Predicting Games Played

Kyle Joecken

Injuries: The bane of any promising fantasy season.

Question

Can we use games played data from the previous few National Hockey League (NHL) seasons to predict how many games a particular skater (*i.e.*, non-goalie) will play in an upcoming season?

In particular, the hope is that games played for various players in their more recently played seasons will help to predict how many games they'll play in future seasons. This is quite difficult, as the data are exceedingly noisy due to the largely random nature of injuries in a fast-paced, full-contact sport. Surely there is some small signal for which we can search.

Data

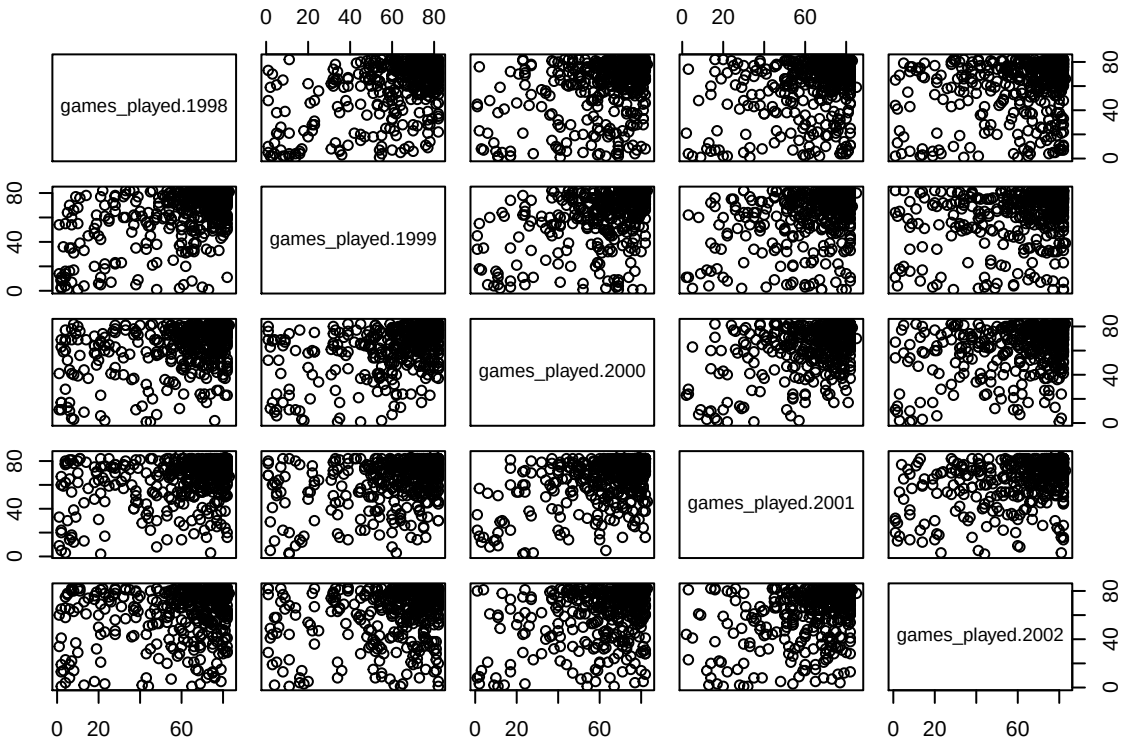
First, we load the data from our local (scraped) database of season-wide NHL data. It is stored in a data frame called `skaterstats`.

Features

Next, we strip out all the unnecessary variables and reshape the data frame, so that each player (identified by their `nhl_num`, or the unique 7-digit number assigned to them by the NHL's website) is a unique observation. Each observation will simply be a list of games played in various seasons. This will leave us with a column for each player called `games_played.####`, where the `####` is a season between 1998 and 2014. We'll also scale up the 2013 season (in which only 48 games were played) by multiplying it by the factor 82/48.

To get some idea of how the data relate to each other, we can plot each variable against each other using the function `pairs()`. Looking at every single season against the others would be too many, so to get a glimpse of the data we first restrict to players that have played in each of the seasons from 1998 to 2002, then plot only those 5 against each other.

```
skatergames <- skaterstats[, c(1, 2, 6)]
skatergames <- reshape(skatergames, timevar = "season",
                       idvar = "nhl_num", direction = "wide")
skatergames$games_played.2013 <- skatergames$games_played.2013 * 82 / 48
gp9802 <- skatergames[!is.na(skatergames$games_played.1998) &
                            !is.na(skatergames$games_played.1999) &
                            !is.na(skatergames$games_played.2000) &
                            !is.na(skatergames$games_played.2001) &
                            !is.na(skatergames$games_played.2002), 1:6]
pairs(gp9802[, 2:6])
```



Now we will fit a few linear regressions, adding progressively older seasons' data as new regressors. We can then run an analysis of variance to see if we are still getting any gains from each successively older season.

```
fit1 <- lm(games_played.2002 ~ games_played.2001, data = gp9802)
fit2 <- lm(games_played.2002 ~ games_played.2001 + games_played.2000, data = gp9802)
fit3 <- lm(games_played.2002 ~ games_played.2001 + games_played.2000 +
                 games_played.1999, data = gp9802)
fit4 <- lm(games_played.2002 ~ games_played.2001 + games_played.2000 +
                 games_played.1999 + games_played.1998, data = gp9802)
anova(fit1, fit2, fit3, fit4)
```

```
## Analysis of Variance Table
##
## Model 1: games_played.2002 ~ games_played.2001
## Model 2: games_played.2002 ~ games_played.2001 + games_played.2000
## Model 3: games_played.2002 ~ games_played.2001 + games_played.2000 + games_played.1999
## Model 4: games_played.2002 ~ games_played.2001 + games_played.2000 + games_played.1999 +
##     games_played.1998
##   Res.Df    RSS Df Sum of Sq     F Pr(>F)
## 1    433 160377
## 2    432 154076  1      6301 17.80  3e-05 ***
## 3    431 152368  1      1707  4.82  0.029 *
## 4    430 152193  1       175  0.49  0.482
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With the low p-value for `fit4` and the realtively low p- and F-values for `fit3`, it seems like `fit2` might be the most reasonable choice to go with of these options. Thus we ignore all but the previous two seasons.

Algorithm

We now try to find a *usual* and simple set of coefficients across various regressions of the type `lm(gp.n ~ gp.n-1 + gp.n-2)`, saving a prediction for `games_played.2014` as a test case. We will collect the coefficients in a matrix called `coeffs` and have a look when finished.

```
##       baseline season n-1 season n-2
## 98-00    20.33     0.4960    0.14369
## 99-01    25.08     0.4825    0.10268
## 00-02    24.01     0.4379    0.16981
## 01-03    21.93     0.4346    0.17992
## 02-04    24.29     0.3111    0.26824
## 03-06    25.09     0.4133    0.17695
## 04-07    22.82     0.4950    0.12400
## 06-08    22.90     0.4095    0.19528
## 07-09    28.32     0.4433    0.09661
## 08-10    30.36     0.4099    0.10393
## 09-11    26.90     0.3810    0.16585
## 10-12    15.83     0.5943    0.12050
## 11-13    23.33     0.4753    0.12609
## means    23.94     0.4449    0.15181
```

Letting $x =$ `games_played.n`, $y =$ `games_played.n-1` and $z =$ `games.played.n-2`, we adopt the model $x \sim 23.9385 + 0.4449y + 0.1518z$.

Alternatively, we could rewrite this equation by replacing y and z with $\bar{y}$ and $\bar{z}$, the number of games *missed* in seasons `n-1` and `n-2`, respectively. This may be slightly more interpretable. Making this replacement and rounding the coefficients to more interpretable numbers, we get that

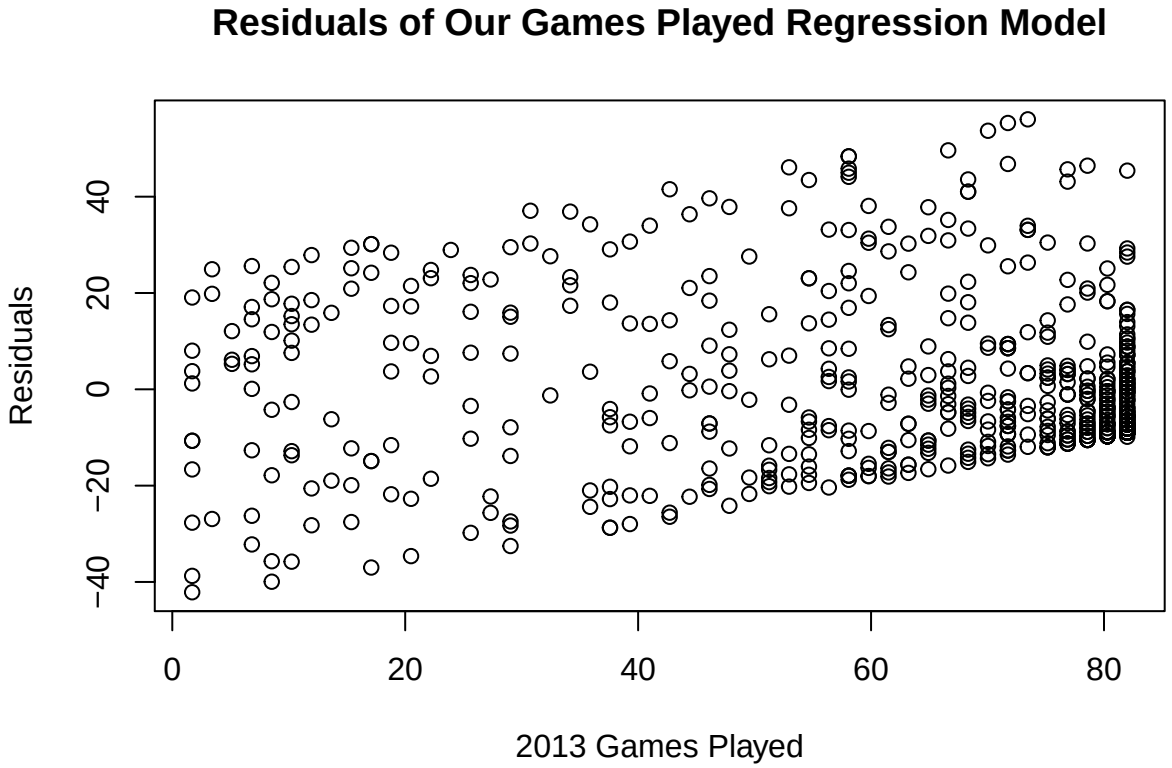
$$x \sim 73 - \frac{4}{9}\bar{y} - \frac{3}{20}\bar{z}.$$

Perhaps not surprisingly, these coefficients are not significantly far off from those achieved from another analysis done for NBA players, who also play an 82-game season. Their coefficients were 76, $\frac{1}{2}$ and $\frac{1}{6}$. I am searching for this source to include it; I saw it linked a few months ago by Nate Silver, but have not been able to find it again.

Evaluation

Let's use the model to "predict" games played in 2014 and compare it to reality to see how we did. We have already separated out the relevant data in the frame `gp1214`. We plot the residuals.

```
gp1214$gp.predicted <- 73 - 4/9 * (82 - gp1214$games_played.2013) -
      3/20 * (82 - gp1214$games_played.2014)
plot(I(gp.predicted - games_played.2014) ~ games_played.2013, data = gp1214,
     main = "Residuals of Our Games Played Regression Model",
     xlab = "2013 Games Played", ylab = "Residuals")
```



Obivously these data are massively noisy, and much variance remains unexplained. Rather than try to tease further through such a hideous data set, we'll use this admittedly specious model until we retrieve our game-by-game data; at that point, classifying games missed by healthy or unhealthy should go a long way toward zeroing in on some signal.