# In Class Asssignment 3

*Lydia Strebe*

*February 12, 2019*

## Calculating 10-Fold CV and LOOCV

The code below creates 32 linear regression models from 5 possible regressors to predict number of wins in a baseball season. Each regression is evaluated using both 10-Fold Cross Validation and Leave One Out Cross Validation.

```
require(boot)
```

```
## Loading required package: boot
```

```
# Reading in the data file
baseball_original=read.csv("C:/Users/lydia/Documents/baseball.csv",header=TRUE)

# Omitting the rows with missing data
baseball=baseball_original[1:420,]

# Creating  arrays to hold the 10-fold CV error and LOOCV error for each regression model
cv_10err = array(0,32)
loocv_err = array(0,32)


# Loop to run all 32 regression models and calculate both the 10-Fold Cross Validation and LOOCV
error
for(i in 0:31){
 a = as.integer(intToBits(i)[1:5])
 if(a[1] == 0 && a[2]==0 && a[3]==0 && a[4]==0 && a[5]==0)
 {
   param0 = "1"
 }
 else{
   param0 = ""
 }
 if(a[1]==1){
   param1 = "+OBP"
 }
 else{
   param1 = ""
 }
 if(a[2]==1){
   param2 = "+SLG"
 }
 else{
   param2 = ""
 }
 if(a[3]==1){
   param3 = "+BA"
 }
 else{
   param3 = ""
 }
 if(a[4]==1){
   param4 = "+OOBP"
 }
 else{
   param4 = ""
 }
 if(a[5]==1){
   param5 = "+OSLG"
 }
 else{
   param5 = ""
 }
 param_list=paste("W~",param0, param1, param2, param3, param4, param5, sep='')
```

```
 fit=glm(as.formula(param_list),data=baseball)
 cv_10err[i+1]=cv.glm(baseball,fit,K=10)$delta[1]
 loocv_err[i+1]=cv.glm(baseball,fit)$delta[1]
}
cv_10err
```

```
##  [1] 135.57613  99.45733 102.41957  96.29499 114.79325  99.02180 103.80817
##  [8]  94.98500  81.70402  32.00918  33.49143  26.33331  44.93327  32.10170
## [15]  31.80157  26.43326  92.15346  40.41098  38.35265  32.42245  53.40052
## [22]  40.57392  36.96755  32.62119  81.14951  29.76891  29.43604  22.67631
## [29]  42.00206  29.52970  27.78543  22.76514
```

```
loocv_err
```

```
##  [1] 135.86341  99.20064 102.70071  96.70280 115.41830  98.95726 103.12207
##  [8]  95.01310  81.57998  31.99998  33.41682  26.41277  44.70642  31.95671
## [15]  31.77785  26.48072  92.36383  40.50615  38.07609  32.36789  53.25857
## [22]  40.40812  36.89112  32.48621  81.12077  29.66546  29.30100  22.77048
## [29]  42.02894  29.45857  27.58645  22.87867
```

# Evaluating the Models

The following code finds and prints the 5 best subsets of regressors as determined by 10-Fold CV and LOOCV.

```
# Sorting the arrays from min to max, including their indices
cv_sorted=sort(cv_10err,index.return=TRUE)
cv_sorted
```

```
## $x
##  [1]  22.67631  22.76514  26.33331  26.43326  27.78543  29.43604  29.52970
##  [8]  29.76891  31.80157  32.00918  32.10170  32.42245  32.62119  33.49143
## [15]  36.96755  38.35265  40.41098  40.57392  42.00206  44.93327  53.40052
## [22]  81.14951  81.70402  92.15346  94.98500  96.29499  99.02180  99.45733
## [29] 102.41957 103.80817 114.79325 135.57613
##
## $ix
##  [1] 28 32 12 16 31 27 30 26 15 10 14 20 24 11 23 19 18 22 29 13 21 25  9
## [24] 17  8  4  6  2  3  7  5  1
```

```
loocv_sorted=sort(loocv_err,index.return=TRUE)
loocv_sorted
```

```
## $x
##  [1]  22.77048  22.87867  26.41277  26.48072  27.58645  29.30100  29.45857
##  [8]  29.66546  31.77785  31.95671  31.99998  32.36789  32.48621  33.41682
## [15]  36.89112  38.07609  40.40812  40.50615  42.02894  44.70642  53.25857
## [22]  81.12077  81.57998  92.36383  95.01310  96.70280  98.95726  99.20064
## [29] 102.70071 103.12207 115.41830 135.86341
##
## $ix
##  [1] 28 32 12 16 31 27 30 26 15 14 10 20 24 11 23 19 22 18 29 13 21 25  9
## [24] 17  8  4  6  2  3  7  5  1
```

```
# Loop  that prints out the regressors of the 5 best models according to 10-fold cv
for(j in 1:5){
 a = as.integer(intToBits(cv_sorted$ix[j]-1)[1:5])
 if(a[1] == 0 && a[2]==0 && a[3]==0 && a[4]==0 && a[5]==0)
 {
   param0 = "1"
 }
 else{
   param0 = ""
 }
 if(a[1]==1){
   param1 = "+OBP"
 }
 else{
   param1 = ""
 }
 if(a[2]==1){
   param2 = "+SLG"
 }
 else{
   param2 = ""
 }
 if(a[3]==1){
   param3 = "+BA"
 }
 else{
   param3 = ""
 }
 if(a[4]==1){
   param4 = "+OOBP"
 }
 else{
   param4 = ""
 }
 if(a[5]==1){
   param5 = "+OSLG"
 }
 else{
   param5 = ""
 }
 param_list=paste("W~",param0, param1, param2, param3, param4, param5, sep='')
 print(param_list)
}
```

```
## [1] "W~+OBP+SLG+OOBP+OSLG"
## [1] "W~+OBP+SLG+BA+OOBP+OSLG"
## [1] "W~+OBP+SLG+OOBP"
## [1] "W~+OBP+SLG+BA+OOBP"
## [1] "W~+SLG+BA+OOBP+OSLG"
```

```
# Loop  that prints out the regressors of the 5 best models according to loocv
for(j in 1:5){
 a = as.integer(intToBits(loocv_sorted$ix[j]-1)[1:5])
 if(a[1] == 0 && a[2]==0 && a[3]==0 && a[4]==0 && a[5]==0)
 {
   param0 = "1"
 }
 else{
   param0 = ""
 }
 if(a[1]==1){
   param1 = "+OBP"
 }
 else{
   param1 = ""
 }
 if(a[2]==1){
   param2 = "+SLG"
 }
 else{
   param2 = ""
 }
 if(a[3]==1){
   param3 = "+BA"
 }
 else{
   param3 = ""
 }
 if(a[4]==1){
   param4 = "+OOBP"
 }
 else{
   param4 = ""
 }
 if(a[5]==1){
   param5 = "+OSLG"
 }
 else{
   param5 = ""
 }
 param_list=paste("W~",param0, param1, param2, param3, param4, param5, sep='')
 print(param_list)
}
```

```
## [1] "W~+OBP+SLG+OOBP+OSLG"
## [1] "W~+OBP+SLG+BA+OOBP+OSLG"
## [1] "W~+OBP+SLG+OOBP"
## [1] "W~+OBP+SLG+BA+OOBP"
## [1] "W~+SLG+BA+OOBP+OSLG"
```

The two methods have basically the same line-up of models. However, the order of a few are switched depending on the method.

The 5 models predict out-of-sample fairly well. The cross validation error ranges from 22 to 28.

Of the top 5 models, the 3rd best (for both methods) uses the fewest regressors (only three): OBP (On-Base-Percentage), SLG (Slugging), and OOBP (Opponent-OBP). According to LOOCV, the out-of-sample error is about 26.4 which is not drastically worse than the best model for which the out-of-sample error is about 22.8.