R Notebook

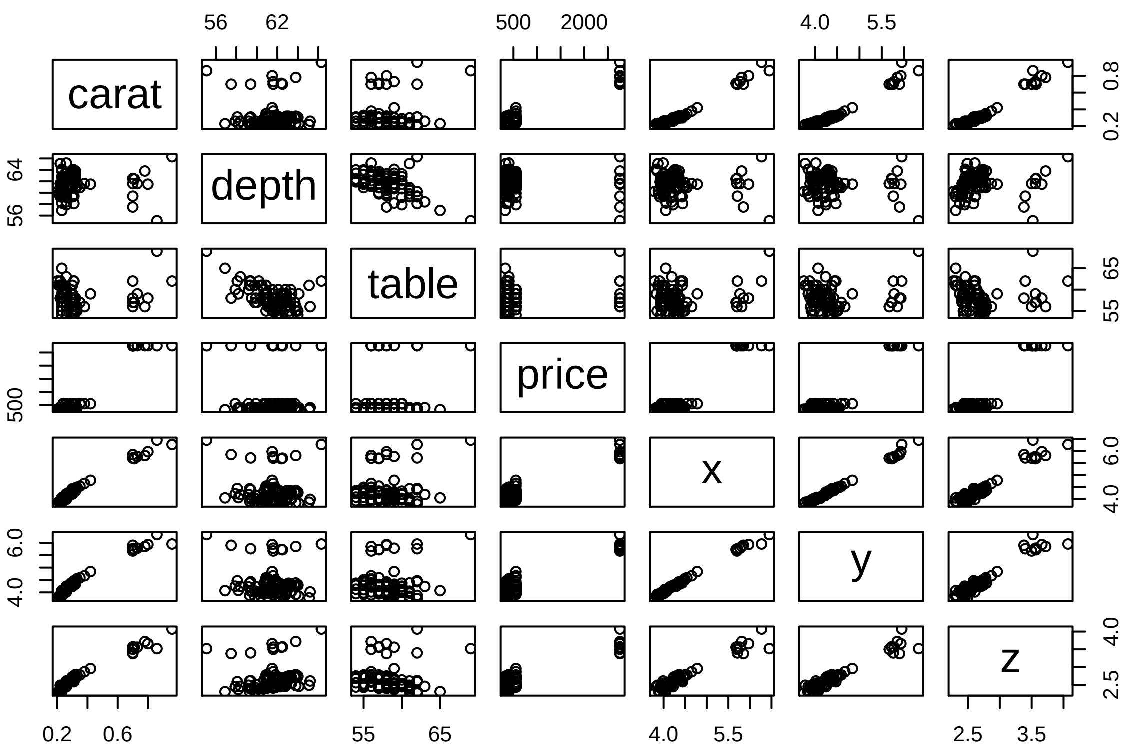
Principal Components Analysis

Let's look for the modes of variability in this diamonds dataset:

```r
library(ggplot2)
head(diamonds)
```

```
## # A tibble: 6 × 10
##   carat       cut color clarity depth table price     x     y     z
##   <dbl>     <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23     Ideal     E     SI2  61.5    55   326  3.95  3.98  2.43
## 2  0.21   Premium     E     SI1  59.8    61   326  3.89  3.84  2.31
## 3  0.23      Good     E     VS1  56.9    65   327  4.05  4.07  2.31
## 4  0.29   Premium     I     VS2  62.4    58   334  4.20  4.23  2.63
## 5  0.31      Good     J     SI2  63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336  3.94  3.96  2.48
```

```r
#let's make a big plot avoiding the three rows that aren't data
plot(diamonds[1:100,-c(2,3,4)])
```




```r
Xraw = diamonds[1:1000,-c(2,3,4)]
```

OK, now we need to calculate a covariance matrix

```r
#subtract the mean from each column. We can do this easily with scale()

#and use scale() to subtract that mean.
X = scale (Xraw, scale=TRUE)
```

```
#what does the scale=FALSE do?
X2= scale (Xraw, scale=FALSE)

head(X)
```

```
##          carat      depth      table     price         x         y
## [1,] -2.351777 -0.1266716 -1.1080875 -2.561461 -2.648771 -2.645833
## [2,] -2.454189 -1.0931966  1.3230841 -2.561461 -2.744745 -2.874601
## [3,] -2.351777 -2.7419744  2.9438653 -2.560270 -2.488815 -2.498768
## [4,] -2.044543  0.3850181  0.1074983 -2.551932 -2.248882 -2.237319
## [5,] -1.942131  0.8967077  0.1074983 -2.550741 -2.024944 -2.041232
## [6,] -2.300572  0.6124357 -0.2976970 -2.549550 -2.664767 -2.678514
##              z
## [1,] -2.635915
## [2,] -2.943750
## [3,] -2.943750
## [4,] -2.122856
## [5,] -1.815021
## [6,] -2.507650
```

```
head(X2)
```

```
##         carat   depth   table    price        x        y        z
## [1,] -0.45928 -0.2228 -2.7347 -2150.54 -1.65594 -1.61918 -1.02753
## [2,] -0.47928 -1.9228  3.2653 -2150.54 -1.71594 -1.75918 -1.14753
## [3,] -0.45928 -4.8228  7.2653 -2149.54 -1.55594 -1.52918 -1.14753
## [4,] -0.39928  0.6772  0.2653 -2142.54 -1.40594 -1.36918 -0.82753
## [5,] -0.37928  1.5772  0.2653 -2141.54 -1.26594 -1.24918 -0.70753
## [6,] -0.44928  1.0772 -0.7347 -2140.54 -1.66594 -1.63918 -0.97753
```

```
#let's calculate our covariance matrix:
coMat  = (t(X)%*%X)/ (nrow(X)-1)
head(coMat)
```

```
##            carat       depth       table       price           x
## carat 1.00000000  0.09528927  0.12698220  0.85755776  0.97759478
## depth 0.09528927  1.00000000 -0.34472255 -0.01258316 -0.05033068
## table 0.12698220 -0.34472255  1.00000000  0.06805856  0.13268231
## price 0.85755776 -0.01258316  0.06805856  1.00000000  0.91114241
## x     0.97759478 -0.05033068  0.13268231  0.91114241  1.00000000
## y     0.97257997 -0.05841130  0.11275493  0.91720237  0.99622344
##               y          z
## carat  0.9725800 0.98088588
## depth -0.0584113 0.21134454
## table  0.1127549 0.03173969
## price  0.9172024 0.89291925
## x      0.9962234 0.96422550
## y      1.0000000 0.96198408
```

```
#compare that to a correlation matrix
head(cor(Xraw))
```

```
##            carat       depth       table       price           x
## carat 1.00000000  0.09528927  0.12698220  0.85755776  0.97759478
## depth 0.09528927  1.00000000 -0.34472255 -0.01258316 -0.05033068
## table 0.12698220 -0.34472255  1.00000000  0.06805856  0.13268231
```

```
## price 0.85755776 -0.01258316  0.06805856  1.00000000  0.91114241
## x     0.97759478 -0.05033068  0.13268231  0.91114241  1.00000000
## y     0.97257997 -0.05841130  0.11275493  0.91720237  0.99622344
##                y          z
## carat  0.9725800 0.98088588
## depth -0.0584113 0.21134454
## table  0.1127549 0.03173969
## price  0.9172024 0.89291925
## x      0.9962234 0.96422550
## y      1.0000000 0.96198408
```

Aha, so a scaled covariance matrix is identical to a correlation matrix. Good to know.

So let's do PCA on the correlation matrix, since the units are different in our column

```r
#Now that we have our covariance/correlation matrix, we can use svd to find the eigen values and vector
out = svd(coMat)
names(out)
```

```
## [1] "d" "u" "v"
```

OK, remember our SVD formulation

$A = EDE^T$

Where A is our covariance matrix, E is our eigenvectors, and D is our Eigenvalues? svd names E=U, and E^T = V

We can also think our eigen matrix as our new covariance matrix, with zeros off the diagonal.

```r
print(out$d)#svd returns only, the diagonal.
```

```
## [1] 4.7880252909 1.3755317522 0.6721094682 0.1531485875 0.0073524900
## [6] 0.0032904137 0.0005419975
```

```r
#so our total covariance is just the sum of out$d
sum(out$d)
```

```
## [1] 7
```

```r
#and the fraction of variance in each is
out$d/sum(out$d)
```

```
## [1] 6.840036e-01 1.965045e-01 9.601564e-02 2.187837e-02 1.050356e-03
## [6] 4.700591e-04 7.742821e-05
```

OK, so now we have 7 indepenent modes of variability, that explain all the variance, independently.

But can we take our data, and "project it" onto this new coordinate system. Yes we can!

```r
PCs = (X %*% out$v) #this is our projection of our data on the eigen vectors. We call this the PC score
dim(PCs)
```

```
## [1] 1000    7
```

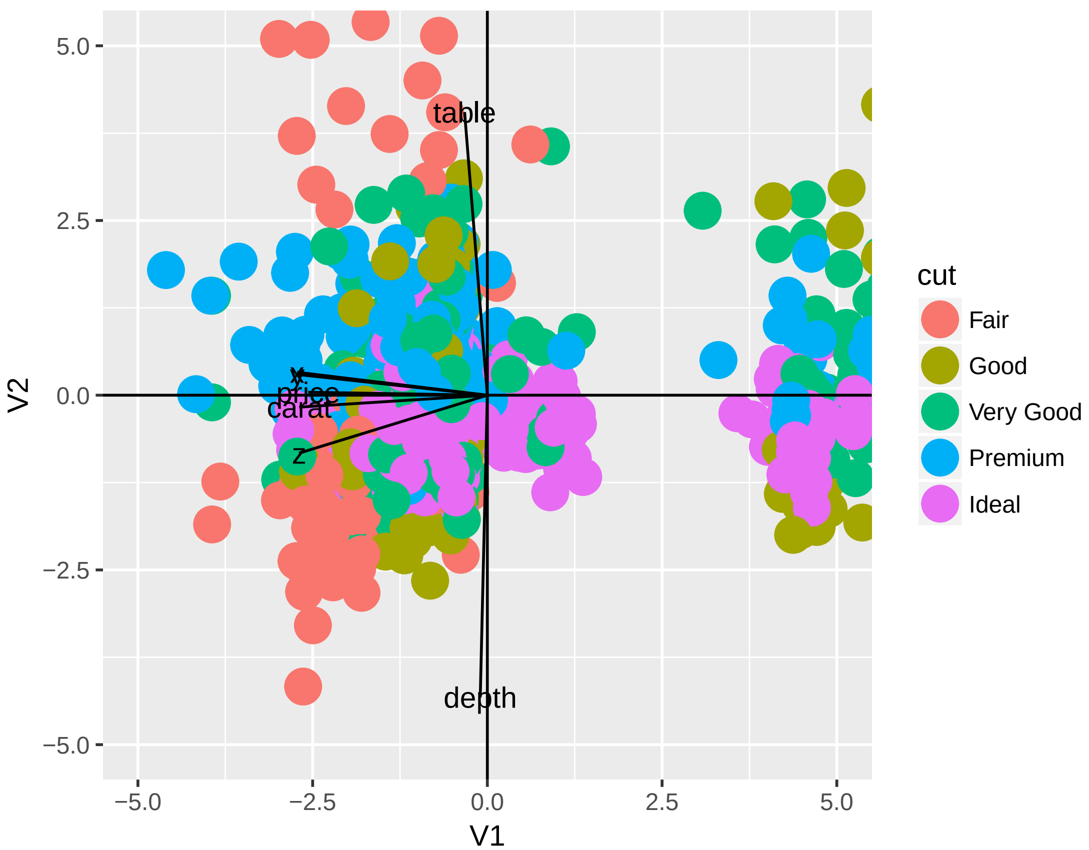
```r
cov(PCs)#did it work? If so each PC score should be uncorrelated with all the others.
```

```
##              [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  4.788025e+00  1.088274e-16  7.547207e-16  1.975972e-15  1.247338e-15
## [2,]  1.088274e-16  1.375532e+00 -5.201193e-15 -6.571450e-16  5.036329e-17
## [3,]  7.547207e-16 -5.201193e-15  6.721095e-01  9.102421e-16  5.596977e-16
## [4,]  1.975972e-15 -6.571450e-16  9.102421e-16  1.531486e-01 -3.104636e-15
## [5,]  1.247338e-15  5.036329e-17  5.596977e-16 -3.104636e-15  7.352490e-03
```

```
## [6,] -2.902639e-16 -9.188373e-17 -2.114764e-16  1.681546e-15  1.015341e-15
## [7,] -1.212482e-15  1.872705e-16 -1.862541e-16  1.231895e-15  6.328519e-16
##               [,6]          [,7]
## [1,] -2.902639e-16 -1.212482e-15
## [2,] -9.188373e-17  1.872705e-16
## [3,] -2.114764e-16 -1.862541e-16
## [4,]  1.681546e-15  1.231895e-15
## [5,]  1.015341e-15  6.328519e-16
## [6,]  3.290414e-03  7.793012e-16
## [7,]  7.793012e-16  5.419975e-04
```

OK, great. How can we explore these data?

```
df = cbind(as.data.frame(PCs),diamonds[1:nrow(X),]) #let's make a big data frame with the original data
df2 = cbind(as.data.frame(out$v),names(diamonds[,-c(2,3,4)])) #and a second with just the eigenvectors
names(df2)[8]="names"
arrowScale=6 #lets setup the length of our arrows as a variable
ggplot(df, aes(x = V1, y= V2))+#and make an awesome plot, we're going to compare PC1 and 2
  geom_point(aes(colour=cut),size=6)+ #first just plotting points and colouring them by the cut
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+#then we'll plot some 0 lines for our reference
  coord_fixed(xlim=c(-5,5),ylim=c(-5,5))+#and set the scale
  geom_segment(data=df2,aes(x=0,y=0,xend = V1*arrowScale, yend = V2*arrowScale))+#Now we'll add lines t
  geom_text(data=df2,aes(x = V1*arrowScale, y = V2*arrowScale,label=names) )#and label those lines
```



More on Wednesday!