101_wk_6_modeling_part1

Seung Hyun Sung

11/18/2021

DS4B 101-R: R FOR BUSINESS ANALYSIS

CUSTOMER SEGMENTATION

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

bike_orderlines_tbl <- read_rds("~/Desktop/University_business_science/DS4B_101/00_data/bike_sales/data_

glimpse(bike_orderlines_tbl)
```

```
## Rows: 15,644
## Columns: 13
## $ order_date     <dttm> 2011-01-07, 2011-01-07, 2011-01-10, 2011-01-10, 2011-0~
## $ order_id       <dbl> 1, 1, 2, 2, 3, 3, 3, 3, 3, 4, 5, 5, 5, 5, 6, 6, 6, 6, 7~
## $ order_line     <dbl> 1, 2, 1, 2, 1, 2, 3, 4, 5, 1, 1, 2, 3, 4, 1, 2, 3, 4, 1~
## $ quantity       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1~
## $ price          <dbl> 6070, 5970, 2770, 5970, 10660, 3200, 12790, 5330, 1570,~
## $ total_price    <dbl> 6070, 5970, 2770, 5970, 10660, 3200, 12790, 5330, 1570,~
## $ model          <chr> "Jekyll Carbon 2", "Trigger Carbon 2", "Beast of the Ea~
## $ category_1     <chr> "Mountain", "Mountain", "Mountain", "Mountain", "Road",~
## $ category_2     <chr> "Over Mountain", "Over Mountain", "Trail", "Over Mounta~
## $ frame_material <chr> "Carbon", "Carbon", "Aluminum", "Carbon", "Carbon", "Ca~
## $ bikeshop_name  <chr> "Ithaca Mountain Climbers", "Ithaca Mountain Climbers",~
## $ city           <chr> "Ithaca", "Ithaca", "Kansas City", "Kansas City", "Loui~
## $ state          <chr> "NY", "NY", "KS", "KS", "KY", "KY", "KY", "KY", "KY", "~
```

1.0 CUSTOMER TRENDS —-

- Get Data in Format that trends can be compared
- – GOAL: Mine Customer Purchase History for similarity to other “like” customers
- – TECHNIQUES: K-Means Clustering, UMAP 2D Projection

1.1 Get Customer Trends —-

Part1: Aggregation

- Aggregating purchasing trends to customer & products is typically the way to go
- Get what are the general trends are between the products
- must be careful on how granular the aggregation can be

Pro tip: When understanding customer trends, it is important to collect:

- The unique customer name
- Attributes related to the product
- A value to measure (e.g. quantity or total price)

Pro tip 2: We convert to customer trends by:

- Aggregating within customer-product groups, then
- Normalising within customer groups to get percentages of product purchases by customer
- need to meausre that we are aggregating on (either quantity or quality, in here quantity or total_price)

```
bike_orderlines_tbl %>%
    # trying to get the essence of what they like
    select(bikeshop_name, price, model, category_1, category_2, frame_material, quantity) %>%
    # summarise + group_by
    group_by(bikeshop_name, price, model, category_1, category_2, frame_material) %>%
    summarise(quantity_purchased = sum(quantity)) %>%
    ungroup()
```

```
## `summarise()` has grouped output by 'bikeshop_name', 'price', 'model', 'category_1', 'category_2'. Y
```

```
## # A tibble: 2,513 x 7
##    bikeshop_name      price model          category_1 category_2  frame_material
##    <chr>              <dbl> <chr>          <chr>      <chr>       <chr>
##  1 Albuquerque Cycles   415 Catalyst 4     Mountain   Sport       Aluminum
##  2 Albuquerque Cycles   480 Catalyst 3     Mountain   Sport       Aluminum
##  3 Albuquerque Cycles   585 Catalyst 2     Mountain   Sport       Aluminum
##  4 Albuquerque Cycles   705 Catalyst 1     Mountain   Sport       Aluminum
##  5 Albuquerque Cycles   815 CAAD8 Claris   Road       Elite Road  Aluminum
##  6 Albuquerque Cycles   815 Trail 5        Mountain   Sport       Aluminum
##  7 Albuquerque Cycles   870 Synapse Claris Road       Endurance ~ Aluminum
##  8 Albuquerque Cycles   980 Trail 4        Mountain   Sport       Aluminum
##  9 Albuquerque Cycles  1030 CAAD8 Sora     Road       Elite Road  Aluminum
## 10 Albuquerque Cycles  1080 Trail 3        Mountain   Sport       Aluminum
## # ... with 2,503 more rows, and 1 more variable: quantity_purchased <dbl>
```

Part2: Normalisation

- Ultimately we want user item format
- Normalise the data: to be compared on -> easiest way is to compute the proportions

```
customer_trends_tbl <- bike_orderlines_tbl %>%
    # trying to get the essence of what they like
    select(bikeshop_name, price, model, category_1, category_2, frame_material, quantity) %>%
    # summarise + group_by
    group_by(bikeshop_name, price, model, category_1, category_2, frame_material) %>%
    summarise(quantity_purchased = sum(quantity)) %>%
    ungroup() %>%

    # Normalisation
    group_by(bikeshop_name) %>%
    mutate(prop_of_total = quantity_purchased/sum(quantity_purchased)) %>%
    ungroup()
```

```
## `summarise()` has grouped output by 'bikeshop_name', 'price', 'model', 'category_1', 'category_2'. Y
```

```
customer_trends_tbl
```

```
## # A tibble: 2,513 x 8
##    bikeshop_name      price model          category_1 category_2  frame_material
##    <chr>              <dbl> <chr>          <chr>      <chr>       <chr>
##  1 Albuquerque Cycles   415 Catalyst 4     Mountain   Sport       Aluminum
##  2 Albuquerque Cycles   480 Catalyst 3     Mountain   Sport       Aluminum
##  3 Albuquerque Cycles   585 Catalyst 2     Mountain   Sport       Aluminum
##  4 Albuquerque Cycles   705 Catalyst 1     Mountain   Sport       Aluminum
##  5 Albuquerque Cycles   815 CAAD8 Claris   Road       Elite Road  Aluminum
##  6 Albuquerque Cycles   815 Trail 5        Mountain   Sport       Aluminum
##  7 Albuquerque Cycles   870 Synapse Claris Road       Endurance ~ Aluminum
##  8 Albuquerque Cycles   980 Trail 4        Mountain   Sport       Aluminum
##  9 Albuquerque Cycles  1030 CAAD8 Sora     Road       Elite Road  Aluminum
## 10 Albuquerque Cycles  1080 Trail 3        Mountain   Sport       Aluminum
## # ... with 2,503 more rows, and 2 more variables: quantity_purchased <dbl>,
## #   prop_of_total <dbl>
```

1.2 Convert to User-Item Format (e.g. Customer-Product) —-

```
customer_product_tbl <- customer_trends_tbl %>%
    select(bikeshop_name, model, prop_of_total) %>%
    spread(key = model, value = prop_of_total, fill = 0)

customer_product_tbl
```

```
## # A tibble: 30 x 98
##    bikeshop_name   `Bad Habit 1` `Bad Habit 2` `Beast of the E~ `Beast of the E~
##    <chr>                   <dbl>         <dbl>            <dbl>            <dbl>
```

```
##  1 Albuquerque Cy~       0.0175        0.00699          0.0105           0.0105
##  2 Ann Arbor Speed       0.00664       0.00997          0.0150           0.00997
##  3 Austin Cruisers       0.00813       0.00407          0.00813          0.00813
##  4 Cincinnati Spe~       0.00512       0                0                0
##  5 Columbus Race ~       0.0102        0                0                0.00508
##  6 Dallas Cycles         0.0128        0.0171           0.00427          0.00427
##  7 Denver Bike Sh~       0.0117        0.0139           0.0183           0.0152
##  8 Detroit Cycles        0.00992       0.0159           0.0119           0.00595
##  9 Indianapolis V~       0.00627       0.00313          0.00940          0.00940
## 10 Ithaca Mountai~       0.0182        0.0111           0.0214           0.0182
## # ... with 20 more rows, and 93 more variables: Beast of the East 3 <dbl>,
## #   CAAD Disc Ultegra <dbl>, CAAD12 105 <dbl>, CAAD12 Black Inc <dbl>,
## #   CAAD12 Disc 105 <dbl>, CAAD12 Disc Dura Ace <dbl>, CAAD12 Red <dbl>,
## #   CAAD12 Ultegra <dbl>, CAAD8 105 <dbl>, CAAD8 Claris <dbl>,
## #   CAAD8 Sora <dbl>, CAAD8 Tiagra <dbl>, Catalyst 1 <dbl>, Catalyst 2 <dbl>,
## #   Catalyst 3 <dbl>, Catalyst 4 <dbl>, F-Si 1 <dbl>, F-Si 2 <dbl>,
## #   F-Si 3 <dbl>, F-Si Black Inc. <dbl>, F-Si Carbon 2 <dbl>, ...
```

2.0 MODELING: K-MEANS CLUSTERING —-

2.1 Performing K-Means —-

kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)

x : numeric matrix of data, or an object that can be coerced to such a matrix (such as a **numeric vector** or a **data frame with all numeric columns**).

K-means picks a random starting point and then iteratively finds the best location for the centers. Choosing nstart > 1 ensures higher likeihood that a good center is found. -> better acurracy.

The kmeans function also has an nstart option that attempts multiple initial configurations and reports on the best one. For example, adding nstart = 25 will generate 25 initial configurations. This approach is often recommended.

The output of kmeans is a list with several bits of information. The most important being:

- cluster: A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
- centers: A matrix of cluster centers.
- totss: The total sum of squares.
- withinss: Vector of within-cluster sum of squares, one component per cluster.
- tot.withinss: Total within-cluster sum of squares, i.e. sum(withinss).
- betweenss: The between-cluster sum of squares, i.e. $totss - tot.withinss$.
- size: The number of points in each cluster.

```
kmeans_obj <- customer_product_tbl %>%
    select(-bikeshop_name) %>%
    # algorithm that performs K-means clustering in R
    kmeans(3, nstart = 100)


kmeans_obj$cluster
```

```
##  [1] 1 1 1 3 3 1 1 1 1 2 1 3 1 3 1 1 1 1 1 1 1 1 2 1 1 1 3 1 2 3
```

2.2 Tidying a K-Means Object —-

tot.withinss: total Within Sum of Squares

The metric we will use to help determine what number of clusters to use

```
kmeans_obj %>%
    broom::tidy()
```

```
## # A tibble: 3 x 100
##   `Bad Habit 1` `Bad Habit 2` `Beast of the E~ `Beast of the E~ `Beast of the E~
##           <dbl>         <dbl>            <dbl>            <dbl>            <dbl>
## 1       0.0124       0.0108           0.0126           0.0118           0.0103
## 2       0.0178       0.00458          0.0121           0.0193           0.00755
## 3       0.00551      0.000446         0.000267         0.00246          0.00180
## # ... with 95 more variables: CAAD Disc Ultegra <dbl>, CAAD12 105 <dbl>,
## #   CAAD12 Black Inc <dbl>, CAAD12 Disc 105 <dbl>, CAAD12 Disc Dura Ace <dbl>,
## #   CAAD12 Red <dbl>, CAAD12 Ultegra <dbl>, CAAD8 105 <dbl>,
## #   CAAD8 Claris <dbl>, CAAD8 Sora <dbl>, CAAD8 Tiagra <dbl>, Catalyst 1 <dbl>,
## #   Catalyst 2 <dbl>, Catalyst 3 <dbl>, Catalyst 4 <dbl>, F-Si 1 <dbl>,
## #   F-Si 2 <dbl>, F-Si 3 <dbl>, F-Si Black Inc. <dbl>, F-Si Carbon 2 <dbl>,
## #   F-Si Carbon 4 <dbl>, F-Si Hi-Mod 1 <dbl>, F-Si Hi-Mod Team <dbl>, ...
```

```
kmeans_obj %>%
    broom::glance()
```

```
## # A tibble: 1 x 4
##   totss tot.withinss betweenss  iter
##   <dbl>        <dbl>     <dbl> <int>
## 1 0.184        0.121    0.0631     2
```

```
broom::augment(kmeans_obj, customer_product_tbl) %>%
    select(bikeshop_name, .cluster)
```

```
## # A tibble: 30 x 2
##    bikeshop_name            .cluster
##    <chr>                    <fct>
##  1 Albuquerque Cycles       1
##  2 Ann Arbor Speed          1
##  3 Austin Cruisers          1
```

```
##  4 Cincinnati Speed         3
##  5 Columbus Race Equipment  3
##  6 Dallas Cycles            1
##  7 Denver Bike Shop         1
##  8 Detroit Cycles           1
##  9 Indianapolis Velocipedes 1
## 10 Ithaca Mountain Climbers 2
## # ... with 20 more rows
```

2.3 How many centers (customer groups) to use? —-

- Here goal is Scree Plot: To make it we need to calculate tot.withinss for many centers
- Iteration: can use purrr::map

Step1: Create function that can be iterated: will make a kmeans mapper

Pro Tip: We can apply broom::glance() row-wise with mutate() + map()

```
# Functions that works on 1 element
center <- 3
kmeans_mapper <- function(centers = 3){
    customer_product_tbl %>%
        select(-bikeshop_name) %>%
        kmeans(centers = centers, nstart = 100)
}

3 %>% kmeans_mapper %>% glance()
```

```
## # A tibble: 1 x 4
##   totss tot.withinss betweenss  iter
##   <dbl>        <dbl>     <dbl> <int>
## 1 0.184        0.121    0.0631     1
```

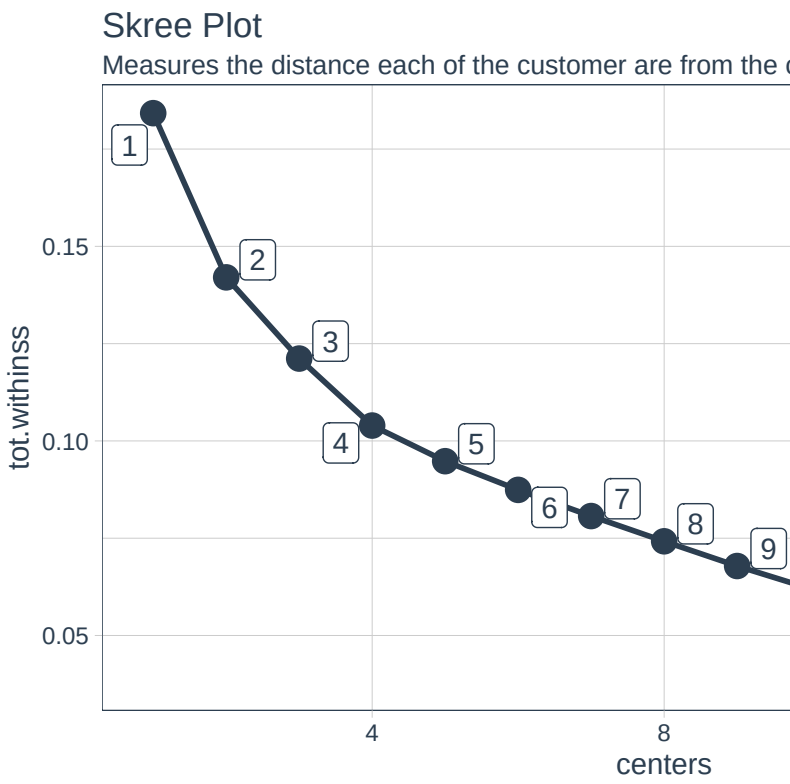
```
# Mapping the function to many elements
kmeans_mapped_tbl <- tibble(centers = 1:15) %>%
    mutate(k_means = centers %>% map(kmeans_mapper),
           glance  = k_means %>% map(glance))

kmeans_mapped_tbl
```

```
## # A tibble: 15 x 3
##    centers k_means  glance
##      <int> <list>   <list>
##  1       1 <kmeans> <tibble [1 x 4]>
##  2       2 <kmeans> <tibble [1 x 4]>
##  3       3 <kmeans> <tibble [1 x 4]>
##  4       4 <kmeans> <tibble [1 x 4]>
##  5       5 <kmeans> <tibble [1 x 4]>
##  6       6 <kmeans> <tibble [1 x 4]>
##  7       7 <kmeans> <tibble [1 x 4]>
##  8       8 <kmeans> <tibble [1 x 4]>
```

```
##  9       9 <kmeans> <tibble [1 x 4]>
## 10      10 <kmeans> <tibble [1 x 4]>
## 11      11 <kmeans> <tibble [1 x 4]>
## 12      12 <kmeans> <tibble [1 x 4]>
## 13      13 <kmeans> <tibble [1 x 4]>
## 14      14 <kmeans> <tibble [1 x 4]>
## 15      15 <kmeans> <tibble [1 x 4]>
```

2.4 Skree Plot —-

```
kmeans_mapped_tbl %>%
    unnest(glance) %>%
    select(centers, tot.withinss) %>%

    # Visualisation
    ggplot(aes(centers, tot.withinss)) +
    geom_point(colour = "#2c3e50", size = 4) +
    geom_line(colour = "#2c3e50", size = 1) +
    ggrepel::geom_label_repel(aes(label = centers), colour = "#2c3e50") +

    # Formatting
    theme_tq() +
    labs(
        # What is the plot + the analysis on
        title = "Skree Plot",
        # Explain what Sktree Plot does
        subtitle = "Measures the distance each of the customer are from the classes K-Means center",
        caption ="Based on the Skree Plot its substantial decrease in variance class-within stops after
        hence we select 4 clusters to segement the customer base."
    )
```



unnest: we can finding the optimal tot.withinss

3.0 VISUALIZATION: UMAP —-

3.1 Use UMAP to get 2-D Projection —-

Once K-Means Clustering is performed, we can use UMPA to help visualise

What UMAP does is it takes high dimensional space to 2D representation

UMAP: A dimensionality reduction technique that captures the struture of a higgh-dimension data set (many numeric columns) in a two column (x and y) data set

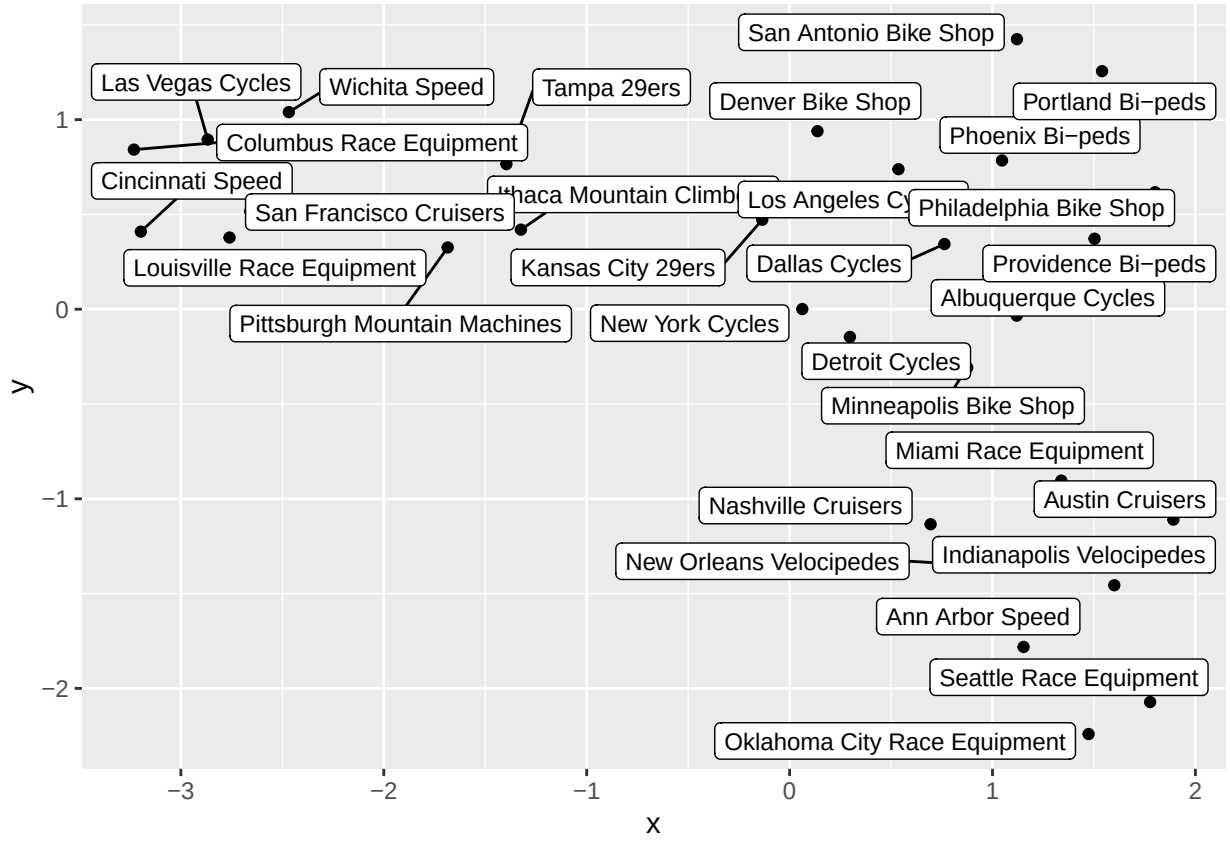
3.2 Use K-Means to Add Cluster Assignments —-

```
umap_obj <- customer_product_tbl %>%
    select(-bikeshop_name) %>%
    umap()

umap_result_tbl <- umap_obj$layout %>%
    as_tibble() %>%
    setNames(c("x", "y")) %>%
    bind_cols(
```

```
        customer_product_tbl %>% select(bikeshop_name)
        )
```

```
## Warning: The ‘x‘ argument of ‘as_tibble.matrix()‘ must have unique column names if ‘.name_repair‘ is
## Using compatibility ‘.name_repair‘.
```

```
umap_result_tbl %>%
    ggplot(aes(x, y)) +
    geom_point() +
    ggrepel::geom_label_repel(aes(label = bikeshop_name), size = 3)
```




```
umap_result_tbl
```

```
## # A tibble: 30 x 3
##         x       y bikeshop_name
##     <dbl>   <dbl> <chr>
##  1  1.12  -0.0339 Albuquerque Cycles
##  2  1.15  -1.78   Ann Arbor Speed
##  3  1.89  -1.11   Austin Cruisers
##  4 -3.20   0.409  Cincinnati Speed
##  5 -3.23   0.842  Columbus Race Equipment
##  6  0.763  0.344  Dallas Cycles
##  7  0.138  0.939  Denver Bike Shop
##  8  0.297 -0.146  Detroit Cycles
```

```
##  9  1.60  -1.46   Indianapolis Velocipedes
## 10 -1.32   0.420  Ithaca Mountain Climbers
## # ... with 20 more rows
```

```
kmeans_mapped_tbl %>%
    # When used on the k_means column, will pull the "list" of k-means objects.
    pull(k_means) %>%
    #pluck(): allows us to extract an element from a list using an number
    pluck(4)
```

```
## K-means clustering with 4 clusters of sizes 13, 8, 6, 3
##
## Cluster means:
##   Bad Habit 1  Bad Habit 2 Beast of the East 1 Beast of the East 2
## 1 0.013802693 0.0124046674         0.012339311         0.012381425
## 2 0.010127309 0.0081567290         0.012970397         0.010809606
## 3 0.005509305 0.0004456328         0.000267094         0.002464269
## 4 0.017842933 0.0045761564         0.012124846         0.019310832
##   Beast of the East 3 CAAD Disc Ultegra   CAAD12 105 CAAD12 Black Inc
## 1         0.013182655      0.0115740208 0.0149839060      0.004709450
## 2         0.005640261      0.0185458546 0.0146149539      0.011151626
## 3         0.001801082      0.0128498888 0.0131915763      0.019470962
## 4         0.007548689      0.0008841733 0.0005274262      0.004141764
##   CAAD12 Disc 105 CAAD12 Disc Dura Ace CAAD12 Red CAAD12 Ultegra   CAAD8 105
## 1    0.0167512345          0.007461632 0.01217294    0.010717975 0.014093013
## 2    0.0157947427          0.008315884 0.02288873    0.020073480 0.018970372
## 3    0.0140125911          0.016429937 0.02119633    0.010226163 0.014117983
## 4    0.0002637131          0.019569479 0.01125606    0.001147886 0.001411599
##   CAAD8 Claris CAAD8 Sora CAAD8 Tiagra   Catalyst 1  Catalyst 2  Catalyst 3
## 1  0.011433978 0.01562627  0.013587168 0.0143367718 0.013874279 0.018260121
## 2  0.017460568 0.01748864  0.018239147 0.0086857029 0.008689460 0.007306948
## 3  0.008965022 0.01360573  0.008164822 0.0003287311 0.002395398 0.000267094
## 4  0.002145549 0.00000000  0.001054852 0.0067942842 0.007320821 0.003200401
##    Catalyst 4     F-Si 1      F-Si 2      F-Si 3 F-Si Black Inc. F-Si Carbon 2
## 1 0.011741300 0.01232986 0.020333717 0.013858022     0.007515234   0.008224163
## 2 0.013484616 0.00707930 0.015032908 0.007698292     0.002158678   0.001946771
## 3 0.001948048 0.00000000 0.001494634 0.002867906     0.006617801   0.011014437
## 4 0.021563915 0.01740943 0.007812402 0.008469597     0.020401529   0.015604349
##   F-Si Carbon 4 F-Si Hi-Mod 1 F-Si Hi-Mod Team   Fat CAAD1   Fat CAAD2
## 1  0.0184439194  0.0047479739      0.006260328 0.005988176 0.012995166
## 2  0.0100258062  0.0006757989      0.002117703 0.002476033 0.006692543
## 3  0.0004230118  0.0100303842      0.010135756 0.011556807 0.000000000
## 4  0.0087324202  0.0139290368      0.010713246 0.019290376 0.007621268
##       Habit 4     Habit 5      Habit 6 Habit Carbon 1 Habit Carbon 2
## 1 0.015224964 0.010558685 0.0141413108    0.005118628   0.0053708763
## 2 0.004316084 0.010898924 0.0104405589    0.001445087   0.0005277191
## 3 0.000000000 0.001382386 0.0004230118    0.009406325   0.0074722118
## 4 0.011250995 0.005289651 0.0042347984    0.016752236   0.0233749513
##   Habit Carbon 3 Habit Carbon SE Habit Hi-Mod Black Inc. Jekyll Carbon 1
## 1   0.0074212654     0.005399994            0.0073313814    0.0080268839
## 2   0.0007853631     0.002068708            0.0003158663    0.0004869932
## 3   0.0117454110     0.009700179            0.0089581155    0.0080848937
## 4   0.0089602888     0.020195895            0.0115197746    0.0185051932
##   Jekyll Carbon 2 Jekyll Carbon 3 Jekyll Carbon 4 Scalpel 29 4
```

```
## 1    0.0065146172    0.0043890540     0.011301750  0.014644215
## 2    0.0007444747    0.0005235075     0.008188386  0.008581718
## 3    0.0103110625    0.0117342980     0.007885248  0.011063747
## 4    0.0210791785    0.0132881212     0.030281835  0.013137898
##   Scalpel 29 Carbon 2 Scalpel 29 Carbon 3 Scalpel 29 Carbon Race Scalpel-Si 5
## 1        0.0093866068         0.007534370            0.005445131  0.009767559
## 2        0.0009243285         0.003114080            0.002583885  0.009972947
## 3        0.0087130196         0.004663749            0.009271865  0.009395607
## 4        0.0145853415         0.013515100            0.028039075  0.014492307
##   Scalpel-Si Black Inc. Scalpel-Si Carbon 2 Scalpel-Si Carbon 3
## 1           0.008249067         0.006411558        0.0055253318
## 2           0.001271297         0.001900291        0.0007996207
## 3           0.011313388         0.006668898        0.0129937385
## 4           0.019538700         0.012254614        0.0342691993
##   Scalpel-Si Carbon 4 Scalpel-Si Hi-Mod 1 Scalpel-Si Race  Slice 105
## 1         0.004927821        0.0065791676    0.0117131112 0.01398411
## 2         0.001566129        0.0008615088    0.0005689129 0.01724161
## 3         0.010273647        0.0135783271    0.0128670043 0.01100159
## 4         0.017786633        0.0180853006    0.0096379385 0.00000000
##   Slice Hi-Mod Black Inc. Slice Hi-Mod Dura Ace D12 Slice Ultegra
## 1             0.003827398               0.005208396    0.01882103
## 2             0.013543162               0.010280295    0.02207715
## 3             0.023493640               0.022961252    0.01337025
## 4             0.016405811               0.007641723    0.00000000
##   Slice Ultegra D12 Supersix Evo 105 Supersix Evo Black Inc.
## 1        0.01474522     0.0139263621             0.003619526
## 2        0.01953066     0.0152111206             0.015364196
## 3        0.01768950     0.0085368555             0.019639410
## 4        0.01118437     0.0008841733             0.008226339
##   Supersix Evo Hi-Mod Dura Ace 1 Supersix Evo Hi-Mod Dura Ace 2
## 1                    0.003855520                    0.005015162
## 2                    0.006797696                    0.009457052
## 3                    0.023052853                    0.015056884
## 4                    0.005666853                    0.021057833
##   Supersix Evo Hi-Mod Team Supersix Evo Hi-Mod Utegra Supersix Evo Red
## 1              0.004651811                0.005168541      0.004859132
## 2              0.013338752                0.012258727      0.015947776
## 3              0.016159403                0.019819091      0.015662413
## 4              0.008753765                0.008769154      0.005460330
##   Supersix Evo Tiagra Supersix Evo Ultegra 3 Supersix Evo Ultegra 4  SuperX 105
## 1        0.0120113072             0.01290187            0.010908348 0.012159237
## 2        0.0180526527             0.02185746            0.019434511 0.014852613
## 3        0.0132322158             0.02018735            0.007013335 0.009595961
## 4        0.0002637131             0.00873331            0.001411599 0.000000000
##   SuperX Hi-Mod CX1 SuperX Rival CX1 SuperX Ultegra Syapse Carbon Tiagra
## 1       0.005341603       0.01414232    0.009249563           0.01061743
## 2       0.010741390       0.01354822    0.019329642           0.01836610
## 3       0.018457356       0.01007350    0.010686315           0.01083672
## 4       0.020345229       0.00000000    0.001147886           0.00000000
##   Synapse Carbon 105 Synapse Carbon Disc 105 Synapse Carbon Disc Ultegra
## 1        0.012124343             0.016061741                 0.005985173
## 2        0.019392078             0.017782381                 0.008273778
## 3        0.008827204             0.009048104                 0.020239265
## 4        0.002295773             0.001147886                 0.007962626
```

```
##   Synapse Carbon Disc Ultegra D12 Synapse Carbon Ultegra 3
## 1                     0.004294859               0.01359110
## 2                     0.012695868               0.01694845
## 3                     0.019516155               0.01409720
## 4                     0.012610471               0.01087797
##   Synapse Carbon Ultegra 4 Synapse Claris Synapse Disc 105
## 1               0.01391907    0.012970439     0.0149048704
## 2               0.02037167    0.016090883     0.0211219205
## 3               0.01163216    0.007880484     0.0096063218
## 4               0.00000000    0.001147886     0.0007911392
##   Synapse Disc Adventure Synapse Disc Tiagra Synapse Hi-Mod Disc Black Inc.
## 1            0.011519875        0.0130158269                    0.003502643
## 2            0.015035726        0.0246075675                    0.007196491
## 3            0.010220133        0.0088130693                    0.019755335
## 4            0.002559486        0.0002637131                    0.009167702
##   Synapse Hi-Mod Disc Red Synapse Hi-Mod Disc Ultegra Synapse Hi-Mod Dura Ace
## 1             0.006978382                 0.005376918             0.004202161
## 2             0.011023980                 0.008643747             0.008990878
## 3             0.024674664                 0.019203354             0.021671579
## 4             0.007435200                 0.007000808             0.013474189
##   Synapse Sora      Trail 1    Trail 2     Trail 3      Trail 4     Trail 5
## 1 0.0133648609 0.0153806548 0.01280364 0.013406852 0.0127528798 0.015531621
## 2 0.0223053056 0.0130605433 0.01087278 0.008864080 0.0097495605 0.007173440
## 3 0.0134624162 0.0006901059 0.00117800 0.001717914 0.0006933515 0.001336898
## 4 0.0005274262 0.0055533638 0.01327273 0.007812402 0.0159978306 0.018477892
##   Trigger Carbon 1 Trigger Carbon 2 Trigger Carbon 3 Trigger Carbon 4
## 1      0.006494281     0.0072887534      0.007317921       0.01508253
## 2      0.001151778     0.0009197311      0.002061166       0.01023017
## 3      0.010344458     0.0080748275      0.015262457       0.01582411
## 4      0.012688116     0.0210433340      0.025935327       0.02326057
##
## Clustering vector:
##  [1] 1 2 2 3 3 1 1 1 2 4 1 3 1 3 2 1 2 2 1 2 1 1 4 1 1 1 3 2 4 3
##
## Within cluster sum of squares by cluster:
## [1] 0.049009382 0.028223868 0.017028496 0.009686516
##  (between_SS / total_SS =  43.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

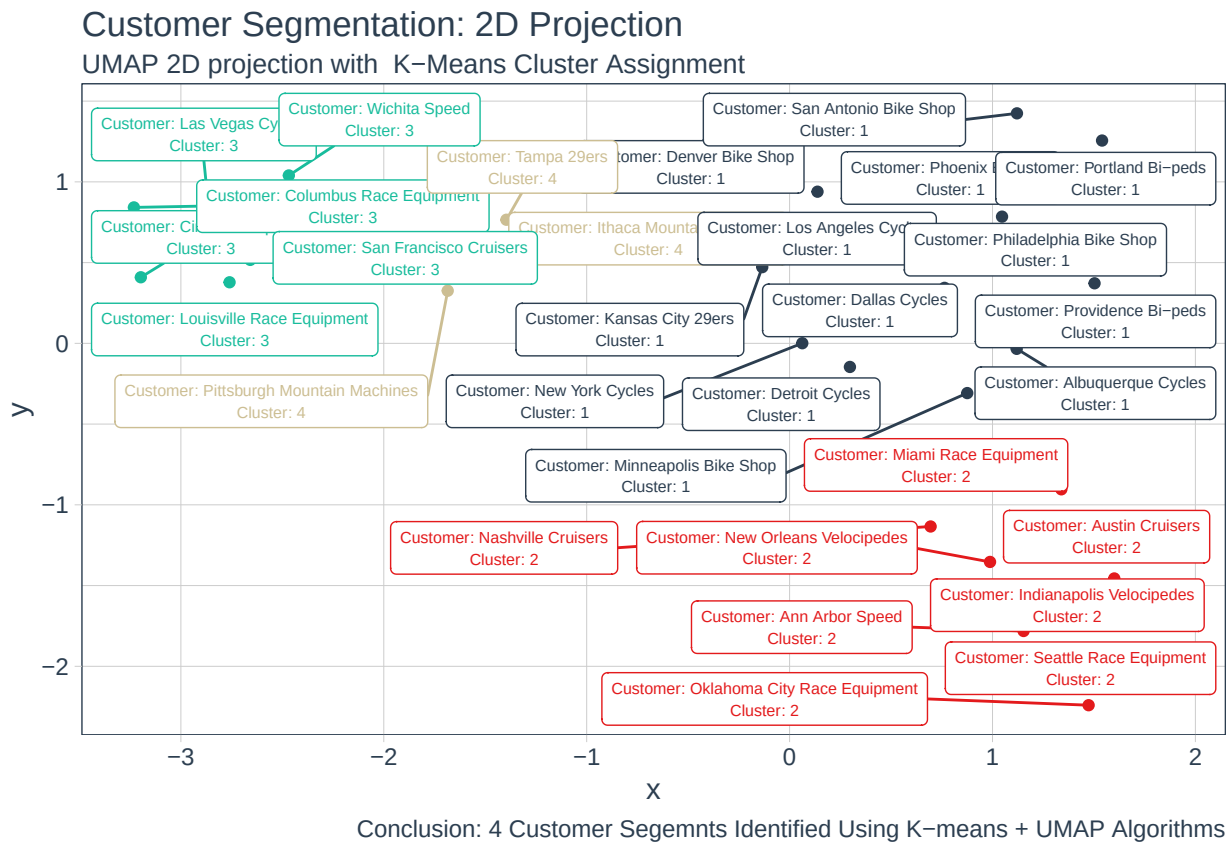
```
# Could also done the same thing by:
kmeans_4_obj <- kmeans_mapped_tbl %>%
    filter(centers == 4) %>%
    pull(k_means) %>%
    pluck(1)

kmeans_4_clusters_tbl  <- kmeans_4_obj %>%
    augment(customer_product_tbl) %>%
    select(bikeshop_name, .cluster)

umap_kmeans_4_results_tbl <- umap_result_tbl %>%
```

```
    left_join(kmeans_4_clusters_tbl)
```

```
## Joining, by = "bikeshop_name"
```

3.3 Visualize UMAP'ed Projections with Cluster Assignments —-

```
umap_kmeans_4_results_tbl %>%
    mutate(label_text = str_glue("Customer: {bikeshop_name}
                                  Cluster: {.cluster}")) %>%
    ggplot(aes(x = x, y = y, colour = .cluster))+
    geom_point() +
    ggrepel::geom_label_repel(aes(label = label_text), size = 2) +
    # Formatting
    theme_tq() +
    scale_colour_tq() +
    labs(
        title = "Customer Segmentation: 2D Projection",
        subtitle = "UMAP 2D projection with  K-Means Cluster Assignment",
        caption = "Conclusion: 4 Customer Segemnts Identified Using K-means + UMAP Algorithms"
    ) +
    theme(legend.position = "none")
```



4.0 ANALYZE PURCHASING TRENDS —-

```
customer_trends_tbl %>%
    pull(price) %>%
    quantile(probs = seq(0,1,1/3))
```

```
##        0% 33.33333% 66.66667%      100%
##       415      2240      4260     12790
```

```
cluser_trend_tbl <- customer_trends_tbl %>%
    # join cluster assigment by bikeshop name
    left_join(umap_kmeans_4_results_tbl, by = "bikeshop_name") %>%
    mutate(price_bin = case_when(price >= quantile(price,
                                                   probs = seq(0,1,1/3))[3]
                                 ~ "high",
                                 price >= quantile(price,
                                                   probs = seq(0,1,1/3))[2]
                                 ~ "medium",
                                 TRUE ~ "low")) %>%
    select(.cluster, model, contains("price"),
           category_1:quantity_purchased) %>%
    # Aggregate quantity purchased by cluster and product attributes
    # group_by_at: A scoped cariant of group_by() that allows us to select columns with tidy_select hel
    group_by_at(.vars = vars(.cluster:frame_material)) %>%
    summarise(total_quantity = sum(quantity_purchased)) %>%
    ungroup() %>%

    # Normalise data
    # Calculate proportion of total
    group_by(.cluster) %>%
    mutate(prop_of_total = total_quantity/sum(total_quantity)) %>%
    ungroup()
```

```
## `summarise()` has grouped output by '.cluster', 'model', 'price', 'price_bin', 'category_1', 'catego
```

```
# Cluster 1 - Low/Medium Price, Road Model Preference
cluser_trend_tbl %>%
    filter(.cluster == 2) %>%
    arrange(desc(prop_of_total)) %>%
    mutate(cum_prop = cumsum(prop_of_total))
```

In marketing do not just show the visualisation, after identifying the clusters dive little deeper and investigate what are the preferences (top products, revenue contribution) are in each clusters.

```
## # A tibble: 97 x 10
##    .cluster model          price price_bin category_1 category_2  frame_material
##    <fct>    <chr>          <dbl> <chr>     <chr>      <chr>       <chr>
```

```
##  1 2        Synapse Disc ~  1250 low       Road       Endurance ~ Aluminum
##  2 2        SuperX Ultegra  2450 medium    Road       Cyclocross  Carbon
##  3 2        CAAD12 Red      3200 medium    Road       Elite Road  Aluminum
##  4 2        Slice Ultegra   2700 medium    Road       Triathalon  Carbon
##  5 2        Synapse Sora    1030 low       Road       Endurance ~ Aluminum
##  6 2        Supersix Evo ~  3200 medium    Road       Elite Road  Carbon
##  7 2        Supersix Evo ~  2660 medium    Road       Elite Road  Carbon
##  8 2        Supersix Evo ~  1840 low       Road       Elite Road  Carbon
##  9 2        CAAD8 Sora      1030 low       Road       Elite Road  Aluminum
## 10 2        Synapse Carbo~  2660 medium    Road       Endurance ~ Carbon
## # ... with 87 more rows, and 3 more variables: total_quantity <dbl>,
## #   prop_of_total <dbl>, cum_prop <dbl>
```

```
get_cluster_tend <- function(cluster = 1){
    cluser_trend_tbl %>%
    filter(.cluster == cluster) %>%
    arrange(desc(prop_of_total)) %>%
    mutate(cum_prop = cumsum(prop_of_total))
}

# Cluster 1 - low/medium Price, Mountain Model, Sport Frame Material Preference
get_cluster_tend(1)
```

```
## # A tibble: 97 x 10
##    .cluster model         price price_bin category_1 category_2   frame_material
##    <fct>    <chr>         <dbl> <chr>     <chr>      <chr>        <chr>
##  1 1        F-Si Carbon 4  2880 medium    Mountain   Cross Count~ Carbon
##  2 1        F-Si 2         2060 low       Mountain   Cross Count~ Aluminum
##  3 1        Trail 5         815 low       Mountain   Sport        Aluminum
##  4 1        Scalpel 29 4   3200 medium    Mountain   Cross Count~ Aluminum
##  5 1        Catalyst 3      480 low       Mountain   Sport        Aluminum
##  6 1        Catalyst 2      585 low       Mountain   Sport        Aluminum
##  7 1        Trail 1        1520 low       Mountain   Sport        Aluminum
##  8 1        Trail 4         980 low       Mountain   Sport        Aluminum
##  9 1        F-Si 1         2340 medium    Mountain   Cross Count~ Aluminum
## 10 1        F-Si 3         1840 low       Mountain   Cross Count~ Aluminum
## # ... with 87 more rows, and 3 more variables: total_quantity <dbl>,
## #   prop_of_total <dbl>, cum_prop <dbl>
```

```
# Cluster 3 - Medium/High Price, Mountain model, Carbon Frame Material Preference
get_cluster_tend(3)
```

```
## # A tibble: 94 x 10
##    .cluster model          price price_bin category_1 category_2  frame_material
##    <fct>    <chr>          <dbl> <chr>     <chr>      <chr>       <chr>
##  1 3        Synapse Hi-Mo~  7460 high      Road       Endurance ~ Carbon
##  2 3        Supersix Evo ~  7990 high      Road       Elite Road  Carbon
##  3 3        Slice Hi-Mod ~  7000 high      Road       Triathalon  Carbon
##  4 3        Slice Hi-Mod ~  4500 high      Road       Triathalon  Carbon
##  5 3        Synapse Hi-Mo~  5860 high      Road       Endurance ~ Carbon
##  6 3        Synapse Carbo~  4800 high      Road       Endurance ~ Carbon
##  7 3        Supersix Evo ~  4260 high      Road       Elite Road  Carbon
##  8 3        Supersix Evo ~  3200 medium    Road       Elite Road  Carbon
```

```
##  9 3        Synapse Carbo~  3730 medium    Road       Endurance ~ Carbon
## 10 3        Synapse Hi-Mo~  5330 high      Road       Endurance ~ Carbon
## # ... with 84 more rows, and 3 more variables: total_quantity <dbl>,
## #   prop_of_total <dbl>, cum_prop <dbl>
```

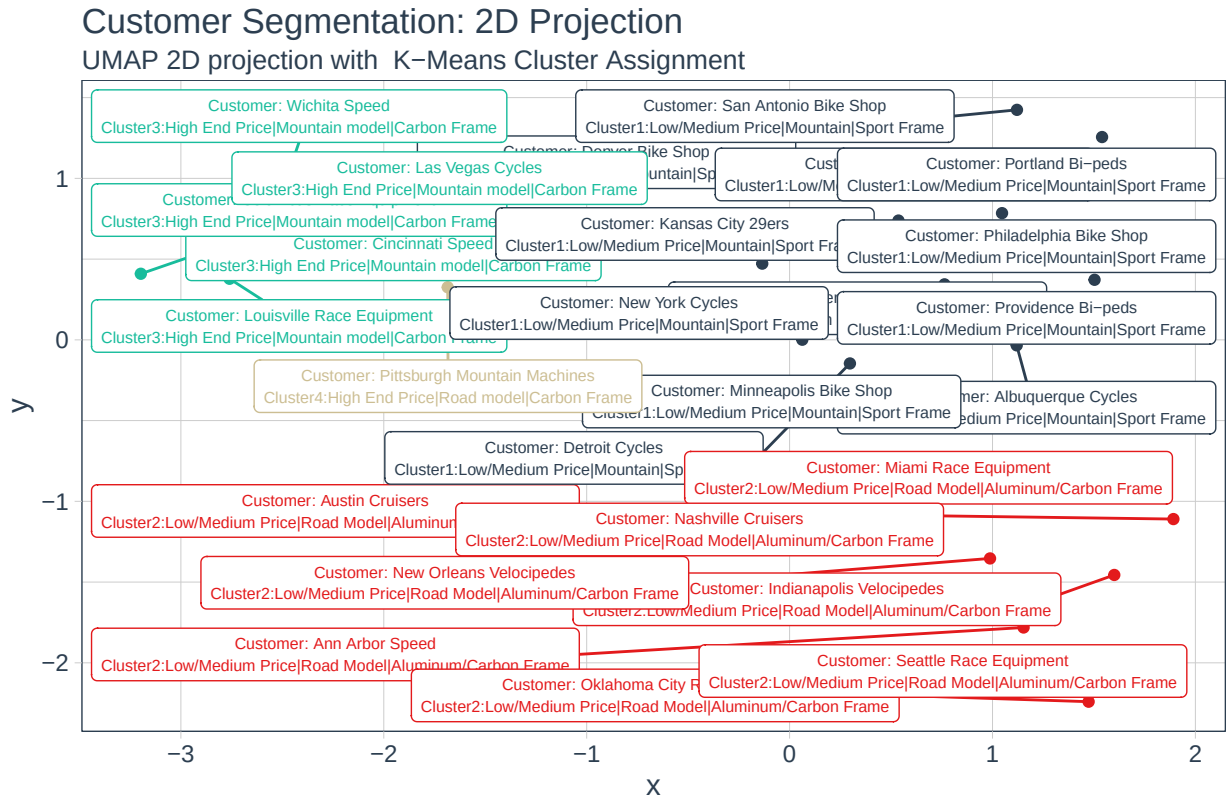
```
# Cluster 3 - High End Price, Road model, Carbon Frame Material Preference
get_cluster_tend(4)
```

```
## # A tibble: 90 x 10
##    .cluster model        price price_bin category_1 category_2    frame_material
##    <fct>    <chr>        <dbl> <chr>     <chr>      <chr>         <chr>
##  1 4        Scalpel-Si ~  5330 high      Mountain   Cross Countr~ Carbon
##  2 4        Trigger Car~  3730 medium    Mountain   Over Mountain Carbon
##  3 4        Jekyll Carb~  3200 medium    Mountain   Over Mountain Carbon
##  4 4        Jekyll Carb~  7990 high      Mountain   Over Mountain Carbon
##  5 4        Scalpel 29 ~  6390 high      Mountain   Cross Countr~ Carbon
##  6 4        Habit Carbo~  4480 high      Mountain   Trail         Carbon
##  7 4        Trigger Car~  5970 high      Mountain   Over Mountain Carbon
##  8 4        Habit Carbo~  5330 high      Mountain   Trail         Carbon
##  9 4        Scalpel-Si ~ 12790 high      Mountain   Cross Countr~ Carbon
## 10 4        Scalpel-Si ~  4260 high      Mountain   Cross Countr~ Carbon
## # ... with 80 more rows, and 3 more variables: total_quantity <dbl>,
## #   prop_of_total <dbl>, cum_prop <dbl>
```

Update Visaulisation

```
cluster_label_tbl <- tibble(
    .cluster = 1:4,
    .cluster_label = c(
        "Low/Medium Price|Mountain|Sport Frame",
        "Low/Medium Price|Road Model|Aluminum/Carbon Frame",
        "High End Price|Mountain model|Carbon Frame",
        "High End Price|Road model|Carbon Frame")) %>%
    mutate(.cluster = as_factor(as.character(.cluster)))

umap_kmeans_4_results_tbl %>%
    left_join(cluster_label_tbl, by = ".cluster") %>%
    mutate(label_text = str_glue("Customer: {bikeshop_name}
                                  Cluster{.cluster}:{.cluster_label}")) %>%
    ggplot(aes(x = x, y = y, colour = .cluster))+
    geom_point() +
    ggrepel::geom_label_repel(aes(label = label_text), size = 2) +
    # Formatting
    theme_tq() +
    scale_colour_tq() +
    labs(
        title = "Customer Segmentation: 2D Projection",
        subtitle = "UMAP 2D projection with  K-Means Cluster Assignment",
        caption = "Conclusion: 4 Customer Segemnts Identified Using K-means + UMAP Algorithms"
    ) + theme(legend.position = "none")
```

```
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Conclusion: 4 Customer Segemnts Identified Using K−means + UMAP Algorithms

```
plot_umap <- function(x, labels,
          main="A UMAP visualization",
          colors=c("#ff7f00", "#e377c2", "#17becf"),
          pad=0.1, cex=0.6, pch=19, add=FALSE, legend.suffix="",
          cex.main=1, cex.legend=0.85) {

  layout = x
  if (is(x, "umap")) {
    layout = x$layout
  }

  xylim = range(layout)
  xylim = xylim + ((xylim[2]-xylim[1])*pad)*c(-0.5, 0.5)
  if (!add) {
    par(mar=c(0.2,0.7,1.2,0.7), ps=10)
    plot(xylim, xylim, type="n", axes=F, frame=F)
    rect(xylim[1], xylim[1], xylim[2], xylim[2], border="#aaaaaa", lwd=0.25)
  }
  points(layout[,1], layout[,2], col=colors[as.integer(labels)],
         cex=cex, pch=pch)
  mtext(side=3, main, cex=cex.main)
```

```
  labels.u = unique(labels)
  legend.pos = "topleft"
  legend.text = as.character(labels.u)
  if (add) {
    legend.pos = "bottomleft"
    legend.text = paste(as.character(labels.u), legend.suffix)
  }

  legend(legend.pos, legend=legend.text, inset=0.03,
         col=colors[as.integer(labels.u)],
         bty="n", pch=pch, cex=cex.legend)
}
```