There actually are two functions in R for Principal component analysis: `princomp()` and `prcomp()`. The difference is that `princomp()` calculates the eigenvectors of the association matrix (covariance or correlation), whereas `prcomp()` performs something called singular-value decomposition of the data matrix. Although both should give the same result, the singular value decomposition approach is preferred so you should use `prcomp()`.

PCA requires a single numeric data matrix as input. The standard format is to have the samples as the row names and the measurements or counts as the column names. The data files you have been using for this class are all set up in such a format. If you do need to swap the rows and columns of a matrix or data frame, that action is called transposing the matrix and the R function is just `t()`. For example, `t(matrix_name)` will transpose the matrix so the rows become the columns and the columns become the rows.

Most of the data files you have will be data frames, with a mixture of numeric and text columns, so you’ll need to obtain just the numeric columns. You could use `cbind()`, but there likely will be many columns so it may be a bit unwieldy. The simplest solution is to specify the columns by their number, in square brackets after the variable name. For example:

```
georoc[,3:29] #just numeric columns 3 through 29
```

The format for `[]` is row, column and 3:29 denotes the sequence of integers from 3 to 29.

The main choice you need to make is whether to use the covariance matrix (if variables measured on the same scale and differences in their variance are meaningful) or the correlation matrix (variables measured in different units so differences in variance are unimportant). The covariance matrix is the default, so to use the correlation matrix:

```
prcomp(matrix, scale=T)
```

The option is called “scale” because the correlation matrix is the same as scaling the variables to all have the same variance.

PCA won’t work if there are missing (NA) values in the data set, so you’ll need to remove them using `na.omit()`. If you only plan to use a few columns from a larger matrix or data frame, be cautious when you use `na.omit()` because it will remove all rows where any column contains NA values. You can also specify `na.omit=T` within the `prcomp()` function, but I don’t advise this approach because it won’t be possible to match the points in the PCA results with categories or other information from the raw data set.

You should store the output of the `prcomp()` function as a variable (I’ll use `pca_res` as an example here).

Use `summary(pca_res)` to get the proportion of variance explained by each principal component axis. This information is quite important, because it describes how important each axis is, and it should be reported as part of the axis label in any PCA plot.
The results also contain the standard deviation (not variance) explained by each Principal Component, as \( \text{pca_res$sdev} \). You can make the scree plot by graphing the standard deviation values squared: \( \text{plot(pca_res$sdev^2)} \).

The most important parts of the PCA results variable are the axis scores (the coordinates of each sample on the principal component axes) and the loadings (the directions in which the original variables increase). For some reason, the axis scores are cryptically named “x.” The loadings are named “rotation.”

You can plot the PCA results using \( \text{biplot(pca.res)} \), but the resulting graph is ugly, has weird axis scales, and you can’t customize the color or symbols of the points to denote different categories. You should construct your own plot from the data in \( \text{pca_res$x} \) and \( \text{pca_res$rotation} \).

Creating plot with points

You have already learned to make an x-y scatterplot, how to change the axis labels, and how to modify the point colors or symbols to denote categories. Remember, to color-code points, use:

\[
... \text{col} = \text{c(“red”, “blue”, …)[data_frame$factor]}\]

If you need to refresh your memory, refer back to the instructions for correlation and regression. Your axis labels should always be something like “PC1 (37.4% of variance explained)” (use the actual percent variance explained).

Adding loadings arrows

To add the loadings, use the \( \text{arrows()} \) function. It requires four numbers as the input (the x,y position of the starting and ending coordinates):

\[
\text{arrows(x0, y0, x1, y1)} \# \text{in the order } x_{\text{start}}, y_{\text{start}}, x_{\text{end}}, y_{\text{end}}.\]

For PCA, the arrows all start at 0, 0 and the ending coordinates will be the first two columns in \( \text{pca.res$rotations} \) (if you are plotting PC1 vs. PC2). You may need to multiply the ending x,y coordinates by some arbitrary number to either shrink or stretch the arrows for visibility.

The default arrowhead size is typically too big for a PCA plot, so I recommend changing it. The \( \text{length=} \) parameter is an optional parameter within the \( \text{arrows()} \) function and can be specified any place after the x,y coordinates. I recommend \( \text{length}=0.15 \). You can also use \( \text{col=} \) to change the color or \( \text{lwd=} \) to change the line width.

Adding loadings text

You’ll also need to add the labels for the loadings. The function to add text to a plot is simply called \( \text{text()} \). It requires three inputs, a vector of the x coordinate position(s), a vector of the y coordinate position(s), and the list of the labels to be used. In the case of PCA loadings, the x and y coordinates will just be the column values in \( \text{pca.res$rotation} \), although you should multiply them by a small increment (like 1.1) so they don’t plot exactly on top of the
arrowheads. You can get the loading names from `rownames(pca.res$rotation)`. Text color can be varied with `col=...` and the text size can be changed with `cex=...` (cex is a multiplier, so 0.75 makes it 75% as big and 1.5 makes it 50% larger).