Principal Components Analysis

We will not go into a lot of details for Principal Components Analysis (PCA), but we can at least get a glimpse of what the technique is supposed to do and how we can use R to do a simple analysis.

First, let's explain what PCA does. Simply put, it is used for *dimension reduction*. Okay, that's a mouthful, so let's give an example. Suppose you take 120 turtles and take 35 measurements of each turtle (for example: length, width, height, length of arm, length of leg, length of head, width of head, etc.). You now have 35 measurements. It might make sense to you that you need to simplify this. 35 variables are way too many to do anything reasonable with.

Suppose you could somehow *reduce* the number of variables and still keep most of the information you collected. Maybe 5 variables will do almost as well as 35. This is what PCA lets you do. Or to put it graphically:

$$\text{35 old variables} \xrightarrow{\quad PCA \quad} \text{Fewer new variables}$$

We can then use these *new* variables in our subsequent analysis and we can be reasonably sure we haven't lost any important information.

Since we have to avoid the mathematical details (we need a lot of matrix algebra), probably the best way illustrate PCA is just to use R to walk through an example. Let's use the iris data set that's built into R, but to keep things simple, we'll concentrate on only *Iris virginica*. To do this, the first thing we need to do is to create a data set with only *I. virginica*:

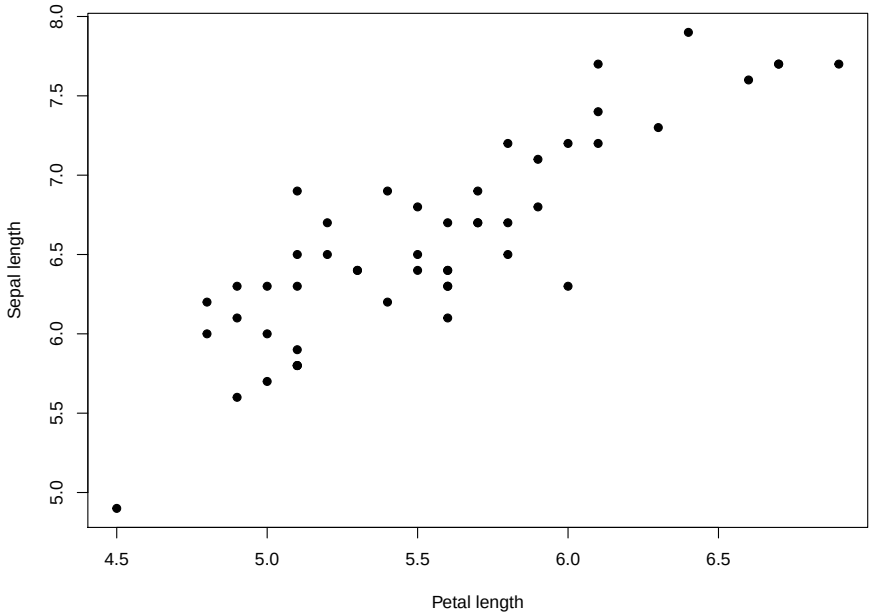
```
# First, we'll subset the iris data set:
virginica <- subset(iris, Species == "virginica")


# If you want to make sure it worked correctly you can just type:
virginica
```

Note the way we subset the data using the `subset` function together with the double `==` to say "equal to" (one equal sign will not work).

Once we've done that, we can plot our data to see what it looks like:

```
# Now plot Petal length vs.  Sepal length:
plot(virginica$Petal.Length, virginica$Sepal.Length)
```

Which gives us the following plot:



If we look at our graph, we notice that the points are scattered from the bottom left to the top right of the graph. One way to think about PCA is that it will take this "direction" (from bottom left to top right) and rotate it so that the variation going from the bottom left to the top right is now parallel to the x-axis. In other words, PCA *rotates* our data so that the maximum variotion is along the x-axis, the second highest variation is on the y-axis (perpendicular to the x-axis), and so on.

But let's proceed with our PCA. In R the command is pretty simple:

```
# This does most of the calculations:
# scale = TRUE standardizes the variations - usually a good idea.
flower <- prcomp(cbind(virginica$Petal.Length, virginica$Sepal.Length),
                 scale = TRUE)
```

And you're probably used to asking R to summarize our analysis by doing:

```
# The basic result shows how much variance is now accounted for in each
of the new "components":
summary(flower)
```

Okay, so we've done our PCA. Now what? What does it all mean? Well, let's look at the summary:

```
Importance of components:
                          PC1     PC2
Standard deviation     1.3654 0.36848
Proportion of Variance 0.9321 0.06789
Cumulative Proportion  0.9321 1.00000
```

Our PCA calculated two new variables which are *linear combinations* of the old variables. Without going into details, that means each of the new variables is a mathematical combination of the two old variables. But notice the second line of our output, the `Proportion of Variance`. Our first new variable now explains 93% of the variation in our original data! If we wanted to do our analysis with just one variable, we could decide to just use PC1 (the name of our new first variable) and we would still have almost all of the variation that was present in *both* of our original variables. This is the idea behind PCA. Of course this idea also works with more than two variables (it's not really needed if all you have is two variables).

But let's show a few other results that might be interesting. We might want to know how these new variables are related to the old variables:

```
# This shows the relationships between the old and new variables:
flower$rotation
```

Which gives us the following result (these are often called *loadings*):

```
            PC1        PC2
[1,] -0.7071068 -0.7071068
[2,] -0.7071068  0.7071068
```

This is basically the correlation between the original variables (labeled `[1,]` and `[2,]`. In this case it's not very interesting (and not too informative), but it shows that PC1 has a strong negative correlation with both of the original variables, while PC2 has a negative correlation with Petal Length and a positive correlation with Sepal Length. We'll look at this a bit more when we do a more complicated example below.

Suppose you want to actually *see* these new variables. We can do that too:

```
# If you want to see the actual principal components do the following:
head(flower$x)
```

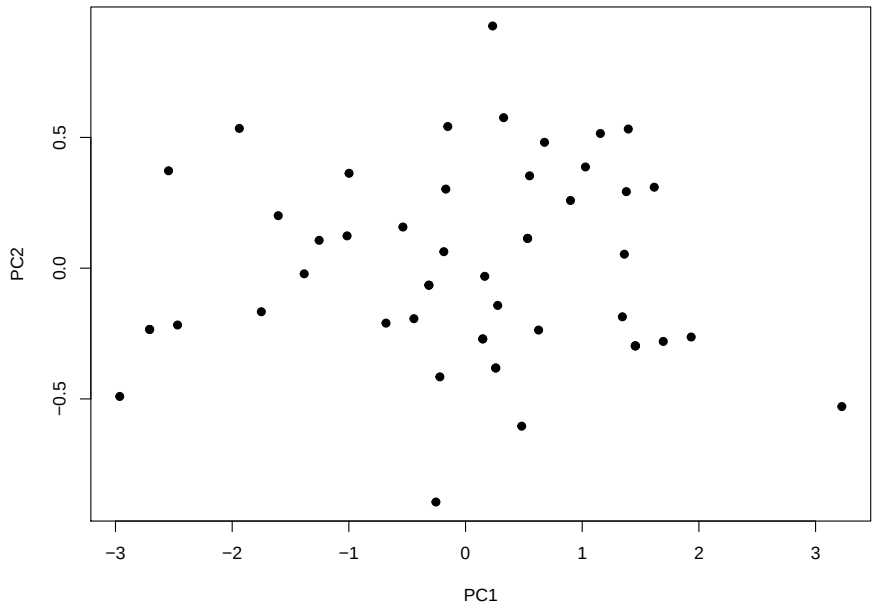
The `head` command tells R to just print the first few lines (we really don't want to see all 50 lines of data):

```
             PC1        PC2
[1,] -0.2537333 -0.8942532
[2,]  1.4553850 -0.2971485
[3,] -1.0152208  0.1234812
[4,]  0.2587607 -0.3817592
[5,] -0.2198891 -0.4156035
[6,] -2.4680920 -0.2173764
```

If we want to see a graph of the new variable we can just just plot the principal components (the `pch` command tells R to use solid circles instead of open circles):

```
# We can also plot these components:
# Notice that most of the variation is now in the x direction
plot(flower$x[,1],flower$x[,2], ylab = "PC2", xlab = "PC1", pch = 19)
```

And here is the graph:



Notice that this plot is actually a bit similar to the first plot we did, but now the highest variation is in the x-direction, and the second highest variation is in the y-direction. Or to put it another way, PC1 is aligned with the x-axis, and PC2 with the y-axis. As mentioned above, PCA basically takes the original data and rotates it so that the maximum variation is along the direction of PC1, and the second, and third, and so on, highest variations are aligned with PC2, PC3, and so one, *subject to the constraint that all the axes for the principal components are at right angles.*

Next we want to talk for a minute about *eigenvalues*. We will not officially define eigenvalues here - if you're interested, look up almost any text on matrix algebra or just look it up on Wikipedia. A PCA uses eigenvalues in several ways, but one way of using eigenvalues is to figure out how much each of the *new* variables is worth in terms of the old variables. Before we get confused, let's just use the following bit of code to calculate them (without explanation):

```
# Now we get the "eigenvalues":
ev <- (flower$sdev)^2
ev

# and the relative variance explained:
varnce <- ev*100/sum(ev)

# and the cummulative variance explained:
cumvar <- cumsum(varnce)

# We now make a "data.frame" and give everything nice names:
nicesummary <- data.frame(eig = ev, variance = varnce,
                cvariance = cumvar)

nicesummary
```

And we can get our results in a nice, neat table:

```
        eig  variance cvariance
1 1.8642247 93.211237  93.21124
2 0.1357753  6.788763 100.00000
```

Basically what an eigenvalue does in this example is that it tells us how much of the original variation is explained by each PC (Principal Component). It does this by comparing it to the "average" variation in each of the original variables. So for our example, our first new variable (PC1) explains as much variation as 1.86 of the original variables. In other words, it's almost as good as the two original variables. Not bad! The second and third columns are simply the percent variance explained by each new variable (we already saw this in our earlier output) and the cumulative variance explained. Why do we want to look at this? Well, we need some reasonably objective way of deciding how many of the new variables we want to keep for any subsequent analysis.

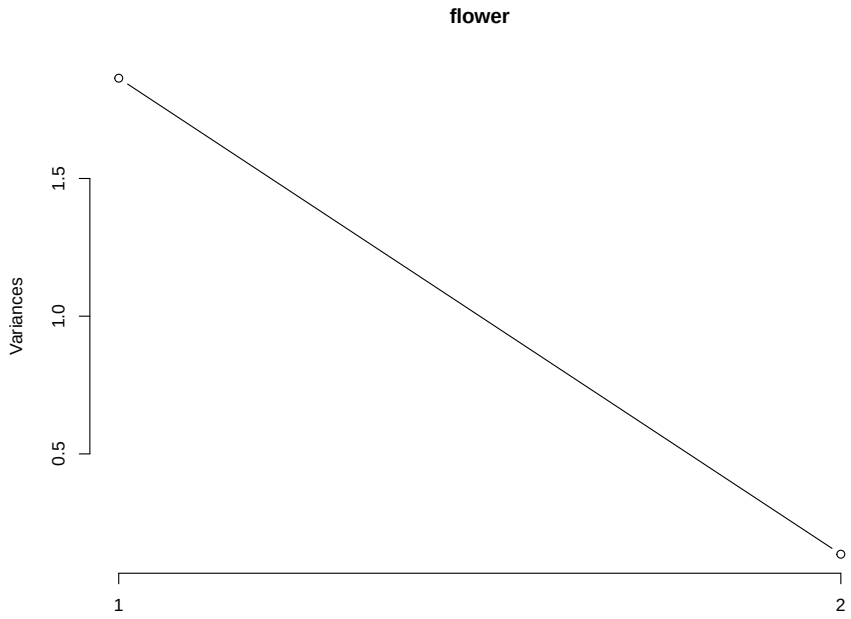
As explained above, PCA is a data reduction technique. It takes many variables and reduces most of the variation to fewer variables which we can then use in our analysis. But, how many of the new variables should we use in our analysis? There are two suggestions. The first one is based on eigenvalues and says to use any PC that has an eigenvalue of

more than 1. Any variable that has an eigenvalue of less than one actually explains *less* variation than one of one of the original variables. The second suggestion is to look at a scree plot. A scree plot plots the variation of the PC's on the y-axis and the number of PC's on the x-axis. The basic idea is to look for a bend in the plot, at which point the subsequent PC's don't give a large reduction in variance.

Once again, before we get confused, let's just take a look (although with two original variables, this isn't very useful):

```
# This is a scree plot - not very useful with only two variables.
# It shows how much variation each component accounts for.
# This is much more useful with more variables (see 2nd example below).
plot(flower,type="lines")
```

And here's our plot (this will work much better in our next example):



We'll go through this plot a bit more in class.

So that's the basic idea behind PCA. We forgot one thing. If we want to use PC's for our analysis instead of our original variables, we need to do know how to interpret our PC's. That's not easy, since they're a combination of the original variables. The best we can usually do is say something like "PC1 seems to be related to a, b, and c", while "PC2 seems to be related to d and e" (assuming we had at least five original variables). We can use the results of the `flower$rotation` command to do this since it shows the relationship between our new variables (PC's) and our old or original variables (the *loadings* from above).

Here's essentially the same code as above, but now summarized in one piece, although this time we're using all four measurements of our flower (we explained the following in class, but the code below has some added annotations):

```
# **********************
# Now let's repeat this, but use all four measurements of our virginica
# flowers:

flower2 <- prcomp(cbind(virginica$Petal.Length, virginica$Sepal.Length,
            virginica$Petal.Width, virginica$Sepal.Width),scale = TRUE)

# The basic result shows how much variance is now accounted for in each
# of the new "components":
summary(flower2)

# Notice that the first two variables explain 85% of the variation.

# This shows the relationships between the new and old variables:
flower2$rotation

# If you want to see the actual principal components:
head(flower2$x)

# Plotting the PC's (note the direction of maximum variability
plot(flower2$x[,1],flower2$x[,2], ylab = "PC2", xlab = "PC1", pch = 19)

# Now we get the "eigenvalues":
ev2 <- (flower2$sdev)^2
ev2

# and the relative variance explained:
varnce2 <- ev2*100/sum(ev2)

# and the cummulative variance explained:
cumvar2 <- cumsum(varnce2)

# We now make a "data.frame" and give everything nice names:
nicesummary2 <- data.frame(eig = ev2, variance = varnce2,
cvariance = cumvar2)

nicesummary2
```

```
# Notice that the first the first PC is the only one that has an
# eigenvalue greater than 1 (the second comes really close).


# Finally, our scree plot. This shows how much variation each PC
# accounts for.  This time it is a bit more informative since
# we're using 4 variables (not two)

plot(flower2,type="lines")

# Based on both the eigenvalues and the scree plot we should
# probably keep one or two PC's for any subsequent analysis.
# (PC's 3 & 4 are useless).
```