

```

data {
  int<lower=0> J;
  real y[J];
  real<lower=0> sigma[J];
}

parameters {
  real mu;
  real<lower=0> tau;
  vector[J] theta_tilde;
}

transformed parameters {
  vector[J] theta = mu + tau * theta_tilde;
}

model {
  mu ~ normal(0, 5);
  tau ~ normal(0, 5);
  theta_tilde ~ normal(0, 1);
  y ~ normal(theta, sigma);
}

```



```

import numpy as np
import tensorflow as tf
import tensorflow_probability as tfp
tfd = tfp.distributions
tfb = tfp.bijectors

class eight_schools_model(tfd._MixtureModel):
    def __init__(self, J, y, sigma):
        self.J = J
        self.y = tf.cast(y, tf.float64)
        self.sigma = tf.cast(sigma, tf.float64)

    def log_prob_one_chain(self, params):
        target = 0
        # data
        J = self.J
        y = self.y
        sigma = self.sigma
        # parameters
        mu = tf.cast(params[0], tf.float64)
        tau = tf.cast(params[1], tf.float64)
        theta_tilde = tf.cast(params[2], tf.float64)
        # target log probability computation
        theta = mu + (tau * theta_tilde)
        target += tf.reduce_sum(tfd.Normal(tf.cast(0, tf.float64),
                                           tf.cast(5, tf.float64)).log_prob(mu))
        target += tf.reduce_sum(tfd.Normal(tf.cast(0, tf.float64),
                                           tf.cast(5, tf.float64)).log_prob(tau))
        target += tf.reduce_sum(tfd.Normal(tf.cast(0, tf.float64),
                                           tf.cast(1, tf.float64)).log_prob(theta_tilde))
        target += tf.reduce_sum(tfd.Normal(theta, sigma).log_prob(y))
        return target

    def log_prob(self, params):
        return tf.vectorized_map(self.log_prob_one_chain, params)

    def parameter_shapes(self, nchains_):
        J = self.J
        y = self.y
        sigma = self.sigma
        return [(nchains_, 1), (nchains_, J)]

    def parameter_bijectors(self):
        J = self.J
        y = self.y
        sigma = self.sigma
        return [tfb.Identity(),
                tfb.Chain([tfb.Shift(tf.cast(0, tf.float64)), tfb.Exp()]),
                tfb.Identity()]

    def parameter_names(self):
        return ["mu", "tau", "theta_tilde"]

```

