swirl

The Johns Hopkins Data Science Lab

July 19, 2016

What is swirl? Part 1

- Swirl is an R package that turns the R console into an interactive learning evironment.
- Students are guided through R programming exercises where they can answer questions in the R console.
- There is no separation between where a student learns to use R, and where they go on to use R for thier own creative purposes.

If you've never used swirl before you shoud try swirl out for yourself:

```
install.packages("swirl")
library(swirl)
swirl()
```

What is swirl? Part 2

- Anyone can create and distribute their own swirl course!
- Creating a swirl course can allow you scale R and data science education within your company or school.
- If you're interested in getting involved in the open source education community you can release your swirl course online and Team swirl will promote it for you!
- There's a package called swirlify which is designed to help you write a swirl course. We'll look into swirlify later.

What is a swirl course?

- A swirl course is a directory that contains all of the files, folders, and lessons associated with the course you are developing.
- Lessons contain most of the content that students interact with, and the course sequentially organizes these lessons.
- A course should conceptually encapsulate a broad concept that you want to teach. For example: "Plotting in R" or "Statistics for Engineers" are broad enough topics that they could be broken down further (into lessons which we'll talk about next).
- When a student start swirl, they will prompted to choose from a list of courses, and then they can choose a lesson within the course they selected.
- Once the student selects a lesson, swirl will start asking the student questions in the R console.

What is a lesson?

- A lesson is a directory that contains all of the files required to execute one unit of instruction inside of swirl.
- For example a "Plotting in R" course might contain the lessons: "Plotting with Base Graphics", "Plotting with Lattice", and "Plotting with ggplot2."
- Every lesson must contain one `lesson.yaml` file which structures the text that the student will see inside the R console while they are using swirl.
- The `lesson.yaml` file contains a sequence of questions that students will be prompted with.

Writing swirl Courses: Setting up the App

First install swirlify:

```
install.packages("swirlify")
library(swirlify)
```

Then set your working directory to wherever you want to create your course and start the Shiny lesson authoring app:

```
setwd(file.path("~", "Desktop"))
swirlify("Lesson 1", "My First Course")
```

Writing swirl Courses: Using the App

For a demo about how to use the Shiny authoring app see the following video: `https://youtu.be/UPL_W-Umgjs`

Writing swirl Courses: Using the R Console

- Alternatively you can use the R console and a text editor to write swirl lessons.
- We highly recommend using RStudio for writing swirl lessons since RStudio provides this editor and console setup by default.

To start a new lesson from the R console set your working directory to where you want to create the course, and then use the `new_lesson()` function to create the course:

```
setwd("/Users/sean/")
new_lesson("My First Lesson", "My First Course")
```

Writing Lessons: Part 1

The `new_lesson()` function will create a file and folder structure like this in your working directory:

```
My_New_Course
- My_First_Lesson
    - lesson.yaml
    - initLesson.R
    - dependson.txt
    - customTests.R
```

Writing Lessons: Part 2

To review, the `new_lesson()` function did the following:

1. A new directory was created in `/Users/sean/` called `My_New_Course`.
2. A new directory was created in `/Users/sean/My_New_Course` called `My_First_Lesson`.

Writing Lessons: Part 2.5

- Several files were created inside of `/Users/sean/My_New_Course/My_First_Lesson`:
  - `lesson.yaml` is where you will write all of the questions for this lesson. (Example)
  - `initLesson.R` is an R script that is run before the lesson starts which is usually used to load data or set up environmental variables. (Example)
  - `dependson.txt` is the list of R packages your lesson will require. swirl will install these packages if the user doesn't already have them installed. (Example)
  - `customTests.R` is where you can write your own tests for student's answers. (Example)

Writing Lessons: Part 3

If everything is set up correctly then `new_lesson()` should have opened up the new `lesson.yaml` file in a text editor. You can now start adding questions to the `lesson.yaml` by using functions that start with `wq_` (write question).

Writing Lessons: Types of Questions

Lessons are sequences of questions that have the following general structure:

```
- Class: [type of question]
  Key1: [value1]
  Key2: [value2]

- Class: [type of question]
  Key1: [value1]
  Key2: [value2]
...
```

The example above shows the high-level structure for two questions.

Writing Lessons: Types of Questions

- Each question is demarcated with a hyphen.
- Every question starts with a `Class` that specifies that question's behavior inside of swirl.
- What follows the class is a set of key-value pairs that will be used to render the question when a student is using swirl.

Writing Lessons: The Meta Question

The first question in every `lesson.yaml` is always the meta question which contains general information about the course. Below is an example of the meta question:

```
- Class: meta
  Course: My Course
  Lesson: My Lesson
  Author: Dr. Jennifer Bryan
  Type: Standard
  Organization: The University of British Columbia
  Version: 2.5
```

The meta question will not be displayed to a student. The only fields you should modify are `Author` and `Organization` fields.

Writing Lessons: Message Questions

Message questions display a string of text in the R console for the student to read. Once the student presses enter, swirl will move on to the next question.

Add a message question using `wq_message()`.

Here's an example message question:

```
- Class: text
  Output: Welcome to my first swirl course!
```

The student will see the following in the R console:

```
| Welcome to my first swirl course!

...
```

Writing Lessons: Command Questions

Command questions prompt the student to type an expression into the R console.

- The `CorrectAnswer` is entered into the console if the student uses the `skip()` function.
- The `Hint` is displayed to the student if they don't get the question right.
- The `AnswerTests` determine whether or not the student answered the question correctly. See the answer testing section for more information.

Add a message question using `wq_command()`.

Writing Lessons: Command Questions

Here's an example command question:

```
- Class: cmd_question
  Output: Add 2 and 2 together using the addition operator.
  CorrectAnswer: 2 + 2
  AnswerTests: omnitest(correctExpr='2 + 2')
  Hint: Just type 2 + 2.
```

The student will see the following in the R console:

```
| Add 2 and 2 together using the addition operator.

>
```

Multiple Choice Questions

Multiple choice questions present a selection of options to the student. These options are presented in a different order every time the question is seen.

- The `AnswerChoices` should be a semicolon separated string of choices that the student will have to choose from.

Add a message question using `wq_multiple()`.

Here's an example multiple choice question:

```
- Class: mult_question
  Output: What is the capital of Canada?
  AnswerChoices: Toronto;Montreal;Ottawa;Vancouver
  CorrectAnswer: Ottawa
  AnswerTests: omnitest(correctVal='Ottawa')
  Hint: This city contains the Rideau Canal.
```

Multiple Choice Questions

The student will see the following in the R console:

```
| What is the capital of Canada?

1: Toronto
2: Montreal
3: Ottawa
4: Vancouver
```

Other Question Types

For complete documentation about writing swirl courses and lessons see the swirlify website: `http://swirlstats.com/swirlify/`

Organizing Lessons: Part 1

Let's revisit the general structure of a swirl course. This is the structure of a course with two lessons:

```
My_New_Course
- My_First_Lesson
   - lesson.yaml
   - initLesson.R
   - dependson.txt
   - customTests.R
- My_Second_Lesson
   - lesson.yaml
   - initLesson.R
   - dependson.txt
   - customTests.R
```

Organizing Lessons: Part 2

- By default each folder in `My_New_Course` will be displayed to the student as a lesson they can select.
- If you want to explicitly specify the order in which lessons are displayed you will need to add a `MANIFEST` file to your course.
- You can do this with the `add_to_manifest()` function, which will add the lesson you are currently working on to the `MANIFEST`. You can also edit the `MANIFEST` yourself in a text editor.

Organizing Lessons: Part 3

The (abridged) `MANIFEST` file below belongs to Team swirl's R Programming course:

```
Basic_Building_Blocks
Workspace_and_Files
Sequences_of_Numbers
Vectors
Missing_Values
Subsetting_Vectors
Matrices_and_Data_Frames
```

Sharing Your Course

Swirlify makes sharing a swirl course easy. We recommend three different methods for sharing a swirl course.

Sharing Your Course as a File

We've developed the `.swc` file type so that you can share your course as a single file. Creating an `.swc` file for your course is easy:

1. Set any lesson in the course you want to share as the current lesson using `set_lesson()`.
2. Create an `.swc` file using the `pack_course()` function. Your `.swc` file will appear in the same directory as the directory that contains the course folder. You also have the option to export the `.swc` file to another directory by specifying the `export_path` argument.

Sharing Your Course as a File

- You can now share your `.swc` file like you would any other file (through email, file sharing services, etc).
- Students can install your course from the `.swc` file by downloading the file and then using the `install_course()` function in swirl, which will prompt them to interactively select the file they downloaded.

Sharing Your Course on GitHub

- We highly encourage you to develop your course on GitHub so that we can better support you if you have questions or need assistance while developing your course.
- Developing your course on GitHub provides the added benefit that your course will be instantly ready to distribute.
- Students can install your course from swirl using the `install_course_github()` function.
- Make sure that your course directory is the root folder of your git repository. For examples of courses that have been shared on GitHub you can browse some of the courses on the Swirl Course Network.

Sharing Your Course on The Swirl Course Network

The goal of the Swirl Course Network is to list and organize all publicly available swirl courses. Visit the homepage of the SCN for more information.

After adding your course to the SCN students will be able to install your course using `install_course("[Name of Your Course]")` in swirl.

Sharing Your Course on The Swirl Course Network

In order to add your course to the SCN:

1. Create an `.swc` file for your course.
2. Fork `https://github.com/swirldev/scn` on GitHub.
3. Add the `.swc` file to your fork.
4. Add an Rmd file to your fork like this one. You can include a description of your course, authors, a course website, and how to install your course.
5. Run `rmarkdown::render_site()` when your current directory is set to your fork.
6. Add, commit, and push your changes to GitHub, then send us a Pull Request.

More Help & Resources

- The swirl Website
- The swirlify Documentation
- The swirl Course Network