Lab 06: Principal Components

PSTAT 131/231, Winter 2018

Learning Objectives

- Review of PCA
- `prcomp()` and `biplot()` functions
- Visualization with PCAs

Principal Components Analysis

Note: The material in this lab was obtained from James et. al. - An Introduction to Statistical Learning with Applications in R,pg. 402

In this lab, we perform PCA on the `USArrests` data set, which is part of the base R package. The rows of the data set contain the 50 states, in alphabetical order.

```
states=row.names(USArrests)
states
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
```

The columns of the data set contain the four variables.

```
names(USArrests)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

We first briefly examine the data. We notice that the variables have vastly different means.

```
summary(USArrests)
```

```
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

We see that there are on average three times as many rapes as murders, and more than eight times as many assaults as rapes. We can also examine the variances of the four variables.

```r
apply(USArrests , 2, var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

Not surprisingly, the variables also have vastly different variances: the UrbanPop variable measures the percentage of the population in each state living in an urban area, which is not a comparable number to the num ber of rapes in each state per 100,000 individuals. If we failed to scale the variables before performing PCA, then most of the principal components that we observed would be driven by the Assault variable, since it has by far the largest mean and variance. Thus, it is important to standardize the variables to have mean zero and standard deviation one before performing PCA.

We now perform principal components analysis using the `prcomp()` function, which is one of several functions in R that perform PCA.

```r
pr.out=prcomp(USArrests, scale=TRUE)
```

By default, the `prcomp()` function centers the variables to have mean zero. By using the option `scale=TRUE`, we scale the variables to have standard deviation one. The output from `prcomp()` contains a number of useful quantities.

```r
names(pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

The center and scale components correspond to the means and standard deviations of the variables that were used for scaling prior to implementing PCA.

```r
pr.out$center
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```r
pr.out$scale
```

```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

The rotation matrix provides the principal component loadings; each column of `pr.out$rotation` contains the corresponding principal component loading vector

```r
pr.out$rotation
```

```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

We see that there are four distinct principal components. This is to be expected because there are in general $min(n-1,p)$ informative principal components in a data set with n observations and p variables.

Using the `prcomp()` function, we do not need to explicitly multiply the data by the principal component loading vectors in order to obtain the principal component score vectors. Rather the 50×4 matrix x has as its columns the principal component score vectors. That is, the kth column is the kth principal component score vector.

```r
dim(pr.out$x)
```

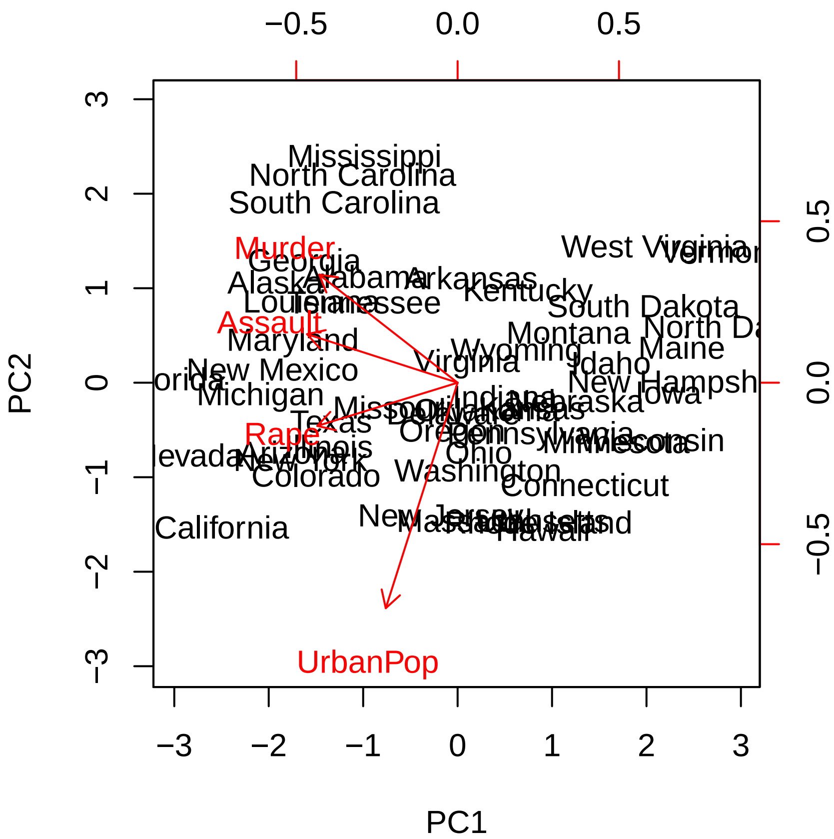
```
## [1] 50  4
```

```r
pr.out$x
```

```
##                        PC1         PC2         PC3          PC4
## Alabama        -0.97566045  1.12200121 -0.43980366  0.154696581
## Alaska         -1.93053788  1.06242692  2.01950027 -0.434175454
## Arizona        -1.74544285 -0.73845954  0.05423025 -0.826264240
## Arkansas        0.13999894  1.10854226  0.11342217 -0.180973554
## California     -2.49861285 -1.52742672  0.59254100 -0.338559240
## Colorado       -1.49934074 -0.97762966  1.08400162  0.001450164
## Connecticut     1.34499236 -1.07798362 -0.63679250 -0.117278736
## Delaware       -0.04722981 -0.32208890 -0.71141032 -0.873113315
## Florida        -2.98275967  0.03883425 -0.57103206 -0.095317042
## Georgia        -1.62280742  1.26608838 -0.33901818  1.065974459
## Hawaii          0.90348448 -1.55467609  0.05027151  0.893733198
## Idaho           1.62331903  0.20885253  0.25719021 -0.494087852
## Illinois       -1.36505197 -0.67498834 -0.67068647 -0.120794916
## Indiana         0.50038122 -0.15003926  0.22576277  0.420397595
## Iowa            2.23099579 -0.10300828  0.16291036  0.017379470
## Kansas          0.78887206 -0.26744941  0.02529648  0.204421034
## Kentucky        0.74331256  0.94880748 -0.02808429  0.663817237
## Louisiana      -1.54909076  0.86230011 -0.77560598  0.450157791
## Maine           2.37274014  0.37260865 -0.06502225 -0.327138529
## Maryland       -1.74564663  0.42335704 -0.15566968 -0.553450589
## Massachusetts   0.48128007 -1.45967706 -0.60337172 -0.177793902
## Michigan       -2.08725025 -0.15383500  0.38100046  0.101343128
## Minnesota       1.67566951 -0.62590670  0.15153200  0.066640316
## Mississippi    -0.98647919  2.36973712 -0.73336290  0.213342049
## Missouri       -0.68978426 -0.26070794  0.37365033  0.223554811
## Montana         1.17353751  0.53147851  0.24440796  0.122498555
## Nebraska        1.25291625 -0.19200440  0.17380930  0.015733156
## Nevada         -2.84550542 -0.76780502  1.15168793  0.311354436
## New Hampshire   2.35995585 -0.01790055  0.03648498 -0.032804291
## New Jersey     -0.17974128 -1.43493745 -0.75677041  0.240936580
## New Mexico     -1.96012351  0.14141308  0.18184598 -0.336121113
## New York       -1.66566662 -0.81491072 -0.63661186 -0.013348844
## North Carolina -1.11208808  2.20561081 -0.85489245 -0.944789648
## North Dakota    2.96215223  0.59309738  0.29824930 -0.251434626
## Ohio            0.22369436 -0.73477837 -0.03082616  0.469152817
## Oklahoma        0.30864928 -0.28496113 -0.01515592  0.010228476
## Oregon         -0.05852787 -0.53596999  0.93038718 -0.235390872
## Pennsylvania    0.87948680 -0.56536050 -0.39660218  0.355452378
## Rhode Island    0.85509072 -1.47698328 -1.35617705 -0.607402746
## South Carolina -1.30744986  1.91397297 -0.29751723 -0.130145378
## South Dakota    1.96779669  0.81506822  0.38538073 -0.108470512
## Tennessee      -0.98969377  0.85160534  0.18619262  0.646302674
## Texas          -1.34151838 -0.40833518 -0.48712332  0.636731051
## Utah            0.54503180 -1.45671524  0.29077592 -0.081486749
## Vermont         2.77325613  1.38819435  0.83280797 -0.143433697
## Virginia        0.09536670  0.19772785  0.01159482  0.209246429
## Washington      0.21472339 -0.96037394  0.61859067 -0.218628161
## West Virginia   2.08739306  1.41052627  0.10372163  0.130583080
## Wisconsin       2.05881199 -0.60512507 -0.13746933  0.182253407
## Wyoming         0.62310061  0.31778662 -0.23824049 -0.164976866
```

We can plot the first two principal components as follows:

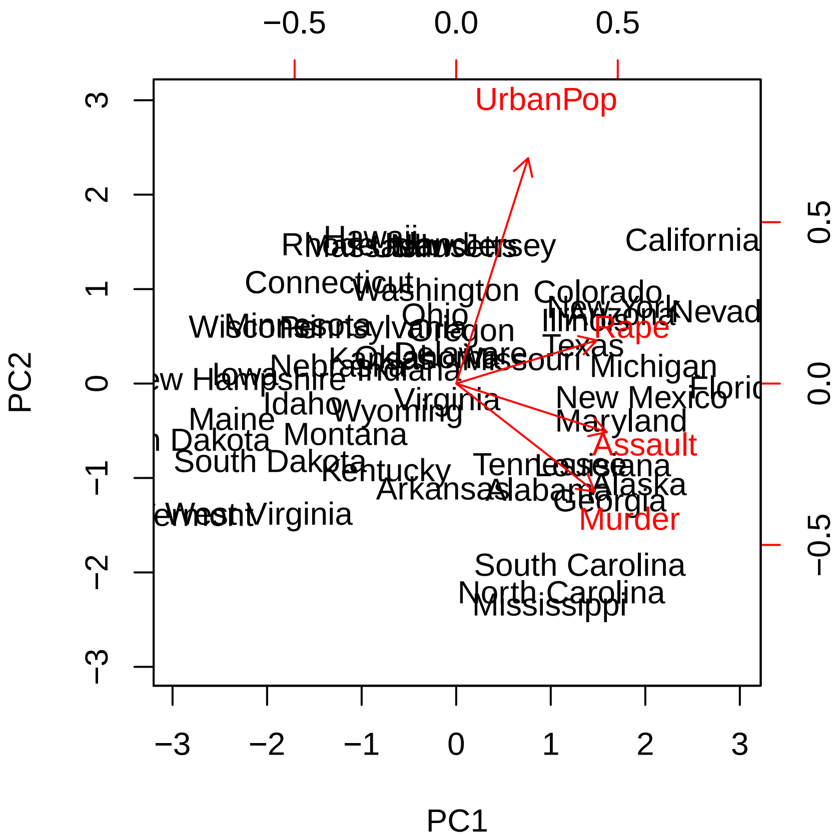
```
biplot(pr.out, scale=0)
```



The `scale=0` argument to `biplot()` ensures that the arrows are scaled to represent the loadings; other values for scale give slightly different biplots with different interpretations.

Notice that this figure is a mirror image of Figure 10.1. Recall that the principal components are only unique up to a sign change, so we can reproduce Figure 10.1 by making a few small changes:

```
pr.out$rotation=-pr.out$rotation
pr.out$x=-pr.out$x
biplot(pr.out, scale=0)
```



How many principal components are needed?

The `prcomp()` function also outputs the standard deviation of each principal component. For instance, on the USArrests data set, we can access these standard deviations as follows:

```
pr.out$sdev
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
```

The variance explained by each principal component is obtained by squaring these:

```
pr.var=pr.out$sdev ^2
pr.var
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

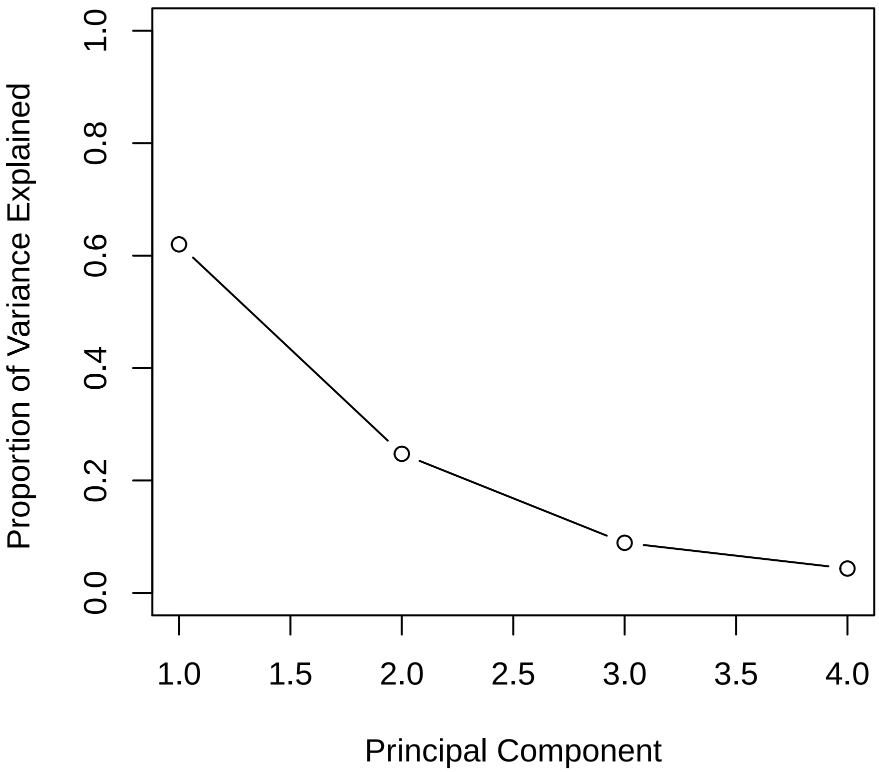
To compute the proportion of variance explained by each principal component, we simply divide the variance explained by each principal component by the total variance explained by all four principal components:

```
pve=pr.var/sum(pr.var)
pve
```

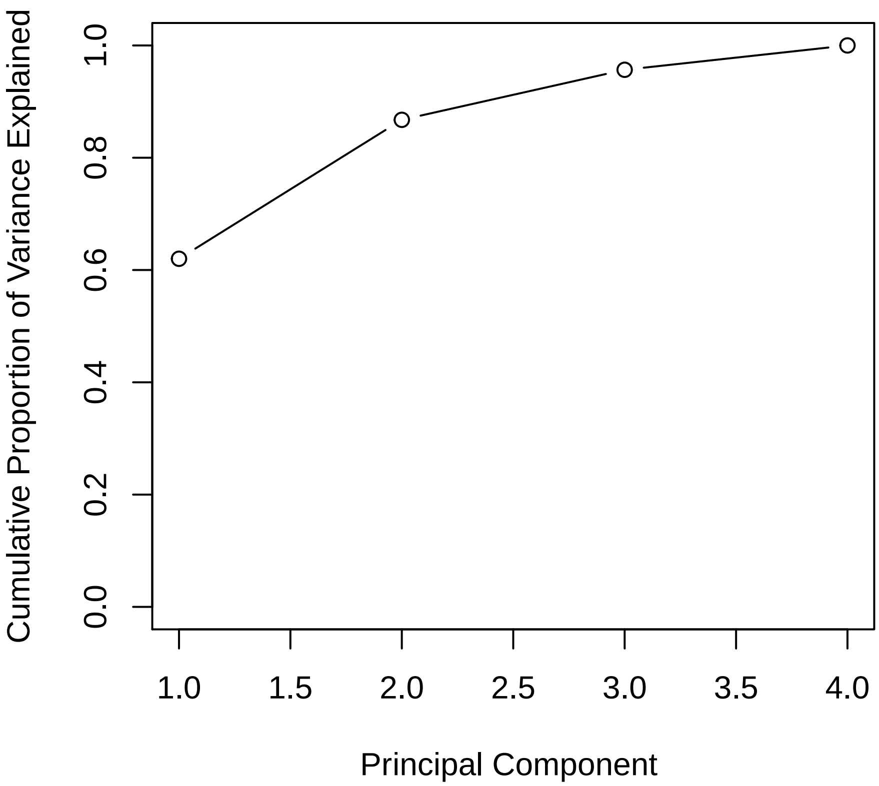
```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

We see that the first principal component explains 62.0% of the variance in the data, the next principal component explains 24.7% of the variance, and so forth. We can plot the PVE explained by each component, as well as the cumulative PVE, as follows:

```
plot(pve, xlab="Principal Component",
     ylab="Proportion of Variance Explained ", ylim=c(0,1),type='b')
```




```r
plot(cumsum(pve), xlab="Principal Component ",
     ylab=" Cumulative Proportion of Variance Explained ", ylim=c(0,1), type='b')
```



The result is shown in Figure 10.4. Note that the function `cumsum()` computes the cumulative sum of the elements of a numeric vector. For instance:

```r
a=c(1,2,8,-3)
cumsum(a)
```

```
## [1]  1  3 11  8
```

Bibliography

[1] James et. al. - An Introduction to Statistical Learning with Applications in R, Eigth Edition. Available at: http://www-bcf.usc.edu/~gareth/ISL/