Statistical Methods 2: Porfolio 1

Kieran Morris

1 USArrests DataSet

The USArrests dataset consists of four columns `Murder`, `Assault`, `UrbanPop` and `Rape`, with 50 columns describing the 50 states in the data. We intend to perform principle component analysis.

1.1 Importing and Processing data

Below we import the datset `USArrests` and remove the `UrbanPop` column as it is not necessary. Then we create two versions of our data, `USArrests` which is centered and is equivilent to X in the notes, and `USArrests_SCALED` which is both centered and scaled, and is equivilent to Z in the notes.

```
data("USArrests")
#Import dataset USArrests (X^0)
USArrests$UrbanPop <- NULL
#Remove UrbanPop column as not needed
USArrests <- as.matrix(USArrests)
#Convert to matrix
USArrests <- scale(USArrests,center = TRUE,scale = FALSE)
#Center the data (get X)
USArrests_SCALED <-scale(USArrests,center = TRUE,scale = TRUE)
#Scale the data (get Z)
```

1.2 Computing the Correlation and Covariance Matrices

Now we can manually compute our correlation matrix R, i.e `USArr_Cor` and our covariance matrix S, i.e `USArr_Cov`. We can then use the `eigen()` function to calculate the eigenvalues and eigenvectors of both R and S. We can then extract the eigenvectors from the `eigen()` function and use them to calculate the principal components of the data, i.e `PCS`. Note that the function `prcomp` will do this automatically, however for educational purposes we perform this computation manually.

```
n = nrow(USArrests)
USArr_Cov <- t(USArrests)%*%USArrests/(n)
USArr_Cor <- t(USArrests_SCALED)%*%USArrests_SCALED/(n)
#Calculate the covariance and correlation matrices
```

1.3 Performing PCA

Now we have our relevant matrices, we can manually perform PCA by eigenvalue decomposition. Again `prcomp` will do this automatically.

```r
USArr_Cov_eigen <- eigen(USArr_Cov)
USArr_Cor_eigen <- eigen(USArr_Cor)
#Calculate the eigenvalues and eigenvectors

USArr_Cov_eVec <- USArr_Cov_eigen$vectors
USArr_Cor_eVec <- USArr_Cor_eigen$vectors
#Extract the eigenvectors from the eigen() function

PCS <- USArrests%*%USArr_Cov_eVec
PCS_Scaled <- USArrests_SCALED%*%USArr_Cor_eVec
#Calculate the principal components
colnames(PCS) <- c("PC1","PC2","PC3")
colnames(PCS_Scaled) <- c("PC1","PC2","PC3")
```

```r
head(PCS)
```

```
##                   PC1        PC2       PC3
## 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
```

```r
head(PCS_Scaled)
```

```
##                  PC1        PC2         PC3
## Alabama    1.1980278 -0.8338118  0.16217848
## Alaska     2.3087473  1.5239622 -0.03833574
## Arizona    1.5033307  0.4983038 -0.87822311
## Arkansas   0.1759894 -0.3247326 -0.07111174
## California 2.0452358  1.2725770 -0.38153933
## Colorado   1.2634133  1.4264063  0.08369314
```

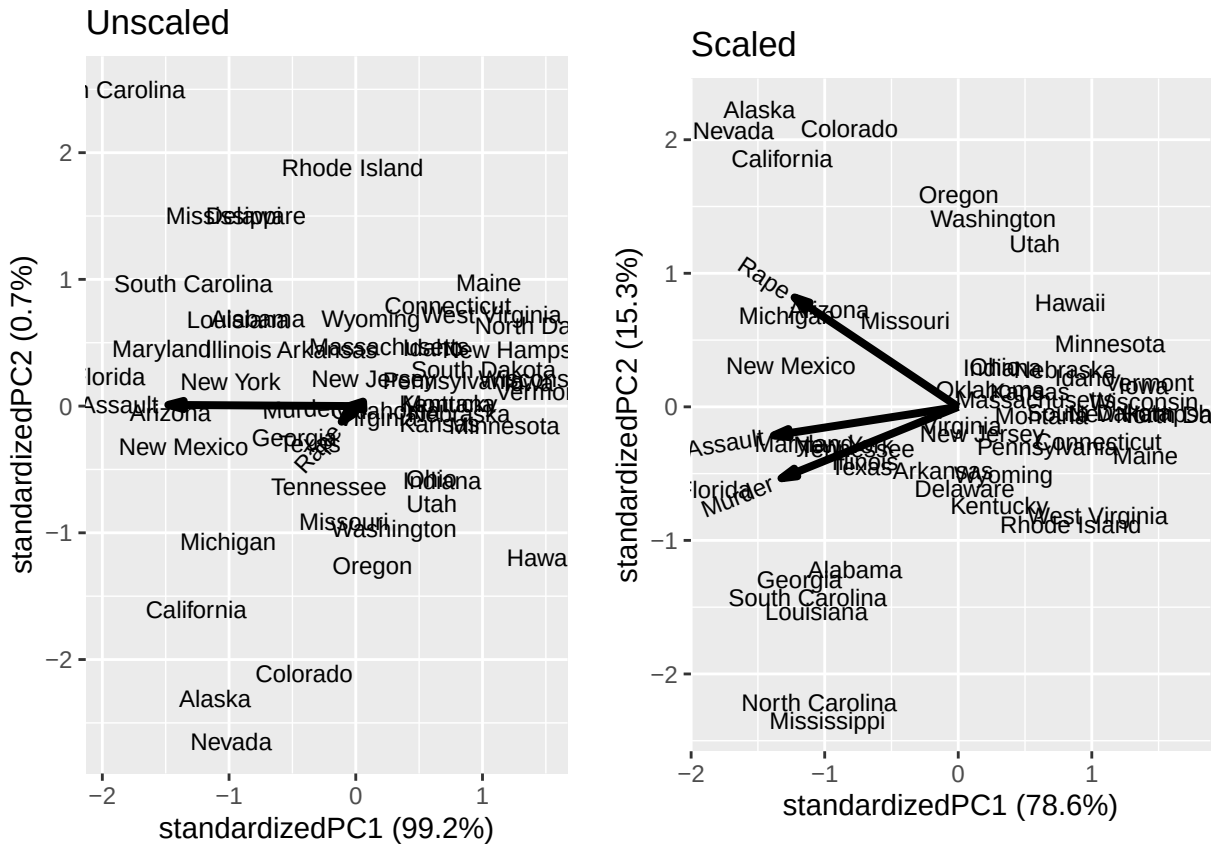
We have demonstrated how to perform PCA manually, and theoretically could additionally plot the data on a `ggplot` with the projections of the variables `Murder`, `Assault` and `Rape`, however time is finite and `prcomp` is so easy to use.

```r
# Perform PCA using prcomp
PCS <- prcomp(USArrests, center = TRUE, scale = FALSE)
PCS_Scaled <- prcomp(USArrests, center = TRUE, scale = TRUE)
PCS_Scaled
```

```
## Standard deviations (1, .., p=3):
## [1] 1.5357670 0.6767949 0.4282154
##
## Rotation (n x k) = (3 x 3):
##                PC1        PC2        PC3
## Murder  -0.5826006 -0.5339532  0.6127565
## Assault -0.6079818 -0.2140236 -0.7645600
## Rape    -0.5393836  0.8179779  0.1999436
```

On the topic of the `prcomp` object, let's analyse it's structure. It is a list with 5 elements, `sdev`, `rotation`, `center`, `scale` and `x`. The first element `sdev` is the standard deviation of the principal components, i.e the square root of the eigenvalues of the covariance matrix. The second element `rotation` is the matrix of eigenvectors (principal components) of the covariance matrix. The third element `center` is the mean of the variables used to center the data. The fourth element `scale` is the standard deviation of the variables used to scale the data. Finally the fifth element `x` is the matrix formed from multiplying the original data by the principal components.

```
# Create the plots
pl1 <- ggbiplot(PCS, labels = rownames(USArrests),ellipse = TRUE)+
ggtitle("Unscaled")
pl2 <- ggbiplot(PCS_Scaled, labels = rownames(USArrests), ellipse = TRUE)+
ggtitle("Scaled")

# Arrange the plots side by side
combined_plot <- gridExtra::arrangeGrob(pl1, pl2, ncol = 2)

grid::grid.draw(combined_plot)
```



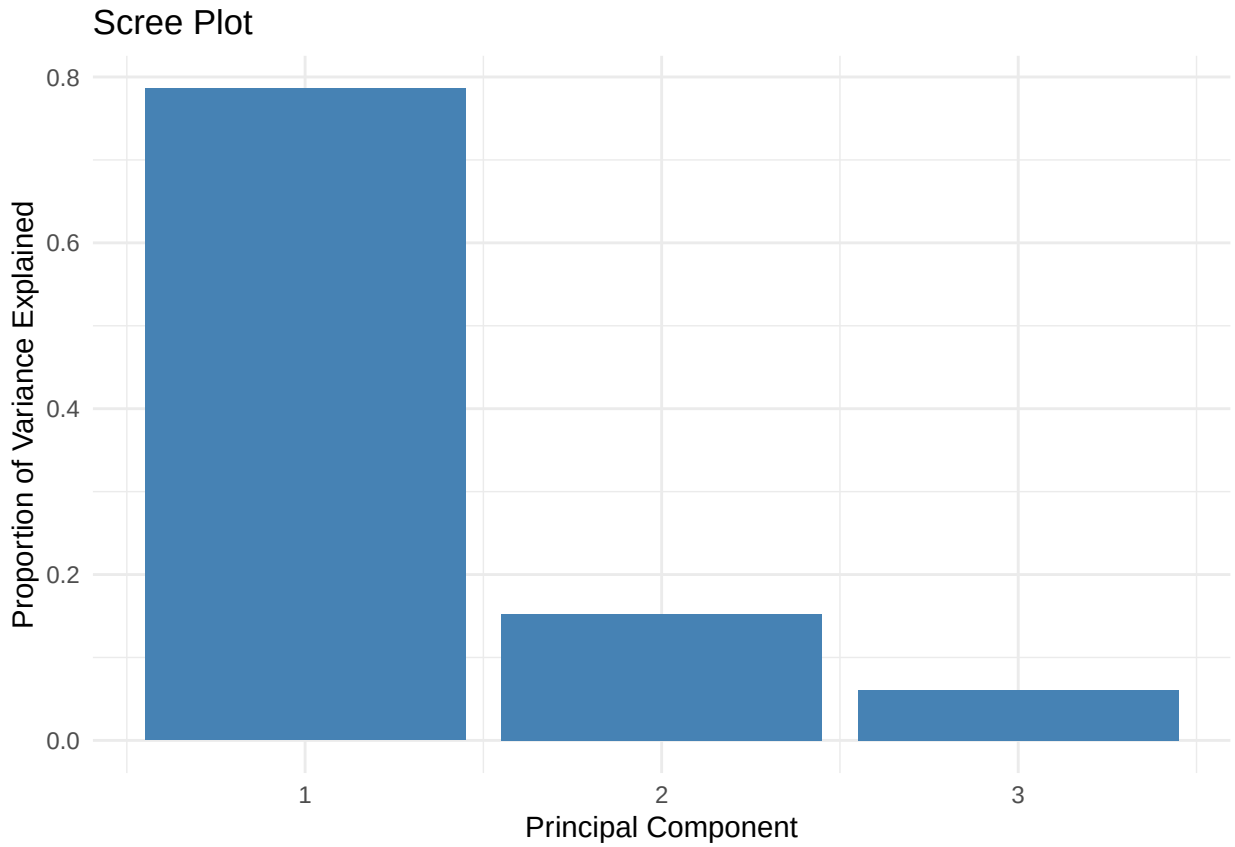

```
# Display the plots
```

We can see from these plots the more informative one is the scaled version, as the variables are all on the same scale. In opposition to the unscaled version where `Assault` dominates in the PC1 direction. From now on we will only consider the scaled (correlation) version.

1.4 Skree plot

A screeplot is the plot of principal components against the variance they contribute to the data. Below we use `ggplot` to compute this.

```
# Extract the proportion of variance
variance <- PCS_Scaled$sdev^2
variance_ratio <- variance / sum(variance)

# Create a data frame
df <- data.frame(Dimension = 1:length(variance_ratio), Variance = variance_ratio)

# Create the bar chart
ggplot(df, aes(x = Dimension, y = Variance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  theme_minimal() +
  labs(x = "Principal Component", y = "Proportion of Variance Explained",
       title = "Scree Plot")
```



This clearly shows that PC1 contributes considerably more variance to the data than anything else. This makes us think that maybe it should be the main component that we consider, however to be more formal we will use Kaiser's criterion.

1.5 Choice of PCA

The quantity q_K is the index of the smallest eigenvalue larger than the mean of the eigenvalues. We can compute this as follows.

```
PC_eigen <- PCS_Scaled$sdev^2
#Take eigenvalues of PCS
q_K <- max(which(PC_eigen > mean(PC_eigen)))
# Compute q_K

q_K
```

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

So we decide to only keep PC1.

2 Iris DataSet

The Iris dataset contains 4 variables `Sepal.Length`, `Sepal.Width`, `Petal.Length` and `Petal.width`. It also has a species column, which we intend to classify by using PCA dimension reduction. Firstly we imported the data and processed it into matrix form.

```
data("iris")
#Import dataset iris

iris_data <- as.matrix(iris[,-5])
#Remove the species column
#Convert to matrix and scale
```

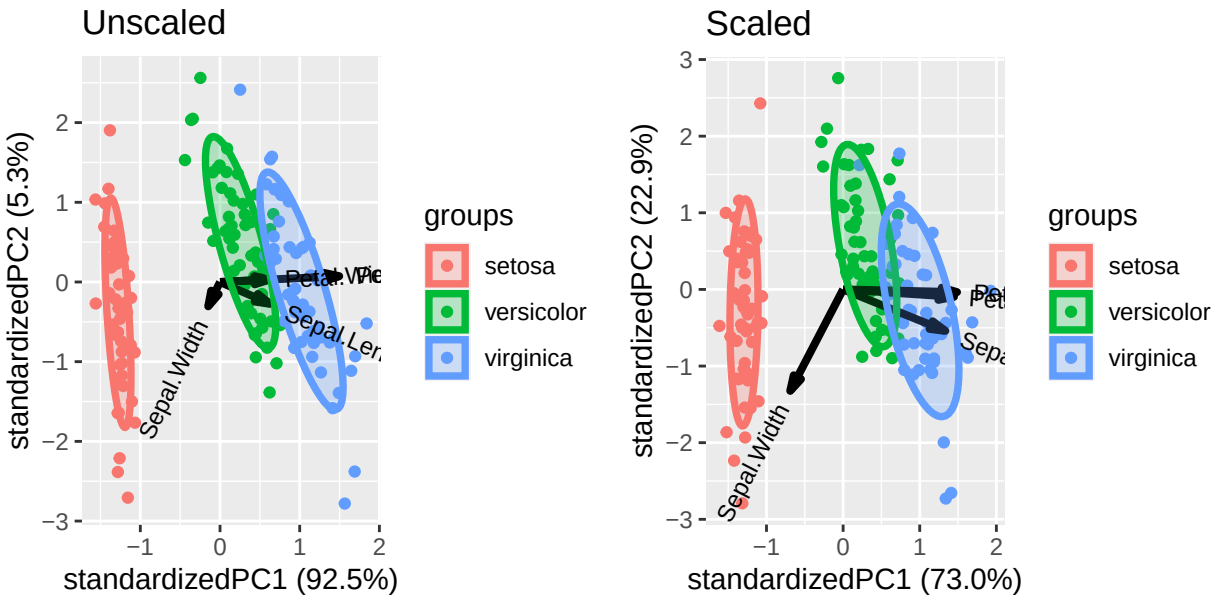
2.1 Performing PCA

```
# Perform PCA using prcomp
PCS <- prcomp(iris_data, center = TRUE, scale = FALSE)
PCS_Scaled <- prcomp(iris_data, center = TRUE, scale = TRUE)

PCS
```

```
## 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
```

2.2 Plotting PC 1&2

```
# Create the plots
pl1 <- ggbiplot(PCS, groups = iris$Species,ellipse = TRUE)+
  ggtitle("Unscaled")
pl2 <- ggbiplot(PCS_Scaled, groups = iris$Species, ellipse = TRUE)+
  ggtitle("Scaled")

# Arrange the plots side by side
combined_plot <- gridExtra::arrangeGrob(pl1, pl2, ncol = 2)

grid::grid.draw(combined_plot)
```



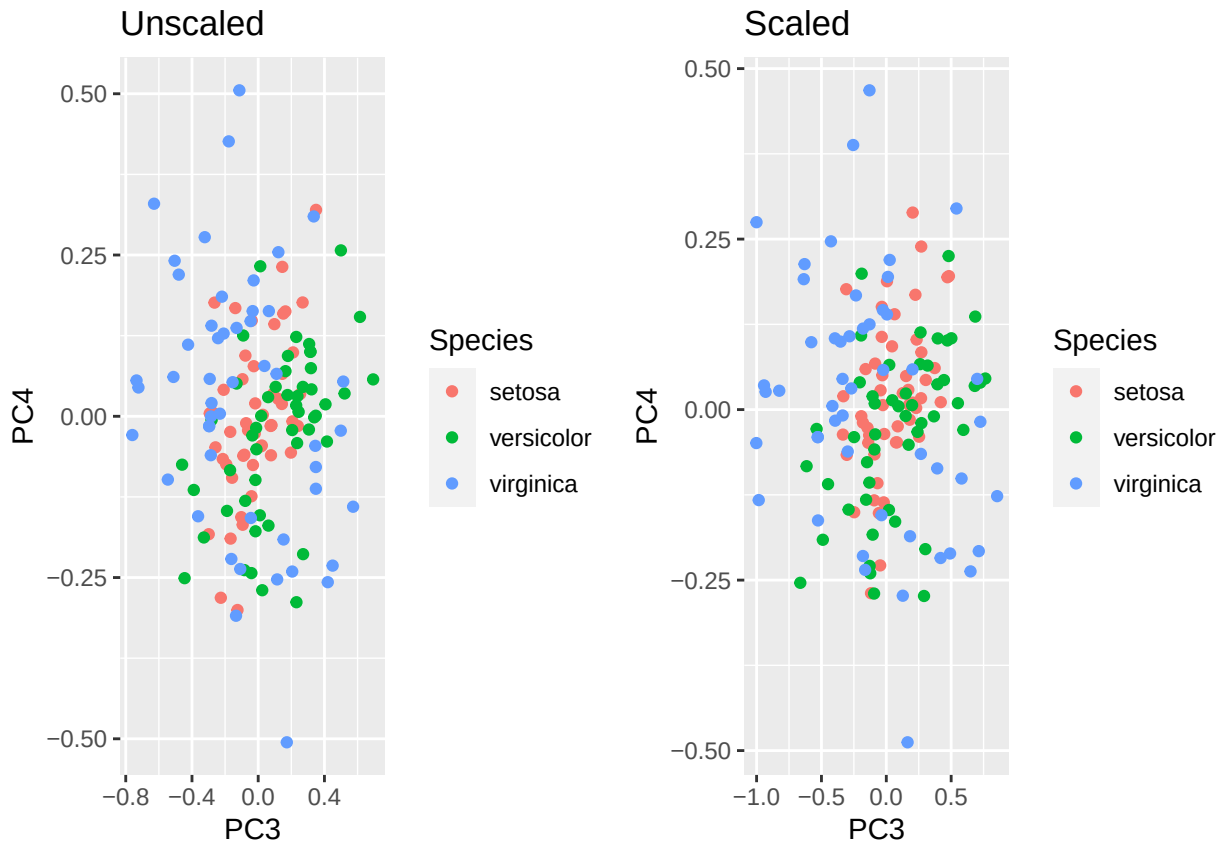
As we can see there is not much difference in the two plots in terms of classification, however the difference in the projection of the variables onto the principle components is clear. For example `Sepal.Width` is much larger when scaled.

2.2.1 Analysis

We can see from the plots that the `setosa` species is clearly seperated from the other two species when we project onto PC1 and PC2. Additionally the two remaining species are pretty well seperated too, a linear classification algorithm would be able to pick out these fairly easily. If we were to choose either Unscaled or Scaled we would choose Unscaled, as it gives a clearer seperation between `versicolor` and `virginica`. Although the difference is very little.

2.3 Plotting PC 3&4

```
# Create a data frame with the scores for PC3 and PC4
df <- data.frame(PC3 = PCS$x[,3], PC4 = PCS$x[,4], Species = iris$Species)

# Create the plot
pl3 <- ggplot(df, aes(x = PC3, y = PC4, color = Species)) +
  geom_point() +
  labs(x = "PC3", y = "PC4", color = "Species") +
  ggtitle("Unscaled")

# Create a data frame with the scores for PC3 and PC4 for the scaled data
df_scaled <- data.frame(PC3 = PCS_Scaled$x[,3], PC4 = PCS_Scaled$x[,4], Species = iris$Species)

# Create the plot for the scaled data
pl4 <- ggplot(df_scaled, aes(x = PC3, y = PC4, color = Species)) +
  geom_point() +
  labs(x = "PC3", y = "PC4", color = "Species") +
  ggtitle("Scaled")

# Arrange the plots side by side
combined_plot <- gridExtra::arrangeGrob(pl3, pl4, ncol = 2)

# Display the plots
grid::grid.draw(combined_plot)
```



2.3.1 Analysis

Clearly here it is total chaos, this backs up the theory that as we continue down the principle components, the variance explained by each component decreases.

3 Regression on Communities and Crime Dataset

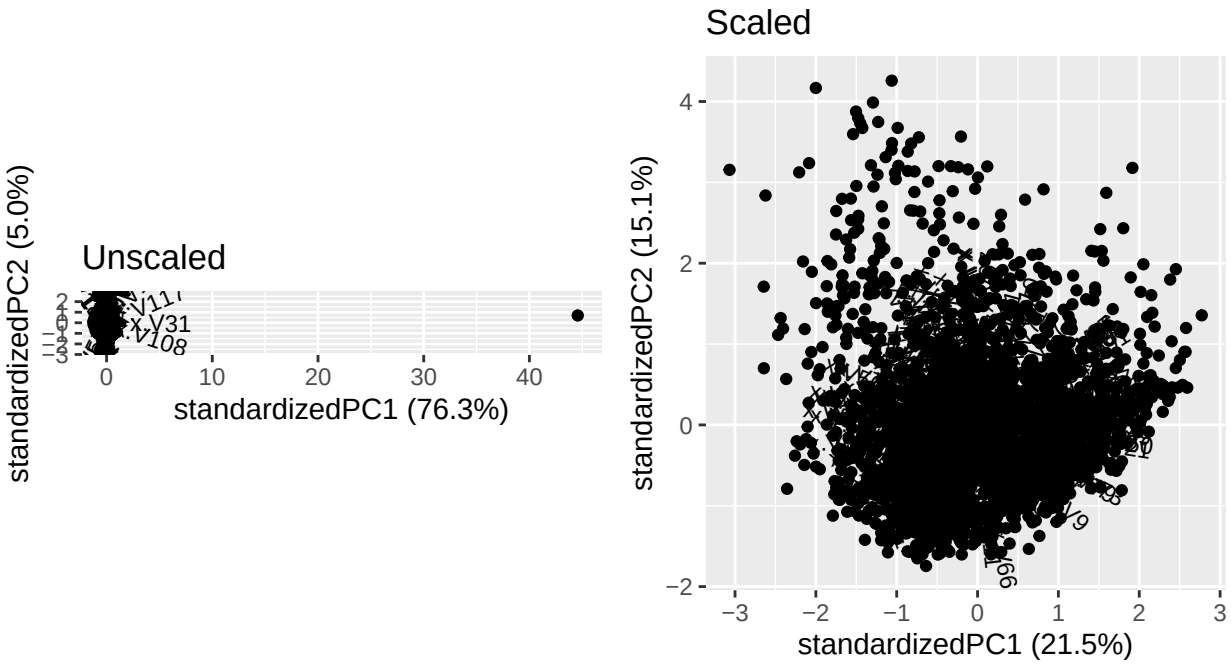
The crimeData dataset is contained in the `mogavs` package. It has 123 variables, including a classifier `y`. Making it an ideal candidate for PCA dimension reduction. As before, below we load and process the dataset into matrix form.

```
library(mogavs)
library(skimr)
data("crimeData")
#Import dataset crimeData
crimeDataCOPY <- crimeData
#Make a copy for y data
crimeData$y <- NULL
#Erase y parameter for now
crimeData <- as.matrix(crimeData)
#Convert to matrix
```

3.1 Performing PCA

Then as before we use `prcomp` to perform PCA on both the scaled and unscaled dataset.

```
PCS <- prcomp(crimeData, center = TRUE, scale = FALSE)
PCS_Scaled <- prcomp(crimeData, center = TRUE, scale = TRUE)
#Perform PCA using prcomp

plCrime1 <- ggbiplot(PCS)+
  ggtitle("Unscaled")
plCrime2 <- ggbiplot(PCS_Scaled)+
  ggtitle("Scaled")

# Arrange the plots side by side

combined_plot <- gridExtra::arrangeGrob(plCrime1, plCrime2, ncol = 2)

grid::grid.draw(combined_plot)
```

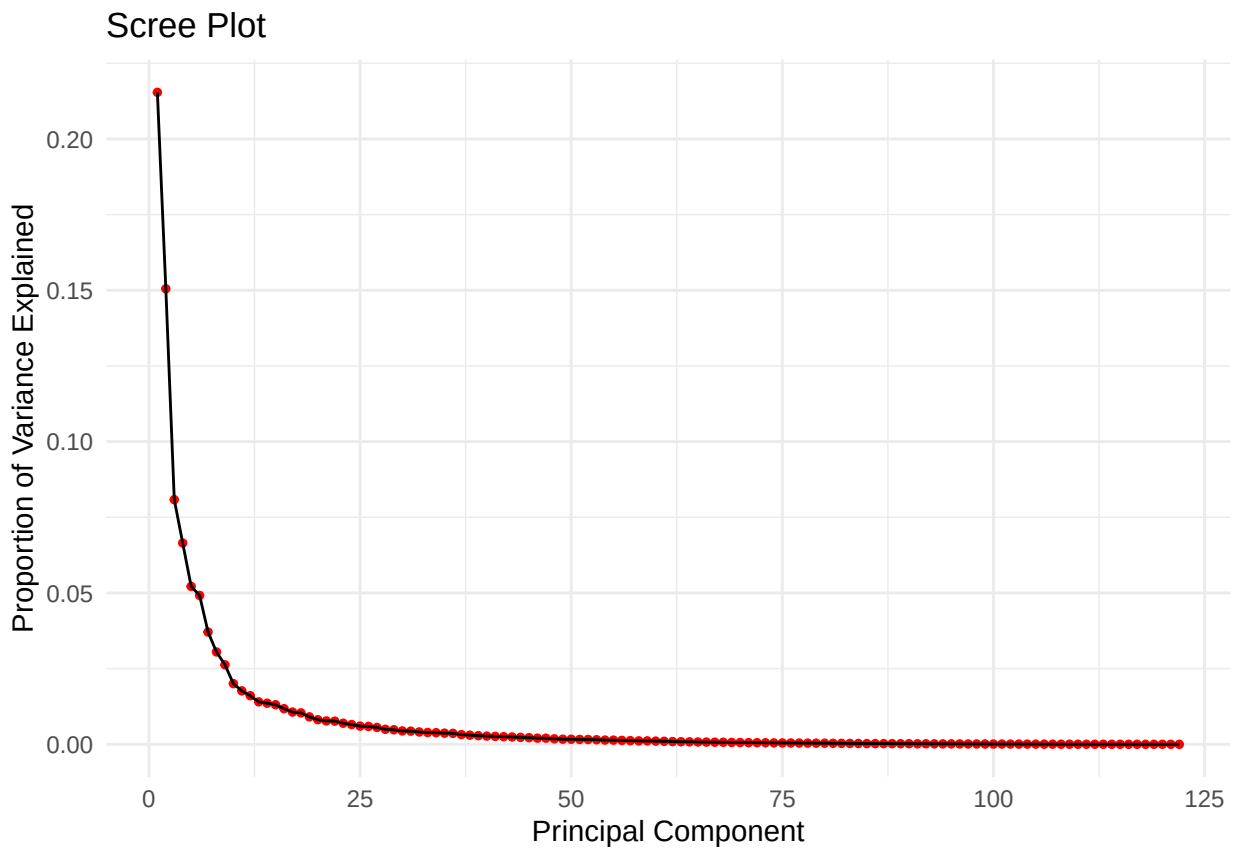


This is hell, I don't know why I thought this would be a good idea. Of course It's going to look this bad we have 123 variables. Let's push forward with the skree plot anyway to find out how many principle components we should keep. We can conclude that using the scaled (correlation matrix) version is probably more informative.

3.2 Component Selection

3.2.1 Scree Plot

```r
# Extract the proportion of variance
variance <- PCS_Scaled$sdev^2
variance_ratio <- variance / sum(variance)

# Create a data frame
df <- data.frame(Dimension = 1:length(variance_ratio), Variance = variance_ratio)

# Create the scree plot
ggplot(df, aes(x = Dimension, y = Variance)) +
  geom_point(color = "red",size = 1) +
  geom_line() +
  theme_minimal() +
  labs(x = "Principal Component", y = "Proportion of Variance Explained",
       title = "Scree Plot")
```



We can see that we have a fairly large amount of principle components before we get diminishing returns. Again to formalize this we'll use Kaiser's criterion.

3.2.2 Kaiser's criterion

```
PC_eigen <- PCS_Scaled$sdev^2
#Take eigenvalues of PCS
q_K <- max(which(PC_eigen > mean(PC_eigen)))
# Compute q_K

q_K
```

```
## [1] 19
```

Kaiser has demanded we only use 19 principal components, and that is what we will do.

3.3 Performing PCA Regression

When performing PCA regression, we first reduce the datset to some number of principal components, then perform a regression algorithm, such as least squares. Then lift our parameters into the original state space.

3.3.1 Regression on PC1-PC19

```
PC_1to19 <- PCS_Scaled$rotation[,1:19]
#Extract the first 19 principal components
Y_pc <- crimeData%*%PC_1to19
#Reduce the dataset to PC1-PC19

Y_pc <- cbind(Y_pc, y = crimeDataCOPY$y)
#Add the y parameter back in

pcRegression <- lm(y~.,data = dplyr::as_tibble(Y_pc))
#Perform regression on PC1-PC19
a <- pcRegression$coefficients[1]
gamma <- pcRegression$coefficients[-1]
```

```
print(a)
```

```
## (Intercept)
##   0.4196766
```

```
print(gamma)
```

```
##         PC1         PC2         PC3         PC4         PC5         PC6
## -0.14092383  0.08222577 -0.08951306  0.05963090  0.03825884  0.03727827
##         PC7         PC8         PC9        PC10        PC11        PC12
##  0.02571112  0.12793322  0.01414416 -0.03232895 -0.03139392  0.02314632
##        PC13        PC14        PC15        PC16        PC17        PC18
##  0.01691285  0.03227959  0.04035222  0.02384719 -0.02814656  0.01333715
##        PC19
##  0.01807441
```

Here we have our linear regression for the first 19 principal components, if we wanted to focus on in those componenets we could stop here, but we have the ability to lift these to our original dataset.

3.3.2 Lifting the Regression

```
beta <- PC_1to19%*%gamma
#Lift the gradient to the original state space
alpha <- a - (t(gamma))%*%(t((PCS_Scaled$rotation[,1:19]))%*%colMeans(crimeData))
#Lift the intercept to the original state space
```

```
head(beta)
```

```
##               [,1]
## x.V6   0.026708402
## x.V7  -0.008497064
## x.V8   0.074574294
## x.V9  -0.065792421
## x.V10  0.007370439
## x.V11  0.001132402
```

```
alpha
```

```
##           [,1]
## [1,] 0.6013743
```

Now we have a regression algorithm which is defined over the entire state space- thanks to the magic of PCA!

3.3.3 Different amounts of PC removal

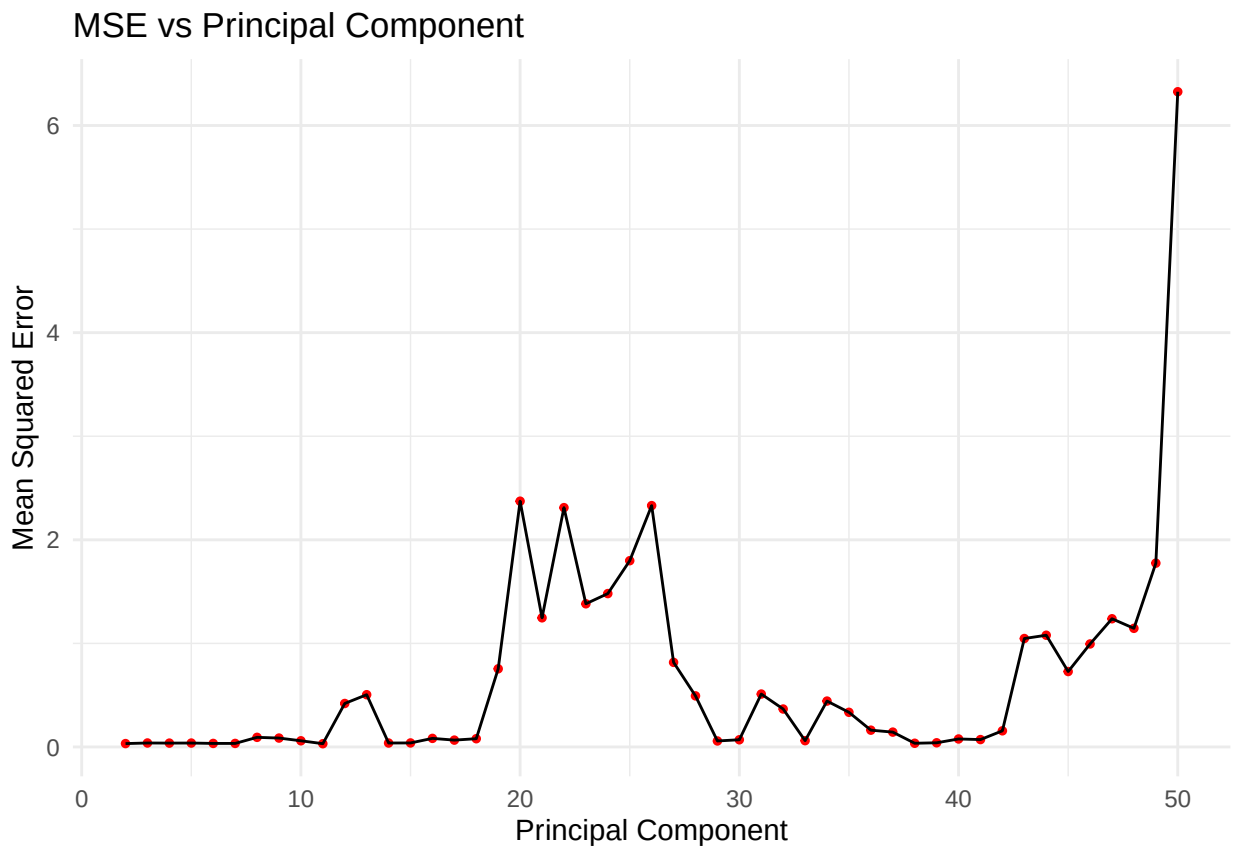
We don't just need to stick to PC19 and above, as we picked this kind of arbitrarily. We construct a function which mimics our previous code but for a general number of principle components and then measure the prediction error.

```
PCRegression <- function(n){
  TrainingData <- crimeData[1:100,]
  PC_n <- PCS_Scaled$rotation[,1:n]
  #Extract the first n principal components
  Y_pc <- TrainingData%*%PC_n
  #Reduce the dataset to PC1-PCn
  TrainingY <- (crimeDataCOPY$y)[1:100]
  Y_pc <- cbind(Y_pc, y = TrainingY)
  #Add the y parameter back in

  pcRegression <- lm(y~.,data = dplyr::as_tibble(Y_pc))
  #Perform regression on PC1-PCn
  a <- pcRegression$coefficients[1]
  gamma <- pcRegression$coefficients[-1]

  beta <- PC_n%*%gamma
  #Lift the gradient to the original state space
  alpha <- a - (t(gamma))%*%(t(PC_n)%*%colMeans(crimeData))
  #Lift the intercept to the original state space

  return(list(alpha = alpha, beta = beta))
}

MSE_PCR <- function(n){
  TestData <- crimeData[101:199,]
  TestY <- (crimeDataCOPY$y)[101:199]
  Reg <- PCRegression(n)
  MSE <- mean((TestY - (TestData%*%Reg$beta + as.numeric(Reg$alpha)))^2)
  return(MSE)
}
```

Now we have the functions to perform our task, lets iterate over the first 100 principal components and plot the error.

```
MSE <- sapply(2:50,MSE_PCR)
#Iterate over the first 100 principal components

Error_data <- data.frame(Dimension = 2:50, MSE = MSE)
```

```r
#Create a dataframe

ggplot(Error_data, aes(x = Dimension, y = MSE)) +
  geom_point(color = "red",size = 1) +
  geom_line() +
  theme_minimal() +
  labs(x = "Principal Component", y = "Mean Squared Error",
       title = "MSE vs Principal Component")
```



What a cool graph! Mean Squared Error is pretty low for the first few principal components, but then rockets up when we begin to include all of the components. This is a result of overfitting, as our coefficients are fitting the training set too closely. This is not necessarily a result of any PCA operation, just of how regression can often overfit. If we increase this further we see more and more overfitting. This graph also backs up the conslusion from Kasier's criterion as the lowest MSE is around the 19 region.