Week 6: Mixture Models

PLSC 40502 - Statistical Models

Review

Previously

- **Survey weighting**
  - Learning about **population** parameters from non-representative samples.
  - Two key components: **selection/non-response model** and a **measurement model** for the population targets.
  - **Calibration** weighting - find weights that satisfy a set of moment condition subject to a loss function.
- **Multilevel Regression and Post-stratification**
  - Can we estimate population proportions for small areas with a national survey?
  - **Yes** - if we pool information from other units.

This week

- **"Exploratory" modeling**
  - Unsupervised "clustering" algorithms
  - K-means clustering
  - Bayesian mixture models
  - Obtaining MLEs or posterior modes via Expectation-Maximization
- **Topic modeling**
  - A mixture model for text:
  - Documents have a topic **distribution**
  - Define a DGP for each word as a function of the document topic distribution and a topic-specific word distribution.
  - Flexible way of representing documents via a lower-dimensional summary.

Finite Mixture Models

Finite Mixture Models

- In conventional **regression** models, we assume a known distribution for the outcome given a set of covariates
  - The covariates could be something like a discrete "group" label (e.g. $X_i = x$)
- In this setting, conditional on X_i, we would model each Y_i with a known distribution (e.g. Normal)

$$Y_i | X_i = x \sim \text{Normal}(\mu_x, \sigma_x^2)$$

μ_x and σ_x^2 are the mean and variance parameters associated with the group $X_i = x$.

- Now imagine that the group indicators are **not observed** but instead rather **latent parameters**
  - We can do inference on these parameters conditional on the observed data and the model.

Finite Mixture Models

- In a **finite** mixture model, we assume that the outcome distribution for each observation i is governed by some discrete latent indicator $z_i \in \{1, 2, \ldots, K\}$
  - $z_1, z_2, \ldots, z_N$ are i.i.d. $\sim \text{Categorical}(\pi)$
- Conditional on the latent variable $z_i = k$, the outcome distribution is known.
- For example, in a **gaussian** finite mixture model, we assume

$$Y_i | z_i = k \sim \text{Normal}(\mu_k, \sigma_k^2)$$

- But unconditionally, Y_i has a **mixture** distribution in that its density is a weighted average of the component densities.
- In a **gaussian** finite mixture model, we have:

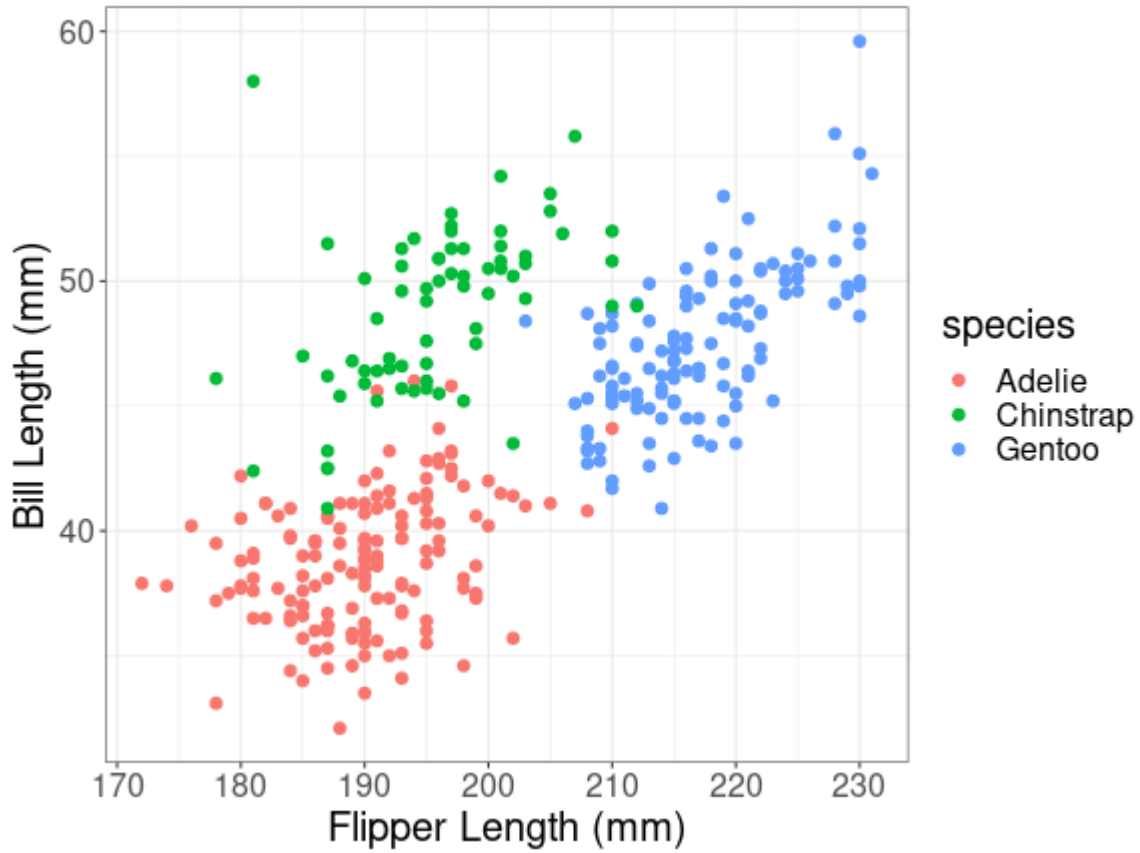
$$Y_i \sim \sum_{k=1}^{K} \pi_k \times \text{Normal}(\mu_k, \sigma_k^2)$$

where $\pi_k = Pr(z_i = k)$

- Our goal is to estimate the cluster means/variances for each cluster and the mixing proportions using likelihood inference.
  - Put priors on the mean/variance parameters and π to make it fully bayesian.

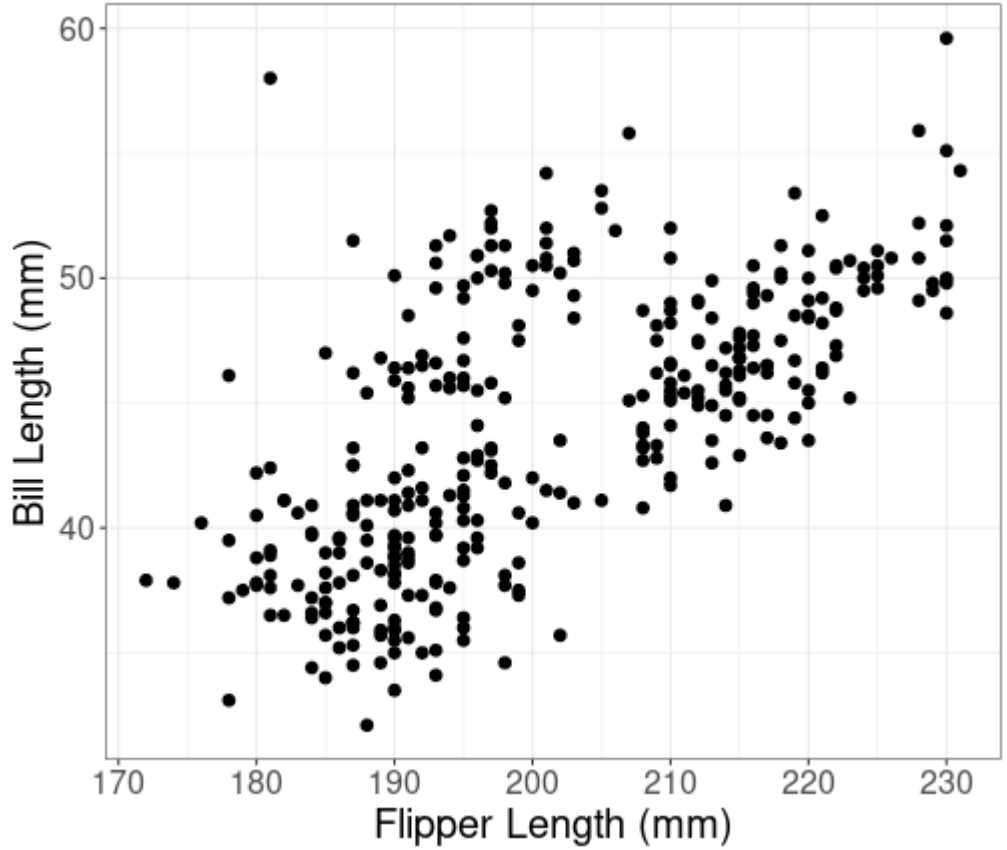
Penguins!

- For our running example, we'll look at the `palmerpenguins` dataset which contains measurements on three species of penguins in the Palmer Archipelago in Antarctica



Clustering

- Suppose we didn't observe the labels, could we still recover the latent "clusters" in the data?

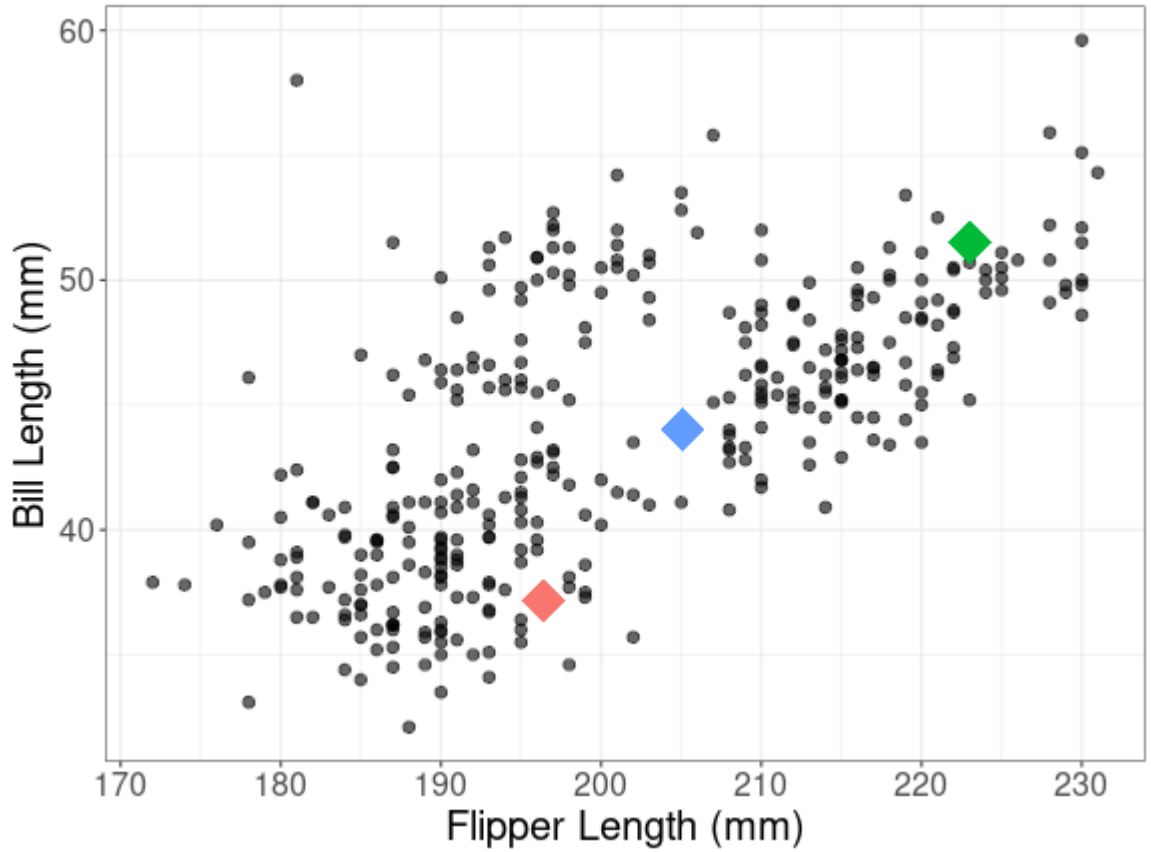


K-means

- A simple algorithm for generating clusters that has no underlying probabilistic model is the **K-means algorithm**
  - Straightforward to implement and still surprisingly popular and effective for a "first cut" at the data.
- **K-means algorithm**
  1. Initialize K distinct clusters by randomly assigning points to one of the k groups
  2. Calculate μ_k as the mean of Y_i in observations in cluster k.
  3. Reassign each point to the cluster k that has the smallest distance between Y_i and μ_k.
  4. Repeat 2-3 until no points change their assignments.

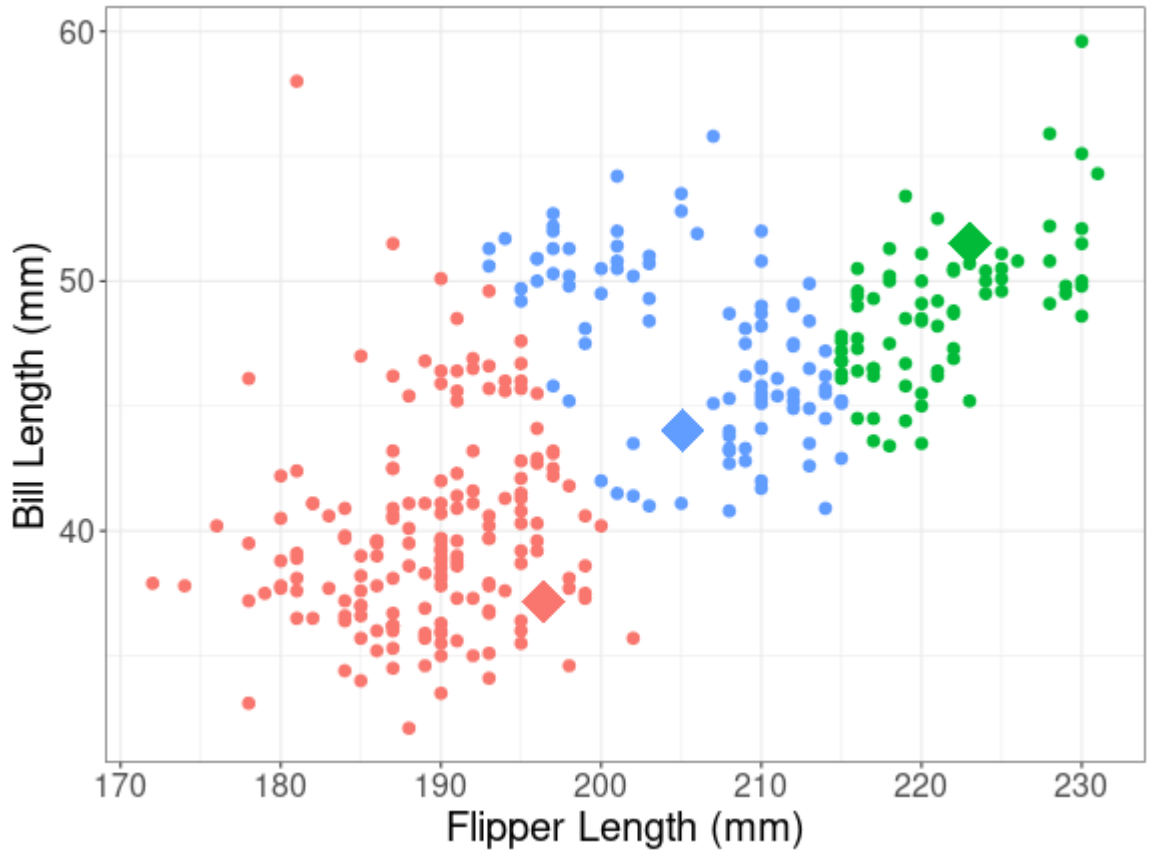
K-means example

- Let $K = 3$ and randomly pick three initial μ_k



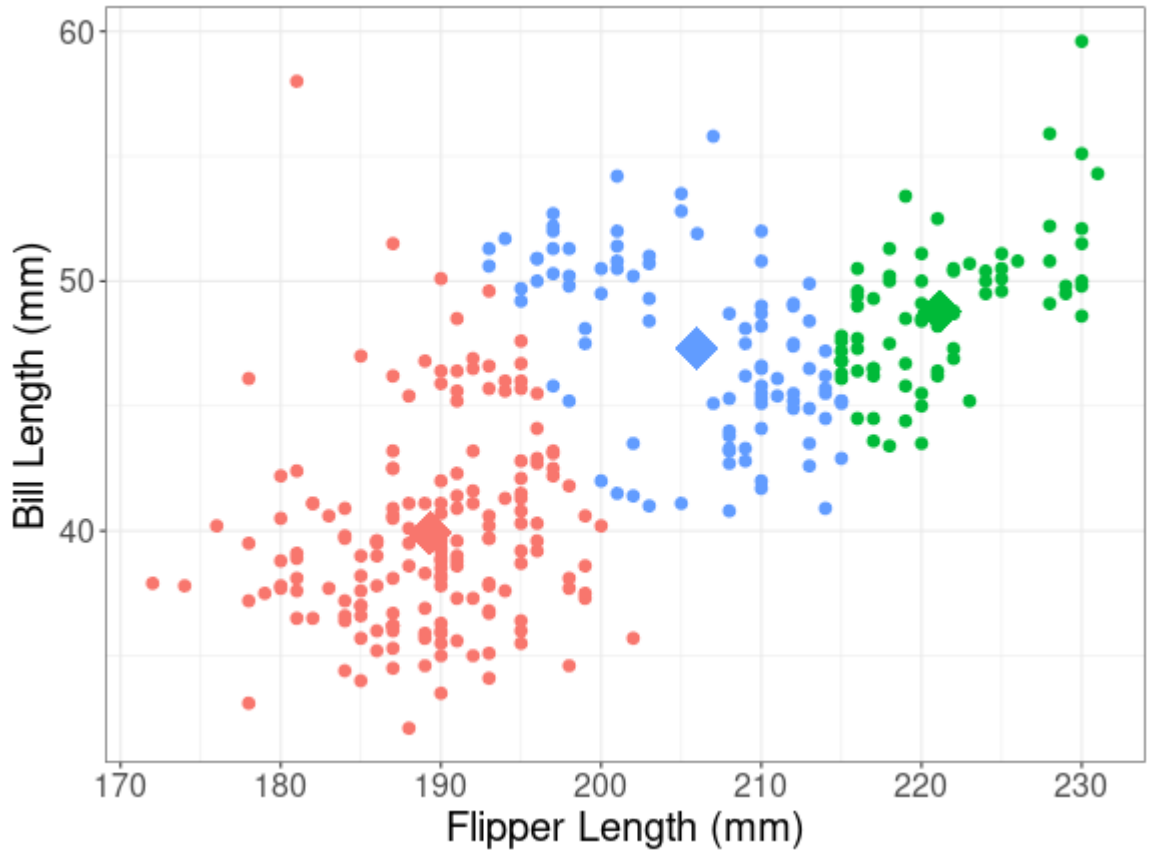
K-means example

- Assign each penguin to the cluster with the nearest cluster centroid.



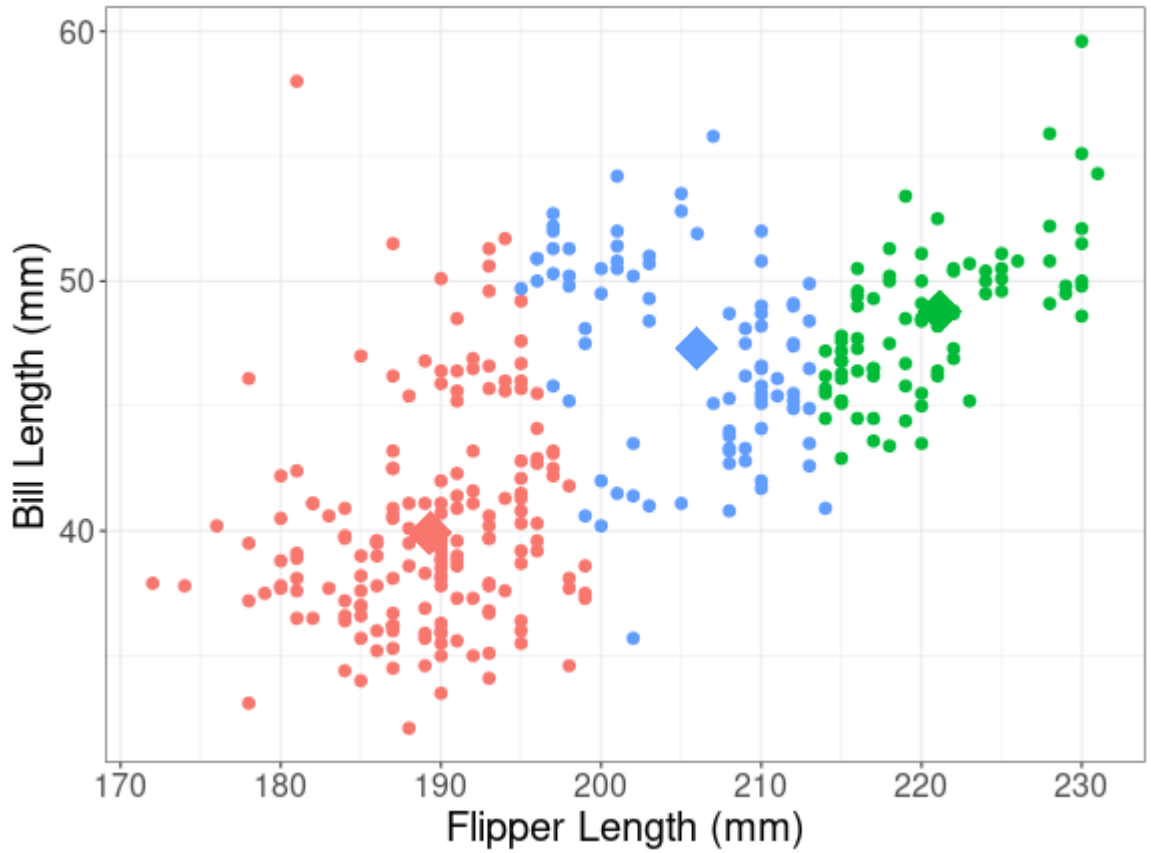
K-means example

- Recompute the cluster means



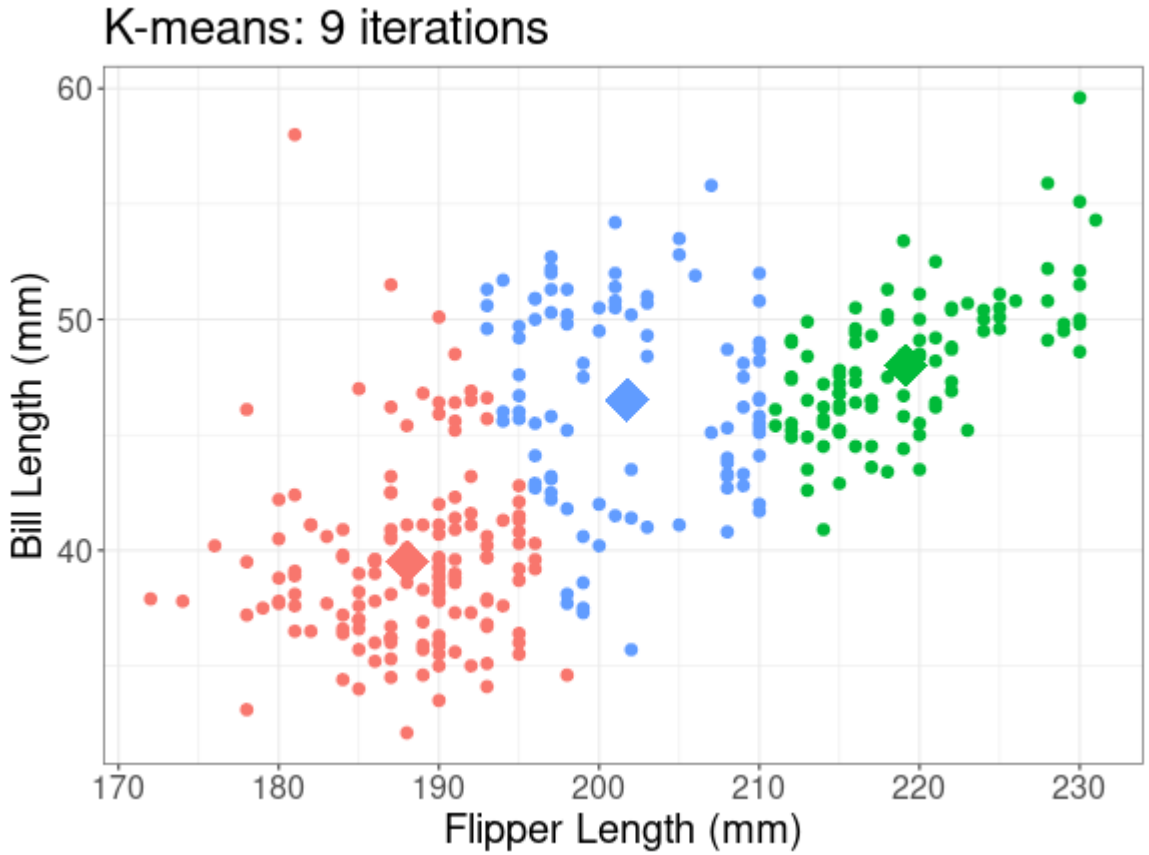
K-means example

- Re-label the points



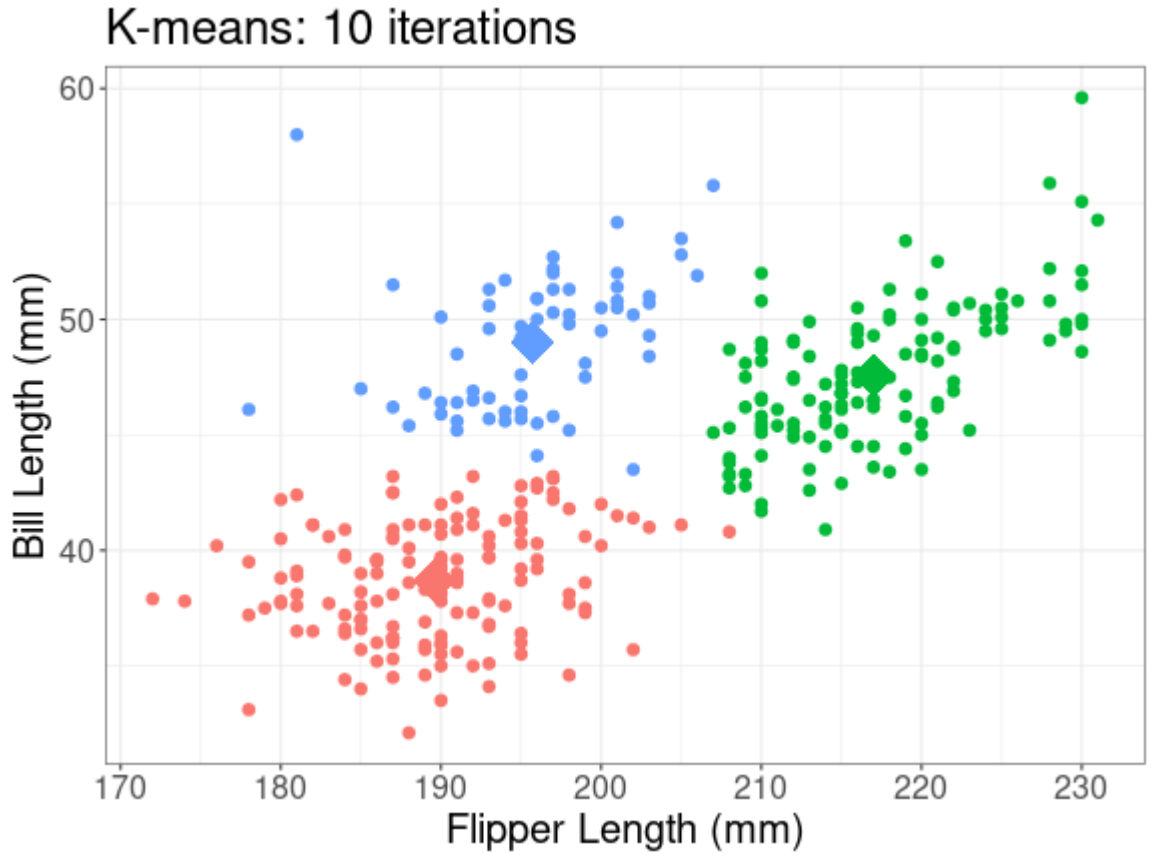
K-means example

- Iterate until convergence



K-means example

- Note some sensitivity to the choice of distance metric. Here's what it looks like if we standardize each Y by it standard deviation.



K-means example

- How do our clusters compare to the **actual** categories? Surprisingly well!

```
table(penguins_complete$cluster2, penguins_complete$species)
```

```
##
##     Adelie Chinstrap Gentoo
##   1    146         5      0
##   2      1         4    122
##   3      4        59      1
```

- But the clusters themselves don't have any underlying interpretation or meaning
  - We **assign** the interpretation to them through inspection!

K-means

- Advantages
  - Easy to implement, fast updating steps and quick convergence.
- Disadvantages
  - Multimodality (we can't really solve this one!)
  - Choice of distance metric matters.
  - Hard to make sense of outcomes that aren't in $\mathbb{R}^d$ (e.g. binary/discrete outcomes)
- With conventional K-means, we have no underlying probability model -- each unit is assigned to a single cluster.
  - Sometimes called **"hard"** K-means.
- We can instead implement an approach sometimes referred to as a "soft" K-means algorithm
  - Our target of inference is the **probability** that a unit belongs to each cluster.

Gaussian Mixture Models

- Assume **latent variables** $z = \{z_1, z_2, \ldots, z_N\}$

$$z_i \underset{\text{i.i.d.}}{\sim} \text{Categorical}(\pi)$$

- Then, the outcome vector Y_i has a multivariate gaussian distribution conditional on $Z_i = k$

$$Y_i | z_i = k \sim \text{Normal}(\mu_k, \Sigma_k)$$

- Our goal is to infer μ_k and Σ_k for each cluster
  - And conditional on these estimates, we can obtain the cluster probabilities $Pr(Z_i = k | Y_i)$

Expectation-Maximization

- Let's write down the marginal likelihood, marginalizing out the latent parameters:

$$\mathcal{L}(\mu, \Sigma, \pi; \mathbf{Y}) = \prod_{i=1}^{N} \sum_{k=1}^{K} f(Y_i|\mu_k, \Sigma_k, Z_i = k) \times p(Z_i = k|\pi)$$

- This is a tricky likelihood to maximize - if we take the log, we get a log of a sum (which doesn't simplify as neatly as a log of a product)

$$\ell(\mu, \Sigma, \pi; \mathbf{Y}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \times \mathcal{N}(Y_i|\mu_k, \Sigma_k) \right)$$

- Suppose we knew the Z_i as well (they were like "data"), then the "complete" log-likelihood would be easier to maximize!

$$\ell(\mu, \Sigma, \pi; \mathbf{Y}, \mathbf{Z}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \log \left(\pi_k^{I(Z_i=k)} \times \mathcal{N}(Y_i|\mu_k, \Sigma_k)^{I(Z_i=k)} \right)$$

Expectation-Maximization

- The **Expectation-Maximization** algorithm is another iterative method for obtaining a maximum likelihood estimate (or maximum a posteriori (MAP) in a Bayesian setting) when the likelihood consists of a sum/integral over the distribution of some **latent variables**
  - It uses the idea of iteratively optimizing a **lower bound** on the likelihood until convergence is reached.
  - The key trick is **Jensen's inequality**. For a **concave** function like the logarithm:

$$\log(E[X]) \geq E[\log(X)]$$

- Consider the general setting where we have a parameter θ, data $\mathbf{X}$ and discrete latent variables Z
- We want to optimize the log-likelihood:

$$\ell(\theta; \mathbf{X}) = \log \sum_{\mathbf{Z}} f(\mathbf{X}, \mathbf{Z}|\theta) = \log \sum_{\mathbf{Z}} f(\mathbf{X}|\mathbf{Z}, \theta) p(\mathbf{Z}|\theta)$$

Expectation-Maximization

- Instead of optimizing the log-likelihood, let's try to come up with a lower-bound. Define an arbitrary distribution on the latent variables $q(\mathbf{Z})$.
- Multiply by 1:

$$\ell(\theta; \mathbf{X}) = \log \sum_{\mathbf{Z}} f(\mathbf{X}|\mathbf{Z}, \theta) p(\mathbf{Z}|\theta) \frac{q(\mathbf{Z})}{q(\mathbf{Z})}$$

- Rearranging terms, we can see that this can be written as an expectation over the distribution q

$$\ell(\theta; \mathbf{X}) = \log E_q \left[\frac{f(\mathbf{X}|\mathbf{Z}, \theta) p(\mathbf{Z}|\theta)}{q(\mathbf{Z})} \right]$$

- By **Jensen's inequality**, we have a lower bound

$$\ell(\theta; \mathbf{X}) \geq E_q \left[\log \left(\frac{f(\mathbf{X}|\mathbf{Z}, \theta) p(\mathbf{Z}|\theta)}{q(\mathbf{Z})} \right) \right]$$

Expectation-Maximization

- And by properties of logs, our lower-bound is:

$$\ell(\theta; \mathbf{X}) \geq E_q[\log(f(\mathbf{X}|\mathbf{Z}, \theta))] + E_q[\log(f\mathbf{Z}|\theta))] - E_q[\log(q(\mathbf{Z})]$$

- We can optimize this **iteratively** by switching between finding an optimal distribution $q^{(t+1)}$ given parameter values $\theta^{(t)}$ and finding parameter values $\theta^{(t+1)}$ given an existing choice of $q^{(t)}$.

1. **Expectation** step: Find $q^{(t+1)}$ (what distribution are we taking the expectation over)
2. **Maximization** step: Find $\theta^{(t+1)}$ (given our q distribution, what is the value of the parameter values that maximizes the lower bound).

E-step

- Can we find a closed-form "optimal" solution for our update $q^{(t+1)}$ given $\theta^{(t)}$?
  - Yes, find the q that makes the inequality an equality!

$$\ell(\theta^{(t)}; \mathbf{X}) \geq E_q\left[\log\left(\frac{f(\mathbf{X}|\mathbf{Z}, \theta^{(t)})p(\mathbf{Z}|\theta^{(t)})}{q(\mathbf{Z})}\right)\right]$$

$$\ell(\theta^{(t)}; \mathbf{X}) \geq \sum_{\mathbf{Z}} \log\left(\frac{f(\mathbf{X}|\mathbf{Z}, \theta^{(t)})p(\mathbf{Z}|\theta^{(t)})}{q(\mathbf{Z})}\right) q(\mathbf{Z})$$

E-step

- Let's try to get the $\mathbf{Z}$ out of the log. Start by factoring the joint distribution in the numerator conditioning on $\mathbf{X}$ instead of $\mathbf{Z}$

$$\ell(\theta^{(t)}; \mathbf{X}) \geq \sum_{\mathbf{Z}} \log \left(\frac{f(\mathbf{Z}|\mathbf{X}, \theta^{(t)}) p(\mathbf{X}|\theta^{(t)})}{q(\mathbf{Z})} \right) q(\mathbf{Z})$$

- Now we can see the optimal choice for q^{t+1} revealed to us. Suppose we plug in $q = f(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$, the conditional distribution of Z given X and $\theta^{(t)}$:

$$\ell(\theta^{(t)}; \mathbf{X}) \geq \sum_{\mathbf{Z}} \log(p(\mathbf{X}|\theta^{(t)}) \times f(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$$

- The logged term no longer depends on $\mathbf{Z}$, so pull it out of the sum

$$\ell(\theta^{(t)}; \mathbf{X}) \geq \log(p(\mathbf{X}|\theta^{(t)}) \times \sum_{\mathbf{Z}} f(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$$

E-step

- The sum is equal to 1 (sum over a PMF/PDF) and we're left with the definition of the marginal log-likelihood, so this expression holds with *equality*

$$\ell(\theta^{(t)}; \mathbf{X}) = \log(p(\mathbf{X}|\theta^{(t)})$$

- So our choice of distribution over which to take the expectation of the joint likelihood is $q^{(t+1)} = f(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$

M-step

- Let's go back to our lower bound - for a given value of $q^{(t)}$, we want to find the $\theta^{(t+1)}$ that maximize the "complete data" log-likelihood

$$\ell(\theta; \mathbf{X}) \geq E_{q^{(t)}}[\log(f(\mathbf{X}|\mathbf{Z}, \theta))] + E_{q^{(t)}}[\log(f(\mathbf{Z}|\theta))] - E_{q^{(t)}}[\log(q^{(t)}(\mathbf{Z})]$$

- Since the third term doesn't depend on θ, we only need to worry about the first two.
  - This is sometimes called the "Q-Function" $Q(\theta|\theta^{(t)})$ or the expected "complete data" log-likelihood

$$Q(\theta|\theta^{(t)}) = E_{\mathbf{Z}|\mathbf{X},\theta^{(t)}}[\log(f(\mathbf{X}, \mathbf{Z}|\theta))]$$

- The M-step sets $\theta^{(t+1)}$ to the value of θ that maximizes this Q-function

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$

Deriving EM for the GMM

- For current values of $\mu^{(t)}$, $\Sigma^{(t)}$, $\pi^{(t)}$, let's derive the Q function.
- We'll start by deriving the conditional distribution $z_i|Y_i, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)}$ using Bayes' rule

$$p(z_i = k|Y_i, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)}) = \frac{p(z_i = k|\pi_k^{(t)}) f(Y_i|\mu^{(t)}, \Sigma^{(t)}, z_i = k)}{f(Y_i|\mu^{(t)}, \Sigma^{(t)})}$$

- Given $z_i = k$, we know the distribution is normal at mean $\mu_k^{(t)}$ and variance $\Sigma_k^{(t)}$. And the denominator is just the marginal

$$\gamma_i^k = p(z_i = k|Y_i, \mu^{(t)}, \Sigma^{(t)}, \pi^{(t)}) = \frac{\pi_k^{(t)} \times \mathcal{N}(Y_i|\mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{j=1}^K \pi_j^{(t)} \times \mathcal{N}(Y_i|\mu_j^{(t)}, \Sigma_j^{(t)})}$$

- These weights, γ_i^k are sometimes called the "responsibility" parameters as they denote the extent to which each cluster is "responsible" for an observation.

Deriving EM for the GMM

- Recall the "complete likelihood"

$$\ell(\mu, \Sigma, \pi; \mathbf{Y}, \mathbf{Z}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \log \left(\pi_k^{I(Z_i=k)} \times \mathcal{N}(Y_i|\mu_k, \Sigma_k)^{I(Z_i=k)} \right)$$

- Simplify it a bit

$$\ell(\mu, \Sigma, \pi; \mathbf{Y}, \mathbf{Z}) = \sum_{i=1}^{N} \sum_{k=1}^{K} I(Z_i = k) \log(\pi_k) + \sum_{i=1}^{N} \sum_{k=1}^{K} I(Z_i = k) \log \mathcal{N}(Y_i|\mu_k, \Sigma_k)$$

- Now, taking the expectation over Z_i, the only component that is not a constant is $I(Z_i = k)$ and $E[I(Z_i = k)] = p(Z_i = k)$ (fundamental bridge).
  - And we got that (conditional) expectation in the previous section: γ_i^k
- So our Q function is

$$Q(\theta|\theta^{(t)}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_i^k \log(\pi_k) + \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_i^k \log \mathcal{N}(Y_i|\mu_k, \Sigma_k)$$

Deriving EM for the GMM

- Closed form solutions are straightforward to obtain for the M-step (and follow from weighted regression)

$$n_k = \sum_{i=1}^{N} \gamma_i^k$$

$$\pi_k = \frac{n_k}{N}$$

$$\mu_k = \frac{1}{n_k} \sum_{i=1}^{N} \gamma_i^k Y_i$$

$$\Sigma_k = \frac{1}{n_k} \sum_{i=1}^{N} \gamma_i^k (Y_i - \mu_k)(Y_i - \mu_k)'$$

Implementing EM

```
gmm_loglik <- function(Y, K, mu, sigma, pi){
  # Log-likelihood of each Y
  lik_normal <- matrix(nrow=nrow(Y), ncol=K)
  for(k in 1:K){
    lik_normal[,k] <- mvtnorm::dmvnorm(Y, mu[k,], sigma[[k]])
  }
  # Log of the sums
  log_likelihood <- sum(apply(lik_normal, 1, function(x) log(sum(x*pi))))
  return(log_likelihood)
}
```

Implementing EM

```
gmm_estep_gamma <- function(Y, K, mu, sigma, pi){
  # Calculate unnormalized gamma_k for each k
  gamma <- matrix(nrow=nrow(Y), ncol=K)
  for (k in 1:K){
    gamma[,k] <- pi[k]*mvtnorm::dmvnorm(Y, mu[k,], sigma[[k]])
  }
  # Normalize the gammas (denominator)
  gamma <- gamma/rowSums(gamma)
  return(gamma)
}
```

Implementing EM

```
sigma_update <- function(Y, K, mu_k, gamma, n_k){ # Super inefficient but works
  # For each k
  sigma <- list()
  for (k in 1:K){
    sigma[[k]] <- matrix(data=0, nrow=ncol(Y), ncol=ncol(Y)) # blank matrix
    for (n in 1:nrow(Y)){
      sigma[[k]] <- sigma[[k]] + gamma[n,k]*outer(Y[n,] - mu_k[k,], Y[n,] - mu_k[k,])
    }
    sigma[[k]] <- sigma[[k]]/n_k[k]
  }
  return(sigma)
}
```

Implementing EM

```
gmm_em <- function(Y, K, maxit=5000, tol=1e-8, verbose=F){
  D <- ncol(Y) # Number of dimensions
  ## Initialize the parameters (pick some reasonable starting points)
  mu_k <- MASS::mvrnorm(K, colMeans(Y), Sigma = var(Y)) # Matrix of means
  sigma_k <- list() # list of covariances
  for (k in 1:K){
    sigma_k[[k]] <- var(Y)
  }
  pi <- rep(1/K, K)
  curr_lik <- gmm_loglik(Y, K, mu_k, sigma_k, pi) #Evaluate current likelihood
  if(verbose) print(str_c("Log-Likelihood: ", curr_lik))
  for(iter in 2:maxit){
    # E-step
    gamma <- gmm_estep_gamma(Y, K, mu_k, sigma_k, pi)
    # M-step
    n_k <- colSums(gamma)
    pi <- n_k/sum(n_k)
    for(k in 1:K){
        mu_k[k,] <- colSums(gamma[,k]*Y)/n_k[k]
    }
    sigma_k <- sigma_update(Y, K, mu_k, gamma, n_k)
    # Check convergence
    new_lik <- gmm_loglik(Y, K, mu_k, sigma_k, pi)
    if (abs(new_lik - curr_lik) < tol){
      gamma <- gmm_estep_gamma(Y, K, mu_k, sigma_k, pi)
      if(verbose)print(str_c("Log-Likelihood: ", curr_lik))
      if(verbose) print(str_c("Log-Likelihood: ", new_lik))
```

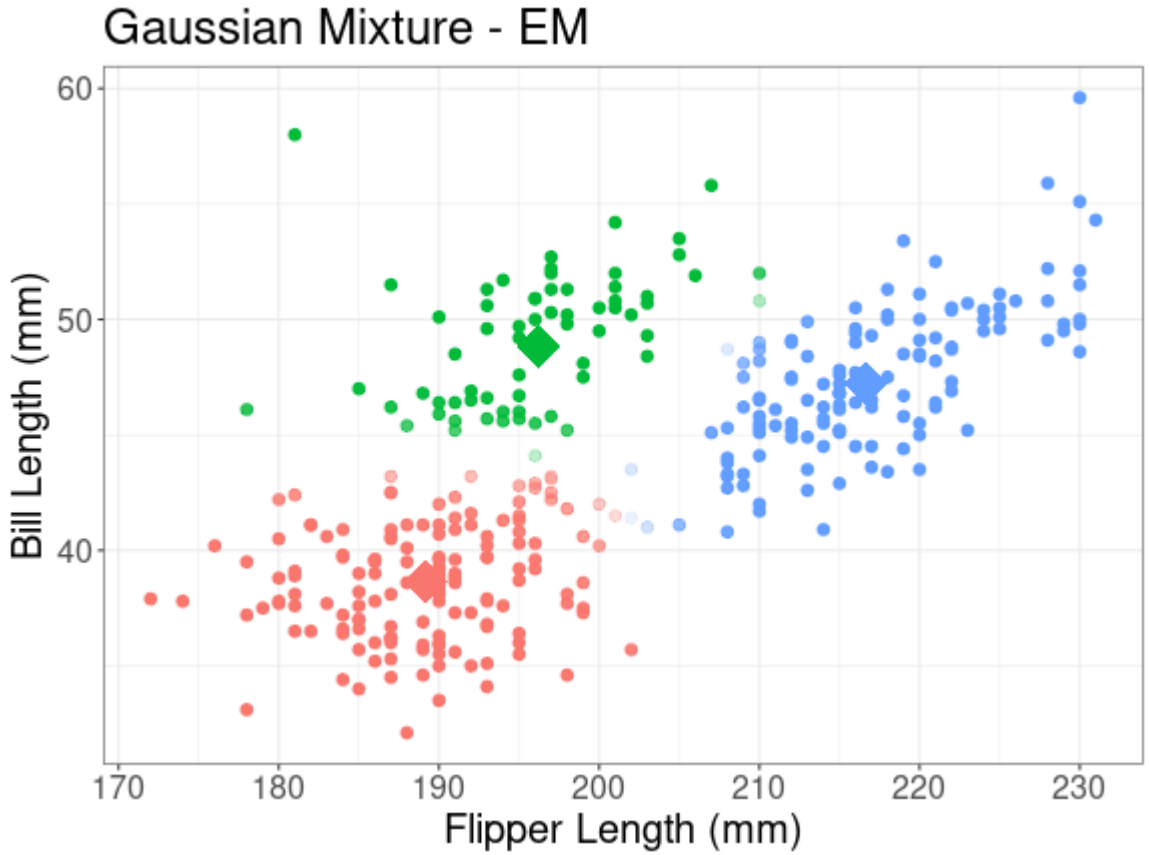
Implementing EM

```
set.seed(60639)
penguins_em_3 <- gmm_em(Y=as.matrix(penguins_complete %>% select(flipper_length_mm, bill_length
```

```
## [1] "Log-Likelihood: -2464.23892416674"
## [1] "Log-Likelihood: -2286.49407632345"
## [1] "Log-Likelihood: -2265.16441563185"
## [1] "Log-Likelihood: -2244.45549391833"
## [1] "Log-Likelihood: -2244.22093530689"
## [1] "Log-Likelihood: -2244.21933979789"
## [1] "Log-Likelihood: -2244.21927829982"
## [1] "Log-Likelihood: -2244.21927596998"
## [1] "Log-Likelihood: -2244.21927591305"
## [1] "Log-Likelihood: -2244.21927590356"
## [1] "Complete!"
```

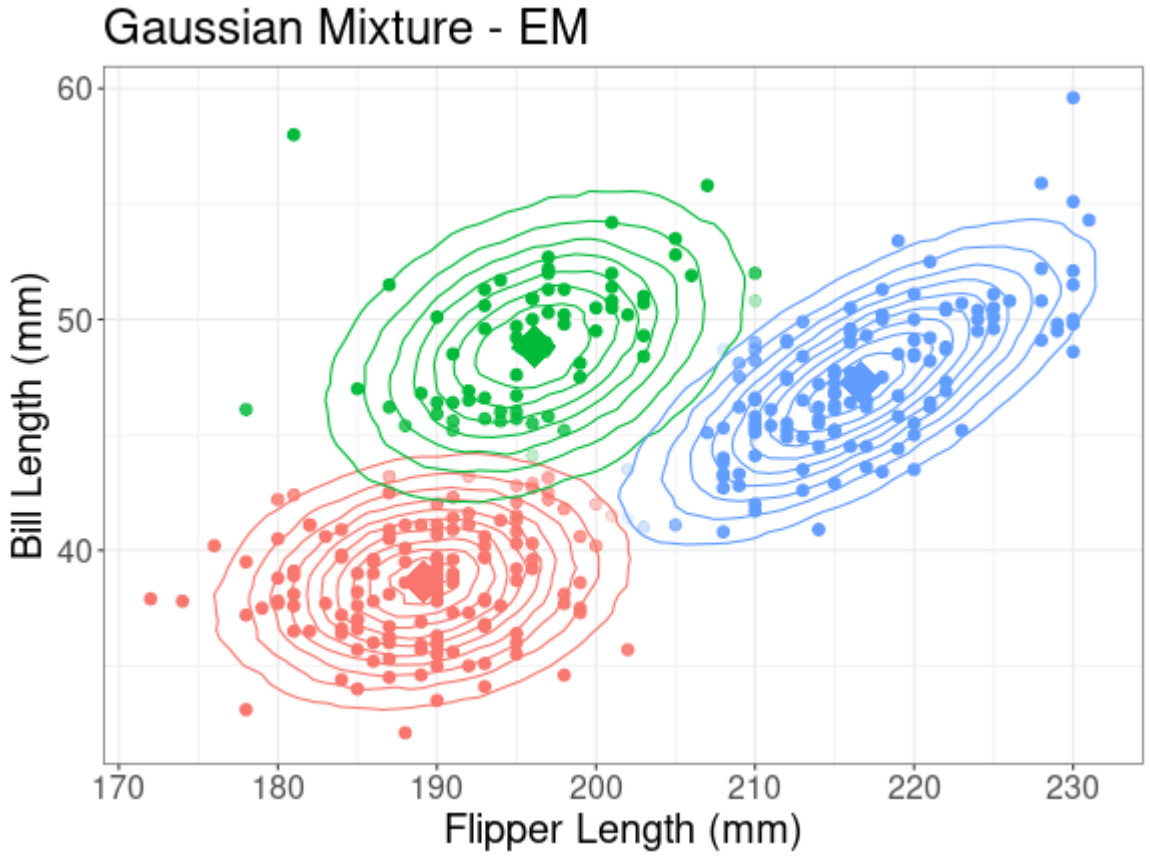
Visualizing EM

- Now our point labels have **probabilities** attached



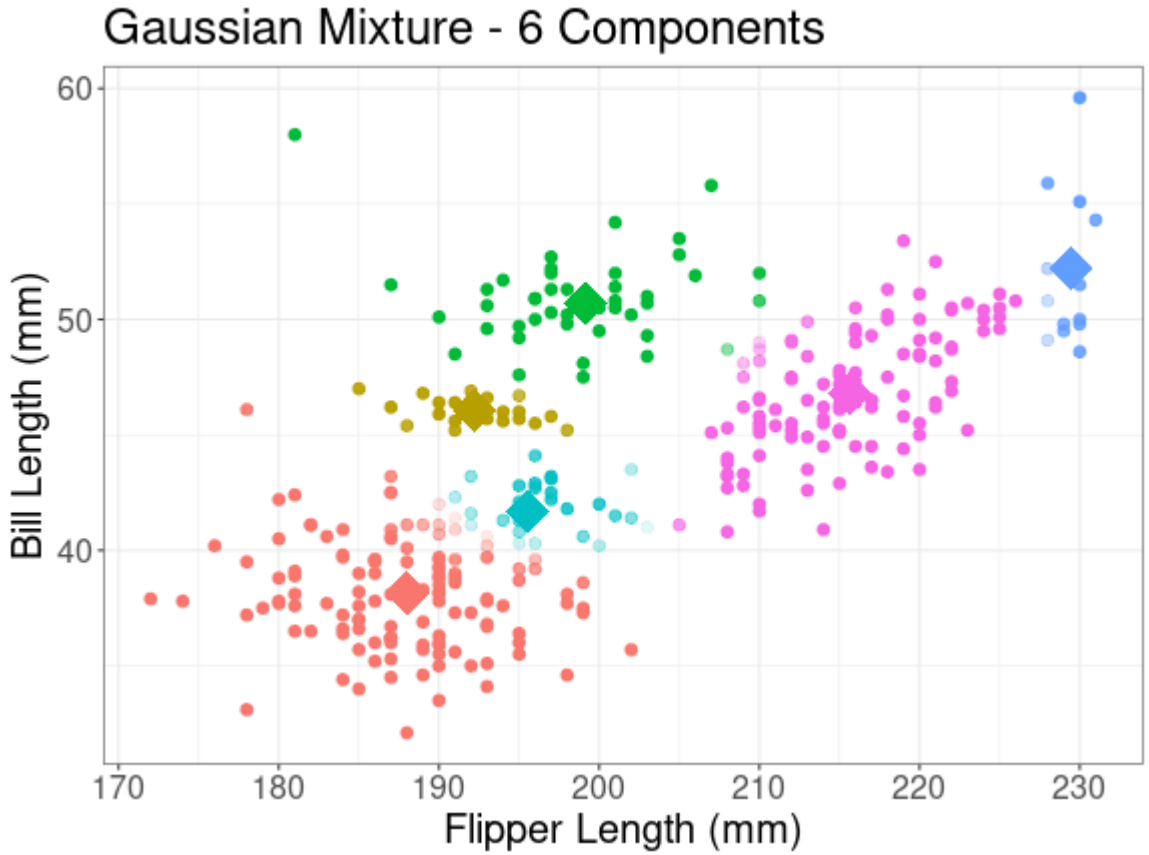
Visualizing EM

- We can also visualize the component distributions



Alternate K

- What does the 6-component mixture look like?



Challenges with mixture models

- **Highly model dependent** - our identification of the cluster centroids + variances depends heavily on our selection of the appropriate distribution for Y_i
  - Normal does okay for outcomes where it's plausible that CLT kicks in, but not true of all Y_i
- **Multi-modality** - Mixture model likelihoods often have multiple modes and EM is only guaranteed to converge to a **local** optimum
  - Can get stuck in a "bad" EM run (**Solution**: Run multiple EM chains with different starting values and pick the one with the best log-likelihood)
  - "Label-switching" problem: Permuting the labels doesn't change the likelihood
- **Challenges with Bayes** - Sampling-based inference can be tricky with GMMs due to the label-switching problem.
  - Can implement many models via MCMC/Gibbs but need tricks to avoid having the chain jump between permutations.
  - Common to use an approximation to the likelihood around the posterior mode obtained via EM.
  - Stan doesn't like sampling discrete latent variables.