Milestone 3

Group D

2024-10-15

Load all the necessary libraries

```
library(tidyverse)   # For data manipulation

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
all conflicts to become errors

library(cluster)     # For clustering algorithms
library(factoextra)  # For visualization

## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

library(ggplot2)
```

Loading the dataset

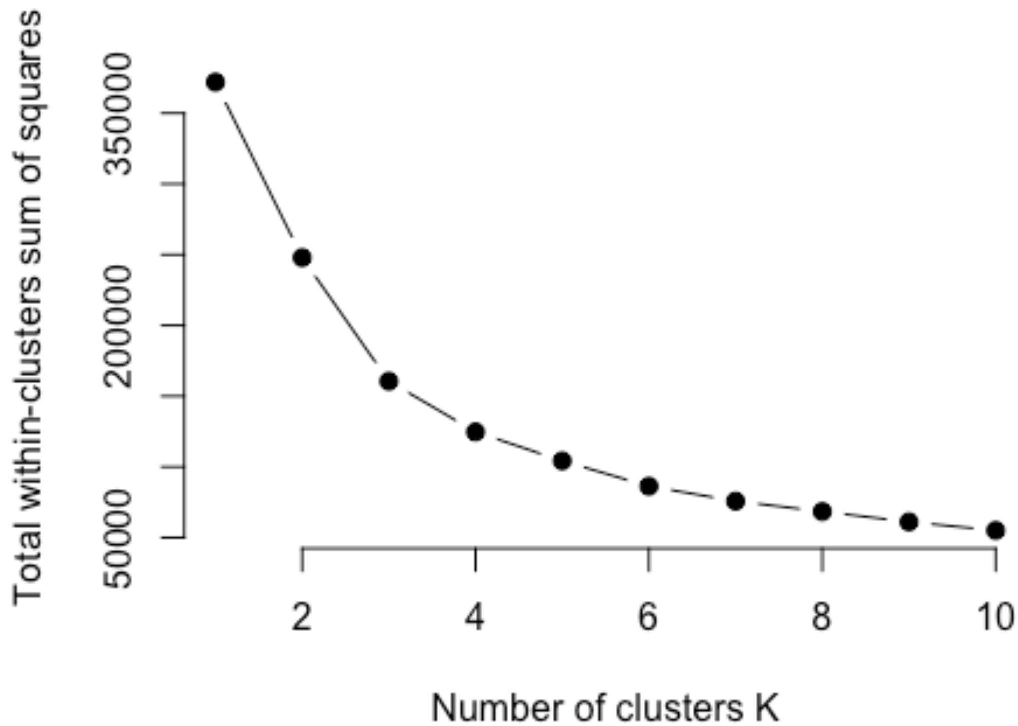
```
# Load the dataset
CustData <- read.csv("Prepared_Data.csv")

# Define affordability based on the threshold
CustData$Affordability <- ifelse(CustData$Annual.Salary >= 50000, "Can
Afford", "Cannot Afford")

# Select relevant features for clustering
data_for_clustering <- CustData %>%
  select(Annual.Salary, yrs_residence, Age) %>%
  na.omit() # Remove rows with missing values

# Standardize the data to ensure all features are on the same scale
data_standardized <- scale(data_for_clustering)
```

Perform PCA

```
pca_result <- prcomp(data_standardized, center = TRUE, scale. = TRUE)

# Extract the first two principal components
pca_data <- as.data.frame(pca_result$x[, 1:2])
pca_data$Affordability <- CustData$Affordability  # Add affordability
information for visualization
```

K-Means clustering

```
set.seed(123)
wss <- function(k) {
  kmeans(pca_data[, 1:2], centers = k, nstart = 25)$tot.withinss
}

# Compute within-cluster sum of squares for k = 1 to k = 10
k.values <- 1:10
wss_values <- map_dbl(k.values, wss)

# Plot the Elbow Method
plot(k.values, wss_values,
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Total within-clusters sum of squares")
```




```
#The Elbow Method is a commonly used technique for determining the optimal
number of clusters (k) in k-means clustering. The optimal k is identified at
the point where the WSS curve begins to flatten, forming an "elbow" shape.
```

Model evaluation

```
# Set a seed for reproducibility
set.seed(123)

# Perform k-means clustering with an appropriate number of clusters
kmeans_result <- kmeans(pca_data[, 1:2], centers = 6, nstart = 25)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 9252800)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 9252800)

# Add cluster assignments to PCA data
pca_data$Cluster <- as.factor(kmeans_result$cluster)


print(kmeans_result)

## K-means clustering with 6 clusters of sizes 40227, 25757, 31537, 26292,
31158, 30085
##
## Cluster means:
```

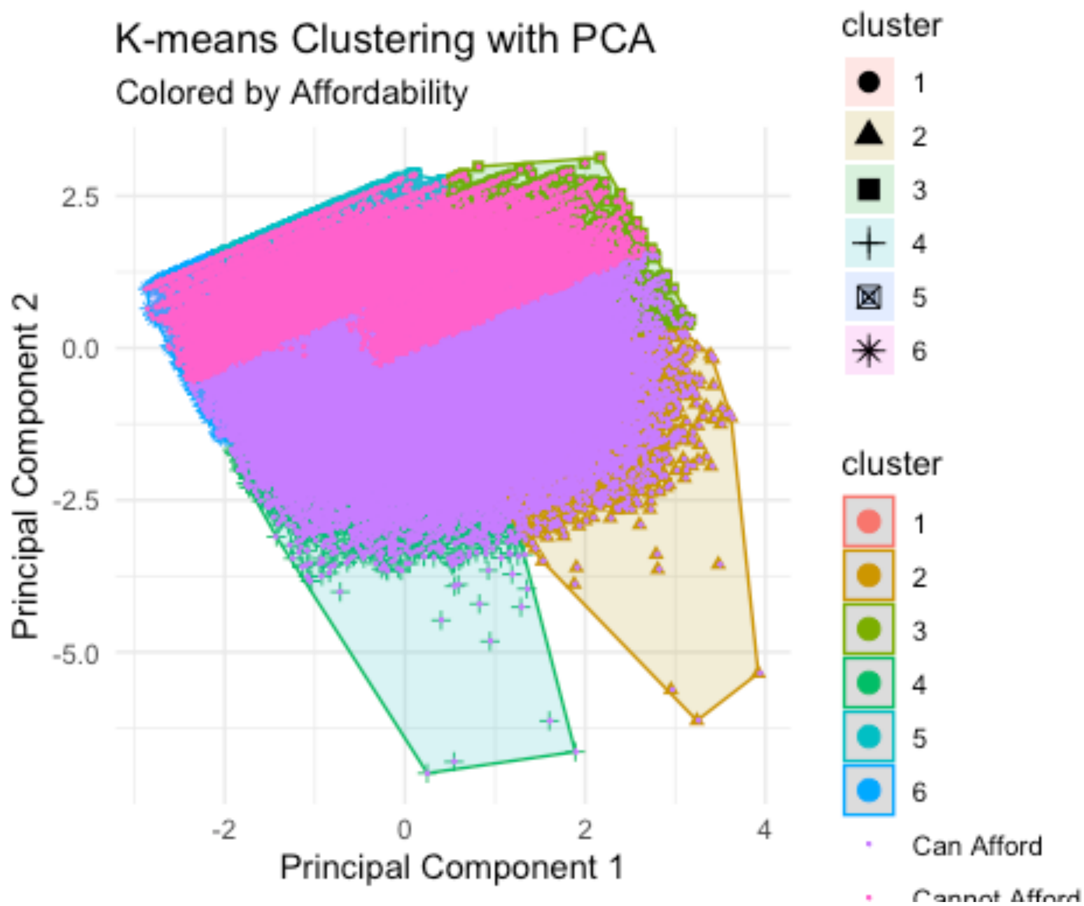
```
##             PC1         PC2
## 1  0.003739749 -0.06353574
## 2  1.293388682 -0.81328263
## 3  1.028008739  0.92733915
## 4 -0.345316833 -1.49064868
## 5 -0.482727177  1.10537858
## 6 -1.388222677 -0.03294574
##
## Clustering vector:
##     1     2     3     4     5     6     7     8     9    10    11    12
13
##     1     1     3     1     2     1     2     5     2     1     5     1
2
##    14    15    16    17    18    19    20    21    22    23    24    25
26
##     6     5     1     4     3     1     1     1     1     3     2     1
1
##    27    28    29    30    31    32    33    34    35    36    37    38
39
##     5     4     1     2     1     5     2     6     6     1     1     3
3
##    40    41    42    43    44    45    46    47    48    49    50    51
52
##     3     2     5     1     5     1     1     6     1     1     1     1
1
##    53    54    55    56    57    58    59    60    61    62    63    64
65
##     2     1     2     6     1     5     1     5     1     2     2     1
1
##    66    67    68    69    70    71    72    73    74    75    76    77
78
##     1     1     6     1     5     3     1     1     3     3     1     1
3
##    79    80    81    82    83    84    85    86    87    88    89    90
91
##     1     2     2     5     1     5     3     3     3     1     3     1
3
##    92    93    94    95    96    97    98    99   100   101   102   103
104
##     3     3     4     3     1     3     2     1     5     2     3     5
1
##   105   106   107   108   109   110   111   112   113   114   115   116
117
##     1     5     1     4     2     2     3     3     2     3     4     2
3
##   118   119   120   121   122   123   124   125   126   127   128   129
130
##     1     2     4     1     2     2     3     3     2     3     2     3
5
##   131   132   133   134   135   136   137   138   139   140   141   142
```

```
143
##     1     1     1     3     3     1     1     3     3     1     4     3 
3 
##   144   145   146   147   148   149   150   151   152   153   154   155 
156 
##     1     2     5     3     3     1     1     2     2     3     1     4 
3 
##   157   158   159   160   161   162   163   164   165   166   167   168 
169 
##     2     1     3     3     3     3     3     4     2     3     3     4 
5 
##   170   171   172   173   174   175   176   177   178   179   180   181 
182 
##     1     1     4     3     3     3     3     3     3     1     1     1 
6 
##   183   184   185   186   187   188   189   190   191   192   193   194 
195 
##     2     1     3     4     2     3     2     4     1     2     3     1 
3 
##   196   197   198   199   200   201   202   203   204   205   206   207 
208 
##     3     3     2     3     3     2     5     4     2     1     4     1 
1 
##   209   210   211   212   213   214   215   216   217   218   219   220 
221 
##     5     3     3     3     1     2     3     3     1     4     1     2 
3 
##   222   223   224   225   226   227   228   229   230   231   232   233 
234 
##     2     3     1     4     2     6     5     3     3     6     3     3 
1 
##   235   236   237   238   239   240   241   242   243   244   245   246 
247 
##     1     1     3     1     1     1     1     5     4     3     2     5 
3 
##   248   249   250   251   252   253   254   255   256   257   258   259 
260 
##     5     3     4     5     2     3     1     6     4     5     1     3 
2 
##   261   262   263   264   265   266   267   268   269   270   271   272 
273 
##     3     1     2     3     5     5     6     1     6     4     3     5 
## 99893 99894 99895 99896 99897 99898 99899 99900 99901 99902 99903 99904 
99905 
##     6     5     6     2     1     5     2     1     4     1     2     1 
4 
## 99906 99907 99908 99909 99910 99911 99912 99913 99914 99915 99916 99917 
99918 
##     4     1     1     2     1     1     4     1     1     6     1     4 
1 
```

```
## 99919 99920 99921 99922 99923 99924 99925 99926 99927 99928 99929 99930
99931
##     1     1     4     5     6     4     4     1     4     6     6     4
1
## 99932 99933 99934 99935 99936 99937 99938 99939 99940 99941 99942 99943
99944
##     5     4     1     6     5     5     1     5     5     4     1     1
5
## 99945 99946 99947 99948 99949 99950 99951 99952 99953 99954 99955 99956
99957
##     6     5     6     6     5     1     2     5     2     5     3     6
4
## 99958 99959 99960 99961 99962 99963 99964 99965 99966 99967 99968 99969
99970
##     5     2     5     3     1     1     5     6     5     2     1     6
4
## 99971 99972 99973 99974 99975 99976 99977 99978 99979 99980 99981 99982
99983
##     4     3     6     2     1     1     4     5     1     5     1     5
4
## 99984 99985 99986 99987 99988 99989 99990 99991 99992 99993 99994 99995
99996
##     4     1     1     1     5     6     6     1     1     3     5     1
3
## 99997 99998 99999
##     3     5     5
##  [ reached getOption("max.print") -- omitted 85057 entries ]
##
## Within cluster sum of squares by cluster:
## [1] 11402.60 15898.70 15870.08 16254.74 12996.60 13884.38
##  (between_SS / total_SS =  76.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

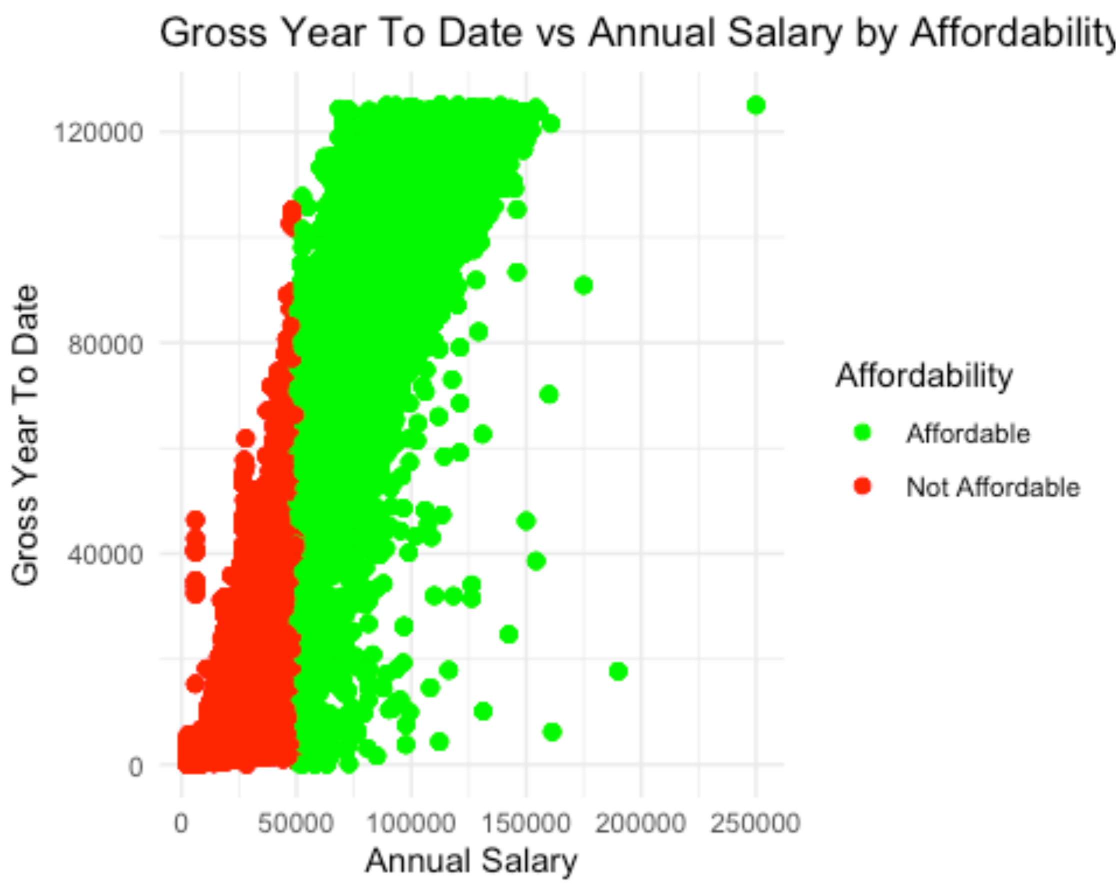
```
cluster_plot <- fviz_cluster(
  kmeans_result,
  data = pca_data[, 1:2],
  geom = "point",
  ellipse.type = "convex", # Draw ellipses around clusters
  #repel = TRUE,            # Avoid overlapping text labels
  #palette = "jco",         # Choose a color palette
  ggtheme = theme_minimal()
)

# Add a custom ggplot layer for affordability coloring
cluster_plot +
```

```
  geom_point(data = pca_data, aes(x = PC1, y = PC2, color = Affordability),
size = 0.1) +
  labs(
    title = "K-means Clustering with PCA",
    subtitle = "Colored by Affordability",
    x = "Principal Component 1",
    y = "Principal Component 2"
  ) +
  theme_minimal() +
  theme(legend.position = "right")
```

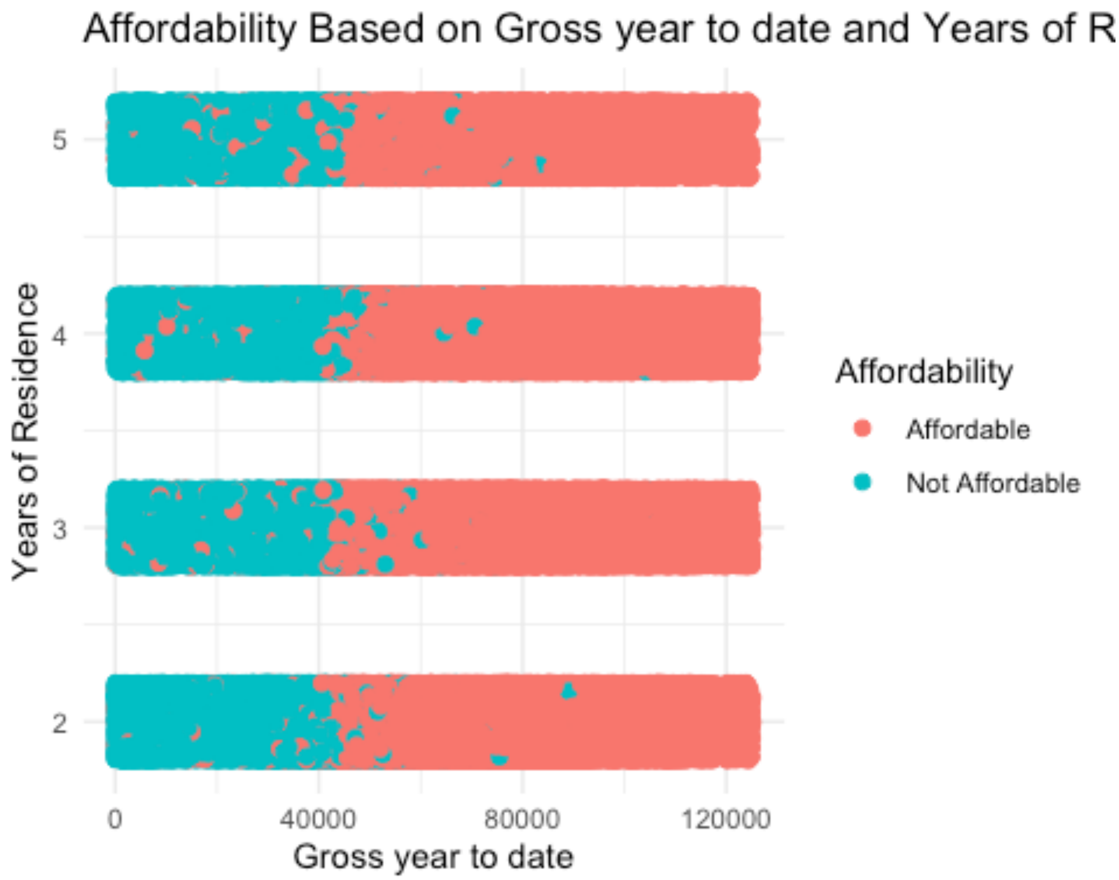



```
CustData$Affordability <- ifelse(CustData$Annual.Salary >= 50000,
"Affordable", "Not Affordable")

# Scatter plot of Gross Year To Date vs Annual Salary, colored by
affordability
ggplot(CustData, aes(x = Annual.Salary, y = Gross_Year_To_Date, color =
Affordability)) +
  geom_point(size = 2) +
  labs(title = "Gross Year To Date vs Annual Salary by Affordability",
       x = "Annual Salary",
       y = "Gross Year To Date") +
  scale_color_manual(values = c("Affordable" = "green", "Not Affordable" =
```

```
"red")) +
  theme_minimal()
```




```
# Optionally, visualize other factors like household size and years of
residence against affordability
ggplot(CustData, aes(x = Gross_Year_To_Date, y = yrs_residence, color =
Affordability)) +
  geom_jitter(width = 0.2, height = 0.2, size = 2) +
  labs(title = "Affordability Based on Gross year to date and Years of
Residence",
       x = "Gross year to date", y = "Years of Residence") +
  theme_minimal()
```




```r
# Optionally, visualize other factors like household size and years of
residence against affordability
ggplot(CustData, aes(x = Annual.Salary, y = household_size, color =
Affordability)) +
  geom_jitter(width = 0.2, height = 0.2, size = 2) +
  labs(title = "Affordability Based on Annual Salary and Household size",
       x = "Annual salary", y = "Household size ") +
  theme_minimal()
```

Each household size had a minimum of 2 and a maximum 3 people in a house