Variational Bayesian Inference Methods in the Prediction of Share Closing Prices

Duy Nguyen & Edward Bai

2024-03-16

Introduction

The ability to forecast accurately whilst taking into account known information is important across all fields and industries. In economics, especially when money-making is concerned, the stakes become even higher thereby demanding a certain degree of accuracy with these predictions. Thus, statisticians and data scientists have come up with many ways to formulate prediction models, to help inform business or investment decisions, that are becoming increasingly accurate as contemporary research bolsters existing methods.

Many of these methods consist of models that involve complex relationships between numerous variables, some of which are latent or unobserved meaning that some models will fare better than others. This study seeks to evaluate a number of these models against one another in the context of forecasting stock price movements to ascertain their respective performances. Specifically, we will be attempting to identify whether key macroeconomic indicators can be used to inform, and therefore improve forecasting techniques. With this in mind, we will inherently be working with large datasets, requiring the use of methods appropriate for large datasets, namely Variational Bayesian Inference models.

Why do we want to use Variational Bayes (VB) for this scenario? This is because VB is often more scalable to large datasets compared to sampling-based methods like Markov Chain Monte Carlo (MCMC). For models where the resulting posterior distributions are likely to be complex and intractable due to involving high-dimensional spaces (or non-conjugate priors), VB methods provide a way to approximate these distributions. Lastly, VB's output provides a measure that enables the comparison and selection of different potential models (Chérief-Abdellatif and Alquier 2018). The rest of the paper will detail the dataset under study as well as the employed methodologies and their evaluations.

Data Preparation & Analysis

Acknowledging the glaring differences between economically developed and currently developing economies, we initially sought data from two different regions to make for a better comparison. However, with this difference also comes the difficulty of data collection and subsequently determining its authenticity. Several nations do not have the agency or capability to reliably report data for various reasons. From this, we narrowed our search to regions that have renowned agencies with established systems of collecting and reporting data. In addition to this, we wanted our model to be based on a globally connected economy that is likely to have significant economic reactions to international factors such as war or recessions. The Organisation for Economic Co-operation and Development (OECD) is a forum consisting of 37 democracies with market-based economies with the purpose of joint development of policy standards to encourage sustainable economic growth. The OECD has an abundant collection of data on various economic indicators that we could use and so we settled on focusing on data for the Canadian economy because it best matches the aforementioned criteria.

Forecasting the movement of any particular stock is rather difficult as this necessitates a model that takes into account the myriad factors that can influence changes in stock price. Attempting to build a model that encompasses all of these potential factors will ultimately not do any good as this risks upsetting the model complexity tradeoff and the model ends up being over-fit or just not meaningful. To mitigate this, we will instead forecast the closing values of the S&P/TSX Composite Index, a benchmark Canadian index representing approximately 70% of the total market capitalization on the Toronto Stock Exchange. The closing price is the 'raw' price which is just the cash value of the last transacted price before the market closes.

Regarding our predictor variables, we have 4 different economic indicators for Canada, as follows.

- Unemployment rate U_t
- Consumer Price Index CPI_t (in percentage per annum)
- Industrial Production Indices I
  - $I_{S,t}$: Total industrial production
  - $I_{M,t}$: Manufacturing
  - $I_{C,t}$: Construction
- Interest Rates R (in percentage per annum)
  - $R_{\text{long},t}$: Long term
  - $R_{\text{short},t}$: Short term

Briefly speaking, these variables were chosen due to the impact and shocks they tend to have on stock market conditions. For example, high unemployment rates may indicate a faltering economy, which may lower investor confidence and hurt the performance of the stock market. Excessive inflation can reduce buying power, put pressure on company profit margins, and weaken stock values. Industrial Production indices measure real output in the three listed sectors and rising values tend to signal economic growth and, in turn, stock prices.

Model Structure

We will employ a Bayesian linear regression model where the response variable TSX_t is regressed on the economic indicators listed above:

$$\text{TSX}_t = \beta_0 + \beta_1 U_t + \beta_2 CPI_t + \beta_3 I_{\text{S},t} + \beta_4 IP_{\text{M},t} + \beta_5 IP_{\text{C},t} + \beta_6 R_{\text{long},t} + \beta_7 R_{\text{short},t} + \epsilon_t$$

It will be computationally prohibitive to approximate the posterior distributions of the model parameters (i.e. Beta coefficients, sigma) given the nature of our data. The high-dimensional data space will lead to high-dimensional integration (due to the number of variables and their cross-interactions) that are not analytically tractable. Here, Variational Bayes provides a practical solution by approximating these complex posteriors with simpler distributions through a Variational Family which is a family of distributions that approximates the posteriors. We will be using the common choice of the mean-field variational family in which the joint distribution of the model parameters is approximated by a product of independent distributions for each parameter:

$$q(\theta) = q(\beta_0) \times q(\beta_1) \times \cdots \times q(\beta_7) \times q(\sigma^2)$$

Here, θ represents all the parameters in the model and each β and σ is assumed to have its own distinct distribution q, typically chosen to be in the same family as the prior (normal and cauchy distributions). This means that we will be using a normal prior for the regression coefficients (β) which is a natural choice when the likelihood is also normal (as is the case with the residuals). This combination leads to a normal

posterior, which is a conjugate prior for the normal likelihood and this will simplify computations. Note that due to the nature of the data, the predictors have large differences in magnitude. In order to make predictions more interpretable, the data has been standardized and then fitted with a simple linear model. To remove any present trends from the data, their residuals are then used for training the VB model.

$$\beta_i \sim N(0, 1) \text{ for } i \in [1, 7]$$

For the variance parameter, we will be using the Half-cauchy prior for its robustness. It's heavy tails enable the accommodation of outliers and large variances, which is quite common from a quick inspection of the data. We could alternatively use an Inverse-Gamma distribution for the variance but this can lead to overly informative priors that affect the robustness of posterior estimates.

$$\sigma \sim Cauchy(0, 2.5)$$

The aim of VB is to minimize the Kullback-Leibler (KL) divergence between the variational distribution $q(\theta)$ and the true posterior $p(\theta|\text{data})$. This is done by maximizing the Evidence Lower Bound (ELBO).

$$\text{ELBO} = \mathbb{E}_q[\log p(y, \theta)] - \mathbb{E}_q[\log q(\theta)]$$

Results

Due to the experimental nature of the `vb()` function from the RStan library, we will also fit a simpler model using the `sampling()` function for Monte Carlo Markov Chains (MCMC) and compare their performances. The models yield the trace plots on the following pages. It is evident for the model with mixtures that the chains have poorly mixed because they appear separated and don't fully explore the parameter space. Moreover, the ELBO values generated from fitting the model via variational bayesian inference initially increases, as we expected, but it quickly plateaus. Repeating the fitting process, we see that the Pareto K diagnostic value from VB fluctuates roughly between 0.7 and 1.5 (See Appendix). Normally, we want Pareto K's between 0.5 and 0.7 which means that this methodology needs extensive fine-tuning and re-specification.

In contrast, the simpler model trace plots exhibits the 'fuzzy caterpillar' shape meaning that the chains for each parameter are mixing well, moving over the same space without obvious trends or repeated patterns. This indicates that the chains are converging to the target distribution.

Criticism and Conclusion

It is obvious that a lot of critique can be said about the model under study. There are still many ways to improve the current model, such as by continuing to fine-tune the model hyper-parameters, using more samples, or employing more diagnostic methods that can give specific insight into how the model can be improved. Building a model in the intrinsically complex problem domain of economics would naturally cause the model to be complex itself, especially when mixtures are incorporated. Along with the identification problem (different parameter values may result in the same mixture distribution), this complexity makes the model rather difficult to fit and interpret. Moreover, high Pareto k diagnostic values suggest that the variational approximation may be missing important parts of the posterior distribution meaning that the model is not capturing the complexity of the data.

In addition, financial time series data can exhibit heavy tails, and a normal likelihood may not be the best prior choice. The model also does not account for potential autocorrelations within the time-series data. Ignoring these temporal dependencies, namely auto-regressive and moving average components) will absolutely lead to a mis-specified model. In this vein, it would be wise to incorporate these new time dependent variables into the current model in order to further this study. Should time permit, addressing these issues should enhance the robustness and reliability of the model estimations.

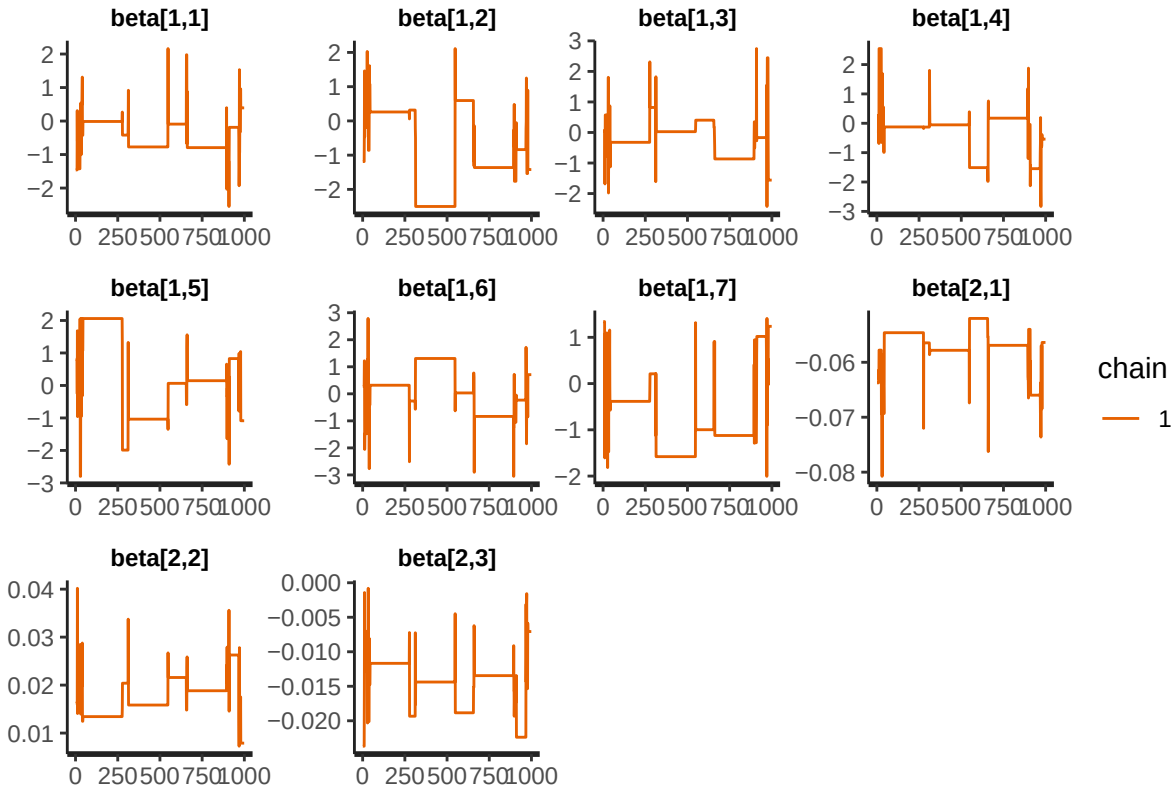


Figure 1: Trace plot for variational model with mixtures

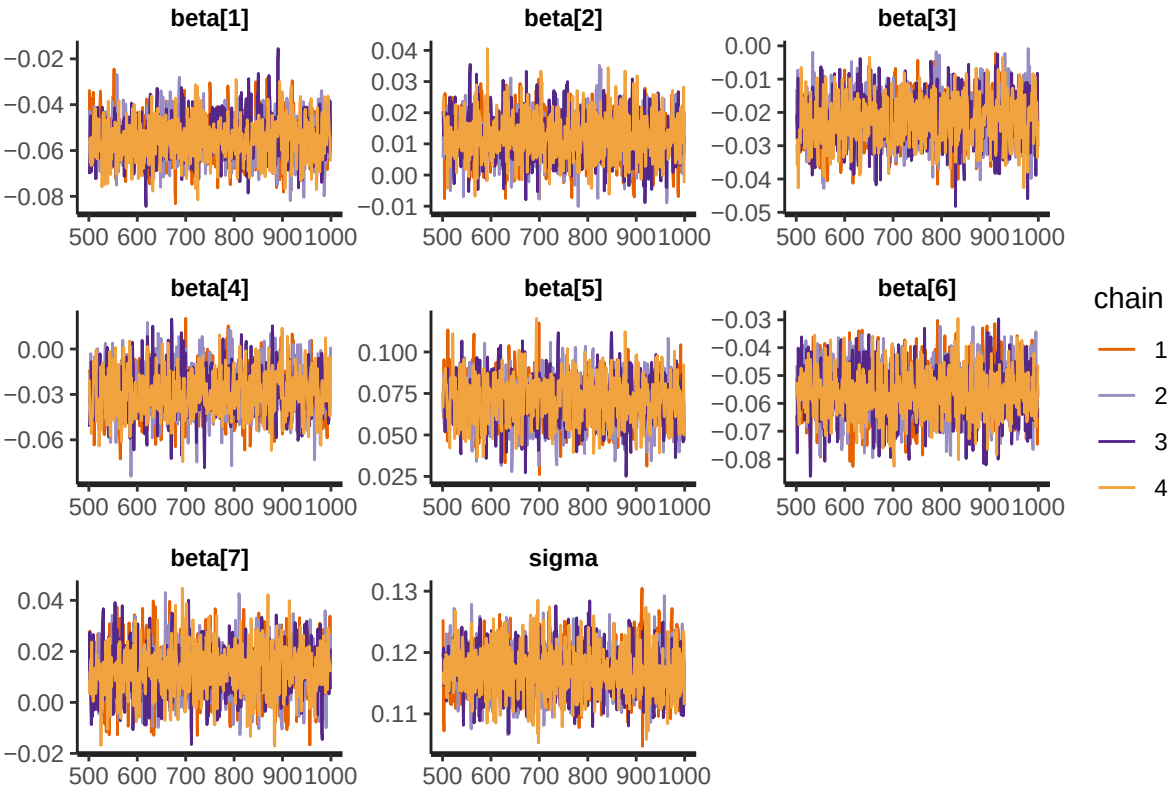


Figure 2: Trace plot for simple model

References

Chérief-Abdellatif, Badr-Eddine, and Pierre Alquier. 2018. "Consistency of Variational Bayes Inference for Estimation and Model Selection in Mixtures." *Electronic Journal of Statistics* 12 (2): 2995–3035. https://doi.org/10.1214/18-EJS1475.

"Consumer Price Indices (CPIs, HICPs), COICOP 1999." 2024. [Data set]. https://data-viewer.oecd.org?chartId=9cef7bd5-43eb-4781-909a-b6e7be4f0e11.

"Financial Market." 2024. [Data set]. https://data-viewer.oecd.org?chartId=36e31dd7-9185-4764-bfd5-b151fa20c29c.

"Infra-Annual Labour Statistics." 2024. [Data set]. https://data-viewer.oecd.org?chartId=492d0147-1557-4d23-bb66-79395de80775.

Labarr, Aric. 2019. "The Bayesians Are Coming! The Bayesians Are Coming! The Bayesians Are Coming to Time Series!" Proceedings of the SAS Global Forum 3188. Institute for Advanced Analytics at North Carolina State University. https://support.sas.com/resources/papers/proceedings19/3188-2019.pdf.

"Production, Sales, Work Started and Orders." 2024. [Data set]. https://data-viewer.oecd.org?chartId=f8c73af7-7711-41aa-a84e-e5ae1f6f2126.

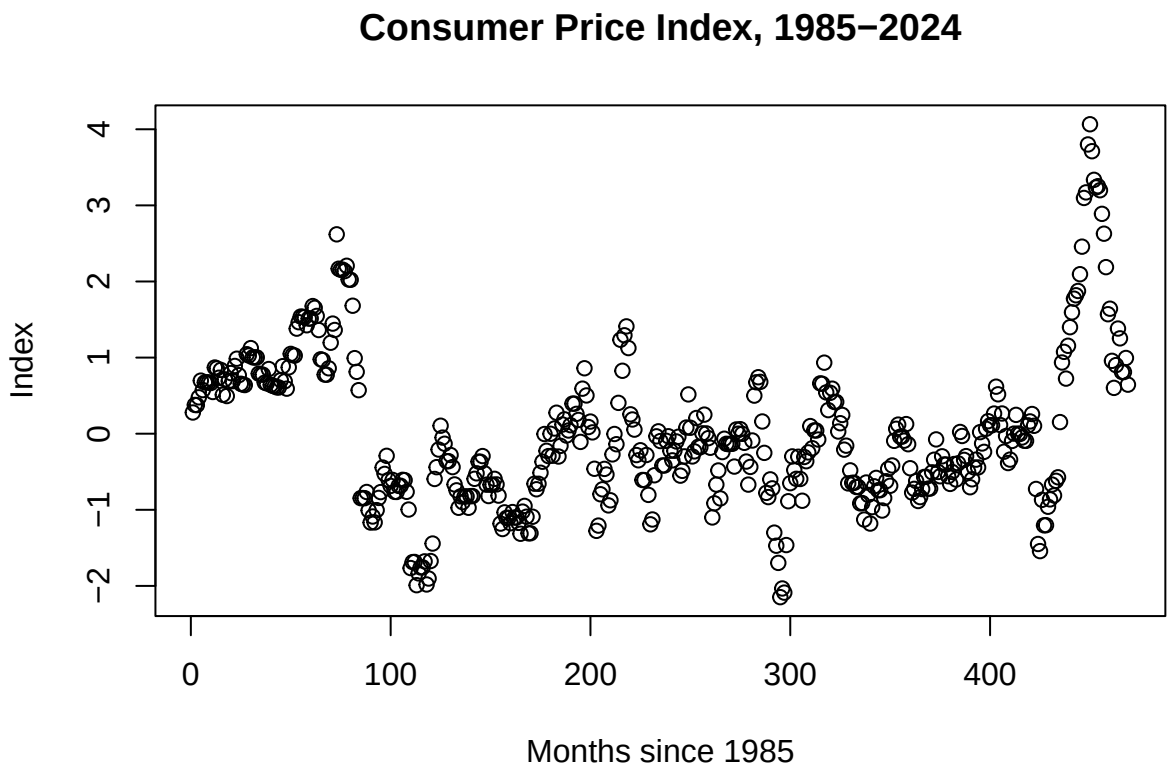
RStan Development Team. 2016. *Stanmodel Method-Vb*. https://rdrr.io/cran/rstan/man/stanmodel-method-vb.html.

Yahoo Finance. 2024. "TSX Index Historical Data." [Data set]. https://ca.finance.yahoo.com/quote/%5EGSPTSE/history.

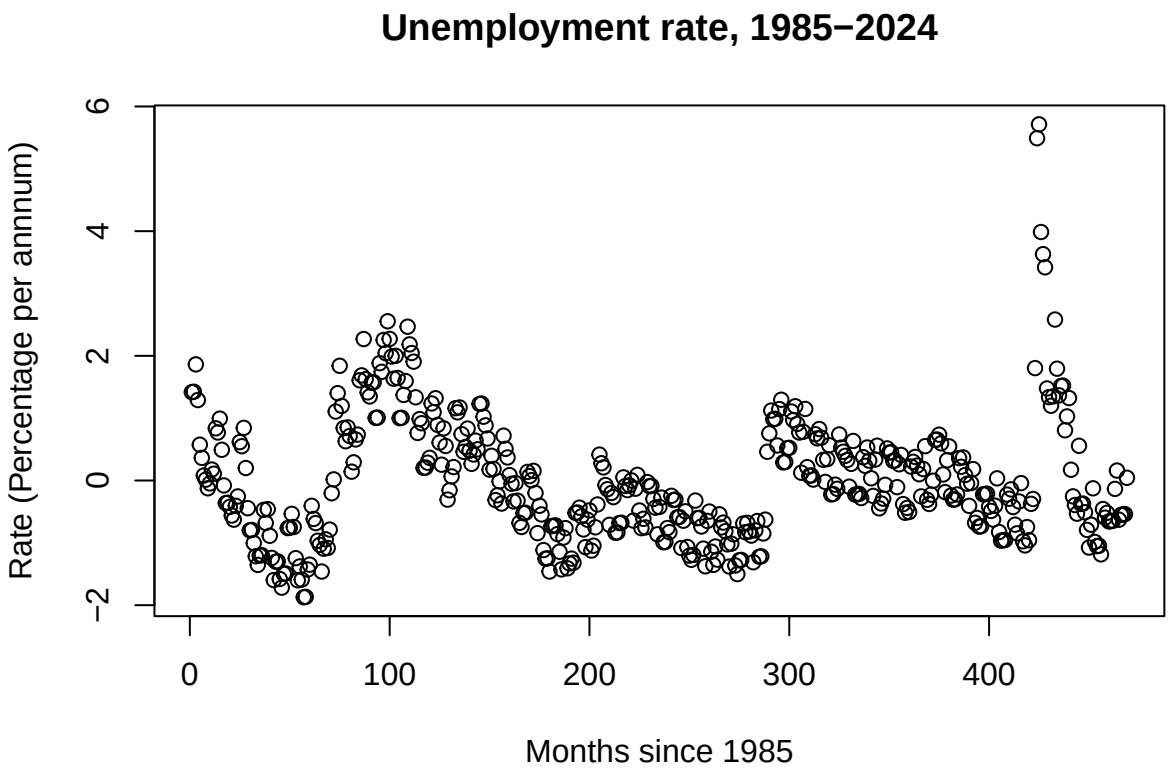
Appendix

The following are plots to show what the data under study look like.

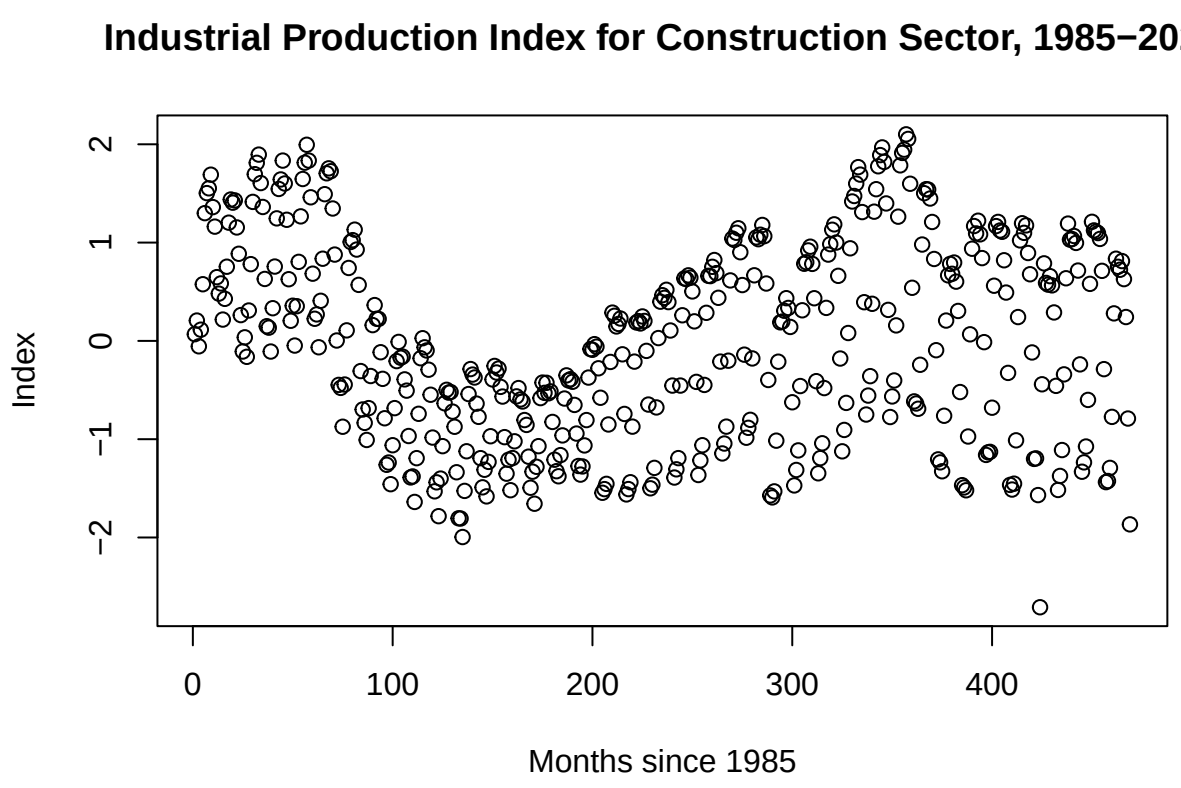
```
plot(inflation$CPI, main="Consumer Price Index, 1985-2024",
     ylab="Index", xlab="Months since 1985")
```



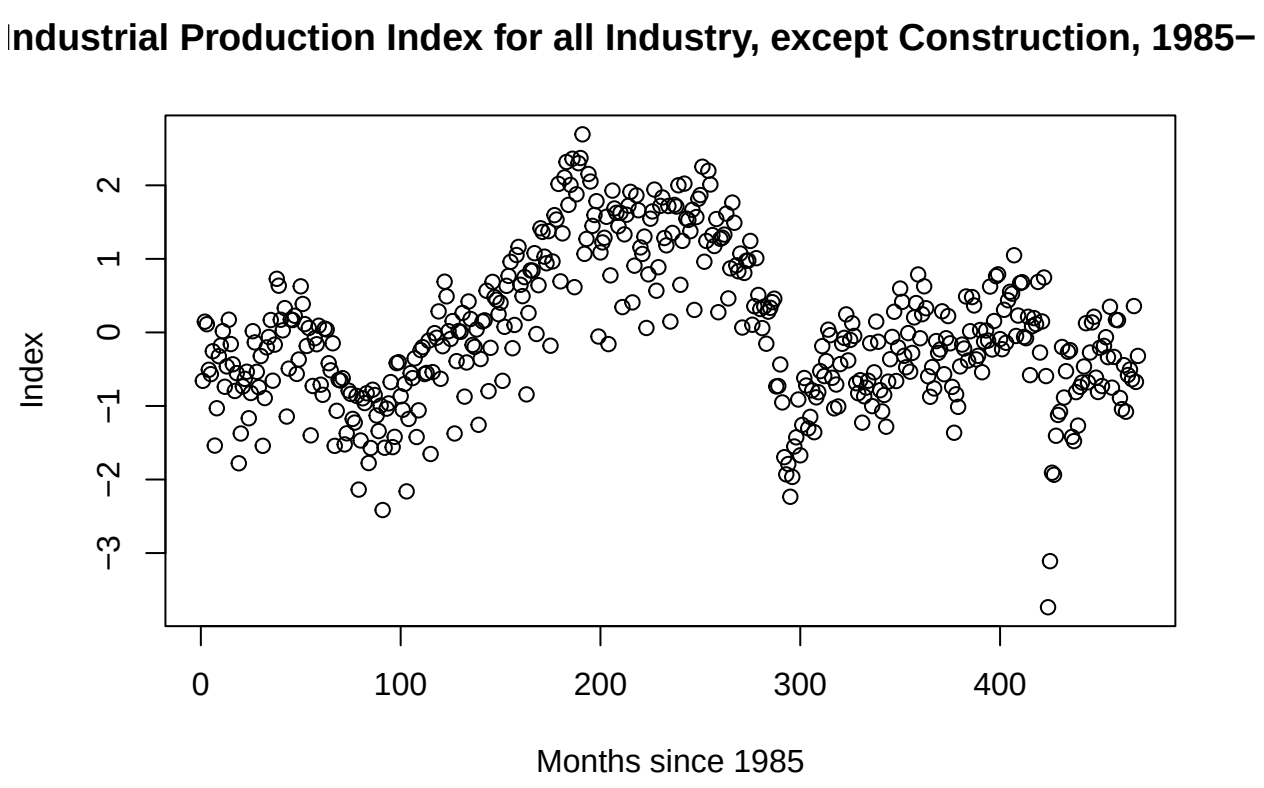

```
plot(unemployment$UNEMPLOYMENT_RATE, main="Unemployment rate, 1985-2024",
     ylab="Rate (Percentage per annnum)", xlab="Months since 1985")
```



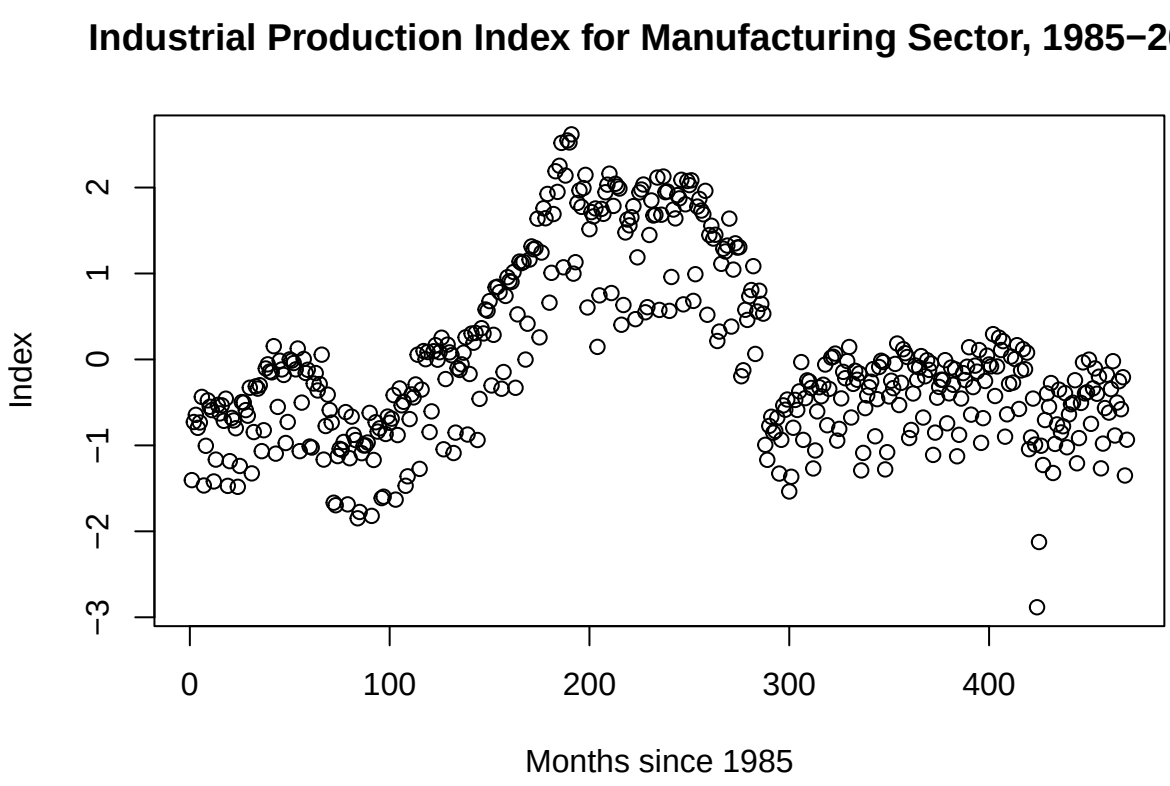

```
plot(industrial$Construction,
     main="Industrial Production Index for Construction Sector, 1985-2024",
     ylab="Index", xlab="Months since 1985")
```



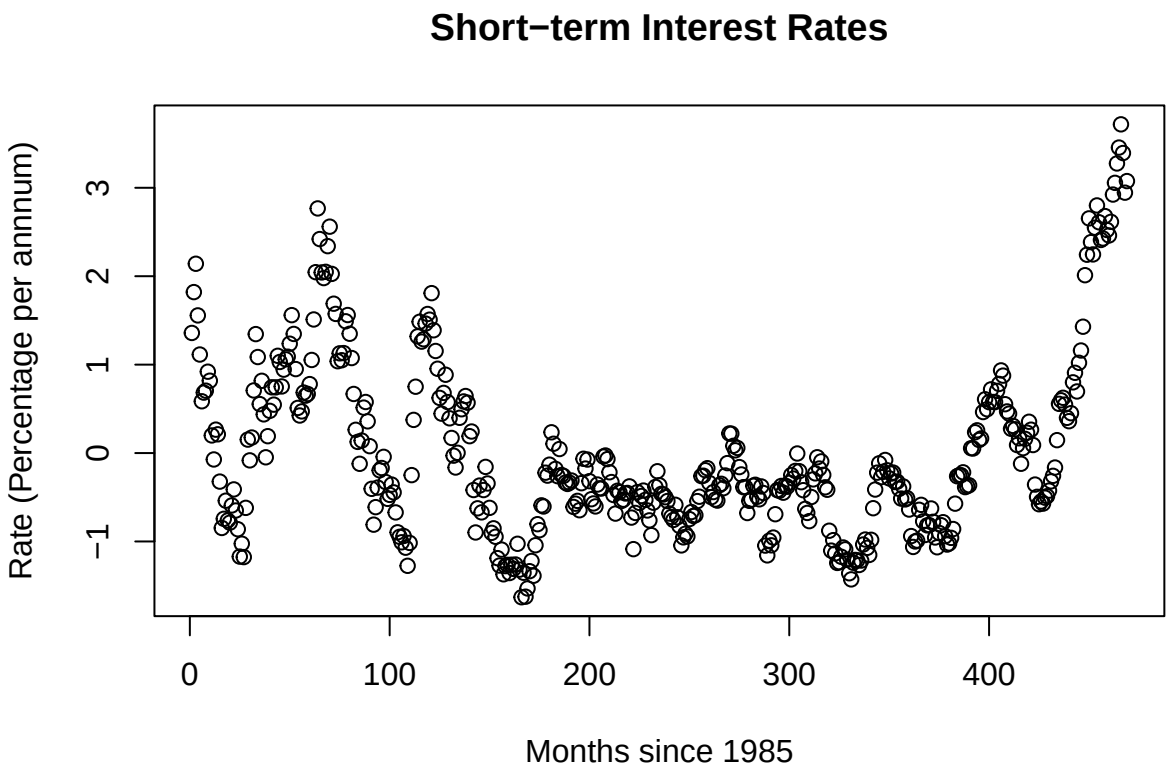

```
plot(industrial$`Industry (except construction)`,
     main="Industrial Production Index for all Industry, except Construction, 1985-2024",
     ylab="Index", xlab="Months since 1985")
```



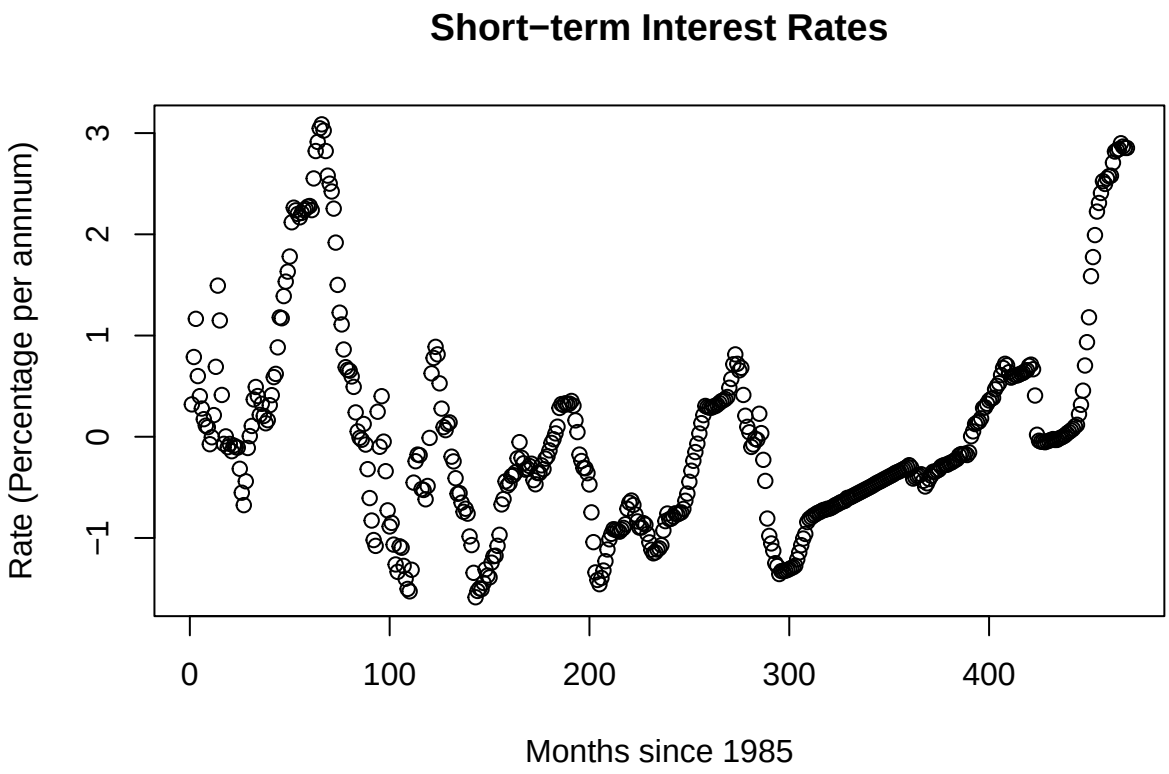

```
plot(industrial$Manufacturing,
     main="Industrial Production Index for Manufacturing Sector, 1985-2024",
     ylab="Index", xlab="Months since 1985")
```



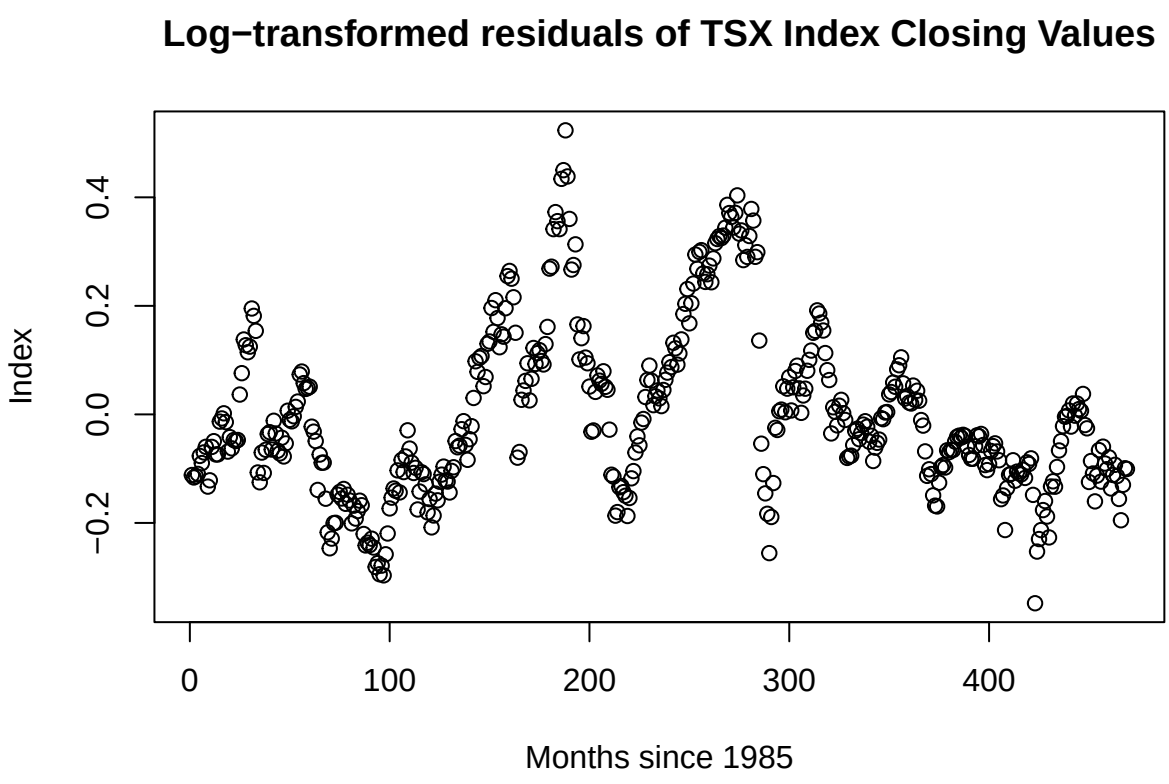

```
plot(interest$`Short-term interest rates`,
     main="Short-term Interest Rates", ylab="Rate (Percentage per annnum)",
     xlab="Months since 1985")
```




```
plot(interest$`Long-term interest rates`,
     main="Short-term Interest Rates", ylab="Rate (Percentage per annnum)",
     xlab="Months since 1985")
```




```
plot(response$Close_log_residuals,
     main="Log-transformed residuals of TSX Index Closing Values",
     ylab="Index", xlab="Months since 1985")
```



The following Stan program is the simpler model, without mixtures, used in the analysis of the data under study.

```
data {
  int<lower=1> N;              // Number of data points
  int<lower=1> K;              // Number of predictors
  matrix[N, K] X;              // Predictor matrix
  vector[N] y;                 // Response variable (TSX index)
}

parameters {
  vector[K] beta;              // Coefficients for predictors
  real<lower=0> sigma;         // Standard deviation of the residuals
}

model {
  // Priors
  beta ~ normal(0, 1);         // Standard normal prior for beta coefficients
  sigma ~ cauchy(0, 2.5);      // Weakly informative prior for sigma

  // Likelihood
  y ~ normal(X * beta, sigma);
}
```

The following Stan program is the model that contains mixtures.

```
data {
  int<lower=1> N;              // Number of data points
  int<lower=1> K;              // Number of predictors
  matrix[N, K] X;              // Predictor matrix
  vector[N] y;                 // Response variable (TSX index)
  int<lower=1> num_components; // Number of components in the mixture
}

parameters {
  vector[K] beta[num_components];   // Coefficients for each component
  real<lower=0> sigma[num_components]; // Standard deviation for each component
  simplex[num_components] mixing_proportions; // Mixing proportions for each component
}

model {
  // Priors
  for (n in 1:num_components) {
    beta[n] ~ normal(0, 1);
    sigma[n] ~ cauchy(0, 2.5);
  }

  // Mixture likelihood
  for (i in 1:N) {
    vector[num_components] component_likelihoods;
    for (n in 1:num_components) {
      component_likelihoods[n] = log(mixing_proportions[n]) +
      normal_lpdf(y[i] | X[i] * beta[n], sigma[n]);
    }
    target += log_sum_exp(component_likelihoods);
  }
}
```

The following code block fits the two models onto the data under study and is followed by their respective outputs which are both discussed in the study.

```
stan_data <- list(N = nrow(predictors),
                  K = ncol(predictors),
                  X = predictors,
                  y = response$Close_log_residuals)

# Fit stan model with variational baye's using ADVI
fit_sampling <- sampling(stock, data = stan_data,
          iter = 1000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
```

```
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.054 seconds (Warm-up)
## Chain 1:                0.047 seconds (Sampling)
## Chain 1:                0.101 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.055 seconds (Warm-up)
## Chain 2:                0.05 seconds (Sampling)
## Chain 2:                0.105 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.9e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
```

```
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.053 seconds (Warm-up)
## Chain 3:                0.05 seconds (Sampling)
## Chain 3:                0.103 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.052 seconds (Warm-up)
## Chain 4:                0.048 seconds (Sampling)
## Chain 4:                0.1 seconds (Total)
## Chain 4:
```

```r
# Prepare data list for stan
stan_data <- list(N = nrow(predictors),
                  K = ncol(predictors),
                  X = predictors,
                  y = response$Close_log_residuals,
                  num_components = 2)

# Fit stan model with variational baye's using ADVI
fit_mixture <- vb(stockmixture, data = stan_data,
          iter = 10000,
          tol_rel_obj = 0.0002,  # Adjusted tolerance for stopping criterion
          output_samples = 1000,
          importance_resampling = TRUE,
          pars=c("beta", "sigma"),
          include=TRUE,
          elbo_samples = 250,
```

```
          init=0,
          algorithm="meanfield")  # Enable importance resampling
```

```
## Chain 1: ------------------------------------------------------------
## Chain 1: EXPERIMENTAL ALGORITHM:
## Chain 1:   This procedure has not been thoroughly tested and may be unstable
## Chain 1:   or buggy. The interface is subject to change.
## Chain 1: ------------------------------------------------------------
## Chain 1:
## Chain 1:
## Chain 1:
## Chain 1: Gradient evaluation took 0.000162 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.62 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Begin eta adaptation.
## Chain 1: Iteration:   1 / 250 [  0%]  (Adaptation)
## Chain 1: Iteration:  50 / 250 [ 20%]  (Adaptation)
## Chain 1: Iteration: 100 / 250 [ 40%]  (Adaptation)
## Chain 1: Iteration: 150 / 250 [ 60%]  (Adaptation)
## Chain 1: Iteration: 200 / 250 [ 80%]  (Adaptation)
## Chain 1: Success! Found best value [eta = 1] earlier than expected.
## Chain 1:
## Chain 1: Begin stochastic gradient ascent.
## Chain 1:   iter             ELBO   delta_ELBO_mean   delta_ELBO_med   notes
## Chain 1:    100         -289.272             1.000            1.000
## Chain 1:    200           95.076             2.521            4.043
## Chain 1:    300          203.461             1.858            1.000
## Chain 1:    400         -543.555             1.737            1.374
## Chain 1:    500          231.055             2.060            1.374
## Chain 1:    600          169.420             1.778            1.374
## Chain 1:    700          258.691             1.573            1.000
## Chain 1:    800          255.689             1.378            1.000
## Chain 1:    900          229.462             1.237            0.533
## Chain 1:   1000          274.703             1.130            0.533
## Chain 1:   1100          213.941             1.059            0.364   MAY BE DIVERGING... INSPECT ELI
## Chain 1:   1200          191.662             0.666            0.345   MAY BE DIVERGING... INSPECT ELI
## Chain 1:   1300          191.879             0.613            0.284   MAY BE DIVERGING... INSPECT ELI
## Chain 1:   1400          232.631             0.493            0.175
## Chain 1:   1500          181.798             0.186            0.175
## Chain 1:   1600          239.335             0.173            0.175
## Chain 1:   1700          236.966             0.140            0.165
## Chain 1:   1800          195.193             0.160            0.175
## Chain 1:   1900          276.924             0.178            0.214
## Chain 1:   2000          251.256             0.172            0.214
## Chain 1:   2100          284.272             0.155            0.175
## Chain 1:   2200          276.718             0.146            0.175
## Chain 1:   2300          287.867             0.150            0.175
## Chain 1:   2400          245.924             0.149            0.171
## Chain 1:   2500          261.247             0.127            0.116
## Chain 1:   2600          273.633             0.108            0.102
## Chain 1:   2700          278.738             0.109            0.102
```

```
## Chain 1:   2800          279.849            0.088            0.059
## Chain 1:   2900          277.502            0.059            0.045
## Chain 1:   3000          267.438            0.053            0.039
## Chain 1:   3100          274.686            0.044            0.038
## Chain 1:   3200          263.969            0.045            0.039
## Chain 1:   3300          265.072            0.041            0.038
## Chain 1:   3400          287.463            0.032            0.038
## Chain 1:   3500          289.306            0.027            0.026
## Chain 1:   3600          272.806            0.028            0.026
## Chain 1:   3700          289.099            0.032            0.038
## Chain 1:   3800          295.620            0.034            0.038
## Chain 1:   3900          271.511            0.042            0.041
## Chain 1:   4000          293.025            0.046            0.056
## Chain 1:   4100          283.320            0.046            0.056
## Chain 1:   4200          281.023            0.043            0.056
## Chain 1:   4300          286.810            0.045            0.056
## Chain 1:   4400          291.996            0.039            0.034
## Chain 1:   4500          277.690            0.043            0.052
## Chain 1:   4600          285.764            0.040            0.034
## Chain 1:   4700          284.647            0.035            0.028
## Chain 1:   4800          289.222            0.034            0.028
## Chain 1:   4900          292.253            0.026            0.020
## Chain 1:   5000          274.909            0.025            0.020
## Chain 1:   5100          292.788            0.028            0.020
## Chain 1:   5200          284.956            0.030            0.027
## Chain 1:   5300          268.256            0.034            0.028
## Chain 1:   5400          292.336            0.041            0.052
## Chain 1:   5500          276.614            0.041            0.057
## Chain 1:   5600          292.458            0.044            0.057
## Chain 1:   5700          288.585            0.045            0.057
## Chain 1:   5800          250.817            0.058            0.061
## Chain 1:   5900          294.799            0.072            0.062
## Chain 1:   6000          284.072            0.070            0.061
## Chain 1:   6100          266.893            0.070            0.062
## Chain 1:   6200          240.218            0.078            0.064
## Chain 1:   6300          294.009            0.090            0.082
## Chain 1:   6400          295.035            0.082            0.064
## Chain 1:   6500          292.971            0.077            0.064
## Chain 1:   6600          287.310            0.074            0.064
## Chain 1:   6700          291.063            0.074            0.064
## Chain 1:   6800          286.829            0.060            0.038
## Chain 1:   6900          292.895            0.047            0.021
## Chain 1:   7000          290.608            0.044            0.020
## Chain 1:   7100          294.393            0.039            0.015
## Chain 1:   7200          233.675            0.054            0.015
## Chain 1:   7300          289.500            0.055            0.015
## Chain 1:   7400          291.129            0.055            0.015
## Chain 1:   7500          282.665            0.058            0.020
## Chain 1:   7600          294.739            0.060            0.021
## Chain 1:   7700          283.760            0.062            0.030
## Chain 1:   7800          290.661            0.063            0.030
## Chain 1:   7900          285.690            0.063            0.030
## Chain 1:   8000          287.635            0.063            0.030
## Chain 1:   8100          291.977            0.063            0.030
```

```
## Chain 1:   8200          291.605             0.037            0.024
## Chain 1:   8300          291.636             0.018            0.017
## Chain 1:   8400          267.893             0.026            0.024
## Chain 1:   8500          288.282             0.030            0.024
## Chain 1:   8600          284.106             0.028            0.017
## Chain 1:   8700          289.445             0.026            0.017
## Chain 1:   8800          276.076             0.028            0.017
## Chain 1:   8900          282.732             0.029            0.018
## Chain 1:   9000          273.083             0.032            0.024
## Chain 1:   9100          291.044             0.036            0.035
## Chain 1:   9200          282.796             0.039            0.035
## Chain 1:   9300          292.005             0.042            0.035
## Chain 1:   9400          294.652             0.034            0.032
## Chain 1:   9500          266.669             0.038            0.032
## Chain 1:   9600          273.650             0.039            0.032
## Chain 1:   9700          273.723             0.037            0.032
## Chain 1:   9800          276.366             0.033            0.029
## Chain 1:   9900          277.998             0.031            0.029
## Chain 1:   10000          293.299             0.033            0.029
## Chain 1: Informational Message: The maximum number of iterations is reached! The algorithm may not ha
## Chain 1: This variational approximation is not guaranteed to be meaningful.
## Chain 1:
## Chain 1: Drawing a sample of size 1000 from the approximate posterior...
## Chain 1: COMPLETED.

## Warning: Pareto k diagnostic value is 0.84. Resampling is unreliable.
## Increasing the number of draws or decreasing tol_rel_obj may help.
```