Dimensionality Reduction on Australian Athletes Data

Yehya Abdelmohsen

5/14/2022

Contents

Introduction

The data presented contains information for athletes from different types of sports. The purpose of collecting the data was to test if blood hemoglobin levels are different when comparing endurance-related athletes to those in power-related events. In this analysis, we will aim to reduce the dimensionality of the data. We will compare two methods of dimensionality reduction, Principal Component Analysis (PCA) and Multidimensional Scaling (MDS).

About the Data

The data presented contains information related to Australian athletes. It contains 202 observations where each observation represents an athlete. There are 13 variables including the class variable which represents the sport the athlete plays. It's important to note that categorical/qualitative variables will be removed before preforming PCA or MDS.

The data is obtained from R, its documentation is linked below.

Source: https://vincentarelbundock.github.io/Rdatasets/doc/DAAG/ais.html

Variables

- Rcc - Red blood cell count. This is a quantitative variable.
- Wcc - White blood cell count, per liter. This is a quantitative variable.
- Hc - Percent of hematocrit. Hematocrit: The ratio of the volume of red blood cells to the total volume of blood. This is a quantitative variable.
- Hg - Hemaglobin concentration in g per decaliter (10 liters). Hemaglobin: A protein that carries oxygen to organs. This is a quantitative variable.
- Ferr - Plasma ferritins in ng. Measures the amount of ferritin in blood. Ferritin is a blood protein that contains iron. This is a quantitative variable.
- Bmi - Body mass index, in kg/m^2. Body mass divided by the square of the height m^2. This is a quantitative variable.
- Ssf - Sum of skin folds. Estimates the percentage of body fat by measuring skin fold thickness. This is a quantitative variable.
- PcBfat - Percentage of body fat. This is a quantitative variable.
- Lbm - Lean body mass in kg. Total body weight minus body fat weight. This is a quantitative variable.
- Ht - Height in cm. This is a quantitative variable.
- Wt - Weight in kg. This is a quantitative variable.
- Sex - A factor representing the sex of the athlete: female and male. This is a qualitative categorical variable.
- Sport - A factor representing the sport the athlete plays B Ball, Field, Gym, Netball, Row, Swim, T 400m, T Sprnt, Tennis, and W Polo. This is a qualitative categorical variable and will be used as the class variable.

Below is an initial look at the data.

rcc	wcc	hc	hg	ferr	bmi
3.96	7.5	37.5	12.3	60	20.56
4.41	8.3	38.2	12.7	68	20.67

ssf	pcBfat	lbm	ht	wt	sex	sport
109.1	19.75	63.32	195.9	78.9	f	B_Ball
102.8	21.30	58.55	189.7	74.4	f	B_Ball

Problem Statement

The data contains quantitative variables we will be utilizing to reduce the dimension of the data. The aim of this analysis:

- Reducing the dimensionality of the data.
- Comparing PCA and MDS.

Some Graphs

In Figure 1 we can see some obvious linear relationships. For example, red blood cell count (rcc) and hematocrit (hc) are linearly related. This is an obvious relationship when one knows that hematocrit is the ratio between red blood cell volume and the total blood volume. Another example is rcc and hemoglobin (hg), which is also obvious since hg is a protein that is inside red blood cells. The final obvious relationship is between hc and hg.

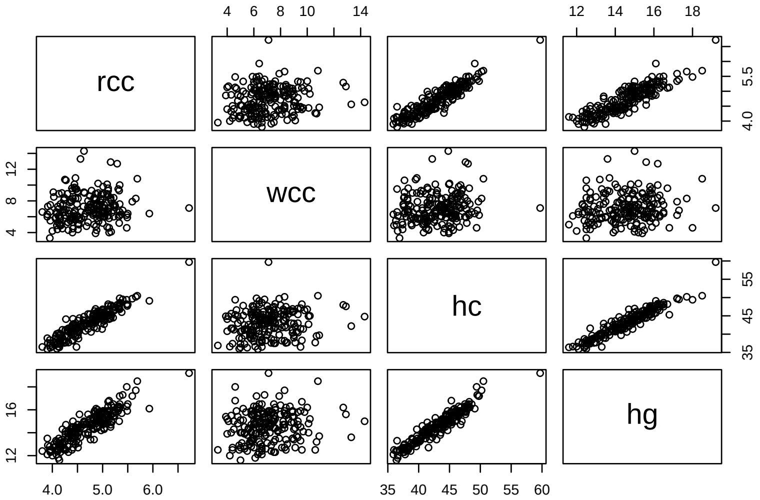


Figure 1: Pair Plot

In Figure 2, we can see one obvious linear relationship between sum of skin folds (ssf) and percentage of body fat (pcBfat). Sum of skin folds estimates the percentage of body fat by measuring skin fold thickness.

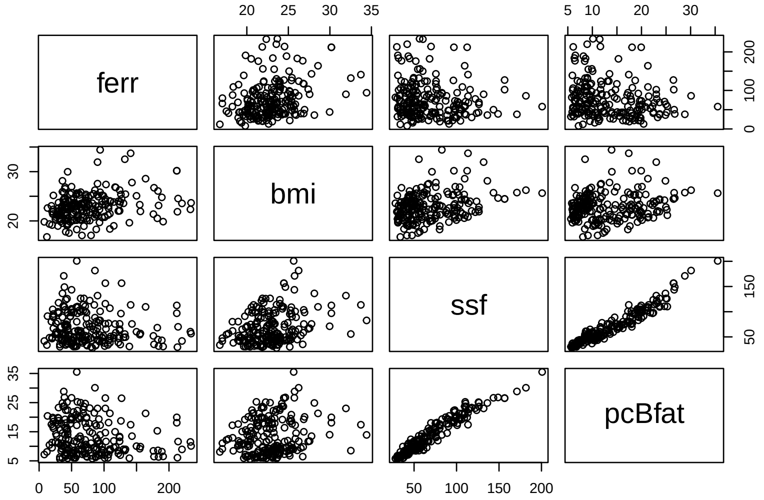


Figure 2: Pair Plot

In Figure 3, we can see some linear relationships between the variables lean body mass, height, and weight. These relationships are obvious and self explanatory.

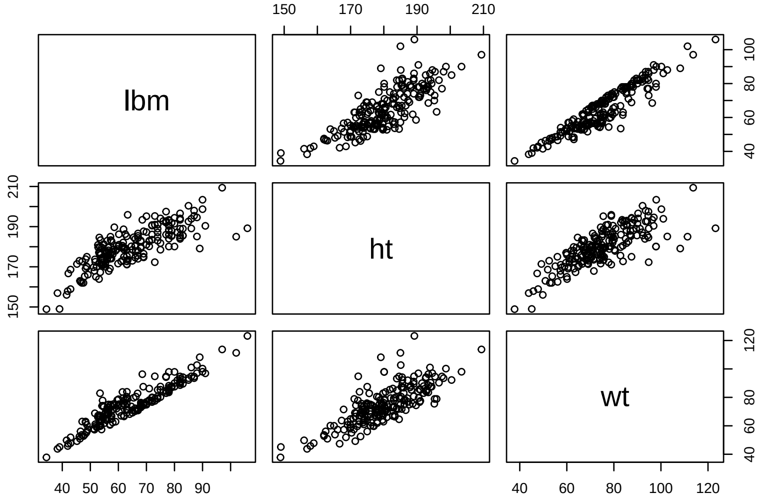


Figure 3: Pair Plot

Looking at the plots in this section we find many linear relationships. This foreshadows the success of dimensionality reduction.

Principal Component Analysis

Non Robust

We scale the data and compute the variance-covariance matrix.

```
scaled_df = scale(df,center=TRUE,scale=TRUE)
cov = cov(scaled_df)
eig = eigen(cov)
```

We also compute the eigen values and eigen vectors of the variance-covariance matrix. The eigen values are shown below.

x
4.9909730
2.5575670
1.1574070
0.8891508
0.7953127
0.4339165
0.1051614
0.0409358
0.0231917
0.0052992
0.0010849

The cumulative sum of total variance explained by the principal components is shown below. Taking the first seven principal components would retain 99.35% of the total variance.

45.37248	68.62309	79.14497	87.22816	94.45828	98.40297	99.35898	99.73113	99.94196	99.99014	100

Again, the scree plot shows that taking the first seven PCs retains a good proportion of the variance. Anything after the seventh principal component will not add too much information.

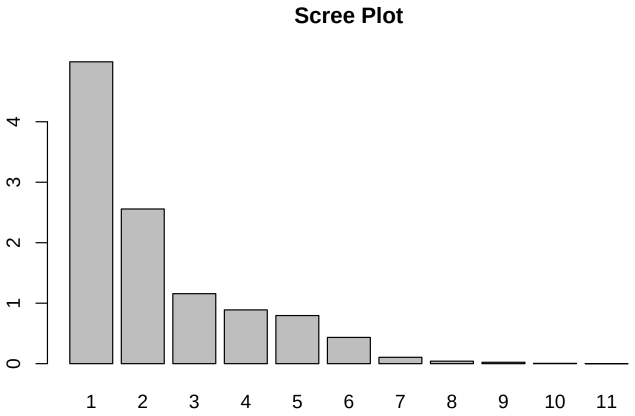


Figure 4: Scree Plot

Below is a look at the new data after preforming PCA and removing the columns corresponding to the lowest eigen values.

```
PC = scaled_df%*%eig$vectors
knitr::kable(head(PC[,1:7],4))
```

2.065050	-1.962587	-1.4162272	-0.0839674	-1.5955587	-1.1088389	-0.1832927
1.900859	-1.517835	-0.3000126	-0.2511224	-1.2494861	-1.1309932	0.3540632
3.384047	-1.169796	-1.5031027	-0.2167857	0.2367684	0.0100873	0.6216680

2.675610	-2.070809	-1.1216785	0.2644651	0.2676794	-1.0229438	-0.0105673

Robust

We use BACON to remove the outliers and do robust PCA. Out of 202 observations, we find 8 outliers.

```
library(robustX)
out = mvBACON(scaled_df)
```

```
## rank(x.ord[1:m,] >= p  ==> chosen m =  44
## MV-BACON (subset no. 1): 44 of 202 (21.78 %)
## MV-BACON (subset no. 2): 176 of 202 (87.13 %)
## MV-BACON (subset no. 3): 188 of 202 (93.07 %)
## MV-BACON (subset no. 4): 191 of 202 (94.55 %)
## MV-BACON (subset no. 5): 194 of 202 (96.04 %)
## MV-BACON (subset no. 6): 194 of 202 (96.04 %)
```

```
robust_df = scaled_df[out$subset,]
```

We then compute the eigen values and eigen vectors of the covariance matrix:

```
eig_pr = eigen(cov(robust_df))
knitr::kable(eig_pr$values)
```

x
4.4542607
2.3109161
1.1214097
0.8723374
0.7051871
0.4003692
0.1044178
0.0383006
0.0234153
0.0032072
0.0007035

Looking at the scree plot I would make the same decision, which is taking the first 7 principal components.

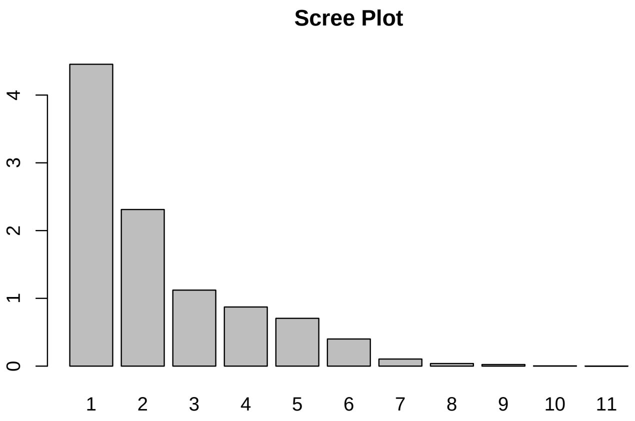


Figure 5: Scree Plot

The cumulative sum of total variance explained by the robust principal components is shown below. Taking the first seven principal components would retain 99.345% of the total variance. This is lower than non robust PCA by around 0.005%.

```
knitr::kable(t(cumsum(eig_pr$values*100/sum(eig_pr$values))))
```

44.38935	67.41901	78.59452	87.28788	94.31549	98.30541	99.34599	99.72768	99.96103	99.99299	100

Below is a look at the new data after preforming PCA on the robust data and removing the columns corresponding to the lowest eigen values.

```
robustPC = robust_df%*%eig_pr$vectors
knitr::kable(head(robustPC[,1:7],4))
```

2.256563	-1.5210834	-1.4166174	0.2065501	1.7062673	-1.2293247	-0.2246019
2.044288	-1.2123511	-0.3017187	0.3190992	1.2378429	-1.2470244	0.3218135
3.528823	-0.5605968	-1.5524897	0.1313583	-0.0004124	-0.0190025	0.6125051
2.921200	-1.5883969	-1.2840223	-0.3849154	-0.0830151	-1.0112437	-0.0215970

Multidimensional Scaling

Non Robust

Computing B:

$$B = XX^T$$

```
B = scaled_df%*%t(scaled_df)
```

Getting the eigen values and eigen vectors of B:

```
eig_B = eigen(B)
```

Computing W:

$$W = V_{11}\Lambda_{11}^{1/2}$$

```
W = eig_B$vec[,1:11]%*%diag(sqrt(abs(eig_B$val[1:11])))
```

Now we look at the scree plot to investigate the amount of variance retained by the components. We can see similar results to PCA, and I think choosing seven columns would be best.

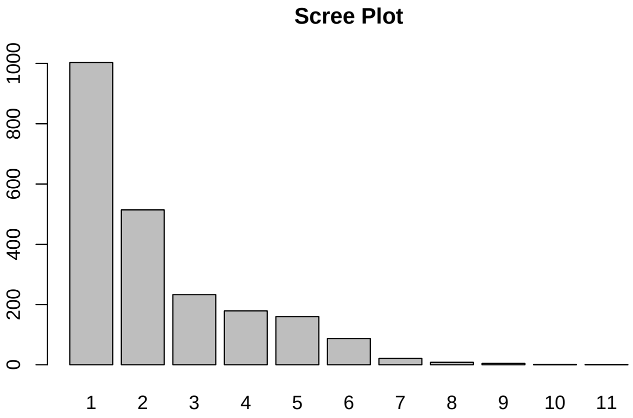


Figure 6: Scree Plot

Also, looking at the cumulative sum we obtain the same results as PCA.

```
##  [1]  45.37248  68.62309  79.14497  87.22816  94.45828  98.40297  99.35898
##  [8]  99.73113  99.94196  99.99014 100.00000
```

Therefore, we will be choosing the first seven components from W. Below are the resulting columns.

-2.065050	-1.962587	1.4162272	-0.0839674	1.5955587	-1.1088389	0.1832927
-1.900859	-1.517835	0.3000126	-0.2511224	1.2494861	-1.1309932	-0.3540632
-3.384047	-1.169796	1.5031027	-0.2167857	-0.2367684	0.0100873	-0.6216680
-2.675610	-2.070809	1.1216785	0.2644651	-0.2676794	-1.0229438	0.0105673

Robust

Computing the robust B:

```
Brobust = robust_df%*%t(robust_df)
```

Computing eigen values of the robust B:

```
eig_Br = eigen(Brobust)
knitr::kable(eig_Br$values[1:11])
```

x
860.0027735
446.0209816
216.4320834
168.3741934
136.7232346
77.3045547
20.1550362
7.3920278
4.5203264
0.6325584
0.1376045

Computing the robust W:

```
Wrobust = eig_Br$vec[,1:11]%*%diag(sqrt(abs(eig_Br$val[1:11])))
```

Now we look at the scree plot to investigate the amount of variance retained by the components. We can see similar results to PCA, and I think choosing seven columns would be best.

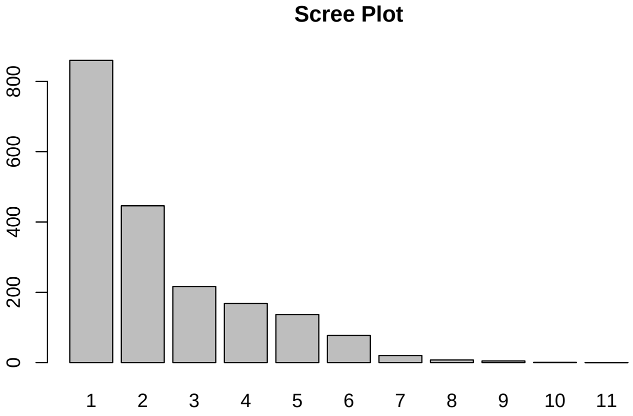


Figure 7: Scree Plot

Also, looking at the cumulative sum we obtain the same results as the robust PCA.

```
##  [1]  44.38276  67.40088  78.57044  87.25985  94.31582  98.30533  99.34548
##  [8]  99.72697  99.96025  99.99290 100.00000
```

Therefore, we will be choosing the first seven components from the robust W. Below are the resulting robust columns.

-2.257534	1.5208741	1.4165750	-0.2015677	1.7090093	-1.225051	0.2238034
-2.045000	1.2122563	0.3016887	-0.3154493	1.2408463	-1.243865	-0.3224169
-3.528693	0.5611977	1.5524981	-0.1316818	-0.0022212	-0.018497	-0.6126237
-2.921054	1.5888286	1.2840232	0.3847206	-0.0829576	-1.011064	0.0213724

Conclusion

In conclusion, there is no drastic difference between doing dimensionality reduction with the robust data and with all the data. This is because the number of outliers in the data accounts for around 4%. Therefore, we reached the same conclusions from robust and non robust dimensionality reduction.

We also conclude that the two methods, PCA and MDS provide very similar results. The components obtained from MDS and those obtained from PCA provide identical PCs but with opposite signs. Also, the first $p = 11$ eigen values of the B matrix in MDS and the matrix X in PCA are not identical. However, an interesting observation is that when we multiply the eigen values of X by $(n - 1)$ where $n = 202$ in our case, we obtain the eigen values of B. Conversely, if we divide the eigen values of the B matrix by $(n - 1)$, we obtain the eigen values of X. An example of the first case is provided below.

```
n=202
eig$values*(n-1)
```

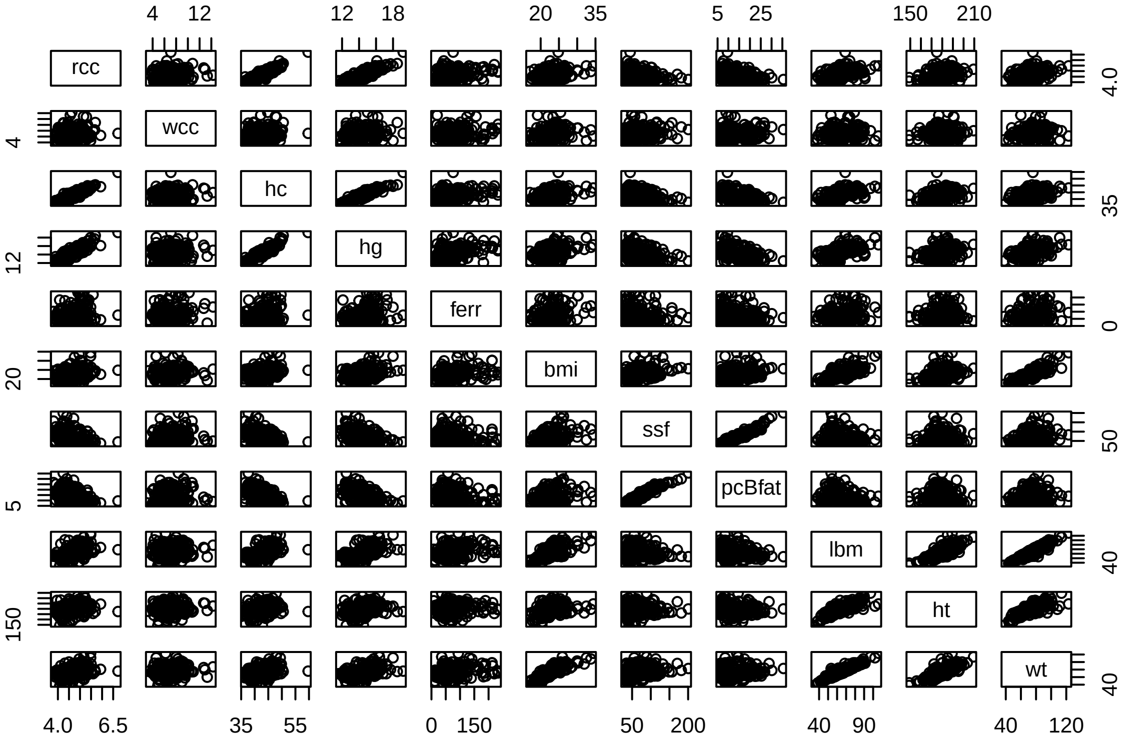
```
##  [1] 1003.1855633  514.0709626  232.6388048  178.7193119  159.8578573
##  [6]   87.2172157   21.1374360    8.2281016    4.6615379    1.0651454
## [11]    0.2180635
```

```
eig_B$values[1:11]
```

```
##  [1] 1003.1855633  514.0709626  232.6388048  178.7193119  159.8578573
##  [6]   87.2172157   21.1374360    8.2281016    4.6615379    1.0651454
## [11]    0.2180635
```

Appendix

```
pairs(df)
```


rcc
wcc
hc
hg
ferr
bmi
ssf
pcBfat
lbm
ht
wt
4
12
12
18
20
35
5
25
150
210
4.0
35
0
50
40
40
4
12
20
5
150
4.0
6.5
35
55
0
150
50
200
40
90
40
120