Week 14 IP- tSNE and PCA

Billiah

2022-04-01

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

```
library(devtools)
```

```
## Loading required package: usethis
```

```
library(factoextra)
```

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

```
library(Rtsne)
```

```
data <- read.csv("http://bit.ly/CarreFourDataset")
head(data)
```

```
##    Invoice.ID Branch Customer.type Gender           Product.line Unit.price
## 1 750-67-8428      A        Member Female      Health and beauty      74.69
## 2 226-31-3081      C        Normal Female Electronic accessories      15.28
## 3 631-41-3108      A        Normal   Male     Home and lifestyle      46.33
## 4 123-19-1176      A        Member   Male      Health and beauty      58.22
## 5 373-73-7910      A        Normal   Male      Sports and travel      86.31
## 6 699-14-3026      C        Normal   Male Electronic accessories      85.39
##   Quantity     Tax      Date  Time     Payment   cogs gross.margin.percentage
## 1        7 26.1415  1/5/2019 13:08     Ewallet 522.83               4.761905
## 2        5  3.8200  3/8/2019 10:29        Cash  76.40               4.761905
## 3        7 16.2155  3/3/2019 13:23 Credit card 324.31               4.761905
## 4        8 23.2880 1/27/2019 20:33     Ewallet 465.76               4.761905
```

```
## 5        7 30.2085  2/8/2019 10:37     Ewallet 604.17                4.761905
## 6        7 29.8865 3/25/2019 18:30     Ewallet 597.73                4.761905
##   gross.income Rating    Total
## 1      26.1415    9.1 548.9715
## 2       3.8200    9.6  80.2200
## 3      16.2155    7.4 340.5255
## 4      23.2880    8.4 489.0480
## 5      30.2085    5.3 634.3785
## 6      29.8865    4.1 627.6165
```

```
sum(is.na(data))
```

```
## [1] 0
```

There are no null values.

```
# Selecting the numerical columns
n_data <- data[c(6:8,12,14:16)]

data_pc<-prcomp(n_data, center = TRUE, scale. = TRUE)

summary(data_pc)
```

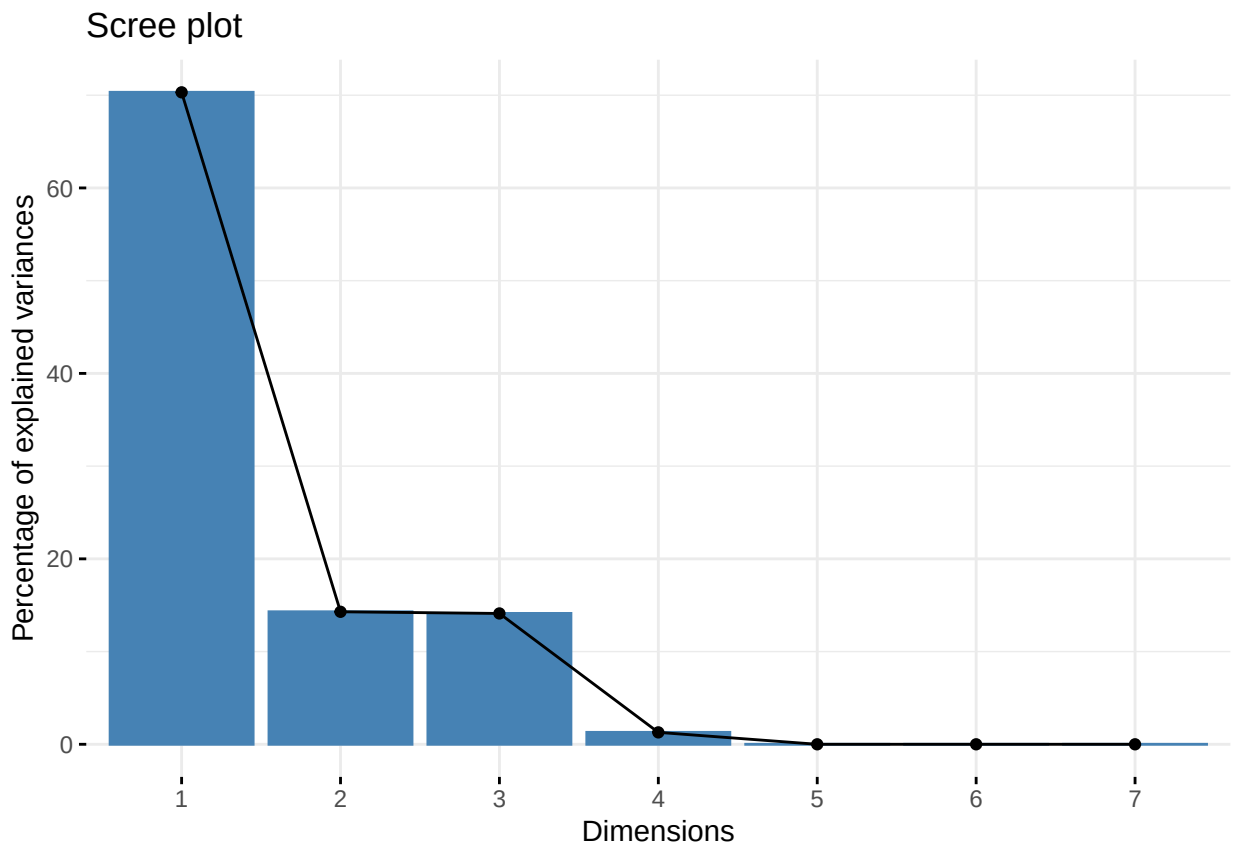
```
## Importance of components:
##                           PC1    PC2    PC3     PC4       PC5       PC6
## Standard deviation     2.2185 1.0002 0.9939 0.30001 2.981e-16 1.493e-16
## Proportion of Variance 0.7031 0.1429 0.1411 0.01286 0.000e+00 0.000e+00
## Cumulative Proportion  0.7031 0.8460 0.9871 1.00000 1.000e+00 1.000e+00
##                            PC7
## Standard deviation     9.831e-17
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

The standard deviation represents the eigenvalues.

Proportion of variancerepresents the amount of variance the component accounts for in the data. In this case, PC1 accounts for >70% of total variance in the data.

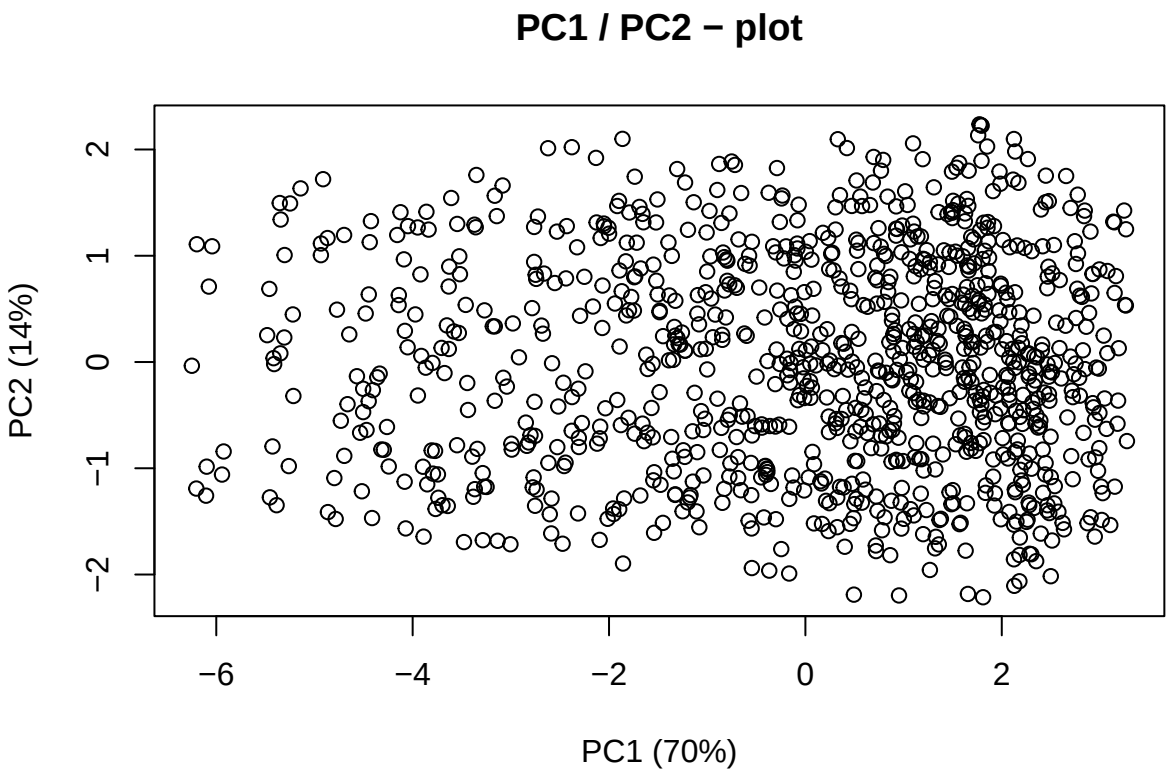
The cumulative proportion represents the accumulated amount of explained variance.If we use the first 3components would be able to account for >98% of total variance.

```
fviz_eig(data_pc)
```



The first 4 components have an elgenvalue of greater than 1 and cumulatively explain a variance $> 98\%$.

```
plot(data_pc$x[,1], data_pc$x[,2], xlab="PC1 (70%)", ylab = "PC2 (14%)", main = "PC1 / PC2 - plot")
```



The first 2 components can cumulatively explain 84% variance in data.

```
tSNE_data <- data$Y %>%
  as.data.frame()
```

```
## for plotting
colors = rainbow(length(unique(data$Branch)))
names(colors) = unique(data$Branch)

## Executing the algorithm on curated data
tsne <- Rtsne(n_data[,-2],  perplexity=30, verbose=TRUE, max_iter = 500)
```

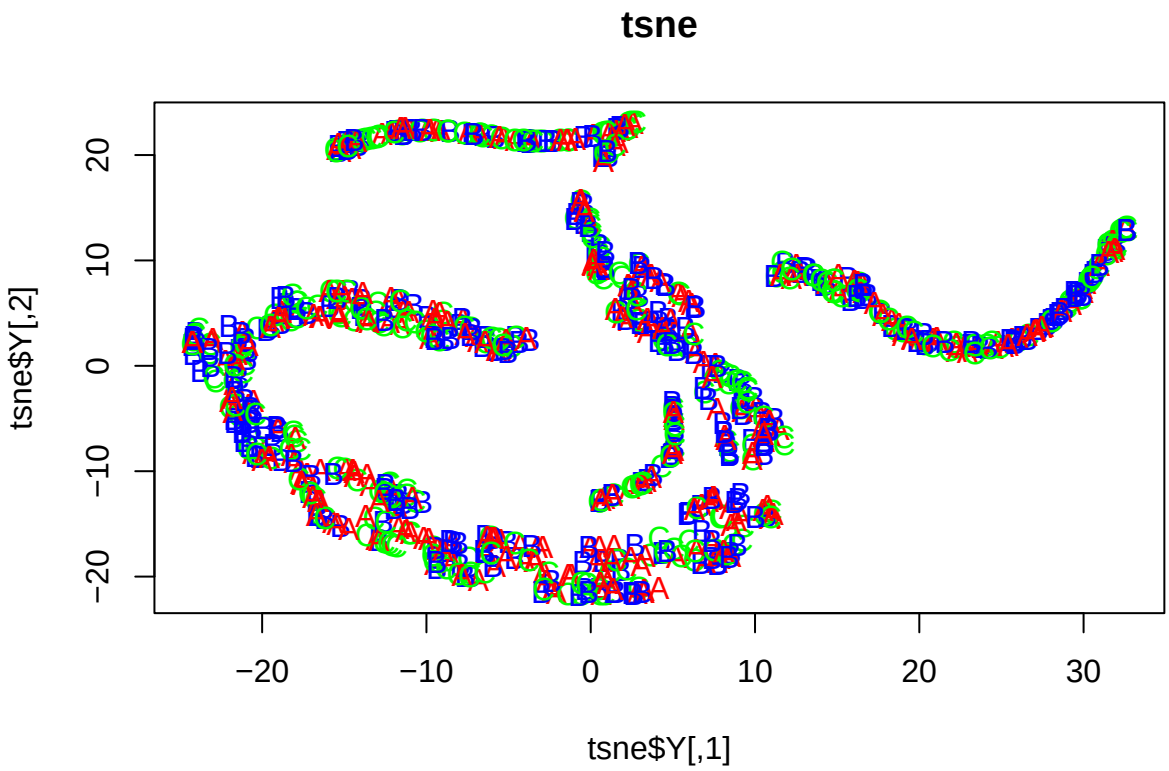
```
## Performing PCA
## Read the 1000 x 6 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.24 seconds (sparsity = 0.101226)!
## Learning embedding...
## Iteration 50: error is 59.553638 (50 iterations in 0.20 seconds)
## Iteration 100: error is 52.748385 (50 iterations in 0.17 seconds)
## Iteration 150: error is 51.706306 (50 iterations in 0.16 seconds)
## Iteration 200: error is 51.309002 (50 iterations in 0.20 seconds)
## Iteration 250: error is 51.108999 (50 iterations in 0.25 seconds)
## Iteration 300: error is 0.574104 (50 iterations in 0.35 seconds)
```

```
## Iteration 350: error is 0.417219 (50 iterations in 0.49 seconds)
## Iteration 400: error is 0.381212 (50 iterations in 0.53 seconds)
## Iteration 450: error is 0.361272 (50 iterations in 0.41 seconds)
## Iteration 500: error is 0.352785 (50 iterations in 0.57 seconds)
## Fitting performed in 3.34 seconds.
```

```
exeTimeTsne<- system.time(Rtsne(n_data[,-2], perplexity=30, verbose=TRUE, max_iter = 500))
```

```
## Performing PCA
## Read the 1000 x 6 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.57 seconds (sparsity = 0.101226)!
## Learning embedding...
## Iteration 50: error is 58.569547 (50 iterations in 0.39 seconds)
## Iteration 100: error is 52.188941 (50 iterations in 0.53 seconds)
## Iteration 150: error is 51.199608 (50 iterations in 0.43 seconds)
## Iteration 200: error is 50.720741 (50 iterations in 0.24 seconds)
## Iteration 250: error is 50.377366 (50 iterations in 0.39 seconds)
## Iteration 300: error is 0.552002 (50 iterations in 0.23 seconds)
## Iteration 350: error is 0.396087 (50 iterations in 0.35 seconds)
## Iteration 400: error is 0.360484 (50 iterations in 0.44 seconds)
## Iteration 450: error is 0.347032 (50 iterations in 0.37 seconds)
## Iteration 500: error is 0.341369 (50 iterations in 0.29 seconds)
## Fitting performed in 3.64 seconds.
```

```
## Plotting
plot(tsne$Y, t='n', main="tsne")
text(tsne$Y, labels=data$Branch, col=colors[data$Branch])
```



There is some structure and patterns to the data.