**Principal Components Analysis (PCA)**

**What is PCA?**

PCA transforms a set of correlated variables into a new, smaller set of uncorrelated variables. These new variables are known as *principal components* (PCs). The PCs are formed by finding linear combinations of the original variables.

For example, with the cereal data, perhaps we could represent the adjusted sugar, fat, and sodium variables by ONLY

y = -0.667sugar – 0.5879fat + 0.4575sodium

without really losing much information about the data. We will examine later if this is true.

Objectives of PCA

PCA is used primarily as an exploratory technique to be followed up with other analyses. The objectives of PCA are to

1. Discover the “true dimension” of the data

If our p original variables can be represented well with just d principal components (d < p) without losing much information, then do it!

The word “dimension” is often used with this to essentially mean number of variables. For example, if we have 10 variables, it is difficult to view graphically all 10 variables simultaneously in a scatter plot (10D or 10-dimensional scatter plot). However, if we can find 3 PCs that essentially represent the 10 original variables without much loss of information, we can view the PCs with just a 3D scatter plot!

1. Try to interpret the PCs (“new” variables)

The PCs are linear combinations of the “original” variables. The weights for each of the original variables may give meaning to the PCs. For example, a weight of 0 could mean that a particular original variable is not important to the PC.

Finding interpretations for the PCs often is VERY difficult.

Uses of PCA

* Data screening

Although plotting can help to identify outliers, it may be difficult when there are MANY variables. PCA will help view the data with a smaller number of variables that may help identify outliers.

* Clustering

Determine how to group like observations. More formal ways are discussed later in our course.

* Predict classifications

When classifications of the observations are already known, PCA can be used to help develop ways to predict classifications. More formal ways are discussed later in our course.

* Regression analysis

When explanatory (independent) variables are highly correlated with each other, “intercorrelation” or “multicollinearity” is said to exist. This can cause estimates of the β’s to be “unstable” (meaning that from sample-to-sample-to-sample, the ’s have a lot of variability). Therefore, interpretation of how an explanatory and response (dependent) variable are related by examining the  may not give good results.

**PCA with the covariance matrix Σ**

Characteristics of the PCs:

* Uncorrelated
* First principal component accounts for as much variability in the data as possible
* Each successive principal component accounts for as much variability in the data as possible (but always less than the previous PC)

Let **x** ~ (**μ**, **Σ**) where **x** is p×1. Notice that we do not use a multivariate normal distribution assumption.

First principal component

 = a11(x1-μ1) + a21(x2-μ2) + … + ap1(xp-μp)

where **a**1 = [a11, …, ap1]′ is a p×1 vector chosen so that Var[] is maximized over all vectors **a**1 with a length of 1.

The maximum value of the variance is the largest eigenvalue of **Σ**, λ1. The variance occurs when **a**1 is the corresponding eigenvector of **Σ**. For a proof, see Johnson and Wichern’s textbook.

Because the variance of  is being maximized, the new variable y1 will explain as much variability of **x** as possible.

Why are we concerned about explaining variability? In statistics, we often equate “variability” with “information”. The more variability that you understand about a data set, the more information that you know about the data set. Refer back to when you were introduced to ANOVA or regression methods when explaining variability was focused on.

Second principal component



where **a**2 is chosen so that Var[] is maximized over all possible vectors **a**2 with a length of 1 and y1 and y2 are uncorrelated. The uncorrelated part implies then that **a**1 and **a**2 are orthogonal (i.e., ).

The maximum value is λ2 (2nd largest eigenvalue from **Σ**) and it occurs when **a**2 is the corresponding eigenvector of **Σ**.

Third, fourth,… principal component

Found in a similar manner as the first and second PCs.

It can be shown that because the eigenvectors are orthogonal, the PCs are orthogonal. This allows an orthogonal axis system to be used with the PCs

For example, we can use a graph with x, y, and z axes like “usual” if there are three principal components. Later in the course, we will learn about ways to not have orthogonal axes.

Question: How many principal components can one have?

Total variance

The covariance matrix can be written as:



The *total variance* is the sum of the diagonal elements:



This can be thought of as a measure of the total amount of information in the original variables.

From the matrix algebra subsection, remember that . A measure of the importance of the jth principal component is then . Note that this is a proportion and remember that Var(yj) = λj. The larger the value, the more variability the jth principal component accounts for.

Determining the number of PCs

We would like to find the smallest number of PCs such that most of the data set variability is accounted for. The dimensionality essentially corresponds to the number of non-zero eigenvalues.

How can we determine the number of PCs then?

* Account for 100γ% of the variability where 0 < γ < 1

Find the smallest value d such that  > γ, where d is the number of PCs that you choose.

* Examine a scree plot

Plot  vs. 1, 2, …, p. When the points on the plot level off close to 0, the corresponding PCs are probably not contributing too much information to understanding the data. The value of d is the number of eigenvalues before the leveling off occurs.

Example:



Choose d = 2 since the plot levels off at .

Estimation

Because **μ** and **Σ** are usually NEVER known, the corresponding estimates





are used instead in calculations. Therefore, we will from now on use

 for j =1,…,p

where the jth largest eigenvalue of  and its corresponding eigenvector are denoted by  and , respectively.

Interpretation

Elements of  = [, ,…, ] are often compared within  to try to interpret the PCs. For example, suppose  is “large” positive and “large” negative while the other components are approximately 0. The PC can be thought of as measuring the difference between the 1st and 2nd original variable. This may have a meaningful interpretation.

PC scores

What are the actual numerical values of the PCs for each observation?

For each observation, we calculate the jth *principal component value* or *score* as:



for the jth PC and the rth observation.

Example: Goblet data (GobletPCA.R, goblet.csv)

Below is the picture of the measurements from earlier in the notes:



From a past practice problem:

Researchers are interested in grouping goblets that have the same shape although they may have different sizes. One way suggested by Manly and Alberto (2016) is to adjust the data by dividing each measurement by X3 (height).

> goblet <- read.csv("C:\\chris\\goblet.csv")

> head(goblet)

 goblet x1 x2 x3 x4 x5 x6

1 1 13 21 23 14 7 8

2 2 14 14 24 19 5 9

3 3 19 23 24 20 6 12

4 4 17 18 16 16 11 8

5 5 19 20 16 16 10 7

6 6 12 20 24 17 6 9

> goblet2 <- data.frame(ID = goblet$goblet,

 w1 = goblet$x1/goblet$x3,

 w2 = goblet$x2/goblet$x3,

 w4 = goblet$x4/goblet$x3,

 w5 = goblet$x5/goblet$x3,

 w6 = goblet$x6/goblet$x3)

> head(goblet2)

 ID w1 w2 w4 w5 w6

1 1 0.5652174 0.9130435 0.6086957 0.3043478 0.3478261

2 2 0.5833333 0.5833333 0.7916667 0.2083333 0.3750000

3 3 0.7916667 0.9583333 0.8333333 0.2500000 0.5000000

4 4 1.0625000 1.1250000 1.0000000 0.6875000 0.5000000

5 5 1.1875000 1.2500000 1.0000000 0.6250000 0.4375000

6 6 0.5000000 0.8333333 0.7083333 0.2500000 0.3750000

Star plot:

> dev.new(width = 11, height = 7)

> stars(x = goblet2[,-1], draw.segments = TRUE, key.loc =

 c(14,10), main = "Goblet star plot", labels =

 goblet2$ID)



Which goblets appear to stand out? Can you make any generalizations about groups for goblets?

Parallel coordinates plot:

> library(MASS)

> col.w5 <- ifelse(test = goblet2$w5 <= median(goblet2$w5), yes = "red", no = "blue")

> parcoord(x = goblet2[,-1], col = col.w5, main = "Goblet

 parallel coordinate plot")



As mentioned in the partial answers for the practice problems, it appears that a large connection between the base and the cup leads to larger values of w1, w2, w4, and w6 (see highlighted lines above). The same type of highlighting for other variables does not produce similar results.

PCA with the estimated covariance matrix:

> pca.cov <- princomp(formula = ~ w1 + w2 + w4 + w5 + w6,

 data = goblet2, cor = FALSE)

> pca.cov

Call:

princomp(formula = ~w1 + w2 + w4 + w5 + w6, data = goblet2, cor = FALSE)

Standard deviations:

 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5

0.30317016 0.13231201 0.08424389 0.06511770 0.03822006

 5 variables and 25 observations.

There is not much information given by simply printing the object. Next, I examine exactly what information is available inside the object and what method functions are available to summarize this information.

> names(pca.cov) #What's inside pca.cov?

[1] "sdev" "loadings" "center" "scale" "n.obs"

[6] "scores" "call"

> class(pca.cov) #Class of pca.save

[1] "princomp"

> #Method functions for objects of class princomp

> methods(class = princomp)

[1] biplot plot predict print summary

see '?methods' for accessing help and source code

Below is the output from using the summary() function:

> summary(pca.cov, loadings = TRUE, cutoff = 0.0)

Importance of components:

 Comp.1 Comp.2 Comp.3

Standard deviation 0.3031702 0.1323120 0.08424389

Proportion of Variance 0.7520422 0.1432412 0.05806924

Cumulative Proportion 0.7520422 0.8952834 0.95335264

 Comp.4 Comp.5

Standard deviation 0.06511770 0.03822006

Proportion of Variance 0.03469505 0.01195231

Cumulative Proportion 0.98804769 1.00000000

Loadings:

 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5

w1 0.758 0.344 0.378 0.368 0.171

w2 0.460 0.194 -0.200 -0.802 -0.259

w4 0.287 -0.697 0.194 0.146 -0.612

w5 0.362 -0.369 -0.739 0.174 0.397

w6 0.036 -0.472 0.484 -0.411 0.610

> mu.hat <- colMeans(goblet2[,-1])

> mu.hat

 w1 w2 w4 w5 w6

0.7079462 0.9040548 0.7231692 0.3303339 0.3882952

> goblet2[1,]

 ID w1 w2 w4 w5 w6

1 1 0.5652174 0.9130435 0.6086957 0.3043478 0.3478261

* The “Standard deviation” row in the “Importance of components” table gives the standard deviation of each principal component  in the first row. Therefore, we have 
* The “Proportion of Variance” row in the “Importance of components” table gives = /5. Two PCs may be enough because 89.5% of the total variability is accounted for by them. Three PCs would definitely be enough (95.3% of the total variability).
* The “Loadings” table gives the eigenvectors. The “loadings” name comes from the eigenvectors being the coefficients in the linear combination with the data.
* The eigenvectors are orthogonal to each other. For example,  = 0.758\*0.344 + + 0.036\*(-0.472) = 0.
* The eigenvectors are normalized to have a length of 1. For example,  = [0.7582 + + 0.0362]1/2 = 1.

First PC:



What is the second PC?



To calculate the PC scores, we can use the above equations with the variable values. For example, the score for the first PC is



The scores component of pca.cov contains all of the scores. The predict() function can be used to find these as well.

> score.cor2 <- pca.cov$scores

> head(score.cor2)

 Comp.1 Comp.2 Comp.3 Comp.4

1 -0.14765328 0.06106004 -0.07828190 -0.064361319

2 -0.26695271 -0.10141822 0.11386906 0.205631593

3 0.09490966 -0.06056653 0.15554367 -0.056555038

4 0.58278467 -0.21248552 -0.06625086 0.009916689

5 0.71019359 -0.09271586 -0.02809389 -0.029515624

6 -0.22386733 -0.03900914 -0.01442359 -0.030554186

 Comp.5

1 0.008322769

2 -0.036653466

3 -0.030894377

4 0.043937358

5 -0.030048015

6 -0.048100304

> # The predict function can be used as well

> head(predict(pca.cov))

 Comp.1 Comp.2 Comp.3 Comp.4

1 -0.14765328 0.06106004 -0.07828190 -0.064361319

2 -0.26695271 -0.10141822 0.11386906 0.205631593

3 0.09490966 -0.06056653 0.15554367 -0.056555038

4 0.58278467 -0.21248552 -0.06625086 0.009916689

5 0.71019359 -0.09271586 -0.02809389 -0.029515624

6 -0.22386733 -0.03900914 -0.01442359 -0.030554186

 Comp.5

1 0.008322769

2 -0.036653466

3 -0.030894377

4 0.043937358

5 -0.030048015

6 -0.048100304

> sum(pca.cov$loadings[,1]\*(goblet2[1,-1] - mu.hat))

[1] -0.1476533

Scree plot:

> plot(pca.cov, type = "lines", main = "Scree plot for cereal data")



The plot suggests one or two PCs may be enough. I would prefer to use two because 89.5% of the total variability is accounted for by them. Also, it is easy to display the scores of two PCs in a scatter plot (shown shortly).

Interpretation the PCs:

* PC #1 is a measure of the overall size of the goblet with little weight given to w6.
* PC #2 is measuring the difference between w1 and w2 (top) versus w4, w5, and w6 (base).
* PC #3 appears to be measuring the difference between w1, w4, and w6 versus w2 and w5. Interpretation???

An important part of making these interpretations is the original variables are measured in the same units.

Below are plots of the PCs:

> common.limits <- c(min(score.cor2[,1:2]), max(score.cor2[,1:2]))

> plot(x = score.cor2[,1], y = score.cor2[,2], xlab = "PC #1", ylab = "PC #2", main = "Principal components", xlim = common.limits, ylim = common.limits, panel.first = grid(col = "lightgray", lty = "dotted"))

> abline(h = 0)

> abline(v = 0)

> text(x = score.cor2[,1], y = score.cor2[,2]+0.03)



> #Bubble plot of first three PCs

> summary(score.cor2[,3])

 Min. 1st Qu. Median Mean 3rd Qu. Max.

-0.13839 -0.05606 -0.01783 0.00000 0.06390 0.15554

> #Bubble needs to contain all values > 0

> PC3.positive2 <- score.cor2[,3] - min(score.cor2[,3])

> symbols(x = score.cor2[,1], y = score.cor2[,2], circles = PC3.positive2, xlab = "PC #1", ylab = "PC #2", main = "Principal components", inches = 0.5, xlim = common.limits, ylim = common.limits, panel.first = grid(col = "lightgray", lty = "dotted"))

> text(x = score.cor2[,1], y = score.cor2[,2])

> abline(h = 0)

> abline(v = 0)



> #Different colors for positive and negative PC #3

> col.symbol <- ifelse(test = score.cor2[,3]>0, yes = "red", no = "blue")

> symbols(x = score.cor2[,1], y = score.cor2[,2], circles = PC3.positive2, xlab = "PC #1", ylab = "PC #2", main = "Principal components", inches = 0.5, xlim = common.limits, ylim = common.limits, panel.first = grid(col = "lightgray", lty = "dotted"), fg = col.symbol)

> text(x = score.cor2[,1], y = score.cor2[,2])

> abline(h = 0)

> abline(v = 0)



> #3D plot

> library(rgl)

> plot3d(x = score.cor2[,1], y = score.cor2[,2], z = score.cor2[,3], xlab = "PC #1", ylab = "PC #2", zlab = "PC #3", type = "h", xlim = common.limits, ylim = common.limits)

> plot3d(x = score.cor2[,1], y = score.cor2[,2], z = score.cor2[,3], add = TRUE, col = "red", size = 6)

> persp3d(x = common.limits, y = common.limits, z = matrix(data = c(0,0,0,0), nrow = 2, ncol = 2), add = TRUE, col = "green")

> grid3d(side = c("x", "y", "z"), col = "lightgray")

> text3d(x = score.cor2[,1], y = score.cor2[,2], z = ore.cor2[,3]+0.02, text = 1:nrow(goblet2))



Questions:

* Are there any outliers?
* Are there possible groupings?

After the analysis is complete, suppose new goblets are found. How can one obtain the PC scores for these new goblets? Use the predict() function. The new observations can be specified in the newdata argument of predict(). Below is a small example where a new observation happens to be the same as the first observation already in the data set.

> goblet2[1,] #To make this easy, let's use the first

 observation values

 ID w1 w2 w4 w5 w6

1 1 0.5652174 0.9130435 0.6086957 0.3043478 0.3478261

> pred.set <- data.frame(w1 = 0.5652174, w2 = 0.9130435, w4 = 0.6086957, w5 = 0.3043478, w6 = 0.3478261)

> predict(object = pca.cov, newdata = pred.set)

 Comp.1 Comp.2 Comp.3 Comp.4

[1,] -0.1476533 0.06106002 -0.07828186 -0.06436134

 Comp.5

[1,] 0.008322733

**Calculations by princomp()**

One could simply use the eigen() function to obtain the eigenvalues and corresponding vectors from the covariance matrix rather than using princomp(). If this was done, you will notice that there will be small differences! This is because princomp() uses the “biased” version of the estimated covariance matrix rather than the “unbiased” version.

This means that the estimated covariance matrix is calculated as

,

the usual “biased” estimate, rather than the unbiased estimate



that we have been using.

What happens to these values as N gets large?

In the end, there will not be meaningful differences between the scores, so we will use the scores as given by $scores or predict().

**Possible issues with PCA**

1. If the original set of variables are already uncorrelated, PCA will not help. Why?
2. PCA does not generally eliminate variables because the PCs are linear combinations of the original variables.
3. The original variables need to be measured in the same units and have similar variances.

Remember that PCA relies heavily on examining the variances of the original variables. Larger variance variables will dominate the other variables in the analysis.

For example, suppose there are three variables x1, x2, and x3 with variances of 98, 1.9, and 0.1, respectively. A way to maximize the variance of the first PC is to have it consist primarily of x1. For example, suppose we choose

y1 = a11(x1-μ1) + a21(x2-μ2) + a31(x3-μ3)

 = 1(x1-μ1) + 0(x2-μ2) + 0(x3-μ3)

 = x1-μ1

In this situation, the first principal component accounts for 98% of the variation in the data! It would very difficult to have any other linear combination that works significantly better. Also, in this situation (depending on the correlations between the variables), the second linear combination will likely be just for x2 and the third linear combination will likely be just for x3. Thus, we are left with what had originally with the data. We will look at an actual data example shortly where something similar happens.

A solution to this problem is to use standardized data. Equivalently, use **P** in place of **Σ** because the **P** is the covariance matrix of standardized random variables.

Note that there may be situations where one wants to weight one variable more than another. For example, suppose all of the variables are measured in the same numerical scale and you want the actual differences in these measurements to matter. This *may* be the case for the goblet data.

**PCA with the Correlation Matrix P**

PCA is most often performed using the correlation matrix **P** rather than the covariance matrix **Σ** to eliminate the problem with different numerical scales being used with variables.

Again, because **P** will not be known, we will use the estimated correlation matrix **R**. The corresponding eigenvalues and eigenectors for **R** are denoted by  and , respectively. On your own, make sure you can write out the PCs!





where  is the rth PC when using the standardized variables.

Determining the number of PCs

* Account for γ% of the variability
* Scree plot
* Find the number of eigenvalues greater than 1

What is the reason for the last bullet?

The analysis is being performed with “standardized data”. If a principal component does not account for more variability than an original variable (1), it may not be important.

PC scores

For each observation, we calculate the jth *principal component value* or *score* as:



for the jth PC and the rth observation.

There is a very small calculation issue with princomp() and predict(). While the correlation matrix is not affected by using 1/N vs. 1/(N – 1) in the covariance matrix (these values cancel out), the values of  are affected. The standardized values will use the biased estimate of the standard deviation in their calculation. The next example shows a small piece of code to fix the issue.

Example: Goblet data (GobletPCA.R, goblet.csv)

Below is the output for using the correlation matrix:

> pca.cor <- princomp(formula = ~ w1 + w2 + w4 + w5 + w6,

 data = goblet2, cor = TRUE, scores = FALSE)

> summary(pca.cor, loadings = TRUE, cutoff = 0.0)

 Comp.1 Comp.2 Comp.3

Standard deviation 1.7449238 1.1330854 0.63601891

Proportion of Variance 0.6089518 0.2567765 0.08090401

Cumulative Proportion 0.6089518 0.8657283 0.94663232

 Comp.4 Comp.5

Standard deviation 0.42528233 0.29321207

Proportion of Variance 0.03617301 0.01719466

Cumulative Proportion 0.98280534 1.00000000

Loadings:

 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5

w1 0.495 0.311 0.393 0.511 0.492

w2 0.490 0.330 0.393 -0.494 -0.502

w4 0.473 -0.412 -0.274 0.508 -0.523

w5 0.505 0.057 -0.666 -0.384 0.387

w6 0.190 -0.788 0.415 -0.299 0.286

Comments:

* PC #1 is a measure of the overall size of the goblet with not as much weight given to w6.
* PC #2 is measuring the difference between w1 and w2 (top) versus w4 and w6 (base). Note that w5 does not play a large role because its value is close to 0. Negative values of y2 indicate a large base (relative to height) and positive values indicate a large top (relative to height).
* PC #3 appears to be measuring the difference between w1, w2, and w6 versus w4 and w5. Interpretation???

Two PCs may be enough because 86.6% of the total variability is accounted for by them. Three PCs would definitely be enough (94.7% of the total variability). Notice that the third eigenvalue is less than 1, so the third PC accounts for less variability than an original variable.

Below are the plots for the PCs when using the correlation matrix.

> # Fix issue with 1/N vs. 1/(N-1)

> pca.cor$scale <- apply(X = goblet2[,2:6], MARGIN = 2, FUN = sd)

> score.cor <- predict(pca.cor, newdata = goblet2)

> head(score.cor)

 Comp.1 Comp.2 Comp.3 Comp.4

[1,] -0.8550658 0.5550454 -0.04593840 -0.553059990

[2,] -1.4929780 -0.9805278 -0.63447839 1.400864315

[3,] 0.6925848 -1.1866658 0.97492541 0.254414390

[4,] 3.9497633 -0.8219633 -0.60111958 -0.273882618

[5,] 4.2321986 0.1571450 -0.08857736 -0.015618439

[6,] -1.0207748 -0.2810525 -0.16578953 -0.002166875

 Comp.5

[1,] -0.07643603

[2,] 0.12848295

[3,] -0.28689963

[4,] 0.28578644

[5,] -0.24777873

[6,] -0.40365104

> par(pty = "s")

> common.limits <- c(min(score.cor[,1:2]),

 max(score.cor[,1:2]))

> plot(x = score.cor[,1], y = score.cor[,2], xlab = "PC

 #1", ylab = "PC #2", main = "Principal components",

 xlim = common.limits, ylim = common.limits, panel.first

 = grid(col = "lightgray", lty = "dotted"))

> abline(h = 0)

> abline(v = 0)

> text(x = score.cor[,1], y = score.cor[,2]+0.2)



> summary(score.cor[,3])

 Min. 1st Qu. Median Mean 3rd Qu. Max.

-1.25300 -0.46400 -0.08858 0.00000 0.44480 1.40400

> # Bubble needs to contain all values > 0

> PC3.positive <- score.cor[,3] - min(score.cor[,3])

> symbols(x = score.cor[,1], y = score.cor[,2], circles =

 PC3.positive, xlab = "PC #1", ylab = "PC #2", main =

 "Principal components", inches = 0.5, xlim =

 common.limits, ylim = common.limits, panel.first =

 grid(col = "lightgray", lty = "dotted"))

> text(x = score.cor[,1], y = score.cor[,2])

> abline(h = 0)

> abline(v = 0)

> # Different colors for positive and negative PC #3

> col.symbol <- ifelse(test = score.cor[,3]>0, yes = "red", no = "blue")

> symbols(x = score.cor[,1], y = score.cor[,2], circles = PC3.positive, xlab = "PC #1", ylab = "PC #2", main = "Principal components", inches = 0.5, xlim = common.limits, ylim = common.limits, panel.first = grid(col = "lightgray", lty = "dotted"), fg = col.symbol)

> text(x = score.cor[,1], y = score.cor[,2])

> abline(h = 0)

> abline(v = 0)

> pos.PC3 <- score.cor[,3]>0

> symbols(x = score.cor[pos.PC3,1], y = score.cor[pos.PC3,

 2], circles = PC3.positive[pos.PC3], xlab = "PC #1",

 ylab = "PC #2", main = "Principal components", inches =

 0.5, xlim = common.limits, ylim = common.limits,

 panel.first = grid(col = "lightgray", lty = "dotted"),

 fg = "red")

> symbols(x = score.cor[!pos.PC3,1], y =score.cor[!pos.PC3,

 2], circles = PC3.positive[!pos.PC3], inches =

 0.5, fg = "blue", add = TRUE)

> text(x = score.cor[,1], y = score.cor[,2])

> abline(h = 0)

> abline(v = 0)



> library(rgl)

> plot3d(x = score.cor[,1], y = score.cor[,2], z =

 score.cor[,3], xlab = "PC #1", ylab = "PC #2", zlab =

 "PC #3", type = "h", xlim = common.limits, ylim =

 common.limits)

> plot3d(x = score.cor[,1], y = score.cor[,2], z =

 score.cor[,3], add = TRUE, col = "red", size = 6)

> persp3d(x = common.limits, y = common.limits,

 z = matrix(data = c(0,0,0,0), nrow = 2, ncol = 2), add

 = TRUE, col = "green") #Plane

> grid3d(side = c("x", "y", "z"), col = "lightgray")

> text3d(x = score.cor[,1], y = score.cor[,2], z = score.cor[,3] + 0.2, text = 1:nrow(goblet2))



Need to rotate the plot to get a good view.

Comments:

* Notice that there appears to be more variability for PC #1 than PC #2 in the scatter plot. Again, this would be expected due to how PCs are found. In the 3D scatter plot, there is even less variability for PC #3. Make sure you take into account the z-axis scale when comparing PC #3 to PC #1 and PC #2 (I did not here so that the points were easier to see).
* The bubble plot can be used to help identify observations in the 3D scatter plot.
* Possible outliers: Goblets #10, #23, #24; examine where #4, #5, #17, and #18 are because we had identified these before with the stars plot
* “Possible” groupings – These are drawn on the bubble plot below with the help of the 3D scatter plot. Note that it can also be helpful to view the 3D scatter plot so that PC #1 or PC #2 are examined straight on. This allows you to essentially look at this one dimension with PC #3



Using PC #1 and PC #2, below are descriptions of the goblet groups:

* Red – Large in overall size
* Blue – Larger cups (tops) relative to the base
* Green – Smaller in overall size than many other goblets
* Yellow – Somewhat smaller cups relative to the base

Because the interpretation for PC #3 is difficult, I did not include it here in my goblet group descriptions. Hopefully, this is where a researcher could provide some insight to the meaning of PC #3 so that it could be included in the interpretation.

Example: Cereal data (CerealPCA.R, cereal.csv)

Because the cereal data has only 3 variables of interest (sugar, fat, and sodium), a PCA probably would not be of interest. However, because the number of variables is small, it is easier to use this data to demonstrate the methods.

How well do you think PCA will work for this data given the estimated correlation matrix?

> cor(cereal[,8:10])

 sugar fat sodium

sugar 1.0000000 0.2397225 -0.1635699

fat 0.2397225 1.0000000 -0.0661432

sodium -0.1635699 -0.0661432 1.0000000

Because sodium is measured on a different scale than fat and sugar content, using the correlation matrix in the PCA would be better than using the covariance matrix.

One could transform sodium from mg to grams and then take sodium/serving size. This would put sugar, fat, and sodium in the same numerical scale. However, given the drastic numerical differences between the sodium and the other variables, I think using the correlation matrix would still be a much better choice.

Below is my R code and output:

> pca.save <- princomp(formula = ~ sugar + fat + sodium, data = cereal, cor = TRUE, scores = TRUE)

> summary(pca.save, loadings = TRUE, cutoff = 0.0)

Importance of components:

 Comp.1 Comp.2 Comp.3

Standard deviation 1.1504216 0.9689554 0.8588687

Proportion of Variance 0.4411567 0.3129582 0.2458851

Cumulative Proportion 0.4411567 0.7541149 1.0000000

Loadings:

 Comp.1 Comp.2 Comp.3

sugar 0.667 0.091 0.739

fat 0.588 0.545 -0.597

sodium -0.458 0.833 0.310

> eigen(cor(cereal[,8:10]))

$values

[1] 1.3234700 0.9388747 0.7376554

$vectors

 [,1] [,2] [,3]

[1,] -0.6670789 0.09084126 0.7394279

[2,] -0.5879285 0.54538679 -0.5974055

[3,] 0.4575433 0.83324733 0.3104079

Comments:

* The “Standard deviation” row in the “Importance of components” table gives the standard deviation of each principal component  in the first row. Therefore, we have , , and .
* The “Proportion of Variance” row in the “Importance of components” table gives /tr(**R**) = /3. None of the eigenvalues have a large proportion. This indicates that the true dimension may be 3.
* The “Loadings” table gives the eigenvectors. The “loadings” name comes from the eigenvectors being the coefficients in the linear combination with the data.
* The eigenvectors are orthogonal to each other. For example,  = 0.667\*(-0.091) + 0.588\*0.545 - 0.458\*0.833 = 0.
* The eigenvectors are normalized to have a length of 1. For example,  = [0.6672 + 0.5882 + (-0.458)2]1/2 = 1.
* Notice the eigenvectors given by eigen() are multiplied by -1 in comparison to what princomp() gives us. Why is this not of concern?
* Scree plot

> plot(pca.save, type = "lines", main = "Scree plot for

 cereal data")

> #screeplot(x = pca.save, type = "lines", main = "Scree

 plot for cereal data") #Same



Be careful with the y-axis scale here (does not start at 0). How many PCs should be used?

* Because the true dimension of the data set does not appear to be less than the original, interpretation of the PCs probably should not be done. If it was done, the first principal component is measuring the sugar and fat content vs. the sodium content. Provided it makes sense to a researcher, this may offer a useful interpretation of the data set.

For illustration purposes, below are the variances of the PCs. Notice that they are the same as the eigenvalues.

> save.score <- predict(pca.save2, newdata = cereal)

> apply(X = save.score, MARGIN = 2, FUN = var)

 Comp.1 Comp.2 Comp.3

1.3234700 0.9388747 0.7376554

> pca.save$sdev^2

 Comp.1 Comp.2 Comp.3

1.3234700 0.9388747 0.7376554

Using the covariance matrix instead

Again, this is done for illustrative purposes:

> cov(cereal[,8:10])

 sugar fat sodium

sugar 0.0223681533 0.0009926903 -0.060242015

fat 0.0009926903 0.0007666194 -0.004509788

sodium -0.0602420149 -0.0045097883 6.064039752

> N <- nrow(cereal)

> cov(cereal[,8:10])\*(N-1)/N

 sugar fat sodium

sugar 0.021808949 0.0009678730 -0.058735965

fat 0.000967873 0.0007474539 -0.004397044

sodium -0.058735965 -0.0043970436 5.912438758

> sum(diag(cov(cereal[,8:10])\*(N-1)/N)) #tr(sigma^\_bias)

[1] 5.934995

> pca.cov <- princomp(formula = ~ sugar + fat + sodium, data = cereal, cor = FALSE)

> summary(pca.cov, loadings = TRUE, cutoff = 0.0)

Importance of components:

 Comp.1 Comp.2 Comp.3

Standard deviation 2.4316718 0.145825069 0.0265059505

Proportion of Variance 0.9962986 0.003582977 0.0001183767

Cumulative Proportion 0.9962986 0.999881623 1.0000000000

Loadings:

 Comp.1 Comp.2 Comp.3

sugar 0.010 0.999 0.045

fat 0.001 0.045 -0.999

sodium -1.000 0.010 0.000

Comments:

* Note that princomp() here uses the biased estimated covariance matrix.
* 
* There is “more” variability for sodium than for sugar and fat. This is because of the units that sodium is measured in.
* The trace of the biased estimate of the covariance matrix is 5.93. This is the total variance.
*  = 2.432 = 5.91,  = 0.0213,  = 0.0007
* The proportion of variability accounted for by the first eigenvalue is 5.91/5.93 = 0.9963. This suggests that the data can be represented in d = 1 dimensions instead of p = 3.
* The first principal component is primarily based on sodium because sugar and fat have values close to 0.

Additional questions:

* Suppose more than 3 PCs are needed for a PCA. How could you represent these graphically?
* Suppose 2D scatter plots and/or 3D scatter plots are used to examine the PCs, but less than 50% of the total variation is accounted for by 3 PCs. Is this o.k. to do?

This could be o.k., but you need to understand that important information in the data could be missed. For example, it is like looking at a scatter plot between x1 and x2 without accounting for x3. This may or may not be o.k. depending on the relationship between all three variables.

* Notice there are no formal inference methods in this subsection. For example, we never calculate a confidence interval for λj. Interestingly, this is rarely done in practice. This is likely due to PCA being an exploratory technique to be followed up with other analyses.