# A Simple Classroom Management Addin for *RStudio Server Professional*

## 1. OVERVIEW

Both authors have taught courses at Bentley University in which they require students to do work in *R* (R Core Team 2016) using *RStudio* (RStudio Team 2015) and *RMarkdown* (Allaire, Cheng, Xie, McPherson, Chang, Allen, Wickham, Atkins, and Hyndman 2016). For Dr. Oury, this has included graduate-level courses in data mining, data science, and machine learning. For Dr. Carter, it includes a single undergraduate course in discrete mathematics. In these courses, many of the homework assignments, handouts, slides, and projects are given to students in *RMarkdown* format. Students are expected to submit work in the same format. This paper introduces a new *RStudio* addin, called "Simple Course Management System," that facilitates the exchange, grading, and real-time supervision of student work in *RStudio*. It also discusses the results of using the addin in a course in Fall 2016, and suggests future extensions.

We assume readers are familiar with each of these tools (*R*, *RStudio*, *RMarkdown*, and *RStudio Server Professional*), but any who are not can learn about their value in the classroom from previous work, such as Baumer, Cetinkaya-Rundel, Bray, Loi, and Horton (2014) and Cetinkaya-Rundel and Rundel (2017). Readers who will find the addin of benefit to their own teaching will find installation instructions in the addin's GitHub repository (Carter 2017). The source code is released under the LGPL version 3. Readers who wish to participate in the development of the addin are welcome to submit pull requests to that same repository.

## 2. CLASSROOM MANAGEMENT FEATURES

Most readers will be familiar with *R*, the programming language for statistical computing, and *RStudio Desktop*, an application for interacting with *R*. *RStudio Server Professional* is a cloud-based version of *RStudio* whose interface is identical to that of *RStudio Desktop*, but it is accessed through the browser, and works with files stored on the same server. The authors' institution ran a server, administered by Dr. Oury, serving *RStudio Server Professional* to Bentley University students in Dr. Carter's discrete mathematics course in Fall 2016. Doing so enabled us to develop an addin with file sharing and live monitoring features, as described in the remainder of this section.

### 2.1. Goals

Prior to the Fall 2016 semester, the primary way Bentley University students interacted with *R* and *RStudio* was through *RStudio Desktop* on their own laptops. The majority of students who learned *R* were graduate students who would soon become business analytics professionals and should have their own copy of *RStudio*. In Fall 2016, Dr. Carter chose *R* as the software

for use in a discrete mathematics course and chose to use *RStudio Server Professional* in the cloud, because doing so brought several benefits. The aim was to make it easy for

1. Students to submit completed assignments to the instructor;
2. Students to share work with teammates;
3. The instructor to distribute course materials to students;
4. The instructor to collect completed assignments from the students;
5. The instructor to return graded assignments to the students; and
6. The instructor to monitor student work on in-class exercises in real time.

Section 2.3 explains how we achieved these goals by developing an addin for *RStudio Server Professional*. In the following section, we review why it was necessary to do so.

### 2.2. Possible Alternative Solutions

Readers may be familiar with other cloud-based options for sharing *R* code, such as GitHub and *RStudio Cloud* (RStudio Team 2018b). In this section, we explain why these services would not meet our needs.

GitHub is a website hosting millions of code repositories and allowing project contributors to make and coordinate changes to these repositories. *RStudio Server Professional* and *RStudio Desktop* both allow projects to be created as GitHub repositories, allowing multiple users to contribute to the same project. Dr. Oury, when working with graduate analytics students, has had success using *RStudio*'s built-in `git` support to share projects across teams using GitHub.

The reason Dr. Carter chose not to use GitHub in his course is that the `git` version control system underlying it is an additional piece of nontrivial software students would need to learn. The course already required students to learn three languages, *R*, *RMarkdown*, and a subset of LaTeX. Requiring them to also learn all the concepts associated with version control and then the specifics of `git` itself would have been too much time spent on computing rather than mathematics. Furthermore, choosing GitHub would not have supported the live monitoring of in-class work supported by the addin presented in this paper.

The *RStudio Cloud* website is a new offering by the *RStudio* company and is specifically geared towards the teaching, learning, and sharing of work done in *R*. It was not in existence in Fall 2016, but even if it had been, it would not have been compatible with the addin discussed herein. Each project in *RStudio Cloud* is its own container, and one container cannot read the contents of another. This prevents the sending, receiving, and monitoring of files across users' projects, which are the core features of the addin presented in this paper.

The *RStudio Cloud* site is still in its alpha phase as of this writing, and if later changes make it possible for one project to read or write files in another, that may make it possible to extend this paper's addin to support *RStudio Cloud*.

### 2.3. Implementation Overview

We used *RStudio Server Professional* to accomplish the goals stated in Section 2.1. We omit here the instructions for installing *RStudio Server Professional* and setting up user accounts, but interested readers can find the details in the technical appendix, Section 7.

The built-in file sharing features of *RStudio Server Professional* accomplish one of our goals right out of the box, sharing files among a team. On the first day, students logged in to the server and the instructor placed them in teams. One designated member from each team created a project and shared it with his or her team members. Those members could then place files in the shared project, copy files from it, or edit shared files simultaneously.

This feature, unfortunately, had some unexpected behavior. Frequently, students who had opened a shared file and then saved a copy in their personal folder found that they were still editing the shared version, thus corrupting others' work. It is not clear whether this behavior is a bug, as it can be challenging to reproduce behaviors reliably in a complex, multi-user environment. But we created a workaround, shown at the end of Section 5.2, and relied on the shared folders support built into *RStudio Server Professional* via that workaround.

Accomplishing the other goals listed in Section 2.1 required the development of the addin that is the subject of this paper. Dr. Carter created an *RStudio* addin that an instructor can install into his or her *RStudio Server Professional* account, and it provides a graphical interface that allows the instructor to

1. Distribute a file to all students in the course;
2. Collect a completed assignment from all students in the course;
3. Return a graded assignment to all students in the course; and
4. Monitor all copies of a file being worked on by all students in the course.

Students do not need to install anything; only the instructor needs to install the addin.

The following section covers details about how the addin works. Sections 4 and 5 cover the students' and instructor's experiences, and Section 6 covers future extensions that could be implemented.

## 3. INTERFACE

This section describes the new functionality and the specific benefits the addin provides. Instructor and student workflows are covered briefly for all relevant tasks. Because it can be challenging to visualize a user interface based only on a text description, we also provide screencasts that demonstrate each workflow. See the addin's GitHub repository for these screencasts (Carter 2017).

As mentioned above, each student and the instructor has an account on the server. Although users log in to *RStudio Server Professional*'s web interface, each account is also an ordinary

File sharing ⟶ menu

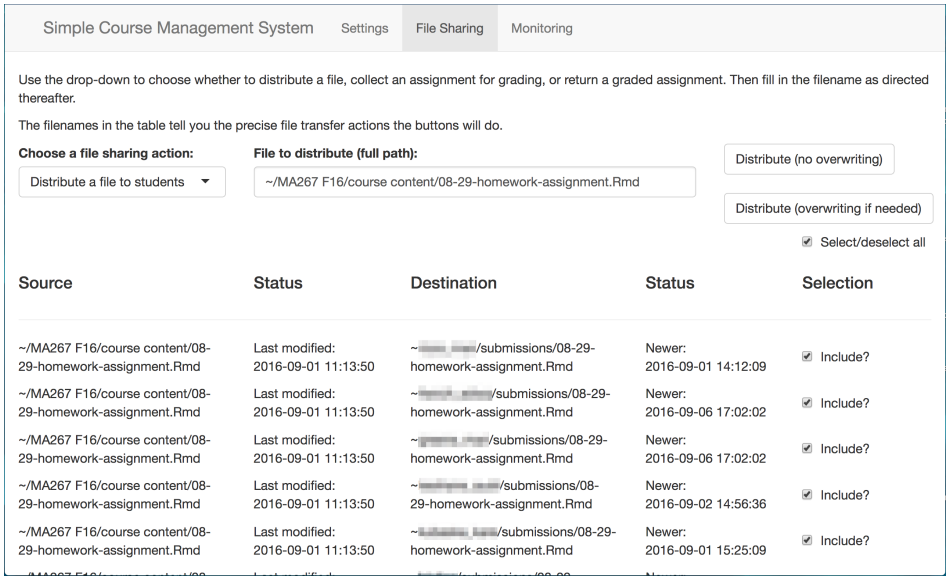


Figure 1: Screenshot of the file distribution tool in the Simple Course Management System addin, with student usernames obscured for privacy

Linux account with a home folder that stores the user's files. By default, no one can read or write to such a folder except its owner. Students keep their coursework in their home folder, which is where *RStudio Server Professional* points its file browser upon login.

The instructor can keep course assets in his or her home folder, or a subfolder thereof. *RStudio Server Professional* lets an instructor upload a zip file containing an entire folder hierarchy of course assets, and it automatically unzips it on the server. Such course assets are not visible to students until the instructor intentionally shares them, as described below.

To facilitate sharing files across these home folders, each student was instructed, on the first day of class, to create a new project named "submissions," as a subfolder of the student's home folder, and to share that project with the instructor. All files students wished to submit to the instructor could be placed in that folder, hence the name. The instructor would also use the same folder for distributing files to students, despite the name.

To automate the file distribution and collection processes, the Simple Course Management System addin has a file sharing feature shown in Figure 1. The feature has three purposes: distributing files, collecting files for grading, and returning graded files. We cover each separately below, plus a separate feature for use during live, in-class activities. (These assume the instructor has gone through some necessary configuration steps, which we will cover in Section 3.6.)

### 3.1. Distributing Files to Students

Readers who wish to watch a video of the content of this section should see the screencast entitled "Distributing a file to students" on the addin's GitHub repository (Carter 2017). To distribute a file to students (such as a handout or homework assignment), the instructor clicks the file sharing menu shown in Figure 1 and chooses "Distribute a file to students."

The instructor then enters the absolute path to the file in the adjacent text field, as shown in that same figure. The interface automatically updates to show, for each student in the course, where the file will be placed, whether a file of the same name already exists there, and if so whether it is newer or older than the file to be copied. By default, all students are selected to receive the file, using the checkboxes in the rightmost column, but the instructor can change this before performing the operation.

After selecting a set of students, the instructor can click either the "Distribute (no overwriting)" or "Distribute (overwriting if needed)" buttons to take the action. The interface automatically updates to show the results (with updated file modification dates) after the operation is complete.

### 3.2. Collecting Student Work

As with the previous section, readers can see a video covering this section's content on the addin's GitHub repository; see the screencast entitled "Collecting student work" (Carter 2017). This workflow assumes that each student has a completed assignment in his or her submissions folder, each with the same filename. This is easy to ensure by having the instructor distribute the assignment using the workflow in Section 3.1; students simply edit the assignment file in place and never need to move or rename it.

The instructor chooses "Collect an assignment" from the same file sharing menu shown in Figure 1, and enters the name of the file to collect (excluding the path) in the adjacent text field. There are no restrictions on file type. The bottom half of the interface updates to show whether the file exists in each student's submissions folder, and if so, the last modification date. It also shows where the file would be moved if the instructor were to collect the assignment now. Each student's username will be added to the filename to prevent collisions and ensure that the instructor knows the author of each collected file.

The same interface of checkboxes lets the instructor choose any subset of students (by default, all) and the buttons "Collect (no overwriting)" and "Collect (overwriting if needed)" let the instructor take the action, at which point the file list updates accordingly.

### 3.3. Providing Feedback

While the processes in Sections 3.1 and 3.2 are not limited to any particular file type, Dr. Carter used them primarily for distributing and collecting *RMarkdown* files. In such files, *RStudio* facilitates giving consistent and prominent grading feedback in two ways. First, *RMarkdown* supports HTML, so comments inserted by the instructor can be styled with colored fonts and backgrounds to make them stand out amidst the other contents of the file, as shown on the right of Figure 2. Students reviewing a graded assignment for instructor

```
snippet CC
  <span style='color:#00aa00;background:#aaffaa;'>
  <b>Correct.</b>
  ${1:explanation}</span>

snippet II
  <span style='color:#aa0000;background:#ffaaaa;'>
  <b>Incorrect.</b>
  (${1:deduction}) ${2:explanation}</span>

snippet PP
  <span style='color:#888800;background:#ffffaa;'>
  <b>Partially correct.</b>
  (${1:deduction}) ${2:explanation}</span>

snippet GG
  <span style='color:#00aa00;background:#aaffaa;font-size:200%;'>
  <b>Grade:</b>
  ${1:earned}/${2:possible}</span>
```

## Hypothetical Student Work

**Grade: 7/10**

1. Write code that generates a sequence of 5 random values from the uniform distribution on [0, 1], assigns the result to a variable, and displays it.

```
mySample <- rep( runif( 1 ), 5 )
mySample
```

```
## [1] 0.4475934 0.4475934 0.4475934 0.4475934 0.4475934
```

**Partially correct.** (-3) You created a random value and repeated it five times, but it's the same random value. Try `runif(5)` instead.

2. Write code that takes the mean of the values generated in the previous question.

```
mean( mySample )
```

```
## [1] 0.4475934
```

**Correct.**

Figure 2: On the left are snippets stored in Dr. Carter's *RStudio Server Professional* settings, used for quickly inserting HTML-formatted feedback into students' homework assignments. On the right are example rendered uses of those snippets in a graded homework assignment. Instructors can quickly insert any of the snippets by typing its name (CC, II, PP, or GG) followed by Shift-Tab.

feedback can easily skim to find instructor feedback because it is visually obvious, the digital analog of a red pen.

Second, *RStudio* has a feature called "Snippets" that makes it easy to insert consistently formatted HTML feedback in students' documents. In the Global Options menu, users can enable code snippets for *RMarkdown* and then create their own text shortcuts that expand into arbitrarily complex HTML feedback, allowing each instructor to design how their grading feedback appears, and to consistently insert that feedback with very few keystrokes while grading. Example snippets used by Dr. Carter appear in Figure 2.

When creating assignments—whether or not they are *RMarkdown* files—it is good practice to create corresponding exercise solutions, to ensure that each exercise is as doable as the instructor expects. When the assignment files are created digitally, as they are in this case, converting the solutions document into the assignment is then simply a matter of one minute's work, deleting the solutions and saving a second copy of the file. After the due date, solution files are then already available to distribute using the means described in Section 3.1, which prevents the need to accept late assignments. Instructors who adopt this practice can also annotate the solutions while grading, to include notes to the class about mistakes common to many students, and important concepts to review before an exam.

Instructors who grade student work this way would either need to simultaneously record the students' grades in a separate gradebook, or automate the extraction of grades from the *RMarkdown* files themselves. While this feature has not yet been implemented in the addin, it can be accomplished with a small amount of *R* code, as discussed in Section 6.

### 3.4. Returning Graded Files to Students

After the instructor has opened each student's file and added feedback, the addin provides a feature for returning the files to the students so that they can see the results of grading. The instructor chooses "Return a graded assignment" from the file sharing menu in Figure 1 and enters the filename. The files formerly collected are returned to their original locations, but with the word "graded" appended to the filename, to distinguish them from the originals. The students can review the instructors' comments at their convenience.

As with the workflows in earlier sections, this one can also be seen in a video on the addin's GitHub repository (Carter 2017).

### 3.5. Live Monitoring of In-Class Work

To create a classroom in which active learning takes place, Dr. Carter often provided his students with worksheets to do and discuss in class, on teams, with instructor supervision. These worksheets were *RMarkdown* files, sometimes with code to be written or analyzed, and sometimes with mathematical ideas to be explored. Students were asked to record their answers in the file.

While students do such worksheets in class, it is helpful for the instructor to be able to monitor their progress, and pause the class occasionally to discuss results together. If one team is getting ahead, it may be useful to have them help a slower team catch up. If all teams are stuck simultaneously, the instructor should notice and address the confusion for the class as a whole. To facilitate this, the addin provides a live worksheet monitoring tool, shown in Figure 3.

While editing his or her *RMarkdown* worksheet, each student periodically uses the "knit" button in *RStudio* to typeset the file and see the results of computations. This produces an HTML file with the same base filename as the original *RMarkdown* file and in the same folder. The addin can monitor changes in such HTML files across all student and team folders. As it does so, it reports results with the interface shown in Figure 3, which updates itself live as students knit their files. This feature is demonstrated in the video entitled "Live monitoring of in-class work" in the addin's GitHub repository (Carter 2017).

The check boxes shown in that figure serve two purposes. First, they indicate student progress. A checkbox is present only if the student's file differs from the instructor's original file in that section, and thus the student has done some work in that section. Thus with a glance at the checkboxes, the instructor can compare progress across all teams in the course.

Second, the instructor can check the box to have that section's content displayed below the grid of checkboxes. Any subset of the boxes may be checked or unchecked, as the instructor chooses. The instructor can show this page on a projector and scroll down to compare students' ideas, demonstrate right answers, or address common mistakes.

This makes sharing student work on the projector trivial, but it is limited to *RMarkdown* files that produce HTML output. Most do, but not all; see Section 6 for details.

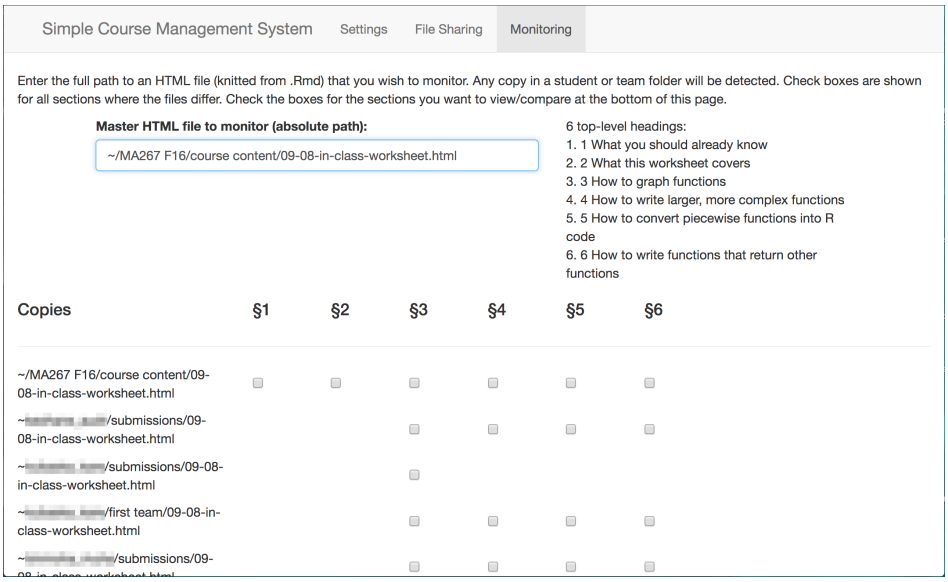


Figure 3: Screenshot of the live monitoring tool for in-class work, with student usernames obscured for privacy

### 3.6. Addin Settings

The final portion of the addin's interface is for specifying settings, as shown in Figure 4. The instructor maintains a list of usernames for all students in the course on the left, and the relevant instructor, student, and team folder names in the middle. Any LaTeX-style definitions that need to be present when typesetting live student work can be listed on the right.

Any change to these settings takes immediate effect and is also immediately saved in a hidden file in the instructor's home folder. The settings are reloaded from that file each time the instructor launches the addin.

## 4. STUDENT RESPONSES

Student opinions expressed in this section come from formal student evaluations of the discrete mathematics course and from informal interactions that Dr. Carter had with students during the Fall 2016 semester.

Students found the behavior of *RStudio Server Professional*'s collaborative editing features to be unpredictable. *RStudio Server Professional* supports multiple users editing the same document at once, via multiple cursors, but this often behaved in ways users did not expect, and they ended up editing files they did not intend to edit, or files belonging to other users. We therefore minimized the need to use this feature in our course. Related challenges with copying shared files were mentioned in Section 2.3, with workarounds given in Section 5.2.

Students were patient with occasional server outages, though their evaluations of the course

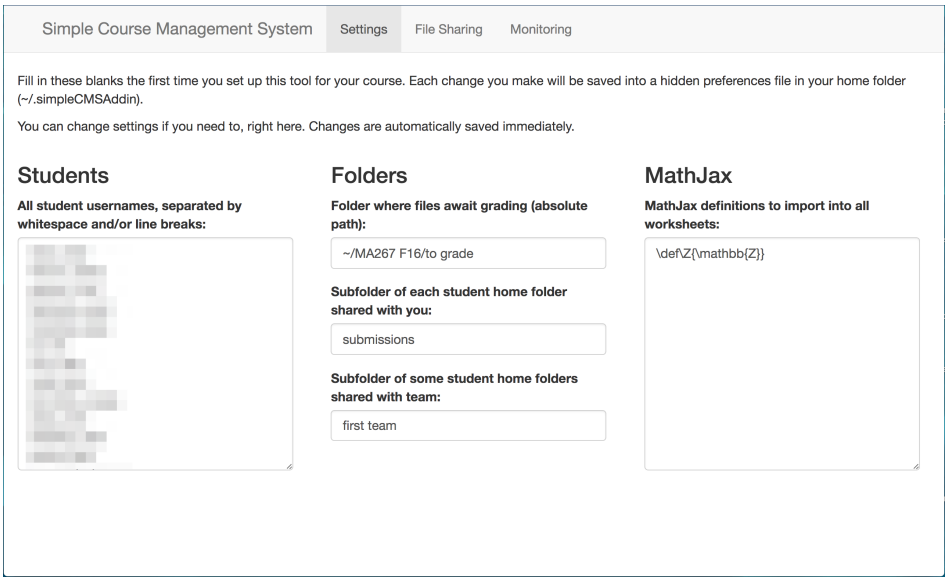


Figure 4: Screenshot of the settings tab for the Simple Course Management System addin, with student usernames obscured for privacy

did mention them as a challenge. *RStudio* technical support worked with us to resolve these issues, as mentioned at the end of Section 7. Students were granted deadline extensions if interruptions occurred at a critical time.

Students who had installed *RStudio Desktop* on their personal machines (usually because of a prior course) mentioned that they would have preferred that the course use *RStudio Desktop* instead of *RStudio Server Professional* in the cloud. Obviously, that would not permit the file sharing conveniences of the addin discussed in this article, nor its live monitoring benefits.

Students were able to access our instance of *RStudio Server Professional* only while on campus, due to Bentley University's network security policies. To work on a project while away on a break, a student had to plan ahead, install *RStudio Desktop* on their personal machine, download their project from the server, and re-upload the resulting work when they returned to campus. This roundabout workflow could be overcome by the use of a file syncing service or distributed version control system, but Dr. Carter did not require students to learn that additional technology in his course, as we mentioned in Section 2.1.

## 5. INSTRUCTOR RESPONSES

### 5.1. Positive

Dr. Carter found it a pleasure to grade student assignments in *RMarkdown* using snippets for feedback, as in Section 3. This sped up the grading process and made it clean and clear. It was also very liberating to walk to and from class without carrying anything. All the course

materials and assignments were in the cloud, and available from the classroom computer upon arrival to class.

Archiving graded assignments is easy to do using *RStudio Server Professional*'s built-in file management features. Once assignments have been graded, and copies returned to students as in Sections 3.3 and 3.4, the instructor can simply move his or her own copies into a folder created for archiving. *RStudio Server Professional*'s file browser makes it easy to move many files to the same destination at once. Archiving is important in case students lose their copy or question a grade, and also makes it possible for the instructor to look back later to see how he or she graded an old assignment, should a student turn in late work that needs to be graded consistently with that of other students.

The use of this addin scales well with lengthy computations and/or large data sets, because the addin itself never loads data sets nor runs computations. Neither distributing nor collecting student files involves knitting the *RMarkdown* files themselves. When performing live monitoring of student work, only the HTML output files are read by the addin; the computations are not re-run. While grading student work, the instructor may choose to re-run a student's *RMarkdown* file, which will redo any computations in it, but the addin does not require doing so. Consequently, courses that use heavy computations or large data sets can use this addin without experiencing performance issues.

Writing an addin is a truly painless process in *RStudio*, and tutorials for learning to do so exist on the web (R Support 2017). If the addin developer works within *RStudio* itself, rebuilding and reinstalling the addin for testing purposes is a matter of a single keyboard shortcut. Furthermore, *RStudio* provides an API for complex addins, including those that do arbitrary manipulation of the currently open document, or display complex user interfaces, as the addin covered in this article does.

### 5.2. Negative

Our first attempts to share files using *RStudio Server Professional* were by creating a "handouts" projects that the instructor shared with all students in the course. The instructor was to place files in it, and students were to copy handouts from it. We abandoned this idea for two reasons.

First, sharing a folder among too many individuals resulted in an Access Control List (ACL) error from the underlying Linux system on the server. Second, copying a file from a shared folder into a non-shared folder was not at all easy. The native file browser in *RStudio* provides a "Copy" action that can operate only within a single folder and a "Move" operation that can work across folders but that removes the original file. The first workaround we attempted was to open the file and save it into a new location, but this had the consequence of unintentional file sharing described in Section 2.3; students attempting to edit the file after saving found that they were still editing a copy shared by other students, thus disrupting others' work.

The shortest workaround we found was as follows.

1. Open the shared project folder.
2. Open the file you wish to copy.
3. From the File menu, choose "Save As."
4. Browse to the desired destination folder and save.
5. Close the file.
6. From the File menu, choose "Open."
7. Choose the new version you just saved.

This was sufficiently cumbersome that we designed the addin to use the "submissions" folder paradigm discussed in Section 3.

## 6. FUTURE EXTENSIONS

The current version of the addin does not have a feature to export student grades to a spreadsheet or learning management system. But that can be accomplished with a small amount of *R* code, given in a technical appendix, Section 8. An interface for running the code could be integrated into the addin itself.

The current implementation of the live monitoring features discussed in Section 3 uses HTML output of `knitr` (Xie 2016), because it is organized into sections that can be queried with `xpath` (Lang and the CRAN Team 2016). However, *RMarkdown* files that use the Shiny runtime for interactive content (as discussed in Doi, Potter, Wong, Alcaraz, and Chi (2016)) don't produce HTML output, so they cannot be monitored in that way. If possible, a future version of the addin should address that limitation.

One nice side effect of using *RMarkdown* is that every time students knit the document (which happens often), it saves the results as both `.Rmd` and `.html` files. Thus regularly the students' work is being saved to disk in a permanent and easy-to-find location, and it would be possible to create tools that silently archive each version, resulting in a history of student work in class each day. Such archives could be mined by education researchers to investigate student mistakes en route to successes, student thought paths, and other data relevant to questions in mathematics, statistics, and computer science education.

## References


Allaire J, Cheng J, Xie Y, McPherson J, Chang W, Allen J, Wickham H, Atkins A, Hyndman R (2016). *rmarkdown: Dynamic Documents for R*. R package version 1.0, URL https://CRAN.R-project.org/package=rmarkdown.

Baumer B, Cetinkaya-Rundel M, Bray A, Loi L, Horton NJ (2014). "R Markdown: Integrating A Reproducible Analysis Tool into Introductory Statistics." *Technology Innovations in Statistics Education*, **8**(1), 1–29.

Carter N (2017). "SimpleCMS." URL https://github.com/nathancarter/simplecms.

Cetinkaya-Rundel M, Rundel CW (2017). "Infrastructure and tools for teaching computing throughout the statistical curriculum." *PeerJ Preprints*, **5**, e3181v1. ISSN 2167-9843. doi:10.7287/peerj.preprints.3181v1. URL https://doi.org/10.7287/peerj.preprints.3181v1.

Doi J, Potter G, Wong J, Alcaraz I, Chi P (2016). "Web Application Teaching Tools for Statistics Using R and Shiny." *Technology Innovations in Statistics Education*, **9**(1), 1–32.

Lang DT, the CRAN Team (2016). *XML: Tools for Parsing and Generating XML Within R and S-Plus*. R package version 3.98-1.4, URL https://CRAN.R-project.org/package=XML.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

R Support (2017). "RStudio Addins." URL https://rstudio.github.io/rstudioaddins.

RStudio Team (2015). *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA. URL http://www.rstudio.com/.

RStudio Team (2018a). "Download RStudio Server." URL http://www.rstudio.com/products/rstudio/download-server.

RStudio Team (2018b). *RStudio Cloud alpha: Do, share, teach and learn data science with R*. RStudio, Inc., Boston, MA. URL http://rstudio.cloud/.

Xie Y (2016). "knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.15.1." URL http://yihui.name/knitr.

## 7. TECHNICAL APPENDIX: SERVER SETUP

Installing *RStudio Server Professional* requires a Linux server on which to run it, someone comfortable with the Linux command line, and administrative (root) access. The server installation procedure requires only about 3–5 commands, all supplied by the *RStudio* company (RStudio Team 2018a). In addition, Linux user accounts must be created for each student and instructor who will log in to *RStudio Server Professional* through its web interface. Creating accounts and passwords for a list of, say, 30 users, and notifying those users of their accounts by email can be streamlined by the use of a scripting language such as `bash`, Perl, Python, or R. An example using `bash` is shown in Figure 5.

Random passwords can be generated in several ways on a Linux system. Dr. Oury used the `openssl` command to do so and used `bash` to automate the process, similar to the method shown in Figure 5.

*RStudio* makes *RStudio Server Professional* freely available to academic institutions. Use of this version of the server requires administrative access to the server to install the license

```
EMAILS="
dr_instructor@example.com
stew_dent@example.com
hardee_worker@example.com
ida_likanay@example.com
"

for ADDRESS in $EMAILS
do
    SPLIT=(${ADDRESS//@/ })
    USERNAME=${SPLIT[0]}
    PASSWORD=‘openssl rand -base64 10‘
    useradd -m -d /home/$USERNAME $USERNAME
    echo $USERNAME:$PASSWORD | chpasswd
    echo "Send an email to $ADDRESS and say that:"
    echo "    Their username is $USERNAME."
    echo "    Their initial password is $PASSWORD."
done
```

Figure 5: A `bash` script for creating user accounts and passwords on a Linux system. This could be extended with automated emails in place of the `echo` commands.

(provided by *RStudio*). The license is installed with a single command, which they also provide.

It is essential that the Linux host system run a time server process (such as `ntpd`) and that this process is accurately synchronized with Internet time service. Failure to do so may lead to the server's thinking that the license has expired, and not permitting users to log in. When we discovered the importance of this synchronization the hard way, the support staff at *RStudio* were very helpful in debugging problems that arose from local time not being synchronized with Internet time.

Naturally, it is important to take care with data privacy on any server. One would not want students to have access to one another's work or grades. Such access (or lack thereof) is contingent upon two things. First, the setup of the accounts as described above does not, by default, give students access to read one another's files. As long as a similar setup is followed, the usual read and write limitations among Linux home folders apply.

Second, students must be told that if they share a project with other students, those students can read the contents of the project's folder. Instructors should warn students not to do this with folders that contain sensitive material such as grades. Instructors who wish to verify that students are complying with this guidance can use the *RStudio Server Professional* interface to inspect, for each student's "submissions" project, the complete list of other users with whom that project has been shared.

```
#
# This function extracts grades from .Rmd files.
#
export.grades <- function ( folder = '.', students, assignments, re ) {
  grades <- matrix( nrow=length( students ), ncol=length( assignments ) )
  rownames( grades ) <- students
  colnames( grades ) <- assignments
  for ( assignment in assignments ) {
    for ( student in students ) {
      innerfolder <- file.path( folder, assignment )
      file <- list.files( innerfolder, paste0( student, '--*-graded.Rmd' ) )[1]
      if ( is.null( file ) ) next()
      content <- paste( readLines( file.path( innerfolder, file ) ), collapse='\n' )
      match <- regmatches( content, regexec( re, content ) )[[1]]
      score <- as.numeric( match[2] ) / as.numeric( match[3] ) * 100
      grades[student,assignment] <- score
    }
  }
  write.csv( grades )
}
#
# Example use:
#
export.grades( students = c( 'jstudent1', 'jstudent2', 'jstudent3' ),
               assignments = c( 'assignment1', 'assignment2', 'assignment3' ),
               re = '<b>Grade:</b>\\s*([0-9.]+)/([0-9.]+)' )
```

Figure 6: Example *R* script that could be used to download scores from all graded *RMarkdown* files into a CSV file for use in a gradebook or Learning Management System.

## 8. TECHNICAL APPENDIX: DOWNLOADING SCORES

This appendix provides *R* code for extracting student grades from graded *RMarkdown* files. The code appears in Figure 6, and would would be run once, at the end of a semester, to loop through all users and all assignments, extracting grades embedded in the corresponding *RMarkdown* files.

The result is a comma-separated values (CSV) file that can be imported into just about any type of gradebook (including any spreadsheet application or a Learning Management System such as Moodle or Blackboard). Such an import is contingent upon the instructor's having used the same usernames for student accounts on *RStudio Server Professional* as in the Learning Management System. This can be accomplished as discussed in Section 7.

The code in Figure 6 also assumes the instructor is using the snippets in Figure 2. Different snippets would require a different regular expression in the final line of code in the figure.