Stat 399:
The Penn Course Review Project

Arjun Gupta
Professor Adi Wyner

Acknowledgements

- To Wharton's G95 division and in particular Scott Romeika, for providing the data that made this project possible.

- To Professor Adi Wyner, for allowing my first real project, and all the mistakes the come with it, to happen under his patient supervision.

- To Justin Bleich, for the always being willing to answer an incessant stream of questions and inspiring a passion for statistics in the spring of my freshman year that has never since wavered.

- To Adel Qalieh, for graciously helping out a desperate Whartonite who found himself in over his head with API calls. Without him, this project would not have been possible.

- To Andrew Shichman, for introducing me to the beauty of the Pandas module and always being willing to troubleshoot my Python code.

- To all my friends who endured me talking about this: I appreciate it. And I'm a dork, I know.

Introduction

Twice a year, about 10,00 undergraduates at Penn (because of graduating seniors in the spring that number is closer to 7,500) must make course selections for the upcoming semester. These decisions are quite important to students as upon these courses rest career ambitions, potential GPA changes, and a whole host of other factors influencing the quality of the upcoming semester. Thus, there is a strong reason to make the most-informed choice possible. Enter Penn Course Review--a repository of knowledge with course quality, difficulty, and instructor difficulty ratings. Abbreviated as PCR, it is accepted as a foremost source of information on classes (so much so that DP columns have discouraged students from relying on it too heavily: http://www.thedp.com/article/2013/10/how-not-to-choose-classes). Many faculty strongly encourage students to look beyond the numbers. But do they? This report seeks understand to degree to which course enrollments can be explained by Penn Course Review ratings, as well as see if, using ratings, enrollments can be predicted.

Throughout this study we will refer to a set of core predictors, and a set of ancillary predictors. We define the set of core predictors as follows:

- Year-What year the course was given
- Fall.Spring-was the course given in the fall or spring?
- Subject-what department the course was in (finance, statistics, etc)
- Section-The section number. The higher the section number, usually the later in the day or week the class meets.
- Core-Is the class part of the Wharton core, which all students must complete to graduate?
- grad.crosslisting-Is the class cross-listed with a graduate class?
- CourseQuality-On a scale from 0-4, how past students rated the quality of the class. Higher is better.
- InstructorQuality- On a scale from 0-4, how past students rated the quality of the instructor. Higher is better.
- CourseDifficulty- On a scale from 0-4, how past students rated the difficulty of the class. Higher is harder.

Ancillary predictors:

- AmountLearned-The amount the student felt like they learned
- WorkRequired-How much work was required for the class?
- ReadingsValue-How valuable were the readings?
- CommAbility-How clearly was the instructor able to communicate course material?
- InstructorAccess-How accessible was the instructor outside of class?
- StimulateInterest-How well was the instructor able to stimulate student interest?
- RecommendMajor-How likely is the student likely to recommend the class to a major in the class' department
- RecommendNonMajor- How likely is the student likely to recommend the class to a non-major in the class' department

Section I: Data preparation

"95% of data science isn't the sexy stuff"

A core part of data science is dealing with the collection, organization and cleaning of the data itself. As the goal of the independent study was to learn all of the facets that it takes to bring an analysis from start to finish, in this section we provide an overview of the many painstaking steps it took achieve a final data frame. Readers only interested in the results of the analysis should skip to sections II and III.

We sought to understand, if possible, how enrollments and drops are driven by (and if they could be predicted by) Penn Course Review ratings among a host of other relevant factors. Thus, this involved creating a data frame with enrollments and drops as a response matched to the relevant predictor values for a given class.

First, this required pulling data from Penn Course Review using the Penn Course Review API. The API was created and is maintained by Penn Labs (for documentation see http://pennlabs.org/docs/pcr.html). As this API is written in Python, and the Python language is intuitive and has very nice data munging modules, we pulled the data and did a large chunk of the cleaning in Python.

An in-depth discussion of the methods of the Penn Course Review API isn't warranted, but much time was spent trying understand its functions and calls. Full credit goes to Adel Qalieh, who very graciously provided the coursedatacompiler() function, when, after much effort, I still could not find a solution. The final script to pull information on all Wharton courses available is as follows:

```
def coursedatacompiler(course):
    excel = []
    stringname=str(course)
    course_listing={}
    course_reviews={}
    course_listing=requests.get("http://api.penncoursereview.com/v1//coursehistories/"
    +stringname+"?token=bafl3oTWt3Wh2tphQmSoHq2QvcbJX7").json()
    course_reviews=requests.get("http://api.penncoursereview.com/v1//coursehistories/"
    +stringname+"/reviews?token=bafl3oTWt3Wh2tphQmSoHq2QvcbJX7").json()
    for section in course_reviews["result"]["values"]:
        excel.append([section["section"]["id"].split("-")[0],section['section']['primary_alias'],
                      section["section"]["name"],section["instructor"]["name"], section["ratings"]["rInstructorQuality"],
        section["ratings"]["rCommAbility"],section['ratings']['rStimulateInterest'],section['ratings']['rAmountLearned'],
    id_to_semester=[(section["id"],section["semester"]) for section in course_listing["result"]["courses"]]
    temp = dict(id_to_semester)


    for row in excel:
        row.append(temp[int(row[0])])
    return excel
```

Not shown is the importing of the API call module "requests". We first define a course data compiler function. The function begins by querying the PCR API for information on a specific course (eg. STAT-101). This query returns a nested json object that contains all the course's historical data, including the ratings. The for-loop section of the function then parsed this object, appending each field of the course's information to the previously initialized list "excel." The remainder of the function deals with navigating the specifics of the json object that the PCR API method created, such that our list "excel" and the instance of the class can later be properly identified and returned when we want to put it into a dataframe.

```
masterlist=[]
for classes in ['ACCT-101',
 'ACCT-102',
 'ACCT-201',
 'ACCT-202',
 'ACCT-203',
 'ACCT-205',
 'ACCT-208',
 'ACCT-230',
 'ACCT-242',
 'ACCT-243',
 'ACCT-297',
 'ACCT-411',
 'ACCT-412',
 'BEPP-201',
 'BEPP-203',
 'BEPP-206',
 'BEPP-210',
 'BEPP-212',
 'BEPP-214',
 'BEPP-220',
 'BEPP-230',
 'STAT-401',
 'STAT-402',
 'STAT-411',
 'STAT-430',
 'STAT-431',
 'STAT-432',
 'STAT-433',
 'STAT-434',
 'STAT-435',
 'STAT-436',
 'STAT-451',
 'STAT-452',
 'STAT-453',
 'STAT-454',
 'STAT-471',
 'STAT-473',
 'STAT-474',
 'STAT-475',
 'STAT-476',
 'TRAN-201']:
   try:
       masterlist.append(coursedatacompiler(classes))
   except: ValueError
   print(ValueError)
```

Armed with our “coursedatacompiler” function, we then began to loop it over all the unique classes that Wharton has had in the past 14 years. These were taken from the course name section of our data frame by Wharton. There were 392 unique classes over these 14 years. However, as we discovered the hard way, not all of the classes have PCR data available for them. So, we implemented a try-catch block, that appended all of the course’s historical information to a previously initialized master list, and if we tried to query for a class PCR didn’t have data for, it simply printed the error. Of the 392 distinct courses Wharton has data for, we had 119 ValueErrors, returning us with a historical information for 273 unique courses.

```
import xlsxwriter
workbook = xlsxwriter.Workbook('originalpull20150905.xlsx')
worksheet = workbook.add_worksheet()
#getting semester and department
row=0
col=0
for i in range(len(masterlist)):
    for uniqueid,class_alias,instructor,name,rInstructorQuality,rCourseQuality,rCourseDifficulty,rCommAbility,rStimulateInterest,
        rAmountLearned, rRecommendNonMajor,rRecommendMajor,rWorkRequired,rReadingsValue,rInstructorAcess,semester in (masterlist[i]):
         worksheet.write_string  (row, col, uniqueid)
         worksheet.write_string(row, col + 1, class_alias)
         worksheet.write_string(row, col + 2, instructor)
         worksheet.write_string(row, col + 3, name)
         worksheet.write_string(row, col + 4, rInstructorQuality)
         worksheet.write_string(row, col + 5, rCourseQuality)
         worksheet.write_string(row, col + 6, rCourseDifficulty)
         worksheet.write_string(row, col + 7, rCommAbility)
         worksheet.write_string(row, col + 8, rStimulateInterest)
         worksheet.write_string(row, col + 9, rAmountLearned)
         worksheet.write_string(row, col + 10, rRecommendNonMajor)
         worksheet.write_string(row, col + 11, rRecommendMajor)
         worksheet.write_string(row, col + 12, rWorkRequired)
         worksheet.write_string(row, col + 13, rReadingsValue)
         worksheet.write_string(row, col + 14, rInstructorAcess)
         worksheet.write_string(row, col + 15, semester)
         row += 1
workbook.close()
```

We then wanted to turn this master list of historical course information into a workable dataframe that could be exported to Microsoft Excel. First the handy xlsxwriter module was used to initialize a new excel workbook. We then looped through each field of information for each class in masterlist and put these put each value of the various fields in a cell of a row for a specific instance of a class. At the end of the loop the row value then accumulated in value, and the script began recording the information for the next class on the next line of the spreadsheet. Row by row, this script built out our Excel dataframe and then exported it. In order to merge this with the dataframe of enrollments provided by Wharton, we had to create unique IDs to identify each unique instance a class had been taught at. The most granular (and thus unique) identifier was of the form Year-Semester-Subject-Course Number-Section

number. We used concatenate functions in Excel to append the relevant fields together in both spreadsheets to prepare for final merging.

There's a twist, however. When querying for information on a class, the Penn Course Review API sometimes returns the information by the course's cross-listed name. For example, when querying for information on STAT-475, Penn Course Review returns information for STAT-975. Our enrollment data only had courses named by their undergraduate code, so we would lose data in the matching process. In order to correct for this, we created a "dictionary" in Excel of graduate cross-listings and their undergraduate equivalents. Then using V-Lookups we replaced each instance of a graduate cross-listing with its undergraduate equivalent.

The PCR data had 6566 instances of classes; Wharton enrollment data had 6290 instances. To join the two frames together, we imported both dataframes into Python, and used the lovely Pandas module to join them on unique IDs like so:

```
enrollmentData = pd.read_csv('EnrollmentData20150906.csv')
PCRData = pd.read_csv('247classes20150905.csv')
df=pd.merge(enrollmentData,PCRData)
```

(The road to discovering Pandas was a long and arduous one—it's was too much of a divergent process to discuss, but we spent a lot of time trying a variety of solutions to combine the data frames with set logic, dictionaries, and other things)

The merged data frame had 4098 unique instances. What happened? The second twist is that, rather arbitrarily, if a course has 2 cross-listings, the API will return 3 exactly identical observations. For example, International Housing Comparisons is listed as REAL-236, FNCE-236, and BEPP-236. In fall 2010 if this class was taught, Penn Course review would return 4 identical observations with the ID 2010C-REAL-236-401. Over the 20 or so classes with cross-listed names, this added up to about about 875 instances that were deleted. The other 1400 or so instances we don't have? Very tough to say. Certainly some of it can be attributed to, over the years, human error in uploading course reviews or maintaining enrollment data. We ourselves found that Econ010 was not included in the Wharton Enrollment data, but it was in the Penn Course Review repository. And then Penn Course Review only went online in 2009—perhaps in the mass upload a good bit of information was lost. It's possible that some of those could have been duplicate observations in the manner we noted above. While it's unlikely that that's enough to explain all 1400 lost instances, after scouring the data set and not finding much more, we choose to move on.

After this, there were two key issue to grapple with:

1) Co-taught classes
2) Cross-listings

Co-taught classes: 2, and sometimes 3 Wharton professors will sometimes join to co-teach a class. We discovered these co-taught classes upon closer observation, when finding that our data frame had 4098. This should imply 4098 distinct unique IDs. However, it turns out that only 3868 of those were unique. Co-taught classes have the same unique ID, but different professors! Penn Course Review records these as 2 or 3 separate observations, one for each professor, with only ratings the fields specific to the instructor differing. In order to correct for this, we wrote the following script, and gratefully acknowledge the help of Andrew Shichman in evening out the kinks:

```
def Find_Dual_CoTaughts_Condense_To_One(dfOfDuplicateListings):
    IDFrame=dfOfDuplicateListings['Unique.ID']
    repeatedIDs=set(IDFrame)
    dfOfCoTaughts=pd.DataFrame()
    counter=0
    for ID in repeatedIDs:
        ListofOnes=[] #a one will be appended if it's different
        dfToCheck=dfOfDuplicateListings.loc[(dfOfDuplicateListings['Unique.ID']==ID)]
        if len(dfToCheck)==2:
            for j in range(len(dfToCheck.columns)-1):
                if dfToCheck.iloc[0,j]==dfToCheck.iloc[1,j]:
                    ListofOnes.append(0)
                else:
                    ListofOnes.append(1)
        if sum(ListofOnes)<=5:
            dfToConcat=pd.DataFrame(index=[counter], columns=dfToCheck.columns)
            for j in dfToCheck.columns:
                if type(dfToCheck[[j]].iloc[0][0])==str and type(dfToCheck[[j]].iloc[1][0])==str:
                    if j == 'instructor':
                        dfToConcat[j].loc[counter]=dfToCheck[[j]].iloc[0][0] + '/' + dfToCheck[[j]].iloc[1][0]
                    else:
                        dfToConcat[j].loc[counter]=dfToCheck[[j]].iloc[1][0]
                else:
                    try:
                        dfToConcat[j].loc[counter]=(dfToCheck[[j]].iloc[0][0]+dfToCheck[[j]].iloc[1][0])/2
                    except: Exception
            dfOfCoTaughts=pd.concat([dfOfCoTaughts,dfToConcat],axis=0)
            counter += 1
    return dfOfCoTaughts
```

From a high level, the script goes through to a data frame of classes that we've identified as potentially being co-taught. It then goes through and checks how many differences exists between a set of observations with the same Unique ID. If it counts less than 5 (the max number of instructor-specific fields) it then proceeds to combine the two observations, averaging the instructor ratings and listing both instructors with a slash between them. The remaining items, which had the same Unique ID but completely different ratings in every field, were discarded as data errors.

Cross-listings: This gave us quite a bit of grief. Many classes are registered with more than one department—Negotiations is MGMT-291, OPIM-291, and LGST-206. First, we went through our 247 classes from Penn course review and cataloged all possible cross-listings each could have. Graduate cross-listings were already taken care of in an earlier step, so those were ignored. Armed with a list of cross-listed classes, how would we deal with the issue? We decided to a cross-listed class to the department its professor came from. This would be judged by all the non-cross-listed classes the Professor taught. If a Negotiations professor taught primarily legal studies classes, his instance of Negotiations would classified as LGST-206. We first pulled all the classes that had the potential to be cross-listed, then wrote a script to find a professors teaching subjects that were not in their department

```
def FindProfNotInHisDept(crosslistedsfromdf,FinalProfDict):
    listofsections=[]
    for i in range(len(crosslistedsfromdf)):
        if crosslistedsfromdf.iloc[i,20]!=FinalProfDict[crosslistedsfromdf.iloc[i,6]]:
            listofsections.append(crosslistedsfromdf.iloc[i,0])
    return listofsections

listofsections=FindProfNotInHisDept(crosslistedsfromdf,FinalProfDict)
```

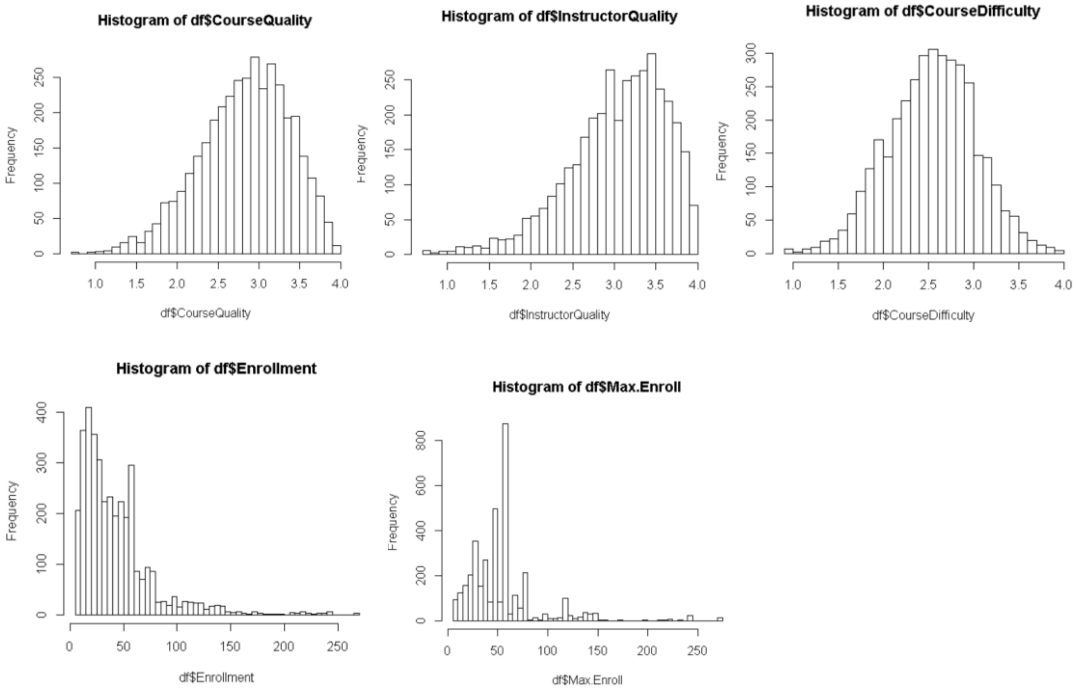
Using these results, we then went through and manually subset on each professor to see what classes he taught, and assign observations accordingly. While we recognize this was a brute force way, coding the decision cases across departments ending up being too complicated to be worthwhile. With these re-assigned we had our final data frame of 3700 observations.

A Quick Look at Important Components of the Data.

We take a preliminary look at some key data:

```
   Enrollment
Min.    :  5.00
1st Qu.: 20.00
Median : 35.00
Mean   : 44.09
3rd Qu.: 57.00
Max.    :270.00
```

```
CourseQuality   InstructorQuality CourseDifficulty
Min.   :0.790   Min.   :0.710     Min.   :0.930
1st Qu.:2.480   1st Qu.:2.700     1st Qu.:2.220
Median :2.880   Median :3.130     Median :2.570
Mean   :2.835   Mean   :3.045     Mean   :2.545
3rd Qu.:3.240   3rd Qu.:3.480     3rd Qu.:2.880
Max.   :4.000   Max.   :4.000     Max.   :4.000
```



Interesting. Despite how much people complain about their classes, by and large they give their instructors and classes decent ratings. We see our response, Enrollments, is quite skewed. The spike on Max Enrollments is attributed to the fact that most of the tiered classrooms in Wharton buildings have a maximum capacity of 78 people.

We present the correlation matrix:

	Enrollment	Max.Enroll	CourseQuality	InstructorQuality	CourseDifficulty	AmountLearned	WorkRequired
Enrollment	1.000000000	0.9018148230	-0.217022519	-0.09246651	0.1398531943	-0.1762037453	0.002717364
Max.Enroll	0.901814823	1.0000000000	-0.272518095	-0.15143688	0.1569592263	-0.2146949932	0.052905318
CourseQuality	-0.217022519	-0.2725180946	1.000000000	0.90502833	-0.0527775970	0.9060649043	-0.006339885
InstructorQuality	-0.092466508	-0.1514368785	0.905028332	1.00000000	-0.0560715195	0.7953801068	-0.052244543
CourseDifficulty	0.139853194	0.1569592263	-0.052777597	-0.05607152	1.0000000000	0.1616276442	0.712564393
AmountLearned	-0.176203745	-0.2146949932	0.906064904	0.79538011	0.1616276442	1.0000000000	0.136495513
WorkRequired	0.002717364	0.0529053176	-0.006339885	-0.05224454	0.7125643927	0.1364955131	1.000000000
ReadingsValue	-0.297195005	-0.3004619963	0.715977385	0.61768155	-0.0383171720	0.7178061520	0.080184251
CommAbility	-0.086038009	-0.1419812250	0.883445890	0.95655255	-0.0824069894	0.7828989659	-0.081149190
InstructorAccess	-0.277227628	-0.2858151010	0.641274691	0.67971104	-0.0405258962	0.5951990830	0.044503497
StimulateInterest	-0.126918016	-0.1809073276	0.873554833	0.92297200	-0.1365787708	0.7476414201	-0.066445948
RecommendMajor	-0.049741287	-0.1127090716	0.860704713	0.78887384	0.0555774433	0.8502355608	0.005526028
RecommendNonMajor	-0.178640850	-0.2291854165	0.801328726	0.72086435	-0.3373453800	0.7033272532	-0.240897272

A few things to note. The first is that Wharton classes tend to fill quite close to maximum. With the correlation between maximum enrollment and enrollment at .9, we feel safe saying that generally our school runs at full capacity. Instructor Quality is highly correlated with Course Quality, which is some solid evidence that the professor really does make the class. However, apparently Course Difficulty has little correlation with either. We might take this in an encouraging way, perhaps that Wharton students are mature enough to separate how difficult the class is with how well it was taught. The ancillary predictors all have high degrees of correlation with each other, and the quality of the class and instructor. How clearly the instructor can communicate is clearly part of his quality rating, and the correlation between the two (.95) confirms this intuition. It also means that the ancillary predictors are likely to be pretty useless, and will only increase what is already bad multi-collinearity, something we will present concrete evidence of shortly.
We now conduct some simple regressions:

Enrollment~Course Quality

```
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       83.713      3.000   27.91   <2e-16 ***
df$CourseQuality  -13.963      1.039  -13.44   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 34.73 on 3697 degrees of freedom
Multiple R-squared:  0.04658,   Adjusted R-squared:  0.04632
F-statistic: 180.6 on 1 and 3697 DF,  p-value: < 2.2e-16
```

Enrollment~Instructor Quality

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           61.184      3.115  19.644  < 2e-16 ***
df$InstructorQuality  -5.600      1.005  -5.572  2.7e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 35.42 on 3697 degrees of freedom
Multiple R-squared:  0.008327,  Adjusted R-squared:  0.008059
F-statistic: 31.04 on 1 and 3697 DF,  p-value: 2.702e-08
```

Enrollment~Ratings

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           45.473      4.243  10.718   <2e-16 ***
df$CourseQuality     -47.489      2.340 -20.295   <2e-16 ***
df$CourseDifficulty    9.762      1.125   8.675   <2e-16 ***
df$InstructorQuality  35.611      2.220  16.043   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 33.3 on 3695 degrees of freedom
Multiple R-squared:  0.1241,    Adjusted R-squared:  0.1234
F-statistic: 174.6 on 3 and 3695 DF,  p-value: < 2.2e-16
```

Our ratings are statistically significant, meaning they do explain a portion of the variation in enrollments that we can say with 95% likelihood (and often much greater) is not due to chance—any time Enrollments are regressed onto any combination of ratings, the overall model comes out statistically significant with a p value of 10^-16. Of course, this doesn't mean that the ratings are good at explaining enrollments, or even make sense. As we see above, the univariate correlations between Course Quality and Enrollments, as well as Instructor Quality and Enrollments, are actually *negative.* Course difficulty actually has a positive univariate relationship with Enrollments. Do people prefer difficult material, poor professors, and worse classes? Probably not. These strange slope coefficients are indication that much more factors are at play when looking to explain variation in course enrollments. Furthermore, as the correlation matrix showed, Instructor Quality is highly correlated with Course Quality, which means multi-collinearity will be an issue when combining predictors for more complete models, further reason why the model with three predictors has the unstable slope coefficients it does (though note that despite the high correlation both still come out as very significant). So we move to building models that capture more of the factors students are probably looking at when choosing classes.

Section II: Modeling Course Enrollments

We have a feeling that our ancillary predictors will not be terribly useful. To demonstrate this, quick and dirty, we throw everything we have into the model and present some highlights of the output:

Enrollments~All predictors

Abbreviated output:

```
Call:
lm(formula = df$Enrollment ~ df$Year + df$Fall.Spring + df$Subject +
    df$Section.. + df$Core + df$grad.crosslisting. + df$CourseQuality +
    df$InstructorQuality + df$CourseDifficulty + df$AmountLearned +
    df$WorkRequired + df$ReadingsValue + df$CommAbility + df$InstructorAccess +
    df$StimulateInterest + df$RecommendMajor + df$RecommendNonMajor)

Residuals:
   Min     1Q Median     3Q    Max
-82.74 -14.45  -1.73  11.17 178.08
```

```
Residual standard error: 25.65 on 3636 degrees of freedom
Multiple R-squared:  0.4885,    Adjusted R-squared:  0.4798
F-statistic: 56.01 on 62 and 3636 DF,  p-value: < 2.2e-16
```

And the VIF table

	GVIF	Df	GVIF^(1/(2*Df))
df$Year	1.757252	12	1.023768
df$Fall.Spring	1.077620	1	1.038085
df$Subject	12.167375	10	1.133078
df$Section..	6.632980	26	1.037056
df$Core	2.238578	1	1.496188
df$grad.crosslisting.	1.324190	1	1.150735
df$CourseQuality	19.446563	1	4.409826
df$InstructorQuality	21.496330	1	4.636413
df$CourseDifficulty	4.459899	1	2.111847
df$AmountLearned	10.127693	1	3.182404
df$WorkRequired	3.301300	1	1.816948
df$ReadingsValue	2.826304	1	1.681161
df$CommAbility	14.211860	1	3.769862
df$InstructorAccess	2.292427	1	1.514076
df$StimulateInterest	9.809454	1	3.132005
df$RecommendMajor	6.602471	1	2.569527
df$RecommendNonMajor	5.852262	1	2.419145

What we take from this:

- The r^2 is .48; we are explaining half the variance in enrollments. Of course, we haven't assessed assumptions.
- VIFs are off the charts! Abiding by the rule of thumb that a VIF of 5 is cause for concern, and 9 of our predictors have VIFs greater than that. It also makes sense that the VIFs of Instructor Quality and Course Quality are the highest. Recall that VIFs are a function of the adjustment regression, that is, the kth predictor regressed on the other k-1 predictors. The set of ancillary predictors,

with things like Instructor Access and Ability to Stimulate Interest of course predict Course and Instructor Quality well.

- We will remove the ancillary predictors for the above and one other reason: no one knows about them! On Penn Course Review, one has to adjust their settings to see these ratings, which not many people know how to do. Only in doing this report did I even learn that they existed. Thus, we posit that these ratings can't add much signal if they're not typically considered by students.
- From here onwards, we will only consider the core predictors.

Baseline: Enrollments~Core predictors

To create a baseline for comparison, we regress Enrollment on core predictors.

The abbreviated output:

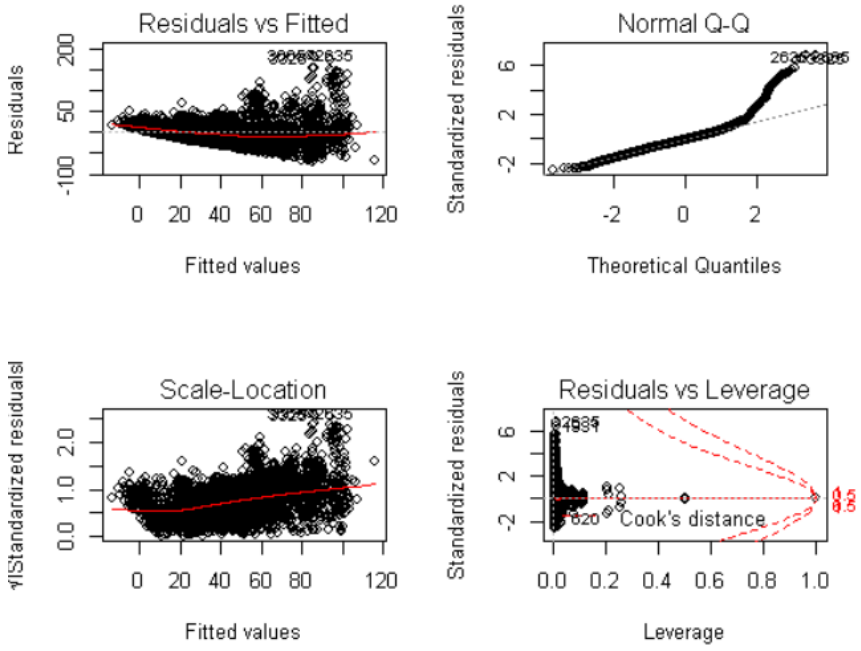
```
Call:
lm(formula = df$Enrollment ~ df$Fall.Spring + df$Subject + df$Section.. +
    df$Core + df$grad.crosslisting. + df$CourseQuality + df$InstructorQuality +
    df$CourseDifficulty)

Residuals:
    Min      1Q  Median      3Q     Max
-70.369 -15.532  -2.476  11.026 184.445

                          Estimate Std. Error t value Pr(>|t|)
df$Core1                  41.02916    1.35749  30.224  < 2e-16 ***
df$grad.crosslisting.1    -4.36930    2.39372  -1.825  0.06803 .
df$CourseQuality         -19.42923    2.22831  -8.719  < 2e-16 ***
df$InstructorQuality      17.97503    1.99782   8.997  < 2e-16 ***
df$CourseDifficulty        0.84912    1.12168   0.757  0.44909
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 27.18 on 3656 degrees of freedom
Multiple R-squared:  0.4231,    Adjusted R-squared:  0.4156
F-statistic: 55.87 on 48 and 3656 DF,  p-value: < 2.2e-16
```

Diagnostics:



Vifs:

	GVIF	Df	GVIF^(1/(2*Df))
df$Fall.Spring	1.034233	1	1.016973
df$Subject	5.364971	10	1.087623
df$Section..	5.140756	32	1.025911
df$Core	2.074442	1	1.440292
df$grad.crosslisting.	1.250712	1	1.118352
df$CourseQuality	7.528344	1	2.743783
df$InstructorQuality	6.724067	1	2.593081
df$CourseDifficulty	1.498391	1	1.224088

```
> durbinWatsonTest(EnrollmentOnCore)
 lag Autocorrelation D-W Statistic p-value
   1       0.3464844      1.306866       0
 Alternative hypothesis: rho != 0
```

Not that great. The some non-linearity, terrible homoscedasticity, residuals veer far from normal and predictors are very collinear. As we will not be transforming any predictors, this will persist throughout our analysis and makes inference difficult, as our slope coefficients become unstable and nonsensical. Indeed, the coefficient of course quality becomes negative. Residuals are autocorrelated. We technically should not interpret this model's coefficients without showing that we've reasonably satisfied assumptions, consider that the p-values of being in the Wharton Core, instructor Quality, and Course Quality are essentially zero. While we acknowledge that we cannot claim to have unbiased standard errors, they will probably significant no matter what we corrections apply. We can; however, look at the VIFs since they are simply functions of the predictors. Subject and Section are just above the warning threshold, which is interesting. Course Quality and Instructor quality are again fairly high, which is to be expected. Course Difficulty however, is not. But also notice how in the effects table it is not even significant controlling for the other predictors. As we continue to try and improve this model, it will be interesting to see whether this result holds.

When heteroscedasticity increases as x increases, we log the response as a fix. We do that next.

Log Enrollments~Core Predictors

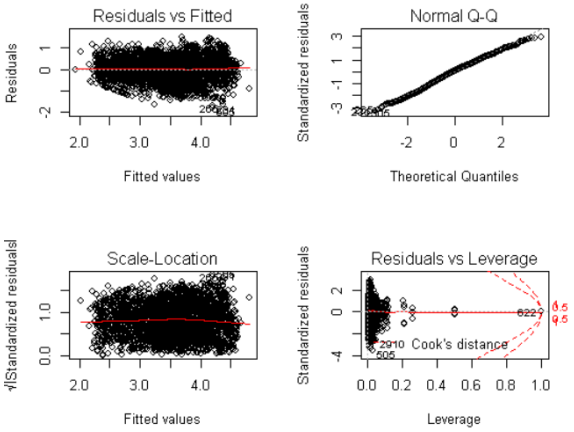
```
Call:
lm(formula = log(df$Enrollment) ~ df$Year + df$Fall.Spring + 
    df$Subject + df$Section.. + df$Core + df$grad.crosslisting. + 
    df$CourseQuality + df$InstructorQuality + df$CourseDifficulty)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.79118 -0.34053  0.03368  0.34475  1.49981 
```

```
                          Estimate   Std. Error  t value  Pr(>|t|)    
df$Core1                  0.663268   0.026161   25.354   < 2e-16 ***
df$grad.crosslisting.1   -0.295563   0.046932   -6.298 3.38e-10 ***
df$CourseQuality         -0.398088   0.043020   -9.253   < 2e-16 ***
df$InstructorQuality      0.350338   0.038611    9.074   < 2e-16 ***
df$CourseDifficulty      -0.054903   0.021706   -2.529 0.011467 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5232 on 3644 degrees of freedom
Multiple R-squared:  0.5036,    Adjusted R-squared:  0.4954 
F-statistic: 61.62 on 60 and 3644 DF,  p-value: < 2.2e-16
```

Diagnostics:




```
> durbinWatsonTest(LogEnrollmentOnCore)
 lag Autocorrelation D-W Statistic p-value
   1       0.2455715      1.508412       0
 Alternative hypothesis: rho != 0
```

```
a<-fitted(LogEnrollmentOnCore)
r2ComparableLogEnrollmentOnCore<-1-(sum((exp(a)-df$Enrollment)^2)
                                    /sum((df$Enrollment-mean(df$Enrollment))^2))
```

```
> r2ComparableLogEnrollmentOnCore
[1] 0.3881281
```

Much better overall in terms of satisfying assumptions. Non-linearity and heteroscedasticity are gone Normality of residuals is much better satisfied. Residuals are still auto correlated, although less so from before. Here we see that this model explains about 50% of the variation of *percent changes* in Enrollments. We also acknowledge that we cannot compare model fits when the response is transformed, something we will address shortly. We have decently met OLS assumptions though, so it would be worth taking the opportunity to look at some highlights of these slopes and p-values.

- Course Difficulty is still not significant controlling for everything else.
- Core is still very significant. Since it has never suffered from multicollinearity, we will offer an interpretation –if a course is in the Wharton core, the estimated average enrollment increases by a factor of e^.66. That's an increase of 93%!
- Instructor and Course Quality are also significant, though we refrain from interpreting the quality variables as multicollinearity is clearly ruining these estimates.
- Being cross-listed as a graduate level course seems to matter, but again is it really meaningful? According to this model the average decrease in enrollment when a course is cross-listed as a graduate course is about 29%.
- Year is not significant, and neither is season.

Of course, we cannot compare r^2s across models when we have transformed the response variable. In order to get a comparable r^2 to our baseline model, we take the fitted values that LogEnrollmentsOnCore generated and transform them back to normal units by exponentation. We then subtract these from the true values, square the result (getting a sum of errors squared) and then divide that by the total squared "error" in the response relative to the mean. Subtracting this value from 1 gives us what we refer to as a "comparable" r^2, because it that can be legitimately compared against the result from our first regression. Despite satisfying all linear model assumptions better, apparently this model does not explain variation as well as the first did (comparable r^2 of .388). We will look at out-of-sample performance in Section III.

Recall that the correlation between maximum enrollment and enrollments was .91, implying that maximum enrollment explains about 83% of all variation in course in the untransformed response, something our other models have fallen quite short of (as measured by comparable r^2). We now turn to two transformations of the response that account for the maximum enrollment of the class in question. In order to control for this class size component, we will first consider taking enrollment as a percentage of max enrollment.

Enrollment/Max Enrollment~Core Predictors

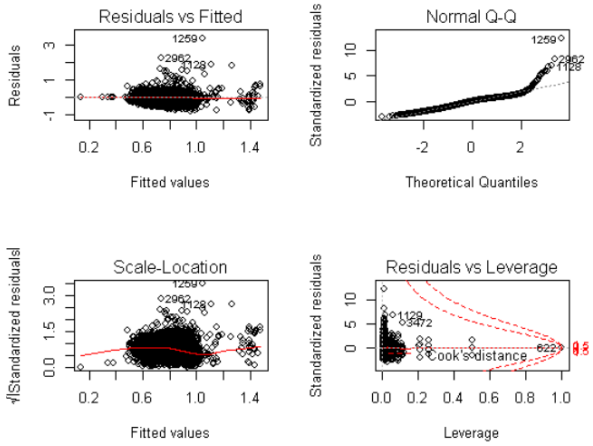
```
Call:
lm(formula = df$PercentEnrollment ~ df$Year + df$Fall.Spring +
    df$Subject + df$Section.. + df$Core + df$grad.crosslisting. +
    df$CourseQuality + df$InstructorQuality + df$CourseDifficulty)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7988 -0.1859  0.0344  0.1732  3.3513
```

```
df$Core1                 0.1246605  0.0138223   9.019  < 2e-16 ***
df$grad.crosslisting.1  -0.0527421  0.0247974  -2.127 0.033493 *
df$CourseQuality        -0.0224670  0.0227304  -0.988 0.323018
df$InstructorQuality     0.0783332  0.0204005   3.840 0.000125 ***
df$CourseDifficulty     -0.0856495  0.0114685  -7.468 1.01e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2765 on 3644 degrees of freedom
Multiple R-squared:  0.1665,    Adjusted R-squared:  0.1528
F-statistic: 12.13 on 60 and 3644 DF,  p-value: < 2.2e-16
```

Diagnostics:




```
> durbinWatsonTest(PercentEnrollmentOnCore)
 lag Autocorrelation D-W Statistic p-value
   1       0.2046335      1.590454       0
 Alternative hypothesis: rho != 0
```

```
a<-fitted(PercentEnrollmentOnCore)*as.vector(df$Max.Enroll)
r2ComparablePercentEnrollmentOnCore<-1-(sum((a-df$Enrollment)^2)
                                        /sum((df$Enrollment-mean(df$Enrollment))^2))
```

```
> r2ComparablePercentEnrollmentOnCore
[1] 0.8276226
```

There is no non-linearity, though we do quite poorly on a couple of observations, evinced by the distribution of residuals. The residuals also veer off of normality in the tails. We once again do a little better than before on auto-correlation. Since we have transformed Y, we follow a similar procedure as before to get a comparable r^2. This time, we do significantly better than before—comparable r^2 of .827! This model does well fitting the observed values once transformed back. But it's not clear that we should interpret the slope coefficients from this model as well. Our prediction intervals will be quite large. In terms of explaining the variance of its own response, it doesn't do well.

We should, however, pause and think about an implication of using percentage of maximum enrollment. Consider the scenario where enrollment in a class with a maximum enrollment of 200 has an enrollment of 160 yields a percentage of 80%. This is considered the same as a class where the maximum enrollment is 20 and the enrollment in one year is 16. Clearly, in the 1st case the loss is quite significant,

while in the second case that could simply be random variation from year to year. In order to correct for this issue, we consider a final response transformation where we transform the response to: (enrollments-mean(historical enrollments of class)/sqrt(n)

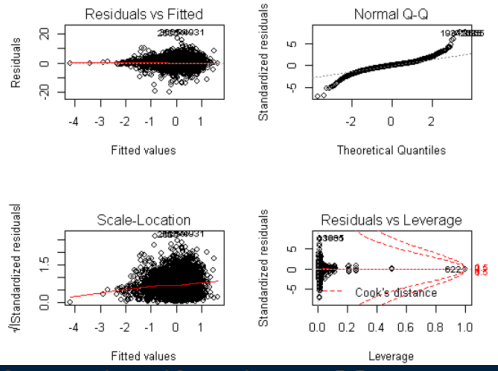
StandardizedEnrollmentDeviations~Core Predictors

```
Call:
lm(formula = df$StandardizedEnrollmentDeviations ~ df$Year +
    df$Fall.Spring + df$Subject + df$Section.. + df$Core + df$grad.crosslisting. +
    df$CourseQuality + df$InstructorQuality + df$CourseDifficulty)

Residuals:
    Min      1Q  Median      3Q     Max
-19.842  -1.354   0.043   1.296  21.096

                        Estimate Std. Error t value Pr(>|t|)
df$Core1                 0.16518    0.13931   1.186 0.235830
df$grad.crosslisting.1   0.37300    0.24992   1.492 0.135667
df$CourseQuality        -0.76004    0.22909  -3.318 0.000917 ***
df$InstructorQuality     0.96510    0.20561   4.694 2.78e-06 ***
df$CourseDifficulty     -0.53935    0.11559  -4.666 3.18e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.786 on 3644 degrees of freedom
Multiple R-squared:  0.03914,   Adjusted R-squared:  0.02332
F-statistic: 2.474 on 60 and 3644 DF,  p-value: 3.275e-09
```




```
> durbinWatsonTest(StandardizedEnrollmentDeviationsOnCore)
 lag Autocorrelation D-W Statistic p-value
   1       0.1853167       1.629299       0
 Alternative hypothesis: rho != 0
```

```
a<-(fitted(StandardizedEnrollmentDeviationsOnCore)*as.vector(sqrt(df$NumberOfObservations)))
    +as.vector(df$Average_Enrollment)
r2ComparableStandardizedEnrollmentDeviationsOnCore<-1-(sum((a-df$Enrollment)^2)
                                    /sum((df$Enrollment-mean(df$Enrollment))^2))
```

```
> r2ComparableStandardizedEnrollmentDeviationsOnCore
[1] 0.7679273
```

This transformation introduces some very bad heteroscedasticity into the mix, as well as ruins normality for the ends of the Normal Q-Q plot. On this issue we are inclined not to log the standardized units, because interpretability is already severely lacking as it is. The residuals are less correlated than before, and our comparable r^2 is also a quite good .767. But in terms of coefficients, we are interpreting how unit increases in the response affect unit increases of standard units. Does intuitively that mean much?

While a large promise of this project was to try and understand the relationship between enrollments and ratings, multi-collinearity and the need to correct for class size render this task quite difficult. Which model is to be believed? On the one hand the models correcting for enrollment produce fitted values that, once transformed, do a decent job of explaining variation in the enrollment of the original response. But they do poorly at explaining their own variation! It's not clear whether interpreting these slope coefficients would yield much, with such low r^2 meaning that variability around the fitted plane is high. This means that our confidence intervals would be quite wide; *however*, *it does not mean* the model and its effects could not have picked up on significant trends in the data. On this note, what trends do we observe that were consistent across our range of models?

- Year as a whole was never significant—it seems there is not much of a time trend to enrollments, and their underlying characteristics have held constant over the past 14 years.
- Fall vs. Spring was never significant. While the spring version of a class can be *very* different from the fall version of a class, it apparently does not make enough of a difference to change the pattern of enrollments systematically. What also may be at play is simply the fact that there are thousands of Wharton students picking classes each semester. Whether a class is better or worse in the spring or fall, by sheer numbers, there's always students who will take it. And even if less students choose to take a class in the "off" semester, it's not enough to be noticeable, even when taking class size into account.
- Whether a course was a core (required) class was always significant. While college may be the best years of their lives, at the end of the day, students want to graduate, and this means all else constant, they tend to pick required classes over non-required class twice as often.
- Instructor Quality is the same as Course Quality to people. Multicollinearity made slopes unstable, however Instructor and Course Quality almost always had p-values smaller than 10^-5.
- Course Difficulty was insignificant before our class size correction, but very significant after.
- The unsophisticated but canonical fix for multicollinearity is to remove the highly correlated predictors, or predictors that are insignificant. What happens when we remove Instructor Quality, or Course Quality from our selection of models?

We will disregard the models where we had not corrected for class size. Taking Instructor Quality out of the Standardized Enrollments model renders Course Quality insignificant:

df$CourseQuality	0.20923	0.09862	2.122	0.033941	*
df$CourseDifficulty	-0.53756	0.11591	-4.638	3.65e-06	***

However, taking Course Quality out of the Standardized Enrollment model gives us the intuitive result:

df$InstructorQuality	0.34756	0.08838	3.933	8.56e-05	***
df$CourseDifficulty	-0.55401	0.11562	-4.792	1.72e-06	***

As does taking Course Quality out of the Percent Enrollment model:

df$InstructorQuality	0.0601214	0.0087572	6.865	7.76e-12	***
df$CourseDifficulty	-0.0861595	0.0114569	-7.520	6.84e-14	***

As does taking Instructor Quality out of Percent Enrollment:

df$CourseQuality	0.0563619	0.0097758	5.765	8.82e-09	***
df$CourseDifficulty	-0.0857194	0.0114901	-7.460	1.07e-13	***

But, which one to report as the true effect? Giving 3 estimations of the impact of a given rating on Enrollments is hardly enlightening. Not to mention these responses are in different units which further

complicates understanding for anyone interested. And what about the fact that the r^2 on the Standardized Enrollment model was 3.9%, and 16.6% on the Percent Enrollments model? The models are not too good at explaining the variation in their own responses. If we were to calculate confidence/prediction intervals, they would be quite wide and likely not that useful. Another approach would have been to use Principal Components Analysis to combine these ratings together. But this would severely hamper interpretability, which was the point in the first place.

So we can establish that the ratings do explain a non-zero portion of the variation in enrollments regardless of how the response is transformed, as across all models they were almost always very significant. Unfortunately, it would seem that, at the present time, it's quite difficult to discern much beyond that.

Perhaps we can gain some insight from our predictive procedures.

Section III: Prediction

We now turn to seeing if we can predict enrollments from Penn Course Review ratings and other relevant factors. We will use Breiman's Random Forests Package in R as well as OLS. The performance metric will be out-of-sample (OOS) RMSE.

It is important to understand the mindset of pure prediction. Here we only less of an interest in understanding the relationship between the response and the predictors; the primary goal is as accurate predictions as possible. We like Breiman's Random Forests procedure—particularly because the random sampling of predictors it employs means our weak predictors will, over a large number of trees, be able to contribute whatever local signal they contain. Since we have many collinear predictors that are significant on their own but don't perform well controlling for the other predictors, this procedure is appealing.

We have 3700 data points, a good number for cross-validation without fear of having too few observations either in the training or hold out data. We will make a slight modification to the typical classic cross-validation scheme, however. Since enrollment data comes in yearly, rather than randomly across years, our cross validation will to hold out a given year's worth of data, train on all other years, and assess performance on the held out year. This more accurately mirrors the context in which the data are generated. This will necessarily force us to leave out the categorical variable Year when we are running the algorithm, however year has consistently proved insignificant, which we believe is sufficient evidence to assume that enrollment patterns don't systematically vary over time. We will try out a few different models, including with the response in standard normal units (enrollments for a given class-mean(historical enrollments for that class))/SD(historical enrollments for that class). In order to compare performance across models and procedures, we will take the fitted values each model generates and re-transform them back to normal units. Then, using the normal units, we calculate an MSE for the OOS data. Taking the square root of this gives us RMSE, or, on average, how many students off are our predictions. We then average these RMSEs across years to give us an average RMSE for that model, which can be compared to others that have transformed responses.

Also, as we are simply seeking the best prediction possible, we will include maximum enrollment as a predictor itself. We did not include max enrollment as a predictor in the modeling portion since it was so strong by itself, we feared it would crowd out the contributions of other variables and make them insignificant. But again, here the primary purpose is not to understand, but predict. There are subject matter justifications too: it may be harder for larger classes to hit capacity (percentage wise) since they are so large, or perhaps it's much easier since they are the large required courses everyone must take. Thus, knowing the size of the class would lend some information even towards predicting enrollments on a size class-relative basis.

A sampling of code follows. Everything is on the set of core predictors defined earlier in this paper unless otherwise specified:

```
df$EnrollmentStandardUnits<-((df$Enrollment-df$Average_Enrollment)/df$StandardDeviationEnrollment)
RF_StandardUnits_OOS_mse_each_semester<-c()
RF_StandardUnits_OOS_rmse_each_semester<-c()
for (Year in levels(df$Year)){

  df_temp<-df[!df$Year==Year,]
  df_temp<-na.omit(df_temp)
  procedure_temp<-randomForest(EnrollmentStandardUnits~Max.Enroll+Fall.Spring+Subject+Section..
                               +Core+grad.crosslisting.+CourseQuality+
                                 InstructorQuality+CourseDifficulty,data=df_temp,importance=T) # train
  holdout<-na.omit(df[df$Year==Year,]) # create hold out data
  score<-predict(procedure_temp,newdata=holdout) # fitteds from test on hold out
  a<-score*as.vector(holdout$Standard_Deviation_Enrollment)+as.vector(holdout$Average_Enrollment) # transform fitteds
  mse<-sum((a-holdout$Enrollment)^2)/nrow(holdout) # compare transformed fitteds to regular to get mse
  rmse<-sqrt(mse)
  RF_StandardUnits_OOS_mse_each_semester<-c(RF_StandardUnits_OOS_mse_each_semester,mse) #  mse each semester
  RF_StandardUnits_OOS_rmse_each_semester<-c(RF_StandardUnits_OOS_rmse_each_semester,rmse) # rmse each semester
}
```

We perform a similar procedure for our set of models, and the mean OOS RMSEs over 13 yearly folds are as follows:
OLS with Enrollments as response (no class size correction): 27.14
OLS with Enrollments standardized by sqrt(n) as response: 16.23
OLS with Percent Enrollment as response: 14.92
OLS with Enrollments in Standard Units as response, **no other predictor except Max Enrollment**: 16.74
Random Forests with Enrollments in Standard Units as response: 13.46
Random Forests excluding Max Enrollment as a predictor with Enrollments in Standard Units as response: 15.487
Random Forests with Enrollments standardized by sqrt(n) as response: 13.23

If we are after predictive ability, one might expect us to include all the ancillary predictors that we left off. We tried a few models with everything (not shown) and found no significant change in the average RMSEs above.

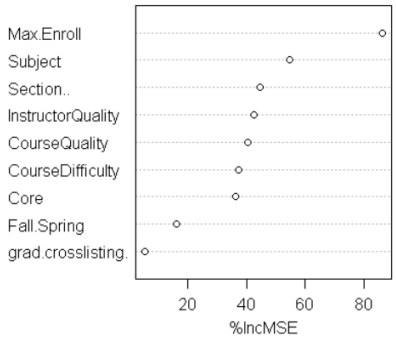
As we can see, with no class size correction we do quite poorly, missing on average 27 students per prediction. But, Random Forests and OLS performance is relatively similar out of sample here, with the difference once we apply the class size correction being at most 3 students (16 vs. 13). Mispredicting a class's enrollment by, on average, 13 students is pretty poor. Particularly when we consider that the average class size of our data set is 43; on average we're 13/43=30% off. Just using Max Enrollments as one predictor does just as well all the core predictors combined!

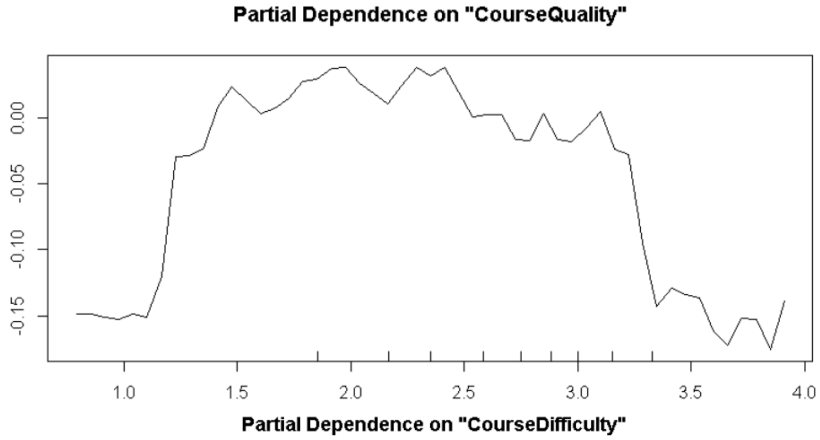
Why is this occurring? To be quite frank, we're not sure. In some models we sought to account for class size twice, by transforming our response to standard units as well as including the Maximum Enrollment as a predictor itself. Yet, we are still doing poorly. One conjecture that might be plausible is that there are just too many outliers that are large enough to make it difficult to predict results well across the board.

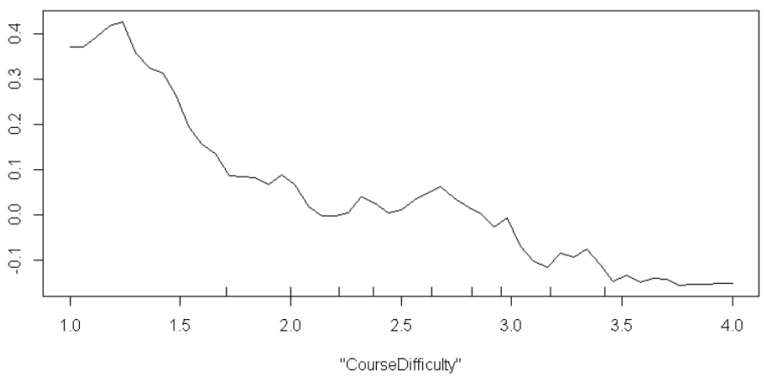
Now, we'd like to see what descriptive things we can learn when we train Random Forests on the all the data except for the most recent year, 2014. We will examine the variable importance plots, and the partial plots.

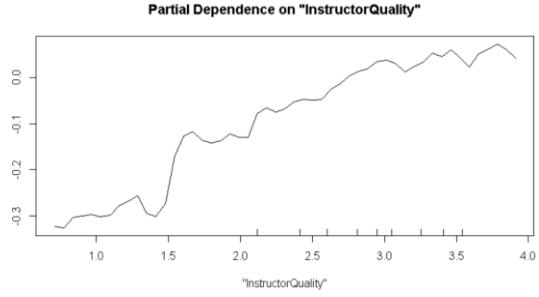
We will choose standard normal units as the response for interpreting the partial plots. It's not clear what the best response unit would be to look at when generating the variable importance and partial plots. Given the issue first raised about percent enrollment as a response (treating 160/200 the same as

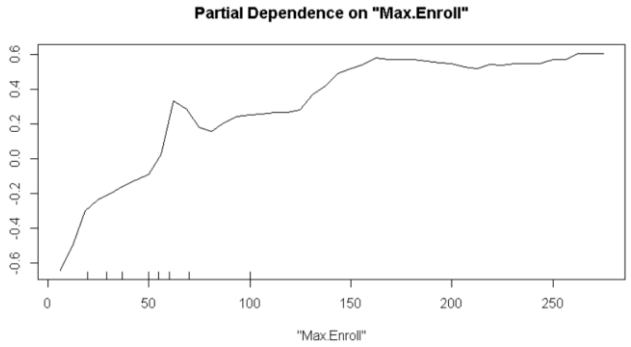
16/20) we will look at the standard normal units as the response. We present the variable importance plot and selected partial plots:

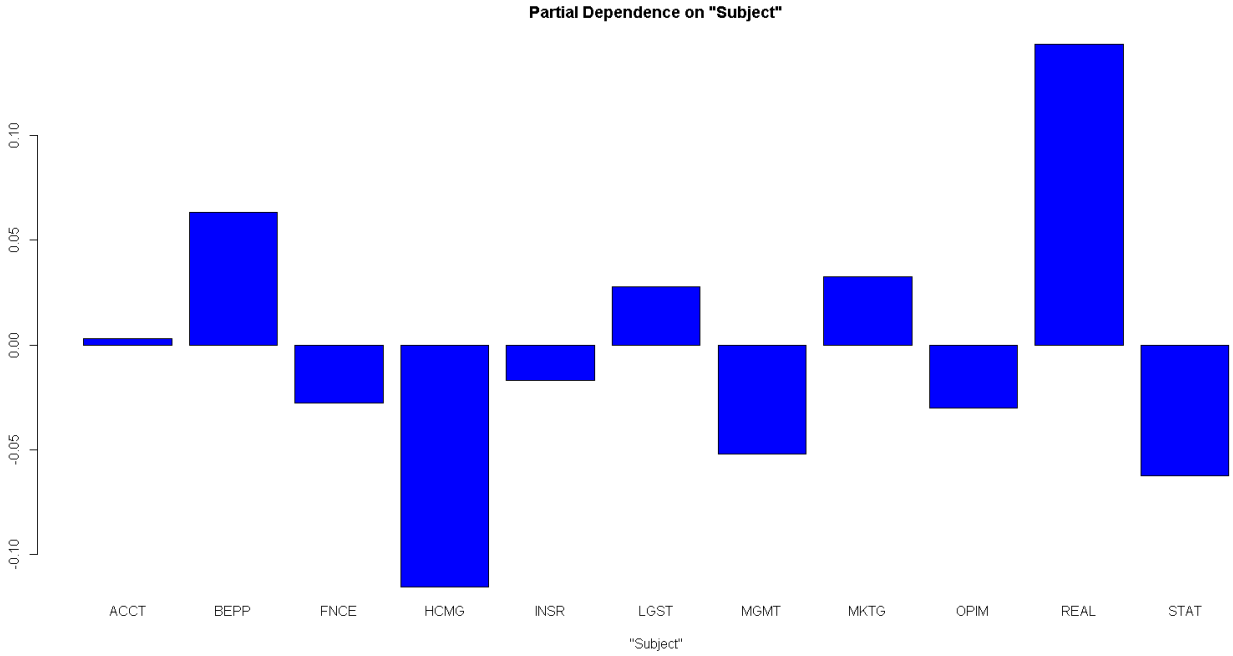






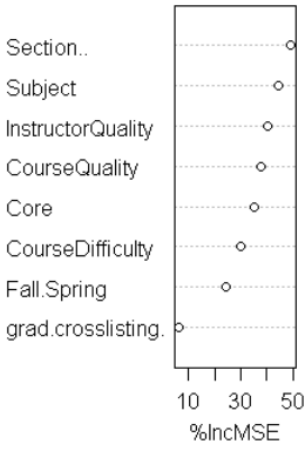


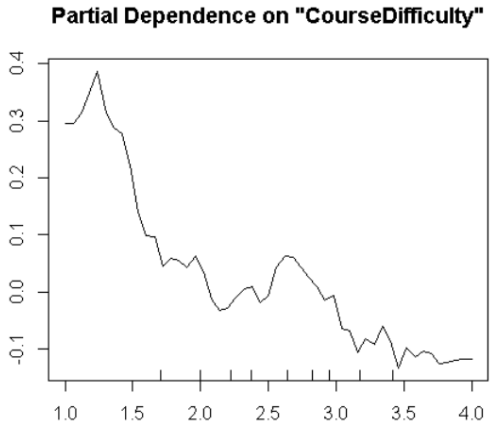


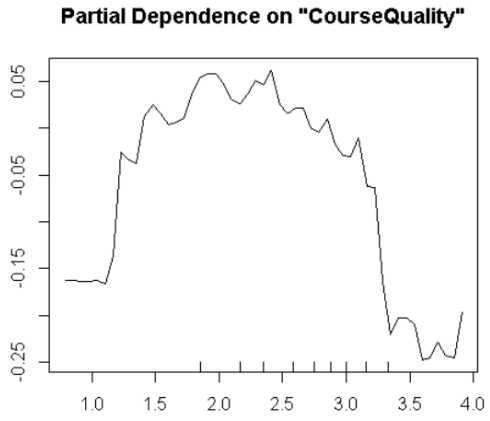


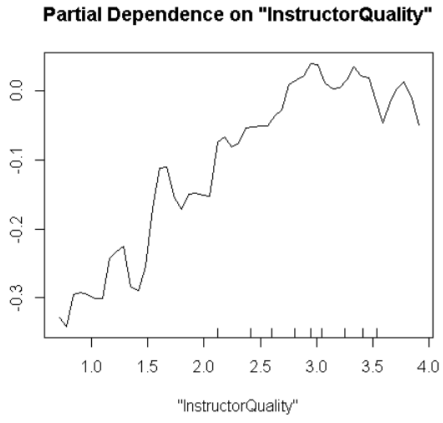
It's interesting to after enrollment, Subject and Section are the best predictors, but core is much further down. Most all of them hover at 40% of MSE though, so they all seem to be contributing. Al l of the partial plots, except for Course Quality, follow the trend we would expect. The effects of a larger class size seem to level off after about 170. And we report the Subject partial plot simply for interest. Apparently people like Real Estate.

For completion's sake, as we did before, we look at the variable importance plot when max enrollment is not included and present selected plots.









The variable importance plot is fairly similar. We see that Course Difficulty and Instructor Quality generally exhibit the expected pattern, although it's not clear what is going on with Course Quality.

Conclusions

We originally set out to see to what degree the variation in Enrollments can be explained by Penn Course Review Ratings, as well as if Penn Course Review ratings could be used to predict enrollment in classes (while of course taking into account other relevant factors such as whether the class is a requirement to graduate). In short, the answer to the first is “a fair amount, but not as much as one might think” and the answer to the second is “no, not well.” If one had to guess, the biggest issue facing both components of this analysis might be outliers in class enrollment patterns. Over the past 14 years, there have been some 30 or so classes where professors have simply decided to allow a good number more people than in the past, and as long as they were seats in the room, they were able to do so. This is one explanation. Perhaps there are others. Despite all our efforts, Just using OLS and a sole predictor, the maximum enrollment, to predict Enrollment size does nearly just as well as Random Forests with the full set of core predictors including Max Enrollment. On the inference side of things, multicollinearity made life difficult. While we weren’t able to quantify exactly how much of a difference an individual predictor made on Enrollments, through our OLS analysis and random forests partial plots, we found that Section and Subject matter a lot. And that Instructor Quality, Course Quality, and Course Difficulty are usually quite significant. Whether the class was part of the Wharton core was significant in OLS analysis but tied with many other predictors on the variable importance plots. Ratings *do* matter. When we correct for class size and include other relevant factors, we can explain around 80% of the variation of untransformed Enrollments. It’s just difficult to quantify to what extent they matter on their own, our results suggest it’s not as much as one might think.

Reflections

This was my first real data science project, and I over the course of the project I became keenly aware of that fact. Only now do I realize how stylized and straightforward the examples and datasets supplied by my instructors in Statistics 101, 102, and 474 were. The goal behind these classes was largely to instill an understanding of concepts, and the intricacies of cleaning with data were details that didn’t need to be bothered with for the purposes of the class. My prior perception was that data collection and cleaning would be 20% of my efforts, the analysis 80%. But, as I learned the hard way, 90 percent of my efforts and pain went towards collecting, cleaning and debugging code at all levels of the project. It’s something I will remember for a long time—the devil really is in the details, and any good data scientist is also an expert at scripting languages.

I have always liked the study of statistics because it gives us precise ways to quantify to what degree we really "know" and don't "know" about any given topic or data set. In a way, statistics is really humanity's best shot at the truth. And, as is often the case with life, the truth is generally pretty messy. After all this effort, I wasn’t able to find much signal. A very large part of this certainly stems from my lack of understanding about how to deal with the web of issues that one runs into with real data.

Furthermore, I truly learned that true response function of success is very, very nonlinear. I would spend hours and hours and make no progress, be followed by 20 minutes of leaps and bounds.

A mentor once told me that the best way to learn data science is, well, to do lots of data science. One can’t understand real world issues and their fixes if one doesn’t gain experience dealing with them. In many ways I learned how *not* to do a data science project, and gained quite a few skills and experiences

along the way. I will take these lessons and apply them to future endeavors with the Wharton Quant Finance team, an analytics consulting project in the healthcare space, and research with Professor Richard Berk. In that spirit, this culmination of this project is not really an end, but rather a beginning.