Lab 6.3 - PCR and PLS Regression

An Introduction to Statistical Learning

We will use the `Hitters` dataset.

```
pacman::p_load(ISLR, pls)

Hitters = na.omit(Hitters)
attach(Hitters)
```

Let's create a matrix with the observations and a targets vector:

```
x = model.matrix(Salary~., data = Hitters)[, -1]
y = Salary
```

Split the data in training and test sets:

```
set.seed(1)
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
```

1. Principal Component Regression (PCR)

Fitting the model

PCR is performed using the `pcr()` function from the `pls` library. Data can be introduced as a `data.frame`, in this case `Hitters`, or as `model.matrix`, `x` and `y`.

```
set.seed(1)
#pcr.fit = pcr(Salary~., data = Hitters, subs = train, scale = TRUE, validation = 'CV')
pcr.fit = pcr(y~x, subs = train, scale = TRUE, validation = 'CV')
```

Setting `scale=TRUE` standardizes the data, which is necessary if the variables are in different ranges, units, etc.

Validation can be computed setting `validation="CV"` for cross-validation, which by default is a 10-fold for each possible M, number of principal components used. If `validation="LOO"`, *leave-one-out* cross-validation is performed.

The results can be printed with `summary()`:

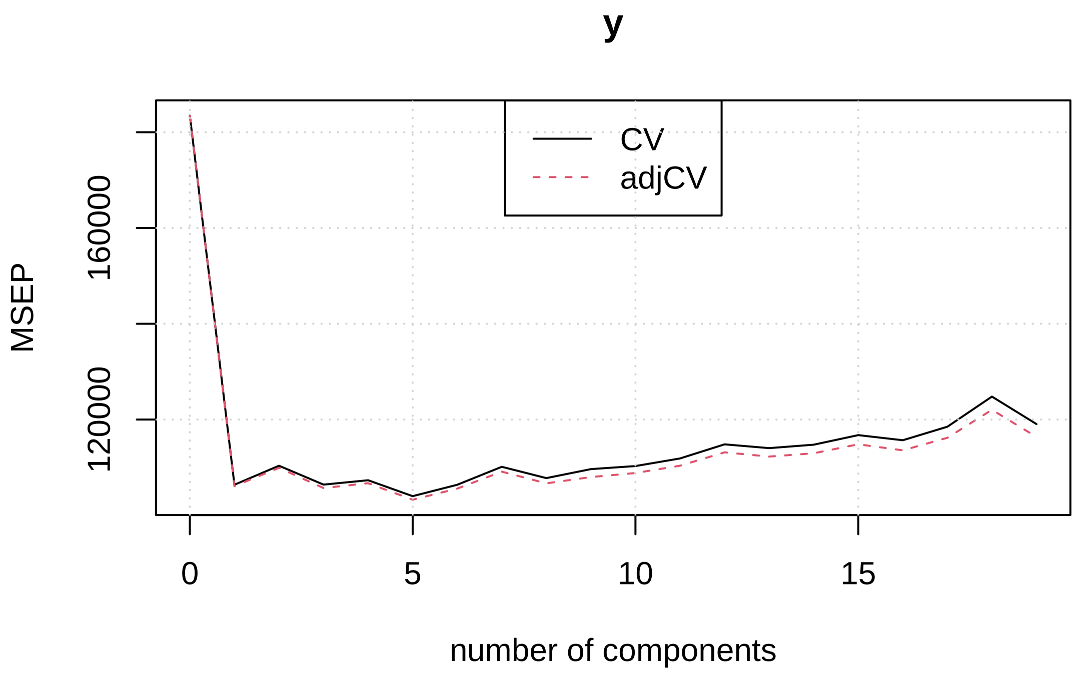
```
summary(pcr.fit)
```

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    326.2    332.2    326.2    327.6    322.5    326.2
## adjCV        428.3    325.7    331.5    325.1    326.7    321.3    324.9
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       331.9    328.3    331.1     332.1     334.5     338.9     337.7
## adjCV    330.3    326.6    328.6     329.9     332.2     336.4     335.1
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        338.7     341.7     340.1     344.2     353.3     345.0
## adjCV     336.1     338.9     337.0     340.9     349.3     341.1
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    39.32    61.57    71.96    80.83    85.95    89.99    93.25    95.34
## y    43.87    43.93    47.36    47.37    49.52    49.55    49.63    50.98
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    96.55     97.61     98.28     98.85     99.22     99.53     99.79
## y    53.00     53.00     53.02     53.05     53.80     53.85     54.03
##    16 comps  17 comps  18 comps  19 comps
## X     99.91     99.97     99.99    100.00
## y     55.85     55.89     56.21     58.62
```

The VC *root mean squared error*, RMSEP, is shown in ascending order of *M*. MSE is just the square of RMSEP.

The results of the CV can be plotted, giving the statistic to be plotted in `val.type`:

```
validationplot(pcr.fit, val.type = 'MSEP', legendpos = 'top'); grid()
```



The plot shows that a model with 5 components has the lowest Cross-Validation error.

`summary()` also prints the *percentage of variance explained* in the predictors and in the response for the different values of *M*.

Making prections

We now use the 6-component model to check performance on the test set. We can pass the new data as as `data.frame`, or as `model.matrix` like before. The number of components is given in `ncomp`:

```
#pcr.pred = predict(pcr.fit, newdata = Hitters, subset = test, ncopm = 6)
pcr.pred = predict(pcr.fit, newdata = x, subset = test, ncomp = 6)

cat(sprintf("Mean error for the 6-component model: %.2f",
            mean((x[test,] - Salary[test])^2)))
```

```
## Mean error for the 6-component model: 1023671.63
```

Fitting the complete model

Finally we refit the model with all the data:

```
pcr.fit = pcr(y~x, scale = TRUE, ncomp = 6)
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 6
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X    38.31    60.16    70.84    79.03    84.29    88.63
## y    40.63    41.58    42.17    43.22    44.90    46.48
```

2. Partial Least Squares Regression (PLSR)

Partial Least Squares regression is a *supervised* alternative tp PCR, which identifies a new set of features, $Z_1, \ldots, Z_M$ that are linear combinations of the original features and then fits a linear model via least squares using this M new features, but making use of the response Y to identify new features that not only approximate the old features well, but also are *related to the response*.

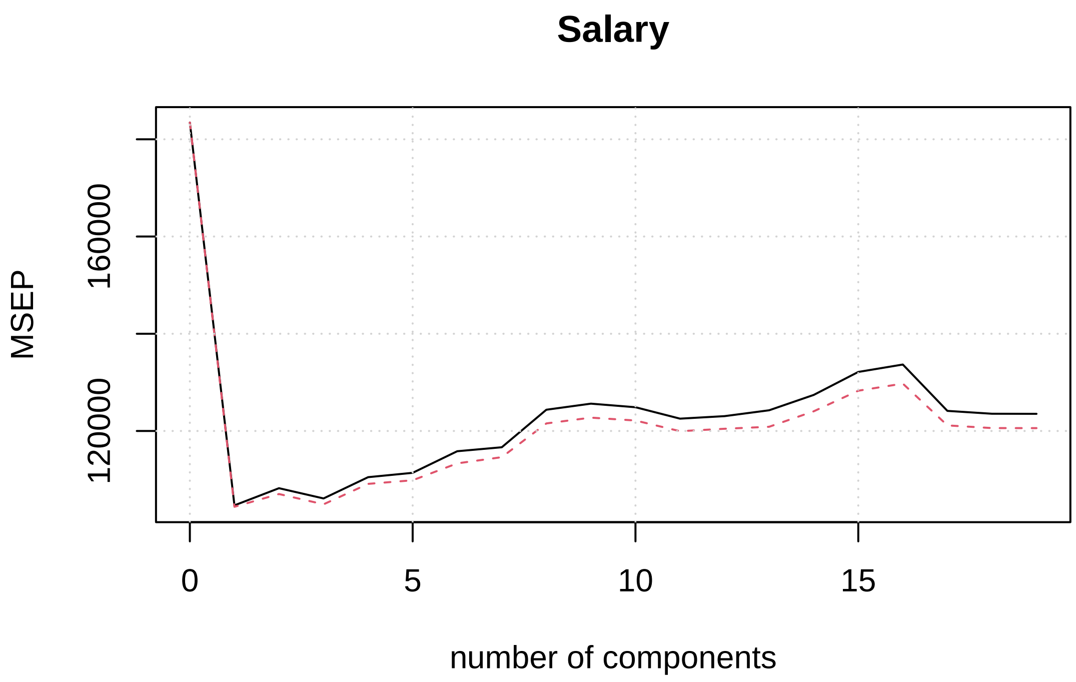
Fitting the model

PSLR is performed using the `pslr()` method in the `pls` library, and its syntax is the same that of the `pcr()` method:

```
set.seed(42)
pls.fit = plsr(Salary~., data = Hitters, subset = train, scale = TRUE, validation = "CV")
summary(pls.fit)
```

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    323.6    329.0    325.8    332.4    333.8    340.3
## adjCV        428.3    323.1    327.2    323.9    330.3    331.5    336.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       341.5    352.7    354.4     353.4     350.0     350.8     352.5
## adjCV    338.5    348.6    350.4     349.5     346.3     347.1     347.7
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        357.0     363.5     365.6     352.3     351.5     351.5
## adjCV     352.2     358.2     360.2     348.1     347.3     347.3
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         39.13    48.80    60.09    75.07    78.58    81.12    88.21    90.71
## Salary    46.36    50.72    52.23    53.03    54.07    54.77    55.05    55.66
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         93.17     96.05     97.08     97.61     97.97     98.70     99.12
## Salary    55.95     56.12     56.47     56.68     57.37     57.76     58.08
##         16 comps  17 comps  18 comps  19 comps
## X          99.61     99.70     99.95    100.00
## Salary     58.17     58.49     58.56     58.62
```

```
validationplot(pls.fit, val.type = 'MSEP'); grid()
```



Making predictions

The lowest CV error occurs for $M = 1$ partial least square directions.

Let's evaluate the test set MSE:

```
pls.pred = predict(pls.fit, newdata = Hitters[test,], ncomp = 1)
mean((pls.pred - Salary[test])^2)
```

```
## [1] 151995.3
```

Fitting the complete model

Let's refit the model with all the data and using `ncomp=2`:

```
pls.fit = plsr(Salary~., data = Hitters, scale = TRUE, ncomp = 2)
summary(pls.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 2
## TRAINING: % variance explained
##         1 comps  2 comps
## X         38.08    51.03
## Salary    43.05    46.40
```

The variance explaned by the 2-components PLS model *in the target* `Salary`, 46.40%, is approximately the same as the obtained with the 6-component PCR model, 46.48%.