Understanding Rolling Admissions

Yao Luo

University of Toronto

Yu Wang

Ryerson University

July 27, 2018

The objective is to get out a quick paper documenting all salient stylized facts. The target outlets are Journal of Public Economics, Quantitative Economics, AEJ-Policy and Economic Inquiry. In this draft, I first document what has been done (what data has been collected, what codes have been written), what needs to do in the next step, and the corresponding budget. I also provide a documentation of codes.

1 Data Sources

The main data source comes from Law School Numbers, the Official Guide to Law Schools and U.S. News and World Report Rankings. The data can be grouped into three parts: individual data, school data and data to check sample representativeness.

1.1 Individual Data

Individual data all come from Law School Numbers. I have scraped the data from 2003 to 2016. I cleaned the data from 2003 to 2016. The statistical analysis will be based on the sample from 2007 to 2011 (i.e. five application cycles) because the size of users are high among those years and are relatively stable. The caveat is that we have a financial crisis in the middle. I have the following variables ready to use:

1. Application Data
   (a) Portfolio of applications, Dates of applications
2. Admission Data
   (a) Portfolio of offers/rejections/waiting lists, Dates of results, Scholarship
3. Enrollment Data
   (a) The enrolled school
4. Personal Characteristics
   (a) LSAT and Undergraduate GPA
   (b) URM status(not yet but easy), Race, Gender (not yet but easy), State, College type and reputation, College Major, Years since college graduation (not yet but easy), Extracurricular activities (Lara is on it)
5. Note that more data can be mined out but I am too lazy to do that. For instance, some people report their majors in "extracurricular activities". Some people report their application and admission data in "additional information".

1.2 Law School Data

School data comes from three places: Law School Numbers, the Official Guide to Law Schools, the U.S. News and World Report Rankings.

1. Admission Deadlines (not yet)
   (a) Source: approximate the deadlines using recent deadline data on Law School Numbers
   (b) Need to be scrapped from Law School Numbers websites.
   (c) Also some data at "/Users/yuwang/Documents/research/research/lawschool/data/raw/applicati set"
2. Law School Rankings (partially)
   (a) Currently I use annual ranking from U.S. News and World Reports. The data are collected from various online resources but are largely incomplete, for instance, sometimes I only have access to the ranks of top 50 schools.
   (b) A better way is to use the actual ranking data. For now, I only have 2010 ranking data in electronic pdf. It needs to be scraped.
   (c) The rest of the actual ranking data needs to be purchased from Amazon.com or photocopied from Qing Gong. Canadian libraries, Penn library, Columbia library, NYU library don't have them. UNC library has 1994, 1995, 1996, 1997, 2000, 2002, 2003, 2005, 2006, 2007, 2008, 2009, 2011, 2012. Columbia library has 2004.
3. Law School Characteristics (partially, Melinda is on it)
   (a) Source: Official Guide to Law Schools

(b) Median GPA/LSAT, faculty-student ratio, bar passage rates, mainly variables used in school rankings.

(c) I ask Melinda to type the data into spreadsheets. She has finished 2004, 2007, 2008, 2010, 2011 (partially). The rest of the hard copy books of Official Guide to Law Schools are stored at her place. She said there 2009 book is not in the pile of books.

(d) The finished data need to be merged to the main data frame.

4. Law School Tuition (roughly)

(a) Probably can be scraped from websites (Law School Tuition Bubble); or copied from Official Guide to Law Schools, or copied from U.S. News and World Report Rankings. Need to be merged back to the main data frame.

`https://lawschooltuitionbubble.wordpress.com/original-research-updated/the-lstb-data/#`

(b) Raw data (probably obtained from Law School Tuition Bubble):

`/Users/yuwang/Documents/research/research/abandon/tuition/data/rawdata/law_school_tuit`
`/Users/yuwang/Documents/research/research/abandon/tuition/data/rawdata/law_school_tuit`

(c) Processed data by Stata:

`/Users/yuwang/Documents/research/research/abandon/tuition/data/statadata/bubble.`
`/Users/yuwang/Documents/research/research/abandon/tuition/data/statadata/usnews1`

1.3 Data to Check Sample Representativeness

1. Official Data from Law Schools on applicants' GPA/LSAT distributions

(a) Source: Official Guide to Law Schools

(b) Cleaned by Yao Luo - from 2006 to 2012. Saved at "/Users/yuwang/Dropbox/lsac archived/Law.xlsx".

2. Official Data on timing of sending applications

   (a) Source: LSAC websites. Saved at "/Users/yuwang/Documents/research/research/lawschool/dat timing of applications". Original weblink:

   `https://www.lsac.org/docs/default-source/publications-(lsac-resources)/dec_2012_lsr.p`

   (b) Cleaned by Qpig. Qpig recovered the underlying numbers generating the PDF graph. The numbers are saved at the same folder.

2 Empirical Analysis

2.1 Data Description and Summary Statistics

1. Basic: Sample 1 = With GPA + LSAT + application + admission data
2. Basic: Sample 2 = Sample 1 with enrollment data
3. Basic: Sample 3 = Sample 2 with race and gender and URM data (as these affect URM status in admission process)
4. Basic: Sample 4 = Sample 3 with college type data
5. Basic: Sample 5 = Sample 4 with college major + extracurricular data + state data
6. Calculate sample size and summary statistics for all these samples by individuals (applicant side)
7. Calculate sample size and summary statistics for application/admission/enrollment data by schools (school side)

2.2 Admission rules change over the application season. [Admission Rules]

1. Challenge: How to control for unobservables of applicants? Early applicants may differ in unobservables from late applicants, and these unobservables may be valued by schools (i.e. non-cognitive ability, enthusiasm shown to schools by applying early, ability to meet deadlines/wrap up materials and deliver early)
2. Basic: Document correlations between admission outcomes and dates of application.
3. Basic: Compare admission outcomes across discrete rounds of applications.
4. Robustness: Include lots of applicant characteristics.
5. Robustness: Compare applicant outcomes who just catch and miss the early rounds of application deadlines.
6. Robustness: Tease out the unobservables by exploiting the correlations of different school outcomes for each applicant.
7. Heterogeneity: Bundle schools by ranks and compare sizes of changes in admission rules over the season.

2.3 Do application decisions depend on timing of the application season? [Application Timing]

1. Basic: Do applicants apply altogether or seperately?
2. Basic: Are applicants more likely to attend the schools they apply earlier?
3. Basic: Do they apply higher ranked schools earlier?
4. Heterogeneity: Repeat the Basics conditional on applicant characteristics.

2.4 Schools send out offers gradually instead of waiting till deadlines. [School Offers Timing]

1. Basic: Graph the percentage of offers sent out by timing of the season.
2. Basic: Regress number of offers sent out by days passed over the season.
3. Heterogeneity: Repeat the Basics conditional on school characteristics.

2.5 Does timing of offers correlate with enrollment decisions? [Enrollment Decisions]

1. Basic: Are applicants more likely to attend the schools they receive offers earlier?

2.6 Are our data representative? [Representativeness]

1. Basic: Report density distribution by application season.
2. Ask Yao Luo how to compare two multi-dimensional distributions.

3 To-do List

1. Data: Process Years since graduation and Gender and URM variable (Easy Coding. 2 hour coding.)
2. Analysis: Work on all "basic" parts
3. Analysis: Work on all "advanced" analysis
4. Data: Merge Melinda data (Google Drive: ryerson email - my drive - law school ranking. 5 hours coding.)

5. Data: Scrape U.S. News 2010 ranking data and merge (Source: /Users/yuwang/Documents/research/r news ranking/041510usnewslawrankings.pdf) (Scrapping: 5 hours coding. Merging: 5 hours coding.)

6. Data: Scrape admission deadline data and merge (Qpig or myself, Workload: 15 hours coding.)

7. Data: Collect tuition data and merge (Source is discussed above. Merging: 5 hours coding.)

8. Data: Merge Lara extracurricular data (Budget: 300 hours/$6000. Merging: 2 hours coding.)

9. Data: Use annual U.S. News ranking data and merge (Wait for Qing Gong. Data Purchase $500. Data Input $1500. Merging: 15 hours coding.)

10. Analysis: Repeat all "basic" analysis

11. Analysis: Repeat all advanced analysis

Can launch out a quick paper. Forget about everything else. With this draft, I can safely accept seminar invitations and make conference submissions. Let me finish the rest of annoying small things and concentrate on this project until child birth! I still have perhaps 2 weeks ahead.

4 Documentation of Codes

There are four folders under "src": "extract", "transform", "load", and "simulation". There are some python files too - these files are to configure the folder and make codes across folders share-able.

4.1 "extract"

"extract" is the web scrapper. It was written by Qpig so the quality is quite reliable. There are two pieces of main codes. The first piece of main code is "crawler.py". I scrapped all the years back - from 2003 to 2016. The scrapped html files are saved at "/Users/yuwang/Documents/research/research/lawschool/data/html". The second piece of main code is "html to json.py". This piece of code extracts some information from html files and converts the extracted information to json files. The json files are saved at "/Users/yuwang/Documents/research/research/lawschool/data/json".

4.2 "transform"

"transform" is the data cleaner. The main code is "generate table.py". "generate table.py" works as a binder of a long series of functions. It cleans the data, calculate summary statistics, and export the data set to Stata for regressions. Now I describe the involved functions block by block. To revise or read these functions, check the "from XX import YY" blocks in the header of "generate table.py".

The first block consists of"process app data", "process rank data", and "merge app rank". "process app data" selects all candidate information - application data, admission data, enrollment data, and detailed personal characteristics. Then it merges all these variables to one data frame, "df app.csv". "process rank data" builds a panel dataset of law school rankings across years. As the law school ranking data come from various website resources, they involve different ways of spelling school names. "process rank data" involves a master school name file produced by Qpig using python package"difflib.get close matches". "merge app rank" attaches law school rankings to the portfolio of schools applied and enrolled by applicants in Law School Numbers.

The next line of function "learn text" exports unique values of each variables to csv files. This function helps manual processing data in the next few steps, as I can actually read

what values are in the data.

The next block of codes cleans text data. It is important to include "fillna('').str.lower()" when loading the data. "fillna('')" replaces all NaN values to empty strings and thus converting the whole column to string values.

I first clean race data by calling "clean race ethnicity". This function bundles race/ethnicity into the official race/ethnicity categories published by the Law School Admission Council. This function relies on a json file called "text race.json" also saved at "transform" folder. The official racial/ethnicity definition is saved at "/Users/yuwang/Documents/research/research/lawschool/races.pdf". See article "URM (Under-Represented Minority) Application FAQ.pdf" in the same folder for a description of URM qualifications.

I then clean state/city data by calling "clean state city".

The next big trunck of codes deal with college names. Users report their college names/types in a highly arbitrary way. It takes a lot of effort to clean them out. As it is virtually impossible to recover the full name of schools for every user, I try to classify schools into quality categories. I break down the work into several steps.

In the first step, I prepare a list of schools from U.S. News and World Report 2018 Ranking Websites. The list of schools have rankings so their quality can be categorized. The lists are saved at "/Users/yuwang/Documents/research/research/lawschool/data/entry" with file names starting with "usnews". I also supplement the U.S. News list with a list of college acronyms (scrapped from Wiki) and a list of college flagship campus. The list is produced using "usnews acronym flagship merge". I then perform a preliminary match between self-reported college names and official list of college names using "prune college name type" (which incorporates another function "match college name type"). The matching results help me move on to the second step.

In the second step, I divide data into several samples. The first sample, "exact", consists of colleges with full, complete, accurate school names. The second sample, "tallal", consists of colleges with incomplete but human-recognizable school names. The third sample,

"topN", does not report school names, but only school ranks, for instance, "big10", "top 20". The fourth sample, "union", reports the union a school belongs to, for instance, "public ivy", "seven sisters". The fifth sample, "vague", only contains some vague descriptions, such as "medicore" or "decent".

In the third step, I impute the name of colleges for all users. The imputation takes two rounds. In the first round, I match school to the official list of schools based on geographical locations - State and City. In the second round, I collect the unmatched schools from the first round, and manually impute a school name. The second round imputation also relies on a supplementary file "data/entry/U.S. News college ranking trends 2014 - The Washington Post.pdf", which contains historical rankings. I tried to impute the school with names of schools maintaining relatively stable historical ranking. The web link is

`https://www.washingtonpost.com/apps/g/page/local/us-news-college-ranking-trends-2014/1292/?nore`

In the first round of imputation, I write several functions to export the samples, and match them to the official list of schools by location. For the "tallal" sample, the functions are "tallal college name type" and "match tallal college name type" respectively. For the "topN" sample, the functions are "topN college name type" and "match topN college name type location" respectively. For the "union" sample, the functions are "match union college name type location". It sounds wierd, but "union college name type" is completely redundant.

In the second round of imputation, I rely on the following functions to manually assign school names: "fill topN merge univ", "fill topN merge lac", "fill union merge univ", "fill vague merge univ". They all rely on some json files in the "transform" folder. I do not impute school names for the "vague" sample, I only roughly divide them into several subcategories depending on the *adj* and tones of users describing the schools.

In the fourth step, I merge back the college information to the main sample from Law School Numbers. The related functions are "school usnews merge", "school usnews details merge", "vague details merge", "tallal usnews details merge", "exact usnews merge", "exact

usnews details merge". "details" means I take into account of State and City when doing the merge. Function "college name conclude" appends all these samples and carries out some proof checks.

With these imputed school names, I can rely on U.S. News and World Report Rankings to assign a numerical rank to them, and thus bundle them into flexible categories. In the end, the key variables describing schools are [name, rank, group, groups]. The "groups" contain values from [positive,neutral,bad, bad school,foreign]. "group" describes the U.S. News group to calculate rankings, for instance, is the school ranked 5th place among national universities or among national LACs or among regional universities? "groups" only apply to the "vague" sample but not other samples. "bad school" value is an exception, it refers to the self-reported schools in the "tallal" sample that does not convey any useful information. They can be dumped in the analysis, or simply bundled together with "bad" valued school names.

The next trunk of codes deal with college majors. Users report their majors in a highly arbitrary way too; and quite a lot of them have double majors. As there are too many majors, I bundle majors into major categories. I break down the work into several steps. In the first step, I download a list of majors and broader major categories from LSAC websites. These reflect how LSAC view majors and bundle them into major categories. The function is "lsac majors". In the second step, I break down double degrees and extract a list of unique values of reported college majors. The function is "breakdown majors". In the third step, I divide data into two samples. The first sample consists of majors that are listed by LSAC websites. The second sample consists the rest of majors. I give the second sample to Tallal to manually complete the full names and classify into LSAC major categories. In the last step, I merge major and major categories classification back to the main sample from Law School Numbers. The functions are "merge majors lsac lsn" and "tallal majors lsac lsn" and "major clean binder". All related functions are written in "process text major.py".

The next trunk of codes deal with extra-curricular activities. Users report their "soft" experiences often in some monologues. I have given the data to Lara for classification. The details on classification can be found at "/Users/yuwang/Documents/research/research/lawschool/data/entry/extra curricular". The codes to generate the files in this Lara folder are written in "process text extracurricular.py" and mainly consist of function "extract extracurriculars".

I also perform a topic modelling method to select the most important topics in "additional information". It turns out that many users report their application and admission processes in "additional information", rather than their unique life experiences. So I will skip cleaning this part of data. The function is based on codes shared by Yujia Wang, "topic modelling yujia", written in "process text extracurricular.py".

I still need perhaps a few lines to process data on "Gender" and "Years since Graduation" and "URM status". The final consolidated data frames at the moment are "data/edit/df details race college major cleaned.csv"'.

I then proceed to clean the data format (i.e. dates, etc). The functions are "clean app rank", "clean app date", "clean sample", and "stat sample". The purpose is to prepare the file for Stata to do regressions. Note need to make sure I feed in the right data frames to these functions, may not be "app matched", "app new", or "app date".

Lastly, I proceed to calculate summary statistics. This part may need more work. Also it may be moved to the "load" folder.

Note that "generate table backup.py" is redundant and completely overwritten by "generate table.py" and can be ignored. The rest of the python codes all seem useful. Note that "select tables" and "select tables details" are two different codes and they have different functions.

4.3 "load"

Currently it is empty.

4.4 "simulation"

Currently it is empty.

4.5 "Stata Codes"

Currently I have a "main.do" at the folder "transform". It carries out the regressions for previous drafts. I plan to move it to the folder of "load" and add more stuff to it.