4.3.1: R - Nonparametric Regression Methods (LOESS, Regression Trees, and Random Forests) Stat 5100: Dr. Bean

Example: Baseball dataset (same as Handout 4.1.1)

```
library(stat5100)
data(baseball)
head(baseball)
```

```
##                Name      Team nAtBat nHits nHome nRuns nRBI nBB YrMajor CrAtBat
## 1    Allanson, Andy Cleveland    293    66     1    30   29  14       1     293
## 2       Ashby, Alan   Houston    315    81     7    24   38  39      14    3449
## 3       Davis, Alan   Seattle    479   130    18    66   72  76       3    1624
## 4     Dawson, Andre  Montreal    496   141    20    65   78  37      11    5628
## 5 Galarraga, Andres  Montreal    321    87    10    39   42  30       2     396
## 6  Griffin, Alfredo   Oakland    594   169     4    74   51  35      11    4408
##   CrHits CrHome CrRuns CrRbi CrBB   League Division Position nOuts nAssts
## 1     66      1     30    29   14 American     East        C   446     33
## 2    835     69    321   414  375 National     West        C   632     43
## 3    457     63    224   266  263 American     West       1B   880     82
## 4   1575    225    828   838  354 National     East       RF   200     11
## 5    101     12     48    46   33 National     East       1B   805     40
## 6   1133     19    501   336  194 American     West       SS   282    421
##   nError Salary Div logSalary
## 1     20     NA  AE        NA
## 2     10  475.0  NW  6.163315
## 3     14  480.0  AW  6.173786
## 4      3  500.0  NE  6.214608
## 5      4   91.5  NE  4.516339
## 6     25  750.0  AW  6.620073
```

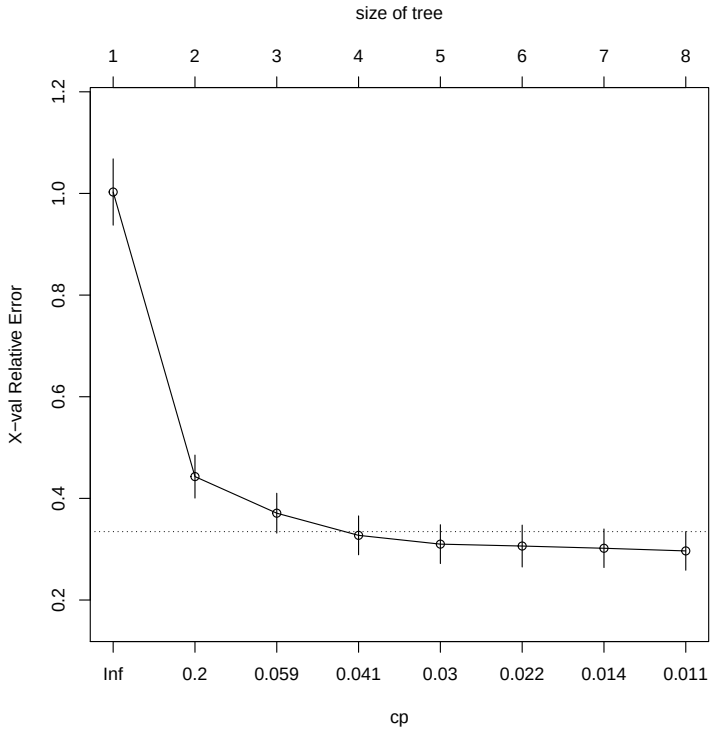
LOESS Regression

```
baseball_loess <- loess(logSalary ~ CrAtBat + nBB, data = baseball,
                        degree = 2, span = 0.6)

crAtBat_range <- seq(min(baseball$CrAtBat), max(baseball$CrAtBat), length = 500)
nBB_range <- seq(min(baseball$nBB), max(baseball$nBB), length = 500)
loess_pred <- function(x, y) {
  predict(baseball_loess, data.frame(CrAtBat = x,
                                     nBB = y))
}

z <- outer(crAtBat_range, nBB_range, loess_pred)

# plotly::plot_ly(
```

```
#   x = crAtBat_range,
#   y = nBB_range,
#   z = outer(crAtBat_range, nBB_range, loess_pred),
#   trace = "contour"
# )
```

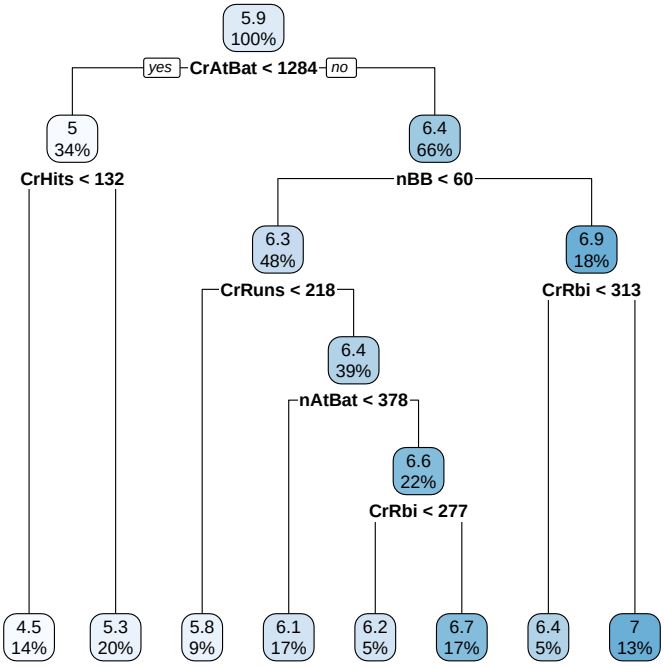
Regression Trees

```
# Fit a regression tree model on the baseball dataset with controls
baseball_rtree_control <- rpart::rpart.control(maxdepth = 15)
baseball_rtree <- rpart::rpart(logSalary ~ nAtBat + nHits + nHome + nRuns +
                                 nRBI + nBB + YrMajor + CrAtBat + CrHits +
                                 CrHome + CrRuns + CrRbi + CrBB + League +
                                 Division + nOuts + nAssts + nError,
                               data = baseball, control = baseball_rtree_control)

# Look at cost-complexity analysis plot (can help you decide on amount
# of pruning in your tree)
rpart::plotcp(baseball_rtree)
```



Based upon the above plot, we would likely choose a value of $cp = 0.03$, but we could also go with $cp = 0.022$. Using this value, we can retrain our regression tree model but with more pruning (specified with cp).

```
# Specify a cp=0.03 value for this regression tree model based upon the above plot.
baseball_final_rtree <-rpart::rpart(logSalary ~ nAtBat + nHits + nHome + nRuns +
                                      nRBI + nBB + YrMajor + CrAtBat + CrHits +
                                      CrHome + CrRuns + CrRbi + CrBB + League +
                                      Division + nOuts + nAssts + nError,
                                    data = baseball, cp = 0.03,
                                    control = baseball_rtree_control)
rpart.plot::rpart.plot(baseball_final_rtree)
```



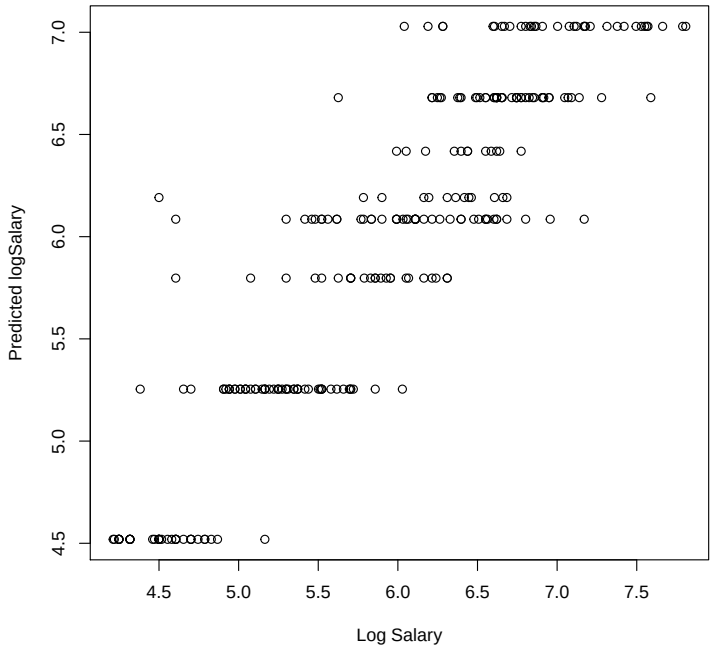
We can also look at the importance of various predictor variables. Note that in the following plot, the higher numbers correspond to more important variables.

```
baseball_final_rtree$variable.importance
```

```
##    CrAtBat     CrHits     CrRuns      CrRbi       CrBB    YrMajor        nBB
## 146.324332 141.326029 135.892077 124.933828 114.069502  81.776124  15.722008
##     CrHome      nRuns     nAtBat      nHits       nRBI      nOuts      nHome
##   9.306841   8.082672   5.981392   5.437629   4.078222   2.446933   1.106655
```

Check out this plot of the known value of logSalary and predicted logSalary:

```
pred_baseball_logSalary <- predict(baseball_final_rtree, baseball)
plot(baseball$logSalary, pred_baseball_logSalary, xlab = "Log Salary",
     ylab = "Predicted logSalary")
```



Questions:

1. What is going on in this plot? Do these patterns in the prediction make sense? If yes, why do they make sense?

2. Recalling output in handout 4.1.1, what do the "important" variables have in common?

Random Forests

```r
# Set a seed for reproducibility
set.seed(10984)

# For the random forest to be trained correctly, we must handle the 118
# missing values in the baseball dataset. Correctly handling NA values is an
# incredibly deep topic, but for the sake of simplicity we will simply just
# remove all rows from the baseball dataframe that have missing values.
baseball <- na.omit(baseball)

# Train a random forest model
baseball_rf <-
  randomForest::randomForest(logSalary ~ nAtBat + nHits + nHome + nRuns + nRBI +
                               nBB + YrMajor + CrAtBat + CrHits + CrHome +
                               CrRuns + CrRbi + CrBB + League + Division +
                               nOuts + nAssts + nError, data = baseball)

# Check out a variable importance plot of the random forest:
randomForest::varImpPlot(baseball_rf)
```