Company Segmentation

Ramy Abougreeda

Contents

Project Summary

My organization wants to know which companies are similar to each other to help in identifying potential customers of a SAAS software solution (e.g. Salesforce CRM or equivalent) in various segments of the market. The Sales Department is very interested in this analysis, which will help them more easily penetrate various market segments.

I will be using stock prices in this analysis. I come up with a method to classify companies based on how their stocks trade using their daily stock returns (percentage movement from one day to the next). This analysis will help my organization determine which companies are related to each other (competitors and have similar attributes).

I will use a combination of `kmeans()` to find groups and `umap()` algorithms to visualize similarity of daily stock returns.

Objectives

Apply my knowledge on K-Means and UMAP along with `dplyr`, `ggplot2`, and `purrr` to create a visualization that identifies subgroups in the S&P 500 Index. I will specifically apply:

- Modeling: `kmeans()` and `umap()`
- Iteration: `purrr`
- Data Manipulation: `dplyr`, `tidyr`, and `tibble`
- Visualization: `ggplot2`

Libraries

```
library(tidyverse)
library(tidyquant)
library(broom)
library(umap)
```

Data

We will be using stock prices in this analysis. The `tidyquant` R package contains an API to retrieve stock prices. The following code is shown so you can see how I obtained the stock prices for every stock in the S&P 500 index. The files are saved in the `week_6_data` directory.

```
# GET ALL STOCKS IN A STOCK INDEX (E.G. SP500)
sp_500_index_tbl <- tq_index("SP500")
sp_500_index_tbl

# PULL IN STOCK PRICES FOR EACH STOCK IN THE INDEX
sp_500_prices_tbl <- sp_500_index_tbl %>%
    select(symbol) %>%
    tq_get(get = "stock.prices")


# SAVING THE DATA

sp_500_prices_tbl %>% write_rds(path = "week_6_data/week_6_data/new_sp_500_prices_tbl.rds")

sp_500_index_tbl %>% write_rds(path = 'week_6_data/week_6_data/new_sp_500_index_tbl.rds')
```

We can read in the stock prices. The data is 1.2M observations. The most important columns for our analysis are:

- `symbol`: The stock ticker symbol that corresponds to a company's stock price
- `date`: The timestamp relating the symbol to the share price at that point in time
- `adjusted`: The stock price, adjusted for any splits and dividends (we use this when analyzing stock data over long periods of time)

```
# STOCK PRICES
new_sp_500_prices_tbl <- read_rds('week_6_data/week_6_data/new_sp_500_prices_tbl.rds')

new_sp_500_prices_tbl %>% head()
```

```
## # A tibble: 6 x 8
##   symbol date        open  high   low close    volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1 AAPL   2015-01-02  27.8  27.9  26.8  27.3 212818400     24.3
## 2 AAPL   2015-01-05  27.1  27.2  26.4  26.6 257142000     23.7
## 3 AAPL   2015-01-06  26.6  26.9  26.2  26.6 263188400     23.7
## 4 AAPL   2015-01-07  26.8  27.0  26.7  26.9 160423600     24.0
## 5 AAPL   2015-01-08  27.3  28.0  27.2  28.0 237458000     24.9
## 6 AAPL   2015-01-09  28.2  28.3  27.6  28.0 214798000     24.9
```

The second data frame contains information about the stocks the most important of which are:

- `company`: The company name
- `sector`: The sector that the company belongs to

```
# SECTOR INFORMATION
new_sp_500_index_tbl <- read_rds("week_6_data/week_6_data/new_sp_500_index_tbl.rds")
new_sp_500_index_tbl %>% head()
```

```
## # A tibble: 6 x 8
##   symbol company        identifier sedol weight sector shares_held local_currency
##   <chr>  <chr>          <chr>      <chr>  <dbl> <chr>        <dbl> <chr>
## 1 AAPL   APPLE INC      037833100  2046~ 0.0723 -        188328306 USD
## 2 NVDA   NVIDIA CORP    67066G104  2379~ 0.0673 -        305619525 USD
## 3 MSFT   MICROSOFT CO~  594918104  2588~ 0.0629 -         92631196 USD
## 4 AMZN   AMAZON.COM I~  023135106  2000~ 0.0414 -        116595738 USD
## 5 META   META PLATFOR~  30303M102  B7TL~ 0.0271 -         27160538 USD
## 6 GOOGL  ALPHABET INC~  02079K305  BYVY~ 0.0227 -         72798116 USD
```

Question

Which stock prices behave similarly?

Answering this question helps us **understand which companies are related**, and we can use clustering to help us answer it!

Let's get started.

Step 1 - Converting stock prices to a standardized format (daily returns)

We know that in order to compare the data, it needs to be standardized or normalized. Why? Because we cannot compare values (stock prices) that are of completely different magnitudes. In order to standardize, we will convert from adjusted stock price (dollar value) to daily returns (percent change from previous day). Here is the formula.

$$return_{daily} = \frac{price_i - price_{i-1}}{price_{i-1}}$$

First, what do we have? We have stock prices for every stock in the SP 500 Index, which is the daily stock prices for over 500 stocks. The data set is over 1.2M observations.

```
new_sp_500_prices_tbl %>% glimpse()
```

```
## Rows: 1,235,532
## Columns: 8
## $ symbol   <chr> "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL~
## $ date     <date> 2015-01-02, 2015-01-05, 2015-01-06, 2015-01-07, 2015-01-08, ~
## $ open     <dbl> 27.8475, 27.0725, 26.6350, 26.8000, 27.3075, 28.1675, 28.1500~
## $ high     <dbl> 27.8600, 27.1625, 26.8575, 27.0500, 28.0375, 28.3125, 28.1575~
## $ low      <dbl> 26.8375, 26.3525, 26.1575, 26.6750, 27.1750, 27.5525, 27.2000~
## $ close    <dbl> 27.3325, 26.5625, 26.5650, 26.9375, 27.9725, 28.0025, 27.3125~
## $ volume   <dbl> 212818400, 257142000, 263188400, 160423600, 237458000, 214798~
## $ adjusted <dbl> 24.34717, 23.66127, 23.66350, 23.99532, 24.91727, 24.94399, 2~
```

My first task is to convert to a tibble named `new_sp_500_daily_returns_tbl` by performing some data transformations:

```
# Applying data transformations

new_sp_500_daily_returns_tbl <- new_sp_500_prices_tbl %>%

    select(symbol, date, adjusted) %>%

    filter(date >= ymd("2018-01-01")) %>%

    group_by(symbol) %>%
    mutate(lag_1 = lag(adjusted)) %>%
    ungroup() %>%

    filter(!is.na(lag_1)) %>%

    mutate(diff = adjusted - lag_1) %>%
    mutate(pct_return = diff / lag_1) %>%

    select(symbol, date, pct_return)

new_sp_500_daily_returns_tbl %>% head()
```

```
## # A tibble: 6 x 3
##   symbol date       pct_return
##   <chr>  <date>          <dbl>
## 1 AAPL   2018-01-03  -0.000174
## 2 AAPL   2018-01-04   0.00464
## 3 AAPL   2018-01-05   0.0114
## 4 AAPL   2018-01-08  -0.00371
## 5 AAPL   2018-01-09  -0.000115
## 6 AAPL   2018-01-10  -0.000229
```

Step 2 - Converting it to User-Item Format

The next step is to convert to a user-item format with the `symbol` in the first column and every other column the value of the *daily returns* (`pct_return`) for every stock at each `date`.

Now that we have the daily returns (percentage change from one day to the next), we can convert to a user-item format. The user in this case is the `symbol` (company), and the item in this case is the `pct_return` at each `date`.

Spreading the `date` column to get the values as percentage returns. Make sure to fill an `NA` values with zeros.

```
# Convert to User-Item Format

stock_date_matrix_tbl <- new_sp_500_daily_returns_tbl %>%
    spread(date, pct_return, fill = 0)

stock_date_matrix_tbl %>% head()
```

```
## # A tibble: 6 x 1,768
##   symbol `2018-01-03` `2018-01-04` `2018-01-05` `2018-01-08` `2018-01-09`
##   <chr>         <dbl>        <dbl>        <dbl>        <dbl>        <dbl>
## 1 A          0.0254      -0.00750      0.0160      0.00215      0.0246
## 2 AAPL      -0.000174     0.00464      0.0114     -0.00371     -0.000115
## 3 ABBV       0.0156      -0.00570      0.0174     -0.0160       0.00754
## 4 ABNB       0            0            0           0            0
## 5 ABT        0.00221     -0.00170      0.00289    -0.00288      0.00170
## 6 ACGL       0.000906     0.00373     -0.00395     0.000113    -0.0129
## # i 1,762 more variables: `2018-01-10` <dbl>, `2018-01-11` <dbl>,
## #   `2018-01-12` <dbl>, `2018-01-16` <dbl>, `2018-01-17` <dbl>,
## #   `2018-01-18` <dbl>, `2018-01-19` <dbl>, `2018-01-22` <dbl>,
## #   `2018-01-23` <dbl>, `2018-01-24` <dbl>, `2018-01-25` <dbl>,
## #   `2018-01-26` <dbl>, `2018-01-29` <dbl>, `2018-01-30` <dbl>,
## #   `2018-01-31` <dbl>, `2018-02-01` <dbl>, `2018-02-02` <dbl>,
## #   `2018-02-05` <dbl>, `2018-02-06` <dbl>, `2018-02-07` <dbl>, ...
```

Step 3 - Perform K-Means Clustering

Next, we'll perform **K-Means clustering**.

```
# Create kmeans_obj for 4 centers

kmeans_obj <- stock_date_matrix_tbl %>%

    select(-symbol) %>%

    kmeans(centers = 4, nstart = 20)
```

Applying glance() to get the `tot.withinss`. `tot.withinss` is the sum of the squared Euclidean distances between each data point and its corresponding cluster centroid.

```
# Applying glance() to get the tot.withinss for centers = 4
broom::glance(kmeans_obj)$tot.withinss
```

```
## [1] 237.0144
```

Step 4 - Find the optimal value of K

We'll use this **custom function** called `kmeans_mapper()` to iterate over many values of "k" using the `centers` argument:

```
kmeans_mapper <- function(center = 3) {
    stock_date_matrix_tbl %>%
        select(-symbol) %>%
        kmeans(centers = center, nstart = 20)
    }
```

Apply the `kmeans_mapper()` and `glance()` functions iteratively using `purrr`.

```
# Use purrr package to map
k_means_mapped_tbl <- tibble(centers = 1:15) %>%
    mutate(k_means = centers %>% map(kmeans_mapper)) %>%
    mutate(glance = k_means %>% map(glance))
```

Next, let's visualize the "tot.withinss" from the glance output as a ***Scree Plot***.

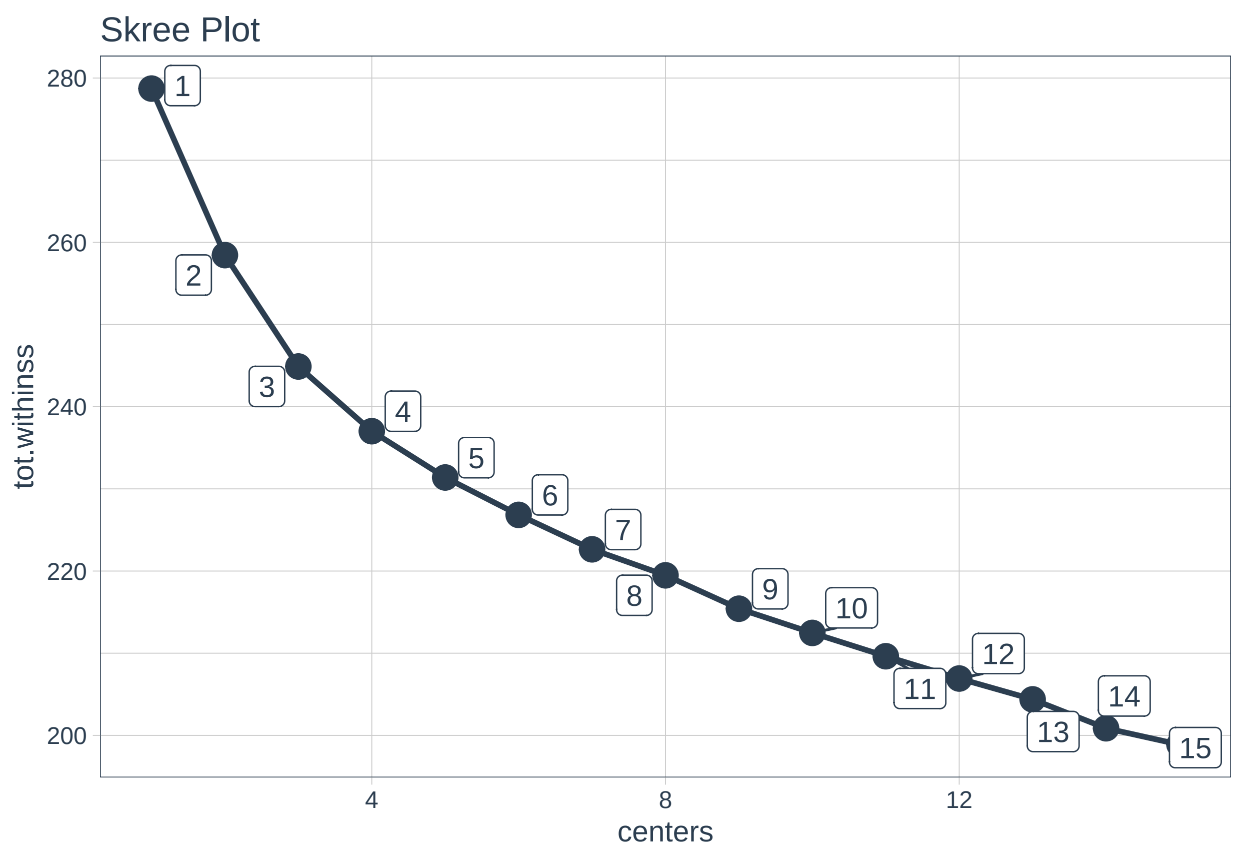
```
# Visualize Scree Plot

k_means_mapped_tbl %>% unnest(glance) %>%
    ggplot(aes(x = centers, y = tot.withinss)) +

    # Geometries
     geom_point(colour = '#2c3e50', size = 4) +
    geom_line(colour = '#2c3e50', linewidth = 1) +

    # Label
    ggrepel::geom_label_repel(aes(label = centers), color = '#2c3e50') +

    # Formatting
    theme_tq() +
    labs(title = "Skree Plot")
```



We can see that the Scree Plot becomes linear (constant rate of change) between 5 and 10 centers for K.

Step 5 - Apply UMAP

Next, let's plot the `UMAP` 2D visualization to help us investigate cluster assignments.

```
# Apply UMAP

umap_results <- stock_date_matrix_tbl %>%
    select(-symbol) %>%
    umap()
umap_results$layout %>% head()
```

```
##            [,1]       [,2]
## [1,] -2.1978804  0.4806881
## [2,] -2.1905463 -1.9822083
## [3,] -0.6000539  0.9732576
## [4,] -1.5066381  2.0622846
## [5,] -1.6137799  0.6040669
## [6,]  1.0724077 -0.1380852
```

Next, we want to combine the `layout` from the `umap_results` with the `symbol` column from the `stock_date_matrix_tbl`.

```r
# Convert umap results to tibble with symbols

umap_results_tbl <- as_tibble(umap_results$layout) %>%
    set_names(c('X', 'Y')) %>%
    bind_cols(stock_date_matrix_tbl %>% select(symbol)) %>%
    select(symbol, X, Y)
umap_results_tbl %>% head()
```

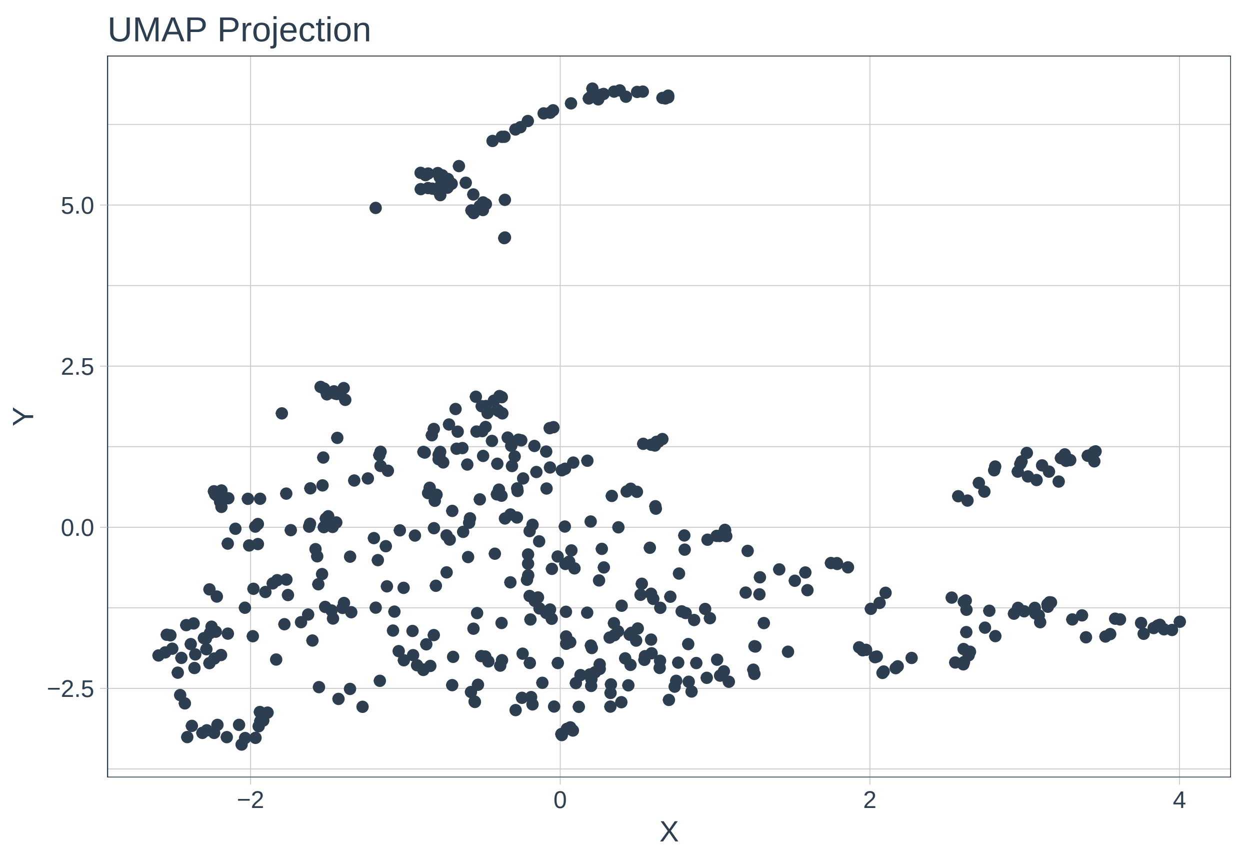
```
## # A tibble: 6 x 3
##   symbol      X      Y
##   <chr>   <dbl>  <dbl>
## 1 A      -2.20   0.481
## 2 AAPL   -2.19  -1.98
## 3 ABBV   -0.600  0.973
## 4 ABNB   -1.51   2.06
## 5 ABT    -1.61   0.604
## 6 ACGL    1.07  -0.138
```

Finally, let's make a quick visualization of the `umap_results_tbl`.

```r
# Visualize UMAP results
umap_results_tbl %>%

    ggplot(aes(X, Y)) +
    geom_point(color = "#2c3e50") +

    theme_tq() +
    labs(title = "UMAP Projection")
```



We can now see that we have some clusters. However, we still need to combine the K-Means clusters and the UMAP 2D representation.

Step 6 - Combine K-Means and UMAP

Next, we combine the K-Means clusters and the UMAP 2D representation

First, pull out the K-Means for 4 Centers. Use this since beyond this value the Scree Plot flattens.

```
# Getting the k_means_obj from the 4th center
k_means_obj <- k_means_mapped_tbl %>%
    filter(centers == 4) %>%
    pull(k_means) %>%
    pluck(1)
```

Next, I'll combine the clusters from the `k_means_obj` with the `umap_results_tbl`.

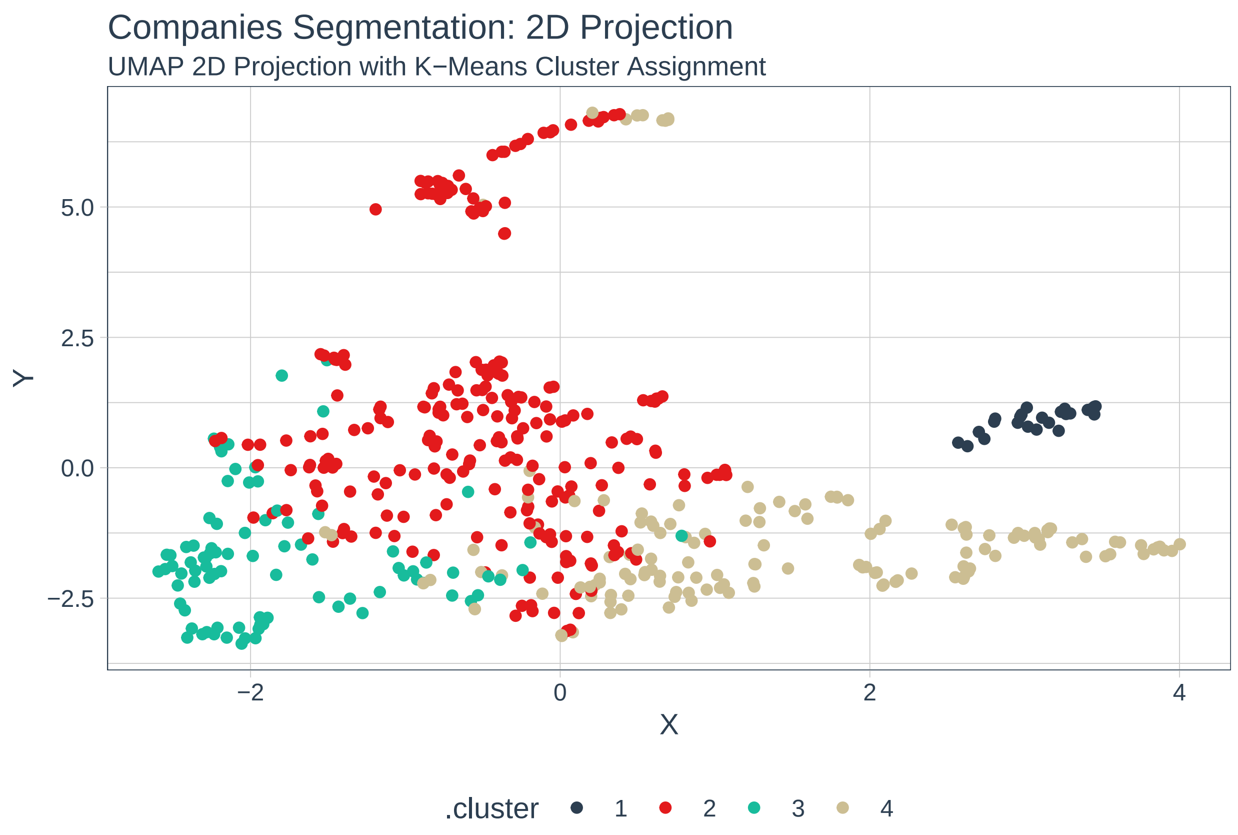
```
umap_kmeans_results_tbl <- k_means_obj %>%
    augment(stock_date_matrix_tbl) %>%
    select(symbol, .cluster) %>%
    left_join(umap_results_tbl, by = "symbol") %>%
    left_join(new_sp_500_index_tbl %>% select(symbol, company, sector),
              by = "symbol")
umap_kmeans_results_tbl %>% head()
```

```
## # A tibble: 6 x 6
##   symbol .cluster      X      Y company                  sector
##   <chr>  <fct>     <dbl>  <dbl> <chr>                    <chr>
## 1 A      2        -2.20   0.481 AGILENT TECHNOLOGIES INC -
## 2 AAPL   3        -2.19  -1.98  APPLE INC                -
## 3 ABBV   2        -0.600  0.973 ABBVIE INC               -
## 4 ABNB   3        -1.51   2.06  AIRBNB INC CLASS A       -
## 5 ABT    2        -1.61   0.604 ABBOTT LABORATORIES      -
## 6 ACGL   2         1.07  -0.138 ARCH CAPITAL GROUP LTD   -
```

Plot the K-Means and UMAP results.

```
# Visualize the combined K-Means and UMAP results
umap_kmeans_results_tbl %>%
    ggplot(aes(X, Y, colour = .cluster)) +
    geom_point() +

    theme_tq() +
    scale_color_tq() +
    labs(
        title = "Companies Segmentation: 2D Projection",
        subtitle = "UMAP 2D Projection with K-Means Cluster Assignment"
    )
```


Companies Segmentation: 2D Projection
UMAP 2D Projection with K−Means Cluster Assignment
Y
X
.cluster