learning is based on **subjective probability**.
- Probability as subjective **degrees of belief**.
- All **unknowns** should be quantified by subjective probability.
- 🧔 "$\Pr(10\text{th decimal of } \pi \text{ is } 5) = 0.1$"
- 🤓 "$\Pr(10\text{th decimal of } \pi \text{ is } 5) = 1.0$"
- **Prior distribution**

$$\theta \sim \mathrm{N}(\mu_0, \tau_0^2)$$



Normal data, known variance - normal prior

- **Posterior distribution**

$$\theta | x_1, ..., x_n \sim \mathrm{N}(\mu_n, \tau_n^2)$$

- **Posterior mean**

$$\mu_n = w\bar{x} + (1 - w)\mu_0$$

with weight on data

$$w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}} = \frac{\text{data info}}{\text{data info} + \text{prior info}}$$

- **Posterior variance**

$$\tau_n^2 = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$

Interactive - Bayes for Gaussian iid model

Data:

n 5

σ 5

Observations used: $\boldsymbol{x} = (15.77, 20.5, 8.26, 14.37, 21)$

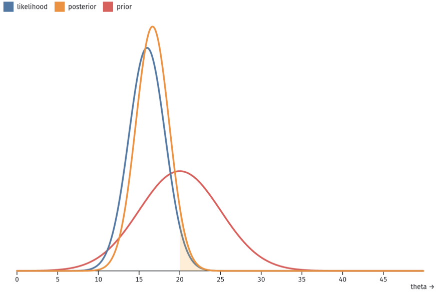
Prior:

μ_0 20

τ_0 5

quantile posterior 20

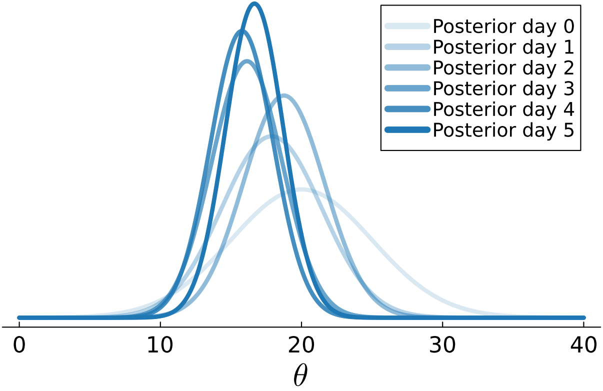
Posterior probability: $P(\theta \geq 20 | \boldsymbol{x}) = 0.0504$



Prior-to-Posterior mapping. The likelihood is normalized.

Bayesian online learning

- Yesterday's posterior is today's prior.



Bayesian Prediction

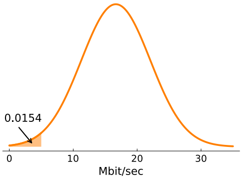
- **Predictive distribution** averages over the unknown parameter

$$\underbrace{p(x_{n+1}|x_{1:n})}_{\text{predictive dist}} = \int \underbrace{p(x_{n+1}|\theta)}_{\text{model}} \underbrace{p(\theta|x_{1:n})}_{\text{posterior}} d\theta$$

- Monte Carlo integration:
  - draw θ from posterior $p(\theta|x_{1:n})$
  - draw x_{n+1} from model $p(x_{n+1}|\theta)$ given that θ
- Normal data, normal prior:

$$x_{n+1}|x_{1:n} \sim N(\mu_n, \sigma^2 + \tau_n^2)$$

- My streaming buffers whenever $x < 5$ Mbit/Sec.



Decision making under uncertainty

- Let $a \in \mathcal{A}$ be an **action**.
  - Example 1: size of energy tax.
  - Example 2: central bank's interest rate.
  - Example 3: resolution at a given battery profile.
- Let θ be an **unknown quantity**.
  - Example 1: Global temperature at year X.
  - Example 2: Inflation Y quarters ahead.
  - Example 3: Gamer's reaction to lowered resolution to save battery.
- Choosing action a when state of nature is θ gives **utility**

$$U(a, \theta)$$

Decision Theory

- The eternal Umbrella decision:

	Rain	Sun
No umbrella	-50	50
Umbrella	10	30

- How much stock should a company keep?
  - θ is the number of items demanded of a product
  - a is the number of items in stock

$$U(a,\theta) = \begin{cases} p \cdot \theta - c_1(a-\theta) & \text{if } a > \theta \text{ [too much stock]} \\ p \cdot a - c_2(\theta - a)^2 & \text{if } a \leq \theta \text{ [too little stock]} \end{cases}$$

Optimal Bayesian decisions

- Ad hoc decision rules:
  - *Minimax*. Minimizes the maximum loss.
  - *Minimax-regret* ... 😴

- **Bayesian theory**: maximize **posterior expected utility** 😍

$$a_{bayes} = \text{argmax}_{a \in \mathcal{A}} \; E_{p(\theta|y)}[U(a, \theta)],$$

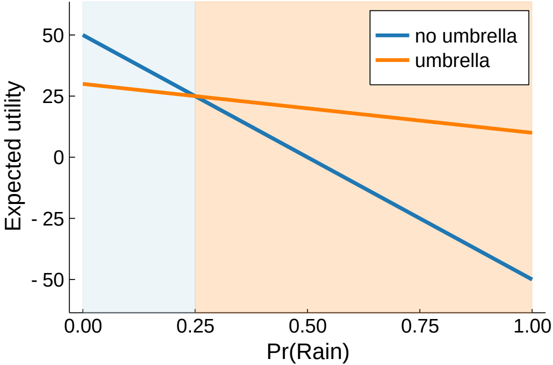
where $E_{p(\theta|y)}$ denotes the posterior expectation.

- **Separation principle**:

1. First obtain posterior $p(\theta|y)$
2. then form utility function $U(a, \theta)$ and finally
3. choose a that maximizes $E_{p(\theta|y)}[U(a, \theta)]$.

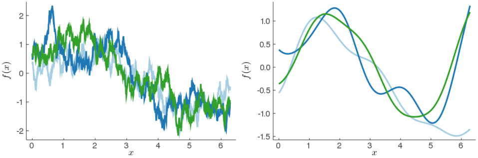
The umbrella decision

	Rain	Sun
No umbrella	-50	50
Umbrella	10	30



Regularization is a prior

- ML: regularization to avoid overfitting.
- **Regularization** can be viewed as a **Bayesian prior**.
- Ridge (**L2**) **regularization** $\Longleftrightarrow$ Normal prior on each β_j
- Lasso (**L1**) **regularization** $\Longleftrightarrow$ Laplace prior on each β_j + mode
- **Gaussian processes**: $y = f(x) + \varepsilon$, and $f(x)$ is smooth a priori.



- Regularization solves the $n < p$ problem. How is this even possible? Because we **add prior information**.

Bayesian computations

- **Sampling** from the posterior:
  - **Gibbs sampling** - when tractable usually robust. Efficiency depends on how parameters are blocked.
  - **Markov Chain Monte Carlo** (MCMC) - general purpose. Need to design a decent proposal distribution. Slow.
  - **Hamiltonian Monte Carlo** (HMC) - general purpose for continuous parameters. High-dim. Slower.
- **Particle systems** to approximate posterior:
  - **Importance sampling** - hard to make efficient when parameters are relatively high-dim.
  - Particle filters/smoothers and **Sequential Monte Carlo** (SMC) - sequential learning, state-space models.
- Posterior **approximation**
  - **Normal approximation** - Bernstein von Mises theorem. Autodiff. Fast.
  - **Variational inference** - approximate posterior by simpler distribution. Autodiff. Fast.

Random walk Metropolis algorithm

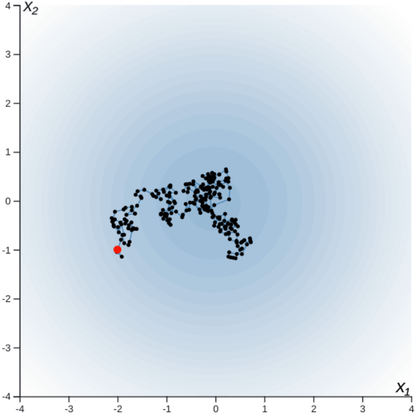
- **Initialize** $\theta^{(0)}$ and iterate for $i = 1, 2, \ldots$

  1. **Sample proposal**: $\theta_p | \theta^{(i-1)} \sim N\left(\theta^{(i-1)}, c \cdot \Sigma\right)$
  2. Compute the **acceptance probability**

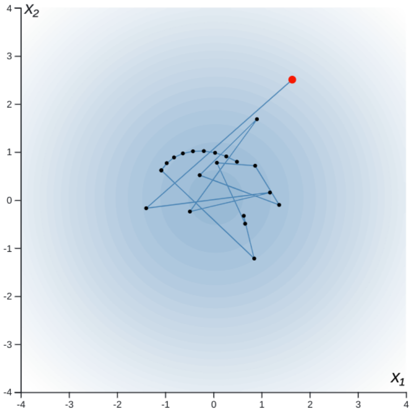
$$\alpha = \min\left(1, \frac{p(\theta_p | \mathbf{x})}{p(\theta^{(i-1)} | \mathbf{x})}\right)$$

  3. With probability α set $\theta^{(i)} = \theta_p$ and $\theta^{(i)} = \theta^{(i-1)}$ otherwise.

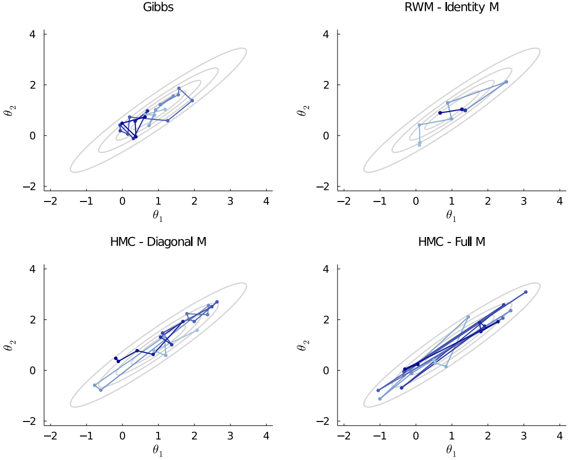
Interactive - Random Walk Metropolis



Interactive - Hamiltonian Monte Carlo



Comparing algorithms - multivariate normal target



Variational Inference

- Approx the posterior $p(\boldsymbol{\theta}|\boldsymbol{x})$ with a (simpler) distribution $q(\theta)$.
- **Mean field Variational Inference (VI)**:

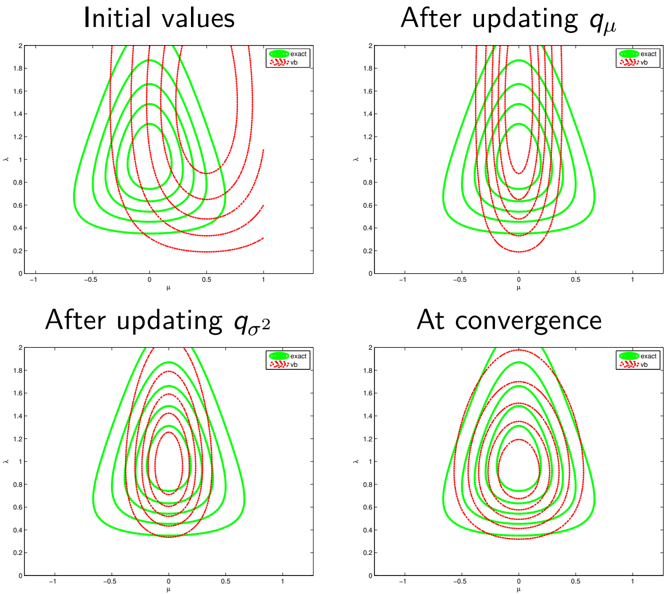
$$q(\boldsymbol{\theta}) = \prod_{i=1}^{p} q_i(\theta_i)$$

- **Parametric VI**: Parametric family $q_\lambda(\boldsymbol{\theta})$ with parameters λ.
- Find the $q(\boldsymbol{\theta})$ that **minimizes the Kullback-Leibler distance** between the true posterior p and the approximation q:

$$KL(q, p) = \int \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\boldsymbol{x})} q(\boldsymbol{\theta}) d\boldsymbol{\theta}.$$

- Enough with proportional form $p(\boldsymbol{\theta}|\boldsymbol{x}) \propto p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})$.

Normal example from Murphy ($\lambda = 1/\sigma^2$)



Probabilistic programming languages for Bayes

- **Stan** is a probabilistic programming language for Bayes based on HMC.
- C++ using the R package `rstan`. Bindings from Python.

- **Turing.jl** is a probabilistic programming language in Julia.
- Written in `Julia`, which is fast natively.

HMC sampling for iid normal model in Turing.jl

```
using Turing

ScaledInverseChiSq(ν,τ²) = InverseGamma(ν/2,ν*τ²/2) # Scaled Inv-χ² distribution

# Setting up the Turing model:
@model function iidnormal(x, μ₀, κ₀, ν₀, σ²₀)
    σ² ~ ScaledInverseChiSq(ν₀, σ²₀)
    θ ~ Normal(μ₀,σ²/κ₀)  # prior
    n = length(x)  # number of observations
    for i in 1:n
        x[i] ~ Normal(θ, √σ²) # model
    end
end

# Set up the observed data
x = [15.77,20.5,8.26,14.37,21.09]

# Set up the prior
μ₀ = 20; κ₀ = 1; ν₀ = 5; σ²₀ = 5^2

# Settings of the Hamiltonian Monte Carlo (HMC) sampler.
α = 0.8
postdraws = sample(iidnormal(x, μ₀, κ₀, ν₀, σ²₀), NUTS(α), 10000, discard_initial = 1000)
```

HMC sampling for iid normal model in rstan

```r
library(rstan)

# Define the Stan model
stanModelNormal = '
// The input data is a vector y of length N.
data {
  // data
  int<lower=0> N;
  vector[N] y;
  // prior
  real mu0;
  real<lower=0> kappa0;
  real<lower=0> nu0;
  real<lower=0> sigma20;
}

// The parameters in the model
parameters {
  real theta;
  real<lower=0> sigma2;
}

model {
  sigma2 ~ scaled_inv_chi_square(nu0, sqrt(sigma20));
  theta ~ normal(mu0,sqrt(sigma2/kappa0));
  y ~ normal(theta, sqrt(sigma2));
}
'

# Set up the observed data
data <- list(N = 5, y = c(15.77, 20.5, 8.26, 14.37, 21.09))

# Set up the prior
prior <- list(mu0 = 20, kappa0 = 1, nu0 = 5, sigma20 = 5^2)

# Sample from posterior using HMC
fit <- stan(model_code = stanModelNormal, data = c(data,prior), iter = 10000 )
```

Modeling the number of bidders in eBay auctions

variable	description	data type	original range
nbids	number of bids	counts	$[0, 12]$
bookvalue	coin's book value	continuous	$[7.5, 399.5]$
startprice	seller's reservation price / book value	continuous	$[0, 1.702]$
minblemish	minor blemish	binary	$[0, 1]$
majblemish	major blemish	binary	$[0, 1]$
negfeedback	large negative feedback score	binary	$[0, 1]$
powerseller	large quantity seller	binary	$[0, 1]$
verified	verified seller on ebay	binary	$[0, 1]$
sealed	unopened package	binary	$[0, 1]$

- **Poisson regression**

$$y_i | \boldsymbol{x}_i \sim \text{Poisson}(\lambda_i)$$
$$\lambda_i = \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})$$

HMC sampling for Poisson regression in Turing.jl

```
using Turing

# Setting up the poisson regression model
@model function poissonReg(y, X, τ)
    p = size(X,2)
    β ~ filldist(Normal(0, τ), p)  # all βⱼ are iid Normal(0, τ)
    λ = exp.(X*β)
    n = length(y)
    for i in 1:n
        y[i] ~ Poisson(λ[i])
    end
end

# HMC sampling from posterior of β
τ = 10    # Prior standard deviation
α = 0.70  # target acceptance probability in NUTS sampler
model = poissonReg(y, X, τ)
chain = sample(model, Turing.NUTS(α), 10000, discard_initial = 1000)
```

- Poisson regression in rstan.

... or TuringGLM.jl with R's formula syntax

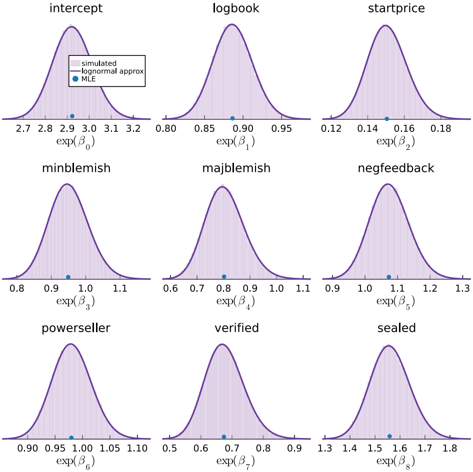
```julia
# Using TuringGLM.jl
using TuringGLM
fm = @formula(nbids ~ logbook + startprice + minblemish +
    majblemish + negfeedback + powerseller + verified + sealed)
model = turing_model(fm, ebay_df; model = Poisson)
chain = sample(model, NUTS(), 10000)
```

- Inspired by the brms package in R.

Marginal posteriors

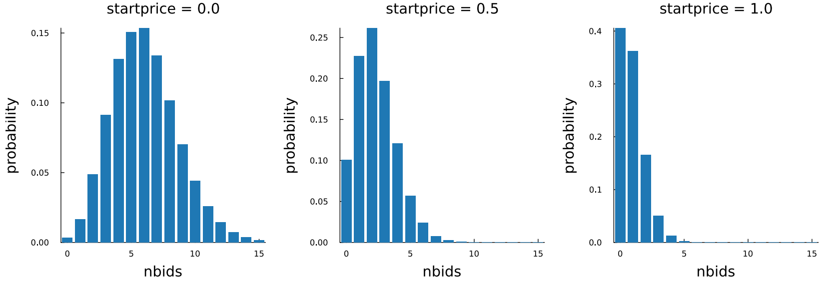
- Multiplicative model

$$E(y|\boldsymbol{x}) = \exp(\beta_0+\beta_1 x_1+\beta_2 x_2) = \exp(\beta_0)\exp(\beta_1)^{x_1}\exp(\beta_2)^{x_2}$$



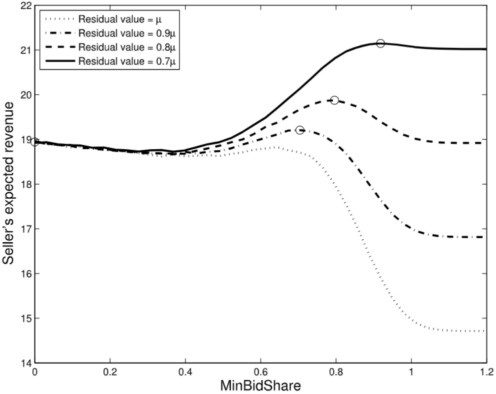
Predictive distributions for different startprice

- Test auction:
  - verified, powerseller with no substantial negative feedback
  - coin with major blemish in sealed packaging
  - book value \$100



Deciding on optimal startprice

- Wegmann and Villani (2011, JBES)
  - Structural model based on bid functions from game theory
  - Models and predicts number of bids and final price.
  - Predictive distribution performance on 50 test auctions.
  - Optimal startprice



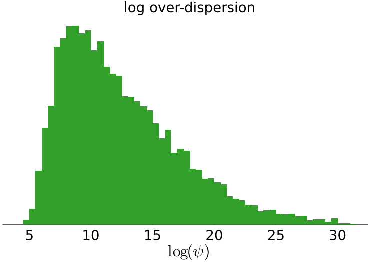
Negative binomial regression in Turing.jl

- **Negative binomial regression**

$$y_i|\boldsymbol{x}_i \sim \mathrm{NegBinomial}\left(\psi, p = \frac{\psi}{\psi + \lambda_i}\right), \quad \lambda_i = \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})$$

- Mean is still λ_i, but variance is larger: $Var(y_i) = \lambda_i(1 + \lambda_i/\psi)$.
- As $\psi \to \infty$ we get Poisson again.

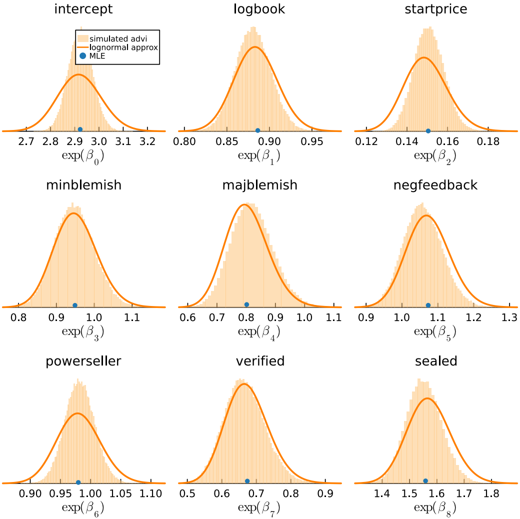
```julia
# Negative binomial regression
@model function negbinomialReg(y, X, τ, μ₀, σ₀)
    p = size(X,2)
    β ~ filldist(Normal(0, τ), p)
    λ = exp.(X*β)
    ψ ~ LogNormal(μ₀, σ₀)
    n = length(y)
    for i in 1:n
        y[i] ~ NegativeBinomial(ψ, ψ/(ψ + λ[i]))
    end
end
```



Variational inference - Poisson regression in Turing.jl

```
# Variational inference for posterior of β
τ = 10    # Prior standard deviation
model = poissonReg(y, X, τ)
nSamples = 10
nGradSteps = 1000
approx_post = vi(model, ADVI(nSamples, nGradSteps))
βsample = rand(approx_post, 1000)
```

Variational inference - Poisson regression in Turing.jl



Some resources for further study

- There are many good Bayesian textbooks, for example:
  - Gelman et al (2013). Bayesian Data Analysis
  - Bishop (2006). Pattern Recognition and Machine Learning
  - McElreath (2022). Statistical Rethinking.
  - Bernardo and Smith (1994). Bayesian Theory.
- Here are some of my own materials:
  - Bayesian Learning - a gentle introduction. Book in progress.
  - Bayesian Learning course - slides, computer labs and exams.
  - Advanced Bayesian Learning course - slides and computer labs.
  - Bayesian Learning - Observable Javascript widgets.
- Turing tutorials with neural nets and variational inference.
- The excellent Stan user guide has a lot of examples.
- PyMC is one of the many PPL for Bayes in Python.