Pitcher prediction using the Lahman database for baseball

Michael Lopez, Skidmore College

Overview

In this lab, we'll gain more experience with the Lahman package, while also learning model comparison tools using multivariate regression. Finally, we'll apply our tools to derive predictions of pitcher performance.

First, recall that we have to load the requisite libraries that we'll need (and we may have to install them, too. As always, once a package is downloaded, you do not need to run the `install.packages()` code again.

```
library(Lahman)
library(mosaic)
```

We're going to start by using the `Teams` data.

```
data(Teams)
head(Teams)
tail(Teams)
```

The `Teams` data contains team-level information for every year of baseball season, from 1871 - 2014. Let's reduce this data frame to only recent seasons. *Note*: The `filter()` command comes from the `dplyr` package, which is one of `R`'s best packages for data manipulation. It loads automatically with `mosaic`, and you can read more by going to the tutorial here.

```
Teams.1 <- filter(Teams, yearID >= 1970)
head(Teams.1)
```

Comparing multiple regression models.

There's an old saying in statistics, attributed to George Box: *all models are wrong, some are useful.* In practice, we never know if our regression model is correctly specified; e.g, that it is really the case the y, x_1, ... and x_{p-1} are linearly related. All we can do is hope...and try a few analytical tools.

Let's try to come up with a few models of `RA`: runs against. First, we start with a recap of multiple regression.

```
fit.1 <- lm(RA ~ HRA + BBA + SOA + HA, data = Teams.1)
msummary(fit.1)
```

1. Write the estimated model above.
2. Using the model in question (1), interpret the coefficient on `HRA`.
3. Note that this coefficient estimate is noticeably different than the one found on the identical data in Lecture 3 ($\hat{\beta}_1 = 2.00$). Can you explain the difference?

After fitting a linear regression model, it is appropriate to check assumptions. First, we check the appropriateness of the normal distribution for residuals.

```
qqnorm(fit.1$resid)
qqline(fit.1$residuals)
```

Next, we compare the residuals to the fitted values, checking for the assumptions of independence among the residuals, as well as the constant variance assumption.

```
xyplot(fit.1$resid ~ fit.1$fitted)
```

4. What do the residual plots suggest about the assumptions of our linear regression model? What about the model makes it possibly a poor fit?

Speaking of residuals, lets take a deeper look at individual predictions.

The 1970 Atlanta Braves allowed 185 home runs, 478 walks, struck out 960 batters, and gave up 1451 hits. As it turns out, the Braves are the first row of our data set.

```
Teams.1[1,]
```

5. Calculate how many runs our model predicts that Braves to have scored. What is the residual for the number of runs allowed by the Braves? Did our model overestimate or underestimate Atlanta's performance?

R-squared

There are lots of ways to measure the success of a regression model. The most common metric is `R-squared`, which you'll recall is the fraction of variability in the outcome which is explained by the regression model. Larger R-squared's are, in principal, better. Using our model, we would interpret the R-squared as follows:

90% of the variability in the number of runs allowed by a team can be explained by the linear model with HRA, BBA, SOA, and E..

While popular, the traditional R-squared is also flawed. Let's see how. In the following code, we'll create two new variables in the `Teams.1` data set, `rand1` and `rand2`, which are random normal variables.

```
set.seed(0)
Teams.1 <- mutate(Teams.1,
                  rand1 = rnorm(nrow(Teams.1)),
                  rand2 = rnorm(nrow(Teams.1)))
head(select(Teams.1, yearID, teamID, rand1, rand2))
```

Let's see what happens when we include `rand1` and `rand2` to our regression fit.

```
fit.2 <- lm(RA ~ HRA + BBA + SOA + HA + rand1 + rand2, data = Teams.1)
msummary(fit.2)
```

Even when we added random noise to the model, R-squared went up!

That's not a good thing, at least when it comes to making model comparisons. In fact, its a property of R-squared that, no matter what variable you add to a given model, the R-squared cannot go down. As a result, R-squared is not useful for model comparisons, but more to gain a sense of how much of a drop in the variability in the outcome can be explained by the model's fit.

In place of R-squared, R also shows a formula for an adjusted R-squared, which penalizes models for adding unneeded parameters. However, this metric also has weaknesses.

Akaike information criterion

One method of comparing several regression models simultaneously (as to pick the one or two best fits) is the `Akaike information criterion`, which is a function of the model fit and its number of parameters. Read more here.

Lower AIC values are better - and adding useless variables to the model serves only to increase the AIC (in general, at least).

The AIC cam be easily calculated in R using the `AIC()` command. For example,

```
AIC(fit.1)
```

As you notice, one downside of AIC is that there is no natural scale with which to make comparisons. However, it is useful to compare several models at once.

6. Which model is preferred, and why? Compare `fit.1` to `fit.2` using the AIC criterion.

Player prediction.

Based on the models fit today and Tuesday, it is obvious that on a team-level, including hits allowed (`HA`) significantly improves the link to runs allowed. As evidence, the R-squared values jumped from about 79% to 90% when including hits (you can either confirm this yourself, or refer to the notes from Tuesday's class). Additionally, the coefficient on `HA` is quite significant.

Can including hits improve our prediction of pitchers? Let's find out.

This first code creates new variables, including non home-run hits (nonHRhits), our old FIP formula, and a new FIP formula.

```
data(Pitching)
Pitchers.1 <- filter(Pitching, yearID >= 2000, IPouts > 500)
Pitchers.1 <- mutate(Pitchers.1,
   nonHRhits = H - HR,
   FIP.org = ((13*HR) + 3*(BB + HBP) - 2*SO)/(IPouts),
   FIP.new = ((13*HR) + 5*nonHRhits  + 3*(BB + HBP) - 2*SO)/(IPouts))
Pitchers.1[3,]
```

Note that as one guess, I valued `nonHRhits` as worth more than walks and hit by pitches, but less than home runs. Our goal, at this point, is to identify whether or not `FIP.new` can better predict future performance, relative to `FIP.old`.

Next, we create a set of new variables, for each pitcher's metrics in the following season.

```
Pitchers.2 <- Pitchers.1 %>%
  arrange(playerID, yearID) %>%
  group_by(playerID) %>%
  mutate(f.ERA = lead(ERA), f.FIP.org = lead(FIP.org), f.FIP.new = lead(FIP.new))
```

Finally, let's look at the correlations between our advanced methods and the more traditional method, ERA.

```
library(corrplot)
cor.matrix <- cor(select(ungroup(Pitchers.2),
    f.ERA, f.FIP.org, f.FIP.new, ERA, FIP.org, FIP.new),
    use="pairwise.complete.obs")
corrplot(cor.matrix, method = "number")
```

7. In terms of predicting era in the following season (`f.ERA`), which metric is best? Why is it inappropriate to use answer with one of the other future variables (`f.FIP.org` or `f.FIP.new`)?

8. Which of the three metrics (ERA, FIP.org, FIP.new) is most closely linked to its own calculation in the following season? Recall, this is relevant - the more repeatable metrics are more the better.

9. Play around with the code below, and come up with your own formulas for `FIP.new`. Your goal should be to outperform `ERA` and `FIP.org` at predicting ERA in the following season. Feel free to change around the weights for each of the current variables, or to try inclusion of other variables, including wild pitches (`WP`), intentional walks (`IBB`), saves (`SV`), etc.

```
Pitchers.1 <- filter(Pitching, yearID >= 2000, IPouts > 50)
Pitchers.1 <- mutate(Pitchers.1,
   nonHRhits = H - HR,
   FIP.org = ((13*HR) + 3*(BB + HBP) - 2*SO)/(IPouts),
   FIP.new = ((13*HR) + 5*nonHRhits  + 3*(BB + HBP) - 2*SO)/(IPouts))

Pitchers.2 <- Pitchers.1 %>%
  arrange(playerID, yearID) %>%
  group_by(playerID) %>%
  mutate(f.ERA = lead(ERA), f.FIP.org = lead(FIP.org), f.FIP.new = lead(FIP.new))

cor.matrix <- cor(select(ungroup(Pitchers.2),
    f.ERA, f.FIP.org, f.FIP.new, ERA, FIP.org, FIP.new),
    use="pairwise.complete.obs")
corrplot(cor.matrix, method = "number")
```

10. Using the code above, play around with the outs cutoff (`IPouts`) of 500, which is shown in the first line. What happens to the correlations when using more of fewer outs? Are there any difference in these changes between the within-year and between year comparisons?