hw03_01

Weijun Zhu

November 12, 2019

Contents


```r
library(tidyverse)
```

```
## -- Attaching packages --------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

1a).

Download the zip file that contains the data. Using R data frames, or Python pandas, transform this datset into a data frame. In doing so you need to perform the following filtering and cleaning of the data: - Unzip the data, and read into a data frame called movieDat - For this homework we need only those features which are numerical. Create a new data frame called nmovieDat and copy only columns of movieDat which are numerical in it. Once you have prepared the nmovieDat data frame, print the head and tail to make sure you have read everything correctly.

```r
# movieDat <- read.csv('C:/Users/zhuwe/Desktop/ML/ML_Homework/hw_code/data/tmdb_5000_movies.csv',
#                          header = TRUE, sep = ',', fill = TRUE, na.strings=c("", "NA"))

movieDat <- read_csv('C:/Users/zhuwe/Desktop/ML/ML_Homework/hw_code/data/tmdb_5000_movies.csv',
                     na="")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
```

```
##   budget = col_double(),
##   id = col_double(),
##   popularity = col_double(),
##   release_date = col_date(format = ""),
##   revenue = col_double(),
##   runtime = col_double(),
##   vote_average = col_double(),
##   vote_count = col_double()
## )

## See spec(...) for full column specifications.
```

```
movieDat <- as.data.frame(movieDat)
```

```
str(movieDat)
```

```
## 'data.frame':    4803 obs. of  20 variables:
##  $ budget              : num  2.37e+08 3.00e+08 2.45e+08 2.50e+08 2.60e+08 2.58e+08 2.60e+08 2.80e+08
##  $ genres              : chr  "[{\"id\": 28, \"name\": \"Action\"}, {\"id\": 12, \"name\": \"Adventu
##  $ homepage            : chr  "http://www.avatarmovie.com/" "http://disney.go.com/disneypictures/pira
##  $ id                  : num  19995 285 206647 49026 49529 ...
##  $ keywords            : chr  "[{\"id\": 1463, \"name\": \"culture clash\"}, {\"id\": 2964, \"name\"
##  $ original_language   : chr  "en" "en" "en" "en" ...
##  $ original_title      : chr  "Avatar" "Pirates of the Caribbean: At World's End" "Spectre" "The Dark
##  $ overview            : chr  "In the 22nd century, a paraplegic Marine is dispatched to the moon Pan
##  $ popularity          : num  150.4 139.1 107.4 112.3 43.9 ...
##  $ production_companies: chr  "[{\"name\": \"Ingenious Film Partners\", \"id\": 289}, {\"name\": \"Tw
##  $ production_countries: chr  "[{\"iso_3166_1\": \"US\", \"name\": \"United States of America\"}, {\
##  $ release_date        : Date, format: "2009-12-10" "2007-05-19" ...
##  $ revenue             : num  2.79e+09 9.61e+08 8.81e+08 1.08e+09 2.84e+08 ...
##  $ runtime             : num  162 169 148 165 132 139 100 141 153 151 ...
##  $ spoken_languages    : chr  "[{\"iso_639_1\": \"en\", \"name\": \"English\"}, {\"iso_639_1\": \"es\
##  $ status              : chr  "Released" "Released" "Released" "Released" ...
##  $ tagline             : chr  "Enter the World of Pandora." "At the end of the world, the adventure b
##  $ title               : chr  "Avatar" "Pirates of the Caribbean: At World's End" "Spectre" "The Dark
##  $ vote_average        : num  7.2 6.9 6.3 7.6 6.1 5.9 7.4 7.3 7.4 5.7 ...
##  $ vote_count          : num  11800 4500 4466 9106 2124 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   budget = col_double(),
##   ..   genres = col_character(),
##   ..   homepage = col_character(),
##   ..   id = col_double(),
##   ..   keywords = col_character(),
##   ..   original_language = col_character(),
##   ..   original_title = col_character(),
##   ..   overview = col_character(),
##   ..   popularity = col_double(),
##   ..   production_companies = col_character(),
##   ..   production_countries = col_character(),
##   ..   release_date = col_date(format = ""),
##   ..   revenue = col_double(),
##   ..   runtime = col_double(),
```

```
##   ..   spoken_languages = col_character(),
##   ..   status = col_character(),
##   ..   tagline = col_character(),
##   ..   title = col_character(),
##   ..   vote_average = col_double(),
##   ..   vote_count = col_double()
##   .. )
```

```r
nmovieDat <-  movieDat %>%
  select(id, budget, popularity, revenue, runtime, vote_average, vote_count)
nmovieDat <- na.omit(nmovieDat)
```

```r
head(nmovieDat)
```

```
##       id   budget popularity    revenue runtime vote_average vote_count
## 1  19995 2.37e+08  150.43758 2787965087     162          7.2      11800
## 2    285 3.00e+08  139.08262  961000000     169          6.9       4500
## 3 206647 2.45e+08  107.37679  880674609     148          6.3       4466
## 4  49026 2.50e+08  112.31295 1084939099     165          7.6       9106
## 5  49529 2.60e+08   43.92699  284139100     132          6.1       2124
## 6    559 2.58e+08  115.69981  890871626     139          5.9       3576
```

```r
tail(nmovieDat)
```

```
##          id budget popularity revenue runtime vote_average vote_count
## 4798  67238      0   0.022173       0      80          7.5          2
## 4799   9367 220000  14.269792 2040920      81          6.6        238
## 4800  72766   9000   0.642552       0      85          5.9          5
## 4801 231617      0   1.444476       0     120          7.0          6
## 4802 126186      0   0.857008       0      98          5.7          7
## 4803  25975      0   1.929883       0      90          6.3         16
```

1b).

Find the mean, variance and standard deviation of each numerical feature and print your results. Comment on whether it is wise to do a scaling of the data before running any type of clustering or not.

```r
print('budget:')
```

```
## [1] "budget:"
```

```r
mean(nmovieDat$budget)
```

```
## [1] 29054015
```

```r
var(nmovieDat$budget)
```

```
## [1] 1.658787e+15
```

```r
sd(nmovieDat$budget)
```

```
## [1] 40728211
```

```r
print('popularity:')
```

```
## [1] "popularity:"
```

```r
mean(nmovieDat$popularity)
```

```
## [1] 21.50109
```

```r
var(nmovieDat$popularity)
```

```
## [1] 1012.535
```

```r
sd(nmovieDat$popularity)
```

```
## [1] 31.82036
```

```r
print('revenue')
```

```
## [1] "revenue"
```

```r
mean(nmovieDat$revenue)
```

```
## [1] 82294907
```

```r
var(nmovieDat$revenue)
```

```
## [1] 2.653067e+16
```

```r
sd(nmovieDat$revenue)
```

```
## [1] 162882368
```

```r
print('runtime')
```

```
## [1] "runtime"
```

```r
mean(nmovieDat$runtime)
```

```
## [1] 106.8759
```

```r
var(nmovieDat$runtime)
```

```
## [1] 511.2996
```

```r
sd(nmovieDat$runtime)
```

```
## [1] 22.61193
```

```r
print('vote_average:')
```

```
## [1] "vote_average:"
```

```r
mean(nmovieDat$vote_average)
```

```
## [1] 6.093189
```

```r
var(nmovieDat$vote_average)
```

```
## [1] 1.419656
```

```r
sd(nmovieDat$vote_average)
```

```
## [1] 1.191493
```

```r
print('vote_count:')
```

```
## [1] "vote_count:"
```

```r
mean(nmovieDat$vote_count)
```

```
## [1] 690.503
```

```r
var(nmovieDat$vote_count)
```

```
## [1] 1524642
```

We need to do scaling of the data, because variables of the data have the huge different size. Like if we do not scaling the data, when we use Gradient Decent Algorithm, it hard to get optimization.

1c).

Now apply the principal component analysis on the data using prcomp in R (or the equivalent in Python.) Based on your response on part 1b) above decide whether the option scale should be true or false. Save the output of prcomp in an object called usarrests.pca To see the components of this object, use the names function and see what information is in it. Print the result. Extract the center and scale. Explain why these values are different (or the same as) the mean and variances above.

```r
nmovies.pca <- prcomp(nmovieDat, retx = TRUE, na.action=na.omit, scale=TRUE)
```

```
## Warning: In prcomp.default(nmovieDat, retx = TRUE, na.action = na.omit, scale = TRUE) :
##  extra argument 'na.action' will be disregarded
```

```r
names(nmovies.pca)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```r
print(nmovies.pca$center)
```

```
##           id       budget   popularity      revenue      runtime
## 5.701632e+04 2.905402e+07 2.150109e+01 8.229491e+07 1.068759e+02
## vote_average   vote_count
## 6.093189e+00 6.905030e+02
```

```r
print(nmovies.pca$scale)
```

```
##           id       budget   popularity      revenue      runtime
## 8.840674e+04 4.072821e+07 3.182036e+01 1.628824e+08 2.261193e+01
## vote_average   vote_count
## 1.191493e+00 1.234764e+03
```

1d).

Print the rotation matrix. rotation: The matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). The function princomp returns this in the element loadings.

```r
nmovies.pca$rotation
```

```
##                     PC1        PC2        PC3         PC4        PC5
## id           0.06607628  0.6041530  0.7361404  0.10116090 -0.2789185
## budget      -0.42961663  0.1582537 -0.3361226  0.36170537 -0.4886930
## popularity  -0.45590140  0.1546114  0.1390846 -0.30787273  0.5391144
## revenue     -0.48759176  0.1744548 -0.1775055  0.04635118 -0.1962855
## runtime     -0.25225920 -0.4494393  0.3944254  0.71141202  0.2627088
## vote_average -0.23266711 -0.5802485  0.3670750 -0.46157304 -0.5011847
## vote_count  -0.49769767  0.1301220  0.0628553 -0.20699014  0.1844160
##                     PC6         PC7
## id           0.02062735  0.01661001
## budget       0.51688006 -0.20147045
## popularity   0.51286699  0.31692609
## revenue     -0.56454514  0.58388693
## runtime     -0.05720747  0.02006374
## vote_average  0.08998643  0.04582901
## vote_count  -0.37322313 -0.71781949
```

1e).

Extract the standard deviation of the principal components, these are the singular values of the data matrix. Verify this by using matrix operations (svd, matrix multiplication, etc.)

```
summary(nmovies.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5     PC6     PC7
## Standard deviation     1.8097 1.1584 0.9178 0.8436 0.6416 0.50763 0.39968
## Proportion of Variance 0.4679 0.1917 0.1203 0.1017 0.0588 0.03681 0.02282
## Cumulative Proportion  0.4679 0.6596 0.7799 0.8816 0.9404 0.97718 1.00000
```

```
# From the documentation of prcomp, we can get:
# sdev: the standaed deviations of principal components(i.e., the square roots of the eigenvalues of th
sqrt(eigen(cor(nmovieDat))$values)
```

```
## [1] 1.8097317 1.1583610 0.9178107 0.8436121 0.6415504 0.5076251 0.3996787
```

Then I try to use matrix multiplication to figure out the standard deviation og the principal components.

```
x = as.matrix(nmovieDat)  # set the data as matrix
c=(t(x)%*%x)/(dim(x)[1]-1)
```

```
svd(c)
```

```
## $d
## [1] 3.492125e+16 8.863817e+14 1.035188e+10 6.011615e+05 5.770976e+03
## [6] 3.924974e+02 1.808549e+00
##
## $u
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,] -1.194859e-04 -4.946328e-04 -9.999991e-01  1.143057e-03  3.692382e-04
## [2,] -2.179491e-01 -9.759600e-01  5.087880e-04  3.074748e-06  1.092921e-06
## [3,] -1.508306e-07 -1.523171e-07 -6.001699e-05 -1.875867e-02 -5.902151e-02
## [4,] -9.759601e-01  2.179491e-01  8.815114e-06  5.626581e-06 -9.484417e-08
## [5,] -2.926393e-07 -1.302834e-06 -3.859935e-04 -1.662575e-02 -9.965412e-01
## [6,] -1.622010e-08 -6.718093e-08 -2.145497e-05 -1.199339e-03 -5.574062e-02
## [7,] -6.292308e-06 -2.316049e-06 -1.135844e-03 -9.996844e-01  1.774828e-02
##               [,6]          [,7]
## [1,]  1.681105e-05  5.454693e-07
## [2,]  2.882480e-08  6.513724e-09
## [3,] -9.980617e-01 -6.120077e-03
## [4,]  1.355214e-08  5.012651e-10
## [5,]  5.958434e-02 -5.548725e-02
## [6,] -2.803573e-03  9.984406e-01
## [7,]  1.774065e-02 -1.601972e-04
##
## $v
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,] -1.194859e-04 -4.946328e-04 -9.999991e-01  1.143057e-03  3.692382e-04
## [2,] -2.179491e-01 -9.759600e-01  5.087880e-04  3.074748e-06  1.092921e-06
```

```
## [3,] -1.508306e-07 -1.523171e-07 -6.001699e-05 -1.875867e-02 -5.902151e-02
## [4,] -9.759601e-01  2.179491e-01  8.815114e-06  5.626581e-06 -9.484417e-08
## [5,] -2.926393e-07 -1.302834e-06 -3.859935e-04 -1.662575e-02 -9.965412e-01
## [6,] -1.622010e-08 -6.718093e-08 -2.145497e-05 -1.199339e-03 -5.574062e-02
## [7,] -6.292308e-06 -2.316049e-06 -1.135844e-03 -9.996844e-01  1.774828e-02
##               [,6]          [,7]
## [1,]  1.681105e-05  5.454693e-07
## [2,]  2.882480e-08  6.513724e-09
## [3,] -9.980617e-01 -6.120077e-03
## [4,]  1.355214e-08  5.012651e-10
## [5,]  5.958434e-02 -5.548725e-02
## [6,] -2.803573e-03  9.984406e-01
## [7,]  1.774065e-02 -1.601972e-04
```

```r
principalComponent = x%*%svd(c)$u
```

```r
for (i in seq(dim(principalComponent)[2])) {
  cat("The", i, "of standard deviation of principal component is: ")
  print(sd(principalComponent[,i]))
}
```

```
## The 1 of standard deviation of principal component is: [1] 165564666
## The 2 of standard deviation of principal component is: [1] 27888949
## The 3 of standard deviation of principal component is: [1] 92889.69
## The 4 of standard deviation of principal component is: [1] 771.7164
## The 5 of standard deviation of principal component is: [1] 56.42566
## The 6 of standard deviation of principal component is: [1] 19.81148
## The 7 of standard deviation of principal component is: [1] 1.333668
```

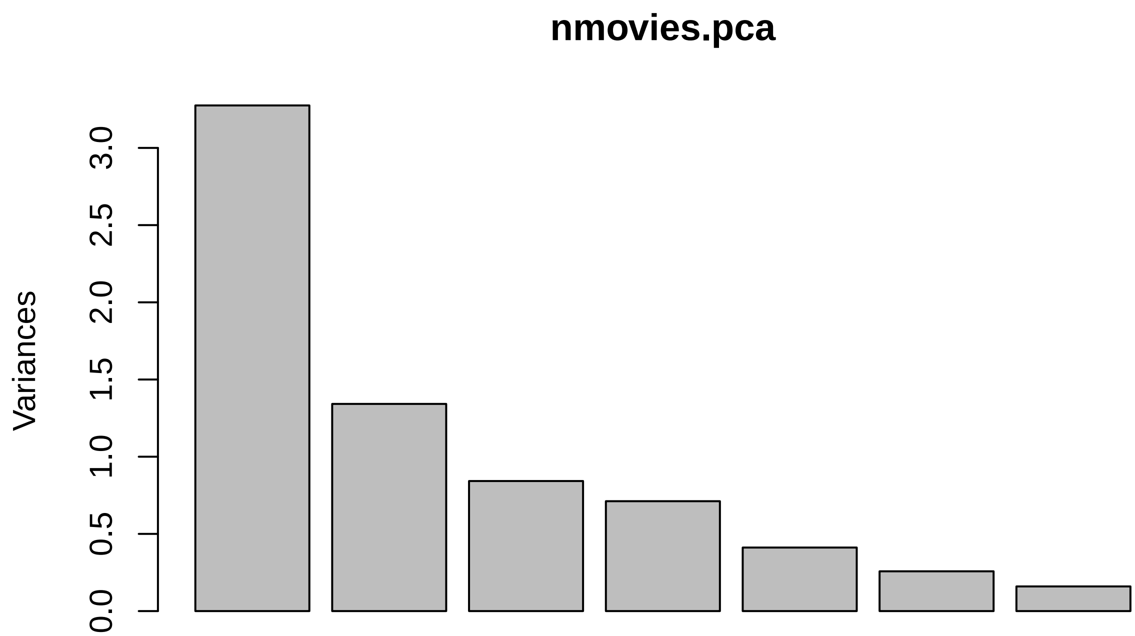
1f).

Print and graph the contribution of each principal component by graphing their variance (from the highest to lowest). You may use the screeplot function in R, see its documentation. Do you think it would be justified to keep all four features, or would it be wise to drop one or more? Justify your answer.

```r
screeplot(nmovies.pca, labels=TRUE)
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): "labels" is not a
## graphical parameter
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "labels" is not a graphical parameter
```

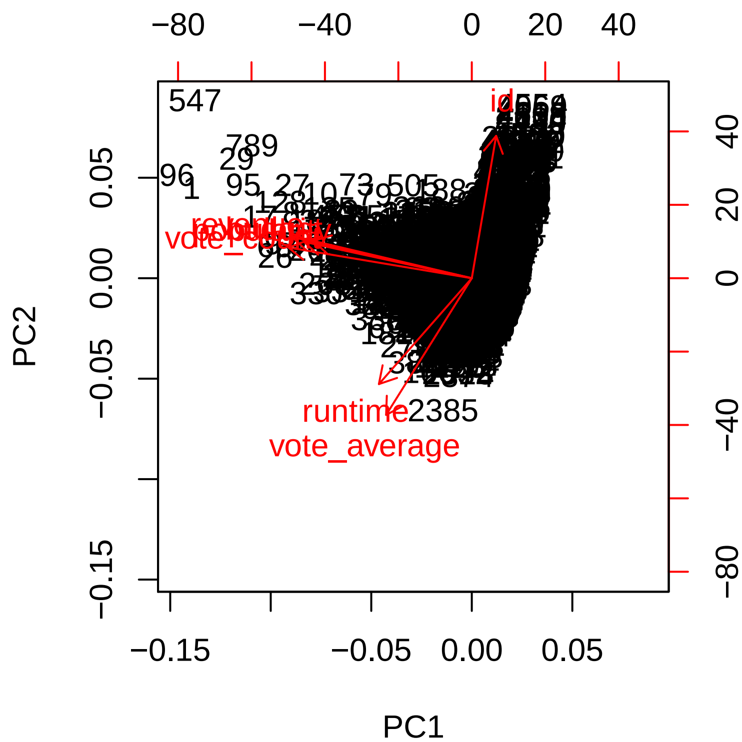


Conclusion: For the first four components, they occupy about 75% of total components, so we can not drop any one of them. If we drop one of these four components, it will affect consquence so much. we can drop the last one of the feature, and keep others features.

1g).

Use the biplot function of R (see its documentation) to plot the "factor loadings" of each data point on the first two principal components. Observe all vectors corresponding to all numerical features. Each one has a PC1 and a PC2 components. Observe the graph. Which features have roughly similar factor loading (coordinates in each PCA direction), and which one(s) are visibly different from the others?

```r
biplot(nmovies.pca)
```



Conclusion: We can see the "vote_average" and "runtime" have the same directions, and "budget", "popularity", "revenue" and "vote_count" have the same directions. "id" has the different direction compared with others variables, and it is visibly different from the others. The projection of "vote_average" on PC2 more influence than the projection of "runtime" on PC2. For those others variables exoect "id", not clearly to see from the graph, but the projection of those four variables strongly influence PC1. The angle between "id" and one of others vector is bigger than 90 degree, that means "id" is not likely to be correlated with one of others variables.