

4051__HW6

Tae Uk You

4/5/2019

2. UK Foods Problem

(a)

```
UKFood_data= read.csv("/Users/Tae/Desktop/UK_Foods.csv",head=TRUE)

## Warning in read.table(file = file, header = header, sep = sep, quote =
## quote, : incomplete final line found by readTableHeader on '/Users/Tae/
## Desktop/UK_Foods.csv'

attach(UKFood_data)
#str(UKFood_data)

rownames(UKFood_data) = UKFood_data[,1]
UKFood_data = UKFood_data[,-1]

# a) Decision Covariance vs Correaltion
# cov(UKFood_data)
# cor(UKFood_data)
```

Answer: Since the scales of variable is quite different, I decide to use correlation matrix

b) principal Component Analysis and Plotting

```
UKFood_pca_corr = prcomp(UKFood_data,scale=TRUE)
summary(UKFood_pca_corr)

## Importance of components:
##              PC1      PC2      PC3      PC4
## Standard deviation    3.4082 2.0562 1.07524 6.344e-16
## Proportion of Variance 0.6833 0.2487 0.06801 0.000e+00
## Cumulative Proportion 0.6833 0.9320 1.00000 1.000e+00

UKFood_pca_corr

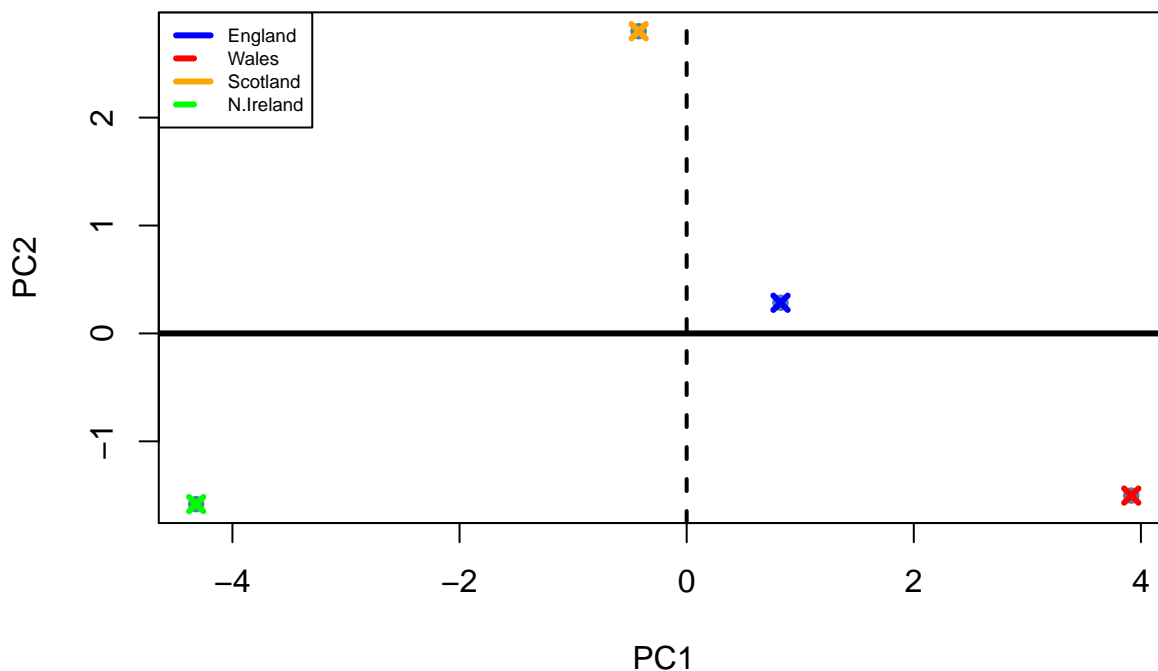
## Standard deviations (1, .., p=4):
## [1] 3.408187e+00 2.056239e+00 1.075241e+00 6.343501e-16
##
## Rotation (n x k) = (17 x 4):
##              PC1      PC2      PC3      PC4
## Cheese      0.24572131 0.24708041 -0.18723888 0.061643943
## Carcass_meat -0.28562914 -0.07716993 -0.15327713 0.930709694
## Other_meat   0.26481055 0.13610704 0.30439545 0.228796673
## Fish         0.28611787 -0.01100602 -0.20499304 0.060921570
## Fats_and_oils 0.12719487 -0.40054501 0.34010234 0.066845358
## Sugars       0.28110120 -0.13684092 0.05073161 0.113149639
## Fresh_potatoes -0.09775901 -0.45468785 0.11338943 -0.051666391
## Fresh_Veg    0.26544997 -0.09648630 -0.35064519 0.058422723
## Other_Veg    0.28708628 -0.09282322 -0.07333507 -0.003341928
## Processed_potatoes 0.12073819 0.41036129 0.32037834 0.131607614
```

```
## Processed_Veg      0.25767815 -0.15396030 -0.33340494 -0.027285680
## Fresh_fruit        0.27890521  0.08174644 -0.24284020  0.054944959
## Cereals            0.17843996 -0.32902948  0.38616448  0.074721387
## Beverages          0.27747513 -0.13773847  0.14839777  0.045781255
## Soft_drinks        -0.22771961  0.29323981  0.17168328 -0.073280859
## Alcoholic_drinks   0.25509484  0.23231808  0.11736728  0.083115464
## Confectionery       0.25275834  0.21105701  0.24529742  0.097538190
```

```
#UKFood_paca_cov = prcomp(UKFood_data[,-1],scale=FALSE)
#summary(UKFood_paca_cov)
```

```
plot(PC2~PC1, pch=19, col="steelblue",
     main = "UK Food PCA Scores",
     data = UKFood_pca_corr$x)
abline(h=0, lwd=3)
abline(v=0, lty=2, lwd=2)
legend("topleft", cex=.6,
     c("England", "Wales", "Scotland", "N.Ireland"),
     col=c("blue", "red", "orange", "green"),
     lty=c(1,2), lwd=c(3,3))
points(0.8266124, 0.284332, pch=4, col="blue", lwd=3)
points(3.9152584, -1.502883, pch=4, col="red", lwd=3)
points(-0.4226016, 2.800442, pch=4, col="orange", lwd=3)
points(-4.3192692, -1.581891, pch=4, col="green", lwd=3)
```

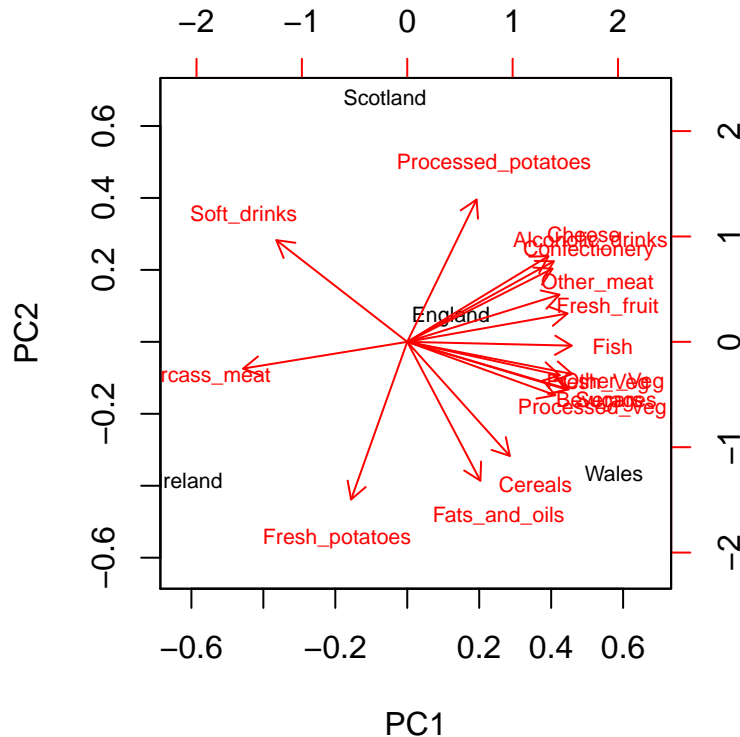
UK Food PCA Scores



Answer: Looking at the PCA score plot, the four countries seem to be different in terms PCA score as they are located far each other.

c) Create a biplot

```
biplot(UKFood_pca_corr,cex=.7,expand=.9)
```



```
#biplot(UKFood_pca_cov)
```

Answer: With variables of Processed_potatoes and Soft_drinks, Scotland is separated from other countries; N.Ireland is separated by Carcass_meat and Fresh_potatoes; Wales is separated by Cereals, Processed_veg and Beverages; and Englad is separated by Alcohol, Chesse, Confectionary, and Other_meat.

3. Cars Problem

a) sample correlation matrix

```
car_data_temp= read.csv("/Users/Tae/Desktop/cars2.csv",head=TRUE)
car_data = car_data_temp[,-c(1:2)]
attach(car_data_temp)
```

```
## The following object is masked from UKFood_data:
```

```
##
```

```
## X
```

```
cor(car_data)
```

```
## Warning in cor(car_data): the standard deviation is zero
```

```
##           sports.car      suv      wagon      minivan pickup
## sports.car      1.00000000 -0.15384461 -0.103240640 -0.08467889    NA
## suv             -0.15384461  1.00000000 -0.120710923 -0.09900817    NA
## wagon           -0.10324064 -0.12071092  1.000000000 -0.06644150    NA
## minivan         -0.08467889 -0.09900817 -0.066441503  1.00000000    NA
## pickup          NA          NA          NA          NA          1
## AWD             -0.08177767  0.44992770  0.077187791 -0.02999842    NA
```

## RWD	0.39605332	-0.24022555	-0.001005335	-0.10500598	NA
## retail.price	0.38970707	0.03112724	-0.057784001	-0.06440341	NA
## invoice.price	0.38379870	0.02614692	-0.054733905	-0.06639972	NA
## engine.size	0.09045504	0.31140988	-0.096828198	0.09393251	NA
## num.cylinders	0.11336455	0.21412366	-0.078844343	0.02241396	NA
## horse.power	0.34450331	0.12157456	-0.078166597	-0.03324887	NA
## city.MPG	-0.11667460	-0.33161752	0.035355603	-0.10716720	NA
## hiway.MPG	-0.09575039	-0.50005204	0.026775940	-0.11668890	NA
## weight	-0.12219563	0.52057274	-0.031976079	0.21889308	NA
## wheel.base.length	-0.39468846	0.20567718	-0.051509004	0.33604894	NA
## length	-0.33305523	0.08062675	-0.053355931	0.21521918	NA
## width	-0.05856826	0.33482537	-0.075907073	0.33819843	NA
##	AWD	RWD	retail.price	invoice.price	
## sports.car	-0.08177767	0.396053322	0.38970707	0.38379870	
## suv	0.44992770	-0.240225552	0.03112724	0.02614692	
## wagon	0.07718779	-0.001005335	-0.05778400	-0.05473390	
## minivan	-0.02999842	-0.105005982	-0.06440341	-0.06639972	
## pickup	NA	NA	NA	NA	
## AWD	1.00000000	-0.284575939	0.10750507	0.10348428	
## RWD	-0.28457594	1.000000000	0.46957483	0.46955967	
## retail.price	0.10750507	0.469574832	1.00000000	0.99912697	
## invoice.price	0.10348428	0.469559670	0.99912697	1.00000000	
## engine.size	0.16000552	0.313030284	0.59941603	0.59362503	
## num.cylinders	0.11662349	0.400149747	0.65435396	0.65087835	
## horse.power	0.13083566	0.429881375	0.83507354	0.83265013	
## city.MPG	-0.28481964	-0.222874365	-0.48521321	-0.48197957	
## hiway.MPG	-0.37942298	-0.181746884	-0.46929101	-0.46587930	
## weight	0.38289569	0.070426724	0.47599389	0.47196255	
## wheel.base.length	0.06231875	0.052005248	0.20335410	0.20336073	
## length	-0.01606648	0.016245049	0.20955305	0.20687806	
## width	0.15784389	0.080623547	0.31300271	0.30578557	
##	engine.size	num.cylinders	horse.power	city.MPG	
## sports.car	0.09045504	0.11336455	0.34450331	-0.1166746	
## suv	0.31140988	0.21412366	0.12157456	-0.3316175	
## wagon	-0.09682820	-0.07884434	-0.07816660	0.0353556	
## minivan	0.09393251	0.02241396	-0.03324887	-0.1071672	
## pickup	NA	NA	NA	NA	
## AWD	0.16000552	0.11662349	0.13083566	-0.2848196	
## RWD	0.31303028	0.40014975	0.42988137	-0.2228744	
## retail.price	0.59941603	0.65435396	0.83507354	-0.4852132	
## invoice.price	0.59362503	0.65087835	0.83265013	-0.4819796	
## engine.size	1.00000000	0.91144092	0.77861667	-0.7053772	
## num.cylinders	0.91144092	1.00000000	0.79114362	-0.6705137	
## horse.power	0.77861667	0.79114362	1.00000000	-0.6718006	
## city.MPG	-0.70537718	-0.67051368	-0.67180055	1.0000000	
## hiway.MPG	-0.70740807	-0.66418858	-0.65046797	0.9411187	
## weight	0.81164046	0.73118102	0.63105810	-0.7361656	
## wheel.base.length	0.63146985	0.55269251	0.39662596	-0.4811264	
## length	0.62458110	0.54636341	0.38142565	-0.4675862	
## width	0.72609915	0.61883429	0.49944686	-0.5868873	
##	hiway.MPG	weight	wheel.base.length	length	
## sports.car	-0.09575039	-0.12219563	-0.39468846	-0.33305523	
## suv	-0.50005204	0.52057274	0.20567718	0.08062675	
## wagon	0.02677594	-0.03197608	-0.05150900	-0.05335593	

```
## minivan      -0.11668890  0.21889308      0.33604894  0.21521918
## pickup      NA      NA      NA      NA
## AWD      -0.37942298  0.38289569      0.06231875 -0.01606648
## RWD      -0.18174688  0.07042672      0.05200525  0.01624505
## retail.price -0.46929101  0.47599389      0.20335410  0.20955305
## invoice.price -0.46587930  0.47196255      0.20336073  0.20687806
## engine.size  -0.70740807  0.81164046      0.63146985  0.62458110
## num.cylinders -0.66418858  0.73118102      0.55269251  0.54636341
## horse.power  -0.65046797  0.63105810      0.39662596  0.38142565
## city.MPG      0.94111867 -0.73616563     -0.48112635 -0.46758621
## hiway.MPG      1.00000000 -0.78860686     -0.45460932 -0.38910914
## weight      -0.78860686  1.00000000      0.75091534  0.65289325
## wheel.base.length -0.45460932  0.75091534      1.00000000  0.86652887
## length      -0.38910914  0.65289325      0.86652887  1.00000000
## width      -0.58274059  0.80662748      0.75683864  0.75135738
##      width
## sports.car  -0.05856826
## suv      0.33482537
## wagon      -0.07590707
## minivan      0.33819843
## pickup      NA
## AWD      0.15784389
## RWD      0.08062355
## retail.price  0.31300271
## invoice.price  0.30578557
## engine.size  0.72609915
## num.cylinders  0.61883429
## horse.power  0.49944686
## city.MPG     -0.58688727
## hiway.MPG     -0.58274059
## weight      0.80662748
## wheel.base.length  0.75683864
## length      0.75135738
## width      1.00000000
```

b) orthogonality check

```
car_data_corr = prcomp(car_data[, -c(1:7)], scale=TRUE)
#car_data_corr
#summary(car_data_corr)
car_orthogonal = t(car_data_corr$rotation[,1])*(car_data_corr$rotation[,2])
sum(car_orthogonal)
```

```
## [1] 4.293441e-17
```

Answer: It is checked that the first principal component is orthogonal to the second principal component. The value is extremely close to 0.

$$c) y_1 = 0.263x_{\text{retail.price}} + 0.262x_{\text{invoice.price}} + 0.347x_{\text{engine.size}} + 0.334x_{\text{num.cylinder}} + 0.319x_{\text{horse.power}} - 0.310x_{\text{city.MPG}} - 0.307x_{\text{hiway.MPG}} + 0.336x_{\text{weight}} + 0.266x_{\text{wheel.base.length}} + 0.257x_{\text{length}} + 0.296x_{\text{width}}$$

d) Screeplot

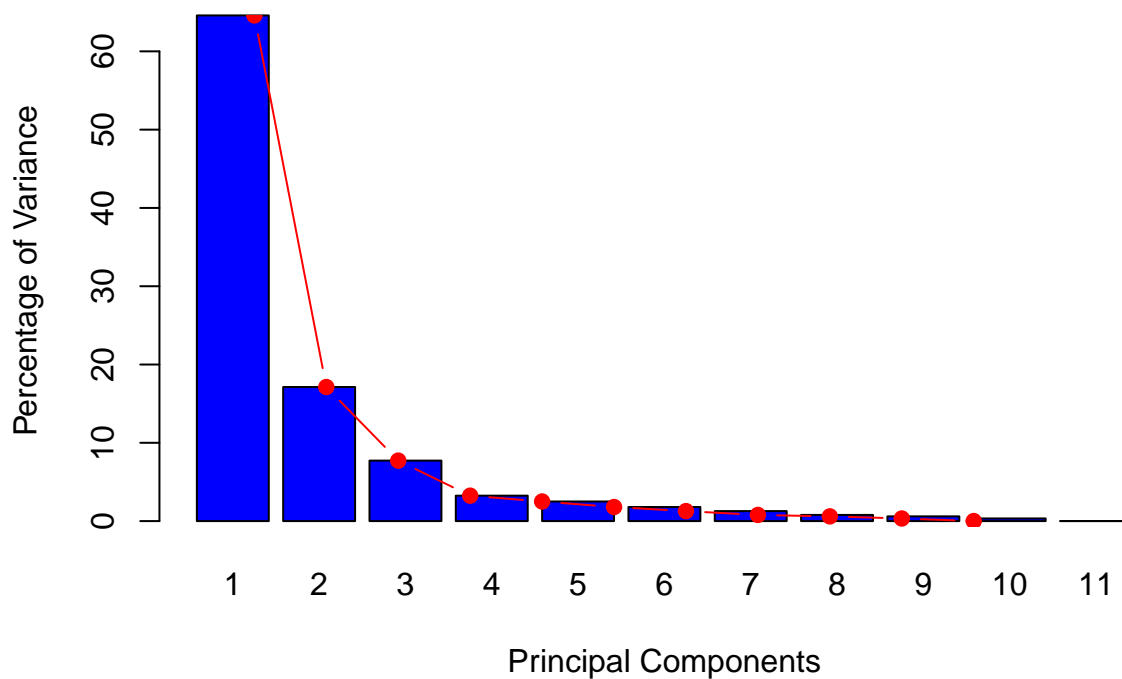
```
# Obtain Eigenvalues
car_eigenvalue = (car_data_corr$sd)^2
# Variances in percentage
car_variance = car_eigenvalue*100/sum(car_eigenvalue)
```

```
# Cumulative Variance
car_cumvar = cumsum(car_variance)
# Create new data frame with these results
car_eigen.pca_summary = data.frame(car_eigenvalue,car_variance,car_cumvar)
car_eigen.pca_summary
```

```
##      car_eigenvalue car_variance car_cumvar
## 1      7.1046384308 64.587622098   64.58762
## 2      1.8839247679 17.126588799   81.71421
## 3      0.8497282852  7.724802592   89.43901
## 4      0.3570154894  3.245595359   92.68461
## 5      0.2754355932  2.503959939   95.18857
## 6      0.1979437155  1.799488322   96.98806
## 7      0.1405192086  1.277447350   98.26550
## 8      0.0866388119  0.787625563   99.05313
## 9      0.0663879807  0.603527097   99.65666
## 10     0.0369773622  0.336157838   99.99281
## 11     0.0007903547  0.007185043  100.00000
```

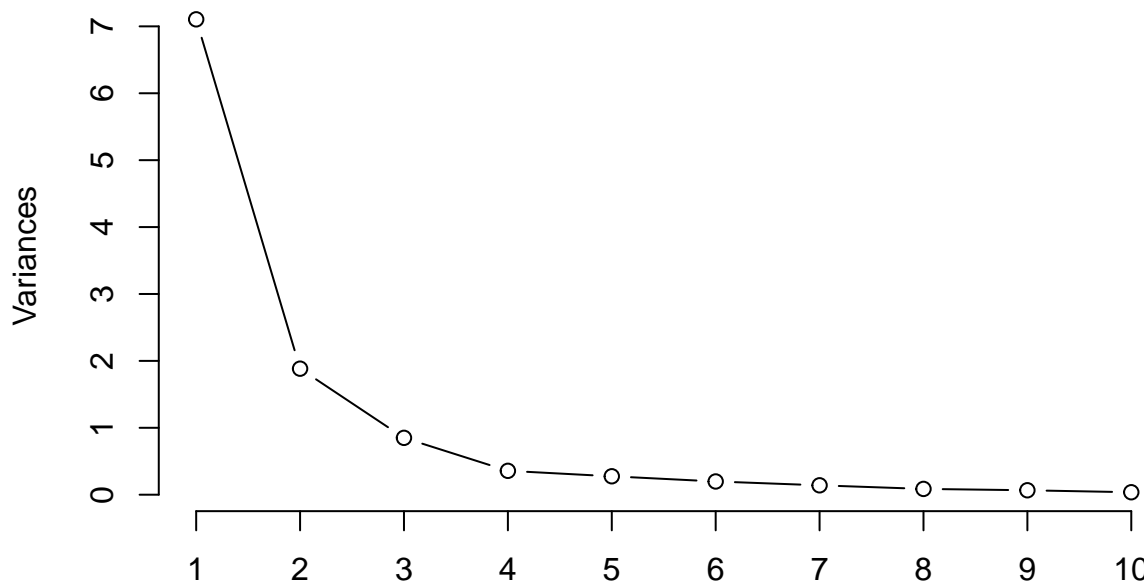
```
# Scree Plot
barplot(car_variance,names.arg=1:nrow(car_eigen.pca_summary),
        xlab="Principal Components",
        ylab="Percentage of Variance",
        col="blue", main="Scree Plot")
lines(x=1:nrow(car_eigen.pca_summary),car_eigen.pca_summary[,2],
      type='b',pch=19,col="red")
```

Scree Plot



```
# Scree Plot 2
screepplot(car_data_corr,type="lines",main="Scree Plot \n Plot 2")
```

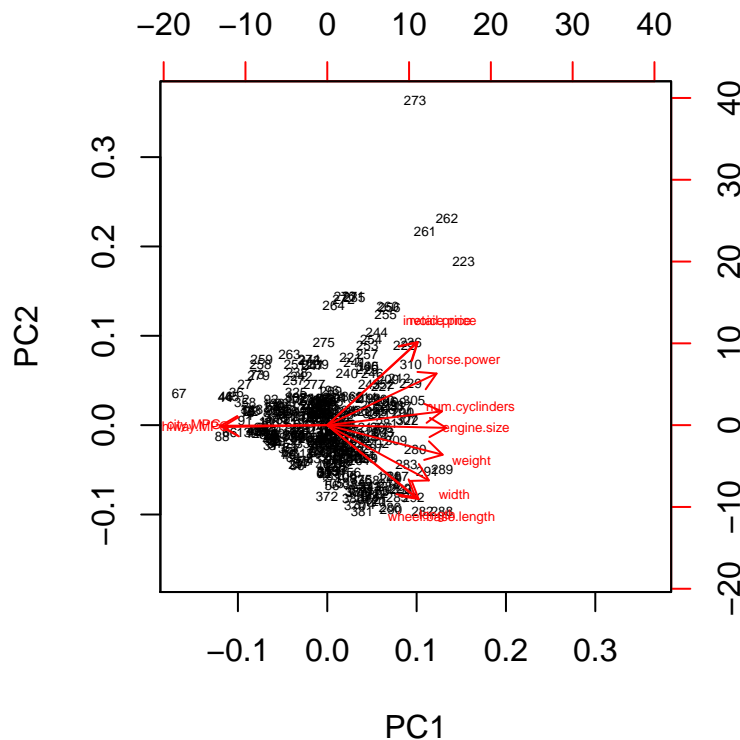
Scree Plot Plot 2



The first eigenvector appears to be a contrast of (city.MGP and hiway.MPG) vs (others). Lookin at the scree plot, the straight line starts to occur around the fourth principal component, so we should retain 3 components. Also, when looking at proportion of variance from the summary, up to 3 principal components already has about 90% variability explained from total variability.

e) Interpret the retained components.

```
biplot(car_data_corr,cex=.4,expand=0.9)
```



Answer: We select city.MGP and hiway.MPG for the first principal components that separate the data. For the second principal components, a contrast of

(retail.price and invoice.price) versus (wheel.base.length and length) creates a separation line for objects.

4. Swiss Bank Notes

a) and b) and c)

```
library(psych)

## Warning: package 'psych' was built under R version 3.5.2
Swiss_data_temp= read.csv("/Users/Tae/Desktop/Swiss.csv",head=TRUE)
attach(Swiss_data_temp)

## The following object is masked from car_data_temp:
##
##      length
Swiss_data = Swiss_data_temp[,-1]

# Trace
#cov(Swiss_data)
tr(cov(Swiss_data)) # 4.494724

## [1] 4.494724

# Eigenvalues
Swiss_pca_cov = prcomp(Swiss_data,scale=FALSE)
sum(Swiss_pca_cov$sd^2) # 4.494724

## [1] 4.494724

# principal Components
Swiss_pca_cov

## Standard deviations (1, .., p=6):
## [1] 1.7321388 0.9672748 0.4933697 0.4412015 0.2919107 0.1884534
##
## Rotation (n x k) = (6 x 6):
##
##           PC1      PC2      PC3      PC4      PC5
## length      0.04377427 -0.01070966 0.3263165 -0.5616918 -0.75257278
## left_width  -0.11216159 -0.07144697 0.2589614 -0.4554588 0.34680082
## right_width -0.13919062 -0.06628208 0.3447327 -0.4153296 0.53465173
## bottom_margin -0.76830499 0.56307225 0.2180222 0.1861082 -0.09996771
## top_margin  -0.20176610 -0.65928988 0.5566857 0.4506985 -0.10190229
## diag_length 0.57890193 0.48854255 0.5917628 0.2584483 0.08445895
##
##           PC6
## length      0.09809807
## left_width  -0.76651197
## right_width 0.63169678
## bottom_margin -0.02221711
## top_margin  -0.03485874
## diag_length -0.04567946

summary(Swiss_pca_cov)

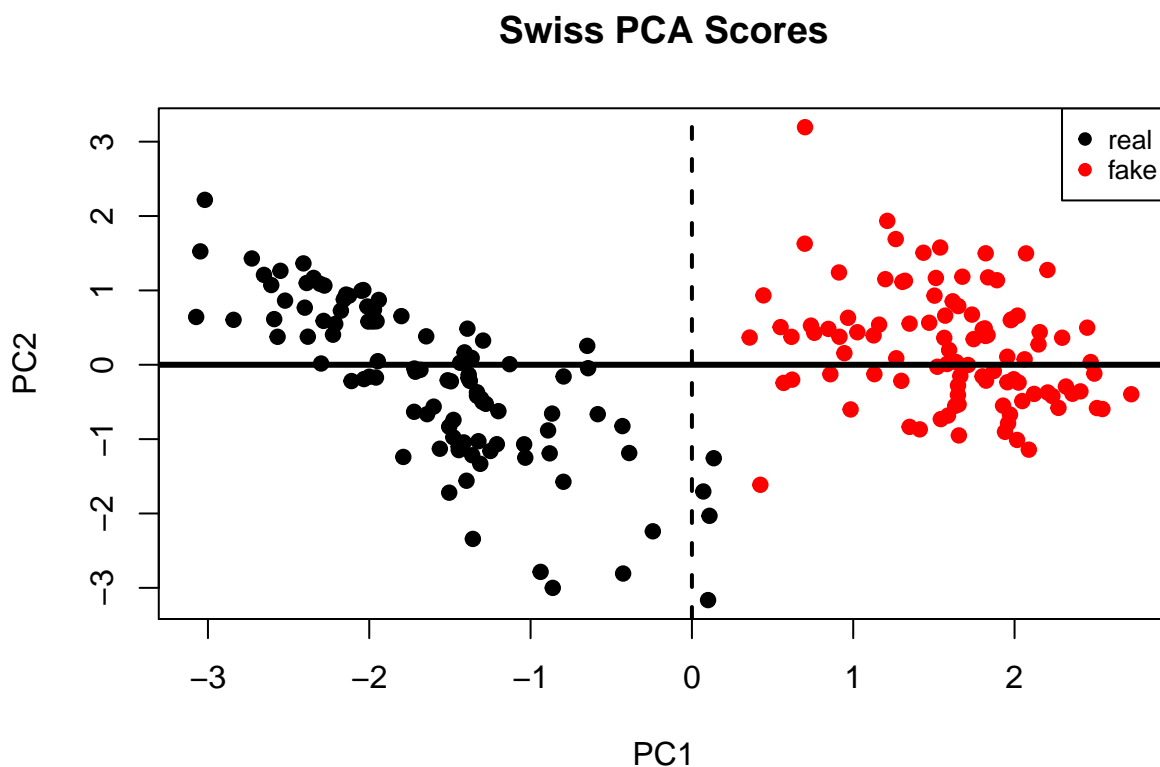
## Importance of components:
##
##           PC1      PC2      PC3      PC4      PC5      PC6
```

```
## Standard deviation      1.7321 0.9673 0.49337 0.44120 0.29191 0.1885
## Proportion of Variance 0.6675 0.2082 0.05416 0.04331 0.01896 0.0079
## Cumulative Proportion  0.6675 0.8757 0.92983 0.97314 0.99210 1.0000
```

Answer: Proved that sum of eigenvalues is equal to the trace of covariance matrix. From the principal components analysis, the first eigenvectors appear to be dominated by `diag_length` and `bottom_margin` followed by `length`. The second principal components explains where it seems to be a contrast of (`bottom_margin`, `top_margin` and `diag_length`) vs (others).

d)

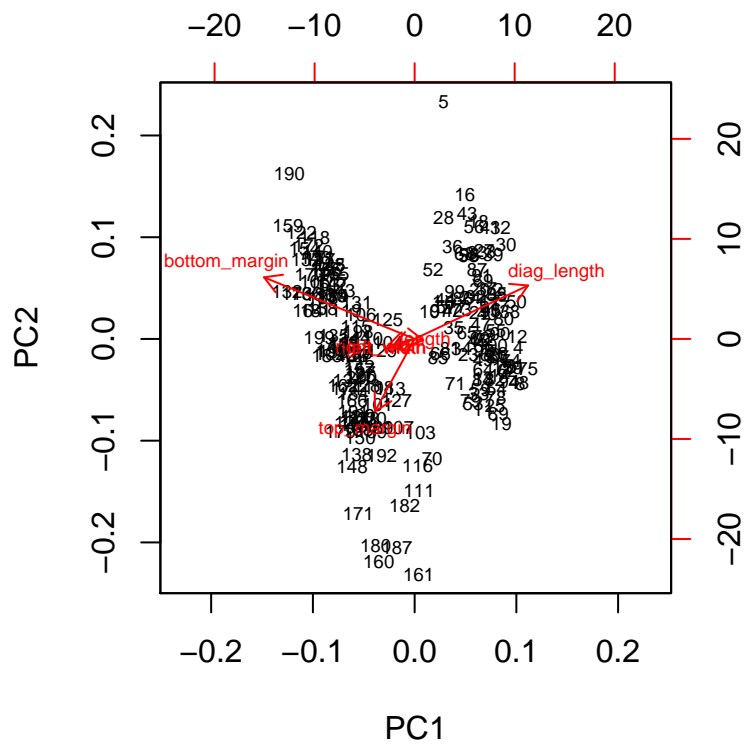
```
plot(PC2~PC1, pch=19, col=note,
     main = "Swiss PCA Scores",
     data = Swiss_pca_cov$x)
abline(h=0, lwd=3)
abline(v=0, lty=2, lwd=2)
legend("topright", cex=.8, legend=c("real", "fake"), pch=19, col=c("black", "red"))
```



Answer: The real and fake bank notes are well separated based on the correlation matrix. Based on the plot of the first two principal components from the correlation matrix, the first principal components separate real and fake bank notes, where the first eigenvectors showed `bottom_margin` and `diag_length` explained the most variability.

e) biplot

```
biplot(Swiss_pca_cov, cex=.6, expand=0.8)
```

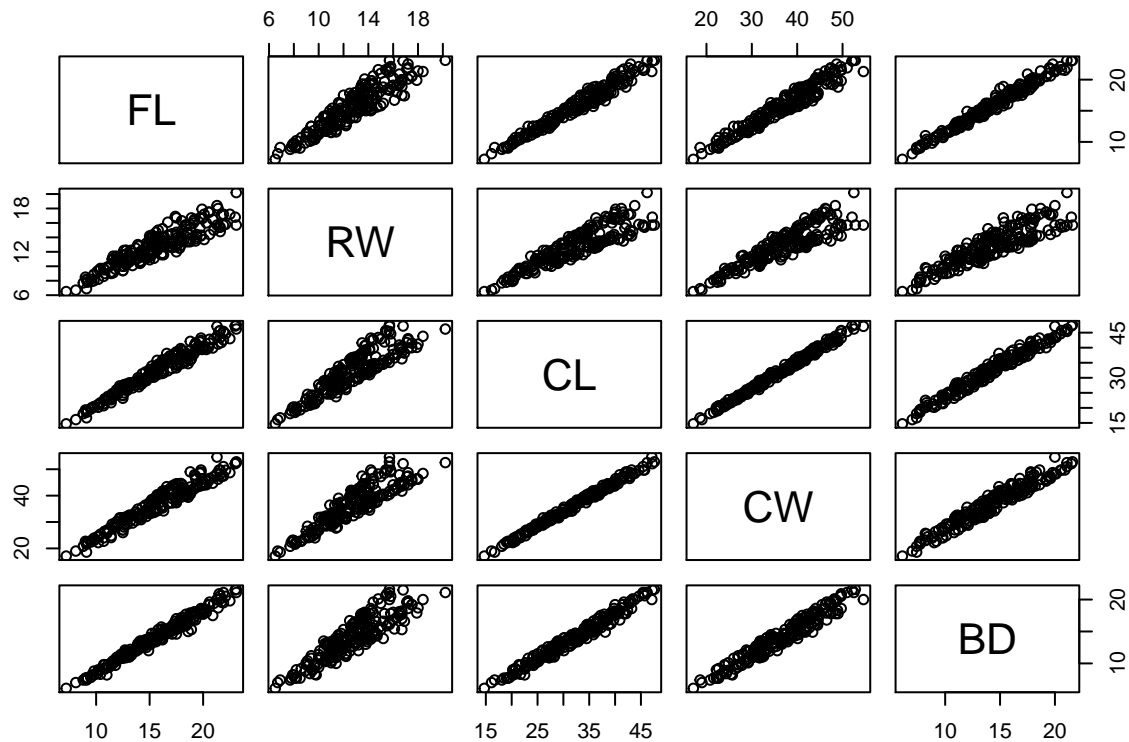


Answer: The biplot indicates that the variable `bottom_margin` and `top margin` separate fake from real, whereas `diag_length` separates real from fake.

5. Crabs

```
library(MASS)
data(crabs)
attach(crabs)

# Look at correlation of variables at glance
plot(crabs[,c(1:3)])
```



```
# Examine covariance and correlation matrix
cov(crabs[, -c(1:3)])
```

```
##           FL           RW           CL           CW           BD
## FL 12.217297  8.158045 24.35668 26.55080 11.822581
## RW  8.158045  6.622078 16.35466 18.23964  7.836659
## CL 24.356677 16.354662 50.67992 55.76138 23.971389
## CW 26.550801 18.239640 55.76138 61.96768 26.091867
## BD 11.822581  7.836659 23.97139 26.09187 11.729065
```

```
cor(crabs[, -c(1:3)])
```

```
##           FL           RW           CL           CW           BD
## FL 1.0000000 0.9069876 0.9788418 0.9649558 0.9876272
## RW 0.9069876 1.0000000 0.8927430 0.9004021 0.8892054
## CL 0.9788418 0.8927430 1.0000000 0.9950225 0.9832038
## CW 0.9649558 0.9004021 0.9950225 1.0000000 0.9678117
## BD 0.9876272 0.8892054 0.9832038 0.9678117 1.0000000
```

```
# Result: From the correlation plot and matrix, we can see that there are strong relationships
# between variables. Because all numeric predictors have same units(mm) and similar range of scale,
# I decided to choose covariance matrix to build up principal components.
```

```
# principal Component Analysis - covariance matrix
crabs_pca_cov = prcomp(crabs[, -c(1:3)], scale=FALSE)
summary(crabs_pca_cov)
```

```
## Importance of components:
```

```
##           PC1           PC2           PC3           PC4           PC5
## Standard deviation 11.8619 1.13879 1.00013 0.36783 0.27913
## Proportion of Variance 0.9825 0.00906 0.00698 0.00094 0.00054
## Cumulative Proportion 0.9825 0.99153 0.99851 0.99946 1.00000
```

```
crabs_pca_cov
```

```
## Standard deviations (1, ..., p=5):  
## [1] 11.8619441  1.1387874  1.0001346  0.3678306  0.2791312  
##  
## Rotation (n x k) = (5 x 5):  
##          PC1          PC2          PC3          PC4          PC5  
## FL 0.2889810  0.3232500 -0.5071698  0.7342907  0.1248816  
## RW 0.1972824  0.8647159  0.4141356 -0.1483092 -0.1408623  
## CL 0.5993986 -0.1982263 -0.1753299 -0.1435941 -0.7416656  
## CW 0.6616550 -0.2879790  0.4913755  0.1256282  0.4712202  
## BD 0.2837317  0.1598447 -0.5468821 -0.6343657  0.4386868
```

```
# principal Component Analysis - correlation matrix  
crabs_pca_corr = prcomp(crabs[, -c(1:3)], scale=TRUE)  
# summary(crabs_pca_corr)  
crabs_pca_corr
```

```
## Standard deviations (1, ..., p=5):  
## [1] 2.18834065 0.38946785 0.21594669 0.10552420 0.04137243  
##  
## Rotation (n x k) = (5 x 5):  
##          PC1          PC2          PC3          PC4          PC5  
## FL 0.4520437  0.1375813  0.53076841  0.696923372  0.09649156  
## RW 0.4280774 -0.8981307 -0.01197915 -0.083703203 -0.05441759  
## CL 0.4531910  0.2682381 -0.30968155 -0.001444633 -0.79168267  
## CW 0.4511127  0.1805959 -0.65256956  0.089187816  0.57452672  
## BD 0.4511336  0.2643219  0.44316103 -0.706636423  0.17574331
```

```
# When covariance matrix is used:  
# Obtain Eigenvalues  
crabs_eigenvalue = (crabs_pca_cov$sd)^2  
# Variances in percentage  
crabs_variance = crabs_eigenvalue*100/sum(crabs_eigenvalue)  
# Cumulative Variance  
crabs_cumvar = cumsum(crabs_variance)  
# Create new data frame with these results  
crabs_eigen_pca_cov_summary = data.frame(crabs_eigenvalue, crabs_variance, crabs_cumvar)  
crabs_eigen_pca_cov_summary
```

```
##   crabs_eigenvalue crabs_variance crabs_cumvar  
## 1    140.70571876    98.24717995    98.24718  
## 2     1.29683676     0.90551084    99.15269  
## 3     1.00026913     0.69843374    99.85112  
## 4     0.13529932     0.09447218    99.94560  
## 5     0.07791423     0.05440328   100.00000
```

```
# When correlation matrix is used:  
# Obtain Eigenvalues  
crabs_eigenvalue2 = (crabs_pca_corr$sd)^2  
# Variances in percentage  
crabs_variance2 = crabs_eigenvalue2*100/sum(crabs_eigenvalue2)  
# Cumulative Variance  
crabs_cumvar2 = cumsum(crabs_variance2)  
# Create new data frame with these results  
crabs_eigen_pca_corr_summary = data.frame(crabs_eigenvalue2, crabs_variance2, crabs_cumvar2)
```

```
crabs_eigen_pca_corr_summary
```

```
##   crabs_eigenvalue2 crabs_variance2 crabs_cumvar2
## 1      4.788834784      95.77669569      95.77670
## 2      0.151685207       3.03370413      98.81040
## 3      0.046632974       0.93265948      99.74306
## 4      0.011135357       0.22270714      99.96577
## 5      0.001711678       0.03423355     100.00000
```

```
# Orthogonality check
```

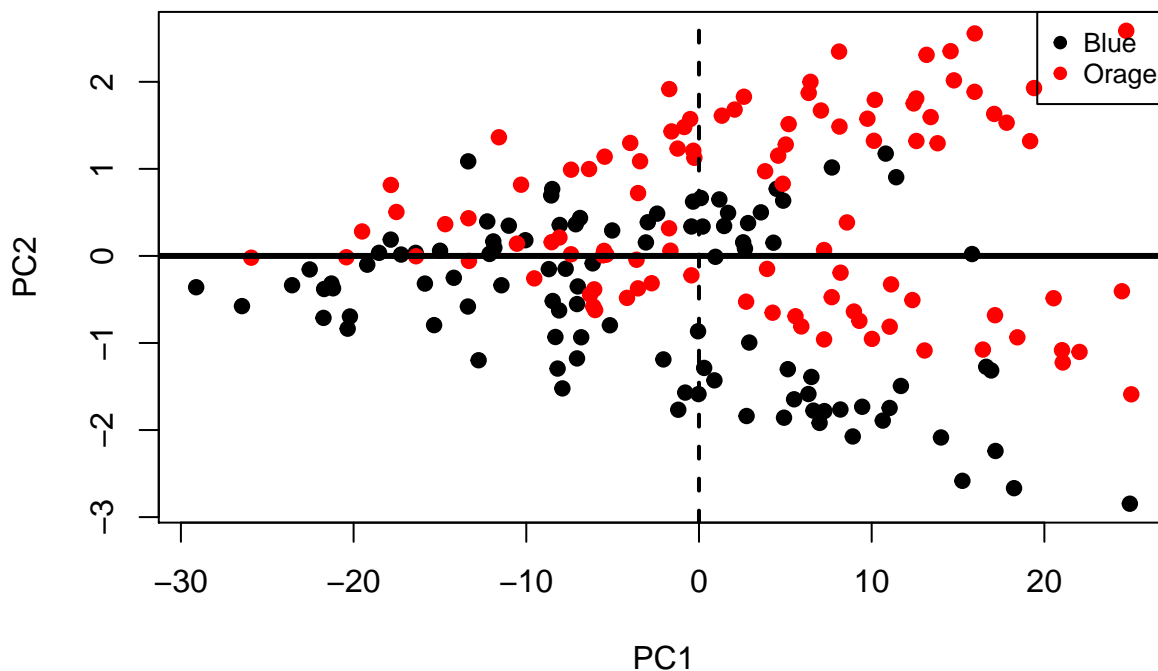
```
crab_orthogonal = t(crabs_pca_cov$rotation[,1])*(crabs_pca_cov$rotation[,2])
sum(crab_orthogonal) #9.020562e-17 (extremely close to 0)
```

```
## [1] 9.020562e-17
```

```
# PCA Score based on covariance
```

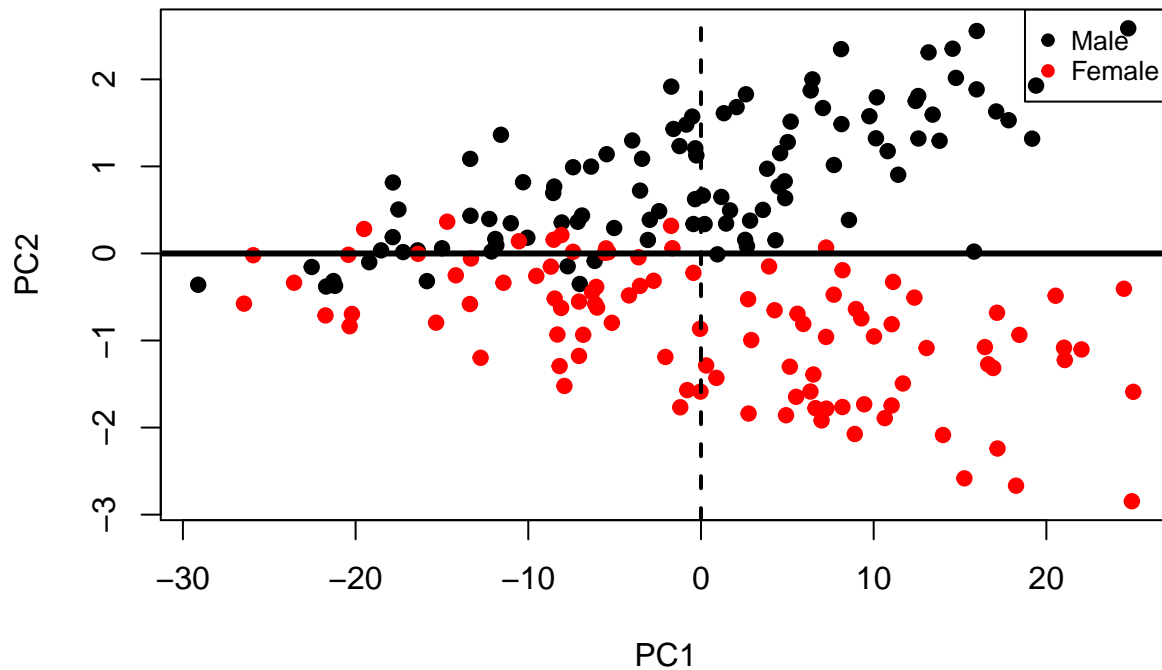
```
plot(PC2~PC1, pch=19, col=sp,
     main="Crab PCA Scores based on Covariance Per Color",
     data = crabs_pca_cov$x)
abline(h=0, lwd=3)
abline(v=0, lty=2, lwd=2)
legend("topright", cex=.8, legend=c("Blue", "Orange"), pch=19, col=c("black", "red"))
```

Crab PCA Scores based on Covariance Per Color



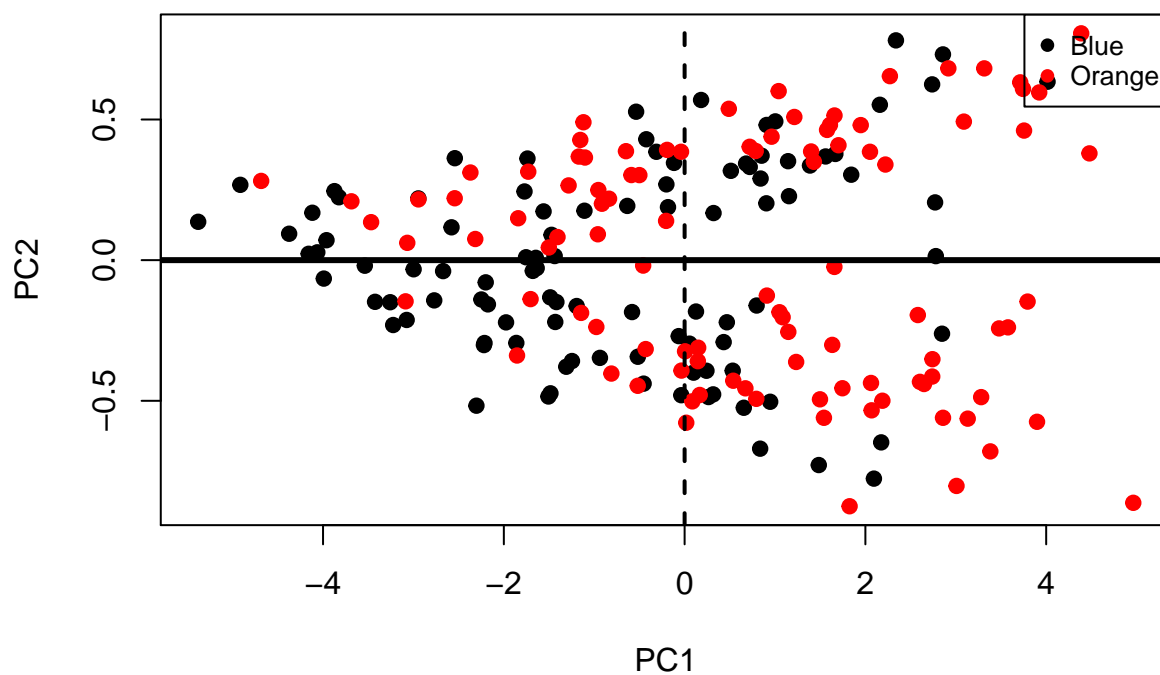
```
plot(PC2~PC1, pch=19, col=sex,
     main="Crab PCA Scores based on Covariance Per Sex",
     data = crabs_pca_cov$x)
abline(h=0, lwd=3)
abline(v=0, lty=2, lwd=2)
legend("topright", cex=.8, legend=c("Male", "Female"), pch=19, col=c("black", "red"))
```

Crab PCA Scores based on Covariance Per Sex



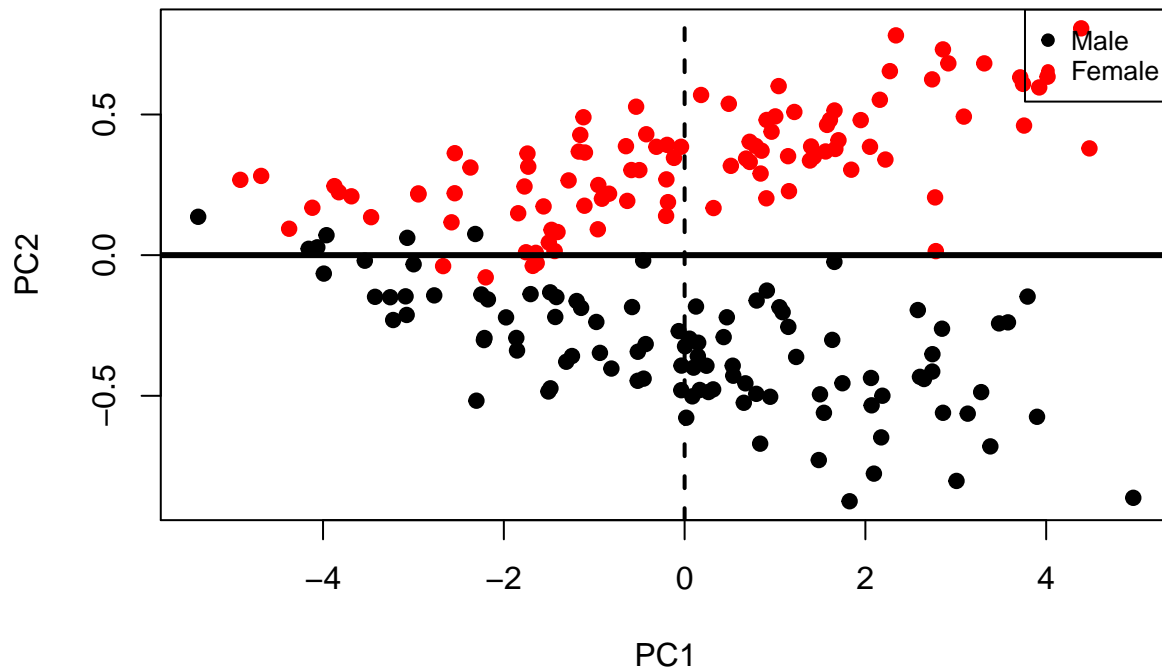
```
# PCA Score based on correlation
plot(PC2~PC1, pch=19, col=sp,
     main = "Crab PCA Scores based on Correlation Per Color",
     data = crabs_pca_corr$x)
abline(h=0, lwd=3)
abline(v=0, lty=2, lwd=2)
legend("topright", cex=.8, legend=c("Blue", "Orange"), pch=19, col=c("black", "red"))
```

Crab PCA Scores based on Correlation Per Color



```
plot(PC2~PC1, pch=19, col=sex,
     main="Crab PCA Scores based on Correlation Per Sex",
     data = crabs_pca_corr$x)
abline(h=0, lwd=3)
abline(v=0,lty=2,lwd=2)
legend("topright",cex=.8,legend=c("Male","Female"),pch=19,col=c("black","red"))
```

Crab PCA Scores based on Correlation Per Sex

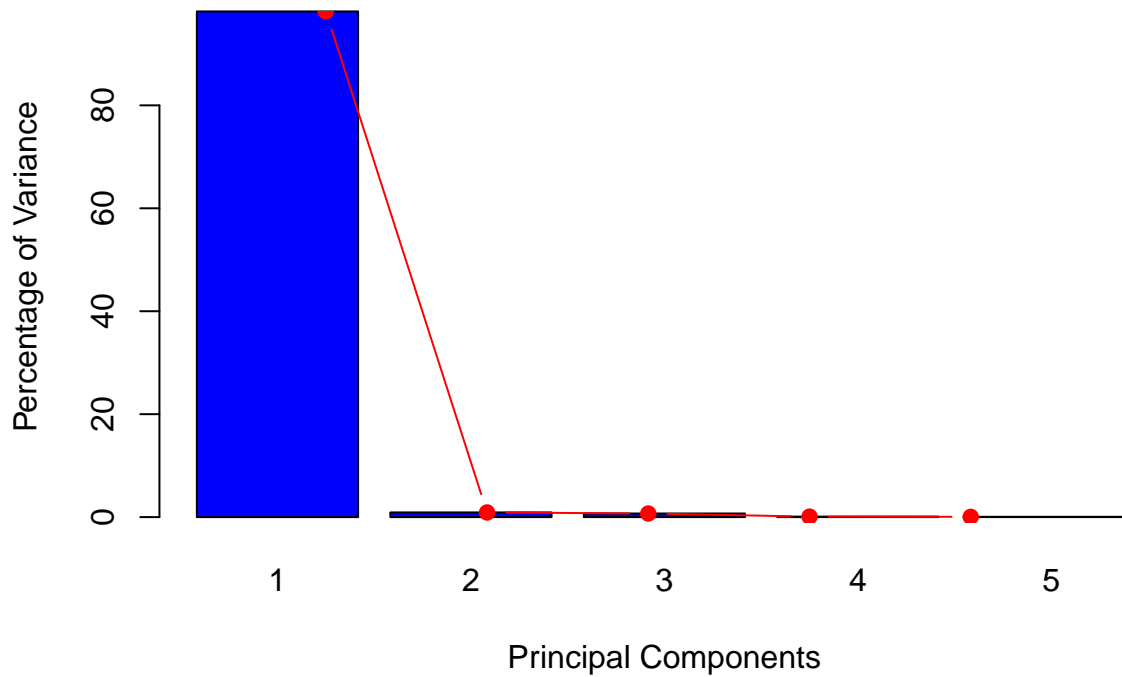


Answer: When the covariance matrix is used, the color of crab is separated quite well when first principal components involve, where it seems to be dominated by CL and CW. The second principal components separate data where RW holds a dominant variability. For separating male and female crab, the second principal components seem to provide a clear separation line by RW.

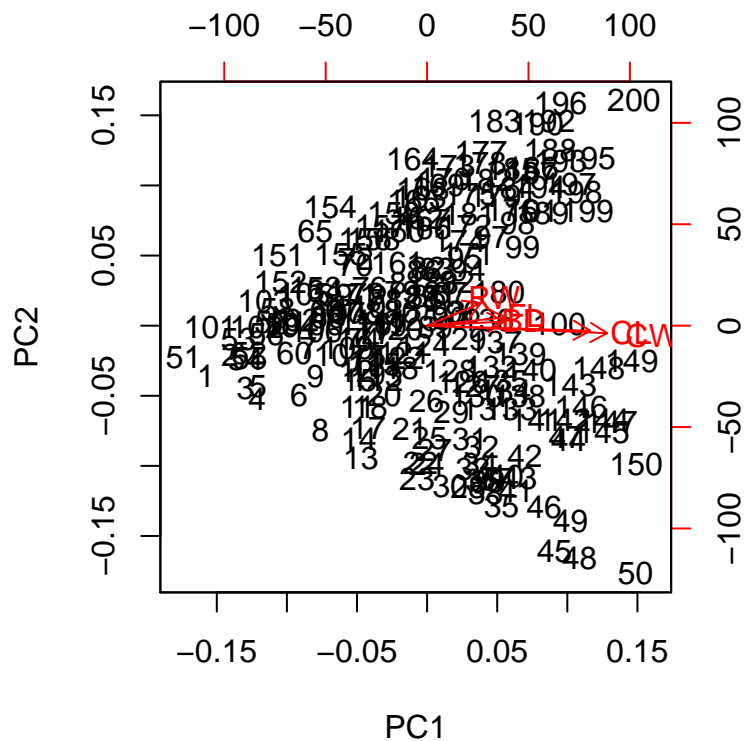
If the covariance matrix had been used, the first principal component appears to equally weigh all 5 variables. The second eigenvector is dominated by that characterizes the sex of crabs. To identify the color of crab, the third component needs to be involved when the covariance matrix is used.

```
# Scree Plot
barplot(crabs_variance, names.arg=1:nrow(crabs_eigen_pca_cov_summary),
        xlab="Principal Components",
        ylab="Percentage of Variance",
        col="blue", main="Scree Plot")
lines(x=1:nrow(crabs_eigen_pca_cov_summary), crabs_eigen_pca_cov_summary[,2],
      type='b', pch=19, col="red")
```

Scree Plot



```
# biplot
biplot(crabs_pca_cov)
```



Answer: From the summary of PCA based on covariance matrix, we know 98% variability is already explained, just liked it is also demonstrated by the scree plot. Thus, I decide to retain two components. Looking at the biplot, CL and CW clearly separates data so those two are my new pattern to be used when explaining this dataset.