# 432 Class 25 Slides

thomaselove.github.io/432

2022-04-14

# Today's Topics

- K-Means Cluster Analysis
- . . . and a little bit of Principal Components Analysis

# Today's R Packages

```
library(janitor); library(here)
library(knitr); library(magrittr)
library(naniar)
library(palmerpenguins)
library(mdsr) # for the world_cities data set
library(tidymodels)
library(tidyverse)

theme_set(theme_bw())
```

# Clustering the 4,000 Biggest Cities in the World

- We'll start with an example from section 9.1.2 of Baumer, Kaplan and Horton's *Modern Data Science with R*.

```
BigCities <- world_cities %>%
    arrange(desc(population)) %>%
    head(4000) %>%
    select(longitude, latitude)

glimpse(BigCities)
```

```
Rows: 4,000
Columns: 2
$ longitude <dbl> 121.45806, 28.94966, -58.37723, 72.88261~
$ latitude  <dbl> 31.22222, 41.01384, -34.61315, 19.07283,~
```

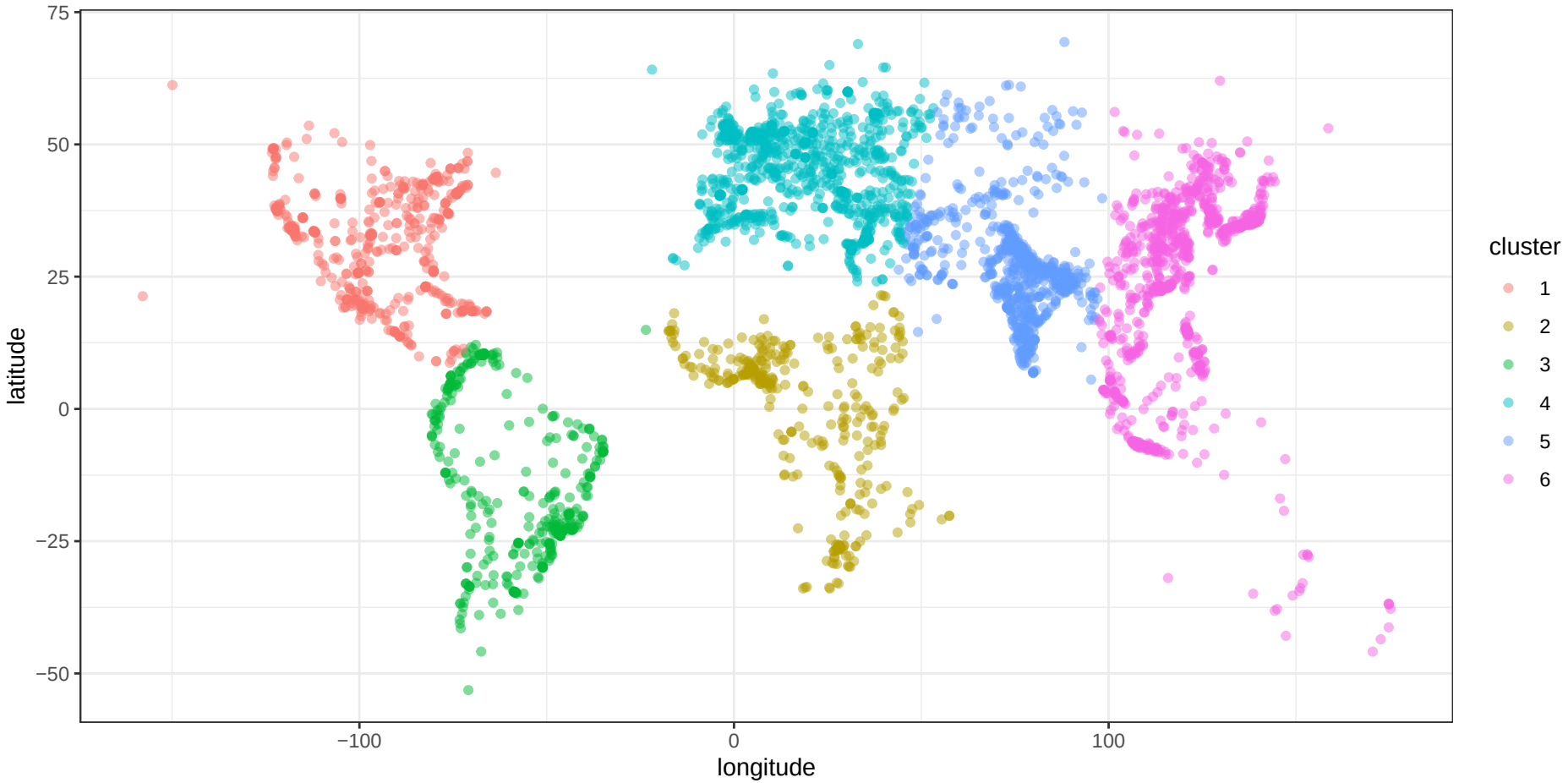
# Cluster by the 6-means algorithm

```
set.seed(432)
city_clusts <- BigCities %>%
    kmeans(centers = 6) %>%
    fitted("classes") %>%
    as.character()

BigCities <- BigCities %>%
    mutate(cluster = city_clusts)

head(BigCities, 2)
```

```
# A tibble: 2 x 3
  longitude latitude cluster
      <dbl>    <dbl> <chr>
1     121.      31.2 6
2      28.9     41.0 4
```

# Map the cities and color by clusters

```
ggplot(BigCities, aes( x = longitude, y = latitude)) +
    geom_point(aes(color = cluster), alpha = 0.5)
```




- The clustering algorithm has (essentially) identified the continents.

# How does K-Mean Clustering Work?

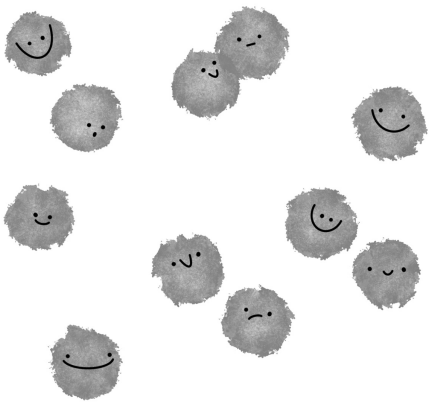
You can visit https://www.tidymodels.org/learn/statistics/k-means/ for a brief explanation of the clustering process using an animation from Allison Horst. Here, I'll look at the pieces one slide at a time.

Allison's materials are available at https://github.com/allisonhorst/stats-illustrations/tree/master/other-stats-artwork

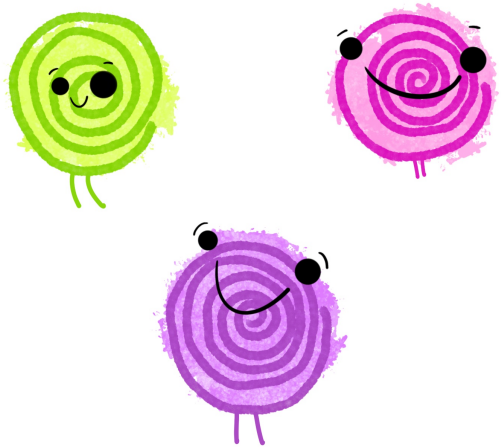
# From Allison Horst (1/11)

**k-means clustering**: assign each observation to one of k clusters based on the nearest cluster centroid.

OBSERVATIONS



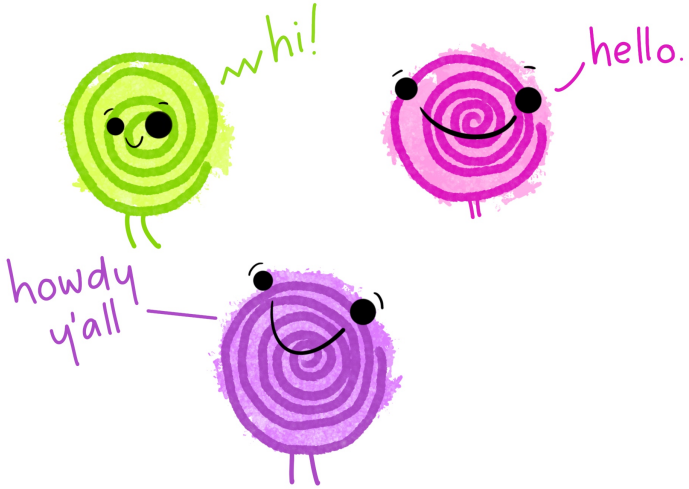
cluster CENTROIDS



# From Allison Horst (2/11)

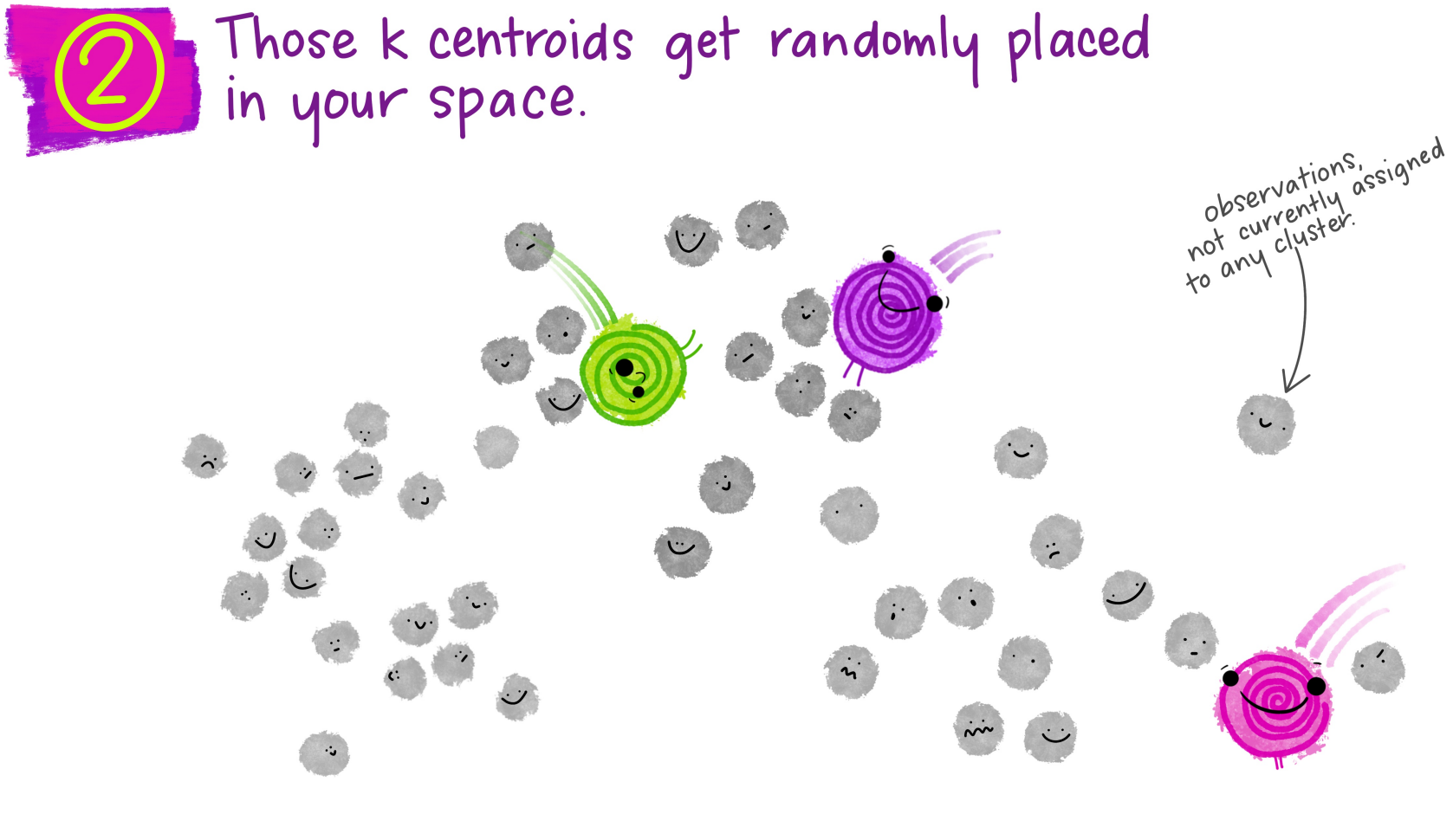

Specify the number of clusters (in this example, k = 3).

Then imagine k cluster centroids are created.

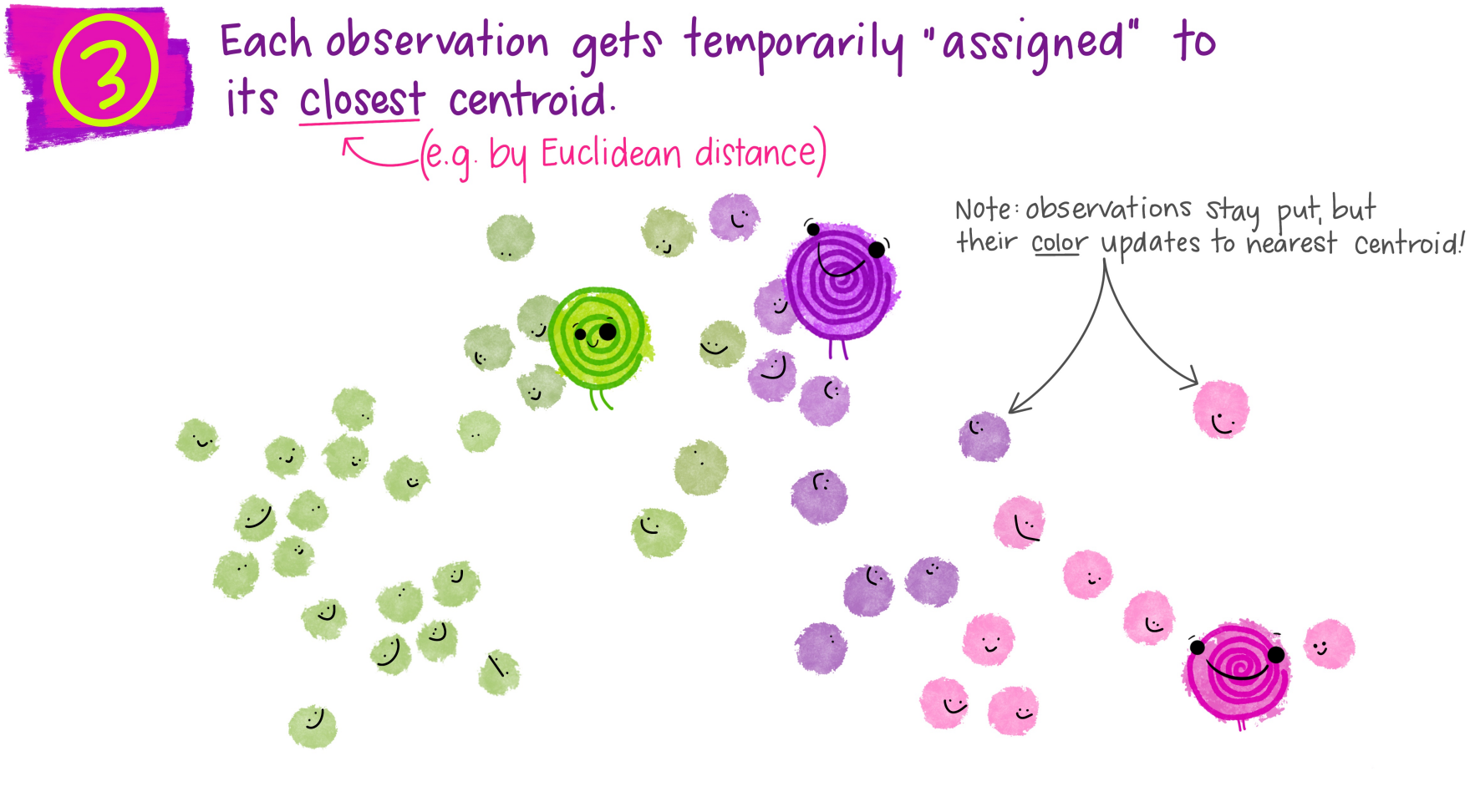



@allison_horst

# From Allison Horst (3/11)

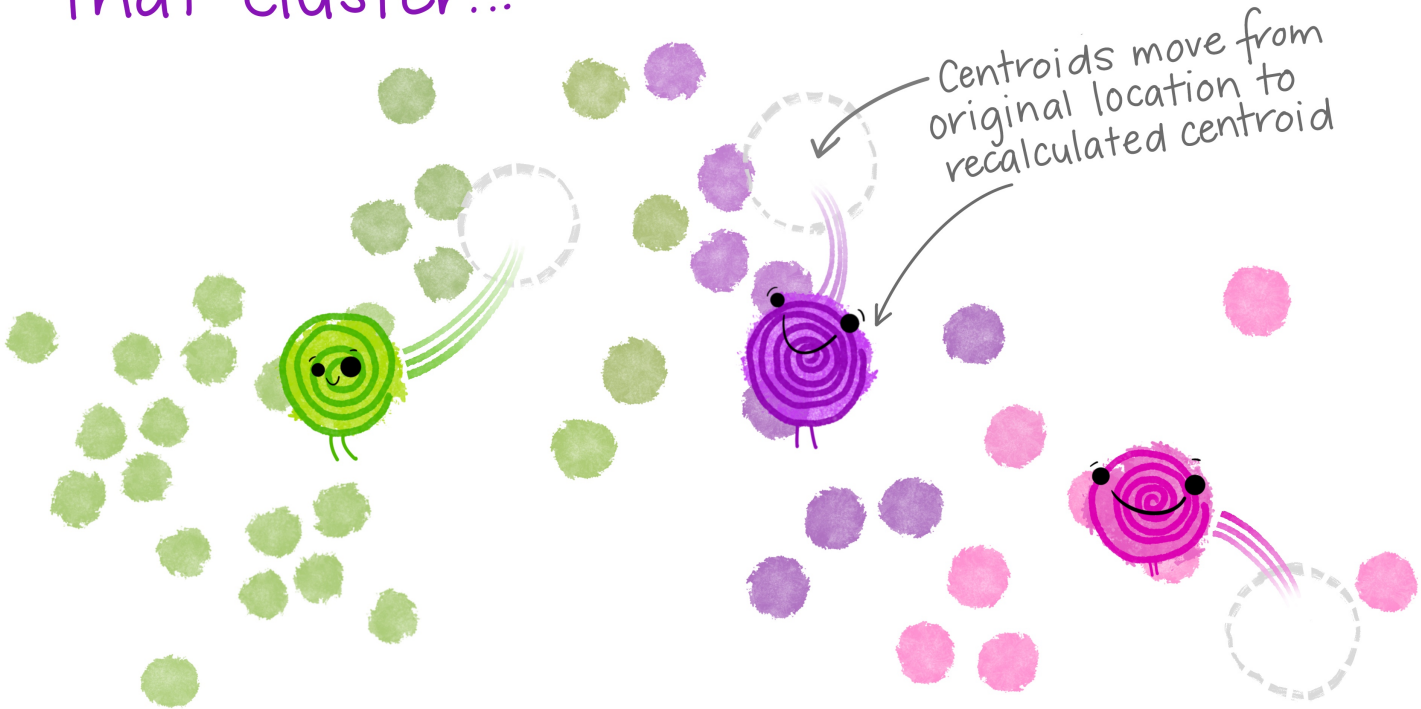


# From Allison Horst (4/11)



# From Allison Horst (5/11)

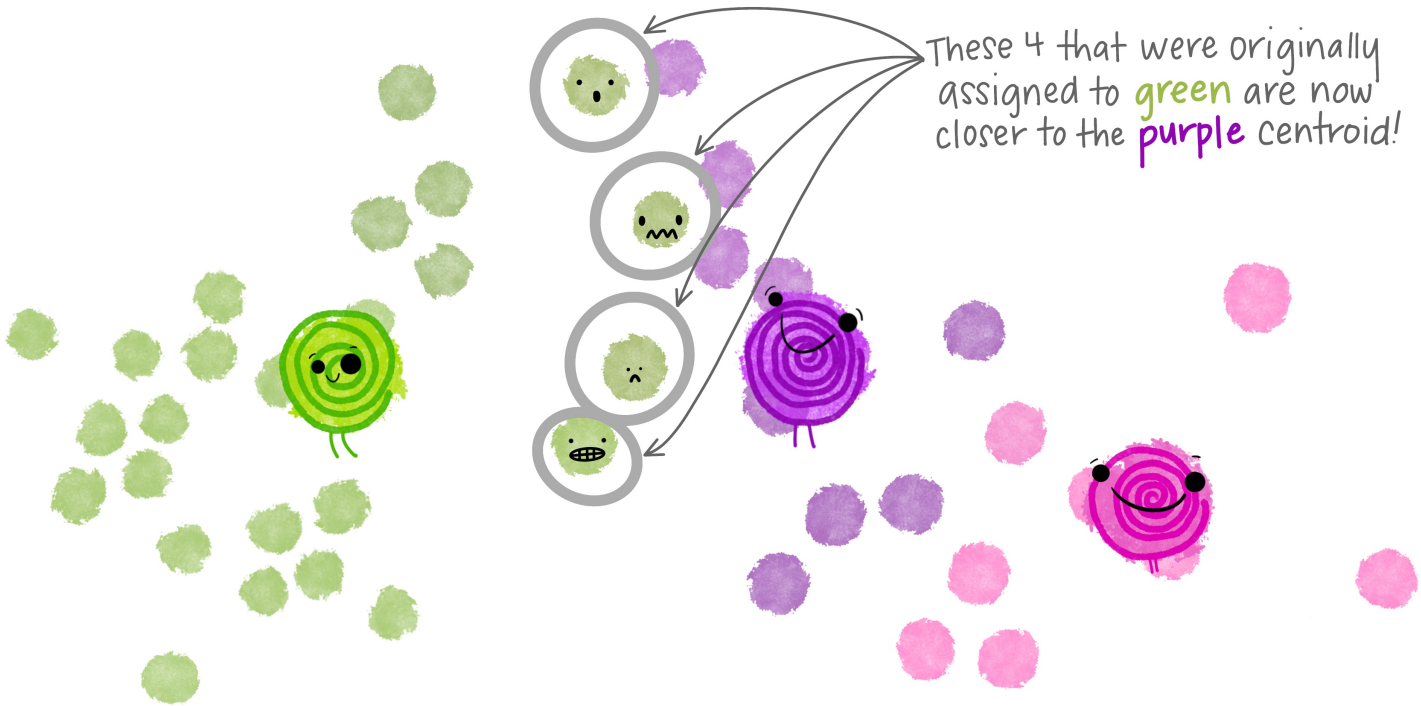

Then the centroid of each cluster is calculated based on all observations assigned to that cluster...



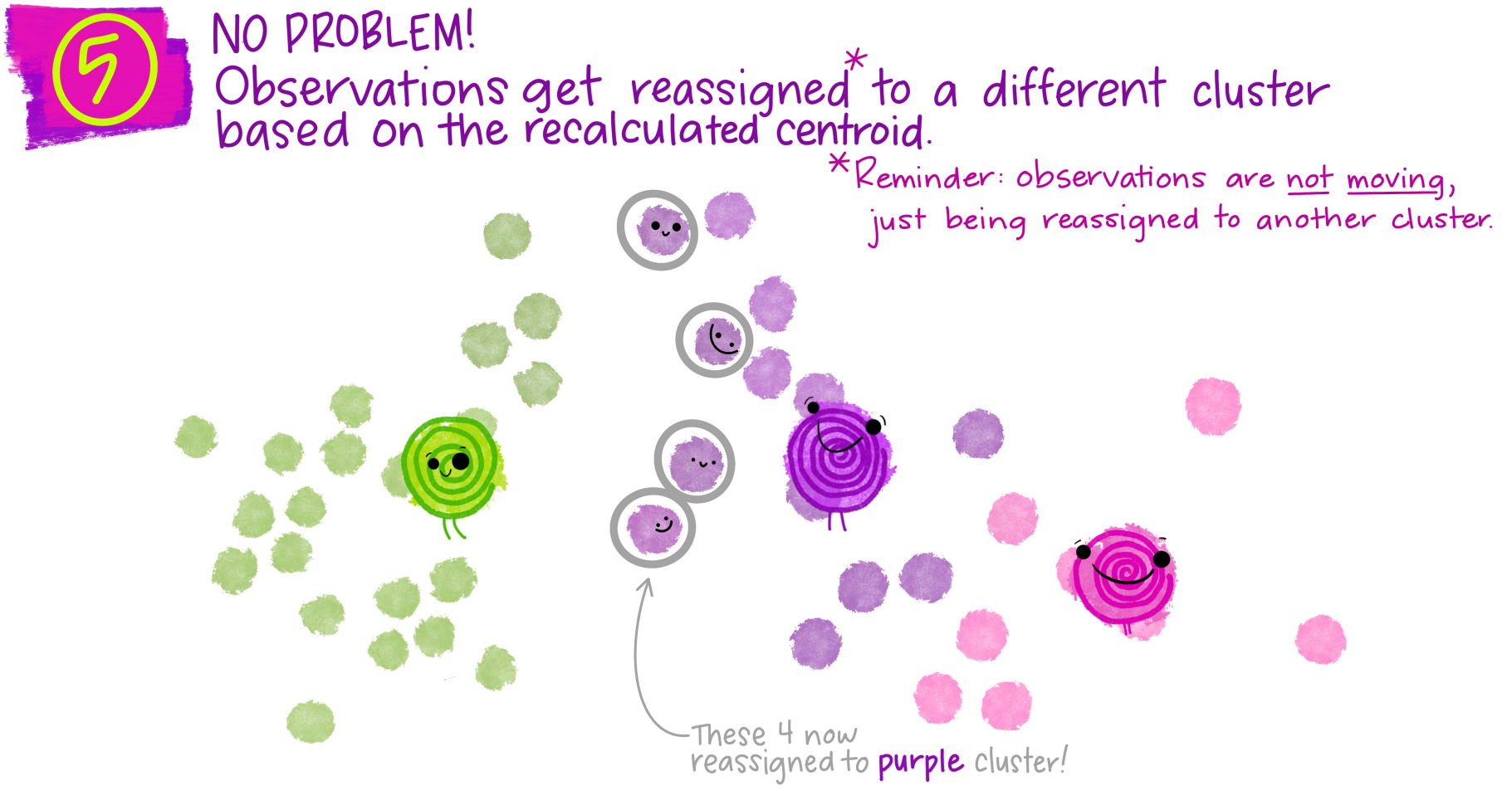

@allison_horst

# From Allison Horst (6/11)

UH OH. Now that the cluster centroids have moved, some of the observations are now closer to a different centroid!

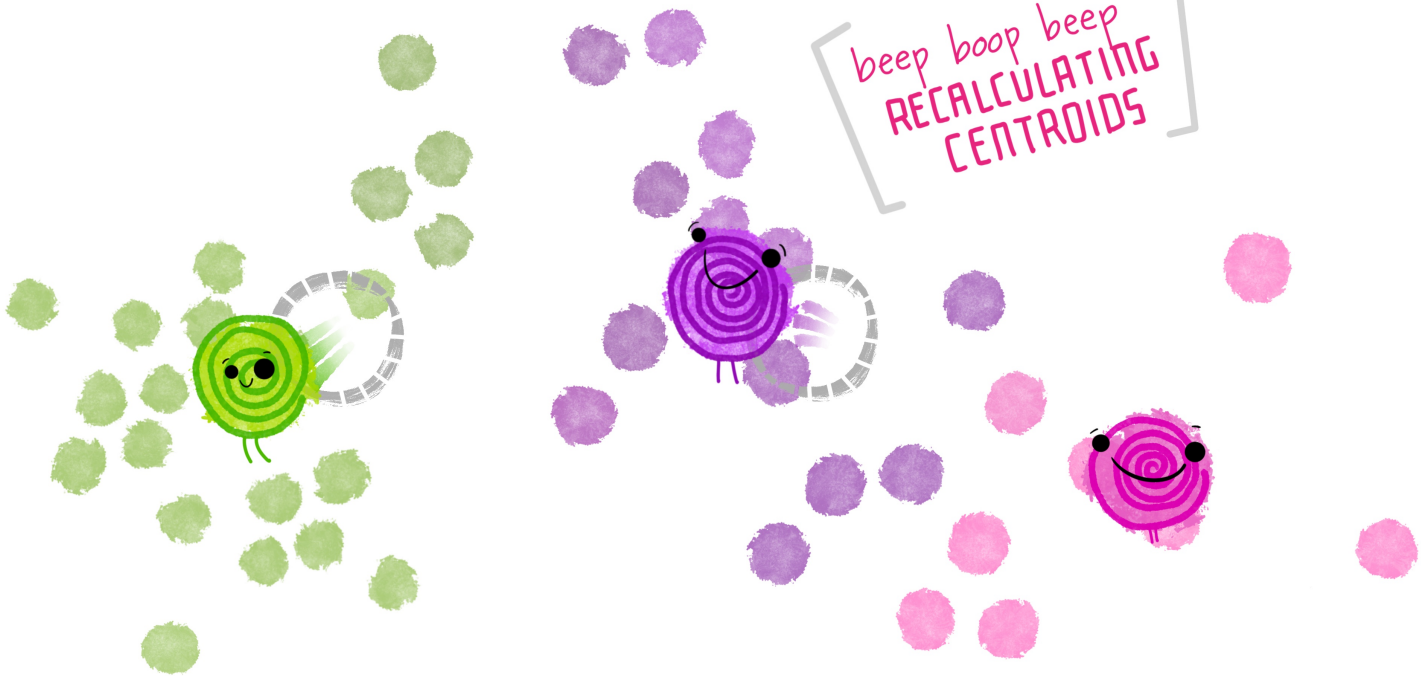


# From Allison Horst (7/11)

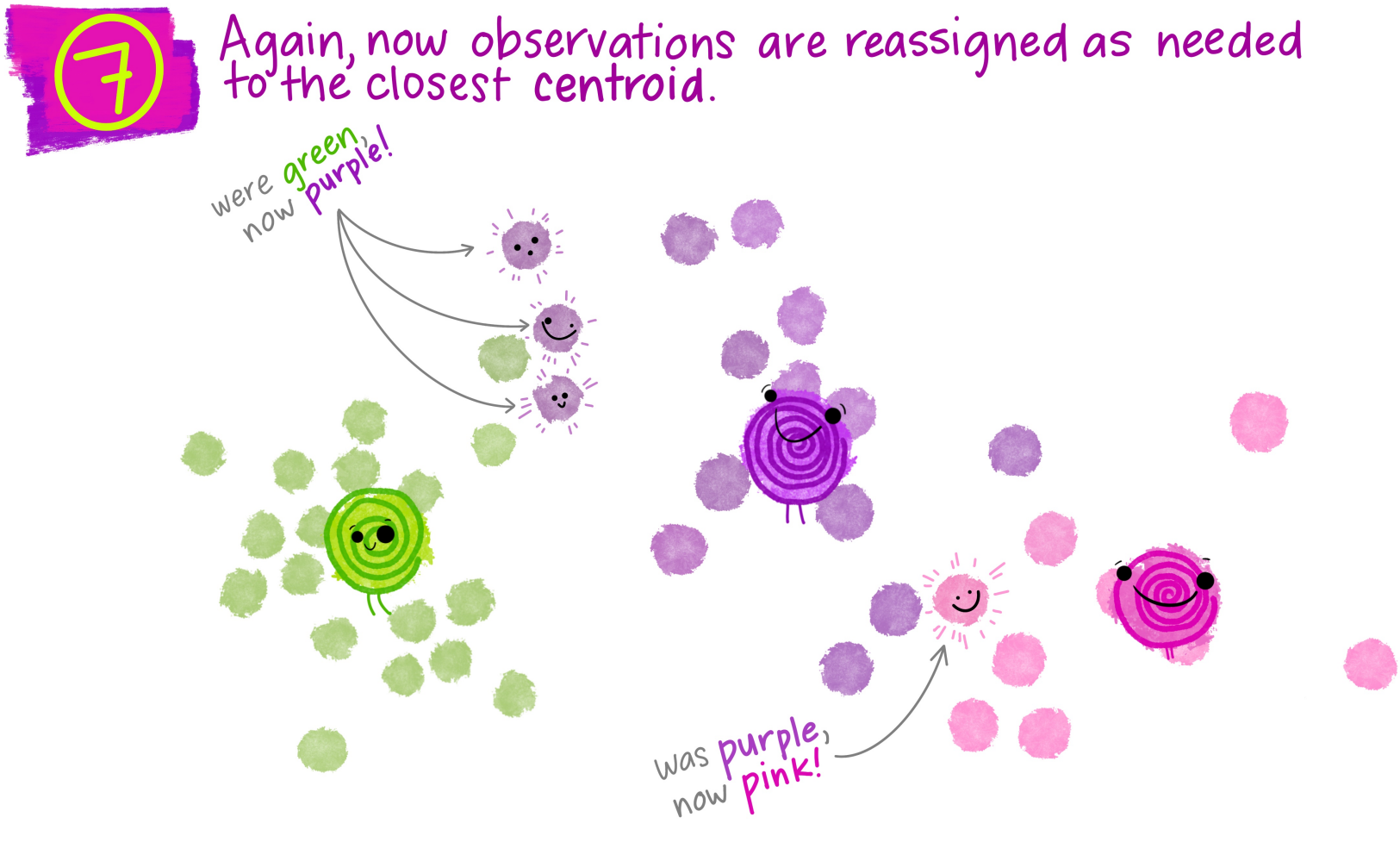


# From Allison Horst (8/11)


But now that observations have been reassigned, the centroids need to move again [recalculate centroids from updated clusters]



# From Allison Horst (9/11)



# From Allison Horst (10/11)

Then the centroid for each cluster is recalculated...

...which means observations will be reassigned...

@allison_horst

# From Allison Horst (11/11)

That iterative process of

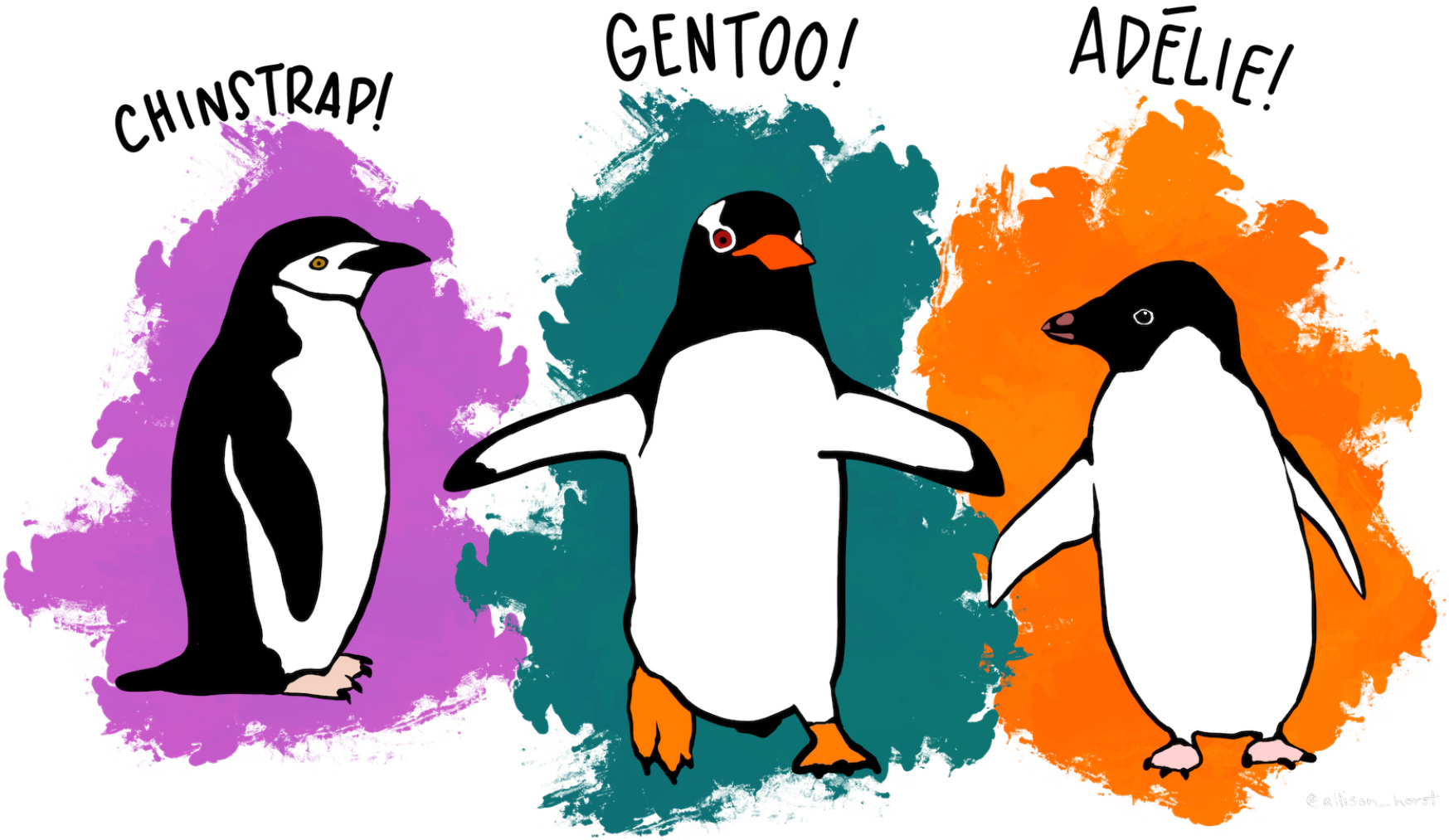
Recalculate cluster centroids
↳ Reassign observations to nearest centroid
↳ Recalculate cluster centroids
↳ Reassign observations to nearest centroid
↳ Recalculate cluster centroids
↳ Reassign observations to nearest centroid

↓

Continues until nothing is moving or being reassigned anymore!

@allison_horst

# The Palmer Penguins

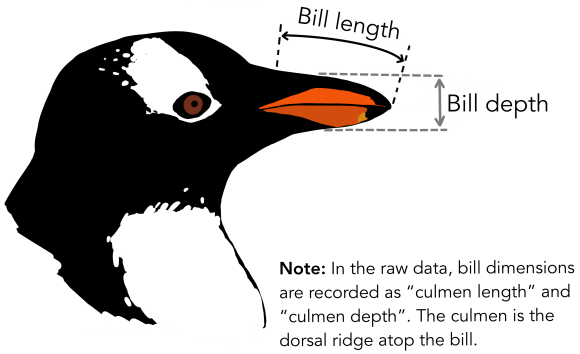


# The Palmer Penguins

```
pen <- penguins %>% ## from palmerpenguins package
    select(bill_length_mm, bill_depth_mm, species) %>%
    drop_na()

glimpse(pen)
```

```
Rows: 342
Columns: 3
$ bill_length_mm <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9,~
$ bill_depth_mm  <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8,~
$ species        <fct> Adelie, Adelie, Adelie, Adelie, Ade~
```

# What is being measured




**Note:** In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

# Getting Started

1. Create a data set with only numeric variables

```
pen1 <- pen %>% select(-species)
```

2. Scale each variable to have mean 0 and sd 1

```
pen1 <- pen1 %>% scale()

summary(pen1)
```

```
 bill_length_mm     bill_depth_mm
 Min.   :-2.16535   Min.   :-2.05144
 1st Qu.:-0.86031   1st Qu.:-0.78548
 Median : 0.09672   Median : 0.07537
 Mean   : 0.00000   Mean   : 0.00000
 3rd Qu.: 0.83854   3rd Qu.: 0.78430
 Max.   : 2.87166   Max.   : 2.20217
```

# Decide on the number of clusters, then run k-means

We know there are three types of penguins included, so let's try $k = 3$. We'll show the remainder of this output on the next slide.

```
pen_clust <- kmeans(pen1, centers = 3)
pen_clust
```

```
K-means clustering with 3 clusters of sizes 153, 64, 125

Cluster means:
  bill_length_mm bill_depth_mm
1     -0.9431819     0.5595723
2      1.1018368     0.7985421
3      0.5903143    -1.0937700

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
 [28] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
 [55] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
```

```
> pen_clust
K-means clustering with 3 clusters of sizes 153, 64, 125

Cluster means:
  bill_length_mm bill_depth_mm
1     -0.9431819     0.5595723
2      1.1018368     0.7985421
3      0.5903143    -1.0937700

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 [50] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [99] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[148] 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 1 3 3 3 3 3 3 3
[197] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 2 3 2 3 3
[246] 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2
[295] 1 2 3 2 2 2 2 2 2 2 1 2 1 2 2 3 2 2 3 2 2 2 2 3 2 2 2 2 2 2 3 2 2 2 1 2 3 2 2 2 2 3 3 2 1 2 2 2

Within cluster sum of squares by cluster:
[1] 87.98608 39.03381 59.35481
 (between_SS / total_SS =  72.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
>
```

# What do `tidy()` and `glance()` do?

```
tidy(pen_clust)
```

```
# A tibble: 3 x 5
  bill_length_mm bill_depth_mm  size withinss cluster
           <dbl>          <dbl> <int>    <dbl> <fct>
1         -0.943          0.560   153     88.0 1
2          1.10           0.799    64     39.0 2
3          0.590         -1.09    125     59.4 3
```

```
glance(pen_clust)
```

```
# A tibble: 1 x 4
  totss tot.withinss betweenss  iter
  <dbl>        <dbl>     <dbl> <int>
1   682         186.      496.     2
```

# What does augment() do?

```
pen_aug <- augment(pen_clust, pen)

glimpse(pen_aug)
```

```
Rows: 342
Columns: 4
$ bill_length_mm <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9,~
$ bill_depth_mm  <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8,~
$ species        <fct> Adelie, Adelie, Adelie, Adelie, Ade~
$ .cluster       <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
```

# Next Two Slides will show the results from . . .

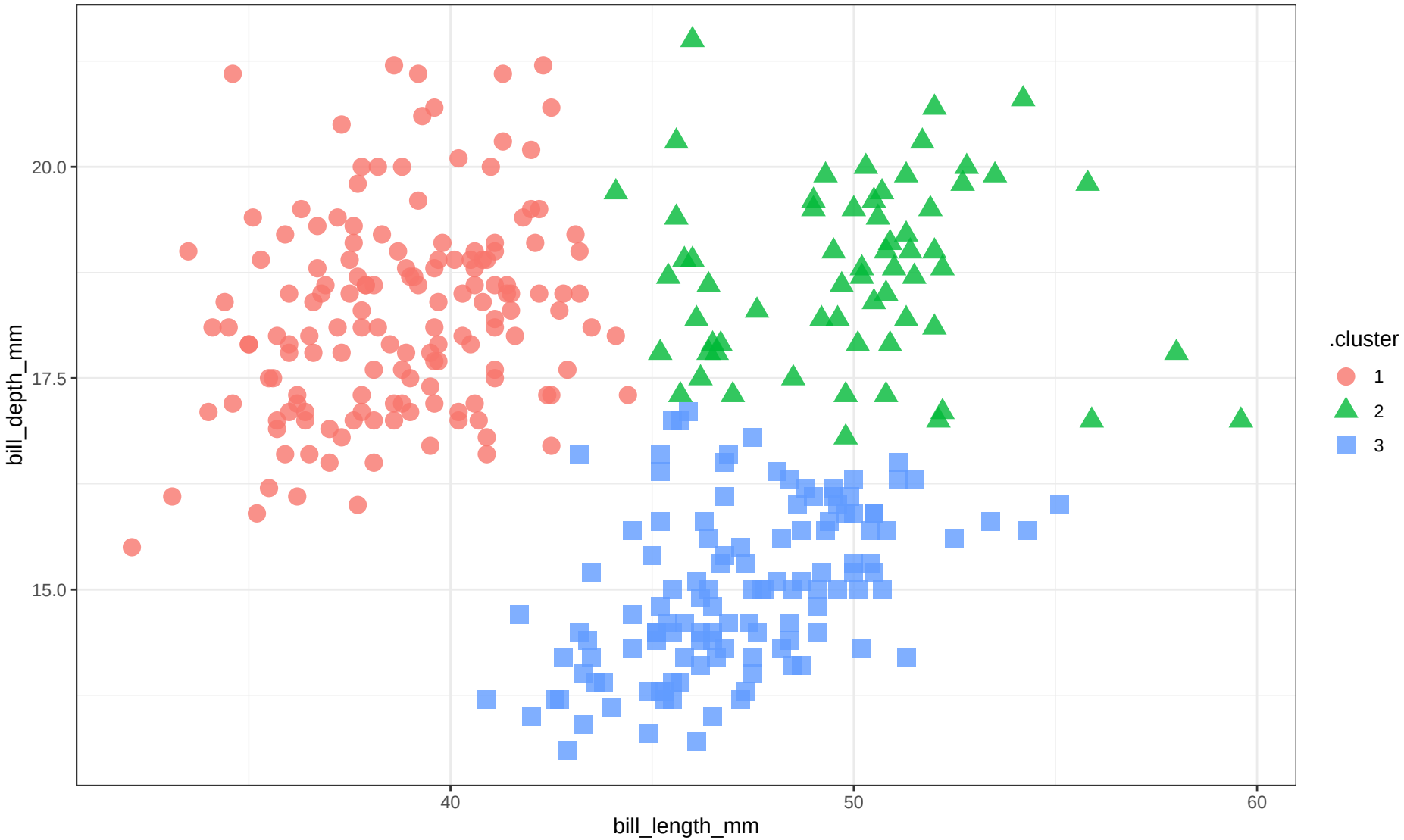
## Plot of clusters

```
ggplot(pen_aug,
       aes(x = bill_length_mm, y = bill_depth_mm)) +
    geom_point(aes(color = .cluster, shape = .cluster),
               size = 4, alpha = 0.8)
```

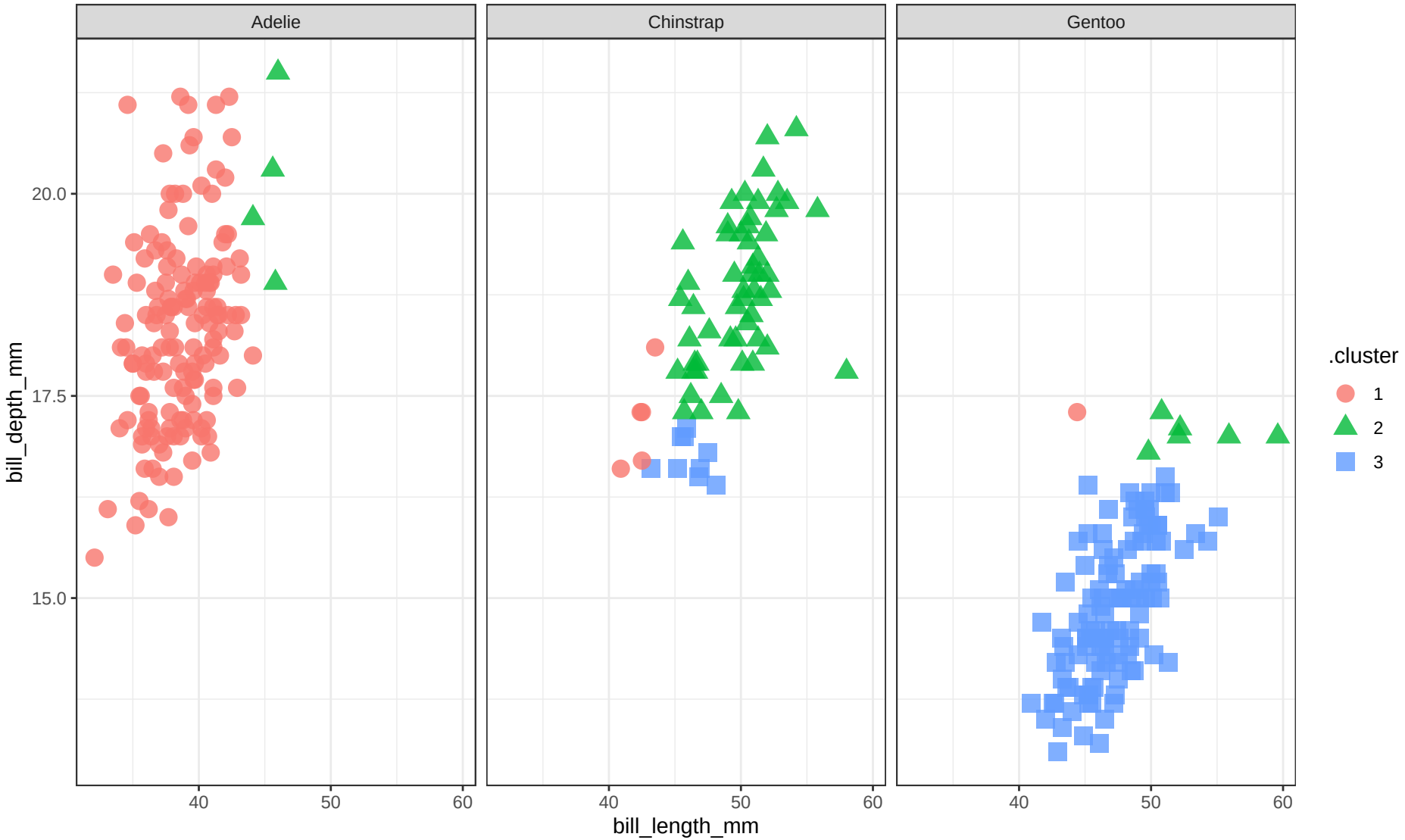
## Plot of clusters, faceted by species

```
ggplot(pen_aug,
       aes(x = bill_length_mm, y = bill_depth_mm)) +
    geom_point(aes(color = .cluster, shape = .cluster),
               size = 4, alpha = 0.8) +
    facet_wrap(~ species)
```

# Plot clusters



# Plot clusters, faceted by species?



# Comparison of Cluster Results to Original `species`

```
pen_aug %>% tabyl(.cluster, species)
```

| .cluster | Adelie | Chinstrap | Gentoo |
|---|---|---|---|
| 1 | 147 | 5 | 1 |
| 2 | 4 | 54 | 6 |
| 3 | 0 | 9 | 116 |

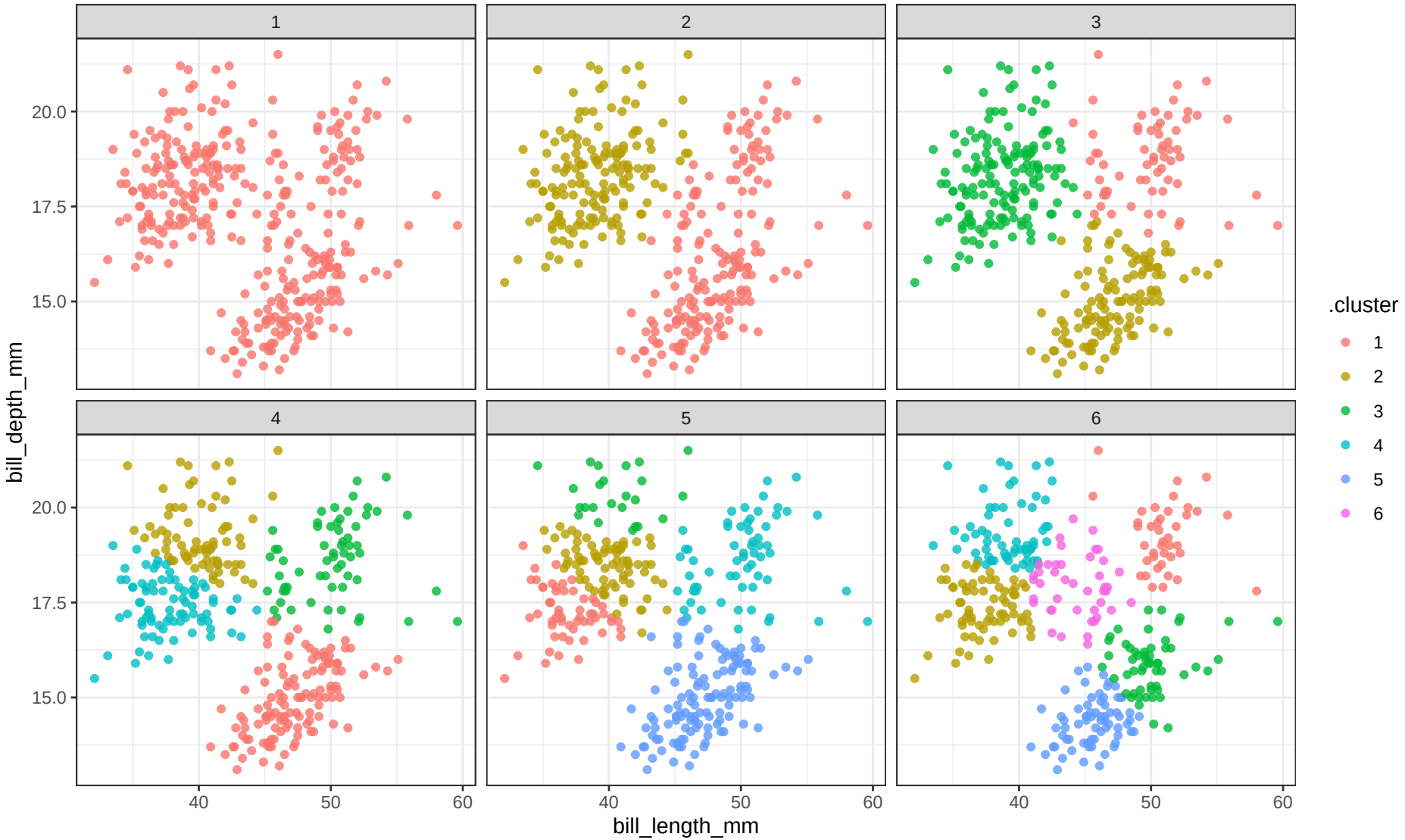
# What if we used a different number of clusters?

```
p_clusts <-
    tibble(k = 1:6) %>%
    mutate(
        pclust = purrr::map(k, ~ kmeans(pen1, .x)),
        ptidy = purrr::map(pclust, tidy),
        pglance = purrr::map(pclust, glance),
        paug = purrr::map(pclust, augment, pen))

p_clusters <- p_clusts %>% unnest(cols = c(ptidy))
p_assigns <- p_clusts %>% unnest(cols = c(paug))
p_clusterings <- p_clusts %>% unnest(cols = c(pglance))
```

# What do these clusters look like? (code)

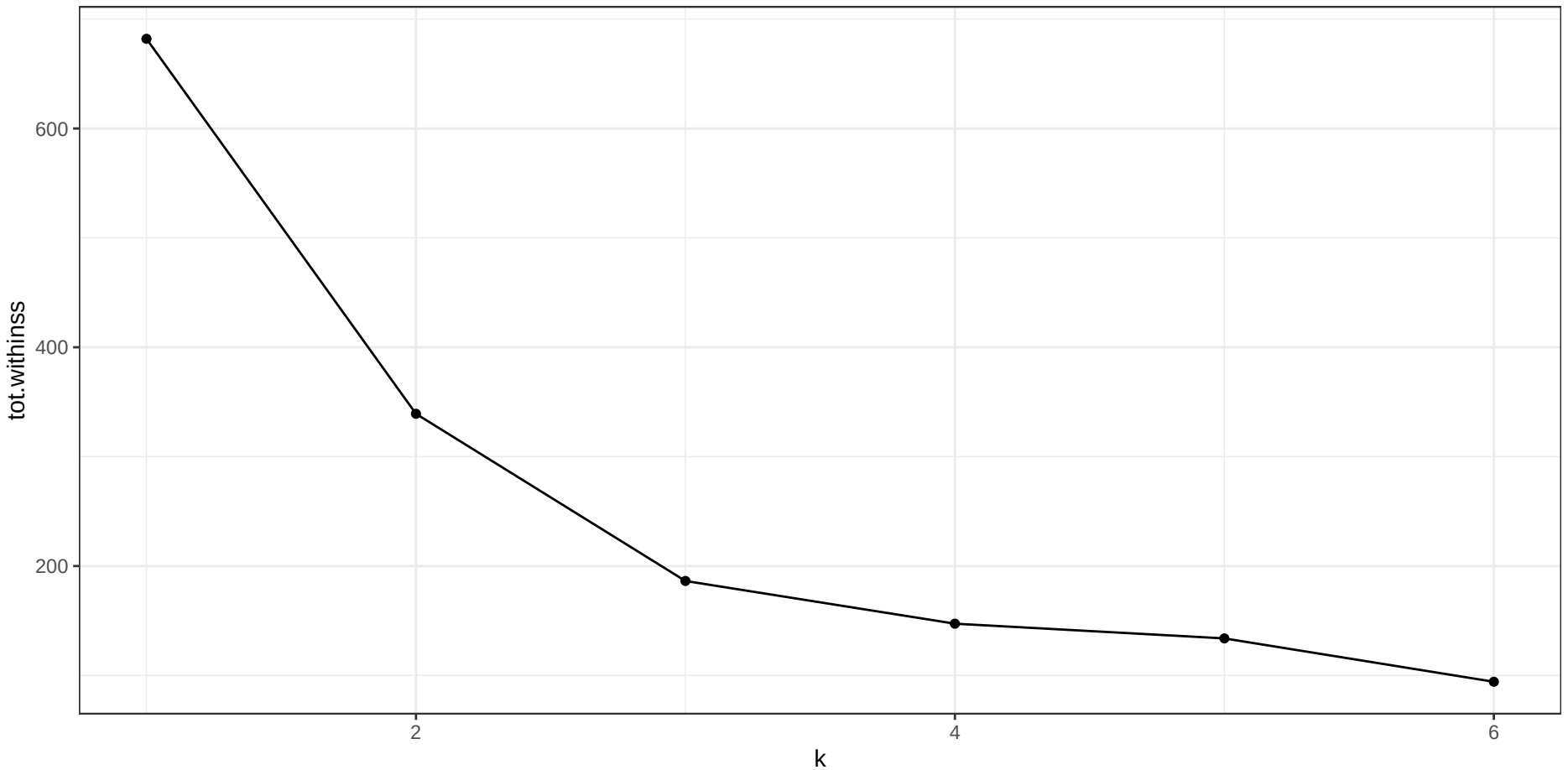
```
ggplot(p_assigns,
       aes(x = bill_length_mm, y = bill_depth_mm)) +
    geom_point(aes(color = .cluster), alpha = 0.8) +
    facet_wrap(~ k)
```

# What these clusters look like



# Scree Plot to determine best choice of k

```
ggplot(p_clusterings, aes(x = k, y = tot.withinss)) +
    geom_line() + geom_point()
```



# Could we consider other penguin characteristics?

- Only if they are numeric.

```
pen_new <- penguins %>%
    select(flipper_length_mm, body_mass_g,
           species, island) %>%
    drop_na()

pen2 <- pen_new %>% select(-species, -island) %>% scale()

pen_clust2 <- kmeans(pen2, centers = 3)
```

# Augment with the clusters, and tabulate

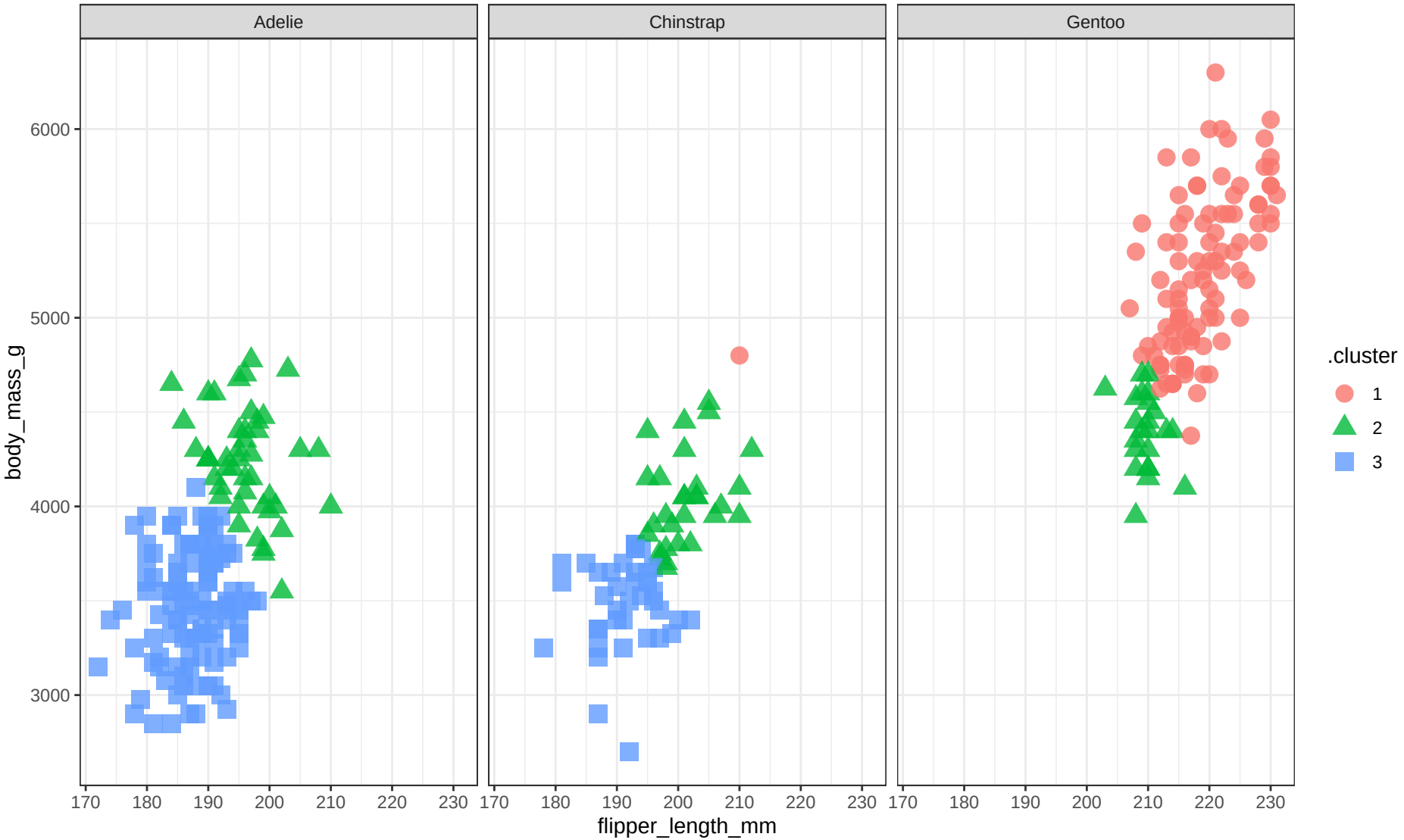
```
pen_aug2 <- augment(pen_clust2, pen_new)
pen_aug2 %>% tabyl(.cluster, species)
```

| .cluster | Adelie | Chinstrap | Gentoo |
|---|---|---|---|
| 1 | 0 | 1 | 100 |
| 2 | 45 | 29 | 23 |
| 3 | 106 | 38 | 0 |

```
pen_aug2 %>% tabyl(.cluster, island)
```

| .cluster | Biscoe | Dream | Torgersen |
|---|---|---|---|
| 1 | 100 | 1 | 0 |
| 2 | 35 | 46 | 16 |
| 3 | 32 | 77 | 35 |

# Plot second set of clusters, facet by species



# Principal Components and the Palmer Penguins

# Principal Components Analysis (PCA)

The goal here is to summarize the information contained in a large set of variables by means of a smaller set of "summary index" values that can be more easily visualized and analyzed.

Statistically, PCA finds lines, planes and hyper-planes in K-dimensional space that approximate the data as well as possible, specifically by identifying components that maximize the variance of the projected data.

> *. . . in regression analysis, the larger the number of explanatory variables allowed, the greater is the chance of overfitting the model, producing conclusions that fail to generalise to other datasets. One approach, especially when there are strong correlations between different possible explanatory variables, is to reduce them to a few principal components and then run the regression against them, a method called principal component regression. (Wikipedia)*

# Principal Components Analysis (PCA) on the Penguins?

Sure. See https://allisonhorst.github.io/palmerpenguins/articles/pca.html which is the basis for my next few slides.

We'll build this within the `tidymodels` framework, and first use a few `recipe` steps to pre-process the data for PCA, specifically:

1. remove all NA values
2. center and scale all predictors

# Penguin PCA Recipe

```
penguin_recipe <-
    recipe(~., data = penguins) %>%
    update_role(species, island, sex,
                year, new_role = "id") %>%
    step_naomit(all_predictors()) %>%
    step_normalize(all_predictors()) %>%
    step_pca(all_predictors(), id = "pca") %>%
    prep()

penguin_pca <-
    penguin_recipe %>%
    tidy(id = "pca")
```

# Results for Penguin PCA

```
penguin_pca
```

```
# A tibble: 16 x 4
   terms                value component id
   <chr>                <dbl> <chr>     <chr>
 1 bill_length_mm       0.455 PC1       pca
 2 bill_depth_mm       -0.400 PC1       pca
 3 flipper_length_mm    0.576 PC1       pca
 4 body_mass_g          0.548 PC1       pca
 5 bill_length_mm      -0.597 PC2       pca
 6 bill_depth_mm       -0.798 PC2       pca
 7 flipper_length_mm -0.00228 PC2       pca
 8 body_mass_g         -0.0844 PC2       pca
 9 bill_length_mm      -0.644 PC3       pca
10 bill_depth_mm        0.418 PC3       pca
11 flipper_length_mm    0.232 PC3       pca
12 body_mass_g          0.597 PC3       pca
```

# An Easier-to-Use Presentation

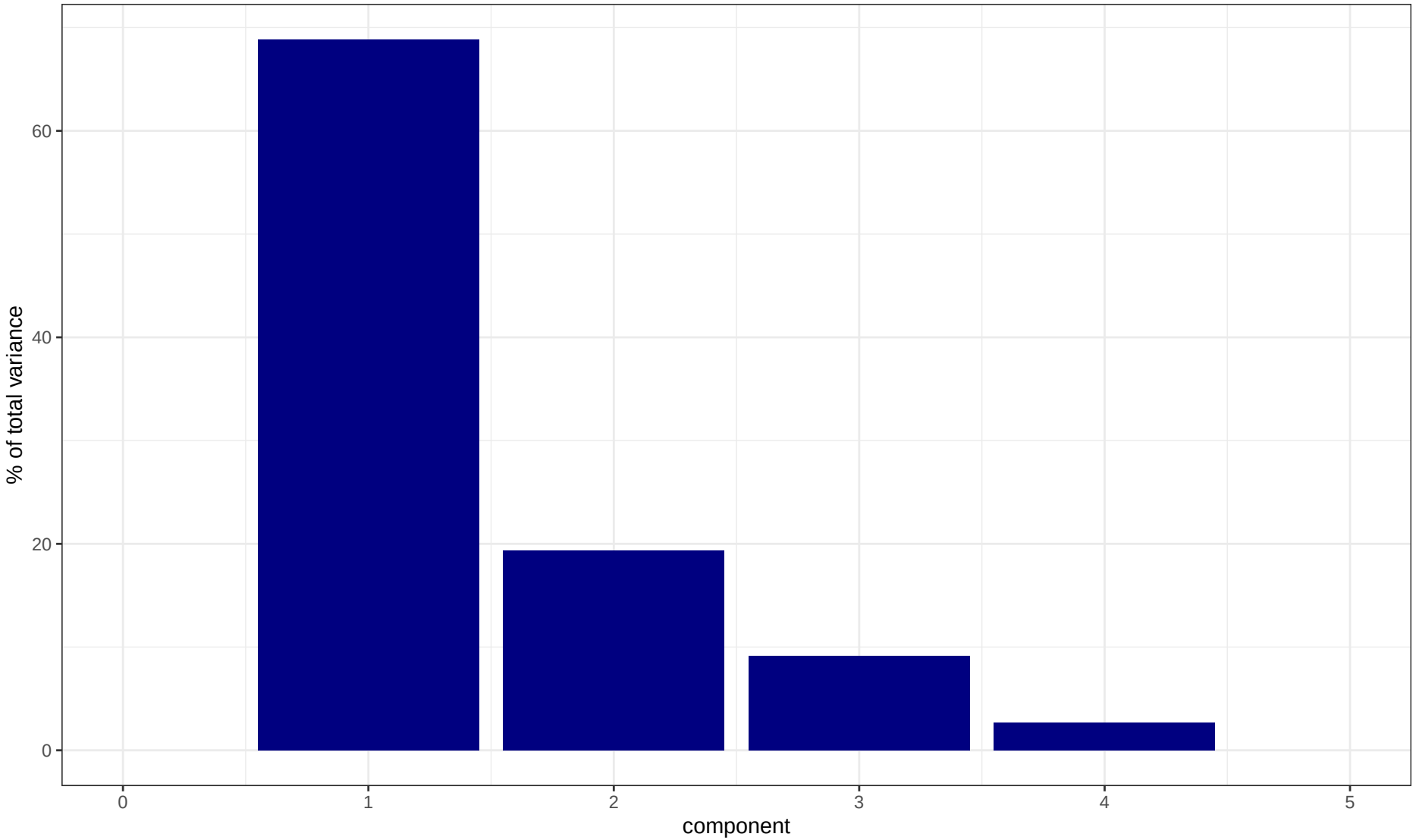
```
penguins %>%
  dplyr::select(body_mass_g, ends_with("_mm")) %>%
  tidyr::drop_na() %>%
  scale() %>%
  prcomp() %>%
  .$rotation
```

```
                     PC1         PC2        PC3
body_mass_g        0.5483502 0.084362920 -0.5966001
bill_length_mm     0.4552503 0.597031143  0.6443012
bill_depth_mm     -0.4003347 0.797766572 -0.4184272
flipper_length_mm  0.5760133 0.002282201 -0.2320840
                     PC4
body_mass_g       -0.5798821
bill_length_mm    -0.1455231
bill_depth_mm      0.1679860
flipper_length_mm  0.7837987
```

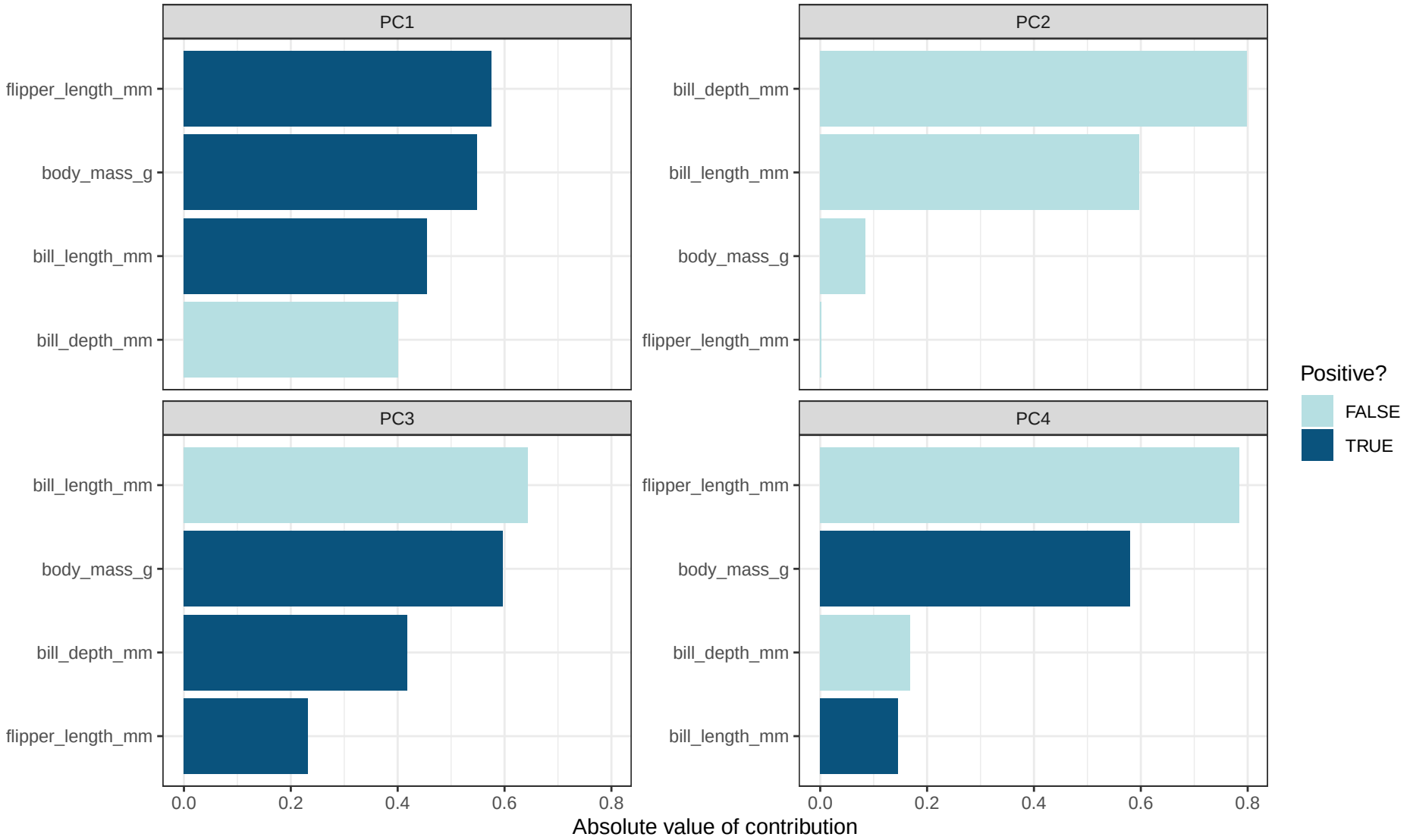
# How much variance does each component account for?

```
penguin_recipe %>%
  tidy(id = "pca", type = "variance") %>%
  dplyr::filter(terms == "percent variance") %>%
  ggplot(aes(x = component, y = value)) +
  geom_col(fill = "navy") +
  xlim(c(0, 5)) +
  ylab("% of total variance")
```

# How much variance does each component account for?



# Plot PCA Loadings



# Plot PCA Loadings (code)

```
penguin_pca %>%
  mutate(terms = tidytext::reorder_within(terms,
                                          abs(value),
                                          component)) %>%
  ggplot(aes(abs(value), terms, fill = value > 0)) +
  geom_col() +
  facet_wrap(~component, scales = "free_y") +
  tidytext::scale_y_reordered() +
  scale_fill_manual(values = c("#b6dfe2", "#0A537D")) +
  labs(
    x = "Absolute value of contribution",
    y = NULL, fill = "Positive?"
  )
```

# Another Example of K-Means Clustering

# The USArrests data, from base R

The USArrests data set from base R is a table containing the number of arrests per 100,000 residents in each US state in 1973 for Murder, Assault and Rape, along with the percentage of the population in each state that lives in urban areas, called UrbanPop.

```
USArr73 <- USArrests %>%
    mutate(state = row.names(USArrests)) %>%
    relocate(state) %>%
    as_tibble()

n_miss(USArr73)
```

```
[1] 0
```

# The cleaned USArr73 tibble

```
USArr73
```

```
# A tibble: 50 x 5
   state       Murder Assault UrbanPop  Rape
   <chr>        <dbl>   <int>    <int> <dbl>
 1 Alabama       13.2     236       58  21.2
 2 Alaska        10       263       48  44.5
 3 Arizona        8.1     294       80  31
 4 Arkansas       8.8     190       50  19.5
 5 California     9       276       91  40.6
 6 Colorado       7.9     204       78  38.7
 7 Connecticut    3.3     110       77  11.1
 8 Delaware       5.9     238       72  15.8
 9 Florida       15.4     335       80  31.9
10 Georgia       17.4     211       60  25.8
# ... with 40 more rows
```

# Scale each variable to have mean 0 and sd 1

```
USArr73_s <- USArr73 %>% select(-state) %>% scale()

row.names(USArr73_s) <- row.names(USArrests)

head(USArr73_s)
```

```
               Murder   Assault   UrbanPop         Rape
Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
Arizona    0.07163341 1.4788032  0.9989801  1.042878388
Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
California 0.27826823 1.2628144  1.7589234  2.067820292
Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

# Fit 3-means clustering

```
kclust <- kmeans(USArr73_s, centers = 3)
kclust
```

```
K-means clustering with 3 clusters of sizes 13, 29, 8

Cluster means:
      Murder    Assault    UrbanPop         Rape
1  0.6950701  1.0394414  0.72263703  1.27693964
2 -0.7010700 -0.7071522 -0.09924526 -0.57773737
3  1.4118898  0.8743346 -0.81452109  0.01927104

Clustering vector:
      Alabama        Alaska       Arizona      Arkansas
            3             1             1             3
   California      Colorado   Connecticut      Delaware
            1             1             2             2
      Florida       Georgia        Hawaii         Idaho
            1             3             2             2
```

# Complete output from `kclust`

```
> kclust
K-means clustering with 3 clusters of sizes 8, 29, 13

Cluster means:
      Murder    Assault    UrbanPop        Rape
1  1.4118898  0.8743346 -0.81452109  0.01927104
2 -0.7010700 -0.7071522 -0.09924526 -0.57773737
3  0.6950701  1.0394414  0.72263703  1.27693964

Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California       Colorado
             1              3              3              1              3              3
   Connecticut       Delaware        Florida        Georgia         Hawaii          Idaho
             2              2              3              1              2              2
      Illinois        Indiana           Iowa         Kansas       Kentucky      Louisiana
             3              2              2              2              2              1
         Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi
             2              3              2              3              2              1
      Missouri        Montana       Nebraska         Nevada  New Hampshire     New Jersey
             3              2              2              3              2              2
    New Mexico       New York North Carolina   North Dakota           Ohio       Oklahoma
             3              3              1              2              2              2
        Oregon   Pennsylvania   Rhode Island South Carolina   South Dakota      Tennessee
             2              2              2              1              2              1
         Texas           Utah        Vermont       Virginia     Washington  West Virginia
             3              2              2              2              2              2
     Wisconsin        Wyoming
             2              2

Within cluster sum of squares by cluster:
[1]  8.316061 53.354791 19.922437
 (between_SS / total_SS =  58.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

# Summary of 3-means clustering

```
summary(kclust)
```

```
             Length Class  Mode
cluster      50     -none- numeric
centers      12     -none- numeric
totss         1     -none- numeric
withinss      3     -none- numeric
tot.withinss  1     -none- numeric
betweenss     1     -none- numeric
size          3     -none- numeric
iter          1     -none- numeric
ifault        1     -none- numeric
```

# Use augment() from broom

```
augment(kclust, USArr73_s)
```

```
# A tibble: 50 x 6
   .rownames    Murder Assault UrbanPop     Rape .cluster
   <chr>         <dbl>   <dbl>    <dbl>    <dbl> <fct>
 1 Alabama       1.24    0.783   -0.521 -0.00342 3
 2 Alaska        0.508   1.11    -1.21   2.48    1
 3 Arizona       0.0716  1.48     0.999  1.04    1
 4 Arkansas      0.232   0.231   -1.07  -0.185   3
 5 California    0.278   1.26     1.76   2.07    1
 6 Colorado      0.0257  0.399    0.861  1.86    1
 7 Connecticut  -1.03   -0.729    0.792 -1.08    2
 8 Delaware     -0.433   0.807    0.446 -0.580   2
 9 Florida       1.75    1.97     0.999  1.14    1
10 Georgia       2.21    0.483   -0.383  0.488   3
# ... with 40 more rows
```

# Use tidy() and glance() from broom

```
tidy(kclust)
```

```
# A tibble: 3 x 7
  Murder Assault UrbanPop    Rape  size withinss cluster
   <dbl>   <dbl>    <dbl>   <dbl> <int>    <dbl> <fct>
1  0.695   1.04    0.723   1.28      13     19.9  1
2 -0.701  -0.707  -0.0992 -0.578     29     53.4  2
3  1.41    0.874  -0.815   0.0193     8      8.32 3
```

```
glance(kclust)
```

```
# A tibble: 1 x 4
  totss tot.withinss betweenss  iter
  <dbl>        <dbl>     <dbl> <int>
1   196         81.6      114.     2
```

# Use 1-9 clusters now

```
kclusts <-
  tibble(k = 1:9) %>%
  mutate(
    kclust = purrr::map(k, ~ kmeans(USArr73_s, .x)),
    tidied = purrr::map(kclust, tidy),
    glanced = purrr::map(kclust, glance),
    augmented = purrr::map(kclust, augment, USArr73_s)
  )
```

```
clusters <- kclusts %>% unnest(cols = c(tidied))
assignments <- kclusts %>% unnest(cols = c(augmented))
clusterings <- kclusts %>% unnest(cols = c(glanced))
```

# Plots on Next Three Slides (code)

## Plot 1

```
ggplot(assignments, aes(x = UrbanPop, y = Murder)) +
    geom_point(aes(color = .cluster), alpha = 0.8) +
    facet_wrap(~ k)
```

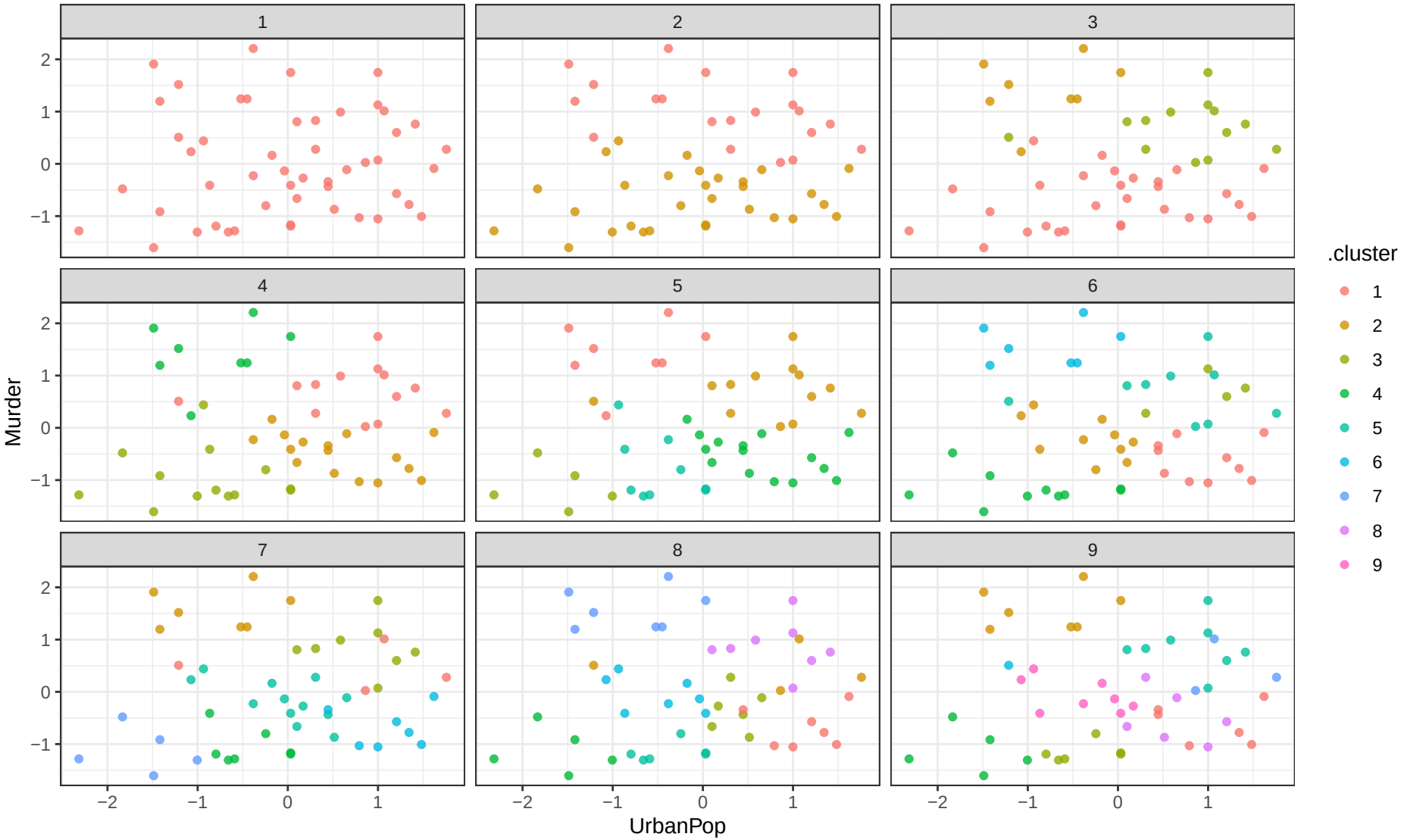
## Plot 2

```
ggplot(assignments, aes(x = UrbanPop, y = Murder)) +
    geom_point(aes(color = .cluster), alpha = 0.8) +
    geom_point(data = clusters, size = 10, shape = "x") +
    facet_wrap(~ k)
```

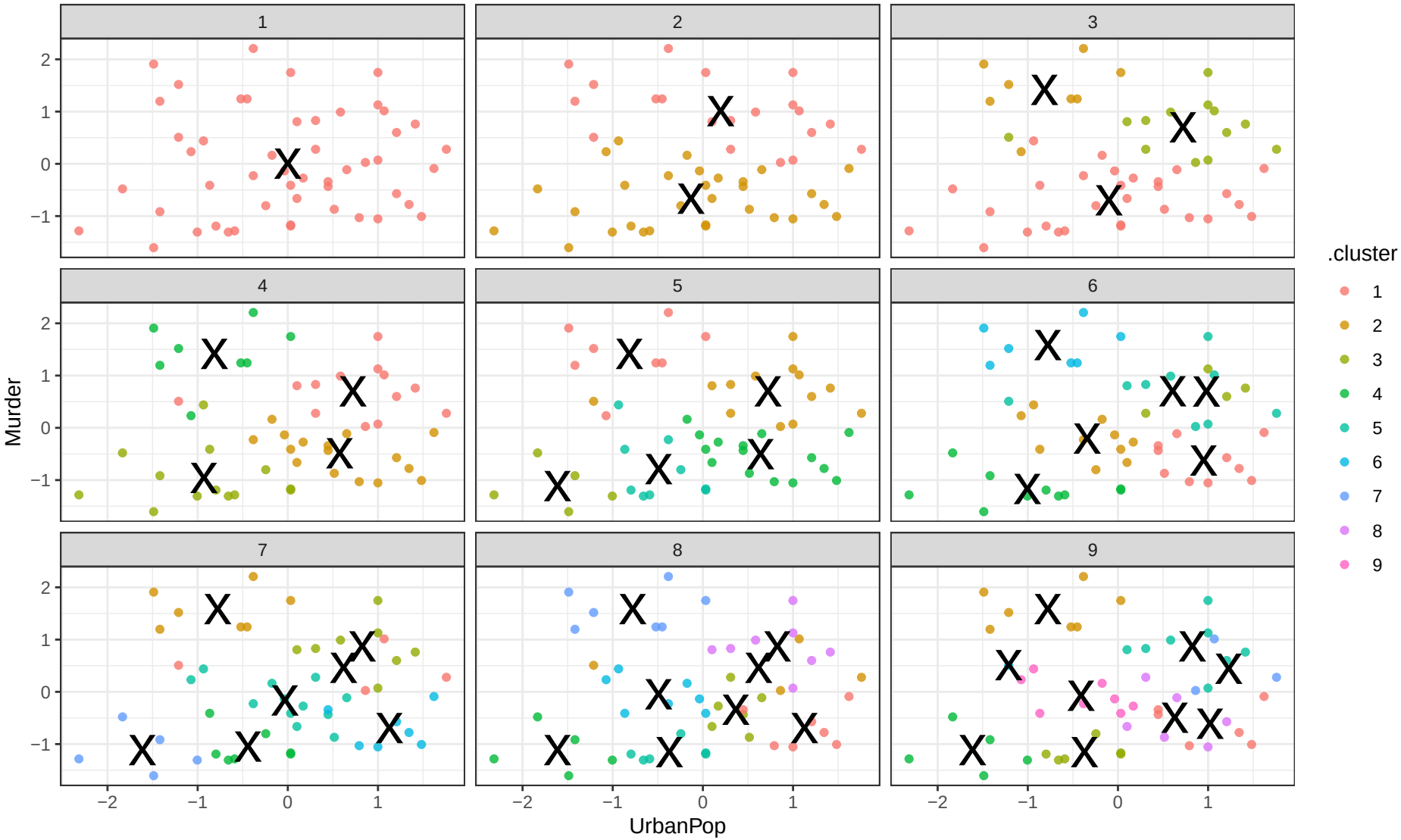
## Plot 3

```
ggplot(clusterings, aes(x = k, y = tot.withinss)) +
    geom_line() + geom_point()
```

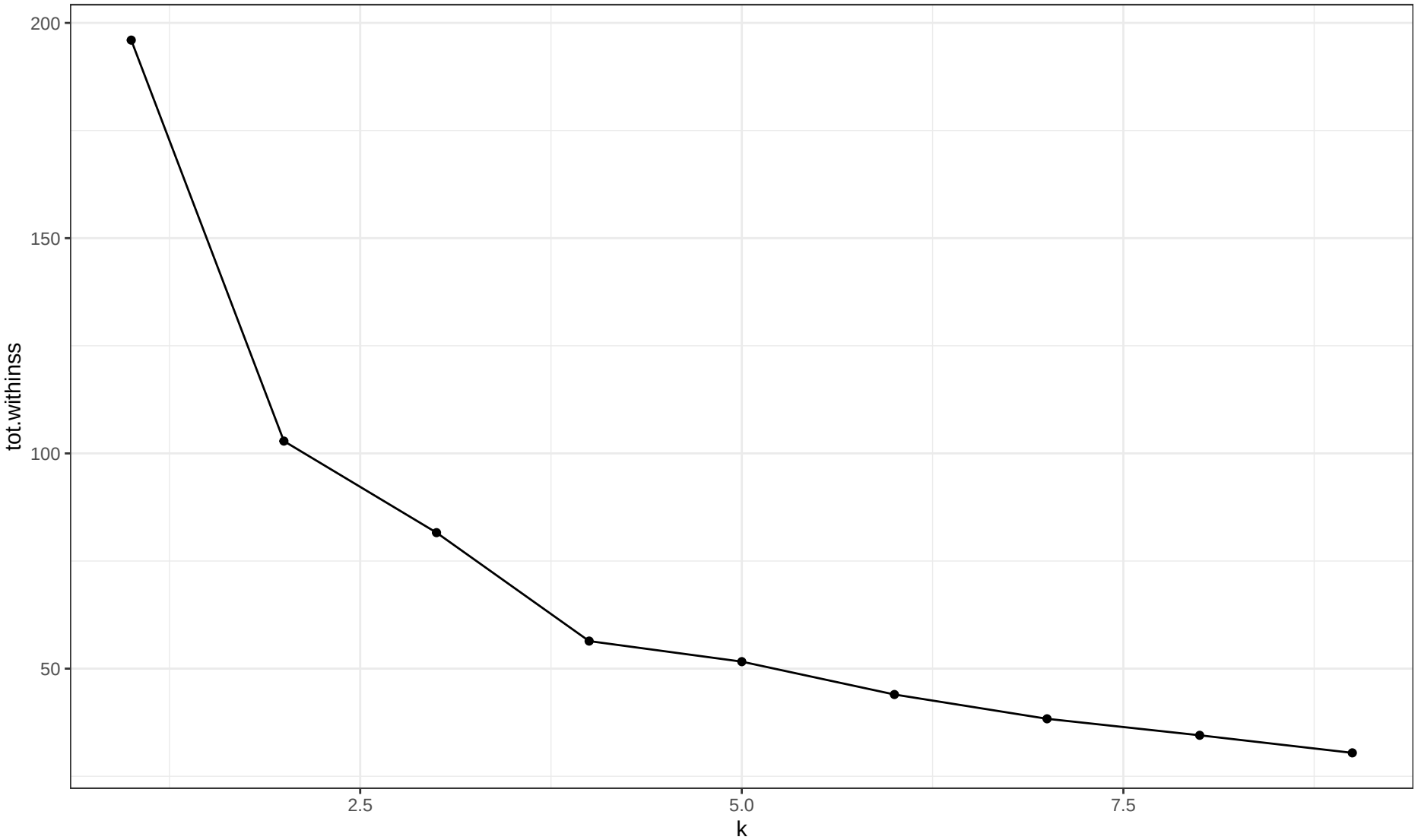
# Plot 1



# Plot 2



# Plot 3



# Next Time

Quiz 2 is due Tuesday at 9 AM (in the morning.)

- No class next Tuesday 2022-04-19.

Last Class is Thursday 2022-04-21. We'll discuss lots of "little things" and dispense some advice for going forward and making the world a better place.