Quantitative Understanding in Biology
Principal Component Analysis

Jinhyun Ju
Jason Banfelder
Luce Skrabanek

December 10, 2019

1 Preface

For the last session in this course, we'll be looking at a common data reduction and analysis technique called principal components analysis, or PCA. This technique is often used on large, multi-dimensional datasets, such as those resulting from high-throughput sequencing experiments. To fully appreciate the mathematics behind PCA, one needs a firmer background in linear algebra than we have developed in the this course (in particular, in eigensystems). Additionally, traditional uses of PCA don't rely on the statistical methods and reasoning we've developed here (although variance is at its core). We include an overview of PCA here because it is a very common data analysis technique that is used on the same kinds of data that you'll be analyzing with many of the methods introduced in this course, and to ensure that you have a good basis for critically evaluating the methods and results reported by others.

2 Introduction

Principal component analysis (PCA) is a simple yet powerful method widely used for analyzing high dimensional datasets. When dealing with datasets such as gene expression measurements, some of the biggest challenges stem from the size of the data itself. Transcriptome wide gene expression data usally have 10,000+ measurements per sample, and commonly used sequence variation datasets have around 600,000 $\sim$ 900,000 measurements per sample. The high dimensionality not only makes it difficult to perform statistical analyses on the data, but also makes the visualization and exploration of the data challenging.

These datasets are typically never fully visualized because they contain many more datapoints than you have pixels on your monitor. PCA can be used as a data reduction tool, enabling you to represent your data with fewer dimensions, and to visualize it in 2-D or 3-D where some patterns that might be hidden in the high dimensions may become apparent.

3 The core concept: Change of basis

Let us think about the problem that we are facing with a simple example that involves the measurement of 5 genes $g_1, \ldots, g_5$. In this case, we would have 5 measurements for each sample, so each sample can be represented as a point in a 5-dimensional space. For example, if the measurements for sample 1 (x_1) were $g_1 = 1.0, g_2 = 2.3, g_3 = 3.7, g_4 = 0.2, g_5 = 0.3$, we can represent x_1 as a 5-dimensional point where each axis corresponds to a gene. (Just like we would define a point on a two-dimensional space with x- and y-axes as (x, y).) So if we had two samples they would look something like this in vector form:

$$x_1 = \begin{pmatrix} 1.0 \\ 2.3 \\ 3.7 \\ 0.2 \\ 0.3 \end{pmatrix} \; x_2 = \begin{pmatrix} 0.8 \\ 1.84 \\ 2.22 \\ 0.12 \\ 0.18 \end{pmatrix} \tag{1}$$

It might be quite obvious, but this means that the point x_1 is defined as 1 unit in the g_1 direction, 2.3 units in the g_2 direction, 3.7 in the g_3 direction and so on. This can be represented as

$$x_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot 1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot 2.3 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \cdot 3.7 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \cdot 0.2 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \cdot 0.3 \tag{2}$$

where the set of 5 unit vectors are the "basis set" for the given data. If the expression levels of all of the 5 genes were independent of each other (or orthogonal, if we talk in vector terms), we would have to look at the data as it is, since knowing the expression level of g_1 will not give us any information about other genes. In such a case, the only way we could reduce the amount of data in our dataset would be to drop a measurement of a gene, and lose the information about that gene's expression level. However, in reality the expression levels of multiple genes tend to be correlated to each other (for example,

pathway activation that bumps up the expression levels of g_1 and g_2 together, or a feedback interaction where a high level of g_3 suppresses the expression level of g_4 and g_5), and we don't have to focus on the expression levels individually. This would mean that by knowing the expression level of g_1 we can get some sense of the expression level of g_2, and from the level of g_3 we can guess the levels of g_4 and g_5. So let's say, for the sake of simplicity, that our two samples obey this relationship perfectly (which will probably never happen in real life) and represent our samples in a more compact way. Something like this might do the job.

$$x_1 = \begin{pmatrix} 1.0 \\ 2.3 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot 1 + \begin{pmatrix} 0 \\ 0 \\ 3.7 \\ 0.2 \\ 0.3 \end{pmatrix} \cdot 1 \qquad x_2 = \begin{pmatrix} 1.0 \\ 2.3 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot 0.8 + \begin{pmatrix} 0 \\ 0 \\ 3.7 \\ 0.2 \\ 0.3 \end{pmatrix} \cdot 0.6 \tag{3}$$

What this means is that we are representing our sample x_1 as a two dimensional point (1,1) in the new space defined by the relationships between genes. This is called a "change of basis" since we are changing the set of basis vectors used to represent our samples. A better representation would be to normalize the basis vectors to have a unit norm, like this where the normalization factor just gets multiplied into the coordinates in the space:

$$x_1 = \begin{pmatrix} 0.398 \\ 0.917 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot 2.51 + \begin{pmatrix} 0 \\ 0 \\ 0.995 \\ 0.054 \\ 0.081 \end{pmatrix} \cdot 3.72 \qquad x_2 = \begin{pmatrix} 0.398 \\ 0.917 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot 2.01 + \begin{pmatrix} 0 \\ 0 \\ 0.995 \\ 0.054 \\ 0.081 \end{pmatrix} \cdot 2.23 \tag{4}$$

In this example, the new basis vectors were chosen based on an assumed model of gene interactions. Additionally, since the expression values followed the model exactly, we were able to perfectly represent the five dimensional data in only two dimensions. PCA follows similar principles, except that the model is not assumed, but based on the variances and correlations in the data itself. PCA attempts to minimize the information lost as you drop higher dimensions, but, unlike the contrived example above, there will probably be some information content in every dimension.

4 Correlation and Covariance

Another way to think about PCA is in terms of removing redundancy between measurements in a given dataset. In the previous example, the information provided by the mea-

surement of g_1 and g_2 were redundant and the same was true for g_3, g_4 and g_5. What we essentially did was get rid of the redundancy in the data by grouping the related measurements together into a single dimension. In PCA, these dependencies, or relationships between measurements, are assessed by calculating the covariance between all the measurements. So how do we calculate covariance? First, we recall that the variance of a single variable is given by

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

Covariance can simply be thought of as variance with two variables, which takes this form

$$cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

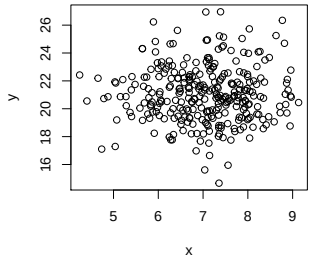
If this looks familiar, you are not mistaken (take a look at the equation for the Pearson correlation coefficient). Correlation between two variables is just a scaled form of covariance, which takes values between -1 and 1.

$$cor(x,y) = \frac{cov(x,y)}{\sigma_x \cdot \sigma_y}$$

So just like correlation, covariance will be close to 0 if the two variables are independent, or will take a positive value if they tend to move in the same direction, or a negative value if the opposite is true.

Let's see what variance and covariance look like for a couple of two-dimensional datasets. We'll begin by creating a dataset with 300 measurements for x and y, where there is no relationship at all between x and y.

```
x <- rnorm(n = 300, mean = 7,  sd = 1)
y <- rnorm(n = 300, mean = 21, sd = 2)
plot(y ~ x)
```

As you can see from the plot above, there not much of a relationship between the xs and the ys. You can quantify this mathematically by computing the covariance matrix for this data.

```
samples <- cbind(x, y)
head(samples)
```

```
##              x        y
## [1,] 7.261869 21.34318
## [2,] 6.756167 20.46687
## [3,] 8.836920 21.03928
## [4,] 7.669822 21.73555
## [5,] 7.366226 25.22898
## [6,] 8.383130 23.49591
```

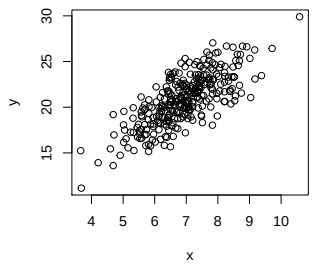
```
cm <- cov(samples)     # compute the covariance matrix
round(cm, digits = 3)  # print it so it is readable
```

```
##       x     y
## x 0.922 0.022
## y 0.022 3.963
```

The diagonals of the covariance matrix hold the variance of each column (recall that variance is just SD^2), and the off-diagonal terms hold the covariances. From the matrix, you can see that the covariances are near zero.

Now let's consider a case where there is a relationship between the xs and ys.

```
x <- rnorm(n = 300, mean = 7, sd = 1)
y <- rnorm(n = 300, mean = 7, sd = 2) + 2 * x
plot(y ~ x)
```

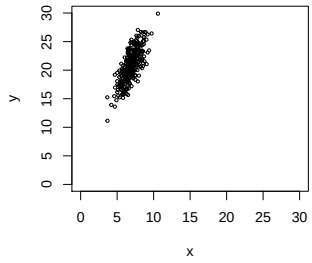
In this case, the plot shows that there is a significant relationship between x and y; this is confirmed by the non-zero off-diagonal elements of the covariance matrix.

```
##       x     y
## x 1.069 2.179
## y 2.179 7.808
```

If you look at the code for this example, you can see that y is a mix of a random component and a weighted portion of x, so these results shouldn't be all that surprising.

If you were given the data in the figure above and asked to best represent each point using only one number instead of two, a few options might come to mind. You could just forget about all of the x values and report the ys. Or you could just forget about the y values and report the xs. While either of these options might seem equally good, replotting the data with equal scales might cause you to change your mind:

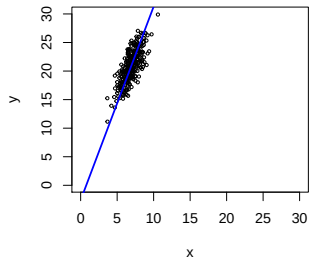
```
plot(y ~ x, xlim = c(0,30), ylim=c(0,30), cex = 0.5)
```



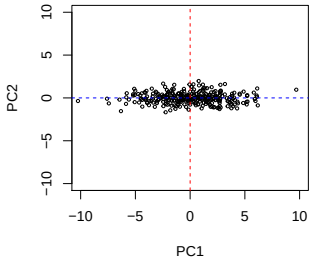
Here we can see that there is more variance in y, so the best that we might do given one value per point is to report the ys, and tell our audience to assume that all the xs were 7. In this case, the estimated location of each point wouldn't be too far from its actual location.

However, since there is a clear correlation between x and y, we can do better by reporting the position along the blue line shown in the plot below.

```
results <- prcomp(samples)
pc1 <- results$rotation[,"PC1"]
samples.xfromed <- results$x
```



Another way of thinking about this line of reasoning is saying that we'll transform the data so that the blue line is our new x-axis. While we're at it, we'll choose the origin of our x-axis to be the center of our data. A plot of the transformed data looks like this (notice that here we've labeled the axes 'PC1' and 'PC2'):



The blue line in the prior plot is the first Principal Component of the data, and the direction orthogonal to it is the second Principal Component. From this demonstration, you can see that PCA is nothing more than a rotation of the data; no information is lost (yet).

So far, we skipped over the details of just how the blue line was determined. In PCA, we choose the first principal component as the direction in the original data that contains the most variance. The second principal component is the direction that is orthogonal (i.e., perpendicular) to PC1 that contains the most remaining variance. In a two dimensional case the second PC is trivially determined since there is only one direction left that is orthogonal to PC1, but in higher dimensions, the remaining variance comes into play. PC3

is chosen as the direction that is orthogonal to both PC1 and PC2, and contains the most remaining variance, and so on... Regardless of the number of dimensions, this all just boils down to a rotation of the original data to align to newly chosen axes. One of the properties of the transformed data is that all of the covariances are zero. You can see this from the covariance matrix of the transformed data for our example:

```
round(cov(samples.xfromed), digits = 3)
```

```
##       PC1   PC2
## PC1 8.451 0.000
## PC2 0.000 0.426
```

5 Interpreting the PCA Results

Once the PCA analysis is performed, one often follows up with a few useful techniques. As our motivation for considering PCA was as a data reduction method, one natural follow-up is to just drop higher PCs so that you're dealing with less data. One question that should naturally occur to you then is how to decide how many principal components to keep. In a study that produces, say, a 15,000 dimensional gene expression dataset, is keeping the first 50 PCs enough? Or maybe one needs to keep 200 components?

It turns out that the answer to this question is buried in the covariance matrix (you need to look at the matrix's eigenvalues, but we won't go into the linear algebra of eigensystems). In particular, the fraction of variance in a given PC relative to the total variance is equal to the eigenvalue corresponding to that PC relative to the sum of all the eigenvalues.

While we'll show the computation in `R` here, don't worry about the details; for now just appreciate that is is pretty easy to compute the fraction of information (i.e., variance) captured in each PC.

```
results$sdev
```

```
## [1] 2.9071091 0.6524789
```

```
explained_variance <-
  round((results$sdev^2) / sum(results$sdev^2), digits = 2)
explained_variance
```

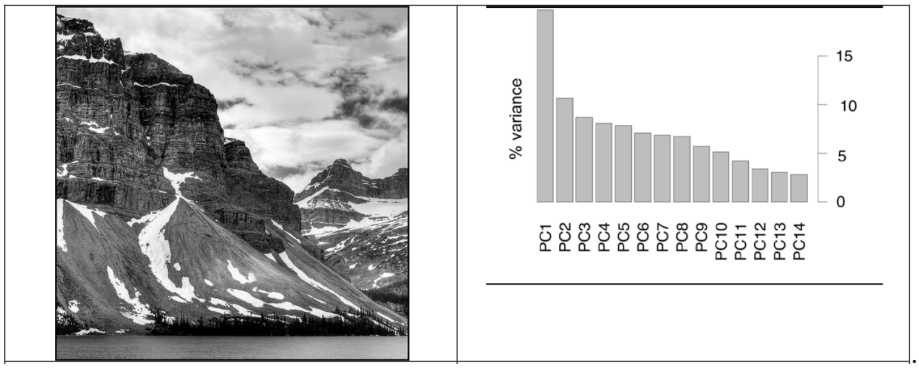
```
## [1] 0.95 0.05
```

Here we see that, for our toy example, 95% of the variation in the data is captured by the first PC.

When dealing with real data sets that have more than two dimensions, there are often

two practices that are carried out. For data sets with a handful of dimensions (say eight or twenty), one typically retains the first two or three PCs. This is usually motivated by simple expediency with regard to plotting, which is a second motivation for PCA. Since you can make plots in two or three dimensions, you can visually explore your high-dimensional data using common tools. The evaluation of how much variance is retained in the original dataset informs how well you can expect your downstream analysis will perform. Typically, you hope to capture around 90% of the variance in these first two or three components.

In very high dimensional datasets, you typically will need much more than two or three PCs to get close to capturing 90% of the variation. Often, your approach will be to retain as many PCs as needed to get the sum of the eigenvalues of the PCs you are keeping to be 90% of the sum of all of them. An alternative is to make a so-called scree plot. This is simply a bar blot of each eigenvalue in descending order, and (hopefully) will be reminiscent of the geological feature after which it is named. The figure below shows both a geological scree, and a typical scree plot from a gene expression study.



Such a plot may help you evaluate where the point of diminishing returns lies. The hope here is that the first few PCs represent real (reduced dimension) structure in the data, and the others are just noise.

6 Further interpretation of PCA results

After performing PCA, it is very tempting to try to interpret the weights (or loadings) of the principal components. Since each principal component is a linear combination of the original variables, it can be of interest to see which variables contribute most to the most important principal components. For example, when looking at gene signatures, those genes that contribute most to the first few PCs may be considered 'more important' genes.

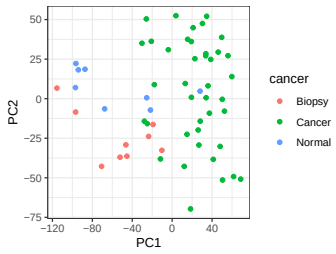
While the PC loadings can sometimes be the source of useful hints about the underlying natural variables of a biological process, one "needs to be more than usually circumspect when interpreting" the loadings (Crawley, The R Book, 2007). Part of this derives from the fact that PCA results are sensitive to scaling, and part of this may be that individual PCs may be very sensitive to noise in the data. When performing PCA on a computer, make sure you know what your program does in terms of scaling (as well as with the underlying numerics).

As scientists, we need as much help as we can to interpret our data, so to say that one should never look at or think about loadings would be impractical. A good rule of thumb may be to treat any interpretations about loading as a hypothesis that needs to be validated by a completely independent means. This is definitely one of those areas where you want to be conservative.

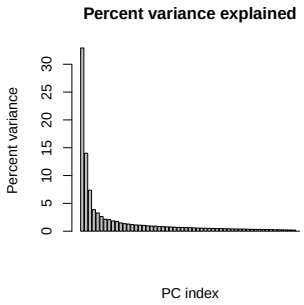
7 Example with real data

Let's look at the results of applying PCA to a real gene expression dataset to get a sense of what we would be looking at when interpreting high-dimensional data. The dataset that we are going to use is a publicly available gene expression dataset, which was generated from bladder cancer and normal cells. Further details can be found here: `http://www.bioconductor.org/packages/release/data/experiment/html/bladderbatch.html`

Generally, after performing PCA on a high dimensional dataset, the first thing to do is visualize the data in 2-D using the first two PCs. Since we have sample information as well, we can use this to color label our data points.



We can see that the samples mostly cluster together according to their "cancer" label on PC1 and PC2. This could mean that the biggest variation in the dataset is due to the status of the tissue. However, we have many more PCs in this case than in our toy dataset so it is important to check how much variance each PC actually explains, and then evaluate the importance of the first two PCs.



We can see that the plot generated from this dataset holds up to its name and shows us that most of the variance is explained by the first 2 or 3 PCs, while the rest have only minor contributions. In such a case, it would be reasonable to reduce the dimensionality using the first few PCs.

8 PCA and scaling

We mentioned briefly above that PCA is sensitive to scaling, and we'll take a moment here to dig a bit deeper into this. We understand that the first PC is chosen to capture the most variance in the data, but it is often forgotten that variance (like SD) has units (recall that if you measure heights of people in feet, the units of SD is feet). This implies that if we collect and record the same data in different units, the variance will be different; for example, if we chose to record the heights of the people enrolled in our study in inches instead of feet, the numerical value of the variance would be 144 times larger in the direction corresponding to our height measurement.

Now consider the direction of PC (e.g., the solid blue line in our prior plot), which is a mixture of x and y. If x and y are of the same units, interpretation of this direction is natural, but if they are not, the units of measure along this dimension are odd at best. We can still plow ahead and perform PCA with the numerical values, but we need to acknowledge that there is an implicit scaling. If x is weight in kg and y in height in m, then we're saying that one meter of variance is worth just as much as one kilo of variance. If we choose to work with heights in cm, we'd get a different PC because height variance would be weighted 1,000 times more.

This phenomenon is generally not a issue when all of our measurements have the same units, since the scaling factor is a constant. So if all dimensions in the original data are gene expression values, there is not much to worry about.

However, in many studies, researchers often mix in other measurements. For example, in trying to determine what combinations of factors might be determinants of bladder cancer,

one might perform a PCA including gene expression values for most of the columns, but also the subject's age, weight, income, education level, etc. Once units are mixed, something needs to be done to rationalize the mismatch. One common approach is to normalize each measurement based on its SD, but this of course changes the original variances. The waters can get murky; and this is often compounded by the fact that some PCA implementations will, by default, normalize all data, while others don't. When critically evaluating others' work, if there isn't an explicit statement about whether data was normalized or not, you might be extra cautious.

9 Further Reading

For a more detailed and intuitive explanation on PCA, we recommend the tutorial written by Jonathon Shlens: `http://arxiv.org/pdf/1404.1100.pdf`

