Statisical Computing with R - Assignment 3

Colin Yip, Student No. 3953629

Exercise 1

Formatting and output config.

```
library(knitr)
library(formatR)
library(palmerpenguins)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 80))
```

Q1

```
# Read in penguins
penguins_data <- palmerpenguins::penguins
# Print penguins to inspect
print(penguins_data)
```

```
## # A tibble: 344 x 8
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Torgersen           39.1          18.7               181        3750
##  2 Adelie  Torgersen           39.5          17.4               186        3800
##  3 Adelie  Torgersen           40.3          18                 195        3250
##  4 Adelie  Torgersen           NA            NA                  NA          NA
##  5 Adelie  Torgersen           36.7          19.3               193        3450
##  6 Adelie  Torgersen           39.3          20.6               190        3650
##  7 Adelie  Torgersen           38.9          17.8               181        3625
##  8 Adelie  Torgersen           39.2          19.6               195        4675
##  9 Adelie  Torgersen           34.1          18.1               193        3475
## 10 Adelie  Torgersen           42            20.2               190        4250
## # i 334 more rows
## # i 2 more variables: sex <fct>, year <int>
```

```r
# Convert penguins to a dataframe
penguins_df <- as.data.frame(penguins_data)
```

penguins is a tibble.

Q2

```r
# Count penguins by species and island
penguins_frequency_dist <- penguins_df %>%
  group_by(species, island) %>%
  count()
# Convert to DF
penguins_frequency_dist_df <- as.data.frame(penguins_frequency_dist)
# Print results
knitr::kable(penguins_frequency_dist_df,
  caption = "Distribution Frequency of Penguins by Species and Island"
)
```

Table 1: Distribution Frequency of Penguins by Species and Island

species	island	n
Adelie	Biscoe	44
Adelie	Dream	56
Adelie	Torgersen	52
Chinstrap	Dream	68
Gentoo	Biscoe	124

There are 3 different species in this dataset. The following are the species found on each island.

```r
# Get unique penguin categories by island
species_by_island_df <- penguins_df %>%
  group_by(island) %>%
  summarise(species = unique(species))
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'island'. You can override using the
## `.groups` argument.
```

```r
# Print results
knitr::kable(species_by_island_df,
  caption = "Penguin Species by Island"
)
```

Table 2: Penguin Species by Island

island	species
Biscoe	Adelie
Biscoe	Gentoo
Dream	Adelie
Dream	Chinstrap
Torgersen	Adelie

Q3

```r
# Filter for Gentoo penguins and get bill length values
gentoo_idx <- which(penguins_df$species == "Gentoo")
gentoo_bill_lengths <- penguins_df[gentoo_idx, ]$bill_length_mm

# Filter for Chinstrap penguins and get bill length values
chinstrap_idx <- which(penguins_df$species == "Chinstrap")
chinstrap_bill_lengths <- penguins_df[chinstrap_idx, ]$bill_length_mm

# Define significance value
alpha <- 0.05

# Do t test to evaluate if chinstrap average length
# is greater than gentoo average length
t_test_results <- t.test(
  x = chinstrap_bill_lengths,
  y = gentoo_bill_lengths,
  alternative = "less",
  var.equal = F
)
t_test_pvalue <- t_test_results$p.value
# Evaluate if p-value of t test is greater than signif. value
if (t_test_pvalue < alpha) {
  print("p-value of t-test is less than alpha, so we reject H0")
} else {
  print("p-value of t-test is greater than alpha, so we do not reject H0")
}
```

```
## [1] "p-value of t-test is greater than alpha, so we do not reject H0"
```

Since a p-value of 0.9961348 is greater than than $\alpha = 0.05$, we cannot reject H_0 that $\mu_C \geq \mu_G$, and conclude that the expected value of bill lengths of Chinstrap penguins is greater than that of Gentoo penguins.

Exercise 2

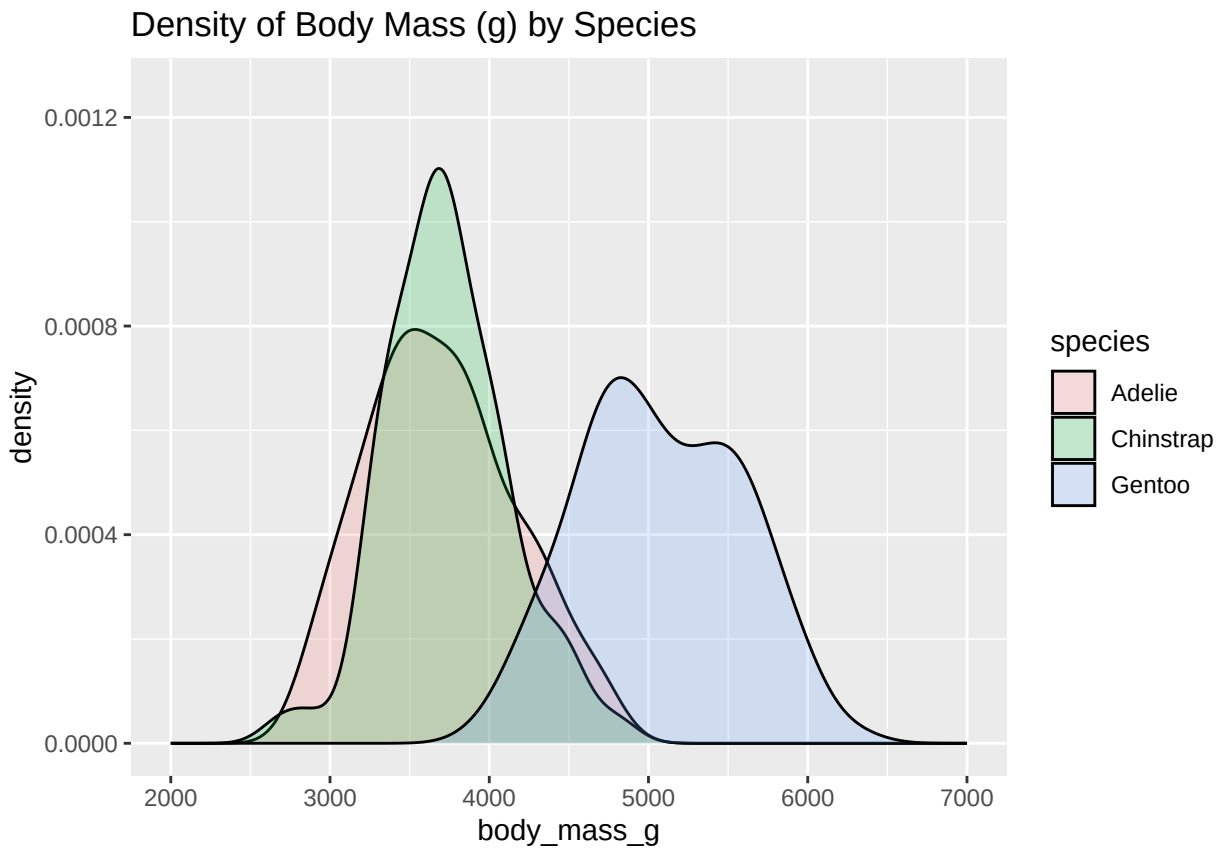
Q1

```
species_na_bool_idx <- is.na(penguins_df$species)
body_mass_na_bool_idx <- is.na(penguins_df$body_mass_g)
combined_bool_idx <- !species_na_bool_idx & !body_mass_na_bool_idx

filtered_penguins_df <- penguins_df[combined_bool_idx, ]
rows_removed_num <- nrow(penguins_df) - nrow(filtered_penguins_df)
```

There are 2 rows removed due to missing `species` or `body_mass` values.

Q2

```
species <- unique(filtered_penguins_df$species)
species_color_list <- c("red", "blue", "green")
combined_species_df <- data.frame()
for (species_i in species) {
  species_idx <- filtered_penguins_df$species == species_i
  species_df <- filtered_penguins_df[species_idx, ]
  species_df_slice <- species_df[c("species", "body_mass_g")]
  combined_species_df <- rbind(combined_species_df, species_df_slice)
}

ggplot(combined_species_df, aes(body_mass_g, fill = species)) +
  geom_density(alpha = 0.2) +
  xlim(2000, 7000) +
  ylim(0, 1.25e-3) +
  ggtitle("Density of Body Mass (g) by Species")
```



It appears that Chinstrap and Adelie pengins have a similar average weight, which is lower than that of Gentoo penguins. Adelie and Gentoo penguins also have wider variance in weight than Chinstrap penguins.

Q3

```
filtered_penguins_df$body_mass_kg <- filtered_penguins_df$body_mass_g / 1000
neg.logl <- function(theta, pi1, pi2, pi3, w1, w2, w3, x) {
  mu1 <- theta[1]
  mu2 <- theta[2]
  mu3 <- theta[3]

  sigma_1 <- exp(theta[4])
  sigma_2 <- exp(theta[5])
  sigma_3 <- exp(theta[6])
  # density of the mixture model:
  f.x1 <- pi1 * dnorm(x, mu1, sd = sigma_1)
  f.x2 <- pi2 * dnorm(x, mu2, sd = sigma_2)
  f.x3 <- pi3 * dnorm(x, mu3, sd = sigma_3)
  # negative log-likelihood:
  -sum(w1 * log(f.x1) + w2 * log(f.x2) + w3 * log(f.x3))
}
```

Q4

```
set.seed(3953629)

em_algo <- function(x, pi1hat_init, pi2hat_init, pi3hat_init, n.iter) {
  # Set n as length of provided x
  n <- length(x)
```

```r
# Set up pi_j for each j component
pi1hat <- rep(NA, n.iter)
pi2hat <- rep(NA, n.iter)
pi3hat <- rep(NA, n.iter)

# Set up p_hat_j for each j component
p1hat <- matrix(NA, n.iter, n)
p2hat <- matrix(NA, n.iter, n)
p3hat <- matrix(NA, n.iter, n)

# Define starting range for pi_j
pi1hat_range <- pi1hat_init * 0.8
pi2hat_range <- pi2hat_init * 0.8
pi3hat_range <- pi3hat_init * 0.8

# Store initial provided pi values
pi1hat[1] <- pi1hat_init
pi2hat[1] <- pi2hat_init
pi3hat[1] <- pi3hat_init

# Calculate t=1 p_hats
p1hat[1, ] <- runif(n, pi1hat_init - pi1hat_range, pi1hat_init + pi1hat_range)
p2hat[1, ] <- runif(n, pi2hat_init - pi2hat_range, pi2hat_init + pi2hat_range)
p3hat[1, ] <- runif(n, pi3hat_init - pi3hat_range, pi3hat_init + pi3hat_range)

# Set up 6 column theta matrix, and t=1 theta guesses
thetahat <- matrix(NA, n.iter, 6)
theta_init_guesses <- c(
  rep(mean(x), 3),
  rep(sd(x), 3)
)

# Find initial log likelihood minimized values of theta
thetahat[1, ] <- optim(
  theta_init_guesses,
  function(theta) {
    neg.logl(
      theta, pi1hat[1], pi2hat[1], pi3hat[1],
      p1hat[1, ], p2hat[1, ], p3hat[1, ], x
    )
  }
)$par

# Iter through n.iter
for (t in 2:n.iter) {
  # E step: update individual probability memberships
  phat_numer.temp <- cbind(
    pi1hat[t - 1] * dnorm(x, thetahat[t - 1, 1], exp(thetahat[t - 1, 4])),
    pi2hat[t - 1] * dnorm(x, thetahat[t - 1, 2], exp(thetahat[t - 1, 5])),
    pi3hat[t - 1] * dnorm(x, thetahat[t - 1, 3], exp(thetahat[t - 1, 6]))
  )
  # Reassign phat values to corresponding phat matrix
  p1hat[t, ] <- phat_numer.temp[, 1] / rowSums(phat_numer.temp)
  p2hat[t, ] <- phat_numer.temp[, 2] / rowSums(phat_numer.temp)
  p3hat[t, ] <- phat_numer.temp[, 3] / rowSums(phat_numer.temp)
  # M step: update parameter estimates
  # Update for corresponding pi_hat
  pi1hat[t] <- mean(p1hat[t, ])
  pi2hat[t] <- mean(p2hat[t, ])
```

```
    pi3hat[t] <- mean(p3hat[t, ])
    # Update optimized theta values
    thetahat[t, ] <- optim(thetahat[t - 1, ], function(theta) {
      neg.logl(
        theta, pi1hat[t], pi2hat[t], pi3hat[t],
        p1hat[t, ], p2hat[t, ], p3hat[t, ], x
      )
    })$par
  }
  # Return final theta values, pi values, and phat values
  return(list(
    theta = thetahat,
    pi1 = pi1hat,
    pi2 = pi2hat,
    pi3 = pi3hat,
    p1 = p1hat,
    p2 = p2hat,
    p3 = p3hat
  ))
}

# Get proportions of penguins
n <- nrow(filtered_penguins_df)
peng_species_prop <- filtered_penguins_df %>%
  group_by(species) %>%
  count()
peng_species_prop$prop <- peng_species_prop$n / n

# Define even probabilities
pi1hat_1 <- 1 / 3
pi2hat_1 <- 1 / 3
pi3hat_1 <- 1 / 3

# Get proportions for pi
pi1hat_2 <- c(peng_species_prop$prop)[1]
pi2hat_2 <- c(peng_species_prop$prop)[2]
pi3hat_2 <- c(peng_species_prop$prop)[3]

# Randomize pi_j
pi1hat_3 <- runif(1, 0, 0.45)
pi2hat_3 <- runif(1, 0, 0.45)
pi3hat_3 <- 1 - pi1hat_3 - pi2hat_3

n.iter <- 500

# Run em_algo n.iter times
run_1 <- em_algo(
  filtered_penguins_df$body_mass_kg,
  pi1hat_1,
  pi2hat_1,
  pi3hat_1,
  n.iter
)
run_2 <- em_algo(
  filtered_penguins_df$body_mass_kg,
  pi1hat_2,
  pi2hat_2,
  pi3hat_2,
  n.iter
```

```
)
run_3 <- em_algo(
  filtered_penguins_df$body_mass_kg,
  pi1hat_3,
  pi2hat_3,
  pi3hat_3,
  n.iter
)
```

The starting points for `run_1` assume even probability across all three penguin distributions.

The starting points for `run_2` follows the empirical frequency distribution in the data.

THe starting points for `run_3` are randomized, ensuring all three π_j still add to 1.

Q5

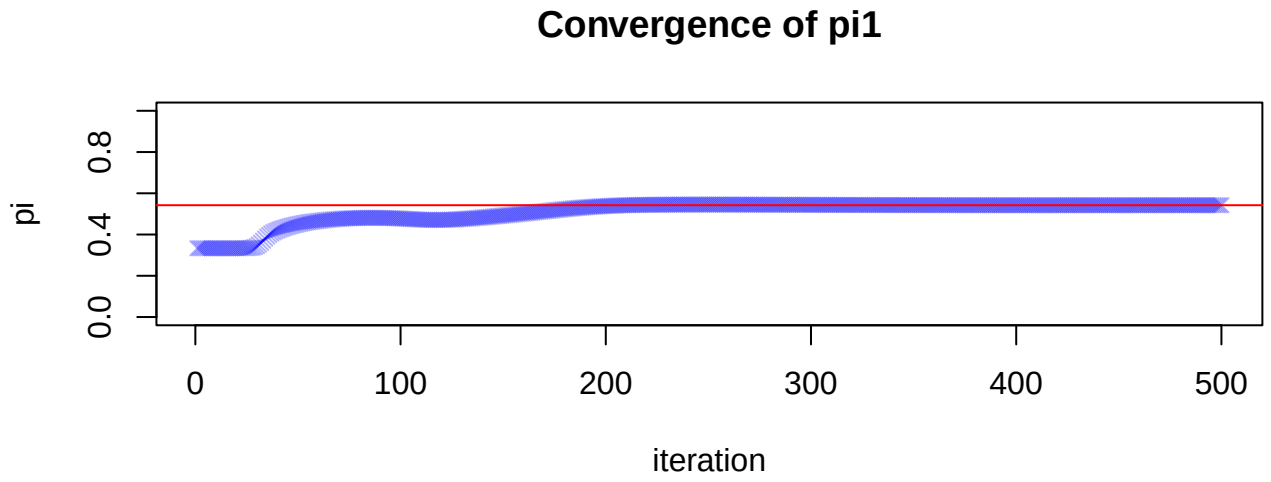
```
# Get actual props of penguins
actual_props <- c(peng_species_prop$prop)

# Get predicted values for each run using p_hat[n.iter] > 0.5
run_1_props <- list(
  Adelie = sum(run_1$p1[n.iter, ] > 0.5) / n,
  Gentoo = sum(run_1$p2[n.iter, ] > 0.5) / n,
  Chinstrap = sum(run_1$p3[n.iter, ] > 0.5) / n
)
run_2_props <- list(
  Chinstrap = sum(run_2$p1[n.iter, ] > 0.5) / n,
  Adelie = sum(run_2$p2[n.iter, ] > 0.5) / n,
  Gentoo = sum(run_2$p3[n.iter, ] > 0.5) / n
)
run_3_props <- list(
  Adelie = sum(run_3$p1[n.iter, ] > 0.5) / n,
  Gentoo = sum(run_3$p2[n.iter, ] > 0.5) / n,
  Chinstrap = sum(run_3$p3[n.iter, ] > 0.5) / n
)

# Build comps
adelie_comp <- run_1_props$Adelie - c(
  run_1_props$Adelie,
  run_2_props$Adelie,
  run_3_props$Adelie
)
chins_comp <- run_1_props$Chinstrap - c(
  run_1_props$Chinstrap,
  run_2_props$Chinstrap,
  run_3_props$Chinstrap
)
gentoo_comp <- run_1_props$Gentoo - c(
  run_1_props$Gentoo,
  run_2_props$Gentoo,
  run_3_props$Gentoo
)

knitr::kable(rbind(adelie_comp, chins_comp, gentoo_comp),
  col.names = c("Run 1", "Run 2", "Run 3")
)
```

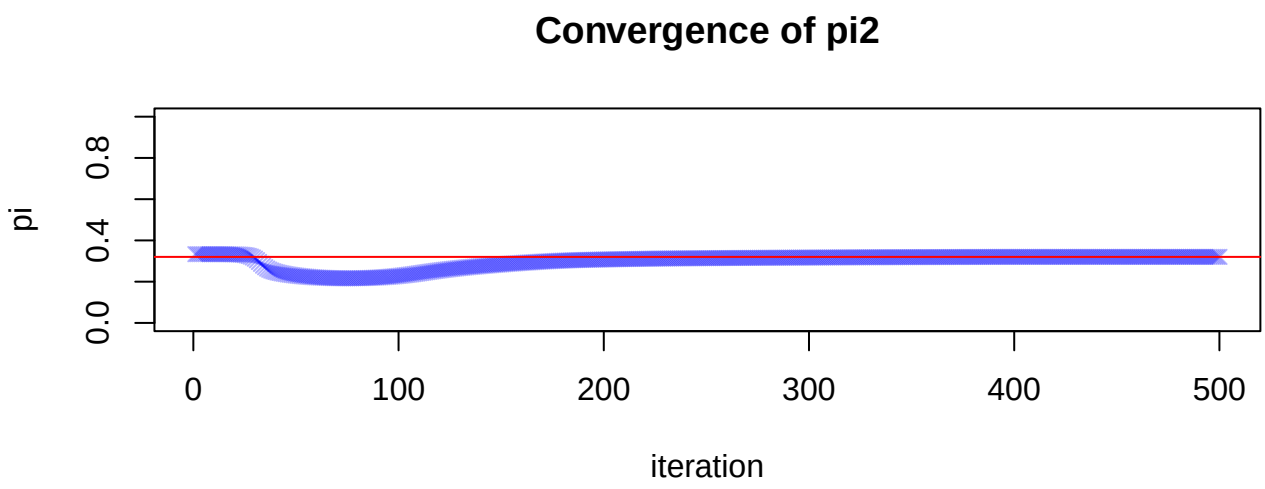
	Run 1	Run 2	Run 3
adelie_comp	0	0.0000000	0
chins_comp	0	0.0087719	0
gentoo_comp	0	-0.0087719	0

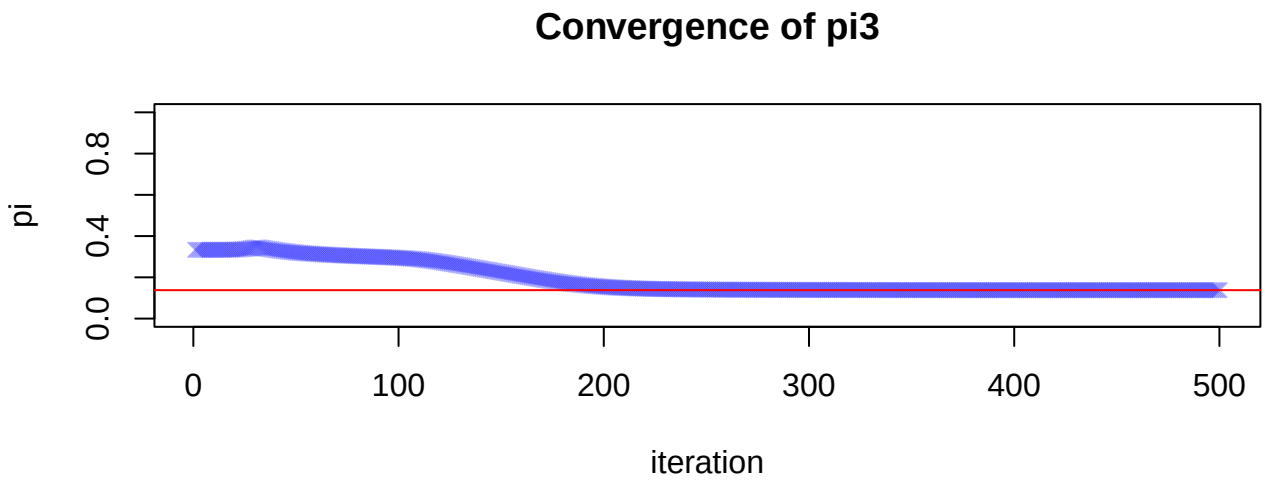
As such, Run 1 or Run 3 would be best given that they return the closest possible proportion relative to the intial observed values. Moving forward, Run 1 will be used.

Q6

```
# For each pi, plot convergence to final pi value
pi_labels <- c("pi1", "pi2", "pi3")
for (pi_lab in pi_labels) {
  plot(run_1[[pi_lab]],
    xlab = "iteration",
    ylab = "pi",
    pch = 4,
    col = rgb(0, 0, 1, 0.3),
    ylim = c(0, 1),
    main = sprintf("Convergence of %s", pi_lab)
  )
  abline(
    h = run_1[[pi_lab]][n.iter],
    col = "red"
  )
}
```







Given stable behaviour after about 200 iterations, the algorithm seems to have converged.

Q7

```
# Get mu and sigma from theta_hat of last run
pi_ests <- c(run_1$pi1[n.iter], run_1$pi2[n.iter], run_1$pi3[n.iter])
mu_ests <- run_1$theta[n.iter, 1:3]
sigma_ests <- run_1$theta[n.iter, 4:6]

# Build DF for summary table
mle_summary_df <- data.frame(rbind(pi_ests, mu_ests, sigma_ests),
  row.names = c("pi", "mu", "sigma")
)

knitr::kable(mle_summary_df,
  col.names = c("j = 1", "j = 2", "j = 3"),
  caption = "MLE of pi, mu, and sigma from Run 1"
)
```

Table 4: MLE of pi, mu, and sigma from Run 1

	j = 1	j = 2	j = 3
pi	0.5420971	0.3201059	0.137797
mu	3.5996788	4.6331208	5.568316
sigma	-1.0264512	-0.9550092	-1.270596

```
# Get mean of each species from data
mean_species_weights <- filtered_penguins_df %>%
  group_by(species) %>%
  summarise(weight = mean(body_mass_kg))
knitr::kable(mean_species_weights,
  col.names = c("Species", "Average Weight"),
  caption = "Average Weight (kg) by Species"
)
```

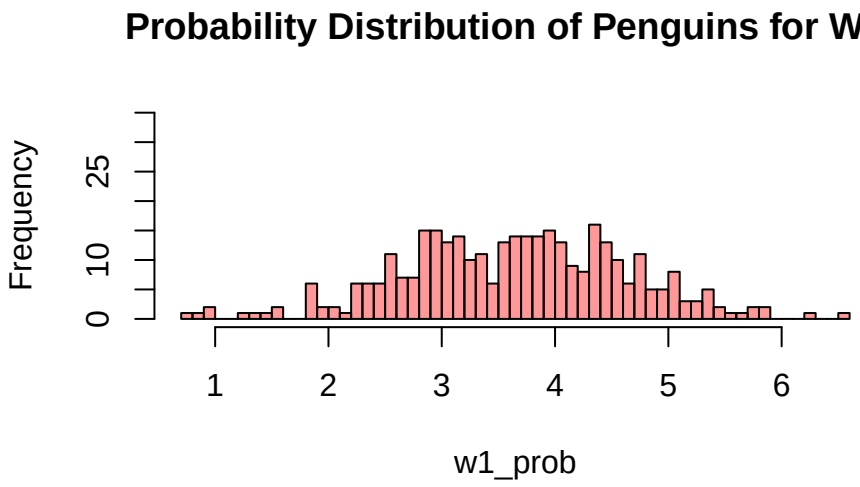
Table 5: Average Weight (kg) by Species

Species	Average Weight
Adelie	3.700662
Chinstrap	3.733088
Gentoo	5.076016

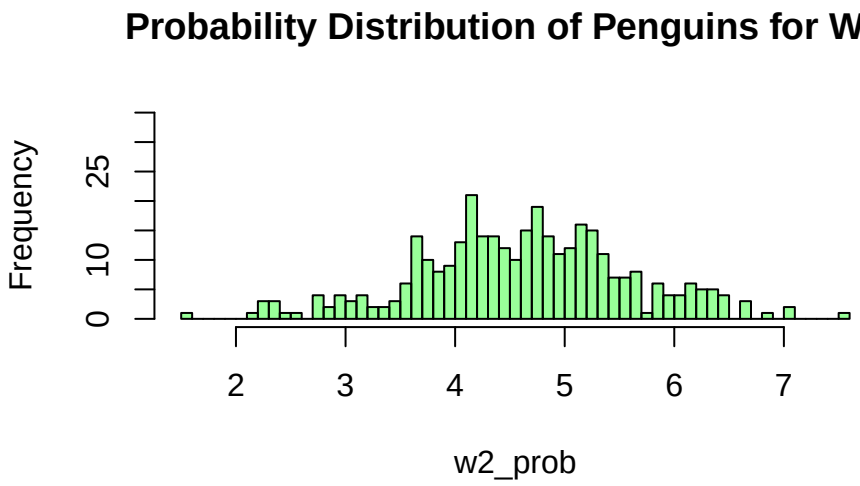
The weights of the components are not similar, with the first group (Adelie), representing the majority of penguins. The estimates of average weight for Adelie are quite close to the actual, but the other two species (Chinstrap and Gentoo), differ from the calculated averages by larger values.

Q8

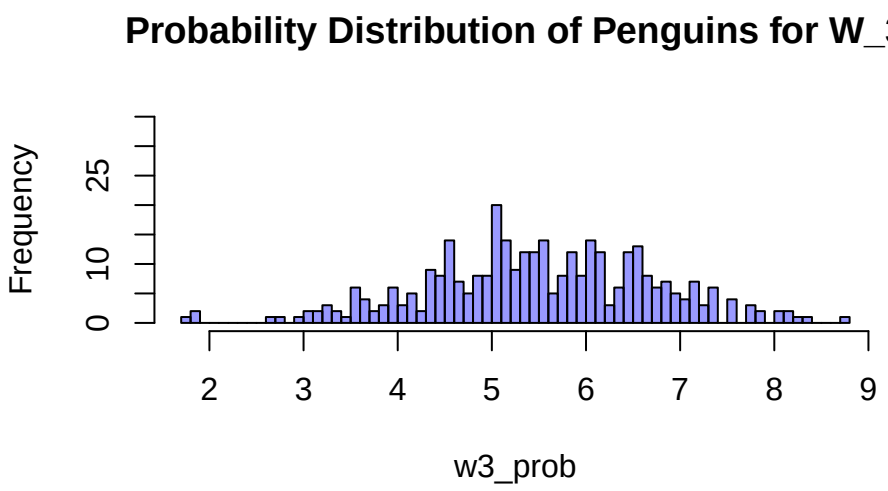
```
# Sample distributions using params defined in theta
w1_prob <- rnorm(n,
                 mean = run_1$theta[n.iter, 1],
                 sd = abs(run_1$theta[n.iter, 4]))
w2_prob <- rnorm(n,
                 mean = run_1$theta[n.iter, 2],
                 sd = abs(run_1$theta[n.iter, 5]))
w3_prob <- rnorm(n,
                 mean = run_1$theta[n.iter, 3],
                 sd = abs(run_1$theta[n.iter, 6]))

# Plot histograms for samples
hist(w1_prob,
  breaks = 50,
  col = rgb(1, 0, 0, 0.4),
  ylim = c(0, 35),
  main = "Probability Distribution of Penguins for W_1"
)
```




```
hist(w2_prob,
  breaks = 50,
  col = rgb(0, 1, 0, 0.4),
  # ylim = c(0, 300),
  ylim = c(0, 35),
  main = "Probability Distribution of Penguins for W_2"
)
```




```
hist(w3_prob,
  breaks = 50,
  col = rgb(0, 0, 1, 0.4),
  # ylim = c(0, 300),
  ylim = c(0, 35),
  main = "Probability Distribution of Penguins for W_3"
)
```



Q9

```
# Encode values to specific component index
predictedComponent <- ifelse(run_1$p1[n.iter, ] > 0.5, 1, 0) +
  ifelse(run_1$p2[n.iter, ] > 0.5, 2, 0) +
  ifelse(run_1$p3[n.iter, ] > 0.5, 3, 0)
print(predictedComponent)
```

```
##   [1] 1 1 1 1 1 1 2 1 2 1 1 1 1 2 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 2 1 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1
##  [75] 2 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 2 1 2 1 2 1 1 1 2 1 2 1 1 1 1 1 2 1 2
## [112] 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1
## [149] 1 1 1 2 3 2 3 3 2 2 2 2 2 2 3 2 3 2 3 1 3 2 3 3 2 2 2 2 2 1 3 2 3 3 2 2 3
## [186] 2 3 2 3 2 3 1 3 2 2 3 2 2 3 2 3 2 3 2 2 2 2 2 2 2 3 2 3 2 3 2 3 2 3 2 3 2
## [223] 2 2 2 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 2 2 2 2 3 2 3 2 3 2 3 2 3
## [260] 2 3 2 3 2 3 2 3 2 3 2 2 3 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
## [297] 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 2 1 2 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1
## [334] 1 1 1 1 1 1 1 1 1
```

Q10

```
# Pivot out sex against predicted component
sex_pc_jfd <- table(filtered_penguins_df$sex, predictedComponent)
knitr::kable(sex_pc_jfd,
  caption = "Joint Frequency Distribution of Sex vs. Predicted W_j Component"
)
```

Table 6: Joint Frequency Distribution of Sex vs. Predicted W_j Component

	1	2	3
female	109	56	0
male	74	45	49

```r
# Pivot out species against predicted components
species_pc_jfd <- table(filtered_penguins_df$species, predictedComponent)
knitr::kable(species_pc_jfd,
  caption = "Joint Frequency Distribution of Species vs. Predicted W_j Component"
)
```

Table 7: Joint Frequency Distribution of Species vs. Predicted W_j Component

	1	2	3
Adelie	124	27	0
Chinstrap	61	7	0
Gentoo	3	71	49

Q11

```r
# Get JDF of sex vs. species
actual_sex_species <- table(filtered_penguins_df$sex, filtered_penguins_df$species)
knitr::kable(actual_sex_species,
  caption = "Joing Frequency Distribution of Sex vs. Species"
)
```

Table 8: Joing Frequency Distribution of Sex vs. Species

	Adelie	Chinstrap	Gentoo
female	73	34	58
male	73	34	61

If a penguin is in PC1, it is most likely a female, as there is a 0.5956284 probability of female from PC1. If a penguin is in PC2, it is most likely a female, as there is a 0.5544554 probability of female from PC2. If a penguin is in PC3, it will be male, as there is a 1 probability of male from PC3 - in short there are no females in PC3.

These guesses would be moderately accurate for PC1 and PC2, but far from reality from PC3 given the actual sex vs. species frequency distribution above.

Q12

Based on the above JDF between PC and species, PC1 would align with Adelie, PC2 would align with Chinstrap, and PC3 would align with Gentoo