examMarks Package

Andrew Davis, Andriani Hadjiconstanti, and Zamfirescu Ana Maria

26/4/2017

Package Overview

```
library(examMarks)
```

- This package is designed to mark student answers to based on given exam answer keys and to analysis the distributions of the marks based on different factors.
- It is also capable of generating random student answers and randomly generated answer keys.

The package is split into 3 parts with helpful datasets included as well.

1. Random Generation of Answers
2. Marking student Answers
3. Analysing the Distribution of the Marks

Datasets

- **students** - dataframe with the ID of students and their respective degree course

```
head(students)
```

```
##      ID              degree
## 1 39502 Biological Sciences
## 2 58085            Genetics
## 3 34055            Genetics
## 4 27854 Biological Sciences
## 5 46107 Biological Sciences
## 6 76693            Genetics
```

- **exams** - dataframe that defines which modules each degree course takes

```
exams
```

```
##   module Biological Sciences Genetics
## 1  BS281                 Yes      Yes
## 2  BS282                 Yes      Yes
## 3  BS283                 Yes Optional
## 4  BS284                  No      Yes
## 5  BS285                 Yes       No
```

Datasets

- **questions** - dataframe that lists how many questions a student is asked in each exam and how many total questions each exam has

```
head(questions)
```

```
##   module questions totalQuestions
## 1  BS281        30            100
## 2  BS282        20             50
## 3  BS283        30            150
## 4  BS284        20            200
## 5  BS285        50            100
```

- **keys** - list with a sample answer key for each exam

```
summary(keys)
```

```
##       Length Class      Mode
## BS281 1      data.frame list
## BS282 1      data.frame list
## BS283 1      data.frame list
## BS284 1      data.frame list
## BS285 1      data.frame list
```

Random Generation of Answers

- To generate an answer sheet for a single student use **generateStudentAnswersForExam()**
- students can answer either a, b, c, d, e, or not at all (N/A) for each question
- if writeToFile == TRUE, then studentID and moduleID must be supplied to create the filename

```
head(generateStudentAnswersForExam(totalNumberofQuestions = 100,
                                   numberOfQuestionsToAnswer= 30,
                                   writeToFile = F))
```

```
##   question answer
## 1        7      d
## 2        8   <NA>
## 3        9      d
## 4       12      d
## 5       13      c
## 6       19      c
```

Random Generation of Answers

- To generate all student answer for a given module use **generateAllStudentAnswersForExam()**
- The default values are set to use the provided datasets
  - these can be manipulated or new datasets or files can be used for individual use
  - use readFromFiles to determine whether or not the arguments should be read as Rdata or files
  - use degreeNames if the degree courses are not called 'Biological Sciences' and 'Genetics'
- if writeToFile == FALSE
  - data is output as a list with 2 elements, the students that took the exam and a list of the answers for each student
- if writeToFile == TRUE
  - a file listing the students who took the exam is created
  - a folder is created with the student answer files within it
- if a module is optional for a given degree, course then students are selected at random to take it, with more students taking it being more likely

Random Generation of Answers

- The student list as the first list element

```
test = generateAllStudentsAnswersForExam('BS281', writeToFile = FALSE)
head(test[[1]])
```

```
##      ID              degree
## 1 39502 Biological Sciences
## 2 58085            Genetics
## 3 34055            Genetics
## 4 27854 Biological Sciences
## 5 46107 Biological Sciences
## 6 76693            Genetics
```

- The next element is the list of dataframes
  - each student's answers is its own element of the list
- each generates the same output as generateStudentAnswersForExam() for each student's answers

```
test = generateAllStudentsAnswersForExam('BS281', writeToFile = FALSE)
test[[1]][2]
```

Random Generation of Answers

- Can also use **createAnswerKey()** to randomly generate an exam key

```
head(createAnswerKey(numberOfQuestions = 30, writeToFile = FALSE,
                     ansOptions = letters[1:5]))
```

```
## [1] "b" "d" "a" "d" "e" "e"
```

- if writing to a file moduleID is required in order to name the answer key

```
createAnswerKey(numberOfQuestions = 30, writeToFile = TRUE, moduleID = 'BS281',
                ansOptions = letters[1:5])
```

Marking Students

- To mark all the students for one exam **markStudentsForExam()**
- This function relies on all relevant files to be in the same directory
  - these include the correct answer files, "number_of_questions.tsv" as well as the folders containing the student answer files for each exam
- The output is in the form of a dataframe with studentID, degree course, and mark as columns

```
head(markStudentsForExam(fileDir = './', ExamFilesDir = './BS281studentAnswerFiles',
                   ModuleID = 'BS281'))
```

```
##   StudentID              Course Mark
## 1     10161 Biological Sciences   23
## 2     11724            Genetics    7
## 3     11826            Genetics   23
## 4     11832 Biological Sciences   13
## 5     13403 Biological Sciences    7
## 6     13652            Genetics    7
```

Marking Students

- as part of the function that will mark all of the exams degree marks need to be added as well
- the **getDegree()** function is used for that
- In this function a mark is entered and it is converted to a degree

```
head(addDegrees(marks))
```

```
##   StudentID              Course Mark Degree
## 1     10161 Biological Sciences   23 failed
## 2     11724            Genetics    7 failed
## 3     11826            Genetics   23 failed
## 4     11832 Biological Sciences   13 failed
## 5     13403 Biological Sciences    7 failed
## 6     13652            Genetics    7 failed
```

Marking Students

- getDegree() is then used in the **addDegrees()** function to add degrees to each student
- The function takes a dataframe with columns of studentID, degree course, and mark as input
- The same dataframe is outputted with an added column including degree

```
getDegree(65)
```

```
## [1] "2:1"
```

Marking Students

- To mark the student answers for all the exams use the **markStudents()** function
- This function takes the directory where everything is located as input
  - This function uses markStudentsForExam(), so all relevant files must be in the same directory
- There is also the option to have the output show on the console or be written into a folder using the writeToFile parameter
- If writeToFile = FALSE, a list of dataframes is created with each one representing the marked answers for a given exam

```
summary(markStudents(fileDir = './'))
```

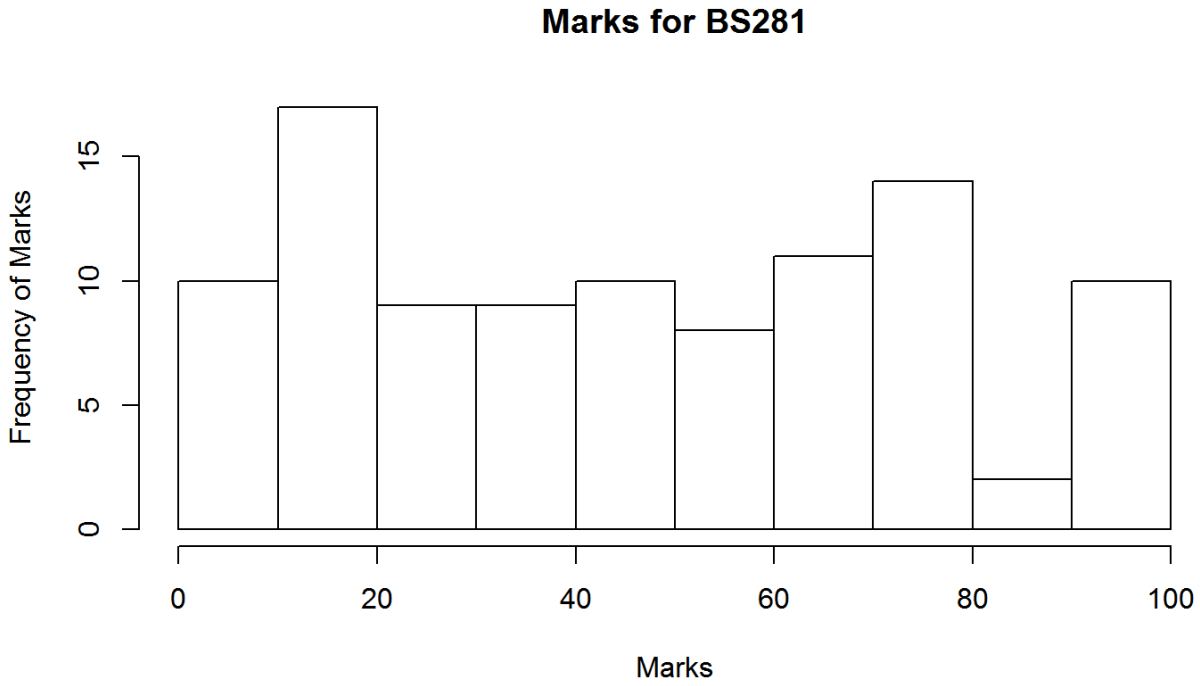
```
##       Length Class      Mode
## BS281 4      data.frame list
## BS282 4      data.frame list
## BS283 4      data.frame list
## BS284 4      data.frame list
## BS285 4      data.frame list
```

- If writeToFile = TRUE, a folder is created with a file for each exams marks

Marking Students

- Using the function **examHist()**, a histogram can be generated using a list containing dataframes for each moduleID and the moduleID being assessed
- each list dataframe should have columns of studentID, degree course, and mark

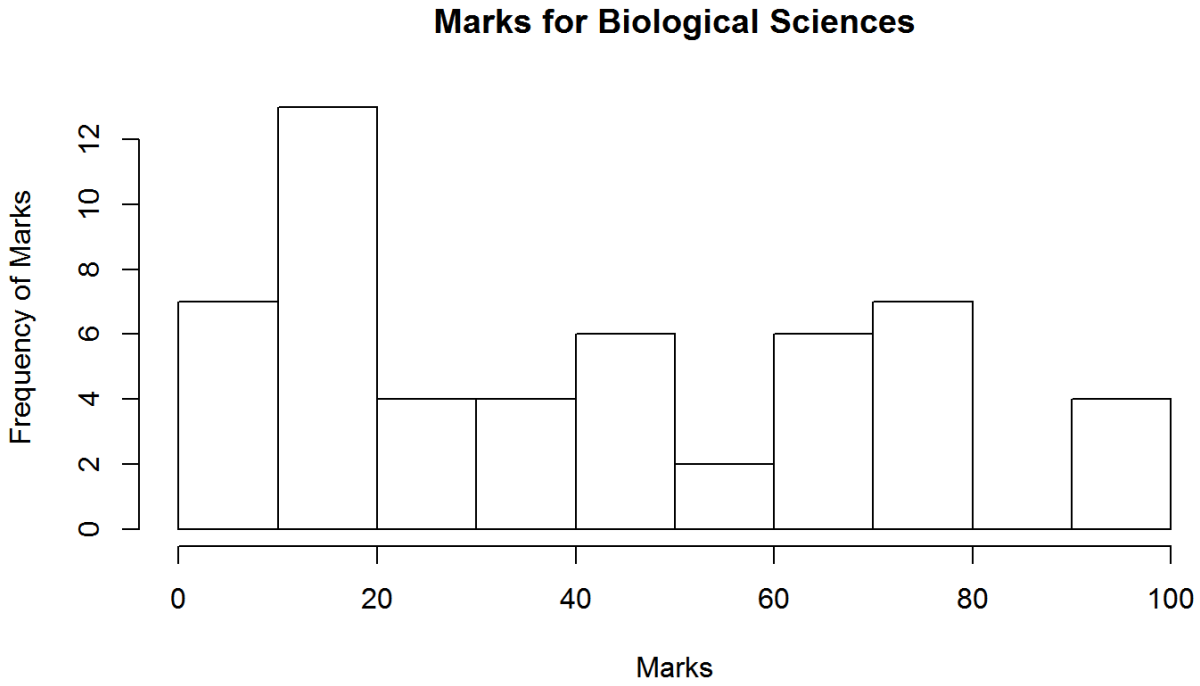
```
examHist(testData, 'BS281')
```



Data Analysis

- Using the function **moduleHist()**, a histogram can be generated using a dataframe and the degree subject being assessed
- The dataframe should have columns of studentID, degree course, and mark

```
moduleHist(testMarks, 'Biological Sciences')
```



Data Analysis

- To analyse the difference in marks between the two degree course within a given module use **testWithinModule()**
- Either a t-test or a wilcoxon test is run
    - A normality test is run to determine which test will be run

```
testWithinModule(testData, 'BS281')
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data[data[, 2] == "Genetics", 3] and data[data[, 2] == "Biological Sciences", 3]
## W = 1569, p-value = 0.02566
## alternative hypothesis: true location shift is not equal to 0
```

Data Analysis

- To analyse the difference in marks between the two degree course for the overall marks use **testBetweenCourse()**
- Either a t-test or a wilcoxon test is run
  - A normality test is run to determine which test will be run

```
testBetweenCourse(testMarks)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data[data[, 2] == "Genetics", 3] and data[data[, 2] == "Biological Sciences", 3]
## W = 1086.5, p-value = 0.2624
## alternative hypothesis: true location shift is not equal to 0
```