HW8_echo

Echo (Wenyi) Xu

Research Question: How do activity categories influence caregivers' moment-to-moment emotions?

Variables - `Activity.Category`: activity that caregivers' were doing when completing the questionnaires: Productive, Social_Face, Social_Virtual, Screen_Time_Leisure, Exercise, Rest_Sleep, Baby_Care - `Composite`: Emotion Composite: Positve Affect, Negative Affect; value from 1 - 5 - `value_wc`: Centered Emotion within and between person

```
# Load the packages
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
```

```
library(tidyr)
library(nlme)
```

```
Attaching package: 'nlme'

The following object is masked from 'package:dplyr':

    collapse
```

```
library(ggplot2)
library(here)
```

```
here() starts at /Users/macbook/Documents/GitHub/PSYC573_Final_Project_Echo
```

```
library(brms)
```

```
Loading required package: Rcpp
```

```
Loading 'brms' package (version 2.22.0). Useful instructions
can be found by typing help('brms'). A more detailed introduction
to the package is available through vignette('brms_overview').


Attaching package: 'brms'

The following object is masked from 'package:stats':

    ar
```

```
library(readxl)  # for reading excel files
library(modelsummary)  # for summarizing data
```

```
`modelsummary` 2.0.0 now uses `tinytable` as its default table-drawing
  backend. Learn more at: https://vincentarelbundock.github.io/tinytable/

Revert to `kableExtra` for one session:

  options(modelsummary_factory_default = 'kableExtra')
  options(modelsummary_factory_latex = 'kableExtra')
  options(modelsummary_factory_html = 'kableExtra')

Silence this message forever:

  config_modelsummary(startup_message = FALSE)
```

```r
library(cmdstanr)  # use two cores
```

```
This is cmdstanr version 0.8.1
```

```
- CmdStanR documentation and vignettes: mc-stan.org/cmdstanr
```

```
- CmdStan path: /Users/macbook/.cmdstan/cmdstan-2.35.0
```

```
- CmdStan version: 2.35.0
```

```r
library(posterior)
```

```
This is posterior version 1.6.0
```

```

Attaching package: 'posterior'
```

```
The following objects are masked from 'package:stats':
```

```
    mad, sd, var
```

```
The following objects are masked from 'package:base':
```

```
    %in%, match
```

```r
library(bayesplot)
```

```
This is bayesplot version 1.11.1
```

```
- Online documentation and vignettes at mc-stan.org/bayesplot
```

```
- bayesplot theme set to bayesplot::theme_default()
```

```
   * Does _not_ affect other ggplot2 plots
```

```
   * See ?bayesplot_theme_set for details on theme setting


Attaching package: 'bayesplot'

The following object is masked from 'package:posterior':

    rhat

The following object is masked from 'package:brms':

    rhat
```

```r
library(knitr)

## Data Import
load("/Volumes/data/Studies/STAR/Data/EMA/Saved Environments/data_11_20_2024.RData")
dfm$Doing <- tolower(dfm$Doing)
```

```r
# Data Cleaning
dfm$Doing <- gsub("e.g.,", "e.g.", dfm$Doing)

# Define groups
productive <- c("work or other productive activities", "chores/errands",
                "personal maintenance",
                "transit")
social_face <- c("socializing face-to-face")
social_virtual <- c("socializing virtually")
screen_time_leisure <- c("movies", "using social media")
exercise <- c("exercising/playing a sport")
rest_sleep <- c("sleeping/napping", "doing absolutely nothing")
baby_care <- c("feeding the baby", "changing the baby")

# Classify activities into categories, accounting for multiple selections, modified includ
dfm <- dfm %>%
  mutate(
    Productive = ifelse(grepl(paste(productive, collapse = "|"), Doing, ignore.case = TRUE
    Social_Face = ifelse(grepl(paste(social_face, collapse = "|"), Doing, ignore.case = TR
    Social_Virtual = ifelse(grepl(paste(social_virtual, collapse = "|"), Doing, ignore.cas
    Screen_Time_Leisure = ifelse(grepl(paste(screen_time_leisure, collapse = "|"), Doing,
    Exercise = ifelse(grepl(paste(exercise, collapse = "|"), Doing, ignore.case = TRUE), 1
```

```
    Rest_Sleep = ifelse(grepl(paste(rest_sleep, collapse = "|"), Doing, ignore.case = TRUE
    Baby_Care = ifelse(grepl(paste(baby_care, collapse = "|"), Doing, ignore.case = TRUE),
  ) %>%
  rowwise() %>%
  mutate(
    # Combine all categories into a single string, separated by commas
    Activity.Category = paste(
      na.omit(unique(c(
        ifelse(Productive == 1, "Productive Activities", NA),
        ifelse(Social_Face == 1, "Socializing Face-to-Face", NA),
        ifelse(Social_Virtual == 1, "Socializing Virtually", NA),
        ifelse(Screen_Time_Leisure == 1, "Screen Time (Leisure)", NA),
        ifelse(Exercise == 1, "Exercise", NA),
        ifelse(Rest_Sleep == 1, "Rest and Sleep", NA),
        ifelse(Baby_Care == 1, "Baby Care", NA)
      ))),
      collapse = ", "
    ),

    # If more than one category is identified, set as "Other"
    # Activity.Category = ifelse(grepl(",", Activity.Category), "Other", Activity.Category

    # Label as "Other" if no category is identified but `Doing` is not NA
    Activity.Category = ifelse(Activity.Category == "" & !is.na(Doing), "Other", Activity.
  ) %>%
  ungroup()

# Creating a model that aggregates across composites for one per entry
dfm.ag <- dfm %>%
  dplyr::group_by(record_id, Composite, Activity.Category, DAY, SIG, Duration..in.seconds.
  dplyr::summarise(across(where(is.numeric), ~mean(.x, na.rm = TRUE)))%>%  # This is model
  ungroup()
```

```
`summarise()` has grouped output by 'record_id', 'Composite',
'Activity.Category', 'DAY', 'SIG'. You can override using the `.groups`
argument.
```

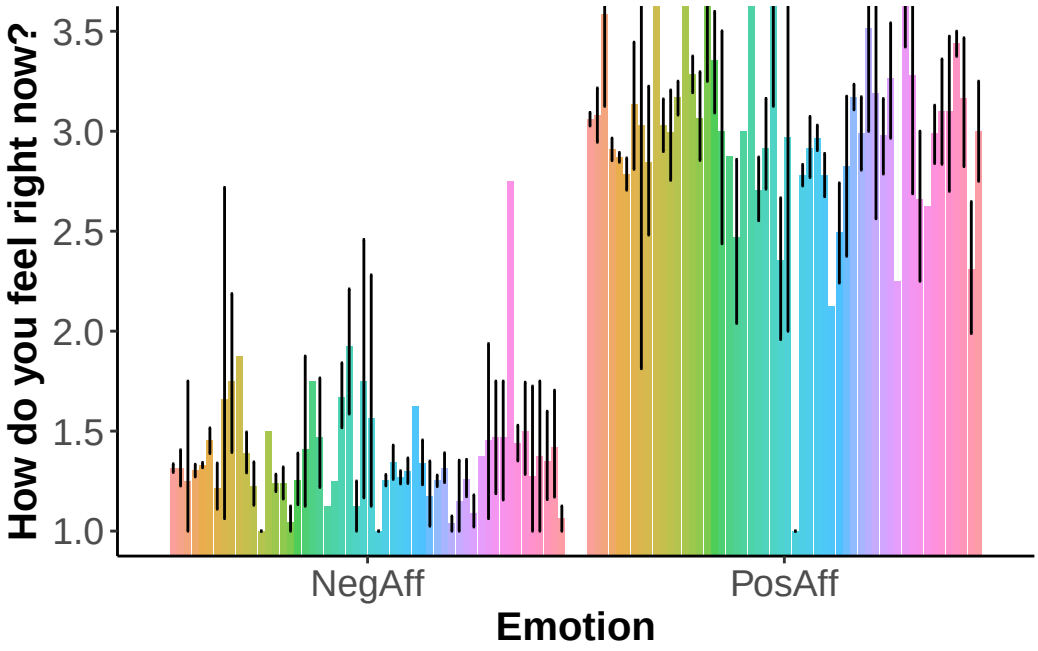
```
#####
# Centering within and between person for each emotion
#####
dfm.ag <- dfm.ag %>% group_by(record_id, Composite) %>%
```

```
  dplyr::mutate(w_mean = mean(value, na.rm = T)) %>% ungroup() %>%
  dplyr::mutate(value_bc = scale(value, center = T, scale = F),
                value_wc = value - w_mean)

dfm.ag <- dfm.ag %>%
  mutate(
    Composite = as.factor(Composite),
    Activity.Category = as.factor(Activity.Category),
    value = as.numeric(value)
  )
dfm.ag <- dfm.ag %>% drop_na(value_wc)
dfm.pos <- dfm.ag %>% filter(Composite == "PosAff")
dfm.neg <- dfm.ag %>% filter(Composite == "NegAff")
dfm.oth <- dfm.ag %>% filter(Composite == "Other")
```

```
## Variable Summary
## plotting
ggplot(
  dfm.ag[!is.na(dfm.ag$Activity.Category) & dfm.ag$Activity.Category != "" &
         dfm.ag$Composite != "Other" & dfm.ag$Activity.Category != "Mix",],
  aes(x = Composite, y = value, fill = Activity.Category)
) +
  my_opts +
  stat_summary(fun.y = mean, geom = "bar", position = position_dodge(width = 0.95), alpha
  stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.3, position = positio
  labs(
    x = "Emotion",
    y = "How do you feel right now?"
  ) +
  my_opts +
  coord_cartesian(ylim = c(1, 3.5)) +
  theme(legend.position = "none")
```

```
Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
i Please use the `fun` argument instead.
```



Model

Let y = value_wc, g = Activity.Category i represents participant j represents number of activity catergory t means observation time points

Model:

$$
\begin{aligned}
y_{ti} &\sim N(\mu_{ti}, \sigma_1) \\
\mu_{ti} &= \sum_j \beta_{1ij} * g_{ij} + \ \beta_{0i} \\
\beta_{0i} &\sim N(\mu_{\beta 0}, \sigma_{\beta 0}) \\
\beta_{1ij} &\sim N(\mu_{\beta_{1ij}}, \sigma_{\beta_{1ij}})
\end{aligned}
$$

y_i and g_{ij} are observables.

Prior:

$$
\begin{aligned}
\sigma_1 &\sim N(0,1) \\
\sigma_{\beta 0} &\sim N(0,1) \\
\mu_{\beta 0} &\sim N(0,1) \\
\mu_{\beta_{1ij}} &\sim N(0,1) \\
\sigma_{\beta_{1ij}} &\sim N(0,1)
\end{aligned}
$$

```r
## Model
#####
# Centering within and between person for each emotion
#####
dfm.ag <- dfm.ag %>% group_by(record_id, Composite) %>%
  dplyr::mutate(w_mean = mean(value, na.rm = T)) %>% ungroup() %>%
  dplyr::mutate(value_bc = scale(value, center = T, scale = T)[,1],
                value_wc = value - w_mean)

dfm.ag <- dfm.ag %>%
  mutate(
    Composite = as.factor(Composite),
    Activity.Category = as.factor(Activity.Category),
    value = as.numeric(value)
  )

dfm.ag <- dfm.ag %>% drop_na(value_bc)
dfm.pos <- dfm.ag %>% filter(Composite == "PosAff")
dfm.neg <- dfm.ag %>% filter(Composite == "NegAff")
dfm.oth <- dfm.ag %>% filter(Composite == "Other")

df = dfm.pos

train_size <- floor(0.9 * nrow(df))
train_ind <- sample(seq_len(nrow(df)), size = train_size)
train_df <- df[train_ind, ]
test_df <- df[-train_ind, ]

# Create the matrix `g` from binary indicator columns
# Replace the column names below with the actual names of your binary columns
binary_columns <- c("Productive", "Social_Face", "Social_Virtual",
                    "Screen_Time_Leisure", "Exercise", "Rest_Sleep", "Baby_Care")

# observations
g <- as.matrix(train_df[binary_columns])
y <- train_df$value_bc
test_g <- as.matrix(test_df[binary_columns])
test_y <- train_df$value_bc
# Number of activities (columns in g)
J <- ncol(g)
# Number of observations (rows in g)
```

```r
N <- nrow(g)

pid <- factor(train_df$record_id)
P <- length(unique(pid))

# Prepare the data list for Stan
stan_data <- list(
  J = J,      # Number of clusters (activities)
  N = N,      # Number of observations
  y = y,      # Response variable
  g = g,      # Transpose g to match Stan's `J x N` format
  P = P,      # number of participants
  pid = pid   # index of participant
)
```

```r
## Results
# Load and compile the Stan model
model <- cmdstan_model("model.stan")
n_chains <- 4

# Fit the model using sampling
fit <- model$sample(
  data = stan_data,
  chains = n_chains,                 # Number of chains
  parallel_chains = n_chains,        # Run chains in parallel
  iter_sampling = 4000,       # Number of iterations
  iter_warmup = 1000,         # Number of warmup iterations
  thin = 5                    # thinning
)
```

```
Running MCMC with 4 parallel chains...

Chain 1 Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 2 Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3 Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4 Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3 Iteration:  100 / 5000 [  2%]  (Warmup)
Chain 1 Iteration:  100 / 5000 [  2%]  (Warmup)
Chain 4 Iteration:  100 / 5000 [  2%]  (Warmup)
Chain 2 Iteration:  100 / 5000 [  2%]  (Warmup)
Chain 3 Iteration:  200 / 5000 [  4%]  (Warmup)
```

```
Chain 1 Iteration:  200 / 5000 [  4%]  (Warmup)
Chain 4 Iteration:  200 / 5000 [  4%]  (Warmup)
Chain 2 Iteration:  200 / 5000 [  4%]  (Warmup)
Chain 1 Iteration:  300 / 5000 [  6%]  (Warmup)
Chain 4 Iteration:  300 / 5000 [  6%]  (Warmup)
Chain 3 Iteration:  300 / 5000 [  6%]  (Warmup)
Chain 2 Iteration:  300 / 5000 [  6%]  (Warmup)
Chain 4 Iteration:  400 / 5000 [  8%]  (Warmup)
Chain 1 Iteration:  400 / 5000 [  8%]  (Warmup)
Chain 3 Iteration:  400 / 5000 [  8%]  (Warmup)
Chain 2 Iteration:  400 / 5000 [  8%]  (Warmup)
Chain 1 Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4 Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2 Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3 Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1 Iteration:  600 / 5000 [ 12%]  (Warmup)
Chain 2 Iteration:  600 / 5000 [ 12%]  (Warmup)
Chain 3 Iteration:  600 / 5000 [ 12%]  (Warmup)
Chain 4 Iteration:  600 / 5000 [ 12%]  (Warmup)
Chain 1 Iteration:  700 / 5000 [ 14%]  (Warmup)
Chain 2 Iteration:  700 / 5000 [ 14%]  (Warmup)
Chain 3 Iteration:  700 / 5000 [ 14%]  (Warmup)
Chain 4 Iteration:  700 / 5000 [ 14%]  (Warmup)
Chain 1 Iteration:  800 / 5000 [ 16%]  (Warmup)
Chain 4 Iteration:  800 / 5000 [ 16%]  (Warmup)
Chain 3 Iteration:  800 / 5000 [ 16%]  (Warmup)
Chain 2 Iteration:  800 / 5000 [ 16%]  (Warmup)
Chain 1 Iteration:  900 / 5000 [ 18%]  (Warmup)
Chain 4 Iteration:  900 / 5000 [ 18%]  (Warmup)
Chain 3 Iteration:  900 / 5000 [ 18%]  (Warmup)
Chain 2 Iteration:  900 / 5000 [ 18%]  (Warmup)
Chain 1 Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1 Iteration: 1001 / 5000 [ 20%]  (Sampling)
Chain 3 Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3 Iteration: 1001 / 5000 [ 20%]  (Sampling)
Chain 4 Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4 Iteration: 1001 / 5000 [ 20%]  (Sampling)
Chain 1 Iteration: 1100 / 5000 [ 22%]  (Sampling)
Chain 2 Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2 Iteration: 1001 / 5000 [ 20%]  (Sampling)
Chain 3 Iteration: 1100 / 5000 [ 22%]  (Sampling)
Chain 4 Iteration: 1100 / 5000 [ 22%]  (Sampling)
Chain 1 Iteration: 1200 / 5000 [ 24%]  (Sampling)
```

```
Chain 3 Iteration: 1200 / 5000 [ 24%]  (Sampling)
Chain 4 Iteration: 1200 / 5000 [ 24%]  (Sampling)
Chain 1 Iteration: 1300 / 5000 [ 26%]  (Sampling)
Chain 3 Iteration: 1300 / 5000 [ 26%]  (Sampling)
Chain 4 Iteration: 1300 / 5000 [ 26%]  (Sampling)
Chain 1 Iteration: 1400 / 5000 [ 28%]  (Sampling)
Chain 3 Iteration: 1400 / 5000 [ 28%]  (Sampling)
Chain 4 Iteration: 1400 / 5000 [ 28%]  (Sampling)
Chain 1 Iteration: 1500 / 5000 [ 30%]  (Sampling)
Chain 3 Iteration: 1500 / 5000 [ 30%]  (Sampling)
Chain 4 Iteration: 1500 / 5000 [ 30%]  (Sampling)
Chain 1 Iteration: 1600 / 5000 [ 32%]  (Sampling)
Chain 3 Iteration: 1600 / 5000 [ 32%]  (Sampling)
Chain 4 Iteration: 1600 / 5000 [ 32%]  (Sampling)
Chain 1 Iteration: 1700 / 5000 [ 34%]  (Sampling)
Chain 3 Iteration: 1700 / 5000 [ 34%]  (Sampling)
Chain 4 Iteration: 1700 / 5000 [ 34%]  (Sampling)
Chain 1 Iteration: 1800 / 5000 [ 36%]  (Sampling)
Chain 3 Iteration: 1800 / 5000 [ 36%]  (Sampling)
Chain 2 Iteration: 1100 / 5000 [ 22%]  (Sampling)
Chain 4 Iteration: 1800 / 5000 [ 36%]  (Sampling)
Chain 1 Iteration: 1900 / 5000 [ 38%]  (Sampling)
Chain 3 Iteration: 1900 / 5000 [ 38%]  (Sampling)
Chain 4 Iteration: 1900 / 5000 [ 38%]  (Sampling)
Chain 1 Iteration: 2000 / 5000 [ 40%]  (Sampling)
Chain 3 Iteration: 2000 / 5000 [ 40%]  (Sampling)
Chain 4 Iteration: 2000 / 5000 [ 40%]  (Sampling)
Chain 1 Iteration: 2100 / 5000 [ 42%]  (Sampling)
Chain 3 Iteration: 2100 / 5000 [ 42%]  (Sampling)
Chain 4 Iteration: 2100 / 5000 [ 42%]  (Sampling)
Chain 1 Iteration: 2200 / 5000 [ 44%]  (Sampling)
Chain 3 Iteration: 2200 / 5000 [ 44%]  (Sampling)
Chain 4 Iteration: 2200 / 5000 [ 44%]  (Sampling)
Chain 1 Iteration: 2300 / 5000 [ 46%]  (Sampling)
Chain 3 Iteration: 2300 / 5000 [ 46%]  (Sampling)
Chain 4 Iteration: 2300 / 5000 [ 46%]  (Sampling)
Chain 1 Iteration: 2400 / 5000 [ 48%]  (Sampling)
Chain 3 Iteration: 2400 / 5000 [ 48%]  (Sampling)
Chain 4 Iteration: 2400 / 5000 [ 48%]  (Sampling)
Chain 1 Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 3 Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 4 Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 1 Iteration: 2600 / 5000 [ 52%]  (Sampling)
```

```
Chain 2 Iteration: 1200 / 5000 [ 24%]  (Sampling)
Chain 3 Iteration: 2600 / 5000 [ 52%]  (Sampling)
Chain 4 Iteration: 2600 / 5000 [ 52%]  (Sampling)
Chain 1 Iteration: 2700 / 5000 [ 54%]  (Sampling)
Chain 3 Iteration: 2700 / 5000 [ 54%]  (Sampling)
Chain 4 Iteration: 2700 / 5000 [ 54%]  (Sampling)
Chain 1 Iteration: 2800 / 5000 [ 56%]  (Sampling)
Chain 3 Iteration: 2800 / 5000 [ 56%]  (Sampling)
Chain 4 Iteration: 2800 / 5000 [ 56%]  (Sampling)
Chain 1 Iteration: 2900 / 5000 [ 58%]  (Sampling)
Chain 3 Iteration: 2900 / 5000 [ 58%]  (Sampling)
Chain 4 Iteration: 2900 / 5000 [ 58%]  (Sampling)
Chain 1 Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3 Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4 Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1 Iteration: 3100 / 5000 [ 62%]  (Sampling)
Chain 3 Iteration: 3100 / 5000 [ 62%]  (Sampling)
Chain 4 Iteration: 3100 / 5000 [ 62%]  (Sampling)
Chain 1 Iteration: 3200 / 5000 [ 64%]  (Sampling)
Chain 3 Iteration: 3200 / 5000 [ 64%]  (Sampling)
Chain 2 Iteration: 1300 / 5000 [ 26%]  (Sampling)
Chain 4 Iteration: 3200 / 5000 [ 64%]  (Sampling)
Chain 1 Iteration: 3300 / 5000 [ 66%]  (Sampling)
Chain 3 Iteration: 3300 / 5000 [ 66%]  (Sampling)
Chain 4 Iteration: 3300 / 5000 [ 66%]  (Sampling)
Chain 1 Iteration: 3400 / 5000 [ 68%]  (Sampling)
Chain 3 Iteration: 3400 / 5000 [ 68%]  (Sampling)
Chain 4 Iteration: 3400 / 5000 [ 68%]  (Sampling)
Chain 1 Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3 Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4 Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1 Iteration: 3600 / 5000 [ 72%]  (Sampling)
Chain 3 Iteration: 3600 / 5000 [ 72%]  (Sampling)
Chain 4 Iteration: 3600 / 5000 [ 72%]  (Sampling)
Chain 1 Iteration: 3700 / 5000 [ 74%]  (Sampling)
Chain 3 Iteration: 3700 / 5000 [ 74%]  (Sampling)
Chain 4 Iteration: 3700 / 5000 [ 74%]  (Sampling)
Chain 1 Iteration: 3800 / 5000 [ 76%]  (Sampling)
Chain 2 Iteration: 1400 / 5000 [ 28%]  (Sampling)
Chain 3 Iteration: 3800 / 5000 [ 76%]  (Sampling)
Chain 4 Iteration: 3800 / 5000 [ 76%]  (Sampling)
Chain 1 Iteration: 3900 / 5000 [ 78%]  (Sampling)
Chain 3 Iteration: 3900 / 5000 [ 78%]  (Sampling)
```

```
Chain 4 Iteration: 3900 / 5000 [ 78%]  (Sampling)
Chain 1 Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3 Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4 Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1 Iteration: 4100 / 5000 [ 82%]  (Sampling)
Chain 3 Iteration: 4100 / 5000 [ 82%]  (Sampling)
Chain 4 Iteration: 4100 / 5000 [ 82%]  (Sampling)
Chain 1 Iteration: 4200 / 5000 [ 84%]  (Sampling)
Chain 3 Iteration: 4200 / 5000 [ 84%]  (Sampling)
Chain 4 Iteration: 4200 / 5000 [ 84%]  (Sampling)
Chain 1 Iteration: 4300 / 5000 [ 86%]  (Sampling)
Chain 3 Iteration: 4300 / 5000 [ 86%]  (Sampling)
Chain 4 Iteration: 4300 / 5000 [ 86%]  (Sampling)
Chain 1 Iteration: 4400 / 5000 [ 88%]  (Sampling)
Chain 3 Iteration: 4400 / 5000 [ 88%]  (Sampling)
Chain 4 Iteration: 4400 / 5000 [ 88%]  (Sampling)
Chain 2 Iteration: 1500 / 5000 [ 30%]  (Sampling)
Chain 1 Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3 Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4 Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1 Iteration: 4600 / 5000 [ 92%]  (Sampling)
Chain 3 Iteration: 4600 / 5000 [ 92%]  (Sampling)
Chain 4 Iteration: 4600 / 5000 [ 92%]  (Sampling)
Chain 1 Iteration: 4700 / 5000 [ 94%]  (Sampling)
Chain 3 Iteration: 4700 / 5000 [ 94%]  (Sampling)
Chain 4 Iteration: 4700 / 5000 [ 94%]  (Sampling)
Chain 1 Iteration: 4800 / 5000 [ 96%]  (Sampling)
Chain 3 Iteration: 4800 / 5000 [ 96%]  (Sampling)
Chain 4 Iteration: 4800 / 5000 [ 96%]  (Sampling)
Chain 1 Iteration: 4900 / 5000 [ 98%]  (Sampling)
Chain 3 Iteration: 4900 / 5000 [ 98%]  (Sampling)
Chain 4 Iteration: 4900 / 5000 [ 98%]  (Sampling)
Chain 1 Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1 finished in 536.0 seconds.
Chain 3 Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3 finished in 541.1 seconds.
Chain 4 Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4 finished in 542.6 seconds.
Chain 2 Iteration: 1600 / 5000 [ 32%]  (Sampling)
Chain 2 Iteration: 1700 / 5000 [ 34%]  (Sampling)
Chain 2 Iteration: 1800 / 5000 [ 36%]  (Sampling)
Chain 2 Iteration: 1900 / 5000 [ 38%]  (Sampling)
Chain 2 Iteration: 2000 / 5000 [ 40%]  (Sampling)
```

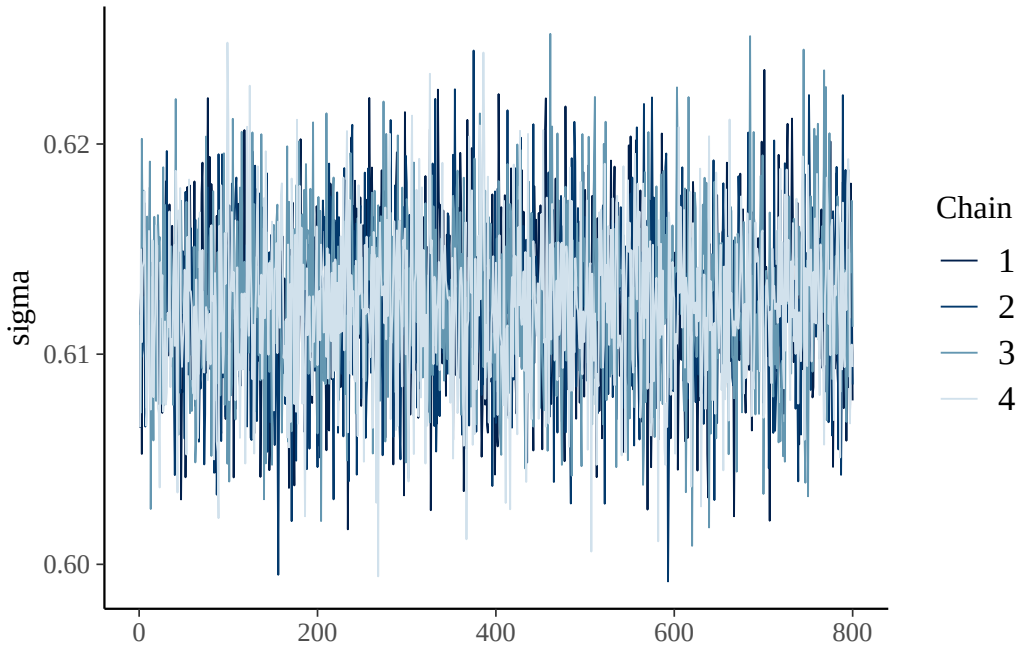
```
Chain 2 Iteration: 2100 / 5000 [ 42%]  (Sampling)
Chain 2 Iteration: 2200 / 5000 [ 44%]  (Sampling)
Chain 2 Iteration: 2300 / 5000 [ 46%]  (Sampling)
Chain 2 Iteration: 2400 / 5000 [ 48%]  (Sampling)
Chain 2 Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 2 Iteration: 2600 / 5000 [ 52%]  (Sampling)
Chain 2 Iteration: 2700 / 5000 [ 54%]  (Sampling)
Chain 2 Iteration: 2800 / 5000 [ 56%]  (Sampling)
Chain 2 Iteration: 2900 / 5000 [ 58%]  (Sampling)
Chain 2 Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2 Iteration: 3100 / 5000 [ 62%]  (Sampling)
Chain 2 Iteration: 3200 / 5000 [ 64%]  (Sampling)
Chain 2 Iteration: 3300 / 5000 [ 66%]  (Sampling)
Chain 2 Iteration: 3400 / 5000 [ 68%]  (Sampling)
Chain 2 Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2 Iteration: 3600 / 5000 [ 72%]  (Sampling)
Chain 2 Iteration: 3700 / 5000 [ 74%]  (Sampling)
Chain 2 Iteration: 3800 / 5000 [ 76%]  (Sampling)
Chain 2 Iteration: 3900 / 5000 [ 78%]  (Sampling)
Chain 2 Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2 Iteration: 4100 / 5000 [ 82%]  (Sampling)
Chain 2 Iteration: 4200 / 5000 [ 84%]  (Sampling)
Chain 2 Iteration: 4300 / 5000 [ 86%]  (Sampling)
Chain 2 Iteration: 4400 / 5000 [ 88%]  (Sampling)
Chain 2 Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2 Iteration: 4600 / 5000 [ 92%]  (Sampling)
Chain 2 Iteration: 4700 / 5000 [ 94%]  (Sampling)
Chain 2 Iteration: 4800 / 5000 [ 96%]  (Sampling)
Chain 2 Iteration: 4900 / 5000 [ 98%]  (Sampling)
Chain 2 Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2 finished in 2183.9 seconds.

All 4 chains finished successfully.
Mean chain execution time: 950.9 seconds.
Total execution time: 2184.2 seconds.

Warning: 83 of 3200 (3.0%) transitions hit the maximum treedepth limit of 10.
See https://mc-stan.org/misc/warnings for details.
```

```r
# Print a summary of the model results
summary <- fit$summary(variables = c("a", "b", "sigma"))
```

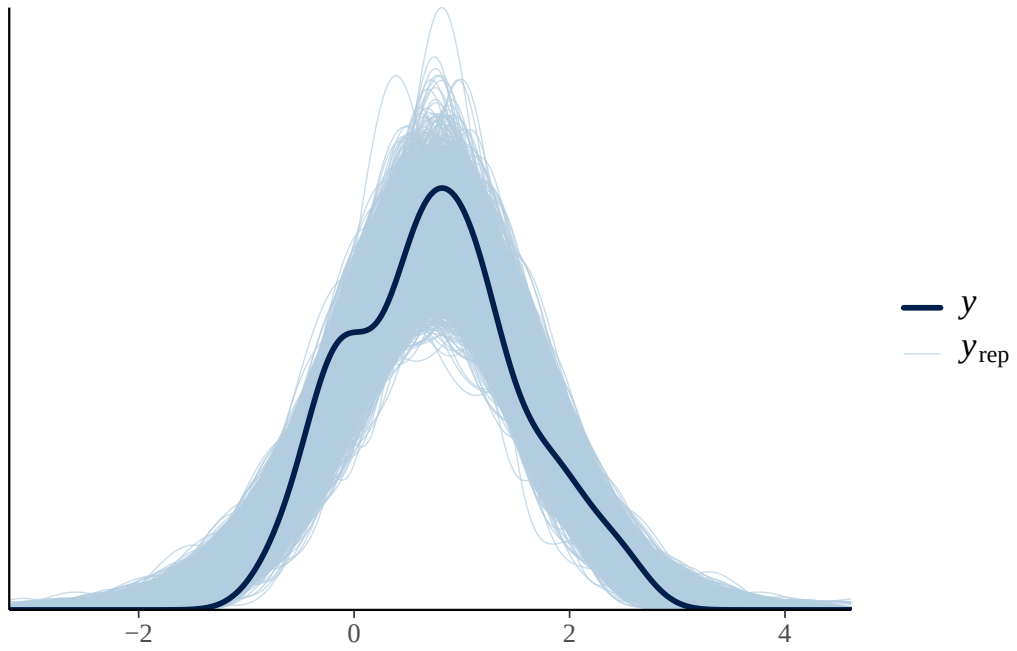
```r
# fit$cmdstan_diagnose()       # Diagnose any issues
fit$draws(variables = c("sigma")) %>% bayesplot::mcmc_trace()  # Trace plot
```




```r
posterior_predictive <- fit$draws(variables = "ytilde")

# Convert posterior predictive samples to a matrix
y_tilde_matrix <- posterior::as_draws_matrix(posterior_predictive)

# Perform posterior predictive density overlay
bayesplot::ppc_dens_overlay(y[1:125], y_tilde_matrix[,1:125])
```



The results suggest that the model fits okay to the actual data. Futher exploration can focus on simpler or model taking account variables like time points.