Principal Component Analysis

Sam Bowyer

2023-01-27

Task 1

First we shall load the `USArrests` dataset and extract the three features of interest.

```
data(USArrests)
X0 = USArrests[, c("Murder", "Assault", "Rape")]
head(X0)
```

```
##            Murder Assault Rape
## Alabama      13.2     236 21.2
## Alaska       10.0     263 44.5
## Arizona       8.1     294 31.0
## Arkansas      8.8     190 19.5
## California    9.0     276 40.6
## Colorado      7.9     204 38.7
```

Next we must center the data.

```
X0 = as.matrix(X0)
n = nrow(X0)
p = ncol(X0)
C = diag(n) - matrix(rep(1/n,n*n), n)
X = C %*% X0
```

We can quickly check that indeed each feature now has zero mean (or at least very close to zero because of unavoidable floating point errors).

```
mean(X[,"Murder"]); mean(X[,"Assault"]); mean(X[,"Rape"])
```

```
## [1] 1.213438e-15
```

```
## [1] -2.685581e-14
```

```
## [1] -1.674771e-15
```

Now we can find the principal components by calculating the covariance matrix and performing a spectral decomposition to obtain the eigenvalues and eigenvectors.

```
# Covariance matrix
S = (1/n) * (t(X) %*% X)

# Spectral decomposition
decomp = eigen(S)
decomp
```

```
## eigen() decomposition
## $values
## [1] 6856.551123   47.685467    6.591443
```

```
##
## $vectors
##             [,1]        [,2]        [,3]
## [1,] -0.04180743  0.02555358  0.99879886
## [2,] -0.99630506 -0.07612980 -0.03975532
## [3,] -0.07502247  0.99677042 -0.02864195
```

The column vectors here show us the loadings of each PC (if we take the absolute value of each entry). In particular we see that the first principal component mainly takes into account the value of the assault in each state, with slightly more importance on rape than murder, whereas the second PC mainly focuses on rape and the third on murder, with close (but small) loadings on assault and rape.

```
A = decomp$vectors
lambda = decomp$values

Y = X %*% A
rownames(Y) <- rownames(USArrests)
```

The principal components are given by the columns of `Y` (in order from left to right).

```
head(Y)
```

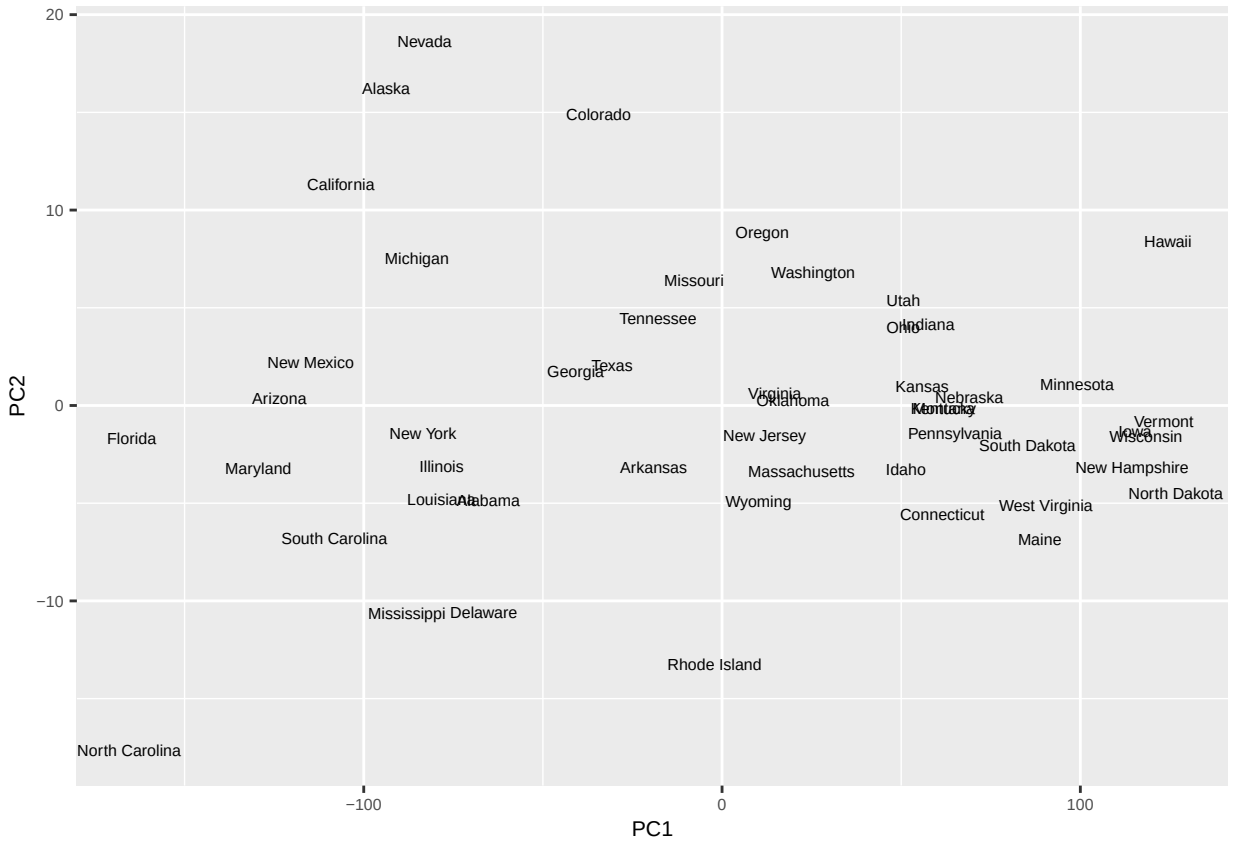
```
##                   [,1]       [,2]      [,3]
## Alabama     -65.22280 -4.8603090  2.812779
## Alaska      -93.73728 16.2271656 -2.124128
## Arizona    -123.53050  0.3621893 -4.867595
## Arkansas    -19.08128 -3.1652836  0.295500
## California -106.35485 11.3245200 -3.528043
## Colorado    -34.43236 14.8838930 -1.709919
```

To plot the two-dimensional reduction of the dataset we can simply use the first two columns of Y.

```
library(ggfortify)
```

```
## Loading required package: ggplot2
```

```
autoplot(Y[,1:2], geom="point", label=T, label.size=2, shape=F, xlab="PC1", ylab="PC2") +
  theme(text=element_text(size=8))
```



However, noting that the three variables are on different scales, i.e. mostly on different orders of magnitude, we will instead perform and analyse PCA with the correlation matrix `R` rather than the covariance matrix. We do this with the `prcomp` command in `R` with the parameter `scale=TRUE`, which scales the data to have unit variance, therefore allowing us to use the correlation matrix, computing the spectral decomposition as $R = B\Lambda B^T$ where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ gives us the eigenvalues of R and where the columns of B give us the eigenvectors of R.

```
pca = prcomp(X, scale=TRUE, retx=TRUE)
```

This returns three objects: `pca$sdev`, `pca$rotation` and `pca$x` (the last of these only returned because we called `prcomp` with the parameter `retx=TRUE`). The first of these gives us the standard deviations of the eigenvalues of the correlation matrix (note that we will then need to square them later when analysing variances), the second gives us the loadings of each principal component such that each column is a PC and the final object `pca$x` gives us the rotated data, Y, which we may truncate by only plotting the first two dimensions.

```
# Square root of correlation matrix eigenvalues
pca$sdev
```

```
## [1] 1.5357670 0.6767949 0.4282154
```

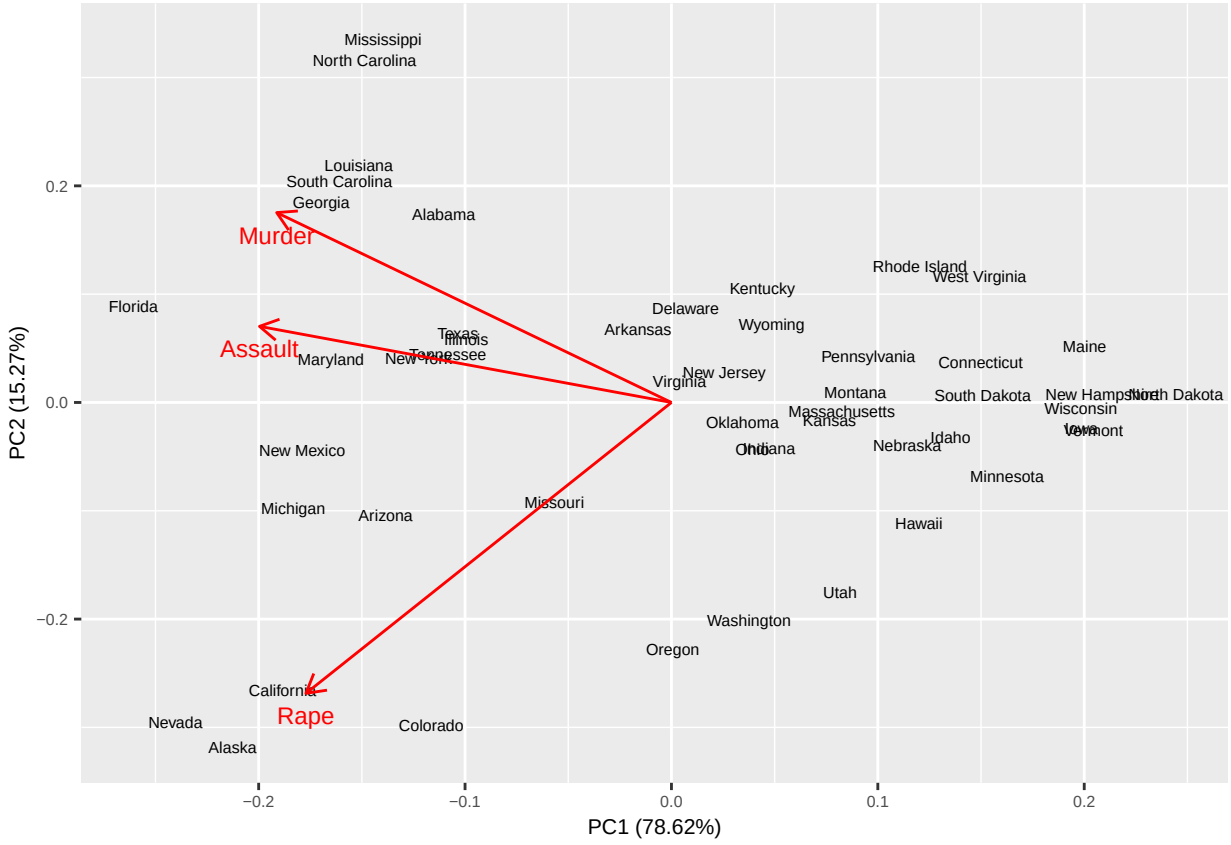
```
# Principal components
pca$rotation
```

```
##                PC1        PC2        PC3
## Murder  -0.5826006  0.5339532 -0.6127565
## Assault -0.6079818  0.2140236  0.7645600
## Rape    -0.5393836 -0.8179779 -0.1999436
```

We can see that using the correlation matrix rather than the covariance matrix for PCA has greatly affected

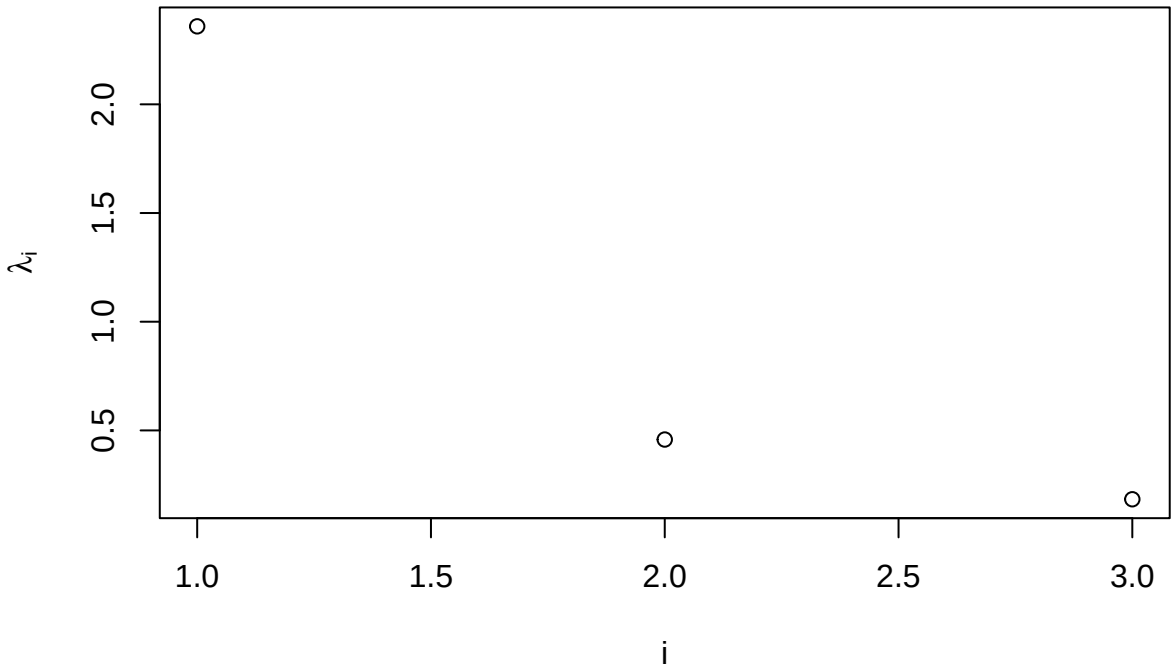
the loadings of each of our principal components. Whereas before each PC was dominated by one of the three variables, the loadings are much more evenly spread now. In particular, the first principal component has very similar loadings for each of the variables, indicating that higher arrests in one of the categories suggest higher arrests in the other two (with especially similar loadings for murder and assault). The second PC then has a much heavier loading on the rape variable and a slightly less heavy loading on assault, with very little coming from assault, whilst the third PC has a loading dominated by assault and murder with a small loading on rape.

```
# Plot the rotated data with the projections of the variables
rownames(pca$x) <- rownames(USArrests)
autoplot(pca, label=T, label.size=2, loadings=T, loadings.label=T,
         loadings.label.size=3, loadings.label.vjust = 1.8, shape=F) +
  theme(text=element_text(size=8))
```



To decide on the number of dimensions q to keep we will first make a scree plot:

```
library(latex2exp)
plot(1:3, pca$sdev^2, ylab=TeX(r'($\lambda_i$)'), xlab="i", main="Scree Plot for the USArrests Dataset")
```



This shows that the first principal component accounts for the vast majority of the variance in the dataset, suggesting that we should reduce the dimensionality to $q = 1$, however, we shall confirm this intuition by calculating q_k (via Kaiser's criterion) and $q_H^{(M)}$ (via Horn's parallel analysis) as well.

```
# Kaiser's criterion
meanLambda = mean(pca$sdev^2)
q_k = 0
for (j in 1:3){
  if (pca$sdev[j]^2 > meanLambda){
    q_k = j
  } else{
    break # since lambda_{i+1} <= lambda_i
  }
}
q_k
```

```
## [1] 1
```

```
# Horn's parallel analysis
S = cor(X)
D = diag(S)
D_sqrt = diag(sqrt(D))
M = 10000
eigenvalues = matrix(rep(0,M*p), nrow=M)
for (m in 1:M){
  X_m = matrix(rnorm(n*p), nrow=n, ncol=p)
  R_m = cor(X_m)
  S_m = D_sqrt %*% R_m %*% D_sqrt
  eigenvalues[m,] = eigen(S_m)$values
}
meanLambdas = apply(eigenvalues, 2, mean)
```

```
q_H = 0
for (j in 1:3){
  if (pca$sdev[j]^2 > meanLambdas[j]){
    q_H = j
  }
}
q_H
```

```
## [1] 1
```

Since we've obtained (with $M = 10000$) that $q_k = q_H^{(M)} = 1$, which is consistent with our intuition from the scree plot, this confirms our belief that we should reduce the dimensionality of the dataset to $q = 1$ by considering only the first principal component, which explains 78.6% of the variance in the dataset (even though adding the second principal component would take that figure up to 93.9%).

```
eigenvalues = pca$sdev^2

# Variance explained by PC1
eigenvalues[1]/sum(eigenvalues)
```

```
## [1] 0.7861934
```

```
# Variance explained by PC1 & PC2
(eigenvalues[1]+eigenvalues[2])/sum(eigenvalues)
```

```
## [1] 0.9388772
```

Task 2

For this task we'll perform PCA on the `IRIS` dataset of flowers using the covariance matrix rather than the correlation matrix since the variables are largely on the same scale already.

```
data(iris)
head(iris)
```

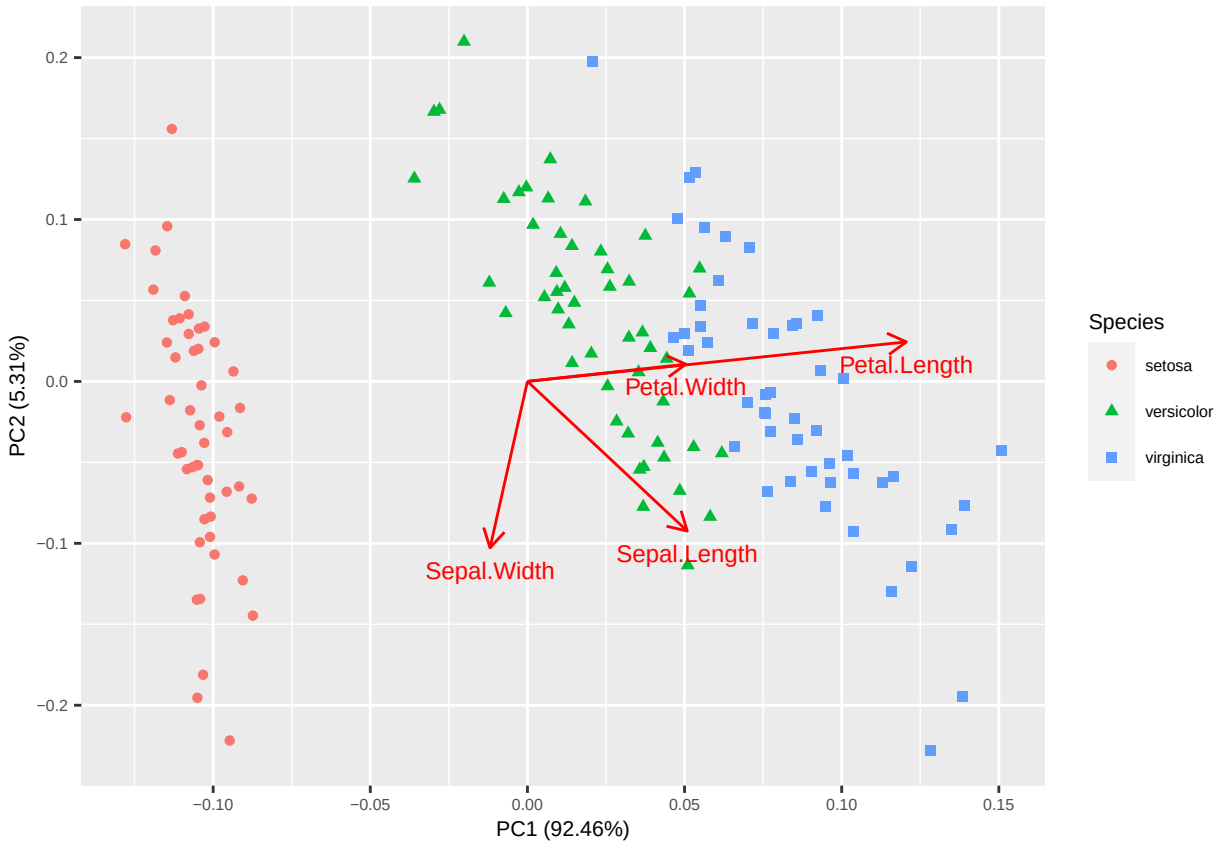
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

We'll perform PCA using the `prcomp` function again with `scale=FALSE` (as it is by default) to use the covariance matrix. (Note also that `prcomp` centers the data for us with the parameter `center=TRUE`.)

```
# Remove the species from the data when performing PCA
X = iris[,1:4]

pca = prcomp(X, center=TRUE, retx=TRUE)

autoplot(pca, data=iris, colour="Species", loadings=T, loadings.label=T,
         loadings.label.size=3, loadings.label.vjust = 1.8, shape="Species")  +
  theme(text=element_text(size=8))
```



The PCA has successfully separated the data points by their species: setosa being the most distinct group and with some slight overlap between the versicolor and virginica clusters. Usefully, by plotting this 2D projection we see that the petal width and petal length are highly related as their projections follow the same direction from the origin, but with petal length larger in magnitude than petal width. In particular, petal width and petal length have relatively large loadings in the first principal component but small loadings in the second principal component, whereas sepal width and sepal length both have larger loadings in PC2 than PC1. This make sense since the two pairs of variables represent measurements on different parts of the plants.

```
pca$rotation
```

```
##                     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
```

It is also important to note here that the first prinicipal component accounts for almost all (92.46%) of the total variance in the dataset, meaning that it would most likely be sensible to choose $q = 1$ as the dimensionality of a PCA reduction performed on this data.

Task 3

For this task, we'll perform principal component regression on the `Communities and Crime` dataset (from the `R` package `mogavs`) using the package `pls`. This dataset contains 122 predictive attributes and one target attribute—representing the number of violent crimes in an area per population—with 1994 samples each. We'll split the data into a training set of size 1595 and a test set of size 399 for a roughly 80/20 training/test split.

```
library(mogavs)
data(crimeData)
```

```
dim(crimeData)
```

```
## [1] 1994  123
```

```
trainIdx = sample(nrow(crimeData), 1595)
train = crimeData[trainIdx,]
test  = crimeData[-trainIdx,]

x_test = test[,1:122]
y_test = test[,123]
```

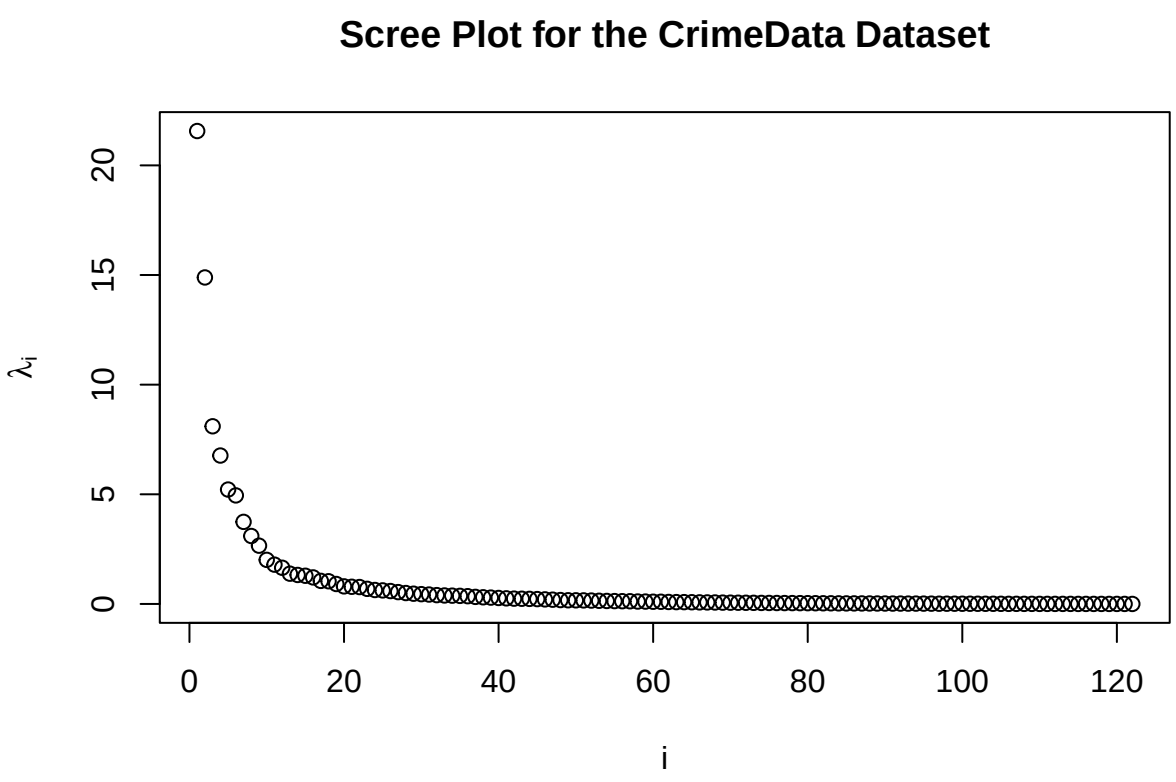
We'll fit a linear regression model using cross-validation with 10 random splits, as well as using the correlation matrix rather than the covariance matrix for the PCA (by setting `scale=TRUE`) since some of the input attributes are on different scales (for example there contains information on the age of residents in an area and also information on rent prices). We can then use the `explvar` command to make a scree plot of the PCA reduction.

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
model = pcr(y~., data=train, scale=TRUE, validation="CV")
plot(1:122, explvar(model), ylab=TeX(r'($\lambda_i$)'), xlab="i", main="Scree Plot for the CrimeData Da
```



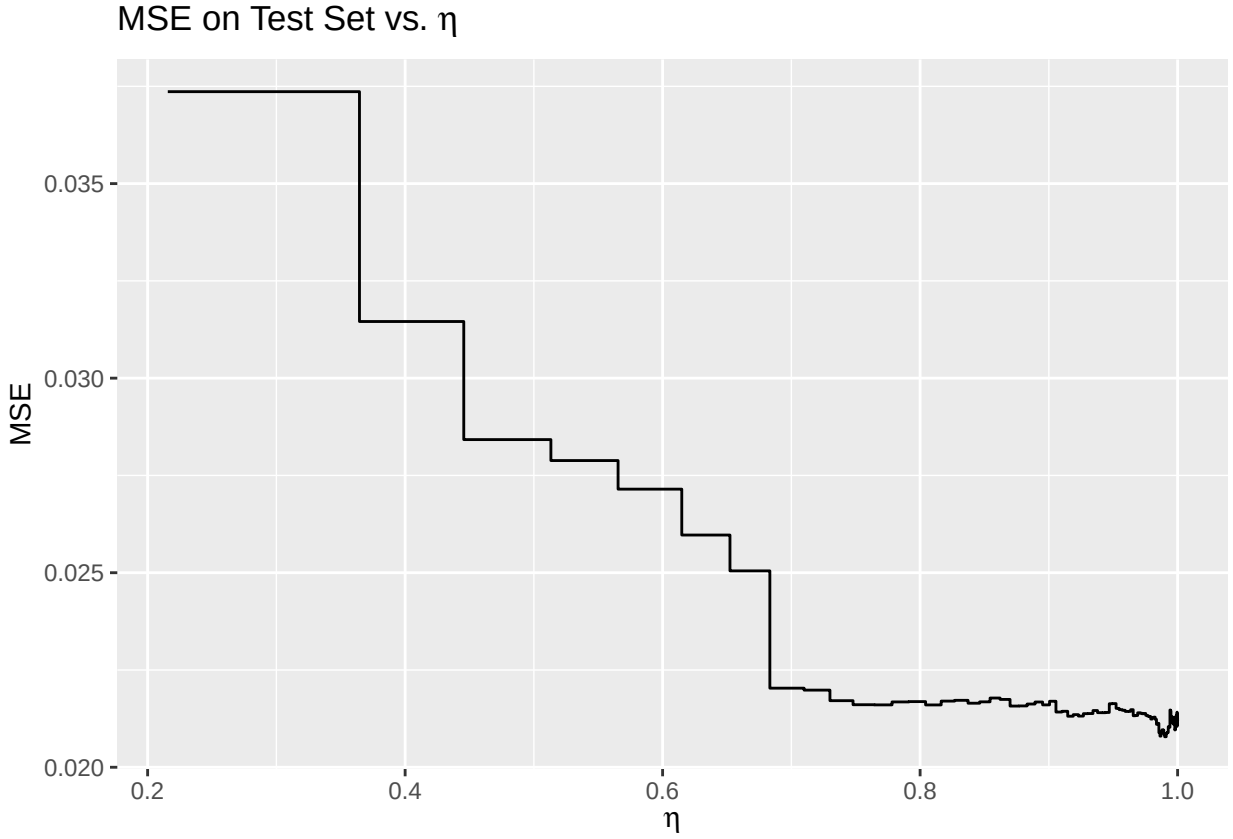
This indicates that we can probably pick a reduction of dimension $1 \leq q \leq 20$ and still represent a decent

proportion of the variance in the dataset, however, we'll analyse the mean squared error (MSE) of test set predictions using various values of q_η for $\eta \in (0,1)$ where (with $p = 122$):

$$q_\eta = \min\left(j : \sum_{k=1}^{j} \lambda_k / \sum_{k=1}^{p} > \eta\right).$$

```
errs = rep(0,122)
for (k in 1:122){
  y_pred = predict(model, x_test, ncomp=k)
  errs[k] = mean((y_pred - y_test)^2)
}

etas = cumsum(explvar(model))/100
df = data.frame(x=etas, y=errs)
ggplot(df, aes(x=x, y=y)) + ylab("MSE") + xlab(TeX(r'($\eta$)')) +
  ggtitle(TeX(r'(MSE on Test Set vs. $\eta$)')) + geom_step()
```



In the above plot, each vertical plot represents the inclusion of the next principal component. We see that each of the first eight principal components give us a significant reduction in error, with an MSE of 0.022 for $q_\eta = 8$ (which is the case for approximately $0.683 \leq \eta < 0.710$) compared to 0.025 for $q_\eta = 7$.

```
# Find MSE for q=7 and q=8
errs[7]; errs[8];
```

```
## [1] 0.02504727
```

```
## [1] 0.02203309
```

```r
# Find smallest eta such that q=8 and q=9
etas[8]; etas[9]
```

```
##    Comp 8
## 0.6833036
```

```
##    Comp 9
## 0.7098749
```

However, after the eighth principal component, adding more to the reduction doesn't lead to a significant improvement in the error until we approach $\eta = 1$ at which point we start seeing an increase in error when adding certain PCs that lead to overfitting. Therefore in order to minimise the predictive error whilst sufficiently reducing the dimensionality of the data we should choose $q = 8$, which still allows us to express 68.3% of the total variance in the dataset (adding the 9th PC would only take us up to 71.0%).

```r
# Total variance explained by first 8 PCs (equivalent to etas[8])
sum(explvar(model)[1:8])
```

```
## [1] 68.33036
```