# Feature Selection Lab

Emily Bellis

6/14/2022

## Set-up

This lab is adapted from the Chp. 6 lab of Introduction to Statistical Learning, except we go a bit more in depth into PCA.

Load (and if necessary, install) the following packages:

```
library(ISLR2)  # for the Hitters dataset
library(pls)    # to implement principal components regression
library(tidyverse)  # for visualization and tidying
```

## Preprocessing and data exploration

Let's take a look at the example dataset. We'll try to predict Salary of a baseball player on the basis of various statistics associated with their performance in the previous year.

```
data(Hitters)
dim(Hitters)
```

```
## [1] 322  20
```

```
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

```
?Hitters
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

It appears we are missing values of salary for 59 players, let's update the the dataframe to drop `NA` values.

```
Hitters <- na.omit(Hitters) # will remove rows with an NA in any column
```

PCA is also not really the best choice of method for dealing with categorical factors. Let's just drop these for now.

```
str(Hitters)
```

```
## 'data.frame':    263 obs. of  20 variables:
##  $ AtBat    : int  315 479 496 321 594 185 298 323 401 574 ...
##  $ Hits     : int  81 130 141 87 169 37 73 81 92 159 ...
##  $ HmRun    : int  7 18 20 10 4 1 0 6 17 21 ...
##  $ Runs     : int  24 66 65 39 74 23 24 26 49 107 ...
##  $ RBI      : int  38 72 78 42 51 8 24 32 66 75 ...
```

```
##  $ Walks    : int  39 76 37 30 35 21 7 8 65 59 ...
##  $ Years    : int  14 3 11 2 11 2 3 2 13 10 ...
##  $ CAtBat   : int  3449 1624 5628 396 4408 214 509 341 5206 4631 ...
##  $ CHits    : int  835 457 1575 101 1133 42 108 86 1332 1300 ...
##  $ CHmRun   : int  69 63 225 12 19 1 0 6 253 90 ...
##  $ CRuns    : int  321 224 828 48 501 30 41 32 784 702 ...
##  $ CRBI     : int  414 266 838 46 336 9 37 34 890 504 ...
##  $ CWalks   : int  375 263 354 33 194 24 12 8 866 488 ...
##  $ League   : Factor w/ 2 levels "A","N": 2 1 2 2 1 2 1 2 1 1 ...
##  $ Division : Factor w/ 2 levels "E","W": 2 2 1 1 2 1 2 2 1 1 ...
##  $ PutOuts  : int  632 880 200 805 282 76 121 143 0 238 ...
##  $ Assists  : int  43 82 11 40 421 127 283 290 0 445 ...
##  $ Errors   : int  10 14 3 4 25 7 9 19 0 22 ...
##  $ Salary   : num  475 480 500 91.5 750 ...
##  $ NewLeague: Factor w/ 2 levels "A","N": 2 1 2 2 1 1 1 2 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:59] 1 16 19 23 31 33 37 39 40 42 ...
##   ..- attr(*, "names")= chr [1:59] "-Andy Allanson" "-Billy Beane" "-Bruce Bochte" "-Bob Boone" ...
```

```r
Hitters <- Hitters %>% select(-c(League, Division, NewLeague)) # remove these three columns
dim(Hitters)
```

```
## [1] 263  17
```

Let's also get a feel for the mean and variance in each of the remaining predictors. Unless features are measured in the same units, we usually scale each variable so that one variable doesn't dominate over the others, just because it is measured on a different scale.

```r
apply(Hitters, 2, mean) # means
```

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks
##  403.642586  107.828897   11.619772   54.745247   51.486692   41.114068
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI
##    7.311787 2657.543726  722.186312   69.239544  361.220532  330.418251
##      CWalks     PutOuts     Assists      Errors      Salary
##  260.266160  290.711027  118.760456    8.593156  535.925882
```

```r
apply(Hitters, 2, var)  # variances
```

```
##        AtBat         Hits        HmRun         Runs          RBI        Walks
## 2.169941e+04 2.036295e+03 7.668694e+01 6.522822e+02 6.699149e+02 4.716740e+02
##        Years       CAtBat        CHits       CHmRun        CRuns         CRBI
## 2.297875e+01 5.228461e+06 4.201628e+05 6.756442e+03 1.096925e+05 1.045666e+05
##       CWalks      PutOuts      Assists       Errors       Salary
## 6.972550e+04 7.836337e+04 2.104837e+04 4.364682e+01 2.035081e+05
```

## Carrying out the PCA

We can carry out PCA using `prcomp` from base R

```r
pr.out <- prcomp(Hitters, scale = T) # scale to have s.d. = 1; values are also centered to have mean ze
```

Let's get a look at the fitted `pr.out` object

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

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

The `center` and `scale` components correspond to the means and standard deviations prior to implementing PCA.

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

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks 
##  403.642586  107.828897   11.619772   54.745247   51.486692   41.114068 
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI 
##    7.311787 2657.543726  722.186312   69.239544  361.220532  330.418251 
##      CWalks     PutOuts     Assists      Errors      Salary 
##  260.266160  290.711027  118.760456    8.593156  535.925882
```

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

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks 
##  147.307209   45.125326    8.757108   25.539816   25.882714   21.718056 
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI 
##    4.793616 2286.582929  648.199644   82.197581  331.198571  323.367668 
##      CWalks     PutOuts     Assists      Errors      Salary 
##  264.055868  279.934575  145.080577    6.606574  451.118681
```

The `rotation` matrix provides the principal component loadings; each column of `pr.out$rotation` contains the corresponding PC loading vector. Note there are 17 distinct PCs. This is to be expected since there are in general $min(n-1, p)$ in a dataset with $n$ observations and $p$ variables.

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

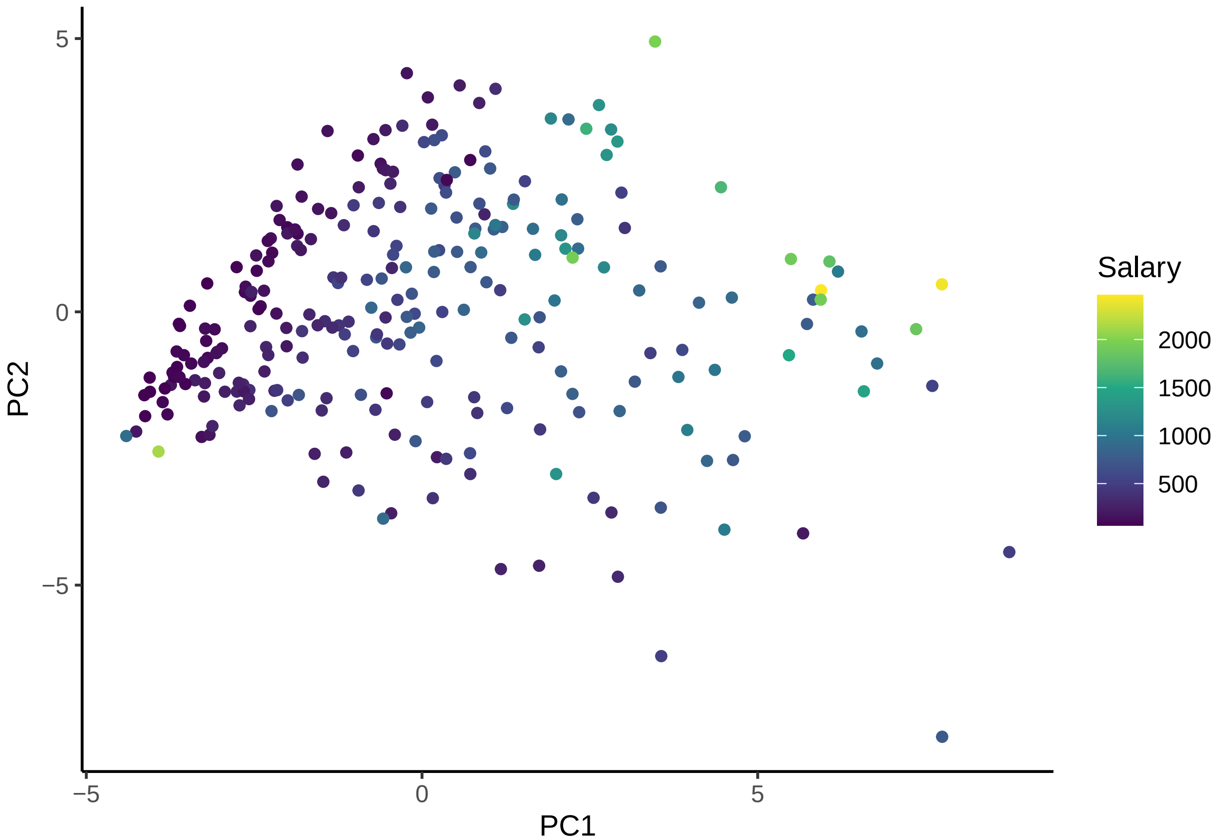
We can also get the principal component scores:

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

## Plotting PCA Results

Many functions and packages exist to plot results of PCA; personally I like to extract the PC scores and create a new dataframe to visualize w/`ggplot` :)

```r
df <- cbind.data.frame(Salary = Hitters$Salary,
                       PC1 = pr.out$x[,1],
                       PC2 = pr.out$x[,2])
ggplot(df, aes(x = PC1, y = PC2, col = Salary)) +
  geom_point() +
  theme_classic() +
  scale_color_viridis_c()
```




### Proportion of Variation Explained

The `prcomp` object also contains the standard deviation of each PC.

```
pr.out$sdev
```

```
##  [1] 2.77339670 2.03026013 1.31485574 0.95754099 0.84109683 0.72374220
##  [7] 0.69841796 0.50090065 0.42525940 0.36390198 0.31201168 0.24364151
## [13] 0.23204483 0.16351047 0.11863984 0.06933950 0.03466841
```

```
pr.var <- pr.out$sdev^2
pr.var
```

```
##  [1] 7.691729243 4.121956205 1.728845616 0.916884744 0.707443873 0.523802766
##  [7] 0.487787650 0.250901462 0.180845560 0.132424652 0.097351288 0.059361186
## [13] 0.053844803 0.026735675 0.014075412 0.004807967 0.001201899
```

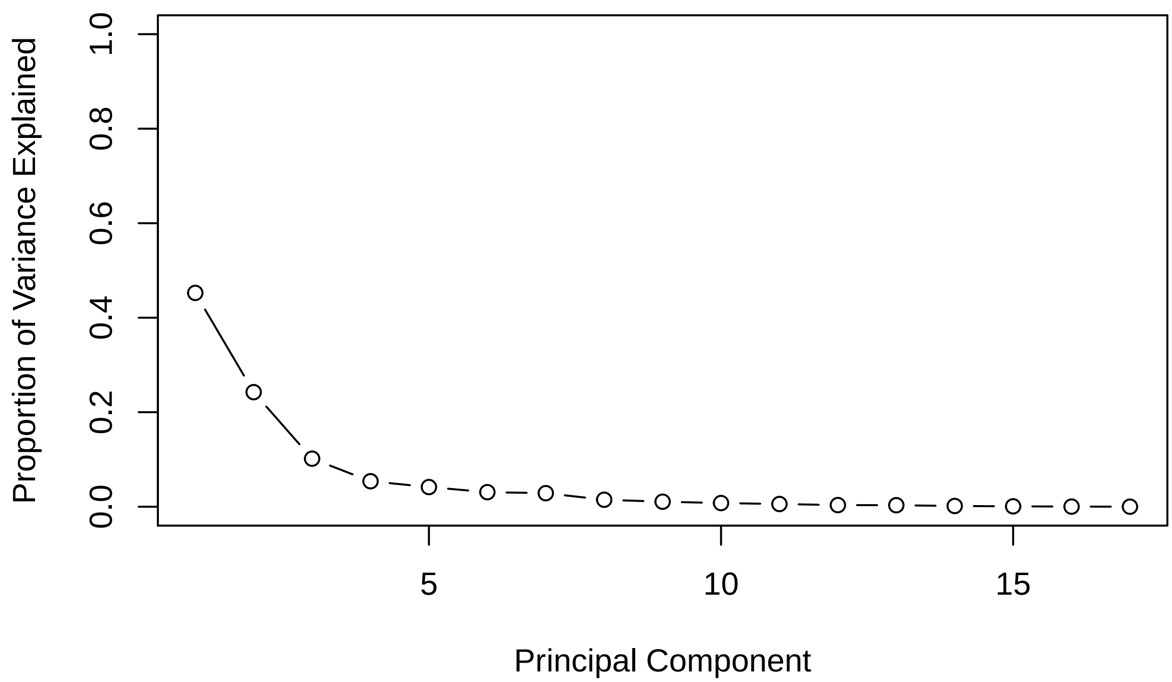
These values can be used to calculate the percent variation explained by each PC axis:

```
pve <- pr.var/sum(pr.var)
pve
```

```
##  [1] 4.524547e-01 2.424680e-01 1.016968e-01 5.393440e-02 4.161435e-02
##  [6] 3.081193e-02 2.869339e-02 1.475891e-02 1.063797e-02 7.789685e-03
## [11] 5.726546e-03 3.491834e-03 3.167341e-03 1.572687e-03 8.279654e-04
## [16] 2.828216e-04 7.069994e-05
```

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

```
    type = "b")
```




### Principal Components Regression

Let's first split the dataset into a portion of the data for training, and a separate hold-out set of 70% of the data for testing.

```
set.seed(1) # so result we get the same results for randomization
train <- sample(1:nrow(Hitters), nrow(Hitters)*.7)
test <- (-train)
```

If we want to carry out principal components regression directly, we can use the `pcr` function from the `pls` package to carry out both PCA and regression at once. It even carries out 10-fold cross-validation for each value of $M$ (number of PCs used).

How many principal components should we use to make the final prediction?

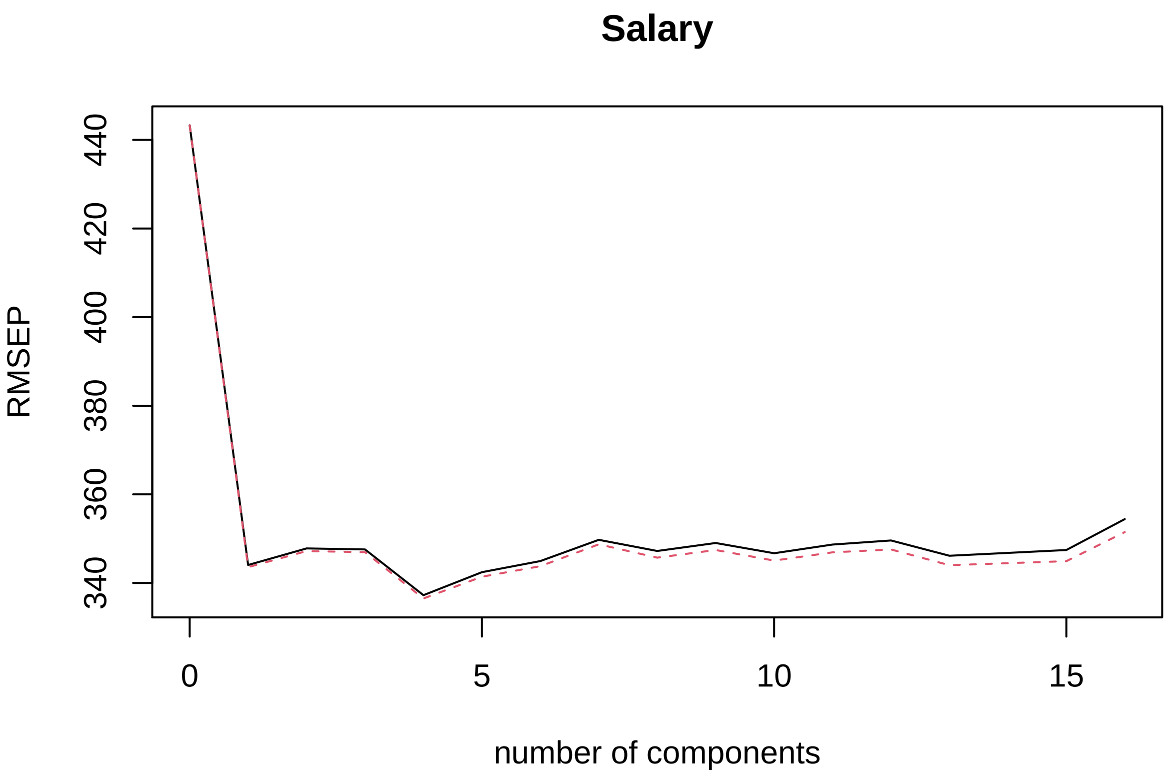
```
pcr.fit <- pcr(Salary ~ ., data = Hitters, scale = T, subset = train, validation = "CV")

summary(pcr.fit)
```

```
## Data:    X dimension: 184 16
##  Y dimension: 184 1
## Fit method: svdpc
## Number of components considered: 16
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           443.3    344.0    347.8    347.6    337.2    342.4    344.9
## adjCV        443.3    343.6    347.2    347.0    336.5    341.4    343.8
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       349.7    347.2    349.0     346.7     348.7     349.6     346.1
## adjCV    348.7    345.7    347.4     345.0     346.9     347.6     344.0
```

```
##         14 comps  15 comps  16 comps
## CV         346.8     347.4     354.4
## adjCV      344.5     344.9     351.5
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X          45.31    71.37    82.57    88.38    92.40    95.10    96.49    97.72
## Salary     41.59    41.73    41.92    46.33    46.42    46.51    46.97    48.19
##          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X          98.46     99.00     99.41     99.72     99.88     99.96     99.99
## Salary     48.27     49.31     49.31     50.21     52.08     52.94     53.81
##          16 comps
## X             100
## Salary         54
```

```r
validationplot(pcr.fit, val.type = "RMSEP") # plot cross-validation Root Mean Squared Error
```




Determine the test MSE as follows:

```r
pcr.pred <- predict(pcr.fit, Hitters[test,], ncomp = 4) # if we choose to include the first four PCs
mean((pcr.pred - Hitters[test,]$Salary)^2)
```

```
## [1] 144261.4
```