Final Project

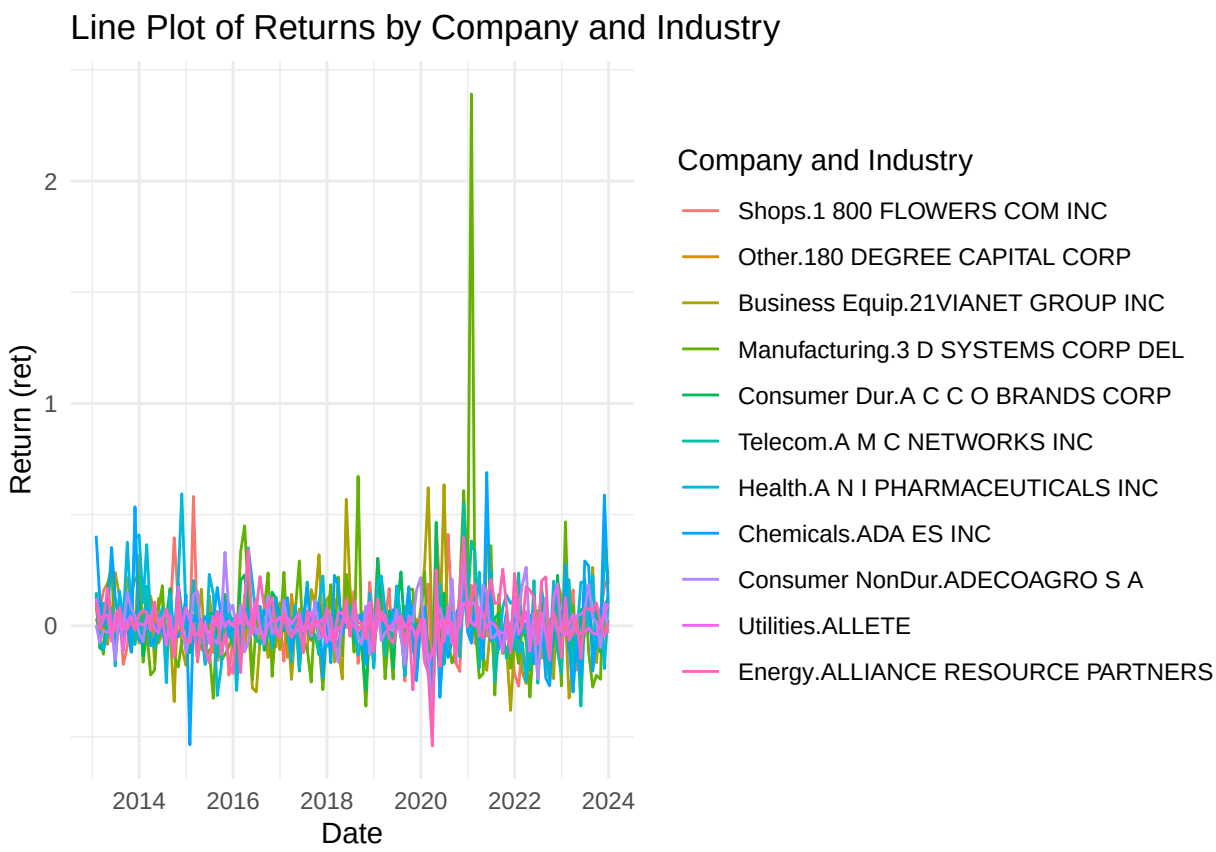
1 Load the data

```
# Read the csv file to load raw data
data <- read.csv("/Users/tracy/Desktop/UMich/STATS 551/Real_Final Project/data_sim.csv")
company <- data
macro <- read.csv("/Users/tracy/Desktop/UMich/STATS 551/Real_Final Project/macro_data.csv")

# Convert the 'date' column to Date type
company$date <- as.Date(company$date, format = "%Y/%m/%d")
macro$date <- as.Date(macro$date, format = "%Y/%m/%d")
```

2 Exploratory Data Analysis

2.1 Visualization of Response Variable

```
library(ggplot2)
ggplot(company, aes(x = date, y = ret, color = interaction(industry, comnam))) +
  geom_line() +
  labs(
    title = "Line Plot of Returns by Company and Industry",
    x = "Date",
    y = "Return (ret)",
    color = "Company and Industry"
  ) +
  theme_minimal() +
  theme(legend.position = "right")
```



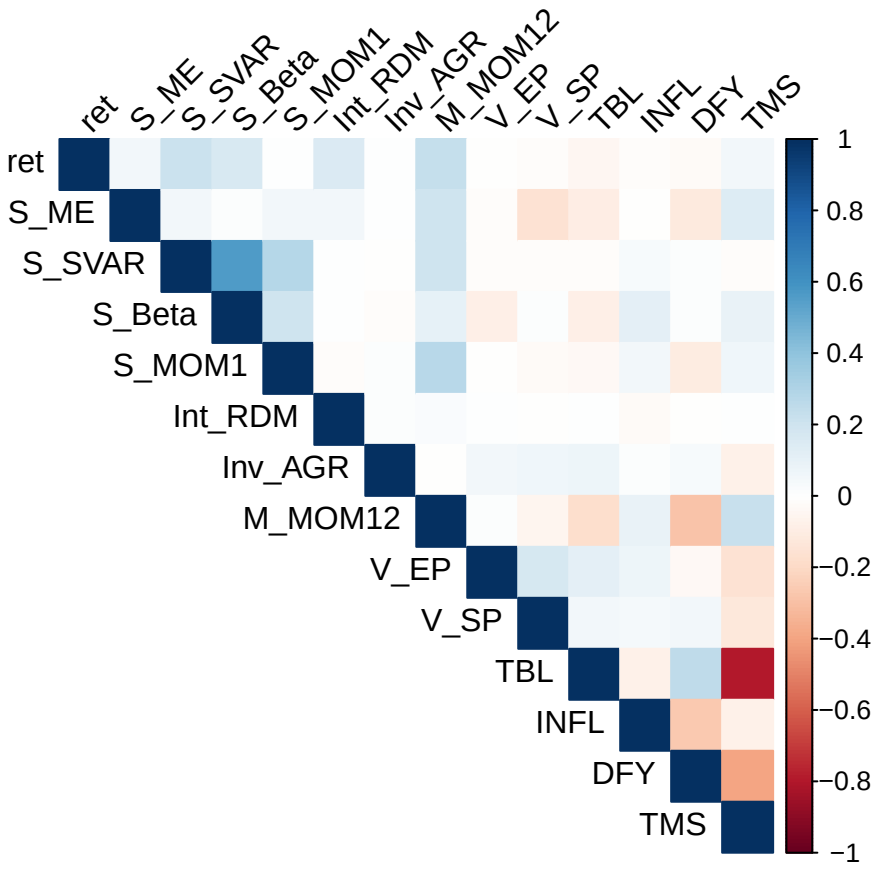
2.2 Multicollinearity Check

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3
```

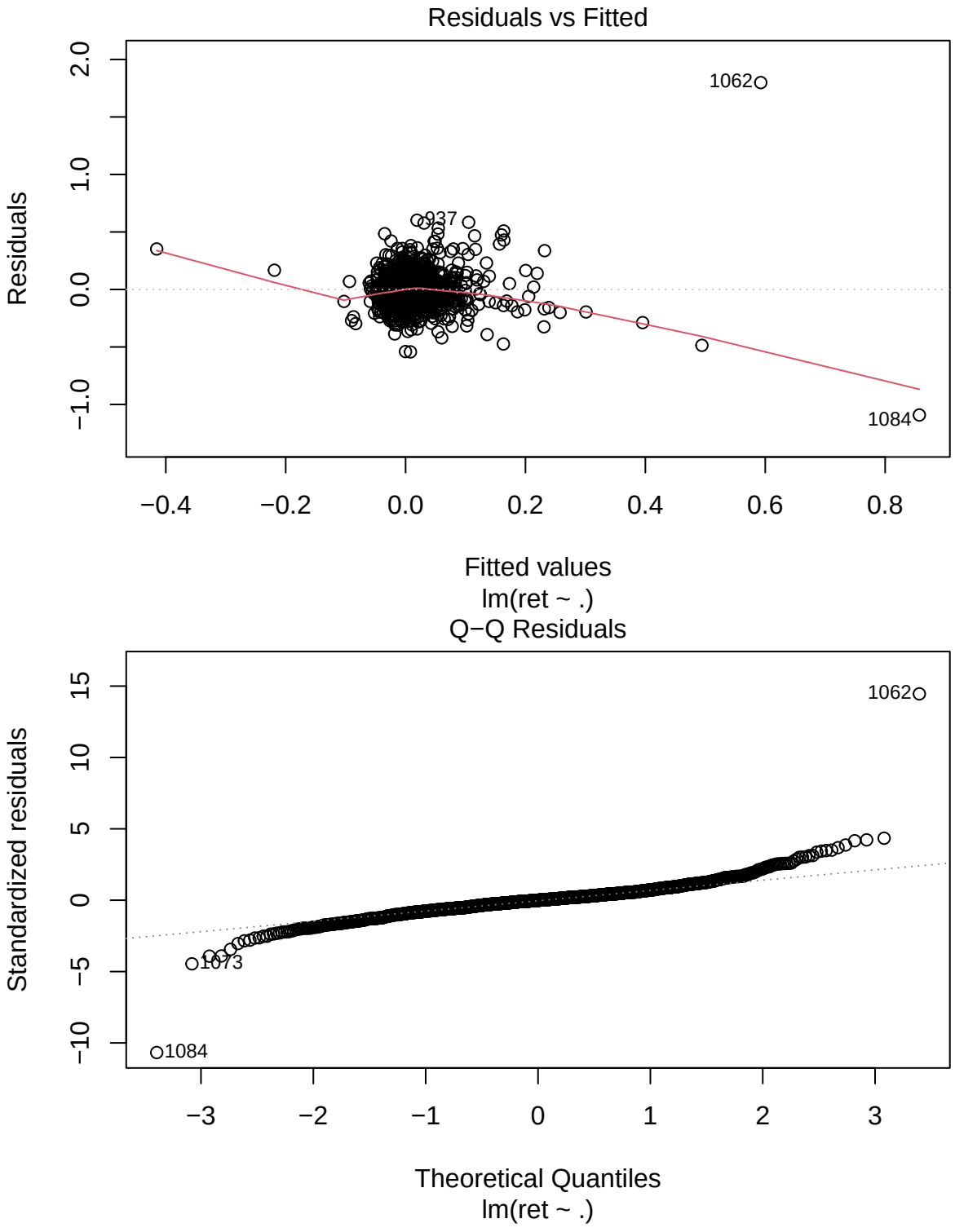
```
## corrplot 0.95 loaded
```

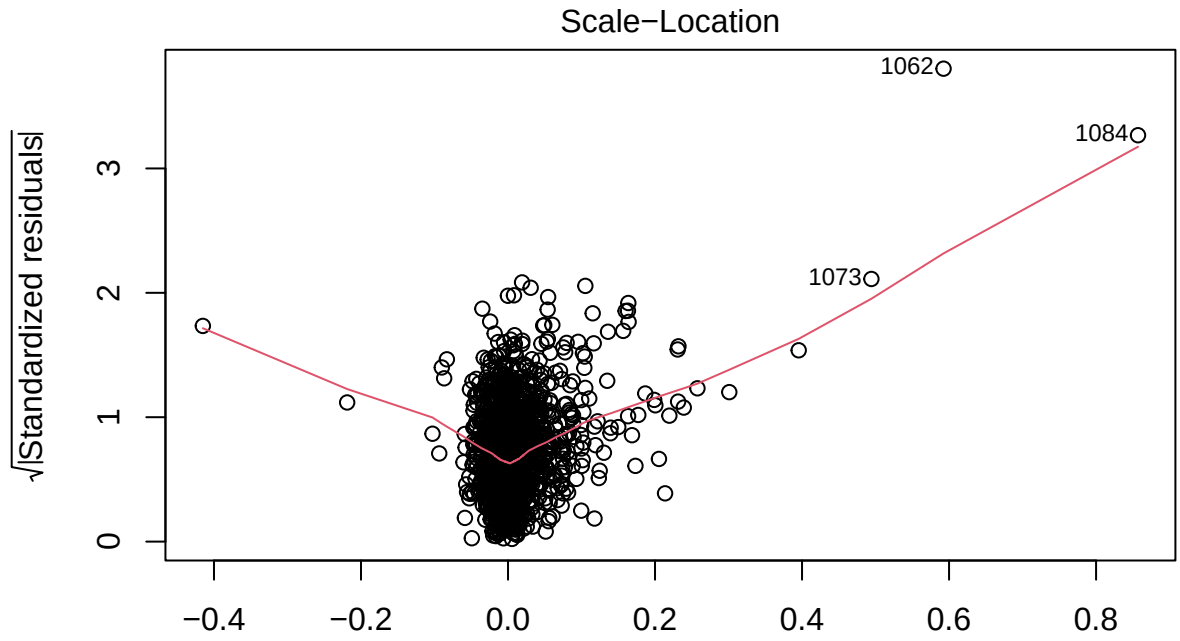
```
# merge macro and company data for correlation
macro$date <- as.Date(macro$date, format = "%Y/%m/%d")
merged_data <- merge(company, macro, by = "date", all.x = TRUE)

# visualize the correlation matrix
num_merge_data <- merged_data[, sapply(merged_data, is.numeric)]
cor_mat <- cor(num_merge_data)
corrplot(cor_mat, method = "color", type = "upper", tl.col = "black", tl.srt = 45)
```

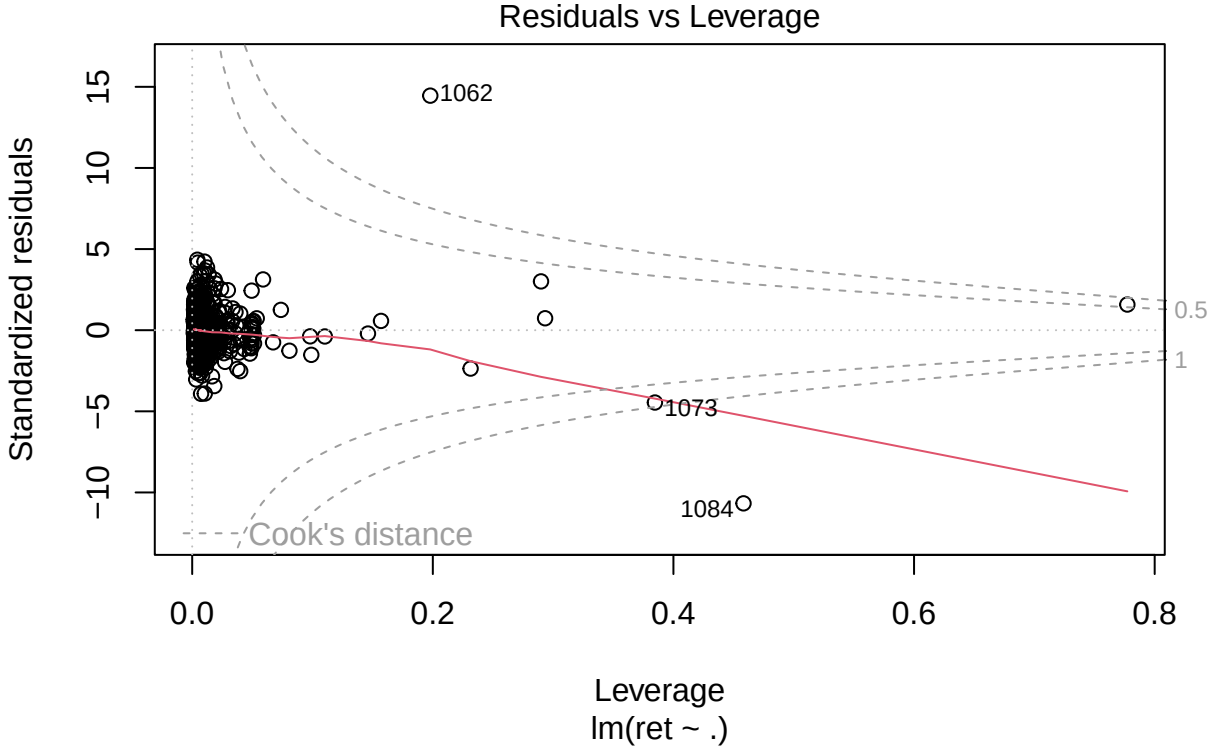


2.3 Linearity and Normality check

```
lm = lm(ret~ ., data = num_merge_data)
plot(lm)
```


Residuals vs Fitted
Residuals
Fitted values
lm(ret ~ .)
1062
937
1084
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(ret ~ .)
1062
1073
1084





3. Bayesian Hierarchical Regression Model

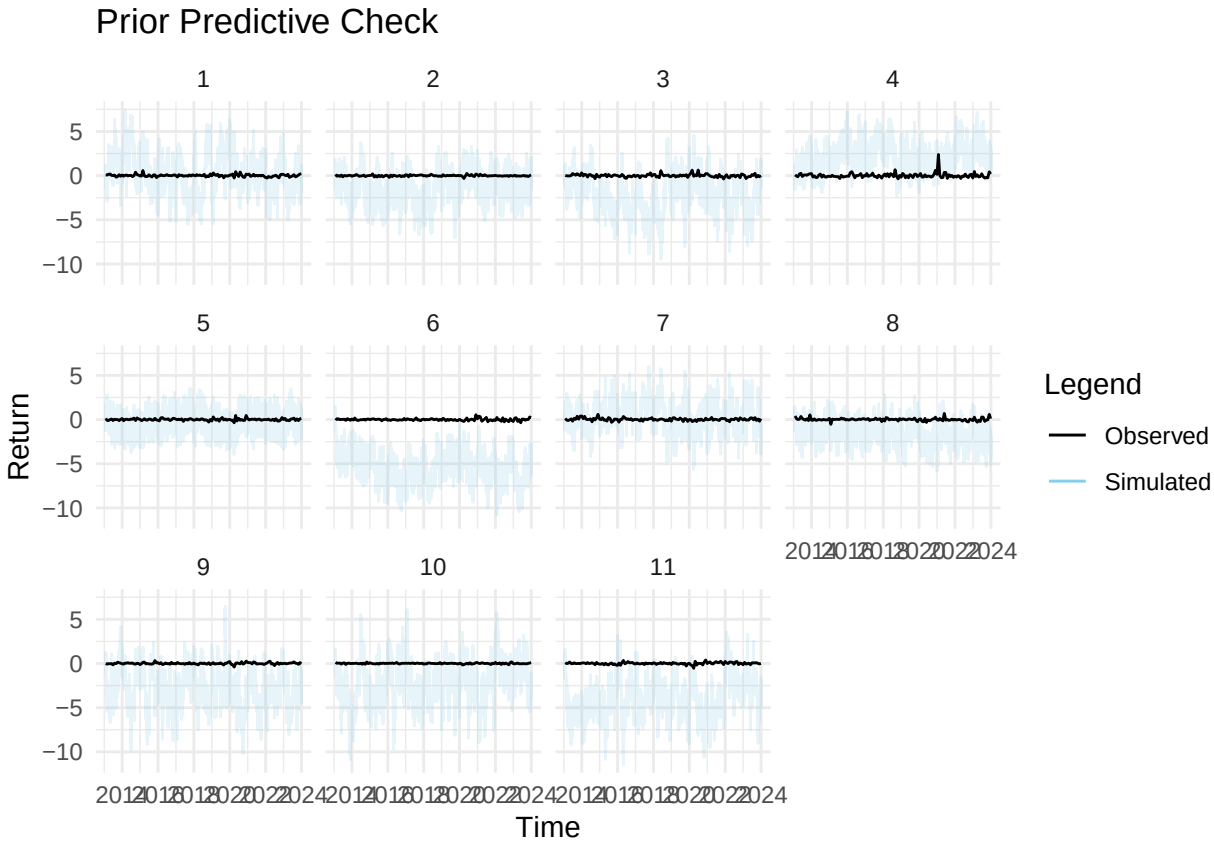
3.0 Prior Choosing

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)

N_prior <- length(unique(data$date))
M_prior <- length(unique(data$comnam))
J_factors <- 4
D <- 100

set.seed(123)

mu_prior <- rnorm(1, mean = 0, sd = 1)
phi_prior <- rnorm(M_prior, mean = 0.5, sd = 0.1)
eta_a_prior <- rnorm(M_prior, mean = 0, sd = 1)
eta_b_prior <- matrix(rnorm(M_prior * J_factors, mean = 0, sd = 1), nrow = M_prior, ncol = J_factors)
sigma_prior <- runif(1, min = 0.1, max = 1)

z_prior <- matrix(rnorm(N_prior * J_factors), nrow = N_prior, ncol = J_factors)

y_prior <- array(NA, dim = c(N_prior, M_prior, D))
for (d in 1:D) {
  y_prior[1, , d] <- rnorm(M_prior, mean = 0, sd = sigma_prior)
  for (n in 2:N_prior) {
    for (m in 1:M_prior) {
      alpha_prior <- eta_a_prior[m]
      beta_prior <- eta_b_prior[m, ]
      z_row <- z_prior[n, ]

      if (is.na(y_prior[n-1, m, d])) y_prior[n-1, m, d] <- 0

      y_prior[n, m, d] <- mu_prior +
                          alpha_prior + # Asset effects
                          phi_prior[m] * y_prior[n-1, m, d] +  # AR term
                          sum(z_row * beta_prior) +  # Macro effects
                          rnorm(1, mean = 0, sd = sigma_prior)  # Noise
    }
  }
}

y_prior_df <- as.data.frame.table(y_prior, responseName = "y_tilde") %>%
  rename(time = Var1, asset = Var2, draw = Var3) %>%
  mutate(
    time = rep(unique(data$date), times = M_prior * D),
    asset = as.numeric(asset),
    draw = as.numeric(draw)
  )
```

```r
prior_observed <- data[, c("date","ret", "comnam")]
prior_observed$comnam <- as.integer(factor(prior_observed$comnam))
names(prior_observed) <- c("time", "ret", "asset")

prior_combined <- left_join(y_prior_df, prior_observed, by = c("time", "asset"))
prior_combined$time <- as.Date(prior_combined$time, format = "%Y/%m/%d")

ggplot(prior_combined, aes(x = time)) +
  geom_line(aes(y = y_tilde, group = interaction(asset), color = "Simulated"), alpha = 0.2) +
  geom_line(aes(y = ret, group = asset, color = "Observed")) +
  facet_wrap(~ asset, ncol = 4) +
  labs(
    title = "Prior Predictive Check",
    x = "Time",
    y = "Return",
    color = "Legend"
  ) +
  scale_color_manual(values = c("Simulated" = "skyblue", "Observed" = "black")) +
  theme_minimal()
```



3.1 Full Model Fit

```r
library(rstan)
```

```
## Warning: package 'rstan' was built under R version 4.3.3
```

```
## Loading required package: StanHeaders
```

```
## Warning: package 'StanHeaders' was built under R version 4.3.3
```

```
##
## rstan version 2.35.0.9000 (Stan version 2.35.0)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
N <- length(unique(data$date))   # number of date period
M <- length(unique(data$comnam)) # number of assets
K <- 4                           # number of macro factors
L <- 9                           # number of company factors
x <- as.matrix(macro[, c("TBL", "INFL", "DFY", "TMS")])
y <- matrix(NA, N, M)
z_list <- list()

#Create asset company information matrix
for (i in 1:M) {
  z_asset <- data[data$comnam == unique(data$comnam)[i], c("S_ME", "S_SVAR", "S_MOM1", "S_Beta", "Int_R
  z_list[[i]] <- as.matrix(z_asset)
  y[, i] <- data$ret[data$comnam == unique(data$comnam)[i]]
}

z <- do.call(rbind, z_list)

stan_data <- list(
  N = N,
  M = M,
  K = K,
  L = L,
  x = x,
  z = z,
  y = y
)

fit <- stan(file = '/Users/tracy/Desktop/UMich/STATS 551/finance/code/hier_3.stan', data = stan_data, i
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 15.0.0 (clang-1500.3.9.4)'
## using SDK: 'MacOSX14.4.sdk'
## clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Framew
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/StanHeade
```

```
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/
## /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
## #include <cmath>
##          ^~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001302 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 13.02 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 598.748 seconds (Warm-up)
## Chain 1:                849.212 seconds (Sampling)
## Chain 1:                1447.96 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000865 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 8.65 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 610.296 seconds (Warm-up)
```

```
## Chain 2:                845.382 seconds (Sampling)
## Chain 2:                1455.68 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000834 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 8.34 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 576.166 seconds (Warm-up)
## Chain 3:                844.418 seconds (Sampling)
## Chain 3:                1420.58 seconds (Total)
## Chain 3:

## Warning: There were 3000 transitions after warmup that exceeded the maximum treedepth. Increase max_
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: The largest R-hat is 3.8, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

3.2 Expected Return Visualization before Model Selection

```
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
posterior <- extract(fit)
mu <- posterior$mu                # Global intercept
eta_a <- posterior$eta_a          # Random intercepts
eta_b <- posterior$eta_b          # Random coefficients for macro factors
theta_a <- posterior$theta_a      # Coefficients for company factors
theta_b <- posterior$theta_b      # Coefficients for macro factors
phi <- posterior$phi              # Autoregressive coefficients
sigma <- posterior$sigma          # Error scale

expected_returns <- array(NA, dim = c(length(mu), N, M))

# Compute expected return from hierarchical Bayesian model
for (iter in 1:length(mu)) {
  for (n in 2:N) {
    for (m in 1:M) {
      row <- (n - 1) * M + m
      z_row <- as.vector(stan_data$z[row, ])

      alpha_i_t <- eta_a[iter, m] + sum(z_row * theta_a[iter, m, ])
      beta_i_t <- eta_b[iter, m] + sum(z_row * theta_b[iter, m, ])

      expected_returns[iter, n, m] <- mu[iter] +
        alpha_i_t +
        phi[iter, m] * stan_data$y[n-1, m] +
        sum(stan_data$x[n, ] * beta_i_t)
    }
  }
}
expected_returns_mean <- apply(expected_returns, c(2, 3), mean)

y_obs <- stan_data$y
y_obs_df <- melt(y_obs)
colnames(y_obs_df) <- c("Time", "Asset", "Observed_Return")
y_obs_df$Observed_Return <- y_obs_df$Observed_Return * 100

y_hier_df <- melt(expected_returns_mean)
colnames(y_hier_df) <- c("Time", "Asset", "Expected_hier_Return")
y_hier_df$Expected_hier_Return <- y_hier_df$Expected_hier_Return * 100

comparison_data <- merge(y_obs_df, y_hier_df, by = c("Time", "Asset"))

ggplot(comparison_data, aes(x = Time)) +
  geom_line(aes(y = Observed_Return, color = "Observed")) +
  geom_line(aes(y = Expected_hier_Return, color = "Expected_hier")) +
  labs(title = paste("Observed vs Expected Returns before Selection"),
       x = "Time", y = "Return (%)") +
  theme_minimal() +
  scale_color_manual(values = c("Observed" = "black", "Expected_hier" = "red"))+
  facet_wrap(~ Asset, ncol = 4)
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```

3.3 Variable Selections

```r
# Initialize inclusion indicators
gamma_x <- rep(1, K)       # Macro predictors in x
gamma_z <- rep(1, L)       # Company predictors in z

# Initialize Gibbs sampler storage
iterations <- 2000
inclusion_x <- matrix(NA, nrow = iterations, ncol = K)
inclusion_z <- matrix(NA, nrow = iterations, ncol = L)

# Calculate posterior odds
for (iter in 1:iterations) {

  for (k in 1:K) {
    log_odds_x <- -0.5 * eta_b[iter, k]^2
    prob_x <- 1 / (1 + exp(-log_odds_x))
    gamma_x[k] <- rbinom(1, 1, prob_x)
  }

  for (l in 1:L) {
    log_odds_z <- -0.5 * sum(theta_a[iter, , l]^2)
    prob_z <- 1 / (1 + exp(-log_odds_z))
    gamma_z[l] <- rbinom(1, 1, prob_z)
  }
```

```
  inclusion_x[iter, ] <- gamma_x
  inclusion_z[iter, ] <- gamma_z
}

macro_names <- colnames(x)  # Macro predictors (x)
company_names <- colnames(z)  # Company predictors (z)

posterior_inclusion_x <- colMeans(inclusion_x)
posterior_inclusion_z <- colMeans(inclusion_z)

# Select predictors based on a threshold
selected_x <- which(posterior_inclusion_x > 0.4)
selected_z <- which(posterior_inclusion_z > 0.4)
cat("Selected Macro Predictors (x):", macro_names[selected_x], "\n")
```

```
## Selected Macro Predictors (x): TBL TMS
```

```
cat("Selected Company Predictors (z):", company_names[selected_z], "\n")
```

```
## Selected Company Predictors (z): S_ME Inv_AGR V_SP
```

3.4 Selected Model Fit

```
N_opt<- length(unique(data$date))   # number of date period
M_opt <- length(unique(data$comnam)) # number of assets
K_opt <- 4                           # numebr of macro factors
L_opt <- 3                           # numebr of company factors
x_opt <- as.matrix(macro[, c("TBL", "INFL", "TMS", "DFY")])
y_opt <- matrix(NA, N, M)
z_list_opt <- list()

for (i in 1:M_opt) {

  z_asset_opt <- data[data$comnam == unique(data$comnam)[i], c("S_ME", "Inv_AGR", "V_SP")]
  z_list_opt[[i]] <- as.matrix(z_asset_opt)
  y_opt[, i] <- data$ret[data$comnam == unique(data$comnam)[i]]
}

z_opt <- do.call(rbind, z_list_opt) #asset matrix


stan_data_opt <- list(
  N = N_opt,
  M = M_opt,
  K = K_opt,
  L = L_opt,
  x = x_opt,                  # Macro factors matrix (N x K)
  z = z_opt,                  # Asset characteristics matrix (N x L)
  y = y_opt                   # Asset returns matrix (N x M)
)

fit_opt <- stan(file = '/Users/tracy/Desktop/UMich/STATS 551/finance/code/hier_3.stan', data = stan_data
```

```
##
```

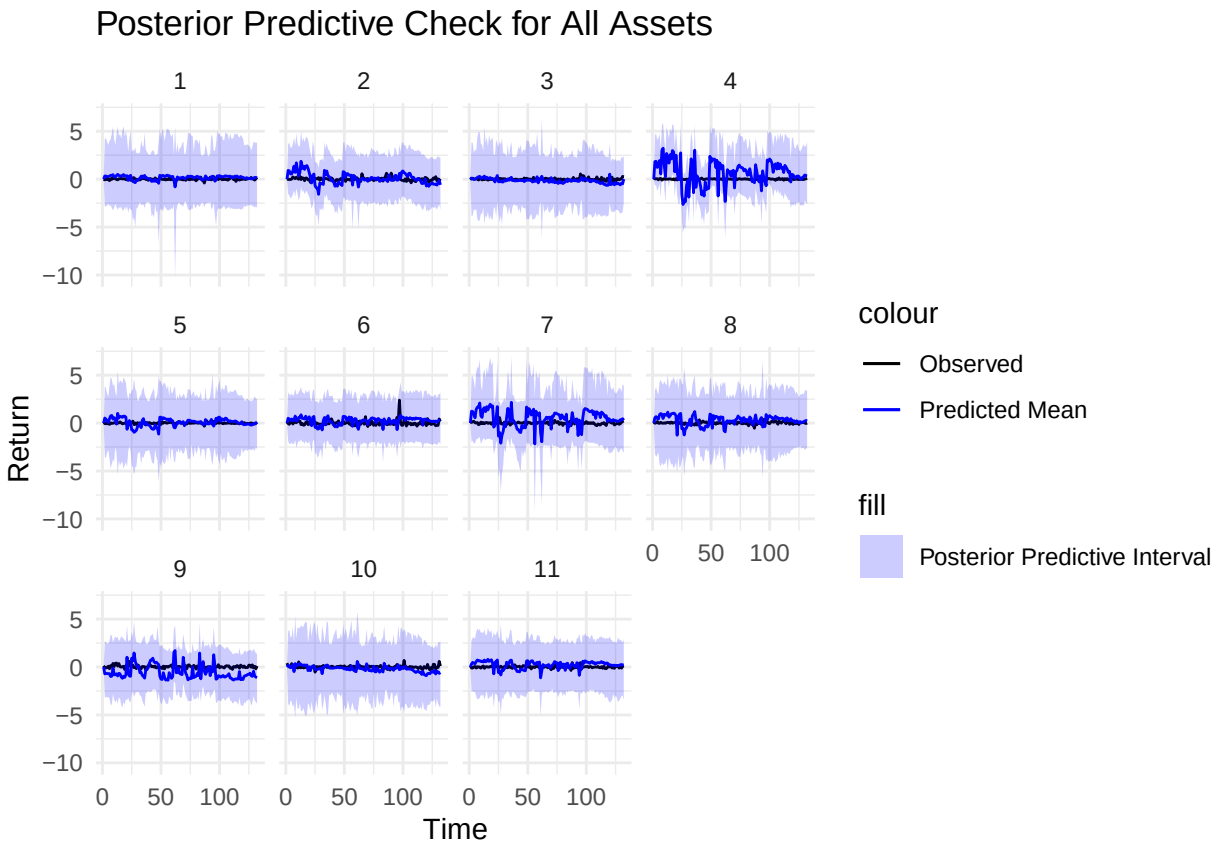
```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000726 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 7.26 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 395.5 seconds (Warm-up)
## Chain 1:                736.442 seconds (Sampling)
## Chain 1:                1131.94 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.00073 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 7.3 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 459.16 seconds (Warm-up)
## Chain 2:                733.291 seconds (Sampling)
## Chain 2:                1192.45 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000723 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 7.23 seconds.
```

```
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 341.028 seconds (Warm-up)
## Chain 3:                35.401 seconds (Sampling)
## Chain 3:                376.429 seconds (Total)
## Chain 3:

## Warning: There were 997 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: There were 2002 transitions after warmup that exceeded the maximum treedepth. Increase max_
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing Information was low. S
## https://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: The largest R-hat is 3.38, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

4 Posterior Predictive Check for Model Fit

```r
library(rstan)

# calculate the posterior prediction of hierarchical model
posterior_opt <- rstan::extract(fit_opt)
mu <- posterior_opt$mu                # Global intercept
eta_a <- posterior_opt$eta_a          # Random intercepts
eta_b <- posterior_opt$eta_b          # Random coefficients for macro factors
theta_a <- posterior_opt$theta_a      # Coefficients for intercepts (alpha)
theta_b <- posterior_opt$theta_b      # Coefficients for macro factors (beta)
phi <- posterior_opt$phi              # Autoregressive coefficients
```

```r
sigma <- posterior_opt$sigma          # Error scale

posterior_predictive <- array(NA, dim = c(length(mu), N_opt, M_opt))

for (iter in 1:length(mu)) {
  for (n in 2:N_opt) {
    for (m in 1:M_opt) {

      row <- (n - 1) * M_opt + m
      z_row <- as.vector(stan_data_opt$z[row, ])

      alpha_i_t <- eta_a[iter, m] + sum(z_row * theta_a[iter, m, ])
      beta_i_t <- eta_b[iter, m] + sum(z_row * theta_b[iter, m, ])

      y_pred_mean <- mu[iter] + alpha_i_t + phi[iter, m] * stan_data_opt$y[n-1, m] +
        sum(stan_data_opt$x[n, ] * beta_i_t)

      posterior_predictive[iter, n, m] <- rnorm(1, mean = y_pred_mean, sd = sigma[iter])
    }
  }
}

posterior_predictive[is.na(posterior_predictive)] <- 0
posterior_predictive_mean <- apply(posterior_predictive, c(2, 3), mean)
posterior_predictive_lower <- apply(posterior_predictive, c(2, 3), quantile, probs = 0.025)
posterior_predictive_upper <- apply(posterior_predictive, c(2, 3), quantile, probs = 0.975)

y_obs_opt <- stan_data_opt$y
y_obs_df_opt <- melt(y_obs_opt)
colnames(y_obs_df_opt) <- c("Time", "Asset", "Observed_Return")

posterior_plot <- data.frame(
  Time = rep(1:N_opt, M_opt),
  Asset = rep(1:M_opt, each = N_opt),
  Observed = as.vector(y_obs_opt),
  Predicted_Mean = as.vector(posterior_predictive_mean),
  Predicted_Lower = as.vector(posterior_predictive_lower),
  Predicted_Upper = as.vector(posterior_predictive_upper)
)


ggplot(posterior_plot, aes(x = Time)) +
  geom_line(aes(y = Observed, color = "Observed")) +
  geom_line(aes(y = Predicted_Mean, color = "Predicted Mean")) +
  geom_ribbon(aes(ymin = Predicted_Lower, ymax = Predicted_Upper, fill = "Posterior Predictive Interval
  labs(title = "Posterior Predictive Check for All Assets", x = "Time", y = "Return") +
  theme_minimal() +
  scale_color_manual(values = c("Observed" = "black", "Predicted Mean" = "blue")) +
  scale_fill_manual(values = c("Posterior Predictive Interval" = "blue")) +
  facet_wrap(~ Asset, ncol = 4)
```


Posterior Predictive Check for All Assets
1
2
3
4
5
6
7
8
9
10
11
Return
Time
colour
Observed
Predicted Mean
fill
Posterior Predictive Interval