2. Appendix: PCA & K-Means

Yuting Weng

```
#### Neccessary Packages ####

library(tidyverse)
library(gamlr)
library(glmnet)
library(factoextra)
library(ggplot2)
library(ggfortify)
```

```
#### Import Data ####

setwd("C:/Users/user/Desktop/Big Data/HW/final")
data <- read.csv("Airbnb_Data.csv")

# delete id and text variables
data <- select(data, -c(id, description, name))
```

```
#### Data Preprocessing (by Mengdi) ####

data$room_type <- as.factor(data$room_type)
data$property_type <- as.factor(data$property_type)
data$bed_type <- as.factor(data$bed_type)
data$cancellation_policy <- as.factor(data$cancellation_policy)
data$city <- as.factor(data$city)
data$host_has_profile_pic <- as.factor(data$host_has_profile_pic)
data$host_identity_verified <- as.factor(data$host_identity_verified)
data$host_response_rate <- as.factor(data$host_response_rate)
data$instant_bookable <- as.factor(data$instant_bookable)

# Convert host_response_rate to numeric by removing the '%' sign
data$host_response_rate <- as.numeric(gsub("%", "", data$host_response_rate))
```

```
# handle amenities variable: transform into multiple columns
data$amenities <- str_replace_all(data$amenities, '[{}"]', '')
amenities_list <- str_split(data$amenities, ",")

all_amenities <- unique(unlist(amenities_list))

for (amenity in all_amenities) {
  data[[amenity]] <- sapply(amenities_list, function(x) amenity %in% x)
}

data <- select(data, -amenities)
```

```r
# handle date variables
data$first_review <- as.Date(data$first_review, format="%Y-%m-%d")
data$last_review <- as.Date(data$last_review, format="%Y-%m-%d")
data$host_since <- as.Date(data$host_since, format="%Y-%m-%d")

# transform date variables into more meaningful variables
data$days_since_first_review <- as.numeric(Sys.Date() - data$first_review)
data$days_since_last_review <- as.numeric(Sys.Date() - data$last_review)
data$host_duration <- as.numeric(Sys.Date() - data$host_since)

# delete first_review, last_review, host_since
data <- select(data, -c(first_review, last_review, host_since))
```

```r
# turn thumbnail_url into a binary categorical variable
data$has_thumbnail <- ifelse(is.na(data$thumbnail_url) | data$thumbnail_url == "", FALSE, TRUE)

# delete thumbnail_url
data <- select(data, -thumbnail_url)
```

```r
count_missing <- function(x) {
  sum(is.na(x) | x == "")
}

# Create a summary of missing values (NA and empty strings) for each column
missing_values_summary <- data %>%
  summarise_all(count_missing) %>%
  gather(key = "variable", value = "missing_count") %>%
  arrange(desc(missing_count))

print(missing_values_summary)
```

```r
# Impute the above numerical variables that have missing values with median values
data$host_response_rate[is.na(data$host_response_rate)] <-
  median(data$host_response_rate, na.rm = TRUE)
data$review_scores_rating[is.na(data$review_scores_rating)] <-
  median(data$review_scores_rating, na.rm = TRUE)
data$days_since_first_review[is.na(data$days_since_first_review)] <-
  median(data$days_since_first_review, na.rm = TRUE)
data$days_since_last_review[is.na(data$days_since_last_review)] <-
  median(data$days_since_last_review, na.rm = TRUE)
data$bathrooms[is.na(data$bathrooms)] <-
  median(data$bathrooms, na.rm = TRUE)
data$host_duration[is.na(data$host_duration)] <-
  median(data$host_duration, na.rm = TRUE)
data$beds[is.na(data$beds)] <- median(data$beds, na.rm = TRUE)
data$bedrooms[is.na(data$bedrooms)] <- median(data$bedrooms, na.rm = TRUE)

# Replace missing values with specific values for categorical columns
data$host_has_profile_pic[is.na(data$host_has_profile_pic) |
                            data$host_has_profile_pic == ''] <- 'f'
data$host_identity_verified[is.na(data$host_identity_verified) |
                              data$host_identity_verified == ''] <- 'f'
data$neighbourhood[is.na(data$neighbourhood) |
```

```
                    data$neighbourhood == ''] <- 'Unknown'
data$zipcode[is.na(data$zipcode) | data$zipcode == ''] <- 'Unknown'

# Convert to factors
data$neighbourhood <- as.factor(data$neighbourhood)
data$zipcode <- as.factor(data$zipcode)
```

```
# extract numerical variable names
numeric_vars <- c("log_price", "accommodates", "bathrooms", "host_response_rate",
                  "latitude", "longitude", "number_of_reviews", "review_scores_rating",
                  "bedrooms", "beds", "days_since_first_review", "days_since_last_review",
                  "host_duration")

# extract categorical variable names
categorical_vars <- setdiff(names(data), numeric_vars)

# standardize numerical variables
data_numeric <- scale(data[numeric_vars])
data_numeric <- as.data.frame(data_numeric)

# combine standardized numerical variables with categorical variables
data_scale <- cbind(data_numeric, data[categorical_vars])
```

```
names(data) <- make.names(names(data), unique = TRUE)

print(names(data))
```

```
#### Further Data Cleaning ####

factor_cols <- names(data_scale)[sapply(data_scale, is.factor)]

# Convert factor columns to numeric values
for (col in factor_cols) {
  data[[col]] <- as.numeric(factor(data[[col]])) - 1
}
```

```
# Identify columns with only TRUE and FALSE values
logical_cols <- sapply(data, function(col) is.logical(col) && all(col %in% c(TRUE, FALSE)))

# Convert logical columns to 0s and 1s
logical_col_names <- names(logical_cols)[logical_cols]
data[logical_col_names] <- lapply(data[logical_col_names], as.integer)
```

```
# Convert "True" and "FALSE" strings to logical
data$cleaning_fee <- tolower(data$cleaning_fee) == "true"

# Convert logical values to numeric 0s and 1s
data$cleaning_fee <- as.integer(data$cleaning_fee)
```

```
# Define categories
Essentials = c("Essentials", "Hangers", "Hair.dryer", "Iron",
               "First.aid.kit",  "Safety.card", "Lock.on.bedroom.door",
```

```
               "TV", "Cable.TV", "Bed.linens", "Extra.pillows.and.blankets",
               "Changing.table")
Facilities = c("Air.conditioning", "Heating", "Breakfast", "Pool",
               "Gym", "Hot.tub", "Elevator", "Elevator.in.building",
               "Washer", "Dryer", "Laptop.friendly.workspace")
Parking = c("Free.parking.on.street","Free.parking.on.premises",
            "Paid.parking.off.premises")
Privacy = c("Private.bathroom", "Private.living.room", "Private.entrance")
Family = c("Family.kid.friendly", "Children.s.books.and.toys",
           "Children.s.dinnerware", "Crib", "High.chair", "Stair.gates",
           "Window.guards", "Table.corner.guards", "Baby.monitor",
           "Baby.bath", "Fireplace.guards", "Game.console",
           "Babysitter.recommendations", "Pack..n.Play.travel.crib")
Safety = c("Fire.extinguisher","Smoke.detector","Indoor.fireplace",
           "Carbon.monoxide.detector")
Pets = c("Pets.allowed", "Dog.s.", "Cat.s.", "Other.pet.s.",
         "Pets.live.on.this.property")
Kitchen = c("Kitchen", "Microwave", "Coffee.maker", "Refrigerator",
            "Dishes.and.silverware", "Dishwasher", "Oven", "Stove",
            "Cooking.basics", "Hot.water.kettle")
Internet = c("Internet","Wireless.Internet", "Ethernet.connection",
             "Buzzer.wireless.intercom")
Self_Checkin = c("Self.Check.In", "Lockbox")
Accessibility = c("Wheelchair.accessible", "Wide.clearance.to.bed",
                  "Accessible.height.bed", "Wide.doorway",
                  "Accessible.height.toilet", "Wide.entryway", "Step.free.access",
                  "Ground.floor.access", "Wide.clearance.to.shower...toilet",
                  "Wide.clearance.to.shower.and.toilet", "Wide.hallway.clearance",
                  "Flat.smooth.pathway.to.front.door",
                  "X.smooth.pathway.to.front.door", "Well.lit.path.to.entrance")
Events = c("Suitable.for.events", "Doorman", "Doorman.Entry")
Others = c("Other", "translation.missing..en.hosting_amenity_49", "V106",
           "Single.level.home","Flat")
Bathroom = c("Bathtub", "Hot.water", "Shampoo", "Bathtub.with.shower.chair",
             "Handheld.shower.head", "Grab.rails.for.shower.and.toilet",
             "Body.soap", "Hand.soap", "Bath.towel", "Hand.or.paper.towel",
             "Toilet.paper")
Outdoor = c("Garden.or.backyard", "Patio.or.balcony", "BBQ.grill",
            "Lake.access", "Beachfront", "Beach.essentials", "Ski.in.Ski.out",
            "Path.to.entrance.lit.at.night", "Waterfront")
Smoking = c("Smoking.allowed")
Miscellaneous = c("Luggage.dropoff.allowed","Outlet.covers",
                  "Long.term.stays.allowed", "Firm.mattress", "Pocket.wifi",
                  "Cleaning.before.checkout", "EV.charger", "Keypad",
                  "Smart.lock")


data_new <- data %>%
  mutate(
    Essentials = ifelse(rowSums(select(., all_of(Essentials))) > 0, 1, 0),
    Facilities = ifelse(rowSums(select(., all_of(Facilities))) > 0, 1, 0),
    Parking = ifelse(rowSums(select(., all_of(Parking))) > 0, 1, 0),
    Privacy = ifelse(rowSums(select(., all_of(Privacy))) > 0, 1, 0),
```

```
    Family = ifelse(rowSums(select(., all_of(Family))) > 0, 1, 0),
    Safety = ifelse(rowSums(select(., all_of(Safety))) > 0, 1, 0),
    Pets = ifelse(rowSums(select(., all_of(Pets))) > 0, 1, 0),
    Kitchen = ifelse(rowSums(select(., all_of(Kitchen))) > 0, 1, 0),
    Internet = ifelse(rowSums(select(., all_of(Internet))) > 0, 1, 0),
    Self_Checkin = ifelse(rowSums(select(., all_of(Self_Checkin))) > 0, 1, 0),
    Accessibility = ifelse(rowSums(select(., all_of(Accessibility))) > 0, 1, 0),
    Events = ifelse(rowSums(select(., all_of(Events))) > 0, 1, 0),
    Others = ifelse(rowSums(select(., all_of(Others))) > 0, 1, 0),
    Bathroom = ifelse(rowSums(select(., all_of(Bathroom))) > 0, 1, 0),
    Outdoor = ifelse(rowSums(select(., all_of(Outdoor))) > 0, 1, 0),
    Smoking = ifelse(rowSums(select(., all_of(Smoking))) > 0, 1, 0),
    Miscellaneous = ifelse(rowSums(select(., all_of(Miscellaneous))) > 0, 1, 0)
  )
```

```
# Define columns to keep
columns_to_keep <- c("log_price", "property_type", "room_type", "accommodates",
                     "bathrooms", "bed_type", "cancellation_policy",
                     "cleaning_fee", "city", "host_has_profile_pic",
                     "host_identity_verified", "host_response_rate",
                     "instant_bookable", "latitude", "longitude", "neighbourhood",
                     "number_of_reviews", "review_scores_rating",
                     "days_since_first_review", "days_since_last_review",
                     "zipcode", "bedrooms", "beds", "Essentials", "Facilities",
                     "Parking", "Privacy", "Family", "Safety", "Pets", "Kitchen",
                     "Internet", "Self_Checkin",  "Accessibility", "Events",
                     "Others", "Bathroom", "Outdoor", "Smoking", "Miscellaneous")

# Create new data frame with only columns to keep
data_new <- data_new %>%
  select(all_of(columns_to_keep))
```

```
#### Principal Component Analysis (PCA)####
#### PCA with only numerical values ####

# Remove the 'log_price' column from data_numeric
data_without_y <- data[numeric_vars][, setdiff(names(data[numeric_vars]), c("log_price"))]

# Perform PCA
pca_result <- prcomp(data_without_y, scale. = TRUE)

# Summary of PCA
summary(pca_result)
```

The summary of PCA shows the standard deviation, proportion of variance, and cumulative proportion explained by each principal component.

```
# The scree plot
plot(pca_result, xlab = "Principal Component", col = "#ff4e4e")
```

```
# Extract the scores of the observations along the principal components
scores <- pca_result$x
```

```r
# Plot the scores of the observations on the first two principal components
plot(scores[,1], scores[,2],
     xlab = "Principal Component 1", ylab = "Principal Component 2",
     main = "Scatter Plot of Principal Components", col="#ff4e4e")
```

```r
##### Selection of the Number of Principal Components:####

# Extract the first K principal components
pca_data <-predict(pca_result)
pca_df <- as.data.frame(pca_data)

log_price <- data_numeric$log_price

kfits <- lapply(1:12, function(K) glm(log_price~., data = pca_df[, 1:K, drop=FALSE]))

aicc <- sapply(kfits, AICc)
plot(aicc, col = "#ff4e4e", main = "AICc Plot for Model Selection")
```

```r
which.min(aicc)
```

```r
##### Generalized Linear Model (GLM):####

# GLM on First K Technique
log_price <- data_numeric$log_price
glm <- glm(log_price ~ ., data = pca_df, family = gaussian)
summary(glm)
```

```r
##### Lasso Technique:####

# Lasso Technique
lasso_model <- cv.glmnet(x=pca_data, y=log_price, nfold=20)

coef(lasso_model)
```

```r
#### PCA with all columns (transforming catgorical columns into dummy)####

# Scale the Data
data_scale2 <- scale(data_new)
data_scale2 <- as.data.frame(data_scale2)
```

```r
# Remove the 'log_price' column
data_without_y <- data_scale2[, !names(data_scale2) %in% "log_price"]

# Perform PCA
pca_result <- prcomp(data_without_y, scale. = FALSE)

# Summary of PCA
summary(pca_result)
```

```r
# The scree plot
plot(pca_result, xlab = "Principal Component", col="#ff4e4e")
```

```r
# Extract the scores of the observations along the principal components
scores <- pca_result$x

# Plot the scores of the observations on the first two principal components
plot(scores[,1], scores[,2],
     xlab = "Principal Component 1", ylab = "Principal Component 2",
     main = "Scatter Plot of Principal Components", col="#ff4e4e")
```

```r
# Extract the first K principal components
pca_data <-predict(pca_result)
pca_df <- as.data.frame(pca_data)

log_price <- data_numeric$log_price

kfits <- lapply(1:39, function(K) glm(log_price~., data = pca_df[, 1:K, drop=FALSE]))

aicc <- sapply(kfits, AICc)
# Find the index of the minimum AICc value
min_index <- which.min(aicc)

# Plot the AICc values and add a vertical line at the minimum
plot(aicc, col = "#ff4e4e", main = "AICc Plot for Model Selection",
     xlab = "Number of Principal Components")
abline(v = min_index, col = "blue", lty = 2)

# Annotate the plot with the number of the principal component
text(min_index, aicc[min_index], labels = paste("Min:", min_index), pos = 3, col = "blue")
```

```r
which.min(aicc)
```

```r
##### Using GLM #####

# GLM on First K Technique
log_price <- data_scale2$log_price
glm <- glm(log_price ~ ., data = pca_df[1:37], family = gaussian)
summary(glm)
```

```r
# Extract coefficients and their p-values
coefficients <- coef(glm)
p_values <- summary(glm)$coefficients[, "Pr(>|t|)"]

# Filter statistically significant coefficients
significant_coefficients <- coefficients[p_values < 0.05]

# Extract names of significant coefficients
significant_coefficient_names <- names(significant_coefficients)

# Count the number of significant coefficients (p-value < 0.05)
num_significant <- sum(p_values < 0.05)
```

```r
# Print the number of significant coefficients
print(significant_coefficient_names)
print(num_significant)
```

```r
##### Using Lasso Regression #####

# Lasso Technique
lasso_model <- cv.glmnet(x=pca_data[, 1:37], y=log_price, nfold=20)

coef(lasso_model)
```

```r
# Extract coefficients
coefficients <- coef(lasso_model)

# Extract variable names
variable_names <- rownames(coefficients)[-1]  # Exclude the intercept term

# Find significant coefficients
significant_indices <- which(coefficients[-1, ] != 0)

# Print names of significant coefficients
significant_variable_names <- variable_names[significant_indices]
print(significant_variable_names)


# Number of significant coefficients
num_significant_coefficients <- sum(coefficients != 0)

# Print number of significant coefficients
print(num_significant_coefficients)
```

```r
# Predict using the Lasso model
lasso_predictions <- predict(lasso_model, newx = pca_data[, 1:37])

# Calculate mean squared error (MSE)
mse <- mean((lasso_predictions - log_price)^2)

# Calculate mean absolute error (MAE)
mae <- mean(abs(lasso_predictions - log_price))

# Calculate R-squared (R2)
actual_mean <- mean(log_price)
ss_total <- sum((log_price - actual_mean)^2)
ss_residual <- sum((log_price - lasso_predictions)^2)
r_squared <- 1 - (ss_residual / ss_total)

# Print the evaluation metrics
print(paste("Mean Squared Error (MSE):", mse))
print(paste("Mean Absolute Error (MAE):", mae))
print(paste("R-squared (R2):", r_squared))
```

```r
#### Kmeans Cluster ####

data_without_price <- data_scale2[, setdiff(names(data_scale2), c("log_price"))]
```

```r
#### 1. Elbow Method (Within-Cluster Sum of Squares): ####

set.seed(123)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(data_without_price, k, nstart = 10 )$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)

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

```r
#### 2. Silhouette Method:####

# Assuming your data is stored in a data frame called "your_data"
sampled_data <- data_without_price[sample(nrow(data_without_price), 25000), ]

# Finding the Optimal Number of Clusters with the Silhouette Method
fviz_nbclust(sampled_data, kmeans, method = "silhouette")
```

```r
# Plotting K2 to K5
k2 <- kmeans(data_without_price, centers = 2, nstart = 25)
k3 <- kmeans(data_without_price, centers = 3, nstart = 25)
k4 <- kmeans(data_without_price, centers = 4, nstart = 25)
k5 <- kmeans(data_without_price, centers = 5, nstart = 25)

# plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = data_without_price) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = data_without_price) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = data_without_price) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = data_without_price) + ggtitle("k = 5")

library(gridExtra)
grid.arrange(p1, p2, p3, p4, nrow = 2)
```

```r
#### K=2 ####

k2$centers
```

```r
# Size of each Cluster
k2$size
```

```r
# Data frame containing cluster centers
cluster_centers <- data.frame(
  cluster = c("Cluster 1", "Cluster 2"),  # Cluster labels
  variable = colnames(k2$centers),     # Variable names
  value = c(k2$centers[1, ], k2$centers[2, ])  # Average values for each variable in each cluster
)

# Bar plot
bar_plot <- ggplot(cluster_centers, aes(x = variable, y = value, fill = cluster)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.5) +
  labs(x = "Variable", y = "Average Value", title = "Cluster Profiles") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Radar chart
radar_plot <- ggplot(cluster_centers, aes(x = variable, y = value,
                                          color = cluster, group = cluster)) +
  geom_line() +
  geom_point(size = 2) +
  labs(x = NULL, y = NULL, title = "Cluster Profiles") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display plots
print(bar_plot)
print(radar_plot)
```

```r
# Top 10 Variables that Most Distinguish the Two Clusters
print(apply(k2$centers,1,function(c) colnames(data_without_price)[order(-c)[1:10]]))
```

```r
# Assign clusters to the original data
clusters_k2 <- data_scale2
clusters_k2$cluster <- k2$cluster

# Calculate summary statistics of log_price for each cluster
cluster_summary_k2 <- clusters_k2 %>%
  group_by(cluster) %>%
  summarize(
    mean_log_price = mean(log_price),
    median_log_price = median(log_price),
    sd_log_price = sd(log_price)
  )
print(cluster_summary_k2)
```

```r
# Visualize the log_price distribution for each cluster with different colors
ggplot(clusters_k2, aes(x = factor(cluster), y = log_price, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.1), size = 3) +
  labs(title = "Log Price Distribution Across Clusters",
```

```
       x = "Cluster",
       y = "Log Price") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
# Visualize the Review Scores Rating distribution for each cluster with different colors
ggplot(clusters_k2, aes(x = factor(cluster), y = review_scores_rating, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.1), size = 3) +
  labs(title = "Review Scores Rating Distribution Across Clusters",
       x = "Cluster",
       y = "Review Scores Rating") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
# Visualize the Property Type distribution for each cluster with different colors
ggplot(clusters_k2, aes(x = factor(cluster), y = property_type, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.1), size = 3) +
  labs(title = "Property Type Distribution Across Clusters",
       x = "Cluster",
       y = "Property Type") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
#### 3. Information Criteria (AIC/BIC): ####

kfit <- lapply(1*(1:10), function(k) kmeans(data_without_price,k))
```

```
source("kIC.R")

kaicc <- sapply(kfit,kIC)
kbic <- sapply(kfit,kIC,"B")
```

```
k_values <- c(1,2,3,4,5,6,7,8,9,10)
optimal_k_aicc <- k_values[which.min(kaicc)]
optimal_k_bic <- k_values[which.min(kbic)]
optimal_k_bic
```

```
kmfs <- kfit[[which(k_values==optimal_k_bic)]]
print(apply(kmfs$centers,1,function(c) colnames(data_without_price)[order(-c)[1:5]]))
```

```
## Size of Each Cluster
kmfs$size
```

```
# Calculate summary statistics of log_price for each cluster
cluster_summary_k10 <-data_with_clusters %>%
  group_by(cluster) %>%
  summarize(
```

```
    mean_log_price = mean(log_price),
    median_log_price = median(log_price),
    sd_log_price = sd(log_price)
  )
print(cluster_summary_k10)
```

```
# Visualize the log_price distribution for each cluster with different colors
ggplot(data_with_clusters, aes(x = factor(cluster), y = log_price, fill = factor(cluster))) +
  geom_boxplot() +
  stat_summary(geom = "text", fun = quantile, aes(label = sprintf("%1.2f", ..y..)),
               position = position_nudge(x = 0.5), size = 2.5) +
  labs(title = "Log Price Distribution Across Clusters",
       x = "Cluster",
       y = "Log Price") +
  scale_fill_discrete(name = "Cluster") +
  theme_minimal()
```

```
k10 <- kmeans(data_without_price, centers = 10, nstart = 25)
autoplot(k10, data_without_price, frame = TRUE)
```