Fisher’s Iris Data

We’re going to look at the iris dataset, that is Fisher’s Iris data that was previously discussed in class. This data is contained in the datasets package, which should already be installed and loaded into R by default. We will perform PCA on this data. To investigate the data enter:

```
?iris
```

```
## starting httpd help server ... done
dim(iris)
```

```
## [1] 150  5
```

```
names(iris)
```

```
```

```
head(iris, n = 5)
```

```
##          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
```

The `names` function returns the variable (column) names of the iris data frame. The `head` function displays the first observations (n = 5 in this case) in the data. What do you notice about the variables in the iris data set?

The function `prcomp` can be used to performing PCAs. Remember to check this function’s help file. We can use `prcomp` to directly apply PCA to a dataset, instead of using the `cov` and `eigen` functions separately, like we did in last week’s lab. We’ve discussed how we obtain principal components (PCs) via an eigendecomposition
of the covariance matrix. The function `prcomp` uses a different but related technique that is more stable numerically. Try the following:

```r
e Fisher1 <- prcomp(iris[, 1:4])
```

The above code ignores the final column 5 of the data set, which is a categorical variable and should be excluded from a PCA. The alternative code `fisher1 <- prcomp(iris[, -5])` produces the same result.

Let’s inspect the output of the PCA:

```r
print(fisher1)
```

```
## Standard deviations (1, .., p=4):
## [1] 2.0562689 0.4926162 0.2796596 0.1543862
##
## Rotation (n x k) = (4 x 4):
## PC1   PC2   PC3   PC4
## Sepal.Length 0.36138659 -0.65658877 0.58202985 0.3154872
## Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
## Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
## Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

The output **Standard deviations** refers to the square root of the eigenvalues of the covariance matrix. **Rotation** are the eigenvectors of the covariance matrix, i.e., the PCs.

**Exercise:**
- Check that the output from `fisher1` agrees with what you find by applying the `eigen` function to the covariance matrix of the Iris data. See Lab 2 if you need any help subsetting the dataset, or on using the `eigen` and `cov` functions.

**Interpreting the Output of a PCA**

As already mentioned, using `prcomp` provides more straightforward access to the relevant results of a PCA. For example, try:

```r
summary(fisher1)
```

```
## Importance of components:
## PC1   PC2   PC3   PC4
## Standard deviation 2.0563 0.49262 0.2797 0.15439
## Proportion of Variance 0.9246 0.05307 0.0171 0.00521
## Cumulative Proportion 0.9246 0.97769 0.9948 1.00000

round(fisher1$rotation, 2)
```

```
## Sepal.Length 0.36 -0.66 0.58 0.32
## Sepal.Width  -0.08 -0.73 -0.60 -0.32
## Petal.Length  0.86  0.17 -0.08 -0.48
## Petal.Width   0.36  0.08 -0.55  0.75
```

The **summary** function provides some helpful statistics about the `prcomp` analysis, specifically related to the eigenvalues obtained in the PCA. `fisher1$rotation` and `fisher1$sdev` respectively call the rotations/PCs and standard deviations/square root of eigenvalues from `fisher1`. The `round` function rounds its first argument to the number of decimal places given by its second argument.
The proportion of the variance explained by each PC helps us to choose the relevant number of PCs to include in a lower dimensional summary of the data. An additional visual tool is the Scree plot. The default Scree plot in R plots the (decreasing) variance of each component – rather than the proportion of total variance explained – as either a bar chart (the default) or as a line chart.

```
plot(fisher1, main = "Fisher's Iris Data")
```

![Fisher's Iris Data](image)

```
plot(fisher1, main = "Fisher's Iris Data", type = "l")
```
In the second command, the argument type="l" results in a line plot, rather than the default bar plot. Note that in this case the `plot` command produced the Scree plot automatically. This is because R knows that `fisher1` is the output of the `prcomp` function. (More formally, its class is "prcomp". This is called an S3 generic method in R.) If we wished to produce the same plot just given a vector of PC standard deviations then we could use the `screeplot` function, or we can construct something ourselves using the default `plot` function. Can you make sense of the following code?

```r
fisher_var_explain <- (fisher1$sdev^2) / (sum(fisher1$sdev^2))
plot(fisher_var_explain, type = "b", main = "Fisher's Iris Data",
     xlab = "No. of components", ylab = "Proportion of variance explained", xaxt = "n")
axis(1, at = 1:4)
```
Fisher's Iris Data

![Graph showing proportion of variance explained vs. number of components.]

**Exercise:**
- What is the appropriate number of PCs for summarizing the Iris data?
- What is an appropriate description for each of these PCs?

**Projecting the data onto the principal components**

Recall that each new PC is just a linear combination of the original data. To see how the values of each observation map on to the PCs, use the `predict` function:

```r
newiris <- predict(fisher1)
head(newiris, n = 5)
```

```
## PC1  PC2  PC3  PC4
## [1,] -2.684126 -0.3193972 0.02791483 0.002262437
## [2,] -2.714142  0.1770012 0.21046427 0.099026550
## [3,] -2.888991  0.1449494 -0.01790026 0.019968390
## [4,] -2.745343  0.3182990 -0.03155937 -0.075575817
## [5,] -2.728717 -0.3267545 -0.09007924 -0.061258593
```

The `predict` function is another generic function (like `print`, `summary`, and `plot`) that predicts the results of a model based on inputted data and a previously fitted model object. In this case, `predict` again recognizes that `fisher1` is the output of the `prcomp` function (i.e., its class is "prcomp") and calculates the value of each observation based on the estimated PC loadings.

**Exercise**
- Find the PC values for observation 10 using either `predict` or `newdata`.

---

5
• Check this result by calculating (in R!) PC values directly using the \texttt{fisher1$rotation} loadings matrix. Note that \texttt{prcomp} centers the data by default before performing the PCA. This means that you will have to subtract the mean of the data from the observation for each variable before multiplying it by the PC coefficients in order to get the same result.

### Visualising results

To visualize multivariate data, one possibility is to produce a scatter plot of one variable against another. For example, using:

```r
plot(iris[, 1], iris[, 2], col=iris[, 5])
legend(6.5, 4.5, legend = levels(iris[, 5]), col = c(1, 2, 3), pch = 1)
```

The first command produces a scatter plot of the first variable in the \texttt{iris} data set against the second variable, and colours the points according to their species type (the fifth variable). The second command adds a legend to the plot, where the first two arguments specify the \textit{x} and \textit{y} co-ordinates of the legend (usually some trial and error is needed here). The third argument provides the phrases to be used in the legend. The fourth and fifth arguments specify the colour and shape of the points. Many more arguments are available. For further information look at the help file.

Both \texttt{iris[,1:4]} and \texttt{newiris} are data sets with four dimensions of variables (the latter being the data set of PC projections for the observations). It would be nice to try and visualise all (or at least several) of the pair-wise scatter plots of these variables at once. A quick way to do this is to use the \texttt{pairs} function:

```r
pairs(iris[,1:4],col=iris[,5])
```
pairs(newiris, col = iris[, 5])
From the second plot you should be able to see that the values of the leading principal component PC1 provide a good separation of the observations by species.

If we only focus on the values of PC1 and PC2 we could do something a little more fancy. Look at the appropriate help files to understand how the following works:

```r
plot(newiris[,1], newiris[,2], type="n", xlab="PC1", ylab="PC2")
text(newiris[,1], newiris[,2], labels=substr(iris[,5],1,2), col=as.integer(iris[,5]))
```
The first line produces an empty plot (because of the argument `type="n"`), whilst the second states where points should be added (the first two arguments), using what symbols (third argument), and by what colouring (final argument).

**PCA using correlation**

In lectures we discussed how PCA is not invariant to the scaling of the variables and that it is often preferable to standardize the data so that each variable has unit variance. This is equivalent to performing an eigendecomposition of the correlation matrix. To do this using the `eigen` function, you can effectively repeat your earlier analysis, substituting the `cor` function with the `cov` function:

```r
eigen(cor(iris[, 1:4]))
```

```r
## eigen() decomposition
## $values
## [1] 2.91849782 0.91403047 0.14675688 0.02071484
## $vectors
## [1,] 0.5210659 -0.37741762 0.7195664 0.2612863
## [2,] -0.2693474 -0.92329566 -0.2443818 -0.1235096
## [3,] 0.5804131 -0.02449161 -0.1421264 -0.8014492
## [4,] 0.5648565 -0.06694199 -0.6342727 0.5235971
```

Alternatively, using the `prcomp` command, we set the additional argument `scale=TRUE`:

```r
prcomp(iris[, 1:4], scale=TRUE)
```
## Standard deviations (1, .., p=4):
## [1] 1.7083611 0.9560494 0.3830886 0.1439265

## Rotation (n x k) = (4 x 4):
## PC1    PC2    PC3    PC4
## Sepal.Length  0.5210659 -0.37741762 0.7195664 0.2612863
## Sepal.Width   -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727 0.5235971

Do the two analyses using `eigen` and `prcomp` still agree?

### Exercise:
- Re-run the Iris data PCA with scaled variables. Do your previous conclusions still hold?

We can check what the original variable variances where by reading the diagonal terms from the result of `cov(iris[,1:4])`, e.g., using the `diag` function. Another approach is to check the variances individually using `var(iris[,1])`, `var(iris[,2])`, etc. (Another approach is to use the `apply` function, but we won’t discuss this here.)

```r
diag(cov(iris[, 1:4]))
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width
## 0.6856935 0.1899794 3.1162779 0.5810063
```

```r
c(var(iris[, 1]), var(iris[,2]), var(iris[, 3]), var(iris[, 4]))
```

```
## [1] 0.6856935 0.1899794 3.1162779 0.5810063
```

In this case the variances are are not so dissimilar (i.e., all within the same order of magnitude) and we would not expect to observe a dramatic difference in results from a PCA on the original data or the standardized data. In this case, which analysis is preferable, in your opinion?

### Olive Oil Data

Go to [https://www.scss.tcd.ie/~arwhite/Teaching/STU33011.html](https://www.scss.tcd.ie/~arwhite/Teaching/STU33011.html) and save the Olive Oil data file ([https://www.scss.tcd.ie/~arwhite/Teaching/STU33011/olive.csv](https://www.scss.tcd.ie/~arwhite/Teaching/STU33011/olive.csv)) in an appropriate location. Read the data into R. Note that the first two variables are categorical, and you should therefore restrict attention to the last 8 variables when performing a PCA. You can do this by e.g., renaming or working directly with `olive[,3:10].`

### Exercise:
- Perform a PCA on both the original Olive Oil data and the standardized Olive Oil data. What are the original variances of the variables?
- Do you notice any important differences between the analyses? What are these? Which analysis is the more reliable?