ST 440 Lab #1

Christian Nielsen

(1) Chapter 1, Problem 4

X_1 and X_2 have joint PMF

x_1	x_2	$\text{Prob}(X_1 = x_1, X_2 = x_2)$
0	0	0.15
1	0	0.15
2	0	0.15
0	1	0.15
1	1	0.20
2	1	0.20

(a) Compute the marginal distribution of X_1.

The marginal distribution is computed by summing over the other variable in the joint PMF.

x_1	$\text{Prob}(X_1 = x_1)$
0	0.3
1	0.35
2	0.35

(b) Compute the marginal distribution of X_2.

x_2	$\text{Prob}(X_2 = x_2)$
0	0.45
1	0.55

(c) Compute the conditional distribution of $X_1 \mid X_2$.

The general expression for the conditional distributions of $X_1 \mid X_2 = x_2$ is

$$f_{1|2}(x_1 \mid X_2 = x_2) = \frac{f(x_1, x_2)}{f_2(x_2)}.$$

So the conditional distribution of $X_1 \mid X_2$ in this case is

$$
\begin{aligned}
f_{1|2}(0 \mid X_2 = 0) &= \frac{f(0,0)}{f_2(0)} = \frac{0.15}{0.45} = 0.3333, \\
f_{1|2}(1 \mid X_2 = 0) &= \frac{f(1,0)}{f_2(0)} = \frac{0.15}{0.45} = 0.3333, \\
f_{1|2}(2 \mid X_2 = 0) &= \frac{f(2,0)}{f_2(0)} = \frac{0.15}{0.45} = 0.3333, \\
f_{1|2}(0 \mid X_2 = 1) &= \frac{f(0,1)}{f_2(1)} = \frac{0.15}{0.55} = 0.2727, \\
f_{1|2}(1 \mid X_2 = 1) &= \frac{f(1,1)}{f_2(1)} = \frac{0.2}{0.55} = .3636, \\
f_{1|2}(2 \mid X_2 = 1) &= \frac{f(2,1)}{f_2(1)} = \frac{0.2}{0.55} = 0.3636
\end{aligned}
$$

Under each condition ($X_2 = 0$ and $X_2 = 1$), the conditional probabilities sum to one, which is required by definition.

(d) Compute the conditional distribution of $X_2 \mid X_1$.

The general expression for the conditional distributions of $X_2 \mid X_1 = x_1$ is

$$
f_{2|1}(x_2 \mid X_1 = x_1) = \frac{f(x_1, x_2)}{f_1(x_1)}.
$$

So the conditional distribution of $X_2 \mid X_1$ in this case is

$$
\begin{aligned}
f_{2|1}(0 \mid X_1 = 0) &= \frac{f(0,0)}{f_1(0)} = \frac{0.15}{0.3} = 0.5, \\
f_{2|1}(1 \mid X_1 = 0) &= \frac{f(0,1)}{f_1(0)} = \frac{0.15}{0.3} = 0.5, \\
f_{2|1}(0 \mid X_1 = 1) &= \frac{f(1,0)}{f_1(1)} = \frac{0.15}{0.35} = 0.4286, \\
f_{2|1}(1 \mid X_1 = 1) &= \frac{f(1,1)}{f_1(1)} = \frac{0.2}{0.35} = 0.5714, \\
f_{2|1}(0 \mid X_1 = 2) &= \frac{f(2,0)}{f_1(2)} = \frac{0.15}{0.35} = 0.4286, \\
f_{2|1}(1 \mid X_1 = 2) &= \frac{f(2,1)}{f_1(2)} = \frac{0.2}{0.35} = 0.5714,
\end{aligned}
$$

Under each condition ($X_1 = 1, X_1 = 1$, and $X_1 = 2$), the conditional probabilities sum to one, which is required by definition.

(e) Are X_1 and X_2 independent? Justify your answer.

X_1 and X_2 are independent if and only if

$$
f(x_1, x_2) = f_1(x_1) f_2(x_2).
$$

We can check if this is true using an example from the tables above:

$$f(0,0) = 0.15 \neq 0.135 = (0.3)(0.45) = f_1(0)f_2(0)$$

This violates the definition for independence, so X_1 and X_2 are **dependent**.

(2) Chapter 1, Problem 6

Assume (X_1, X_2) have bivariate PDF

$$f(x_1, x_2) = \frac{1}{2\pi}(1 + x_1^2 + x_2^2)^{-3/2}$$

(a) Plot the conditional distribution of $X_1 \mid X_2 = x_2$ for $x_2 \in \{-3, -2, -1, 0, 1, 2, 3\}$ (preferably on the same plot).

We will do this using `R`, first we will make a function for the bivariate PDF

```
# Joint PDF
f <- function(x1, x2){
  (1/(2*pi)) * (1 + x1^2 + x2^2)^(-3/2)
}
```

Recall that

$$f_{1|2}(x_1 \mid X_2 = x_2) = \frac{f(x_1, x_2)}{f_2(x_2)}.$$

So in order to plot the conditional distribution of $X_1 \mid X_2 = x_2$, we must first calculate $f_2(x_2)$. To do this we must integrate the joint PDF over x_1

$$\begin{aligned} f_2(x_2) &= \int_{-\infty}^{\infty} \frac{1}{2\pi}(1 + x_1^2 + x_2^2)^{-3/2} dx_1 \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} (u)^{-3/2} dx_1 \\ &= \vdots \\ &= \frac{1}{\pi}(x_2^2 + 1)^{-1}. \end{aligned}$$

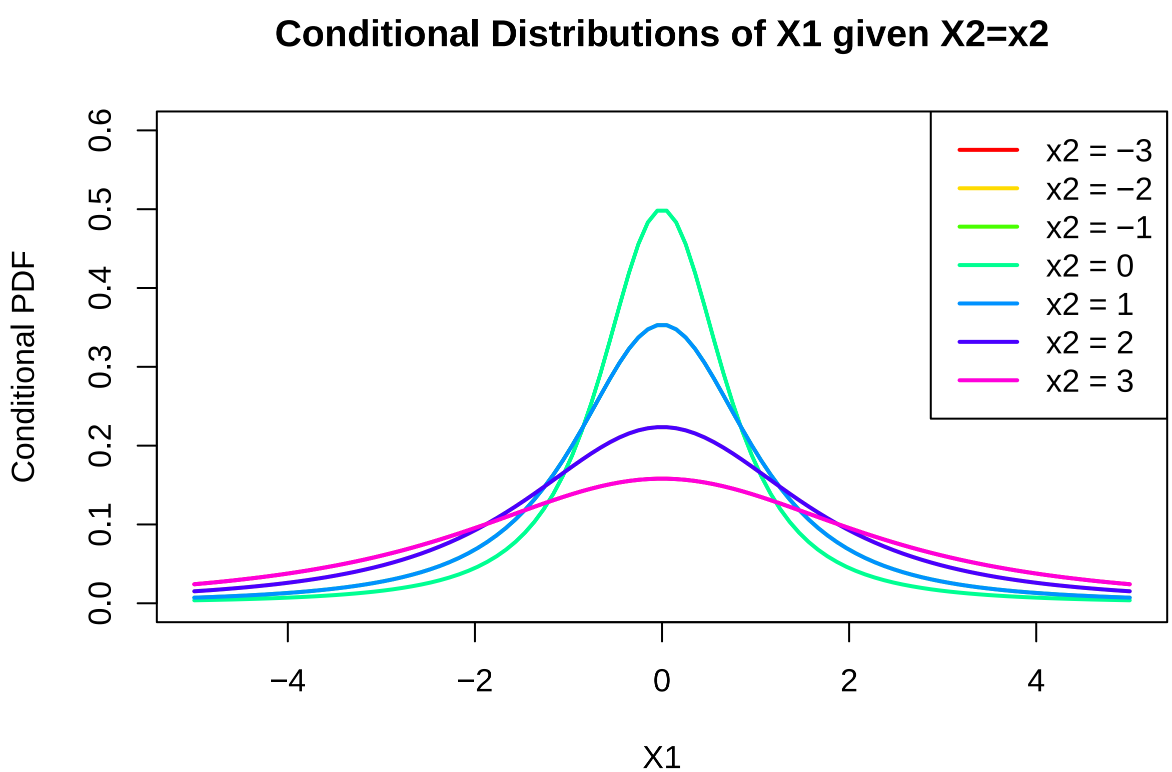
Similarly (this will be helpful when checking independence),

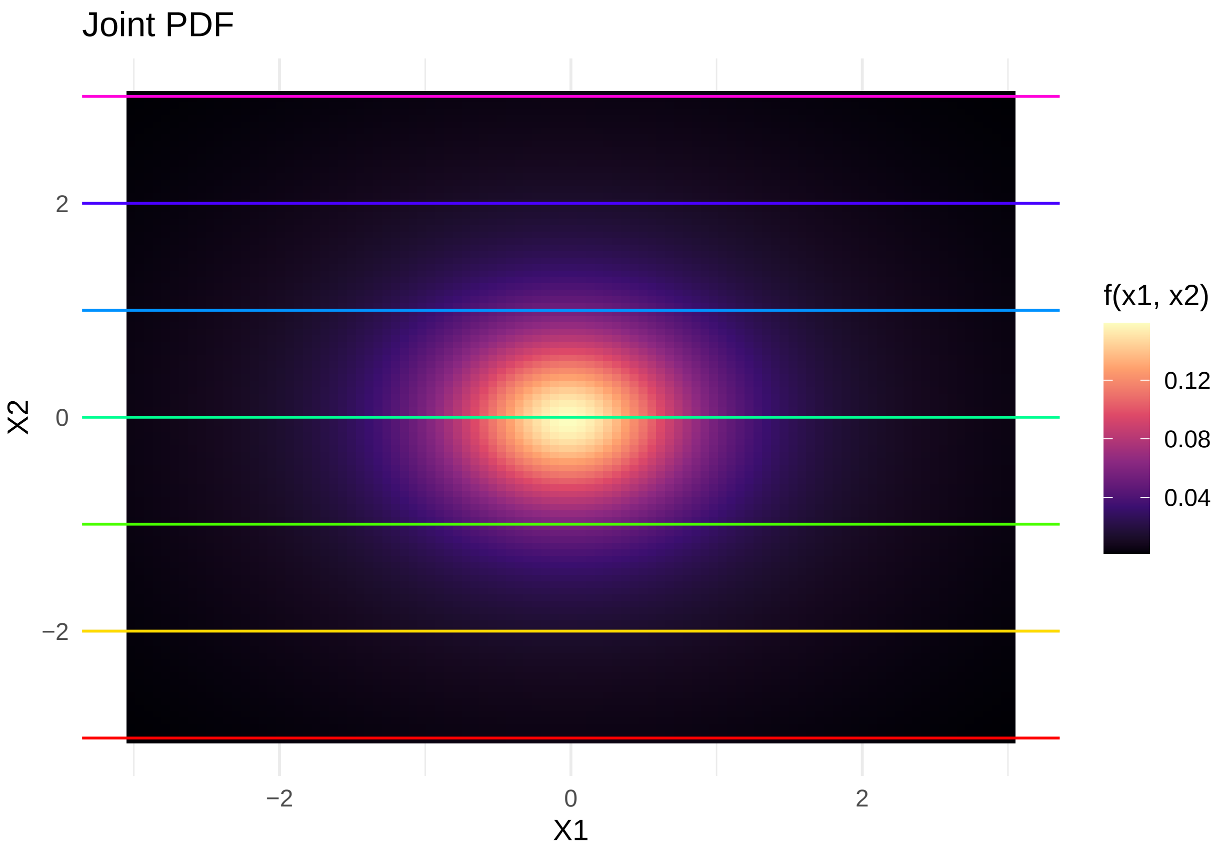
$$\begin{aligned} f_1(x_1) &= \int_{-\infty}^{\infty} \frac{1}{2\pi}(1 + x_2^2 + x_1^2)^{-3/2} dx_2 \\ &= \vdots \\ &= \frac{1}{\pi}(x_1^2 + 1)^{-1}. \end{aligned}$$

```r
# Define the marginal PDFs f_1(x1) and f_2(x2) by integrating the joint PDF over
# x2 and x1 respectively
f_1 <- function(x1) {
  (1/pi) * (x1^2 + 1)^-1
}

f_2 <- function(x2) {
  (1/pi) * (x2^2 + 1)^-1
}

# Define the conditional PDF of X1 given X2
f_conditional <- function(x1, x2){
  f(x1, x2)/f_2(x2)
}
```





(b) Do X_1 and X_2 appear to be correlated? Justify your answer.

$$\mathbb{E}[X_1] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 f(x_1, x_2) dx_1 dx_2 = 0,$$

$$\mathbb{E}[X_2] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_2 f(x_1, x_2) dx_2 dx_1 = 0,$$

and

$$\mathbb{E}[X_1 X_2] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 f(x_1, x_2) dx_1 dx_2 = 0,$$

so

$$\text{Cov}[X_1, X_2] = \mathbb{E}[X_1 X_2] - \mathbb{E}[X_1]\mathbb{E}[X_2] = 0.$$

Since $\text{Cov}[X_1, X_2] = 0$, X_1 and X_2 are **uncorrelated**.

(c) Do X_1 and X_2 appear to be independent? Justify your answer.

```r
marg_X1_vals <- sapply(x1_vals, f_1)
marg_X2_vals <- sapply(x2_vals, f_2)

# Check if f_1(x1) * f_2(x2) equals the joint PDF
check_independence <- sapply(1:length(x1_vals), function(i) {
  prod_vals <- f_1(x1_vals[i]) * f_2(x2_vals[i])
  joint_val <- f_conditional(x1_vals[i], x2_vals[i])
  return(abs(prod_vals - joint_val) < 1e-6)  # tolerance for numerical error
})

all(check_independence)  # TRUE if independent, FALSE otherwise
```

```
## [1] FALSE
```

The random variables X_1 and X_2 are **dependent** since the product of the marginal distributions does not equal the joint PDF. We can also see in the conditional distribution of $X_1 \mid X_2 = x_2$ plot that the spread of the bell curve **depends** on the value of x_2.

(3) Chapter 1, Problem 9

For this problem pretend we are dealing with a language with a six-word dictionary

$$\{\text{fun, sun, sit, sat, fan, for}\}.$$

An extensive study of literature written in this language reveals that all words are equally likely except that "for" is α times as likely as the other words. Further study reveals that:

i. Each keystroke is an error with probability θ.
ii. All letters are equally likely to produce errors.
iii. Given that a letter is typed incorrectly it is equally likely to be any other letter.
iv. Errors are independent across letters.

For example, the probability of correctly typing "fun" (or any other word) is $(1-\theta)^3$, the probability of typing "pun" or "fon" when intending to type is "fun" is $\theta(1-\theta)^2$, and the probability of typing "foo" or "nnn" when intending to type "fun" is $\theta^2(1-\theta)$. Use Bayes' rule to develop a simple spell checker for this language.

```r
spell_check <- function(typed_word, alpha, theta) {
  # Define the dictionary
  dictionary <- c("fun", "sun", "sit", "sat", "fan", "for")

  # Calculate the prior probabilities
  denom <- 5 + alpha
  Prob_for <- alpha / denom
  Prob_other <- 1 / denom

  # Define a function to calculate the likelihood P(t | w)
  # t is the typed word, w is the word in the dictionary
```

```r
  P_t_given_w <- function(t, w, theta) {
    matches <- sum(strsplit(t, NULL)[[1]] == strsplit(w, NULL)[[1]])
    k <- nchar(w)

    # Calculate likelihood based on number of matching letters
    if (matches == k) {
      return((1 - theta)^k)
    } else if (matches == k - 1) {
      return(theta * (1 - theta)^(k - 1))
    } else if (matches == k - 2) {
      return(theta^2 * (1 - theta)^(k - 2))
    } else if (matches == k - 3) {
      return(theta^3 * (1 - theta)^(k - 3))
    }
    return(0)
  }

  # Likelihoods for each word in the dictionary
  likelihoods <- sapply(dictionary, function(w) P_t_given_w(typed_word, w, theta))

  # Use Bayes Rule to calculate posterior probabilities
  posteriors <- sapply(1:length(dictionary), function(i) {
    w <- dictionary[i]
    if (w == "for") {
      likelihoods[i] * Prob_for
    } else {
      likelihoods[i] * Prob_other
    }
  })

  # Normalize the posterior probabilities to sum to 1
  posterior_sum <- sum(posteriors)
  posteriors <- posteriors / posterior_sum

  # Return the posterior probabilities
  return(data.frame(posterior = posteriors))
}
```

For each of the typed words “sun”, “the”, “foo”, give the probability that each word in the dictionary was the intended word. Perform this for the parameters below:

(a) $\alpha = 2$ and $\theta = 0.1$.

```r
spell_check("sun", alpha = 2, theta = 0.1)
```

```
##      posterior
## fun 0.09654350
## sun 0.86889154
## sit 0.01072706
## sat 0.01072706
## fan 0.01072706
## for 0.00238379
```

```r
spell_check("the", alpha = 2, theta = 0.1)
```

```
##     posterior
## fun 0.1428571
## sun 0.1428571
## sit 0.1428571
## sat 0.1428571
## fan 0.1428571
## for 0.2857143
```

```r
spell_check("foo", alpha = 2, theta = 0.1)
```

```
##       posterior
## fun 0.049180328
## sun 0.005464481
## sit 0.005464481
## sat 0.005464481
## fan 0.049180328
## for 0.885245902
```

(b) $\alpha = 50$ and $\theta = 0.1$.

```r
spell_check("sun", alpha = 50, theta = 0.1)
```

```
##      posterior
## fun 0.09131905
## sun 0.82187148
## sit 0.01014656
## sat 0.01014656
## fan 0.01014656
## for 0.05636979
```

```r
spell_check("the", alpha = 50, theta = 0.1)
```

```
##      posterior
## fun 0.01818182
## sun 0.01818182
## sit 0.01818182
## sat 0.01818182
## fan 0.01818182
## for 0.90909091
```

```r
spell_check("foo", alpha = 50, theta = 0.1)
```

```
##        posterior
## fun 0.0022107590
## sun 0.0002456399
## sit 0.0002456399
## sat 0.0002456399
## fan 0.0022107590
## for 0.9948415623
```

(c) $\alpha = 2$ and $\theta = 0.95$.

```r
spell_check("sun", alpha = 2, theta = 0.95)
```

```
##        posterior
## fun 1.281965e-03
## sun 6.747183e-05
## sit 2.435733e-02
## sat 2.435733e-02
## fan 2.435733e-02
## for 9.255786e-01
```

```r
spell_check("the", alpha = 2, theta = 0.95)
```

```
##     posterior
## fun 0.1428571
## sun 0.1428571
## sit 0.1428571
## sat 0.1428571
## fan 0.1428571
## for 0.2857143
```

```r
spell_check("foo", alpha = 2, theta = 0.95)
```

```
##       posterior
## fun 0.016918967
## sun 0.321460374
## sit 0.321460374
## sat 0.321460374
## fan 0.016918967
## for 0.001780944
```

Comment on the changes you observe in these three cases.

When the word **"sun"** is typed, which is in the dictionary, the spell checker correctly classifies it when $\theta = 0.1$. However, when $\theta = 0.95$, the spell checker thinks the user is typing "for". This is because the keystroke error rate is set too high and "for" is α times as likely to occur.

By letter index, the word **"the"** has no matches with the words in the dictionary. In each case the spell checker classifies the typed word as "for" since it is α times as likely to occur.

The word **"foo"** is one letter off from "for." The spell checker correctly classifies the word when $\theta = 0.1$. However, when $\theta = 0.95$., the keystroke error rate is set too high, so the spell checker thinks that typing an "f" is a mistake, as a result the spell checker misclassifies the word.

(4)

If 70% of a population is vaccinated, and the hospitalization rate is 5 times higher for an unvaccinated person than a vaccinated person, what is the probability that a person is vaccinated given they are hospitalized?

Given

$$\text{Prob(Vaccinated)} = 0.7 \implies \text{Prob(Unvaccinated)} = 0.3$$

and

$$\text{Prob(Hospitalized} \mid \text{Vaccinated)} = \frac{1}{5}\text{Prob(Hospitalized} \mid \text{Unvaccinated)}.$$

Let

$$\text{Prob(Hospitalized} \mid \text{Vaccinated)} = r$$

and

$$\text{Prob(Hospitalized} \mid \text{Unvaccinated)} = 5r.$$

So

$$\text{Prob(Hospitalized)} = r \cdot 0.7 + 5r \cdot 0.3 = r \cdot 2.2.$$

We need to calculate Prob(Vaccinated | Hospitalized). This can be done using Bayes' rule.

$$\begin{aligned}
\text{Prob(Vaccinated} \mid \text{Hospitalized)} &= \frac{\text{Prob(Vaccinated) Prob(Hospitalized} \mid \text{Vaccinated)}}{\text{Prob(Hospitalized)}} \\
&= \frac{0.7 \cdot r}{2.2 \cdot r} \\
&= \frac{0.7}{2.2} \\
&= 0.3182.
\end{aligned}$$

Code Appendix

```r
library(ggplot2)
library(viridis)
library(knitr)
# Joint PDF
f <- function(x1, x2){
  (1/(2*pi)) * (1 + x1^2 + x2^2)^(-3/2)
}
# Define the marginal PDFs f_1(x1) and f_2(x2) by integrating the joint PDF over
# x2 and x1 respectively
f_1 <- function(x1) {
  (1/pi) * (x1^2 + 1)^-1
}

f_2 <- function(x2) {
  (1/pi) * (x2^2 + 1)^-1
}

# Define the conditional PDF of X1 given X2
f_conditional <- function(x1, x2){
  f(x1, x2)/f_2(x2)
}
# Create a sequence of x1 values
x1_vals <- seq(-5, 5, length.out = 100)

# Define a vector of x2 values
x2_vals <- c(-3, -2, -1, 0, 1, 2, 3)

# Plot the conditional distributions for different x2 values
plot(NULL, xlim=c(-5, 5), ylim=c(0, .6), xlab="X1", ylab="Conditional PDF",
     main="Conditional Distributions of X1 given X2=x2")
colors <- rainbow(length(x2_vals))

for (i in 1:length(x2_vals)) {
  x2 <- x2_vals[i]
  y_vals <- sapply(x1_vals, function(x1) f_conditional(x1, x2))
  lines(x1_vals, y_vals, col=colors[i], lwd=2)
}

legend("topright", legend=paste("x2 =", x2_vals), col=colors, lwd=2)
# Set up a grid of values for x1 and x2
x1_vals <- seq(-3, 3, length.out = 100)
x2_vals <- seq(-3, 3, length.out = 100)

# Create a data frame for the joint PDF
joint_pdf_vals <- expand.grid(x1 = x1_vals, x2 = x2_vals)
joint_pdf_vals$z <- mapply(f, joint_pdf_vals$x1, joint_pdf_vals$x2)
# Plot the joint PDF as a contour plot
ggplot(joint_pdf_vals, aes(x = x1, y = x2, z = z)) +
  geom_tile(aes(fill = z), width = 0.1, height = 0.1) +
  scale_fill_viridis_c(option = "magma", name = "f(x1, x2)") +
  ggtitle("Joint PDF") +
```

```r
  labs(x = "X1", y = "X2") +
  geom_hline(yintercept = -3, color = colors[1]) +
  geom_hline(yintercept = -2, color = colors[2]) +
  geom_hline(yintercept = -1, color = colors[3]) +
  geom_hline(yintercept = 0, color = colors[4]) +
  geom_hline(yintercept = 1, color = colors[5]) +
  geom_hline(yintercept = 2, color = colors[6]) +
  geom_hline(yintercept = 3, color = colors[7]) +
  theme_minimal()
marg_X1_vals <- sapply(x1_vals, f_1)
marg_X2_vals <- sapply(x2_vals, f_2)

# Check if f_1(x1) * f_2(x2) equals the joint PDF
check_independence <- sapply(1:length(x1_vals), function(i) {
  prod_vals <- f_1(x1_vals[i]) * f_2(x2_vals[i])
  joint_val <- f_conditional(x1_vals[i], x2_vals[i])
  return(abs(prod_vals - joint_val) < 1e-6)  # tolerance for numerical error
})

all(check_independence)  # TRUE if independent, FALSE otherwise
spell_check <- function(typed_word, alpha, theta) {
  # Define the dictionary
  dictionary <- c("fun", "sun", "sit", "sat", "fan", "for")

  # Calculate the prior probabilities
  denom <- 5 + alpha
  Prob_for <- alpha / denom
  Prob_other <- 1 / denom

  # Define a function to calculate the likelihood P(t | w)
  # t is the typed word, w is the word in the dictionary
  P_t_given_w <- function(t, w, theta) {
    matches <- sum(strsplit(t, NULL)[[1]] == strsplit(w, NULL)[[1]])
    k <- nchar(w)

    # Calculate likelihood based on number of matching letters
    if (matches == k) {
      return((1 - theta)^k)
    } else if (matches == k - 1) {
      return(theta * (1 - theta)^(k - 1))
    } else if (matches == k - 2) {
      return(theta^2 * (1 - theta)^(k - 2))
    } else if (matches == k - 3) {
      return(theta^3 * (1 - theta)^(k - 3))
    }
    return(0)
  }

  # Likelihoods for each word in the dictionary
  likelihoods <- sapply(dictionary, function(w) P_t_given_w(typed_word, w, theta))

  # Use Bayes Rule to calculate posterior probabilities
  posteriors <- sapply(1:length(dictionary), function(i) {
```

```r
    w <- dictionary[i]
    if (w == "for") {
      likelihoods[i] * Prob_for
    } else {
      likelihoods[i] * Prob_other
    }
  })

  # Normalize the posterior probabilities to sum to 1
  posterior_sum <- sum(posteriors)
  posteriors <- posteriors / posterior_sum

  # Return the posterior probabilities
  return(data.frame(posterior = posteriors))
}
spell_check("sun", alpha = 2, theta = 0.1)
spell_check("the", alpha = 2, theta = 0.1)
spell_check("foo", alpha = 2, theta = 0.1)
spell_check("sun", alpha = 50, theta = 0.1)
spell_check("the", alpha = 50, theta = 0.1)
spell_check("foo", alpha = 50, theta = 0.1)
spell_check("sun", alpha = 2, theta = 0.95)
spell_check("the", alpha = 2, theta = 0.95)
spell_check("foo", alpha = 2, theta = 0.95)
```