College Pathway Analytics

INFO 5200 Learning Analytics: Week 12 Homework

[[Cole Walsh, 4399966]]

In this homework, you will learn how to analyze enrollment record data to identify patterns that can inform policy decisions about an academic curriculum or what information to provide to students as they plan their courses. You are given a synthetic dataset with an authentic correlation structure for students who have graduated in one of three majors (major 1, 2, and 3).

Learning Objectives

1. Understand the structure of course enrollment data
2. Identify toxic course pairings
3. Identify course-major relationships to give students feedback about path-dependencies

Scenario

You are approached by a vice provost for undergraduate studies to inform upcoming policies about curriculum changes. You are asked to provide guidance on two high-level questions:

(1) Which courses should we advise students not to take in the same semester?

(2) What can we tell students about how their first-year course choices influence their likely major?

Data

The synthetic dataset contains one record per student course enrollment.

Variable	Data Type	Definition
student_id	numeric	Unique student identifier
major_id	numeric	Unique major identifier
course_id	numeric	Unique course identifier
term	numeric	Semester number in temporal order; e.g. 1=Fall 2017, 2=Sping 2018, 3=Fall 2018, etc.

Exploring the Data

Before starting to answer any questions, take some time to understand the structure of the dataset. The block below will not be evaluated in the knitted report (eval=F). You can use this space to try out different approaches to explore the data and test your understanding of it.

```
head(a)
table(a$major_id, a$term)
length(unique(a$course_id))
hist(a$grade)
```

Part 1. Which courses should we advise students not to take in the same semester?

The goal is to identify course pairings that should be avoided because students have earned lower grades when taking them together compared to taking them some time apart.

Question 1: Which pairs of courses show lower grades when students take them together than when they take them apart? (Tip: follow the instructions below; or try it your own way; the combn() function will be useful regardless; it is not a bad idea to use for-loops to solve this.)

```
#######################################
####### BEGIN INPUT: Question 1 #######
#######################################

# First, narrow the set of observations to courses that are frequently chosen (say at least 20 times) a

a[, 1:4] <- a[, 1:4] %>%
  lapply(., as.factor) %>%
  data.frame(.)

# Drop duplicate courses
a.noDuplicates <- a[!duplicated(a[, c('student_id', 'term', 'course_id')]),]

# To data.table for future computations
a.dt <- as.data.table(a.noDuplicates)

a.dt.frequent <- a.dt[, Course.Frequency := .N,
                      by = course_id][Course.Frequency >= 20, N.Courses := .N,
                                      by = .(student_id,
                                             term)][N.Courses > 1,
                                                    !c('Course.Frequency','N.Courses')]


# Second, given this smaller dataset, identify all actual course pairings in the dataset (i.e. which pa

# Get all pairs and re-order with lowest course number first
a.pairs <- a.dt.frequent[, as.data.table(t(combn(course_id, 2))),
                         by = .(student_id,
                                term)][, `:=`(Course1 = as.character(pmin(as.numeric(V1),
                                                                          as.numeric(V2))),
                                               Course2 = as.character(pmax(as.numeric(V1),
                                                                           as.numeric(V2)))),
                                        ][, -c('V1', 'V2'),]

# Get grades of courses using joins
a.pairs.grade1 <- right_join(a.dt.frequent, a.pairs, by = c('student_id', 'term',
                                                            'course_id' = 'Course1')) %>%
  select(student_id, term, course_id, grade, Course2) %>%
  `colnames<-`(c('student_id', 'term', 'Course1', 'Grade1', 'Course2'))
```

```
## Warning: Column `course_id`/`Course1` joining factor and character vector,
## coercing into character vector
```

```r
a.pairs.grades <- right_join(a.dt.frequent, a.pairs, by = c('student_id', 'term',
                                                            'course_id' = 'Course2')) %>%
  select(student_id, term, course_id, grade, Course1) %>%
  `colnames<-`(c('student_id', 'term', 'Course2', 'Grade2', 'Course1')) %>%
  left_join(a.pairs.grade1, ., by = c('student_id', 'term', 'Course1', 'Course2'))
```

```
## Warning: Column `course_id`/`Course2` joining factor and character vector,
## coercing into character vector
```

```r
# Third, compute the average grade for the courses the student received when taking each pair of course

# Average grades of pairs
a.pairs.grades$Avg.Grade <- rowMeans(a.pairs.grades[, c('Grade1', 'Grade2')])

# Fourth, aggregate by course pairs and compute the average paired grade and frequency of occurance. Th

a.pairs.grades.dt = as.data.table(a.pairs.grades)

Course.Pairs.Grades <- a.pairs.grades.dt[, .(Avg.Paired.Grade = mean(Avg.Grade), N.Pairs = .N),
                                         by = .(Course1, Course2)][N.Pairs >= 20]


# Fifth, going back to the full dataset, find students who took the same common course pairs identified

for(pair in 1:nrow(Course.Pairs.Grades)){

  Course.1 <-  Course.Pairs.Grades[pair, Course1]
  Course.2 <- Course.Pairs.Grades[pair, Course2]

  Course1.dt <- a.dt[course_id == Course.1] %>%
    select(student_id, term, course_id, grade)

  Courses.dt <- a.dt[course_id == Course.2] %>%
    select(student_id, term, course_id, grade) %>%
    left_join(Course1.dt, ., by = ('student_id')) %>%
    filter(term.x != term.y)

  Courses.dt$Avg.grade <- rowMeans(Courses.dt[, c('grade.x', 'grade.y')])

  if(pair > 1){
    Unpaired.df <- Unpaired.df %>%
      add_row(Course1 = Course.1, Course2 = Course.2, Avg.Unpaired.Grade = mean(Courses.dt$Avg.grade))
  } else {
    Unpaired.df <- data.frame(Course1 = Course.1, Course2 = Course.2,
                              Avg.Unpaired.Grade = mean(Courses.dt$Avg.grade))
  }
}

# Sixth, compare the paired and unpaired average grade for each common course pair. Write down which FO

Course.Pairs.Grades %>%
  left_join(., Unpaired.df, by = c('Course1', 'Course2')) %>%
  mutate(Diff = Avg.Paired.Grade - Avg.Unpaired.Grade) %>%
```

```
  arrange(Diff)
```

```
## Warning: Column `Course1` joining character vector and factor, coercing
## into character vector
```

```
## Warning: Column `Course2` joining character vector and factor, coercing
## into character vector
```

```
##    Course1 Course2 Avg.Paired.Grade N.Pairs Avg.Unpaired.Grade
## 1      946     947         2.416591      22           2.898716
## 2        8     934         2.385682      22           2.699079
## 3      185     934         2.761471      51           2.972063
## 4      186     949         3.205385      26           3.307529
## 5      185     949         3.102419      31           3.204458
## 6      185     186         3.317486     181           3.410653
## 7      186     951         2.920000      23           3.011333
## 8      192     934         2.993200      25           3.070748
## 9      192     952         3.125500      20           3.201579
## 10     193     934         2.928393      28           2.994888
## 11     186     934         2.907979      47           2.971538
## 12     193     946         3.060128      39           3.121552
## 13     186     585         2.765833      42           2.804852
## 14     186     980         3.383250      40           3.420524
## 15     192     946         3.166400      25           3.192842
## 16     193     980         3.402604      48           3.397552
## 17     185     585         2.783019      53           2.772276
## 18     193     585         2.818478      23           2.803080
## 19     186     946         3.121212      33           3.101416
## 20     946     949         3.006200      25           2.962632
## 21     193     952         3.149750      20           3.105732
## 22     186    1126         3.680208      24           3.617099
## 23     186     193         3.465345      29           3.399010
## 24     585     946         2.662286      35           2.572857
## 25     185     980         3.433167      30           3.343402
## 26     192     193         3.527198     182           3.434625
## 27     946     980         3.135156      32           3.040116
## 28       8     192         3.249773      22           3.149836
## 29     934     946         2.838056      36           2.731748
## 30       8     193         3.166111      27           3.049921
## 31     192     980         3.628571      35           3.492876
## 32     947     949         3.156667      33           2.981023
## 33       8     186         3.154259      27           2.978214
## 34     185     946         3.242857      35           3.020212
## 35     585     934         2.586161      56           2.337375
## 36     950     952         2.948452      42           2.589615
## 37     193     950         3.451905      21           3.019265
## 38     152     154         3.347340      47           2.750625
## 39     661     663         3.037222      63           2.273929
##            Diff
## 1  -0.482125307
## 2  -0.313397129
## 3  -0.210592904
## 4  -0.102144796
```

```
## 5  -0.102038476
## 6  -0.093167221
## 7  -0.091333333
## 8  -0.077548031
## 9  -0.076078947
## 10 -0.066495203
## 11 -0.063559738
## 12 -0.061423519
## 13 -0.039018519
## 14 -0.037274017
## 15 -0.026442466
## 16  0.005052083
## 17  0.010742445
## 18  0.015398551
## 19  0.019796459
## 20  0.043568421
## 21  0.044018293
## 22  0.063109568
## 23  0.066334589
## 24  0.089428571
## 25  0.089765027
## 26  0.092572802
## 27  0.095039971
## 28  0.099936662
## 29  0.106307588
## 30  0.116190476
## 31  0.135695084
## 32  0.175643939
## 33  0.176044974
## 34  0.222645022
## 35  0.248785714
## 36  0.358836996
## 37  0.432640056
## 38  0.596715426
## 39  0.763293651
```

```
# Write down the pairs here:

# 946, 947
# 8, 934
# 185, 934
# 186, 949

########################################
########################################
```

Part 2: How students' first-year course choices influence their likely major

Question 2: For the courses that students commonly take in their first term, how does the choice of which ones they enroll in influence their likelihood of majoring in a field?

```
########################################
####### BEGIN INPUT: Question 2 #######
########################################

# First, identify the most commonly taken courses in the student's first term for all students (note th

a.dt <- as.data.table(a.noDuplicates)
a.FirstTerm <- a.dt[, First.term := min(as.numeric(term)),
                    by = .(student_id)][term == First.term, Course.Frequency := .N,
                                        by = .(course_id)][Course.Frequency >= 20]

a.FirstTerm %>%
  group_by(course_id) %>%
  summarize(n())
```

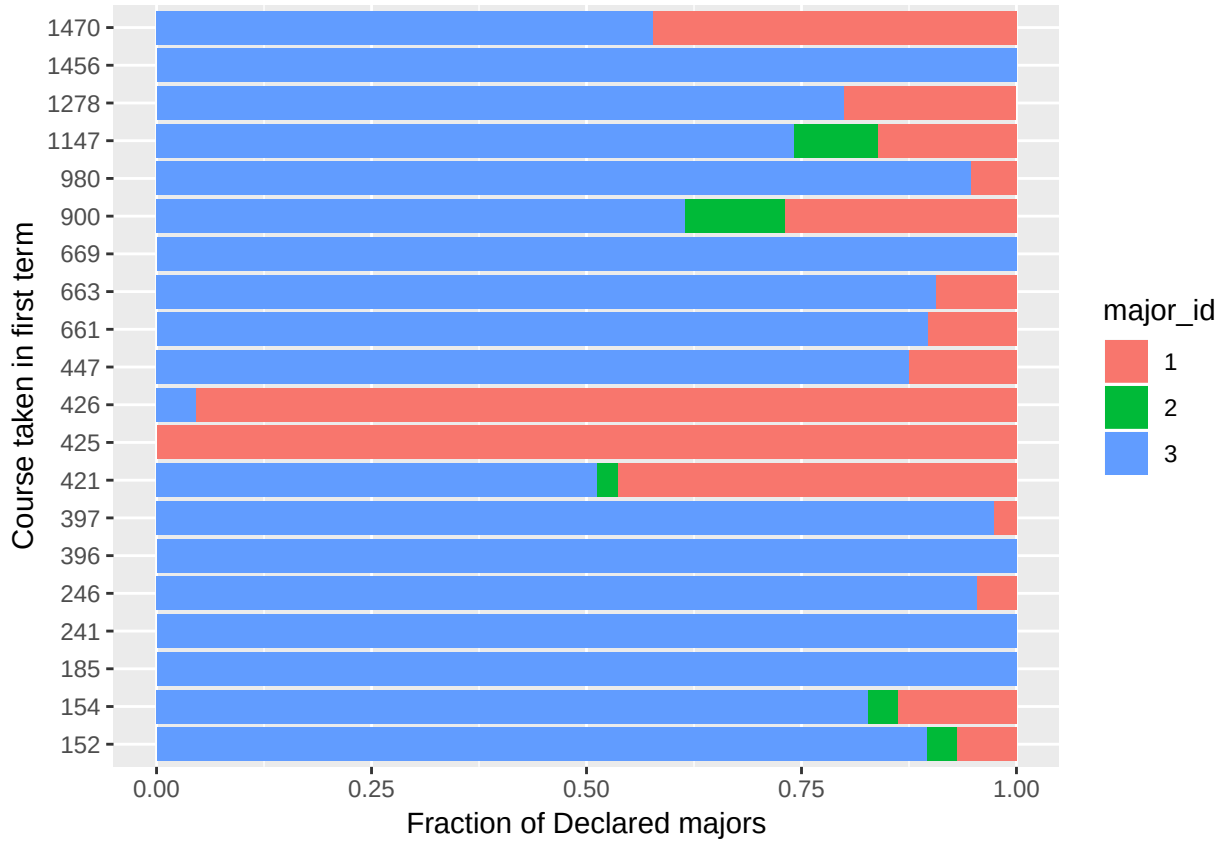
```
## # A tibble: 20 x 2
##    course_id `n()`
##    <fct>     <int>
##  1 152          29
##  2 154          29
##  3 185          27
##  4 241          32
##  5 246          22
##  6 396          23
##  7 397          39
##  8 421          41
##  9 425          22
## 10 426          22
## 11 447          24
## 12 661          29
## 13 663          32
## 14 669          24
## 15 900          26
## 16 980          57
## 17 1147         31
## 18 1278         30
## 19 1456         21
## 20 1470         26
```

```
# Second, compute the likelihood that a student majors in each of the three majors conditional on enrol

a.FirstTerm[, .(Frac_Major1 = mean(major_id == 1), Frac_Major2 = mean(major_id == 2),
                Frac_Major3 = mean(major_id == 3)), by = .(course_id)]
```

```
##     course_id Frac_Major1 Frac_Major2 Frac_Major3
##  1:       669  0.00000000  0.00000000  1.00000000
##  2:      1147  0.16129032  0.09677419  0.74193548
##  3:       241  0.00000000  0.00000000  1.00000000
##  4:       152  0.06896552  0.03448276  0.89655172
##  5:       900  0.26923077  0.11538462  0.61538462
##  6:       246  0.04545455  0.00000000  0.95454545
##  7:       397  0.02564103  0.00000000  0.97435897
##  8:      1278  0.20000000  0.00000000  0.80000000
```

```
##  9:       154  0.13793103  0.03448276  0.82758621
## 10:      1456  0.00000000  0.00000000  1.00000000
## 11:       426  0.95454545  0.00000000  0.04545455
## 12:       980  0.05263158  0.00000000  0.94736842
## 13:       185  0.00000000  0.00000000  1.00000000
## 14:       421  0.46341463  0.02439024  0.51219512
## 15:      1470  0.42307692  0.00000000  0.57692308
## 16:       425  1.00000000  0.00000000  0.00000000
## 17:       447  0.12500000  0.00000000  0.87500000
## 18:       661  0.10344828  0.00000000  0.89655172
## 19:       663  0.09375000  0.00000000  0.90625000
## 20:       396  0.00000000  0.00000000  1.00000000
```

```r
# Third, make a visualization that shows the likelihood of majoring in each major (1,2,3) after taking

ggplot(a.FirstTerm, aes(x = course_id, fill = major_id)) +
  geom_bar(position = 'fill') +
  coord_flip() +
  labs(x = 'Course taken in first term', y = 'Fraction of Declared majors')
```




```r
# Fourth, complete the blanks:
# - Students who take course 669 are most likely to major in 3.
# - Students who take course 425 are most likely to major in 1.
# - Students who take course 421 have about equl probability of majoring in 1 and 3.
```

```
####################################
####################################
```

Self-reflection

Briefly summarize your experience on this homework. What was easy, what was hard, what did you learn?

- I took this opportunity to try to learn more about using data.table. This was frustrating at times, but in the end I only needed to use one 'for' loop so I think this was a worthwhile experience.

Submit Homework

This is the end of the homework. Please **Knit a PDF report** that shows both the R code and R output and upload it on the EdX platform. Alternatively, you can Knit it as a "doc", open it in Word, and save that as a PDF.

Important: Be sure that all your code is visible. If the line is too long, it gets cut off. If that happens, organize your code on several lines.