Lab 3: Parameters & Estimators

Graded Out of 35 Points

Jennifer Bradley

2025-02-18

We will again be using our Palmer Penguins data today for some problems, so let's load that package in now:

```
library(palmerpenguins)
```

```
Warning: package 'palmerpenguins' was built under R version 4.4.2
```

```
penguins <- penguins[penguins$species!="Chinstrap"
                     & !is.na(penguins$body_mass_g),]
```

1. Gauging Estimator Effectiveness

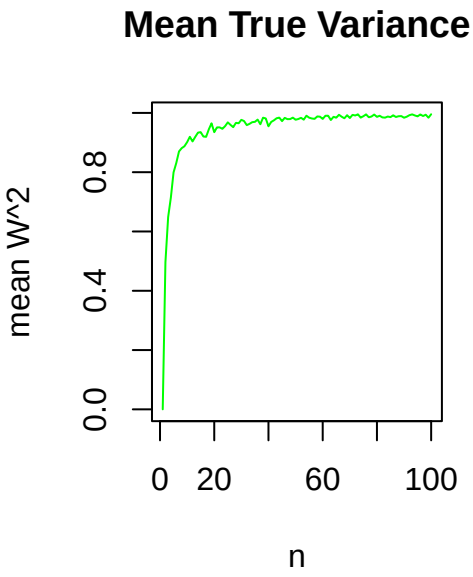
Problem 1.1 (5 points)

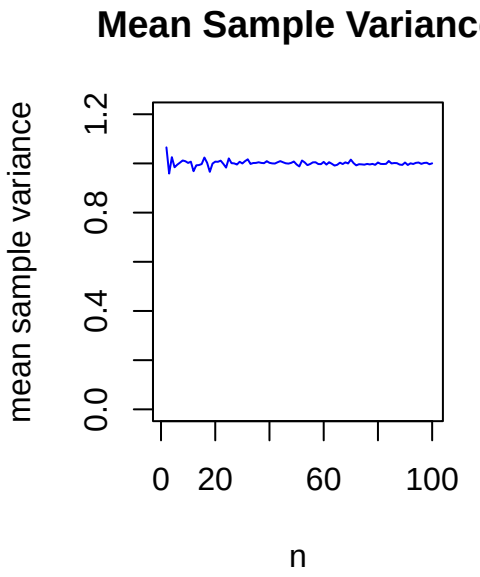
A. (1 Point) Write a function to calculate W^2 from a given sample

```
True_XVar <- function(Xset){
  Xrunning_sum <- 0
  Xmean <- mean(Xset)

  for (i in 1:length(Xset)) {
    Xrunning_sum <- Xrunning_sum + ((Xset[i] - Xmean)^2)
  }

    Xvar_result <- Xrunning_sum/length(Xset)

    }
```

B. (2 Points) Create two plots (using function `plot` with option `type="l"`) showing (1) how the mean (*i.e.*, expected value) of W^2 over 1000 samples of size n changes with increasing sample size (vary n from 1 to 100), for data sampled from a standard normal distribution, and (2) how the mean (*i.e.*, expected) sample variance changes with increasing sample size for the same distribution (using function `var`).

```
##: initializing the mean value of true var (W2), the mean value of sample
#var and distribution for the simulation.
True_Xvar_results <- c()
True_Xvar_means <- c()
Sample_Xvar_results <- c()
Sample_Xvar_means <- c()

##: Calculate the mean value of true var of random variable X 100 times and
#plot the distribution of means
for(i in 1:100){

  for(j in 1:1000){
      True_Xvar_results[j] <- True_XVar(rnorm(i))
          }
      True_Xvar_means[i] <- mean(True_Xvar_results)
  }


##: Calculate the mean value of sample var of random variable X
for(i in 1:100){

  for(j in 1:1000){
      Sample_Xvar_results[j] <- var(rnorm(i))
          }
      Sample_Xvar_means[i] <- mean(Sample_Xvar_results)
  }

##: Plot
par(mfrow = c(1,2))
plot(True_Xvar_means, type = "l", col = "green", x = 1:100, xlab = "n",
     ylab = "mean W^2", main = "Mean True Variance")
plot(Sample_Xvar_means, type = "l", col = "blue", x = 1:100,
     ylim = c(0.0,1.2), xlab = "n", ylab = "mean sample variance",
     main = "Mean Sample Variance")
```





C. (2 Points) Interpret your plots from part B. Do we see evidence of bias? Is this bias more pronounced at some values of n than others?

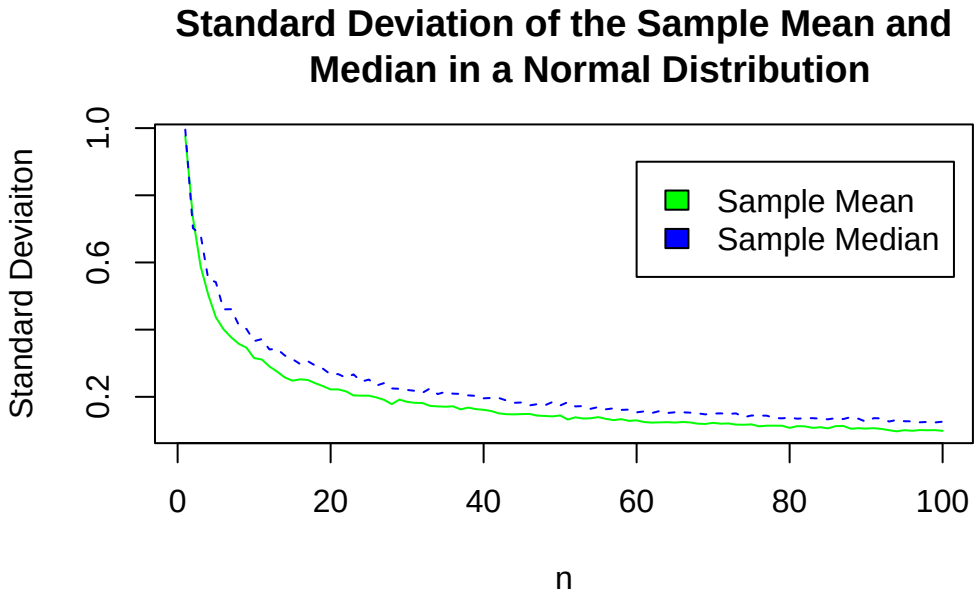
> **Note**
>
> There is evidence of bias in the mean of W^2 when n is small (specifically $1 < n < 20$), while the mean of sample variance shows no bias or significant variation in respect to n.

Problem 1.2 (5 points)

A. (3 Points) Create one plot (using function `plot` with option `type="l"`) showing (1) how the standard deviation of the sample mean (*i.e.* standard error) calculated from 1000 samples of size n changes with increasing sample size (vary n from 1 to 100), for data sampled from a standard normal distribution, **AND** (2) the same for the standard deviation of the sample median **on the same plot**.

```
Sample_Std_results <- c()
Sample_Std_means <- c()
Sample_Median_results <- c()
Sample_Std_medians <- c()

for(i in 1:100){
```

```
  for(j in 1:1000){
      Sample_Std_results[j] <- mean(rnorm(i))
  }

      Sample_Std_means[i] <- sd(Sample_Std_results)
  }


for(i in 1:100){

  for(j in 1:1000){
      Sample_Std_results[j] <- median(rnorm(i))
  }

      Sample_Std_medians[i] <- sd(Sample_Std_results)
  }

plot(Sample_Std_means, type = "l", col = "green", x = 1:100, xlab = "n",
     ylab = "Standard Deviaiton",
     main = "Standard Deviation of the Sample Mean and
     Median in a Normal Distribution")

points(Sample_Std_medians, type = "l", col = "blue", x = 1:100, xlab = "n",
       ylab = "std of the median", main = "Standard Deviation of the Sample
       Variance", lty=2)

legend(60,0.9, legend = c("Sample Mean", "Sample Median"),
       fill = c("green", "blue"))
```



B. (2 Points) Interpret your plot from part A. What does this plot tell us about the consistency and efficiency of these two estimators when applied to normally distributed data?

> **Note**
>
> When applied to a normal distribution, the standard deviation of both the mean and median becomes much smaller as n increases from 1 to 100. This implies that when using the standard deviation of the mean and median a higher sample size should be considered (based on the plot above, I would recommend no less than n = 20) for a high consistency. The std of the sample mean decreases faster than the std of the sample median as n increases, which means that the sample mean can be used as an estimator at slightly smaller n values.

2. Maximum Likelihood Estimation

Problem 2.1 (10 Points)

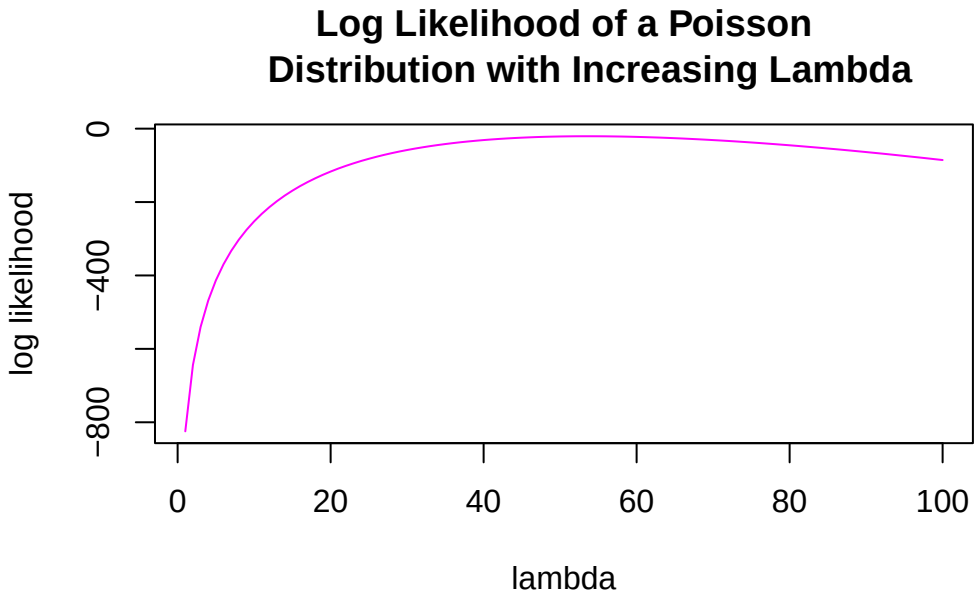
```
burst_data <- read.csv("https://raw.githubusercontent.com/jlw-ecoevo/jlw-ecoevo.github.io/re
```

A. (2 points) First, write an R function that takes a vector of data N and a value for the rate parameter λ and calculates the **log likelihood** of that data.

```r
log_likelihood_poisson <- function(Pset, max_rate){
  LLP_Results <- c()
  Log_Prob_Density <- c()
  for(lambda_value in 1:max_rate){

    for(observation_index in 1:length(Pset)){
      Log_Prob_Density[observation_index] <-
        log(dpois(Pset[observation_index], lambda_value))

    }
    LLP_Results[lambda_value] <- sum(Log_Prob_Density)
  }
  return(LLP_Results)
}
```

B. (2 points) Using only the first 5 observations in `burst_data` (`burst_data$NumPhage[1:5]`), use your function from Part A to calculate the log likelihood for a range of λ from 1 to 50. Plot how the log likelihood changes with lambda.

```r
LLP <- log_likelihood_poisson(burst_data$NumPhage[1:5], 100)

plot(x=c(1:100), y=LLP, type = "l", col = "magenta",
     main = "Log Likelihood of a Poisson
     Distribution with Increasing Lambda", xlab = "lambda",
     ylab = "log likelihood")
```



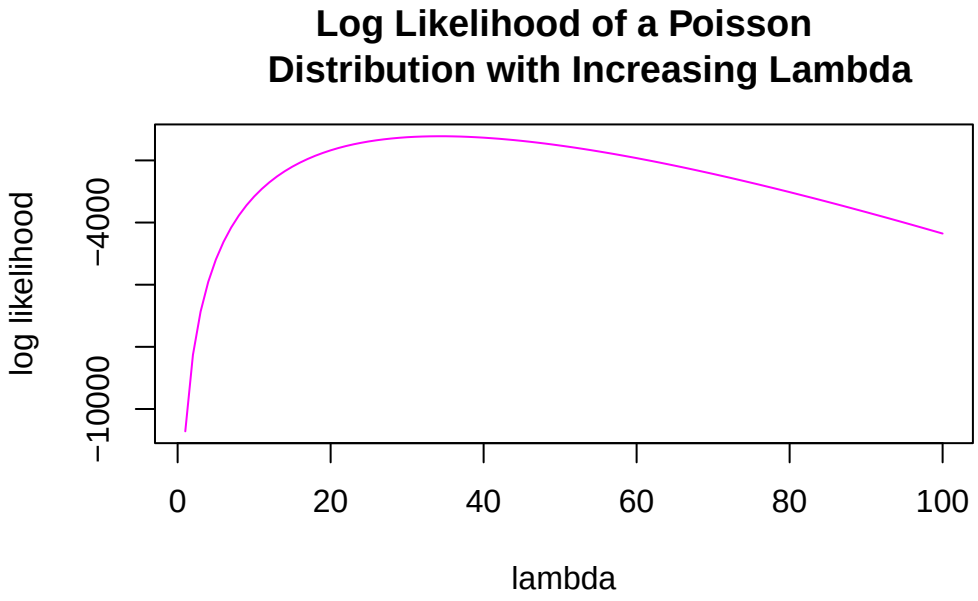
C. (2 points) What is the approximate MLE for λ using your result from part B? Try using the `which.max` function but be careful how you use it.

```
which.max(LLP)
```

```
[1] 54
```

D. (2 points) Repeat Parts B-C using the full dataset. Do you notice anything different about your plot? Discuss.

```
##: Part B
LLPFull <- log_likelihood_poisson(burst_data$NumPhage, 100)

plot(x=c(1:100), y=LLPFull, type = "l", col = "magenta",
     main = "Log Likelihood of a Poisson
     Distribution with Increasing Lambda", xlab = "lambda",
     ylab = "log likelihood")
```




```
##: Part C

which.max(LLPFull)
```

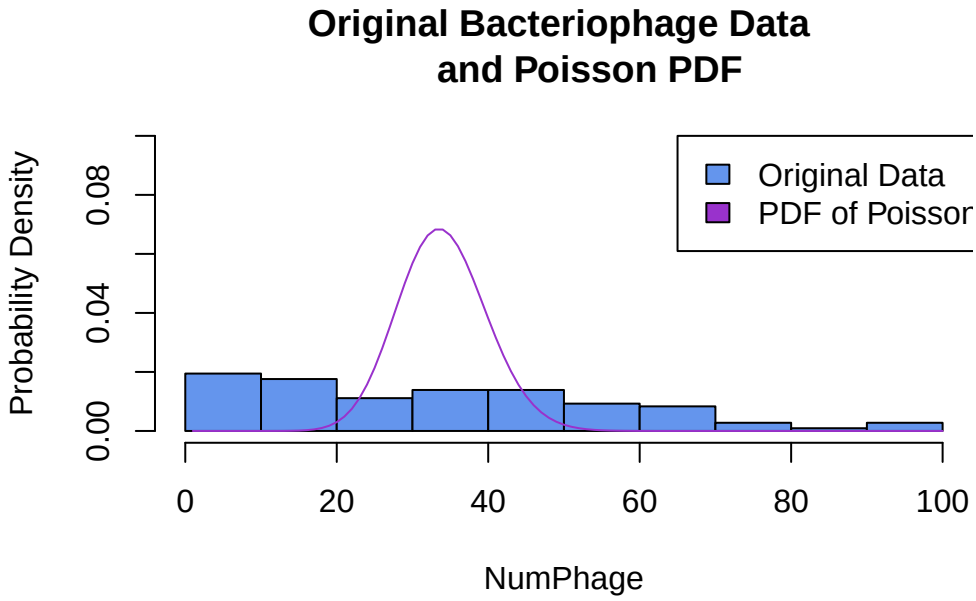
```
[1] 34
```

> **Note**
>
> The log likelihood as lambda increases past ~40 decreases at a faster rate in the full data set than in the partial data set. The MLE for the partial data set is larger than the MLE for the full data set. It is possible that the first 5 observations that were sampled were not done in a truly random fashion, or that observations are not independent of each other.
> (smaller dataset <- less representative, 1st 5 may have been biased)

E. (2 points) Now, plot your MLE solution against your original data. It should be apparent from this plot that the Poisson model is a poor fit for our data. Discuss some reasons why this might be (briefly).

```
sorted_burst_data <- sort(burst_data$NumPhage)
hist(sorted_burst_data, freq = FALSE, main = "Original Bacteriophage Data
     and Poisson PDF", ylab = "Probability Density", xlab = "NumPhage" ,
```

```
     ylim = c(0,0.1), col = "cornflowerblue")

points(dpois(1:100, 34), type = "l", col = "darkorchid")
legend(65,0.1, legend = c("Original Data", "PDF of Poisson Dist."),
       fill = c("cornflowerblue", "darkorchid"))
```



> **Note**
>
> The Poisson distribution could be a poor fit for this data set because the rate may be changing as more cells are lysed, and the rate remains constant for all sizes of n in a Poisson distribution.

Problem 2.2 (10 Points)

A. (2 Points) Write a function to calculate the **likelihood** for a single observation y_i of body mass that takes y_i, μ_1, μ_2, σ_1, σ_2, and p as inputs. Because of what we will be doing later on in this question, we want our function to take our parameters as a single vector with named entries `mu1`, `mu1`, `sd1`, `sd2`, and `p` so that you can reference them with `parameters["mu1"]` (and similar). Your function should take the following general form

```
params <- c("mu1", "mu2", "sd1", "sd2", "p")
```

```r
likelihoodSingleObservation <- function(params,yi){

  yi_likelihood = ((params["p"] * dnorm(yi, params["mu1"],
  params["sd1"])) + ((1 - params["p"]) * dnorm(yi, params["mu2"],
  params["sd2"])))
  return(yi_likelihood)
}
```

B. (2 Points) Write a function to calculate the **log likelihood** for a vector of observations $y = \{y_1, y_2, ..., y_n\}$ and given parameter vector `parameters`. This function will need to call the function you wrote in Part A (yes, you can call a function inside of another function). Your function should take the following general form:

```r
y <- c()

likelihoodAllObservations <- function(params,y){

  if(params["p"] < 0 |
     params["p"] > 1 |
     params["sd1"] < 0 |
     params["sd2"] < 0){
    return(-Inf)
  } else {

    y_log_likelihood = sum(log(likelihoodSingleObservation(params, y)))

    return(y_log_likelihood)
  }
}
```

C. (0 Points) Now we want to use this function to fit parameters to our penguin data. Unlike in Problem 2.1, just plotting the likelihoods won't work because we are dealing with so many parameters here. Instead we will use a function for numerical optimization that will search possible parameter combinations, score them using our likelihood function and data, and then return the best result. Run the following code and report your MLE for your parameters (note how we removed NA values prior to estimation):

```r
parameterInitialVals = c(mu1 = 3500, sd1 = 450, mu2 = 5000, sd2 = 450,
                         p = 0.5)
maximum = optim(par = parameterInitialVals,
                fn = likelihoodAllObservations, y = penguins$body_mass_g,
control = list(fnscale = -1))
MLE = maximum$par
print(paste0("My Maximum Likelihood Estimate for ",names(MLE),": ", MLE))
```

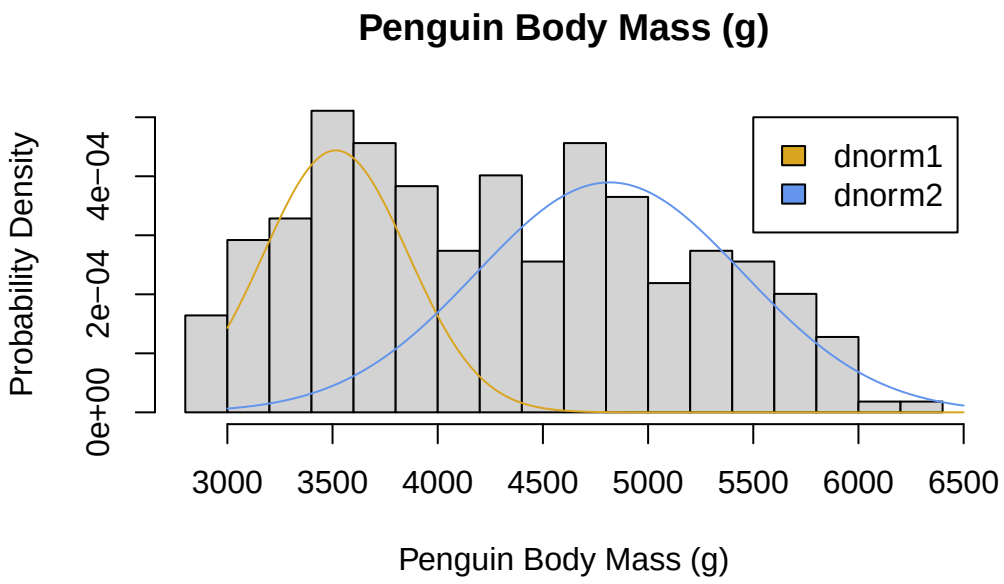
```
[1] "My Maximum Likelihood Estimate for mu1: 3515.6665385862"
[2] "My Maximum Likelihood Estimate for sd1: 342.703635105696"
[3] "My Maximum Likelihood Estimate for mu2: 4817.2426968284"
[4] "My Maximum Likelihood Estimate for sd2: 633.64435427738"
[5] "My Maximum Likelihood Estimate for p: 0.381251453066743"
```

Note: numerical optimizers are not guaranteed to find the global best solution for a problem and might get stuck in local (suboptimal) solutions. There are many ways to deal with this that we won't get into in this class, but suffice to say that your solution can depend heavily on the initial parameter values you provide to start the search process when using these optimizers.

D. (2 Points) Now, plot your MLE solution against your original data:

- First, make a histogram of the original penguin data using the `freq=FALSE` option to get a density plot. For best results, use options `breaks=15` and `ylim=c(0,5e-4)` (I got these values just by fiddling with the plot until it looked nice).
- Then, add lines to your plot representing the two fitted normal distributions you parameterized with densities scaled by p and $(1-p)$. In other words, plot $pf(y|\mu_1, \sigma_1)$ and $(1-p)*f(y|\mu_2, \sigma_2)$. Recall that you can obtain the pdf of a normal distribution using the function `dnorm` (just plug in your fitted parameters). You can add lines to an existing plot using the `points` function with argument `type="l"`. Please make the two lines different colors so that they are easy to distinguish.

```
hist(penguins$body_mass_g, freq = FALSE, breaks = 15, ylim = c(0,5e-4),
     main = "Penguin Body Mass (g)", xlab = "Penguin Body Mass (g)",
     ylab = "Probability Density")

points(y = (0.381251453066743 * dnorm(3000:6500, 3515.6665385862,
      342.703635105696)), x = 3000:6500, type = "l", col = "goldenrod")

points(y = ((1 - 0.381251453066743) * dnorm(3000:6500, 4817.2426968284,
      633.64435427738)), x = 3000:6500, type = "l", col = "cornflowerblue")

legend(5500,5e-04, legend = c("dnorm1", "dnorm2"), fill = c("goldenrod",
      "cornflowerblue"))
```



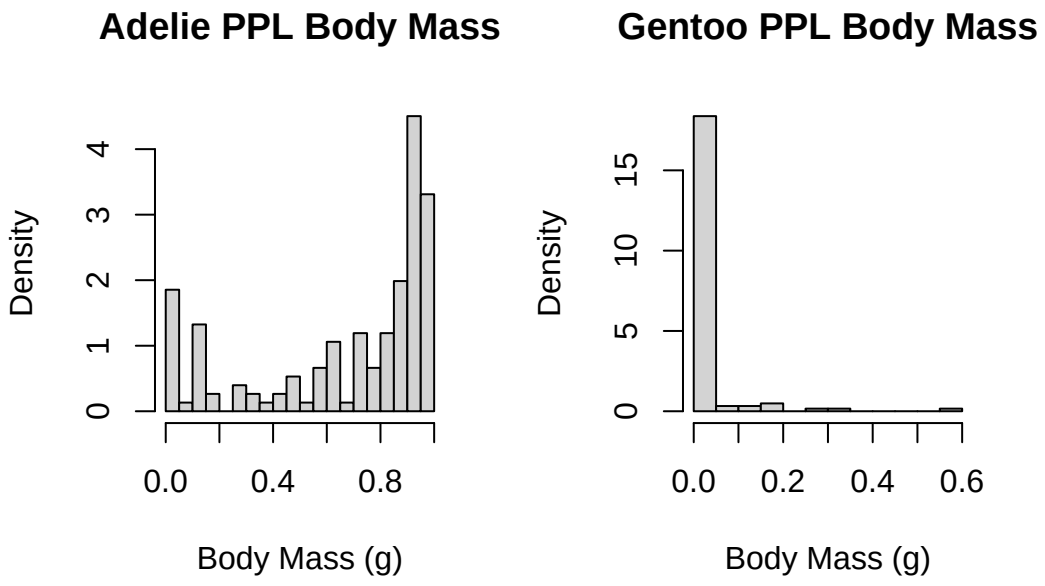
E. (2 Points) Now, we want to be able to, in a principled way, assign individual Penguins to our two groups. This is a common problem when clustering data, and our Gaussian Mixture model is essentially an approach for performing such clustering. What we want is the "posterior predictive probability" that penguin belongs to distribution 1 (or, one minus that probability that is belongs to distribution 2). The word "posterior" means it comes after incorporating information from the original data, and "predictive" because we want to predict group membership. We can write this probability as:

$$P(y \text{ from } X_1 | y, \text{ parameters}) = \frac{pf(y|\mu_1, \sigma_1)}{pf(y|\mu_1, \sigma_1) + (1-p)f(y|\mu_2, \sigma_2)}$$

Write a function to calculate this probability, apply it to your penguin data using your parameter estimates from Part C, and then plot two histograms depicting the distribution of this probability for the two penguin species (Adelie and Gentoo) in your dataset. Note that the denominator in this equation is just the output of the function your wrote in Part A.

```
a_penguins <- subset(penguins, penguins$species=="Adelie")
g_penguins <- subset(penguins, penguins$species=="Gentoo")

param_estimates <- c(3515.6665385862, 4817.2426968284, 342.703635105696,
                     633.64435427738, 0.381251453066743)
```

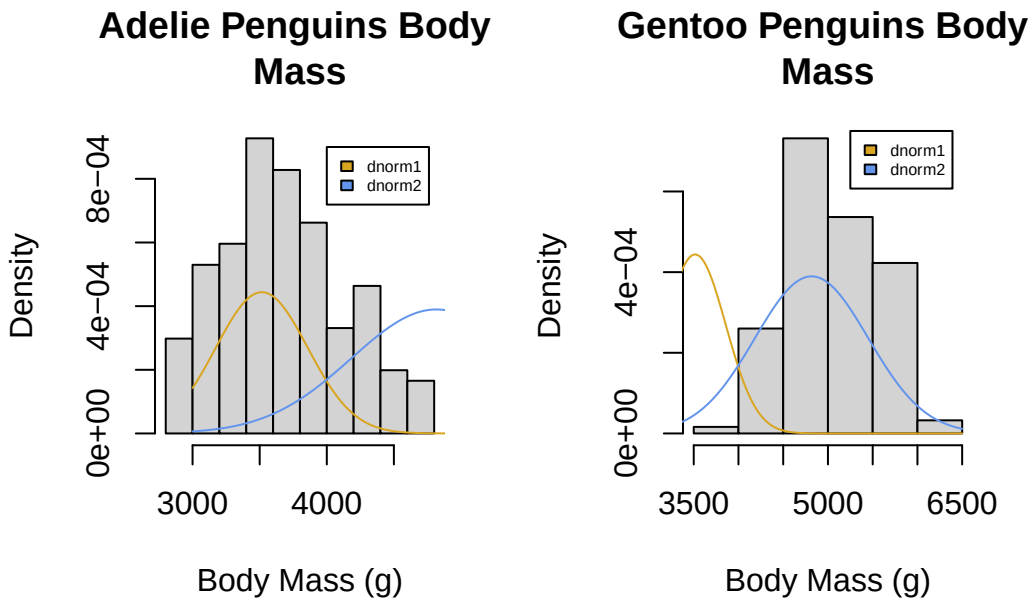
```r
post_pred_likelihood <- function(param_estimates,ypp){

  ypp_likelihood = ((param_estimates[5] * dnorm(ypp, param_estimates[1],
                   param_estimates[3]))) / ((param_estimates[5] * dnorm(ypp,
                   param_estimates[5]) * dnorm(ypp, param_estimates[2],
                   param_estimates[4])))

  return(ypp_likelihood)
}

ppl_adelie <- post_pred_likelihood(param_estimates, a_penguins$body_mass_g)
ppl_gentoo <- post_pred_likelihood(param_estimates, g_penguins$body_mass_g)

par(mfrow=c(1,2))
hist(ppl_adelie, freq = FALSE, breaks = 15, xlab = "Body Mass (g)",
     main = "Adelie PPL Body Mass")
hist(ppl_gentoo, freq = FALSE, breaks = 10, xlab = "Body Mass (g)",
     main = "Gentoo PPL Body Mass")
```



F. (1 Point) Now I want you to make two plots nearly identical to Part D, but for each individual species in the dataset (Adelie or Gentoo). In this case, the only difference is that you should subset the data in your histogram to each species for their respective

plot, still plotting the same MLE solution on top of these histograms. I am **NOT** asking you to re-fit your MLE, just change your plotting code.

```
par(mfrow=c(1,2))
hist(a_penguins$body_mass_g, freq = FALSE,
     main = "Adelie Penguins Body \n Mass ", xlab = "Body Mass (g)")
points(y = (0.381251453066743 * dnorm(3000:6500, 3515.6665385862,
      342.703635105696)), x = 3000:6500, type = "l", col = "goldenrod")
points(y = ((1 - 0.381251453066743) * dnorm(3000:6500, 4817.2426968284,
      633.64435427738)), x = 3000:6500, type = "l", col = "cornflowerblue")
legend(4000,9e-04, cex = 0.5, legend = c("dnorm1", "dnorm2"),
       fill = c("goldenrod", "cornflowerblue"))

hist(g_penguins$body_mass_g, freq = FALSE,
     main = "Gentoo Penguins Body \n Mass ", xlab = "Body Mass (g)")
points(y = (0.381251453066743 * dnorm(3000:6500, 3515.6665385862,
      342.703635105696)), x = 3000:6500, type = "l", col = "goldenrod")
points(y = ((1 - 0.381251453066743) * dnorm(3000:6500, 4817.2426968284,
      633.64435427738)), x = 3000:6500, type = "l", col = "cornflowerblue")
legend(5250,7.5e-04, cex = 0.5, legend = c("dnorm1", "dnorm2"),
       fill = c("goldenrod", "cornflowerblue"))
```



G. (1 Point) Provide a brief interpretation of your results from Parts E and F. Why might this have been a particularly useful analysis if we didn't know our species labels beforehand?

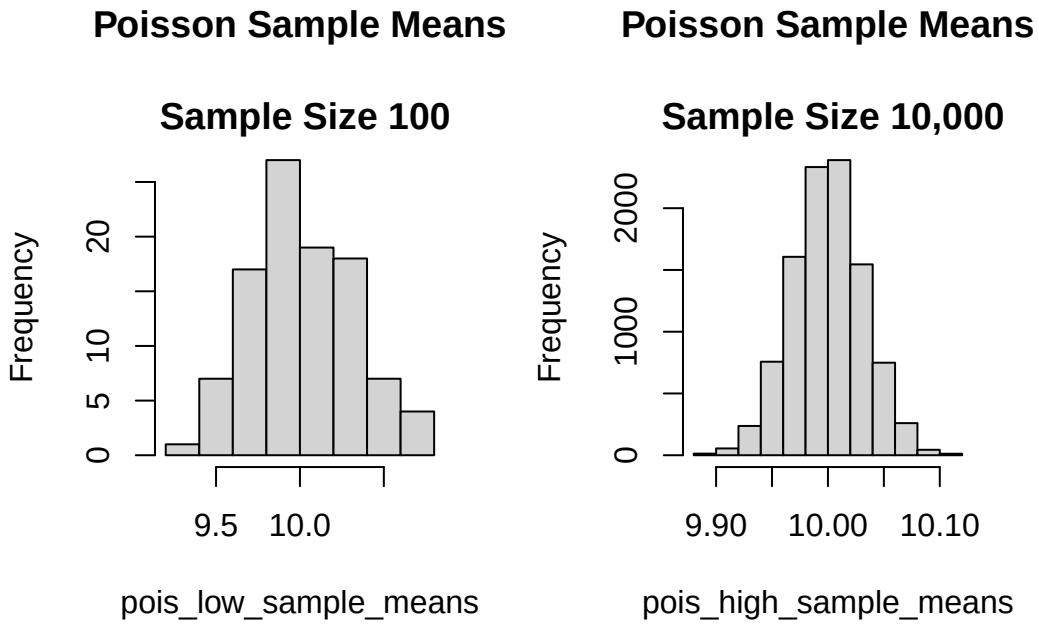
> **Note**
>
> Interpretation:
> This would be a useful analysis had we not known the species subsets beforehand because the bimodal distribution of the data suggests a separation in processes that lead to their distributions and/or implies that there are two distinct groups in the whole set of samples.

3. Central Limit Theorem

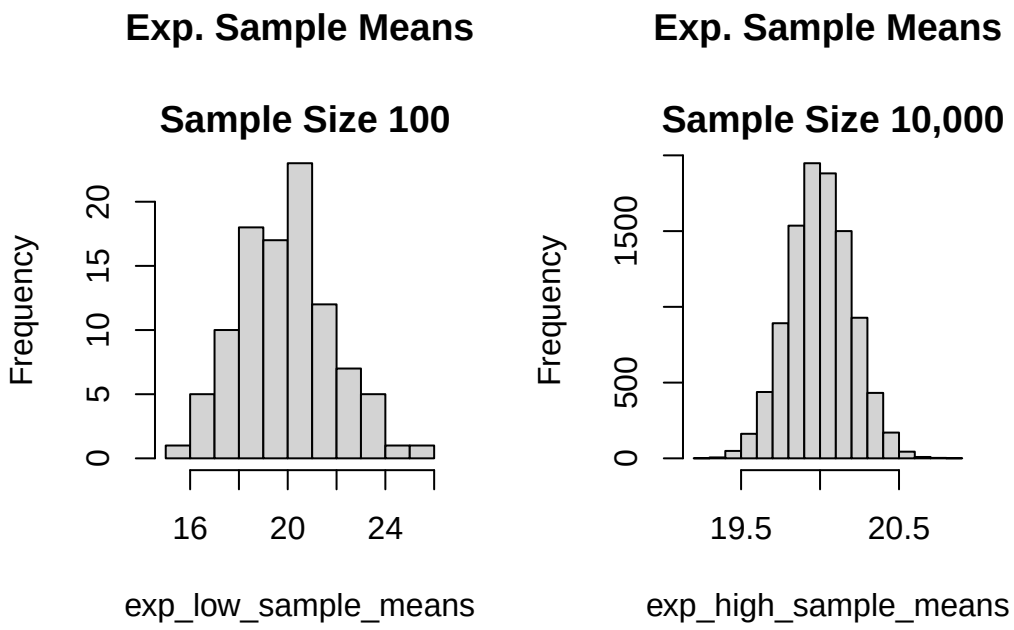
Problem 3.1 (5 Points)

```
##///// High Sample Values
exp_high_sample_means <- c()

exp_high_sample_size = 10000
for(i in 1:exp_high_sample_size){

  exp_high_sample_means[i] <- mean(rexp(exp_high_sample_size, 0.05))
  }

pois_high_sample_means <- c()

pois_high_sample_size = 10000
for(i in 1:pois_high_sample_size){
  pois_high_sample_means[i] <- mean(rpois(pois_high_sample_size, 10))
}

##///// Low Sample Values
exp_low_sample_means <- c()

exp_low_sample_size = 100
for(i in 1:exp_low_sample_size){

  exp_low_sample_means[i] <- mean(rexp(exp_low_sample_size, 0.05))
  }

pois_low_sample_means <- c()
```

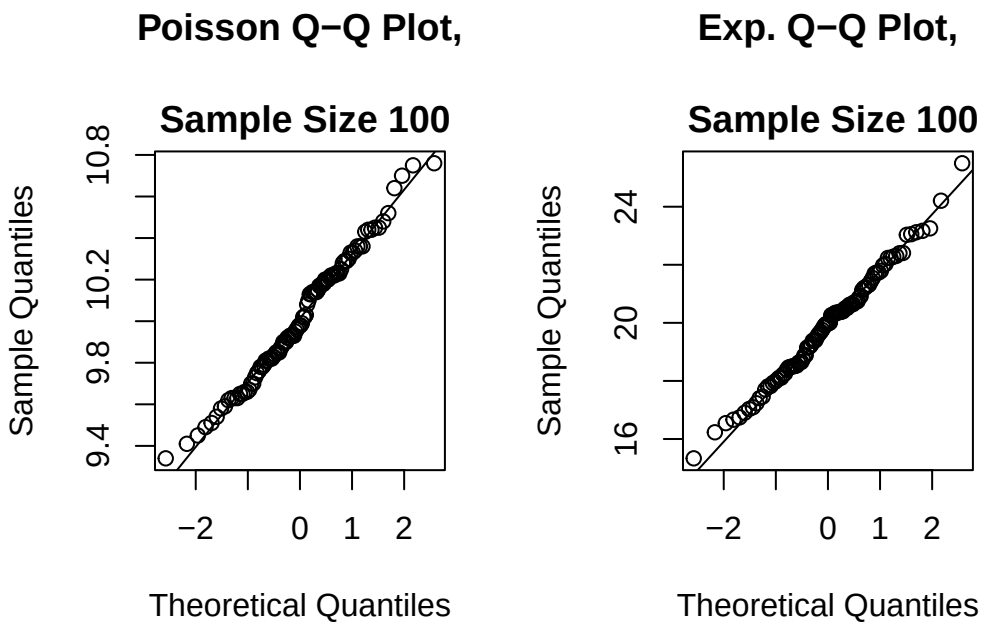
```r
pois_low_sample_size = 100
for(i in 1:pois_low_sample_size){
  pois_low_sample_means[i] <- mean(rpois(pois_low_sample_size, 10))
}

par(mfrow=c(1,2))
hist(pois_low_sample_means,
main = "Poisson Sample Means
\n Sample Size 100")
hist(pois_high_sample_means,
main = "Poisson Sample Means
\n Sample Size 10,000")
```



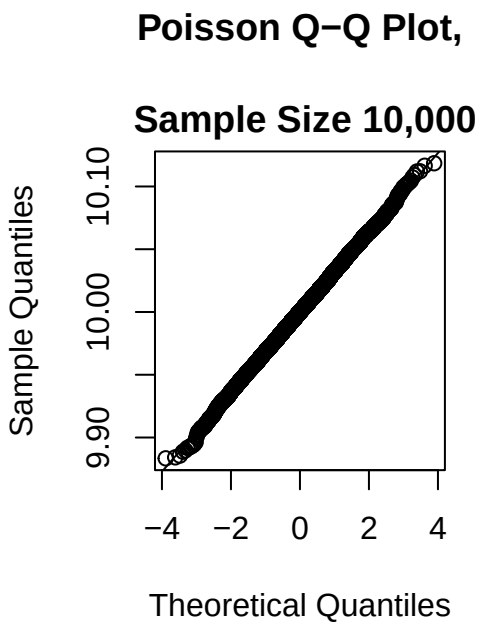

```r
par(mfrow=c(1,2))
hist(exp_low_sample_means,
main = "Exp. Sample Means
\n Sample Size 100")
hist(exp_high_sample_means,
main = "Exp. Sample Means
\n Sample Size 10,000")
```

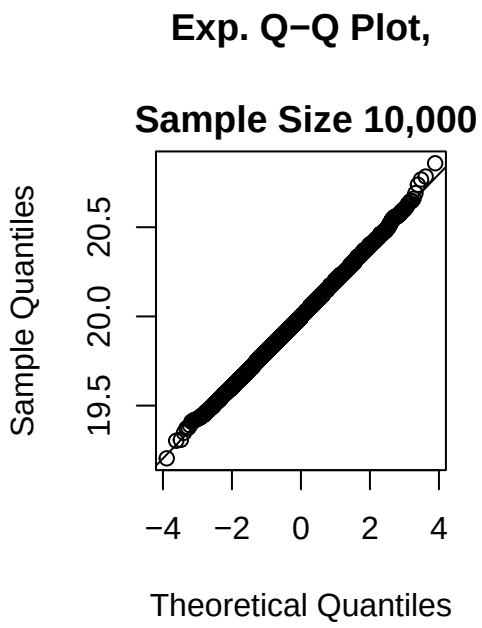



```
par(mfrow=c(1,2))
qqnorm(pois_low_sample_means,
main = "Poisson Q-Q Plot,
\n Sample Size 100")
points(qqline(pois_low_sample_means))

qqnorm(exp_low_sample_means,
main = "Exp. Q-Q Plot,
\n Sample Size 100")
points(qqline(exp_low_sample_means))
```




```r
par(mfrow=c(1,2))
qqnorm(pois_high_sample_means,
main = "Poisson Q-Q Plot,
\n Sample Size 10,000")
points(qqline(pois_high_sample_means))

qqnorm(exp_high_sample_means,
main = "Exp. Q-Q Plot,
\n Sample Size 10,000")
points(qqline(exp_high_sample_means))
```





Note

To demonstrate the Central Limit Theorem, I first generated 4 distributions, and used a for loop adding the mean of a sample of increasing size until a specified parameter (100 for small sample size, and 10,000 for high sample size) to a vector of means. The following were plotted as a histogram of low/high sample means and Q-Q Plots:

- 1 discrete Poisson distribution with a small sampling size of 100 (pois_low_sample_means)
- 1 continuous exponential distribution with a small sampling size of 100 (exp_low_sample_means)
- 1 discrete Poisson distribution with a high sampling size of 10,000 (pois_high_sample_means)
- 1 continuous exponential distribution with a high sampling size of 10,000 (exp_high_sample_means)

The results show that for both exponential and poisson distributions as the sampling size increased from 100 to 10,000:

- The distribution of the sample means became more normalized on the histograms.
- The Q-Q plot of the distributions better matched the control qqline

> This supports the Central Limit Theorem's core concept that the distribution of sample means in almost any distribution (no matter how skewed) will approach normal form at high sampling sizes.

4. List of All Problems (Above, 35 Points)

- 1.1(5 Points)
- 1.2 (5 Points)
- 2.1 (10 Points)
- 2.2 (10 Points)
- 3.1 (5 Points)

5. Acknowledgements

Problems 2.1 and 2.2 adapted from problem sets shared by Dr. Philip L.F. Johnson. Problem 3.1 adapted from problem sets shared by Dr. Heather Lynch.