moodleR: Randomized Moodle Quizzes with R

Dr. Wolfgang Rolke

Jube 13, 2021

R

First you need to download the package *moodleR.zip* from

https://github.com/WolfgangRolke/moodleR/tree/main/library

extract the compressed files and move the folder to your R\library directory. Then you can run

```
library(moodleR)
```

You also need the ShinyMatrix library:

```
install.packages("shinyMatrix")
```

to have everything in order.

If you are not familiar with R I have a lengthy introduction to R that I gave at a workshop in Hamburg, Germany in 2017 http://academic.uprm.edu/wrolke/research/Introduction%20%20to%20R.pdf, or you can study the (much more detailed) material of my R course at http://academic.uprm.edu/wrolke/Computing-with-R

Basic Workflow

1) Create a file quizxyz.R either with imoodleR or with an ASCII word processor.

2) Create a file called *newquiz.xml*. This is done automatically by the shiny app, or by running

```
make.xml(quizxyz, 20)
```

3) Go to moodle, the Question section, and Import newquiz.xml.

imoodleR

The best way to get started is by using the interactive shiny app. Start it with

```
imoodleR()
```

It is largely self-explanatory (I hope!) and also has a few hints on what to do. Here are some general suggestions:

- when you type in the name (and folder) of a previously created quiz all the information will be displayed, so one doesn't have to start from scratch each time.
- empty spaces should be avoided, so for example don't use 'Year 1' as a nimae of an object but use 'Year1'.

quizxyz.R

The "heart"of moodleR are files that can be turned into a newquiz.xml. These have to have a certain structure.

Here is a very simple example of a problemxxx.R:

```
example1 <- function() {
   category <- "moodler / Percentage 1"
   quizname <- "problem - "
#----------------------
# Question 1
#----------------------
   n <- sample(200:500, 1)
   p <- runif(1, 0.5, 0.6)
   x <- rbinom(1, n, p)
   per <- round(x/n*100, 1)
   qtxt <- paste0("Question 1: In a survey of ", n, "
      people ", x, " said that they prefer Coke
      over Pepsi. So the percentage of people who prefer
      Coke over Pepsi is
      {:NM:%100%", per,":0.1~%80%", per, ":0.5}%")
   htxt <- "Don't forget to multiply by 100, don'tinlcude % sign in answer"
   atxt <- paste0("Question 1: x/n*100 = ", x, "/", n,
          "*100 = ", per, " (rounded to 1 digit)")
#----------------------
# Question 2
#----------------------
   p1 <- round(runif(1, 50, 60), 1)
   if(per<p1) mc <- c("{:MC:~%100%lower~%0%the same~%0%higher}", " < ")
   if(p1==per) mc <- c("{:MC:~%0%lower~%100%the same~%0%higher}", " = ")
   if(per>p1) mc <- c("{:MC:~%0%lower~%0%the same~%100%higher}", " > ")
   qtxt <- paste0(qtxt, "<p>Question 2: In a survey some
     years ago the percentage was ", p1, "%.
    So the percentage now is ", mc[1])
   htxt <- ""
   atxt <- paste0(atxt, "<p>Question 2: ", per , mc[2], p1)

   list(qtxt = paste0("<h5>",qtxt,"</h5>"),
        htxt = paste0("<h5>", htxt,"</h5>"),
        atxt = paste0("<h5>", atxt,"</h5>"),
        category = category, quizname = quizname)
}
```

The output of the routine has to be a list with four elements:

a) **category**: this is the category under which moodle will file the the problem.

b) **quizname**: the name of the problem (usually just problem -)

c) **qtxt**: the ENTIRE text of the problem as it will appear to the students.

d) **htxt**: Hints that students see after their first attempt

e) **atxt**: the ENTIRE text of the answers that the students see after the deadline for the quiz has passed.

Note the <h5> at then end of the routine. This is html code for *heading 5*. I use this here to make the text a bit more readable in moodle. But more generally we can use any html code and moodle knows what to do with it. The same is true for latex, as we will see later.

CLOZE

moodle has a number of formats for writing questions. I use the cloze format exclusively because it has a number of features that work well in the kinds of quizzes I typically do. There are three basic question types:

a) Multiple Choice

Example: For this data set the mean is {1:MC:~%0%lower~%0%the same~%100%higher}.

In this case the students are presented with a drop-down box with the three options: lower, the same and higher.

higher is the correct answer, so it gets 100%, the others get 0%. One can also give (say) 65% for partial credit. The 1 in the front means the problem is worth 1 point.

b) Numerical Answers

Example: The mean is {2:NM:%100%54.7:0.1~%80%54.7:0.5}.

Now the students see a box and have to type in a number.

Here 54.7 is the correct answer, which get's 100%. Any answer within ± 0.1 is also correct, allowing for rounding to 1 digit. The answer 55 gets 80% (a bit to much rounding) .

c) Text answers:

Example: The correct method for analysis is the {1:SA:correlation coefficient}

Here the student sees a box and has to type in some text.

There is also {1:SAC:correlation coefficient} if the case has to match.

This type of problem has the issue of empty spaces. So if the student typed correlation coefficient with two spaces it would be judged wrong. The solution is to use *s: {1:SA:*correlation*coefficient*}

In the package *moodleR* there are some routines that make this easier:

- *mc* creates the code needed for multiple choice questions. It has a number of standard choices already implemented.

```
mc(options, w)
```

here options is a character vector with the choices (or a number for some common ones that I predefined), w is the vector of percentages, usually 100 for the correct answer and 0 for the others.

- *nm* for numerical questions:

```
nm(x, w, eps, ndigits, pts = 1)
```

x is a vector of possible answers, w is the vector of percentages and eps is a vector of acceptable range ±. Alternatively one can use the argument ndigits. With (say) ndigits=1 the answer has to be rounded to one digit behind the decimal, all other roundings get partial credit.

If just one number is correct, say 54.7, and the answer has to be given exactly, use *nm(54.7)*.

- *sa* for text answers

```
sa(txt, w=100, caps=TRUE, pts=1)
```

the function automatically adds *s in a number of places. txt can be a vector.

With these routines we can update out moodle quiz:

```
example2   <- function() {
   category <- "moodler / Percentage 2"
   quizname <- "problem - "
#----------------------
# Question 1
#----------------------
   n <- sample(200:500, 1)
   p <- runif(1, 0.5, 0.6)
   x <- rbinom(1, n, p)
   per <- round(x/n*100, 1)
   qtxt <- paste0("Question 1: In a survey of ", n, "
       people ", x, " said that they prefer Coke over
       Pepsi. So the percentage of people who prefer
        Coke over Pepsi is ",
      nm(c(per, per), c(100, 80), c(0.1, 0.5)), "%")
   htxt <- "Don't forget to multiply by 100, don'tinlcude % sign in answer"
   atxt <- paste0("Question 1: x/n*100 = ", x, "/", n, "*100 = ", per, " (rounded to 1 digit)")
#----------------------
# Question 2
#----------------------
   p1 <- round(runif(1, 0.5, 0.6)*100, 1)
   if(per<p1) {w <- c(100, 0, 0); amc <-"<"}
   if(per==p1) {w <- c(0, 100,  0); amc <-"="}
   if(per>p1) {w <- c(0, 0, 100); amc <-">"}
   opts <- c("lower", "the same", "higher")

   qtxt <- paste0(qtxt, "<p>Question 2: In a survey some
        years ago the percentage was ", p1, "%.
        So the percentage now is ", mc(opts, w))
   htxt <- ""
   atxt <- paste0(atxt, "<p>Question 2: ", per , amc, p1)

   list(qtxt = paste0("<h5>",qtxt,"</h5>"),
        htxt = paste0("<h5>", htxt,"</h5>"),
        atxt = paste0("<h5>", atxt,"</h5>"),
        category = category, quizname = quizname)
}
```

Creating the .xml file

Once the quizxyz.R is written we can read it into R and then use the *make.xml* routine to create the moodle input file *newuiz.xml*:

```
make.xml(quizxyz, 20)
```

does it all!

Next we need to open Moodle and import this file.

Note Moodle has two Import places. One is for importing existing questions from other courses. We need to go to Questions - Import, select XML, drop newquiz.xml into the box and hit enter.

Multiple Stories

In statistics we like to use word problems. So how can we do that? Essentially we can make up a couple of stories:

```
example3 <- function(whichstory) {
   category <- paste0("moodler / Percentage - Story - ", whichstory)
   quizname <- "problem -"

   if(whichstory==1) {
      n <- sample(200:500, 1)
      p <- runif(1, 0.5, 0.6)
      x <- rbinom(1, n, p)
      per <- round(x/n*100, 1)
      qtxt <- paste0("In a survey of ", n, " people ", x, "
        said that they prefer Coke over Pepsi. So the
        percentage of people who prefer Coke over
        Pepsi is ",
        nm(c(per, per), c(100, 80), c(0.1, 0.5)), "%")
   }
   if(whichstory==2) {
      n <- sample(1000:1200, 1)
      p <- runif(1, 0.45, 0.55)
      x <- rbinom(1, n, p)
      per <- round(x/n*100, 1)
      qtxt <- paste0("In a survey of ", n, " likely
        voters ", x, " said that they would vote for
        candidate A. So the percentage of people who
        will vote for candidate A is ",
        nm(c(per, per), c(100, 80), c(0.1, 0.5)), "%")
   }
   if(whichstory==3) {
      n <- sample(100:200, 1)
      p <- runif(1, 0.75, 0.95)
      x <- rbinom(1, n, p)
      per <- round(x/n*100, 1)
      qtxt <- paste0("In a survey of ", n, " people ",
        x, " said that they are planning a vacation
        this summer. So the percentage of people who
        are planning a vacation is ", nm(c(per, per),
```

```
        c(100, 80), c(0.1, 0.5)), "%")
   }
   htxt <- ""
   atxt <- paste0("x/n*100 = ", x, "/", n, "*100 = ",
          per, " (rounded to 1 digit)")

   list(qtxt = paste0("<h5>",qtxt,"</h5>"),
        htxt = paste0("<h5>", htxt,"</h5>"),
        atxt = paste0("<h5>", atxt,"</h5>"),
        category = category, quizname = quizname)
}
```

Note we can match each story with likely numbers, so in the Coke vs Pepsi story n is between 200 and 500 whereas in the votes story it is between 1000 and 1200.

Note that this routine has the argument *whichstory*, which we can add to genquiz or make.xml:

```
make.xml(example3, 20, whichstory=2)
```

Data Sets

Often in Statistics we have data sets the students need to use. So these data sets need to be displayed properly in the quiz and it must be easy for the students to "transfer" them to R.

To display the data in the quiz use the *moodle.table* function. If it is called with a vector it arranges it as a table with (ncol=) 10 colums. If it is called with a matrix or data frame it makes a table as is.

To get the data from the quiz into R Resma3 has the routine *paste.data()*. All the students have to do is highlight the data (including column names if present) in the quiz with the mouse, right click copy, switch to R and run

```
paste.data()
```

There is now an object called x in R. If it is a single vector it can be used as is, say

```
mean(x)
```

If the data was a table with several columns the dataframe is also automatically attached, so the students can immediately start using them.

So we could have another version of

```
example4 <-
function() {
   category <- "moodler / Percentage from Raw Data"
   quizname <- "problem - "

   n <- sample(200:500, 1)
   p <- runif(1, 0.5, 0.6)
   x <- sample(c("Coke", "Pepsi"), size=n,
               replace=TRUE, prob=c(p,1-p))
   per <- round(table(x)[1]/n*100, 1)
```

```
   qtxt <- paste0("In a survey people were asked whether
    they prefer Coke over Pepsi. Their answers
    are below. So the percentage of people who
    prefer Coke over Pepsi is ",
    nm(c(per, per), c(100, 80), c(0.1, 0.5)), "%<hr>",
    moodle.table(x))
   htxt <- ""
   atxt <- paste0("x/n*100 = ", table(x)[1], "/", n,
            "*100 = ", per, " (rounded to 1 digit)")

   list(qtxt = paste0("<h5>",qtxt,"</h5>"),
        htxt = paste0("<h5>", htxt,"</h5>"),
        atxt = paste0("<h5>", atxt,"</h5>"),
        category = category, quizname = quizname)
}
```

Graphics

Often in statistics we use graphics. We can do this as follows: first you need to install the R package **base64**. Then we need the function

```
png64 <- function(plt) {
  pngfile <- tempfile()
  png(pngfile, width = 400, height = 400)
  print(plt)
  dev.off()
  pltout <- img(pngfile, Rd = TRUE, alt = "a")
  m <- nchar(pltout)
  pltout <- substring(pltout, 6, m-1)
  pltout
}
```

Now we can write

```
example5 <- function(bell=TRUE) {
  require(base64)
   category <- paste0("moodler /
      bell-shaped? - ", ifelse(bell, "Yes", "No"))
   quizname <- "problem - "

   n <- 1000
   if(bell) x <- rnorm(n, 10, 3)
   else x<- rchisq(n, 2) + 8

   plt <- hplot(x, n=50, returnGraph=TRUE)
   plt64 <- png64(plt)
   if(bell) mmc <- mc(5, c(100, 0))
   else mmc <- mc(5, c(0, 100))
   qtxt <- paste0("The data shown in this histogram ",
              mmc, " bell-shaped<hr>", plt64)

   htxt <- ""
```

```
   if(bell) atxt <- "It is bell-shaped"
   else atxt <- "It is not bell-shaped"

   list(qtxt = paste0("<h5>", qtxt, "</h5>"),
        htxt = paste0("<h5>", htxt, "</h5>"),
        atxt = paste0("<h5>", atxt, "</h5>"),
        category = category, quizname = quizname)
}
```

Non Statistics Courses

The same basic ideas can be used to make random quizzes for other courses. In principle any other computer language can be used as well, but I will stick with R and make a quiz for calculus:

```
example6 <- function() {
   category <- "moodler / Integral"
   quizname <- "problem - "

   A <- round(runif(1), 1)
   B <- round(runif(1, 1, 2), 1)
   fun <- function(x) {x*exp(x)}
   I <- round(integrate(fun, A, B)$value, 2)
   qtxt <- paste0("$$\\int_{", A, "}^{", B, "} xe^x dx = $$",
   nm(I, eps=0.1))
   htxt <- ""
   atxt <- paste0("$$\\int_{", A, "}^{", B, "} xe^x dx = $$", I)
   list(qtxt = paste0("<h5>", qtxt, "</h5>"),
        htxt = paste0("<h5>", htxt, "</h5>"),
        atxt = paste0("<h5>", atxt, "</h5>"),
        category = category, quizname = quizname)
}
```

Note here we see that one can use latex notation in Moodle quizzes, and so display formulas!