Post01

Changhua Yu

10/23/2017

Exploring the math behind the Principle Component Analysis's and its potential in scientific researches

Intro:

> Caged in a database that stores millions of genetic information for every single cell in your body, you are trying to find out the tiny sub-population that causes the malfunctioning of the body. You feel like an ant in a Möbius strip, and there's simply no way to get a clear image. Until: you find magic of dimensionality reduction, and the intertwined threads floating around are straightened, flattened and ordered.

> In reality, this magic is the statistical tool for dimensionality reduction and clustering, which include the Principle Component Analysis Method that I'm going to introduce. In this post, I will delve deeply into the methodology behind the PCA analysis, and try to explore its potential of processing data and specifically, genomic data, in the context of R language.

Motivation:

> Last summer, I interned at Yale University's stem cell center and joined a research group focusing on the functionality of neutrophils, the most common immune cells in human bodies. As a bioengineering major student, I always try to explore more on the interdisciplinary sides of biology that intersect with math and engineering, but I realized the importance of statistics in molecular biology for the first time. The group was trying to utilize RStudio for PCA analysis on the single-cell RNA sequencing results to identify sub-populations of cell based on their performance of genomic features. However, all the researchers in that lab were focusing on physiologic concentrations and did not have time to explore a totally new area of knowledge, and so the lab design was postponed.

> I almost have forgotten about this experience since then, until we went over the PCA basics during stat133 and created a real example in the homework. Thus, I decided to focus on a further investigation on the mathematic and statistic knowledge behind PCA analysis and some of its application in current biology researches.

Audience:

> Although this post focuses on applications of PCA in biological researches as I regard it as a great opportunity to explore my personal interests, it stills aims to present a general introduction on PCA's real life application and mathematic deduction to all my classmates who may not have a biology background. Terminology will be generally avoided, simplified, or fully explained, and I hope everyone can enjoy this post as an additional real life exposure to PCA in order to facilitate a better understanding of the knowledge.

What is PCA: *MATH* AND BACKGROUND

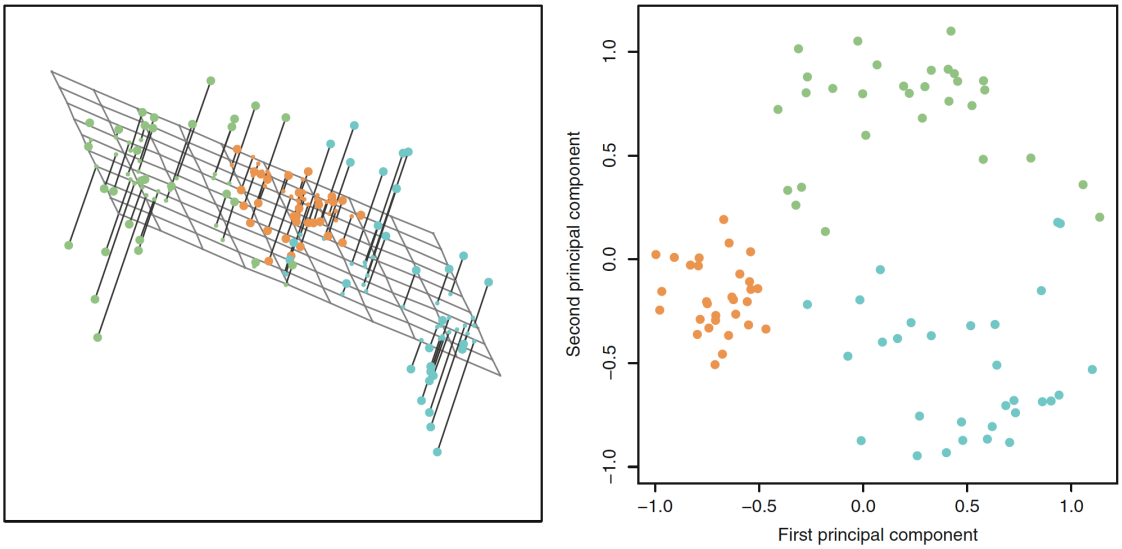
PCA aims to retained the variation displayed in the original data as much as possible in a lower dimension.

> Suppose that we have 100 samples tested on 1000 independent variables and their corresponding values were collected in an assay. When we attempt to find the difference between each of the 100 samples, we need a brute-force approach that examines 1000 bar plots generated from each single variable. Further comparison between each sample is hard to visualize and may lead to confusing conclusions. Alternatively, we may generate linear combinations of the 1000 axes, and then select the three most interesting axes to draw a 3D map, and the coordinate of each point is their values mapped to x, y, and z axis.[11]
>
> Indeed, the first principle component is generated following two rules:
> 1. Find a linear combination $z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \ldots + \phi_{p1}x_{ip}$ that has the largest variance Sum(Var(z_{i1})), where i is the total number of samples (from 1 to 100 in our case). p is the number of variables that each represents an axis, and p is 1000 in our case [7] 2. Normalize the sum of the square of z_{i1} to 1, in order to

avoid unreasonably large variance.

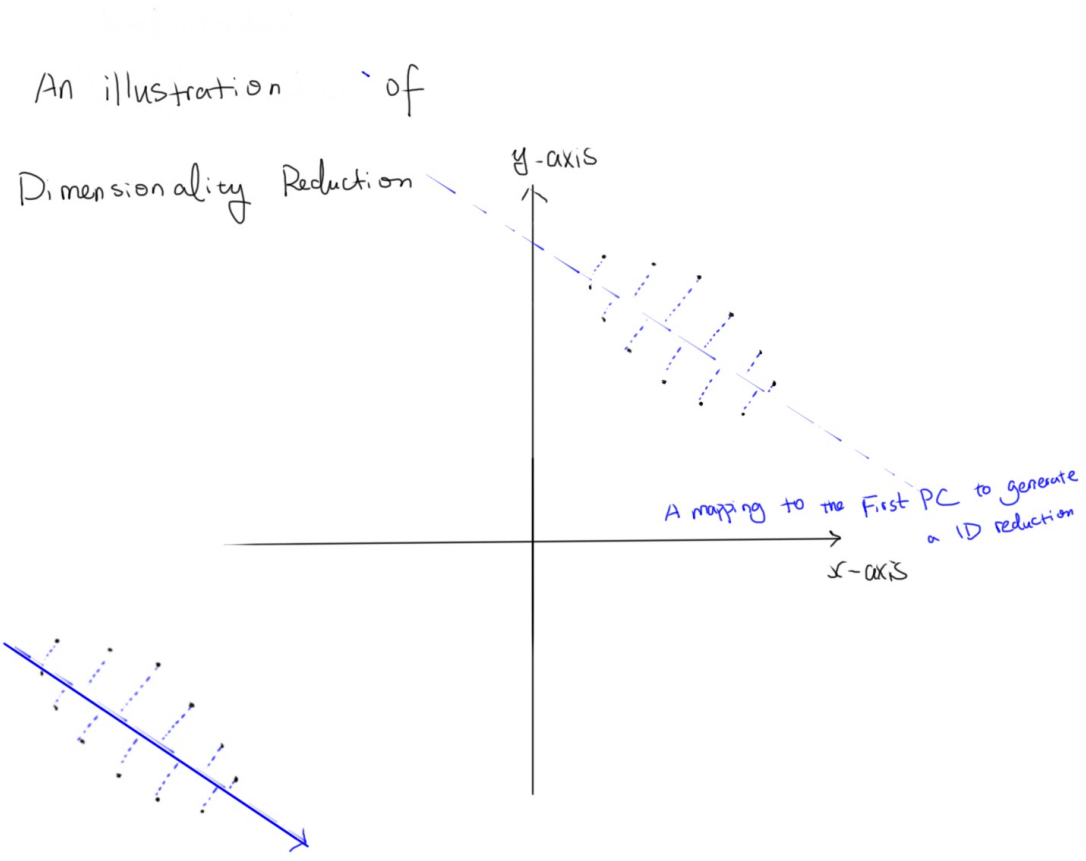
The ϕ_{i1} are called the loadings of PC1, and they altogether define a direction in the given space that the data vary the most. The z_{i1} are called the scores of each sample with respect of PC1. The second PC is constructed in a similar way; however, it is constrained to be uncorrelated with PC1, which should be an axis orthogonal to PC1 (the x, y, and z axes are orthogonal and thus uncorrelated to each other, for instance). This goal is achieved by a matrix manipulation method that we are going to cover later in the post. [8,10]

The first n PCs form the best fitting n-dimension surface that is closest to the original observations



3D demonstration of PCA mapping (source: The machine learning book)

As displayed in the figure above, the points in a 3-D space are being mapped to a plane formed by PC1 and PC2, where the Euclidian distance between the point and its projection onto the plane is minimized. In such way, the 2-D plot formed by the scores of the first two PCs is the most accurate capture of the variance of the original data, and the 2-D plot is more convenient for comparison, clustering and further analysis, while remaining most of the original dataset's information.



Hand-drawn PCA mapping demonstration in 2-D space

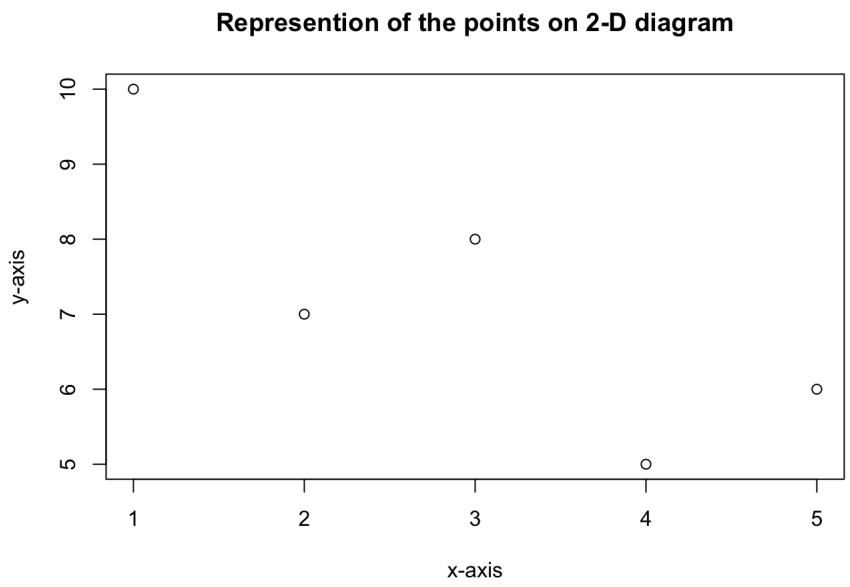
> Although further exploration on the topic requests some extra knowledge in linear algebra, I simplified the case into 2-D occasions and the examples can be easily visualized and understood by everyone in the class. For people who struggle with the idea of“eigen”, the eigenvalues and eigenvectors can be easily calculated by R built in function without further knowledge. All you need to know is that *the eigenvalues represent the variance covered by the corresponding eigenvector, and the eigenvectors are orthogonal to each other in order construct uncorrelated axes*. Thus, we select the first n eigenvectors with the largest eigenvalues to map the higher dimension values onto a lower dimension surface (n is 2 and the surface is a line in our case) that minimize the differences between the original data and mapped scores

> *Given my limiting latex skills, a handwritten version of mathematics deduction of the examples that expands on the eigenvalue calculation is attached below if you are interested in the detailed matrix manipulation.*

> The M*t(M) operation generates a symmetric matrix that has orthogonal eigenvectors capable of constructing PCs. Each eigenvector contains the loadings of PC in order to build our axes of PC, and the eigenvalues are the variance covered by the corresponding PC. The first several PCs are being selected to construct a lower dimension plot with less axes than before, and each point is mapped by its score on each PC. [8,9]

> To make this process more understandable, I prepare 2 examples that reduce two dimensional data into one dimension: Take an normal 2-D example to aid understanding:

```
# load the ggplot2 package for later use
library(ggplot2)

# create a matrix that contains 5 points in a 2-D space
row_name = c('point 1','point 2','point 3','point 4','point 5')
dat = matrix(c(1,2,3,4,5,10,7,8,5,6), nrow = 5,ncol = 2)

#Visualize the dat on a 2-D scatter plot
plot(dat,xlab = 'x-axis',ylab = 'y-axis',main = 'Represention of the points on 2-D diagram')
```




```
# Get a new matrix by multiplying dat_transposed and dat for eigenvector calculations
dat_transposed = t(dat)
eig_prepare = dat_transposed %*% dat

# inspect the new data
eig_prepare
```

```
##      [,1] [,2]
## [1,]   55   98
## [2,]   98  274
```

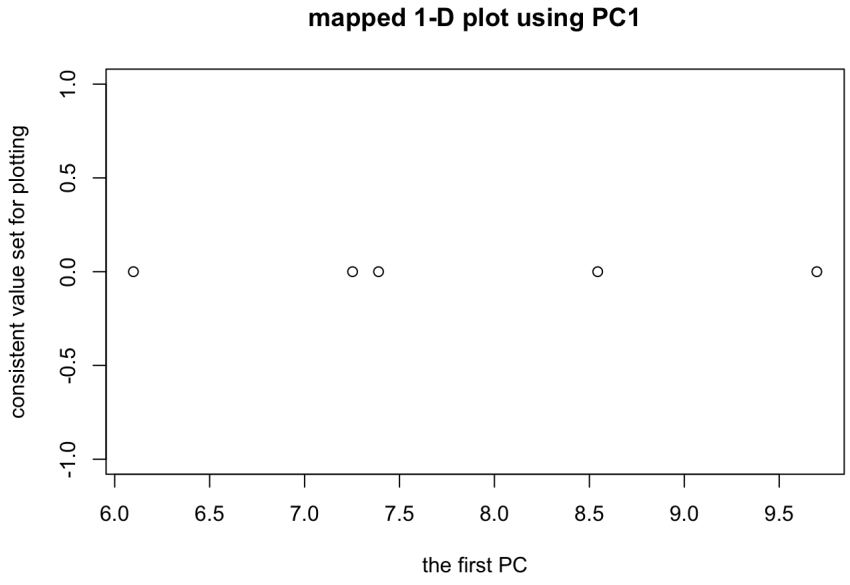
```r
# find out the eigenvalues and corresponding eigenvectors using eigen() command
eig = eigen(eig_prepare)
#inspect eig
eig
```

```
## eigen() decomposition
## $values
## [1] 311.44982  17.55018
##
## $vectors
##           [,1]       [,2]
## [1,] 0.3569648 -0.9341179
## [2,] 0.9341179  0.3569648
```

```r
# assign eigenvalues and eigenvectors respectively
eigenvalues = eig$values
eigenvectors = eig$vectors

# As the eigenvectors and values are displayed in decreasing order, we may pick the first PC with the first column
PC1 = eigenvectors[,1]

# The transformed data can be calculated by mapping each point onto the first PC
transformed_data = dat %*% PC1

# set the 1D points with a uniform y value 0 to align them on a single axis
transformed_data_plot = cbind(transformed_data,matrix(rep(0,5),ncol = 1,nrow = 5))

# The transformed matrix for plotting
transformed_data_plot
```

```
##          [,1] [,2]
## [1,] 9.698143    0
## [2,] 7.252754    0
## [3,] 8.543837    0
## [4,] 6.098448    0
## [5,] 7.389531    0
```

```r
#  Plot the mapped 1-D plot with only one axis PC1
plot(transformed_data_plot,xlab = 'the first PC', ylab = 'consistent value set for plotting',main = 'mapped 1-D plot using PC1')
```



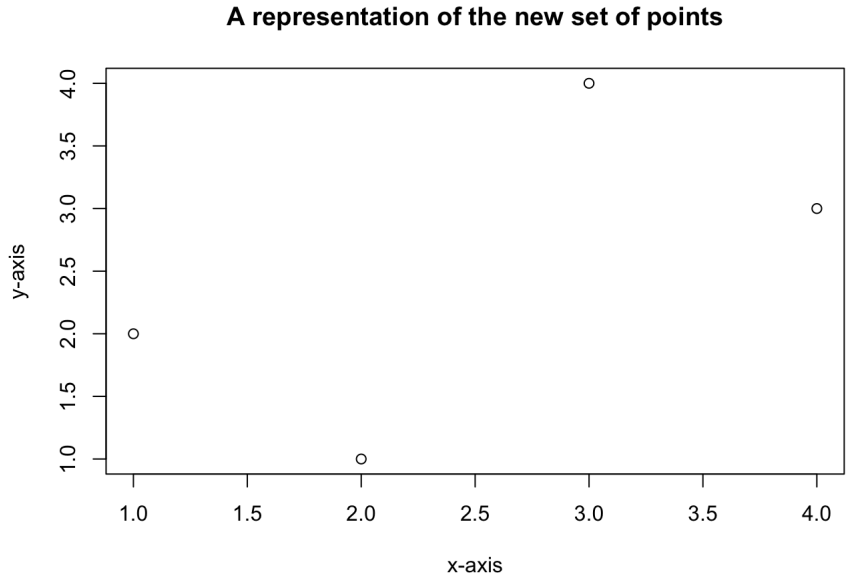
> In the first dataset, five points that are relatively spread out are being mapped onto the line that minimize the Euclidian distance between the original data and the mapped points.

> Take an abnormal 2-D example to aid understanding:

```
# create a matrix that contains 5 points in a 2-D space
dat_new = matrix(c(1,2,3,4,2,1,4,3), nrow = 4,ncol = 2)

# inspect the new dat matrix, notice that the points are symmetical to an axis
dat_new
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    2    1
## [3,]    3    4
## [4,]    4    3
```

```
plot(dat_new,xlab = 'x-axis',ylab = 'y-axis',main = 'A representation of the new set of points')
```




```
# Get a new matrix by multiplying dat_transposed and dat for eigenvector calculations following similar command
dat_transposed_new = t(dat_new)
eig_prepare_new = dat_transposed_new %*% dat_new
eig_prepare_new
```

```
##      [,1] [,2]
## [1,]   30   28
## [2,]   28   30
```

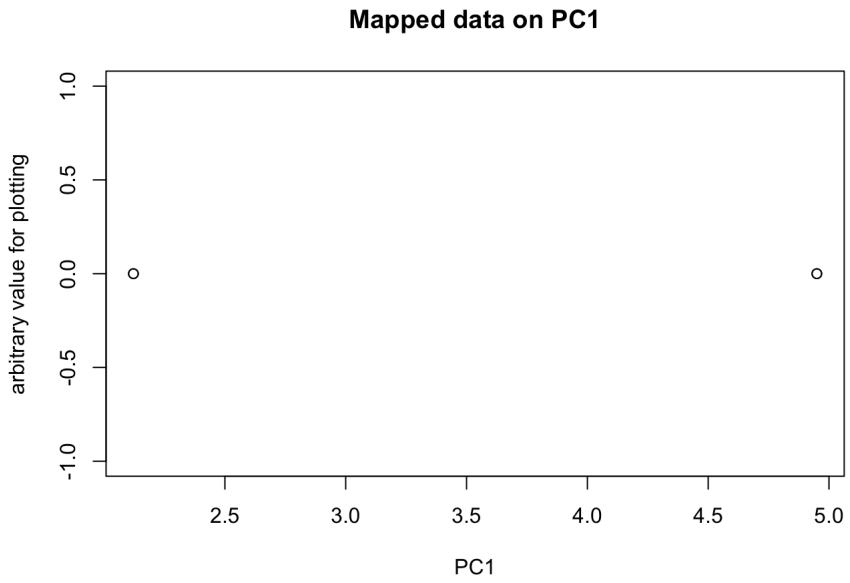
```
# find out the eigenvalues and corresponding eigenvectors using eigen() command as described above
eig_new = eigen(eig_prepare_new)
eig_new
```

```
## eigen() decomposition
## $values
## [1] 58  2
##
## $vectors
##           [,1]       [,2]
## [1,] 0.7071068 -0.7071068
## [2,] 0.7071068  0.7071068
```

```
eigenvalues_new = eig_new$values
eigenvectors_new = eig_new$vectors

# As the eigenvectors and values are displayed in decreasing order, we may pick the first PC with the first column
PC1 = eigenvectors_new[,1]

# The transformed data can be calculated by mapping each point onto the first PC
transformed_data_new = dat_new %*% PC1

# set the 1D points with a uniform y value 0 to align them on a single axis
transformed_data_plot_new = cbind(transformed_data_new,matrix(rep(0,4),ncol = 1,nrow = 4))

# inspected the mapped matrix for plotting
transformed_data_plot_new
```

```
##           [,1] [,2]
## [1,] 2.121320    0
## [2,] 2.121320    0
## [3,] 4.949747    0
## [4,] 4.949747    0
```

```
# Plot the mapped 1-D plot with only one axis PC1
plot(transformed_data_plot_new, xlab = 'PC1',ylab = 'arbitrary value for plotting',main = 'Mapped data on PC1')
```



> In the second example, an abnormal dataset that contains four points with a symmetrical axis between them were mapped onto that axis, and as we expected, the symmetrical points are mapped onto the same point, which further confirm the efficiency of this dimension reduction method, in that not all the variance are being accounted but the most significant variance are augmented. [11]

Built-in PCA Implementation in R

> After the ideal 2-dimensional examples that aim to delve deeper into the mathematical and geometric meaning of PCA method, I going to perform the common way of doing PCA in R by the R built-in dataset 'Iris', which is also related to the biological researches that contains the length and width of petal and sepal of difference species of iris flower. Please go over the chunk below with the commented instruction. [1] Using R build-in function to proceed PCA:

```
# a quick view on the built-in dataset iris
head(iris,5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

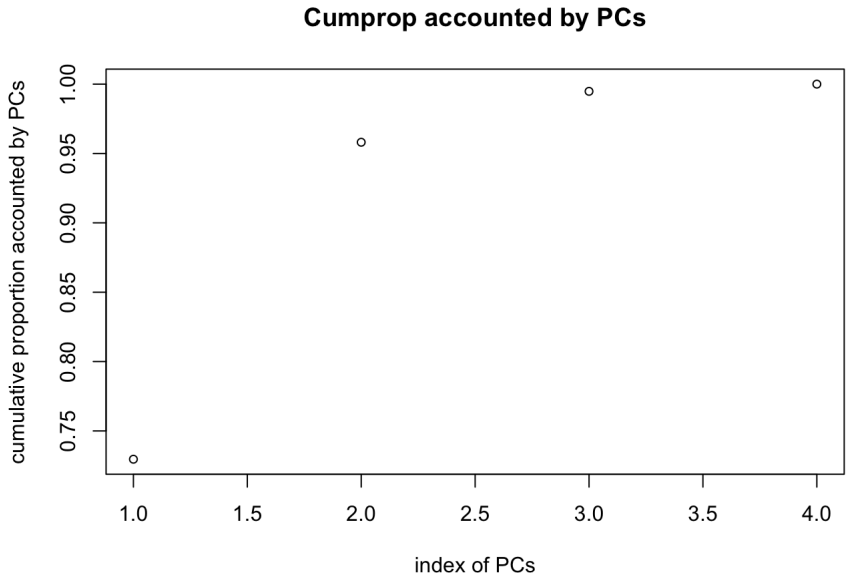
```
# using the prcomp command to do PCA on a R build-in dataset
dat_2 = iris[, 1:4]

#Apply PCA using prcomp()
pca = prcomp(dat_2,scale. = TRUE)
typeof(pca)
```

```
## [1] "list"
```

```
# record the eigenvalue, the proposition accounted and cummulative proposition in a data from
eigs = data.frame(eigenvalue = round(pca$sdev^2,4),prop = round(pca$sdev^2/sum(pca$sdev^2),4),cumprop = round(cums
um(pca$sdev^2/sum(pca$sdev^2)),4))

# display the data frame
eigs
```

```
##   eigenvalue   prop cumprop
## 1     2.9185 0.7296  0.7296
## 2     0.9140 0.2285  0.9581
## 3     0.1468 0.0367  0.9948
## 4     0.0207 0.0052  1.0000
```

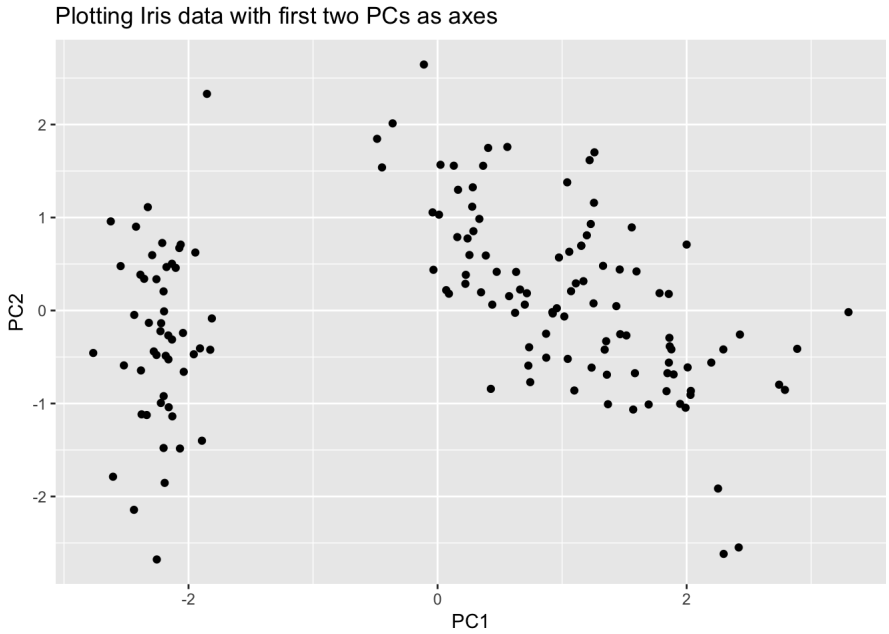
```
# plot the cumprop graph of the variance accumulatively accounted by Principle components
plot(eigs$cumprop,cex = 0.8,xlab = 'index of PCs',ylab = 'cumulative proportion accounted by PCs', main = 'Cumprop accounted by PCs')
```




```
# Use the first two PCs to get a scatterplot of the teams

# Get the loadings of the first two PCs
loadings <- data.frame(pca$rotation[,c(1,2)])
# inspect loadings
head(loadings,5)
```

```
##                     PC1         PC2
## Sepal.Length  0.5210659 -0.37741762
## Sepal.Width  -0.2693474 -0.92329566
## Petal.Length  0.5804131 -0.02449161
## Petal.Width   0.5648565 -0.06694199
```

```
# Get the mapped scores of the points onto the first two PCs
scores = data.frame(pca$x[,c(1,2)])

# plot the mapped points on the 2-D plot with PC1 and PC2 as the two orthogonal axis
pca_plot = ggplot(scores,aes(x= PC1,y = PC2),) + geom_point(stat = 'identity') + ggtitle("Plotting Iris data with first two PCs as axes")

pca_plot
```



We apply 'prcomp', one of the most common R PCA analysis tool to the dataset. The command returns a list that contains several fields of calculated parameters. We square the sdev to get variance, which are the eigenvectors, and I show that the first PC accounts for the most of the variance and then each PC gradually decreases in the cumulative proportion diagram. Finally, we apply only the first two PC to visualize the transformed data in a x-y plot. In the result, we can clearly observe the grouping and separation patterns of difference data clusters, and characterizes the subpopulations within our samples. Through this PCA analysis, the data can be better visualized, represented, and classified.

Example of real-world PCA analysis using an online gene dataset

The next example is an adoption of the genetic informatics workshop by Harvard University that I found interesting and understandable. The workshop used commands not covered in class for data processing and plotting, and so I adjust those while using the same data as it did for PCA analysis. Preprocessing of the genomic data were included in a separate R.script file, in which low gene cells and cells with a shallow sequencing depth (insufficient amount of data collected) were filtered out. [2,6] Those are exactly the same as the workshop's page, and since it's out of the scope of my post, I will focus on the PCA analysis on the processed data instead. After applying the built-in PCA command following the process described above, we can make a scatterplot that maps the higher dimension RNA sequencing data (the amount of different genes in each cell) onto a plane for better visualization. A Case Study using RNA sequencing data

```
getwd()
```

```
## [1] "/Users/changhua/Desktop/Stat133/stat133-hws-fall17/post01/report"
```

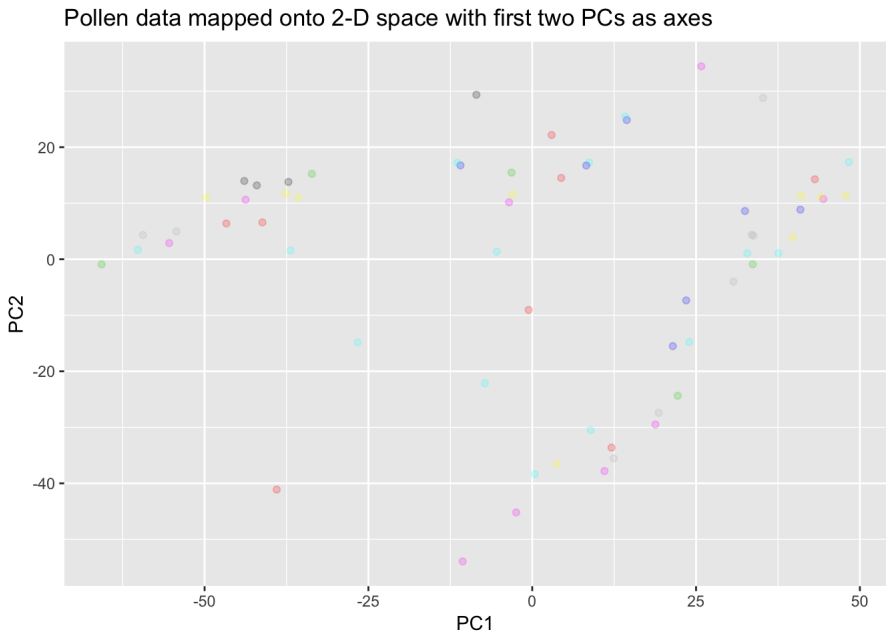
```
# use principal component analysis for dimensionality reduction
load("../data/processed_pollen.rda")

# bind the preprocessed matrix to the current environment
mat = mat

# apply the built-in pcs method prcomp to the transposed matrix
base.pca <- prcomp(t(mat))

#Use the first two PCs to get a scatterplot
loadings <- data.frame(base.pca$rotation[,c(1,2)])
#inspect the loadings
head(loadings,5)
```

```
##                PC1          PC2
## A1BG  -0.004177120 -0.001524068
## A2M   -0.001648788 -0.003317099
## A2MP1  0.001902913  0.001920840
## AAAS  -0.006980808 -0.005015325
## AACS   0.001330823 -0.001183051
```

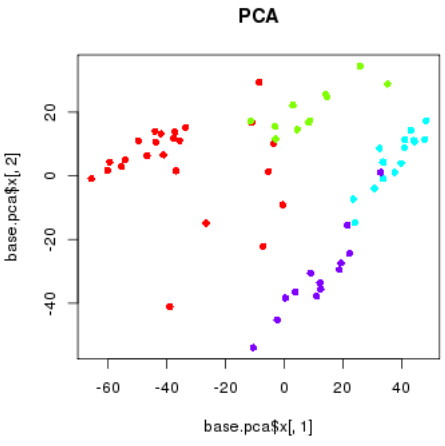
```r
# get the scores of each point with respect to the first two PCs
scores = data.frame(base.pca$x[,c(1,2)])
# inspect the scores
head(scores,5)
```

```
##                 PC1      PC2
## Hi_NPC_1  -37.20525 13.79578
## Hi_NPC_10 -42.02341 13.18660
## Hi_NPC_14 -35.70278 11.03455
## Hi_NPC_15 -43.93825 13.96919
## Hi_NPC_13 -37.56301 11.72726
```

```r
# The the points with a reduced dimension map that only contain 2 axes PC1 and PC2
pca_plot = ggplot(scores,aes(x= PC1,y = PC2)) + geom_point(stat = 'identity' ,color= scores[,2]+100,alpha = 0.2) +
ggtitle('Pollen data mapped onto 2-D space with first two PCs as axes')
pca_plot
```



> The group did a further coloring based on different cell types, and we can see that our mapping is able to highlight most of the distinct cell subpopulations. [10]



The processed plot with colors indicating cell types (source:'https://hms-dbmi.github.io/scw/analysis-of-heterogeneity-and-subpopulations.html')

Discussion & Conclusions

> In this post, we go over the basic math behind PCA, how it is computed both arithmetically and geometrically, and how to implement it using R built-in function. Furthermore, we delve into the real life example in current scientific research to gain a better understanding of how it is used.

> From identifying cell's abnormality in cancer, to visualize the trajectory of neuron-cell development, the PCA method holds a great potential in the field of life science research. [3,4] It should be noted that the method's accuracy relies on the sufficient amount of data collected, a challenge that is gradually conquered by

> advancement in microfluidic equipment, robotics and a lot of other techniques. Some other methods including tSNE, and k-mean clustering is also often applied in further clustering analysis. [4,5] Finally, the post shows that all the subjects are interrelated in various ways, in that the beauty of biology is revealed by statistical methods

References

Dataset used:

```
 1. Pollen AA, Nowakowski TJ, Shuga J, Wang X, Leyrat AA, Lui JH, Li N, Szpankowski L, Fowler B, Chen P, Ramalinga
m N, Sun G, Thu M, Norris M, Lebofsky R, Toppani D, Kemp DW 2nd, Wong M, Clerkson B, Jones BN, Wu S, Knutsson L, A
lvarado B, Wang J, Weaver LS, May AP, Jones RC, Unger MA, Kriegstein AR, West JA Nat Biotechnol. 2014 Oct; 32(10):
1053-8.

 2. R Built-in Iris
```

The Application in Biological Science

```
 3. Abraham G, Inouye M (2014) Fast Principal Component Analysis of Large-Scale Genome-Wide Data. PLoS ONE9(4): e9
3766.

 4. Fan, Jean et al. “Characterizing Transcriptional Heterogeneity through Pathway and Gene Set Overdispersion Ana
lysis.” Nature methods 13.3 (2016): 241—244. PMC. Web. 29 Oct. 2017.

 5. R package for analyzing single-cell RNA-seq data   https://github.com/hms-dbmi/scde
```

Algebra and deduction of PCA

```
 6. Harvard Single Cell Workshop https://hms-dbmi.github.io/scw/analysis-of-heterogeneity-and-subpopulations.html


 7. Laura Diane Hamilton's Intro to PCA http://www.lauradhamilton.com/introduction-to-principal-component-analysis
-pca

 8. Stanford Infolab Textbook on PCA http://infolab.stanford.edu/~ullman/mmds/ch11.pdf

 9. A tutorial on Principal Components Analysis by Lindsay Smith http://www.cs.otago.ac.nz/cosc453/student_tutoria
ls/principal_components.pdf

 10. https://cran.r-project.org/web/packages/matlib/vignettes/eigen-ex1.html

 11. G. James et al., An Introduction to Statistical Learning: with Applications in R, Springer Texts in Statistic
s, DOI 10.1007/978-1-4614-7138-7
```


Appendix: Handwritten datailed matrix operations

A Handwritten Go through of whats happening in R

- Step One

we have five points (1, 10), (2, 7), (3, 8), (4, 5), (5, 6) that are displayed in a 5x2 Matrix

$$M = \begin{bmatrix} 1 & 10 \\ 2 & 7 \\ 3 & 8 \\ 4 & 5 \\ 5 & 6 \end{bmatrix}$$

- Step Two

we then calculate $M^T \times M$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 10 & 7 & 8 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 10 \\ 2 & 7 \\ 3 & 8 \\ 4 & 5 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 55 & 98 \\ 98 & 274 \end{bmatrix}$$

STEP THREE

we then calculate the eigenvalues and corresponding eigenvectors by

$$\det \left| \begin{bmatrix} 55-\lambda & 98 \\ 98 & 274-\lambda \end{bmatrix} \right| = 0$$

$$(35-\lambda)(274-\lambda) - 98^2 = 0$$

$\lambda_1 = 311.4\ldots$ $\lambda_2 = 17.55\ldots$

STEP FOUR

we thus pick the largest eigenvalue's eigenvector as the 1st PC

$$\begin{bmatrix} 0.3569\ldots \\ 0.934\ldots \end{bmatrix}$$

And Transform the data by mapping with 1st PC

$$\begin{bmatrix} 1 & 10 \\ 2 & 7 \\ 3 & 8 \\ 4 & 5 \\ 5 & 6 \end{bmatrix} \times \begin{bmatrix} 0.3569\ldots \\ 0.934\ldots \end{bmatrix} = \begin{bmatrix} 9.6\ldots \\ 7.2\ldots \\ 8.5\ldots \\ 6.09\ldots \\ 7.38\ldots \end{bmatrix}$$

we reduce 1 dimension by using the first PC :) !