Baseball Simulation Analysis

Alex Lampert

January 14, 2020

Description: A simple analysis of Hardball Dynasty - an online baseball simulation. The game generates fictional players that have pitching, hitting, and fielding ratings, and tracks their statistics over their careers. The user takes over as general manager of a franchise, playing against other online users. The motivation behind this analysis is to try to (1) uncover which ratings are most important, and (2) project OPS for hitters and ERA for pitchers (these are overall measures of performance) using OLS regression.

The models are straightforward, and as follows:

$$ERA = X_1\beta_1 + \epsilon_1$$
$$OPS = X_2\beta_2 + \epsilon_2$$

Where X_1 is the design matrix of pitcher ratings (X_2 for hitters), β_1 and β_2 is estimated by minimizing the RSS, and ϵ_1 and ϵ_2 are assumed to follow some distribution with mean zero. It is assumed that ERA and OPS are linear in their respective covariates. These are the only necessary assumptions for OLS.

The decision to use OLS regression is not just for simplicity, but based on the belief that the simulation does not do anything mechanically that would be non-linear in nature. For instance, I strongly believe the programmers of the game designed ratings such that a "power" rating of 50 is truly twice the power abilithy of someone with a 25. Thus, I believe the true underlying model that I'm trying to estimate is linear, and probably additive (however, it's possible that there should be interaction terms). Because pulling data from the game is extremely tedious, I only have about 400 observations, and don't believe I have the power to accurately estimate interaction terms.

I first analyze the pitchers. As part of the exploratory phase, I create indicator variables for the varying levels of pitching ratings.

```r
library(MASS)
library(car)
```

```
## Loading required package: carData
```

```r
#################
### PITCHERS
#################

pitchers = read.csv("C:/Users/lamps/Documents/HBD/data/HBD_data_pitchers.csv")
options(contrasts = c("contr.treatment", "contr.poly"))

### Data Prep
# create flags for the different levels of pitching ratings
attach(pitchers)
pitchers$p1_61_70 = ifelse(p1>61 & p1<70,1,0)
pitchers$p1_71_80 = ifelse(p1>71 & p1<80,1,0)
pitchers$p1_81_90 = ifelse(p1>81 & p1<90,1,0)
pitchers$p1_91_100 = ifelse(p1>91 & p1<100,1,0)

pitchers$p2_0 = ifelse(p2==0,1,0)
pitchers$p2_1_10 = ifelse(p2>1 & p2<10,1,0)
pitchers$p2_11_20 = ifelse(p2>11 & p2<20,1,0)
pitchers$p2_21_30 = ifelse(p2>21 & p2<30,1,0)
```

```r
pitchers$p2_31_40 = ifelse(p2>31 & p2<40,1,0)
pitchers$p2_41_50 = ifelse(p2>41 & p2<50,1,0)
pitchers$p2_51_60 = ifelse(p2>51 & p2<60,1,0)
pitchers$p2_61_70 = ifelse(p2>61 & p2<70,1,0)
pitchers$p2_71_80 = ifelse(p2>71 & p2<80,1,0)
pitchers$p2_81_90 = ifelse(p2>81 & p2<90,1,0)
pitchers$p2_91_100 = ifelse(p2>91 & p2<100,1,0)

pitchers$p3_0 = ifelse(p3==0,1,0)
pitchers$p3_1_10 = ifelse(p3>1 & p3<10,1,0)
pitchers$p3_11_20 = ifelse(p3>11 & p3<20,1,0)
pitchers$p3_21_30 = ifelse(p3>21 & p3<30,1,0)
pitchers$p3_31_40 = ifelse(p3>31 & p3<40,1,0)
pitchers$p3_41_50 = ifelse(p3>41 & p3<50,1,0)
pitchers$p3_51_60 = ifelse(p3>51 & p3<60,1,0)
pitchers$p3_61_70 = ifelse(p3>61 & p3<70,1,0)
pitchers$p3_71_80 = ifelse(p3>71 & p3<80,1,0)
pitchers$p3_81_90 = ifelse(p3>81 & p3<90,1,0)
pitchers$p3_91_100 = ifelse(p3>91 & p3<100,1,0)

pitchers$p4_0 = ifelse(p4==0,1,0)
pitchers$p4_1_10 = ifelse(p4>1 & p4<10,1,0)
pitchers$p4_11_20 = ifelse(p4>11 & p4<20,1,0)
pitchers$p4_21_30 = ifelse(p4>21 & p4<30,1,0)
pitchers$p4_31_40 = ifelse(p4>31 & p4<40,1,0)
pitchers$p4_41_50 = ifelse(p4>41 & p4<50,1,0)
pitchers$p4_51_60 = ifelse(p4>51 & p4<60,1,0)
pitchers$p4_61_70 = ifelse(p4>61 & p4<70,1,0)
pitchers$p4_71_80 = ifelse(p4>71 & p4<80,1,0)
pitchers$p4_81_90 = ifelse(p4>81 & p4<90,1,0)
pitchers$p4_91_100 = ifelse(p4>91 & p4<100,1,0)

pitchers$p5_0 = ifelse(p5==0,1,0)
pitchers$p5_1_10 = ifelse(p5>1 & p5<10,1,0)
pitchers$p5_11_20 = ifelse(p5>11 & p5<20,1,0)
pitchers$p5_21_30 = ifelse(p5>21 & p5<30,1,0)
pitchers$p5_31_40 = ifelse(p5>31 & p5<40,1,0)
pitchers$p5_41_50 = ifelse(p5>41 & p5<50,1,0)
pitchers$p5_51_60 = ifelse(p5>51 & p5<60,1,0)
pitchers$p5_61_70 = ifelse(p5>61 & p5<70,1,0)
pitchers$p5_71_80 = ifelse(p5>71 & p5<80,1,0)
pitchers$p5_81_90 = ifelse(p5>81 & p5<90,1,0)
pitchers$p5_91_100 = ifelse(p5>91 & p5<100,1,0)

# create park factors (to account for playing in a "pitchers park" or a "hitters park")
pitchers$hr = pitchers$hrlf + pitchers$hrrf
pitchers$hits = pitchers$singles + pitchers$doubles + pitchers$triples
```

Now I explore some models. Because each statline has a varying degree of innings pitched attached to it, I weight the model by number of outs recorded. For example, a pitcher with 10 innings pitched (30 outs) holds less weight than a pitcher with 200 innings pitched.

```r
# Full model, weighted
m1 = lm(era ~ ctrl + vl + vr + vel + grb + throw + p1 + p2 + p3 + p4 + p5 + singles
        + doubles + triples + hrlf + hrrf, data=pitchers, weight=outs)

# Break the pitches into buckets
m2 = lm(era ~ ctrl + vl + vr + vel + grb + throw + p1_61_70 + p1_71_80
        + p1_81_90 + p1_91_100 + p2_41_50 +  p2_51_60 + p2_61_70 + p2_71_80 + p2_81_90
        + p2_91_100 + p3_0 + p3_1_10 + p3_11_20 + p3_21_30 + p3_31_40 + p3_41_50 +
          p3_51_60 + p3_61_70+p3_71_80+p3_81_90 + p4_0 + p4_1_10 + p4_11_20 + p4_21_30
        + p4_31_40 + p4_41_50 + p4_51_60 + p4_61_70+p4_71_80+p4_81_90 + p5_0 + p5_1_10
        + p5_11_20 + p5_21_30 + p5_31_40 + p5_41_50 + p5_51_60 + p5_61_70 + singles +
          doubles + triples + hrlf + hrrf, data=pitchers, weight=outs)

# Simplify the park factor, remove pitch buckets
m3 = lm(era ~ ctrl + vl + vr + vel + grb + throw + p1 + p2 + p3 + p4 + p5 + hits + hr
        , data=pitchers, weight=outs)

# Remove park factor
m4 = lm(era ~ ctrl + vl + vr + vel + grb + throw + p1 + p2 + p3 + p4 + p5,
        ata=pitchers, weight=outs)
```

```
## Warning: In lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok,
##     ...) :
##  extra argument 'ata' will be disregarded
```

```r
# Interact velocity and groundball based on suspicion that there exists interaction effect
m5 = lm(era ~ ctrl + vl + vr + vel*grb + throw + p1 + p2 + p3 + p4 + p5,
        data=pitchers, weight=outs)
```

I check VIF, variance inflation factor, to determine whether any of the variables are highly multi-collinear. HRlf, and HRrf are unsurprisingly highly multi-collinear. It is likely that they are highly correlated with each other. The rule of thumb is that anything above 10 warrants removal from the model.

```r
car::vif(m1)
```

```
##     ctrl       vl       vr      vel      grb    throw       p1       p2
## 1.127224 1.996408 2.144061 1.126489 1.080964 2.118277 1.201410 1.148625
##       p3       p4       p5  singles  doubles  triples     hrlf     hrrf
## 1.534462 1.926883 1.464251 2.779794 5.381646 2.545292 7.409926 9.748718
```

Lastly I save the predictions (fitted values) from each model, and export them to Excel. Once in Excel, I average the predictions from the four models to give me the final predictions that I use to evaluate true skill. While this is somewhat quick and dirty, it gives me a competitive advantage over other GMs in that I don't have to rely on staring at player ratings to determine true skill. I can also use the model coefficients to predict the value of future players (note that these are technically "predictions", while the others were "fitted values").

As a reminder, the values that I'm predicting are ERA - earned run average, which measures the average number of earned runs a pitcher allows per nine innings. In other words, it's a measure of pitching performance.

```r
m1_preds = predict(m1)
m2_preds = predict(m2)
m3_preds = predict(m3)
m4_preds = predict(m4)

outdata = pitchers
outdata$m1 = m1_preds
outdata$m2 = m2_preds
outdata$m3 = m3_preds
```

```
outdata$m4 = m4_preds

#write.csv(outdata, "C:/Users/lamps/Documents/HBD/Results/HBD_pitcher_out.csv")
```

I next analyze the hitters. OPS - on base plus slugging percentage, is a widely used measure of hitter skill, somewhat analagous to ERA. I regress hitter ratings against OPS, similar to how I did for pitchers.

```
library(MASS)
library(car)
#library(xlsx)

###############
### HITTERS
###############

# read data
hitters = read.csv("C:/Users/lamps/Documents/HBD/data/HBD_data_hitters.csv")
attach(hitters)
```

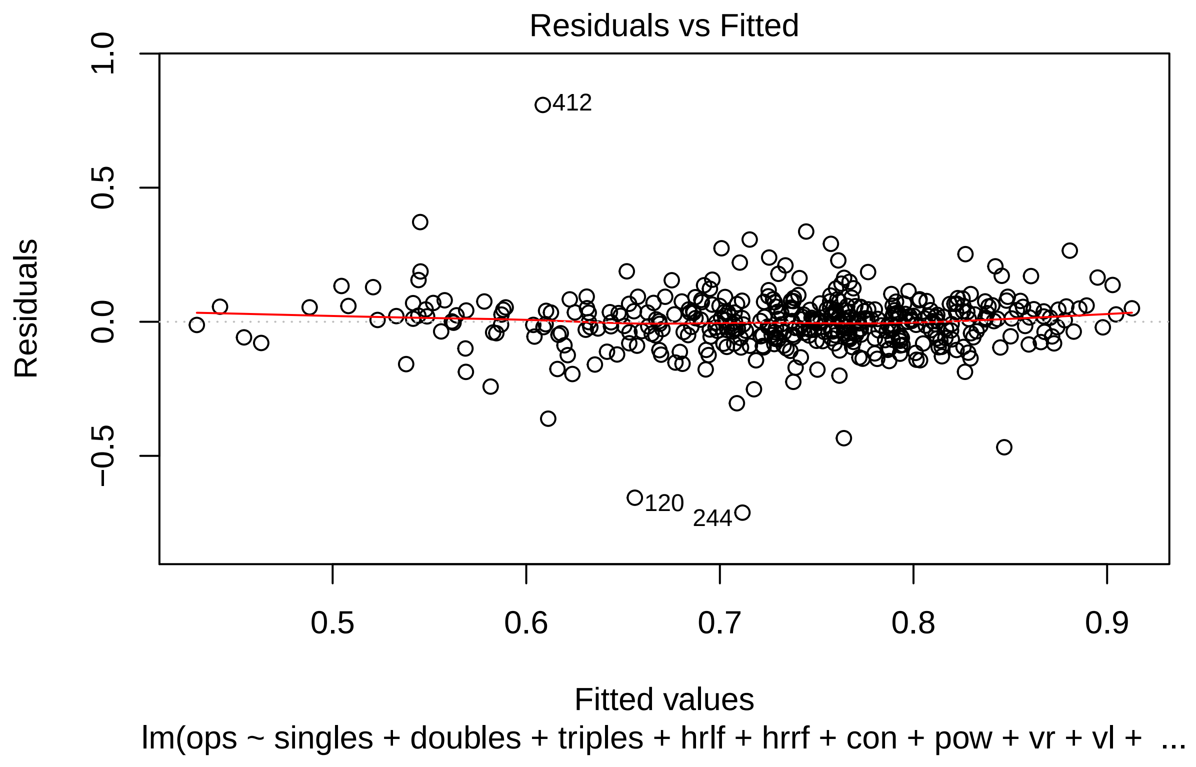
```
## The following objects are masked from pitchers:
##
##     age, bat, doubles, dur, frn, health, hrlf, hrrf, ht, name,
##     pos, sal, singles, throw, triples, vl, vr, wt
```

```
options(contrasts = c("contr.treatment", "contr.poly"))

#Full model, weighted
m.full.wgtd = lm(ops ~ singles + doubles + triples + hrlf + hrrf + con + pow + vr + vl
                 + eye + base + spd + bat, data=hitters, weight=ab)
```

A quick check of the residuals vs fitted confirms that the linear model fit is okay, and that the assumption of constant variance is met in order to interpret the coefficient standard errors.

```
plot(m.full.wgtd, which = c(1,1))
```



```
car::vif(m.full.wgtd)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## singles 2.694457  1        1.641480
## doubles 5.438157  1        2.331986
## triples 2.687840  1        1.639463
## hrlf    7.235065  1        2.689808
## hrrf    9.206578  1        3.034234
## con     1.164312  1        1.079033
## pow     1.291412  1        1.136403
## vr      1.408485  1        1.186796
## vl      1.397116  1        1.181996
## eye     1.181401  1        1.086923
## base    1.296706  1        1.138730
## spd     1.311508  1        1.145211
## bat     1.144717  2        1.034367
```

```
summary(m.full.wgtd)
```

```
##
## Call:
## lm(formula = ops ~ singles + doubles + triples + hrlf + hrrf +
##     con + pow + vr + vl + eye + base + spd + bat, data = hitters,
##     weights = ab)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0458 -0.8023  0.0369  0.8342  3.9252
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2783962  0.0307980   9.039  < 2e-16 ***
## singles      0.0058613  0.0035729   1.641  0.10166
## doubles     -0.0027169  0.0048102  -0.565  0.57250
## triples     -0.0021231  0.0031805  -0.668  0.50479
## hrlf        -0.0049947  0.0050497  -0.989  0.32319
## hrrf         0.0086660  0.0057536   1.506  0.13278
## con          0.0011944  0.0002248   5.313 1.76e-07 ***
## pow          0.0024402  0.0001923  12.689  < 2e-16 ***
## vr           0.0015794  0.0002751   5.742 1.82e-08 ***
## vl           0.0004793  0.0002594   1.848  0.06532 .
## eye          0.0013557  0.0002556   5.304 1.84e-07 ***
## base        -0.0001414  0.0002448  -0.578  0.56384
## spd          0.0004574  0.0001681   2.721  0.00678 **
## batR        -0.0133302  0.0092711  -1.438  0.15124
## batS        -0.0095646  0.0119397  -0.801  0.42355
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.221 on 414 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5407
## F-statistic: 36.99 on 14 and 414 DF,  p-value: < 2.2e-16
```

```r
full_preds = predict(m.full.wgtd)
```

Here, I decide to use stepwise model selection with AIC as the criteria. AIC is asymptotically efficient, meaning that it is a good criteria to use when prediction accuracy is the goal. BIC, on the other hand, often results in a sparser model that and is therefore good for variable selection. R begins with the full model and sequentially removes variables until a local minimum (possibly global minimum) AIC occurs. A summary of the selected model is given.

```r
hitters = read.csv("C:/Users/lamps/Documents/HBD/data/HBD_data_hitters.csv")
attach(hitters)
```

```
## The following objects are masked from hitters (pos = 3):
##
##     ab, acc, age, arm, avg, base, bat, con, doubles, dur, eye,
##     fld, frn, glove, health, hr, hrlf, hrrf, ht, name, obp, ops,
##     park, pc, pct, pos, pow, rbi, rf, rng, sal, sb, singles, slg,
##     spd, throw, triples, vl, vr, wt

## The following objects are masked from pitchers:
##
##     age, bat, doubles, dur, frn, health, hrlf, hrrf, ht, name,
##     pos, sal, singles, throw, triples, vl, vr, wt
```

```r
#Select a model
step.model = stepAIC(m.full.wgtd, direction = "both", trace = FALSE)
summary(step.model)
```

```
##
## Call:
## lm(formula = ops ~ hrrf + con + pow + vr + vl + eye + spd, data = hitters,
##     weights = ab)
```

```
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3192 -0.8126  0.0464  0.8286  4.4910
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.2572857  0.0264818   9.716  < 2e-16 ***
## hrrf        0.0046360  0.0019199   2.415  0.01617 *
## con         0.0011701  0.0002218   5.277 2.11e-07 ***
## pow         0.0024841  0.0001894  13.114  < 2e-16 ***
## vr          0.0016343  0.0002657   6.151 1.80e-09 ***
## vl          0.0004226  0.0002493   1.695  0.09078 .
## eye         0.0014006  0.0002531   5.534 5.51e-08 ***
## spd         0.0004373  0.0001556   2.810  0.00518 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.22 on 421 degrees of freedom
## Multiple R-squared:  0.5487, Adjusted R-squared:  0.5412
## F-statistic: 73.12 on 7 and 421 DF,  p-value: < 2.2e-16
```

```
AIC_preds = predict(step.model)
```

The model with the lowest AIC removes many of the park factors, along with the handedness of the batter (batR, batS).

```
#outdata = mydata
#outdata$aic_preds = AIC_preds
#outdata$full_preds = full_preds
#write.csv(outdata, "C:/Users/lamps/Documents/HBD/HBD_hitters_out.csv")
```

I lastly attempt a model without stadium effects, and include all of the ratings that I deem important through personal experience.

```
#Model without stadium effects
m1 = lm(ops ~ con + pow + vr + vl + eye + spd + bat, data=hitters, weight=ab)
preds = predict(m1)
outdata = hitters
outdata$m1 = preds
#write.csv(outdata, "C:/Users/lamps/Documents/HBD/HBD_hitters_out.csv")
```