Managing Big Data
First Assignment 2022-2023

Davoulou Foteini 1054757 Vasiliki Iliopoulou 1067639

Dimakopoulos Andreas 1067668 Moustakas Andreas 1067656

Giannakos Giorgos 1038608

November 1, 2022

Contents

A Paper Summary

What is the purpose of the research?

The primary goal of this study is to simplify the economic system's activities into a single hyper-industry.

What statistical methods were used?

In terms of statistical techniques, this study uses principal components analysis (PCA), an efficient dimensionality reduction method that creates pertinent features by combining the original data in linear fashion. In order to determine how many groups of industries should be formed, the Silhouette method is used followed by the k-means clustering algorithm.

What kind of data was used (i.e. where did they get the data from)? What was the size of the data?

The data comes from the 2002, 2007, 2012 US input-output benchmarks and the most recently published 2019 input-output table. This data-table has 70 columns and 70 rows since we eliminated the fictitious home industry and housing industry due to their numerous imputations.

Why was the Principal Component Analysis (PCA) method used?

Because of its benefits, the PCA is particularly useful for detecting the relative importance of different industries in the operation of the economy when input-output data is included. The results of the PCA are then compared to the overall BL and FL values.

How is the Principal Component Analysis's output interpreted?

PCA takes the process of identifying key businesses in new areas. There is a strong association between the most powerful computers and the overall number of connections between industries. A finding that supports the application of PCA to input-output data in order to identify important industries.

Has the research's initial goal been attained?

The ability to cluster and dendrogram the industries allowed this study to successfully accomplish its main objectives. The way the networks are presented reveals the interactions and connections among the clusters' hierarchical positions and the connections between the clusters' industries. However, future studies should concentrate on using the aforementioned methods to input-output data from other nations and years. In this regard, more thorough input-output statistics for the sector will improve our comprehension of how different industries are connected to one another and how economies are changing structurally.

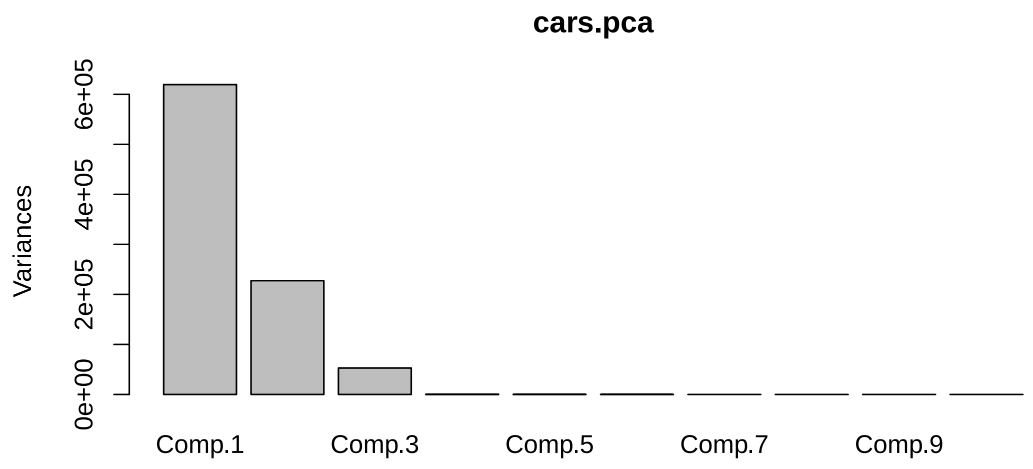
B Principal Component Analysis

B.1 R

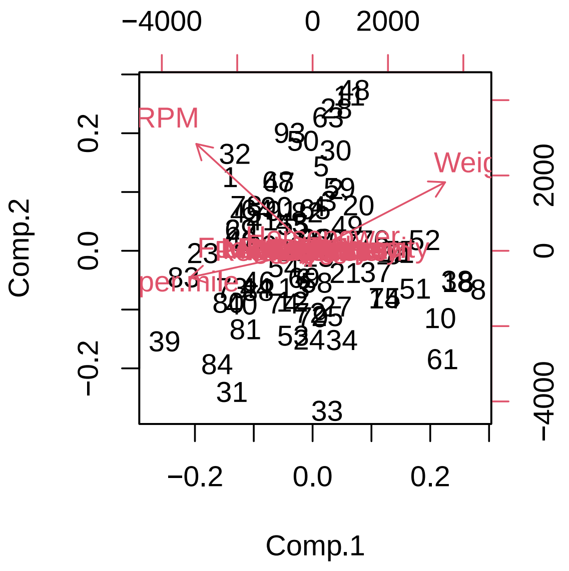
```
        ##########  Erwthma 3a ###########
library(MASS)

graphics.off() ; rm(list = ls(all = TRUE)) ; cat("\014");

data("Cars93")

summary(Cars93)

colSums(is.na(Cars93)) #where are the missing values if any?
df<- na.omit(Cars93) #remove the missing values
cars.pca <- princomp(df[,c(4:8,12:15,17:25)], cor = FALSE,scores = TRUE)

summary(cars.pca)

plot(cars.pca)
biplot(cars.pca)
```

Table 1: Table of Principal Components

	Comp 1	Comp 2	Comp 3	Comp 4	Comp 5
Standard deviation	787.0555500	476.9781520	230.13274971	1.967222e+01	1.003801e+01
Proportion of Variance	0.6879009	0.2526458	0.05881281	4.297557e-04	1.118949e-04
Cumulative Proportion	0.6879009	0.9405467	0.99935948	9.997892e-01	9.999011e-01
	Comp 6	Comp 7	Comp 8	Comp 9	Comp 10
Standard deviation	6.693292e+00	4.208960e+00	2.941145e+00	2.369849e+00	1.811967e+00
Proportion of Variance	4.975017e-05	1.967273e-05	9.606117e-06	6.236722e-06	3.645993e-06
Cumulative Proportion	9.999509e-01	9.999705e-01	9.999802e-01	9.999864e-01	9.999900e-01
	Comp 11	Comp 12	Comp 13	Comp 14	Comp 15
Standard deviation	1.691019e+00	1.571690e+00	1.138831e+00	1.104324e+00	9.648784e-01
Proportion of Variance	3.175498e-06	2.743144e-06	1.440236e-06	1.354279e-06	1.033856e-06
Cumulative Proportion	9.999932e-01	9.999960e-01	9.999974e-01	9.999988e-01	9.999998e-01
	Comp 16	Comp 17	Comp 18		
Standard deviation	3.798738e-01	2.201354e-01	1.937776e-02		
Proportion of Variance	1.602484e-07	5.381397e-08	4.169869e-10		
Cumulative Proportion	9.999999e-01	1.000000e+00	1.000000e+00		



cars.pca
Variances
0e+00
2e+05
4e+05
6e+05
Comp.1
Comp.3
Comp.5
Comp.7
Comp.9



RPM
Weig
per.mile
Horsepower
Comp.1
Comp.2
−4000
0
2000
−0.2
0.0
0.2

B.2 Python

```
########## Erwthma 3 ###########
print("Erwthma 3")

import pandas as pd
from sklearn import decomposition
from sklearn.decomposition import PCA
import csv

#read csv
data=pd.read_csv("/Users/dhmako/Downloads/winequalitywhite.csv",
                                sep=";", header=0, decimal=".")


print(data.head())

pca = decomposition.PCA(n_components=4)
pca.fit(data)

print("Eigenvectors:")
print(pca.components_)

transformeddata = pca.transform(data)

print("\nNew data points, on the space defined by the
                                Eigenvectors")
print( transformeddata)
print("\n Variance")
print (pca.explained_variance_)
print (pca.explained_variance_ratio_)
print (pca.explained_variance_ratio_.cumsum())
```

C Euclidean Distance

C.1 Answers

Euclidean distance formula is $d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$

i)
A
The Euclidean distance between x and y is 0
B
The Euclidean distance between x and y is 40.78382
C
The Euclidean distance between x and y is 24.21059
D
The Euclidean distance between x and y is 0.3464102
ii)
A
Using the Euclidean distance formula we are lead to believe user 3 is the closest to user 5 with distance between them being 3000.
B
Since we are using the Euclidean distance, the minutes variable—which has the largest numbers—has the greatest influence on our outcome.

C.2 R

```
        ##########  Erwthma 4a ###########
graphics.off() ; rm(list = ls(all = TRUE)) ; cat("\014");

euclideanDistance<- function(x, y) sqrt(sum((x - y)^2))

#a
x<-c(1,2,3,4,5,6)
y<-c(1,2,3,4,5,6)
print("Euclidean␣distance␣between␣x␣and␣y␣is:␣")
euclideanDistance(x,y)

#b
x=c(-0.5, 1, 7.3, 7, 9.4, -8.2, 9, -6, -6.3)
y=c(0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3)
print("Euclidean␣distance␣between␣x␣and␣y␣is:␣")
euclideanDistance(x,y)

#c
x=c(-0.5, 1, 7.3, 7, 9.4, -8.2)
y=c(1.25, 9.02, -7.3, -7, 5, 1.3)
print("Euclidean␣distance␣between␣x␣and␣y␣is:␣")
euclideanDistance(x,y)

#d
x=c(0, 0, 0.2)
y=c(0.2, 0.2, 0)
print("Euclidean␣distance␣between␣x␣and␣y␣is:␣")
euclideanDistance(x,y)

##########  Erwthma 4b ###########
x=c(25000,14,7)
y=c(42000,17,9)
z=c(55000,22,5)
n=c(27000,13,11)
m=c(58000,21,13) ##target of comparison

eucldistvector=c(euclideanDistance(x,m),euclideanDistance(y,m),
euclideanDistance(z,m),euclideanDistance(n,m))

print("The␣user␣that␣is␣most␣similiar␣to␣user␣with␣id=5␣is")
which.min(eucldistvector)
```

C.3 Python

```
##########  Erwthma 4a ###########
print("Erwthma 4a")

import numpy as np
import pandas as pd
import math

#%erwthma1
#a
x=np.array([1,2,3,4,5,6])
y=np.array([1,2,3,4,5,6])
print("Euclidean distance between x and y is: ")
eucldist = math.sqrt(sum((x-y)**2))
print(eucldist)

#b
x=np.array([-0.5,1,7.3,7,9.4,-8.2,9,-6,-6.3])
y=np.array([0.5,-1,-7.3,-7,-9.4,8.2,-9,6,6.3])
print("Euclidean distance between x and y is: ")
eucldist = math.sqrt(sum((x-y)**2) )
print(eucldist)

#c
x=np.array([-0.5,1,7.3,7,9.4,-8.2])
y=np.array([1.25,9.02,-7.3,-7,5,1.3])
print("Euclidean distance between x and y is: ")
eucldist = math.sqrt(sum((x-y)**2) )
print(eucldist)

#d
x=np.array([0,0,0.2])
y=np.array([0.2,0.2,0])
print("Euclidean distance between x and y is: ")
eucldist = math.sqrt(sum((x-y)**2) )
print(eucldist)

##########  Erwthma 4b ###########
print("Erwthma 4b")

x=np.array([25000,14,7])
y=np.array([42000,17,9])
z=np.array([55000,22,5])
n=np.array([27000,13,11])
m=np.array([58000,21,13]) #target of comparison

dist=np.array([(math.sqrt(sum((x-m)**2))),(math.sqrt(sum((y-m)**2
                                ))),(math.sqrt(sum((z-m)**2))),(
                                math.sqrt(sum((n-m)**2)))])

print("\n The user that is the most similiar to user with id=5 is
                                ")
print(pd.Series(dist).idxmin() + 1)
```

D Cosine Similarity

D.1 Answers

The cosine similarity formula is cosine similarity $= S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$

A
The cosine similarity between x and y is -0.7739558
B
The cosine similarity between x and y is -0.6546319
C
The cosine similarity between x and y is -0.140921
D
The cosine similarity between x and y is 1

D.2 R

```
        ##########  Erwthma 5 ###########
install.packages(lsa)

graphics.off() ; rm(list = ls(all = TRUE)) ; cat("\014");

library(SnowballC) #essential for lsa
library(lsa) #for cosine similiriaty function

cossim <- function(A,B) { (sum(A*B))/sqrt((sum(A^2))*(sum(B^2))) }

#a
x=c(9.32, -8.3, 0.2)
y=c(-5.3, 8.2, 7)
cossim(x,y)

#b
x=c(6.5, 1.3, 0.3, 16, 2.4, -5.2, 2, -6, -6.3)
y=c(0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3)
cossim(x,y)

#c
x=c(-0.5, 1, 7.3, 7, 9.4, -8.2)
y=c(1.25, 9.02, -7.3, -7, 15, 12.3)
cossim(x,y)

#d
x=c(2, 8, 5.2)
y=c(2, 8, 5.2)
cossim(x,y)
```

D.3 Python

```python
##########  Erwthma 5 ###########
print("Erwthma 5")

import numpy as np
from numpy.linalg import norm


A = np.array([9.32, -8.3, 0.2])
B = np.array([-5.3, 8.2, 7])

cosine = np.dot(A,B)/(norm(A)*norm(B))

print("Cosine Similarity:", cosine)

A= np.array([6.5, 1.3, 0.3, 16, 2.4, -5.2, 2, -6, -6.3])
B= np.array([0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3])

cosine = np.dot(A,B)/(norm(A)*norm(B))

print("Cosine Similarity:", cosine)

A= np.array([-0.5, 1, 7.3, 7, 9.4, -8.2])
B= np.array([1.25, 9.02, -7.3, -7, 15, 12.3])

cosine = np.dot(A,B)/(norm(A)*norm(B))

print("Cosine Similarity:", cosine)

A= np.array([2, 8, 5.2])
B= np.array([2, 8, 5.2])

cosine = np.dot(A,B)/(norm(A)*norm(B))

print("Cosine Similarity:", cosine)
```

E Nominal Distance

E.1 Answers

We define distance between two categorical vectors as 1 when values between vectors of the same column are different
A
Nominal distance between x and y is 3
B
Nominal distance between x and y is 0
C
Nominal distance between x and y is 3

E.2 R

```
        ##########  Erwthma 6 ###########
graphics.off() ; rm(list = ls(all = TRUE)) ; cat("\014");

I interpret distance between two character vectors as 1 when values between
two vectors of the same column are different
nominalDistance <- function(x,y) sum(a==FALSE)

x <-c("Green","Potato","Ford")
y <-c("Tyrian␣purple","Pasta","Opel")
a<-x==y
nominalDistance(x,y)


x <-c("Eagle", "Ronaldo","Real␣madrid" , "Prussian␣blue", "Michael␣Bay")
y <-c("Eagle", "Ronaldo", "Real␣madrid", "Prussian␣blue", "Michael␣Bay")
a<-x==y
nominalDistance(x,y)

x <-c("Werner␣Herzog", "Aquirre,␣the␣wrath␣of␣God", "Audi", "Spanish␣red")
y <-c("Martin␣Scorsese", "Taxi␣driver", "Toyota", "Spanish␣red")
a<-x==y
nominalDistance(x,y)
```

E.3 Python

```
##########  Erwthma 6 ###########
print("Erwthma 6")

import numpy as np

x = np.array(["Green", "Potato", "Ford"])
y = np.array(["Tyrian purple", "Pasta", "Opel"])

nominaldis = np.sum(x!=y)
print(" The distance between the nominal arrays is" , nominaldis)

x = np.array(["Eagle", "Ronaldo", "Real Madrid", "Prussian blue", "
                                  Michael Bay"])
y = np.array(["Eagle", "Ronaldo", "Real Madrid", "Prussian blue", "
                                  Michael Bay"])

nominaldis = np.sum(x!=y)
print(" The distance between the nominal arrays is" , nominaldis)

x = np.array (["Werner Herzog", "Aquirre,the wrath of God", "Audi",
                                   "Spanish red"])
y = np.array (["Martin Scorsese", "Taxi driver", "Toyota", "Spanish
                                   red"])
nominaldis = np.sum(x!=y)
print(" The distance between the nominal arrays is" , nominaldis)
```