Modeling Delay Propagation in Vancouver Bus Transit Network using Bayesian Methods

STAT 447C: Bayesian Statistics

Submitted by:

Utsav Singh
https://github.com/Utsav02/TranslinkBayes

Instructor:

Alexandre Bouchard-Côté

Department of Statistics

The University of British Columbia

April 19, 2025

Summary

Metro Vancouver's bus transit system, operated by TransLink, is one of the most comprehensive in North America. However, like all other urban transit systems, it experiences unpredictable delays due to congestion, weather, and operational inefficiencies. This project explores how Bayesian inference can be applied to model delay propagation in a high-frequency bus network of Vancouver using real-time GTFS data. The objective is to develop a probabilistic framework for delay prediction while capturing uncertainty, leveraging historical delay data and hierarchical Bayesian modeling.

Literature Review

Rodriguez-Deniz and Villani (2022) introduced a robust Bayesian framework for real-time bus delay prediction across high-frequency transit systems. Their model accounted for the heavy-tailed nature of delay distributions using a Student-t likelihood and included hierarchical structure to capture variations at the trip and stop levels. Most interesting was their use of geometrically discounted past delays, and dynamic modeling of the scale and degrees-of-freedom parameters which allowed them to model time-varying uncertainty.

In contrast, this implementation adopts and simplifies the core components of their framework. A Student-t likelihood is used to maintain robustness to outliers and incorporate random effects for both trips and stops. Covariates such as previous stop delay, hour of the day, and day of the week are included and extend the model with an autoregressive (AR(1)) term to partially capture short-term temporal dependencies. To simplify and test the implementation first, instead of using a weighted lag structure that Rodriguez et al. (2022) used, this implementation just uses a single lag. This assumption in simpler logic can be framed as: a bus' delay is affected by how late it was on the previous stop.

Problem Formulation

The core inference task in this project is to predict bus delays at individual stops using a Bayesian model that incorporates spatial, temporal, and historical records. Given observed features such as prior stop delay, distance, and time-based covariates (hour and weekday), the model will try to estimate the current delay and provide uncertainty estimates for this prediction. The model outputs can assist both transit operators and riders by forecasting potential lateness and its likelihood, which is crucial in high-frequency urban bus networks.

For the project, this implementation fits under the **Bayesian regression and time series modeling** theme, and extends it by introducing hierarchical structure (trip and stop-level variation), temporal dependence through an AR(1) term, and spatial features through distance traveled. Given the real-world nature of the problem and the availability of large, noisy datasets, and using Rodriguez-Deniz and Villani (2022) as the reference, the use of a robust Student-t likelihood further strengthens the model's reliability.

Dataset Description

The project makes use of data obtained through the real-time and static GTFS API from TransLink. Static data comes from TransLink's website (https://www.translink.ca/about-us/doing-business-with-translink/app-developer-resources/gtfs/gtfs-data) and the real-time data is stored by a python script running on the local computer every five minutes in an SQLite database. For this report, only one route's data has been modeled: Route 6641 for Bus 99.

trip_id	route_id	stop_id	stop_sequence	actual_arrival	delay_seconds	previous_stop_delay	shape_dist_traveled
14265500	6641	271	2	2025-04-03 14:15:16+00:00	252	0	0.3567

Model Specification

The delay at each bus stop i is modeled using a Student-t likelihood to account for the heavy-tailed nature of the data:

$$y_i \sim \text{Student-}t(\nu, \mu_i, \sigma)$$

- y_i is the observed delay (in seconds) at bus stop i,
- μ_i is the expected delay predicted by the model for stop i,
- ν is the degrees of freedom parameter, allowing the distribution to capture outliers,
- σ is the standard deviation of the Student-t distribution.

The expected delay μ_i (whose values are positive for delay, negative for early, and zero for on time) is modeled as a linear combination of spatial, temporal, and hierarchical effects:

$$\mu_i = \alpha + \beta_{\text{prev}} \cdot \text{prev-delay}_i + \beta_{\text{dist}} \cdot \text{distance}_i + \beta_{\text{hour}[h_i]} + \beta_{\text{dow}[d_i]} + \text{trip-effect}_{t_i} + \text{stop-effect}_{s_i} + \phi \cdot \text{prev-delay}_i$$

- α is the intercept representing the average delay across the system.
- β_{prev} is the coefficient for the delay at the previous stop reflecting how upstream delays propagate.
- β_{dist} is the coefficient for the cumulative shape distance from the trip's origin (distance traveled (km)).
- $\beta_{\text{hour}[h_i]}$ is a coefficient for the hour of the day to account for temporal variation like rush hours.
- $\beta_{\text{dow}[d_i]}$ is a coefficient for the day of the week, accounting for weekday vs. weekend effects.
- trip-effect_{t_i} is a random effect for each trip t_i, capturing trip level variability.
- stop-effect_{s_i} is a random effect for each stop s_i, capturing spatial variation based on each stop.
- $\phi \cdot \text{prev-delay}_i$: AR(1) dependecny modeling autocorrelation between consecutive delays within a trip.

  This model structure allows for both fixed and random effects, enabling interpretable inference while accommodating the hierarchical and noisy nature of transit delay data. The Student-t distribution provides robustness to outliers, which is more than a common occurrence in the real-world due to many reasons.

Prior Choices

To ensure the model remains interpretable, following prior distributions for each parameter were used:

- $\beta_{\cdot} \sim \mathcal{N}(0, 2)$:
  A weakly informative Gaussian prior is placed on all regression coefficients. This centers the coefficients around zero, encouraging simplicity while still allowing sufficient flexibility for the model to learn from the data. The standard deviation of 2 was selected to allow moderate deviation from zero, based on choices in related work (Rodriguez-Deniz & Villani, 2022 used $\mathcal{N}(0, 1)$).
- $\sigma, \sigma_{\text{trip}}, \sigma_{\text{stop}} \sim \text{Exponential}(1)$:
  Exponential priors are placed on the scale parameters for the observation noise and the standard deviations of the trip-level and stop-level random effects. These priors are appropriate for scale parameters because they constrain values to be positive while favoring smaller magnitudes. This encourages shrinkage of the random effects when data is not informative, thus preventing overfitting.

- $\nu \sim \text{Gamma}(2, 0.1)$:
  A gamma prior is used for the degrees of freedom in the Student-t distribution. The chosen parameters encourage values close to 2, which resembles heavy-tailed behavior, making the model robust to outliers.
- $\phi \sim \text{Uniform}(-1, 1)$:
  Based on time-series course, this prior is placed on the autoregressive parameter. The uniform distribution ensures the parameter stays within the stationarity region of AR(1) models (i.e., $|\phi| < 1$).

Model Diagnostics

Two models were compared in this project. A baseline model excluding both stop-level random effects and autoregressive (AR(1)) components was fitted. This was followed by an extended model that incorporated both terms to capture spatial and temporal dependencies in delay propagation. Several comparisons were performed after checking that the chains mixed well. Below are the results and interpretation of them.

Posterior Parameter Comparison

Table 1: Posterior Means for Selected Parameters (Base vs Extended Model)

Parameters	Base	Extended
alpha	12.27	2.10
beta_prev	0.97	0.91
beta_dist	-1.48	-1.16
sigma	50.93	24.96
nu	2.71	2.02
phi	NA	0.08

Interpretation

- **Intercept**: The baseline has a much larger intercept (12.27) compared to the extended model (2.10). We can account this huge difference on the fact that the extended model distributes baseline variability more realistically across trip and stop-level effects and the baseline has to compensate for it.
- **Previous Delay Coefficient**: Both models show positive coefficients (0.91, 0.97), indicating that delays tend to propagate. The slightly lower coefficient in the extended model may result from shared predictive power with AR(1) and stop-level effects.
- **Distance Coefficient**: Both coefficients are negative, implying stops on route far away from origin are associated with smaller delays.
- **Residual Standard Deviation (sigma):** It is substantially lower in the extended model, suggesting that introducing stop-level and AR(1) dependence has significantly improved the model's ability to capture variability in delay.
- **Degrees of Freedom**: When ν is smaller, the distribution has heavy tail. It is consistent with real-world delay distributions that include extreme cases.
- **Autoregressive Coefficient**: Present only in the extended model (0.08), it captures autocorrelation in delays. We see that it adds marginal predictive strength beyond previous delay.

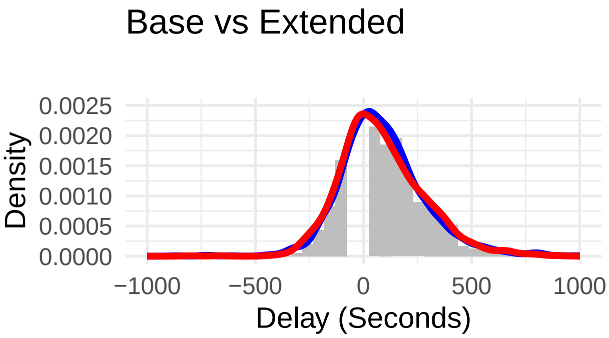
Leave One Out (LOO):

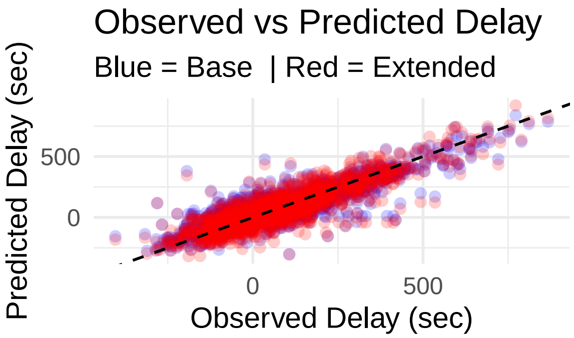
Table 2: LOO Summary for Base and Extended Models

Model	ELPD LOO	SE	P_LOO	LOOIC
Base	-8808.45	42.03	7.48	17616.90
Extended	-8057.62	54.44	41.69	16115.24

- The extended model outperforms the base model in predictive accuracy, as shown by its higher ELPD LOO (expected log predictive density) and lower LOOIC (leave-one-out information criterion). This means the extended model is better at predicting unseen data while accounting for uncertainty. The extended model also has a higher effective number of parameters (p_loo), suggesting it is more flexible and complex. This can be considered a trade-off which implies that the extended model captures more structure in the data such as stop-level and autocorrelated effects leading to improved predictive performance but more complex.

Visual Analysis using Posterior Distribution Check and Residuals





Based on the two plots, we can say that the baseline model performs decent, closely matching the observed delay distribution (left-side) and aligning predicted values along the diagonal line (right-side). The distribution check shows that both models replicate the observed distribution effectively. In the scatter plot, both models show tight clustering around the diagonal, indicating accurate predictions.

Limitations

This project has several limitations, primarily stemming from data quality and scope. The dataset was collected using a custom script, which occasionally failed, introducing missing or inconsistent data points. Additionally, the analysis focused on a single bus route (Route 6641 Bus 99), and results may not generalize to other routes. The autoregressive component was limited to a single lag (AR(1)), compared to what Rodriguez et al, (2022) used.

Conclusion

This analysis demonstrates that while the extended model, with added stop-level effects and an AR(1) temporal structure, offers a slightly improved fit, the gain over the base model is modest. The base model already captures the core patterns in delay propagation using simpler hierarchical components, such as trip-level effects and key covariates. This highlights the strength and practicality of Bayesian models in capturing meaningful transit dynamics while keeping it simple. Nonetheless, the extended model provides more insight and robustness to extreme delays, which can be valuable for domain experts.

References

Rodriguez-Deniz, H., & Villani, M. (2022). Robust Real-Time Delay Predictions in a Network of High-Frequency Urban Buses. *IEEE Transactions on Intelligent Transportation Systems*, 23(9), 16305–16318. DOI: 10.1109/TITS.2022.3146759.

Format for the report was taken from STAT 450: Case Studies in Statistics

Appendix

There is a section where after downloading the models from github, one can explore without having to fit it again as it took a long time to compile given the amount of data.

Loading the Data

```r
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
library(dplyr)
library(DBI)
library(RSQLite)
library(ggplot2)
library(lubridate)
library(rstan)
library(loo)
```

```r
# Connect to database
conn <- dbConnect(SQLite(), "../database/gtfs_realtime.db")
stop_delays <- dbReadTable(conn, "stop_delays")
dbDisconnect(conn)

conn_static <- dbConnect(SQLite(), "../database/gtfs_static.db")
stop_times <- dbReadTable(conn_static, "stop_times")
trips <- dbReadTable(conn_static, "trips")
dbDisconnect(conn_static)
```

```r
# Connect to database
conn <- dbConnect(SQLite(), "../database/gtfs_realtime.db")
stop_delays <- dbReadTable(conn, "stop_delays")
dbDisconnect(conn)

conn_static <- dbConnect(SQLite(), "../database/gtfs_static.db")
stop_times <- dbReadTable(conn_static, "stop_times")
trips <- dbReadTable(conn_static, "trips")
dbDisconnect(conn_static)
```

```r
stop_delays <- stop_delays %>%
  mutate(
    timestamp = as.POSIXct(timestamp, tz = "UTC"),
    delay_seconds = as.numeric(delay_seconds),
    stop_sequence = as.integer(stop_sequence),
    previous_stop_delay = as.numeric(previous_stop_delay)
  )

stop_times <- stop_times %>%
  mutate(
    trip_id = as.character(trip_id),
    stop_id = as.character(stop_id),
    stop_sequence = as.integer(stop_sequence),
    shape_dist_traveled = as.numeric(shape_dist_traveled)
  )

trips <- trips %>%
  mutate(
    trip_id = as.character(trip_id),
    direction_id = as.factor(direction_id)
  )

latest_timestamp <- max(stop_delays$timestamp, na.rm = TRUE)
filtered_delays <- stop_delays #%>%
  #filter(timestamp >= latest_timestamp - days(30))

deduped_delays <- filtered_delays %>%
  group_by(trip_id, stop_id) %>%
  slice_max(order_by = timestamp, n = 1) %>%
  ungroup()

deduped_delays <- deduped_delays %>%
  mutate(
    trip_id = as.character(trip_id),
    stop_id = as.character(stop_id)
  )
```

```r
merged_data <- deduped_delays %>%
  left_join(stop_times %>% dplyr::select(trip_id, stop_id, shape_dist_traveled, stop_sequence),
            by = c("trip_id", "stop_id", "stop_sequence")) %>%
  left_join(trips %>% dplyr::select(trip_id, direction_id), by = "trip_id") %>%
  mutate(
    direction_id = as.factor(direction_id)
  )

merged_data <- merged_data %>%
  group_by(trip_id) %>%
  mutate(
    shape_dist_traveled = shape_dist_traveled - min(shape_dist_traveled, na.rm = TRUE)
  ) %>%
  ungroup()


merged_data <- merged_data %>%
  mutate(
    hour = hour(timestamp),
    day_of_week = wday(timestamp, label = TRUE, abbr = TRUE),
    hour_day = interaction(hour, day_of_week, sep = "_"),
    is_rush_hour = case_when(
      hour %in% 7:9 | hour %in% 16:18 ~ TRUE,
      TRUE ~ FALSE
    ),
    is_weekend = day_of_week %in% c("Sat", "Sun"),
  )
```

```
filter_data <- function(merged_data){
  merged_data %>%
  filter(!is.na(delay_seconds),
         !is.na(previous_stop_delay),
         !is.na(stop_sequence),
         !is.na(hour),
         !is.na(shape_dist_traveled),
         !is.na(trip_id),
         !is.na(stop_id)) %>%
  mutate(
    trip_index = as.integer(as.factor(trip_id)),
    stop_index = as.integer(as.factor(stop_id)),
    hour = as.integer(hour) + 1,
    dow = as.integer(as.factor(day_of_week))
  )
}
# this was done after the final model was ran just to compare if direction did anything
filtered_data_dir_0 <- filter_data(merged_data_dir_0)
filtered_data_dir_1 <- filter_data(merged_data_dir_1)
```

```
## this was the data used for training attaching in github for reproducability
data <- read.csv("data.csv")
prepare_stan_data <- function(filtered_data){
    list(
    N = nrow(filtered_data),
    delay = filtered_data$delay_seconds,
    prev_delay = filtered_data$previous_stop_delay,
    distance = filtered_data$shape_dist_traveled,
    hour = filtered_data$hour + 1,
    dow = filtered_data$dow,
    trip = filtered_data$trip_index,
    stop = filtered_data$stop_index,
    N_trips = length(unique(filtered_data$trip_index)),
    N_stops = length(unique(filtered_data$stop_index)),
    max_hour = max(filtered_data$hour) + 1,
    max_dow = max(filtered_data$dow)
  )
}

stan_data_dir_0 <- prepare_stan_data(data)
#stan_data_dir_1 <- prepare_stan_data(filtered_data_dir_1)
```

Stan Code

```stan
// delay_model_hier.stan
data {
  int<lower=0> N;
  vector[N] delay;
  vector[N] prev_delay;
  vector[N] distance;
  int<lower=1> hour[N];
  int<lower=1> dow[N];
  int<lower=1> trip[N];
  int<lower=1> N_trips;
  int<lower=1> max_hour;
  int<lower=1> max_dow;
}
parameters {
  real alpha;
  real beta_prev;
  real beta_dist;
  vector[max_hour] beta_hour;
  vector[max_dow] beta_dow;
  vector[N_trips] trip_effect;
  real<lower=0> sigma;
  real<lower=0> sigma_trip;
  real<lower=2> nu;
}
model {
  vector[N] mu;

  alpha ~ normal(0, 10);
  beta_prev ~ normal(0, 2);
  beta_dist ~ normal(0, 2);
  beta_hour ~ normal(0, 2);
  beta_dow ~ normal(0, 2);
  trip_effect ~ normal(0, sigma_trip);
  sigma ~ exponential(1);
  sigma_trip ~ exponential(1);
  nu ~ gamma(2, 0.1);

  for (n in 1:N) {
    mu[n] = alpha + beta_prev * prev_delay[n] + beta_dist * distance[n] +
            beta_hour[hour[n]] + beta_dow[dow[n]] + trip_effect[trip[n]];
  }
  delay ~ student_t(nu, mu, sigma);
}
generated quantities {
  vector[N] log_lik;
  vector[N] y_rep;
  for (i in 1:N) {
    real mu_i = alpha
              + beta_prev * prev_delay[i] + beta_dist * distance[i]
              + beta_hour[hour[i]] + beta_dow[dow[i]] + trip_effect[trip[i]];
    log_lik[i] = student_t_lpdf(delay[i] | nu, mu_i, sigma);
    y_rep[i] = student_t_rng(nu, mu_i, sigma);
  }
```

```
}


// delay_model_ar_spatial.stan
data {
  int<lower=0> N;
  vector[N] delay;
  vector[N] prev_delay;
  vector[N] distance;
  int<lower=1> hour[N];
  int<lower=1> dow[N];
  int<lower=1> trip[N];
  int<lower=1> stop[N];
  int<lower=1> N_trips;
  int<lower=1> N_stops;
  int<lower=1> max_hour;
  int<lower=1> max_dow;
}
parameters {
  real alpha;
  real beta_prev;
  real beta_dist;
  vector[max_hour] beta_hour;
  vector[max_dow] beta_dow;
  vector[N_trips] trip_effect;
  vector[N_stops] stop_effect;
  real<lower=0> sigma;
  real<lower=0> sigma_trip;
  real<lower=0> sigma_stop;
  real<lower=2> nu;
  real<lower=-1, upper=1> phi;
}
model {
  vector[N] mu;

  alpha ~ normal(0, 10);
  beta_prev ~ normal(0, 2);
  beta_dist ~ normal(0, 2);
  beta_hour ~ normal(0, 2);
  beta_dow ~ normal(0, 2);
  trip_effect ~ normal(0, sigma_trip);
  stop_effect ~ normal(0, sigma_stop);
  sigma ~ exponential(1);
  sigma_trip ~ exponential(1);
  sigma_stop ~ exponential(1);
  nu ~ gamma(2, 0.1);
  phi ~ uniform(-1, 1);

  for (n in 1:N) {
    mu[n] = alpha + beta_prev * prev_delay[n] + beta_dist * distance[n] +
            beta_hour[hour[n]] + beta_dow[dow[n]] +
            trip_effect[trip[n]] + stop_effect[stop[n]] + phi * prev_delay[n];
  }
  delay ~ student_t(nu, mu, sigma);
}
generated quantities {
  vector[N] log_lik;
  vector[N] y_rep;
  for (i in 1:N) {
    real mu_i = alpha
              + beta_prev * prev_delay[i]
              + beta_dist * distance[i] + beta_hour[hour[i]] + beta_dow[dow[i]]
              + trip_effect[trip[i]] + stop_effect[stop[i]]  + phi * prev_delay[i];
    log_lik[i] = student_t_lpdf(delay[i] | nu, mu_i, sigma);
    y_rep[i] = student_t_rng(nu, mu_i, sigma);
  }
}
```

```
options(mc.cores = 2)
model_base <- stan_model(file = "delay_model_hier.stan")
model_extended <- stan_model(file = "delay_model_ar_spatial.stan")
```

```
run_and_save_models <- function(stan_data, route_choice, dir_choice,
                                iter = 2000, chains = 4, seed = 6286, thin = 2) {

  control_params <- list(adapt_delta = 0.99, max_treedepth = 12)

  message(glue::glue("Starting model fitting for route {route_choice}, direction {dir_choice}"))
  overall_start <- Sys.time()

#Base model
  start_base <- Sys.time()
  fit_base <- sampling(
    model_base,
    data = stan_data,
    iter = iter,
    chains = chains,
    seed = seed,
    thin = thin,
    control = control_params,
    init_r = 0.5
  )
  end_base <- Sys.time()
  message(glue::glue("Base model finished in {round(difftime(end_base, start_base, units = 'mins'), 2)}
  saveRDS(fit_base, glue::glue("models/fit_base_route{route_choice}_dir{dir_choice}.rds"))

  # Extended model
  start_extended <- Sys.time()
  fit_extended <- sampling(
    model_extended,
    data = stan_data,
    iter = iter,
    chains = chains,
    seed = seed,
    thin = thin,
    control = control_params,
```

```r
    init_r = 0.5
  )
  end_extended <- Sys.time()
  message(glue::glue("Extended model finished in {round(difftime(end_extended, start_extended, units =

  saveRDS(fit_extended, glue::glue("models/fit_extended_route{route_choice}_dir{dir_choice}.rds"))

  overall_end <- Sys.time()
  message(glue::glue("Total time: {round(difftime(overall_end, overall_start, units = 'mins'), 2)} minu

  return(list(base = fit_base, extended = fit_extended))
}
```

```r
fits_0 <- run_and_save_models(stan_data_dir_0, route_choice = 6641, dir_choice = 0)
```

Run from here to explore model

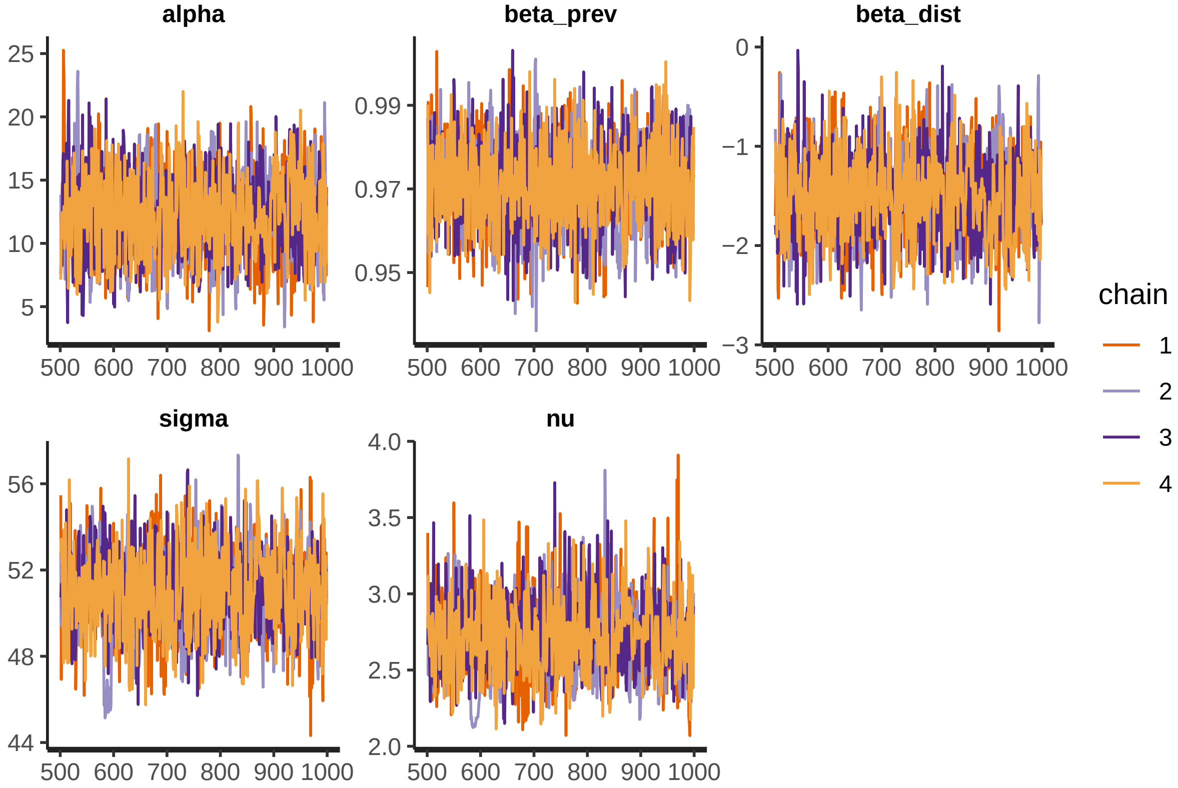
```r
library(rstan)
library(ggplot2)
library(knitr)
library(dplyr)
library(lubridate)
library(bayesplot)
fit_base = readRDS("models/fit_base_1000.rds")
fit_extended = readRDS("models/fit_extended_1000.rds")

params <- c("alpha", "beta_prev", "beta_dist", "sigma", "nu", "phi")
summ_base <- summary(fit_base, pars = intersect(params, names(fit_base@sim$samples[[1]])))$summary
summ_ext <- summary(fit_extended, pars = intersect(params, names(fit_extended@sim$samples[[1]])))$summa

param_table <- data.frame(
  Parameters = rownames(summ_ext),
  Base = ifelse(rownames(summ_ext) %in% rownames(summ_base), summ_base[, "mean"], NA),
  Extended = summ_ext[, "mean"],
  check.names = FALSE
)
kable(param_table, caption = "Posterior Means for Selected Parameters (Base vs Extended Model)", digits
```

Table 3: Posterior Means for Selected Parameters (Base vs Extended Model)

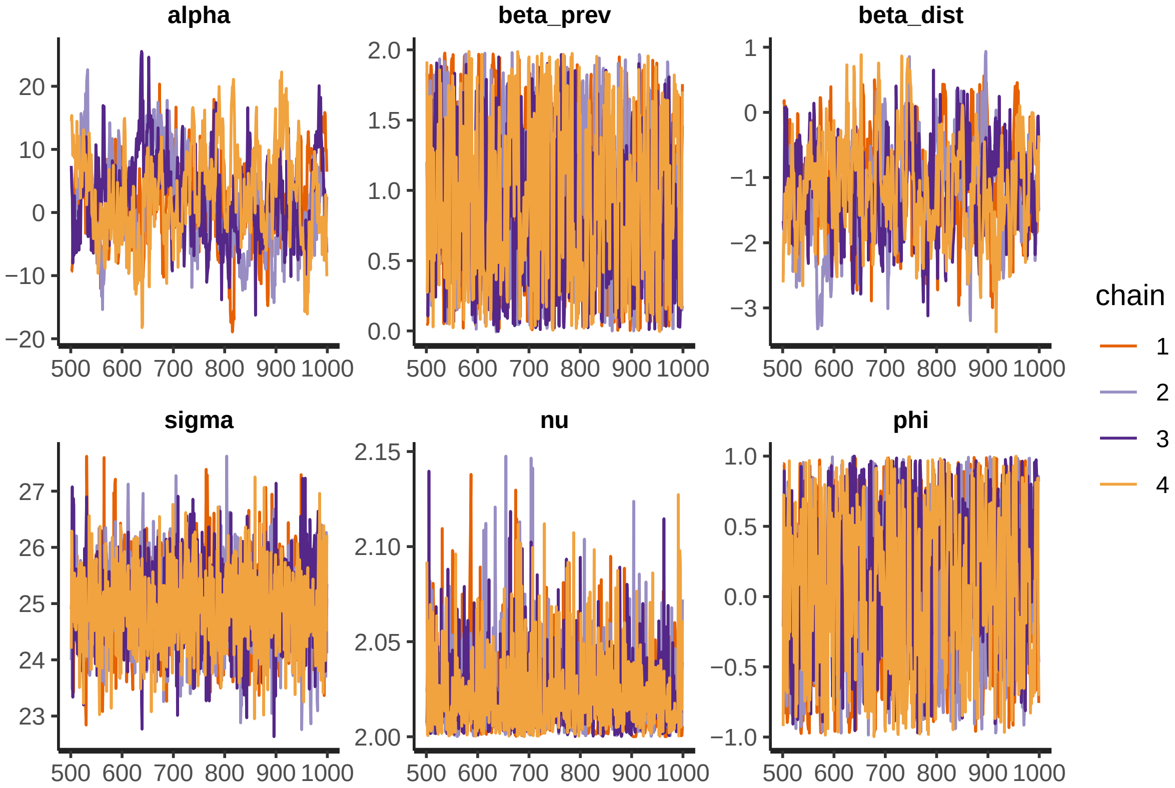
Parameters	Base	Extended
alpha	12.27	2.10
beta_prev	0.97	0.91
beta_dist	-1.48	-1.16
sigma	50.93	24.96
nu	2.71	2.02
phi	NA	0.08

Examples of plots that were explored but not added in report.

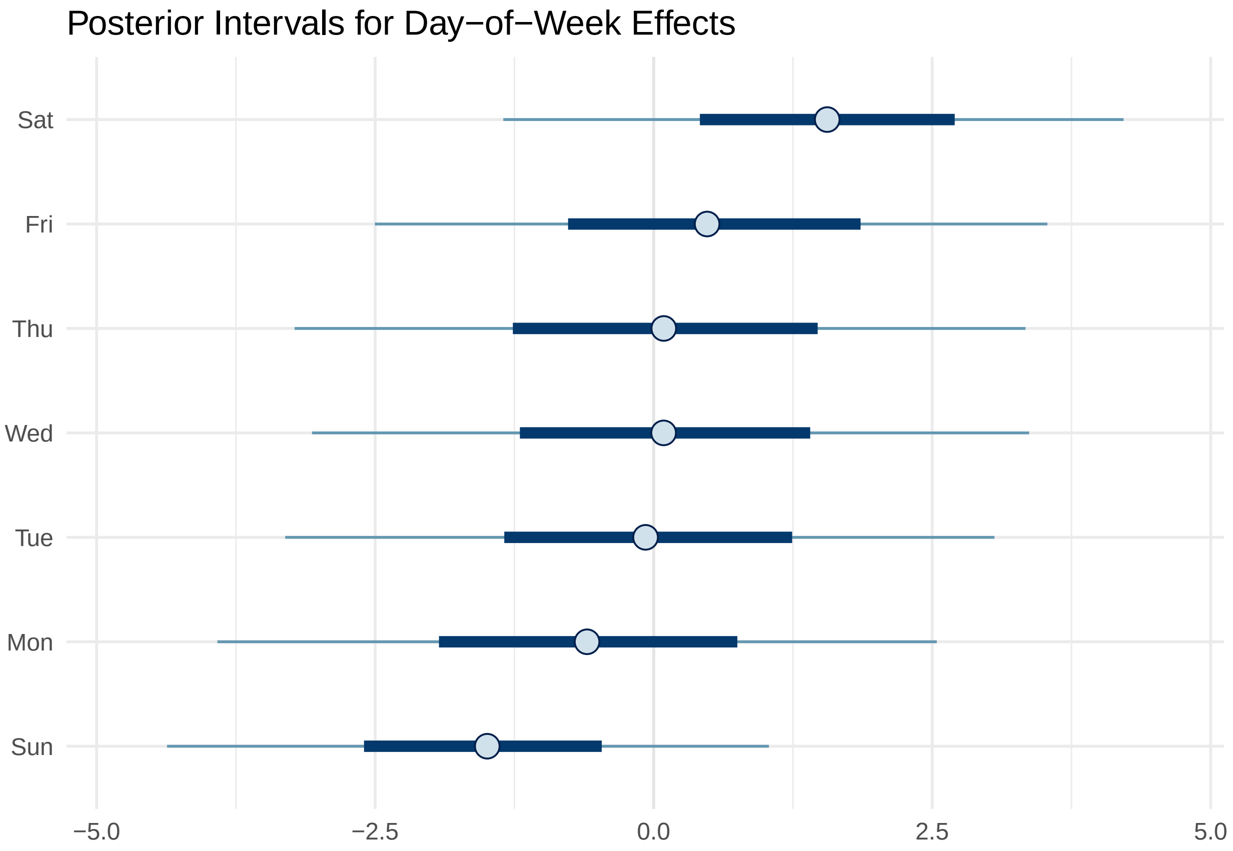
```
traceplot(fit_base, pars = c("alpha", "beta_prev", "beta_dist", "sigma", "nu"))
```



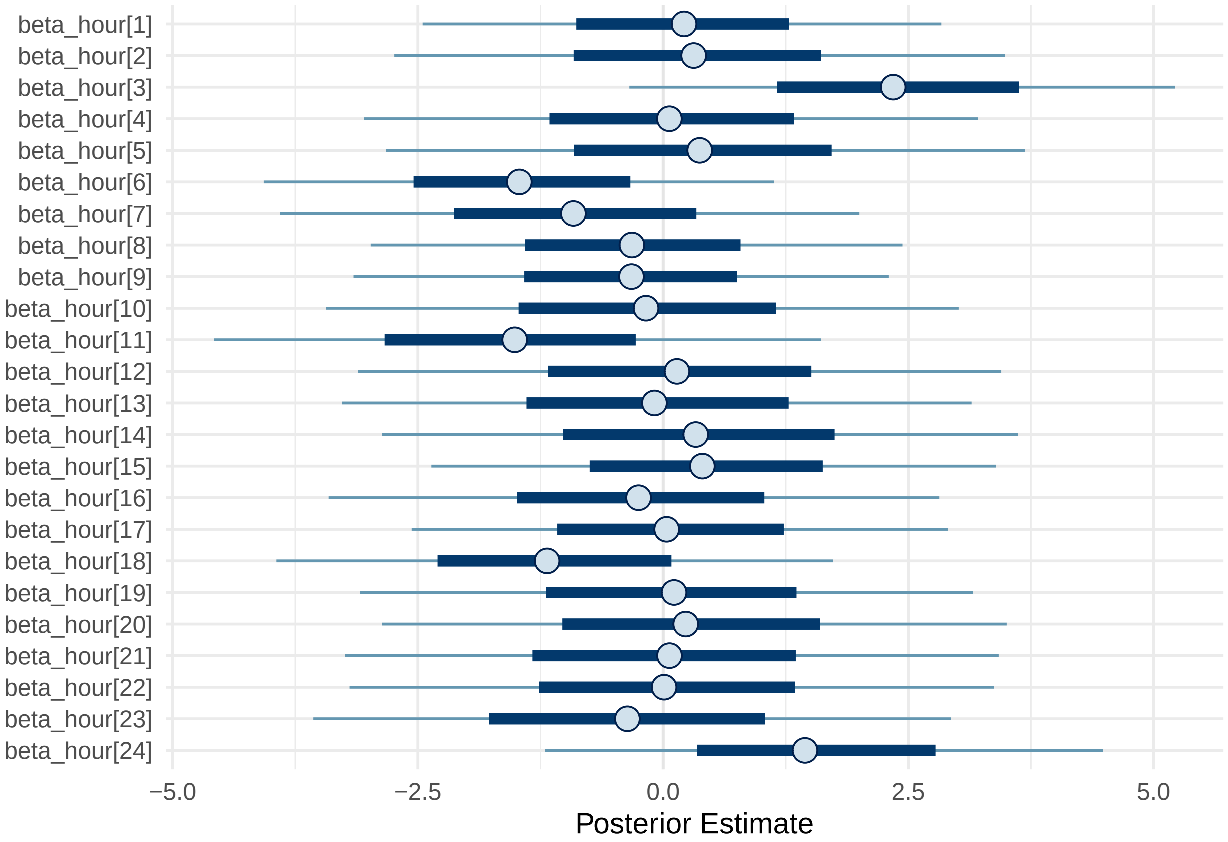
```
traceplot(fit_extended, pars = c("alpha", "beta_prev", "beta_dist", "sigma", "nu", "phi"))
```



```
dow_labels <- wday(1:7, label = TRUE, abbr = TRUE)
mcmc_intervals(as.array(fit_extended), regex_pars = "beta_dow") +
  scale_y_discrete(labels = as.character(dow_labels)) +
  labs(title = "Posterior Intervals for Day-of-Week Effects") +
  theme_minimal()
```



```r
mcmc_intervals(as.array(fit_extended), regex_pars = "beta_hour") +
  theme_minimal() +
  scale_x_continuous(name = "Posterior Estimate")
```


beta_hour[1]
beta_hour[2]
beta_hour[3]
beta_hour[4]
beta_hour[5]
beta_hour[6]
beta_hour[7]
beta_hour[8]
beta_hour[9]
beta_hour[10]
beta_hour[11]
beta_hour[12]
beta_hour[13]
beta_hour[14]
beta_hour[15]
beta_hour[16]
beta_hour[17]
beta_hour[18]
beta_hour[19]
beta_hour[20]
beta_hour[21]
beta_hour[22]
beta_hour[23]
beta_hour[24]
−5.0
−2.5
0.0
2.5
5.0
Posterior Estimate