TMA4268 Statistical Learning

Chapter 10: Unsupervised Learning - Lab 1

Thiago G. Martins, Department of Mathematical Sciences, NTNU

Spring 2020

Lab 1: Principal Components Analysis

We are going to use the `USArrests` dataset. It contains crime statistics per 100000 residents in 50 states of the USA. The crime types included are Assault, Murder and Rape. In addition, it contains information about the percent of the population in each state living in urban areas.

Data Exploration

Data rows

The rows of the data contains the 50 states, in alphabetical order.

```
states=row.names(USArrests) # part of the base R package
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"
```

Data columns

The columns of the data contain the four variables

```
names(USArrests)
```

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

Mean and variance

The variables have vastly different means

```
apply(USArrests, 2, mean)
```

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

The same is true for the variances

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

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

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.

PCA

We now go on to apply PCA on the standardized variables (mean 0 and std. dev. 1). 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.

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

PCA output

The output from `prcomp()` contains a number of useful quantities.

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

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

`center` and `scale`

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

```
pr.out$center
```

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

```
pr.out$scale
```

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

`rotation`

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

```
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
```

Score vectors

The 50×4 matrix `x` has as its columns the principal component score vectors. That is, the kth column is the k-th principal component score vector.

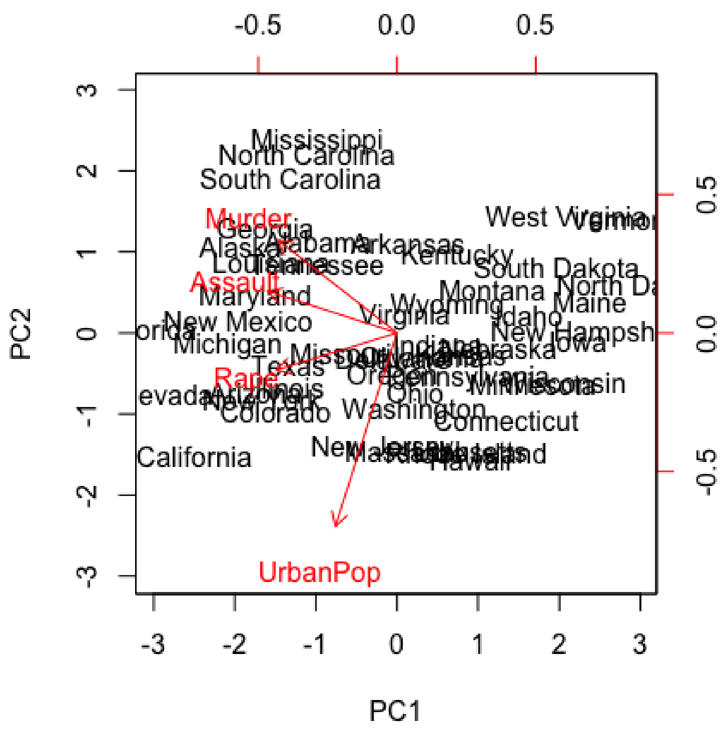
```
dim(pr.out$x)
```

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

Biplot

We can plot the first two principal components as follows:

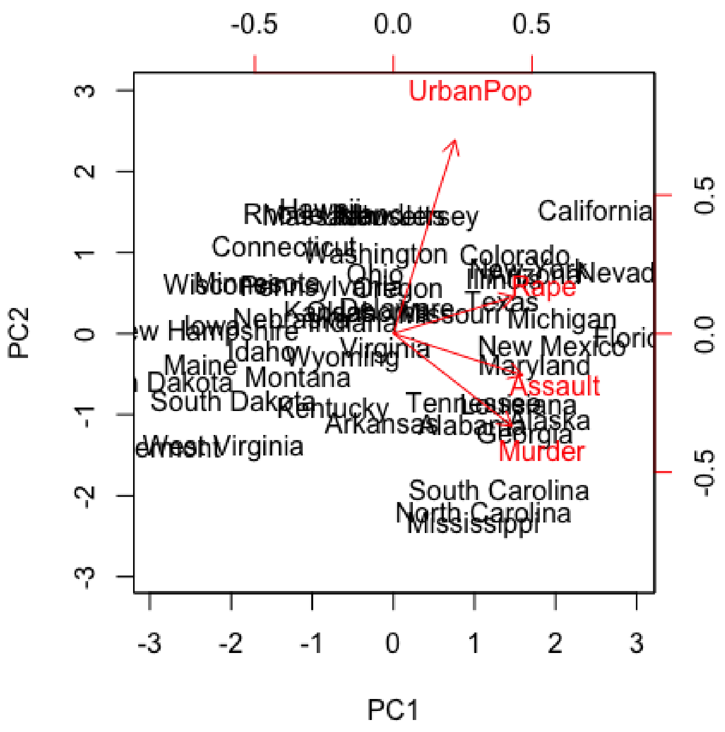
```
biplot(pr.out, scale=0)
```



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.

Remember that the PCs are unique up to a sign change, so the code below should give equivalent results

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



Proportion of variance explained (PVE)

The `prcomp()` function also outputs the standard deviation of each principal component.

```
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
```

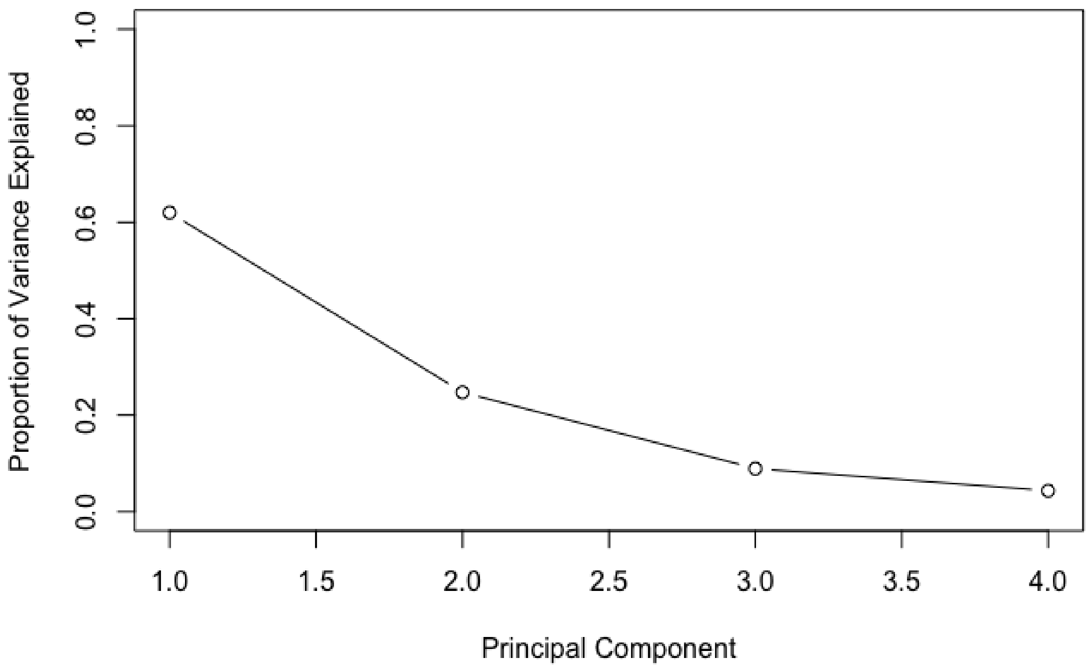
We can then compute the proportion of variance explained by each principal component

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

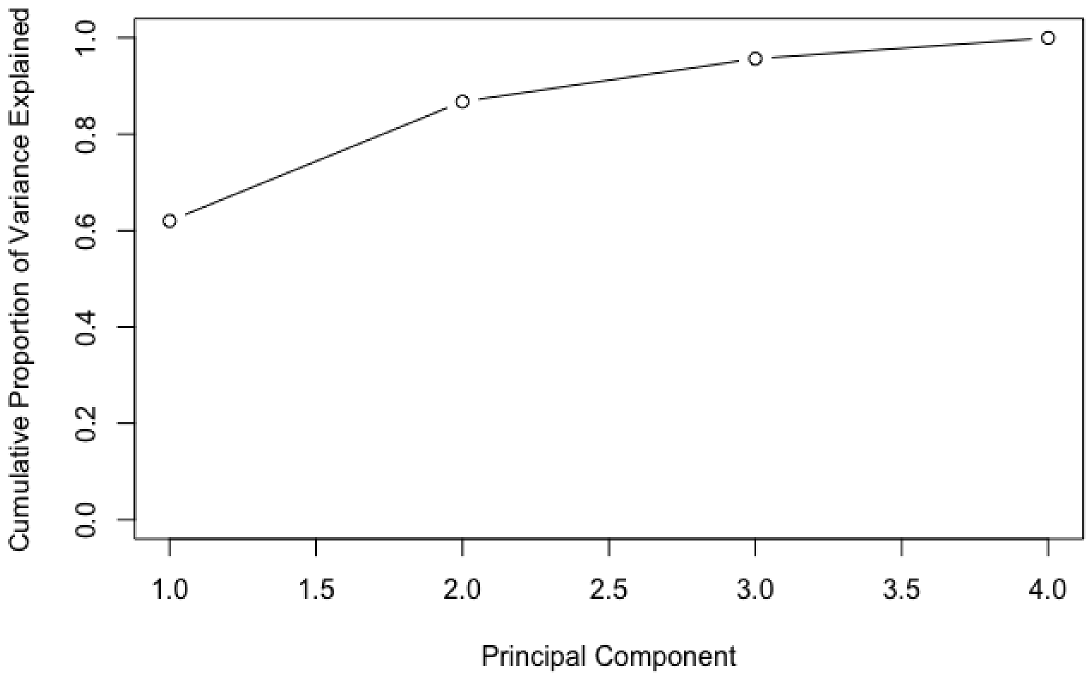
```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

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')
```



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


Cumulative Proportion of Variance Explained
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Principal Component