DQ, McDonalds, Sonic Clusters

2023-04-20

Loading Libraries and Data

First, we load our libraries:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggrepel)
library(broom)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(purrr)
```

Now we need to import our data set, and create data sets containing just data on McDonalds, Sonic and DQ respecively:

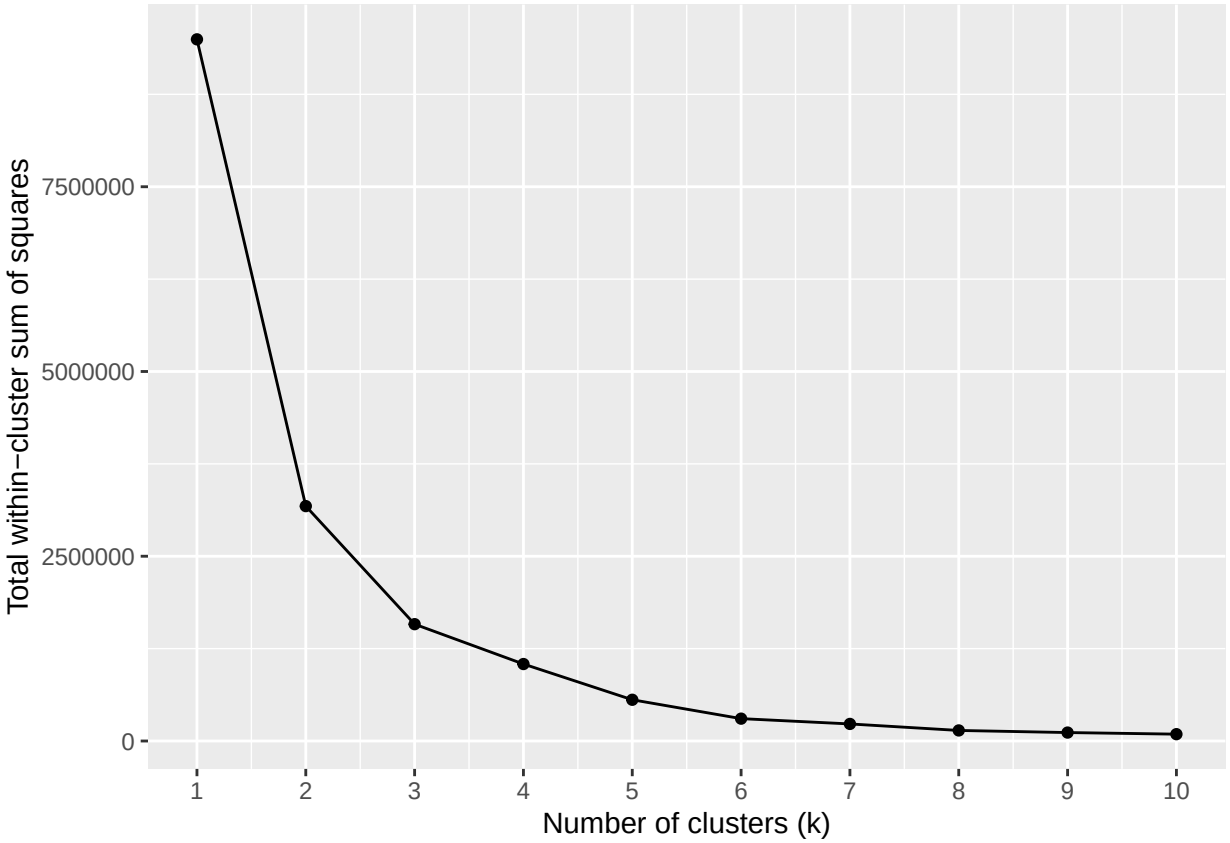
```
#Loading in the CSV
nutritional_data <-  read.csv("data/nutritioninfo.csv")

#McDonald's Data Set For 2-way clusters
mcd_nutrition <- nutritional_data %>%
  filter(restaurant == "Mcdonalds")%>%
  select(item, protein, calories)

#McDonald's Data Set for all variables
mcd_nutrition_data <- nutritional_data %>%
  filter(restaurant == "Mcdonalds")

#Dairy Queen Data Set for 2-way clusters
dq_nutrition <- nutritional_data %>%
```

```
  filter(restaurant == "Dairy Queen")%>%
  select(item, protein, calories)

#Dairy Queen Data Set for all variables
dq_nutrition_data <- nutritional_data %>%
  filter(restaurant == "Dairy Queen")

#Sonic Data set
sonic_nutrition <- nutritional_data %>%
   filter(restaurant == "Sonic")%>%
  select(item, protein, calories)

#Sonic Data Set for all variables
sonic_nutrition_data <- nutritional_data %>%
   filter(restaurant == "Sonic")
```

McDonald's K-Means Clustering

Now, We are going to make elbow plots to decide how many clusters we want for each McDonalds:

```
mcd_clusters_data <- mcd_nutrition %>%
  select(calories, protein)%>%
  drop_na

# Iterate through clustering algorithm for 10 different values of k
elbow_plot1 <- tibble(k = 1:10) %>%
  mutate(
    # List-column of 10 kmeans objects
    # (apply `kmeans()` to each value of `k`)
    kmeans_mcd = purrr::map(k, ~kmeans(mcd_clusters_data, .x, nstart = 20)),
    # List-column of "glanced" model summaries for each kmeans object
    # (apply `glance()` to each corresponding result after running `kmeans()`)
    glanced = purrr::map(kmeans_mcd, glance)) %>%
  # Turn `glanced` list-column into regular tibble columns
  unnest(cols = c(glanced))

# Construct elbow plot
ggplot(elbow_plot1, aes(x = k, y = tot.withinss)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = 1:10) +
  labs(x = "Number of clusters (k)",
       y = "Total within-cluster sum of squares")
```



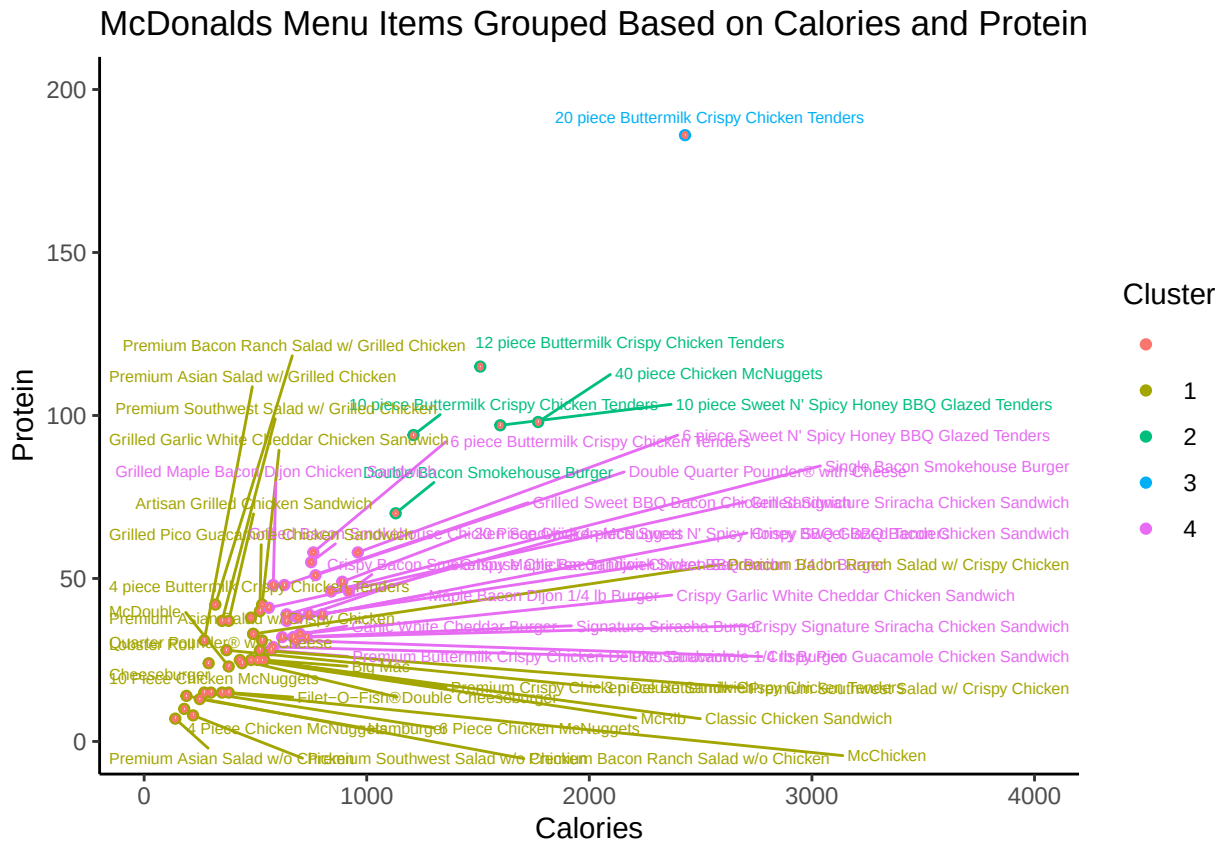
Based on the Elbow Plot Above, I am going to use four clusters, because there is still a large jump from three clusters to four, but then a very small jump between four clusters and five clusters.

Now we are going to cluster based on calories and protein, and create a graph and a list of which items are in which clusters

```
# set the seed for reproducibility
set.seed(23)

# Perform k-means clustering with k = 3
mcd_clusers_4 <- mcd_clusters_data %>%
  kmeans(centers = 4, nstart = 20)

mcd_clusers_c4 <- augment(mcd_clusers_4, mcd_nutrition)

ggplot(mcd_clusers_c4, aes(x = calories, y = protein)) +
  geom_point(aes(color = .cluster)) +
  geom_text_repel(aes(label = item, color = .cluster),
                  size = 2, max.overlaps = 200, show.legend = FALSE) +
  scale_x_continuous(breaks = scales::breaks_width(200)) +
  scale_y_continuous(breaks = scales::breaks_width(25)) +
  # Add centroid labels to plot
  geom_label(data = mcd_clusers_c4, aes(label = "", color = ""),
```

```
            size = 0.1,
            label.r = unit(0.05, "lines"),
            label.size = 0.5,
            label.padding = unit(0.05, "lines"),
            show.legend = FALSE) +
  labs(title = "McDonalds Menu Items Grouped Based on Calories and Protein",
       x = "Calories",
       y = "Protein",
       color = "Cluster") +
  theme_classic() +
  xlim(c(0,4000)) +
      ylim(c(0,200))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```




```
#Get Important Data Points and What items are in which cluster
mcd_summaries <- tidy(mcd_clusers_4)
mcd_summaries
```

```
## # A tibble: 4 x 5
##   calories protein  size withinss cluster
```

```
##      <dbl>   <dbl> <int>    <dbl> <fct>
## 1     384.    25.1    29  422469. 1
## 2    1444     94.8     5  289359. 2
## 3    2430    186       1       0  3
## 4     715.    41.3    22  277586. 4
```

```
mcd_clusers_4$centers
```

```
##    calories   protein
## 1  383.7931  25.10345
## 2 1444.0000  94.80000
## 3 2430.0000 186.00000
## 4  714.5455  41.31818
```

```
cluster_list <- split(mcd_nutrition$item, mcd_clusers_4$cluster)
cluster_list
```

```
## $`1`
##  [1] "Artisan Grilled Chicken Sandwich"
##  [2] "Big Mac"
##  [3] "Cheeseburger"
##  [4] "Classic Chicken Sandwich"
##  [5] "Double Cheeseburger"
##  [6] "Filet-O-Fish®"
##  [7] "Grilled Garlic White Cheddar Chicken Sandwich"
##  [8] "Hamburger"
##  [9] "Lobster Roll"
## [10] "McChicken"
## [11] "McDouble"
## [12] "McRib"
## [13] "Grilled Pico Guacamole Chicken Sandwich"
## [14] "Premium Crispy Chicken Deluxe Sandwich"
## [15] "Quarter Pounder® with Cheese"
## [16] "3 piece Buttermilk Crispy Chicken Tenders"
## [17] "4 piece Buttermilk Crispy Chicken Tenders"
## [18] "4 Piece Chicken McNuggets"
## [19] "6 Piece Chicken McNuggets"
## [20] "10 Piece Chicken McNuggets"
## [21] "Premium Asian Salad w/o Chicken"
## [22] "Premium Asian Salad w/ Grilled Chicken"
## [23] "Premium Asian Salad w/ Crispy Chicken"
## [24] "Premium Bacon Ranch Salad w/o Chicken"
## [25] "Premium Bacon Ranch Salad w/ Grilled Chicken"
## [26] "Premium Bacon Ranch Salad w/ Crispy Chicken"
## [27] "Premium Southwest Salad w/o Chicken"
## [28] "Premium Southwest Salad w/ Grilled Chicken"
## [29] "Premium Southwest Salad w/ Crispy Chicken"
##
## $`2`
## [1] "Double Bacon Smokehouse Burger"
## [2] "10 piece Buttermilk Crispy Chicken Tenders"
## [3] "12 piece Buttermilk Crispy Chicken Tenders"
## [4] "40 piece Chicken McNuggets"
```

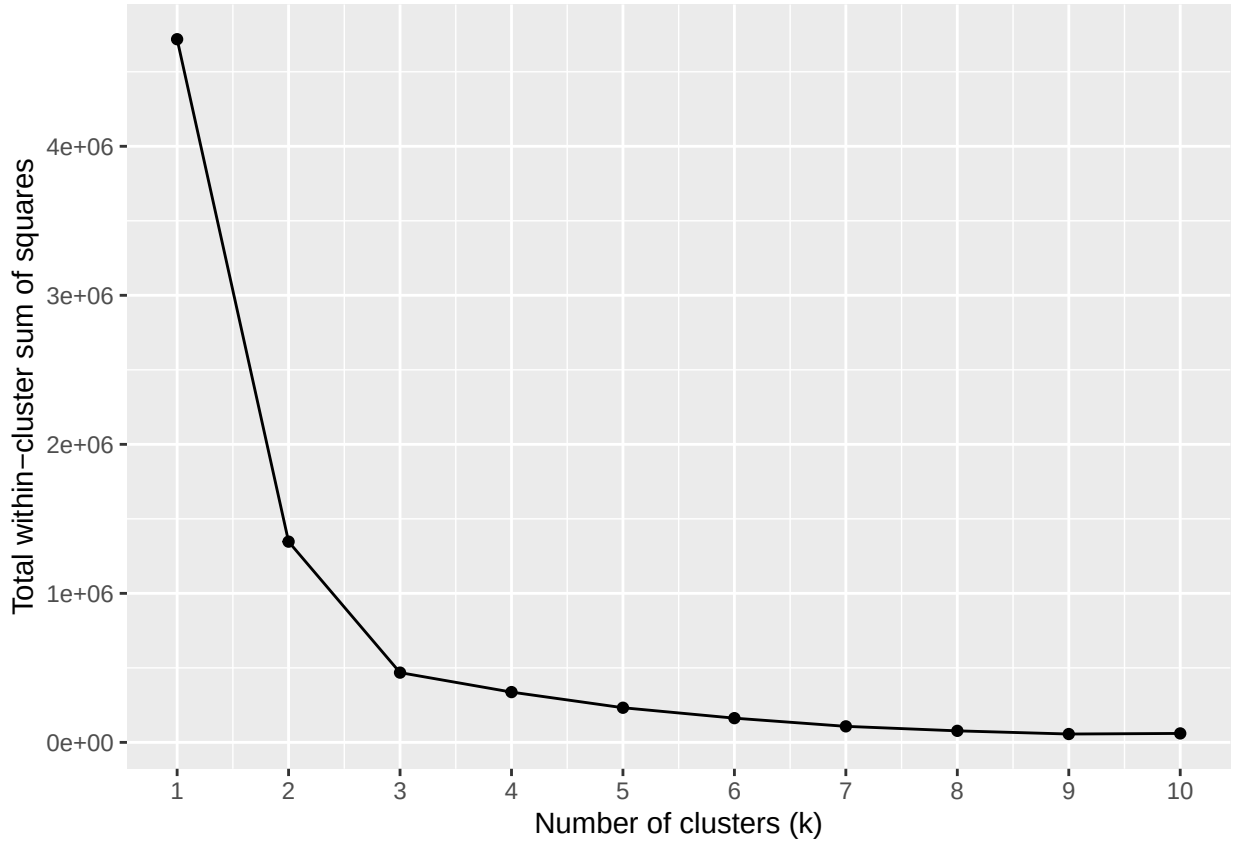
```
## [5] "10 piece Sweet N' Spicy Honey BBQ Glazed Tenders"
##
## $`3`
## [1] "20 piece Buttermilk Crispy Chicken Tenders"
##
## $`4`
##  [1] "Single Bacon Smokehouse Burger"
##  [2] "Grilled Bacon Smokehouse Chicken Sandwich"
##  [3] "Crispy Bacon Smokehouse Chicken Sandwich"
##  [4] "Double Quarter Pounder® with Cheese"
##  [5] "Garlic White Cheddar Burger"
##  [6] "Crispy Garlic White Cheddar Chicken Sandwich"
##  [7] "Maple Bacon Dijon 1/4 lb Burger"
##  [8] "Grilled Maple Bacon Dijon Chicken Sandwich"
##  [9] "Crispy Maple Bacon Dijon Chicken Sandwich"
## [10] "Pico Guacamole 1/4 lb Burger"
## [11] "Crispy Pico Guacamole Chicken Sandwich"
## [12] "Premium Buttermilk Crispy Chicken Deluxe Sandwich"
## [13] "Signature Sriracha Burger"
## [14] "Grilled Signature Sriracha Chicken Sandwich"
## [15] "Crispy Signature Sriracha Chicken Sandwich"
## [16] "Sweet BBQ Bacon 1/4 lb Burger"
## [17] "Grilled Sweet BBQ Bacon Chicken Sandwich"
## [18] "Crispy Sweet BBQ Bacon Chicken Sandwich"
## [19] "6 piece Buttermilk Crispy Chicken Tenders"
## [20] "20 Piece Chicken McNuggets"
## [21] "4 piece Sweet N' Spicy Honey BBQ Glazed Tenders"
## [22] "6 piece Sweet N' Spicy Honey BBQ Glazed Tenders"
```

Now we repeat the same process with sonic:

First an elbow plot to find the best number of clusters to use

```
sonic_clusters_data <- sonic_nutrition %>%
  select(calories, protein)%>%
  drop_na

# Iterate through clustering algorithm for 10 different values of k
elbow_plot2 <- tibble(k = 1:10) %>%
  mutate(
    # List-column of 10 kmeans objects
    # (apply `kmeans()` to each value of `k`)
    kmeans_sonic = purrr::map(k, ~kmeans(sonic_clusters_data, .x, nstart = 20)),
    # List-column of "glanced" model summaries for each kmeans object
    # (apply `glance()` to each corresponding result after running `kmeans()`)
    glanced = purrr::map(kmeans_sonic, glance)) %>%
  # Turn `glanced` list-column into regular tibble columns
  unnest(cols = c(glanced))

# Construct elbow plot
ggplot(elbow_plot2, aes(x = k, y = tot.withinss)) +
```

```
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = 1:10) +
  labs(x = "Number of clusters (k)",
       y = "Total within-cluster sum of squares")
```



Based on this elbow plot I am going to use three clusters because again there is a large drop in within-cluster sum of squares, and not a very large drop from 3 to four.

Now we are going to cluster based on calories and protein, and create a graph and a list of which items are in which clusters

```
# set the seed for reproducibility
set.seed(23)

# Perform k-means clustering with k = 3
sonic_clusers_3 <- sonic_clusters_data %>%
  kmeans(centers = 3, nstart = 20)

sonic_clusers_c3 <- augment(sonic_clusers_3, sonic_nutrition)

ggplot(sonic_clusers_c3, aes(x = calories, y = protein)) +
  geom_point(aes(color = .cluster)) +
```

```
  geom_text_repel(aes(label = item, color = .cluster),
                  size = 2, max.overlaps = 200, show.legend = FALSE) +
  scale_x_continuous(breaks = scales::breaks_width(200)) +
  scale_y_continuous(breaks = scales::breaks_width(25)) +
  # Add centroid labels to plot
  geom_label(data = sonic_clusers_c3, aes(label = "", color = ""),
             size = 0.1,
             label.r = unit(0.05, "lines"),
             label.size = 0.5,
             label.padding = unit(0.05, "lines"),
             show.legend = FALSE) +
  labs(title = "Sonic Menu Items Grouped Based on Calories and Protein",
       x = "Calories",
       y = "Protein",
       color = "Cluster") +
  theme_classic() +
  xlim(c(0,2500)) +
      ylim(c(0,90))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```r
#Get Important Data Points and What items are in which cluster
sonic_summaries <- tidy(sonic_clusers_3)
sonic_summaries
```

```
## # A tibble: 3 x 5
##   calories protein  size withinss cluster
##      <dbl>   <dbl> <int>    <dbl> <fct>
## 1     378.    18.7    24  189883. 1
## 2     682.    32.8    19  150001. 2
## 3    1146     47.6    10  128210. 3
```

```r
sonic_clusers_3$centers
```

```
##    calories  protein
## 1  377.5000 18.66667
## 2  682.1053 32.78947
## 3 1146.0000 47.60000
```

```r
cluster_list <- split(sonic_nutrition$item, sonic_clusers_3$cluster)
cluster_list
```

```
## $`1`
##  [1] "Jr. Burger"
##  [2] "Jr. Chili Cheeseburger"
##  [3] "Jr. Deluxe Burger"
##  [4] "Jr. Deluxe Cheeseburger"
##  [5] "Veggie Burger W/ Ketchup"
##  [6] "Veggie Burger With Mustard"
##  [7] "Veggie Burger W/ Mustard"
##  [8] "Grilled Chicken Sandwich"
##  [9] "Chicken Strip Sandwich"
## [10] "3 Piece Crispy Chicken Tender Dinner"
## [11] "5 Piece Crispy Chicken Tender Dinner"
## [12] "Small Jumbo Popcorn Chicken"
## [13] "Small Spicy Jumbo Popcorn Chicken"
## [14] "3 Piece Super Crunch Chicken Strips"
## [15] "4 Piece Super Crunch Chicken Strips"
## [16] "Ultimate Chicken Club"
## [17] "All Beef All-american Style Dog - 6\""
## [18] "All Beef Chicago Dog - 6\""
## [19] "All Beef Chili Cheese Coney - 6\""
## [20] "All Beef New York Dog - 6\""
## [21] "All Beef Regular Hot Dog - 6\""
## [22] "Cheesy Bacon Pretzel Dog - 6 In."
## [23] "Corn Dog"
## [24] "The Original Pretzel Dog"
##
## $`2`
##  [1] "Hatch Green Chile Cheeseburger"
##  [2] "Jalapeno Burger"
##  [3] "Jr. Double Cheeseburger"
##  [4] "Sonic Bacon Cheeseburger (w/mayo)"
```

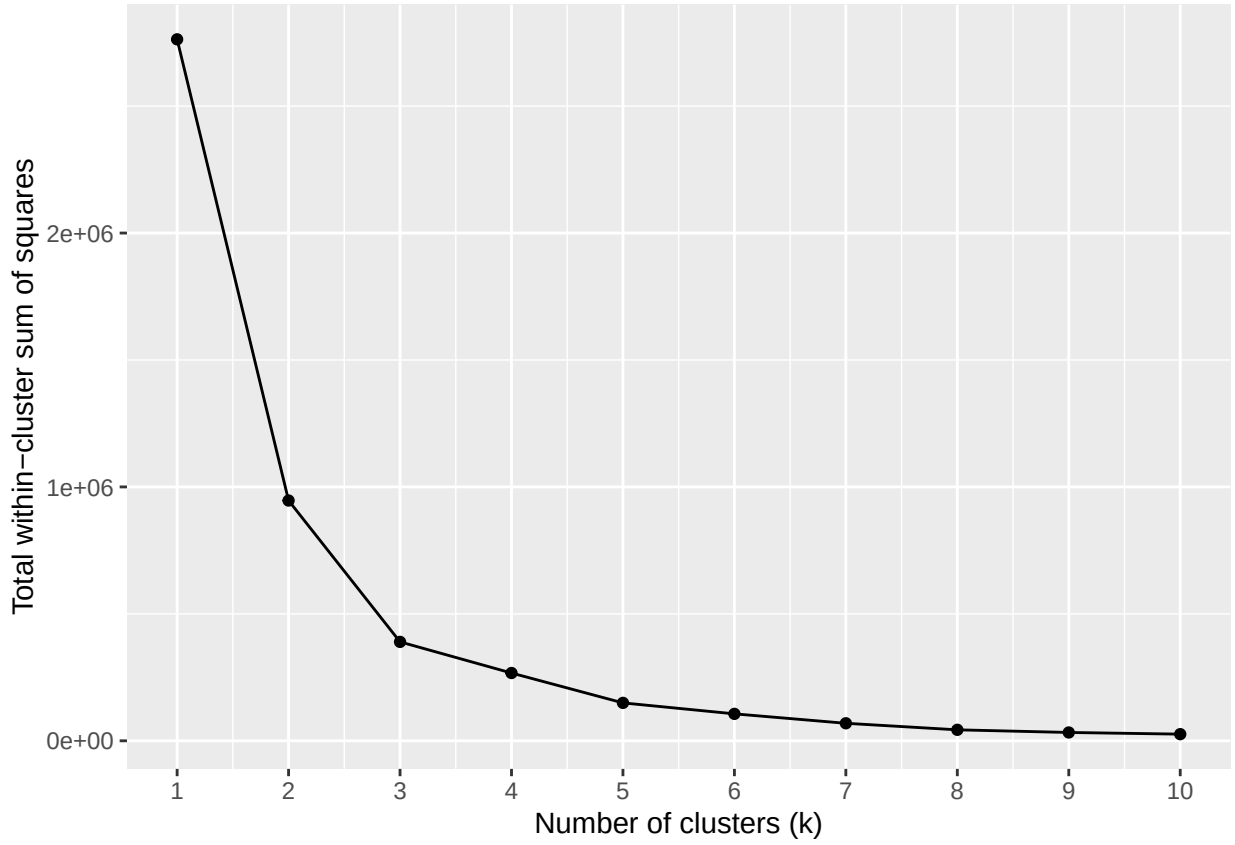
```
##  [5] "Sonic Burger W/ Mustard"
##  [6] "Sonic Burger W/ Ketchup"
##  [7] "Sonic Burger W/ Mayonnaise"
##  [8] "Sonic Cheeseburger W/ Mustard"
##  [9] "Sonic Cheeseburger W/ Ketchup"
## [10] "Sonic Cheeseburger W/ Mayonnaise"
## [11] "Grilled Asiago Caesar Chicken Club Sandwich"
## [12] "Crispy Asiago Caesar Chicken Club Sandwich"
## [13] "Crispy Chicken Sandwich"
## [14] "Deluxe Ultimate Chicken Sandwich"
## [15] "Large Jumbo Popcorn Chicken"
## [16] "Large Spicy Jumbo Popcorn Chicken"
## [17] "5 Piece Super Crunch Chicken Strips"
## [18] "Traditional Ultimate Chicken Sandwich"
## [19] "Footlong Quarter Pound Coney"
##
## $`3`
##  [1] "Super Sonic Bacon Double Cheeseburger (w/mayo)"
##  [2] "Super Sonic Double Cheeseburger W/ Mustard"
##  [3] "Super Sonic Double Cheeseburger W/ Ketchup"
##  [4] "Super Sonic Double Cheeseburger W/ Mayo"
##  [5] "Super Sonic Jalapeno Double Cheeseburger"
##  [6] "Buffalo Dunked Ultimate Chicken Sandwich"
##  [7] "Garlic Parmesan Dunked Ultimate Chicken Sandwich"
##  [8] "3 Piece Super Crunch Chicken Strip Dinner"
##  [9] "4 Piece Super Crunch Chicken Strip Dinner"
## [10] "5 Piece Super Crunch Chicken Strip Dinner"
```

Now we are going to do the same process but with Dairy Queen:

First an elbow plot to determin number of clusters:

```r
dq_clusters_data <- dq_nutrition %>%
  select(calories, protein)%>%
  drop_na

# Iterate through clustering algorithm for 10 different values of k
elbow_plot3 <- tibble(k = 1:10) %>%
  mutate(
    # List-column of 10 kmeans objects
    # (apply `kmeans()` to each value of `k`)
    kmeans_dq = purrr::map(k, ~kmeans(dq_clusters_data, .x, nstart = 20)),
    # List-column of "glanced" model summaries for each kmeans object
    # (apply `glance()` to each corresponding result after running `kmeans()`)
    glanced = purrr::map(kmeans_dq, glance)) %>%
  # Turn `glanced` list-column into regular tibble columns
  unnest(cols = c(glanced))

# Construct elbow plot
ggplot(elbow_plot3, aes(x = k, y = tot.withinss)) +
  geom_point() +
```

```
  geom_line() +
  scale_x_continuous(breaks = 1:10) +
  labs(x = "Number of clusters (k)",
       y = "Total within-cluster sum of squares")
```



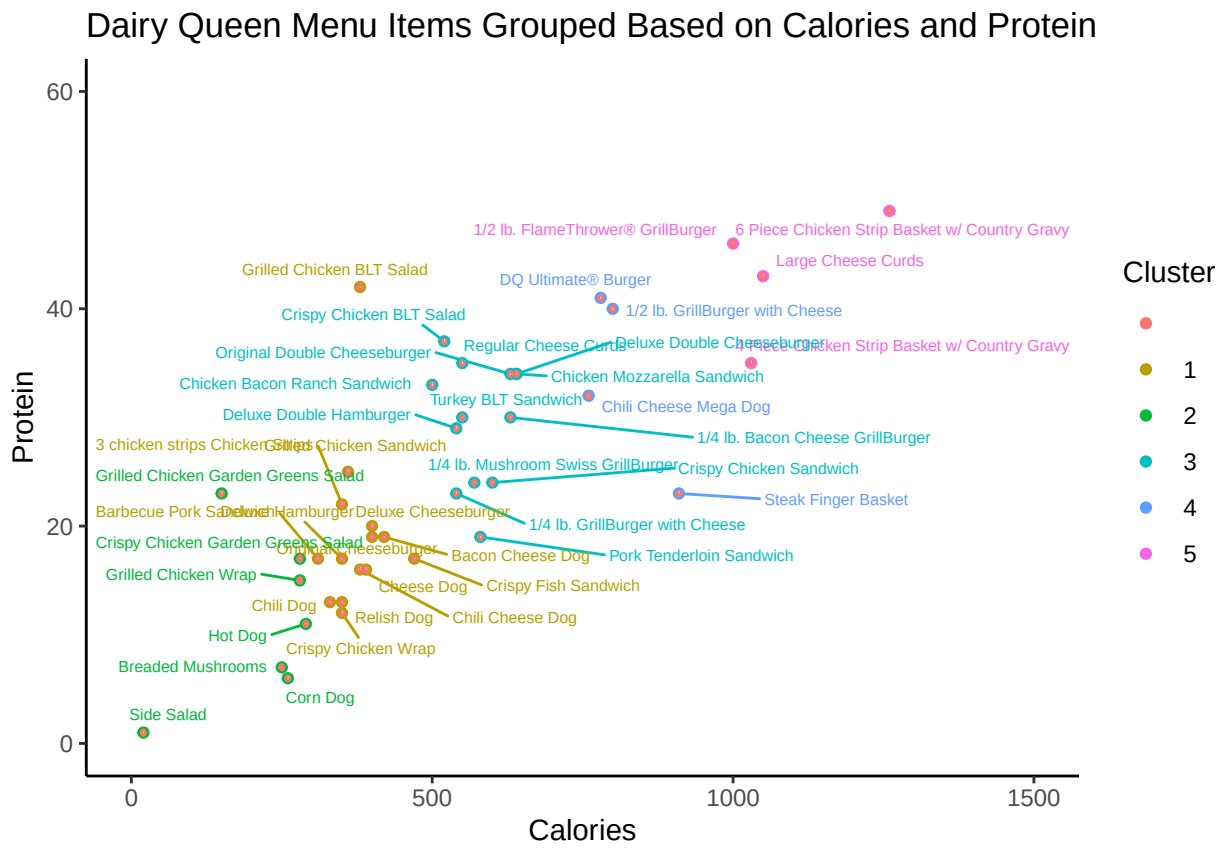
For Dairy Queen I am going to use five clusters because while there is not a large jump between 3 and 4 clusters there is a relatively large jump from 3 to 4.

Now we are going to cluster based on calories and protein, and create a graph and a list of which items are in which clusters

```
# set the seed for reproducibility
set.seed(23)

# Perform k-means clustering with k = 3
dq_clusers_5 <- dq_clusters_data %>%
  kmeans(centers = 5, nstart = 20)

dq_clusers_c5 <- augment(dq_clusers_5, dq_nutrition)

ggplot(dq_clusers_c5, aes(x = calories, y = protein)) +
  geom_point(aes(color = .cluster)) +
  geom_text_repel(aes(label = item, color = .cluster),
```

```
                  size = 2, max.overlaps = 200, show.legend = FALSE) +
  scale_x_continuous(breaks = scales::breaks_width(200)) +
  scale_y_continuous(breaks = scales::breaks_width(25)) +
  # Add centroid labels to plot
  geom_label(data = dq_clusers_c5, aes(label = "", color = ""),
             size = 0.1,
             label.r = unit(0.05, "lines"),
             label.size = 0.5,
             label.padding = unit(0.05, "lines"),
             show.legend = FALSE) +
  labs(title = "Dairy Queen Menu Items Grouped Based on Calories and Protein",
       x = "Calories",
       y = "Protein",
       color = "Cluster") +
  theme_classic() +
  xlim(c(0,1500)) +
      ylim(c(0,60))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```



```
#Get Important Data Points and What items are in which cluster
dq_summaries <- tidy(dq_clusers_5)
dq_summaries
```

```
## # A tibble: 5 x 5
##   calories protein  size withinss cluster
##      <dbl>   <dbl> <int>    <dbl> <fct>
## 1     374.    19.1    14   22269. 1
## 2     219.    11.4     7   59821. 2
## 3     576.    29.7    13   27880. 3
## 4     812.    34       4   13685  4
## 5    1085     43.2     4   42209. 5
```

```
dq_clusers_5$centers
```

```
##    calories  protein
## 1  374.2857 19.14286
## 2  218.5714 11.42857
## 3  576.1538 29.69231
## 4  812.5000 34.00000
## 5 1085.0000 43.25000
```

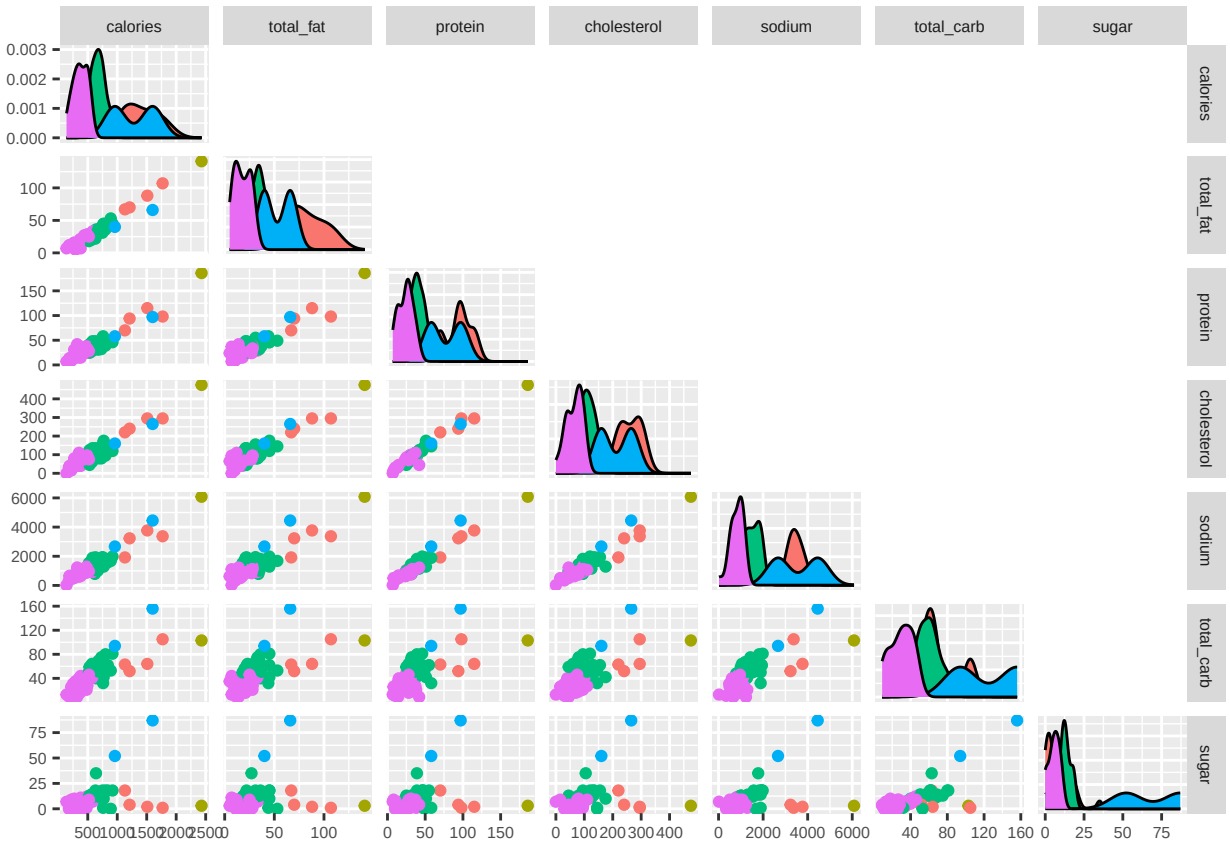
```
cluster_list <- split(dq_nutrition$item, dq_clusers_5$cluster)
cluster_list
```

```
## $`1`
##  [1] "Original Cheeseburger"           "Bacon Cheese Dog"
##  [3] "Cheese Dog"                      "Chili Cheese Dog"
##  [5] "Chili Dog"                       "Relish Dog"
##  [7] "Barbecue Pork Sandwich"          "Crispy Fish Sandwich"
##  [9] "Deluxe Cheeseburger"             "Deluxe Hamburger"
## [11] "3 chicken strips Chicken Strips" "Crispy Chicken Wrap"
## [13] "Grilled Chicken BLT Salad"       "Grilled Chicken Sandwich"
##
## $`2`
## [1] "Hot Dog"                             "Breaded Mushrooms"
## [3] "Corn Dog"                            "Crispy Chicken Garden Greens Salad"
## [5] "Grilled Chicken Garden Greens Salad" "Grilled Chicken Wrap"
## [7] "Side Salad"
##
## $`3`
##  [1] "1/4 lb. Bacon Cheese GrillBurger"   "1/4 lb. GrillBurger with Cheese"
##  [3] "1/4 lb. Mushroom Swiss GrillBurger" "Original Double Cheeseburger"
##  [5] "Regular Cheese Curds"               "Deluxe Double Cheeseburger"
##  [7] "Deluxe Double Hamburger"            "Pork Tenderloin Sandwich"
##  [9] "Chicken Bacon Ranch Sandwich"       "Chicken Mozzarella Sandwich"
## [11] "Crispy Chicken BLT Salad"           "Crispy Chicken Sandwich"
## [13] "Turkey BLT Sandwich"
##
## $`4`
## [1] "1/2 lb. GrillBurger with Cheese" "Chili Cheese Mega Dog"
## [3] "DQ Ultimate® Burger"             "Steak Finger Basket"
```

```
##
## $`5`
## [1] "1/2 lb. FlameThrower® GrillBurger"
## [2] "4 Piece Chicken Strip Basket w/ Country Gravy"
## [3] "6 Piece Chicken Strip Basket w/ Country Gravy"
## [4] "Large Cheese Curds"
```

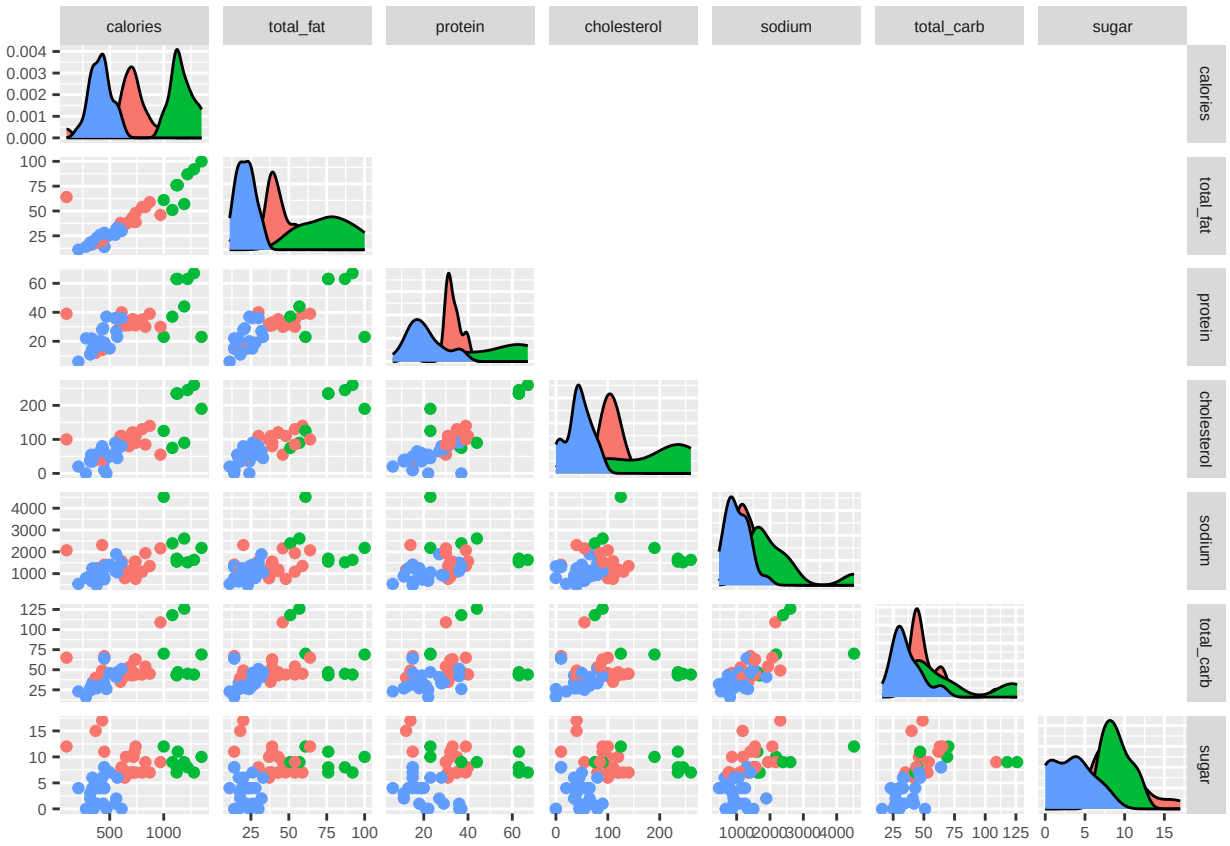
Now I am going to cluster based on all the categories, Just in case anyone is interested in that:

```
#This for mcdonalds (we used 4 clusters before, so we are going to use that again for consistency)
#getting rid of non-numeric variables
mcd_clusters_data_full <- mcd_nutrition_data %>%
  select(calories, total_fat, protein, cholesterol, sodium, total_carb, sugar)%>%
  drop_na

mcd_full_scaled <- mcd_clusters_data_full %>%
  # Standardize all numeric variables (subtract mean and divide by SD)
  mutate(across(where(is.numeric),  ~scale(.)[,1], .names = "{.col}_scaled")) %>%
  select(calories_scaled, total_fat_scaled, protein_scaled
         , cholesterol_scaled, sodium_scaled, total_carb_scaled, sugar_scaled) %>%
  drop_na()

#clustering using standarardized variables:
set.seed(23)

#clustering
mcd_kmean_full_scaled <- mcd_full_scaled %>%
  kmeans(centers = 5, nstart = 20)

#adding the items and restaurants back in
mcd_nutrition_full_scaled_clusters <- augment(mcd_kmean_full_scaled, mcd_nutrition_data) %>%
  rename(cluster_scaled = .cluster)

#creating the plot
ggpairs(mcd_nutrition_full_scaled_clusters, aes(color = cluster_scaled),
        columns = c("calories",
                    "total_fat",
                    "protein",
                    "cholesterol",
                    "sodium",
                    "total_carb",
                    "sugar"),
        upper = list(continuous = "blank")) +
  theme(text = element_text(size = 8))
```




```
#This for sonic (we used 3 clusters before, so we are going to use that again for consistency)
#getting rid of non-numeric variables
sonic_clusters_data_full <- sonic_nutrition_data %>%
  select(calories, total_fat, protein, cholesterol, sodium, total_carb, sugar)%>%
  drop_na

sonic_full_scaled <- sonic_clusters_data_full %>%
  # Standardize all numeric variables (subtract mean and divide by SD)
  mutate(across(where(is.numeric),  ~scale(.)[,1], .names = "{.col}_scaled")) %>%
  select(calories_scaled, total_fat_scaled, protein_scaled
         , cholesterol_scaled, sodium_scaled, total_carb_scaled, sugar_scaled) %>%
  drop_na()

#clustering using standarardized variables:
set.seed(23)

#clustering
sonic_kmean_full_scaled <- sonic_full_scaled %>%
  kmeans(centers = 3, nstart = 20)

#adding the items and restaurants back in
sonic_nutrition_full_scaled_clusters <- augment(sonic_kmean_full_scaled, sonic_nutrition_data) %>%
  rename(cluster_scaled = .cluster)

#creating the plot
ggpairs(sonic_nutrition_full_scaled_clusters, aes(color = cluster_scaled),
```

```
        columns = c("calories",
                    "total_fat",
                    "protein",
                    "cholesterol",
                    "sodium",
                    "total_carb",
                    "sugar"),
        upper = list(continuous = "blank")) +
  theme(text = element_text(size = 8))
```




```
#This is for dq (we used 5 clusters before, so we are going to use that again for consistency)
#getting rid of non-numeric variables
dq_clusters_data_full <- dq_nutrition_data %>%
  select(calories, total_fat, protein, cholesterol, sodium, total_carb, sugar)%>%
  drop_na

dq_full_scaled <- dq_clusters_data_full %>%
  # Standardize all numeric variables (subtract mean and divide by SD)
  mutate(across(where(is.numeric),  ~scale(.)[,1], .names = "{.col}_scaled")) %>%
  select(calories_scaled, total_fat_scaled, protein_scaled
         , cholesterol_scaled, sodium_scaled, total_carb_scaled, sugar_scaled) %>%
  drop_na()

#clustering using standarardized variables:
set.seed(23)
```

```
#clustering
dq_kmean_full_scaled <- dq_full_scaled %>%
  kmeans(centers = 5, nstart = 20)

#adding the items and restaurants back in
dq_nutrition_full_scaled_clusters <- augment(dq_kmean_full_scaled, dq_nutrition_data) %>%
  rename(cluster_scaled = .cluster)

#creating the plot
ggpairs(dq_nutrition_full_scaled_clusters, aes(color = cluster_scaled),
        columns = c("calories",
                    "total_fat",
                    "protein",
                    "cholesterol",
                    "sodium",
                    "total_carb",
                    "sugar"),
        upper = list(continuous = "blank")) +
  theme(text = element_text(size = 8))
```