Stan Demo

BSDA 2025

Väinö Yrjänäinen

Today's goals

- Get Stan installed
- Replicate simple examples with Stan
- Import own data and tinker with the models (if time allows)

Stan Installation

Installing stan

- The easiest choice for most of you is rstan
  - R and RStudio have nice stan features
- You can also use pystan or other versions if you want to

Before installing

- Check your R version
- What operating systems are you using?

Installing rstan

- Instructions: https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
- Requires C++ compliler

```r
install.packages("rstan")
```

Installing pystan

- Create and activate conda environment
- Install pystan and arviz for R hat etc.[1]

```
pip install pystan
pip install arviz
```

[1] https://discourse.mc-stan.org/t/pystan-parameter-summary-ess-and-rhat/22538

Verifying the installation


```r
example(stan_model, package = "rstan", run.dontrun = TRUE)
```

- Run the eight schools model[1][2]

[1] https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started#example-1-eight-schools

[2] https://pystan.readthedocs.io/en/latest/#quick-start

Examples

Three things

- Normally distributed data, no priors
- Binomial example
- Delayed flights

Normally distributed data

To model data $y_i \in \mathbb{R}$

$$y_i \sim \mathcal{N}(\mu, \sigma^2)$$

$$i \in \{1, ..., N\}$$

Add prior

- Flat prior for μ?
- $\sigma^2 \sim \mathrm{Exp}(\lambda)$?

Gaussian model

```
// The input data is a vector 'y' of length 'N'.
data {
  int<lower=0> N;
  vector[N] y;
}

// The parameters accepted by the model. Our model
// accepts two parameters 'mu' and 'sigma'.
parameters {
  real mu;
  real<lower=0> sigma;
}

// The model to be estimated. We model the output
// 'y' to be normally distributed with mean 'mu'
// and standard deviation 'sigma'.
model {
  //sigma ~ exponential(100.0);
  y ~ normal(mu, sigma);
}
```

Binomial model

$$n_{j,i} \sim \mathrm{Binomial}\left(N_j, \theta_j\right)$$
$$i \in \{1, ..., N\}$$
$$j \in \{0, 1\}$$

j are the treatment and control groups

Binomial model


```
data {
  int<lower=0> n1;
  int<lower=0> m1;
  int<lower=0> n2;
  int<lower=0> m2;
}
parameters {
  real<lower=0.0, upper=1.0> theta1;
  real<lower=0.0, upper=1.0> theta2;
}
model {
  theta1 ~ beta(3,1);
  theta2 ~ beta(3,1);
  n1 ~ binomial(n1 + m1, theta1);
  n2 ~ binomial(n2 + m2, theta2);
}
generated quantities {
  real or1 = theta1 / (1 - theta1);
  real or2 = theta2 / (1 - theta2);
  real ratio = or1 / or2;
  real logodds = log(ratio);
}
```

Delayed trains

date	duration	scheduled_duration
25/09/2025	01:02	0:57
29/09/2025	01:15	0:57
30/09/2025	01:00	0:57
01/09/2025	01:25	0:57
02/10/2025	01:05	0:57

Delayed flights

- I want to know when I arrive when the departure is, say 8.30
  - Model the duration of the next train trip (posterior predictive)
- Use the Lognormal model

Posterior predictive

For each posterior sample of (μ, σ^2), we draw

$$\tilde{y}_i \sim \text{Lognormal}(\mu, \sigma^2)$$

We also calculate whether you make it on time

$$\text{ontime} = \mathbb{I}(\tilde{y}_i \leq 10.0)$$

Use your own data

- Find some univariate data
  - Steps you took last week, points scored by your football team in the last 5 games, heights of people in your family, absolutely whatever
- Use the normal.stan model

Change the model

- Try out some new priors
- Use eg. ‘gamma’ instead of ‘normal’