Open Data Mashup Day: OULAD example

Jakub Kuzilek, Martin Hlosta and Zdenek Zdrahal

10-11-2015

Introduction

This document demonstrates how to use the Open University Learning Analytics dataset (OULAD). The following sections will guide you through the environment setup, data downloading and data manipulation. All codes presented in the document can be downloaded from OULAD webpage. All examples are deployed using **R version 3.2.2** ("Fire Safety") and **RStudio** version 0.99.486. We will be using only a subset of the whole dataset developed for the purpose of the Open Data Mashup Day in London.

Environment setup

First of all you need to download and install **R version 3.2.2** and **RStudio**. After installation of the required software, we need to install package `data.table`, which provides enhanced fuctionality for the `data.frame` data type in R, by executing the following command:

```
install.packages("data.table")
```

After installing, we need to load the library `data.table` into the environment. This is done by executing this command:

```
library(data.table)
```

Example task

Compare an average result of registered female students, who engaged with Virtual Learning Environment in 28th day of presentation, from two different presentations in 1st and 2nd Tutor Marked Assigment (TMA) using combined weighted score.

Data preparation

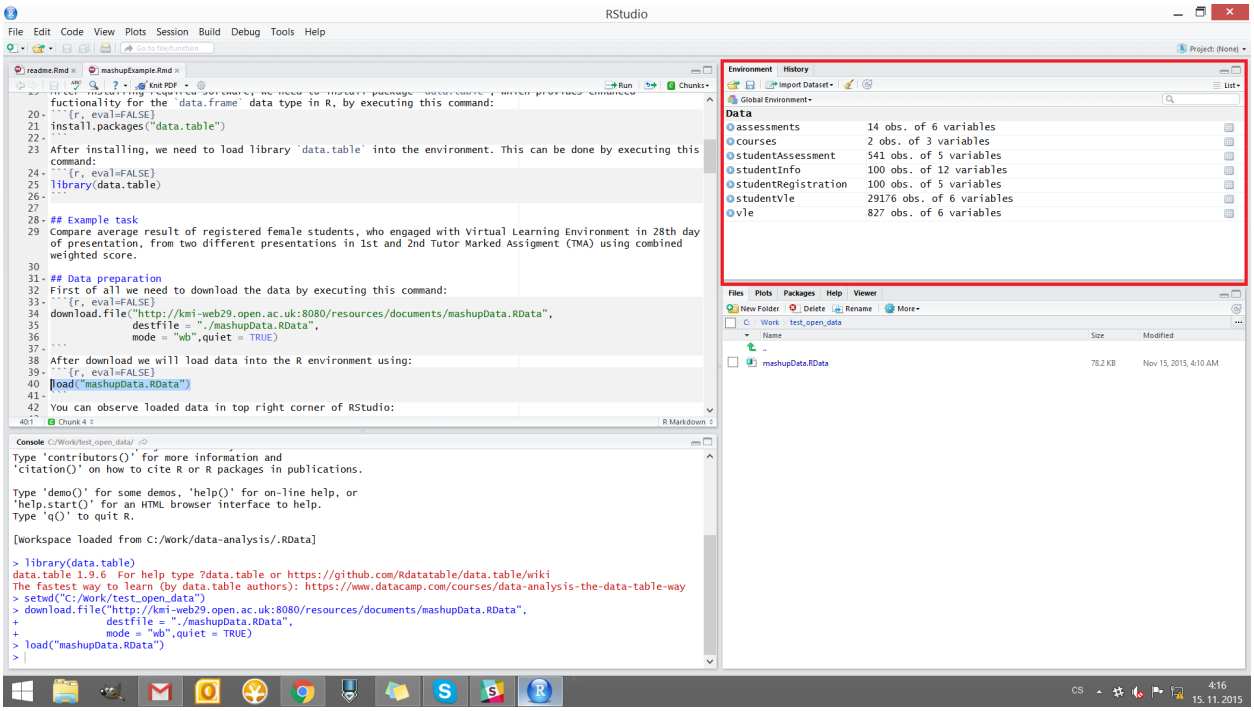
First of all we need to download the data by executing:

```
download.file("http://kmi-web29.open.ac.uk:8080/resources/documents/mashupData.RData",
              destfile = "./mashupData.RData",
              mode = "wb",quiet = TRUE)
```

Then we will load the data into the R environment using:

```
load("mashupData.RData")
```

You can observe the loaded data in the top right corner of RStudio:



Solution

Now, we have the environment set up, the data ready for use and we can start solving the *Example Task*.

Selecting students

First we need to select female students only. The demographic information about students is presented in table `studentInfo`, thus we will manipulate this table to select only female students using following command:

```
femaleStudents <- studentInfo[gender == "F"]
```

The result of selection will be stored in the new variable `femaleStudents` containing demographics of all female students in both presentations.

Selecting registered students

We are interested only in students, who were registered in 28th day of presentation, thus we need to filter out students who unregistered from the module before and during this day. Registration data are stored in table `studentRegistration`. We will select only registered students by executing:

```
registeredStudents <- studentRegistration[
                        is.na(date_unregistration) | date_unregistration > 28
                        ]
```

Note that students, who finished the course have value `NA` in the `date_unregistration` column, thus we need to select students with `NA` (finished the module) or unregistered after 28th day of presentation.

Selecting VLE active students

We are also interested in students, who were active in the VLE at 28th day of presentation. Thus we need the information contained in table `studentVle`. We need to select log entries with `date` column equal to 28. Because we need only the identification of the student, and the presentation he/she studied we will extract only three columns as follows:

```
activeStudents <- studentVle[date == 28][,
                                         .(id_student,
                                           code_module,
                                           code_presentation)
                                         ]

setkey(activeStudents, id_student, code_module, code_presentation)

activeStudents <- unique(activeStudents)
```

By setting the table primary key (columns, which identifies the record) using `setkey` command followed by the `unique` command, we will select only unique identification rows, which is exactly what we need.

Combining student scores from first and second TMA

We have prepared the subsets of data answering parts of the task. Now we have to select the first and second TMA from each presentation, get their weights and use them to calculate the combined score for each student. Assessment weights are stored in table `assessments`. First, we will extract the `id_assessment` values of all four assessments required to solve our task:

```
assessments_codes <- assessments[assessment_type == "TMA"]
assessments_codes <- assessments_codes[order(date)][1:4, id_assessment]
```

There exist more types of assessments, We are interested only in TMAs, so first we need to filter only them from the rest. Then the table is reordered according to the assessment cutoff date (note that the first and second assessments have cutoff dates lower than third one in all presentations) and then we extract first four identifiers from the reordered table, which represents the identifiers of the first and second TMA in both presentations. Next we will use `assessments_codes` to extract all neccessary information from `assessments` table executing the following command:

```
selectedAssessments <- assessments[id_assessment %in% assessments_codes]
```

Now we have got the required information about the assessments and we need to combine this information with the student results in `studentAssessment` table. This can be done by setting primary keys for `selectedAssessments` and `studentAssessments` and then joining them together leaving only the first and second TMA results:

```
setkey(selectedAssessments, id_assessment)
setkey(studentAssessment, id_assessment)

studentAssessmentWithWeights <- studentAssessment[selectedAssessments]
```

Finally, we need to combine the TMA results with the corresponding weights by executing:

```
studentAssessmentWithWeights[, weightedScore := score*weight/100]
```

And then for each student we calculate the sum of weighted score of all the assessments together by executing:

```
studentResults <- studentAssessmentWithWeights[,
                                               .(score = sum(weightedScore)),
                                               by=.(id_student,
                                                    code_module,
                                                    code_presentation)
                                               ]
```

Putting everything together

Finally, we have ready everything we need. Let start combining partial answers together. First we will combine `femaleStudents` with `registeredStudents` to get those female students, who are still progressing in module at 28th day of presentation, by executing:

```
setkey(femaleStudents, id_student, code_module, code_presentation)
setkey(registeredStudents, id_student, code_module, code_presentation)

registeredFemaleStudents <- femaleStudents[registeredStudents, nomatch=0]
```

Next we combine the new table with table `activeStudents` to select those students, who were active at 28th day of presentation, executing this command:

```
activeRegisteredFemaleStudents <-
                registeredFemaleStudents[activeStudents, nomatch=0]
```

The last step is adding score to each selected students by executing the following command:

```
activeRegisteredFemaleStudentsWithScore <-
                activeRegisteredFemaleStudents[studentResults, nomatch = 0]
```

Comparison of student results from different presentations

Now we have all the information we needed and we can solve the task by executing the following command:

```
print(activeRegisteredFemaleStudentsWithScore[,
                                              .(mean.score = mean(score)),
                                              by=code_presentation])
```

And the result is:

```
##    code_presentation mean.score
## 1:             2013J     16.155
## 2:             2014J      7.500
```

Licensing and contact information

- Jakub Kuzilek (jakub.kuzilek@gmail.com)
- Zdenek Zdrahal (zdenek.zdrahal@open.ac.uk)
- Martin Hlosta (martin.hlosta@open.ac.uk)