Univariate, Bivariate, and Multivariate Statistics Using R

Quantitative Tools for Data Analysis and Data Science

Daniel J. Denis

WILEY

10

Principal Component Analysis

LEARNING OBJECTIVES

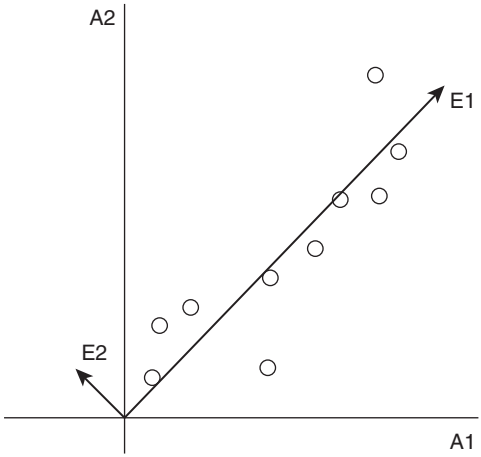
- Understand the nature of principal component analysis, and perspectives on how it may be useful.
- Understand the differences between principal component analysis and exploratory factor analysis.
- How to run and interpret a component analysis in R using `prcomp()` and `princomp()` functions.
- How to plot correlation matrices among variables to assess magnitudes of correlation.
- How to make potential substantive sense of components derived from PCA.

Principal component analysis, or "PCA" for short, is a **data reduction** technique used to account for variance in a set of *p* variables into fewer than *p* dimensions. Traditional PCA assumes your variables are **continuous** in nature. As an example, suppose a researcher has 100 variables in his or her data set. In these variables is a certain amount of variance. That is, the entire set of variables contains a particular amount of variability. If we wanted to know the **total variance** of the variables, we could sum them up and obtain a measure of the total variance. What PCA seeks to do is take this original set of variables in *p* dimensions, and performs a transformation on these variables such that the total amount of original variance is preserved. For example, if the 100 variables accounted for a total variance of 50, the PCA transformation will not change this total amount of variance. What it will do is attempt to explain this variance in as few new "components" as possible, where the first few components extracted will account for most of the variance in the original set of variables. At least that is the hope, if the technique is successful on a substantive level. The goal is to have most of the variance load on the first few components and hence "reduce the data" down to these few

Univariate, Bivariate, and Multivariate Statistics Using R: Quantitative Tools for Data Analysis and Data Science, First Edition. Daniel J. Denis.

components. Instead of needing all of the variables, the researcher can instead focus on the first few components that account for most of the variance in the variables.

As an example of what PCA accomplishes, consider the following plot adapted from Denis (2016):



In the plot, the original axes are A1 and A2. The "new" axes after the transformation are E1 and E2. Though there are several ways of interpreting how PCA works, one common way is viewing the sample components as generated by rotating the coordinate axes so that they pass through the scatter in the plot of maximum variance (Johnson and Wichern, 2007). Hence, we can see that what PCA has done is performed a **rotation** such that the new axes optimize some function of the data. What function is maximized? As we will see, the rotation occurs such that the **first component accounts for as much variance in the original variables as possible**, while the **second component accounts for as much of the variance in the original variables as possible, but subject to the fact that it is orthogonal to the first component**. Notice that in the figure, the axes are perpendicular to one another, which is another way of saying that they are at 90° angles. This is the condition of orthogonality.

10.1 Principal Component Analysis Versus Factor Analysis

Before we begin, we need to issue a caveat and warning. In the next chapter, we will be studying something called **exploratory factor analysis (EFA)**. For a variety of reasons, PCA and EFA are often relegated to being the same or at least

strikingly similar techniques. You may hear in your communications with some such things as "Oh, whether you do a component analysis or a factor analysis you'll get the same results." However, it is imperative, from the outset, that you do not equate component analysis with factor analysis, as they are not the same procedure, and there are very important and key distinguishing features between them. We cannot discuss these important differences yet, because we haven't surveyed either technique sufficiently, but we will a bit later after we have some of the basics under our belts. Until then, however, take it on trust that the two techniques are sufficiently different that they merit their own separate chapters.

Principal component analysis (PCA) and exploratory factor analysis (EFA) are not equivalent procedures and should not be treated as such. There are key differences between them and the kinds of substantive (scientific) conclusions that can be drawn from each.

10.2 A Very Simple Example of PCA

PCA is usually applied to data sets that contain many variables, sometimes hundreds, since it is for these data sets especially that we wish to reduce dimensionality. For example, if our data set has 1000 variables, then PCA might be useful to reduce the dimensionality down to 2 or 3 dimensions (if the procedure were successful). In this way, data like this makes PCA a potentially suitable analytic approach because of the numerous variables or dimensions we start out with. Other times we wish to conduct a PCA on data sets with a smaller number of variables, for instance, 5–10, to see if we can likewise reduce or uncover the dimensionality of the data. Regardless of how many variables you may be starting out with, PCA may be a suitable technique for your data.

To understand PCA from first principles, however, it is very useful and pedagogical to start off with a very easy example featuring only two variables. Why begin with an example with only two variables? We do so because it allows one to appreciate a bit more what PCA actually **does** instead of getting lost in a more complex example where we may fail to appreciate its underlying mechanics. With techniques such as PCA and EFA, it is far too easy to get immersed in one's substantive theory and coincidentally conclude exactly what you set out to find! That's why it's important to study PCA on its

own merits first and foremost, so one can appreciate what the technique can versus cannot accomplish.

What is more, our first example of a PCA is actually one featured by a pioneer of component analysis, Karl Pearson, infamous statistician who first introduced the technique in a paper written in 1901. Once we can appreciate how PCA works on such a small sample, we will be in good shape to consider even the most complex of data sets on which component analysis may be well-suited.

10.2.1 Pearson's 1901 Data

As our first easy example of PCA then, consider the data given by Karl Pearson in 1901 on two variables, *x* and *y*. We will designate these variables as simply *x* and *y* without giving them substantive or research names, so we can observe quite simply what PCA does with these variables:

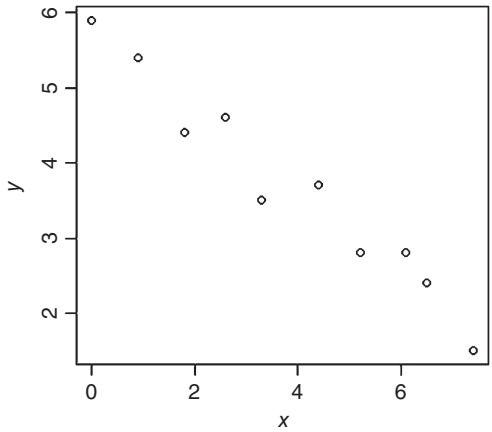
x	*y*
0.00	5.90
0.90	5.40
1.80	4.40
2.60	4.60
3.30	3.50
4.40	3.70
5.20	2.80
6.10	2.80
6.50	2.40
7.40	1.50

We first create a data frame in R for these two variables:

```
>x<-c(0, 0.9, 1.80, 2.60, 3.30, 4.40, 5.20, 6.10, 6.50, 7.40)
>y<-c(5.90, 5.40, 4.40, 4.60, 3.50, 3.70, 2.80, 2.80, 2.40, 1.50)
> pca.data <- data.frame(x, y)
> pca.data
     x   y
1  0.0 5.9
2  0.9 5.4
3  1.8 4.4
4  2.6 4.6
5  3.3 3.5
6  4.4 3.7
7  5.2 2.8
8  6.1 2.8
9  6.5 2.4
10 7.4 1.5
```

Let's obtain a plot of these data:

```
> plot(x, y)
```



These data are, of course, in two dimensions, *x* and *y*. The question PCA asks is:

Can we transform these variables on two dimensions into two new variables also on two dimensions, but such that most of the variance in the variables is accounted for by the first dimension only?

That is, as mentioned at the outset, PCA will seek to perform a transformation on the variables, with the hope that the first dimension will account for most of the original total variance. Be careful to note here that nowhere have we yet reduced the number of dimensions. In the transformation, **PCA will generally generate as many dimensions as there are original variables** in the data set. However, the hope is that most of the variance in the original variables will be accounted for by the first dimension in this case. Had we 100 original dimensions, we would have likewise wanted to transform these into 100 new dimensions, with the hope that most of the variance in the original dimensions (variables) could be accounted for by the first few components. This is the nature of PCA – it can be interpreted as nothing more than a transformation of the original axes onto new dimensions. The characteristics of these new dimensions (components) and how many of these we may choose to retain is another story (a story we will unpack shortly).

To perform the PCA, we first need something to represent how the variables *x* and *y* **covary** together. For this, we will build the **covariance matrix** of *x* and *y*. In R, we will call this object by the name of `A`:

```
> A <- cov(pca.data)
> A
          x         y
x  6.266222 -3.381111
y -3.381111  1.913333
```

Recall the nature of a **covariance matrix**. It contains variances of the variables along the main diagonal going from top left to bottom right. For our data, the variance of x is equal to 6.26, and the variance of y is equal to 1.91. The covariances between x and y are given in the off-diagonal of the matrix. We can see in the matrix that the covariance between x and y is equal to −3.38. Notice these numbers are the same in the matrix, since there is only a single covariance between x and y. That is, the number in the bottom left of the matrix is the same as the number in the upper right. The covariance matrix is an example of a **symmetric matrix**, meaning that the upper triangular part of it is the same as the lower triangular part. The reason why we have built a covariance matrix of x and y is because in a moment we will ask R to perform a component analysis directly on this matrix.

For demonstration, and to confirm that we created our covariance matrix correctly (as well that R computed it correctly), we compute the variances and covariance of x and y manually in R. First, we confirm that the variances have been computed correctly:

```
> var(x)
[1] 6.266222
> var(y)
[1] 1.913333
```

We see that the variances computed match those in the covariance matrix. We could have also used the `diag()` function on the matrix to obtain the values along the main diagonal, which of course correspond to the above variances:

```
> diag(A)
       x        y
6.266222 1.913333
```

Next, let's compute the covariance:

```
> cov(x,y)
[1] -3.381111
```

We confirm that the covariance of -3.38 matches that computed in the covariance matrix.

10.2.2 Assumptions of PCA

Because PCA does not invoke a statistical model as do most other methods in this book, used descriptively at least, the assumptions for PCA are quite limited. These are best not regarded as true "assumptions" at all such as we would have

in ANOVA or regression, for example, where we are estimating parameters. As discussed, PCA at its core is simply a transformation of data onto new dimensions. No assumptions are needed to perform such a transformation. However, certain conditions should be in place before such a transformation can be performed. For one, the covariance or correlation matrix should contain sufficient covariance or correlation to make the PCA worthwhile. For instance, if the correlation matrix were an **identity matrix**, having values of 1 along the main diagonal and zeroes everywhere else, then performing PCA would make little sense.

We can verify that the correlation matrix is not an identity matrix by conducting what is known as **Bartlett's test of sphericity**. To run the test, we can use the **rela** package, and will employ the function `paf()`. Along with this test will also be reported the **Kaiser–Meyer–Olkin (KMO) measure of sampling adequacy**. The KMO is a measure of the likely utility of conducting a factor analysis in the sense of providing an estimate of the proportion of variance in variables that might be due to underlying factors. It is less relevant for components analysis, though we nonetheless demonstrate it here in conjunction with Bartlett's. Values higher than 0.70 (or so) are preferred, though even if one obtains a much smaller value (we obtain 0.5 below), this does not imply somehow that the analysis should absolutely not be conducted. We adapt our code again from Schumacker (2016):

```
> library(rela)
> paf.pca = paf(pca.data,eigcrit=1, convcrit=.001)
[1] "Your dataset is not a numeric object."
```

You may receive this error message when attempting to run the `paf()` function. To make things work here, simply convert the data frame `pca.data` to a matrix in this case, then try the `paf()` function again on this new matrix:

```
> pca.data.matrix <- as.matrix(pca.data)
> paf.pca = paf(pca.data.matrix,eigcrit=1, convcrit=.001)
> paf.pca

$KMO
[1] 0.5

$Bartlett
[1] 23.013
```

We have printed only partial output from the `paf()` function, revealing the KMO and Bartlett values. KMO is a bit low at 0.5, but again, there is nothing wrong

with proceeding with the PCA/factor analysis anyway. Bartlett's is equal to 23.013. To get a significance test on this, we compute (Schumacker, 2016):

```
> cortest.bartlett(pca.data.matrix, n = 10)
$`chisq`
[1] 23.013

$p.value
[1] 1.6092e-06

$df
[1] 1
```

The null hypothesis for Bartlett's is that the correlation matrix is an **identity matrix**. The alternative hypothesis is that it is not. Since we have obtained a very small *p*-value, we can safely reject the null and conclude the alternative that the matrix is not an identity matrix. That is, we have evidence to suggest we have sufficient correlation in the matrix to proceed. Again, however, it needs to be emphasized that these measures are simply guidelines and nothing more. There is nothing inherently wrong with performing a component/factor analysis on data that do not meet the above guidelines. In a matrix that resembles an identity matrix, for instance, the resulting analysis will likely not be successful, but that does not mean the procedure cannot still be run to confirm this. The above tests do not by themselves restrict one from running the procedure.

10.2.3 Running the PCA

Having produced our covariance matrix, we are now ready to run the PCA in R. We will first use R's `princomp()` function to obtain PCA results, then later demonstrate an alternative function in R to obtain the PCA:

```
> pca <- princomp(covmat = A)
> summary(pca)

Importance of components:
                          Comp.1      Comp.2
Standard deviation     2.8479511 0.262164656
Proportion of Variance 0.9915973 0.008402695
Cumulative Proportion  0.9915973 1.000000000
```

We summarize the above output:

- The standard deviation of the first component is equal to 2.8479511. To get the variance, we square the standard deviation and obtain 8.11. Directly below this number, the `Proportion of Variance` accounted for by this first component

is 0.99, which for the first component, is also equal to the cumulative proportion as well.

- The standard deviation of the second component is equal to 0.262164656. To get the variance, we again square the standard deviation and obtain 0.0687. Directly below this number is the proportion of variance accounted for by this second component, equal to 0.008. The cumulative proportion across both components is 1.0.

How were the above numbers computed? The total variance in the variables is equal to their sum, so when we sum the variance across the two variables, we have: 6.266222 + 1.913333 = 8.18. The first component accounts for a certain proportion of this total variance. Since its variance is 8.11, it accounts for a proportion of 8.11/8.18 = 0.99. Likewise for the second component, it accounts for a proportion of 0.0687/8.18 = 0.008. Hence, notice what the PCA has done – it has taken the two original variables or dimensions, and transformed them into two new dimensions on which the variance of the original variables has been **redistributed** across the new dimensions. **Clearly, the PCA has redistributed most of the variance in the original variables to the first component, such that instead of having to interpret both original variables, we can accomplish pretty much just as much by interpreting only the first component without loss of much "information."** This is the essence and purpose behind PCA. The total variance has been preserved, but now most of it is accounted for by the first derived component. This is why the first component is called, appropriately, the "principal" component, in that it accounts for most of the variance in the original variables, while, as we will soon discuss, being **orthogonal** with remaining components.

Of note, we could have also gotten the PCA by specifying the variables in our analysis directly rather than coding the covariance matrix, using `prcomp()`:

```
> fit.pca <- prcomp( ~ x + y)
> fit.pca
Standard deviations (1, .., p=2):
[1] 2.8479511 0.2621647

Rotation (n x k) = (2 x 2):
         PC1         PC2
x  0.8778562 -0.4789243
y -0.4789243 -0.8778562
```

Notice in the above code that the function `prcomp()` is acting not on a covariance matrix in the call statement, but rather directly on the variables x and y. Regardless of the approach, the result is the same. However, as one can imagine, especially for data sets with a very large number of variables, listing each variable

as we did above by ~ x + y would be very time consuming for anything other than a minimal number of original variables, and so constructing the relevant covariance matrix is usually a preferable strategy.

10.2.4 Loadings in PCA

Now we said that PCA performed a transformation on the original variables such that most of the variance in the original variables is now attributed to the first component, while a much smaller amount of variance in the original variables is attributed to the second component. The next question is,

What is the nature of these new components?

That is, we wish to know their "make-up." What is the first component actually composed of? What is the second component composed of? It would make sense that the new components are actually made of, in some sense, the original variables. When you think about it, there really isn't any other option, since the new components can only be composed of variables originally submitted to the component analysis. This fact in itself is another example of why **the most important part of any statistical analysis are the variables you subject to that analysis**. So, when we are asking the question about the make-up of a component, that component can only be made up of the variables you derived it from. Now, this can seem like a no-brainer, but it is a no-brainer that is often overlooked and underappreciated. If you add or subtract one or more variables from the original set, it stands that the component structure will change. Hence, when you draw a conclusion from your component analysis that you have "found" a component in your data, that "finding" is only as good as the variables subjected to the component analysis, and not independent of it. Obvious, right? But very important to remember. Again, component analysis is not "discovering" components, it is merely issuing a transformation of the original information subjected to it. It, as is true of any statistical technique, is only as good as the data on which it is applied.

The most important element of virtually any statistical analysis, including principal component, is the data subjected to that analysis. If you add one or two or more variables, the solution will likely change, and so will your substantive conclusions. The selection of variables and quality of data, which includes measurement and psychometric properties, is always paramount to any statistical analysis. Statistical analyses cannot change bad data into good data.

The new components are what are referred to as **linear combinations** of the original variables. Though the idea of these linear combinations is similar in spirit to that studied in MANOVA and discriminant analysis, the nature of these linear combinations is different, in that they have different properties than those featured in discriminant analysis. Hence, be sure not to equate linear combinations in one procedure with linear combinations in the other since they are not constructed the same way. The linear combinations in discriminant analysis are constructed such that they maximize group differences on the grouping variable. The linear combinations in PCA are constructed such that they maximize variance on the first few components extracted. The degree to which the original variables "contribute" to each of the respective components is measured by the extent to which each original variable "loads" onto that component. This will make much more sense in a moment. For now, let's ask R to generate loadings of the PCA we performed earlier:

```
> pca$loadings

Loadings:
  Comp.1 Comp.2
x  0.878  0.479
y -0.479  0.878

               Comp.1 Comp.2
SS loadings       1.0    1.0
Proportion Var    0.5    0.5
Cumulative Var    0.5    1.0
```

We summarize what the above tells us:

- Component 1 has a loading for x equal to 0.878, and a loading of −0.479 for y. That is, the first component is the following linear combination:

$$\textbf{Component 1} = \mathbf{0.878}(x) - \mathbf{0.479}(y)$$

- Component 2 has a loading for x equal to 0.479, and a loading of 0.878 for y. That is, the second component is the following linear combination:

$$\textbf{Component 2} = \mathbf{0.479}(x) + \mathbf{0.878}(y)$$

- The SS loadings for each component refer to the sums of squared weights for each column of loadings. One constraint of PCA is that the sum of these squared loadings for each component sum to 1.0. This is a mathematical constraint set by

design in the procedure so that the variances of components cannot grow without bound. That is, for components 1 and 2, respectively, we can verify in R:

```
> (0.878)^2 + (-0.479)^2
[1] 1.000325
> (0.479)^2 + (0.878)^2
[1] 1.000325
```

10.3 What Are the Loadings in PCA?

You might be asking at this point how the above loadings are obtained. We know they make up elements of each component, but we have not yet discussed how they are actually derived. To understand how they are derived, we need to return to the concepts of eigenvalues and eigenvectors first discussed in Chapter 2. Though technically speaking there are differences between how `prcomp()` and `princomp()` obtain eigenvalues and eigenvectors (`prcomp()` uses a technique called **singular value decomposition**, while `princomp()` uses eigen decomposition), a common theme is the extraction of eigenvalues and eigenvectors, which we briefly review with reference to eigen decomposition.

Recall that for every square matrix $\mathbf{A}$, it is a mathematical fact that a scalar λ and vector $\mathbf{x}$ can be obtained so that the following equality holds true:

$$\mathbf{Ax} = \lambda\mathbf{x}$$

What this meant is that as we perform a transformation on the vector $\mathbf{x}$ via the matrix $\mathbf{A}$, this equated to transforming the vector $\mathbf{x}$ by multiplying it by a "special scalar" λ. This scalar we called an **eigenvalue** and $\mathbf{x}$ was referred to as an **eigenvector**.

In this brief recollection of eigen analysis, we have effectively summarized the technical underworkings of PCA. When using eigen decomposition, PCA features an **eigen analysis**, not on any arbitrary matrix, but rather on the covariance matrix containing the variables we subjected to the PCA. There is a wealth more to learn about eigen analysis, and the interested reader is encouraged to consult any book on linear algebra for more details, or a comprehensive book that discusses the theory of multivariate analysis more extensively, such as Johnson and Wichern (2007) or Rencher and Christensen (2012). **To the mathematician,**

many multivariate statistical methods reduce, on a structural level, to eigen analysis.

In R, we can demonstrate the computation of the relevant eigenvalues and eigenvectors that we obtained in the above PCA by requesting an eigen decomposition directly:

```
> eigen(A)
eigen() decomposition
$`values`
[1] 8.11082525 0.06873031

$vectors
           [,1]       [,2]
[1,] -0.8778562 -0.4789243
[2,]  0.4789243 -0.8778562
```

We can see that the eigenvalues generated of 8.11 and 0.068 match those obtained from our PCA. We can also see that other than a change in sign, the magnitude of the loadings are the same. Again, this is as far as we take the topic of eigenvalues and eigenvectors in PCA. The interested reader is referred to the aforementioned sources for more details. The key point for now is to appreciate how universal eigen analysis is across both mathematics and the sciences, and is the workhorse of **dimensionality reduction**.

10.4 Properties of Principal Components

In our example above, we derived the components, but we only touched on a few of the properties specific to them. We now summarize and extend on that discussion here. Components obtained in a PCA obey the following properties:

1) **The sum of the obtained eigenvalues of the derived components from the covariance matrix matches the sum of the original variances of the raw variables.** For our data, recall the sum of the eigenvalues was equal to 8.18. This number is the same that we obtain when summing the original variances, those of 6.27 and 1.91 = 8.18. Why is this property important? It is vital to understanding what PCA does because it reveals that PCA does not change the amount of "information" in the original variables in terms of how much variance they account for, it merely transforms this information (variance) onto new dimensions. That is, **principal components preserves**

the original total variance in the variables. It simply redistributes this variance across new components, that is, across new dimensions.

2) **Successive components (which recall, are linear combinations of the original variables) are orthogonal**. When we obtain principal components, the first component obtained is that which explains the maximum amount of variance in the original variables as possible. This is the first principal component. The second component obtained explains the maximum amount of variance in the original variables, however, subject to the constraint that it be orthogonal to the first component. In this way, extracted components do no overlap.
3) **The sum of squared loadings for each component equals 1.0.** That is, the variance of each component is maximized subject to the constraint that the length of each component be equal to 1.0. Why is this constraint relevant? It is important because if we did not subject the components to some constraint such as this, then in theory, the **variance of the given component could grow without bound**. That is, the variance of the component could get larger and larger if we did not have some way of regulating it. Constraining the sum of squared loadings to equal 1.0 is one way of making sure the variance of each component is "kept in check" in some sense. One caveat: in some software programs such as SPSS, the sum of squared loadings in its program for PCA is typically not equal to 1.0, but rather will equal the corresponding eigenvalue for the given component. There are reasons for this that are beyond the scope of this book (they relate to how SPSS subsumes PCA under the technique of factor analysis rather than its own independent method) but it's the principle that is worth remembering for now, and that is putting some kind of constraint on the length of the eigenvector.

10.5 Component Scores

Having generated two components, we now wish to obtain **component scores** on each one. That is, for any given case in our data, we wish to know the component score for that given individual on each component. We can obtain component scores by computing them manually on each component as a linear combination of the observed variables x and y. For the first component then, we compute:

```
> comp.1 = 0.8778562*(x-3.82) -0.4789243*(y-3.7)
> comp.1
```

```
[1] -4.4070441 -3.3775114 -2.1085165 -1.5020164 -0.3607004
0.5091566
[7] 1.6424734 2.4325440 2.9752562 4.1963587
```

Note carefully how the above was computed: we simply weighted x and y by the corresponding weights for the eigenvector designated for that component. We subtracted the mean of each variable in doing so, because by default, `prcomp()` typically computes components by first centering x and y to have mean zero (James et al. 2013).

Likewise, we can compute scores for the second component:

```
> comp.2 = 0.4789243*(x-3.82) +0.8778562*(y-3.7)
> comp.2

[1]  0.10179281  0.09389658 -0.35292775  0.20578293
-0.42461188  0.27777609
[7] -0.12915505  0.30187682  0.14230406 -0.21673465
```

Now if we have computed these components correctly, they should have variances equal to the eigenvalues we extracted from the PCA. We can easily check this by computing the variances of each component:

```
> var(comp.1)
[1] 8.110825
> var(comp.2)
[1] 0.06873031
```

Notice that the above variances for each component match the respective eigenvalues from the PCA.

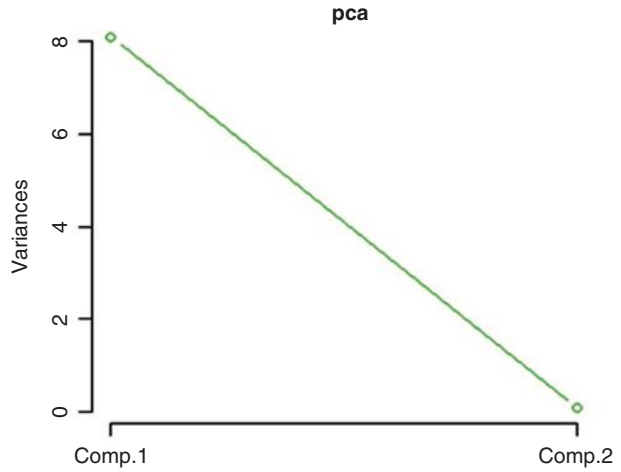
10.6 How Many Components to Keep?

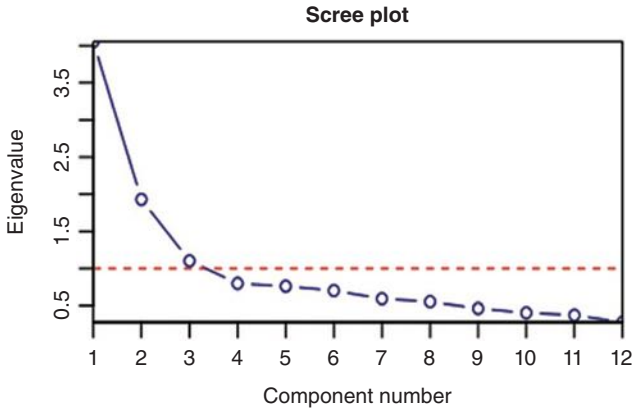
10.6.1 The Scree Plot as an Aid to Component Retention

The big question in PCA from a **scientific** rather that statistical perspective, is usually how many components to keep. This is probably the most substantively difficult part of the entire PCA, because **you cannot rely on the statistical method to tell you how many you should keep**. Instead, the decision is

often made rather **subjectively**, but with the aid of some statistical tools. One such tool is the **scree plot**, a graphical device that plots the eigenvalues (variances) for each component. In a scree plot, one looks for a "bend" in the plot to help determine the number of components to retain. We can obtain a scree plot for our data as follows:

```
> screeplot(pca, type="lines", col=3)
```





Now of course, having only two components possible for the current analysis, the first plot does not truly reveal any real "insight" into how many components to keep. There is no bend in the plot when there are only two components. Again, however, with such a simple example, this makes it possible to see what the scree plot actually accomplishes, which is simply graphing the eigenvalues for each component. By itself, it doesn't really tell us much more than that. When we conduct an analysis with more than two components a bit later, the plot will look a bit more complex. The second plot, for instance, is a scree plot from a more complex analysis. Arguably, the bend occurs roughly at an eigenvalue of approximately 2 or 3, which according to **Kaiser's rule**, would recommend to keep components with eigenvalues greater than 1.0 when analyzing a correlation matrix. The logic behind Kaiser's rule, that of keeping components with eigenvalues greater than 1.0 in a correlation matrix, is that any component having an eigenvalue more than 1.0 accounts for more than the average total variance (since each variable contributes a unit variance). In a correlation matrix, the average variance is equal to 1.0. In a covariance matrix, this will not necessarily be the case, but the rule can still be applied, that of retaining components that exhibit greater that average variance (whatever that number turns out to be, it will likely not be 1.0 as was true for the correlation matrix). Kaiser's rule is a guideline at best, and in no way should be used as "hard evidence" regarding how many components to retain. If as a researcher you're retaining components based only on Kaiser's rule, your thought process regarding component analysis is incomplete.

10.7 Principal Components of USA Arrests Data

We demonstrate another PCA, this time on the **USA Arrests** data, which is a data frame available in the base package in R. The variables in this data sets are `murder`, `assault`, `urbanpop`, and `rape`. The cases in the analysis are the states. Following James et al. (2013), we detail the PCA for these data, and refer you to their analysis (pp. 373–385) to highlight one important element of principal components, that **they are unique only up to a change in sign**. Hence, the positives and negatives on component loadings could change depending on what programs are used to obtain the components (as we saw earlier), but this is not indicative of results being "wrong" or

contradictory. This is what we mean by saying **the components are unique, but only up to a change in sign**.

Before conducting the analysis, let's get a look at our data:

```
> head(USArrests)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

Let's obtain a matrix of correlations for these data:

```
> cor(USArrests)
             Murder   Assault   UrbanPop      Rape
Murder   1.00000000 0.8018733 0.06957262 0.5635788
Assault  0.80187331 1.0000000 0.25887170 0.6652412
UrbanPop 0.06957262 0.2588717 1.00000000 0.4113412
Rape     0.56357883 0.6652412 0.41134124 1.0000000
```

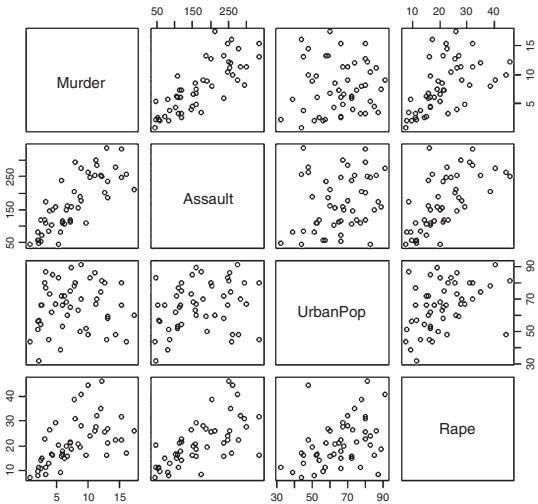
We can also get the correlations individually if we wanted, by specifying the data frame, then requesting which variables we want to correlate:

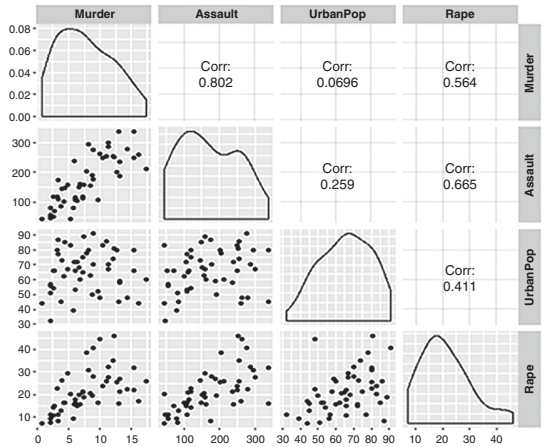
```
> cor(USArrests$Murder, USArrests$Assault)
[1] 0.8018733
```

Recall that the function of `$` here is to select the given variables from the data frame that we wish to correlate.

Inspecting a correlation matrix of numbers is one thing, but an even more powerful way to interpret data is via **visualization**. Below we produce the corresponding scatterplots of these correlations, as well as a more elaborate plot containing both bivariate and univariate information (down the main diagonal):

```
> pairs(USArrests)
> library(GGally)
> ggpairs(USArrests)
```


Murder
Assault
UrbanPop
Rape


Murder
Assault
UrbanPop
Rape
Corr:
0.802
Corr:
0.0696
Corr:
0.564
Corr:
0.259
Corr:
0.665
Corr:
0.411

We use the `prcomp()` function to get the PCA:

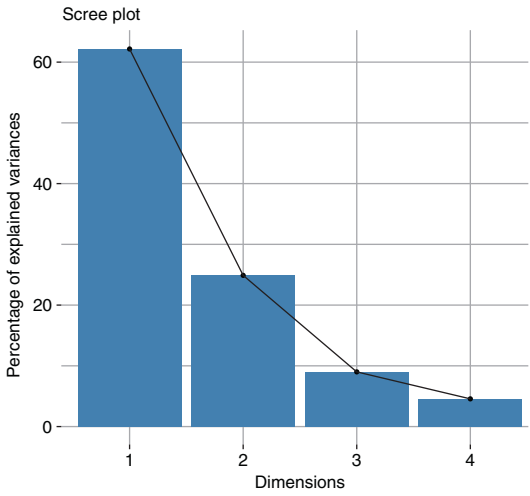
```
> usa.comp <- prcomp(USArrests, scale=TRUE)
> usa.comp
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

- By specifying `scale=TRUE`, we've requested that the analysis be done on variables with standard deviation 1. That is, the variables are scaled to have unit variance, and by default `prcomp()` centers the variables to have mean zero (James et al. 2013). The analysis is being performed on the **correlation** matrix rather than the **covariance** matrix.
- Four variables were subjected to the analysis, and so four components were generated, each having standard deviations equal to 1.57, 0.99, 0.59, and 0.41.
- The loadings for each component are given next. It would appear at first glance that `murder`, `assault`, and `rape` load moderately well on PC1, while `urbanpop` does not, but does load strongly on PC2. `Rape` loads quite high on PC3, and `murder` and `assault` load fairly high on PC4.

Next, we obtain a scree plot of the component analysis (Kassambara, 2017).

```
> library(FactoMineR)
> library(factoextra)
> scree <- fviz_eig(usa.comp)
> scree
```



We can see from the plot that the first component extracted indeed accounts for much of the variance in the original variables.

10.8 Unstandardized Versus Standardized Solutions

PCA can be performed on either the original **unstandardized** data, or on data **standardized** to have mean equal to zero and standard deviation equal to 1. This transformation is equivalent to analyzing the **correlation matrix** (standardized data) rather than the **covariance matrix** (Izenman, 2008).

The choice of whether or not to standardize is not a simple one, and the decision to standardize should not be made without careful consideration. If the original variables are not measured in the same units, and have wildly different variances and are hence not considered "commensurate," then this could have a rather dramatic effect on the PCA solution. The loadings for the variables with the greatest variance may be exaggerated in a sense since they contribute the most original variance. However, as noted by Rencher and Christensen (2012):

> Generally, extracting components from **S** [covariance matrix] rather than **R** [correlation matrix] remains closer to the spirit and intent of principal component analysis, especially if the components are to be used in further computations ... However, in some cases, the principal components will be more interpretable if **R** is used. (pp. 419–420).

As noted by Jolliffe (2002), standardization does have the potential drawback of making statistical inference with regard to components more difficult, though since PCA is typically used as a descriptive rather than inferential tool, this should not by itself impede the decision to standardize. Other authors on the subject are even more adamant about virtually always standardizing:

> Because it is undesirable for the principal components obtained to depend on an arbitrary choice of scaling, we typically scale each variable to have standard deviation one before we perform PCA. (James et al. 2013, p. 381)

As a demonstration of how a change in scale can influence the results of a PCA, consider a comparison of component analyses on the USArrests data analyzed by covariance versus correlation matrix. To analyze on the covariance rather than the correlation matrix (as we did earlier), we specify `scale = FALSE`:

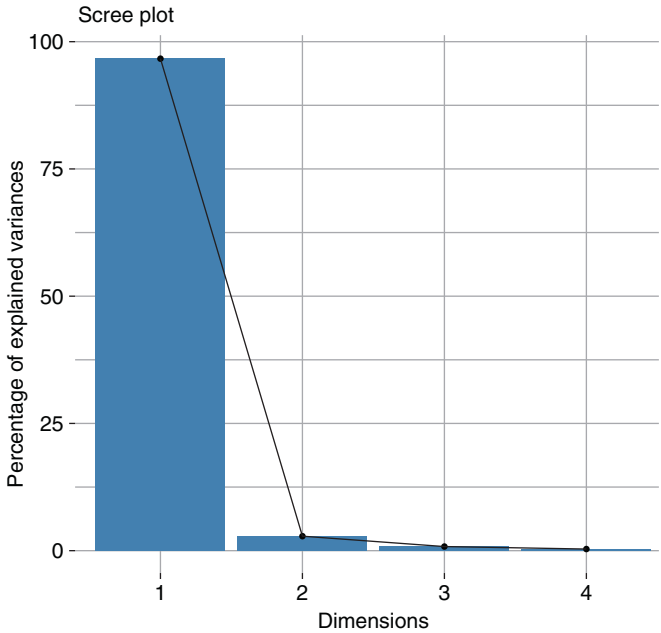
```
> usa.comp.cov <- prcomp(USArrests, scale=FALSE)
> usa.comp.cov
```

```
Standard deviations (1, .., p=4):
[1] 83.732400 14.212402  6.489426  2.482790

Rotation (n x k) = (4 x 4):
                PC1         PC2         PC3         PC4
Murder   0.04170432 -0.04482166  0.07989066 -0.99492173
Assault  0.99522128 -0.05876003 -0.06756974  0.03893830
UrbanPop 0.04633575  0.97685748 -0.20054629 -0.05816914
Rape     0.07515550  0.20071807  0.97408059  0.07232502
```

We immediately note that the solution is not the same as when the correlation matrix was analyzed. The resulting scree plot highlights this difference even more. We see below that the first component extracted accounts for much more variance in the unstandardized solution.

```
> scree.cov <- fviz_eig(usa.comp.cov)
> scree.cov
```



So what's the answer? To standardize or not? Like so many questions in statistics and mathematical modeling in general, there is usually no definitive answer. The safest route is probably to standardize, but as emphasized by Rencher, the

covariance matrix is probably more true to the spirit of PCA. If in doubt, there is nothing wrong with running **both solutions**, comparing them, and allowing your readers or audience to see the differences. Above all, the question of whether or not to standardize is pedagogically meaningful, because it highlights, in a strong sense, how such decisions can have huge impacts on results. It serves to emphasize that when it comes to statistics and data analysis, what you get often depends on what you go in with. It's not quite the same as an archeological dig, for example, and should never be treated as such. **The instrument you use to do the digging often influences the nature of what is found. Rarely can a statistical model, by itself, uniquely characterize reality. Statistical analysis does not discover "truth," it merely transforms, translates, and summarizes data subject to the constraints we impose on it**.

It is also one reason why no **single analysis** will provide you with complete definitive insight. The issues of whether to standardize or not, though of course a technical issue, quickly become philosophical as well, in that if the input matters so much to the output or what is "discovered," then it would seem that independent criteria should exist for determining the input. However, across most statistical methods, such independent criteria often do not exist. A similar issue arises, as we will see, in cluster analysis, in which choice of **distance measure** can have rather dramatic effects on the ensuing cluster solution, and often there is no definitive recommendation over which distance measure to use in all situations or all circumstances.

If you do wish to standardize, R offers a convenient way to translate between matrices using the `cov2cor()` function if you wish to specify the correlation rather than covariance matrix directly. As an example, recall the covariance matrix A featured earlier on variables *x* and *y*. Below we transform that matrix into a correlation matrix. We name the new matrix `R`:

```
> A
        x       y
x  6.2662 -3.3811
y -3.3811  1.9133
```

The corresponding correlation matrix is the following:

```
> R = cov2cor(A)
> R
         x        y
x  1.00000 -0.97648
y -0.97648  1.00000
```

The **psych** package also has a utility for performing component analysis using the function `principal()`, though not demonstrated here.

Exercises

1 Describe the purpose of principal component analysis, and define what is meant by a principal component. First define it technically, then discuss how it may be useful substantively.

2 Why is it true that PCA will generally generate as many components as there are original variables? Why does this make sense?

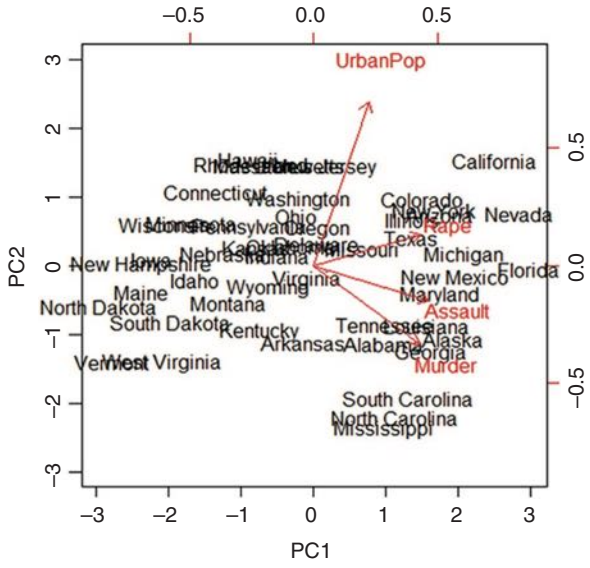
3 What does it mean to impose a constraint on components such that their length is equal to 1.0? Why is this constraint relevant and important? What is the consequence if such a constraint is not enforced?

4 Unpack the equation for an eigen analysis. What does the equation for eigenvalues and eigenvectors communicate?

5 Describe how the second component in a PCA is extracted. What are the two conditions it must satisfy?

6 For the component analysis conducted on the USArrests data, attempt to give meaning to the 4 components extracted. Are you able to describe what they might represent?

7 Explain what is being communicated by the following plot (known as a "biplot") generated on the USArrests data of this chapter:



8 Briefly discuss the decision regarding whether to analyze the covariance or correlation matrix in a principal component analysis. When might one solution be more appropriate than the other?

9 Perform a principal component analysis on the iris data's variables `sepal length` through to `petal width` (exclude `Species`), and summarize your findings. How many components were extracted? How many did you decide to keep? Why? Obtain a scree plot of the solution. Does the scree plot help in your decision?

```
> head(iris)
  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
6          5.4          3.9           1.7          0.4  setosa
```

10 Obtain pairwise correlation plots for the variables subjected to the PCA in #9 using the following, and describe both the univariate and bivariate distributions that appear:

```
> pairs()
> library(GGally)
> ggpairs()
```