1. **Question Format and Score: Crossover Design**. Three weeks ago you took your midterm exam. Unbeknownst to you, the exam was also serving as a crossover experiment. There were four versions (A, B, C, D) where versions A/C shared one structure and versions B/D shared another. The treatment of interest was question *format*: whether a question was presented as a single compound sentence or as separate sentences/bullet points. Questions 21 and 23 each had a compound and a separated version; the two structures (A/C vs B/D) ensured that if Q21 was compound then Q23 was separated, and vice versa.

   Exam versions were randomly assigned to students in blocks by session (morning or afternoon).

   a. What are the experimental units in this study? What are the measurement units?
   b. What is the treatment? How many levels does it have?
   c. Why is this a crossover design rather than a simple completely randomized design? What is the advantage of each student receiving both formats?
   d. What is the blocking factor, and why was blocking used?

Part a.
The experimental units are the individual students. Each student is assigned to one exam version, which determines the format of each question. The measurement units are also the individual students, since scores are recorded at the student level (one score per question per student).

Part b.
The treatment is the question format with two levels: compound (a single sentence combining several sub-questions) and separated (each sub-question as a separate sentence or bullet point).

Part c.
This is a crossover design because every student is exposed to *both* treatment levels: each student answers one question in compound format and one question in separated format. The advantage is that each student serves as their own control. This eliminates between-student variability from the treatment effect estimate, since the comparison of compound vs. separated is made *within* each student.

Part d.
The blocking factor is the exam session (morning vs afternoon). Blocking was used because students in different sessions might differ systematically (e.g., the morning group might include students with different preparation habits or alertness levels). Randomly assigning exam versions within each session ensures that version assignment is balanced with respect to session, preventing the session from confounding the treatment comparison.

2. **Coffee and Reaction Time: Crossover Design**. A researcher wants to know whether drinking coffee before a cognitive test improves reaction time. She recruits 20 subjects and uses a crossover design: each subject takes the cognitive test twice, once after drinking coffee and once after drinking decaf (a placebo). The order (coffee-first vs decaf-first) is randomly assigned, with a one-week washout period between sessions.

   a. What is the primary advantage of this crossover design compared to a completely randomized design that assigns 10 subjects to coffee and 10 to decaf?

b. The researcher is concerned about a *carryover effect*: caffeine from the first session might still affect performance in the second session even after a week. Explain why it is a threat to this crossover design.
c. If carryover effects are present and asymmetric (i.e., the carryover from coffee to decaf is different from the carryover from decaf to coffee), explain why the crossover estimator $\bar{Z}$ (the average of the observed differences) from the previous problem would be biased.
d. One way to test for carryover is to compare the *total* response (session 1 + session 2) between the two sequence groups. Explain the logic behind this test: under no carryover, why should the sequence groups have the same expected total?

Part a.
The primary advantage is that each subject serves as their own control. By measuring both coffee and decaf performance on the same individual, we eliminate between-subject variability (differences in baseline reaction time, cognitive ability, sleep habits, etc.) from the treatment comparison. This typically yields a much more precise estimate of the treatment effect than a between-subjects design with the same number of subjects.

Part b.
A carryover effect occurs when the treatment received in one period continues to affect the response in a subsequent period, even after a washout. In this case, residual caffeine or a caffeine-induced physiological change from the first session might persist into the second session. For example, if a subject drinks coffee in session 1, they might still have elevated alertness in session 2 even though they receive decaf. This threatens the design because the second-period response no longer reflects just the second-period treatment—it is contaminated by the first-period treatment.

Part c.
The derivation of $\bar{Z}$ as an unbiased estimator assumed that each subject's potential outcome under a treatment in a given slot depends only on that treatment and that slot, not on what treatment was given in the other slot. When carryover effects are present and asymmetric, the potential outcomes in period 2 depend on the treatment received in period 1 in a way that differs between the two sequences. The sign-flipping in $\bar{Z}$ cancels out symmetric nuisance effects (like period effects), but asymmetric carryover adds different biases to the two sequence groups that do not cancel. In this case, we'd expect the carryover effect to mask the difference between coffee and decaf in one sequence, leading to an underestimate of the true treatment effect.

Part d.
Under no carryover, a subject's response in each session depends only on the treatment received in that session (and possibly a period effect). The total response $T_i = R_{i1} + R_{i2}$ for a subject thus equals:

- Coffee-first ($D_i = 1$): $T_i = Y_i(\text{coffee}, 1) + Y_i(\text{decaf}, 2)$
- Decaf-first ($D_i = 0$): $T_i = Y_i(\text{decaf}, 1) + Y_i(\text{coffee}, 2)$

Under no carryover, both sequences produce totals that are functions of the same set of potential outcomes (just evaluated at different periods). If we further assume no treatment-by-period interaction, the expected total is the same for both sequences. A significant difference in totals between the two sequence groups therefore suggests the presence of carryover effects.

3. **Coffee and Reaction Time: Analysis**. The researcher from the previous problem ran her crossover experiment and collected data. You can generate a synthetic version of her dataset with the following code.

```
library(tidyverse)
set.seed(40)
n <- 20
coffee <- tibble(
    subject  = 1:n,
    sequence = rep(c("coffee_first", "decaf_first"), each = n / 2),
    subject_ability = rnorm(n, mean = 250, sd = 30)
) |>
    mutate(
        period1 = case_when(
            sequence == "coffee_first" ~ subject_ability - 8 + rnorm(n, 0, 10),
            sequence == "decaf_first"  ~ subject_ability + rnorm(n, 0, 10)
        ),
        period2 = case_when(
            sequence == "coffee_first" ~ subject_ability + 5 + rnorm(n, 0, 10),
            sequence == "decaf_first"  ~ subject_ability - 8 + 5 + rnorm(n, 0, 10)
        )
    ) |>
    select(subject, sequence, period1, period2)
```

Here, `period1` and `period2` are reaction times (in milliseconds) on the cognitive test in session 1 and session 2 respectively. We'll denote these Y_{i1} and Y_{i2} for subject i. Lower is better. The data-generating process includes a true treatment effect of coffee (coffee lowers reaction time by 8 ms) and a period effect (reaction time increases by 5 ms in period 2, perhaps due to fatigue or reduced novelty). There is no carryover effect.

a. For each subject, compute Z_i, the within-subject difference in reaction time (decaf $-$ coffee). Be careful: which period corresponds to the coffee score depends on the subject's sequence. Add this column to the data frame and compute $\bar{Z}$.

b. Conduct a randomization test of the sharp null hypothesis that coffee has no effect on any subject's reaction time.

   i. State the null hypothesis in terms of potential outcomes.
   ii. Simulate 1,000 test statistics under the null by re-randomizing the sequence assignment (a complete randomization of 10 to each sequence).
   iii. Plot the null distribution with the observed statistic and report the two-sided p-value.

c. Reshape the data into long format (one row per subject per period) and fit a linear model with fixed effects for subject:

$$Y_{ij} = \beta_0 + \beta_1 \text{treatment}_{ij} + \beta_2 \text{period}_{ij} + \alpha_i + \varepsilon_{ij}$$

where α_i is a fixed effect for subject i. Report and interpret $\hat{\beta}_1$. Compare this with the result from the randomization test.

d. Now fit a simpler model that omits the period effect: $Y_{ij} = \beta_0 + \beta_1 \text{treatment}_{ij} + \alpha_i + \varepsilon_{ij}$. How does the estimate of the treatment effect change? Explain why the crossover design protects the treatment effect estimate from period effects even when period is not included in the model.

Part a.
For coffee-first subjects: period 1 is coffee, period 2 is decaf, so $Z_i = Y_{i2} - Y_{i1}$ (decaf − coffee).

For decaf-first subjects: period 1 is decaf, period 2 is coffee, so $Z_i = Y_{i1} - Y_{i2}$ (decaf − coffee).

```
library(tidyverse)
set.seed(40)
n <- 20
coffee <- tibble(
    subject  = 1:n,
    sequence = rep(c("coffee_first", "decaf_first"), each = n / 2),
    subject_ability = rnorm(n, mean = 250, sd = 30)
) |>
    mutate(
        period1 = case_when(
            sequence == "coffee_first" ~ subject_ability - 8 + rnorm(n, 0, 10),
            sequence == "decaf_first"  ~ subject_ability + rnorm(n, 0, 10)
        ),
        period2 = case_when(
            sequence == "coffee_first" ~ subject_ability + 5 + rnorm(n, 0, 10),
            sequence == "decaf_first"  ~ subject_ability - 8 + 5 + rnorm(n, 0, 10)
        )
    ) |>
    select(subject, sequence, period1, period2)

coffee <- coffee |>
    mutate(Z = ifelse(sequence == "coffee_first",
                      period2 - period1,   # decaf - coffee
                      period1 - period2))  # decaf - coffee

obs_stat <- mean(coffee$Z)
obs_stat
```

```
[1] 12.1976
```

The observed $\bar{Z}$ is the average within-subject difference in reaction time (decaf minus coffee). A positive value indicates that reaction time was higher (slower) under decaf, consistent with a beneficial effect of coffee.

Part b.

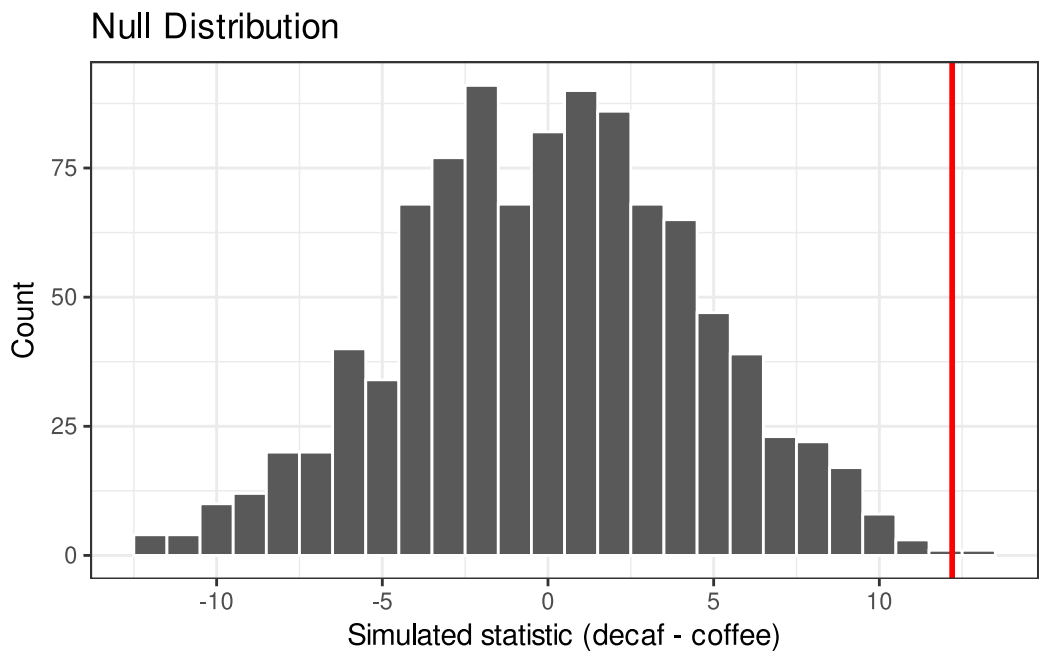
i. The sharp null hypothesis is:

$$H_0 : Y_i(\text{coffee}, k) = Y_i(\text{decaf}, k) \quad \text{for all subjects } \ i \ \text{ and periods } \ k = 1, 2$$

That is, for every subject, the reaction time on the cognitive test is the same regardless of whether they drank coffee or decaf.

ii. & iii.

```
sim_stat <- function(df) {
    df2 <- df |>
        mutate(sequence = sample(sequence)) |>
        mutate(Zsim = ifelse(sequence == "coffee_first",
                             period2 - period1,
                             period1 - period2))
    mean(df2$Zsim)
}

set.seed(40)
null_stats <- replicate(1000, sim_stat(coffee))

ggplot(data.frame(stat = null_stats), aes(x = stat)) +
    geom_histogram(binwidth = 1, color = "white") +
    geom_vline(xintercept = obs_stat, color = "red", linewidth = 1) +
    labs(title = "Null Distribution",
         x = "Simulated statistic (decaf - coffee)",
         y = "Count") +
    theme_bw()
```




```
p_val <- mean(abs(null_stats) >= abs(obs_stat))
p_val
```

```
[1] 0.002
```

Part c.

```
coffee_long <- coffee |>
    pivot_longer(cols = c(period1, period2),
                 names_to = "period",
                 values_to = "rt") |>
    mutate(
        period = ifelse(period == "period1", "1", "2"),
        treatment = case_when(
            sequence == "coffee_first" & period == "1" ~ "coffee",
            sequence == "coffee_first" & period == "2" ~ "decaf",
            sequence == "decaf_first"  & period == "1" ~ "decaf",
            sequence == "decaf_first"  & period == "2" ~ "coffee"
        ),
        subject = as.factor(subject)
    )

m1 <- lm(rt ~ treatment + period + subject, data = coffee_long)
summary(m1)
```

```
Call:
lm(formula = rt ~ treatment + period + subject, data = coffee_long)

Residuals:
    Min      1Q  Median      3Q     Max 
-13.294  -6.631   0.000   6.631  13.294 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)    256.319112   8.591052  29.836  < 2e-16 ***
treatmentdecaf  12.197603   3.663237   3.330 0.003728 ** 
period2         -0.006956   3.663237  -0.002 0.998506    
subject2         2.018489  11.584172   0.174 0.863617    
subject3       -34.407444  11.584172  -2.970 0.008199 ** 
subject4       -42.220912  11.584172  -3.645 0.001853 ** 
subject5       -34.714917  11.584172  -2.997 0.007740 ** 
subject6       -52.509702  11.584172  -4.533 0.000258 ***
subject7       -64.702679  11.584172  -5.585 2.66e-05 ***
subject8        36.793479  11.584172   3.176 0.005229 ** 
subject9       -21.310679  11.584172  -1.840 0.082374 .  
subject10      -59.216903  11.584172  -5.112 7.29e-05 ***
subject11      -18.477684  11.584172  -1.595 0.128102    
subject12      -40.682458  11.584172  -3.512 0.002490 ** 
subject13        4.664708  11.584172   0.403 0.691926    
subject14      -36.958587  11.584172  -3.190 0.005068 ** 
subject15        0.375149  11.584172   0.032 0.974522    
subject16       20.561416  11.584172   1.775 0.092817 .  
subject17        5.597910  11.584172   0.483 0.634753    
subject18        6.347388  11.584172   0.548 0.590467    
subject19       49.348556  11.584172   4.260 0.000471 ***
subject20      -41.069023  11.584172  -3.545 0.002312 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.58 on 18 degrees of freedom
Multiple R-squared:  0.9432,    Adjusted R-squared:  0.8768 
F-statistic: 14.22 on 21 and 18 DF,  p-value: 2.538e-07
```

The coefficient on `treatmentdecaf` (relative to coffee) estimates the average increase in reaction time when switching from coffee to decaf, adjusting for period and subject effects. This should be close to 8 ms (the true treatment effect in the data-generating process). Including subject as a fixed effect absorbs between-subject variability, making this equivalent to a within-subject analysis. The model-based result should align with the randomization test.

Part d.

```
m2 <- lm(rt ~ treatment + subject, data = coffee_long)
summary(m2)
```

```

Call:
lm(formula = rt ~ treatment + subject, data = coffee_long)

Residuals:
    Min      1Q  Median      3Q     Max 
-13.298  -6.629   0.000   6.629  13.298 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)    256.3156     8.1697  31.374  < 2e-16 ***
treatmentdecaf  12.1976     3.5655   3.421 0.002865 ** 
subject2         2.0185    11.2752   0.179 0.859817    
subject3       -34.4074    11.2752  -3.052 0.006566 ** 
subject4       -42.2209    11.2752  -3.745 0.001373 ** 
subject5       -34.7149    11.2752  -3.079 0.006179 ** 
subject6       -52.5097    11.2752  -4.657 0.000172 ***
subject7       -64.7027    11.2752  -5.738 1.57e-05 ***
subject8        36.7935    11.2752   3.263 0.004090 ** 
subject9       -21.3107    11.2752  -1.890 0.074106 .  
subject10      -59.2169    11.2752  -5.252 4.54e-05 ***
subject11      -18.4777    11.2752  -1.639 0.117711    
subject12      -40.6825    11.2752  -3.608 0.001873 ** 
subject13        4.6647    11.2752   0.414 0.683716    
subject14      -36.9586    11.2752  -3.278 0.003958 ** 
subject15        0.3751    11.2752   0.033 0.973805    
subject16       20.5614    11.2752   1.824 0.083985 .  
subject17        5.5979    11.2752   0.496 0.625251    
subject18        6.3474    11.2752   0.563 0.580051    
subject19       49.3486    11.2752   4.377 0.000324 ***
subject20      -41.0690    11.2752  -3.642 0.001733 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.28 on 19 degrees of freedom
Multiple R-squared:  0.9432,    Adjusted R-squared:  0.8833 
F-statistic: 15.76 on 20 and 19 DF,  p-value: 6.195e-08
```

The estimate of the treatment effect is nearly identical whether or not period is included in the model. This is because the crossover design ensures that the treatment effect estimate is based on *within-subject*

comparisons, and the balanced assignment of sequences ensures that the period effect cancels out. In the coffee-first group, the within-subject difference $Z_i = Y_{i2} - Y_{i1}$ confounds treatment and period, but in the decaf-first group the confounding goes in the opposite direction. Averaging across both groups cancels the period effect, making the treatment estimate robust even without explicitly modeling period. With subject fixed effects in the model, the treatment coefficient is identified entirely from within-subject variation, which is exactly what $\bar{Z}$ captures.

4. **Sequential Probability Ratio Test: Tinkering with Parameters**. The SPRT is a powerful tool, but its performance depends on the choice of parameters. In this exercise, we will explore how changing the parameters p_0, p_1, α, and β affects the behavior of the test that was shown in class[1].

   a. Using the same simulation of a single run that was shown in class, lower both the error rates (α and β) to represent a more stringent test. Plot the same simulated run with the new thresholds. How does the decision process change with the new thresholds? Does it take more or fewer samples to reach a decision?

   b. Choose a fixed value for p_0 (e.g., 0.5) and vary p_1 (e.g., 0.6, 0.7, 0.8). For each value of p_1, use a full simulation to calculate the expected number of samples needed to reach a decision under both hypotheses.

   c. Now, fix p_1 (e.g., 0.7) and vary p_0 (e.g., 0.5, 0.4, 0.3). Again, calculate the expected number of samples needed for each scenario.

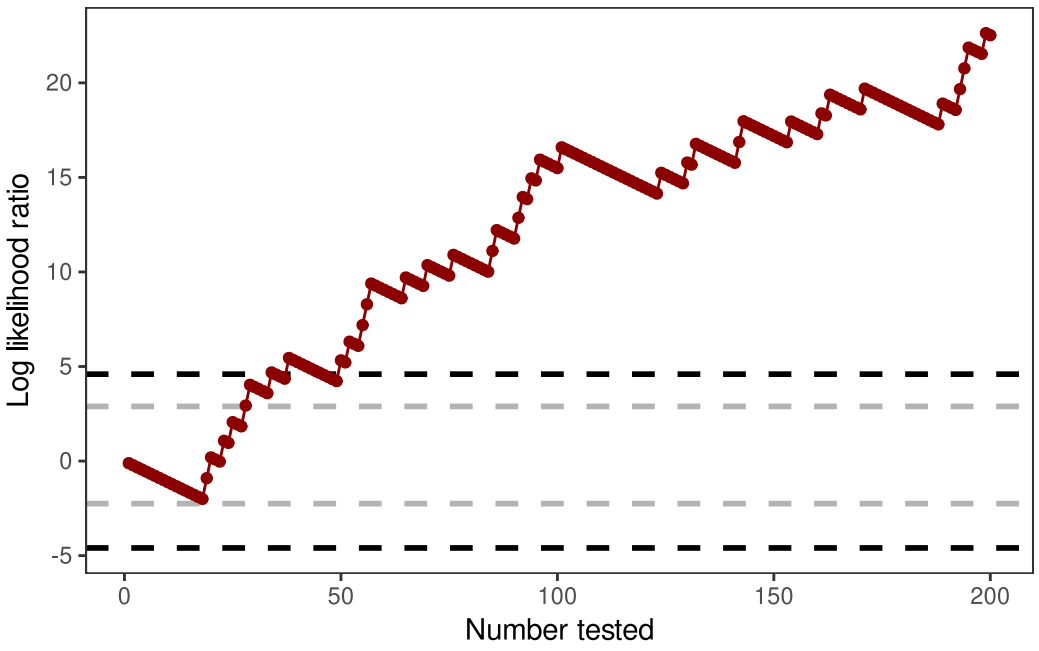
   d. Summarize your findings. How do the choices of p_0 and p_1 affect the efficiency of the SPRT? What trade-offs do you observe when adjusting the error rates α and β?

Part a.
To represent a more stringent test, we will lower the error rates from the class defaults ($\alpha = 0.05, \beta = 0.10$) to $\alpha = 0.01$ and $\beta = 0.01$.

```
library(tidyverse)

# Original parameters from class
p0 <- 0.05
p1 <- 0.15
n_max <- 200

# Original Thresholds
alpha_orig <- 0.05
beta_orig <- 0.10
logA_orig <- log((1 - beta_orig) / alpha_orig)
logB_orig <- log(beta_orig / (1 - alpha_orig))

# New Stringent Thresholds
alpha_new <- 0.01
beta_new <- 0.01
logA_new <- log((1 - beta_new) / alpha_new)
logB_new <- log(beta_new / (1 - alpha_new))

# Simulate the same single run
set.seed(3405)
y <- rbinom(n_max, size = 1, prob = p1)
```

[1] See the slides for Sequential Analysis I for code to simulate the performance of the SPRT.

```
log_lr <- cumsum(y * log(p1 / p0) + (1 - y) * log((1 - p1) / (1 - p0)))

df_path <- tibble(n = 1:n_max, log_lr = log_lr)

ggplot(df_path, aes(x = n, y = log_lr)) +
   geom_hline(yintercept = c(logA_orig, logB_orig), linetype = "dashed", color =
"gray70", linewidth = 1) +
  geom_hline(yintercept = c(logA_new, logB_new), linetype = "dashed", color = "black",
linewidth = 1) +
  geom_line(color = "darkred") +
  geom_point(color = "darkred") +
  labs(x = "Number tested",
       y = "Log likelihood ratio") +
  theme_bw() +
  theme(panel.grid = element_blank())
```



The decision process requires more evidence to cross the thresholds. The original thresholds (gray dashed lines) are narrower, meaning the test would have stopped and rejected the null hypothesis earlier. The new stringent thresholds (black dashed lines) are wider, so the log-likelihood ratio path must travel further. Consequently, it takes strictly more samples to reach a decision when we demand lower error rates.

Part b.

```
library(purrr)

sprt_sim <- function(p_true, p0, p1, alpha = 0.05, beta = 0.10, n_max = 1000) {
```

```
  logA <- log((1 - beta) / alpha)
  logB <- log(beta / (1 - alpha))

  x <- rbinom(n_max, size = 1, prob = p_true)
  log_lr <- cumsum(x * log(p1 / p0) + (1 - x) * log((1 - p1) / (1 - p0)))

  stop_reject <- which(log_lr >= logA)[1]
  stop_accept <- which(log_lr <= logB)[1]

  # Handle cases where boundaries aren't crossed within n_max
  if (is.na(stop_reject)) stop_reject <- Inf
  if (is.na(stop_accept)) stop_accept <- Inf

  n_used <- min(stop_reject, stop_accept, n_max)
  return(n_used)
}

set.seed(158)
n_sims <- 1000
p0_fixed <- 0.5
p1_vec <- c(0.6, 0.7, 0.8)

results_b <- map_dfr(p1_vec, function(p1_val) {
  # Simulate average samples needed under H0 (true prob is p0)
  n_h0 <- replicate(n_sims, sprt_sim(p_true = p0_fixed, p0 = p0_fixed, p1 = p1_val))
  # Simulate average samples needed under H1 (true prob is p1)
  n_h1 <- replicate(n_sims, sprt_sim(p_true = p1_val, p0 = p0_fixed, p1 = p1_val))

  tibble(
    p1 = p1_val,
    E_N_H0 = mean(n_h0),
    E_N_H1 = mean(n_h1)
  )
})

results_b
```

```
# A tibble: 3 × 3
     p1 E_N_H0 E_N_H1
  <dbl>  <dbl>  <dbl>
1   0.6  102.   123.
2   0.7   25.9   31.0
3   0.8   10.9   13.8
```

Part c.

```
set.seed(158)
p1_fixed <- 0.7
p0_vec <- c(0.5, 0.4, 0.3)

results_c <- map_dfr(p0_vec, function(p0_val) {
  n_h0 <- replicate(n_sims, sprt_sim(p_true = p0_val, p0 = p0_val, p1 = p1_fixed))
  n_h1 <- replicate(n_sims, sprt_sim(p_true = p1_fixed, p0 = p0_val, p1 = p1_fixed))

  tibble(
    p0 = p0_val,
    E_N_H0 = mean(n_h0),
    E_N_H1 = mean(n_h1)
  )
})

results_c
```

```
# A tibble: 3 × 3
     p0 E_N_H0 E_N_H1
  <dbl>  <dbl>  <dbl>
1   0.5  25.5   32.6
2   0.4  12.2   14.9
3   0.3   7.18   8.49
```

Part d.

The efficiency of the SPRT clearly depends very heavily on the choices of p_0 and p_1, in particular the distance between them. As the gap between p_0 and p_1 increases (e.g., from $\mid 0.6 - 0.5 \mid = 0.1$ to $\mid 0.8 - 0.5 \mid = 0.3$), the expected number of samples needed to reach a decision drops dramatically. A higher gap means that it is easier to distinguish between the null and the alternate scenarios, thus making the SPRT reject faster.

Adjusting the error rates alters the boundaries. Lowering α and β pushes the stopping boundaries $\log(A)$ and $\log(B)$ further away from zero. The direct trade-off is between confidence and cost: demanding a lower probability of making a Type I or Type II error requires more evidence to accumulate, which directly increases the expected sample size necessary to conclude the test.

5. **Adaptive Assignment: Implementing UCB**. A pharmaceutical company is running a clinical trial comparing two pain relief drugs. Each patient reports a pain relief score on a continuous scale from 0 to 1 (higher is better). The company wants to use an adaptive assignment strategy that balances learning which drug is better with assigning more patients to the better-performing drug.

   The following code generates potential outcomes for $N = 30$ subjects under both treatments.

```
library(tidyverse)
set.seed(42)
N <- 30
po <- tibble(
  subject = 1:N,
  Y_A = rbeta(N, 4, 6),
  Y_B = rbeta(N, 5, 5)
)
```

   Recall the UCB algorithm with Hoeffding's Inequality. The upper confidence bound for arm j at time t is:

$$U_j = \hat{\mu}_j + \sqrt{\frac{\ln(t)}{2\,n_j}}$$

   where $\hat{\mu}_j$ is the sample mean of responses observed so far for arm j, n_j is the number of subjects assigned to arm j so far, and t is the index of the current subject. At each step, the algorithm assigns the next subject to the arm with the highest U_j.

   a. Write code to implement the UCB algorithm on these potential outcomes. Initialize by assigning the first subject to arm A and the second to arm B, then use the UCB rule for subjects 3 through 30[2]. Store the results in a data frame with columns for the subject index, assignment, and observed response.

   b. Create a plot of the cumulative fraction of subjects assigned to arm B (n_B/t) as a function of the subject index t. Describe what you see: when is the algorithm exploring (assigning roughly evenly) and when is it exploiting (favoring one arm)?

   c. For a single time step of your choosing between $t = 5$ and $t = 10$, report the values of $\hat{\mu}_A$, $\hat{\mu}_B$, the addition to the mean (aka the Hoeffding bonus), and the resulting U_j. Verify that the algorithm's assignment matches the arm with the higher U_j. What role does the bonus term play when one arm has been sampled much less than the other?

   d. Compute the *total regret* of the UCB algorithm on these 30 subjects and compare it with the expected total regret of a balanced completely randomized design that assigns 15 subjects to each arm[3]. The per-subject regret is $\max(Y_i(A), Y_i(B)) - Y_i(d_i)$, where d_i is the arm actually assigned. Does the

[2]Start off by simply writing the code to calculate U_j after observing the first two subjects, and then assign the third subject accordingly. Next, wrap that logic in a loop to handle all subjects up to 30.

[3]To compute the expected total regret of a balanced CR design, you can simulate many random assignments of 15 subjects to each arm, compute the total regret for each simulation, and then take the average.

UCB algorithm achieve lower total regret than what we would expect had we used a balanced CR design?

Part a.

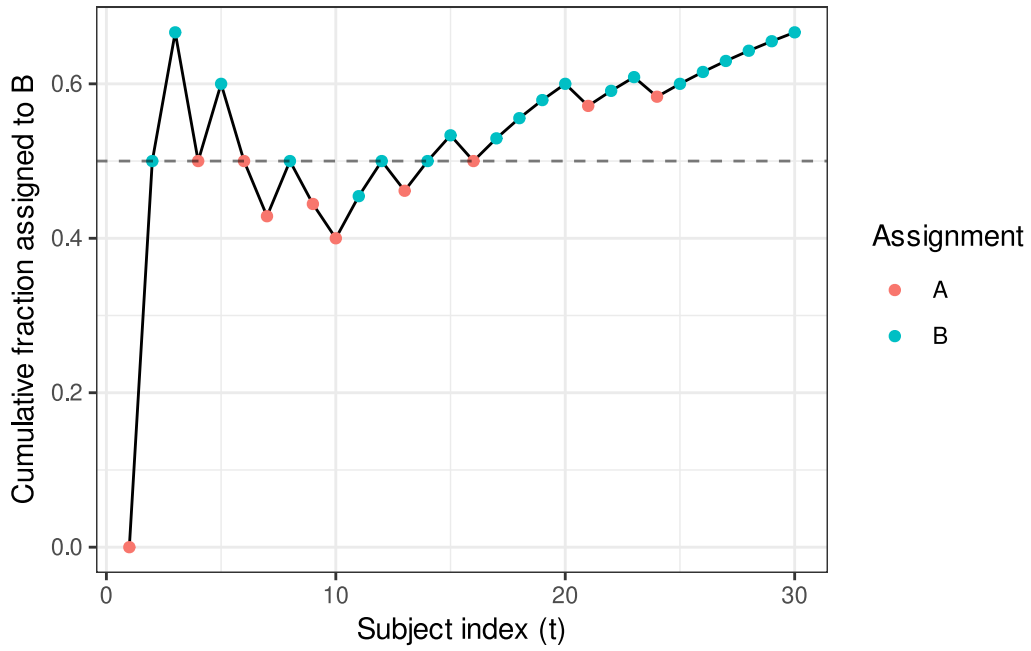
```
library(tidyverse)
set.seed(42)
N <- 30
po <- tibble(
  subject = 1:N,
  Y_A = rbeta(N, 4, 6),
  Y_B = rbeta(N, 5, 5)
)

assignments <- character(N)
responses   <- numeric(N)

# Initialize
assignments[1] <- "A"
responses[1]   <- po$Y_A[1]
assignments[2] <- "B"
responses[2]   <- po$Y_B[2]

# UCB loop
for (t in 3:N) {
  idx_A <- which(assignments[1:(t - 1)] == "A")
  idx_B <- which(assignments[1:(t - 1)] == "B")

  n_A    <- length(idx_A)
  n_B    <- length(idx_B)
  ybar_A <- mean(responses[idx_A])
  ybar_B <- mean(responses[idx_B])

  ucb_A <- ybar_A + sqrt(log(t) / (2 * n_A))
  ucb_B <- ybar_B + sqrt(log(t) / (2 * n_B))

  if (ucb_A >= ucb_B) {
    assignments[t] <- "A"
    responses[t]   <- po$Y_A[t]
  } else {
    assignments[t] <- "B"
    responses[t]   <- po$Y_B[t]
  }
}
```

```
trial <- tibble(
  subject    = 1:N,
  assignment = assignments,
  response   = responses
)

trial
```

```
# A tibble: 30 × 3
   subject assignment response
     <int> <chr>         <dbl>
 1       1 A             0.305
 2       2 B             0.348
 3       3 B             0.506
 4       4 A             0.473
 5       5 B             0.436
 6       6 A             0.693
 7       7 A             0.383
 8       8 B             0.257
 9       9 A             0.389
10      10 A             0.651
# ℹ 20 more rows
```

Part b.

```
trial |>
  mutate(frac_B = cumsum(assignment == "B") / subject) |>
  ggplot(aes(x = subject, y = frac_B)) +
  geom_line() +
  geom_point(aes(color = assignment)) +
  geom_hline(yintercept = 0.5, linetype = "dashed", alpha = 0.5) +
  labs(x = "Subject index (t)",
       y = "Cumulative fraction assigned to B",
       color = "Assignment") +
  theme_bw()
```



For roughly the first 15 subjects, the fraction assigned to B fluctuates around 0.5: the algorithm is exploring, alternating between arms as the UCBs trade places. After that, the fraction climbs steadily as the algorithm increasingly exploits arm B, which has accumulated a higher sample mean.

Part c.

```
# Example at t = 7 (after 6 subjects assigned)
t_check <- 7
idx_A <- which(assignments[1:(t_check - 1)] == "A")
idx_B <- which(assignments[1:(t_check - 1)] == "B")

n_A    <- length(idx_A)
n_B    <- length(idx_B)
ybar_A <- mean(responses[idx_A])
ybar_B <- mean(responses[idx_B])

bonus_A <- sqrt(log(t_check) / (2 * n_A))
bonus_B <- sqrt(log(t_check) / (2 * n_B))

tibble(
  arm     = c("A", "B"),
  n_k     = c(n_A, n_B),
  ybar    = round(c(ybar_A, ybar_B), 2),
  bonus   = round(c(bonus_A, bonus_B), 2),
  UCB     = round(c(ybar_A + bonus_A, ybar_B + bonus_B), 2)
)
```

```
# A tibble: 2 × 5
  arm     n_k  ybar bonus   UCB
  <chr> <int> <dbl> <dbl> <dbl>
1 A         3  0.49  0.57  1.06
2 B         3  0.43  0.57  1
```

The arm with the higher UCB matches the assignment the algorithm made at $t = 7$ (which can be verified by checking `trial$assignment[7]`). When one arm has been sampled fewer times, its bonus term $\sqrt{(b-a)^2 \ln(t)/(2n_k)}$ is larger, inflating its UCB. This encourages the algorithm to try the under-sampled arm, which is the mechanism that drives exploration.

Part d.

```
trial_full <- trial |>
  left_join(po, by = "subject") |>
  mutate(
    best     = pmax(Y_A, Y_B),
    regret   = best - response
  )

ucb_regret <- sum(trial_full$regret)

# Balanced CRD: simulate 5000 randomizations
set.seed(158)
balanced_regrets <- replicate(5000, {
  d <- sample(rep(c("A", "B"), each = N / 2))
  r <- ifelse(d == "A", po$Y_A, po$Y_B)
  sum(pmax(po$Y_A, po$Y_B) - r)
})

tibble(
  design              = c("UCB", "Balanced CRD (mean)"),
  total_regret        = round(c(ucb_regret, mean(balanced_regrets)), 2)
)
```

```
# A tibble: 2 × 2
  design              total_regret
  <chr>                      <dbl>
1 UCB                         2.7
2 Balanced CRD (mean)         2.83
```

The UCB algorithm achieves lower total regret than the average balanced randomization because it shifts patients to the better-performing arm, reducing the number of subjects assigned to the inferior treatment.