How to use R package exams as a replacement for scantrons

Filipe Alberto

2025-11-22

Contents

1 Introduction

This manual is intended to facilitate the use of the R package exams as a replacement tool for scantrons. Note that, beyond creating exams and associated scantrons, the package can be used to create quiz questions that can be uploaded into Canvas (topic not covered here). In brief, the package will produce a PDF with the exam and a scantron (maximum 45 questions). Questions can be single choice (one correct answer) or multiple choice (one or more correct answers). You can easily create different versions of the exam, by shuffling or randomly selecting, or both, questions from a larger number of possible answer choices.

Students answer their exam questions on the answer sheet, the instructor then scans the answer sheets with the department's photocopy machine (so fast). The resulting PDF containing scans of all students' answer sheets is analyzed in R using functions from the R exams package. A file containing the answers, the right/wrong keys, and scans of the answer sheets is produced for each student. A text file with the information for all students is also produced.

Using this output file, I coded a few more exam diagnostic descriptive statistics to mimic the most relevant information provided by the exam center when analyzing scantrons. Below you will find detailed instructions for installing the package and its requirements, a pipeline, and a few notes to ensure your scantrons scan correctly. I will provide some suggestions for exam day, given the differences between this system and the Scantron. I will also provide links to the R package exams webpage, where you can find plenty of additional information.

2 R package exams installation and requirements

The details for the package installation can be found here. I summarized the relevant information below.

2.1 Installing R and RStudio

Start by installing Rstudio if you don't have it. You will need R installed in your system for the installation to work. For Windows, the Rtools package should also be installed. Here is a great page explaining how to install R and Rtools on Windows. Essentially, install R first, then RStudio, open RStudio and look in the lower left console what the version of R is (e.g., 4.3), then install the Rtools version with the same number.

2.2 Install the R exams package

The code below should install the stable version of the package and all other R packages it depends on.

```
install.packages("exams", dependencies = TRUE)
```

In some setups (e.g., on macOS or with an older version of R), it may be necessary to add the argument type = "source" to the command above.

```
install.packages("exams", dependencies = TRUE,type="source")
```

2.3 LaTeX requirements

For producing PDF output, R/exams uses the typesetting system LaTeX internally. Suppose no LaTeX distribution (like TeXLive or MikTeX) is already installed. In that case, TinyTeX is a lightweight distribution that can be easily obtained and maintained with the tinytex R package (already installed in the step above). TinyTeX can be installed from within R with:

```
tinytex::install_tinytex()
```

2.4 Ghostscript

The photocopy machine scans the scantrons into a single PDF file, which later needs to be converted to PNG internally. For this reason, Ghostscript must be installed and available in the system path. Ghostscript is

usually installed on Linux, but might require additional installation on Windows and macOS.

For Windows, follow these instructions. For macOS, Via Homebrew (https://brew.sh) in the terminal.

```
brew install ghostscript
```

For Linux, if it is not already installed, install Ghostscript from your distribution, e.g., Debian/Ubuntu.

```
sudo apt-get install ghostscript
```

2.5 R Markdown

This is necessarty to run the descriptive statistic R Markdown script that will produce tables and plots similar to what we get from UWM's exam center. We already took care of Latex above, which is necessary if you want to render the diagnostic plots to PDF. However, I suggest rendering to HTML for correct visualization.

```
install.packages(c("rmarkdown","knitr"))
```

2.6 Setting the R session working directory

If you haven't used R much before, you may get frustrated by the concept of the R session working directory. This is the directory in your computer associated with a particular R session you might be running. To keep things tidy, you should have different folders for each of your projects, particularly here, for other exams. Beyond file organization, this will make your life much easier when reading files into R by eliminating the need to write long file paths. If the file being read is in the working directory, these R commands will require only the file name. The same goes for writing files from R and finding them in the R session working directory without having to write long paths.

The easiest way to manage which folder is the current working directory of the R session is through RStudio's *Project* concept. Thus, follow these steps:

- Create a directory on your machine from which you want to work (e.g., BIOSCI465-Exam.I).
- Open RStudio and click on the second icon in the upper left corner (Create a Project). Select *existing directory* and browse through your files to find the directory
- Once you select the folder and hit the return key, the project is created. You can check that the working directory is correctly set with the following code:

```
getwd()
```

Inside the working directory, you will find a file with the extension *.Rproj*. You can double-click on this file the next time you want to start RStudio to resume your work.

2.6.1 Note for Windows users

For quite some time now, the Windows File Explorer program has hidden file extensions. It only shows the file name without the dot and three-character code that constitutes the extension (which are part of the file name!). R needs the entire file name to recognize files, and is case sensitive (i.e., File.txt is different from file.txt). Change the default behavior in Windows File Explorer so you can see the full names of your files. A quick browse should show you how to do this for your Windows version.

3 First steps

To check that the installed software works, try running some examples from the exercise template gallery, e.g., *dist.Rmd* or *ttest.Rmd*.

To install all the example files, use the following code:

```
library("exams")  #this loads the package once installed,
                  #you need to load the package for each new session
exams_skeleton(markup = "markdown",
  writer = c("exams2html", "exams2pdf", "exams2moodle"))
```

The code above copies all R/Markdown files to the current working directory along with demo scripts, illustrating the usage of exams2html() and exams2pdf() for customizable HTML and PDF output, respectively.

You can test that the following code opens a webpage with one of the questions created by the code above (located in the exercises folder). This is a quick test to see if the installation above worked. If you use Windows, the path in the code below should be edited to ".\exercises\ttest.Rmd".

```
exams2html("./exercises/ttest.Rmd")
```

See the First Steps tutorial for detailed instructions. Below, I will provide a shorter version using single- and multiple-choice questions as examples, since these are the types of questions on our Scantron exams. Note that you can also create numeric questions (questions that require a numerical answer) if you want to develop questions to upload to Canvas.

3.1 Creating individual questions

The text files used to create individual questions can be produced for Markdown or LaTeX. Markdown is easier, and that's what I will use here. An R Markdown file can be edited directly in RStudio. Just open a new text file and save it with a name that reminds you what the question is about. Don't forget to add a .Rmd extension to the file name (e.g., myQuestion.Rmd).

Below is an example of the code necessary for an Rmd file for a single-choice question. Further below, I will explain the different fields in this file.

```
Question
========
What is the seat of the federal
authorities in Switzerland
(i.e., the de facto capital)?

Answerlist
----------
* Basel
* Bern
* Geneva
* Lausanne
* Zurich
* St. Gallen
* Vaduz

Solution
========
There is no de jure capital but
the de facto capital and seat of
the federal authorities is Bern.

Answerlist
----------
* False
* True
* False
```

```
* False
* False
* False
* False

Meta-information
================
exname: Swiss Capital
extype: schoice
exsolution: 0100000
exshuffle: 5
```

First, the *Question* field is straightforward and lists seven answer alternatives under *Answerlist*. Here, the answer list can contain more possible answers than will be printed in the exam, so that you can produce shuffled versions of your exams.

The correct solution (Bern) is declared in the *metainformation* field below.

Do not alter the general formatting of the dashes and the headers, like *Question* or *Answerlist*, and do not add spaces or tabs before writing these headers. I suggest copying the code from one of the Rmd files in the exercises folder and editing it to your own questions.

The *Solution* is an optional field that can be used to provide some general feedback (that explains why Bern is correct in this question). It is also possible to have no solution, or just a general text, or just the answer list.

In the *Meta-information*, each exercise must specify at least an *extype*, and an *exsolution*, and should typically also have a short descriptive *exname*. The *extype* for single-choice questions is *schoice* (and *mchoice* for multiple correct answers). The *exsolution* is simply a binary coding of the seven answer alternatives (zero for incorrect and one for correct). The *exshuffle* is used to randomize the options in the answer list; here set to 5, the maximum number that fits in the printed exam answer sheet, but could be as low as 2, e.g., for TRUE/FALSE questions. With the choice of 5, 1 correct and four random wrong alternatives are subsampled and shuffled from the *answerlist* above, respectively.

3.2 Adding figures to your questions

You can add figures generated by R (using markdown code chunks) or imported from your computer. To import a figure from somewhere in your computer, insert and edit the following R Markdown code in the text of your Rmd question file (usually in the *Question* field).

```
begin{figure}[H]
\centering
 \includegraphics[width=16cm,
height=9cm]{/home/filipe/Documents/jobs/UW-Milwaukee/Teaching/Rexams/images/Correlation_sign.png}
 \caption{Draw points here to represent the associations.}
\end{figure}
```

You can change the width and height of the figure and add a caption by editing the code above. Don't forget to edit the path to your image. The row specyfying a centered image (centering) is optional.

Create as many questions as you would like, following these principles. Next, we will take a look at how to create exams from these individual question files.

4 Creating an exam and answer sheet (scantrom) ready to print

4.1 Creating the exam

The official R exams package tutorial on this topic can be found here. Below is an abbreviated version.

Create a R script (a new text file with extension .R) in RStudio for each new exam. This will contain all the code necessary for creating the exam, evaluating the scans, and grading.

First, create a list with all the questions you want in the exam. Here we are using questions stored in the exercises folder. In practice, use files that you have authored and stored somewhere locally.

```
library("exams") #if not loaded yet

# a small only example  with 5 questions
myexam <- list(
  "tstat2.Rmd",
  "ttest.Rmd",
  "relfreq.Rmd",
  "anova.Rmd",
  "cholesky.Rmd"
)
```

Next, we create a small exam with only n = 2 randomly drawn versions, storing the resulting PDF files (plus metadata) on disk in a new output directory, *nops_pdf*. To customize the exam, we assign different point values to the exercises and display the corresponding point values at the beginning of each question.

Before you run the code below make sure you change the R working directory to the folder where your *.Rmd* questions (used above to create *my.exam* list) are saved, if saved in a different directory. Use the *Files* tab in RStudio's lower-right window to navigate to that folder, then select the gear icon and choose *Set as Working directory*.

```
library("exams") #if not loaded yet

set.seed(403) #to make sure you could repeat these exact permutations

ex1 <- exams2nops(myexam, n = 2,
  dir = "nops_pdf", name = "Exam1", date = "2015-07-29",
  points = c(1, 1, 1, 2, 2), showpoints = TRUE,logo="uwm.png",blank=0,
    institution="UW-Milwaukee",title="BIOSCI 465, Biostatistics exam I",
  usepackage = "float",samepage = TRUE)
```

Above, you should customize the *name* argument used here to add a prefix to the output files. The *title* should be used to name the exam and will be printed on the answer sheet, e.g., "BIOSCI 465, Biostatistics exam I". You can also add a UWM logo; you will need the logo PNG file stored locally and the figure path. It is easier if you place this PNG file in the working directory of the R session and then use the code as in the example above. The *institution* argument will also be printed in the answer sheet. As mentioned above, the *n*=2 creates two versions of the exam; you can create more for large classes. The argument *dir* specifies the directory where exams and some metadata will be saved. If this directory doesn't currently exist in your working directory R will create it. Finally, the *blank* argument specifies the number of blank pages to print after the questions booklet. Students can use these for notes if you prefer to provide the blank paper yourself.

After running the function above, the output directory will then contain three files generated by the code above: "Exam_I.rds", "Exam_I1.pdf", and "Exam_I2.pdf". The two PDF files are the exams (their first page is the answer sheet).

Furthermore, the metainformation about the exam (exam IDs, questions, correct and wrong answer alternatives) is stored in the "Exam_I.rds" file (serialized R data). This is crucial for evaluating the exam later.

5 Printing the exams and the answer sheet

Below are a few recommendations and warnings about printing the exams.

- I found that **setting the paper to A4** on my printer, when printing the first page (answer sheet, Figure 1), provided the correct scaling that solved issues I was having with the evaluation of scans (see below). Print the template on your own printer, then bring it to the photocopy machine to make all the copies. Print more answer sheets than the number of exams needed, in case students need a new one; they will be using a pen instead of a pencil, so no erasing is possible. To minimize the number of students requesting new answer sheets, advise them to mark the correct answers first in the exam questions' booklet and then, at the end of answering all questions, complete the answer sheet.
- The authors recommend not to scale the printout (i.e., without "Fit to printable area") and to staple the exams in the top-left corner. I stapled only the question part (questions' booklet) and left the answer sheets (first page) separate. More work is required to supply the exams and answer sheets separately, particularly if more than one exam version is used. Still, way less work is needed later by not having to unstaple the answer sheet from the questions' booklet part.
- Answer sheets for different versions of your exam (two versions in this example) have different numbers; the questions' booklet comes printed on all pages with the matching version number.
- There is no field on the first page of the questions' booklet section for the student's name, only on the answer sheet. I suggest you ask the students to write their names on the first page of the questions' booklet (if you are like me and want the questions back).
- The package uses the registration number to match grades with student names. The maximum number of digits is seven; UWM has way too many student identification numbers! I suggest using the Panther ID, which is a nine-digit number but starts with 99 for all students (I believe). Tell your students to use the last seven digits as their code in the answering sheet (see image below).
- By default, the PDFs from exams2nops() have a blank second page for duplex printing (without content on the back of the answer sheet).

6 Conducting the exam

A few key differences worth mentioning with this system, when compared with the departing Scantron, that affect the usual way we conduct this type of exam.

- The answer sheet is completed with a pen and printed on regular paper. Combined with the fact that the scan is sensitive to any marking, it is thus impossible to erase marks. **Students can correct errors by filling the whole surface of a previously crossed box with ink (i.e, a black square)**. However, if they want to revert the selection to the original, they will need a new answer sheet.
- Given the last point, please require students to first complete the exam directly in the question's booklet and only at the end fill in the answer sheet. Instruct students that if they make an error, they can cover the box entirely with ink; the scan will not detect a black square as an answer. This will minimize the number of additional answer sheets you need to give students.
- If you have different exam versions, when giving students new answer sheets, remember that the answer sheets are matched with the different exam versions. Both answer sheet and questions' booklet are marked with matching numbers.
- When giving a new answer sheet to a student, immediately remove the old one to minimize the chances of cheating. It is probably not a problem if you have a few different exam versions. Removing the old answer sheet will only work if students first complete their answers on the questions' booklet and only at the end on the answer sheet; otherwise, they will want to copy from one answer sheet to the other.
- If a student forgot their ID, note their name and tell them to use a seven-digit number (of your choice) that you are sure to be unique, like 9999999 (give different numbers to different students facing this issue). Note the "new" number too because you will later need it to edit the registration file (see below).

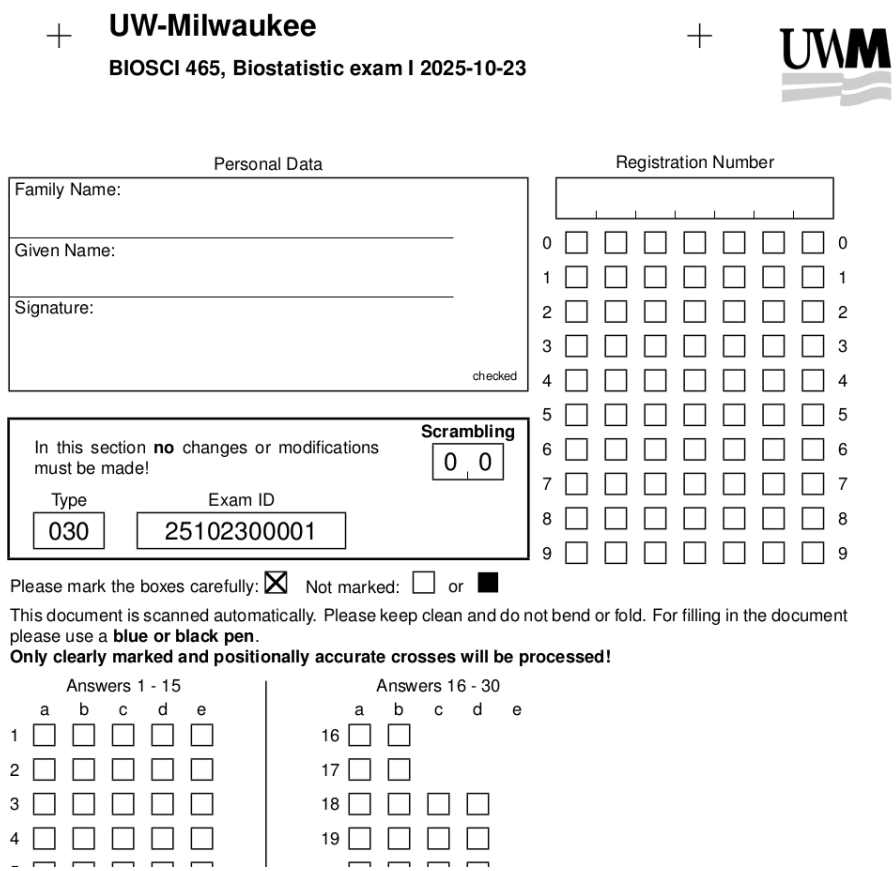


Figure 1: Top part of the exam answer sheet, with instructions on how to fill the form. Note the differences from the scantrons previously used.

7 Scan answer sheets

After students complete the exam, bring the answer sheets and a USB stick to the photocopy machine. You can also bring a laptop (or use the computer in the room) to quickly review the resulting PDF. The scanner will produce a single PDF containing scans for all students. The code allows for some rotation due to how well the machine fed the answer sheets (try to observe the feeding process to detect if any excessive rotation occurs; it's fast). Once the scan is complete, you can quickly view it on a computer. The scan is quick, but if you see any excessive rotation, scan it all again. If you have many students, you can repeat the scan only for those who showed excessive rotation. You can then use Adobe Acrobat or other software later to delete and replace pages, and produce a single PDF containing all the students' scans. (I add that in my limited experience, the machine swallowed my 40 answer sheets in seconds without creating any rotation.) Below are some notes on what I have learned when using our photocopy machine for this purpose.

- Place answering sheets on the top feeder.
- Insert your USB drive on the machine.
- After logging in, in the screen options, select **scan**, then scan into **Memory media**, then select your drive.
- I could not find an option that is a single-sided scan; therefore, to avoid ending up with a PDF file that has a blank page every other page, choose **Options** (lower right corner of the screen) and then **Skip Blank originals**.
- Start the scan and wait for the screen to show that the recording to the memory stick is completed. Safely remove the USB stick from the machine by following the on-screen instructions.

8 Evaluation of scanned answer sheets

8.1 Loading and checking the scans in R

There is a dedicated function to read the PDF containing all the scans and report any errors.

```
nops_scan(dir="scans")
```

The argument *dir* specifies the folder name (*scans* here) in the R working directory where the PDF file containing all the answer sheets was saved.

Next, we evaluate the scans, but before that we need to create a registration file.

8.2 Creating a registration file

The registration file enables the package to reconcile the registration numbers that students entered on the answer sheets with their names. This should be a file with fields separated by semi-colons (;). The registration file should have the format shown below, when viewed in a text editor, showing only the first three students.

registration;name;id
1234567;John Mitchel;John_Mitchel
2345678;Jane Miller;Jane_Miller
3456789;Maria Baptista;Maria_Baptista

The *registration* field should have the last seven digits of the student's Panther ID. This ID is shown in PAWS, from where you can export the roster table (in the upper-right corner of the Paws roster table). You can either edit this file from what you get from PAWS or you can use the *createRegister* function in R to convert from this format to the *exams* registration format. We will look at the latter option next.

8.2.1 Converting a PAWS roster table to a registration file

The function *createRegister* can be used to create the registration file from a PAWS roster table download. A few notes below.

- First, the function is not part of the *exams* package. Therefore, to make it available in R, after you download the materials from GitHub, open the script containg the function *createRegister.R* in RStudio. Select all code and run (Control+Enter or the Run icon on the top right corner on the top bar of RStudio script window).
- Second, the file downloaded from PAWS has an odd format once open in a text editor or Excel. Each student takes two rows of the table, and there are merged cells (in Excel). Open the file in Excel to check if this is the case. The downloaded file from Canvas should look like below (only three imaginary students shown), showing two rows used by student. The conversion function will read the file in the odd format.
- If you have a U/G class don't forget to download the roster tables for both classes and add them into a single file in Excel.

```
Notify,ID,Name,Campus ID,Audit,Units,Program and Plan,Level,Status Note,Phone
,1740226,"Mitchel, John",99-123-4567,,3,Letters & Science Undergrad -,Senior,,414/999-9999
,,,,,,Biological Sci BS/Pre Veterinary Med,,,
,1544567,"Miller,Jane",99-234-5678,,3,Letters & Science Undergrad -,Junior,,414/414-7777
,,,,,,Biological Sci BS/Neuroscience BS/Pre Medicine,,,
,1249531,"Baptista, Maria",99-345-6789,,3,Letters & Science Undergrad -,Junior,,262/888-3194
,,,,,,Biological Sci BS,,,
```

- From Excel save the file as a comma-separated file (*.csv*). In the function *createRegister* run example below, we named this file *Canvas Download.csv*. The function *createRegister* will take care of converting the format to what is required by *nops_eval* function.
- Below, an example of using the function to convert between formats. The only two parameters are the file name downloaded from Canvas and the output file name. The output file is writen to the working directory, ready to be used by *nops_eval* in the next step .

```
createRegister(PawsF="Paws Download.csv", resgisterOut="Exam-2025-10-23.csv")
```

If you **had to provide a *ad hoc* ID number to a student who did not have their number during the exam**, you will need to edit this file with that number for the scan to work correctly.

8.3 Exam evaluation

We are now ready to use the function *nops_eval* to evaluate the exams and produce the output files.

```
ev1 <- nops_eval(
    register = "Exam-2025-10-23.csv",
    solutions = "Exam_I.rds",
    scans = Sys.glob("scans/nops_scan_20251013182655.zip"),
    eval = exams_eval(partial = TRUE, negative = FALSE,rule="false"),
    interactive = FALSE)
```

- In the code above, I assume the registration file was named *Exam-2025-10-23.csv* and is saved in the R session working directory. This file name is given to the function argument *register*.
- The *solutions* argument is an *.rds* file created above by the *exams2nops* function when creating the exam.
- The *scans* argument specifies the path to the *zip* file in the same folder as the PDF files with your scans, here named *scans*.

- The *eval* argument defines if multiple choice questions (one or more correct answers) are to be evaluated as "all or nothing", where all correct answers and no incorrect answer need to be selected to get full points, or partial credit can be given. You can read more about partial credit for multiple choice in the *exams_eval* function's manual. Above, I used partial points with a rule that penalizes conservatively wrong choices. I have yet to explore the options here fully, but there are different rules you can apply if you select partial credit. If you import questions to Canvas (a topic not covered here), partial credit is always given following Canvas rules. You can learn more by reading the function's manual using the command below.

```
# Opens the function manual page in RStudio's lower right window
?exams_eval
```

This function might detect errors and ask you to correct them.

8.4 Correcting errors in the scans

The *nops_eval* function might report an error that can be fixed using the *nops_fix* function. The error will report the exam number(s) where the error requiring your attention was found. The numbers correspond to the page order in the PDF file containing the scanned answer sheets for all students (saved in the *scans* directory). Essentially, *nops_fix* launches an interactive web page where you can see the exam scan and mark yourself the boxes where there were doubts or absent data, by using your mouse to click through a digital version of the answering sheet boxes. Hopefully, everything scans correctly and you don't have to corrcet any answer sheets. Below is an example of how to use the function. It requires the name of a *.zip* file that is created by the *nop_scan* function and written to the same directory where the scans were recorded.

```
nops_fix("scans/nops_scan_20251012115937.zip"),exam=34,field="answers",display="interactive"))
```

The code above will open an interactive page showing, on the left side, the scan of exam number 34, and, on the right side, an interactive representation of all the fields and information detected by the scan (Figure 2). You can correct the "Registration number" by typing or editing directly in the box that shows the scanned number. For the answers, you can click or unclick the selections shown to make your corrections.

When you are done with your corrections on the interactive browser page, return to RStudio. Usually, in RStudio, you only need to hit return after selecting the R console (the lower-left box) once you're there. However, in my system, I got a message saying that the clipboard could not be copied. The solution was easy: press Ctrl+V (paste) directly into the RStudio console. A lot of code is pasted from the browser output into the R console, then hit return, and that's it.

8.5 Reviewing the evaluation results

Once all errors are corrected, *nops_eval* will run to completion producing two output files. First, a *nops_eval.zip* file that, once uncompressed, contains one subfolder per student name. Inside these folders, you will find an HTML file containing a summary table showing which answers the student got right or wrong (Figure 3).

Review all these HTML files to see whether the scan detected more than one box marked for a single-choice question. The scan is very sensitive to any markings on the boxes, even if they are just part of a single line. Because it takes time to go through all these HTML files, the code below will convert the individual HTML files into a single PDF for all the students to facilitate their inspection. Note the use of paths to locations in your computer that will need to be edited. Also, paths in Windows use a backward slash " \ ", while in macOS and Linux a forward slash" / " is used (like shown below).

```
install.packages("psycModel") # only need to do this once
library(psycModel)

#Below, use full path to your nops_eval folder
files<-list.files("/home/filipe/Documents/jobs/UW-Milwaukee/Teaching/Rexams/Exam.I.F.25/nops_eval/")
```

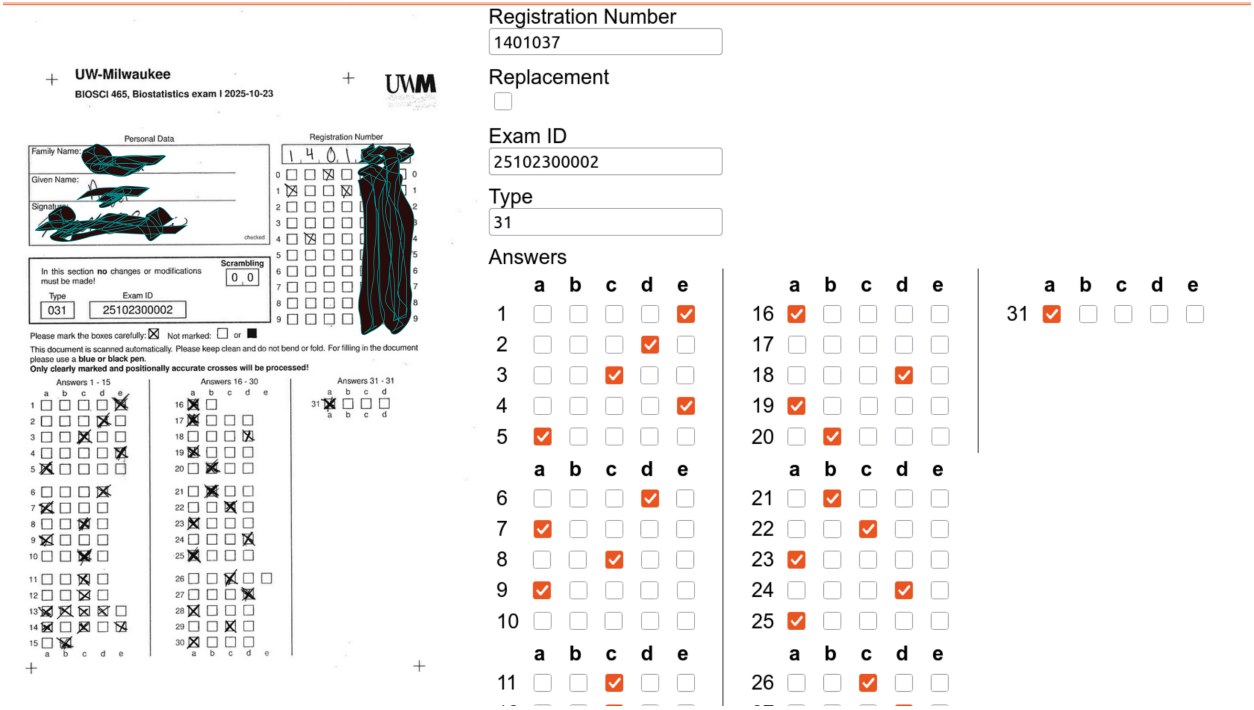
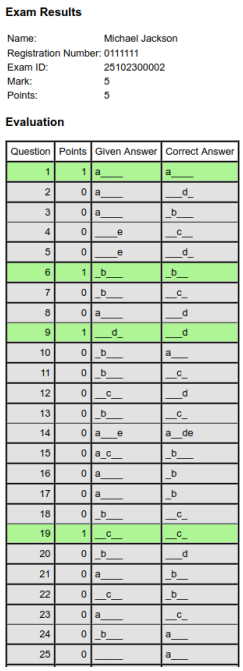


Figure 2: Example of the interactive webpage created by the function *nops_fix*. The student information was redacted for confidentiality. Note how this student made some rough crosses that went beyond the box bounds, resulting in some answers for which no answer was detected (questions 10 and 17). The user can correct this by clicking the boxes the student marked on his answer sheet, conveniently shown on the left side.

```
#Below, match the name of the registration file but replace the csv by html extension
Cstring<-"registration.html"
#likewise but with pdf extension
CstringPDF<-"registration.pdf"
nstudents<-length(files)

for(f in 1:nstudents) {
    html_to_pdf(file_path  = paste0("./nops_eval/",files[f],"/",Cstring))

    system2("cp",args=c( paste0("./nops_eval/",files[f],"/",CstringPDF),
                         paste0("./nops_eval/",files[f],".pdf")  )
                    )
print(paste0("Completed conversion from html to pdf of ",files[f]," evaluation") )
}

#merge all pdfs into a single file
install.packages("qpdf") # only need to do this once
library(qpdf)

# CHANGE the R directory temporally to the nops_eval folder
# In RStudio's lower right box click on the Files tab to navigate to nops_eval
# click on the blue gear-like icon on that box's top menu and select "Set as working directory"

PDFfiles<-list.files(pattern=".pdf")
qpdf::pdf_combine(input=PDFfiles, output = "merged evaluation forms.pdf")
```

```
#Don't forget to change back to the original R session working directory
```

A file named *merged evaluation forms.pdf*, containg the pdfs for all the exams, should now be written in the *nops_eval* folder.



Exam Results

Name: Michael Jackson
Registration Number: 0111111
Exam ID: 25102300002
Mark: 5
Points: 5

Evaluation

Question	Points	Given Answer	Correct Answer
1	1	a____	a____
2	0	a____	___d_
3	0	a____	_b___
4	0	___e	__c__
5	0	___e	___d_
6	1	_b___	_b___
7	0	_b___	__c_
8	0	a____	___d
9	1	___d_	___d
10	0	_b___	a___
11	0	_b___	__c_
12	0	__c__	___d
13	0	_b___	__c_
14	0	a___e	a__de
15	0	a_c__	_b___
16	0	a____	_b
17	0	a____	_b
18	0	_b___	__c_
19	1	__c__	__c_
20	0	_b___	___d
21	0	a____	_b__
22	0	__c__	_b__
23	0	a____	__c_
24	0	_b___	a___
25	0	____	a___

Figure 3: Top part of the nops_eval HTML output for a given student. A second part of this HTML file (not shown here) contains the scan of the answer sheet.

If upon reviewing the pdf you find problems with the scans go back to *nops_fix* instructions to fix the problem.

The second file produced by the *nops_eval* function is a text file named *nops_eval.csv* containing the evaluation data for all students. Although this file contains the answers and key data for all questions and students, the format is in a sequence of zeros and ones. You should be careful when opening it in Excel so that leading zeros don't disappear. Overall, this file is only useful as raw evaluation data from which to run other descriptive statistics (see below).

9 Generating descriptive statistics for your exam

Here, you will run an R Markdown script (*markdown for evaluation.Rmd*) directly in RStudio. This script will load *nops_eval.csv* and *Exam_I.rds* files and create a series of diagnostic tables and plots from these data. After initial simulations, this script was tested with my Exam I from Biostats 465 (Fall 2025) using two exam versions (about 20 students per version). It included 31 questions, with 29 single- and two multiple-choice.

First, this script requires a few R packages that you might not have installed yet. During the installation of package *kableExtra* you might get a message asking if you want to install packages that require compilation; say yes.

```
# remember that you only need to install packages once
install.packages(c("DT","kableExtra"))
```

We are now ready to run the script.

- Save the RMarkdown script *markdown for evaluation.Rmd* in the R session working directory that you have been working from.
- Start by opening the script *markdown for evaluation.Rmd* in RStudio and change the names in the *r setup* chunk to match your file names. This section is found at the top of the script. You want to change the *"nops_eval.csv"* and the *"Exam_I.rds"* to match the names you used above.
- If you have to correct an answering key from your exam, see advanced troubleshooting below and rename the *rds* file in the process (recommended), don't forget to replace the new name in here (you probably ran this RMarkdown script already once before with the original *Rds* file and then looking at the stats discovered the problem with the key).

```
{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
nops_eval.file<-"nops_eval.csv"
rds.file<-"Exam_I.rds"
```

- Next, look for the *knit* icon on the RStudio top bar. Use the down-pointing arrow in the icon to make sure it is selected to *knit to HTML*. Then press the knit button. An RStudio webpage preview window will open if there are no errors, showing a preview of the HTML with all the diagnostics. At the top of the window, press *Open in Browser*. An HTML file and an associated folder are recorded in the R session working directory. Running this script will also write a text file named *Results table.txt* to the working directory. The file contains the points per student, total percentages, and registration ID (the first table in the HTML file). This file is handy to prepare your Canvas upload.

10 Uploading exam grades to Canvas

A function, named *import2Canvas*, can assist with this annoying task.

- After download from GitHub, open the function script in RStudio (file *import2Canvas.R*), select all and run it (Control+Enter or the the Run icon on the top right corner of RStudio's script window).

Below an example of using the function, with more details about its arguments below the R code.

```
import2Canvas(PawsF="Paws Download.csv",
              EvalResults="Results table.txt",
              CanvasDownload="2025-11-03T1154_Grades-BIO_SCI_465_examI.csv",
        outputF="grades to upload.csv")
```

- The function requires only the names of three input and one output files. The first argument *PawsF*, takes the Paws roster file that was used above to create the registration file. This is the odd format file with two rows per student. Use the same exact file as above. **Do not forget to merge undergrad and graduate student roster files** from the PAWS downloads for each class roster.
- The second argument *EvalResults* takes the file that is produced by running the RMarkdown diagnostic script. Once you run that script, this file should be found in the folder where the RMarkdown script is saved (often the same as the R session working directory).
- The *CanvasDownload* argument takes the name of the file exported from the Canvas grade book. For simplicity, in Canvas, select only the grade item that corresponds to this exam. Then do "export current gradebook view". Open the exported file and select the table that contains all the left columns until, and including, the exam item. Select and copy to a new file and save as a comma-separated file.
- The last argument, *outputF* can be used to modify the name of the exported file. In this example, the function will write a file named *"grades to upload.csv"* to the working directory.

- Once you run the function **you need to replace the header of the output file by the original header** in the file exported from Canvas. This is because R modifies the header names that include spaces. Do this in Excel and then save, keeping the *.csv* format.
- Inspect the output file grades showing as NA. You might have their values (e.g., students that took the exam in a different date) that should be edited here. Replace any other NAs with the value zero.

The file should be ready to upload now. Canvas will recognize the exam name and ask you to confirm. You can use this to upload an item that is still not part of your grade book, by naming the last column with a new name not used in your graded assignments.

11 Troubleshooting

11.1 Detecting a problem with the key of one question

I wish I had never had to correct my exam evaluations after looking at the results from the exam center. Unfortunately, I have found errors with my correct answer keys more than once. Using the exams R package and upon visualizing the descriptive statistics generated in the previous point, you might find problems with the correct answer key for one or more questions. The last table in the HTML, under *Test report by exam type*, is a good place to look for problems, as wrong (in your key) answers predominantly selected by students as correct are highlighted in red with a yellow background. The code and steps below are intended to correct the answer key.

The correct answer key is represented internally in R exams as a binary variable made of TRUE or FALSE values.

Create a small R script and run the code below:

First, read the *Rds* file.

```
rdsExamdata<-readRDS("Exam_I.rds")
```

You probably know from the diagnostics which question had the wrong key. You access the correct answer key with the following code:

```
rdsExamdata[[1]][[14]]$metainfo$solution
```

In the code above the first [[1]] is accessing the first version of the exam (here, *Exam_I.1*) while the second [[14]] is accessing the data for question 14. The specific information for the correct key is recorded in the metadata/solution slot.

In my example, the code above produced the following result:

```
[1] TRUE,FALSE,FALSE,FALSE,FALSE
```

Let's say the last option was actually the correct one for this single-choice question. You could correct the key with the following code:

```
rdsExamdata[[1]][[14]]$metainfo$solution<-c(FALSE,FALSE,FALSE,FALSE,TRUE)
```

The correction is now saved in the R object *rdsExamdata*. We need to export it to the working directory so that we can repeat the exam evaluation. **Don't forget to rename the Rds file**.

```
saveRDS(object=rdsExamdata,file="corrected_Exam_I.rds")
```

Next, we need to repeat the scan evaluation. But before that, **delete or move from the working directory** the *nops_eval.zip*, the folder *nops_eval* you extracted from it, and the *nops_eval.csv* file that contains the raw results table. I found that if these files are present, rerunning the function will not update the results, without any warning to the user. Do not forget to **update the name of the corrected *Rds* file** in the ocde below.

```
ev1 <- nops_eval(
    register = "Exam-2025-10-23.csv",
    solutions = "corrected_Exam_I.rds", #change this to corrected file
    scans = Sys.glob("scans/nops_scan_20251013182655.zip"),
    eval = exams_eval(partial = FALSE, negative = FALSE),
    interactive = FALSE
)
```

The function will run fast. You don't have to redo any of the scan corrections you might have previously made, because all of that is saved in the original *.Rds* file, and all we did was change a single field in the first version of the exam for question 14's solution key.

You can now rerun the R markdown file to get updated descriptive statistics. **Do not forget to update** the *Rds* input file at the top of the R Markdown script before you rerun the descriptive statistics with the question key corrected.

Finally, **update the key in the original *Rds* file for that question** (see above in Creating individual questions), or you will have the same problem the next time you use this question to create an exam.

11.2 Adding students that completed the exam on a different date

More often than not, one or more students will need to complete the exam after the schedulled date. Often I grade these exams "manually" and post the final grade directly on Canvas. The code below is useful to add the evaluation of these late exams to the *nops_eval* file so that you can include the new data in the exam descriptive statistics output.

You could scan the answer sheet of late exams and then add the new scanned answer sheets to the PDF document for the rest of the class. However, you would need to redo all the corrections that you might have done previously. A more efficient way is to score "manually" the late exams into an excel file and use the function *Add2nops_eval.R* to create an updated (and renamed) *nops_eval.csv* file. Then run the descriptive R markdown script file again. Note that the function *Add2nops_eval.R* is not part of the *exams* R package. To make it available in your system open the function file, select all and run it in R.

- First, you need to create an input file with the evaluation of the late exams. Prepare a file with the format below in Excel:

```
registration    name    exam    scrambling  Q.1 Q.2 Q.3 Q.4 Q.5 Q.6 Q.7 Q.8 Q.9 Q.10
7676551 John Sick   25102300001 "00"    a   b   c   b   c   c   b   d   a   ae
9876543 Mary Flu    25102300002 "00"    a   b   b   b   d   c   d   b   a   ad
```

- The fields include registration, name, exam, scrambling (I don't know what that is, just use "00"), and questions. In the name field, use one or more spaces (NOT a tab) to separate given and family name. For questions, fill with low case letters with the answer students selected. Note that for multiple choice answers (last question, Q.10) the letters corresponding to the students' answers are concatenated. Save this file as a tab delimited text file. Alternatively, if creating this table in Excel, select the table and copy it to a text editor, for example by creating a new text file in RStudio, and save it there. Cell separation in Excel is interpreted as tab separation between values in a text file.
- In the function run example below, we named the late exam evaluation file as *LateExam.txt*.

```
Add2nops_eval(eval.F="nops_eval.csv",
                            late.eval.F="LateExam.txt",
                            output="nops_eval_updated.csv",
                            rds.file="CorrectedExamI.Rds",
                            mark= c(0,0.5, 0.6, 0.75, 0.85,1),
                            partial=TRUE,negative=FALSE,rule="false")
)
```

There are a few arguments to the *Add2nops_eval* function:

- The *eval.F* takes the name of the *nops_eval.csv* file with the raw evaluation for all students.
- The *late.eval.F* takes the name of the tab-delimited text file (*LateExam.txt*) with the "manual" evaluation of late exams (see above in this section).
- The *output* is the name of the output file containing the merged *nops_eval.csv* and late exam data. This dataframe is returned by the function and also writen to the working directory so that you can rerun the descriptive statistics script.
- The *rds.file* takes the name of the *Rds* file containing the metadata used for this exam.
- The *mark* contains a vector of thresholds to convert from points to a mark ranging from 1 (best) to 5 (worst). I kept the same thresholds as used by the exam evaluation function. If you change this argument when evaluating your exams don't forget to change here. I don't use these marks to grade but had to include this since it is a column in the *nops_eval* file.
- The arguments, *partial*, *negative*, and *rule* need to match the values used by function *nops_eval* above, so that multiple choice questions in late and original exams are evaluated with the same rules.