Team3Package

Andrew Davis, Andriani Hadjiconstanti, Zamfirescu Ana Maria, and Hannah Leila Tesfay

26-04-2017

```
library(examMarks)
```

This package is designed to generate random answers to a given exam for a given number of students. It can then mark exam answers as well as generating visual representaions of the exam marks and analyzing the mark distribution as well.

Generating random answers

generateStudentAnswersForExam()

By using generateStudentAnswersForExam(), one is able to create an answer sheet based on the total number of questions in an exam and the amount of questions that the student is asked. moduleID and studentID is only required if you are writing to a file as they form the file name.

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

```
##   question answer
## 1        1      e
## 2        6   <NA>
## 3        8      d
## 4       14      c
## 5       16      d
## 6       26      b
```

If one wants to write the generated answer sheet to a file then the file will be saved according to moduleID and studentID. For example this file would be saved as ‘BS281_answers_12.tsv’.

```
generateStudentAnswersForExam(totalNumberofQuestions = 100,
                              numberOfQuestionsToAnswer= 30, writeToFile = T,
                              moduleID = 'BS281', studentID = '12')
```

generateAllStudentsAnswersForExam()

Using the generateAllStudentsAnswersForExam() function one is able to generate answer sheets for multiple students. This function uses the generateStudentAnswersForExam() function witnin it. Datasets are included in this package and are listed as defualt values to demo how the functions runs. readFromFiles must equal TRUE if one needs to use external files to run the function.

In this case only moduleID is required. The output is split into a list with the first part conatining a list of the students taking the module and the second part of the list conatining a list of all the student answers.

```
test = generateAllStudentsAnswersForExam('BS285', writeToFile = FALSE)
head(test[[1]])
```

```
##      ID              degree
## 1 39502 Biological Sciences
## 4 27854 Biological Sciences
## 5 46107 Biological Sciences
## 7  2166 Biological Sciences
## 8 32613 Biological Sciences
## 9 20426 Biological Sciences
```

```
head(data.frame(test[[2]][1]))
```

```
##   X39502.question X39502.answer
## 1               4             c
## 2               9             d
## 3              12             e
## 4              13             c
## 5              14             a
## 6              15             c
```

If one wants to write the answers to files then a folder is created with all of the student answers contained within. A file is also created that lists the students that took the exam.

```
generateAllStudentsAnswersForExam('BS285', writeToFile = TRUE)
```

createAnswerKey()

A randomly generated answer key can also be created using the createAnswerKey() function. The number of questions and the answer options must be specified.

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

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

If the answer key is being written to a file then it is saved according to moduleID. For example this file be called 'correct_answers_BS281.dat'

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

Datasets provided

exams

A dataframe that indicates what modules each degree course takes, with Yes, No, or Optional being listed.

```
head(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
```

keys

A list containing 5 different exam keys of varying length. All keys have possible answers being from “a” to “e”.

students

A dataframe with the ID and degree course for each student.

```
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
```

questions

A dataframe with the total number of questions in an exam and the amount of questions that need to be answered.

```
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
```

Marking Students

markStudentsForExam()

Using this function a dataframe can be created which contains the marks for each student for a given exam. The marks are based on the exam answers randomly generated from generateAllStudentsAnswersForExam().

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

```
##   StudentID              Course Mark
## 1     10161 Biological Sciences   20
## 2     11724            Genetics    3
## 3     11826            Genetics   17
## 4     11832 Biological Sciences   17
## 5     13403 Biological Sciences   10
## 6     13652            Genetics   10
```

getDegree()

This function is designed to be used in the addDegree() function to add the degree to each student. A singular mark is converted to a degree based on the mark.

```
getDegree(61)
```

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

addDegrees()

The purpose of this function is to add a degree for each student, amending the dataframe as an output.

```
head(addDegrees(testMarks))
```

```
##   studentID              course mark Degree
## 1         1 Biological Sciences   61    2:1
## 2         2            Genetics   45    3rd
## 3         3 Biological Sciences   43    3rd
## 4         4            Genetics   22 failed
## 5         5 Biological Sciences   97    1st
## 6         6            Genetics   66    2:1
```

markStudents()

Outputs a list of dataframes of student marks for each module or creates a folder with files with the marks for each exam depending on if writeToFile is TRUE or FALSE.

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

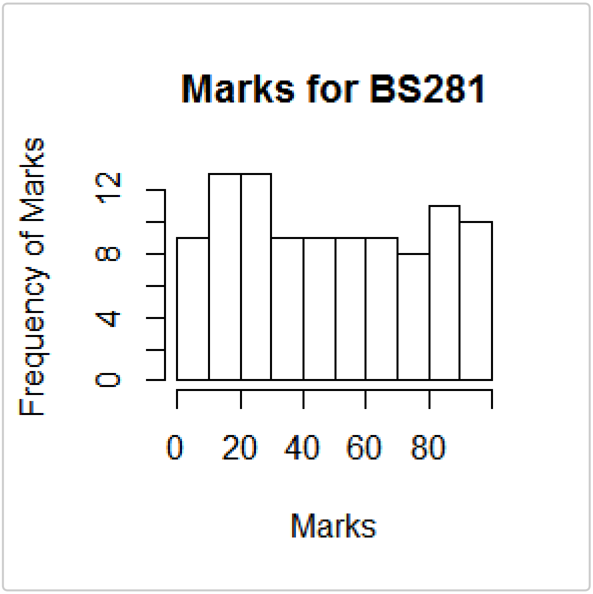
```
##       Length Class      Mode
## BS281 4      data.frame list
## BS282 4      data.frame list
## BS283 4      data.frame list
## BS284 4      data.frame list
## BS285 4      data.frame list
```

Data Analysis

examHist()

A histogram is generated using a list conatining dataframes for each moduleID, with columns of studentID, degree course, and mark, and the moduleID being assesed.

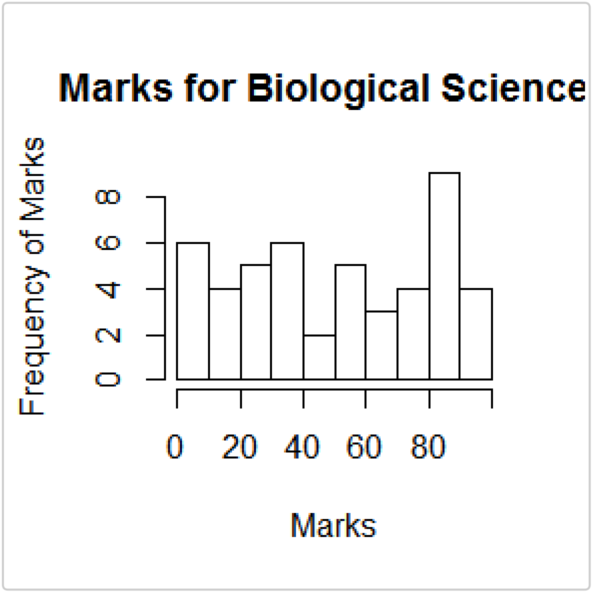
```
examHist(test, 'BS281')
```



moduleHist()

A histogram is generated using a dataframe, with columns of studentID, degree course, and mark, and the degree subject being assesed.

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



testWithinModule()

Either a t-test or a wilcoxin test is run to test for a difference in the distribution of marks between degree courses for a module. A normality test is run to determine which test will be run.

```
testWithinModule(test, 'BS281')
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data[data[, 2] == "Genetics", 3] and data[data[, 2] == "Biological Sciences", 3]
## W = 1146.5, p-value = 0.4858
## alternative hypothesis: true location shift is not equal to 0
```

testBetweenCourse()

Either a t-test or a wilcoxin test is run to test for a difference in the distribution of marks between degree courses for overall marks. 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 = 1330.5, p-value = 0.5715
## alternative hypothesis: true location shift is not equal to 0
```