Lab 1

Ananya Bhaktaram

2025-09-09

Part A

Multivariate Gaussian (aka “normal” - it is not) distribution: the univariate Gaussian distribution can be extended to handle a vector of two (“bivariate”) or more (“multivariate”) random variables of length n. In this case, the mean vector also has length n and the variance/covariance matrix is n x n (n rows and n columns). The n diagonal elements of the square covariance matrix contain the variances for the n individual random variable, and the off-diagonal elements are the covariances for each of the “n choose 2” (n(n-1)/2) possible pairs of random variables.

For instance, the multivariate Gaussian distribution for $\begin{pmatrix} Y_{i1} \\ Y_{i2} \\ Y_{i3} \end{pmatrix}$ is given by:

$$\begin{pmatrix} Y_{i1} \\ Y_{i2} \\ Y_{i3} \end{pmatrix} \sim MVN \left(\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{12} & \sigma_1\sigma_3\rho_{13} \\ \sigma_1\sigma_2\rho_{12} & \sigma_2^2 & \sigma_2\sigma_3\rho_{23} \\ \sigma_1\sigma_3\rho_{13} & \sigma_2\sigma_3\rho_{23} & \sigma_3^2 \end{pmatrix} \right)$$

Question 1:

What do the values μ_2 and ⟦σ_2⟧ ^2 represent? Hand draw the distribution of Y_i2 labeling the mean and standard deviation in your figure. (10 pts)

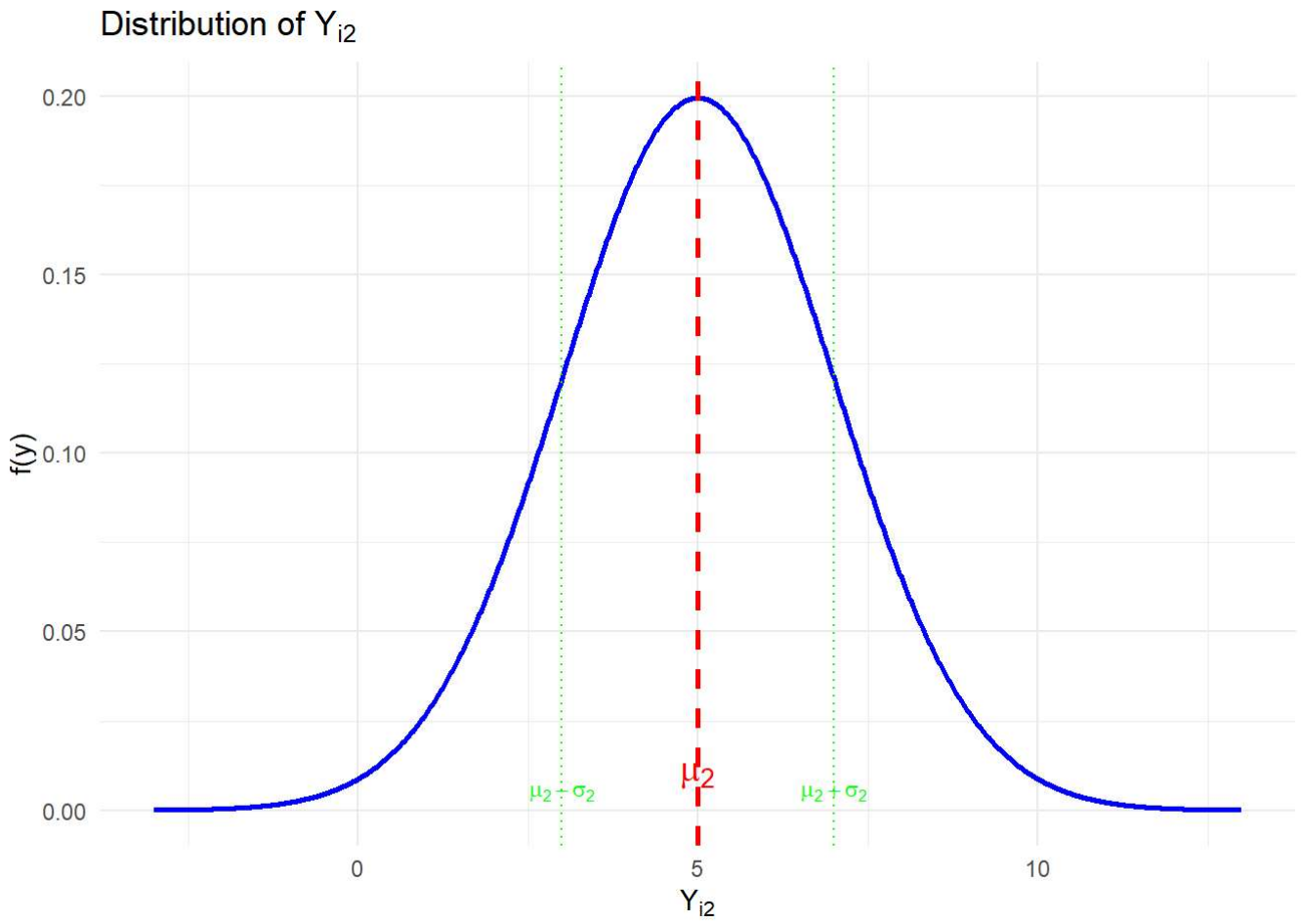
μ_2 is the population variable mean at Y_i2.

σ_2⟧ ^2 is the populations variance at time Y_i2.

```r
## Visualization of Y_i2 distribution
library(ggplot2)

library(ggplot2)

# Set parameters
mu2 <- 5
sigma2 <- 2

# Create data frame
x <- seq(mu2 - 4*sigma2, mu2 + 4*sigma2, length.out = 1000)
df <- data.frame(x = x, y = dnorm(x, mu2, sigma2))

# Create plot
ggplot(df, aes(x = x, y = y)) +
  geom_line(color = "blue", size = 1) +
  geom_vline(xintercept = mu2, color = "red", linetype = "dashed", size = 1) +
  geom_vline(xintercept = c(mu2 - sigma2, mu2 + sigma2),
             color = "green", linetype = "dotted") +
  annotate("text", x = mu2, y = 0.01, label = expression(mu[2]),
           color = "red", size = 5) +
  annotate("text", x = mu2 - sigma2, y = 0.005,
           label = expression(mu[2] - sigma[2]), color = "green", size = 3) +
  annotate("text", x = mu2 + sigma2, y = 0.005,
           label = expression(mu[2] + sigma[2]), color = "green", size = 3) +
  labs(title = expression("Distribution of Y"[i2]),
       x = expression("Y"[i2]),
       y = "f(y)") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'
## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'
## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'
```



Question 2:

What does the value σ_1 σ_3 ρ_13 represent? (10 pts)

The covariance between the first population mean and the third population mean, which is equivalent to SQRT Var (Yi1) SQRT Var (Yi3) * Corr (Yi1Yi3).

Question 3:

Interpret the values ρ_12, ρ_13 and ρ_23? (10 pts)

ρ_12 represents the correlation between time point 1 and 2.

ρ_13 represents the correlation between time point 1 and 3.

ρ_23 represents the correlation between time point 2 and 3.

Part B

Consider the multivariate Gaussian model for our application where, for each individual, we have 5 repeated measurements of the SF-36 mental health score at hospital discharge and then monthly for 4 months. The parameters associated with this multivariate (n = 5) Gaussian distribution comprise: 5 means, 5 variances and (5×(5-1))/2=10 pairwise correlations.

Let Y_(ij)be the SF-36 mental health score for subject i = 1, …, 100, at visit j = 1, 2, 3, 4, 5 corresponding to hospital discharge (j = 1), and subsequent monthly assessments (j = 2, 3, 4, 5).

Assume the data for subject i are generated at random from a multivariate Gaussian distribution with: means 35, 38, 43, 49, 48 for j = 1, 2, 3, 4, 5; equal variances over time of 100; and correlation ρjk for times j and k of ρ12 = 0.85, ρ13 = 0.80, ρ14 = 0.72, ρ15 = 0.69, ρ23 = 0.85, ρ24 = 0.80, ρ25 = 0.72, ρ34 = 0.85, ρ35 = 0.80, ρ45 = 0.85.

Question 4

In the space below, practice your understanding of the notation by writing out the multivariate (n=5) Gaussian distribution. (10 pts)

$$\begin{pmatrix} Y_{i1} \\ Y_{i2} \\ Y_{i3} \\ Y_{i4} \\ Y_{i5} \end{pmatrix} \sim MVN \left(\begin{pmatrix} 35 \\ 38 \\ 43 \\ 39 \\ 48 \end{pmatrix}, \begin{pmatrix} 1.00 & 0.85 & 0.80 & 0.72 & 0.69 \\ 0.85 & 1.00 & 0.85 & 0.80 & 0.72 \\ 0.80 & 0.85 & 1.00 & 0.85 & 0.80 \\ 0.72 & 0.80 & 0.85 & 1.00 & 0.85 \\ 0.69 & 0.72 & 0.80 & 0.85 & 1.00 \end{pmatrix} \right)$$

Question 5:

Describe in a sentence or two the pattern you observe in the correlations for pairs of observations. Use the “autocorrelation matrix” (ACM) above to obtain the “autocorrelation function”(ACF): (u)=corr(Y_{ij },Y_{ij+u}), u=1,…,4. Briefly describe the approach/method you used to obtain the ACF from the ACM. Given your lay understanding of the SF-36 measure, how is nature likely to work to produce the ACF pattern you observe. (10 pts)

The pattern shown in the correlations generally decreases as the time gap increases, which is typical for longitudinal data as measurements that are closer to one another in time are more likely to be highly correlated.

The autocorrelation function measures the correlation between measurements separated by u time units: (1) = the correlation between consecutive time points (lag 1) (2) = the correlation between time points that are 2 units apart (lag 2) (3) = the correlation between time points that are 3 units apart (lag 3) (4) = the correlation between time points that are 4 units apart (lag 4)

```
## Calculating the autocorrelation function

# Read in correlation matrix
R <- matrix(c(
  1.00, 0.85, 0.80, 0.72, 0.69,
  0.85, 1.00, 0.85, 0.80, 0.72,
  0.80, 0.85, 1.00, 0.85, 0.80,
  0.72, 0.80, 0.85, 1.00, 0.85,
  0.69, 0.72, 0.80, 0.85, 1.00
), nrow = 5, byrow = TRUE)

# Simple function to extract autocorrelations
get_autocorr <- function(corr_matrix) {
  n <- nrow(corr_matrix)
  autocorr <- numeric(n-1)

  for(u in 1:(n-1)) {
    # Extract diagonal u positions above main diagonal
    autocorr[u] <- corr_matrix[1, 1+u]  # Since structure is stationary
  }

  return(autocorr)
}

# Calculate
rho <- get_autocorr(R)
names(rho) <- paste0("lag_", 1:4)
print(rho)
```

```
## lag_1 lag_2 lag_3 lag_4
##  0.85  0.80  0.72  0.69
```

Because the correlation between any two time points only depends on their distance (the lag) not their absolute positions all correlations at the same lag are equal. Given that the the lags are symmetric the autocorrelation function is able to summarize the pattern between time points that are stored in the matrix. In this case, because the effects are stationary the correlations given by the ACF and the ACM are the same.

For example, an interpretation of both the ACM and ACF would be the same. In this case, a Patient's mental health (SF-36 score) at discharge (Time 1) has a 0.85 correlation with their 1-month follow-up (Time 2). Similarly, because the distance between Time 2 and Time 3 is the same as that between Time 1 and Time 2 this tells us that a Patient's SF-36 score at their 1 month follow up (Time 2) has a 0.85 correlation with their SF-36 score at their 2-month follow-up (Time 3). Overall, we see that the correlation decreases as the lag (time between follow-ups) increases. For example, the patient's SF-36 score at discharge (Time 1) only has a 0.69 correlation with their score at 4-month follow-up (TIme 5).

Part C: Simulating Multivariate Gaussian Data

Question 6:

Simulate 100 random draws from the multivariate Gaussian distribution for the SF-36 data whose true mean and covariance matrix is specified above (and in the code). Report the first 5 draws in the space below. (10 pts)

```
library(mvtnorm)
set.seed(02022022) # Important for reproducibility
mm <- c(35, 38, 43, 49, 48)
C <- matrix(c(1.00,0.85,0.80,0.72,0.69,
              0.85,1.00,0.85,0.80,0.72,
              0.80,0.85,1.00,0.85,0.80,
              0.72,0.80,0.85,1.00,0.85,
              0.69,0.72,0.80,0.85,1.00),
            nrow = 5)
sigma <- C * 100
y <- rmvnorm(n=100, mean=mm, sigma=sigma)
id <- seq(1,100)
dat <- as.data.frame(cbind(y,id))
names(dat) <- c("y0","y1","y2","y3","y4","id")

head(dat, 5)
```

```
##         y0       y1       y2       y3       y4 id
## 1 32.12414 43.65175 53.33750 60.83278 57.53808  1
## 2 34.77750 33.98437 36.74835 35.05546 36.15158  2
## 3 14.84156 18.32857 26.59074 29.08847 29.43191  3
## 4 18.49030 27.81742 28.91882 39.05716 34.13245  4
## 5 24.50810 23.75607 34.01307 38.81020 39.54751  5
```

Question 7:

Display your simulated data using a spaghetti plot and a pairs plot. Calculate the mean and covariance matrix for your sample of n=100 simulated vectors. Use these plots and estimates to describe the patterns of potential scientific interest in the SF-36 data. (10 pts)

```
# Load Libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.4     ✓ tibble    3.3.0
## ✓ purrr     1.0.4     ✓ tidyr     1.3.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(nlme)
```

```
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(mvtnorm)
library(joineR)
```

```
## Loading required package: survival
```

```
library (GGally)
```

```
# Use your simulated SF-36 data (stored as dat)
# Convert to long format for analysis
df_long <- dat %>%
  pivot_longer(cols = y0:y4,
               names_to = "time_point",
               values_to = "SF36_score") %>%
  mutate(
    time_numeric = as.numeric(str_extract(time_point, "\\d+")),
    time_label = case_when(
      time_numeric == 0 ~ "Discharge",
      time_numeric == 1 ~ "Month 1",
      time_numeric == 2 ~ "Month 2",
      time_numeric == 3 ~ "Month 3",
      time_numeric == 4 ~ "Month 4"
    )
  )

# Count unique subjects
n <- length(unique(df_long$id))
print(paste0("Number of subjects: ", n))
```

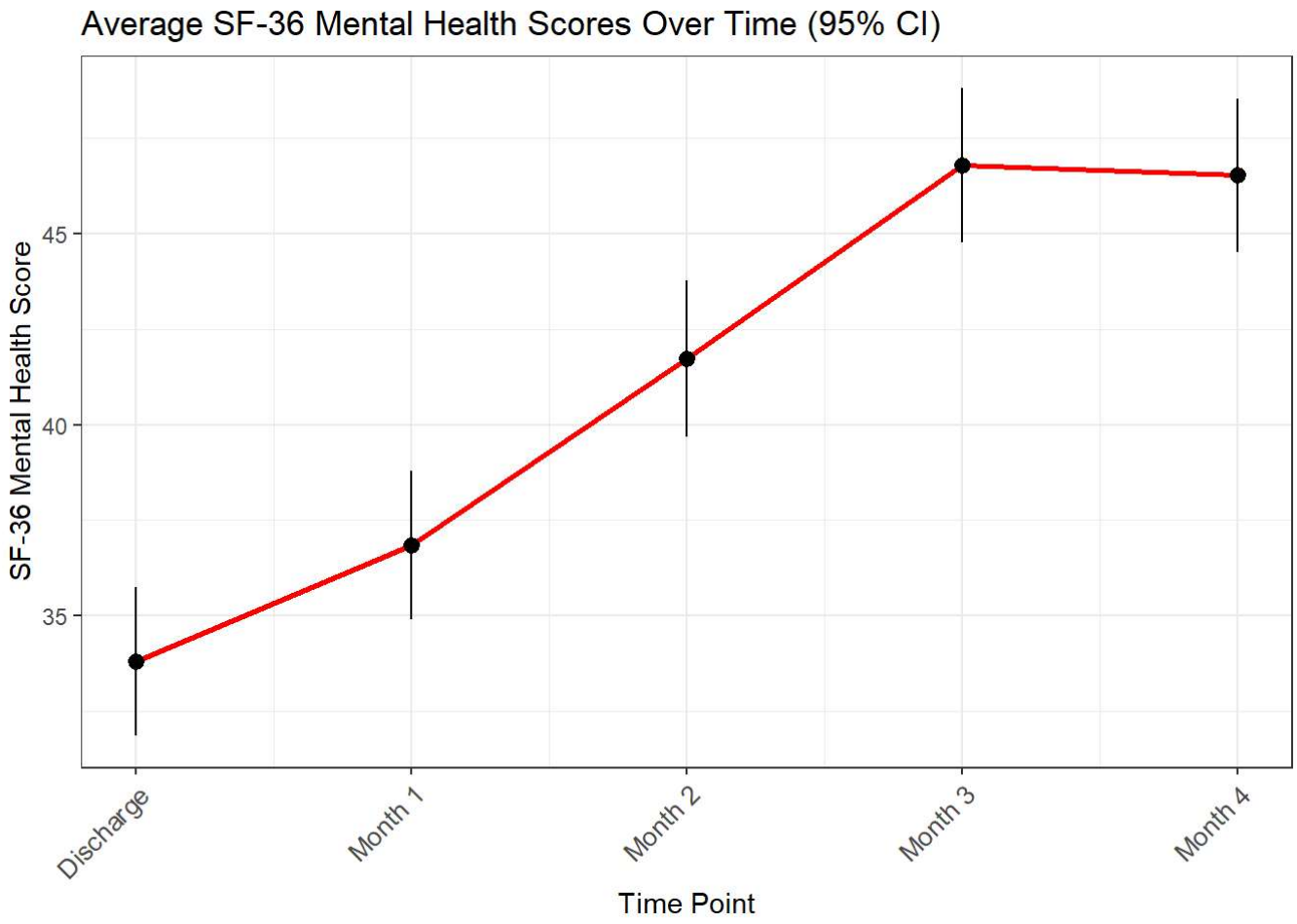
```
## [1] "Number of subjects: 100"
```

```
# Generate summaries for plotting
time.summaries <- df_long %>%
    group_by(time_numeric, time_label) %>%
    summarise(avgSF36 = mean(SF36_score),
              sdSF36 = sd(SF36_score),
              medSF36 = median(SF36_score),
              q75SF36 = quantile(SF36_score, 0.75),
              q25SF36 = quantile(SF36_score, 0.25),
              .groups = 'drop')

print(time.summaries)
```

```
## # A tibble: 5 × 7
##   time_numeric time_label avgSF36 sdSF36 medSF36 q75SF36 q25SF36
##          <dbl> <chr>        <dbl>  <dbl>   <dbl>   <dbl>   <dbl>
## 1            0 Discharge     33.8   9.70    34.2    38.8    27.6
## 2            1 Month 1       36.8   9.77    36.9    43.2    31.7
## 3            2 Month 2       41.7  10.2     40.7    50.0    34.3
## 4            3 Month 3       46.8  10.1     47.4    54.6    39.2
## 5            4 Month 4       46.5  10.1     46.1    54.3    39.8
```

Visualize the sample means with a 95% confidence interval

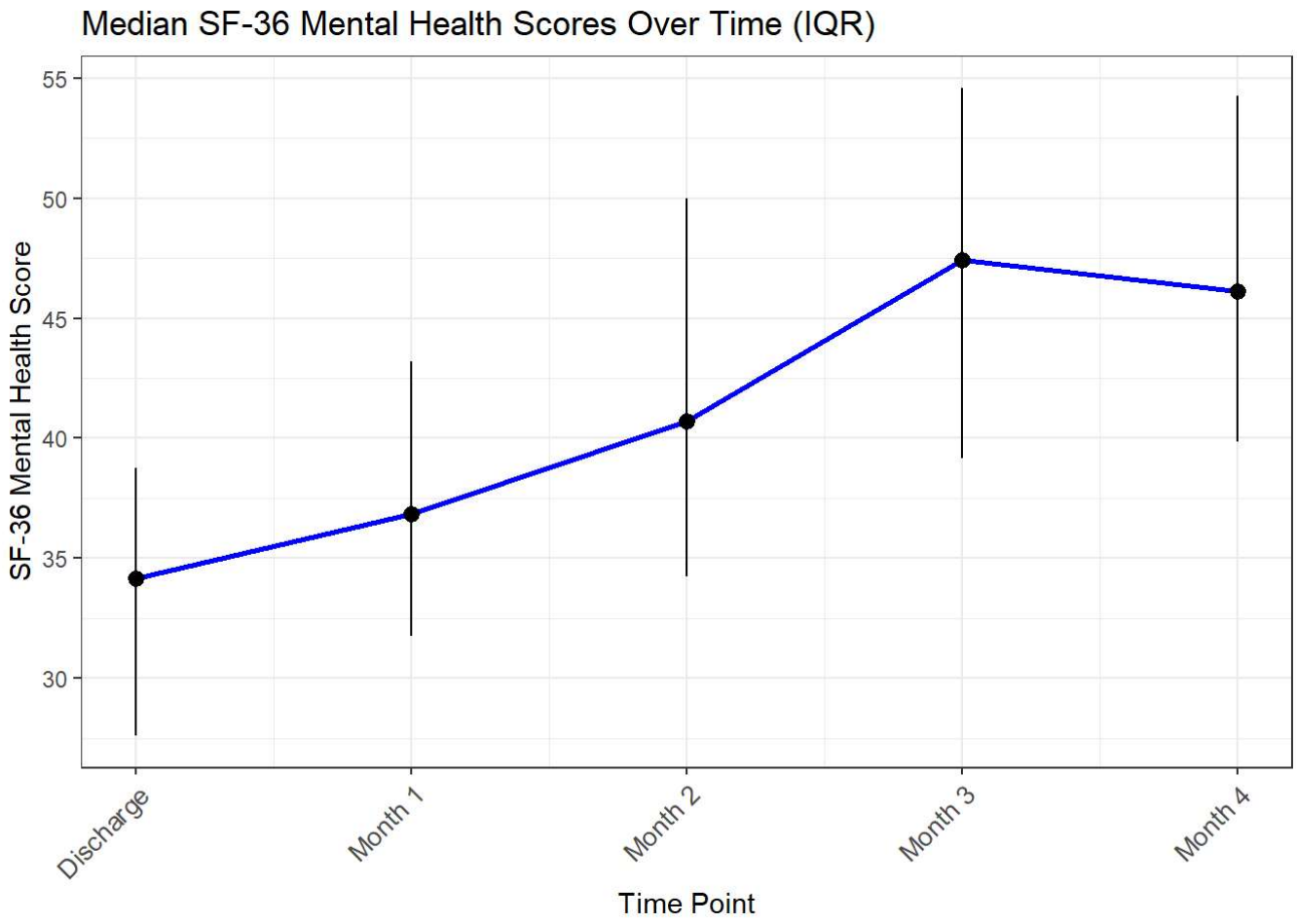
```
time.summaries %>%
    ggplot() +
    geom_point(aes(x = time_numeric, y = avgSF36), color = "red", size = 2) +
    geom_line(aes(x = time_numeric, y = avgSF36), color = "red", size = 1) +
  geom_pointrange(aes(x = time_numeric, y = avgSF36,
                      ymin = avgSF36 - 2*sdSF36/sqrt(n),
                      ymax = avgSF36 + 2*sdSF36/sqrt(n))) +
    scale_x_continuous(breaks = 0:4,
                       labels = c("Discharge", "Month 1", "Month 2", "Month 3", "Month 4")) +
  scale_y_continuous(breaks = seq(30, 55, 5)) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10)) +
    xlab("Time Point") +
    ylab("SF-36 Mental Health Score") +
    ggtitle("Average SF-36 Mental Health Scores Over Time (95% CI)")
```



Plot

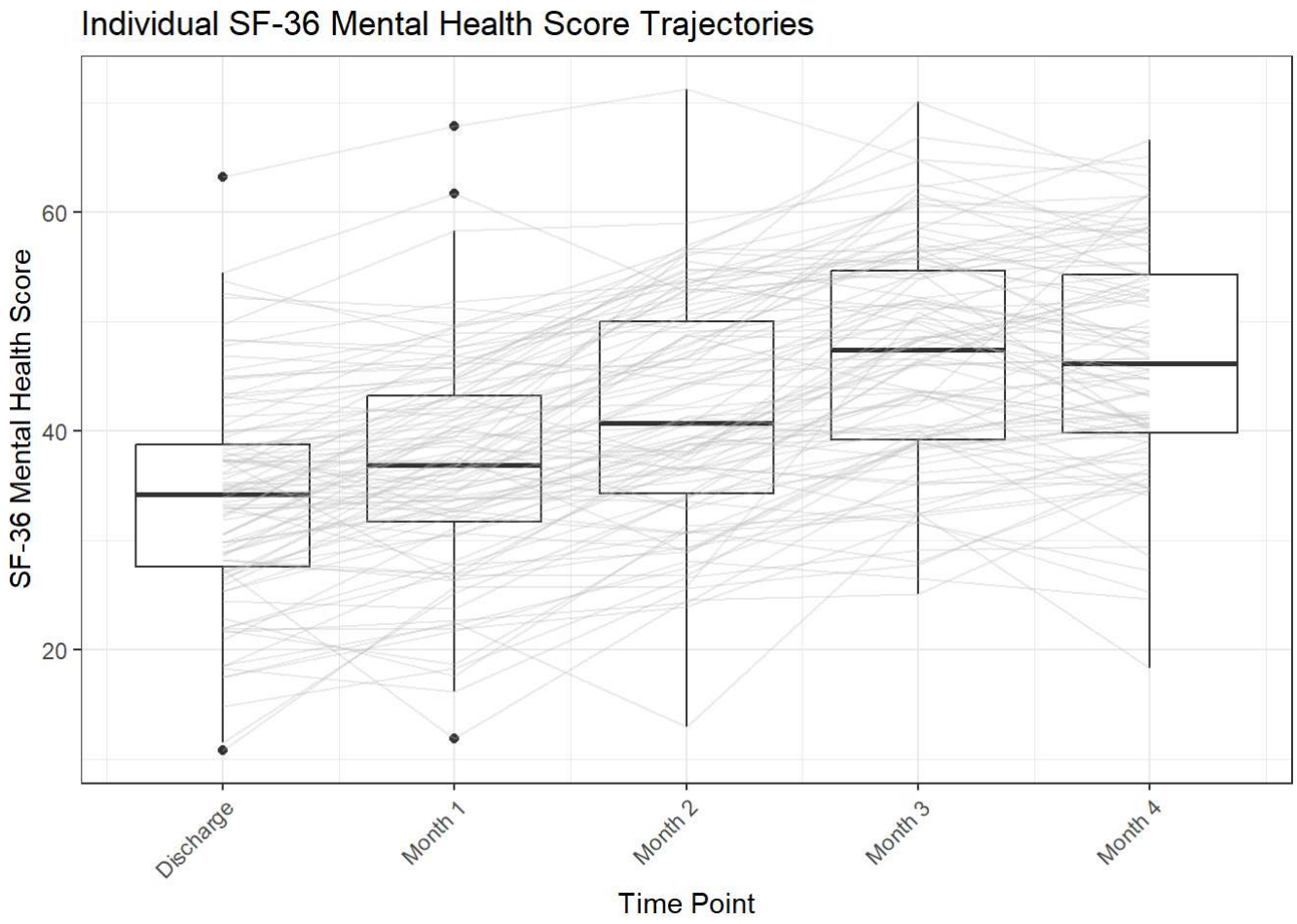
median and IQR

```
time.summaries %>%
  ggplot() +
  geom_point(aes(x = time_numeric, y = medSF36), color = "blue", size = 2) +
  geom_line(aes(x = time_numeric, y = medSF36), color = "blue", size = 1) +
  geom_pointrange(aes(x = time_numeric, y = medSF36,
                      ymin = q25SF36, ymax = q75SF36)) +
  scale_x_continuous(breaks = 0:4,
                     labels = c("Discharge", "Month 1", "Month 2", "Month 3", "Month 4")) +
  scale_y_continuous(breaks = seq(30, 55, 5)) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10)) +
  xlab("Time Point") +
  ylab("SF-36 Mental Health Score") +
  ggtitle("Median SF-36 Mental Health Scores Over Time (IQR)")
```



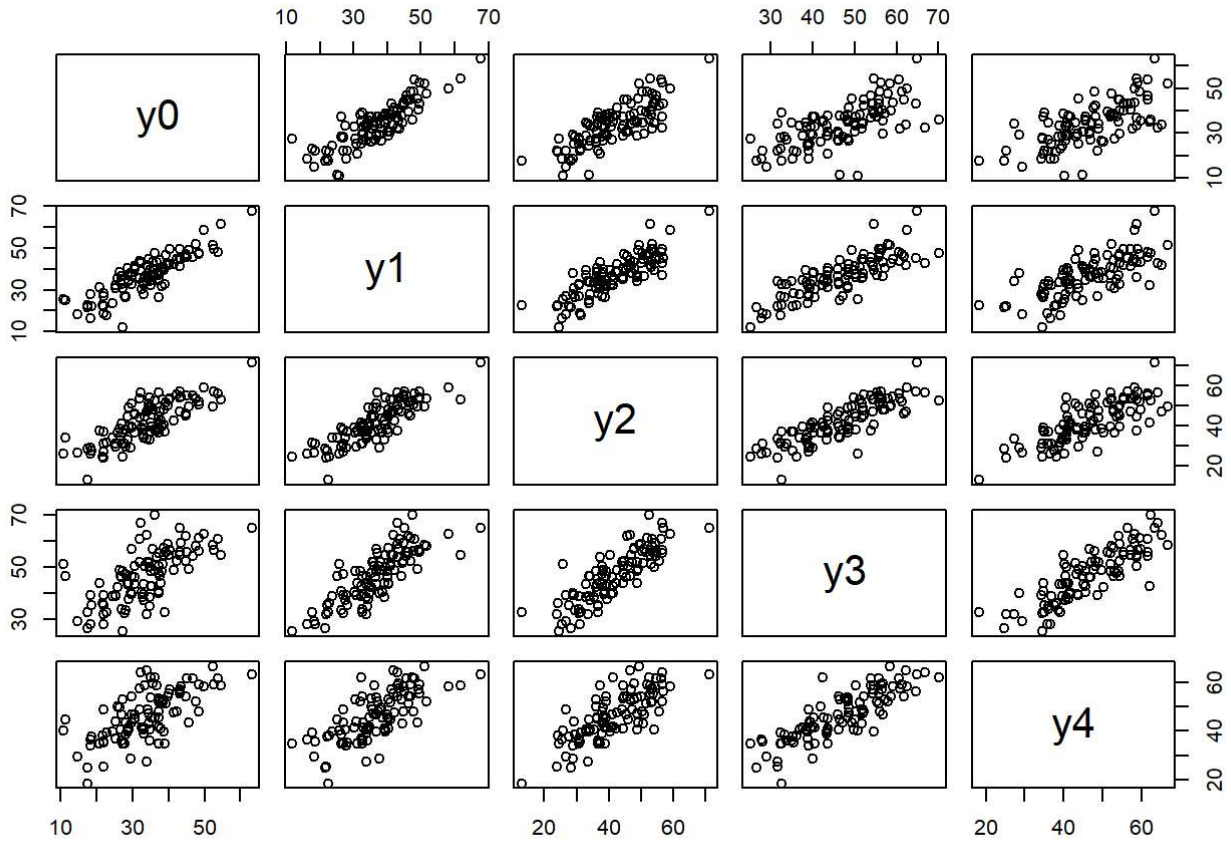
Look at distributions within each timepoint using boxplots

```
df_long %>%
  ggplot() +
  geom_boxplot(aes(group = time_numeric, x = time_numeric, y = SF36_score)) +
  theme_bw() +
  scale_x_continuous(breaks = 0:4,
                     labels = c("Discharge", "Month 1", "Month 2", "Month 3", "Month 4")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  xlab("Time Point") +
  ylab("SF-36 Mental Health Score") +
  geom_line(aes(group = id, x = time_numeric, y = SF36_score),
            alpha = 0.3, colour = "grey") +
  ggtitle("Individual SF-36 Mental Health Score Trajectories")
```



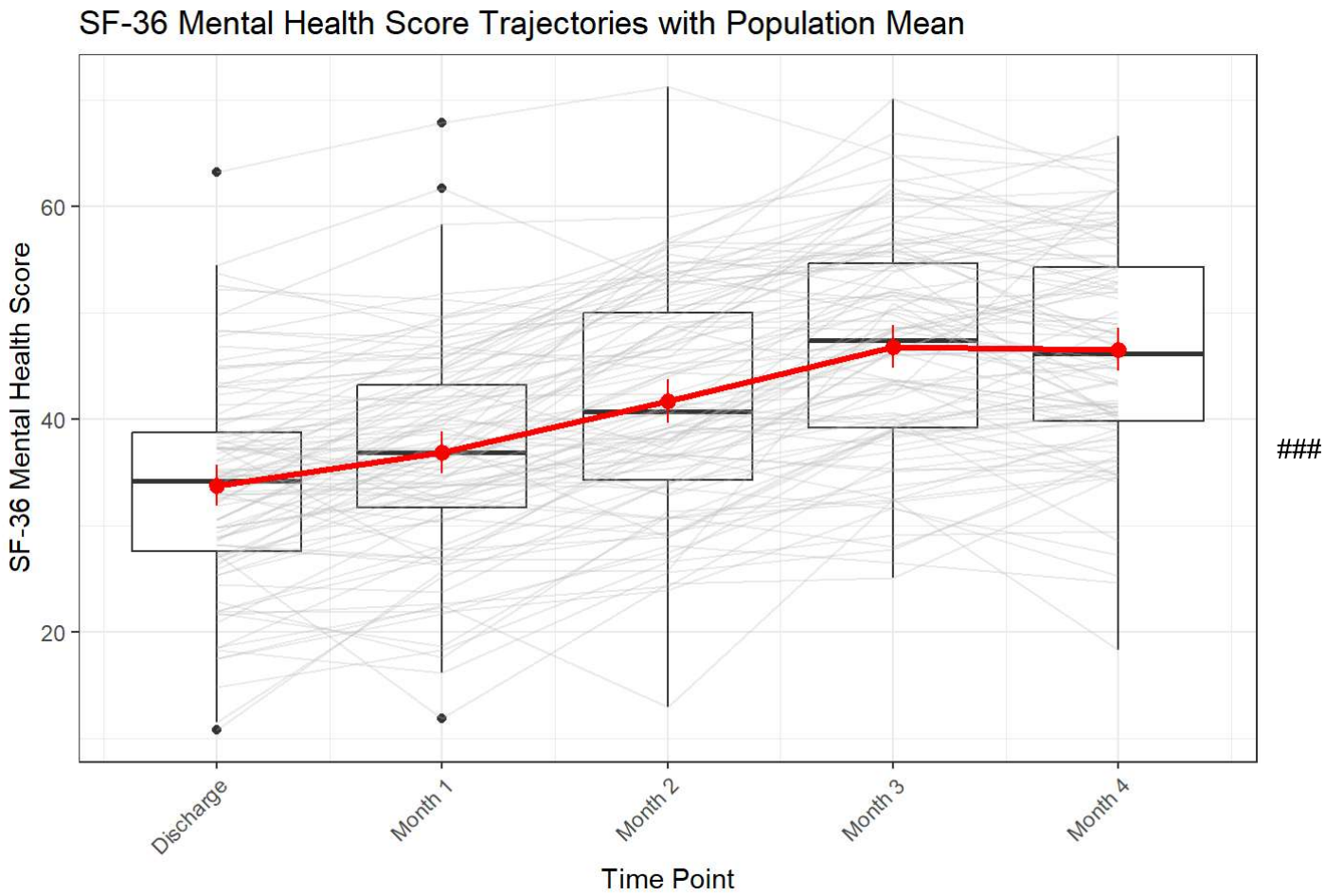
Plot

```
pairs_dat <- subset(dat,select=-id)
pairs(pairs_dat)
```



Overlay mean trajectory to create final spaghetti plot

```
ggplot(df_long) +
  geom_boxplot(aes(group = time_numeric, x = time_numeric, y = SF36_score)) +
    geom_line(aes(group = id, x = time_numeric, y = SF36_score),
              alpha = 0.3, colour = "grey") +
    theme_bw() +
    scale_x_continuous(breaks = 0:4,
                       labels = c("Discharge", "Month 1", "Month 2", "Month 3", "Month 4")) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    xlab("Time Point") +
    ylab("SF-36 Mental Health Score") +
  geom_point(data = time.summaries, aes(x = time_numeric, y = avgSF36),
             color = "red", size = 2) +
  geom_line(data = time.summaries, aes(x = time_numeric, y = avgSF36), #This is assing the mean
trajctory line
            color = "red", size = 1.2) +
  geom_pointrange(data = time.summaries,
                  aes(x = time_numeric, y = avgSF36,
                      ymin = avgSF36 - 2*sdSF36/sqrt(n),
                      ymax = avgSF36 + 2*sdSF36/sqrt(n)),
                  color = "red") +
  ggtitle("SF-36 Mental Health Score Trajectories with Population Mean")
```



Question 8: 8. Calculate the autocorrelation matrix and autcorrelation function for your simulated data. Describe what else these estimates contribute to our understanding of the SF-36 data. (10 pts)

```
# Calculate covariance matrix
cov_mat = dat %>%
  select(-id) %>%
  cov()

cov_mat[upper.tri(cov_mat)] = NA
cov_mat %>%
  as.data.frame()
```

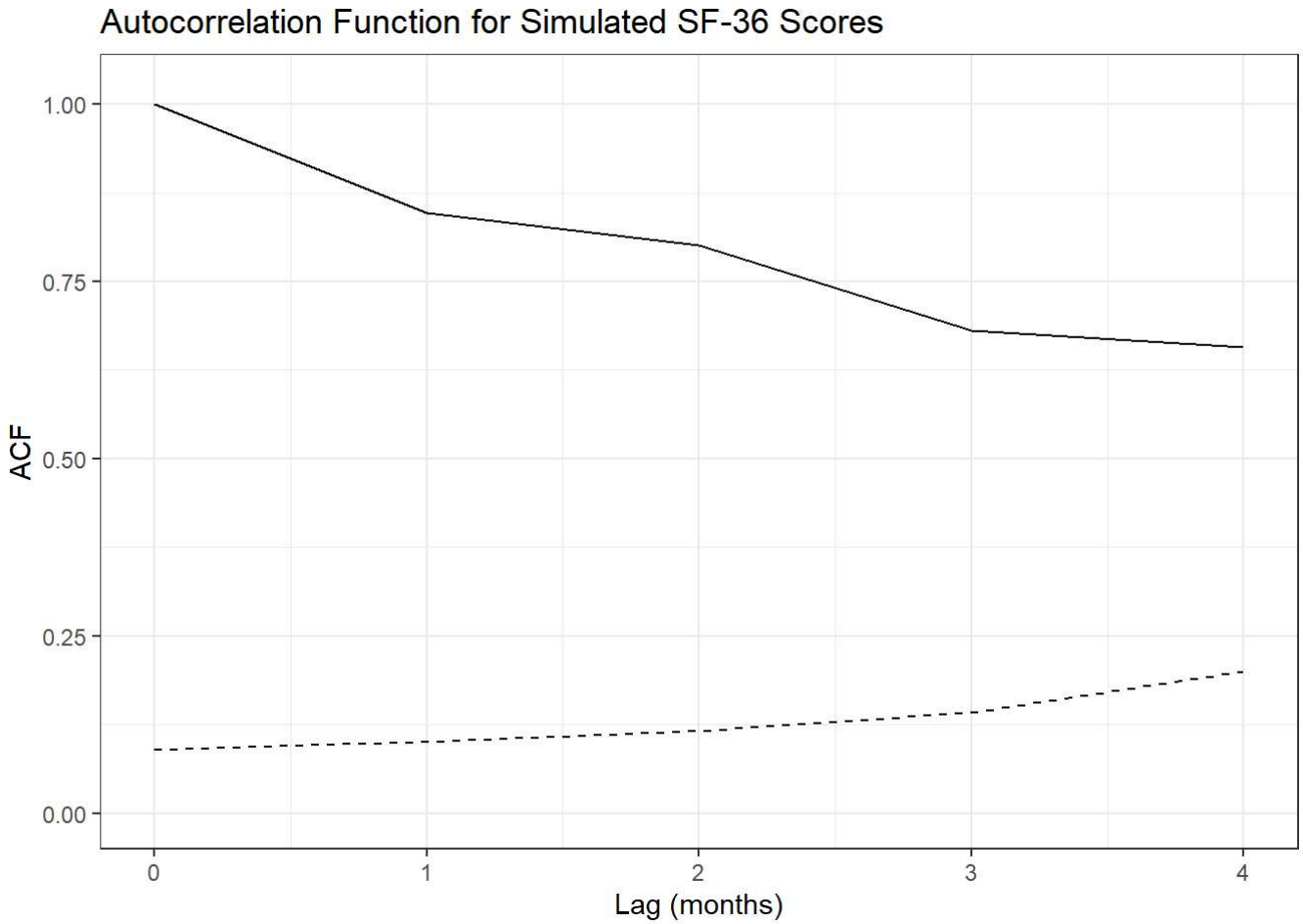
```
##          y0       y1        y2        y3       y4
## y0 94.06231       NA        NA        NA       NA
## y1 80.83114 95.43100        NA        NA       NA
## y2 78.99086 84.30181 104.42008        NA       NA
## y3 64.89921 80.48477  86.62631 101.81906       NA
## y4 65.49442 70.66869  79.81576  85.48809 101.9478
```

Calculate autocorrelation of the residuals

```
# The autocor command requires you to give: the residual, time variable, id variable
fit = gls(SF36_score~as.factor(time_numeric),df_long)
acf1 = ACF(fit,form= ~1|id)
acf1
```

```
##   lag       ACF
## 1   0 1.0000000
## 2   1 0.8470482
## 3   2 0.8013558
## 4   3 0.6809990
## 5   4 0.6579970
```

```
# generate the lower bound
acf1 = acf1 %>%
    mutate(upper = 2 * 1 / sqrt(100 * (5-lag)))

# Make the plot
acf1 %>%
  ggplot() +
  geom_line(aes(x=lag, y=ACF))  +
  geom_line(aes(x=lag, y=upper), linetype=2) +
  xlab("Lag (months)") +
  ylim(0,1.02) +
  theme_bw() +
  ggtitle("Autocorrelation Function for Simulated SF-36 Scores")
```



The solid line of this plot displays the autocorrelation function (ACF) for the simulated Sf-36 mental health data, while the dashed line shows the upper bound of a 95% confidence interval. Values above the line indicate significant autocorrelation, while values below this line could be attributed to random chance. This shows high autocorrelation

at all lags, with an overall decreasing trend, meaning that measurements that are closer in time are more similar to one another than those that are more distant. All values are above the 95% confidence interval indicating that the temporal dependence displayed in the simulated data is statistically significant.

This tells us that current SF-36 scores strongly predict future scores, and that overall mental health improvements or declines tend to persist across time.

Question 9:

Now generate a new data set in a multilevel manner, producing a 100 x 5 matrix, call it Y. (10 pts)

To generate each row, 5x1 vectors $Y_i = \begin{pmatrix} Y_{i1} & Y_{i2} & Y_{i3} & Y_{i4} & Y_{i5} \end{pmatrix}$ for each person i=1,,100, we use the equation

$$Y_{ij} = m_j + b_i + e_{ij} \quad j = 1, \ldots, 5; i = 1, \ldots, 100$$

where m_j is the mean value for the jth random variable common to all 100 people (no subscript i) (we will use $m = \begin{pmatrix} 35 & 38 & 43 & 49 & 48 \end{pmatrix}$ from above), b_i is a scalar “random intercept”, common to all 5 observations for person i, but different among people, and e_{ij} is a residual or “error” specific to observation j for person i. We assume that: (1) the 100 b_i’s are independent and identically distributed (“IID”) draws from a Gaussian distribution with mean 0 and variance 120; (2) the 500=100x5 e_{ij}’s are iid Gaussian with mean 0 and variance 100, and that (3) the b_i’s are independent of the e_{ij}’s.

1. generate a 100 x 5 matrix of e_{ij}’s by drawing 100 realizations from a 5-dimensional Gaussian distribution with mean 0 and covariance matrix that is diagonal with 100 on the main diagonal and 0 off the diagonal. Because the covariances are all 0 and the data are multivariate Gaussian, the repeated observations for an individual are independent of one another.
2. generate a vector of 100 independent univariate Gaussian variates with mean zero and variance 120. These values are the 100 “random intercepts”, one for each person (row).
3. Create the 100 x 5 matrix of Y values by adding the mean vector (repeated over rows), random intercepts (repeated over columns), and the random errors matrix.

Simulate new data

```
library(mvtnorm)

# Define population parameters
set.seed(02022022) # Important for reproducibility

# Population means (m_j) are the same for all 100 people
mm <- c(35, 38, 43, 49, 48)
cat("Population means m_j:", mm, "\n")
```

```
## Population means m_j: 35 38 43 49 48
```

```
# Model parameters
n_subjects <- 100 # i = 1, 2, ... 100 people
n_times <- 5 # j = 1,2,3,4,5  (5 time points)

# Establish Variance parameters
var_b <- 120 # Variance of random intercepts b1
var_e <- 100 # Variance of random errors e_ij
cat("Random intercept variance (var_b):", var_b, "\n")
```

```
## Random intercept variance (var_b): 120
```

```
cat("Random error variance (var_e):", var_e, "\n\n")
```

```
## Random error variance (var_e): 100
```

Generate Random Intercepts

```
cat("Generating random intercepts b_i...\n")
```

```
## Generating random intercepts b_i...
```

```
# Generate 100 random intercepts (one per person)
b_i <- rnorm(n = n_subjects, mean = 0, sd = sqrt(var_b))

cat("First 10 random intercepts:", round(b_i[1:10], 2), "\n")
```

```
## First 10 random intercepts: -16.69 3.61 12.32 12.23 6.33 8.2 1.26 0.02 -18.35 -10.49
```

```
cat("Sample mean of b_i (should ≈ 0):", round(mean(b_i), 2), "\n")
```

```
## Sample mean of b_i (should ≈ 0): -1.34
```

```
cat("Sample variance of b_i (should ≈ 120):", round(var(b_i), 2), "\n\n")
```

```
## Sample variance of b_i (should ≈ 120): 118.34
```

Generate Random Errors Matrix

```
# Generate 500 = 100×5 independent random errors
e_ij <- matrix(rnorm(n = n_subjects * n_times, mean = 0, sd = sqrt(var_e)),
               nrow = n_subjects, ncol = n_times)

cat("Error matrix dimensions:", dim(e_ij), "\n")
```

```
## Error matrix dimensions: 100 5
```

```
cat("Sample mean of all e_ij (should ≈ 0):", round(mean(e_ij), 2), "\n")
```

```
## Sample mean of all e_ij (should ≈ 0): -0.83
```

```
cat("Sample variance of all e_ij (should ≈ 100):", round(var(as.vector(e_ij)), 2), "\n")
```

```
## Sample variance of all e_ij (should ≈ 100): 107.12
```

```
cat("Sample correlation between person 1 and person 2 errors:",
    round(cor(e_ij[1, ], e_ij[2, ]), 3), "\n")
```

```
## Sample correlation between person 1 and person 2 errors: -0.533
```

Create Component Matrices for the Model

```
# Mean matrix: replicate m_j for each person (100 identical rows)
m_matrix <- matrix(rep(mm, n_subjects),
                   nrow = n_subjects, ncol = n_times, byrow = TRUE)

# Random intercept matrix: replicate each b_i across all time points
b_matrix <- matrix(rep(b_i, n_times),
                   nrow = n_subjects, ncol = n_times, byrow = FALSE)

cat("Mean matrix (first 3 rows):\n")
```

```
## Mean matrix (first 3 rows):
```

```
print(m_matrix[1:3, ])
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   35   38   43   49   48
## [2,]   35   38   43   49   48
## [3,]   35   38   43   49   48
```

```
cat("Random intercept matrix (first 3 rows):\n")
```

```
## Random intercept matrix (first 3 rows):
```

```
print(round(b_matrix[1:3, ], 2))
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]
## [1,] -16.69 -16.69 -16.69 -16.69 -16.69
## [2,]   3.61   3.61   3.61   3.61   3.61
## [3,]  12.32  12.32  12.32  12.32  12.32
```

Creation of simulated Y matrix by adding all of the components

```
# Create Y matrix by adding all components
Y <- m_matrix + b_matrix + e_ij

cat("Simulated Y matrix dimensions:", dim(Y), "\n")
```

```
## Simulated Y matrix dimensions: 100 5
```

```
cat("First 5 subjects' Y values:\n")
```

```
## First 5 subjects' Y values:
```

```
print(round(Y[1:5, ], 2))
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 11.82 10.60 35.44 36.36 39.66
## [2,] 52.61 49.44 51.47 44.00 54.56
## [3,] 37.12 57.33 49.92 63.87 66.98
## [4,] 40.22 52.25 52.98 52.63 48.83
## [5,] 42.66 37.25 47.76 51.75 50.20
```

Add subject ID's to Simulated Y Matrix

```
# Create subject IDs
id <- seq(1, 100)

# Add IDs directly to Y matrix as a new column
Y <- cbind(Y, id)

# Add column names to the Y matrix
colnames(Y) <- c("y0", "y1", "y2", "y3", "y4", "id")

cat("First 5 rows of Y matrix:\n")
```

```
## First 5 rows of Y matrix:
```

```
print(round(Y[1:5, ], 2))
```

```
##         y0    y1    y2    y3    y4 id
## [1,] 11.82 10.60 35.44 36.36 39.66  1
## [2,] 52.61 49.44 51.47 44.00 54.56  2
## [3,] 37.12 57.33 49.92 63.87 66.98  3
## [4,] 40.22 52.25 52.98 52.63 48.83  4
## [5,] 42.66 37.25 47.76 51.75 50.20  5
```

```
cat("Structure of Y matrix:\n")
```

```
## Structure of Y matrix:
```

```
str(Y)
```

```
##  num [1:100, 1:6] 11.8 52.6 37.1 40.2 42.7 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "y0" "y1" "y2" "y3" ...
```

Question 10:

Again, display your simulated data using a spaghetti plot and pairs plot. (10 pts)

Convert Y matrix to long format in order to plot

```
# Convert Y matrix to long format for ggplot
Y_long <- Y %>%
  as.data.frame() %>%
  pivot_longer(cols = y0:y4,
               names_to = "time_point",
               values_to = "SF36_score") %>%
  mutate(
    time_numeric = as.numeric(str_extract(time_point, "\\d+")),
    time_label = case_when(
      time_numeric == 0 ~ "Discharge",
      time_numeric == 1 ~ "Month 1",
      time_numeric == 2 ~ "Month 2",
      time_numeric == 3 ~ "Month 3",
      time_numeric == 4 ~ "Month 4"
    )
  )
```

Calculate summary statistics for the new data

```
n <- length(unique(Y_long$id))  # Number of subjects (should be 100)

time.summaries <- Y_long %>%
    group_by(time_numeric, time_label) %>%
    summarise(avgSF36 = mean(SF36_score),
              sdSF36 = sd(SF36_score),
              medSF36 = median(SF36_score),
              q75SF36 = quantile(SF36_score, 0.75),
              q25SF36 = quantile(SF36_score, 0.25),
              .groups = 'drop')

cat("Summary statistics for new simulated data:\n")
```

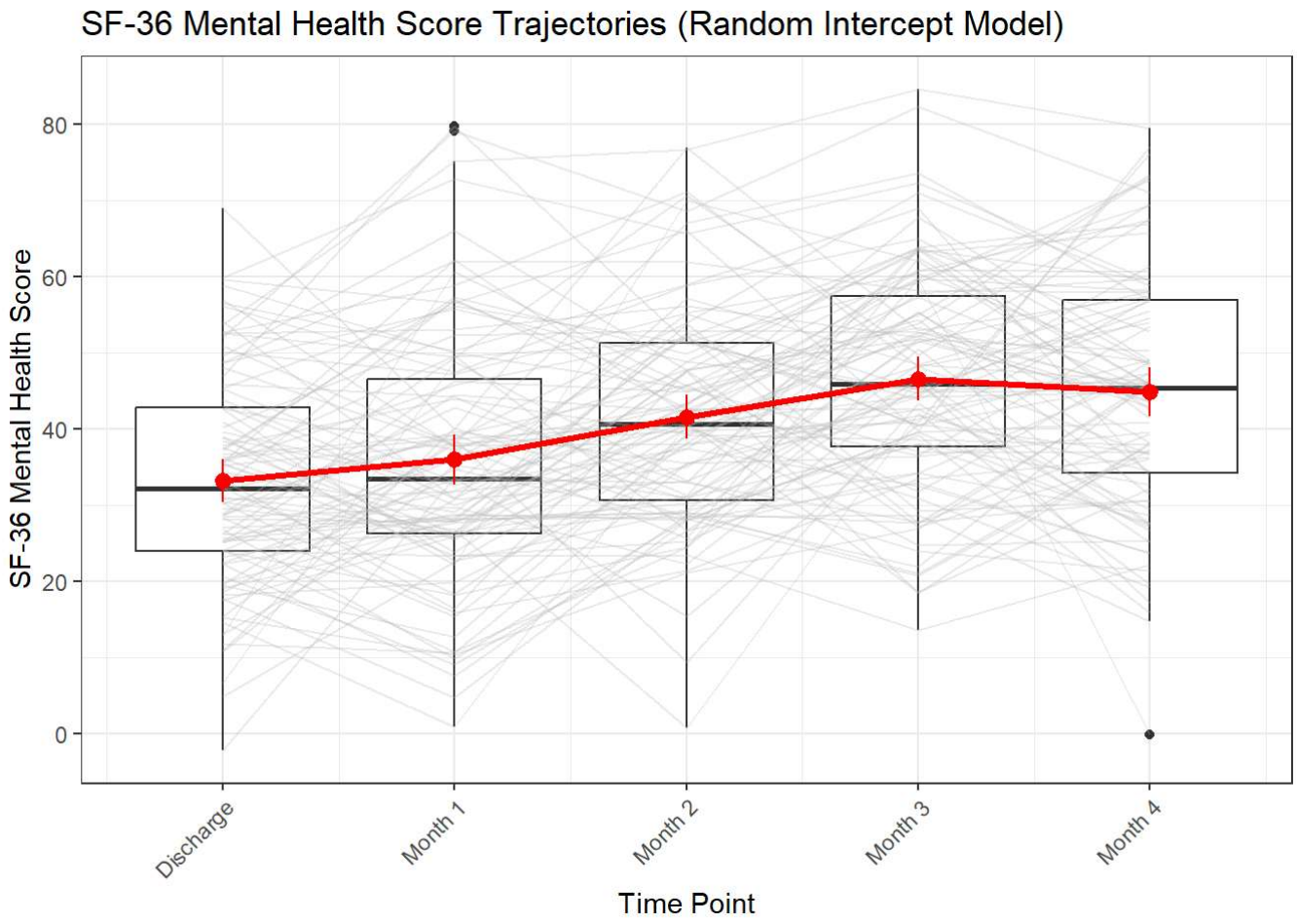
```
## Summary statistics for new simulated data:
```

```
print(time.summaries)
```

```
## # A tibble: 5 × 7
##   time_numeric time_label avgSF36 sdSF36 medSF36 q75SF36 q25SF36
##          <dbl> <chr>        <dbl>  <dbl>   <dbl>   <dbl>   <dbl>
## 1            0 Discharge     33.2   14.1    32.2    42.8    24.0
## 2            1 Month 1       36.0   16.4    33.5    46.5    26.3
## 3            2 Month 2       41.6   14.4    40.6    51.3    30.6
## 4            3 Month 3       46.6   14.6    45.9    57.4    37.7
## 5            4 Month 4       44.8   16.0    45.3    56.9    34.2
```

Create the spaghetti plot

```
ggplot(Y_long) +
  geom_boxplot(aes(group = time_numeric, x = time_numeric, y = SF36_score)) +
    geom_line(aes(group = id, x = time_numeric, y = SF36_score),
              alpha = 0.3, colour = "grey") +
    theme_bw() +
    scale_x_continuous(breaks = 0:4,
                       labels = c("Discharge", "Month 1", "Month 2", "Month 3", "Month 4")) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    xlab("Time Point") +
    ylab("SF-36 Mental Health Score") +
  geom_point(data = time.summaries, aes(x = time_numeric, y = avgSF36),
             color = "red", size = 2) +
  geom_line(data = time.summaries, aes(x = time_numeric, y = avgSF36),
            color = "red", size = 1.2) +
  geom_pointrange(data = time.summaries,
                  aes(x = time_numeric, y = avgSF36,
                      ymin = avgSF36 - 2*sdSF36/sqrt(n),
                      ymax = avgSF36 + 2*sdSF36/sqrt(n)),
                  color = "red") +
  ggtitle("SF-36 Mental Health Score Trajectories (Random Intercept Model)")
```



Question 11:

Calculate the autocorrelation matrix and autocorrelation function for these simulated data. (10 pts)

Calculate covariance matrix for the Y

```
# Convert Y_long data frame into a wide dataframe
Y_wide <- Y_long %>%
  select(id, time_point, SF36_score) %>%
  pivot_wider(names_from = time_point,
              values_from = SF36_score)

# Calculate covariance matrix
cov_mat <- Y_wide %>%
  select(-id) %>%
  cov()

cov_mat[upper.tri(cov_mat)] = NA
cov_mat %>%
  as.data.frame()
```

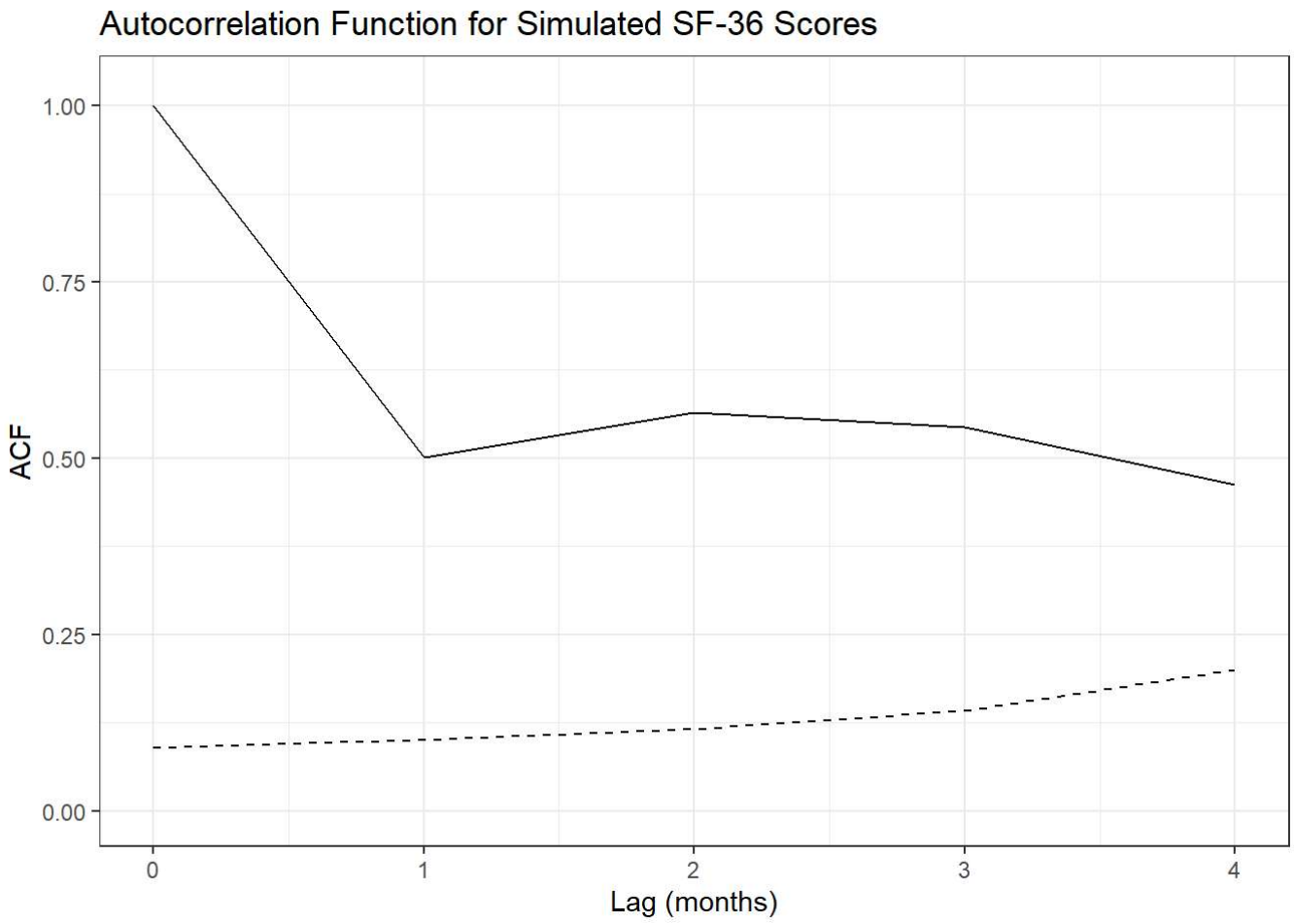
```
##           y0       y1        y2       y3       y4
## y0 198.07054       NA        NA       NA       NA
## y1 121.02617 270.1706        NA       NA       NA
## y2 120.07482 125.2115 207.13596       NA       NA
## y3  96.64017 136.2157  95.74171 211.8508       NA
## y4 105.62905 151.9266 130.89263 116.3550 254.7693
```

Calculate autocorrelation of the residuals

```
# The autocor command requires you to give: the residual, time variable, id variable
fit = gls(SF36_score~as.factor(time_numeric),Y_long)
acf1 = ACF(fit,form= ~1|id)
acf1
```

```
##   lag       ACF
## 1   0 1.0000000
## 2   1 0.5016807
## 3   2 0.5650673
## 4   3 0.5441493
## 5   4 0.4624751
```

```
# generate the lower bound
acf1 = acf1 %>%
    mutate(upper = 2 * 1 / sqrt(100 * (5-lag)))

# Make the plot
acf1 %>%
  ggplot() +
  geom_line(aes(x=lag, y=ACF))  +
  geom_line(aes(x=lag, y=upper), linetype=2) +
  xlab("Lag (months)") +
  ylim(0,1.02) +
  theme_bw() +
  ggtitle("Autocorrelation Function for Simulated SF-36 Scores")
```



Question 12:

Using the SF-36 example, explain the scientific meaning of the means, random intercepts, variance of the random intercepts, residuals, and variance of the residuals. (10 pts)

The means represent the average trajectory of the SF-36 score across the entire population at that specific time lag.

The random intercepts represent the baseline value or starting point for each individual. In this case, its each individual person's SF-36 score at baseline.

The variance of the random intercepts tells us the degree of heterogeneity we anticipate around the mean function. In this case, how far we expect an individual's specific variance, in regards to their SF-36 score, to deviate from the mean.

The residuals for each individual observation represent the unexplained error within each subject over time.

The variance of the residuals measures the unexplained variability in the SF-36 scores after accounting for the individual and time-varying effects.

```
# R Session Information
options(width = 120)
sessioninfo::session_info()
```

```
## Warning in system2("quarto", "-V", stdout = TRUE, env = paste0("TMPDIR=", : running command
'"quarto"
## TMPDIR=C:/Users/anany/AppData/Local/Temp/RtmpExr0Gd/file11b4773e6fa6 -V' had status 1
```

```
## ─ Session info ─────────────────────────────────────────────────────────────
─────────────────────────
##  setting  value
##  version  R version 4.5.1 (2025-06-13 ucrt)
##  os       Windows 11 x64 (build 26100)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United States.utf8
##  ctype    English_United States.utf8
##  tz       America/New_York
##  date     2025-09-09
##  pandoc   3.4 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
##  quarto   NA @ C:\\PROGRA~1\\RStudio\\RESOUR~1\\app\\bin\\quarto\\bin\\quarto.exe
##
## ─ Packages ─────────────────────────────────────────────────────────────────
─────────────────────────
##  package      * version date (UTC) lib source
##  bslib          0.9.0   2025-01-30 [1] CRAN (R 4.5.0)
##  cachem         1.1.0   2024-05-16 [1] CRAN (R 4.5.0)
##  cli            3.6.5   2025-04-23 [1] CRAN (R 4.5.0)
##  crayon         1.5.3   2024-06-20 [1] CRAN (R 4.5.0)
##  digest         0.6.37  2024-08-19 [1] CRAN (R 4.5.0)
##  dplyr        * 1.1.4   2023-11-17 [1] CRAN (R 4.5.0)
##  evaluate       1.0.4   2025-06-18 [1] CRAN (R 4.5.0)
##  farver         2.1.2   2024-05-13 [1] CRAN (R 4.5.0)
##  fastmap        1.2.0   2024-05-15 [1] CRAN (R 4.5.0)
##  forcats      * 1.0.0   2023-01-29 [1] CRAN (R 4.5.1)
##  generics       0.1.4   2025-05-09 [1] CRAN (R 4.5.0)
##  GGally       * 2.4.0   2025-08-23 [1] CRAN (R 4.5.1)
##  ggplot2      * 3.5.2   2025-04-09 [1] CRAN (R 4.5.0)
##  ggstats        0.10.0  2025-07-02 [1] CRAN (R 4.5.1)
##  glue           1.8.0   2024-09-30 [1] CRAN (R 4.5.0)
##  gtable         0.3.6   2024-10-25 [1] CRAN (R 4.5.0)
##  hms            1.1.3   2023-03-21 [1] CRAN (R 4.5.0)
##  htmltools      0.5.8.1 2024-04-04 [1] CRAN (R 4.5.0)
##  joineR       * 1.2.8   2023-01-22 [1] CRAN (R 4.5.1)
##  jquerylib      0.1.4   2021-04-26 [1] CRAN (R 4.5.0)
##  jsonlite       2.0.0   2025-03-27 [1] CRAN (R 4.5.0)
##  knitr          1.50    2025-03-16 [1] CRAN (R 4.5.0)
##  labeling       0.4.3   2023-08-29 [1] CRAN (R 4.5.0)
##  lattice        0.22-7  2025-04-02 [2] CRAN (R 4.5.1)
##  lifecycle      1.0.4   2023-11-07 [1] CRAN (R 4.5.0)
##  lubridate    * 1.9.4   2024-12-08 [1] CRAN (R 4.5.1)
##  magrittr       2.0.3   2022-03-30 [1] CRAN (R 4.5.0)
##  Matrix         1.7-3   2025-03-11 [2] CRAN (R 4.5.1)
##  mvtnorm      * 1.3-3   2025-01-10 [1] CRAN (R 4.5.1)
##  nlme         * 3.1-168 2025-03-31 [2] CRAN (R 4.5.1)
##  pillar         1.10.2  2025-04-05 [1] CRAN (R 4.5.0)
##  pkgconfig      2.0.3   2019-09-22 [1] CRAN (R 4.5.0)
##  purrr        * 1.0.4   2025-02-05 [1] CRAN (R 4.5.0)
##  R6             2.6.1   2025-02-15 [1] CRAN (R 4.5.0)
```

```
##  RColorBrewer   1.1-3   2022-04-03 [1] CRAN (R 4.5.0)
##  readr        * 2.1.5   2024-01-10 [1] CRAN (R 4.5.1)
##  rlang          1.1.6   2025-04-11 [1] CRAN (R 4.5.0)
##  rmarkdown      2.29    2024-11-04 [1] CRAN (R 4.5.0)
##  rstudioapi     0.17.1  2024-10-22 [1] CRAN (R 4.5.0)
##  S7             0.2.0   2024-11-07 [1] CRAN (R 4.5.1)
##  sass           0.4.10  2025-04-11 [1] CRAN (R 4.5.0)
##  scales         1.4.0   2025-04-24 [1] CRAN (R 4.5.0)
##  sessioninfo    1.2.3   2025-02-05 [1] CRAN (R 4.5.0)
##  statmod        1.5.0   2023-01-06 [1] CRAN (R 4.5.1)
##  stringi        1.8.7   2025-03-27 [1] CRAN (R 4.5.0)
##  stringr      * 1.5.1   2023-11-14 [1] CRAN (R 4.5.0)
##  survival     * 3.8-3   2024-12-17 [2] CRAN (R 4.5.1)
##  tibble       * 3.3.0   2025-06-08 [1] CRAN (R 4.5.0)
##  tidyr        * 1.3.1   2024-01-24 [1] CRAN (R 4.5.0)
##  tidyselect     1.2.1   2024-03-11 [1] CRAN (R 4.5.0)
##  tidyverse    * 2.0.0   2023-02-22 [1] CRAN (R 4.5.1)
##  timechange     0.3.0   2024-01-18 [1] CRAN (R 4.5.1)
##  tzdb           0.5.0   2025-03-15 [1] CRAN (R 4.5.1)
##  utf8           1.2.6   2025-06-08 [1] CRAN (R 4.5.0)
##  vctrs          0.6.5   2023-12-01 [1] CRAN (R 4.5.0)
##  withr          3.0.2   2024-10-28 [1] CRAN (R 4.5.0)
##  xfun           0.52    2025-04-02 [1] CRAN (R 4.5.0)
##  yaml           2.3.10  2024-07-26 [1] CRAN (R 4.5.0)
##
##  [1] C:/Users/anany/AppData/Local/R/win-library/4.5
##  [2] C:/Program Files/R/R-4.5.1/library
##  * ── Packages attached to the search path.
##
## ──────────────────────────────────────────────────────────────────────────────
```