Schedule Summer Math Courses

University of Washington, Math Department
Yanmeng(Anny) Kong, Linni Cai

UNIVERSITY of WASHINGTON
DEPARTMENT OF MATHEMATICS

Problem Description

- Comunity partner:

  Sarah Garner, director of the Math Department's Advising office

- Issue:

  The number of sections offered for math courses is not appropriate sometimes

- Cause:

  Students' tuition during the summer quarter goes mostly toward the salary of the summer quarter instructors

Problem Description - Example

Math 120

Year	Percent. Enrollment sec 1	Percent. Enrollment sec 2	Percent. Enrollment sec 3
2008	57.14(1050✓)	71.43(940✓)	77.14(1200✓)
2009	72.00(1050✓)	88.00(940✗)	104.00(1200✗)
2010	60.00(1200✓)	74.29(1050✓)	
2011	85.71(1050✗)	97.14(1200✗)	
2012	42.86(1050✗)	74.29(1200✓)	80.00(940✓)
2013	62.86(1050✓)	63.33(1200✓)	
2014	28.57(1050✗)	74.29(1200✓)	
2015	91.43(1200✗)		
2016	85.71(1200✗)		
2017	88.57(1200✗)		
2018	100.00(1200✗)		

- 100% enrollment in 2018 is not appropriate

Goal

1.1 Goal:

For this project, our goal is to *find appropriate schedules of sections for the following 9 non-major math courses,*

$$\textit{MATH 111/120/124/125/126/307/308/309/324,}$$

such that each section has $57\%^1 - 80\%^2$ *student enrollment for at least* $95\%^3$ *of the time based on historical data.*

As requested, we would like to find appropriate schedules for the 2019 math summer courses, including the number of sections and time slots of sections.

Impacts

Afterwards, we will be able to answer the following questions for Mrs. Garner:

1. What's the recommended # sections and time slots for 9 non-major math courses?
2. If we want to open/close a new section, which section would we recommend to open/close first?
3. If we want to open/close a new course, which course would we recommend to open/close first?

Mathematical Model

3 Mathematical Model

With the 2008 – 2018 summer course end-of-quarter schedules provided by Mrs. Garner, we analyzed the raw data with some manipulations in Python and some visualizations in R.

To solve three questions mentioned in section 1.2, we have two stages:

1. Use linear programming to get the appropriate number of sections for each course at year 2019;

2. Use the Poisson simulation to find the specific time slot we want to select for each course at year 2019.

Stage I - Linear Programming

8.1.3 Objective function:

- To maximize the total number of sections:

$$max \sum_{i} S_{i,2019} \tag{1}$$

where

$S_{i,2019}$	Predicted total # sections for course i in year 2019.

2. **Compute linear programming I** By applying constraints, we calculated the appropriate number of sections for each course each year.

```
# course -> year -> appro. # sections
initialize dictionary s

for each course
    for each year
        let c2Up,c2Lo,c1Up,c1Lo be 0
        for each enroll percentage enroll
            if enroll >= 80 and enroll < 100
                if only one enroll percentage
                    c1Up += 1
                else c2Up += 1
            else if enroll >= 29 and enroll < 57
                c2Lo += 1
            else if enroll >= 100
                c1Up += 1
            else if enroll < 29
                c1Lo += 1
        let appNSec be appropriate \# sections of the year and the course
        let nSec be original \# sections of the year and the course
        set appNSec = nSec + c2Up / 2.0 - c2Lo / 2.0
                    + c1Up - c1Lo and round down
        if appNSec - nSec > 1:
            appNSec = nSec + 1
        else if nSec - appNSec > 1:
            appNSec = nSec - 1
        if appNSec < 1:
            appNSec = 1
```

Find appropriate # sections for **previous years**

- Discussed in detail in constraint section in Appendix 8.1.4

3. **Compute linear programming II**

By adding weights and round down for the final result, the following algorithm presents the final # sections for each course in 2019.

```
let weight = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6])
let w = [x / sum(weight) for each x in weight]

# s = is course -> year -> appro. # sections
let S be a dictionary and make it a mapping such that
course -> appro.# sections with weights in 2019

for each course i in s
    let appSec, j be 0
    for each year y in s[i]
        set appSec += w[j] * years[y]
        j++
    set s[i] = appSec with two decimals
```

Find # sections for each course in **2019** using weighted average of previous years

Results - Stage

4.1 Stage I

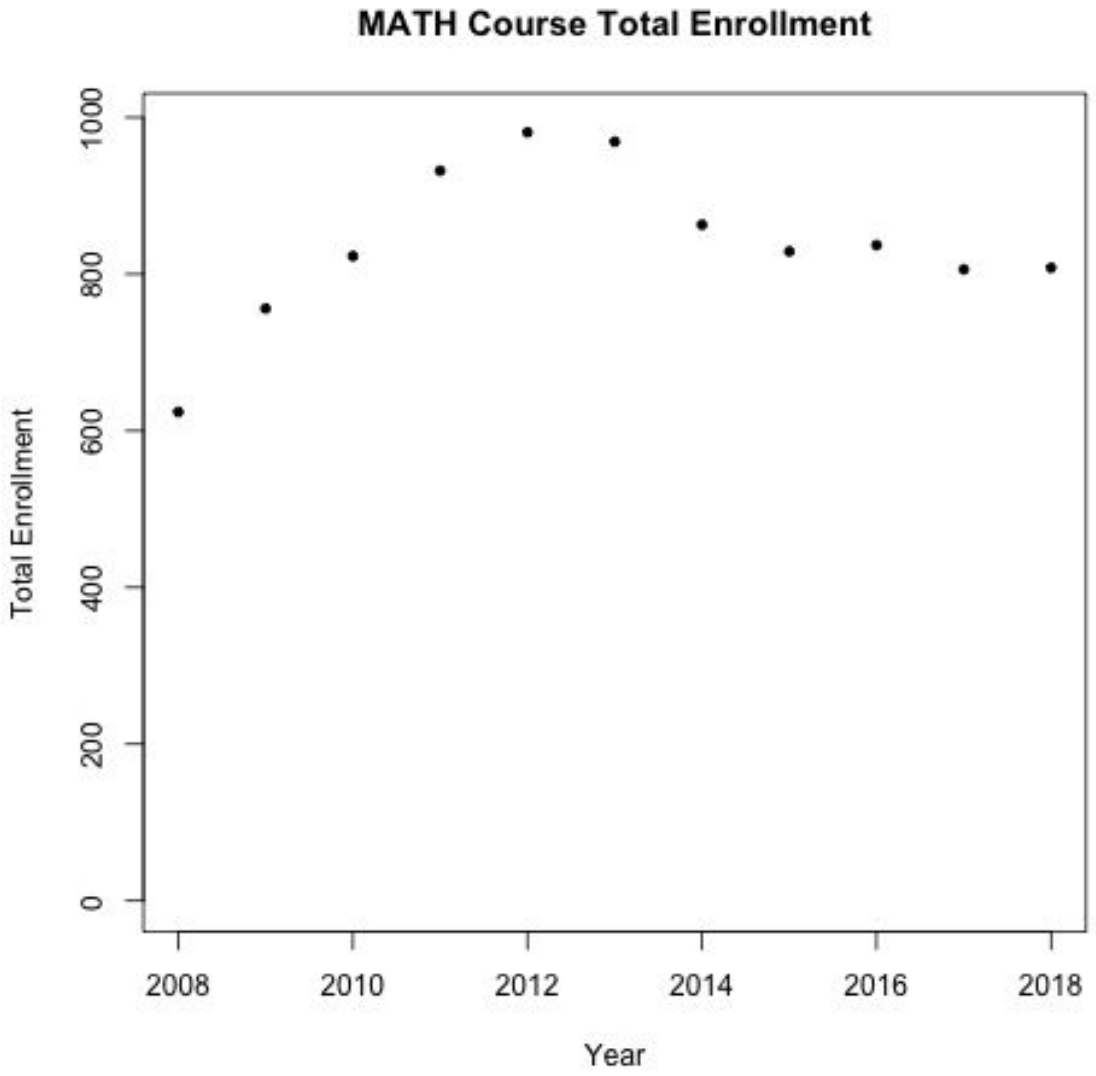
By running the Python algorithm, we have the resulting recommended # sections as below.

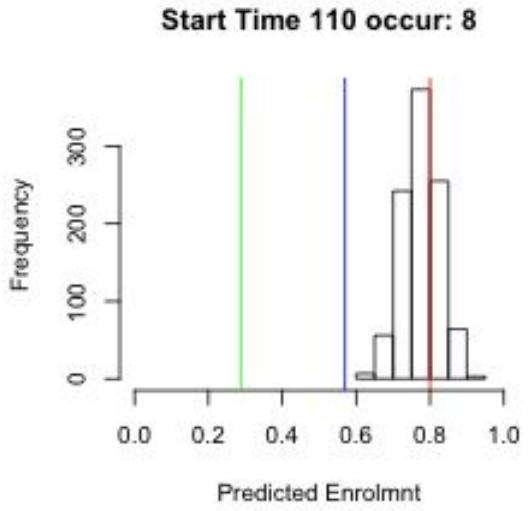
Course	# sections
111	1
120	2
124	3
125	4
126	6
307	6
308	6
309	3
324	4

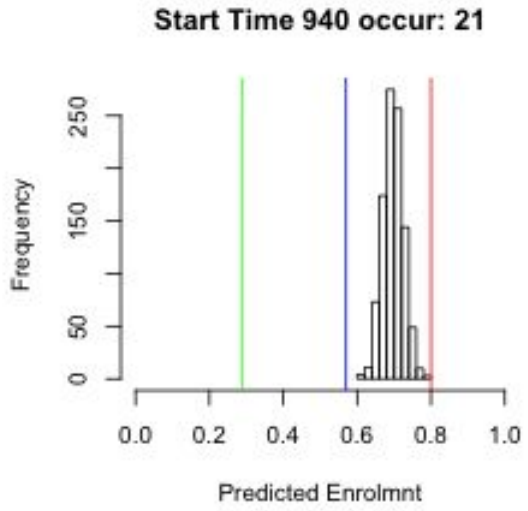
Table 1: The number of sections for each course in 2019

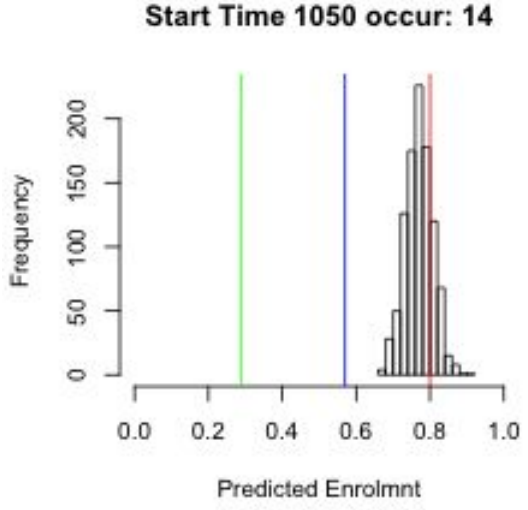
In our interpretation, for course Math 111, we recommend scheduling 1 section for 2019 summer; for course Math 120, we recommend scheduling 2 sections for 2019 summer; for course Math 124, etc.

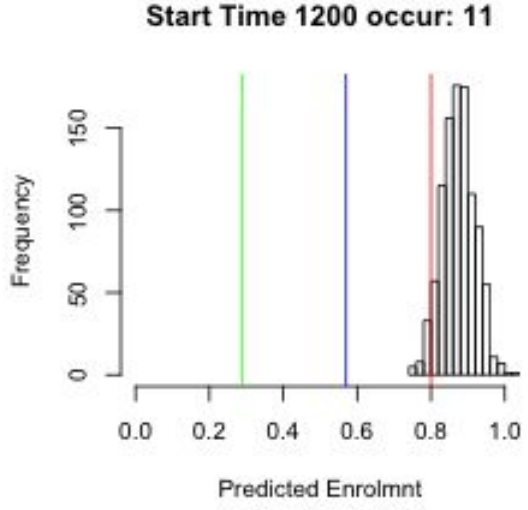
Poisson Distribution











Poisson Distribution

3.2 Stage II

we calculate weighted average as following:

- With fixed i and k

$$\frac{\sum_j pe_{i,j,k} * S_{i,j} * w'_{j-2007}}{\sum_j S_{i,j} * w'_{j-2007}} \tag{12}$$

For example:

- We have calculated the weighted average for MATH 111 in time slot 10:50 am.

$$\frac{pe_{i,k,j=2008} * S_{i,j=2008} * w'_1 + pe_{i,k,j=2009} * S_{i,j=2009} * w'_2 + \ldots + pe_{i,k,j=2018} * S_{i,j=2018} * w'_{11}}{S_{i,j=2008} * w'_1 + S_{i,j=2009} * w'_2 + \ldots + S_{i,j=2018} * w'_{11}} \tag{13}$$

Monte Carlo Simulation

```r
X = rnorm(100)
X_med = median(X)
X_med
```

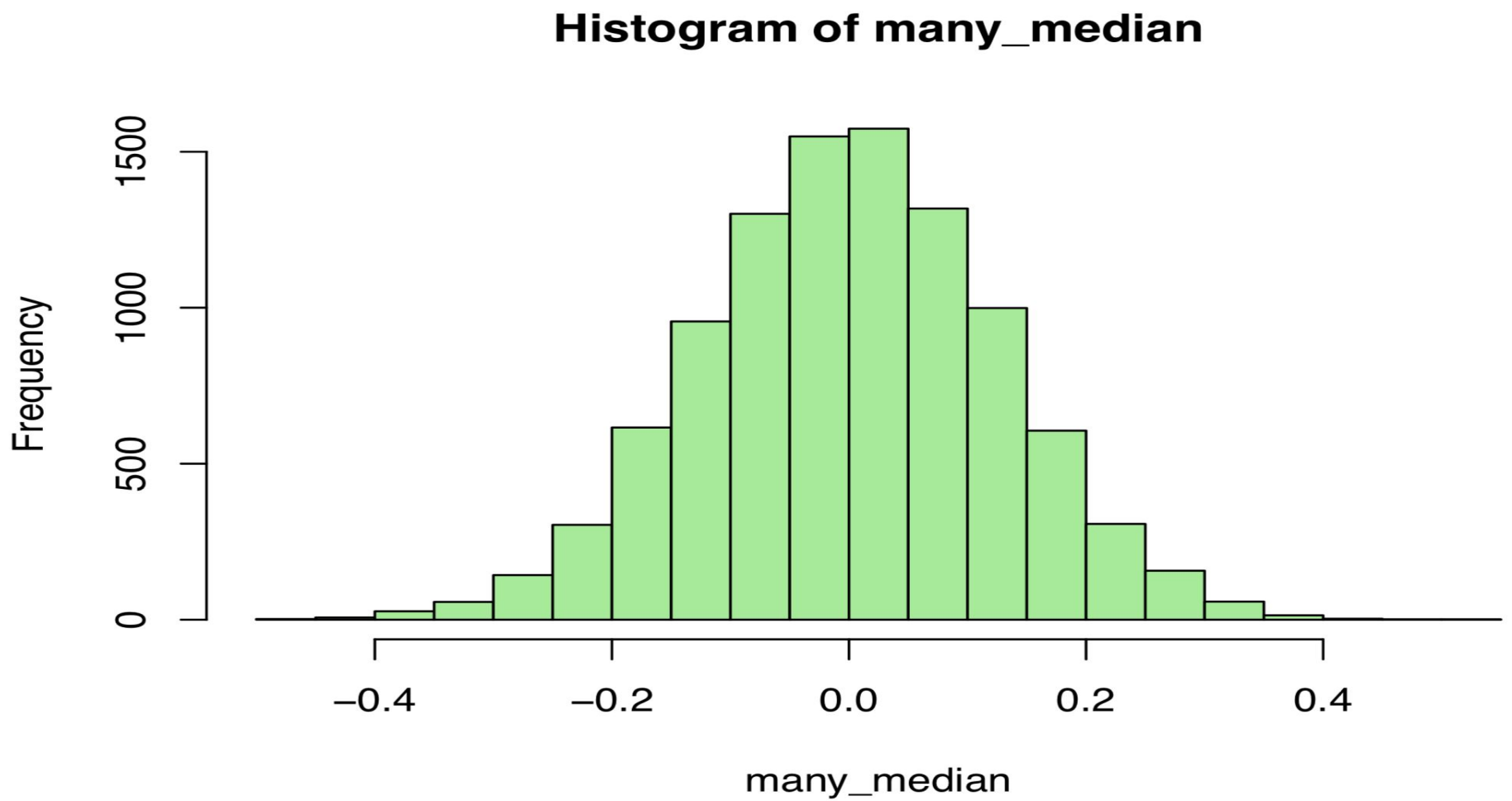
```
## [1] 0.01708379
```

```r
N = 10000
  # number of repitions
many_median = rep(NA, N)
for(i in 1:N){
  X = rnorm(100)
  X_med = median(X)
  many_median[i] = X_med
    # save the median in each repitition
}
head(many_median)
```

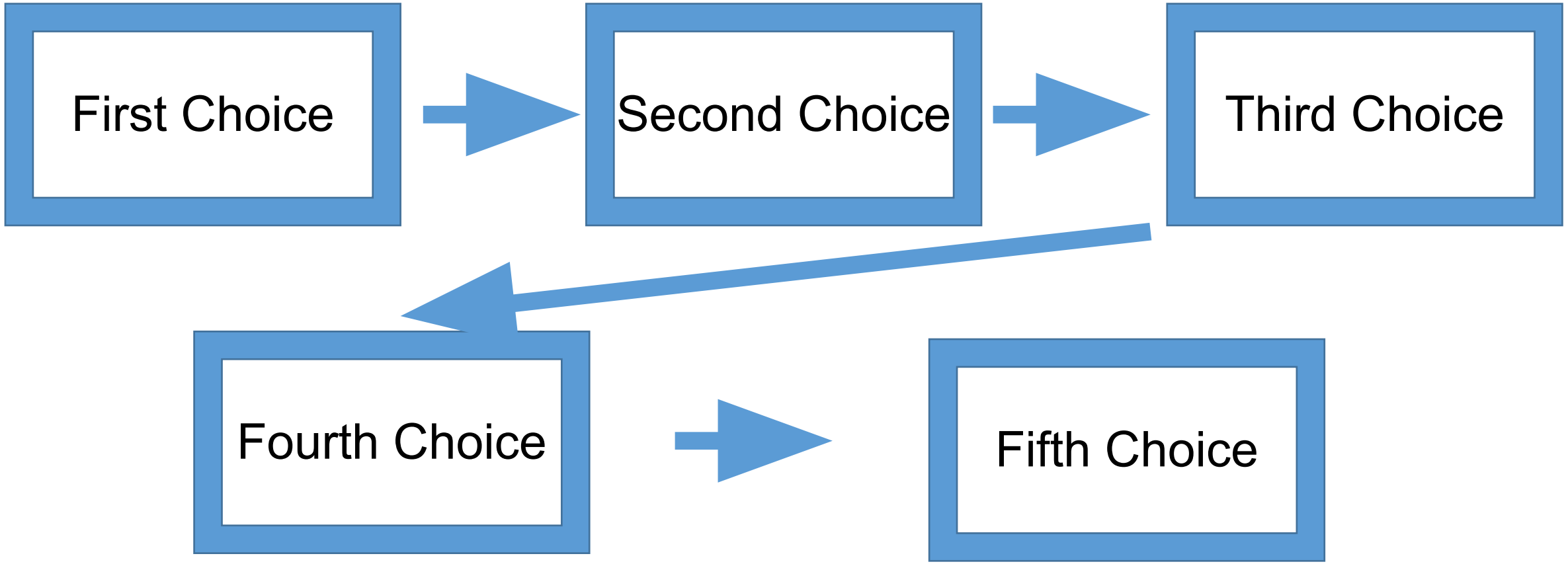
```
## [1]  0.03550759  0.13135906 -0.14435323 -0.12738564  0.03880981 -0.01144368
```

Monte Carlo Simulation

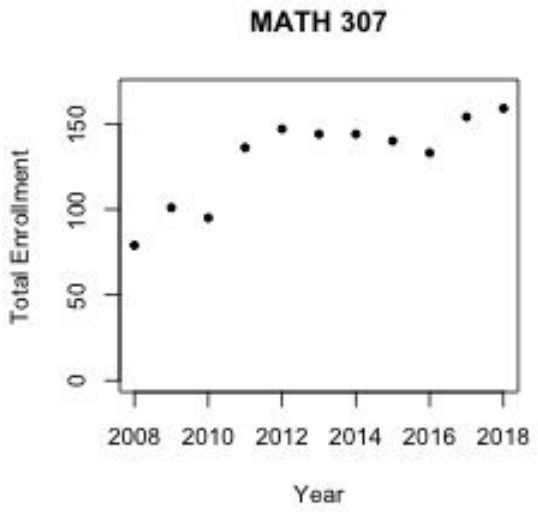
```
hist(many_median, col="lightgreen")
```

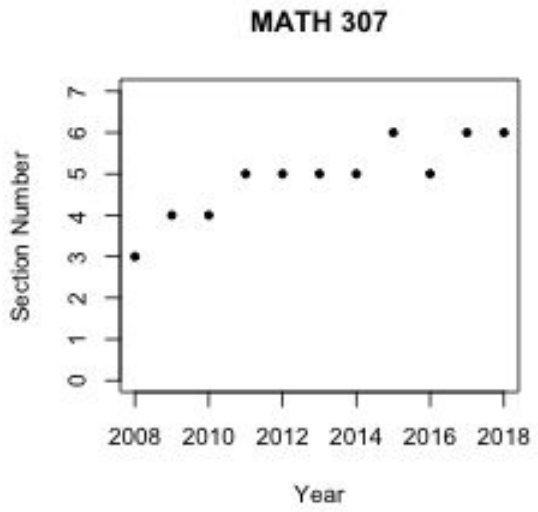


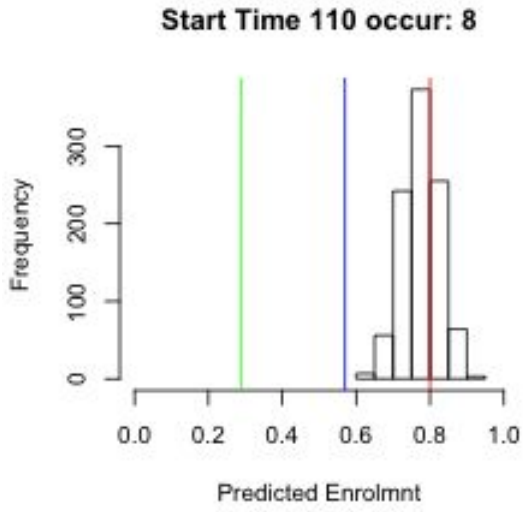
Priority Queue

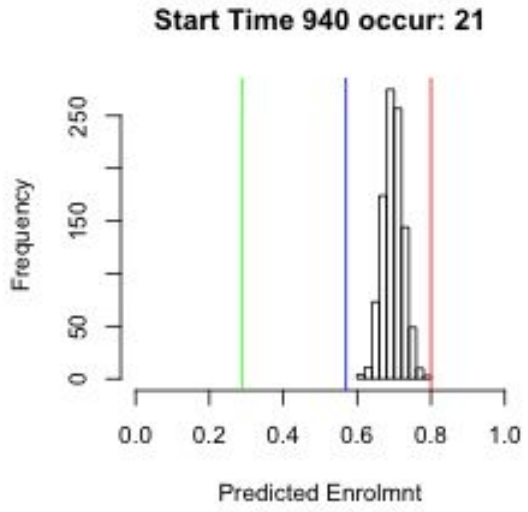


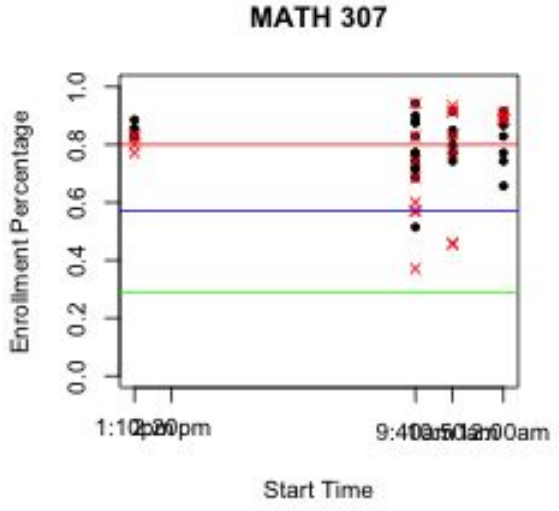
Priority Queue

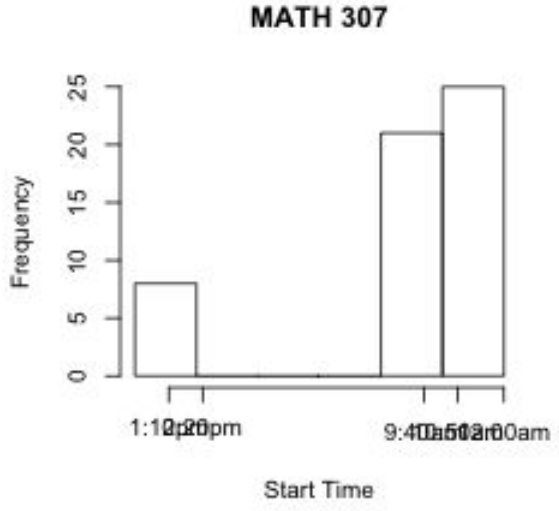


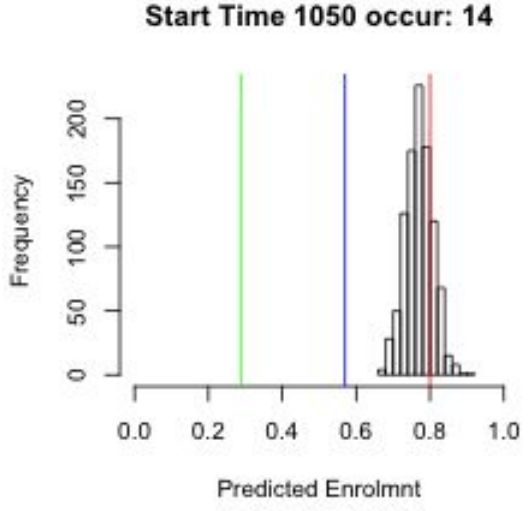


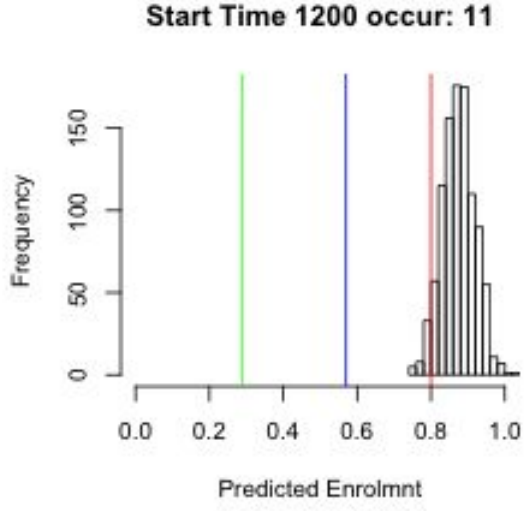










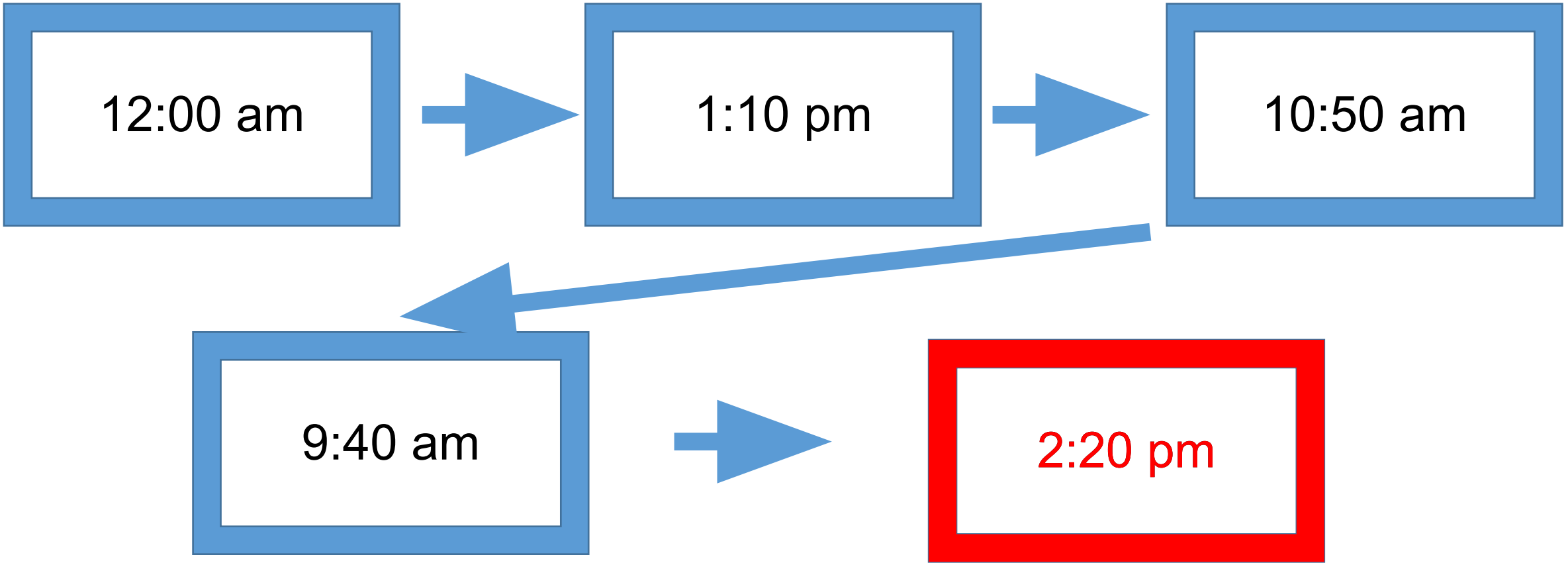


Priority Queue

```
[1] 307
[1] "performance:"
  Section     > 0.8 LB    > 0.8 UB 0.57~0.8 LB
5    1200 9.021621e-01 1.024763617  0.02689106
1     110 2.849505e-01 0.355998235  0.63080647
4    1050 1.797598e-01 0.237201916  0.73876009
3     940 2.531781e-05 0.005571643  0.93800402
  0.57~0.8 UB > 0.57 LB > 0.57 UB
5  0.05215797  0.938973  1.063952
1  0.73412518  0.938973  1.063952
4  0.85016884  0.938973  1.063952
3  1.06292115  0.938973  1.063952

  Section Nmuber
1     110      1
2     220      1
3     940      1
4    1050      1
5    1200      2
```

Priority Queue



Priority Queue

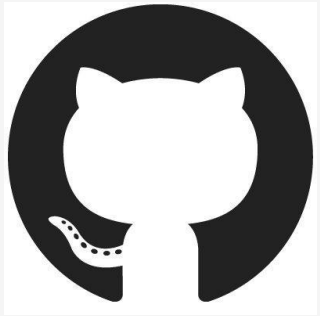
Time Slot	Number of Sections
9:40 am	1
10:50 am	1
12:00 am	2
1:10 pm	1
2:20 pm	1

Table 9: Section Schedule for Math 307 in 2019 (6 sections)

For MATH 307, this represent that we suggest to open six sections, one at 8:30 am, one at 9:40 am, one at 10:50 am, two sections at 12:00 am, one at 1:10 pm, one at 2:20 pm.

Thank you

Get more details of our project in Github



https://github.com/MathSummerProject/MathSummerScheduling