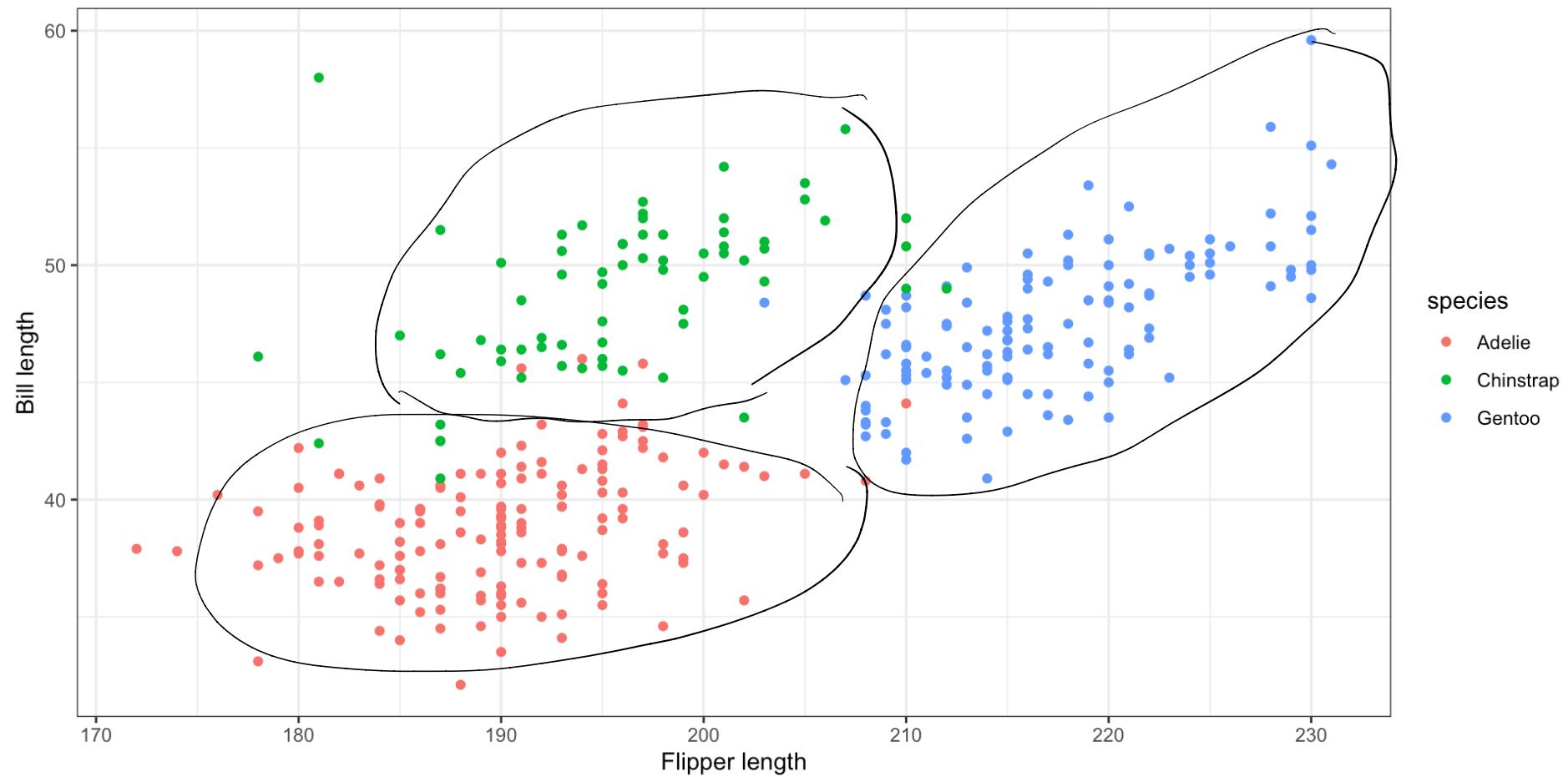


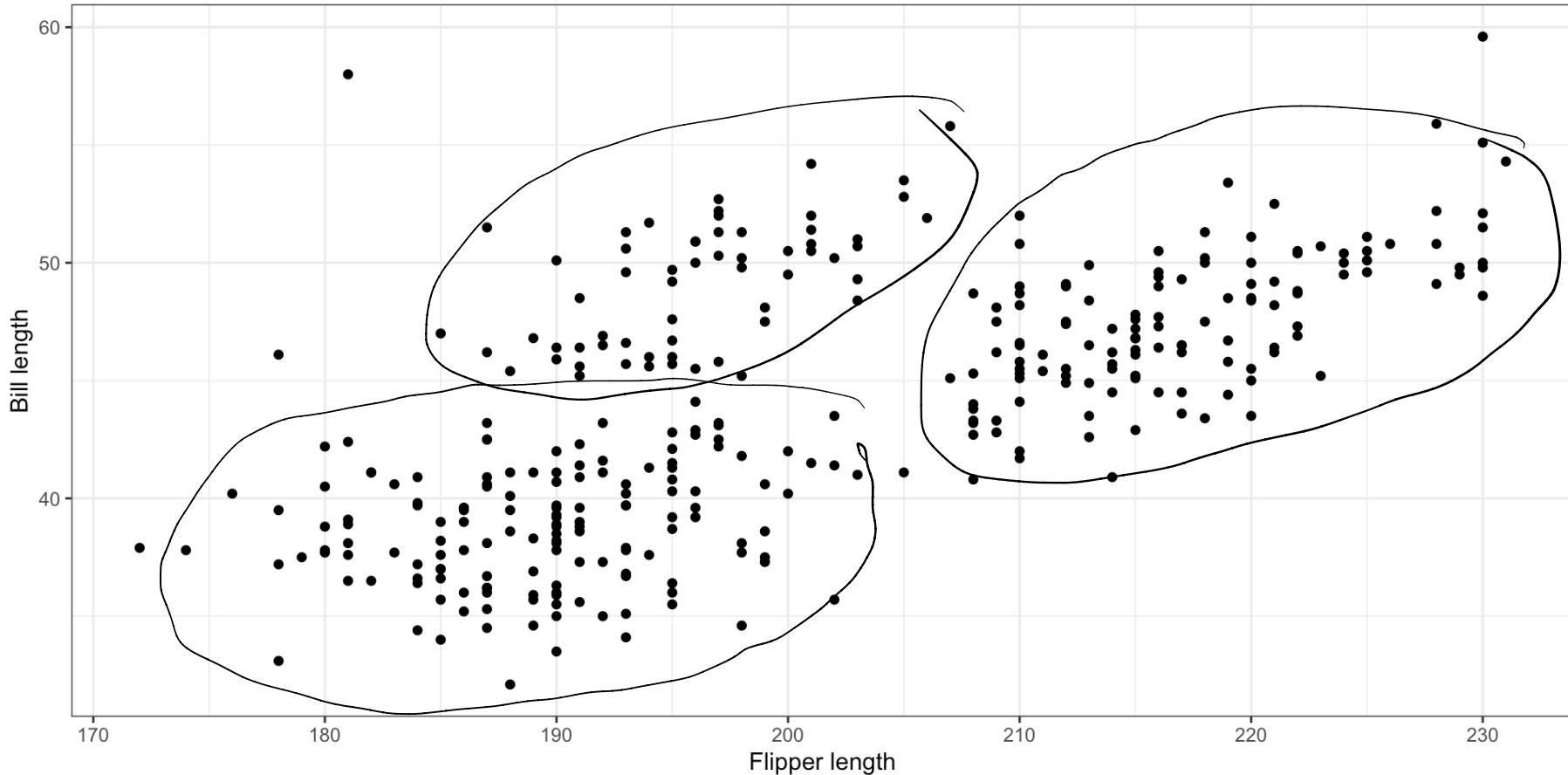
Lecture 27: Implementing statistical methods

Penguins data

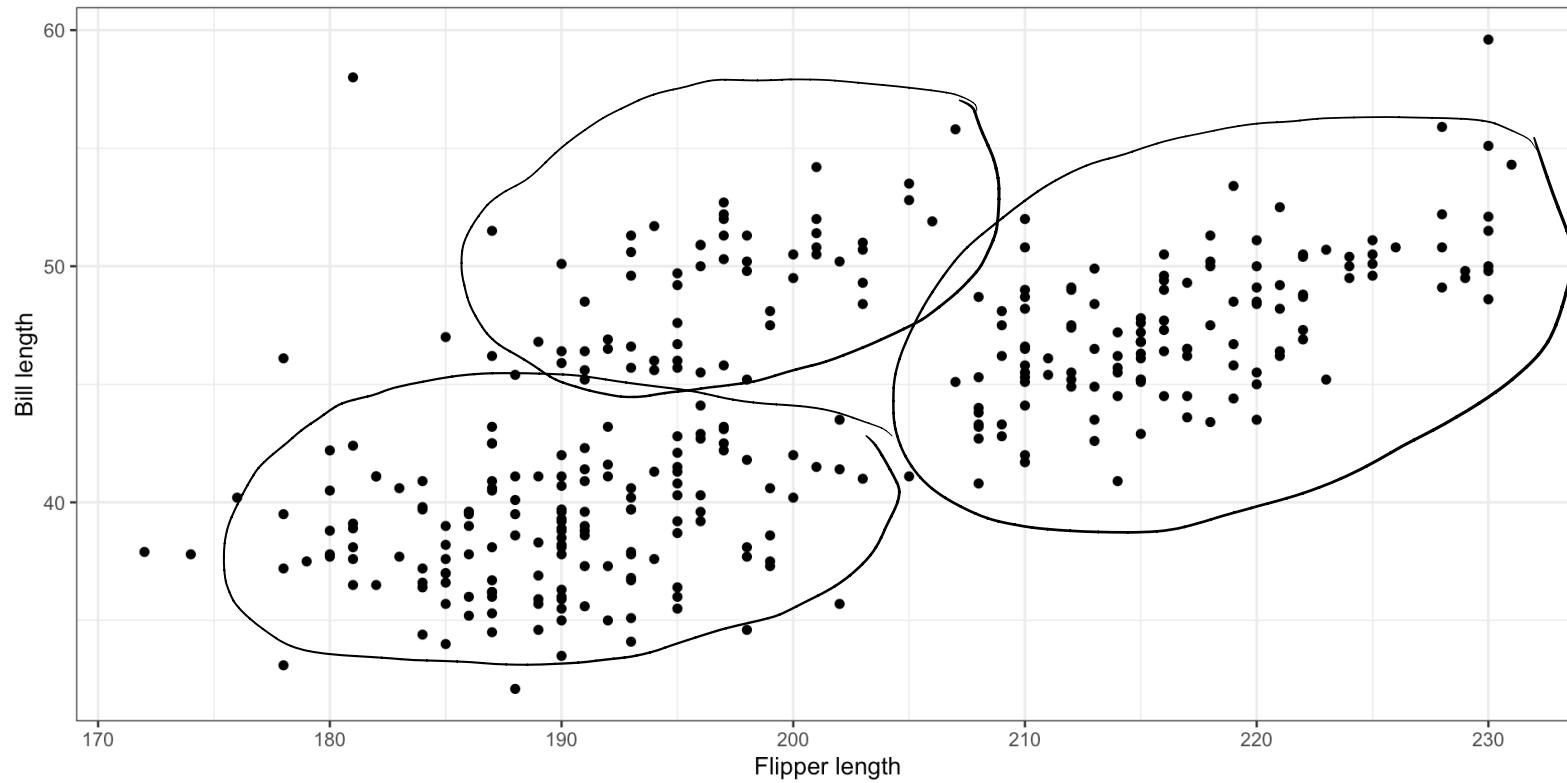


Penguins data

What if we didn't know the species?

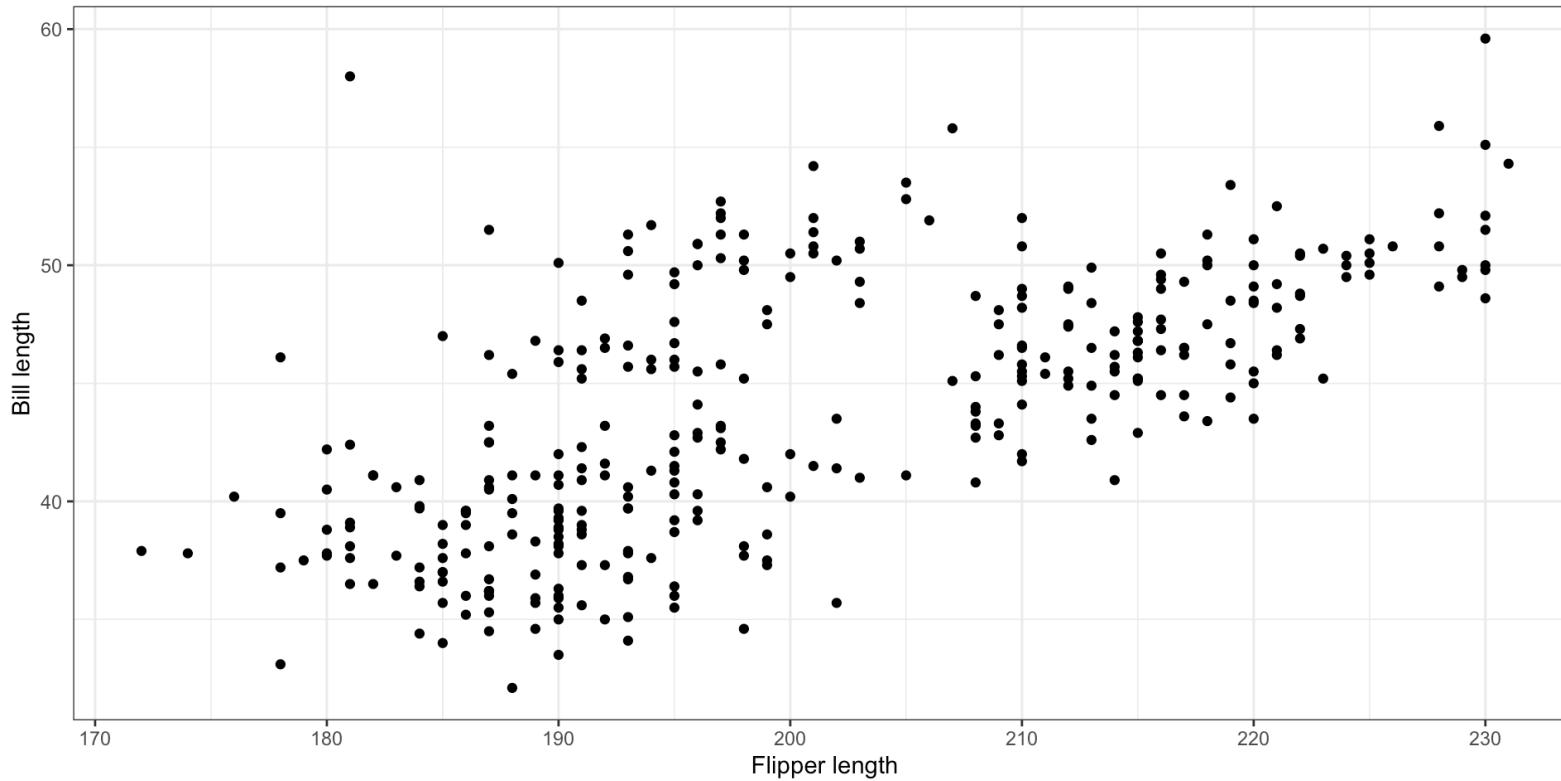


Though excercise



With your neighbor, discuss how you might identify potential groups in this scatterplot, *without* knowing species information.

Clustering

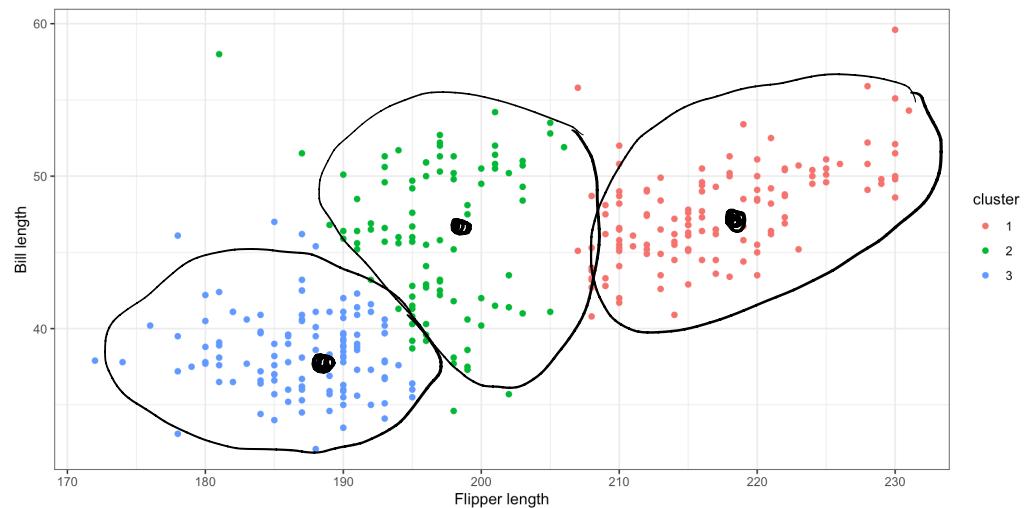


Clustering is the process of dividing data into groups based on the observed variables.

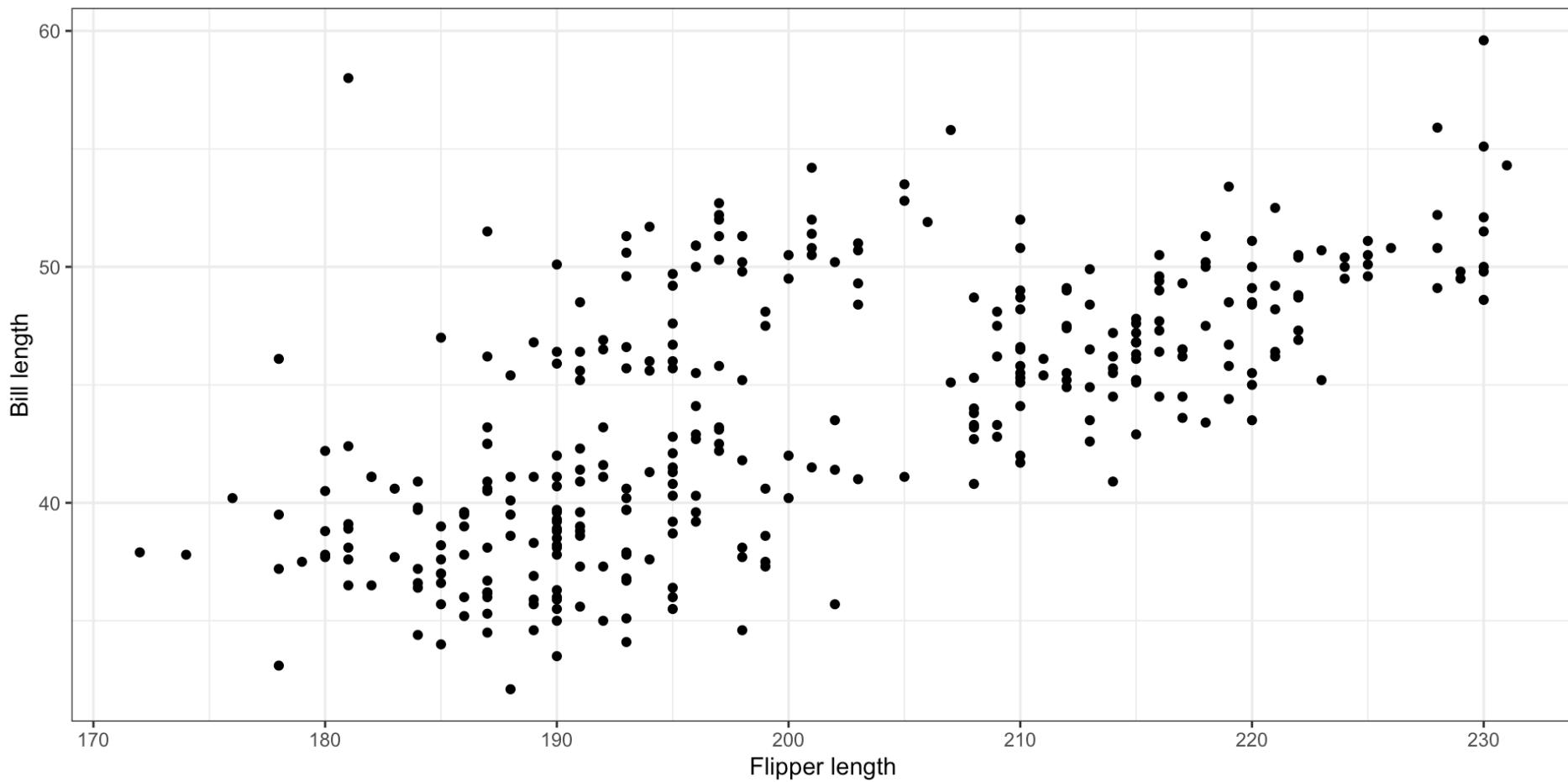
↖ # of clusters (subgraphs) in the data

k-means clustering

```
1 # cluster with k = 3
2 clusters <- kmeans(df, 3, algorithm = "Lloyd", iter.max=20)$cluster
3
4 df |>
5   mutate(cluster = as.factor(clusters)) |>
6   ggplot(aes(x = flipper_length_mm, y = bill_length_mm, color = cluster))
7   geom_point() +
8   labs(x = "Flipper length", y = "Bill length") +
9   theme_bw()
```



k-means clustering



k-means clustering

(Lloyd's algorithm)

Iterative algorithm:

- ① Initialize: given a value of k (# clusters), randomly select k observations to be the initial group means
- ② Calculate distances: For every observation in the data, calculate distance to each of the k group means
- ③ Assign groups: assign each observation to the nearest group mean
- ④ Update means: For each group, re-calculate the group mean as the mean of points assigned to that group in step 3
- ⑤ Iterate: repeat steps 2-4 until the algorithm converges (or a max # of iterations reached)

Recommendations

- Before putting code in a function, write it for an example dataset. *Then* generalize
- Start small (one row, one iteration, etc.), then build up
- Work step by step

Step 1: initialization

Initialization: Randomly select k observations from the dataset to be the *initial* group means

```
1 head(df)  
  
# A tibble: 6 × 2  
  flipper_length_mm bill_length_mm  
        <int>            <dbl>  
1          181            39.1  
2          186            39.5  
3          195            40.3  
4          193            36.7  
5          190            39.3  
6          181            38.9
```

← first 6 rows of df
(full df has a few hundred rows)

Question: How would we randomly choose $k = 3$ rows from our dataset?

```
n <- nrow(df)  
df[sample(1:n, 3), ]  
~~~~~  
random sample of row #s
```

← (replace = F is default)

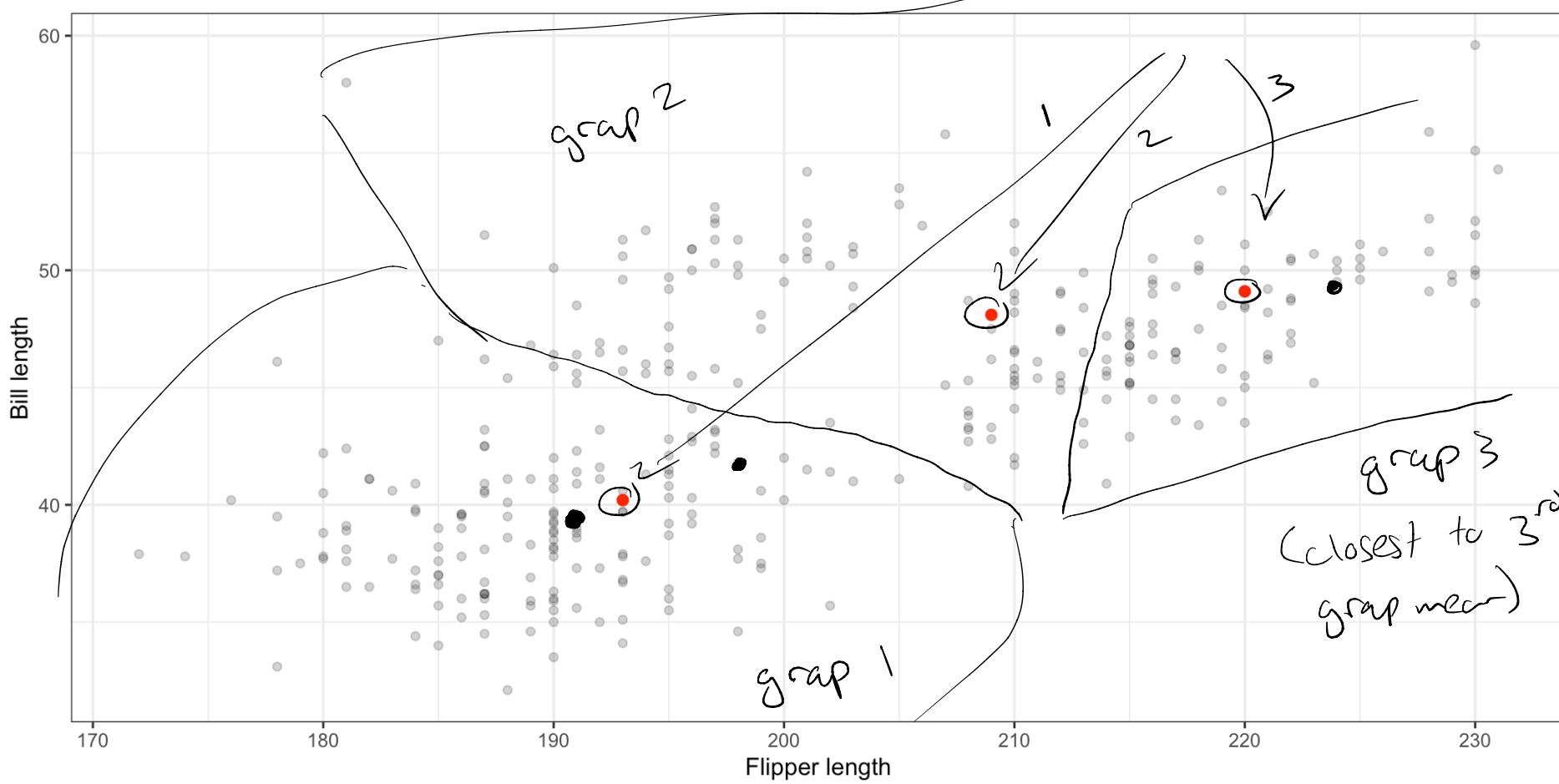
Step 1: initialization

Initialization: Randomly select k observations from the dataset to be the *initial* group means

```
1 k <- 3
2 n <- nrow(df)
3
4 # choose the initial means
5 means <- df[sample(1:n, k), ]
6 means

# A tibble: 3 × 2
  flipper_length_mm bill_length_mm
            <int>          <dbl>
1           209          48.1
2           220          49.1
3           193          40.2
```

Step 1: initialization



$$\text{dist}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Step 2: calculate distances

Calculate distances: For every observation in the dataset, calculate the distance to each of the k group means

```
1 means
```

```
# A tibble: 3 × 2
  flipper_length_mm bill_length_mm
        <int>            <dbl>
1        209             48.1
2        220             49.1
3        193             40.2
```

```
1 head(df)
```

```
# A tibble: 6 × 2
  flipper_length_mm bill_length_mm
        <int>            <dbl>
1        181             39.1
2        186             39.5
3        195             40.3
4        193             36.7
5        190             39.3
6        181             38.9
```

Distance (obs 1, group mean 1):

$$\sqrt{(181 - 209)^2 + (39.1 - 48.1)^2}$$

Distance² (obs 1, grp mean 1):

$$(181 - 209)^2 + (39.1 - 48.1)^2$$

Step 2: calculate distances

Question: How could we calculate these distances in R?

```
1 means
```

```
# A tibble: 3 × 2
  flipper_length_mm bill_length_mm
        <int>          <dbl>
1        209           48.1
2        220           49.1
3        193           40.2
```

```
1 head(df)
```

```
# A tibble: 6 × 2
  flipper_length_mm bill_length_mm
        <int>          <dbl>
1        181           39.1
2        186           39.5
3        195           40.3
4        193           36.7
5        190           39.3
6        181           38.9
```

Step 2: calculate distances

```
1 means[1,]
```

A tibble: 1 × 2

	flipper_length_mm	bill_length_mm
1	<int>	<dbl>

209 48.1

(x_1, y_1)

```
1 head(df)
```

A tibble: 6 × 2

	flipper_length_mm	bill_length_mm
1	<int>	<dbl>

181 39.1

(x_2, y_2)

2 186 39.5

3 195 40.3

4 193 36.7

5 190 39.3

6 181 38.9

$181 - 209 \quad 39.1 - 48.1 \quad e.s$

```
1 sweep(df, 2, means[1,])
```

	flipper_length_mm	bill_length_mm
[1,]	-28	-9.0
[2,]	-23	-8.6
[3,]	-14	-7.8
[4,]	-16	-11.4
[5,]	-19	-8.8
[6,]	-28	-9.2

Subtract first group mean from each row in data

($x_2 - x_1, y_2 - y_1$)

[7 ,]	-14	-8 .9
[8 ,]	-16	-14 .0
[9 ,]	-19	-6 .1
[10 ,]	-23	-10 .3
[11 ,]	-29	-10 .3
[12 ,]	-27	-7 .0
[13 ,]	-18	-9 .5

Step 2: calculate distances

```
1 sweep(df, 2, means[1,])
```

	flipper_length_mm	bill_length_mm	
[1,]	-28	-9.0	$(x_2 - x_1, y_2 - y_1)$
[2,]	-23	-8.6	
[3,]	-14	-7.8	

Question: Now how do I get the distances?

$\text{sweep}(\text{df}, 2, \text{means}[1,])^2$

$$(-28)^2 \quad (-9.0)^2$$

$$(-23)^2 \quad (-8.6)^2$$

:

:

etc

etc.

rowSums(sweep(df, 2, means[1,])²)

$$(-28)^2 + (-9.0)^2$$

$$(-23)^2 + (-8.6)^2$$

... etc.

Step 2: calculate distances

```
1 sweep(df, 2, means[1,])  
  
  flipper_length_mm bill_length_mm  
[1,]           -28            -9.0  
[2,]           -23            -8.6  
[3,]           -14            -7.8  
  
1 rowSums(sweep(df, 2, means[1,])^2)  
[1] 865.00 602.96 256.84
```

Squared distances to the
first group mean
for each row in the data

So far

```
1 k <- 3
2 n <- nrow(df)
3
4 # choose the initial means
5 means <- df[sample(1:n, k), ]
6
7 # calculate distances for the first group mean
8 rowSums(sweep(df, 2, means[1,])^2)
```

Question: How would we modify this code to get distances for *each* group mean?

Step 2: calculate distances

```
1 dists <- matrix(nrow = n, ncol = k)
2 for(i in 1:k){
3   dists[,i] <- rowSums((sweep(df, 2, means[i,]))^2)
4 }
5
6 head(dists)
```

	[,1]	[,2]	[,3]
[1,]	865.00	1621.00	145.21
[2,]	602.96	1248.16	49.49
[3,]	256.84	702.44	4.01
[4,]	385.96	882.76	12.25
[5,]	438.44	996.04	9.81
[6,]	868.64	1625.04	145.69

matrix to hold (squared) distances
to each group mean

distance matrix

n rows (one row per observation)

k columns (one column per group (cluster))

dists [i,j] : (squared)
distance between observation i and
group mean j

Step 3: Assign points to the nearest group mean

```
1 head(dists)
```

	[,1]	[,2]	[,3]	group
[1,]	865.00	1621.00	145.21	3
[2,]	602.96	1248.16	49.49	3
[3,]	256.84	702.44	4.01	3
[4,]	385.96	882.76	12.25	
[5,]	438.44	996.04	9.81	
[6,]	868.64	1625.04	145.69	

Question: Which group mean does each row get assigned to?

Step 3: Assign points to the nearest group mean

```
1 head(dists)
```

```
 [,1]    [,2]    [,3]  
[1,] 865.00 1621.00 145.21  
[2,] 602.96 1248.16 49.49  
[3,] 256.84 702.44  4.01  
[4,] 385.96 882.76 12.25  
[5,] 438.44 996.04  9.81  
[6,] 868.64 1625.04 145.69
```

Question: How could I write code to assign each row?

`which.min(...)` ← function to find the index of the minimum

example: `which.min(dists[1,])` => 3

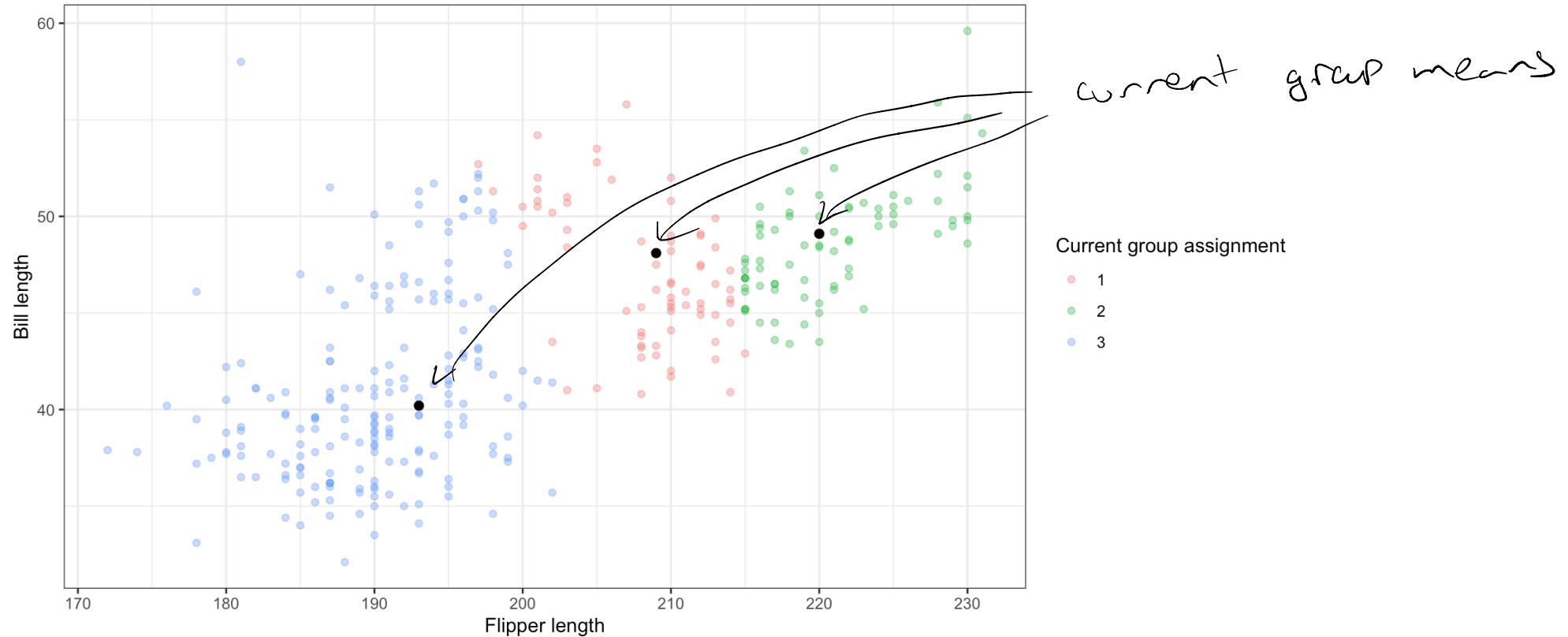
iterate: for loop?

```
for (i in 1:n){  
  groups[i] <- which.min(dists[i,])  
}
```

alternative:
`apply(dists, 1, which.min)`

Step 3: Assign points to the nearest group mean

```
1 groups <- apply(dists, 1, which.min)
```



Next step:

use current group assignments to
update our group means

Step 4: Re-calculate group means

Update means: For each group, re-calculate the new group mean as the mean of the points assigned to that group

```
1 head(df)  
2  
3 head(groups)
```

	flipper_length_mm	bill_length_mm
[1,]	190	38.8
[2,]	220	49.1
[3,]	196	50.0
[4,]	197	52.2
[5,]	209	42.8
[6,]	210	42.0

[1] 3 2 3 3 1 1

= group 3

want: average flipper length } average bill length for
each group

option 1:
df %>%
 mutate(groups = groups) %>%
 filter(groups == 3) %>%

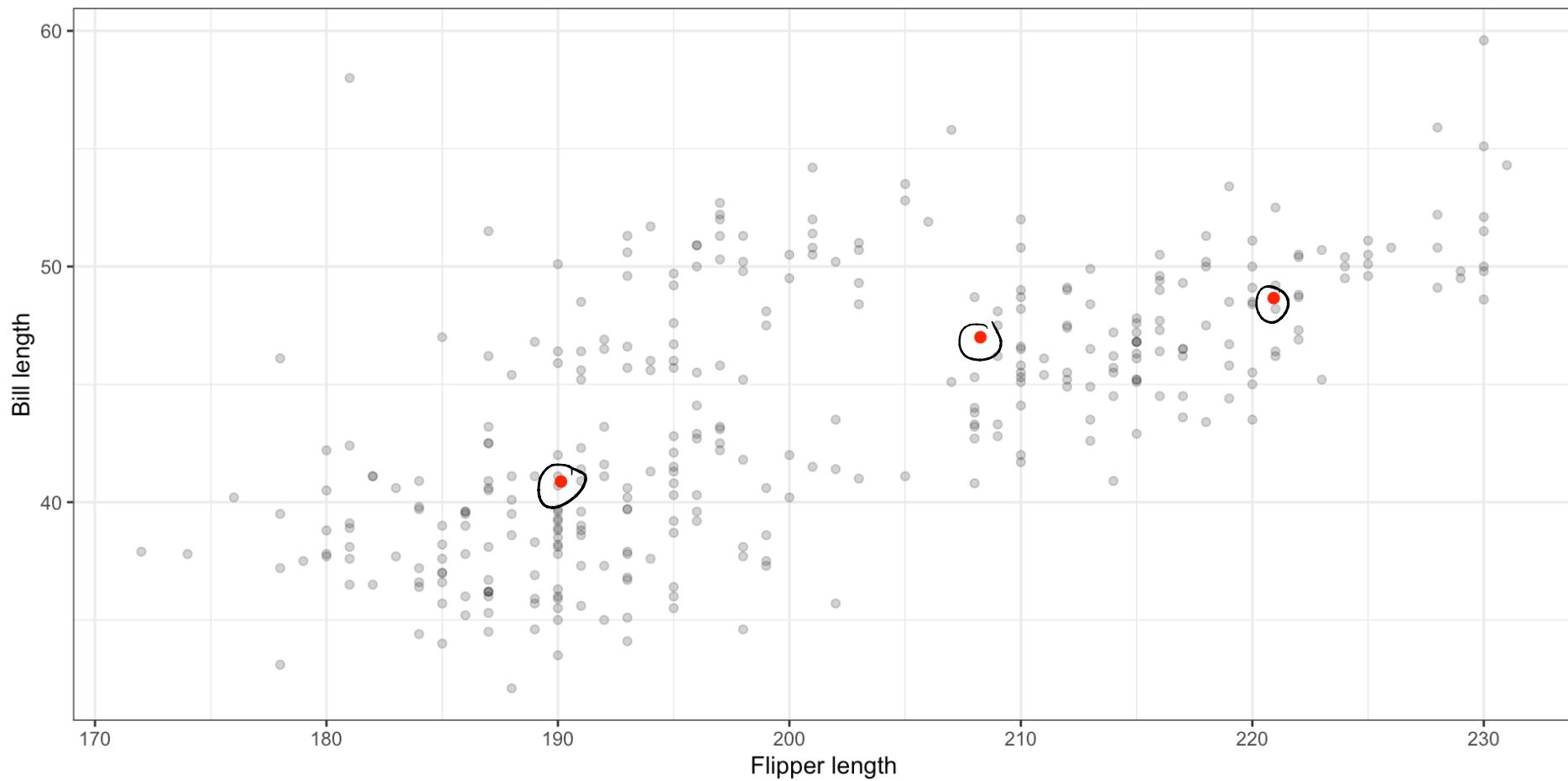
option 2:
colMeans(df[groups == 3,])

Step 4: Re-calculate the group means

```
1 for(i in 1:k){  
2   means[i,] <- colMeans(df[groups == i,])  
3 }  
4  
5 means
```

	flipper_length_mm	bill_length_mm
[1,]	208.2535	47.00000
[2,]	220.9231	48.66026
[3,]	190.1295	40.87461

Step 4: Re-calculate the group means



So far:

means [i,] \leftarrow ith row of matrix

```
1 # choose the initial means  
2 means <- df[sample(1:n, k), ]  
3  
4 # calculate distances  
5 dists <- matrix(nrow = n, ncol = k)  
6 for(i in 1:k){  
7   dists[,i] <- rowSums((sweep(df, 2, means[i,]))^2)  
8 }  
9  
10 # assign groups  
11 groups <- apply(dists, 1, which.min)  
12  
13 # update means  
14 for(i in 1:k){  
15   means[i,] <- colMeans(df[groups == i,])  
16 }
```

↑ fill in all the means for group ; (mean flipper length)

↑ mean bill length

Question: What should we do next?

go back : \rightarrow re-compute distances
 \rightarrow re-compute group assignments ↑
 \rightarrow re-compute group means
(while loop)

