Introduction to Multilevel Models

2017-03-16

Preparation

We will use two datasets located in the *nbaLevel1.sav* file and the *nbaLevel2.sav* file. These data include player-level attributes for $n = 300$ NBA players, and team-level attributes for $N = 30$ different teams. The source of these data is: Woltman, Feldstein, MacKay, & Rocchi (2012). We will use these data to explore the question of whether how good a player is (`Shots_on_five`) predicts variation in life satisfaction.

The player-level attributes in the *nbaLevel1.sav* file include:

- `Team_ID`: The team ID number for each player
- `Shots_on_five`: A proxy for player quality/success. This indicates the number of successful shots (out of five taken). Higher values indicate a more succesful player.
- `Life_Satisfaction`: Score on a survey of life satisfaction. Scores range from 5 to 25, with higher scores indicating more life satisfaction.

The team-level attributes in the *nbaLevel2.sav* file include:

- `Team_ID`:The team ID number
- `Coach_Experience`: Years of coaching experience in the NBA

Both files have the extension *.sav*. This is the file saving extension SPSS appends to data files. To read a SAV file, we use the `read_sav()` function from the **haven** library. This function has similar syntax to the `read_csv()` we have been using. If you are using the `Import` button in RStudio, you can select `From SPSS...` Below we use this function to read in both datasets.

```
# Load haven library
library(haven)

# Read in player-level data
nbaL1 = read_sav(file = "~/Google Drive/Documents/EPsy-8252/data/nbaLevel1.sav")
head(nbaL1)
```

Team_ID	Shots_on_five	Life_Satisfaction
01	3	18.8
01	3	18.0
01	4	21.0
01	4	20.5
01	3	19.0
01	2	12.1

```
# Read in team-level data
nbaL2 = read_sav(file = "~/Google Drive/Documents/EPsy-8252/data/nbaLevel2.sav")
head(nbaL2)
```

Team_ID	Coach_Experience
01	2
02	3
03	2
04	2

Team_ID	Coach_Experience
05	1
06	2

```
# Load other libraries
library(AICcmodavg)
library(dplyr)
library(ggplot2)
library(sm)
```

Merge the Player- and Team-Level Data

Before analyzing the data, we need to merge the two datasets together. To do this, we will use the `left_join()` function from the **dplyr** package. **dplyr** includes six different join functions. You can read about several different join functions here.

```
nba = left_join(nbaL1, nbaL2, by = "Team_ID")
head(nba)
```

Team_ID	Shots_on_five	Life_Satisfaction	Coach_Experience
01	3	18.8	2
01	3	18.0	2
01	4	21.0	2
01	4	20.5	2
01	3	19.0	2
01	2	12.1	2

Conventional Linear Regression

To examine the research question using conventional linear regression, we would regress life satisfaction on the player success variable (and any potential covariates). For now, we will only focus on player-success.

Fit Linear Models

```
lm.1 = lm(Life_Satisfaction ~ 1 + Shots_on_five, data = nba)
summary(lm.1)
```

```
##
## Call:
## lm(formula = Life_Satisfaction ~ 1 + Shots_on_five, data = nba)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.943 -1.783  0.107  1.727  5.387
##
## Coefficients:
##               Estimate Std. Error t value            Pr(>|t|)
## (Intercept)      5.694      0.303    18.8 <0.0000000000000002 ***
## Shots_on_five    3.660      0.108    34.0 <0.0000000000000002 ***
```

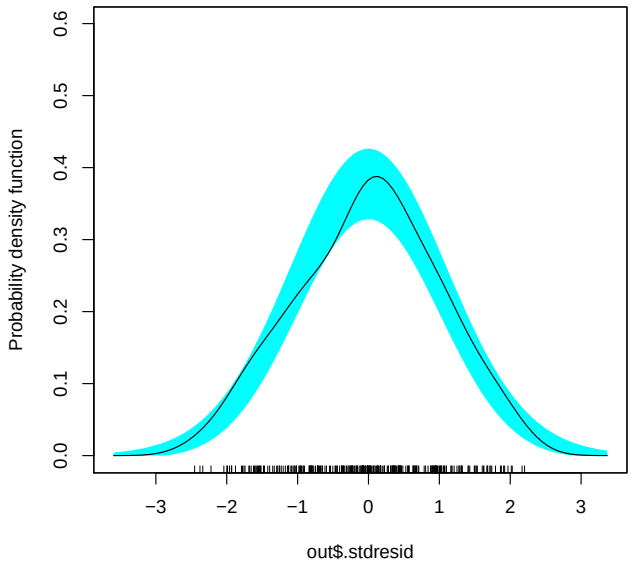
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.44 on 298 degrees of freedom
## Multiple R-squared:  0.795,  Adjusted R-squared:  0.795
## F-statistic: 1.16e+03 on 1 and 298 DF,  p-value: <0.0000000000000002
```

To have faith in the analytic results from this model, we need to evaluate whether the assumptions are satisfied.

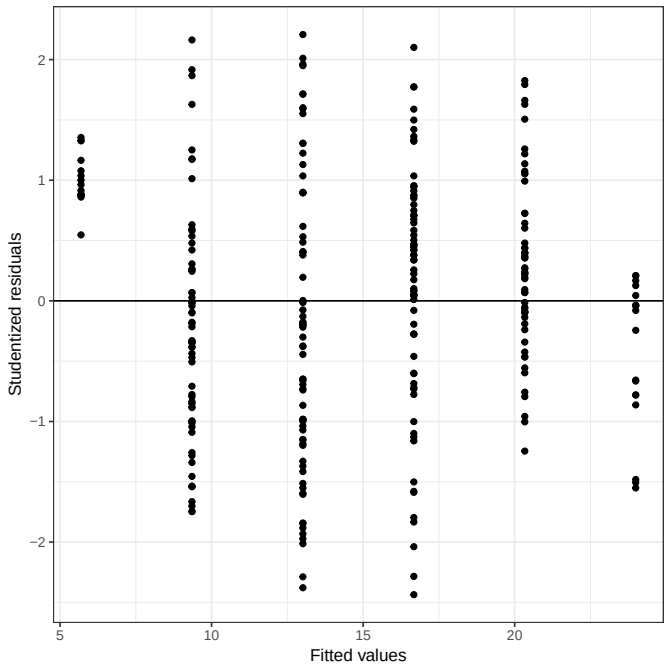
Examine Residuals

```
# Obtain the fortified data frame
out = fortify(lm.1)

# Normality
sm.density(out$.stdresid, model = "normal")
```




```
# All other assumptions
ggplot(data = out, aes(x = .fitted, y = .stdresid)) +
    geom_point() +
    geom_hline(yintercept = 0) +
    theme_bw() +
  xlab("Fitted values") +
  ylab("Studentized residuals")
```

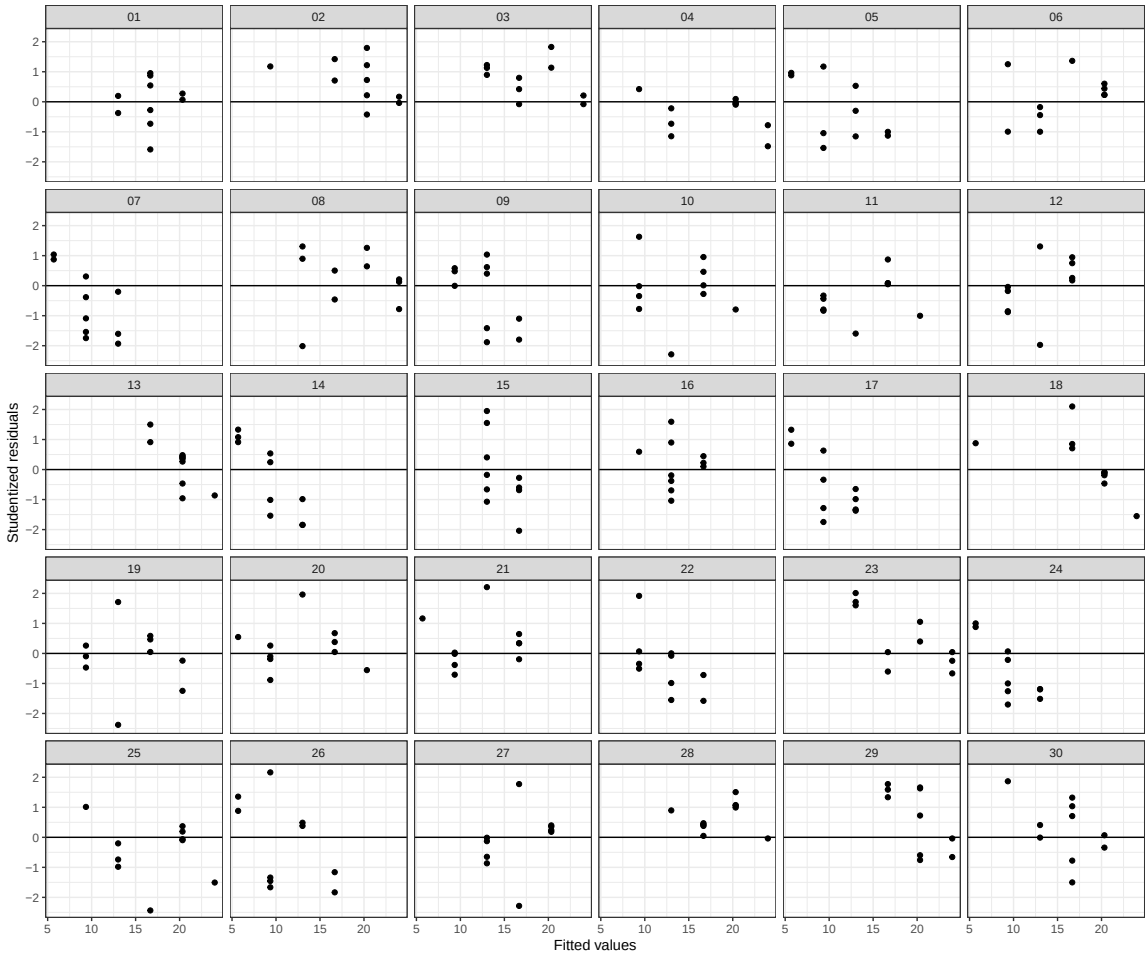


The assumptions of linearity, normality, and homoscedasticity seem reasonably satisfied. The assumption of independnece, however, is probably not tenable. The life satisfaction scores (and thus the residuals) are probably more correlated within teams than between teams—this is a violation of independence.

If we have a variable that indicates team, We can examine this by plotting the residuals separately for each team.

```
# Add Team_ID variable to fotified data
out$Team_ID = nba$Team_ID

### Show residuals by team
ggplot(data = out, aes(x = .fitted, y = .stdresid)) +
    geom_point() +
    geom_hline(yintercept = 0) +
    theme_bw() +
  xlab("Fitted values") +
  ylab("Studentized residuals") +
    facet_wrap(~Team_ID, nrow = 5)
```



The residuals are systematically over or under 0 within teams (e.g., Team 11, Team 28). This is a sign of non-independence of the residuals. To account for this we need to use a method that models the correlation among the residuals within teams. This is what multilevel models bring to the table. By correctly modeling the non-independence, we get more accurate standard errors.

Another benefit of using multilevel models is that we also get variation accounted for estimates at both the team- and player-levels. This disaggregating of the variation allows us to see which level is explaining more variation and to study predictors appropriate to explaining that variation. For example, suppose you found that of the variation in student achievement scores, 96% was at the student-level, and 3% was at the classroom-level, and 1% was at the school-level. By including school-level predictors in the model, you would only be "chipping away" at that 1%. You should focus your attention on student-level predictors!

Conceptual Idea of Multilevel Modeling

In this section we will outline the conceptual ideas behind multilevel modeling using conventional regression. It is important to realize that this is just conceptual in nature. Its purpose is only to help you understand the output you get from a multilevel model.

To begin, we remind you of the fitted regression equation we obtained earlier, when we regressed life stais-faction on player success for the $n = 300$ players:

$$\hat{\text{Life Satisfaction}}_i = 5.70 + 3.66(\text{Player Success}_i)$$

Multilevel modeling actually fits a global model (like the one above) AND a team-specific model for each team. Conceptually this is like fitting a regression model for each team separately. Below I will do this, but keep in mind that this is only to help you understand.

```
models = nba %>%
  group_by(Team_ID) %>%
  do( mod = lm(Life_Satisfaction  ~ Shots_on_five, data = .) ) %>%
  broom::tidy(mod)

models
```

```
## Source: local data frame [60 x 6]
## Groups: Team_ID [30]
##
##    Team_ID          term estimate std.error statistic   p.value
##      <chr>         <chr>    <dbl>     <dbl>     <dbl>     <dbl>
## 1       01   (Intercept)     4.73     3.041      1.55 0.1587542
## 2       01 Shots_on_five     3.98     0.992      4.01 0.0038931
## 3       02   (Intercept)     9.96     1.803      5.52 0.0005595
## 4       02 Shots_on_five     2.97     0.467      6.35 0.0002204
## 5       03   (Intercept)     9.03     1.504      6.01 0.0003207
## 6       03 Shots_on_five     3.20     0.432      7.40 0.0000760
## 7       04   (Intercept)     5.66     1.257      4.50 0.0020008
## 8       04 Shots_on_five     3.38     0.353      9.57 0.0000118
## 9       05   (Intercept)     7.04     1.251      5.63 0.0004927
## 10      05 Shots_on_five     2.33     0.688      3.39 0.0094799
## # ... with 50 more rows
```

The `tidy()` function (from the **broom** library) displays the coefficients for each team-specific model. As an example, let's focus on the fitted model for Team 01.

$$\widehat{\text{Life Satisfaction}}_i = 4.73 + 3.98(\text{Player Success}_i)$$

Team 01's intercept is lower than the intercept from the global model (by 0.97) and it's slope is higher than the slope from the global model (by 0.32). We can actually re-write the team-specific model using these ideas:

$$\widehat{\text{Life Satisfaction}}_i = \Big[5.70 - 0.97\Big] + \Big[3.66 + 0.32\Big](\text{Player Success}_i)$$

In the language of multilevel modeling, the global intercept and slope are referred to as *fixed-effects*. Thus, the intercept fixed effect is 5.70, and the slope fixed effect is 3.66. The team-specific deviations from the fixed effect values are referred to as *random-effects*. The deviation of -0.97 is the random-effect for the intercept for Team 01, and the deviation of $+0.32$ is the random-effect for the slope for Team 01. Each team could potentially have a different random-effect for intercept and slope. For example, writing the team-specific fitted equation for Team 02 in this manner,

$$\begin{aligned}\widehat{\text{Life Satisfaction}}_i &= 9.96 + 2.97(\text{Player Success}_i) \\ &= \Big[5.70 + 4.26\Big] + \Big[3.66 + 0.69\Big](\text{Player Success}_i),\end{aligned}$$

we find that the random-effect of intercept for Team 02 is 4.26 and the random-effect of slope for Team 02 is 0.69.

Fitting the Multilevel Model in Practice

In practice, we use the `lmer()` function from the **lme4** library to fit multilevel models. This function will essentially do what we did in the previous section, but rather than independently fitting the team-specific models, it will fit all these models simultaneously and make use of the information in all the clusters (teams) to do this. This will result in better estimates for both the fixed- and random-effects.

The syntax looks similar to the syntax we use in `lm()` except now we split it into two parts. The first part `Life_Satisfaction ~ 1 + Shots_on_five` is identical to what we use for `lm()`. This indicates that we want fixed-effects for both the intercept and the slope (the fixed-effect of player success). We also have to declare that we want random-effects for these two coefficients. The second part ofd the syntax declares this: `(1 + Shots_on_five | Team_ID)`. This says fit random-effects for the intercept and slope and do so for each team.

```
library(lme4)

# Fit multilevel model
lmer.1 = lmer(Life_Satisfaction ~ 1 + Shots_on_five + (1 + Shots_on_five | Team_ID), data = nba)
```

To view the fixed-effects, we use the `fixef()` function.

```
fixef(lmer.1)
```

```
##   (Intercept) Shots_on_five
##          6.43          3.29
```

To view the team-specific random-effects, we use the `ranef()` function.

```
ranef(lmer.1)
```

```
## $Team_ID
##    (Intercept) Shots_on_five
## 01      0.0779        0.0805
## 02      0.3611        0.3732
## 03      0.3844        0.3973
## 04     -0.0737       -0.0762
## 05     -0.2381       -0.2461
## 06      0.1734        0.1792
## 07     -0.4770       -0.4930
## 08      0.1781        0.1841
## 09     -0.2167       -0.2240
## 10     -0.0346       -0.0357
## 11     -0.1054       -0.1089
## 12      0.0627        0.0648
## 13      0.1753        0.1811
## 14     -0.3967       -0.4100
## 15     -0.0826       -0.0854
## 16      0.0969        0.1001
## 17     -0.4050       -0.4186
## 18      0.1208        0.1249
## 19     -0.0207       -0.0214
## 20      0.1285        0.1328
## 21      0.1865        0.1927
## 22     -0.2569       -0.2655
## 23      0.2504        0.2588
## 24     -0.4726       -0.4884
## 25     -0.1134       -0.1172
## 26     -0.2439       -0.2521
```

```
## 27      0.0678        0.0701
## 28      0.3840        0.3969
## 29      0.3433        0.3548
## 30      0.1463        0.1512
```

From these two sets of information, we can re-construct each team-specific fitted equation if we are so inclined. For example, to construct the team-specific equation for Team 30, we first write the global equation (from the fixed-effects):

$$\widehat{\text{Life Satisfaction}}_i = 6.43 + 3.29(\text{Player Success}_i)$$

Then we add the random-effects for Team 30 to the respective fixed-effects:

$$\begin{aligned}\widehat{\text{Life Satisfaction}}_i &= \Big[6.43 + 0.1463\Big] + \Big[3.29 + 0.1512\Big](\text{Player Success}_i) \\ &= 6.58 + 3.44(\text{Player Success}_i)\end{aligned}$$

Variance Components: Quantifying Variation in the Random-Effects

Looking over the random-effects, it is clear that the random-effects for intercept vary across teams. The random-effects for slope also vary across team. It is common to quantify this variation by computing the standard deviation (or variance) for each set of random-effects. We can obtain these estimates by using the `VarCorr()` function.

```
VarCorr(lmer.1)
```

```
##  Groups   Name          Std.Dev. Corr
##  Team_ID  (Intercept)   0.305
##           Shots_on_five 0.315    1.00
##  Residual               2.260
```

The output gives standard deviations of the random-effects. To get variances, we square these values. For example to obtain the variance component for the intercept random-effects,

```
0.305 ^ 2
```

```
## [1] 0.093
```

There is also a standard deviation for the residuals. This is a measure of the variation in the player-level errors. Remember, even if we use a team-specific equation to predict life satisfaction, there will still be deviation between the predicted value and a player's actual life satisfaction.

Recall that the goal in regression modeling is to explain variation. In a multilevel model, we can now try to explain between-team variation (the variation in intercepts and slopes) and within-team variation (residual). Comparing the size of the variance components, we see that the within-team variation (residual) is a lot larger than the between-team variation. This suggests that we may want to focus on including predictors that vary across players rather than on team-level predictors.

References

Woltman, H., Feldstein, A., MacKay, J. C., & Rocchi, M. (2012). An introduction to hierarchical linear modeling. *Tutorials in Quantitative Methods for Psychology*, *8*(1), 52–69.