Clustering Airbnb listings in Naples

Introduction

This section outlines the process of cleaning and transforming Airbnb data. The dataset contains various features like price, room type, amenities, and others. We will prepare this data for further analysis and cluster analysis.

Loading Required Libraries

We start by loading the necessary libraries.

```
library(tidyverse, quietly = TRUE)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(mgsub)
library(stringr)

# URL of the compressed CSV file
url <- "http://data.insideairbnb.com/italy/campania/naples/2023-06-21/data/listings.csv.gz"

# Temporary file to store the downloaded data
temp_file <- tempfile(fileext = ".csv.gz")

# Download the file
download.file(url, temp_file)

# Read the compressed CSV file into a data frame
airbnb_data <- read_csv(temp_file)
```

```
## Rows: 8973 Columns: 75
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (25): listing_url, source, name, description, neighborhood_overview, pi...
## dbl  (37): id, scrape_id, host_id, host_listings_count, host_total_listings_...
## lgl   (8): host_is_superhost, host_has_profile_pic, host_identity_verified, ...
## date  (5): last_scraped, host_since, calendar_last_scraped, first_review, la...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Clean up
unlink(temp_file)
```

Data Cleaning

We remove rows where 'price' or 'review_scores_rating' is missing.

```
airbnb_data <- drop_na(airbnb_data, price, review_scores_rating)
```

The `gsub` function replaces all occurrences of dollar signs ($) and commas (,) in the price column with an empty string. This is needed because these characters are not valid in a numerical context.

`as.numeric(...)`: The resulting string is then converted to a numeric data type.

`mutate(airbnb_data, price = ...)`: The mutate function from the dplyr package replaces the original price column with the newly transformed numeric column.

```
airbnb_data <- mutate(airbnb_data, price = as.numeric(gsub("[$,]", "", price)))
```

We need to ensure that the date columns are in the correct format for further analysis.

```
# Convert date columns to Date type
airbnb_data <- mutate(airbnb_data, last_review = as.Date(last_review, format="%Y-%m-%d"))
```

We transform the room type into a categorical variable and remove any special characters from the listing names.

```
# Convert 'room_type' to factor, remove special characters from 'name'
airbnb_data <- mutate(airbnb_data, room_type = as.factor(room_type),
                      name = gsub("[^[:alnum:][:space:]]", "", name))
```

Here, we calculate the price per guest and filter out listings with extreme values.

```
# Compute price per guest and filter out extreme values
airbnb_data <- mutate(airbnb_data, price_per_guest = price / accommodates) %>%
  filter(price_per_guest <= 1000)
```

We extract and convert the bathroom details, including whether the bathroom is shared or private.

```
# Extract and convert bathroom details
airbnb_data <- airbnb_data %>%
  mutate(
    bathrooms = as.numeric(str_extract(bathrooms_text, "\\d+")),
    bathrooms = if_else(
      str_detect(bathrooms_text, "shared"),
      bathrooms / 2,
      bathrooms
    ),
    bathrooms = if_else(
      str_detect(bathrooms_text, "private"),
      bathrooms,
      bathrooms
    )
  )
```

We convert percentage strings to numeric values for easier analysis.

```
# Define a function to safely convert the percentage string to numeric
convert_to_numeric <- function(x) {
```

```
  num <- as.numeric(str_replace(x, "%", ""))
  ifelse(is.na(num), NA, num / 100)
}
```

We convert host acceptance and response rates to a numerical format.

```
# Use the function to safely convert the columns
airbnb_data <- airbnb_data %>%
  mutate(
    host_acceptance_rate = suppressWarnings(convert_to_numeric(host_acceptance_rate)),
    host_response_rate = suppressWarnings(convert_to_numeric(host_response_rate))
  )
```

We compute the total number of amenities available for each listing and add this information as a new column.

```
# Compute and add the number of amenities as a new column
airbnb_data <- mutate(airbnb_data, num_amenities = str_count(amenities, ",") + 1)
```

We convert certain date-related columns to Date type and calculate how long the host has been active, the time since the first and last review.

```
# Convert 'host_since', 'first_review', and 'last_review' to Date and calculate durations
current_date <- as.Date("2023-06-21")
airbnb_data <- mutate(airbnb_data, host_since = as.Date(host_since),
                      first_review = as.Date(first_review),
                      last_review = as.Date(last_review),
                      host_length = as.integer(current_date - host_since),
                      since_first_review = as.integer(current_date - first_review),
                      since_last_review = as.integer(current_date - last_review))
```

We create a binary variable to indicate whether a host has multiple listings or not.

```
# Add 'multiple_host_listings' based on the condition
airbnb_data <- mutate(airbnb_data, multiple_host_listings = if_else(calculated_host_listings_count > 1,
```

We create new variables for each distinct amenity and host verification. These variables will be binary (Yes/No) and will help in understanding the importance of each feature for the listings.

We'll be using `mgsub` for multiple string replacements and `stringr` for string manipulations.

First, we clean up the amenities and host verifications strings to make them easier to handle.

```
amenities <- as.data.frame(mgsub(airbnb_data$amenities, c("\\{", "\\}", "\\,", "\\'", "\\[", "\\]", "\"
host_verifications <- as.data.frame(mgsub(airbnb_data$host_verifications, c("\\[", "\\]", "\\,", "\\'")
```

We initialize lists to store the amenities and host verifications for each listing.

```
output_list_verifications <- list()
output_list_verifications_count <- list()
output_list_amenities <- list()
output_list_amenities_count <- list()
```

We loop through each listing to populate the lists initialized above.

```
for(i in 1:nrow(host_verifications)){
  output_list_verifications[i] <- str_split(host_verifications[i, ], " ")[1]
  output_list_verifications_count[i] <- length(output_list_verifications[[i]])
  output_list_amenities[i] <- str_split(amenities[i, ], "  ")[1]
```

```
  output_list_amenities_count[i] <- length(output_list_amenities[[i]])
}
```

Select the most frequent host verifications to use as column names

```
dummy_host_verifications_cols <- output_list_verifications[[which.max(output_list_verifications_count)]]
```

Create an empty data frame with the same number of rows as airbnb_data and columns as dummy_host_verifications_cols

```
dummy_host_verifications_df <- matrix(nrow = nrow(airbnb_data), ncol = length(dummy_host_verifications_c
colnames(dummy_host_verifications_df) <- dummy_host_verifications_cols
```

Select the most frequent amenities to use as column names

```
dummy_amenities_cols <- output_list_amenities[[which.max(output_list_amenities_count)]]
```

Create an empty data frame with the same number of rows as airbnb_data and columns as

```
dummy_amenities_cols
dummy_amenities_df <- matrix(nrow = nrow(airbnb_data), ncol = length(dummy_amenities_cols)) %>% as.data
colnames(dummy_amenities_df) <- dummy_amenities_cols
```

Combine the original airbnb_data with the dummy data frames

```
airbnb_data <- cbind(airbnb_data, dummy_host_verifications_df, dummy_amenities_df)
```

Populate the host verifications dummy columns

```
for(l in 1:length(output_list_verifications)){
  for(j in 82:84){
    if(colnames(airbnb_data)[j] %in% output_list_verifications[[l]]){
      airbnb_data[l, j] <- "Yes"
    } else{
      airbnb_data[l, j] <- "No"
    }
  }
}
```

Populate the amenities dummy columns

```
for(l in 1:length(output_list_amenities)){
  for(j in 85:ncol(airbnb_data)){
    if(colnames(airbnb_data)[j] %in% output_list_amenities[[l]]){
      airbnb_data[l, j] <- "Yes"
    } else{
      airbnb_data[l, j] <- "No"
    }
  }
}
```

Fill NA values in the new dummy columns with “No”

```
for(j in 82:ncol(airbnb_data)){
  for(i in 1:nrow(airbnb_data)){
    if(is.na(airbnb_data[i, j]) == TRUE){
      airbnb_data[i, j] <- "No"
    }
  }
  # Convert the new dummy columns to factors
```

```
  airbnb_data[, j] <- factor(airbnb_data[, j], levels = c("Yes", "No"))
}
```

Define lists of columns to be converted to factors and numerics

```
factor_cols <- c("instant_bookable", "host_has_profile_pic", "host_identity_verified",
                 "host_is_superhost", "room_type", "property_type", "host_response_time",
                 "host_listings_count", "host_total_listings_count", "neighbourhood_cleansed",
                 "has_availability")

numeric_cols <- c("accommodates", "host_listings_count", "host_total_listings_count", "beds",
                  "minimum_nights", "maximum_nights", "minimum_minimum_nights", "maximum_minimum_nights
                  "minimum_maximum_nights", "maximum_maximum_nights", "host_length", "since_first_revie
                  "since_last_review", "number_of_reviews", "availability_365", "availability_90",
                  "availability_60", "availability_30", "calculated_host_listings_count")

# Apply the transformations
airbnb_data <- airbnb_data %>%
  mutate(
    across(all_of(factor_cols), as.factor),
    across(all_of(numeric_cols), as.numeric)
  )
```

Remove unnecessary columns

```
airbnb_data <- airbnb_data |>
  select(-c(id, host_verifications, neighbourhood_group_cleansed, first_review, amenities, number_of_rev

# Remove rows with missing values
airbnb_data <- airbnb_data[complete.cases(airbnb_data),]
```

```
# Identify numeric and factor variables
numeric_vars <- names(airbnb_data)[sapply(airbnb_data, class) == 'numeric']
factor_vars <- names(airbnb_data)[sapply(airbnb_data, class) == 'factor']

# Create new data frames containing only numeric and factor variables
df_numeric <- airbnb_data[, numeric_vars]
df_factor <- airbnb_data[, factor_vars]

# Remove specific numeric columns that are not needed
df_numeric <- df_numeric |>
  select(-c("minimum_minimum_nights","maximum_minimum_nights",
            "minimum_maximum_nights", "maximum_maximum_nights",
            "minimum_nights_avg_ntm","maximum_nights_avg_ntm" ))

# Combine the numeric and factor data frames to create the final airbnb_data (4238 x 135)
airbnb_data <- cbind(df_numeric, df_factor)
```

Calculate average nightly price per guest per neighborhood

```
library(geojsonio)
```

```
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
```

```
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)

## Registered S3 method overwritten by 'geojsonsf':
##   method        from
##   print.geojson geojson

##
## Attaching package: 'geojsonio'

## The following object is masked from 'package:base':
##
##     pretty
```

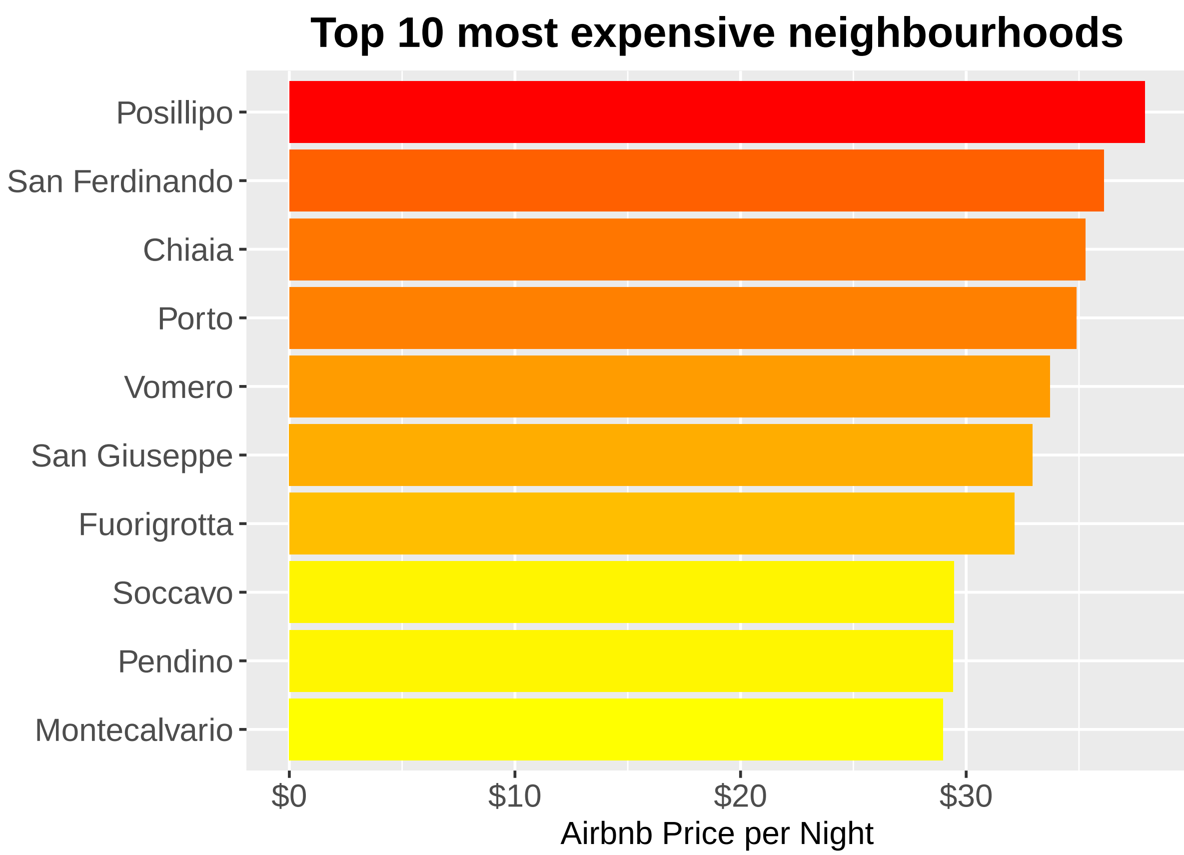
```r
library(leaflet)

nb_geo <- geojson_read('http://data.insideairbnb.com/italy/campania/naples/2023-06-21/visualisations/ne

nb_geo@data$neighbourhood = as.factor(nb_geo@data$neighbourhood)

night_neighbourhood <- airbnb_data %>% group_by(neighbourhood_cleansed) %>% summarize(avg_night_price =

ggplot(night_neighbourhood[1:10, ], aes(x = reorder(neighbourhood_cleansed, avg_night_price),  y = avg_
                                        fill = avg_night_price)) +
  geom_bar(stat = "identity") + ggtitle("Top 10 most expensive neighbourhoods") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        axis.title = element_text(hjust = 0.5, size = 12),
        axis.text = element_text(size = 12)) + scale_y_continuous(labels = function(x) paste0("$", x)) +
  xlab("") + ylab("Airbnb Price per Night") + coord_flip() + theme(legend.position = "none") + scale_fi
```




```
colnames(night_neighbourhood)[1] <- "neighbourhood"
nb_geo@data <- left_join(nb_geo@data, night_neighbourhood[, 1:2]) %>% as.data.frame()
```

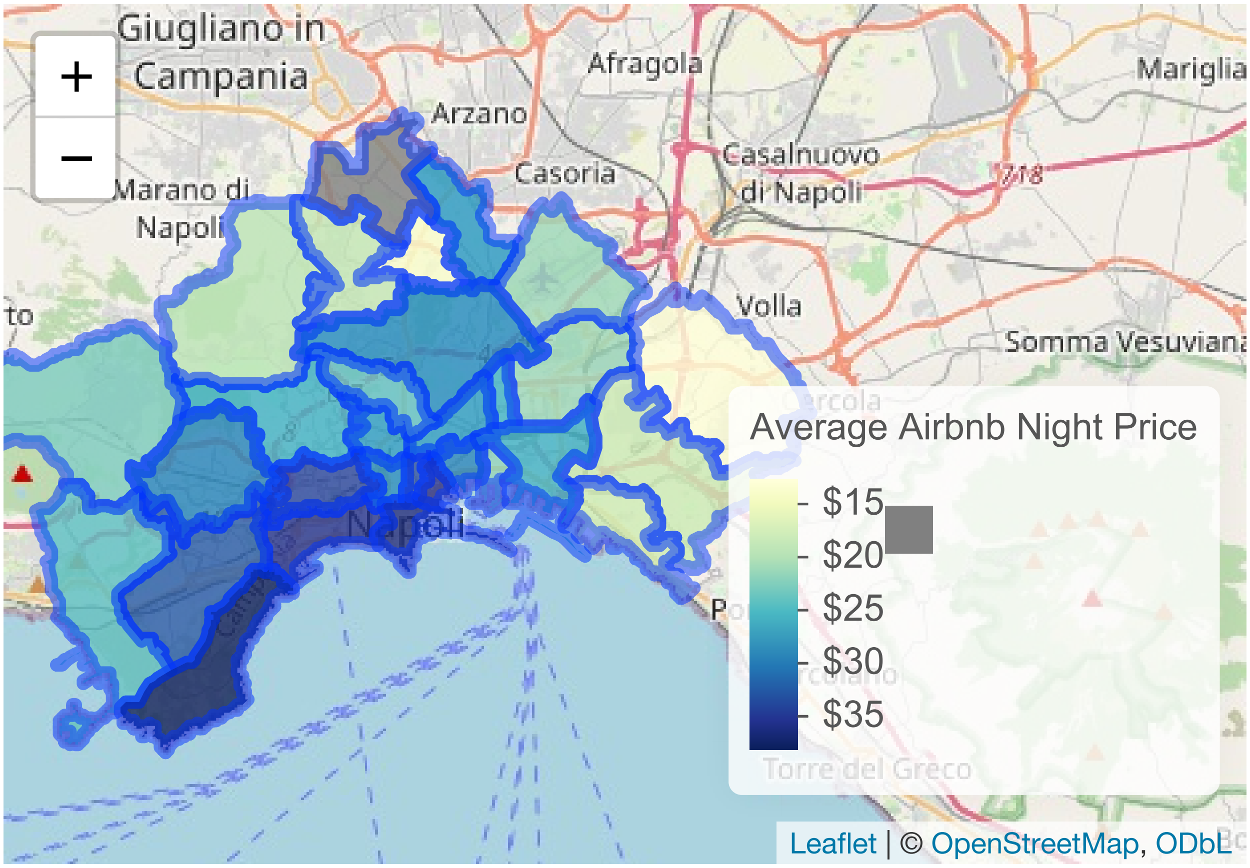
```
## Joining with `by = join_by(neighbourhood)`
```

```
for(i in 1:nrow(nb_geo@data)){
  if(is.na(nb_geo@data[i, "avg_night_price"]) == TRUE){
    nb_geo@data[i, 3] <- mean(nb_geo@data[complete.cases(nb_geo@data), 3])
  } else if(is.na(nb_geo@data[i, "avg_night_price"]) == FALSE){
  }
}


pal <- colorNumeric(
  palette = "YlGnBu",
  domain = nb_geo@data$avg_night_price
)

price_per_neighbourhood <- leaflet(nb_geo) %>%
  addTiles() %>% setView(lng = 14.305573, lat = 40.853294, zoom = 10.5) %>%
  addPolygons(stroke = TRUE, fillColor = ~ pal(avg_night_price), fillOpacity = 0.8,
              highlight = highlightOptions(weight = 2,
                                           color = ~ pal(avg_night_price),
                                           fillOpacity = 1,
                                           bringToFront = TRUE),
              label = ~neighbourhood,
              smoothFactor = 0.2,
              popup = ~ paste(paste(neighbourhood,":"), "<br/>","<b/>", paste("Avg Night Price: ", "$",
```

```
  addLegend("bottomright", pal = pal, values = ~avg_night_price, opacity = 1.0,
            title = "Average Airbnb Night Price",
            labFormat = labelFormat(prefix = "$"), na.label="")
price_per_neighbourhood
```



Clustering

Calculate Gower distance and apply PAM (2 to 10 clusters, returns cluster with best ASW criterion value)

```
library(cluster)
library(fpc)
gower_dist <- daisy(airbnb_data, metric = "gower")
```

```
## Warning in daisy(airbnb_data, metric = "gower"): binary variable(s) 32 treated
## as interval scaled
```

```
pam_fit <- pamk(gower_dist, krange = 2:10, criterion = "asw")
```

Apply K-Prototypes (2 to 6 clusters, 20 reps)

```
library(clustMixType)

kpres2 <- kproto(cbind(df_numeric, df_factor), 2, nstart = 1,verbose = FALSE)
kpres3 <- kproto(cbind(df_numeric, df_factor), 3, nstart = 1,verbose = FALSE)
kpres4 <- kproto(cbind(df_numeric, df_factor), 4, nstart = 1,verbose = FALSE)
kpres5 <- kproto(cbind(df_numeric, df_factor), 5, nstart = 1,verbose = FALSE)
kpres6 <- kproto(cbind(df_numeric, df_factor), 6, nstart = 1,verbose = FALSE)
```

```r
print(table(pam_fit$pamobject$clustering, kpres2$cluster))

require(mclust)
```

```
## Loading required package: mclust

## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.

##
## Attaching package: 'mclust'

## The following object is masked from 'package:purrr':
##
##     map
```

```r
adjustedRandIndex(pam_fit$pamobject$clustering, kpres2$cluster)
```

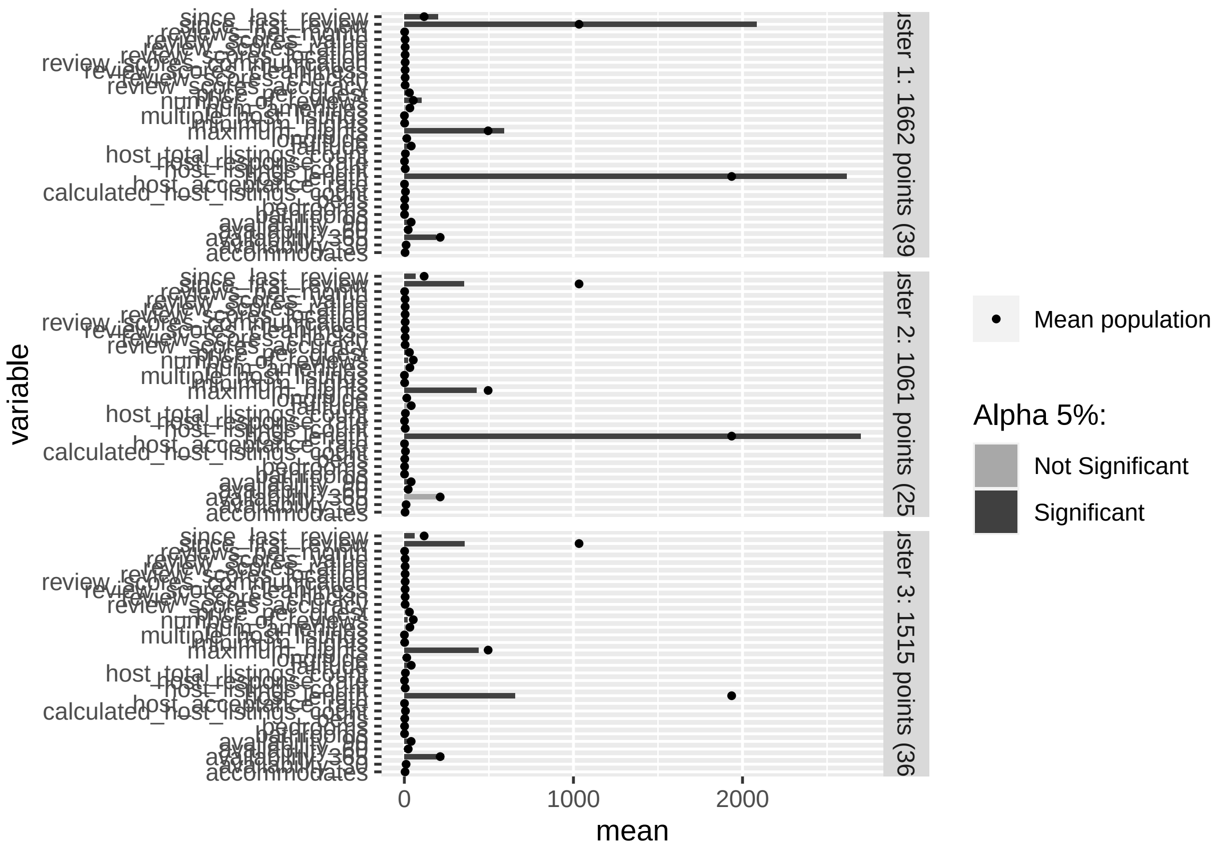
Calculate Total Variation Distance and apply PAM (2 to 10 clusters, returns cluster with best criterion value)

```r
library(kmed)
airbnb_dist <- distmix(airbnb_data, method = "ahmad", idnum = 1:32, idcat = 33:103)

#run the sfkm algorihtm on airbnb_dist
(simplekm <- skm(airbnb_dist, ncluster = 3, seeding = 50))

#calculate silhouette of the K-medoids result of airbnb data set
silairbnb <- sil(airbnb_dist, simplekm$medoid, simplekm$cluster, title = "Silhouette plot of Airbnb data

barplotnum(df_numeric, simplekm$cluster, alpha = 0.05)
```




```
adjustedRandIndex(simplekm$cluster, kpres3$cluster)
```

We use the leaflet package to display listings from the two groups using lat/long information. This will give us an idea of geographical distribution of the clusters.

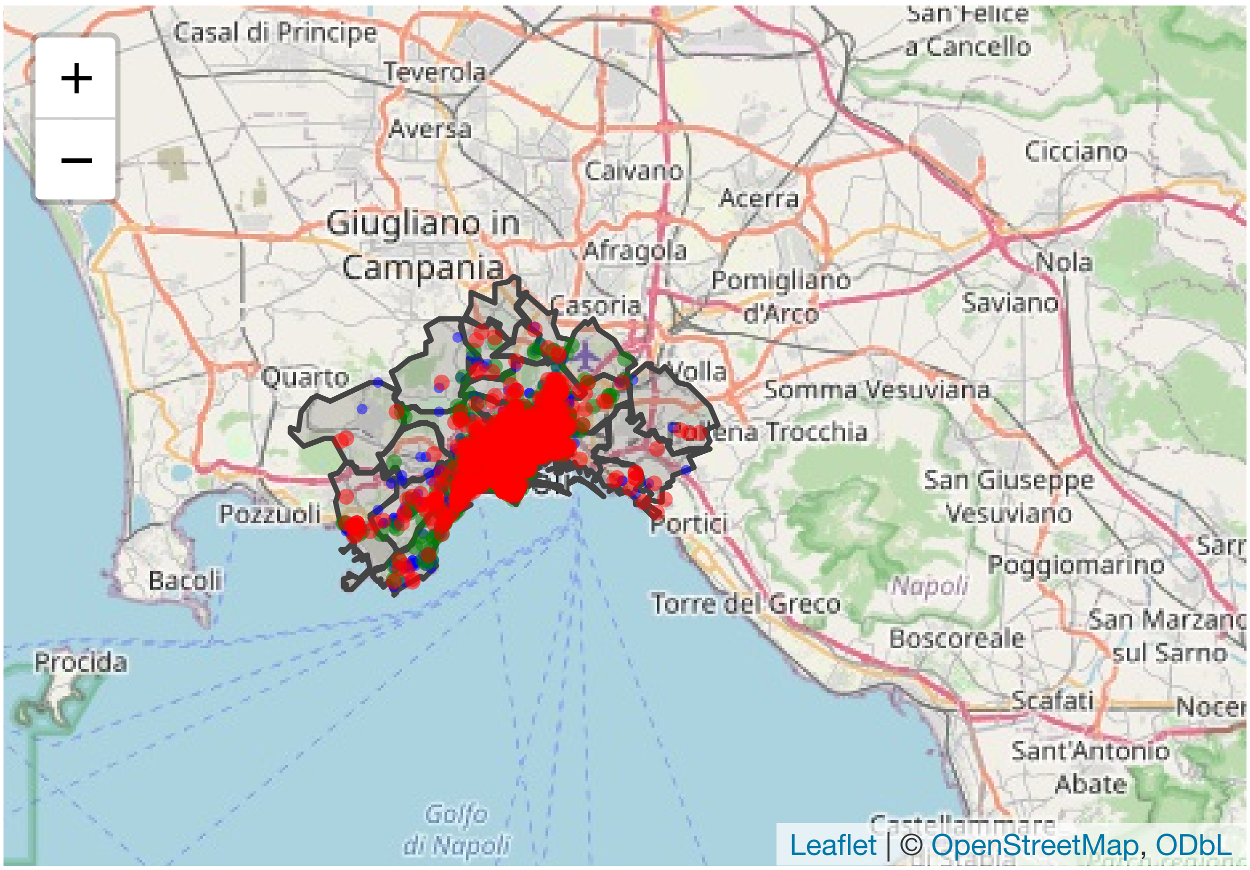
```
c1 <- cbind(df_numeric, df_factor) %>%
  filter(simplekm$cluster == 1)

c2 <- cbind(df_numeric, df_factor) %>%
  filter(simplekm$cluster == 2)

c3 <- cbind(df_numeric, df_factor) %>%
  filter(simplekm$cluster == 3)

library(leaflet)
leaflet() %>% setView(lng = 14.305573, lat = 40.853294, zoom = 10) %>%
  addTiles() %>%
  addPolygons(data = nb_geo, color = "#444444", weight = 2, opacity = 1) %>%
  addCircleMarkers(  lng = c1$longitude,
                     lat = c1$latitude,
                     radius = 2,
                     stroke = FALSE,
                     color = "blue",
                     fillOpacity = 0.5,
                     group = "c1"
  ) %>%
  addCircleMarkers(  lng = c2$longitude,
                     lat = c2$latitude,
```

```
                    radius = 3,
                    stroke = FALSE,
                    color = "green",
                    fillOpacity = 0.5,
                    group = "c2"
)%>%
addCircleMarkers(  lng = c3$longitude,
                    lat = c3$latitude,
                    radius = 3,
                    stroke = FALSE,
                    color = "red",
                    fillOpacity = 0.5,
                    group = "c3"
)
```



Describe the clusters

```
library(FactoMineR)

desc_clus <- catdes(cbind(airbnb_data,as.factor(simplekm$cluster)),136)
```

```
head(desc_clus$quanti$`1`)
```

```
##                          v.test Mean in category Overall mean sd in category
## since_first_review    56.895685      2084.287605  1033.249882    625.3287917
```

```
## number_of_reviews     33.200953       102.729242    52.901840    101.0716232
## host_length           30.833476      2615.486161  1935.814299    759.1718861
## since_last_review     14.270478       200.267750   117.035394    448.5341414
## maximum_nights        10.230976       591.163057   495.470505    537.9548789
## review_scores_checkin  6.824372         4.870271     4.826005      0.1788259
##                        Overall sd       p.value
## since_first_review     965.852446  0.000000e+00
## number_of_reviews       78.467442 1.042979e-241
## host_length           1152.520377 9.329814e-209
## since_last_review      304.947772  3.342773e-46
## maximum_nights         489.027031  1.440596e-24
## review_scores_checkin    0.339137  8.831073e-12
```

```
head(desc_clus$category$`1`)
```

```
##                                Cla/Mod  Mod/Cla   Global       p.value
## .Heating=Yes                  55.72770 71.41998 50.25956 4.316730e-111
## .Bidet=No                     55.24674 58.60409 41.59981  7.506912e-73
## .Dining.table=No              52.64188 64.74128 48.23030  1.404199e-67
## .Crib=Yes                     64.86797 33.99519 20.55215  1.942340e-66
## .Hangers=Yes                  44.42558 94.46450 83.38839  1.083563e-61
## property_type=Entire rental unit 50.71199 66.42599 51.36857  1.271447e-56
##                                 v.test
## .Heating=Yes                  22.39588
## .Bidet=No                     18.05274
## .Dining.table=No              17.36951
## .Crib=Yes                     17.21811
## .Hangers=Yes                  16.57349
## property_type=Entire rental unit 15.85631
```

Mod/Cla = Within cluster Cla/Mod = Across cluster

For the Heating = "Yes":

55.7% of the listings who have Heating belong to Cluster 1

71.4% of the listings who belong to Cluster 1 have Heating

50.2% of the whole population has Heating

The category "Heating=Yes" is over represented (v-test > 0) among listings in the first (blue) cluster whereas "calculated_host_listings_count" is under represented (v-test < 0).

```
head(desc_clus$quanti$`2`)
```

```
##                               v.test Mean in category Overall mean
## host_length                24.896104     2698.5984920 1935.8142992
## host_total_listings_count 10.404123        9.0499529    6.1300142
## host_listings_count        9.863029        7.9340245    5.3688060
## multiple_host_listings     6.754528        0.6013195    0.5115621
## availability_90            6.101806       44.8473139   40.4263804
## availability_60            5.762648       25.7822809   22.9738084
##                           sd in category   Overall sd       p.value
## host_length                  627.0084359 1152.5203769 8.199329e-137
## host_total_listings_count     13.0068623   10.5571476  2.374333e-25
## host_listings_count           11.9589063    9.7834572  6.020503e-23
## multiple_host_listings         0.4896268    0.4998663  1.433015e-11
## availability_90               27.3780264   27.2542281  1.048765e-09
```

```
## availability_60               18.5552314   18.3326975  8.280424e-09
```

```
head(desc_clus$category$`2`)
```

```
##                                     Cla/Mod  Mod/Cla   Global      p.value
## property_type=Entire condo        38.44515 34.02451 22.15668 2.293104e-25
## host_has_profile_pic=TRUE         26.07940 99.05749 95.09202 2.271544e-15
## .Heating=No                       30.31309 60.22620 49.74044 2.671760e-15
## .Pack.\\u2019n.playTravel.crib=No 26.90797 90.38643 84.09627 1.469343e-11
## .Dining.table=Yes                 29.17046 60.32045 51.76970 1.086550e-10
## .Bidet=Yes                        28.56566 66.63525 58.40019 2.401612e-10
##                                      v.test
## property_type=Entire condo        10.407438
## host_has_profile_pic=TRUE          7.925541
## .Heating=No                        7.905353
## .Pack.\\u2019n.playTravel.crib=No  6.750897
## .Dining.table=Yes                  6.454391
## .Bidet=Yes                         6.333179
```

Homework: Describe the clusters!