Catch A Cheater

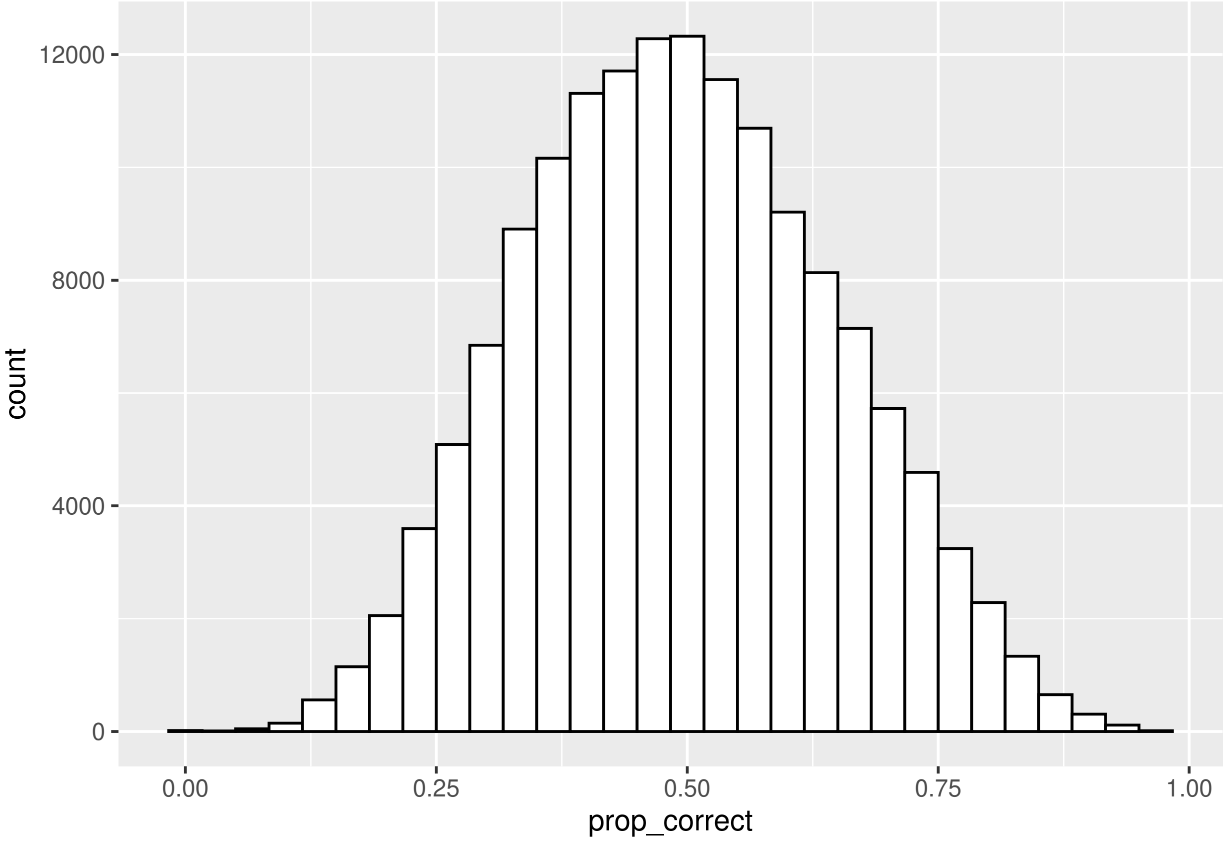
Harrison Jia

5/9/2021

General distribution of percent correct for each person who has competed in this triva league.

```
## `summarise()` has grouped output by 'user_id'. You can override using the `.groups` argument.
```

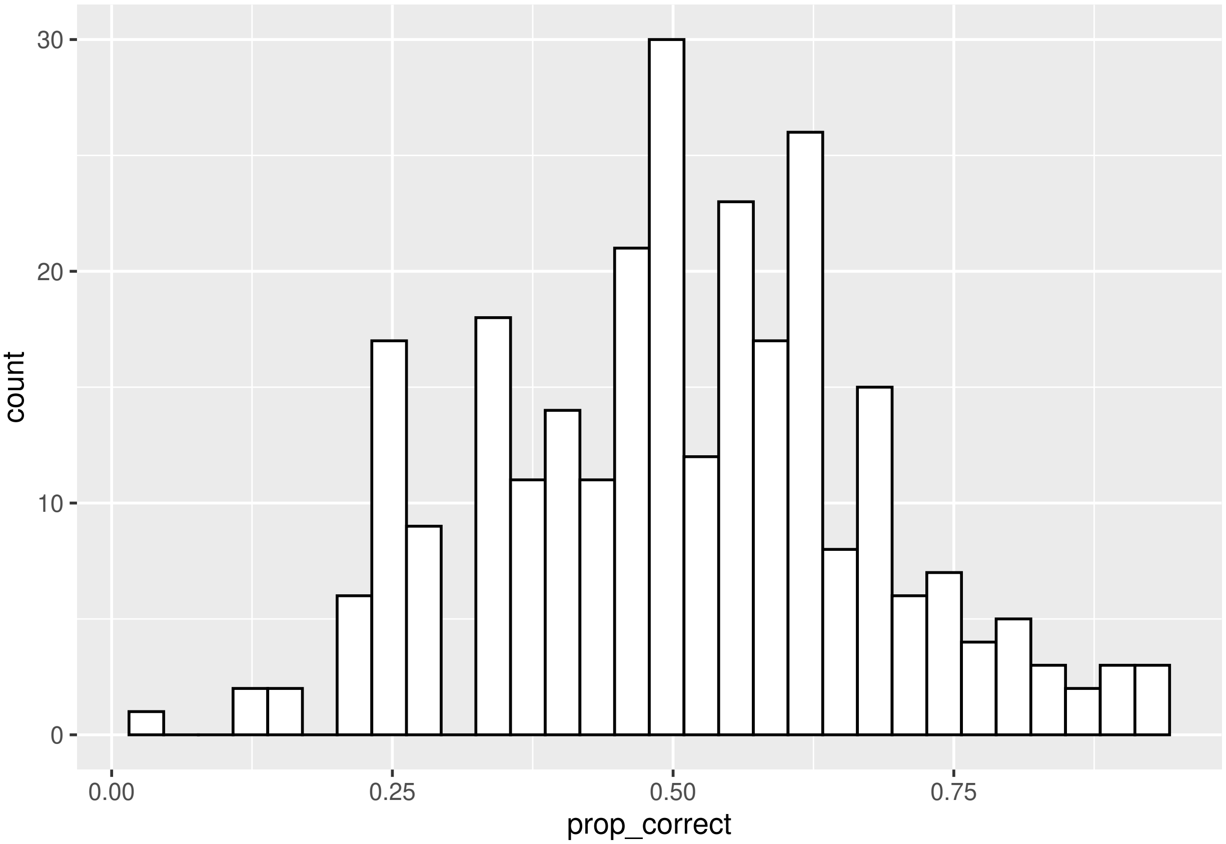
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Genderal distribution of percent correct for each person who competed in the championship round.

```
## `summarise()` has grouped output by 'id'. You can override using the `.groups` argument.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We have a similar mean it seems but we do need to account for the fact that the people in the championship round likely were towards the right side of the bell curve in the first graph.

For each person who made it into the championchips in a given year, what was their correct rate specifically for that year?

Ordering by year given the season, yeilds the following results.

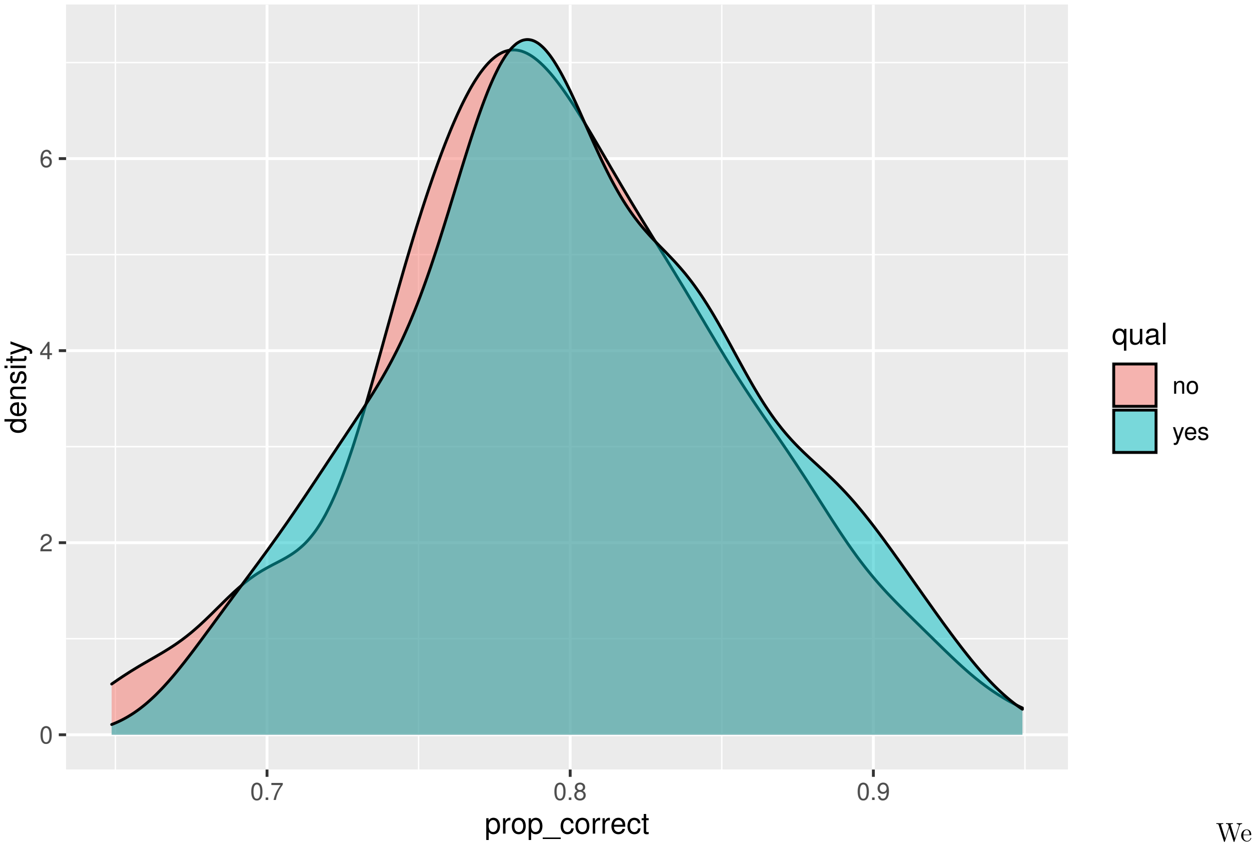
```
## `summarise()` has grouped output by 'user_id'. You can override using the `.groups` argument.
```

Interesting to see that there seems to be a slow increase on average per year for those have gotten to the championships at least once.

```
## [1] 0.7873438
## [1] 0.797712
## [1] 0.8007327
## [1] 0.8006588
```

Are questions getting easier or people getting better? Or maybe just pure noise and that the average is around the same as before.

Since championship data gives us data on when each person attended the finals, we should see if on average there is a difference in regular season scores for years they do go vs years they don't.



We see that there seems to be a marginally better performance when they do but the difference here looks really small.

They both also have very simimlar variance values (in turn SD should be similar as well). This is also true for the mean (printed first here).

```
## [1] 0.800386
```

```
## [1] 0.7951008
```

```
## [1] 0.003297817
```

```
## [1] 0.00340905
```

Printed in order of mean of scores in year of qualification, mean of scores in year did not attend or not qualify, variance in respective order.

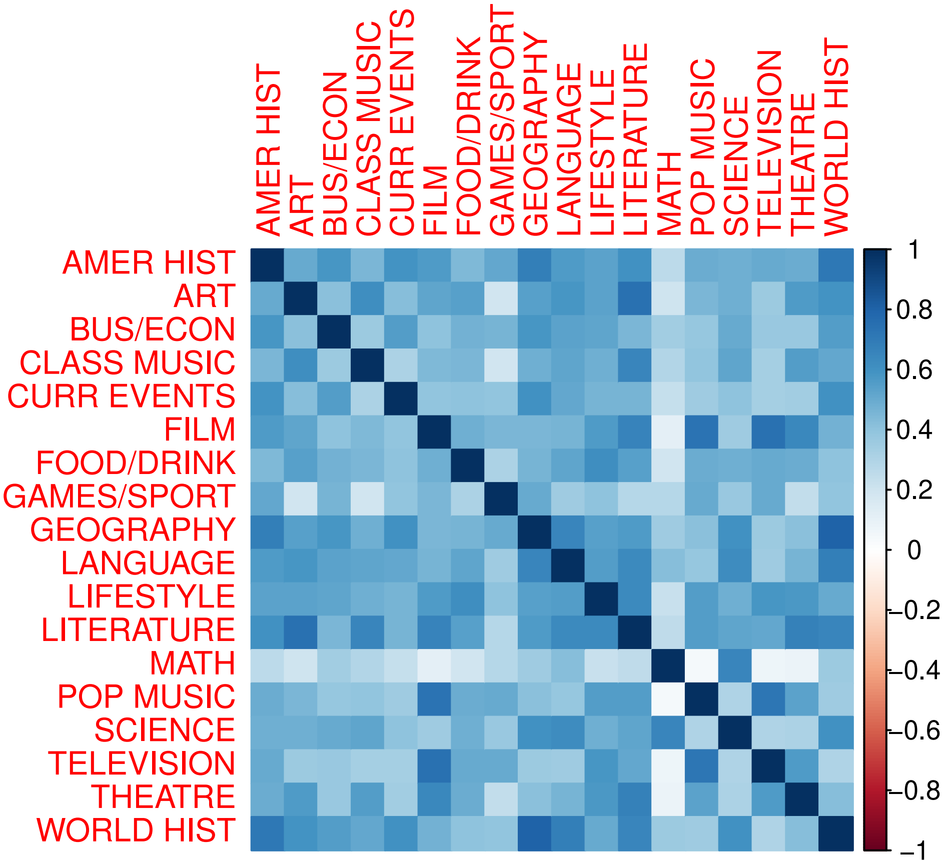
So far on a macro scale, it seems that performance per year and relative to attending the finals are nearly in distiguishable. It can very possibly just be a result of a low percentage of people actually cheating or did so for all the years. Likewise if there are those who chose to cheat and changed from not cheating before and that they make up a lower percentage then by CLT and LLN it may be nearly impossible to see them in a full distribution.

Correlation between subjects?

```
## `summarise()` has grouped output by 'user_id'. You can override using the `.groups` argument.
```

```
## Using prop_correct as value column.  Use the value argument to cast to override this choice
```

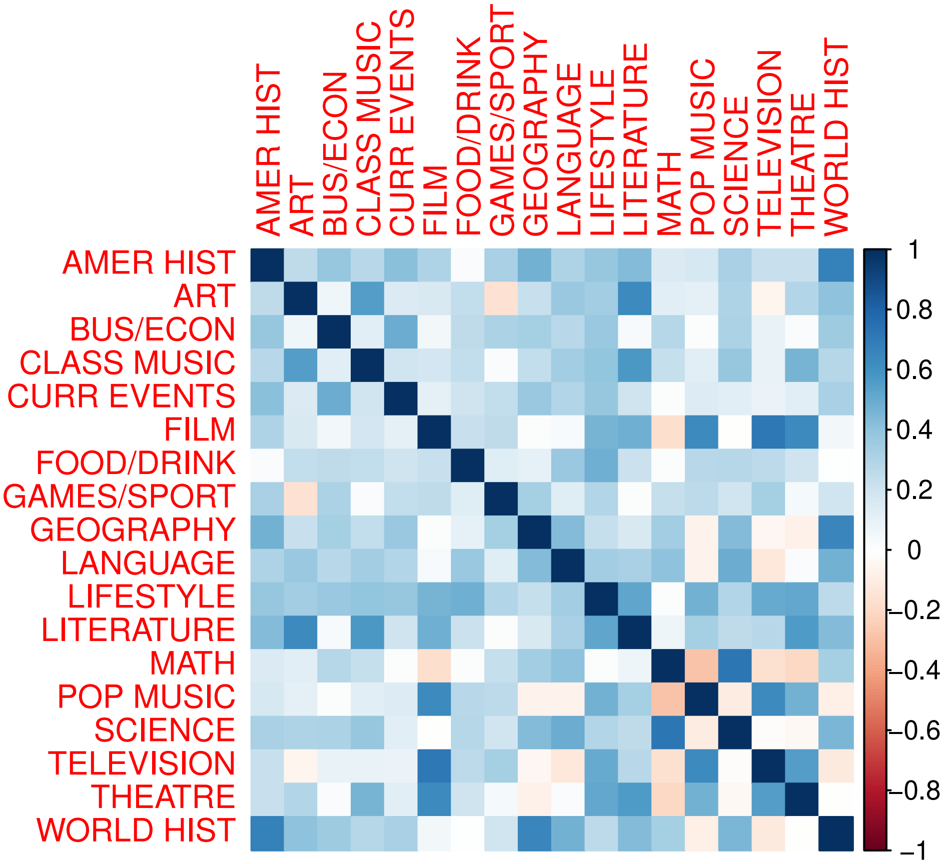
Clearly we can see that there is. Math correlates heavily with science as one might expect but relative to others with each other it underperforms in every other subject especiall pop music. There may be a few places with values close to 0 or uncorrelated with each other, but there is no insance where there is any negative correlation.



Now if we focus only on those who made it to the championship rounds, we get much less correlation all around and now also see some negative correlation.

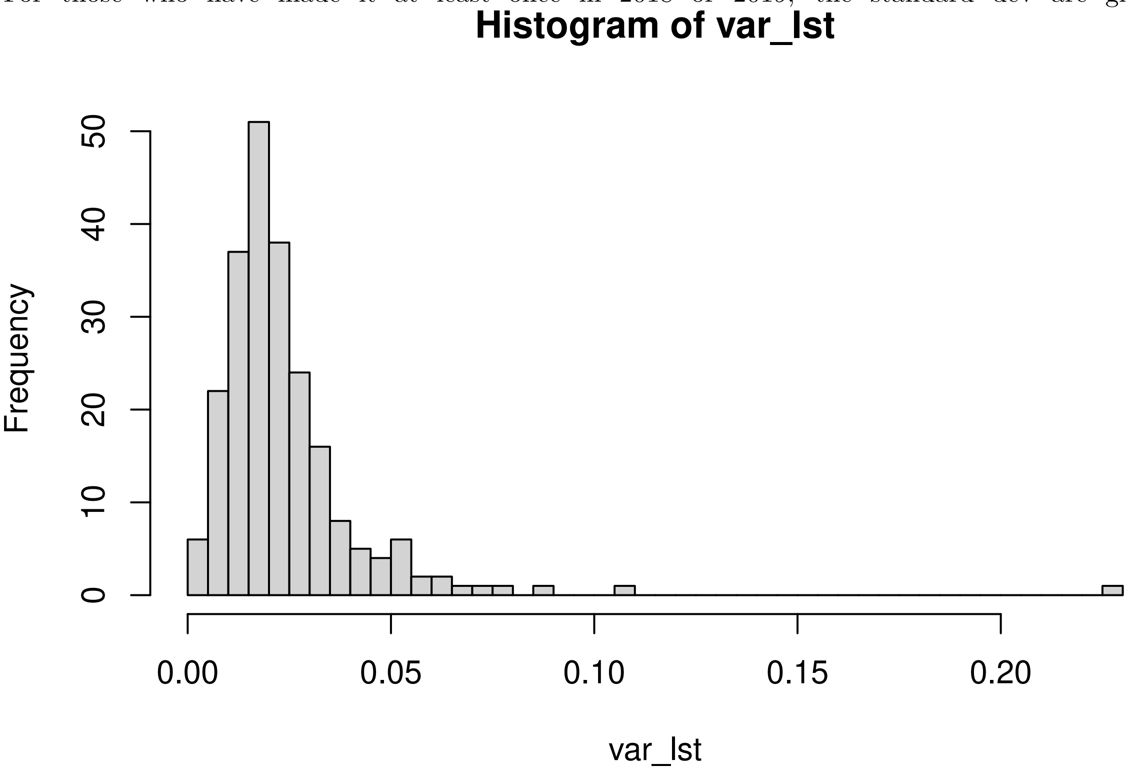
```
## `summarise()` has grouped output by 'user_id'. You can override using the `.groups` argument.
```

```
## Using prop_correct as value column.  Use the value argument to cast to override this choice
```


AMER HIST
ART
BUS/ECON
CLASS MUSIC
CURR EVENTS
FILM
FOOD/DRINK
GAMES/SPORT
GEOGRAPHY
LANGUAGE
LIFESTYLE
LITERATURE
MATH
POP MUSIC
SCIENCE
TELEVISION
THEATRE
WORLD HIST
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

How much do scores vary for chose in championship and not?

For those who have made it at least once in 2018 or 2019, the standard dev are given below



Interesting to see a few values far away from the rest

Let's examine the one person that is at the far right, we will see what their standard dev is and scores were in the championship and what their scores were for the years they participated in the regular season.

```
## [1] 0.2268816
```

```
## [1] 201
```

```
## [1] 18337
```

```
## [1] 0.25
```

```
## [1] 2019
## attr(,"format.stata")
## [1] "%8.0g"
```

```
## [1] 0.3600000 0.7662771 0.8416667 0.8200000
```

He/She (id = 18337) went to the finals in 2019, at the time they scored around 84% accuracy the highest they have ever done. Also interesting is that he/she scored 36% in 2017. Interesting.

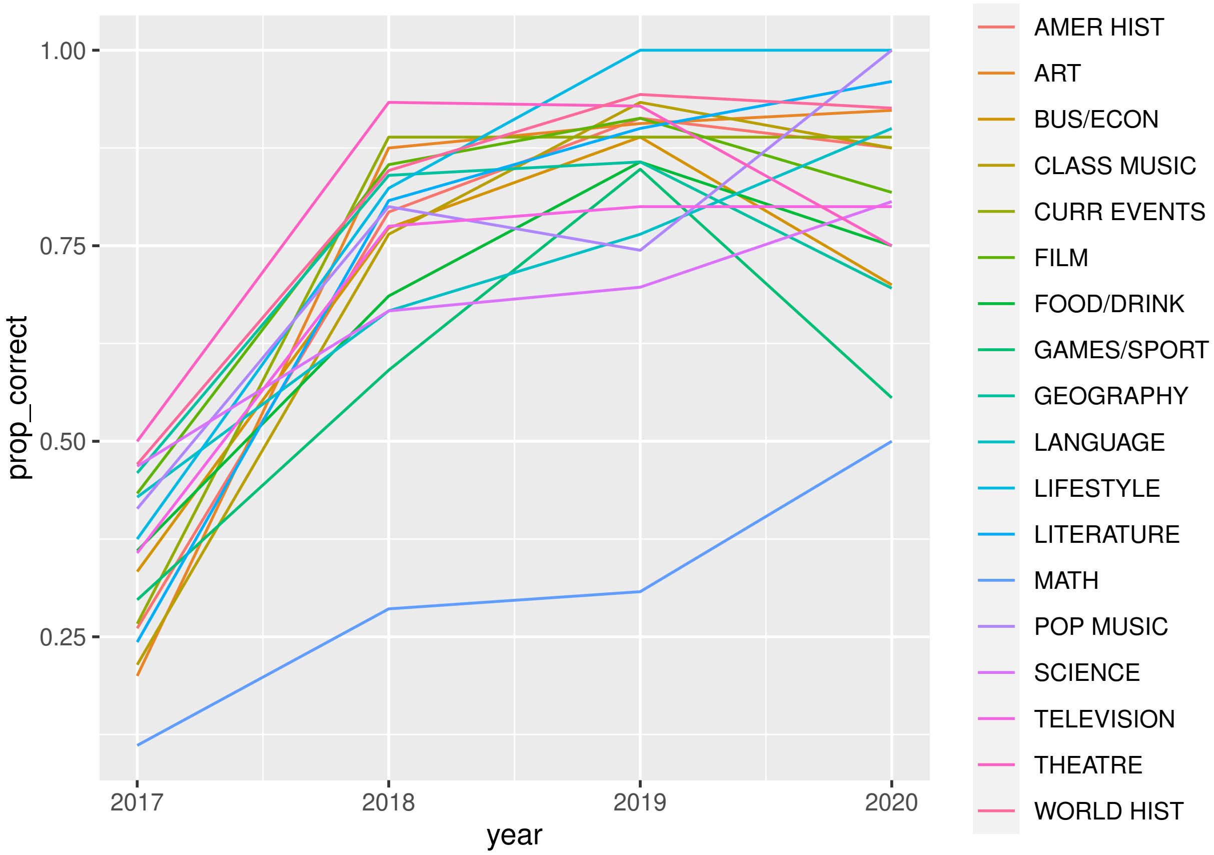
Examining 18337's stats based on category and year

```
## `summarise()` has grouped output by 'category'. You can override using the `.groups` argument.
```

```
## # A tibble: 72 x 3
## # Groups:   category [18]
##    category   year prop_correct
##    <chr>     <dbl>        <dbl>
##  1 AMER HIST  2017        0.261
##  2 AMER HIST  2018        0.793
##  3 AMER HIST  2019        0.913
##  4 AMER HIST  2020        0.875
##  5 ART        2017        0.2
##  6 ART        2018        0.875
##  7 ART        2019        0.906
##  8 ART        2020        0.923
##  9 BUS/ECON   2017        0.333
## 10 BUS/ECON   2018        0.773
## # ... with 62 more rows
```

Interestingly, the jump between 2017 to years after is all around the same - that he/she improved the same amount is so many categories except GAMES/SPORT (a bit less), MATH (constient but still low raw %) and Science. Subjects that may be harder to search for online.

```
## # A tibble: 12 x 3
## # Groups:   category [3]
##    category     year prop_correct
##    <chr>       <dbl>        <dbl>
##  1 GAMES/SPORT  2017        0.297
##  2 GAMES/SPORT  2018        0.591
##  3 GAMES/SPORT  2019        0.848
##  4 GAMES/SPORT  2020        0.556
##  5 MATH         2017        0.111
##  6 MATH         2018        0.286
##  7 MATH         2019        0.308
##  8 MATH         2020        0.5
##  9 SCIENCE      2017        0.468
## 10 SCIENCE      2018        0.667
## 11 SCIENCE      2019        0.697
## 12 SCIENCE      2020        0.806
```



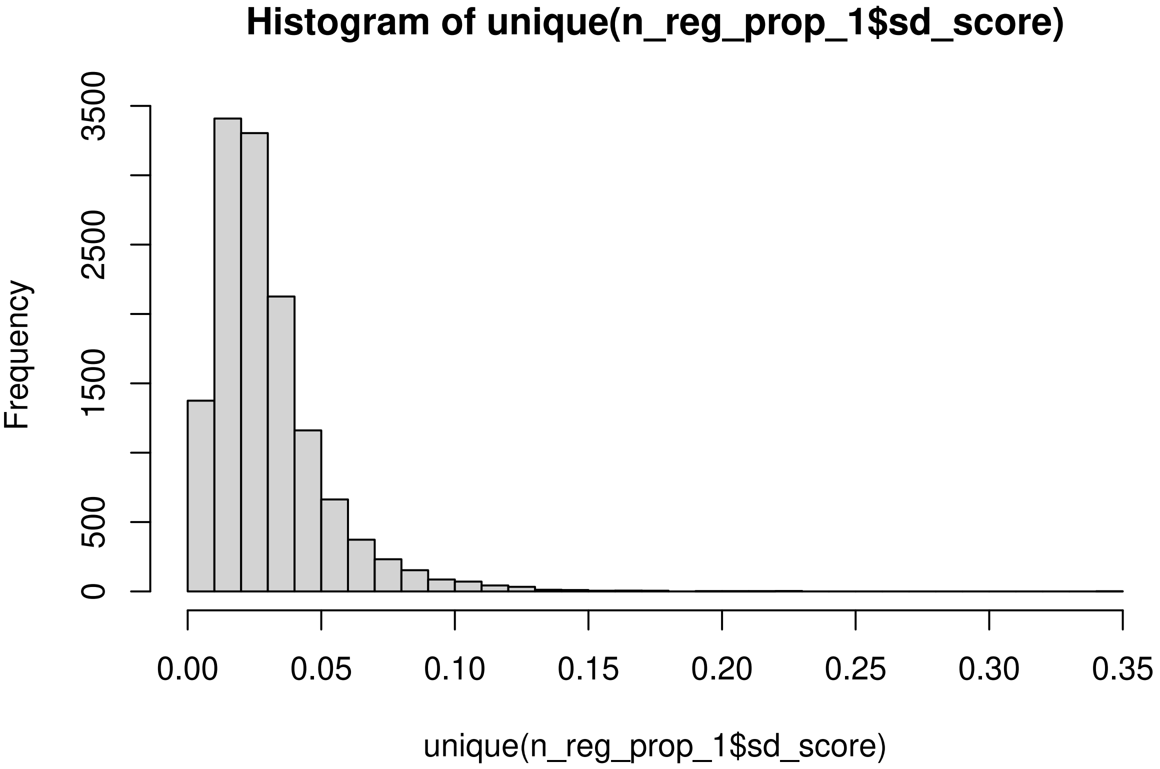
Although messy, we can see the clear distinct change in math vs the rest and gaming/sport and science als violating the natural growth in a lesser extent.

It seems quite clear that this person is cheating, some pointers to learn from this so far is that this likely cheater improved alot within 1 year, could not cheat his/her way through math the same way while he could with gaming and science but also relative to the other subject much less so as well. Either improved his/her math skills or learned to cheat better in math (this pattern is alto true with science).

This variance based analysis may spot some easy to detect cheaters but if a cheater started in 2017 or earlier, they would not be seen as easily.

How about the regular season and the people who did not qualify?

```
## `summarise()` has grouped output by 'user_id'. You can override using the `.groups` argument.
```



Now what about the questions themselves?

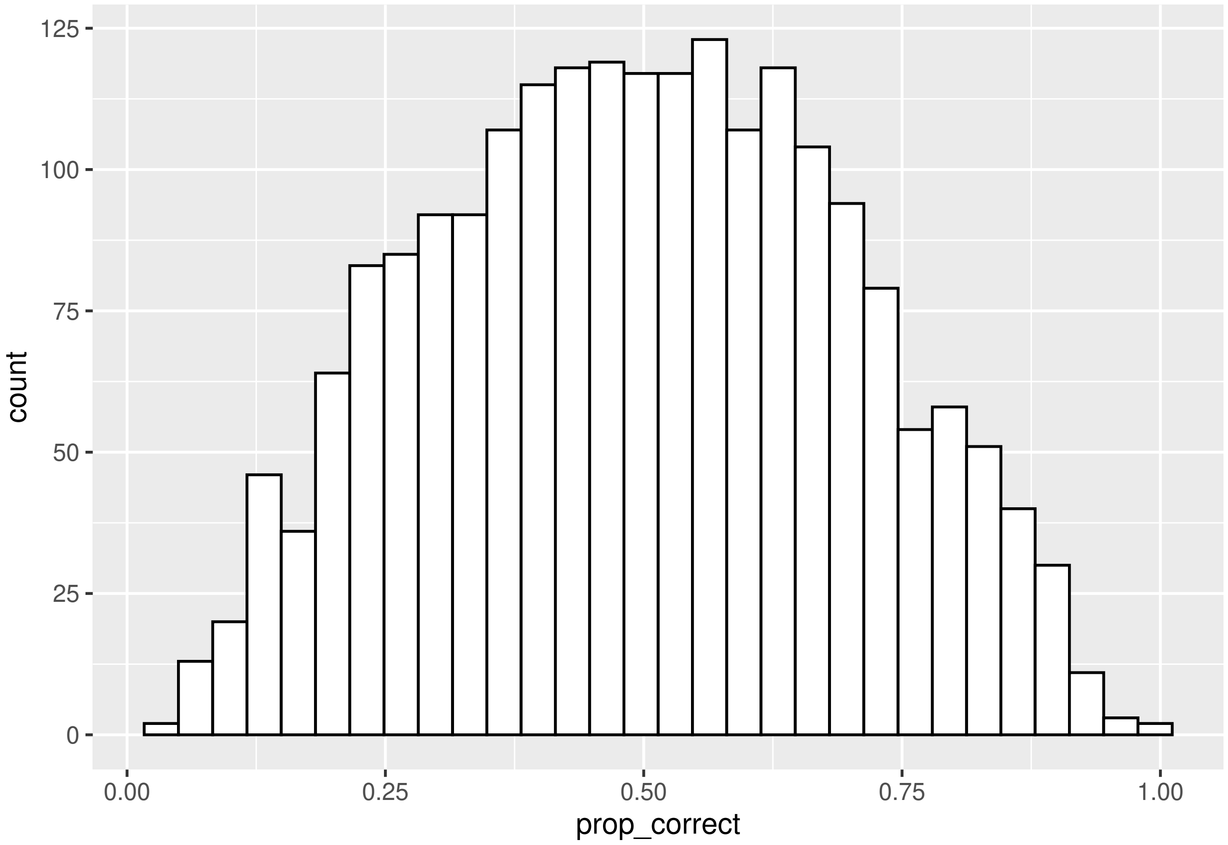
We see that relative difficulty is not determined by the question number.

```
## # A tibble: 6 x 2
##   question prop_correct
## *    <dbl>        <dbl>
## 1        1        0.517
## 2        2        0.494
## 3        3        0.480
## 4        4        0.489
## 5        5        0.487
## 6        6        0.499
```

However question difficulty varies around the mean of about 0.5.

```
## `summarise()` has grouped output by 'season', 'match'. You can override using the `.groups` argument
```

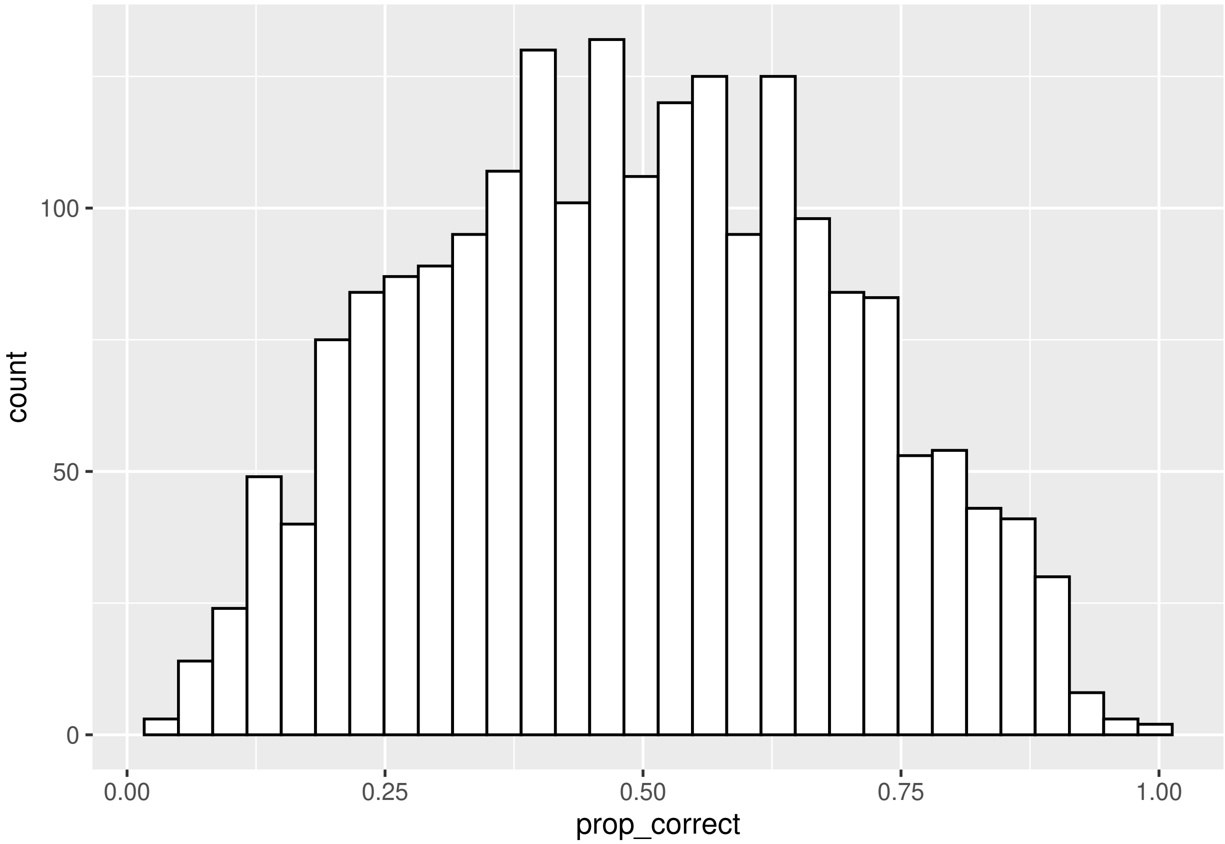
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Non-qualifiers regular season data.

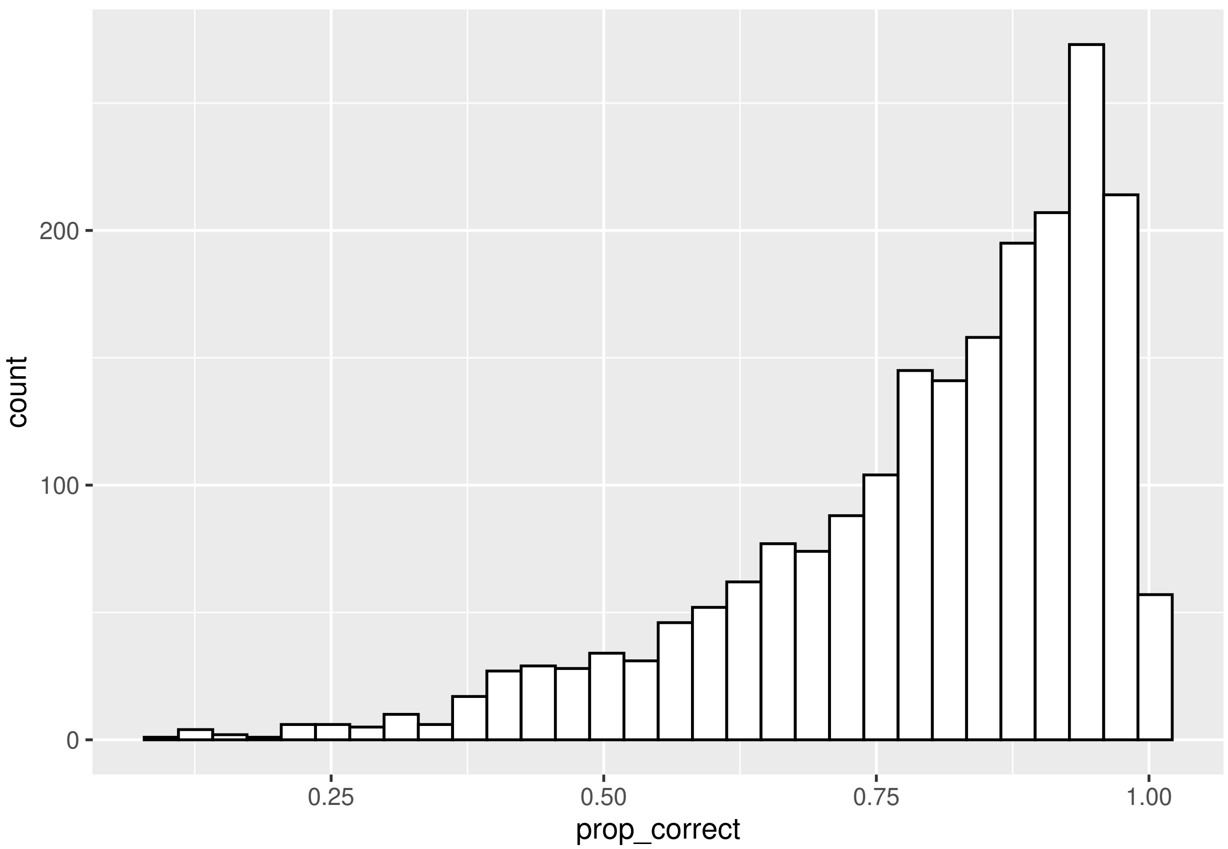
```
## `summarise()` has grouped output by 'season', 'match'. You can override using the `.groups` argument
```

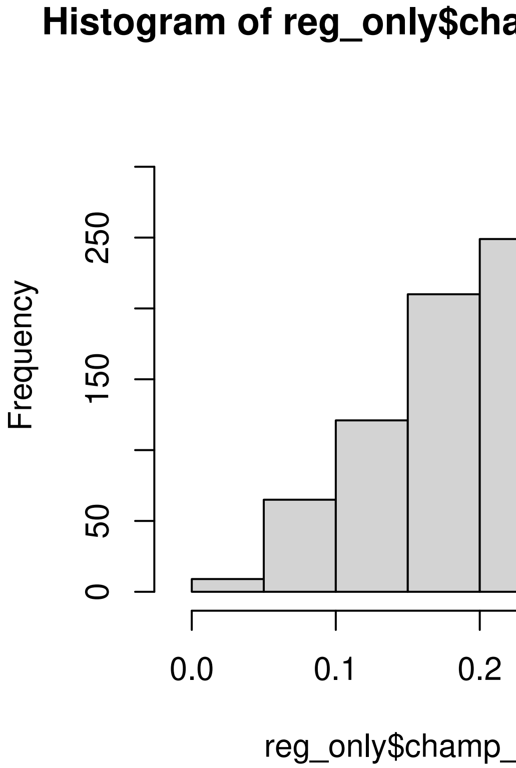
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Qualifiers' regular season data.

```
## `summarise()` has grouped output by 'season', 'match'. You can override using the `.groups` argument
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```





Difference in proportion correct based on those who qualified and those who didn't.

How about proportion correct by category?

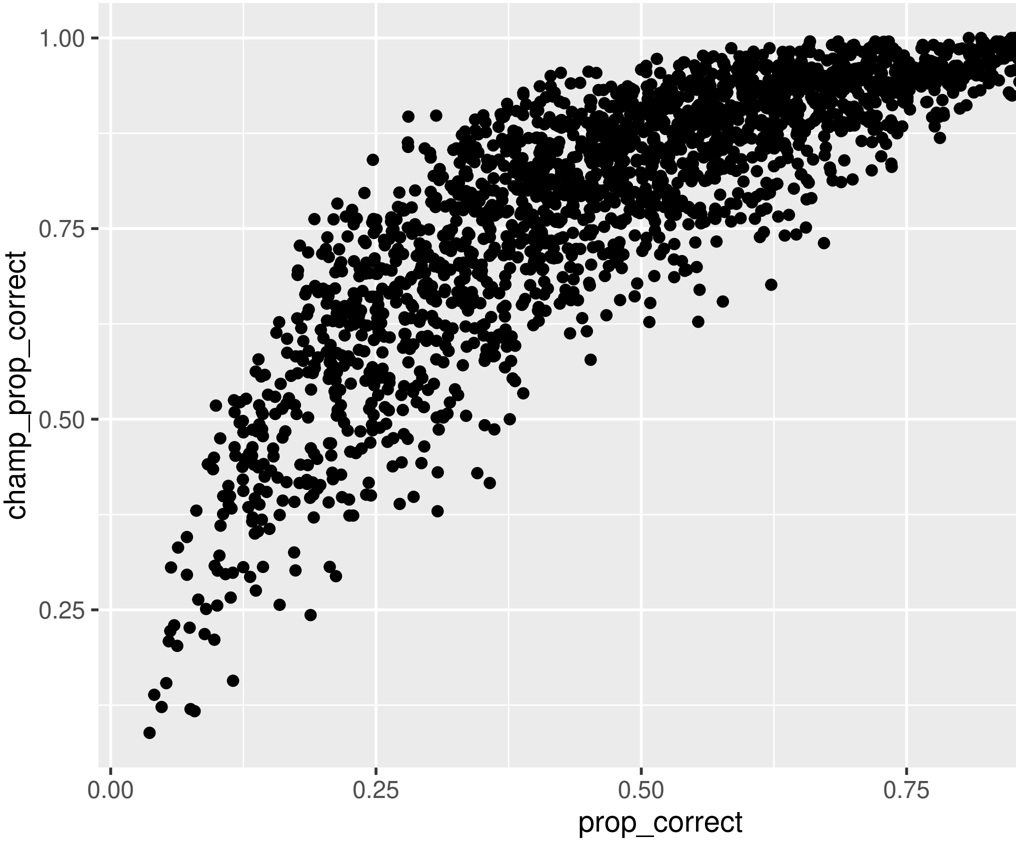
We can see that for all regular season players the proportion correct varies a bit around the mean of about

0.5 whereas generally chose who make it to the finals do around 0.25 better with some variance as well.

```
## # A tibble: 18 x 3
##    category    prop_correct champ_diff
##    <chr>              <dbl>      <dbl>
##  1 AMER HIST          0.548      0.332
##  2 ART                0.404      0.321
##  3 BUS/ECON           0.501      0.247
##  4 CLASS MUSIC        0.390      0.337
##  5 CURR EVENTS        0.577      0.239
##  6 FILM               0.511      0.303
##  7 FOOD/DRINK         0.588      0.229
##  8 GAMES/SPORT        0.473      0.259
##  9 GEOGRAPHY          0.485      0.356
## 10 LANGUAGE           0.575      0.281
## 11 LIFESTYLE          0.530      0.294
## 12 LITERATURE         0.458      0.332
## 13 MATH               0.472      0.273
## 14 POP MUSIC          0.484      0.297
## 15 SCIENCE            0.491      0.311
## 16 TELEVISION         0.513      0.265
## 17 THEATRE            0.426      0.316
## 18 WORLD HIST         0.490      0.364
```

What about the relation between the two?

We can see that generally questions are easy for most of championship quallifiers before around the 0.30 mark where it sharply decresses from there.



Interesting to note: Questions that have relatively higher correct rate among championship qualifiers and lower overall correct rate may tell us the most about who makes it or not, we would need to dig deeper to examine this.

Percentile of the distributions for no qualifiers only and and qualifiers only.

```
##         0%        25%        50%        75%       100% 
## 0.03664454 0.33653990 0.48683662 0.63995026 0.99964677
```

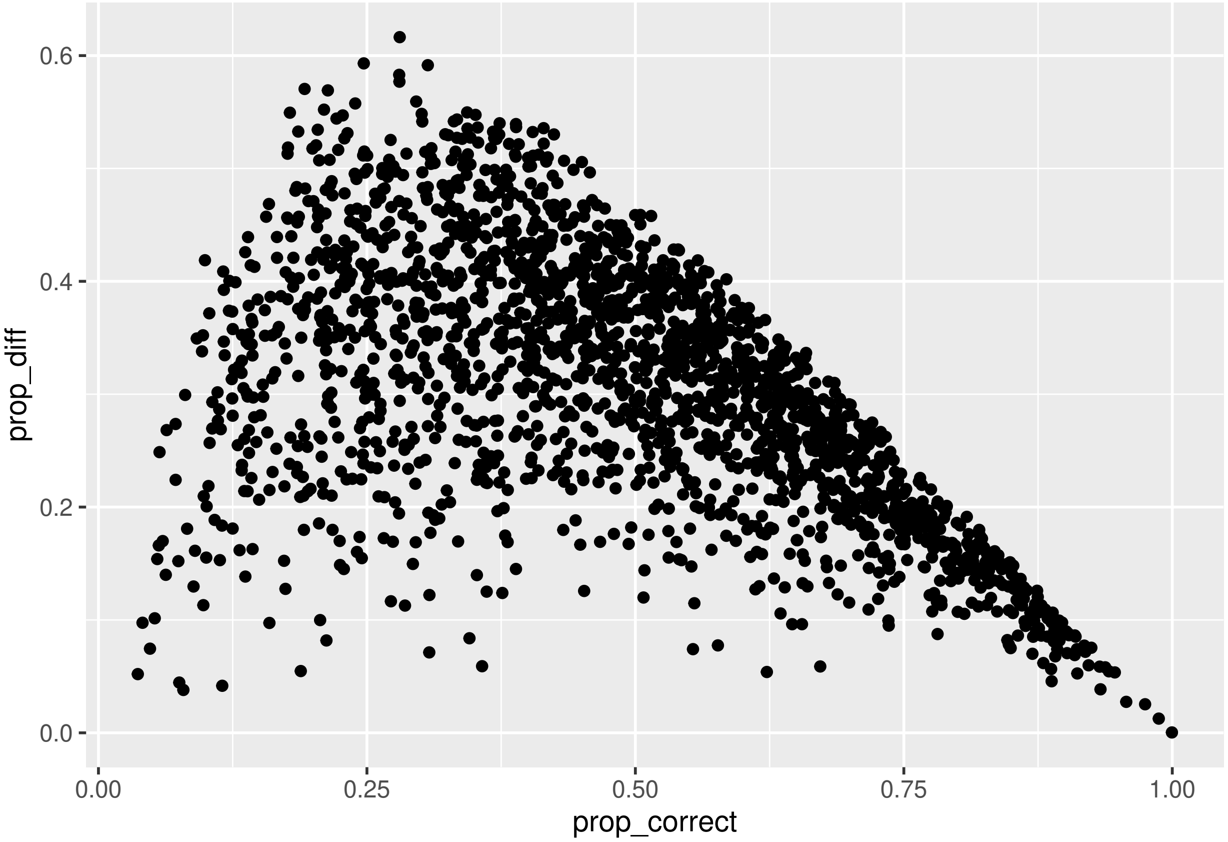
```
##         0%        25%        50%        75%       100% 
## 0.08866995 0.71100917 0.84432550 0.92817680 1.00000000
```

Predicting the likelihood that a certain questions is answered or not by a person within this field. To do this we will use mostly things we discovered towards the latter half of the data exploration. We have some leads such as the difference in question difficulty depending on what your average score is and if you made it to the championship rounds. It may be a bit hard to examine or determine a relation between specific question types with person as this may lead to serious multicolinearity issues as we saw above.

Take only those with 450 questions answered or more to minimize data overloading on computer as well as negating effects of smaller sample sizes.

```
## `summarise()` has grouped output by 'user_id', 'big_diff', 'is_hard'. You can override using the `.g
```

Given a questions that has a large difference (or not) and is easy or hard relative to the population what is the liklihood that someone will get a question right? Here big diff is synonymous for will there be a large effect given that someone scored high enough to make it to the finals round or not. Using the symmetry of the width of answers (aka difference between those who were qualifiers and the general rounds), we can get the variables realtively close to uncorrelated. Model may be over simplified but gets a good idea as to what happens given data on a particular question but also knowing that a person is or isn't a qualifier lead to different answers. Is hard threshold was set at 0.20 around the peak of prop_diff and easy threshold was at 0.6 and above. Big diff was set at 0.3 using the graph as well. Year did not show much effect on the questions as a whole.

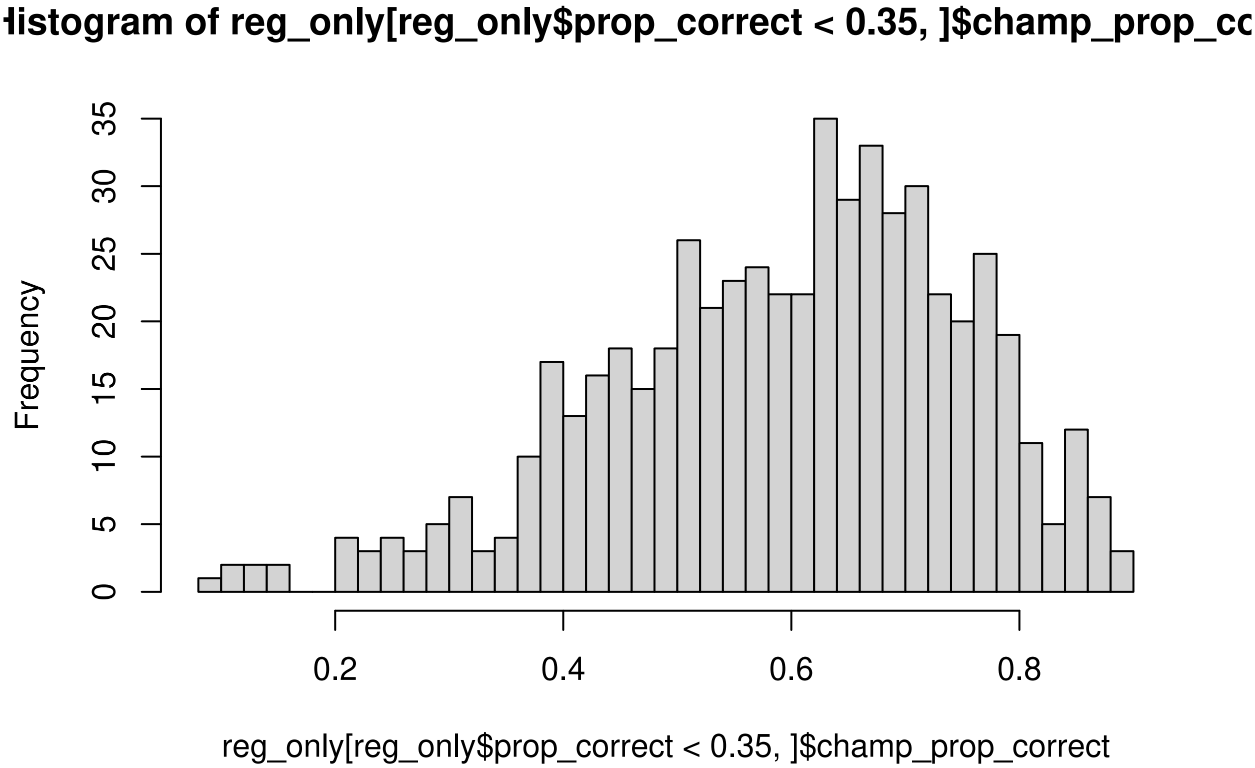


```
##
## Call:
## lm(formula = prop_correct ~ big_diff + is_hard + is_easy + big_diff *
##     is_hard + big_diff * is_easy, data = df_smaller)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.62908 -0.10390 -0.01558  0.09914  0.78428
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.427692   0.001322  323.48   <2e-16 ***
## big_diff         -0.021566   0.001870  -11.53   <2e-16 ***
## is_hard          -0.296729   0.001870 -158.69   <2e-16 ***
## is_easy           0.306647   0.001870  164.00   <2e-16 ***
## big_diff:is_hard  0.045348   0.002644   17.15   <2e-16 ***
## big_diff:is_easy -0.083695   0.002644  -31.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1572 on 84840 degrees of freedom
## Multiple R-squared:  0.6665, Adjusted R-squared:  0.6665
## F-statistic: 3.392e+04 on 5 and 84840 DF,  p-value: < 2.2e-16
```

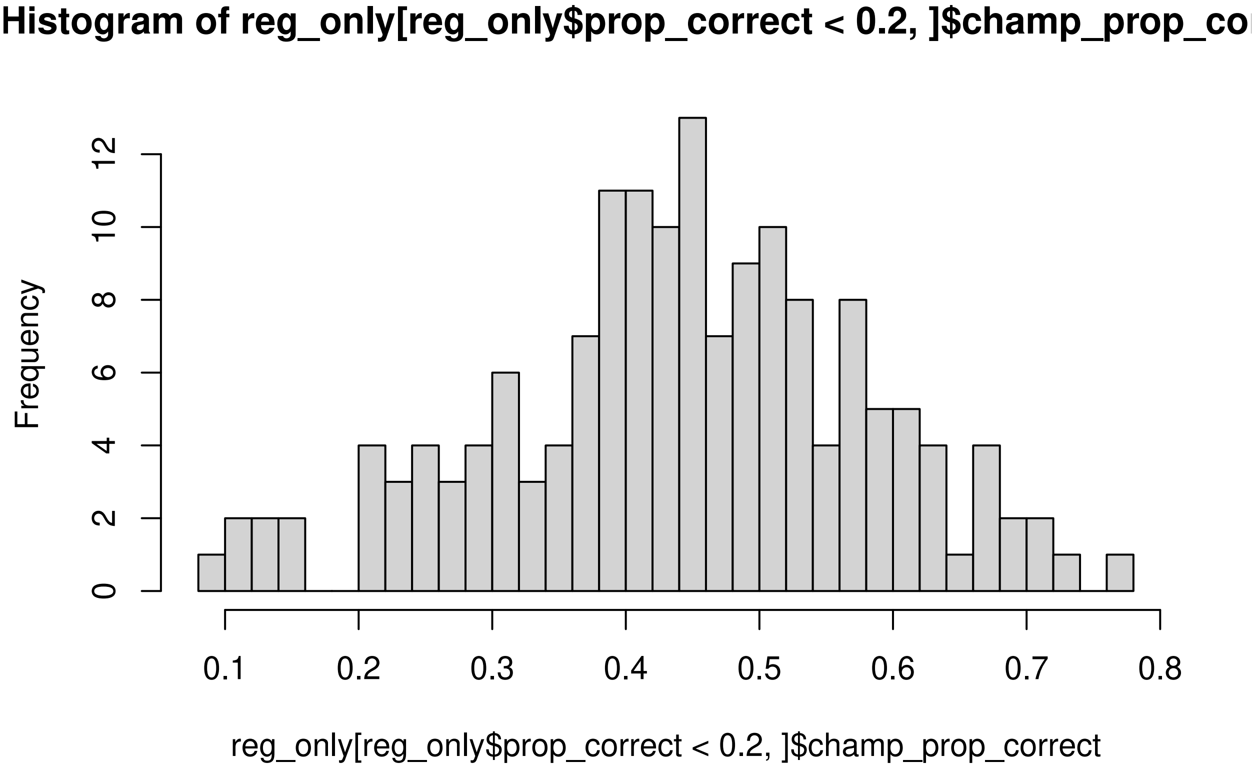
Now we turn our focus onto the finals

We want to figure out a way to determine how a person would perform in the final rounds given their scores in the regular season. Naturally we would expect those who can answer the harder questions in the regular season would likely do well in the finals. A person's ability to get hard or even medium questions correct in regular season would expect to do well in the finals round compared to someone who aces the easy one but struggles a bit more in the harder ones. We saw in the data exploration that variance of scores per year seem to point towards some potential effects.

Looking at the harder questions distribution (again the categoory of questions where the general population scored below 0.35) we can see that this data distribution is quite similar for questions presented in the finals however it is slightly "easier".



If we tune this a bit lower we may get a closer distribution. (0.2)



So we look at players ability to answer questions with > 0.2 general correct rate. We noticed there may be negative effects with variance/sd. In addition, we will remove players with less than 450 questions answered as this tend to over inflate sd and variance or produce NA's (per year).

Predicted chance of getting question right in finals (round 1 and 2) = score in regular season against questions with <0.2 success rate - (coeff) * sd of score over years.

Now to test the model on the real championship data.

```
## `summarise()` has grouped output by 'user_id'. You can override using the `.groups` argument.
```

```
##
## Call:
## lm(formula = prop_final ~ sd + prop_hard, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46575 -0.09881  0.00010  0.10797  0.37720
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.43477    0.03619  12.012   <2e-16 ***
## sd          -0.31832    0.52806  -0.603   0.5472
## prop_hard    0.13653    0.07235   1.887   0.0604 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.159 on 224 degrees of freedom
## Multiple R-squared:  0.01727,    Adjusted R-squared:  0.008501
## F-statistic: 1.969 on 2 and 224 DF,  p-value: 0.142
```

Seems as through the effect of prop_hard or getting questions correct on ones that were typically harder in the regular season did not have a significant affect. Likewise the standard deviation effect was insignificant although it was quite negative. So from this analysis we cannot determine who had cheated or not.

(Note, my computer crashed and I had lost quite a few of data tables that I tried to recode back in but somehow after at least 3 hrs of recoding gotten different results from before, and with each other as well, the values and blocks of data below were from that analysis were I had found that there was no ssignificance with prop_hard, but under type-1 alpha 0.05 there was significance with sd - However lessoned learned about how I need to keep better notes on what I code outside of R markdown.)

Of those who qualified for 2020, we can see that there are some people with potential substantial effect on their expected performance relative to the sd in their yearly performance. Setting a hard threshold say -0.1 would get us around 20 or so of the 490 participants cheating. A threshold lower say -0.05 would get us around 100 of the 490 participants posibbly cheating.

```
## # A tibble: 6 x 2
## # Groups:   user_id [6]
##   user_id potential_effect
##     <dbl>            <dbl>
## 1   18337           -0.284
## 2    7185           -0.242
## 3    8913           -0.201
## 4   15829           -0.168
## 5    4211           -0.154
## 6    6270           -0.145
```

Now to predict number of questions each contestant will get in the upcoming championship.

For this we will keep the variance effect, as to those who don't cheat, their scores will be nearly completely unimpacted. We will also use 2020 scores, by regressing 2018 scores onto themselves.

```
## `summarise()` has grouped output by 'id', 'year'. You can override using the `.groups` argument.
```

```
##
## Call:
## lm(formula = champion_data2018[champion_data2018$round == 1,
##     ]$prop_correct ~ data_2018_reg_qualifiers_hard$prop_hard +
##     data_2018_reg_qualifiers_hard$sd_score)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.51133 -0.13118 -0.02537  0.16665  0.46955
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                              0.22499    0.07699   2.922 0.004172
## data_2018_reg_qualifiers_hard$prop_hard  0.52358    0.13169   3.976 0.000122
## data_2018_reg_qualifiers_hard$sd_score  -0.34891    1.99112  -0.175 0.861199
##
## (Intercept)                             **
## data_2018_reg_qualifiers_hard$prop_hard ***
## data_2018_reg_qualifiers_hard$sd_score
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2124 on 117 degrees of freedom
## Multiple R-squared:  0.1199, Adjusted R-squared:  0.1049
## F-statistic: 7.973 on 2 and 117 DF,  p-value: 0.0005675
```

Now using this model on the 2020 data we get the following:

```
## # A tibble: 490 x 3
##    user_id prop_hard predict
##      <dbl>     <dbl>   <dbl>
##  1      25     0.823   0.352
##  2     129     0.782   0.342
##  3     151     0.74    0.331
##  4     177     0.727   0.334
##  5     204     0.807   0.379
##  6     226     0.77    0.363
##  7     390     0.632   0.291
##  8     394     0.79    0.375
##  9     493     0.733   0.346
## 10     512     0.783   0.373
## # ... with 480 more rows
```