Unsupervised Learning (clustering)

K Arnold

Q&A

- Midterm average was 85%.

> Can we use inference in projects?

- Many people asked "which variables are important for predicting?" -- Wednesday approach works for that.
- Others want to know specific relationships. See the plot strategy in Lab 10, and the `infer` package vignette from prep reading.

> How much difference in MAE (or RMSE, specificity, etc.) is meaningful?

- Look at the confidence intervals.
- Depends on your problem!

Project Logistics

- No final exam, just project.
- Next milestone: by Thanksgiving, have some initial EDA
- Proposal feedback is in-progress

Other logistics

- No quiz, no homework this week; just work on projects
- Midterm project feedback is coming...

> Which other packages do you use? (besides `tidyverse`)

- `glue`: for constructing strings `"{nrows(data)} rows"`
- `patchwork` for arranging plots
- `knitr` for `include_graphics` (or just `![](image-url)`)

Unsupervised Learning

- So far we have been doing *supervised* learning, where have a *target* we're trying to predict.
  - "How much will these homes sell for?"
  - "How long will this person spend watching this video?"
- **Unsupervised** learning works when we don't have an exact target to predict, or we want to explore relationships in the data.
  - "What general types of homes are on the market right now?"
  - "What are some different segments of our customer base?"
  - "Are there distinct types of Covid-19 symptoms?"
- **Clustering** is one very common type of unsupervised learning.

Clustering

Goal: put observations into groups

- Those in the *same* group should be *similar to each other*
- Those in *different* groups should be *different*.

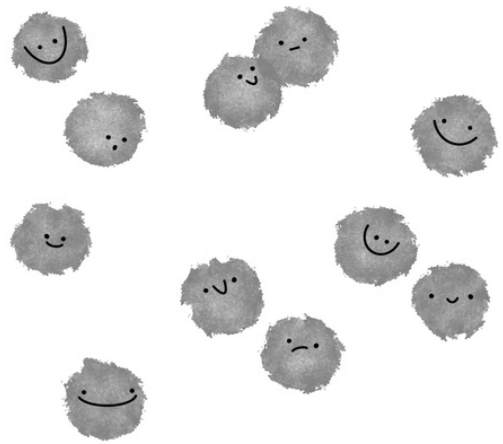
Crucial questions:

- How many groups?
- How do we define "similar" / "different"?

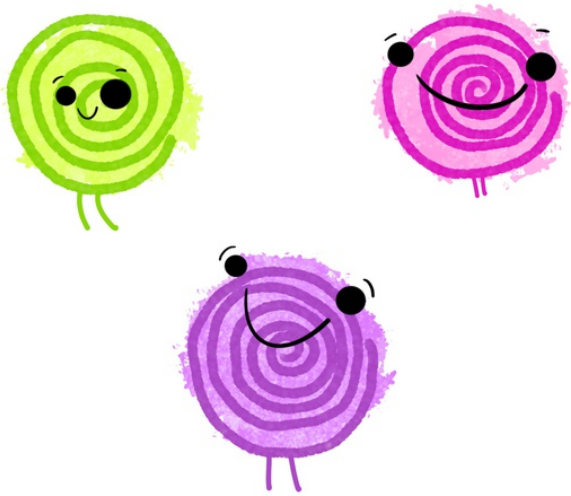
k-means clustering

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

OBSERVATIONS

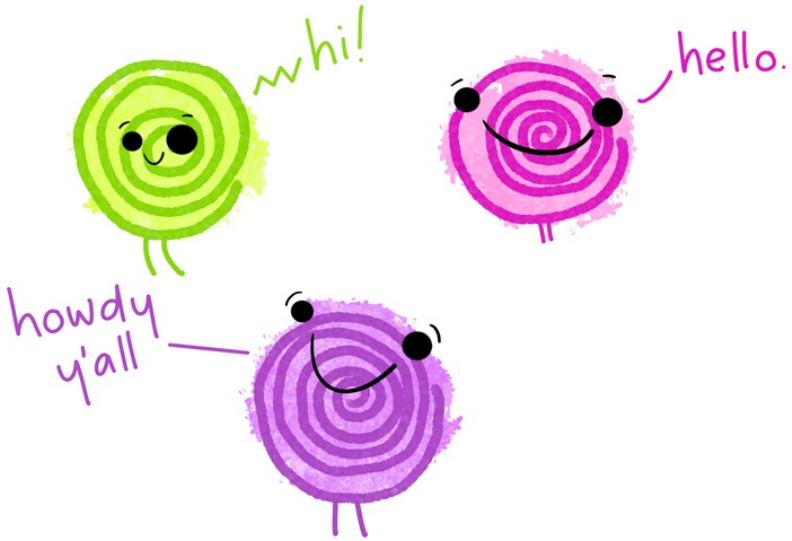


cluster CENTROIDS

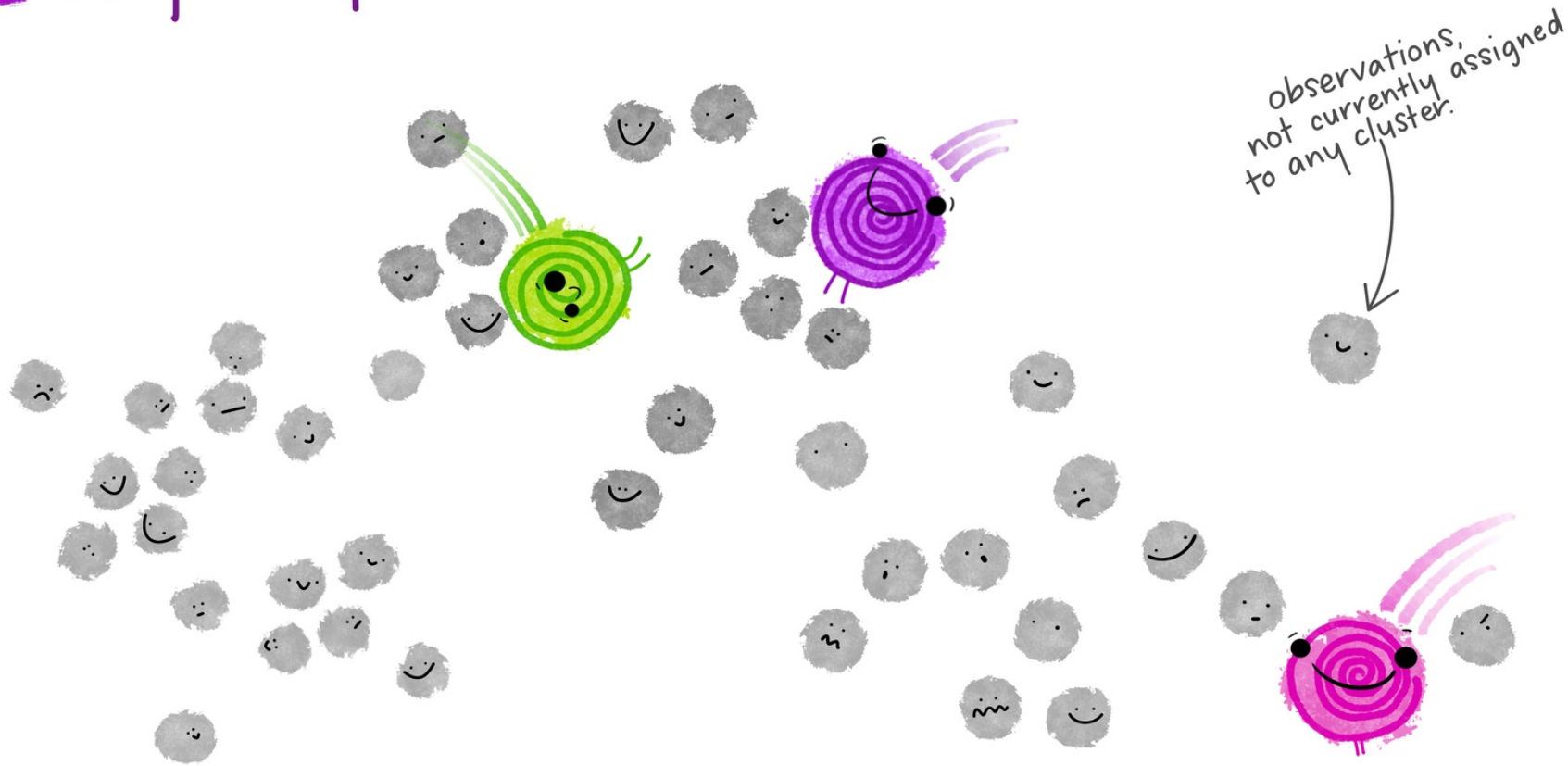


1 Specify the number of clusters (in this example, k = 3).

Then imagine k cluster centroids are created.

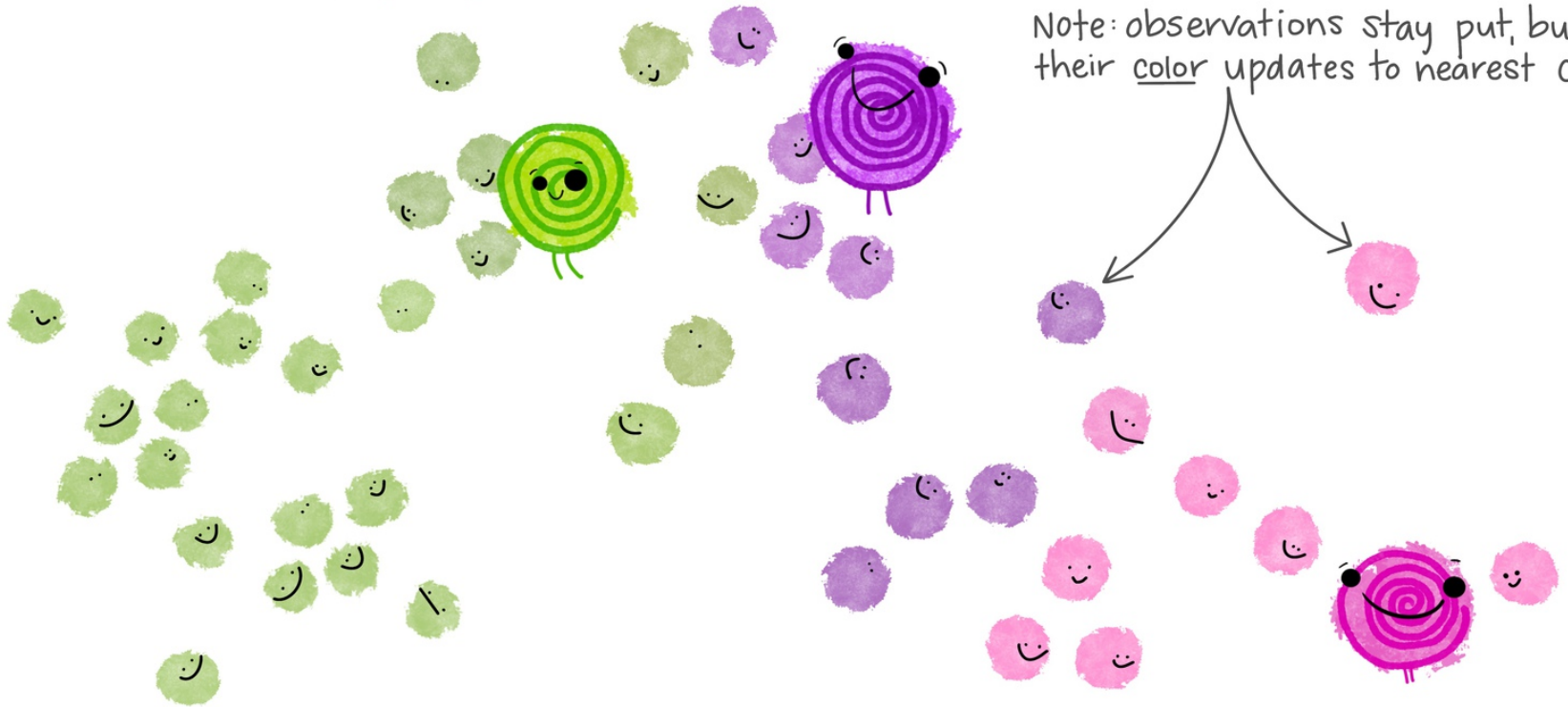


2 Those k centroids get randomly placed in your space.



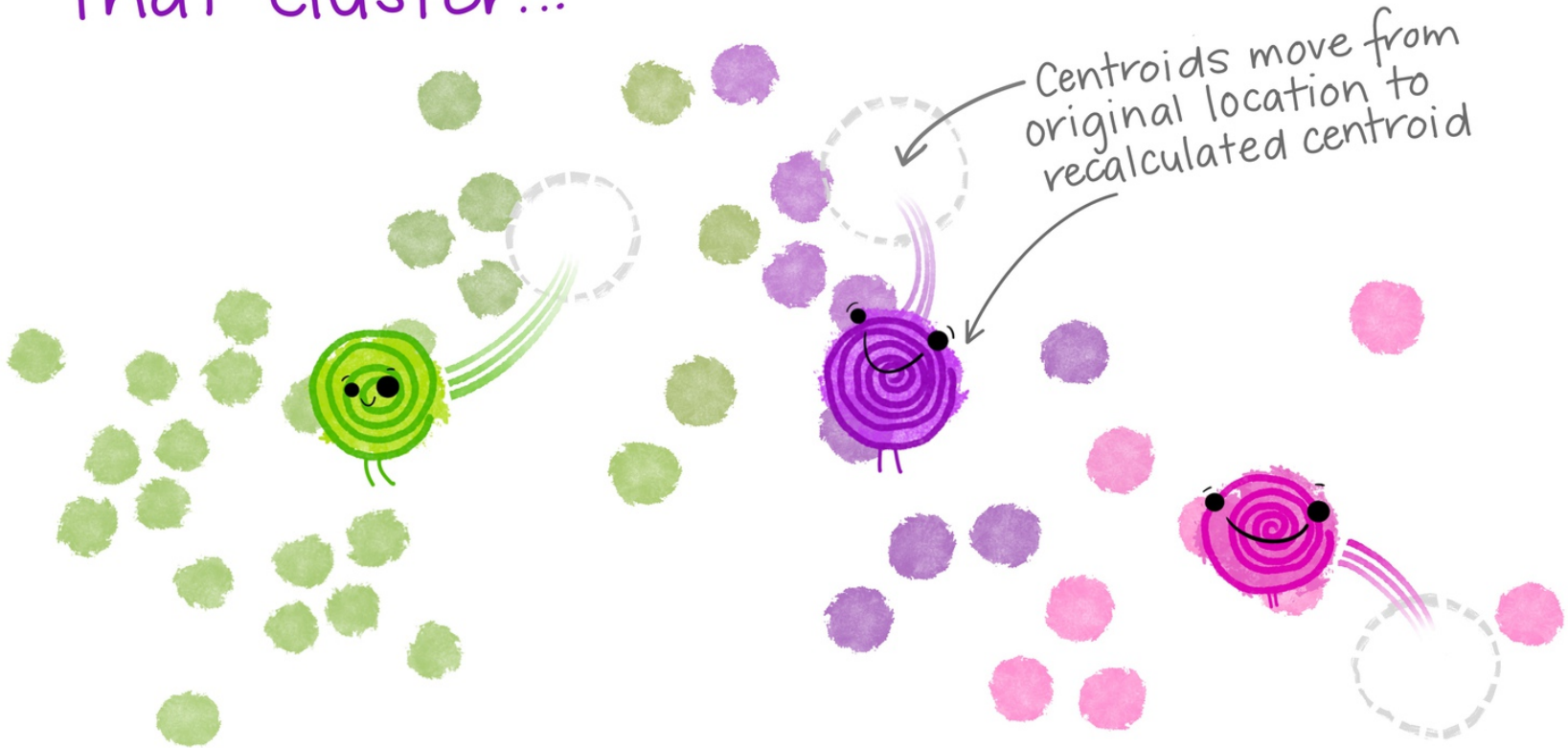
Each observation gets temporarily "assigned" to its closest centroid.

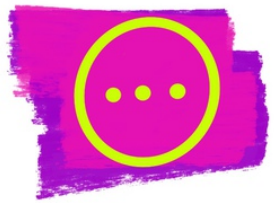
(e.g. by Euclidean distance)



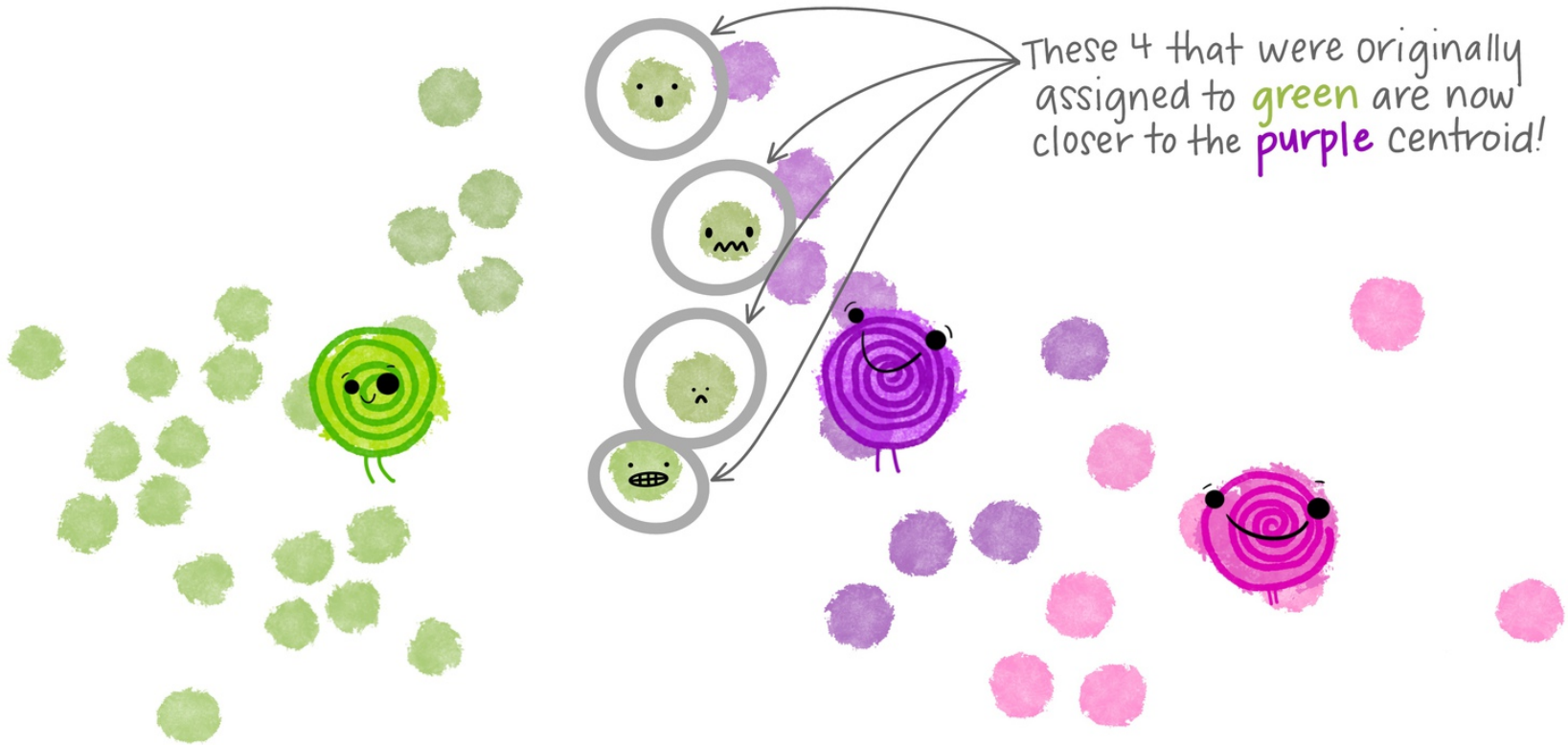
4

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



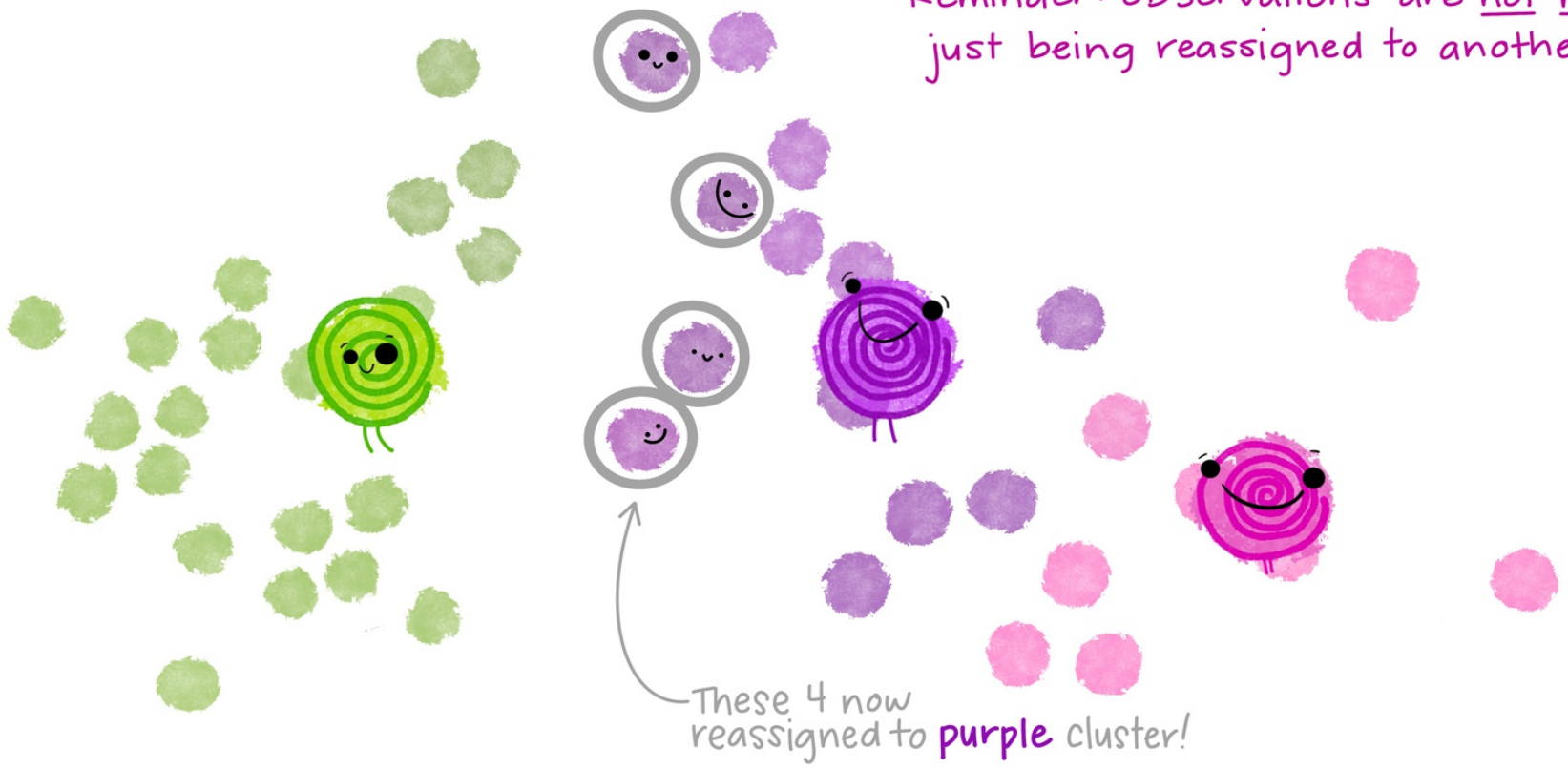


UH OH. Now that the cluster centroids have moved, some of the observations are now closer to a different centroid!



NO PROBLEM!
Observations get reassigned* to a different cluster based on the recalculated centroid.

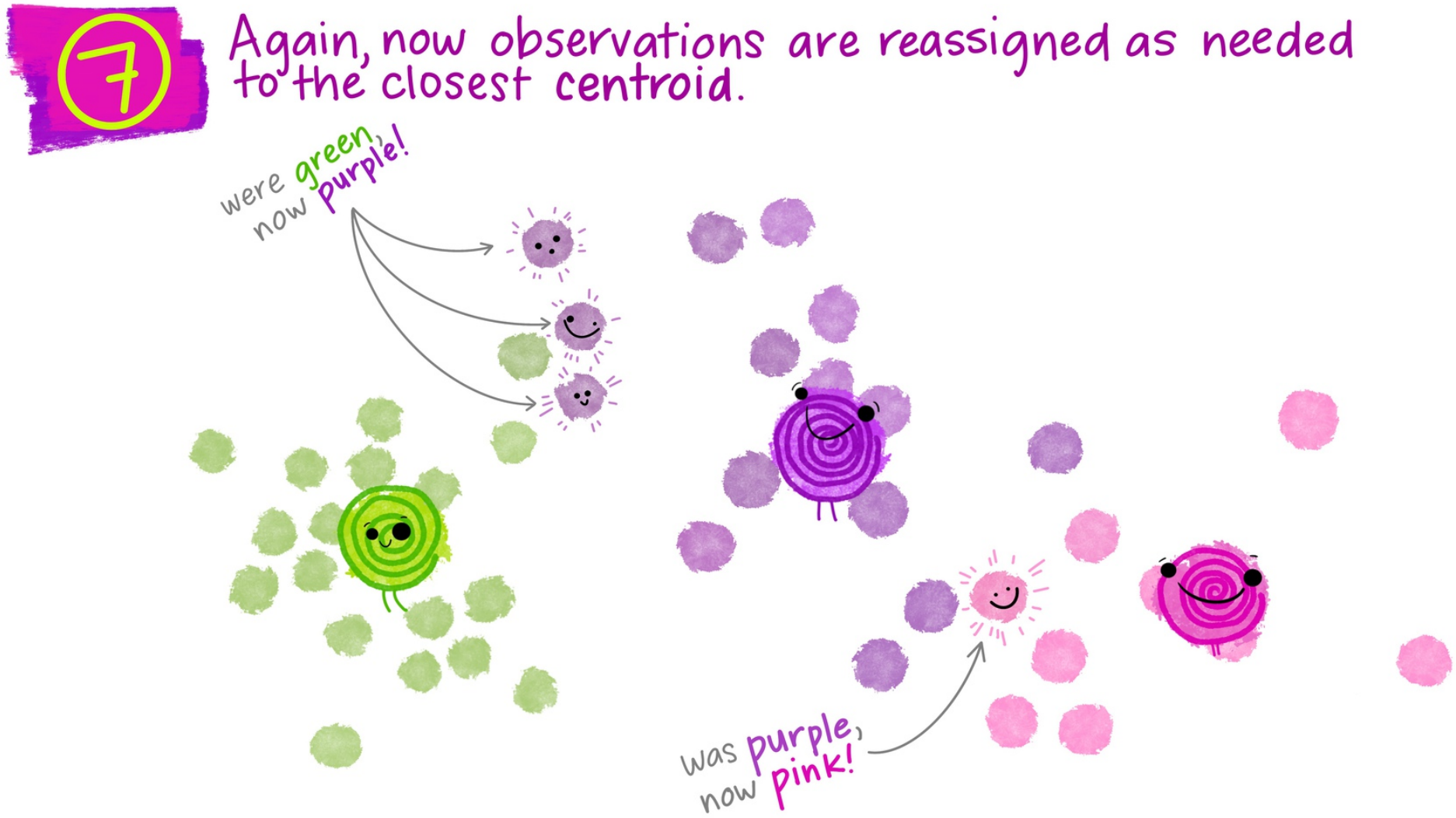
*Reminder: observations are not moving, just being reassigned to another cluster.

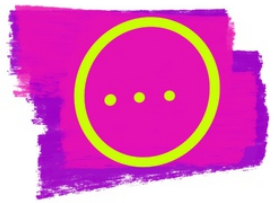


@allison_horst

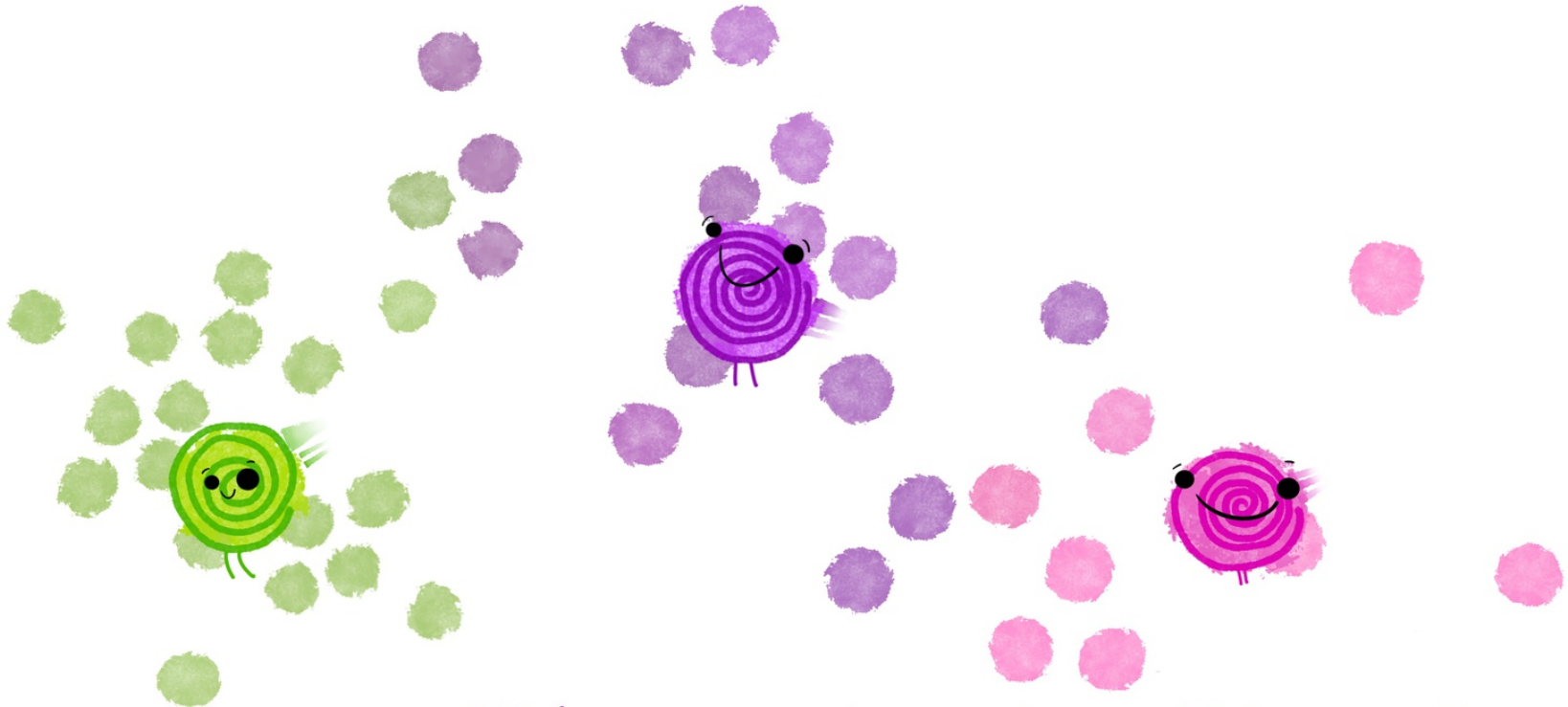
But now that observations have been reassigned, the centroids need to move again [recalculate centroids from updated clusters]

@allison_horst

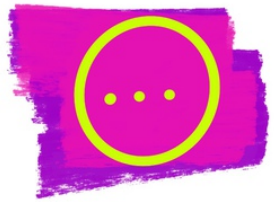

7
Again, now observations are reassigned as needed to the closest centroid.
were green, now purple!
was purple, now pink!
@allison_horst



Then the centroid for each cluster is recalculated...



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



That iterative process of

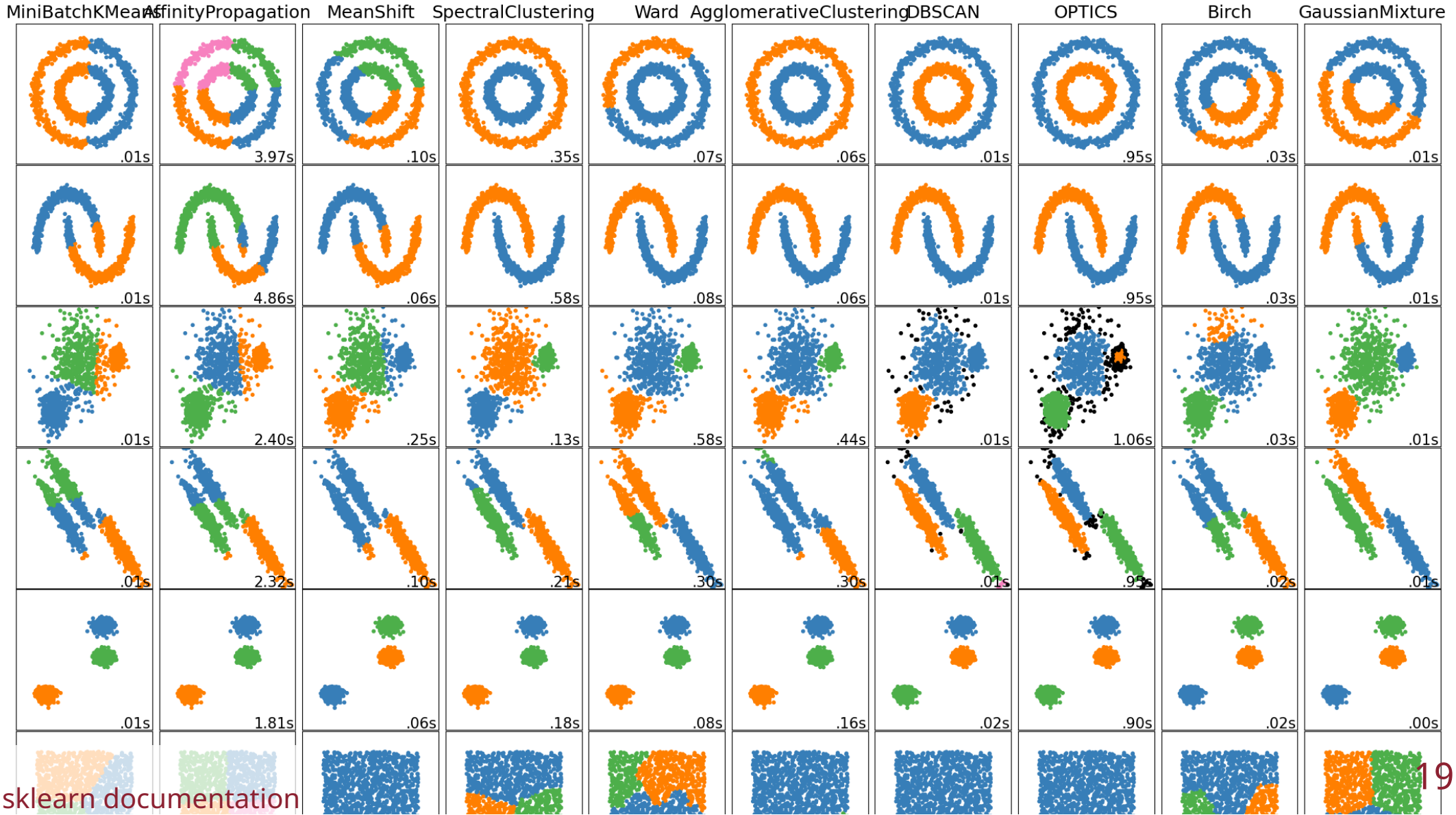
Recalculate cluster centroids
↳ Reassign observations to nearest centroid
↳ Recalculate cluster centroids
↳ Reassign observations to nearest centroid
↳ Recalculate cluster centroids
↳ Reassign observations to nearest centroid

↓

Continues until nothing is moving or being reassigned anymore!

Which means the iteration is done and each observation is assigned to its final cluster.

Many types of clustering algorithms



Source: sklearn documentation

```
set.seed(20201120)
clustering_results <-
  recipe( ~ Latitude + Longitude + Gr_Liv_Area, data = ames_all) %>%
  step_range(Gr_Liv_Area, min = 0, max = 1) %>%
  step_range(Latitude, min = 0, max = 1) %>%
  prep() %>%
  bake(new_data = ames_all) %>%
  kmeans(nstart = 4, centers = 3)

ames_with_clusters <- ames_all %>%
  mutate(.cluster = as.factor(clustering_results$cluster))
```

```
glance(clustering_results)
```

```
## # A tibble: 1 x 4
##   totss tot.withinss betweenss  iter
##   <dbl>        <dbl>     <dbl> <int>
## 1  180.         65.5      114.     4
```

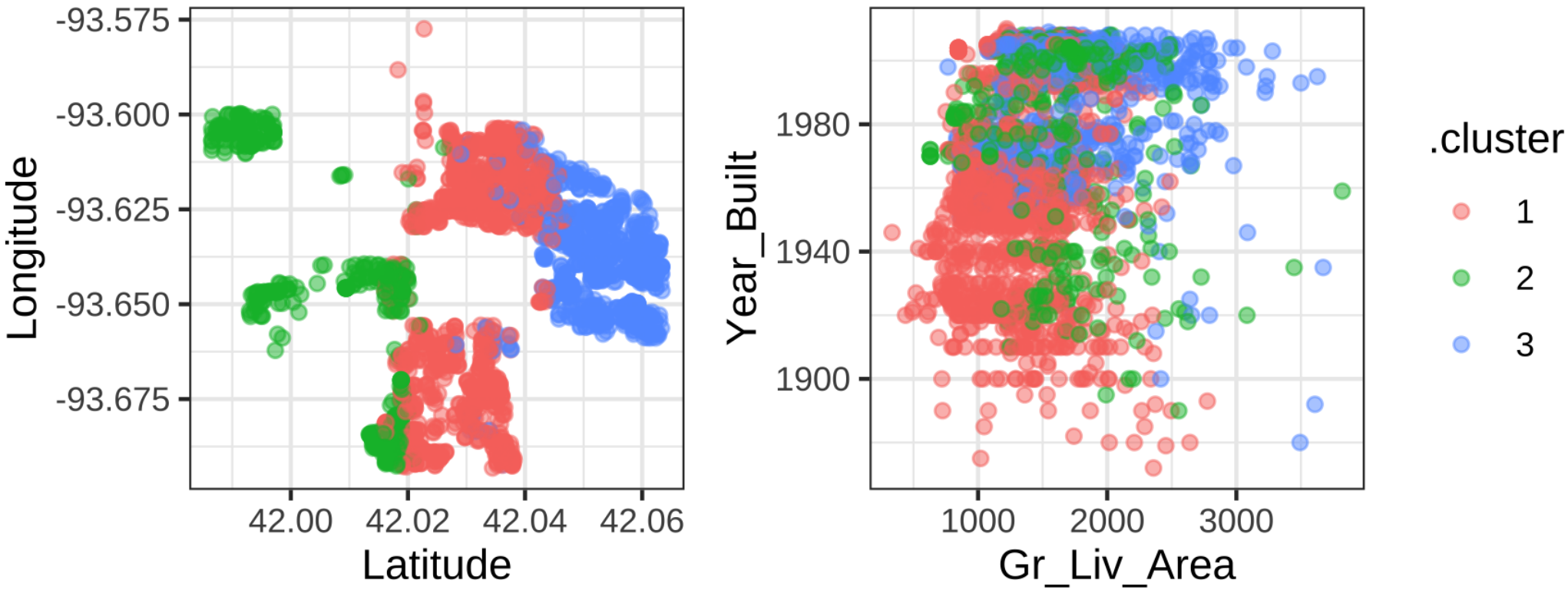
```
tidy(clustering_results)
```

```
## # A tibble: 3 x 6
##   Latitude Longitude Gr_Liv_Area  size withinss cluster
##      <dbl>     <dbl>       <dbl> <int>    <dbl> <fct>
## 1    0.579     -93.6       0.268  1161     23.8 1
## 2    0.264     -93.6       0.362   449     18.2 2
```

```
latlong_plot <-
  ggplot(ames_with_clusters, aes(x = Latitude, y = Longitude, color = .cluster)) +
    geom_point(alpha = .5)

year_area_plot <-
  ggplot(ames_with_clusters, aes(x = Gr_Liv_Area, y = Year_Built, color = .cluster)) +
    geom_point(alpha = .5)

latlong_plot + year_area_plot + plot_layout(guides='collect')
```

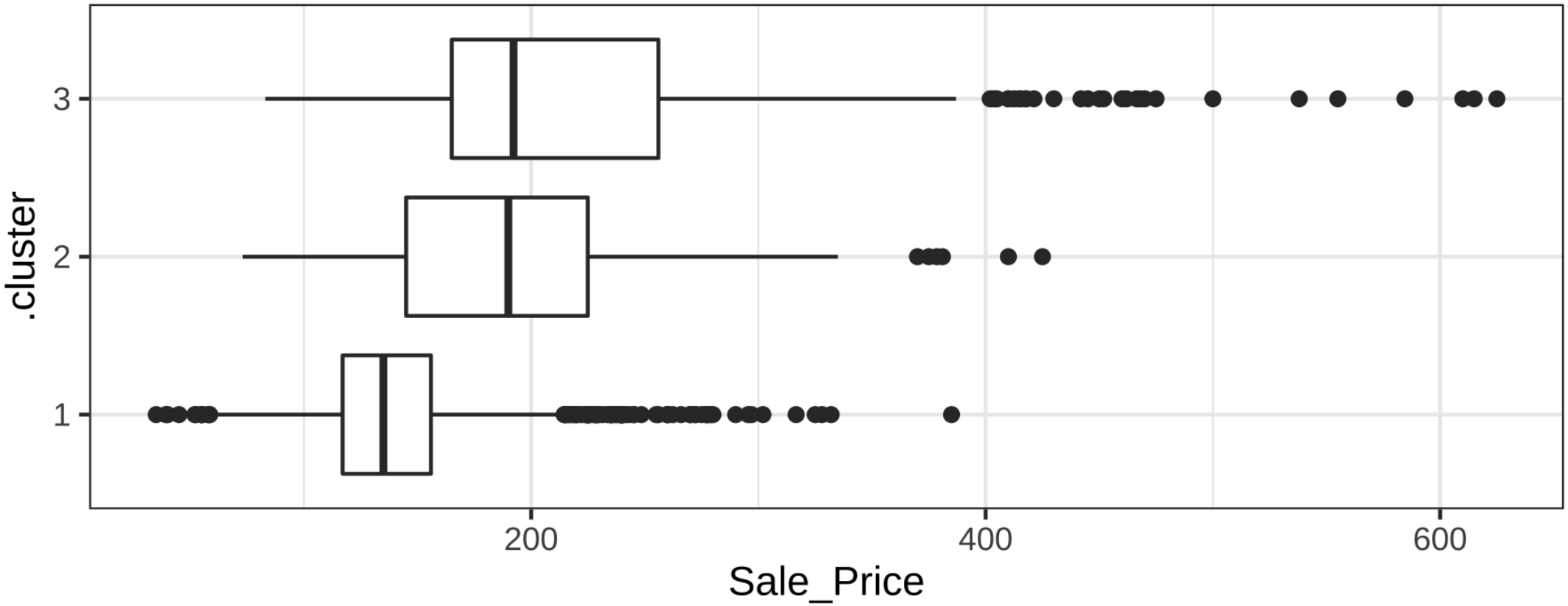


Activities

1. What differences do you notice between the plot on the left and the plot on the right?
2. Try increasing the number of `centers`. What changes about both plots?
3. Try changing the formula to `~ Year_Built` (removing latitude and longitude). What can you say about the age of homes in different parts of town?
4. Try adding `Gr_Liv_Area` to the recipe's formula (`Latitude + Longitude + Gr_Liv_Area`). What changes about both plots? Why are they different?
5. Try adding `step_range(Gr_Liv_Area, min = 0, max = 1)` to the recipe construction pipeline. What changes about both plots? Why?
6. Try adding a `step_range` for `Latitude` (but not `Longitude`). What changes and why?
7. Now add a `step_range` for `Longitude`. What changes and why?
8. Try changing `max` to `10` for `Gr_Liv_Area`. Then try `max = 0.1`. What changes and why?
9. Try adding `Year_Built`.

Do the patterns captured by these clusters also happen to relate to sale price?

```
ames_with_clusters %>%
  ggplot(aes(x = Sale_Price, y = .cluster)) + geom_boxplot()
```



Appendix

```
library(tidymodels)
library(patchwork)
```

```
#data(ames, package = "modeldata")
ames <- AmesHousing::make_ames()
ames_all <- ames %>%
  filter(Gr_Liv_Area < 4000, Sale_Condition == "Normal") %>%
  mutate(across(where(is.integer), as.double)) %>%
  mutate(Sale_Price = Sale_Price / 1000)
rm(ames)
```