Case Study 1.1.1 - Genetic Codes

Shawn Gong

08/05/2020

Introduction

This is an case study from MITx PRO - Data Science and Big Data Analytics: Making Data-Driven Decisions about an application of clustering. The data provided is the genetics encoding of a DNA fragment of c.crescentus.

```
library(seqinr)
library(cluster)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::count()  masks seqinr::count()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

```
## The following object is masked from 'package:seqinr':
##
##     dotPlot
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

Pre-processing the data

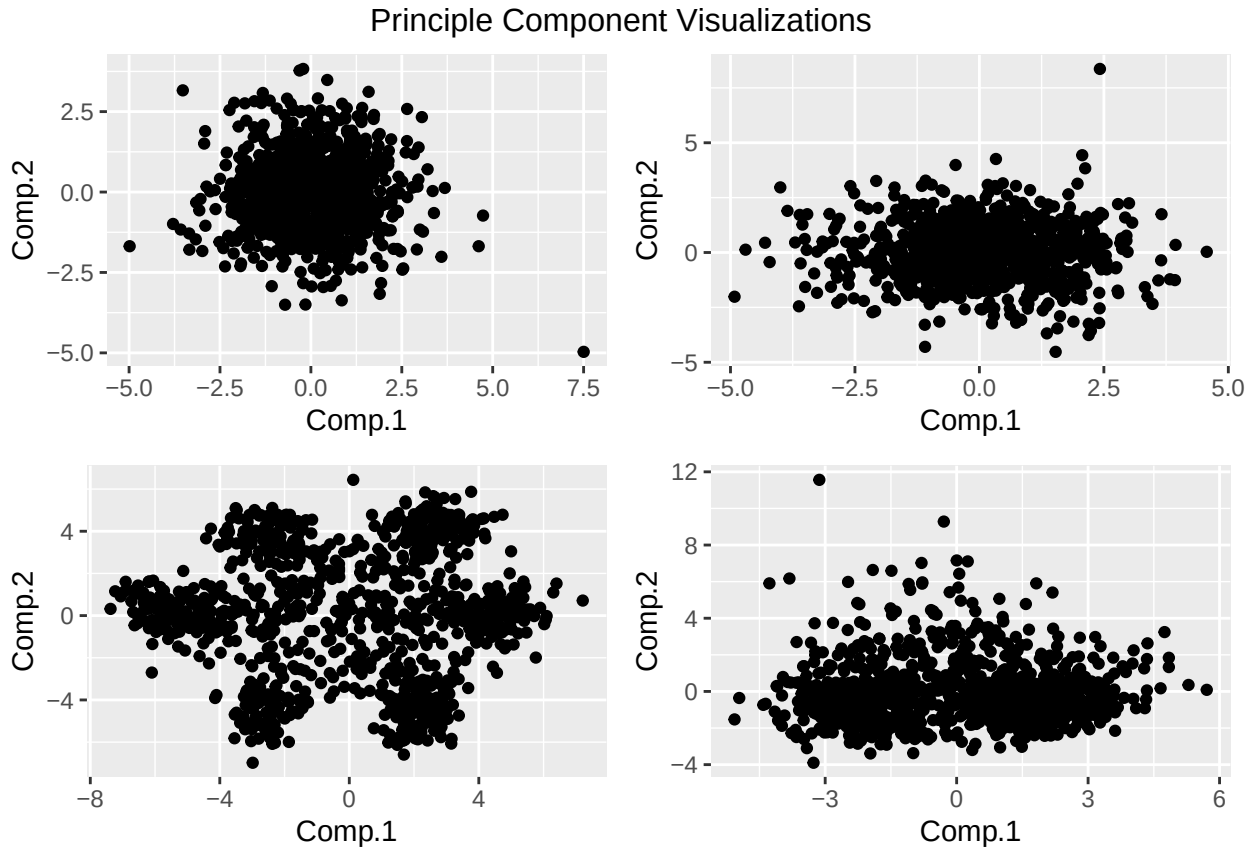
```r
# All possible combinations of 4 genetic units settings
source <- c("a", "t", "c", "g")

one_letter <- do.call(paste0, expand.grid(source))
two_letter <- do.call(paste0, expand.grid(source, source))
three_letter <- do.call(paste0, expand.grid(source, source, source))
four_letter <- do.call(paste0, expand.grid(source, source, source, source))


# Function to calculate the frequency of a string
tensiSplit <- function(string,size) {
  str_extract_all(string, paste0('.{1,',size,'}'))
}


CalcFreq <- function(x, word_vector){
  freq_list = list()
  for (word in word_vector){
    word_matrix <- matrix(tensiSplit(x, nchar(word)))
    frequency <- str_count(word_matrix, word)
    freq_list[[word]] <- frequency
  }
  return(as.data.frame(freq_list))
}

one_word <- CalcFreq(dna_seq$Genetic_letter, one_letter)
two_word <- CalcFreq(dna_seq$Genetic_letter, two_letter)
three_word <- CalcFreq(dna_seq$Genetic_letter, three_letter)
four_word <- CalcFreq(dna_seq$Genetic_letter, four_letter)
```

PCA Visualization

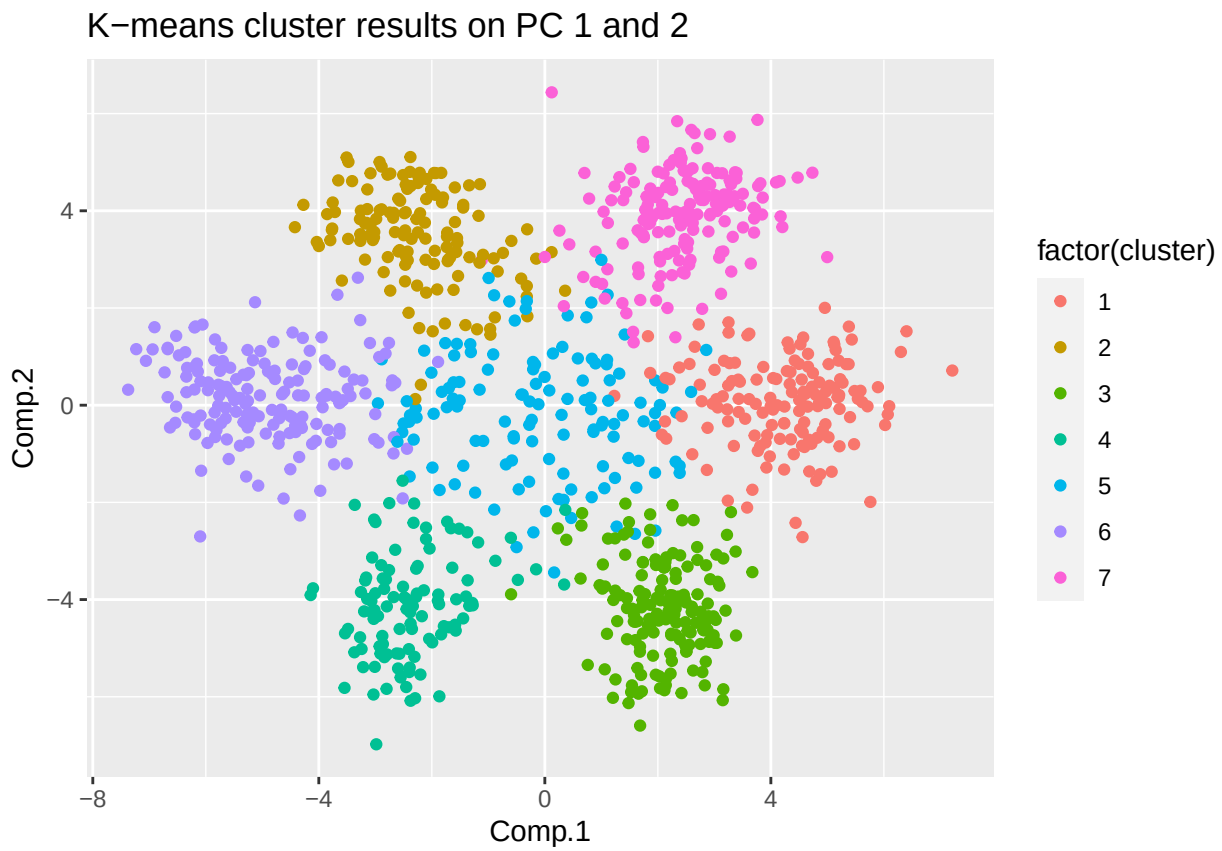
```r
# Normalization
normalization_one <- preProcess(one_word, method = c("center", "scale"))
normalization_two <- preProcess(two_word, method = c("center", "scale"))
normalization_three <- preProcess(three_word, method = c("center", "scale"))
normalization_four <- preProcess(four_word, method = c("center", "scale"))
```

```
normalized_one <- predict(normalization_one, one_word)
normalized_two <- predict(normalization_two, two_word)
normalized_three <- predict(normalization_three, three_word)
normalized_four <- predict(normalization_four, four_word)
```

```
# PCA Visualization
pca_one <- as.data.frame(princomp(normalized_one)$scores)
pca_two <- as.data.frame(princomp(normalized_two)$scores)
pca_three <- as.data.frame(princomp(normalized_three)$score)
pca_four <- as.data.frame(princomp(normalized_four)$score)

par(mfrow = c(2,2))

p1 <- pca_one %>%
  ggplot(aes(x = Comp.1, y = Comp.2)) +
  geom_point()
p2 <- pca_two %>%
  ggplot(aes(x = Comp.1, y = Comp.2)) +
  geom_point()

p3 <- pca_three %>%
  ggplot(aes(x = Comp.1, y = Comp.2)) +
  geom_point()

p4 <- pca_four %>%
  ggplot(aes(x = Comp.1, y = Comp.2)) +
  geom_point()

grid.arrange(p1, p2, p3, p4, nrow = 2, top = "Principle Component Visualizations")
```




```
#Clustering

cluster_threewords <- kmeans(pca_three, 7)

pca_three$cluster <- cluster_threewords$cluster


pca_three %>%
  ggplot(aes(x = Comp.1, y = Comp.2, colour = factor(cluster))) +
  geom_point() +
  ggtitle("K-means cluster results on PC 1 and 2")
```




```
# Mean Silhoutte Score
si1 <- mean(silhouette(pca_three$cluster, dist(pca_three[,1:64], "euclidean"))[, 3])

si1
```

```
## [1] 0.1198851
```