Principal Component Analysis at a Glance

Pedram Doroudchi

1 Introduction

PCA is a resourceful unsupervised machine learning method for multivariate data analysis. It helps us to study a data set of quantitative variables measured on a set of objects in an efficient yet powerful manner. More technically, PCA offers an optimal low-dimensional representation of variation in our data. This is done by finding vectors in a real inner product space that are statistically uncorrelated, or orthogonal in linear algebra terms (Mei, 2009). Given a full-rank covariance or correlation matrix, the Real Spectral Theorem ensures that we will find an orthonormal basis of eigenvectors since such matrices are real-valued and symmetric. PCA also offers a great dual representation of objects' proximities and variables' correlations.

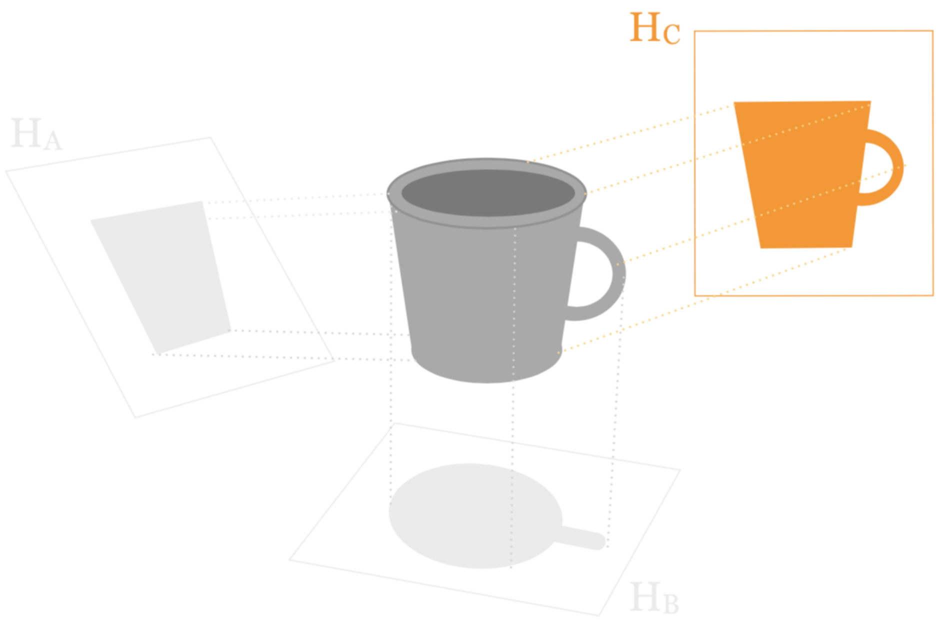


Figure 1: Potential projection subspaces H_A, H_B, H_C. Clearly H_C captures enough variation to decipher the form of the mug while H_A and H_B do not (Sanchez & Marzban, 2020).

Essentially, PCA solves the problem of finding a low-dimensional subspace that best fits our cloud of data. This is done by computing principal components, orthogonal linear combinations of the data variables. It is helpful to think about a three-dimensional cloud of data shaped like a coffee mug, as in Figure 1 above. If we wanted to find a meaningful two-dimensional representation of our coffee mug, we would want to project it onto a two-dimensional plane that maintains the figure of the handle, thus preserving enough variation to decipher it is in fact a mug. This is exactly the goal of PCA and it may be extended to multi-dimensional vector spaces, or multivariate data. Now that we have a good grasp of the idea, let's explore how it's done.

2 PCA

2.1 Projection

For the purposes of this exposition, assume we are dealing with a mean-centered data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ such that we have n observations of p variables. Now with the p-dimensional data cloud in mind, we know it has some centroid $\mathbf{g}$, which could be thought of as the average observation. We now want to project all of our data points onto the ideal one-dimensional subspace, or a single axis, of our data cloud. This axis must run through the centroid of our cloud since it acts as the origin of our vector space. Suppose we take a unit vector $\mathbf{v_k}$ along this axis originating at $\mathbf{g}$. To orthogonally project our observations onto this axis, we would find the scalar projections of our points on $\mathbf{v_k}$.

Recall the scalar projection of observation $\mathbf{x_i}$ onto $\mathbf{v_k}$ is defined by:

$$comp_{\mathbf{v_k}}(\mathbf{x_i}) = \frac{\mathbf{x_i}^\mathsf{T}\mathbf{v_k}}{||\mathbf{v_k}||} = \mathbf{x_i}^\mathsf{T}\mathbf{v_k} = z_{ik} \tag{1}$$

where the second equality holds because $\mathbf{v_k}$ is a unit vector.

Notice z is subscripted by ik, meaning it is the i-th individual's k-th principal component coordinate. In other words, it is the i-th individual's scalar projection onto the k-th axis of the orthogonal basis we seek for the data cloud. Now that we have a good idea of how we'll project our vectors, how do we find the best subspace?

2.2 Maximizing Projected Inertia

Remember that we want a subspace representation of our coffee mug that maintains the distances between points in our data cloud, thereby preserving the variance of the data. One way to measure this idea is using the inertia of our data cloud $\mathbf{X}$, defined by:

$$I_{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n} d^2(i,g) \tag{1}$$

which is precisely the average squared distance to the centroid of the data cloud, or the variance assuming our centroid is centered at 0. Thus essentially we want to find the subspace U that maximizes the projected inertia:

$$\max_{U} \left\{ \frac{1}{n}\sum_{i=1}^{n} d_U^2(i,g) \right\} \tag{2}$$

For now, assume $U \subseteq \mathbb{R}$. This amounts to finding our first and most telling principal component. Thus the distance between observation $\mathbf{x_i}$ and the centroid $\mathbf{g}$ is equal to the scalar projection of $\mathbf{x_i}$ onto our unit vector $\mathbf{v_1}$ which runs through centroid $\mathbf{g}$. Then:

$$\frac{1}{n}\sum_{i=1}^{n} d^2(i,g) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x_i}^\mathsf{T}\mathbf{v_1})^2 = \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2 \tag{3}$$

To be clear $\mathbf{v}_k$ becomes $\mathbf{v}_1$ and z_{ik} becomes z_{i1} because we are focusing our attention to finding the first PC. Thus our maximization problem simply becomes:

$$\max_{\mathbf{v_1}} \left\{ \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x_i}^\mathsf{T}\mathbf{v_1})^2 \right\} \quad \text{s.t.} \quad \mathbf{v_1}^\mathsf{T}\mathbf{v_1} = 1 \tag{4}$$

Now assume that we have mean-centered data, meaning $\mathbf{g} = \mathbf{0}$. Thus the projected inertia is simply the variance of our projected data. Then:

$$I_U = \frac{1}{n}\sum_{i=1}^{n} d^2(i,0) = \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2 \tag{5}$$

Using the fact that $\mathbf{z}_1 = (z_{11}, ..., z_{n1})$ and $\mathbf{x_i}^\mathsf{T}\mathbf{v_1} = z_{i1}$ we may rewrite the above as:

$$I_U = \frac{1}{n}\mathbf{z_1}^\mathsf{T}\mathbf{z_1} = \frac{1}{n}\mathbf{v_1}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{v_1} \tag{6}$$

Our maximization problem finally becomes:

$$\max\left\{\frac{1}{n}\mathbf{v_1}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{v_1}\right\} \quad \text{s.t.} \quad \mathbf{v_1}^\mathsf{T}\mathbf{v_1} = 1 \tag{7}$$

The method of Lagrange multipliers will quickly help us find a solution to this problem. Define a Lagrangian by:

$$\mathcal{L} = \frac{1}{n}\mathbf{v_1}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{v_1} - \lambda_1\left(\mathbf{v_1}^\mathsf{T}\mathbf{v_1} - 1\right) \tag{8}$$

Taking the derivative with respect to $\mathbf{v_1}$ and through simple algebraic manipulation we find an incredible solution:

$$\frac{1}{n}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{v_1} = \lambda_1\mathbf{v_1} \tag{9}$$

For simplicity of notation, let $\mathbf{S} = \frac{1}{n}\mathbf{X}^\mathsf{T}\mathbf{X} \in \mathbb{R}^{p\times p}$, which is the covariance matrix of our data since we assumed mean-centered data. Of course, a matrix is symmetric when it is equal to its transpose. Thus our matrix $\mathbf{S}^\mathsf{T} = (\frac{1}{n}\mathbf{X}^\mathsf{T}\mathbf{X})^\mathsf{T} = \frac{1}{n}\mathbf{X}^\mathsf{T}\mathbf{X} = \mathbf{S}$ is symmetric. It follows that $\mathbf{v_1}$ is an eigenvector of our covariance matrix corresponding to eigenvalue λ_1. We will prove later that λ_1 is in fact the projected inertia, or variance, of the first principal component.

Clearly the PCA solution is deeply intertwined with the eigenvalues of our covariance matrix. The Spectral Theorem ensures that every full-rank data matrix has a powerful PCA solution attained by spectral decomposition of the data's covariance matrix. Let's examine $\mathbf{S}$ as the matrix of a self-adjoint operator to arrive at an intuitive proof of one of the most powerful theorems in linear algebra.

2.3 The Adjoint and Self-Adjoint Operators

Definition *(Adjoint).* Suppose $T \in \mathcal{L}(V, W)$. The adjoint of T is the function $T^* : W \to V$ such that

$$\langle Tv, w\rangle = \langle v, T^*w\rangle$$

for all $v \in V$ and $w \in W$.

Given an inner product space, the adjoint is simply a linear map. Notice that $\mathbf{S} \in \mathbb{R}^{p\times p}$ is a square matrix. In other words, it may be represented by a linear operator. Linear operators share a special relationship with the adjoint.

Definition *(Self-Adjoint).* An operator $T \in \mathcal{L}(V)$ is called self-adjoint if $T = T^*$. In other words, T is self-adjoint if and only if $\langle Tv, w\rangle = \langle v, Tw\rangle$ for all $v, w \in V$.

Theorem. Eigenvalues of a self-adjoint operator are real.

Proof. Left as an exercise to the reader.

How do we know whether the operator associated with our symmetric matrix $\mathbf{S}$ is self-adjoint? Let us explore a few more fundamental ideas before arriving at the answer.

Definition *(Conjugate transpose).*

The conjugate transpose of an m-by-n matrix is the n-by-m matrix obtained by interchanging the rows and columns and then taking the complex conjugate of each entry.

Lemma. $\mathcal{M}(T^*)$ is the conjugate transpose of $\mathcal{M}(T)$ assuming orthonormal bases.

Theorem. $T \in \mathcal{L}(V)$ is self-adjoint if and only if $\mathcal{M}(T) = \mathcal{M}(T^*)$ with respect to an orthonormal basis.

Proof. This is an obvious result by the definition of a self-adjoint operator.

Thus we know that the linear operator associated with our matrix is self-adjoint because $\mathbf{S} \in \mathbb{R}^{p \times p}$ is real-valued and symmetric, meaning $\mathbf{S}$ must equal its conjugate transpose. The following theorem will make the proof of the Spectral Theorem more clear.

Theorem *(Self-adjoint operators and invariant subspaces).*

Suppose $T \in \mathcal{L}(V)$ is self-adjoint and U is a subspace of V invariant under T, meaning $u \in U \implies Tu \in U$. Then

(a) $U^\perp = \{v \in V : \langle v, u \rangle = 0 \ \forall \ u \in U\}$ is invariant under T.

(b) $T|_U \in \mathcal{L}(U)$ is self-adjoint.

(c) $T|_U \in \mathcal{L}(U^\perp)$ is self-adjoint.

Proof. Proving (a) first, suppose $v \in U^\perp$. Let $u \in U$. Then

$$\langle Tv, u \rangle = \langle v, Tu \rangle = 0$$

where the first equality holds because T is self-adjoint and the second equality holds because U is invariant under T and because $v \in U^\perp$. Since this equation holds for all $u \in U$, it follows that $Tv \in U^\perp$. Thus $U^\perp$ is invariant under T.

To prove (b), let $u, v \in U$, then

$$\langle (T|_U)u, v \rangle = \langle Tu, v \rangle = \langle u, Tv \rangle = \langle u, (T|_U)v \rangle.$$

Thus $T|_U$ is self-adjoint.

(c) follows from (b) by replacing U with $U^\perp$ and using the proven fact that $U^\perp$ is invariant under T.

These usefal facts will come in handy for the following proof of the Real Spectral Theorem.

Theorem *(Real Spectral).* Suppose $\mathbb{F} = \mathbb{R}$ and $T \in \mathcal{L}(V)$. Then the following are equivalent:

(a) T is self-adjoint.

(b) V has an orthonormal basis consisting of eigenvectors of T.

(c) T has a diagonal matrix with respect to some orthonormal basis of V.

Proof. Suppose (c) holds. Then T has a diagonal matrix with respect to some orthonormal basis of V. A diagonal matrix is equal to its transpose. Furthermore, our diagonal matrix is equal to its conjugate transpose because $\mathbb{F} = \mathbb{R}$. Thus $T = T^*$ and T is self-adjoint, meaning (a) holds.

Now we will prove (a) implies (b) using induction on dim V. The base case when dim $V = 1$ holds because it can be shown that every self-adjoint operator has an eigenvalue and thereby an orthonormal eigenbasis. Now assume dim $V > 1$ and that our hypothesis holds for all real inner product spaces with dimension smaller than that of V. We know $T \in \mathcal{L}(V)$ is self-adjoint. Let u be an eigenvector of T with $||u|| = 1$. Let

$U = span(u)$. Then U is a one-dimensional subspace of V invariant under T. So by the previous theorem $T|_U \in \mathcal{L}(U^\perp)$ is self-adjoint. By our induction hypothesis, there exists an orthonormal basis of $U^\perp$ consisting of eigenvectors of $T|_{U^\perp}$. Adjoining u to this orthonormal basis of $U^\perp$, we have an orthonormal basis of V consisting of eigenvectors of T, thus proving that (a) implies (b).

It is obvious that (b) implies (c) and hence we are done.

Since we know that the linear operator associated with our matrix $\mathbf{S}$ is self-adjoint, we know it is orthogonally diagonalizable. Diagonalizing a symmetric matrix is equivalent to obtaining its eigenvalue, or spectral, decomposition. Thus, assuming full rank, the solution to PCA can always be obtained by performing an eigendecomposition on $\mathbf{S} = \frac{1}{n}\mathbf{X}^\mathsf{T}\mathbf{X} \in \mathbb{R}^{p\times p}$.

2.4 Eigendecomposition of S

Since $\mathbf{S} \in \mathbb{R}^{p\times p}$ is orthogonally diagonalizable, we may represent it as:

$$\mathbf{S} = \frac{1}{n}\mathbf{X}^\mathsf{T}\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\mathsf{T} \tag{1}$$

where $\mathbf{V} \in \mathbb{R}^{p\times p}$ is an orthogonal matrix consisting of the eigenvectors of $\mathbf{S}$ and $\mathbf{\Lambda} \in \mathbb{R}^{p\times p}$ is a diagonal matrix consisting of the eigenvalues of $\mathbf{S}$. Recall that our one-dimensional maximization problem simply amounted to finding the first eigenvector, or loading in PC jargon, of $\mathbf{V}$ and corresponding first eigenvalue of $\mathbf{\Lambda}$. So how do we find the principal component matrix? It's simple. Recall that $\mathbf{x_i}^\mathsf{T}\mathbf{v_k} = z_{ik}$. It follows that our PC matrix $\mathbf{Z} \in \mathbb{R}^{n\times p}$ may be defined by:

$$\mathbf{Z} = \mathbf{X}\mathbf{V} \tag{2}$$

Looking at the k-th PC, or score, we get the equation:

$$\mathbf{z_k} = \mathbf{X}\mathbf{v_k} = v_{1k}\mathbf{x_1} + v_{2k}\mathbf{x_2} + \cdots + v_{pk}\mathbf{x_p} \tag{3}$$

Claim. mean$(\mathbf{z_k}) = 0$

Proof. We assumed mean-centered data. Thus mean$(\mathbf{x_j}) = 0$ for all $j = 1, ..., p \implies$ mean$(\mathbf{z_k}) = 0$ using equation (3) above.

Claim. var$(\mathbf{z_k}) = \lambda_k$

Proof.

$$var(\mathbf{z_k}) = \frac{1}{n}\mathbf{z_k}^\mathsf{T}\mathbf{z_k} \tag{1}$$

$$= \frac{1}{n}(\mathbf{X}\mathbf{v_k})^\mathsf{T}(\mathbf{X}\mathbf{v_k}) \tag{2}$$

$$= \frac{1}{n}\mathbf{v_k}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{v_k} \tag{3}$$

$$= \mathbf{v_k}^\mathsf{T}\mathbf{S}\mathbf{v_k} \tag{4}$$

$$= \mathbf{v_k}^\mathsf{T}\lambda_k\mathbf{v_k} \tag{5}$$

$$= \lambda_k \tag{6}$$

since $\mathbf{v_k}$ was assumed to be a unit vector for all $k = 1, ..., p$.

Thus the k-th eigenvalue of **S** is the variance of the k-th principal component and the sum of all eigenvalues of **S** measures the total inertia, or variability, of our data. Let's turn our attention to some interesting examples that showcase the dimension-reducing power of PCA.

3 Examples and Applications in R

3.1 NBA

In our first example, we will perform PCA from scratch. Our data set involves NBA team statistics for the 2018-19 regular season. You can find the data here. The data is taken straight from the NBA website and contains 30 rows and 28 columns but we will only use 10 specific features in our example.

We begin by importing and tidying our data:

```
# import data
dat <- read.csv('nba-teams-2019.csv', stringsAsFactors = FALSE)

# filter for variables we want to work with
variables <- c('W', 'L', 'PTS', 'FGM', 'X3PM', 'REB',
               'AST', 'TOV', 'STL', 'BLK'
               )

# standardize data matrix X
X <- scale(dat[ ,variables])

# fix row names as NBA team names
rownames(X) <- dat$TEAM

# print first few entries of data frame
head(X[ ,1:5], 5)
```

```
##               W           L        PTS        FGM       X3PM
## ATL -0.99761393  0.99761393  0.5122341  0.1953667  1.0866755
## BOS  0.66507595 -0.66507595  0.2923562  0.6140096  0.8216327
## BRK  0.08313449 -0.08313449  0.2434944 -0.4625007  0.9541541
## CHO -0.16626899  0.16626899 -0.1229688 -0.5223069  0.3578078
## CHI -1.57955539  1.57955539 -1.5399598 -0.7615314 -1.4974918
```

```
# compute correlation matrix S
n <- nrow(X)
S <- (1/(n-1)) * t(X) %*% X

# eigendecomposition of S
EVD <- eigen(S)

# diagonal matrix of eigenvalues of S
D <- diag(EVD$values)

# matrix of loadings/eigenvectors
V <- EVD$vectors
rownames(V) <- colnames(X)
```

```
colnames(V) <- paste0('V', 1:ncol(V))
head(V[ ,1:5], 5)
```

```
##              V1          V2          V3          V4         V5
## W     0.4060038  0.29891549 -0.14260733  0.02637302 -0.1723575
## L    -0.4060038 -0.29891549  0.14260733 -0.02637302  0.1723575
## PTS   0.4072402 -0.01061588  0.08175797 -0.05629045  0.4664931
## FGM   0.3736246 -0.16367180  0.20553866  0.35560648  0.3875309
## X3PM  0.2184656  0.24649676 -0.34906618 -0.65429465  0.2222588
```

```
# matrix of PCs
Z <- X %*% V
rownames(Z) <- rownames(X)
colnames(Z) <- paste0('PC', 1:ncol(Z))
head(Z[ ,1:5], 5)
```

```
##            PC1        PC2        PC3        PC4         PC5
## ATL  0.2231280 -2.5397419  0.5089382 -2.2126979  0.89751288
## BOS  1.5022961  0.2729577 -1.5730956  0.7690020  0.17175446
## BRK -0.1708175  0.8519063  1.1147358 -1.6601551  0.63550810
## CHO -0.8322882  1.5375297 -0.7145734  0.5285939 -0.11428099
## CHI -3.5934103 -0.4726772  0.3408941  0.6109442 -0.02423137
```

```
# eigenvalues
eigenvalues <- EVD$values
eigenvalues
```

```
##  [1]  4.574094e+00  1.581170e+00  1.191798e+00  1.000163e+00  6.600746e-01
##  [6]  3.862735e-01  3.543119e-01  1.968092e-01  5.530558e-02 -9.665814e-16
```

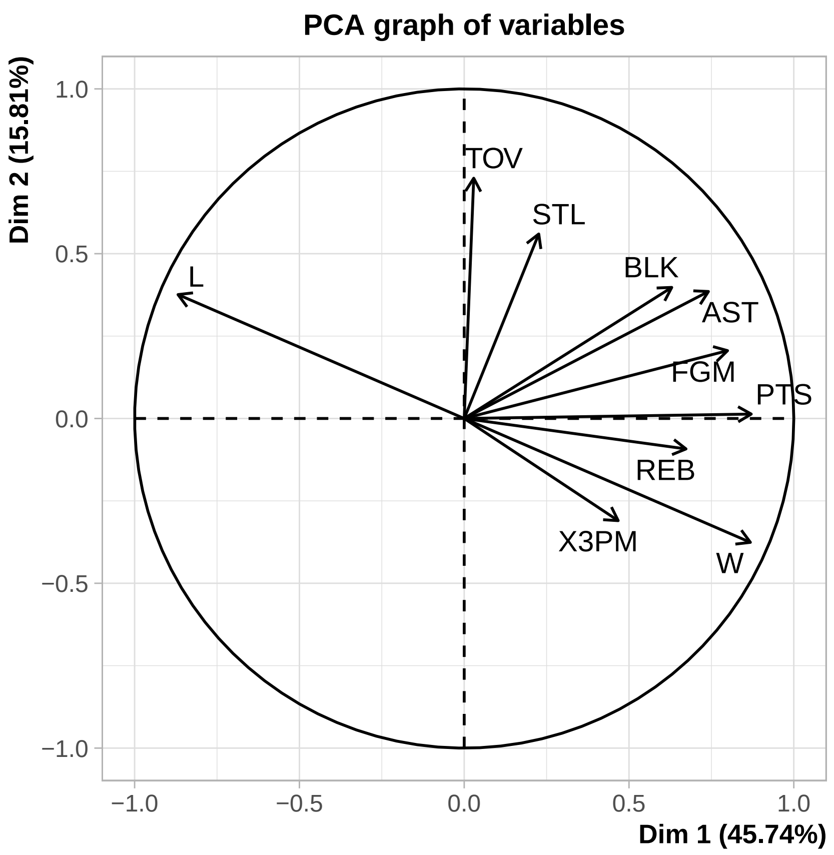
```
# compute sum
sum(eigenvalues)
```

```
## [1] 10
```

Recall that the eigenvalues above measure the variance of each PC. Clearly our first PC carries the majority of variance, which is quite common among datasets. This is also a great example of why PCA is a dimension-reduction method, because we could do away with the tenth PC here without losing virtually any information. More about that in the following example. Of course, we would expect the sum of our eigenvalues to equal 10 because this is precisely the total variance of our dataset. Recall that we standardized our data matrix, meaning each variable was scaled to have unit variance. Thus our solution above is precisely what the theory proclaims.

Now that we have computed the PCs, how could we further interpret their meaning and relation to the original variables? Plotting a Circle of Correlations with respect to the first two principal components will help us in this regard:

```
# plot circle of correlations
pca <- PCA(X, graph = FALSE)
plot(pca, choix = 'var')
```



The Circle of Correlations above represents graphically how correlated each data feature is with respect to the first and second PCs. For instance, W and PTS are positively correlated with PC1 while logically L is the mirror opposite of W. We see that TOV is positively correlated with PC2 while PTS is practically uncorrelated with PC2. It seems then that PC1 measures a team's offensive ability while PC2 perhaps measures it's defensive strength. In the following example we will look at how to determine which principal components to keep and which to do away with.

3.2 Decathlon

We'll now look at a new PCA visualization example involving decathlon data from the R package "FactoMineR" (Le, Josse, & Husson, 2008).

We will begin by creating a data frame of ten decathlon events from the 2004 Athen Olympic Games. As promised, we will use the "prcomp" function that is preinstalled within the R library package being loaded below. For brevity, the necessary code to produce the PCA componenets is included but its output will not be printed.

```
# load package and dataset
library(FactoMineR)
data(decathlon)

# first ten decathlon events only
dat <- decathlon[ ,1:10]

# perform PCA with prcomp()
pca_prcomp <- prcomp(dat, scale. = TRUE)

# list elements of the above object
```

```r
names(pca_prcomp)

# create vector of eigenvalues
evalues <- (pca_prcomp$sdev) ^ 2

# create matrix of loadings
loadings <- pca_prcomp$rotation

# create matrix of principal components
scores <- pca_prcomp$x

# Z = XV sanity check (score matrix = data * loadings)
z <- scale(as.matrix(dat)) %*% loadings
```

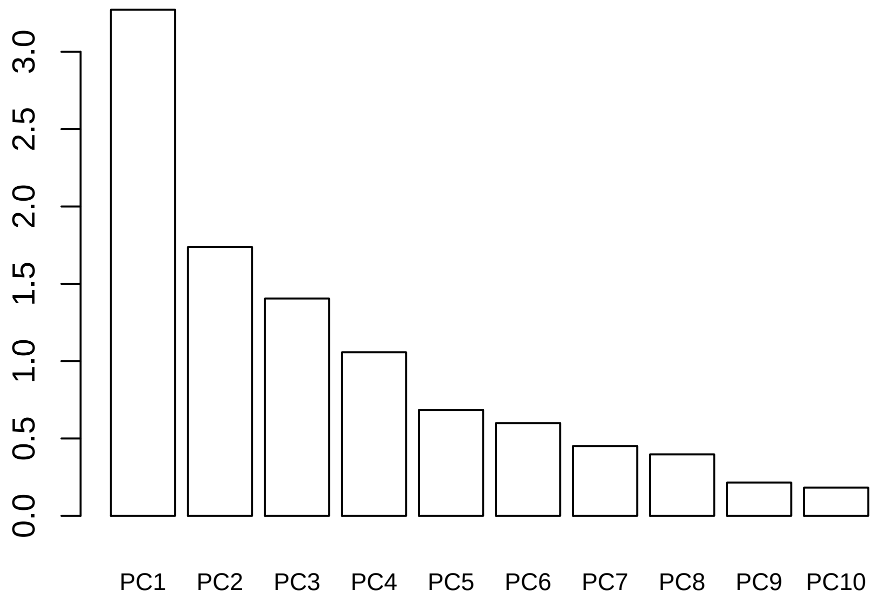
Now that we have our PCs, how many should we hold on to? It helps to make a bar plot to compare the proportion of variance explained by each PC, like so:

```r
# create percentage vector (multiply by 10 because total variance is 10)
percentage <- evalues * 10

# create cumulative percentages vector
cumulative.percentage <- c()
for (i in 1:10) {
cumulative.percentage[i] <- sum(percentage[1:i])
}

# combine into table
df <- data.frame(evalues, percentage, cumulative.percentage)

# create plot
barplot(evalues, names.arg = colnames(loadings), density = 0, cex.names = 0.75)
```



Perhaps we could determine a cutoff so that we keep enough PCs to retain 70% of the variance of the dataset. Another option is that we could do away with PCs that explain less than the average PC does. The Kaiser Rule is the most commonly used approach and suggests to keep PCs with eigenvalues greater than or equal to 1.0. How many PCs would you retain with respect to the bar plot above?

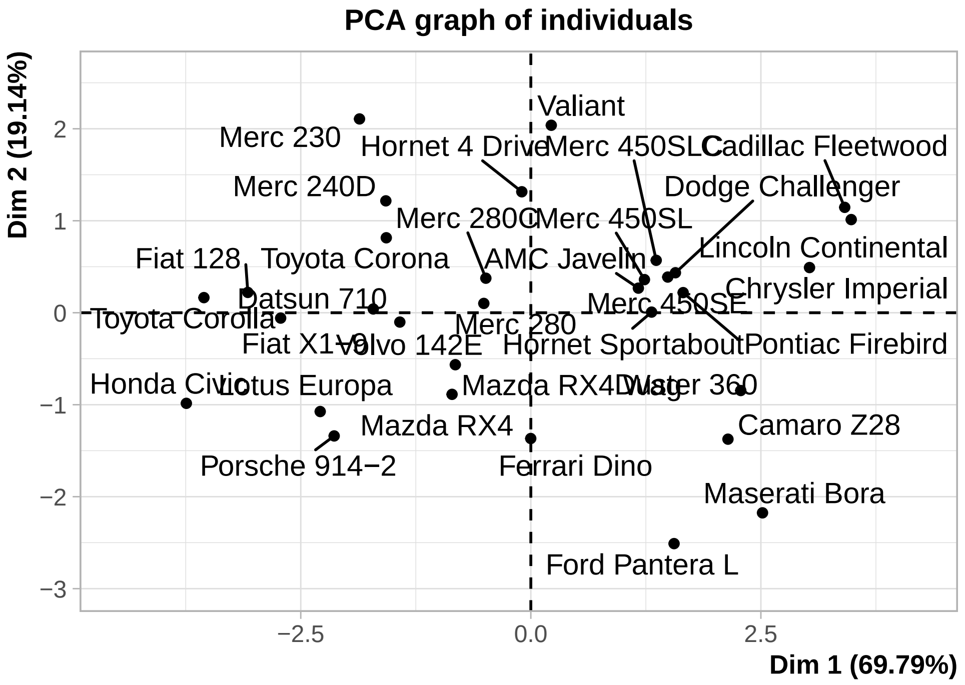
In our final example, we will use the "mtcars" dataset included in R to visualize our observations in a low-dimensional space with respect to our first two PCs.

3.3 Empty Cars

We examine the dataset "mtcars" found within R. The data was extracted from the 1974 *Motor Trend* US magazine. It comprises fuel consumption and other aspects of automobile design and performance for 32 '73-74 model automobiles. Again, necessary preparation of the dataset will be left as an exercise to the reader for brevity. Keep in mind that PCA works with quantitative, not categorical, variables. We will always want to filter our dataset appropriately so that we're left only with quantitative variables.

The "prcomp" function makes PCA visualization easy:

```r
# plot scatterplot with respect to first two PCs
pca <- PCA(X, graph = FALSE)
plot(pca)
```



The scatterplot above presents an effective and efficient way to visualize and compare our data's observations. The first PC, capturing almost 70% of the variance of the data, perhaps seeks to explain the sheer power of the vehicle. Big-bodied cars with plenty of horsepower like the Pontiac Firebird have positive scores with respect to this first dimension while more compact cars like the Toyota Corolla and Honda Civic have negative scores. It's interesting to note that vehicles with a positive PC1 score are primarily American brands while those with a negative score are mostly Japanese and European makes. Thus PCA verifies the old "American muscle" adage. What do you think the second PC seeks to convey?

4 Conclusion

In this brief exposition we've examined a host of inventive methods PCA has to offer in order to better visualize and understand quantitative data. This paper was intended to highlight the fundamental reasoning behind PCA and its importance with respect to large datasets. It cannot be overstated that PCA offers an elegant solution to reducing redundant datasets that are becoming more common as the advancement of modern-day technology permits.

5 References

[1] Mei, M. (2009). *Principal Component Analysis.* The University of Chicago. Retrieved from http://www.math.uchicago.edu/~may/VIGRE/VIGRE2009/REUPapers/Mei.pdf

[2] Sanchez, G. & Marzaban, E. (2020). *All Models Are Wrong: Concepts of Statistical Learning.* Retrieved from https://allmodelsarewrong.github.io/

[3] Axler, S. (2015). *Linear Algebra Done Right* (Third Edition). Springer International Publishing.

[4] Le, S., Josse, J., & Husson, F. (2008). *FactoMineR: An R Package for Multivariate Analysis.* Journal of Statistical Software, 25(1), 1-18. 10.18637/jss.v025.i01

[5] R Core Team (2020). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. Retrieved from https://www.R-project.org/.