Latent Variable Model and Data Generation

1 Introduction

This document describes the theoretical framework for a latent variable model used to generate data and estimate model parameters using Bayesian inference. The model is implemented in Stan and involves the generation of latent variables and observed data using a Vector Autoregressive (VAR) process.

2 Data Generation

We generate data for $N_{\text{people}} = 10$ individuals over $N_{\text{timepoints}} = 50$ time points, with an initial burn-in period of $N_{\text{burnins}} = 25$ time points. The latent variables are generated using a VAR(1) model, and the observed data is generated by adding noise to the latent variables.

2.1 VAR(1) Model

The latent variables $\boldsymbol{F}_t = [F_{t1}, F_{t2}]^\top$ are generated using the following VAR(1) process:

$$\boldsymbol{F}_t = \boldsymbol{\Phi}\boldsymbol{F}_{t-1} + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),$$

where $\boldsymbol{\Phi}$ is the coefficient matrix:

$$\boldsymbol{\Phi} = \begin{bmatrix} 0.5 & 0.2 \\ -0.3 & 0.4 \end{bmatrix}.$$

2.2 Observed Data

The observed data $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ are generated by adding Gaussian noise to the latent variables:

$$Y_{1tij} = F_{ti1} + \eta_{1tij}, \quad \eta_{1tij} \sim \mathcal{N}(0, 0.1),$$

$$Y_{2tij} = F_{ti2} + \eta_{2tij}, \quad \eta_{2tij} \sim \mathcal{N}(0, 0.1),$$

for $i = 1, \ldots, N_{\text{people}}$, $j = 1, 2, 3$, and $t = 1, \ldots, N_{\text{timepoints}}$.

```
> # Remove all objects
> rm(list = ls())
> # Load necessary libraries
> library(forecast)
> library(rstan)
> library(loo)
> library(ggplot2)
> library(bayesplot)
> # Define parameters for data generation
> n_people <- 10
> n_burnins <- 25
> n_timepoints <- 50
> # Data generation for the latent variable model
> set.seed(123)
> F <- array(0, dim = c(n_people, 2, n_timepoints + n_burnins))
> Y1 <- array(0, dim = c(n_people, 3, n_timepoints))
> Y2 <- array(0, dim = c(n_people, 3, n_timepoints))
> # Coefficients for VAR(1) model
> phi <- matrix(c(0.5, 0.2, -0.3, 0.4), nrow = 2, ncol = 2)
> for (i in 1:n_people) {
+   # Initial values for F
+   F[i, , 1:2] <- matrix(rnorm(4, 0, 1), nrow = 2) # Initialize the first two timepoints
+
+   # Generate latent variables F using VAR(1) model
+   for (t in 3:(n_burnins + n_timepoints)) {
+     F[i, , t] <- phi %*% F[i, , t-1] + rnorm(2, 0, 0.1)
+   }
+
+   # Generate observed data Y1 and Y2 from latent variables F
+   for (j in 1:3) {
+     Y1[i, j, ] <- F[i, 1, (n_burnins + 1):(n_burnins + n_timepoints)] + rnorm(n_timepoints
+     Y2[i, j, ] <- F[i, 2, (n_burnins + 1):(n_burnins + n_timepoints)] + rnorm(n_timepoints
+   }
+ }
> # Prepare data for Stan
> data_list <- list(
+   N_people = n_people,
+   N_timepoints = n_timepoints,
+   Y1 = Y1,
+   Y2 = Y2
+ )
> # Set Stan options
> options(mc.cores = parallel::detectCores())
> rstan_options(auto_write = TRUE)
```

3 Stan Model

The Stan model defines the prior distributions for the parameters, the likelihood of the data given the parameters, and the generated quantities for posterior predictive checks.

3.1 Data Block

The data block specifies the input data for the Stan model:

```
data {
  int<lower=1> N_people;
  int<lower=1> N_timepoints;
  matrix[3, N_timepoints] Y1[N_people];
  matrix[3, N_timepoints] Y2[N_people];
}
```

3.2 Parameters Block

The parameters block defines the parameters to be estimated:

```
parameters {
  matrix[N_timepoints, 2] F[N_people]; // Latent variables F1 and F2
  real<lower=-1, upper=1> phi1;        // AR(1) coefficient for F1
  real<lower=-1, upper=1> phi2;        // AR(1) coefficient for F2
  real<lower=0> sigma[2, 3];           // Standard deviation of observation noise
}
```

3.3 Model Block

The model block defines the prior distributions and the likelihood:

```
model {
  phi1 ~ uniform(-1, 1); // Prior for AR(1) coefficient for F1
  phi2 ~ uniform(-1, 1); // Prior for AR(1) coefficient for F2

  for (i in 1:N_people) {
    for (t in 2:N_timepoints) { // AR(1) model
      F[i, t, 1] ~ normal(phi1 * F[i, t-1, 1], 1);
      F[i, t, 2] ~ normal(phi2 * F[i, t-1, 2], 1);
    }
    F[i, 1, 1] ~ normal(0, 1); // Initial state for F1
    F[i, 1, 2] ~ normal(0, 1); // Initial state for F2

    for (t in 1:N_timepoints) {
      for (k in 1:3) {
        Y1[i, k, t] ~ normal(F[i, t, 1], sigma[1, k]);
```

```
        Y2[i, k, t] ~ normal(F[i, t, 2], sigma[2, k]);
      }
    }
  }
}
```

3.4 Generated Quantities Block

The generated quantities block calculates the log likelihood and posterior predictive distributions:

```
generated quantities {
  vector[N_timepoints] log_lik[N_people];
  matrix[3, N_timepoints] y_hat1[N_people];
  matrix[3, N_timepoints] y_hat2[N_people];

  for (i in 1:N_people) {
    for (t in 1:N_timepoints) {
      log_lik[i, t] = 0;
      for (k in 1:3) {
        y_hat1[i, k, t] = normal_rng(F[i, t, 1], sigma[1, k]);
        y_hat2[i, k, t] = normal_rng(F[i, t, 2], sigma[2, k]);
        log_lik[i, t] += normal_lpdf(Y1[i, k, t] | F[i, t, 1], sigma[1, k]) +
                         normal_lpdf(Y2[i, k, t] | F[i, t, 2], sigma[2, k]);
      }
    }
  }
}
```

```
> # Stan code for latent variable model
> stan_code <- "
+ data {
+   int<lower=1> N_people;
+   int<lower=1> N_timepoints;
+   matrix[3, N_timepoints] Y1[N_people];
+   matrix[3, N_timepoints] Y2[N_people];
+ }
+ parameters {
+   matrix[N_timepoints, 2] F[N_people]; // Latent variables F1 and F2
+   real<lower=-1, upper=1> phi1;        // AR(1) coefficient for F1
+   real<lower=-1, upper=1> phi2;        // AR(1) coefficient for F2
+   real<lower=0> sigma[2, 3];           // Standard deviation of observation noise
+ }
+ model {
+   phi1 ~ uniform(-1, 1); // Prior for AR(1) coefficient for F1
+   phi2 ~ uniform(-1, 1); // Prior for AR(1) coefficient for F2
+
```

```
+   for (i in 1:N_people) {
+     for (t in 2:N_timepoints) { // AR(1) model
+       F[i, t, 1] ~ normal(phi1 * F[i, t-1, 1], 1);
+       F[i, t, 2] ~ normal(phi2 * F[i, t-1, 2], 1);
+     }
+     F[i, 1, 1] ~ normal(0, 1); // Initial state for F1
+     F[i, 1, 2] ~ normal(0, 1); // Initial state for F2
+
+     for (t in 1:N_timepoints) {
+       for (k in 1:3) {
+         Y1[i, k, t] ~ normal(F[i, t, 1], sigma[1, k]);
+         Y2[i, k, t] ~ normal(F[i, t, 2], sigma[2, k]);
+       }
+     }
+   }
+ }
+ generated quantities {
+   vector[N_timepoints] log_lik[N_people];
+   matrix[3, N_timepoints] y_hat1[N_people];
+   matrix[3, N_timepoints] y_hat2[N_people];
+
+   for (i in 1:N_people) {
+     for (t in 1:N_timepoints) {
+       log_lik[i, t] = 0;
+       for (k in 1:3) {
+         y_hat1[i, k, t] = normal_rng(F[i, t, 1], sigma[1, k]);
+         y_hat2[i, k, t] = normal_rng(F[i, t, 2], sigma[2, k]);
+         log_lik[i, t] += normal_lpdf(Y1[i, k, t] | F[i, t, 1], sigma[1, k]) +
+                          normal_lpdf(Y2[i, k, t] | F[i, t, 2], sigma[2, k]);
+       }
+     }
+   }
+ }
+ "

> # Compile Stan model
> stan_model <- stan_model(model_code = stan_code)
```

4 Model Fitting

```
> # Set initial values based on data
> init_values <- function() {
+   F_init <- array(0, dim = c(n_people, n_timepoints, 2))
+   for (i in 1:n_people) {
+     for (t in 1:n_timepoints) {
```

```
+       F_init[i, t, 1] <- mean(data_list$Y1[i, , t])
+       F_init[i, t, 2] <- mean(data_list$Y2[i, , t])
+     }
+   }
+   list(
+     F = F_init,
+     phi1 = 0.5,
+     phi2 = 0.5,
+     sigma = matrix(runif(6, 0.1, 1), nrow = 2)
+   )
+ }
> # Fit the model with better initial values
> fit <- sampling(stan_model, data = data_list, iter = 2000, chains = 4, init = init_values)
> # Extract samples
> samples <- extract(fit)
```

5 Evaluation Metrics

The model is evaluated using several metrics:

5.1 Mean Squared Error (MSE)

The MSE is calculated for each time point and each individual to measure the accuracy of the predicted values:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

5.2 Log Likelihood

The log likelihood is averaged over iterations to assess the model fit:

$$\log L = \frac{1}{T} \sum_{t=1}^{T} \log p(\boldsymbol{y}_t | \boldsymbol{F}_t, \sigma)$$

5.3 Prediction Intervals

95% prediction intervals are calculated for the posterior predictive distribution to assess the model uncertainty:

$$\hat{y}_{1,\text{lower}} = \text{quantile}(y_{\text{hat1}}, 0.025)$$

$$\hat{y}_{1,\text{upper}} = \text{quantile}(y_{\text{hat1}}, 0.975)$$

```
> # Obtain the mean of the posterior predictive distribution
> y_hat1_mean <- apply(samples$y_hat1, c(2, 3, 4), mean) # Result is (n_people, 3, n_timepoi
> y_hat2_mean <- apply(samples$y_hat2, c(2, 3, 4), mean) # Result is (n_people, 3, n_timepoi
> # Define function to calculate MSE
> calculate_mse <- function(y_hat, y_true) {
+   return(mean((y_hat - y_true)^2))
+ }
> # Calculate MSE
> mse1 <- array(0, dim = c(n_people, n_timepoints))
> mse2 <- array(0, dim = c(n_people, n_timepoints))
> for (i in 1:n_people) {
+   for (t in 1:n_timepoints) {
+     mse1[i, t] <- calculate_mse(y_hat1_mean[i, , t], data_list$Y1[i, , t])
+     mse2[i, t] <- calculate_mse(y_hat2_mean[i, , t], data_list$Y2[i, , t])
+   }
+ }
> # Plot MSE
> mse_df1 <- data.frame(
+   Time = rep(1:n_timepoints, n_people),
+   MSE = as.vector(mse1),
+   Person = rep(1:n_people, each = n_timepoints)
+ )
> mse_df2 <- data.frame(
+   Time = rep(1:n_timepoints, n_people),
+   MSE = as.vector(mse2),
+   Person = rep(1:n_people, each = n_timepoints)
+ )
> # Save MSE plots
> pdf("mse_plot_y1.pdf")
> ggplot(mse_df1, aes(x = Time, y = MSE, color = as.factor(Person))) +
+   geom_line() +
+   labs(title = "MSE for Y1", x = "Time", y = "MSE", color = "Person")
> dev.off()
> pdf("mse_plot_y2.pdf")
> ggplot(mse_df2, aes(x = Time, y = MSE, color = as.factor(Person))) +
+   geom_line() +
+   labs(title = "MSE for Y2", x = "Time", y = "MSE", color = "Person")
> dev.off()
> # Plot posterior distributions of VAR coefficients and Cholesky factors
> posterior <- as.array(fit)
> pdf("posterior_distributions_VAR.pdf")
> mcmc_areas(posterior, pars = c("phi1", "phi2"))
> dev.off()
> # Plot predicted vs observed values for the first person
> pdf("predicted_vs_observed_y1.pdf")
> person_index <- 1
```

```
> for (k in 1:3) {
+   plot(1:n_timepoints, data_list$Y1[person_index, k, ], type = "l", col = "blue", ylim = r
+   lines(1:n_timepoints, y_hat1_mean[person_index, k, ], col = "red")
+   title(paste("Observed vs Predicted Y1 for Person", person_index, "Variable", k))
+   legend("topright", legend = c("Observed", "Predicted"), col = c("blue", "red"), lty = 1)
+ }
> dev.off()
> pdf("predicted_vs_observed_y2.pdf")
> person_index <- 1
> for (k in 1:3) {
+   plot(1:n_timepoints, data_list$Y2[person_index, k, ], type = "l", col = "blue", ylim = r
+   lines(1:n_timepoints, y_hat2_mean[person_index, k, ], col = "red")
+   title(paste("Observed vs Predicted Y2 for Person", person_index, "Variable", k))
+   legend("topright", legend = c("Observed", "Predicted"), col = c("blue", "red"), lty = 1)
+ }
> dev.off()
> # Extract log likelihood
> log_lik <- extract_log_lik(fit)
> # Check the structure of log_lik
> str(log_lik)  # Should show a matrix with dimensions (iterations, 10 * 50)
> # Reshape log_lik to (iterations, n_people, n_timepoints)
> iterations <- dim(log_lik)[1]
> log_lik_reshaped <- array(log_lik, dim = c(iterations, n_people, n_timepoints))
> # Average log likelihood over iterations
> log_lik_mean <- apply(log_lik_reshaped, c(2, 3), mean)  # Average over iterations, keeping
> # Optionally, average over people to get a single time series
> log_lik_mean_over_people <- apply(log_lik_mean, 2, mean)
> # Plot log likelihood over time
> pdf("log_likelihood_over_time.pdf")
> plot(1:n_timepoints, log_lik_mean_over_people, type = "l", col = "black", ylab = "Log Like
> title("Log Likelihood over Time")
> dev.off()
> # Calculate 95% prediction intervals
> y_hat1_lower <- apply(samples$y_hat1, c(2, 3, 4), quantile, probs = 0.025)
> y_hat1_upper <- apply(samples$y_hat1, c(2, 3, 4), quantile, probs = 0.975)
> # Plot prediction intervals vs observed values for the first person
> pdf("prediction_intervals_y1.pdf")
> for (k in 1:3) {
+   plot(1:n_timepoints, data_list$Y1[person_index, k, ], type = "l", col = "blue", ylim = r
+   lines(1:n_timepoints, y_hat1_mean[person_index, k, ], col = "red")
+   lines(1:n_timepoints, y_hat1_lower[person_index, k, ], col = "grey", lty = 2)
+   lines(1:n_timepoints, y_hat1_upper[person_index, k, ], col = "grey", lty = 2)
+   title(paste("Observed vs Predicted Y1 with Prediction Interval for Person", person_index
+   legend("topright", legend = c("Observed", "Predicted", "95% Interval"), col = c("blue",
+ }
> dev.off()
```

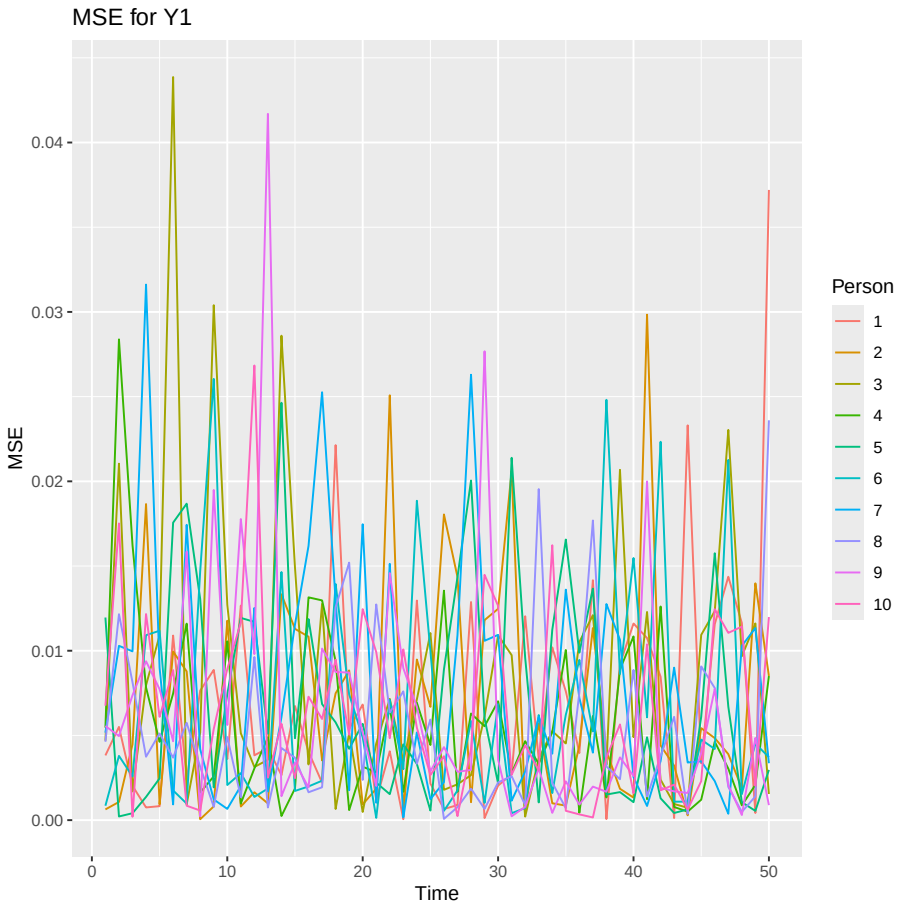

Figure 1: MSE for Y1

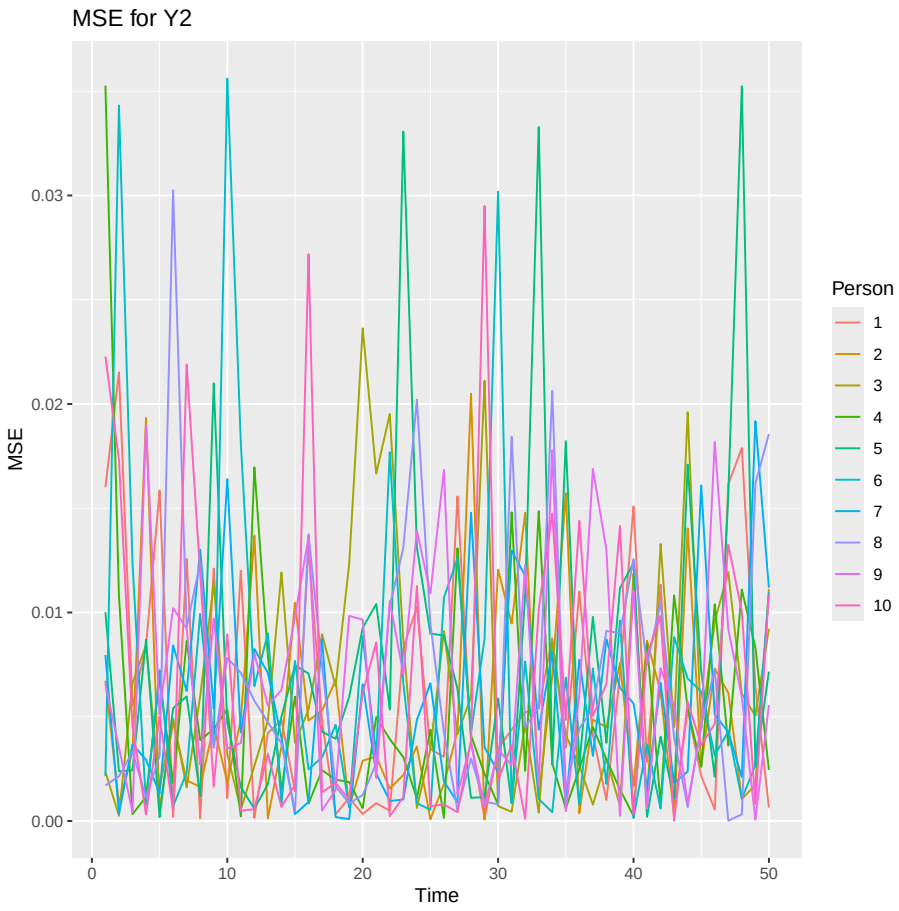

Figure 2: MSE for Y2

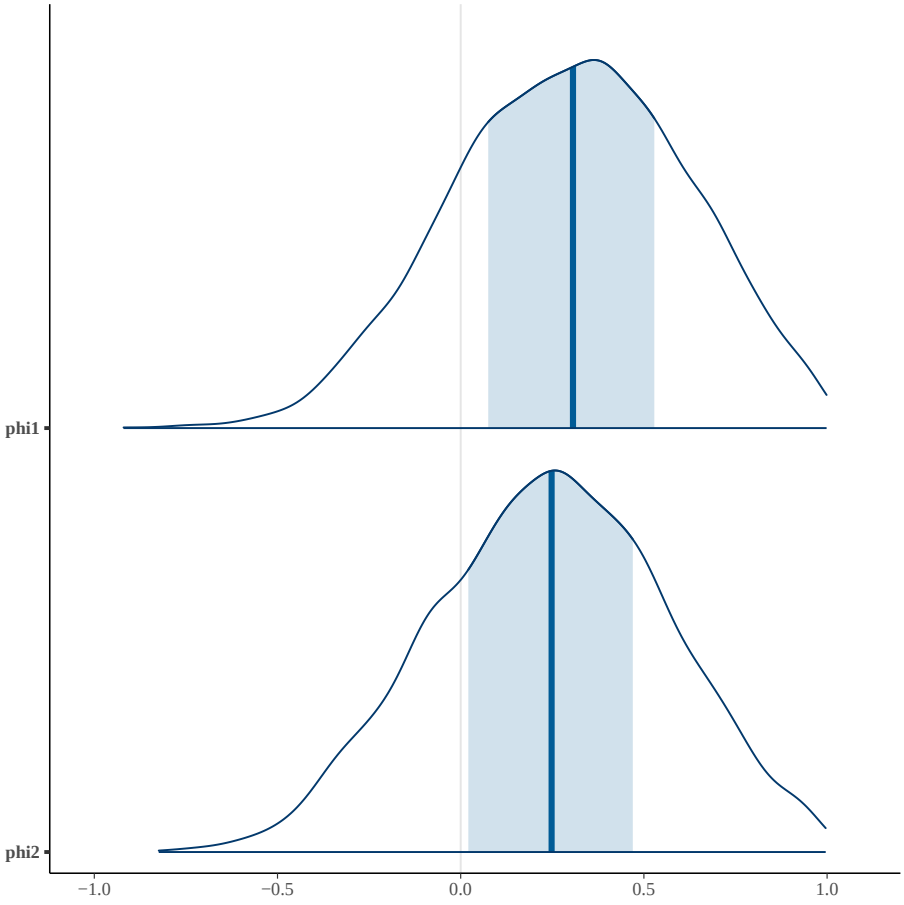

Figure 3: Posterior Distributions of VAR Coefficients and Cholesky Factors

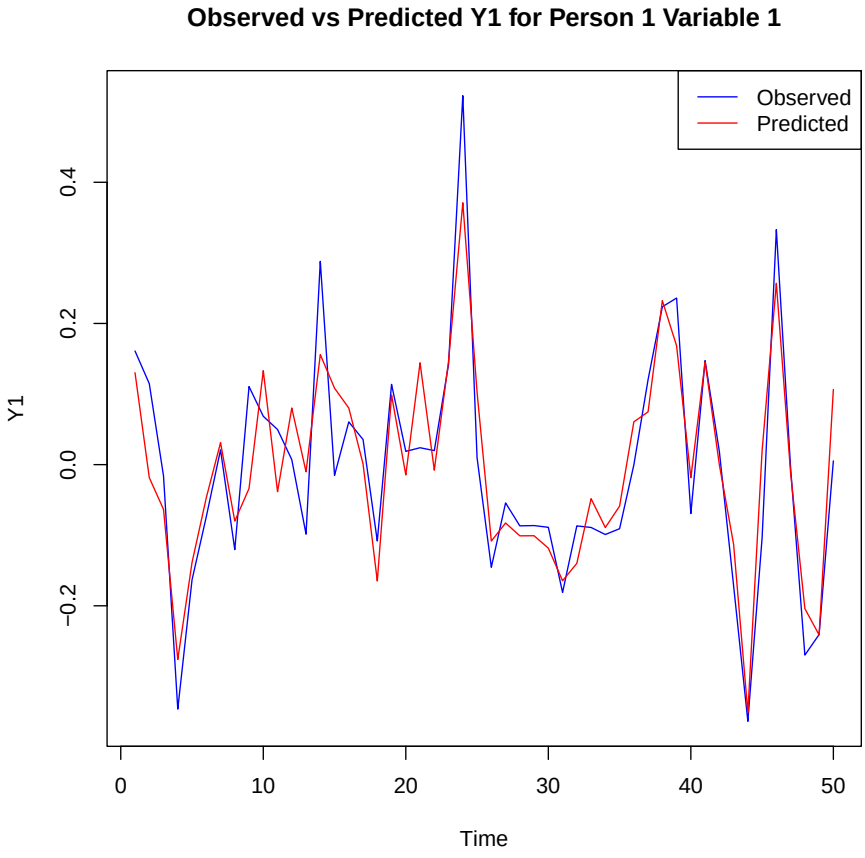

Figure 4: Observed vs Predicted Y1 for Person 1

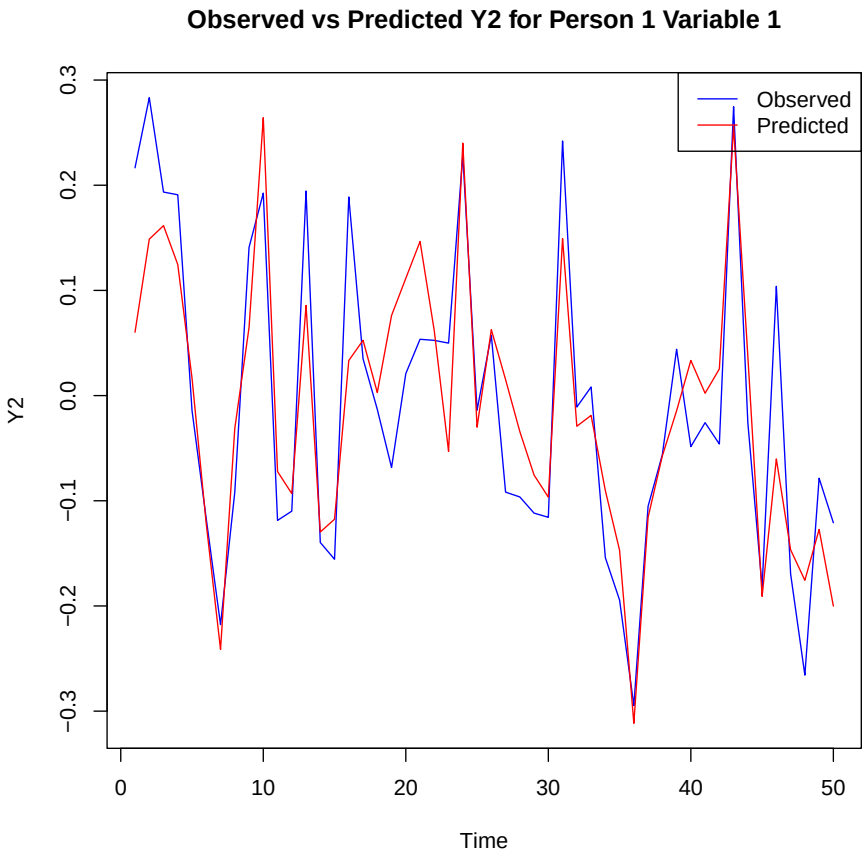

Figure 5: Observed vs Predicted Y1 for Person 2

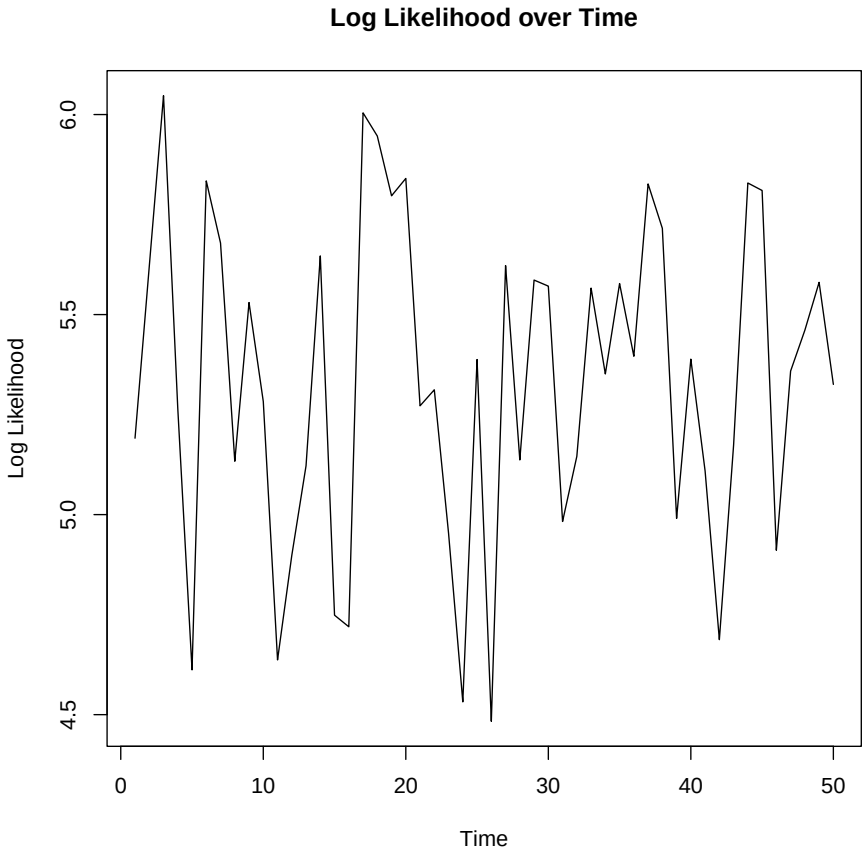

Figure 6: Log Likelihood over Time

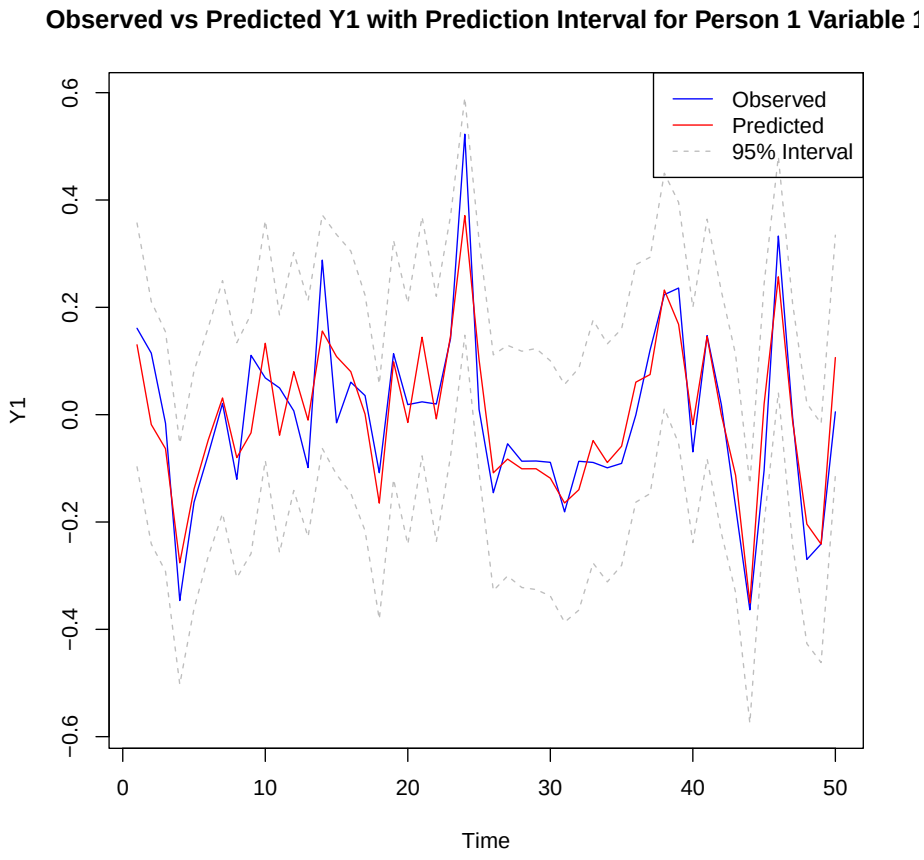

Figure 7: Observed vs Predicted Y1 with Prediction Interval for Person 1