Variable Subset Selection and Principal Components Analysis

03/03/2020

1 Introduction

1.1 Objectives

- Familiarise with variable subset selection in R
- Understand how the different VSS strategy work
- Undestand and analyse the output results of Principal Component Analysis using R

1.2 Prerequiste

Libraries needed:

- MASS
- leaps
- ggbiplot
- datasets

```r
library(MASS)
library(leaps)
library(datasets)
library(ggbiplot)
```

ggbiplot might require a little more work to install:

```r
library("devtools")
install_github("vqv/ggbiplot")
library("ggbiplot")
```

2 Variable subset selction

Data used for this section is the CPUs dataset which contains data about the performance of 209 different computer CPUs. We will remove the 'name' and 'estperf variables' we will not need for this exercise.

```
data(cpus)
cpus$name <- NULL
cpus$estperf <- NULL
```

We want to analyse the difference between 3 subset selection methods:

- Exhaustive: tests every possible combination of variables. It will always obtain the optimal combination of varibles.

  ```
  regfit.full.exhaustive <- regsubsets(perf ~ .,
                            data = cpus, method = "exhaustive")

  subset.summary.exhaustive <- summary(regfit.full.exhaustive)
  ```

- Backward: Starts from all variables and removes at each step the the one that brings the least improvment.

  ```
  regfit.full.backward <- regsubsets(perf ~ .,
                          data = cpus, method = "backward")

  subset.summary.backward <- summary(regfit.full.backward)
  ```

- Forward: Starts with no variables and adds at each step the one bringing the most improvments

  ```
  regfit.full.forward <- regsubsets(perf ~ .,
                          data = cpus, method = "forward")

  subset.summary.forward <- summary(regfit.full.forward)
  ```

We can look at the subset proposed by each selection. the following shows the subsets selected by each method for each number of variables (from 1 to 6):

```
subset.summary.exhaustive$outmat

##          syct mmin mmax cach chmin chmax
## 1  ( 1 ) " "  " "  "*"  " "  " "   " "
```

```
## 2  ( 1 ) " "  " "  "*"  "*"  " "   " "
## 3  ( 1 ) " "  "*"  "*"  " "  " "   "*"
## 4  ( 1 ) " "  "*"  "*"  "*"  " "   "*"
## 5  ( 1 ) "*"  "*"  "*"  "*"  " "   "*"
## 6  ( 1 ) "*"  "*"  "*"  "*"  "*"   "*"
```

```
subset.summary.backward$outmat
```

```
##          syct mmin mmax cach chmin chmax
## 1  ( 1 ) " "  "*"  " "  " "  " "   " "
## 2  ( 1 ) " "  "*"  " "  " "  " "   "*"
## 3  ( 1 ) " "  "*"  "*"  " "  " "   "*"
## 4  ( 1 ) " "  "*"  "*"  "*"  " "   "*"
## 5  ( 1 ) "*"  "*"  "*"  "*"  " "   "*"
## 6  ( 1 ) "*"  "*"  "*"  "*"  "*"   "*"
```

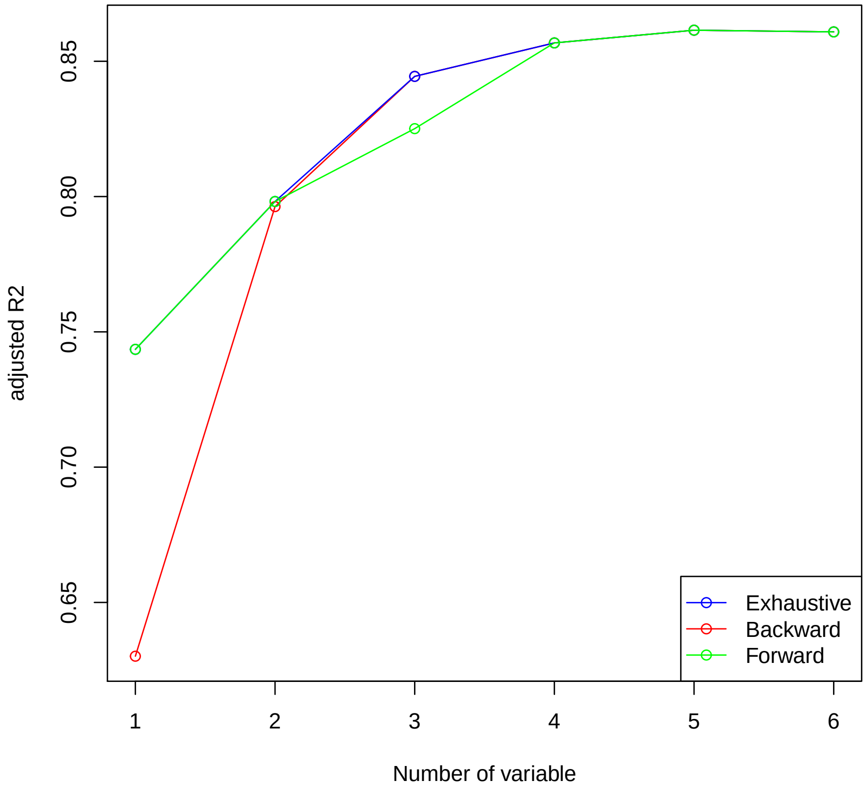
```
subset.summary.forward$outmat
```

```
##          syct mmin mmax cach chmin chmax
## 1  ( 1 ) " "  " "  "*"  " "  " "   " "
## 2  ( 1 ) " "  " "  "*"  "*"  " "   " "
## 3  ( 1 ) " "  "*"  "*"  "*"  " "   " "
## 4  ( 1 ) " "  "*"  "*"  "*"  " "   "*"
## 5  ( 1 ) "*"  "*"  "*"  "*"  " "   "*"
## 6  ( 1 ) "*"  "*"  "*"  "*"  "*"   "*"
```

Try to understand how the forward and the backward methods obtained those selections.

We can compare the results variable sebsets obtained by visualing the adjusted R^2 (you can use any other measures here):

```
plot(subset.summary.backward$adjr2, type = "o", col = "red", ylab = "adjusted R2", xlab = "N
lines(subset.summary.exhaustive$adjr2, type = "o", col = "blue")
lines(subset.summary.forward$adjr2, type = "o", col = "green")
legend("bottomright", col = c("blue","red", "green"), legend = c("Exhaustive", "Backward", "
```



Task : The exhaustive serach will always find the best combination of variable. We can see there is a difference between the backward and the forward method. The subset selected for 4, 5 and 6 variables are the same. But, the forward method finds a better subset for 1 and 2 variables. And the backward method is better for 3 variables. Considering how they work, can you explain why?

3 Principal Components Analysis

3.1 First look

Let's apply a PCA to the iris data set

```
data(iris)
#note that here, we remove the last column which is hte class column
pca.iris <- prcomp(iris[,-5], scale = TRUE)
pca.iris

## Standard deviations (1, .., p=4):
## [1] 1.7083611 0.9560494 0.3830886 0.1439265
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

We can first analyse the variance that is explained by each component

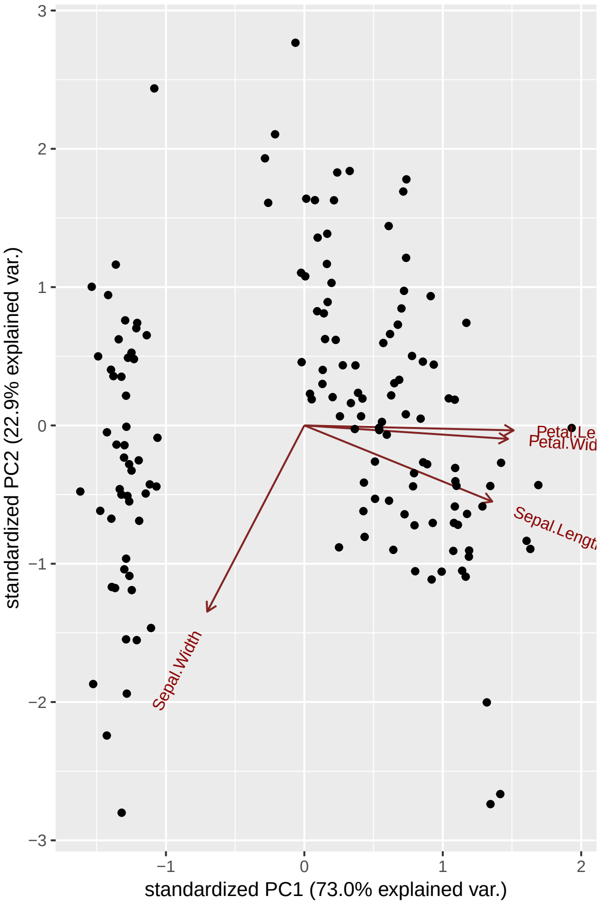
```
summary(pca.iris)

## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

We can see that the first component accounts for 72.96% of the variance in the dataset and the second one 22.85%. Hence, the cumulative proportion of the variance of PC1 and PC2 is 95.81%. In this case, PCA has allowed us to reduce the dimensionality of the dataset to 2 instead of 4.

We can then visualise the different projections of the data on the 2 first principal components:

```
ggbiplot(pca.iris)
```

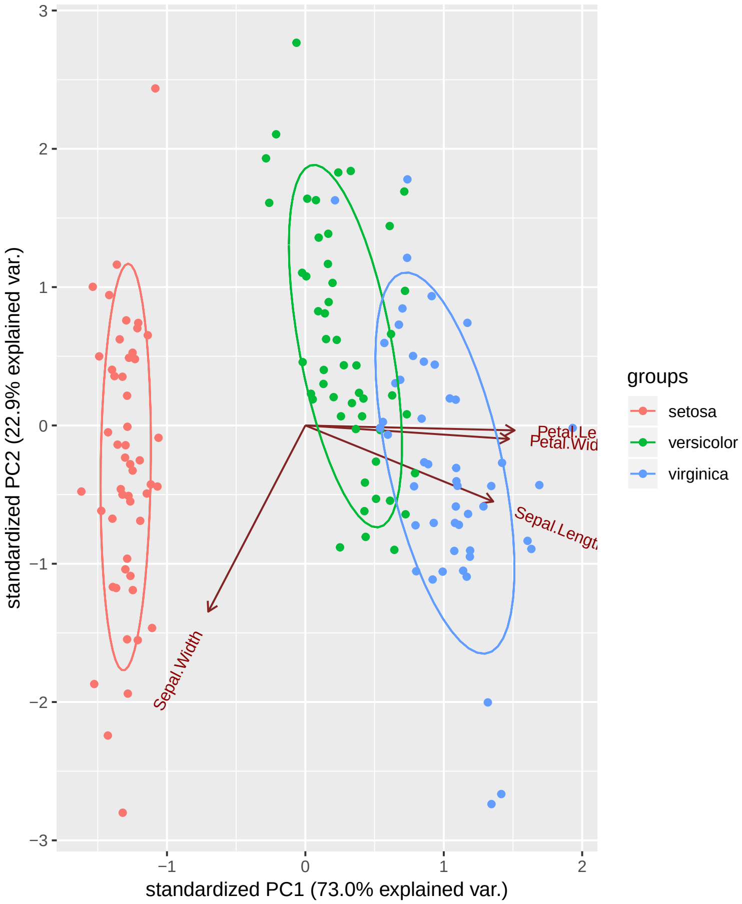


This visualisation also always us to visualise the participation of each variable in each component.

Task: Which variables have the most weight in the first principal component? Which variables have the most weight in the second principal components. Does it make sense if you compare this output with the correlation matrix?

PCA is an unsupervised learning technique. It means that we are not interested in using it for classification or regression purposes. In the case of the iris data set, the pricipal component analysis is done regardless of the class of a each observation. However, it can be used to visualise classified clustered data using:

```r
ggbiplot(pca.iris, groups = iris$Species, ellipse = TRUE)
```



3.2 Your turn

Repeat the process using the USArrests dataset:

```
data("USArrests")
```

- Apply PCA on this dataset and look at the biplot.
- What does it tell us about California? How about Vermont?