Assignment 3

Xiang Li

2023/11/30

Exercise 1

1

```
is(penguins)
```

```
## [1] "tbl_df"     "tbl"        "data.frame" "list"       "oldClass"
## [6] "vector"
```

```
penguins_df = as.data.frame(penguins)
```

The type of penguins is tibble.

2

```
# Compute the (joint) frequency distribution
n = nrow(penguins_df)
count_df = count(penguins_df, species, island)
count_df$pmf = count_df$n/n
count_df
```

```
##     species    island   n       pmf
## 1    Adelie    Biscoe  44 0.1279070
## 2    Adelie     Dream  56 0.1627907
## 3    Adelie Torgersen  52 0.1511628
## 4 Chinstrap     Dream  68 0.1976744
## 5    Gentoo    Biscoe 124 0.3604651
```

```
# How many penguin species
unique(penguins_df$species)
```

```
## [1] Adelie    Gentoo    Chinstrap
## Levels: Adelie Chinstrap Gentoo
```

Based on the table I get, there are totally three species and Adelie penguins are present on different islands.

3

```
x = penguins_df[penguins_df$species == "Chinstrap", ]$bill_length_mm
y = penguins_df[penguins_df$species == "Gentoo", ]$bill_length_mm
hp_test = t.test(x = x, y = y, alternative = "less", var.equal = FALSE)
hp_test$p.value
```

```
## [1] 0.9961348
```

Based on the p value, I conclude that the null hypothesis is not rejected, there is not enough evidence to support $\mu_C < \mu_G$.

Exercise 2

1

```
# check if missing data
is_na_df = sapply(penguins_df[, c("species", "sex", "body_mass_g")],
    is.na)
sum(is_na_df)
```

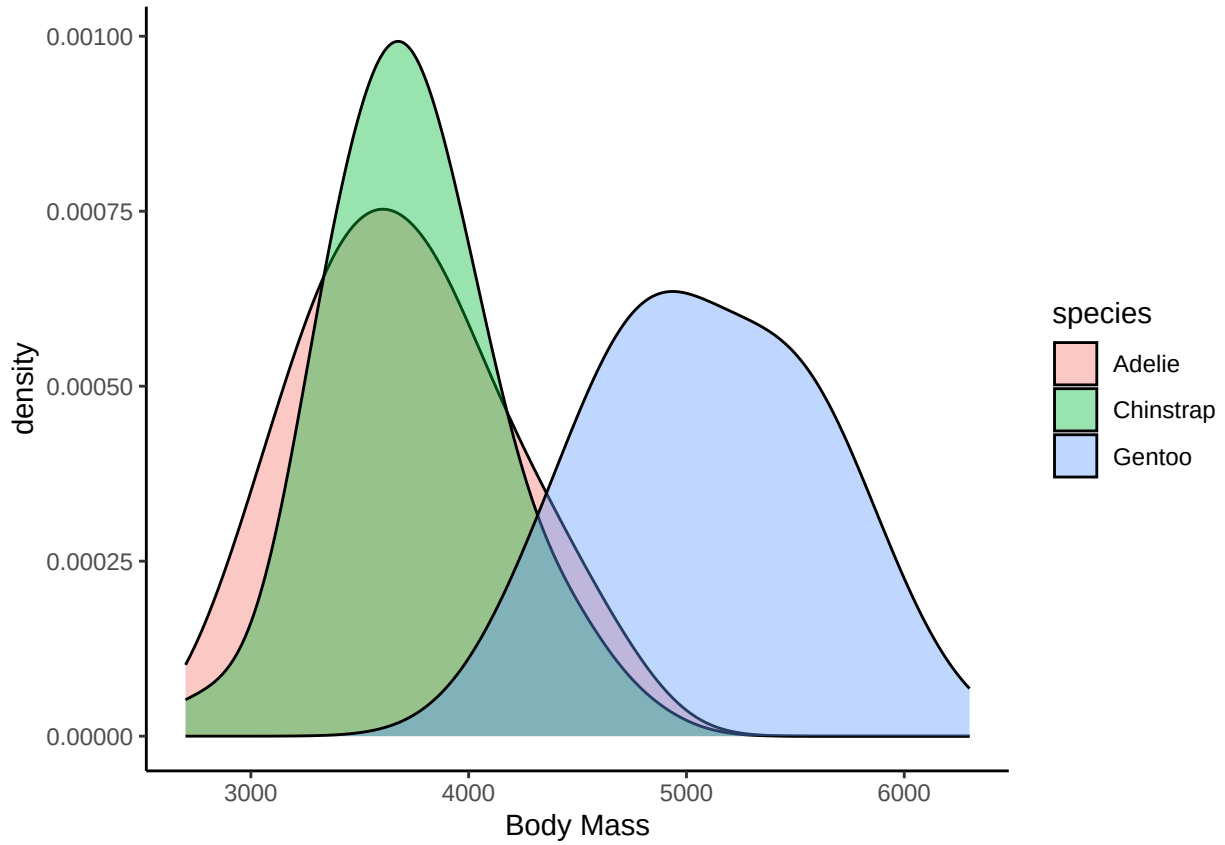
```
## [1] 13
```

```
# remove missing data
penguins_df1 = penguins_df[rowSums(is_na_df) == 0, ]
```

Since sum(is_na_df) > 0, there is missing data.

2

```
ggplot(data = penguins_df1, aes(x = body_mass_g, group = species,
    fill = species)) + geom_density(adjust = 1.5, alpha = 0.4) +
    theme_classic() + labs(x = "Body Mass")
```



The distribution of body mass from each species is approximately normal. The expected values of body mass for Adelie penguins and Chinstrap penguins are similar. The expected value of body mass for Gentoo penguins is larger than other two.

3

```
penguins_df1$body_mass_kg = penguins_df1$body_mass_g/1000
neg_logl = function(theta, pi1, pi2, w1, w2, x) {
    mu1 = theta[1]
    sigma1 = theta[2]
    mu2 = theta[3]
    sigma2 = theta[4]
    mu3 = theta[5]
    sigma3 = theta[6]
    result = -sum(w1 * log(pi1 * dnorm(x, mu1, sigma1)) +
        w2 * log(pi2 * dnorm(x, mu2, sigma2)) + (1 - w1 -
        w2) * log((1 - pi1 - pi2) * dnorm(x, mu3, sigma3)))
    return(result)
}
```

4

```
set.seed(4013115)
EM_func = function(x, n.iter) {
    # 1. preallocate objects
    n = length(x)
    pi1hat = rep(NA, n.iter)
    pi2hat = rep(NA, n.iter)
    p1hat = matrix(NA, n.iter, n)
    p2hat = matrix(NA, n.iter, n)
    thetahat = matrix(NA, n.iter, 6)
    # 2: initialize the algorithm
    pi1hat[1] = runif(n = 1, min = 0.1, max = 0.3)
    pi2hat[1] = runif(n = 1, min = 0.1, max = 0.3)
    p1hat[1, ] = runif(n, 0.1, 2 * pi1hat[1] - 0.1)
    p2hat[1, ] = runif(n, 0.1, 2 * pi2hat[1] - 0.1)
    # 3. first M step:
    thetahat[1, ] = optim(c(1, 1, 1, 1, 1, 1), function(theta) neg_logl(theta,
        pi1hat[1], pi2hat[1], p1hat[1, ], p2hat[1, ], x))$par
    # 4. run the EM:
    for (t in 2:n.iter) {
        # E step: update individual probability
        # memberships
        p.temp = cbind(pi1hat[t - 1] * dnorm(x, thetahat[t -
            1, 1], thetahat[t - 1, 2]), pi2hat[t - 1] *
            dnorm(x, thetahat[t - 1, 3], thetahat[t - 1,
                4]), (1 - pi1hat[t - 1] - pi2hat[t - 1]) *
            dnorm(x, thetahat[t - 1, 5], thetahat[t - 1,
                6]))
        p1hat[t, ] = p.temp[, 1]/rowSums(p.temp)
        p2hat[t, ] = p.temp[, 2]/rowSums(p.temp)
        # M step: update parameter estimates
        pi1hat[t] = mean(p1hat[t, ])
        pi2hat[t] = mean(p2hat[t, ])
        thetahat[t, ] = optim(thetahat[t - 1, ], function(theta) neg_logl(theta,
            pi1hat[t], pi2hat[t], p1hat[t, ], p2hat[t, ],
            x))$par
    }
    # 5: compute the loglikelihood at the end of the
    # algorithm
    loglikFinal = -neg_logl(theta = thetahat[n.iter, ],
        pi1 = pi1hat[n.iter], pi2 = pi2hat[n.iter], w1 = p1hat[n.iter,
            ], w2 = p2hat[n.iter, ], x)
    # 6: define the exports
    out = list(thetahat = thetahat, pi1hat = pi1hat, pi2hat = pi2hat,
        p1hat = p1hat, p2hat = p2hat, logl = loglikFinal)
    return(out)
}
rep_matrix = replicate(3, EM_func(penguins_df1$body_mass_kg,
    500))
rep_matrix
```

```
##          [,1]          [,2]          [,3]
```

```
## thetahat numeric,3000   numeric,3000   numeric,3000
## pi1hat   numeric,500    numeric,500    numeric,500
## pi2hat   numeric,500    numeric,500    numeric,500
## p1hat    numeric,166500 numeric,166500 numeric,166500
## p2hat    numeric,166500 numeric,166500 numeric,166500
## logl     -454.8023      -457.1999      -452.1382
```

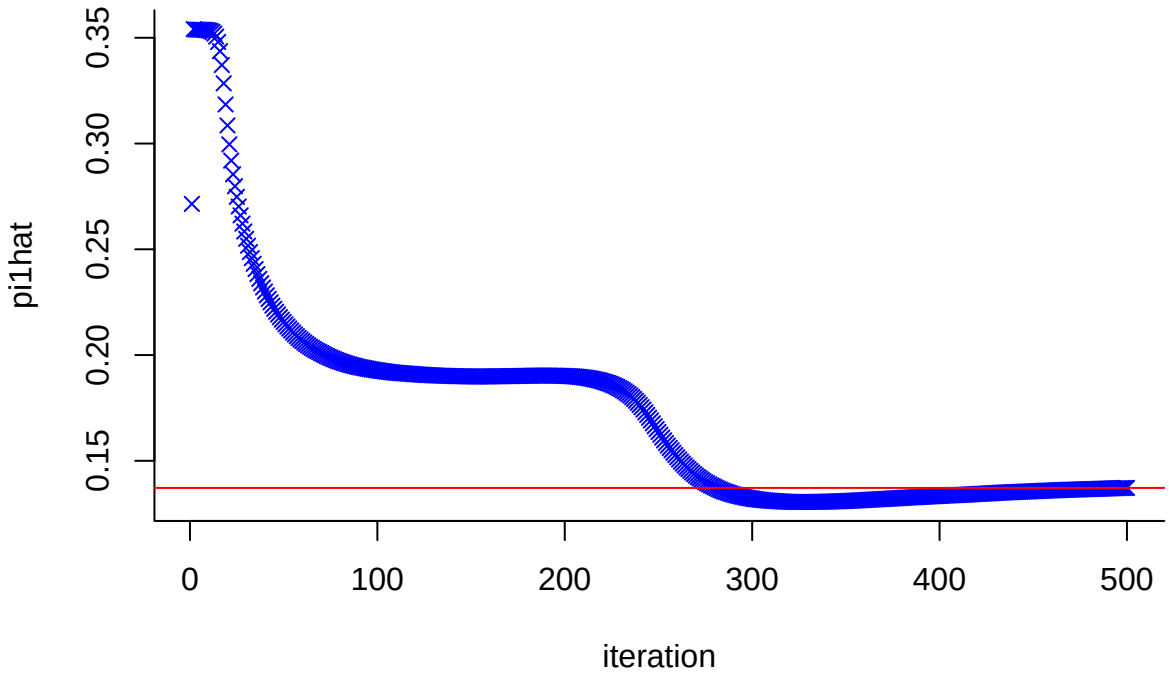
How to choose starting points: randomly choose π_1 between $[0.1, 0.3]$ and π_2 between $[0.1, 0.3]$, then generate p_{i1} by $E(p_{i1}) = \pi_1$ and p_{i2} by $E(p_{i2}) = \pi_2$.

5

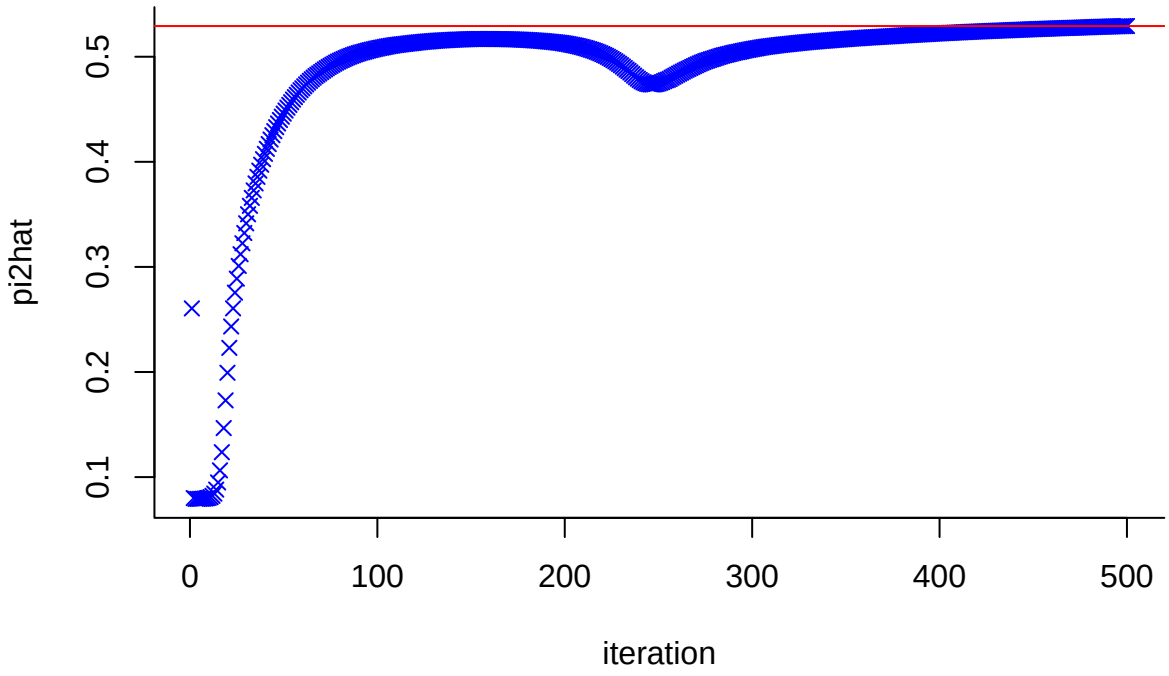
I should pick the solution corresponding to the smallest final loglikelihood function value, i.e. the 2nd solution.

6

```
pi1hat = rep_matrix[2, 2]$pi1hat
par(bty = "l")
plot(x = 1:500, y = pi1hat, ylab = "pi1hat", xlab = "iteration",
    pch = 4, col = "blue")
abline(h = pi1hat[500], col = "red")
```



```
pi2hat = rep_matrix[3, 2]$pi2hat
par(bty = "l")
plot(x = 1:500, y = pi2hat, ylab = "pi2hat", xlab = "iteration",
    pch = 4, col = "blue")
abline(h = pi2hat[500], col = "red")
```



Based on the scatter plots above, the algorithm appears to converge afer 500 iterations.

7

```
pi1_result = pi1hat[500]
pi2_result = pi2hat[500]
pi3_result = 1 - pi1_result - pi2_result
mu1_result = rep_matrix[1, 2]$thetahat[500, 1]
sigma1_result = rep_matrix[1, 2]$thetahat[500, 2]
mu2_result = rep_matrix[1, 2]$thetahat[500, 3]
sigma2_result = rep_matrix[1, 2]$thetahat[500, 4]
mu3_result = rep_matrix[1, 2]$thetahat[500, 5]
sigma3_result = rep_matrix[1, 2]$thetahat[500, 6]
list(pi1 = pi1_result, pi2 = pi2_result, pi3 = pi3_result,
    mu1 = mu1_result, mu2 = mu2_result, mu3 = mu3_result,
    sigma1 = sigma1_result, sigma2 = sigma2_result, sigma3 = sigma3_result)
```

```
## $pi1
```

```
## [1] 0.1371821
##
## $pi2
## [1] 0.5291915
##
## $pi3
## [1] 0.3336264
##
## $mu1
## [1] 5.572965
##
## $mu2
## [1] 3.592194
##
## $mu3
## [1] 4.620556
##
## $sigma1
## [1] 0.2800833
##
## $sigma2
## [1] 0.3525823
##
## $sigma3
## [1] 0.412746
```

The weights of the 3 components are not similar.

```
mean_mass_df = aggregate(penguins_df1[, "body_mass_kg",
    drop = FALSE], by = list(species = penguins_df1$species),
    FUN = mean)
mean_mass_df
```

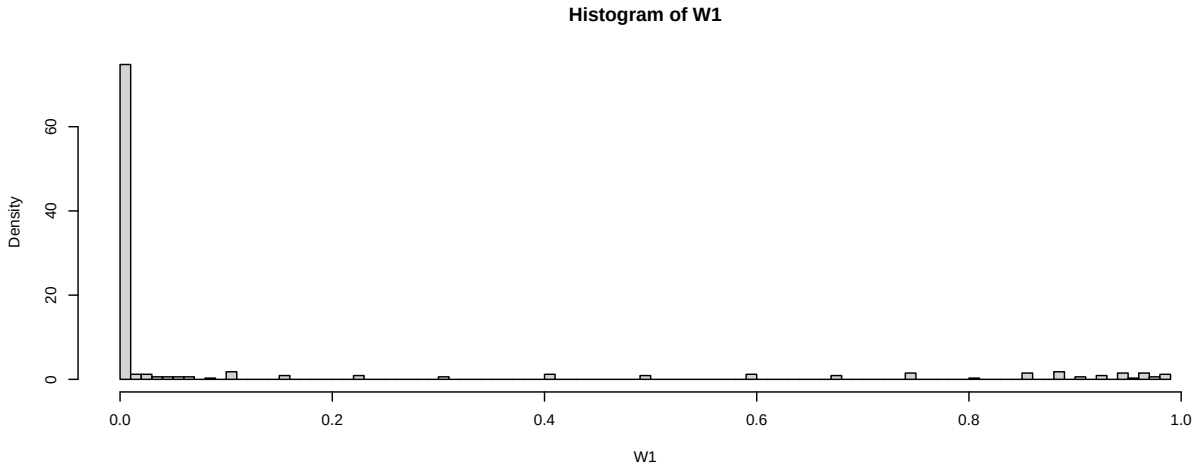
```
##     species body_mass_kg
## 1    Adelie     3.706164
## 2 Chinstrap     3.733088
## 3    Gentoo     5.092437
```

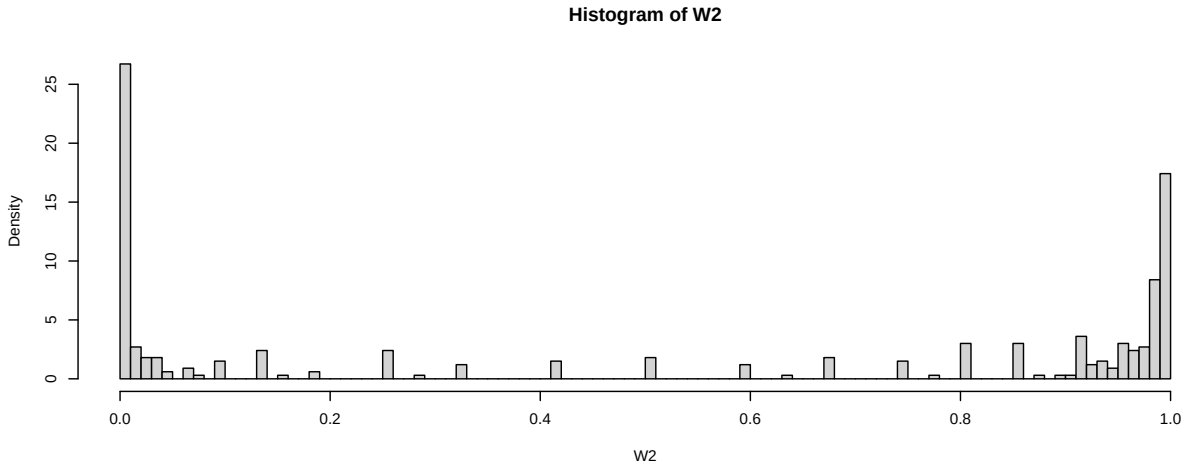
The estimates of μ_1, μ_2, μ_3 are very different from the mean weights.

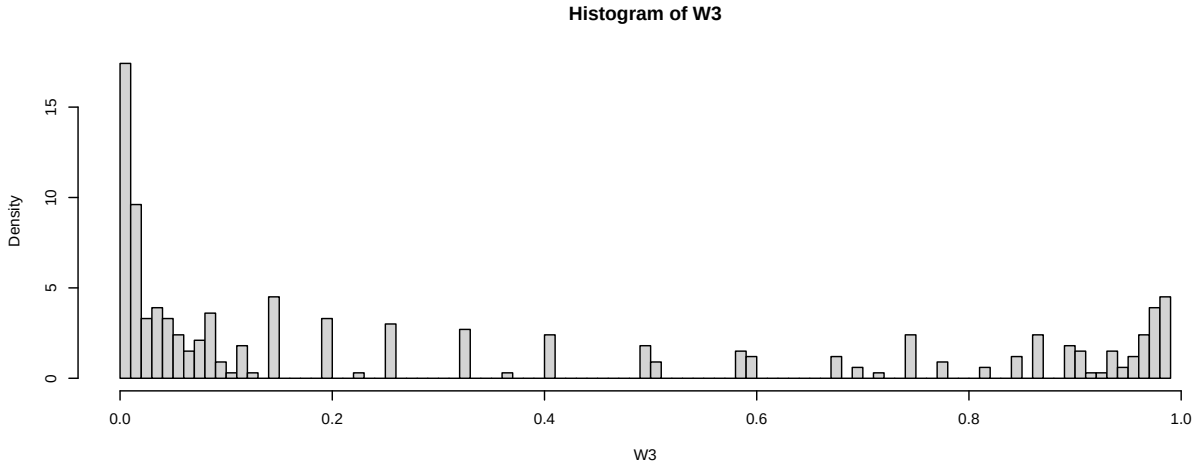
8

```
x = penguins_df1$body_mass_kg
W_denomi = dnorm(x, mu1_result, sigma1_result) * pi1_result +
    dnorm(x, mu2_result, sigma2_result) * pi2_result + dnorm(x,
    mu3_result, sigma3_result) * pi3_result
W1 = dnorm(x, mu1_result, sigma1_result) * pi1_result/W_denomi
W2 = dnorm(x, mu2_result, sigma2_result) * pi2_result/W_denomi
W3 = dnorm(x, mu3_result, sigma3_result) * pi3_result/W_denomi
par(mfrow = c(3, 1))
hist(W1, breaks = 100, probability = TRUE)
```

```
hist(W2, breaks = 100, probability = TRUE)
hist(W3, breaks = 100, probability = TRUE)
```







9

```
W = as.data.frame(cbind(W1, W2, W3))
predictedComponent = apply(W, 1, which.max)
head(predictedComponent, 10)
```

```
##  [1] 2 2 2 2 2 2 3 2 2 3
```

10

```
penguins_df1$predictedComponent = predictedComponent
count_df1 = count(penguins_df1, sex, predictedComponent)
count_df1$pmf = count_df1$n/n
count_df1
```

```
##      sex predictedComponent   n       pmf
## 1 female                  2 109 0.3168605
## 2 female                  3  56 0.1627907
## 3   male                  1  46 0.1337209
## 4   male                  2  74 0.2151163
## 5   male                  3  48 0.1395349
```

```
count_df2 = count(penguins_df1, species, predictedComponent)
count_df2$pmf = count_df2$n/n
count_df2
```

```
##     species predictedComponent   n         pmf
## 1    Adelie                  2 120 0.348837209
## 2    Adelie                  3  26 0.075581395
## 3 Chinstrap                  2  61 0.177325581
## 4 Chinstrap                  3   7 0.020348837
## 5    Gentoo                  1  46 0.133720930
## 6    Gentoo                  2   2 0.005813953
## 7    Gentoo                  3  71 0.206395349
```

11

```
p_1 = sum(count_df1[count_df1$predictedComponent == 1, "pmf"])
p_male_1 = count_df1[count_df1$sex == "male" & count_df1$predictedComponent ==
    1, "pmf"]/p_1
p_2 = sum(count_df1[count_df1$predictedComponent == 2, "pmf"])
p_male_2 = count_df1[count_df1$sex == "male" & count_df1$predictedComponent ==
    2, "pmf"]/p_2
p_3 = sum(count_df1[count_df1$predictedComponent == 3, "pmf"])
p_male_3 = count_df1[count_df1$sex == "male" & count_df1$predictedComponent ==
    3, "pmf"]/p_3
list(p_male_1 = p_male_1, p_female_1 = 1 - p_male_1, p_male_2 = p_male_2,
    p_female_2 = 1 - p_male_2, p_male_3 = p_male_3, p_female_3 = 1 -
        p_male_3)
```

```
## $p_male_1
## [1] 1
##
## $p_female_1
## [1] 0
##
## $p_male_2
## [1] 0.4043716
##
## $p_female_2
## [1] 0.5956284
##
## $p_male_3
## [1] 0.4615385
##
## $p_female_3
## [1] 0.5384615
```

If $predictedComponent = 1$, I can guess that the penguin is a male. Because based on the table of sex and predictedComponent, I find that $P(sex = male|predictedComponent = 1) > 0.5$, the accurate would be 100%.

If $predictedComponent = 2$, I can guess that the penguin is a female. Because based on the table of sex and predictedComponent, I find that $P(sex = female|predictedComponent = 2) > 0.5$, the accurate would be 59.56%.

If $predictedComponent = 3$, I can guess that the penguin is a female. Because based on the table of sex and predictedComponent, I find that $P(sex = female|predictedComponent = 3) > 0.5$, the accurate would be 53.85%.

12

```
p_A_1 = count_df2[count_df2$species == "Adelie" & count_df2$predictedComponent ==
    1, "pmf"]/p_1
p_C_1 = count_df2[count_df2$species == "Chinstrap" & count_df2$predictedComponent ==
    1, "pmf"]/p_1
p_G_1 = count_df2[count_df2$species == "Gentoo" & count_df2$predictedComponent ==
    1, "pmf"]/p_1
p_A_2 = count_df2[count_df2$species == "Adelie" & count_df2$predictedComponent ==
    2, "pmf"]/p_2
p_C_2 = count_df2[count_df2$species == "Chinstrap" & count_df2$predictedComponent ==
    2, "pmf"]/p_2
p_G_2 = count_df2[count_df2$species == "Gentoo" & count_df2$predictedComponent ==
    2, "pmf"]/p_2
p_A_3 = count_df2[count_df2$species == "Adelie" & count_df2$predictedComponent ==
    3, "pmf"]/p_3
p_C_3 = count_df2[count_df2$species == "Chinstrap" & count_df2$predictedComponent ==
    3, "pmf"]/p_3
p_G_3 = count_df2[count_df2$species == "Gentoo" & count_df2$predictedComponent ==
    3, "pmf"]/p_3
list(p_A_1 = p_A_1, p_C_1 = p_C_1, p_G_1 = p_G_1, p_A_2 = p_A_2,
    p_C_2 = p_C_2, p_G_2 = p_G_2, p_A_3 = p_A_3, p_C_3 = p_C_3,
    p_G_3 = p_G_3)
```

```
## $p_A_1
## numeric(0)
##
## $p_C_1
## numeric(0)
##
## $p_G_1
## [1] 1
##
## $p_A_2
## [1] 0.6557377
##
## $p_C_2
## [1] 0.3333333
##
## $p_G_2
## [1] 0.01092896
##
## $p_A_3
## [1] 0.25
##
## $p_C_3
## [1] 0.06730769
##
## $p_G_3
## [1] 0.6826923
```

If $predictedComponent = 1$, I can guess that the penguin is a Gentoo penguin. Because based on the table of species and predictedComponent, I find that $P(species = Gentoo|predictedComponent = 1) > 0.5$, the accurate would be 100%.

If $predictedComponent = 2$, I can guess that the penguin is an Adelie penguin. Because based on the table of species and predictedComponent, I find that $P(species = Adelie|predictedComponent = 2) > 0.5$, the accurate would be 65.57%.

If $predictedComponent = 3$, I can guess that the penguin is a Gentoo penguin. Because based on the table of species and predictedComponent, I find that $P(species = Gentoo|predictedComponent = 3) > 0.5$, the accurate would be 68.27%.