PCA Portfolio

Rachel Wood

2023-01-27

Task 1

Using the Covariance Matrix

We first load the data and obtain the centered matrix:

```
data("USArrests")
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
## Connecticut       3.3     110       77 11.1
## Delaware          5.9     238       72 15.8
## Florida          15.4     335       80 31.9
## Georgia          17.4     211       60 25.8
## Hawaii            5.3      46       83 20.2
## Idaho             2.6     120       54 14.2
## Illinois         10.4     249       83 24.0
## Indiana           7.2     113       65 21.0
## Iowa              2.2      56       57 11.3
## Kansas            6.0     115       66 18.0
## Kentucky          9.7     109       52 16.3
## Louisiana        15.4     249       66 22.2
## Maine             2.1      83       51  7.8
## Maryland         11.3     300       67 27.8
## Massachusetts     4.4     149       85 16.3
## Michigan         12.1     255       74 35.1
## Minnesota         2.7      72       66 14.9
## Mississippi      16.1     259       44 17.1
## Missouri          9.0     178       70 28.2
## Montana           6.0     109       53 16.4
## Nebraska          4.3     102       62 16.5
## Nevada           12.2     252       81 46.0
## New Hampshire     2.1      57       56  9.5
## New Jersey        7.4     159       89 18.8
```

```
## New Mexico       11.4     285       70 32.1
## New York         11.1     254       86 26.1
## North Carolina   13.0     337       45 16.1
## North Dakota      0.8      45       44  7.3
## Ohio              7.3     120       75 21.4
## Oklahoma          6.6     151       68 20.0
## Oregon            4.9     159       67 29.3
## Pennsylvania      6.3     106       72 14.9
## Rhode Island      3.4     174       87  8.3
## South Carolina   14.4     279       48 22.5
## South Dakota      3.8      86       45 12.8
## Tennessee        13.2     188       59 26.9
## Texas            12.7     201       80 25.5
## Utah              3.2     120       80 22.9
## Vermont           2.2      48       32 11.2
## Virginia          8.5     156       63 20.7
## Washington        4.0     145       73 26.2
## West Virginia     5.7      81       39  9.3
## Wisconsin         2.6      53       66 10.8
## Wyoming           6.8     161       60 15.6
```

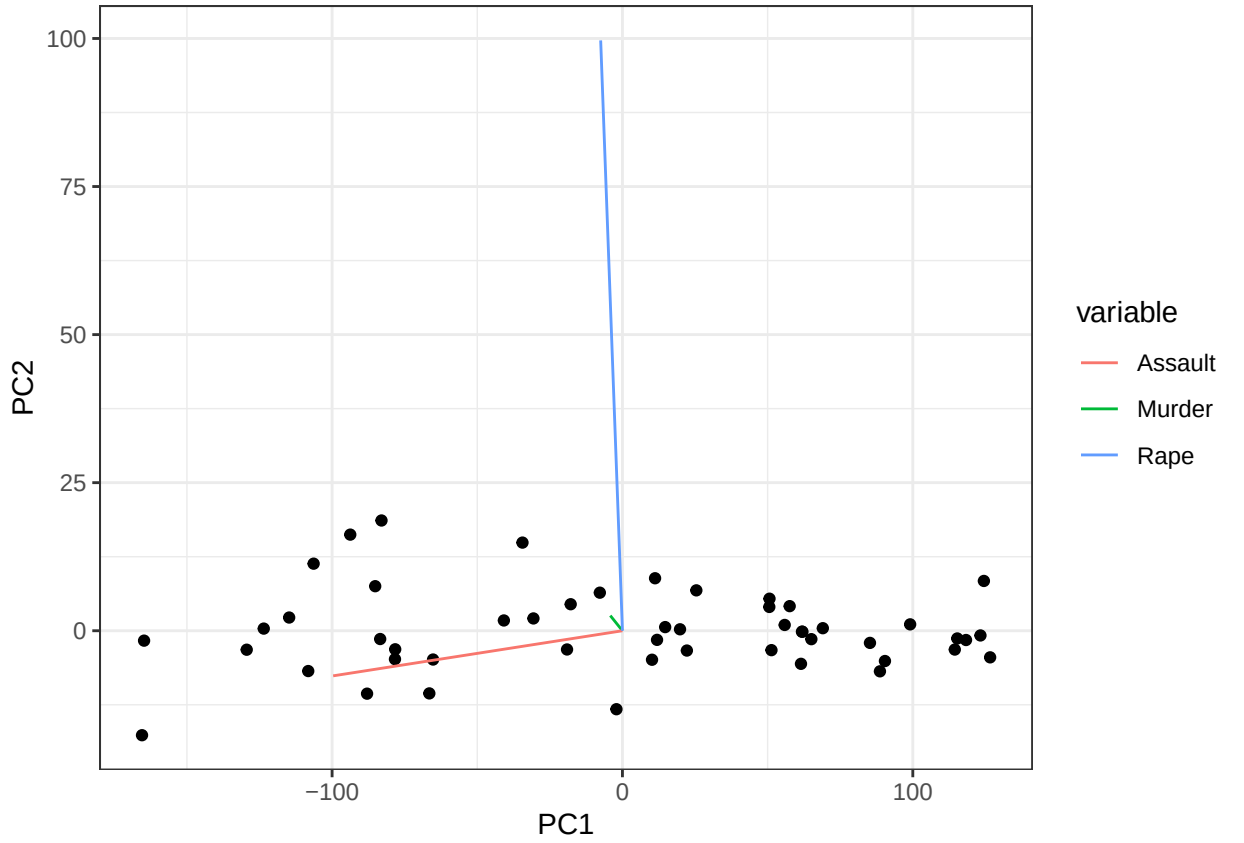
```r
X_0 <- USArrests %>%
  select(-UrbanPop) %>%
  as.matrix()

n <- nrow(X_0)

C_n <- diag(nrow = n) - matrix(1, nrow = n, ncol = n) /n

X <- C_n %*% X_0
```

Using the Covariance Matrix

We now obtain the covariance matrix and perform PCA manually

```r
S <- t(X) %*% X /n

S_eigen <- eigen(S)

v <- S_eigen$values

A <- S_eigen$vectors


Y <- X %*% A

colnames(Y) <- c("PC1", "PC2", "PC3")


colnames(A) <- c("PC1", "PC2", "PC3")
```

```r
A <- A* 100
A <- as_tibble(A) %>%
  dplyr::mutate( xstart = 0, ystart = 0, variable = colnames(X_0) , .before =1)


Y <- as_tibble(Y)

ggplot() +
  geom_point(data = Y, aes(PC1, PC2)) +
  geom_segment(data = A, aes(x =xstart, y= ystart, xend = PC1, yend = PC2, color = variable)) +
  labs(x = "PC1", y = "PC2")
```



Using the Correlation Matrix

We can use the function `prcomp` to perform PCA. We specify `scale. = TRUE` to ensure the correlation matrix is used instead of the covariance matrix. The function `prcomp()` returns a list with 3 elements:

- sdev: the eigenvalues of the corelation matrix. The diagonal values of Λ are the square of these
- rotation: the matrix with columns made up of the eigenvectors of the correlation matrix. This is equivalent to the A matrix from the lecture notes
- x: the values of the original dataset after projection onto the principle components

```
arrest_PCA <- prcomp(~ Murder + Assault + Rape, data = USArrests, scale. = TRUE)
arrest_PCA
```

```
## 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
```

The first column of the rotation matrix corresponds to the first principle components. Since all these are the same sign, this likely represents a weighted average of the variables. The second component represents a contrast between Rape and the other rates.

We can now use `ggbiplot` to plot the vectors of the original variables and new positions of the data points for the first two components:

```
library(ggbiplot)
```

```
## Loading required package: plyr
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

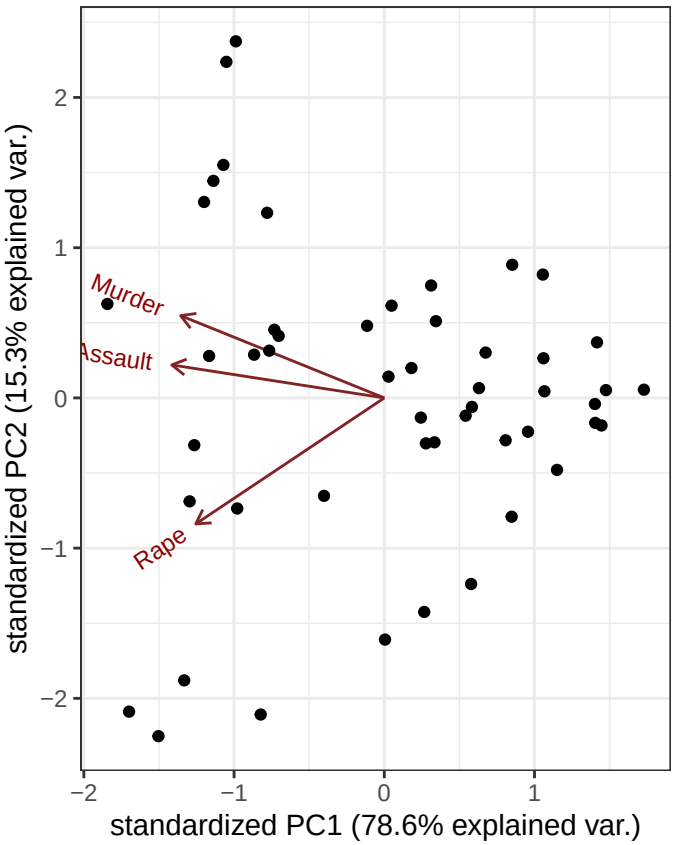
```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## Loading required package: scales
```

```
## Loading required package: grid
```

```
ggbiplot(arrest_PCA)
```



Given the values of the principle component vectors, we know the observations on the right hand side of the plot would correspond to states with lower crime rates in general, with the opposite being true for the left hand side.

We also expect the states below the x-axis to have higher rates of rape but lower rates of murder and assault than the average, and vice versa for the states above the x-axis.

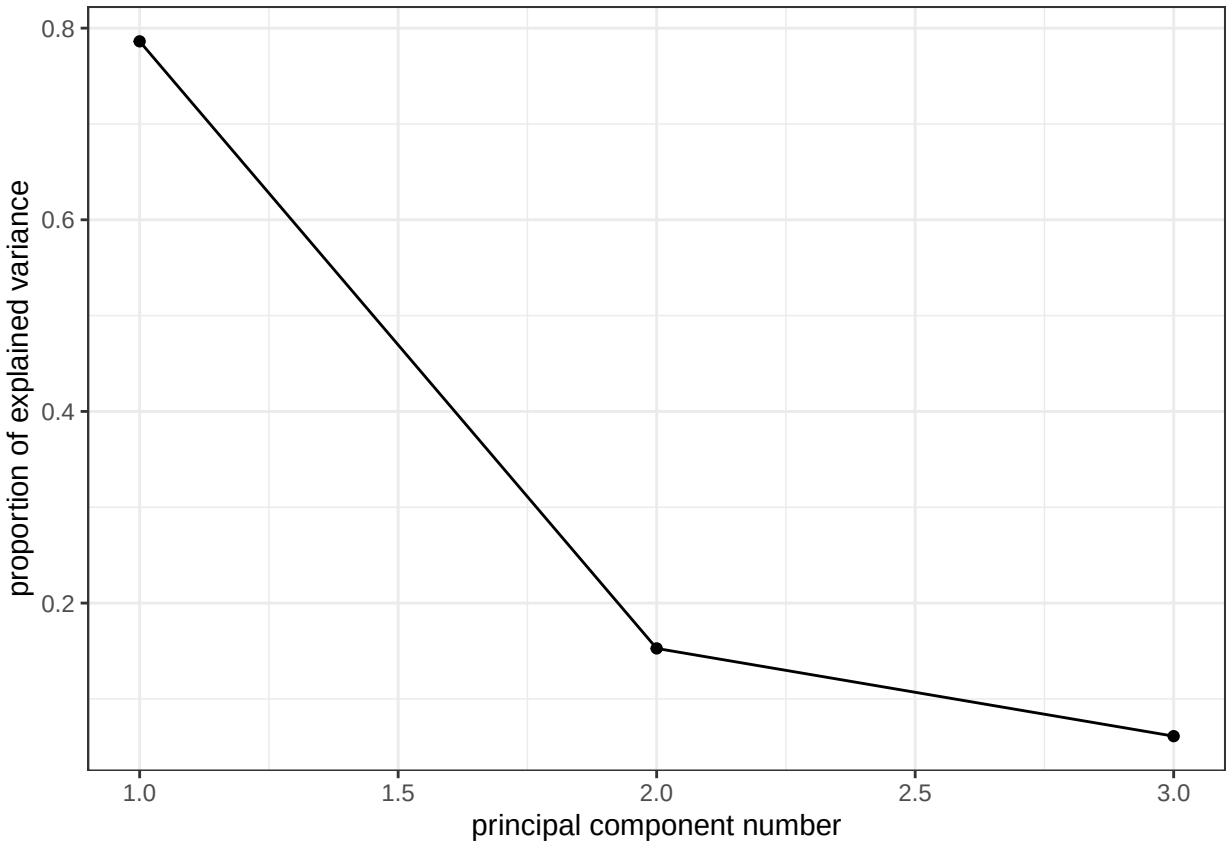
Correlation vs. Covariance

It's clear our model based off the correlation matrix is preferable as this scales our variables equally as opposed to one variable dominating the analysis. For the rest of this section thus, we will only consider the correlation matrix.

Choosing the number of Principle Components

We now make a screeplot and compute q_K and q_H:

```
ggscreeplot(arrest_PCA)
```




```
q_K <- max(which(arrest_PCA$sdev^2 >  mean(arrest_PCA$sdev ^2)))
q_K
```

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

```
D <- diag(arrest_PCA$sdev^2)
M <- 50
n<- nrow(X)
p <- ncol(X)

eigenvals<- matrix(nrow = M, ncol = p)
for (m in 1:M){
  X_m <- matrix(rnorm(n*p), nrow = n, ncol = p)

  R_m <- t(X_m) %*% X_m /n
  S_m <- sqrt(D) %*% R_m %*% sqrt(D)

  eigenvals[m,] <- eigen(S_m)$values
}
mean_eigen <- colMeans(eigenvals)

q_H <- max(which(arrest_PCA$sdev^2 > mean_eigen))
q_H
```

```
## [1] 3
```

We then choose 2 components as a compromise of the two criteria.

Task 2

For this task we use the prestige data set, a Canadian dataset containing the levels of prestige, income, and female participation in a variety of professions

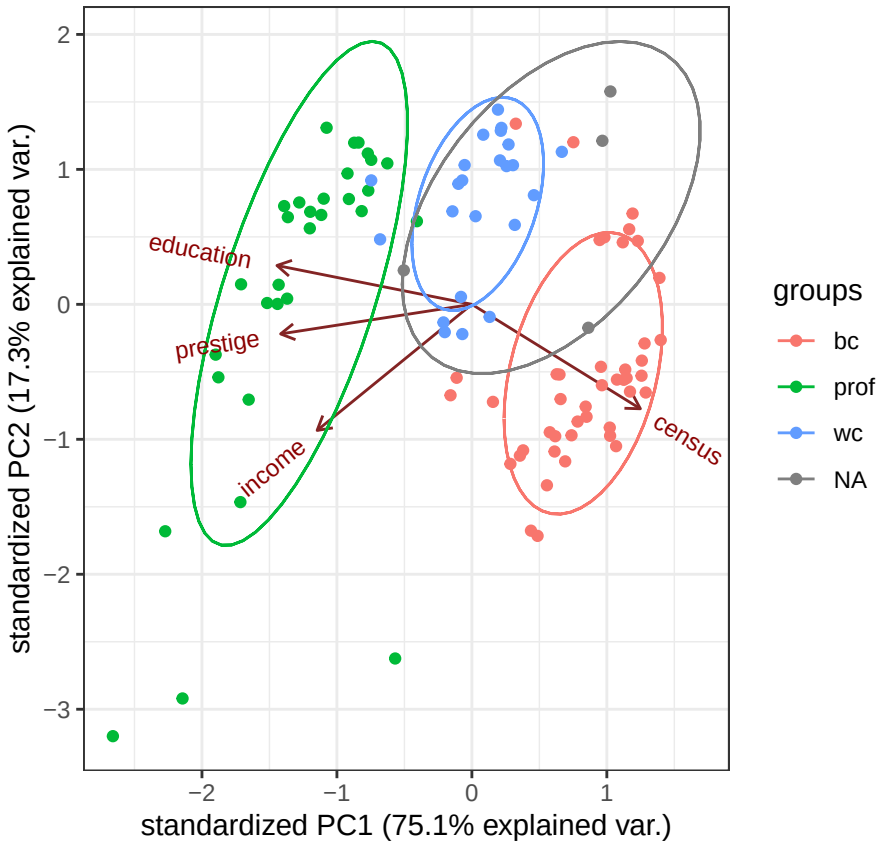
```
library(carData)
data("Prestige")
head(Prestige)
```

```
##                    education income women prestige census type
## gov.administrators      13.11  12351 11.16     68.8   1113 prof
## general.managers        12.26  25879  4.02     69.1   1130 prof
## accountants             12.77   9271 15.70     63.4   1171 prof
## purchasing.officers     11.42   8865  9.11     56.8   1175 prof
## chemists                14.62   8403 11.68     73.5   2111 prof
## physicists              15.64  11030  5.13     77.6   2113 prof
```

We can see the covariates are on different scales and thus we choose to use the corelation matrix to avoid one variable dominating the components over the others.

We fit the model and plot our projected observations with their color associated with the `type` of the profession:

```
prestige_pca <- prcomp(~education + income + prestige + census, data = Prestige, scale. = TRUE)

ggbiplot(prestige_pca, groups = Prestige$type, ellipse = TRUE)
```



We can see our components provide quite distinct groupings for each of the types, with the exception of the missing data values. It is likely that these are missing due to them not fitting well into the other categories and thus it makes sense that they are not particularly close to each other.

Task 3

For this task, we use the Crimes and Community dataset to perform PCR. First we load the data:

```
library(mogavs)

data("crimeData")
head(crimeData)
```

```
##   x.V6 x.V7 x.V8 x.V9 x.V10 x.V11 x.V12 x.V13 x.V14 x.V15 x.V16 x.V17 x.V18
## 1 0.19 0.33 0.02 0.90  0.12  0.17  0.34  0.47  0.29  0.32  0.20   1.0  0.37
## 2 0.00 0.16 0.12 0.74  0.45  0.07  0.26  0.59  0.35  0.27  0.02   1.0  0.31
## 3 0.00 0.42 0.49 0.56  0.17  0.04  0.39  0.47  0.28  0.32  0.00   0.0  0.30
## 4 0.04 0.77 1.00 0.08  0.12  0.10  0.51  0.50  0.34  0.21  0.06   1.0  0.58
## 5 0.01 0.55 0.02 0.95  0.09  0.05  0.38  0.38  0.23  0.36  0.02   0.9  0.50
## 6 0.02 0.28 0.06 0.54  1.00  0.25  0.31  0.48  0.27  0.37  0.04   1.0  0.52
##   x.V19 x.V20 x.V21 x.V22 x.V23 x.V24 x.V25 x.V26 x.V27 x.V28         x.V29
## 1  0.72  0.34  0.60  0.29  0.15  0.43  0.39  0.40  0.39  0.32  2.700000e-01
## 2  0.72  0.11  0.45  0.25  0.29  0.39  0.29  0.37  0.38  0.33  1.600000e-01
## 3  0.58  0.19  0.39  0.38  0.40  0.84  0.28  0.27  0.29  0.27  7.000000e-02
## 4  0.89  0.21  0.43  0.36  0.20  0.82  0.51  0.36  0.40  0.39  1.600000e-01
```

```
## 5  0.72  0.16  0.68  0.44  0.11  0.71  0.46  0.43  0.41  0.28 -2.775558e-17
## 6  0.68  0.20  0.61  0.28  0.15  0.25  0.62  0.72  0.76  0.77  2.800000e-01
##   x.V30 x.V31 x.V32 x.V33 x.V34 x.V35 x.V36 x.V37 x.V38 x.V39 x.V40 x.V41 x.V42
## 1  0.27  0.36  0.41  0.08  0.19  0.10  0.18  0.48  0.27  0.68  0.23  0.41  0.25
## 2  0.30  0.22  0.35  0.01  0.24  0.14  0.24  0.30  0.27  0.73  0.57  0.15  0.42
## 3  0.29  0.28  0.39  0.01  0.27  0.27  0.43  0.19  0.36  0.58  0.32  0.29  0.49
## 4  0.25  0.36  0.44  0.01  0.10  0.09  0.25  0.31  0.33  0.71  0.36  0.45  0.37
## 5  0.74  0.51  0.48  0.00  0.06  0.25  0.30  0.33  0.12  0.65  0.67  0.38  0.42
## 6  0.52  0.48  0.60  0.01  0.12  0.13  0.12  0.80  0.10  0.65  0.19  0.77  0.06
##   x.V43 x.V44 x.V45 x.V46 x.V47 x.V48 x.V49 x.V50 x.V51 x.V52 x.V53 x.V54 x.V55
## 1  0.52  0.68  0.40  0.75  0.75  0.35  0.55  0.59  0.61  0.56  0.74  0.76  0.04
## 2  0.36  1.00  0.63  0.91  1.00  0.29  0.43  0.47  0.60  0.39  0.46  0.53  0.00
## 3  0.32  0.63  0.41  0.71  0.70  0.45  0.42  0.44  0.43  0.43  0.71  0.67  0.01
## 4  0.39  0.34  0.45  0.49  0.44  0.75  0.65  0.54  0.83  0.65  0.85  0.86  0.03
## 5  0.46  0.22  0.27  0.20  0.21  0.51  0.91  0.91  0.89  0.85  0.40  0.60  0.00
## 6  0.91  0.49  0.57  0.61  0.58  0.44  0.62  0.69  0.87  0.53  0.30  0.43  0.00
##   x.V56 x.V57 x.V58 x.V59 x.V60 x.V61 x.V62 x.V63 x.V64 x.V65 x.V66 x.V67 x.V68
## 1  0.14  0.03  0.24  0.27  0.37  0.39  0.07  0.07  0.08  0.08  0.89  0.06  0.14
## 2  0.24  0.01  0.52  0.62  0.64  0.63  0.25  0.27  0.25  0.23  0.84  0.10  0.16
## 3  0.46  0.00  0.07  0.06  0.15  0.19  0.02  0.02  0.04  0.05  0.88  0.04  0.20
## 4  0.33  0.02  0.11  0.20  0.30  0.31  0.05  0.08  0.11  0.11  0.81  0.08  0.56
## 5  0.06  0.00  0.03  0.07  0.20  0.27  0.01  0.02  0.04  0.05  0.88  0.05  0.16
## 6  0.11  0.04  0.30  0.35  0.43  0.47  0.50  0.50  0.56  0.57  0.45  0.28  0.25
##   x.V69 x.V70 x.V71 x.V72 x.V73 x.V74 x.V75        x.V76 x.V77 x.V78 x.V79
## 1  0.13  0.33  0.39  0.28  0.55  0.09  0.51 5.000000e-01  0.21  0.71  0.52
## 2  0.10  0.17  0.29  0.17  0.26  0.20  0.82 5.551115e-17  0.02  0.79  0.24
## 3  0.20  0.46  0.52  0.43  0.42  0.15  0.51 5.000000e-01  0.01  0.86  0.41
## 4  0.62  0.85  0.77  1.00  0.94  0.12  0.01 5.000000e-01  0.01  0.97  0.96
## 5  0.19  0.59  0.60  0.37  0.89  0.02  0.19 5.000000e-01  0.01  0.89  0.87
## 6  0.19  0.29  0.53  0.18  0.39  0.26  0.73 5.551115e-17  0.02  0.84  0.30
##   x.V80 x.V81 x.V82 x.V83 x.V84 x.V85 x.V86 x.V87 x.V88 x.V89 x.V90 x.V91 x.V92
## 1  0.05  0.26  0.65  0.14  0.06  0.22  0.19  0.18  0.36  0.35  0.38  0.34  0.38
## 2  0.02  0.25  0.65  0.16  0.00  0.21  0.20  0.21  0.42  0.38  0.40  0.37  0.29
## 3  0.29  0.30  0.52  0.47  0.45  0.18  0.17  0.16  0.27  0.29  0.27  0.31  0.48
## 4  0.60  0.47  0.52  0.11  0.11  0.24  0.21  0.19  0.75  0.70  0.77  0.89  0.63
## 5  0.04  0.55  0.73  0.05  0.14  0.31  0.31  0.30  0.40  0.36  0.38  0.38  0.22
## 6  0.16  0.28  0.25  0.02  0.05  0.94  1.00  1.00  0.67  0.63  0.68  0.62  0.47
##   x.V93 x.V94 x.V95 x.V96 x.V97 x.V98 x.V99 x.V100 x.V101      x.V102
## 1  0.46  0.25  0.04     0  0.12  0.42  0.50   0.51   0.64  0.03000000
## 2  0.32  0.18  0.00     0  0.21  0.50  0.34   0.60   0.52  0.01451576
## 3  0.39  0.28  0.00     0  0.14  0.49  0.54   0.67   0.56  0.14547393
## 4  0.51  0.47  0.00     0  0.19  0.30  0.73   0.64   0.65  0.17316663
## 5  0.51  0.21  0.00     0  0.11  0.72  0.64   0.61   0.53  0.04764374
## 6  0.59  0.11  0.00     0  0.70  0.42  0.49   0.73   0.64 -0.05970565
##        x.V103    x.V104      x.V105      x.V106      x.V107       x.V108
## 1  0.13000000 0.9600000  0.17000000  0.06000000  0.18000000  0.440000000
## 2  0.08440597 0.9410779  0.12599033  0.15974948  0.12538673  0.401382740
## 3  0.93142475 0.8794121  0.72976438 -0.04309098  0.47068554 -0.008239111
## 4  0.13268258 0.7745510  0.23644131  0.32323375  0.32453268  0.386322043
## 5  0.10386955 0.9069438  0.14177808  0.10681839 -0.03054083  0.261017040
## 6 -0.14598767 1.0355235 -0.05264927  0.04062974  0.11050494  0.161569540
##        x.V109    x.V110       x.V111     x.V112     x.V113     x.V114    x.V115
## 1  0.13000000 0.9400000  0.930000000 0.03000000 0.07000000  0.1000000 0.0700000
## 2  0.08395217 0.7219940  0.738630710 0.03069683 0.11414126  0.2443999 0.1078572
```

```
## 3  0.93219145 0.6493796  0.371866581 0.26713529 0.23826430  0.3090536 0.4379136
## 4  0.13399135 0.3109949 -0.000452741 0.94891382 0.21127942 -0.2684516 0.9272990
## 5  0.10335618 0.7163808  0.659744223 0.23109376 0.05902612 -0.1922939 0.1883248
## 6 -0.14549389 0.7700580  0.468699564 0.01006291 0.42741082  0.5833132 0.4056361
##        x.V116      x.V117      x.V118 x.V119 x.V120 x.V121      x.V122
## 1  0.02000000  0.57000000  0.29000000   0.12   0.26   0.20  0.06000000
## 2 -0.13244800  0.56198920  0.50084956   0.02   0.12   0.45  0.01079104
## 3  0.18943152  0.37269214  0.78329534   0.01   0.21   0.02  0.31819989
## 4  0.17896430 -0.05816711  0.46450793   0.02   0.39   0.28  0.22252338
## 5  0.08234996  0.22091197 -0.01927682   0.04   0.09   0.02  0.16656419
## 6 -0.06383089 -0.05974813  0.34212041   0.01   0.58   0.10 -0.01337640
##        x.V123      x.V124     x.V125 x.V126     x.V127    y
## 1  0.04000000  0.90000000  0.5000000   0.32 0.14000000 0.20
## 2  0.03319794  1.00349596  0.8909102   0.00 0.11741576 0.67
## 3  0.14312878 -0.06232894 -0.4010886   0.00 0.89509664 0.43
## 4  0.18831739  0.76754173  0.2852024   0.00 0.12034589 0.12
## 5  0.03137339  1.06875289  0.9080916   0.00 0.19743955 0.03
## 6 -0.03206947  1.14041670  0.3868220   0.00 0.03310828 0.14
```

We see that there are many explanatory variables and a response `y`. We model this with a linear model:

$$y_i \ \alpha + \beta^T x_i^0$$

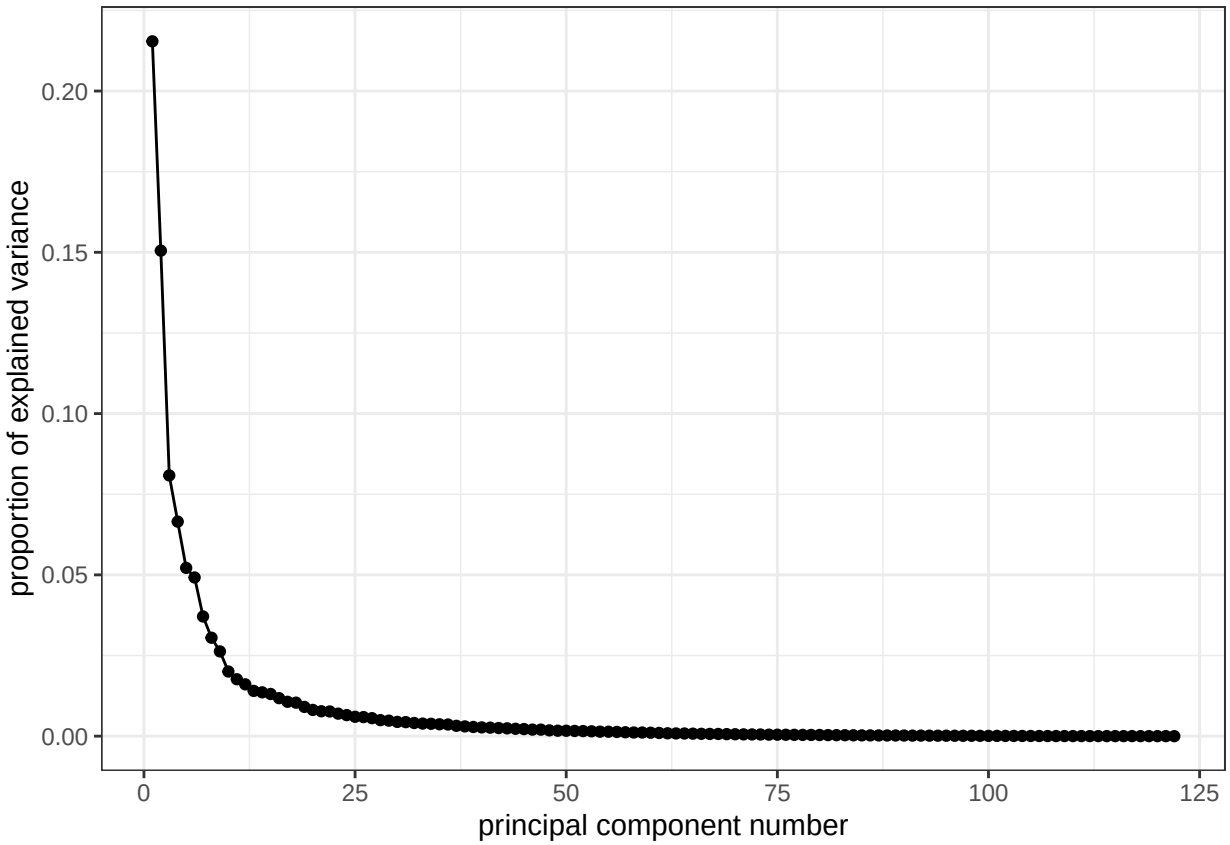
Fitting a PCA Model

We first perform PCA on the explanatory variables

```
crimeData <- as_tibble(crimeData)
X <- crimeData %>%
  dplyr::select(- y)
y <- crimeData %>%
  dplyr::select(y)
crime_PCA <- prcomp(X, scale. = TRUE)
```

Choosing the Number of Components

```
ggscreeplot(crime_PCA)
```



We can see from the plot the proportion of explained variance deteriorates quickly. We use Kaiser's criterion and Horn's parallel analysis to select an appropriate number of components.

Kaiser's information suggests we keep the components whose corresponding eigenvalues are greater than the mean:

```
q_K <- max(which(crime_PCA$sdev^2 >  mean(crime_PCA$sdev ^2)))
q_K
```

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

We can use the `paran` package to implement Horn's parallel analysis and obtain, q_H the number of components we should keep

```
library(paran)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
q_H <- paran(scale(X))
```

```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 3660 iterations, using the mean estimate
##
## ------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## ------------------------------------------------
## 1          25.746367   26.278998      0.532630
## 2          17.857791   18.362765      0.504973
## 3           9.376851   9.861050      0.484198
## 4           7.646746    8.113534      0.466787
## 5           5.913634    6.364539      0.450905
## 6           5.565615    6.001657      0.436041
## 7           4.103374    4.525737      0.422363
## 8           3.312588    3.722059      0.409470
## 9           2.808004    3.204856      0.396852
## 10          2.059599    2.444395      0.384796
## 11          1.779848    2.153235      0.373387
## 12          1.597175    1.959437      0.362261
## 13          1.359464    1.710966      0.351502
## 14          1.316213    1.657406      0.341193
## 15          1.264028    1.594835      0.330806
## 16          1.115809    1.436687      0.320878
## ------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (16 components retained)
```

We will use $q = q_H$ as Horn's analysis accounts for the fact that the observations in the original dataset are finite.

Performing PCR

We now obtain $A_{1:16}$ and corresponding $Y_{1:16}$ matrix:

```
A_q <- crime_PCA$rotation[,1:16]

Y_q <- as.matrix(X) %*% A_q
Y_q <- as_tibble(Y_q)
Y_q <- cbind(y, Y_q)
```

We can now fit a linear model using the first 16 components as covariates and obtain our a and γ

```
pcr_model <- lm(y ~., data = Y_q)

a <- pcr_model$coefficients[1]

gamma <- pcr_model$coefficients[-1]
```

We can use the coefficients from this model to obtain estimates of the α and β from our original model:

```
beta <- A_q %*% gamma

alpha <- a - colMeans(as.matrix(X))%*%  beta
```

PCR with different values of q

We first split our data into testing and training sets and obtain a PCA model on the training set. We also create a vector of possible values of η:

```
crimes.train <- crimeData[1:1500,]
y.train <- crimes.train %>%
  dplyr:: select(y)

X.train <- crimes.train %>%
  dplyr::select(-y)


crimes.test <- crimeData[-(1:1500),]
y.test <- crimes.test %>%
  dplyr:: select(y)

X.test <- crimes.test %>%
  dplyr::select(-y)

train_PCA <- prcomp(X.train)
```

We create a vector of possible values of η and obtain

$$q_\eta = \min\left\{ j : \frac{\sum_{k=1}^{j} \lambda_k}{\sum_{k=1}^{p} \lambda_k} > \eta \right\}$$

We then fit a linear model with q_η components and obtain the relevant prediction errors

```
eta <- seq(0.8, 0.999, 0.002)


errors <- vector(mode = "numeric", length = length(eta))

for (i in 1:length(eta)){
  sum_eigen <- cumsum(train_PCA$sdev^2)
  n <- length(sum_eigen)
  sum_eigen <- sum_eigen / sum_eigen[n]
  q_eta <- min(which(sum_eigen > eta[i]))
```

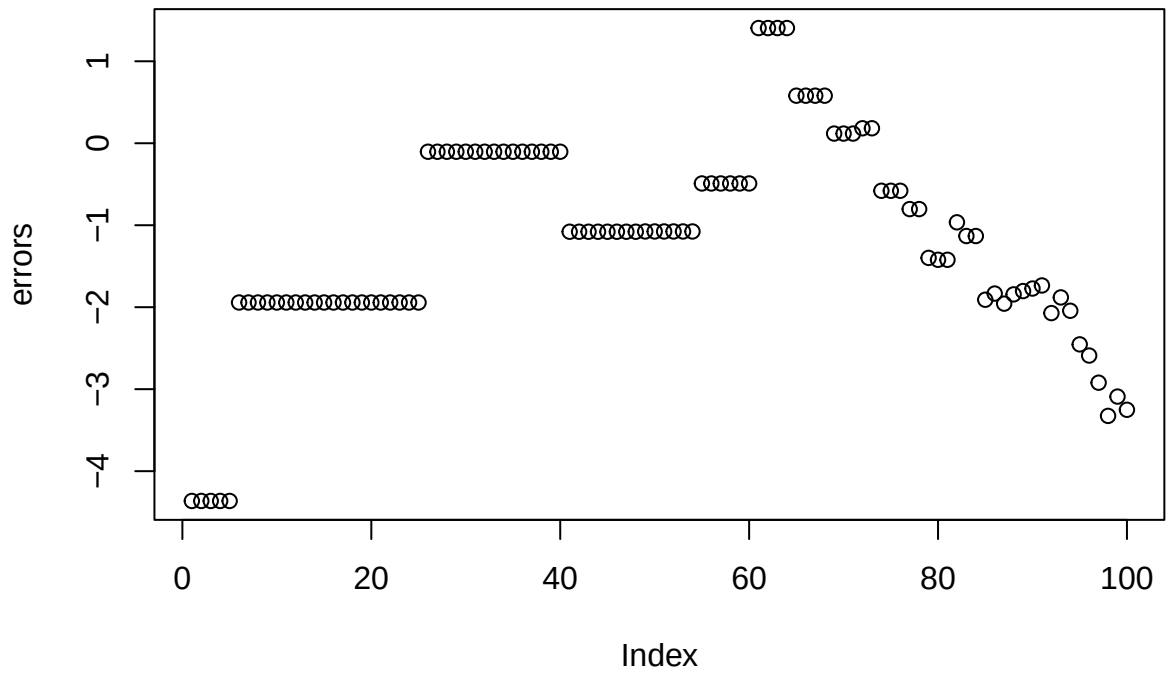
```
  print(q_eta)
  if (q_eta ==1){
    A_q <- train_PCA$rotation[ ,1]
    A_q <- as.matrix(A_q, ncol = 1)
    colnames(A_q) = "PC1"

  }else {
    A_q <- train_PCA$rotation[ ,1:q_eta]
  }

  Y_train <- as.matrix(X.train) %*% A_q

  Y_train <- as_tibble(Y_train) %>%
    cbind(y = y.train)
  train.model <- lm(y ~., data = Y_train)

  Y_test <- as.matrix(X.test) %*% A_q

  Y_test<- as_tibble(Y_test) %>%
    cbind( y= y.test)



  predicted <- predict.lm(train.model, Y_test)

  errors[i] <- sum(y.test - predicted)

}
```

```
## [1] 1
## [1] 1
## [1] 1
## [1] 1
## [1] 1
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
## [1] 2
```

```
## [1] 2
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 3
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 5
## [1] 5
## [1] 5
## [1] 5
## [1] 5
## [1] 5
## [1] 6
## [1] 6
## [1] 6
## [1] 6
## [1] 6
## [1] 6
## [1] 7
## [1] 7
## [1] 7
## [1] 7
## [1] 8
## [1] 8
## [1] 8
## [1] 8
## [1] 9
## [1] 9
## [1] 9
## [1] 10
## [1] 10
## [1] 11
## [1] 11
## [1] 11
## [1] 12
## [1] 12
```

```
## [1] 13
## [1] 14
## [1] 14
## [1] 15
## [1] 16
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 23
## [1] 24
## [1] 26
## [1] 28
## [1] 30
## [1] 33
## [1] 36
## [1] 40
## [1] 44
## [1] 51
## [1] 64
```

```
plot(errors)
```



```
error_data <- as_tibble(eta, errors)
ggplot(error_data, aes(x = eta, y = errors)) +
  geom_line()
```