Daily trip distribution analysis of Oslo bicycle-sharing system using PCA and K-means clustering

CYPLAN 257

Ola Tranum Arnegaard*

This document serves as an overview of the scripts used to analyze the trip dynamics in Oslo bicycle-sharing system (BSS). The analysis is based on the trip distribution histograms from the weekday trip data of 2019, as well as some demographic information of the city. The results show that Oslo BSS is heavily influenced by a work/school commuter pattern.

Packages

These are the packages that are needed to perform the analysis.

```
library(tidyr)
library(plyr)
library(dplyr)
library(lubridate)
library(timeDate)
library(qcc)
```

Pre-processing

Fetching the data. Available at *https://oslobysykkel.no/en/open-data/historical*. Note that the variable *trips_path* contains a previously initialized path.

```
trips_raw <- read.csv(trips_path, header=TRUE)
```

This section is based on bicycle pickups. To calculate the results for drop-offs, simply change the *start* keyword to *end*. Selecting the desired columns as well as changing the date and time format.

```
trips_processed_1 <- trips_raw %>%
  select(start_station_name, started_at) %>%
  mutate(day = day(started_at),
         month = month(started_at),
         isWeekday = isWeekday(started_at),
         hour = hour(started_at)) %>%
  select(-started_at)
```

*UC Berkeley, BGA Exchange Fall 2019, Norwegian University of Science and Technology

Selecting only weekdays, as weekends are not iincluded in the scope of this project.

```
trips_processed_2 <- trips_processed_1 %>%
  filter(isWeekday == TRUE) %>%
  select(-isWeekday)
```

Grouping trips for each hour of the day, all days, per station.

```
trips_processed_3 <- trips_processed_2 %>%
  group_by(start_station_name, day, month, hour) %>%
  dplyr::summarize(trips = n())
```

Creating a column for each hour, for each day and station.

```
trips_processed_4 <- trips_processed_3 %>%
  spread(hour, trips)
```

NA may appear for some trip combinations. To avoid problems later in the analysis, *NA's* are swapped with zeros. Also, zeros are added to non-operating hours of the day.

```
trips_processed_4[is.na(trips_processed_4)] <- 0

trips_processed_4$`1` <- 0
trips_processed_4$`2` <- 0
trips_processed_4$`3` <- 0
trips_processed_4$`4` <- 0
```

Removing unnecessary columns, grouping per station (one row per station) and calculating the mean of each column.

```
trips_processed_5 <- trips_processed_4[,c(-2, -3)] %>%
  group_by(start_station_name) %>%
  dplyr::summarise_all(funs(mean))
```

Selecting only the distribution data and normalizing each row.

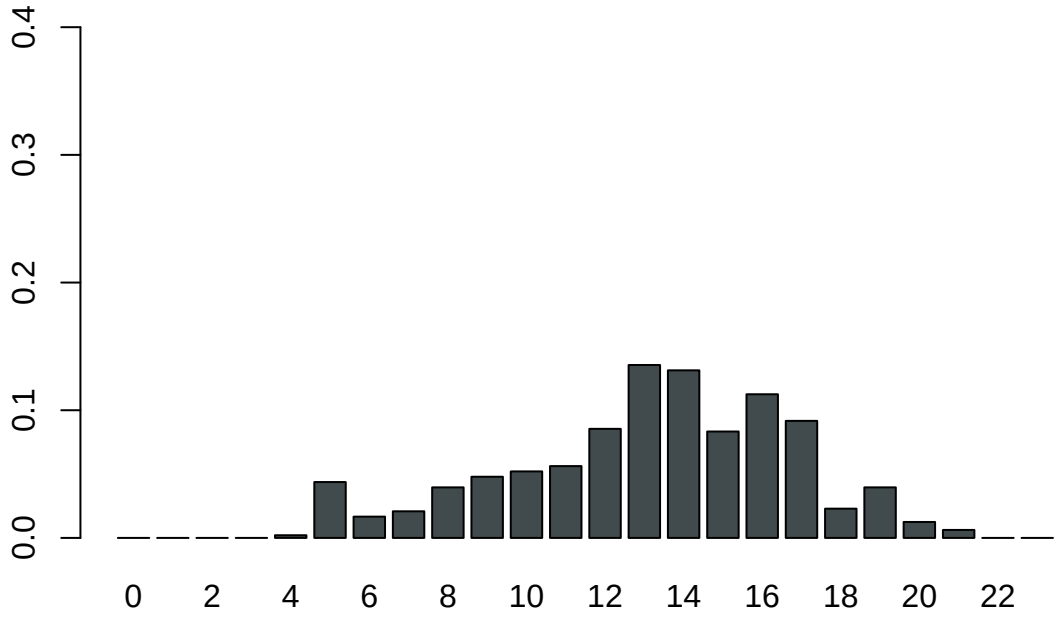
```
mat <- as.matrix(trips_processed_5[,2:25])
trips_normalized <- mat/rowSums(mat)
trips_normalized <- as.data.frame(trips_normalized)
```

Visualization

A simple function to visualize a given day.

```
plot_station <- function(y, i){
  x = 0:23
  barplot(height = t(t(y[i,])),
          names.arg = x,
          col="#414a4c",
```

```
          space = 0.25,
          ylim=c(0,0.4))
}
day <- 10
plot_station(trips_normalized, day)
```



Methods

PCA

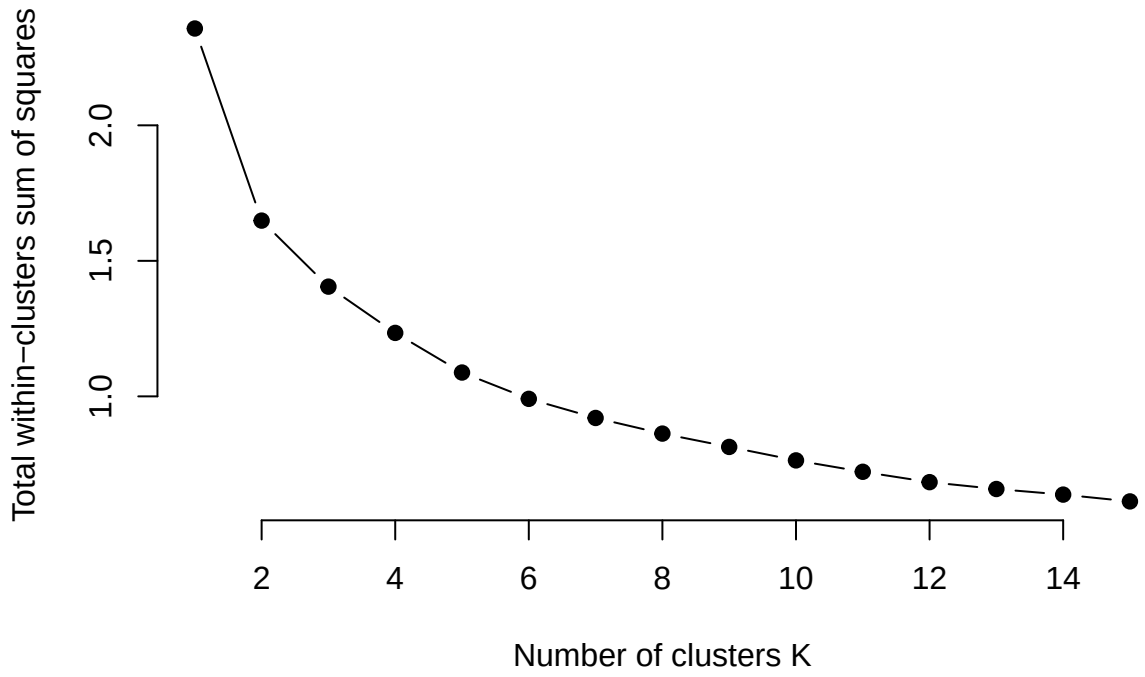
```
principal_components <- 7
trips_normalized.pca <- prcomp(trips_normalized)
df <- trips_normalized.pca$x[,1:principal_components]
```

K-means

Elbow method

This method is applied to find the optimal number of clusters to use in K-means.

```
k.max <- 15
data <- trips_normalized.pca$x[,1:principal_components]
wss <- sapply(1:k.max,
              function(k){kmeans(data,
                                 k,
                                 nstart=50,iter.max = 15 )$tot.withinss})
plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```



K-means

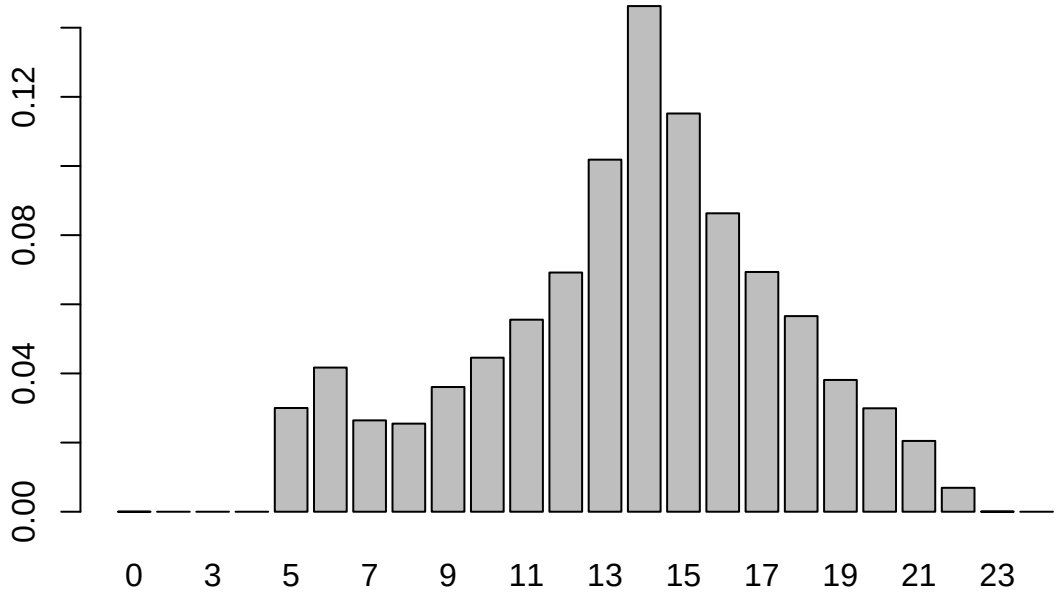
```
k <- 3
kmeans <- kmeans(df, k)
```

Reconstruction of Centers

Reconstructing each of the centers for each clusters. One of the reconstructinos is visualized below.

```
centers = kmeans$centers
means = colMeans(trips_normalized)
Xhat = centers[, 1:principal_components] %*%
  t(trips_normalized.pca$rotation[, 1:principal_components])
Xhat = scale(Xhat, center = -means, scale = FALSE)

barplot(Xhat[1,])
```



Adding the chosen cluster for each station

```
station <- trips_processed_5[,1]
station_cluster = data.frame(stations, df$cluster)
```

Fetching the metadata for the various station. Note that the variable *metadata_path* contains a previously initialized path.

```
station_lon_lat <- read.csv(metadata_path, header = TRUE) %>%
  select(start_station_name, start_station_latitude, start_station_longitude)
```

Joining the metadata and clusters to a new dataframe. The final dataframe may be utilized in software such as CartoDB to visualize the stations. Note that the variable *save_path* contains a previously initialized path.

```
data_complete <- inner_join(station_cluster, station_lon_lat)

write.csv(data_complete, file = save_path)
```

Critical stations

Joining the dataframes for pickups and drop-offs to a new dataframe. Note that the variable *pickups_path* and *dropoffs_path* contains a previously initialized paths.

```
pickups <- read.csv(pickups_path, header = TRUE)

dropoffs <- read.csv(dropoffs_path, header = TRUE)

dropoffs <- dropoffs %>% dplyr::rename(DropOffsCluster = df.cluster)
pickups <- pickups %>% dplyr::rename(PickUpsCluster = df.cluster)
joined <- inner_join(pickups, dropoffs)
```

Selecting clusters with unwanted combinations of pickup and drop-off distributions. For more information, see the project paper.

```
joined <- joined %>% select(-1)
joined <- joined %>% filter((PickUpsCluster == 3 & DropOffsCluster == 3) |
                            (PickUpsCluster == 1 & DropOffsCluster == 2))

write.csv(joined, file = save_path)
```